UNIVERSIDAD LASALLISTA BENAVENTE

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

Con estudios incorporados a la

Universidad Nacional Autónoma de México

CLAVE: 8793-16

"INTRODUCCIÓN A LA TECNOLOGÍA ASP.NET"

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

HÉCTOR FLORES ACEVEDO

ASESORA: **ING. MAYA GISELA VILLAGÓMEZ TORRES**

Celaya, Gto. **Junio de 2003**

Agradecimientos personales

A Dios, por haberme dado la vida, la visión, el talento y señalarme el camino a seguir. A mis padres, Héctor Carlos Flores Herrera y Rosario Acevedo Mondragón por darme los valores y principios que tanto me han servido en la vida, como son la disciplina, la constancia, la responsabilidad y la tenacidad, ellos son y serán siempre mis maestros más importantes y mi orgullo más grande. A mi hermana y mi familia por estar siempre conmigo. A mi novia Rocío por darme la fuerza y el apoyo para seguir adelante. A todos mis amigos en especial a Heriberto Arreguín, Arturo Scotto, Rosa Gonzáles, Peter Kallai y muchos otros, por brindarme su invaluable amistad.

Agradecimientos académicos

A la Universidad Lasallista Benavente. Al director de la carrera de Ingeniería en Computación, el Ing. Miguel Ángel Jamaica Arreguín. A todos los maestros que me han acompañado desde el principio de mi educación, particularmente a todos aquellos que no solamente se conformaron con dar una clase, sino a dar ese *extra* que hizo la diferencia en mí para convertirme en el profesionista que soy ahora.

Quiero agradecer al Ing. Francisco Gutiérrez Vera del Tecnológico de Celaya, por ser la primera persona en inculcarme la programación de computadoras, motivándome a mí y a otros estudiantes a buscar siempre la calidad del software.

Al Ing. Alejandro Guzmán Zazueta por compartir sus valiosos comentarios, conocimientos y recursos, sin mayor recompensa que el aprendizaje de sus alumnos.

En especial, quiero agradecer a dos personas que sin ellas esta tesis jamás se hubiera llevado a cabo, a mis dos asesores: la Ing. Maya Gicela Villagómez Torres y la Lic. Araceli Lupercio Ramos, de todo corazón agradezco su paciencia y dedicación a mi trabajo.

A todos los autores de libros que han sido la fuente más importante de mis conocimientos; Herbert Schildt, Fco. Javier Ceballos, Erich R. Bühler, Jeffrey P. McManus, Scot Hillier, Rob Hawthorne, Stephen R. Davis, entre muchos otros.

Agradecimientos especiales

A Carlos Slim Helú, por ser la fuente más grande de inspiración para mí y para otros estudiantes, por medio de la Fundación Telmex pude asistir a varios eventos donde conocí a gente emprendedora como Carlos Slim Helú, Felipe Gonzáles, Bill Clinton, Ana Guevara, Hugo Sánchez, Albin Toffler, Magic Jonson, entre muchos otros.

Antes de conocer a Carlos Slim, uno queda impresionado por la capacidad empresarial que posee, pero después de conocerlo, uno queda más impresionado por el gran ser humano que es y lo mucho que quiere a México.

Carlos Slim Helú me enseñó, por medio de su ejemplo y sus palabras, que la clave del éxito está en el poder de elección, ***saber que sí y qué no***, en el trabajo duro, en la sencillez y en la familia.

A la Fundación Telmex le agradezco su apoyo incondicional, el trabajo en la formación de líderes es invaluable para México, la mayor aportación que la fundación hizo para mí, fue el darme la motivación y la visión para desarrollar todo el potencial que hay en mí, además de recordarme la gran responsabilidad que tengo con mi país.

Agradezco a la Universidad Nacional Autónoma de México, la máxima casa de estudios, por apoyarme con la beca que me permitió continuar mis estudios sin problemas económicos. Estoy seguro que algún día seré uno de los muchos orgullos de esta gran institución, cuando uno es becario se tiene una responsabilidad muy grande con México, no los defraudaré.

En caso de omitir a alguien espero que me disculpe, el apoyo de muchas personas fué necesario para poder llegar a la culminación de mi carrera.

A todos ellos, muchas gracias.

Héctor Flores Acevedo

Índice

Introducción

Anteriormente, el tener una aplicación en red que pudiera acceder a una o varias bases de datos, implicaba tener un programa servidor y varios programas cliente instalados en cada computadora. Aunque este método funcionaba, la escalabilidad estaba muy limitada ya que para hacer algún cambio en el programa se tenían que actualizar todas las computadoras cliente; imagine que la aplicación estuviera instalada en 50 computadoras o más.

ASP.NET es la versión mejorada de ASP ó páginas activas de servidor. Esta tecnología tiene la capacidad de utilizar el procesador del servidor, para realizar operaciones solicitadas por cualquier aplicación cliente.

ASP.NET utiliza un navegador de Internet como Microsoft Internet Explorer o Netscape Navigator como interfaz de usuario, además de ser muy escalable, ya que se pueden hacer cambios en la aplicación residente en el servidor y cualquier computadora cliente puede ver los cambios en forma inmediata. Se puede utilizar el software existente debido a que la mayoría de las computadoras tienen la capacidad de poder entrar a Internet y por lo tanto tienen algún tipo de navegador de Internet; se pueden ocupar esos equipos para poder acceder a la aplicación que se encuentra en el servidor.

La facilidad para el cliente de poder acceder en forma remota a datos y aplicaciones residentes en un servidor abre un mundo de posibilidades.

El objetivo de este estudio es *introducir al lector en la tecnología ASP.NET y determinar cómo puede simplificar el desarrollo de aplicaciones en Internet, además de darle las bases para realizar sus propias soluciones informáticas.*

ASP.NET comienza a ser utilizado por los desarrolladores en sitios como: USA Today (www.usatoday.com), Time Warner Telecom (www.twtelecom.com/default.aspx), Dell host (www.dellhost.com), Microsoft (www.microsoft.com), American Express (www.americanexpress.com), jetblue Airways (www.jetblue.com), Dollar rent a car (www.dollar.com), 2003 Sundance Film Festival (festival.sundance.org), BICSI (www.bicsi.org), etc.

ASP.NET es totalmente compatible con ASP, sitios que actualmente utilicen esta tecnología tenderán a migrar al nuevo modelo. Algunos ejemplos de estas empresas y

organizaciones son: Telefonos de México USA (www.telmexusa.com), Motorola México (www.motorola.com.mx), la Secretaría de Hacienda (www.shcp.gob.mx), el Banco de México (www.banxico.org.mx), la sección amarilla (www.seccionamarilla.com.mx), NASDAQ (www.nasdaq.com), Monster (www.monster.com), etc.

En este estudio utilizo la tecnología de Microsoft basada en Windows 2000 Server SP2, Infraestructura .NET, Microsoft SQL Server 2000 SP2, Internet Information Server (IIS) 5.0, ASP.NET, Visual Basic.NET, Visual Studio.NET, ADO.NET.

Es importante mencionar que para el desarrollo de aplicaciones Web, es posible utilizar otras dos opciones además de Visual Studio.NET, la primera es utilizar Visual Basic.NET Standard Edition, que requiere Windows 2000 Profesional o Windows XP Profesional, la segunda es la versión gratuita de Microsoft para el desarrollo de aplicaciones Web, ASP.NET Web Matrix Project, se puede descargar de la página (www.asp.net), requiere Windows 2000 Profesional o Windows XP Home/Profesional, la aplicación incluye un servidor Web, por lo que no necesita IIS. Personalmente recomiendo utilizar Visual Studio.NET, ya que los ejercicios están basados en esa aplicación, aunque es posible utilizar las otras opciones con algunas diferencias en el ambiente de desarrollo.

En el primer capítulo se da una introducción a la plataforma .NET, el objetivo es que el lector conozca las características de la infraestructura .NET y las ventajas que trae con respecto al modelo anterior. Se explica el por qué del nacimiento de la infraestructura .NET, cómo se resuelven los problemas que existían con el modelo de componentes (COM), se describen algunas de las clases más importantes de .NET, en general se comentan las nuevas características de la infraestructura .NET, para que el lector se familiarice con lo términos.

En el segundo capítulo se dan algunos antecedentes sobre la evolución de las páginas Web. El objetivo de este capítulo es que el lector se familiarice con las funciones y propiedades de las páginas activas de servidor o ASP, además de darse cuenta de todas las ventajas que le ofrece esta tecnología, como los controles enlazables, la separación de interfaz y código, seguridad integrada con Windows 2000 Server, etc. Una parte muy importante, es donde se hace mención de las variables de aplicación y de sesión, debido a que la programación en Internet es diferente respecto a la programación para una PC, se debe tener cuidado en el almacenamiento de datos o configuraciones especiales, ya que cada vez que se genera un evento, la página

completa se regenera, de allí la importancia de este tipo de variables. En este capítulo se da una introducción a la estructura general de ASP.NET, todos los archivos que lo conforman y su organización, se explican algunos de los eventos más importantes y para qué sirven.

El tercer capítulo está dedicado al modelo ADO.NET, con el objetivo de que el desarrollador se familiarice con su estructura y jerarquía, para que comprenda las ventajas que este nuevo modelo ofrece, y cómo puede aplicarlos en sus propios desarrollos. En este capítulo, se explican las diferencias entre el modelo de dos y tres capas, cuándo se debe de utilizar un modelo u otro, se describe cómo hacer una conexión con un origen de datos utilizando ADO.NET, realizar transacciones de datos, se explica el objeto *DataSet* que vino a sustituir el *RecordSet* de ADO; sin embargo, el primero tiene características muy por arriba de su antecesor, como la capacidad de guardar relaciones, restricciones, además de poder almacenar más de una tabla. Se dan varios ejemplos, pero quizás el más importante del capítulo es aquel en donde damos de alta, de baja y editamos valores de una base de datos.

El cuarto capítulo se refiere a los Servicios Web XML, su objetivo es dar a conocer sus principales características e informar al desarrollador de la importancia que estos servicios tienen en la industria del software. Desde mi punto de vista, los Servicios Web XML son lo más selecto de Visual Studio.NET. Pueden ser consumidos por cualquier lenguaje y sistema operativo porque el formato es XML, tienen infinidad de posibles usos en cualquier tipo de industria.

Durante bastante tiempo se ha tratado de publicar componentes en la Web sin mucho éxito; no obstante, los Servicios Web XML son componentes de software diseñados para esta tarea. Conforme se realicen los ejercicios de este capítulo, el desarrollador se dará cuenta de lo sencillo que es crear estos servicios, y de utilizarlos en aplicaciones. Es importante que el desarrollador se especialice en esta área, porque los Servicios Web XML es la primer tecnología que puede ser compartida por cualquier plataforma y lenguaje de programación; por lo tanto, los desarrolladores que trabajan en diferentes plataformas pueden consumir estos servicios sin mayor problema.

Espero que el lector disfrute este trabajo tanto como lo hice yo, es importante recalcar que esta tesis es solamente una introducción para el desarrollador de Visual Basic que desea aprovechar los beneficios de Internet. Existen otros excelentes libros

de los cuales el lector puede estudiar, espero que este trabajo sirva como pilar para el estudio de aplicaciones Web.

A quién va dirigida esta obra

A desarrolladores de software de nivel intermedio ó avanzado que utilicen la tecnología de Microsoft que necesiten desarrollar aplicaciones Web. A desarrolladores que actualmente utilizan ASP y que deseen migrar hacia ASP.NET. A diseñadores que utilicen tecnologías diferentes a las de Microsoft y que quieran conocer un punto de vista diferente en el desarrollo de aplicaciones Web. A todos aquellos que deseen saber sobre la infraestructura .NET y cómo se relaciona con ASP.NET.

Recomendaciones

Para sacar más provecho a este trabajo, es recomendable que el lector tenga conocimientos de programación, que haya trabajado previamente con Visual Basic 6.0, que tenga conocimientos de bases de datos en Microsoft SQL Server 2000 y Windows 2000 Server SP2. Este trabajo está enfocado a introducir las principales características de la tecnología ASP.NET con Visual Basic.NET, y cómo puede un desarrollador de nivel intermedio sacar el mayor provecho al desarrollo en Internet.

Requerimientos básicos

Los requerimientos de software pueden variar, se puede utilizar para el desarrollo de aplicaciones Web: ASP.NET Web Matrix Project, es gratuito y se puede descargar en la dirección http://www.asp.net, Visual Basic.NET Standard Edition ó Visual Studio.NET en sus diferentes versiones Academic, Professional, Enterprise Developer ó Enterprise Architect. La comparación de cada versión y su precio está en http://msdn.microsoft.com/vstudio/howtobuy/choosing.aspx. Recomiendo que el lector adquiera el siguiente software para realizar los ejercicios de esta obra.

Software requerido para desarrollar aplicaciones ASP.NET

- Windows 2000 Server SP2, Windows 2000 Professional SP2, Windows XP Profesional (recomiendo mucho) ó Windows Server 2003.
- Microsoft SQL Server 2000. La versión Enterprise Developer y Enterprise Architect de Visual Studio.NET 2003, trae incluída una versión de prueba de Microsoft SQL Server 2000.
- Visual Studio.NET, cualquier versión.
- .NET Framework. La infraestructura .NET ó .NET Framework se instala con Visual Studio.NET, en Windows Server 2003 ya viene incluída.
- Internet Information Server 5.0 ó superior. En Windows 2000 Server, IIS se configura automáticamente, en Windows XP Professional es necesario configurarlo.

Hardware requerido para el desarrollo de aplicaciones ASP.NET

- Procesador: Pentium II a 450 MHz. Se recomienda Pentium III a 600 MHz o superior.
- 160 MB en RAM. Se recomienda 256 MB en memoria RAM.
- Se recomiendan 6 GB disponibles en disco duro.
- Unidad de CD-ROM ó DVD-ROM.
- Super VGA (1024x768) ó mayor resolución con 256 colores
- Mouse

Unas palabras antes de comenzar

Espero que encuentre este trabajo de gran utilidad para usted, todos los ejercicios fueron probados y revisados minuciosamente. Tenga paciencia y no sea tan duro consigo mismo, cualquier problema que tenga, ya sea de configuración ó de sintaxis se puede resolver. ASP.NET es una tecnología muy noble, desde mi punto de vista, es una de las mejores opciones para el desarrollo de aplicaciones Web.

CAPÍTULO I

Plataforma .NET

1.1 Antecedentes

A finales del año 2000 Microsoft anunció el lanzamiento de un nuevo entorno de desarrollo llamado Visual Studio.NET, en un solo ambiente contiene Visual Basic .NET, C# y Visual C++.

Con Visual Basic.NET se dió 100% orientación a objetos, ya que anteriormente carecía de herencia, además de montarse sobre una nueva infraestructura llamada .NET, con la única desventaja de que se tuvieron que cambiar varios conceptos del lenguaje, que pueden llegar a ser difíciles de comprender para el programador ajeno a lenguajes orientados a objetos.

Microsoft necesitaba una herramienta para desarrollar aplicaciones informáticas en Internet, anteriormente se habían dado varias soluciones parciales, en Visual Basic 6.0 se pueden crear aplicaciones DHML por medio de un diseñador de DHTML, y también se cuentan con aplicaciones IIS para páginas de servidor activo, sin embargo estas herramientas no satisfacen las necesidades del programador.

Dentro de Visual Studio 6.0 se incorporó una herramienta llamada Visual Interdev, que permite desarrollar páginas de servidor activo o ASP (Active Server Pages), en el que se utiliza un lenguaje llamado Basic Script, sin embargo es difícil para el programador aprender otro lenguaje para poder hacer aplicaciones Web. Entonces se ideó Visual Basic.NET, que permite con el mismo lenguaje de programación, desarrollar aplicaciones para Internet sin tener que aprender un lenguaje completamente nuevo.

El modelo de componentes es una parte esencial para el desarrollo de aplicaciones, la nueva estructura permite la escalabilidad y evita muchos problemas que existían con los componentes, como el *infierno de las DLL*. El modelo COM (Component Object Model) el cual Erich R. Buhler[1] explica: *"COM puede ser visto como un contrato entre una aplicación cliente y una aplicación servidora de intercomunicación de objetos. De esta forma, la mayoría de aplicaciones comparte bibliotecas de otras aplicaciones o del sistema operativo en forma natural, sin demasiadas complicaciones para establecer la misma."*, lo que quiere decir que el desarrollador puede hacer un

[1] BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 4.

componente ya sea EXE (ejecutable) o DLL (biblioteca dinámica). Un componente de software se define como código empaquetado que puede ser utilizado por otras aplicaciones sin importar el lenguaje en que fue programado.

Las DLL o librerías dinámicas son entidades independientes creadas en cualquier lenguaje compatible con Microsoft, que pueden hacer una tarea, la única restricción de las DLL es que tienen que ejecutarse en el mismo proceso que la aplicación.

Los componentes ejecutables o EXE funcionan de forma similar, pueden ser llamados desde cualquier aplicación, con la ventaja de que no necesariamente tienen que estar ejecutándose en el mismo proceso de la aplicación cliente.

Compartir componentes tiene un gran auge en la industria de la informática, sin embargo existe un problema llamado "infierno de las DLL", esto quiere decir que una biblioteca compartida funciona perfectamente bien mientras todas las aplicaciones que la ocupen tengan la versión de componente con el que fueron programados originalmente; sin embargo si instalamos una nueva aplicación que tenga una versión de componente más nueva, entonces el componente anterior se actualiza y todas las aplicaciones que estaban utilizando el otro, ahora utilizarán el nuevo. Cabe mencionar que cuando se crea una versión nueva de un componente es conveniente que sea compatible con aplicaciones que utilicen una versión anterior, sin embargo no siempre es así y las aplicaciones no trabajan como deberían.

A excepción de Windows 2000 y Windows XP que tienen una mejora, los componentes son compartidos y no pueden ser exclusivos para una aplicación en particular.

Con el nacimiento de Internet, las tecnologías voltearon la cara hacia el desarrollo de aplicaciones Web, la propuesta de Microsoft es ASP.NET (Active Server Pages), en lo que a esta tesis concierne se expondrán sus principales características.

1.2 Plataforma .NET de Microsoft

El objetivo de la plataforma .NET es el de simplificar el desarrollo de aplicaciones en Internet por medio de las herramientas y las tecnologías que ofrece.

Además de ser compatible con toda la infraestructura actual de Internet como es HTTP (hypertext transfer protocol), XML (Extensible Markup Language) y SOAP (Simple Object Access Protocol).

Las tecnologías que forman la plataforma .NET (véase figura 1.1) son:

1. **La infraestructura .NET o .NET Framework**. Está basada en el CLR (Common Language Runtime) que tiene como objetivo unificar a todos los lenguajes de la plataforma .NET como son Microsoft Visual Basic.NET, Microsoft Visual C++, Microsoft C#, Microsoft J#. Para no tener que distribuir una librería de ejecución a cada lenguaje, todos tienen los mismos archivos de ejecución. A partir de Windows Server 2003, la infraestructura .NET ya se encuentra incluída. También se puede descargar en forma gratuita del sitio de Microsoft.
2. **Construcción de bloques de servicios .NET o .NET Building Block Services**. Son servicios destinados a ser invocados por cualquier sistema operativo que soporte SOAP (Simple Object Access Protocol). Este tipo de servicio supone que una computadora pueda acceder a un Servicio Web XML vía Internet o a nivel local.
3. **Visual Studio .NET**. Es un ambiente de desarrollo orientado a la nueva generación de aplicaciones Windows e Internet bajo la infraestructura .NET.
4. **Servidores Empresariales .NET o Enterprise Servers**. Herramientas que integran el entorno de la empresa utilizando tecnología de Microsoft.
    a. ***Microsoft SQL Server***. Manejador de base de datos relacional, que utiliza el lenguaje Transact-SQL para consultas, permite el transporte de datos por medio del formato XML, además de tener un acceso seguro por medio de Internet utilizando HTTP, entre otras aplicaciones.
    b. ***Microsoft BizTalk Server 2000***. Provee una integración de negocio a negocio o business-to-business, el objetivo es integrar las aplicaciones de la empresa entre plataformas y organizaciones por Internet.
    c. ***Microsoft Host Integration Server 2000***. Integra aplicaciones, datos y redes entre sistemas, permite unir tecnologías de Internet e Intranet mientras se mantienen los sistemas existentes. Microsoft Host Integration Server 2000 es el reemplazo de SNA Server.

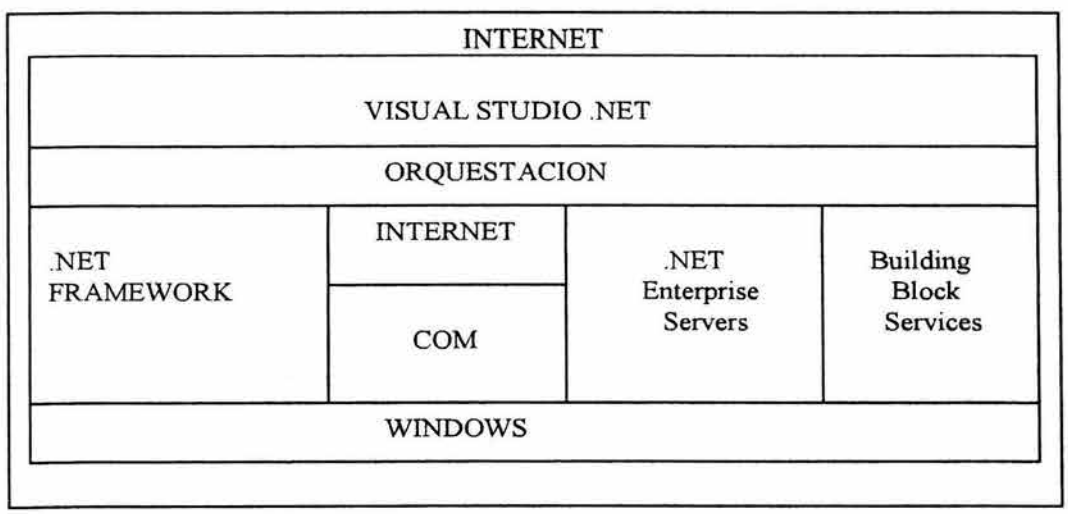
d. ***Microsoft Exchange 2000 Enterprise Server***. Se encarga de administrar la llegada y salida de correo electrónico a las organizaciones, así como de trabajar en conjunto con otras aplicaciones de Microsoft.

e. ***Microsoft Application Center 2000***. Se encarga de proveer herramientas administrativas y de descarga para aplicaciones en Internet, así como de trabajar con varios servidores como si fuera uno solo.

f. ***Microsoft Internet Security and Acceleration Server 2000***. Provee conectividad a Internet en una forma segura, rápida y manejable, también es un cortafuego empresarial multicapa y escalable.

g. ***Microsoft Commerce Server 2000***. Herramienta que permite realizar comercio electrónico de una manera sencilla y segura.

Figura 1.1: Visual Studio .NET y sus componentes



Fuente: MICROSOFT CORPORATION, ***Programming with Microsoft Visual Basic .NET***, Microsoft Corporation, Colombia, 2002, Módulo 1, p. 2.

1.3 Infraestructura .NET y sus componentes

La infraestructura .NET o .NET Framework como mejor se le conoce es un conjunto de tecnologías que abarcan varias áreas. Los componentes de la infraestructura .NET son: Common Language Runtime (CLR), .NET Framework Class Library, ADO.NET, ASP.NET.

1.4 Common Language Runtime (CLR)

La mayoría de los lenguajes necesitan un componente que trabaje conjuntamente con el sistema operativo, para que provea a la aplicación los recursos necesarios para su funcionamiento.

Hasta antes de .NET cada lenguaje tenía su propio componente de ejecución o *Runtime* como lo llamaremos de ahora en adelante. Esto ocasionaba los siguientes problemas:

a. El *Runtime* del lenguaje debía de estar siempre disponible para la aplicación, lo que no siempre era posible en una página Web, ya que no se sabía si la computadora cliente tenía instalado el componente para hacer funcionar la aplicación.
b. Las funciones entre los lenguajes eran diferentes aún y cuando se hicieran las mismas tareas, la forma en que se abrían los archivos en Visual C++ eran totalmente diferentes a como lo hacía Visual Basic.
c. Al utilizar un componente que estaba programado en un lenguaje diferente, del que se estaba utilizando en una aplicación, no siempre los tipos de datos eran compatibles.

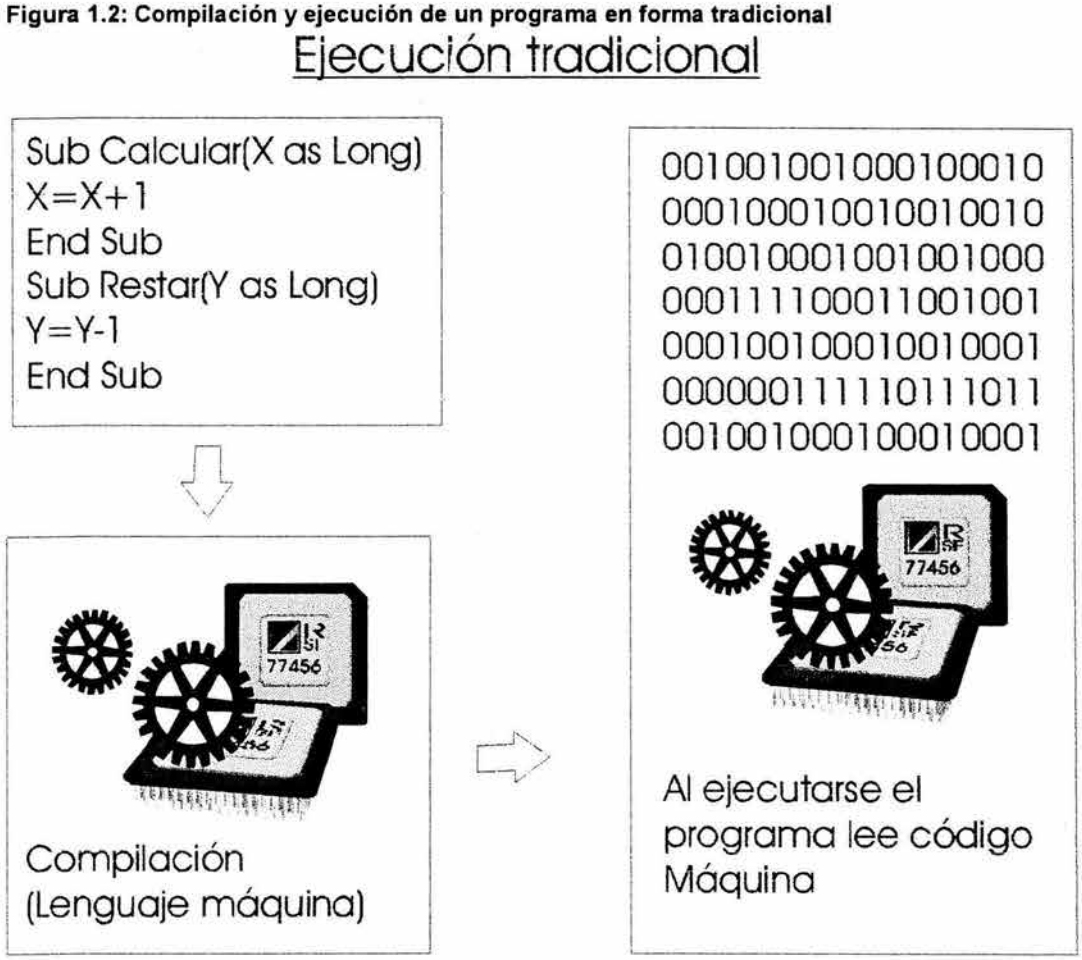
Todos estos problemas se resuelven con el CLR (Common Language Runtime), como su nombre lo dice es un lenguaje de ejecución común para todas las aplicaciones. Ahora todas las funciones de los lenguajes de la plataforma .NET realizan sus tareas en forma similar. Para que existiera compatibilidad entre los lenguajes, se tuvieron que hacer cambios, ahora es más fácil compartir componentes que sean compatibles entre lenguajes.

1.5 Código administrado

Anteriormente la forma en que un programa se compilaba consistía en traducir las instrucciones del lenguaje en código máquina, cuando el programa era ejecutado entonces el código máquina era leído por el procesador (véase figura 1.2).

Con la nueva versión de Visual Studio .NET, el programa desarrollado no se transforma directamente en código máquina, sino en un lenguaje llamado MSIL (Microsoft Intermediate Language) o lenguaje intermedio de Microsoft, estas instrucciones solamente pueden ser entendidas por un compilador llamado JIT (Just In Time compiler) o compilador instantáneo. Cuando se ejecuta la aplicación este compilador interpreta el lenguaje intermedio MSIL, lo checa para ver si no trae errores de memoria y lo ejecuta. Cuando una aplicación se genera en MSIL se le llama **código**

Figura 1.2: Compilación y ejecución de un programa en forma tradicional



Fuente: BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 14.

administrado y los creados en la plataforma .NET se le llaman **ensamblados**. Con este nuevo formato, cualquier ensamblado puede ser ejecutado por cualquier sistema operativo si se cuenta con el compilador JIT (Just In Time compiler), por ahora

solamente los sistemas operativos de Microsoft cuentan con JIT sin embargo en un futuro otros sistemas operativos podrán contar con este compilador (véase figura 1.3).

1.6 Ensamblados y DLL

Como se había mencionado antes, varias DLL (Dynamic Link Library) de la misma biblioteca pero con diferentes versiones, no pueden ejecutarse al mismo tiempo.

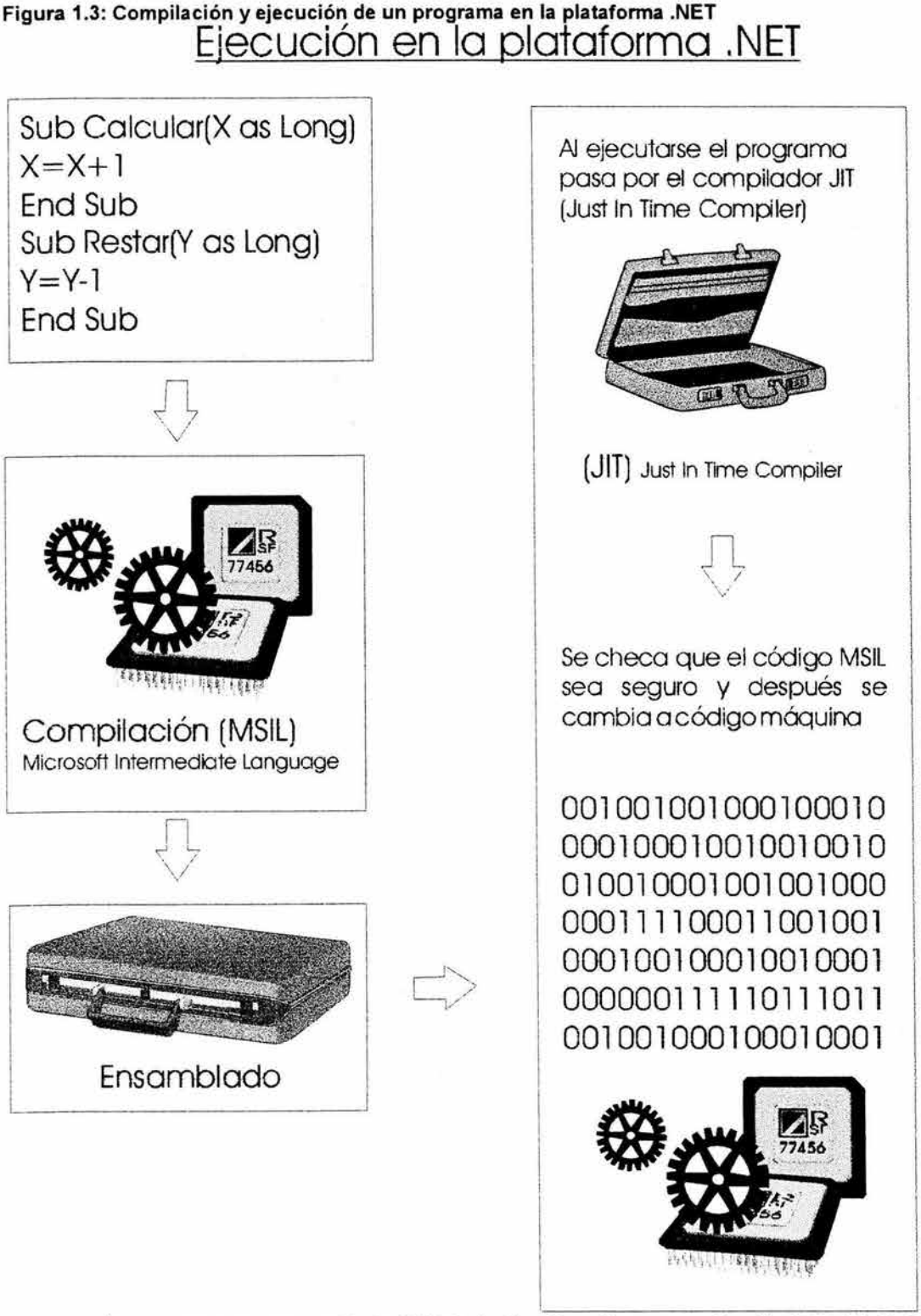
Un componente hecho con código **no administrado** tiene que registrarse en la máquina antes de ser utilizado, el problema aumenta cuando se utiliza un componente distribuido DCOM (Distributed Component Object Model), la ventaja de los ensamblados de .NET es que no tienen que registrarse, ya que la descripción del componente se encuentra dentro de él mismo. Con solo copiar el componente en la carpeta donde está la aplicación, se puede hacer uso de él.

Los ensamblados (código administrado) están compuestos de dos partes: la implementación y el manifiesto que incluye la descripción del ensamblado (véase figura 1.4). Cuando el sistema operativo detecta un ensamblado en vez de COM, entonces el CLR (Common Language Runtime) checa que existan todos los recursos y que no hayan errores, el JIT (Just In Time compiler) interpreta el lenguaje intermedio (MSIL), lo transforma a código máquina, ejecutándolo después.

Existen tres características de los ensamblados:

a. Es más fácil hacer la instalación, ya que con el simple hecho de que se copie el ensamblado dentro de la misma carpeta donde esta la aplicación, puede hacerse uso de éste.
b. Se pueden tener componentes privados, porque no afecta a otras aplicaciones que utilicen otra versión de componente, ya que si los ensamblados necesarios para una aplicación se encuentran agrupados en la carpeta de la aplicación, no tendrán conflictos con otros componentes.
c. Se pueden tener componentes compartidos, lo que ayuda a que varias aplicaciones compartan la misma biblioteca al mismo tiempo.

Figura 1.3: Compilación y ejecución de un programa en la plataforma .NET

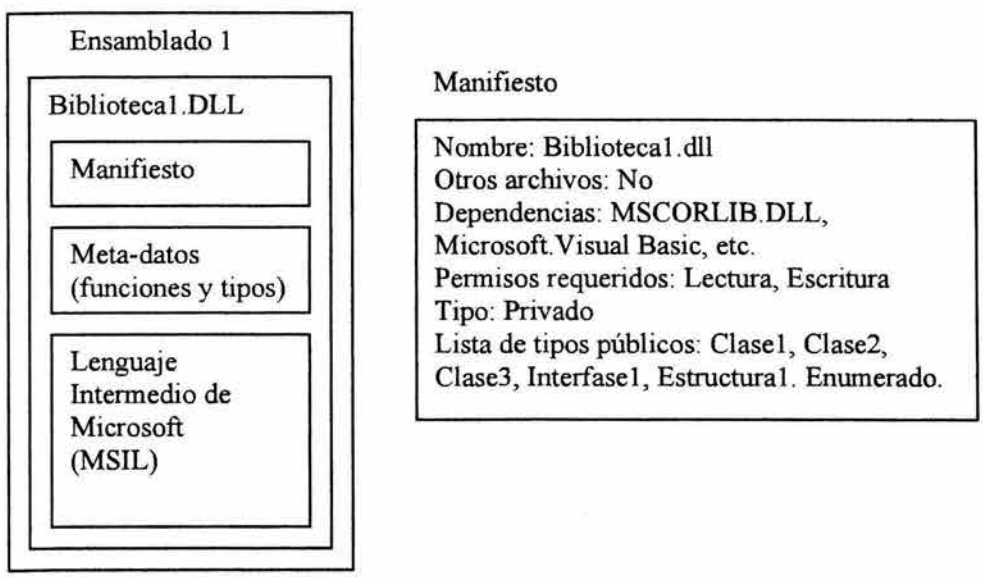


Fuente: BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 14.

Como lo dijimos antes, los ensamblados no necesitan del archivo de registro para poder localizarse, sin embargo utilizan un Caché de ensamblados global o GAC (Global Assembly Caché). Es un sitio donde puede estar el ensamblado para que pueda ser ubicado por todas las aplicaciones que lo necesite.

Cuando se compila un programa en Visual Studio .NET, el compilador deja indicado qué versión de biblioteca necesita, al ser ejecutado el programa, CLR (*Common Language Runtime*) se dará cuenta de los recursos que necesita ese programa para su correcto funcionamiento y proporcionará la biblioteca necesaria. En caso de que se instale otra versión más actual de la misma biblioteca, la aplicación utilizará la versión original, esto quiere decir que pueden coexistir diferentes versiones de una misma biblioteca, siendo privadas para la aplicación que la requiera.

Figura 1.4: Estructura de un ensamblado



Fuente: BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 18.

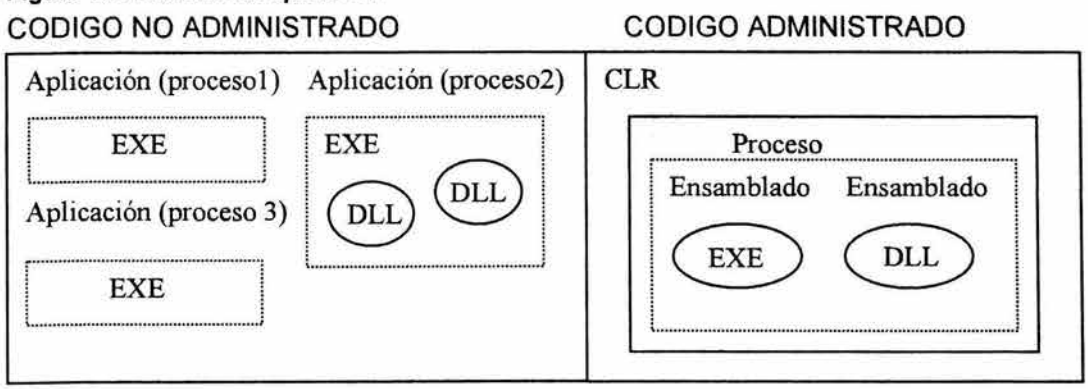
1.7 Dominios de la aplicación

La utilización de componentes ofrece la ventaja de tener una o varias bibliotecas que pueden ser utilizadas por diferentes aplicaciones; como se mencionó antes, existen

dos tipos de componentes, las DLL (Dynamic Link Library) y los EXE (executables), la diferencia entre éstos dos es que los EXE son independientes y crean un proceso por cada aplicación que los requiera, por ejemplo: si 25 aplicaciones solicitan un EXE, entonces se generarán 25 procesos, en caso de que el componente EXE llegara a fallar, la aplicación que lo mandó llamar no necesariamente debe detenerse, porque un EXE es un programa totalmente independiente. Las DLL no generan un proceso cada vez que son llamadas, pero sí dependen de un solo proceso general para poder ser ejecutada; la desventaja que tiene es que si llega a fallar una DLL es probable que la aplicación se detenga.

En teoría, los EXE son componentes ideales, sin embargo si varios clientes en Internet solicitaran un componente, se tendría que generar un proceso por cada pedido, lo que saturaría las capacidades del sistema operativo.

Figura 1.5: Dominio de aplicación



Fuente: BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 21.

CLR (Common Language Runtime) arregla este problema con el llamado dominio de aplicación, que se puede definir como un gran proceso que alberga a todas las aplicaciones EXE o DLL, véase la figura 1.5.

1.8 El recolector de basura

La plataforma .NET tiene un servicio llamado *"el recolector de basura"*, se encarga de liberar en forma automática la memoria de todos los objetos que no están siendo utilizados.

Anteriormente el programador tenía que destruir un objeto después de utilizarlo para poder liberar la memoria que ocupaba, esto genera dos problemas; el primero es que el programador inexperto no siempre sabe que debe de liberar la memoria y el segundo son las referencias circulares.

En COM (Componente Object Model), cada vez que se crea un objeto se incrementa un contador interno de referencias, y cada que la referencia es liberada se decrementa el contador, en el instante en que el número de referencias llega a cero, se libera la memoria.

Sin embargo, en ocasiones un objeto A hace referencia a un objeto B y viceversa, lo que genera una referencia circular; como se puede ver, ambos objetos están encadenados ya que no se puede realmente liberar una referencia al objeto A porque el otro objeto depende de ella. Todo se basa en el orden, si no es el correcto entonces la memoria no es liberada.

Debido a esto, el recolector de basura se encarga de liberar la memoria de los objetos que ya no son utilizados, además existen métodos para forzar al recolector de basura a que se ejecute en un momento determinado.

1.9 Librería de clases de la infraestructura .NET

La infraestructura .NET incluye una gran cantidad de clases ya compiladas como ensamblados, para que el usuario pueda hacer uso de ellas. Estas clases ayudan a simplificar la programación, ya que muchas tareas comunes ya están incluidas en las clases.

Al ser un gran número de clases, se organizan en forma jerárquica, debido a la herencia, una clase puede tener subclases y ésta a su vez puede tener otras, es por eso que se empaquetan en los llamados **espacios de nombres** o **namespaces**.

Estas clases se encuentran centralizadas, por lo que desde cualquier lenguaje de la plataforma .NET pueden ser llamadas con casi la misma sintaxis, además de existir una documentación de éstas, lo cual es importante porque una vez que se conozcan algunas de estas clases no será difícil implementarlas en otro lenguaje de la plataforma .NET. El *espacio de nombres* ***System*** contiene las clases fundamentales como son los eventos, manejadores de eventos, interfaces, excepciones de procesamiento, atributos, etc. ***System.Collections*** es el encargado de proveer listas,

tablas y en general la agrupación de datos. ***System.IO*** se encarga de los flujos de datos. ***System.NET*** se encarga del protocolo TCP/IP y de los sockets. ***System.Windows.Forms*** son clases encargadas de construir la interfaz del cliente, anteriormente no se tenía tanto control sobre la interfaz como se tiene ahora, ya que no es necesario acceder a la API (Application Programming Interface), sino que éstas clases proveen lo necesario para modificar la interfaz gráfica. ***System.Drawing*** se encarga de proveer la funcionalidad de gráficos.

1.10 ADO.NET

ADO.NET o Active X Data Objects .NET permite que las aplicaciones puedan utilizar datos de uno o más orígenes, permitiendo desconectarse de la base de datos para poder manipular los datos en forma local, con lo que se libera al servidor de tener una conexión constante con la aplicación cliente. ADO.NET ofrece características más avanzadas que ADO, tiene soporte para XML (Extensible Markup Language). Sus principales *espacio de nombres* son: ***System.Data*** y ***System.XML*** que proveen el soporte para XML.

1.11 ASP.NET

Las páginas activas de servidor .NET son utilizadas en el servidor para construir aplicaciones en Internet. Debido a que utiliza el CLR (Common Language Runtime) puede utilizarse Visual Basic.NET para programarse. Este modelo está basado en los estándares de Internet como son HTTP (Hypertext Transfer Protocol), XML (Extensive Markup Language) y SOAP (Simple Object Access Protocol), es por eso que puede ser accedido por cualquier cliente conectado a Internet.

Los Servicios Web XML permiten bloques de construcción para el desarrollo de aplicaciones en Internet.

System.Web se refiere a los servicios de seguridad, configuración, además de los componentes necesarios para compartir Servicios Web XML.

System.Web.Services. Se encarga de controlar los requerimientos de los servicios Web.

System.Web.UI provee controles HTML y controles Web. Los primeros son controles sencillos, los segundos son controles más complejos como por ejemplo el DataGrid.

1.12 Configuración

Es momento de realizar la configuración necesaria, para que usted pueda desarrollar aplicaciones ASP.NET con Visual Studio.NET. Dependiendo del sistema operativo con que cuente en su computadora se harán las modificaciones correspondientes. El principal componente a configurar es el servidor Web de Microsoft, que es el ***Internet Information Server*** ó ***IIS versión 5.0*** ó superior.

Asumo que usted tiene un conocimiento bastante amplio del sistema operativo con que cuenta, en caso contrario, recomiendo la instalación de XP Profesional, ya que lo encuentro más sencillo de configurar que Windows 2000 Server.

Es importante tener una fuente de información adicional a este trabajo de tesis para encontrar respuestas a sus preguntas e información en general, para ello se encuentra el sitio de Microsoft para desarrolladores en:

http://msdn.microsoft.com/library/default.asp

Allí, encontrará un menú con nodos en la parte izquierda de la página. Para ver información sobre .NET oprima *.NET Development*, encontrará información sobre ASP.NET, Visual Studio.NET, Visual Basic.NET y muchos más.

Otro sitio importante de Microsoft, se encuentra en la siguiente dirección:

http://msdn.microsoft.com/asp.net/

Un sitio de preguntas y respuestas de ASP.NET se encuentra en:

http://www.kbalertz.com/

No es necesario darse de alta en este sitio, con que escriba la palabra o frase a buscar en la caja de texto *search* se presentarán los resultados.

Otro se encuentra en http://www.dotnet247.com/247reference/default.aspx

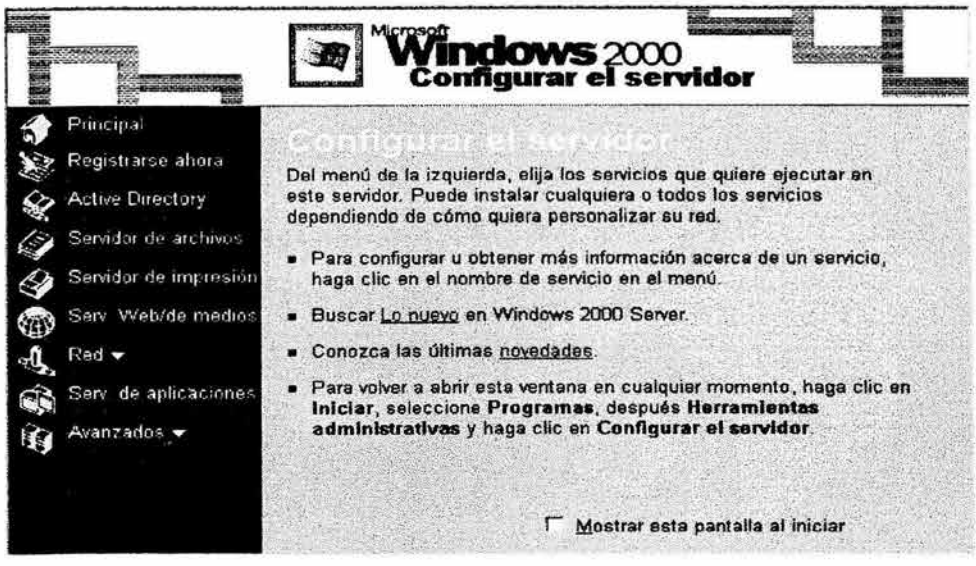
En caso de no encontrar la información necesaria en los sitios antes mencionados, *la mejor forma de encontrar la solución a un problema es escribiendo en la caja de texto de un buscador como Google, el número y definición del error que*

muestra la pantalla del explorador de Internet. Recomiendo buscar información en grupos de ayuda, puede encontrar muchos en Internet, ya que problemas similares a los que usted pudiera tener ya fueron resueltos allí.

1.12.1 Configuración Windows 2000 Server

La configuración en Windows 2000 Server no es compleja, solamente necesita configurar el *Directorio Activo* e *Internet Information Server*, por medio de los asistentes que incluye el sistema operativo. Para configurar el servidor oprima *Inicio + Programas + Herramientas Administrativas + Configurar el servidor,* entonces aparecerá el panel que se muestra en la figura 1.6.

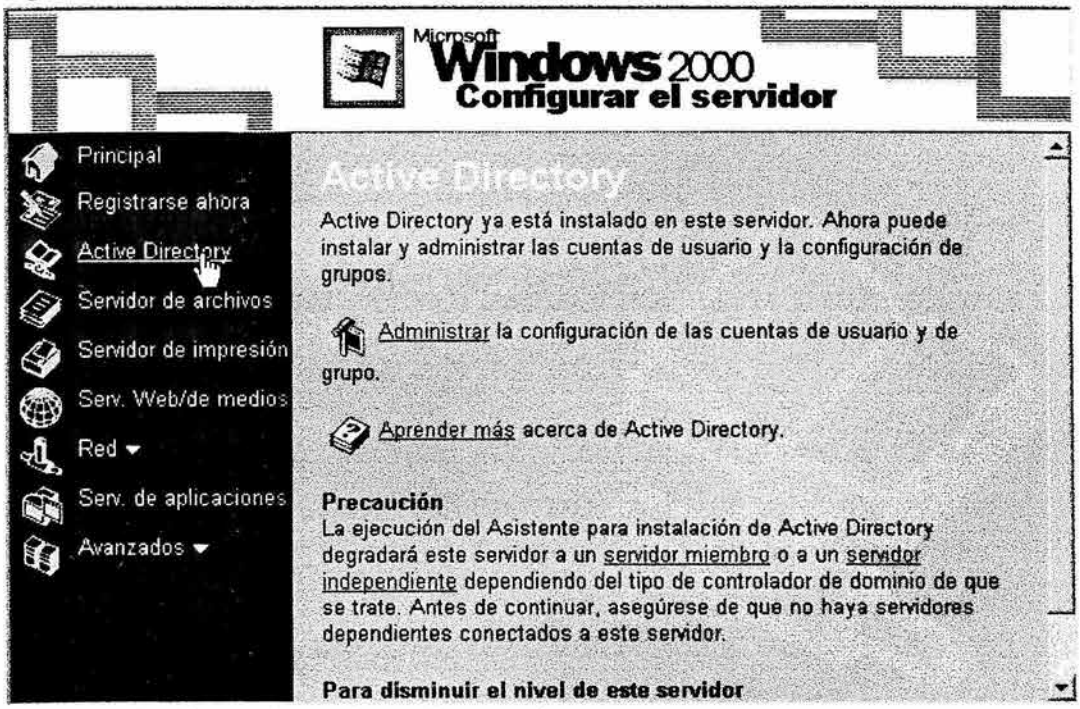
Figura 1.6: Panel para configurar el servidor



Fuente: Aplicación creada por el autor de esta tesis.

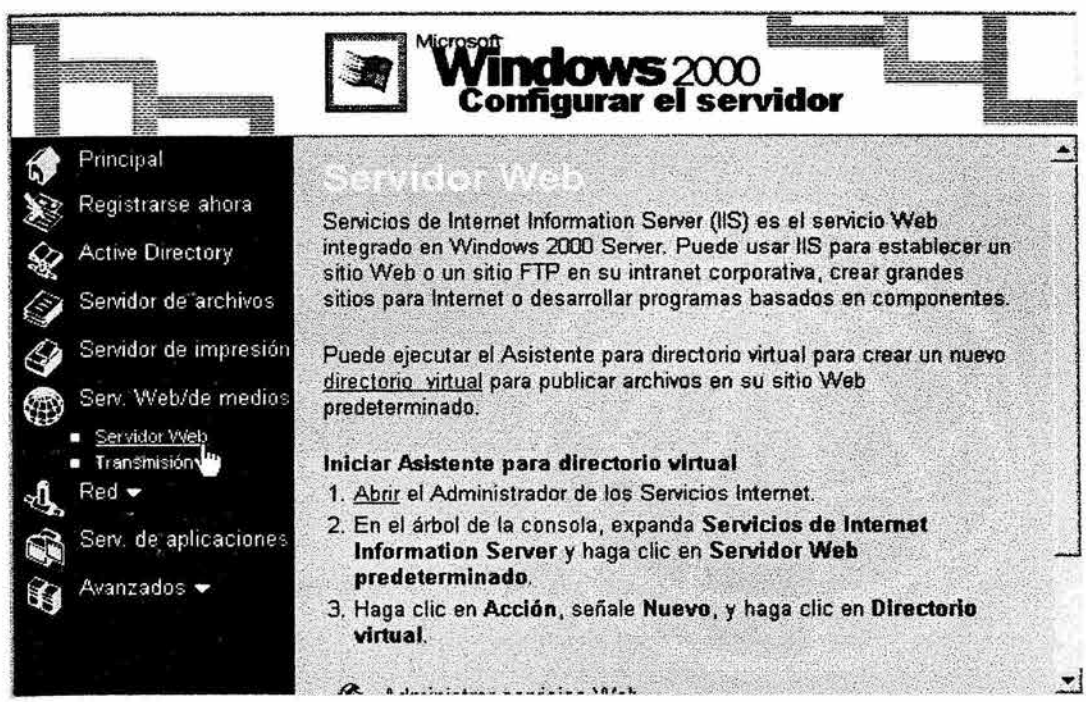
El *Directorio Activo* debe ser configurado por medio del asistente, si es que usted no lo tiene configurado previamente, oprima *Active Directory* para que se muestren las diferentes opciones, como se muestra en la figura 1.7, para mayor información revise la siguiente dirección http://www.wown.info/j_helmig/w2ksvrin.htm. El servidor Web (IIS) también debe de ser configurado, para esta operación oprima *Serv. Web/de medios + Servidor Web*, véase figura 1.8.

Figura 1.7: Panel para configurar el Active Directory



Fuente: Aplicación creada por el autor de esta tesis.

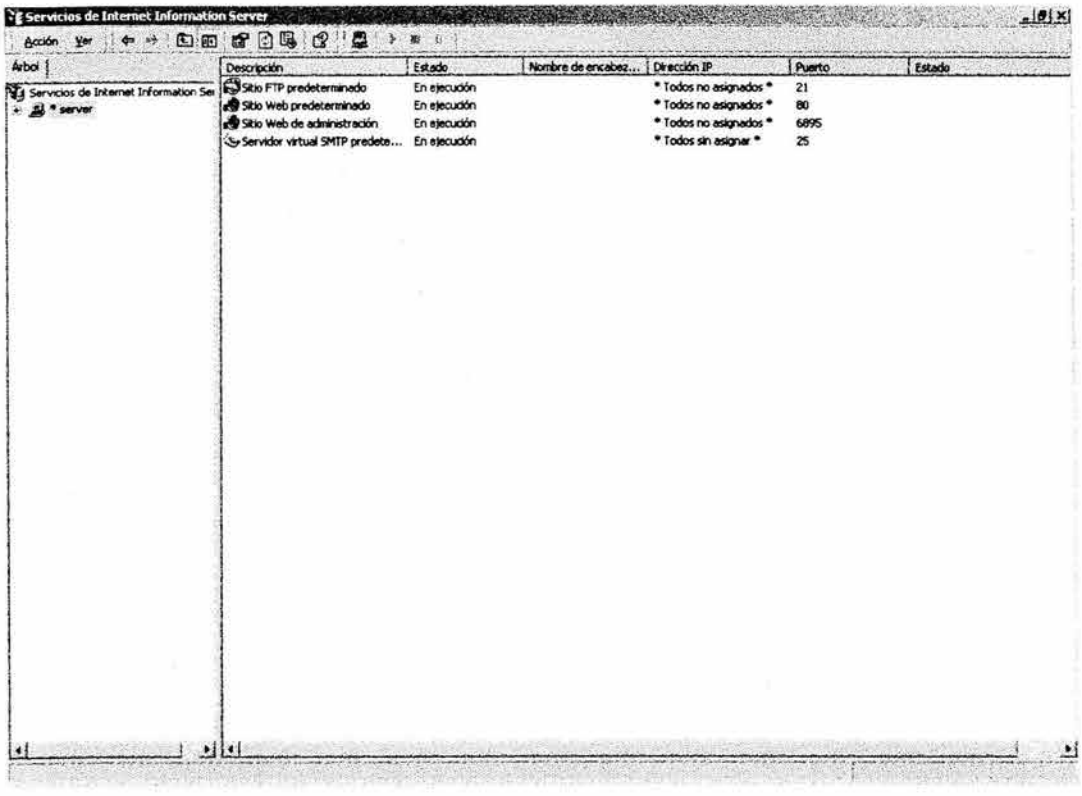
Figura 1.8: Panel para configurar el servidor Web IIS



Fuente: Aplicación creada por el autor de esta tesis.

