

La Biblia de Visual Basic .NET

Luis Dueñas

Acerca del Autor

Luis Dueñas Huaroto es un especialista en desarrollo de Software con más de 20 años de experiencia, tiene estudios superiores en Economía, Computación y Maestría en Ingeniería de Sistemas con especialización en Ingeniería de Software.

El autor tiene las certificaciones: Microsoft Certified Professional (MCP), Microsoft Office User Specialist (MOUS), Microsoft Certified Solution Developer .NET (MCSD), Microsoft Certified Technology Specialist (MCTS) y Microsoft Certified Trainer (MCT), además conoce las versiones de Visual Basic 3,4,5,6 y .NET desde sus inicios habiendo trabajado con todas las versiones de .NET Framework y Visual Studio.

Actualmente se desempeña como consultor Senior y Team Leader en una importante Consultora de Sistemas, además de laborar como docente hace más de 10 años en ISILTECH que es el centro de especialización del Instituto San Ignacio de Loyola.

También realiza capacitaciones de desarrollo sobre la plataforma .NET a Empresas, entre las cuales tenemos: Provías Nacional del Ministerio de Transportes y Comunicaciones, Policía Nacional del Perú, Clínica Ricardo Palma, Banco INTERBANK, BCTS Consulting, entre otros.

Para cualquier contacto con el llamar al teléfono 997500549 o enviar un mail a Luis.duenas@bctsconsulting.com

Dedicatoria

Este libro va dedicado a las personas que cada día me obligan a investigar más y a aprender nuevos temas, en especial a mis alumnos y mis compañeros de trabajo.

Sobre todo va dedicado para mis alumnos más antiguos como: Raúl Antón, Marcelino Cabanaconza, Luis y Nano Agape, Juan Carlos Moreno y Edward Schwarz.

Indice

Introducción

Capítulo 1: .NET Framework 4, Visual Studio 2010 y Visual Basic .NET 2010

1. Introducción a Microsoft .NET Framework 4
 - 1.1. Componentes de Microsoft .NET Framework
 - 1.2. Biblioteca de Clases de .NET Framework (BCL)
 - 1.3. Motor de Ejecución de Lenguaje Común (CLR)
 - 1.4. Novedades del .NET Framework 4
2. Visual Studio .NET 2010
 - 2.1. Descripción del Entorno Integrado de Desarrollo (IDE)
 - 2.2. Novedades de Visual Studio 2010
3. Visual Basic .NET 2010
 - 3.1. Estructuras de Control de Flujo
 - 3.2. Tipos de Datos en Visual Basic
 - 3.3. Operadores y Expresiones en Visual Basic
 - 3.4. Variables en Visual Basic
 - 3.5. Novedades de Visual Basic 2010

Capítulo 2: Programando con Visual Basic .NET 2010

1. Trabajar con el Entorno de Windows y el Sistema
 - 1.1. Obtener Información del Sistema
 - 1.2. Obtener Directorios Especiales de Windows
 - 1.3. Acceder al Registro de Windows
 - 1.4. Trabajar con Procesos o Tareas del Sistema
2. Manejo de Entrada y Salida
 - 2.1. Manejar Unidades, Directorios, Rutas y Archivos
 - 2.2. Leer y Escribir en un Archivo
 - 2.3. Dividir y Unir Archivos
 - 2.4. Comprimir y Descomprimir un Archivo
3. Programación Orientada a Objetos (POO)
 - 3.1. Introducción a la POO y MDD
 - 3.2. Creando Bibliotecas de Clases
 - 3.3. Creando una Aplicación que use las Bibliotecas de Clases
4. Programación Asíncrona y Paralelismo

- 4.1. Threads
- 4.2. Eventos Asíncronos
- 4.3. Delegados Callbacks
- 4.4. Programación Paralela
- 5. Criptografía o Cifrado de Datos
 - 5.1. Cifrado Simétrico por Caracter
 - 5.2. Cifrado Simétrico por Bloques
 - 5.3. Valores Hash Criptográficos

Capítulo 3: Accediendo a Datos con ADO .NET

- 1. Trabajando en Forma Conectada
 - 1.1. Introducción al Acceso a Datos con ADO .NET
 - 1.2. Conectarse a un Origen de Datos
 - 1.3. Ejecutando Comandos de Selección
 - 1.4. Realizando un Mantenimiento Conectado
- 2. Trabajando en Forma Desconectada con DataSet
 - 2.1. Trabajando con Tablas y Relaciones
 - 2.2. Trabajando con Vistas
 - 2.3. Mantenimiento Desconectado con DataSet
- 3. Trabajando en Forma Desconectada con Listas de Objetos
 - 3.1. Llenando una Lista de Objetos
 - 3.2. Filtrando Datos en una Lista de Objetos
 - 3.3. Ordenando y Buscando datos en una Lista de Objetos
 - 3.4. Mantenimiento con Objetos
- 4. LINQ
 - 4.1. LINQ a DataSet
 - 4.2. LINQ a SQL
 - 4.3. LINQ a Entidades

Capítulo 4: Desarrollando Aplicaciones Windows Forms

- 1. Trabajando con el Formulario y los Controles Windows
 - 1.1. Trabajando con el Formulario Windows
 - 1.2. Usando Controles Básicos
 - 1.3. Usando Controles de Listas
 - 1.4. Usando Controles de Vistas
- 2. Creando Formularios MDIs, Menús, Diálogos y Barras

- 2.1. Creando Formularios MDIs
- 2.2. Creando Menús
- 2.3. Usando Dialogos de Windows
- 2.4. Agregando Barras
- 3. Usando el Control DataGridView
 - 3.1. Personalizando Columnas en el DataGridView
 - 3.2. Mostrando una Imagen en una Columna
 - 3.3. Personalizando las Cabeceras de las Columnas
 - 3.4. Graficando en el DataGridView
 - 3.5. Paginando en un DataGridView
- 4. Creando una Biblioteca de Controles Windows
 - 4.1. Creando Controles Extendidos
 - 4.2. Creando Controles de Usuario
 - 4.3. Creando Controles Personalizados
- 5. Creando Reportes e Impresiones en Windows Forms
 - 5.1. Usando PrintDocument
 - 5.2. Informes de Microsoft
 - 5.3. Trabajando con Word
 - 5.4. Trabajando con Excel
 - 5.5. Usando el Control Chart

Capítulo 5: Desarrollando Aplicaciones Web con ASP .NET

- 1. Trabajando con el Formulario y los Controles Web
 - 1.1. Introducción a ASP .NET
 - 1.2. Creando un Simple Sitio Web
 - 1.3. Usando Controles Web Intrínsecos
 - 1.4. Usando Controles Web de Imágenes
 - 1.5. Usando el Control FileUpload y Controles de Validación
- 2. Mejorando el Diseño y Navegabilidad del Sitio Web
 - 2.1. Creando y usando Hojas de Estilos en Cascada
 - 2.2. Paginas Principales y Controles de Navegación
 - 2.3. Usando Controles de Vistas MultiView y Views
- 3. Usando el Control GridView
 - 3.1. Personalizando Columnas en el GridView
 - 3.2. Paginando en el GridView
 - 3.3. Ordenando en el GridView

- 3.4. Mantenimiento de Datos en el GridView
- 4. Creando Plantillas en Controles Enlazados a Datos
 - 4.1. Trabajando con el Control Repeater
 - 4.2. Trabajando con el Control DataList
 - 4.3. Creando Plantillas Jerárquicas

Capítulo 6: Desarrollando Aplicaciones con WPF

- 1. Creando Aplicaciones Básicas con WPF
 - 1.1. Introducción a WPF
 - 1.2. Trabajando con Ventanas
 - 1.3. Trabajando con Páginas
 - 1.4. Usando Cuadros de Diálogo
- 2. Usando Enlace de Datos
 - 2.1. Introducción al Enlace de Datos
 - 2.2. Usando Enlace de Datos
 - 2.3. Usando Conversión de Datos
 - 2.4. Usando Plantillas de Datos
 - 2.5. Usando el Control DataGrid
- 3. Manejando Documentos
 - 3.1. Introducción al Manejo de Documentos
 - 3.2. Trabajando con Documentos Fijos
 - 3.3. Creando Anotaciones en Documentos
 - 3.4. Trabajando con Documentos Dinámicos
- 4. Manejando Multimedia
 - 4.1. Introducción al Manejo de Multimedia
 - 4.2. Implementando Voz
 - 4.3. Trabajando con Audio y Video

Índice de Ejemplos del Libro

Enlaces de Referencia

Introducción

Tratar de escribir una obra completa sobre desarrollo de aplicaciones usando Visual Basic en .NET Framework es muy ambicioso, por lo cual se ha dividido en 2 partes, éste libro es la primera parte de este esfuerzo y contiene 6 capítulos.

El primero es conceptual y brinda información sobre .NET Framework, Visual Studio y Visual Basic. El segundo capítulo nos enseña como programar en Visual Basic usando algunas clases de .NET Framework como el entorno, entrada y salida, programación orientada a objetos, programación asíncrona y seguridad de datos.

El tercer capítulo trata el acceso a datos usando ADO .NET y se divide en programación conectada, programación desconectada con DataSet y también con Lista de Objetos, al final se revisa LINQ.

El cuarto capítulo enseña el desarrollo de aplicaciones para Windows usando WinForms y contiene temas como el formulario y los controles, creación de MDIs y menús, uso de diálogos y barras de herramientas, también trata detalladamente el control DataGridView y cómo crear una librería de controles para Windows y finalmente diferentes formas de crear reportes e impresiones en Windows.

En el quinto capítulo se aprenderá a desarrollar aplicaciones Web usando ASP .NET Web Forms, iniciando con el formulario y los controles Web, mejorando el diseño usando Hojas de Estilo en Cascada y mejorando la navegabilidad mediante controles de navegación y paginas principales, también usando controles MultiView y Views para usar fichas o tabs.

En el último capítulo de este libro veremos Windows Presentations Foundation (WPF) para crear aplicaciones Windows y del Explorador, trabajaremos con enlace de datos, manejo de documentos y finalmente implementaremos características Multimedia.

Como parte del segundo libro estaría quedando ASP .NET MVC, Silverlight, Windows Communication Foundation (WCF), Windows Workflow Foundation (WWF), Desarrollo con MS Office en .NET y Desarrollo con MS SharePoint en .NET.

Capítulo 1: .NET Framework 4, Visual Studio 2010 y Visual Basic .NET 2010

En este primer capítulo del libro daremos una revisión rápida de la arquitectura y componentes de la plataforma de desarrollo Microsoft .NET Framework 4, las características de la Herramienta de Desarrollo Visual Studio 2010 y del Lenguaje Visual Basic .NET 2010.

En los 3 veremos las características principales así como las novedades de esta versión, tanto de plataforma, del entorno de desarrollo de Visual Studio así como del Lenguaje Visual Basic.

1. Introducción a Microsoft .NET Framework 4

Microsoft .NET Framework 4.0 es la última plataforma de desarrollo Microsoft que permite crear todo tipo de aplicaciones desde aplicaciones de consola para programas en lotes o batch, aplicaciones Windows para Cliente/Servidor, aplicaciones Web para Internet o Intranet, Servicios Web para interactuar con otras plataformas, Servicios Windows para ejecutar tareas en segundo plano, y otros tipos de aplicaciones.

También nos permite crear Librerías o Bibliotecas de clases reusables, ya sea a nivel de código, de controles, de servicios, de flujos, etc., las cuales pueden utilizarse en diferentes tipos de aplicaciones.

A partir de la versión 3.0 se incluyó en el .NET Framework el poder crear aplicaciones de mejor presentación visual con WPF (Windows Presentations Foundation), también se unificó las aplicaciones distribuidas que antes se implementaban mediante Web Services, COM+ y NET Remoting en un solo conjunto de componentes llamado WCF (Windows Communication Foundation) y también se dio la posibilidad de crear aplicaciones de flujo de trabajo o colaboración con WWF (Windows Workflow Foundation).

Para realizar el desarrollo de aplicaciones o librerías en .NET el desarrollador puede elegir entre muchos Lenguajes .NET algunos de Microsoft como C#, J#, Visual Basic, Visual C++ y otros de diferentes proveedores: como ADA, APL, ASML, BETA, BF, C, Clarion#, COBOL, Cobra, CULE, E#, Eiffel, Flash, Forth, Fortran, G#, Jaskel, JavaScript, LISP, LOGO, Mercury, Modula 2, Oberon, Pascal, Perl, PHP, Prolog, Python, RPG, Ruby, Scala, Scheme, Smaltalk, etc.

Esta última característica de los lenguajes .NET es una diferencia fundamental con respecto a JAVA en donde el desarrollador solo programa usando un lenguaje (Java), en cambio con .NET Framework, podemos elegir entre muchos lenguajes, esta elección se hará de acuerdo a la experiencia que tenga la persona, en este libro vamos a usar el lenguaje Basic, es decir Visual Basic .NET.

Visual Basic .NET es recomendable para las personas que se inician en el desarrollo de software y no han visto ningún lenguaje, pero también para aquellos desarrolladores con experiencia que vienen de Visual Basic 3, 4, 5

o 6; de Foxpro, Visual Foxpro o Power Builder; ya que tiene características similares a estos, como el Modelo Conducido por Eventos (Event Driven Model o MDE), además de ser Orientado a Objetos (OOP), entre otras características modernas.

1.1. Componentes de Microsoft .NET Framework

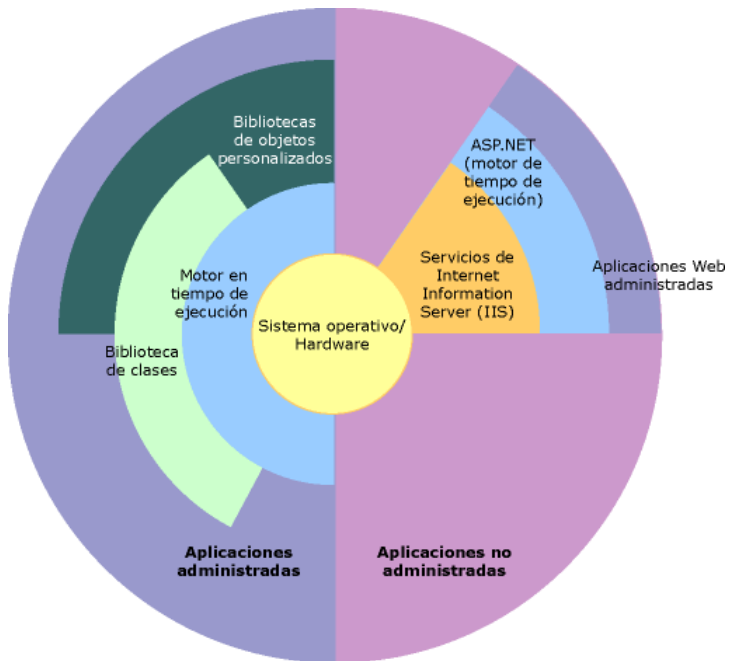
Para desarrollar necesitamos una plataforma o entorno de desarrollo al cual se le conoce como Framework de Desarrollo, el .NET Framework es la plataforma que necesitamos para realizar cualquier tipo de desarrollo y se compone de 2 elementos principales:

- Las Bibliotecas de Clase Base (BCL): conjunto de librerías de contienen clases que permiten crear la aplicación.
- El Motor de Ejecución de Lenguaje Común (CLR): conjunto de programas que permiten compilar (a código nativo) y ejecutar la aplicación entre otras tareas que se detallan más adelante.

Cuando desarrollemos aplicaciones con .NET Framework, debemos tener en cuenta el tipo de Aplicación:

- Aplicación Administrada: manejada por el CLR o motor de ejecución de .NET, por ejemplo las aplicaciones de Consola, Windows, WPF, etc.
- Aplicación No Administrada: manejada por otro motor, por ejemplo las Aplicaciones Web son manejadas por el motor de ejecución de ASP NET y por el IIS, las aplicaciones COM+ son manejadas por el DLLHost, etc.

Gráfico 1.1: .NET Framework en contexto



1.2. Biblioteca de Clases de .NET Framework (BCL)

La BCL es una colección de más de 5000 tipos orientados a objetos distribuidos en más de 80000 miembros, los tipos se clasifican en:

- Tipos por Valor: Se copia el valor a la variable
 - Tipos Simples del .Net Framework: char, boolean, byte, int, etc.
 - Enumeraciones.
 - Estructuras.
- Tipos por referencia
 - Tipos complejos de .NET Framework: String, StringBuilder, DataSet, FileStream, StreamReader, StreamWriter, etc.
 - Clases.
 - Enumeraciones.

Cada tipo contiene elementos o miembros que pueden ser:

- Constructores y Destructores
- Propiedades, Metodos y Eventos
- Campos (variables publicas)
- Delegados (punteros a funciones)
- Enumeraciones
- Operadores

A su vez los tipos están organizados físicamente en Librerías (Assemblies) y lógicamente en Espacios de Nombres (Namespace). Por ejemplo los tipos simples se encuentran en la librería System.Core.dll, los tipos principales en System.dll, el formulario y los controles Windows Forms en System.Windows.Forms.dll, etc.

La BCL tiene más de 255 espacios de nombres principales que agrupan a los tipos por categorías de uso, la mayoría inician con System y unos cuantos con Microsoft, entre los principales tenemos:

Espacio nombres	Descripción
Microsoft.VisualBasic	Contienen clases que admiten la compilación y generación de código mediante el lenguaje Visual Basic. Los espacios de nombres secundarios contienen tipos que proporcionan servicios al compilador de Visual Basic y tipos que incluyen compatibilidad con el modelo de aplicaciones de Visual Basic, el espacio de nombres My, expresiones lambda y conversión de código.
Microsoft.Win32	Proporcionan tipos que administran eventos provocados por el sistema operativo, que manipulan el Registro del sistema, y que representan identificadores de archivo y del sistema operativo.
System	Contiene clases que le permiten hacer coincidir con los URI con plantillas URI y grupos de plantillas URI.
System.Activities	Contienen todas las clases necesarias para crear y trabajar con actividades en Windows Workflow Foundation.
System.Collections	Contienen tipos que definen varios objetos de colección estándar, especializados y genéricos.
System.Configuration	Contienen tipos para administrar datos de

	configuración, como datos de archivos de configuración de equipos o aplicaciones. Los espacios de nombres secundarios contienen tipos que se emplean para configurar un ensamblado, escribir instaladores personalizados de componentes, y admitir un modelo conectable para agregar o quitar funcionalidad tanto de aplicaciones cliente como de aplicaciones servidor.
System.Data	Contienen clases para tener acceso a datos y administrarlos desde distintos orígenes. El espacio de nombres de nivel superior y una serie de espacios de nombres secundarios forman conjuntamente la arquitectura ADO.NET y los proveedores de datos de ADO.NET. Por ejemplo, hay disponibles proveedores para SQL Server, Oracle, ODBC y OleDb. Otros espacios de nombres secundarios contienen clases empleadas por Entity Data Model (EDM) de ADO.NET y por Servicios de datos de WCF.
System.Diagnostics	Contienen tipos que le permiten interactuar con procesos del sistema, registros de eventos y contadores de rendimiento. Los espacios de nombres secundarios contienen tipos para interactuar con herramientas de análisis del código, admitir contratos, ampliar la compatibilidad en tiempo de diseño con la supervisión e instrumentación de aplicaciones, registrar datos de eventos mediante el subsistema Seguimiento de

	eventos para Windows (ETW), leer registros de eventos y escribir en ellos y recopilar datos de rendimiento, y para leer y escribir información de símbolos de depuración.
System.Drawing	Contiene tipos que admiten funcionalidad básica de gráficos GDI+. Los espacios de nombres secundarios admiten funcionalidad avanzada de gráficos bidimensionales y vectoriales, funcionalidad avanzada de procesamiento de imágenes, y servicios tipográficos y relacionados con la impresión. Un espacio de nombres secundario también contiene tipos que extienden la lógica y el dibujo de la interfaz de usuario en tiempo de diseño.
System.Globalization	Contiene clases que definen información relativa a la referencia cultural, incluido el idioma, el país o región, los calendarios utilizados, los modelos de formato de fecha, divisa y números, y el criterio de ordenación de las cadenas. Estas clases son útiles para escribir aplicaciones internacionalizadas.
System.IO	Contienen tipos que admiten entrada y salida, incluida la posibilidad de leer y escribir datos en flujos de forma sincrónica o asincrónica, comprimir datos en flujos, crear y usar almacenes aislados, asignar archivos al espacio de direcciones lógicas de una aplicación, almacenar varios objetos de datos en un único contenedor, comunicarse mediante

	canalizaciones anónimas o con nombre, implementar el registro personalizado, y administrar el flujo de datos hacia y desde puertos serie.
System.Linq	Contienen tipos que admiten consultas que emplean Language-Integrated Query (LINQ). Esto incluye tipos que representan consultas como objetos en árboles de expresión.
System.Media	Contiene clases para reproducir archivos de sonido y obtener acceso a los sonidos que proporciona el sistema.
System.Messaging	Contienen tipos que le permiten conectar con colas de mensajes en la red, así como supervisarlas y administrarlas, y enviar, recibir o inspeccionar mensajes. Un espacio de nombres secundario contiene clases que se pueden usar para ampliar la compatibilidad en tiempo de diseño de clases de mensajería.
System.Net	Contienen clases que proporcionan una interfaz de programación sencilla para diversos protocolos de red, tienen acceso mediante programación y actualizan valores de configuración para los espacios de nombres System.Net, definen directivas de caché para recursos web, redactan y envían correo electrónico, representan encabezados de Extensiones multipropósito de correo Internet (MIME), tienen acceso a datos de tráfico de red y a

	información de direcciones de red, y tienen acceso a funcionalidad de red punto a punto. Otros espacios de nombres secundarios proporcionan una implementación administrada de la interfaz Windows Sockets (Winsock) y brindan acceso a secuencias de red para proteger las comunicaciones entre hosts.
System.Reflection	Contienen tipos que proporcionan una vista administrada de los tipos, métodos y campos cargados, y que pueden crear e invocar tipos de forma dinámica. Un espacio de nombres secundario contiene tipos que permiten a un compilador u otra herramienta emitir metadatos y el lenguaje intermedio de Microsoft (MSIL).
System.Runtime	Contienen tipos que admiten la interacción de una aplicación con CLR, y tipos que habilitan características como almacenamiento en caché de datos de la aplicación, control avanzado de excepciones, activación de aplicaciones dentro de dominios de aplicación, interoperabilidad COM, aplicaciones distribuidas, serialización y deserialización, y control de versiones, etc.
System.Security	Contienen clases que representan el sistema de seguridad y los permisos de .NET Framework. Los espacios de nombres secundarios proporcionan tipos que controlan el acceso a objetos protegibles y los auditan, permiten autenticación, ofrecen

	servicios criptográficos, controlan el acceso a operaciones y recursos según una directiva, y admiten la administración de derechos del contenido creado por la aplicación.
System.ServiceModel	Contienen los tipos necesarios para compilar aplicaciones cliente y de servicio de Windows Communication Foundation (WCF).
System.Service Process	Contienen tipos que le permiten implementar, instalar y controlar aplicaciones de servicios de Windows, y extienden la compatibilidad en tiempo de diseño con aplicaciones de servicios de Windows.
System.Speech	Contienen tipos que admiten reconocimiento de voz.
System.Text	Contienen tipos para la codificación de caracteres y la manipulación de cadenas. Un espacio de nombres secundario le permite procesar texto usando expresiones regulares.
System.Threading	Contienen tipos que habilitan la programación multiproceso. Un espacio de nombres secundario proporciona tipos que simplifican el trabajo de escribir código simultáneo y asíncrono.
System.Web	Contienen tipos que habilitan la comunicación entre el explorador y el servidor. Los espacios de nombres secundarios incluyen tipos que admiten autenticación de formularios de ASP.NET, servicios

	de aplicación, almacenamiento en caché de datos en el servidor, configuración de aplicaciones ASP.NET, datos dinámicos, controladores HTTP, serialización de JSON, incorporación de funcionalidad AJAX a ASP.NET, seguridad de ASP.NET y servicios Web.
System.Windows	Contienen tipos usados en aplicaciones de Windows Presentation Foundation (WPF), incluidos clientes de animación, controles de interfaz de usuario, enlace de datos y conversión de tipos. System.Windows.Forms y sus espacios de nombres secundarios se emplean para desarrollar aplicaciones de Windows Forms.
System.Workflow	Contienen tipos usados para desarrollar aplicaciones que emplean Windows Workflow Foundation. Estos tipos ofrecen compatibilidad en tiempo de diseño y en tiempo de ejecución con reglas y actividades para configurar, controlar, hospedar y depurar el motor en tiempo de ejecución de flujos de trabajo.
System.Xaml	Contienen tipos que admiten el análisis y el procesamiento del lenguaje XAML.
System.Xml	Contienen tipos para el procesamiento de XML. Los espacios de nombres secundarios admiten la serialización de documentos o secuencias XML, esquemas XSD, XQuery 1.0 y XPath 2.0, y LINQ a XML.

1.3. Motor de Ejecución de Lenguaje Común (CLR)

El CLR administra la memoria, la ejecución de subprocesos, la ejecución de código la comprobación de la seguridad, compilación a código nativo, entre otros servicios del sistema, es decir todos los servicios que se usan al ejecutar un programa.

El desarrollador crea una aplicación manejada o administrada en cualquiera de los Lenguajes .NET de Alto Nivel mencionados como C# o Visual Basic, luego lo compila con el compilador del lenguaje: CSC.exe para C#, VBC.exe para Visual Basic creándose un ensamblado (Assembly) que se encuentra en Lenguaje Intermedio de Microsoft (MSIL).

Al querer ejecutar este Assembly el CLR invoca al Compilador Just In Time que convierte el código MSIL en código nativo el cual se encuentra en Lenguaje de Bajo Nivel y es este ultimo código que el CLR ejecuta para comunicarse con el sistema (RAM, CPU, Video, Sistema de Archivos, Bases de Datos, etc.)

Cuando se inicia la aplicación el CLR se encarga de crear objetos en memoria mediante el Class Loader y también se encarga de liberar de la memoria los objetos que no están siendo usados cuando haga falta más memoria, el encargado de esta tarea es el Garbage Collector.

Otras funciones del CLR son administrar los subprocesos o subtareas, a veces conocidas como hilos (Threads), controlar las excepciones o errores en tiempo de ejecución, administrar la seguridad del código .NET otorgándole los privilegios de acuerdo a reglas o políticas de seguridad, entre otras funciones implícitas en toda ejecución de una aplicación.

El CLR también administra la seguridad de tipos mediante un Sistema de Tipos Común (CTS) que obliga a que todos los fabricantes de código .NET cumplan con las especificaciones de tipos que el CLR obliga, es por eso que podemos escribir código en un lenguaje .NET como C# y usarlo en una aplicación Visual Basic y viceversa, lo cual es conocido como Neutralidad del Lenguaje.

En resumen, el motor en tiempo de ejecución (CLR) ofrece las siguientes ventajas:

- Mejoras en el rendimiento de la aplicación (performance).
- Capacidad para utilizar fácilmente componentes desarrollados en otros lenguajes.
- Tipos extensibles que proporciona una biblioteca de clases
- Características del lenguaje como herencia, interfaces y sobrecarga para la programación orientada a objetos.
- Compatibilidad con subprocesamiento libre explícito que permite la creación de aplicaciones multiprocesos escalables.
- Compatibilidad con el control de excepciones estructurado.
- Compatibilidad con atributos personalizados.
- Recolección de elementos no utilizados.
- Emplear delegados en lugar de punteros a funciones para mayor seguridad y protección de tipos.

1.4. Novedades del .NET Framework 4

En esta versión del .NET Framework existen muchas novedades entre las cuales podemos mencionar:

- Mejoras en funciones del entorno del sistema: System.Environment
 - Nuevas propiedades para detectar si el Sistema Operativo es de 64 bits: `is64BitOperatingSystem` y si el Proceso es de 64 Bits: `is64BitProcess`.
 - Más valores devueltos para las carpetas o folders especiales del sistema en la enumeración `SpecialFolder`.
- Mejoras en funciones de entrada y salida: System.IO
 - Nuevos métodos para leer gran cantidad de directorios: `Directory.EnumerateDirectories` y para leer gran cantidad de archivos de uno o más directorios: `Directory.EnumerateFiles`.
 - Nuevos métodos para manejar archivos usando todas sus líneas: `File.ReadAllLines`, `File.WriteAllLines`, `File.AppendAllLines`.
 - Nuevo método para copiar una secuencia de memoria en otra: `Stream.CopyTo`.
 - La nueva sobre carga de `Path.Combine` permite combinar rutas de accesos de archivos.
 - Mejora de los algoritmos de compresión de archivos: `GZipStream` y `DeflateStream` para no aumentar tamaño de los archivos ya comprimidos y también eliminación de restricción del tamaño de 4GB.
 - Compatibilidad con archivos asignados a memoria, lo cual sirve para editar archivos muy grandes y crear memoria compartida para la comunicación entre procesos: `MemoryMappedFile`.
- Mejoras en Diagnóstico y Rendimiento: System.Diagnostics

- Cálculo del uso del procesador y de la memoria por dominio de la aplicación y no solo por proceso.
- Monitoreo y recopilación de estadísticas para todos los dominios de aplicación con la propiedad `AppDomain.MonitoringIsEnabled`.
- Administrar excepciones que indican un estado de proceso dañado mediante el atributo `System.Runtime.ExceptionServices.HandleProcessCorruptedStateExceptionsAttribute`.
- Mejora en tratamiento de Cadenas: `System.String` y `System.Text.StringBuilder`
 - Se ha sobrecargado los métodos: `String.Join` y `String.Concat`.
 - Nuevo método para verificar si una cadena esta vacía o nula o con espacios en blanco: `String.IsNullOrEmpty`.
 - Nuevo método para limpiar el objeto constructor de cadenas: `StringBuilder.Clear`.
- Mejora en manejo de Colecciones: `System.Collections`
 - Nueva clase genérica `SortedSet` que permite ordenar automáticamente los elementos del conjunto después de una inserción, eliminación o búsqueda de elementos.
- Programación Paralela: `System.Threading.Tasks`
 - Paralelismo de datos mediante: `Parallel.For` y `Parallel.ForEach`
 - Paralelismo de tareas mediante: `Task.Start`, `Task.FactoryStartNew` y `Parallel.Invoke`.
 - LINQ Paralelo (PLINQ) mediante operadores: `AsParallel`, `AsSequential`, `AsOrdered`, `ForAll`, `AsUnordered`, `WithCancellation`, `WithDegreeOfParallelism`.
- Mejoras en Acceso a Datos: `System.Data`
 - ADO .NET Entity Framework:

- Objetos que ignoran la persistencia
- Funciones en consultas LINQ.
- Generación código personalizado capa de objeto.
- Datos dinámicos:
 - Validación automática basada en restricciones definidas en el modelo de datos.
- Servicios de Datos de WCF:
 - Enlace de datos.
 - Contar las entidades de un conjunto de entidades.
 - Paginación controlada por servidor.
 - Proyecciones de consultas.
 - Proveedores de servicios de datos personalizados.
 - Transmitir por secuencias recursos binarios
- Mejoras en ASP NET: System.Web
 - Nueva API caché, estado de sesión.
 - Nuevo administrador precarga de aplicación.
 - Compatibilidad mejorada con estándares web.
 - Nuevas características en controles de datos.
 - Mejora en administración de estados de vista.
 - Nuevo control Chart para graficos.
 - Mejoras en MVC.
 - Datos dinámicos.
 - Compatibilidad en Microsoft Ajax Library.

- Intellisense mejorado para JScript.
- Autocompletar HTML y ASP .NET
- Mejoras en Windows Presentations Foundation (WPF):
 - Nuevos controles: Calendar, DataGrid y DatePicker.
 - VisualStateManager: cambio estados control.
 - Gráficos y animación admite redondeo diseño
 - Mejora en la presentación de texto y color.
 - Enlace en Command de InputBinding, objetos dinámicos y la propiedad Text.
 - XBAP admite la comunicación con la página web y la implementación de plena confianza.
 - System.Windows.Shell permite comunicarse con la barra de tareas de Windows 7.
 - Mejoras en WPF y Silverlight Designer VS2010.
- Mejoras en Windows Communication Foundation (WCF):
 - Activación basada en la configuración: no svc.
 - System.Web.Routing: direcciones URL sin ex.
 - Compatibilidad varios enlaces de sitios de IIS.
 - Servicio enrutamiento: mensajes según contenido.
 - Compatibilidad con WS-Discovery.
 - Servicios de flujo trabajo: integra WCF y WWF.
 - Características de WCF REST:
 - Caché de servicios Web HTTP.
 - Compatibilidad con formatos Web HTTP.

- Página de ayuda de los servicios Web HTTP.
- Control de Errores Web HTTP.
- Compatibilidad con JavaScript en dominios: JSON.
- Configuración simplificada.
- Mejoras en Windows Workflow Foundation (WWF):
 - Modelo mejorado de actividad de flujo de trabajo.
 - Opciones completas de actividad composición.
 - Biblioteca de actividades integrada ampliada.
 - Modelo explícito de datos de actividad.
 - Opciones mejoradas de hospedaje, persistencia y seguimiento:
 - Persistencia explícita mediante la actividad Persist.
 - Persistencia sin descarga.
 - Impedir la persistencia mediante zonas sin persistencia.
 - Uso de transacciones de ambiente del host.
 - Grabación de información de seguimiento en el registro de eventos.
 - Reanudación de flujos de trabajo pendientes usando un objeto Bookmark.
 - Mayor facilidad para extender el diseñador de WWF.

2. Visual Studio .NET 2010

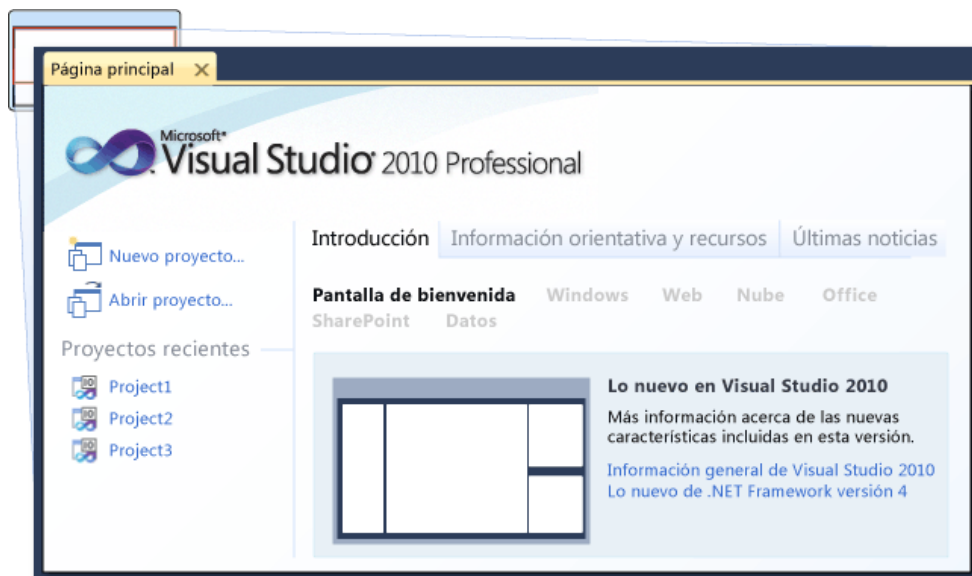
Visual Studio .NET es la herramienta de desarrollo Microsoft que utiliza la plataforma de desarrollo .NET Framework, para crear rápidamente aplicaciones .NET de todo tipo. Podría crear aplicaciones simples usando el bloc de notas y un compilador .NET, pero si la aplicación es compleja y tiene muchas pantallas demoraría demasiado, el Visual Studio le permite simplificar todo el desarrollo de desarrollo e inclusive las pruebas.

A continuación se muestra la descripción del IDE y las novedades de Visual Studio 2010 obtenidas del MSDN de Microsoft (ver referencias 6 y 7 al final del libro).

2.1. Descripción del Entorno Integrado de Desarrollo (IDE)

La gama de productos de Visual Studio comparte un único entorno de desarrollo integrado (IDE) que se compone de varios elementos: la barra de menús, la barra de herramientas Estándar, varias ventanas de herramientas que se acoplan u ocultan automáticamente a la izquierda, en la parte inferior y a la derecha, así como en el espacio del editor. Las ventanas de herramientas, menús y barras de herramientas disponibles dependen del tipo de proyecto o archivo en el que esté trabajando.

Gráfico 1.2: IDE con la configuración de desarrollo general aplicada



Dependiendo de la configuración aplicada y de las subsiguientes personalizaciones que haya realizado, variará la colocación de las ventanas de herramientas y de otros elementos en el IDE. Puede cambiar la configuración mediante el Import and Export Settings Wizard. Al seleccionar la opción Restablecer todas las configuraciones, se puede cambiar el lenguaje de programación predeterminado.

Puede desplazarse y acoplar ventanas con facilidad mediante el rombo de guía visual u ocultar temporalmente las ventanas utilizando el comando Ocultar automáticamente.

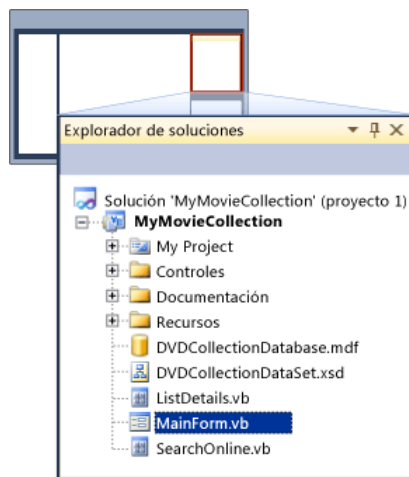
Puede utilizar el modelo de automatización de Visual Studio para automatizar y extender el IDE.

Sistema de Proyectos

Las soluciones y los proyectos contienen elementos en forma de referencias, conexiones de datos, carpetas y archivos necesarios para crear la aplicación. Un contenedor de tipo solución puede contener varios proyectos y un contenedor de tipo proyecto normalmente contiene varios elementos.

El Explorador de Soluciones muestra soluciones, sus proyectos y los elementos incluidos en dichos proyectos. En el Explorador de soluciones, puede abrir archivos para editar, agregar nuevos archivos a un proyecto y ver las propiedades de las soluciones, proyectos y elementos.

Gráfico 1.3: Ventana del Explorador de Soluciones



Editores y Diseñadores

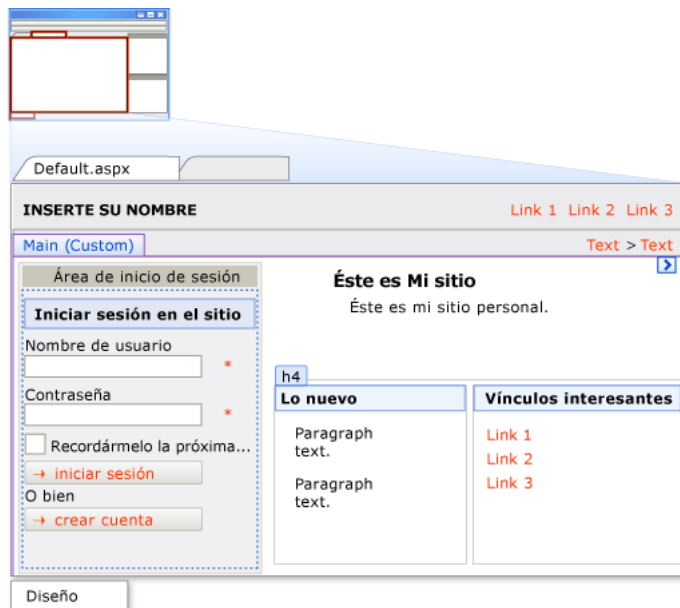
El editor y los diseñadores que utilice dependerán del tipo de archivo o documento que esté creando. El Editor de texto es el procesador de textos

básico del IDE, mientras que el Editor de código es el editor de código fuente básico.

Otros editores, como el Editor CSS, el Diseñador HTML y el Diseñador de páginas Web, comparten muchas de las características del Editor de código, junto con mejoras específicas en el tipo de código o de marcado admitido.

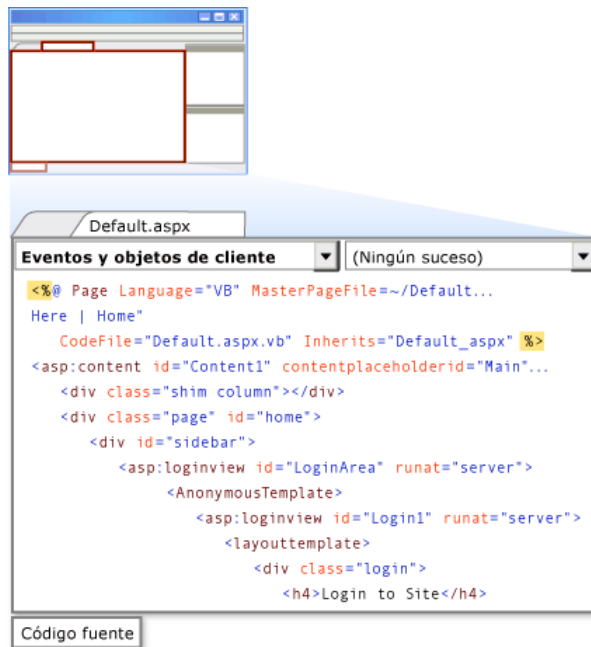
Los editores y diseñadores normalmente tienen dos vistas: una vista de diseño gráfica y la vista de código subyacente o vista de código fuente. La vista de diseño le permite especificar la ubicación de los controles y otros elementos en la interfaz de usuario o la página web. Puede arrastrar controles desde el cuadro de herramientas y colocarlos en la superficie de diseño.

Gráfico 1.4: Diseñador de Páginas Web vista Diseño



La vista Código fuente muestra el código fuente del archivo o documento. Esta vista admite ayudas de codificación como IntelliSense, secciones de código plegables, Refactorización (C#) e inserción de fragmentos de código. Otras características incluyen el ajuste automático de línea, los marcadores y la visualización de números de línea, por citar algunos.

Gráfico 1.5: Diseñador de Páginas Web vista Código Fuente



Algunos editores, como el Diseñador de páginas web y el Diseñador XAML, también proporcionan una vista híbrida que le permite ver la vista del gráfico y del código de un archivo simultáneamente. Esta vista se llama la Vista dividida.

Gráfico 1.6: Diseñador de Páginas Web vista Dividida

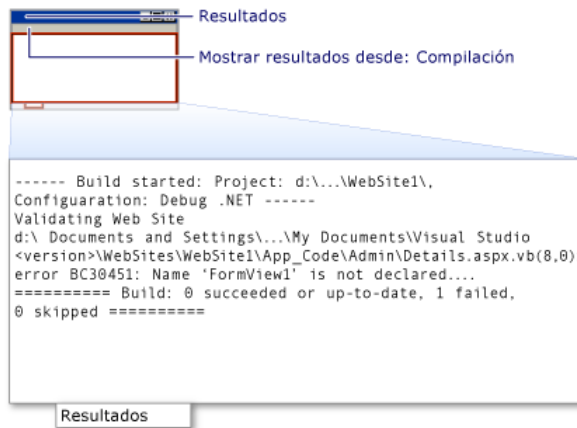


Herramientas de compilación y depuración

Visual Studio proporciona un sólido conjunto de herramientas de compilación y depuración. Con las configuraciones de compilación puede seleccionar los componentes que se van a generar, excluir los que no se van a generar y determinar cómo se van a generar los proyectos seleccionados y en qué plataforma. Puede tener configuraciones de compilación para soluciones y para proyectos.

Cuando genera, está comenzando el proceso de depuración. La compilación de la aplicación le ayuda a detectar errores de compilación. Estos errores pueden deberse a una sintaxis incorrecta, a palabras clave mal escritas o a divergencias entre los tipos. La Resultados (Ventana) muestra estos tipos de errores.

Gráfico 1.7: Ventana de salida con información de compilación



Después de generar la aplicación, puede utilizar el depurador para detectar y corregir problemas como errores lógicos y semánticos que se descubren en tiempo de ejecución. En el modo de interrupción, puede examinar las variables locales y otros datos pertinentes utilizando herramientas como Ventanas de variables y la Ventana Memoria.

Gráfico 1.8: Ventana de formulario VB en modo interrupción

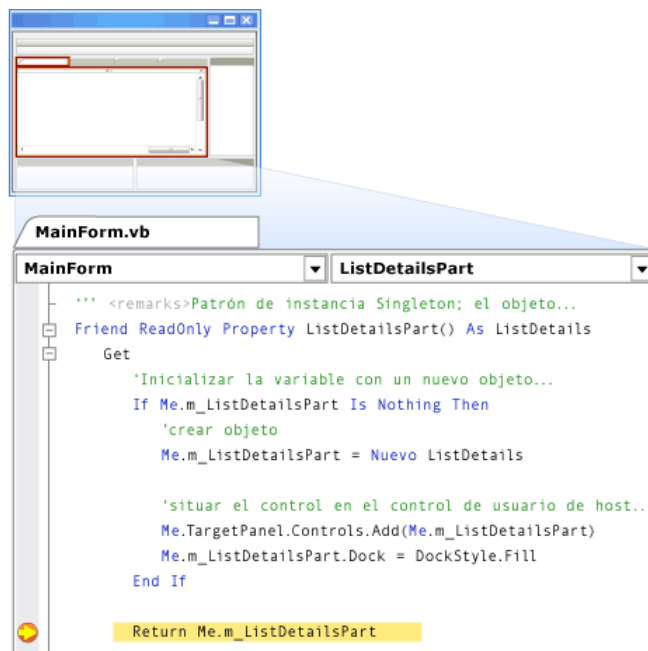


Gráfico 1.9: Ventanas de herramientas de depuración



La Ventana Lista de errores muestra errores, advertencias y otros mensajes relacionados con la depuración.

2.2. Novedades de Visual Studio 2010

A continuación presentamos de forma resumida las novedades de Visual Studio 2010:

- Novedades en Lenguajes
 - Visual Basic 2010: Continuación de línea implícita, las propiedades implementadas automáticamente y los inicializadores de colección.
 - Visual C# 2010: Tipo dynamic, los argumentos opcionales y con nombre, la programación de Office mejorada y la varianza.
 - Visual C++ 2010: Expresiones lambda, el declarador de referencias de valores R y las palabras clave auto, decltype y static_assert.
- Novedades del Editor de Visual Studio
 - Comportamiento de acoplamiento mejorado.
 - Zoom.
 - Selección de cuadros.
 - Jerarquía de llamadas.
 - Navegar a.
 - Resaltar referencias.
 - Generar a partir del uso.
 - Modo de sugerencia de Intellisense.
- Novedades en el desarrollo de Office
 - Desarrollar Soluciones para Microsoft Office 2010.
 - Mayor compatibilidad con la cinta de opciones en las soluciones para Microsoft Office 2010.

- .NET Framework 4 como destino.
- Microsoft Office 2010 incluye el Motor en tiempo de ejecución de Visual Studio Tools para Office.
- Implementar las soluciones de Office para todos los usuarios.
- Implementar varias soluciones de Office en un solo paquete.
- Realizar acciones adicionales después de la instalación de la solución de Office.
- Novedades en el desarrollo de Aplicaciones de Datos
 - Conectarse a orígenes de datos
 - Enlace de datos de arrastrar y colocar para WPF.
 - Enlace de datos de arrastrar y colocar para Silverlight.
 - Conectar a datos en varios objetos.
 - Extender consultas en el control EntityDataSource.
 - Herramientas de Entity Data Model
 - Generación de código de capa de objeto personalizado.
 - Compatibilidad con Model-First.
 - Compatibilidad con tipos complejos.
 - Servicio de asignación de nombres.
 - Funcionalidad mejorada del Explorador de modelos.
 - Extensibilidad de Entity Designer.
- Novedades en el desarrollo con SharePoint
 - Crear y ejecutar las pruebas unitarias y depurar las aplicaciones de SharePoint con IntelliTrace. (Requiere Service Pack 1).
 - Importar, modificar y ampliar paquetes de soluciones (.wsp).

- Desarrollar soluciones de SharePoint con plantillas para los proyectos y elementos de proyecto.
 - Diseñar formularios de asociación e iniciación para flujos de trabajo secuenciales y de estados.
 - Agregar e integrar datos back-end usando modelos de Conectividad a datos profesionales (BDC).
 - Crear elementos web y páginas de aplicación de sitios de SharePoint.
 - Examinar los sitios de SharePoint con el Explorador de servidores.
 - Empezar a depurar las aplicaciones de SharePoint presionando F5.
 - Crear y validar paquetes de soluciones.
 - Ampliar los elementos de proyecto de SharePoint existentes y agregar menús contextuales.
- Novedades en desarrollo de Reportes con Microsoft Reports
 - Diseñador de informes para el esquema RDL 2008.
 - Nuevo Asistente para informes.
 - Mejoras en los controles ReportViewer.
 - Compatibilidad de AJAX en el control de servidor web de ASP .NET
 - Mejoras de programación en los controles ReportViewer.

3. Visual Basic .NET 2010

Visual Basic 2010 es la última versión del lenguaje Visual Basic de Microsoft que continúa con la facilidad de las versiones anteriores pero agrega nueva funcionalidad para mejorar el desarrollo.

En esta parte veremos algunas características del Lenguaje, tales como las estructuras de control de flujo, los tipos de datos, operadores y variables, además de las novedades de la versión 2010, dicha información es también obtenida del MSDN de Microsoft (ver referencias 8, 9, 10, 11 y 12 al final del libro).

3.1. Estructuras de Control de Flujo

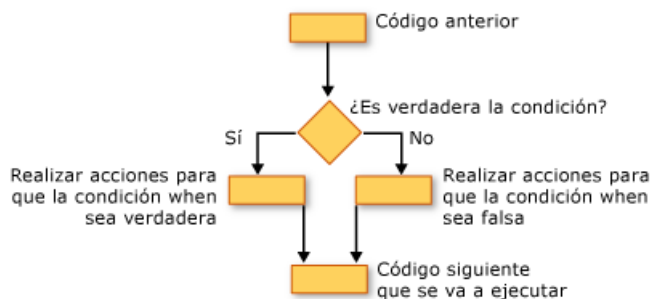
Las estructuras de control de flujo se pueden clasificar en:

Estructuras de decisión

Visual Basic permite probar condiciones y realizar diferentes operaciones en función de los resultados de la prueba. Puede comprobar si una condición es verdadera o falsa, los distintos valores de una expresión o las diferentes excepciones que se generan al ejecutar una serie de instrucciones.

En el siguiente ejemplo se muestra una estructura de decisión que prueba si el valor de una condición es true y emprende distintas acciones en función del resultado.

Gráfico 1.10: Ejemplo de una estructura de decisión



- **Construcción If...Then...Else:** Permiten probar una o más condiciones y ejecutar una o más instrucciones en función de cada condición. Puede probar las condiciones y tomar medidas de las maneras siguientes:
 - Ejecutar una o más instrucciones si una condición es True.
 - Ejecutar una o más instrucciones si una condición es False.
 - Ejecutar algunas instrucciones si una condición es True y otras si es False.
 - Probar una condición adicional si una condición anterior es False.

La estructura de control que proporciona todas estas posibilidades es Instrucción If...Then.... Puede utilizar una versión de una línea si tiene simplemente una comprobación y una instrucción para ejecutar. Si tiene un conjunto más complejo de condiciones y acciones, puede utilizar la versión de varias líneas.

'Sintaxis de múltiples líneas:

```
If condición1 [ Then ]  
    [ instrucciones ]  
[ ElseIf condición2 [ Then ]  
    [instrucciones] ]  
[ Else  
    [instrucciones] ]  
End If
```

'Sintaxis de una simple línea:

```
If condición Then [instrucción] [ Else [instrucción] ]
```

- **Construcción Select...Case:** Permite evaluar una expresión una vez y ejecutar distintos conjuntos de instrucciones basados en diferentes valores posibles.

```
Select [ Case ] expression  
    [ Case listaExpresiones  
        [instrucciones] ]  
    [ Case Else
```

[instrucciones]]

End Select

- **Construcción Try...Catch...Finally:** Permiten ejecutar un conjunto de instrucciones en un entorno que **conserva el control si una de las instrucciones provoca una excepción**. Puede tomar distintas medidas para excepciones diferentes. Opcionalmente, puede especificar un bloque de código que se ejecuta antes de salir de la construcción Try...Catch...Finally completa, sin tener en cuenta el resultado.

Try

[instrucciones]

[Exit Try]

[Catch [exception [As type]] [When expression]

[instrucciones]

[Exit Try]]

[Catch ...]

[Finally

[instrucciones]]

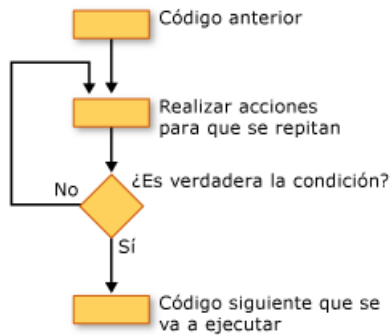
End Try

Estructuras de bucles

Las estructuras de bucles de Visual Basic **permiten ejecutar una o varias líneas de código de forma repetitiva**. Puede repetir las instrucciones de una estructura de bucles hasta que una condición sea True, una condición sea False, un número de veces especificado o una vez para cada objeto de una colección.

En el siguiente ejemplo se muestra una estructura de bucle que ejecuta un conjunto de instrucciones hasta que una condición se convierta en verdadera.

Gráfico 1.11: Ejemplo de una estructura de bucles



- **Bucles While:** La construcción While...End While ejecuta un conjunto de instrucciones mientras la condición especificada en la instrucción While sea True.

```
While condición  
    [instrucciones]  
    [ Exit While ]  
    [instrucciones]  
End While
```

- **Bucles Do:** La construcción Do...Loop le permite probar una condición al comienzo o al final de una estructura de bucle. También puede especificar si repite el bucle mientras la condición sigue siendo True o hasta que se convierta en True.

```
Do { While | Until } condición  
    [instrucciones]  
    [ Exit Do ]  
    [instrucciones]  
Loop  
ó  
Do  
    [instrucciones]  
    [ Exit Do ]  
    [instrucciones]  
Loop { While | Until } condición
```

- **Bucles For:** La construcción For...Next **ejecuta el bucle un número fijo de veces.** Utiliza una variable de control de bucle, también denominada contador para realizar el seguimiento de las repeticiones. Especifica los valores de inicio y fin de este contador, y puede especificar opcionalmente la cantidad en la que se incrementa de una repetición a la siguiente.

```
For contador [ As TipoDato ] = inicio To fin [ Step paso ]  
    [instrucciones]  
    [ Continue For ]  
    [instrucciones]  
    [ Exit For ]  
    [instrucciones]  
Next [contador]
```

- **Bucles For Each:** La construcción For Each...Next ejecuta un conjunto de instrucciones una vez para cada elemento de una colección. Especifica la variable de control de bucle pero no tiene que determinar los valores de inicio y fin para ella.

```
For Each elemento [ As TipoDato ] In lista  
    [instrucciones]  
    [ Continue For ]  
    [instrucciones]  
    [ Exit For ]  
    [instrucciones]  
Next [ elemento ]
```

Estructuras de control adicionales

- **Construcción Using...End Using:** Establece un bloque de instrucciones dentro del cual utiliza un recurso como una conexión de SQL. Puede adquirir el recurso opcionalmente con la instrucción Using. Cuando sale del bloque Using, Visual Basic dispone automáticamente del recurso para que esté disponible para otro código. El recurso debe ser local y ser descartable.

```
Using { listarecurso | expressionrecurso }  
    [instrucciones]  
End Using
```

- Construcción With...End With: Permite especificar una referencia de objeto una vez y ejecutar luego una serie de instrucciones que tienen acceso a sus miembros. Esto puede simplificar su código y mejorar el rendimiento porque Visual Basic no tiene que restablecer la referencia para cada instrucción que tiene acceso a él.

```
With object  
    [instrucciones]  
End With
```


3.2. Tipos de Datos en Visual Basic

El tipo de datos de un elemento de programación hace referencia al tipo de datos que puede contener y a cómo se almacenan dichos datos. Los tipos de datos se aplican a todos los valores que pueden almacenarse en la memoria del equipo o participar en la evaluación de una expresión. Cada variable, literal, constante, enumeración, propiedad, parámetro de procedimiento, argumento de procedimiento y valor devuelto por un procedimiento tiene un tipo de datos.

Tipos de datos declarados

A menos que utilice la programación sin tipos, debe declarar los tipos de datos de todos los elementos de programación.

Un elemento de programación se define con una instrucción de declaración y su tipo de datos se especifica con la cláusula As. La tabla siguiente muestra las instrucciones utilizadas para declarar diversos elementos.

Elemento de programación	Declaración de tipos de datos
Variable	En una Instrucción Dim Dim amount As Double Static yourName As String Public billsPaid As Decimal = 0
Literal	Con un carácter de tipo literal Dim searchChar As Char = "."C
Constante	En una Instrucción Const Const modulus As Single = 4.17825F

Enumeración	En una Instrucción Enum Public Enum colors
Propiedad	En una Instrucción Property Property region() As String
Parámetro de procedimiento	En una Instrucción Sub, Function u Operador Sub addSale(ByVal amount As Double)
Argumento de procedimiento	En el código de llamada; cada argumento es un elemento de programación que ya se ha declarado o una expresión que contiene los elementos declarados subString = Left(inputString, 5)
Valor devuelto por procedimiento	En una Instrucción u Operador Function convert(ByVal b As Byte) As String

Resumen de Tipos de datos

En la tabla siguiente se muestran los tipos de datos de Visual Basic .NET, los tipos compatibles con Common Language Runtime, su asignación de almacenamiento nominal y sus intervalos de valores.

Tipo de Visual Basic	Tipo CLR	Asignación de almacenamiento nominal	Intervalo de valores
Boolean	Boolean	En función de la plataforma de	True o False

		implementación	
Byte	Byte	1 byte	0 a 255 (sin signo)
Char (carácter individual)	Char	2 bytes	0 a 65535 (sin signo)
Fecha	DateTime	8 bytes	0:00:00 (medianoche) del 1 de enero de 0001 a 11:59:59 p.m. del 31 de diciembre de 9999.
Decimal	Decimal	16 bytes	0 a +/- 79.228.162.514.264.337.593.5 43.950.335 (+/-7,9... E+28) [†] sin separador decimal; 0 a +/- 7,92281625142643375935439 50335 con 28 posiciones a la derecha del decimal; el número distinto de cero más pequeño es +/- 0,000000000000000000000000 00001 (+/-1E-28) [†]
Double (punto)	Double	8 bytes	-1,79769313486231570E+308 a -4,94065645841246544E-

flotante de precisión doble)			324 ⁺ para los valores negativos; 4,94065645841246544E-324 a 1,79769313486231570E+308 ⁺ para los valores positivos
Integer	Int32	4 bytes	-2.147.483.648 a 2.147.483.647 (con signo)
Long (entero largo)	Int64	8 bytes	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 (9,2...E+18 ⁺) (con signo)
Objeto	Object (clase)	4 bytes en plataforma de 32 bits 8 bytes en plataforma de 64 bits	Cualquier tipo puede almacenarse en una variable de tipo Object
SByte	SByte	1 byte	-128 a 127 (con signo)
Short (entero corto)	Int16	2 bytes	-32.768 a 32.767 (con signo)
Single (punto)	Single	4 bytes	-3,4028235E+38 a - 1,401298E-45 ⁺ para los

flotante de precisión sencilla)			valores negativos; 1,401298E-45 a 3,4028235E+38 [†] para los valores positivos
String (longitud variable)	String (clase)	En función de la plataforma de implementación	0 a 2.000 millones de caracteres Unicode aprox.
UInteger	UInt32	4 bytes	0 a 4.294.967.295 (sin signo)
ULong	UInt64	8 bytes	0 a 18.446.744.073.709.551.615 (1,8...E+19 [†]) (sin signo)
User-Defined (estructura)	(hereda de ValueType)	En función de la plataforma de implementación	Cada miembro de la estructura tiene un intervalo de valores determinado por su tipo de datos y es independiente de los intervalos de valores correspondientes a los demás miembros.
UShort	UInt16	2 bytes	0 a 65.535 (sin signo)

Consumo de memoria

Al declarar un tipo de datos básico, no debe suponerse que su consumo de memoria es igual a su asignación de almacenamiento nominal. Esto se debe a las consideraciones siguientes:

- **Asignación de almacenamiento:** El CLR puede asignar el almacenamiento en función de las características actuales de la plataforma en la que se ejecuta la aplicación. Si la memoria está casi completa, se pueden empaquetar los elementos declarados de la forma más estrecha posible. En otros casos, se podrían alinear las direcciones de memoria a los límites del hardware naturales para optimizar el rendimiento.
- **Ancho de plataforma:** La asignación de almacenamiento en una plataforma de 64 bits es diferente a la asignación en una plataforma de 32 bits.

Las mismas consideraciones se aplican a cada miembro de un tipo de datos compuesto, como una estructura o una matriz. No se pueden sumar simplemente todas las asignaciones de almacenamiento nominales de los miembros de tipo. Además, existen otras consideraciones, como las siguientes:

- **Sobrecarga.** Algunos tipos compuestos tienen requisitos adicionales de memoria. Por ejemplo, una matriz utiliza memoria adicional para la matriz en sí y para cada dimensión. En una plataforma de 32 bits, esta sobrecarga corresponde a 12 bytes y 8 bytes por cada dimensión. En una plataforma de 64 bits, los requisitos se duplican.
- **Diseño de almacenamiento.** No debe suponerse que el orden de almacenamiento en la memoria es igual al orden de declaración. Ni siquiera pueden hacerse predicciones sobre la alineación de bytes, como un límite de 2 bytes o de 4 bytes. Si define una clase o estructura y necesita controlar el diseño de almacenamiento de sus miembros, puede aplicar el atributo `StructLayoutAttribute` a la clase o estructura.

Una variable `Object` que haga referencia a un tipo de datos básico o compuesto, utiliza 4 bytes además de los datos contenidos en el tipo de datos.

3.3. Operadores y Expresiones en Visual Basic

Un operador es un elemento de código que realiza una operación en uno o más elementos de código que contienen valores. Los elementos de valor incluyen variables, constantes, literales, propiedades, valores devueltos de procedimientos Function y Operator y expresiones.

Una expresión es una serie de elementos de valor combinados con operadores, que produce un nuevo valor. Los operadores actúan sobre los elementos de valor realizando cálculos, comparaciones y otras operaciones.

Operadores aritméticos

Operador	Operación	Ejemplo
+	Suma	Dim x As Integer = 67 + 34
-	Resta	Dim y As Integer = 67 - 34
*	Multiplicación	Dim x As Double = 50 * 20 Resultado: 1000
/	División decimal	Dim y As Double = 50 / 20 Resultado: 2.5
^	Exponenciación	Dim x As Double = 5 ^ 2 Resultado: 25
\	División entera	Dim x As Integer = 50 \ 20

		Resultado: 2
Mod	Residuo	Dim y As Double = 50 Mod 20 Resultado: 10

Operadores de comparación

Operador	Operación	Ejemplo
=	Igualdad	23 = 33 ' False 23 = 23 ' True
<>	Desigualdad	23 <> 33 ' True 23 <> 23 ' False
<	Menor que	23 < 33 ' True 23 < 23 ' False
>	Mayor que	23 > 33 ' False 33 > 23 ' True
<=	Menor o igual que	23 <= 33 ' True 33 <= 23 ' False

>=	Mayor o igual que	23 >= 33 ' False 23 >= 23 ' True

Operadores de concatenación

Operador	Operación	Ejemplo
+	Suma 2 números o cadenas. Los 2 operandos deben ser del mismo tipo sino se genera un error.	1 + 2 = 3 "1" + "2" = "12" 1 + "2" = Error
&	Solo se usa para sumar cadenas. Realiza una conversión implícita si uno o los 2 operandos son números.	1 & 2 = "12" "1" & "2" = "12" 1 & "2" = "12"

Operadores lógicos

Operador	Operación	Ejemplo
Not	Negación lógica.	Dim x As Boolean = true Dim y As Boolean = Not x ' False

And	Conjunción lógica.	<code>a = 23 > 14 And 11 > 8 ' True</code> <code>b = 14 > 23 And 11 > 8 ' False</code>
Or	Disyunción lógica.	<code>c = 23 > 14 Or 8 > 11 ' True</code> <code>d = 23 > 67 Or 8 > 11 ' False</code>
Xor	Exclusión lógica.	<code>e = 23 > 67 Xor 11 > 8 ' True</code> <code>f = 23 > 14 Xor 11 > 8 ' False</code> <code>g = 14 > 23 Xor 8 > 11 ' False</code>
AndAlso	Conjunción lógica con corto circuito.	<code>a = 23 < 14 AndAlso 11 > 8 ' False</code> Al ser True la primera condición sale
OrElse	Disyunción lógica con corto circuito.	<code>b = 23 > 14 OrElse 8 > 11 ' True</code> Al ser True la primera condición sale

3.4. Variables en Visual Basic

Visual Basic, al igual que la mayoría de los lenguajes de programación, usa variables para almacenar los valores. Una variable tiene un nombre (la palabra que se usa para referirse al valor que contiene la variable). Una variable también tiene un tipo de datos, que determina el tipo de datos que puede almacenar la variable. Una variable puede representar una matriz si tiene que almacenar un conjunto indizado de elementos de datos estrechamente relacionados entre sí.

La inferencia de tipos de variable local permite declarar las variables sin tener que indicar de forma explícita un tipo de datos. En lugar de ello, el compilador deduce el tipo de la variable a partir del tipo de la expresión de inicialización.

Niveles de Declaración de Variables

- Valor local y variables miembros

Una variable local es aquella que se declara dentro de un procedimiento. Una variable miembro es un miembro de un tipo de Visual Basic; se declara en el nivel de módulo, dentro de una clase, estructura o módulo, pero no dentro de ningún procedimiento interno de esa clase, estructura o módulo.

- Variables compartidas y de instancias

La categoría de una variable miembro, en una clase o estructura, depende de que la variable esté o no compartida. Si una variable se declara con la palabra clave `Shared`, es una variable compartida, y existe en una única copia compartida por todas las instancias de la clase o estructura.

De lo contrario, es una variable de instancia, y se crea una copia independiente de ella para cada instancia de la clase o estructura. Una copia determinada de una variable de instancia sólo está disponible en la instancia para la cual se creó. Es independiente de una copia en cualquier otra instancia.

Declarar el tipo de dato

La cláusula `As` de la instrucción de declaración permite definir el tipo de datos o de objetos de la variable que se está declarando. Se puede especificar cualquiera de los siguientes tipos para una variable:

- Un tipo de datos básico, como `Boolean`, `Long` o `Decimal`.
- Un tipo de datos compuesto, como una matriz o una estructura.
- Un tipo de objeto o clase, definido en su aplicación o en otra aplicación
- Clase de .NET Framework, como `Label` o `TextBox`
- Un tipo de interfaz, como `Comparable` o `IDisposable`

Se pueden declarar varias declarar distintas variables en la misma instrucción sin necesidad de repetir el tipo de datos. En las instrucciones siguientes, las variables `i`, `j` y `k` se declaran como tipo `Integer`, `l` y `m` como `Long`, y `x` e `y` como `Single`:

```
Dim i, j, k As Integer
```

```
Dim l, m As Long, x, y As Single
```

Inferencia de tipo de variable local

La inferencia de tipos se usa para determinar los tipos de datos de las variables locales que se han declarado sin ninguna cláusula `As`. El compilador deduce el tipo de la variable a partir del tipo de la expresión de inicialización. Esto permite declarar variables sin especificar un tipo de forma explícita. En el ejemplo siguiente, `num1` y `num2` son con establecimiento inflexible de tipos como enteros.

```
Public Sub inferenceExample()  
    ' Usando declaración explícita.  
  
    Dim num1 As Integer = 3  
  
    ' Usando inferencia de tipo variable local.
```

```
Dim num2 = 3
```

```
End Sub
```

Declarar características

El período de duración de una variable representa el tiempo durante el cual la variable está disponible para que pueda ser utilizada. En general, una variable existe mientras el elemento que lo declara (como un procedimiento o clase) siga existiendo. En algunos casos es posible extender la duración de una variable.

El ámbito de una variable está formado por todo código que puede hacer referencia a la variable sin tener que calificar su nombre. El ámbito de una variable está determinado por la ubicación en la que se haya declarado la variable. El código de una región determinada puede utilizar las variables definidas en dicha región sin necesidad de especificar los nombres de las variables.

El nivel de acceso de una variable es la extensión de código que tiene permiso para tener acceso a ella. El modificador de acceso (como Public o Private) que utiliza en la instrucción Dim es quien determina esto.

3.5. Novedades de Visual Basic 2010

Propiedades auto implementadas

Las propiedades auto implementadas proporcionan una sintaxis abreviada que permite especificar rápidamente una propiedad de una clase sin tener que escribir el código Get y Set para la propiedad.

Inicializadores de colección

Los inicializadores de colección proporcionan una sintaxis abreviada que permite crear una colección y rellenarla con un conjunto inicial de valores. Los inicializadores de colección son útiles cuando se está creando una colección a partir de un conjunto de valores conocidos como, por ejemplo, una lista de opciones de menú o categorías.

Continuación de línea implícita

En muchos casos, la continuación de línea implícita permite continuar una instrucción en la línea consecutiva siguiente sin utilizar el carácter de subrayado (_).

Expresiones lambda de múltiples líneas y subrutinas

La compatibilidad con la expresión lambda se ha expandido para admitir las subrutinas además de las funciones de lambda de múltiples líneas y subrutinas.

Nueva opción de la línea de comandos para especificar una versión de lenguaje

La opción /langversion de la línea de comandos hace que el compilador acepte únicamente la sintaxis que sea válida en la versión especificada de Visual Basic.

Compatibilidad con la equivalencia de tipos

Ahora se puede implementar una aplicación que contiene información de tipo incrustada en lugar de información de tipos que se importa desde un ensamblado de interoperabilidad primario (PIA). Con la información de tipos incrustada, la aplicación puede utilizar los tipos en un motor en tiempo de ejecución sin necesidad de una referencia al ensamblado en tiempo de ejecución. Si se publican varias versiones del ensamblado del runtime, la aplicación que contiene la información de tipos incrustada puede funcionar con las diferentes versiones sin que sea necesario volver a compilarla.

Compatibilidad dinámica

Visual Basic enlaza a los objetos de los lenguajes dinámicos como IronPython e IronRuby.

Covarianza y contravarianza

La covarianza permite usar un tipo más derivado que el especificado por el parámetro genérico, mientras que la contravarianza permite utilizar un tipo menos derivado. Esto permite la conversión implícita de las clases que implementan interfaces variantes y proporciona mayor flexibilidad a la hora de hacer coincidir las firmas de método con tipos de delegado variantes. Se pueden crear interfaces y delegados variantes mediante las nuevas palabras clave In y Out. .NET Framework también incluye compatibilidad con la varianza para varios delegados e interfaces genéricos existentes, incluidos la interfaz IEnumerable(Of T) y los delegados Action(Of T) y Func(Of TResult).

Navegar a

Se puede usar la característica Navegar a para buscar un símbolo o un archivo en código fuente. Puede buscar palabras clave incluidas en un símbolo concatenado mediante notación Camel o caracteres de subrayado a fin de dividir dicho símbolo en palabras clave.

Resaltar referencias

Al hacer clic en un símbolo en el código fuente, todas las instancias de ese símbolo se resaltan en el documento.

En muchas estructuras de control, al hacer clic en una palabra clave, se resaltan todas las palabras clave en la estructura. Por ejemplo, al hacer clic en If en una construcción If...Then...Else, se resaltan todas las instancias de If, Then, ElseIf, Else y End If de la construcción.

Para desplazarse al siguiente o anterior símbolo resaltado, puede usar CTRL+MAYÚS+FLECHA ABAJO o CTRL+MAYÚS+FLECHA ARRIBA.

Generar a partir del uso

La característica Generar a partir del uso permite usar clases y miembros antes de definirlos. Puede generar un código auxiliar para cualquier clase, constructor, método, propiedad, campo o enumeración que desee utilizar pero no ha definido todavía. Puede generar nuevos tipos y miembros sin salir de su ubicación actual en el código. De este modo, se minimizan las interrupciones en el flujo de trabajo.

La característica Generar a partir del uso admite estilos de programación como el desarrollo de pruebas en primer lugar. Para obtener más información, vea Generar a partir del uso.

Modo de sugerencia de IntelliSense

IntelliSense proporciona ahora dos alternativas para completar las instrucciones de IntelliSense: el modo de finalización y el modo de sugerencia. El modo de sugerencia se utiliza cuando las clases y los miembros se usan antes de definirlos.

Preguntas de Repaso

1. Qué es .NET Framework?
2. Menciona 3 tipos de aplicaciones que se pueden crear en .NET Framework.
3. Que tecnologías aparecieron con .NET Framework 3?
4. Menciona 5 lenguajes .NET.
- 5.Cuál es el criterio principal para seleccionar un Lenguaje .NET?
6. Que lenguaje es recomendable para aquellos que no tienen experiencia con C, Java o la POO.
7. Cuáles son los componentes principales de .NET Framework?
8. Que es la BCL del .NET Framework?
9. Cómo se clasifican los tipos en .NET Framework?
10. Menciona 3 tipos complejos por referencia.
- 11.Cuál es la principal función de un espacio de nombres (Namespace)?
12. Menciona 5 espacios de nombres de la BCL.
13. Qué es el CLR del .NET Framework?
14. Cómo se llama el programa del CLR que se encarga de liberar automáticamente la memoria que no esta siendo utilizada?
15. En qué tipo de lenguaje escriben el código los desarrolladores .NET
16. Qué es un ensamblado (Assembly) y en que lenguaje se encuentra?

17. Qué tipo de código ejecuta el CLR y en que lenguaje se encuentra?
18. Menciona 3 funciones del CLR.
19. Cuáles son los nuevos métodos de la clase Environment para verificar sistemas y procesos de 64 bits?
20. Cuáles son los nuevos métodos de la clase Directory para listar directorios y archivos?
21. Con qué clase se puede manejar los archivos asignados a memoria?
22. Qué métodos se han agregado para la clase String y StringBuilder respectivamente para facilitar el manejo de cadenas?
23. Cuál es el espacio de nombres para trabajar con programación paralela?
24. Menciona 3 mejoras en ASP .NET.
25. Cuáles son los nuevos controles que han aparecido en WPF?
26. Cuál es la ventaja principal de usar el Visual Studio 2010 en el desarrollo de una Aplicación?
27. Menciona 3 ventanas del IDE de Visual Studio 2010.
28. Menciona 3 novedades del Editor de Visual Studio 2010.
29. Con qué otros productos Microsoft se integra el desarrollo desde Visual Studio 2010?
30. Cuáles son las estructuras de decisión de Visual Basic .NET?
31. Cuáles son las estructuras de bucles de Visual Basic .NET?

32. Qué estructura adicional de Visual Basic simplifica la sintaxis y el acceso a un objeto?
33. Menciona 5 tipos de datos simples en Visual Basic.
34. Menciona 5 operadores aritméticos de Visual Basic?
35. Qué operador es más rápido para sumar 2 cadenas: "+" o "&".
36. Cuáles son los operadores lógicos de corto circuito en Visual Basic?
37. Qué tipo de variable debe usarse en lo posible en Visual Basic?
38. Menciona 3 novedades de Visual Basic 2010.

Capítulo 2: Programando con Visual Basic .NET 2010

En este segundo capítulo veremos cómo usar el lenguaje Visual Basic .NET 2010 para obtener información del entorno, acceder a las carpetas especiales de Windows, acceder al registro de Windows, trabajar con procesos o tareas, manejar los recursos de entrada y salida tales como unidades, directorios, rutas y archivos. También veremos cómo programar orientado a objetos creando y usando una librería de clases.

Además se verán las diferentes técnicas que hay en .NET para implementar programación asíncrona tales como Threads, Callbacks, Eventos Asíncronos y Programación Paralela.

Al final del capítulo se verá el tema de asegurar los datos de la aplicación mediante la encriptación o el cifrado, ya sea simétrico o asimétrico, también como realizar comprobaciones mediante resúmenes o valores hash.

1. Trabajar con el Entorno de Windows y el Sistema

Para trabajar con el entorno de Windows existe una clase llamada **Environment** que se encuentra en el espacio de nombres System, la cual nos brinda información del sistema y si deseamos acceder al registro de Windows ya sea para leer o escribir podemos usar la clase **Registry** que pertenece al espacio de nombres Microsoft.Win32, además podemos obtener los procesos o tareas ejecutándose en el sistema, abrir y cerrar tareas mediante la clase **Process** de System.Diagnostics.

1.1. Obtener Información del Sistema

La clase Environment tiene algunas propiedades que nos dan información del sistema, tales como: MachineName, OSVersion, ProcessorCount, UserDomainName, UserInteractive y UserName.

En .NET Framework 4 la clase Environment tiene 2 nuevas propiedades: is64BitOperatingSystem y is64BitProcess que dan información sobre Sistema y Procesos de 62 bits respectivamente.

Iniciamos nuestro primer ejemplo del libro con una aplicación Windows Forms que muestre información del sistema.

Demo 01

Abra el Visual Studio 2010 y realice los siguientes pasos:

- Del menu "File", seleccionar "New Project".
- En el diálogo seleccionar como lenguaje "Visual Basic" y luego "Windows".
- Seleccionar la plantilla "Windows Forms Application".
- Escribir como nombre de la aplicación: Demo01 y botón "OK".
- Cambiar de nombre al formulario de form1.vb a frmInfoSistema.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmInfoSistema FixedSingle False False 300, 300 CenterScreen Información del Sistema
Label1	Name AutoSize Location Text	lblNombrePC True 12,22 Nombre de la PC:
TextBox1	Name Location ReadOnly Size	txtNombrePC 118, 15 True 141, 20
Label2	Name AutoSize Location Text	lblVersionSO True 12,61 Versión del Sistema:
TextBox2	Name Location ReadOnly Size	txtVersionSO 118, 54 True 141, 20
Label3	Name AutoSize Location Text	lblNroProcesadores True 12,100 Nro Procesadores:
TextBox3	Name Location ReadOnly Size	txtNroProcesadores 118, 93 True 141, 20
Label4	Name AutoSize Location Text	lblNombreUsuario True 12,139 Nombre Usuario:
TextBox4	Name Location ReadOnly Size	txtNombreUsuario 118, 132 True 141, 20
Label5	Name AutoSize	lblSistema64Bits True

	Location Text	12,178 Sistema 64 Bits:
TextBox5	Name Location ReadOnly Size	txtSistema64Bits 118, 171 True 141, 20
Label6	Name AutoSize Location Text	lblProceso64Bits True 12,217 Proceso 64 Bits:
TextBox6	Name Location ReadOnly Size	txtProceso64Bits 118, 210 True 141, 20

- El diseño del formulario debe quedar similar al gráfico 2.1:

Gráfico 2.1: Diseño del formulario de información del sistema

The image shows a Windows application window titled "Información del Sistema". Inside the window, there are six text boxes arranged vertically, each preceded by a label. The labels are: "Nombre de la PC:", "Versión del Sistema:", "Nro Procesadores:", "Nombre Usuario:", "Sistema 64 Bits:", and "Proceso 64 Bits:". Each text box is empty, indicating that the data has not yet been populated or is being shown in a template state.

- Ingresar al editor de código (cuarto botón del explorador de soluciones) y escribir el siguiente código.

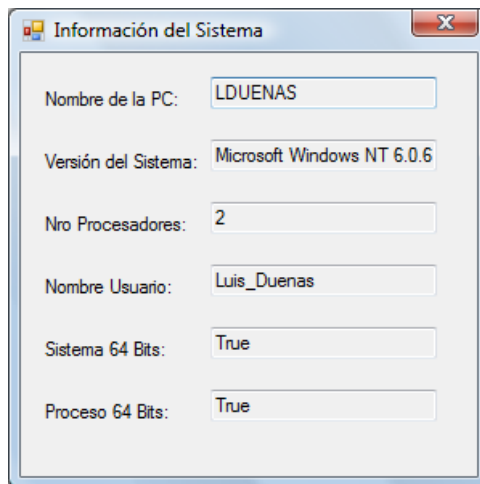
```
Private Sub MostrarInformacionSistema(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    txtNombrePC.Text = Environment.MachineName
    txtVersionSO.Text = Environment.OSVersion.VersionString
```



```
txtNroProcesadores.Text = Environment.ProcessorCount  
txtNombreUsuario.Text = Environment.UserName  
txtSistema64Bits.Text = Environment.Is64BitOperatingSystem  
txtProceso64Bits.Text = Environment.Is64BitProcess  
End Sub
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.2: Ejecución del formulario información del sistema



1.2. Obtener Directorios Especiales de Windows

La clase Environment tiene una enumeración llamada SpecialFolders que tiene una lista con los nombres de todas las carpetas o directorios especiales de Windows, si deseamos obtener la ruta o ubicación en el sistema de archivos (File System) de una carpeta se usa el método GetFolderPath.

A continuación crearemos una aplicación que lista los nombres y rutas de las carpetas especiales de Windows, las cuales han aumentado con la versión 4 del .NET Framework.

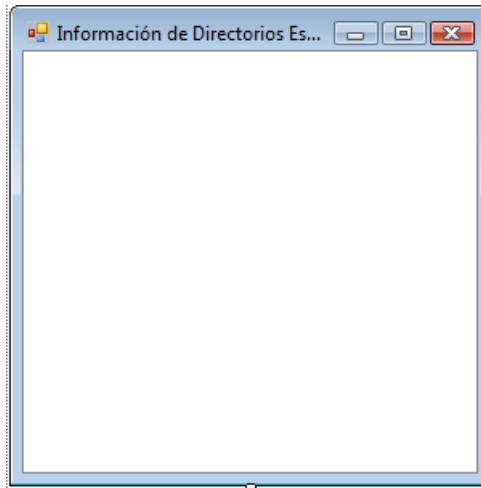
Demo 02

- Crear una aplicación Windows Forms en Visual Basic llamada: Demo02.
- Cambiar de nombre al formulario de form1.vb a frmDirectWindows.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmDirectWindows
	Text	Información Directorios Window
	WindowState	Maximized
ListView1	Name	lvwDirectorio
	Dock	Fill
	FullRowSelect	True
	GridLines	True
	HotTracking	True

- El diseño del formulario debe quedar similar al gráfico 2.3:

Gráfico 2.3: Diseño del formulario con directorios de Windows



- Ingresar al editor de código y escribir el siguiente código.

```
Private Sub ListarDirectoriosWindows(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    With lvwDirectorio
```

```
        .Columns.Add("Nombre", 300, HorizontalAlignment.Left)
```

```
        .Columns.Add("Ruta", 600, HorizontalAlignment.Left)
```

```
        .View = View.Details
```

```
    End With
```

```
    Dim Tipo As Object = GetType(Environment.SpecialFolder)
```

```
    Dim Directorios() As String = [Enum].GetNames(Tipo)
```

```
    Dim Folder As Environment.SpecialFolder
```

```
    Dim Fila As ListViewItem
```

```
    For Each Directorio As String In Directorios
```

```
        Folder = [Enum].Parse(Tipo, Directorio)
```

```
        Fila = lvwDirectorio.Items.Add(Directorio)
```

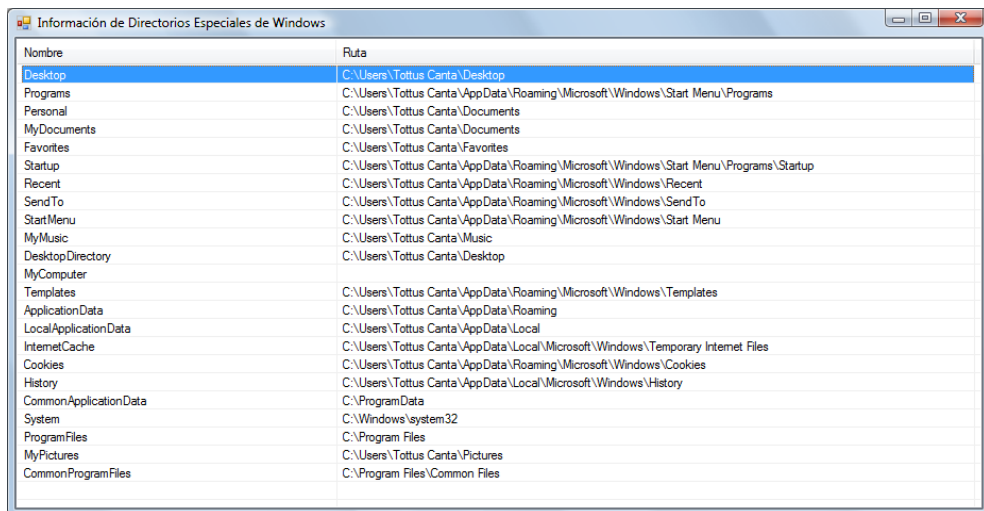
```
        Fila.SubItems.Add(Environment.GetFolderPath(Folder))
```

```
    Next
```

```
End Sub
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.4: Ejecución del formulario con directorios de Windows



Información de Directorios Especiales de Windows

Nombre	Ruta
Desktop	C:\Users\Tottus Canta\Desktop
Programs	C:\Users\Tottus Canta\AppData\Roaming\Microsoft\Windows\Start Menu\Programs
Personal	C:\Users\Tottus Canta\Documents
MyDocuments	C:\Users\Tottus Canta\Documents
Favorites	C:\Users\Tottus Canta\Favorites
Startup	C:\Users\Tottus Canta\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
Recent	C:\Users\Tottus Canta\AppData\Roaming\Microsoft\Windows\Recent
SendTo	C:\Users\Tottus Canta\AppData\Roaming\Microsoft\Windows\SendTo
Start Menu	C:\Users\Tottus Canta\AppData\Roaming\Microsoft\Windows\Start Menu
MyMusic	C:\Users\Tottus Canta\Music
DesktopDirectory	C:\Users\Tottus Canta\Desktop
MyComputer	
Templates	C:\Users\Tottus Canta\AppData\Roaming\Microsoft\Windows\Templates
ApplicationData	C:\Users\Tottus Canta\AppData\Roaming
LocalApplicationData	C:\Users\Tottus Canta\AppData\Local
InternetCache	C:\Users\Tottus Canta\AppData\Local\Microsoft\Windows\Temporary Internet Files
Cookies	C:\Users\Tottus Canta\AppData\Roaming\Microsoft\Windows\Cookies
History	C:\Users\Tottus Canta\AppData\Local\Microsoft\Windows\History
CommonApplicationData	C:\ProgramData
System	C:\Windows\system32
ProgramFiles	C:\Program Files
MyPictures	C:\Users\Tottus Canta\Pictures
CommonProgramFiles	C:\Program Files\Common Files

1.3. Acceder al Registro de Windows

La mayoría de programas de Windows guardan su configuración en el Registro de Windows, el cual se puede ver usando la utilidad Regedit.exe, desde .NET también podemos acceder a una sección del registro usando la clase Registry y creando un RegistryKey, luego usamos los métodos GetSubKeyNames para obtener todas las secciones y OpenSubKey para abrir una sección, tal como se describe en el siguiente ejemplo.

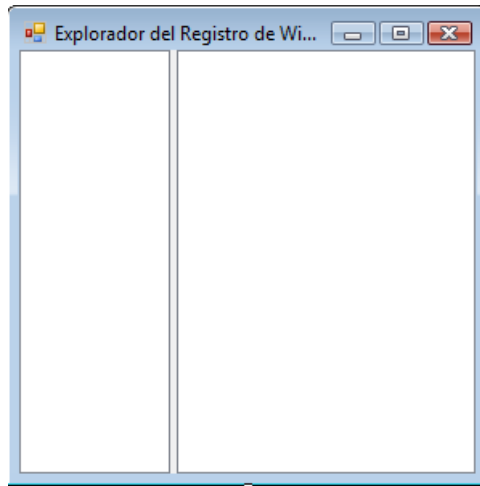
Demo 03

- Crear una aplicación Windows Forms en Visual Basic llamada: Demo03.
- Cambiar de nombre al formulario form1.vb por frmRegistroWindows.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name Text WindowState	frmRegistroWindows Explorador Registro de Windows Maximized
SplitContainer1	Name Dock	scRegistro Fill
TreeView1 (izquierda)	Name Dock	twvRegistro Fill
ListView1 (derecha)	Name Dock	lvwRegistro Fill

- El diseño del formulario debe quedar similar al gráfico 2.5:

Gráfico 2.5: Diseño del formulario registro de Windows



- Ingresar al editor de código y escribir el siguiente código.

```
Imports Microsoft.Win32
```

```
Imports System.Security.Permissions
```

```
<RegistryPermission(SecurityAction.Demand)> _
```

```
Public Class frmRegistroWindows
```

```
    Private Sub CrearNodos(ByVal nodo As TreeNode, ByVal reg As RegistryKey)
```

```
        Dim secciones() As String = reg.GetSubKeyNames
```

```
        Dim seccion As String
```

```
        Dim nuevoNodo As TreeNode
```

```
        Dim nuevoReg As RegistryKey
```

```
        For Each seccion In secciones
```

```
            nuevoNodo = nodo.Nodes.Add(seccion)
```

```
            nuevoReg = reg.OpenSubKey(seccion)
```

```
            If nuevoReg IsNot Nothing Then
```

```
                CrearNodos(nuevoNodo, nuevoReg)
```

```
            End If
```

```
        Next
```

```
    End Sub
```

```
    Private Sub ListarRegistro(ByVal sender As System.Object, _
```

```
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        With lvwRegistro
```

```
            .Columns.Add("Nombre", 100, HorizontalAlignment.Left)
```

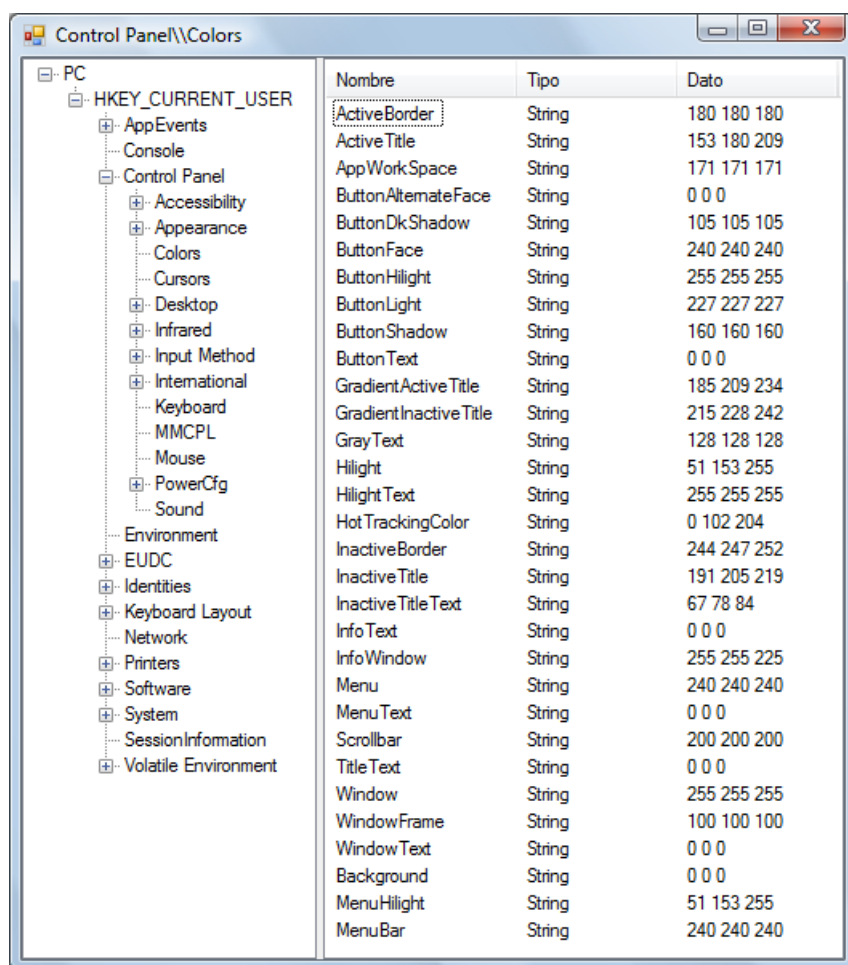
```
.Columns.Add("Tipo", 100, HorizontalAlignment.Left)
.Columns.Add("Dato", 100, HorizontalAlignment.Left)
.View = View.Details
End With
Dim nodoPC As TreeNode = twwRegistro.Nodes.Add("PC")
Dim nodoHKCU As TreeNode = _
nodoPC.Nodes.Add("HKEY_CURRENT_USER")
Dim reg As RegistryKey = Registry.CurrentUser
CrearNodos(nodoHKCU, reg)
End Sub
```

```
Private Function ObtenerRuta(ByVal e As TreeNode) As String
    Dim S As String = e.Text
    If e.Level > 2 Then
        S = ObtenerRuta(e.Parent) + "\\\" + S
    End If
    Return (S)
End Function
```

```
Private Sub MostrarValores(ByVal sender As System.Object, ByVal e As _
System.Windows.Forms.TreeViewEventArgs) Handles twwRegistro.AfterSelect
    lvwRegistro.Items.Clear()
    If e.Node.Level > 0 Then
        Dim reg As RegistryKey = Registry.CurrentUser
        Dim seccion As String = ObtenerRuta(e.Node)
        Me.Text = seccion
        Dim registro As RegistryKey = reg.OpenSubKey(seccion)
        If registro IsNot Nothing Then
            Dim claves() As String = registro.GetValueNames
            Dim clave As String
            Dim fila As ListViewItem
            For Each clave In claves
                fila = lvwRegistro.Items.Add(clave)
                fila.SubItems.Add(registro.GetValueKind(clave).ToString)
                fila.SubItems.Add(registro.GetValue(clave))
            Next
        End If
    End If
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.6: Ejecución del formulario registro de Windows



The screenshot shows the 'Control Panel\Colors' window. On the left is a tree view of system settings, with 'Colors' expanded. On the right is a table listing various system color names, their data type (all 'String'), and their RGB values.

Nombre	Tipo	Dato
ActiveBorder	String	180 180 180
Active Title	String	153 180 209
AppWorkSpace	String	171 171 171
ButtonAlternateFace	String	0 0 0
ButtonDkShadow	String	105 105 105
ButtonFace	String	240 240 240
ButtonHighlight	String	255 255 255
ButtonLight	String	227 227 227
ButtonShadow	String	160 160 160
ButtonText	String	0 0 0
GradientActive Title	String	185 209 234
GradientInactive Title	String	215 228 242
GrayText	String	128 128 128
Highlight	String	51 153 255
Highlight Text	String	255 255 255
HotTrackingColor	String	0 102 204
InactiveBorder	String	244 247 252
Inactive Title	String	191 205 219
Inactive Title Text	String	67 78 84
InfoText	String	0 0 0
InfoWindow	String	255 255 225
Menu	String	240 240 240
Menu Text	String	0 0 0
Scrollbar	String	200 200 200
TitleText	String	0 0 0
Window	String	255 255 255
WindowFrame	String	100 100 100
WindowText	String	0 0 0
Background	String	0 0 0
MenuHighlight	String	51 153 255
MenuBar	String	240 240 240

1.4. Trabajar con Procesos o Tareas del Sistema

Para trabajar con procesos o tareas del sistema, se usa la clase Process del espacio de nombres System.Diagnostics, la cual tiene métodos como GetProcesses que devuelve una lista con los nombres de todos los procesos que se encuentran actualmente en ejecución, también podemos detectar todos los procesos por nombre de un programa usando el método GetProcessesByName. Para iniciar el proceso (ejecutar el programa) podemos usar el método Start y para cerrarlo podemos usar los métodos Close o Kill, tal como se muestra en el siguiente ejemplo.

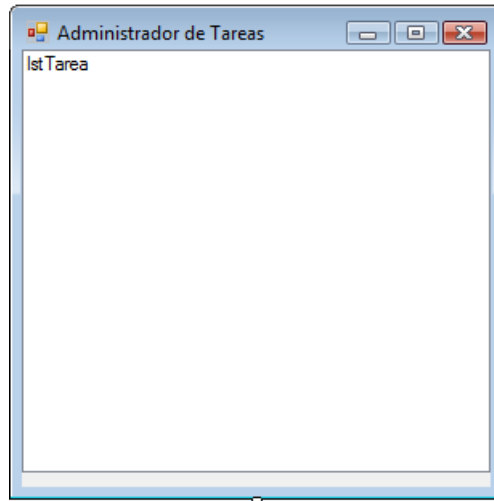
Demo 04

- Crear una aplicación Windows Forms en Visual Basic llamada: Demo04.
- Cambiar de nombre al formulario form1.vb por frmListarTareas.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListarTarea
	Size	300,300
	StartPosition	CenterScreen
	Text	Administrador de Tareas
ListBox1	Name	IstTarea
	Dock	Fill
ContextMenuStrip1	Name	mnuTarea
	Item1	mnuEjecutarTarea
	Item2	mnuCerrarTarea
Timer1	Name	tmrRefrescarTarea
	Enabled	True
	Interval	5000

- El diseño del primer formulario debe quedar similar al gráfico 2.7:

Gráfico 2.7: Diseño del formulario listar tareas



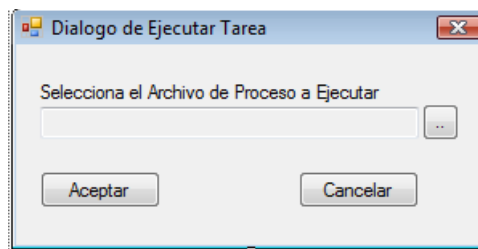
- Agregar otro formulario, del menú "Project" seleccionar "Add Windows Form", escribir como nombre: frmEjecutarTarea.vb.
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form2	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmEjecutarTarea FixedSingle False False 300,150 CenterScreen Dialogo de Ejecutar Tarea
Label1	Name AutoSize Location Text	lblArchivo True 12, 23 Selecciona Archivo de Proceso a Ejecutar
TextBox1	Name Location Size	txtArchivo 15,39 236,20
Button1	Name Cursor Location Size	btnMostrarDialogoAbrir Hand 253,37 23,23

	Text	...
Button2	Name Cursor DialogResult Location Size Text	btnAceptar Hand OK 15,79 75,23 Aceptar
Button3	Name Cursor DialogResult Location Size Text	btn Cancelar Hand Cancel 176,79 75,23 Cancelar

- El diseño del segundo formulario debe quedar similar al gráfico 2.8:

Gráfico 2.8: Diseño del formulario ejecutar tarea



- Ingresar al editor de código y escribir el siguiente código para el primer formulario.

```
Imports System.Diagnostics 'Process
```

```
Public Class frmListarTarea
```

```
    Private Sub ListarTareas()
```

```
        lstTarea.Items.Clear()
```

```
        Dim oProcesos() As Process = Process.GetProcesses
```

```
        Dim oProceso As Process
```

```
        lstTarea.BeginUpdate()
```

```
        For Each oProceso In oProcesos
```

```
            lstTarea.Items.Add(oProceso.ProcessName)
```

```
        Next
```

```
        lstTarea.EndUpdate()
```

End Sub

```
Private Sub CargarDatos(ByVal sender As System.Object, ByVal e As _  
System.EventArgs) Handles MyBase.Load
```

```
    lstTarea.Sorted = True
```

```
    ListarTareas()
```

End Sub

```
Private Sub EjecutarTarea(ByVal sender As System.Object, ByVal e As _  
System.EventArgs) Handles mnuEjecutarTarea.Click
```

```
    If frmEjecutarTarea.ShowDialog = Windows.Forms.DialogResult.OK _
```

```
    AndAlso frmEjecutarTarea.txtArchivo.Text <> "" Then
```

```
        Process.Start(frmEjecutarTarea.txtArchivo.Text)
```

```
        ListarTareas()
```

```
    End If
```

End Sub

```
Private Sub CerrarTarea(ByVal sender As Object, ByVal e As _  
System.EventArgs) Handles mnuCerrarTarea.Click
```

```
    Dim oProcesos() As Process = _
```

```
    Process.GetProcessesByName(lstTarea.Text)
```

```
    If oProcesos.Length > 0 Then
```

```
        Dim oProceso As Process
```

```
        For Each oProceso In oProcesos
```

```
            oProceso.Kill()
```

```
        Next
```

```
        ListarTareas()
```

```
    End If
```

End Sub

```
Private Sub RefrescarTarea(ByVal sender As System.Object, ByVal e As _  
System.EventArgs) Handles tmrRefrescarTarea.Tick
```

```
    ListarTareas()
```

End Sub

End Class

- Ahora escribir el siguiente código para el segundo formulario.

```
Imports System.Windows.Forms 'OpenFileDialog
```

```
Public Class frmEjecutarTarea
```

```
Private Sub MostrarDialogoAbrir(ByVal sender As System.Object, ByVal e _  
As System.EventArgs) Handles btnMostrarDialogoAbrir.Click  
    Dim ofd As New OpenFileDialog  
    ofd.Title = "Selecciona Programa a Ejecutar"  
    ofd.Filter = "Archivos de programa|*.exe"  
    If ofd.ShowDialog = Windows.Forms.DialogResult.OK Then  
        txtArchivo.Text = ofd.FileName  
    End If  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.9: Ejecución del formulario listar tareas

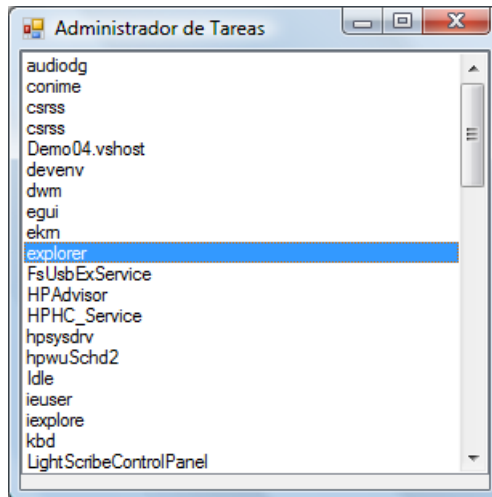
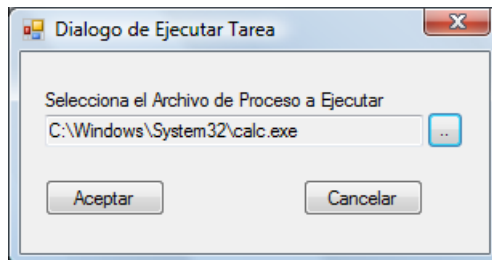


Gráfico 2.10: Ejecución del formulario ejecutar tarea



2. Manejo de Entrada y Salida

.NET Framework tiene una amplia variedad de clases que permiten controlar los recursos de entrada y salida, las cuales en su mayoría se encuentran en el espacio de nombres **System.IO**.

Hay clases que se usan para entrada y salida de archivos como Directory y DirectoryInfo, DriveInfo, File y FileInfo, FileStream, FileSystemInfo, Path, DeflateStream, GZipStream y SerialPort.

Otras clases se usan para leer y escribir secuencias o flujos, tales como BinaryReader y BinaryWriter, StreamReader y StreamWriter, StringReader y StringWriter, TextReader y TextWriter.

Finalmente, hay clases comunes de secuencias que son usadas para diversos propósitos como: BufferedStream, CryptoStream, MemoryStream y NetworkStream.

Para obtener más información sobre este tema ver referencia 13 al final del libro.

En .NET existen 2 formas de usar un miembro de una clase, creando una instancia de la clase y llamando al miembro o usando un miembro compartido, por ejemplo las clases DirectoryInfo, DriveInfo, FileInfo se tienen que crear objetos para usar sus métodos, pero también podemos hacerlo directamente a través de las clases con miembros estáticos o compartidos Directory, File, Path, etc.

2.1 Manejar Unidades, Directorios, Rutas y Archivos

Para manejar unidades debemos usar la clase DriveInfo, para manejar directorios debemos usar la clase con miembros estáticos Directory o instanciar la clase DirectoryInfo.

Para manejar Rutas de archivos se usa la clase con miembros estáticos Path y para obtener información de un Archivo se usa la clase FileInfo.

A continuación presentamos un ejemplo sobre una aplicación que explora directorios de una cierta unidad seleccionada y a la vez muestra archivos.

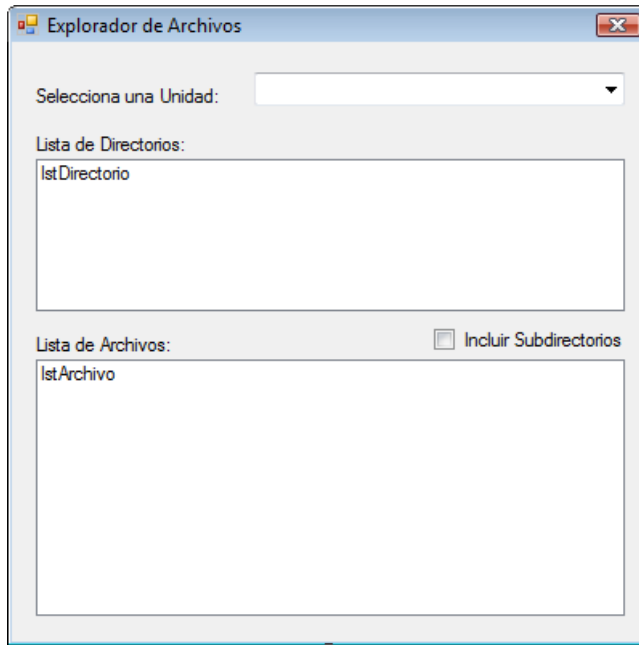
Demo 05

- Crear una aplicación Windows Forms en Visual Basic llamada: Demo05.
- Cambiar de nombre al formulario de form1.vb a frmExplorador.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmExploradorArchivos FixedSingle False False 400, 400 CenterScreen Explorador de Archivos
Label1	Name AutoSize Location Text	lblUnidad True 12,27 Selecciona una Unidad:
ComboBox1	Name Location Size	cboUnidad 151, 19 231, 21
Label2	Name AutoSize Location Text	lblDirectorio True 12,57 Lista de Directorios:
ListBox1	Name Location Size	lstDirectorio 15, 73 367, 95
Label3	Name AutoSize Location Text	lblArchivo True 12,182 Lista de Archivos:
CheckBox1	Name AutoSize Location Text	chkIncluirSubdirectorios True 263, 178 Incluir Subdirectorios
ListBox2	Name Location Size	lstArchivo 15, 198 367, 160

- El diseño del formulario debe quedar similar al gráfico 2.11:

Gráfico 2.11: Diseño del formulario Explorador de Archivos



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO
```

```
Public Class frmExploradorArchivos
```

```
    Private Sub ListarUnidades(ByVal sender As System.Object, _  
        ByVal e As System.EventArgs) Handles MyBase.Load  
        Dim Unidades() As DriveInfo = DriveInfo.GetDrives()  
        Dim Unidad As DriveInfo  
        cboUnidad.Items.Clear()  
        For Each Unidad In Unidades  
            cboUnidad.Items.Add(Unidad.Name)  
        Next  
    End Sub
```

```
    Private Sub ListarDirectorios(ByVal sender As System.Object, _  
        ByVal e As System.EventArgs) Handles cboUnidad.SelectedIndexChanged
```



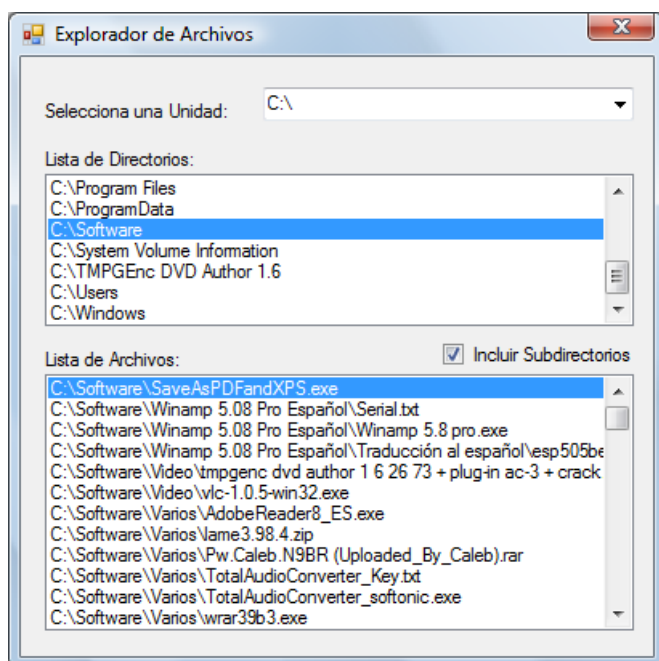
```
Try
    Dim Directorios As IEnumerable = _
        Directory.EnumerateDirectories(cboUnidad.Text)
    Dim bs As New BindingSource
    bs.DataSource = Directorios
    lstDirectorio.DataSource = bs
Catch ex As Exception
    MessageBox.Show(ex.Message)
End Try
End Sub

Private Sub ListarArchivos(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles lstDirectorio.SelectedIndexChanged
    Try
        Dim N As Integer = If(chkIncluirSubdirectorios.Checked, 1, 0)
        Dim Archivos As IEnumerable = _
            Directory.EnumerateFiles(lstDirectorio.SelectedItem, "*.*", N)
        Dim bs As New BindingSource
        bs.DataSource = Nothing
        bs.DataSource = Archivos
        lstArchivo.DataSource = bs
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End Sub

Private Sub MostrarArchivosSubdirectorios(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles _
    chkIncluirSubdirectorios.CheckedChanged
    ListarArchivos(Nothing, Nothing)
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.12: Ejecución del formulario Explorador de Archivos



2.2. Leer y Escribir en un Archivo

Para leer y escribir archivos en .NET existen muchas formas, por ejemplo para leer secuencialmente archivos de texto podemos usar la clase StreamReader y para escribir se usa la clase StreamWriter. Si deseamos leer y escribir archivos binarios entonces debemos usar BinaryReader y BinaryWriter.

La clase FileStream es la clase principal con la cual podemos especificar el modo de trabajo con el archivo: lectura, escritura, adición; el tipo de acceso: lectura o escritura y si estará compartido para leer o escribir. Esta debe usarse en conjunto con las otras clases mencionadas.

En el siguiente ejemplo crearemos una aplicación Chat en la cual si el archivo que guarda los mensajes esta compartido en un directorio de red, entonces todos los usuarios podrían ver mensajes de los demás, ya que el archivo estará disponible para todos.

Demo 06

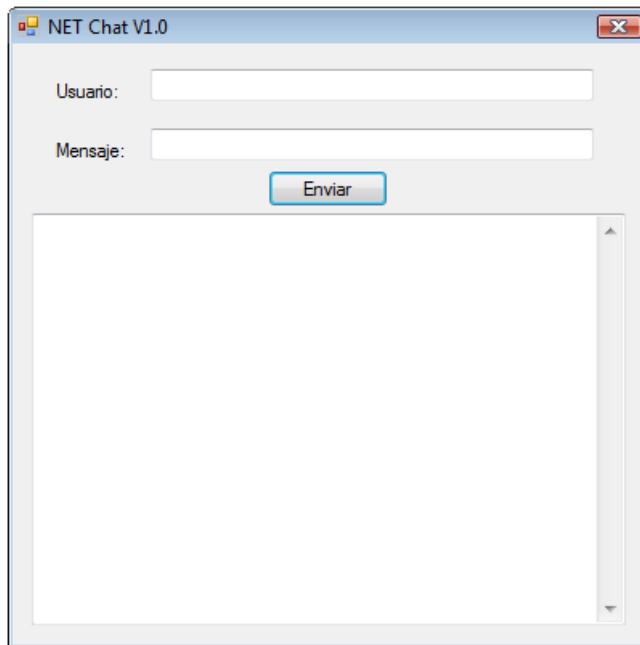
- Crear una aplicación Windows Forms en Visual Basic llamada: Demo06.
- Cambiar de nombre al formulario de form1.vb a frmChat.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmChat
	AcceptButton	btnEnviar
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400, 400
	StartPosition	CenterScreen
Label1	Text	NET Chat V1.0
	Name	lblUsuario
	AutoSize	True
	Location	24,23
TextBox1	Text	Usuario:
	Name	txtUsuario
	Location	86,16

	Size	276,20
Label2	Name AutoSize Location Text	lblMensaje True 24,60 Mensaje:
TextBox2	Name Location Size	txtMensaje 86,53 276,20
Button1	Name Cursor Location Size Text	btnEnviar Hand 159,79 75,23 Enviar
TextBox3	Name Location Multiline ScrollBars Size	txtPanel 12,106 True Vertical 370,257

- El diseño del formulario debe quedar similar al gráfico 2.13:

Gráfico 2.13: Diseño del formulario Chat



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO
```

```
Public Class frmChat
```

```
    Private Archivo As String = "Chat.txt"
```

```
    Private WithEvents tmrRefrescar As New Timer
```

```
    Private Sub MostrarMensajes()
```

```
        If File.Exists(Archivo) Then
```

```
            Using fs As New FileStream(Archivo, FileMode.Open, FileAccess.Read, FileShare.Read)
```

```
                Using sr As New StreamReader(fs)
```

```
                    txtPanel.Text = sr.ReadToEnd
```

```
                End Using
```

```
            End Using
```

```
        End If
```

```
        txtPanel.SelectionLength = txtPanel.Text.Length
```

```
        txtPanel.ScrollToCaret()
```

```
    End Sub
```

```
    Private Sub EnviarMensaje(ByVal sender As System.Object, ByVal e As _  
System.EventArgs) Handles btnEnviar.Click
```

```
        Using fs As New FileStream(Archivo, FileMode.Append, FileAccess.Write, FileShare.Write)
```

```
            Using sw As New StreamWriter(fs)
```

```
                sw.WriteLine("Fecha y Hora: {0}", Now.ToString)
```

```
                sw.WriteLine("Usuario: {0}", txtUsuario.Text)
```

```
                sw.WriteLine("Dice: {0}", txtMensaje.Text)
```

```
                sw.WriteLine(New String("_", 50))
```

```
            End Using
```

```
        End Using
```

```
        MostrarMensajes()
```

```
    End Sub
```

```
    Private Sub IniciarTimer(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles MyBase.Load
```

```
        txtUsuario.Text = Environment.UserName
```

```
        tmrRefrescar.Interval = 5000
```

```
        tmrRefrescar.Enabled = True
```

```
        tmrRefrescar.Start()
```

End Sub

Private Sub tmrRefrescar_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrRefrescar.Tick

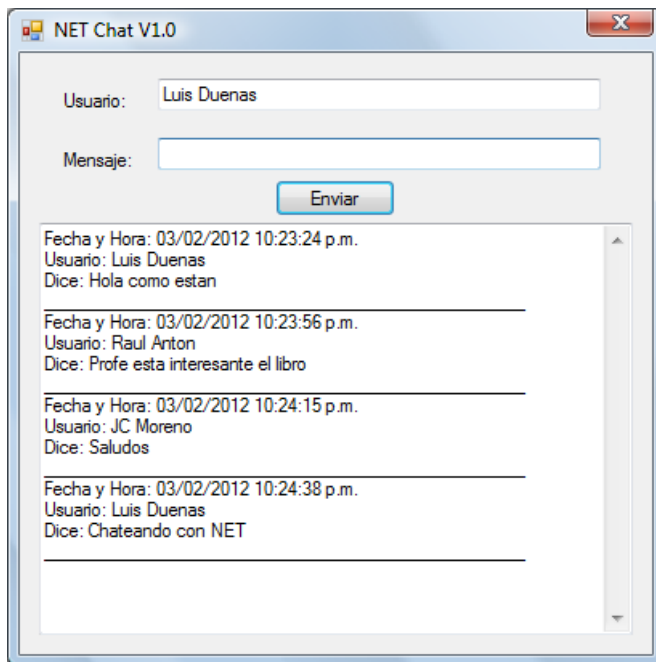
MostrarMensajes()

End Sub

End Class

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.14: Ejecución del formulario Chat



2.3. Dividir y Unir Archivos

En muchas ocasiones se necesita partir un archivo muy grande en varios archivos más pequeños por ejemplo para enviarlos por mail y en caso contrario al recibir los archivos separados debemos unirlos en uno solo.

Para dividir el archivo necesitamos averiguar su tamaño usando la propiedad Length de la clase FileInfo, luego calculamos la cantidad de partes en que se va a dividir e iniciamos la lectura del archivo escribiéndolo en un directorio con el nombre del archivo los diferentes archivos obtenidos usando para ambos casos la clase FileStream.

Finalmente, para unir todos los archivos del directorio se debe crear un flujo de escritura e ir leyendo todos los archivos y escribirlos en dicho flujo también con la clase FileStream.

Además en este ejemplo se hace uso de la clase Path para devolver el nombre del directorio: GetDirectoryName, el nombre del archivo con y sin extensión: GetFileName y GetFileNameWithoutExtension.

Demo 07

- Crear una aplicación Windows Forms en Visual Basic llamada: Demo07.
- Cambiar de nombre al formulario de form1.vb a frmDivideUne.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

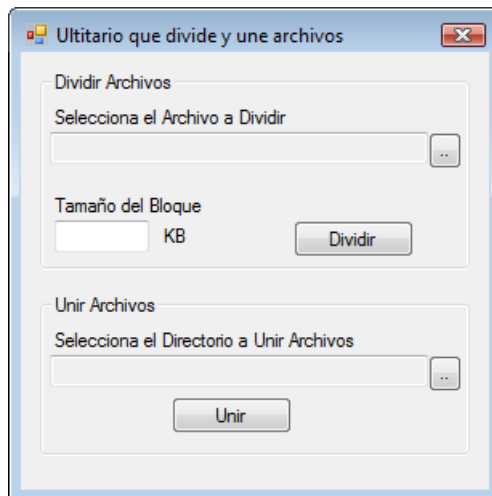
Objeto	Propiedad	Valor
Form1	Name	frmDivideUne
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	300, 300
	StartPosition	CenterScreen
	Text	Ultitario que divide y une archiv.
GroupBox1	Name	grpDividir
	Location	12,12
	Size	268,124
	Text	Dividir Archivos
Label1	Name	lblArchivo

	AutoSize Location Text	True 6,22 Selecciona el Archivo a Dividir
TextBox1	Name Location Size	txtArchivo 6,38 236,20
Button1	Name Cursor Location Size Text	btnAbrir Hand 242,38 21,23 ...
Label2	Name AutoSize Location Text	lblTamaño True 6,77 Tamaño del Bloque
TextBox3	Name Location Size	txtTamaño 9,93 59,20
Label3	Name AutoSize Location Text	lblKB True 74,96 KB
Button2	Name Cursor Enabled Location Size Text	btnDividir Hand False 158,93 75,23 Dividir
GroupBox2	Name Location Size Text	grpUnir 12,151 268, 101 Unir Archivos
Label4	Name AutoSize Location Text	lblDirectorio True 6,22 Selecciona el Directorio a Unir Archivos
TextBox4	Name Location Size	txtDirectorio 6,38 236,20
Button3	Name Cursor	btnDirectorio Hand

	Location Size Text	242,38 21,23 ...
Button4	Name Cursor Enabled Location Size Text	btnUnir Hand False 82,64 75,23 Unir

- El diseño del formulario debe quedar similar al gráfico 2.15:

Gráfico 2.15: Diseño del formulario que divide y une archivos



- Ingresar al editor de código y escribir el siguiente código.

Imports System.IO 'FileInfo, File, Path

Public Class frmDivideUne

Private Sub MostrarDialogoAbrir(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAbrir.Click

Dim ofd As New OpenFileDialog

With ofd

.Title = "Selecciona el Archivo a dividir"

.Filter = "Todos los archivos|*.*"

If .ShowDialog = Windows.Forms.DialogResult.OK Then

```
        txtArchivo.Text = .FileName
    End If
End With
End Sub
```

```
Private Sub ValidarDigitos(ByVal sender As Object, _
    ByVal e As System.Windows.Forms.KeyPressEventArgs) _
    Handles txtTamaño.KeyPress
    e.Handled = Not (Char.IsDigit(e.KeyChar) Or e.KeyChar = _
        ControlChars.Back)
End Sub
```

```
Private Sub HabilitarBotonDividir(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles txtArchivo.TextChanged, _
    txtTamaño.TextChanged
    btnDividir.Enabled = (txtArchivo.Text <> "" AndAlso _
        txtTamaño.Text <> "")
End Sub
```

```
Private Sub DividirArchivo(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnDividir.Click
    Dim fi As New FileInfo(txtArchivo.Text)
    Dim TT As Long = fi.Length
    Dim TB As Long = Long.Parse(txtTamaño.Text) * 1024
    If TT > TB Then
        Dim NB As Byte = TT \ TB
        Dim TUB As Long = TT Mod TB
        Dim I, J As Integer
        Dim Carpeta As String = _
            Path.GetFileNameWithoutExtension(txtArchivo.Text)
        Dim Ruta As String = Path.GetDirectoryName(txtArchivo.Text)
        Dim Directorio As String = Ruta & "\" & Carpeta
        Directory.CreateDirectory(Directorio)
        Dim Archivo As String = Directorio & "\" & _
            Path.GetFileName(txtArchivo.Text)
        Dim Buffer(TB) As Byte
        Using fsR As New FileStream(txtArchivo.Text, FileMode.Open, _
            FileAccess.Read, FileShare.Read)
            For I = 0 To NB - 1
                Using fsW As New FileStream(Directorio & "\" & _
```

```
Path.GetFileNameWithoutExtension(Archivo) & _
I.ToString.PadLeft(3, "0"), FileMode.Create, FileAccess.Write, _
FileShare.Write)
    For J = 0 To TB - 1
        fsW.WriteByte(fsR.ReadByte)
    Next
End Using
Next
If TUB <> 0 Then
    Using fsW As New FileStream(Directorio & "\" & _
Path.GetFileNameWithoutExtension(Archivo) & _
I.ToString.PadLeft(3, "0"), FileMode.Create, FileAccess.Write, _
FileShare.Write)
        For J = 0 To TUB - 1
            fsW.WriteByte(fsR.ReadByte)
        Next
    End Using
End If
End Using
MessageBox.Show("Archivos fueron divididos", "Aviso", _
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
Else
    MessageBox.Show("Tamaño del Bloque es mayor al Archivo", "Aviso",
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
End If
End Sub

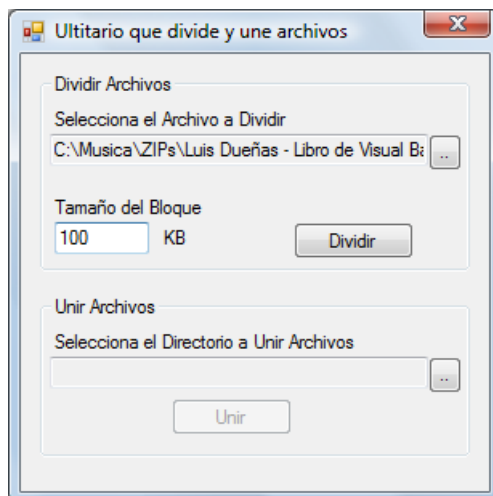
Private Sub SeleccionarDirectorio(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnDirectorio.Click
    Dim fbd As New FolderBrowserDialog
    fbd.SelectedPath = "C:\Lduenas\NET\LibroVB2010\"
    If fbd.ShowDialog = Windows.Forms.DialogResult.OK Then
        txtDirectorio.Text = fbd.SelectedPath
    End If
End Sub

Private Sub HabilitarBotonUnir(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles txtDirectorio.TextChanged
    btnUnir.Enabled = txtDirectorio.Text <> ""
End Sub
```

```
Private Sub UnirArchivos(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnUnir.Click
    Dim Directorio As String = txtDirectorio.Text
    Dim Archivos() As String = Directory.GetFiles(Directorio)
    Dim fi As FileInfo
    Dim Archivo As String = Path.GetDirectoryName(Directorio) & "\" & _
Path.GetFileName(Directorio) & ".txt"
    Using fsW As New FileStream(Archivo, FileMode.Create, FileAccess.Write,
    FileShare.Write)
        Dim I As Integer
        For I = 0 To Archivos.Length - 1
            Using fsR As New FileStream(Archivos(I), FileMode.Open, _
            FileAccess.Read, FileShare.Read)
                fi = New FileInfo(Archivos(I))
                Dim Buffer(fi.Length - 1) As Byte
                fsR.Read(Buffer, 0, Buffer.Length)
                fsW.Write(Buffer, 0, Buffer.Length)
            End Using
        Next
    End Using
    MessageBox.Show("Archivos fueron unidos", "Aviso", _
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.16: Ejecución del formulario que divide y une archivos



2.4. Comprimir y Descomprimir un Archivo

En ocasiones necesitamos comprimir y descomprimir archivos, .NET Framework tiene el espacio de nombres System.IO.Compression en el cual se encuentran 2 clases que permiten implementar esta funcionalidad: DeflateStream y GZipStream.

La clase DeflateStream implementa el algoritmo de compresión y descompresión de archivos sin pérdidas que es una combinación de 2 algoritmos: LZ77 Y Huffman, mientras que la clase GZipStream implementa el algoritmo gzip que realiza una comprobación para detectar daños en los datos. Este último algoritmo es mas popular que el primero siendo usado por algunos utilitarios de compresión como WinZip y WinRAR.

La programación o el uso de ambas clases es similar en .NET, para comprimir un archivo debemos leer el archivo en una secuencia de bytes (Buffer) y luego escribir la secuencia comprimida usando el método Write de la clase DeflateStream o GZipStream en un flujo de salida o archivo destino.

Para descomprimir debemos ir descomprimiendo por bloques y grabando cada bloque descomprimido en un archivo de salida hasta que no se logre leer nada en el archivo de origen o comprimido.

A continuación presentamos un ejemplo que permite comprimir y descomprimir un archivo usando la clase DeflateStream.