El servidor Web se puede ver en *Inicio + Programas + Herramientas Administrativas + Administrador de servicios de Internet*, véase la figura 1.9.

Figura 1.9: Servicios de Internet Information Server

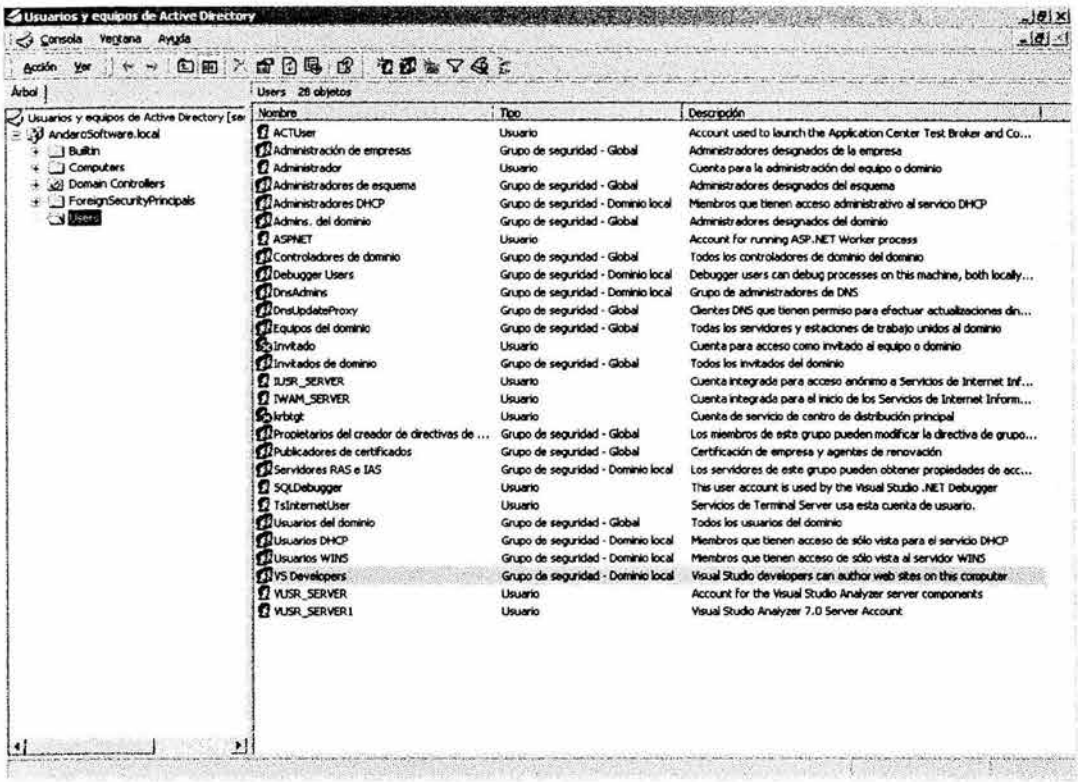


Fuente: Aplicación creada por el autor de esta tesis.

En caso de tener mayores problemas debe de revisar la documentación del sistema operativo. Cuando se hacen aplicaciones para servidores Windows, es muy importante empaparse con el tema de seguridad, porque automáticamente Windows restringe muchos permisos.

Es importante revisar que usted esté dado de alta dentro del *Directorio Activo* como usuario de **VS Developers**, para eso oprima *Inicio + Programas + Herramientas Administrativas + Usuarios y equipos de Active Directory*. Expanda el nodo de *mi_computadora.local* y haga clic en *users*, después clic derecho en **VS Developers** y seleccione *propiedades*, véase la figura 1.10. Finalmente oprima *miembros* y revise que se encuentra dentro de la lista, en caso contrario oprima *agregar*, véase la figura 1.11.

Figura 1.10: Usuarios y equipos de Active Directory



Fuente: Aplicación creada por el autor de esta tesis.

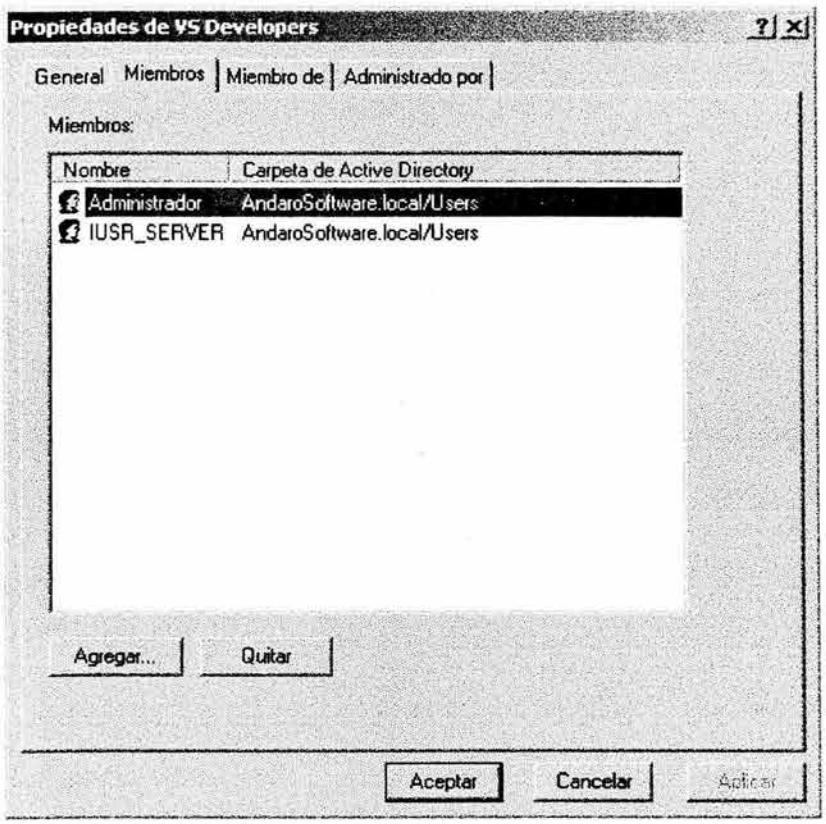
Para mayor información y descarga del *service Pack,* ingrese a la página principal de Windows 2000 (http://www.microsoft.com/windows2000/).

1.12.2 Configuración Windows XP Profesional

Encuentro más sencillo de configurar este sistema operativo que Windows 2000. Windows XP Profesional solamente requiere de la configuración del servidor Web IIS, ya que no está instalado por defecto. Se debe crear un directorio virtual de la siguiente forma:

- *Inicio + Programas + Herramientas Adminsitrativas + Internet Information Services* ó *Administrador de Servicios de Internet.*
- Después expanda el nodo de *Web Sites* ó *sitios Web* del nodo *mi_computadora.*

Figura 1.11: Miembros dentro de VS Developers



Fuente: Aplicación creada por el autor de esta tesis.

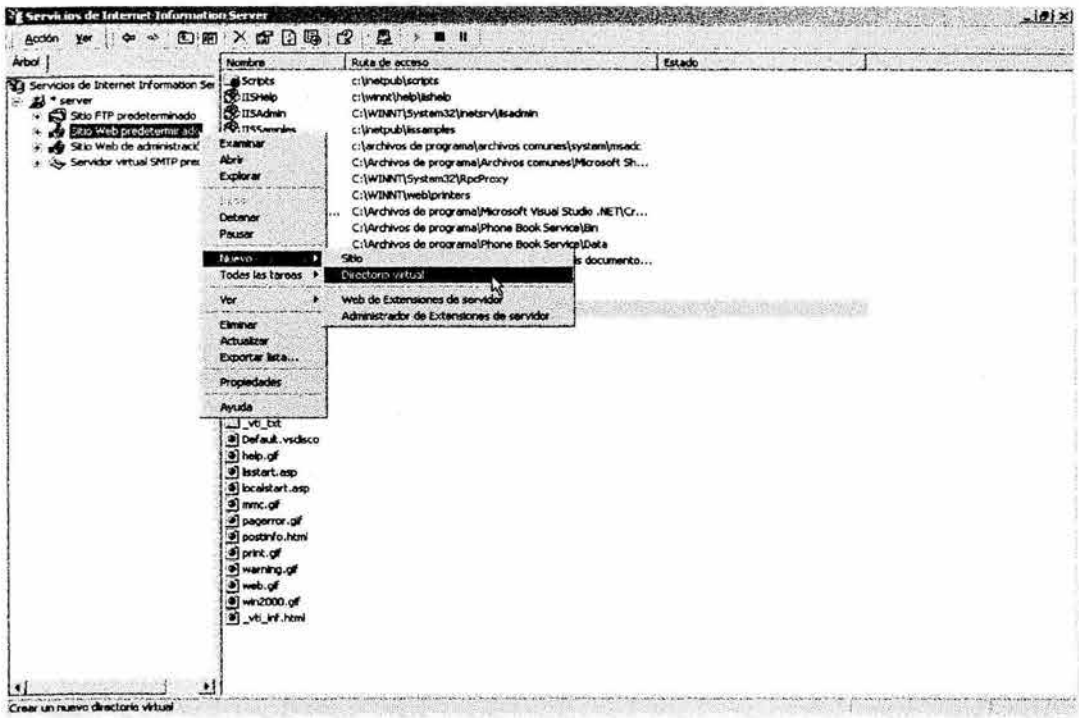
- Haga clic derecho en *Default Web Site* ó *sitio Web predeterminado,* apunte a *Nuevo,* y después haga clic en *Virtual Directory* ó *Directorio Virtual,* véase la figura 1.12. Cuando aparezca el asistente oprima *Siguiente*, véase figura 1.13.
- En *alias* del directorio virtual, escriba **MSADC**, y oprima *Siguiente,* véase figura 1.14.
- Oprima el botón *Browse*, véase figura 1.15. Busque cualquiera de las siguientes direcciones y oprima *Siguiente,* véase figura 1.16.

  C:\Archivos de programa\Archivos Comunes\System\msadc

  C:\Program Files\Common Files\System\msadc
- En los permisos de acceso, seleccione *Ejecutar (por ejemplo, aplicaciones ISAPI o CGI)* ó *Execute (such as ISAPI applications or CGI).* Previamente se encuentran seleccionados *Lectura* ó *Read* y *Ejecutar secuencias de comandos (por ejemplo, ASP)* ó *Run scripts such as ASP,* véase figura 1.17.

- Haga clic en *Siguiente*, y oprima *Finalizar.*

Figura 1.12: Crear un directorio virtual

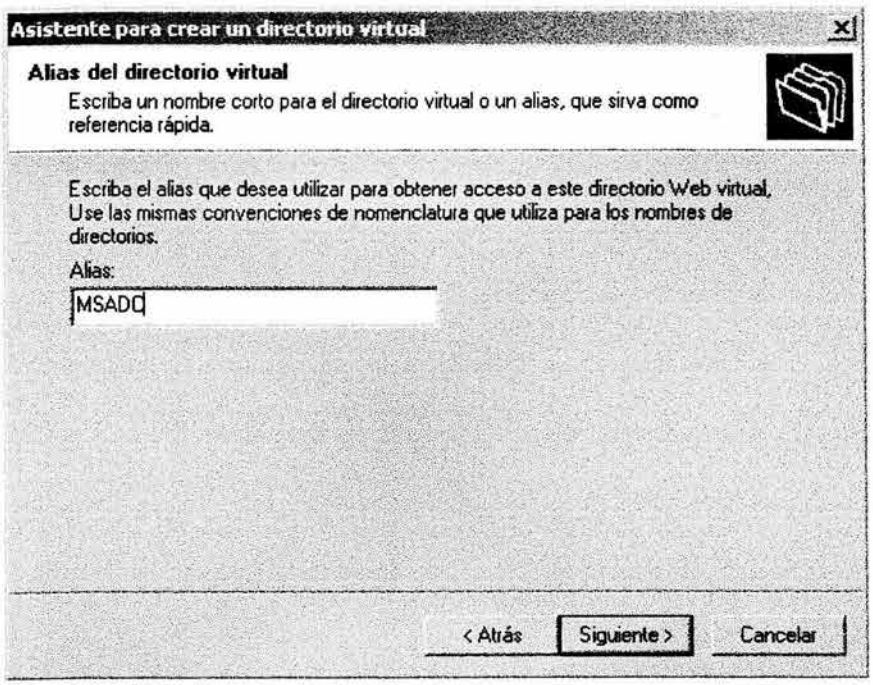


Fuente: Aplicación creada por el autor de esta tesis.

Figura 1.13: Asistente para crear un directorio virtual

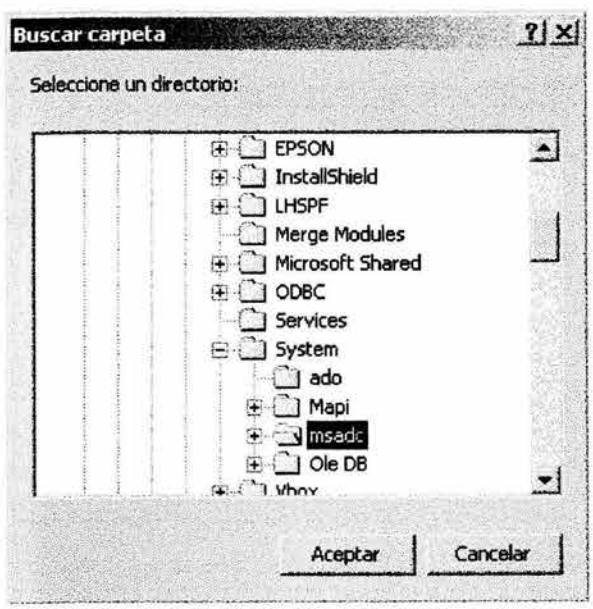
Fuente: Aplicación creada por el autor de esta tesis.

Figura 1.14: Escribir el alias del directorio virtual



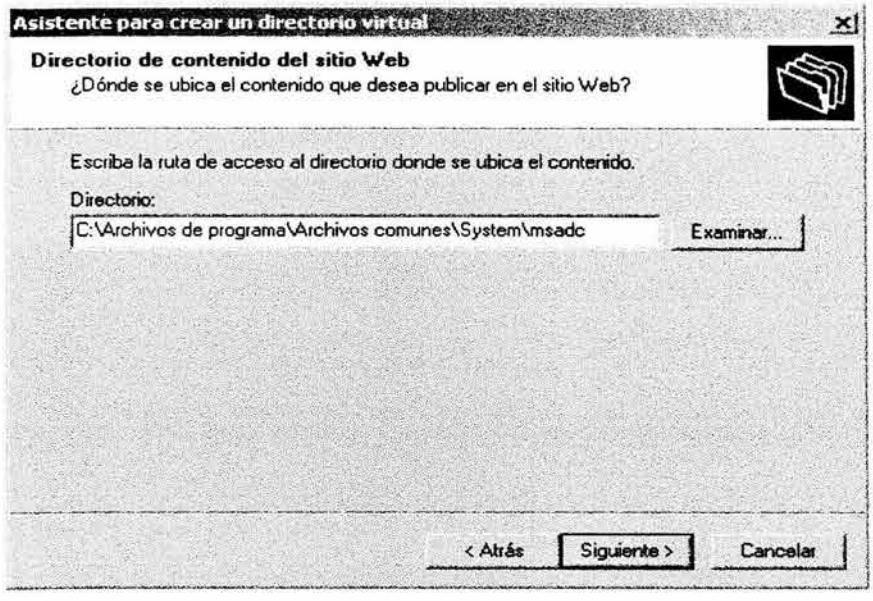
Fuente: Aplicación creada por el autor de esta tesis.

Figura 1.15: Buscar la ruta para posicionar el directorio



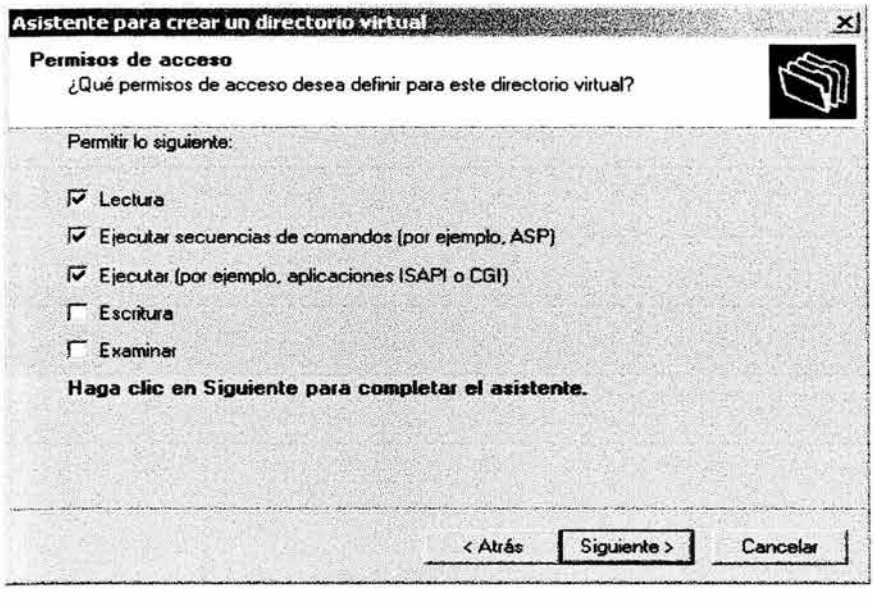
Fuente: Aplicación creada por el autor de esta tesis.

Figura 1.16: Ubicación final del directorio virtual



Fuente: Aplicación creada por el autor de esta tesis.

Figura 1.17: Agregar permisos necesarios



Fuente: Aplicación creada por el autor de esta tesis.

Para mayor información y descarga del *service Pack*, ingrese a la página principal de Windows XP (http://www.microsoft.com/windowsxp/default.asp).

1.12.3 Configuración Windows Server 2003

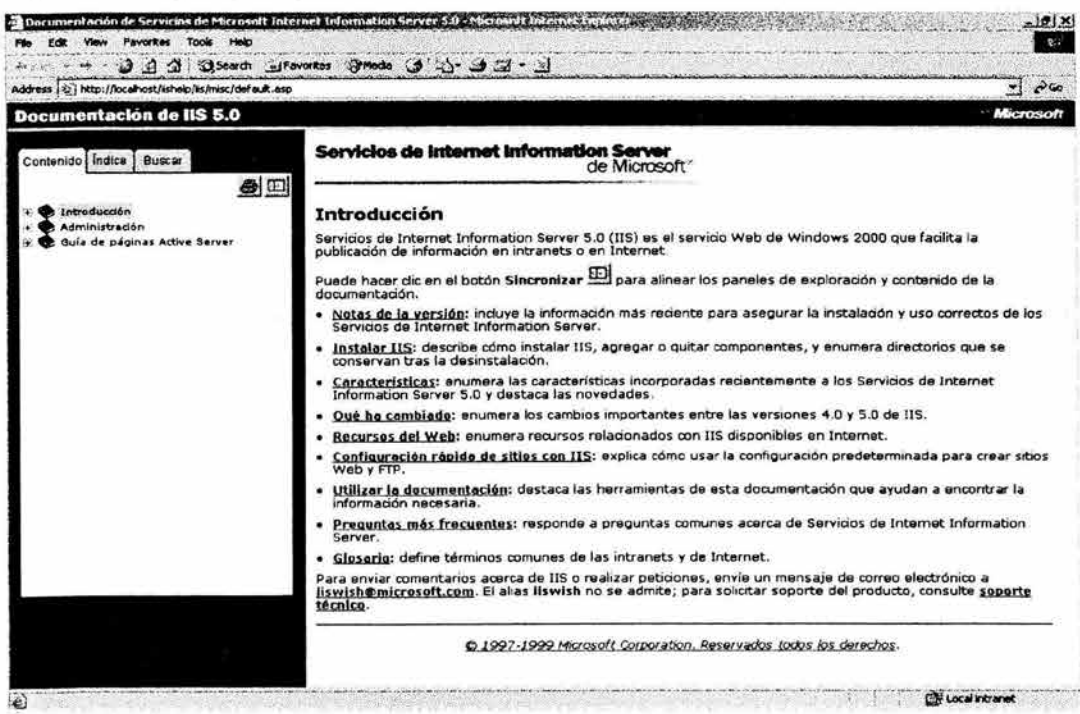
La plataforma .NET se incluye dentro de este sistema operativo, no debe de ser mayor problema instalar Visual Studio.NET, ni Microsoft SQL Server 2000. Debe de revisar la documentación de la configuración del servidor Web IIS versión 6.0.

Para mayor información, ingrese a la página de Windows Server 2003 (http://www.microsoft.com/windowsserver2003/default.mspx). La página del servidor Web (IIS 6.0) está en (http://www.microsoft.com/windowsserver2003/iis/default.mspx) y http://www.microsoft.com/windowsserver2003/community/centers/iis/default.asp.

1.12.4 Probar el servidor Web

Para probar que esté configurado correctamente el servidor Web, abra Internet Explorer y en la caja de texto donde está el URL escriba: http://localhost, http://127.0.0.1 ó http://mi_nombre_de_computadora, véase figura 1.18.

Figura 1.18: Página principal del servidor Web IIS



Fuente: Aplicación creada por el autor de esta tesis.

Es importante mencionar que software como *Kazaa,* crea su propio directorio virtual, por lo que debe de probar cualquiera de las direcciones URL antes mencionadas, hasta que aparezca la página Web predeterminada. Cuando desarrolla aplicaciones, es conveniente tener solamente las aplicaciones que necesita para desarrollar.

Las aplicaciones ASP.NET desarrolladas en Visual Studio.NET, se guardarán en la ruta predeterminada:

<Unidad>:\InetPub\wwwroot

Cuando quiera acceder a las aplicaciones en forma local, se debe de escribir la siguiente dirección ó URL en el navegador de Internet:

http://localhost/Mi_Nombre_Proyecto/Mi_Forma_Web.aspx ó

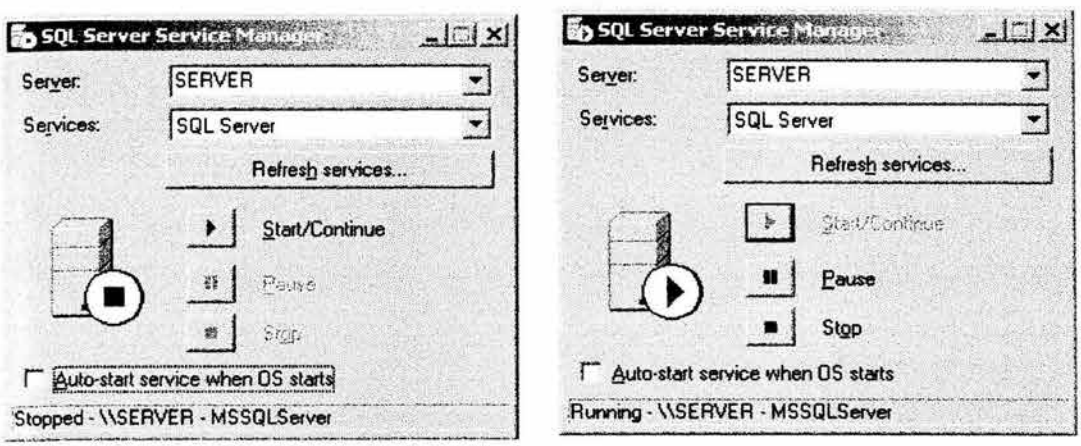
http://127.0.0.1/ Mi_Nombre_Proyecto/Mi_Forma_Web.aspx ó

http://Mi_Computadora/Mi_Nombre_Proyecto/Mi_Forma_Web.aspx

1.12.5 Microsoft SQL Server 2000

Microsoft SQL Server 2000 debe de estar funcionando cuando se ejecuten los ejercicios, porque algunos de ellos requieren de la base de datos *pubs.* Para revisar esto, oprima *Inicio* + *Microsoft SQL Server* + *Administrador de Servicios.* Debe de encontrarse activo, en caso contrario actívelo, véase figura 1.19.

Figura 1.19: SQL Server 2000 detenido y funcionando



Fuente: Aplicación creada por el autor de esta tesis.

Para mayor información y descarga del *service Pack,* ingrese a la página principal de Microsoft SQL Server (http://www.microsoft.com/sql/).

1.12.6 Machine.config

Este archivo es muy importante, debido a que en él se encuentra la configuración de la computadora. Por medio de éste se configuran muchas opciones, como permisos y propiedades específicas. Es importante que el lector estudie este archivo e investigue las opciones que tiene. Algunos errores de configuración están asociados con la falta de configuración de este archivo. Se encuentra en la siguiente ruta:

C:\WINNT\Microsoft.NET\Framework\ v1.0.3705\CONFIG\machine.config

Por lo pronto me he encontrado que cuando trabajo en Windows 2000 Server, necesito cambiar un parámetro en el archivo ***machine.config,*** para poder ejecutar los Servicios Web XML. Abra el archivo y busque la etiqueta ***<processModel*** cambie el parámetro ***userName*** por ***"SYSTEM"***, quedando de la siguiente forma:

```
<processModel enable="true" timeout="Infinite" idleTimeout="Infinite"
shutdownTimeout="0:00:05" requestLimit="Infinite" requestQueueLimit="5000"
restartQueueLimit="10" memoryLimit="60" webGarden="false"
cpuMask="0xffffffff" userName="SYSTEM" password="AutoGenerate"
logLevel="Errors" clientConnectedCheck="0:00:05"
comAuthenticationLevel="Connect" comImpersonationLevel="Impersonate"
responseRestartDeadlockInterval="00:09:00"
responseDeadlockInterval="00:03:00" maxWorkerThreads="25"
maxIoThreads="25"/>
```

Antes de hacer cualquier cambio, cheque si el sistema ya lo configuró antes de hacerlo usted.

1.12.7 Descargar aplicaciones ASP.NET

La forma de descargar archivos ASP.NET en el servidor es muy sencilla, toda aplicación .NET requiere que se instale la infraestructura .NET, es gratuita y la puede descargar en la dirección:

http://msdn.microsoft.com/netframework/downloads/howtoget.aspx

La carpeta con los archivos del proyecto deben de copiarse dentro del directorio virtual, el servidor Web sigue una ruta para encontrar los archivos solicitados para mostrarlos en el explorador, allí es en donde se deben de copiar los archivos de ASP.NET, en la mayoría de las computadoras se encuentra en:

C:\Inetpub\wwwroot\Ponga_aquí_la_carpeta_de_su_proyecto

1.12.8 Crear un proyecto ASP.NET

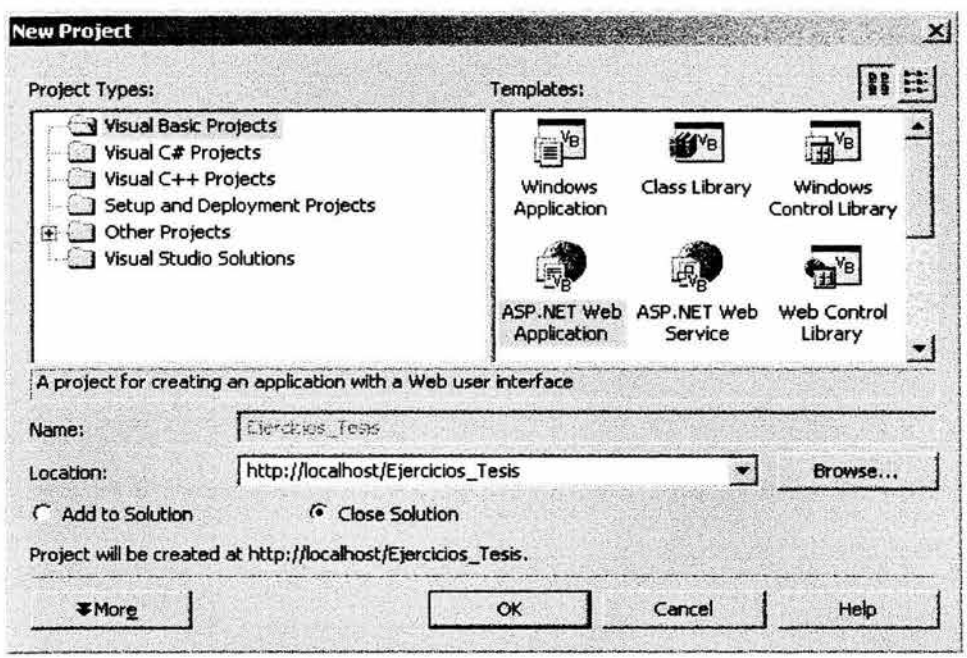
Los ejercicios de esta tesis requieren la creación de un nuevo proyecto ASP.NET, dentro del cual se agregarán todas las formas Web, que se irán creando conforme se avance en la lectura de este trabajo.

Abra Visual Studio.NET, seleccione *New Project + ASP.NET Web application*, en *location* escriba lo siguiente:

http://localhost/Ejercicios_Tesis

Oprima aceptar, entonces se conectará con el servidor creándose un nuevo proyecto, véase la figura 1.20.

Figura 1.20: Crear un nuevo proyecto ASP.NET



Fuente: Aplicación creada por el autor de esta tesis.

A partir de ahora todas las formas Web que cree, tendrán la siguiente estructura: http://Nombre de servidor / Nombre de proyecto / Nombre de la forma Web

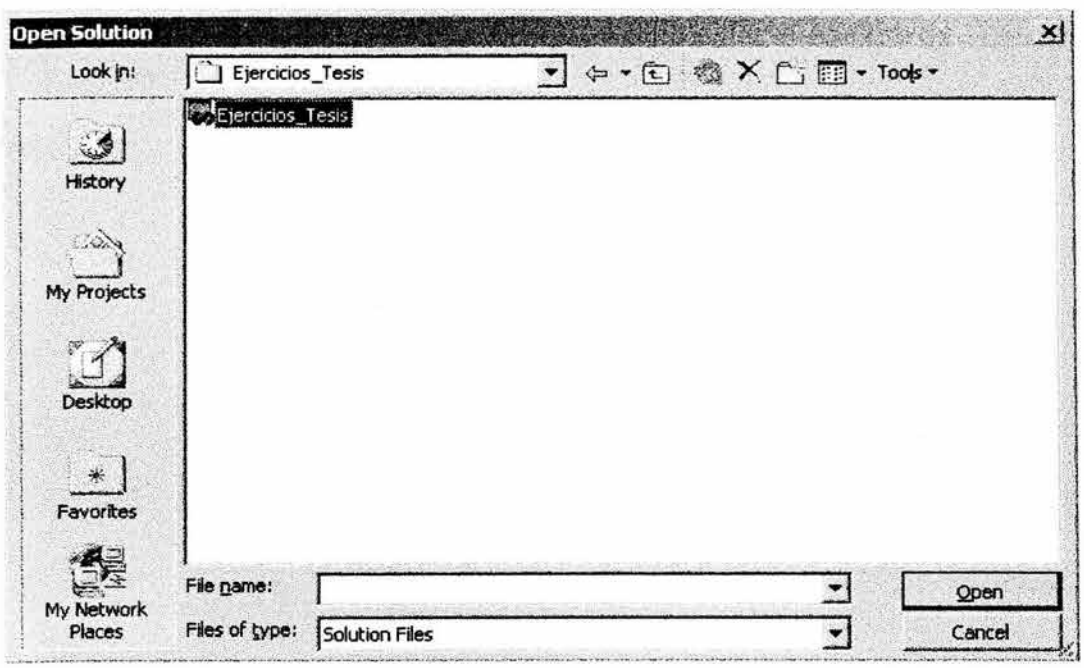
http://localhost/Ejercicios_Tesis/Mi_FormaWeb1.aspx

http://localhost/Ejercicios_Tesis/Mi_FormaWeb2.aspx

1.12.9 Agregar una forma Web

En caso de tener cerrado el proyecto **Ejercicios_Tesis**, ábralo de la siguiente forma: oprima *File* + *Open Solution*, generalmente se encuentra en la carpeta **Visual Studio Projects** dentro de la carpeta **Mis documentos**, véase la figura 1.21.

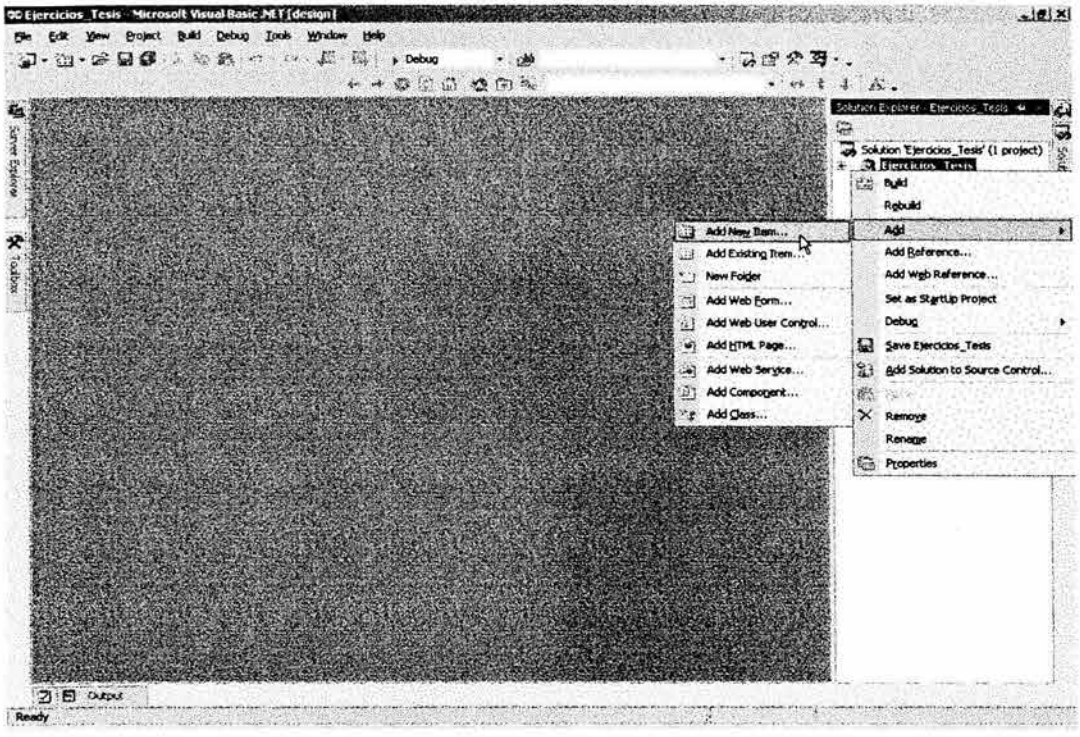
Figura 1.21: Crear un nuevo proyecto ASP.NET



Fuente: Aplicación creada por el autor de esta tesis.

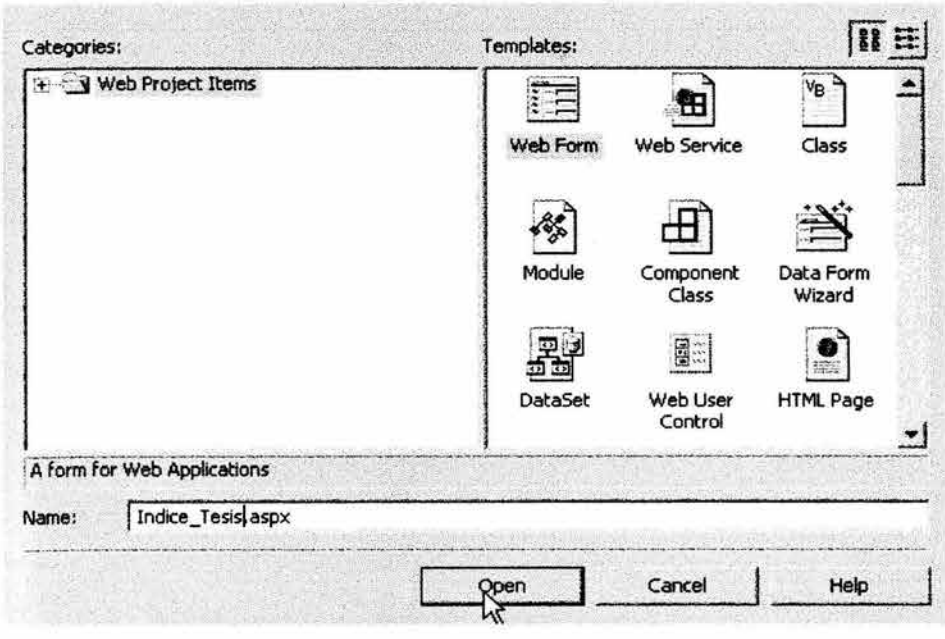
Cuando tenga el proyecto abierto, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add* + *Add New Item*, véase figura 1.22. Entonces se abre una forma que le pide que seleccione el nuevo objeto, seleccione **WebForm** y escriba el nombre **Indice_Tesis.aspx**, que es el nombre que le he dado a la forma en este ejercicio, véase figura 1.23.

Figura 1.22: Agregar una forma Web paso uno



Fuente: Aplicación creada por el autor de esta tesis.

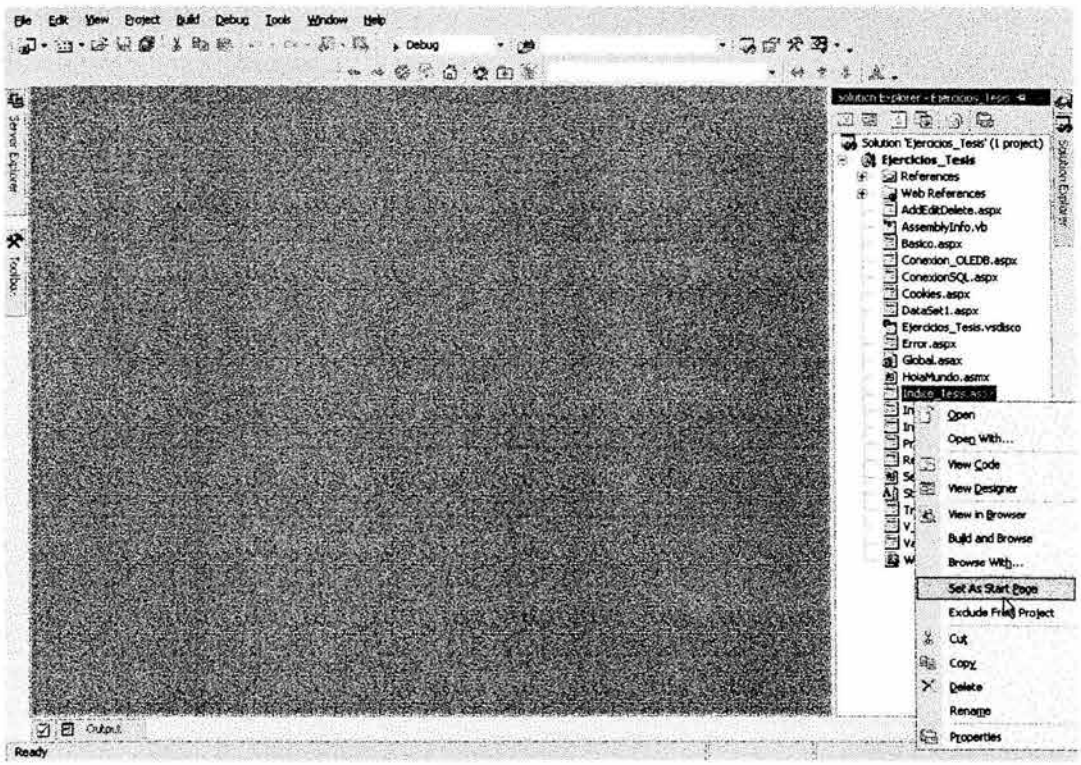
Figura 1.23: Agregar una forma Web paso dos



Fuente: Aplicación creada por el autor de esta tesis.

Diríjase a *Solution Explorer* y pulse con el botón derecho sobre esta forma, seleccionando *Set As Start Page*, véase figura 1.24. Todo proyecto necesita una página Web de inicio, en este caso **Indice_Tesis.aspx**.

Figura 1.24: Indice_Tesis como forma Web de inicio



Fuente: Aplicación creada por el autor de esta tesis.

1.12.10 Crear un menú principal

Es recomendable crear un menú principal, donde por medio de *HyperLinks* ó ligas se pueda dirigir a las páginas solicitadas. El control *HyperLink* tiene una propiedad llamada *NavigateUrl*, la cual contiene una dirección a dónde dirigirse en caso que se oprima la liga.

HyperLink1.NavigateUrl="http://localhost/Mi_Proyecto/mi_direccion.aspx"

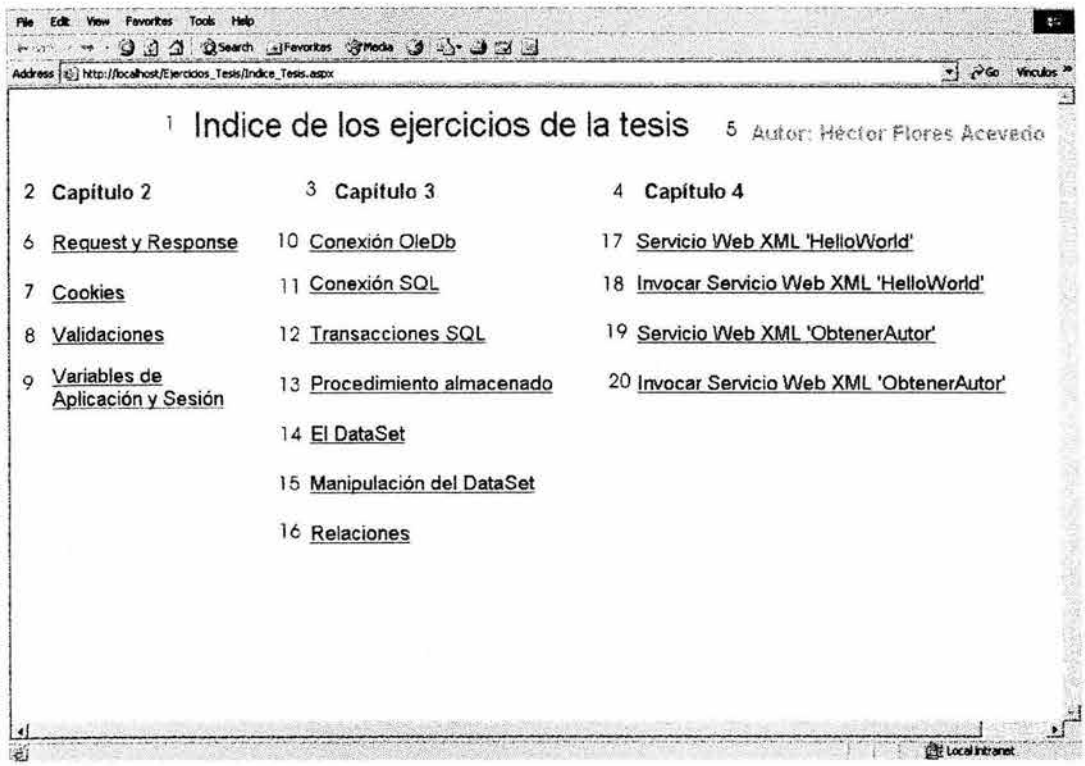
En la forma Web *Indice_Tesis.aspx* agregue los siguientes controles:

1) Label1, 2) Label2, 3) Label3, 4) Label4, 5) Label5 6) HyperLink1, 7) HyperLink2, 8) HyperLink3, 9) HyperLink4, 10) HyperLink5, 11) HyperLink6, 12)

HyperLink7, 13) HyperLink8, 14) HyperLink9, 15) HyperLink10, 16) HyperLink11, 17) HyperLink12, 18) HyperLink13, 19) HyperLink14, 20) HyperLink15.

De tal forma que queden acomodados como se muestra en la figura 1.25. Las propiedades de cada control como son: nombre, color, fuente, etc. Se escribirán dentro del código, no en la caja de propiedades, por lo que usted solamente verá los controles con sus valores por defecto. El escribir el código de configuración de cada control, lo hago para que usted sepa hacerlo sin utilizar la caja de propiedades.

Figura 1.25: Controles del menú principal



Fuente: Aplicación creada por el autor de esta tesis.

Para escribir el código de la forma Web, hacemos doble clic sobre ésta. Automáticamente se posiciona en el archivo con extensión **.aspx.vb**, que es donde se encuentra el código Visual Basic.NET. Posiciónese en el nodo que tiene la etiqueta *Web Form Designer Generated Code*, haga clic. Sitúese en el procedimiento ***Page_Init***, que como usted recuerda es el primero que se invoca cuando se llama la página Web. El código completo queda de la siguiente forma:

Código de Indice_Tesis.aspx

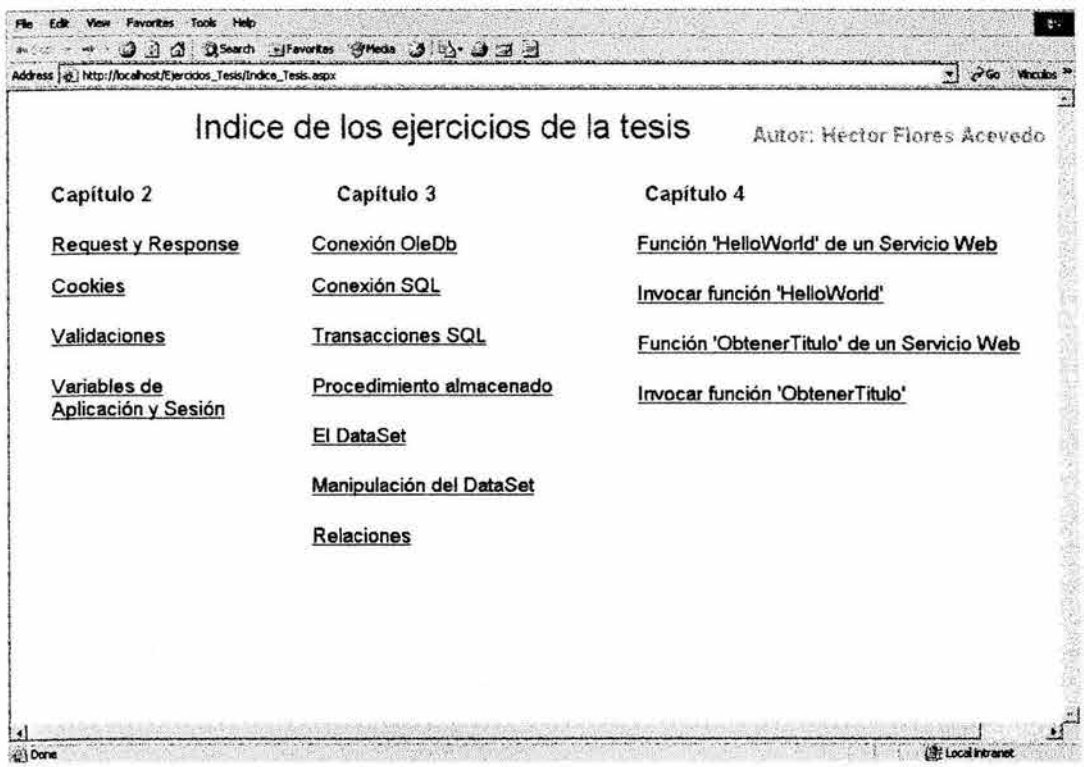
```
Public Class Indice_Tesis
    Inherits System.Web.UI.Page
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents HyperLink2 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink3 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink4 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink5 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink6 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink7 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink8 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink9 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink10 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink11 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink12 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents HyperLink13 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink14 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents HyperLink15 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Request y Response"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Basico.aspx"

        HyperLink2.Font.Size = FontUnit.Parse(14)
        HyperLink2.Font.Name = "Arial"
        HyperLink2.Text = "Cookies"
        HyperLink2.NavigateUrl = "http://localhost/Ejercicios_Tesis/Cookies.aspx"

        HyperLink3.Font.Size = FontUnit.Parse(14)
        HyperLink3.Font.Name = "Arial"
        HyperLink3.Text = "Validaciones"
        HyperLink3.NavigateUrl = "http://localhost/Ejercicios_Tesis/Validators.aspx"

        HyperLink4.Font.Size = FontUnit.Parse(14)
        HyperLink4.Font.Name = "Arial"
        HyperLink4.Text = "Variables de Aplicación y Sesión"
        HyperLink4.NavigateUrl = _
        "http://localhost/Ejercicios_Tesis/V_Aplicacion_Sesion.aspx"

        HyperLink5.Font.Size = FontUnit.Parse(14)
        HyperLink5.Font.Name = "Arial"
        HyperLink5.Text = "Conexión OleDb"
        HyperLink5.NavigateUrl = _
        "http://localhost/Ejercicios_Tesis/Conexion_OLEDB.aspx"

        HyperLink6.Font.Size = FontUnit.Parse(14)
        HyperLink6.Font.Name = "Arial"
        HyperLink6.Text = "Conexión SQL"
        HyperLink6.NavigateUrl = "http://localhost/Ejercicios_Tesis/ConexionSQL.aspx"

        HyperLink7.Font.Size = FontUnit.Parse(14)
        HyperLink7.Font.Name = "Arial"
        HyperLink7.Text = "Transacciones SQL"
        HyperLink7.NavigateUrl = _
        "http://localhost/Ejercicios_Tesis/Transacciones_SQL.aspx"

        HyperLink8.Font.Size = FontUnit.Parse(14)
```

```
        HyperLink8.Font.Name = "Arial"
        HyperLink8.Text = "Procedimiento almacenado"
        HyperLink8.NavigateUrl = "http://localhost/Ejercicios_Tesis/ProcAlmac.aspx"

        HyperLink9.Font.Size = FontUnit.Parse(14)
        HyperLink9.Font.Name = "Arial"
        HyperLink9.Text = "El DataSet"
        HyperLink9.NavigateUrl = "http://localhost/Ejercicios_Tesis/DataSet1.aspx"

        HyperLink10.Font.Size = FontUnit.Parse(14)
        HyperLink10.Font.Name = "Arial"
        HyperLink10.Text = "Manipulación del DataSet"
        HyperLink10.NavigateUrl = "http://localhost/Ejercicios_Tesis/AddEditDelete.aspx"

        HyperLink11.Font.Size = FontUnit.Parse(14)
        HyperLink11.Font.Name = "Arial"
        HyperLink11.Text = "Relaciones"
        HyperLink11.NavigateUrl = "http://localhost/Ejercicios_Tesis/Relaciones.aspx"

        HyperLink12.Font.Size = FontUnit.Parse(14)
        HyperLink12.Font.Name = "Arial"
        HyperLink12.Text = "Función 'HelloWorld' de un Servicio Web"
        HyperLink12.NavigateUrl = "http://localhost/Ejercicios_Tesis/HolaMundo.asmx"

        HyperLink13.Font.Size = FontUnit.Parse(14)
        HyperLink13.Font.Name = "Arial"
        HyperLink13.Text = "Invocar función 'HelloWorld'"
        HyperLink13.NavigateUrl = _
        "http://localhost/Ejercicios_Tesis/InvocarHolaMundo.aspx"

        HyperLink14.Font.Size = FontUnit.Parse(14)
        HyperLink14.Font.Name = "Arial"
        HyperLink14.Text = "Función 'ObtenerTitulo' de un Servicio Web"
        HyperLink14.NavigateUrl = "http://localhost/Ejercicios_Tesis/Service1.asmx"

        HyperLink15.Font.Size = FontUnit.Parse(14)
        HyperLink15.Font.Name = "Arial"
        HyperLink15.Text = "Invocar función 'ObtenerTitulo'"
        HyperLink15.NavigateUrl = "http://localhost/Ejercicios_Tesis/InvocarWSXML.aspx"

        Label1.Text = "Indice de los ejercicios de la tesis"
        Label1.Font.Size = FontUnit.Parse(25)
        'Etiqueta de color verde
        Label1.ForeColor = Color.Green
        Label1.Font.Name = "Arial"

        Label2.Text = "Capítulo 2"
        Label2.Font.Size = FontUnit.Parse(15)
        Label2.Font.Bold = True
        'Etiqueta de color verde
        Label2.ForeColor = Color.Red
        Label2.Font.Name = "Arial"

        Label3.Text = "Capítulo 3"
        Label3.Font.Bold = True
        Label3.Font.Size = FontUnit.Parse(15)
        'Etiqueta de color verde
        Label3.ForeColor = Color.Red
        Label3.Font.Name = "Arial"

        Label4.Text = "Capítulo 4"
        Label4.Font.Bold = True
        Label4.Font.Size = FontUnit.Parse(15)
        'Etiqueta de color verde
        Label4.ForeColor = Color.Red
        Label4.Font.Name = "Arial"

        Label5.Text = "Autor: Héctor Flores Acevedo"
        Label5.Font.Bold = True
        Label5.Font.Size = FontUnit.Parse(15)
        Label5.ForeColor = Color.CadetBlue
        Label5.Font.Name = "Arial"
    End Sub

#End Region
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load
```

```
        'Put user code to initialize the page here
    End Sub
End Class
```

Para compilar la aplicación oprima *Debug + Start*, también puede oprimir la tecla *F5* para la misma operación. En los siguientes ejercicios de este trabajo de tesis, se agregarán las formas Web que correspondan con los *HyperLinks*, cabe mencionar que el nombre de estas formas Web, debe coincidir con la dirección escrita en la propiedad *NavigateUrl*. El resultado final se muestra en la figura 1.26.

Figura 1.26: Menú principal Indice_Tesis.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

1.12.11 Contactar al autor

En *asunto* favor escribir **tesis_ASP**, antes de mandar algún correo revise la documentación que le proporciono y los sitios que le recomiendo, sea paciente en esperar la respuesta, gracias. Héctor Flores Acevedo, **hectorf44@yahoo.com**.

1.13 Conclusión del capítulo

La plataforma .NET propone muchas ventajas, tanto para el programador como para el usuario final. El centralizar el *Runtime* por medio del CLR (Common Language Runtime) y las clases, de alguna forma ayuda a que la migración entre lenguajes de Microsoft sea muy sencilla. El código administrado, permite que las aplicaciones se ejecuten en forma segura, esto no quiere decir que no existirán errores de sintaxis, sino que antes de ejecutar una aplicación, el compilador instantáneo ó JIT, revisará que se encuentren todas las condiciones necesarias para ejecutar el programa. Los ensamblados tienen la propiedad de poderse autodefinir, por lo que no es necesario registrarlos en la máquina a utilizarse, lo que libera a las aplicaciones del uso del archivo de registro. El CLR permite que varios ensamblados se ejecuten dentro de un solo proceso, evitando sobrecargar la máquina realizando un proceso por aplicación.

El recolector de basura es un avance importante, porque libera al programador de tener que administrar la memoria de la computadora, la infraestructura se encarga de esto. La librería de clases de la infraestructura .NET, permite al desarrollador utilizar una gran variedad de clases, que dan funcionalidades a las aplicaciones, reduciendo la cantidad de código y aumentado su productividad. ADO.NET ofrece características muy útiles, es un modelo que permite desconectarse del origen de datos una vez que no requiera más información, reconectándose cuando sea necesario. Permite guardar la estructura completa de una base de datos, así como soporte para XML.