Demo 08

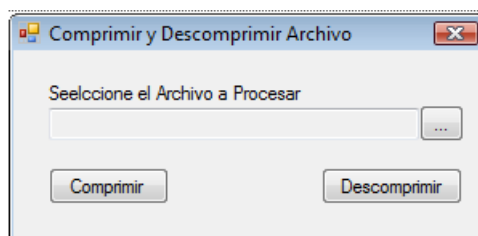
- Crear una aplicación Windows Forms en Visual Basic llamada: Demo08.
- Cambiar de nombre al formulario de form1.vb a frmCompresion.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmCompresion
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	300,150

	StartPosition Text	CenterScreen Comprimir y Descomprimir Archivo
Label1	Name AutoSize Location Text	lblArchivo True 19, 20 Seleccione el Archivo a Procesar
TextBox1	Name Location Size	txtArchivo 22,36 231,20
Button1	Name Cursor Location Size Text	btnAbrir Hand 253,34 23,23 ...
Button2	Name Cursor Location Size Text	btnComprimir Hand 22,73 75,23 Comprimir
Button3	Name Cursor Location Size Text	btnDescomprimir Hand 192,73 88,23 Descomprimir

- El diseño del formulario debe quedar similar al gráfico 2.17:

Gráfico 2.17: Diseño del formulario que comprime un archivo



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'File, FileStream
Imports System.IO.Compression 'DeflateStream
```

Public Class frmCompresion

```
Private Sub MostrarDialogoAbrir(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAbrir.Click
```

```
    Dim ofd As New OpenFileDialog  
    ofd.Title = "Selecciona el archivo a procesar"  
    ofd.Filter = "Todos los archivos|*.*"  
    If ofd.ShowDialog = DialogResult.OK Then  
        txtArchivo.Text = ofd.FileName
```

```
    End If  
End Sub
```

```
Private Sub ComprimirArchivo(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnComprimir.Click  
    Dim Buffer() As Byte = File.ReadAllBytes(txtArchivo.Text)  
    Dim Archivo As String = String.Format("{0}.cmp", txtArchivo.Text)  
    Using fsDestino As New FileStream(Archivo, FileMode.Create, _  
FileAccess.Write, FileShare.Write)
```

```
        Using ds As New DeflateStream(fsDestino, _  
CompressionMode.Compress)  
            ds.Write(Buffer, 0, Buffer.Length)
```

```
        End Using  
    End Using  
    MessageBox.Show("Archivo fue comprimido")  
End Sub
```

```
Private Sub DescomprimirArchivo(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnDescomprimir.Click  
    Dim Buffer(100) As Byte  
    Dim Archivo As String = txtArchivo.Text.Remove(txtArchivo.Text.Length  
- Path.GetExtension(txtArchivo.Text).Length)  
    Using fsOrigen As New FileStream(txtArchivo.Text, FileMode.Open, _  
FileAccess.Read, FileShare.Read)
```

```
        Using fsDestino As New FileStream(Archivo, FileMode.Create, _  
FileAccess.Write, FileShare.Write)  
            Using ds As New DeflateStream(fsOrigen, _  
CompressionMode.Decompress)
```

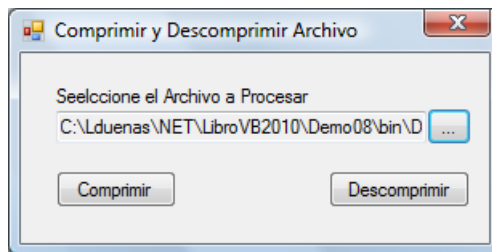
```
                Dim BytesDescomprimidos As Integer  
                Do While True  
                    BytesDescomprimidos = ds.Read(Buffer, 0, Buffer.Length)
```



```
If BytesDescomprimidos > 0 Then
    fsDestino.Write(Buffer, 0, BytesDescomprimidos)
Else
    Exit Do
End If
Loop
End Using
End Using
End Using
MessageBox.Show("Archivo fue descomprimido")
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.18: Ejecución del formulario que comprime archivo



3. Programación Orientada a Objetos (POO)

En este tema veremos cómo programar usando objetos en Visual Basic .NET, primero revisaremos los conceptos básicos de la Programación Orientada a Objetos (POO) y el Modelo de Desarrollo Distribuido (MDD), luego veremos la diferencia de cómo es una aplicación tradicional sin POO ni MDD y luego implementaremos POO y MDD mediante la creación de Librerías de Clases especializadas en servicios para usarlas en una aplicación.

3.1 Introducción a la POO y MDD

Según el MSDN: Un objeto es una combinación de código y datos que puede tratarse como una unidad. Un objeto puede ser una porción de una aplicación, como un control o un formulario. Una aplicación entera también puede ser un objeto.

Cuando se crea una aplicación en Visual Basic, se trabaja constantemente con objetos. Se pueden usar los objetos proporcionados por Visual Basic, como controles, formularios y objetos de acceso a datos. También se pueden usar los objetos de otras aplicaciones en la aplicación de Visual Basic. Incluso pueden crearse objetos propios y agregarles propiedades y métodos adicionales. Los objetos actúan como bloques de creación prefabricados para programas: permiten escribir una porción de código y utilizarla una y otra vez. Para más información ver referencia 14 al final del libro.

Capas o Servicio Lógicos

En toda aplicación siempre existen servicios lógicos los cuales podemos clasificarlos en:

1. Servicios de Usuario

1.1. Interface de Usuario (IU): CUI (Interface de Usuario Carácter) y GUI (Interface de Usuario Grafica)

- Aplicación Windows
- Aplicación Web
- Librería de Controles Windows
- Librería de Controles Web
- Librería de Formularios Windows

1.2. Procesamiento de la Interface de Usuario

- Librería de Código de Usuario General
- Librería de Código de Windows
- Librería de Código de Web

2. Servicios del Negocio o Empresariales

2.1. Entidades del Negocio

- Librería de Entidades del Negocio

2.2. Reglas del Negocio

- Librería de Reglas del Negocio

3. Servicios de Datos

3.1. Acceso a Datos

- Librería Acceso a Datos General:

Ej: ADO NET, SQLHelper, OracleHelper, EntLib, MyBatis, NHibernate

- Librería de Acceso a Datos Ap:

Ej: Base de Datos del Negocio

3.2. Agentes de Servicios

- Librería de Agentes de Servicio

Ej: Otras Bases de Datos, ActiveDirectory, Mails, Web Services, etc.

Tradicionalmente todos estos servicios se han escrito en la misma aplicación, pero si separamos el código en capas se le conoce como **Modelo de Desarrollo Distribuido (MDD)** y es muy útil para reusar código y crear aplicaciones empresariales escalables y mantenibles.

A continuación vamos a crear una Aplicación Windows que calcule el área de un triángulo en función a sus lados usando el Modelo de Programación Tradicional y luego en los siguientes temas veremos cómo distribuirla en capas.

Demo 09

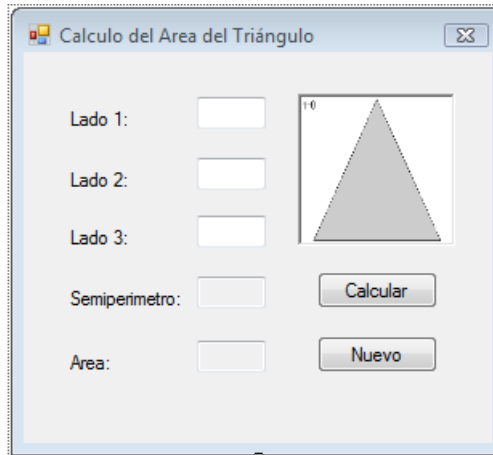
- Crear una aplicación Windows Forms en Visual Basic llamada: Demo09.
- Cambiar de nombre al formulario de form1.vb a frmTriangulo.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmTriangulo
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	300,280
	StartPosition	CenterScreen
	Text	Calculo del Area del Triángulo
Label1	Name	lblLado1
	AutoSize	True
	Location	26, 35
	Text	Lado 1:
TextBox1	Name	txtLado1
	Location	108,28
	Size	43,20
Label2	Name	lblLado2
	AutoSize	True
	Location	26, 73
	Text	Lado 2:
TextBox2	Name	txtLado2
	Location	108,66
	Size	43,20

Label3	Name AutoSize Location Text	lblLado3 True 26, 109 Lado 3:
TextBox3	Name Location Size	txtLado3 108,102 43,20
Label4	Name AutoSize Location Text	lblSemiperimetro True 26, 147 Semiperimetro:
TextBox4	Name Location ReadOnly Size	txtSemiperimetro 108,140 True 43,20
Label5	Name AutoSize Location Text	lblArea True 26, 187 Area:
TextBox5	Name Location ReadOnly Size	txtArea 108,180 True 43,20
PictureBox1	Name BorderStyle Image Location Size SizeMode	picTriangulo Fixed3D Triangulo.gif 171,26 98,95 StretchImage
Button1	Name Cursor Location Size Text	btnCalcular Hand 183,137 75,23 Calcular
Button2	Name Cursor Location Size Text	btnNuevo Hand 183,177 75,23 Nuevo

- El diseño del formulario debe quedar similar al gráfico 2.19:

Gráfico 2.19: Diseño del formulario que calcula el área del triángulo



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'FileStream, StreamWriter
```

```
Public Class frmTriangulo
```

```
Private Sub CalculaArea(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCalcular.Click
    If txtLado1.Text <> "" AndAlso txtLado2.Text <> "" AndAlso _
        txtLado3.Text <> "" Then
        Dim Lado1 As Integer = Integer.Parse(txtLado1.Text)
        Dim Lado2 As Integer = Integer.Parse(txtLado2.Text)
        Dim Lado3 As Integer = Integer.Parse(txtLado3.Text)
        If Lado1 + Lado2 >= Lado3 AndAlso Lado2 + Lado3 >= Lado1 _
            AndAlso Lado1 + Lado3 >= Lado2 Then
            Dim SP As Decimal = (Lado1 + Lado2 + Lado3) / 2
            Dim Area As Decimal = Math.Sqrt(SP * (SP - Lado1) * _
                (SP - Lado2) * (SP - Lado3))
            txtSemiperimetro.Text = SP.ToString
            txtArea.Text = Area.ToString
            Using fs As New FileStream("Triangulo.txt", FileMode.Append, _
                FileAccess.Write, FileShare.Write)
                Using sw As New StreamWriter(fs)
                    sw.WriteLine("{0},{1},{2},{3},{4}", Lado1, Lado2, Lado3, _
                        SP, Area)
                End Using
            End Using
        End If
    End If
End Sub
```

```
End Using
End Using
Else
    MessageBox.Show("Los Lados No forman un Triangulo")
End If
Else
    MessageBox.Show("Falta ingresar un Lado")
End If
End Sub
```

```
Private Sub LimpiarDatos(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnNuevo.Click
    txtLado1.Clear()
    txtLado2.Clear()
    txtLado3.Clear()
    txtSemiperimetro.Clear()
    txtArea.Clear()
    txtLado1.Focus()
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.20: Ejecución del formulario que calcula el área del triángulo

The screenshot shows a Windows application window titled "Calculo del Area del Triángulo". Inside the window, there are five input fields arranged vertically on the left: "Lado 1:" with value 3, "Lado 2:" with value 4, "Lado 3:" with value 5, "Semiperimetro:" with value 6, and "Area:" with value 6. To the right of these fields is a diagram of a triangle with a dashed line representing its height. At the bottom right of the form, there are two buttons: "Calcular" and "Nuevo".

3.2. Creando Bibliotecas de Clases

A continuación vamos a crear varias bibliotecas de clases para distribuir el código en capas o niveles lógicos, de la siguiente forma: el Demo 10 tendrá la Librería de Código de Usuario para Windows, el Demo 11 la Librería de Entidad del Negocio, el Demo12 la Librería de Acceso a Datos y el Demo13 la Librería de Reglas del Negocio.

Creando la Librería de Código de Usuario

Demo 10

- Del menú "File", seleccionar "New Project" y luego "Class Library".
- Escribir como nombre físico al proyecto: Demo10.
- Cambiar de nombre a la clase de Class1.vb a ucFormulario.vb.
- Del menú "Project" seleccionar "Add References" y luego "System.Windows.Forms".
- Escribir el siguiente código:

```
Imports System.Windows.Forms 'Control, MessageBox
```

```
Namespace LibUserCodeWinForms
```

```
Public Class ucFormulario
```

```
Public Shared Sub LimpiarControles(ByVal contenedor As Object)
```

```
Dim X As Control
```

```
For Each X In contenedor.Controls
```

```
    If TypeOf X Is TextBox Then CType(X, TextBox).Clear()
```

```
    If TypeOf X Is CheckBox Then _
```

```
        CType(X, CheckBox).Checked = False
```

```
    If TypeOf X Is RadioButton Then _
```

```
        CType(X, RadioButton).Checked = False
```

```
    If TypeOf X Is ComboBox Then _
```

```
        CType(X, ComboBox).SelectedIndex = 0
```

```
    If TypeOf X Is ListBox Then CType(X, ListBox).SelectedIndex = 0
```

```
    If TypeOf X Is DateTimePicker Then _
```



```
        CType(X, DateTimePicker).Value = Now
    Next
End Sub

Public Shared Sub MostrarMensaje(ByVal Mensaje As String, _
    ByVal txt As TextBox)
    Mensaje = String.Format("Ingresa {0}", Mensaje)
    MessageBox.Show(Mensaje, "Aviso", MessageBoxButtons.OK, _
        MessageBoxIcon.Exclamation)
    txt.Focus()
End Sub
End Class

End Namespace
```

- Del menú "Project" seleccionar "Demo10 Properties", luego borrar Demo10 del "Root namespace".

Nota: Por defecto el espacio de nombres (Namespace) de todo ensamblado (Assembly) es el nombre del proyecto, por ejemplo Demo10, pero si por código se crea un Namespace entonces este se agrega al inicial, es por eso que eliminamos el inicial para definir uno.

- Del menú "Build" seleccionar "Build Demo10".

Nota: Una librería o biblioteca de clases No se puede ejecutar con F5, lo que debemos hacer es compilarla (se creará un archivo con extensión dll).

Creando la Librería de Entidades del Negocio

Demo 11

- Del menú "File", seleccionar "New Project" y luego "Class Library".
- Escribir como nombre físico al proyecto: Demo11.
- Cambiar de nombre a la clase de Class1.vb a beTriangulo.vb.
- Escribir el siguiente código:

```
Namespace LibBusinessEntity
```

```
Public Class beTriangulo
    Private _Lado1 As Integer
    Private _Lado2 As Integer
    Private _Lado3 As Integer
    Private _SemiPerimetro As Decimal
    Private _Area As Decimal
```

```
Public Property Lado1() As Integer
    Get
        Return (_Lado1)
    End Get
    Set(ByVal value As Integer)
        _Lado1 = value
    End Set
End Property
```

```
Public Property Lado2() As Integer
    Get
        Return (_Lado2)
    End Get
    Set(ByVal value As Integer)
        _Lado2 = value
    End Set
End Property
```

```
Public Property Lado3() As Integer
    Get
```

```
        Return (_Lado3)
    End Get
    Set(ByVal value As Integer)
        _Lado3 = value
    End Set
End Property

Public Property SemiPerimetro() As Decimal
    Get
        Return (_SemiPerimetro)
    End Get
    Set(ByVal value As Decimal)
        _SemiPerimetro = value
    End Set
End Property

Public Property Area() As Decimal
    Get
        Return (_Area)
    End Get
    Set(ByVal value As Decimal)
        _Area = value
    End Set
End Property
End Class

End Namespace
```

- Del menú "Project" seleccionar "Demo11 Properties", luego borrar Demo11 del "Root namespace".
- Del menú "Build" seleccionar "Build Demo11" y crear la dll.

Creando la Librería de Acceso a Datos

Demo 12

- Del menú "File", seleccionar "New Project" y luego "Class Library".
- Escribir como nombre físico al proyecto: Demo12.
- Hacer una referencia a la librería de entidad de negocio creada en el Demo 11: menú "Project", "Add References", ficha "Browse" y seleccionar el archivo Demo11.dll.

Nota: El archivo dll se encuentra en la carpeta del proyecto dentro de la carpeta Release que se encuentra dentro de la carpeta Bin.

- Cambiar de nombre a la clase de Class1.vb a daTriangulo.vb.
- Escribir el siguiente código:

```
Imports LibBusinessEntity 'beTriangulo
Imports System.IO 'FileStream, StreamWriter

Namespace LibDataAccess

    Public Class daTriangulo
        Public Sub GrabarArchivo(ByVal obeTriangulo As beTriangulo, _
            ByVal Archivo As String)
            Using fs As New FileStream(Archivo, FileMode.Append, _
                FileAccess.Write, FileShare.Write)
                Using sw As New StreamWriter(fs)
                    sw.WriteLine("{0},{1},{2},{3},{4}", obeTriangulo.Lado1, _
                        obeTriangulo.Lado2, obeTriangulo.Lado3, _
                        obeTriangulo.SemiPerimetro, obeTriangulo.Area)
                End Using
            End Using
        End Sub
    End Class

End Namespace
```

- Del menú "Project" seleccionar "Demo12 Properties", luego borrar Demo12 del "Root namespace".
- Del menú "Build" seleccionar "Build Demo12" y crear la dll.

Creando la Librería de Reglas del Negocio

Demo 13

- Del menú "File", seleccionar "New Project" y luego "Class Library".
- Escribir como nombre físico al proyecto: Demo13.
- Hacer una referencia a la librería de entidad de negocio creada en el Demo 11: menú "Project", "Add References", ficha "Browse" y seleccionar el archivo Demo11.dll.
- También hacer referencia a la librería de acceso a datos creada en el Demo 12: menú "Project", "Add References", ficha "Browse" y seleccionar el archivo Demo12.dll.
- Cambiar de nombre a la clase de Class1.vb a brTriangulo.vb.
- Escribir el siguiente código:

```
Imports LibBusinessEntity
Imports LibDataAccess
```

```
Namespace LibBusinessRules
```

```
Public Class brTriangulo
    Public Function EsTriangulo(ByVal obeTriangulo As beTriangulo) _
        As Boolean
        Dim exito As Boolean = obeTriangulo.Lado1 + _
            obeTriangulo.Lado2 >= obeTriangulo.Lado3 _
            AndAlso obeTriangulo.Lado2 + obeTriangulo.Lado3 >=
            obeTriangulo.Lado1 AndAlso obeTriangulo.Lado1 +
            obeTriangulo.Lado3 >= obeTriangulo.Lado2
        Return (exito)
    End Function
End Class
```

```
Public Function CalcularSemiPerimetro(ByVal obeTriangulo As  
beTriangulo) As Decimal  
    Dim SP As Decimal = (obeTriangulo.Lado1 + obeTriangulo.Lado2 + _  
        obeTriangulo.Lado3) / 2  
    Return (SP)  
End Function
```

```
Public Function CalcularArea(ByVal obeTriangulo As beTriangulo) _  
As Decimal  
    Dim Area As Decimal = Math.Sqrt(obeTriangulo.SemiPerimetro * _  
        (obeTriangulo.SemiPerimetro - obeTriangulo.Lado1) * _  
        (obeTriangulo.SemiPerimetro - obeTriangulo.Lado2) * _  
        (obeTriangulo.SemiPerimetro - obeTriangulo.Lado3))  
    Return (Area)  
End Function
```

```
Public Sub GrabarArchivo(ByVal obeTriangulo As beTriangulo, _  
ByVal Archivo As String)  
    Dim odaTriangulo As New daTriangulo  
    odaTriangulo.GrabarArchivo(obeTriangulo, Archivo)  
End Sub  
End Class
```

End Namespace

- Del menú "Project" seleccionar "Demo13 Properties", luego borrar Demo13 del "Root namespace".
- Del menú "Build" seleccionar "Build Demo13" y crear la dll.

3.3. Creando una Aplicación que use las Bibliotecas de Clases

Finalmente, crearemos la aplicación que use las librerías creadas anteriormente, esta tiene que referenciar solo a las librerías de usuarios y las de negocios pero no puede directamente trabajar con la librería de datos, ya que está se encapsulo a través de la librería de reglas de negocio.

Demo 14

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Escribir como nombre físico al proyecto: Demo14.
- Hacer una referencia a la librería de código de usuario para Windows creada en el Demo 10: menú "Project", "Add References", ficha "Browse" y seleccionar el archivo Demo10.dll.
- También hacer referencia a la librería de entidades de negocio creada en el Demo 11: menú "Project", "Add References", ficha "Browse" y seleccionar el archivo Demo11.dll.
- También hacer referencia a la librería de reglas de negocio creada en el Demo 13: menú "Project", "Add References", ficha "Browse" y seleccionar el archivo Demo13.dll.
- Eliminar el formulario Form1 dando clic derecho y seleccionar "Delete".
- Agregar el formulario creado en el Demo09, clic derecho al proyecto, menú "Add", "Existing Item", seleccionar "frmTriangulo.vb"
- Borrar el código existente y escribir lo siguiente:

```
Imports LibBusinessEntity
Imports LibBusinessRules
Imports LibUserCodeWinForms
```

```
Public Class frmTriangulo
```

```
Private Sub CalculaArea(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnCalcular.Click
    If txtLado1.Text <> "" Then
        If txtLado2.Text <> "" Then
            If txtLado3.Text <> "" Then
                Dim obeTriangulo As New beTriangulo
                With obeTriangulo
                    .Lado1 = Integer.Parse(txtLado1.Text)
                    .Lado2 = Integer.Parse(txtLado2.Text)
                    .Lado3 = Integer.Parse(txtLado3.Text)
                End With
                Dim obrTriangulo As New brTriangulo
                With obrTriangulo
                    If .EsTriangulo(obeTriangulo) Then
                        obeTriangulo.SemiPerimetro = _
                            .CalcularSemiPerimetro(obeTriangulo)
                        obeTriangulo.Area = .CalcularArea(obeTriangulo)
                        txtSemiperimetro.Text = _
                            obeTriangulo.SemiPerimetro.ToString
                        txtArea.Text = obeTriangulo.Area.ToString
                        .GrabarArchivo(obeTriangulo, "Triangulo.txt")
                    Else
                        MessageBox.Show("Los Lados No forman un Triangulo")
                    End If
                End With
            Else
                ucFormulario.MostrarMensaje("Lado 3", txtLado1)
            End If
        Else
            ucFormulario.MostrarMensaje("Lado 2", txtLado1)
        End If
    Else
        ucFormulario.MostrarMensaje("Lado 1", txtLado1)
    End If
End Sub

Private Sub LimpiarDatos(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnNuevo.Click
    ucFormulario.LimpiarControles(Me)
    txtLado1.Focus()
```


End Sub
End Class

- Configurar el formulario de inicio de la aplicación, clic derecho al proyecto, seleccionar "Properties", luego ficha "Application", en la sección "Startup Form" cambiar Form1 por frmTriangulo y cerrar la ventana.
- Grabar y ejecutar la aplicación.

Nota: La ventana que se mostrará será la misma de la figura 2.20.

4. Programación Asíncrona y Paralelismo

Cuando ejecutamos varias tareas en una aplicación estas se ejecutan en forma secuencial o síncrona, teniendo la desventaja que si la tarea demora demasiado la aplicación cliente queda colgada hasta que se termine de procesar dicha tarea.

Para resolver este problema existen técnicas de programación asíncrona que nos permiten ejecutar tareas en simultaneo sin que la aplicación se detenga o se pause, por ejemplo al mostrar gran cantidad de líneas de texto de un archivo o al leer y mostrar miles de registros de una base de datos.

En .NET Framework existen varias técnicas para implementar programación asíncrona, tales como:

- Métodos asíncronos mediante IAsyncResult
- Programación asíncrona mediante Delegados
- Programación asíncrona mediante el Modelo Basado en Eventos
- Programación paralela

Para ver más información sobre este tema consultar la referencia 15 al final del libro.

.NET tiene muchas clases que permiten implementar las diferentes técnicas de programación asíncrona mencionadas, entre las cuales tenemos: Thread, ThreadPool, Delegate, Tasks, etc. La mayoría se encuentran en el espacio de nombres System.Threading.

4.1 Threads

La clase Thread se usa para manejar subtareas y puede ser útil para iniciar una subtarea en segundo plano e ir mostrando progresivamente los resultados, por ejemplo leer un archivo de texto e ir mostrando sus líneas en una Lista.

Esta clase tiene métodos como Start que inician la ejecución de un subproceso, luego podemos pausar usando el método Suspend y reanudar este usando el método Resume, si deseamos cancelar el subproceso iniciado usaremos el método Abort.

A continuación veremos diferentes ejemplos del uso de la clase Thread, el primero de ellos el Demo 15 usa un solo Thread, el Demo 16 usa 2 Threads dependientes y el Demo17 usa un grupo de Thread o ThreadPool.

Thread Simple

Demo 15

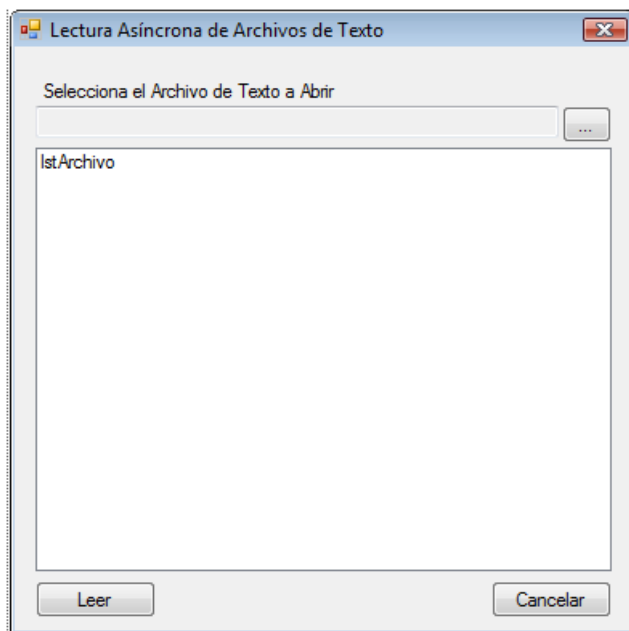
- Crear una aplicación Windows Forms en Visual Basic llamada: Demo15.
- Cambiar de nombre al formulario de form1.vb a frmLectorArchivo.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmLectorArchivo FixedSingle False False 400,400 CenterScreen Lectura Asíncrona de Archivos de Texto
Label1	Name AutoSize Location Text	lblNombre True 15, 21 Selecciona el Archivo de Texto a Abrir
TextBox1	Name Location ReadOnly Size	txtNombre 13,37 True 326,20
Button1	Name Cursor Location	btnAbrir Hand 341,37

	Size Text	31,23 ...
ListBox1	Name Location Size	IstArchivo 13,63 359,264
Button2	Name Cursor Location Size Text	btnLeer Hand 13,333 75,23 Leer
Button3	Name Cursor Location Size Text	btnCancelar Hand 297,333 75,23 Cancelar

- El diseño del formulario debe quedar similar al gráfico 2.21:

Gráfico 2.21: Diseño del formulario que lee archivos asíncronamente



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'StreamReader
Imports System.Text 'StringBuilder, Encoding
Imports System.Threading 'Thread

Public Class frmLectorArchivo
    Private objHilo As Thread

    Private Sub LeerArchivo()
        btnLeer.Text = "Detener"
        btnCancelar.Enabled = True
        Using sr As New StreamReader(txtNombre.Text, Encoding.Default)
            Do While Not sr.EndOfStream
                lstArchivo.Items.Add(sr.ReadLine)
            Loop
        End Using 'Close y Dispose
        btnLeer.Text = "Leer"
        btnCancelar.Enabled = False
    End Sub

    Private Sub MostrarDialogoAbrir(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnAbrir.Click
        Dim ofd As New OpenFileDialog 'Dialogo de Abrir de Windows
        ofd.Title = "Selecciona el archivo a abrir"
        ofd.Filter = "Archivos de texto|*.txt"
        If ofd.ShowDialog() = Windows.Forms.DialogResult.OK Then
            txtNombre.Text = ofd.FileName
            lstArchivo.Items.Clear()
        End If
    End Sub

    Private Sub LeerArchivoPorLinea(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnLeer.Click
        If btnLeer.Text = "Leer" Then
            objHilo = New Thread(AddressOf LeerArchivo)
            objHilo.Start()
        Else
            If btnLeer.Text = "Detener" Then
                objHilo.Suspend()
                btnLeer.Text = "Continuar"
                btnCancelar.Enabled = False
            End If
        End If
    End Sub
```

```
Else
    objHilo.Resume()
    btnLeer.Text = "Detener"
    btnCancelar.Enabled = True
End If
End If
End Sub
```

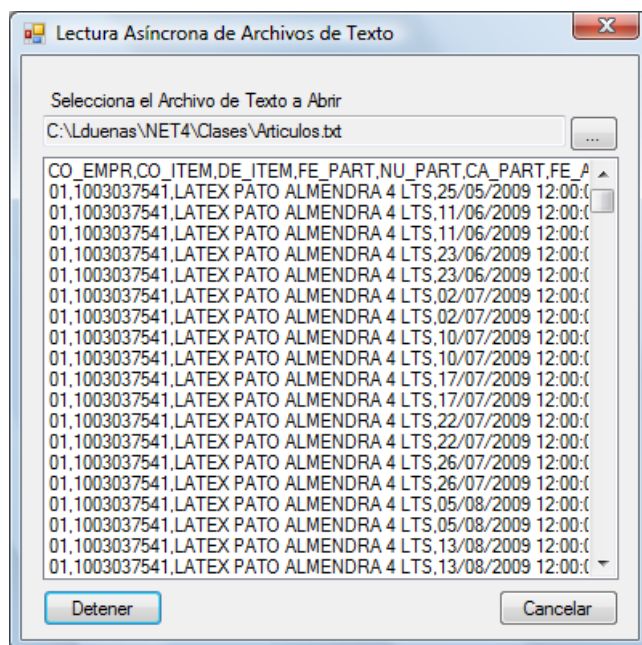
```
Private Sub ConfigurarSubproceso(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.Load
    CheckForIllegalCrossThreadCalls = False
End Sub
```

```
Private Sub CancelarSubproceso(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnCancelar.Click
    objHilo.Abort()
    btnCancelar.Enabled = False
    txtNombre.Clear()
    lstArchivo.Items.Clear()
    btnLeer.Text = "Leer"
End Sub
```

```
Private Sub HabilitarBoton(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles txtNombre.TextChanged
    btnLeer.Enabled = (txtNombre.Text <> "")
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.22: Ejecución del formulario que lee archivos asíncronamente



Threads Dependientes

Demo 16

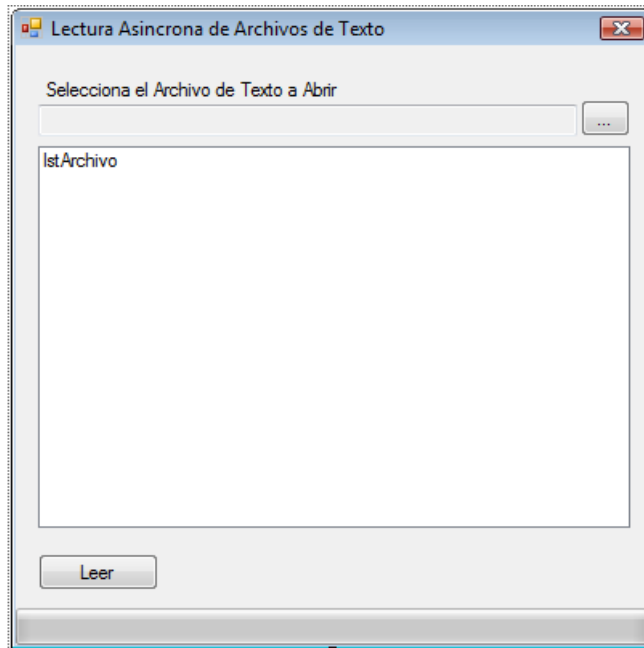
- Crear una aplicación Windows Forms en Visual Basic llamada: Demo16.
- Cambiar de nombre al formulario de form1.vb a frmLectorArchivo.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmLectorArchivo FixedSingle False False 400,400 CenterScreen Lectura Asíncrona de Archivos de Texto
Label1	Name AutoSize Location Text	lblNombre True 15, 21 Selecciona el Archivo de Texto a Abrir
TextBox1	Name Location ReadOnly Size	txtNombre 13,37 True 326,20
Button1	Name Cursor Location Size Text	btnAbrir Hand 341,37 31,23 ...
ListBox1	Name Location Size	lstArchivo 13,63 359,238
Button2	Name Cursor Location Size Text	btnLeer Hand 13,333 75,23 Leer
ProgressBar1	Name	pbrArchivo

	Dock	Bottom

- El diseño del formulario debe quedar similar al gráfico 2.23:

Gráfico 2.23: Diseño del formulario que lee archivos asíncronamente con barra de progreso



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'StreamReader
Imports System.Text 'StringBuilder, Encoding
Imports System.Threading 'Thread
```

```
Public Class frmLectorArchivo
    Private objHiloContar As Thread
    Private objHiloLeer As Thread
    Private CL As Long
```

```
Private Sub ContarLineasArchivo()
    CL = 0
    Using sr As New StreamReader(txtNombre.Text, _
        Encoding.Default)
```

```
Do While Not sr.EndOfStream
    sr.ReadLine()
    CL = CL + 1
Loop
End Using
End Sub

Private Sub LeerArchivo()
    Dim B As Long = CL \ pbrArchivo.Maximum
    Dim C As Long 'Contador
    btnLeerPorLinea.Enabled = False
    Using sr As New StreamReader(txtNombre.Text, Encoding.Default)
        Do While Not sr.EndOfStream
            lstArchivo.Items.Add(sr.ReadLine)
            If C Mod B = 0 Then
                If pbrArchivo.Value < pbrArchivo.Maximum Then
                    pbrArchivo.Value += 1
                End If
            End If
            C = C + 1
        Loop
    End Using 'Close y Dispose
    btnLeerPorLinea.Enabled = True
End Sub

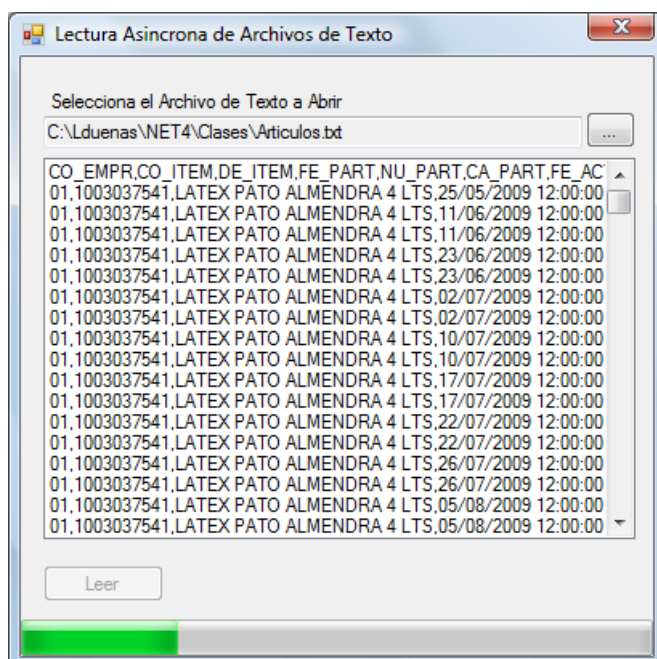
Private Sub MostrarDialogoAbrir(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAbrir.Click
    Dim ofd As New OpenFileDialog 'Dialogo de Abrir de Windows
    ofd.Title = "Selecciona el archivo a abrir"
    ofd.Filter = "Archivos de texto|*.txt"
    If ofd.ShowDialog() = Windows.Forms.DialogResult.OK Then
        txtNombre.Text = ofd.FileName
        lstArchivo.Items.Clear()
    End If
End Sub

Private Sub LeerArchivoPorLinea(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnLeerPorLinea.Click
    objHiloContar = New Thread(AddressOf ContarLineasArchivo)
    objHiloContar.Start()
```

```
objHiloContar.Join()  
objHiloLeer = New Thread(AddressOf LeerArchivo)  
objHiloLeer.Start()  
End Sub  
  
Private Sub ConfigurarSubproceso(ByVal sender As Object, _  
ByVal e As System.EventArgs) Handles Me.Load  
    CheckForIllegalCrossThreadCalls = False  
End Sub  
  
Private Sub HabilitarBoton(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles txtNombre.TextChanged  
    btnLeerPorLinea.Enabled = (txtNombre.Text <> "")  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

**Gráfico 2.24: Ejecución del formulario que lee archivos
asíncronamente con barra de progreso**



ThreadPool

Demo 17

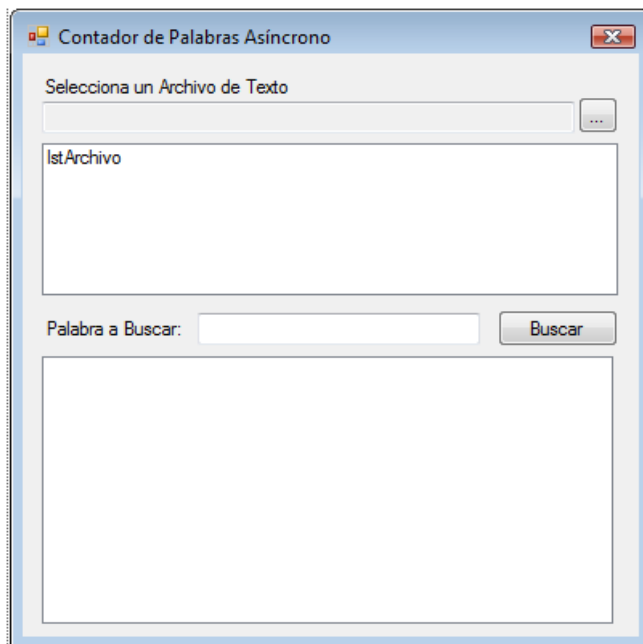
- Crear una aplicación Windows Forms en Visual Basic llamada: Demo17.
- Cambiar de nombre al formulario de form1.vb a frmContadorPalabras.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmContadorPalabras FixedSingle False False 400,400 CenterScreen Contador de Palabras Asíncrono
Label1	Name AutoSize Location Text	lblArchivo True 11, 14 Selecciona un Archivo de Texto
TextBox1	Name Location ReadOnly Size	txtArchivo 12,30 True 332,20
Button1	Name Cursor Location Size Text	btnAbrir Hand 346,27 25,23 ...
ListBox1	Name Location Size	lstArchivo 12,56 359,95
Label2	Name AutoSize Location Text	lblPalabra True 12, 165 Palabra a Buscar:
TextBox2	Name Location Size	txtPalabra 109,162 176,20

Button2	Name Cursor Location Size Text	btnBuscar Hand 296,160 75,23 Buscar
ListView1	Name FullRowSelect GridLines HotTracking Location Size	lvwResultado True True True 12,189 356,167

- El diseño del formulario debe quedar similar al gráfico 2.25:

Gráfico 2.25: Diseño del formulario contador de palabras asíncrono



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'File, FileStream, StreamReader
Imports System.Threading 'ThreadPool
```

```
Public Class frmContadorPalabras
```

```
Private Sub ContarPalabras(ByVal Archivo As String)
    Dim CLP As Integer = 0
    If File.Exists(Archivo) Then
        Using fs As New FileStream(Archivo, FileMode.Open, _
            FileAccess.Read, FileShare.Read)
            Using sr As New StreamReader(fs)
                Do While Not sr.EndOfStream
                    If sr.ReadLine.ToUpper.Contains _
                        (txtPalabra.Text.ToUpper) Then
                        CLP += 1
                    End If
                Loop
            End Using
        End Using
        Dim Fila As ListViewItem = lvwResultado.Items.Add(Archivo)
        Fila.SubItems.Add(CLP)
        Fila.SubItems.Add(Thread.CurrentThread.ManagedThreadId)
    End If
End Sub

Private Sub CrearColumnasListView(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    With lvwResultado.Columns
        .Add("Archivo", 200, HorizontalAlignment.Left)
        .Add("Total", 60, HorizontalAlignment.Left)
        .Add("IdThread", 80, HorizontalAlignment.Left)
    End With
    lvwResultado.View = View.Details
    CheckForIllegalCrossThreadCalls = False
End Sub

Private Sub btnAbrir_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnAbrir.Click
    Dim ofd As New OpenFileDialog "Dialogo de Abrir
    ofd.Title = "Seleccione archivo de texto a añadir"
    ofd.Filter = "Archivos de texto|*.txt"
    If ofd.ShowDialog = Windows.Forms.DialogResult.OK Then
        txtArchivo.Text = ofd.FileName
        lstArchivo.Items.Add(ofd.FileName)
        lstArchivo.SelectedIndex = 0
    End If
End Sub
```

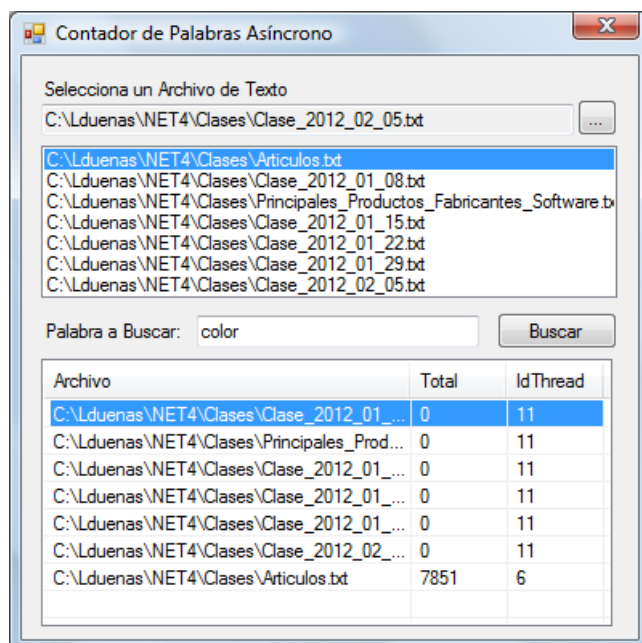
```
End If  
End Sub
```

```
Private Sub HabilitarBotonBuscar(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles _  
txtPalabra.TextChanged, lstArchivo.SelectedIndexChanged  
    btnBuscar.Enabled = (lstArchivo.Items.Count > 0 AndAlso _  
    txtPalabra.Text.Trim <> "")  
End Sub
```

```
Private Sub Buscar(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnBuscar.Click  
    Dim I As Integer  
    For I = 0 To lstArchivo.Items.Count - 1  
        ThreadPool.QueueUserWorkItem(AddressOf _  
            ContarPalabras, lstArchivo.Items(I))  
    Next  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.26: Ejecución del formulario contador de palabras asíncrono



4.2. Eventos Asíncronos

El modelo de programación asíncrono basado en eventos permite implementar de manera simple el subprocesamiento y algunas clases de .NET Framework tienen dicho soporte como los Servicios Web y los Servicios WCF.

A continuación veremos 2 ejemplos: uno que es un Servicio WCF que ejecuta una consulta de datos la cual tiene una pausa de 10 segundos para demorar la consulta y el otro es una aplicación Windows que consume dicho servicio pero en forma asíncrona usando el modelo de eventos.

Demo 18

- Del menú "File", seleccionar "New Web Site" y luego "WCF Service".
- Crear un Servicio WCF en Visual Basic llamado: Demo18.
- Cambiar de nombre a la interface de IService1.vb a IEmpleado.vb
- Borrar el código mostrado y escribir el siguiente código:

```
Namespace ServicioWCF
    <ServiceContract()> _
    Public Interface IEmpleado
        <OperationContract()> _
        Function ListarEmpleados() As DataSet
    End Interface
End Namespace
```

- Cambiar de nombre a la clase de Service1.svc a Empleado.svc
- Borrar el código mostrado y escribir el siguiente código:

```
Imports System.Data 'DataSet
Imports System.Data.SqlClient 'SqlConnection, SqlDataAdapter

Namespace ServicioWCF
    Public Class Empleado
        Implements ServicioWCF.IEmpleado
```

```
Public Function ListarEmpleados() As DataSet _
Implements ServicioWCF.IEmpleado.ListarEmpleados
    Dim dst As New DataSet
    Using con As New SqlConnection("uid=sa;pwd=123456; _
data source=Lduenas\MCTS;initial catalog=Northwind")
        con.Open()
        Dim dap As New SqlDataAdapter("WaitFor Delay '00:00:10'; _
Select EmployeeID,LastName,FirstName From Employees", con)
        dap.Fill(dst, "Empleados")
    End Using 'con.Close
    Return (dst)
End Function
End Class
End Namespace
```

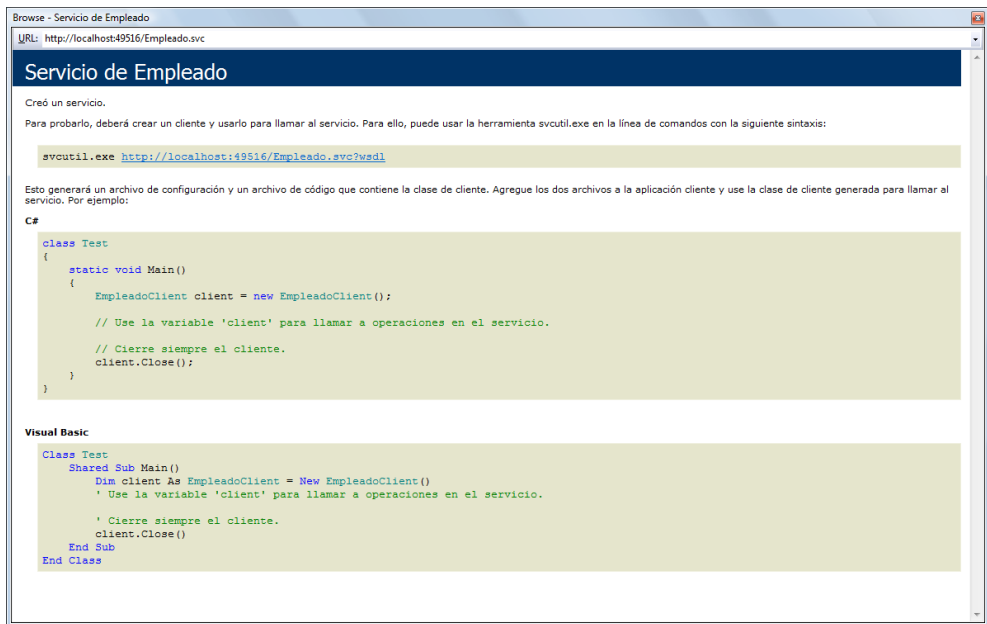
Nota: La cadena de conexión debe ser modificada de acuerdo al usuario (uid) y clave (pwd) de SQL que se hubiera creado, además cambiar el nombre del servidor (data source).

- Del menú "Project" seleccionar "Demo18 Properties", luego borrar Demo18 del "Root namespace".
- Clic derecho sobre el archivo Empleado.svc y seleccionar "View Markup", cambiar el código a lo siguiente

```
<%@ ServiceHost Language="VB" Debug="true"
Service="ServicioWCF.Empleado" CodeBehind="Empleado.svc.vb" %>
```

- Clic derecho sobre el archivo Empleado.svc y seleccionar "View in Browser".

Gráfico 2.27: Ejecución del Servicio WCF de Empleado



- Clic al enlace que contiene el archivo wsdl y copiar la dirección (URL).

Demo 19

- Abrir otro Visual Studio para crear la aplicación de prueba.

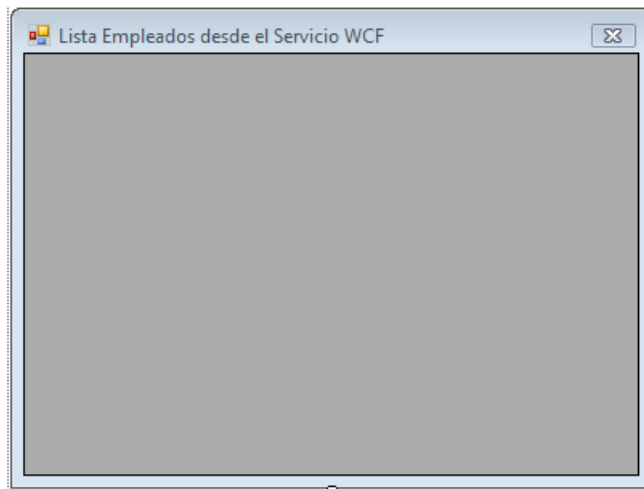
Nota: No puede cerrar el otro proyecto del Servicio WCF ya que debe estar ejecutándose para poder consumirlo desde la aplicación.

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo19.
- Cambiar de nombre al formulario de form1.vb a frmListaEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmListaEmpleado FixedSingle False False 400,300 CenterScreen Lista Empleados desde el Servicio WCF
DataGridView1	Name Dock ReadOnly SelectionMode	dgvEmpleado Fill True FullRowSelect

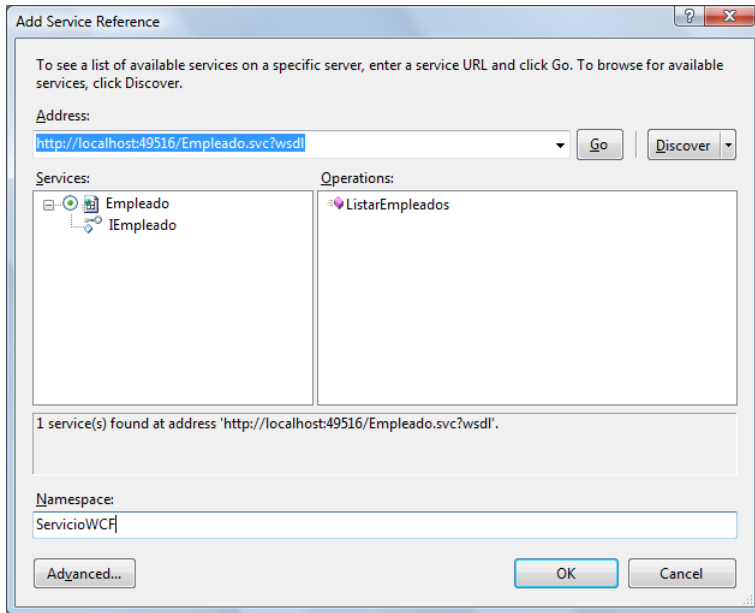
- El diseño del formulario debe quedar similar al gráfico 2.28:

Gráfico 2.28: Diseño del formulario que lista empleados desde un Servicio WCF asíncrono



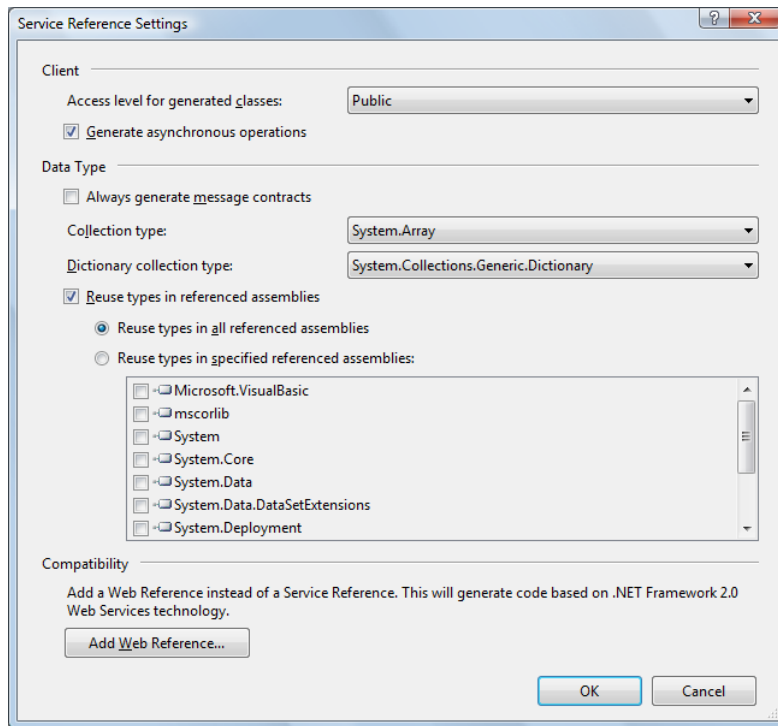
- Hacer una referencia al Servicio WCF, clic derecho al proyecto Demo19 y seleccionando "Add Service Reference".
- En la ventana pegar la dirección del wsdl del Servicio WCF anteriormente creado, similar a la figura mostrada.

Gráfico 2.29: Ventana de Adicionar Referencia a un Servicio



- Escribir un espacio de nombres para el servicio, por ejemplo Servicio WCF y clic en el botón "Advanced", se mostrará una ventana similar al gráfico 2.30.
- En la ventana de configuración del servicio, activar la casilla "Generar operaciones asíncronas".

Gráfico 2.30: Ventana de Configurar la Referencia al Servicio



- Escribir el siguiente código en el formulario:

```
Public Class frmListaEmpleado
    Private dst As New DataSet
    Private WithEvents oEmpleado As New ServicioWCF.EmpleadoClient

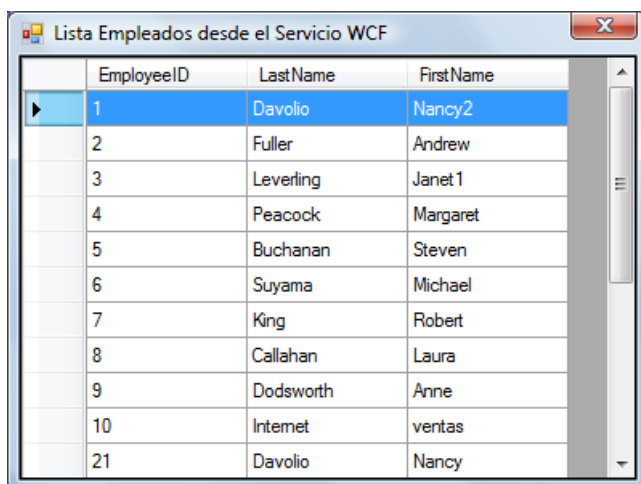
    Private Sub IniciarLlamadaAsincrona(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.Load
        oEmpleado.ListarEmpleadosAsync()
    End Sub

    Private Sub FinalizarLlamadaAsincrona(ByVal sender As Object, _
        ByVal e As ServicioWCF.ListarEmpleadosCompletedEventArgs) Handles _
        oEmpleado.ListarEmpleadosCompleted
        dst = e.Result
        Dim mi As New MethodInvoker(AddressOf MostrarResultados)
        Me.Invoke(mi)
    End Sub
```

```
Private Sub MostrarResultados()  
    dgvEmpleado.DataSource = dst.Tables(0)  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación.

Gráfico 2.31: Ejecución del formulario que lista empleados desde un Servicio WCF asíncrono



EmployeeID	LastName	FirstName
1	Davolio	Nancy2
2	Fuller	Andrew
3	Leverling	Janet 1
4	Peacock	Margaret
5	Buchanan	Steven
6	Suyama	Michael
7	King	Robert
8	Callahan	Laura
9	Dodsworth	Anne
10	Internet	ventas
21	Davolio	Nancy

Nota: Observar como aparece el formulario y se puede continuar trabajando con él mientras se está ejecutando la llamada al Servicio WCF hasta que pasado 10 segundos se lista los datos en la grilla.

4.3. Delegados Callbacks

La forma más sencilla de implementar la programación asíncrona es usar el Modelo de Eventos, pero no todas las clases de .NET Framework soportan este modelo de eventos, para esto la solución es implementar los delegados Callbacks.

La mayoría de desarrolladores profesionales usan el Modelo de Desarrollo Distribuido (MDD) y tienen escrito mucho código en Librerías .NET que son usadas por aplicativos Windows, Web, WPF, etc. Si al llamar a un método de una clase de una librería desde una aplicación, si esta llamada se demora dejaría colgada a la aplicación.

Para resolver este problema debemos implementar los delegados Callbacks que consisten en crear una función delegado con la misma firma que la del método a llamar, luego creamos un objeto de tipo delegado que apunte al método. Usaremos el método BeginInvoke de este objeto para iniciar la llamada asíncrona y el método EndInvoke para finalizar la llamada desde el procedimiento de regreso o Callback.

A continuación crearemos un par de ejemplos para demostrar los delegados Callback, el Demo 20 que será una Librería de Datos con un método que demore 10 segundos y luego liste los empleados desde una base de datos y el Demo21 que es la aplicación que consume la librería en forma asíncrona usando un delegado Callback.

Demo 20

- Del menú “File”, seleccionar “New Project” y luego “Class Library”.
- Escribir como nombre físico al proyecto: Demo20.
- Cambiar de nombre a la clase de Class1.vb a daEmpleado.vb.
- Escribir el siguiente código:

```
Imports System.Data 'DataSet
Imports System.Data.SqlClient 'SqlConnection, SqlDataAdapter

Namespace LibreriaDatos
    Public Class daEmpleado
        Public Function ListarEmpleados() As DataSet
            Dim dst As New DataSet
            Using con As New SqlConnection("uid=sa;pwd=123456; _
            data source=Lduenas\MCTS;initial catalog=Northwind")
                con.Open()
                Dim dap As New SqlDataAdapter("WaitFor Delay '00:00:10'; _
                Select EmployeeID,LastName,FirstName From Employees", con)
                dap.Fill(dst, "Empleados")
            End Using 'con.Close
            Return (dst)
        End Function
    End Class
End Namespace
```

- Del menú “Project” seleccionar “Demo20 Properties”, luego borrar Demo20 del “Root namespace”.
- Del menú “Build” seleccionar “Build Demo20” para crear la dll.

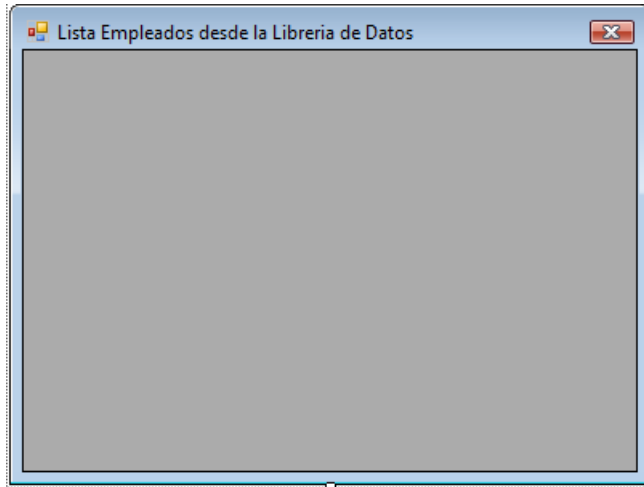
Demo 21

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo21.
- Cambiar de nombre al formulario de form1.vb a frmListaEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaEmpleado
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,300
	StartPosition	CenterScreen
	Text	Lista Empleados desde la Librería de Datos
DataGridView1	Name	dgvEmpleado
	Dock	Fill
	ReadOnly	True
	SelectionMode	FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 2.32:

Gráfico 2.32: Diseño del formulario que lista empleados desde la librería de datos asíncrona



- Hacer una referencia a la librería de clases creada anteriormente, del menú "Project", seleccionar "Add References", ficha "Browse" y luego ubicar el archivo Demo20.dll
- Escribir el siguiente código en el formulario:

```
Public Class frmListaEmpleado
    Private dst As New DataSet
    Private odaEmpleado As New LibreriaDatos.daEmpleado
    Delegate Function DelegadoListarEmpleados() As DataSet
    Private oDelegado As New DelegadoListarEmpleados _
        (AddressOf odaEmpleado.ListarEmpleados)

    Private Sub IniciarLlamadaAsincrona(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.Load
        Dim acb As New AsyncCallback(AddressOf FinalizarLlamadaAsincrona)
        oDelegado.BeginInvoke(acb, Nothing)
    End Sub

    Private Sub FinalizarLlamadaAsincrona(ByVal iar As IAsyncResult)
        dst = oDelegado.EndInvoke(iar)
        Dim mi As New MethodInvoker(AddressOf MostrarResultados)
        Me.Invoke(mi)
    End Sub
```

```
Private Sub MostrarResultados()  
    dgvEmpleado.DataSource = dst.Tables(0)  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación.

Gráfico 2.33: Ejecución del formulario que lista empleados desde la librería de datos asíncrona



	EmployeeID	LastName	FirstName
▶	1	Davolio	Nancy2
	2	Fuller	Andrew
	3	Leverling	Janet 1
	4	Peacock	Margaret
	5	Buchanan	Steven
	6	Suyama	Michael
	7	King	Robert
	8	Callahan	Laura
	9	Dodsworth	Anne
	10	Internet	ventas
	21	Davolio	Nancy

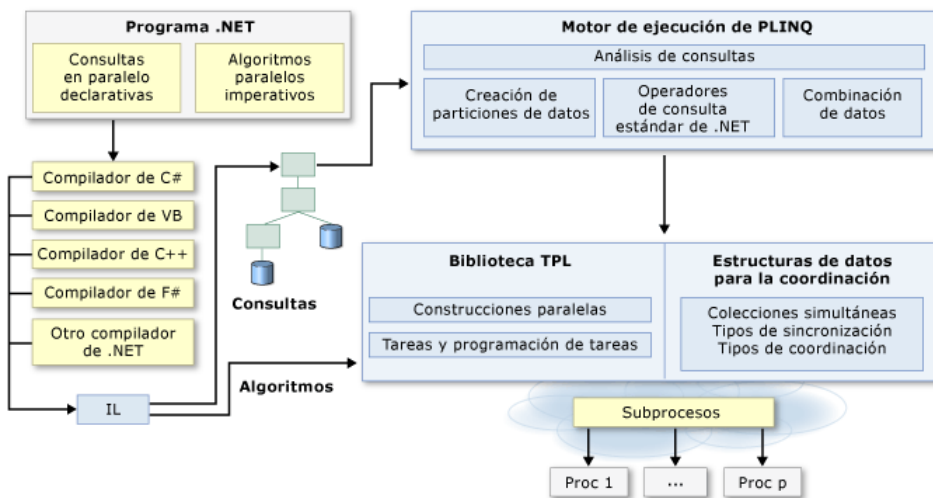
4.4. Programación Paralela

La programación paralela permite aprovechar los computadores con varios procesadores o núcleos de procesamiento, permitiendo crear subtareas en diferentes procesadores en vez de hacerlo en uno solo. Actualmente, la mayoría de PCs disponen de más de un procesador y es importante tomar ventaja de esta característica desde nuestras aplicaciones.

Anteriormente, era difícil programar código para manejar varios procesadores, pero con .NET Framework 4 se ha incorporado la Librería de Procesamiento de Tareas en paralelo más conocida como la TPL.

Existen 2 formas de implementar la programación paralela en .NET: Paralelismo de Datos y Paralelismo de Tareas. También Microsoft ha extendido la programación paralela en LINQ al cual se le conoce como PLINQ. El gráfico mostrado a continuación resume la arquitectura de la programación paralela en .NET Framework 4.

Gráfico 2.34: Arquitectura de programación paralela en .NET



Para conocer más sobre este tema ver la referencia 16 al final del libro. A continuación presentamos un ejemplo sobre Paralelismo de Datos y el otro sobre Paralelismo de Tareas.

Paralelismo de Datos

El paralelismo de datos hace referencia a los escenarios en los que la misma operación se realiza simultáneamente (es decir, en paralelo) en elementos de una colección o matriz de origen. Varias sobrecargas de los métodos `ForEach` y `For` admiten el paralelismo de los datos. En las operaciones paralelas de datos, se crean particiones de la colección de origen para que varios subprocesos puedan funcionar simultáneamente en segmentos diferentes.

TPL admite el paralelismo de datos a través de la clase `System.Threading.Tasks.Parallel`. Esta clase proporciona las implementaciones paralelas basadas en método de los bucles `For` y `For Each` en Visual Basic. Se escribe la lógica del bucle para un bucle `Parallel.For` o `Parallel.ForEach` de forma muy similar a como se escribiría un bucle secuencial. No tiene que crear los subprocesos ni poner en la cola los elementos de trabajo.

En el siguiente ejemplo se muestra un bucle `For` en paralelo que permite crear 1000 gráficos de círculos con fondos degradados generados al azar.

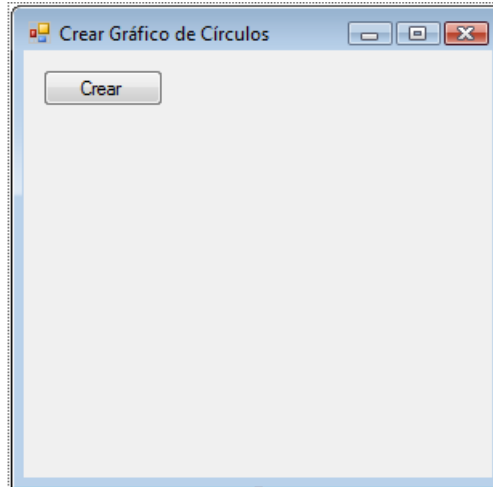
Demo 22

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo22.
- Cambiar de nombre al formulario de `form1.vb` a `frmGraficoCirculos.vb`
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmGraficoCirculos
	Size	300,300
	StartPosition	CenterScreen
	Text	Crear Gráfico de Círculos
	WindowState	MaximizedBox
Button1	Name	btnCrearCirculos
	Location	12, 12
	Size	75, 23
	Text	Crear

- El diseño del formulario debe quedar similar al gráfico 2.35:

Gráfico 2.35: Diseño del formulario Crear Círculos



- Escribir el siguiente código en el formulario:

```
Imports System.Drawing.Drawing2D
Imports System.Threading
Imports System.Threading.Tasks
```

```
Public Class frmGrafico
```

```
    Private oAzar As New Random
```

```
    Private Function GenerarColorAzar() As Color
```

```
        Thread.Sleep(10)
```

```
        Dim R, G, B As Integer
```

```
        R = oAzar.Next(255)
```

```
        G = oAzar.Next(255)
```

```
        B = oAzar.Next(255)
```

```
        Return (Color.FromArgb(R, G, B))
```

```
    End Function
```

```
    Private Sub CrearCirculo(ByVal N As Integer)
```

```
        Dim X, Y As Integer
```

```
        Dim rec As Rectangle
```

```
        Dim deg As LinearGradientBrush
```



```
X = oAzar.Next(Me.Width)
Thread.Sleep(10)
Y = oAzar.Next(Me.Height)
rec = New Rectangle(X, Y, 50, 50)
deg = New LinearGradientBrush(rec, _
    GenerarColorAzar, GenerarColorAzar, _
    LinearGradientMode.BackwardDiagonal)
Me.CreateGraphics.FillEllipse(deg, rec)
Me.CreateGraphics.DrawString(N.ToString, _
    New Font("Courier New", 10), Brushes.White, _
    X + 15, Y + 15)
End Sub

Private Sub CrearCirculos(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCrearCirculos.Click
    Parallel.For(1, 1000, Sub(I)
        CrearCirculo(I)
    End Sub)
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.36: Ejecución del formulario Crear Círculos



Paralelismo de Tareas

El paralelismo de tareas hace referencia a la ejecución simultánea de una o varias tareas independientes. Una tarea representa una operación asincrónica y, en ciertos aspectos, se asemeja a la creación de un nuevo subproceso o elemento de trabajo ThreadPool, pero con un nivel de abstracción mayor. Las tareas proporcionan dos ventajas fundamentales:

1. Un uso más eficaz y más escalable de los recursos del sistema.
2. Un mayor control mediante programación del que se puede conseguir con un subproceso o un elemento de trabajo.

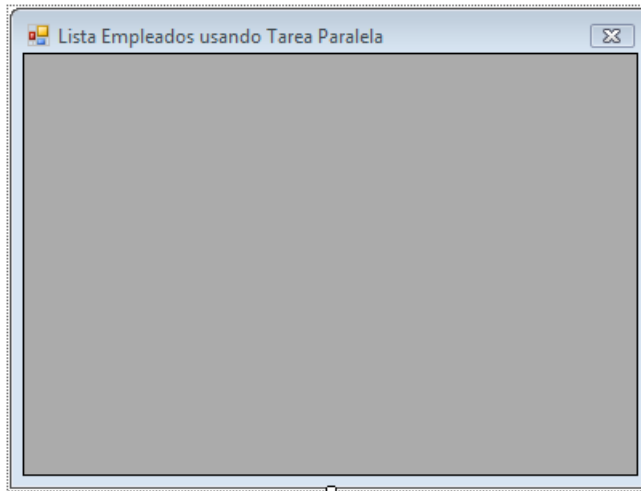
Demo 23

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo23.
- Cambiar de nombre al formulario de form1.vb a frmListaEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaEmpleado
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,300
	StartPosition	CenterScreen
	Text	Lista Empleados usando Tarea Paralela
DataGridView1	Name	dgvEmpleado
	Dock	Fill
	ReadOnly	True
	SelectionMode	FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 2.37:

Gráfico 2.37: Diseño del formulario Lista Empleados



- Escribir el siguiente código en el formulario:

Imports System.Data

Imports System.Data.SqlClient

Imports System.Threading.Tasks

Public Class frmListaEmpleado

Public Function ListarEmpleados() As DataSet

Dim dst As New DataSet

Using con As New SqlConnection("uid=sa;pwd=123456; _
data source=Lduenas\MCTS;initial catalog=Northwind")

con.Open()

Dim dap As New SqlDataAdapter("WaitFor Delay '00:00:10'; _
Select EmployeeID,LastName,FirstName From Employees", con)
dap.Fill(dst, "Empleados")

End Using 'con.Close

Return (dst)

End Function

Private Sub CargarDatos(ByVal sender As System.Object, _

ByVal e As System.EventArgs) Handles MyBase.Load

Dim dst As Task(Of DataSet) = Task.Factory.StartNew(Function()

ListarEmpleados())

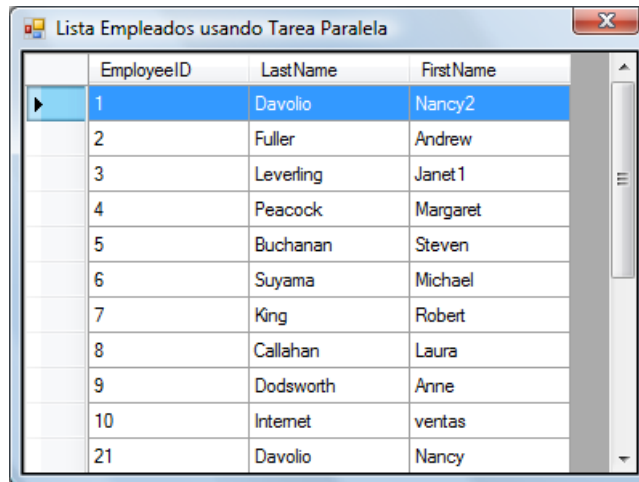
dst.Wait()

dgvEmpleado.DataSource = dst.Result.Tables(0)

End Sub
End Class

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.38: Ejecución del formulario Lista Empleados



	EmployeeID	LastName	FirstName
▶	1	Davolio	Nancy2
	2	Fuller	Andrew
	3	Leverling	Janet1
	4	Peacock	Margaret
	5	Buchanan	Steven
	6	Suyama	Michael
	7	King	Robert
	8	Callahan	Laura
	9	Dodsworth	Anne
	10	Internet	ventas
	21	Davolio	Nancy

5. Criptografía o Cifrado de Datos

La Criptografía o Cifrado de Datos permiten asegurar los datos de una aplicación cuando estos son sensitivos, es decir, cuando no se quiere que se vea su contenido, por ejemplo una planilla de sueldos. Sobre todo ahora con Internet es fácil que los datos sean interceptados por terceros, para lo cual debemos usar el cifrado o encriptación.

.NET Framework proporciona el espacio de nombres System.Security.Cryptography que contiene muchas clases que nos ayudan a realizar las principales tareas criptográficas como: cifrado simétrico, cifrado asimétrico, firmas criptográficas y valores hash criptográficos.

La diferencia entre el cifrado simétrico y el asimétrico, es que, el cifrado simétrico, usa la misma clave para encriptar y desencriptar los datos, mientras que el asimétrico, usa 2 claves: una pública y otra privada, las cuales tienen una relación matemática.

Para obtener más información sobre este tema ver la referencia 17 al final del libro.

5.1 Cifrado Simétrico por Caracter

Este tipo de cifrado es el más fácil de implementar y el menos seguro, ya que solo basta transformar cada carácter de la data original por uno nuevo mediante una transformación simple, por ejemplo, si a cada código de un caracter le sumamos un número lo estaremos cifrando y si le restamos el mismo número lo estaremos descifrando, tal como se muestra en el siguiente ejemplo.

Demo 24

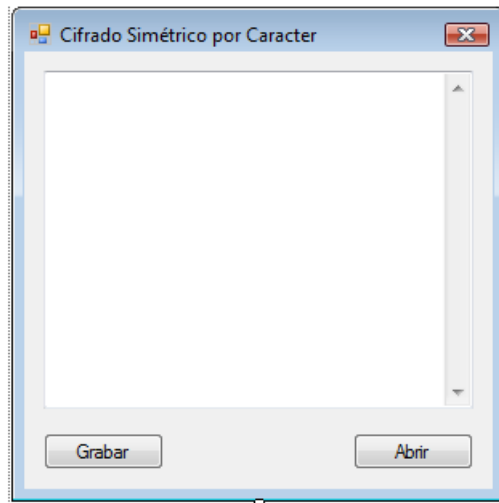
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo24.
- Cambiar de nombre al formulario de form1.vb a frmCifradoTexto.vb

- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmCifradoTexto FixedSingle False False 300,300 CenterScreen Cifrado Simétrico por Carácter
TextBox1	Name Location MultiLine ScrollBars Size	txtArchivo 12,12 True Vertical 268,211
Button1	Name Cursor Location Size Text	btnGrabar Hand 12,238 75,23 Grabar
Button2	Name Cursor Location Size Text	btnAbrir Hand 205,238 75,23 Abrir

- El diseño del formulario debe quedar similar al gráfico 2.39:

Gráfico 2.39: Diseño del formulario cifrado simétrico por caracter



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'FileStream
```

```
Public Class frmCifradoTexto
```

```
Private Sub GrabarArchivo(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnGrabar.Click  
    Dim sfd As New SaveFileDialog 'Dialogo de Guardar  
    sfd.Title = "Selecciona el archivo a guardar"  
    sfd.Filter = "Archivos de texto|*.txt"  
    If sfd.ShowDialog = Windows.Forms.DialogResult.OK Then  
        Using fs As New FileStream(sfd.FileName, FileMode.Create, _  
FileAccess.Write, FileShare.Write)  
            Dim I As Integer  
            Dim C As String  
            Dim N As Byte  
            For I = 0 To txtArchivo.Text.Length - 1  
                C = txtArchivo.Text.Substring(I, 1)  
                N = Asc(C)  
                If N < 245 Then fs.WriteByte(N + 10)  
            Next  
        End Using  
        Me.Text = sfd.FileName  
    End If  
End Sub
```

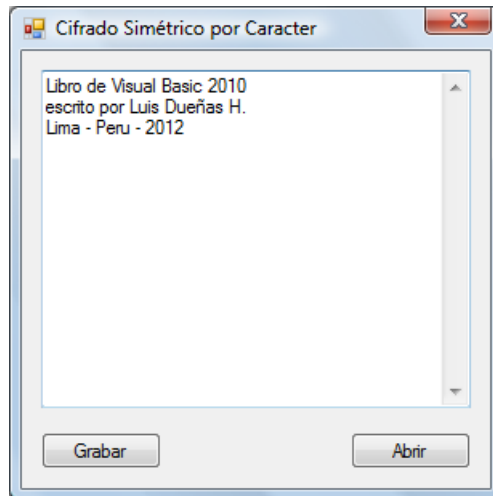


```
End If
End Sub

Private Sub AbrirArchivo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAbrir.Click
    Dim ofd As New OpenFileDialog 'Dialogo de Abrir
    ofd.Title = "Selecciona el archivo a abrir"
    ofd.Filter = "Archivos de texto|*.txt"
    If ofd.ShowDialog = Windows.Forms.DialogResult.OK Then
        Using fs As New FileStream(ofd.FileName, FileMode.Open, _
        FileAccess.Read, FileShare.Read)
            Dim I As Integer
            Dim C As String
            Dim N As Byte
            Dim S As String = ""
            Dim fi As New FileInfo(ofd.FileName)
            For I = 0 To fi.Length - 1
                N = fs.ReadByte
                C = Chr(N - 10)
                S = S + C
            Next
            txtArchivo.Text = S
        End Using
        Me.Text = ofd.FileName
    End If
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

**Gráfico 2.40: Ejecución del formulario cifrado simétrico por
caracter**



5.2. Cifrado Simétrico por Bloques

El cifrado simétrico por bloques o cifrado de clave secreta es aquel que usa la misma clave para encriptar o cifrar y para desencriptar o descifrar. Este es muy rápido y se usa en gran cantidad de datos. Entre los más populares tenemos el DES, TripleDES, Estandar de Cifrado Avanzado (CA), etc.

Todos los algoritmos de cifrado simétrico usan un vector de inicialización de datos (IV) y una clave secreta (PK), lo que los diferencia son el tamaño de ambos, mientras más grande son serán más seguros (más difíciles de descifrar).

A continuación veremos un ejemplo del uso del cifrado simétrico en .NET Framework, clases como: SymmetricAlgorithm, DESCryptoServiceProvider, TripleDESCryptoServiceProvider y RijndaelManaged.

Demo 25

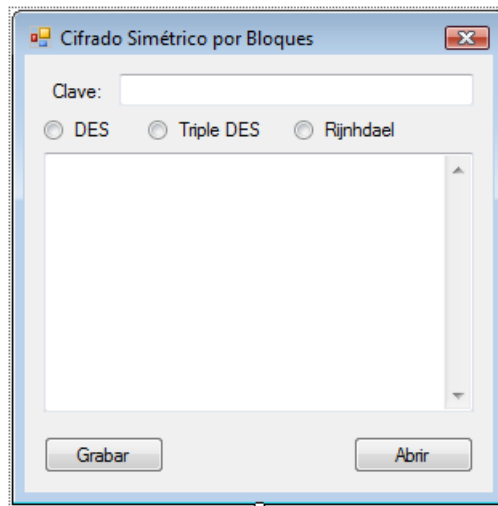
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo25.
- Cambiar de nombre al formulario de form1.vb a frmCifradoTexto.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmCifradoTexto
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	300,300
	StartPosition	CenterScreen
	Text	Cifrado Simétrico por Bloques
Label1	Name	lblClave
	AutoSize	True
	Location	15, 16
	Text	Clave:
TextBox1	Name	txtClave
	Location	59,12

	Size	221,20
RadioButton1	Name AutoSize Location Text	rbDES True 12,38 DES
RadioButton2	Name AutoSize Location Text	rbTripleDES True 77,38 Triple DES
RadioButton3	Name AutoSize Location Text	rbRijnhdael True 168,38 Rijnhdael
TextBox2	Name Location MultiLine ScrollBars Size	txtArchivo 12,61 True Vertical 268,162
Button1	Name Cursor Location Size Text	btnGrabar Hand 12,238 75,23 Grabar
Button2	Name Cursor Location Size Text	btnAbrir Hand 205,238 75,23 Abrir

- El diseño del formulario debe quedar similar al gráfico 2.41:

Gráfico 2.41: Diseño del formulario cifrado simétrico por bloques



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'FileStream, CryptoStream
```

```
Imports System.Security.Cryptography 'SymmetricAlgorithm
```

```
Imports System.Text 'Encoding
```

```
Public Class frmCifradoTexto
```

```
    Private sa As SymmetricAlgorithm
```

```
    Private clave() As Byte
```

```
    Private vectorInicio() As Byte
```

```
    Private Data() As Byte
```

```
    Private Sub GrabarArchivo(ByVal sender As System.Object, _
```

```
        ByVal e As System.EventArgs) Handles btnGrabar.Click
```

```
        Dim sfd As New SaveFileDialog 'Dialogo de Guardar
```

```
        sfd.Title = "Selecciona el archivo a guardar"
```

```
        sfd.Filter = "Archivos de texto|*.txt"
```

```
        If sfd.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
            Try
```

```
                clave = Encoding.Default.GetBytes _
```

```
                    (txtClave.Text.PadRight(sa.Key.Length, "*"))
```

```
                Data = Encoding.Default.GetBytes(txtArchivo.Text)
```

```
                sa.Key = clave
```

```
                Using fs As New FileStream(sfd.FileName, _
```

```
        FileMode.Create, FileAccess.Write, FileShare.Write)
        Using cs As New CryptoStream(fs, sa.CreateEncryptor, _
        CryptoStreamMode.Write)
            cs.Write(Data, 0, Data.Length)
        End Using
    End Using
    Me.Text = sdf.FileName
Catch ex As Exception
    MessageBox.Show("Ocurrio un error al grabar")
End Try
End If
End Sub

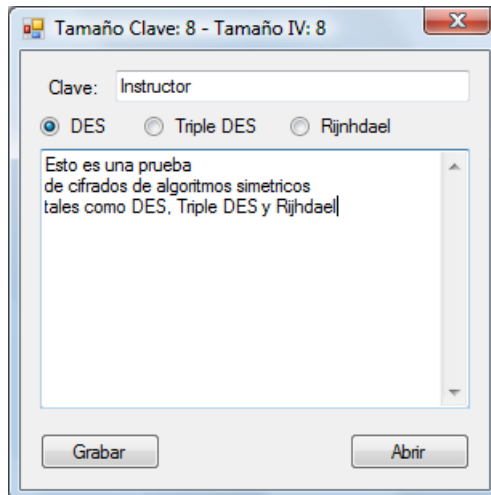
Private Sub AbrirArchivo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAbrir.Click
    Dim ofd As New OpenFileDialog 'Dialogo de Abrir
    ofd.Title = "Selecciona el archivo a abrir"
    ofd.Filter = "Archivos de texto|*.txt"
    If ofd.ShowDialog = Windows.Forms.DialogResult.OK Then
        Try
            clave = Encoding.Default.GetBytes _
            (txtClave.Text.PadRight(sa.Key.Length, "*"))
            Data = File.ReadAllBytes(ofd.FileName)
            sa.Key = clave
            Using ms As New MemoryStream
                Using cs As New CryptoStream(ms, sa.CreateDecryptor, _
                CryptoStreamMode.Write)
                    cs.Write(Data, 0, Data.Length)
                End Using
                txtArchivo.Text = Encoding.Default.GetString(ms.ToArray())
            End Using
            Me.Text = ofd.FileName
        Catch ex As Exception
            MessageBox.Show("Clave No coincide")
        End Try
    End If
End Sub

Private Sub IniciarAlgoritmoSimetrico(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles rbDES.CheckedChanged, _
```

```
rbTripleDES.CheckedChanged, rbRijndael.CheckedChanged
If rbDES.Checked Then
    sa = New DESCryptoServiceProvider
ElseIf rbTripleDES.Checked Then
    sa = New TripleDESCryptoServiceProvider
ElseIf rbRijndael.Checked Then
    sa = New RijndaelManaged
End If
Dim frase As String = "Mi mama"
vectorInicio = Encoding.Default.GetBytes _
(frase.PadRight(sa.IV.Length, "*"))
sa.IV = vectorInicio
txtClave.MaxLength = sa.Key.Length
Me.Text = String.Format("Tamaño Clave: {0} - Tamaño IV: {1}", _
sa.Key.Length, sa.IV.Length)
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.42: Ejecución del formulario cifrado simétrico por bloques



5.3. Valores Hash Criptográficos

Un valor hash es una representación numérica de un conjunto de datos y se usa para asegurar la integridad de los mensajes, es decir, comprobar que no fueron alterados.

Si a un mensaje de texto le aplicamos un algoritmo hash para obtener un valor hash o resumen y luego enviamos el mensaje con el resumen, cuando llegue a su destinatario para verificar que no fue alterado nuevamente se aplica el algoritmo y se obtiene otro resumen, si no coincide con el inicial significa que este fue alterado.

En vez de comparar cada byte de un archivo grande solo podemos comparar los resúmenes o valores hash generados para verificar archivos. En .NET existen varias clases que implementan algoritmos hash, tales como: HMACSHA1, MACTripleDES, MD5CryptoServiceProvider, RIPEMD160, SHA1Managed, SHA256Managed, SHA384Managed y SHA512Managed.

A continuación crearemos una aplicación Windows que compare 2 archivos de texto para saber si son iguales o no, usando comparación completa y también valores hash.

Demo 26

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo26.
- Cambiar de nombre al formulario de form1.vb a frmVerificarArchivos.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

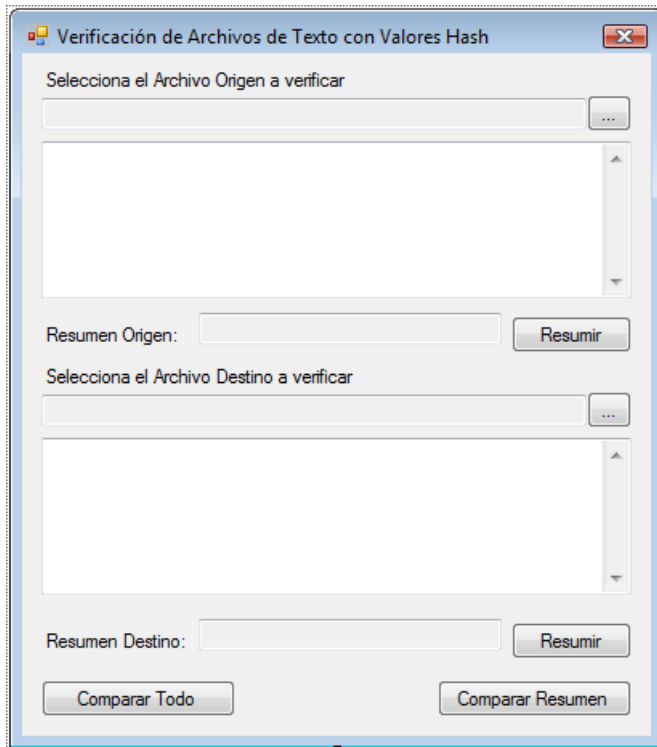
Objeto	Propiedad	Valor
Form1	Name	frmVerificarArchivos
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,460
	StartPosition	CenterScreen
	Text	Verificación de Archivos de

		Texto con Valores Hash
Label1	Name AutoSize Location Text	lblArchivoOrigen True 12, 10 Selecciona el Archivo Origen a verificar
TextBox1	Name Location ReadOnly Size	txtArchivoOrigen 12,28 True 340,20
Button1	Name Cursor Location Size Text	btnAbrirArchivoOrigen Hand 352,26 28,23 ...
TextBox2	Name Location MultiLine ScroolBars Size	txtDataOrigen 12,55 True Vertical 368,98
Label2	Name AutoSize Location Text	lblResumenOrigen True 12,169 Resumen Origen:
TextBox3	Name Location ReadOnly Size	txtResumenOrigen 110,162 True 189,20
Button2	Name Cursor Location Size Text	btnCrearResumenOrigen Hand 305,164 75,23 Resumir
Label3	Name AutoSize Location Text	lblArchivoDestino True 12,195 Selecciona el Archivo Destino a verificar
TextBox4	Name Location ReadOnly Size	txtArchivoDestino 12,213 True 340,20

Button3	Name Cursor Location Size Text	btnAbrirArchivoDestino Hand 352,211 28,23 ...
TextBox5	Name Location MultiLine ScroolBars Size	txtDataDestino 12,240 True Vertical 368,98
Label4	Name AutoSize Location Text	lblResumenDestino True 12,360 Resumen Destino:
TextBox6	Name Location ReadOnly Size	txtResumenDestino 110,353 True 189,20
Button4	Name Cursor Location Size Text	btnCrearResumenDestino Hand 305,355 75,23 Resumir
Button5	Name Cursor Location Size Text	btnCompararTodo Hand 12,391 121,23 Comparar Todo
Button6	Name Cursor Location Size Text	btnCompararResumen Hand 259,391 121,23 Comparar Resumen

- El diseño del formulario debe quedar similar al gráfico 2.43:

Gráfico 2.43: Diseño del formulario verificación valores hash



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.IO 'FileStream, StreamReader
Imports System.Text 'Encoding
Imports System.Security.Cryptography 'SHA1Managed
```

```
Public Class frmVerificarArchivos
```

```
Private Sub Abrir(ByVal txtArchivo As TextBox, ByVal txtData As TextBox)
    Dim ofd As New OpenFileDialog 'Dialogo de Abrir
    With ofd
        .Title = "Abrir Archivo de Texto"
        .Filter = "Archivo de texto|*.txt"
        If .ShowDialog = Windows.Forms.DialogResult.OK Then
            Using fs As New FileStream(.FileName, FileMode.Open, _
                FileAccess.Read, FileShare.Read)
                Using sr As New StreamReader(fs, Encoding.Default)
                    txtArchivo.Text = .FileName
                
```

```
        txtData.Text = sr.ReadToEnd
    End Using
End Using
End If
End With
End Sub
```

```
Private Sub Resumir(ByVal txtData As TextBox, _
    ByVal txtResumen As TextBox)
    Dim sha1 As New SHA1Managed
    Dim Data() As Byte = Encoding.Default.GetBytes(txtData.Text)
    Dim Resumen() As Byte = sha1.ComputeHash(Data)
    txtResumen.Text = Encoding.Default.GetString(Resumen)
End Sub
```

```
Private Sub AbrirArchivoOrigen(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnAbrirArchivoOrigen.Click
    Abrir(txtArchivoOrigen, txtDataOrigen)
End Sub
```

```
Private Sub AbrirArchivoDestino(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnAbrirArchivoDestino.Click
    Abrir(txtArchivoDestino, txtDataDestino)
End Sub
```

```
Private Sub CompararTodo(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCompararTodo.Click
    If txtDataOrigen.Text.Length = txtDataDestino.Text.Length Then
        Dim TiempoInicio As DateTime = Now
        Dim I As Integer
        Dim strDataOrigen As String = txtDataOrigen.Text
        Dim strDataDestino As String = txtDataDestino.Text
        Dim iguales As Boolean = True
        For I = 0 To strDataOrigen.Length - 1
            If strDataOrigen.Substring(I, 1) <> _
                strDataDestino.Substring(I, 1) Then
                iguales = False
                Exit For
            End If
        Next
    End If
```

```
Dim TiempoFin As DateTime = Now
Dim TiempoDemorado As TimeSpan = _
TiempoFin.Subtract(TiempoInicio)
If iguales Then
    MessageBox.Show("Archivos son iguales", _
        TiempoDemorado.Milliseconds.ToString, MessageBoxButtons.OK, _
        MessageBoxIcon.Information)
Else
    MessageBox.Show("Archivos No son iguales, por contenido
        distinto", _
        TiempoDemorado.Milliseconds.ToString, MessageBoxButtons.OK, _
        MessageBoxIcon.Exclamation)
End If
Else
    MessageBox.Show("Archivos No son iguales, por tamaño distinto",
        "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
End If
End Sub

Private Sub CrearResumenOrigen(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCrearResumenOrigen.Click
    Resumir(txtDataOrigen, txtResumenOrigen)
End Sub

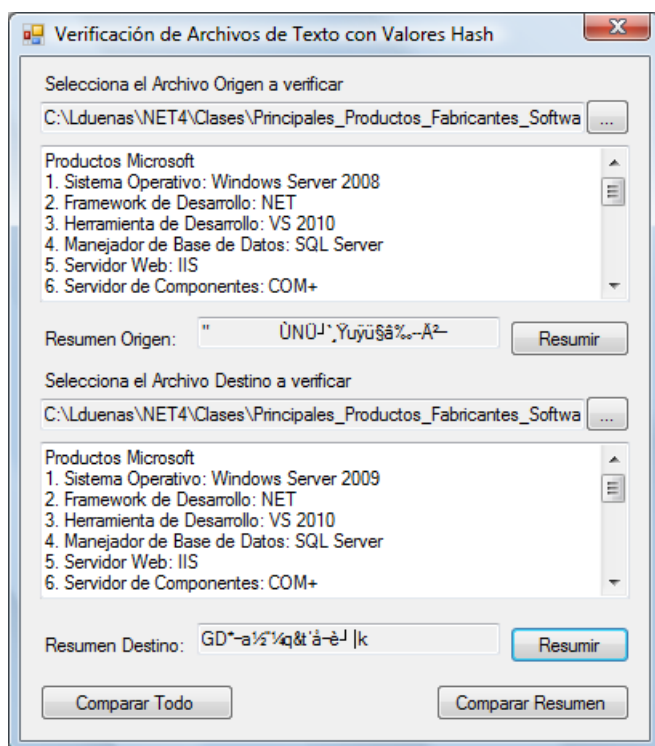
Private Sub CrearResumenDestino(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCrearResumenDestino.Click
    Resumir(txtDataDestino, txtResumenDestino)
End Sub

Private Sub CompararResumen(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCompararResumen.Click
    If txtDataOrigen.Text.Length = txtDataDestino.Text.Length Then
        Dim TiempoInicio As DateTime = Now
        Dim iguales As Boolean = (txtResumenOrigen.Text = _
            txtResumenDestino.Text)
        Dim TiempoFin As DateTime = Now
        Dim TiempoDemorado As TimeSpan = _
            TiempoFin.Subtract(TiempoInicio)
        If iguales Then
```

```
        MessageBox.Show("Archivos son iguales", _  
        TiempoDemorado.Milliseconds.ToString, _  
        MessageBoxButtons.OK, MessageBoxIcon.Information)  
    Else  
        MessageBox.Show("Archivos No son iguales, por contenido  
        distinto", _  
        TiempoDemorado.Milliseconds.ToString, MessageBoxButtons.OK, _  
        MessageBoxIcon.Exclamation)  
    End If  
Else  
    MessageBox.Show("Archivos No son iguales, por tamaño distinto", _  
    "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)  
End If  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 2.44: Ejecución del formulario verificación valores hash



Preguntas de Repaso

1. Cómo se llama la clase de .NET que obtiene información del entorno o sistema?
2. Cuáles son las 2 nuevas propiedades de la clase Environment para trabajar con sistemas y procesos de 64 bits?
3. Con qué propiedad de la clase Environment se obtiene el nombre de la PC?
4. Cómo se llama la Enumeración de la clase Environment que tiene los nombres de los directorios especiales de Windows.
5. Con qué método de la clase Environment obtenemos la ruta de un directorio especial?
6. Para qué se usa el Registro de Windows?
7. En qué espacio de nombres se encuentra la clase Registry que accede al registro de Windows.
8. Cómo se llama la clase de .NET que permite trabajar con una entrada o clave del registro de Windows.
9. Menciona 2 métodos de la clase RegistryKey.
10. Cómo se llama la clase de .NET que permite trabajar con procesos o tareas del sistema y en que espacio de nombres se encuentra?
11. Con qué método de la clase Process se obtienen todos los procesos que se encuentran actualmente en ejecución?
12. Con qué método de la clase Process se puede ejecutar un programa o aplicación?

13. Cómo se llama el espacio de nombres que permite manejar la entrada y salida?
14. Menciona 3 clases que permitan manejar entrada y salida?
15. Menciona 3 clases que permitan manejar secuencias o flujos?
- 16.Cuál es la diferencia entre una clase instanciable y otra con miembros compartidos?
17. Cuáles son los 2 métodos nuevos de la clase Directory en .NET 4?
18. Con qué clases en .NET se puede leer y escribir archivos secuencialmente?
19. Qué clase de entrada y salida debemos usar para indicar el modo de apertura de un archivo, el nivel de acceso y si estará compartido para lectura y/o escritura?
20. Con qué clase obtenemos información sobre un archivo, por ejemplo su tamaño?
21. Qué clase de .NET Framework debemos usar para obtener la ruta de un archivo, su nombre o extensión?
- 22.Cuál es la diferencia de los algoritmos de compresión Deflate y Gzip?
23. Menciona 2 métodos de la clase DeflateStream.
24. Cuáles son los servicios lógicos en los que puede clasificarse toda aplicación.
25. Cómo se dividen los Servicios de Negocio o Servicios Empresariales?
26. Qué tipo de miembros contiene una clase Entidad del Negocio?
27. Qué tipo de miembros contiene una clase Regla del Negocio?

28. Qué extensión genera un proyecto de tipo Librería de Clases?
29. Para que se usan las técnicas de programación asíncrona?
30. Menciona 3 técnicas de programación asíncrona.
31. Cuando se usa la clase Thread?
32. Menciona 3 métodos de la clase Thread.
33. Qué método de la clase Thread debe usarse para que la subtarea actual espere antes de ejecutar otra subtarea cuando hay dependencia de subtareas?
34. Con qué clase y método implementamos agrupación de subtareas para ahorrar la cantidad de hilos creados en memoria cuando son muchas llamadas?
- 35.Cuál es la forma mas simple de implementar llamadas asíncronas a un servicio web?
36. Para que se usan los delegados CallBacks?
37. Qué es la programación paralela?
38. En qué consiste el paralelismo de datos y cómo se implementa en NET?
39. En qué consiste el paralelismo de tareas y cómo se implementa en NET?
40. Cual es el espacio de nombres de .Net que se usa para implementar las tareas criptográficas?
41. Menciona las 4 principales tareas criptográficas.
42. Que és el cifrado simétrico o de clave secreta?

- 43. Menciona 3 clases de .Net que implementen cifrado simétrico?
- 44. Cuales son los 2 arreglos de bytes que se usan en el cifrado simétrico por bloques para cifrar y descifrar la data?
- 45.Cuál es el uso de los Valores Hash Criptográficos?
- 46. Menciona 3 clases que implementen algoritmos Hash cifrados?

Capítulo 3: Accediendo a Datos con ADO .NET

Inicaremos este capítulo con una breve introducción al acceso a datos en ADO .NET en la cual veremos su arquitectura, es decir sus componentes básicos que son los proveedores de datos y el DataSet.

Luego veremos como trabajar en forma conectada para acceder a diferentes orígenes de datos como MS SQL Server, MS Access, MS Excel y DBF; luego como ejecutar comandos de selección de un valor, una fila, varias filas y varios conjuntos de filas; después aprenderemos como realizar mantenimiento de datos en forma conectada mediante comandos de actualización.

Después veremos como trabajar en forma desconectada usando DataSet, es decir como crear tablas, relaciones y vistas; como operar sobre los datos: filtrar, ordenar y buscar; también como hacer mantenimiento desconectado y enviar los cambios al origen de datos.

En la cuarta parte veremos como trabajar en forma desconectada, pero usando listas de objetos en vez de DataSet, ya que consumen menos memoria y también permiten realizar todas las operaciones en forma desconectada: tales como filtro, ordenación y búsqueda. Además aprenderemos como crear un mantenimiento con objetos.

Finalmente, revisaremos LINQ para consultar datos de un DataSet, de una tabla SQL y de una lista de objetos, es decir: LINQ a DataSet, LINQ a SQL y LINQ a Entidades.

1. Trabajando en Forma Conectada

En este tema trataremos sobre como trabajar en forma conectada, es decir, realizar operaciones en línea o directamente sobre la base de datos, para lo cual la conexión debe estar abierta antes de realizar la operación, pero antes revisaremos los fundamentos del acceso a datos con ADO .NET, en especial veremos su arquitectura.

1.1. Introducción al Acceso a Datos con ADO .NET

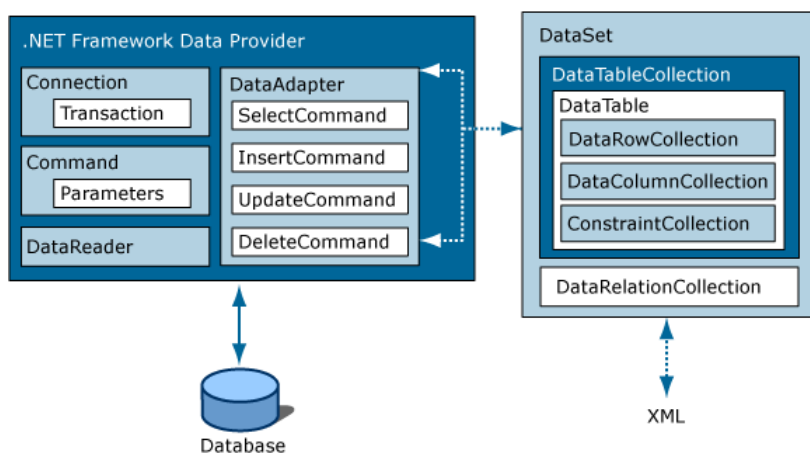
ADO .NET es el Modelo de Programación de Datos de Microsoft que se usa para acceder a diferentes orígenes de datos desde .NET Framework, es el sucesor de ADO (ActiveX Data Objects); anteriormente otros modelos usados fueron RDO (Remote Data Objects) y DAO (Data Access Objects).

La arquitectura de ADO .NET se divide en 2 componentes:

- Los Proveedores de Datos de .NET Framework: Permite conectarse a un origen de datos y manejar datos solo lectura, a través de clases como Connection, Command, DataAdapter y DataReader. Se divide en:
 - Proveedores Nativos: Acceden a un origen de datos específico como MS SQL Server (SQLClient) y Oracle (OracleClient).
 - Proveedores Generales: Acceden a diferentes orígenes de datos, a esta categoría pertenecen OLEDB y ODBC, que pueden acceder a MS Excel, MS Access, DBF, DB2, Sybase, Informix, etc.
- El DataSet: Es el repositorio desconectado, que es independiente del origen de datos y permite manejar la data en forma local y también enviar de regreso los cambios al origen de datos.

A continuación presentamos el gráfico 3.1 con la arquitectura de ADO .NET que incluye estos 2 componentes y sus principales clases relacionadas.

Gráfico 3.1: Arquitectura de ADO .NET



ADO .NET también se relaciona con otras tecnologías de datos tales como LINQ, Entity Framework, WCF Data Services y XML. Para ver más información sobre ADO .NET ver la referencia 18 al final del libro.

1.2. Conectarse a un Origen de Datos

El primer paso para conectarse a un origen de datos es conocer bien la cadena de conexión que debemos usar y esta depende del origen de datos o base de datos a la cual deseamos acceder.

Conectarse a una Base de Datos de SQL Server

Para conectarse a una Base de Datos de MS SQL Server debemos usar el espacio de nombres `System.Data.SqlClient` el cual contiene las clases `SqlConnection` para manejar la conexión a la Base de Datos: abrir y cerrar; `SqlCommand` para ejecutar comandos o instrucciones SQL: `Select`, `Insert`, `Update` o `Delete` y `SqlDataReader` para almacenar en una fila los datos que se obtengan de una base de datos.

La cadena de conexión (propiedad `ConnectionString`) para SQL Server consta de varias partes que podemos dividirla en:

- Información de autenticación: Esta depende del tipo de seguridad, que puede ser de 2 tipos:
 - Seguridad de Window: Si la seguridad es Integrada o de Windows (Single Sign On) usar lo siguiente:
 - `Integrated security=yes|true|SSPI;`
 - `Trusted_connection=yes|true|SSPI;`
 - Seguridad de SQL Server y Windows: Si la seguridad es mixta se puede usar la cadena anterior o un usuario de SQL Server:
 - `user id=usuario;password=clave`
 - `uid=usuario;pwd=clave`
- Información de Base de Datos: Esta depende del número de instancias o instalaciones que se hayan hecho en el servidor:
 - Una sola instancia: Si solo hay una instancia instalada usar:

- server=nombre_servidor;database=base_datos
- Varias instancias: Si hay varias instancias instaladas en el mismo servidor usar lo siguiente:
 - data source=nombre_servidor\ nombre_instancia;initial catalog=base_datos

A continuación presentamos un ejemplo simple de cómo conectarse a la base de datos Northwind de un servidor SQL Server llamado Lduenas que tiene una instancia llama MCTS con el usuario sa cuya clave es del 1 al 6.

Nota: En la mayoría de ejemplos del libro vamos a usar la base de datos Northwind, por tanto la cadena de conexión será la misma. Esta debe ser cambiada de acuerdo a los datos de su servidor y de preferencia hay que crear un nuevo usuario que no se el sa.

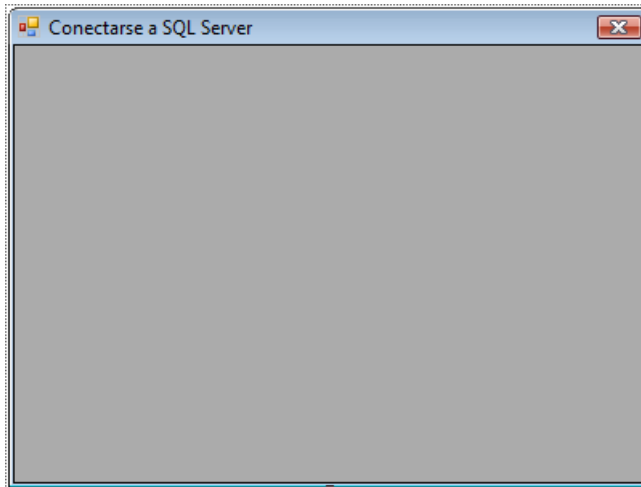
Demo 27

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo27.
- Cambiar de nombre al formulario de form1.vb a frmConexionSQL.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmConexionSQL
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,300
	StartPosition	CenterScreen
	Text	Conectarse a SQL Server
DataGridView1	Name	dgvEmpleado
	Dock	Fill
	ReadOnly	True
	SelectionMode	FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 3.2:

Gráfico 3.2: Diseño del formulario para conectarse a SQL Server



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.SqlClient 'SqlConnection, SqlCommand, SqlDataReader
```

```
Public Class frmConexionSQL
```

```
    Private Sub ListarEmpleados(ByVal sender As System.Object, _
```

```
    ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        'Definir un objeto intermediario para enlace de datos
```

```
        Dim bs As New BindingSource
```

```
        'Definir la conexion al SQL Server instalado en la PC
```

```
        Using con As New SqlConnection _
```

```
            ("uid=sa;pwd=123456;data source=Lduenas\MCTS;initial  
            catalog=Northwind")
```

```
        'Abrir la conexion
```

```
        con.Open()
```

```
        'Definir el comando con el Select a ejecutar sobre la conexion
```

```
        Dim cmd As New SqlCommand _
```

```
            ("Select EmployeeID,LastName,FirstName From Employees", con)
```

```
        'Definir y crear el DataReader al ejecutar el comando
```

```
        Dim drd As SqlDataReader = cmd.ExecuteReader
```

```
        'Enlazar el DataReader al intermediario
```

```
        bs.DataSource = drd
```

```
        'Cerrar el DataReader
```

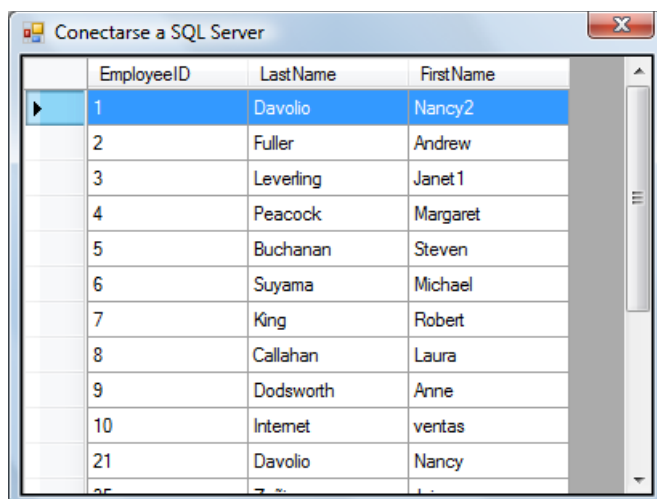


```
drd.Close()  
End Using 'Se cierra la conexion automaticamente  
'Enlazar el intermediario a la grilla  
dgvEmpleado.DataSource = bs  
End Sub  
End Class
```

Nota: En el código anterior, se ha tenido que crear un BindingSource ya que el DataReader no es enlazable al control DataGridView, en cambio el BindingSource si es enlazable.

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.3: Ejecución del formulario para conectarse a SQL Server



	EmployeeID	LastName	FirstName
▶	1	Davolio	Nancy2
	2	Fuller	Andrew
	3	Leverling	Janet1
	4	Peacock	Margaret
	5	Buchanan	Steven
	6	Suyama	Michael
	7	King	Robert
	8	Callahan	Laura
	9	Dodsworth	Anne
	10	Internet	ventas
	21	Davolio	Nancy
	25	...	...

Conectarse a una Base de Datos de MS Access

Para conectarse a una Base de Datos de MS Access usar el espacio de nombres System.Data.OleDb el cual contiene clases similares al SQLClient pero inician con el prefijo OleDb, tales como: OleDbConnection, OleDbCommand y OleDbDataReader.

La cadena de conexión para una base de datos de MS Access es la siguiente: "provider=Microsoft.Jet.OleDb.4.0;data source=base_datos.mdb" Si la base de datos tuviera clave hay que aumentar el usuario y la clave a la cadena de conexión.

A continuación crearemos una aplicación que nos permita visualizar cualquier archivo de base de datos MS Access, listando sus tablas y al seleccionar una tabla mostrar una lista con sus campos y la data que esta contiene. Esta aplicación nos será útil para examinar archivos de base de datos cuando el programa MS Access no este instalado en una PC.

Demo 28

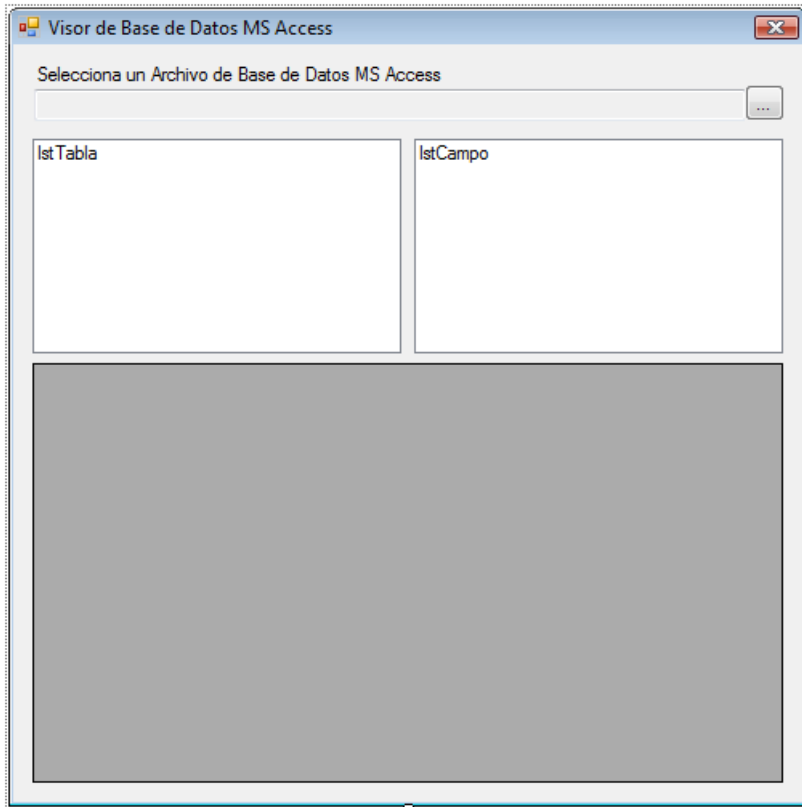
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo28.
- Cambiar de nombre al formulario de form1.vb a frmVisorAccess.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmVisorAccess FixedSingle False False 500,500 CenterScreen Visor de Base de Datos MS Access
Label1	Name AutoSize Location	lblArchivo True 12, 11

	Text	Selecciona un Archivo de Base de Datos MS Access
TextBox1	Name Location ReadOnly Size	txtArchivo 13,28 True 444,20
Button1	Name Cursor Location Size Text	btnAbrir Hand 456,25 25,23 ...
ListBox1	Name Location Size	lstTabla 12,59 230,134
ListBox2	Name Location Size	lstCampo 250,59 230,134
DataGridView1	Name Location ReadOnly SelectionMode Size	dgvData 12,199 True FullRowSelect 468,262

- El diseño del formulario debe quedar similar al gráfico 3.4:

Gráfico 3.4: Diseño del formulario Visor MS Access



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.OleDb 'OleDbConnection, OleDbCommand, _  
OleDbDataReader  
Imports System.Data
```

```
Public Class frmVisorAccess
```

```
Private con As New OleDbConnection
```

```
Private Sub ListarTablas(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAbrir.Click
```

```
Dim ofd As New OpenFileDialog
```

```
ofd.Title = "Selecciona un archivo de base de datos MS Access"
```

```
ofd.Filter = "Archivos de MS Access|*.mdb"
```

```
If ofd.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
txtArchivo.Text = ofd.FileName
```

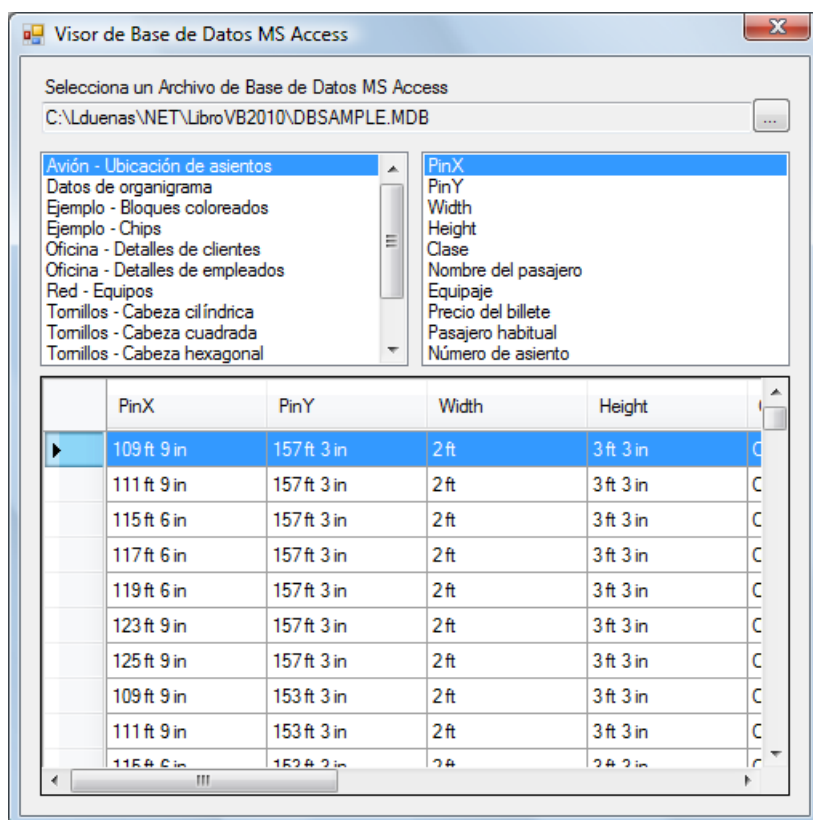
```
con = New OleDbConnection _  
    (String.Format("provider=Microsoft.Jet.Oledb.4.0;Data Source={0}", _  
        txtArchivo.Text))  
con.Open()  
lstCampo.Items.Clear()  
Dim Rest(3) As String  
Rest(3) = "Table"  
Dim Tabla As DataTable = con.GetSchema("Tables", Rest)  
lstTabla.DataSource = Tabla  
lstTabla.DisplayMember = "Table_Name"  
End If  
End Sub
```

```
Private Sub ListarCampos(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles lstTabla.SelectedIndexChanged  
    Dim cmd As New OleDbCommand _  
        (String.Format("Select * From [{0}]", lstTabla.SelectedItem(2)), con)  
    Dim drd As OleDbDataReader = cmd.ExecuteReader  
    lstCampo.Items.Clear()  
    lstCampo.BeginUpdate()  
    For I As Integer = 0 To drd.FieldCount - 1  
        lstCampo.Items.Add(drd.GetName(I))  
    Next  
    lstCampo.EndUpdate()  
    lstCampo.SelectedIndex = 0  
    Dim bs As New BindingSource  
    bs.DataSource = drd  
    dgvData.DataSource = bs  
End Sub  
End Class
```

Advertencia: El código anterior no está programado para base de datos MS Access con contraseña o clave, para ello tiene que modificarse la cadena de conexión.

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.5: Ejecución del formulario Visor MS Access



Conectarse a un Archivo de MS Excel

Para conectarse a un Archivo de MS Excel también se usa el espacio de nombres System.Data.OleDb, al igual que MS Access usa el mismo proveedor que es Microsoft Jet pero al final se configura en propiedades extendidas Excel 8.0.

La cadena de conexión para un archivo de MS Excel es la siguiente: "provider=Microsoft.Jet.OleDb.4.0;data source=archivo_excel.xls;extended properties=Excel 8.0".

Siguiendo la idea del ejemplo anterior, crearemos ahora una aplicación que permita ver cualquier archivo de MS Excel, listando las hojas de cálculo que tiene el archivo y al seleccionar una hoja mostrando sus campos y la data de la hoja.

Demo 29

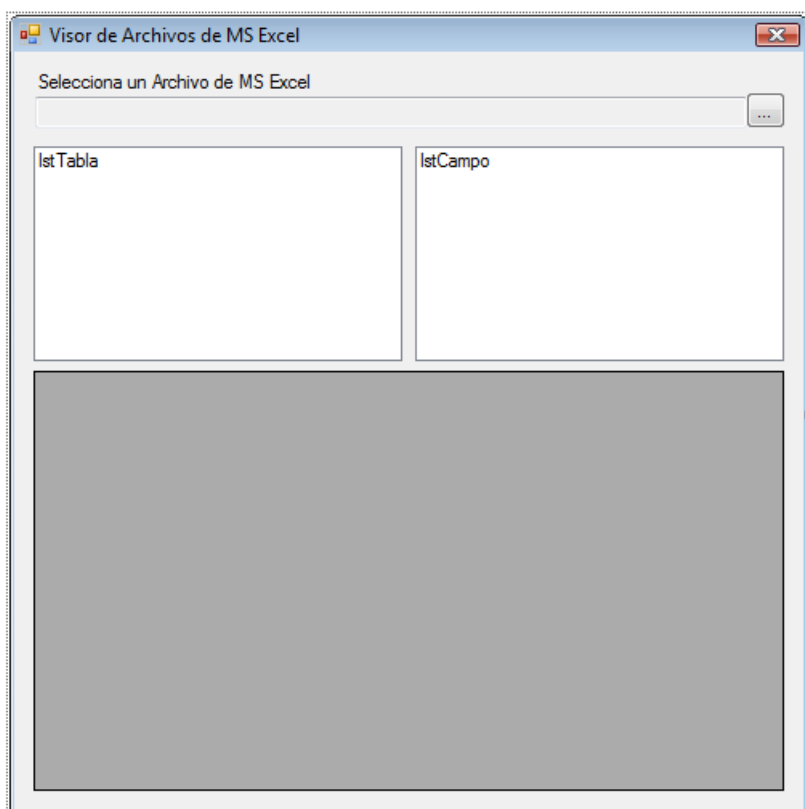
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo29.
- Cambiar de nombre al formulario de form1.vb a frmVisorExcel.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmVisorExcel FixedSingle False False 500,500 CenterScreen Visor de Archivos de MS Excel
Label1	Name AutoSize Location Text	lblArchivo True 12, 11 Selecciona un Archivo de MS Excel
TextBox1	Name	txtArchivo

	Location ReadOnly Size	13,28 True 444,20
Button1	Name Cursor Location Size Text	btnAbrir Hand 456,25 25,23 ...
ListBox1	Name Location Size	lstTabla 12,59 230,134
ListBox2	Name Location Size	lstCampo 250,59 230,134
DataGridView1	Name Location ReadOnly SelectionMode Size	dgvData 12,199 True FullRowSelect 468,262

- El diseño del formulario debe quedar similar al gráfico 3.6:

Gráfico 3.6: Diseño del formulario Visor MS Excel



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.OleDb 'OleDbConnection, OleDbCommand, _  
OleDbDataReader  
Imports System.Data
```

```
Public Class frmVisorAccess
```

```
Private con As New OleDbConnection
```

```
Private Sub ListarTablas(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAbrir.Click
```

```
Dim ofd As New OpenFileDialog
```

```
ofd.Title = "Selecciona un archivo de MS Excel"
```

```
ofd.Filter = "Archivos de MS Excel|*.xls"
```

```
If ofd.ShowDialog = Windows.Forms.DialogResult.OK Then
```

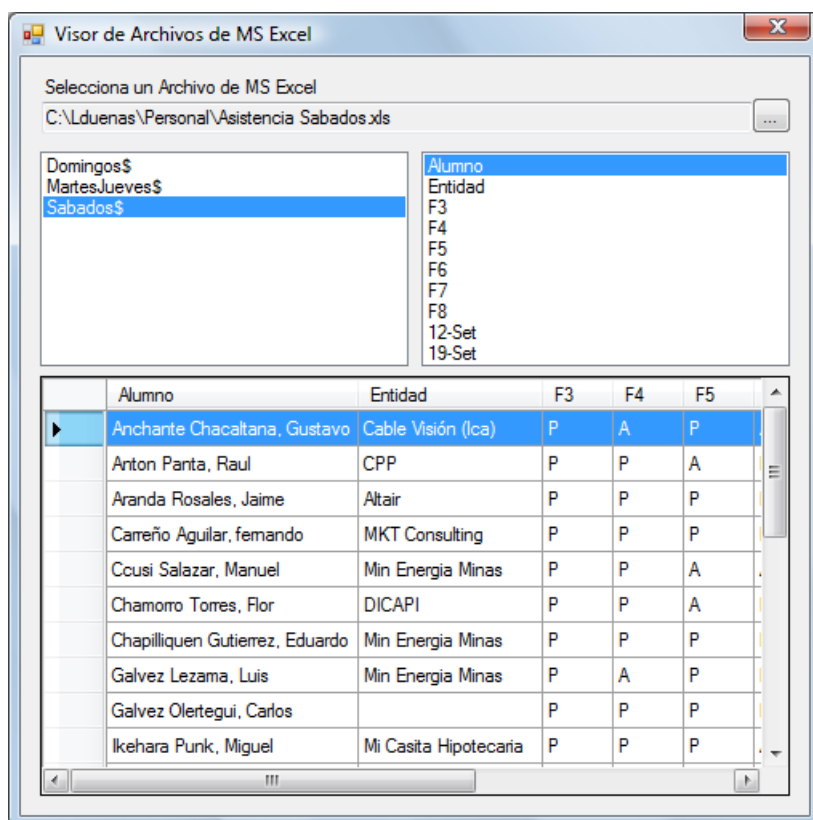
```
txtArchivo.Text = ofd.FileName
```

```
con = New OleDbConnection _  
(String.Format("provider=Microsoft.Jet.Oledb.4.0;Data Source={0};  
extended properties=Excel 8.0", txtArchivo.Text))  
con.Open()  
lstCampo.Items.Clear()  
Dim Rest(3) As String  
Rest(3) = "Table"  
Dim Tabla As DataTable = con.GetSchema("Tables", Rest)  
lstTabla.DataSource = Tabla  
lstTabla.DisplayMember = "Table_Name"  
End If  
End Sub
```

```
Private Sub ListarCampos(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles lstTabla.SelectedIndexChanged  
    Dim cmd As New OleDbCommand _  
    (String.Format("Select * From [{0}]", lstTabla.SelectedItem(2)), con)  
    Dim drd As OleDbDataReader = cmd.ExecuteReader  
    lstCampo.Items.Clear()  
    lstCampo.BeginUpdate()  
    For I As Integer = 0 To drd.FieldCount - 1  
        lstCampo.Items.Add(drd.GetName(I))  
    Next  
    lstCampo.EndUpdate()  
    lstCampo.SelectedIndex = 0  
    Dim bs As New BindingSource  
    bs.DataSource = drd  
    dgvData.DataSource = bs  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.7: Ejecución del formulario Visor MS Excel



Conectarse a un Archivo DBF

Los archivos de bases de datos (DBF) es un formato antiguo usado por muchas empresas que contienen solo una tabla que puede estar indexada. A veces es necesario obtener datos de este tipo de formato para lo cual también tenemos que usar el espacio de nombres System.Data.OleDb.

La cadena de conexión para un archivo DBF de DBase, FoxBase, Clipper, etc, es la siguiente: "provider=Microsoft.Jet.Oledb.4.0;data source=ruta_archivo_dbf;extended properties=Dbase III".

Al momento de especificar el comando Select ira el nombre del archivo (físico) como nombre de tabla (lógico).

Advertencia: El nombre del archivo DBF no debe tener más de 8 caracteres, de lo contrario se originaría un error al acceder al archivo.

A continuación veremos como leer los datos del archivo Ubigeo.dbf provisto por el INEI (Instituto Nacional de Estadística e Informática) que contiene los distritos de las diferentes provincias de los departamentos del Perú en una sola tabla o archivo. Para descargar el archivo ver la referencia 19 al final del libro.

Ademas de leer datos del DBF también el ejemplo nos muestra como copiar los datos hacia SQL Server en forma masiva usando la clase SqlBulkCopy que es muy eficiente para copiar gran cantidad de datos en vez de insertar fila x fila que consume muchos recursos y demora demasiado.

Nota: Es necesario para realizar la copia masiva que se haya creado la tabla Ubigeo en la base de datos Northwind con 4 campos: CodDpto char(2), CodProv char(2), CodDist char(2) y Nombre varchar(200).

Demo 30

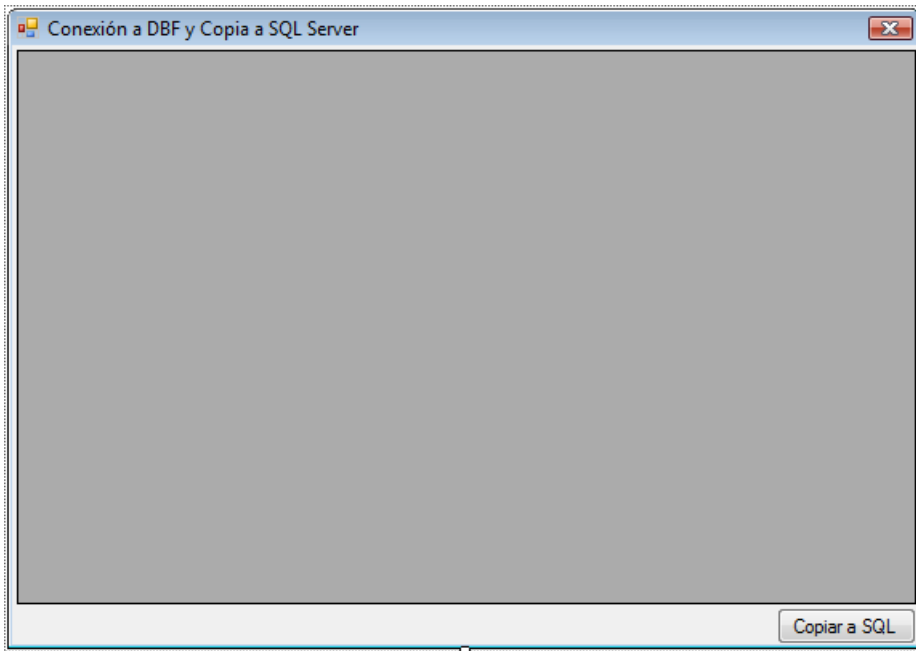
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo30.
- Cambiar de nombre al formulario de form1.vb a frmConexionDBF.vb

- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmConexionDBF FixedSingle False False 570,400 CenterScreen Conexión a DBF y Copia a SQL Server
DataGridView1	Name AutoSizeColumnsMode Location ReadOnly SelectionMode Size	dgvUbigeo AllCells 3,3 True FullRowSelect 561,346
Button1	Name Cursor Location Size Text	btnCopiarA_SQL Hand 477,351 87, 23 Copiar a SQL

- El diseño del formulario debe quedar similar al gráfico 3.8:

Gráfico 3.8: Diseño del formulario conexión a DBF y copia a SQL



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.OleDb 'Origen: DBF
```

```
Imports System.Data.SqlClient 'Destino: SQL Server
```

```
Public Class frmConexionDBF
```

```
    Private Tabla As New DataTable
```

```
    Private Sub ListarUbigeo(ByVal sender As System.Object, _  
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Using con As New OleDbConnection _
```

```
            ("provider=Microsoft.Jet.Oledb.4.0; data source=  
            C:\Lduenas\NET\LibroVB2010;extended properties=Dbase III")
```

```
        con.Open()
```

```
        Dim dap As New OleDbDataAdapter("Select * From Ubigeo", con)
```

```
        dap.Fill(Tabla)
```

```
    End Using
```

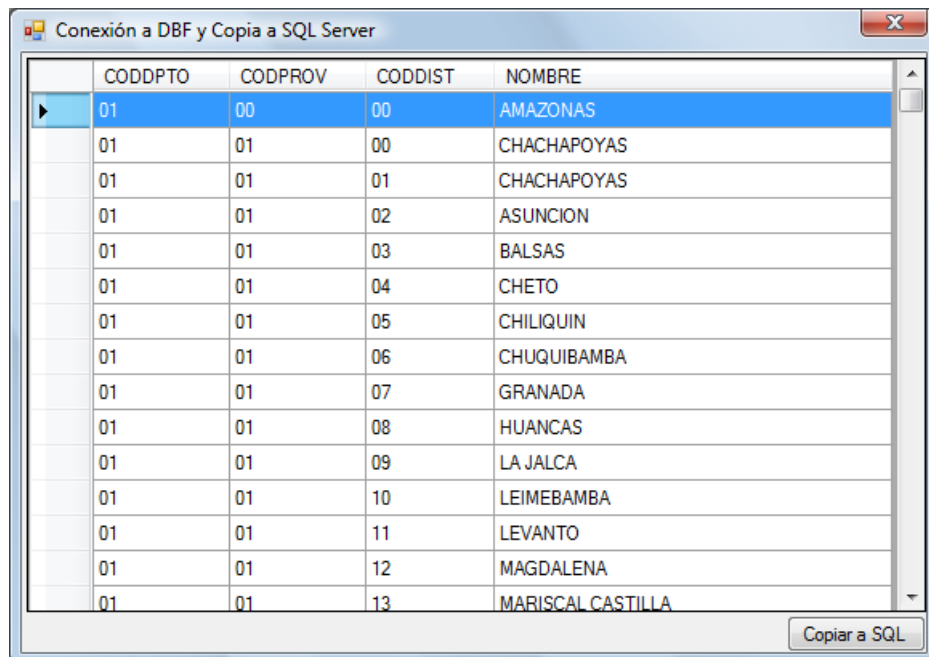
```
    dgvUbigeo.DataSource = Tabla
```

```
End Sub
```

```
Private Sub CopiarA_SQL(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnCopiarA_SQL.Click
    Dim TiempoInicio As DateTime = Now
    Using con As New SqlConnection _
        ("uid=sa;pwd=123456;data source=Lduenas\MCTS;
        database=Northwind")
        con.Open()
        Dim bc As New SqlBulkCopy(con)
        bc.DestinationTableName = "Ubigeo"
        bc.WriteToServer(Tabla)
    End Using
    Dim TiempoFin As DateTime = Now
    Dim Duracion As TimeSpan = TiempoFin.Subtract(TiempoInicio)
    MessageBox.Show(Duracion.TotalMilliseconds)
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.9: Ejecución del formulario conexión a DBF y copia a SQL



	CODDPTO	CODPROV	CODDIST	NOMBRE
▶	01	00	00	AMAZONAS
	01	01	00	CHACHAPOYAS
	01	01	01	CHACHAPOYAS
	01	01	02	ASUNCION
	01	01	03	BALSAS
	01	01	04	CHETO
	01	01	05	CHILQUIN
	01	01	06	CHUQUIBAMBA
	01	01	07	GRANADA
	01	01	08	HUANCAS
	01	01	09	LA JALCA
	01	01	10	LEIMBAMBA
	01	01	11	LEVANTO
	01	01	12	MAGDALENA
	01	01	13	MARISCAL CASTILLA

En esta primera parte solo hemos visto como conectarse a MS SQL Server, MS Access, MS Excel y DBF. Para ver más información de cómo conectarse a cualquier otros orígenes de datos ver la referencia 20 al final del libro.

1.3. Ejecutando Comandos de Selección

En esta segunda parte aprenderemos como ejecutar comandos que devuelvan datos de una base de datos SQL Server, para lo cual usaremos las clases: SqlConnection, SqlCommand y SqlDataReader.

Primero veremos como traer un solo dato mediante el método ExecuteScalar de la clase SqlCommand, luego como traer una fila con el método ExecuteReader usando el parámetro commandBehaviour.singleRow, finalmente como traer varias filas usando el parámetro commandBehaviour.SingleResult y traer varios conjuntos de filas usando el método ExecuteReader sin parametros.

Ejecutar comandos de Selección que devuelvan un solo valor

Si deseamos ejecutar un comando o instrucción SQL Select que devuelve un simple valor, que puede ser una cadena, un número o cualquier tipo de dato, se usa el método ExecuteScalar de la clase SqlCommand.

A continuación un ejemplo que devuelve el nombre completo del empleado ingresando su código.

Demo 31

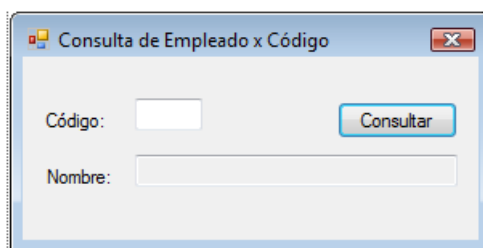
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo31.
- Cambiar de nombre al formulario de form1.vb a frmConsultaEmpleado .vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name AcceptButton FormBorderStyle	frmConsultaEmpleado btnConsultar FixedSingle

	MaximizeBox MinimizeBox Size StartPosition Text	False False 300,150 CenterScreen Consulta de Empleado x Código
Label1	Name AutoSize Location Text	lblCodigo True 12, 33 Código:
TextBox1	Name Location MaxLength Size	txtCodigo 70,26 2 42,20
Button1	Name Cursor Enabled Location Size Text	btnConsultar Hand False 196,28 75,23 Consultar
Label2	Name AutoSize Location Text	lblNombre True 12, 68 Nombre:
TextBox2	Name Location ReadOnly Size	txtNombre 70, 61 True 201,20

- El diseño del formulario debe quedar similar al gráfico 3.10:

Gráfico 3.10: Diseño del formulario Consulta Empleado



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.SqlClient 'SqlConnection, SqlCommand
```

```
Public Class frmConsultaEmpleado
```

```
Private Sub Validarigitos(ByVal sender As Object, _  
ByVal e As System.Windows.Forms.KeyPressEventArgs) _  
Handles txtCodigo.KeyPress  
    e.Handled = Not (Char.IsDigit(e.KeyChar) Or _  
    e.KeyChar = ControlChars.Back)  
End Sub
```

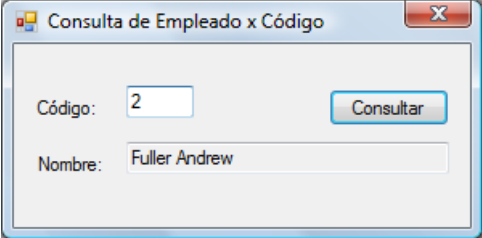
```
Private Sub HabilitarBoton(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles txtCodigo.TextChanged  
    btnConsultar.Enabled = (txtCodigo.Text <> "")  
End Sub
```

```
Private Sub ConsultarNombre(ByVal sender As Object, _  
ByVal e As System.EventArgs) Handles btnConsultar.Click  
    Using con As New SqlConnection("uid=sa;pwd=123456;  
data source=Lduenas\MCTS;initial catalog=Northwind")  
        con.Open()  
        Dim SQL As String = String.Format _  
("Select LastName + ' ' + FirstName From Employees Where  
EmployeeID={0}", txtCodigo.Text)  
        Dim cmd As New SqlCommand(SQL, con)  
        Dim oNombre As Object = cmd.ExecuteScalar  
        If oNombre IsNot Nothing Then  
            txtNombre.Text = CType(oNombre, String)  
        Else  
            txtNombre.Clear()  
            MessageBox.Show("Código No existe")  
        End If  
    End Using  
End Sub  
End Class
```

Nota: En el código anterior No es necesario controlar el nulo ya que el campo Apellido y Nombre del Empleado es obligatorio.