ASP.NET permite desarrollar aplicaciones en Internet, utilizando cualquier lenguaje de Visual Studio .NET, por lo que un desarrollador que programe aplicaciones Windows con Visual Basic.NET, C#, Visual C++, J#, no tendrá que aprender un nuevo lenguaje para desarrollar aplicaciones en Internet. Por lo que la migración de Windows hacia Internet es inmediata. Los Servicios Web XML proponen un nuevo modelo para compartir ensamblados por medio de Internet. Anteriormente era difícil trabajar con componentes distribuidos, la gran ventaja del Servicio Web XML es que cuando es consumido, lo único que se transporta es XML (extensible markup language), que es un lenguaje de la misma familia de HTML y que puede ser entendido por cualquier sistema operativo, con esto se tiene un ensamblado que puede comunicarse con cualquier otra computadora y que puede publicarse en Internet.

CAPÍTULO II

ASP.NET

2.1 Antecedentes

Internet se ha convertido en una herramienta para la humanidad, el desarrollo de aplicaciones para la Web es la meta de todo programador de sistemas de información. La idea de tener una gran red en la cual se puedan compartir archivos y realizar comercio electrónico, es hoy una realidad.

Primera generación

Desde sus inicios se propusieron soluciones para el desarrollador, la primera generación se basó en HTML y el protocolo de transferencia de texto HTTP. En donde lo único que se transfería eran páginas estáticas las cuales no tenían interacción dinámica con el usuario.

Segunda generación

En la segunda generación se incluyeron los controles ActiveX, DHTML o HTML dinámico, Applets, Java Script, VbScript, los cuales se aplicaban en la parte del navegador de Internet para evitar dar vueltas innecesarias al servidor.

En la parte del servidor también se incluyeron nuevas tecnologías, en donde ciertas partes de la página se podían procesar en el servidor y después regresarlas al navegador de Internet.

Compendio de tecnologías utilizadas en la segunda generación de Internet.[2]

- **ASP**. Tecnología de Microsoft que ofrece acceso programático a los recursos del servidor, y adicionalmente brinda la posibilidad de resolver secciones de una página antes de que esta sea enviada al cliente.
- **HTTP o Hyper Text Transfer Protocol**. Protocolo utilizado para la transferencia de información a través de Internet.
- **HTML.** Estándar de marcación utilizado en Internet, que permite definir en un mismo archivo los datos y la presentación de una página.

---

[2] BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 687.

- **VBScript, JavaScript, JScript.** Lenguajes desarrollados para la utilización en productos para Internet, los cuales resultan ser un subconjunto de los lenguajes originales.
- **PHP, Perl.** Lenguajes exclusivamente desarrollados para la creación de páginas dinámicas. Son muy parecidos a C, y pueden ser empleados en IIS mediante la inclusión de agregados provistos por terceros.
- **JSP.** Tecnología abierta que ofrece acceso programático a los recursos del servidor, y que adicionalmente ofrece la posibilidad de resolver secciones de una página antes de que ésta sea enviada al cliente. La principal diferencia con ASP reside en que existen varios proveedores que ofrecen soporte para la misma.
- **Applet, Control ActiveX (ahora ensamblados).** Módulos independientes de código capaces de ser transferidos a través de la red y ejecutados por el explorador. El primero puede ser utilizado por cualquiera que cuente con una máquina virtual de Java, mientras que los restantes requieren de un sistema operativo Windows e Internet Explorer.

Tercera generación

En esta generación se agregan herramientas para el desarrollo de aplicaciones en dispositivos móviles. Se crean los Servicios Web XML y los servicios de XML Voice.

2.2 ¿Qué es ASP.NET?

ASP.NET o Active Server Pages.NET son documentos de texto que tienen secciones a ser reemplazadas por el servidor cuando el usuario realiza una petición.

ASP.NET consta de dos archivos independientes: uno para el formulario y otro para el código. Un formulario Web es similar a un formulario Windows además de ser muy sencillo, ya que con solo arrastrar los controles sobre éste se puede crear una interfaz gráfica.

En lo que respecta a la codificación es muy similar a como se hace en la programación bajo Windows. El archivo que contiene el código se compila como un ensamblado, también permite el uso de *espacios de nombres*, para que se incluyan clases en el código.

Se puede decir que ASP.NET reúne un conjunto de tecnologías que facilitan el desarrollo de aplicaciones para la Web.

2.3 Formularios Web

Cuando se programa en Windows todo el código se encuentra dentro de la computadora del usuario. Así, cuando oprimimos un botón se ejecuta el código asociado con éste. En ASP.NET funciona de la siguiente forma, cuando se oprime un botón en un formulario Web, como el código del evento se encuentra en el servidor, el método **post** de la etiqueta HTML <form> manda el contenido de las cajas de texto hacia la página destino.

Ejemplo:

```
<form name="Forma" method="post" action="DestinoWeb.aspx" id="Forma">
```

Toda ejecución de un evento se realiza en el servidor y no en la computadora cliente y cada vez que un evento sea generado, la página completa se regenera totalmente. No todos los eventos son enviados en el instante en que se producen, sino que se ponen en una cola de espera, esperando mandarse conjuntamente con otros eventos para evitar dar doble vuelta, ya que cada vez que se hace un viaje al servidor genera trafico, que se trata de minimizar.

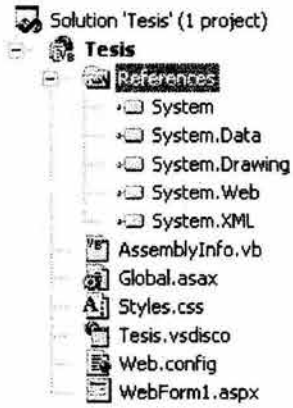
2.4 Componentes de un proyecto ASP.NET

Cada que se crea un proyecto tipo aplicación Web en Visual Studio.NET, se generan automáticamente varios módulos, como se muestran en la figura 2.1.

- **Carpeta de referencia.** Carpeta en donde están las referencias, ensamblados o *espacios de nombres* necesarios para que la aplicación funcione, además se pueden agregar nuestros propios *espacios de nombre*, véase figura 2.2.
- **AssemblyInfo.vb.** Contiene información sobre el proyecto, como su nombre y su descripción para después ser escrita en el manifiesto.

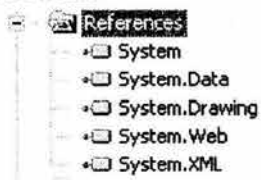
- **<nombre del proyecto>.vsdisco.** Este es un módulo de descubrimiento, su utilidad se encuentra dentro de los Servicios Web XML. Por medio de este módulo se pueden conocer los servicios que otorga la aplicación.

Figura 2.1: Vista del explorador de soluciones



Fuente: Proyecto de Visual Basic.NET de Microsoft Visual Studio.NET

Figura 2.2: Vista de la carpeta de referencias



Fuente: Proyecto de Visual Basic.NET de Microsoft Visual Studio.NET

- **Web.config.** Como su nombre lo dice *config* viene de configuración, este módulo se encarga de establecer la configuración del proyecto al momento de la ejecución. A continuación se presentan algunos valores que están dentro de este módulo.
  a) **FileEncoding.** Se encarga de especificar el tipo de codificación que se utilizará en ASP.NET el cual será necesario para leer las páginas con extensión ***aspx***.
  b) **RequestEncoding.** Se encarga de indicar la codificación que tendrán los paquetes de información que entran del explorador al servidor, por lo general por el método *Post*.
  c) **ResponseEncoding.** Se encarga de indicar la codificación que tendrán los paquetes de información que salen hacia el explorador.

d) **Culture.** Se encarga de indicar funciones y tipos específicos de un idioma o un país, como son los formatos de fecha, hora, etc.

e) **UiCulture.** Se refiere a *user interface culture* o la cultura a utilizar en los controles de la forma.

- **Style.css.** Debido a que todos los exploradores interpretarán la información de una página en forma parecida, pero no de la misma forma, es posible adicionar información para que los elementos de la página puedan ser vistos en su forma original.
- **Global.asax y Global.asax.vb** Estos archivos definen varios eventos globales, los cuales son iniciados cuando se ejecuta la aplicación.
- **Webform.aspx.** Este módulo contiene la información de la representación gráfica de la página Web.
- **WebForm.aspx.vb.** Este módulo contiene el código de los eventos que se puedan generar en la aplicación, recordemos que la parte gráfica y el código se encuentran en archivos separados, por esta razón se puede hacer un ensamblado solamente con el código, y no con la parte gráfica.

2.5 Funcionalidades exclusivas de ASP.NET

Se puede utilizar el *espacio de nombres System.Web* para poder disponer de una variedad de clases que darán funcionalidad a nuestras aplicaciones Web.

- **HttpResponse.** Cuando se necesita mandar información al cliente como *cookies*, salida HTML y direcciones se utiliza la clase *HttpResponse* mediante el objeto *Response* y el método *Write*.

```
Response.Write("La fecha es: " & Now.Date)
```

HttpRequest. Cuando se solicita información del explorador de una computadora cliente, tal como *cookies*, valores de las formas o una consulta se utiliza la clase *HttpRequest* mediante el objeto *Request*. Cabe mencionar que en ASP.NET muchas de las funciones de solicitud se hacen en forma automática, por esto se puede prescindir de éste para obtener valores, aunque sí se utiliza para algunas otras funciones.

```
Valor=Request.Form("txtCaja.Text")
```

- **IsClientConnected( ) As Bolean.** Debido a que los procesos se realizan en el servidor, éstos pueden tomar tiempo y es necesario saber si el usuario sigue conectado, la función *IsClienteConnected* regresa un valor verdadero o falso que indica el resultado, se utiliza en conjunción con el objeto *Response*.

```
If Response.IsClientConnected Then
  'Se realiza un proceso en caso de que el usuario este conectado.
End If
```

- **Objeto.Redirect("http://paginaWeb").** El objeto *Response* tiene un método llamado *Redirect* el cual dirige la página actual hacia otra; sin embargo, hay que tomar en cuenta que cuando el navegador de Internet encuentra una página que dirige hacia otra, el servidor envía la orden: *"se debe de ir a otra página"*, después de esto el explorador toma la dirección destino y va hacia esa página, por lo tanto este método ocasiona una ida y vuelta extra al servidor.

```
Response.Redirect("http://www.miNuevaPagina.com")
```

2.6 Cómo obtener variables del servidor

El servidor cuenta con variables especiales que pueden ser consultadas por el usuario, estas variables pueden regresarnos información como por ejemplo la versión de navegador de Internet del que se está haciendo la petición, la versión del servidor Web, etc. La propiedad *ServerVariables* se utiliza para esta operación.

Por ejemplo, para obtener el navegador utilizado:

```
Request.ServerVariables("HTTP_USER_AGENT")
```

Para obtener la versión de servidor

```
Request.ServerVariables("SERVER_SOFTWARE")
```

Si queremos saber todas las variables de la propiedad *ServerVariables*, asignamos a una variable la siguiente instrucción:

```
Variable=Request.ServerVariables(0)
```

Ejercicio 2.1

Objetivo:

Que usted se familiarice con las instrucciones *Response* y *Request*, se pretende que al oprimir el primer botón se muestre en pantalla el mensaje *¿Hola cómo estás?*

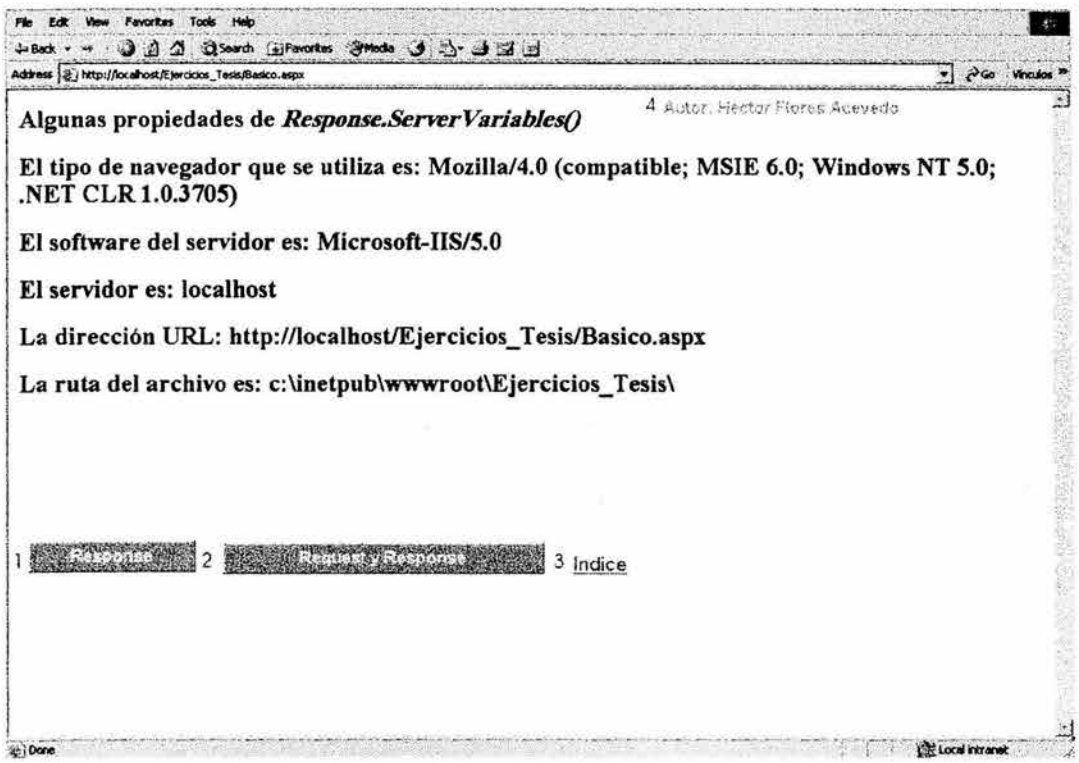
Al oprimir el segundo botón se muestren algunas de las propiedades de la instrucción *Server Variables*.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **Basico.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 2.3: 1) Button1, 2) Button2, 3) HyperLink1, 4) Label1.

Figura 2.3: Controles de la forma Web Basico.aspx



Fuente: Aplicación creada por el autor de esta tesis.

Escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Request y Response*.

Código de Basico.aspx

```
Public Class Basico
    Inherits System.Web.UI.Page
    Protected WithEvents Button2 As System.Web.UI.WebControls.Button
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
     Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        Label1.Text = "Autor: Héctor Flores Acevedo"
        Label1.Font.Bold = True
        Label1.Font.Size = FontUnit.Parse(12)
        Label1.ForeColor = Color.CadetBlue
        Label1.Font.Name = "Arial"

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Button1.Text = "Response"
        Button1.Font.Name = "Arial"
        Button1.Font.Size = FontUnit.Parse(12)
        Button1.BackColor = Color.CadetBlue
        Button1.Font.Bold = True
        Button1.ForeColor = Color.White

        Button2.Text = "Request y Response"
        Button2.Font.Name = "Arial"
        Button2.Font.Size = FontUnit.Parse(12)
        Button2.BackColor = Color.CadetBlue
        Button2.Font.Bold = True
        Button2.ForeColor = Color.White


    End Sub

#End Region
    'Procedimiento que se genera cada que se regenera la página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load
        'Put user code to initialize the page here
    End Sub

    'Procedimiento que se ejecutará cuando se hace clic en el botón de Response
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button1.Click

        'Mando escribir en pantalla un mensaje
        Response.Write("<H2>¿Hola cómo estás?</H2>")

    End Sub

    'Procedimiento que se ejecutará cuando se hace clic en el botón de Response y Request
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button2.Click

        'Declaro una variable de tipo Object como auxiliar
        Dim Valor As Object

        'Mando escribir un mensaje a la pantalla del navegador de Internet
        Response.Write("<H2>Algunas propiedades de <I>Response.ServerVariables()</I></H2>")
```

```
        'Mando escribir el tipo de navegador que se utiliza
        Valor = Request.ServerVariables("HTTP_USER_AGENT")
        Response.Write("<H2>El tipo de navegador que se utiliza es: " & Valor & "</H2>")

        'Mando escribir el tipo de software que utiliza el navegador
        Valor = Request.ServerVariables("SERVER_SOFTWARE")
        Response.Write("<H2>El software del servidor es: " & Valor & "</H2>")

        'Mando escribir el servidor que manda la información
        Valor = Request.ServerVariables("HTTP_HOST")
        Response.Write("<H2>El servidor es: " & Valor & "</H2>")

        'Mando escribir la dirección URL
        Valor = Request.ServerVariables("HTTP_REFERER")
        Response.Write("<H2>La dirección URL: " & Valor & "</H2>")

        'Mando escribir la ruta del archivo
        Valor = Request.ServerVariables("APPL_PHYSICAL_PATH")
        Response.Write("<H2>La ruta del archivo es: " & Valor & "</H2>")
    End Sub
End Class
```

2.7 Gestión de estado

Debido a que las páginas Web son recreadas totalmente cada vez que un navegador de Internet las solicita, necesitamos tener una forma de mantener el estado de la página Web.

La clase *StateBag* tiene una propiedad llamada *ViewState*, que permite mantener un valor por múltiples pedidos.

```
ViewState("Dato")=30
```

En caso de querer obtener el valor de esa variable:

```
Response.Write(ViewState("Dato"))
```

Al momento de mandar la página al explorador, la variable se incluye en forma escondida

```
<input type="hidden" name="__VIEWSTATE" value="Do…NnU=" />
```

Erich R. Bühler[3] comenta acerca de *ViewState*: *"La utilización de un campo oculto con el estado de los controles de la página es una solución ingeniosa, ya que no emplea recursos del servidor como era el caso en las variables de sesión, por lo que no cuenta con las restricciones de las mismas. No obstante, una página con muchos controles podría tener un campo ViewState demasiado extenso, lo que daría como resultado que la página fuese más grande. Sin embargo, Microsoft afirma que éste no*

[3] BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 793.

es un punto importante a tener en cuenta, ya que no afecta notoriamente al rendimiento final de la transferencia de la misma."

2.8 Cookies

Una *cookie* sirve para poder guardar alguna información en forma de texto en la computadora del cliente, para que en futuras ocasiones pueda ser leída por el servidor. La clase *HttpCookie* tiene varias propiedades como *Expires, Value, Name*, las cuales se pueden acceder por *Response* o por *Request*.

Este método tiene sus ventajas debido a que libera al servidor de guardar datos sobre las preferencias de un usuario. Se puede poner una fecha límite después de la cual la *cookie* se eliminará, se puede determinar el alcance de la variable a determinadas páginas, el valor máximo que puede almacenar una *cookie* es de 8192 bytes. Las *cookies* tienen estas desventajas:

- Solamente se pueden almacenar cadenas de texto.
- El usuario puede modificar o eliminar una *cookie*.
- Existen navegadores de Internet que no permiten su escritura.

Para crear una cookie escribiríamos el siguiente código:

```
Dim Cookie as New HttpCookie("Nombre")
Cookie.Value="Juan Pérez"

'quiere decir que expirará 30 días después de esta fecha.
Cookie.Expires=now.AddDays(30)

'Mando como respuesta esa cookie.
Response.Cookies.Add(Cookie)
```

Otra forma sería la siguiente:

```
Response.Cookies("NombreDeUsuario").Value="Juan Pérez"

'La cookie tendrá una vigencia de un mes a partir de la fecha actual.
Response.Cookies("NombreDeUsuario").Expires=Now.AddMonths(1)
```

Ejercicio 2.2

Objetivo:

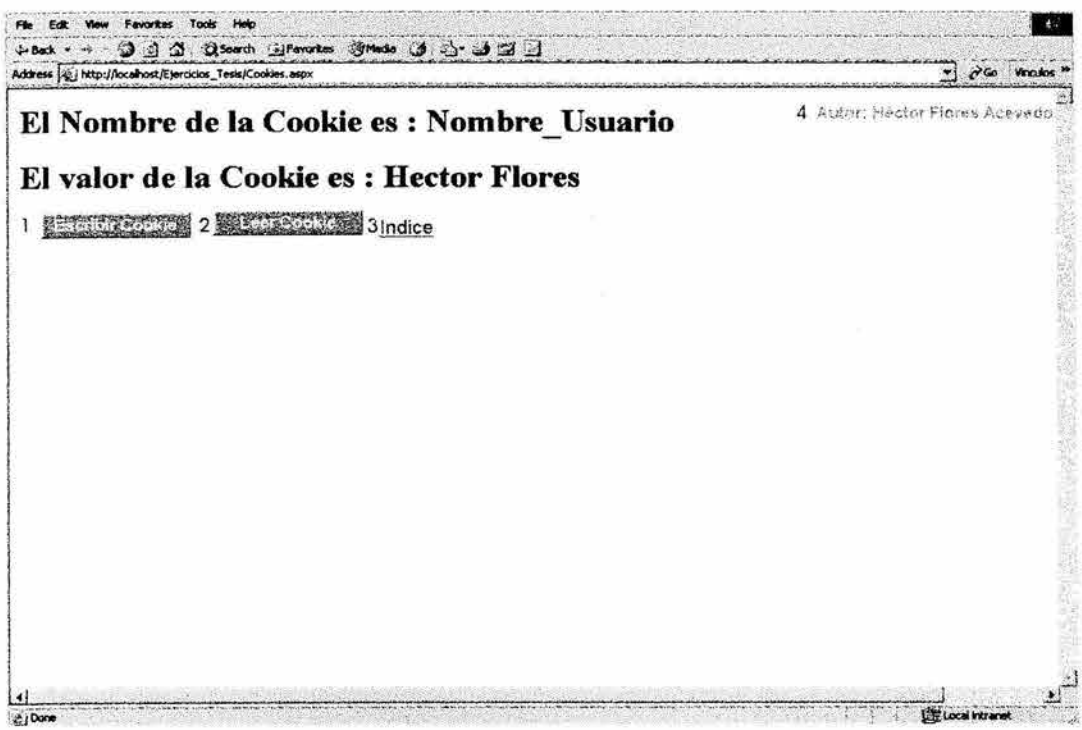
Al oprimir el primer botón se guardará el valor *Héctor Flores* dentro de la variable *Nombre_Autor*, además de darle una vigencia de un mes. En caso de que ya exista la *Cookie*, no permitirá que se vuelva a guardar. Al oprimir el segundo botón, se leerá el contenido de la *Cookie* y el nombre de la variable.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add* + *Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **Cookies.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 2.4: 1) Button1, 2) Button2, 3) HyperLink1, 4) Label1.

Figura 2.4: Controles de la forma Web Cookies.aspx



Fuente: Aplicación creada por el autor de esta tesis.

Escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Cookies*.

Código de Cookies.aspx

```
Public Class Cookies
    Inherits System.Web.UI.Page
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Button2 As System.Web.UI.WebControls.Button

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro cada una de las propiedades de los controles a utilizar

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Button1.Text = "Escribir Cookie"
        Button1.Font.Name = "Arial"
        Button1.Font.Size = FontUnit.Parse(12)
        Button1.BackColor = Color.CadetBlue
        Button1.Font.Bold = True
        Button1.ForeColor = Color.White

        Button2.Text = "Leer Cookie"
        Button2.Font.Name = "Arial"
        Button2.Font.Size = FontUnit.Parse(12)
        Button2.BackColor = Color.CadetBlue
        Button2.Font.Bold = True
        Button2.ForeColor = Color.White

        Label1.Text = "Autor: Héctor Flores Acevedo"
        Label1.Font.Bold = True
        Label1.Font.Size = FontUnit.Parse(12)
        Label1.ForeColor = Color.CadetBlue
        Label1.Font.Name = "Arial"

    End Sub

#End Region

    'Procedimiento que se genera cuando se carga una página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load
        'Put user code to initialize the page here
    End Sub

    'Procedimiento que se genera cuando se quiere escribir una cookie
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button1.Click

        'En caso de que la Cookie no exista entonces escribela
        If (Request.Cookies("Nombre_Usuario") Is Nothing) Then

            'Escribe el valor de la Cookie
            Response.Cookies("Nombre_Usuario").Value = "Hector Flores"
```

```
            'Indico la fecha de terminación
            Response.Cookies("Nombre_Usuario").Expires = Now.AddMonths(1)
            Response.Write("<H1>Se ha escrito la Cookie</H1>")
        Else
            'En caso de que sí exista la Cookie
            Response.Write("<H1>La Cookie ya existe, no la puedes sobreescribir</H1>")
        End If
    End Sub

    'Procedimiento que se genera cuando se quiere leer una cookie
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button2.Click

        'Declaro una variable de tipo Object
        Dim Valor As Object

        'En caso de que tenga un valor la Cookie "Nombre_Usuario"
        If Not (Request.Cookies("Nombre_Usuario") Is Nothing) Then

            'Nombre de la Cookie
            Valor = Request.Cookies("Nombre_Usuario").Name
            Response.Write("<H1>El Nombre de la Cookie es : " & Valor & "</H1>")

            'Valor de la Cookie
            Valor = Request.Cookies("Nombre_Usuario").Value
            Response.Write("<H1>El valor de la Cookie es : " & Valor & "</H1>")

        Else

            'En caso de que no exista la Cookie
            Response.Write("<H1>No existe la cookie</H1>")

        End If
    End Sub

End Class
```

2.9 Variables de aplicación

Como lo explicamos antes, cada que una página es solicitada se regenera completamente. Es por eso que no guarda valores, y se utilizan las variables de aplicación, en las que no se permiten almacenar valores que puedan ser compartidos por todos los usuarios que acceden a la aplicación.

El objeto *Application* permite que se almacene información dentro de la memoria del servidor Web, mientras que el servicio de IIS (Internet Information Server) no sea detenido.

Por ejemplo, podemos indicar en qué momento se inició la aplicación y almacenarlo en una variable llamada *T_DeComienzo*.

```
'Now indica la fecha y hora actual del servidor.
Application("T_DeComienzo")=Now
```

Para acceder a esta variable lo podemos hacer de la siguiente forma:

```
Response.Write("La aplicación comenzó: " & Application("T_DeComienzo"))
```

Escribirá en la pantalla del navegador de Internet la fecha en que comenzó la aplicación.

También podríamos almacenar el número de visitantes que ha tenido nuestra página, para comenzar necesitamos asignar algún valor a la variable.

```
Application.Add("Clientes",2)
```

Para obtener el valor de esta variable podemos utilizar el método *Get* del objeto *Application*, cabe mencionar que no es necesario declarar previamente la variable.

```
Variable=Application.Get("Clientes")
```

2.10 Lock y Unlock

Los miembros *Lock* y *Unlock* se utilizan para que no existan problemas de concurrencia, ya que se debe alterar una variable por sesión a la vez.

Ejemplo:

```
'Pongo un candado para que nadie más, además del que accedió primero,
'pueda alterar la variable. Además de incrementar en uno el contador
Application.Lock  Application("Contador")=Application("Contador") + 1

'Quito el candado para que otra sesión pueda aumentar el contador
Application.Unlock
```

2.11 Variables de sesión

Cuando se necesita guardar información por usuario, entonces se utilizan las *variables de sesión*. La diferencia de este tipo de variables contra las de aplicación es que las variables de sesión tienen un conjunto de valores por usuario y las de aplicación son globales, las comparten todos los usuarios que acceden a la aplicación.

El valor de estas variables son mantenidas mientras la sesión no termine, ya sea que el tiempo máximo de espera se alcance o el usuario cambie a otra página que no pertenezca a la misma aplicación.

Es importante establecer lo que es una sesión para ASP.NET, cada ventana del navegador de Internet abierta es interpretada como una sesión. Por ejemplo, si tenemos tres ventanas del navegador abiertas en la misma computadora, será interpretada por ASP.NET como tres sesiones abiertas, sin embargo generalmente existe una sola ventana abierta del navegador de Internet por usuario.

Para guardar la fecha y hora en que comenzó la sesión.

```
Session("InicioDeSesión")=Now
```

Para obtener el valor.

```
Response.Write("La sesión comenzó: " & Session("InicioDeSesión"))
```

Si quisiera establecer el color de fondo de cada usuario se almacenaría de la siguiente forma:

```
Session.Add("Color","Verde")
```

Para obtener el valor:

```
ColorDeUsuario=Session("Color")
```

El estado de una sesión se puede almacenar de las siguientes formas:

1. Como parte del proceso de la aplicación.
2. Como un servicio de Microsoft Windows, en el que se permite a la aplicación ASP.NET compartir datos entre muchos procesadores (*Web gardens*).
3. Como dato en una base de datos de SQL Server, lo que permite que se pueda compartir la información entre muchas computadoras (*Web farms*).

Es importante mencionar que ASP.NET no necesita *cookies* para guardar el estado de sesión, en el archivo *Web.Config* se puede especificar que se codifique el identificador de sesión en el URL (Uniform Resource Locator), con lo cual cualquier navegador de Internet, sin importar que no permita el uso de *cookies,* puede saber la información de sesión.

2.12 Inicialización con Global.asax

Es posible inicializar variables de aplicación o de sesión en el archivo *Global.asax*, si se necesita guardar información se utilizan variables de aplicación o de sesión. En ASP.NET existen varios eventos en la vida de la aplicación, algunos de éstos son:

- **Eventos de aplicación.** Los eventos de aplicación son aquellos que son compartidos por todos los usuarios.

  Application_Start. Se ocasiona cuando el primer usuario trata por primera vez de ingresar a la aplicación.

Application_BeginRequest. Cuando un usuario solicita un URL.

Application_EndRequest. Cuando la solicitud ha sido completada.

Application_End. Se activa cuando ya no hay instancias de la clase Global.

- **Eventos de sesión.** Los eventos de sesión son aquellos que son exclusivos a un usuario.

  Session_Start. Es activado cuando una nueva sesión se va a generar.

  Session_End. Se activa cuando termina una sesión, por cualquiera de las siguientes razones:

  - Cuando se invoca el método ***Session.Abandon*** dentro del código de la aplicación.
  - Cuando se ha excedido el tiempo de inactividad. Este tiempo es establecido automáticamente en veinte minutos, sin embargo se puede configurar como usted lo desee, por medio de la propiedad ***Session.Timeout***. Hay que tomar en cuenta, que la configuración de un tiempo muy largo consume mayores recursos del servidor.

Ejercicio 2.3

Objetivo:

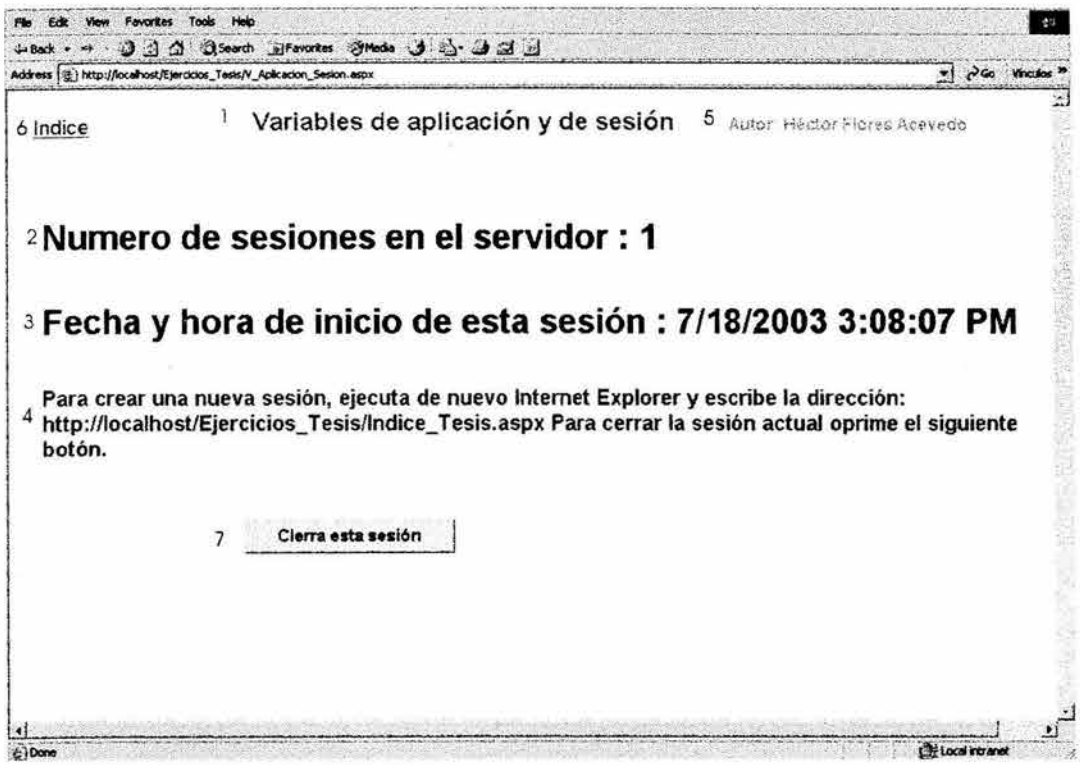
Almacenar una variable de aplicación llamada ***Contador_Aplicacion***, que cuente el número de sesiones activas que tiene la aplicación actualmente. Almacenar una variable de sesión llamada ***Hora_Inicio,*** que guarde la fecha y hora en que se inicia cada sesión. Para dar termino a la sesión, se encuentra un botón que invoca a la función ***Session.Abandon()***.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **V_Aplicacion_Sesion.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 2.5: 1) Label1, 2) Label2, 3) Label3, 4) Label4, 5) Label5, 6) HyperLink1, 7) Button1.

Figura 2.5: Controles de la forma Web V_Aplicacion_Sesion.aspx



Fuente: Aplicación creada por el autor de esta tesis.

Recuerde que para ASP.NET, cada ventana del navegador de Internet que acceda a la aplicación es considerada como una nueva sesión. Cuando oprime dentro del navegador File + New + Window, se abre una segunda ventana hija de la ventana original, en este caso no se genera una nueva sesión. Solamente se genera una nueva cuando oprime el icono de Internet Explorer, deshabilita la opción Work Offline del menú File + Work Offline, y escribe el URL:

http://localhost/Ejercicios_Tesis/V_Aplicacion_Sesion.aspx

El código de este ejercicio se escribe en dos archivos, el primero en *Global.asax* y el segundo en *V_Aplicacion_Sesion.aspx*. En el primer archivo se inicializa a cero la variable *Contador_Aplicacion* dentro de la función *Application_Start*, se hace el incremento de esta variable cuando inicia una sesión dentro de la función

Session_Start, se decrementa cuando termina una sesión dentro de la función *Session_End*. La fecha y hora en que comienza la sesión se guarda en la variable de sesión *Hora_Inicio*.

Código de Global.asax

```
Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
    ' Se genera cuando la aplicación comienza
    'Inicio la variable Contador_Aplicacion con cero
    Application("Contador_Aplicacion") = 0
End Sub

Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
    ' Se genera cuando una sesión comienza
    'Le sumo uno al contador
    Application("Contador_Aplicacion") = Int(Application("Contador_Aplicacion")) + 1

    'Indico la hora en que comenzó la sesión
    Session("Hora_Inicio") = Now
End Sub

Sub Application_BeginRequest(ByVal sender As Object, ByVal e As EventArgs)
    ' Se genera al principio de un pedido

End Sub

Sub Application_AuthenticateRequest(ByVal sender As Object, ByVal e As EventArgs)
    ' Se genera cuando se pide una autentificación
End Sub

Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
    ' Se genera cuando hay un error
End Sub

Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
    ' Se genera cuando la sesión termina, ya sea por un tiempo de inactividad
    ' o porque el metodo Session.Abandon sea llamado.
    ' Le resto uno al contador
    Application("Contador_Aplicacion") = Int(Application("Contador_Aplicacion")) - 1
End Sub

Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
    ' Se genera cuando la aplicación termina
End Sub
```

Escriba el siguiente código dentro del archivo *V_Aplicacion_Sesion.aspx*, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Variables de Aplicación y Sesión*.

Código de V_Aplicacion_Sesion.aspx

```
'Este ejercicio está relacionado con el archivo Global.axa
'porque allí se declaran las variables de aplicación y de sesión

Public Class V_Aplicacion_Sesion
    Inherits System.Web.UI.Page
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
```

```
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro las propiedades de cada control

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Label1.Text = "Variables de aplicación y de sesión"
        Label1.Font.Name = "Arial"
        Label1.ForeColor = Color.Green
        Label1.Font.Size = FontUnit.Parse(18)
        Label1.Font.Bold = True

        Label2.Text = String.Empty
        Label2.Font.Name = "Arial"
        Label2.ForeColor = Color.DarkBlue
        Label2.Font.Size = FontUnit.Parse(25)
        Label2.Font.Bold = True

        Label3.Text = String.Empty
        Label3.Font.Name = "Arial"
        Label3.ForeColor = Color.DarkBlue
        Label3.Font.Size = FontUnit.Parse(25)
        Label3.Font.Bold = True

        Label4.Text = "Para crear una nueva sesión, ejecuta de nuevo Internet Explorer " & _
        "y escribe la dirección: http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx " & _
        "Para cerrar la sesión actual oprime el siguiente botón."
        Label4.Font.Name = "Arial"
        Label4.ForeColor = Color.DarkGreen
        Label4.Font.Size = FontUnit.Parse(15)
        Label4.Font.Bold = True

        Label5.Text = "Autor: Héctor Flores Acevedo"
        Label5.Font.Bold = True
        Label5.Font.Size = FontUnit.Parse(12)
        Label5.ForeColor = Color.CadetBlue
        Label5.Font.Name = "Arial"

        Button1.Text = "Cierra esta sesión"
        Button1.Font.Bold = True
        Button1.Font.Size = FontUnit.Parse(12)
        Button1.ForeColor = Color.DarkBlue
        Button1.BackColor = Color.Beige
        Button1.Font.Name = "Arial"

    End Sub

#End Region

    'Procedimiento que se genera cuando se carga una página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load

        Dim Contador As Object
        Dim Hora As Date

        'Bloqueo el acceso a los demás usuarios, para que sea el unico en acceder
        'a la variable Contador_Aplicacion.
        Application.Lock()
```

```
        'Obtengo el valor de la variable de aplicación 'Contador_Aplicacion'
        Contador = Application.Get("Contador_Aplicacion")

        'Desbloqueo el acceso a la variable, para que otros usuarios puedan accederla
        Application.UnLock()

        Hora = Session("Hora_Inicio")

        'Mando imprimir el número de sesión, la hora y fecha de inicio
        Label2.Text = "Numero de sesiones en el servidor : " & Contador
        Label3.Text = "Fecha y hora de inicio de esta sesión : " & Hora

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button1.Click
        'Termina la sesión
        Session.Abandon()
        ' Deshabilito el control
        Button1.Enabled = False
    End Sub
End Class
```

2.13 Configuración de aplicaciones ASP.NET

Como lo mencionamos antes existe un archivo llamado *Web.config*, que contiene información acerca de la configuración de la aplicación. El formato del texto es XML (Extensible Markup Language), que es un lenguaje similar a HTML que nos permite definir nuestras propias etiquetas. El archivo *Web.config* se debe instalar en el mismo directorio virtual de la aplicación.

La forma de instalar aplicaciones ASP.NET se realiza copiando los archivos necesarios al directorio virtual de la computadora que tiene el IIS (Internet Information Server), el registro de los componentes no es necesario debido a que dentro de él va contenida toda la información necesaria para su descripción, lo cual elimina los problemas del registro de los componentes utilizados en la aplicación.

Cuando se actualiza una aplicación ASP.NET no es necesario volver a reiniciar el servicio del servidor Web o IIS, la plataforma .NET detecta la actualización y la ejecuta sin tener que alterar el estado actual del servidor.

Es importante mencionar que se debe de instalar previamente la infraestructua .NET o .NET Framework, que se encuentra en el sitio Web de Microsoft.

http://msdn.microsoft.com/netframework/downloads/howtoget.aspx

2.14 Seguridad en ASP.NET

La seguridad es una parte esencial en toda aplicación Web, se debe de autentificar a los usuarios, validar tarjetas de crédito, transacciones, etc. El espacio de nombres *System.Web.Security* se encarga de la seguridad en ASP.NET.

El archivo *Web.config* puede tener información sobre la seguridad en la aplicación, se pueden establecer opciones de seguridad. Es posible guardar en este documento con formato XML el acceso a usuarios por ejemplo:

```
<authentification mode="Windows"/>
<authorization>
  <allow users="Juan,Pedro" />
  <deny users="*" />
  <allow roles="Admins" />
  <deny roles="*" />
</authorization>
```

IIS 5.0 y ASP.NET pueden validar a los usuarios por medio de los siguientes métodos:

- **Autentificación de Windows.** No es necesario codificar en la aplicación la autentificación, ya que se realiza conjuntamente con el servidor IIS 5.0 y Windows. Hay tres formas de autentificación:
  1) ***Basic***. La información no se encripta, antes de llegar al servidor Web aparece una caja de diálogo que pide el nombre de usuario y password.
  2) ***Digest***. Es casi igual a la anterior sin embargo la información se encripta antes de llegar al servidor Web.
  3) ***Integrated***. Esta opción es más recomendable para una Intranet que para Internet, ya que se utiliza como identificación la cuenta de dominio de Windows.
- **Autentificación por medio de *Microsoft Passport*.** Este tipo de autentificación utiliza la autentificación que se utiliza en *Microsoft Passport*, todos los sitios que utilicen esta autentificación tendrán el mismo nombre de usuario y password, por lo que es recomendable.
- **Autentificación por medio de *Cookies*.** La autentificación por medio de *cookie* consiste en dejar un archivo de texto en la máquina cliente con información del usuario, para que cuando se acceda de nuevo a la página, se tome la información del usuario. No es muy conveniente este tipo de autentificación debido a que el

usuario puede alterarla, borrarla o simplemente puede habilitar al navegador de Internet para que no reciba *cookies*.

2.15 Estructura de las formas Web

En las aplicaciones desarrolladas bajo Visual Studio.NET el código del lenguaje como funciones, clases, procedimientos, eventos, etc., se encuentra en un archivo. La parte visual como controles y etiquetas HTML, se encuentran en otro. Microsoft hace esta separación para facilitar el desarrollo para el programador.

- **Archivo .aspx.** Contiene todos los elementos visibles de la forma, como son controles Web, etiquetas HTML, etc.
- **Archivo .aspx.vb.** Cuando se utiliza como lenguaje Visual Basic.NET, se agrega esta extensión. En este archivo se encuentran todos los eventos, procedimientos y funciones. También aquí se declaran variables, al ser este archivo únicamente de código se puede compilar automáticamente como ensamblado y utilizar clases previamente creadas.

2.16 Controles de formularios Web

Los controles de los formularios Web, al ser solicitados por el navegador de Internet, se dibujan en la pantalla del navegador para ser la interfaz gráfica. Los elementos que se presentan son controles HTML, dependiendo del software que utilice para la creación de las páginas, lo que al final se traduce es HTML, un control puede formarse por varios elementos HTML. Se puede hacer una separación lógica de los tipos de controles de formularios Web:

- **Estándar.** Son controles HTML sencillos, que se pueden acceder por cualquier navegador de Internet, en caso de que un control no exista, se simula con una combinación de otros controles HTML. Algunos de éstos son:
  1) **TextBox**. Es una caja de texto, que sirve para que el usuario ingrese datos, para después ser mandados al servidor.
  2) **Button**. Se encarga de ejecutar código cuando se oprime.

3) **Hyperlink**. Se encarga de dirigir al usuario a la dirección especificada.
4) **Label**. Es una etiqueta que se encarga de mostrar información.
5) **Image**. Se encarga de mostrar una imagen dentro del formulario Web.
6) **DropDownList**. Lista que permite mostrar opciones al usuario, en forma desplegable.
7) **ListBox**. Caja de texto que tiene una lista de texto donde el usuario puede elegir una opción, la diferencia con la anterior es que es estática, no es desplegable.
8) **CheckBox**. Consiste en una caja de verificación, generalmente se alinean varias cajas por medio del control *CheckBoxList*, pueden haber varias cajas seleccionadas al mismo tiempo.
9) **OptionButton** Es un botón de selección, generalmente se alinean varios botones de opción para seleccionar una opción dentro de varias opciones.
10) **Panel**. Es un contenedor de varios controles dentro de un formulario Web.
11) **LinkButton**. Es un botón que se encarga de mandar al usuario a otra dirección, es similar al control *Hyperlink*.

- **Controles complejos.** Este tipo de controles tienen mas funciones, ya que además de estar formados por varios controles HTML, utilizan recursos del navegador de Internet y del servidor. Algunos de estos controles son:
  1) **Calendar**. Simula un calendario en pantalla, ya sea para que el usuario ponga una fecha o simplemente quiera mostrar una.
  2) **XML**. Permite transformar XSL a un documento XML.
  3) **CrystalReportViewer**. Permite mostrar reportes hechos en *Crystal Reports* (software para crear reportes) en el navegador de Internet.
  4) **AdRotator**. Permite rotar imágenes en pantalla junto con otras opciones adicionales.
- **Controles de validación.** Estos controles se encargan de validar el contenido de los controles de la forma, debido a que gran parte del código en una página Web consiste en validar datos, con estos controles de validación se elimina gran parte de código. Básicamente lo que se hace es enlazarlos con el control a validar, en caso de que los datos no sean válidos aparecerá un texto indicando el error.

  Tipos de controles de validación:

1) **RequiredFieldValidator**. Se encarga de validar si un control tiene contenido.
2) **CompareValidator**. Comprueba que dos controles tengan los mismos valores.
3) **RangeValidator**. Se encarga de validar que un control tenga un valor que se encuentre dentro de un rango preestablecido.
4) **RegularExpressionValidator**. Define un patrón para el texto, como una plantilla para poder escribir correos electrónicos, código postal, etc.
5) **CustomValidator**. Permite al programador establecer una validación que los demás controles no ofrece. Por ejemplo, validar que el número introducido en una caja de texto sea par ó impar. La propiedad ***EnableClientScript=false***, permite que se ejecute código que se encuentra servidor para validar el contenido de un control, si esta propiedad no se deshabilita, cualquier código que se desee ejecutar para validar un control desde el servidor no se ejecutará.
6) **ValidationSummary**. Este control permite mostrar todos los mensajes de error en uno solo. A partir de Internet Explorer 4, es posible configurar este control para que muestre todos los mensajes en un *message box*.

- **Controles enlazados a datos.** Este tipo de controles tiene la propiedad de poderse enlazar con alguna fuente de datos, como puede ser un manejador de bases de datos, sin embargo también permiten que el usuario interactúe con ellos dando altas, bajas, cambios, etc. Son muchas las propiedades de estos controles. Algunos de estos controles son:

1) **DataGrid**.
2) **DataList**.
3) **Repeater**. (Permite únicamente la lectura de datos).

Ejercicio 2.4

Objetivo:

Mostrar el uso de los siguientes controles de validación:

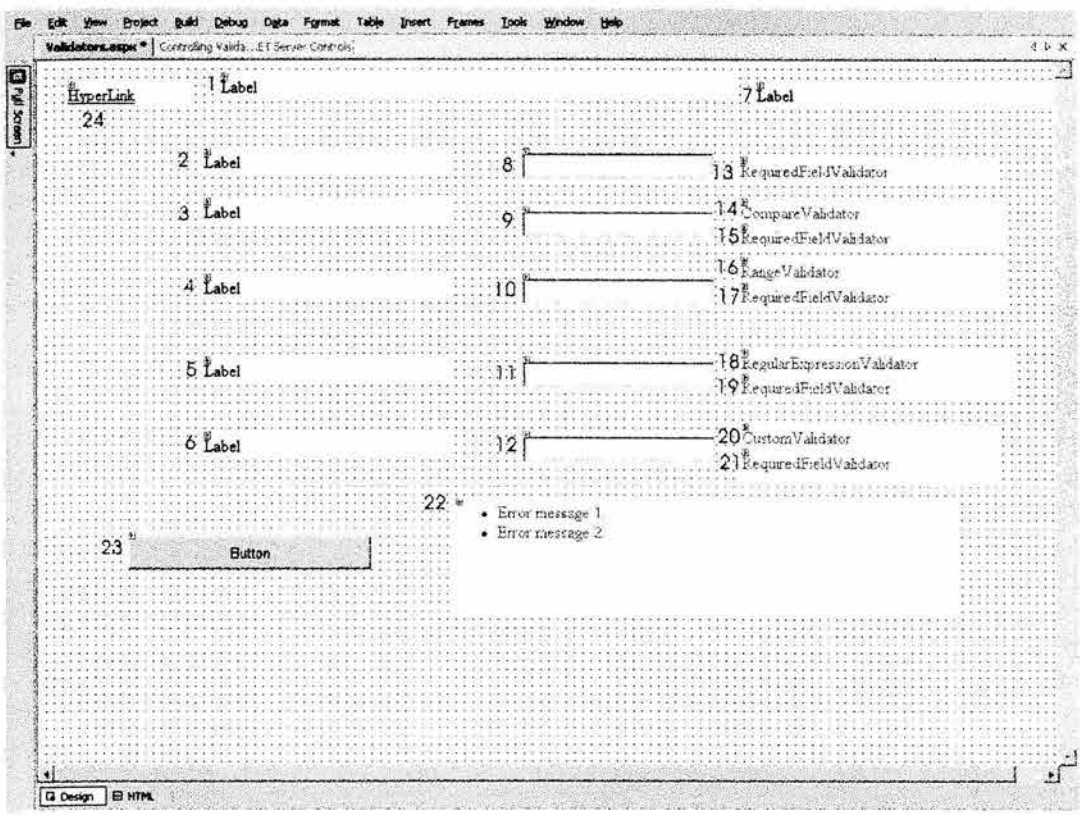
- **RequiredFieldValidator.** Valida que cada caja de texto tenga un contenido.
- **CompareValidator.** Valida que el valor de la segunda caja de texto sea el mismo que el valor de la primera.

- **RegularExpressionValidator.** Valida que el valor de la caja de texto tenga un valor numérico de cinco cifras, porque es un código postal.
- **RangeValidator.** Valida que el valor de la caja de texto tenga un valor numérico entre 0 y 10.
- **CustomValidator.** Valida que el valor de la caja de texto sea par.
- **ValidationSummary.** Contiene todos los mensajes de validación.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add* + *Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **Validators.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Figura 2.6: Controles de la forma Web Vallidators.aspx



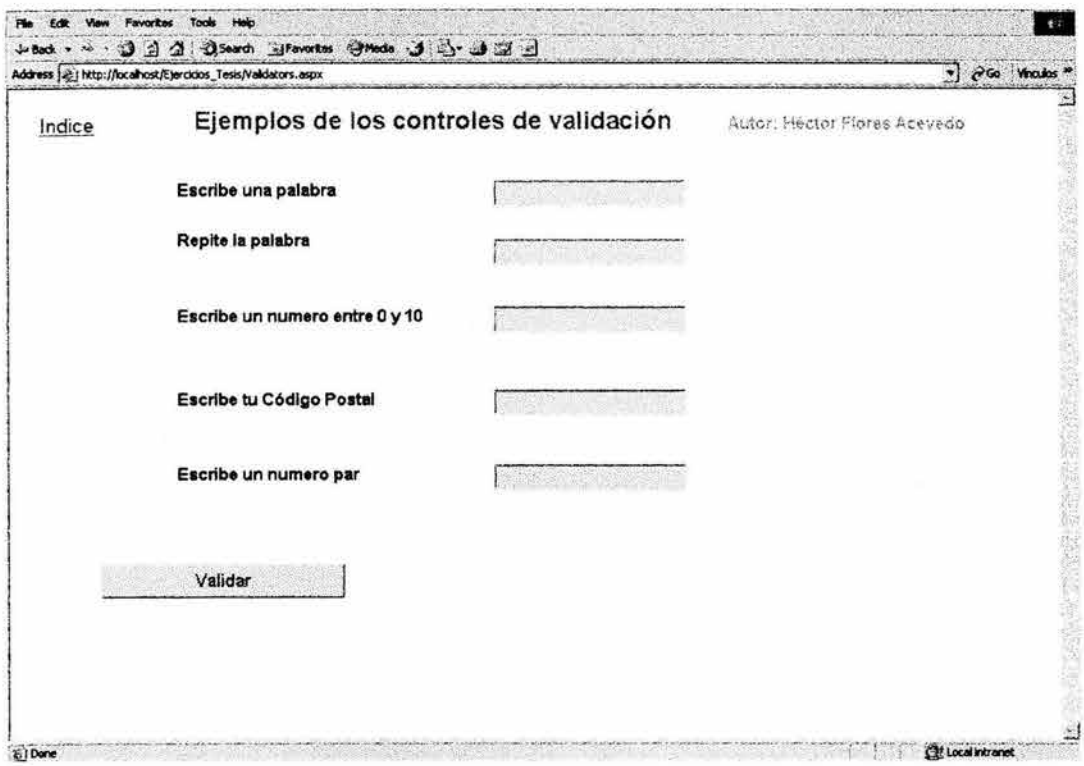
Fuente: Aplicación creada por el autor de esta tesis.

Agregue los siguientes controles ordenados de acuerdo a la figura 2.6:

1)Label1, 2)Label2, 3)Label3, 4)Label4, 5)Label5, 6)Label6, 7)Label7, 8)Textbox1, 9)Textbox2, 10)Textbox3, 11)Textbox4, 12)Textbox5, 13)RequiredFieldValidator1, 14)CompareValidator1, 15)RequiredFieldValidator2, 16)RangeValidator1, 17)RequiredFieldValidator3, 18)RegularExpressionValidator1, 19)RequiredFieldValidator4, 20)CustomValidator1, 21)RequiredFieldValidator5, 22)ValidationSummary1, 23)Button1, 24)HyperLink1.

Escriba el siguiente código dentro del archivo *Validators.aspx*, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Validaciones*, entonces se muestra la forma Web de la figura 2.7.

Figura 2.7: Forma Web Validators.aspx



Fuente: Aplicación creada por el autor de esta tesis.

Código de Validators.aspx

```
Public Class Validators
    Inherits System.Web.UI.Page
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
```

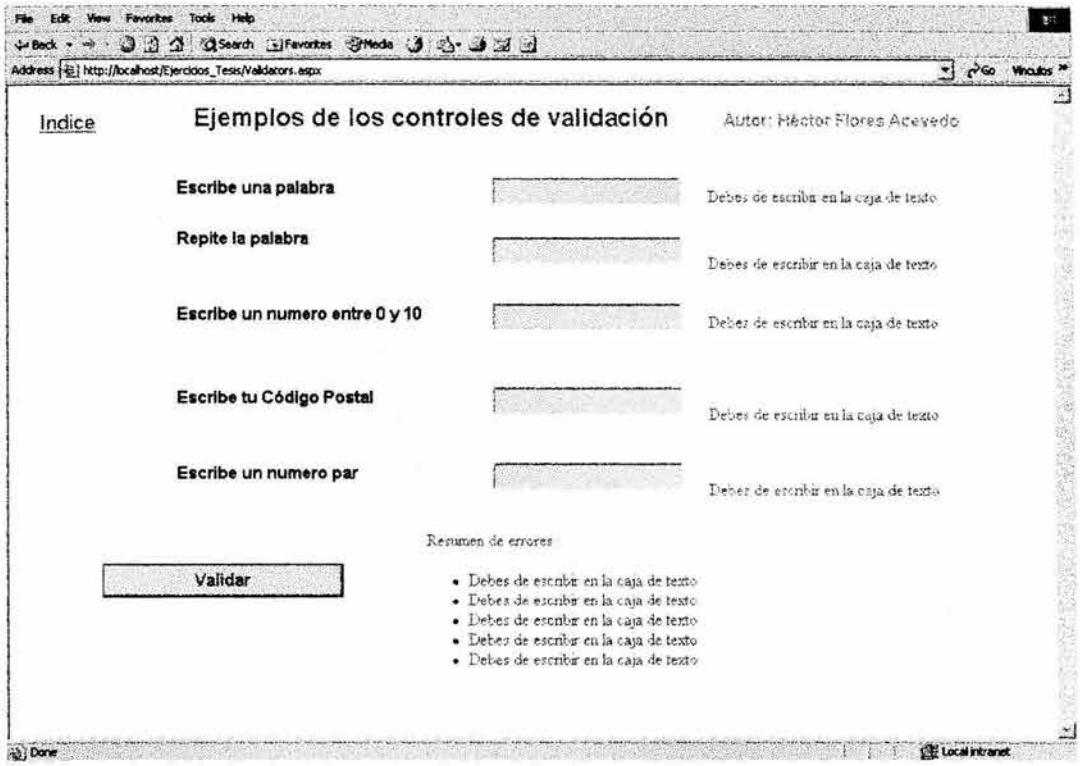
```
    Protected WithEvents TextBox2 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBox3 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBox4 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents Label6 As System.Web.UI.WebControls.Label
    Protected WithEvents CustomValidator1 As System.Web.UI.WebControls.CustomValidator
    Protected WithEvents Textbox5 As System.Web.UI.WebControls.TextBox
    Protected WithEvents RegularExpressionValidator1 As _
                         System.Web.UI.WebControls.RegularExpressionValidator
    Protected WithEvents RangeValidator1 As System.Web.UI.WebControls.RangeValidator
    Protected WithEvents CompareValidator1 As System.Web.UI.WebControls.CompareValidator
    Protected WithEvents RequiredFieldValidator1 As _
                         System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RequiredFieldValidator2 As _
                         System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RequiredFieldValidator3 As _
                         System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RequiredFieldValidator4 As _
                         System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RequiredFieldValidator5 As _
                         System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents ValidationSummary1 As _
                         System.Web.UI.WebControls.ValidationSummary
    Protected WithEvents Label7 As System.Web.UI.WebControls.Label
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro la propiedad de cada control a utilizar

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Label1.Text = "Ejemplos de los controles de validación"
        Label1.Font.Size = FontUnit.Parse(18)
        Label1.ForeColor = Color.Green
        Label1.Font.Name = "Arial"
        Label1.Font.Bold = True

        Label2.Text = "Escribe una palabra"
        Label2.Font.Size = FontUnit.Parse(12)
        Label2.ForeColor = Color.DarkBlue
        Label2.Font.Name = "Arial"
        Label2.Font.Bold = True

        Label3.Text = "Repite la palabra"
        Label3.Font.Size = FontUnit.Parse(12)
        Label3.ForeColor = Color.DarkBlue
        Label3.Font.Name = "Arial"
        Label3.Font.Bold = True

        Label4.Text = "Escribe un numero entre 0 y 10"
        Label4.Font.Size = FontUnit.Parse(12)
        Label4.ForeColor = Color.DarkBlue
        Label4.Font.Name = "Arial"
        Label4.Font.Bold = True

        Label5.Text = "Escribe tu Código Postal"
        Label5.Font.Size = FontUnit.Parse(12)
        Label5.ForeColor = Color.DarkBlue
```

```
Label5.Font.Name = "Arial"
Label5.Font.Bold = True

Label6.Text = "Escribe un numero par"
Label6.Font.Size = FontUnit.Parse(12)
Label6.ForeColor = Color.DarkBlue
Label6.Font.Name = "Arial"
Label6.Font.Bold = True

Label7.Text = "Autor: Héctor Flores Acevedo"
Label7.Font.Bold = True
Label7.Font.Size = FontUnit.Parse(12)
Label7.ForeColor = Color.CadetBlue
Label7.Font.Name = "Arial"

Button1.Text = "Validar"
Button1.Font.Name = "Arial"
Button1.ForeColor = Color.DarkGreen
Button1.BackColor = Color.Wheat
Button1.Font.Bold = True
Button1.Font.Size = FontUnit.Parse(12)

TextBox1.Font.Name = "Arial"
TextBox1.ForeColor = Color.DarkGreen
TextBox1.BackColor = Color.Wheat
TextBox1.Font.Bold = True
TextBox1.Font.Size = FontUnit.Parse(12)

TextBox2.Font.Name = "Arial"
TextBox2.ForeColor = Color.DarkGreen
TextBox2.BackColor = Color.Wheat
TextBox2.Font.Bold = True
TextBox2.Font.Size = FontUnit.Parse(12)

TextBox3.Font.Name = "Arial"
TextBox3.ForeColor = Color.DarkGreen
TextBox3.BackColor = Color.Wheat
TextBox3.Font.Bold = True
TextBox3.Font.Size = FontUnit.Parse(12)

TextBox4.Font.Name = "Arial"
TextBox4.ForeColor = Color.DarkGreen
TextBox4.BackColor = Color.Wheat
TextBox4.Font.Bold = True
TextBox4.Font.Size = FontUnit.Parse(12)

Textbox5.Font.Name = "Arial"
Textbox5.ForeColor = Color.DarkGreen
Textbox5.BackColor = Color.Wheat
Textbox5.Font.Bold = True
Textbox5.Font.Size = FontUnit.Parse(12)

'Este control me permite establecer una fución especifica
'de validación para un caja de texto, que con ningún otro
'control es posible
CustomValidator1.ControlToValidate = "TextBox5"
CustomValidator1.ErrorMessage = "Debes de escribir un numero par"

'Este control permite que se pueda validar una caja de texto,
'con máscaras especificas, como direcciones de correo electrónico,
'cantidad de numeros.
RegularExpressionValidator1.ControlToValidate = "TextBox4"
RegularExpressionValidator1.EnableClientScript = False
RegularExpressionValidator1.ValidationExpression = "\d{5}"
RegularExpressionValidator1.ErrorMessage = "El CP debe tener 5 numeros"

'Este control valida que el valor dentro de la caja de texto
'esté dentro de cierto rango preestablecido
RangeValidator1.ControlToValidate = "TextBox3"
RangeValidator1.ErrorMessage = "El numero debe de ser entre 0 y 10"
'Indica que se introduce numeros
RangeValidator1.Type = ValidationDataType.Integer
RangeValidator1.MinimumValue = 0
RangeValidator1.MaximumValue = 10

'Este control me sirve para comparar dos cajas de texto
CompareValidator1.ErrorMessage = "La caja uno y dos deben ser iguales"
```

```
        CompareValidator1.ControlToCompare = "TextBox1"
        CompareValidator1.ControlToValidate = "TextBox2"
        'Se comparan cadenas de caracteres
        CompareValidator1.Type = ValidationDataType.String
        'Deben de ser iguales
        CompareValidator1.Operator = ValidationCompareOperator.Equal
        CompareValidator1.EnableClientScript = False

        'Este control me sirve para validar que se introduzca un valor en la caja
        'de texto1.
        RequiredFieldValidator1.ErrorMessage = "Debes de escribir en la caja de texto"
        RequiredFieldValidator1.ControlToValidate = "TextBox1"
        RequiredFieldValidator1.EnableClientScript = False

        'Este control me sirve para validar que se introduzca un valor en la caja
        'de texto2.
        RequiredFieldValidator2.ErrorMessage = "Debes de escribir en la caja de texto"
        RequiredFieldValidator2.ControlToValidate = "TextBox2"
        RequiredFieldValidator2.EnableClientScript = False

        'Este control me sirve para validar que se introduzca un valor en la caja
        'de texto3.
        RequiredFieldValidator3.ErrorMessage = "Debes de escribir en la caja de texto"
        RequiredFieldValidator3.ControlToValidate = "TextBox3"
        RequiredFieldValidator3.EnableClientScript = False

        'Este control me sirve para validar que se introduzca un valor en la caja
        'de texto4.
        RequiredFieldValidator4.ErrorMessage = "Debes de escribir en la caja de texto"
        RequiredFieldValidator4.ControlToValidate = "TextBox4"
        RequiredFieldValidator4.EnableClientScript = False

        'Este control me sirve para validar que se introduzca un valor en la caja
        'de texto5.
        RequiredFieldValidator5.ErrorMessage = "Debes de escribir en la caja de texto"
        RequiredFieldValidator5.ControlToValidate = "TextBox5"
        RequiredFieldValidator5.EnableClientScript = False

        'Este control permite que se puedan mostrar todos los errores,
        'en un solo control.
        ValidationSummary1.HeaderText = "Resumen de errores :"
        ValidationSummary1.DisplayMode = ValidationSummaryDisplayMode.BulletList

    End Sub

#End Region

    'Procedimiento que se genera cuando se carga la página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load
        'Put user code to initialize the page here
    End Sub

    'Este procedimiento sirve para establecer una función al control CustomValidator
    Private Sub CustomValidator1_ServerValidate(ByVal source As Object, ByVal args As _
    System.Web.UI.WebControls.ServerValidateEventArgs) Handles _
    CustomValidator1.ServerValidate

        'Con este código checo si el número es par o impar
        If args.Value Mod 2 = 0 Then
            args.IsValid = True
        Else
            args.IsValid = False
        End If
    End Sub
End Class
```

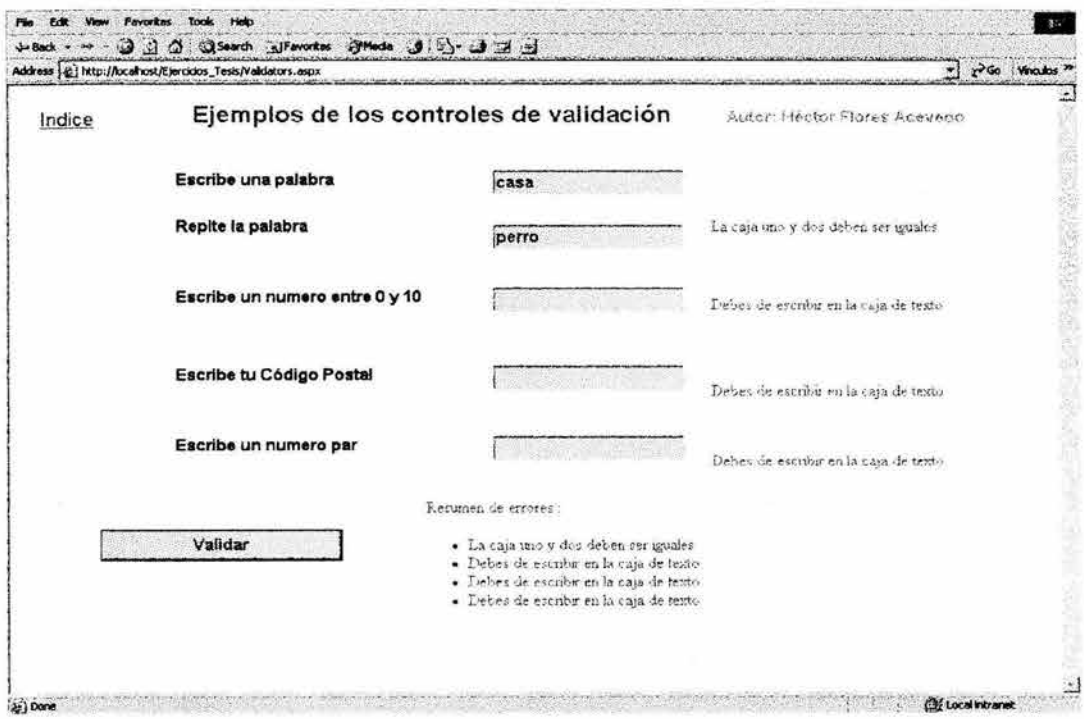
Las diferentes etapas por las que puede pasar la forma Web, se muestran en las siguientes figuras.

Figura 2.8 Se oprime el botón *Validar* y no hay nada escrito en las cajas de texto.



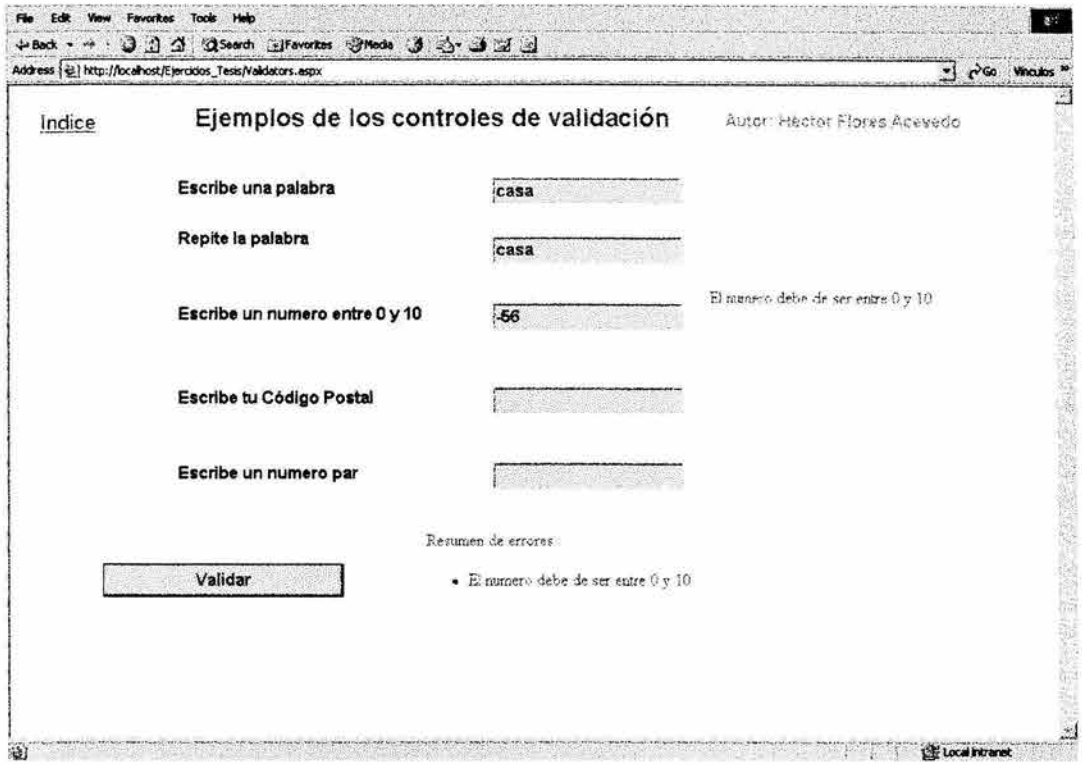
Fuente: Aplicación creada por el autor de esta tesis.

Figura 2.9 La segunda caja de texto no contiene lo mismo que la primer caja.



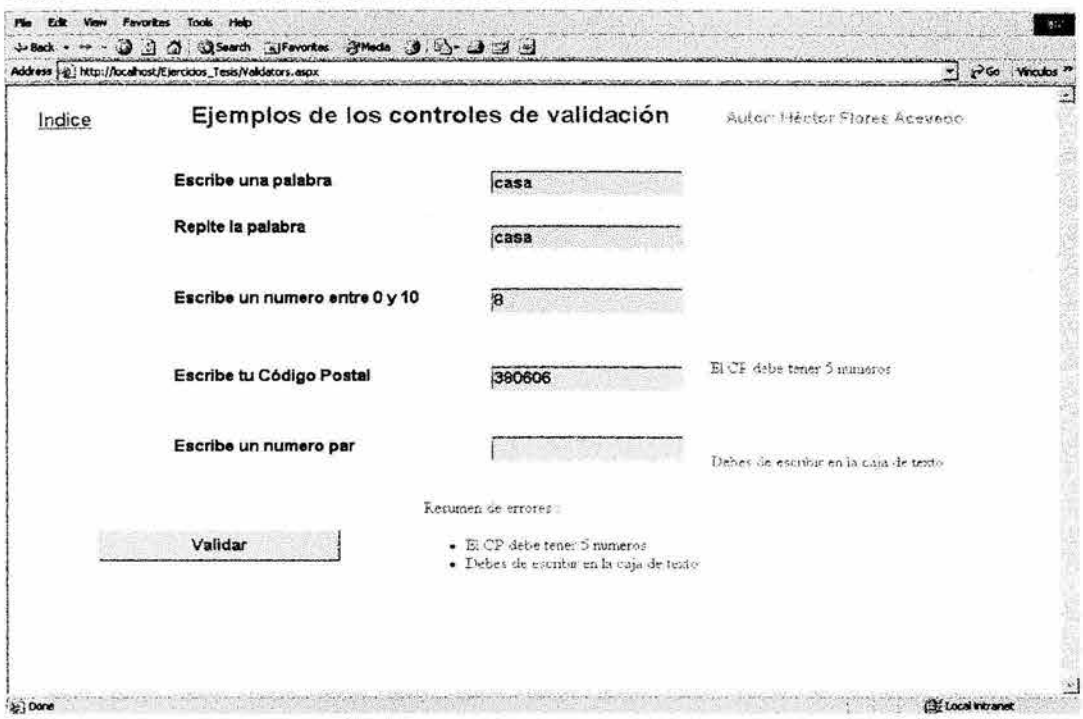
Fuente: Aplicación creada por el autor de esta tesis.

Figura 2.10 El valor de la caja de texto está fuera del rango 0-10.



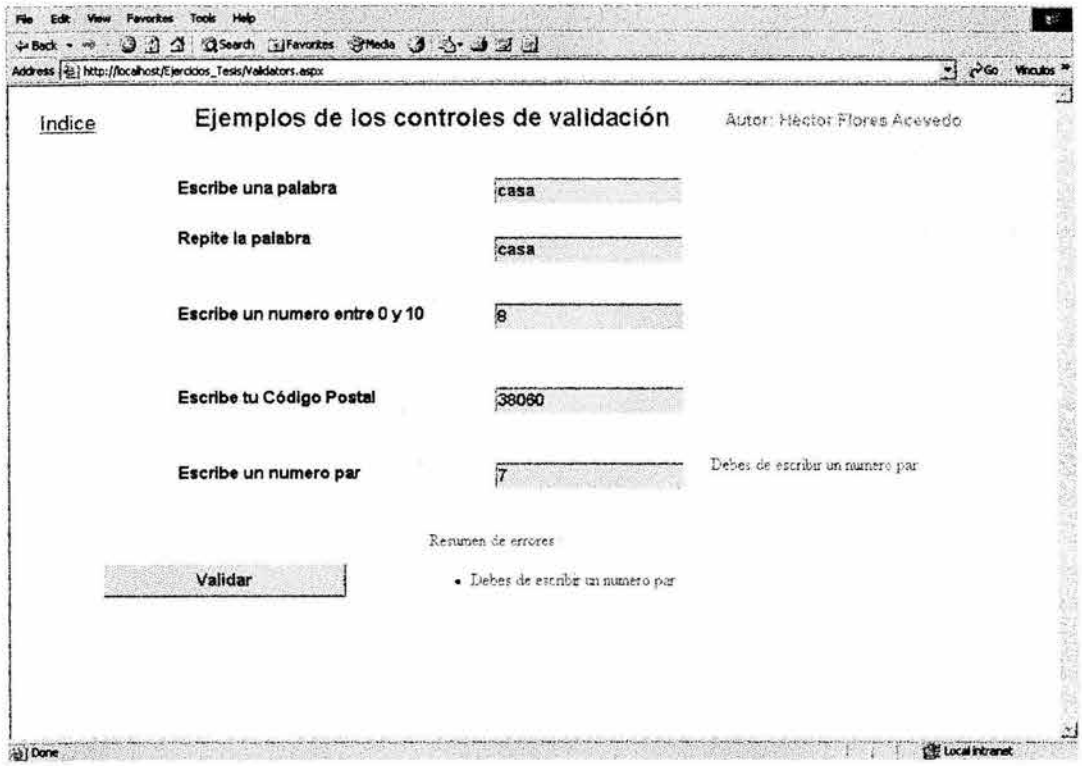
Fuente: Aplicación creada por el autor de esta tesis.

Figura 2.11 El valor de la caja de texto no tiene cinco cifras.



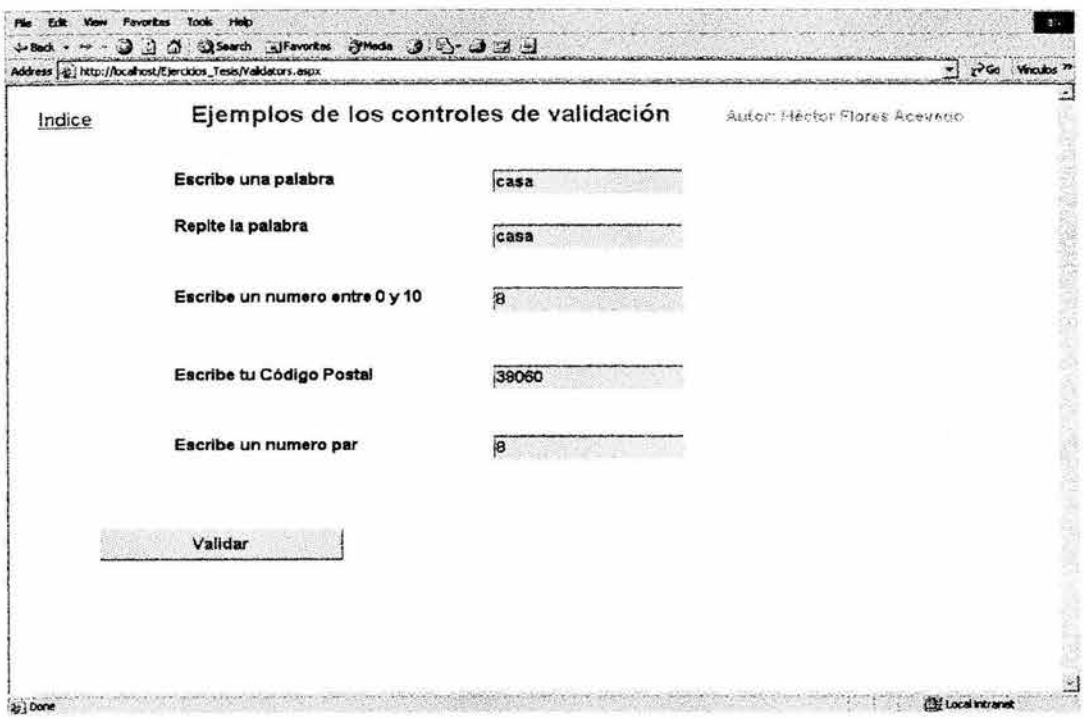
Fuente: Aplicación creada por el autor de esta tesis.

Figura 2.12 El valor de la caja de texto no es un número par.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 2.13 Todos los valores de las cajas de texto son válidos.



Fuente: Aplicación creada por el autor de esta tesis.

2.17 Eventos

Toda interacción que tenga el usuario con la página Web, requiere de código programado para que realice las tareas que se especifican. Existen dos tipos de eventos:

- **Eventos de página o Page Events.** Las páginas Web heredan sus características de la clase *Page*, que permite que dentro de la página puedan existir controles, los eventos propios para la página son:
  1) **Page_Init**. Se genera cuando la página es iniciada y requiere que todos sus controles se configuren.
  2) **Page_Load**. Se genera después del evento *Page_Init*, generalmente se utiliza para inicializar valores y las propiedades de los controles.
  3) **Page_PreRender**. Se genera después de *Page_Load* y antes que *Unload*, la diferencia con *Load* es que los cambios que se generaron en *Page_Load* ya han sido efectuados.
  4) **Page_Unload**. Antes de que la página sea mandada al navegador de Internet, se genera este evento. Generalmente se utiliza para cerrar conexiones de bases de datos y liberación de recursos.
- **Eventos de controles.** Cada control que exista en un formulario tiene su propio evento, para permitir una interacción con el usuario. Dependiendo del valor de *AutoPostBack*, si es verdadero todo evento va directamente al servidor en el instante en que se generó, en caso contrario se pondrá en una cola de espera para evitar dar muchas vueltas, cuando se reúnen un conjunto de eventos, entonces se mandan todos en paquete.

2.18 Conclusión del capítulo

ASP.NET ofrece muchas herramientas con las que podemos desarrollar aplicaciones en la Web de forma sencilla, por ejemplo, los controles enlazables a datos que permiten con muy poco código tener aplicaciones funcionales. Una aplicación ASP.NET se puede compilar como un ensamblado, lo que da total independencia de la parte visual, ya que no tenemos una mezcla de código y de etiquetas HTML.

Anteriormente con ASP era difícil utilizar clases externas o reutilizar código, generalmente se tenía que pegar en varias páginas, lo que generaba código repetido, sin embargo en ASP.NET es posible reutilizar código y hacer una programación similar a como se hace en Windows.

En este capítulo se explica el funcionamiento de ASP.NET, cómo funciona la relación cliente servidor, los procedimientos y funciones que permiten una interacción con el usuario. Es importante subrayar que las aplicaciones en Internet aunque parezcan iguales a las aplicaciones diseñadas para una PC, son diferentes en su estructura interna. Debido a que la seguridad es uno de los factores más importantes, el desarrollador se encuentra limitado cuando desea almacenar valores en las computadoras cliente, es por eso que nacieron las *cookies,* a través del capítulo se dieron varios métodos para poder almacenar configuraciones especiales de usuarios; en Internet es muy utilizado para saber, por ejemplo; que un usuario ya se dió de alta en un sitio y que no es necesario volverlo a registrar, ó inclusive volver a pedir la palabra de acceso.

Es importante entender que la mayoría del código se encuentra en el servidor y que cuando se genera un evento, el servidor es el que realiza la tarea, es por esto que cuando uno se inscribe a un sitio en Internet o checa el correo electrónico, la respuesta a la interacción con los controles de la forma no es inmediata, debido a que tiene que hacer un viaje completo al servidor y regresar con la respuesta. Es probable que en un futuro este sistema mejore, ya sea porque se incremente la velocidad de transmisión o porque el modelo actual evolucione.

En este capítulo se pretendió introducir al lector con la tecnología ASP.NET, para que el desarrollador juzgue por sí mismo sus características. Microsoft siempre ha tratado de ofrecer herramientas fáciles de utilizar, para que la codificación de un proyecto no sea más complejo que el diseño del mismo. Conforme se familiarice con ASP.NET se dará cuenta que es un ambiente amigable y poderoso al mismo tiempo, que permite crear proyectos ambiciosos, con pocas líneas de código.

Hay que recordar que uno de los mayores problemas del diseñador de software, es el tiempo. El contar con funciones que permitan desarrollar aplicaciones con mayor facilidad, sin sacrificar la calidad del producto final, es uno de los mayores factores a tomar en cuenta, cuando se escoge una tecnología y una herramienta del desarrollo.

CAPÍTULO III

ADO.NET

3.1 Introducción a ADO.NET

A través del tiempo los desarrolladores de sistemas de cómputo se han dado cuenta, que una de las arquitecturas más escalables son los sistemas distribuidos.

El programa principal se encuentra en una máquina servidora y por medio de un navegador de Internet como Internet Explorer o Netscape Navigator, las demás computadoras cliente acceden a la aplicación que se encuentra en el servidor.

Es importante mencionar que la información puede ser extraída de más de un origen de datos; esto significa que se puede tener un conjunto de datos de diferentes manejadores de datos e inclusive de diferentes sistemas operativos.

La mayoría de aplicaciones que acceden a uno o más orígenes de datos tienen las siguientes características:

a) Establecen una sesión o conexión con el origen de datos

b) Generan una instrucción Transact-SQL

c) Regresan los valores de la consulta

El establecer una sesión o conexión con el origen de datos es uno de los procesos más caros a nivel servidor, porque se deben asignar recursos para crear una sesión y mantenerla activa.

La mayoría de las aplicaciones de dos capas abren la conexión al principio de la aplicación y la cierran al término de ésta, lo cual puede ser bueno en ambientes controlados, pero terrible en ambientes donde se desconoce el número de usuarios, como en Internet.

El resultado de la ejecución de la sentencia SQL generalmente regresa un conjunto de filas que son gestionadas localmente en la computadora cliente, lo que permite un alivio a los recursos del servidor, por eso siempre es recomendable hacer consultas que regresen el menor numero de filas, en caso de necesitarse más en el transcurso del programa, se debe hacer una nueva consulta.

Aunque la mayoría de las aplicaciones abren una conexión al principio de la aplicación, ésta se encuentra sin actividad la mayor parte del tiempo.

3.2 Arquitectura de dos capas

Este tipo de arquitectura es una de las más utilizadas por los desarrolladores (véase figura 3.1), tiene la característica de estar continuamente unida a la base de datos, los usuarios interactúan directamente con los datos para agregar, borrar, editar, etc.

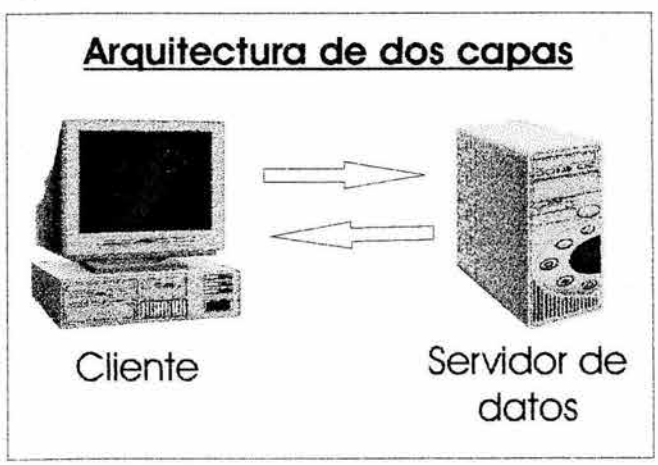
Las reglas de negocios se encuentran en la base de datos por medio de *triggers* o se encuentran directamente en el código de la aplicación, la forma más sencilla de saber si una aplicación es de dos capas es la siguiente: si se escriben directamente las consultas y el acceso a datos en la aplicación, entonces es una aplicación con una arquitectura de este tipo.

Se utiliza el bloqueo de registros para evitar que dos usuarios hagan cambios a un mismo registro al mismo tiempo.

En el caso de aplicaciones ASP.NET, la conexión se abre solamente si se requiere hacer alguna consulta con la base de datos, cerrándose después.

El desarrollador, debe de saber el modelo exacto de la base de datos porque todas las llamadas en la base de datos son sentencias SQL, en el caso de hacer algún cambio en la aplicación, se debe de reestructurar completamente, porque el código de acceso a la base de datos se encuentra escrito directamente en la aplicación, lo que sacrifica la escalabilidad.

Figura 3.1



Fuente: Diagrama creado por el autor de esta tesis.

La arquitectura de dos capas se caracteriza por tener una relación directa con la base de datos. Este modelo funciona en ciertos ambientes controlados, pero no en ambientes grandes y de continuo cambio.

3.3 Arquitectura de tres capas

En esta arquitectura la base de datos se encuentra separada de la interfaz de usuario por medio de una capa intermedia llamada **reglas de negocios** ó **servicios de negocios** (véase figura 3.2), hacer esta separación otorga ventajas inmediatas ya que la capa intermedia puede procesar los datos antes de ser mandados a la interfaz de usuario, además de poder presentar la información en diferentes formatos, también elimina la estrecha relación entre comandos SQL, procedimientos y la interfaz de usuario, pudiéndose hacer infinidad de cambios sin que se tengan que hacer grandes cambios en el código, porque se encuentra encapsulado en la capa intermedia. Esta técnica es superior al uso de *triggers* que se encuentran en la capa de datos.

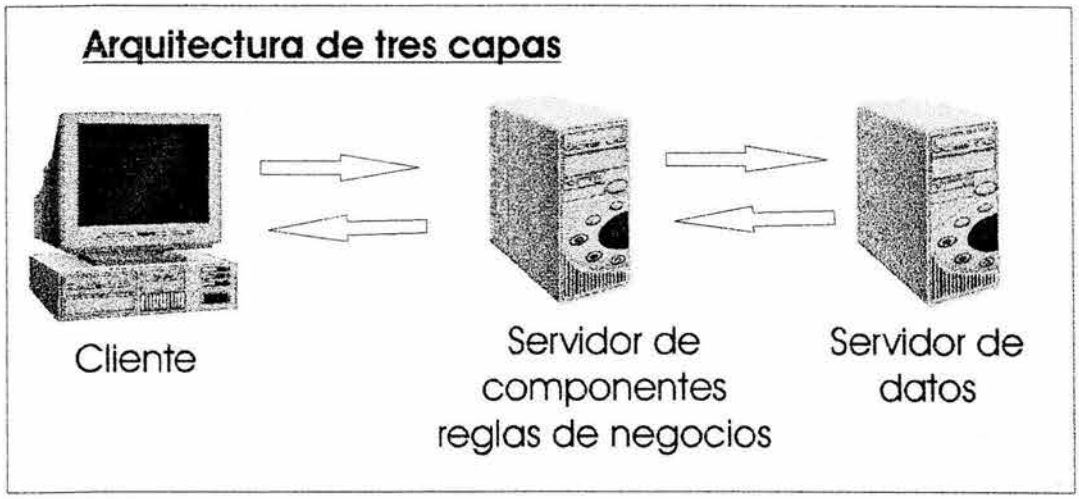
Es posible tener componentes compartidos en esta arquitectura, lo que permite que la aplicación instalada en varias computadoras cliente pueda acceder al mismo componente. Cambiando código en el componente automáticamente se actualizan todas las aplicaciones.

Una arquitectura de tres capas requiere más trabajo para el ingeniero de software, además de requerir mayor tiempo para planearse. Existen ocasiones en que es mejor utilizar una arquitectura de dos niveles, dependiendo del ambiente en que se utilice la aplicación.

En ésta arquitectura la conexión se mantiene abierta, solamente cuando se requiere hacer algún acceso la base de datos, terminado el proceso se cierra la conexión y se liberan recursos del servidor.

Para disminuir el costo tan grande de este proceso, el servidor tiene conexiones previamente creadas y abiertas listas para ser utilizadas, a este mecanismo se le llama *pool*, de tal modo que cuando se solicite el acceso a la base de datos se entregará una de estas conexiones, como si fuera un préstamo y al terminar de utilizarla se regresará al *pool*.

Figura 3.2



Fuente: Diagrama creado por el autor de esta tesis.

3.4 Propuesta ADO.NET

Esta tecnología se basa en la desconexión automática de la base de datos, para no consumir recursos excesivos del servidor, sino solamente cuando se solicite.

El tipo de ejecución unidireccional se refiere a los comandos *Transact-SQL* que no regresan registros como resultado de una consulta, como son: modificación, inserción o borrado. El modelo desconectado no requiere mayor complejidad debido a que una vez ejecutada la sentencia SQL, se desconecta la conexión.

El tipo de ejecución bidireccional es más complejo, ya que además de ejecutar una sentencia SQL debe de regresar un conjunto de registros como resultado de una consulta, los cuales son almacenados en forma de copia local. Estos registros se convierten en una colección de datos que puede ser manipulada sin necesitar estar conectada con el origen de datos. La estructura que contiene a estos registros es llamada ***DataSet***. Cuando sea necesario mantener la compatibilidad con sistemas ya existentes, es posible crear estructuras que interactúen con estos sistemas, lo que permite que el desarrollador pueda crear tablas personalizadas, para que se adapten a sistemas específicos, además que la estructura creada puede tener las mismas características que las de una base de datos.

Los registros que se encuentren en forma de copia local se pueden ordenar, filtrar, borrar, agregar, modificar por lo que es muy flexible su utilización dentro de las aplicaciones, todo esto sin que se modifique el origen de datos, sino hasta que se haga una sincronización, esto lo hace ADO.NET en forma automática.

En caso de que dos o más usuarios modifiquen un registro de su copia local y después quieran hacer la sincronización con el origen de datos, debe de existir algún mecanismo para evitar un conflicto.

Otra situación se refiere a las relaciones entre las tablas, como sabemos en las bases de datos relacionales existen tablas, relaciones y llaves primarias. Es común encontrar tablas que dependan de otras, ahora es posible copiar las relaciones de las tablas del origen de datos.

3.5 Ventajas de ADO.NET

- ADO.NET tiene la ventaja de poder hacer una copia local de las tablas, campos y relaciones entre tablas, de un origen de datos.
- Contiene un *pool de conexiones* que disminuye la carga del servidor creando conexiones antes de ser utilizadas.
- La información puede ser guardada en formato XML (Extensible Markup Language), este lenguaje de marcas se ha convertido en un estándar a nivel mundial ideal para transportar datos, permite su libre circulación entre aplicaciones, servidores y diferentes sistemas operativos.

3.6 Desventaja de ADO.NET

- Visual Basic.NET está totalmente orientado a objetos, existe una extensa variedad de *espacios de nombres* o *namespaces* (véase Capítulo I), que pueden ser utilizados por el programador. En ADO era muy sencillo acceder a los datos de un origen de datos, sin embargo no contaba con las ventajas del nuevo modelo. En aplicaciones que requieran una conexión constante con la base de datos puede ser más recomendable utilizar ADO en vez de ADO.NET.

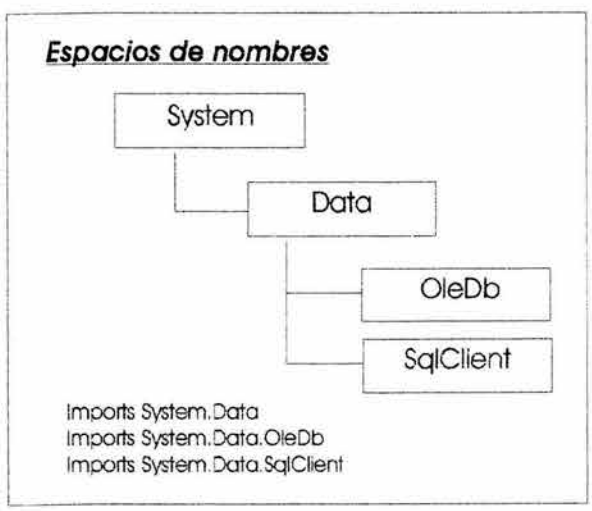
Dependiendo del tipo de aplicación se utilizará un modelo u otro, lo que sí puedo asegurar, es que para arquitecturas en donde el número de usuarios es desconocido o tenderá a aumentar, ADO.NET es mejor opción. En el caso de querer un control total sobre la modificación de la base de datos, incluyendo las relaciones y llaves primarias ADO.NET es el candidato. Si solamente se quieren acceder los datos para mostrar alguna tabla en una computadora o en una pequeña LAN (Local Area Network) bajo la arquitectura de dos capas, ADO funcionará perfectamente.

3.7 Jerarquía de ADO.NET

Como se explicó en el primer capítulo, existen *espacios de nombres* o *namespaces* los cuales se definen como: *paquetes de clases que son agrupados de acuerdo a su función o características comunes.*

Los *espacios de nombres* se encargan de administrar y organizar las diferentes clases con las que cuenta el lenguaje, tal como lo haríamos al administrar los archivos en carpetas y éstas a su vez en subcarpetas, de esta forma se crean árboles jerárquicos. En el caso de ADO.NET se organizan como lo muestra la figura 3.3.

Figura 3.3



Fuente: BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 385.

De acuerdo a la tarea que realicen se clasifican en dos:

a) **Consumidores de datos.** Están encargados de establecer la conexión con el origen de datos dependiendo del proveedor de datos tenemos *OleDb* que es genérico y *SqlClient* para Microsoft SQL Server. Se puede acceder a cualquier proveedor de datos por medio de *OleDb*, sin embargo el rendimiento mejora si se utiliza el *espacio de nombres* especifico para el proveedor, actualmente solamente se tiene *SqlClient* para Microsoft SQL Server, sin embargo es muy probable que un futuro no muy lejano cada proveedor de datos provea su propio espacio de nombres para un mejor rendimiento.

b) **Gestores de información.** Después de haber sido obtenida la información, independientemente del proveedor de datos, sin importar que se haya accedido en forma específica o genérica, la información será tratada en forma común.

3.8 Conexión con un origen de datos

3.8.1 Conexión con un origen de datos genérico

Para establecer una sesión con un origen de datos genérico primeramente debemos de importar los espacios de nombres requeridos:

```
'Importo el espacio de nombres System.Data.OleDb
Imports System.Data.OleDb
```

Se define una variable de tipo *OleDbConnection*:

```
'Variable de tipo conexión de OleDb
Dim Conexion As OleDbConnection
```

Se define una variable de tipo *String* que contendrá la cadena de conexión a la base de datos, por ejemplo:

Para Microsoft Access

```
'Defino la conexión
CadenaConexion = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
"Data Source=MiBaseDeDatos.mdb"
```

Para Microsoft SQL Server 2000

```
'Defino la conexión
CadenaConexion = "Provider=SQLOLEDB.1;" & _
"User Id=sa;password=MiClaveDeSQLServer" & _
"Persist Security Info=False; InitialCatalog=BaseDeDatos" & _
"Data Source=(local)"
```

Para configurar la conexión, creo una instancia de la clase *OleDbConnection* que recibirá como parámetros la cadena de conexión previamente creada.

```
'Creo el objeto Conexion
Conexion = New OleDbConnection(CadenaConexion)

'Abro la conexión
Conexion.Open()
```

Después definimos una variable de tipo *OleDbCommand* para ejecutar una sentencia *Transact-SQL* o un procedimiento almacenado.

```
'Variable que define un comando
Dim ComandoOleDb As OleDbCommand

'Sentencia SQL
Dim sQry As String

'Defino la sentencia SQL, en este ejemplo es la de borrado
sQry = "DELETE FROM MiTabla WHERE Condicion"

'Creo un comando nuevo, poniendo como parámetros la sentencia
'SQL y la conexión
ComandoOleDb = New OleDbCommand(sQry, Conexion)
```

Para acceder a un conjunto de registros, la forma más eficiente es por medio del *DataReader*, es el único objeto que consume una conexión todo el tiempo, sin embargo solamente tiene sentido de acceso de datos hacia delante, por lo que no se puede regresar al registro previo. Se declara de la siguiente forma:

```
'Variable de tipo Lector de datos
Dim LectorOleDb As OleDbDataReader
```

Para poder cargar el lector de datos el objeto *Command* tiene un método que pone los datos de la consulta en un objeto *DataReader*.

```
'Cargo el lector de datos
LectorOleDb = ComandoOleDb.ExecuteReader
```

Para que pueda obtenerse el valor de cada fila del lector de datos, tenemos el método *Read* que permite leer una fila a la vez, sin embargo se debe de poner dentro de un ciclo de iteración, ya que automáticamente pasa al siguiente registro, cuando se encuentra al final regresa el valor booleano *false*.

Además de obtener la fila correspondiente debemos de obtener el número de columna, por lo cual utilizamos la propiedad *Item*, comenzando desde el cero para la primera columna.

A continuación se presenta el código de una iteración con un ciclo, con el método *Read*, depositando los valores en un control *ListBox*, que accede a la segunda columna de una tabla previamente cargada.

```
'Obligo a que se lea el lector de datos, renglón por renglón
Do While LectorOleDb.Read()
  'El índice indica el número de columna a extraer
  ListBox1.Items.Add(LectorOleDb.Item(1))
Loop

'Cierro el lector de datos
LectorOleDb.Close()
'Cierro la conexión
Conexion.Close()
```

Cabe mencionar que el *DataReader* no soporta valores nulos o *Dbnull*, por lo que se utiliza el método ***IsDBNull (índice de columna)*** del *DataReader.*

Ejercicio 3.1

Objetivo:

Demostrar la conexión con un origen de datos de un proveedor genérico, el resultado final es una lista que contiene todos los nombres de los empleados, de la tabla *Employee,* de la base de datos de Microsoft Access ***xtreme.mdb,*** la cual se encuentra en la siguiente ruta:

```
C:\Archivos de programa\Microsoft Visual Studio .NET\Crystal Reports\
Samples\Database\xtreme.mdb
```

La versión de Visual Studio .NET que utilizo es la Enterprise Architecture Edition, en ella se incluye Crystal Reports, que es un software para generar reportes. Dentro de éste se encuentra la base de datos *xtreme.mdb*. En caso que tenga una versión diferente de Visual Studio .NET ó no encuentra la base datos, es posible que usted utilice otra, solamente tiene que hacer pequeñas modificaciones en el código; como cambiar la ruta y el nombre de la tabla. Analice el código y encuentre los puntos clave dónde se pueden hacer los cambios.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto.

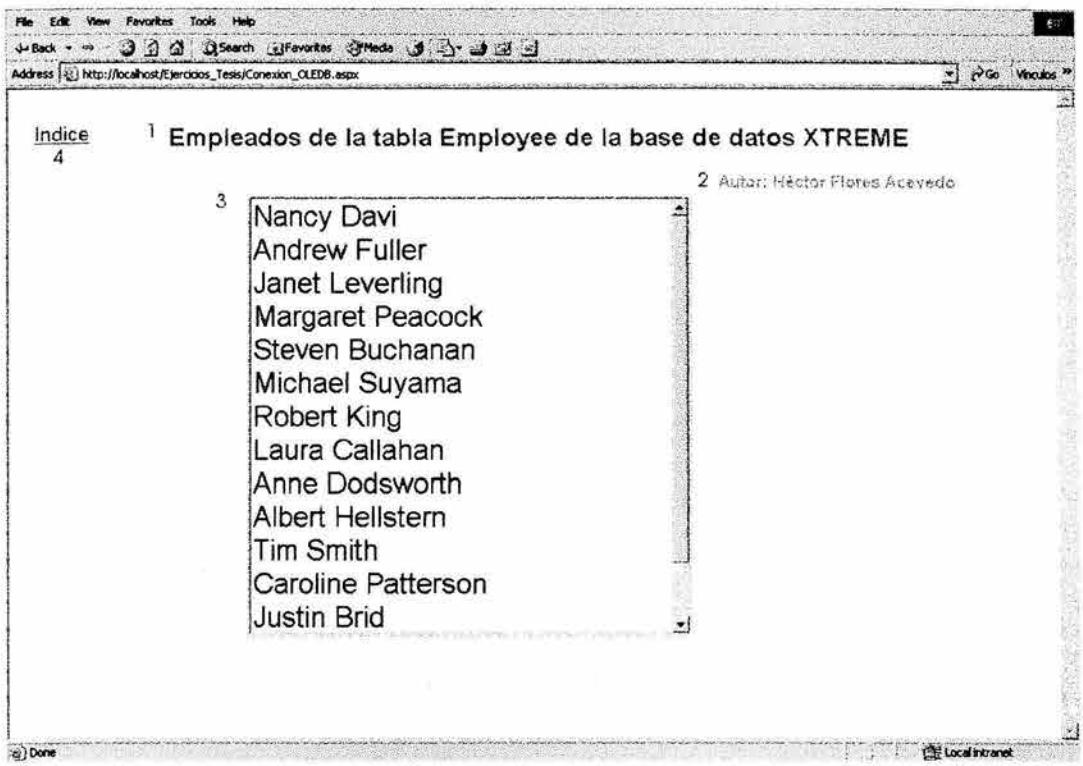
Seleccione *WebForm* y escriba el nombre **Conexion_OLEDB.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 3.4:

1)Label1, 2)Label2, 3)ListBox1, 4)HyperLink1.

Después escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Conexion OLEDB*.

Figura 3.4: Controles de la forma Web Conexion_OLEDB.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

Código de Conexion_OLEDB.aspx

```
'Importo el espacio de nombres para el adaptador genérico
Imports System.Data.OleDb

Public Class Conexion_OLEDB

    Inherits System.Web.UI.Page

    'Controles a utilizar, los cuales tienen asociados eventos.
    Protected WithEvents ListBox1 As System.Web.UI.WebControls.ListBox
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label

#Region " Web Form Designer Generated Code "
```

```
    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    'Variable de tipo conexión de OleDb
    Private Conexion As OleDbConnection

    'Este metodo se genera cuando la página es iniciada y requiere que todos sus
    'controles se configuren
    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro cada una de las propiedades de los controles a utilizar

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        'Con Font.Size puedo establecer el tamaño de la letra del control, siempre
        'y cuando utilice FontUnit.Parse(unidad)
        ListBox1.Font.Size = FontUnit.Parse(20)
        'Escojo el color azul para la lista
        ListBox1.ForeColor = Color.Blue

        'Pongo el nombre a la etiqueta que va sobre la lista
        Label1.Text = "Empleados de la tabla  " & vbCrLf _
        & "Employee de la base de datos XTREME"
        Label1.Font.Size = FontUnit.Parse(18)
        'Indico que el titulo esta en negritas
        Label1.Font.Bold = True
        'Escogo el color verde para la etiqueta
        Label1.ForeColor = Color.Green
        Label1.Font.Name = "Arial"

        Label2.Text = "Autor: Héctor Flores Acevedo"
        Label2.Font.Bold = True
        Label2.Font.Size = FontUnit.Parse(12)
        Label2.ForeColor = Color.CadetBlue
        Label2.Font.Name = "Arial"

    End Sub

#End Region

    'Este código se ejecuta cada vez que se cargue la página Web
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load

        'Variable que contiene la cadena especifica de la conexión
        Dim Cadena_De_Conexion As String

        'Variable que define un comando
        Dim Comando_OleDb As OleDbCommand

        'Sentencia de SQL
        Dim sQry As String

        'Variable de tipo Lector de datos
        Dim Lector_OleDb As OleDbDataReader

        'Defino la conexión
        Cadena_De_Conexion = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data source=C:\Archivos de programa\Microsoft Visual Studio .NET\" & _
        "Crystal Reports\Samples\Database\xtreme.mdb"

        'Creo el objeto de conexión
        Conexion = New OleDbConnection(Cadena_De_Conexion)

        'Abro la conexión
        Conexion.Open()
        'Defino la sentencia de Transact-SQL
```

```
        sQry = "SELECT * FROM Employee"

        'Creo un comando nuevo, poniendo como parametros la sentencia de
        'Transact-SQL y la conexión
        Comando_OleDb = New OleDbCommand(sQry, Conexion)

        'Cargo el lector de datos
        Lector_OleDb = Comando_OleDb.ExecuteReader

        'Obligo a que se lea el lector de datos, renglón por renglón
        Do While Lector_OleDb.Read()
            'El indice indica el numero de columna a extraer, en este caso
            'es el apellido y nombre de cada empleado
            ListBox1.Items.Add(Lector_OleDb.Item(3) & " " & Lector_OleDb.Item(2))
        Loop

        'Cierro el lector de datos
        Lector_OleDb.Close()

    End Sub

    'Procedimiento que se genera cuando se descarga una página
    Private Sub Page_Unload(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles MyBase.Unload
        'Cierro la conexión
        Conexion.Close()
    End Sub
End Class
```

3.8.2 Conexión con Microsoft SQL Server

Para establecer una sesión con un origen de datos específico, en este caso Microsoft SQL Server 2000, primero se debe importar el *espacio de nombre* requerido:

```
'Importo el espacio de nombres SqlClient
Imports System.Data.SqlClient
```

Se crea una instancia de la clase *SqlConnection*, se escribe la cadena de conexión, y se abre.

```
'Declaración de conexión
Public sqlconn As New SqlClient.SqlConnection()

'Defino la cadena de conexión
sqlconn.ConnectionString = "user id=sa;password=MiClave;initial" & _
" catalog=MiBaseDeDatos;data source=(local)"

'abro la conexión
sqlconn.Open()
```

Es posible escribir una cadena de conexión diferente, que utilice la seguridad de Windows en lugar de especificarla en el código, este tipo de conexión es la que utilizaremos en los ejercicios:

```
'Defino la cadena de conexión
sqlconn.ConnectionString = "Integrated Security=True;" & _
                           "initial catalog=Mi base de datos;" & _
                           "data source=(local)"
'abro la conexión
sqlconn.Open()
```

Después definimos una variable de tipo *SqlCommand*, debido a que se abre una conexión para ejecutar una sentencia SQL o un procedimiento almacenado.

```
'Declaración del comando sql
Dim sqlComando As SqlCommand

'Creo el objeto comando
sqlComando = New SqlCommand()

'Le digo al comando a que conexión se refiere
sqlComando.Connection = sqlconn
'Tipo de comando, indico que será una sentencia de Transact-SQL
sqlComando.CommandType = CommandType.Text

'Escribo la sentencia
sqlComando.CommandText = "Sentencia de Transact SQL"
```

Podemos utilizar un lector de datos o *DataReader*, como en el ejemplo anterior, para poder cargarlo. El objeto *sqlCommand* utiliza el método *ExecuteReader* que pone los datos de la consulta en un objeto *DataReader*.

```
'Declaro el lector de datos SQL
Dim LectorDatos As SqlDataReader

'Lleno el lector de datos con la información del comando
LectorDatos = sqlComando.ExecuteReader
```

Para que pueda obtenerse el valor de cada fila del lector de datos, utilizamos el método *Read* que permite leer una fila a la vez, sin embargo se debe de poner dentro de un ciclo de iteración, ya que automáticamente pasa al siguiente registro, cuando se encuentra al final regresa el valor booleano *false*.

Además de obtener la fila correspondiente debemos de obtener su número de columna, por lo cual utilizamos la propiedad *Item*, comenzando desde el cero para la primera columna.

Ejercicio 3.2

Objetivo:

Demostrar la conexión con un origen de datos de un proveedor específico, se utiliza la tabla *Titles* de la base de datos muestra de Microsoft SQL Server 2000 llamada p*ubs*.

Es posible que usted utilice otra base de datos, solamente tiene que hacer pequeñas modificaciones en el código.

Procedimiento:

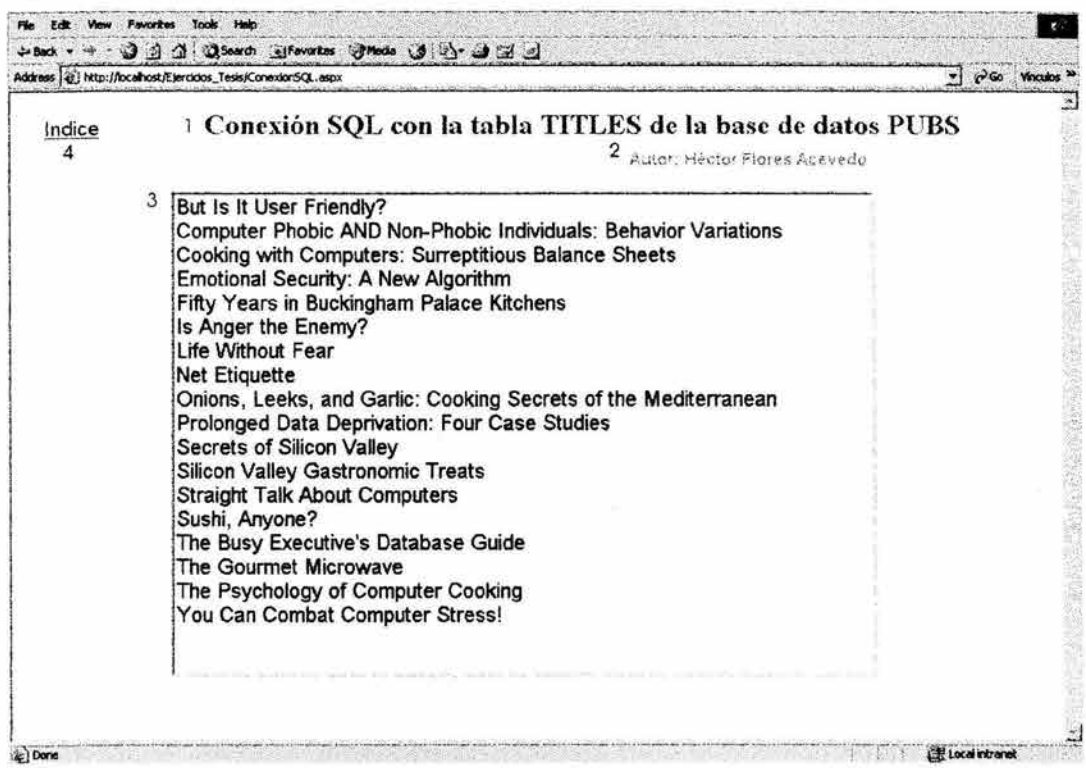
Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **ConexionSQL.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 3.5:

1)Label1, 2)Label2, 3)ListBox1, 4)HyperLink1.

Después escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Conexión SQL*. Es importante que Microsoft SQL Server 2000 esté funcionando.