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.11: Ejecución del formulario Consulta Empleado



Consulta de Empleado x Código

Código: 2 Consultar

Nombre: Fuller Andrew

Ejecutar comandos de Selección que devuelvan una fila

Si deseamos ejecutar un comando o instrucción SQL Select que devuelve un registro o una fila, se usa el método ExecuteReader de la clase SqlCommand seguido del parámetro CommandBehaviour.SingleRow.

A continuación un ejemplo que devuelve el apellido, nombre, fecha de nacimiento y la foto del empleado ingresando su código.

Nota: La tabla Employees de la BD Northwind tiene un campo Photo de tipo Image pero los registros ingresados son Bitmaps, para ver la foto hay que ingresar nuevos registros agregando arreglo de bytes. En el ejemplo se ha agregado una foto para el registro 102.

Demo 32

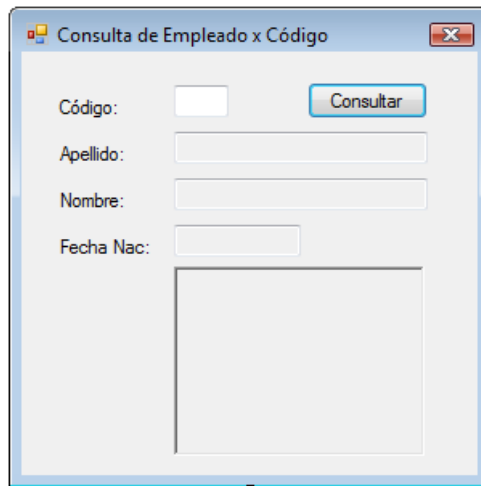
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo32.
- Cambiar de nombre al formulario de form1.vb a frmConsultaEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name AcceptButton FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmConsultaEmpleado btnConsultar FixedSingle False False 300,300 CenterScreen Consulta de Empleado x Código
Label1	Name AutoSize Location Text	lblCodigo True 21, 28 Código:
TextBox1	Name	txtCodigo

	Location MaxLength Size	95,21 3 34,20
Button1	Name Cursor Enabled Location Size Text	btnConsultar Hand False 178,19 75,23 Consultar
Label2	Name AutoSize Location Text	lblApellido True 21,57 Apellido:
TextBox2	Name Location ReadOnly Size	txtApellido 95, 50 True 158,20
Label3	Name AutoSize Location Text	lblNombre True 21,86 Nombre:
TextBox3	Name Location ReadOnly Size	txtNombre 95, 79 True 158,20
Label4	Name AutoSize Location Text	lblFechaNac True 21, 115 Fecha Nac:
TextBox4	Name Location ReadOnly Size	txtFechaNac 95, 108 True 79,20
PictureBox1	Name BorderStyle Location Size SizeMode	picFoto Fixed3D 95, 134 156, 118 StretchImage

- El diseño del formulario debe quedar similar al gráfico 3.12:

Gráfico 3.12: Diseño del formulario Consulta Empleado



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.SqlClient 'SqlConnection, SqlCommand, SqlDataReader
Imports System.IO 'File
```

```
Public Class frmConsultaEmpleado
```

```
    Dim Curriculum() As Byte
```

```
    Private Sub ValidarDigitos(ByVal sender As Object, _
        ByVal e As System.Windows.Forms.KeyPressEventArgs) _
        Handles txtCodigo.KeyPress
        e.Handled = Not (Char.IsDigit(e.KeyChar) Or _
        e.KeyChar = ControlChars.Back)
    End Sub
```

```
    Private Sub HabilitarBoton(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles txtCodigo.TextChanged
        btnConsultar.Enabled = (txtCodigo.Text <> "")
    End Sub
```

```
    Private Sub ConsultarEmpleado(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles btnConsultar.Click
        Using con As New SqlConnection _
            ("uid=sa;pwd=123456;data source=Lduenas\MCTS;
            initial catalog=Northwind")
```

```
con.Open()
Dim SQL As String = String.Format _
("Select LastName,FirstName,BirthDate,Photo From Employees
Where EmployeeID={0}", txtCodigo.Text)
Dim cmd As New SqlCommand(SQL, con)
Dim drd As SqlDataReader = _
cmd.ExecuteReader(CommandBehavior.SingleRow)
If drd IsNot Nothing Then
    If drd.HasRows Then
        drd.Read()
        txtApellido.Text = drd.GetString(0)
        txtNombre.Text = drd.GetString(1)
        txtFechaNac.Text = If(drd.IsDBNull(2), "", _
        drd.GetDateTime(2).ToString("d"))
        Dim Foto() As Byte = If(drd.IsDBNull(3), Nothing, _
        drd.GetValue(3))
        If Foto IsNot Nothing AndAlso Foto.Length > 0 Then
            Dim ms As New MemoryStream(Foto)
            Try
                picFoto.Image = Image.FromStream(ms)
            Catch ex As Exception
                picFoto.Image = Nothing
            End Try
        Else
            picFoto.Image = Nothing
        End If
    Else
        txtApellido.Clear()
        txtNombre.Clear()
        txtFechaNac.Clear()
        picFoto.Image = Nothing
        MessageBox.Show("Código No existe")
    End If
End If
End Using
End Sub
End Class
```

Nota: En el código anterior Si es necesario controlar los nulos ya que los campos Fecha de Nacimiento (BirthDate) y Foto (Photo) del Empleado

soportan nulos y se pueden caer si se muestra en el TextBox y el PictureBox y no tienen valores. Para controlar los nulos se esta usando el método IsDBNull del DataReader.

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.13: Ejecución del formulario Consulta Empleado



Consulta de Empleado x Código

Código: 102 Consultar

Apellido: Mona

Nombre: Chita

Fecha Nac: 01/01/2000



Ejecutar comandos de Selección que devuelvan varias filas

Para ejecutar un comando o instrucción SQL Select que devuelve varios registros o filas, se usa el método ExecuteReader de la clase SqlCommand seguido del parámetro CommandBehaviour.SingleResult.

A continuación un ejemplo que devuelve el apellido, nombre, fecha de nacimiento y la foto del empleado ingresando su código.

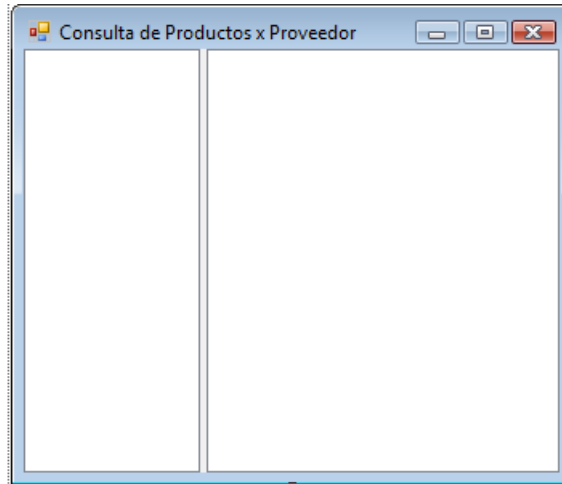
Demo 33

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo33.
- Cambiar de nombre al formulario de form1.vb a frmProductosx Proveedor.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmProductosxProveedor FixedSingle False False 350,300 CenterScreen Consulta de Productos x Proveedor
ImageList1	Name Images	ilsProveedor Agregar 4 imágenes
SplitContainer1	Name Dock	scConsulta Fill
TreeView1 (izquierda)	Name Dock	tvwProveedor Fill
ListView1 (derecha)	Name Dock FullRowSelect GridLines HotTracking	lvwProducto Fill True True True

- El diseño del formulario debe quedar similar al gráfico 3.14:

Gráfico 3.14: Diseño del formulario Consulta de Productos x Proveedor



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.SqlClient 'SqlConnection, SqlCommand, SqlDataReader
```

```
Public Class frmProductosxProveedor
```

```
Private strConexion As String = _  
    "uid=sa;pwd=123456;data source=Lduenas\MCTS;initial  
    catalog=Northwind"
```

```
Private Sub ListarProveedores(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    'Configurar el TreeView
```

```
    Dim nodoRaiz As TreeNode = _  
    twvProveedor.Nodes.Add("P", "Proveedor", 0, 1)
```

```
    Using con As New SqlConnection(strConexion)  
        con.Open()  
        Dim cmd As New SqlCommand _  
        ("Select SupplierID,CompanyName From Suppliers Order By 2", con)  
        Dim drd As SqlDataReader = _  
        cmd.ExecuteReader(CommandBehavior.SingleResult)  
        If drd IsNot Nothing Then
```

```
Dim pos_Codigo As Integer = drd.GetOrdinal("SupplierID")
Dim pos_Nombre As Integer = drd.GetOrdinal("CompanyName")
Do While drd.Read
    nodoRaiz.Nodes.Add(drd.GetInt32(pos_Codigo).ToString, _
        drd.GetString(pos_Nombre), 2, 3)
Loop
End If
End Using
'Configurar el ListView
With lvwProducto
    .Columns.Add("Codigo", 60, HorizontalAlignment.Right)
    .Columns.Add("Nombre", 200, HorizontalAlignment.Left)
    .Columns.Add("Precio", 60, HorizontalAlignment.Right)
    .Columns.Add("Stock", 60, HorizontalAlignment.Right)
    .View = View.Details
End With
End Sub
```

```
Private Sub ListarProductosxProveedor(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.TreeViewEventArgs) _
Handles twwProveedor.AfterSelect
    lvwProducto.Items.Clear()
    If e.Node.Level = 1 Then
        Using con As New SqlConnection(strConexion)
            con.Open()
            Dim SQL As String = String.Format _
                ("Select ProductID,ProductName,IsNull(UnitPrice,0) As _
                Precio,IsNull(UnitsInStock,0) As Stock From Products Where _
                SupplierID={0} Order By 1", e.Node.Name)
            Dim cmd As New SqlCommand(SQL, con)
            Dim drd As SqlDataReader = _
                cmd.ExecuteReader(CommandBehavior.SingleResult)
            If drd IsNot Nothing Then
                Dim Fila As ListViewItem
                Do While drd.Read
                    Fila = lvwProducto.Items.Add(drd.GetInt32(0))
                    Fila.SubItems.Add(drd.GetString(1))
                    Fila.SubItems.Add(drd.GetDecimal(2).ToString("n2"))
                    Fila.SubItems.Add(drd.GetInt16(3))
                Loop
            End If
        End Using
    End If
End Sub
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Ejecutar comandos de Selección que devuelvan varios conjuntos de filas

Muchas veces es necesario cargar varias tablas en una aplicación, y tradicionalmente ejecutamos varios comandos o instrucciones SQL para realizar este cometido, sin embargo, con un solo comando con varios Select o un procedimiento almacenado que tenga varios Select se puede realizar lo mismo.

Para ejecutar un comando o instrucción SQL Select que devuelve varios conjuntos de registros o filas, se usa el método ExecuteReader de la clase SqlCommand sin parámetros, por defecto se obtiene el primer Select, para obtener el siguiente conjunto de registros se usa el método NextResult del DataReader.

A continuación un ejemplo que consulta 5 tablas de la base de datos Northwind: Productos, Empleados, Clientes, Proveedores y Categorías usando un solo comando y un solo DataReader (fila).

Demo 34

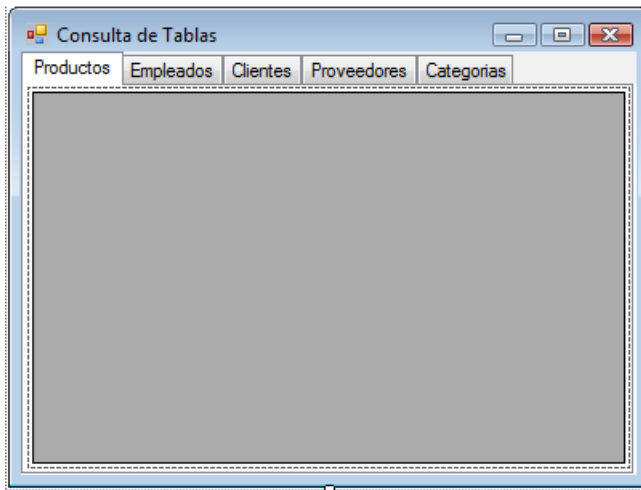
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo34.
- Cambiar de nombre al formulario de form1.vb a frmConsultaTablas.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmConsultaTablas
	Size	400,300
	Text	Consulta de Tablas
	WindowState	Maximized
TabControl1	Name	tabConsulta
	Dock	Fill
	TabPage1	Productos
	TabPage2	Empleados
	TabPage3	Clientes

	TabPage4 TabPage5	Proveedores Categorias
DataGridView1	Container Name Dock ReadOnly SelectionMode	TabPage1 dgvProducto Fill True FullRowSelect
DataGridView2	Container Name Dock ReadOnly SelectionMode	TabPage2 dgvEmpleado Fill True FullRowSelect
DataGridView3	Container Name Dock ReadOnly SelectionMode	TabPage3 dgvCliente Fill True FullRowSelect
DataGridView4	Container Name Dock ReadOnly SelectionMode	TabPage4 dgvProveedor Fill True FullRowSelect
DataGridView5	Container Name Dock ReadOnly SelectionMode	TabPage5 dgvCategoria Fill True FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 3.16:

Gráfico 3.16: Diseño del formulario Consulta de Tablas



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.SqlClient 'SqlConnection, SqlCommand, SqlDataReader
```

```
Public Class frmConsultaTablas
```

```
Private Sub CargarDatos(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles MyBase.Load  
    Dim bs1 As New BindingSource  
    Dim bs2 As New BindingSource  
    Dim bs3 As New BindingSource  
    Dim bs4 As New BindingSource  
    Dim bs5 As New BindingSource  
    Using con As New SqlConnection _  
        ("uid=sa;pwd=123456;data source=Lduenas\MCTS;initial  
        catalog=Northwind")  
        con.Open()  
        Dim cmd As New SqlCommand("Select * From Products;  
        Select * From Employees; Select * From Customers;  
        Select * From Suppliers; Select * From Categories", con)  
        Dim drd As SqlDataReader = cmd.ExecuteReader  
        bs1.DataSource = drd  
        drd.NextResult()  
        bs2.DataSource = drd  
        drd.NextResult()
```



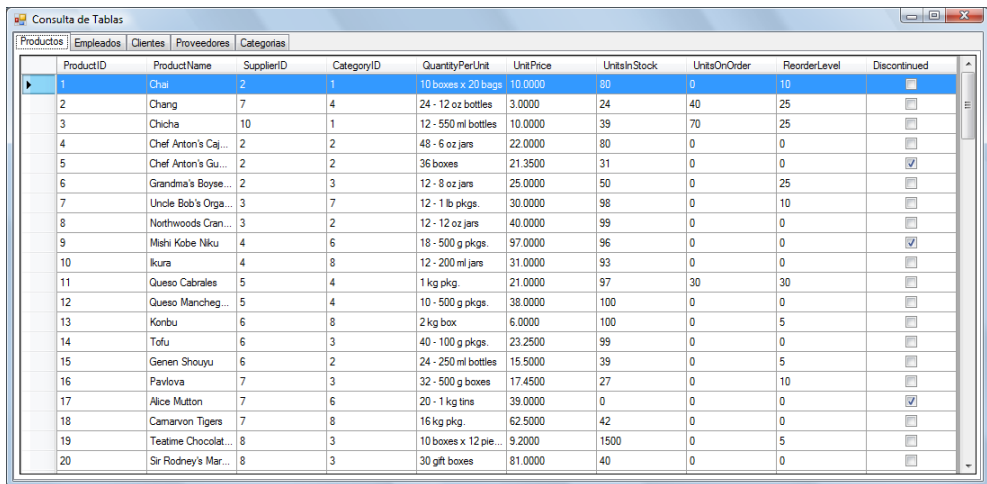
```

bs3.DataSource = drd
drd.NextResult()
bs4.DataSource = drd
drd.NextResult()
bs5.DataSource = drd
drd.Close()
End Using
dgvProducto.DataSource = bs1
dgvEmpleado.DataSource = bs2
dgvCliente.DataSource = bs3
dgvProveedor.DataSource = bs4
dgvCategoria.DataSource = bs5
End Sub
End Class

```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.17: Ejecución del formulario Consulta de Tablas



The screenshot shows a Windows application window titled 'Consulta de Tablas'. It has a tabbed interface with tabs for 'Productos', 'Empleados', 'Clientes', 'Proveedores', and 'Categorías'. The 'Productos' tab is active, displaying a table with 10 columns: ProductID, ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, and Discontinued. The table contains 20 rows of product data, including items like 'Chai', 'Chang', 'Chicha', and 'Queso Cabrales'.

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	Chai	2	1	10 boxes x 20 bags	10.0000	80	0	10	☐
2	Chang	7	4	24 - 12 oz bottles	3.0000	24	40	25	☐
3	Chicha	10	1	12 - 550 ml bottles	10.0000	39	70	25	☐
4	Chef Anton's Caj...	2	2	48 - 6 oz jars	22.0000	80	0	0	☐
5	Chef Anton's Gu...	2	2	36 boxes	21.3500	31	0	0	☒
6	Grandma's Boyse...	2	3	12 - 8 oz jars	25.0000	50	0	25	☐
7	Uncle Bob's Orga...	3	7	12 - 1 lb pkgs.	30.0000	98	0	10	☐
8	Northwoods Cran...	3	2	12 - 12 oz jars	40.0000	99	0	0	☐
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.0000	96	0	0	☒
10	Ikura	4	8	12 - 200 ml jars	31.0000	93	0	0	☐
11	Queso Cabrales	5	4	1 kg pkg	21.0000	97	30	30	☐
12	Queso Mancheg...	5	4	10 - 500 g pkgs.	38.0000	100	0	0	☐
13	Konbu	6	8	2 kg box	6.0000	100	0	5	☐
14	Tofu	6	3	40 - 100 g pkgs.	23.2500	99	0	0	☐
15	Genen Shouyu	6	2	24 - 250 ml bottles	15.5000	39	0	5	☐
16	Pavlova	7	3	32 - 500 g boxes	17.4500	27	0	10	☐
17	Alice Mutton	7	6	20 - 1 kg tins	39.0000	0	0	0	☒
18	Carnarvon Tigers	7	8	16 kg pkg	62.5000	42	0	0	☐
19	Teatime Chocolat...	8	3	10 boxes x 12 pie...	9.2000	1500	0	5	☐
20	Sir Rodney's Mar...	8	3	30 gift boxes	81.0000	40	0	0	☐

1.4. Realizando un Mantenimiento Conectado

En esta tercera parte aprenderemos como ejecutar comandos que inserten, actualizen y eliminen datos mediante el método `ExecuteNonQuery` de la clase `Command`. Este método devuelve un número que indica la cantidad de registros afectados.

A continuación, crearemos un mantenimiento de la tabla Empleados, es decir veremos como adicionar, actualizar y eliminar registros en dicha tabla.

Demo 35

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo35.
- Cambiar de nombre al formulario de `form1.vb` a `frmMantenimiento Empleado.vb`
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmMantenimientoEmpleado
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,400
	StartPosition	CenterScreen
	Text	Mantenimiento de Empleados Conectado
Label1	Name	lblCodigo
	AutoSize	True
	Location	26, 33
	Text	Código:
TextBox1	Name	txtCodigo
	Location	100,26
	ReadOnly	True
	Size	47,20

Label2	Name AutoSize Location Text	lblApellido True 26, 59 Apellido:
TextBox2	Name Location Size	txtApellido 100,52 178,20
Label3	Name AutoSize Location Text	lblNombre True 26, 85 Nombre:
TextBox3	Name Location Size	txtNombre 100,78 178,20
Label4	Name AutoSize Location Text	lblFechaNac True 26, 111 Fecha Nac:
DateTimePicker1	Name Format Location Size	dtpFechaNac Short 100,105 88,20
Button1	Name Cursor Location Size Text	btnNuevo Hand 297, 22 75, 23 Nuevo
Button2	Name Cursor Location Size Text	btnAdicionar Hand 297, 22 75, 50 Adicionar
Button3	Name Cursor Location Size Text	btnActualizar Hand 297, 78 75, 23 Actualizar
Button4	Name Cursor Location Size Text	btnEliminar Hand 297, 106 75, 23 Eliminar

DataGridView1	Name	dgvEmpleado
	Location	13, 151
	MultiSelect	False
	ReadOnly	True
	SelectionMode	FullRowSelect
	Size	367, 210

- El diseño del formulario debe quedar similar al gráfico 3.18:

Gráfico 3.18: Diseño del formulario Mantenimiento de Empleados

- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data.SqlClient 'SqlConnection, SqlCommand, SqlDataReader
Imports System.Drawing.Drawing2D 'LinearGradientBrush
```

```
Public Class frmMantenimientoEmpleado
    Private strConexion As String = _
        "uid=sa;pwd=123456;data source=Lduenas\MCTS;
        initial catalog=Northwind"
    Private bs As New BindingSource
    Private N As Integer
```

```
Private Sub DibujarRecuadro(ByVal sender As Object, _
ByVal e As System.Windows.Forms.PaintEventArgs) Handles Me.Paint
    Dim recData As New Rectangle(15, 10, 270, 130)
    Dim degData As New LinearGradientBrush _
(recData, Color.Aqua, Color.Blue, _
    LinearGradientMode.BackwardDiagonal)
    e.Graphics.FillRectangle(degData, recData)
    Dim recBoton As New Rectangle(290, 10, 90, 130)
    Dim degBoton As New LinearGradientBrush _
(recBoton, Color.Aqua, Color.Green, _
    LinearGradientMode.BackwardDiagonal)
    e.Graphics.FillRectangle(degBoton, recBoton)
End Sub
```

```
Private Sub PersonalizarGrilla()
    dgvEmpleado.Columns(0).HeaderText = "ID"
    dgvEmpleado.Columns(0).Width = 40
    dgvEmpleado.Columns(1).HeaderText = "Apellido"
    dgvEmpleado.Columns(1).Width = 80
    dgvEmpleado.Columns(2).HeaderText = "Nombre"
    dgvEmpleado.Columns(2).Width = 80
    dgvEmpleado.Columns(3).HeaderText = "Fecha Nac"
    dgvEmpleado.Columns(3).Width = 100
End Sub
```

```
Private Sub ListarEmpleados(ByVal con As SqlConnection)
    Dim cmd As New SqlCommand _
("Select EmployeeID,LastName,FirstName,BirthDate From Employees
    Order By 1", con)
    Dim drd As SqlDataReader = _
cmd.ExecuteReader(CommandBehavior.SingleResult)
    bs.DataSource = drd
    dgvEmpleado.DataSource = bs
    drd.Close()
End Sub
```

```
Private Sub CargarDatos(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    Using con As New SqlConnection(strConexion)
        Try
```

```
        con.Open()
        ListarEmpleados(con)
    Catch ex As SqlException
        MessageBox.Show("No se pudo conectar a la BD")
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End Using
PersonalizarGrilla()
End Sub

Private Sub MostrarDatos(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles dgvEmpleado.SelectionChanged
    With dgvEmpleado.CurrentRow
        txtCodigo.Text = .Cells(0).Value
        txtApellido.Text = .Cells(1).Value
        txtNombre.Text = .Cells(2).Value
        dtpFechaNac.Value = .Cells(3).Value
    End With
End Sub

Private Sub Nuevo(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnNuevo.Click
    txtCodigo.Clear()
    txtApellido.Clear()
    txtNombre.Clear()
    dtpFechaNac.Value = Now
End Sub

Private Sub AdicionarEmpleado(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnAdicionar.Click
    If txtApellido.Text.Trim <> "" Then
        If txtNombre.Text.Trim <> "" Then
            Dim Diferencia As TimeSpan = _
                Now.Date.Subtract(dtpFechaNac.Value.Date)
            If Diferencia.Days > 0 Then
                Using con As New SqlConnection(strConexion)
                    Try
                        con.Open()
                        Dim cmd As New SqlCommand _
```

```
("Insert Into Employees(LastName,FirstName,BirthDate)
  Values(@Apellido,@Nombre,@FechaNac)", con)
cmd.Parameters.Add("@Apellido", SqlDbType.VarChar,
20).Value = txtApellido.Text
cmd.Parameters.Add("@Nombre", SqlDbType.VarChar,
10).Value = txtNombre.Text
cmd.Parameters.Add("@FechaNac",
SqlDbType.DateTime).Value = dtpFechaNac.Value
N = cmd.ExecuteNonQuery
ListarEmpleados(con)
bs.Position = bs.Count - 1
If N > 0 Then
  MessageBox.Show("Empleado Adicionado", "Aviso", _
  MessageBoxButtons.OK, MessageBoxIcon.Information)
Else
  MessageBox.Show("Empleado No se pudo Adicionar",
  "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
End If
Catch ex As SqlException
  MessageBox.Show("No se pudo realizar la operación")
Catch ex As Exception
  MessageBox.Show(ex.Message)
End Try
End Using
Else
  MessageBox.Show("La Fecha de Nacimiento debe ser menor a hoy")
  dtpFechaNac.Focus()
End If
Else
  MessageBox.Show("Falta ingresar el Nombre", "Aviso", _
  MessageBoxButtons.OK, MessageBoxIcon.Information)
  txtNombre.Focus()
End If
Else
  MessageBox.Show("Falta ingresar el Apellido", "Aviso", _
  MessageBoxButtons.OK, MessageBoxIcon.Information)
  txtApellido.Focus()
End If
End Sub
```

```
Private Sub ActualizarEmpleado(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnActualizar.Click
    If txtApellido.Text.Trim <> "" Then
        If txtNombre.Text.Trim <> "" Then
            Dim Diferencia As TimeSpan = _
                Now.Date.Subtract(dtpFechaNac.Value.Date)
            If Diferencia.Days > 0 Then
                Using con As New SqlConnection(strConexion)
                    Try
                        con.Open()
                        Dim cmd As New SqlCommand("Update Employees Set
                        LastName=@Apellido,FirstName=@Nombre,BirthDate=
                        @FechaNac Where EmployeeID=@Codigo", con)
                        cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value =
                        Integer.Parse(txtCodigo.Text)
                        cmd.Parameters.Add("@Apellido", SqlDbType.VarChar,
                        20).Value = txtApellido.Text
                        cmd.Parameters.Add("@Nombre", SqlDbType.VarChar,
                        10).Value = txtNombre.Text
                        cmd.Parameters.Add("@FechaNac",
                        SqlDbType.DateTime).Value = dtpFechaNac.Value
                        N = cmd.ExecuteNonQuery
                        ListarEmpleados(con)
                        If N > 0 Then
                            MessageBox.Show("Empleado Actualizado", "Aviso", _
                                MessageBoxButtons.OK, MessageBoxIcon.Information)
                        Else
                            MessageBox.Show("Empleado No se pudo Actualizar
                                porque fue eliminado", _
                                "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                        End If
                    Catch ex As SqlException
                        MessageBox.Show("No se pudo realizar la operación")
                    Catch ex As Exception
                        MessageBox.Show(ex.Message)
                    End Try
                End Using
            Else
                MessageBox.Show("La Fecha de Nacimiento debe ser menor a hoy")
                dtpFechaNac.Focus()
            End If
        End If
    End If
End Sub
```



```
End If
Else
    MessageBox.Show("Falta ingresar el Nombre", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    txtNombre.Focus()
End If
Else
    MessageBox.Show("Falta ingresar el Apellido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    txtApellido.Focus()
End If
End Sub

Private Sub EliminarEmpleado(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnEliminar.Click
    If txtCodigo.Text <> "" Then
        If MessageBox.Show("Estas seguro de eliminarlo", "Aviso", _
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) = _
            Windows.Forms.DialogResult.Yes Then
            Using con As New SqlConnection(strConexion)
                Try
                    con.Open()
                    Dim cmd As New SqlCommand("Delete From Employees
                    Where EmployeeID=@Codigo", con)
                    cmd.Parameters.Add("@Codigo", SqlDbType.Int).Value = _
                        Integer.Parse(txtCodigo.Text)
                    N = cmd.ExecuteNonQuery
                    ListarEmpleados(con)
                    If N > 0 Then
                        MessageBox.Show("Empleado Eliminado", "Aviso", _
                            MessageBoxButtons.OK, MessageBoxIcon.Information)
                    Else
                        MessageBox.Show("Empleado No se pudo Eliminar", _
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                    End If
                Catch ex As SqlException
                    MessageBox.Show("No se pudo realizar la operación")
                Catch ex As Exception
                    MessageBox.Show(ex.Message)
                End Try
            End Using
        End If
    End If
End Sub
```

```
End Using
End If
Else
    MessageBox.Show("Falta seleccionar el Empleado", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
End If
End Sub

End Class
```

Nota: En el código anterior en vez de usar 2 contenedores para agrupar los datos del empleado y los botones como por ejemplo los controles GroupBox o panel se ha dibujado ambos rectángulos con un degradado.

Importante: En cuanto a rendimiento es mejor dibujar un rectángulo que usar un contenedor.

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

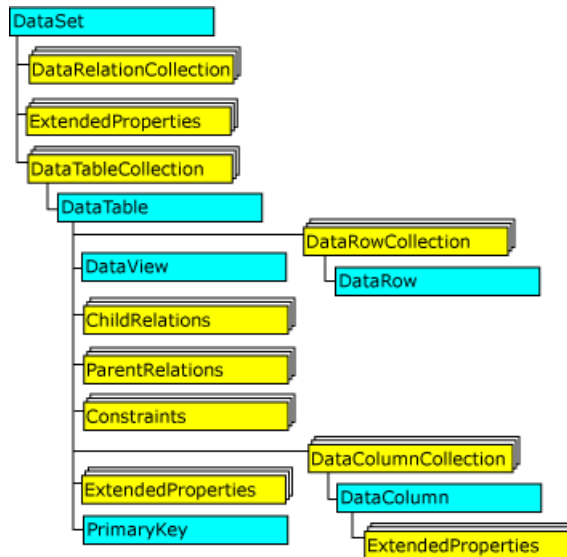
Gráfico 3.19: Ejecución del formulario Mantenimiento de Empleados Conectado

	ID	Apellido	Nombre	Fecha Nac
▶	1	Davolio	Nancy2	08/12/1948
	2	Fuller	Andrew	19/02/1952
	3	Leverling	Janet1	30/08/1963
	4	Peacock	Margaret	19/09/1937
	5	Buchanan	Steven	04/03/1955
	6	Suyama	Michael	02/07/1963
	7	King	Robert	29/05/1960
	8	Callahan	Laura	09/01/1958
	9	Dodsworth	Alice	27/01/1966

2. Trabajando en Forma Desconectada con DataSet

El DataSet es una representación en memoria de los datos el cual consta de tablas y relaciones, tal como se muestra en la siguiente figura.

Gráfico 3.20: Modelo de objetos del DataSet



En esta parte veremos como trabajar en forma desconectada usando el DataSet. Primero debemos entender porqué y cuando trabajar en forma desconectada y luego como crear un DataSet, con tablas, relaciones y vistas para manejar los datos.

No siempre debemos ejecutar todas las consultas en línea contra la base de datos ya que si existen muchos usuarios y si la red presenta problemas de ancho de banda la consulta demoraría demasiado sobre todo para Internet.

Si la data que deseamos consultar no es demasiada y no cambia constantemente, es preferible trabajar en forma desconectada, es decir, realizar las consultas localmente, puede ser en un DataSet o en una Lista de Objetos.

Por ejemplo para consultar los datos de los empleados, clientes o proveedores no es común que estos datos cambien a cada momento,

también si consultamos el ubigeo o departamento, provincia y distrito, no es necesario al seleccionar un departamento ir a la base de datos para traer las provincias ni tampoco al elegir la provincia ir a la base de datos para traer los distritos, sino que podemos cargar las 3 tablas en memoria y hacer las consultas locales ya que dicha data cambia después de tiempo.

Para más información sobre el DataSet ver la referencia 21 al final del libro.

A continuación veremos como trabajar en forma desconectada usando DataSet y en el siguiente tema lo haremos con listas de objetos.

2.1. Trabajando con Tablas y Relaciones

Un DataSet tiene como elementos tablas (DataTables) y relaciones (Data Relations). Las tablas son los repositorios de datos y las relaciones los vínculos entre ellas. Una tabla de un DataSet se puede crear en forma independiente o se puede crear a partir de una base de datos.

El DataSet puede tener varias tablas de diferentes bases de datos e inclusive pueden estar relacionadas entre ellas para favorecer la integridad de los datos, además una tabla puede tener varias relaciones.

A continuación presentamos un ejemplo que lista las ordenes de un cliente en forma desconectada usando tablas relacionadas y creando vistas para cada fila de la tabla padre.

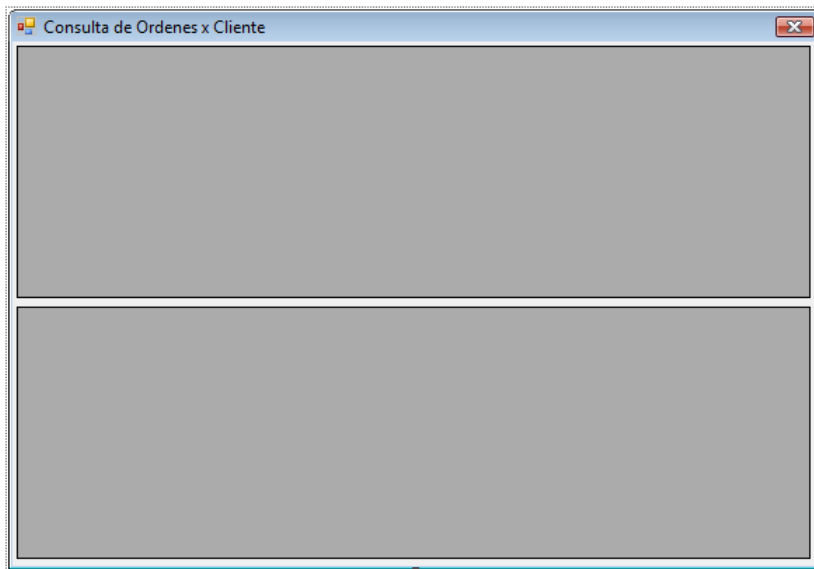
Demo 36

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo36.
- Cambiar de nombre al formulario de form1.vb a frmConsultaOrdenes Cliente.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmConsultaOrdenesCliente FixedSingle False False 580,400 CenterScreen Consulta de Ordenes x Cliente
DataGridView1	Name AutoSizeColumnsM Location MultiSelect ReadOnly SelectionMode Size	dgvCliente AllCells 3, 3 False True FullRowSelect 566, 180
DataGridView1	Name AutoSizeColumnsM Location MultiSelect ReadOnly SelectionMode Size	dgvOrden AllCells 3, 189 False True FullRowSelect 566, 180

- El diseño del formulario debe quedar similar al gráfico 3.21:

Gráfico 3.21: Diseño del formulario Consulta de Ordenes x Cliente



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data 'DataSet, DataRelation
```

```
Imports System.Data.SqlClient 'SqlConnection, SqlDataAdapter
```

```
Public Class frmConsultaOrdenesCliente
```

```
    'Definir el Repositorio para las 2 tablas
```

```
    Private dst As New DataSet
```

```
    Private Sub ListarOrdenesCliente(ByVal sender As System.Object, _
```

```
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Using con As New SqlConnection("uid=sa;pwd=123456;  
            data source=Lduenas\MCTS;initial catalog=Northwind")
```

```
            'Abrir la conexión a SQL Server
```

```
            con.Open()
```

```
            'Definir la intruccion SQL a ejecutar
```

```
            Dim dap As New SqlDataAdapter("Select CustomerId,CompanyName,  
                Address From Customers;Select OrderID,CustomerID,OrderDate From  
                Orders", con)
```

```
            'Ejecutar la instruccion SQL y crear 2 tablas
```

```
            dap.Fill(dst)
```

```
            'Definir una relacion entre las 2 tablas
```

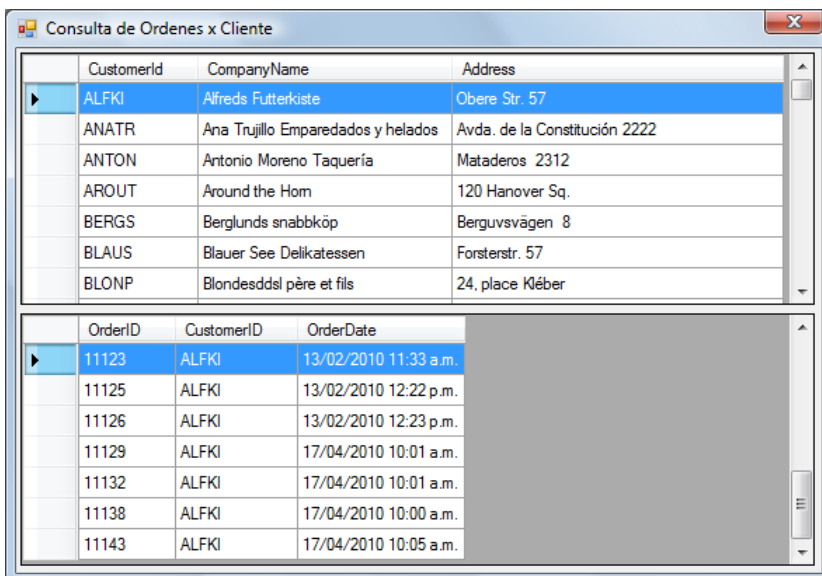
```
            Dim drn As New DataRelation("Relacion", _
```

```
dst.Tables(0).Columns(0), dst.Tables(1).Columns(1))
'Agregar la relacion al DataSet
dst.Relations.Add(drn)
End Using
'Enlazar la grilla a la tabla de Clientes
dgvCliente.DataSource = dst.Tables(0)
End Sub

Private Sub FiltrarOrdenesxCiente(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles dgvCliente.SelectionChanged
'Filtrar Ordenes creando una vista de las filas relacionadas
dgvOrden.DataSource = dst.Tables(0).DefaultView _
(dgvCliente.CurrentCell.RowIndex).CreateChildView("Relacion")
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.22: Ejecución del formulario Consulta de Ordenes x Cliente



	CustomerId	CompanyName	Address
▶	ALFKI	Alfreds Futterkiste	Obere Str. 57
	ANATR	Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222
	ANTON	Antonio Moreno Taquería	Mataderos 2312
	AROUT	Around the Horn	120 Hanover Sq.
	BERGS	Berglunds snabbköp	Berguvsvägen 8
	BLAUS	Blauer See Delikatessen	Forsterstr. 57
	BLONP	Blondesddsl père et fils	24, place Kléber

	OrderID	CustomerID	OrderDate
▶	11123	ALFKI	13/02/2010 11:33 a.m.
	11125	ALFKI	13/02/2010 12:22 p.m.
	11126	ALFKI	13/02/2010 12:23 p.m.
	11129	ALFKI	17/04/2010 10:01 a.m.
	11132	ALFKI	17/04/2010 10:01 a.m.
	11138	ALFKI	17/04/2010 10:00 a.m.
	11143	ALFKI	17/04/2010 10:05 a.m.

2.2. Trabajando con Vistas

Las vistas son cursores de solo lectura obtenidos a partir de una tabla y se usan para filtrar, ordenar y buscar registros, entre otros usos. La clase .NET que representa a una vista es el DataView y solo se puede crear una vista mediante la propiedad DefaultView de una tabla.

Filtrar Registros

Esta operación consiste en seleccionar solo los registros que cumplan una condición o filtro de selección y se realiza mediante la propiedad RowFilter de la clase DataView.

Una vez aplicado el filtro, si deseamos obtener nuevamente toda la data, es decir quitar el filtro aplicado, solo basta configurar la propiedad RowFilter en una cadena vacía.

A continuación presentamos un ejemplo que permite filtrar los productos por código o por nombre del producto, si no se escribe nada se muestran todos los productos.

Demo 37

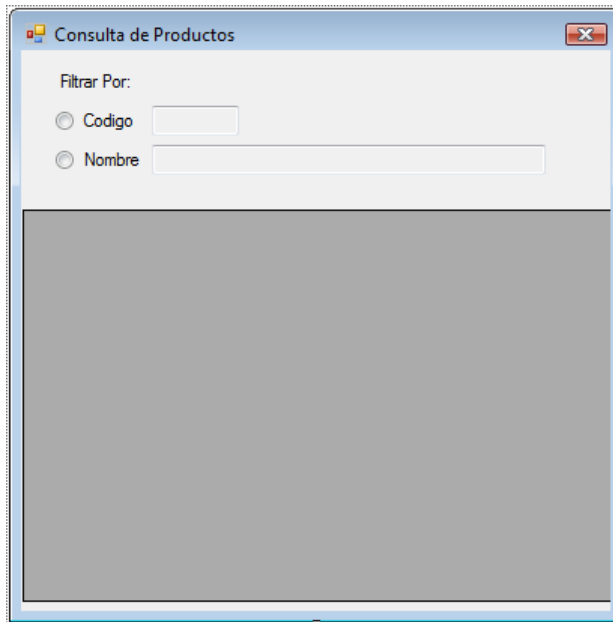
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo37.
- Cambiar de nombre al formulario de form1.vb a frmConsultaProducto.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmConsultaProducto
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False

	Size StartPosition Text	400,400 CenterScreen Consulta de Productos
Label1	Name AutoSize Location Text	lblTitulo True 23, 14 Filtrar Por:
RadioButton1	Name AutoSize Location Text	rbCodigo True 23, 38 Codigo
TextBox1	Name Location ReadOnly Size	txtCodigo 87, 37 True 58, 20
RadioButton2	Name AutoSize Location Text	rbNombre True 23, 64 Nombre
TextBox2	Name Location ReadOnly Size	txtNombre 87, 63 True 262, 20
DataGridView1	Name AutoSizeColumnsM Location ReadOnly SelectionMode Size	dgvProducto AllCells 1, 106 True FullRowSelect 392, 261

- El diseño del formulario debe quedar similar al gráfico 3.23:

Gráfico 3.23: Diseño del formulario Consulta de Productos



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data 'DataSet, DataView
Imports System.Data.SqlClient 'SqlConnection, SqlDataAdapter
Imports System.Drawing.Drawing2D 'LinearGradientBrush

Public Class frmConsultaProducto
    Private dv As DataView 'Vista de Productos

    Private Sub DibujarRecuadro(ByVal sender As Object, _
        ByVal e As System.Windows.Forms.PaintEventArgs) Handles Me.Paint
        Dim recData As New Rectangle(10, 10, 370, 90)
        Dim degData As New LinearGradientBrush _
            (recData, Color.Aqua, Color.Blue, _
            LinearGradientMode.BackwardDiagonal)
        e.Graphics.FillRectangle(degData, recData)
    End Sub

    Private Sub ListarProductos(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim dst As New DataSet
```

```
Using con As New SqlConnection("uid=sa;pwd=123456;
    data source=Lduenas\MCTS;initial catalog=Northwind")
con.Open()
Dim dap As New SqlDataAdapter _
    ("Select ProductID,ProductName,UnitPrice From Products", con)
dap.Fill(dst, "Productos")
End Using
dv = dst.Tables(0).DefaultView
dgvProducto.DataSource = dv
End Sub
```

```
Private Sub HabilitarCodigo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles rbCodigo.CheckedChanged
    If rbCodigo.Checked Then
        txtCodigo.ReadOnly = False
        txtNombre.ReadOnly = True
        txtCodigo.Focus()
    End If
End Sub
```

```
Private Sub HabilitarNombre(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles rbNombre.CheckedChanged
    If rbNombre.Checked Then
        txtCodigo.ReadOnly = True
        txtNombre.ReadOnly = False
        txtNombre.Focus()
    End If
End Sub
```

```
Private Sub FiltrarPorCodigo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles txtCodigo.TextChanged
    If txtCodigo.Text.Trim <> "" Then
        dv.RowFilter = String.Format _
            ("ProductID={0}", txtCodigo.Text)
    Else
        dv.RowFilter = ""
    End If
End Sub
```

```
Private Sub FiltrarPorNombre(ByVal sender As System.Object, _
```

```
ByVal e As System.EventArgs) Handles txtNombre.TextChanged
    If txtNombre.Text.Trim <> "" Then
        dv.RowFilter = String.Format _
            ("ProductName Like '%{0}%', txtNombre.Text)
    Else
        dv.RowFilter = ""
    End If
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.24: Ejecución del formulario Consulta de Productos

	ProductID	ProductName	UnitPrice
▶	6	Grandma's Boysenberry Spread	25.0000
	12	Queso Manchego La Pastora	38.0000
	20	Sir Rodney's Marmalade	81.0000
	30	Nord-Ost Matjeshering	25.8900
	32	Mascarpone Fabioli	32.0000
	44	Gula Malacca	19.4500
	49	Maxilaku	20.0000
	51	Manjimup Dried Apples	53.0000
	124	Marsal	5.0000
	163	Rezen Mas	2.0000

Ordenar y Buscar Registros

Si la data que queremos consultar es demasiada mejor seria filtrar los registros, pero si la data no es mucha, entonces, podemos ubicarnos sobre el registro buscado.

Para buscar registros se usa el método Find de la clase DataView, pero necesita que la data esté ordenada por una o más columnas y esto se realiza con la propiedad Sort del DataView.

Advertencia: Si se llama al método Find del DataView sin antes haber ordenado la data usando la propiedad Sort del DataView se generará una excepción o error en tiempo de ejecución.

Demo 38

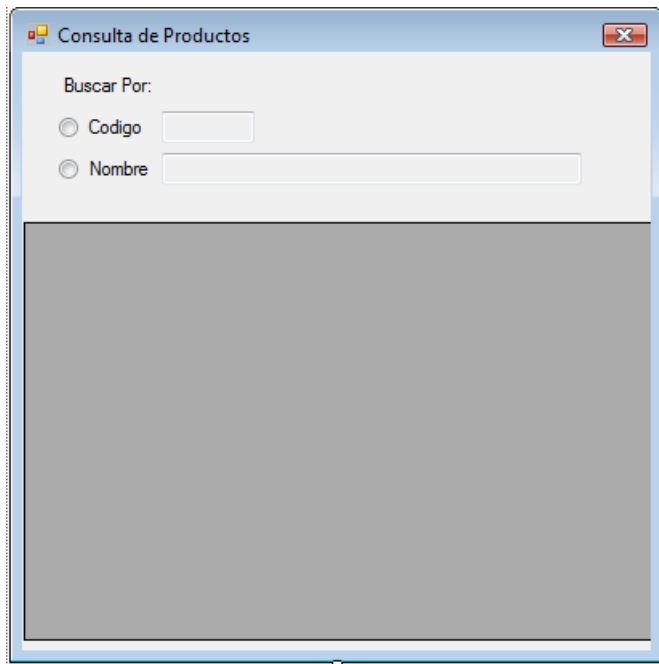
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo38.
- Cambiar de nombre al formulario de form1.vb a frmConsulta Producto.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmConsultaProducto FixedSingle False False 400,400 CenterScreen Consulta de Productos
Label1	Name AutoSize Location Text	lblTitulo True 23, 14 Buscar Por:
RadioButton1	Name AutoSize	rbCodigo True

	Location Text	23, 38 Codigo
TextBox1	Name Location ReadOnly Size	txtCodigo 87, 37 True 58, 20
RadioButton2	Name AutoSize Location Text	rbNombre True 23, 64 Nombre
TextBox2	Name Location ReadOnly Size	txtNombre 87, 63 True 262, 20
DataGridView1	Name AutoSizeColumnsM Location ReadOnly SelectionMode Size	dgvProducto AllCells 1, 106 True FullRowSelect 392, 261

- El diseño del formulario debe quedar similar al gráfico 3.25:

Gráfico 3.25: Diseño del formulario Consulta de Productos



- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data 'DataSet, DataView
```

```
Imports System.Data.SqlClient 'SqlConnection, SqlDataAdapter
```

```
Imports System.Drawing.Drawing2D 'LinearGradientBrush
```

```
Public Class frmConsultaProducto
```

```
    Private dv As DataView 'Vista de Productos
```

```
    Private bs As New BindingSource
```

```
    Private Sub DibujarRecuadro(ByVal sender As Object, _
```

```
        ByVal e As System.Windows.Forms.PaintEventArgs) Handles Me.Paint
```

```
        Dim recData As New Rectangle(10, 10, 370, 90)
```

```
        Dim degData As New LinearGradientBrush _
```

```
            (recData, Color.Aqua, Color.Blue, _
```

```
            LinearGradientMode.BackwardDiagonal)
```

```
        e.Graphics.FillRectangle(degData, recData)
```

```
    End Sub
```

```
    Private Sub QuitarOrdenDataGridView()
```

```
Dim I As Integer
For I = 0 To dgvProducto.ColumnCount - 1
    dgvProducto.Columns(I).SortMode = _
        DataGridViewColumnSortMode.NotSortable
Next
End Sub

Private Sub ListarProductos(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    Dim dst As New DataSet
    Using con As New SqlConnection("uid=sa;pwd=123456;
        data source=Lduenas\MCTS;initial catalog=Northwind")
        con.Open()
        Dim dap As New SqlDataAdapter _
            ("Select ProductID,ProductName,UnitPrice From Products", con)
        dap.Fill(dst, "Productos")
    End Using
    dv = dst.Tables(0).DefaultView
    bs.DataSource = dv
    dgvProducto.DataSource = bs
    QuitarOrdenDataGridView()
End Sub

Private Sub HabilitarCodigo(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles rbCodigo.CheckedChanged
    If rbCodigo.Checked Then
        dv.Sort = "ProductID"
        txtCodigo.ReadOnly = False
        txtNombre.ReadOnly = True
        txtCodigo.Focus()
    End If
End Sub

Private Sub HabilitarNombre(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles rbNombre.CheckedChanged
    If rbNombre.Checked Then
        dv.Sort = "ProductName"
        txtCodigo.ReadOnly = True
        txtNombre.ReadOnly = False
        txtNombre.Focus()
    End If
End Sub
```



```
End If  
End Sub
```

```
Private Sub BuscarPorCodigo(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles txtCodigo.TextChanged  
    If txtCodigo.Text.Trim <> "" Then  
        Dim pos As Integer = dv.Find(txtCodigo.Text)  
        If pos > -1 Then  
            bs.Position = pos  
        End If  
    End If  
End Sub
```

```
Private Sub BuscarPorNombre(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles txtNombre.TextChanged  
    If txtNombre.Text.Trim <> "" Then  
        Dim Criterio As String = String.Format _  
("ProductName Like '{0}%', txtNombre.Text)  
        Dim dr() As DataRow = dv.Table.Select(Criterio)  
        If dr.Length > 0 Then  
            Dim pos As Integer = dv.Find(dr(0)(1))  
            If pos > -1 Then bs.Position = pos  
        End If  
    End If  
End Sub  
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.26: Ejecución del formulario Consulta de Productos

Consulta de Productos

Buscar Por:

☒ Codigo 5

☐ Nombre

ProductID	ProductName	UnitPrice
1	Chai	10.0000
2	Chang	3.0000
3	Chicha	10.0000
4	Chef Anton's Cajun Seasoning	22.0000
5	Chef Anton's Gumbo Mix	21.3500
6	Grandma's Boysenberry Spread	25.0000
7	Uncle Bob's Organic Dried Pears	30.0000
8	Northwoods Cranberry Sauce	40.0000
9	Mishi Kobe Niku	97.0000
10	Ikura	31.0000
11	Queso Cabrales	21.0000

2.3. Mantenimiento Desconectado con DataSet

En esta parte veremos como realizar modificaciones a las tablas del DataSet: adicionar, actualizar y eliminar registros; y luego como enviar dichos cambios realizados de regreso a la base de datos.

Para adicionar un registro primero se toma la estructura de un registro de la tabla y se crea una fila usando el método `NewRow` del `DataTable`, luego se llena de datos la fila y se agrega este objeto a la colección de filas (`Rows`) del `DataTable`.

Para actualizar un registro se obtiene el registro actual y se modifican los datos directamente.

Para eliminar un registro existen 2 formas: usar el método `Remove` del `DataTable` o el método `Delete` del `DataRow`, la diferencia es que el `Remove` marca el registro pero no lo borra de la tabla para después enviar dicha eliminación a la base de datos; en cambio, el `Delete` lo elimina de la tabla y ya no se puede enviar la eliminación a la base de datos.

Para obtener los cambios que se vienen realizando en un DataSet o un DataTable se usa el método `GetChanges` que por defecto obtiene los registros modificados y adicionados a la(s) tabla(s). Este tiene un parámetro que indica que tipo de cambios puedes visualizar.

Finalmente, para enviar los cambios realizados: adiciones, actualizaciones y eliminaciones a la(s) tabla(s) se usa el método `Update` del `DataAdapter`, pero debe haberse configurado las propiedades: `InsertCommand`, `UpdateCommand` y `DeleteCommand` del `DataAdapter` con la instrucción SQL a ejecutar y con los parámetros respectivos para cada instrucción.

Advertencia: Si se llama al método `Update` del `DataAdapter` sin existir los `Insert`, `Update` y `Select` respectivos se generará una excepción o error en tiempo de ejecución.

Nota: Existe una forma de crear automáticamente los `InsertCommand`, `UpdateCommand` y `DeleteCommand` del `DataAdapter` basados en su `SelectCommmand`, usando para ello la clase `CommandBuilder` y pasando como parámetro en el constructor el `DataAdapter`.

A continuación veremos un ejemplo donde apliquemos todo lo comentado para realizar un mantenimiento desconectado sobre la tabla empleados y donde podamos enviar los cambios por lotes hacia la base de datos.

Demo 39

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo39.
- Cambiar de nombre al formulario de form1.vb a frmMantenimiento Empleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmMantenimientoEmpleado FixedSingle False False 400,430 CenterScreen Mantenimiento de Empleados Desconectado
Label1	Name AutoSize Location Text	lblCodigo True 26, 33 Código:
TextBox1	Name Location ReadOnly Size	txtCodigo 100,26 True 47,20
Label2	Name AutoSize Location Text	lblApellido True 26, 59 Apellido:
TextBox2	Name Location Size	txtApellido 100,52 178,20
Label3	Name	lblNombre

	AutoSize Location Text	True 26, 85 Nombre:
TextBox3	Name Location Size	txtNombre 100,78 178,20
Label4	Name AutoSize Location Text	lblFechaNac True 26, 111 Fecha Nac:
DateTimePicker1	Name Format Location Size	dtpFechaNac Short 100,105 88,20
Button1	Name Cursor Location Size Text	btnNuevo Hand 297, 22 75, 23 Nuevo
Button2	Name Cursor Location Size Text	btnAdicionar Hand 297, 22 75, 50 Adicionar
Button3	Name Cursor Location Size Text	btnActualizar Hand 297, 78 75, 23 Actualizar
Button4	Name Cursor Location Size Text	btnEliminar Hand 297, 106 75, 23 Eliminar
DataGridView1	Name Location MultiSelect ReadOnly SelectionMode Size	dgvEmpleado 13, 151 False True FullRowSelect 367, 210
Button5	Name Cursor	btnVerCambios Hand

	Location Size Text	12, 369 100, 23 Ver Cambios
Button6	Name Cursor Location Size Text	btnGrabarCambios Hand 280, 369 100, 23 Grabar Cambios

- El diseño del formulario debe quedar similar al gráfico 3.27:

Gráfico 3.27: Diseño del formulario Mantenimiento de Empleados Desconectado

- Ingresar al editor de código y escribir el siguiente código.

```
Imports System.Data 'DataSet, DataTable
Imports System.Data.SqlClient 'SqlConnection, SqlDataAdapter,
SqlCommandBuilder
Imports System.Drawing.Drawing2D 'LinearGradientBrush
```

```
Public Class frmMantenimientoEmpleado
    Private strConexion As String = _
        "uid=sa;pwd=123456;data source=Lduenas\MCTS;initial
        catalog=Northwind"
    Private dtb As New DataTable
    Private con As SqlConnection
    Private dap As New SqlDataAdapter _
        ("Select EmployeeID,LastName,FirstName,BirthDate From Employees", con)
    Private bs As New BindingSource

    Private Sub DibujarRecuadro(ByVal sender As Object, _
        ByVal e As System.Windows.Forms.PaintEventArgs) Handles Me.Paint
        Dim recData As New Rectangle(15, 10, 270, 130)
        Dim degData As New LinearGradientBrush _
            (recData, Color.Aqua, Color.Blue, _
            LinearGradientMode.BackwardDiagonal)
        e.Graphics.FillRectangle(degData, recData)
        Dim recBoton As New Rectangle(290, 10, 90, 130)
        Dim degBoton As New LinearGradientBrush _
            (recBoton, Color.Aqua, Color.Green, _
            LinearGradientMode.BackwardDiagonal)
        e.Graphics.FillRectangle(degBoton, recBoton)
    End Sub

    Private Sub QuitarOrdenDataGridView()
        Dim I As Integer
        For I = 0 To dgvEmpleado.ColumnCount - 1
            dgvEmpleado.Columns(I).SortMode = _
                DataGridViewColumnSortMode.NotSortable
        Next
    End Sub

    Private Sub PersonalizarGrilla()
        dgvEmpleado.Columns(0).HeaderText = "ID"
        dgvEmpleado.Columns(0).Width = 40
        dgvEmpleado.Columns(1).HeaderText = "Apellido"
        dgvEmpleado.Columns(1).Width = 80
        dgvEmpleado.Columns(2).HeaderText = "Nombre"
        dgvEmpleado.Columns(2).Width = 80
        dgvEmpleado.Columns(3).HeaderText = "Fecha Nac"
```

```
dgvEmpleado.Columns(3).Width = 100
QuitarOrdenDataGridView()
End Sub
```

```
Private Sub ListarEmpleados()
    If dtb.Rows.Count > 0 Then dtb.Clear()
    dap.SelectCommand.Connection = con
    dap.Fill(dtb)
    bs.DataSource = dtb
    dgvEmpleado.DataSource = bs
    PersonalizarGrilla()
End Sub
```

```
Private Sub CargarDatos(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    con = New SqlConnection(strConexion)
    con.Open()
    ListarEmpleados()
    Dim sbd As New SqlCommandBuilder(dap)
    con.Close()
End Sub
```

```
Private Sub MostrarDatos(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles dgvEmpleado.SelectionChanged
    With dgvEmpleado.CurrentRow
        txtCodigo.Text = .Cells(0).Value
        txtApellido.Text = .Cells(1).Value
        txtNombre.Text = .Cells(2).Value
        dtpFechaNac.Value = .Cells(3).Value
    End With
End Sub
```

```
Private Sub Nuevo(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnNuevo.Click
    txtCodigo.Text = dtb.Rows(dtb.Rows.Count - 1)(0) + 1
    txtApellido.Clear()
    txtNombre.Clear()
    dtpFechaNac.Value = Now
    txtApellido.Focus()
End Sub
```



```
Private Sub AdicionarEmpleado(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAdicionar.Click
    If txtApellido.Text.Trim <> "" Then
        If txtNombre.Text.Trim <> "" Then
            Dim Diferencia As TimeSpan = _
                Now.Date.Subtract(dtpFechaNac.Value.Date)
            If Diferencia.Days > 0 Then
                Dim Fila As DataRow = dtb.NewRow
                Fila(0) = txtCodigo.Text
                Fila(1) = txtApellido.Text
                Fila(2) = txtNombre.Text
                Fila(3) = dtpFechaNac.Value
                dtb.Rows.Add(Fila)
                bs.Position = bs.Count - 1
            Else
                MessageBox.Show("La Fecha de Nacimiento debe ser menor a
                hoy")
                dtpFechaNac.Focus()
            End If
        Else
            MessageBox.Show("Falta ingresar el Nombre", "Aviso", _
                MessageBoxButtons.OK, MessageBoxIcon.Information)
            txtNombre.Focus()
        End If
    Else
        MessageBox.Show("Falta ingresar el Apellido", "Aviso", _
            MessageBoxButtons.OK, MessageBoxIcon.Information)
        txtApellido.Focus()
    End If
End Sub
```

```
Private Sub ActualizarEmpleado(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnActualizar.Click
    If txtApellido.Text.Trim <> "" Then
        If txtNombre.Text.Trim <> "" Then
            Dim Diferencia As TimeSpan = _
                Now.Date.Subtract(dtpFechaNac.Value.Date)
            If Diferencia.Days > 0 Then
                Dim N As Integer = _
```

```
        dgvEmpleado.CurrentRow.RowIndex
        Dim Fila As DataRow = dtb.Rows(N)
        Fila.BeginEdit()
        Fila(0) = txtCodigo.Text
        Fila(1) = txtApellido.Text
        Fila(2) = txtNombre.Text
        Fila(3) = dtpFechaNac.Value
        Fila.EndEdit()
    Else
        MessageBox.Show("La Fecha de Nacimiento debe ser menor a
            hoy")
        dtpFechaNac.Focus()
    End If
Else
    MessageBox.Show("Falta ingresar el Nombre", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    txtNombre.Focus()
End If
Else
    MessageBox.Show("Falta ingresar el Apellido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    txtApellido.Focus()
End If
End Sub

Private Sub EliminarEmpleado(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnEliminar.Click
    If txtCodigo.Text <> "" Then
        If MessageBox.Show("Estas seguro de eliminarlo", "Aviso", _
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) = _
            Windows.Forms.DialogResult.Yes Then
            Dim N As Integer = dgvEmpleado.CurrentRow.RowIndex
            dtb.Rows(N).Delete()
        End If
    Else
        MessageBox.Show("Falta seleccionar el Empleado", "Aviso", _
            MessageBoxButtons.OK, MessageBoxIcon.Information)
    End If
End Sub
```

```
Private Sub CerrarGrilla(ByVal sender As Object, ByVal e As EventArgs)
    sender.Parent.Close()
End Sub
```

```
Private Sub VerCambios(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnVerCambios.Click
    Dim frm As New Form
    frm.StartPosition = FormStartPosition.CenterScreen
    frm.FormBorderStyle = Windows.Forms.FormBorderStyle.None
    Dim dgv As New DataGridView
    dgv.DataSource = dtb.GetChanges
    dgv.SelectionMode = DataGridViewSelectionMode.FullRowSelect
    dgv.ReadOnly = True
    dgv.AllowUserToAddRows = False
    dgv.Dock = DockStyle.Fill
    AddHandler dgv.Click, AddressOf CerrarGrilla
    frm.Controls.Add(dgv)
    frm.Size = New Size(400, 200)
    frm.ShowDialog()
End Sub
```

```
Private Sub GrabarCambios(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnGrabarCambios.Click
    Try
        con.Open()
        dap.UpdateBatchSize = 1
        dap.Update(dtb)
        ListarEmpleados()
        MessageBox.Show("Operacion realizada con exito", _
            "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information)
    Catch ex As Exception
        MessageBox.Show("No se pudo realizar la operacion", _
            "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Error)
    Finally
        If con.State = ConnectionState.Open Then con.Close()
    End Try
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.28: Ejecución del formulario Mantenimiento de Empleados Desconectado

Mantenimiento de Empleados Desconectado

Código: 1

Apellido: Davolio

Nombre: Nancy2

Fecha Nac: 08/12/1948

Nuevo

Adicionar

Actualizar

Eliminar

	ID	Apellido	Nombre	Fecha Nac
▶	1	Davolio	Nancy2	08/12/1948
	2	Fuller	Andrew	19/02/1952
	3	Leverling	Janet1	30/08/1963
	4	Peacock	Margaret	19/09/1937
	5	Buchanan	Steven	04/03/1955
	6	Suyama	Michael	02/07/1963
	7	King	Robert	29/05/1960
	8	Callahan	Laura	09/01/1958
	9	Dodsworth	Alice	27/01/1966

Ver Cambios

Grabar Cambios

3. Trabajando en Forma Desconectada con Listas de Objetos

Otra forma de trabajar desconectados es usando listas de objetos, sobre todo si trabajamos con el Modelo de Desarrollo Distribuido (MDD) y usamos entidades del negocio, es preferible usar lista de objetos al DataSet, ya que se consume menos memoria debido a que el DataSet no solo tiene la data de las tablas sino también su metadata o definición para cada elemento o atributo.

En .NET Framework existe el espacio de nombres System.Collection. Generics que permite trabajar con listas genéricas de objetos. En esta parte aprenderemos como crear una lista de objetos, como filtrar datos de la lista, como ordenar, buscar y finalmente como realizar un mantenimiento con objetos.

3.1 Llenando una Lista de Objetos

Para trabajar con Listas de Objetos primero es necesario crear los Procedimientos Almacenados, luego las diferentes Librerías de Clases iniciando con las Entidades del Negocio, luego el Acceso a Datos y después las Reglas del Negocio, finalmente se crea la aplicación que usa las librerías de negocio: entidades y reglas.

Creando los Procedimientos Almacenados

Es preferible usar procedimientos almacenados de SQL Server en vez de enviar desde la aplicación la instrucción SQL, ya que los procedimientos almacenados se compilan en el servidor y se encuentran preparados para su ejecución, lo que lo hace mas rápido.

Para crear los procedimientos almacenados podemos usar cualquier cliente de SQL Server como el Microsoft SQL Server Management Studio si es que se ha instalado el cliente de SQL Server 2005 o 2008, también podríamos usar el Explorador de Servidores de Visual Studio 2010.

Crear los siguientes procedimientos almacenados:

1. Procedimiento almacenado que lista empleados

```
Create Procedure usp_Employees_Sel
AS
Select EmployeeID,LastName,FirstName,
IsNull(BirthDate,"") As BirthDate
From Employees
```

2. Procedimiento almacenado que lista productos

```
Create Procedure usp_Products_Sel
AS
SELECT
    ProductID,
    ProductName,
    IsNull(SupplierID,0) As SupplierID,
    IsNull(CategoryID,0) As CategoryID,
    IsNull(UnitPrice,0) As UnitPrice,
    IsNull(UnitsInStock,0) As UnitsInStock
FROM Products
ORDER BY 1
```

3. Procedimiento almacenado que lista categorías

```
Create Procedure usp_Categories_Sel
AS
Select CategoryID,CategoryName
From Categories Order By 2
```

4. Procedimiento Almacenado que lista proveedores

```
Create Procedure usp_Suppliers_Sel
AS
Select SupplierID,CompanyName
From Suppliers Order By 2
```

Nota: Para nombrar el procedimiento se ha usado el prefijo usp seguido del Nombre de la tabla y como sufijo el tipo de operación SQL: sel indica selección, ins indica inserción, upd indica actualización y del indica eliminación. Para separar el prefijo, la tabla y el sufijo se usa _.

Creando la Librería de Entidades del Negocio

Después de crear los procedimientos almacenados hay que crear las clases entidades del negocio que se obtienen de los datos que queremos manejar de cada tabla.

Demo Northwind.LibBusinessEntities

- Del menú "File", seleccionar "New Project" y luego "Class Library".
- Escribir como nombre físico al proyecto: Northwind.LibBusinessEntities.
- Cambiar de nombre a la clase de Class1.vb a beEmpleado.vb.
- Escribir el siguiente código para la clase beEmpleado:

```
Public Class beEmpleado
    Public Property Codigo As Integer
    Public Property Apellido As String
    Public Property Nombre As String
    Public Property FechaNac As DateTime
End Class
```

Nota: En la versión 2010 de Visual Basic ya se puede crear directamente las propiedades sin necesidad de definir variables locales y los métodos Get y Set para propiedades de lectura y escritura.

- Agregar una nueva clase para Productos: del menú "Project", "Add Class" y escribir como nombre: beProducto.
- Escribir el siguiente código para la clase beProducto:

```
Public Class beProducto
    Public Property Codigo As Integer
    Public Property Nombre As String
    Public Property IdProveedor As Integer
    Public Property IdCategoria As Integer
    Public Property PrecioUnitario As Decimal
    Public Property Stock As Short
End Class
```

- Agregar una nueva clase para Categorías: del menú "Project", "Add Class" y escribir como nombre: beCategoria.
- Escribir el siguiente código para la clase beCategoria:

```
Public Class beCategoria
    Public Property Codigo As Integer
    Public Property Nombre As String
End Class
```

- Agregar una nueva clase para Proveedores: del menú "Project", "Add Class" y escribir como nombre: beProveedor.
- Escribir el siguiente código para la clase beProveedor:

```
Public Class beProveedor
    Public Property Codigo As Integer
    Public Property Nombre As String
End Class
```

- Finalmente, del menú "Build" seleccionar "Build Northwind.LibBusiness Entities" para crear la dll con las entidades del negocio.

Creando la Librería de Acceso a Datos

Después de crear la librería con las clases entidades se crea la librería de acceso a datos que contiene los métodos que ejecutan los procedimientos almacenados mediante comandos de ADO .NET y llenan la lista de objetos para devolverlos a las reglas del negocio.

Para crear listas de objetos primero hay que hacer referencia a la librería de entidades creada anteriormente.

Cada método de acceso a datos tiene como parámetro la clase Connection para que abriendo una sola conexión desde la regla de negocio se pueda llamar a varios métodos de los servicios de acceso a datos.

Demo Northwind.LibDataAccess

- Del menú "File", seleccionar "New Project" y luego "Class Library".
- Escribir como nombre físico al proyecto: Northwind.LibDataAccess.
- Hacer una referencia a la librería de entidades creada: seleccionar el proyecto, clic derecho "Add References", seleccionar ficha "Browse" y buscar el archivo Northwind.LibBusinessEntities.dll creado.
- Cambiar de nombre a la clase de Class1.vb a daEmpleado.vb.
- Escribir el siguiente código para la clase daEmpleado:

```
Imports Northwind.LibBusinessEntities
Imports System.Data.SqlClient
```

```
Public Class daEmpleado
```

```
    Public Function fListar(ByVal con As SqlConnection) As List(Of beEmpleado)
        Dim lobeEmpleado As New List(Of beEmpleado)
```

```
        Dim cmd As New SqlCommand("usp_Employees_Sel", con)
        cmd.CommandType = CommandType.StoredProcedure
        cmd.CommandTimeout = 60
        Dim drd As SqlDataReader = _
            cmd.ExecuteReader(CommandBehavior.SingleResult)
        If drd IsNot Nothing Then
```

```
Dim posCodigo As Integer = drd.GetOrdinal("EmployeeID")
Dim posApellido As Integer = drd.GetOrdinal("LastName")
Dim posNombre As Integer = drd.GetOrdinal("FirstName")
Dim posFechaNac As Integer = drd.GetOrdinal("BirthDate")
Dim obeEmpleado As beEmpleado
Do While drd.Read
    obeEmpleado = New beEmpleado
    With obeEmpleado
        .Codigo = drd.GetInt32(posCodigo)
        .Apellido = drd.GetString(posApellido)
        .Nombre = drd.GetString(posNombre)
        .FechaNac = drd.GetDateTime(posFechaNac)
    End With
    lobeEmpleado.Add(obeEmpleado)
Loop
drd.Close()
End If

Return (lobeEmpleado)
End Function
End Class
```

Nota: Es importante definir el objeto obeEmpleado fuera del bucle Do While y dentro de este crear una sola instancia del objeto para ir agregándolo a la lista de objetos al final.

- Agregar una nueva clase para Productos: del menú "Project", "Add Class" y escribir como nombre: daProducto.
- Escribir el siguiente código para la clase daProducto:

```
Imports Northwind.LibBusinessEntities
Imports System.Data.SqlClient

Public Class daProducto
    Public Function fListar(ByVal con As SqlConnection) As List(Of beProducto)
        Dim lobeProducto As New List(Of beProducto)

        Dim cmd As New SqlCommand("usp_Products_Sel", con)
        cmd.CommandType = CommandType.StoredProcedure
        cmd.CommandTimeout = 60
```

```
Dim drd As SqlDataReader = _
cmd.ExecuteReader(CommandBehavior.SingleResult)
If drd IsNot Nothing Then
    Dim posCodigo As Integer = drd.GetOrdinal("ProductID")
    Dim posNombre As Integer = drd.GetOrdinal("ProductName")
    Dim posIdProveedor As Integer = drd.GetOrdinal("SupplierID")
    Dim posIdCategoria As Integer = drd.GetOrdinal("CategoryID")
    Dim posPrecioUnitario As Integer = drd.GetOrdinal("UnitPrice")
    Dim posStock As Integer = drd.GetOrdinal("UnitsInStock")
    Dim obeProducto As beProducto
    Do While drd.Read
        obeProducto = New beProducto
        With obeProducto
            .Codigo = drd.GetInt32(posCodigo)
            .Nombre = drd.GetString(posNombre)
            .IdProveedor = drd.GetInt32(posIdProveedor)
            .IdCategoria = drd.GetInt32(posIdCategoria)
            .PrecioUnitario = drd.GetDecimal(posPrecioUnitario)
            .Stock = drd.GetInt16(posStock)
        End With
        lobeProducto.Add(obeProducto)
    Loop
    drd.Close()
End If

Return (lobeProducto)
End Function
End Class
```

- Agregar una nueva clase para Categorías: del menú "Project", "Add Class" y escribir como nombre: daCategoria.
- Escribir el siguiente código para la clase daCategoria:

```
Imports Northwind.LibBusinessEntities
Imports System.Data.SqlClient

Public Class daCategoria
    Public Function fListar(ByVal con As SqlConnection) As List(Of beCategoria)
        Dim lobeCategoria As New List(Of beCategoria)
```

```
Dim cmd As New SqlCommand("usp_Categories_Sel", con)
cmd.CommandType = CommandType.StoredProcedure
cmd.CommandTimeout = 60
Dim drd As SqlDataReader = _
cmd.ExecuteReader(CommandBehavior.SingleResult)
If drd IsNot Nothing Then
    Dim posCodigo As Integer = drd.GetOrdinal("CategoryID")
    Dim posNombre As Integer = drd.GetOrdinal("CategoryName")
    Dim obeCategoria As beCategoria
    Do While drd.Read
        obeCategoria = New beCategoria
        With obeCategoria
            .Codigo = drd.GetInt32(posCodigo)
            .Nombre = drd.GetString(posNombre)
        End With
        lobeCategoria.Add(obeCategoria)
    Loop
    drd.Close()
End If

Return (lobeCategoria)
End Function
End Class
```

- Agregar una nueva clase para Proveedores: del menú "Project", "Add Class" y escribir como nombre: daProveedor.
- Escribir el siguiente código para la clase daProveedor:

```
Imports Northwind.LibBusinessEntities
Imports System.Data.SqlClient

Public Class daProveedor
    Public Function fListar(ByVal con As SqlConnection) As List(Of beProveedor)
        Dim lobeProveedor As New List(Of beProveedor)

        Dim cmd As New SqlCommand("usp_Suppliers_Sel", con)
        cmd.CommandType = CommandType.StoredProcedure
        cmd.CommandTimeout = 60
        Dim drd As SqlDataReader = _
        cmd.ExecuteReader(CommandBehavior.SingleResult)
```

```
If drd IsNot Nothing Then
    Dim posCodigo As Integer = drd.GetOrdinal("SupplierID")
    Dim posNombre As Integer = drd.GetOrdinal("CompanyName")
    Dim obeProveedor As beProveedor
    Do While drd.Read
        obeProveedor = New beProveedor
        With obeProveedor
            .Codigo = drd.GetInt32(posCodigo)
            .Nombre = drd.GetString(posNombre)
        End With
        lobeProveedor.Add(obeProveedor)
    Loop
    drd.Close()
End If

Return (lobeProveedor)
End Function
End Class
```

- Finalmente, del menú "Build" seleccionar "Build Northwind.LibData Access" para crear la dll con los servicios de acceso a datos.

Creando la Librería de Reglas del Negocio

Después de crear la librería de Acceso a Datos hay que crear la librería de Reglas del Negocio, la cual debe usar la de acceso a datos y las entidades del negocio.

Esta contiene métodos llamados reglas de negocio que definen y abren la conexión contra el origen de datos y llaman a servicios de datos pasándole como parámetro la conexión abierta, también se encargan de controlar la integridad de datos a través de las transacciones.

Hay 2 formas de trabajar con la cadena de conexión desde esta capa, una es leyendo automáticamente la cadena de conexión desde una variable en la aplicación (appSettings) y la otra es pasar en el constructor o en cada método la cadena de conexión desde la aplicación. Para los ejemplos vamos a usar la primera técnica.

Demo Northwind.LibBusinessRules

- Del menú "File", seleccionar "New Project" y luego "Class Library".
- Escribir como nombre físico al proyecto: Northwind.LibBusinessRules.
- Hacer una referencia a la librería de entidades creada: seleccionar el proyecto, clic derecho "Add References", seleccionar ficha "Browse" y buscar el archivo Northwind.LibBusinessEntities.dll creado.
- También hacer una referencia a la librería de acceso a datos: Northwind.LibDataAccess.dll.
- Cambiar de nombre a la clase de Class1.vb a brEmpleado.vb.
- Escribir el siguiente código para la clase brEmpleado:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibDataAccess
Imports System.Data.SqlClient
```

```
Public Class brEmpleado
    Private strConexion As String
```

```
Public Sub New()  
    Dim asr As New System.Configuration.AppSettingsReader  
    strConexion = asr.GetValue("conNW", Type.GetType("System.String"))  
End Sub  
  
Public Function Listar() As List(Of beEmpleado)  
    Dim lobeEmpleado As New List(Of beEmpleado)  
    Using con As New SqlConnection(strConexion)  
        Try  
            con.Open()  
            Dim odaEmpleado As New daEmpleado  
            lobeEmpleado = odaEmpleado.fListar(con)  
        Catch ex As Exception  
            'Grabar el Log de error  
            lobeEmpleado = Nothing  
        End Try  
    End Using  
    Return (lobeEmpleado)  
End Function  
End Class
```

Advertencia: Si en la aplicación cliente que llama a la librería de reglas del negocio no se crea un archivo de configuración con una clave llamada conNW en la sección del appSetting conteniendo la cadena de conexión se generará una excepción al conectarse a la base de datos, la cual es controlada por la estructura Try..Catch.

- Agregar una nueva clase para Productos: del menú "Project", "Add Class" y escribir como nombre: brProducto.
- Escribir el siguiente código para la clase brProducto:

```
Imports Northwind.LibBusinessEntities  
Imports Northwind.LibDataAccess  
Imports System.Data.SqlClient  
  
Public Class brProducto  
    Private strConexion As String  
  
    Public Sub New()  
        Dim asr As New System.Configuration.AppSettingsReader
```

```
strConexion = asr.GetValue("conNW", Type.GetType("System.String"))
End Sub
```

```
Public Function Listar() As List(Of beProducto)
    Dim lobeProducto As New List(Of beProducto)
    Using con As New SqlConnection(strConexion)
        Try
            con.Open()
            Dim odaProducto As New daProducto
            lobeProducto = odaProducto.fListar(con)
        Catch ex As Exception
            'Grabar el Log de error
            lobeProducto = Nothing
        End Try
    End Using
    Return (lobeProducto)
End Function
End Class
```

- Agregar una nueva clase para Categorías: del menú "Project", "Add Class" y escribir como nombre: brCategoría.
- Escribir el siguiente código para la clase brCategoría:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibDataAccess
Imports System.Data.SqlClient

Public Class brCategoría
    Private strConexion As String

    Public Sub New()
        Dim asr As New System.Configuration.AppSettingsReader
        strConexion = asr.GetValue("conNW", Type.GetType("System.String"))
    End Sub

    Public Function Listar() As List(Of beCategoría)
        Dim lobeCategoría As New List(Of beCategoría)
        Using con As New SqlConnection(strConexion)
            Try
                con.Open()
```



```
Dim odaCategoria As New daCategoria
lobeCategoria = odaCategoria.fListar(con)
Catch ex As Exception
    'Grabar el Log de error
    lobeCategoria = Nothing
End Try
End Using
Return (lobeCategoria)
End Function
End Class
```

- Agregar una nueva clase para Proveedores: del menú "Project", "Add Class" y escribir como nombre: brProveedor.
- Escribir el siguiente código para la clase brProveedor:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibDataAccess
Imports System.Data.SqlClient

Public Class brProveedor
    Private strConexion As String

    Public Sub New()
        Dim asr As New System.Configuration.AppSettingsReader
        strConexion = asr.GetValue("conNW", Type.GetType("System.String"))
    End Sub

    Public Function Listar() As List(Of beProveedor)
        Dim lobeProveedor As New List(Of beProveedor)
        Using con As New SqlConnection(strConexion)
            Try
                con.Open()
                Dim odaProveedor As New daProveedor
                lobeProveedor = odaProveedor.fListar(con)
            Catch ex As Exception
                'Grabar el Log de error
                lobeProveedor = Nothing
            End Try
        End Using
        Return (lobeProveedor)
    End Function
End Class
```

End Function
End Class

- Finalmente, del menú "Build" seleccionar "Build Northwind.LibBusiness Rules" para crear la dll con los servicios de reglas del negocio.

Creando la Aplicación que usa las Librerías

Finalmente, después de crear todas las librerías hay que crear la aplicación la cual solo debe usar la librería de del negocio: entidades y reglas, esta no puede usar la de acceso a datos ya que fue encapsulada desde la regla del negocio.

Además es importante crear el archivo de configuración para definir la cadena de conexión que espera las reglas del negocio.

A continuación un par de ejemplos que listan objetos desde las librerías, el primero trabaja con los Empleados y el segundo con los Productos.

Demo 40

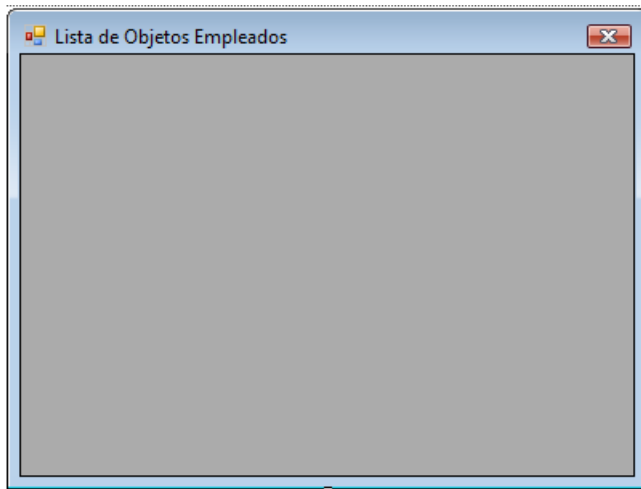
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo40.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmListaEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaEmpleado
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,300

	StartPosition Text	CenterScreen Lista de Objetos Empleados
DataGridView1	Name AutoSizeColumnsM Dock ReadOnly SelectionMode	dgvEmpleado AllCells Fill True FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 3.29:

Gráfico 3.29: Diseño del formulario Lista de Empleados



- Ingresar al editor de código y escribir el siguiente código:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Public Class frmListaEmpleado
```

```
    Private Sub ListarEmpleados(ByVal sender As System.Object, _
```

```
    ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Dim obrEmpleado As New brEmpleado
```

```
        Dim lbeEmpleado As List(Of beEmpleado) = obrEmpleado.Listar
```

```
        dgvEmpleado.DataSource = lbeEmpleado
```

```
    End Sub
```

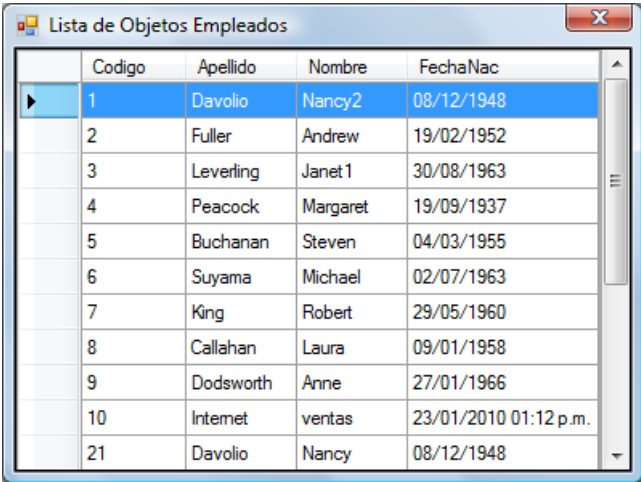
```
End Class
```

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.30: Ejecución del formulario Lista de Empleados



	Codigo	Apellido	Nombre	FechaNac
▶	1	Davolio	Nancy2	08/12/1948
	2	Fuller	Andrew	19/02/1952
	3	Leverling	Janet1	30/08/1963
	4	Peacock	Margaret	19/09/1937
	5	Buchanan	Steven	04/03/1955
	6	Suyama	Michael	02/07/1963
	7	King	Robert	29/05/1960
	8	Callahan	Laura	09/01/1958
	9	Dodsworth	Anne	27/01/1966
	10	Internet	ventas	23/01/2010 01:12 p.m.
	21	Davolio	Nancy	08/12/1948

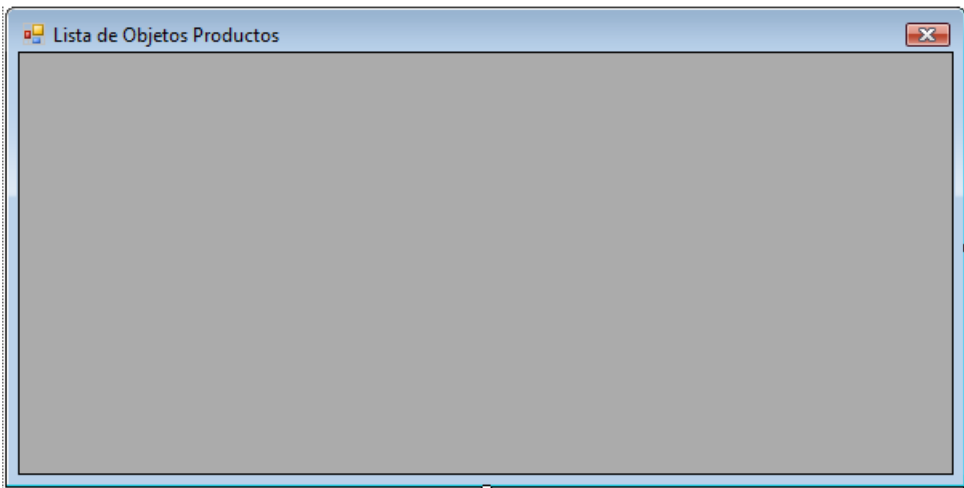
Demo 41

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo41.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmListaProducto.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaProducto
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	600,300
	StartPosition	CenterScreen
	Text	Lista de Objetos Productos
DataGridView1	Name	dgvProducto
	AutoSizeColumnsMode	AllCells
	Dock	Fill
	ReadOnly	True
	SelectionMode	FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 3.31:

Gráfico 3.31: Diseño del formulario Lista de Productos



- Ingresar al editor de código y escribir el siguiente código:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Public Class frmListaProducto
```

```
    Private Sub ListarProductos(ByVal sender As System.Object, _
```

```
    ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Dim obrProducto As New brProducto
```

```
        Dim lbeProducto As List(Of beProducto) = obrProducto.Listar
```

```
        dgvProducto.DataSource = lbeProducto
```

```
    End Sub
```

```
End Class
```

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
```

```
  <appSettings>
```

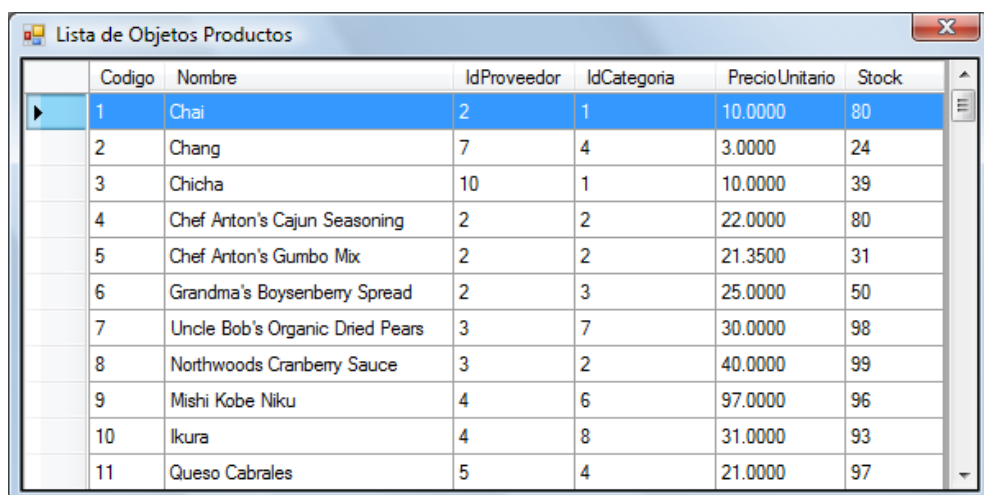
```
    <add key="conNW" value="uid=sa;pwd=123456;
```

```
    data source=Lduenas\MCTS;initial catalog=Northwind"/>
```

```
</appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.32: Ejecución del formulario Lista de Productos



	Codigo	Nombre	IdProveedor	IdCategoria	PrecioUnitario	Stock
▶	1	Chai	2	1	10.0000	80
	2	Chang	7	4	3.0000	24
	3	Chicha	10	1	10.0000	39
	4	Chef Anton's Cajun Seasoning	2	2	22.0000	80
	5	Chef Anton's Gumbo Mix	2	2	21.3500	31
	6	Grandma's Boysenberry Spread	2	3	25.0000	50
	7	Uncle Bob's Organic Dried Pears	3	7	30.0000	98
	8	Northwoods Cranberry Sauce	3	2	40.0000	99
	9	Mishi Kobe Niku	4	6	97.0000	96
	10	Ikura	4	8	31.0000	93
	11	Queso Cabrales	5	4	21.0000	97

3.2 Filtrando Datos en una Lista de Objetos

Una vez llena la lista de objetos la operación mas común es filtrar datos, es decir consultar los registros que cumplan con una cierta condición, para realizar esta operación en .NET Framework se puede hacer de diversas formas: usando la estructura For y un If, mediante predicados o usando LINQ.

En esta parte veremos como usar For If y también predicados y mas adelante veremos como usar LINQ.

Usando For - If

La forma clásica de seleccionar solo los elementos que cumplen con una condición es aplicar un bucle por ejemplo un For y luego preguntar si se cumple una condición If.

Demo 42

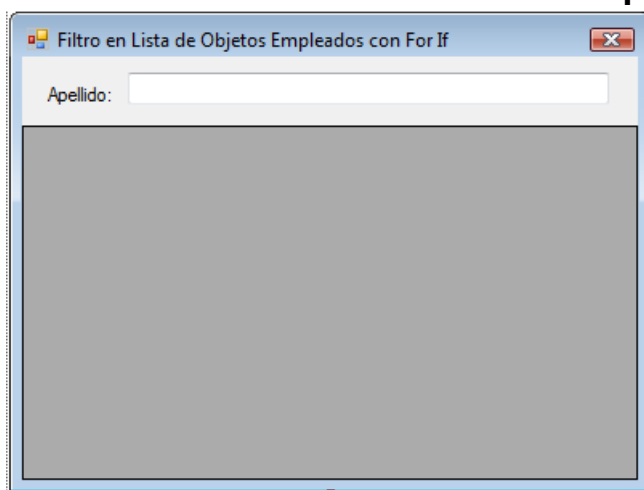
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo42.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmFiltroEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmFiltroEmpleado
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,300
	StartPosition	CenterScreen
	Text	Filtro en Lista de Objetos

		Empleados con For If
Label1	Name AutoSize Location Text	lblApellido True 13, 17 Apellido:
TextBox1	Name Location Size	txtApellido 66, 10 300, 20
DataGridView1	Name AutoSizeColumnsM Location ReadOnly SelectionMode Size	dgvEmpleado AllCells 0, 43 True FullRowSelect 384, 221

- El diseño del formulario debe quedar similar al gráfico 3.33:

Gráfico 3.33: Diseño del formulario Filtro de Empleados



- Ingresar al editor de código y escribir el siguiente código:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Public Class frmFiltroEmpleado
    Private lbeEmpleado As List(Of beEmpleado)
```

```
Private Sub ListarEmpleados(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles MyBase.Load  
    Dim obrEmpleado As New brEmpleado  
    lbeEmpleado = obrEmpleado.Listar  
    dgvEmpleado.DataSource = lbeEmpleado  
End Sub
```

```
Private Sub FiltrarEmpleadosPorApellido(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles txtApellido.TextChanged  
    Dim lbeFiltro As New List(Of beEmpleado)  
    Dim obeEmpleado As beEmpleado  
    For I As Integer = 0 To lbeEmpleado.Count - 1  
        If lbeEmpleado(I).Apellido.ToUpper.Contains _  
        (txtApellido.Text.ToUpper) Then  
            obeEmpleado = New beEmpleado  
            obeEmpleado.Codigo = lbeEmpleado(I).Codigo  
            obeEmpleado.Apellido = lbeEmpleado(I).Apellido  
            obeEmpleado.Nombre = lbeEmpleado(I).Nombre  
            obeEmpleado.FechaNac = lbeEmpleado(I).FechaNac  
            lbeFiltro.Add(obeEmpleado)  
        End If  
    Next  
    dgvEmpleado.DataSource = lbeFiltro  
End Sub  
End Class
```

Nota: En el código anterior se ha creado un objeto y se ha llenado, ya que no se puede asignar un objeto a otro porque se haría por referencia.

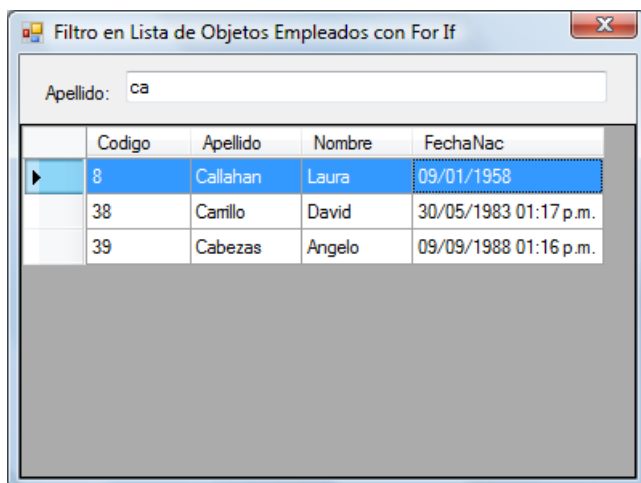
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>  
<appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
    data source=Lduenas\MCTS;initial catalog=Northwind"/>  
</appSettings>
```

</configuration>

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.34: Ejecución del formulario Filtro de Empleados



	Codigo	Apellido	Nombre	FechaNac
▶	8	Callahan	Laura	09/01/1958
	38	Camilo	David	30/05/1983 01:17 p.m.
	39	Cabezas	Angelo	09/09/1988 01:16 p.m.

Usando Predicados

La forma moderna de filtrar una lista de objetos es usar el método FindAll de la lista de objetos que usa predicados que son delegados o punteros a funciones.

Usar delegados es más legible que escribir la condición de filtro dentro de un bucle. A continuación un par de ejemplos que muestra como filtrar empleados y productos usando predicados.

Demo 43

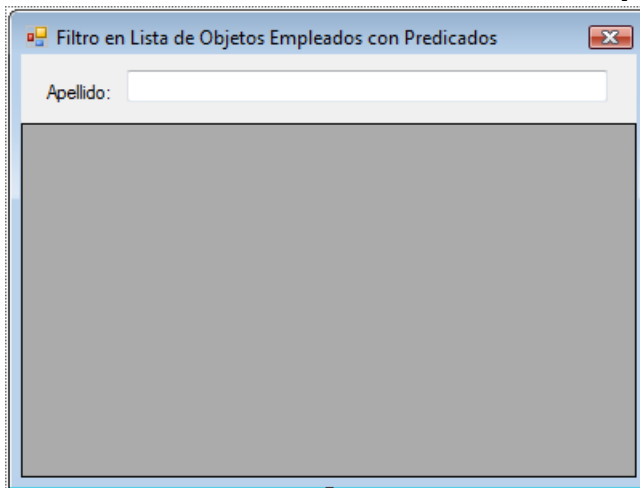
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo43.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmFiltroEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmFiltroEmpleado FixedSingle False False 400,300 CenterScreen Filtro en Lista de Objetos Empleados con Predicados
Label1	Name AutoSize Location Text	lblApellido True 13, 17 Apellido:
TextBox1	Name Location Size	txtApellido 66, 10 300, 20
DataGridView1	Name	dgvEmpleado

	AutoSizeColumnsM Location ReadOnly SelectionMode Size	AllCells 0, 43 True FullRowSelect 384, 221

- El diseño del formulario debe quedar similar al gráfico 3.35:

Gráfico 3.35: Diseño del formulario Filtro de Empleados



- Ingresar al editor de código y escribir el siguiente código:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Public Class frmFiltroEmpleado
```

```
    Private lbeEmpleado As List(Of beEmpleado)
```

```
    Private Sub ListarEmpleados(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Dim obrEmpleado As New brEmpleado
```

```
        lbeEmpleado = obrEmpleado.Listar
```

```
        dgvEmpleado.DataSource = lbeEmpleado
```

```
    End Sub
```

```
    Private Function BuscarEmpleado(ByVal obeEmpleado As beEmpleado) _
        As Boolean
```

```
Return (obeEmpleado.Apellido.ToUpper.Contains _  
(txtApellido.Text.ToUpper))  
End Function
```

```
Private Sub FiltrarEmpleadosPorApellido(ByVal sender As System.Object,  
ByVal e As System.EventArgs) Handles txtApellido.TextChanged  
    Dim lbeFiltro As New List(Of beEmpleado)  
    Dim pred As New Predicate(Of beEmpleado)(AddressOf BuscarEmpleado)  
    dgvEmpleado.DataSource = lbeEmpleado.FindAll(pred)  
End Sub  
End Class
```

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
    data source=Lduenas\MCTS;initial catalog=Northwind"/>  
  </appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.36: Ejecución del formulario Filtro de Empleados

The screenshot shows a Windows application window titled "Filtro en Lista de Objetos Empleados con Predicados". Inside the window, there is a search field labeled "Apellido:" containing the text "ca". Below the search field is a table with the following data:

	Codigo	Apellido	Nombre	FechaNac
▶	8	Callahan	Laura	09/01/1958
	38	Camilo	David	30/05/1983 01:17 p.m.
	39	Cabezas	Angelo	09/09/1988 01:16 p.m.

The first row of the table is highlighted in blue. The rest of the window area below the table is a solid gray rectangle.

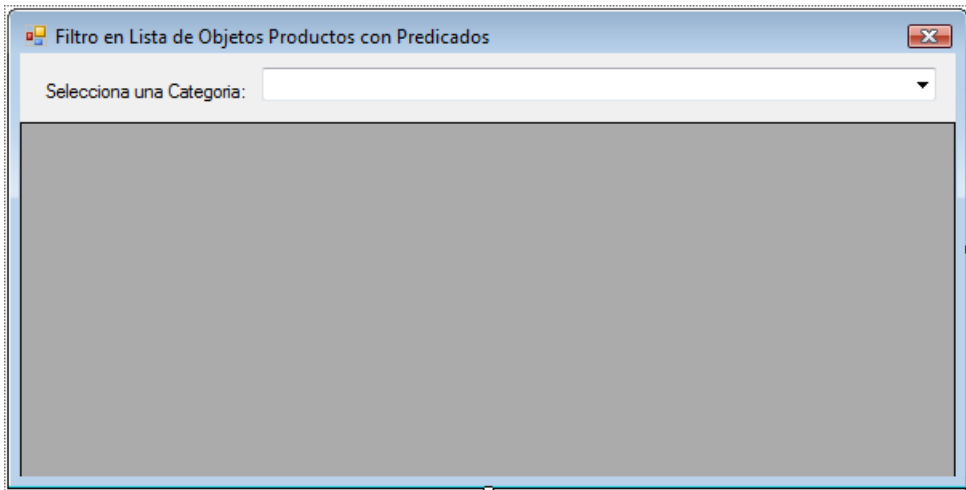
Demo 44

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo44.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmFiltroProducto.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmFiltroProducto FixedSingle False False 600,300 CenterScreen Filtro en Lista de Objetos Productos con Predicados
Label1	Name AutoSize Location Text	lblCategoria True 13, 17 Selecciona una Categoria:
ComboBox1	Name Location Size	cboCategoria 151, 9 421, 21
DataGridView1	Name AutoSizeColumnsMode Location ReadOnly SelectionMode Size	dgvProducto AllCells 0, 43 True FullRowSelect 584, 225

- El diseño del formulario debe quedar similar al gráfico 3.37:

Gráfico 3.37: Diseño del formulario Filtro de Productos x Categoría



- Ingresar al editor de código y escribir el siguiente código:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Public Class frmFiltroProducto
```

```
    Private lbeProducto As List(Of beProducto)
```

```
    Private Sub ListarProductos(ByVal sender As System.Object, _
```

```
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Dim obrCategoria As New brCategoria
```

```
        Dim lbeCategoria As List(Of beCategoria) = obrCategoria.Listar
```

```
        Dim obeCategoria As New beCategoria
```

```
        obeCategoria.Codigo = 0
```

```
        obeCategoria.Nombre = "Seleccione"
```

```
        lbeCategoria.Insert(0, obeCategoria)
```

```
        With cboCategoria
```

```
            .DataSource = lbeCategoria
```

```
            .DisplayMember = "Nombre"
```

```
            .ValueMember = "Codigo"
```

```
        End With
```

```
        Dim obrProducto As New brProducto
```

```
lbeProducto = obrProducto.Listar
End Sub

Private Function BuscaProducto(ByVal obeProducto As beProducto)
    Return (obeProducto.IdCategoria = cboCategoria.Selected.Value)
End Function

Private Sub FiltrarProductosxCategoria(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles _
    cboCategoria.SelectionChangeCommitted
    If cboCategoria.SelectedIndex = 0 Then
        dgvProducto.DataSource = Nothing
    Else
        Dim pred As New Predicate(Of beProducto)(AddressOf BuscaProducto)
        dgvProducto.DataSource = lbeProducto.FindAll(pred)
    End If
End Sub
End Class
```

Nota: En el código anterior se ha creado un objeto obeCategoria para agregar un elemento a la lista de objetos de tal forma que aparezca Seleccione en el ComboBox. Nunca debe insertarse un registro directamente en la tabla solo para ver este elemento en un control.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.38: Ejecución del formulario Filtro de Productos x Categoría

Filtro en Lista de Objetos Productos con Predicados

Selecciona una Categoría: Beverages

	Codigo	Nombre	IdProveedor	IdCategoria	PrecioUnitario	Stock
▶	1	Chai	2	1	10.0000	80
	3	Chicha	10	1	10.0000	39
	24	Guaraná Fantástica	10	1	4.5000	20
	34	Sasquatch Ale	16	1	14.0000	111
	35	Steeleye Stout	16	1	18.0000	20
	38	Côte de Blaye	18	1	263.5000	17
	39	Chartreuse verte	18	1	18.0000	69
	43	Ipoh Coffee	20	1	46.0000	17
	67	Laughing Lumberjack Lager	16	1	14.0000	1

3.3 Ordenando y Buscando datos en una Lista de Objetos

Cuando los datos a consultar son demasiados podemos usar el filtro pero si los datos no son demasiados podemos usar búsqueda de datos por una cierta columna para lo cual primero deben estar ordenados los datos por dicha columna.

Para ordenar se usa el método Sort de la lista de objetos seguido de un comparador que es una clase que implementa la interface IComparer conteniendo una función que recibe como parámetros 2 objetos y retorna un entero indicando el tipo de orden.

Para buscar un objeto se usa el método FindIndex de la lista de objetos que devuelve un entero indicando la posición en la lista de objetos en la que se encuentra el dato buscado a través de predicados.

A continuación se presenta un ejemplo que implementa una búsqueda por apellido de los empleados.

Demo 45

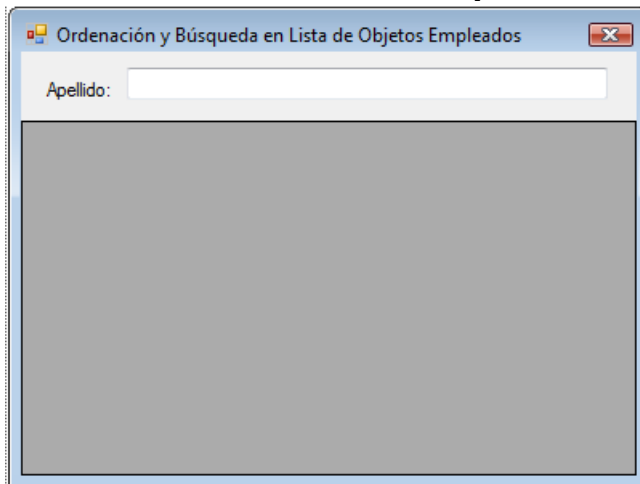
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo45.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmBuscaEmpleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmBuscaEmpleado
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,300
	StartPosition	CenterScreen

	Text	Ordenación y Búsqueda en Lista de Objetos Empleados
Label1	Name AutoSize Location Text	lblApellido True 13, 17 Apellido:
TextBox1	Name Location Size	txtApellido 66, 10 300, 20
DataGridView1	Name AutoSizeColumnsM Location ReadOnly SelectionMode Size	dgvEmpleado AllCells 0, 43 True FullRowSelect 384, 221

- El diseño del formulario debe quedar similar al gráfico 3.39:

Gráfico 3.39: Diseño del formulario Búsqueda de Empleados



- Agregar una clase para crear un comparador para ordenar la lista de empleados: del menú "Project" seleccionar "Add Class" y escribir como nombre: ucComparaEmpleado.
- Escribir el siguiente código:

```
Imports Northwind.LibBusinessEntities
```

```
Public Class ucComparaEmpleado
    Implements IComparer(Of beEmpleado)

    Public Function Compare(ByVal x As _
        Northwind.LibBusinessEntities.beEmpleado, _
        ByVal y As Northwind.LibBusinessEntities.beEmpleado) As Integer _
        Implements System.Collections.Generic.IComparer _
            (Of Northwind.LibBusinessEntities.beEmpleado).Compare
        Return (x.Apellido.ToUpper.CompareTo(y.Apellido.ToUpper))
    End Function
End Class
```

Nota: Al escribir la línea con la instrucción Implements y dar Enter el editor crea automáticamente la plantilla de código con la función a implementar.

- Ingresar al código del formulario y escribir lo siguiente:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules

Public Class frmBuscaEmpleado
    Private lbeEmpleado As List(Of beEmpleado)
    Private bs As New BindingSource

    Private Sub ListarEmpleados(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim obrEmpleado As New brEmpleado
        lbeEmpleado = obrEmpleado.Listar
        Dim oucComparaEmpleado As New ucComparaEmpleado
        lbeEmpleado.Sort(oucComparaEmpleado)
        bs.DataSource = lbeEmpleado
        dgvEmpleado.DataSource = bs
    End Sub

    Private Function BuscarEmpleado(ByVal obeEmpleado As beEmpleado) _
        As Boolean
        Return (obeEmpleado.Apellido.ToUpper.StartsWith _
            (txtApellido.Text.ToUpper))
    End Function
```

```
Private Sub BuscarEmpleadosPorApellido(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles txtApellido.TextChanged
    Dim lbeFiltro As New List(Of beEmpleado)
    Dim pred As New Predicate(Of beEmpleado)(AddressOf BuscarEmpleado)
    Dim pos As Integer = lbeEmpleado.FindIndex(pred)
    If pos > -1 Then bs.Position = pos
End Sub
End Class
```

Nota: En el código anterior la grilla no se enlaza directamente a la lista de objetos sino se hace a través del BindingSource para poder posicionarse sobre la grilla.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.40: Ejecución del formulario Búsqueda de Empleados

Ordenación y Búsqueda en Lista de Objetos Empleados

Apellido:

	Codigo	Apellido	Nombre	FechaNac
	5	Buchanan	Steven	04/03/1955
	39	Cabezas	Angelo	09/09/1988 01:16 p.m.
	8	Callahan	Laura	09/01/1958
▶	38	Camillo	David	30/05/1983 01:17 p.m.
	21	Davolio	Nancy	08/12/1948
	113	Davolio	Nancy2	08/12/1948
	1	Davolio	Nancy2	08/12/1948
	9	Dodsworth	Anne	27/01/1966
	36	Duenas	Luis	12/09/1970 01:15 p.m.

3.4 Mantenimiento con Objetos

En esta última parte veremos como realizar un mantenimiento usando objetos, para lo cual necesitamos primero crear los procedimientos almacenados que inserten, actualizen y eliminen, luego agregar los servicios de datos y reglas de negocio para finalmente crear la aplicación de mantenimiento.

A continuación presentamos un ejemplo de mantenimiento con objetos para los Empleados.

Creando los Procedimientos Almacenados

Crear los siguientes procedimientos almacenados:

- Procedimiento almacenado que adiciona un empleado

```
Create Procedure usp_Employees_Ins
    @LastName nvarchar(20),
    @FirstName nvarchar(10),
    @BirthDate datetime
AS
INSERT INTO Employees
(
    LastName,
    FirstName,
    BirthDate
)
VALUES
(
    @LastName,
    @FirstName,
    @BirthDate
)
return @@identity
```

- Procedimiento almacenado que actualiza un empleado

```
Create Procedure usp_Employees_Upd
```

```
@EmployeeID int,  
@LastName nvarchar(20),  
@FirstName nvarchar(10),  
@BirthDate datetime  
AS  
UPDATE Employees  
SET  
    LastName=@LastName,  
    FirstName=@FirstName,  
    BirthDate=@BirthDate  
WHERE  
    EmployeeID=@EmployeeID
```

- Procedimiento almacenado que elimina un empleado

```
Create Procedure usp_Employees_Del  
    @EmployeeID int  
AS  
DELETE FROM Employees  
WHERE EmployeeID=@EmployeeID
```

Modificando la Librería de Acceso a Datos

- Del menú "File", seleccionar "Open Project" y luego abrir el proyecto: "Northwind.LibDataAccess".
- Modificar la clase daEmpleado para aumentar lo siguiente:

```
Public Function fAdicionar(ByVal con As SqlConnection, _
    ByVal obeEmpleado As beEmpleado) As Integer
    Dim N As Integer = -1
    Dim cmd As New SqlCommand("usp_Employees_Ins", con)
    cmd.CommandType = CommandType.StoredProcedure
    Dim par1 As SqlParameter = _
    cmd.Parameters.Add("@LastName", SqlDbType.NVarChar, 20)
    par1.Direction = ParameterDirection.Input
    par1.Value = obeEmpleado.Apellido
    Dim par2 As SqlParameter = _
    cmd.Parameters.Add("@FirstName", SqlDbType.NVarChar, 10)
    par2.Direction = ParameterDirection.Input
    par2.Value = obeEmpleado.Nombre
    Dim par3 As SqlParameter = _
    cmd.Parameters.Add("@BirthDate", SqlDbType.DateTime)
    par3.Direction = ParameterDirection.Input
    par3.Value = obeEmpleado.FechaNac
    Dim par4 As SqlParameter = _
    cmd.Parameters.Add("@@identity", SqlDbType.Int)
    par4.Direction = ParameterDirection.ReturnValue
    N = cmd.ExecuteNonQuery
    If N > 0 Then
        Return (par4.Value)
    Else
        Return (-1)
    End If
End Function
```

```
Public Function fActualizar(ByVal con As SqlConnection, _
    ByVal obeEmpleado As beEmpleado) As Boolean
    Dim Exito As Boolean = False
    Dim cmd As New SqlCommand("usp_Employees_Upd", con)
    cmd.CommandType = CommandType.StoredProcedure
    Dim par1 As SqlParameter = _
```

```
cmd.Parameters.Add("@EmployeeID", SqlDbType.Int)
par1.Direction = ParameterDirection.Input
par1.Value = obeEmpleado.Codigo
Dim par2 As SqlParameter = _
cmd.Parameters.Add("@LastName", SqlDbType.NVarChar, 20)
par2.Direction = ParameterDirection.Input
par2.Value = obeEmpleado.Apellido
Dim par3 As SqlParameter = _
cmd.Parameters.Add("@FirstName", SqlDbType.NVarChar, 10)
par3.Direction = ParameterDirection.Input
par3.Value = obeEmpleado.Nombre
Dim par4 As SqlParameter = _
cmd.Parameters.Add("@BirthDate", SqlDbType.DateTime)
par4.Direction = ParameterDirection.Input
par4.Value = obeEmpleado.FechaNac
Dim N As Integer = cmd.ExecuteNonQuery
Exito = (N > 0)
Return (Exito)
End Function
```

```
Public Function fEliminar(ByVal con As SqlConnection, _
ByVal CodEmpleado As Integer) As Boolean
    Dim Exito As Boolean = False
    Dim cmd As New SqlCommand("usp_Employees_Del", con)
    cmd.CommandType = CommandType.StoredProcedure
    Dim par As SqlParameter = _
    cmd.Parameters.Add("@EmployeeID", SqlDbType.Int)
    par.Value = CodEmpleado
    Dim N As Integer = cmd.ExecuteNonQuery
    Exito = (N > 0)
    Return (Exito)
End Function
```

- Finalmente, del menú "Build" seleccionar "Build Northwind.LibData Access" para actualizar la dll con los cambios realizados.

Modificando la Librería de Reglas del Negocio

- Del menú "File", seleccionar "Open Project" y luego abrir el proyecto: "Northwind.LibBusinessRules".
- Modificar la clase brEmpleado para aumentar lo siguiente:

```
Public Function Adicionar(ByVal obeEmpleado As beEmpleado) As Integer
    Dim N As Integer = -1
    Using con As New SqlConnection(strConexion)
        Try
            con.Open()
            Dim odaEmpleado As New daEmpleado
            N = odaEmpleado.fAdicionar(con, obeEmpleado)
        Catch ex As Exception
            'Grabar el Log de error
            N = -1
        End Try
    End Using
    Return (N)
End Function
```

```
Public Function Actualizar(ByVal obeEmpleado As beEmpleado) As Boolean
    Dim exito As Boolean = False
    Using con As New SqlConnection(strConexion)
        Try
            con.Open()
            Dim odaEmpleado As New daEmpleado
            exito = odaEmpleado.fActualizar(con, obeEmpleado)
        Catch ex As Exception
            'Grabar el Log de error
        End Try
    End Using
    Return (exitito)
End Function
```

```
Public Function Eliminar(ByVal CodEmpleado As Integer) As Boolean
    Dim exito As Boolean = False
    Using con As New SqlConnection(strConexion)
        Try
            con.Open()
```

```
    Dim odaEmpleado As New daEmpleado
    exito = odaEmpleado.fEliminar(con, CodEmpleado)
Catch ex As Exception
    'Grabar el Log de error
End Try
End Using
Return (exito)
End Function
```

- Finalmente, del menú "Build" seleccionar "Build Northwind.LibBusiness Rules" para actualizar la dll con los cambios realizados.

Demo 46

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo46.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmMantenimiento Empleado.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmMantenimientoEmpleado FixedSingle False False 400,400 CenterScreen Mantenimiento de Empleados con Objetos
Label1	Name AutoSize Location Text	lblCodigo True 26, 33 Código:
TextBox1	Name Location ReadOnly Size	txtCodigo 100,26 True 47,20
Label2	Name AutoSize Location Text	lblApellido True 26, 59 Apellido:
TextBox2	Name Location Size	txtApellido 100,52 178,20
Label3	Name AutoSize	lblNombre True

	Location Text	26, 85 Nombre:
TextBox3	Name Location Size	txtNombre 100,78 178,20
Label4	Name AutoSize Location Text	lblFechaNac True 26, 111 Fecha Nac:
DateTimePicker1	Name Format Location Size	dtpFechaNac Short 100,105 88,20
Button1	Name Cursor Location Size Text	btnNuevo Hand 297, 22 75, 23 Nuevo
Button2	Name Cursor Location Size Text	btnAdicionar Hand 297, 22 75, 50 Adicionar
Button3	Name Cursor Location Size Text	btnActualizar Hand 297, 78 75, 23 Actualizar
Button4	Name Cursor Location Size Text	btnEliminar Hand 297, 106 75, 23 Eliminar
DataGridView1	Name Location MultiSelect ReadOnly SelectionMode Size	dgvEmpleado 13, 151 False True FullRowSelect 367, 210

- El diseño del formulario debe quedar similar al gráfico 3.41:

Gráfico 3.41: Diseño del formulario Mantenimiento de Empleados

- Ingresar al editor de código y escribir el siguiente código.