Figura 3.5: Controles de la forma Web ConexionSQL.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

Código de ConexionSQL.aspx

```
'Importo el espacio de nombres SqlClient
Imports System.Data.SqlClient

Public Class ConexionSQL
    Inherits System.Web.UI.Page

    Protected WithEvents ListBox1 As System.Web.UI.WebControls.ListBox
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Contiguro cada una de las propiedades de los controles a utilizar

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Label1.ForeColor = Color.Green
        Label1.Text = "Conexión SQL con la tabla TITLES de la base de datos PUBS"
        Label1.Font.Size = FontUnit.Parse(20)
        ListBox1.ForeColor = Color.Blue
        ListBox1.Font.Size = FontUnit.Parse(15)

        Label2.Text = "Autor: Héctor Flores Acevedo"
        Label2.Font.Bold = True
        Label2.Font.Size = FontUnit.Parse(12)
        Label2.ForeColor = Color.CadetBlue
        Label2.Font.Name = "Arial"

    End Sub

#End Region

    'Declaración de conexión
    Public sqlconn As New SqlClient.SqlConnection()

    'Declaración del comando sql
    Dim sqlComando As SqlCommand

    'Procedimiento que se genera cada vez que se carga la página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load

        'Creo el objeto comando
        sqlComando = New SqlCommand()

        'Defino la cadena de conexión
        sqlconn.ConnectionString = "Integrated Security=True;" & _
                                   "initial catalog=pubs;" & _
                                   "data source=(local)"
        'abro la conexión
        sqlconn.Open()

        'Le digo al comando a que conexión se refiere
        sqlComando.Connection = sqlconn

        'El tipo de comando es una sentencia Transact-SQL
        sqlComando.CommandType = CommandType.Text
```

```
        sqlComando.CommandText = "SELECT title from titles"

        'Declaro el lector de datos SQL
        Dim LectorDatos As SqlDataReader

        'Lleno el lector de datos con la informacion del comando
        LectorDatos = sqlComando.ExecuteReader

        'Pongo el resultado de la consulta en una lista
        Do While LectorDatos.Read
            ListBox1.Items.Add(LectorDatos.Item(0))
        Loop

        'Cierro el lector de datos
        LectorDatos.Close()

    End Sub

    'Procedimiento que se genera cuando se descarga la página
    Private Sub Page_Unload(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles MyBase.Unload
        'Termino la conexión
        sqlconn.Close()
    End Sub
End Class
```

3.9 Consultas que no regresan datos

En ocasiones necesitamos acceder a la base de datos para realizar alguna modificación, borrado o inserción de datos. Cuando no necesitamos que se regresen filas de información, el objeto *Command* tiene un método llamado *ExecuteNonQuery*, que permite que se ejecute una sentencia de *Transact-SQL* sin que devuelva registros, su sintaxis es la siguiente:

ExecuteNonQuery() As Integer

3.10 Transacciones

La mayoría de las aplicaciones realizan varias acciones en la base de datos para ejecutar un proceso, como por ejemplo una venta, en donde se necesita que todas las operaciones sean realizadas en conjunto o que no se haga ninguna, para esta función se encuentran las **transacciones**. ADO.NET tiene métodos que permiten que los procesos antes mencionados se realicen en forma transparente.

Lo primero que hay que hacer es definir una variable de tipo *Transaction*, en caso de que sea un proveedor genérico se utiliza el prefijo *OleDb* y si es un proveedor específico de Microsoft SQL Server entonces se utiliza *SqlClient*.

```
'Variable encargada de la transacción con un proveedor genérico
Dim Transaccion As OleDbTransaction

'Variable encargada de la transacción con Microsoft SQL Server
Dim Transaccion As SqlTransaction
```

El objeto *Connection* tiene un método llamado *BeginTransaction*, que indica el comienzo del conjunto de sentencias SQL o procedimientos almacenados que componen toda la transacción.

```
'Indico que comenzará una transacción
Transaccion = MiVariableDeConexion.BeginTransaction
```

Posteriormente, a cada comando que se quiera incluir en la transacción, le será asignada la variable *Transaccion* a su método *Transaction*.

Para *OleDbCommand*:

```
'Creo un comando nuevo, poniendo como parámetros la sentencia
'SQL y la conexión
MiComandoOleDb = New OleDbCommand(MiSentenciaSQL, MiConexion)

'relaciono el comando con una transacción
MiComandoOleDb.Transaction = Transaccion

'Indico que se ejecute el comando
MiComandoOleDb.ExecuteNonQuery()
```

Para *SqlCommand*:

```
'Creo un comando nuevo, poniendo como parámetros la sentencia
'SQL y la conexión
MiComandoSql = New SqlCommand(MiSentenciaSQL, MiConexion)

'relaciono el comando con una transacción
MiComandoSql.Transaction = Transaccion

'Indico que se ejecute el comando
MiComandoSql.ExecuteNonQuery()
```

Para finalizar, indicamos que se consuma la transacción por medio del método *Commit* de la variable *Transaccion,* o en caso de ocurrir algún error que se deshagan los cambios por medio del método *RollBack*.

Ejercicio 3.3

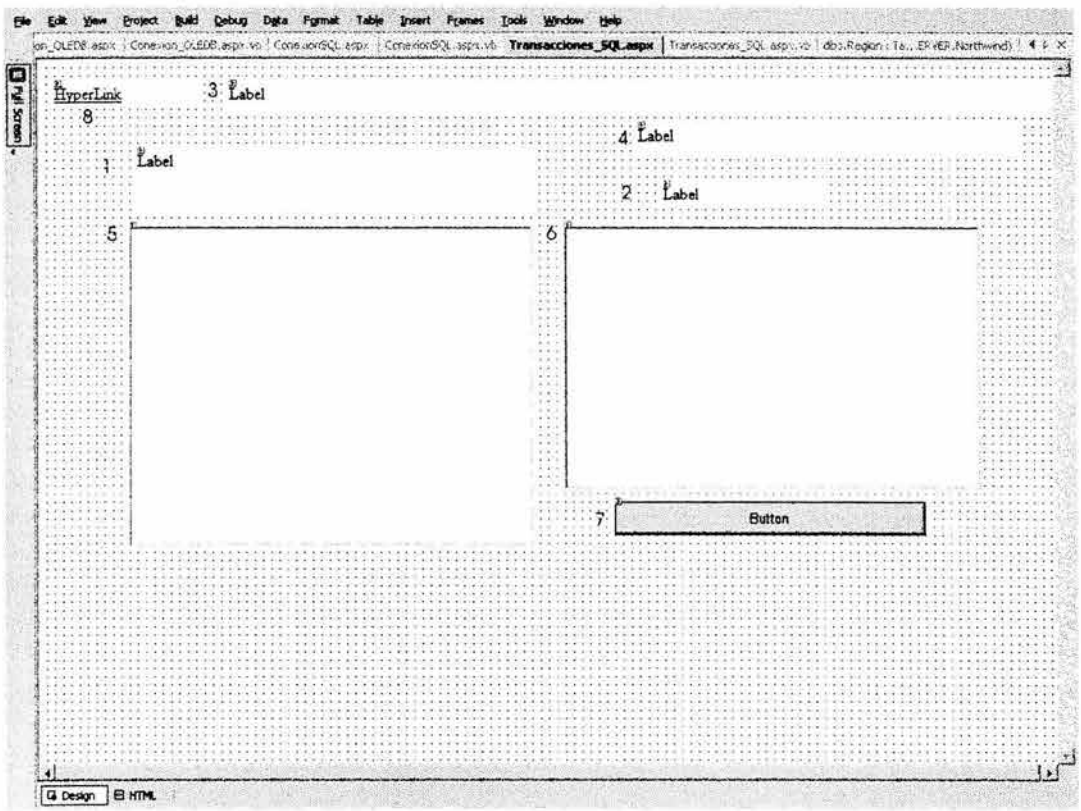
Objetivo:

Agregar dos registros por medio de una transacción a la tabla *Region*, de la base de datos *Northwind*, de Microsft SQL Server. Si se agregaron previamente estos dos registros, entonces marcará un error. Los registros agregados tienen los siguientes valores: *RegionId=100* y *101*, *RegionDescription=Description1* y *Description2*.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **Transacciones_SQL.aspx**, que es el nombre que le he dado a la forma en este ejercicio. Agregue los siguientes controles ordenados de acuerdo a la figura 3.6: 1)Label1, 2)Label2, 3)Label3, 4)Label4, 5)TextBox1, 6)TextBox2, 7)Button1, 8)HyperLink1.

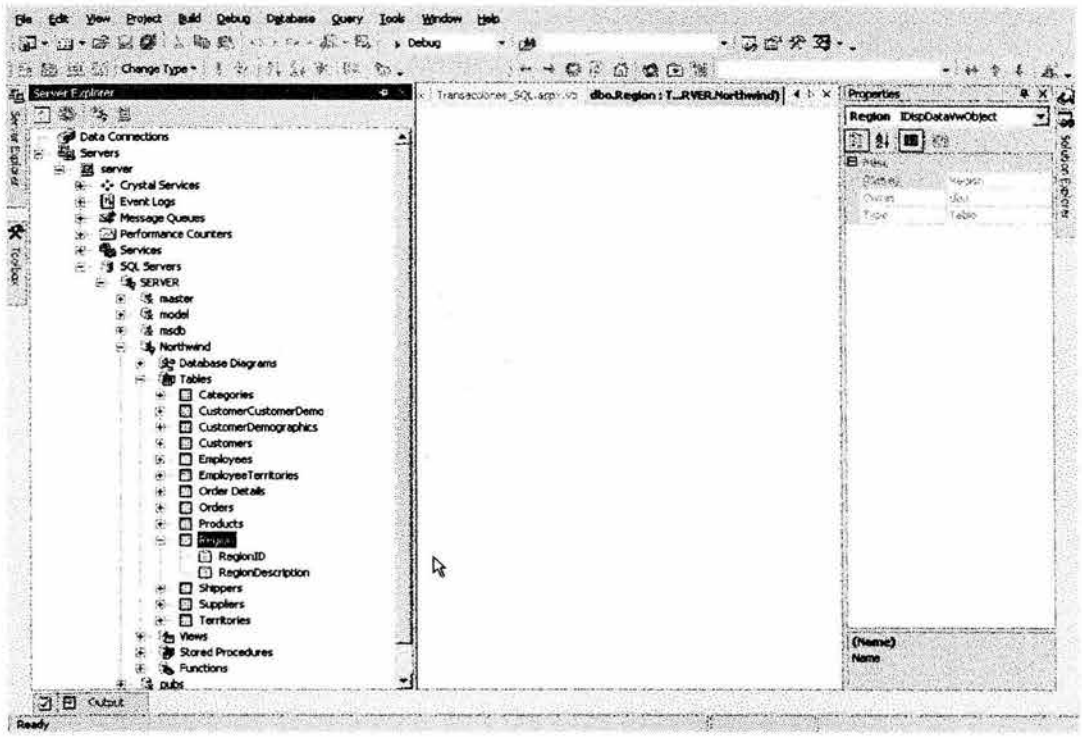
Figura 3.6: Controles de la forma Web Transacciones_SQL.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

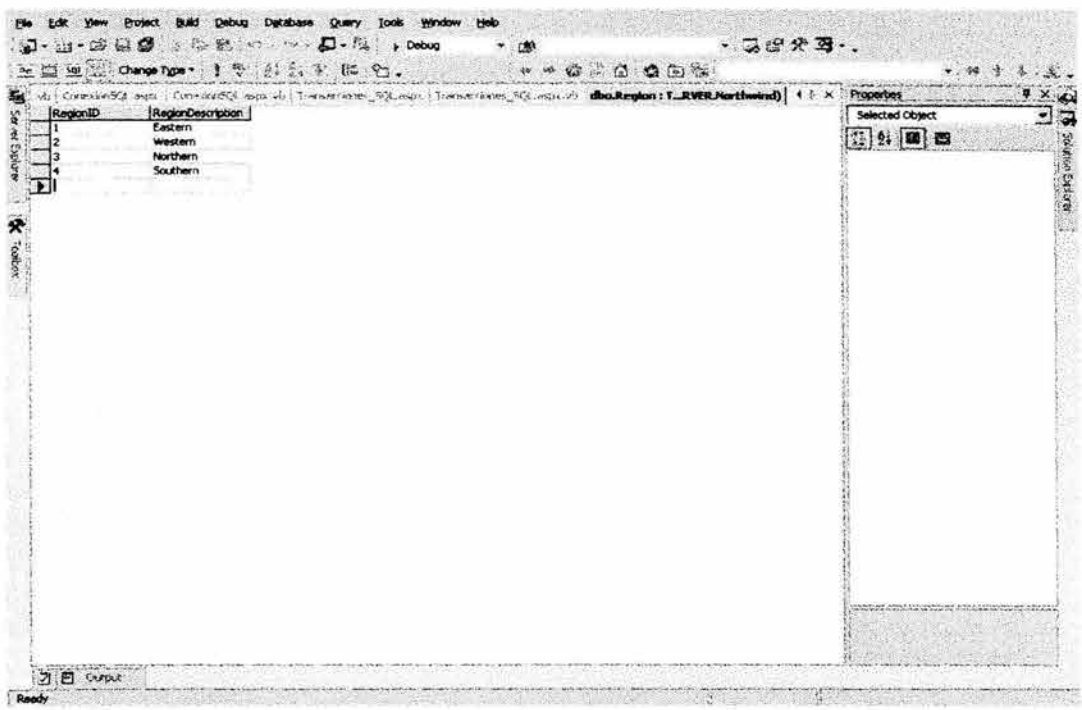
Después escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Transacciones SQL*. Es importante que Microsoft SQL Server 2000 esté funcionando. Cuando se encuentre dentro de la forma Web, no oprima el botón. Antes hay que revisar la base de datos,

Figura 3.7: Explorador de servidores de Visual Studio .NET



Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.8: Tabla Region, de la base de datos Northwind, antes de la transacción.



Fuente: Aplicación creada por el autor de esta tesis.

para ver su estado actual. Para esta operación cierre la ventana de Internet Explorer, regrese a Visual Studio .NET, en el menú oprima *View + Server Explorer*, sitúese en esa ventana y abra los nodos hasta encontrar *SQL Servers*, siga abriendo los nodos hasta que encuentre *Northwind*, después abra *Tables*, hasta encontrar *Region*, como se muestra en la figura 3.7. Por último haga doble clic sobre *Region*, entonces aparecen los registros de esta tabla, como se muestra en la figura 3.8.

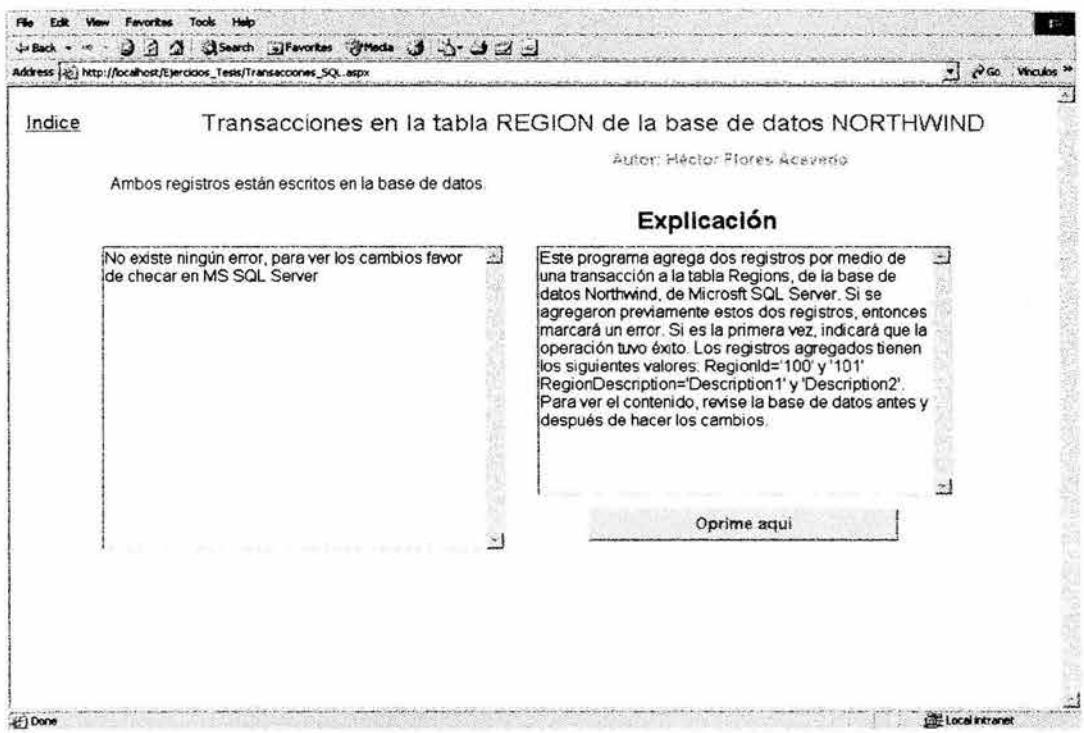
Código de Transacciones_SQL.aspx

```
'Incluyo el espacio de nombres SqlClient
Imports System.Data.SqlClient

Public Class Transacciones_SQL
    Inherits System.Web.UI.Page

    'Controles de la forma que tienen asociados eventos
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBox2 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label

    'Variable de tipo conexión
    Dim myConnection As SqlConnection
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label

    'Variable de tipo cadena de caracteres
    Dim Cadena_Conexion As String

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro las propiedades de cada control

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Button1.Text = "Oprime aqui"
        Button1.Font.Name = "Arial"
        Button1.Font.Size = FontUnit.Parse(12)
        Button1.ForeColor = Color.Crimson
        Button1.BackColor = Color.Wheat
        Button1.Font.Bold = True

        Label1.Text = String.Empty
        Label1.Font.Size = FontUnit.Parse(12)
        Label1.ForeColor = Color.Blue
        Label1.Font.Name = "arial"

        Label3.Text = "Transacciones en la tabla REGIONS de la base de datos Northwind"
        Label3.Font.Size = FontUnit.Parse(18)
```

```
        Label3.ForeColor = Color.Green
        Label3.Font.Name = "arial"

        Label4.Text = "Autor: Héctor Flores Acevedo"
        Label4.Font.Bold = True
        Label4.Font.Size = FontUnit.Parse(12)
        Label4.ForeColor = Color.CadetBlue
        Label4.Font.Name = "Arial"

        TextBox1.Text = String.Empty
        'Con esta propiedad indico que es multilinea
        TextBox1.TextMode = TextBoxMode.MultiLine
        TextBox1.Font.Name = "Arial"
        TextBox1.Font.Size = FontUnit.Parse(12)

        Label2.Text = "Explicación"
        Label2.Font.Size = FontUnit.Parse(18)
        Label2.ForeColor = Color.Blue
        Label2.Font.Name = "arial"
        Label2.Font.Bold = True

        'Con esta propiedad indico que es multilinea
        TextBox2.TextMode = TextBoxMode.MultiLine
        TextBox2.Font.Name = "Arial"
        TextBox2.Font.Size = FontUnit.Parse(12)
        TextBox2.Text = "Este programa agrega dos registros por medio de una " & _
        "transacción a la tabla Regions, de la base de datos Northwind, de " & _
        "Microsft SQL Server. Si se agregaron previamente estos dos registros," & _
        " entonces marcará un error. Si es la primera vez, indicará que la " & _
        "operación tuvo éxito. Los registros agregados tienen los siguientes" & _
        " valores: RegionId='100' y '101' RegionDescription='Description1' y " & _
        "'Description2'. Para ver el contenido, revise la base de datos " & _
        "antes y después de hacer los cambios."
    End Sub

#End Region

    'Procedimiento que se genera cuando se carga una página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load

        'Escribo la cadena de conexión
        Cadena_Conexion = "Integrated Security=True;" & _
        "Data Source=(local);Initial Catalog=Northwind"

        'Creo la conexión
        myConnection = New SqlConnection(Cadena_Conexion)

        'Abro la conexión
        myConnection.Open()

    End Sub

    'Evento que se genera cuando se oprime el boton.
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button1.Click

        'Comienzo la transacción en forma local
        Dim myTrans As SqlTransaction = myConnection.BeginTransaction

        'Creo un comando de tipo Sql
        Dim myCommand As SqlCommand = New SqlCommand()

        'Le asigno la conexión al comando
        myCommand.Connection = myConnection

        'Indico que ese comanado tiene asociada una transacción
        myCommand.Transaction = myTrans

        'Lanzo la excepción
        Try

            'Agrego dos registros, la primera vez que los intente agregar,
            'no generará error. La segunda vez, si generará error porque
            'ya se encuentran dentro de la base de datos
            myCommand.CommandText = "Insert into Region (RegionID, RegionDescription) " & _
            "VALUES (101,'Description1')"
```

```
            'Se ejecuta la instrucción SQL
            myCommand.ExecuteNonQuery()

            myCommand.CommandText = "Insert into Region (RegionID, RegionDescription) " & _
            "VALUES (102,'Description2')"

            'Se ejecuta la instrucción SQL
            myCommand.ExecuteNonQuery()

            'Concluyo la transacción
            myTrans.Commit()

            TextBox1.Text = "No existe ningún error, para ver los cambios favor de " & _
                            "checar en MS SQL Server"
            Label1.Text = "Ambos registros están escritos en la base de datos."

        'En caso de que haya ocurrido algún error se genera la siguiente excepción.
        Catch valor As Exception

            'Se deshace toda operación
            myTrans.Rollback()

            TextBox1.Text = valor.ToString
            Label1.Text = "Ningún registro fué escrito en la base de datos, porque ya " & _
                          "se encuentran grabados. " & _
            "Si quiere agregarlos de nuevo, debe de borrarlos primero. RegionID=100,101"

            'En cualquiera de las dos situaciones, siempre se ejecutará el codigo
            'que se encuentra en Finally
        Finally
            'Cierro la conexión
            myConnection.Close()
        End Try
    End Sub
End Class
```

Figura 3.9: Se agregan dos registros dentro de la forma Web.



Fuente: Aplicación creada por el autor de esta tesis.

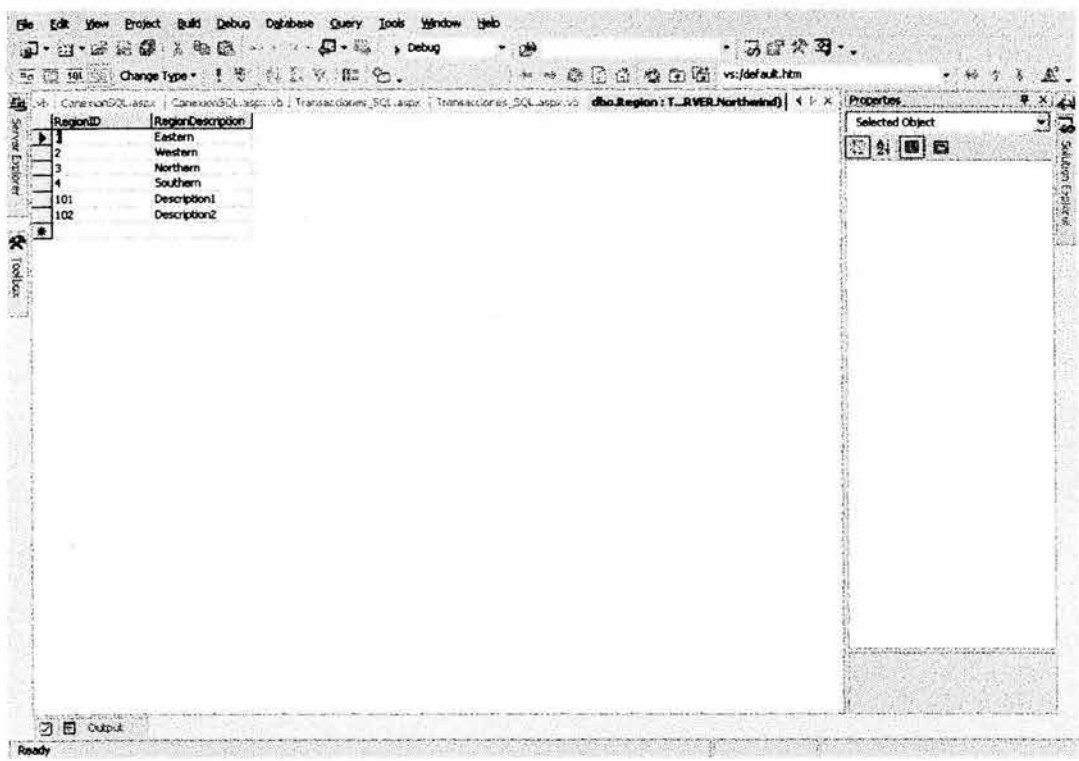
Al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Transacciones SQL,* entonces se agregan los dos registros como se muestra en la figura 3.9. Cierre la ventana de Internet Explorer, regrese a Visual Studio .NET, sitúese sobre la pestaña que dice *dbo.Region* y ciérrela. Regrese al explorador de servidores y efectúe el procedimiento para ver la tabla *Region*, esto lo hace para refrescar los datos, después de hacer esto aparecen la tabla actualizada como se muestra en la figura 3.10.

Ejecute de nuevo la aplicación, sitúese en la forma Web *Transacciones SQL,* oprima el botón. Como se dará cuenta, véase la figura 3.11, el sistema no le permite agregar los registros, porque ya se encuentran dentro de la base de datos.

Para realizar el ejercicio desde un principio, necesita borrar los dos registros agregados, la forma de hacer esto es seleccionando los registros, hacer clic derecho y seleccionar *delete,* entonces aparece un mensaje que pide la confirmación, oprima *yes.*

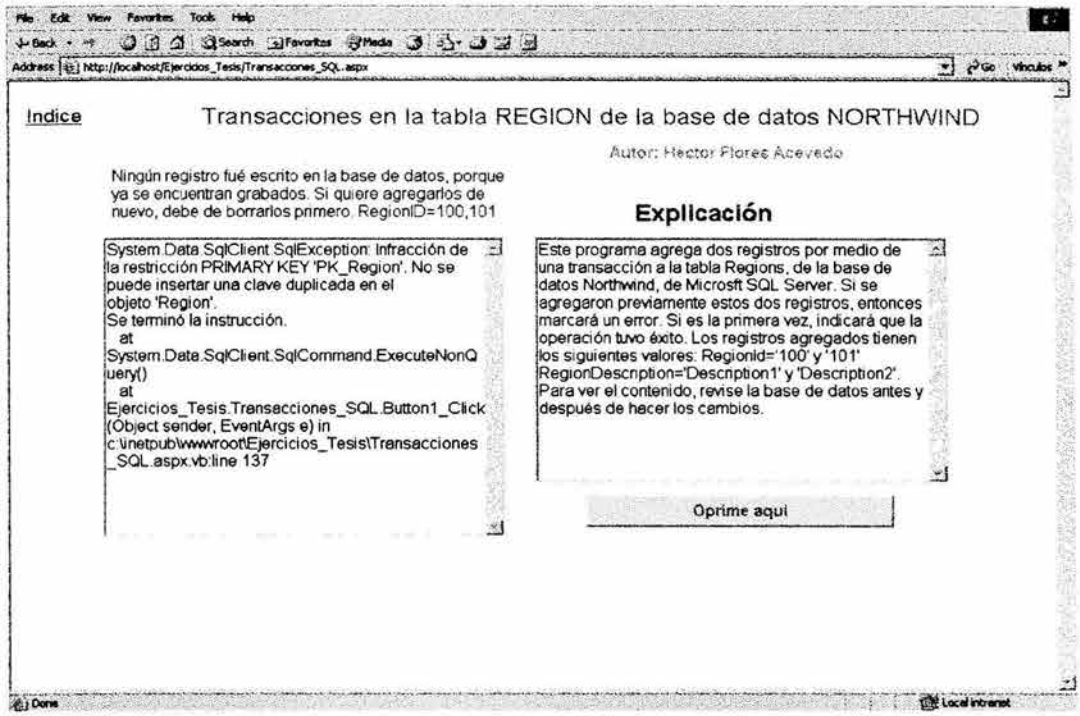
Ahora puede ejecutar la aplicación y repetir el ejercicio, siguiendo las instrucciones antes mencionadas.

Figura 3.10: Tabla Region, de la base de datos Northwind, después de la transacción.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.11: La forma Web no permite que se agreguen los dos registros.



Fuente: Aplicación creada por el autor de esta tesis.

3.11 Procedimientos almacenados

Rob Hawthorne[4] define un procedimiento almacenado de la siguiente forma: *"Un procedimiento almacenado es un conjunto de instrucciones de Transact-SQL que están contenidas dentro de un solo objeto. Al igual que la vista, un procedimiento almacenado no contiene datos, sino sólo referencias a las tablas base que son las que guardan los datos"*.

Una de las grandes ventajas de los procedimientos almacenados, es que residen en un solo lugar, lo que permite que se centralicen todos los procedimientos de una aplicación.

Generalmente los procedimientos almacenados contienen parámetros de entrada, que son precedidos por el símbolo @.

---

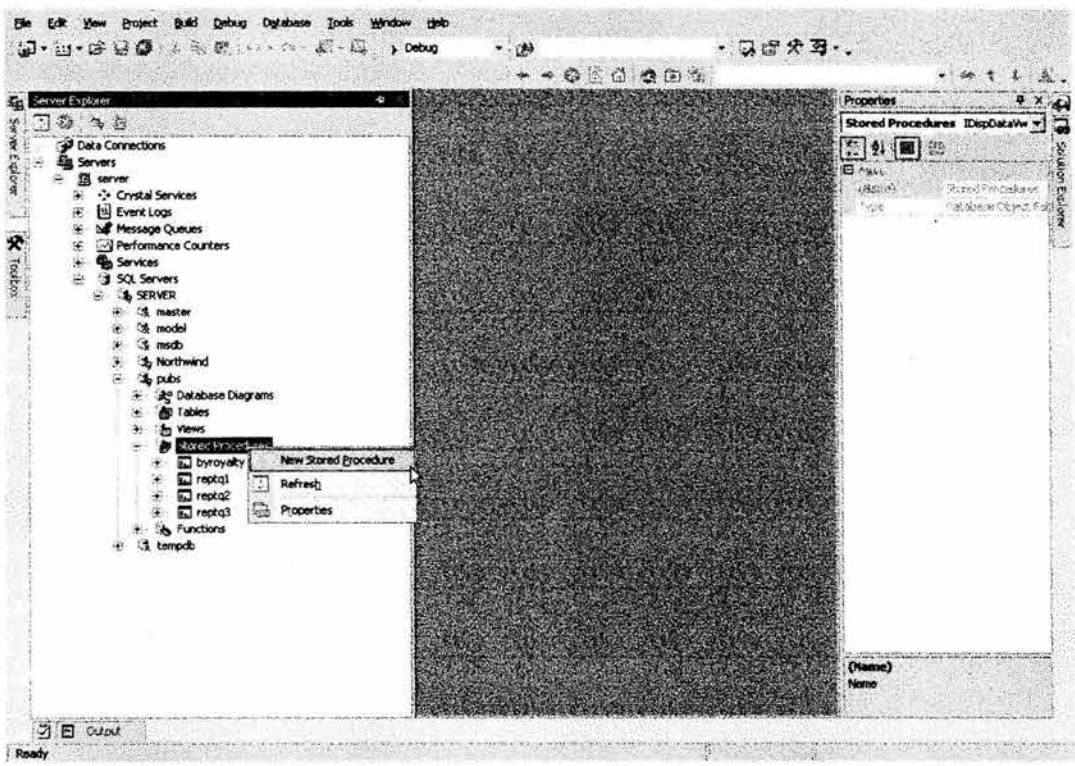
[4] HAWTHORNE, Rob, ***Desarrollo de bases de datos en Microsoft SQL Server 2000***, Pearson Educación de México, México, 2002, p. 138.

El siguiente es un ejemplo, de un procedimiento almacenado creado en la base de datos *pubs* de Microsoft SQL Server:

```
CREATE PROCEDURE dbo.TitulosPorPrecio
@precio MONEY
AS
SELECT title FROM titles WHERE price<@precio
RETURN
```

@precio será la variable introducida desde el exterior, para poder hacer la selección de acuerdo al precio del libro.

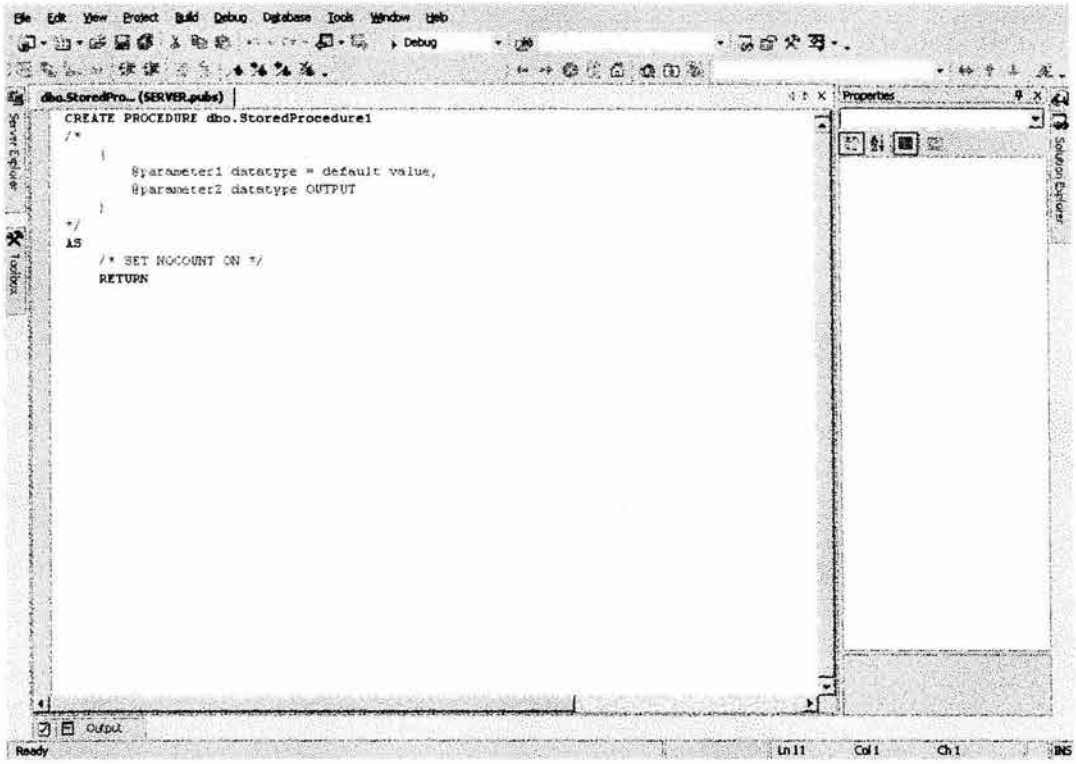
Figura 3.12: Crear un procedimiento almacenado desde Visual Studio .NET.



Fuente: Aplicación creada por el autor de esta tesis.

Puede crear el procedimiento almacenado desde Microsoft SQL Server ó desde el ambiente de desarrollo. Para esta operación sitúese en Visual Studio .NET, en el menú oprima *View + Server Explorer*, sitúese en esa ventana y abra los nodos hasta encontrar *SQL Servers*, siga abriendo los nodos hasta que encuentre *Pubs*, después abra *Store Procedures*, haga doble clic derecho y seleccione *New Store Procedure*, como se muestra en la figura 3.12.

Figura 3.13: Estructura de un procedimiento almacenado.

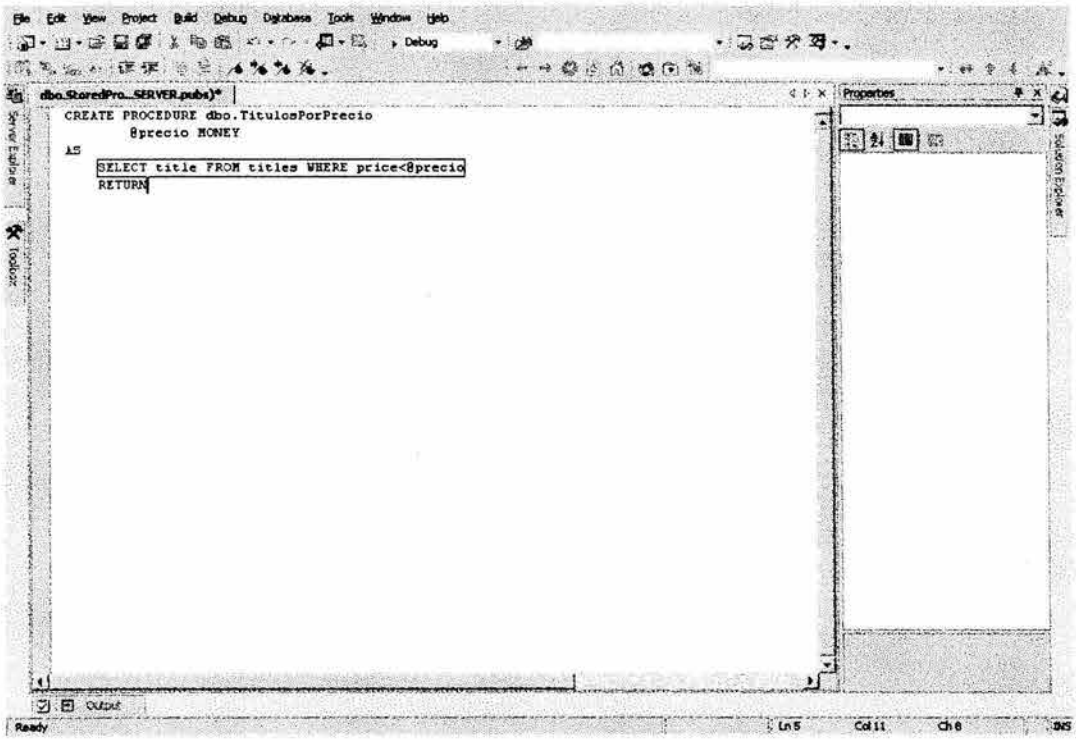


Fuente: Aplicación creada por el autor de esta tesis.

Visual Studio .NET escribe el código general para un procedimiento almacenado. Modifíquelo de acuerdo al código que anteriormente le mostré y grábelo, en caso de no poder realizar esta operación, es probable que tenga al algún problema en la sintaxis, el resultado final se muestra en la figura 3.14.

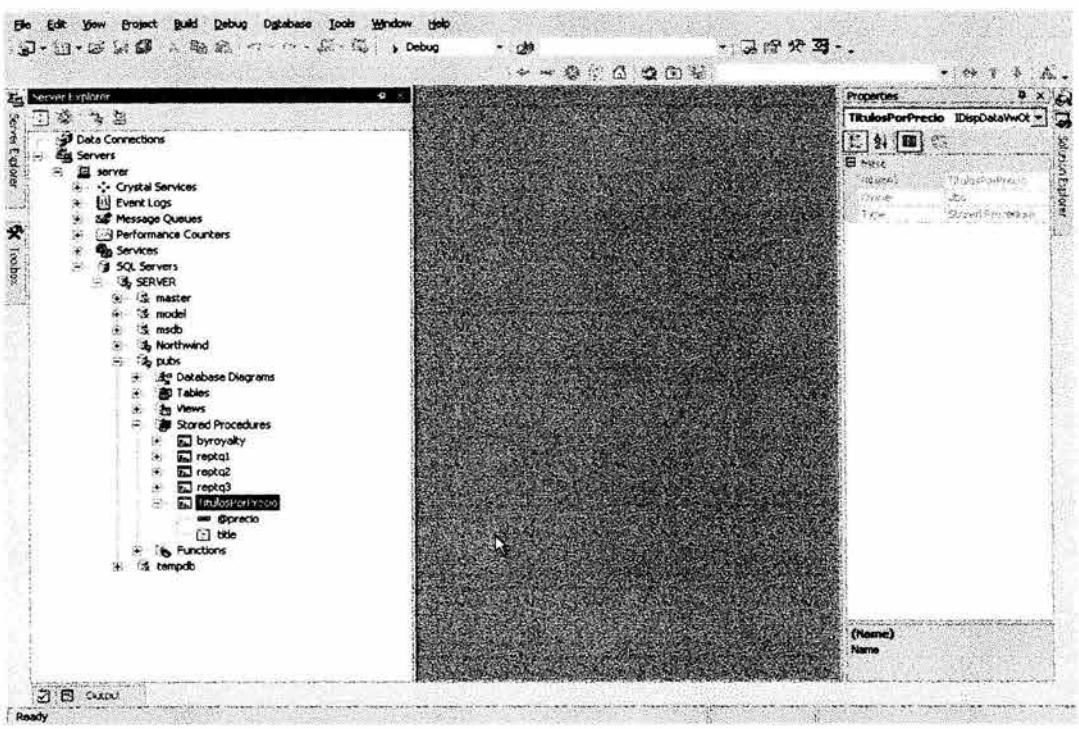
Para revisar que el procedimiento almacenado ha sido creado con éxito, posiciónese sobre *Stored Procedures*, dentro de *Server Explorer*. Entonces verá el procedimiento *TitulosPorPrecio*, si abre el nodo encuentra la variable *precio* y la tabla *titles*, si hace clic en cualquiera de ellos, dentro del panel de propiedades se muestran las características del procedimiento almacenado, de la variable y de la tabla *titles*, como se muestra en la figura 3.15.

Figura 3.14: Procedimiento almacenado *TitulosPorPrecio*.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.15: Propiedades del procedimiento almacenado *TitulosPorPrecio*.



Fuente: Aplicación creada por el autor de esta tesis.

Para que un procedimiento almacenado pueda ser invocado desde Visual Basic.NET, es necesario declarar previamente la variable *command*, en este caso utilizaremos el *espacio de nombres sqlClient* porque es un proveedor especifico.

Se necesitan los siguientes pasos para ejecutar un procedimiento almacenado con el objeto *command*:

1) Crear el objeto **Command.**
2) Poner la propiedad **CommandType** a **StoreProcedure.**
3) Establecer el nombre del procedimiento en **CommandText**
4) Utilizar el método **Add** para poner los parámetros.
5) Establecer la propiedad **ParameterDirection.**
6) En caso de utilizar un lector de datos o **DataReader** ejecutar el método **ExecuteReader.**
7) Utilizar el Lector de datos o **DataReader** para ver los resultados del procedimiento almacenado.

Ejercicio 3.4

Objetivo:

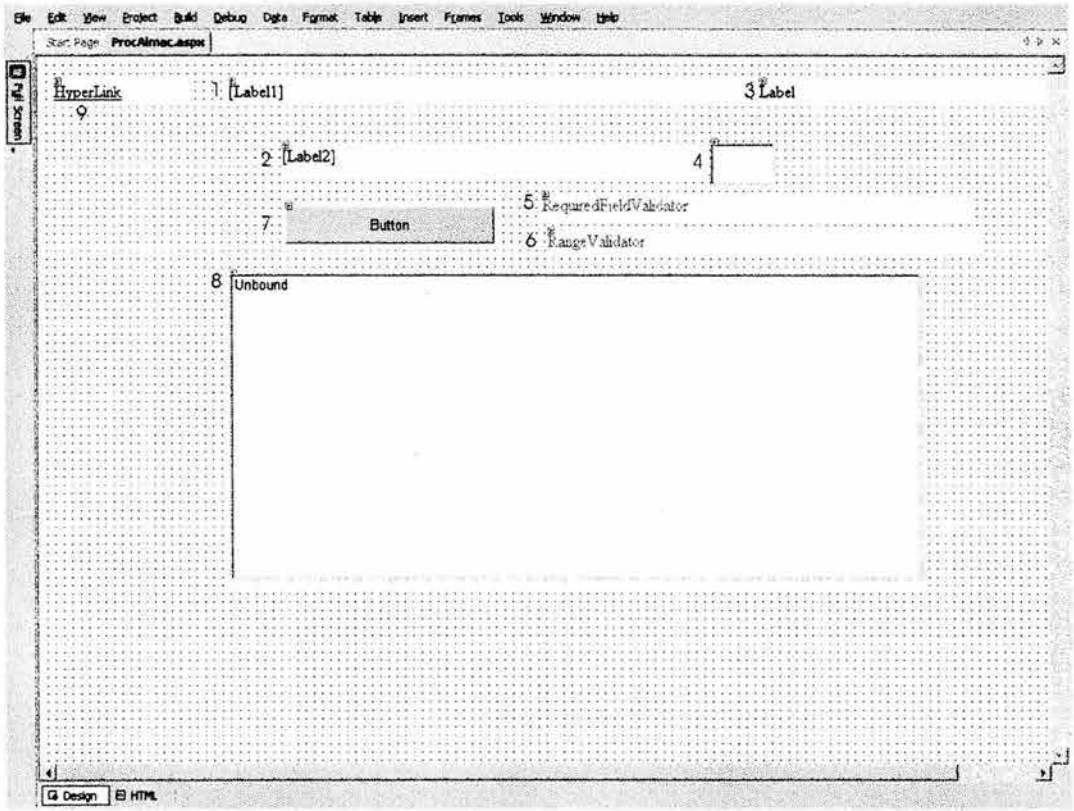
Invocar al procedimiento almacenado creado previamente *TitulosPorPrecio*, se utiliza la tabla *Titles* de la base de datos de Microsoft SQL Server 2000 llamada *pubs*. Es posible que usted utilice otra base de datos, solamente tiene que hacer pequeñas modificaciones en el código.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add* + *Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **ProcAlmac.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 3.16: 1)Label1, 2)Label2, 3)Label3, 4)TextBox1, 5)RequiredFieldValidator1, 6)RangeValidator1, 7)Button1, 8)ListBox1, 9)HyperLink1.

Figura 3.16: Controles de la forma Web ProcAlmac.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

Después escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Procedimiento almacenado*.

Código de ProcAlmac.aspx

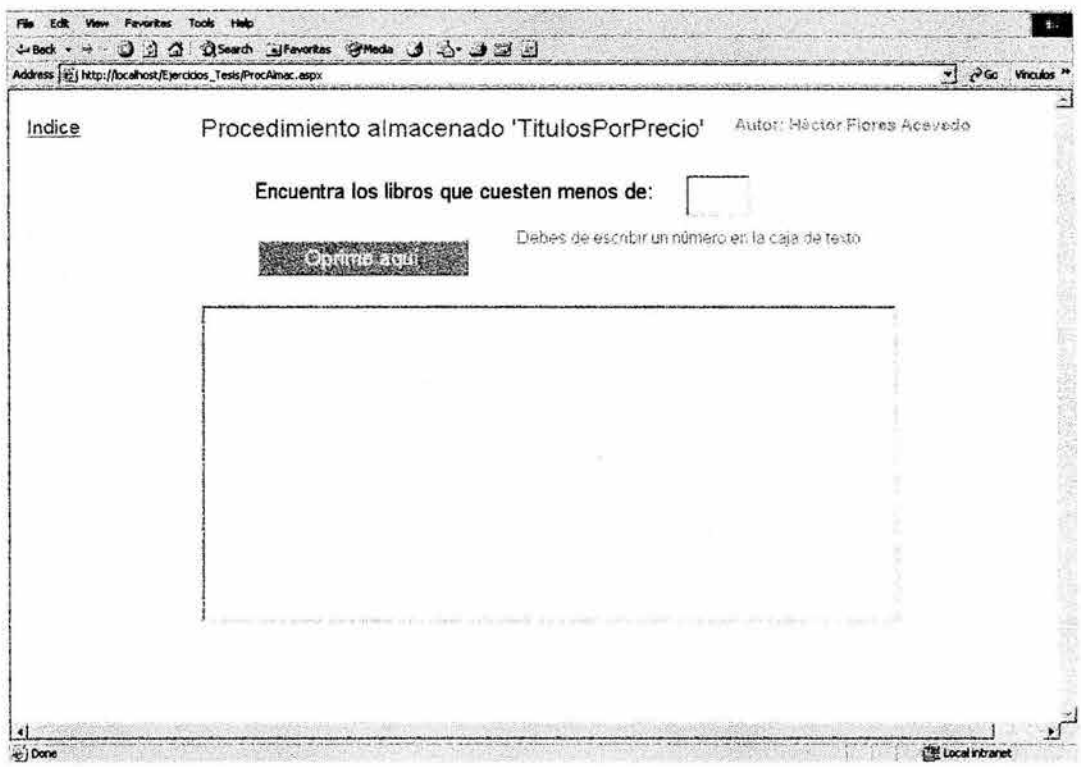
```
'Importo el espacio de nombres SqlClient
Imports System.Data.SqlClient

Public Class ProcAlmac
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro las propiedades de los controles a utilizar
```

```
        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Label1.Font.Size = FontUnit.Parse(18)
        Label1.Font.Name = "Arial"
        Label1.ForeColor = Color.Green
        Label1.Text = "Procedimiento almacenado 'TitulosPorPrecio'"

        Label2.Font.Size = FontUnit.Parse(15)
        Label2.Font.Name = "Arial"
        Label2.ForeColor = Color.Blue
        Label2.Text = "Encuentra los libros que cuesten menos de:"

        Label3.Text = "Autor: Héctor Flores Acevedo"
        Label3.Font.Bold = True
        Label3.Font.Size = FontUnit.Parse(12)
        Label3.ForeColor = Color.CadetBlue
        Label3.Font.Name = "Arial"

        Button1.Text = "Oprime aquí"
        Button1.Font.Name = "Arial"
        Button1.Font.Size = FontUnit.Parse(15)
        Button1.ForeColor = Color.White
        Button1.BackColor = Color.CadetBlue

        TextBox1.Font.Size = FontUnit.Parse(18)
        TextBox1.TabIndex = 0
        TextBox1.Font.Name = "Arial"
        TextBox1.Font.Bold = True
        TextBox1.ForeColor = Color.Blue
        TextBox1.BackColor = Color.White

        ListBox1.Font.Size = FontUnit.Parse(15)
        ListBox1.Font.Name = "Arial"
        ListBox1.ForeColor = Color.Blue
        ListBox1.BackColor = Color.White

        'Inidico el control a validar
        RequiredFieldValidator1.ControlToValidate = "TextBox1"
        'Defino el tipo de letra
        RequiredFieldValidator1.Font.Name = "Arial"
        'Defino el tamaño de la letra
        RequiredFieldValidator1.Font.Size = FontUnit.Parse(12)
        RequiredFieldValidator1.ErrorMessage = "Debes de escribir un número en " & _
                                               "la caja de texto"
        RequiredFieldValidator1.Display = ValidatorDisplay.Static

        'Esta propiedad permite que se pueda validar la caja de texto en forma
        'programática, antes que se despliegue el mensaje de error
        RequiredFieldValidator1.EnableClientScript = False

        RangeValidator1.ControlToValidate = "TextBox1"
        'Defino el tipo de letra
        RangeValidator1.Font.Name = "Arial"
        'Defino el tamaño de la letra
        RangeValidator1.Font.Size = FontUnit.Parse(12)
        RangeValidator1.ErrorMessage = "El valor debe de ser mayor que cero"
        'Valor minimo y maximo, rango en que debe de estar el valor de la caja de texto.
        RangeValidator1.MinimumValue = 0
        RangeValidator1.MaximumValue = 10000
        'Tipo de valor a validar
        RangeValidator1.Type = ValidationDataType.Double

        'Esta propiedad permite que se pueda validar la caja de texto en forma
        'programática, antes que se despliegue el mensaje de error
        RangeValidator1.EnableClientScript = False
    End Sub


#End Region

    'Controles que tienen eventos asociados por eso se indica
    'WithEvents
    Protected WithEvents ListBox1 As System.Web.UI.WebControls.ListBox
```

```
Protected WithEvents Label1 As System.Web.UI.WebControls.Label
Protected WithEvents Label2 As System.Web.UI.WebControls.Label

'Declaración de conexión
Public sqlconn As New SqlClient.SqlConnection()

'Controles que tienen asociados eventos
Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox
Protected WithEvents Button1 As System.Web.UI.WebControls.Button
Protected WithEvents RequiredFieldValidator1 As _
                System.Web.UI.WebControls.RequiredFieldValidator
Protected WithEvents RangeValidator1 As System.Web.UI.WebControls.RangeValidator
Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
Protected WithEvents Label3 As System.Web.UI.WebControls.Label

'Declaracion del comando sql
Dim sqlComando As SqlCommand

'Cada que se carga la página se genera este procedimiento
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load

    'Defino la cadena de conexión
    sqlconn.ConnectionString = "Integrated Security=True;initial catalog=pubs;" & _
    "data source=(local)"

    'abro la conexión
    sqlconn.Open()

End Sub

'Procedimiento que se ejecuta cuando se hace clic en el botón
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click

    'La propiedad Validate, permite que ese instante se validen las cajas
    RequiredFieldValidator1.Validate()
    RangeValidator1.Validate()

    'Si ambas validaciones son verdaderas entonces ejecuta el código
    If RequiredFieldValidator1.IsValid And RangeValidator1.IsValid Then

        'Valido con una excepción que exita el procedimiento almacenado
        Try
            'Creo el objeto comando
            sqlComando = New SqlCommand()

            'Le digo al comando a que conexión se refiere
            sqlComando.Connection = sqlconn

            'Tipo de comando es procedimiento almacenado
            sqlComando.CommandType = CommandType.StoredProcedure

            'Nombre del procedimiento almacenado, se debe de crear previamente
            sqlComando.CommandText = "TitulosPorPrecio"

            'Defino y creo un nuevo parámetro Sql con su respectivo
            'valor
            Dim ParamSQL As New SqlParameter("@precio", SqlDbType.Money)

            'Indico que la dirección del parámetro es de entrada
            ParamSQL.Direction = ParameterDirection.Input

            'Defino el valor del parámetro
            ParamSQL.Value = Trim(TextBox1.Text)

            'Agrego el parametro a la colección de parámetros del
            'comando.
            sqlComando.Parameters.Add(ParamSQL)

            'Declaro el lector de datos SQL
            Dim LectorDatos As SqlDataReader

            'Lleno el lector de datos con la información del comando
            LectorDatos = sqlComando.ExecuteReader

            'Cada que cargo la lista la vuelvo a limpiar
```

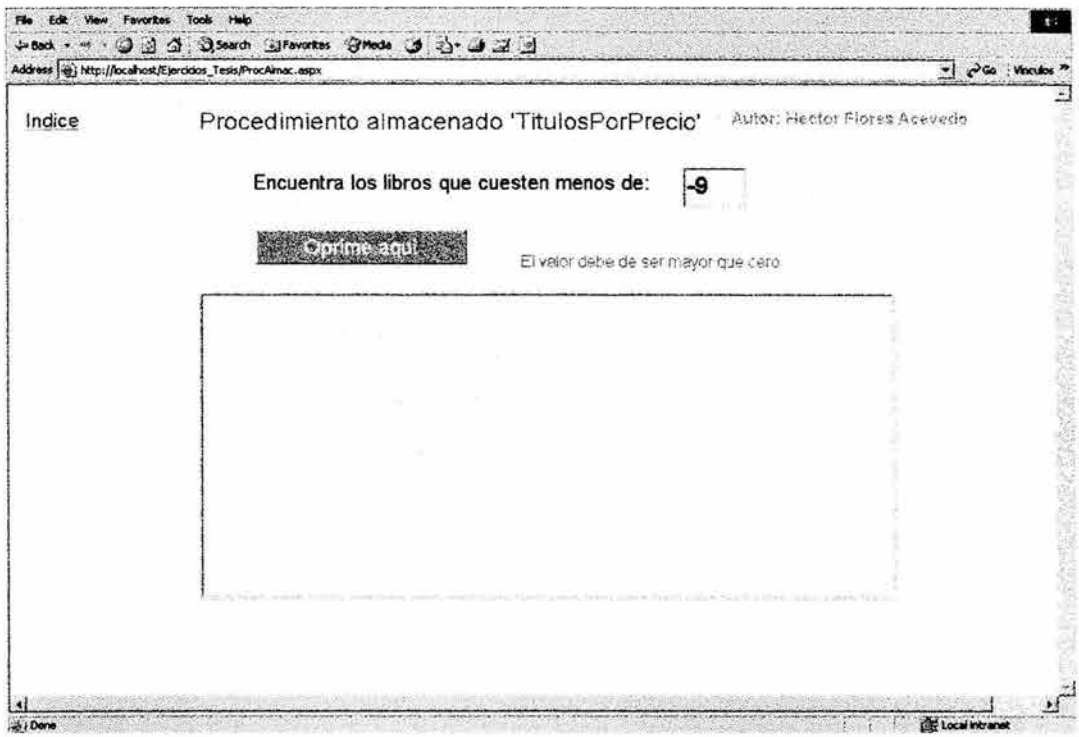
```
            ListBox1.Items.Clear()

            'Pongo el resultado de la consulta en una lista
            Do While LectorDatos.Read
                ListBox1.Items.Add(LectorDatos.Item(0))
            Loop

            'Cierro el lector de datos
            LectorDatos.Close()

            'En caso de que no sea valido el contenido de la caja de texto entonces
            'ejecuta el siguiente código.

        Catch valor As Exception
            'En caso de que no exista el procedimiento almacenado, mando a una
            'página de error
            Response.Redirect("http://localhost/Ejercicios_Tesis/Error.aspx")
        End Try
    Else
        'Limpia la lista
        ListBox1.Items.Clear()
    End If

    'Termino la conexion
    sqlconn.Close()

End Sub
End Class
```

Figura 3.17: La forma Web marca el primer error.