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Imports System.Drawing.Drawing2D
```

```
Public Class frmMantenimientoEmpleado
```

```
    Private lbeEmpleado As New List(Of beEmpleado)
```

```
    Private obrEmpleado As New brEmpleado
```

```
    Private bs As New BindingSource
```

```
    Private Sub DibujarRecuadro(ByVal sender As Object, _
```

```
        ByVal e As System.Windows.Forms.PaintEventArgs) Handles Me.Paint
```

```
        Dim recData As New Rectangle(15, 10, 270, 130)
```

```
        Dim degData As New LinearGradientBrush _
```

```
            (recData, Color.Aqua, Color.Blue, _
```

```
            LinearGradientMode.BackwardDiagonal)
```

```
        e.Graphics.FillRectangle(degData, recData)
```

```
        Dim recBoton As New Rectangle(290, 10, 90, 130)
```

```
        Dim degBoton As New LinearGradientBrush _
```

```
            (recBoton, Color.Aqua, Color.Green, _
```

```
LinearGradientMode.BackwardDiagonal)
e.Graphics.FillRectangle(degBoton, recBoton)
End Sub
```

```
Private Sub PersonalizarGrilla()
    dgvEmpleado.Columns(0).HeaderText = "ID"
    dgvEmpleado.Columns(0).Width = 40
    dgvEmpleado.Columns(1).HeaderText = "Apellido"
    dgvEmpleado.Columns(1).Width = 80
    dgvEmpleado.Columns(2).HeaderText = "Nombre"
    dgvEmpleado.Columns(2).Width = 80
    dgvEmpleado.Columns(3).HeaderText = "Fecha Nac"
    dgvEmpleado.Columns(3).Width = 100
End Sub
```

```
Private Sub ListarEmpleados()
    lbeEmpleado = obrEmpleado.Listar
    bs.DataSource = lbeEmpleado
    dgvEmpleado.DataSource = bs
End Sub
```

```
Private Sub CargarDatos(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    ListarEmpleados()
    PersonalizarGrilla()
End Sub
```

```
Private Sub MostrarDatos(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles dgvEmpleado.SelectionChanged
    With dgvEmpleado.CurrentRow
        txtCodigo.Text = .Cells(0).Value
        txtApellido.Text = .Cells(1).Value
        txtNombre.Text = .Cells(2).Value
        dtpFechaNac.Value = .Cells(3).Value
    End With
End Sub
```

```
Private Sub Nuevo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnNuevo.Click
    txtCodigo.Clear()
```

```
txtApellido.Clear()
txtNombre.Clear()
dtpFechaNac.Value = Now
txtApellido.Focus()
End Sub

Private Sub AdicionarEmpleado(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAdicionar.Click
    If txtApellido.Text.Trim <> "" Then
        If txtNombre.Text.Trim <> "" Then
            Dim Diferencia As TimeSpan = _
                Now.Date.Subtract(dtpFechaNac.Value.Date)
            If Diferencia.Days > 0 Then
                Dim obeEmpleado As New beEmpleado
                With obeEmpleado
                    .Apellido = txtApellido.Text
                    .Nombre = txtNombre.Text
                    .FechaNac = dtpFechaNac.Value
                End With
                Dim N As Integer = obrEmpleado.Adicionar(obeEmpleado)
                If N > 0 Then
                    MessageBox.Show("Se Adicionó el Empleado", _
                        "Aviso", MessageBoxButtons.OK, _
                        MessageBoxIcon.Information)
                Else
                    MessageBox.Show("No se pudo Adicionar el Empleado", _
                        "Error", MessageBoxButtons.OK, _
                        MessageBoxIcon.Exclamation)
                End If
                ListarEmpleados()
                bs.Position = bs.Count - 1
            Else
                MessageBox.Show("La Fecha de Nacimiento debe ser menor a hoy")
                dtpFechaNac.Focus()
            End If
        Else
            MessageBox.Show("Falta ingresar el Nombre", "Aviso", _
                MessageBoxButtons.OK, MessageBoxIcon.Information)
            txtNombre.Focus()
        End If
    End Sub
```

```
End If
Else
    MessageBox.Show("Falta ingresar el Apellido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    txtApellido.Focus()
End If
End Sub

Private Sub ActualizarEmpleado(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnActualizar.Click
    If txtApellido.Text.Trim <> "" Then
        If txtNombre.Text.Trim <> "" Then
            Dim Diferencia As TimeSpan = _
                Now.Date.Subtract(dtpFechaNac.Value.Date)
            If Diferencia.Days > 0 Then
                Dim obeEmpleado As New beEmpleado
                With obeEmpleado
                    .Codigo = Integer.Parse(txtCodigo.Text)
                    .Apellido = txtApellido.Text
                    .Nombre = txtNombre.Text
                    .FechaNac = dtpFechaNac.Value
                End With
                Dim exito As Boolean = obrEmpleado.Actualizar(obeEmpleado)
                If exito Then
                    MessageBox.Show("Se Actualizó el Empleado", _
                        "Aviso", MessageBoxButtons.OK, _
                        MessageBoxIcon.Information)
                Else
                    MessageBox.Show("No se pudo Actualizar el Empleado", _
                        "Error", MessageBoxButtons.OK, _
                        MessageBoxIcon.Exclamation)
                End If
                Dim pos As Integer = bs.Position
                ListarEmpleados()
                bs.Position = pos
            Else
                MessageBox.Show("La Fecha de Nacimiento debe ser menor a hoy")
                dtpFechaNac.Focus()
            End If
        End If
    End If
End Sub
```

```
Else
    MessageBox.Show("Falta ingresar el Nombre", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    txtNombre.Focus()
End If
Else
    MessageBox.Show("Falta ingresar el Apellido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    txtApellido.Focus()
End If
End Sub

Private Sub EliminarEmpleado(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnEliminar.Click
    If txtCodigo.Text <> "" Then
        If MessageBox.Show("Estas seguro de eliminarlo", "Aviso", _
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) = _
            Windows.Forms.DialogResult.Yes Then
            Dim CodEmpleado As Integer = Integer.Parse(txtCodigo.Text)
            Dim exito As Boolean = obrEmpleado.Eliminar(CodEmpleado)
            If exito Then
                MessageBox.Show("Se Eliminó el Empleado", _
                    "Aviso", MessageBoxButtons.OK, _
                    MessageBoxIcon.Information)
            Else
                MessageBox.Show("No se pudo Eliminar el Empleado", _
                    "Error", MessageBoxButtons.OK, _
                    MessageBoxIcon.Exclamation)
            End If
            Dim pos As Integer = bs.Position
            ListarEmpleados()
            bs.Position = pos
        End If
    Else
        MessageBox.Show("Falta seleccionar el Empleado", "Aviso", _
            MessageBoxButtons.OK, MessageBoxIcon.Information)
    End If
End Sub
End Class
```

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
      data source=Lduenas\MCTS;initial catalog=Northwind"/>  
  </appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.42: Ejecución del formulario Mantenimiento de Empleados con Objetos

	ID	Apellido	Nombre	Fecha Nac
▶	1	Davolio	Nancy2	08/12/1948
	2	Fuller	Andrew	19/02/1952
	3	Leverling	Janet1	30/08/1963
	4	Peacock	Margaret	19/09/1937
	5	Buchanan	Steven	04/03/1955
	6	Suyama	Michael	02/07/1963
	7	King	Robert	29/05/1960
	8	Callahan	Laura	09/01/1958
	9	Dodsworth	Anna	27/01/1966

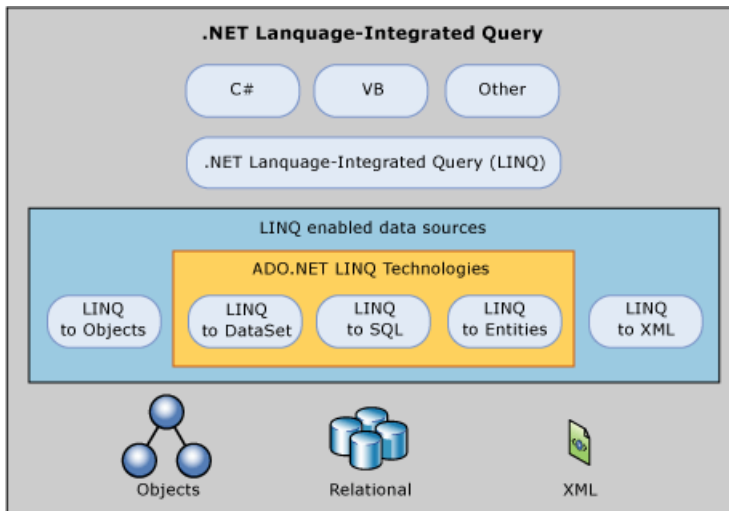
4. LINQ

Las Consultas Integradas en el Lenguaje (Language Integrated Query o LINQ) permiten manejar los datos desde el Lenguaje de programación Visual Basic en vez de hacerlo mediante instrucciones SQL, como Transact SQL para SQL Server.

El objetivo principal de LINQ es que en vez de que el programador use diferentes lenguajes o modelos de programación para acceder a datos lo haga usando un solo lenguaje de consulta que trabaja con los datos en memoria.

Existen varios tipos de LINQ como lo muestra el gráfico 3.43: a objetos, a DataSet, a SQL, a Entidades y a XML. LINQ a objetos permite trabajar con listas o arreglos de objetos en vez de usar predicados; LINQ a DataSet, SQL y a Entidades permiten trabajar con bases de datos en vez de usar el DataSet e instrucciones SQL; finalmente, LINQ a XML permite acceder a XML en vez de usar XMLTextReader o DOM de XML.

Gráfico 3.43: ADO .NET y LINQ



Para ver mas información sobre LINQ ver la referencia 22 al final del libro. En esta parte solo veremos LINQ con bases de datos es decir LINQ a DataSet, LINQ a SQL y LINQ a Entidades.

4.1 LINQ a DataSet

LINQ a DataSet facilita el trabajo con el DataSet y se implementa a través de los métodos de extensión en las clases DataRowExtensions y DataTableExtensions. Estas extensiones agregan operadores que permiten comparar secuencias de filas así como métodos que proporcionan acceso a los campos de un DataRow.

Para habilitar la funcionalidad de LINQ a DataSet hay que hacer referencia al ensamblado System.Data.DataSetExtensions.dll y para usar el DataSet se usa el ensamblado System.Data.dll.

Nota: Al crear una aplicación Windows, ya viene por defecto referenciado ambas librerías, es decir, ya estamos listos para trabajar con LINQ a DataSet.

Las consultas de LINQ a DataSet se pueden escribir en 2 sintaxis distintas: sintaxis de expresiones de consulta y sintaxis de consultas basadas en métodos.

Sintáxis de Expresiones de Consulta

Las expresiones de consulta son una sintaxis de consulta declarativa. Esta sintaxis permite a un desarrollador escribir consultas en Visual Basic en un formato similar a SQL. Si se utiliza la sintaxis de expresiones de consulta, se pueden realizar incluso operaciones complejas de filtrado, ordenación y agrupamiento en orígenes de datos con código mínimo.

Demo 47

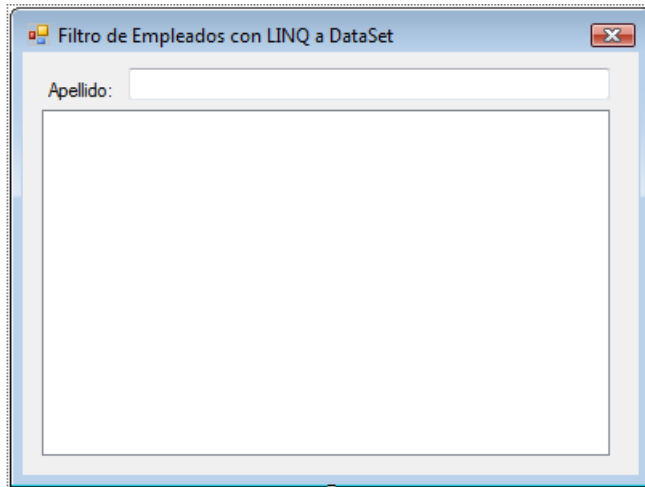
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo47.
- Cambiar de nombre al formulario de form1.vb a frmFiltroEmpleado.vb

- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmFiltroEmpleado FixedSingle False False 400,300 CenterScreen Filtro de Empleados con LINQ a DataSet
Label1	Name AutoSize Location Text	lblApellido True 13, 17 Apellido:
TextBox1	Name Location ReadOnly Size	txtApellido 66,10 True 300,20
ListView1	Name FullRowSelect GridLines HotTracking Location Size	lvwEmpleado True True True 12, 36 354, 216

- El diseño del formulario debe quedar similar al gráfico 3.44:

Gráfico 3.44: Diseño del formulario Filtro de Empleados LINQ a DataSet



- Ingresar al editor de código y escribir el siguiente código:

```
Imports System.Data
Imports System.Data.SqlClient

Public Class frmFiltroEmpleado
    Private dst As New DataSet

    Private Sub LlenarListView()
        Dim fila As ListViewItem
        IvwEmpleado.BeginUpdate()
        For Each empleado In dst.Tables(0).Rows
            fila = IvwEmpleado.Items.Add(empleado(0))
            fila.SubItems.Add(empleado(1))
            fila.SubItems.Add(empleado(2))
        Next
        IvwEmpleado.EndUpdate()
    End Sub

    Private Sub ListarEmpleados(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Using con As New SqlConnection("uid=sa;pwd=123456;
            data source=Lduenas\MCTS;initial catalog=Northwind")
            con.Open()
```

```
Dim dap As New SqlDataAdapter _
("Select EmployeeID,LastName,FirstName From Employees", con)
dap.Fill(dst, "Empleados")
End Using
With lvwEmpleado
    .Columns.Add("Codigo", 50, HorizontalAlignment.Right)
    .Columns.Add("Apellido", 100, HorizontalAlignment.Left)
    .Columns.Add("Nombre", 100, HorizontalAlignment.Left)
    .View = View.Details
End With
LlenarListView()
End Sub

Private Sub FiltrarEmpleadosPorApellido(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles txtApellido.TextChanged
    lvwEmpleado.Items.Clear()
    If txtApellido.Text <> "" Then
        Dim res = From empleado In dst.Tables(0).AsEnumerable _
        Select empleado Where empleado(1).ToString.ToUpper. _
        Contains(txtApellido.Text.ToUpper)
        lvwEmpleado.BeginUpdate()
        Dim fila As ListViewItem
        For Each empleado In res
            fila = lvwEmpleado.Items.Add(empleado.Field(Of Int32)(0))
            fila.SubItems.Add(empleado.Field(Of String)(1))
            fila.SubItems.Add(empleado.Field(Of String)(2))
        Next
        lvwEmpleado.EndUpdate()
    Else
        LlenarListView()
    End If
End Sub
End Class
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.45: Ejecución del formulario Filtro de Empleados con LINQ a DataSet

Apellido: C

Codigo	Apellido	Nombre
4	Peacock	Margaret
5	Buchanan	Steven
8	Callahan	Laura
38	Camilo	David
39	Cabezas	Angelo

Sintáxis de Consultas basadas en Métodos

La otra forma de trabajar con consultas de LINQ a DataSet es usar las consultas basadas en métodos. La sintaxis de consultas basadas en métodos es una secuencia de llamadas a métodos directas a los métodos de operador de LINQ, pasando expresiones lambda como parámetros.

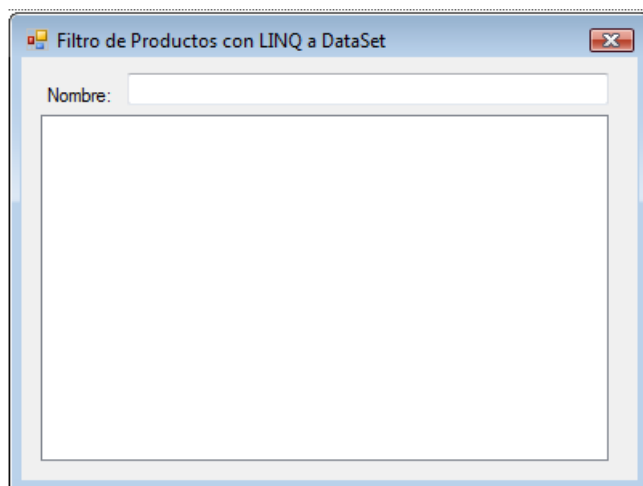
Demo 48

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo48.
- Cambiar de nombre al formulario de form1.vb a frmFiltroProducto.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmFiltroProducto FixedSingle False False 400, 300 CenterScreen Filtro de Productos con LINQ a DataSet
Label1	Name AutoSize Location Text	lblNombre True 13, 17 Nombre:
TextBox1	Name Location ReadOnly Size	txtNombre 66,10 True 300,20
ListView1	Name FullRowSelect GridLines HotTracking Location Size	lvwProducto True True True 12, 36 354, 216

- El diseño del formulario debe quedar similar al gráfico 3.46:

Gráfico 3.46: Diseño del formulario Filtro de Productos LINQ a DataSet



- Ingresar al editor de código y escribir el siguiente código:

```
Imports System.Data
Imports System.Data.SqlClient
```

```
Public Class frmFiltroProducto
    Private dst As New DataSet
```

```
    Private Sub LlenarListView()
        Dim fila As ListViewItem
        lvwProducto.BeginUpdate()
        For Each empleado In dst.Tables(0).Rows
            fila = lvwProducto.Items.Add(empleado(0))
            fila.SubItems.Add(empleado(1))
            fila.SubItems.Add(empleado(2))
        Next
        lvwProducto.EndUpdate()
    End Sub
```

```
    Private Sub ListarEmpleados(ByVal sender As System.Object, _
```

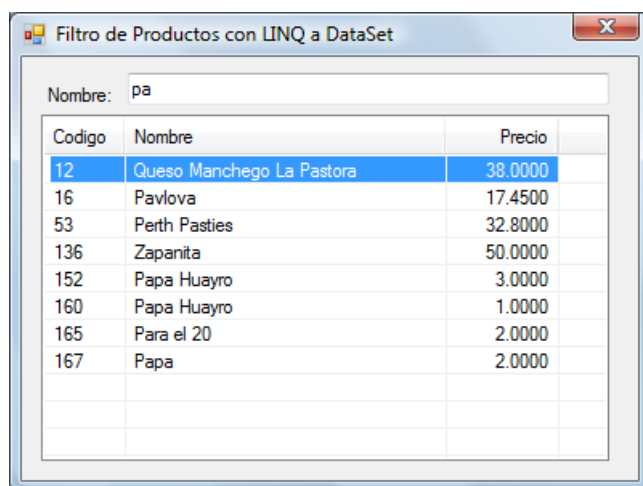
```
ByVal e As System.EventArgs) Handles MyBase.Load
    Using con As New SqlConnection("uid=sa;pwd=123456;
    data source=Lduenas\MCTS;initial catalog=Northwind")
        con.Open()
        Dim dap As New SqlDataAdapter _
        ("Select ProductID,ProductName,UnitPrice From Products", con)
        dap.Fill(dst, "Productos")
    End Using
    With lvwProducto
        .Columns.Add("Codigo", 50, HorizontalAlignment.Right)
        .Columns.Add("Nombre", 200, HorizontalAlignment.Left)
        .Columns.Add("Precio", 70, HorizontalAlignment.Right)
        .View = View.Details
    End With
    LlenarListView()
End Sub

Private Sub FiltrarProductosPorNombre(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles txtNombre.TextChanged
    lvwProducto.Items.Clear()
    If txtNombre.Text <> "" Then
        Dim res = dst.Tables(0).AsEnumerable() _
        .Select(Function(producto As DataRow) New With _
        { _
            .ProductID = producto.Field(Of Integer)("ProductID"), _
            .ProductName = producto.Field(Of String)("ProductName"), _
            .UnitPrice = producto.Field(Of Decimal)("UnitPrice") _
        })
        lvwProducto.BeginUpdate()
        Dim fila As ListViewItem
        For Each producto In res
            If producto.ProductName.ToUpper.Contains _
            (txtNombre.Text.ToUpper) Then
                fila = lvwProducto.Items.Add(producto.ProductID)
                fila.SubItems.Add(producto.ProductName)
                fila.SubItems.Add(producto.UnitPrice)
            End If
        Next
        lvwProducto.EndUpdate()
    Else
```

```
LlenarListView()  
End If  
End Sub  
End Class
```

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.47: Ejecución del formulario Filtro de Productos con LINQ a DataSet



Codigo	Nombre	Precio
12	Queso Manchego La Pastora	38.0000
16	Pavlova	17.4500
53	Perth Pasties	32.8000
136	Zapanita	50.0000
152	Papa Huayro	3.0000
160	Papa Huayro	1.0000
165	Para el 20	2.0000
167	Papa	2.0000

4.2 LINQ a SQL

En LINQ a SQL, el modelo de datos de una base de datos relacional se asigna a un modelo de objetos expresado en el lenguaje de programación usado por el programador. Cuando la aplicación se ejecuta, LINQ a SQL convierte a SQL las consultas integradas en el lenguaje en el modelo de objetos y las envía a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ a SQL los vuelve a convertir en objetos con los que pueda trabajar en su propio lenguaje de programación.

Hay 2 herramientas que permiten implementar LINQ a SQL: Object Relational Designer y SqlMetal. Los desarrolladores de Visual Studio normalmente utilizan el Object Relational Designer, que proporciona una interfaz de usuario para implementar muchas de las características de LINQ a SQL. La segunda forma de generar código es usando por línea de comandos el utilitario SQL Metal.

LINQ a SQL permite realizar todas las operaciones que se realizan generalmente con SQL: selección, adición, actualización y eliminación de registros.

Demo 49

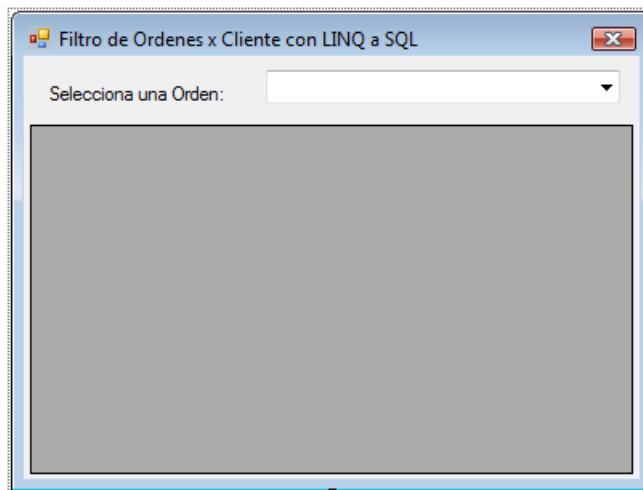
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo49.
- Cambiar de nombre al formulario de form1.vb a frmOrdenesxCiente.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmOrdenesxCiente
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	Size	400,300
	StartPosition	CenterScreen

	Text	Filtro de Ordenes x Cliente con LINQ a SQL
Label1	Name AutoSize Location Text	lblOrden True 13, 17 Selecciona una Orden:
ComboBox1	Name Location Size	cboOrden 151, 9 221, 21
DataGridView1	Name AllowUserToAddRow AllowUserToDeleteRow Location ReadOnly SelectionMode Size	dgvCliente False False 4, 43 True FullRowSelect 376, 218

- El diseño del formulario debe quedar similar al gráfico 3.46:

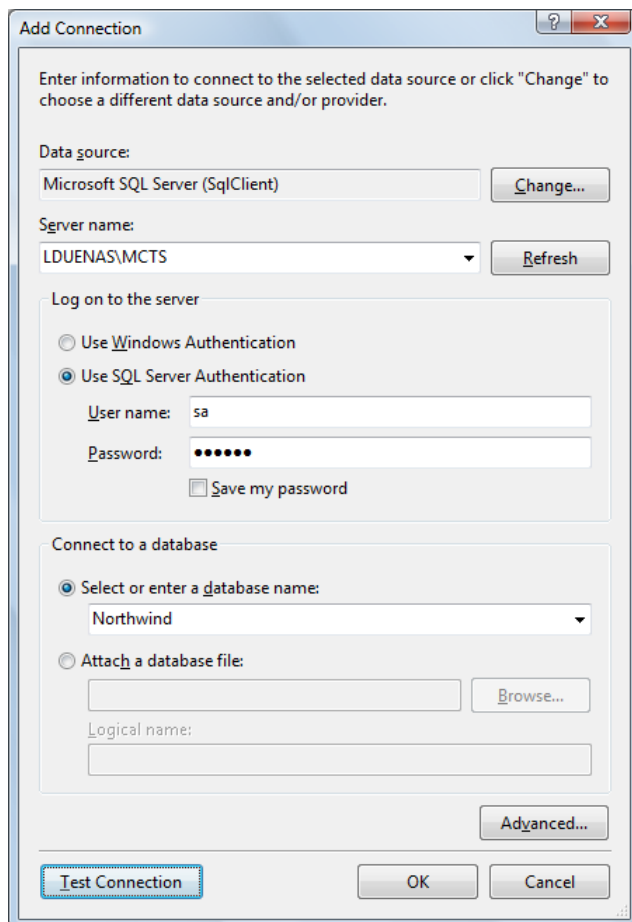
Gráfico 3.48: Diseño del formulario Filtro de Ordenes x Cliente con LINQ a SQL



- Agregar un modelo de clases de datos, del menú "Project" seleccionar "Add New Item" y en categoría elegir "Data" luego seleccionar "LINQ To SQL Classes".

- Escribir como nombre del modelo "ModeloClasesNW.dbml" y aparecerá el diseñador de clases.
- Clic al enlace que dice "Server Explorer" y aparecerá la ventana del explorador de servidores.
- Clic derecho a "Data Connections" en la ventana del explorador de servidores y seleccionar "Add Connections"
- Aparecerá el diálogo de agregar conexiones como lo muestra la figura siguiente.

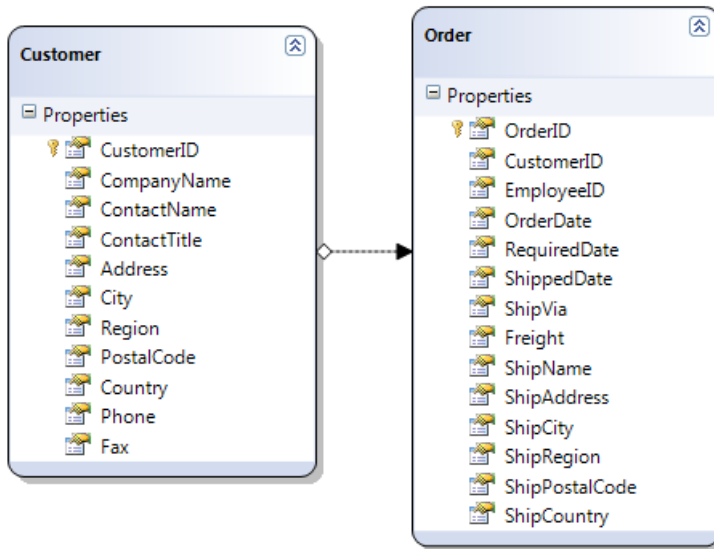
Gráfico 3.48: Diálogo de agregar conexiones



- Ingresar los datos de la conexión a la base de datos Northwind y "OK".

- Expandir la carpeta tablas y arrastrar las tablas "Customers" y "Orders" hacia el diseñador de clases.
- Se crearán 2 tablas en el diseñador tal como lo muestra la siguiente figura.

Gráfico 3.49: Diseñador Relacional de Objetos



- Escribir el siguiente código en el formulario:

```
Public Class frmOrdenesxCiente
Public Class frmOrdenesxCiente
    Private oNW As New ModeloClasesNWDataContext

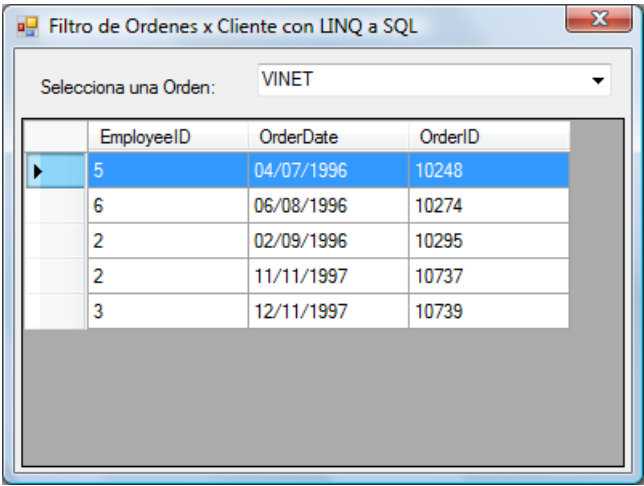
    Private Sub ListarClientes(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        cboOrden.DataSource = oNW.Orders
        cboOrden.DisplayMember = "CompanyName"
        cboOrden.ValueMember = "CustomerID"
        FiltrarOrdenesxCiente(Nothing, Nothing)
    End Sub

    Private Sub FiltrarOrdenesxCiente(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles
        cboOrden.SelectionChangeCommitted
```

```
Dim CodCliente As String = cboOrden.SelectedValue
dgvCliente.DataSource = From orden In oNW.Orders _
Where orden.CustomerID = CodCliente _
Select orden.OrderID, orden.OrderDate, orden.EmployeeID
End Sub
End Class
```

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.50: Ejecución del formulario Filtro de Ordenes x Cliente con LINQ a SQL



	EmployeeID	OrderDate	OrderID
▶	5	04/07/1996	10248
	6	06/08/1996	10274
	2	02/09/1996	10295
	2	11/11/1997	10737
	3	12/11/1997	10739

4.3 LINQ a Entidades

ADO.NET Entity Framework permite a los desarrolladores crear aplicaciones de acceso a datos programando con un modelo de aplicaciones conceptuales en lugar de programar directamente con un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y el mantenimiento necesarios para las aplicaciones orientadas a datos.

LINQ a Entidades permite escribir consultas contra el modelo conceptual de Entity Framework mediante un lenguaje como Visual Basic. Las consultas con Entity Framework se representan mediante consultas de árboles de comandos, que se ejecutan en el contexto del objeto.

LINQ a Entidades convierte las consultas de LINQ en consultas de árboles de comandos, ejecuta las consultas en Entity Framework y devuelve objetos que se pueden usar tanto en Entity Framework como en LINQ.

Los pasos para crear y ejecutar una consulta de LINQ a Entidades son:

1. Crear una instancia de `ObjectQuery` en `ObjectContext`.
2. Crear una consulta de LINQ a Entidades en con la instancia de `ObjectQuery`.
3. Convertir los operadores y expresiones de consulta estándar de LINQ en árboles de comandos.
4. Ejecutar la consulta, con representación de un árbol de comandos, en el origen de datos. Las excepciones producidas en el origen de datos durante la ejecución se pasan directamente al cliente.
5. Devolver los resultados de la consulta al cliente (Materialización).

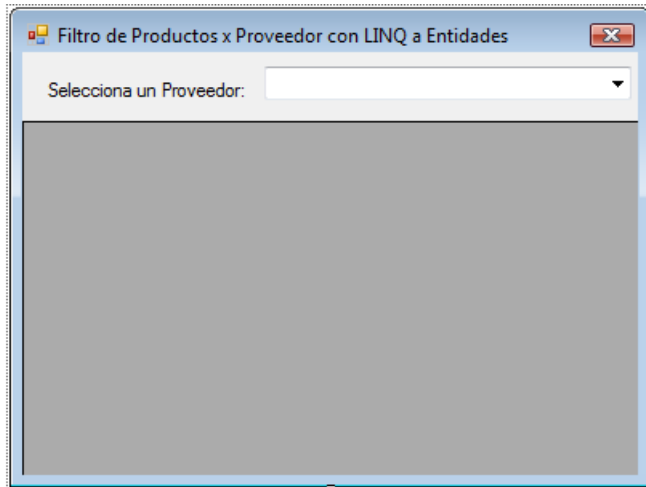
Demo 50

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo50.
- Cambiar de nombre al formulario de form1.vb a frmProductosx Proveedor.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmProductosxProveedor FixedSingle False False 400,300 CenterScreen Filtro de Productos x Proveedor con LINQ a Entidades
Label1	Name AutoSize Location Text	lblProveedor True 13, 17 Selecciona un Proveedor:
ComboBox1	Name Location Size	cboProveedor 151, 9 229, 21
DataGridView1	Name AutoSizeColumnsMode Location ReadOnly SelectionMode Size	dgvProducto AllCells 2, 43 True FullRowSelect 392, 225

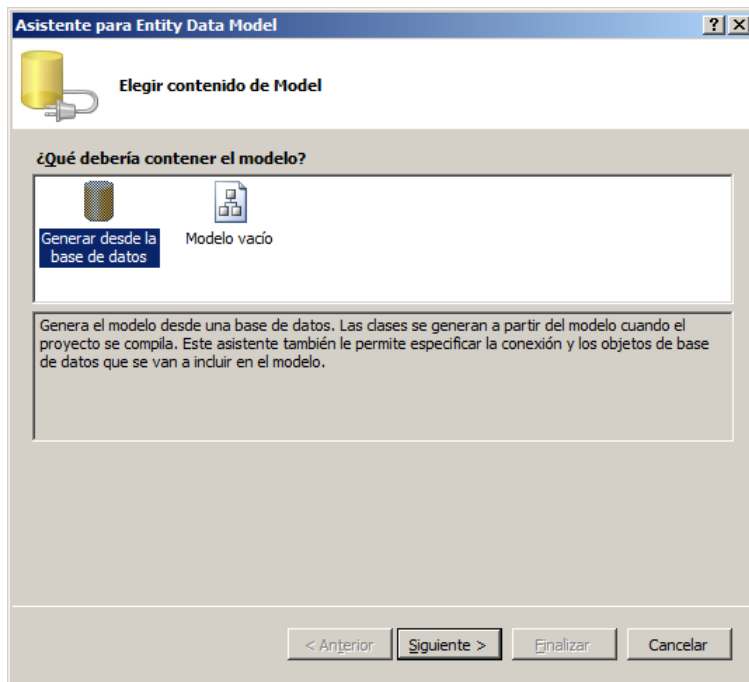
- El diseño del formulario debe quedar similar al gráfico 3.51:

Gráfico 3.51: Diseño del formulario Filtro de Productos x Proveedor con LINQ a Entidades



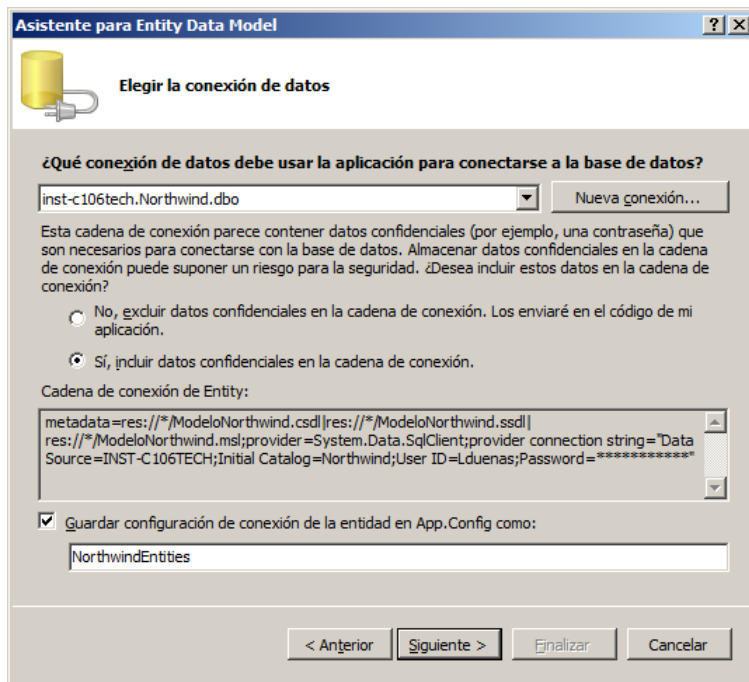
- Agregar un Modelo de Datos de Entidades: del menú "Project", seleccionar "Add New Item" y luego en la sección "Data" elegir "ADO NET Entity Data Model".
- Ingresar como nombre al Modelo: ModeloNorthwind.edmx y "OK".
- Aparecerá el diálogo para seleccionar el contenido del modelo, tal como se muestra en la figura 3.52.

Gráfico 3.52: Asistente Entity Data Model – Elegir contenido



- Seleccionar "Generar desde la Base de Datos" y botón "Siguiente".
- Se mostrará el diálogo para seleccionar o crear la conexión a la base de datos.

Gráfico 3.53: Asistente Entity Data Model – Elegir conexión

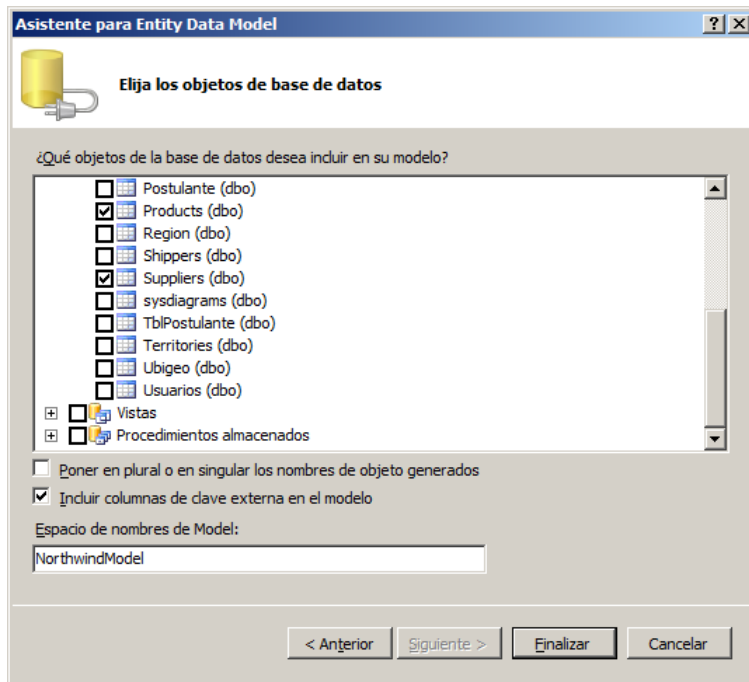


- Ingresar como nombre para la configuración de la entidad: "Northwind Entities" y botón "Siguiente".

Nota: Marcar la opción "Si, incluir datos confidenciales en la cadena de conexión" si la seguridad es Mixta.

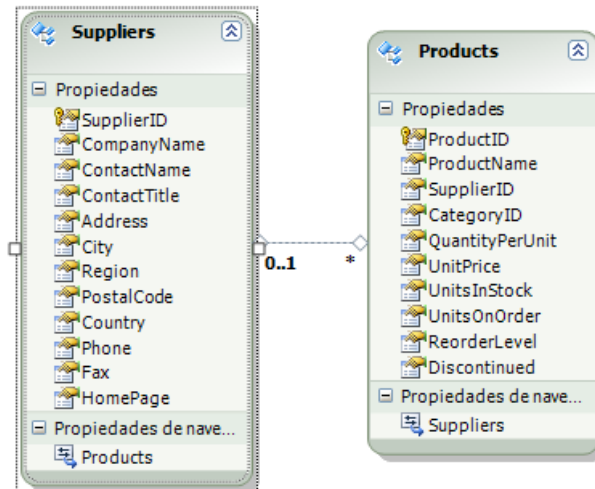
- Apacera el diálogo de elegir objetos de bases de datos como se muestra en la siguiente figura.

Gráfico 3.54: Asistente Entity Data Model – Elegir objetos de base de datos



- Expandir tablas y seleccionar las tablas “Products” y “Suppliers”, escribir como espacios de nombre: “NorthwindModel” y “Finalizar”.
- Se creará un modelo de datos similar al mostrado en la siguiente figura.

Gráfico 3.55: Diseñador con el Modelo de Datos de Entidades



- Escribir el siguiente código en el formulario:

```
Public Class frmProductosxProveedor
    Private oNW As New NorthwindEntities

    Private Sub ListarProveedores(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        With cboProveedor
            .DataSource = oNW.Suppliers
            .DisplayMember = "CompanyName"
            .ValueMember = "SupplierID"
        End With
        FiltrarProductosxProveedor(Nothing, Nothing)
    End Sub

    Private Sub FiltrarProductosxProveedor(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles _
        cboProveedor.SelectionChangeCommitted
        Dim IdProveedor As Integer = cboProveedor.SelectedValue
        dgvProducto.DataSource = From oProducto In oNW.Products _
            Where oProducto.Suppliers.SupplierID = IdProveedor _
            Select oProducto.ProductID, oProducto.ProductName, _
            oProducto.UnitPrice
    End Sub
```

End Class

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 3.56: Ejecución del formulario Filtro de Productos x Proveedor con LINQ a Entidades

	ProductID	ProductName	UnitPrice
▶	1	Chai	18.0000
	2	Chang	19.0000
	3	Aniseed Syrup	10.0000

Preguntas de Repaso

1. Qué es ADO .NET?
2. Cuáles son los 2 componentes principales de ADO .NET?
3. Menciona los 2 proveedores nativos de datos de ADO .NET.
4. Menciona los 2 proveedores generales de datos de ADO .NET.
5. Menciona las clases que debe implementar todo proveedor de datos?
6. Porqué es importante la cadena de conexión y con qué propiedad de la clase Connection se define?
7. Cómo sería la cadena de conexión a una base de datos de SQL Server que tiene seguridad de Windows y hay una sola instancia instalada.
8. Qué proveedor de datos se usa para conectarse a una base de datos de MS Access, un archivo de MS Excel o un archivo DBF?
9. En qué se diferencia la cadena de conexión a una base de datos de MS Access de la de un archivo Excel?
10. Cómo se puede obtener las tablas de una base de datos de MS Access o las hojas de un archivo de MS Excel?
- 11.Cuál es la forma más eficiente de copiar gran cantidad de datos hacia SQL Server?
12. Con qué clase de .NET se implementa la copia masiva hacia una base de datos de MS SQL Server?
13. Con qué método de la clase Command se ejecutan comandos de selección de un valor?

14. Con qué método de la clase Command se ejecutan comandos de selección de una fila?
15. Con qué método de la clase Command se ejecutan comandos de selección de varias filas?
16. Con qué método de la clase Command se ejecutan comandos de selección de varios conjuntos de filas?
17. Cómo se llama la clase que permite almacenar una sola fila a la vez?
- 18.Cuál es la diferencia entre un DataReader y un DataSet?
19. Cómo se controlan los nulos mediante el DataReader?
20. Con qué método de la clase DataReader se puede leer el siguiente conjunto de registros en caso de ser varios Select?
21. Con qué método de la clase Command se ejecuta un comando que inserta, actualiza o elimina registros?
22. Qué es un DataSet y para que sirve?
23. Cuáles son los elementos de un DataSet?
24. Para que sirve el DataAdapter?
25. Con qué método del DataAdapter se crea una tabla en el DataSet desde un origen de datos?
26. Cómo se crea una vista o DataView?
27. Con qué propiedad del DataView se filtran registros?
28. Con qué propiedad del DataView se ordenan registros?
29. Con qué método del DataView se busca un registro por una clave?

30. Qué debe hacerse al DataView antes de realizar una búsqueda, sino se genera una excepción?
31. De qué formas se puede eliminar registros de un DataTable?
32. Cómo se pueden obtener los cambios realizados en un DataTable o DataSet antes de enviarlos hacia la base de datos?
33. Con qué método del DataAdapter se envían los cambios de regreso a la base de datos?
- 34.Cuál es la ventaja de trabajar desconectado usando Listas de Objetos en vez del DataSet?
35. Cómo se filtran objetos de una lista de objetos?
36. Cómo se ordenan objetos de una lista de objetos?
37. Cómo se buscan datos en una lista de objetos?
38. Menciona 3 métodos extendidos que permitan realizar cálculos en listas de objetos.
39. Para qué sirve LINQ y Cuántos tipos de LINQ existen?
40. De qué formas podemos acceder a bases de datos con LINQ?
41. Qué clases implementan LINQ a DataSets?
42. De qué formas se puede escribir la sintáxis de LINQ a DataSets?
43. Qué herramienta del Visual Studio se usa para implementar muchas características de LINQ a SQL?
44. Qué tipo de operaciones se pueden realizar con LINQ a SQL?

- 45. Qué es ADO .NET Entity Framework?
- 46. A qué convierte LINQ a Entidades las consultas de LINQ.
- 47. Cuáles son los pasos para crear y ejecutar una consulta de LINQ a Entidades?.

Capítulo 4: Desarrollando Aplicaciones Windows Forms

Las aplicaciones Windows o de escritorio son aquellas basadas en ventanas y mensajes de Windows, también conocidas como Aplicaciones Cliente Inteligentes.

En .NET Framework para implementar este tipo de aplicaciones existen el espacio de nombres System.Windows.Forms el cual tiene clases como Form que representa al formulario y otras como Label, TextBox, Button, etc., que representan a los controles Windows Forms.

Con los formularios Windows Forms podemos crear clientes inteligentes que aprovechen las ventajas de Windows tales como los gráficos, la seguridad, manejo de entrada y salida, etc.

Aunque hasta el momento todos los ejemplos del libro han sido aplicaciones para Windows, en este capítulo veremos formalmente como crear aplicaciones que usen el formulario y los controles Windows Forms en la primera parte.

En la segunda parte mejoraremos la interface de usuario creando formularios MDIs, agregando menús, usando diálogos y agregando barras de herramientas para hacer más fácil el acceso a la aplicación a los usuarios.

En la tercera parte trataremos con mayor detalle el control DataGridView que es principal control Windows Forms usado para mostrar datos en una aplicación, para lo cual se verá como personalizar columnas, cabeceras, celdas y filas; también aprenderemos como paginar datos cuando existen gran cantidad de registros y finalmente como graficar dentro de la grilla.

En la cuarta parte crearemos librerías de controles para Windows, incluyendo los 3 tipos de controles: extendidos, de usuario y personalizados además veremos como agregar características para usarse en Visual Studio.

Finalmente, veremos como crear reportes, informes e impresiones usando diferentes formas como: usando el objeto PrintDocument y los diálogos de

impresión, también usando el ReportViewer para crear reporte de datos y usando Office para crear informes en MS Word y exportar datos a Excel y crear gráficos de datos.

1. Trabajando con el Formulario y los Controles Windows

El Formulario es el contenedor principal de las aplicaciones Windows Forms y sobre este se encuentran los controles. En esta primera parte veremos como configurar propiedades, trabajar con métodos y programar eventos del formulario (clase Form) y los controles Windows estándares.

1.1. Trabajando con el Formulario Windows

Un formulario Form es una representación de cualquier ventana mostrada en su aplicación. La clase Form se puede utilizar para crear ventanas estándar, de herramientas, sin bordes y flotantes. También puede utilizar la clase Form para crear las ventanas modales como un cuadro de diálogo. Un tipo especial de formulario, el formulario de interfaz de múltiples documentos (MDI), puede contener otros formularios denominados formularios MDI secundarios. Los formularios MDI se crean estableciendo la propiedad IsMdiContainer en true. Los formularios MDI secundarios se crean estableciendo la propiedad MdiParent en el formulario MDI principal que contendrá el formulario secundario.

Utilizando las propiedades disponibles en la clase Form, puede determinar el aspecto, tamaño, color y las funciones de administración de la ventana o cuadro de diálogo que está creando. La propiedad Text le permite especificar el título que aparecerá en la barra de título de la ventana. Las propiedades Size y DesktopLocation le permiten definir el tamaño y la ubicación de la ventana cuando se muestra en la pantalla. Puede utilizar la propiedad de color ForeColor para cambiar el color predeterminado de primer plano de todos los controles incluidos en el formulario. Las propiedades FormBorderStyle, MinimizeBox y MaximizeBox le permiten controlar si se puede minimizar o maximizar el formulario, o si se puede cambiar el tamaño en tiempo de ejecución.

Además de las propiedades, puede utilizar los métodos de la clase para manipular un formulario. Por ejemplo, puede utilizar el método ShowDialog para mostrar un formulario como un cuadro de diálogo modal. El método SetDesktopLocation se puede usar para situar el formulario en el escritorio.

Los eventos de la clase Form le permiten responder a las acciones realizadas en el formulario. Puede utilizar el evento Activated para realizar operaciones como actualizar los datos mostrados en los controles del formulario cuando se activa el formulario.

Puede utilizar un formulario como la clase de inicio de su aplicación colocando un método llamado Main en la clase. En el método Main, agregue el código necesario para crear y mostrar el formulario. También será necesario agregar el atributo STAThread al método Main para que se ejecute el formulario. Cuando se cierra el formulario de inicio, también se cierra la aplicación.

Para obtener mas información sobre los formularios de Windows Forms ver la referencia 23 al final del libro.

A continuación presentaremos un ejemplo de cómo crear un formulario No rectangular sino en forma de elipse, usando graficos y un temporizador para desaparecer progresivamente el formulario al dar clic.

Demo 51

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo51.
- Cambiar de nombre al formulario de form1.vb a frmGraficoElipse.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmGraficoElipse
	Cursor	Hand
	FormBorderStyle	None
	Text	
	WindowState	Maximized

- El diseño del formulario debe quedar similar al gráfico 4.1:

Gráfico 4.1: Diseño del formulario Grafico de Elipse



- Escribir el siguiente código en el formulario:

```
Imports System.Drawing.Drawing2D 'LinearGradientMode, GraphicsPath
```

```
Public Class frmGraficoElipse
```

```
    Private WithEvents tmrGrafico As New Timer
```

```
    Private N As Integer
```

```
    Private Sub GraficarFormulario(ByVal sender As Object, ByVal e As _  
        System.Windows.Forms.PaintEventArgs) Handles Me.Paint
```

```
        Dim X As Integer = Me.Width / 2
```

```
        Dim Y As Integer = Me.Height / 2
```

```
        Dim rec As New Rectangle(X - 400, Y - 200, 800, 400)
```

```
        Dim deg As New LinearGradientBrush(rec, Color.Aqua, Color.Blue, _  
            LinearGradientMode.BackwardDiagonal)
```

```
        e.Graphics.FillEllipse(deg, rec)
```

```
        e.Graphics.DrawString("Visual Basic 2010", New Font("Arial", 50), _  
            Brushes.White, X - 270, Y - 100)
```

```
        e.Graphics.DrawString("Por Luis Dueñas H", New Font("Arial", 40), _  
            Brushes.Yellow, X - 220, Y + 20)
```

```
        Dim gp As New GraphicsPath
```

```
        gp.AddEllipse(rec)
```

```
        Me.Region = New Region(gp)
```

```
    End Sub
```

```
    Private Sub HabilitarTimer(ByVal sender As Object, ByVal e As _
```

```
System.EventArgs) Handles Me.Click
    With tmrGrafico
        .Interval = 10
        .Enabled = True
        .Start()
    End With
End Sub

Private Sub DesvanecerFormulario(ByVal sender As Object, ByVal e As _
System.EventArgs) Handles tmrGrafico.Tick
    Me.Opacity = (100 - N) / 100
    N += 1
    If N = 100 Then Me.Close()
End Sub
End Class
```

Nota: En el código anterior en el evento Paint del formulario se crea el grafico y luego se configura la propiedad Region del formulario en una nueva región la cual contiene la elipse.

Importante: No solo se puede cambiar la apariencia del formulario sino de cualquier control Windows Forms ya que todos heredan de control que tiene la propiedad región que permite cambiar la forma.

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.2: Ejecución del formulario Grafico de Elipse



1.2. Usando Controles Básicos

La mayoría de aplicaciones hace uso de los controles básicos de Windows tales como las etiquetas (clase Label), los cuadros de texto (clase TextBox) y los botones de comandos (clase Button). En este tema veremos como usar dichos controles para crear aplicaciones básicas.

En el siguiente ejemplo se muestra como crear un dialogo de Login que permita al usuario registrarse verificando su usuario y su clave desde un archivo de texto.

Demo 52

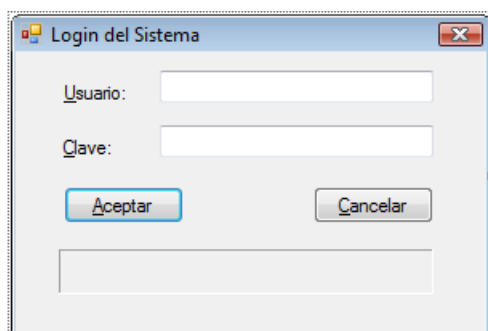
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo52.
- Cambiar de nombre al formulario de form1.vb a frmLogin.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name AcceptButton CancelButton FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmLogin btnAceptar btnCancelar FixedSingle False False 300, 200 CenterScreen Login del Sistema
Label1	Name AutoSize Location Text	lblUsuario True 27, 19 &Usuario:
TextBox1	Name Location MaxLength Size Tag	txtUsuario 90,12 20 171,20 Ingrese el Usuario

Label2	Name AutoSize Location Text	lblClave True 27, 53 &Clave:
TextBox2	Name Location MaxLength Size Tag UseSystemPasswordChar	txtClave 90,46 10 171,20 Ingresa la clave True
Button1	Name Cursor Enabled Location Size Text	btnAceptar Hand False 30, 84 75, 23 &Aceptar
Button2	Name Cursor Location Size Text	btnCancelar Hand 186, 84 75, 23 &Cancelar
Label3	Name AutoSize BorderStyle Location Size TextAlign	lblMensaje False Fixed3D 27, 123 234,28 MiddleCenter

- El diseño del formulario debe quedar similar al gráfico 4.1:

Gráfico 4.3: Diseño del formulario Login



- Abrir el bloc de notas y crear un archivo de texto con los nombres de los usuarios y sus claves, por ejemplo ingresar:

Lduenas,1000
Trojas,2000
Mfuentes,3000

- Escribir el siguiente código en el formulario:

Imports System.IO

Public Class frmLogin

Private contador As Integer

Private Sub HabilitarBotonAceptar(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles _
txtUsuario.TextChanged, txtClave.TextChanged
 btnAceptar.Enabled = (txtUsuario.Text<>"" AndAlso txtClave.Text<>"")
End Sub

Private Sub MostrarMensaje(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles _
txtUsuario.Enter, txtClave.Enter
 lblMensaje.Text = sender.Tag
End Sub

Private Sub BorrarMensaje(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles _
txtUsuario.Leave, txtClave.Leave
 lblMensaje.Text = ""
End Sub

Private Sub Aceptar(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAceptar.Click
 Dim Archivo As String = _
 "C:\Lduenas\NET\LibroVB2010\Usuarios.txt"
 If File.Exists(Archivo) Then
 Dim Encontro As Boolean = False
 Using fs As New FileStream(Archivo, _
 FileMode.Open, FileAccess.Read, FileShare.Read)
 Using sr As New StreamReader(fs)

```
Dim Usuario() As String
Do While Not sr.EndOfStream
    Usuario = sr.ReadLine.Split(",")
    If Usuario.Length = 2 Then
        If Usuario(0) = txtUsuario.Text _
            AndAlso Usuario(1) = txtClave.Text Then
            Encontro = True
            Exit Do
        End If
    End If
Loop
contador += 1
End Using
End Using
If Encontro Then
    MessageBox.Show("Login valido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    Me.Close()
Else
    MessageBox.Show("Login Invalido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
End If
If contador = 3 Then
    MessageBox.Show("Acceso Denegado", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    Me.Close()
End If
End If
End Sub

Private Sub Cancelar(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCancelar.Click
    Me.Close()
End Sub
End Class
```

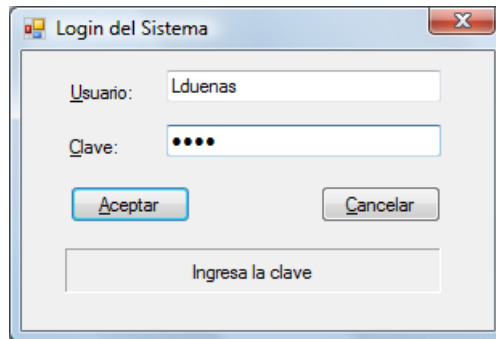
Nota: En el código anterior la ruta del archivo Usuarios.txt tiene que modificarse de acuerdo a la ubicación donde se encuentre su archivo. Además la propiedad AcceptButton del formulario permite que al dar Enter

sobre cualquier Texto del formulario se dispare el procedimiento asociado al botón Aceptar y la propiedad CancelButton permite que al pulsar Escape se dispare el procedimiento asociado al botón Cancelar.

Importante: El & antes de un Texto funciona como ShortKey sobre un control debido a que la propiedad UseMnemonic es true, si se desea que el & funcione como un Literal configurar la propiedad UseMnemonic en False. Además para configurar un Texto como Password hay 2 formas: establecer la propiedad PasswordChar en un carácter o configurar la propiedad UseSystemPasswordChar en true.

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.4: Ejecución del formulario Login



1.3. Usando Controles de Listas

Los controles de listas en Windows Forms son el cuadro de lista (clase `ListBox`), cuadro combinado (clase `ComboBox`) y cuadro de lista con casillas (clase `CheckedListBox`).

Todos tienen una colección de elementos (propiedad `Items`), la cual tiene métodos comunes como `Add` para agregar un elemento, `AddRange` para agregar una lista de elementos, `Remove` y `RemoveAt` para eliminar un elemento por su nombre o índice y `Clear` para eliminar todos los elementos de la lista.

También tienen propiedades comunes como `SelectedIndex`, `SelectedItem` y eventos comunes como `SelectedIndexChanged` y `SelectedValueChanged`, pero el `ComboBox` además posee el evento `SelectionChangedCommitted` que ocurre cuando un usuario selecciona un elemento de la lista desplegable.

Para llenar los elementos de una lista lo más recomendable es cargar todos los elementos a la vez usando el método `AddRange` de la propiedad `Items`, pero sino se puede usar la forma clásica que es usando un bucle y llenando cada elemento usando el método `Add` de la propiedad `Items`, pero antes de iniciar el bucle usar el método `BeginUpdate` para congelar cambios en pantalla y trabajar solo en memoria y al final después del bucle usar el método `EndUpdate` para actualizar la pantalla. Esto evita el parpadeo que ocurre al cargar las listas.

A continuación mostramos un ejemplo de cómo cargar una lista con las culturas o países y sus respectivos idiomas usando la clase `CultureInfo` ubicada en el espacio de nombres `System.Globalization`; el usuario elegirá a que grupo pertenecerá: A o B y podrá mover elementos entre listas mostrando siempre el total de elementos de las listas.

Demo 53

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo53.

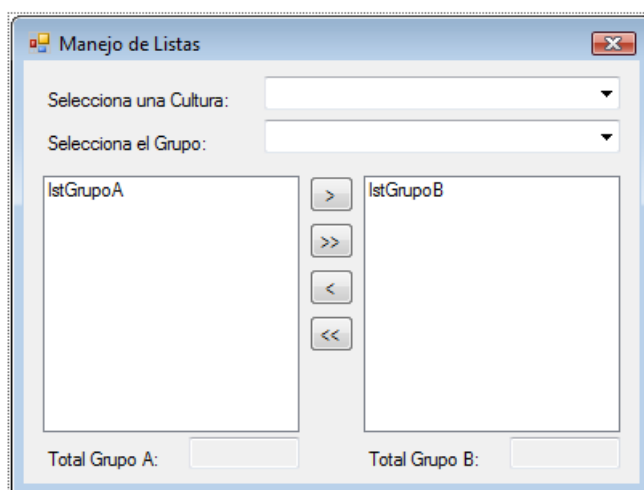
- Cambiar de nombre al formulario de form1.vb a frmListas.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmListas FixedSingle False False 400, 300 CenterScreen Manejo de Listas
Label1	Name AutoSize Location Text	lblUsuario True 27, 19 &Usuario:
TextBox1	Name Location MaxLength Size Tag	txtUsuario 90,12 20 171,20 Ingresa el Usuario
Label2	Name AutoSize Location Text	lblClave True 27, 53 &Clave:
TextBox2	Name Location MaxLength Size Tag UseSystemPasswordChar	txtClave 90,46 10 171,20 Ingresa la clave True
Button1	Name Cursor Enabled Location Size Text	btnAceptar Hand False 30, 84 75, 23 &Aceptar
Button2	Name Cursor Location Size	btnCancelar Hand 186, 84 75, 23

	Text	&Cancelar
Label3	Name	lblMensaje
	AutoSize	False
	BorderStyle	Fixed3D
	Location	27, 123
	Size	234,28
	TextAlign	MiddleCenter

- El diseño del formulario debe quedar similar al gráfico 4.5:

Gráfico 4.5: Diseño del formulario Manejo de Listas



- Escribir el siguiente código en el formulario:

```
Imports System.Globalization 'CultureInfo
```

```
Public Class frmListas
```

```
Private Sub LlenarListas(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    'Llenar la Lista de Culturas
```

```
    With cboCultura
```

```
        .Items.AddRange(CultureInfo.GetCultures _
            (CultureTypes.AllCultures))
```

```
        .Items.Insert(0, "Seleccione")
```

```
        .SelectedIndex = 0
```

```
End With
'Llenar la Lista de Grupos
With cboGrupo
    .BeginUpdate()
    .Items.Add("Seleccione")
    .Items.Add("Grupo A")
    .Items.Add("Grupo B")
    .EndUpdate()
    .SelectedIndex = 0
End With
End Sub
```

```
Private Sub AgregarElemento(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles _
    cboCultura.SelectionChangeCommitted,
    cboGrupo.SelectionChangeCommitted
    If cboCultura.SelectedIndex > 0 Then
        If cboGrupo.SelectedIndex = 1 Then
            If lstGrupoA.Items.IndexOf(cboCultura.SelectedItem) = -1 Then
                If lstGrupoB.Items.IndexOf(cboCultura.SelectedItem) = -1 Then
                    lstGrupoA.Items.Add(cboCultura.SelectedItem)
                Else
                    MessageBox.Show("El elemento ya fue agregado al Grupo B", _
                        "Aviso", MessageBoxButtons.OK, _
                        MessageBoxIcon.Exclamation)
                End If
                txtTotalGrupoA.Text = lstGrupoA.Items.Count
            Else
                MessageBox.Show("El elemento ya fue agregado al Grupo A", _
                    "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            End If
        ElseIf cboGrupo.SelectedIndex = 2 Then
            If lstGrupoB.Items.IndexOf(cboCultura.SelectedItem) = -1 Then
                If lstGrupoA.Items.IndexOf(cboCultura.SelectedItem) = -1 Then
                    lstGrupoB.Items.Add(cboCultura.SelectedItem)
                Else
                    MessageBox.Show("El elemento ya fue agregado al Grupo A", _
                        "Aviso", MessageBoxButtons.OK, _
                        MessageBoxIcon.Exclamation)
                End If
            End If
        End If
    End Sub
```



```
        txtTotalGrupoB.Text = lstGrupoB.Items.Count
    Else
        MessageBox.Show("El elemento ya fue agregado al Grupo B", _
            "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
End If
End If
End Sub

Private Sub MostrarTotalListas()
    txtTotalGrupoA.Text = lstGrupoA.Items.Count
    txtTotalGrupoB.Text = lstGrupoB.Items.Count
End Sub

Private Sub MoverUnoGrupoB(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnMoverUnoGrupoB.Click
    If lstGrupoA.Items.Count > 0 Then
        If lstGrupoA.SelectedIndex > -1 Then
            lstGrupoB.Items.Add(lstGrupoA.SelectedItem)
            lstGrupoA.Items.Remove(lstGrupoA.SelectedItem)
            MostrarTotalListas()
        Else
            MessageBox.Show("Selecciona el elemento a mover en el Grupo A", _
                "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
        End If
    Else
        MessageBox.Show("No hay elementos que mover en el Grupo A", _
            "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
End Sub

Private Sub MoverTodoGrupoB(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnMoverTodoGrupoB.Click
    If lstGrupoA.Items.Count > 0 Then
        lstGrupoB.Items.AddRange(lstGrupoA.Items)
        lstGrupoA.Items.Clear()
        MostrarTotalListas()
    Else
        MessageBox.Show("No hay elementos que mover en el Grupo A", _
            "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
End Sub
```

```
End If
End Sub

Private Sub MoverUnoGrupoA(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnMoverUnoGrupoA.Click
    If lstGrupoB.Items.Count > 0 Then
        If lstGrupoB.SelectedIndex > -1 Then
            lstGrupoA.Items.Add(lstGrupoB.SelectedItem)
            lstGrupoB.Items.Remove(lstGrupoB.SelectedItem)
            MostrarTotalListas()
        Else
            MessageBox.Show("Selecciona el elemento a mover en el Grupo B", _
"Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
        End If
    Else
        MessageBox.Show("No hay elementos que mover en el Grupo B", _
"Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
End Sub

Private Sub MoverTodoGrupoA(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnMoverTodoGrupoA.Click
    If lstGrupoB.Items.Count > 0 Then
        lstGrupoA.Items.AddRange(lstGrupoB.Items)
        lstGrupoB.Items.Clear()
        MostrarTotalListas()
    Else
        MessageBox.Show("No hay elementos que mover en el Grupo B", _
"Aviso", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
End Sub
End Class
```

Nota: En el código anterior para mover todos los elementos de una lista no se ha usado un bucle sino mediante el método AddRange de la propiedad Items se ha agregado todos los elementos de la otra lista (mediante su propiedad Items).

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.6: Ejecución del formulario Manejo de Listas

The screenshot shows a Windows-style application window titled "Manejo de Listas". Inside the window, there are two dropdown menus at the top. The first is labeled "Selecciona una Cultura:" and has "Español" selected. The second is labeled "Selecciona el Grupo:" and has "Grupo B" selected. Below these are two list boxes. The left list box contains "Arabe", "Búlgaro", and "Checo", with "Arabe" selected. The right list box contains "Danés", "Inglés", and "Español". Between the two list boxes are four buttons: ">", ">>", "<", and "<<". At the bottom of the window, there are two labels with text boxes: "Total Grupo A:" followed by a text box containing "3", and "Total Grupo B:" followed by a text box containing "3".

1.4. Usando Controles de Vistas

En Windows Forms existen 2 controles de vistas que son la vista de árbol (clase `TreeView`) y la vista de lista (clase `ListView`). El `TreeView` muestra información jerárquica y su propiedad más importante es `Nodes` que es una colección de tipo `TreeNode`s y como toda colección de .NET Framework tiene los métodos `Add`, `AddRange`, `Remove`, `RemoveAt` y `Clear`.

El `TreeView` tiene una propiedad `ImageList` que permite asociar a una lista de imágenes, de tal forma de ver una imagen en cada nodo añadido al árbol, además el evento mas importante de este es el `AfterSelecte` que ocurre después de selecciona un nodo, éste evento posee un parámetro de tipo `TreeViewEventArgs` que contiene información del nodo, como por ejemplo el nivel, el texto que se muestra, etc.

Por su parte, el `ListView` es un control que presenta 5 tipos de vistas: iconos grandes, iconos pequeños, lista, detalles y mosaicos, las cuales se controlan mediante la propiedad `View`. Su propiedad más importante es `Items` que es una colección de tipo `ListViewItem` y al igual que el `TreeView` tiene los métodos `Add`, `Add Range`, `Remove`, `RemoveAt` y `Clear`.

El `ListView` también tiene propiedades que indican que imágenes van a visualizarse en los elementos, para iconos grandes se usa la propiedad `LargeImageList` y para iconos pequeños o la vista detalles se usa la propiedad `SmallImageList`.

Para presentar mejor la lista de elementos en el `ListView` se puede usar las propiedades `GridLines`, `HotTracking`, `HoverSelection` y `FullRowSelect`. En cuanto a eventos uno de los más usados es el evento `ColumnClick` que ocurre al dar clic a las cabeceras del `ListView`.

A continuación presentamos un ejemplo muy interesante que simula la funcionalidad de un Explorador de Windows, inicialmente se carga las unidades y se lista los directorios raíces de éstas, cuando el usuario seleccione un directorio en el `TreeView` se cargarán sus subdirectorios y si tuviera archivos estos se mostrarán el `ListView` en la vista detalle con 5 columnas con el nombre del archivo, extensión, tamaño, fecha de creación y fecha de modificación.

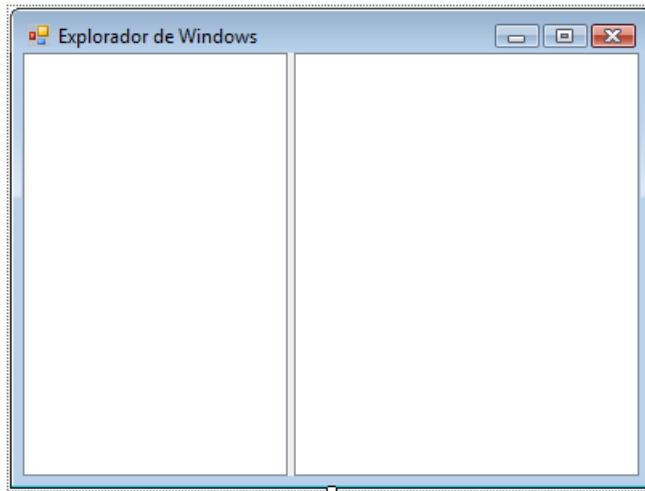
Demo 54

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo54.
- Cambiar de nombre al formulario de form1.vb a frmExplorador.vb
- Configurar el formulario y los controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmExplorador FixedSingle False False 400,300 CenterScreen Explorador de Windows
SplitContainer1	Name Dock	scExplorador Fill
TreeView1 (izquierda)	Name Dock	twwDirectorio Fill
ListView1 (derecha)	Name Dock FullRowSelect GridLines HotTracking	lvwArchivo Fill True True True

- El diseño del formulario debe quedar similar al gráfico 4.7:

Gráfico 4.7: Diseño del formulario Explorador de Windows



- Escribir el siguiente código en el formulario:

```
Imports System.IO
```

```
Public Class frmExplorador  
    Private ils As New ImageList
```

```
    Private Sub ListarUnidades(ByVal sender As System.Object, _  
        ByVal e As System.EventArgs) Handles MyBase.Load  
        Dim Ruta As String = "C:\Lduenas\NET\LibroVB2010\Imagenes\  
        ils.Images.Add("PC", Image.FromFile(Ruta + "PC.ico"))  
        ils.Images.Add("Drive", Image.FromFile(Ruta + "Drive.ico"))  
        ils.Images.Add("Folder", Image.FromFile(Ruta + "Folder.ico"))  
        ils.Images.Add("File", Image.FromFile(Ruta + "File.ico"))  
        twwDirectorio.ImageList = ils  
        With lvwArchivo  
            .SmallImageList = ils  
            .Columns.Add("Nombre Archivo", 300, HorizontalAlignment.Left)  
            .Columns.Add("Extensión", 100, HorizontalAlignment.Left)  
            .Columns.Add("Tamaño", 100, HorizontalAlignment.Left)  
            .Columns.Add("Fecha Creación", 150, HorizontalAlignment.Left)  
            .Columns.Add("Fecha Modificación", 150, HorizontalAlignment.Left)  
            .View = View.Details  
        End With  
        Dim nodoPC As TreeNode = twwDirectorio.Nodes.Add _
```

```
("PC", Environment.MachineName, 0, 0)
Dim Unidades() As DriveInfo = DriveInfo.GetDrives
Dim nodoUnidad As TreeNode
For Each Unidad In Unidades
    nodoUnidad = nodoPC.Nodes.Add(Unidad.Name, _
    Unidad.Name.Replace("\", ""), 1, 1)
    ListarDirectorios(nodoUnidad, Unidad.Name)
Next
twwDirectorio.ExpandAll()
End Sub

Private Sub ListarDirectorios(ByVal nodo As TreeNode, _
ByVal Ruta As String)
    Try
        Dim Directorios() As String = Directory.GetDirectories(Ruta)
        For Each Directorio In Directorios
            nodo.Nodes.Add(Path.GetFileName(Directorio), _
            Path.GetFileName(Directorio), 2, 2)
            ListarArchivos(Directorio)
        Next
    Catch ex As Exception
    End Try
End Sub

Private Sub ListarArchivos(ByVal Ruta As String)
    Try
        Dim Archivos() As String = Directory.GetFiles(Ruta)
        lvwArchivo.Items.Clear()
        Dim fila As ListViewItem
        Dim fi As FileInfo
        For Each Archivo In Archivos
            fi = New FileInfo(Archivo)
            fila = lvwArchivo.Items.Add _
            (Path.GetFileNameWithoutExtension(Archivo), 3)
            fila.SubItems.Add(Path.GetExtension(Archivo))
            fila.SubItems.Add(fi.Length)
            fila.SubItems.Add(fi.CreationTime.ToShortDateString)
            fila.SubItems.Add(fi.LastWriteTime.ToShortDateString)
        Next
    Catch ex As Exception
```

```
End Try
End Sub

Private Sub ListarSubdirectorios(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.TreeViewEventArgs) _
Handles tvwDirectorio.AfterSelect
    If e.Node.Level > 1 Then
        e.Node.Nodes.Clear()
        Dim Ruta As String = ObtenerRuta(e.Node)
        ListarDirectorios(e.Node, Ruta)
        ListarArchivos(Ruta)
    Else
        lvwArchivo.Items.Clear()
    End If
End Sub

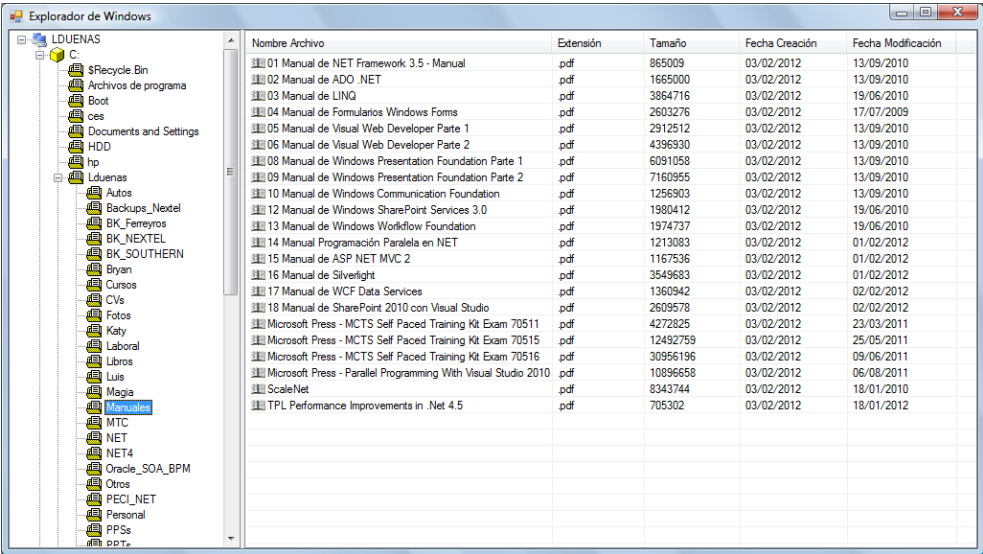
Private Function ObtenerRuta(ByVal e As TreeNode) As String
    Dim S As String = e.Text
    If e.Level > 1 Then
        S = ObtenerRuta(e.Parent) + "\" + S
    End If
    Return (S)
End Function
End Class
```

Nota: En el código anterior se crea dinámicamente el ImageList con 4 imágenes, las 3 primeras son para el TreeView y la cuarta para el ListView, deberás cambiar la ruta donde se encuentran las imágenes de acuerdo a donde se ubiquen en tu PC.

También hay que aclarar que siempre se verifica que el nivel sea mayor a 1 ya que el nivel 0 es el nodo raíz con el nombre de la PC, el nivel 1 son las unidades y a partir del nodo 2 empiezan los directorios.

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.8: Ejecución del formulario Explorador de Windows



2. Creando Formularios MDIs, Menús, Diálogos y Barras

En esta parte mejoraremos el diseño de la interface de usuario creando formularios MDIs, agregando menús: principal y contextual, usando diálogos comunes de Windows como los diálogos de archivos: abrir y guardar, diálogo de color y diálogo de fuente y finalmente, agregando barras de herramientas y barras de estado.

2.1. Creando Formularios MDIs

Un formulario MDI es una Interface de Múltiples de Documentos, es decir una ventana que puede contener ventanas hijas, las cuales pueden organizarse en cascada, mosaico horizontal, mosaico vertical o como iconos.

La mayoría de Software usan MDIs o Interface de Múltiples Documentos para trabajar, como por ejemplo el Visual Studio, el SQL Server Managment Studio, o los Sistemas de Información de muchas Empresas.

En Windows Forms para crear un formulario MDI solo basta configurar la propiedad `IsMdiContainer` del formulario en `True`, esto hace que el formulario tome un fondo gris, el cual no es un simple fondo sino en realidad se agrega un control de tipo `MdiClient` que es un contenedor.

Importante: Si se desea cambiar el fondo de un formulario MDI no se debe hacer configurando la propiedad `BackColor` del formulario sino la del control `MdiClient` creado, por ejemplo si no hay controles en el MDI entonces para dar un fondo rojo se debe escribir lo siguiente: `Me.Controls(0).BackColor=Color.Red`.

Para crear un formulario MDI hijo, es necesario configurar la propiedad `MDIParent` del formulario hijo con la del padre, para reconocer si el formulario MDI padre tiene hijos creados se usa la propiedad `MDIChildren` que tiene una propiedad `Length`.

Finalmente, para obtener el formulario hijo que se encuentra activo se usa la propiedad `ActiveMdiChild` del formulario padre.

A continuación un simple ejemplo que usa un formulario de Login (creado en el Demo 52) para ingresar al formulario MDI principal el cual se muestra degradado y con el nombre del usuario logueado.

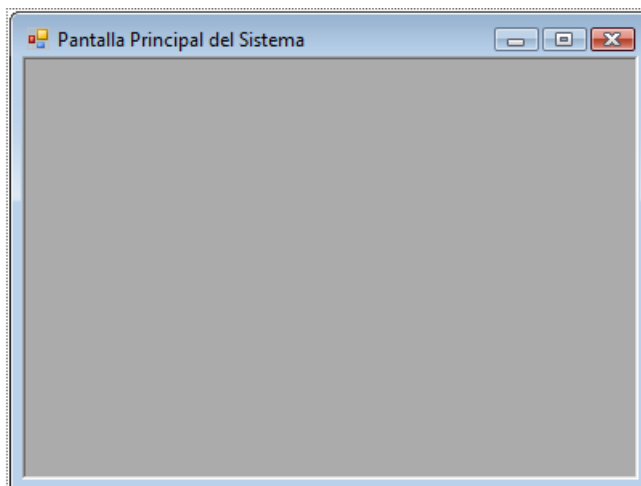
Demo 55

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo55.
- Cambiar de nombre al formulario de form1.vb a frmPrincipal.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmPrincipal
	IsMdiContainer	True
	Size	400,300
	Text	Pantalla Principal del Sistema
	WindowState	Maximized

- El diseño del formulario debe quedar similar al gráfico 4.9:

Gráfico 4.9: Diseño del formulario MDI Principal



- Escribir el siguiente código en el formulario:

```
Imports System.Drawing.Drawing2D
```

```
Public Class frmPrincipal
```

```
    Public Shared Usuario As String
```

```
    Private Sub GraficarMDI(ByVal sender As Object, _
```

```
        ByVal e As PaintEventArgs)
```

```
        Dim rec As New Rectangle(0, 0, Me.Width, Me.Height)
```

```
        Dim deg As New LinearGradientBrush(rec, Color.Aqua, _
```

```
        Color.Blue, LinearGradientMode.BackwardDiagonal)
```

```
        e.Graphics.FillRectangle(deg, rec)
```

```
        e.Graphics.DrawString("Usuario actual: " + Usuario, _
```

```
        New Font("Arial", 50), Brushes.White, 300, 300)
```

```
    End Sub
```

```
    Private Sub IniciarConfiguracion(ByVal sender As System.Object, _
```

```
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Dim oFondo As MdiClient = Me.Controls(0)
```

```
        AddHandler oFondo.Paint, AddressOf GraficarMDI
```

```
    End Sub
```

```
End Class
```

- Agregar el formulario de Login creado en el Demo 52: seleccionar el proyecto, del menú "Project", seleccionar "Add Existing Item", navegar hasta la carpeta Demo 52 y seleccionar "frmLogin.vb".
- Modificar el código del formulario Login como sigue:

```
Imports System.IO
```

```
Public Class frmLogin
```

```
    Private contador As Integer
```

```
    Private Sub HabilitarBotonAceptar(ByVal sender As System.Object, _
```

```
        ByVal e As System.EventArgs) Handles _
```

```
        txtUsuario.TextChanged, txtClave.TextChanged
```

```
        btnAceptar.Enabled = (txtUsuario.Text <> "" _
```

```
        AndAlso txtClave.Text <> "")
```

```
    End Sub
```

```
Private Sub MostrarMensaje(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles _  
    txtUsuario.Enter, txtClave.Enter  
    lblMensaje.Text = sender.Tag  
End Sub
```

```
Private Sub BorrarMensaje(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles _  
    txtUsuario.Leave, txtClave.Leave  
    lblMensaje.Text = ""  
End Sub
```

```
Private Sub Aceptar(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles btnAceptar.Click  
    Dim Archivo As String = _  
        "C:\Lduenas\NET\LibroVB2010\Archivos\Usuarios.txt"  
    If File.Exists(Archivo) Then  
        Dim Encontro As Boolean = False  
        Using fs As New FileStream(Archivo, _  
            FileMode.Open, FileAccess.Read, FileShare.Read)  
            Using sr As New StreamReader(fs)  
                Dim Usuario() As String  
                Do While Not sr.EndOfStream  
                    Usuario = sr.ReadLine.Split(",")  
                    If Usuario.Length = 2 Then  
                        If Usuario(0) = txtUsuario.Text _  
                            AndAlso Usuario(1) = txtClave.Text Then  
                            frmPrincipal.Usuario = Usuario(0)  
                            Encontro = True  
                            Exit Do  
                        End If  
                    End If  
                Loop  
                contador += 1  
            End Using  
        End Using  
        If Encontro Then  
            frmPrincipal.Show()  
            Me.Close()  
        End If  
    End If  
End Sub
```

```
Else
    MessageBox.Show("Login Invalido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
End If
If contador = 3 Then
    MessageBox.Show("Acceso Denegado", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    Me.Close()
End If
End If
End Sub

Private Sub Cancelar(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCancelar.Click
    Me.Close()
End Sub
End Class
```

Nota: No olvidar que el Login usa un archivo de texto que se encuentra en una cierta ruta.

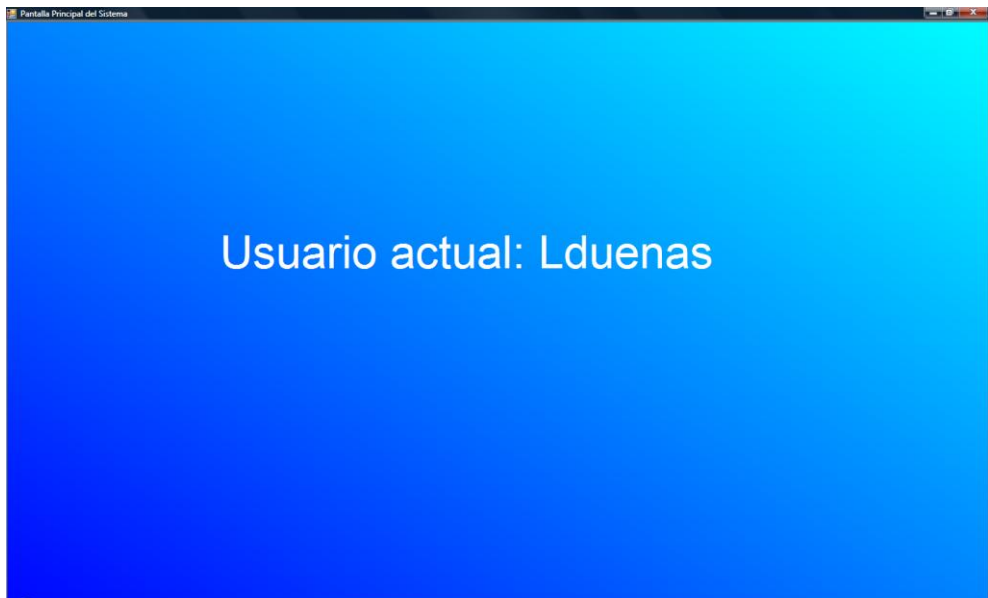
- Configurar el formulario de inicio y la forma del cierre: seleccionar el proyecto, dar clic derecho "Properties" y aparecerá la figura siguiente.

Gráfico 4.10: Ficha de configuración de la aplicación

The image shows a Windows application configuration dialog box. It has several sections: 'Assembly name' with a text box containing 'Demo55'; 'Root namespace' with a text box containing 'Demo55'; 'Application type' with a dropdown menu set to 'Windows Forms Application'; 'Icon' with a dropdown menu set to '(Default Icon)' and a small icon preview; 'Startup form' with a dropdown menu set to 'frmLogin'; two buttons: 'Assembly Information...' and 'View UAC Settings'; a checked checkbox 'Enable application framework'; a section titled 'Windows application framework properties' containing three checked checkboxes: 'Enable XP visual styles', 'Make single instance application', and 'Save My.Settings on Shutdown'; 'Authentication mode' with a dropdown menu set to 'Windows'; 'Shutdown mode' with a dropdown menu set to 'When last form closes'; 'Splash screen' with a dropdown menu set to '(None)'; and a 'View Application Events' button.

- En la opción "Startup form" elegir frmLogin y en "Shutdown mode" elegir When last form closes.
- Grabar y ejecutar la aplicación.
- Aparecerá el formulario Login primero, ingresar el usuario y la clave y se mostrará el MDI como en la siguiente figura.

Gráfico 4.11: Ejecución del formulario MDI Principal



2.2. Creando Menús

Existen 2 tipos de menús en las aplicaciones Windows: uno es el menú principal (clase MenuStrip) y el otro es el menú contextual (clase ContextMenuStrip). Ambos menús tienen una colección de elementos de tipo ToolStripMenuItem.

Así como todos los controles de Windows Forms, existen 2 formas de crear ambos tipos de menús: en tiempo de diseño y en tiempo de ejecución o dinámicamente.

Para crear un menú Principal en tiempo de diseño arrastrar el control MenuStrip, llenar las opciones del menú directamente en el diseñador del formulario, luego configurar las propiedades de cada opción como su nombre (Name), imagen (Image), tecla de acceso directo (ShortcutKeys) y luego programar en el evento clic de cada opción el código respectivo.

Para crear un menú contextual el procedimiento es similar que para el menú principal, es decir, arrastrar el control ContextMenuStrip, llenar sus opciones, configurar sus propiedades y programar las opciones, la diferencia es que para que se muestre al dar clic derecho sobre un cierto control o sobre el formulario, es necesario configurar la propiedad "ContextMenuStrip" del control o formulario donde se va a visualizar el menú desplegable.

A continuación presentamos un par de ejemplos muy interesantes que crean menús dinámicamente; el primero el Demo 56 crea a partir de un archivo XML las opciones del menú principal en un formulario MDI mediante el objeto XmlDocument de XML DOM, además ejecuta 3 tipos de acciones al seleccionarse una opción: abre un formulario si el prefijo de la acción inicia con "frm", ejecuta un programa si el sufijo es ".exe" y en caso contrario ejecuta un procedimiento con el nombre de la acción, para realizar esto último usamos "Reflection".

El segundo ejemplo, el Demo 57 crea un menú contextual dinámicamente, este se muestra al dar clic derecho sobre cuadros de textos para seleccionar una fecha de un calendario y llenar los textos en vez de escribir la fecha.

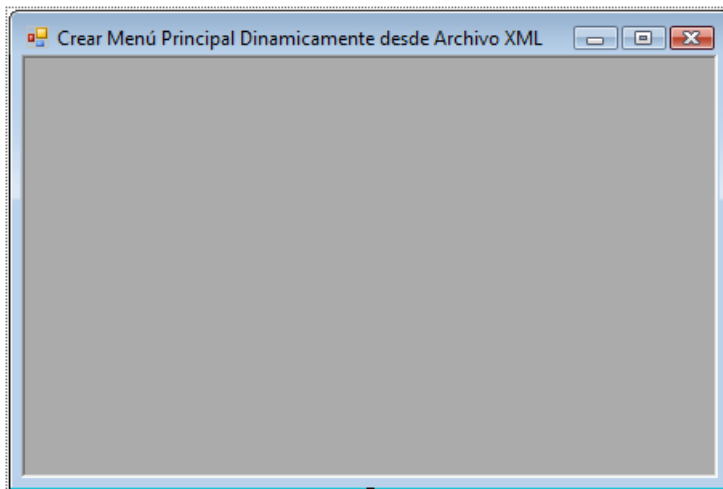
Demo 56

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo56.
- Cambiar de nombre al formulario de form1.vb a frmPrincipal.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmPrincipal
	IsMdiContainer	True
	Size	450,300
	Text	Crear Menú Principal Dinamica mente desde Archivo XML
	WindowState	Maximized

- El diseño del formulario debe quedar similar al gráfico 4.12:

Gráfico 4.12: Diseño del formulario MDI Principal



- Agregar un archivo XML: del menú "Project", seleccionar "Add New Item" y en la sección "Data" elegir "XML File" y escribir como nombre: "Menus.xml".
- Escribir en el archivo XML las opciones que iran en el menú principal:

```
<?xml version="1.0" encoding="utf-8" ?>
<Menus>
  <Menu Nombre="Mantenimiento">
    <Menu Nombre="Categorias" Accion="frmCategorias"/>
    <Menu Nombre="Clientes" Accion="frmClientes"/>
    <Menu Nombre="Proveedores">
      <Menu Nombre="Nacionales" Accion="frmProveedoresNacionales"/>
      <Menu Nombre="Extranjeros" Accion="frmProveedoresExtranjeros"/>
    </Menu>
  </Menu>
  <Menu Nombre="Procesos">
    <Menu Nombre="Crear Comprobantes">
      <Menu Nombre="Factura" Accion="frmCrearFactura"/>
      <Menu Nombre="Boleta" Accion="frmCrearBoleta"/>
    </Menu>
    <Menu Nombre="Planillas">
      <Menu Nombre="Empleados" Accion="frmPlanillaEmpleados"/>
      <Menu Nombre="Obreros" Accion="frmPlanillaObreros"/>
    </Menu>
  </Menu>
  <Menu Nombre="Programas">
    <Menu Nombre="Windows">
      <Menu Nombre="Bloc de Notas" Accion="Notepad.exe"/>
      <Menu Nombre="Calculadora" Accion="Calc.exe"/>
      <Menu Nombre="Explorador" Accion="Explorer.exe"/>
    </Menu>
    <Menu Nombre="Office">
      <Menu Nombre="Word" Accion="Winword.exe"/>
      <Menu Nombre="Excel" Accion="Excel.exe"/>
      <Menu Nombre="Power Point" Accion="PowerPnt.exe"/>
    </Menu>
  </Menu>
  <Menu Nombre="Salir" Accion="Salir">
  </Menu>
</Menus>
```

Nota: El archivo XML define un documento XML el cual tiene reglas de sintáxis obligatorias sino se genera un error, entre ellas: debe existir un solo elemento raíz, toda etiqueta de inicio debe tener un fin, es case sensitive, atributos deben estar entre comillas simples o dobles.

En el archivo creado "Menus" y "Menu" son conocidos como elementos y "Nombre" y "Accion" son conocidos como atributos.

- Agregar los siguientes formularios al proyecto: frmCategorias, frmClientes, frmProveedoresNacionales, frmProveedoresExtranjeros, frmCrearFactura, frmCrearBoleta, frmPlanillaEmpleados y frmPlanilla Obreros.

Nota: Todos los formularios agregados no es necesario que contengan controles solo es para probar que cargen.

- Modificar el código del formulario Principal como sigue:

```
Imports System.Xml 'XmlDocument
Imports System.Reflection 'Assembly

Public Class frmPrincipal
    Private Sub CargarXml(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim ArchivoXml As String = _
            "C:\Lduenas\NET\LibroVB2010\Demo56\Menus.xml"
        Dim oDocXml As New XmlDocument
        oDocXml.Load(ArchivoXml)
        Dim mnuPrincipal As New MenuStrip
        For Each oNodo As XmlNode In _
            oDocXml.DocumentElement.ChildNodes
            Dim oMenu As New ToolStripMenuItem _
                (oNodo.Attributes("Nombre").Value)
            EjecutarAccion(oNodo, oMenu)
            mnuPrincipal.Items.Add(oMenu)
            CrearMenu(oNodo, oMenu)
        Next
        Me.Controls.Add(mnuPrincipal)
    End Sub
```

```
Private Sub CrearMenu(ByVal oNodoRaiz As XmlNode, _
ByVal oMenuRaiz As ToolStripMenuItem)
    For Each oNodo As XmlNode In oNodoRaiz.ChildNodes
        Dim oMenu As New ToolStripMenuItem _
            (oNodo.Attributes("Nombre").Value)
        EjecutarAccion(oNodo, oMenu)
        oMenuRaiz.DropDownItems.Add(oMenu)
        CrearMenu(oNodo, oMenu)
    Next
End Sub

Private Sub EjecutarAccion(ByVal oNodo As XmlNode, _
ByVal oMenu As ToolStripMenuItem)
    If oNodo.Attributes("Accion") IsNot Nothing _
        AndAlso oNodo.Attributes("Accion").Value <> "" Then
        Dim Accion As String = oNodo.Attributes("Accion").Value
        oMenu.Tag = Accion.Substring(0, Accion.Length)
        If Accion.StartsWith("frm") Then
            AddHandler oMenu.Click, AddressOf AbrirForm
        ElseIf Accion.EndsWith(".exe") Then
            AddHandler oMenu.Click, AddressOf EjecutarPrograma
        Else
            AddHandler oMenu.Click, AddressOf EjecutarProcedimiento
        End If
    End If
End Sub

Private Sub AbrirForm(ByVal sender As Object, _
ByVal e As EventArgs)
    Dim Ensamblado As Assembly = _
        [Assembly].GetEntryAssembly
    Dim Tipo As Type = _
        Ensamblado.GetType("Demo56." + sender.Tag)
    If Tipo IsNot Nothing Then
        Dim obj As Object = _
            Activator.CreateInstance(Tipo)
        If TypeOf obj Is Form Then
            Dim frm As Form = CType(obj, Form)
            frm.MdiParent = Me
        End If
    End If
End Sub
```

```
        frm.Show()
    End If
End If
End Sub

Private Sub EjecutarProcedimiento(ByVal sender As Object, _
    ByVal e As EventArgs)
    Dim obj As Type = Me.GetType
    If obj IsNot Nothing Then
        Dim X As Object = _
            Activator.CreateInstance(obj)
        Dim metodo As MethodInfo = _
            obj.GetMethod(sender.Tag)
        If metodo IsNot Nothing Then _
            metodo.Invoke(X, Nothing)
    End If
End Sub

Private Sub EjecutarPrograma(ByVal sender As Object, _
    ByVal e As EventArgs)
    Process.Start(sender.Tag)
End Sub

Public Sub Salir()
    MessageBox.Show("Hasta la próxima")
    Application.Exit()
End Sub
End Class
```

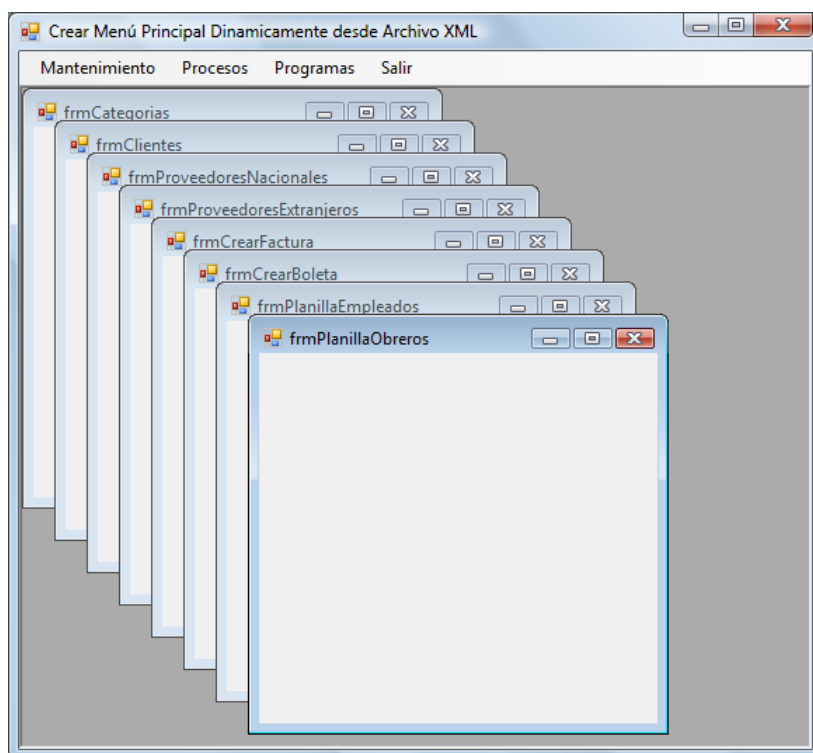
Nota: Para cargar el archivo XML como un árbol de nodos se usa el método Load de la clase XmlDocument, la propiedad DocumentElement de esta clase representa al elemento raíz en nuestro caso Menus y childNodes a todos los nodos hijos en nuestro caso Menu. Primero se carga las opciones del menú principal y luego las de nivel secundario.

Importante: Al igual que otros ejemplos del libro hacemos uso de la recursividad para volver a ejecutar un procedimiento, en este caso la rutina CrearMenu se llama mientras existan nodos hijos o menus en el XML.

También es importante mencionar que en la propiedad Tag del menú se ha guardado la acción que tiene el nombre del formulario a abrir, el nombre del proceso a ejecutar si es un programa o el nombre del procedimiento a ejecutar.

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.13: Ejecución del formulario MDI Principal



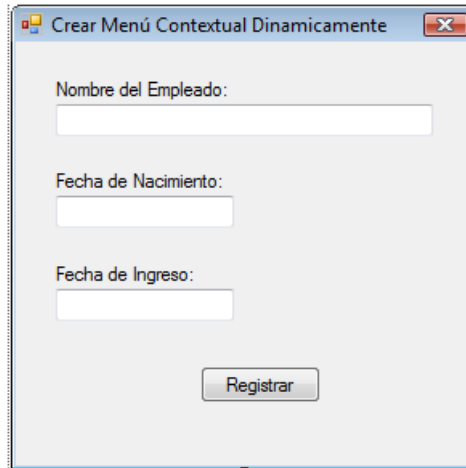
Demo 57

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo57.
- Cambiar de nombre al formulario de form1.vb a frmRegistroFechas.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name FormBorderStyle MaximizeBox MinimizeBox Size StartPosition Text	frmRegistroFechas FixedSingle False False 300, 300 CenterScreen Crear Menú Contextual Dinamicamente
Label1	Name AutoSize Location Text	lblNombre True 22, 23 Nombre del Empleado:
TextBox1	Name Location Size	txtNombre 25,39 235,20
Label2	Name AutoSize Location Text	lblFechaNacimiento True 22, 80 Fecha de Nacimiento:
TextBox2	Name Location Size	txtFechaNacimiento 25,96 111,20
Label3	Name AutoSize Location Text	lblFechaIngreso True 22, 138 Fecha de Ingreso:
TextBox3	Name Location Size	txtFechaIngreso 25,154 111,20

- El diseño del formulario debe quedar similar al gráfico 4.14:

Gráfico 4.14: Diseño del formulario Registro de Fechas

The image shows a Windows application window titled "Crear Menú Contextual Dinamicamente". Inside the window, there is a light gray background. At the top, there are three text input fields. The first is labeled "Nombre del Empleado:", the second "Fecha de Nacimiento:", and the third "Fecha de Ingreso:". Each label is positioned to the left of its corresponding text box. Below these three text boxes, centered horizontally, is a button with the text "Registrar". The window has a standard Windows title bar with a close button (X) in the top right corner.

- Escribir el siguiente código en el formulario:

Public Class frmRegistroFechas

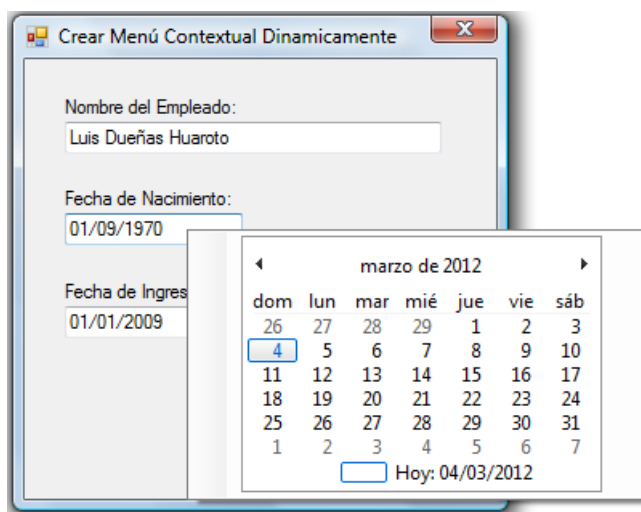
```
Private Sub CrearMenuContextualCalendario(ByVal sender As _  
System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  
    Dim mcCalendario As New MonthCalendar  
    Dim host As New ToolStripControlHost(mcCalendario)  
    Dim mnuCalendario As New ContextMenuStrip  
    mnuCalendario.Items.Add(host)  
    txtFechaNacimiento.ContextMenuStrip = mnuCalendario  
    txtFechaIngreso.ContextMenuStrip = mnuCalendario  
    AddHandler mcCalendario.DateChanged, AddressOf AsignarFecha  
End Sub
```

```
Private Sub AsignarFecha(ByVal sender As Object, ByVal e As EventArgs)  
    Dim FechaSeleccionada As Date = CType(sender, _  
MonthCalendar).SelectionStart  
    CType(sender.Parent, ContextMenuStrip).Close()  
    Me.ActiveControl.Text = FechaSeleccionada  
End Sub  
End Class
```

Importante: El ToolStripControlHost permite agregar cualquier tipo de control a un menú, no solo un calendario, sino podría ser una grilla, un control de usuario, etc.

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.15: Ejecución del formulario Registro de Fechas



Nota: Para que aparezca el calendario dar clic derecho sobre el cuadro de texto de la fecha de nacimiento o la fecha de ingreso, seleccionar un día y se mostrará sobre el cuadro de texto la fecha seleccionada.

2.3. Usando Diálogos de Windows

En Windows Forms tenemos 5 diálogos comunes de Windows que son: el diálogo de abrir archivo (clase OpenFileDialog), el diálogo de guardar archivo (clase SaveFileDialog), el diálogo de mostrar directorios (clase FolderBrowserDialog), el diálogo de Color (clase ColorDialog) y el diálogo de Fuente (clase FontDialog).

Todos los diálogos tienen el método ShowDialog que permite mostrar el diálogo, los diálogos de archivos: OpenFileDialog y SaveFileDialog comparten propiedades comunes como el Title que configura el título del diálogo, Filter que especifica los tipos de archivos a mostrar, FileName que indica el archivo seleccionado, etc.

Por su parte la propiedad más importante del diálogo de color o ColorDialog es la propiedad "Color" que representa el color seleccionado del diálogo, igualmente el diálogo de fuente o FontDialog tiene como propiedad mas importante la propiedad "Font" que devuelve la fuente seleccionada del diálogo y finalmente el diálogo de mostrar directorios o FolderBrowserDialog tiene la propiedad "SelectedPath" que representa la ruta del directorio seleccionado en el diálogo.

A continuación un ejemplo de cómo trabajar con los diálogos de fuente y color para configurar una marquesina o mensaje en movimiento a través de un menú de configuración.

Demo 58

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo58.
- Cambiar de nombre al formulario de form1.vb a frmMarquesina.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmMarquesina

	Size Text WindowState	300, 300 Marquesina Configurable Maximized
MenuStrip1	Name	mnuPrincipal
MenuItem1	Name Text	mnuConfiguracion Configuracion
MenuItem1.1	Name Text	mnuIniciar Iniciar
MenuItem1.2	Name Text	mnuMensaje Mensaje
MenuItem1.3	Name Text	mnuFuente Fuente
MenuItem1.4	Name Text	mnuColor Color

- El diseño del formulario debe quedar similar al gráfico 4.16:

Gráfico 4.16: Diseño del formulario Marquesina Configurable



- Escribir el siguiente código en el formulario:

```
Public Class frmMarquesina
    Private lblMensaje As New Label
    Private WithEvents tmrAnimar As New Timer

    Private Sub ConfiguracionPorDefecto(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
With lblMensaje
    .AutoSize = True
    .Text = "Visual Basic 2010"
    .Font = New Font("Arial", 50)
    .ForeColor = Color.Blue
End With
Me.Controls.Add(lblMensaje)
lblMensaje.Location = New Point(-lblMensaje.Width, 300)
tmrAnimar.Interval = 100
End Sub
```

```
Private Sub HabilitarOpciones(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles _
mnuConfiguracion.DropDownOpening
    Dim habilitado As Boolean = mnuIniciar.Text.Equals("Iniciar")
    mnuMensaje.Enabled = habilitado
    mnuFuente.Enabled = habilitado
    mnuColor.Enabled = habilitado
End Sub
```

```
Private Sub IniciarAnimacion(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuIniciar.Click
    If mnuIniciar.Text = "Iniciar" Then
        tmrAnimar.Enabled = True
        tmrAnimar.Start()
        mnuIniciar.Text = "Detener"
    Else
        tmrAnimar.Stop()
        tmrAnimar.Enabled = False
        mnuIniciar.Text = "Iniciar"
    End If
End Sub
```

```
Private Sub tmrAnimar_Tick(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles tmrAnimar.Tick
    If lblMensaje.Location.X < Me.Width Then
        lblMensaje.Location = New Point(lblMensaje.Location.X + 10, _
        lblMensaje.Location.Y)
    Else
        lblMensaje.Location = New Point(-lblMensaje.Width, 300)
    End If
End Sub
```

```
End If  
End Sub
```

```
Private Sub CambiarMensaje(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles mnuMensaje.Click  
    lblMensaje.Text = InputBox("Nuevo Mensaje:", _  
        "Cambiar Mensaje", lblMensaje.Text)  
    lblMensaje.Location = New Point(-lblMensaje.Width, 300)  
End Sub
```

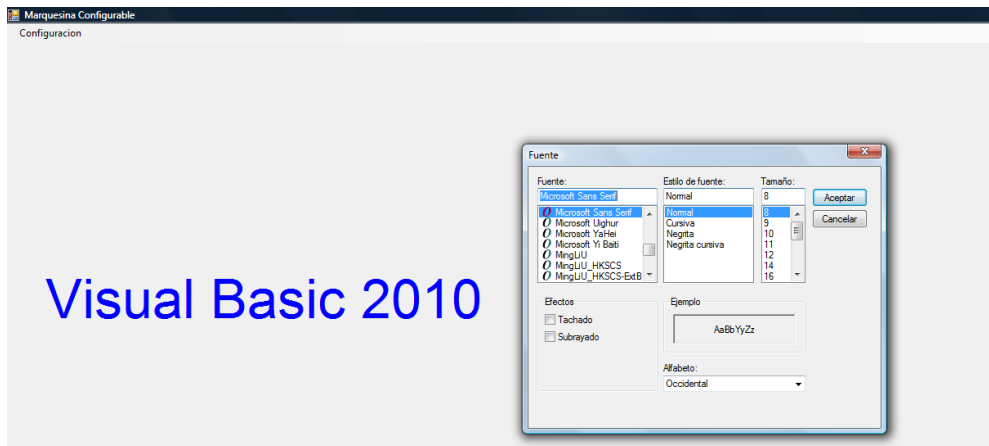
```
Private Sub CambiarFuente(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles mnuFuente.Click  
    Dim fd As New FontDialog  
    If fd.ShowDialog = Windows.Forms.DialogResult.OK Then  
        lblMensaje.Font = fd.Font  
        lblMensaje.Location = New Point(-lblMensaje.Width, 300)  
    End If  
End Sub
```

```
Private Sub CambiarColor(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles mnuColor.Click  
    Dim cd As New ColorDialog  
    If cd.ShowDialog = Windows.Forms.DialogResult.OK Then  
        lblMensaje.ForeColor = cd.Color  
    End If  
End Sub  
End Class
```

Nota: En el código anterior solo se habilita las opciones de configuración del mensaje, fuente y color mientras el mensaje no este en movimiento, es decir para configurar la marquesina hay que detenerla.

- Grabar y luego ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.17: Ejecución del formulario Marquesina Configurable



2.4. Agregando Barras

En esta última parte de mejorar la interface de usuario veremos como agregar una barra de herramientas (clase ToolStrip) y barra de estado (clase StatusStrip).

Ambas barras se componen de una colección de elementos de tipo ToolStripItem, que podría ser un botón o ToolStripButton, un separador o ToolStripSeparator, una etiqueta o ToolStripLabel, un Combo o ToolStripComboBox, un texto o ToolStripTextBox, etc. Cada uno de estos elementos (ToolStripItem) pueden tener un texto (propiedad Text), una imagen (propiedad Image), una mensaje al pasar el mouse (propiedad ToolTipText), etc. Para programar la barra de herramientas se escribe el código en el evento Click de cada opción o ToolStripItem.

La diferencia entre ambas barras es que la barra de herramientas puede tener cualquier objeto de tipo ToolStripItem, en cambio la barra de estado solo puede tener etiquetas, barras de progreso, botón desplegable y botón de división.

A continuación presentamos un ejemplo completo de cómo mejorar la interface de usuario usando MDIs, Menús principal y contextual, diálogos de Windows y barras de herramientas y de estado. Este ejemplo se trata de un Editor de Documentos de Texto Enriquecido (RTF) similar al WordPad el cual permite crear un documento, abrir, guardar, hacer preview, configurar pagina, imprimir, copiar, cortar, pegar, cambiar fuente, color de fondo, ejecutar programas de Windows y de Office y organizar ventanas hijas dentro del MDI padre.

Demo 59

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo59.
- Cambiar de nombre al formulario de form1.vb a frmEditor.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name Size IsMdiContainer Text WindowState	frmEditor 500, 300 True Editor de Documentos de Texto Enriquecido Maximized
MenuStrip1	Name MdiWindowListItem	mnuPrincipal mnuVentana
MenuItem1	Name Text	mnuArchivo Archivo
MenuItem1.1	Name Image ShortcutKeys Text	mnuNuevo Nuevo.png Ctrl + N &Nuevo
MenuItem1.2	Name Image ShortcutKeys Text	mnuAbrir Abrir.png Ctrl + A &Abrir
MenuItem1.3	Name Image ShortcutKeys Text	mnuGuardar Guardar.png Ctrl + G &Guardar
MenuItem1.4	Name Text	mnuLinea -
MenuItem1.5	Name Text	mnuPreview1 Preview 1
MenuItem1.6	Name Text	mnuPreview2 Preview 2
MenuItem1.7	Name Text	mnuPageSetup Page Setup
MenuItem1.8	Name ShortcutKeys Text	mnuPrint Ctrl + P &Print
MenuItem2	Name Text	mnuEdicion Edición
MenuItem2.1	Name Image ShortcutKeys Text	mnuCopiar Copiar.png Ctrl + C &Copiar

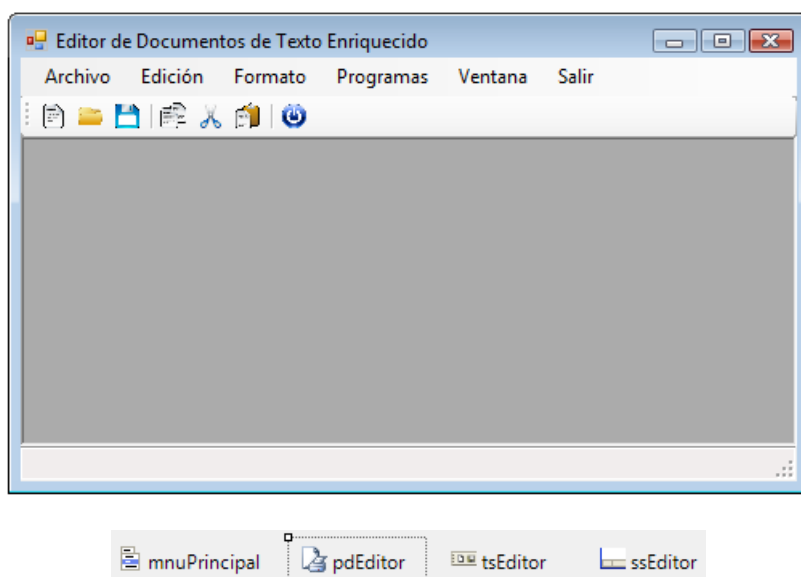
MenuItem2.2	Name Image ShortCutKeys Text	mnuCortar Cortar.png Ctrl + X Co&rtar
MenuItem2.3	Name Image ShortCutKeys Text	mnuPegar Pegar.png Ctrl + V &Pegar
MenuItem3	Name Text	mnuFormato Formato
MenuItem3.1	Name Text	mnuFuente Fuente
MenuItem3.2	Name Text	mnuColorFondo Color de Fondo
MenuItem4	Name Text	mnuProgramas Programas
MenuItem4.1	Name Text	mnuWindows Windows
MenuItem4.1.1	Name Tag Text	mnuCalculadora Calc Calculadora
MenuItem4.1.2	Name Tag Text	mnuBlocDeNotas Notepad Bloc de Notas
MenuItem4.1.3	Name Tag Text	mnuExplorador Explorer Explorador
MenuItem4.2	Name Text	mnuOffice Office
MenuItem4.2.1	Name Tag Text	mnuWord WinWord Word
MenuItem4.2.2	Name Tag Text	mnuExcel Excel Excel
MenuItem4.2.3	Name Tag Text	mnuInternetExplorer IExplore Internet Explorer
MenuItem5	Name Text	mnuVentana Ventana
MenuItem5.1	Name Tag	mnuCascada 0

	Text	Cascada
MenuItem5.2	Name Tag Text	mnuMosaicoHorizontal 1 Mosaico Horizontal
MenuItem5.3	Name Tag Text	mnuMosaicoVertical 2 Mosaico Vertical
MenuItem5.4	Name Tag Text	mnuOrganizarIconos 3 Organizar Iconos
MenuItem6	Name Text	mnuSalir Salir
ToolStrip1	Name	tsEditor
ToolStripButton1	Name Image Text ToolTipText	tsbNuevo Nuevo.png Nuevo Nuevo
ToolStripButton2	Name Image Text ToolTipText	tsbAbrir Abrir.png Abrir Abrir
ToolStripButton3	Name Image Text ToolTipText	tsbGuardar Guardar.png Guardar Guardar
ToolStripSeparator1	Name	tssArchivo
ToolStripButton4	Name Image Text ToolTipText	tsbCopiar Copiar.png Copiar Copiar
ToolStripButton5	Name Image Text ToolTipText	tsbCortar Cortar.png Cortar Cortar
ToolStripButton6	Name Image Text ToolTipText	tsbPegar Pegar.png Pegar Pegar
ToolStripSeparator2	Name	tssEdicion
ToolStripButton7	Name Image Text	tsbSalir Salir.png Salir

	ToolTipText	Salir
StatusStrip1	Name	ssEditor
ToolStripStatusLabel1	Name AutoSize Size Text	tsslArchivo False 121, 17
ToolStripStatusLabel2	Name AutoSize Size Text	tsslFechaHora False 121, 17
PrintDocument1	Name	pdEditor

- El diseño del formulario debe quedar similar al gráfico 4.18:

Gráfico 4.18: Diseño del formulario MDI Editor



- Agregar el formulario de Login creado en el Demo 55: seleccionar el proyecto, del menú "Project", seleccionar "Add Existing Item", navegar hasta la carpeta Demo 55 y seleccionar "frmLogin.vb".
- Modificar el código del formulario Login como sigue:

```
Imports System.IO
```

Public Class frmLogin

Private contador As Integer

```
Private Sub HabilitarBotonAceptar(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles _  
txtUsuario.TextChanged, txtClave.TextChanged  
    btnAceptar.Enabled = (txtUsuario.Text <> "" _  
    AndAlso txtClave.Text <> "")  
End Sub
```

```
Private Sub MostrarMensaje(ByVal sender As Object, _  
ByVal e As System.EventArgs) Handles _  
txtUsuario.Enter, txtClave.Enter  
    lblMensaje.Text = sender.Tag  
End Sub
```

```
Private Sub BorrarMensaje(ByVal sender As Object, _  
ByVal e As System.EventArgs) Handles _  
txtUsuario.Leave, txtClave.Leave  
    lblMensaje.Text = ""  
End Sub
```

```
Private Sub Aceptar(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAceptar.Click  
    Dim Archivo As String = _  
    "C:\Lduenas\NET\LibroVB2010\Archivos\Usuarios.txt"  
    If File.Exists(Archivo) Then  
        Dim Encontro As Boolean = False  
        Using fs As New FileStream(Archivo, _  
        FileMode.Open, FileAccess.Read, FileShare.Read)  
            Using sr As New StreamReader(fs)  
                Dim Usuario() As String  
                Do While Not sr.EndOfStream  
                    Usuario = sr.ReadLine.Split(",")  
                    If Usuario.Length = 2 Then  
                        If Usuario(0) = txtUsuario.Text _  
                        AndAlso Usuario(1) = txtClave.Text Then  
                            Encontro = True  
                            Exit Do  
                        End If  
                    End If  
            End Using  
        End Using  
    End If
```

```

        End If
    Loop
    contador += 1
End Using
End Using
If Encontro Then
    frmEditor.Show()
    Me.Close()
Else
    MessageBox.Show("Login Invalido", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
End If
If contador = 3 Then
    MessageBox.Show("Acceso Denegado", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    Me.Close()
End If
End If
End Sub

Private Sub Cancelar(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnCancelar.Click
    Me.Close()
End Sub
End Class

```

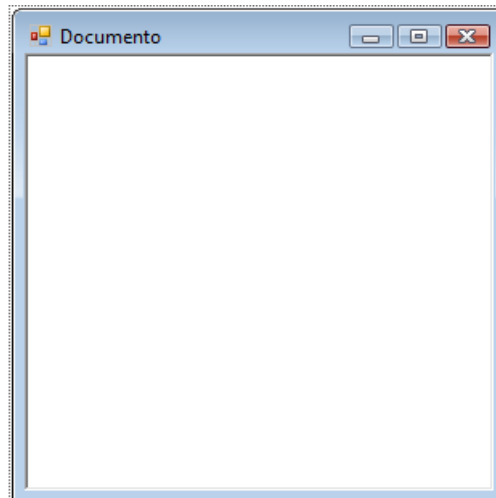
- Agregar un nuevo formulario: seleccionar el proyecto, del menú "Project", seleccionar "Add Windows Forms", escribir como nombre "frmDocumento.vb".
- Configurar el formulario Documento de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmDocumento
	Size	300, 300
	Text	Documento
ContextMenuStrip1	Name	mnuEdicion
ToolStripMenuItem1	Name	mnuCopiar
	Image	Copiar.png
	Text	Copiar

ToolStripMenuItem2	Name Image Text	mnuCortar Cortar.png Cortar
ToolStripMenuItem3	Name Image Text	mnuPegar Pegar.png Pegar
RichTextBox1	Name ContextMenuStrip Dock	rtbDocumento mnuEdicion Fill

- El diseño del formulario debe quedar similar al gráfico 4.19:

Gráfico 4.19: Diseño del formulario Documento



- Escribir el siguiente código en el formulario Documento:

```
Public Class frmDocumento
```

```
    Private Sub Copiar(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles mnuCopiar.Click
        Clipboard.SetText(rtbDocumento.SelectedText)
    End Sub
```

```
    Private Sub Cortar(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles mnuCortar.Click
        Clipboard.SetText(rtbDocumento.SelectedText)
```

```
    rtbDocumento.SelectedText = ""  
End Sub
```

```
Private Sub Pegar(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles mnuPegar.Click  
    rtbDocumento.SelectedText = Clipboard.GetText  
End Sub  
End Class
```

- Regresar al formulario Editor y escribir el siguiente código:

```
Imports System.Drawing.Drawing2D 'LinearGradientBrush  
Imports System.IO 'Path  
  
Public Class frmEditor  
    Private NDoc As Integer  
    Private WithEvents fd As New FontDialog  
  
    Private Sub IniciarCarga(ByVal sender As System.Object, _  
        ByVal e As System.EventArgs) Handles MyBase.Load  
        Clipboard.Clear()  
        Dim X As Control  
        For Each X In Me.Controls  
            If TypeOf X Is MdiClient Then  
                Dim Area As MdiClient = CType(X, MdiClient)  
                AddHandler Area.Paint, AddressOf Graficar  
            End If  
        Next  
        tsslFechaHora.Text = Now.ToLongTimeString  
        tsslFechaHora.ToolTipText = Now.ToString  
    End Sub  
  
    Private Sub Graficar(ByVal sender As Object, _  
        ByVal e As System.Windows.Forms.PaintEventArgs)  
        Dim rec As Rectangle = Me.ClientRectangle  
        Dim deg As New LinearGradientBrush(rec, Color.Aqua, _  
            Color.Blue, LinearGradientMode.BackwardDiagonal)  
        e.Graphics.FillRectangle(deg, rec)  
        e.Graphics.DrawString("El Poder de .NET", _  
            New Font("Arial", 100), Brushes.Blue, 100, 200)  
    End Sub
```



```
Private Sub EjecutarPrograma(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles mnuCalculadora.Click, _  
    mnuBlocDeNotas.Click, mnuExplorador.Click, mnuWord.Click, _  
    mnuExcel.Click, mnuInternetExplorer.Click  
    Process.Start(String.Format("{0}.exe", sender.Tag))  
End Sub
```

```
Private Sub Nuevo(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles _  
    mnuNuevo.Click, tsbNuevo.Click  
    NDoc += 1  
    Dim oDoc As New frmDocumento  
    oDoc.Text = String.Format("Documento {0}", NDoc)  
    oDoc.MdiParent = Me  
    oDoc.Show()  
    tsslArchivo.Text = Me.ActiveMdiChild.Text  
End Sub
```

```
Private Sub OrganizarVentanas(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles mnuCascada.Click, _  
    mnuMosaicoHorizontal.Click, mnuMosaicoVertical.Click, _  
    mnuOrganizarIconos.Click  
    Me.LayoutMdi(sender.Tag)  
End Sub
```

```
Private Sub Abrir(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles mnuAbrir.Click, _  
    tsbAbrir.Click  
    Dim ofd As New OpenFileDialog  
    ofd.Title = "Selecciona archivo rtf"  
    ofd.Filter = "Archivos de formato enriquecido|*.rtf"  
    If ofd.ShowDialog() = Windows.Forms.DialogResult.OK Then  
        If Me.MdiChildren.Length = 0 Then mnuNuevo.PerformClick()  
        Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)  
        rtb.LoadFile(ofd.FileName, RichTextBoxStreamType.RichText)  
        Me.ActiveMdiChild.Text = Path.GetFileName(ofd.FileName)  
        tsslArchivo.Text = Me.ActiveMdiChild.Text  
    End If  
End Sub
```

```
Private Sub Guardar(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuGuardar.Click, _
tsbGuardar.Click
    Dim sfd As New SaveFileDialog
    sfd.Title = "Selecciona archivo rtf"
    sfd.Filter = "Archivos de formato enriquecido|*.rtf"
    If sfd.ShowDialog() = Windows.Forms.DialogResult.OK Then
        Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)
        rtb.SaveFile(sfd.FileName, RichTextBoxStreamType.RichText)
        Me.ActiveMdiChild.Text = Path.GetFileName(sfd.FileName)
        tsslArchivo.Text = Me.ActiveMdiChild.Text
    End If
End Sub
```

```
Private Sub HabilitarMenusArchivo(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles _
mnuArchivo.DropDownOpening
    Dim Habilitado As Boolean = _
(Me.MdiChildren.Length > 0)
    mnuGuardar.Enabled = Habilitado
    mnuPreview1.Enabled = Habilitado
    mnuPreview2.Enabled = Habilitado
    mnuPageSetup.Enabled = Habilitado
    mnuPrint.Enabled = Habilitado
End Sub
```

```
Private Sub CrearPagina(ByVal sender As System.Object, _
ByVal e As System.Drawing.Printing.PrintPageEventArgs) _
Handles pdEditor.PrintPage
    Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)
    Dim X As Integer = e.MarginBounds.Left
    Dim Y As Integer = e.MarginBounds.Top
    e.Graphics.DrawString(rtb.Text, rtb.Font, _
    New SolidBrush(rtb.ForeColor), X, Y)
End Sub
```

```
Private Sub PreviewConControl(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuPreview1.Click
    Dim frm As New Form
```

```
frm.WindowState = FormWindowState.Maximized
frm.Text = "Preview del Documento"
Dim ppc As New PrintPreviewControl
ppc.Document = pdEditor
ppc.Dock = DockStyle.Fill
frm.Controls.Add(ppc)
frm.ShowDialog()
End Sub

Private Sub PreviewConDialogo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuPreview2.Click
    Dim ppd As New PrintPreviewDialog
    ppd.Document = pdEditor
    ppd.ShowDialog()
End Sub

Private Sub PageSetup(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuPageSetup.Click
    Dim psd As New PageSetupDialog
    psd.Document = pdEditor
    psd.ShowDialog()
End Sub

Private Sub Print(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuPrint.Click
    Dim pd As New PrintDialog
    pd.AllowCurrentPage = True
    pd.AllowPrintToFile = True
    pd.AllowSelection = True
    pd.AllowSomePages = True
    pd.UseEXDialog = True
    pd.Document = pdEditor
    If pd.ShowDialog() = _
    Windows.Forms.DialogResult.OK Then
        pdEditor.Print()
    End If
End Sub

Private Sub HabilitarMenuEdicion(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles _
```

```
mnuEdicion.DropDownOpening
    Dim Habilitado As Boolean = _
    (Me.MdiChildren.Length > 0)
    If Not Habilitado Then
        mnuCopiar.Enabled = False
        mnuCortar.Enabled = False
        mnuPegar.Enabled = False
    Else
        Dim rtb As RichTextBox = _
        Me.ActiveMdiChild.Controls(0)
        Habilitado = (rtb.SelectedText <> "")
        mnuCopiar.Enabled = Habilitado
        mnuCortar.Enabled = Habilitado
        mnuPegar.Enabled = Clipboard.GetText <> ""
    End If
End Sub

Private Sub Copiar(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles _
mnuCopiar.Click, tsbCopiar.Click
    Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)
    Clipboard.SetText(rtb.SelectedText)
End Sub

Private Sub Cortar(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles _
mnuCortar.Click, tsbCortar.Click
    Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)
    Clipboard.SetText(rtb.SelectedText)
    rtb.SelectedText = ""
End Sub

Private Sub Pegar(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles _
mnuPegar.Click, tsbPegar.Click
    Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)
    rtb.SelectedText = Clipboard.GetText
End Sub

Private Sub HabilitarMenuFormato(ByVal sender As Object, _
```

```
ByVal e As System.EventArgs) Handles _
mnuFormato.DropDownOpening
    Dim Habilitado As Boolean = (Me.MdiChildren.Length > 0)
    mnuFuente.Enabled = Habilitado
    mnuColorDeFondo.Enabled = Habilitado
End Sub

Private Sub CambiarFuente()
    Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)
    If rtb.SelectedText <> "" Then
        rtb.SelectionFont = fd.Font
        rtb.SelectionColor = fd.Color
    Else
        rtb.Font = fd.Font
        rtb.ForeColor = fd.Color
    End If
End Sub

Private Sub Fuente(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuFuente.Click
    fd.ShowColor = True
    fd.ShowApply = True
    fd.ShowHelp = True
    If fd.ShowDialog() = Windows.Forms.DialogResult.OK Then
        CambiarFuente()
    End If
End Sub

Private Sub AplicarFuente(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles fd.Apply
    CambiarFuente()
End Sub

Private Sub MostrarAyuda(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles fd.HelpRequest
    Help.ShowHelp(Me, "http://www.google.com")
End Sub

Private Sub ColorDeFondo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuColorDeFondo.Click
```

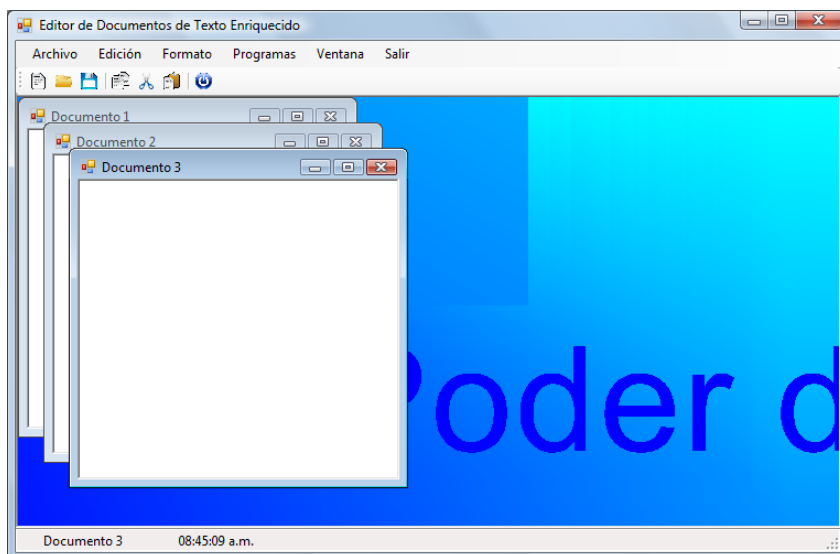
```
Dim cd As New ColorDialog
If cd.ShowDialog() = Windows.Forms.DialogResult.OK Then
    Dim rtb As RichTextBox = Me.ActiveMdiChild.Controls(0)
    rtb.BackColor = cd.Color
End If
End Sub

Private Sub Salir(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles _
mnuSalir.Click, tsbSalir.Click
    Me.Close()
End Sub
End Class
```

Nota: En el código anterior se usa la propiedad Tag de los menús de programas para especificar el programa a abrir y la propiedad tag de los menús de ventanas para indicar el tipo de organización de las ventanas.

- Configurar el formulario de inicio y la forma del cierre: seleccionar el proyecto, dar clic derecho "Properties" y aparecerá una ventana similar a la mostrada anteriormente en la figura 4.10.
- En la opción "Startup form" elegir frmLogin y en "Shutdown mode" elegir When last form closes.
- Grabar y ejecutar la aplicación. Aparecerá el formulario Login primero, ingresar el usuario y clave y se mostrará el MDI como en la figura 4.20.

Gráfico 4.20: Ejecución del formulario MDI Editor



3. Usando el Control DataGridView

El control DataGridView es el control mas usado en las aplicaciones de Windows Forms debido a su facilidad para mostrar datos y su flexibilidad para presentar información de todo tipo: texto, números, fechas, gráficos, etc. En este parte examinaremos mas a fondo dicho control sobre todo para presentar información personalizada.

El DataGridView tiene un conjunto de propiedades que inician con Allow User que permiten configurar varias características como permitir adicionar registros (AllowUserToAddRows), permitir eliminar registros (AllowUserToDeleteRows), permitir ordenar columnas (AllowUserToOrderColumns), permitir cambiar tamaño de columnas y filas (AllowUserToResizeColumns y AllowUserToResizeRows).

También tiene propiedades que permiten ajustar automáticamente el tamaño de las columnas o filas al dato mas ancho o alto, estas son AutoSizeColumnsMode y AutoSizeRowsMode, las cuales deben configurarse en AllCells.

Para indicar el modo de selección si es por celda, fila o columna se usa la propiedad SelectionMode, por ejemplo para pintar todo el registro o fila configurar su valor en FullRowSelect. Por defecto, se pueden seleccionar varias filas en la grilla, si se desea que solo una este seleccionada configurar la propiedad MultiSelect en false.

En cuanto a los eventos del DataGridView los más usados son el CellContentClick que ocurre al dar clic a una celda, también el SelectionChanged que ocurre al cambiar de fila. Si deseas formatear celdas o graficar se puede usar el evento CellFormatting y si se desea repintar las celdas por ejemplo para cambiar su apariencia usar el evento CellPaint.

Finalmente el DataGridView tiene 2 formas de trabajar: una es enlazado a datos o automático que se usa cuando la cantidad de registros no es demasiado y la otra es el modo virtual o manual que se usa para especificar solo un numero de columnas y filas deseadas. Para ver más información sobre el DataGridView consultar la referencia 24 al final del libro.

3.1. Personalizando Columnas en el DataGridView

La forma más común de trabajar es enlazar el DataGridView a un origen de datos a través de la propiedad DataSource que por defecto muestra todas las columnas del origen de datos ya que la propiedad AutoGenerateColumns esta en "True" y crea automáticamente las columnas de la grilla infiriendo para los datos lógicos una columna de tipo check y para el resto textos.

Para personalizar las columnas del DataGridView hay que configurar la propiedad AutoGenerateColumns en "False", luego enlazar la grilla y después crear las columnas personalizadas y agregarlas al DataGridView. Existen varios tipos de columnas: DataGridViewTextBoxColumn que representa datos de cadenas, números o fechas; DataGridViewCheckBoxColumn que se usa para datos lógicos; DataGridViewComboBoxColumn para mostrar listas desplegables; DataGridViewButtonColumn para mostrar botones de comandos y DataGridViewLinkColumn para ver enlaces.

Todos los tipos de columnas tienen la propiedad DataPropertyName donde hay que especificar la columna del origen de datos a mostrar, además existen otras propiedades como HeaderText para el texto de las cabeceras, Width para el ancho yDefaultCellStyle que permite configurar el estilo de la celda: alineación, formato, color de fondo, color de texto, etc.

A continuación un ejemplo de cómo personalizar columnas para la lista de productos mostrando columnas de selección (checks), columnas con colores y formato de moneda, columnas de tipo lista desplegable y columnas con botones que muestran un formulario creado dinámicamente.

Demo 60

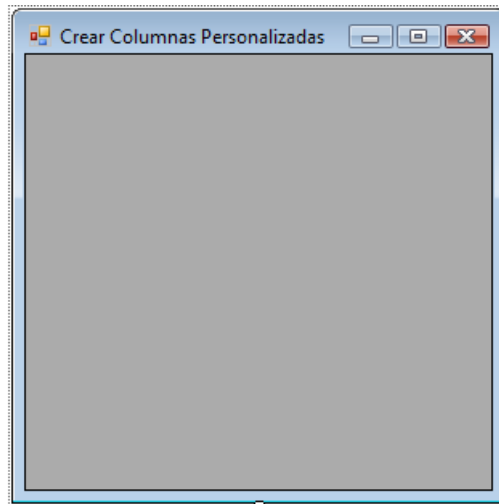
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo60.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmListaProducto.vb

- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaProducto
	Size	300, 300
	Text	Crear Columnas Personalizadas
	WindowState	Maximized
DataGridView1	Name	dgvProducto
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	Dock	Fill
	MultiSelect	False
	SelectionMode	FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 4.19:

Gráfico 4.21: Diseño del formulario Lista de Productos



- Escribir el siguiente código en el formulario:

Imports Northwind.LibBusinessEntities

Imports Northwind.LibBusinessRules

Public Class frmListaProducto

Private lobeProducto As New List(Of beProducto)

Private lobeCategoria As New List(Of beCategoria)

Private Sub ConfigurarColumnas()

Dim c0 As New DataGridViewCheckBoxColumn

With c0

.Width = 30

.HeaderText = "Sel"

End With

dgvProducto.Columns.Add(c0)

Dim c1 As New DataGridViewTextBoxColumn

With c1

.DataPropertyName = "Codigo"

.HeaderText = "ID"

.Width = 50

.ReadOnly = True

.DefaultCellStyle.Alignment = _

DataGridViewContentAlignment.MiddleRight

.DefaultCellStyle.BackColor = Color.Red

.DefaultCellStyle.ForeColor = Color.White

End With

dgvProducto.Columns.Add(c1)

Dim c2 As New DataGridViewTextBoxColumn

With c2

.DataPropertyName = "Nombre"

.HeaderText = "Nombre del Producto"

.Width = 250

.DefaultCellStyle.Alignment = _

DataGridViewContentAlignment.MiddleLeft

.DefaultCellStyle.BackColor = Color.White

.DefaultCellStyle.ForeColor = Color.Red

End With

dgvProducto.Columns.Add(c2)

Dim c3 As New DataGridViewTextBoxColumn

With c3

.DataPropertyName = "PrecioUnitario"

.HeaderText = "P.Unit"

.Width = 80

.DefaultCellStyle.Alignment = _

DataGridViewContentAlignment.MiddleRight

.DefaultCellStyle.Format = "c2"

```
.DefaultCellStyle.BackColor = Color.Red
.DefaultCellStyle.ForeColor = Color.White
End With
dgvProducto.Columns.Add(c3)
Dim c4 As New DataGridViewComboBoxColumn
With c4
    .DataSource = lobeCategoria
    .DisplayMember = "Nombre"
    .ValueMember = "Codigo"
    .DataPropertyName = "IdCategoria"
    .HeaderText = "Nombre de la Categoria"
    .Width = 250
    .DefaultCellStyle.Alignment = _
DataGridViewContentAlignment.MiddleLeft
    .DefaultCellStyle.BackColor = Color.White
    .DefaultCellStyle.ForeColor = Color.Red
End With
dgvProducto.Columns.Add(c4)
Dim c5 As New DataGridViewButtonColumn
With c5
    .HeaderText = "Comando"
    .Text = "Ver Detalle"
    .UseColumnTextForButtonValue = True
End With
dgvProducto.Columns.Add(c5)
End Sub

Private Sub CargarData(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    Dim obrProducto As New brProducto
    lobeProducto = obrProducto.Listar

    Dim obrCategoria As New brCategoria
    lobeCategoria = obrCategoria.Listar

    dgvProducto.AutoGenerateColumns = False
    dgvProducto.DataSource = lobeProducto
    ConfigurarColumnas()
End Sub
```

```
Private Sub MostrarFormBoton(ByVal sender As Object, _  
ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) _  
Handles dgvProducto.CellContentClick  
    If e.ColumnIndex = 5 Then  
        Dim frm As New Form  
        frm.StartPosition = FormStartPosition.CenterScreen  
        frm.Text = String.Format("Detalle del Producto {0}", _  
dgvProducto.CurrentRow.Cells(1).Value)  
        frm.ShowDialog()  
    End If  
End Sub  
End Class
```

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
      data source=Lduenas\MCTS;initial catalog=Northwind"/>  
  </appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.22: Ejecución del formulario Lista de Productos

Crear Columnas Personalizadas

Sel	ID	Nombre del Producto	P.Unit	Nombre de la Categoría	Comando
☐	1	Chai	S/. 10.00	Beverages	Ver Detalle
☐	2	Chang	S/. 3.00	Dairy Products	Ver Detalle
☐	3	Chicha	S/. 10.00	Beverages	Ver Detalle
☐	4	Chef Anton's Cajun Seasoning	S/. 22.00	Condiments	Ver Detalle
☐	5	Chef Anton's Gumbo Mix	S/. 21.35	Condiments	Ver Detalle
☐	6	Grandma's Boysenberry Spread	S/. 25.00	Confections	Ver Detalle
☐	7	Uncle Bob's Organic Dried Pears	S/. 30.00	Produce	Ver Detalle
☐	8	Northwoods Cranberry Sauce	S/. 40.00	Condiments	Ver Detalle
☐	9	Mishi Kobe Niku	S/. 97.00	Meat/Poultry	Ver Detalle
☐	10	Ikura	S/. 31.00	Seafood	Ver Detalle
☐	11	Queso Cabrales	S/. 21.00	Dairy Products	Ver Detalle
☐	12	Queso Manchego La Pastora	S/. 38.00	Dairy Products	Ver Detalle
☐	13	Konbu	S/. 6.00	Seafood	Ver Detalle
☐	14	Tofu	S/. 23.25	Confections	Ver Detalle
☐	15	Genen Shouyu	S/. 15.50	Condiments	Ver Detalle

3.2. Mostrando una Imagen en una Columna

Muchas veces necesitamos mostrar columnas de tipo imagen en una columna de la grilla, sin importar que sea de una base de datos o de un archivo de disco el procedimiento es similar, hay que crear una columna de tipo `DataGridViewImageColumn` y en el evento `CellFormatting` hay que asignar una imagen o bitmap a la propiedad `value` de la celda de dicha columna.

A continuación veremos un ejemplo de cómo mostrar una imagen para las categorías de productos las cuales se encuentran almacenadas en un cierto directorio y tienen como nombre el código de la categoría con extensión `jpg` y para los códigos que no se encuentran existe el archivo `No.jpg`.

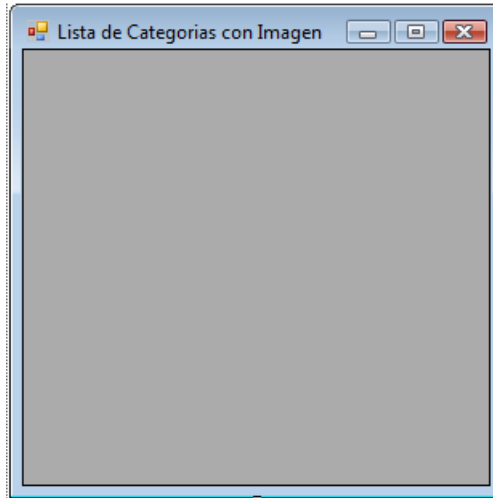
Demo 61

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo61.
- Hacer una referencia a las librerías `Northwind.LibBusinessEntities` y `Northwind.LibBusinessRules`.
- Cambiar de nombre al formulario de `form1.vb` a `frmListaCategoria.vb`
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaCategoria
	Size	300, 300
	Text	Lista de Categorías con Imagen
	WindowState	Maximized
DataGridView1	Name	dgvCategoria
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	Dock	Fill
	MultiSelect	False
	SelectionMode	FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 4.23:

Gráfico 4.23: Diseño del formulario Lista de Categorías



- Escribir el siguiente código en el formulario:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.IO
```

```
Public Class frmListaCategoria
    Private lobeCategoria As New List(Of beCategoria)
    Private Ruta As String = _
        "C:\Lduenas\NET\LibroVB2010\Imagenes\JPG\Categorias\"

    Private Sub ListarCategorias(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim obrCategoria As New brCategoria
        lobeCategoria = obrCategoria.Listar
        dgvCategoria.DataSource = lobeCategoria
        Dim c As New DataGridViewImageColumn
        With c
            .DataPropertyName = "Codigo"
            .Width = 200
            .HeaderText = "Foto"
        End With
    End Sub
End Class
```



```
End With
dgvCategoria.Columns.Add(c)
End Sub

Private Sub MostrarImagen(ByVal sender As Object, _
ByVal e As DataGridViewCellFormattingEventArgs) Handles _
dgvCategoria.CellFormatting
    If e.RowIndex > -1 Then
        If e.ColumnIndex = 2 Then
            Dim archivo As String = String.Format _
                ("{0}{1}.jpg", Ruta, e.Value)
            If Not File.Exists(archivo) Then archivo = _
                String.Format("{0}No.jpg", Ruta)
            Dim img As Image = Image.FromFile(archivo)
            e.Value = New Bitmap(img, New Size(200, 150))
        End If
    End If
End Sub
End Class
```

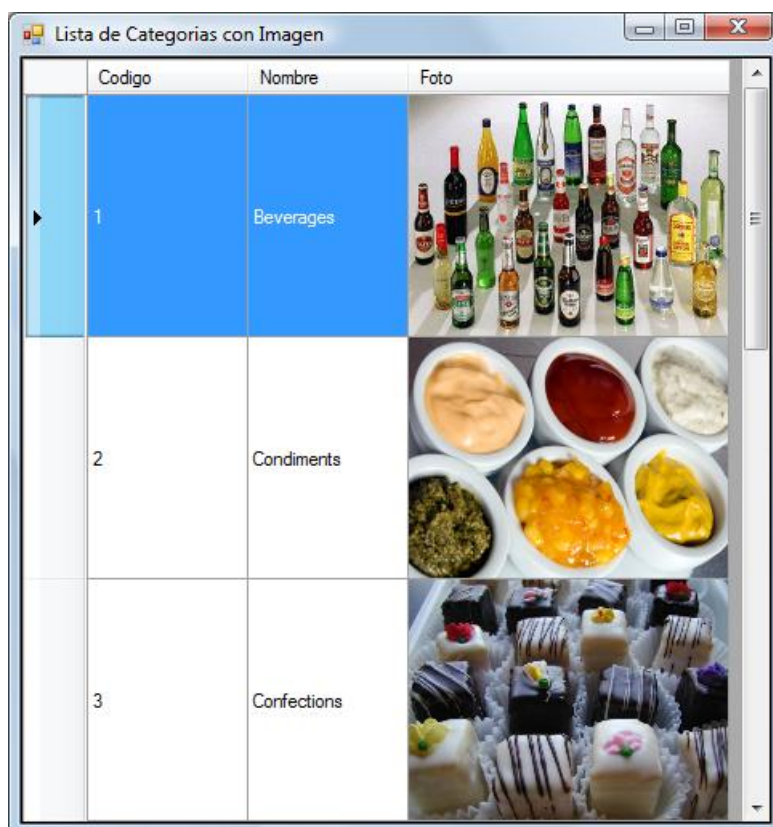
Nota: En el código anterior para configurar el tamaño de la imagen es necesario crear un Image y luego asignar a la propiedad Value de la celda el Bitmap con la imagen del tamaño deseado.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.24: Ejecución del formulario Lista de Categorías



3.3. Personalizando las Cabeceras de las Columnas

Otra tarea especial en una grilla es configurar las cabeceras para mejorar su apariencia por ejemplo degradandola o también incorporar funcionalidad de selección, filtro, etc.

Para personalizar las cabeceras del DataGridView hay que programar en el evento CellPainting para dibujar un nuevo fondo, texto o agregar un control como por ejemplo una lista desplegable para poder filtrar o un check para poder seleccionar todo.

A continuación veremos un ejemplo de cómo cambiar las cabeceras de la grilla de productos para colocar las celdas degradadas y agregar una lista desplegable con las categorías para poder filtrar los productos por categoría tan solo seleccionando la cabecera.

Demo 62

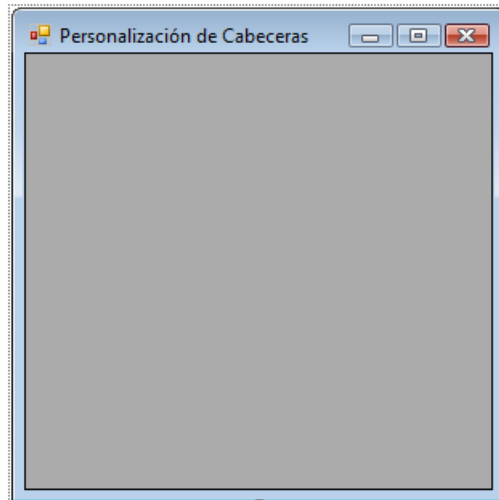
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo62.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmFiltroProducto.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmFiltroProducto
	Size	300, 300
	Text	Personalización de Cabeceras
	WindowState	Maximized
DataGridView1	Name	dgvProducto
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	Dock	Fill

	MultiSelect SelectionMode	False FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 4.25:

Gráfico 4.25: Diseño del formulario Filtro de Productos



- Escribir el siguiente código en el formulario:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.Drawing.Drawing2D 'LinearGradientBrush

Public Class frmFiltroProducto
    Private lobeProducto As New List(Of beProducto)
    Private lobeCategoria As New List(Of beCategoria)
    Private WithEvents cboCategoria As New ComboBox

    Private Sub CambiarAnchosGrilla()
        Dim asr As New System.Configuration.AppSettingsReader
        Dim strAnchos As String = asr.GetValue _
            ("AnchoGrilla", GetType("System.String"))
        If Not strAnchos.Equals("") Then
            Dim aAnchos() As String = strAnchos.Split(",")
            If aAnchos.Length > 0 Then
                For I = 0 To aAnchos.Length - 1
```

```
                dgvProducto.Columns(I).Width = aAnchos(I)
            Next
        End If
    End If
End Sub
```

```
Private Sub CargarDatos(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    Dim obrProducto As New brProducto
    lobeProducto = obrProducto.Listar
```

```
    Dim obrCategoria As New brCategoria
    lobeCategoria = obrCategoria.Listar
```

```
    Dim obeCategoria As New beCategoria
    obeCategoria.Codigo = 0
    obeCategoria.Nombre = "Todo"
    lobeCategoria.Insert(0, obeCategoria)
```

```
    With cboCategoria
        .DataSource = lobeCategoria
        .DisplayMember = "Nombre"
        .ValueMember = "Codigo"
    End With
```

```
    dgvProducto.Controls.Add(cboCategoria)
```

```
    dgvProducto.DataSource = lobeProducto
    CambiarAnchosGrilla()
End Sub
```

```
Private Sub FormatearColumnas(ByVal sender As Object, _
    ByVal e As DataGridViewCellFormattingEventArgs) Handles _
    dgvProducto.CellFormatting
```

```
    If e.RowIndex > -1 Then
        Dim Tipo As String = e.Value.GetType.ToString
        If Tipo.Equals("System.String") Then
            e.CellStyle.Alignment = _
                DataGridViewContentAlignment.MiddleLeft
        Else
```

```
e.CellStyle.Alignment = _
DataGridViewContentAlignment.MiddleRight
If Tipo.Equals("System.Decimal") Then
    e.CellStyle.Format = "n2"
End If
End If
End If
End Sub

Private Sub CambiarCabeceraGrilla(ByVal sender As Object, _
ByVal e As DataGridViewCellPaintingEventArgs) Handles _
dgvProducto.CellPainting
    'Si es la cabecera
    If e.RowIndex = -1 AndAlso e.ColumnIndex > -1 Then
        Dim Titulo As String = ""
        Dim rec As New Rectangle(e.CellBounds.X, _
e.CellBounds.Y, e.CellBounds.Width, e.CellBounds.Height)
        Dim deg As New LinearGradientBrush(rec, Color.Aqua, _
Color.Green, LinearGradientMode.BackwardDiagonal)
        e.Graphics.FillRectangle(deg, rec)
        If e.ColumnIndex = 0 Then
            Titulo = "Codigo"
        ElseIf e.ColumnIndex = 1 Then
            Titulo = "Nombre del Producto"
        ElseIf e.ColumnIndex = 2 Then
            Titulo = "Id Proveedor"
        ElseIf e.ColumnIndex = 3 Then
            cboCategoria.Location = New Point(e.CellBounds.X, _
e.CellBounds.Y)
            cboCategoria.Width = e.CellBounds.Width
        ElseIf e.ColumnIndex = 4 Then
            Titulo = "Pre Unit"
        ElseIf e.ColumnIndex = 5 Then
            Titulo = "Stock"
        End If
        e.Graphics.DrawString(Titulo, New Font("Arial", 8), _
Brushes.Blue, e.CellBounds.X + 2, e.CellBounds.Y + 2)
        e.Handled = True
    End If
End Sub
```

```
Private Function BuscaProducto(ByVal obeProducto As beProducto) As
Boolean
    Return (obeProducto.IdCategoria = cboCategoria.Selected.Value)
End Function

Private Sub FiltrarProductosxCategoria(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles _
cboCategoria.SelectionChangeCommitted
    If cboCategoria.SelectedIndex = 0 Then
        dgvProducto.DataSource = lobeProducto
    Else
        Dim pred As New Predicate(Of beProducto) _
(AddressOf BuscaProducto)
        dgvProducto.DataSource = lobeProducto.FindAll(pred)
    End If
End Sub
End Class
```

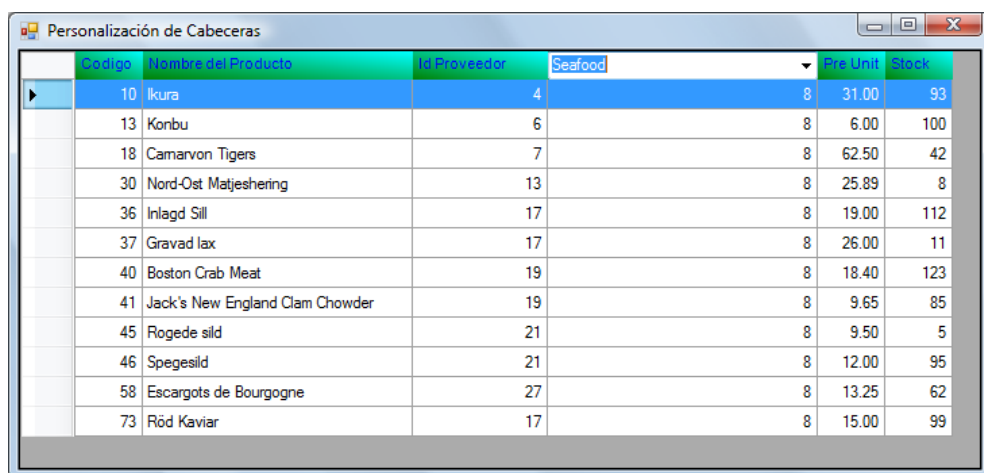
Nota: En el código anterior para configurar el tamaño de las columnas se hace dinámicamente desde el archivo de configuración mediante una clave llamada AnchoGrilla.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
<appSettings>
  <add key="conNW" value="uid=sa;pwd=123456;
    data source=Lduenas\MCTS;initial catalog=Northwind"/>
  <add key="AnchoGrilla" value="50,200,100,200,50,50"/>
</appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.26: Ejecución del formulario Filtro de Productos



	Codigo	Nombre del Producto	Id Proveedor	Seafood	Pre Unit	Stock
▶	10	Ikura	4	8	31.00	93
	13	Konbu	6	8	6.00	100
	18	Camarvon Tigers	7	8	62.50	42
	30	Nord-Ost Matjeshering	13	8	25.89	8
	36	Inlagd Sill	17	8	19.00	112
	37	Gravad lax	17	8	26.00	11
	40	Boston Crab Meat	19	8	18.40	123
	41	Jack's New England Clam Chowder	19	8	9.65	85
	45	Rogede sild	21	8	9.50	5
	46	Spegesild	21	8	12.00	95
	58	Escargots de Bourgogne	27	8	13.25	62
	73	Röd Kaviar	17	8	15.00	99

3.4. Graficando en el DataGridView

Si deseamos crear un gráfico dentro de las celdas de un DataGridView podemos usar el evento CellFormatting y construir dinámicamente el gráfico de acuerdo al valor de una celda numérica la cual será representada por una imagen, por ejemplo una barra que represente los precios de los productos como se demuestra en el siguiente ejemplo.

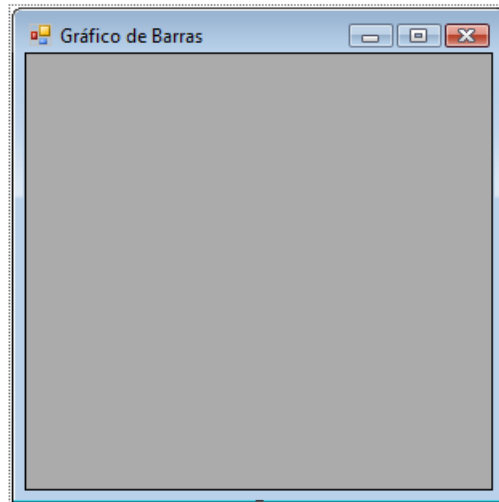
Demo 63

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo63.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmGraficoBarras.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmGraficoBarras
	Size	300, 300
	Text	Gráfico de Barras
	WindowState	Maximized
DataGridView1	Name	dgvProducto
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	Dock	Fill
	MultiSelect	False
	SelectionMode	FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 4.27:

Gráfico 4.27: Diseño del formulario Gráfico de Barras



- Escribir el siguiente código en el formulario:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.Drawing.Drawing2D 'LinearGradientBrush

Public Class frmGraficoBarras
    Private lobeProducto As New List(Of beProducto)

    Private Sub ListarProductos(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim obrProducto As New brProducto
        lobeProducto = obrProducto.Listar
        dgvProducto.AutoGenerateColumns = False
        dgvProducto.DataSource = lobeProducto
        Dim c1 As New DataGridViewTextBoxColumn
        With c1
            .DataPropertyName = "Nombre"
            .HeaderText = "Nombre del Producto"
            .Width = 200
            .DefaultCellStyle.Alignment = _
                DataGridViewContentAlignment.MiddleLeft
        End With
        dgvProducto.Columns.Add(c1)
```

```
Dim c2 As New DataGridViewImageColumn
With c2
    .DataPropertyName = "PrecioUnitario"
    .HeaderText = "Gráfico de Precios de Productos"
    .Width = 600
End With
dgvProducto.Columns.Add(c2)
End Sub

Private Function GenerarColorAzar() As Color
    System.Threading.Thread.Sleep(10)
    Dim oAzar As New Random
    Dim R As Integer = oAzar.Next(255)
    Dim G As Integer = oAzar.Next(255)
    Dim B As Integer = oAzar.Next(255)
    Dim X As Color = Color.FromArgb(R, G, B)
    Return (X)
End Function

Private Sub CrearGraficoPrecios(ByVal sender As Object, _
ByVal e As DataGridViewCellFormattingEventArgs) Handles _
dgvProducto.CellFormatting
    If e.RowIndex > -1 Then
        If e.ColumnIndex = 1 Then
            Dim bmp As New Bitmap(570, 30)
            Dim grafico As Graphics = Graphics.FromImage(bmp)
            If e.Value IsNot Nothing Then
                Dim Precio As Decimal = Decimal.Parse(e.Value)
                grafico.DrawString(String.Format _
("{0:n2}", Precio).PadLeft(6, " "), _
                New Font("Courier New", 10), Brushes.Blue, 0, 5)
                Dim rec As New Rectangle(70, 5, Precio, 20)
                Dim deg As New LinearGradientBrush(rec, _
GenerarColorAzar, GenerarColorAzar, _
LinearGradientMode.BackwardDiagonal)
                grafico.FillRectangle(deg, rec)
                e.Value = bmp
            Else
                e.Value = Nothing
            End If
        End If
    End If
```

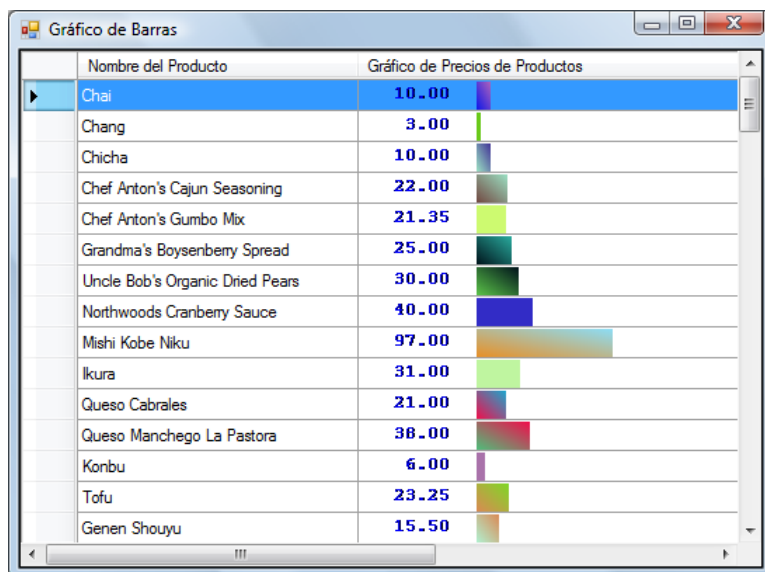
```
End If
End If
End Sub
End Class
```

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
    <add key="AnchoGrilla" value="50,200,100,200,50,50"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.28: Ejecución del formulario Gráfico de Barras



3.5. Paginando en un DataGridView

El DataGridView por defecto trabaja en forma enlazada, es decir muestra todas las columnas y filas del origen de datos que se configure, en los temas anteriores vimos como mostrar solo las columnas deseadas, pero no vimos como limitar el número de filas que se visualizan sobre todo si son miles de registros.

Si se enlaza un DataGridView a muchos registros se tendrá problemas de performance ya que NET Framework se demorará en pintar la gran cantidad de celdas, para esto es conveniente trabajar en modo Virtual o modo manual el cual consiste en crear uno mismo las columnas y filas a mostrar.

Lo primero que hay que hacer para paginar en el DataGridView es configurar la propiedad AutoGenerateColumns en "False", luego configurar la propiedad VirtualMode en "True", después crear las columnas a mostrar y configurar la propiedad RowCount en el número de filas a visualizar.

Lo mas importante es programar en el evento CellValueNeeded el valor que se va a mostrar en cada celda y para que se actualize una página es necesario llamar al método Refresh.

A continuación un ejemplo de paginación de registros para los productos de tal forma que se muestren de 20 en 20 y podamos ver la primera página, ir a la página anterior, ir a la página siguiente y ver la ultima página.

Demo 64

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo64.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmPaginacion Producto.vb

- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name Size Text WindowState	frm PaginacionProductos 300, 300 Paginación de Productos Maximized
DataGridView1	Name AllowUserToAddRows AllowUserToDeleteRows Anchor Location MultiSelect SelectionMode Size	dgvProducto False False Top, Bottom, Left, Right 3,3 False FullRowSelect 288, 241
Button1	Name Anchor Cursor Location Size Text	btnPrimero Bottom, Right Hand 61, 250 28, 23 <<
Button2	Name Anchor Cursor Location Size Text	btnAnterior Bottom, Right Hand 90, 250 28, 23 <
TextBox1	Name Anchor Location ReadOnly Size Text TextAlign	txtPagina Bottom, Right 119, 253 True 64, 20 Center
Button3	Name Anchor Cursor Location Size Text	btnSiguiente Bottom, Right Hand 184, 250 28, 23 >
Button4	Name	btnUltimo

	Anchor	Bottom, Right
	Cursor	Hand
	Location	213, 250
	Size	28, 23
	Text	>>

- El diseño del formulario debe quedar similar al gráfico 4.29:

Gráfico 4.29: Diseño del formulario Paginación de Productos



- Escribir el siguiente código en el formulario:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Public Class frmPaginacionProducto
```

```
    Private lobeProducto As New List(Of beProducto)
```

```
    Private NumRegPag As Integer = 20
```

```
    Private IndicePag As Integer = 0
```

```
    Private TotalPag As Integer = 0
```

```
    Private IndiceFila As Integer = 0
```

```
    Private Sub CrearColumnas()
```

```
        dgvProducto.Columns.Add("Codigo", "Codigo")
```

```
        dgvProducto.Columns.Add("Nombre", "Nombre")
```

```
        dgvProducto.Columns.Add("PrecioUnitario", "PreUnit")
```

```
dgvProducto.Columns.Add("Stock", "Stock")
End Sub

Private Sub MostrarPagina()
    txtPagina.Text = String.Format("{0} de {1}", _
        IndicePag + 1, TotalPag)
End Sub

Private Sub ListarProductos(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    Dim obrProducto As New brProducto
    lobeProducto = obrProducto.Listar
    If lobeProducto.Count Mod NumRegPag = 0 Then
        TotalPag = lobeProducto.Count / NumRegPag
    Else
        TotalPag = (lobeProducto.Count / NumRegPag) + 1
    End If
    MostrarPagina()

    dgvProducto.AutoGenerateColumns = False
    dgvProducto.VirtualMode = True
    CrearColumnas()
    dgvProducto.RowCount = NumRegPag
End Sub

Private Sub EscribirCelda(ByVal sender As Object, _
    ByVal e As DataGridViewCellValueEventArgs) Handles _
    dgvProducto.CellValueNeeded
    IndiceFila = NumRegPag * IndicePag + e.RowIndex
    If IndiceFila < lobeProducto.Count Then
        If e.ColumnIndex = 0 Then
            e.Value = lobeProducto(IndiceFila).Codigo
        ElseIf e.ColumnIndex = 1 Then
            e.Value = lobeProducto(IndiceFila).Nombre
        ElseIf e.ColumnIndex = 2 Then
            e.Value = lobeProducto(IndiceFila).PrecioUnitario
        ElseIf e.ColumnIndex = 3 Then
            e.Value = lobeProducto(IndiceFila).Stock
        End If
    End If
End Sub
```


End Sub

```
Private Sub IrPrimeraPagina(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnPrimero.Click  
    IndicePag = 0  
    dgvProducto.Refresh()  
    MostrarPagina()  
End Sub
```

```
Private Sub IrPaginaAnterior(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAnterior.Click  
    If IndicePag > 0 Then IndicePag -= 1  
    dgvProducto.Refresh()  
    MostrarPagina()  
End Sub
```

```
Private Sub IrPaginaSiguiente(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnSiguiente.Click  
    If IndicePag < TotalPag - 1 Then IndicePag += 1  
    dgvProducto.Refresh()  
    MostrarPagina()  
End Sub
```

```
Private Sub IrUltimaPagina(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnUltimo.Click  
    IndicePag = TotalPag - 1  
    dgvProducto.Refresh()  
    MostrarPagina()  
End Sub  
End Class
```

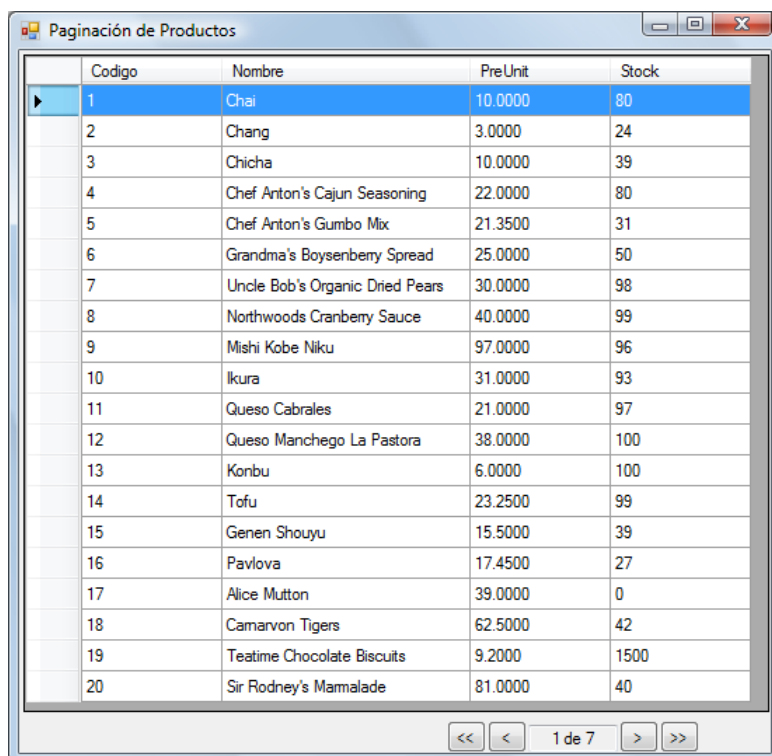
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;
```

```
data source=Lduenas\MCTS;initial catalog=Northwind"/>
<add key="AnchoGrilla" value="50,200,100,200,50,50"/>
</appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.30: Ejecución del formulario Paginación de Productos



	Codigo	Nombre	PreUnit	Stock
▶	1	Chai	10.0000	80
	2	Chang	3.0000	24
	3	Chicha	10.0000	39
	4	Chef Anton's Cajun Seasoning	22.0000	80
	5	Chef Anton's Gumbo Mix	21.3500	31
	6	Grandma's Boysenberry Spread	25.0000	50
	7	Uncle Bob's Organic Dried Pears	30.0000	98
	8	Northwoods Cranberry Sauce	40.0000	99
	9	Mishi Kobe Niku	97.0000	96
	10	Ikura	31.0000	93
	11	Queso Cabrales	21.0000	97
	12	Queso Manchego La Pastora	38.0000	100
	13	Konbu	6.0000	100
	14	Tofu	23.2500	99
	15	Genen Shouyu	15.5000	39
	16	Pavlova	17.4500	27
	17	Alice Mutton	39.0000	0
	18	Camarvon Tigers	62.5000	42
	19	Teatime Chocolate Biscuits	9.2000	1500
	20	Sir Rodney's Marmalade	81.0000	40

4. Creando una Biblioteca de Controles Windows

Para reusar el código se crea una Biblioteca de Clases, pero si queremos reusar Controles debemos crear una Biblioteca de Controles Windows que nos permitan ahorrar tiempo en tareas comunes como validación en controles, botones comunes, diálogos comunes, grillas de consulta, grillas paginadas, etc.

Los controles que podemos crear en Windows Forms se clasifican en 3: controles extendidos, controles de usuario y controles personalizados. En todos ellos podemos crear propiedades, métodos, eventos, etc, también podemos sobre escribir el comportamiento de ciertos eventos.

La diferencia entre los 3 es que el primero (control extendido) solo representa un control al que hereda, el segundo (control de usuario) se compone de 2 o mas controles Windows y el tercero (control personalizado) se tiene que graficar o dibujar. Para obtener más información de cómo Desarrollar controles personalizados de formularios Windows Forms con .NET Framework vea la referencia 25 al final del libro.

4.1. Creando Controles Extendidos

Los controles extendidos o ampliados son aquéllos que heredan de un cierta clase de System.Windows.Forms, como por ejemplo de la clase TextBox para crear un control que solo acepte números, o de la clase DataGridView para que la grilla tenga la configuración de solo lectura o de la clase ListView para crear una propiedad DataSource y poder enlazarlo a un origen de datos.

Cada vez que se quiera sobre escribir el comportamiento de un control o mejorar su funcionalidad lo que debemos hacer es crear un control que herede de dicha clase.

A continuación crearemos una Librería de Controles para Windows y agregaremos un par de clases: la primera que herede de un TextBox para crear el control Número y el segundo que herede del DataGridView para

crear el control GrillaConsulta, ambos usaran el prefijo "wic" que indica que son "Wi"ndows "C"ontrols.

Demo General.LibControlsWinForms

- Del menú "File", seleccionar "New Project" y luego "Windows Forms Control Library".
- Escribir como nombre físico al proyecto: General.LibControlsWinForms.
- Eliminar el control que aparece por defecto dando clic derecho y seleccionando "Eliminar".

Crear el Control Número

- Agregar una clase llamada "wicNumero".
- Escribir el siguiente código para la clase wicNumero:

```
Public Class wicNumero
    Inherits System.Windows.Forms.TextBox
    Public Property SgteControl() As Control

    Protected Overrides Sub OnKeyPress _
        (ByVal e As System.Windows.Forms.KeyPressEventArgs)
        If e.KeyChar = ControlChars.Cr Then
            If SgteControl IsNot Nothing Then _
                SgteControl.Focus()
        Else
            e.Handled = Not (Char.IsDigit(e.KeyChar) Or _
                e.KeyChar = ControlChars.Back)
        End If
    End Sub
End Class
```

Nota: En el código anterior se ha creado una propiedad SgteControl que permite especificar a que control vamos a avanzar al pulsar Enter sobre el número. Además se ha sobre escrito el evento OnKeyPress para permitir ingresar solo digitos y BackSpace.

Crear el Control Grilla Consulta

- Agregar una nueva clase llamada "wicGrillaConsulta".
- Escribir el siguiente código para la clase wicGrillaConsulta:

```
Imports System.Windows.Forms
```

```
Public Class wicGrillaConsulta  
    Inherits DataGridView
```

```
    Public Sub New()  
        Me.AllowUserToAddRows = False  
        Me.AllowUserToDeleteRows = False  
        Me.ReadOnly = True  
        Me.SelectionMode = _  
            DataGridViewSelectionMode.FullRowSelect  
    End Sub
```

```
    Private Sub FormatearColumnasGrilla(ByVal sender As Object, _  
        ByVal e As DataGridViewCellFormattingEventArgs) _  
        Handles Me.CellFormatting  
        If e.RowIndex > -1 Then  
            Dim Tipo As String = e.Value.GetType.ToString  
            If Tipo.Equals("System.String") Then  
                e.CellStyle.Alignment = _  
                    DataGridViewContentAlignment.MiddleLeft  
            Else  
                e.CellStyle.Alignment = _  
                    DataGridViewContentAlignment.MiddleRight  
                If Tipo.Equals("System.Decimal") Then  
                    e.CellStyle.Format = "n2"  
                End If  
            End If  
        End If  
    End Sub  
End Class
```

Nota: En el código anterior se ha modificado el constructor para que cada vez que se cree una instancia de la grilla se configure las propiedades para que sea solo lectura. Además en el evento CellFormatting se ha

programado para que todas las columnas de texto se alineen a la izquierda y el resto: números y fechas se alineen a la derecha, además las columnas que tienen decimales aparecerán con 2 decimales por defecto.

4.2. Creando Controles de Usuario

Los controles de usuario o compuestos son aquéllos que se componen de 2 o más controles de Windows a los cuales se les conoce como controles constituyentes, los controles de usuario heredan de la clase UserControl de System. Windows.Forms.

Este control es el más fácil de crear porque se dispone del diseñador de Windows Forms para arrastrar los controles que van a formar el control de usuario; casi siempre es necesario crear propiedades al control para exponer las propiedades de sus controles constituyentes.

A continuación vamos a agregar 3 controles de usuario a nuestra Librería de Controles creada anteriormente: el primero wicBotonesMante que permite tener los botones de Nuevo, Adicionar, Actualizar y Eliminar de un mantenimiento; el segundo wicDialogo que permite seleccionar un archivo a abrir o guardar y configurar un texto con la ruta seleccionada; finalmente el tercer control y el más importante wicGrillaPaginada que permite realizar paginación desconectada mediante listas de objetos.

Crear el Control Botones de Mantenimiento

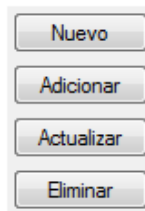
- Agregar un control de usuario: del menú "Project" seleccionar "Add User Control"
- Escribir como nombre del control: "wicBotonesMante".
- Agregar y configurar controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
UserControl	Name	wicBotonesMante
	Size	80, 118
Button1	Name	btnNuevo

	Cursor Location Size Text	Hand 3,3 75, 23 Nuevo
Button2	Name Cursor Location Size Text	btnAdicionar Hand 3, 32 75, 50 Adicionar
Button3	Name Cursor Location Size Text	btnActualizar Hand 3, 61 75, 23 Actualizar
Button4	Name Cursor Location Size Text	btnEliminar Hand 3, 90 75, 23 Eliminar

- El diseño del control debe quedar similar al gráfico 4.31:

Gráfico 4.31: Diseño del control de Usuario wicBotonesMante



- Escribir el siguiente código para el control wicBotonesMante:

```
Public Class wicBotonesMante
    Public Event Elegir(ByVal N As Integer)

    Private Sub Nuevo(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnNuevo.Click
        RaiseEvent Elegir(1)
    End Sub
```

```
Private Sub Adicionar(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAdicionar.Click  
    RaiseEvent Elegir(2)  
End Sub
```

```
Private Sub Actualizar(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnActualizar.Click  
    RaiseEvent Elegir(3)  
End Sub
```

```
Private Sub Eliminar(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnEliminar.Click  
    RaiseEvent Elegir(4)  
End Sub  
End Class
```

Nota: En el código anterior se ha creado un evento llamado “Elegir” el cual se desencadena al dar clic sobre cada botón, para lo cual se pasa un número que indica que botón es el que desencadenó el evento.

Crear el Control Diálogo

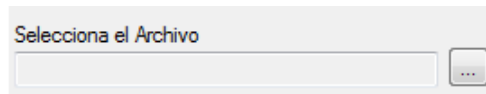
- Agregar un control de usuario: del menú “Project” seleccionar “Add User Control”
- Escribir como nombre del control: “wicDialogo”.
- Agregar y configurar controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
UserControl	Name Size	wicDialogo 270, 52
Label1	Name AutoSize Location Text	lblArchivo True 0, 9 Selecciona el Archivo
TextBox1	Name Location ReadOnly Size	txtArchivo 3, 25 True 235,20

Button1	Name	btnMostrarDialogo
	Cursor	Hand
	Location	243, 22
	Size	24, 23
	Text	...

- El diseño del control debe quedar similar al gráfico 4.32:

Gráfico 4.32: Diseño del control de Usuario wicDialogo



- Escribir el siguiente código para el control wicDialogo:

```
Public Class wicDialogo
    Private _TipoDialogo As TiposDialogo
    Private _Filtro As String
    Public Event SeleccionarArchivo()

    Public Enum TiposDialogo
        DialogoAbrir = 0
        DialogoGuardar = 1
    End Enum

    Public Property TituloTexto() As String
        Get
            Return (lblArchivo.Text)
        End Get
        Set(ByVal value As String)
            lblArchivo.Text = value
        End Set
    End Property

    Public Property TituloColor() As Color
        Get
            Return (lblArchivo.ForeColor)
        End Get
        Set(ByVal value As Color)
            lblArchivo.ForeColor = value
        End Set
    End Property
End Class
```

```
End Set
End Property

Public Property TipoDialogo() As TiposDialogo
    Get
        Return (_TipoDialogo)
    End Get
    Set(ByVal value As TiposDialogo)
        _TipoDialogo = value
    End Set
End Property

Public Property Filtro() As String
    Get
        Return (_Filtro)
    End Get
    Set(ByVal value As String)
        _Filtro = value
    End Set
End Property

Public Property NombreArchivo() As String
    Get
        Return (txtArchivo.Text)
    End Get
    Set(ByVal value As String)
        txtArchivo.Text = value
    End Set
End Property

Private Sub MostrarDialogo(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnMostrarDialogo.Click
    Dim fd As OpenFileDialog
    If _TipoDialogo = TiposDialogo.DialogoAbrir Then
        fd = New OpenFileDialog
    Else
        fd = New SaveFileDialog
    End If
    fd.Title = lblArchivo.Text
    fd.Filter = _Filtro
```

```

    If fd.ShowDialog = DialogResult.OK Then
        txtArchivo.Text = fd.FileName
        RaiseEvent SeleccionarArchivo()
    End If
End Sub
End Class

```

Nota: En el código anterior se ha creado una propiedad enumerada TipoDialogo que especifica si lo que se va a mostrar es el dialogo de abrir o guardar. Además se ha creado un evento que se dispara cuando se elige un archivo del diálogo.

Crear el Control Grilla Paginada

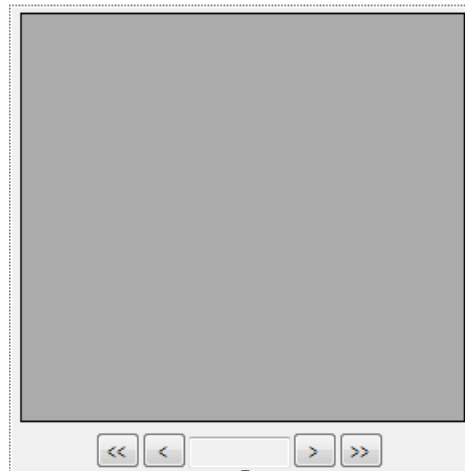
- Agregar un control de usuario: del menú "Project" seleccionar "Add User Control"
- Escribir como nombre del control: "wicGrillaPaginada".
- Agregar y configurar controles de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
UserControl	Name MinimumSize Size	wicGrillaPaginada 290, 290 290, 290
DataGridView1	Name AllowUserToAddRows AllowUserToDeleteRows Anchor Location MultiSelect SelectionMode Size	dgvProducto False False Top, Bottom, Left, Right 5, 3 False FullRowSelect 278, 255
Button1	Name Anchor Cursor Location Size Text	btnPrimero Bottom, Right Hand 52, 264 28, 23 <<
Button2	Name	btnAnterior

	Anchor Cursor Location Size Text	Bottom, Right Hand 81, 264 28, 23 <
TextBox1	Name Anchor Location ReadOnly Size Text TextAlign	txtPagina Bottom, Right 110, 267 True 64, 20 Center
Button3	Name Anchor Cursor Location Size Text	btnSiguiente Bottom, Right Hand 175, 264 28, 23 >
Button4	Name Anchor Cursor Location Size Text	btnUltimo Bottom, Right Hand 204, 264 28, 23 >>

- El diseño del control debe quedar similar al gráfico 4.33:

Gráfico 4.33: Diseño del control de Usuario wicGrillaPaginada



- Escribir el siguiente código para el control wicGrillaPaginada:

```
Imports System.Windows.Forms
```

```
Imports System.Reflection
```

```
Public Class wicGrillaPaginada
```

```
    Private _DataSource As Object
```

```
    Private _NumRegPag As Integer
```

```
    Private IndicePag As Integer = 0
```

```
    Private TotalPag As Integer = 0
```

```
    Private IndiceFila As Integer = 0
```

```
    Public Property DataSource() As Object
```

```
        Get
```

```
            Return (_DataSource)
```

```
        End Get
```

```
        Set(ByVal value As Object)
```

```
            _DataSource = value
```

```
            If _DataSource IsNot Nothing Then
```

```
                If _DataSource.Count Mod _NumRegPag = 0 Then
```

```
                    TotalPag = _DataSource.Count \ _NumRegPag
```

```
                Else
```

```
                    TotalPag = (_DataSource.Count \ _NumRegPag) + 1
```

```
                End If
```

```
MostrarPagina()

    dgv.AutoGenerateColumns = False
    dgv.VirtualMode = True
    CrearColumnas()
    dgv.RowCount = _NumRegPag
End If
End Set
End Property

Public Property NumRegPag() As Integer
    Get
        Return (_NumRegPag)
    End Get
    Set(ByVal value As Integer)
        _NumRegPag = value
    End Set
End Property

Private Sub CrearColumnas()
    Dim Propiedades() As PropertyInfo = _
        _DataSource(0).GetType.GetProperties
    For Each Propiedad In Propiedades
        dgv.Columns.Add(Propiedad.Name, Propiedad.Name)
    Next
End Sub

Private Sub MostrarPagina()
    txtPagina.Text = String.Format("{0} de {1}", _
        IndicePag + 1, TotalPag)
End Sub

Private Sub EscribirCelda(ByVal sender As Object, _
    ByVal e As DataGridViewCellValueEventArgs) Handles _
    dgv.CellValueNeeded
    IndiceFila = _NumRegPag * IndicePag + e.RowIndex
    If IndiceFila < _DataSource.Count Then
        Dim campo As String = dgv.Columns(e.ColumnIndex).Name
        e.Value = _DataSource(IndiceFila).GetType. _
            GetProperty(campo).GetValue(_DataSource(IndiceFila), Nothing)
```

```
End If
End Sub

Private Sub IrPrimeraPagina(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnPrimero.Click
    IndicePag = 0
    dgv.Refresh()
    MostrarPagina()
End Sub

Private Sub IrAnteriorPagina(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnAnterior.Click
    If IndicePag > 0 Then IndicePag -= 1
    dgv.Refresh()
    MostrarPagina()
End Sub

Private Sub IrSiguientePagina(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnSiguiente.Click
    If IndicePag < TotalPag - 1 Then IndicePag += 1
    dgv.Refresh()
    MostrarPagina()
End Sub

Private Sub btnUltimo_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnUltimo.Click
    IndicePag = TotalPag - 1
    dgv.Refresh()
    MostrarPagina()
End Sub
End Class
```

Nota: En el código anterior se ha creado 2 propiedades: una para configurar el origen de datos (lista de objetos) y otra para especificar el número de registros por página a visualizar. También se usa Reflection para obtener las propiedades del tipo de objeto de la lista de objetos.

Creando Controles Personalizados

Un control personalizado (Custom Control) es un control que se tiene que crear desde cero mediante gráficos, éste hereda de la clase Control del espacio de nombres System.Windows.Forms.

Crear este tipo de controles no es tan fácil ya que se tiene que realizar por código mediante gráficos y se usa justamente para crear pantallas gráficas.

A continuación agregaremos a la librería de controles creada el control Logo que dibuja un rectángulo degradado en pantalla con un mensaje y finalmente probaremos en una aplicación de Windows (Demo 65) todos los controles anteriormente creados.

Crear el Control Logo

- Agregar un control personalizado: del menú "Project" seleccionar "Add New Item" y luego elegir "Custom Control".
- Escribir como nombre del control: "wicLogo".
- Escribir el siguiente código para el control wicLogo:

```
Imports System.Drawing.Drawing2D
```

```
Public Class wicLogo
```

```
    Public Property TituloTexto() As String
```

```
    Public Property TituloFuente() As Font
```

```
    Public Property TituloColor() As Color
```

```
    Public Property TituloPosicion() As New Point(0, 0)
```

```
    Public Property FondoColor1() As Color
```

```
    Public Property FondoColor2() As Color
```

```
    Protected Overrides Sub OnPaint _
```

```
    (ByVal e As System.Windows.Forms.PaintEventArgs)
```

```
        Dim rec As New Rectangle(0, 0, Me.Width, Me.Height)
```

```
        Dim deg As New LinearGradientBrush(rec, FondoColor1, _  
        FondoColor2, LinearGradientMode.BackwardDiagonal)
```

```
        e.Graphics.FillRectangle(deg, rec)
```

```
        e.Graphics.DrawString(TituloTexto, TituloFuente, _  
        New SolidBrush(TituloColor), TituloPosicion)
```


End Sub

End Class

- Finalmente, del menú "Build" seleccionar "Build General.LibControls WinForms" para crear la dll con todos los controles creados.

Demo 65

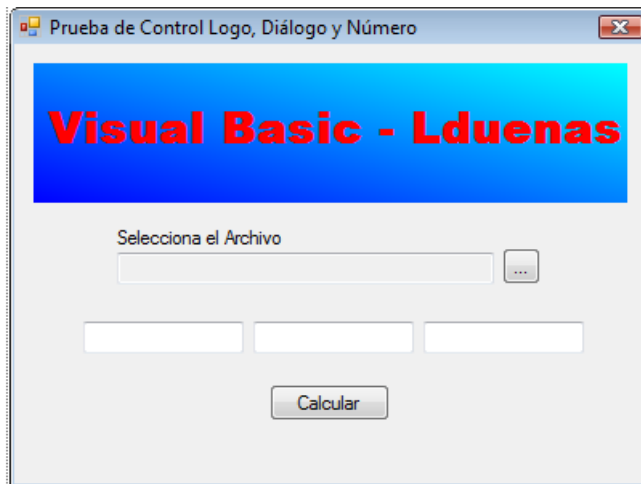
- Del menú "File", seleccionar "Add", "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo65.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmPrueba1.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmPrueba1
	FormBorderStyle	FixedSingle
	Size	400, 300
	MaximizeBox	False
	MinimizeBox	False
	Text	Prueba de Control Logo, Diálogo y Número
	StartPosition	CenterScreen
WicLogo1	Name	Logo
	FondoColor1	Aqua
	FondoColor1	Blue
	Location	12, 12
	Size	370, 87
	TituloColor	Red
	TituloFuente	Arial Black, 20.25pt
	TituloPosicion	5, 20
WicDialogo1	TituloTexto	Visual Basic – Lduenas
	Name	DialogoAbrir
	Location	61, 105
	Size	270, 52

WinNumero1	Name Location SgteControl Size	Numero1 43, 173 Numero2 100, 20
WinNumero2	Name Location SgteControl Size	Numero2 149, 173 Numero3 100, 20
WinNumero3	Name Location SgteControl Size	Numero3 255, 173 btnCalcular 100, 20
Button1	Name Cursor Location Text	btnCalcular Hand 159, 212 Calcular

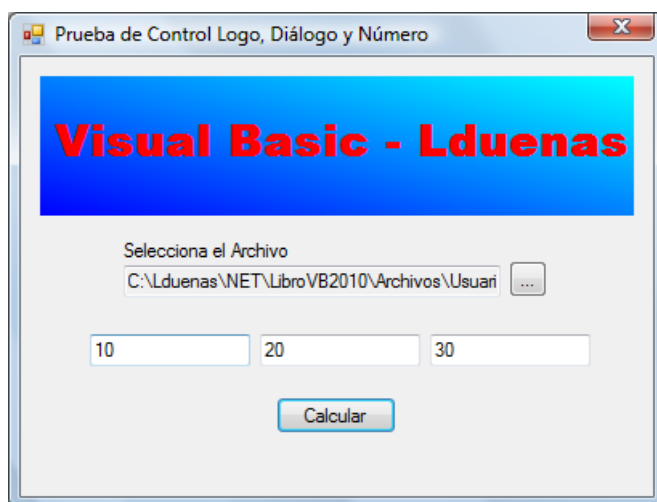
- El diseño del formulario debe quedar similar al gráfico 4.34:

Gráfico 4.34: Diseño del formulario Prueba1



- Configurar la aplicación Windows como el que inicia: clic derecho al proyecto y seleccionar "Set as StartUp Project".
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.35: Ejecución del formulario Prueba1

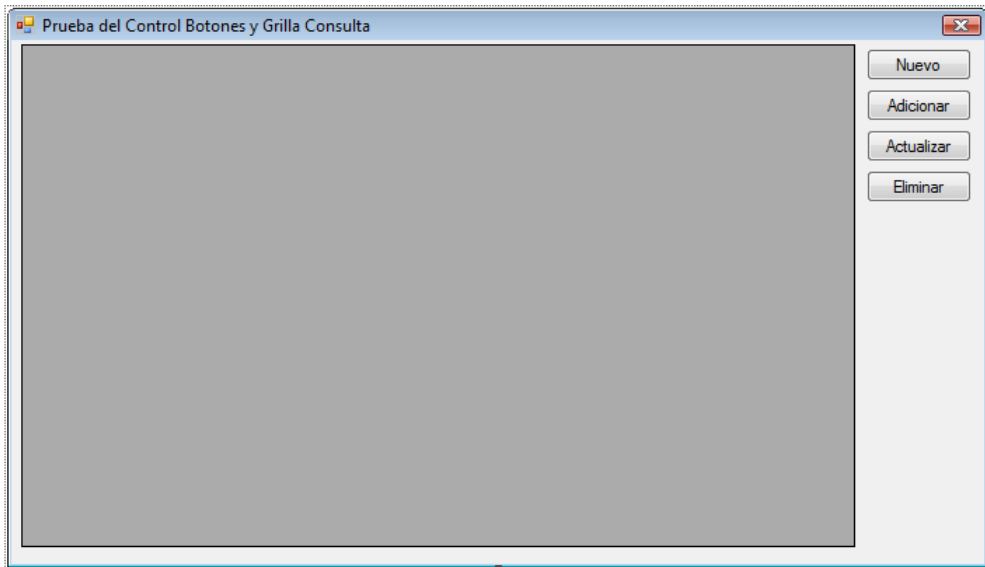


- Agregar otro formulario a la aplicación y llamarle frmPrueba2.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmPrueba2
	FormBorderStyle	FixedSingle
	Size	700, 400
	MaximizeBox	False
	MinimizeBox	False
	Text	Prueba del Control
	StartPosition	Botones y Grilla Consulta CenterScreen
WicGrillaConsulta1	Name	gcProducto
	Location	7, 3
	Size	595, 359
WicBotonesMante1	Name	bmProducto
	Location	607, 3
	Size	80, 118

- El diseño del formulario debe quedar similar al gráfico 4.36:

Gráfico 4.36: Diseño del formulario Prueba2



- Escribir el siguiente código para el formulario de prueba 2:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Public Class frmPrueba2
```

```
    Private Sub ListarProductos(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim obrProducto As New brProducto
        Dim lobeProducto As List(Of beProducto) = obrProducto.Listar
        gcProducto.DataSource = lobeProducto
    End Sub
```

```
End Class
```

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
```

```
<appSettings>
  <add key="conNW" value="uid=sa;pwd=123456;
    data source=Lduenas\MCTS;initial catalog=Northwind"/>
</appSettings>
</configuration>
```

- Configurar como formulario de inicio a frmPrueba2.
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.37: Ejecución del formulario Prueba2

Codigo	Nombre	IdProveedor	IdCategoria	PrecioUnitario	Stock
1	Chai	2	1	10.00	80
2	Chang	7	4	3.00	24
3	Chicha	10	1	10.00	39
4	Chef Anton's Cajun Seasoning	2	2	22.00	80
5	Chef Anton's Gumbo Mix	2	2	21.35	31
6	Grandma's Boysenberry Spread	2	3	25.00	50
7	Uncle Bob's Organic Dried Pears	3	7	30.00	98
8	Northwoods Cranberry Sauce	3	2	40.00	99
9	Mishi Kobe Niku	4	6	97.00	96
10	Ikura	4	8	31.00	93
11	Queso Cabrales	5	4	21.00	97
12	Queso Manchego La Pastora	5	4	38.00	100
13	Konbu	6	8	6.00	100
14	Tofu	6	3	23.25	99
15	Genen Shouyu	6	2	15.50	39
16	Pavlova	7	3	17.45	27
17	Alice Mutton	7	6	39.00	0

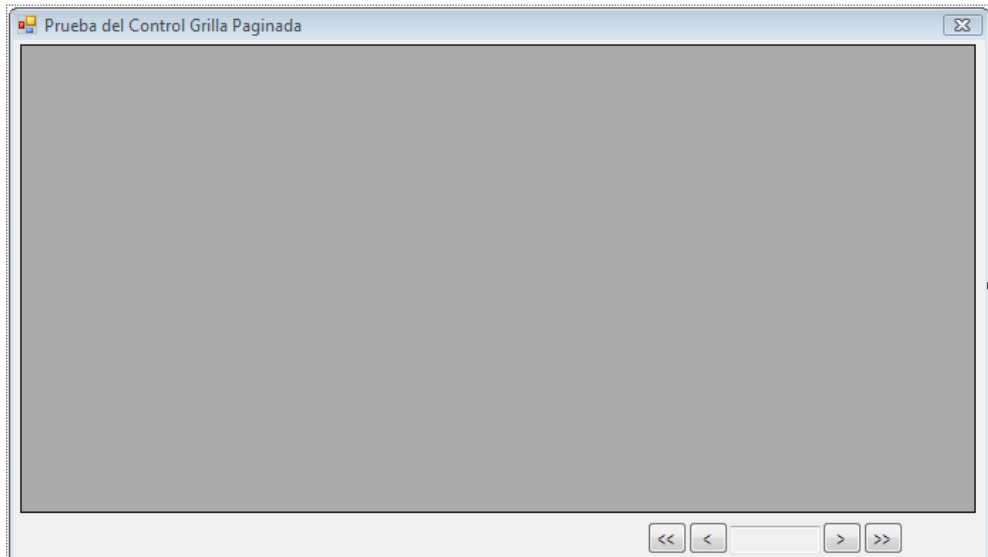
- Agregar otro formulario a la aplicación y llamarle frmPrueba3.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmPrueba3
	FormBorderStyle	FixedSingle
	Size	700, 400
	MaximizeBox	False
	MinimizeBox	False
	Text	Prueba del Control Grilla Paginada
	StartPosition	CenterScreen

WicGrillaPaginada1	Name Dock	gpProducto Fill

- El diseño del formulario debe quedar similar al gráfico 4.38:

Gráfico 4.38: Diseño del formulario Prueba3



- Escribir el siguiente código para el formulario de prueba 3:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Public Class frmPrueba3
```

```
    Private Sub ListarProductos(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Dim obrProducto As New brProducto
```

```
        Dim lobeProducto As List(Of brProducto) = obrProducto.Listar
```

```
        gpProducto.NumRegPag = 20
```

```
        gpProducto.DataSource = lobeProducto
```

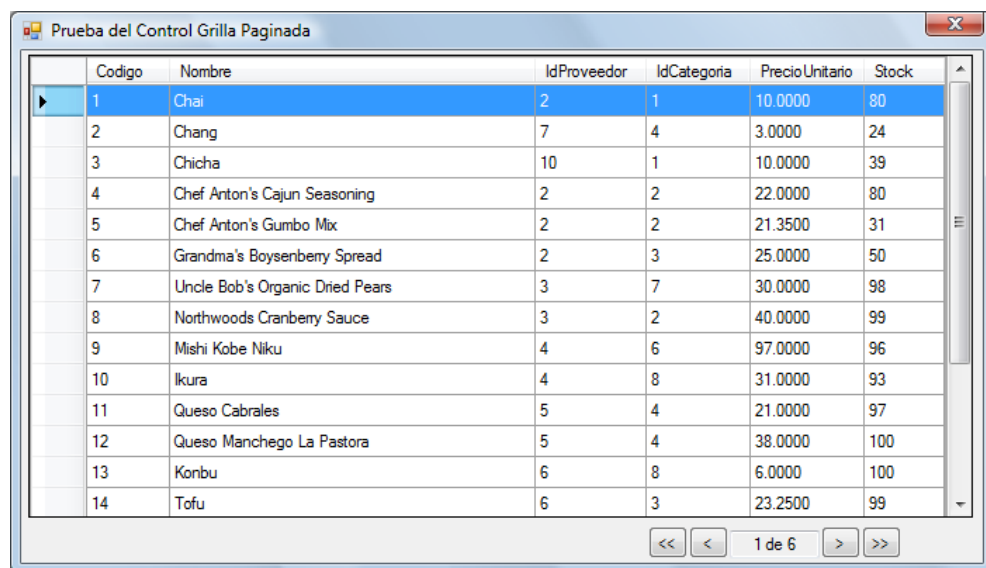
```
    End Sub
```

```
End Class
```

Importante: Es necesario que primero se llene la propiedad NumRegPag antes que la propiedad DataSource, ya que al establecer la propiedad DataSource se configura el numero de registros a visualizar.

- Configurar como formulario de inicio a frmPrueba3.
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.39: Ejecución del formulario Prueba3



	Codigo	Nombre	IdProveedor	IdCategoria	PrecioUnitario	Stock
▶	1	Chai	2	1	10.0000	80
	2	Chang	7	4	3.0000	24
	3	Chicha	10	1	10.0000	39
	4	Chef Anton's Cajun Seasoning	2	2	22.0000	80
	5	Chef Anton's Gumbo Mix	2	2	21.3500	31
	6	Grandma's Boysenberry Spread	2	3	25.0000	50
	7	Uncle Bob's Organic Dried Pears	3	7	30.0000	98
	8	Northwoods Cranberry Sauce	3	2	40.0000	99
	9	Mishi Kobe Niku	4	6	97.0000	96
	10	Ikura	4	8	31.0000	93
	11	Queso Cabrales	5	4	21.0000	97
	12	Queso Manchego La Pastora	5	4	38.0000	100
	13	Konbu	6	8	6.0000	100
	14	Tofu	6	3	23.2500	99

5. Creando Reportes e Impresiones en Windows Forms

En esta última parte aprenderemos como crear reportes e imprimir desde una aplicación Windows Forms, para lo cual existen varias formas, como por ejemplo usar el objeto PrintDocument para crear el documento a imprimir usando gráficos y diálogos para la visualización de la vista preliminar, configurar la página o impresión.

Si se desea un reporte de datos complejo la mejor alternativa seria usar el Microsoft ReportViewer que permite crear reportes usando como origen de datos objetos y permite realizar agrupaciones, subtotales, fórmulas, además permite exportar el reporte a un archivo de Excel o PDF.

Otra alternativa de crear un documento para imprimir es usar Microsoft Word sobre todo si ya existe una plantilla en formato doc o docx la cual debe llenarse con data de la aplicación o quizás crear un cuadro de análisis usando Microsoft Excel el cual también nos permitirá graficar.

5.1. Usando PrintDocument

La clase PrintDocument que se encuentra en el espacio de nombres System.Drawing.Printing se usa para crear contenido a imprimir mediante el uso de graficos (clase Graphics). Esta clase tiene varios eventos pero el principal es el evento PrintPage donde se escribe el código de la página o páginas que se desean imprimir.

Otros eventos de la clase PrintDocument son BeginPrint cuando se inicia la impresión, EndPrint cuando se finaliza la impresión y QueryPageSettings que es usado para cambiar la configuración de la impresión.

Para realizar una vista preliminar (preview) del documento que se ha construido en el evento PrintPage del PrintDocument existen 2 formas: usar el control PrintPreviewControl o usar el diálogo PrintPreviewDialog, la diferencia es que el primero es un simple control que hay que agregar a un formulario y no tiene funcionalidad de zoom, ver varias páginas, etc. que si trae incorporado el diálogo de preview.

También existen 2 diálogos más de impresión: uno es el PageSetupDialog que permite cambiar la configuración de la página y el otro es el PrintDialog que permite especificar la salida de la impresión.

El control de preview y los 3 dialogos de impresión tienen una propiedad Document donde hay que asociar el objeto PrintDocument para poder referenciar al documento que se desea imprimir. Para obtener más información de cómo imprimir usando la clase PrintDocument ver la referencia 26 al final del libro.

A continuación veremos un ejemplo de cómo crear un reporte de productos en varias hojas usando el PrintDocument y los 3 diálogos de impresión, los cuales se verán al dar clic derecho sobre la grilla de productos.

Demo 66

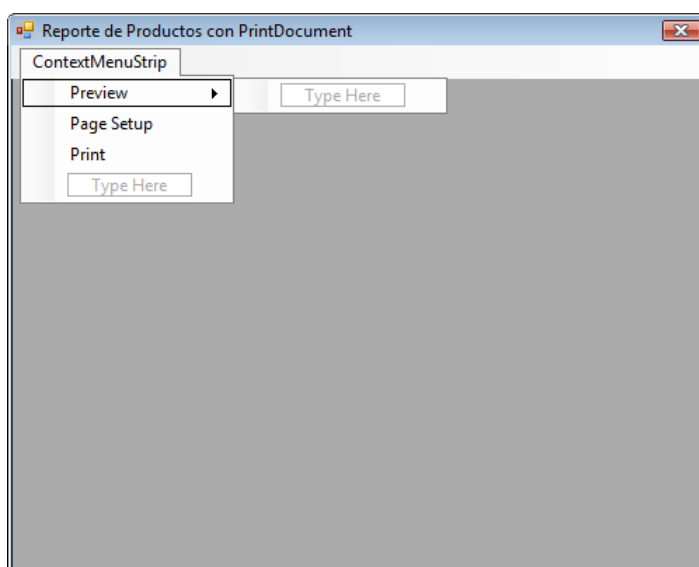
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo66.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmReporteProducto.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmReporteProducto
	FormBorderStyle	FixedSingle
	Size	500, 400
	MaximizeBox	False
	MinimizeBox	False
	Text	Reporte de Productos con PrintDocument
	StartPosition	CenterScreen
ContextMenuStrip1	Name	mnuImpresion
ToolStripMenuItem1	Name	mnuPreview
	Text	Preview
ToolStripMenuItem2	Name	mnuPageSetup
	Text	PageSetup

ToolStripMenuItem3	Name Text	mnuPrint Print
DataGridView1	Name AllowUserToAddRows AllowUserToDeleteRows ContextMenuStrip Dock ReadOnly SelectionMode	dgvProducto False False mnuImpresion Fill True FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 4.40:

Gráfico 4.40: Diseño del formulario Reporte de Productos



- Escribir el siguiente código para el formulario:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.Drawing.Printing 'PrintDocument
Imports System.Drawing.Drawing2D

Public Class frmReporteProducto
    Private lobeProducto As New List(Of beProducto)
    Private WithEvents pd As New PrintDocument
```

```
Private cr As Integer
Private totPags As Integer
Private numPag As Integer

Private Sub ListarProductos(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    Dim obrProducto As New brProducto
    lobeProducto = obrProducto.Listar()
    dgvProducto.DataSource = lobeProducto
End Sub

Private Sub ImprimirPagina(ByVal sender As Object, _
ByVal e As PrintPageEventArgs) Handles pd.PrintPage
    Dim x As Integer = e.MarginBounds.Left
    Dim y As Integer = e.MarginBounds.Top
    Dim rec As New Rectangle(x, y - 20, _
e.MarginBounds.Width, e.MarginBounds.Height)
    Dim deg As New LinearGradientBrush(rec, Color.Aqua, _
Color.Blue, LinearGradientMode.BackwardDiagonal)
    Dim fuente As New Font("Arial", 10)
    Dim brocha As Brush = Brushes.Blue
    Dim brochaTitulo As Brush = Brushes.Red
    Dim tlp As Integer = e.MarginBounds.Height \ _
(fuente.GetHeight + 20)
    totPags = (lobeProducto.Count \ (tlp - 1))
    If lobeProducto.Count Mod (tlp - 1) > 0 Then totPags += 1
    numPag += 1
    With e.Graphics
        '.FillRectangle(deg, rec)
        x = e.MarginBounds.Left
        .DrawString("Codigo", fuente, brochaTitulo, x, y)
        x = x + 100
        .DrawString("Nombre", fuente, brochaTitulo, x, y)
        x = x + 300
        .DrawString("Precio Unit", fuente, brochaTitulo, x, y)
        x = x + 100
        .DrawString("Stock", fuente, brochaTitulo, x, y)
        y = y + fuente.GetHeight + 20
        Dim I As Integer
        For I = 0 To tlp - 2
```

```
If cr = lobeProducto.Count Then Exit For
x = e.MarginBounds.Left
.DrawString(lobeProducto(cr).Codigo, _
fuente, brocha, x, y)
x = x + 100
.DrawString(lobeProducto(cr).Nombre, _
fuente, brocha, x, y)
x = x + 300
.DrawString(lobeProducto(cr).PrecioUnitario, _
fuente, brocha, x, y)
x = x + 100
.DrawString(lobeProducto(cr).Stock, _
fuente, brocha, x, y)
y = y + fuente.GetHeight + 20
cr += 1
Next
.DrawString(String.Format("Pag {0} de {1}", _
numPag, totPag), fuente, brochaTitulo, 400, _
e.MarginBounds.Top + e.MarginBounds.Height)
e.HasMorePages = (cr < lobeProducto.Count - 1)
End With
End Sub

Private Sub Preview(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuPreview.Click
    cr = 0
    Dim ppd As New PrintPreviewDialog
    ppd.Document = pd
    ppd.ShowDialog()
End Sub

Private Sub PageSetup(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuPageSetup.Click
    Dim psd As New PageSetupDialog
    psd.Document = pd
    psd.ShowDialog()
End Sub

Private Sub Print(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuPrint.Click
```

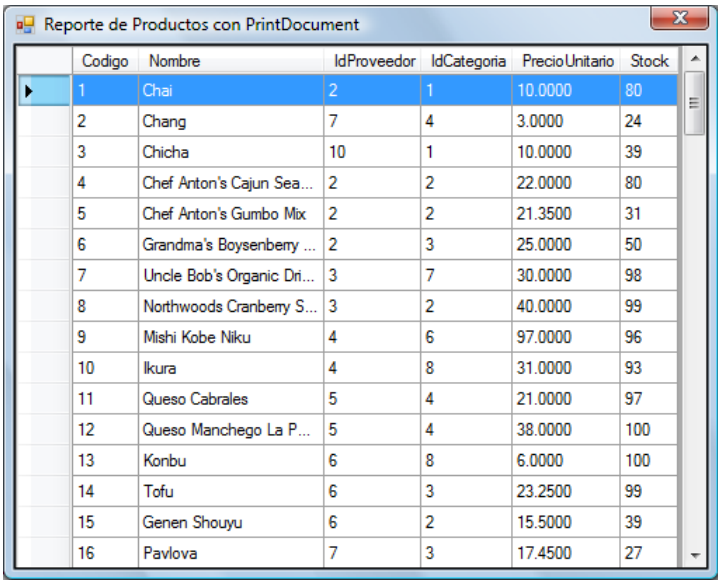
```
Dim pdg As New PrintDialog
pdg.Document = pd
pdg.ShowDialog()
End Sub
End Class
```

Nota: En el evento PrintPage el parámetro e tiene una propiedad HasMore Pages que especifica si se va a crear una nueva pagina, es decir se crearan nuevas paginas mientras el contador de registros sea menor al último registro.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

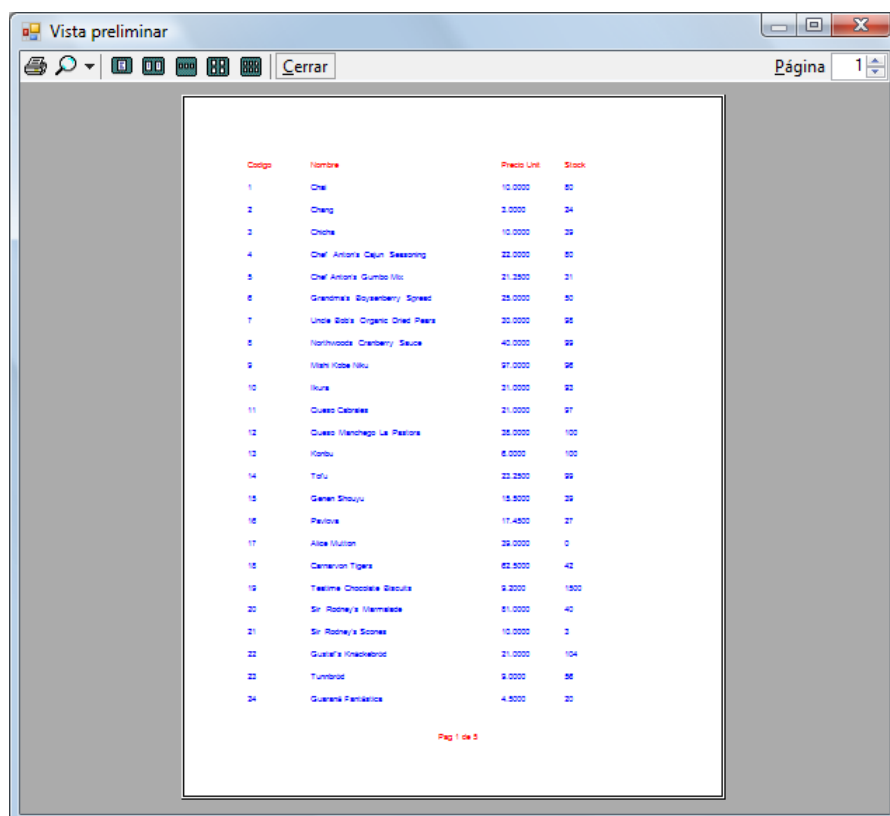
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.41: Ejecución del formulario Reporte de Productos

	Codigo	Nombre	IdProveedor	IdCategoria	PrecioUnitario	Stock
▶	1	Chai	2	1	10.0000	80
	2	Chang	7	4	3.0000	24
	3	Chicha	10	1	10.0000	39
	4	Chef Anton's Cajun Sea...	2	2	22.0000	80
	5	Chef Anton's Gumbo Mix	2	2	21.3500	31
	6	Grandma's Boysenberry ...	2	3	25.0000	50
	7	Uncle Bob's Organic Dri...	3	7	30.0000	98
	8	Northwoods Cranberry S...	3	2	40.0000	99
	9	Mishi Kobe Niku	4	6	97.0000	96
	10	Ikura	4	8	31.0000	93
	11	Queso Cabrales	5	4	21.0000	97
	12	Queso Manchego La P...	5	4	38.0000	100
	13	Konbu	6	8	6.0000	100
	14	Tofu	6	3	23.2500	99
	15	Genen Shouyu	6	2	15.5000	39
	16	Pavlova	7	3	17.4500	27

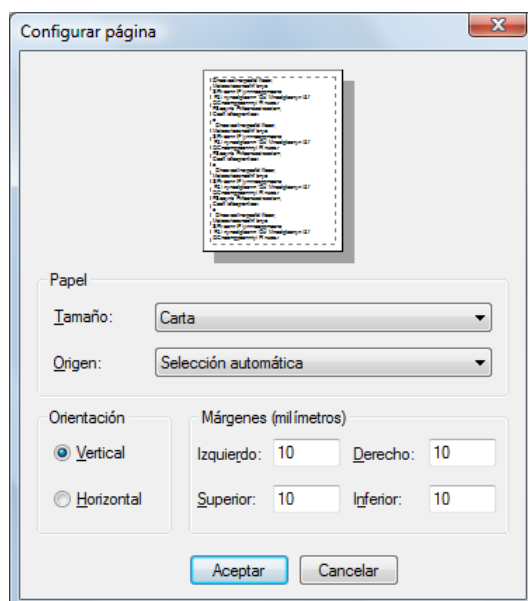
- Clic derecho sobre la grilla de productos y se presentará un menú contextual, seleccionar la opción Preview y se verá la siguiente figura.

Gráfico 4.42: Diálogo de Preview del Reporte de Productos



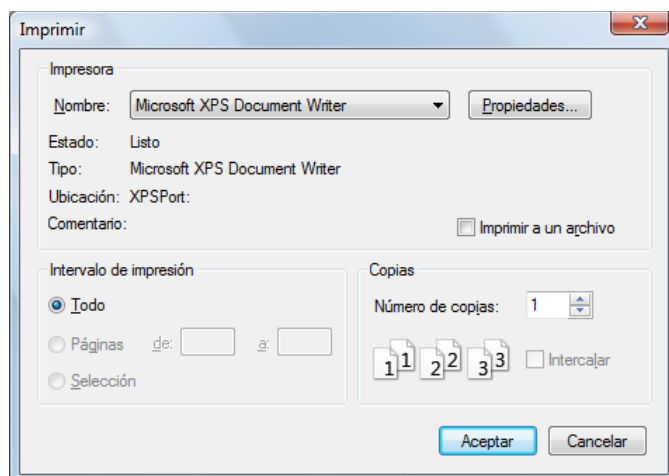
- Clic derecho sobre la grilla de productos y del menú contextual ahora seleccionar la opción Page Setup y se verá la siguiente figura.

Gráfico 4.43: Diálogo de Configurar Página



- Finalmente, clic derecho sobre la grilla de productos y del menú contextual ahora seleccionar la opción Print y se verá la figura 4.44.

Gráfico 4.44: Diálogo de Imprimir



5.2. Informes de Microsoft

Para crear reportes de datos podemos usar los Informes de Microsoft los cuales incluyen funcionalidad de diseñar el reporte usando controles y un diseñador mejorado y también la capacidad de poder presentarlos usando un visor conocido como el control "ReportViewer" que tiene una versión para Windows Forms y otra para ASP .NET AJAX.

En Visual Studio 2010 se han mejorado algunas características para los Informes de Microsoft entre ellas:

- Diseñador de Informes para el esquema: Lenguaje de Definición de Informes (RDL 2008), por ejemplo mejoras en el Diseñador de Informes.
- Nuevo Asistente para Informes que simplifica la definición de datos y el diseño del informe.
- Mejoras en los controles ReportViewer, por ejemplo Exportar a Word.
- Mejoras de programación en los controles ReportViewer, por ejemplo Modelo de Eventos más completo.

Diseñador de Informes

El Diseñador de informes de Visual Studio proporciona una interfaz fácil de usar para crear informes sólidos que incluyen datos procedentes de varios tipos de orígenes de datos.

En Visual Studio, los informes se guardan como archivos de definición de informe del cliente (.rdlc). Estos archivos se basan en el mismo esquema que los archivos de definición de informe (.rdl) publicados en los servidores de informes de SQL Server Reporting Services, pero se guardan y se procesan de manera distinta a los archivos .rdl. En tiempo de ejecución, los archivos .rdlc se procesan localmente, y los archivos .rdl se procesan remotamente.

Nota: El Diseñador de informes de Visual Studio es similar a la interfaz de usuario de Business Intelligence Development Studio de SQL Server 2008 o posterior, excepto en que no tiene la funcionalidad de vista previa y en que guarda los informes en archivos .rdlc en lugar de archivos .rdl.

Controles ReportViewer

Microsoft Visual Studio 2010 incluye la funcionalidad de diseño de informes y los controles ReportViewer, que le permiten agregar informes con todas las características a las aplicaciones personalizadas. Los informes pueden contener datos tabulares, agregados y multidimensionales. Los controles ReportViewer le permitirán procesar y mostrar el informe en la aplicación. Hay dos versiones del control. El control de servidor web ReportViewer es un control AJAX de ASP.NET que se utiliza para hospedar informes en proyectos de ASP.NET. El control de Windows Forms ReportViewer se utiliza para hospedar informes en proyectos de aplicaciones Windows.

Puede configurar los dos controles para ejecutarlos en el modo de procesamiento local o en el modo de procesamiento remoto. La configuración del modo de procesamiento afecta a todo el informe, desde el diseño hasta la implementación.

El modo de procesamiento local hace referencia al procesamiento que realiza el control ReportViewer en la aplicación cliente. Todo el procesamiento del informe se realiza en el proceso local con los datos suministrados por la aplicación. Para crear los informes utilizados en el modo de procesamiento local, puede utilizar la plantilla Proyecto de informe de Visual Studio.

El modo de procesamiento remoto hace referencia al procesamiento de informes que realiza un servidor de informes de SQL Server 2008 Reporting Services o posterior. En el modo de procesamiento remoto, el control ReportViewer se utiliza como un visor para representar un informe procesado en un servidor de informes de Reporting Services. Todo el procesamiento, desde la recuperación de datos hasta el procesamiento del informe, se realiza en el servidor de informes. Para utilizar el modo de

procesamiento remoto, debe tener una copia con licencia de SQL Server 2008 o posterior.

Para obtener más información sobre los informes de Microsoft consultar la referencia 27 al final del libro y para ver ejemplos y tutoriales sobre el control ReportViewer ver la referencia 28 al final del libro.

A continuación presentamos un ejemplo de cómo crear un Reporte de Empleados usando el Diseñador de Reportes y cómo presentarlo usando el control ReportViewer para Windows Forms.

Demo 67

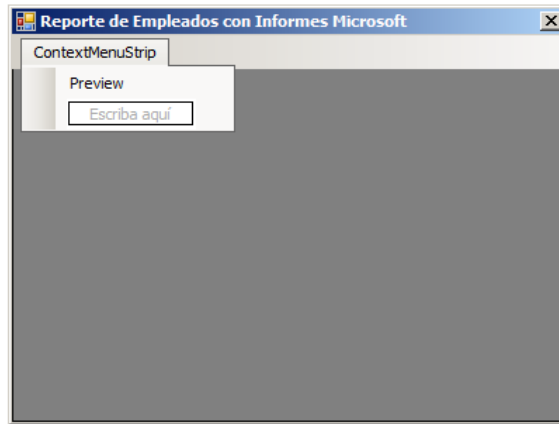
- Del menú "File", seleccionar "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo67.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmListaEmpleado.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaEmpleado
	FormBorderStyle	FixedSingle
	Size	400, 300
	MaximizeBox	False
	MinimizeBox	False
	Text	Reporte de Empleados con Informes Microsoft
	StartPosition	CenterScreen
ContextMenuStrip1	Name	mnuImpresion
ToolStripMenuItem1	Name	mnuPreview
	Text	Preview
DataGridView1	Name	dgvEmpleado
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	ContextMenuStrip	mnuImpresion
	Dock	Fill

	ReadOnly SelectionMode	True FullRowSelect

- El diseño del formulario debe quedar similar al gráfico 4.45:

Gráfico 4.45: Diseño del formulario Lista Empleados



- Agregar un nuevo reporte: del menú "Project", seleccionar "Add New Item".
- En la sección "Reporting" elegir "Report" y escribir como nombre: "ReporteEmpleados.rdlc".
- En el menú "Data", seleccionar "Add New Data Source..."
- Se mostrará el asistente para la configuración de orígenes de datos similar a la figura 4.46.
- Seleccionar como origen de datos la opción "Object" y clic en el botón "Next".
- Se mostrará la opción para seleccionar objetos del asistente para la configuración de orígenes de datos similar a la figura 4.47.
- Seleccionar la clase "beEmpleado" de la librería "Northwind.LibBusiness Entities" y clic en el botón "Finish".

Gráfico 4.46: Asistente Origenes de Datos – Elegir tipo de origen

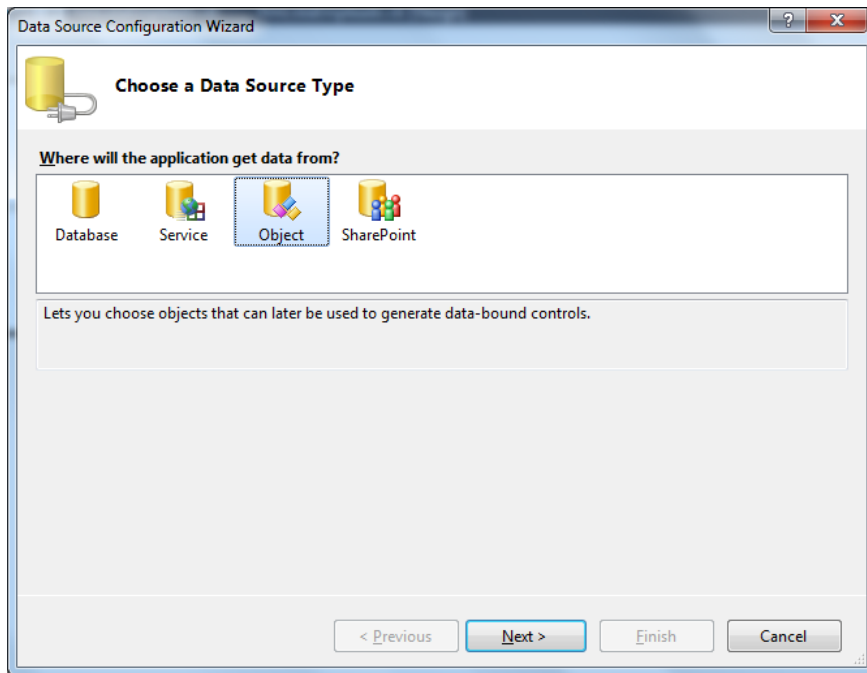
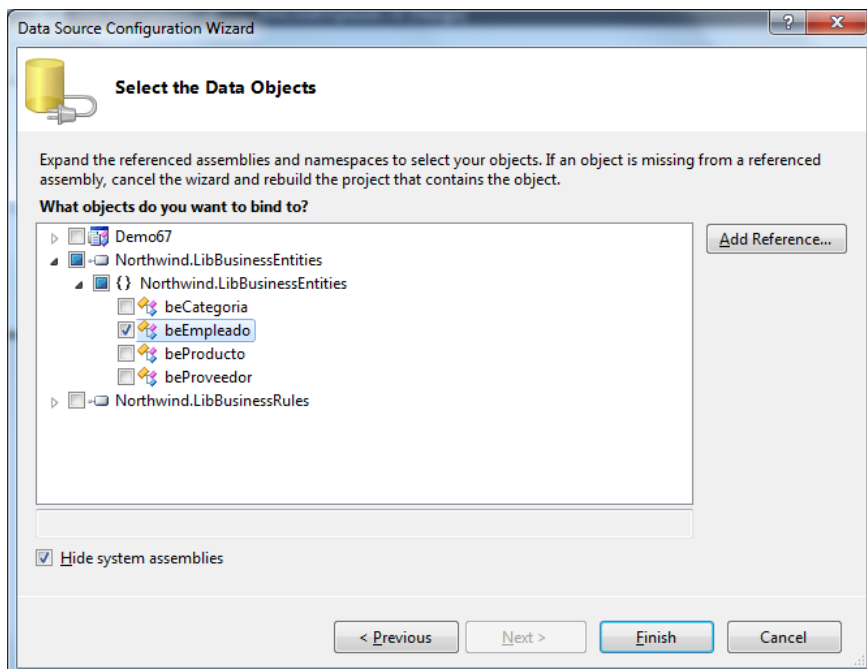
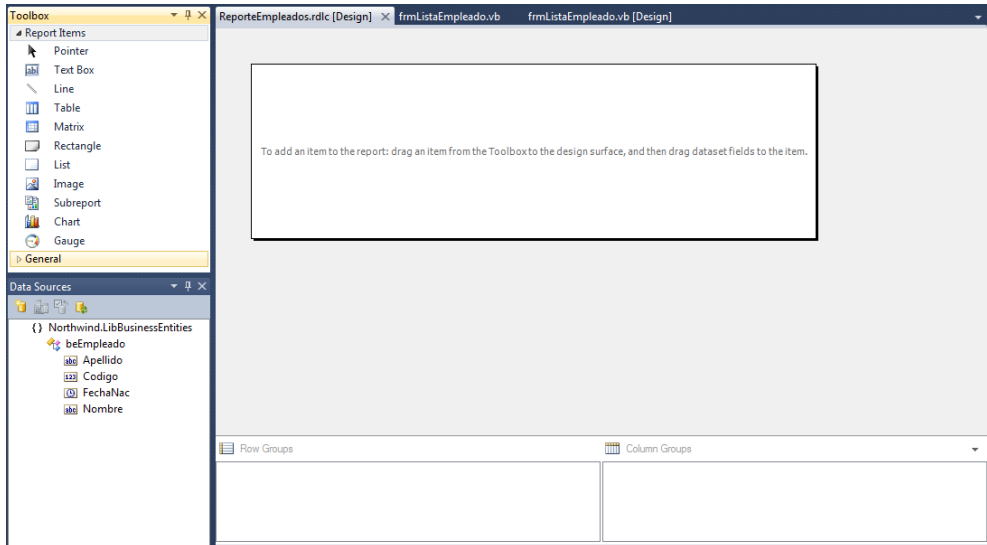


Gráfico 4.47: Asistente Origenes de Datos – Seleccionar objetos



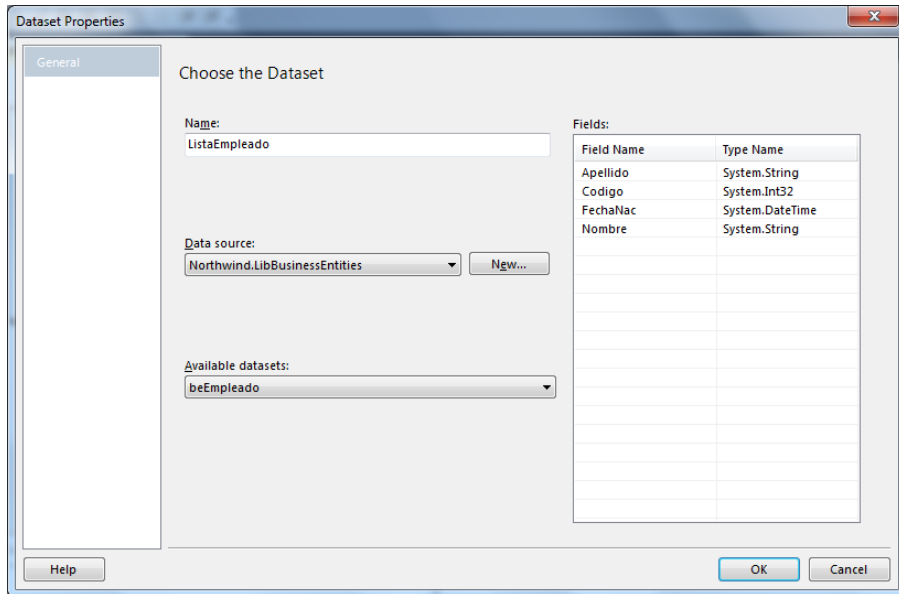
- Se mostrará el diseñador de informes de Microsoft similar a la figura 4.48.

Gráfico 4.48: Diseñador de Informes Microsoft



- En la parte superior izquierda se muestra el cuadro de herramientas con los controles para crear el reporte.
- En la parte inferior derecha se aprecia el origen de datos creado para especificar que campo queremos visualizar en el reporte.
- Arrastrar un control "TextBox" del cuadro de herramientas y escribir: "Lista de Empleados".
- Clic derecho a cualquier barra de herramientas y seleccionar la barra "Report Formatting".
- De la barra configurar el texto a 14 pt, negrita, centrado, color de texto "rojo" y color de fondo "agua".
- Del cuadro de herramientas arrastrar el control "Tabla" y aparecerá un diálogo similar al de la figura 4.49.

Gráfico 4.49: Diálogo de Propiedades del conjunto de datos



- Escribir como nombre para el conjunto de datos "ListaEmpleado", en la opción "Data Source" elegir "Northwind.LibBusinessEntities" y luego en "Available datasets" seleccionar "beEmpleado".
- En la tabla clic en la primera celda de la última fila y seleccionar el campo "Codigo".
- En la segunda celda seleccionar "Apellido" y en la tercera "Nombre".
- Seleccionar la última columna, dar clic derecho y del menú contextual seleccionar "Insert Column", luego "Right".
- En la cuarta columna seleccionar el campo "FechaNac".
- Mover la tabla y seleccionar cada columna y dar los siguientes anchos en la propiedad size, width: 2 para la primera columna, 4 para la segunda y tercera columna y 3 para la cuarta columna.
- Seleccionar la fila de las cabeceras y configurar en negrita, centrado y color de fondo "gris".

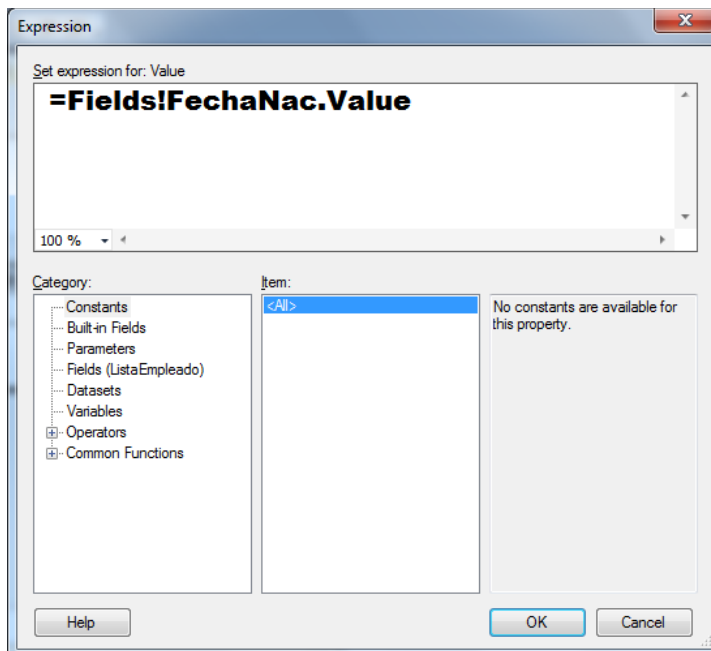
- Dar clic derecho a cualquier barra de herramientas y seleccionar la barra "Report Borders".
- Seleccionar la tabla, y de la barra de bordes elegir "Outside Border".
- El diseño del reporte debe quedar como se muestra en la figura 4.50.

Gráfico 4.50: Diseño del Reporte de Empleados

Lista de Empleados			
Codigo	Apellido	Nombre	Fecha Nac
[Codigo]	[Apellido]	[Nombre]	[FechaNac]

- Por defecto los campos datetime aparecerán como fecha y hora, para no mostrar la hora en el campo "FechaNac".
- Clic derecho sobre la cuarta celda y seleccionar "Expression".

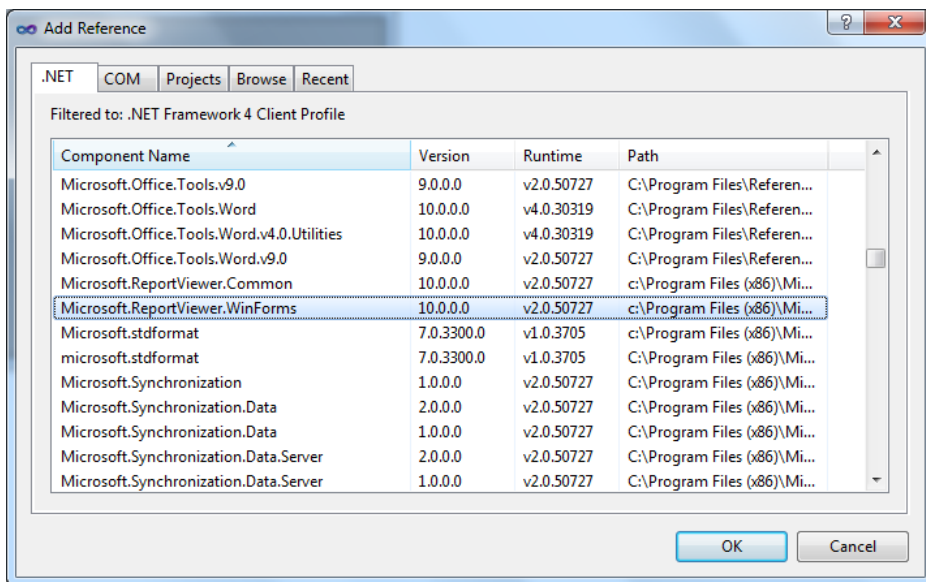
Gráfico 4.51: Diálogo de Expresiones de Informes Microsoft



Nota: En el editor de expresiones se puede escribir fórmulas usando campos del origen de datos, operadores, funciones, etc.

- Expandir la categoría "Common Functions", seleccionar "Text" y luego "FormatDateTime".
- Cortar el valor de la expresión, dar doble clic a la función, pegar lo cortado y completar la expresión como se muestra a continuación: =FormatDateTime(Fields!FechaNac.Value,DateFormat.ShortDate).
- Alinear la celda de la fecha a la derecha, cerrar la ventana del diseñador de reporte e ir al formulario.
- Hacer una referencia a la librería del visor de reportes para Windows: del menú "Project", "Add references" y elegir "Microsoft.ReportViewer.WinForms".

Gráfico 4.52: Diálogo para agregar referencia al ReportViewer



- Escribir el siguiente código para el formulario:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports Microsoft.Reporting.WinForms
```

```
Public Class frmListaEmpleado
    Private lobeEmpleado As New List(Of beEmpleado)

    Private Sub ListarEmpleados(ByVal sender As Object, _
        ByVal e As EventArgs) Handles MyBase.Load
        Dim obrEmpleado As New brEmpleado
        lobeEmpleado = obrEmpleado.Listar
        dgvEmpleado.DataSource = lobeEmpleado
    End Sub

    Private Sub VerPreviewReporte(ByVal sender As Object, _
        ByVal e As EventArgs) Handles mnuPreview.Click
        Dim frm As New Form
        Dim rv As New ReportViewer
        With rv.LocalReport
            .ReportPath = _
                "C:\Data\DemosLibro\LibroVB2010\Demo67\ReporteEmpleados.rdlc"
            .DataSources.Add(New ReportDataSource _
                ("ListaEmpleado", lobeEmpleado))
        End With
        rv.RefreshReport()
        rv.Dock = DockStyle.Fill
        frm.Controls.Add(rv)
        frm.Text = "Preview del Reporte de Empleados"
        frm.WindowState = FormWindowState.Maximized
        frm.ShowDialog()
    End Sub
End Class
```

Nota: En el código anterior se crea dinámicamente un formulario con un visor el cual para mostrar el reporte necesita configurar 2 propiedades de su propiedad LocalReport que son ReportPath con la ruta del reporte y DataSource con el origen de datos asociado al reporte y con la lista de objetos a llenar.

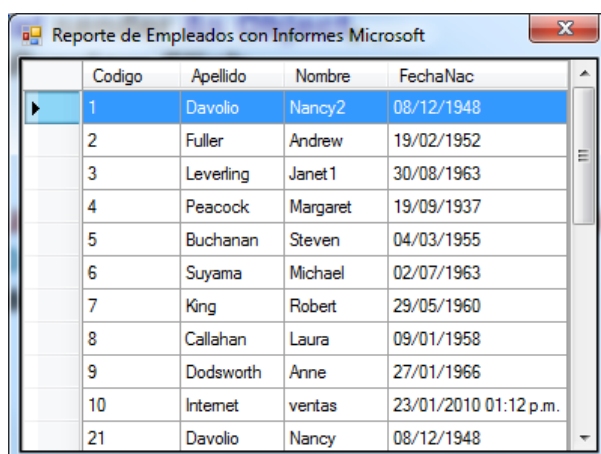
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".

- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 4.53: Ejecución del formulario Lista Empleados



	Codigo	Apellido	Nombre	FechaNac
1	Davolio	Nancy2	08/12/1948	
2	Fuller	Andrew	19/02/1952	
3	Leverling	Janet1	30/08/1963	
4	Peacock	Margaret	19/09/1937	
5	Buchanan	Steven	04/03/1955	
6	Suyama	Michael	02/07/1963	
7	King	Robert	29/05/1960	
8	Callahan	Laura	09/01/1958	
9	Dodsworth	Anne	27/01/1966	
10	Internet	ventas	23/01/2010 01:12 p.m.	
21	Davolio	Nancy	08/12/1948	

- Clic derecho a la grilla de empleados y del menú contextual seleccionar "Preview".
- Se mostrará la ventana del reporte similar a la figura 4.54.

Gráfico 4.54: Ejecución del Reporte de Empleados



Preview del Reporte de Empleados

1 of 1 100% Find | Next

Lista de Empleados

Codigo	Apellido	Nombre	Fecha Nac
1	Davolio	Nancy2	12/8/1948
2	Fuller	Andrew	2/19/1952
3	Leverling	Janet1	8/30/1963
4	Peacock	Margaret	9/19/1937
5	Buchanan	Steven	3/4/1955
6	Suyama	Michael	7/2/1963
7	King	Robert	5/29/1960
8	Callahan	Laura	1/9/1958
9	Dodsworth	Anne	1/27/1966
10	Internet	ventas	1/23/2010
21	Davolio	Nancy	12/8/1948

Importante: En la ventana del visor podremos realizar las siguientes acciones con el reporte:

- Avanzar a cualquier página.
- Imprimir el reporte.
- Ajustar la vista de la página.
- Configurar la página: tamaño, tipo, orientación y márgenes.
- Exportar: Excel, PDF y Word.
- Configurar Zoom.
- Buscar un texto dentro del reporte, etc.

5.3. Trabajando con Word

En muchas ocasiones necesitamos crear dinámicamente un documento de Word con los datos de la aplicación, para esto debemos usar automatización COM o ActiveX.

Desde .NET cuando usamos otras tecnologías como COM, ActiveX o APIs trabajamos con `System.Runtime.InteropServices`, es decir creamos un puente entre el CLR de .NET y otro Motor de Ejecución externo.

Para usar un componente COM desde Visual Basic como Word o Excel, existen 2 formas:

- Enlace en tiempo de diseño (Early Binding)

Es el mecanismo más usado y consiste en referenciar a una librería de tipos, la ventaja es que permite crear objetos y usar miembros disponiendo de la ayuda del IDE que usemos, la desventaja es que nos hacemos dependiente de una versión específica del programa, por ejemplo MS Word 2010.

- Enlace en tiempo de ejecución (Late Binding)

Mediante esta técnica no es necesario hacer referencia a ninguna librería de tipos en especial ya que se crean los objetos usando el método `CreateObject` (solo para Visual Basic), la ventaja es que no dependemos de ninguna versión del programa, se toma la última existente, la desventaja es que no disponemos de ayuda para crear los objetos y usar los métodos es decir hay que conocer de memoria todos los componentes, recomendable para expertos en dicho componente.

La recomendación que haría sería para desarrollar hacerlo usando la librería de tipos (enlace en tiempo de diseño) y al final antes de desplegar a producción quitar la referencia y cambiar a enlace en ejecución.

A continuación un ejemplo de cómo crear documentos de Word dinámicamente, primero mediante enlace en tiempo de diseño llenaremos un formato de ficha de producto predefinido con los datos del producto seleccionado y segundo mediante enlace en tiempo de ejecución crearemos una lista con todos los productos.

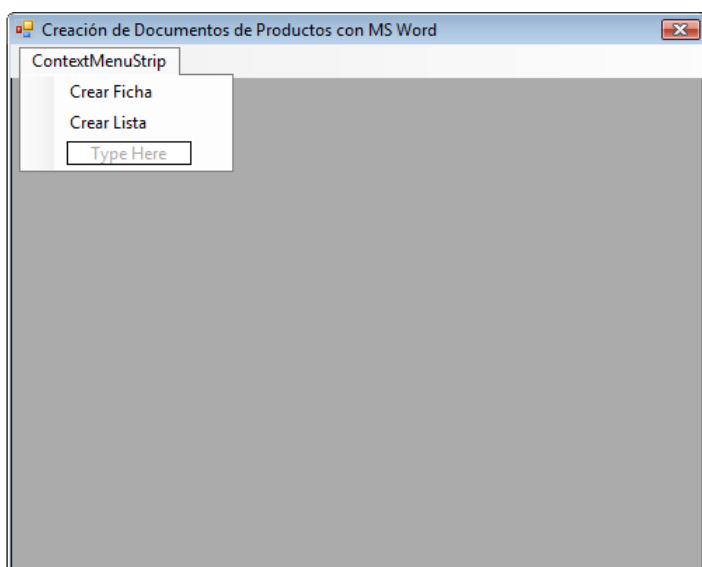
Demo 68

- Del menú "File", seleccionar "Add", "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo68.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmListaProducto.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaProducto
	FormBorderStyle	FixedSingle
	Size	500, 400
	MaximizeBox	False
	MinimizeBox	False
	Text	Creación de Documentos de Productos con MS Word
	StartPosition	CenterScreen
ContextMenuStrip1	Name	mnuDocumento
ToolStripMenuItem1	Name	mnuCrearFicha
	Text	Crear Ficha
ToolStripMenuItem2	Name	mnuCrearLista
	Text	Crear Lista
DataGridView1	Name	dgvProducto
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	ContextMenuStrip	mnuDocumento
	Dock	Fill
	ReadOnly	True
	SelectionMode	FullRowSelect

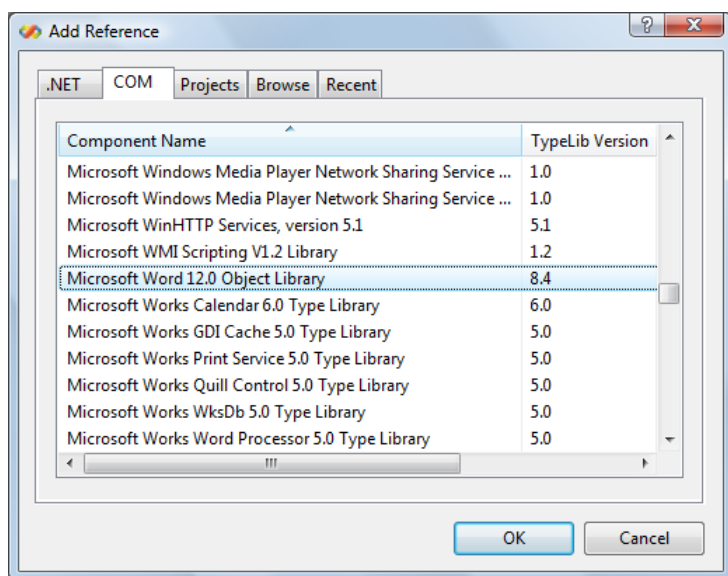
- El diseño del formulario debe quedar similar al gráfico 4.55:

Gráfico 4.55: Diseño del formulario Lista Productos



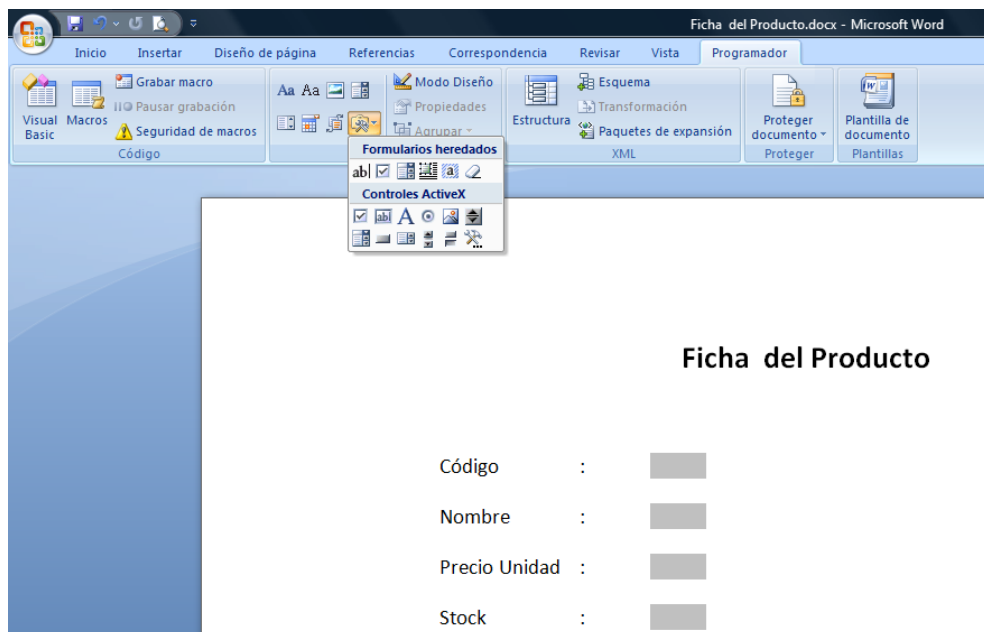
- Hacer una referencia a la librería de tipos de MS Word: clic derecho al proyecto, seleccionar "Add Reference" y luego en la ficha "COM" seleccionar "Microsoft Word 12.0 Object Library" como se ve debajo.

Gráfico 4.56: Diálogo de Agregar Referencia a MS Word



- Crear un documento en MS Word para la Ficha del Producto, similar al de la siguiente figura.

Gráfico 4.57: Diseño del Documento Ficha del Producto en Word



Nota: Para crear la Ficha del Producto en MS Word se debe activar la "Ficha del Programador" y luego usar de la barra "Formularios Heredados" el control "Campo de texto", el cual se muestra como una sombra gris.

Grabe el documento como "Ficha del Producto.docx" en alguna ubicación donde podamos recuperarlo mediante el programa.

- Escribir el siguiente código para el formulario:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Public Class frmListaProducto
    Private lobeProducto As New List(Of beProducto)
    Private Ruta As String = _
        "C:\Lduenas\NET\LibroVB2010\Archivos\"
```

```
    Private Sub ListarProductos(ByVal sender As System.Object, _
```



```
ByVal e As System.EventArgs) Handles MyBase.Load
```

```
Dim obrProducto As New brProducto
```

```
lobeProducto = obrProducto.Listar()
```

```
dgvProducto.DataSource = lobeProducto
```

```
End Sub
```

```
Private Sub CrearFicha(ByVal sender As System.Object, _
```

```
ByVal e As System.EventArgs) Handles mnuCrearFicha.Click
```

```
If dgvProducto.SelectedRows.Count > 0 Then
```

```
With dgvProducto.CurrentRow
```

```
Dim DocPlantilla As String = _
```

```
String.Format("{0}Ficha del Producto.docx", Ruta)
```

```
Dim DocFicha As String = _
```

```
String.Format("{0}Ficha del Producto {1}.docx", _
```

```
Ruta, .Cells(1).Value)
```

```
Dim oWord As New Microsoft.Office.Interop.Word.Application
```

```
oWord.Visible = True
```

```
oWord.Documents.Open(DocPlantilla)
```

```
oWord.ActiveDocument.Fields(1).Result.Text = _
```

```
.Cells(0).Value
```

```
oWord.ActiveDocument.Fields(2).Result.Text = _
```

```
.Cells(1).Value
```

```
oWord.ActiveDocument.Fields(3).Result.Text = _
```

```
.Cells(4).Value
```

```
oWord.ActiveDocument.Fields(4).Result.Text = _
```

```
.Cells(5).Value
```

```
oWord.ActiveDocument.SaveAs(DocFicha)
```

```
End With
```

```
End If
```

```
End Sub
```

```
Private Sub CrearLista(ByVal sender As System.Object, _
```

```
ByVal e As System.EventArgs) Handles mnuCrearLista.Click
```

```
Dim oWord As Object = CreateObject("Word.Application")
```

```
oWord.Visible = True
```

```
oWord.Documents.Add()
```

```
With oWord.Selection
```

```
.Font.Size = 16
```

```
.Font.ColorIndex = 2
```

```
.TypeText("Lista de Productos")
```

```
.TypeParagraph()  
Dim Rango As Object = .Range  
.Tables.Add(Rango, dgvProducto.Rows.Count + 1, _  
    dgvProducto.Columns.Count)  
.Style = "Tabla con cuadrícula"  
.Tables(1).ApplyStyleHeadingRows = True  
.Tables(1).ApplyStyleFirstColumn = True  
.Tables(1).ApplyStyleColumnBands = True  
For I = 0 To dgvProducto.ColumnCount - 1  
    .Tables(1).Rows(1).Cells(I + 1).Select()  
    .Font.Size = 12  
    .Font.ColorIndex = 6  
    .TypeText(dgvProducto.Columns(I).HeaderText)  
Next  
.Font.ColorIndex = 1  
For I = 0 To dgvProducto.RowCount - 1  
    For J = 0 To dgvProducto.ColumnCount - 1  
        .Tables(1).Rows(I + 2).Cells(J + 1).Select()  
        .Font.Size = 12  
        .Font.ColorIndex = 1  
        .TypeText(dgvProducto.Rows(I).Cells(J).Value.ToString)  
    Next  
Next  
End With  
Dim DocLista As String = _  
String.Format("{0}Lista de Productos.docx", Ruta)  
oWord.ActiveDocument.SaveAs(DocLista)  
End Sub  
End Class
```

Nota: Observe al escribir el código de Crear la Ficha como se muestra la ayuda de las clases y miembros de cada clase de Word ya que se usa el enlace en tiempo de diseño, pero en cambio, al Crear la Lista mediante enlace en tiempo de ejecución (CreateObject) no se dispone de la ayuda.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".

- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

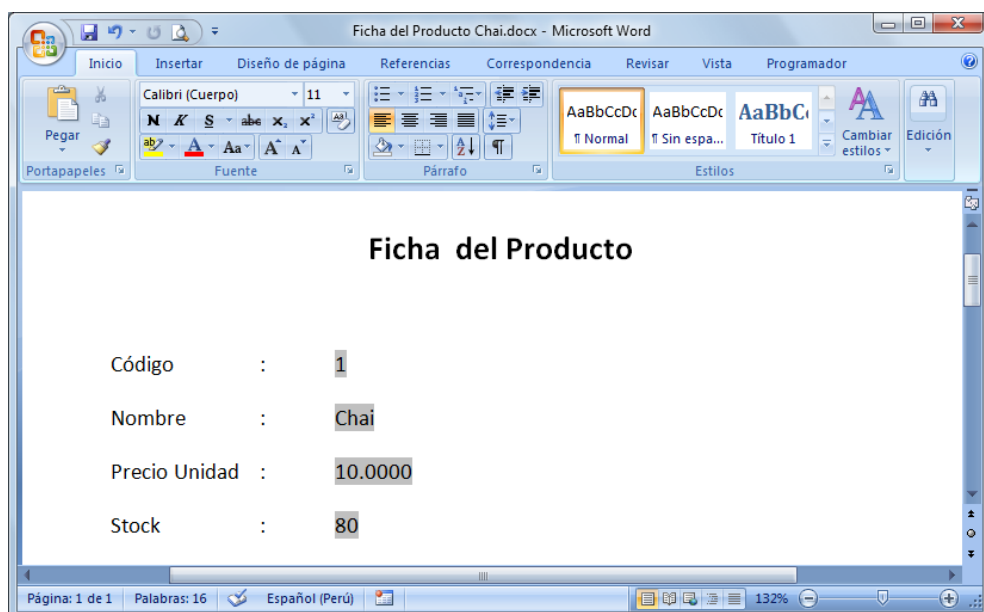
Gráfico 4.58: Ejecución del formulario Lista Productos



	Codigo	Nombre	IdProveedor	IdCategoria	PrecioUnitario	Stock
▶	1	Chai	2	1	10.0000	80
	2	Chang	7	4	3.0000	24
	3	Chicha	10	1	10.0000	39
	4	Chef Anton's Cajun Sea...	2	2	22.0000	80
	5	Chef Anton's Gumbo Mix	2	2	21.3500	31
	6	Grandma's Boysenberry ...	2	3	25.0000	50
	7	Uncle Bob's Organic Dri...	3	7	30.0000	98
	8	Northwoods Cranberry S...	3	2	40.0000	99
	9	Mishi Kobe Niku	4	6	97.0000	96
	10	Ikura	4	8	31.0000	93
	11	Queso Cabrales	5	4	21.0000	97
	12	Queso Manchego La Pa...	5	4	38.0000	100
	13	Konbu	6	8	6.0000	100
	14	Tofu	6	3	23.2500	99
	15	Genen Shouyu	6	2	15.5000	39
	16	Pavlova	7	3	17.4500	27

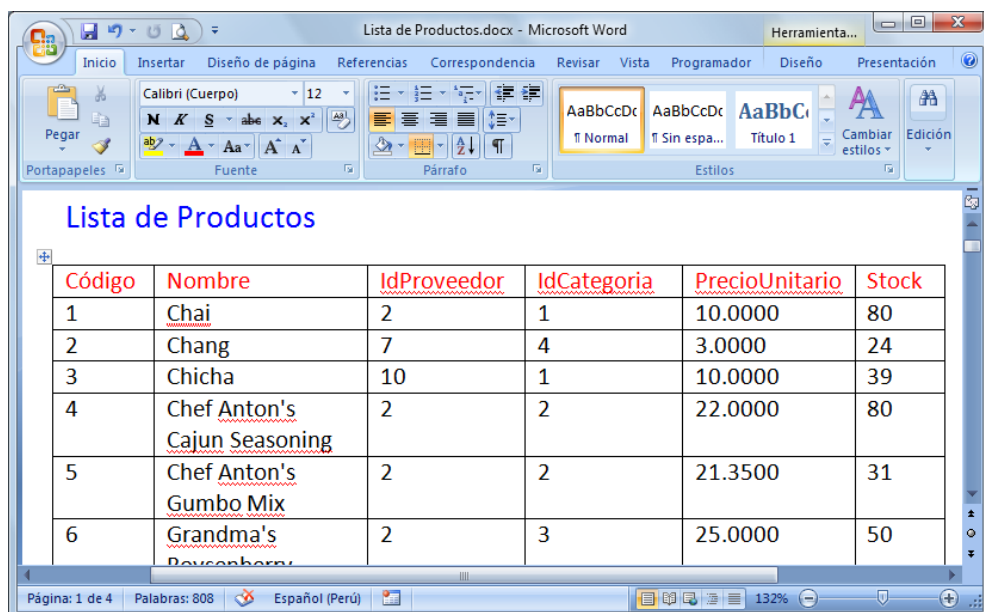
- Clic derecho sobre la grilla de productos y se presentará un menú contextual, seleccionar la opción Crear Ficha y se verá la figura 4.57.