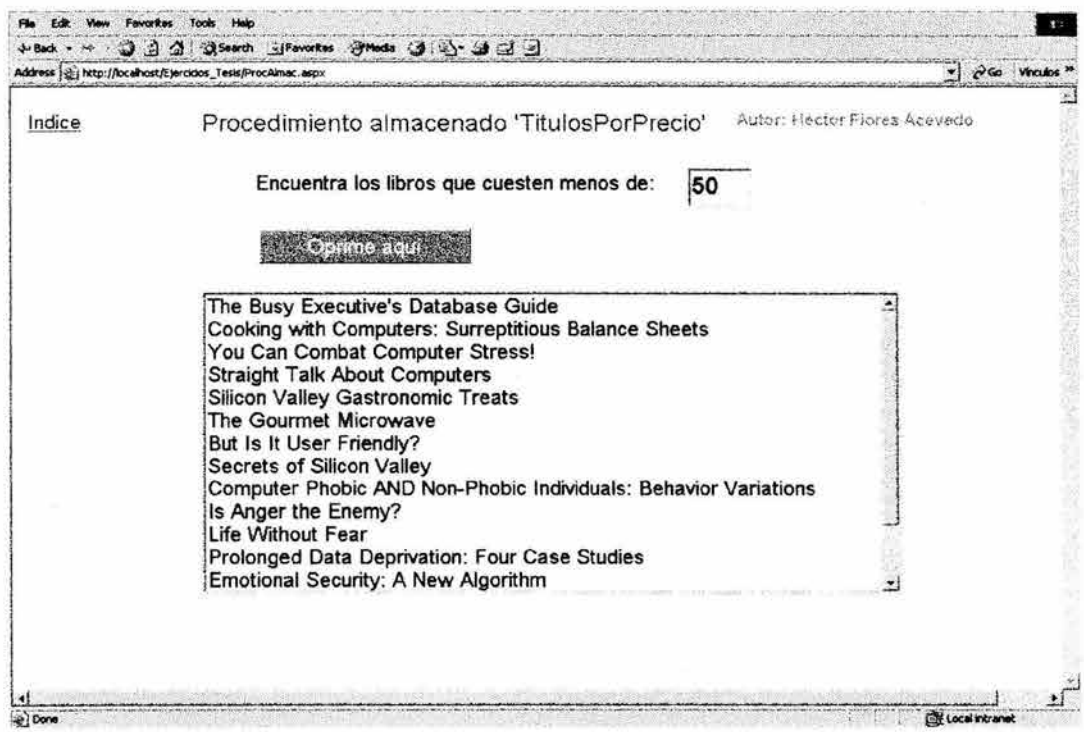
Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.18: La forma Web marca el segundo error.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.19: La forma Web invoca al procedimiento almacenado con éxito.



Fuente: Aplicación creada por el autor de esta tesis.

Al ejecutar la aplicación y situarse en la forma Web *ProcAlmac.aspx*, pueden existir tres casos. El primero es cuando oprime el botón sin haber escrito un número en la caja de texto, entonces marca el primer error, como se muestra en la figura 3.17. El segundo es cuando escribe un número menor que cero y después oprime el botón, entonces marca el segundo error, porque el número debe ser mayor que cero, como se muestra en la figura 3.18. El tercer caso es cuando escribe un número mayor que cero, entonces se invoca el procedimiento almacenado, regresando todos los libros que tengan un precio menor al número que se encuentra en la caja de texto, como se muestra en la figura 3.19.

3.12 Datos desconectados en ADO.NET

Cuando se piensa en la tecnología .NET, se piensa en un conjunto de tecnologías que permiten al programador, desarrollar aplicaciones que vayan más allá de una PC o una pequeña red. Es bien sabido que entre más grandes sean las aplicaciones mayores problemas se encuentran.

En el caso de ADO.NET, Microsoft se preocupó principalmente en el problema que se genera cuando muchos usuarios se encuentran conectados a un origen de datos, no siempre utilizan la conexión, sin embargo sí consumen recursos del servidor, lo que ocasiona que se sature el sistema.

A partir de Visual Basic 6.0 se manejan los datos desconectados por medio de los *recordsets* y su propiedad *ActiveConnection*. En Visual Basic.NET se han hecho mejoras, la copia local de los registros se encuentra automáticamente desconectada del origen de datos.

Cuando se requiere acceder a los datos se establece la conexión, cuando se ha obtenido lo que se buscaba, se desconecta del origen de datos, se manipula la información, en caso de necesitar actualizar la información, entonces se hace una sincronización entre la aplicación y el origen de datos. Esto tiene la ventaja de aliviar al servidor de mantener una sesión abierta con la aplicación cliente.

Utiliza XML como formato para transmitir datos, que permite que cualquier sistema operativo pueda interpretar estos datos, así como mandarlos a cualquier otro origen de datos.

3.13 DataSet

El *DataSet* es uno de los objetos más importantes de ADO.NET, Erich R.Buhler[5] lo define así: *"Esencialmente, un objeto **DataSet** obtiene un conjunto de filas de una o más tablas, y posteriormente las aloja en memoria. Una vez realizado este proceso, el mismo procede a desconectarse del origen de datos.*

*El **DataSet** se encarga posteriormente de la gestión local de las filas obtenidas, así como también de las posibles acciones a ser realizadas sobre las mismas (modificaciones, eliminaciones, etc.).Todas las acciones serán entonces aplicadas sobre la copia privada o local de datos, sin que ello interfiera en el origen. Posteriormente, se necesitará sincronizar la información con este último, a los efectos que los cambios puedan hacerse efectivos".*

Como podemos ver, el *DataSet* ofrece beneficios muy importantes, además de poder almacenar más de una consulta y de establecer relaciones entre las tablas. Esto quiere decir, que en un *DataSet* se pueden guardar varias tablas incluyendo sus relaciones, lo que otorga ventajas inmediatas ya que podemos establecer las mismas restricciones que existen en la base de datos.

La idea de los *DataSets* es utilizarlos conjuntamente con los Servicios Web XML, los cuales veremos en el siguiente capítulo, por ahora podemos decir que los Servicios Web XML son procedimientos que pueden ser llamados en forma remota a través de Internet.

En teoría, una aplicación cliente pediría datos a un Servicio Web XML, que llenaría el objeto *DataSet* regresándolos al cliente, posteriormente desconectándose del origen de datos.

Después de hacer la manipulación de los datos obtenidos, regresan al Servicio Web XML conectándose para hacer los cambios deseados en el origen de datos.

---

[5] BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 401.

3.14 DataTable

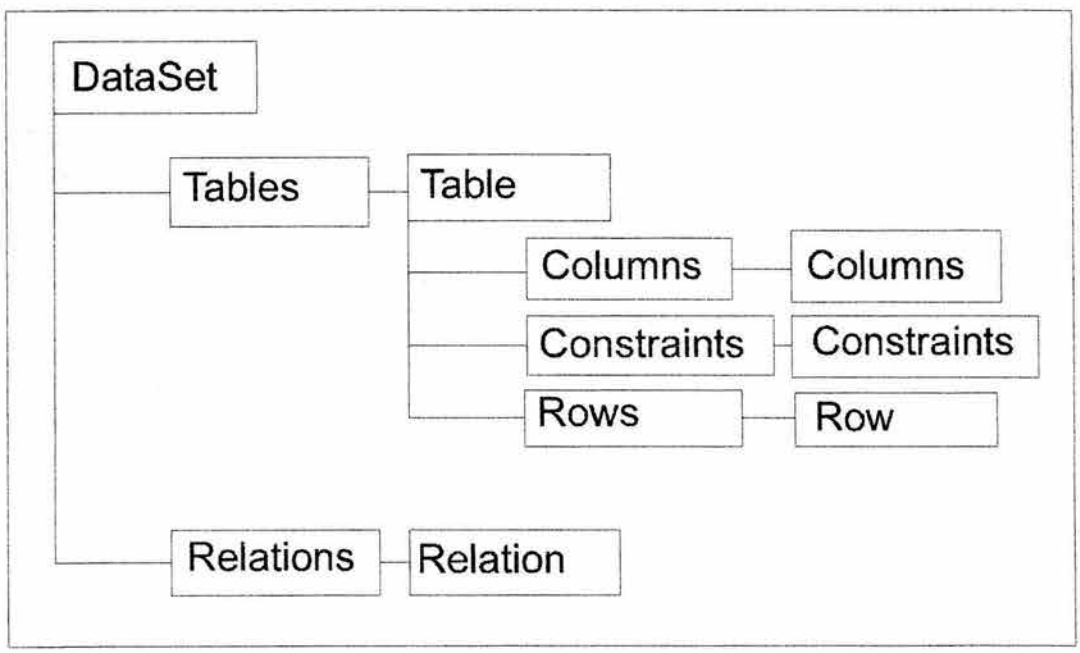
El *DataSet* está compuesto por tablas llamadas *DataTable*, pensemos en un *DataSet* como en una copia local de la base de datos con la que nos estamos conectando, como sabemos el modelo entidad relación está compuesta de entidades y relaciones, que ahora forman parte del *DataSet*.

El *DataTable* está compuesto por una colección de columnas y renglones, que sirven para manipular campos.

3.15 DataRelations

Contiene las relaciones existentes entre las entidades introducidas en el *DataSet*, que permiten establecer restricciones entre ellas (véase figura 3.20).

Figura 3.20: El objeto *DataSet*



Fuente: MICROSOFT CORPORATION, ***Programming with Microsoft Visual Basic .NET***, Microsoft Corporation, Colombia, 2002, Módulo 8, p. 23.

Ejercicio 3.5

Objetivo:

Describir la forma de llenar un *DataSet*, aprender el funcionamiento del *DataAdapter* que sirve para sincronizar el *DataSet* con el origen de datos, introducir el uso del *DataGrid*, que permite que se cargue automáticamente con el *DataSet*. En este caso utilizamos el *espacio de nombres SqlClient*, se utiliza la tabla *Titles* de la base de datos muestra de SQL Server 2000 llamada *pubs*.

Es posible que usted utilice otra base de datos, solamente tiene que hacer pequeñas modificaciones en el código.

```
'SqlDataAdapter("Consulta en Transct-SQL",Conexión al origen)
MiAdaptadorSQL =New SqlDataAdapter(MiSentenciaSQL,MiConexion)
```

El control *DataGrid* es uno de los controles mas utilizados para mostrar datos, como se puede ver en la figura permite la configuración de sus propiedades. Una de sus características más poderosas, es que se pueden establecer botones de inserción, borrado y edición.

Para que pueda ser cargado y enlazado el *DataGrid,* se utiliza el siguiente código:

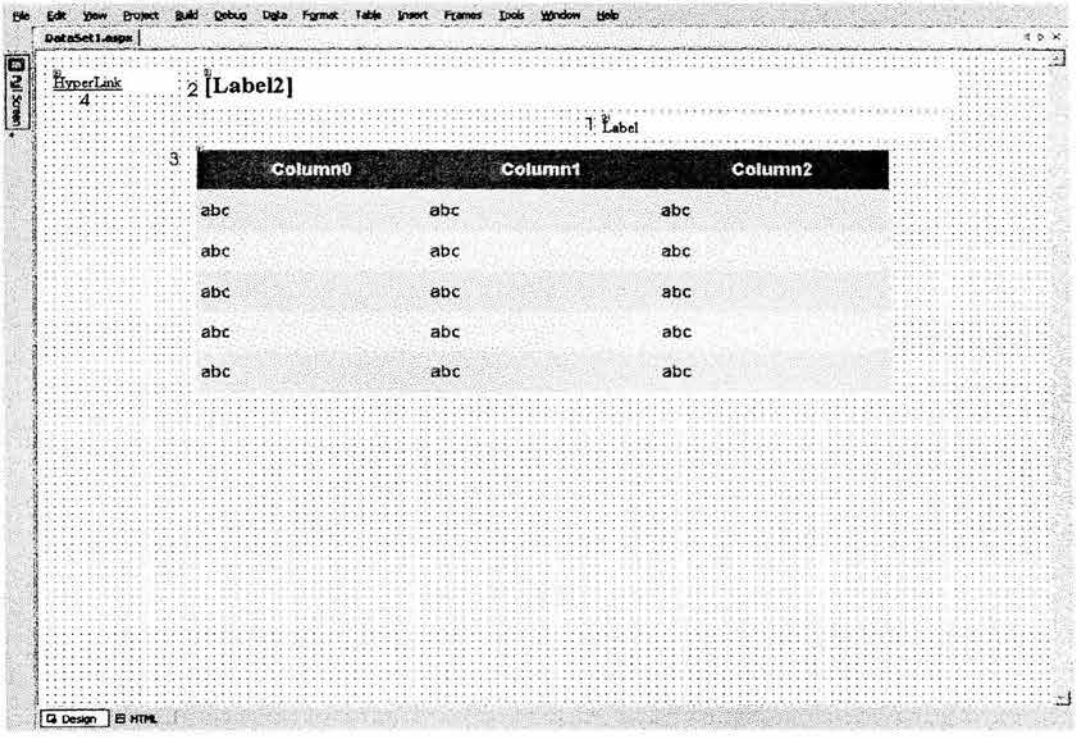
```
'Enlazo el DataSet al DataGrid
DataGrid1.DataSource = DataSetSQL.Tables("MiNombreDeTabla")

'Permite enlazar el control DataGrid con el DataSet
DataGrid1.DataBind()
```

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add* + *Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **DataSet1.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 3.21: 1)Label1, 2)Label2, 3)DataGrid1, 4) HyperLink1.

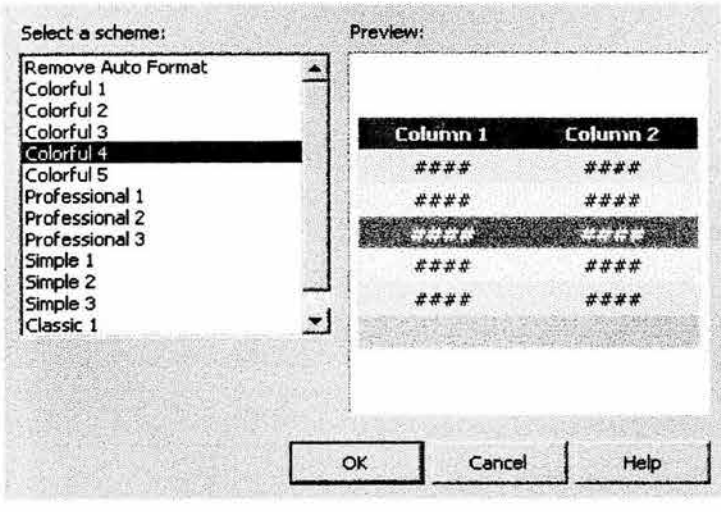
Figura 3.21: Controles de la forma Web DataSet1.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

Después de agregar el control *DataGrid* haga clic derecho sobre éste, seleccione *Auto Format* y por último *colorful 4*, como se muestra en la figura 3.22.

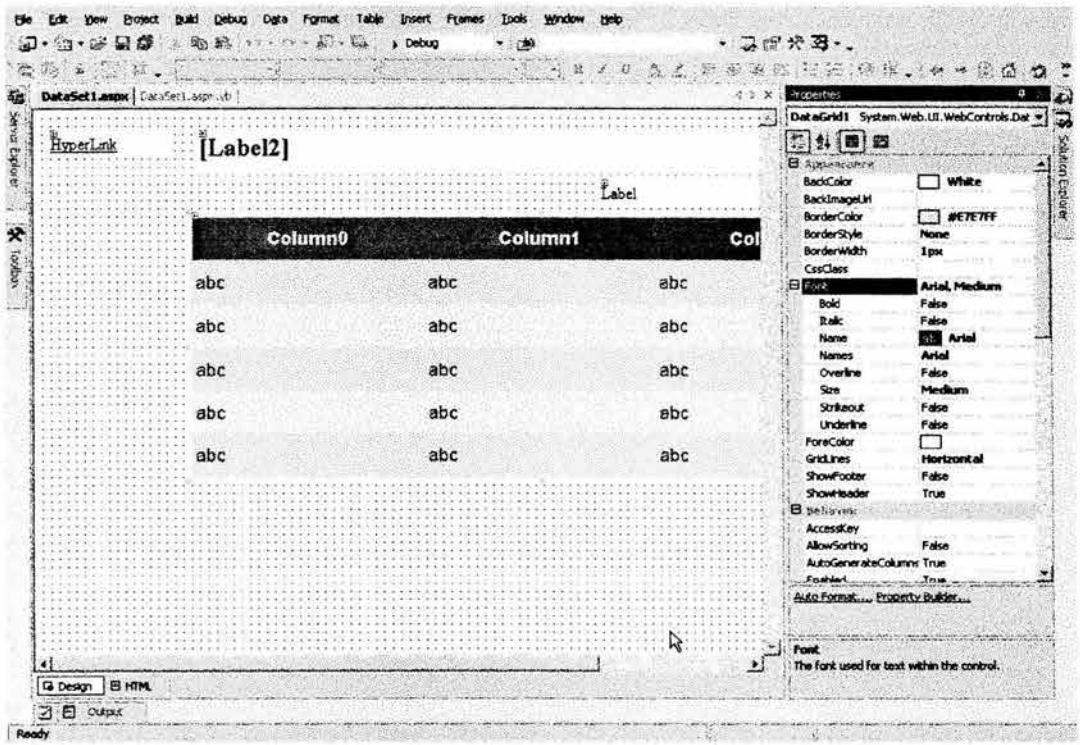
Figura 3.22: Autoformat.



Fuente: Aplicación creada por el autor de esta tesis.

Es posible configurar las propiedades del *DataGrid* por medio del panel de propiedades, como por ejemplo el tipo y tamaño de letra. Recomiendo que explore las diferentes propiedades de este control.

Figura 3.23: Modificación del DataGrid por medio del panel de propiedades.



Fuente: Aplicación creada por el autor de esta tesis.

Escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *El DataSet*. El resultado final se muestra en la figura 3.24.

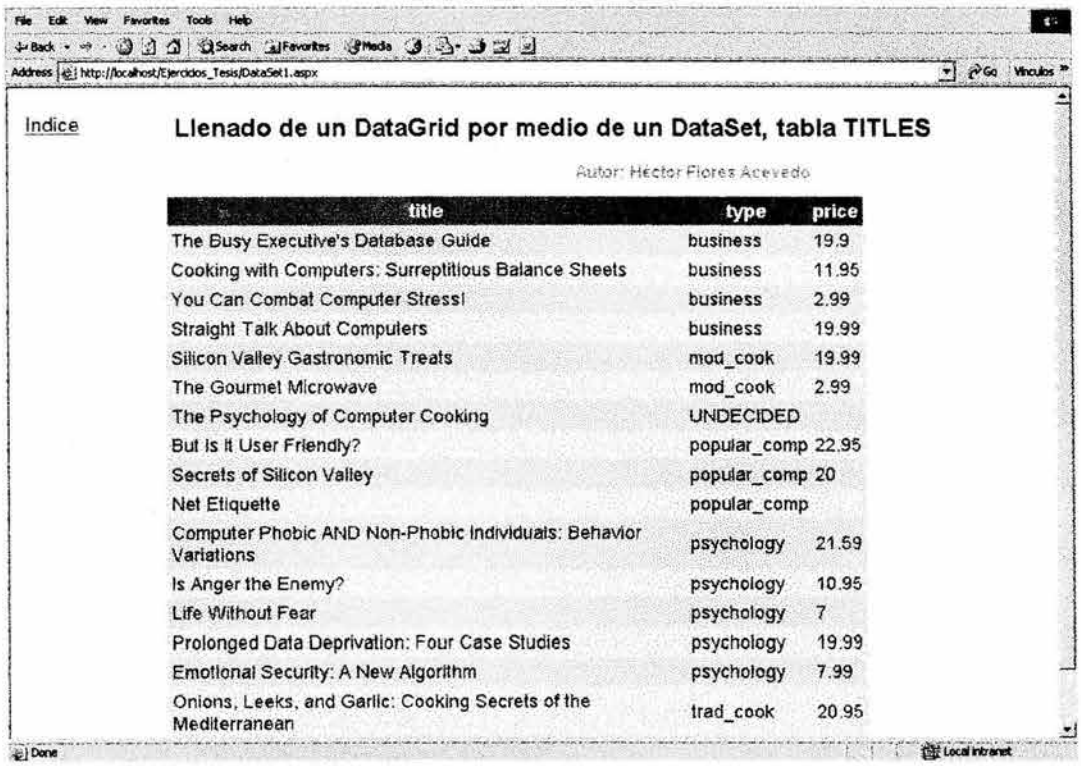
Código de DataSet1.aspx

```
'Incluyo el espacio de nombres SqlClient
Imports System.Data.SqlClient

Public Class DataSet1
    Inherits System.Web.UI.Page
    'Este codigo indica que se incluyen los siguientes
    'controles, los cuales llevan relacionados eventos.
    Protected WithEvents DataGrid1 As System.Web.UI.WebControls.DataGrid
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
```

```
#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro cada propiedad de los controles

        Label1.Text = "Autor: Héctor Flores Acevedo"
        Label1.Font.Bold = True
        Label1.Font.Size = FontUnit.Parse(12)
        Label1.ForeColor = Color.CadetBlue
        Label1.Font.Name = "Arial"

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Label2.Font.Name = "Arial"
        Label2.Font.Size = FontUnit.Parse(18)
        Label2.Font.Bold = True
        Label2.Text = "Llenado de un DataGrid por medio de un DataSet, tabla TITLES"
    End Sub

#End Region

    'Declaro una variable de tipo SqlConnection
    Dim connSql As SqlConnection

    'Declaro un adaptador de tipo Sql
    Dim AdaptadorSQL As SqlDataAdapter

    'Declaro un DataSet
    Dim DataSetSQL As DataSet

    'Este procedimiento se genera cuando se carga la página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load

        'Creo una nueva conexión
        connSql = New SqlConnection()

        'Establezco la cadena de conexión
        connSql.ConnectionString = "Integrated Security=True;" & _
                "initial catalog=pubs;data Source=(local)"

        'Abro la conexión
        connSql.Open()

        'Creo un adpatador de tipo SQL
        AdaptadorSQL = New SqlDataAdapter("SELECT title,type,price FROM titles", connSql)

        'Creo un DataSet
        DataSetSQL = New DataSet()

        'Lleno el DataSet con el contenido del adaptador.
        AdaptadorSQL.Fill(DataSetSQL, "Titulos")

        'Enlazo el DataSet al DataGrid.
        DataGrid1.DataSource = DataSetSQL.Tables("Titulos")

        'Permite enlazar el control DataGrid con el DataSet
        DataGrid1.DataBind()

    End Sub

    'Procedimiento que se genera cuando se descarga una página
    Private Sub Page_Unload(ByVal sender As Object, ByVal e As System.EventArgs) _
```

```
    Handles MyBase.Unload

        'Cierro la conexión
        connSql.Close()

    End Sub
End Class
```

Figura 3.24: Resultado final del ejercicio.



title	type	price
The Busy Executive's Database Guide	business	19.9
Cooking with Computers: Surreptitious Balance Sheets	business	11.95
You Can Combat Computer Stress!	business	2.99
Straight Talk About Computers	business	19.99
Silicon Valley Gastronomic Treats	mod_cook	19.99
The Gourmet Microwave	mod_cook	2.99
The Psychology of Computer Cooking	UNDECIDED	
But Is It User Friendly?	popular_comp	22.95
Secrets of Silicon Valley	popular_comp	20
Net Etiquette	popular_comp	
Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	21.59
Is Anger the Enemy?	psychology	10.95
Life Without Fear	psychology	7
Prolonged Data Deprivation: Four Case Studies	psychology	19.99
Emotional Security: A New Algorithm	psychology	7.99
Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	20.95

Fuente: Aplicación creada por el autor de esta tesis.

3.16 Manipulación del DataSet

Es común que además de utilizar el *DataSet* para mostrar datos necesitemos agregar, modificar y borrar datos. Debido a que el *DataSet* trabaja en forma desconectada hay que diferenciar dos partes: la primera consiste en que el *DataSet* trabaja en forma local, si se agregan datos a éste, no necesariamente significa que se estén agregando al origen de datos, es el mismo caso para la edición y para el borrado.

La segunda parte consiste en utilizar el adaptador de datos para sincronizar la información con el origen de datos.

3.16.1 Agregar renglones

Debido a que cada tabla del *DataSet* consiste en una colección de renglones y columnas, cuando se quiera agregar un nuevo renglón a la copia local, para después enlazarla al origen de datos necesitamos crear en forma programática un renglón, establecer sus características y luego agregarlo a la colección de renglones de la tabla seleccionada. Como dijimos antes, un *DataSet* puede almacenar más de una tabla de la base de datos.

Ejemplo:

```
'Creo un nuevo renglón de la tabla Titulos en la copia local
Dim MiNuevoRenglon As DataRow = _
DataSetSQL.Tables("NombreDeTabla").NewRow

MiNuevoRenglon("campo1_Tabla") = CajaDeTexto1.Text
.
.
MiNuevoRenglon("campoN_Tabla") = CajaDeTextoN.Text

'Agrego el renglón a la colección
MiDataSetSQL.Tables("NombreDeTabla").Rows.Add(MiNuevoRenglon)

'Creo el CommandBuilder para que se pueda generar
'automáticamente la sentencia SQL de inserción
MiConstructorDeComando = New SqlCommandBuilder(MiAdaptadorSQL)

'Lo debo de sincronizar con el origen de datos
MiAdaptadorSQL.Update(MiDataSetSQL, "NombreDeTabla")

'Confirmo los cambios
MiDataSetSQL.Tables("NombreDeTabla").AcceptChanges()
```

3.16.2 Editar renglones

En ADO no se requería hacer ediciones, con el simple hecho de grabar nuevamente el registro con los valores, se hacía la operación; sin embargo, en ADO.NET, es necesario indicar dónde empieza la edición y dónde termina. Se declara una variable de tipo *DataRow* que contendrá el renglón a editar, se mandará llamar el método *BeginEdit* para indicar en donde comienza la edición, al terminar de introducir los valores en el renglón, se mandará llamar la función *EndEdit* para indicar en donde termino la edición. Por último se actualiza el origen de datos, por medio del adaptador.

Ejemplo:

```
'Declaro una variable del tipo DataTable
Dim Tabla As DataTable

'Declaro una variable del tipo DataRow
Dim Renglon As DataRow
```

```
'Asigno a la variable antes declarada una tabla del DataSet
Tabla = MiDataSetSQL.Tables("MiTablaDelDataSet")

'Asigno un renglón de la tabla
Renglon = Tabla.Rows(NumeroDeRenglon)

'Comienzo la edición
Renglon.BeginEdit()

Renglon("campo1") = CajaDeTexto1.Text
.
.
Renglon("campoN") = CajaDeTextoN.Text

'Termino la edición
Renglon.EndEdit()

'Creo el CommandBuilder para que se pueda generar
'automáticamente la sentencia SQL de actualización
MiConstructorDeComandos  = New SqlCommandBuilder(MiAdaptadorSQL)

'Lo debo de sincronizar con el origen de datos
MiAdaptadorSQL.Update(MiDataSetSQL, "NombreDeTabla")

'Confirmo los cambios
MiDataSetSQL.Tables("NombreDeTabla").AcceptChanges()
```

3.16.3 Borrar renglones

Para borrar renglones tanto del *DataSet* como del origen de datos es necesario obtener el renglón que se quiere eliminar. A continuación se manda llamar el método *Delete*, quedando eliminado de la copia local, por último se sincroniza con el origen de datos por medio del adaptador, quedando eliminado también del origen de datos.

Ejemplo:

```
'Declaro una variable de tipo DataTable
Dim Tabla As DataTable

'Asigno a la variable Tabla una de las tablas del DataSet
Tabla = DataSetSQL.Tables("MiNombreDeTabla")

'Borra el renglón de la colección
Tabla.Rows(PonerNumeroDeRenglon).Delete()

'Creo el CommandBuilder para que se pueda generar
'automáticamente la sentencia SQL de eliminación
Dim MiConstructorDeComandos As New SqlCommandBuilder(MiAdaptadorSQL)

'Sincronizo los cambios con la base de datos.
AdaptadorSQL.Update(MiDataSetSQL, "MiNombreDeTabla")

'Confirmo los cambios
DataSetSQL.Tables("MiNombreDeTabla").AcceptChanges()
```

Ejercicio 3.6

Objetivo:

Describir la forma de agregar, editar y eliminar registros en un origen de datos, utilizando como interfase una página activa de servidor. En este caso se utiliza el *espacio de nombres sqlClient*, aunque se podría también utilizar el *espacio de nombres OleDbClient* variando el código.

Es importante hacer hincapié en que cada vez que se genera un evento, como por ejemplo un clic, la página es regenerada completamente, es por eso que se hace uso de la instrucción ***IsPostBack***, que checa que las instrucciones que se encuentren dentro de ella, solamente se ejecuten la primera vez que se cargue la página. Otro punto importante es que no es conveniente almacenar valores en las variables locales, porque estos valores se pierden. Es por eso que en el archivo *Global.asax*, que es creado automáticamente, asignamos valores que utilizaremos en el transcurso de la aplicación.

Hay dos formas de almacenamiento, una por aplicación, que quiere decir que cada que se inicie la aplicación se guardará o manipulará la variable sin importar cuántas sesiones se creen. La otra es por sesión, en donde cada vez que se cree una sesión se guardará o manipulará la variable. Dentro del código se utiliza la variable *EsAgregar*, su valor depende si el usuario ha oprimido el botón de *Agregar* ó *Editar*. Debido a que se utiliza un solo procedimiento para grabar la información, esta variable indica si su valor es verdadero, que se debe agregar un nuevo registro, si su valor es falso, entonces se debe modificar un registro ya existente.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **AddEditDelete.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

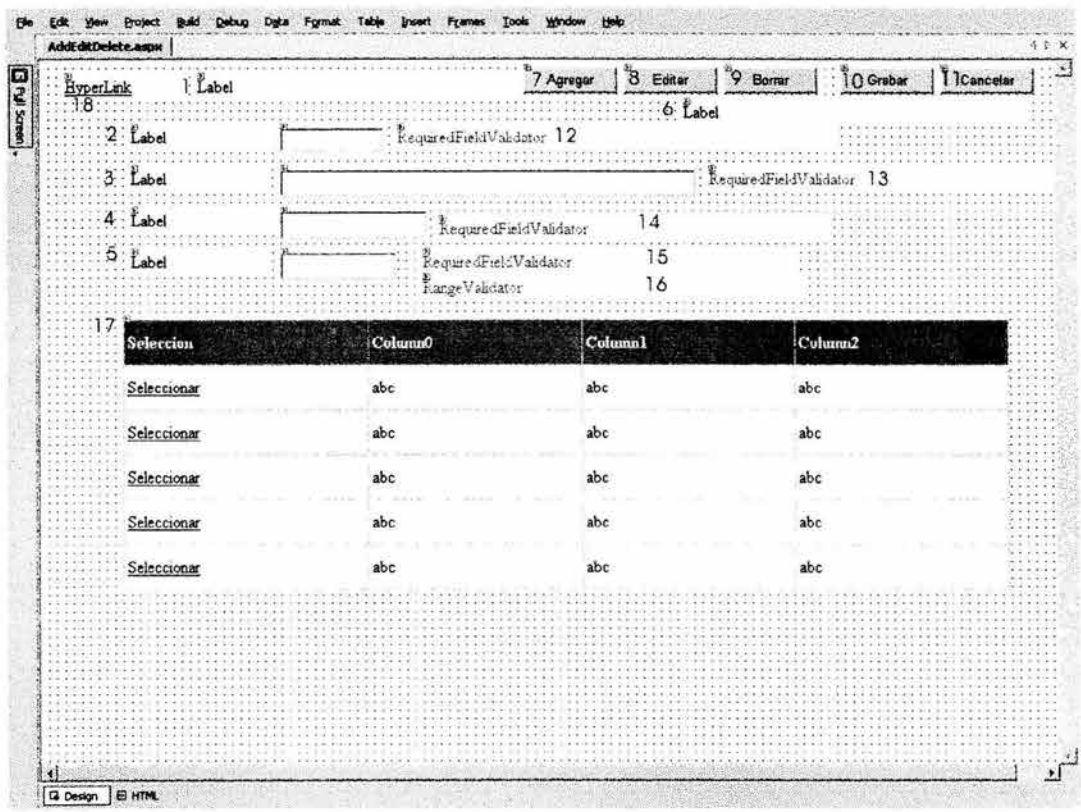
Agregue los siguientes controles ordenados de acuerdo a la figura 3.25: 1)Label1, 2)Label2, 3)Label3, 4)Label4, 5)Label5, 6)Label6, 7) Button con ID=cmdAgregar, 8)Button con ID=cmdEditar, 9)Button con ID=cmdBorrar, 10)Button

con ID=cmdGrabar, 11)Button con ID=cmdCancelar, 12)RequiredFieldValidator1, 13)RequiredFieldValidator2, 14)RequiredFieldValidator3, 15)RequiredFieldValidator4, 16)RangeValidator1, 17)DataGrid1, 18) HyperLink1.

Después de agregar el control *DataGrid* haga clic derecho sobre éste, seleccione *Auto Format* y por último *Professional 2,* como se muestra en la figura 3.26.

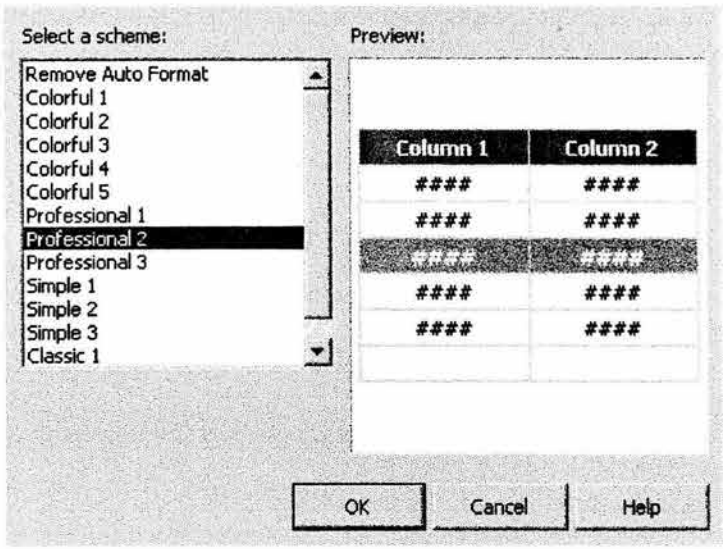
Para poder seleccionar un renglón completo, es necesaria una columna extra que permita realizar esta operación. En propiedades seleccione *Columns* y haga clic en *(Collection)*. En propiedades (véase figura 3.27) haga clic en *Button Column* y agregue *Select*, después establezca el texto de cabecera, el *Command name* es el nombre con el cual será identificado en forma programática. Después de establecer estas propiedades oprima el botón *OK*.

Figura 3.25: Controles de la forma Web AddEditDelete.aspx.



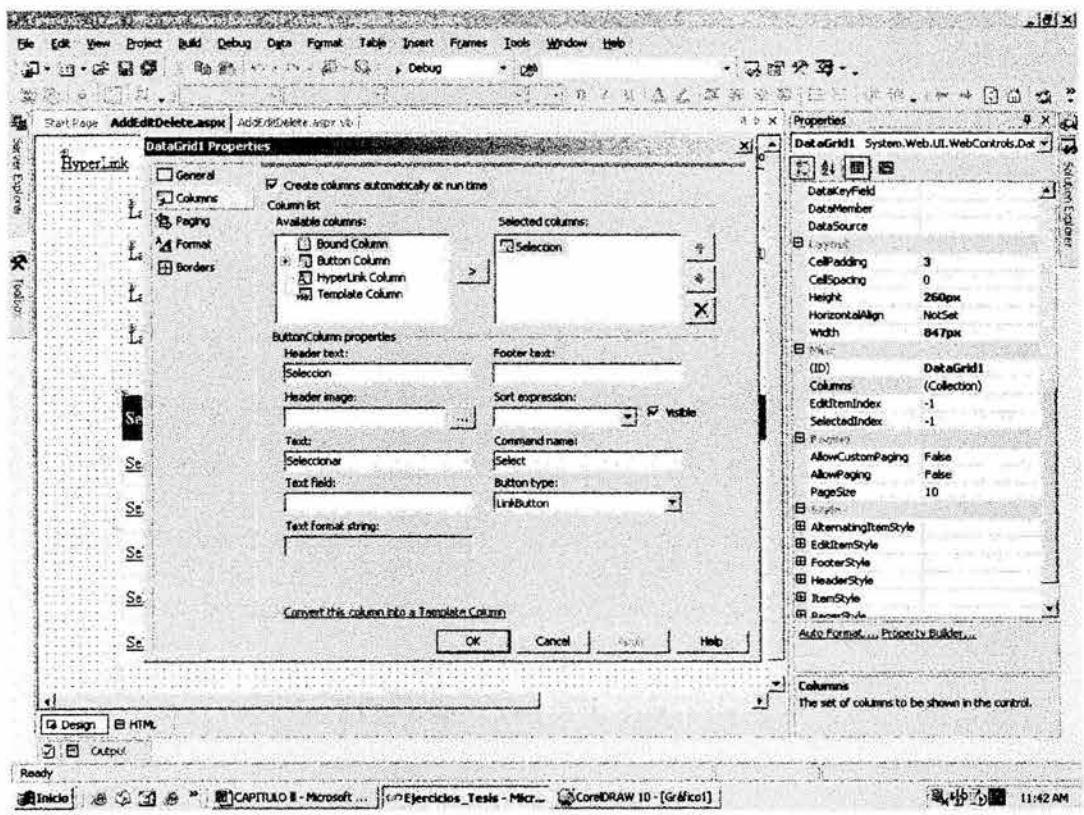
Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.26: Auto Format.



Fuente: Aplicación creada por el autor de esta tesis.

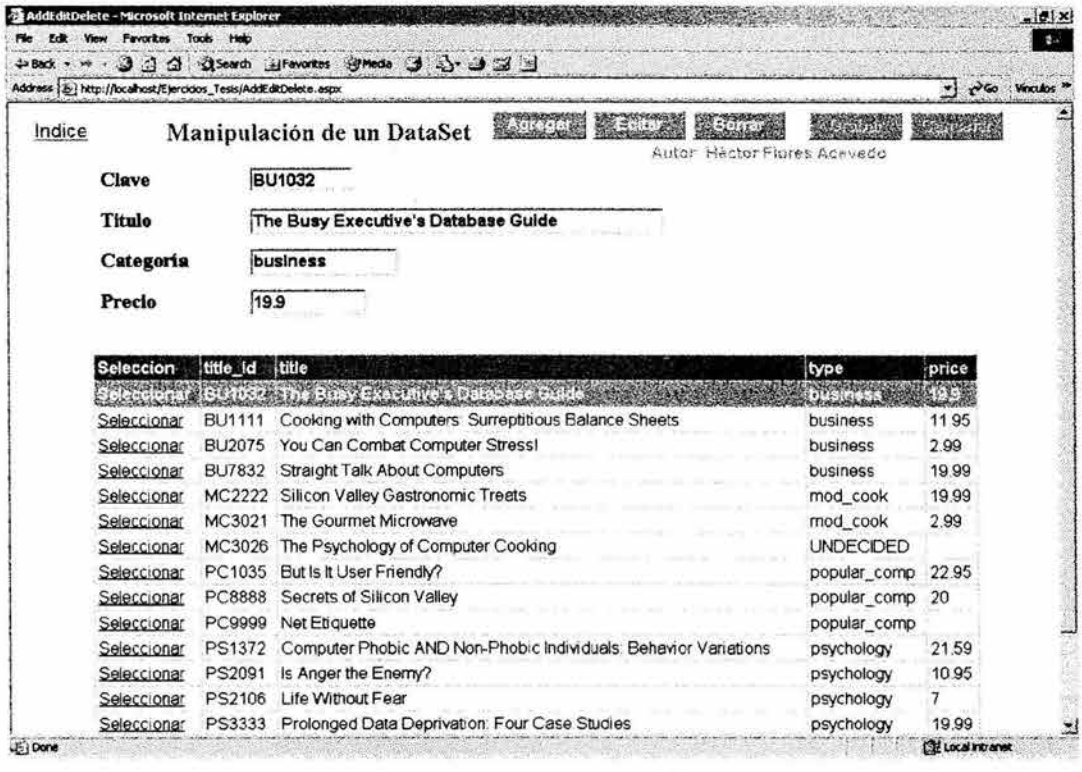
Figura 3.27: Propiedad *Columns* de *Misc*.



Fuente: Aplicación creada por el autor de esta tesis.

El control *DataGrid* es el más utilizado para mostrar datos, así como para manipularlos, es conveniente que el lector profundice más en el tema estudiando la ayuda en línea. Es posible aumentar columnas que tengan funcionalidades de edición, borrado, actualización y cancelación. El resultado final se muestra en la figura 3.28.

Figura 3.28: Resultado final de la forma Web AddEditDelete.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

Escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Manipulación del DataSet*.

Código de AddEditDelete.aspx

```
'Indico que todas las variables deben de ser declaradas para su uso.
Option Explicit On

'Espacio de nombres SqlClient
Imports System.Data.SqlClient

Public Class AddEditDelete
    Inherits System.Web.UI.Page

    'Todos los controles agregados, indica
    'que todos tienen asociados eventos
    Protected WithEvents DataGrid1 As System.Web.UI.WebControls.DataGrid
```

```
    Protected WithEvents cmdAgregar As System.Web.UI.WebControls.Button
    Protected WithEvents cmdEditar As System.Web.UI.WebControls.Button
    Protected WithEvents cmdBorrar As System.Web.UI.WebControls.Button
    Protected WithEvents cmdGrabar As System.Web.UI.WebControls.Button
    Protected WithEvents cmdCancelar As System.Web.UI.WebControls.Button
    Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox
    Protected WithEvents TextBox2 As System.Web.UI.WebControls.TextBox
    Protected WithEvents TextBox3 As System.Web.UI.WebControls.TextBox
    Protected WithEvents RequiredFieldValidator1 As _
                         System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RequiredFieldValidator2 As _
                        System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RequiredFieldValidator3 As _
                        System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RequiredFieldValidator4 As _
                        System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents RangeValidator1 As System.Web.UI.WebControls.RangeValidator
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label6 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBox4 As System.Web.UI.WebControls.TextBox

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'En esta parte de código se ponen las propiedades de los controles
        'como color, tamaño, tipo de letra, etc.

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = _
                    "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        DataGrid1.Font.Name = "Arial"

        TextBox1.Font.Size = FontUnit.Parse(12)
        TextBox1.Font.Bold = True
        TextBox1.ForeColor = Color.DarkBlue

        TextBox2.Font.Size = FontUnit.Parse(12)
        TextBox2.Font.Bold = True
        TextBox2.ForeColor = Color.DarkBlue

        TextBox3.Font.Size = FontUnit.Parse(12)
        TextBox3.Font.Bold = True
        TextBox3.ForeColor = Color.DarkBlue

        TextBox4.Font.Size = FontUnit.Parse(12)
        TextBox4.Font.Bold = True
        TextBox4.ForeColor = Color.DarkBlue

        Label1.Font.Size = FontUnit.Parse(18)
        Label1.Font.Bold = True
        Label1.Text = "Manipulación de un DataSet"
        Label1.ForeColor = Color.DarkGreen

        Label2.Font.Size = FontUnit.Parse(15)
        Label2.Font.Bold = True
        Label2.Text = "Clave"
        Label2.ForeColor = Color.Black
```

```
Label3.Font.Size = FontUnit.Parse(15)
Label3.Font.Bold = True
Label3.Text = "Titulo"
Label3.ForeColor = Color.Black

Label4.Font.Size = FontUnit.Parse(15)
Label4.Font.Bold = True
Label4.Text = "Categoría"
Label4.ForeColor = Color.Black

Label5.Font.Size = FontUnit.Parse(15)
Label5.Font.Bold = True
Label5.Text = "Precio"
Label5.ForeColor = Color.Black

Label6.Text = "Autor: Héctor Flores Acevedo"
Label6.Font.Bold = True
Label6.Font.Size = FontUnit.Parse(12)
Label6.ForeColor = Color.CadetBlue
Label6.Font.Name = "Arial"

'Inidico el control a validar
RequiredFieldValidator1.ControlToValidate = "TextBox1"
'Defino el tipo de letra
RequiredFieldValidator1.Font.Name = "Arial"
'Defino el tamaño de la letra
RequiredFieldValidator1.Font.Size = FontUnit.Parse(12)
RequiredFieldValidator1.ErrorMessage = _
           "Debes de escribir un código en la caja de texto"
RequiredFieldValidator1.Display = ValidatorDisplay.Static
'Esta propiedad permite que se pueda validar la caja de texto en forma
'programática, antes que se despliegue el mensaje de error
RequiredFieldValidator1.EnableClientScript = False

'Inidico el control a validar
RequiredFieldValidator2.ControlToValidate = "TextBox2"
'Defino el tipo de letra
RequiredFieldValidator2.Font.Name = "Arial"
'Defino el tamaño de la letra
RequiredFieldValidator2.Font.Size = FontUnit.Parse(12)
RequiredFieldValidator2.ErrorMessage = _
           "Debes de escribir un título en la caja de texto"
RequiredFieldValidator2.Display = ValidatorDisplay.Static
'Esta propiedad permite que se pueda validar la caja de texto en forma
'programática, antes que se despliegue el mensaje de error
RequiredFieldValidator2.EnableClientScript = False

'Inidico el control a validar
RequiredFieldValidator3.ControlToValidate = "TextBox3"
'Defino el tipo de letra
RequiredFieldValidator3.Font.Name = "Arial"
'Defino el tamaño de la letra
RequiredFieldValidator3.Font.Size = FontUnit.Parse(12)
RequiredFieldValidator3.ErrorMessage = _
                  "Debes de escribir una categoría en la caja de texto"
RequiredFieldValidator3.Display = ValidatorDisplay.Static
'Esta propiedad permite que se pueda validar la caja de texto en forma
'programática, antes que se despliegue el mensaje de error
RequiredFieldValidator3.EnableClientScript = False

'Inidico el control a validar
RequiredFieldValidator4.ControlToValidate = "TextBox4"
'Defino el tipo de letra
RequiredFieldValidator4.Font.Name = "Arial"
'Defino el tamaño de la letra
RequiredFieldValidator4.Font.Size = FontUnit.Parse(12)
RequiredFieldValidator4.ErrorMessage = _
                      "Debes de escribir una cifra en la caja de texto"
RequiredFieldValidator4.Display = ValidatorDisplay.Static
'Esta propiedad permite que se pueda validar la caja de texto en forma
'programática, antes que se despliegue el mensaje de error
RequiredFieldValidator4.EnableClientScript = False

RangeValidator1.ControlToValidate = "TextBox4"
'Defino el tipo de letra
RangeValidator1.Font.Name = "Arial"
'Defino el tamaño de la letra
```

```
        RangeValidator1.Font.Size = FontUnit.Parse(12)
        RangeValidator1.ErrorMessage = "El valor debe de ser mayor que cero"
        'Valor minimo y maximo, rango en que debe de estar el valor de
        'la caja de texto.
        RangeValidator1.MinimumValue = 0
        RangeValidator1.MaximumValue = 10000
        'Tipo de valor a validar
        RangeValidator1.Type = ValidationDataType.Double
        'Esta propiedad permite que se pueda validar la caja de texto en forma
        'programática, antes que se despliegue el mensaje de error
        RangeValidator1.EnableClientScript = False

        cmdAgregar.Text = "Agregar"
        cmdAgregar.Font.Name = "Arial"
        cmdAgregar.Font.Bold = True
        cmdAgregar.Font.Size = FontUnit.Parse(12)
        cmdAgregar.ForeColor = Color.White
        cmdAgregar.BackColor = Color.CadetBlue

        cmdEditar.Text = "Editar"
        cmdEditar.Font.Name = "Arial"
        cmdEditar.Font.Bold = True
        cmdEditar.Font.Size = FontUnit.Parse(12)
        cmdEditar.ForeColor = Color.White
        cmdEditar.BackColor = Color.CadetBlue

        cmdBorrar.Text = "Borrar"
        cmdBorrar.Font.Name = "Arial"
        cmdBorrar.Font.Bold = True
        cmdBorrar.Font.Size = FontUnit.Parse(12)
        cmdBorrar.ForeColor = Color.White
        cmdBorrar.BackColor = Color.CadetBlue

        cmdGrabar.Text = "Grabar"
        cmdGrabar.Font.Name = "Arial"
        cmdGrabar.Font.Bold = True
        cmdGrabar.Font.Size = FontUnit.Parse(12)
        cmdGrabar.ForeColor = Color.White
        cmdGrabar.BackColor = Color.CadetBlue

        cmdCancelar.Text = "Cancelar"
        cmdCancelar.Font.Name = "Arial"
        cmdCancelar.Font.Bold = True
        cmdCancelar.Font.Size = FontUnit.Parse(12)
        cmdCancelar.ForeColor = Color.White
        cmdCancelar.BackColor = Color.CadetBlue

    End Sub

#End Region

    'Variable de tipo conexion de SQL
    Dim connSQL As SqlConnection

    'Coleccion de celdas de la tabla.
    Dim CeldaGrid As TableCellCollection

    'Declaro el objeto DataSet
    Dim DataSetSQL As DataSet

    'Variable de tipo adaptador de SQL
    Dim AdaptadorSQL As SqlDataAdapter


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

        'Variable de tipo texto, para guardar la cadena de conexión
        Dim ConexionSQL As String

        'Creo la conexión
        connSQL = New SqlConnection()

        'Cadena de conexión
        ConexionSQL = _
        "Integrated Security=True;Initial Catalog=Pubs;Data Source=(local)"
```

```
        'Asigno la cadena de conexión a la variable de tipo conexión
        connSQL.ConnectionString = ConexionSQL
        'Mando llamar un procedimiento para poder cargar el DataGrid
        Call CargarDataGrid(connSQL)

        'En caso que sea la primera vez que se carga la página
        If Not IsPostBack Then

            'Mando llamar un procedimiento para habilitar los botones de
            'Agregar, Borrar y Editar. Deshabilitando Guardar y Cancelar.
            Call HabilitarBotones()

            'Permito que se vea el DataGrid, pero solamente la primera vez
            'que cargo la página.
            DataGrid1.Visible = True
        End If

        'Reestablezco Label6, porque lo he utilizado para desplegar
        'mensajes de error.
        Label6.Text = "Autor: Héctor Flores Acevedo"
    End Sub

    'Deshabilito cancelar y grabar
    Private Sub HabilitarBotones()
        Me.cmdAgregar.Enabled = True
        Me.cmdEditar.Enabled = True
        Me.cmdBorrar.Enabled = True
        Me.cmdCancelar.Enabled = False
        Me.cmdGrabar.Enabled = False
        'No permito que se pueda escribir en cajas
        Me.TextBox1.ReadOnly = True
        Me.TextBox2.ReadOnly = True
        Me.TextBox3.ReadOnly = True
        Me.TextBox4.ReadOnly = True
    End Sub

    'Habilito cancelar y grabar
    Private Sub DesHabilitarBotones()
        Me.cmdAgregar.Enabled = False
        Me.cmdEditar.Enabled = False
        Me.cmdBorrar.Enabled = False
        Me.cmdCancelar.Enabled = True
        Me.cmdGrabar.Enabled = True
        'Permito que se pueda escribir en cajas
        Me.TextBox1.ReadOnly = False
        Me.TextBox2.ReadOnly = False
        Me.TextBox3.ReadOnly = False
        Me.TextBox4.ReadOnly = False

    End Sub

    'Este procedimiento se genera cuando quiero cargar el DataGrid
    Private Sub CargarDataGrid(ByVal Conexion As SqlConnection)

        'Variable de tipo String
        Dim sQry As String

        'Creo el objeto DataSet
        DataSetSQL = New DataSet()

        'Escribo una sentencia de Transact-SQL
        sQry = "SELECT title_id,title,type,price FROM titles ORDER BY title_id"

        'Creo el objeto adaptador
        AdaptadorSQL = New SqlDataAdapter(sQry, Conexion)

        'Lleno el adaptador con la tabla Titles
        AdaptadorSQL.Fill(DataSetSQL, "Titulos")

        'Indico la fuente con el que se llenará el DataGrid
        DataGrid1.DataSource = DataSetSQL

        'Para que no se reescriban viejos valores
        'Si es la primera vez
        If IsPostBack = False Then
```