Gráfico 4.59: Ejecución del Documento Ficha de Producto en Word



- Clic derecho sobre la grilla de productos y del menú contextual ahora seleccionar la opción Crear Lista y se verá la siguiente figura.

Gráfico 4.60: Ejecución del Documento Lista de Productos Word



5.4. Trabajando con Excel

Excel es una de las aplicaciones mas usadas del mercado para realizar cálculos y crear gráficos de datos. Desde .NET podemos usar el potencial de Excel para realizar cálculos complejos o gráficos de datos de diversos tipos, lo que seria trabajoso en caso de hacerlo desde cero por código.

Al igual que Word la técnica de enlace puede ser en tiempo de diseño o en ejecución. La programación es muy similar lo que cambia es el modelo de objetos, en Word se trabaja con Documentos (Documents) y en Excel trabajamos con Libros (WorkBooks).

A continuación un ejemplo de creación de una hoja de Excel con los precios de los productos y un gráfico de barras en 3D.

Demo 69

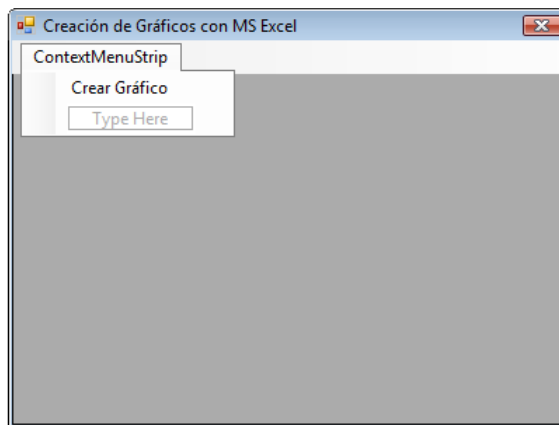
- Del menú "File", seleccionar "Add", "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo69.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmListaProducto.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaProducto
	FormBorderStyle	FixedSingle
	Size	400, 300
	MaximizeBox	False
	MinimizeBox	False
	Text	Creación de Gráficos con MS Excel
	StartPosition	CenterScreen
ContextMenuStrip1	Name	mnuExcel
ToolStripMenuItem1	Name	mnuCrearGrafico
	Text	Crear Gráfico
DataGridView1	Name	dgvProducto

	AllowUserToAddRows AllowUserToDeleteRows ContextMenuStrip Dock ReadOnly SelectionMode	False False mnuExcel Fill True FullRowSelect

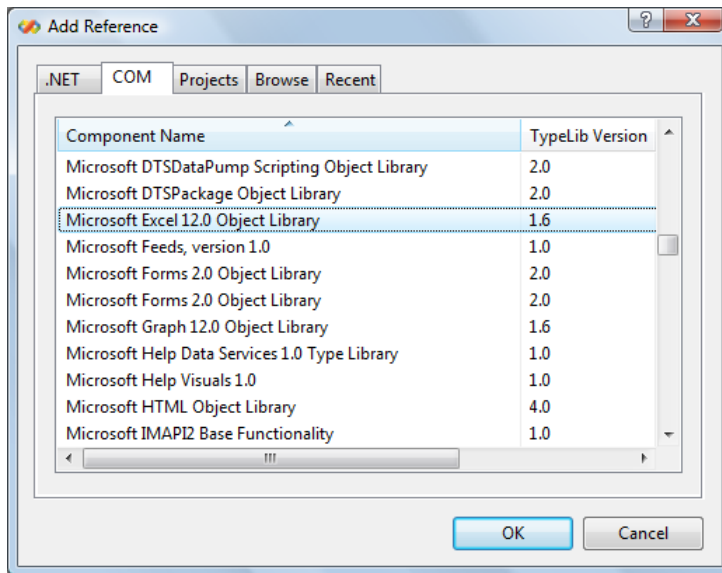
- El diseño del formulario debe quedar similar al gráfico 4.61:

Gráfico 4.61: Diseño del formulario Lista Productos



- Hacer una referencia a la librería de tipos de MS Excel: clic derecho al proyecto, seleccionar "Add Reference" y luego en la ficha "COM" seleccionar "Microsoft Excel 12.0 Object Library" como se ve en la figura 4.62.

Gráfico 4.62: Diálogo de Agregar Referencia a MS Excel



- Escribir el siguiente código para el formulario:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Public Class frmListaProducto
```

```
    Private lobeProducto As New List(Of beProducto)
```

```
    Private Sub ListarProductos(ByVal sender As System.Object, _  
        ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        Dim obrProducto As New brProducto
```

```
        lobeProducto = obrProducto.Listar()
```

```
        dgvProducto.DataSource = lobeProducto
```

```
        dgvProducto.Columns(0).Visible = False
```

```
        dgvProducto.Columns(1).Width = 250
```

```
        dgvProducto.Columns(2).Visible = False
```

```
        dgvProducto.Columns(3).Visible = False
```

```
        dgvProducto.Columns(4).Width = 80
```

```
        dgvProducto.Columns(4).DefaultCellStyle.Alignment = _
```

```
        DataGridViewContentAlignment.MiddleRight
```

```
        dgvProducto.Columns(4).DefaultCellStyle.Format = "n2"
```

```
        dgvProducto.Columns(5).Visible = False
```

```
    End Sub
```

```
Private Sub CrearFicha(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles mnuCrearGrafico.Click
    Dim oExcel As New Microsoft.Office.Interop.Excel.Application
    Dim oRango As Microsoft.Office.Interop.Excel.Range
    oExcel.Visible = True
    oExcel.Workbooks.Add()
    With oExcel
        .Cells(1, 1).Value = "Descripción del Producto"
        .Cells(1, 2).Value = "Precio Unitario"
        For I = 0 To dgvProducto.Rows.Count - 1
            .Cells(I + 2, 1).Value = dgvProducto.Rows(I).Cells(1).Value
            .Cells(I + 2, 2).Value = dgvProducto.Rows(I).Cells(4).Value
        Next
        .Columns.AutoFit()
        .Range("A1").Select()
        oRango = .Selection.CurrentRegion
        .ActiveWorkbook.Charts.Add()
    End With
    With oExcel.ActiveChart
        .Location(1)
        .SetSourceData(oRango, 2)
        .ChartType = -4100
        .ChartTitle.Text = "Gráfico de Precios de Productos"
    End With
End Sub
End Class
```

Nota: En el código anterior el método AutoFit de la colección Columns permite autoajustar el ancho de todas las columnas automáticamente.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

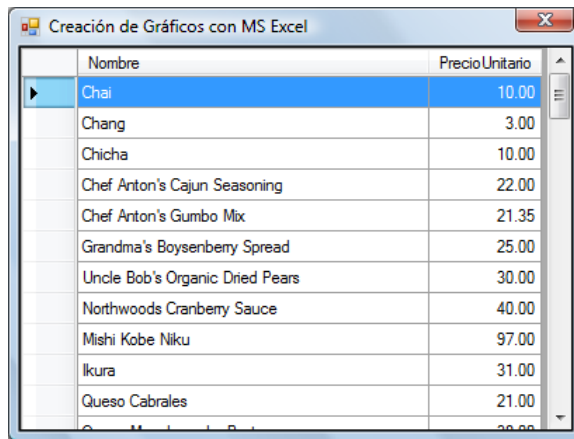
```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;"
```



```
data source=Lduenas\MCTS;initial catalog=Northwind"/>
</appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

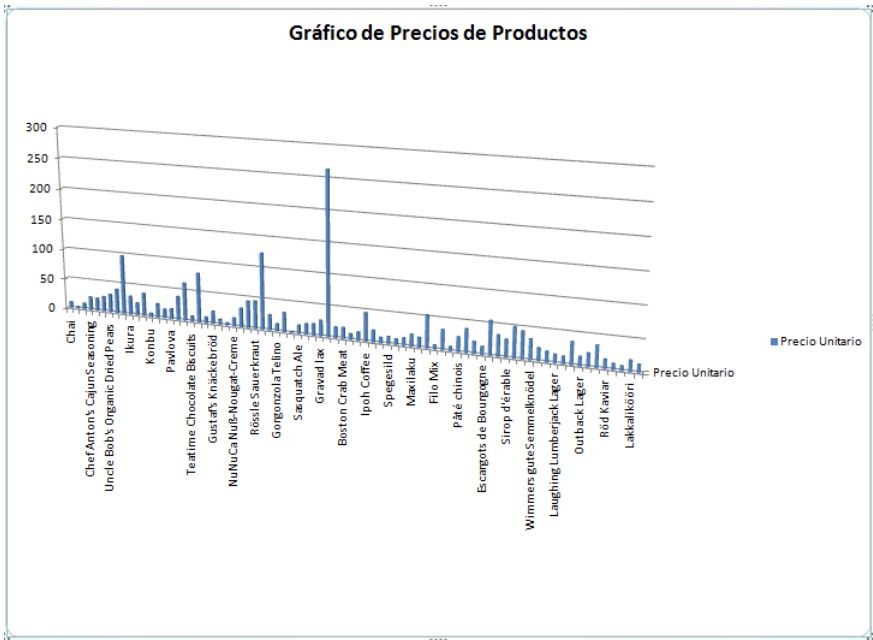
Gráfico 4.63: Ejecución del formulario Lista Productos



Nombre	PrecioUnitario
Chai	10.00
Chang	3.00
Chicha	10.00
Chef Anton's Cajun Seasoning	22.00
Chef Anton's Gumbo Mix	21.35
Grandma's Boysenberry Spread	25.00
Uncle Bob's Organic Dried Pears	30.00
Northwoods Cranberry Sauce	40.00
Mishi Kobe Niku	97.00
Ikura	31.00
Queso Cabrales	21.00

- Clic derecho sobre la grilla de productos y se presentará un menú contextual, seleccionar la opción Crear Gráfico.
- Se creará dinámicamente un libro de Excel con 2 hojas: la primera hoja tiene 2 columnas que muestran el nombre y el precio de los productos y la segunda hoja tiene el gráfico similar al que se muestra en la figura 4.64.

Gráfico 4.64: Gráfico de Barras de Precios de Productos en Excel



5.5. Usando el Control Chart

El Control Chart permite crear gráfico de datos sin necesidad de salir del entorno de desarrollo de .NET y recurrir a Excel o librerías de terceros. Este se encuentra en System.Windows.Forms.DataVisualization.Charting en la clase Chart para lo cual es necesario hacer referencia a la librería DataVisualization de Windows Forms.

Las 2 propiedades más importantes de la clase Chart son las colecciones ChartAreas y Series. La propiedad de colección Series almacena objetos Series, que se utilizan para almacenar datos que serán mostrados, junto con los atributos de esos datos. La propiedad de colección ChartAreas almacena objetos ChartArea, que se utilizan principalmente para dibujar uno o más gráficos mediante un conjunto de ejes.

Para crear un gráfico con el control Chart es necesario configurar las propiedades del objeto Serie como sigue: la propiedad ChartArea con el nombre de un objeto ChartArea previamente definido, la propiedad XValueMember con el nombre del campo o propiedad que irá en el eje de las X y la propiedad YValueMembers con el nombre o nombres de series que se graficarán en el eje de las Y.

Para obtener más información sobre el control Chart ver la referencia 29 al final del libro. A continuación, el siguiente ejemplo muestra como crear un gráfico de productos usando el control Chart de Windows Forms.

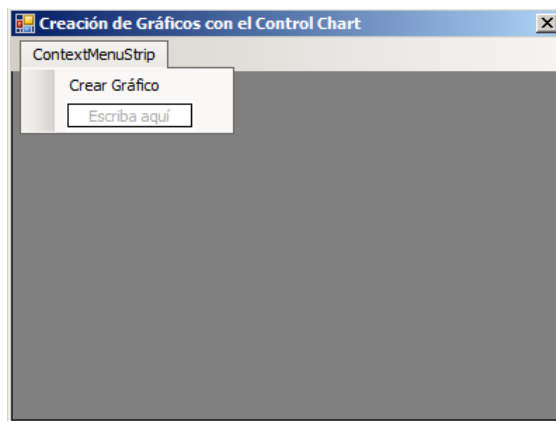
Demo 70

- Del menú "File", seleccionar "Add", "New Project" y luego "Windows Forms Application".
- Crear una aplicación en Visual Basic llamada: Demo70.
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre al formulario de form1.vb a frmListaProducto.vb
- Configurar el formulario de acuerdo a la siguiente tabla:

Objeto	Propiedad	Valor
Form1	Name	frmListaProducto
	FormBorderStyle	FixedSingle
	Size	400, 300
	MaximizeBox	False
	MinimizeBox	False
	Text	Creación de Gráficos con el Control Chart
	StartPosition	CenterScreen
ContextMenuStrip1	Name	mnuChart
ToolStripMenuItem1	Name	mnuCrearGráfico
	Text	Crear Gráfico
DataGridView1	Name	dgvProducto
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	ContextMenuStrip	mnuChart
	Dock	Fill
	ReadOnly	True
	SelectionMode	FullRowSelect

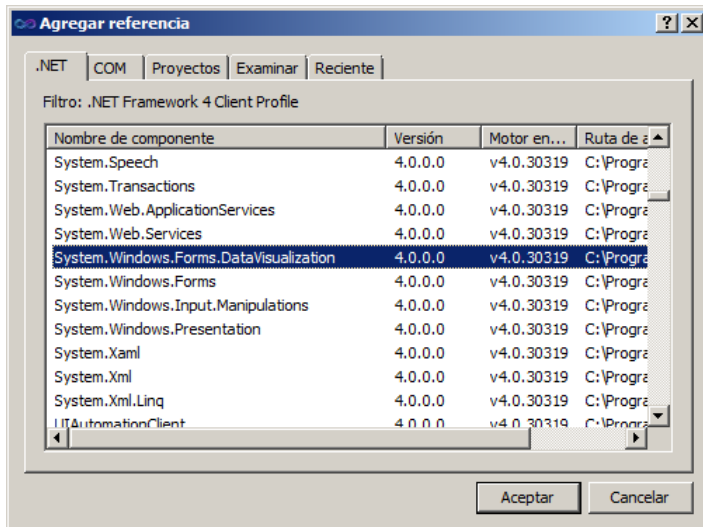
- El diseño del formulario debe quedar similar al gráfico 4.65:

Gráfico 4.65: Diseño del formulario Lista Productos



- Hacer una referencia a la librería de tipos DataVisualization: clic derecho al proyecto, seleccionar "Add Reference" y luego en la ficha "NET" seleccionar "System.Windows.Forms.DataVisualization" como se ve en la figura 4.66.

Gráfico 4.66: Diálogo de Agregar Referencia a DataVisualization



- Escribir el siguiente código para el formulario:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.Windows.Forms.DataVisualization
Imports System.Windows.Forms.DataVisualization.Charting
```

```
Public Class frmListaProducto
```

```
    Private lobeProducto As New List(Of beProducto)
```

```
    Private Sub ListarProductos(sender As System.Object, e As  
System.EventArgs) Handles MyBase.Load
```

```
        Dim obrProducto As New brProducto
```

```
        lobeProducto = obrProducto.Listar()
```

```
        dgvProducto.DataSource = lobeProducto
```

```
        dgvProducto.Columns(0).Visible = False
```

```
        dgvProducto.Columns(1).Width = 250
```

```
        dgvProducto.Columns(2).Visible = False
```

```
        dgvProducto.Columns(3).Visible = False
```

```
        dgvProducto.Columns(4).Width = 80
```

```
        dgvProducto.Columns(4).DefaultCellStyle.Alignment = _
```

```
DataGridViewContentAlignment.MiddleRight
```

```
        dgvProducto.Columns(4).DefaultCellStyle.Format = "n2"
```

```
        dgvProducto.Columns(5).Visible = False
```

End Sub

```
Private Sub CrearGráfico(sender As System.Object, e As System.EventArgs)
Handles mnuCrearGráfico.Click
    Dim oArea As New ChartArea("Area")
    Dim oSerie As New Series("Serie")
    With oSerie
        .ChartArea = "Area"
        .ChartType = SeriesChartType.Column
        .XValueMember = "Nombre"
        .YValueMembers = "PrecioUnitario"
        .IsValueShownAsLabel = True
        .Color = Color.Red
    End With
    Dim oGrafico As New Chart
    With oGrafico
        .BackColor = Color.Aqua
        .Dock = DockStyle.Fill
        .Titles.Add(String.Format("Gráfico de Productos"))
        .Titles(0).ForeColor = Color.Blue
        .Titles(0).Font = New Font("Arial", 20)
        .ChartAreas.Add(oArea)
        .Series.Add(oSerie)
        .DataSource = lobeProducto
    End With
    Dim frmGrafico As New Form
    With frmGrafico
        .Text = "Grafico"
        .WindowState = FormWindowState.Maximized
        .Controls.Add(oGrafico)
        .ShowDialog()
    End With
End Sub
End Class
```

Importante: Si no se especifica la propiedad ChartArea de la clase Series no se visualiza el gráfico.

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".

- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

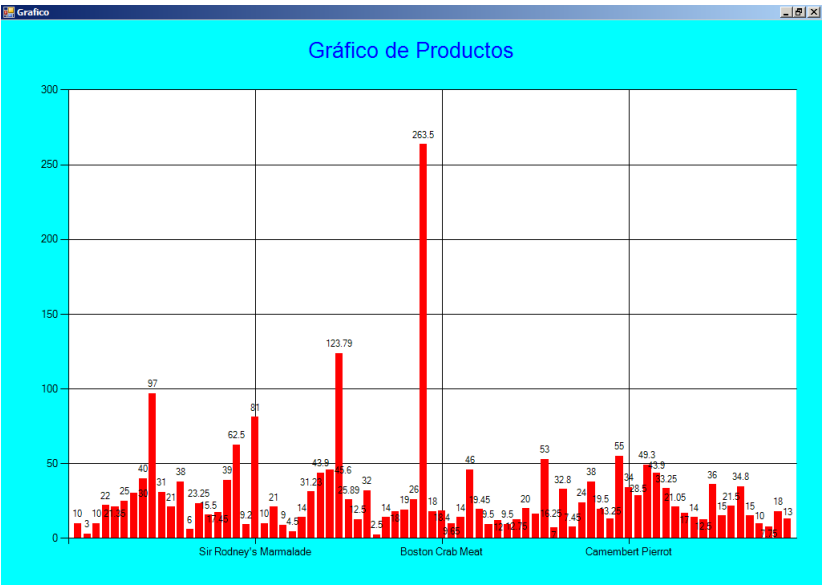
Gráfico 4.67: Ejecución del formulario Lista Productos



Nombre	PrecioUnitario
Chai	10.00
Chang	3.00
Chicha	10.00
Chef Anton's Cajun Seasoning	22.00
Chef Anton's Gumbo Mix	21.35
Grandma's Boysenberry Spread	25.00
Uncle Bob's Organic Dried Pears	30.00
Northwoods Cranberry Sauce	40.00
Mishi Kobe Niku	97.00
Ikura	31.00
Queso Cabrales	21.00
Queso Manchego La Pastora	28.00

- Clic derecho sobre la grilla de productos y se presentará un menú contextual, seleccionar la opción Crear Gráfico.
- Se creará dinámicamente un formulario conteniendo un gráfico de barras de los precios de los productos, similar al que se muestra en la figura 4.68.

Gráfico 4.68: Gráfico de Barras de Precios con el Control Chart



Preguntas de Repaso

1. Cual es la ventaja de crear aplicaciones Windows con Windows Forms?
2. Qué propiedades del formulario deben configurarse para que no pueda modificarse de tamaño?
3. Qué propiedad del formulario debe configurarse para mostrar en el formulario un gráfico en forma de elipse o cualquier forma deseada?
4. En qué evento del formulario debe programarse la creación de un gráfico?
5. Menciona 3 controles básicos de Windows Forms.
6. Qué propiedad del control TextBox permite limitar la cantidad de caracteres ingresados?
7. De qué formas puede crearse un TextBox que permita ingresar una contraseña o password?
8. Qué debe hacerse para que al dar Enter sobre cualquier control de entrada se dispare el procedimiento asociado al clic de un botón?
9. De forma similar, qué debe hacerse para que al pulsar la tecla Escape se dispare el procedimiento asociado al clic de un botón de salida?
10. Qué se debe configurar en un control Label para que muestre el símbolo & como un carácter mas o literal.
11. Menciona 3 controles de listas de Windows Forms.
- 12.Cuál es la propiedad más importante de los controles de listas?
13. Menciona 3 métodos de la propiedad Items para manejar listas.

14. Menciona el evento más importante en el cual generalmente se programa cuando se selecciona un elemento de una lista.
- 15.Cuál es la forma mas recomendada de llenar los elementos de una lista.
16. Si la lista se va a llenar elemento por elemento qué métodos debe usarse antes y después de llenar la lista para evitar escribir cada elemento en pantalla?
17. Menciona los 2 controles de vistas en Windows Forms.
- 18.Cuál es la propiedad más importante del control TreeView?
- 19.Cuál es la propiedad más importante del control ListView?
20. Cuáles son las vistas que puede presentar el control ListView?
21. Cuáles son las propiedades donde se configuran las imágenes a mostrar en un ListView?
22. Cómo se llama el control Windows Forms que permite dividir el formulario en 2 partes, por ejemplo para separar a un TreeView de un ListView?
23. Qué control permite guardar una colección de imágenes?
24. Qué es un formulario MDI y cómo se crea en Windows Forms?
25. Cómo se puede cambiar el fondo de un formulario MDI?.
26. Qué propiedad del formulario MDI padre indica la cantidad de formularios hijos activos?
27. Qué debe configurarse en Visual Studio para trabajar con un formulario Login que al autenticarse se cierre y abra un formulario MDI que cuando se cierre finalice toda la aplicación?.

28. Cuántos tipos de menús hay y cómo se crean en Windows Forms?
29. Qué propiedad es necesario configurar de un control para que se muestre un menú contextual?
30. Cómo se puede cargar dinámicamente un formulario teniendo como dato su nombre?
31. Qué tipo de control permite agregar cualquier control como opción o elemento de un menú?
32. Cómo se llama el control de Windows Forms que muestra un calendario con los días de un mes?
33. Menciona los 5 diálogos comunes de Windows que son componentes de Windows Forms.
34. Cómo se llama el método común que tienen todos los diálogos de Windows que permiten mostrar el diálogo en pantalla?
35. Qué propiedades comunes tienen los diálogos de archivos OpenFileDialog y SaveFileDialog en Windows Forms?
- 36.Cuál es la propiedad más importante del diálogo de color?
- 37.Cuál es la propiedad más importante del diálogo de fuente?
38. Cómo se llama el diálogo que permite mostrar directorios?
39. Menciona las barras que se pueden crear en Windows Forms.
40. Cómo se llama la colección de elementos que tienen ambas barras en Windows Forms?
41. Qué tipo de ToolStripItem tiene la Barra de Herramientas que no tiene la Barra de Estado?

42. Cuál es el control mas usado para presentar una lista de datos en Windows Forms?
43. Menciona 3 propiedades Allow del control DataGridView de Windows.
44. Qué pasos debe realizar para crear un DataGridView con columnas personalizadas?
45. Menciona los 5 tipos de columnas personalizadas que puede tener el DataGridView.
46. Cómo se llama la propiedad de la columna personalizada que especifica que dato (campo o propiedad) vamos a presentar?
47. Qué pasos debe realizar para crear un DataGridView con una columna que muestre una imagen?
48. En qué evento del DataGridView hay que programar para formatear las columnas del control?
49. Cómo se configura el tamaño de la imagen mostrada en la columna del DataGridView?
50. En qué evento del DataGridView hay que programar para personalizar las cabeceras del control?
51. Cómo se crea un gráfico personalizado dentro de una columna del control GridView?
52. Para qué se usa el modo Virtual del DataGridView?
53. Qué propiedades del DataGridView hay que configurar para crear una paginación de datos?
54. En qué evento del DataGridView hay que programar para mostrar los valores de las celdas en una paginación?

55. Qué método del DataGridView hay que invocar para actualizar los valores de las celdas mostradas en la paginación?
56. Cómo se clasifican los controles que se pueden crear en Windows?
57. Qué tipo de control crearía si desea aumentar las características de un control existente en Windows Forms?
58. Qué tipo de control crearía si desea usar varios controles de Windows?
59. De qué clase hereda el control personalizado o dibujado?
60. Mencione 3 formas de crear un reporte en Windows Forms.
61. Para qué se usa la clase PrintDocument y en qué espacio de nombres se encuentra?
62. En qué evento de la clase PrintDocument se tiene que escribir código para crear la pagina a imprimir?
63. Menciona el control y los 3 diálogos que se pueden asociar a la clase PrintDocument para crear reportes.
- 64.Cuál es la diferencia entre el control PrintPreviewControl y el diálogo PrintPreview al crear la vista preliminar del reporte?
65. En qué consiste los Informes de Microsoft y cuáles son sus dos componentes principales?
66. Para qué sirve el control ReportViewer?
67. Cuáles son las barras de herramientas que se usan en el diseñador de informes de Microsoft.
68. Qué propiedades del control ReportViewer deben configurarse para mostrar un reporte de datos?

- 69. A qué formatos puede exportar el ReportViewer?
- 70.Cuál es la diferencia entre crear un gráfico en Excel o usar el control Chart?
- 71. Qué propiedades son las más importantes de la clase Chart?
- 72. En qué espacio de nombres se ubica la clase Chart?

Capítulo 5: Desarrollando Aplicaciones Web con ASP .NET

En este capítulo abordaremos la creación de aplicaciones o sitios Web mediante ASP .NET usando el Formulario y los Controles Web.

Al inicio veremos como crear un simple sitio web usando Formularios y Controles Web sobre todo los controles web estándares simples o intrínsecos, también aprenderemos como trabajar con imágenes y usar controles de validación, además veremos como subir archivos desde la pagina al servidor web con el control FileUpload.

En la segunda parte de este capítulo mejoraremos el diseño del sitio web usando Hojas de Estilo en Cascada (CSS) y aumentaremos las características de navegabilidad usando controles de navegación, controles de vistas y creando páginas principales y de contenido.

En la tercera parte se verá en detalle el control web GridView, iniciando por personalizar sus columnas, paginar registros, ordenar columnas, realizar un mantenimiento sobre el mismo control: adicionar, actualizar y eliminar registros de una BD usando objetos.

Como cuarto tema veremos como crear plantillas en controles enlazados a datos como el Repeater y DataList, esto nos permitirá personalizar la apariencia del control a la forma que deseemos, por ejemplo crearemos una plantilla jerárquica para mostrar una grilla dentro de otra.

1. Trabajar con el Formulario y los Controles Web

La forma más fácil de crear una aplicación web en ASP .NET es usar los Formularios y Controles Web que son clases ubicadas dentro del espacio de nombres: System.Web.UI para Page y System.Web.UI.WebControls para los controles Web.

Para crear el formulario y los controles web podemos usar cualquier editor de texto y escribir código en Visual Basic, pero para realizar más rápido el desarrollo se usa el Visual Studio .NET el cual para la creación de sitios web es conocido como: Visual Web Developer.

Antes de iniciar con la creación de aplicaciones web daremos una introducción a ASP .NET para conocer mejor sus características y ventajas, conoceremos la clasificación de los controles en ASP .NET y también entenderemos el ciclo de vida de una página creada mediante formularios web en ASP .NET.

1.1. Introducción a ASP .NET

ASP .NET es un modelo de desarrollo unificado que es parte del .NET Framework y permite crear de forma fácil y con un mínimo de código Aplicaciones Web, es decir, aplicaciones para Internet que se usen desde cualquier Browser.

Para obtener mas información sobre ASP NET ver la referencia 30 al final del libro.

Características de ASP .NET

- Usa las Librerías de Clases del .NET Framework (BCL), que contienen miles de tipos para ser usados, entre clases, interfaces, estructuras, etc.
- Para escribir código del servidor se puede usar cualquier Lenguaje .NET disponible, entre ellos Visual Basic, C#, J#, Cobol, Delphi, etc.

- El código esta totalmente compilado en el servidor.
- Las páginas web de ASP .NET que usan Controles Web del servidor se pueden ver en cualquier Browser porque generan HTML y javascript en el cliente.
- El resultado de la página se puede guardar en Cache para una mayor velocidad de respuesta.
- No solo se puede crear aplicaciones web, sino también aplicaciones web móviles y servicios web.
- Para configurar la aplicación web se puede realizar mediante el Administrador del IIS pero también mediante el archivo Web.config
- Para manejar eventos de página, sesión, aplicación, etc, no solo se puede hacer creando controladores o módulos HTTP sino también se puede programar en los eventos del archivo Global.asax.
- Para asegurar la aplicación web se puede hacer mediante el IIS, Formularios Web o los servicios de Passport.

Formularios y Controles en ASP .NET

Un Formulario Web es el contenedor de controles web que se envían desde el cliente al servidor y que guardan el estado o los valores de los controles de entrada enviados, también tiene la capacidad de ejecutar código del lado del servidor.

Un Formulario Web tiene controles del servidor los cuales se clasifican en:

- Controles HTML del Servidor: Son similares a los controles HTML del cliente sino que tienen el atributo `runat="server"` que indican que también pueden ejecutar código en el servidor.
- Controles Web: Son controles especiales creados por Microsoft para facilitar el trabajo de crear aplicaciones web y se clasifican en:

➤ Controles Estándares

- Controles de Datos
- Controles de Validación
- Controles de Navegación
- Controles de Inicio de Sesión
- Controles de Informes
- Controles de Servidor AJAX
- Controles de Elementos Web
- Controles de Datos Dinámicos

Nota: Los controles HTML se encuentran en `System.Web.UI.HtmlControls` y los controles Web se encuentran en `System.Web.UI.WebControls`.

Ciclo de Vida de una Página Web en ASP .NET

Desde que un usuario llama a una pagina ASP .NET hasta que se presenta el resultado en el Browser ocurren una serie de actividades que es necesario conocer para entender el funcionamiento de una página y poder programar adecuadamente ésta, estas actividades son:

- Por primera vez se solicita (Request) una página ASP .NET (aspx) del sitio web a través de un Browser mediante un protocolo, por ejemplo: Http.
- El Parser de ASP .NET analiza la sintaxis de todas las etiquetas de controles web del servidor: `<asp:control runat="server">` y si alguna tuviera un error se muestra una excepción de parser en el cliente.
- El código del servidor de la página escrito en algún lenguaje .NET como Visual Basic es compilado por el Compilador del Lenguaje .NET por ejemplo para Visual Basic es el VBC.exe y si alguna instrucción tuviera error se muestra una excepción de compilación en el cliente.

- Al compilar el código de alto nivel se crea un ensamblado en lenguaje de nivel intermedio de Microsoft (MSIL).
- El compilador JIT del CLR convierte el código intermedio en bajo nivel creando un ensamblado nativo.
- El CLR ejecuta el código nativo y forma la respuesta HTML.
- La respuesta (Response) HTML es enviada al cliente.
- El cliente recibe el HTML y el parser analiza si la sintáxis de la página esta bien formada y si no lo esta se muestra un error.
- Una vez listo la página el Browser lo presenta.
- En el caso de haber secuencias de comandos (scripts) de lado del cliente, el Motor de Ejecución de Scripts ejecuta el respectivo código.
- Si se devuelve la página al servidor (PostBack) a través de un botón por ejemplo o cualquier control de entrada con propiedad AutoPostBack en true, la pagina ya no se parsea ni se compila, por estar en memoria.
- El CLR ejecuta nuevamente el código nativo del servidor y forma la respuesta HTML que es enviada al cliente, y asi se vuelve a repetir el ciclo antes mencionado solo los últimos pasos.

Finalmente, es necesario entender que existen 2 procesos bien diferentes que son el cliente compuesto por el Browser, por ejemplo el Internet explorer (Iexplore.exe) y el Servidor Web de Microsoft que es el Internet Information Server (IIS) cuyo proceso se encuentra en InetInfo.exe.

Una aplicación web en ASP .NET puede tener código que se ejecuta en el cliente o en el servidor, para el cliente podemos usar Javascript, DOM de HTML, JQuery, AJAX, etc y para el servidor podemos usar Formularios Web, Controles Web, Lenguajes .NET, Librerías de Clases de .NET (BCL), etc.

1.2. Creando un Simple Sitio Web

Para crear un sitio web en ASP .NET con Visual Web Developer se puede crear en 3 ubicaciones distintas:

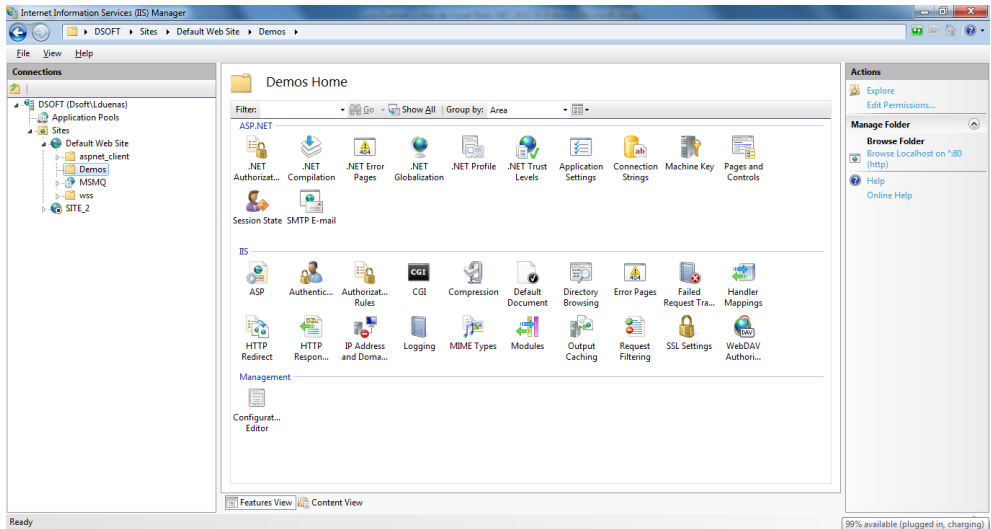
- Sistema de archivos (File System): No necesita tener instalado el servidor web IIS, sino trabaja en forma local con un servidor virtual.
- HTTP: Necesita tener instalado localmente un servidor web IIS y es el mas recomendable para desarrollar aplicaciones para Internet.
- FTP: Crea un sitio web en cualquier servidor remoto disponible mediante FTP, se usa cuando no tenemos un servidor propio y alquilamos uno (Hosting).

En todos los ejemplos de este libro usaremos la segunda opción: HTTP para lo cual primero es necesario tener instalado el IIS antes de instalar el Visual Studio .NET por lo que registra unos scripts de ASP .NET en el IIS. Además crearemos en el directorio por defecto de IIS: C:\inetpub\wwwroot una carpeta llamada Demos para todos los sitios web creados en el libro.

Importante: Si ha instalado el IIS después del Visual Studio y no se puede crear sitios web con ASP .NET No desinstale el Visual Studio, ingrese al explorador de Windows a: C:\Windows\Microsoft.NET\Framework64, luego en la carpeta v2.0.50727 arrastre el archivo aspnet_regiis.exe al diálogo de ejecutar (Run) de Windows (tecla Windows + R) y al final aumente -i para instalar ASP .NET 2. También hay que ingresar a la carpeta v4.0.30319 y registrar el mismo archivo para ASP .NET 4.

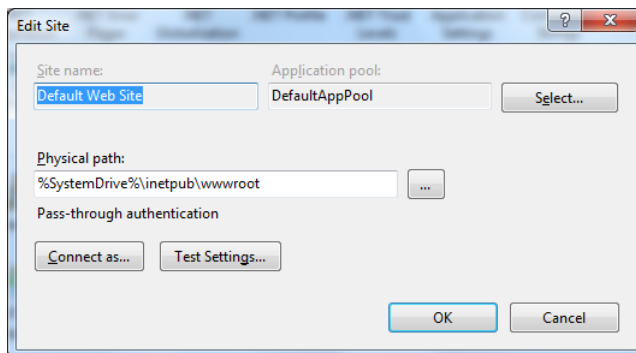
Una vez instalado el IIS y registrado ASP .NET en IIS puede ingresar al Administrador de IIS para ver los sitios web que va a crear, del menú "Inicio", seleccionar "Panel de Control", "Herramientas Administrativas" y luego "Administrador de Servicios de Información de Internet", y se verá una ventana similar a la mostrada en la figura 5.1

Gráfico 5.1: Ventana del Administrador de IIS



Seleccione el nodo "Default Web Site" y en el menú "Actions" del lado derecho clic en "Basic Settings" para ver la configuración del directorio raíz, y se mostrará un diálogo similar a la siguiente figura.

Gráfico 5.2: Diálogo de Configuración Básica Sitio Web x Defecto



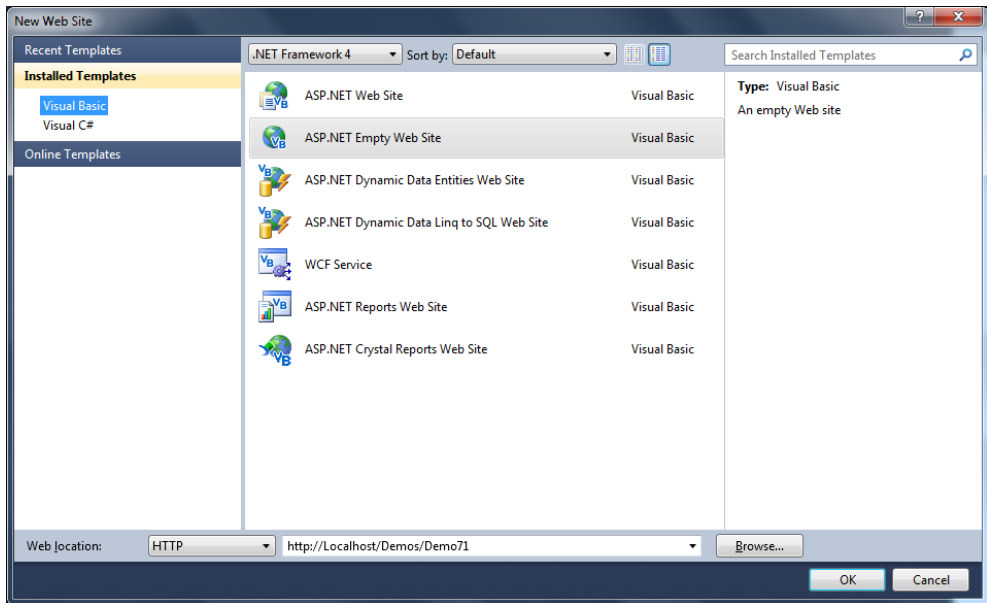
Nota: Por defecto todos los sitios web creados heredarán la configuración del directorio raíz del IIS, aunque se pueden cambiar por aplicación.

Ahora si, estamos listos para crear nuestro primer sitio web como lo muestra el Demo 71 que tiene 3 páginas, la primera es un menú con 2 enlaces: uno a la primera página y el otro a la segunda.

Demo 71

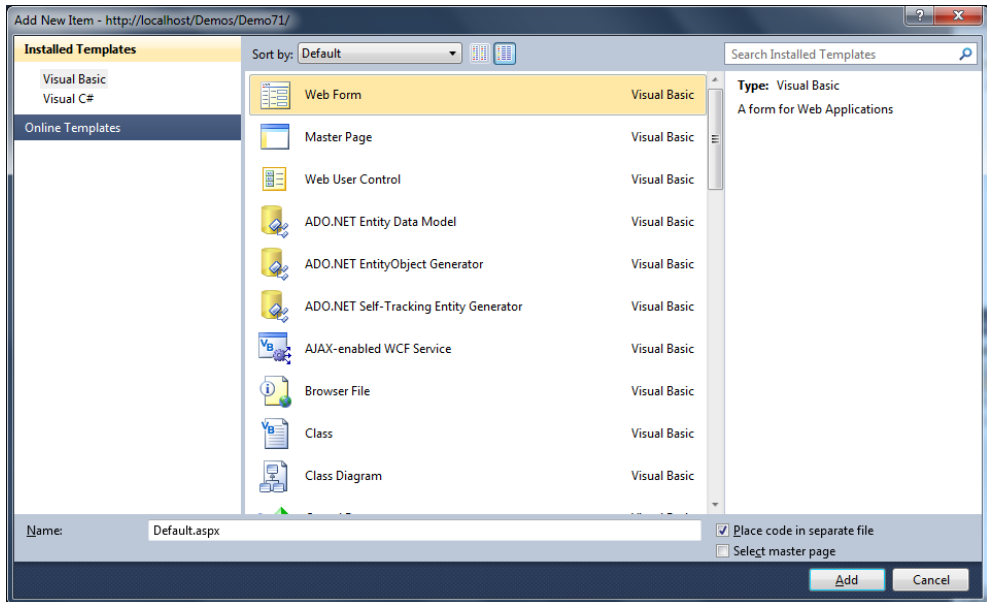
- Del menú "File", seleccionar "Add", "New Web Site" y aparecerá una ventana similar a la mostrada en la figura 5.3.

Gráfico 5.3: Ventana de Creación de un Nuevo Sitio Web



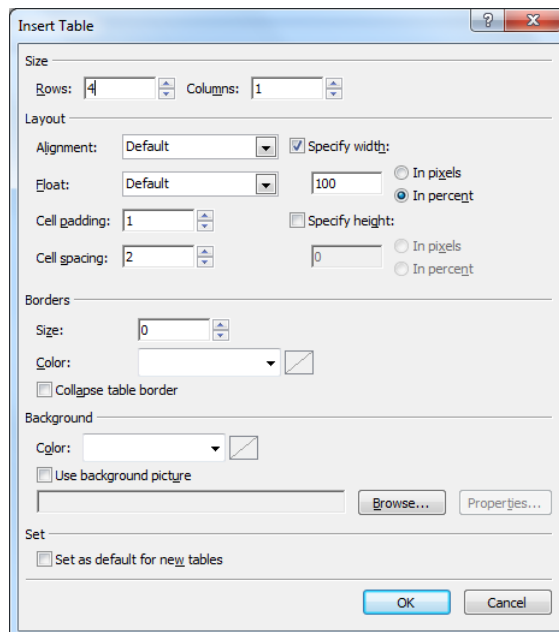
- En el lado izquierdo seleccionar como lenguaje "Visual Basic".
- En el lado derecho seleccionar como plantilla de proyecto: "ASP .NET Empty Web Site".
- En el lado inferior elegir como ubicación HTTP y en la dirección escribir: `http://localhost/Demos/Demo71` y clic en el botón "OK".
- Se creará un proyecto de sitio web vacío que solo trae un archivo de configuración: `web.config`.
- Agregar una nueva página de formulario web: del menú "Website" seleccionar "Add New Item" y se mostrará una ventana similar al de la figura 5.4.
- Seleccionar la plantilla "Web Form" y dejarlo con el nombre sugerido "Default.aspx", clic en el botón "Add".

Gráfico 5.4: Ventana de Creación de un Nuevo Item



- Ubicarse dentro del div de la página e insertar una tabla: del menú "Table" seleccionar "Insert Table" y se verá el diálogo de insertar tabla.

Gráfico 5.5: Diálogo de Insertar Tabla



- Configurar que se necesita 4 filas y 1 columna y clic en el botón "OK".
- En la celda de la primera fila escribir el literal: "Un Simple Sitio Web en ASP .NET".
- En la celda de la segunda fila escribir otro literal: "Seleccione una página".
- En la celda de la tercera fila arrastrar el control "Hiperlink" del cuadro de herramientas y configurar su propiedad ID a "hlkPrimera" y su propiedad Text a "Primera página".
- En la celda de la cuarta fila también arrastrar otro "Hiperlink" y configurar su propiedad ID a "hlkSegunda" y su propiedad Text a "Segunda página".
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item" y llamarle "Primera.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 2 filas por 1 columna.
- En la celda de la primera fila escribir el literal: "Esta es la Primera Página".
- En la celda de la segunda fila arrastrar un control Hiperlink y configurar su propiedad ID a "hlkRegresar", su propiedad Text a "Regresar" y su propiedad NavigateUrl a "Default.aspx".
- Agregar otra nueva página: del menú "Website", seleccionar "Add New Item" y llamarle "Segunda.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 2 filas por 1 columna.
- En la celda de la primera fila escribir el literal: "Esta es la Segunda Página".
- En la celda de la segunda fila arrastrar un control Hiperlink y configurar su propiedad ID a "hlkRegresar", su propiedad Text a "Regresar" y su propiedad NavigateUrl a "Default.aspx".

- Regresar a la primera página (Default.aspx) y configurar la propiedad NavigateUrl de los 2 controles Hiperlink: hlkPrimera apuntando a Primera.aspx y hlkSegunda apuntando a Segunda.aspx.
- Finalmente, para probar la página dar clic derecho sobre la página Default.aspx en la ventana del explorador de soluciones y del menú contextual seleccionar "View in Browser".

1.3. Usando Controles Web Intrínsecos

Los Controles Web Intrínsecos son controles web estándares que al momento de convertirse en HTML cada uno tiene una equivalencia a un solo control HTML, lo cual lo podemos representar en la siguiente tabla:

Control Web Intr.	Control HTML generado
BulletList	..<
Button	<input type="submit" ../>
Checkbox	<input type="checkbox" ../>
DropDownList	<select><option>..<</option></select>
FileUpload	<input type="file" ../>
HiddenField	<input type="hidden" ../>
HiperLink	..<
Image	
Label	
ListBox	<select size="4"><option>..<</option></select>
Literal	Literal
Panel	..<
RadioButton	<input type="radio" ../>
Table	<table><tr><td>.. <td>< table><="" td="" tr><=""></td><>
TextBox TextMode=SingleLine	<input type="text" ../>
TextBox TextMode=MultiLine	<textarea rows="filas" cols="columnas" ../>
TextBox TextMode=Password	<input type="password" ../>

Nota: De la lista hay controles HTML que son compuestos como la lista de viñetas , las listas <select> y las tablas <table> los cuales tienen mas elementos pero que forman parte del elemento padre como si fueran uno solo.

A continuación un ejemplo de cómo usar controles web estándares intrínsecos, para registrar el curso seleccionado por un alumno en un archivo de texto. En este ejemplo usaremos los controles Label, TextBox, DropDownList y Button. También veremos como validar usando Javascript.

Demo 72

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo72`.
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", luego "Web Form" y llamarle "FichaAlumno.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 5 filas por 2 columnas.
- Diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celdas 1 y 2 combinadas	Text Size Td Align	Ficha del Alumno 6 Center
Literal 2 fila 2, celda 1	Text Td width	Ingresa el Nombre 20%
TextBox1 fila 2, celda 2	ID Width Td width	txtNombre 300px 80%
Literal 3 fila 3, celda 1	Text Td width	Selecciona el Curso 20%
DropDownList fila 3, celda2	ID Td width	ddlCurso 80%
Literal 4 fila 4, celda 1	Text Td width	Seleccione el Turno 20%
RadioButtonList fila 4, celda 2	ID RepeatDirection Td width	rb1Turno Horizontal 80%
Button1 fila 5, celdas 1 y 2 combinadas	ID Text OnClick	btnRegistrar Registrar <code>return(ValidarDatosAlumno());</code>

- El diseño de la página debe quedar similar al gráfico 5.6:

Gráfico 5.6: Diseño de la Página Ficha del Alumno

- Cambiar de la vista "Design" a la vista "Source" y escribir dentro de la sección "Head" el siguiente código para validar el ingreso de datos:

```
<script language="javascript" type="text/javascript">
    function ValidarTexto(Text, Mensaje) {
        var Texto = document.getElementById(Text);
        if (Texto != null) {
            if (Texto.value.replace(/^\s+|\s+$/g, "").length == 0) {
                alert('Ingresa ' + Mensaje);
                Texto.focus();
                return false;
            }
        }
        return true;
    }

    function ValidarCombo(Comb, Mensaje) {
        var Combo = document.getElementById(Comb);
        if (Combo != null) {
            if ((Combo.value == ' ') || (Combo.value == '') ||
                (Combo.value == '0')) {
                alert('Selecciona ' + Mensaje);
                Combo.focus();
                return false;
            }
        }
        return true;
    }

    function ValidarRadioButton2Opciones(rbl, mensaje) {
        var rbl1 = document.getElementById(rbl + "_0");
```

```
var rbl2 = document.getElementById(rbl + "_1");
if (rbl1.checked == false && rbl2.checked == false) {
    alert("Selecciona " + mensaje);
    return false;
}
return true;
}

function ValidarDatosAlumno() {
    if (ValidarTexto("txtNombre", "Nombre del Alumno") == false)
        return false;
    if (ValidarCombo("ddlCurso", "Curso a llevar") == false) return false;
    if (ValidarRadioButton2Opciones("rblTurno", "Turno a estudiar") ==
        false) return false;
    return true;
}
```

Nota: Se ha creado 4 funciones de Javascript, 3 son genéricas y una específica, las genéricas nos permiten validar la primera cualquier texto obligatorio, la segunda que se seleccione una opción de un combo y la tercera una opción de un RadioButtonList de 2 opciones. La cuarta función usa las 3 funciones para validar y pasa solo los controles y el mensaje.

- Escribir el siguiente código en la página:

```
Imports System.IO
```

```
Partial Class FichaAlumno
```

```
    Inherits System.Web.UI.Page
```

```
    Protected Sub Page_Load(ByVal sender As Object, _
```

```
        ByVal e As System.EventArgs) Handles Me.Load
```

```
        If Not Page.IsPostBack Then
```

```
            ddlCurso.Items.Add(New ListItem("Seleccione", ""))
            ddlCurso.Items.Add(New ListItem("Windows", "01"))
            ddlCurso.Items.Add(New ListItem("ASP NET", "02"))
            ddlCurso.Items.Add(New ListItem("WPF", "03"))
            ddlCurso.Items.Add(New ListItem("WCF", "04"))
            ddlCurso.Items.Add(New ListItem("WWF", "05"))
            rblTurno.Items.Add(New ListItem("Mañana", "M"))
            rblTurno.Items.Add(New ListItem("Tarde", "T"))
```

```
End If
End Sub

Protected Sub btnRegistrar_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnRegistrar.Click
    Dim Archivo As String = Server.MapPath("Alumnos.txt")
    Using fs As New FileStream(Archivo, FileMode.Append, _
    FileAccess.Write, FileShare.Write)
        Using sw As New StreamWriter(fs)
            sw.WriteLine("{0},{1},{2}", txtNombre.Text, _
            ddlCurso.SelectedItem.Text, rblTurno.SelectedItem.Text)
        End Using
    End Using
    ClientScript.RegisterStartupScript(Page.GetType, "Mensaje", _
    "alert('El alumno ha sido registrado');", True)
End Sub
End Class
```

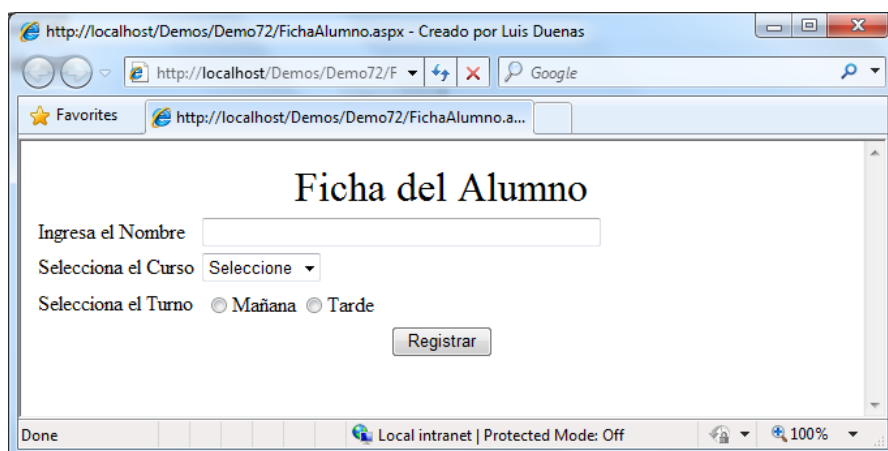
Importante: En el código anterior se usa `Server.MapPath` para devolver la ruta donde se está ejecutando la página actual, es decir la ruta del directorio Demo72 (`C:\inetpub\wwwroot\Demos\Demo72`) ya que si no se especifica la ruta, al trabajar con archivos en ASP .NET, por defecto se lee o se escribe donde está corriendo el proceso de IIS que es "InetInfo.exe" ubicado en el directorio `C:\Windows\System32\inetsrv`.

Nota: Además para leer o escribir un archivo es necesario dar permiso en el Servidor Web al usuario respectivo, en nuestro caso como la seguridad del sitio es por defecto anónima se usa el usuario invitado de Internet: `IIS_IUSRS`. Este debe tener permisos para escribir archivo de lo contrario se generará una excepción.

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Nota: La primera vez que se ejecuta una aplicación se mostrará un mensaje indicando que se harán cambios en el archivo `Web.Config` para depuración, seleccionar el botón "OK".

Gráfico 5.7: Ejecución de la Página Ficha del Alumno

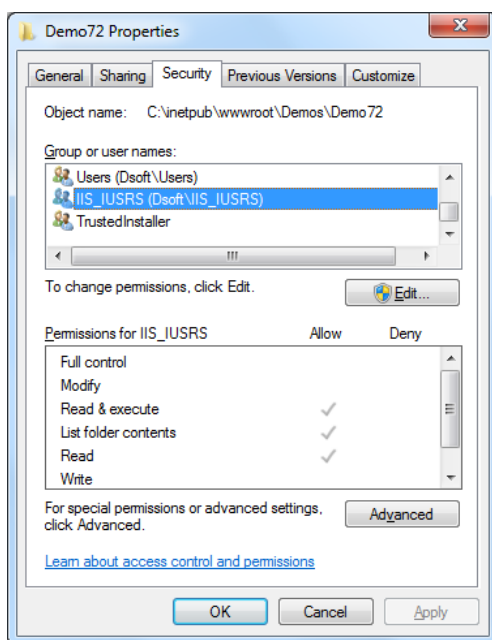


- No ingrese ningún dato y de clic al botón registrar y verá como la aplicación le ira pidiendo ingresar cada dato ya que todos son obligatorios.

Importante: La validación se esta haciendo en el cliente usando Javascript y asi evitamos viajes innecesarios al servidor para hacer validaciones.

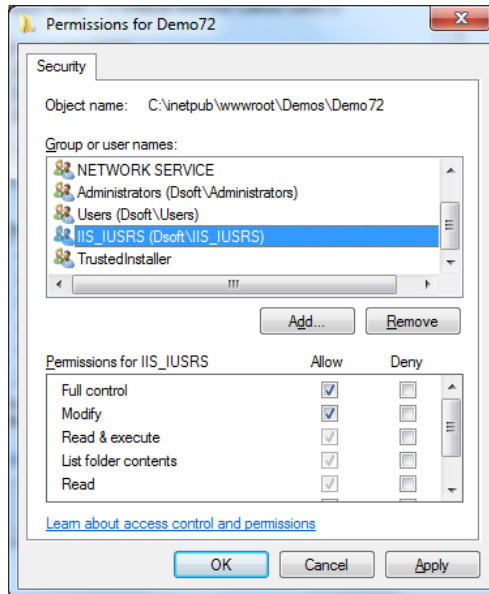
- Ingrese todos los datos y clic al botón "Registrar", ocurrirá una excepción con el siguiente mensaje de error: "Access to the path 'C:\inetpub\wwwroot\Demos\Demo72\Alumnos.txt' is denied."
- Ingrese al Explorador de Windows hasta la carpeta Demo 72 para dar permisos de escritura al usuario invitado de Internet.
- Clic derecho sobre la carpeta y del menú contextual seleccionar "Properties".
- Aparecerá una ventana similar a la mostrada en la figura 5.8.

Gráfico 5.8: Ventana de Propiedades de una Carpeta



- Seleccionar la ficha "Security" y clic en el botón "Edit".
- Aparecerá una ventana similar a la mostrada en la figura 5.9

Gráfico 5.9: Ventana de Permisos de una Carpeta



- Seleccionar el usuario "IIS_IUSRS" y marcar la casilla "Full Control".
- Clic en el botón "OK" y nuevamente intentar registrar alumnos.
- Finalmente, verifique que exista el archivo Alumnos.txt en el directorio Demo72.

1.4. Usando Controles Web de Imágenes

En ASP NET existen 3 controles Web de imágenes que son el Image, el ImageButton y el ImageMap. El primero de ellos el control Image solo muestra una imagen en el cliente, el segundo se usa para mostrar una imagen y programar una acción el servidor en algún evento como Click y el tercero de ellos el ImageMap crea un Mapa (control HTML Map) que define regiones (elemento HTML Area) con diferentes imágenes a mostrar.

El control Image tiene varias propiedades pero la más importante es ImageUrl que indica la ruta de la imagen que se va a mostrar, esta puede configurarse en tiempo de diseño o también en tiempo de ejecución en el caso de crearse el gráfico dinámicamente.

A continuación un ejemplo que crea dinámicamente una imagen para comprobar si el que ingresa a la página es una persona o una aplicación (CAPTCHA: **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part), para lo cual después de crearse el gráfico con 4 caracteres al azar se pide ingresar y se valida mediante Javascript si lo ingresado es igual a lo generado por la aplicación.

Demo 73

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://Localhost/Demos/Demo73`.
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", luego "Web Form" y llamarle "Seguridad.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 5 filas por 1 columna.
- Ingresar a la vista "Source" y modificar la tabla como sigue:

<table style="width:100%;text-align:center">

- Cambiar a la vista "Design" y diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Size	Seguridad mediante CAPTCHA 6
Image1 fila 2, celda 1	Name Height Width	imgCaptcha 80px 200px
Literal2 fila 3, celda1	Text	Ingresa el código generado arriba:
Input Text fila 3, celda1	Id Size	txtClaveUsuario 10
Button1 fila 4, celda 1	ID Text OnClick	btnValidar Validar return(ValidarClave());
HiperLink1 fila 5, celda 1	ID NavigateUrl Text	hlkVerFoto ~/LasVegas.jpg Ver Foto en Las Vegas
HiddenField1 fila 5, celda 1	ID	hdfClaveOriginal

- El diseño de la página debe quedar similar al gráfico 5.10:

Gráfico 5.10: Diseño de la Página Seguridad con CAPTCHA

Seguridad mediante CAPTCHA



Ingresa el código generado arriba:

[Ver Foto en Las Vegas](#)

HiddenField - hdfClaveOriginal

- Cambiar de la vista "Design" a la vista "Source" y escribir dentro de la sección "Head" el siguiente código para validar el ingreso de datos:

```
<script language="javascript" type="text/javascript">
function ValidarClave() {
    var txtClaveUsuario = document.getElementById("txtClaveUsuario");
    var claveOriginal = document.getElementById("hdfClaveOriginal").value;
    if (txtClaveUsuario.value != claveOriginal) {
        alert("Clave incorrecta");
        txtClaveUsuario.focus();
        return false;
    }
    return true;
}
</script>
```

- Escribir el siguiente código en la página:

```
Imports System.Drawing
Imports System.Drawing.Drawing2D
```

```
Partial Class Seguridad
    Inherits System.Web.UI.Page
```

```
Private Function GenerarCaracterAzar() As Char
    Dim oAzar As New Random
    Threading.Thread.Sleep(20)
    Dim N As Integer = oAzar.Next(26) + 65
    Return (Chr(N))
End Function
```

```
Private Sub CrearGrafico()
    Dim rec As New Rectangle(0, 0, 200, 80)
    Dim deg As New LinearGradientBrush(rec, Color.Aqua, _
    Color.Blue, LinearGradientMode.BackwardDiagonal)
    Dim bmp As New Bitmap(200, 80)
    Dim grafico As Graphics = Graphics.FromImage(bmp)
    grafico.FillRectangle(deg, rec)
    Dim clave As String = ""
    For I = 1 To 4
        clave += GenerarCaracterAzar()
    Next
    Dim archivoImg As String = Server.MapPath _
    (String.Format("{0}.jpg", clave))
```

```
hdfClaveOriginal.Value = clave
grafico.DrawString(clave, New Font("Arial", 40), _
Brushes.White, 10, 10)
Dim oAzar As New Random
Dim X1, Y1, X2, Y2 As Integer
Dim R, G, B As Integer
Dim X As Color
Dim Lapis As Pen
For I = 1 To 10
    R = oAzar.Next(255)
    G = oAzar.Next(255)
    B = oAzar.Next(255)
    X = Color.FromArgb(R, G, B)
    Lapis = New Pen(New SolidBrush(X), 2)
    X1 = oAzar.Next(200)
    Y1 = oAzar.Next(80)
    X2 = oAzar.Next(200)
    Y2 = oAzar.Next(80)
    grafico.DrawLine(Lapis, X1, Y1, X2, Y2)
Next
bmp.Save(archivoImg)
imgCaptcha.ImageUrl = String.Format _
("http://Localhost/Demos/Demo73/{0}.jpg", clave)
End Sub
```

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    If Not Page.IsPostBack Then
        CrearGrafico()
    End If
End Sub
```

```
Protected Sub btnValidar_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnValidar.Click
    hlkVerFoto.Visible = True
End Sub
End Class
```

Importante: Para generar un número al azar se usa la clase Random, pero si se llama consecutivamente al método Next este siempre devolvería

el mismo número, para evitar esto se debe hacer una pausa de 20 milisegundos, en este caso usando el método Sleep de la clase Thread.

Nota: Antes de ejecutar la aplicación tiene que dar permisos para escribir el archivo con el Captcha generado al usuario IIS_IUSRS.

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.11: Ejecución de la Página Seguridad CAPTCHA sin Link



- Ingresar un código equivocado y se hará una validación en el cliente mostrando un error.
- Ingresar el código mostrado y se irá al servidor para habilitar el Link similar a lo mostrado en la figura 5.12.

Gráfico 5.12: Ejecución de la Página Seguridad CAPTCHA con Link



- Finalmente, clic al Link para ver la pagina con la foto.

1.5. Usando el Control FileUpload y Controles de Validación

En esta sección revisaremos 2 temas: el primero como subir un archivo desde el cliente al servidor web usando el control Web FileUpload y el segundo tema como validar datos en el cliente usando los controles Web de Validación.

El control FileUpload es la forma más simple de subir un archivo desde la página hacia el servidor Web, tiene propiedades para manejar el archivo que ha sido enviado desde el cliente con el PostBack, tales como HasFile que indica si se ha enviado un archivo, PostedFile que representa al archivo enviado, FileBytes que devuelve un arreglo de bytes con su contenido, FileContent que devuelve un objeto FileStream o FileName que obtiene el nombre del archivo en el cliente.

Además el control FileUpload tiene el método SaveAs que permite guardar el archivo en el disco del servidor web. Por defecto el tamaño máximo del archivo a enviar es de 4MB y el límite es de 4GB, para aumentar el tamaño del archivo configurar la propiedad maxRequestLength del elemento httpRuntime en el archivo de configuración web.Config. Para obtener más información sobre el control FileUpload ver la referencia 31 al final del libro.

Antes de enviar datos al servidor es necesario validar si los datos fueron ingresados por el usuario, si concuerdan con el tipo a recibir y si son válidos, para esto generalmente usamos código en Javascript, pero también podemos usar los controles de Validación de ASP .NET entre los cuales tenemos:

- RequiredFieldValidator: Se usa para validar que el ingreso de un campo sea obligatorio.
- CompareValidator: Valida los datos ingresados por el usuario con respecto a una constante, el valor de otro control o si es de un tipo de dato específico, por ejemplo una fecha, un número entero, etc.
- Rangevalidator: Comprueba que el valor de un control se encuentre en un intervalo dado por las propiedades MinimunValue y MaximunValue.

- **RegularExpressionValidator:** Valida que el valor de un control cumpla con un patrón o expresión regular, por ejemplo una dirección de correo electrónico, una dirección URL, un DNI, un número telefónico, etc.
- **CustomValidator:** Se usa para personalizar la lógica de validación y puede ser en el lado del cliente usando la propiedad **ClientValidationFunction** o en el servidor programando el evento **ServerValidate**.

Todos los controles de validación comparten propiedades comunes como:

- **ControlToValidate:** Especifica el control que se va a validar.
- **Display:** Indica la forma como se presentan los mensajes, que puede ser estática o dinámica (sin dejar espacios).
- **ErrorMessage:** Especifica el mensaje que aparecerá en el control de resumen de validación cuando ocurre un error.
- **Text:** Indica el texto que se muestra en el propio control de validación cuando ocurre un error.

Finalmente, para mostrar el resumen de todos los errores originados por los controles de validación esta el control "ValidationSummary" que tiene propiedades como:

- **DisplayMode:** Especifica el tipo de mensaje, puede ser con viñetas, lista o simple texto.
- **ShowMessageBox:** Si está en true se muestran los errores en un cuadro de mensaje (alert).
- **ShowSummary:** Si está en true (por defecto) se muestran los errores en la página.

Para obtener más información sobre los controles de validación de ASP .NET puede ver la referencia 32 al final del libro.

A continuación un ejemplo sobre una Bolsa de Trabajo que permite a los postulantes registrar su nombre y fecha de nacimiento en un archivo de texto y enviar un archivo con su curriculum y otro con su foto los cuales se guardan en el servidor, previa validación usando controles de ASP .NET,

también podremos visualizar en una tabla todos los datos de los postulantes registrados.

Demo 74

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo74`.
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", luego "Web Form" y llamarle "Menu.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 4 filas por 1 columna.
- Diseñar la página Menú dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Size	Bolsa de Trabajo 6
Literal2 fila 2, celda1	Text	Seleccione una opción
HiperLink1 fila 3, celda1	ID NavigateUrl Text	hIkRegistro Registro.aspx Registro del Postulante
HiperLink12 fila 4, celda 1	ID NavigateUrl Text	hIkLista Lista.aspx Lista de Postulantes

- El diseño de la página Menú debe quedar similar al gráfico 5.13:

Gráfico 5.13: Diseño de la Página Menú

Bolsa de Trabajo

Selecione una Opción

[Registro del Postulante](#)

[Lista de Postulantes](#)

- Agregar otra página: del menú "Website", seleccionar "Add New Item", luego "Web Form" y llamarle "Registro.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 8 filas por 2 columnas.
- Diseñar la página Registro dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celdas 1 y 2 combinadas	Text Size	Bolsa de Trabajo 6
Literal2 fila 2, celda1	Text Td Width	Nombre 20%
TextBox1 fila 2, celda2	ID Width Td Width	txtNombre 200px 80%
RequiredField Validator1 fila 2, celda 2	ID ControlToValidate ErrorMessage Text	rfvNombre txtNombre 1 Ingresa el Nombre 1
Literal3 fila 3, celda1	Text Td Width	Fecha de Nacimiento 20%
TextBox2 fila 3, celda2	ID Width Td Width	txtFechaNac 100px 80%
RequiredField Validator2 fila 3, celda 2	ID ControlToValidate Display ErrorMessage Text	rfvFechaNac txtFechaNac Dynamic 2 Ingresa la Fecha de Nac. 2
Compare Validator1 fila 3, celda 2	ID ControlToValidate Display ErrorMessage	cvFechaNac txtFechaNac Dynamic 3 Fecha inválida

	Text Operator Type	3 DataTypeCheck DateTime
Literal4 fila 4, celda1	Text Td Width	Archivo CV 20%
FileUpload1 fila 4, celda2	ID Td Width	fupCV 80%
RequiredField Validator3 fila 4, celda 2	ID ControlToValidate ErrorMessage Text	rfvCV fupCV 4 Seleccione el CV 4
Literal5 fila 5, celda1	Text Td Width	Archivo Foto 20%
FileUpload2 fila 5, celda2	ID Td Width	fupFoto 80%
RequiredField Validator4 fila 5, celda 2	ID ControlToValidate ErrorMessage Text	rfvFoto fupFoto 5 Seleccione la Foto 5
Button1 fila 6, celdas 1 y 2 combinadas	ID Text Td align	btnRegistrar Registrar center
HiperLink1 fila 7, celdas 1 y 2 combinadas	ID NavigateUrl Text Td align	hlkRegresar Menu.aspx Regresar Center
ValidationSummary1 fila 8, celdas 1 y 2 combinadas	ID DisplayMode ShowMessageBox ShowSummary	vsrRegistro List True False

- El diseño de la página Registro debe quedar similar al gráfico 5.14:

Gráfico 5.14: Diseño de la Página Registro

- Escribir el siguiente código en la página:

Imports System.IO

Partial Class Registrar

Inherits System.Web.UI.Page

Protected Sub btnRegistrar_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnRegistrar.Click

Dim ArchivoTxt As String = _

Server.MapPath("Postulantes.txt")

Using sw As New StreamWriter(ArchivoTxt, True)

sw.WriteLine("{0},{1}", txtNombre.Text, txtFechaNac.Text)

End Using

Dim ArchivoCV As String = _

Server.MapPath(String.Format("CVs/{0}{1}", _
txtNombre.Text, Path.GetExtension(fupCV.FileName)))

fupCV.SaveAs(ArchivoCV)

Dim ArchivoFoto As String = _

Server.MapPath(String.Format("Fotos/{0}{1}", _
txtNombre.Text, Path.GetExtension(fupFoto.FileName)))

fupFoto.SaveAs(ArchivoFoto)

End Sub

End Class

Nota: Antes de ejecutar la aplicación debe crear 2 carpetas para almacenar los archivos: CVs y Fotos y también tiene que dar permisos para escribir en dichas carpetas los archivos al usuario IIS_IUSRS.

- Agregar otra página: del menú "Website", seleccionar "Add New Item", luego "Web Form" y llamarle "Lista.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 3 filas por 1 columna.
- Diseñar la página Lista dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Size	Lista de Postulantes 6
Literal1 (Control) fila 2, celda1	ID	litPostulante
HiperLink1 fila 3, celda 1	ID NavigateUrl Text	hlkRegresar Menu.aspx Regresar

- El diseño de la página Lista debe quedar similar al gráfico 5.15:

Gráfico 5.15: Diseño de la Página Lista



- Escribir el siguiente código en la página:

```
Imports System.IO
```

```
Partial Class Listar
```

```
    Inherits System.Web.UI.Page
```

```
    Protected Sub Page_Load(ByVal sender As Object, _
```

```
        ByVal e As System.EventArgs) Handles Me.Load
```

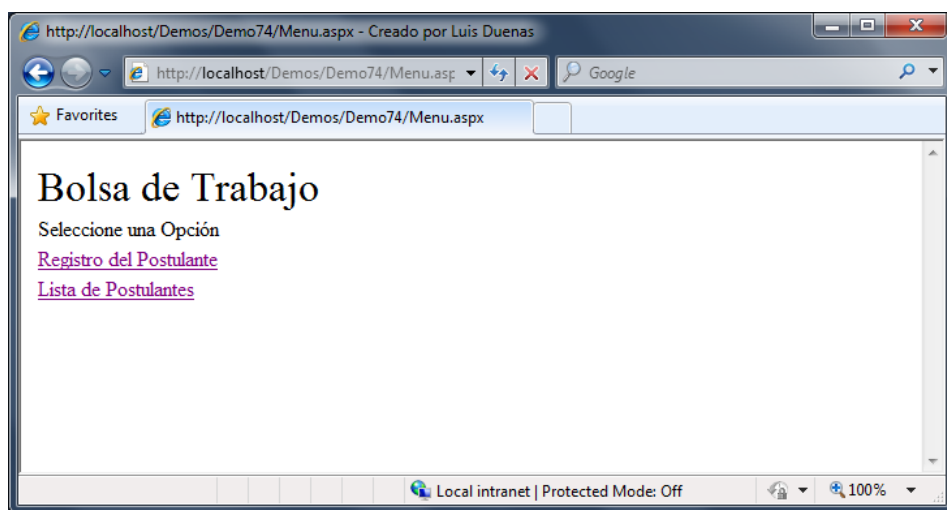
```
        If Not Page.IsPostBack Then
```

```
Dim sb As New StringBuilder
sb.AppendLine("<table>")
sb.AppendLine("<tr bgcolor='blue'>")
sb.AppendLine("<td style='width:200px'><font color='white'>Nombre  
del Postulante</font></td>")
sb.AppendLine("<td style='width:100px'><font color='white'>Fecha  
Nac</font></td>")
sb.AppendLine("<td style='width:200px'><font color='white'>CV  
</font></td>")
sb.AppendLine("<td style='width:200px'><font color='white'>Foto  
</font></td>")
sb.AppendLine("</tr>")
Dim ArchivoTxt As String = Server.MapPath("Postulantes.txt")
Dim RutaCV As String = Server.MapPath("CVs/")
Dim ArchivoCV As String
Dim RutaFoto As String = Server.MapPath("Fotos/")
Dim ArchivoFoto As String
Dim Postulante() As String
If File.Exists(ArchivoTxt) Then
    Using sr As New StreamReader(ArchivoTxt)
        Do While Not sr.EndOfStream
            Postulante = sr.ReadLine.Split(",")
            ArchivoCV = String.Format("{0}{1}.docx", RutaCV,  
Postulante(0))
            ArchivoFoto = String.Format("{0}{1}.jpg", RutaFoto,  
Postulante(0))
            sb.AppendLine("<tr>")
            sb.Append("<td style='width:200px'>")
            sb.Append(Postulante(0))
            sb.AppendLine("</td>")
            sb.Append("<td style='width:100px'>")
            sb.Append(Postulante(1))
            sb.AppendLine("</td>")
            sb.AppendLine("<td style='width:200px'>")
            If File.Exists(ArchivoCV) Then
                sb.Append("<a href='http://localhost/Demos/Demo74  
/CVs/'>")
                sb.Append(String.Format("{0}.docx", Postulante))
                sb.Append(">")
                sb.Append(String.Format("{0}.docx", Postulante))
```

```
        sb.Append("</a>")
    End If
    sb.AppendLine("</td>")
    sb.AppendLine("<td style='width:200px'>")
    If File.Exists(ArchivoFoto) Then
        sb.Append("<a href='http://Localhost/Demos/Demo74/
        Fotos/'>")
        sb.Append(String.Format("{0}.jpg", Postulante))
        sb.Append(">")
        sb.Append(String.Format("{0}.jpg", Postulante))
        sb.Append("</a>")
    End If
    sb.AppendLine("</td>")
    sb.AppendLine("</tr>")
Loop
End Using
End If
sb.AppendLine("</table>")
litPostulante.Text = sb.ToString
End If
End Sub
End Class
```

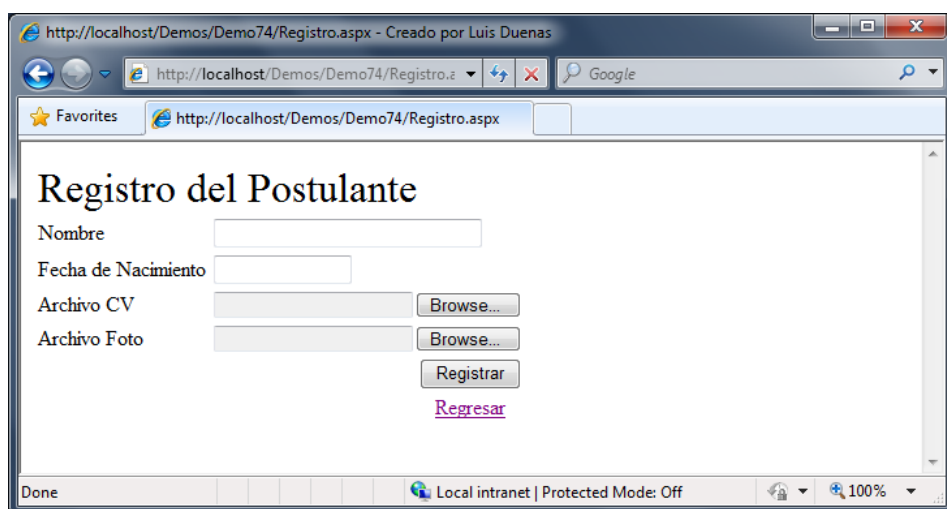
- Configurar la página Menu como pagina de inicio, dando clic derecho sobre esta y del menú contextual seleccionar "Set as Start Page".
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.16: Ejecución de la Página Menú



- Seleccionar la opción Registro del Postulante.

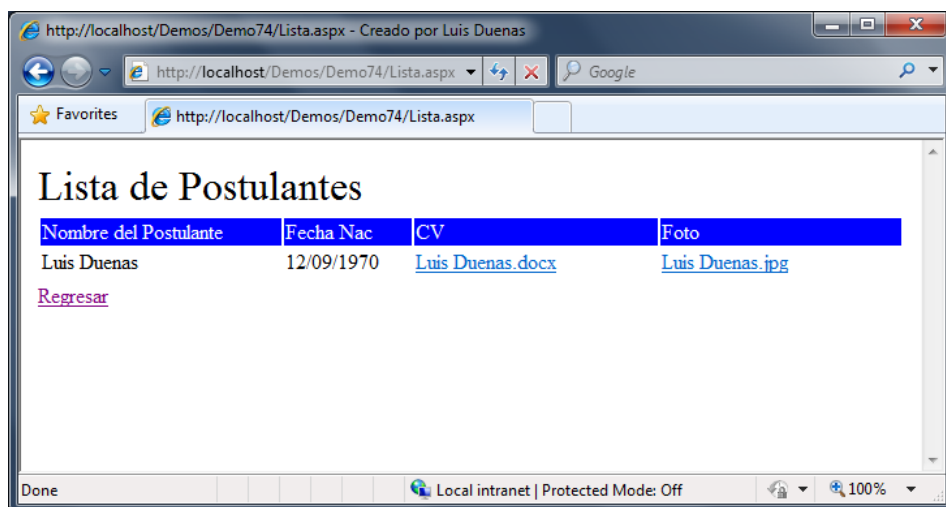
Gráfico 5.17: Ejecución de la Página Registro



- Clic al botón Registrar sin ingresar ningún dato y se presentará un cuadro de mensaje con los errores de ingreso.
- Ingresar el nombre del postulante y su fecha de nacimiento.
- Seleccionar un archivo docx de Word para el CV.

- Seleccionar un archivo Jpg para la foto.
- Clic al botón Registrar y se creará el archivo Postulantes.txt en el servidor y también se grabará 2 archivos en las carpetas CVs y Fotos con el nombre del postulante.
- Agregar todos los postulantes que desee y después clic al enlace de Regresar para ir a la página principal.
- Seleccionar la opción Lista de Postulantes y verá todos los postulantes ingresados.

Gráfico 5.18: Ejecución de la Página Lista



- Finalmente, podrá visualizar el documento de Word o ver la foto del postulante en una página dando clic al enlace dentro de la tabla.

2. Mejorando el Diseño y la Navegabilidad del Sitio Web

Para mejorar el diseño del Sitio Web podemos usar las Hojas de Estilo en Cascada para aplicar un formato común tanto a controles HTML como controles Web y para mejorar la navegabilidad podemos usar Paginas Principales (Master Pages) y Paginas de Contenido (Content Page) usando también en dichas páginas los controles de navegación como el Menú, TreeView y SiteMapPath.

Además si deseamos incorporar Fichas (Tabs) dentro de nuestra aplicación la forma mas fácil de lograrlo es usar los controles de Vistas: MultiView y Views los cuales permite crear varias paginas o vistas en una sola página.

2.1. Creando y usando Hojas de Estilo en Cascada

Las Hojas de Estilos en Cascada (Cascading Style Sheets) o CSS es un lenguaje usado para definir la presentación de una pagina HTML o un documento XML.

Mediante las hojas de estilo podemos simplificar el diseño de las páginas dando un estilo común de acuerdo al elemento, por ejemplo para los titulos, subtítulos, etiquetas, botones, fichas, números, textos, controles de solo lectura, etc.

Las ventajas de usar CSS son que una página puede tener varias hojas de estilo aplicándole el elemento: `<link href="Archivo.css" rel="stylesheet" type="text/css" />`, luego usando el atributo `class` para los controles HTML y la propiedad `cssClass` para los controles Web.

En Visual Web Developer existe una barra de herramientas denominada "Style Sheet" que permite crear y modificar estilos en un archivo CSS usando diálogos y también escribiendo en el editor de código.

A continuación veremos un ejemplo de cómo crear un archivo de hoja de estilos y aplicárselo a la página Ficha del Alumno creada en el Demo 72, esta hoja de estilos tiene 4 clases de estilos: Título, Subtitulo, Botón y un color de Fondo para la página.

Demo 75

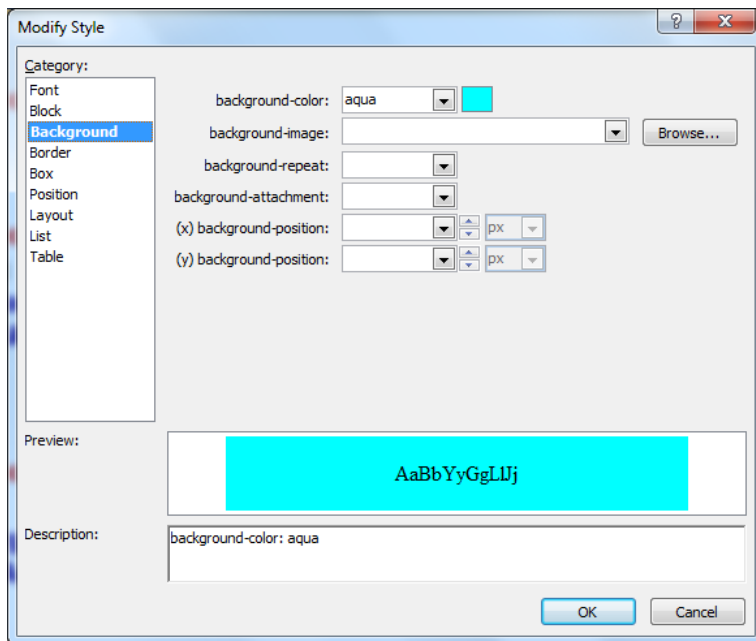
- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo75`.
- Agregar una página existente: del menú "Website", seleccionar "Add Existing Item".
- Ingresar a la carpeta Demo72 y abrir el archivo "FichaAlumno.aspx".
- Agregar un archivo CSS: del menú "Website", seleccionar "Add New Item", luego "Style Sheet" y escribir como nombre "Estilo.css".
- El archivo creado viene con un estilo para el cuerpo de la página llamado Body.
- Mostrar la Barra de herramientas de estilos: clic derecho a cualquier barra de herramientas y seleccionar "Style Sheet".

Gráfico 5.19: Barra de Herramientas de Hojas de Estilo



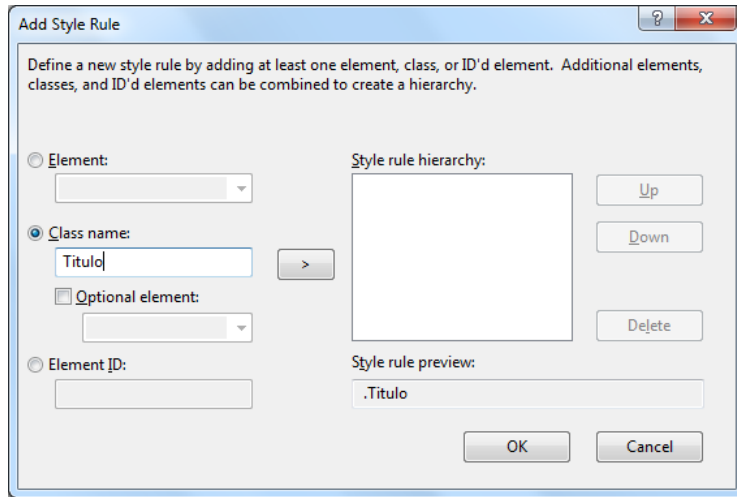
- Clic al segundo botón "Build Style" de la barra para modificar el estilo del Body y aparecerá un diálogo similar al de la figura 5.20.

Gráfico 5.20: Diálogo de Modificar Estilo



- Seleccionar la categoría "Background" y cambiar el color de fondo a "aqua" y clic en el botón "OK".
- Ubicarse debajo del Body y clic al primer botón "Add Style Rule" de la barra de herramientas de estilos.

Gráfico 5.21: Diálogo de Agregar Regla de Estilo



- Seleccionar la opción "Class name" y escribir como nombre "Titulo" y clic al botón ">" para pasarlo a la lista de reglas de estilos y botón "OK"
- Se creará un nuevo estilo llamado Titulo ubicarse dentro de las llaves y clic al segundo botón "Build Style".
- Se mostrará el dialogo visto en la figura 5.20 para modificar el estilo.
- Seleccionar en la categoría "Font" la fuente "Arial Black", en tamaño "x-large" y en color "White".
- Seleccionar en la categoría "Background" el color de fondo "Black".
- Complete los otros estilos para que quede como el siguiente código:

```
body
{
    background-color: aqua;
}
.Titulo
{
    background-color: black;
    color: white;
    font-size: x-large;
    font-family: 'Arial Black';
}
```

```
.Subtitulo
{
    background-color: white;
    color: blue;
    font-size: large;
    font-family: Arial, Helvetica, sans-serif;
}
.Boton
{
    background-color: blue;
    color: white;
}
```

- Grabe el proyecto para que se guarde el archivo con los estilos creados.
- Cierre la ventana de hoja de estilo y abra la pagina ficha del alumno.
- Arrastre el archivo CSS encima de la página y observe que cambia de fondo a "aqua".
- Ingrese a la vista "Source" para verificar que se ha agregado al código HTML: <link href="Estilo.css" rel="stylesheet" type="text/css" />.
- Regrese a la vista "Design" seleccione la primera celda y en la propiedad "Class" de la celda (Td) elija "Titulo".
- Seleccione la celda del literal con el nombre y configure la propiedad "Class" al estilo "Subtitulo".
- También haga lo mismo para las celdas con los literales del curso y turno y configure su propiedad "Class" en "Subtitulo".
- Seleccione el botón Registrar y configure su propiedad "CssClass" en "Boton".
- El diseño de la página debe quedar como se muestra en la figura 5.22

Gráfico 5.22: Diseño de la Página Ficha del Alumno

Ficha del Alumno

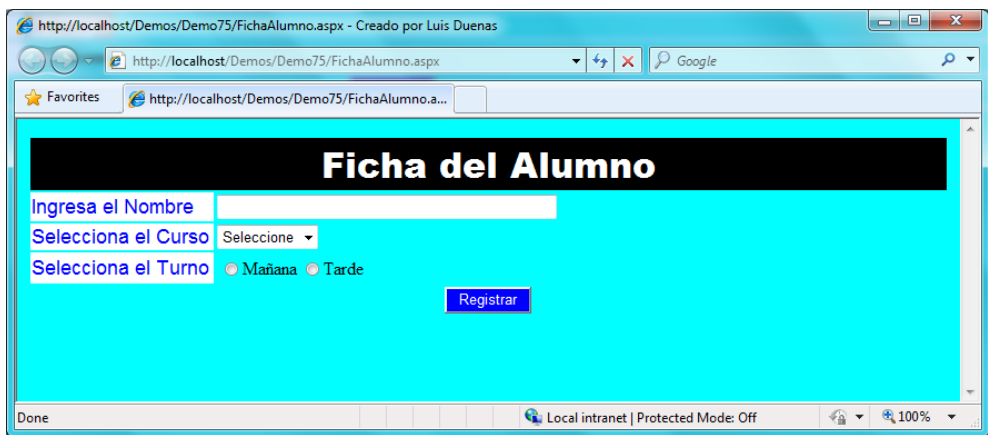
Ingresa el Nombre

Selecciona el Curso

Selecciona el Turno ☐ Unbound

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.23: Ejecución de la Página Ficha del Alumno



Nota: No olvidarse que la aplicación crea un archivo de texto, por tanto hay que dar permiso al usuario IIS_IUSRS.

2.2. Páginas Principales y Controles de Navegación

Las Páginas Principales (Master Page) se usan para crear un mismo diseño o plantilla para varias páginas conocidas como Páginas de Contenido (Content Pages), las cuales se combinan con la página principal en tiempo de ejecución para formar una sola página.

Las páginas principales de ASP .NET son archivos de extensión .master e inician con la directiva de procesamiento @Master, a diferencia de las páginas de formularios web ASP .NET que son archivos de extensión .aspx y contienen la directiva @Page.

Una página principal debe tener como mínimo un control ContentPlace Holder que indica la sección donde irá la página de contenido, por su parte la página de contenido debe tener el atributo MasterPageFile que indica cual es la página principal con la cual se unirá, además tiene un control Content para los Scripts y otro para el contenido de la página.

Las páginas principales permiten ahorrar la escritura de código centralizando en un solo archivo las partes comunes de todas las páginas de contenido, por ejemplo la misma cabecera, el mismo menú, el mismo pie, etc. Para obtener más información sobre Páginas Principales en ASP .NET ver la referencia 33 al final del libro.

Por su parte los Controles de Navegación permiten acceder de forma directa a las páginas del sitio web, entre ellos tenemos:

- SiteMapPath: Permite mostrar la ubicación de la página actual dentro de un mapa de sitio definido en el archivo Web.sitemap, además permite navegar directamente a cualquier página del sitio.
- Menú: Este muestra un conjunto de opciones o elementos a partir de un origen de datos que puede ser un objeto SiteMapDataSource el cual se puede enlazar a un archivo XML o un Mapa de Sitio definido en el archivo Web.sitemap, también lo podemos llenar programáticamente obteniendo los elementos desde una tabla de base de datos, etc.
- TreeView: Visualiza información jerárquica y también puede estar enlazado a un SiteMapDataSource lo cual mostraría las páginas del sitio

web en forma jerárquica o también puede llenarse por código usando la colección de nodos (Nodes).

A continuación un ejemplo que nos demuestra como crear una página principal y varias páginas de contenido, así como también aprenderemos a usar los controles de navegación para navegar por las páginas del sitio web de un instituto que tiene diversas unidades de negocio que ofrecen diversos programas.

Demo 76

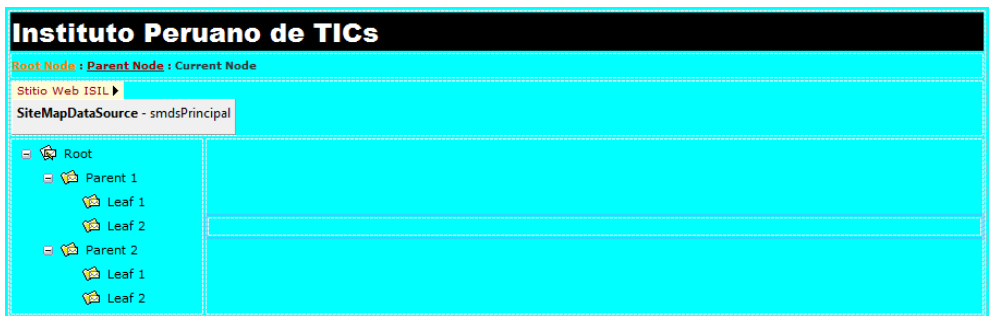
- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://Localhost/Demos/Demo76`.
- Agregar el archivo de hoja de estilo creado en el demo anterior: del menú "Website", seleccionar "Add Existing Item".
- Ingresar al Demo 75 y abrir el archivo "Estilo.css".
- Agregar una nueva página principal: del menú "Website", seleccionar "Add New Item".
- Seleccionar la opción "Master Page" e ingresar como nombre: "Principal.master".
- Arrastrar el archivo css hacia la pagina principal para aplicar el estilo.
- Cortar el control ContentPlaceHolder y estando dentro del div agregar una tabla de 4 filas x 2 columnas.
- Diseñar la página principal dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celdas 1 y 2	Text CssClass	Instituto Peruano de TICs Titulo

combinadas		
SiteMapPath1 fila 2, celdas 1 y 2 combinadas	ID Auto Format	smpPrincipal Colorful
Menu1 fila 3, celdas 1 y 2 combinadas	ID Auto Format Orientation	mnuPrincipal Colorful Horizontal
TreeView1 fila 4, celda 1	ID Auto Format Td width	twvPrincipal Inbox 20%
ContentPlaceHolder1 Fila 4, celda 2	ID Td width	ContentPlaceHolder1 80%

- El diseño de la página Lista debe quedar similar al gráfico 5.23:

Gráfico 5.23: Diseño de la Página Principal



- Agregar un archivo de mapa de sitio: del menú "Website", seleccionar "Add New Item", luego seleccionar "Site Map".
- Modificar el archivo "Web.sitemap" para que quede de la siguiente forma:

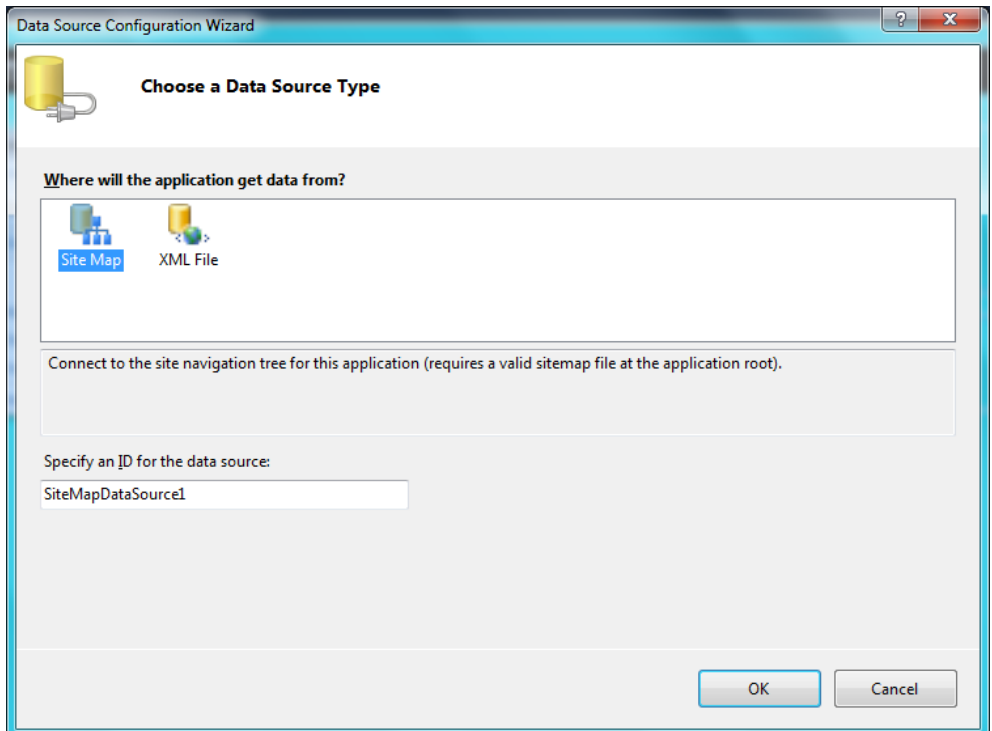
```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="Default.aspx" title="Sitio Web Instituto Peruano de
  TICs" description="Pagina por Defecto">
    <siteMapNode url="Carrera.aspx" title="Carreras" description="Carreras
    del Instituto">
      <siteMapNode url="Computacion.aspx" title="Computación"
      description="Carrera de Computacion"/>
    <siteMapNode url="Marketing.aspx" title="Marketing"
```

```
        description="Carrera de Marketing"/>
    <siteMapNode url="Diseño.aspx" title="Diseño" description="Carrera de
        Diseño"/>
</siteMapNode>
<siteMapNode url="ISILTECH.aspx" title="ISILTECH"
    description="Capacitación para Profesionales">
    <siteMapNode url="PECI.aspx" title="PECI" description="Especializacion
        Desarrollo"/>
    <siteMapNode url="CISCO.aspx" title="CISCO"
        description="Especializacion en Redes"/>
    </siteMapNode>
</siteMapNode>
</siteMap>
```

- Cerrar la ventana del archivo de mapa de sitio.
- Agregar una pagina de contenido: seleccionar la pagina principal y del menú "Website", "Add New Item", "Web Form", pero marcar la casilla inferior "Select Master Page" y escribir como nombre "Default.aspx", luego seleccionar la página "Principal.master".
- Ubicarse dentro del div de la página de contenido Default y escribir el literal: "Página principal del Instituto", configurar la propiedad class a "Subtitulo".
- Agregar una pagina de contenido llamada "Carrera.aspx" y escribir en el div el literal: "Página de Carreras del Instituto", luego configurar la propiedad class a "Subtitulo".
- Agregar una pagina de contenido llamada "Computacion.aspx" y escribir en el div el literal: "Página de la Carrera de Computación", luego configurar la propiedad class a "Subtitulo".
- Agregar una pagina de contenido llamada "Marketing.aspx" y escribir en el div el literal: "Página de la Carrera de Marketing", luego configurar la propiedad class a "Subtitulo".
- Agregar una pagina de contenido llamada "Diseño.aspx" y escribir en el div el literal: "Página de la Carrera de Diseño", luego configurar la propiedad class a "Subtitulo".

- Agregar una pagina de contenido llamada "ISILTECH.aspx" y escribir en el div el literal: "Página de ISILTECH", luego configurar la propiedad class a "Subtitulo".
- Agregar una pagina de contenido llamada "PECI.aspx" y escribir en el div el literal: "Página de los Programas PECI", luego configurar la propiedad class a "Subtitulo".
- Agregar una pagina de contenido llamada "CISCO.aspx" y escribir en el div el literal: "Página del Programa de CISCO", luego configurar la propiedad class a "Subtitulo".
- Cerrar todas las ventanas abiertas y regresar a la página principal.
- Seleccionar el menú, dar clic a la etiqueta inteligente (Smart Tag) y en la opción "Choose Data Source" elegir "New Data Source".
- Se presentará un diálogo similar al mostrado en la figura 5.24.

Gráfico 5.24: Diálogo de Configuración del Origen de Datos

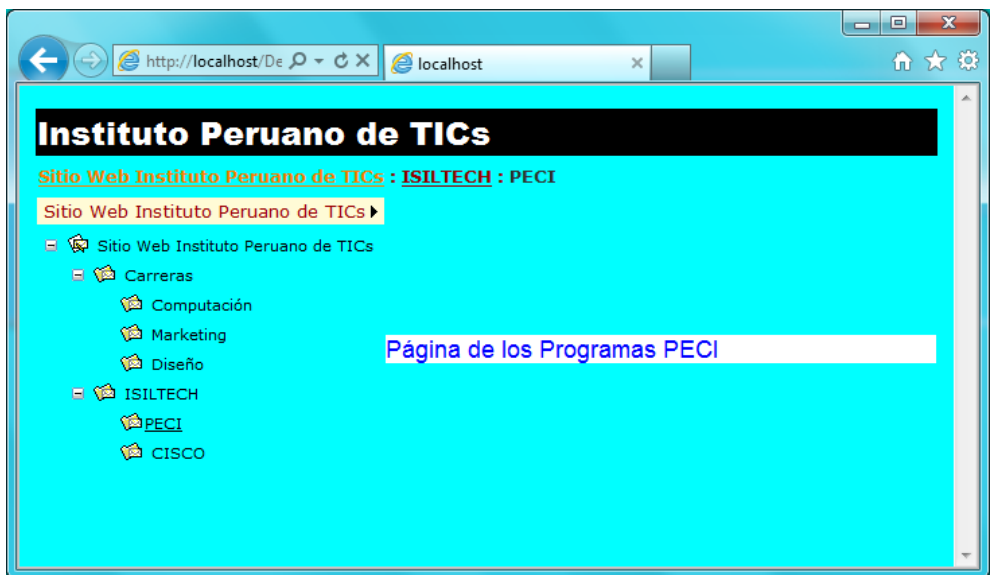


- Seleccionar la opción "Site Map" y asignar como nombre al objeto "SiteMapDataSource" el de "smdsPrincipal".

Nota: Observe como se ha configurado la propiedad DataSourceID del control Menu en el objeto "smdsPrincipal".

- Seleccionar el TreeView y configurar la propiedad DataSourceID al objeto "smdsPrincipal" creado.
- Configurar la página "Default.aspx" como pagina de inicio, dando clic derecho sobre esta y del menú contextual seleccionar "Set as Start Page".
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.25: Ejecución de la Página Principal



- Navegar através del Menú o el TreeView y observar el control SiteMapPath como muestra la ubicación donde nos encontramos, además de poder ir a una cierta página dando clic al enlace mostrado.

Con esto hemos terminado el uso de páginas principales y controles de navegación y como se habrán dado cuenta no hay ninguna línea de programación, pero sin embargo hemos obtenido los resultados esperados

que es hacer el acceso más fácil a los usuarios del sitio web con el mínimo esfuerzo.

2.3. Usando Controles de Vistas MultiView y Views

Cuando la información de una pagina es demasiada se puede agrupar en secciones para lo cual es muy común usar Fichas (Tabs) para ver la información agrupada por categorías.

En ASP .NET podemos implementar esto de 2 formas: la primera es usando una pagina principal con botones que simulen las fichas y al dar clic mostrar la página de contenido, pero el problema es que tendríamos que guardar el estado de cada página en algún lado como por ejemplo en variables de sesión y la programación sería considerable.

La segunda alternativa para trabajar con fichas es usar los controles de vistas MultiView y varios Views, en cada View va la sección que se desea presentar como una página, sin embargo todas las secciones o "paginas" se encuentran en una sola y por defecto se puede conservar el estado sin necesidad de usar variables de sesión u otros objetos ni programar para lograr este fin.

El control MultiView tiene una propiedad `ActiveViewIndex` que inicia en 0 y especifica el índice del control View que tiene la sección o "página" que queremos mostrar.

Esta técnica para implementar fichas es muy interesante pero hay que tener en cuenta que aunque solo se vea una sección en la página internamente se esta guardando el estado de todas las secciones, por lo que el tamaño de la página en el cliente podría ser excesivamente grande y lo mejor es deshabilitar el estado de los controles que no van a usarse o que ya se están cargando directamente de un origen de datos en línea.

En el siguiente ejemplo veremos una consulta de datos de 3 tablas diferentes: Empleados, Categorías y Productos usando un MultiView y 3 Views donde al cargar cada uno se realiza una llamada a reglas de negocio para obtener los datos, por tanto podemos configurar la propiedad `EnableViewState` en falso para que no guarde estado y disminuya el código HTML en el cliente ya que la consulta es en línea y no hay necesidad de guardar el estado o valores de las 3 grillas.

Demo 77

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: http://localhost/Demos/Demo77.
- Hacer una referencia a la librería Northwind.LibBusinessRules.

Nota: En Visual Web Developer al referenciar la principal librería esta copia todas sus librerías dependientes, a diferencia de una aplicación Windows.

- Agregar el archivo de hoja de estilo creado en el demo 75: del menú "Website", seleccionar "Add Existing Item".
- Ingresar al Demo 75 y abrir el archivo "Estilo.css".
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", "Web Form" y escribir como nombre "Consultas.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 3 filas por 1 columna.
- Diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Class	Consultas de Datos Titulo
Button1 fila 2, celda1	ID Text	btnEmpleado Empleados
Button2 fila 2, celda 1	ID Text	btnCategoria Categorías
Button3 fila 2, celda 1	ID Text	btnProducto Productos
MultiView1 fila 3, celda 1	ID	mvConsulta
View1 (dentro del	ID	vwEmpleado

MultiView)		
Literal2 (dentro del View1)	Text	Lista de Empleados
GridView1 (dentro del View1)	ID EnableViewState Auto Format	gvEmpleado False Colorful
View2 (dentro del MultiView)	ID	vwCategoria
Literal3 (dentro del View2)	Text	Lista de Categorías
GridView2 (dentro del View2)	ID EnableViewState Auto Format	gvCategoria False Simple
View3 (dentro del MultiView)	ID	vProducto
Literal4 (dentro del View3)	Text	Lista de Productos
GridView3 (dentro del View3)	ID EnableViewState Auto Format	gvProducto False Oceanica

- El diseño de la página Consultas debe quedar similar al gráfico 5.26:

Gráfico 5.26: Diseño de la Página de Consultas

Consultas de Datos			
Empleados	Categorías	Productos	
			mvConsulta
			vwEmpleado
Lista de Empleados			
Column0	Column1	Column2	
abc	abc	abc	
abc	abc	abc	
abc	abc	abc	
abc	abc	abc	
abc	abc	abc	
			vwCategoria
Lista de Categorías			
Column0	Column1	Column2	
abc	abc	abc	
abc	abc	abc	
abc	abc	abc	
abc	abc	abc	
abc	abc	abc	
			vwProducto
asp:GridView#gvProducto			
Column0	Column1	Column2	
abc	abc	abc	
abc	abc	abc	

- Escribir el siguiente código en la página:

Imports Northwind.LibBusinessEntities

Imports Northwind.LibBusinessRules

Partial Class Consultas

Inherits System.Web.UI.Page

Protected Sub ListarEmpleados(ByVal sender As Object, _

ByVal e As System.EventArgs) Handles vwEmpleado.Load

Dim obrEmpleado As New brEmpleado

Dim lobeEmpleado As List(Of beEmpleado) = obrEmpleado.Listar

gvEmpleado.DataSource = lobeEmpleado

gvEmpleado.DataBind()

End Sub

Protected Sub ListarCategorias(ByVal sender As Object, _

ByVal e As System.EventArgs) Handles vwCategoria.Load

Dim obrCategoria As New brCategoria

Dim lobeCategoria As List(Of beCategoria) = obrCategoria.Listar

```
gvCategoria.DataSource = lobeCategoria
gvCategoria.DataBind()
End Sub
```

```
Protected Sub ListarProductos(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles vwProducto.Load
    Dim obrProducto As New brProducto
    Dim lobeProducto As List(Of beProducto) = obrProducto.Listar
    gvProducto.DataSource = lobeProducto
    gvProducto.DataBind()
End Sub
```

```
Protected Sub Page_Load(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.Load
    If Not Page.IsPostBack Then
        mvConsulta.ActiveViewIndex = 0
    End If
End Sub
```

```
Protected Sub VerConsultaEmpleados(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnEmpleado.Click
    mvConsulta.ActiveViewIndex = 0
End Sub
```

```
Protected Sub VerConsultaCategorias(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnCategoria.Click
    mvConsulta.ActiveViewIndex = 1
End Sub
```

```
Protected Sub VerConsultaProductos(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnProducto.Click
    mvConsulta.ActiveViewIndex = 2
End Sub
End Class
```

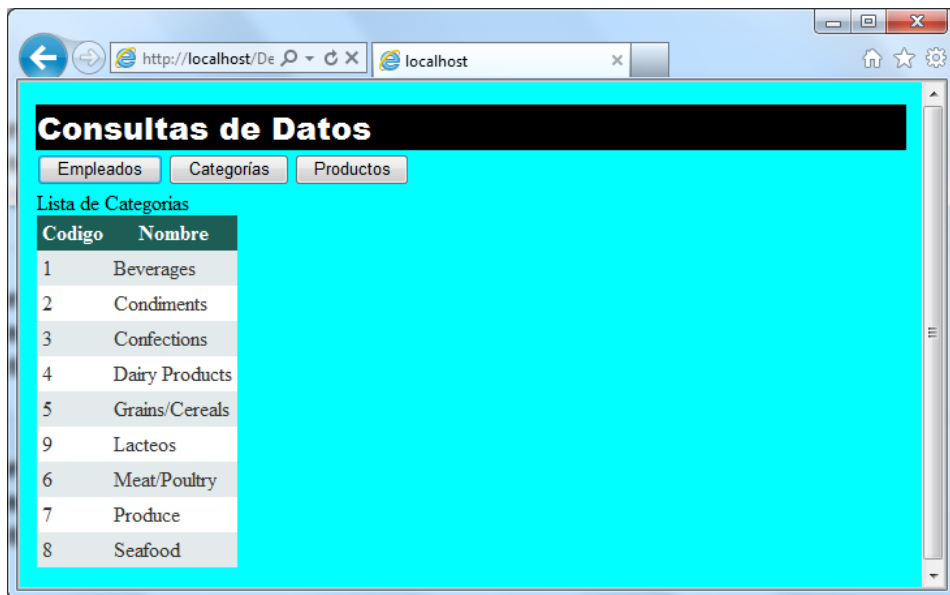
Nota: En el código anterior se ha programado en el evento Load de los controles Views los cuales se disparan cada vez que se llama a la propiedad ActiveViewIndex del control MultiView, en nuestro caso al dar clic a los botones de la parte superior.

- Modificar el archivo de configuración web.config para agregar la clave conNW en la sección appSettings, tal como se muestra:

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" strict="false" explicit="true"
    targetFramework="4.0"/>
  </system.web>
  <appSettings>
    <add key="conNW" value="uid=CursoNET; pwd=123456;
    data source=DSOFT\DSOFT;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.27: Ejecución de la Página de Consultas



- Seleccione cada consulta de datos y vea el código fuente generado en el cliente: clic derecho a la página y del menú contextual seleccionar "View Source".

- Sobre todo observe el campo oculto: "__VIEWSTATE" que guarda el estado de la página, el cual no tiene mucha data.
- Pare la ejecución de la aplicación y cambie la propiedad "EnableViewState" a "true" (valor por defecto) de los controles GridView.
- Ejecute nuevamente la aplicación y vea que el código fuente ha crecido bastante debido a que siempre guarda el estado de las 3 grillas.

3. Usando el Control GridView

Al igual que en Windows Forms, en ASP .NET Web Forms el control mas usado es el GridView, este permite mostrar columnas enlazadas a un origen de datos mediante su propiedad DataSource, también permite paginar configurando la propiedad AllowPaging u ordenar columnas configurando la propiedad AllowSorting, editar una fila mediante la propiedad EditIndex, mostrar una cabecera mediante la propiedad ShowHeader o un pie de pagina mediante la propiedad ShowFooter, etc.

En cuanto a sus principales métodos tenemos el DataBind que ejecuta el enlace de datos, el método SelectRow que permite seleccionar una fila por su índice de fila, el método SetEditRow que permite ingresar al modo edición por el índice de fila y el método SetPageIndex que permite seleccionar una página por su índice de página.

Entre sus principales eventos tenemos los eventos PageIndexChanging y el PageIndexChanged que ocurren mientras y después de cambiar el índice de página, los eventos Sorting y Sorted que ocurren mientras y después que se ordena una columna, los eventos SelectedIndexChanging y SelectedIndexChanged que ocurren al cambiar el índice de fila seleccionada

También existen eventos que se disparan cuando trabajamos con botones de comandos de editar como el RowEditing, o el comando actualizar que dispara los eventos RowUpdating y RowUpdated, el comando de cancelar que dispara el RowCancelEdit, el comando de eliminar que dispara los eventos RowDeleting y RowDeleted.

Finalmente, se encuentran los eventos RowCreated y RowDataBound, el primero se usa para crear dinámicamente controles antes de crearse cada fila y el segundo se usa para dar formato a las filas y ocurre cada vez que se muestra una fila.

En esta parte veremos como personalizar las columnas mostradas, como paginar las filas, como ordenar columnas y finalmente como realizar un mantenimiento dentro del propio control GridView. Para obtener más información sobre el control GridView ver la referencia 34 al final del libro.

3.1. Personalizando Columnas en el GridView

Por defecto cuando se enlaza un control GridView a un origen de datos se muestran todas las columnas del origen de datos ya que se crean en tiempo de ejecución debido a que la propiedad `AutoGenerateColumns` esta en `true`.

Para personalizar las columnas del GridView se puede realizar programando la propiedad `AutoGenerateColumns` en `false` y creando las columnas a verse mediante código o mediante el diálogo de campos del control, para lo cual se selecciona la etiqueta inteligente (SmartTag) asociada al control y se elige "Edit Columns", luego se deshabilita la casilla "Auto-generate fields" y se van agregando los "Boundfields" o columnas enlazadas a mostrarse.

En cada columna se configuran sus propiedades `HeaderText` que indica el texto de la cabecera, `DataField` que indica el campo (si es una tabla) o propiedad (si es un objeto) que se va a mostrar del origen de datos, además de otras propiedades de formato.

A continuación un ejemplo que muestra como personalizar las columnas de los productos para ver solo el código, nombre, precio unitario y stock, alineando a la derecha los números y mostrando en formato moneda con 2 decimales el precio.

Demo 78

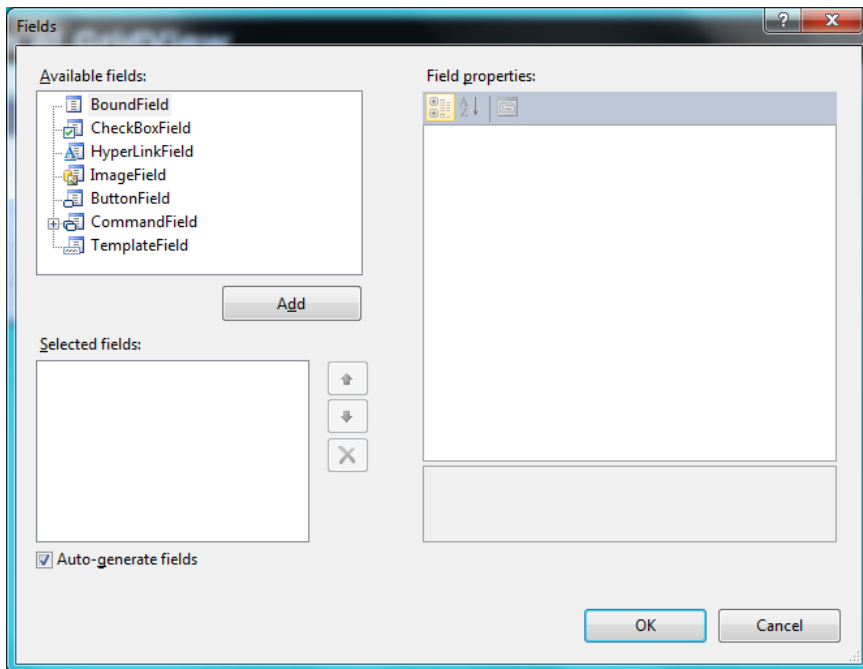
- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo78`.
- Hacer una referencia a la librería `Northwind.LibBusinessRules`.
- Agregar el archivo de hoja de estilo creado en el demo 75: del menú "Website", seleccionar "Add Existing Item".

- Ingresar al Demo 75 y abrir el archivo "Estilo.css".
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", "Web Form" y escribir como nombre "ListaProductos.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 3 filas por 1 columna.
- Diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Class	Personalizar Columnas en el GridView Titulo
Literal2 fila 2, celda1	Text Class	Lista de Productos Subtitulo
Button2 fila 3, celda 1	ID Auto Format	gvProducto Oceanica

- Clic a la etiqueta inteligente ubicada al lado izquierdo superior de la grilla y del menú contextual seleccionar "Edit Columns".
- Se mostrará un diálogo similar al de la figura 5.28:

Gráfico 5.28: Diálogo de configurar Campos del GridView



- Desmarcar la casilla de verificación "Auto-generate fields" de la parte derecha inferior para no generar columnas en ejecución.
- Agregar 4 columnas de tipo "BoundField" y cambiarlas como sigue:

Columna	Propiedad	Valor
BoundField1	HeaderText	Código
	DataField	Codigo
	ItemStyle	
	HorizontalAlign	Right
BoundField2	Width	70px
	HeaderText	Descripción
	DataField	Nombre
	ItemStyle	
BoundField3	Width	300px
	HeaderText	Precio
	DataField	PrecioUnitario
	DataFormatString	{0:c2}
	ItemStyle	
	HorizontalAlign	Right

	Width	100px
BoundField4	HeaderText DataField ItemStyle HorizontalAlign Width	Stock Stock Right 70px

- El diseño de la página Lista de Productos debe quedar similar al gráfico 5.29:

Gráfico 5.29: Diseño de la Página Lista de Productos

Personalizar Columnas en el GridView				
Lista de Productos				
Codigo	Descripcion	Precio	Stock	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	

- Escribir el siguiente código en la página:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Partial Class _Default
    Inherits System.Web.UI.Page
```

```
    Protected Sub Page_Load(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.Load
        If Not Page.IsPostBack Then
            Dim obrProducto As New brProducto
            Dim lobeProducto As List(Of beProducto) = obrProducto.Listar
            gvProducto.DataSource = lobeProducto
            gvProducto.DataBind()
        End If
    End Sub
```

```
End Sub
End Class
```

- Modificar el archivo de configuración web.config para agregar la clave conNW en la sección appSettings, tal como se muestra:

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" strict="false" explicit="true"
    targetFramework="4.0"/>
  </system.web>
  <appSettings>
    <add key="conNW" value="uid=CursoNET; pwd=123456;
    data source=DSOFT\DSOFT;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.30: Ejecución de la Página Lista de Productos



Personalizar Columnas en el GridView

Lista de Productos

Codigo	Descripcion	Precio	Stock
1	Chai	S/. 10.00	81
2	Chang	S/. 3.00	24
3	Chicha	S/. 10.00	39
4	Chef Anton's Cajun Seasoning	S/. 22.00	80
5	Chef Anton's Gumbo Mix	S/. 21.35	31
6	Grandma's Boysenberry Spread	S/. 25.00	50
7	Uncle Bob's Organic Dried Pears	S/. 30.00	98
8	Northwoods Cranberry Sauce	S/. 40.00	99
9	Mishi Kobe Niku	S/. 97.00	96
10	Ikura	S/. 31.00	93

3.2. Paginando en el GridView

Cuando los registros son demasiados es mejor no presentarlos todos, para esto se puede paginar o mostrar solo unos cuantos. En una aplicación web en ASP .NET podemos tener 3 tipos de paginaciones:

- **Paginación en el Servidor de Datos:** Si se trabaja con procedimientos almacenados consistiría en devolver solo una cierta página o conjunto de registros, la consulta estaría en línea pero lo malo sería que estaríamos conectándonos constantemente al servidor para obtener los datos.
- **Paginación en el Servidor Web:** Consiste en obtener los datos una sola vez del Servidor de Datos, desconectarse, guardarlos en memoria y enviar al cliente solo la página deseada. Esto evitaría conexiones constantes al servidor de datos aligerando su rendimiento pero lo malo es que no estamos en línea sino trabajando desconectados.
- **Paginación en el Cliente Web:** Consiste en obtener los datos del servidor de datos, recogerlos en el servidor web y enviarlos al cliente via Response y manejarlos en el cliente con Javascript, AJAX, JQuery, etc. Esto evitaría viajes tanto al servidor web como el servidor de datos pero aumentaría considerablemente el tamaño de la página en el cliente y lo podría volver lento.

De las 3 alternativas presentadas la mejor por escalabilidad y rendimiento (según mi opinión) es la segunda, es decir paginar en el Servidor Web y evitar viajes al servidor de datos y no sobre cargar al cliente. Esta forma de paginación la podemos implementar usando el GridView.

Para paginar en el GridView configurar la propiedad AllowPaging en True, definir el tamaño de la página configurando la propiedad PageSize y luego programar en el evento PageIndexChanging configurando el índice de página (propiedad PageIndex) en el índice seleccionado por el usuario (e.NewPageIndex) y finalmente enlazar nuevamente el control GridView.

A continuación un ejemplo basado en el demo anterior que lista los productos pero esta vez paginado de 10 en 10.

Demo 79

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo79`.
- Hacer una referencia a la librería `Northwind.LibBusinessRules`.
- Agregar el archivo de hoja de estilo y la página Lista de Productos creado en el demo anterior: del menú "Website", seleccionar "Add Existing Item".
- Ingresar al Demo 78 y seleccionar los archivos "Estilo.css" y "ListaProductos.aspx".
- Modificar el título de la página (literal 1) a "Paginando en el GridView".
- Seleccionar el GridView "gvProducto" y configurar la propiedad "AllowPaging" en "True".
- El diseño de la página Lista de Productos Paginada debe quedar similar al gráfico 5.31:

Gráfico 5.31: Diseño de la Página Lista de Productos Paginada

Paginando en el GridView

Lista de Productos

Codigo	Descripcion	Precio	Stock
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound

1 2

- Escribir el siguiente código en la página:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Partial Class _Default
```

```
    Inherits System.Web.UI.Page
```

```
    Private lobeProducto As New List(Of beProducto)
```

```
    Private Sub EnlazarGridView()
```

```
        gvProducto.DataSource = lobeProducto
```

```
        gvProducto.DataBind()
```

```
    End Sub
```

```
    Protected Sub Page_Load(ByVal sender As Object, _
```

```
        ByVal e As System.EventArgs) Handles Me.Load
```

```
        If Not Page.IsPostBack Then
```

```
            Dim obrProducto As New brProducto
```

```
            lobeProducto = obrProducto.Listar
```

```
            Session("Productos") = lobeProducto
```

```
            EnlazarGridView()
```

```
        End If
```

```
    End Sub
```

```
Protected Sub PaginarListaProductos(ByVal sender As Object, _  
ByVal e As System.Web.UI.WebControls.GridViewPageEventArgs) _  
Handles gvProducto.PageIndexChanging  
    lobeProducto = Session("Productos")  
    gvProducto.PageIndex = e.NewPageIndex  
    EnlazarGridView()  
End Sub  
End Class
```

Importante: En el código anterior se usa un objeto de sesión para guardar la lista de productos ya que al enviar los datos al cliente se pierden los valores de los objetos, es por eso que en las aplicaciones web se usan variables de sesión para guardar dichos datos en el servidor web y no perderlos.