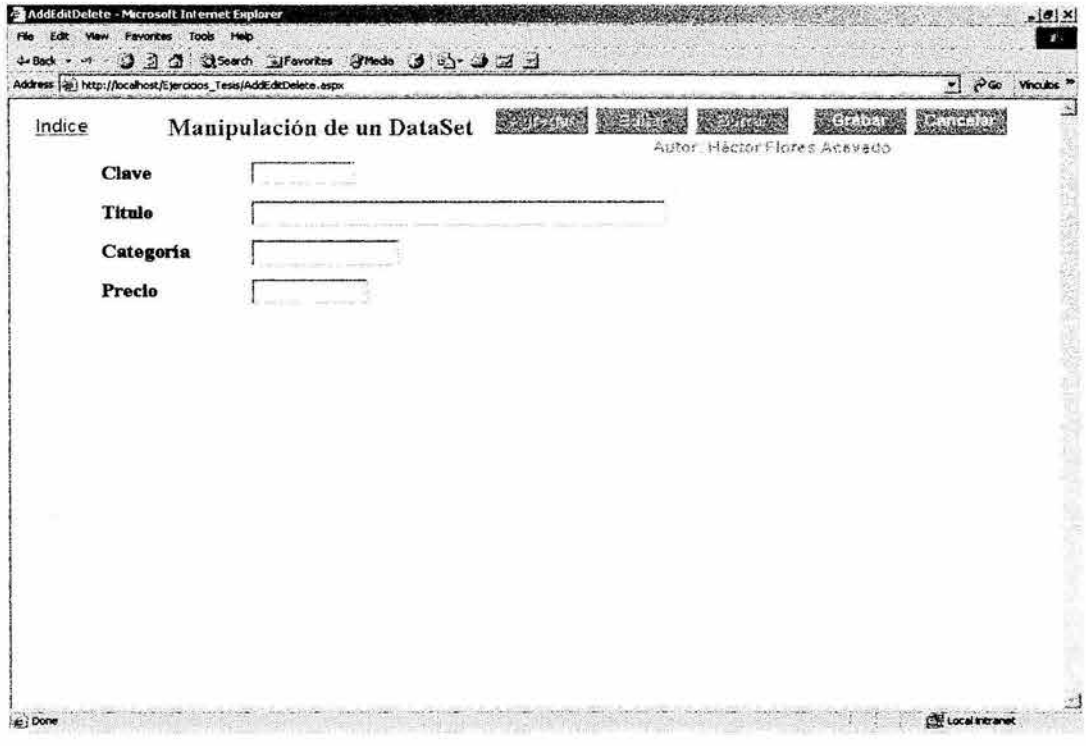
```
        'Uno el DataGrid con el contenido de la tabla Titles
        DataGrid1.DataBind()

        'selecciono el primer elemento
        DataGrid1.SelectedIndex = 0

        'Pongo todas las celdas del Item cero que es el primer elemento.
        CeldaGrid = DataGrid1.Items(0).Cells()

        'Mando llamar un procedimiento para llenar las cajas de texto
        Call LlenarCajas(CeldaGrid)
    End If
End Sub

'Procedimiento que se genera cuando oprimo la primer columna,
'donde se encuentra seleccionar
Private Sub DataGrid1_ItemCommand(ByVal source As Object, ByVal e As _
System.Web.UI.WebControls.DataGridCommandEventArgs) Handles _
DataGrid1.ItemCommand

    'Checo si el tipo de comando es el de seleccionar
    If e.CommandName = "Select" Then

        'Se seleccionara el renglón seleccionado.
        DataGrid1.SelectedIndex = e.Item.ItemIndex

        'Cargo todas las celdas
        CeldaGrid = e.Item.Cells

        'Mando llamar un procedimiento para llenar las cajas de texto
        Call LlenarCajas(CeldaGrid)
    End If
End Sub

'Procedimiento para llenar las cajas de texto
Private Sub LlenarCajas(ByVal CeldaGrid As TableCellCollection)
    TextBox1.Text = CeldaGrid(1).Text
    TextBox2.Text = CeldaGrid(2).Text
    TextBox3.Text = CeldaGrid(3).Text
    TextBox4.Text = CeldaGrid(4).Text
End Sub

'Procedimiento que se genera cuando oprimo el botón de Agregar
Private Sub cmdAgregar_Click(ByVal sender As System.Object, ByVal e As _
System.EventArgs) Handles cmdAgregar.Click

    'La variable de sesión 'EsAgregar', la pongo en verdadero.
    'Indica que se va a insertar un valor, para que cuando se
    'oprima el botón de Grabar se realice dicha operación
    Session("EsAgregar") = True
    'Vacío la caja de texto
    TextBox1.Text = String.Empty
    TextBox2.Text = String.Empty
    TextBox3.Text = String.Empty
    TextBox4.Text = String.Empty
    Call DesHabilitarBotones()
    'Deshabilito el DataGrid
    DataGrid1.Visible = False
End Sub

'Procedimiento que se genera cuando oprimo el botón de Grabar
Private Sub cmdGrabar_Click(ByVal sender As System.Object, ByVal e As _
System.EventArgs) Handles cmdGrabar.Click

    'Valido las cajas de texto
    If RequiredFieldValidator1.IsValid And RequiredFieldValidator2.IsValid And _
        RequiredFieldValidator3.IsValid And RequiredFieldValidator4.IsValid _
        And RangeValidator1.IsValid Then

        'Variable de tipo SqlCommandBuilder
        Dim sqlCommBuilder As SqlCommandBuilder

        'La variable "EsAgregar" se definió en Global.axa, ya que
        'las páginas ASP no recuerdan las variables, porque
        'la página se regenera completamente después de un evento.
        If Session("EsAgregar") = True Then
```

```
        Try
            'Creo un nuevo renglón de la tabla Titulos en la copia local
            Dim NuevoRenglon As DataRow = DataSetSQL.Tables("Titulos").NewRow
            'Creo una vista, para que muestre los datos ordenados
            Dim vista As New DataView()

            'Pongo el valor de las cajas de texto en el nuevo renglón
            NuevoRenglon("title_id") = TextBox1.Text
            NuevoRenglon("title") = TextBox2.Text
            NuevoRenglon("type") = TextBox3.Text
            NuevoRenglon("price") = Double.Parse(TextBox4.Text)

            'Agrego el renglon a la colección de renglones
            DataSetSQL.Tables("Titulos").Rows.Add(NuevoRenglon)

            'Creo el CommandBuilder para que se pueda generar
            'automáticamente la sentencia SQL de inserción
            sqlCommBuilder = New SqlCommandBuilder(AdaptadorSQL)

            'Sincronizo con el origen de datos
            AdaptadorSQL.Update(DataSetSQL, "Titulos")

            'Confirmo los cambios
            DataSetSQL.Tables("Titulos").AcceptChanges()

            'Le cargo a la vista la del DataSetSQL "Titulos"
            vista.Table = DataSetSQL.Tables("Titulos")

            'Establezco un criterio de ordenación de menor a mayor
            vista.Sort = "title_ID ASC"

            'Cargo el DataGrid con el valor de la vista
            DataGrid1.DataSource = vista

            'Actualizo el control DataGrid
            DataGrid1.DataBind()

            'Me posiciono en el renglon donde se encuentra el nuevo libro
            'vista.Find() devuelve el renglón donde se encuentra la columna
            'que sirve como llave, en este caso title_id, es por eso que
            'pongo el valor de la primer caja de texto como parámetro.
            DataGrid1.SelectedIndex = vista.Find(TextBox1.Text)

            'Obtengo los valores del nuevo libro, a partir de su posición
            'en el DataGrid
            CeldaGrid = DataGrid1.Items(DataGrid1.SelectedIndex).Cells()

            'Procedimiento para llenar las cajas de texto
            Call LlenarCajas(CeldaGrid)

        Catch mensaje As Exception
            Label6.Text = "Error, la clave de ese libro ya existe"
            Exit Sub
        End Try

    'En caso de que sea edición
Else
            'Declaro una variable de tipo DataTable
            Dim Tabla As DataTable

            'Declaro una variable de tipo DataRow
            Dim Renglon As DataRow

            'Le asigno la tabla "Titulos" a la variable Tabla
            Tabla = DataSetSQL.Tables("Titulos")

            'Le asigno el renglón seleccionado del DataGrid de la tabla
            '"Titulos"
            Renglon = Tabla.Rows(DataGrid1.SelectedIndex)

            'Comienzo la edición
            Renglon.BeginEdit()

            'Asigno al renglón el contenido de las cajas de texto
            Renglon("title_id") = TextBox1.Text
            Renglon("title") = TextBox2.Text
            Renglon("type") = TextBox3.Text
```

```
                Renglon("price") = CDbl(TextBox4.Text)

                'Termino la edición
                Renglon.EndEdit()

                'Creo el CommandBuilder para que se pueda generar
                'automáticamente la sentencia SQL de insercion
                sqlCommBuilder = New SqlCommandBuilder(AdaptadorSQL)

                'Lo debo de sincronizar con el origen de datos
                AdaptadorSQL.Update(DataSetSQL, "Titulos")

                'Confirmo los cambios
                DataSetSQL.Tables("Titulos").AcceptChanges()

                'Actualizo el control DataGrid
                DataGrid1.DataBind()

            End If

            'Llamo el procedimiento para habilitar los botones de
            'Edición, Borrado e Inserción
            Call HabilitarBotones()

            'Permito que se vea el DataGrid
            DataGrid1.Visible = True
        End If
    End Sub

    'En caso de que se desee cancelar toda operación
    Private Sub cmdCancelar_Click(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles cmdCancelar.Click

        'Pongo las validaciones a verdadero, porque no importa si se
        'dejan cajas vacias ó valores incorrectos porque
        'se está haciendo una cancelación

        RequiredFieldValidator1.IsValid = True
        RequiredFieldValidator2.IsValid = True
        RequiredFieldValidator3.IsValid = True
        RequiredFieldValidator4.IsValid = True
        RangeValidator1.IsValid = True

        'Almaceno en CeldaGrid los valores de la fila que tiene seleccionado
        'el DataGrid
        CeldaGrid = DataGrid1.Items(DataGrid1.SelectedIndex).Cells()

        'Lleno las cajas con los valores de CeldaGrid.
        Call LlenarCajas(CeldaGrid)

        Call HabilitarBotones()
        'Habilito el DataGrid
        DataGrid1.Visible = True
    End Sub

    'Procedimiento que se genera cuando oprimo el Botón de Borrar
    Private Sub cmdBorrar_Click(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles cmdBorrar.Click

        'Declaro una variable de tipo DataTable
        Dim Tabla As DataTable

        'Declaro una variable de tipo Long
        Dim Renglon As Long

        'Le asigno a la variable Tabla, la tabla "Titulos"
        Tabla = DataSetSQL.Tables("Titulos")

        'Asigno a la variable Renglon, el renglón seleccionado del DataGrid
        Renglon = DataGrid1.SelectedIndex

        'Borra el renglon de la colección
        Tabla.Rows(Renglon).Delete()

        'Creo el CommandBuilder para que se pueda generar
        'automaticamente la sentencia SQL de borrado
        Dim sqlCommBuilder As New SqlCommandBuilder(AdaptadorSQL)
```

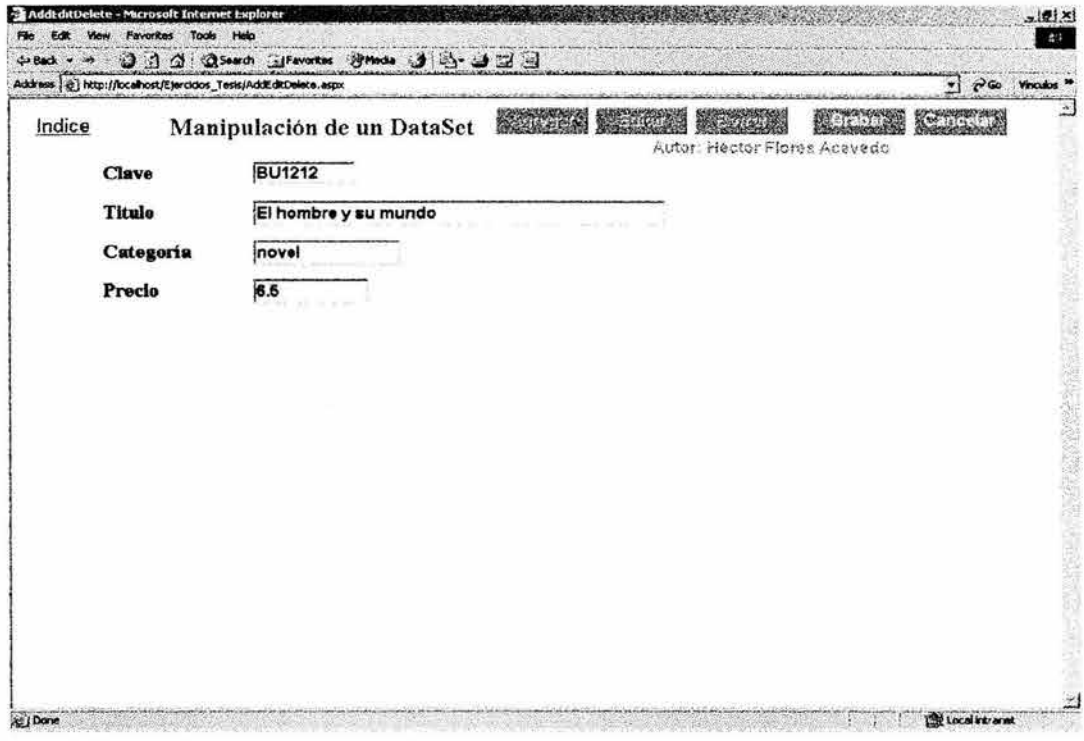
```
        'Sincronizo los cambios con la base de datos.
        AdaptadorSQL.Update(DataSetSQL, "Titulos")
        'Confirmo los cambios
        DataSetSQL.Tables("Titulos").AcceptChanges()

        'Actualizo el control DataGrid
        DataGrid1.DataBind()

        'Automáticamente se posiciona en el siguiente renglón
        CeldaGrid = DataGrid1.Items(DataGrid1.SelectedIndex).Cells()

        'Actualiza las cajas de texto
        Call LlenarCajas(CeldaGrid)
        Call HabilitarBotones()

    End Sub

    'Procedimiento que se genera cuando se oprime el botón de edición
    Private Sub cmdEditar_Click(ByVal sender As System.Object, ByVal e As _
    System.EventArgs) Handles cmdEditar.Click

        'Significa que no estoy agregando, lo utilizo como
        'bandera para el momento en que oprima Grabar.
        Session("EsAgregar") = False

        Call DesHabilitarBotones()
        'Cuando se edita no se puede permitir que se modifique el código
        'del producto, en todo caso se debe de borrar y agregar como un
        'nuevo producto.
        TextBox1.ReadOnly = True
        'No permito que se vea el DataGrid
        DataGrid1.Visible = False
    End Sub

End Class
```

Figura 3.29: Se oprime el botón de Agregar.



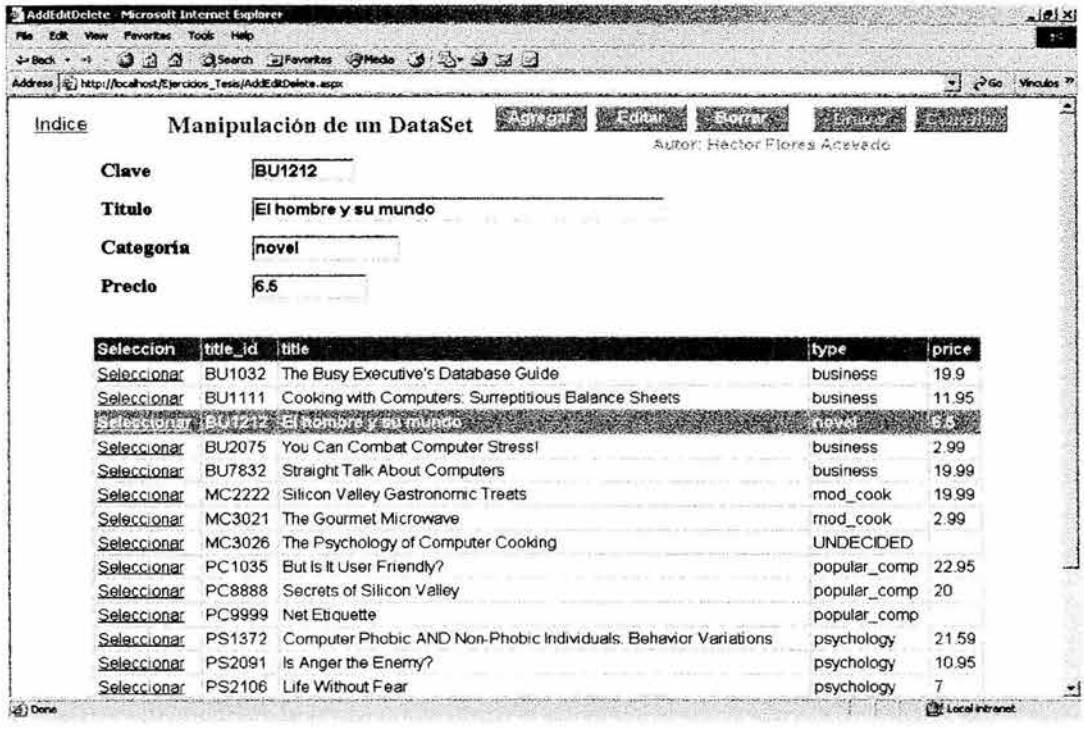
Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.30: Se da de alta un nuevo libro.



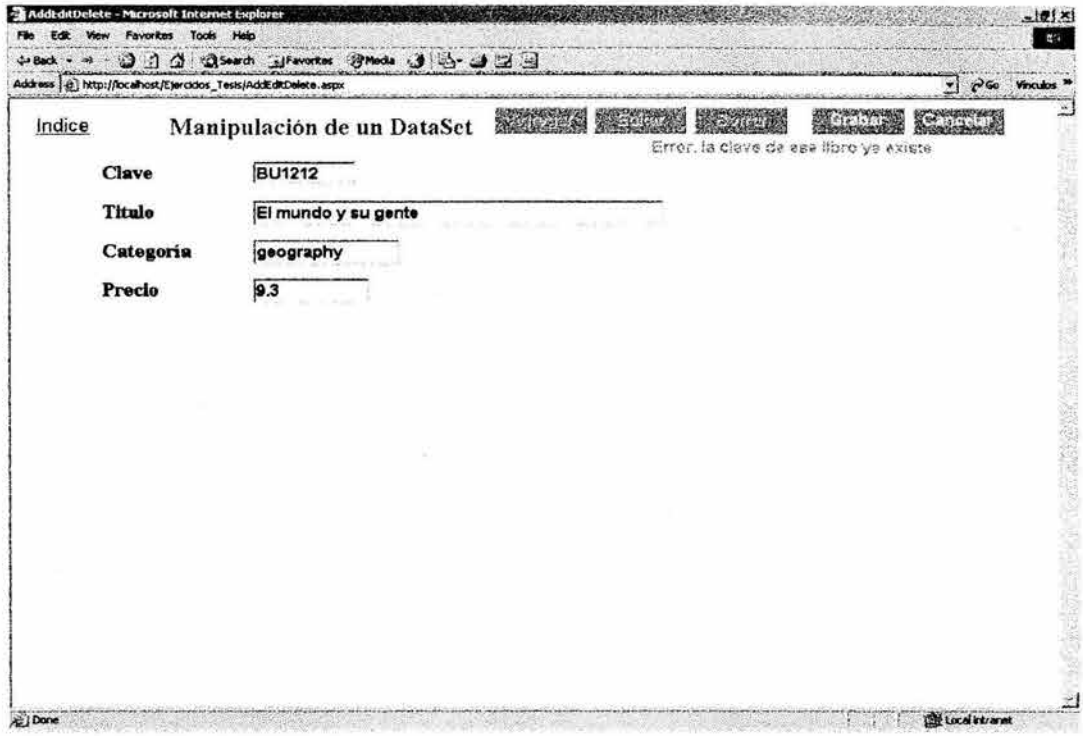
Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.31: Se oprime el botón de Grabar y aparece en el DataGrid.



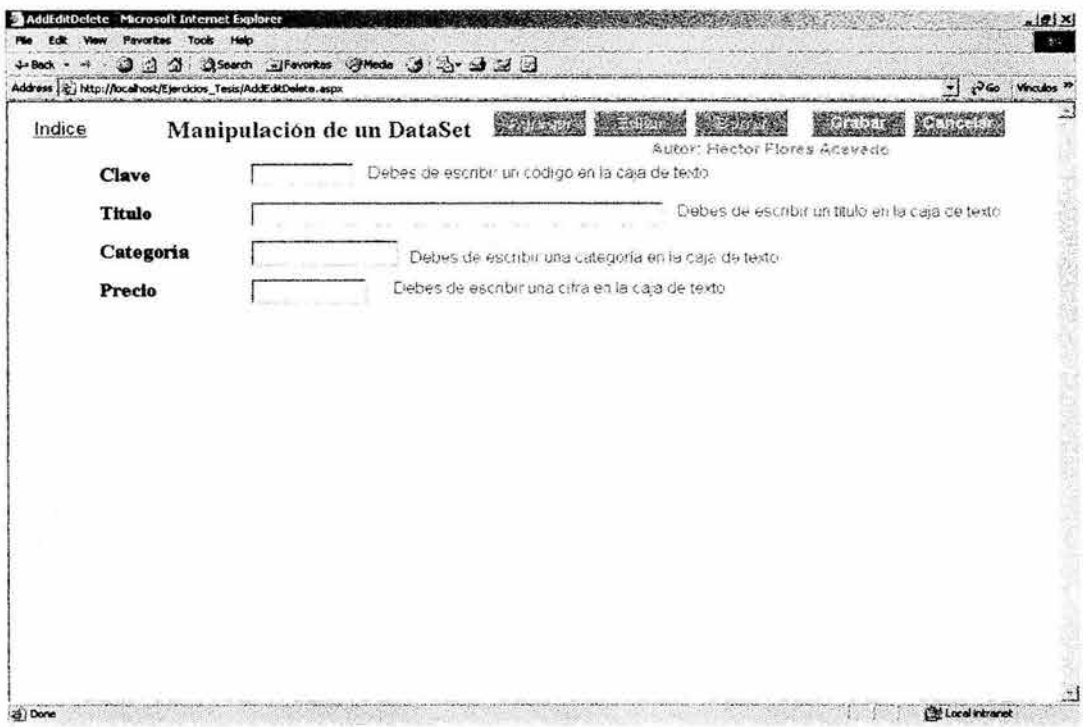
Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.32: Se intenta dar de alta un libro con la misma clave, sin éxito.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.33: Se intenta dar de alta un libro sin introducir datos en las cajas de texto, sin éxito.

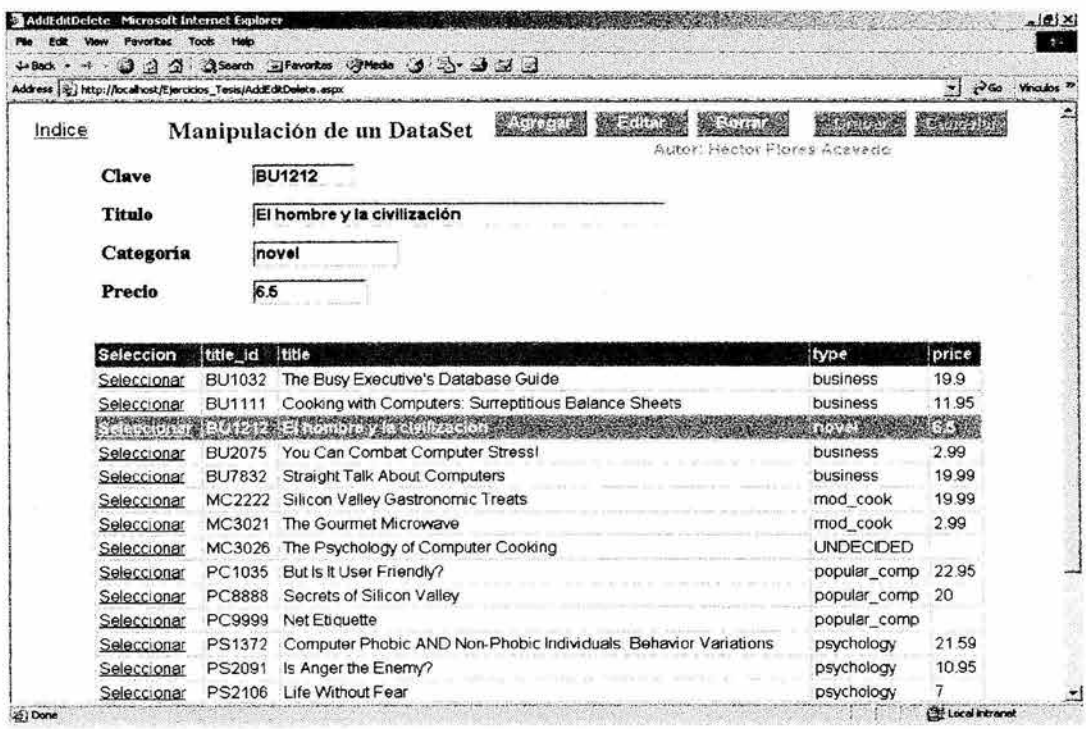


Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.34: Se edita el libro que se agregó anteriormente.

Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.35: Cuando oprime el botón Grabar, aparece en el DataGrid con las modificaciones.



Fuente: Aplicación creada por el autor de esta tesis.

En casos de edición y de inserción de registros, es posible oprimir el botón *Cancelar,* para regresar la forma Web a su estado original.

3.17 Relaciones

El poder establecer relaciones en un manejador de base de datos es una de las características más importantes, porque permite mantener una estructura y una relación entre los datos.

En Visual Basic 6.0 y particularmente en ADO, es posible obtener la información de un origen de datos, pero no las relaciones, ni las restricciones establecidas en el manejador de base de datos. ADO.NET establece la clase *DataRelation*, que permite mantener la integridad referencial por medio de las relaciones.

Para poder crear una relación entre dos tablas, es necesario crear un adaptador que a su vez cargará la tabla primaria y la relacionada dentro de un *DataSet*, hay que recordar que éste puede almacenar más de una tabla.

En el siguiente ejemplo, se creará una relación entre la tabla *publishers* y *titles* de la base de datos *pubs* de Microsoft SQL Server 2000.

```
'Declaro un adaptador del tipo SqlDataAdapter
Dim AdaptadorSQL As SqlDataAdapter

'Declaro el DataSet que contendrá las dos tablas
Dim DataSet1 As DataSet

'Creo el primer adaptador con la información de la tabla publishers
AdaptadorSQL = New SqlDataAdapter("Select pub_id," & _
   "pub_name,city, state,country from publishers", conexionSQL)

'Introduzco la primer tabla al DataSet previamente creado
AdaptadorSQL.Fill(DataSet1, "Editores")

 'Creo el segundo adaptador con la información de la tabla titles
AdaptadorSQL = New SqlDataAdapter("Select pub_id," & _
   "title, type, price from titles", conexionSQL)

'Introduzco la segunda tabla al DataSet previamente creado
 AdaptadorSQL.Fill(DataSet1, "Titulos")

'Primero le pongo el nombre a la relación, después indico cuales serán
'las columnas relacionadas.
 Dim relacion As New DataRelation("RelEditorialTitulos", _
 DataSet1.Tables("Editores").Columns("pub_id"), _
 DataSet1.Tables("Titulos").Columns("pub_id"))

 'Agrego la relación al Dataset
 DataSet1.Relations.Add(relacion)
```

3.18 Acceso a datos relacionados

El uso de las relaciones permite el acceso a los datos de tablas relacionadas. El método *GetChildRows* regresa un objeto *DataRow*, el cual se sincroniza con un valor seleccionado de la tabla primaria.

En el siguiente ejemplo, se obtendrán los valores de todos los registros que se encuentran relacionados con una editorial seleccionada. Previamente se estableció una relación entre la tabla *publishers* y *titles* de la base de datos *pubs* de Microsoft SQL Server 2000, como se mostró en el ejemplo anterior.

```
'Se declara el DataSet que contendrá las dos tablas
Dim DataSet1 As DataSet

'Declaro dos variables de tipo renglón
Dim PubRow, TitleRow As DataRow

'Declaro una variable de tipo arreglo que contendrá renglones
Dim TitleRows() As DataRow

'La variable PubRow  contendrá el renglón seleccionado
PubRow = DataSet1.Tables("Editores").Rows(0)

'Meto dentro de un arreglo de renglones, todos los
'renglones que son de la misma editorial
TitleRows = PubRow.GetChildRows("RelEditorialTitulos")

'Al ser una colección deben de accederse de la siguiente forma
For Each TitleRow In TitleRows
  ListBox1.Items.Add(TitleRow("title").ToString)
Next
```

Ejercicio 3.7

Objetivo:

Establecer una relación entre las tablas *Publishers* y *Titles*, de la base de datos *pubs*, de Microsoft SQL Server 2000. En un *DataGrid* se muestran todas las editoriales de la tabla *Publishers*, cuando se selecciona alguna de ellas se muestran todos los libros de la tabla *Titles*, que se encuentran relacionados con esa editorial, si no existe ningún libro relacionado no se muestra nada. En este ejercicio se utiliza el *espacio de nombres sqlClient*, recuerde que cada vez que se genera un evento, como por ejemplo un clic, la página es regenerada completamente, es por eso que se hace uso de la instrucción *IsPostBack*, que checa que las instrucciones que se encuentren dentro de ella solamente se ejecuten la primera vez que se cargue la página.

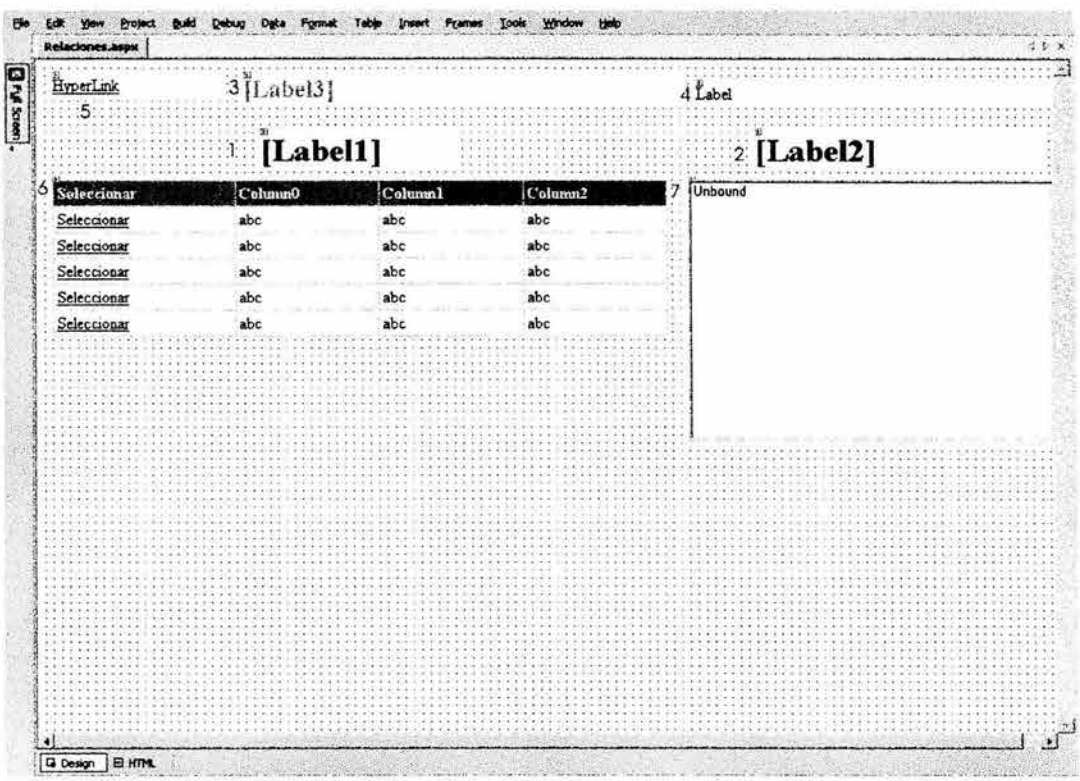
Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **Relaciones.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 3.36: 1)Label1, 2)Label2, 3)Label3, 4)Label4, 5)HyperLink1 6)DataGrid con ID=Editorial, 7)ListBox1.

Después de agregar el control *DataGrid* haga clic derecho sobre éste, seleccione *Auto Format* y por último *Professional 2,* como se muestra en la figura 3.37. Para poder seleccionar un renglón completo, es necesaria una columna extra que permita realizar esta operación. En propiedades seleccione *Columns* y haga clic en *(Collection)*.

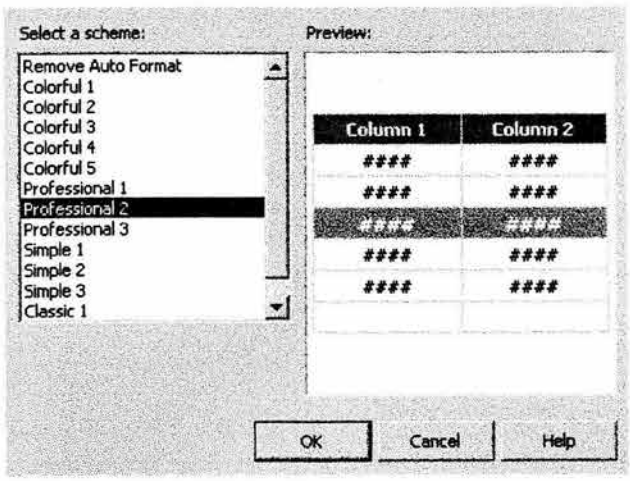
Figura 3.36: Controles de la forma Web Relaciones.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

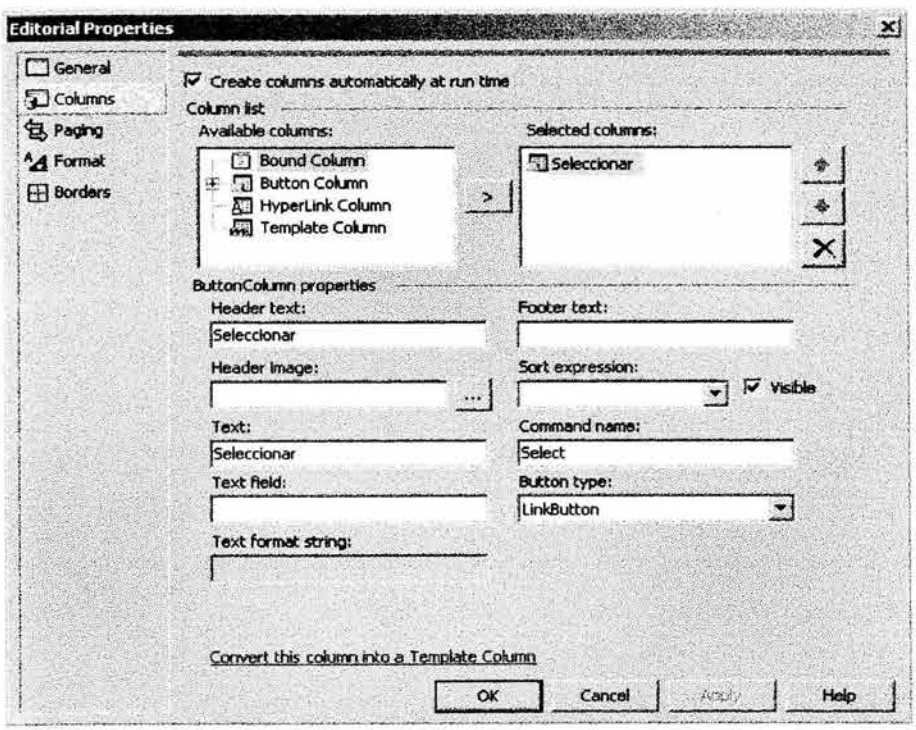
En propiedades (véase figura 3.38) haga clic en *Button Column* y agregue *Select*, después establezca el texto de cabecera, el *Command name* es el nombre con el cual será identificado en forma programática. Después de establecer estas propiedades oprima el botón *OK*.

Figura 3.37: Auto Format.



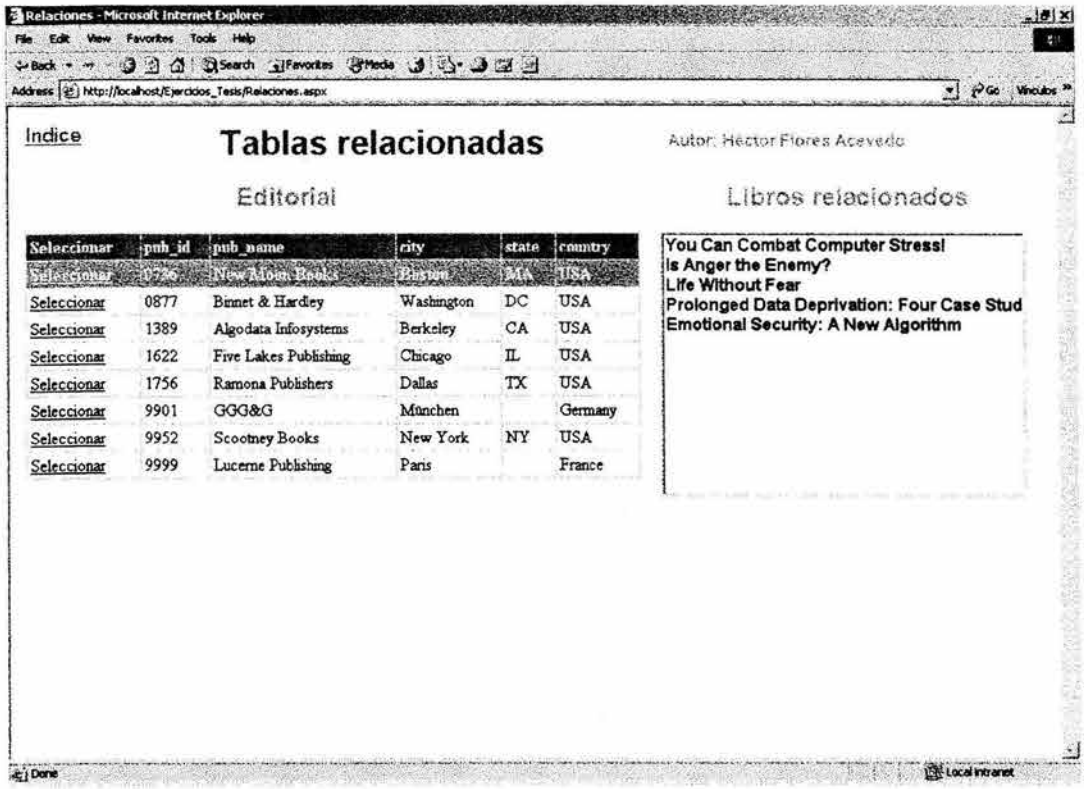
Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.38: Propiedad *Columns* de *Misc*.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 3.39: Resultado final de la forma Relaciones.aspx.



Seleccionar	pub_id	pub_name	city	state	country
Seleccionar	0736	New Moon Books	Boston	MA	USA
Seleccionar	0877	Binnet & Hardley	Washington	DC	USA
Seleccionar	1389	Algodata Infosystems	Berkeley	CA	USA
Seleccionar	1622	Five Lakes Publishing	Chicago	IL	USA
Seleccionar	1756	Ramona Publishers	Dallas	TX	USA
Seleccionar	9901	GGG&G	München		Germany
Seleccionar	9952	Scootney Books	New York	NY	USA
Seleccionar	9999	Lucerne Publishing	Paris		France

Fuente: Aplicación creada por el autor de esta tesis.

Escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Relaciones*.

Código de Relaciones.aspx

```
'Importo el espacio de nombres necesario
Imports System.Data.SqlClient

Public Class Relaciones
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro las propiedades de los controles a utilizar

        Label4.Text = "Autor: Héctor Flores Acevedo"
        Label4.Font.Bold = True
```

```
        Label4.Font.Size = FontUnit.Parse(12)
        Label4.ForeColor = Color.CadetBlue
        Label4.Font.Name = "Arial"

        Label1.Text = "Editorial"
        Label1.Font.Bold = True
        Label1.Font.Size = FontUnit.Parse(18)
        Label1.ForeColor = Color.CadetBlue
        Label1.Font.Name = "Arial"

        Label2.Text = "Libros relacionados"
        Label2.Font.Bold = True
        Label2.Font.Size = FontUnit.Parse(18)
        Label2.ForeColor = Color.CadetBlue
        Label2.Font.Name = "Arial"

        Label3.Text = "Tablas relacionadas"
        Label3.Font.Bold = True
        Label3.Font.Size = FontUnit.Parse(25)
        Label3.ForeColor = Color.Green
        Label3.Font.Name = "Arial"

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        ListBox1.Font.Size = FontUnit.Parse(12)
        ListBox1.Font.Name = "Arial"
        ListBox1.ForeColor = Color.DarkBlue
        ListBox1.Font.Bold = True

    End Sub

#End Region

    'Controles agregados, el WithEvents indica que tienen asociados eventos
    Protected WithEvents Editorial As System.Web.UI.WebControls.DataGrid
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents ListBox1 As System.Web.UI.WebControls.ListBox
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label

    'Variables generales que utilizare en la aplicación

    'Variable de conexión
    Dim conexionSQL As SqlConnection

    'Contiene la cadena de conexión a SQL Server
    Dim CadenaConexion As String

    'Los adaptadores son necesarios para un DataSet
    Dim AdaptadorSQL As SqlDataAdapter

    'El DataSet que contendrá las dos tablas
    Dim DataSet1 As DataSet

    'Declaro dos variables de tipo renglón
    Dim PubRow, TitleRow As DataRow

    'Arreglo que contiene renglones
    Dim TitleRows() As DataRow

    'Procedimiento que se genera cuando se carga la página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load

        'Cadena de conexión para la base de datos
        CadenaConexion = "Integrated Security=True;" & _
        "initial catalog=pubs; data source=(local)"

        'Creo una nueva conexión
        conexionSQL = New SqlConnection()
```

```
        'Asigno la cadena de conexión
        conexionSQL.ConnectionString = CadenaConexion

        'Abro la conexión
        conexionSQL.Open()

        'Creo el dataset
        DataSet1 = New DataSet()

        'Lleno el dataSet con 2 tablas, con Publishers y con Titles
        'Ambas tienen el campo pub_id relacionado

        'Creo el primer adaptador
        AdaptadorSQL = New SqlDataAdapter("Select pub_id," & _
                "pub_name,city, state,country from publishers", conexionSQL)

        AdaptadorSQL.Fill(DataSet1, "Editores")

        'Creo el segundo adaptador
        AdaptadorSQL = New SqlDataAdapter("Select pub_id," & _
        "title, type, price from titles", conexionSQL)
        AdaptadorSQL.Fill(DataSet1, "Titulos")

        'Establezco la relación, primero le pongo el nombre a la
        'relación, después pongo las columnas relacionadas
        Dim relacion As New DataRelation("RelEditorialTitulos", _
        DataSet1.Tables("Editores").Columns("pub_id"), _
        DataSet1.Tables("Titulos").Columns("pub_id"))

        'Agrego la relación al dataset
        DataSet1.Relations.Add(relacion)

        'Si es la primera vez que se carga la página entonces ejecuta el código
        If Not IsPostBack = True Then

            'Le asigno al DataGrid el valor de la primer tabla del DataSet
            Editorial.DataSource = DataSet1.Tables(0)

            'Asigno a la variable un renglón de la tabla seleccionada del DataSet
            PubRow = DataSet1.Tables("Editores").Rows(0)

            'Meto dentro de un arreglo de renglones, todos los
            'renglones que son de la misma editorial
            TitleRows = PubRow.GetChildRows("RelEditorialTitulos")

            'Al ser una colección se deben recuperar los datos así
            For Each TitleRow In TitleRows
                ListBox1.Items.Add(TitleRow("title").ToString)
            Next

            'Enlazo el grid editorial
            Editorial.DataBind()

            'El primer elemento lo selecciono
            Editorial.SelectedIndex = 0
        End If

    End Sub

    'Procedimiento que se genera cuando oprimo la primer columna del DataSet
    Private Sub Editorial_ItemCommand(ByVal source As Object, ByVal e _
    As System.Web.UI.WebControls.DataGridCommandEventArgs) Handles Editorial.ItemCommand

        'Checo si el tipo de comando es el de seleccionar
        If e.CommandName = "Select" Then

            'Se seleccionará el renglón seleccionado.
            Editorial.SelectedIndex = e.Item.ItemIndex

            'Asigno a la variable un renglón de la tabla seleccionada del DataSet
            PubRow = DataSet1.Tables("Editores").Rows(Editorial.SelectedIndex)

            'Meto dentro de un arreglo de renglones, todos los
            'renglones que son de la misma editorial
            TitleRows = PubRow.GetChildRows("RelEditorialTitulos")

            'Limpio el control
```

```
            ListBox1.Items.Clear()

            'Al ser una colección se deben recuperar los datos así
            For Each TitleRow In TitleRows
                ListBox1.Items.Add(TitleRow("title").ToString)
            Next
        End If
    End Sub
End Class
```

3.19 Uso de restricciones establecidas

En una base de datos generalmente se encuentran restricciones, que sirven para guardar la estructura y la integridad referencial de los datos.

ADO.NET permite guardar automáticamente las restricciones originales, aunque éstas se pueden configurar manualmente, se pueden evitar muchos problemas con solamente copiar la estructura de la base de datos.

El método ***FillSchema*** se encarga de copiar las restricciones en el DataSet, cuando se crea una relación entre tablas, las restricciones son agregadas a las columnas en forma automática.

FillSchema (Nombre del DataSet, schematype.Source, Nombre de la tabla)

En el siguiente bloque de código se muestra cómo guardar las restricciones de la tabla *titles* de la base de datos *pubs*, y después cargar la tabla en un *DataSet*.

```
'Creo un nuevo adaptador de tipo SqlDataAdapter
AdaptadorSQL = _
New SqlClient.SqlDataAdapter("Select * from titles", ConexionSQL)

'Lleno el DataSet con las restricciones de la tabla titles
AdaptadorSQL.FillSchema(DataSet1, SchemaType.Source, "Titulos")

'Lleno el DataSet con los datos de la tabla titles
AdaptadorSQL.Fill(DataSet1, "Titulos")
```

3.20 Conclusión del capítulo

ADO.NET es un modelo enfocado al desarrollo de aplicaciones en Internet y ambientes donde el número de usuarios tiende a crecer. Personalmente trabajé mucho tiempo con ADO, su sencillez hizo que fuera la mejor opción para mí en el acceso a bases de datos; sin embargo conforme se desarrollan aplicaciones más grandes, la

capacidad del servidor se encuentra cada vez más reducida con el aumento de usuarios.

El *DataSet* se comporta como una copia local de la base de datos, que puede almacenar varias tablas, así como sus relaciones; esto es de gran utilidad, ya que localmente el usuario puede tener acceso a una copia de la base de datos con todas sus reglas sin tener que estar conectado.

Pienso que el modelo ADO.NET tardará en ser asimilado por los desarrolladores, porque no es sencillo de aprender, y al principio se tienen muchos tropiezos, inclusive para el programador experimentado que solía trabajar con ADO.

Una aportación muy importante de éste modelo, es que puede transportar datos en formato XML (extensible markup language).

Ahora más que nunca es posible consumir y exportar datos a diferentes orígenes de datos, sin importar el sistema operativo sobre el que se trabaje, ya que XML es un estándar a nivel internacional en el transporte de datos.

Con el paso del tiempo el modelo ADO.NET será esencial para cualquier programador, actualmente Internet se hace más grande y más fuerte, en algunos años todo dispositivo estará conectado a Internet y la solución de Microsoft para el acceso de datos, ya está disponible para los desarrolladores.

Anteriormente muchas de las características implantadas en el nuevo modelo ADO.NET, debían ser simuladas por medio de colecciones de datos. Era bastante complejo hacer esto, porque se debía hacer una programación avanzada, y generalmente era difícil depurar las aplicaciones. Podría pensarse que ADO.NET tiene funciones y procedimientos sobrados, ya que con el modelo anterior, ADO, se pueden realizar muchas de las mismas tareas con menos esfuerzo. Parte de esto es cierto, ya que para aplicaciones pequeñas o que no requieran expandirse, ADO es una excelente opción, pero cuando ya se habla de aplicaciones muy grandes y no solamente eso, sino de aplicaciones que no sabemos el numero de usuarios que se conectarán a ella, entonces es necesario un modelo más confiable, que nos permita liberar al servidor de la sobrecarga de una infinidad de usuarios conectados a la base de datos, pero sin sacrificar el desempeño de las aplicaciones desarrolladas.

Creo que Microsoft se ha esmerado en dos áreas principales: una de ellas es ADO.NET y la otra son los Servicios Web XML. En varias publicaciones he leído que gran parte de los desarrolladores, que trabajan con herramientas de Microsoft, están

muy contentos con las ventajas que ofrece ADO.NET, una vez que se está acostumbrado a las funciones y procedimientos de este modelo, es muy fácil el desarrollo de aplicaciones.

Otro punto que me parece interesante y de gran avance, es que se puedan establecer relaciones, ahora es más fácil establecer la integridad referencial, se obtienen beneficios instantáneos con el hecho de que el *DataSet* pueda almacenar más de una tabla.

Es importante que el lector investigue otras tecnología para el acceso de datos, para que se dé cuenta de la riqueza de ADO.NET, desde mi punto de vista no hay mejor opción que ésta para el desarrollo de aplicaciones que sean utilizadas por una infinidad de usuarios.

En el ejercio 3.6 del tema Manipulación de DataSet, utilizo una vista para ordenar los datos que se encuentran dentro del DataSet, de tal forma que después de agregar un registro a la base de datos, me desconecto de ella y muestro todos los registros utilizando el DataSet por medio de una vista ordenada. Desde el punto de vista de usuario, esto no tiene mayor importancia, porque la mayoría de los sistemas de información hacen lo mismo, sin embargo, desde el punto de vista de programación por un lado libero al servidor de base de datos de hacer una nueva consulta para mostrar datos y por el otro, gestiono localmente la información manipulando a mi antojo el DataSet, ya sea para ordenar datos en forma ascendente, descendente ó simplemente hacer consultas a nivel local, sin tener que molestar al servidor cada vez que requiera una consulta.

De allí la hermosura de ésta tecnología, que está diseñada para trabajar con una cantidad muy grande de usuarios, evitando saturar el servidor de base de datos con solicitudes excesivas.

En este capítulo se han mencionado algunas de las características más importantes de ADO.NET, sin embargo hay que recalcar que existen infinidad de características más. El acceso a diferentes orígenes de datos se vuelve cada vez más importante, por lo que tener las herramientas más poderosas para el acceso a la información, es vital para los desarrolladores. Espero que éste capítulo le haya sido de gran utilidad.

CAPÍTULO IV

Servicios Web XML

4.1 ¿Qué es un Servicio Web XML?

La programación de componentes ha tenido un gran auge en los últimos años, los desarrolladores se han dado cuenta que es más sencillo darle mantenimiento a un paquete de código, que depurar toda la aplicación para hacer alguna modificación.

La carta fuerte de Microsoft hasta Visual Studio 6.0 era el modelo de componentes COM, que permite el uso de componentes de software en el equipo local o utilizando el modelo distribuido DCOM, que comparte un componente en una red local. Este modelo funciona bien en situaciones ideales en donde todos los equipos utilizan tecnología Microsoft; sin embargo, no se tenía una solución fiable para poder consumir procedimientos o módulos a través de Internet.

Se necesitaba un modelo que pudiera llamar procedimientos por medio del protocolo HTTP y que pudiera transferirse en un formato abierto como es XML, además de poder ser consumido por cualquier aplicación, independientemente del sistema operativo y del lenguaje de programación.

Entonces nació SOAP, protocolo simple de acceso a objetos, que es un mecanismo que permite llamar procedimientos remotos, utiliza HTTP como medio y XML como formato para los datos. Es por esto que a *un componente en la Web que utilice SOAP y que pueda invocar procedimientos y funciones en forma remota, se le llama Servicio Web XML.*

4.2 Funcionamiento en Visual Basic.NET

Un Servicio Web XML consta de funciones y procedimientos que pueden ser publicados en la Web. Visual Studio.NET crea un proyecto ASP.NET que permite que el Servicio Web XML pueda ser consumido por otras aplicaciones.

Cuando la página ASP.NET que contiene el Servicio Web XML recibe la invocación de una función o procedimiento, la busca y la ejecuta. El contenido de la solicitud se regresa a la aplicación en formato XML. Es importante hacer notar que los Servicios Web XML son consumidos por aplicaciones y no directamente por los usuarios.

La forma en que son referenciados los Servicios Web XML en Visual Studio.NET, es por medio de la opción *Agregar referencia Web*, en forma similar a la inserción de componentes en Visual Basic 6.0.

Los archivos que forman un Servicio Web XML, según Erich R. Bühler,[6] son:

- **References.** Carpeta que contiene las referencias a los diferentes ensamblados locales.
- **AssemblyInfo.vb.** Almacena información con respecto al proyecto actual (título, empresa, etc.).
- **Global.asax.** Contiene procedimientos asociados a eventos globales, de igual forma que una aplicación Web ASP.NET.
- **Web.config.** Almacena directivas a seguir para todos los módulos contenidos dentro del directorio del proyecto.
- **<Nombre del proyecto>.vsdisco.** Permite en forma dinámica descubrir los diferentes servicios Web ofrecidos por un sitio. Por defecto, éste inspecciona solamente la carpeta actual.
- ***.asmx.** Contiene la puerta de entrada Web para utilizar el servicio.
- ***.asmx.vb.** Contiene la lógica de los diferentes métodos del servicio. Este archivo será posteriormente compilado a MSIL.

4.3 Web Services Description Language (WSDL)

Cuando se tiene un Servicio Web XML, es necesario conocer las funciones y procedimientos de éste, así como sus parámetros.

En el modelo COM, es necesario que los componentes sean registrados en la computadora para que por medio del archivo de registro se puedan localizar. Debido a que los Servicios Web XML tienen la información dentro del ensamblado, no se registran en el archivo de registro.

WSDL o lenguaje para descripción de Servicio Web tiene toda la información para hacer uso del servicio. El formato de este documento puede ser leído desde

[6] BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 802.

cualquier aplicación y sistema operativo debido a que utiliza XML, que como lo mencionamos anteriormente es un formato abierto.

Visual Studio.NET genera automáticamente este documento, cada vez que se crea un Servicio Web XML.

La infraestructura de los servicios tiene las siguientes características:

- **Protocolos abiertos de Internet.** Protocolos tales como XML, HTTP, XML *Schema Definition* (XSD) y *Simple Mail Transfer Protocol*. Estos protocolos permiten la comunicación libre con el Servicio Web XML, sin importar el lenguaje de programación, ni el sistema operativo.
- **XML y SOAP.** SOAP es un protocolo que permite la transferencia de datos y comandos, el formato que utiliza es XML.
- **Web Services Description Language (WSDL).** Informa los métodos y argumentos del servicio Web disponibles para las aplicaciones cliente.

4.4 Funciones y subrutinas

El uso de funciones y procedimientos en los Servicios Web XML, son similares a los usados en ASP.NET; no obstante, los primeros deben de ser públicos y utilizar el atributo ***WebMethod.***

Ejemplo:[7]

```
<WebMethod()> Public Function AgregarUsuario(ByVal strNombre As _
String) As String
  'Función que agrega un usuario y regresa un numero de identificación
  .
  .
  Return strNuevoId
End Function

<WebMethod()> Public Sub BorrarUsuario(ByVal strId)
  'Funcionalidad que borra un usuario basado en su identificación
  .
  .
End Function
```

[7] MICROSOFT CORPORATION, ***Programming with Microsoft Visual Basic .NET***, Microsoft Corporation, Colombia, 2002, Módulo 7, p. 46.

Ejercicio 4.1:

Objetivo:

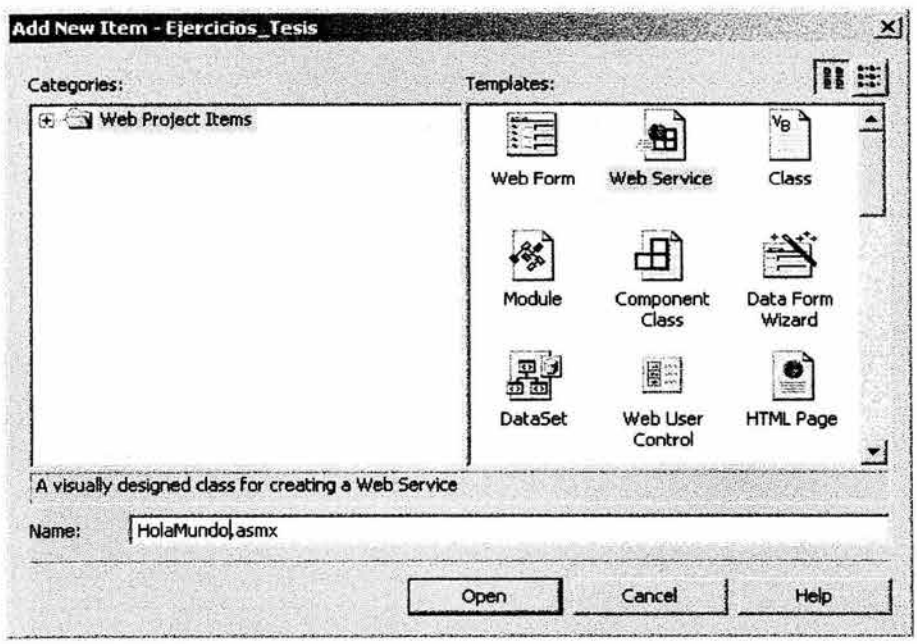
Crear un Servicio Web XML que muestre en formato XML el mensaje *"Hello World"*; este ejemplo viene por defecto dentro del código generado por el compilador, solamente que se encuentra entre apóstrofos para que no tenga efecto.

```
' WEB SERVICE EXAMPLE
' The HelloWorld() example service returns the string Hello World.
' To build, uncomment the following lines then save and build the project.
' To test this web service, ensure that the .asmx file is the start page
' and press F5.
'<WebMethod()> Public Function HelloWorld() As String
'     HelloWorld = "Hello World"
' End Function
```

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *Web Service* y escriba el nombre **HolaMundo.asmx**, que es el nombre que le he dado a la forma en este ejercicio, véase la figura 4.1.