- Modificar el archivo de configuración web.config para agregar la clave conNW en la sección appSettings, tal como se muestra:

```
<?xml version="1.0"?>  
<configuration>  
    <system.web>  
        <compilation debug="true" strict="false" explicit="true"  
targetFramework="4.0"/>  
    </system.web>  
    <appSettings>  
        <add key="conNW" value="uid=CursoNET; pwd=123456;  
data source=DSOFT\DSOFT;initial catalog=Northwind"/>  
    </appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.32: Ejecución de la Página Lista de Productos Paginada

Paginando en el GridView

Lista de Productos

Codigo	Descripcion	Precio	Stock
1	Chai	S/. 10.00	81
2	Chang	S/. 3.00	24
3	Chicha	S/. 10.00	39
4	Chef Anton's Cajun Seasoning	S/. 22.00	80
5	Chef Anton's Gumbo Mix	S/. 21.35	31
6	Grandma's Boysenberry Spread	S/. 25.00	50
7	Uncle Bob's Organic Dried Pears	S/. 30.00	98
8	Northwoods Cranberry Sauce	S/. 40.00	99
9	Mishi Kobe Niku	S/. 97.00	96
10	Ikura	S/. 31.00	93

1 2 3 4 5 6 7 8 9 10 ...

3.3. Ordenando en el GridView

Para acceder rápidamente a los datos es necesario a veces incorporar la funcionalidad de ordenar las columnas ya sea en forma ascendente o descendente, con esto el usuario puede ubicar rápidamente los datos de acuerdo a un criterio.

Para ordenar el GridView de ASP .NET se configura la propiedad Allow Sorting en True y luego se debe configurar la propiedad SortExpression de cada columna especificando el campo que se va a ordenar, también debemos programar en el evento Sorting la ordenación del origen de datos de acuerdo al campo o columna especificada en la propiedad SortExpression.

A continuación un ejemplo basado en el demo anterior de la paginación de productos, a la cual le vamos a aumentar para que se pueda ordenar en forma ascendente y descendente por cada columna de los productos.

Demo 80

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como pnombre: `http://Localhost/Demos/Demo80`.
- Hacer una referencia a la librería Northwind.LibBusinessRules.
- Agregar el archivo de hoja de estilo y la página Lista de Productos creado en el demo anterior: del menú "Website", seleccionar "Add Existing Item".
- Ingresar al Demo 79 y seleccionar los archivos "Estilo.css" y "ListaProductos.aspx".
- Modificar el título de la página (literal 1) a "Ordenando en el GridView".

- Seleccionar el GridView "gvProducto" y configurar la propiedad "AllowSorting" en "True".
- Clic a la etiqueta inteligente ubicada al lado izquierdo superior de la grilla y del menú contextual seleccionar "Edit Columns".
- Seleccionar cada columna y escribir en la propiedad SortExpression lo mismo que tiene la propiedad DataField, es decir el campo a ordenar.
- El diseño de la página Lista de Productos Ordenada debe quedar similar al gráfico 5.33:

Gráfico 5.33: Diseño de la Página Lista de Productos Ordenada

Ordenando en el GridView				
Lista de Productos				
Codigo	Descripcion	Precio	Stock	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
1 2				

- Agregar una clase al App_Code: del menú "Website", seleccionar "Add New Item", luego "Class", escribir como nombre "ucCompara.vb".
- Preguntará si se desea incluir la clase en la carpeta especial App_Code confirmar que Sí.
- Modificar el código de la clase como sigue:

```
Imports Microsoft.VisualBasic
Imports System.Collections.Generic
```

```
Public Class ucCompara(Of T)
```

```
Implements IComparer(Of T)
Private Campo As String
Private TipoOrden As TiposOrden

Public Enum TiposOrden
    Ascendente = 0
    Descendente = 1
End Enum

Public Sub New(ByVal vCampo As String, _
ByVal vTipoOrden As TiposOrden)
    Campo = vCampo
    TipoOrden = vTipoOrden
End Sub

Public Function Compare(ByVal x As T, ByVal y As T) As _
Integer Implements IComparer(Of T).Compare
    Dim valX As Object = _
x.GetType.GetProperty(Campo).GetValue(x, Nothing)
    Dim valY As Object = _
y.GetType.GetProperty(Campo).GetValue(y, Nothing)
    If TipoOrden = TiposOrden.Ascendente Then
        Return (valX.CompareTo(valY))
    Else
        Return (valY.CompareTo(valX))
    End If
End Function
End Class
```

Importante: En el código anterior se está creando una clase que permite ordenar cualquier tipo de objeto en forma ascendente o descendente usando para ello reflection. No solo se podrá usar para ordenar los productos de este ejemplo sino cualquier lista de objetos, por lo que es conveniente que se agregue la clase a una librería para que pueda ser reusable.

- Escribir el siguiente código en la página:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

Partial Class _Default

Inherits System.Web.UI.Page

Private lobeProducto As New List(Of beProducto)

Private Sub EnlazarGridView()

gvProducto.DataSource = lobeProducto

gvProducto.DataBind()

End Sub

Protected Sub Page_Load(ByVal sender As Object, _

ByVal e As System.EventArgs) Handles Me.Load

If Not Page.IsPostBack Then

Dim obrProducto As New brProducto

lobeProducto = obrProducto.Listar

Session("Productos") = lobeProducto

EnlazarGridView()

End If

End Sub

Protected Sub PaginarListaProductos(ByVal sender As Object, _

ByVal e As System.Web.UI.WebControls.GridViewPageEventArgs) _

Handles gvProducto.PageIndexChanging

lobeProducto = Session("Productos")

gvProducto.PageIndex = e.NewPageIndex

EnlazarGridView()

End Sub

Protected Sub OrdenarListaProductos(ByVal sender As Object, _

ByVal e As System.Web.UI.WebControls.GridViewSortEventArgs) _

Handles gvProducto.Sorting

lobeProducto = Session("Productos")

Dim N As Integer = 0

If ViewState(e.SortExpression) Is Nothing Then

N = 0

Else

If ViewState(e.SortExpression) = "1" Then

N = 0

Else

N = 1

End If

```
End If
ViewState(e.SortExpression) = N
Dim oCompara As New ucCompara(Of beProducto) _
(e.SortExpression, N)
lobeProducto.Sort(oCompara)
gvProducto.DataSource = lobeProducto
gvProducto.DataBind()
End Sub

Protected Sub CambiarSimboloOrdenacion(ByVal sender As Object, _
ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs) _
Handles gvProducto.RowCreated
    Dim gv As GridView = sender
    If e.Row.RowType = DataControlRowType.Header Then
        Dim Orden As String = "0"
        Dim celda As New TableCell
        Dim Dibujo As String = "▲"
        Dim Campo As String = ""
        For Each celda In e.Row.Cells
            Campo = CType(celda.Controls(0), LinkButton).Text
            If Campo = "Descripcion" Then
                Orden = ViewState("Nombre")
            ElseIf Campo = "Precio" Then
                Orden = ViewState("PrecioUnitario")
            Else
                Orden = ViewState(Campo)
            End If
            If Orden = "0" Then
                Dibujo = "▲"
            Else
                Dibujo = "▼"
            End If
            Dim lc As New LiteralControl(Dibujo)
            celda.Controls.Add(lc)
        Next
    End If
End Sub
End Class
```

Nota: En el código anterior se usa el evento RowCreated para crear dinámicamente los símbolos de ordenación que indiquen si la columna esta ordenada en forma ascendente o descendente. Para ello se toma el nombre de la cabecera como campo pero para el caso del nombre la cabecera dice Descripcion y para el caso del PrecioUnitario la cabecera dice solo Precio, es por eso que se hace una validación y conversión al nombre del campo.

- Modificar el archivo de configuración web.config para agregar la clave conNW en la sección appSettings, tal como se muestra:

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" strict="false" explicit="true"
      targetFramework="4.0"/>
  </system.web>
  <appSettings>
    <add key="conNW" value="uid=CursoNET; pwd=123456;
      data source=DSOFT\DSOFT;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.34: Ejecución de la Página Lista de Productos Ordenada

Ordenando en el GridView

Lista de Productos

Codigo ▼	Descripcion ▼	Precio ▼	Stock ▼
1	Chai	S/. 10.00	81
2	Chang	S/. 3.00	24
3	Chicha	S/. 10.00	39
4	Chef Anton's Cajun Seasoning	S/. 22.00	80
5	Chef Anton's Gumbo Mix	S/. 21.35	31
6	Grandma's Boysenberry Spread	S/. 25.00	50
7	Uncle Bob's Organic Dried Pears	S/. 30.00	98
8	Northwoods Cranberry Sauce	S/. 40.00	99
9	Mishi Kobe Niku	S/. 97.00	96
10	Ikura	S/. 31.00	93

1 2 3 4 5 6 7 8 9 10 ...

3.4. Mantenimiento de Datos en el GridView

Es esta última sección sobre el GridView veremos como realizar un mantenimiento de registros sobre el propio control GridView, para lo cual crearemos plantillas a nivel de columnas para agregar los botones de editar, actualizar, cancelar y eliminar.

Mediante la propiedad `EditIndex` podemos especificar que se va a editar una fila de la grilla y para regresar al modo lectura solo se configura la propiedad en -1.

Cuando enlazamos el control GridView a un origen de datos como por ejemplo una lista de objetos, desde el HTML de la grilla para referirnos a un campo o propiedad podemos realizarlo de 2 formas: `DataBinder.Eval` o también `Container.DataItem`.

Es preferible usar la segunda técnica en el enlace de datos, si el origen de datos fuese una tabla de un `DataSet` para recuperar una columna se usaría lo siguiente: `Container.DataItem("Campo")`, en cambio, si el origen de datos es una lista de objetos, para recuperar una propiedad se usaría lo siguiente: `Container.DataItem.Propiedad`.

Para saber la ubicación o el índice de la fila en la cual nos encontramos sobre la grilla se puede usar lo siguiente `Container.DataItemIndex`, todas estas instrucciones son de lado del servidor y necesitan escribirse como tal usando: `<%#Instrucción%>` desde el HTML de la grilla.

A continuación un ejemplo que ilustra como crear un GridView con columnas personalizadas de tipo plantillas para crear los botones de Editar, Actualizar, Cancelar y Eliminar, además de como programarlos de lado del servidor y hacer validaciones de lado del cliente como por ejemplo confirmar la eliminación. En este ejemplo usted aprenderá como adicionar, actualizar y eliminar productos usando el control web GridView. Además la lista de productos se encuentra paginada para una mejor visualización.

Demo 81

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo81`.
- Hacer una referencia a la librería `Northwind.LibBusinessRules`.
- Agregar el archivo de hoja de estilo creado en el demo anterior: del menú "Website", seleccionar "Add Existing Item".
- Ingresar al Demo 80 y seleccionar el archivo "Estilo.css".
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", "Web Form" y escribir como nombre "Mantenimiento Empleados.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 4 filas por 1 columna.
- Diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Class	Mantenimiento en el GridView Titulo
Literal2 fila 2, celda1	Text Class	Lista de Empleados Subtitulo
ImageButton1 fila 3, celda 1	ID Height ImageUrl Tooltip Width	ibNuevo 20px Imagenes/Nuevo.png Nuevo 20px
GridView1 fila 4, celda 1	ID Auto Format AllowPaging	gvEmpleado Oceanica True

- Clic a la etiqueta inteligente ubicada al lado izquierdo superior de la grilla y del menú contextual seleccionar "Edit Columns".
- Desmarcar la casilla de verificación "Auto-generate fields" de la parte derecha inferior para no generar columnas en ejecución.
- Agregar las siguientes columnas:

Columna	Propiedad	Valor
TemplateField1	HeaderText	Actualizar
TemplateField2	HeaderText	Eliminar
BoundField1	HeaderText DataField ReadOnly ItemStyle Width	Código Codigo True 60px
BoundField2	HeaderText DataField ItemStyle Width	Apellido Apellido 100px
BoundField3	HeaderText DataField ItemStyle Width	Nombre Nombre 100px
TemplateField3	HeaderText	Fecha Nac

- Ingresar a la vista "Source" y completar las 3 columnas de tipo plantilla:

```

<asp:TemplateField HeaderText="Actualizar">
  <ItemTemplate>
    <asp:HiddenField ID="hdfCorrelativo1"
      Value="< %#Container.DataItemIndex%>" runat="server" />
    <asp:ImageButton ID="ibEditar" OnClick="EditarEmpleado"
      ImageUrl="Imágenes/Editar.png" ToolTip="Editar" width="20px"
      height="20px" runat="server"/>
  </ItemTemplate>
</EditItemTemplate>
  <asp:HiddenField ID="hdfCorrelativo2"
    Value="< %#Container.DataItemIndex%>" runat="server" />
  <asp:ImageButton ID="ibGuardar" OnClick="GuardarEmpleado"
    ImageUrl="Imágenes/Guardar.png" ToolTip="Guardar" width="20px"
  >

```

```
        height="20px" runat="server"/>
        <asp:ImageButton ID="ibCancelar" OnClick="CancelarEmpleado"
            ImageUrl="Imágenes/Cancelar.png" ToolTip="Cancelar" width="20px"
            height="20px" runat="server"/>
    </EditItemTemplate>
</asp:TemplateField>

<asp:TemplateField HeaderText="Eliminar">
    <ItemTemplate>
        <asp:ImageButton ID="imgEliminar"
            ImageUrl="~/Imágenes/Eliminar.png" ToolTip="Eliminar" Width="20px"
            Height="20px" style="cursor:hand" runat="server"
            OnClientClick="return(confirm('Seguro de eliminar'));"
            OnClick="EliminarEmpleado" />
    </ItemTemplate>
</asp:TemplateField>

<asp:TemplateField HeaderText="Fecha Nac">
    <ItemTemplate>
        <asp:Label ID="lblFechaNac"
            Text='<%#ucRutinas.DevolverFecha(Container.DataItem.FechaNac)%>'
            runat="server" />
    </ItemTemplate>
    <EditItemTemplate>
        <asp:TextBox ID="txtFechaNac"
            Text='<%#ucRutinas.DevolverFecha(Container.DataItem.FechaNac)%>'
            runat="server" />
    </EditItemTemplate>
    <ItemStyle Width="100px" />
</asp:TemplateField>
```

- El diseño de la página debe quedar similar al gráfico 5.35:

Gráfico 5.35: Diseño de la Página Mantenimiento de Productos

Mantenimiento en el GridView						
Lista de Empleados						
	Actualizar	Eliminar	Codigo	Apellido	Nombre	Fecha Nac
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
			Databound	Databound	Databound	Databound
1 2						

- Agregar una clase al App_Code: del menú "Website", seleccionar "Add New Item", luego "Class", escribir como nombre "ucRutinas.vb".
- Preguntará si se desea incluir la clase en la carpeta especial App_Code confirmar que Sí.
- Modificar el código de la clase como sigue:

```
Imports Microsoft.VisualBasic
```

```
Public Class ucRutinas
```

```
    Public Shared Function DevolverFecha _
```

```
        (ByVal Fecha As DateTime) As String
```

```
        If Fecha.Year <= 1900 Then
```

```
            Return ("")
```

```
        Else
```

```
            Return (String.Format("{0:d}", Fecha))
```

```
        End If
```

```
    End Function
```

```
    Public Shared Function AsignarFecha _
```

```
        (ByVal Fecha As String) As DateTime
```

```
        If Fecha = "" Then
```

```
        Return (New Date(1900, 1, 1))
    Else
        Return (DateTime.Parse(Fecha))
    End If
End Function
End Class
```

Nota: Estas 2 funciones son usadas una por la grilla para devolver una fecha como cadena y la segunda para asignar una fecha, ambas manejan las fechas vacias como el 1 de Enero de 1900.

- Escribir el siguiente código en la página:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules

Partial Class _MantenimientoEmpleados
    Inherits System.Web.UI.Page

    Private lobeEmpleado As New List(Of beEmpleado)
    Private obrEmpleado As New brEmpleado

    Private Sub EnlazarGridView()
        gvEmpleado.DataSource = lobeEmpleado
        gvEmpleado.DataBind()
    End Sub

    Private Sub ListarEmpleados()
        lobeEmpleado = obrEmpleado.Listar
        Session("Empleados") = lobeEmpleado
        EnlazarGridView()
    End Sub

    Protected Sub Page_Load(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.Load
        If Not Page.IsPostBack Then
            ListarEmpleados()
        End If
    End Sub

    Protected Sub PaginarEmpleados _
```

```
(ByVal sender As Object, ByVal e As GridViewPageEventArgs) _  
Handles gvEmpleado.PageIndexChanging  
    lobeEmpleado = Session("Empleados")  
    gvEmpleado.PageIndex = e.NewPageIndex  
    EnlazarGridView()  
End Sub
```

```
Protected Sub EliminarEmpleado(ByVal sender As Object, _  
ByVal e As EventArgs)  
    Dim img As ImageButton = sender  
    Dim celda As DataControlFieldCell = img.Parent  
    Dim fila As GridViewRow = celda.Parent  
  
    Dim Codigo As Integer = Integer.Parse(fila.Cells(2).Text)  
    Dim N As Integer = obrEmpleado.Eliminar(Codigo)  
    If N > 0 Then  
        ClientScript.RegisterStartupScript(Page.GetType, "Mensaje", _  
            "<script language='javascript'>alert('Empleado eliminado');</script>")  
    Else  
        ClientScript.RegisterStartupScript(Page.GetType, "Mensaje", _  
            "<script language='javascript'>alert('Empleado No se pudo  
            eliminar');</script>")  
    End If  
    ListarEmpleados()  
End Sub
```

```
Protected Sub NuevoEmpleado(ByVal sender As Object, _  
ByVal e As ImageClickEventArgs) Handles ibNuevo.Click  
    ViewState("operacion") = 1 'Adicionar  
    lobeEmpleado = Session("Empleados")  
    Dim obeEmpleado As New beEmpleado  
    lobeEmpleado.Add(obeEmpleado)  
    Dim IndiceUltimaPagina As Integer = _  
    lobeEmpleado.Count \ gvEmpleado.PageSize  
    If lobeEmpleado.Count Mod gvEmpleado.PageSize = 0 Then _  
        IndiceUltimaPagina -= 1  
    gvEmpleado.PageIndex = IndiceUltimaPagina  
    gvEmpleado.EditIndex = _  
    (lobeEmpleado.Count Mod gvEmpleado.PageSize) - 1  
    EnlazarGridView()
```

End Sub

```
Private Function DevolverCorrelativo(ByVal sender As Object) As Integer
    Dim ib As ImageButton = CType(sender, ImageButton)
    Dim celda As DataControlFieldCell = _
        CType(ib.Parent, DataControlFieldCell)
    Dim hdf As HiddenField = _
        CType(celda.Controls(1), HiddenField)
    Return (hdf.Value)
End Function
```

```
Protected Sub EditarEmpleado(ByVal sender As Object, _
    ByVal e As EventArgs)
    ViewState("operacion") = 2 'Actualizar
    lobeEmpleado = Session("Empleados")
    Dim N As Integer = DevolverCorrelativo(sender)
    gvEmpleado.EditIndex = _
        N - gvEmpleado.PageIndex * gvEmpleado.PageSize
    EnlazarGridView()
End Sub
```

```
Protected Sub GuardarEmpleado(ByVal sender As Object, _
    ByVal e As EventArgs)
    lobeEmpleado = Session("Empleados")
    Dim N As Integer = DevolverCorrelativo(sender)
    Dim pos As Integer = _
        N - gvEmpleado.PageIndex * gvEmpleado.PageSize
    Dim obeEmpleado As New beEmpleado
    With obeEmpleado
        .Apellido = CType(gvEmpleado.Rows(pos).Cells(3).Controls(0), _
            TextBox).Text
        .Nombre = CType(gvEmpleado.Rows(pos).Cells(4).Controls(0), _
            TextBox).Text
        .FechaNac = ucRutinas.AsignarFecha(CType _
            (gvEmpleado.Rows(pos).Cells(5).Controls(1), TextBox).Text)
    End With
    If ViewState("operacion") = "1" Then
        'Adicionar un Empleado
        Dim IdEmpleado As Integer = obrEmpleado.Adicionar(obeEmpleado)
        If IdEmpleado > 0 Then
```



```
ClientScript.RegisterStartupScript(Type.GetType("System.String"),
    "Mensaje", _
    "<script lenguaje='javascript'>alert('Se registro el empleado " + _
    IdEmpleado.ToString + "');</script>")
Else
    ClientScript.RegisterStartupScript(Type.GetType("System.String"),
    "Mensaje", _
    "<script lenguaje='javascript'>alert('No se pudo registrar el
    empleado');</script>")
End If
ElseIf ViewState("operacion") = "2" Then
    'Actualizar un Empleado
    obeEmpleado.Codigo = gvEmpleado.Rows(pos).Cells(2).Text
    Dim exito As Boolean = obrEmpleado.Actualizar(obeEmpleado)
    If exito Then
        ClientScript.RegisterStartupScript(Type.GetType("System.String"),
            "Mensaje", _
            "<script lenguaje='javascript'>alert('Se actualizo el
            empleado');</script>")
    Else
        ClientScript.RegisterStartupScript(Type.GetType("System.String"),
            "Mensaje", _
            "<script lenguaje='javascript'>alert('No se pudo actualizar el
            empleado');</script>")
    End If
End If
gvEmpleado.EditIndex = -1
ViewState("operacion") = ""
ListarEmpleados()
End Sub

Protected Sub CancelarEmpleado(ByVal sender As Object, _
    ByVal e As EventArgs)
    lobeEmpleado = Session("Empleados")
    gvEmpleado.EditIndex = -1
    EnlazarGridView()
End Sub
End Class
```

Nota: En el código anterior se usa un objeto ViewState para guardar el valor de la operación elegida, 1 es para dicionar y 2 para actualizar, este

valor se usa porque existe un solo botón para guardar cualquier operación. El objeto ViewState guarda dicho valor en el cliente y lo recupera en el servidor.

- Modificar el archivo de configuración web.config para agregar la clave conNW en la sección appSettings, tal como se muestra:

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" strict="false" explicit="true"
    targetFramework="4.0"/>
  </system.web>
  <appSettings>
    <add key="conNW" value="uid=CursoNET; pwd=123456;
    data source=DSOFT\DSOFT;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.36: Ejecución de la Página Mantenimiento de Productos

4. Creando Plantillas en Controles Enlazados a Datos

Las plantillas (templates) son un conjunto de elementos, controles HTML y código de servidor incrustado que se usan para personalizar la apariencia de un control web enlazado a datos.

Una plantilla permite personalizar el diseño de un control web y especificar que controles se mostrarán en modo lectura usando `ItemTemplate` y que se mostrará en modo edición usando `EditItemTemplate`. También podemos definir que irá en la cabecera del control usando `HeaderTemplate` y que irá en el pie usando `FooterTemplate`.

Los controles web que soportan plantillas son los enlazados a datos: tales como el `Repeater`, `DataList`, `GridView`, `DetailsView`, `FormView`; así como también los controles de navegación: `Menu` y `SiteMapPath`, entre otros.

Para darle dinamismo a las plantillas se puede usar código incrustado del servidor mediante el elemento: `<% %>`, por ejemplo podemos usar enlace de datos mediante `<%#Container.DataItem.Propiedad%>` para obtener el valor de una propiedad de una fila o también usar una función definida en el código como protegida: `<%#Función(Par1,Par2,...)%>`.

En realidad las plantillas permiten crear aplicaciones de datos con poco código y muy personalizado. Es más fácil usar plantillas con código incrustado del servidor que programar en el evento `RowCreated` o `RowDataBound` de un `GridView` para crear subtotales, ocultar elementos, asociar scripts de lado del cliente, etc.

En esta parte veremos cómo usar plantillas en el control `Repeater`, luego en el control `DataList` y finalmente como crear plantillas jerárquicas para mostrar un control dentro de otro. Algunos usos de las plantillas jerárquicas son ver los detalles de cada orden en una misma grilla, los productos de una cierta categoría, los distritos de cada provincia de cada departamento, etc.

Para obtener más información sobre plantillas de controles de servidor web ver la referencia 35 al final del libro.

4.1. Trabajando con el Control Repeater

El control Repeater es el más ligero de todos los controles enlazados a datos, tiene pocas propiedades y cuando se enlaza a un origen de datos no se muestra ningún dato, ya que necesita que se cree su plantilla para indicar que es lo que se va a mostrar.

Este control soporta solo 5 tipos de plantillas: HeaderTemplate, FooterTemplate, ItemTemplate, AlternatingItemTemplate y Separator Template. El principal elemento de una plantilla (que nunca debe faltar) es ItemTemplate que indica el diseño de cada fila a repetirse.

Cuando se especifica el elemento AlternatingItemTemplate las filas pares salen con este diseño y las impares con lo que se especifique en ItemTemplate.

Finalmente, para especificar el índice de la fila actual del control Repeater podemos usar el siguiente código: `<%#Container.ItemIndex%>`.

A continuación se verá un ejemplo de como crear una plantilla para el control Repeater que muestre en una tabla los datos de las categorías: código, nombre y foto la cual es extraída de una carpeta que tiene archivos jpg con el código como nombre de archivo, es decir 1.jpg, 2.jpg, etc y para las categorías nuevas que no tengan foto se mostrará el archivo No.jpg.

Demo 82

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo82`.
- Hacer una referencia a la librería Northwind.LibBusinessRules.
- Agregar el archivo de hoja de estilo creado en el demo anterior: del menú "Website", seleccionar "Add Existing Item".

- Ingresar al Demo 81 y seleccionar el archivo "Estilo.css".
- Crear una carpeta llamada "Categorias" y agregar 9 archivos de imagen: del 1.jpg al 8.jpg y el archivo No.jpg
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", "Web Form" y escribir como nombre "ListaCategorias.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 3 filas por 1 columna.
- Diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Class	Plantilla en el control repeater Titulo
Literal2 fila 2, celda1	Text Class	Lista de Categorías Subtitulo
Repeater1 fila 3, celda 1	ID	rpCategoria

- Ingresar a la vista "Source" y modificar el control Repeater como sigue:

```
<asp:Repeater ID="rpCategoria" runat="server">
  <HeaderTemplate>
    <table>
      <tr bgcolor="blue">
        <td style="width:70px">
          <font color="white">Codigo</font>
        </td>
        <td style="width:200px">
          <font color="white">Nombre</font>
        </td>
        <td style="width:100px">
          <font color="white">Foto</font>
        </td>
      </tr>
    </HeaderTemplate>
    <ItemTemplate>
      <tr bgcolor="white">
        <td style="width:70px">
```

```
<font color="blue">
  <%# Container.DataItem.Codigo%>
</font>
</td>
<td style="width:200px">
  <font color="blue">
    <%# Container.DataItem.Nombre%>
  </font>
</td>
<td style="width:100px">
  <font color="blue">
    <img src='<%#VerImagen(Container.DataItem.Codigo)%>'
      width="100" height="80" alt="" />
  </font>
</td>
</tr>
</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
```

- El diseño de la página debe quedar similar al gráfico 5.37:

Gráfico 5.37: Diseño de la Página Lista de Categorías**Plantilla en el Control Repeater**

Lista de Categorías

Codigo	Nombre	Foto
Databound	Databound	
Databound	Databound	
Databound	Databound	
Databound	Databound	
Databound	Databound	

- Escribir el siguiente código en la página:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.IO
```

```
Partial Class _ListaCategorias
    Inherits System.Web.UI.Page
```

```
Protected Sub Page_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
    If Not Page.IsPostBack Then
        Dim obrCategoria As New brCategoria
        Dim lobeCategoria As List(Of beCategoria) = _
            obrCategoria.Listar()
        rpCategoria.DataSource = lobeCategoria
        rpCategoria.DataBind()
```



```
End If
End Sub
```

```
Protected Function VerImagen(ByVal Codigo As Integer) As String
    Dim Archivo As String = Server.MapPath( _
        String.Format("Categorias/{0}.jpg", Codigo))
    Dim URL As String = String.Format _
        ("http://localhost/Demos/Demo82/Categorias/{0}.jpg", Codigo)
    If Not File.Exists(Archivo) Then
        URL = "http://localhost/Demos/Demo82Categorias/No.jpg"
    End If
    Return (URL)
End Function
End Class
```

Nota: En el código anterior la función VerImagen esta enlazada a la propiedad src del control HTML image para devolver el archivo correspondiente al código de categoría, sino existe se devuelve el archivo No.jpg.

- Modificar el archivo de configuración web.config para agregar la clave conNW en la sección appSettings, tal como se muestra:

```
<?xml version="1.0"?>
<configuration>
    <system.web>
        <compilation debug="true" strict="false" explicit="true"
            targetFramework="4.0"/>
    </system.web>
    <appSettings>
        <add key="conNW" value="uid=CursoNET; pwd=123456;
            data source=DSOFT\DSOFT;initial catalog=Northwind"/>
    </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.38: Ejecución de la Página Lista de Categorías



4.2. Trabajando con el Control DataList

Trabajar con el Control DataList es muy similar al control Repeater anteriormente visto, la diferencias básicas son que el DataList tiene mas propiedades que permiten controlar la presentación de la plantilla tales como la propiedad RepeatColumns que indica la cantidad de columnas a mostrar por defecto se muestra en una, la propiedad RepeatDirection que indica si se desea ver en forma horizontal o vertical y la propiedad RepeatLayout que indica el tipo de diseño en tabla, flujo, lista ordenada o sin ordenar.

Además mientras el control Repeater solo puede crearse la plantilla desde la vista "Source" en el HTML, el control DataList tiene soporte para crear la plantilla en modo "Design" dando clic a la etiqueta inteligente al lado izquierdo superior del control y seleccionando "Edit Templates".

También otra característica es que soporta todos los tipos de plantillas: las 5 soportadas por el control Repeater (HeaderTemplate, FooterTemplate, ItemTemplate, AlternatingItemTemplate y SeparatorTemplate), más EditItemTemplate y SelectedItemTemplate, más todas las plantillas de estilos: ItemStyle, EditItemStyle, SelectedItemStyle, AlternatingItemStyle, etc.

A continuación se verá un ejemplo de como crear una plantilla para el control DataList que muestre en una tabla los datos de las empleados: código, apellido, nombre, fecha de nacimiento y foto la cual es extraída de una carpeta que tiene archivos jpg con el código como nombre de archivo, es decir 1.jpg, 2.jpg, etc y para los empleados que no tengan foto se mostrará el archivo No.jpg.

Demo 83

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "Add", "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".

- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://Localhost/Demos/Demo83`.
- Hacer una referencia a la librería `Northwind.LibBusinessRules`.
- Agregar el archivo de hoja de estilo creado en el demo anterior: del menú "Website", seleccionar "Add Existing Item".
- Ingresar al Demo 82 y seleccionar el archivo "Estilo.css".
- Crear una carpeta llamada "Empleados" y agregar varios archivos de imagen: del 1.jpg al N.jpg y el archivo No.jpg
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", "Web Form" y escribir como nombre "ListaEmpleados.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 3 filas por 1 columna.
- Diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Class	Plantilla en el control DataList Titulo
Literal2 fila 2, celda1	Text Class	Lista de Empleados Subtitulo
DataList1 fila 3, celda 1	ID	dlsEmpleado

- Ingresar a la vista "Source" y modificar el control DataList como sigue:

```
<asp:DataList ID="dlsEmpleado" runat="server">
  <HeaderTemplate>
    <table width="600px">
      <tr bgcolor="green">
        <td><font color="yellow">Codigo</font></td>
        <td><font color="yellow">Apellido</font></td>
        <td><font color="yellow">Nombre</font></td>
        <td><font color="yellow">Fecha Nac</font></td>
        <td><font color="yellow">Foto</font></td>
      </tr>
    </table>
  </HeaderTemplate>
```

```

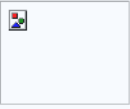
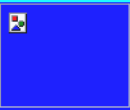
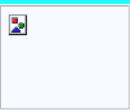
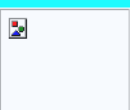
<ItemTemplate>
  <tr bgcolor="white">
    <td><font color="blue"><%#Eval("Codigo")%></font></td>
    <td><font color="blue"><%#Eval("Apellido")%></font></td>
    <td><font color="blue"><%#Eval("Nombre")%></font></td>
    <td><font color="blue"><%#String.Format("{0:d}",
      Eval("FechaNac"))%></font></td>
  <td>
    <font color="blue">
      <asp:Image ImageUrl='<%#ObtenerImagen
        (Eval("Codigo"))%>'
        width="100px" height="80px" runat="server" />
    </font>
  </td>
</tr>
</ItemTemplate>
<AlternatingItemTemplate>
  <tr bgcolor="blue">
    <td><font color="white"><%#Eval("Codigo")%></font></td>
    <td><font color="white"><%#Eval("Apellido")%></font></td>
    <td><font color="white"><%#Eval("Nombre")%></font></td>
    <td><font color="white"><%#String.Format("{0:d}",
      Eval("FechaNac"))%></font></td>
  <td>
    <font color="white">
      <asp:Image ImageUrl='<%#ObtenerImagen
        (Eval("Codigo"))%>'
        width="100px" height="80px" runat="server" />
    </font>
  </td>
</tr>
</AlternatingItemTemplate>
<FooterTemplate>
  </table>
</FooterTemplate>
</asp:DataList>

```

Nota: Recordar que al existir el elemento AlternatingItem las filas pares tomaran este estilo y las impares el del ItemTemplate.

- El diseño de la página debe quedar similar al gráfico 5.39:

Gráfico 5.39: Diseño de la Página Lista de Empleados

Plantilla en el control DataList				
Lista de Empleados				
Código	Apellido	Nombre	Fecha Nac	Foto
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	
Databound	Databound	Databound	Databound	

- Escribir el siguiente código en la página:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.IO
```

```
Partial Class _Default
    Inherits System.Web.UI.Page
```

```
Protected Sub Page_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
    'Si es la primera vez que carga la pagina
    If Not Page.IsPostBack Then
        Dim obrEmpleado As New brEmpleado
        Dim lobeEmpleado As New List(Of beEmpleado)
        lobeEmpleado = obrEmpleado.Listar
        dlsEmpleado.DataSource = lobeEmpleado
        dlsEmpleado.DataBind()
    End If
```

End Sub

```
Protected Function ObtenerImagen(ByVal codigo As Integer) As String
    Dim archivo As String = String.Format("{0}{1}.jpg", _
        Server.MapPath("Empleados/"), codigo)
    If File.Exists(archivo) Then
        Return (String.Format("~/Empleados/{0}.jpg", codigo))
    Else
        Return ("~/Empleados/No.jpg")
    End If
End Function
End Class
```

Nota: En el código anterior la función `ObtenerImagen` esta enlazada a la propiedad `ImageUrl` del control web `Image` para devolver el archivo correspondiente al código de empleado, sino existe se devuelve el archivo `No.jpg`.

Importante: Observar que en el ejemplo anterior, la función del servidor se enlazó a un control HTML y ahora a un control web, es decir podemos usar código incrustado del servidor en ambos tipos de controles.

- Modificar el archivo de configuración `web.config` para agregar la clave `conNW` en la sección `appSettings`, tal como se muestra:

```
<?xml version="1.0"?>
<configuration>
    <system.web>
        <compilation debug="true" strict="false" explicit="true"
            targetFramework="4.0"/>
    </system.web>
    <appSettings>
        <add key="conNW" value="uid=CursoNET; pwd=123456;
            data source=DSOFT\DSOFT;initial catalog=Northwind"/>
    </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla `F5`.

Gráfico 5.40: Ejecución de la Página Lista de Empleados



The screenshot shows a web browser window with the address bar displaying 'http://localhost/De' and 'localhost'. The page title is 'Plantilla en el control DataList'. Below the title, there is a section labeled 'Lista de Empleados' which contains a table with five columns: 'Codigo', 'Apellido', 'Nombre', 'Fecha Nac', and 'Foto'. The table lists five employees with their respective IDs, last names, first names, birth dates, and photos.

Codigo	Apellido	Nombre	Fecha Nac	Foto
1	Davolio	Nancy2	08/12/1948	
2	Fuller	Andrew	19/02/1952	
3	Leverling	Janet1	30/08/1963	
4	Peacock	Margaret	19/09/1937	
5	Buchanan	Steven	04/03/1955	

4.3. Creando Plantillas Jerárquicas

Como último tema sobre plantillas crearemos plantillas jerárquicas, es decir dentro de un control enlazado a datos incluiremos otro control enlazado a datos, similar a un TreeView con la diferencia de que el TreeView solo puede ver una columna, en cambio con las plantillas jerarquicas podemos ver la cantidad de columnas que deseemos.

A continuación, presentaremos un ejemplo de una lista de categorías que a su vez cada categoría presenta una lista de productos, para esto necesitamos crear una carpeta con las categorías y también un par de imágenes para los iconos de expandir y contraer para ver los productos.

Demo 84

- Crear un nuevo sitio web vacío: del menú "File", seleccionar "New Web Site".
- Seleccionar como lenguaje "Visual Basic" y como plantilla "ASP .NET Empty Web Site".
- Seleccionar como ubicación del sitio web "HTTP" y escribir como nombre: `http://localhost/Demos/Demo84`.
- Hacer una referencia a la librería Northwind.LibBusinessRules.
- Agregar el archivo de hoja de estilo creado en el demo anterior: del menú "Website", seleccionar "Add Existing Item".
- Ingresar al Demo 83 y seleccionar el archivo "Estilo.css".
- Crear una carpeta llamada "Categorias" y agregar varios archivos de imagen: del 1.jpg al N.jpg y el archivo No.jpg
- Agregar una nueva página: del menú "Website", seleccionar "Add New Item", "Web Form" y escribir como nombre "ProductosPor Categorias.aspx".
- Ubicarse dentro del div de la nueva página e insertar una tabla de 3 filas por 1 columna.

- Diseñar la página dentro de las celdas de la tabla como sigue:

Objeto	Propiedad	Valor
Literal1 fila 1, celda 1	Text Class	Plantillas Jerárquicas en DataList Titulo
Literal2 fila 2, celda1	Text Class	Lista de Categorías y sus Productos Subtitulo
DataList1 fila 3, celda 1	ID	dlsCategoria

- Ingresar a la vista "Source" y modificar el control DataList como sigue:

```
<asp:DataList ID="dlsCategoria" runat="server">
  <HeaderTemplate>
    <table width="600px">
      <tr bgcolor="blue">
        <td style="width:100px">
          <font color="white">Codigo</font>
        </td>
        <td style="width:300px">
          <font color="white">Nombre de Categoria</font>
        </td>
        <td style="width:200px">
          <font color="white">Foto Categoria</font>
        </td>
      </tr>
    </HeaderTemplate>
    <ItemTemplate>
      <tr bgcolor="white">
        <td style="width:100px">
          <font color="blue">
            )'
            alt="" style="cursor:hand" width="20px" height="20px" />
            < %# Container.DataItem.Codigo%>
          </font>
        </td>
        <td style="width:300px">
          <font color="blue">
```

```
        <%# Container.DataItem.Nombre%>
    </font>
</td>
<td style="width:200px">
    <font color="blue">
        <asp:Image
ImageUrl='<%#ObtenerFotoCategoria(Container.DataItem.Codigo)%>'
Width="100px" Height="70px" runat="server" />
    </font>
</td>
</tr>
<tr>
    <td style="width:100px"></td>
    <td style="width:500px" colspan="2">
        <asp:DataList ID="dlsProducto"
DataSource="<%#ListarProductosPorCategoria
(Container.DataItem.Codigo)%>"
runat="server" style="display:none">
        <HeaderTemplate>
            <table>
                <tr bgcolor="red">
                    <td style="width:100px">
                        <font color="white">Codigo</font>
                    </td>
                    <td style="width:200px">
                        <font color="white">Nombre del Producto</font>
                    </td>
                    <td style="width:100px">
                        <font color="white">Precio</font>
                    </td>
                    <td style="width:100px">
                        <font color="white">Stock</font>
                    </td>
                </tr>
            </HeaderTemplate>
            <ItemTemplate>
                <tr bgcolor="white">
                    <td style="width:100px">
                        <font color="red">
                            <%# Container.DataItem.Codigo%>
```

```
        </font>
    </td>
    <td style="width:200px">
        <font color="red">
            <%# Container.DataItem.Nombre%>
        </font>
    </td>
    <td style="width:100px">
        <font color="red">
            <%# Container.DataItem.PrecioUnitario%>
        </font>
    </td>
    <td style="width:100px">
        <font color="red">
            <%# Container.DataItem.Stock%>
        </font>
    </td>
</tr>
</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:DataList>
</td>
</tr>
</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:DataList>
```

Nota: Para llenar los productos de cada categoría se hace uso de una función del servidor que estará definida mas adelante.

- Aumentar el siguiente script en la cabecera de la página (sección Head):

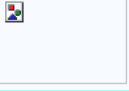
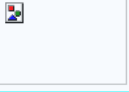
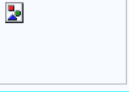
```
<script language="javascript" type="text/javascript">
    function Expandir(img, correlativo) {
        if (img.src == "http://localhost/Demos/Demo84/Contraer.png") {
            img.src = "http://localhost/Demos/Demo84/Expandir.png";
```

```
    }  
    else {  
        img.src = "http://localhost/Demos/Demo84/Contraer.png";  
    }  
    var tabla = document.getElementById("dlsCategoria_dlsProducto_" +  
        correlativo);  
    if (tabla.style.display == "" || tabla.style.display == "inline") {  
        tabla.style.display = "none";  
    }  
    else {  
        tabla.style.display = "inline";  
    }  
}  
</script>
```

Nota: La función anterior tiene como parámetros la imagen a la cual se dio clic y el índice de la fila actual, lo que permite expandir o colapsar la lista de productos que en realidad es una tabla con un nombre compuesto y al final tiene dicho índice, el cual se obtiene con: `Container.ItemIndex`.

- El diseño de la página debe quedar similar al gráfico 5.41:

Gráfico 5.41: Diseño de la Página Lista de Categorías y Productos

Plantillas Jerárquicas en DataList		
Lista de Categorías y sus Productos		
Codigo	Nombre de Categoría	Foto Categoría
 Databound	Databound	 
 Databound	Databound	 
 Databound	Databound	 
 Databound	Databound	 
 Databound	Databound	 

- Escribir el siguiente código en la página:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Imports System.IO
```

```
Partial Class _ProductosPorCategorias
```

```
    Inherits System.Web.UI.Page
```

```
    Private lobeProducto As New List(Of beProducto)
```

```
    Private CodigoCategoría As Integer
```

```
    Protected Sub Page_Load(ByVal sender As Object, _
```

```
        ByVal e As System.EventArgs) Handles Me.Load
```

```
        If Not Page.IsPostBack Then
```

```
            'Llenar las Categorías
```

```
            Dim lobeCategoría As New List(Of beCategoría)
```

```
            Dim obrCategoría As New brCategoría
```

```
            lobeCategoría = obrCategoría.Listar
```

```
            'Llenar los Productos
```

```
Dim obrProducto As New brProducto
lobeProducto = obrProducto.Listar
Session("Productos") = lobeProducto
ViewState("N") = 0
'Enlazar el control
dlsCategoria.DataSource = lobeCategoria
dlsCategoria.DataBind()
End If
End Sub
```

```
Protected Function ObtenerFotoCategoria _
(ByVal codigo As Integer) As String
Dim Archivo As String = String.Format("{0}{1}.jpg", _
Server.MapPath("Categorias/"), codigo)
If File.Exists(Archivo) Then
Return (String.Format("~/Categorias/{0}.jpg", codigo))
Else
Return ("~/Categorias/No.jpg")
End If
End Function
```

```
Private Function BuscarProductos _
(ByVal obeProducto As beProducto) As Boolean
Return (obeProducto.IdCategoria = CodigoCategoria)
End Function
```

```
Protected Function ListarProductosPorCategoria _
(ByVal codCategoria As Integer) As List(Of beProducto)
lobeProducto = Session("Productos")
CodigoCategoria = codCategoria
Dim pred As New Predicate(Of beProducto) _
(AddressOf BuscarProductos)
Return (lobeProducto.FindAll(pred))
End Function
End Class
```

Nota: En el código anterior se está usando predicados para filtrar los productos de una cierta categoría.

- Modificar el archivo de configuración web.config para agregar la clave conNW en la sección appSettings, tal como se muestra:

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" strict="false" explicit="true"
targetFramework="4.0"/>
  </system.web>
  <appSettings>
    <add key="conNW" value="uid=CursoNET; pwd=123456;
data source=DSOFT\DSOFT;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 5.42: Ejecución de la Página Lista de Categorías y Productos

Codigo	Nombre de Categoría	Foto Categoría	
1	Beverages		
Codigo	Nombre del Producto	Precio	Stock
1	Chai	10.0000	81
3	Chicha	10.0000	39
24	Guaraná Fantástica	4.5000	20
34	Sasquatch Ale	14.0000	111
35	Steeleye Stout	18.0000	20
38	Côte de Blaye	100.0000	17
39	Chartreuse verte	18.0000	69
43	Ipoh Coffee	46.0000	17
67	Laughing Lumberjack Lager	14.0000	1
70	Outback Lager	15.0000	14
75	Rhönbräu Klosterbier	7.7500	125

Preguntas de Repaso

1. En que espacio de nombres se encuentra ASP .NET?
2. Con qué nombre se le conoce al IDE de Visual Studio para desarrollar Sitios Web?
3. Menciona 3 Lenguajes con que podemos desarrollar aplicaciones web en ASP .NET
4. Qué características de ASP .NET hacen que las aplicaciones sean mas rápidas?
5. Por qué las páginas ASP .NET que usan controles web pueden verse en cualquier Browser?.
6. Qué es un formulario web en ASP .NET?
7. Cómo se clasifican los controles del servidor?
8. Por qué una pagina ASP .NET es lenta solo la primera vez que carga?
9. Cómo se llama el proceso que ejecuta los servicios del IIS?
10. En cuantas ubicaciones se puede crear un nuevo sitio web?
11. Qué ejecutable debe correr en caso de que no este registrado ASP .NET en IIS en vez de volver a instalar primero el IIS y luego Visual Studio?
12. Qué es un control web intrínseco?
13. Mencione 5 controles web intrínsecos.
14. Cuando se trabaja con archivos en una aplicación web, cuál es el directorio por defecto donde se leen o escriben los archivos en disco?

15. A qué usuario debemos dar permiso en la carpeta para que la aplicación web pueda leer o escribir un archivo en ASP .NET?
16. En que lado debe realizarse las validaciones de una página: en el cliente o el servidor y porqué?
17. Cómo se llama el algoritmo de seguridad usado para comprobar si el que usa el sistema, por ejemplo la pagina es un usuario o un programa.
18. Qué debe hacerse para que al generar varios números al azar con consecutivamente no devuelvan el mismo número?
19. Cuál es la forma mas simple de subir un archivo desde la pagina hacia el servidor?
20. Cuál es el tamaño máximo del archivo por defecto al usar el control FileUpload y que debe hacerse para aumentar la capacidad?
21. Cuál es la alternativa mas simple para validar sin usar directamente Javascript en el cliente en ASP NET?
22. Menciona 3 controles de validación de ASP .NET?
23. Con qué control web se valida que un campo sea obligatorio?
24. Qué control de validación usaría para validar una fecha?
25. Qué control de validación usaría para validar un RUC, DNI ó teléfono?
26. Menciona 3 propiedades comunes de todos los controles de validación.
27. Cuál es la propiedad mas importante del control de validación que indica qué control se debe validar?
28. Cómo se llama el control que muestra un resumen con los mensajes de error de todos los controles de validación?

29. Para qué se usan las Hojas de Estilo en Cascada?
30. Qué ventajas tiene usar archivos CSS en una página ASP .NET?
31. Para qué se usan las páginas principales y qué extensión tienen?
32. Qué ventajas tiene usar páginas principales en ASP .NET?
33. Menciona los 3 controles web de navegación.
34. Con qué control web podemos ver la ubicación donde nos encontramos en el sitio web?
35. Cómo se llama el archivo XML donde se definen las páginas que tendrá el sitio web?
36. Cómo se llama el objeto que permite enlazar un mapa de sitio en un Menú o en un TreeView?
- 37.Cuál es la forma más simple de implementar fichas en un sitio web de ASP .NET?
38. Con qué propiedad del control web MultiView se especifica que vista (View) se va a visualizar?
39. Que propiedad tienen los controles web que indica que se debe guardar el estado o los valores en el cliente para que no se pierdan al hacer Postback?
- 40.Cuál es el control web mas usado en una aplicación con ASP .NET?
41. Cómo se llama la propiedad que permite crear dinámicamente las columnas en un control GridView?
42. Qué pasos debe realizar para personalizar columnas en un GridView?
43. Cuantos tipos de paginación podemos tener en una aplicación web?

44. Qué propiedades debe configurarse para paginar un GridView?
45. En qué evento hay que programar la paginación del GridView?
46. Qué propiedad debe configurarse para ordenar las columnas en un GridView?
47. En qué evento hay que programar la ordenación en el GridView?
48. Qué evento del control GridView puede usarse para crear dinámicamente controles?
49. Qué instrucción de enlace de datos es mas eficiente: `DataSourceBinder.Eval` o `DataSourceContainer.DataItem`?
50. Con qué propiedad del GridView se especifica el registro que se va a editar?
- 51.Cuál es la diferencia entre guardar un dato usando el objeto `Session` y usando el objeto `ViewState`?
52. Qué se debe hacer para que una columna del GridView al pasar al modo edición no sea siempre un `TextBox`?
- 53.Cuál es la ventaja de usar plantillas en controles enlazados a datos?
54. Menciona 3 controles web que soporten plantillas.
55. Si se usa un control `Repeater` y se enlaza a un origen de datos, que hace falta para visualizar sus datos?
- 56.Cuál es el principal elemento de una plantilla que nunca debe faltar y especifica el diseño de las filas a repetirse?
57. Con qué etiquetas (tags) se indica en una plantilla que vamos a usar código incrustado del servidor, por ejemplo para llamar a una función?

- 58.Cuál es la diferencia del control DataList con respecto al Repeater?
- 59. Qué indica el elemento AlternatingItemTemplate en una plantilla?
- 60. En qué tipo de controles (HTML o Web) puede usarse el código incrustado del servidor?
- 61. Qué es una plantilla jerárquica y para qué sirve?
- 62. En una plantilla de un DataList con qué código incrustado del servidor se obtiene el índice de la fila actual enlazada al origen de datos?

Capítulo 6: Desarrollando Aplicaciones WPF

WPF es un conjunto de componentes que permiten crear aplicaciones de alto impacto visual usando para ello un lenguaje de marcado de aplicaciones XML conocido por sus siglas como XAML, además las aplicaciones construidas en WPF pueden ser para Windows o ejecutarse en un Browser (XBAPs).

A diferencia de las aplicaciones tradicionales Windows o Web en WPF se reduce enormemente la escritura del código en un lenguaje como Visual Basic ya que se podría hacer la mayor parte usando solo XAML, es decir a nivel de diseño. Además otras ventajas de WPF son la capacidad para manejar multimedia, documentos, enlace de datos, graficos en 3D, animaciones, etc.

Al inicio de este capítulo conoceremos los fundamentos de WPF: su arquitectura, características y clasificación de controles, además veremos como usar controles WPF creando aplicaciones Windows y Aplicaciones para el Browser XAML (XBAPs).

Luego veremos todo lo relacionado al enlace de datos, desde como usar controles para realizar enlace simple y complejo, cómo dar formato o transformar los valores de los controles enlazados a datos, hasta usar plantillas para personalizar la apariencia del control, inclusive como crear plantillas jerárquicas. Además se incluye el control DataGrid.

En la tercera sección trataremos sobre el manejo de documentos en WPF, presentaremos documentos fijos usando el DocumentViewer, crearemos anotaciones y también trabajaremos con documentos dinámicos usando el control FlowDocumentReader.

Por último, aprenderemos como incorporar la funcionalidad de multimedia en nuestras aplicaciones WPF, desde incluir voz hasta reproducir audio y video usando el control MediaElement.

1. Creando Aplicaciones Básicas en WPF

Iniciaremos este capítulo con una introducción a WPF, en la cual veremos sus fundamentos: arquitectura, características y cómo se clasifican sus controles.

Después aprenderemos a crear aplicaciones básicas en WPF tanto en Windows como para el Explorador, ambas usan los mismos controles WPF, la diferencia es que el contenedor de los controles en una aplicación Windows WPF es el objeto Window y en el Explorador es el objeto Page.

Además en WPF para Windows las aplicaciones tienen plena confianza y pueden ejecutar cualquier acción en el sistema, en cambio en WPF para el Explorador (por defecto) solo tiene confianza parcial y no puede ejecutar todas las acciones como usar diálogos.

También veremos como usar cuadros de diálogos, los cuales pueden ser cuadros de mensajes, cuadros de diálogos comunes como abrir, guardar e imprimir y cuadros de diálogos personalizados.

1.1. Introducción a WPF

Windows Presentations Foundation (WPF) es un modelo de programación unificado para trabajar con aplicaciones de alto impacto visual que usen gráficos, documentos, multimedia, etc. El núcleo de WPF es un motor de representación independiente de la resolución y basado en vectores construido para aprovechar al máximo el hardware de gráficos moderno.

Arquitectura de WPF

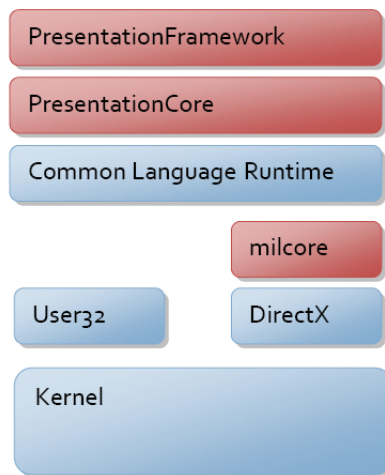
La arquitectura de WPF esta compuesta por un conjunto de 3 componentes principales que son:

- **PresentationFramework:** Es un componente manejado por el CLR que se encarga de proveer los controles WPF.

- PresentationCore: Es un componente manejado por el CLR que contiene las características básicas de WPF.
- Milcore: Es un componente No manejado por el CLR que permite la integración con DirectX para manejar gráficos en 3D.

En el gráfico 6.1 se puede apreciar la arquitectura de los componentes que forman parte de WPF.

Gráfico 6.1: Arquitectura de componentes de WPF



El espacio de nombres donde se encuentran la mayoría de clases de WPF es: `System.Windows`, y las principales clases de WPF son las siguientes:

- `System.Object`
- `System.Threading.DispatcherObject`
- `System.Windows.DependencyObject`
- `System.Windows.Media.Visual`
- `System.Windows.UIElement`
- `System.Windows.FrameworkElement`
- `System.Windows.Controls.Control`

Para obtener más información sobre la Arquitectura de WPF ver la referencia 36 al final del libro.

Características de WPF

Entre las principales características de WPF que lo diferencian de otros componentes del .NET Framework como Windows Forms o Web Forms tenemos:

- **Lenguaje de Marcado de Aplicaciones XML (XAML):** Este permite en forma declarativa mediante etiquetas (tags) realizar la presentación y el trabajo con controles WPF, minimizando el uso de un lenguaje de programación .NET.
- **Tipos de Aplicaciones WPF:** Se pueden crear aplicaciones Windows independientes (exe) basadas en ventanas, o aplicaciones que se ejecutan en un explorador (Browser) conocidas como XBAPs que son siglas de XAML Browser Applications.
- **Controles Integrados:** Sin importar el tipo de aplicación WPF, los controles están disponibles para ambos tipos de aplicaciones.
- **Enlace de Datos:** WPF dispone de un nuevo modelo de enlace de datos que permite enlazar cualquier origen de datos a un control o destino de enlace.
- **Gráficos:** WPF permite crear gráficos independientes de la resolución, con precisión mejorada, gráficos avanzados en 2D y 3D.
- **Animación:** También podemos crear animaciones de forma simple, podemos animar alguna propiedad de un control.
- **Multimedia:** Con WPF podemos aprovechar reproducir voz mediante la librería Speech, o reproducir audio y video mediante el control MediaElement.
- **Documentos:** WPF permite trabajar con 3 tipos de documentos que son documentos de flujo, documentos fijos y documentos de XML Paper

Specification (XPS). Además podemos crear anotaciones sobre dichos documentos.

- Estilos y Plantillas: Ambos permiten cambiar la apariencia de un control para simplificar el diseño de una aplicación WPF.
- Recursos: Podemos definir una sección donde se encuentren elementos reusables como por ejemplo estilos, plantillas, etc.

Clasificación de Controles WPF

Los controles WPF los podemos clasificar en:

- Botones: Button y RepeatButton.
- Cuadros de diálogo: OpenFileDialog, PrintDialog y SaveFileDialog.
- Entradas manuscritas digitales: InkCanvas y InkPresenter.
- Documentos: DocumentViewer, FlowDocumentPageViewer, Flow DocumentReader, FlowDocumentScrollViewer y StickyNoteControl.
- Entrada: TextBox, RichTextBox y PasswordBox.
- Diseño: Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, Virtualizing StackPanel, Window y WrapPanel.
- Multimedia: Image, MediaElement y SoundPlayerAction.
- Menús: ContextMenu, Menu y ToolBar.
- Navegación: Frame, Hyperlink, Page, NavigationWindow y TabControl.
- Selección: CheckBox, ComboBox, ListBox, TreeView y RadioButton, Slider.
- Información para el usuario: AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock y ToolTip.

1.2. Trabajando con Ventanas

Al crear aplicaciones WPF para Windows se crea por defecto una clase window o ventana sobre la cual se realiza el diseño usando un contenedor principal, por defecto el Grid y Controles WPF.

Las ventanas tienen propiedades comunes como "Title" que indica el texto a mostrar en la barra de título, "ResizeMode" que indica si la ventana es modificable o no, "WindowStartupLocation" que especifica la ubicación donde se mostrará la ventana o "WindowState" que especifica si se mostrará maximizada, minimizada o normal.

Para navegar entre ventanas se usa los métodos Show y ShowDialog, el primero muestra la ventana como no modal y el segundo como modal, es decir el segundo como diálogo.

Por su parte el diseño sobre la ventana tiene que realizarse sobre un contenedor que por defecto es el Grid, sobre el cual se pueden definir filas y columnas para realizar un mejor diseño y sobre estas celdas ubicar los controles WPF.

También existen otros contenedores para agrupar controles como StackPanel que agrupa en forma vertical, DockPanel que agrupa en forma horizontal, WrapPanel que agrupa en forma consecutiva y va bajando, etc.

Finalmente, para programar se hace sobre eventos de la ventana o de los controles, en el caso de la ventana sus principales eventos son: "Loaded" que ocurre cuando carga la ventana, Closing que ocurre cuando se está cerrando, "Closed" cuando ya se cerró y "Unloaded" cuando se descargó.

Todo procedimiento manejador de eventos en WPF debe tener 2 parámetros: sender de tipo System.Object y e de tipo System.Windows.RoutedEventArgs.

Para obtener más información sobre ventanas de WPF ver la referencia 37 al final del libro.

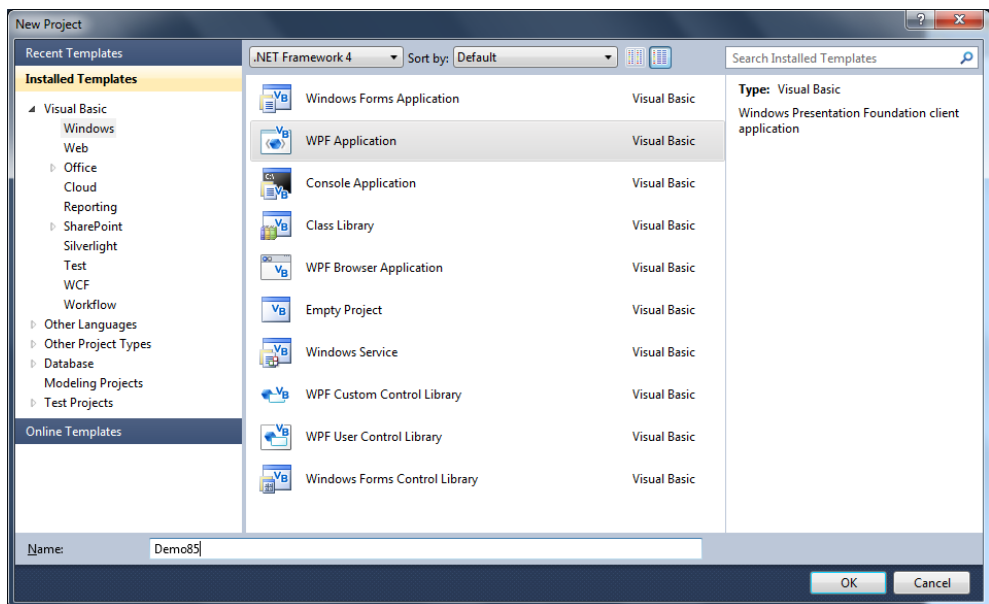
A continuación una aplicación WPF para Windows que tiene 2 ventanas, la primera permite registrar los datos del alumno en un archivo de texto: Nombre, FechaNac, Sexo y Distrito; mientras la segunda permite listar los

alumnos registrados en un ListView de 4 columnas usando para el llenado una clase de entidad llamada beAlumno.

Demo 85

- Crear una nueva aplicación WPF para Windows: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application" similar al diálogo mostrado en la figura 6.2.

Gráfico 6.2: Diálogo de crear nueva aplicación WPF Windows



- Escribir en el nombre de la aplicación "Demo85" y clic en "OK".
- Por defecto se crearán 2 archivos xaml: "Application.xaml" y "Main Window.xaml"

Nota: El primero contiene la definición de los espacios de nombre usados por WPF y cual es la ventana (archivo xaml) que inicia, que por defecto es "MainWindow.xaml".

- Cambiar de nombre a la ventana "MainWindow.xaml" por "Ficha Alumno.xaml".

- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="FichaAlumno"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="Aplicación WPF Windows de Alumnos" Height="300" Width="350"
WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
<Grid>
    <Grid.Background>
        <LinearGradientBrush>
            <GradientStop Color="Aqua" Offset="1"/>
            <GradientStop Color="Blue" Offset="0.5"/>
        </LinearGradientBrush>
    </Grid.Background>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="50"/>
        <ColumnDefinition Width="100"/>
        <ColumnDefinition Width="120"/>
        <ColumnDefinition Width="50*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="30"/>
        <RowDefinition Height="30"/>
        <RowDefinition Height="30"/>
        <RowDefinition Height="30"/>
        <RowDefinition Height="60"/>
        <RowDefinition Height="30"/>
        <RowDefinition Height="30"/>
        <RowDefinition Height="30*" />
    </Grid.RowDefinitions>
    <Label Name="lblTitulo" Grid.Row="1" Grid.Column="1"
        Grid.ColumnSpan="2" Content="Ficha de Registro del Alumno"
        HorizontalAlignment="Center" Foreground="White" FontSize="16"/>
    <Label Name="lblNombre" Grid.Row="2" Grid.Column="1"
        Content="Nombre:" Foreground="White"
        VerticalAlignment="Center" />
    <TextBox Name="txtNombre" Grid.Row="2" Grid.Column="2"
        Width="100" Height="20"
        HorizontalAlignment="Left"/>
    <Label Name="lblFechaNac" Grid.Row="3" Grid.Column="1"
        Content="Fecha Nac:" Foreground="White"
```

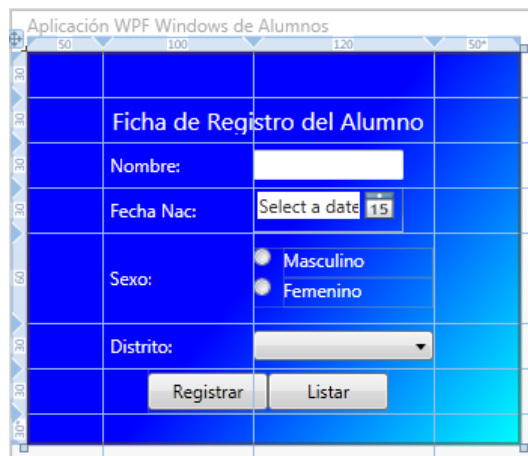
```
HorizontalAlignment="Left" VerticalAlignment="Center"/>
<DatePicker Name="dpFechaNac" Grid.Row="3" Grid.Column="2"
Width="100" HorizontalAlignment="Left"/>
<Label Name="lblSexo" Grid.Row="4" Grid.Column="1"
Content="Sexo:" Foreground="White"
HorizontalAlignment="Left" VerticalAlignment="Center"/>
<Grid Grid.Row="4" Grid.Column="2"
HorizontalAlignment="Left" VerticalAlignment="Center">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="20"/>
<ColumnDefinition Width="100"/>
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="20"/>
<RowDefinition Height="20"/>
</Grid.RowDefinitions>
<RadioButton Name="rbMasculino" Grid.Row="0"
Grid.Column="0" Height="20"/>
<TextBlock Name="lblMasculino" Grid.Row="0"
Grid.Column="1" Height="20" Foreground="White">
Masculino</TextBlock>
<RadioButton Name="rbFemenino" Grid.Row="1"
Grid.Column="0" Height="20"/>
<TextBlock Name="lblFemenino" Grid.Row="1"
Grid.Column="1" Height="20" Foreground="White">
Femenino</TextBlock>
</Grid>
<Label Name="lblDistrito" Grid.Row="5" Grid.Column="1"
Content="Distrito:" Foreground="White"
HorizontalAlignment="Left" VerticalAlignment="Center"/>
<ComboBox Name="cboDistrito" Grid.Row="5" Grid.Column="2"
HorizontalAlignment="Left" Width="120" Height="20"/>
<DockPanel Grid.Row="6" Grid.Column="1" Grid.ColumnSpan="2"
HorizontalAlignment="Center">
<Button Name="btnRegistrar"
Content="Registrar" Cursor="Hand" Width="80" Height="25"
ToolTip="Grabar los datos del Alumno"/>
<Button Name="btnListar"
Content="Listar" Cursor="Hand" Width="80" Height="25"
ToolTip="Visualiza un diálogo con la lista de alumnos"/>
```

```
</DockPanel>
</Grid>
</Window>
```

Nota: En el código anterior se define un fondo azul lineal degradado para el Grid y también se definen 4 columnas y 8 filas sobre las cuales se ubican los controles WPF.

- El diseño de la ventana debe quedar como se muestra en el gráfico 6.3.

Gráfico 6.3: Diseño de la ventana Ficha del Alumno



- Insertar un archivo de texto para definir los distritos: del menú "Project", seleccionar "Add New Item", en la categoría "General", seleccionar "Text File" y escribir como nombre: "Distritos.txt".
- Ingresar los Distritos en cada línea del archivo, por ejemplo: Ate, Barranco, Comas, Chorrillos, Miraflores, Pueblo Libre, San Isidro, SJL, San Miguel y Otros.
- Escribir el siguiente código en el archivo vb asociado al xaml:

```
Imports System.IO
```

```
Class FichaAlumno
```

```
    Private Ruta As String = "C:\Data\DemosLibro\LibroVB2010\Demo85\"
```

```
    Private Sub ListarDistritos(ByVal sender As System.Object, _
```

```
ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim Archivo As String = Ruta + "Distritos.txt"
    If File.Exists(Archivo) Then
        Dim Distritos() As String = File.ReadAllLines(Archivo)
        cboDistrito.ItemsSource = Distritos
    End If
End Sub

Private Sub RegistrarAlumno(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs) Handles btnRegistrar.Click
    If txtNombre.Text <> "" Then
        If dpFechaNac.SelectedDate IsNot Nothing Then
            If rbMasculino.IsChecked OrElse rbFemenino.IsChecked Then
                If cboDistrito.SelectedIndex > -1 Then
                    Using sw As New StreamWriter(Ruta + "Alumnos.txt", True)
                        sw.WriteLine("{0},{1},{2},{3}", _
                            txtNombre.Text, String.Format("{0:d}", _
                                dpFechaNac.SelectedDate), _
                                If(rbMasculino.IsChecked, "Masculino", "Femenino"), _
                                cboDistrito.Text)
                    End Using
                    txtNombre.Clear()
                    dpFechaNac.SelectedDate = Nothing
                    rbMasculino.IsChecked = False
                    rbFemenino.IsChecked = False
                    cboDistrito.SelectedIndex = -1
                Else
                    MessageBox.Show("Selecciona el Distrito")
                    cboDistrito.Focus()
                End If
            Else
                MessageBox.Show("Selecciona el Sexo")
            End If
        Else
            MessageBox.Show("Selecciona la Fecha")
            dpFechaNac.Focus()
        End If
    Else
        MessageBox.Show("Ingresa el Nombre")
        txtNombre.Focus()
    End If
End Sub
```



```
End If
End Sub

Private Sub MostrarDialogoListar(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs) Handles btnListar.Click
    Dim oLista As New ListaAlumnos
    oLista.ShowDialog()
End Sub
End Class
```

Nota: En el código anterior al cargar la ventana se verifica si existe el archivo "Distritos.txt" y si es así se lee todas sus líneas y se enlaza al combo de distritos. Además para registrar el alumno en el archivo de texto "Alumnos.txt" es necesario ingresar todos los datos.

- Agregar otra ventana al proyecto: del menú "Project", seleccionar "Add Window", escribir como nombre: "ListaAlumnos.xaml" y click en "Add".
- Escribir el siguiente código en el archivo xaml:

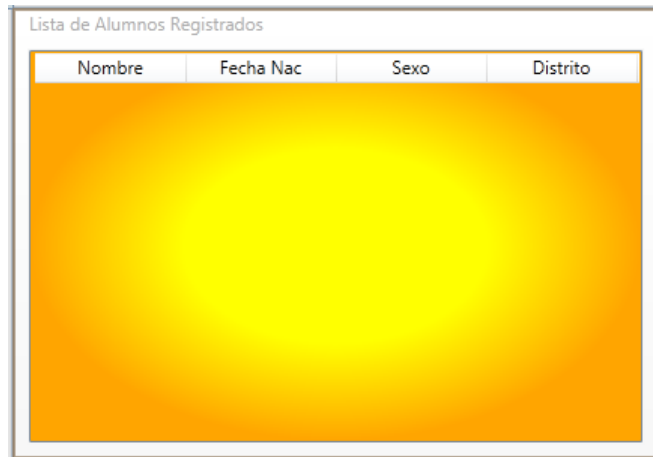
```
<Window x:Class="ListaAlumnos"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="Lista de Alumnos Registrados" Height="300" Width="430"
WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
<Grid>
<ListView Name="lvwAlumno">
<ListView.Background>
<RadialGradientBrush>
<GradientStop Color="orange" Offset="1"/>
<GradientStop Color="yellow" Offset="0.5"/>
</RadialGradientBrush>
</ListView.Background>
<ListView.View>
<GridView>
<GridViewColumn Header="Nombre" Width="100"
DisplayMemberBinding="{Binding Path=Nombre}"/>
<GridViewColumn Header="Fecha Nac" Width="100"
DisplayMemberBinding="{Binding Path=FechaNac}"/>
<GridViewColumn Header="Sexo" Width="100"
DisplayMemberBinding="{Binding Path=Sexo}"/>
```

```
<GridViewColumn Header="Distrito" Width="100"  
    DisplayMemberBinding="{Binding Path=Distrito}"/>  
</GridView>  
</ListView.View>  
</ListView>  
</Grid>  
</Window>
```

Nota: En el código anterior se define un fondo radial naranja para el Grid y se agrega un ListView con 4 columnas enlazadas a una entidad.

- El diseño de la ventana debe quedar como se muestra en el gráfico 6.4.

Gráfico 6.4: Diseño de la ventana Lista de Alumnos



- Agregar una clase entidad para los alumnos: del menú "Project", seleccionar "Add Class", escribir como nombre: "beAlumno" y "Add".
- Escribir el siguiente código en la clase beAlumno:

```
Public Class beAlumno  
    Public Property Nombre As String  
    Public Property FechaNac As String  
    Public Property Sexo As String  
    Public Property Distrito As String  
End Class
```

- Cerrar la ventana de la clase y regresar a la ventana Lista de Alumnos.

- Escribir el siguiente código en el archivo vb asociado al archivo "ListaAlumnos.xaml":

```
Imports System.IO
```

```
Public Class ListaAlumnos
```

```
Private Sub LeerArchivo(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded  
    Dim Archivo As String = _  
        "C:\Data\DemosLibro\LibroVB2010\Demo85\Alumnos.txt"  
    If File.Exists(Archivo) Then  
        Dim Alumnos() As String = File.ReadAllLines(Archivo)  
        Dim Alumno() As String  
        Dim obeAlumno As beAlumno  
        For I = 0 To Alumnos.Length - 1  
            obeAlumno = New beAlumno  
            Alumno = Alumnos(I).Split(",")  
            obeAlumno.Nombre = Alumno(0)  
            obeAlumno.FechaNac = Alumno(1)  
            obeAlumno.Sexo = Alumno(2)  
            obeAlumno.Distrito = Alumno(3)  
            lwAlumno.Items.Add(obeAlumno)  
        Next  
    End If  
End Sub  
End Class
```

Nota: En el código anterior al cargar la ventana se verifica si existe el archivo "Alumnos.txt" y si es así se lee todas sus líneas y se recorre cada línea para llenar el objeto obeAlumno y agregarlo al ListView.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "FichaAlumno.xaml".
- Grabar y luego ejecutar la aplicación pulsando F5.
- Se mostrará una ventana similar al de la figura 6.5.

Gráfico 6.5: Ejecución de la ventana Ficha del Alumno

Aplicación WPF Windows de Alumnos

Ficha de Registro del Alumno

Nombre: Luis Duenas

Fecha Nac: 12/09/1970 15

Sexo: ☒ Masculino ☐ Femenino

Distrito: San Isidro

Registrar Listar

- Clic al botón "Registrar" para ver que se active la validación, luego registrar varios alumnos.
- Finalmente, clic al botón "Listar" para ver todos los alumnos registrados

Gráfico 6.6: Ejecución de la ventana Lista de Alumnos

Lista de Alumnos Registrados

Nombre	Fecha Nac	Sexo	Distrito
Luis Duenas	12/09/1970	Masculino	San Isidro
Raul Anton	16/02/1980	Masculino	San Miguel
Ana Perez	26/07/1985	Femenino	Miraflores
Teresa Rojas	01/10/1975	Femenino	Pueblo Libre

1.3. Trabajando con Páginas

Al crear aplicaciones WPF del Explorador XAML o XBAP se crea por defecto una clase page o página sobre la cual se realiza el diseño usando un contenedor principal, por defecto el Grid y Controles WPF.

Las aplicaciones WPF del Explorador XAML se pueden alojar solo en Microsoft Internet Explorer a partir de la versión 6, pero también en contenedores de navegación alternativos:

- Frame, para hospedar islas de contenido navegable en páginas o ventanas.
- NavigationWindow, para hospedar contenido navegable en una ventana completa.

Los controles WPF para las páginas son los mismos que para una ventana, la diferencia principal es que desde una aplicación WPF XBAP por defecto no se puede ejecutar cualquier acción como en Windows, ya que tiene solo confianza parcial. Si deseamos que pueda trabajar con diálogos, archivos, el registro de Windows, etc hay que darle plena confianza al ensamblado.

La otra diferencia al programar con páginas es que no existen los eventos "Closing" y "Closed" que son exclusivos de la ventana, solo el "Loaded" y "Unloaded".

Para navegar entre páginas existe la clase NavigationService que tiene el método "Navigate" para ir a una determinada página, los métodos "GoBack" para retroceder y "GoForward" para avanzar a la siguiente página, además de propiedades para validar el avance como "CanGoBack", "CanGoForward" y "CurrentSource" que devuelve la URI de la última página navegada.

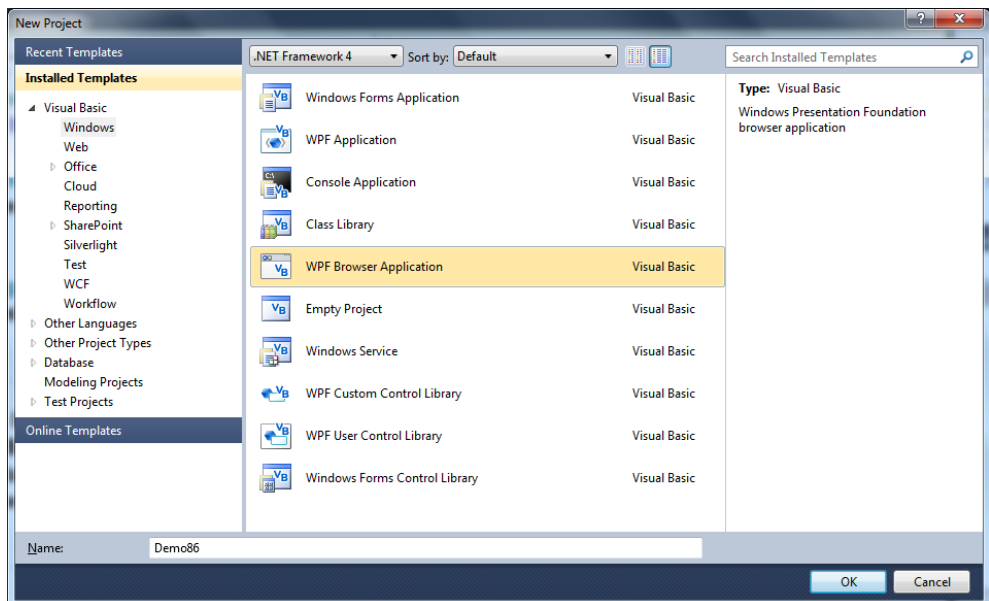
Para obtener más información sobre navegación ver la referencia 38 al final del libro.

A continuación, un ejemplo basado en el demo anterior de Registro y Lista de Alumnos pero esta vez como una aplicación del explorador XAML o XBAP, en donde aprenderemos cómo navegar entre páginas y cómo configurar la seguridad de la aplicación XBAP para darle confianza total.

Demo 86

- Crear una nueva aplicación WPF del Explorador XAML: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Browser Application" similar al diálogo mostrado en la figura 6.7.

Gráfico 6.7: Diálogo de crear nueva aplicación WPF del Navegador



- Escribir en el nombre de la aplicación "Demo86" y clic en "OK".
- Se crearán 2 archivos xaml: "Application.xaml" y "Page1.xaml"
- Cambiar de nombre a la página "Page1.xaml" por "FichaAlumno.xaml".
- Copiar todo lo que está dentro de la sección Grid de la ventana FichaAlumno.xaml del demo anterior a la sección grid de la página:

```
<Page x:Class="FichaAlumno"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
mc:Ignorable="d"
d:DesignHeight="300" d:DesignWidth="400"
Title=" Aplicación WPF del Explorador">
<Grid>
  <Grid.Background>
    <LinearGradientBrush>
      <GradientStop Color="Aqua" Offset="1"/>
      <GradientStop Color="Blue" Offset="0.5"/>
    </LinearGradientBrush>
  </Grid.Background>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="50"/>
    <ColumnDefinition Width="100"/>
    <ColumnDefinition Width="120"/>
    <ColumnDefinition Width="50*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="60"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30*" />
  </Grid.RowDefinitions>
  <Label Name="lblTitulo" Grid.Row="1" Grid.Column="1"
    Grid.ColumnSpan="2"
    Content="Ficha de Registro del Alumno"
    HorizontalAlignment="Center"
    Foreground="White" FontSize="16"/>
  <Label Name="lblNombre" Grid.Row="2" Grid.Column="1"
    Content="Nombre:" Foreground="White"
    VerticalAlignment="Center" />
  <TextBox Name="txtNombre" Grid.Row="2" Grid.Column="2"
    Width="100" Height="20"
    HorizontalAlignment="Left"/>
  <Label Name="lblFechaNac" Grid.Row="3" Grid.Column="1"
    Content="Fecha Nac:" Foreground="White"
```

```
HorizontalAlignment="Left" VerticalAlignment="Center"/>
<DatePicker Name="dpFechaNac" Grid.Row="3" Grid.Column="2"
Width="100" HorizontalAlignment="Left"/>
<Label Name="lblSexo" Grid.Row="4" Grid.Column="1"
Content="Sexo:" Foreground="White"
HorizontalAlignment="Left" VerticalAlignment="Center"/>
<Grid Grid.Row="4" Grid.Column="2" HorizontalAlignment="Left"
VerticalAlignment="Center">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="20"/>
<ColumnDefinition Width="100"/>
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="20"/>
<RowDefinition Height="20"/>
</Grid.RowDefinitions>
<RadioButton Name="rbMasculino" Grid.Row="0"
Grid.Column="0" Height="20"/>
<TextBlock Name="lblMasculino" Grid.Row="0"
Grid.Column="1" Height="20" Foreground="White">
Masculino</TextBlock>
<RadioButton Name="rbFemenino"
Grid.Row="1" Grid.Column="0" Height="20"/>
<TextBlock Name="lblFemenino" Grid.Row="1"
Grid.Column="1" Height="20" Foreground="White">
Femenino</TextBlock>
</Grid>
<Label Name="lblDistrito" Grid.Row="5" Grid.Column="1"
Content="Distrito:" Foreground="White"
HorizontalAlignment="Left" VerticalAlignment="Center"/>
<ComboBox Name="cboDistrito" Grid.Row="5" Grid.Column="2"
HorizontalAlignment="Left" Width="120" Height="20"/>
<DockPanel Grid.Row="6" Grid.Column="1" Grid.ColumnSpan="2"
HorizontalAlignment="Center">
<Button Name="btnRegistrar"
Content="Registrar" Cursor="Hand" Width="80" Height="25"
ToolTip="Grabar los datos del Alumno"/>
<Button Name="btnListar"
Content="Listar" Cursor="Hand" Width="80" Height="25"
ToolTip="Visualiza una pagina con la lista de alumnos registrados"/>
```

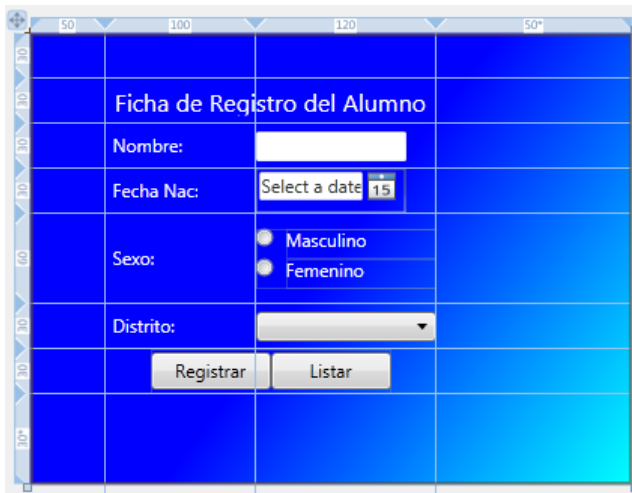


```
</DockPanel>  
</Grid>  
</Page>
```

Nota: En realidad con respecto al demo anterior solo ha cambiado el título de la página y el Tooltip del botón Listar.

- El diseño de la página debe quedar como se muestra en el gráfico 6.8.

Gráfico 6.8: Diseño de la página Ficha del Alumno



- Agregar el archivo de texto distritos.txt y la clase de entidad beAlumno.vb creadas en el demo anterior al proyecto actual.
- Escribir el siguiente código en el archivo vb asociado al xaml:

```
Imports System.IO
```

```
Class FichaAlumno
```

```
    Private Ruta As String = "C:\Data\DemosLibro\LibroVB2010\Demo86\"
```

```
    Private Sub ListarDistritos(ByVal sender As System.Object, _  
        ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded  
        Dim Archivo As String = Ruta + "Distritos.txt"  
        If File.Exists(Archivo) Then  
            Dim Distritos() As String = File.ReadAllLines(Archivo)  
            cboDistrito.ItemsSource = Distritos
```

```
End If
End Sub

Private Sub RegistrarAlumno(ByVal sender As Object, _
ByVal e As System.Windows.RoutedEventArgs) Handles btnRegistrar.Click
    If txtNombre.Text <> "" Then
        If dpFechaNac.SelectedDate IsNot Nothing Then
            If rbMasculino.IsChecked OrElse rbFemenino.IsChecked Then
                If cboDistrito.SelectedIndex > -1 Then
                    Using sw As New StreamWriter(Ruta + "Alumnos.txt", True)
                        sw.WriteLine("{0},{1},{2},{3}", _
                            txtNombre.Text, String.Format("{0:d}", _
                                dpFechaNac.SelectedDate), _
                                If(rbMasculino.IsChecked, "Masculino", "Femenino"), _
                                cboDistrito.Text)
                    End Using
                    txtNombre.Clear()
                    dpFechaNac.SelectedDate = Nothing
                    rbMasculino.IsChecked = False
                    rbFemenino.IsChecked = False
                    cboDistrito.SelectedIndex = -1
                Else
                    MessageBox.Show("Selecciona el Distrito")
                    cboDistrito.Focus()
                End If
            Else
                MessageBox.Show("Selecciona el Sexo")
            End If
        Else
            MessageBox.Show("Selecciona la Fecha")
            dpFechaNac.Focus()
        End If
    Else
        MessageBox.Show("Ingresa el Nombre")
        txtNombre.Focus()
    End If
End Sub

Private Sub MostrarPaginaListar(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs) Handles btnListar.Click
```

```
Dim oLista As New ListaAlumnos
NavigationService.Navigate(New ListaAlumnos, UriKind.Relative)
End Sub
End Class
```

Nota: En el código anterior solo ha cambiado la ruta del archivo a Demo86 y el procedimiento manejador de eventos "MostrarPaginaListar" para navegar a la otra página usando el método Navigate de la clase Navigation Service.

- Agregar otra página al proyecto: del menú "Project", seleccionar "Add Page", escribir como nombre: "ListaAlumnos.xaml" y click en "Add".
- Copiar todo lo que está dentro de la sección Grid de la ventana ListaAlumnos.xaml del demo anterior a la sección grid de la nueva página y aumentar al final un botón para regresar a la página anterior:

```
<Page x:Class="ListaAlumnos"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="350" d:DesignWidth="400"
      Title="ListaAlumnos">
  <Page.Background>
    <RadialGradientBrush>
      <GradientStop Color="orange" Offset="1"/>
      <GradientStop Color="yellow" Offset="0.5"/>
    </RadialGradientBrush>
  </Page.Background>
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="300"/>
      <RowDefinition Height="30"/>
    </Grid.RowDefinitions>
    <ListView Name="lvwAlumno" Height="300"
              VerticalAlignment="Top">
      <ListView.Background>
        <RadialGradientBrush>
```

```
<GradientStop Color="orange" Offset="1"/>
<GradientStop Color="yellow" Offset="0.5"/>
</RadialGradientBrush>
</ListView.Background>
<ListView.View>
  <GridView>
    <GridViewColumn Header="Nombre" Width="100"
      DisplayMemberBinding="{Binding Path=Nombre}"/>
    <GridViewColumn Header="Fecha Nac" Width="100"
      DisplayMemberBinding="{Binding Path=FechaNac}"/>
    <GridViewColumn Header="Sexo" Width="100"
      DisplayMemberBinding="{Binding Path=Sexo}"/>
    <GridViewColumn Header="Distrito" Width="100"
      DisplayMemberBinding="{Binding Path=Distrito}"/>
  </GridView>
</ListView.View>
</ListView>
<Button Name="lblRegresar" Foreground="Blue" Content="Regresar"
  Grid.Row="1" HorizontalAlignment="Center">
</Button>
</Grid>
</Page>
```

Nota: En realidad con respecto al demo anterior solo ha cambiado el alto y ancho de la página y ha aumentado el botón para Regresar.

- El diseño de la página debe quedar como se muestra en el gráfico 6.9.

Gráfico 6.9: Diseño de la página Lista de Alumnos



- Escribir el siguiente código en el archivo vb asociado al archivo "ListaAlumnos.xaml":

Imports System.IO

Class ListaAlumnos

```
Private Sub LeerArchivo(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim Archivo As String = _
        "C:\Data\DemosLibro\LibroVB2010\Demo86\Alumnos.txt"
    If File.Exists(Archivo) Then
        Dim Alumnos() As String = File.ReadAllLines(Archivo)
        Dim Alumno() As String
        Dim obeAlumno As beAlumno
        For I = 0 To Alumnos.Length - 1
            obeAlumno = New beAlumno
            Alumno = Alumnos(I).Split(",")
            obeAlumno.Nombre = Alumno(0)
            obeAlumno.FechaNac = Alumno(1)
            obeAlumno.Sexo = Alumno(2)
            obeAlumno.Distrito = Alumno(3)
            lwAlumno.Items.Add(obeAlumno)
        Next
    End If
End Sub
```

```
End If  
End Sub
```

```
Private Sub RegresarPaginaAnterior(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles lblRegresar.Click  
    NavigationService.GoBack()  
End Sub  
End Class
```

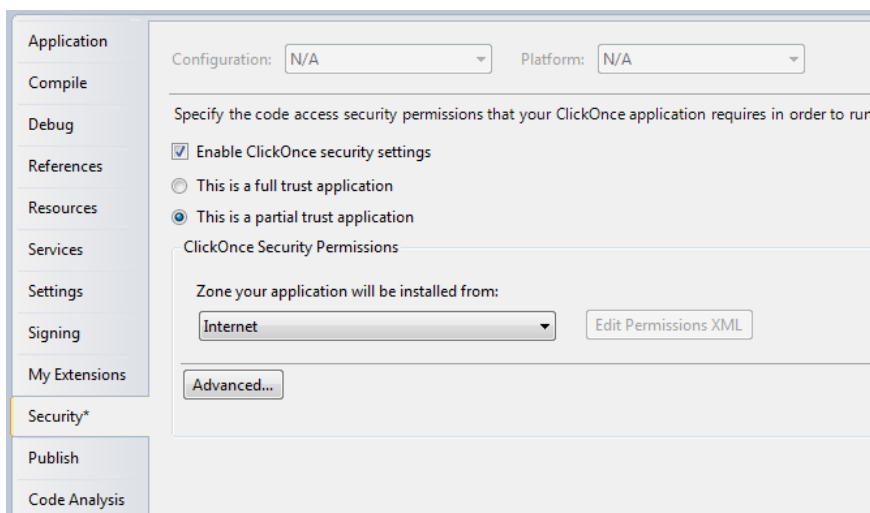
Nota: En realidad con respecto al demo anterior solo ha cambiado la ruta del archivo a Demo86 y ha aumentado el código para regresar a la página anterior.

- Regresar a la ventana "Application.xaml" y cambiar la página que inicia (atributo StartupUri) a "FichaAlumno.xaml".

Nota: Antes de ejecutar la aplicación debemos configurar la seguridad de la aplicación WPF para el Browser cambiándola de confianza parcial a plena confianza, sino al ejecutar ocurrirá un error al leer y escribir en archivos.

- Clic derecho sobre el nombre del proyecto y del menú contextual elegir "Properties".
- Aparecerá la ventana de propiedades, seleccionar la ficha "Security"

Gráfico 6.10: Ficha Seguridad de la ventana de propiedades



- Cambiar la seguridad eligiendo la primera opción en vez de la segunda: "This is a full trust application", luego cerrar la ventana.
- Grabar y luego ejecutar la aplicación pulsando F5.
- Se mostrará una página similar al de la figura 6.11.

Gráfico 6.11: Ejecución de la página Ficha del Alumno

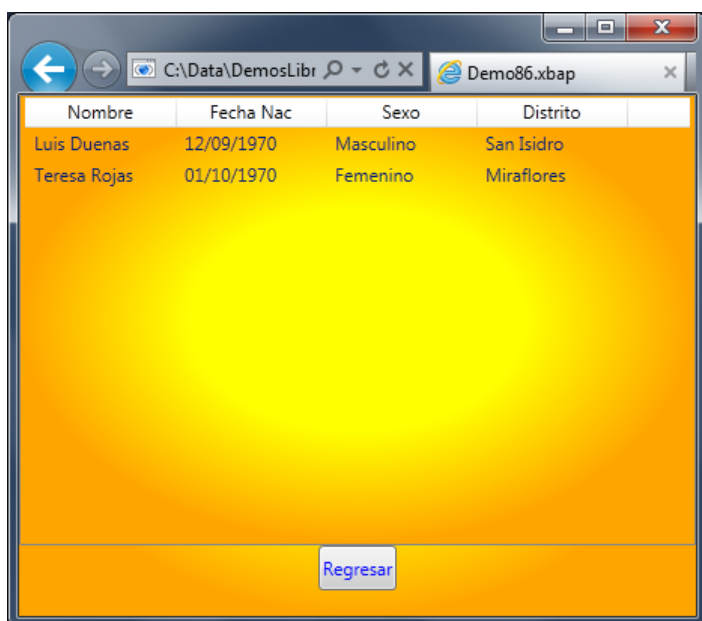


The screenshot shows a web browser window with the address bar displaying 'C:\Data\DemosLibr' and the file 'Demo86.xbap'. The page has a blue background and contains a form titled 'Ficha de Registro del Alumno'. The form includes the following fields and controls:

- Nombre:** A text input field containing 'Luis Duenas'.
- Fecha Nac:** A date picker showing '12/09/1970' with a calendar icon.
- Sexo:** Two radio buttons labeled 'Masculino' (selected) and 'Femenino'.
- Distrito:** A dropdown menu showing 'San Isidro'.
- Buttons:** Two buttons at the bottom labeled 'Registrar' and 'Listar'.

- Clic al botón "Registrar" para ver que se active la validación, luego registrar algunos alumnos.
- Ahora, clic al botón "Listar" para ver todos los alumnos registrados

Gráfico 6.12: Ejecución de la página Lista de Alumnos



- Finalmente, clic al botón “Regresar” para ir a la página anterior.

1.4. Usando Cuadros de Diálogo

En WPF las aplicaciones Windows se componen de una ventana principal pero a veces se requieren más de una ventana o ventanas secundarias a las cuales se les denomina cuadros de diálogo.

Los cuadros de diálogo pueden ser de 2 tipos: modales y no modales; los cuadros de diálogo modales necesitan cerrarse para seguir trabajando con la ventana principal, en cambio, los cuadros de diálogo no modales no, se puede seguir trabajando con la ventana principal una vez abierto (no es necesario cerrar).

WPF permite crear 3 tipos de cuadros de diálogo: cuadros de mensaje, cuadros de diálogo comunes de Windows y cuadros personalizados por el usuario. En este tema nos centraremos en los cuadros de diálogos comunes de Windows, entre los cuales tenemos: abrir, guardar e imprimir.

Para usar los cuadros de diálogo de archivo es necesario usar el espacio de nombres Microsoft.Win32 los cuales tienen las clases OpenFileDialog para manejar el diálogo de abrir archivo y SaveFileDialog para manejar el diálogo de guardar archivo.

Ambos diálogos tienen el método ShowDialog que muestra el diálogo y devuelve un valor lógico que es True en el caso que el usuario acepte la operación de abrir o guardar o False si el usuario cancela la operación.

Por su parte para usar el diálogo de Imprimir se usa la clase PrintDialog que se encuentra en System.Windows.Controls, es decir forma parte de los controles de WPF. Esta clase tiene el método ShowDialog para mostrar el diálogo y el método PrintVisual que permite imprimir el contenido de cualquier control, es decir un gráfico.

Para obtener más información sobre cuadros de diálogo ver la referencia 39 al final del libro.

A continuación, un ejemplo de una aplicación Windows WPF que es un Editor de Textos que tiene un Menú con 4 opciones: nuevo, abrir, guardar e imprimir textos escritos sobre un RichTextBox. Aquí se usan los cuadros de diálogo comunes de Windows.

Demo 87

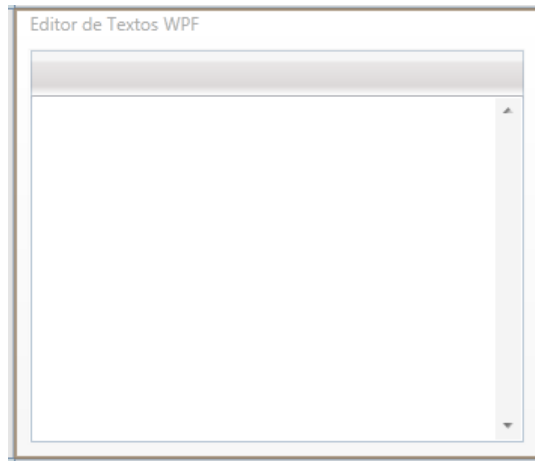
- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo87" y clic en "OK".
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Editor.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="Editor"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Editor de Textos WPF" Height="300" Width="350"
    WindowState="Maximized">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="30"/>
            <RowDefinition Height="300*" />
        </Grid.RowDefinitions>
        <Menu Name="mnuEditor" Grid.Row="0"
            MenuItem.Click="SeleccionaOpcion" />
        <RichTextBox Name="txtEditor" Grid.Row="1"
            VerticalScrollBarVisibility="Visible" />
    </Grid>
</Window>
```

Nota: En el código anterior se definen 2 filas sobre las cuales se ubican los controles WPF Menu y RichTextBox, el menú tiene asociado un procedimiento manejador de eventos al dar clic en cada opción.

- El diseño de la ventana debe quedar como se muestra en el gráfico 6.13.

Gráfico 6.13: Diseño de la ventana Editor de Textos WPF



- Escribir código en el archivo vb asociado al xaml:

```
Imports Microsoft.Win32 'OpenFileDialog, SaveFileDialog
Imports System.IO
```

```
Class Editor
```

```
Private Sub CrearOpcionesMenu(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    mnuEditor.Items.Add("Nuevo")
    mnuEditor.Items.Add("Abrir")
    mnuEditor.Items.Add("Guardar")
    mnuEditor.Items.Add("Imprimir")
End Sub
```

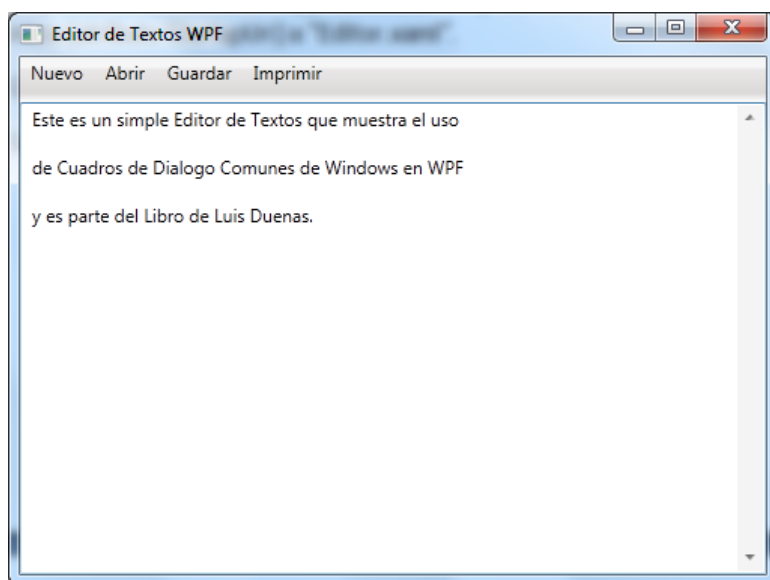
```
Private Sub SeleccionaOpcion(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs)
    Dim mnuOpcion As MenuItem = e.OriginalSource
    If mnuOpcion.Header = "Nuevo" Then
        txtEditor.SelectAll()
        txtEditor.Selection.Text = ""
    ElseIf mnuOpcion.Header = "Abrir" Then
        Dim ofd As New OpenFileDialog
        ofd.Title = "Abrir un archivo de texto"
        ofd.Filter = "Archivos de texto|*.txt"
        If ofd.ShowDialog = True Then
```

```
        txtEditor.Selection.Load _
        (File.OpenRead(ofd.FileName), "Text")
        Me.Title = Path.GetFileName(ofd.FileName)
    End If
    ElseIf mnuOpcion.Header = "Guardar" Then
        Dim sfd As New SaveFileDialog
        sfd.Title = "Guardar un archivo de texto"
        sfd.Filter = "Archivos de texto|*.txt"
        If sfd.ShowDialog = True Then
            txtEditor.SelectAll()
            txtEditor.Selection.Save _
            (File.OpenWrite(sfd.FileName), "Text")
            Me.Title = Path.GetFileName(sfd.FileName)
        End If
    ElseIf mnuOpcion.Header = "Imprimir" Then
        Dim pd As New PrintDialog
        If pd.ShowDialog() = True Then
            pd.PrintVisual(Me, "Mira esto")
        End If
    End If
End Sub
End Class
```

Nota: En el código anterior se agregan 4 opciones al menú al cargar la ventana y al seleccionar una opción se evalúa el texto de la cabecera de la opción y de acuerdo al valor se realiza una acción.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "Editor.xaml".
- Grabar y luego ejecutar la aplicación pulsando F5.
- Se mostrará una ventana similar al de la figura 6.14.

Gráfico 6.14: Ejecución de la ventana Editor de Textos WPF



- Probar cada opción del menú, sobre todo abrir, guardar e imprimir y observar como se presentan los diálogos comunes de Windows.
- Realizar una prueba de impresión al seleccionar la opción Imprimir.

Nota: Si no tiene impresora conectada puede enviar la salida a un archivo XPS, el cual después será usado al aprender como leer documentos XPS.

2. Usando Enlace de Datos

En esta parte veremos como WPF implementa el enlace de datos para trabajar de forma simple la presentación de los datos almacenados en los orígenes de datos sobre la IU mediante los controles enlazados a datos.

Una de las principales ventajas de WPF con respecto a Windows Forms o ASP .NET es que el enlace de datos se encuentra disponible en la mayoría de controles que pueden enlazarse a diferentes orígenes de datos, tales como objetos del CLR por ejemplo Listas, objetos de ADO .NET como por ejemplo DataTable, objetos XML, Servicios Web, etc.

Primero revisaremos qué es el enlace de datos y como nos ayuda a simplificar el trabajo con los datos, para lo cual veremos primero como trabajar sin enlace de datos usando la lectura y escritura de datos.

Luego en la segunda parte implementaremos el enlace de datos tanto simple usando la propiedad DataContext y también el enlace complejo usando la propiedad ItemsSource de los controles de listas como ListBox, ComboBox, ListView, etc.

Como tercer tema aprenderemos a crear convertidores de datos personalizados que ayuden a mejorar los datos que se presentan en los controles enlazados usando para ello una clase que implemente la interface de datos IValueConverter.

Finalmente, veremos como usar las plantillas de datos para mejorar el diseño de los controles enlazados a datos y cambiar su aspecto por defecto, por ejemplo en vez de que el ListBox o ComboBox presenten una sola columna podemos crear una plantilla que permita crear varias columnas y usarla en dichos controles de listas.

En este último tema también trataremos de las plantillas jerárquicas de datos que permiten realizar la presentación de datos en varios niveles sobre un control, por ejemplo en el TreeView para que muestre una tabla dentro de otra como si fuera un TreeTable o TreeGrid.

2.1. Introducción al Enlace de Datos

Para trabajar con datos lo primero que se hace es obtener la data desde un "Repositorio de Datos" como por ejemplo una base de datos, luego esta es almacenada dentro de la aplicación en una estructura en memoria a la cual se le denomina "Origen de Datos" como por ejemplo un arreglo o lista de objetos, finalmente estos datos deben presentarse en pantalla en la Interface de Usuario (IU) sobre los controles.

Para realizar esta presentación de los datos existen 2 técnicas o mecanismos:

- **Método Tradicional:** Consiste en leer del origen de datos y escribir los valores sobre los controles de la IU. Es una forma antigua de presentar datos y hay que escribir demasiado código.
- **Método del Enlace de Datos (Data Binding):** Consiste en crear una conexión entre el origen de datos y los controles de la IU, de tal forma que sea automática los cambios sobre el origen en la IU y viceversa.

Las ventajas de WPF con respecto al enlace de datos son:

- Mayor cantidad de controles que permiten enlace de datos.
- Mayor cantidad de propiedades que se pueden enlazar a datos.
- Más orígenes de datos enlazables como arreglos, listas de objetos, tablas, XML, servicios web, etc.
- Intermediarios que realizan operaciones de ordenación, filtro y navegación por los datos, tal como el `CollectionView`.
- Uso de convertidores de datos, plantillas de datos y reglas de validación de datos, etc.

Para obtener más información sobre el enlace de datos ver la referencia 40 al final del libro.

A continuación veremos un par de ejemplos de como trabajar de forma tradicional para presentar los datos: primero una lista de datos y luego una consulta simple de datos.

Demo 88

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo88" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Lista Empleados.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ListaEmpleados"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Leer y Escribir en un control Lista" Height="350" Width="350"
  ResizeMode="NoResize" WindowStartupLocation="CenterScreen">
  <Grid>
    <ListBox Name="lstEmpleado">
      <ListBox.Background>
        <ImageBrush ImageSource="C:\Data\DemosLibro\LibroVB2010\
          Imagenes\JPG\Empleados\3.jpg" Opacity="0.5"/>
      </ListBox.Background>
    </ListBox>
  </Grid>
</Window>
```

Nota: En el código anterior solo se incluye un control ListBox con un fondo de imagen que esta semi transparente mediante la propiedad opacity.

- El diseño de la ventana debe quedar como se muestra en el gráfico 6.15.

Gráfico 6.15: Diseño de la ventana Lista de Empleados



- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Class ListaEmpleados
```

```
Private Sub ListarEmpleados(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim obrEmpleado As New brEmpleado
    Dim lobeEmpleado As List(Of beEmpleado) = obrEmpleado.Listar
    For I = 0 To lobeEmpleado.Count - 1
        lstEmpleado.Items.Add(String.Format("{0} {1}", _
            lobeEmpleado(I).Apellido, lobeEmpleado(I).Nombre))
    Next
End Sub
End Class
```

Nota: En el código anterior al cargar la ventana se llena una lista de objetos de tipo empleado y luego se recorre cada elemento de la lista para mostrar el apellido y el nombre del empleado en el control ListBox.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ListaEmpleados.xaml".

- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
      data source=Lduenas\MCTS;initial catalog=Northwind"/>  
  </appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.16: Ejecución de la ventana Lista de Empleados



Demo 89

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo89" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Consulta Empleados.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ConsultaEmpleados"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Consulta de Empleados - Leer y Escribir" Height="350" Width="350"
  WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
  <Grid>
    <Grid.Background>
      <ImageBrush ImageSource="C:\Data\DemosLibro\LibroVB2010\
        Imagenes\JPG\Empleados\3.jpg" Opacity="0.5"/>
    </Grid.Background>
    <Grid.RowDefinitions>
      <RowDefinition Height="30" />
      <RowDefinition Height="30" />
      <RowDefinition Height="30" />
      <RowDefinition Height="30" />
      <RowDefinition Height="30" />
      <RowDefinition Height="100" />
      <RowDefinition Height="30" />
      <RowDefinition Height="30" />
      <RowDefinition Height="30*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="50" />
```

```
<ColumnDefinition Width="100" />
<ColumnDefinition Width="120" />
<ColumnDefinition Width="50*" />
</Grid.ColumnDefinitions>
<Label Name="lblTitulo" Grid.Row="0" Grid.Column="1"
    Grid.ColumnSpan="2" Content="Consulta de Empleados"
    FontSize="16" HorizontalAlignment="Center" />
<Label Name="lblCodigo" Grid.Row="1" Grid.Column="1"
    VerticalAlignment="Center" Content="Codigo"
    FontSize="12" HorizontalAlignment="Left" />
<TextBox Name="txtCodigo" Grid.Row="1" Grid.Column="2"
    Width="50" Height="20" HorizontalAlignment="Left" />
<Label Name="lblApellido" Grid.Row="2" Grid.Column="1"
    Content="Apellido" VerticalAlignment="Center"
    FontSize="12" HorizontalAlignment="Left" />
<TextBox Name="txtApellido" Grid.Row="2" Grid.Column="2"
    Width="100" Height="20" HorizontalAlignment="Left" />
<Label Name="lblNombre" Grid.Row="3" Grid.Column="1"
    Content="Nombre" VerticalAlignment="Center"
    FontSize="12" HorizontalAlignment="Left" />
<TextBox Name="txtNombre" Grid.Row="3" Grid.Column="2"
    Width="100" Height="20" HorizontalAlignment="Left" />
<Label Name="lblFechaNac" Grid.Row="4" Grid.Column="1"
    Content="Fecha Nac" VerticalAlignment="Center"
    FontSize="12" HorizontalAlignment="Left" />
<DatePicker Name="dpFechaNac" Grid.Row="4" Grid.Column="2"
    Width="100" Height="20" HorizontalAlignment="Left" />
<Label Name="lblFoto" Grid.Row="5" Grid.Column="1"
    Content="Foto" VerticalAlignment="Center"
    FontSize="12" HorizontalAlignment="Left" />
<Image Name="imgFoto" Grid.Row="5" Grid.Column="2"
    VerticalAlignment="Center" Width="120" Height="100" />
<DockPanel Grid.Row="6" Grid.Column="1" Grid.ColumnSpan="2"
    HorizontalAlignment="Center">
    <Button Name="btnPrimero" Content="&lt;&lt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al primer registro"/>
    <Button Name="btnAnterior" Content="&lt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al anterior registro"/>
    <TextBox Name="txtPosicion" Width="80" Height="25"
        IsReadOnly="True" HorizontalContentAlignment="Center"/>
```

```
<Button Name="btnSiguiente" Content="&gt;" Width="35"
      Height="25" Cursor="Hand" ToolTip="Ir al siguiente registro"/>
<Button Name="btnUltimo" Content="&gt;&gt;" Width="35"
      Height="25" Cursor="Hand" ToolTip="Ir al último registro"/>
</DockPanel>
</Grid>
</Window>
```

Nota: En el código anterior se define 9 filas y 4 columnas en el Grid para ubicar los controles: 3 textos para mostrar el código, apellido y nombre, un datepicker para la fecha y un image para la foto. Además hay 4 botones para realizar el desplazamiento de registro.

- El diseño de la ventana debe quedar como se muestra en el gráfico 6.17.

Gráfico 6.17: Diseño de la ventana Consulta de Empleados



- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.IO
```

```
Class ConsultaEmpleados
    Private lobeEmpleado As New List(Of beEmpleado)
```

Private N As Integer

```
Private Function ObtenerImagen(ByVal Codigo As Integer) As String
    Dim Ruta As String = _
        "C:\Data\DemosLibro\LibroVB2010\Imagenes\JPG\Empleados\"
    Dim Archivo As String = String.Format("{0}{1}.jpg", Ruta, Codigo)
    If Not File.Exists(Archivo) Then _
        Archivo = String.Format("{0}No.jpg", Ruta)
    Return (Archivo)
End Function
```

```
Private Sub MostrarEmpleado()
    txtCodigo.Text = lobeEmpleado(N).Codigo
    txtApellido.Text = lobeEmpleado(N).Apellido
    txtNombre.Text = lobeEmpleado(N).Nombre
    dpFechaNac.SelectedDate = lobeEmpleado(N).FechaNac
    imgFoto.Source = New BitmapImage _
        (New Uri(ObtenerImagen(lobeEmpleado(N).Codigo)))
    txtPosicion.Text = String.Format("{0} de {1}", N + 1, _
        lobeEmpleado.Count)
End Sub
```

```
Private Sub CargarEmpleados(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim obrEmpleado As New brEmpleado
    lobeEmpleado = obrEmpleado.Listar
    MostrarEmpleado()
End Sub
```

```
Private Sub IrAlPrimerRegistro(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles btnPrimero.Click
    N = 0
    MostrarEmpleado()
End Sub
```

```
Private Sub IrAnteriorRegistro(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles btnAnterior.Click
    If N > 0 Then N = N - 1
    MostrarEmpleado()
End Sub
```

```
Private Sub IrSiguienteRegistro(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnSiguiente.Click  
    If N < lobeEmpleado.Count - 1 Then N = N + 1  
    MostrarEmpleado()  
End Sub  
  
Private Sub IrUltimoRegistro(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnUltimo.Click  
    N = lobeEmpleado.Count - 1  
    MostrarEmpleado()  
End Sub  
End Class
```

Nota: En el código anterior al cargar la ventana se llena la lista de objetos y se llama a la subrutina `MostrarEmpleado` que presenta los datos de un registro incluyendo la foto para lo cual se usa un índice del registro que deseamos presentar, cada vez que usamos los botones de desplazamiento se cambia el índice al registro que queremos mostrar.

Importante: Para cargar la foto en el control `image` se asigna a la propiedad `Source` del `image` a un objeto `BitmapImage` el cual debe pasar la URI con la ruta del archivo de imagen, para esto se usa la función `Obtener Imagen` que permite asignar el archivo `No.jpg` a los códigos que no tengan su archivo correspondiente.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo `StartupUri`) a "ConsultaEmpleados.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave `conNW` en la sección `appSettings`:

```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
```

```
</appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.18: Ejecución de la ventana Consulta de Empleados

The screenshot shows a Windows application window titled "Consulta de Empleados - Leer y Escribir". The main area is titled "Consulta de Empleados" and displays the following information:

- Codigo: 1
- Apellido: Davolio
- Nombre: Nancy
- Fecha Nac: 08/12/1948
- Foto: A small portrait photo of Nancy Davolio.

At the bottom of the window, there are navigation buttons: "<<", "<", "1 de 26", ">", and ">>".

2.2. Usando Enlace de Datos

En este tema veremos como trabajar con enlace de datos entre los controles WPF y el origen de datos, el cual generalmente será listas de objetos en nuestro caso ya que las librerías de reglas de negocio devuelven listas de objetos a la aplicación.

Existen dos tipos de enlaces: **simple** y **complejo**; el enlace simple consiste en mostrar sobre el valor de una propiedad un dato enlazado por ejemplo el valor de un campo de una fila o una propiedad de un objeto; mientras que el enlace complejo podría mostrar varios valores de varias columnas de una tabla o propiedades de una lista de objetos.

Para implementar el enlace complejo en WPF los controles de varias filas o elementos como el ListBox, ComboBox, ListView, TreeView y DataGrid tienen la propiedad **ItemsSource** que indica cual es su origen de datos. Los controles ListBox y ComboBox tienen las propiedades DisplayMemberPath y SelectedValuePath donde se configura el nombre del campo que se va a mostrar y guardar respectivamente.

Para implementar el enlace simple en WPF se configura la propiedad **DataContext** del contenedor y luego en la propiedad respectiva de cada control, por ejemplo Text para un TextBox, Content para un Label, ImageSource para una imagen, etc, se realiza el enlace usando la clase Binding con el atributo Path asignando el nombre del campo o propiedad.

Una vez enlazados los controles para realizar desplazamientos se puede usar un intermediario de datos como el **CollectionView** que tiene métodos como MoveCurrentToFirst para ir al primer registro, MoveCurrentToPrevious para regresar al anterior, MoveCurrentToNext para ir al siguiente, MoveCurrentToLast para ir al último o MoveCurrentToPosition para ir a cualquier posición. Además de un apr de propiedades que verifican el desbordamiento: IsCurrentBeforeFirst y IsCurrentAfterLast.

Seguidamente presentaremos un par de ejemplos de cómo usar el enlace de datos, el primero trata del enlace complejo en un control ListBox que lista los nombres de los productos y el segundo muestra como implementar el enlace simple en una consulta de productos.

Demo 90

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo90" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Lista Productos.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ListaProductos"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Enlace de Datos - Lista de Productos" Height="350" Width="350"
  ResizeMode="NoResize" WindowStartupLocation="CenterScreen">
  <Grid>
    <ListBox Name="lstProducto" Foreground="Blue"
      DisplayMemberPath="Nombre" SelectedValuePath="Codigo">
      <ListBox.Background>
        <ImageBrush ImageSource="C:\Data\DemosLibro\LibroVB2010\
          Imagenes\JPG\Paisajes\MachuPicchu.jpg" Opacity="0.5"/>
      </ListBox.Background>
    </ListBox>
  </Grid>
</Window>
```

Nota: En el código anterior solo se incluye un control ListBox con un fondo de imagen que esta semi transparente mediante la propiedad opacity. Además se configura las propiedades DisplayMemberPath para indicar que se muestra el nombre del producto y SelectedValuePath para indicar que se guarda el código del producto.

- El diseño de la ventana debe quedar como se ve en el siguiente gráfico.

Gráfico 6.19: Diseño de la ventana Lista de Productos



- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Class ListaProductos
```

```
Private Sub ListarProductos(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded  
    Dim obrProducto As New brProducto  
    Dim lobeProducto As List(Of beProducto) = obrProducto.Listar  
    lstProducto.ItemsSource = lobeProducto  
End Sub
```

```
Private Sub MostrarCodigo(ByVal sender As System.Object, _  
ByVal e As System.Windows.Controls.SelectionChangedEventArgs) _  
Handles lstProducto.SelectionChanged  
    Me.Title = String.Format("El código es {0}", lstProducto.SelectedValue)  
End Sub
```

```
End Class
```

Nota: En el código anterior al cargar la ventana se llena la lista de objetos de productos y se enlaza al control ListBox. Además cada vez que se

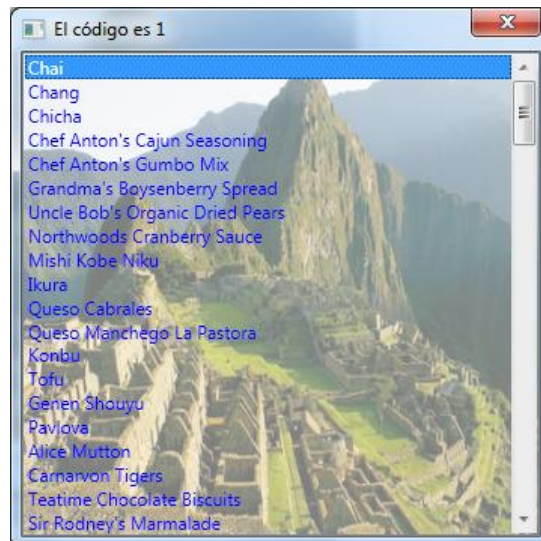
selecciona un producto de la lista se muestra en la barra de título de la ventana el código seleccionado del producto.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ListaProductos.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
      data source=Lduenas\MCTS;initial catalog=Northwind"/>  
  </appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.20: Ejecución de la ventana Lista de Productos



Demo 91

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo91" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Consulta Productos.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ConsultaProductos"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Enlace de Datos - Consulta de Productos" Height="350" Width="350"
    WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
    <Grid Name="grdProducto">
        <Grid.Background>
            <ImageBrush
                ImageSource="C:\Data\DemosLibro\LibroVB2010\Imagenes\JPG\Paisajes\MachuPicchu.jpg" Opacity="0.5"/>
        </Grid.Background>
        <Grid.RowDefinitions>
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="50" />
<ColumnDefinition Width="60" />
<ColumnDefinition Width="160" />
<ColumnDefinition Width="50*" />
</Grid.ColumnDefinitions>
<Label Name="lblTitulo" Grid.Row="1" Grid.Column="1"
    Grid.ColumnSpan="2" Content="Consulta de Productos"
    FontSize="16" Foreground="Blue"
    HorizontalAlignment="Center" />
<Label Name="lblCodigo" Grid.Row="2" Grid.Column="1"
    VerticalAlignment="Center" Content="Codigo"
    FontSize="12" Foreground="Blue"
    HorizontalAlignment="Left" />
<TextBox Name="txtCodigo" Grid.Row="2" Grid.Column="2"
    Width="50" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=Codigo}"/>
<Label Name="lblNombre" Grid.Row="3" Grid.Column="1"
    Content="Nombre" Foreground="Blue"
    VerticalAlignment="Center" FontSize="12"
    HorizontalAlignment="Left" />
<TextBox Name="txtNombre" Grid.Row="3" Grid.Column="2"
    Width="160" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=Nombre}"/>
<Label Name="lblPrecio" Grid.Row="4" Grid.Column="1"
    Content="Precio" Foreground="Blue"
    VerticalAlignment="Center" FontSize="12"
    HorizontalAlignment="Left" />
<TextBox Name="txtPrecio" Grid.Row="4" Grid.Column="2"
    Width="50" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=PrecioUnitario}"/>
<Label Name="lblStock" Grid.Row="5" Grid.Column="1"
    Content="Stock" Foreground="Blue"
    VerticalAlignment="Center" FontSize="12"
    HorizontalAlignment="Left" />
<TextBox Name="txtStock" Grid.Row="5" Grid.Column="2"
    Width="50" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=Stock}"/>
<DockPanel Grid.Row="6" Grid.Column="1" Grid.ColumnSpan="2"
    HorizontalAlignment="Center">
    <Button Name="btnPrimero" Content="&lt;&lt;" Width="35"
```

```

        Height="25" Cursor="Hand" ToolTip="Ir al primer registro"/>
<Button Name="btnAnterior" Content="&lt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al anterior registro"/>
<TextBox Name="txtPosicion" Width="80" Height="25"
        IsReadOnly="True" HorizontalContentAlignment="Center"/>
<Button Name="btnSiguiente" Content="&gt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al siguiente registro"/>
<Button Name="btnUltimo" Content="&gt;&gt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al último registro"/>
</DockPanel>
<Button Name="btnBuscar" Grid.Row="7" Grid.Column="1"
        Grid.ColumnSpan="2" Content="Buscar x Código" Width="110"
        Height="25" Cursor="Hand" ToolTip="Ir al último registro"/>
</Grid>
</Window>

```

Nota: En el código anterior se define 9 filas y 4 columnas en el Grid para ubicar los controles: 4 textos para mostrar el código, nombre, precio y stock de los productos. Además hay 4 botones para realizar el desplazamiento de registro y un botón para buscar por código.

- El diseño de la ventana debe quedar como se ve en el siguiente gráfico.

Gráfico 6.21: Diseño de la ventana Consulta de Productos



- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
Imports System.IO
```

```
Class ConsultaProductos
```

```
    Private lobeProducto As New List(Of beProducto)
    Private WithEvents cv As CollectionView
    Private nCodigo As Integer
```

```
    Private Sub CargarProductos(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
        Dim obrProducto As New brProducto
        lobeProducto = obrProducto.Listar
        grdProducto.DataContext = lobeProducto
        cv = CollectionViewSource.GetDefaultView(lobeProducto)
        MostrarPosicion(Nothing, Nothing)
    End Sub
```

```
    Private Sub MostrarPosicion(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles cv.CurrentChanged
        txtPosicion.Text = String.Format("{0} de {1}", _
        cv.CurrentPosition + 1, cv.Count)
    End Sub
```

```
    Private Sub IrAlPrimerRegistro(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles btnPrimero.Click
        cv.MoveCurrentToFirst()
    End Sub
```

```
    Private Sub IrAnteriorRegistro(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles btnAnterior.Click
        cv.MoveCurrentToPrevious()
        If cv.IsCurrentBeforeFirst Then cv.MoveCurrentToFirst()
    End Sub
```

```
    Private Sub IrSiguienteRegistro(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles btnSiguiente.Click
        cv.MoveCurrentToNext()
        If cv.IsCurrentAfterLast Then cv.MoveCurrentToLast()
```


End Sub

```
Private Sub IrUltimoRegistro(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnUltimo.Click  
    cv.MoveCurrentToLast()  
End Sub
```

```
Private Function BuscarProducto(ByVal obeProducto As beProducto) _  
As Boolean  
    Return (obeProducto.Codigo = nCodigo)  
End Function
```

```
Private Sub BuscarPorCodigo(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnBuscar.Click  
    Dim strCodigo As String = InputBox _  
    ("Ingresa el código", "Búsqueda de Productos", "1")  
    If strCodigo <> "" Then  
        Dim exito As Boolean = Integer.TryParse(strCodigo, nCodigo)  
        If exito Then  
            Dim pred As New Predicate(Of beProducto) _  
            (AddressOf BuscarProducto)  
            Dim pos As Integer = lobeProducto.FindIndex(pred)  
            If pos > -1 Then  
                cv.MoveCurrentToPosition(pos)  
            Else  
                MessageBox.Show("No existe el código ingresado", "Aviso")  
            End If  
        Else  
            MessageBox.Show("El código debe ser entero", "Aviso")  
        End If  
    Else  
        MessageBox.Show("El código esta vacío", "Aviso")  
    End If  
End Sub  
End Class
```

Nota: En el código anterior al cargar la ventana se llena la lista de productos y se enlaza la propiedad DataContext del Grid a la lista de objetos, luego se crea el CollectionView usando el método GetDefaultView de la clase CollectionViewSource.

Importante: Para mostrar la posición actual se usa el evento Current Changed del CollectionView y para buscar por código se usa predicados.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ConsultaProductos.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.22: Ejecución de la ventana Consulta de Productos



2.3. Usando Conversión de Datos

Cuando enlazamos controles a datos los valores que se muestran son exactamente como se encuentran en el origen de datos; pero la mayoría de veces, queremos que estos valores se muestren en un formato personalizado por ejemplo los números con 2 decimales, los datos de tipo fecha y hora solo debe mostrarse la fecha, o en vez de ver un texto ver una imagen, etc.

Todo esto se logra en WPF usando convertidores de datos personalizados para lo cual debemos crear una clase que implemente la interface `IValueConverter` ubicada en el espacio de nombres `System.Windows.Data`, a continuación, programar los métodos `Convert` y `ConvertBack`. Los convertidores pueden cambiar los datos de un tipo a otro, traducir datos basados en información de referencia cultural o modificar otros aspectos de la presentación.

Los convertidores de valores tienen en cuenta la referencia cultural. Los métodos `Convert` y `ConvertBack` tienen un parámetro `culture` que indica la información de referencia cultural. Si la información de referencia cultural es no importante para la conversión, puede pasar por alto este parámetro en el convertidor personalizado.

Los métodos `Convert` y `ConvertBack` también cuentan con un parámetro denominado `parameter` para que pueda usar la misma instancia del convertidor con parámetros distintos. Por ejemplo, puede escribir a un convertidor de formato que genere formatos de datos diferentes basados en el parámetro de entrada que use. Puede utilizar la propiedad `ConverterParameter` de la clase `Binding` para pasar un parámetro como argumento a los métodos `Convert` y `ConvertBack`.

A continuación veremos un par de ejemplos sobre como usar conversión de datos en el enlace de datos, el primer ejemplo muestra una lista de productos con la columna precio unitario con 2 decimales usando una clase de conversión llamada `FormatoDecimal`, además las columnas del `ListView` pueden ordenarse en forma ascendente o descendente por cualquiera de los campos mostrados: código, nombre, precio unitario o stock.

En el segundo ejemplo se muestra una consulta de empleados que muestra su código, apellido, nombre, fecha de nacimiento y foto del empleado, aquí se crean 2 clases de conversión de datos: una para mostrar solo la fecha de nacimiento sin hora y el segundo para mostrar la ruta de la imagen de la foto del empleado según su código.

Demo 92

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo92" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Crear una clase para la conversión a 2 decimales: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoDecimal.vb" y clic en "Add".
- Escribir el siguiente código en la clase:

```
Public Class FormatoDecimal
    Implements IValueConverter

    Public Function Convert(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.Convert
        Return (String.Format("{0:n2}", value))
    End Function

    Public Function ConvertBack(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.ConvertBack
        Return (String.Format("{0:n2}", value))
    End Function
End Class
```

End Class

Nota: En el código anterior ambos métodos reciben un valor de un cierto tipo por ejemplo decimal y se devuelve una cadena formateada a 2 decimales.

- Cerrar la ventana con la clase y compilar la aplicación para poder definir un recurso que apunte a la clase creada.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Lista Productos.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ListaProductos"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:ld="clr-namespace:Demo92"
    Title="Conversión de Datos - Ordenación en ListView"
    Height="350" Width="400"
    Name="ListaProductos" ResizeMode="NoResize"
    WindowStartupLocation="CenterScreen">
    <Window.Resources>
        <ld:FormatoDecimal x:Key="fd"/>
    </Window.Resources>
    <Grid>
        <ListView Name="lvwProducto" Foreground="White"
            GridViewColumnHeader.Click="Ordenar" >
            <ListView.Background>
                <LinearGradientBrush>
                    <GradientStop Color="aqua" Offset="0.8"/>
                    <GradientStop Color="blue" Offset="0.3"/>
                </LinearGradientBrush>
            </ListView.Background>
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Codigo" Width="60"
                        DisplayMemberBinding="{Binding Path=Codigo}" />
                    <GridViewColumn Header="Nombre" Width="180"
                        DisplayMemberBinding="{Binding Path=Nombre}" />
                    <GridViewColumn Header="PrecioUnitario" Width="60"
```

```
        DisplayMemberBinding="{Binding Path=PrecioUnitario,  
        Converter={StaticResource fd}}"/>  
        <GridViewColumn Header="Stock" Width="60"  
        DisplayMemberBinding="{Binding Path=Stock}"/>  
    </GridView>  
    </ListView.View>  
    </ListView>  
    </Grid>  
    </Window>
```

Nota: En el código anterior se define un espacio de nombres en XAML con el prefijo "ld" (siglas del autor: Luis Dueñas) que apunta al Demo92, luego se incluye un recurso en la ventana llamado "fd" que apunta a la clase FormatoDecimal y por último se aplica este recurso mediante el atributo Converter en el enlace de datos del precio unitario.

Importante: Para ordenar los datos al dar clic en las cabeceras de las columnas del ListView se asocia el procedimiento "Ordenar" al evento clic del objeto "GridViewColumnHeader" del ListView.

- El diseño de la ventana debe quedar como se ve en el siguiente gráfico.

Gráfico 6.23: Diseño de la ventana Lista de Productos



- Crear una clase genérica para ordenar listas de objetos (en nuestro caso lista de productos): del menú "Project", seleccionar "Add Class", escribir como nombre: "ucCompara.vb" y clic en "Add".
- Escribir el siguiente código en la clase:

```
Imports System.Collections.Generic
```

```
Public Class ucCompara(Of T)  
    Implements IComparer(Of T)  
    Private Campo As String  
    Private TipoOrden As TiposOrden
```

```
    Public Enum TiposOrden  
        Ascendente = 0  
        Descendente = 1  
    End Enum
```

```
    Public Sub New(ByVal vCampo As String, ByVal vTipoOrden As TiposOrden)  
        Campo = vCampo  
        TipoOrden = vTipoOrden  
    End Sub
```

```
    Public Function Compare(ByVal x As T, ByVal y As T) As _  
        Integer Implements IComparer(Of T).Compare  
        Dim valX As Object = _  
            x.GetType.GetProperty(Campo).GetValue(x, Nothing)  
        Dim valY As Object = _  
            y.GetType.GetProperty(Campo).GetValue(y, Nothing)  
        If TipoOrden = TiposOrden.Ascendente Then  
            Return (valX.CompareTo(valY))  
        Else  
            Return (valY.CompareTo(valX))  
        End If  
    End Function  
End Class
```

Nota: En el código anterior en el constructor se pasa como parámetro el campo que queremos ordenar y el tipo de orden: 0 es ascendente y 1 es descendente, luego usando reflection en el método Compare se devuelve que objeto es mayor o menor según sea el tipo de orden.

- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Class ListaProductos
```

```
    Private lobeProducto As New List(Of beProducto)
```

```
    Private Sub ListarProductos(ByVal sender As System.Object, _
```

```
    ByVal e As System.Windows.RoutedEventArgs) _
```

```
    Handles MyBase.Loaded
```

```
        Dim obrProducto As New brProducto
```

```
        lobeProducto = obrProducto.Listar()
```

```
        lvwProducto.ItemsSource = lobeProducto
```

```
    End Sub
```

```
    Private Sub Ordenar(ByVal sender As System.Object, _
```

```
    ByVal e As System.Windows.RoutedEventArgs)
```

```
        Dim columna As GridViewColumnHeader = _
```

```
        CType(e.OriginalSource, GridViewColumnHeader)
```

```
        Dim campo As String = columna.Column.Header
```

```
        Dim N As Integer = If(columna.Tag = 0, 1, 0)
```

```
        columna.Background = _
```

```
        If(N = 0, Brushes.Yellow, Brushes.White)
```

```
        columna.Foreground = _
```

```
        If(N = 0, Brushes.Red, Brushes.Blue)
```

```
        columna.Tag = N
```

```
        Dim oucCompara As New _
```

```
        ucCompara(Of beProducto)(campo, N)
```

```
        lobeProducto.Sort(oucCompara)
```

```
        lvwProducto.ItemsSource = Nothing
```

```
        lvwProducto.ItemsSource = lobeProducto
```

```
    End Sub
```

```
End Class
```

Nota: En el código anterior al cargar la ventana se llena la lista de productos y se enlaza la propiedad `ItemsSource` del `ListView` a la lista de objetos, luego cada vez que se da clic en las cabeceras del `ListView` se ordenan los datos y se cambia de color de fondo y texto la cabecera.

Importante: Para obtener la cabecera a la cual se dio clic se usa el parámetro `e.OriginalSource` que devuelve el `GridViewColumnHeader`.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo `StartupUri`) a "ListaProductos.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave `conNW` en la sección `appSettings`:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.24: Ejecución de la ventana Lista de Productos



Codigo	Nombre	PrecioUnit	Stock
1	Chai	10.00	81
2	Chang	3.00	24
3	Chicha	10.00	39
4	Chef Anton's Cajun Seasoning	22.00	80
5	Chef Anton's Gumbo Mix	21.35	31
6	Grandma's Boysenberry Spread	25.00	50
7	Uncle Bob's Organic Dried Pear:	30.00	98
8	Northwoods Cranberry Sauce	40.00	99
9	Mishi Kobe Niku	97.00	96
10	Ikura	31.00	93
11	Queso Cabrales	21.00	97
12	Queso Manchego La Pastora	38.00	100
13	Konbu	6.00	100
14	Tofu	23.25	99

Demo 93

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo93" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Agregar una clase para la conversión a solo fecha: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoFecha.vb" y clic en "Add".
- Escribir el siguiente código en la clase FormatoFecha:

```
Public Class FormatoFecha
    Implements IValueConverter

    Public Function Convert(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.Convert
        Return (String.Format("{0:d}", value))
    End Function

    Public Function ConvertBack(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.ConvertBack
        Return (String.Format("{0:d}", value))
    End Function
End Class
```

Nota: En el código anterior los métodos devuelven una fecha como cadena sin hora.

- Agregar otra clase para la conversión a una ruta de archivos de imagen: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoImagen.vb" y clic en "Add".
- Escribir el siguiente código en la clase FormatoImagen:

```
Imports System.IO
```

```
Public Class FormatoImagen
```

```
    Implements IValueConverter
```

```
    Private Ruta As String = _
```

```
        "C:\Data\DemosLibro\LibroVB2010\Imagenes\JPG\Empleados\"
```

```
    Private Archivo As String
```

```
    Public Function Convert(ByVal value As Object, _
```

```
        ByVal targetType As System.Type, ByVal parameter As Object, _
```

```
        ByVal culture As System.Globalization.CultureInfo) As Object _
```

```
    Implements System.Windows.Data.IValueConverter.Convert
```

```
        Archivo = String.Format("{0}{1}.jpg", Ruta, value)
```

```
        If File.Exists(Archivo) Then
```

```
            Return (Archivo)
```

```
        Else
```

```
            Return (String.Format("{0}No.jpg", Ruta))
```

```
        End If
```

```
    End Function
```

```
    Public Function ConvertBack(ByVal value As Object, _
```

```
        ByVal targetType As System.Type, ByVal parameter As Object, _
```

```
        ByVal culture As System.Globalization.CultureInfo) As Object _
```

```
    Implements System.Windows.Data.IValueConverter.ConvertBack
```

```
        Archivo = String.Format("{0}{1}.jpg", Ruta, value)
```

```
        If File.Exists(Archivo) Then
```

```
            Return (Archivo)
```

```
        Else
```

```
            Return (String.Format("{0}No.jpg", Ruta))
```

```
        End If
```

```
    End Function
```

```
End Class
```

Nota: En el código anterior los métodos devuelven la ruta de un archivo de imagen según el valor que debería ser el código del empleado, sino existe el archivo se devuelve el archivo de imagen No.jpg.

- Cerrar las ventanas con las 2 clases y compilar la aplicación para poder definir un recurso que apunte a las clases creadas.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Consulta Empleados.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ConsultaEmpleados"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:ld="clr-namespace:Demo93"
    Title="Conversión de Datos - Fechas e Imágenes"
    Height="350" Width="350" ResizeMode="NoResize"
    WindowStartupLocation="CenterScreen">
    <Window.Resources>
        <ld:FormatoFecha x:Key="ff" />
        <ld:FormatoImagen x:Key="fi" />
    </Window.Resources>
    <Grid Name="grdEmpleado">
        <Grid.Background>
            <LinearGradientBrush>
                <GradientStop Color="aqua" Offset="0.8"/>
                <GradientStop Color="blue" Offset="0.3"/>
            </LinearGradientBrush>
        </Grid.Background>
        <Grid.RowDefinitions>
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="100" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30" />
            <RowDefinition Height="30*" />
        </Grid.RowDefinitions>
```

```
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="50" />
    <ColumnDefinition Width="100" />
    <ColumnDefinition Width="120" />
    <ColumnDefinition Width="50*" />
</Grid.ColumnDefinitions>
<Label Name="lblTitulo" Grid.Row="0" Grid.Column="1"
    Grid.ColumnSpan="2" Content="Consulta de Empleados"
    FontSize="16" Foreground="White"
    HorizontalAlignment="Center" />
<Label Name="lblCodigo" Grid.Row="1" Grid.Column="1"
    VerticalAlignment="Center" Content="Codigo"
    FontSize="12" Foreground="White"
    HorizontalAlignment="Left" />
<TextBox Name="txtCodigo" Grid.Row="1" Grid.Column="2"
    Width="50" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=Codigo}"/>
<Label Name="lblApellido" Grid.Row="2" Grid.Column="1"
    Content="Apellido" Foreground="White"
    VerticalAlignment="Center" FontSize="12"
    HorizontalAlignment="Left" />
<TextBox Name="txtApellido" Grid.Row="2" Grid.Column="2"
    Width="100" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=Apellido}"/>
<Label Name="lblNombre" Grid.Row="3" Grid.Column="1"
    Content="Nombre" Foreground="White"
    VerticalAlignment="Center" FontSize="12"
    HorizontalAlignment="Left" />
<TextBox Name="txtNombre" Grid.Row="3" Grid.Column="2"
    Width="100" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=Nombre}"/>
<Label Name="lblFechaNac" Grid.Row="4" Grid.Column="1"
    Content="Fecha Nac" VerticalAlignment="Center"
    FontSize="12" Foreground="White"
    HorizontalAlignment="Left" />
<TextBox Name="txtFechaNac" Grid.Row="4" Grid.Column="2"
    Width="100" Height="20" HorizontalAlignment="Left"
    Text="{Binding Path=FechaNac,Converter={StaticResource ff}}"/>
<Label Name="lblFoto" Grid.Row="5" Grid.Column="1"
    Content="Foto" Foreground="White"
```

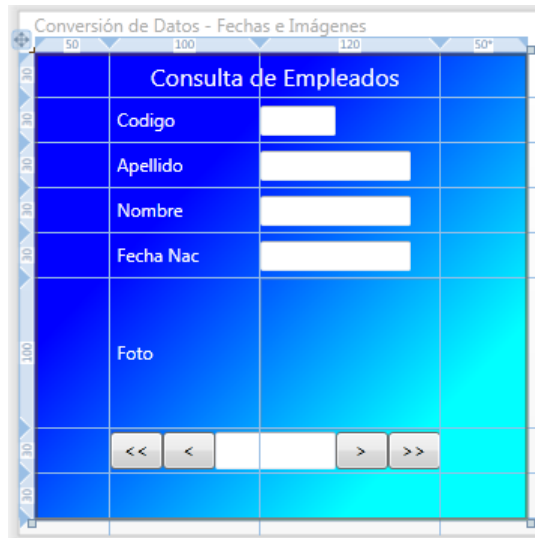
```
        VerticalAlignment="Center" FontSize="12"
        HorizontalAlignment="Left" />
<Image Name="imgFoto" Grid.Row="5" Grid.Column="2"
        VerticalAlignment="Center" Width="120" Height="100"
        Source="{Binding Path=Codigo,Converter={StaticResource fi}}"/>
<DockPanel Grid.Row="6" Grid.Column="1" Grid.ColumnSpan="2"
        HorizontalAlignment="Center">
    <Button Name="btnPrimero" Content="&lt;&lt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al primer registro"/>
    <Button Name="btnAnterior" Content="&lt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al anterior registro"/>
    <TextBox Name="txtPosicion" Width="80" Height="25"
        IsReadOnly="True" HorizontalContentAlignment="Center"/>
    <Button Name="btnSiguiente" Content="&gt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al siguiente registro"/>
    <Button Name="btnUltimo" Content="&gt;&gt;" Width="35"
        Height="25" Cursor="Hand" ToolTip="Ir al último registro"/>
</DockPanel>
</Grid>
</Window>
```

Nota: En el código anterior se define un espacio de nombres en XAML con el prefijo "ld" (siglas del autor: Luis Dueñas) que apunta al Demo93, luego se incluye 2 recursos en la ventana: uno llamado "ff" que apunta a la clase FormatoFecha y otro llamado "fi" que apunta a la clase FormatoImagen.

Por último se aplican estos recursos mediante el atributo Converter en el enlace de datos, "fff" a la fecha de nacimiento y "fi" a la imagen.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.25.

Gráfico 6.25: Diseño de la ventana Consulta de Empleados



- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Class ConsultaEmpleados
```

```
    Private lobeEmpleado As New List(Of beEmpleado)
```

```
    Private WithEvents cv As CollectionView
```

```
    Private Sub CargarEmpleados(ByVal sender As System.Object, _  
        ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
```

```
        Dim obrEmpleado As New brEmpleado
```

```
        lobeEmpleado = obrEmpleado.Listar
```

```
        grdEmpleado.DataContext = lobeEmpleado
```

```
        cv = CollectionViewSource.GetDefaultView(lobeEmpleado)
```

```
        MostrarPosicion(Nothing, Nothing)
```

```
    End Sub
```

```
    Private Sub MostrarPosicion(ByVal sender As Object, _
```

```
        ByVal e As System.EventArgs) Handles cv.CurrentChanged
```

```
        txtPosicion.Text = String.Format("{0} de {1}", _
```

```
            cv.CurrentPosition + 1, cv.Count)
```

```
    End Sub
```

```
Private Sub IrAlPrimerRegistro(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnPrimero.Click  
    cv.MoveCurrentToFirst()  
End Sub
```

```
Private Sub IrAnteriorRegistro(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnAnterior.Click  
    cv.MoveCurrentToPrevious()  
    If cv.IsCurrentBeforeFirst Then cv.MoveCurrentToFirst()  
End Sub
```

```
Private Sub IrSiguienteRegistro(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnSiguiente.Click  
    cv.MoveCurrentToNext()  
    If cv.IsCurrentAfterLast Then cv.MoveCurrentToLast()  
End Sub
```

```
Private Sub IrUltimoRegistro(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles btnUltimo.Click  
    cv.MoveCurrentToLast()  
End Sub  
End Class
```

Nota: En el código anterior al cargar la ventana se llena la lista de empleados y se enlaza la propiedad DataContext del Grid a la lista de objetos y se crea el CollectionView mediante el método GetDefaultView de la clase CollectionViewSource. Para el desplazamiento se usa los métodos del objeto CollectionView y también para mostrar la posición actual se programa en el evento CurrentChanged.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ConsultaEmpleados.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:


```
<configuration>  
  <appSettings>  
    <add key="conNW" value="uid=sa;pwd=123456;  
      data source=Lduenas\MCTS;initial catalog=Northwind"/>  
  </appSettings>  
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.26: Ejecución de la ventana Consulta de Empleados



2.4. Usando Plantillas de Datos

Las plantillas de datos se usan para mejorar la presentación de datos en los controles enlazados a datos, por ejemplo los controles de listas `ListBox` y `ComboBox` solo muestran una columna cuando están enlazados a un origen de datos mediante la propiedad `DisplayMemberPath`, en cambio si usamos plantillas de datos podrían mostrar varias columnas con cualquier tipo de dato.

Para implementar plantillas de datos en controles WPF se debe crear un objeto de tipo `DataTemplate` y asociarlo a la propiedad `ItemTemplate` del control. Dentro del `DataTemplate` para definir varios controles es necesario incluir un contenedor como el `Grid`, `StackPanel`, `DockPanel`, etc.

También hay controles como el `TreeView` que soporta plantilla jerárquica, la cual se implementa con el objeto `HierarchicalDataTemplate`, para definir a los detalles se usa su propiedad `ItemTemplate`.

Además cuando se trabaja con plantillas jerárquicas se debe tener un origen de datos jerárquico es decir por ejemplo una lista de objetos que a su vez tenga como propiedad otra lista de objetos de tal forma que se enlace solo el objeto raíz y desde aquí se acceda a las demás listas.

Un origen de datos jerárquico también facilita la sincronización entre listas en vez de estar programando podría ser automatico los filtros usando la propiedad `IsSynchronizedWithCurrentItem` en `true` y configurando el objeto `CurrentItem` en el enlace de datos como lo demuestra el siguiente demo.

Para obtener más información sobre plantillas de datos ver la referencia 41 al final del libro.

A continuación dos ejemplos de plantillas de datos, el primero es una consulta de productos por categoría usando sincronización de controles y el segundo es un ejemplo de plantillas jerárquicas que muestra las órdenes con sus respectivos detalles. Para ambos ejercicios es necesario crear nuevos procedimientos almacenados y agregar métodos a las librerías de clases creadas, tanto las entidades de negocios, capa de datos y reglas de negocio.

Creando los Procedimientos Almacenados

Crear los siguientes procedimientos almacenados:

1. Procedimiento almacenado que lista ordenes

```
Create Procedure usp_Orders_Sel
As
Select
    o.OrderID As Orden,
    c.CompanyName As Cliente,
    e.LastName+' '+e.FirstName As Empleado,
    o.OrderDate As FechaOrden
From Orders o,Customers c,Employees e
Where
    o.CustomerID=c.CustomerID And
    o.EmployeeID=e.EmployeeID
Order By 1
```

2. Procedimiento almacenado que lista detalles de ordenes

```
Create Procedure usp_OrderDetails_Sel
As
Select
    d.OrderID As IdOrden,
    p.ProductID As IdProducto,
    p.ProductName As DesProducto,
    d.UnitPrice As PrecioUnitario,
    d.Quantity As Cantidad,
    d.UnitPrice*d.Quantity As PrecioTotal
From [Order Details] d,Products p
Where
    d.ProductID=p.ProductID
Order By 1
```

Modificando la Librería de Entidades del Negocio

- Del menú "File", seleccionar "Open Project".
- Ingresar a la carpeta donde se encuentra la librería de entidades del negocio: Northwind.LibBusinessEntities y abrir el proyecto.
- Abrir la clase beCategoria y modificar el código como sigue:

```
Public Class beCategoria
    Public PropertyCodigo() As Integer
    Public PropertyNombre() As String
    Public PropertyListaProductos As List(Of beProducto)
End Class
```

- Agregar una nueva clase entidad para las ordenes: del menú "Project", seleccionar "Add Class", ingresar como nombre: "beOrden.vb" y "Add".
- Escribir el siguiente código para la clase:

```
Public Class beOrden
    Public PropertyIdOrden() As Integer
    Public PropertyNombreCliente() As String
    Public PropertyNombreEmpleado() As String
    Public PropertyFechaOrden() As DateTime
    Public PropertyListaDetalles As List(Of beDetalle)
End Class
```

- Agregar una nueva clase entidad para los detalles: del menú "Project", seleccionar "Add Class", ingresar el nombre: "beDetalle.vb" y clic "Add".
- Escribir el siguiente código para la clase:

```
Public Class beDetalle
    Public PropertyIdOrden() As Integer
    Public PropertyIdProducto() As Integer
    Public PropertyNombreProducto() As String
    Public PropertyPrecioUnitario() As Decimal
    Public PropertyCantidad() As Short
    Public PropertyPrecioTotal() As Decimal
End Class
```

- Compilar la librería de entidades y cerrar el proyecto.

Modificando la Librería de Acceso a Datos

- Del menú "File", seleccionar "Open Project".
- Ingresar a la carpeta donde se encuentra la librería de acceso a datos: Northwind.LibDataAccess y abrir el proyecto.
- Agregar una nueva clase de datos para las ordenes: del menú "Project", seleccionar "Add Class", ingresar como nombre: "daOrden.vb" y "Add".
- Escribir el siguiente código para la clase:

```
Imports Northwind.LibBusinessEntities
Imports System.Data.SqlClient
```

```
Public Class daOrden
```

```
    Public Function fListar(ByVal con As SqlConnection) As List(Of beOrden)
        Dim lobeOrden As New List(Of beOrden)
```

```
        Dim cmd As New SqlCommand("usp_Orders_Sel", con)
        cmd.CommandType = CommandType.StoredProcedure
        cmd.CommandTimeout = 60
```

```
        Dim drd As SqlDataReader = _
        cmd.ExecuteReader(CommandBehavior.SingleResult)
```

```
        If drd IsNot Nothing Then
```

```
            Dim posOrden As Integer = drd.GetOrdinal("Orden")
            Dim posCliente As Integer = drd.GetOrdinal("Cliente")
            Dim posEmpleado As Integer = drd.GetOrdinal("Empleado")
            Dim posFechaOrden As Integer = drd.GetOrdinal("FechaOrden")
            Dim obeOrden As beOrden
```

```
            Do While drd.Read
```

```
                obeOrden = New beOrden
```

```
                With obeOrden
```

```
                    .IdOrden = drd.GetInt32(posOrden)
```

```
                    .NombreCliente = drd.GetString(posCliente)
```

```
                    .NombreEmpleado = drd.GetString(posEmpleado)
```

```
                    .FechaOrden = drd.GetDateTime(posFechaOrden)
```

```
                End With
```

```
        lobeOrden.Add(obeOrden)
    Loop
    drd.Close()
End If

Return (lobeOrden)
End Function
End Class
```

- Agregar una nueva clase de datos para los detalles: del menú "Project", seleccionar "Add Class", ingresar como nombre: "daDetalle.vb" y "Add".
- Escribir el siguiente código para la clase:

```
Imports Northwind.LibBusinessEntities
Imports System.Data.SqlClient

Public Class daDetalle
    Public Function fListar(ByVal con As SqlConnection) As List(Of beDetalle)
        Dim lobeDetalle As New List(Of beDetalle)

        Dim cmd As New SqlCommand("usp_OrderDetails_Sel", con)
        cmd.CommandType = CommandType.StoredProcedure
        cmd.CommandTimeout = 60
        Dim drd As SqlDataReader = _
            cmd.ExecuteReader(CommandBehavior.SingleResult)
        If drd IsNot Nothing Then
            Dim posIdOrden As Integer = drd.GetOrdinal("IdOrden")
            Dim posIdProducto As Integer = drd.GetOrdinal("IdProducto")
            Dim posDesProducto As Integer = drd.GetOrdinal("DesProducto")
            Dim posPrecioUnitario As Integer = drd.GetOrdinal("PrecioUnitario")
            Dim posCantidad As Integer = drd.GetOrdinal("Cantidad")
            Dim posPrecioTotal As Integer = drd.GetOrdinal("PrecioTotal")
            Dim obeDetalle As beDetalle
            Do While drd.Read
                obeDetalle = New beDetalle
                With obeDetalle
                    .IdOrden = drd.GetInt32(posIdOrden)
                    .IdProducto = drd.GetInt32(posIdProducto)
                    .NombreProducto = drd.GetString(posDesProducto)
                    .PrecioUnitario = drd.GetDecimal(posPrecioUnitario)
```

```
        .Cantidad = drd.GetInt16(posCantidad)
        .PrecioTotal = drd.GetDecimal(posPrecioTotal)
    End With
    lobeDetalle.Add(obeDetalle)
Loop
drd.Close()
End If

Return (lobeDetalle)
End Function
End Class
```

- Compilar la librería de acceso a datos y cerrar el proyecto.

Modificando la Librería de Reglas de Negocio

- Del menú "File", seleccionar "Open Project".
- Ingresar a la carpeta donde se encuentra la librería de reglas de negocio: Northwind.LibBusinessRules y abrir el proyecto.
- Abrir la clase brCategoria y modificar el código como sigue:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibDataAccess
Imports System.Data.SqlClient

Public Class brCategoria
    Private strConexion As String
    Private IdCategoria As String

    Public Sub New()
        Dim asr As New System.Configuration.AppSettingsReader
        strConexion = asr.GetValue("conNW", Type.GetType("System.String"))
    End Sub

    Public Function Listar() As List(Of beCategoria)
        Dim lobeCategoria As New List(Of beCategoria)
        Using con As New SqlConnection(strConexion)
```

```
Try
    con.Open()
    Dim odaCategoria As New daCategoria
    lobeCategoria = odaCategoria.fListar(con)
Catch ex As Exception
    'Grabar el Log de error
    lobeCategoria = Nothing
End Try
End Using
Return (lobeCategoria)
End Function

Private Function BuscarProductos(ByVal obeProducto As beProducto) _
As Boolean
    Return (obeProducto.IdCategoria = IdCategoria)
End Function

Public Function ListarConProductos() As List(Of beCategoria)
    Dim lobeCategoria As New List(Of beCategoria)
    Using con As New SqlConnection(strConexion)
        Try
            con.Open()
            Dim odaCategoria As New daCategoria
            lobeCategoria = odaCategoria.fListar(con)
            Dim odaProducto As New daProducto
            Dim lobeProducto As List(Of beProducto) = _
            odaProducto.fListar(con)
            Dim pred As New Predicate(Of beProducto) _
            (AddressOf BuscarProductos)
            For I = 0 To lobeCategoria.Count - 1
                IdCategoria = lobeCategoria(I).Codigo
                lobeCategoria(I).ListaProductos = _
                lobeProducto.FindAll(pred)
            Next
        Catch ex As Exception
            'Grabar el Log de error
            lobeCategoria = Nothing
        End Try
    End Using
    Return (lobeCategoria)
```


End Function
End Class

Nota: En el código anterior el método ListarConProductos se conecta a la base de datos y ejecuta 2 consultas, una que trae todas las categorías y otra que trae todos los productos, luego se recorre cada categoría y usando un predicado se llena los productos de cada categoría.

- Agregar una nueva clase de datos para las ordenes: del menú "Project", seleccionar "Add Class", ingresar como nombre: "brOrden.vb" y "Add".
- Escribir el siguiente código para la clase:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibDataAccess
Imports System.Data.SqlClient

Public Class brOrden
    Private strConexion As String
    Private IdOrden As Integer

    Public Sub New()
        Dim asr As New System.Configuration.AppSettingsReader
        strConexion = asr.GetValue("conNW", Type.GetType("System.String"))
    End Sub

    Private Function BuscarDetalles(ByVal obeDetalle As beDetalle) _
        As Boolean
        Return (obeDetalle.IdOrden = IdOrden)
    End Function

    Public Function ListarConDetalles() As List(Of beOrden)
        Dim lobeOrden As New List(Of beOrden)
        Using con As New SqlConnection(strConexion)
            Try
                con.Open()
                Dim odaOrden As New daOrden
                lobeOrden = odaOrden.fListar(con)
                Dim odaDetalle As New daDetalle
                Dim lobeDetalle As List(Of beDetalle) = _
                    odaDetalle.fListar(con)
```

```
Dim pred As New Predicate(Of beDetalle) _  
    (AddressOf BuscarDetalles)  
For I = 0 To lobeOrden.Count - 1  
    IdOrden = lobeOrden(I).IdOrden  
    lobeOrden(I).ListaDetalles = _  
        lobeDetalle.FindAll(pred)  
Next  
Catch ex As Exception  
    'Grabar el Log de error  
    lobeOrden = Nothing  
End Try  
End Using  
Return (lobeOrden)  
End Function  
End Class
```

Nota: En el código anterior el método ListarConDetalles se conecta a la base de datos y ejecuta 2 consultas, una que trae todas las órdenes y otra que trae todos los detalles, luego se recorre cada orden y usando un predicado se llena los detalles de cada orden.

- Compilar la librería de reglas de negocio y cerrar el proyecto.

Demo 94

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo94" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Crear una clase para la conversión a 2 decimales: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoDecimal.vb" y clic en "Add".
- Escribir el siguiente código en la clase:

```
Public Class FormatoDecimal
    Implements IValueConverter

    Public Function Convert(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.Convert
        Return (String.Format("{0:n2}", value))
    End Function

    Public Function ConvertBack(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.ConvertBack
        Return (String.Format("{0:n2}", value))
    End Function
End Class
```

- Agregar otra clase para la conversión a una ruta de archivos de imagen: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoImagen.vb" y clic en "Add".

- Escribir el siguiente código en la clase FormatoImagen:

```
Imports System.IO
```

```
Public Class FormatoImagen
```

```
    Implements IValueConverter
```

```
    Private Ruta As String = _
```

```
        "C:\Data\Demolibro\LibroVB2010\Imagenes\JPG\Categorias\"
```

```
    Private Archivo As String
```

```
    Public Function Convert(ByVal value As Object, _
```

```
        ByVal targetType As System.Type, ByVal parameter As Object, _
```

```
        ByVal culture As System.Globalization.CultureInfo) As Object _
```

```
    Implements System.Windows.Data.IValueConverter.Convert
```

```
        Archivo = String.Format("{0}{1}.jpg", Ruta, value)
```

```
        If File.Exists(Archivo) Then
```

```
            Return (Archivo)
```

```
        Else
```

```
            Return (String.Format("{0}No.jpg", Ruta))
```

```
        End If
```

```
    End Function
```

```
    Public Function ConvertBack(ByVal value As Object, _
```

```
        ByVal targetType As System.Type, ByVal parameter As Object, _
```

```
        ByVal culture As System.Globalization.CultureInfo) As Object _
```

```
    Implements System.Windows.Data.IValueConverter.ConvertBack
```

```
        Archivo = String.Format("{0}{1}.jpg", Ruta, value)
```

```
        If File.Exists(Archivo) Then
```

```
            Return (Archivo)
```

```
        Else
```

```
            Return (String.Format("{0}No.jpg", Ruta))
```

```
        End If
```

```
    End Function
```

```
End Class
```

- Cerrar las ventanas con las 2 clases y compilar la aplicación para poder definir un recurso que apunte a las clases creadas.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Consulta ProductosCategoria.xaml".

- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ConsultaProductosCategoria"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:ld="clr-namespace:Demo94"
  Title="Plantillas de Datos - Sincronización de Listas"
  Height="400" Width="450" ResizeMode="NoResize"
  WindowStartupLocation="CenterScreen"
  Name="ConsultaProductosCategoria">
  <Window.Resources>
    <ld:FormatoDecimal x:Key="fd"/>
    <ld:FormatoImagen x:Key="fi"/>
    <DataTemplate x:Key="dtCategoria">
      <Grid>
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="70"/>
          <ColumnDefinition Width="50"/>
          <ColumnDefinition Width="220"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
          <RowDefinition Height="40"/>
        </Grid.RowDefinitions>
        <Image Name="imgFoto" Grid.Column="0"
          Source="{Binding Path=Codigo,
            Converter={StaticResource fi}}"/>
        <Label Name="lblCodigo" Grid.Column="1"
          Content="{Binding Path=Codigo}"/>
        <Label Name="lblNombre" Grid.Column="2"
          Content="{Binding Path=Nombre}"/>
      </Grid>
    </DataTemplate>
    <DataTemplate x:Key="dtProducto">
      <Grid>
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="50"/>
          <ColumnDefinition Width="180"/>
          <ColumnDefinition Width="50"/>
          <ColumnDefinition Width="50"/>
        </Grid.ColumnDefinitions>
        <Label Name="lblCodProd" Grid.Column="0"
```

```
        Content="{Binding Path=Codigo}"
        HorizontalAlignment="Right"
        Foreground="Yellow"/>
<Label Name="lblNomProd" Grid.Column="1"
        Content="{Binding Path=Nombre}"
        Foreground="Yellow"/>
<Label Name="lblPrecio" Grid.Column="2"
        Content="{Binding Path=PrecioUnitario,
        Converter={StaticResource fd}}"
        HorizontalAlignment="Right"
        Foreground="Yellow"/>
<Label Name="lblStock" Grid.Column="3"
        Content="{Binding Path=Stock}"
        HorizontalAlignment="Right"
        Foreground="Yellow"/>
</Grid>
</DataTemplate>
</Window.Resources>
<Grid Name="grdCategoria">
    <Grid.Background>
        <LinearGradientBrush>
            <GradientStop Color="yellow" Offset="1"/>
            <GradientStop Color="orange" Offset="0.5"/>
        </LinearGradientBrush>
    </Grid.Background>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="30"/>
        <ColumnDefinition Width="370"/>
        <ColumnDefinition Width="30*"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="30"/>
        <RowDefinition Height="40"/>
        <RowDefinition Height="260"/>
        <RowDefinition Height="30*"/>
    </Grid.RowDefinitions>
    <Label Name="lblTitulo" Grid.Row="0" Grid.Column="1"
        Foreground="Blue" FontSize="14" FontWeight="Bold"
        HorizontalAlignment="Center">
        Consulta de Productos x Categoría</Label>
```

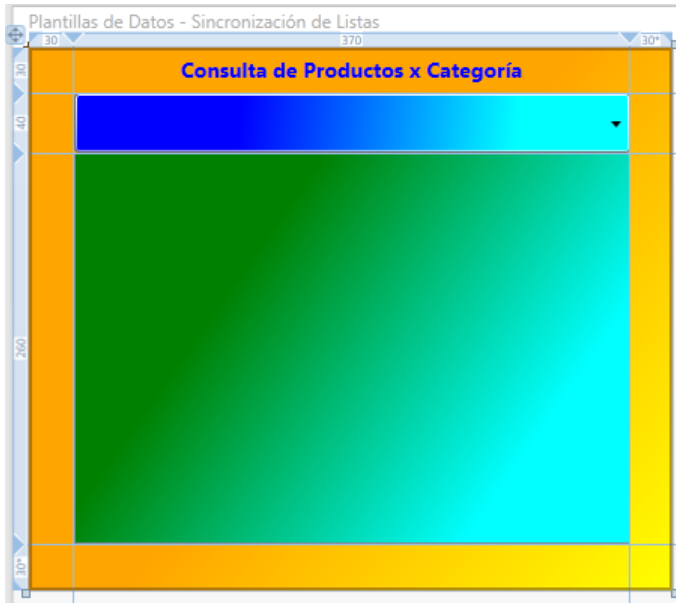
```
<ComboBox Name="cboCategoria" Grid.Row="1" Grid.Column="1"
  ItemsSource="{Binding Path=}"
  ItemTemplate="{StaticResource dtCategoria}"
  IsSynchronizedWithCurrentItem="True">
  <ComboBox.Background>
    <LinearGradientBrush>
      <GradientStop Color="aqua" Offset="0.8"/>
      <GradientStop Color="blue" Offset="0.3"/>
    </LinearGradientBrush>
  </ComboBox.Background>
</ComboBox>
<ListBox Name="lstProducto" Grid.Row="2" Grid.Column="1"
  ItemsSource="{Binding Path=CurrentItem.ListaProductos}"
  ItemTemplate="{StaticResource dtProducto}">
  <ListBox.Background>
    <LinearGradientBrush>
      <GradientStop Color="aqua" Offset="0.8"/>
      <GradientStop Color="green" Offset="0.3"/>
    </LinearGradientBrush>
  </ListBox.Background>
</ListBox>
</Grid>
</Window>
```

Nota: En el código anterior se definen 2 recursos para conversión de datos: fd para FormatoDecimal y fi para FormatoImagen; también se definen 2 plantillas de datos, dtCategoria con 3 columnas: una imagen, código y nombre de la categoría y dtProducto con 4 columnas: código, nombre, precio y stock del producto.

Importante: Para aplicar la plantilla de datos se usa la propiedad Item Template y para sincronizar las 2 listas se configura la propiedad IsSynchronizedWithCurrentItem de la primera lista en true y la propiedad ItemsSource de la segunda lista en CurrentItem.ListaProductos para filtrar automáticamente los productos de acuerdo a la categoría seleccionada en el ComboBox.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.27.

Gráfico 6.27: Diseño de la ventana Consulta de Productos x Categoría



- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Class ConsultaProductosCategoria
```

```
    Private lobeCategoria As New List(Of beCategoria)
```

```
    Private Sub CargarDatos(ByVal sender As System.Object, _
```

```
        ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
```

```
        Dim obrCategoria As New brCategoria
```

```
        lobeCategoria = obrCategoria.ListarConProductos
```

```
        grdCategoria.DataContext = lobeCategoria
```

```
    End Sub
```

```
End Class
```

Nota: En el código anterior al cargar la ventana se llena la lista de categorías donde cada categoría tiene una lista de productos y se enlaza la propiedad DataContext del Grid a la lista de categorías.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ConsultaProductosCategoria.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.28: Ejecución de la ventana Consulta de Productos x Categoría



Demo 95

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo95" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Crear una clase para la conversión a 2 decimales: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoDecimal.vb" y clic en "Add".
- Escribir el siguiente código en la clase:

```
Public Class FormatoDecimal
    Implements IValueConverter

    Public Function Convert(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.Convert
        Return (String.Format("{0:n2}", value))
    End Function

    Public Function ConvertBack(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.ConvertBack
        Return (String.Format("{0:n2}", value))
    End Function
End Class
```

- Agregar una clase para la conversión a solo fecha: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoFecha.vb" y clic en "Add".

- Escribir el siguiente código en la clase FormatoFecha:

```
Public Class FormatoFecha
    Implements IValueConverter

    Public Function Convert(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.Convert
        Return (String.Format("{0:d}", value))
    End Function

    Public Function ConvertBack(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.ConvertBack
        Return (String.Format("{0:d}", value))
    End Function
End Class
```

- Cerrar las ventanas con las 2 clases y compilar la aplicación para poder definir un recurso que apunte a las clases creadas.
- Cambiar de nombre a la ventana "MainWindow.xaml" por "ListaDetallesOrden.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ListaDetallesOrden"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:ld="clr-namespace:Demo95"
    Title="Plantilla de Datos Jerárquica en control TreeView"
    Height="400" Width="400" ResizeMode="CanResize"
    WindowStartupLocation="CenterScreen"
    Name="ListaDetallesOrden"
    WindowState="Maximized">
    <Window.Resources>
        <ld:FormatoFecha x:Key="ff"/>
        <ld:FormatoDecimal x:Key="fd"/>
        <HierarchicalDataTemplate x:Key="hdtOrden"
```

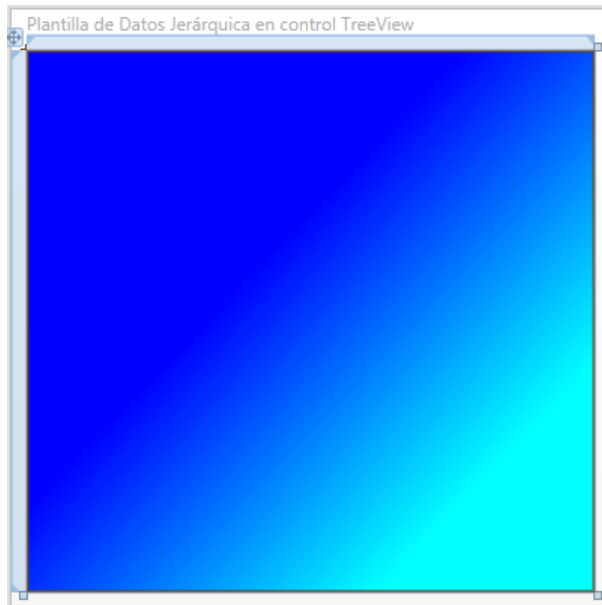
```
ItemsSource="{Binding Path=ListaDetalles}">
<DockPanel>
  <TextBlock Text="{Binding Path=IdOrden}"
    Width="80" Foreground="White" />
  <TextBlock Text="{Binding Path=NombreCliente}"
    Width="200" Foreground="White"/>
  <TextBlock Text="{Binding Path=NombreEmpleado}"
    Width="150" Foreground="White"/>
  <TextBlock Text="{Binding Path=FechaOrden,
    Converter={StaticResource ff}}"
    Width="70" Foreground="White"/>
</DockPanel>
<HierarchicalDataTemplate.ItemTemplate>
  <DataTemplate>
    <Grid>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="50"/>
        <ColumnDefinition Width="10"/>
        <ColumnDefinition Width="240"/>
        <ColumnDefinition Width="10"/>
        <ColumnDefinition Width="50"/>
        <ColumnDefinition Width="10"/>
        <ColumnDefinition Width="50"/>
        <ColumnDefinition Width="10"/>
        <ColumnDefinition Width="50"/>
      </Grid.ColumnDefinitions>
      <Label Grid.Column="0" Foreground="Yellow"
        Content="{Binding Path=IdProducto}"/>
      <Label Grid.Column="2" Foreground="Yellow"
        Content="{Binding Path=NombreProducto}" />
      <Label Grid.Column="4" Foreground="Yellow"
        Content="{Binding Path=PrecioUnitario,
        Converter={StaticResource fd}}"
        HorizontalContentAlignment="Right"/>
      <Label Grid.Column="6" Foreground="Yellow"
        Content="{Binding Path=Cantidad}"
        HorizontalContentAlignment="Right"/>
      <Label Grid.Column="8" Foreground="Yellow"
        Content="{Binding Path=PrecioTotal,
        Converter={StaticResource fd}}"
```

```
        HorizontalContentAlignment="Right"/>
    </Grid>
</DataTemplate>
</HierarchicalDataTemplate.ItemTemplate>
</HierarchicalDataTemplate>
</Window.Resources>
<Grid>
    <TreeView Name="tvwOrden"
        ItemTemplate="{StaticResource hdtOrden}">
        <TreeView.Background>
            <LinearGradientBrush>
                <GradientStop Color="aqua" Offset="0.8"/>
                <GradientStop Color="blue" Offset="0.4"/>
            </LinearGradientBrush>
        </TreeView.Background>
    </TreeView>
</Grid>
</Window>
```

Nota: En el código anterior se definen 2 recursos para conversión de datos: fd para FormatoDecimal y ff para FormatoFecha; también se define una plantilla de datos jerárquica llamada hdtOrden con 4 columnas para la cabecera: Id de orden, nombre del cliente, nombre del empleado y fecha de la orden y 5 columnas para los detalles: Id del producto, nombre del producto, precio unitario, cantidad y precio total. Esta plantilla se aplica al control TreeView en su propiedad ItemTemplate.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.29.

Gráfico 6.29: Diseño de la ventana Lista de Detalles de Ordenes



- Escribir código en el archivo vb asociado al xaml:

```
Imports Northwind.LibBusinessEntities
```

```
Imports Northwind.LibBusinessRules
```

```
Class ListaDetallesOrden
```

```
Private Sub CargarDatos(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
```

```
Try
```

```
Dim obrOrden As New brOrden
```

```
Dim lobeOrden As List(Of beOrden) = obrOrden.ListarConDetalles
```

```
twvOrden.ItemsSource = lobeOrden
```

```
Catch ex As Exception
```

```
End Try
```

```
End Sub
```

```
End Class
```

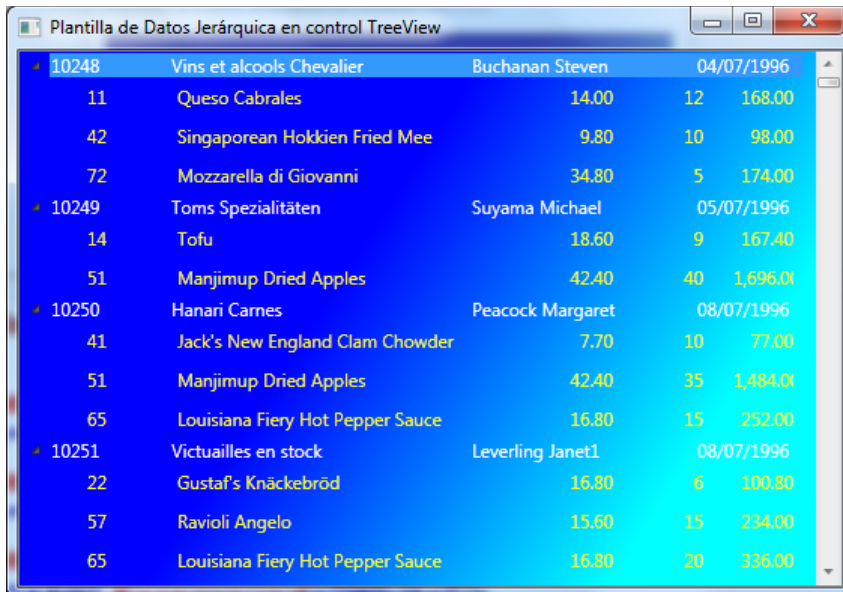
Nota: En el código anterior al cargar la ventana se llena la lista de órdenes donde cada orden tiene una lista de detalles y se enlaza la propiedad `ItemsSource` del `TreeView` a la lista de órdenes.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ListaDetallesOrden.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.30: Ejecución de la ventana Lista de Detalles de Ordenes



10248	Vins et alcools Chevalier	Buchanan Steven	04/07/1996
11	Queso Cabrales	14.00	12 168.00
42	Singaporean Hokkien Fried Mee	9.80	10 98.00
72	Mozzarella di Giovanni	34.80	5 174.00
10249	Toms Spezialitäten	Suyama Michael	05/07/1996
14	Tofu	18.60	9 167.40
51	Manjimup Dried Apples	42.40	40 1,696.00
10250	Hanari Carnes	Peacock Margaret	08/07/1996
41	Jack's New England Clam Chowder	7.70	10 77.00
51	Manjimup Dried Apples	42.40	35 1,484.00
65	Louisiana Fiery Hot Pepper Sauce	16.80	15 252.00
10251	Victuailles en stock	Leverling Janet1	08/07/1996
22	Gustaf's Knäckebröd	16.80	6 100.80
57	Ravioli Angelo	15.60	15 234.00
65	Louisiana Fiery Hot Pepper Sauce	16.80	20 336.00

2.5. Usando el Control DataGrid

Hasta antes de la versión 2010 la única forma de mostrar un conjunto de datos en filas y varias columnas era usar el ListView o aplicar una plantilla al ListBox para que tenga varias columnas. Con .NET Framework 4 y Visual Studio 2010 WPF trae el nuevo control DataGrid que simplifica este punto.

El control DataGrid proporciona una forma flexible de mostrar una colección de datos en filas y columnas. El DataGrid incluye tipos de columna integradas y una columna de plantilla para hospedar contenido personalizado. El tipo de fila integrado incluye una sección de detalles desplegable que se puede utilizar para mostrar contenido adicional debajo de los valores de celda.

Para enlazar el DataGrid, configure la propiedad ItemsSource a un origen de datos. Cada fila de la cuadrícula de datos se enlaza a un objeto en el origen de datos, y cada columna de la cuadrícula de datos se enlaza a una propiedad del objeto de datos.

De manera predeterminada, el control DataGrid genera automáticamente columnas cuando se establece la propiedad ItemsSource. El tipo de columna que se genera depende del tipo de datos de la columna. Los tipos de columnas en el DataGrid pueden ser: DataGridTextColumn, DataGridCheckBoxColumn y DataGridComboBoxColumn.

Puede personalizar las columnas del DataGrid usando la propiedad AutoGenerateColumns configurándola en false y creando sus propias columnas, además puede personalizar el estilo usando la propiedad Style o cambiar el aspecto de todo el control usando ControlTemplate.

Para obtener más información sobre el control DataGrid de WPF ver la referencia 42 al final del libro.

A continuación un ejemplo de cómo crear columnas personalizadas en el DataGrid para mostrar el código, nombre, precio unitario y stock de los productos, alineando el precio y stock a la derecha y formateando a 2 decimales el precio.

Demo 96

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo96" y clic en "OK".
- Hacer una referencia a las librerías Northwind.LibBusinessEntities y Northwind.LibBusinessRules.
- Crear una clase para la conversión a 2 decimales: del menú "Project", seleccionar "Add Class", escribir como nombre: "FormatoDecimal.vb" y clic en "Add".
- Escribir el siguiente código en la clase:

```
Public Class FormatoDecimal
    Implements IValueConverter

    Public Function Convert(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.Convert
        Return (String.Format("{0:n2}", value))
    End Function

    Public Function ConvertBack(ByVal value As Object, _
        ByVal targetType As System.Type, ByVal parameter As Object, _
        ByVal culture As System.Globalization.CultureInfo) As Object _
        Implements System.Windows.Data.IValueConverter.ConvertBack
        Return (String.Format("{0:n2}", value))
    End Function
End Class
```

- Cerrar la ventana con la clase creada y compilar la aplicación para poder definir un recurso que permita usar la clase.

- Cambiar de nombre a la ventana "MainWindow.xaml" por "Lista Productos.xaml".
- Escribir el siguiente código en el archivo xaml:

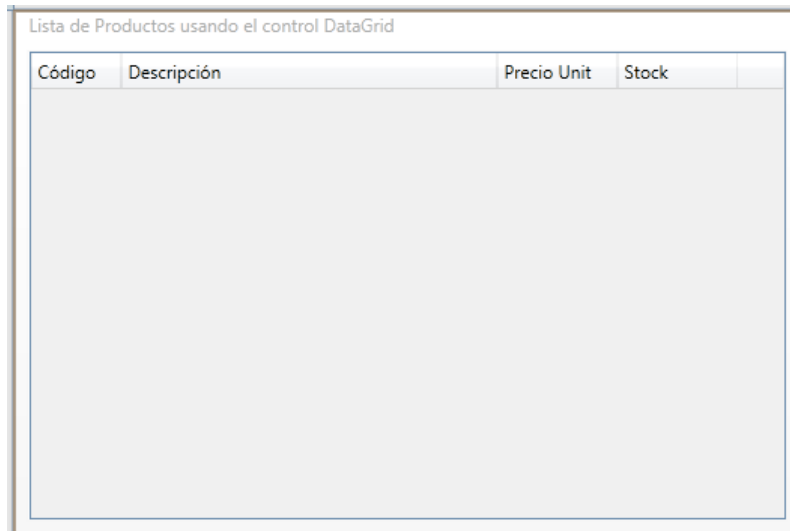
```
<Window x:Class="ListaProductos"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:ld="clr-namespace:Demo96"
    Title="Lista de Productos usando el control DataGrid"
    Height="350" Width="525" ResizeMode="NoResize"
    WindowStartupLocation="CenterScreen" >
    <Window.Resources>
        <ld:FormatoDecimal x:Key="fd"/>
    </Window.Resources>
    <Grid>
        <DataGrid Name="dgProducto" AutoGenerateColumns="False">
            <DataGrid.RowBackground>
                <LinearGradientBrush>
                    <GradientStop Color="Aqua" Offset="1"/>
                    <GradientStop Color="Green" Offset="0.5"/>
                </LinearGradientBrush>
            </DataGrid.RowBackground>
            <DataGrid.Columns>
                <DataGridTextColumn Header="Código" Width="60"
                    Binding="{Binding Path=Codigo}"/>
                <DataGridTextColumn Header="Descripción" Width="250"
                    Binding="{Binding Path=Nombre}"/>
                <DataGridTextColumn Header="Precio Unit" Width="80"
                    Binding="{Binding Path=PrecioUnitario,
                    Converter={StaticResource fd}}">
                    <DataGridTextColumn.CellStyle>
                        <Style>
                            <Setter Property="FrameworkElement.
                                HorizontalAlignment"
                                Value="Right"/>
                        </Style>
                    </DataGridTextColumn.CellStyle>
                </DataGridTextColumn>
                <DataGridTextColumn Header="Stock" Width="80"
                    Binding="{Binding Path=Stock}"/>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Window>
```

```
<DataGridTextBoxColumn.CellStyle>
  <Style>
    <Setter Property="FrameworkElement.
      HorizontalAlignment"
      Value="Right"/>
  </Style>
</DataGridTextBoxColumn.CellStyle>
</DataGridTextBoxColumn>
</DataGrid.Columns>
</DataGrid>
</Grid>
</Window>
```

Nota: En el código anterior se define un recurso para conversión de datos llamado `fd` para aplicar la clase `FormatoDecimal` y se incluye solo un control `DataGrid` que define sus filas con un fondo degradado y 4 columnas, 2 de las cuales tienen estilo para alinear a la derecha los datos.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.31.

Gráfico 6.31: Diseño de la ventana Lista de Productos



- Escribir código en el archivo `vb` asociado al `xaml`:

```
Imports Northwind.LibBusinessEntities
Imports Northwind.LibBusinessRules
```

```
Class ListaProductos
    Private Sub ListarProductos(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded
        Dim obrProducto As New brProducto
        Dim lobeProducto As List(Of beProducto) = obrProducto.Listar
        dgProducto.ItemsSource = lobeProducto
    End Sub
End Class
```

Nota: En el código anterior al cargar la ventana se llena la lista de productos y se enlaza la propiedad ItemsSource del DataGrid a la lista de productos.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ListaProductos.xaml".
- Agregar un archivo de configuración a la aplicación: del menú "Project", seleccionar "Add New Item", en la categoría "General" seleccionar "Application Configuration File".
- Agregar en el archivo de configuración la clave conNW en la sección appSettings:

```
<configuration>
  <appSettings>
    <add key="conNW" value="uid=sa;pwd=123456;
      data source=Lduenas\MCTS;initial catalog=Northwind"/>
  </appSettings>
</configuration>
```

- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.32: Ejecución de la ventana Lista de Productos

Código	Descripción	Precio Unit	Stock
1	Chai	10.00	81
2	Chang	3.00	24
3	Chicha	10.00	39
4	Chef Anton's Cajun Seasoning	22.00	80
5	Chef Anton's Gumbo Mix	21.35	31
6	Grandma's Boysenberry Spread	25.00	50
7	Uncle Bob's Organic Dried Pears	30.00	98
8	Northwoods Cranberry Sauce	40.00	99
9	Mishi Kobe Niku	97.00	96
10	Ikura	31.00	93
11	Queso Cabrales	21.00	97
12	Queso Manchego La Pastora	38.00	100
13	Konbu	6.00	100
14	Tofu	23.25	99
15	Genen Shouyu	15.50	39
16	Paulownia	17.45	27

Importante: Por defecto el DataGrid trae incorporado la funcionalidad de ordenar cada columna en forma ascendente y descendente al dar clic a las cabeceras de las columnas.

3. Manejando Documentos

En esta parte trataremos el manejo de documentos en WPF, iniciando con una revisión de las capacidades de manejo de documentos que tiene WPF que incluyen los documentos estáticos, documentos dinámicos, controles para manejo de documentos, anotaciones, documentos XPS, etc.

Luego pasaremos a manejar los documentos fijos que mantienen siempre el contenido aunque cambie el tamaño de la ventana o la resolución de la pantalla, para lo cual veremos el control `DocumentViewer`. En esta parte veremos como trabajar con documentos XPS mediante la clase `XpsDocument` y el control `DocumentViewer`, aquí crearemos como ejemplo un visor de archivos XPS.

Seguidamente veremos como trabajar con anotaciones en WPF que incluyen las notas rápidas y el resaltado de texto, aquí crearemos como ejemplo un visor para archivos de Word y Excel, a los cuales le podremos incluir las anotaciones.

Finalmente, se verá el trabajo con documentos dinámicos usando los controles `FlowDocumentReader`, `FlowDocumentPageViewer` y `FlowDocumentScrollViewer`, aquí crearemos como ejemplo un visor de archivos de texto.

3.1. Introducción al Manejo de Documentos

WPF permite el manejo de documentos lo cual incluye la presentación, el empaquetado y la seguridad. Para esto clasifica a los documentos en 2 categorías: fijos y dinámicos.

Los documentos fijos son aquéllos que mantienen su ubicación y tamaño sin importar el dispositivo, es decir son del tipo “Lo que se ve es lo que se obtiene” o sus siglas en inglés WYSIWYG (What You See Is What You Get), entre sus usos tenemos las publicaciones, procesamiento de texto, diseño de formularios, etc, donde lo que se en pantalla es lo mismo que saldrá impreso.

Por su parte los documentos dinámicos se ajustan o cambian de ubicación y tamaño de acuerdo al tamaño de la ventana, la resolución de la pantalla, etc, con el objetivo principal de facilitar la lectura al usuario. Además, los documentos dinámicos tienen varias características integradas que incluyen la búsqueda, modos de presentación que optimizan la legibilidad y la capacidad de cambiar el tamaño y aspecto de las fuentes.

WPF tiene controles que permiten visualizar documentos; para los documentos fijos se usa el control `DocumentViewer` que muestra contenido de tipo `FixedDocument` y para los documentos dinámicos se tienen los controles: `FlowDocumentReader`, `FlowDocumentPageViewer` y `FlowDocumentScrollView`, de ellos el primero permite elegir varios tipos de visualización de página.

También podemos incluir en nuestros documentos comentarios o “anotaciones” para agregar notas rápidas o resaltar información de interés usando `Microsoft Annotations Framework`.

Finalmente, WPF permite trabajar con documentos XPS (XML Paper Specification) que son documentos fijos y que se usa como formato de impresión nativa en Windows Vista, para lo cual se tiene la clase `XpsDocument` que se encuentra en `System.Windows.Xps.Packaging`.

Para obtener más información sobre Documentos en WPF ver la referencia 43 al final del libro.

3.2. Trabajando con Documentos Fijos

Según Microsoft: "Los documentos fijos están diseñados para las aplicaciones que requieren una presentación What You See Is What You Get (WYSIWYG) precisa, independiente del hardware de pantalla o de impresión utilizado".

Los usos típicos para los documentos fijos incluyen la creación de publicaciones, el procesamiento de textos y el diseño de formularios, donde es vital que se respete el diseño de página original.

Un documento fijo mantiene la colocación posicional precisa de los elementos de contenido con independencia del dispositivo de pantalla o de impresión utilizado. El diseño de la página permanece inalterado en todos los casos, aunque la calidad del documento se maximiza de acuerdo con las funciones de cada dispositivo.

Por su parte, un documento XML Paper Specification (XPS) es un paquete que contiene uno o más documentos fijos junto con todos los recursos y la información necesarios para la presentación.

XPS también es el formato de archivo nativo de cola de impresión de Windows Vista. XpsDocument se almacena en el conjunto de datos ZIP estándar, y puede incluir una combinación de XML y componentes binarios, como imágenes y archivos de fuente.

La clase XpsDocument tiene los siguientes usos:

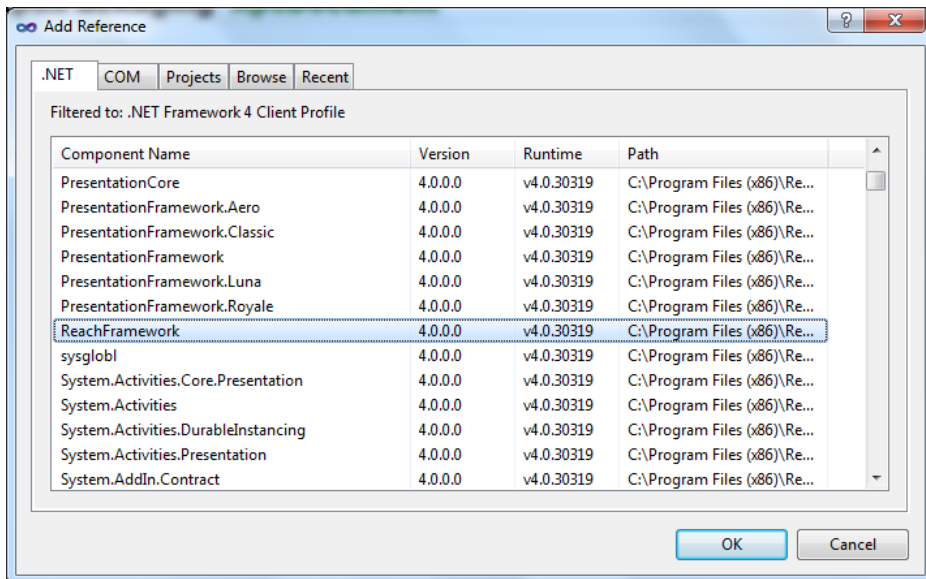
- Lectura, escritura y almacenamiento de contenido y recursos de documentos fijos en un único archivo portátil y fácil de distribuir.
- Presentación de documentos con la aplicación Visor de XPS.
- Generación de documentos en el formato de salida de cola de impresión de Windows Vista.
- Enrutamiento directo de documentos a las impresoras compatibles con XPS.

A continuación un ejemplo de cómo leer documentos fijos XPS en WPF.

Demo 97

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo97" y clic en "OK".
- Hacer una referencia a la librería donde está Xps: del menú "Project", seleccionar "Add References", del diálogo elegir: "ReachFramework".

Gráfico 6.33: Diálogo de Referencia de la librería ReachFramework



- Cambiar de nombre a la ventana "MainWindow.xaml" por "Visor XPS.xaml".
- Escribir el siguiente código en el archivo xaml:

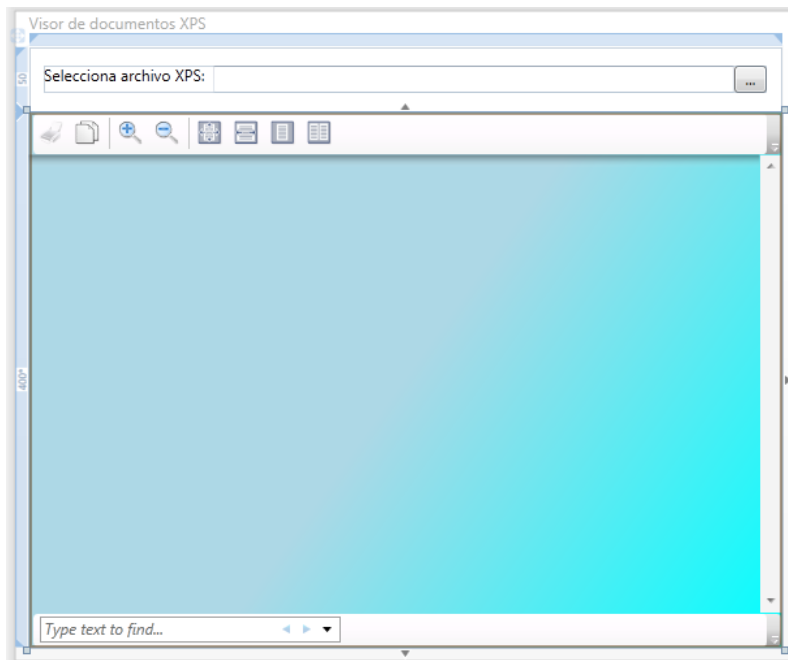
```
<Window x:Class="VisorXPS"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Visor de documentos XPS" Height="500" Width="600"
    ResizeMode="NoResize"
```

```
WindowStartupLocation="CenterScreen">
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="50"/>
    <RowDefinition Height="400*"/>
  </Grid.RowDefinitions>
  <DockPanel Grid.Row="0" VerticalAlignment="Center"
    HorizontalAlignment="Center">
    <TextBlock Text="Selecciona archivo XPS: "/>
    <TextBox Name="txtArchivo" Width="400"/>
    <Button Name="btnAbrir" Content="..." Width="25"
      Cursor="Hand" ToolTip="Abrir archivo XPS"/>
  </DockPanel>
  <DocumentViewer Name="dvXPS" Grid.Row="1">
    <DocumentViewer.Background>
      <LinearGradientBrush>
        <GradientStop Color="Aqua" Offset="1"/>
        <GradientStop Color="LightBlue" Offset="0.5"/>
      </LinearGradientBrush>
    </DocumentViewer.Background>
  </DocumentViewer>
</Grid>
</Window>
```

Nota: En el código anterior se definen 2 filas para el Grid, en la primera va un DockPanel que contiene un bloque de texto con un mensaje, un cuadro de texto y un botón para mostrar el diálogo de abrir para seleccionar el archivo XPS. En la segunda fila va el DocumentViewer que tiene un fondo degradado.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.34.

Gráfico 6.34: Diseño de la ventana Visor de XPS



- Escribir código en el archivo vb asociado al xaml:

```
Imports Microsoft.Win32 'OpenFileDialog
Imports System.IO 'FileAccess
Imports System.Windows.Xps.Packaging 'XpsDocument
```

Class VisorXPS

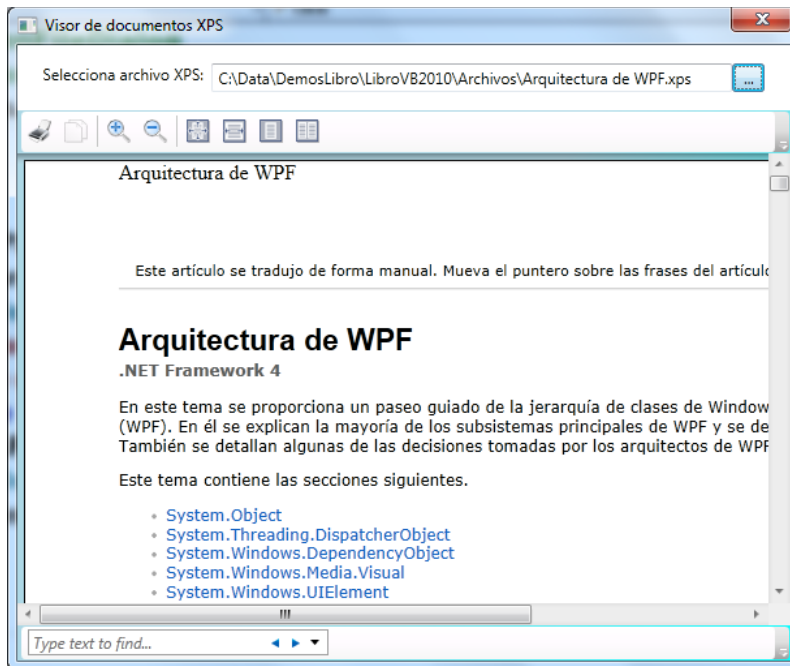
```
Private Sub AbrirArchivoXPS(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs) Handles btnAbrir.Click
    Dim ofd As New OpenFileDialog
    ofd.Title = "Selecciona archivo XPS a abrir"
    ofd.Filter = "Archivos de documentos XPS|*.xps"
    If ofd.ShowDialog = True Then
        txtArchivo.Text = ofd.FileName
        Dim xps As New XpsDocument(ofd.FileName, FileAccess.Read)
        dvXPS.Document = xps.GetFixedDocumentSequence
    End If
End Sub
End Class
```

Nota: En el código anterior al dar clic en el botón “Abrir” se muestra el diálogo de abrir con archivos XPS, al elegir uno se crea un objeto XpsDocument para enlazar el archivo al visor.

Importante: Para enlazar el documento XPS al visor (DocumentViewer) se usa la propiedad Document que recibe un FixedDocument el cual se obtiene a través del método GetFixedDocumentSequence del objeto XpsDocument.

- Regresar a la ventana “Application.xaml” y cambiar la ventana que inicia (atributo StartupUri) a “VisorXPS.xaml”.
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.35: Ejecución de la ventana Visor de XPS



3.3. Creando Anotaciones en Documentos

Las anotaciones permiten aumentar comentarios a ciertas partes del texto de un documento y puede ser de 3 tipos: una nota rápida, un resaltado o una nota de tinta. Los controles WPF de visualización de documentos ya sean fijos como el `DocumentViewer` o dinámicos como el `FlowDocumentReader`, `FlowDocumentPageViewer` y `FlowDocumentScrollView` permiten anotaciones.

Para implementar anotaciones en documentos es necesario usar la clase `AnnotationService` ubicada en el espacio de nombres `System.Windows.Annotations` y luego se podrá crear un resaltado con `CreateHighlightCommand` o una nota rápida con `CreateTextStickyNoteCommand` o una nota de tinta con `CreateInkStickyNoteCommand`.

Para quitar los resaltados se usa `ClearHighlightsCommand`, para eliminar notas de texto rápidas se usa `DeleteStickyNotesCommand` y para eliminar todas las anotaciones del documento se usa `DeleteAnnotationsCommand`.

Para crear una anotación, el usuario suele seleccionar primero algún texto o elemento de interés y, a continuación, hacer clic con el botón secundario del mouse para mostrar un `ContextMenu` de opciones de anotación.

Para obtener más información sobre Anotaciones en WPF ver la referencia 44 al final del libro.

A continuación un ejemplo que demuestra como crear y eliminar anotaciones en documentos de Word y Excel usando un visor de documentos creado convirtiendo dichos archivos en archivos XPS y luego visualizándolos mediante el `DocumentViewer` usando para ello la clase `XpsDocument`.

En el ejemplo una vez seleccionado el archivo de Word o Excel y convertido a XPS, debe seleccionar un texto, palabra o párrafo y dar clic derecho para mostrar un menú contextual con las opciones de anotaciones, ya sea para crear o para eliminar.

Demo 98

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo98" y clic en "OK".
- Hacer una referencia a la librería donde está Xps: del menú "Project", seleccionar "Add References", del diálogo elegir: "ReachFramework".
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Visor Office.xaml".
- Escribir el siguiente código en el archivo xaml:

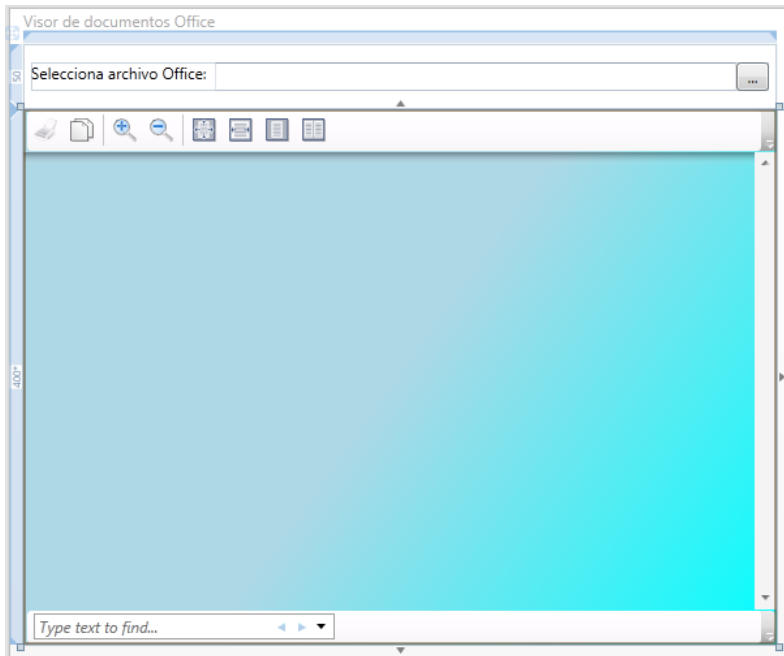
```
<Window x:Class="VisorOffice"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:ann="clr-namespace:System.Windows.Annotations;assembly=
  PresentationFramework"
  Title="Visor de documentos Office" Height="500" Width="600"
  ResizeMode="NoResize"
  WindowStartupLocation="CenterScreen">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="50"/>
      <RowDefinition Height="400*" />
    </Grid.RowDefinitions>
    <DockPanel Grid.Row="0" VerticalAlignment="Center"
      HorizontalAlignment="Center">
      <TextBlock Text="Selecciona archivo Office: " />
      <TextBox Name="txtArchivo" Width="400"/>
      <Button Name="btnAbrir" Content="..." Width="25"
        Cursor="Hand" ToolTip="Abrir archivo XPS"/>
    </DockPanel>
    <DocumentViewer Name="dvXPS" Grid.Row="1">
      <DocumentViewer.Background>
        <LinearGradientBrush>
```

```
<GradientStop Color="Aqua" Offset="1"/>
<GradientStop Color="LightBlue" Offset="0.5"/>
</LinearGradientBrush>
</DocumentViewer.Background>
<DocumentViewer.ContextMenu>
  <ContextMenu>
    <MenuItem Command="ApplicationCommands.Copy" />
    <Separator />
    <MenuItem Command="ann:AnnotationService.
      CreateHighlightCommand"
      Header="Adicionar Resaltado" />
    <MenuItem Command="ann:AnnotationService.
      CreateTextStickyNoteCommand"
      Header="Adicionar Nota de Texto" />
    <MenuItem Command="ann:AnnotationService.
      CreateInkStickyNoteCommand"
      Header="Adicionar Nota Tinta" />
    <Separator />
    <MenuItem Command="ann:AnnotationService.
      ClearHighlightsCommand"
      Header="Eliminar Resaltado" />
    <MenuItem Command="ann:AnnotationService.
      DeleteStickyNotesCommand"
      Header="Eliminar Notas" />
    <MenuItem Command="ann:AnnotationService.
      DeleteAnnotationsCommand"
      Header="Eliminar Todo" />
  </ContextMenu>
</DocumentViewer.ContextMenu>
</DocumentViewer>
</Grid>
</Window>
```

Nota: En el código anterior se definen 2 filas para el Grid, en la primera va un DockPanel que contiene un bloque de texto con un mensaje, un cuadro de texto y un botón para mostrar el diálogo de abrir para seleccionar el archivo doc o xls. En la segunda fila va el DocumentViewer que tiene un fondo degradado y un menú contextual con 7 opciones, una para copiar, 3 para agregar anotaciones y 3 para eliminar anotaciones.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.36.

Gráfico 6.36: Diseño de la ventana Visor de Office



- Escribir código en el archivo vb asociado al xaml:

```
Imports Microsoft.Win32 'OpenFileDialog
Imports System.IO 'FileAccess
Imports System.Windows.Xps.Packaging 'XpsDocument
Imports System.Runtime.InteropServices 'Marshal
Imports System.Windows.Annotations 'AnnotationService
Imports System.Windows.Annotations.Storage 'XmlStreamStore
```

Class VisorOffice

```
Private Sub AbrirArchivoOffice(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles btnAbrir.Click
    Dim ofd As New OpenFileDialog
    ofd.Title = "Selecciona archivo Office a abrir"
    ofd.Filter = "Archivos de Office|*.doc;*.xls"
    If ofd.ShowDialog = True Then
        txtArchivo.Text = ofd.FileName
        Dim docXPS As String = String.Format("{0}\{1}.xps", _
            Path.GetDirectoryName(ofd.FileName), _
```



```
Path.GetFileNameWithoutExtension(ofd.FileName))
If Path.GetExtension(ofd.FileName) = ".doc" Then
    Dim oWord As Object = CreateObject("Word.Application")
    Dim oDocumentos As Object = oWord.Documents
    Dim oDocumento As Object = oDocumentos.Open(ofd.FileName)
    oDocumento.ExportAsFixedFormat(docXPS, 18)
    oDocumento.Close(True)
    Marshal.ReleaseComObject(oDocumento)
    oDocumento = Nothing
    Marshal.ReleaseComObject(oDocumentos)
    oDocumentos = Nothing
    oWord.Quit()
    Marshal.ReleaseComObject(oWord)
    oWord = Nothing
Else
    Dim oExcel As Object = CreateObject("Excel.Application")
    Dim oLibros As Object = oExcel.Workbooks
    Dim oLibro As Object = oLibros.Open(ofd.FileName)
    Dim oHoja As Object = oLibro.ActiveSheet
    oHoja.ExportAsFixedFormat(1, docXPS)
    Marshal.ReleaseComObject(oHoja)
    oHoja = Nothing
    oLibro.Close(True)
    Marshal.ReleaseComObject(oLibro)
    oLibro = Nothing
    Marshal.ReleaseComObject(oLibros)
    oLibros = Nothing
    oExcel.Quit()
    Marshal.ReleaseComObject(oExcel)
    oExcel = Nothing
End If
Dim xps As New XpsDocument(docXPS, IO.FileAccess.Read)
dvXPS.Document = xps.GetFixedDocumentSequence
Dim fs As New FileStream("Notas.txt", FileMode.Create, _
    FileAccess.Write)
Dim X As AnnotationService = New AnnotationService(dvXPS)
Dim data As XmlStreamStore = New XmlStreamStore(fs)
If X.IsEnabled Then
    X.Disable()
End If
```

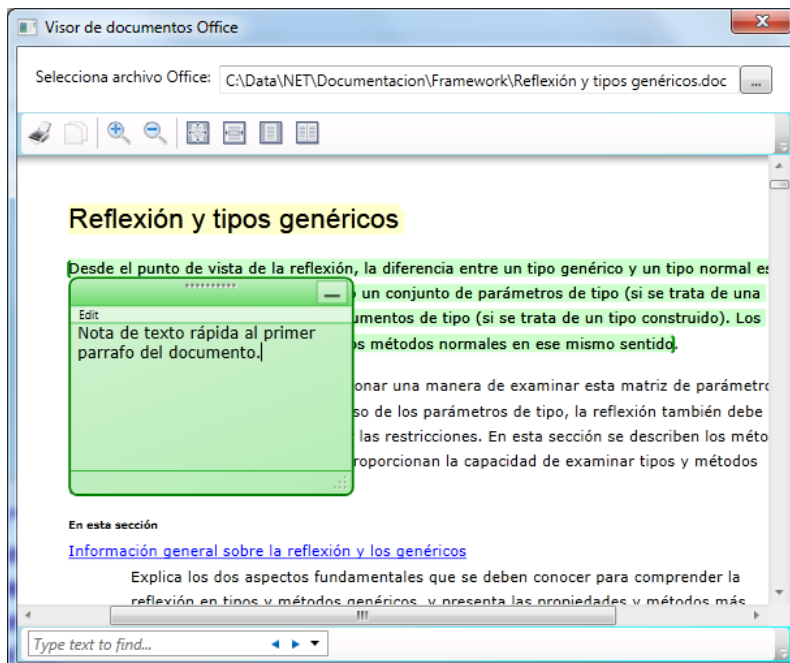
```
X.Enable(data)
End If
End Sub
End Class
```

Nota: En el código anterior al dar clic en el botón “Abrir” se muestra el diálogo de abrir con archivos doc y xls, dependiendo del tipo de archivo a abrir se crea una instancia de Word o de Excel y se usa el método ExportAsFixedFormat para convertir el archivo en XPS, luego se crea un XpsDocument con dicho archivo y se enlaza al visor, finalmente se habilita las anotaciones sobre el documento.

Importante: Si se crea instancias de Word o Excel mediante código, es necesario definir explícitamente todas los objetos usados y luego cerrarlos y liberarlos de memoria, sino se queda “pegado” el ejecutable en memoria.

- Regresar a la ventana “Application.xaml” y cambiar la ventana que inicia (atributo StartupUri) a “VisorOffice.xaml”.
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.37: Ejecución de la ventana Visor de Office



3.4. Trabajando con Documentos Dinámicos

Según Microsoft: “los documentos dinámicos están diseñados para optimizar su presentación y legibilidad y son óptimos para su uso cuando la facilidad de lectura constituye el principal escenario de consumo del documento”.

En lugar de establecerse en un diseño predefinido, este tipo de documentos ajusta y recoloca dinámicamente su contenido basándose en las variables de tiempo de ejecución, tales como el tamaño de la ventana, la resolución del dispositivo y las preferencias opcionales del usuario.

Una página web constituye un ejemplo sencillo de un documento dinámico donde se da formato al contenido de la página dinámicamente para ajustarlo a la ventana activa.

Para mostrar un documento dinámico, WPF tiene 3 controles:

- **FlowDocumentReader:** Tiene distintos modos de visualización, incluido el modo de visualización de una sola página (una página a la vez), dos páginas a la vez (formato de lectura de libro) y desplazamiento continuo (sin límite).
- **FlowDocumentPageViewer:** Muestra el contenido en el modo de visualización de una sola página.
- **FlowDocumentScrollViewer:** Muestra el contenido en modo de desplazamiento continuo.

Nota: De manera predeterminada, se muestra siempre una barra de desplazamiento vertical y la barra de desplazamiento horizontal se vuelve visible cuando es necesario. La IU predeterminada para FlowDocument ScrollViewer no incluye barra de herramientas; sin embargo, se puede utilizar la propiedad `IsToolBarVisible` para habilitar una barra de herramientas integrada.

A continuación un ejemplo de visualización de documentos dinámicos de tipo texto: txt, xml, html mediante el control FlowDocumentReader que permite ver el documento de varias formas.

Demo 99

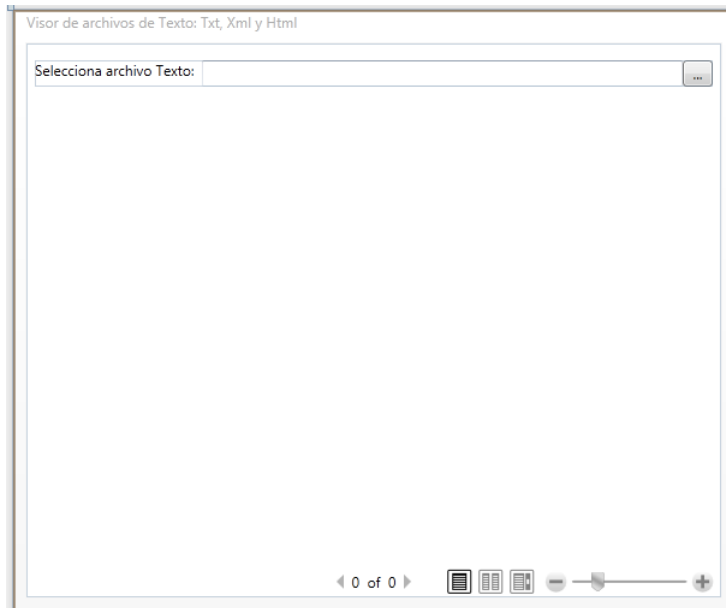
- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo99" y clic en "OK".
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Visor Texto.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="VisorTexto"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Visor de archivos de Texto: Txt, Xml y Html"
    Height="500" Width="600"
    WindowStartupLocation="CenterScreen">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition Height="400*" />
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0" VerticalAlignment="Center"
            HorizontalAlignment="Center">
            <TextBlock Text="Selecciona archivo Texto: "/>
            <TextBox Name="txtArchivo" Width="400"/>
            <Button Name="btnAbrir" Content="..." Width="25"
                Cursor="Hand" ToolTip="Abrir archivo XPS"/>
        </DockPanel>
        <FlowDocumentReader Name="fdrTexto" Grid.Row="1" />
    </Grid>
</Window>
```

Nota: En el código anterior se definen 2 filas para el Grid, en la primera va un DockPanel que contiene un bloque de texto con un mensaje, un cuadro de texto y un botón para mostrar el diálogo de abrir para seleccionar el archivo txt, xml o html. En la segunda fila va el FlowDocumentReader.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.38.

Gráfico 6.38: Diseño de la ventana Visor de Texto



- Escribir código en el archivo vb asociado al xaml:

```
Imports Microsoft.Win32 'OpenFileDialog
Imports System.IO 'FileAccess
Imports System.Text 'Encoding
Imports System.Windows.Documents 'FlowDocument,Paragraph
```

Class VisorTexto

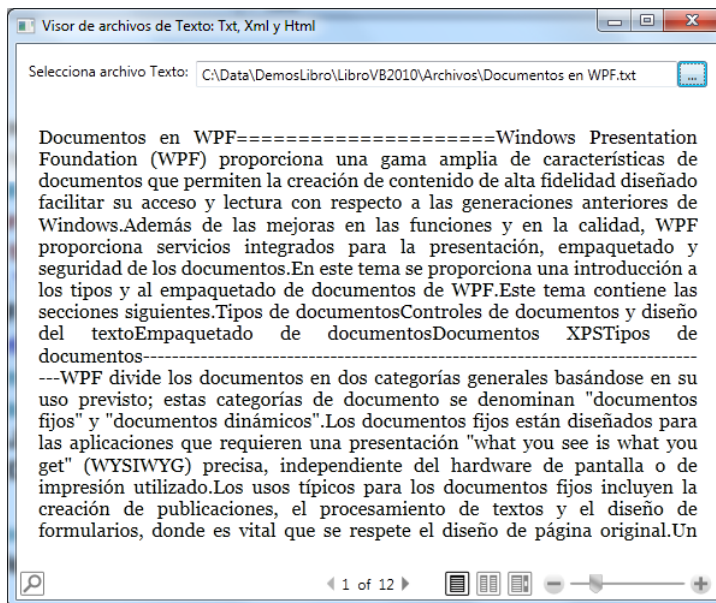
```
Private Sub AbrirArchivoTexto(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs) Handles btnAbrir.Click
    Dim ofd As New OpenFileDialog
    ofd.Title = "Selecciona archivo de texto a abrir"
    ofd.Filter = "Archivos de texto txt, xml y html|*.txt;*.xml;*.html"
    If ofd.ShowDialog = True Then
        txtArchivo.Text = ofd.FileName
        Dim Contenido() As String = File.ReadAllLines _
(ofd.FileName, Encoding.Default)
        Dim parrafo As New Paragraph
        For I = 0 To Contenido.Length - 1
```

```
        parrafo.Inlines.Add(Contenido(I))
    Next
    Dim fd As New FlowDocument(parrafo)
    fdrTexto.Document = fd
End If
End Sub
End Class
```

Nota: En el código anterior al dar clic en el botón "Abrir" se muestra el diálogo de abrir con archivos txt, xml y html, al seleccionar un archivo se lee todas sus líneas y se hace un recorrido por cada una para agregarla a la colección de líneas del objeto párrafo y finalmente se crea un objeto Flow Document pasando como parámetro el párrafo, éste es enlazado a la propiedad document del visor (FlowDocumentReader).

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "VisorOffice.xaml".
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.39: Ejecución de la ventana Visor de Office



4. Manejando Multimedia

En esta última parte veremos cómo trabajar con Multimedia en WPF, es decir con voz, audio y video. Para esto primero daremos una introducción al manejo de multimedia en WPF, revisando las APIs Multimedia que básicamente constan de dos clases que son `MediaElement` y `MediaPlayer`. Además conoceremos los modos de reproducción que son el modo independiente y el modo reloj (Clock).

Seguidamente, veremos como usando la librería `System.Speech.dll` podemos implementar el reconocimiento de voz en nuestras aplicaciones mejorando la accesibilidad. Específicamente, usaremos la clase `SpeechSynthesizer` que se encuentra en el espacio de nombres `System.Speech.Synthesis`.

Con esta clase podremos reproducir la voz usando métodos como `Speak` o su correspondiente versión asíncrona que es `SpeakAsync`. Además podremos controlar las características de la voz como la velocidad usando la propiedad `Rate` o el volumen usando la propiedad `Volume`.

Finalmente, trataremos como reproducir audio y video usando la clase `MediaElement` que tiene la propiedad `Source` para especificar la URI con el archivo a reproducir, el método `Play` que permite reproducir el audio o video y programar en los eventos `MediaOpened` que ocurre cuando inicia la reproducción y `MediaEnded` que ocurre cuando finaliza la reproducción.

Todos estos temas se entenderán mejor con los ejemplos que incluyen un reproductor de voz, un reproductor de sonido o audio y un reproductor de video para dar un final feliz a esta obra.

4.1. Introducción al Manejo de Multimedia

Las características multimedia de WPF permiten integrar audio y vídeo en las aplicaciones para mejorar la experiencia del usuario.

APIs Multimedia

Las clases `MediaElement` y `MediaPlayer` se utilizan para presentar contenido de audio o vídeo. Estas clases se pueden controlar interactivamente o mediante un reloj. Estas clases pueden utilizar el control Microsoft Windows Media Player 10 para la reproducción multimedia. La clase que se utiliza depende del escenario.

`MediaElement` es una clase `UIElement` compatible con sistema de diseño que se puede utilizar como contenido de muchos controles. También se puede utilizar en Extensible Application Markup Language (XAML), así como en el código. `MediaPlayer`, por otro lado, está diseñado para los objetos `Drawing` y carece de compatibilidad de diseño. Los objetos multimedia cargados mediante `MediaPlayer` únicamente se pueden presentar utilizando `VideoDrawing` o interactuando directamente con `DrawingContext`. `Media Player` no se puede utilizar en XAML.

Modos de Reproducción Multimedia

Para entender la reproducción multimedia en WPF, es preciso entender los distintos modos en que se pueden reproducir los objetos multimedia. `MediaElement` y `MediaPlayer` se pueden utilizar en dos modos multimedia diferentes: el modo independiente y el modo de reloj.

El modo multimedia se determina mediante la propiedad `Clock`. Cuando `Clock` es `null`, el objeto multimedia está en modo independiente. Cuando `Clock` no es `null`, el objeto multimedia está en modo de reloj. De manera predeterminada, los objetos multimedia están en modo independiente.

Para obtener más información sobre multimedia ver la referencia 45 al final del libro.

4.2. Implementando Voz

Aunque no es parte del Framework de WPF sino de .NET Framework, incluir voz es una característica multimedia que nos ayuda a aumentar la accesibilidad a las aplicaciones sobre todo para aquellas personas que tienen dificultades con la vista, o también para juegos, programas de instrucciones, etc.

Existe el espacio de nombres System.Speech que contienen tipos de admiten el reconocimiento de voz, la cual se divide en los espacios de nombres: AudioFormat, Recognition y Synthesis. Esta última es la que nos permite implementar voz en nuestras aplicaciones.

La clase **SpeechSynthesizer** tiene métodos como Speak y SpeakAsync que permiten emitir voz; la diferencia es que Speak no es asíncrono, es decir mientras reproduce la voz la IU queda detenida hasta que termine, mientras SpeakAsync es asíncrono, es decir no detiene la IU mientras reproduce el sonido ya que ejecuta otro subproceso.

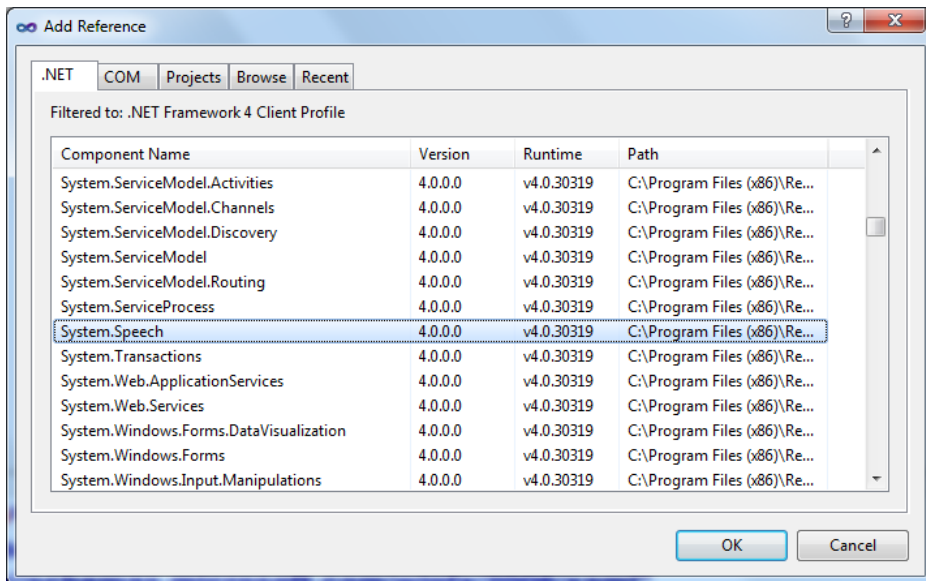
Esta clase también tiene propiedades como Volume que controla el volumen de la voz y va en el rango de enteros de 0 a 100, también podemos controlar la velocidad con que se pronuncia la voz usando la propiedad Rate que va en el rango de enteros de -10 a 10.

En resumen, es muy fácil incluir la reproducción de voz en nuestras aplicaciones, tal como se demuestra en el siguiente ejemplo en la cual se crea un reproductor de voz que pronuncia cualquier palabra o frase escrita y donde podemos controlar su velocidad y su volumen usando para ello los controles Slider de WPF.

Demo 100

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo100" y clic en "OK".
- Agregar una referencia a la librería de reconocimiento de voz: del menú "Project", seleccionar "Add References" y del diálogo elegir "System.Speech".

Gráfico 6.40: Diálogo de Referencia de la librería Speech



- Cambiar de nombre a la ventana "MainWindow.xaml" por "Reproductor Voz.xaml".
- Escribir el siguiente código en el archivo xaml:

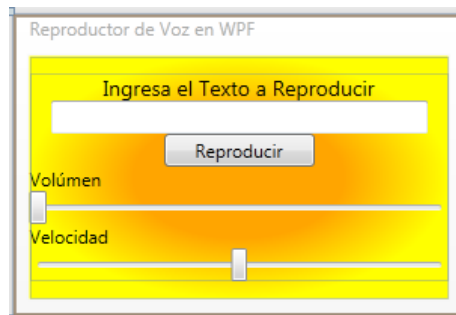
```
<Window x:Class="ReproductorVoz"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reproductor de Voz en WPF" Height="200" Width="300"
```

```
WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
<Grid>
  <Grid.Background>
    <RadialGradientBrush>
      <GradientStop Color="Yellow" Offset="1"/>
      <GradientStop Color="Orange" Offset="0.5"/>
    </RadialGradientBrush>
  </Grid.Background>
  <StackPanel VerticalAlignment="Center">
    <TextBlock Name="tbMensaje" Text="Ingresa el Texto a Reproducir"
      FontSize="14" HorizontalAlignment="Center"/>
    <TextBox Name="txtFrase" Width="250"/>
    <Button Name="btnReproducir" Content="Reproducir" Width="100"
      Cursor="Hand" ToolTip="Reproducir voz del texto escrito"/>
    <TextBlock Name="tbVolumen" Text="Volúmen"/>
    <Slider Name="sldVolumen" Minimum="0" Maximum="100"/>
    <TextBlock Name="tbVelocidad" Text="Velocidad"/>
    <Slider Name="sldVelocidad" Minimum="-10" Maximum="10"/>
  </StackPanel>
</Grid>
</Window>
```

Nota: En el código anterior se usa un Grid con fondo degradado en círculo que tiene un StackPanel el cual contiene un bloque de texto con un mensaje, un cuadro de texto para ingresar la frase, un botón para reproducir y 2 bloques de textos y barras de desplazamiento para cambiar el volumen y la velocidad de la voz.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.38.

Gráfico 6.41: Diseño de la ventana Reproductor de Voz



- Escribir código en el archivo vb asociado al xaml:

```
Imports System.Speech.Synthesis 'SpeechSynthesizer
```

```
Public Class ReproductorVoz
```

```
    Private oVoz As New SpeechSynthesizer
```

```
    Private Sub ReproducirVoz(ByVal sender As System.Object, _  
        ByVal e As System.Windows.RoutedEventArgs) Handles btnReproducir.Click  
        oVoz.SpeakAsync(txtFrase.Text)  
    End Sub
```

```
    Private Sub CambiarVolumen(ByVal sender As System.Object, _  
        ByVal e As System.Windows.RoutedPropertyChangedEventArgs(Of  
            System.Double)) _  
        Handles sldVolumen.ValueChanged  
        oVoz.Volume = sldVolumen.Value  
    End Sub
```

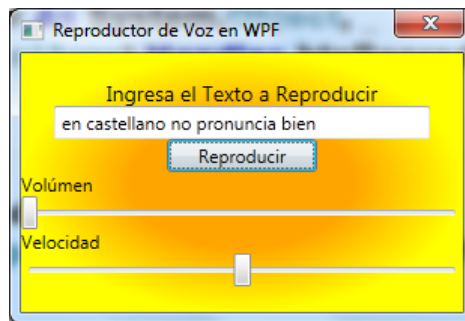
```
    Private Sub CambiarVelocidad(ByVal sender As System.Object, _  
        ByVal e As System.Windows.RoutedPropertyChangedEventArgs(Of  
            System.Double)) _  
        Handles sldVelocidad.ValueChanged  
        oVoz.Rate = sldVelocidad.Value  
    End Sub  
End Class
```

Nota: En el código anterior al dar clic en el botón "Reproducir" se llama al método `SpeakAsync` del objeto `SpeechSynthesizer` para reproducir lo

escrito en el cuadro de texto. También al arrastrar las barras de desplazamiento se cambia el volumen y la velocidad de la voz.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ReproductorVoz.xaml".
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.42: Ejecución de la ventana Reproductor de Voz



Nota: Cuando pruebe la reproducción notará que el traductor no pronuncia tan bien el castellano y mejor lo hace en inglés.

4.3. Trabajando con Audio y Video

Agregar objetos multimedia a una aplicación es tan simple como agregar un control `MediaElement` a la user interface (UI) de la aplicación y proporcionar una clase `Uri` a los objetos multimedia que se desea incluir. Todos los tipos de objetos multimedia que admite Microsoft Windows Media Player 10 se admiten en WPF.

Las propiedades `LoadedBehavior` y `UnloadedBehavior` controlan el comportamiento de `MediaElement` cuando `IsLoaded` es `true` o `false`, respectivamente. La enumeración `MediaState` que establece el estado de las propiedades afecta al comportamiento de reproducción multimedia.

Por ejemplo, el valor predeterminado de `LoadedBehavior` es `Play` y el valor predeterminado de `UnloadedBehavior` es `Close`. Esto significa que, tan pronto como se carga `MediaElement` y se completa el preprocesamiento, el objeto multimedia empieza a reproducirse. Cuando se completa la reproducción, se cierra el objeto multimedia y se liberan todos los recursos multimedia.

Para especificar el archivo de sonido o video a reproducir hay que configurar la propiedad `Source` del control `MediaElement` asignando la URI con el archivo a reproducir.

Para reproducir el audio o video debemos usar el método `Play` del control `MediaElement` e inmediatamente se desencadenará el evento `MediaOpened` y cuando finaliza la reproducción ocurrirá el evento `MediaEnded`.

En el primer evento (`MediaOpened`) podemos determinar la duración del audio o el video usando la propiedad `NaturalDuration` y en el último evento (`MediaEnded`) podemos volver a iniciar la reproducción si se necesita.

En el siguiente ejemplo se muestra como reproducir una lista de archivos de sonido y en el otro ejemplo se muestra como reproducir video mediante el control `MediaElement`. Además en el último ejemplo se hace uso de un reloj o `Timer` para WPF usando la clase `DispatcherTimer`.

Demo 101

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo101" y clic en "OK".
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Reproductor Sonido.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ReproductorSonido"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reproductor de Sonido en WPF" Height="420" Width="420"
    ResizeMode="NoResize" WindowStartupLocation="CenterScreen">
    <Grid Name="grdFondo">
        <StackPanel>
            <StackPanel.Background>
                <LinearGradientBrush>
                    <GradientStop Color="Aqua" Offset="1"/>
                    <GradientStop Color="LightBlue" Offset="0.5"/>
                </LinearGradientBrush>
            </StackPanel.Background>
            <Label Name="lblTitulo" HorizontalAlignment="Center"
                FontSize="14">Selecciona archivos de sonido a reproducir</Label>
            <DockPanel VerticalAlignment="Center"
                HorizontalAlignment="Center">
                <TextBox Name="txtArchivo" Width="350"/>
                <Button Name="btnAbrir" Content="..." Width="25"
                    Cursor="Hand" ToolTip="Abrir archivo de sonido"/>
            </DockPanel>
            <ListBox Name="lstArchivo" Height="300">
                <ListBox.Background>
                    <RadialGradientBrush>
                        <GradientStop Color="Yellow" Offset="1"/>
                        <GradientStop Color="Orange" Offset="0.5"/>
                    </RadialGradientBrush>
                </ListBox.Background>
            </ListBox>
        </StackPanel>
    </Grid>
</Window>
```

```
</RadialGradientBrush>
</ListBox.Background>
</ListBox>
<Button Name="btnReproducir" Content="Reproducir"
  Cursor="Hand" Width="100" Height="25" IsEnabled="False" />
<MediaElement Name="meSonido" LoadedBehavior="Manual" />
</StackPanel>
</Grid>
</Window>
```

Nota: En el código anterior se usa un Grid que tiene un StackPanel con fondo degradado el cual contiene un bloque de texto con un mensaje, un DockPanel que tiene un cuadro de texto para el archivo seleccionado y un botón para el diálogo de abrir, una lista para los archivos seleccionados de audio la cual esta degradada en círculo, un botón para reproducir la lista y finalmente un MediaElement para realizar la reproducción.

Importante: Es necesario iniciar la propiedad LoadBehaviour del control MediaElement en Manual sino tratará de reproducir el video y ocurrirá error.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.43.

Gráfico 6.43: Diseño de la ventana Reproductor de Sonido



- Escribir código en el archivo vb asociado al xaml:

```
Imports Microsoft.Win32 'OpenFileDialog
```

```
Class ReproductorSonido
```

```
    Private C As Integer
```

```
    Private Sub AgregarArchivoSonido(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) _
```

```
Handles btnAbrir.Click
```

```
    Dim ofd As New OpenFileDialog
```

```
    ofd.Title = "Selecciona los archivos de sonido"
```

```
    ofd.Filter = "Archivos de sonido|*.mp3;*.wav"
```

```
    If ofd.ShowDialog = True Then
```

```
        txtArchivo.Text = ofd.FileName
```

```
        btnReproducir.IsEnabled = True
```

```
        lstArchivo.Items.Add(ofd.FileName)
```

```
    End If
```

```
End Sub
```

```
    Private Sub ReproducirLista(ByVal sender As System.Object, _  
ByVal e As System.Windows.RoutedEventArgs) _
```

```
Handles btnReproducir.Click
```

```
    lstArchivo.Focus()
```

```
    lstArchivo.SelectedIndex = 0
```

```
    meSonido.Source = New Uri(lstArchivo.Items(C))
```

```
    meSonido.Play()
```

```
End Sub
```

```
    Private Sub IniciarLista(ByVal sender As Object, _  
ByVal e As System.Windows.RoutedEventArgs) _
```

```
Handles meSonido.MediaOpened
```

```
    Dim duracion As TimeSpan = _
```

```
    meSonido.NaturalDuration.TimeSpan
```

```
    Me.Title = String.Format("Duracion: {0}:{1}", _
```

```
    duracion.Minutes, duracion.Seconds)
```

```
End Sub
```

```
    Private Sub FinalizarLista(ByVal sender As Object, _  
ByVal e As System.Windows.RoutedEventArgs) _
```

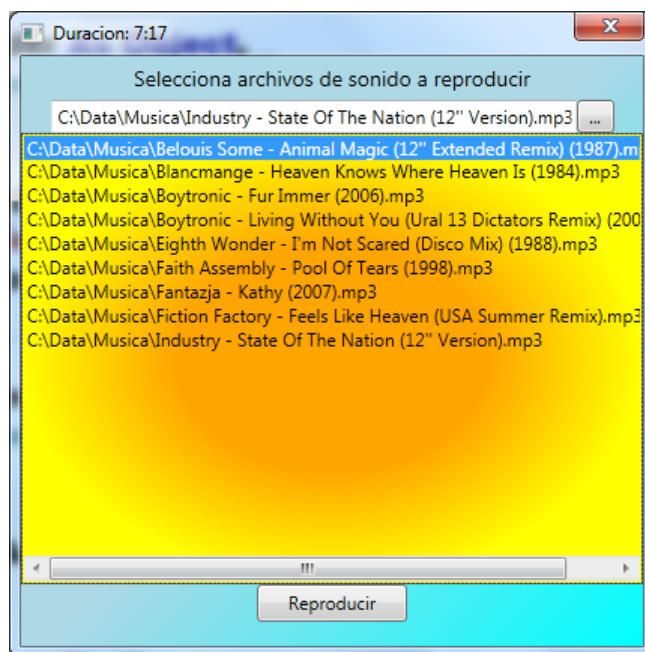
```
Handles meSonido.MediaEnded
```

```
C += 1
meSonido.Source = New Uri(lstArchivo.Items(C))
meSonido.Play()
lstArchivo.SelectedIndex = C
If C = lstArchivo.Items.Count Then C = 0
End Sub
End Class
```

Nota: En el código anterior al dar clic en el botón de “Abrir” se mostrará el diálogo con archivos mp3 y wav, al seleccionarlo se agrega a la lista; después de tener la lista completa al dar clic en el botón “Reproducir” se inicia la reproducción del primer archivo de la lista y al finalizar el audio en el evento MediaEnded se apunta al siguiente archivo de la lista para reproducirlo, si se termina toda la lista se inicia nuevamente en el primero.

- Regresar a la ventana “Application.xaml” y cambiar la ventana que inicia (atributo StartupUri) a “ReproductorSonido.xaml”.
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.44: Ejecución de la ventana Reproductor de Sonido



Demo 102

- Crear una nueva aplicación Windows WPF: del menú "File", seleccionar "New Project".
- En el diálogo elegir "Visual Basic", luego "Windows", seleccionar "WPF Application".
- Escribir en el nombre de la aplicación "Demo102" y clic en "OK".
- Cambiar de nombre a la ventana "MainWindow.xaml" por "Reproductor Video.xaml".
- Escribir el siguiente código en el archivo xaml:

```
<Window x:Class="ReproductorVideo"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reproductor de Video en WPF" Height="420" Width="420"
    ResizeMode="NoResize" WindowStartupLocation="CenterScreen">
    <Grid Name="grdFondo">
        <Grid.Background>
            <LinearGradientBrush>
                <GradientStop Color="aqua" Offset="0.8"/>
                <GradientStop Color="blue" Offset="0.4"/>
            </LinearGradientBrush>
        </Grid.Background>
        <StackPanel>
            <Label Name="lblTitulo" HorizontalAlignment="Center"
                Foreground="White" FontSize="14">
                Selecciona archivo de video</Label>
            <DockPanel VerticalAlignment="Center"
                HorizontalAlignment="Center">
                <TextBox Name="txtArchivo" Width="350"/>
                <Button Name="btnAbrir" Content="..." Width="25"
                    Cursor="Hand" ToolTip="Abrir archivo de video"/>
            </DockPanel>
            <Button Name="btnReproducir" Content="Reproducir"
                Cursor="Hand" Width="100" Height="25" IsEnabled="False" />
            <MediaElement Name="meSonido" Width="350" Height="220"
                LoadedBehavior="Manual" />
        </StackPanel>
    </Grid>
</Window>
```

```
<Label Name="lblTiempo" HorizontalAlignment="Center">
    00:00</Label>
<Slider Name="sldDuracion" Minimum="0" Maximum="100"/>
</StackPanel>
</Grid>
</Window>
```

Nota: En el código anterior se usa un Grid que tiene un fondo degradado lineal el cual contiene un bloque de texto con un mensaje, un DockPanel que tiene un cuadro de texto para el archivo seleccionado y un botón para el diálogo de abrir, un botón para reproducir el video, un MediaElement para ver el video y una etiqueta y una barra de desplazamiento para avanzar el video.

- El diseño de la ventana debe quedar como se ve en el gráfico 6.45.

Gráfico 6.45: Diseño de la ventana Reproductor de Video



- Escribir código en el archivo vb asociado al xaml:

```
Imports Microsoft.Win32 'OpenFileDialog
Imports System.Windows.Threading 'DispatcherTimer
```

Class ReproductorVideo

Private duracion As TimeSpan

Private X, C As Integer

Private WithEvents tmrDuracion As New DispatcherTimer

Private Min, Sec As Integer

Private Sub MostrarDialogoAbrir(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs) Handles btnAbrir.Click

Dim ofd As New OpenFileDialog

ofd.Title = "Selecciona archivo de video"

ofd.Filter = "Archivos de video|*.mpg;*.mpeg;*.avi;*.mp4;*.wmv"

If ofd.ShowDialog = True Then

txtArchivo.Text = ofd.FileName

btnReproducir.IsEnabled = True

End If

End Sub

Private Sub ReproducirVideo(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs) _
Handles btnReproducir.Click

meSonido.Source = New Uri(txtArchivo.Text)

meSonido.Play()

End Sub

Private Sub FinalizarReproduccion(ByVal sender As Object, _
ByVal e As System.Windows.RoutedEventArgs) _
Handles meSonido.MediaEnded

Min = 0

Sec = 0

sldDuracion.Value = 0

tmrDuracion.Stop()

tmrDuracion.IsEnabled = False

End Sub

Private Sub MostrarTamañoVideo(ByVal sender As Object, _
ByVal e As System.Windows.RoutedEventArgs) _
Handles meSonido.MediaOpened

tmrDuracion.Interval = New TimeSpan(0, 0, 1)

tmrDuracion.IsEnabled = True

tmrDuracion.Start()

```
duracion = meSonido.NaturalDuration.TimeSpan
X = duracion.TotalSeconds \ 100
Me.Title = String.Format("Duracion Video (mm:ss) = {0}:{1}", _
duracion.Minutes.ToString.PadLeft(2, "0"), _
duracion.Seconds.ToString.PadLeft(2, "0"))
End Sub
```

```
Private Sub MostrarProgreso(ByVal sender As Object, _
ByVal e As EventArgs) Handles tmrDuracion.Tick
    C = C + 1
    If C Mod 60 = 0 Then
        Min += 1
        Sec = 0
    Else
        Sec += 1
    End If
    lblTiempo.Content = String.Format("{0}:{1}", _
Min.ToString.PadLeft(2, "0"), Sec.ToString.PadLeft(2, "0"))
    If C Mod X = 0 Then
        sldDuracion.Value += 1
    End If
End Sub
```

```
Private Sub ModificarAvance(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedPropertyChangedEventArgs(Of
System.Double)) _
Handles sldDuracion.ValueChanged
    Dim seg As Integer = (sldDuracion.Value * duracion.TotalSeconds) / 100
    If sldDuracion.Value < 100 Then
        meSonido.Position = New TimeSpan(0, 0, seg)
        lblTiempo.Content = String.Format("{0}:{1}", _
meSonido.Position.Minutes.ToString.PadLeft(2, "0"), _
meSonido.Position.Seconds.ToString.PadLeft(2, "0"))
    Else
        sldDuracion.Value = 0
        tmrDuracion.Stop()
        tmrDuracion.IsEnabled = False
    End If
End Sub
End Class
```

Nota: En el código anterior al dar clic en el botón de "Abrir" se mostrará el diálogo con archivos de video, al dar clic en el botón "Reproducir" se inicia la reproducción del video y se dispara el evento MediaOpened en el cual se habilita un Timer y se muestra el tiempo de duración. El timer muestra el tiempo de avance del video. Si se avanza o retrocede con la barra de desplazamiento se actualiza la duración actual.

Importante: Para trabajar con procesos en segundo plano cada cierto tiempo en WPF no se debe usar el Timer de System.Timers sino el DispatcherTimer de System.Windows.Threading.

- Regresar a la ventana "Application.xaml" y cambiar la ventana que inicia (atributo StartupUri) a "ReproductorVideo.xaml".
- Grabar y ejecutar la aplicación creada pulsando la tecla F5.

Gráfico 6.46: Ejecución de la ventana Reproductor de Video



Preguntas de Repaso

1. Qué se puede hacer en WPF que no se puede hacer en WinForms no ASP .NET?
2. Cuáles son los principales componentes en la arquitectura de WPF?
3. Menciona 5 características de WPF.
4. Qué lenguaje de marcas usa WPF?
5. Cuántos tipos de aplicaciones se pueden crear en WPF?
6. Menciona 5 tipos de controles de WPF.
7. Cómo se llama el contenedor principal de controles que viene por defecto en una ventana o página en WPF?
8. Menciona otros 3 contenedores de controles WPF.
9. Menciona el evento de inicio que ocurre al cargar una ventana y los eventos de cierre de ventana.
10. Qué debe configurarse para que una cierta ventana de nuestra aplicación inicie primero?
11. Qué es una aplicación WPF del Explorador o XBAP?
12. Que tipo de seguridad tiene por defecto una aplicación WPF del explorador?
13. Qué clase se usa para navegar entre páginas de una aplicación del explorador?
14. Qué se debe hacer para que una aplicación del explorador pueda acceder a archivos, cuadros de diálogo, registro de Windows, etc.?

15. Cómo se clasifican los cuadros de diálogo?
16. Qué tipos de cuadros de diálogo se pueden crear en WPF?
17. Qué espacio de nombres se debe importar para usar los diálogos de archivos de Windows?
- 18.Cuál es el único cuadro de diálogo interno de WPF?
19. Cómo se llama las clases para implementar los diálogos de abrir y guardar respectivamente?
20. Con qué método del cuadro de diálogo de imprimir (PrintDialog) se puede imprimir cualquier contenido inclusive gráficos?
21. Cuantas técnicas tenemos para presentar datos en una aplicación?
22. Cuáles son las ventajas principales de WPF en el enlace de datos con respecto a WinForms y WebForms?
23. Cuántos tipos de enlaces (Data Bindings) existen?
24. Qué propiedad de los controles de listas implementa el enlace complejo en WPF?
25. Menciona 3 controles que soporten enlace complejo.
26. Qué debe hacerse para crear enlace simple?
27. Menciona 3 controles que soporten enlace simple.
28. Cómo se llama la clase que hace de intermediario entre los controles y el origen de datos en WPF?
29. Menciona los métodos de desplazamiento del objeto CollectionView.

30. Con qué propiedades del CollectionView se verifican el desbordamiento de registro?
31. En qué evento del CollectionView podemos programar para mostrar la posición actual del origen de datos cuando este cambie?
32. Qué propiedades de los controles listas permiten configurar el campo a mostrar y el campo a guardar?
33. Para qué se usa la conversión de datos en WPF?
34. Qué interface tenemos que implementar para realizar la conversión de datos?
35. Qué métodos hay que programar en una clase que implemente dicha interface?
36. Cómo se aplica en el XAML una clase creada para formatear o convertir un tipo de dato en un control enlazado?
37. Cómo se puede ordenar las columnas en el control ListView?
38. Cómo se obtiene la cabecera a la cual se dio clic en un ListView?
39. Cómo se crea un objeto CollectionView?
40. Para qué se usan las plantillas de datos en WPF?
41. Qué tipo de objeto permite crear una plantilla de datos y qué propiedad del control enlazado debemos configurar para aplicar el objeto creado?
42. Cómo se sincronizan 2 listas enlazadas a datos sin necesidad de programar el filtro?
43. Qué es una plantilla jerárquica y cómo se implementa en WPF?
44. Qué control de WPF soporta una plantilla jerárquica?

45. Qué requisitos debe tener el origen de datos antes de enlazar a un control que usará una plantilla jerárquica?
46. Qué control WPF presenta de forma simple filas y columnas sin necesidad de usar plantillas?
47. Cuáles son los tipos de columnas que puede tener el control DataGrid?
48. Qué es necesario hacer al DataGrid para que pueda ordenar los datos al dar clic a la cabeceras de sus columnas?
49. Cuántos tipos de documentos maneja WPF?
50. Qué control se usa para visualizar documentos fijos como los XPS?
51. Qué controles se usan para visualizar documentos dinámicos como los archivos de texto, xml, html, etc.
52. Qué significa XPS y cómo se implementa en WPF?
53. Cómo se llama la librería que permite trabajar con documentos XPS?
54. Qué propiedad del control DocumentViewer es necesario configurar para enlazar un documento?
55. Menciona los 3 tipos de anotaciones que hay en WPF.
56. Qué clase es necesario usar para implementar anotaciones en WPF?
57. Menciona 3 métodos de la clase AnnotationService que sirvan para crear anotaciones.
58. Menciona 3 métodos de la clase AnnotationService que sirvan para eliminar o borrar anotaciones.
- 59.Cuál es la ventaja de los documentos dinámicos?

- 60. Cómo se puede convertir un archivo de Word o Excel a XPS?
- 61. Qué se debe hacer si se usa por código trabajar con Word o Excel desde .NET?
- 62. Cuáles son las 2 clases que implementan Multimedia en WPF?
- 63. Cómo se llama la librería de clases que implementa el reconocimiento de voz en WPF?
- 64. Cómo se llama la clase que permite reproducir voz?
- 65. Qué métodos tiene la clase SpeechSynthesizer para reproducir la voz?
- 66. Qué propiedades del objeto SpeechSynthesizer permiten controlar el volumen y la velocidad de la voz?
- 67. Cómo se llama la clase (control) que permite reproducir audio y video?
- 68. Qué propiedad del control MediaElement especifica la URI con el archivo de audio o video a reproducir?
- 69. Qué eventos ocurren al iniciar y finalizar la reproducción usando el MediaElement?
- 70. Qué clase y en que espacio de nombres se encuentra el objeto que permite ejecutar en segundo plano cada cierto tiempo una acción en una aplicación WPD?
- 71. Ultima pregunta: Te gusto o no el libro?

Índice de Ejemplos del Libro

Demo 01: Información del Sistema

Demo 02: Información de Directorios Especiales de Windows

Demo 03: Explorador del Registro de Windows (RegEdit)

Demo 04: Administrador de Tareas

Demo 05: Explorador de Archivos V1.0

Demo 06: Chat

Demo 07: Dividir y Unir Archivos

Demo 08: Comprimir y Descomprimir Archivos

Demo 09: Cálculo del Área del Triángulo sin Librerías

Demo 10: Librería de Código de Usuario para Triángulo

Demo 11: Librería de Entidad de Negocio para Triángulo

Demo 12: Librería de Acceso a Datos para Triángulo

Demo 13: Librería de Reglas de Negocio para Triángulo

Demo 14: Cálculo del Área del Triángulo con Librerías

Demo 15: Lectura asíncrona de archivos de texto

Demo 16: Lectura asíncrona de archivos de texto con Barra de Progreso

Demo 17: Contador de palabras en archivos asíncrono

Demo 18: Servicio WCF que lista Empleados

Demo 19: Aplicación que usa el Servicio WCF que lista empleados en forma asíncrona con Eventos

Demo 20: Librería de acceso a datos que lista empleados

Demo 21: Aplicación que usa la Librería que lista empleados en forma asíncrona con Delegados CallBack.

Demo 22: Gráfico de Círculos usando For Paralelo

Demo 23: Lista de Empleados usando Tarea Paralela

Demo 24: Cifrado Simétrico por Caracter

Demo 25: Cifrado Simétrico por Bloques

Demo 26: Verificar archivos usando valores Hash

Demo 27: Conectarse a una base de datos de MS SQL Server

Demo 28: Visor de bases de datos de MS Access

Demo 29: Visor de archivos de MS Excel

Demo 30: Conectarse a una archivo DBF

Demo 31: Consulta de empleados por código que muestra nombre

Demo 32: Consulta de empleados por código que muestra todos sus datos

Demo 33: Consulta de productos por proveedor

Demo 34: Consulta de varias tablas

Demo 35: Mantenimiento conectado de empleados

Demo 36: Consulta de ordenes por cliente

Demo 37: Filtro de productos por código y nombre

Demo 38: Búsqueda de productos por código y nombre

Demo 39: Mantenimiento desconectado de empleados

Northwind.LibBusinessEntities: Librería de Entidades de Negocio Northwind

Northwind.LibDataAccess: Librería de Acceso a Datos Northwind

Northwind.LibBusinessRules: Librería de Reglas del Negocio Northwind

Demo 40: Lista de Empleados con objetos que usa Librerías Northwind

Demo 41: Lista de Productos con objetos que usa Librerías Northwind

Demo 42: Filtro en Lista de objetos empleados con For - If

Demo 43: Filtro en Lista de objetos empleados con predicados

Demo 44: Filtro de productos por categoría con predicados

Demo 45: Ordenación y búsqueda de objetos para empleados

Demo 46: Mantenimiento de empleados con objetos

Demo 47: Filtro de empleados con LINQ a DataSet

Demo 48: Filtro de productos con LINQ a DataSet

Demo 49: Filtro de ordenes por cliente con LINQ a SQL

Demo 50: Filtro de productos por proveedor con LINQ a Entidades

Demo 51: Formulario en forma de elipse con gráficos

Demo 52: Formulario de Login con controles básicos

Demo 53: Manejo de Listas en Windows Forms

Demo 54: Explorador de Windows V2.0

Demo 55: Formulario MDI con gráfico de fondo

Demo 56: Crear Menú Principal dinámicamente desde un archivo XML

Demo 57: Crear Menú Contextual dinámicamente con un Calendario

Demo 58: Marquesina configurable

Demo 59: Editor de Documentos de Texto de Formato Enriquecido

Demo 60: Columnas Personalizadas en lista de Productos

Demo 61: Lista de Categorías con Imágenes desde Archivos

Demo 62: Personalizar Cabeceras para Filtrar Productos por Categoría

Demo 63: Gráfico de Barras de Precios de Productos

Demo 64: Paginación en una Lista de Productos

General.LibControlsWinForms: Librería de Controles de Windows Forms

Demo 65: Aplicación de Prueba de Librería de Controles Windows

Demo 66: Reporte de Productos con PrintDocument

Demo 67: Reporte de Empleados con Informes Microsoft

Demo 68: Crear Documentos de Productos con MS Word

Demo 69: Crear Informes y Gráfico de Productos con MS Excel

Demo 70: Crear Gráfico de Productos con el Control Chart

Demo 71: Simple sitio web para navegar por 3 páginas

Demo 72: Ficha de Registro de Alumnos en archivos de texto

Demo 73: Seguridad usando un CAPTCHA

Demo 74: Bolsa de Trabajo con control FileUpload y controles de Validación

Demo 75: Ficha de Registro de Alumnos usando Hojas de Estilos

Demo 76: Sitio Web de un Instituto que usa Página Principal, Páginas de Contenido y Controles de Navegación

Demo 77: Consultas de Datos usando fichas con controles de vistas
MultiView y Views

Demo 78: Lista de Productos con Columnas Personalizadas

Demo 79: Lista de Productos Paginada

Demo 80: Lista de Productos Paginada y Ordenada con Símbolos de Orden

Demo 81: Mantenimiento de Productos en el GridView

Demo 82: Lista de Categorías usando Plantillas en el control Repeater

Demo 83: Lista de Empleados usando Plantillas en el control DataList

Demo 84: Lista de Productos x Categoría usando Plantillas Jerárquicas

Demo 85: Aplicación WPF Windows que Registra y Lista Alumnos

Demo 86: Aplicación WPF para el Browser XAML (XBAP) de Alumnos

Demo 87: Editor de Textos WPF con Diálogos Comunes de Windows

Demo 88: Leer y Escribir en Lista de Empleados

Demo 89: Leer y Escribir en Consulta de Empleados

Demo 90: Enlace de Datos en Lista de Productos

Demo 91: Enlace de Datos en Consulta de Productos

Demo 92: Conversión de Datos y Ordenación en un ListView de Productos

Demo 93: Conversión de Datos en una Consulta de Empleados

Demo 94: Plantillas de Datos y Sincronización de Listas en Consulta de Productos por Categoría

Demo 95: Plantilla de Datos Jerárquica en TreeView lista Detalles x Orden

Demo 96: Lista de Productos usando el control DataGrid

Demo 97: Visor de Documentos XPS como Documento Fijo

Demo 98: Visor de Archivos de Word y Excel con Anotaciones

Demo 99: Visor de Archivos de Texto como Documento Dinámico

Demo 100: Reproductor de Voz

Demo 101: Reproductor de Sonido

Demo 102: Reproductor de Video

Enlaces de Referencias

1. Información general y conceptual sobre .NET Framework

<http://msdn.microsoft.com/es-es/library/zw4w595w.aspx>

2. Lenguajes .NET

<http://www.dotnetpowered.com/languages.aspx>

3. Bibliotecas de clases de .NET Framework

<http://msdn.microsoft.com/es-es/library/gg145045.aspx>

4. Common Language Runtime (CLR)

<http://msdn.microsoft.com/es-es/library/8bs2ecf4.aspx>

5. Lo Nuevo en .NET Framework 4

<http://msdn.microsoft.com/es-es/library/ms171868.aspx>

6. Paseo rápido por el entorno de desarrollo integrado (IDE)

<http://msdn.microsoft.com/es-es/library/ms165088.aspx>

7. Novedades de Visual Studio 2010

<http://msdn.microsoft.com/es-es/library/bb386063.aspx>

8. Estructuras de Control de Flujo

<http://msdn.microsoft.com/es-es/library/ca8tdhcs.aspx>

9. Tipos de Datos en Visual Basic

<http://msdn.microsoft.com/es-es/library/ffsdktda.aspx>

10. Operadores en Visual Basic

<http://msdn.microsoft.com/es-es/library/a1w3te48.aspx>

11. Variables en Visual Basic

<http://msdn.microsoft.com/es-es/library/cd6hcy37.aspx>

12. Novedades de Visual Basic 2010

<http://msdn.microsoft.com/es-es/library/we86c8x2.aspx>

13. Entrada y Salida de archivos básica

<http://msdn.microsoft.com/es-es/library/336wast5.aspx>

14. Objetos y Clases de Visual Basic

<http://msdn.microsoft.com/es-es/library/527aztek.aspx>

15. Información general sobre la programación asíncrona

<http://msdn.microsoft.com/es-es/library/ms228963.aspx>

16. Programación paralela en .NET Framework

<http://msdn.microsoft.com/es-es/library/dd460693.aspx>

17. Servicios criptográficos

<http://msdn.microsoft.com/es-es/library/92f9ye3s.aspx>

18. Información general sobre ADO .NET

<http://msdn.microsoft.com/es-es/library/h43ks021.aspx>

19. Descargar archivo de Ubigeo 2008 del INEI

<http://iinei.inei.gob.pe/iinei/siscodes/datos/UBIgeo2008.ZIP>

20. Información sobre cadenas de conexión

<http://www.connectionstrings.com/>

21. DataSets de ADO .NET

<http://msdn.microsoft.com/es-es/library/zb0sdh0b.aspx>

22. LINQ y ADO .NET

<http://msdn.microsoft.com/es-es/library/bb399365.aspx>

23. Introducción a los formularios Windows Forms

<http://msdn.microsoft.com/es-es/library/ms229601.aspx>

24. Información General del Control DataGridView

[http://msdn.microsoft.com/es-es/library/k39d6s23\(v=vs.80\).aspx](http://msdn.microsoft.com/es-es/library/k39d6s23(v=vs.80).aspx)

25. Desarrollar controles personalizados de formularios Windows Forms

<http://msdn.microsoft.com/es-es/library/6hws6h2t.aspx>

26. Funcionalidad para imprimir en formularios Windows Forms

<http://msdn.microsoft.com/es-es/library/xdt36c58.aspx>

27. Informes de Microsoft

<http://msdn.microsoft.com/es-es/library/bb885185.aspx>

28. Control ReportViewer

<http://www.gotreportviewer.com/>

29. Control Chart

<http://msdn.microsoft.com/es-es/library/system.windows.forms.data.visualization.charting.chart.aspx>

30. Información general sobre ASP .NET

<http://msdn.microsoft.com/es-es/library/4w3ex9c2.aspx>

31. Clase FileUpload

[http://msdn.microsoft.com/es-es/library/system.web.ui.webcontrols.fileupload\(v=vs.80\).aspx](http://msdn.microsoft.com/es-es/library/system.web.ui.webcontrols.fileupload(v=vs.80).aspx)

32. Controles de Validación ASP .NET

[http://msdn.microsoft.com/es-es/library/debza5t0\(v=vs.80\).aspx](http://msdn.microsoft.com/es-es/library/debza5t0(v=vs.80).aspx)

33. Páginas Principales en ASP .NET

[http://msdn.microsoft.com/es-es/library/18sc7456\(v=vs.90\).aspx](http://msdn.microsoft.com/es-es/library/18sc7456(v=vs.90).aspx)

34. Clase GridView

<http://msdn.microsoft.com/es-es/library/system.web.ui.webcontrols.gridview.aspx>

35. Plantillas de Controles de Servidor Web ASP .NET

<http://msdn.microsoft.com/es-es/library/h59db326.aspx>

36. Arquitectura de WPF

<http://msdn.microsoft.com/es-es/library/ms750441.aspx>

37. Información general sobre ventanas de WPF

<http://msdn.microsoft.com/es-es/library/ms748948.aspx>

38. Información general sobre navegación

<http://msdn.microsoft.com/es-es/library/ms750478.aspx>

39. Información general sobre cuadros de diálogo

<http://msdn.microsoft.com/es-es/library/aa969773.aspx>

40. Información general sobre el enlace de datos

<http://msdn.microsoft.com/es-es/library/ms752347.aspx>

41. Información general sobre plantillas de datos

<http://msdn.microsoft.com/es-es/library/ms742521.aspx>

42. Clase DataGridView de WPF

<http://msdn.microsoft.com/es-es/library/system.windows.controls.datagrid.aspx>

43. Documentos en WPF

<http://msdn.microsoft.com/es-es/library/ms748388.aspx>

44. Información general sobre anotaciones

<http://msdn.microsoft.com/es-es/library/ms748864.aspx>

45. Información general sobre Multimedia

<http://msdn.microsoft.com/es-es/library/aa970915.aspx>