Figura 4.1: Crear el Servicio Web XML HolaMundo.asmx.



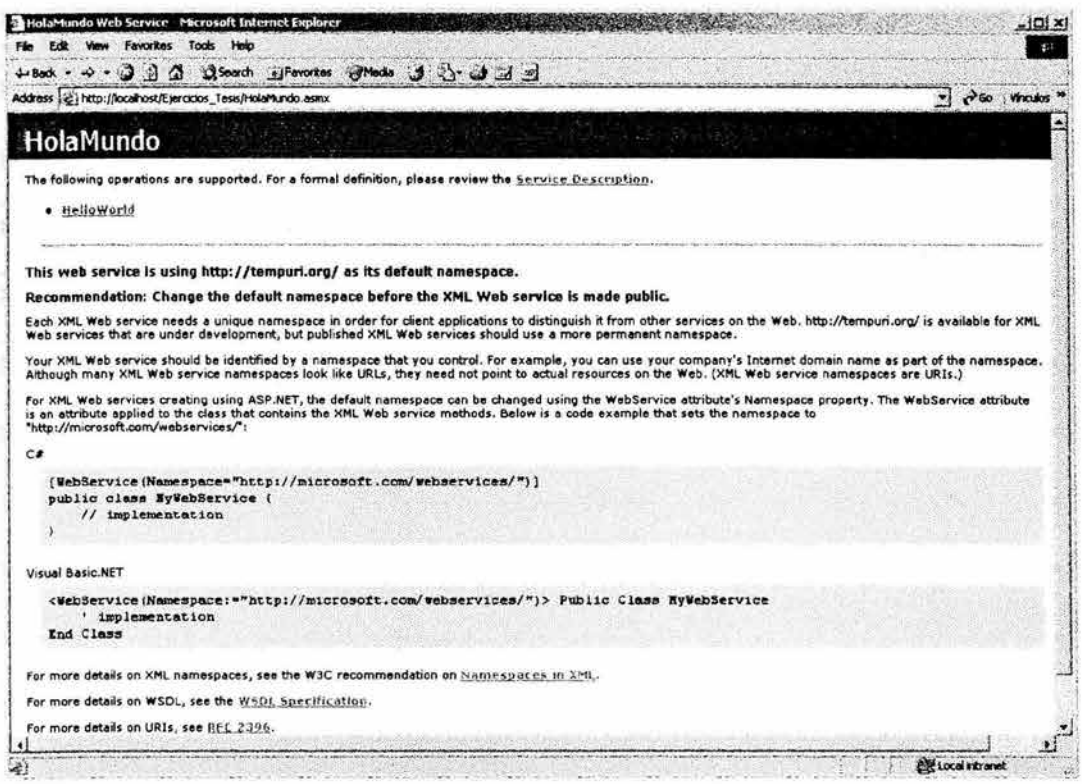
Fuente: Aplicación creada por el autor de esta tesis.

Después de haber creado el Servicio Web XML, se dará cuenta que en el panel *solution explorer*, se encuentra agregado el servicio. Haga doble clic sobre éste. Cuando aparezca la forma Web del servicio, haga clic sobre el hipervínculo que dice ***"click here to switch to code view"***. Quite los apóstrofos que se encuentran al principio de la función *HelloWorld*, de tal forma que el código quede de la siguiente forma:

```
' WEB SERVICE EXAMPLE
' The HelloWorld() example service returns the string Hello World.
' To build, uncomment the following lines then save and build the project.
' To test this web service, ensure that the .asmx file is the start page
' and press F5.
'
<WebMethod()> Public Function HelloWorld() As String
    HelloWorld = "Hello World"
End Function
```

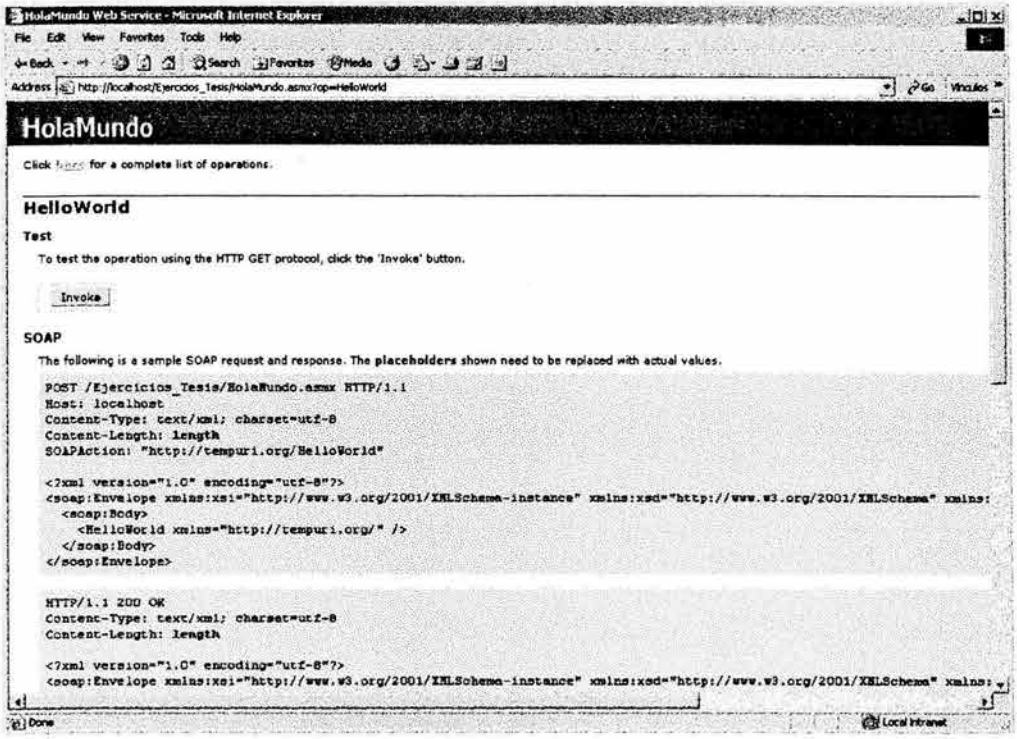
Grabe y oprima F5, cuando se encuentre en el menú de inicio oprima el hipervínculo *Servicio Web XML "HelloWorld"*. Automáticamente se genera una página que contiene las recomendaciones y características del servicio (véase figura 4.2).

Figura 4.2: Página de descripción del servicio.



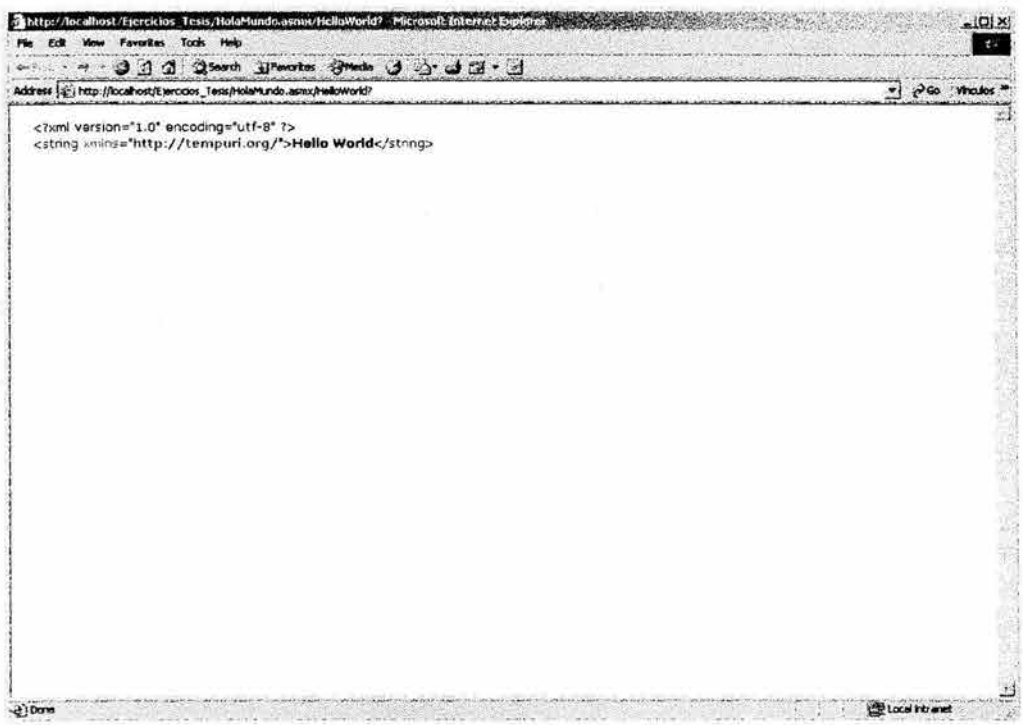
Fuente: Aplicación creada por el autor de esta tesis.

Figura 4.3: Página que invoca al servicio.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 4.4: Se muestra el mensaje *Hello World* en formato XML.



Fuente: Aplicación creada por el autor de esta tesis.

Al oprimir el hipervínculo *HelloWorld*, que es el nombre de la función dentro del Servicio Web XML, lo llevará a la siguiente página. Esta sirve para invocar al servicio (véase figura 4.3), en caso de necesitar introducir algún parámetro, como lo veremos en otro ejemplo, entonces se crearán las cajas de texto necesarias. Por último oprima el botón *Invoke*, el resultado de la invocación es un documento en formato XML que muestra el mensaje *Hello World*, el cual puede ser consumido por cualquier aplicación que soporte este formato.

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://tempuri.org/">Hello World</string>
```

Es recomendable que cada Servicio Web XML tenga su propia dirección para poder identificarse en la Web; generalmente una empresa que tiene su propio sitio, debe establecer un espacio para los Servicios Web XML con su propia dirección.

Erich R. Bühler[8] comenta: *"Debido a que el resultado final es un documento XML, el mismo podría ser consumido por cualquier aplicación o sistema operativo, ya sea utilizando SOAP, XML DOM, o simplemente procesando éste como un archivo de texto, en el caso de que no se cuente con el soporte de ninguna de las tecnologías citadas."*

Ejercicio 4.2:

Objetivo:

Crear un Servicio Web XML que ingrese a la base de datos *pubs* de Microsoft SQL Server 2000 y que obtenga un *DataSet* con el conjunto de libros que cumplan con el criterio de búsqueda.

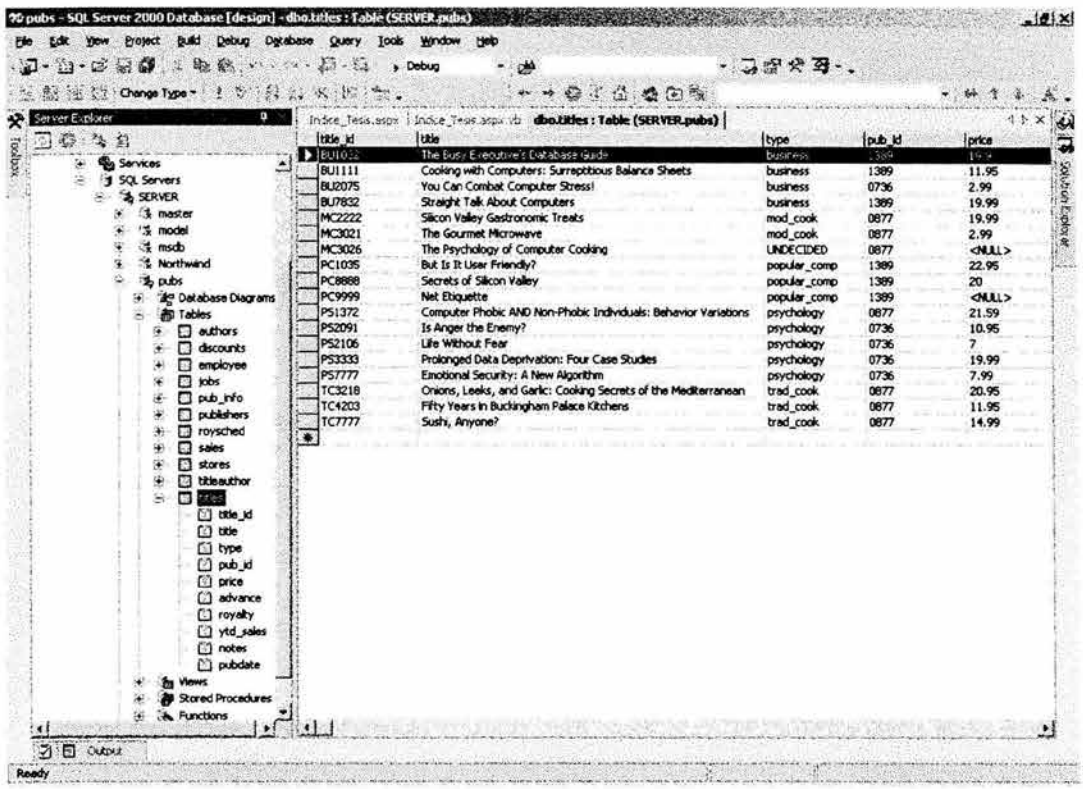
La aplicación mostrará en formato XML todos los libros encontrados, en caso de que existan. Recordemos que XML es el formato que manejan los Servicios Web XML para transportar datos.

Es importante que el servidor Microsoft SQL Server esté funcionando para que se pueda acceder. Dentro de Visual Studio .NET, oprima el menú *View* y *Server explorer*. Haga clic en los nodos hasta encontrar *SQL Servers* + *SERVER* + *pubs* +

[8] BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, p. 809.

Tables, por último haga doble clic en *titles*, entonces aparecen los datos almacenados en esta tabla, como se muestra en la figura 4.5.

Figura 4.5: Tabla *titles* de la base de datos *pubs*



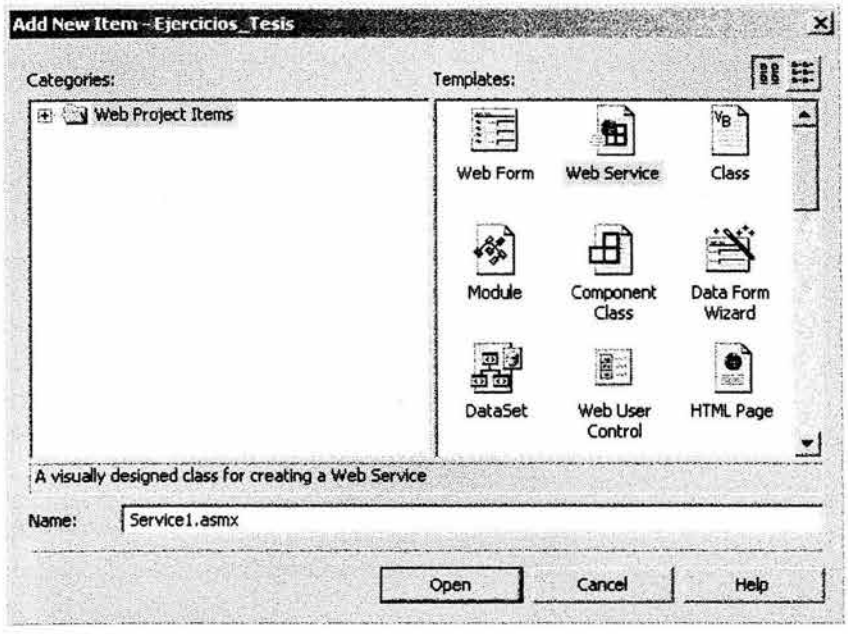
Fuente: Visual Studio.NET 2000.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add* + *Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *Web Service* y escriba el nombre **Service1.asmx**, que es el nombre que le he dado a la forma en este ejercicio, véase la figura 4.6.

Después de haber creado el Servicio Web XML, se dará cuenta que en el panel *solution explorer*, se encuentra agregado el servicio. Haga doble clic sobre éste. Cuando aparezca la forma Web del servicio, haga clic sobre el hipervínculo que dice ***"click here to switch to code view"***. Escriba el siguiente código donde sea requerido.

Figura 4.6: Crear el Servicio Web XML Service1.asmx.



Fuente: Aplicación creada por el autor de esta tesis.

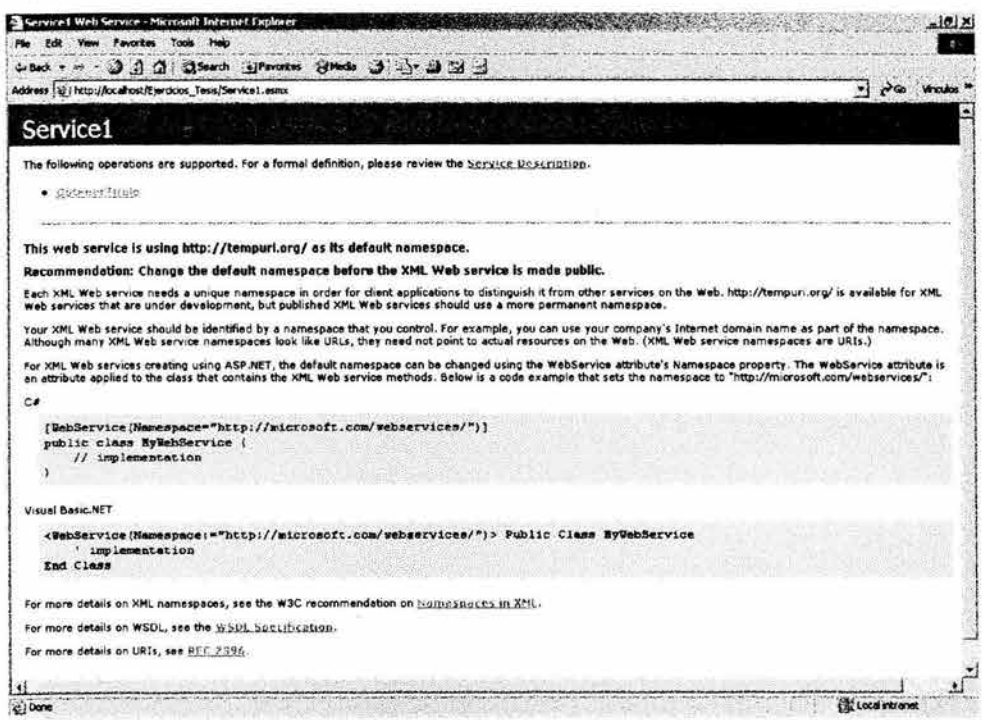
Código de Service1.asmx

```
Imports System.Web.Services
Imports System.Data.SqlClient


<WebService(Namespace:="http://tempuri.org/")> Public Class Service1
    Inherits System.Web.Services.WebService

#Region " Web Services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web Services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web Services Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Web Services Designer
    'It can be modified using the Web Services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web Services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
```

```
    End Sub
#End Region


    'Funcion que regresará el Dataset del TITULO especificado
    <WebMethod()> Public Function ObtenerTitulo(ByVal title_id As String) As DataSet
        Dim StrComando As String
        Dim conexionsql As New SqlConnection()

        conexionsql.ConnectionString = _
            "Integrated Security=True;Initial Catalog=pubs;Data Source=(local)"

        conexionsql.Open()

        StrComando = "SELECT * FROM titles WHERE title_id LIKE '" & title_id & "%'"

        Dim AdaptadorSQL As New SqlDataAdapter(StrComando, conexionsql)

        Dim DS As New DataSet()

        AdaptadorSQL.Fill(DS, "Titulos")
        Return (DS)

    End Function
End Class
```

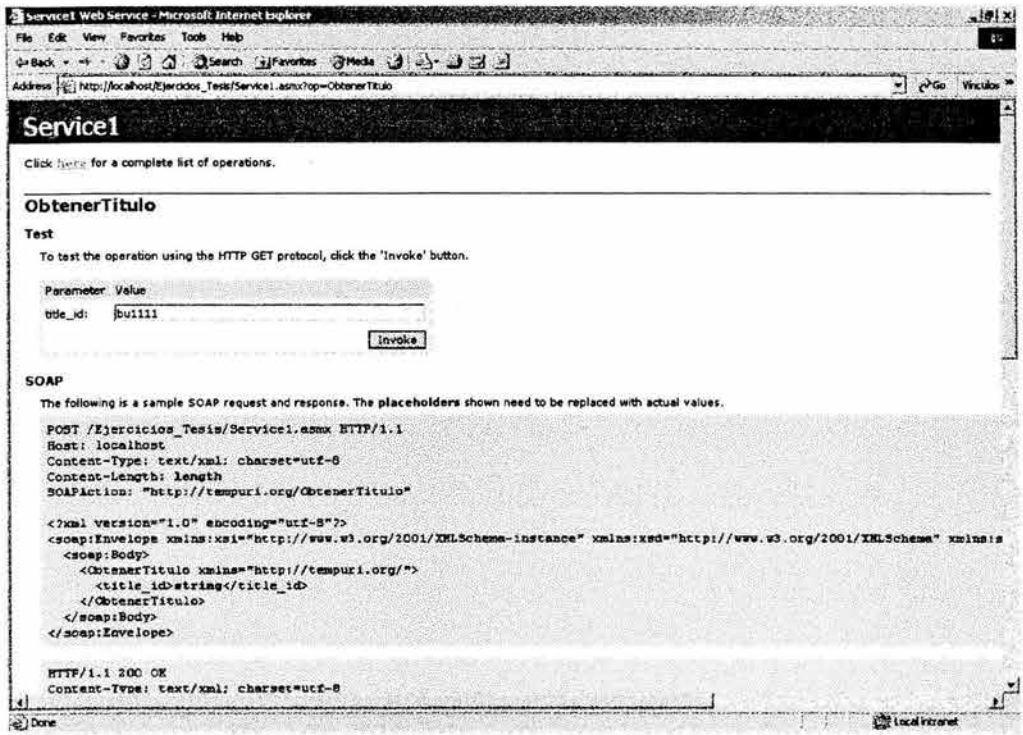
Grabe y oprima F5, cuando se encuentre en el menú de inicio oprima el hipervínculo *Función 'ObtenerTitulos' de un Servicio Web*. Automáticamente se genera una página que contiene las recomendaciones y características del servicio (véase figura 4.7).

Figura 4.7: Página de descripción del servicio.



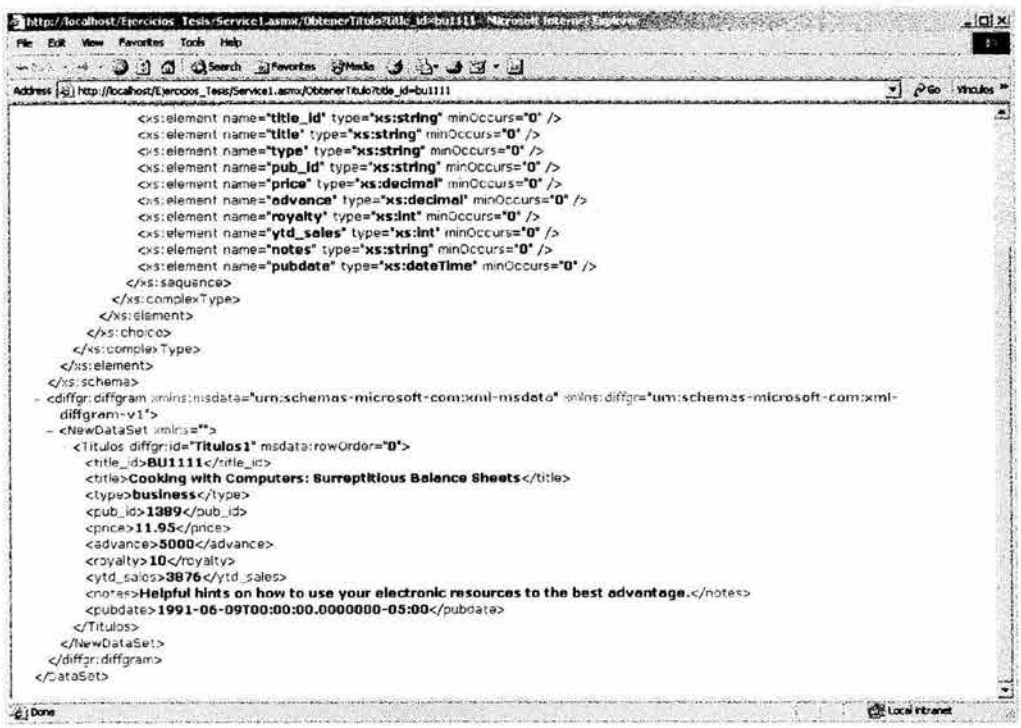
Fuente: Aplicación creada por el autor de esta tesis.

Figura 4.8: Página que invoca al servicio.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 4.9: Se muestra el resultado de la búsqueda en formato XML.



Fuente: Aplicación creada por el autor de esta tesis.

Al oprimir el hipervínculo *ObtenerTitulo*, que es el nombre de la función dentro del Servicio Web XML, lo lleva a la página que sirve para invocar al servicio (véase figura 4.8), debido a que se necesita introducir un parámetro, se creó una caja de texto. Escriba *BU1111* para que el servicio regrese todos los libros que cumplen con el criterio en el campo *title_id*, como se muestra en la figura 4.9.

Debido a que se utilizó la instrucción ***like%,*** se puede regresar como resultado de uno a muchos libros que cumplan con el criterio.

```
StrComando= "SELECT * FROM titles WHERE title_id LIKE '" & title_id & "%'"
```

Por último oprima el botón *Invoke*, el resultado de la invocación es un documento en formato XML que regresa todos los libros que cumplan con el criterio de la búsqueda.

4.5 Formas de invocar un Servicio Web XML

- **Método GET**. Permite invocar al servicio por medio del llamado URL extendido, que consiste en anexar los parámetros en la dirección de Internet, la desventaja de este método es que no puede transportar objetos, como *DataSet*. Ejemplo: http://localhost/ServicioWeb1/ObtenerInfo?Parametro=125487
- **Método POST**. La información no se hará visible en la línea de URL, ya que viajará escondida dentro del paquete; sin embargo, tampoco puede invocar objetos como el *DataSet*, ni tipos de datos complejos.
- **SOAP**. Este protocolo de mensajes puede invocar tipos de datos complejos, lo que da mucha flexibilidad, ya que no existen limitaciones como con los otros métodos.

4.6 Consumo de un Servicio Web XML

Los Servicios Web XML tienen la propiedad de poder ser consumidos en forma remota; esto significa que cualquier aplicación escrita en Java, Pascal, VB, C++, etc., que soporte la tecnología .NET puede hacer uso de los procedimientos y funciones que se encuentren en un servidor remoto que ofrezca estos servicios.

Solamente basta con agregar una referencia, para poderlo utilizar como si estuviera en la máquina local, de tal forma que una aplicación podría estar utilizando varios Servicios Web XML y el usuario final no se daría cuenta. Para que puedan ser accedidos, se debe de contar con una conexión a Internet, si es que los servicios se encuentran en otro servidor. Varios Servicios Web XML se encuentran disponibles en Internet, algunos de éstos se ofrecen en forma gratuita, otros requieren de un pago por su uso, y algunos otros están restringidos para usuarios autorizados.

4.7 Seguridad

Cuando se agrega una referencia Web en Visual Studio.NET, se debe crear una instancia a la clase del Servicio Web XML. Este servicio tiene la propiedad *credential*, que es usada para obtener o establecer la autentificación de un cliente para un servicio Web.

Cuando se crea una instancia del servicio, se deben de poner las credenciales específicas para el mecanismo de autentificación, antes de hacer la llamada al método del Servicio Web XML. Debido a que el acceso anónimo se deshabilitó cuando se hizo la solicitud del servicio, se deben de establecer las credenciales de acceso.

Por defecto, las llamadas a los servicios Web no heredan los permisos de acceso del cliente que los invoca. En ese caso es probable que se genere el error *"401, Access Denied"*.

Para establecer los permisos necesarios para acceder a los servicios, se escribe el siguiente código en la aplicación que invoca al Servicio Web XML:

```
MiServicioWeb.Credentials = System.Net.CredentialCache.DefaultCredentials
```

Default Credentials representa los permisos del sistema de seguridad actual en el que la aplicación está funcionando. En una aplicación cliente, usualmente son los permisos de Windows (nombre de usuario, password y dominio). En el caso de aplicaciones ASP.NET, son los permisos del usuario identificado en el proceso ASP, o el usuario en el modo de *impersonation*. En este caso *DefaultCredentials* representa la cuenta de usuario.

Ejercicio 4.3:

Objetivo

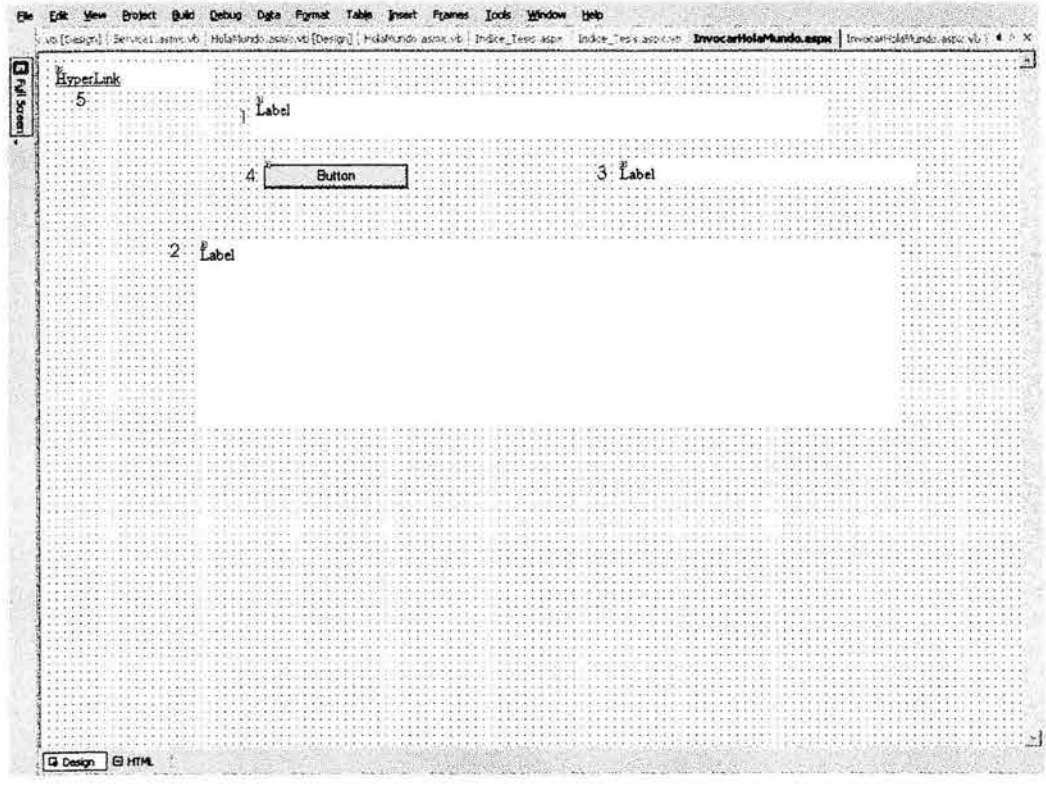
Invocar la función *HelloWorld* del Servicio Web XML previamente creado. Cuando el usuario oprima un botón en la forma Web, se desplegará en la pantalla el mensaje "Hello World".

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add* + *Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **InvocarHolaMundo.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 4.10: 1)Label1, 2)Label2, 3)Label3, 4)Button1, 5)HyperLink1.

Figura 4.10: Controles de la forma Web InvocarHolaMundo.aspx.

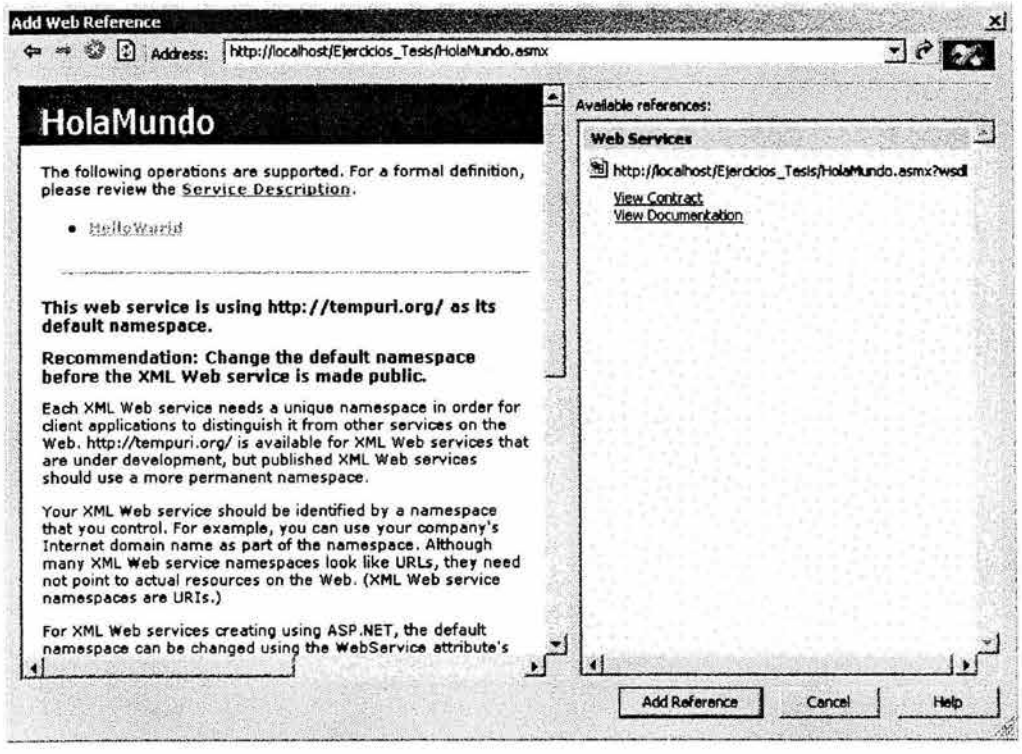


Fuente: Aplicación creada por el autor de esta tesis.

Para poder utilizar un Servicio Web XML, debe hacer referencia a éste, por lo tanto vaya al menú, oprima *Project + Add Web Reference*. Aparecerá una forma que le pide el URL del servicio, escriba ***http://localhost/Ejercicios_Tesis/HolaMundo.asmx***. Haga clic en el botón con la flecha que se encuentra a continuación. Por último oprima *Add Reference* (véase figura 4.11).

Diríjase al explorador de soluciones, y como se puede dar cuenta se agregó un nuevo nodo llamado *localhost*, dentro de la carpeta *Web references*. Haga clic con el botón derecho sobre este nodo, seleccione *rename* y escriba **ServicioHolaMundo**, puede escribir cualquier otro nombre, excepto el que da por defecto la aplicación.

Figura 4.11: Agregando una referencia a un Servicio Web XML.



Fuente: Aplicación creada por el autor de esta tesis.

Escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Invocar función 'Hello World'*, el resultado final se muestra en la figura 4.12.

Figura 4.12: Resultado final después de invocar al servicio Web XML.

Fuente: Aplicación creada por el autor de esta tesis.

Código de InvocarHolaMundo.aspx

```
Public Class InvocarHolaMundo
    Inherits System.Web.UI.Page
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro cada propiedad de los controles a utilizar

        Label3.Text = "Autor: Héctor Flores Acevedo"
        Label3.Font.Bold = True
        Label3.Font.Size = FontUnit.Parse(12)
        Label3.ForeColor = Color.CadetBlue
        Label3.Font.Name = "Arial"

        HyperLink1.Font.Size = FontUnit.Parse(14)
```

```
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Label1.Text = "Invocación del Servicio Web XML 'Hola Mundo'"
        Label1.Font.Name = "Arial"
        Label1.ForeColor = Color.Green
        Label1.Font.Size = FontUnit.Parse(18)
        Label1.Font.Bold = True

        Label2.Text = String.Empty
        Label2.Font.Name = "Arial"
        Label2.ForeColor = Color.DarkBlue
        Label2.Font.Size = FontUnit.Parse(75)
        Label2.Font.Bold = True

        Button1.Text = "Invocar"
        Button1.Font.Name = "Arial"
        Button1.ForeColor = Color.White
        Button1.BackColor = Color.CadetBlue
        Button1.Font.Size = FontUnit.Parse(15)
        Button1.Font.Bold = True

    End Sub

#End Region

    'Procedimiento que se genera cuando se carga la página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load
        'Put user code to initialize the page here
    End Sub
    'Procedimiento que se genera cuando se oprime el botón
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button1.Click

        'Se crea el objeto de tipo ServicioHolaMundo
        Dim ServicioWeb As New ServicioHolaMundo.HolaMundo()

        'Excepción que permite que el código se ejecute en forma confiable
        Try
            'Permite otorgar el permiso para acceder al Servicio Web XML
            ServicioWeb.Credentials = _
            System.Net.CredentialCache.DefaultCredentials

            'Mando llamar la función HelloWorld
            Label2.Text = ServicioWeb.HelloWorld

        Catch valor As Exception
            'En caso de que no se encuentre el servicio Web XML
            Label2.Font.Size = FontUnit.Parse(12)
            Label2.Text = "El Servicio Web XML no se encuentra"
            'Se termina la excepción
        End Try
    End Sub
End Class
```

Ejercicio 4.4:

Objetivo

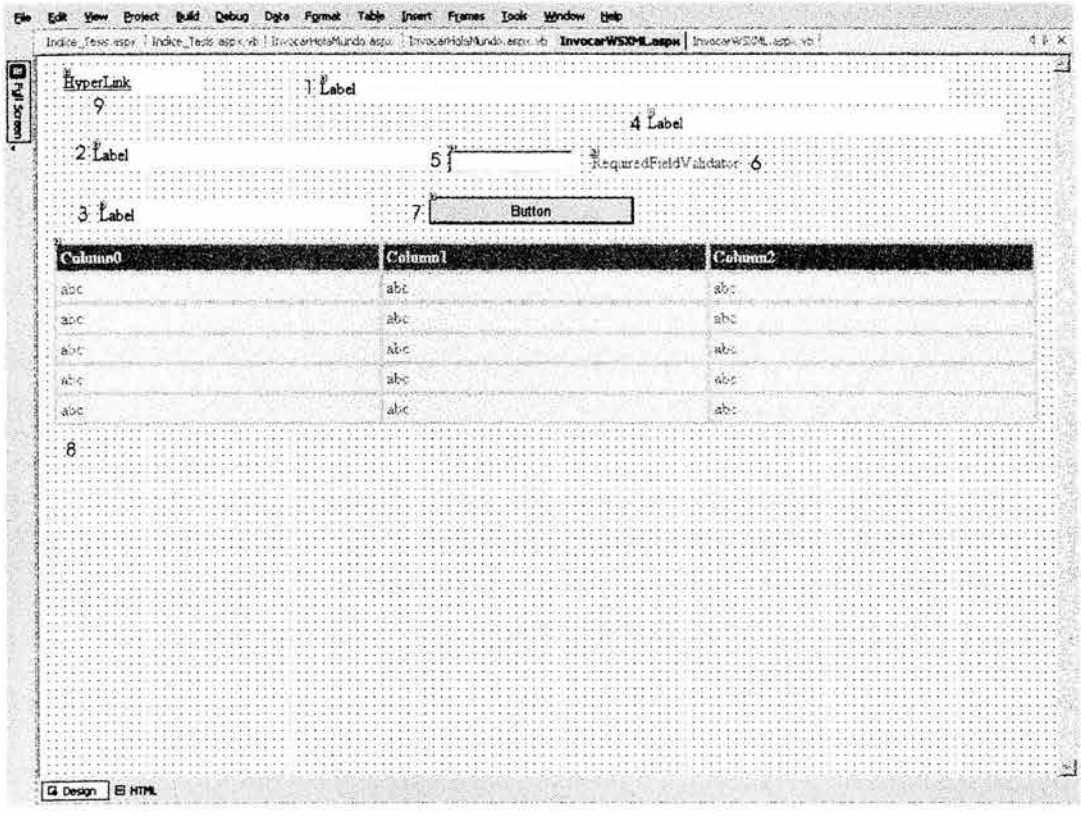
Invocar la función *ObtenerTitulo* del Servicio Web XML previamente creado. Después que el usuario escriba todo o parte del código de un libro y oprima el botón con el texto *"Oprime aquí"*, se desplegará un *DataGrid* con todos los títulos de los libros que cumplen con el criterio de la búsqueda.

Procedimiento:

Abra el proyecto *Ejercicios_Tesis* que se creó en el primer capítulo, sitúese en *solution explorer*, haga clic derecho sobre el nombre del proyecto, seleccione *Add + Add New Item*, entonces se abre una forma que le pide que seleccione el nuevo objeto. Seleccione *WebForm* y escriba el nombre **InvocarWSXML.aspx**, que es el nombre que le he dado a la forma en este ejercicio.

Agregue los siguientes controles ordenados de acuerdo a la figura 4.13: 1)Label1, 2)Label2, 3)Label3, 4)Label4, 5)TextBox1, 6)RequiredFieldValidator1, 7)Button1, 8)DataGrid1, 9)HyperLink1.

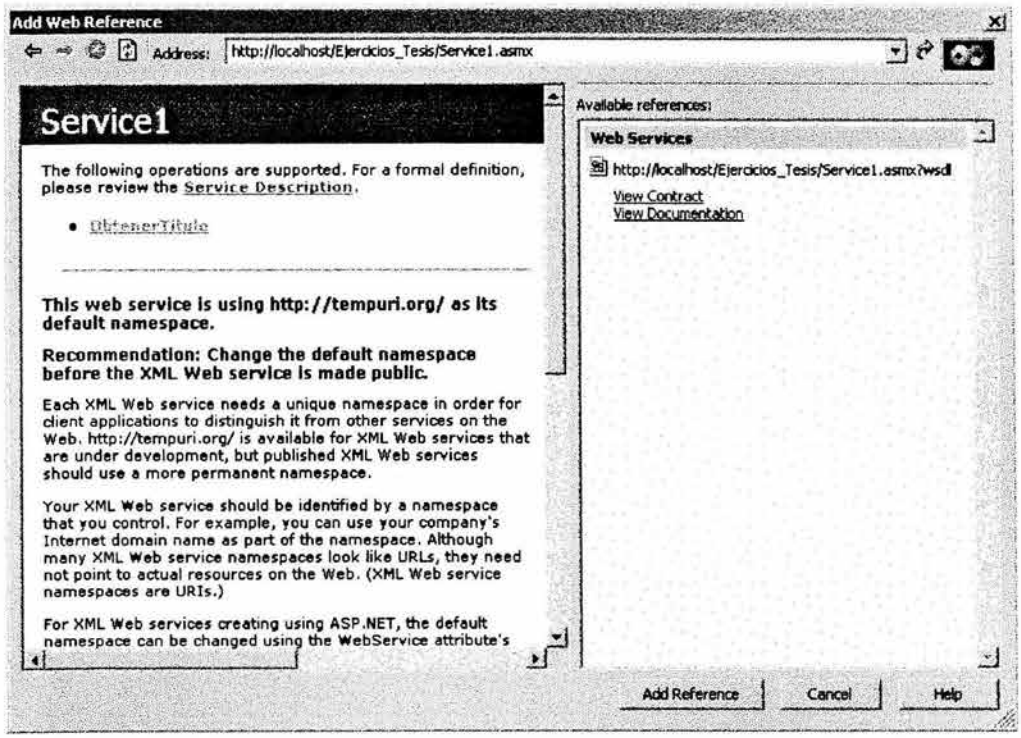
Figura 4.13: Controles de la forma Web InvocarWSXML.aspx.



Fuente: Aplicación creada por el autor de esta tesis.

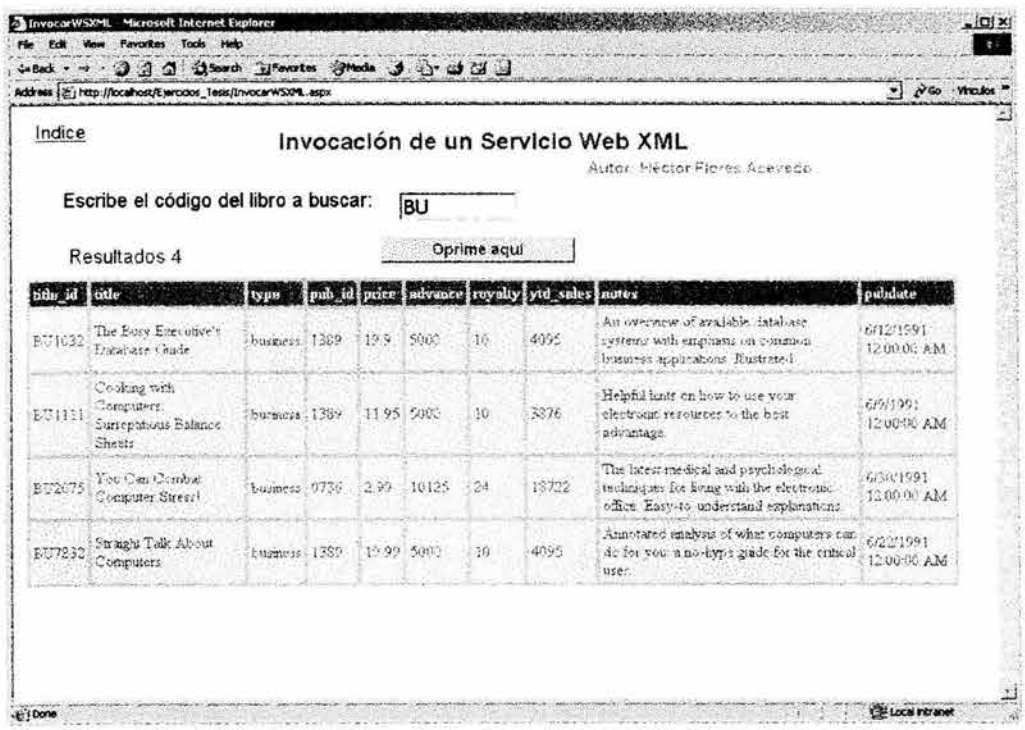
Para poder utilizar un Servicio Web XML, debe hacer referencia a éste, por lo tanto vaya al menú, oprima *Project + Add Web Reference*. Aparecerá una forma que le pide el URL del servicio, escriba ***http://localhost/Ejercicios_Tesis/Service1.asmx***.

Figura 4.14: Agregando una referencia a un Servicio Web XML.



Fuente: Aplicación creada por el autor de esta tesis.

Figura 4.15: Resultado final después de invocar al servicio Web XML.



Fuente: Aplicación creada por el autor de esta tesis.

Haga clic en el botón con la flecha que se encuentra a continuación. Por último oprima *Add Reference* (véase figura 4.14).

Diríjase al explorador de soluciones, y como se puede dar cuenta se agregó un nuevo nodo llamado *localhost*, dentro de la carpeta *Web references*. Haga clic con el botón derecho sobre este nodo, seleccione *rename* y escriba **ServicioWeb**, puede escribir cualquier otro nombre, excepto el que da por defecto la aplicación.

Escriba el siguiente código, al terminar ejecute la aplicación oprimiendo *F5*, cuando se encuentre en el menú de inicio, oprima *Invocar función 'ObtenerTitulo'*, el resultado final se muestra en la figura 4.15.

Código de InvocarWSXML.aspx

```
Public Class InvocarWSXML
    Inherits System.Web.UI.Page
    Protected WithEvents DataGrid1 As System.Web.UI.WebControls.DataGrid
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents RequiredFieldValidator1 As _
                    System.Web.UI.WebControls.RequiredFieldValidator
    Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        'Configuro cada propiedad de los controles a utilizar

        Label4.Text = "Autor: Héctor Flores Acevedo"
        Label4.Font.Bold = True
        Label4.Font.Size = FontUnit.Parse(12)
        Label4.ForeColor = Color.CadetBlue
        Label4.Font.Name = "Arial"

        HyperLink1.Font.Size = FontUnit.Parse(14)
        HyperLink1.Font.Name = "Arial"
        HyperLink1.Text = "Indice"
        HyperLink1.NavigateUrl = "http://localhost/Ejercicios_Tesis/Indice_Tesis.aspx"
        HyperLink1.ForeColor = Color.Red

        Label1.Text = "Invocación de un Servicio Web XML"
        Label1.Font.Name = "Arial"
        Label1.Font.Size = FontUnit.Parse(18)
        Label1.Font.Bold = True
        Label1.ForeColor = Color.Green

        Label2.Text = "Escribe el código del libro a buscar: "
        Label2.Font.Name = "Arial"
        Label2.Font.Size = FontUnit.Parse(15)
        Label2.ForeColor = Color.Blue
```

```
        Button1.BackColor = Color.AntiqueWhite
        Button1.Font.Name = "Arial"
        Button1.Text = "Oprime aquí"
        Button1.Font.Bold = True
        Button1.ForeColor = Color.Brown
        Button1.Font.Size = FontUnit.Parse(12)

        Label3.Text = "Resultados "
        Label3.Font.Name = "Arial"
        Label3.Font.Size = FontUnit.Parse(15)
        Label3.ForeColor = Color.Red

        TextBox1.Text = String.Empty
        TextBox1.Font.Name = "Arial"
        TextBox1.Font.Size = FontUnit.Parse(15)
        TextBox1.ForeColor = Color.Black

        'Valores para configurar el control de validación
        RequiredFieldValidator1.ControlToValidate() = "TextBox1"
        RequiredFieldValidator1.ErrorMessage = _
                    "Favor de poner un valor en la caja de texto"
        RequiredFieldValidator1.Display = ValidatorDisplay.Static
        RequiredFieldValidator1.Font.Name = "Arial"
        RequiredFieldValidator1.Font.Size = FontUnit.Parse(12)

        'Esta propiedad permite que se pueda validar la caja de texto en forma
        'programática, antes que se despliegue el mensaje de error
        RequiredFieldValidator1.EnableClientScript = False

    End Sub

#End Region

    'Procedimiento que se genera cuando se carga la página
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles MyBase.Load

        'No permito que se vea el DataGrid, ni el Label3.
        DataGrid1.Visible = False
        Label3.Visible = False

    End Sub

    'Este procedimiento se ejecuta cuando se oprime el botón
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button1.Click

        'Valido la caja de texto
        If RequiredFieldValidator1.IsValid Then

            'se crea el objeto de tipo ServicioWeb
            Dim ServicioTitulos As New ServicioWeb.Service1()

            'Declaro una variable de tipo String como auxiliar
            Dim TituloID As String

            'El valor de la caja de texto se lo asigno a la variable TituloID
            TituloID = TextBox1.Text

            'Permito que se vea el DataGrid y el Label3
            DataGrid1.Visible = True
            Label3.Visible = True

            'Excepción que permite que el código se realice en forma confiable
            Try

                'Permite otorgar el permiso para acceder al Servicio Web XML
                ServicioTitulos.Credentials = _
                System.Net.CredentialCache.DefaultCredentials

                'Se invoca el servicio por medio de una función
                DataGrid1.DataSource = ServicioTitulos.ObtenerTitulo(TituloID)

                'Se enlaza con el DataGrid
                DataGrid1.DataBind()

                'Escribo el número de resultados
```

```
                Label3.Text = "Resultados " + Str(DataGrid1.Items.Count())

            Catch Exc As Exception

                'En caso de que exista un error no permito que se vea el Grid
                'y despliego la etiqueta de error.
                Label3.Text = "Error" + Exc.Message
                'Termina la excepción
            End Try
        Else
            'En caso de que no se haya escrito nada en la caja de texto no
            'permito que se vea el DataGrid, ni el Label3.
            DataGrid1.Visible = False
            Label3.Visible = False
        End If
    End Sub

End Class
```

4.8 Impersonation

El sitio de ayuda de Microsoft,[9] así como la ayuda de Microsoft Visual Studio.NET, define *impersonation* de la siguiente manera:

Impersonation es cuando ASP.NET ejecuta código de un usuario autentificado. En IIS 6 la identidad por defecto es *NetworkService*, sin embargo en ASP se utiliza por defecto *impersonation*. Ésta puede ejecutar aplicaciones ASP.NET con la identidad del cliente que está operando.

En el archivo Web.config se controla el modo *impersonation*. Se tienen las siguientes opciones:

- ***Impersonation* habilitada**. Se permite la ejecución de una aplicación ASP.NET a un usuario anónimo de Internet o a un usuario autentificado.

  <identity impersonate="true" />

- ***Impersonation* habilitada y previamente configurada**. En este caso se utiliza la identidad configurada previamente en Web.config.

  <identity impersonate="true" name="domain\user" password="password" />

- ***Impersonation* deshabilitada**. Esta es la configuración por defecto. Por cuestiones de compatibilidad es recomendable escribir la etiqueta *impersonate*, en este caso se pone la configuración por defecto de ASP, la cual tiene el valor de falso.

[9] http://msdn.microsoft.com/library/default.asp

<identity impersonate="false" />

Hay que ser cuidadosos al usar *impersonation*, ya que es posible que una aplicación procese código utilizando permisos que no fueron anticipados por el diseñador de aplicaciones. Por ejemplo, si su aplicación utiliza *impersonate* con un usuario autentificado de una Intranet, podría tener privilegios extras que usted no desea que tenga. Por otro lado, si un usuario está más restringido de lo esperado, el usuario no podría utilizar la aplicación.

Impersonation se utiliza también para acceder a una página que realice consultas a una base de datos, ya que se pueden restringir o dar libertades a los usuarios. Para mayor información consulte la ayuda de Visual Studio.NET.

4.9 Características específicas de los Servicios Web XML

- **Propiedad URL.** Cuando se hace referencia a un Servicio Web XML, se debe de especificar una dirección para que se pueda acceder; sin embargo, cuando se está en la etapa de desarrollo no siempre será la misma dirección de la empresa en donde se ponga a funcionar la aplicación. Es por eso que la instancia del Servicio Web XML tiene un método llamado URL, que permite establecer en tiempo de ejecución la dirección del Servicio Web XML.

```
MiServicioWeb.Url = http://Direccion/ServicioWeb.asmx
```

- **Propiedad Timeout.** Es posible establecer el tiempo máximo de espera de un Servicio Web XML, en una aplicación al que se haga referencia, la propiedad *Timeout* recibe un valor en milisegundos.

```
MiServicioWeb.Timeout=2000
```

- **Buffer response.** En la programación del Servicio Web XML existe una etiqueta llamada **<WebMethod>** que se encuentra antes del nombre de la funciones del Servicio Web XML. Dentro de ella se puede establecer la leyenda *(BufferResponse)*, que permite que la información solicitada se mande en un solo bloque, si tiene un valor verdadero; esto significa que se mandará hasta que toda la información esté completa. Es importante mencionar que esta técnica se debe utilizar en servidores con memoria suficiente.

```
<WebMethod(BufferResponse:=True)> Function MiFuncionWeb(parámetros) as _
Valor de Retorno
```

- **Cache Duration.** Dentro de la etiqueta **<WebMethod>** se puede establecer la leyenda *(CacheDuration)*, que permite almacenar por un número de segundos establecidos por el desarrollador los resultados de solicitudes previas, evitando una sobrecarga al servidor en caso de que se vuelvan a solicitar los mismos resultados. El valor por defecto es de cero.

```
<WebMethod(CacheDuration:=50)> Function MiFuncionWeb(parámetros) as _
Valor de Retorno
```

- **Description.** Es posible dar una descripción de cada función que se escriba en el Servicio Web XML, para que cuando se haga referencia a ella, el desarrollador pueda saber de qué se trata.

```
<WebMethod(Description:="Esta función hace varias cosas")> _
 Function MiFuncionWeb(parámetros) as Valor de Retorno
```

- **Namespace.** Como lo mencioné anteriormente, cada Servicio Web XML debe de tener su propio *espacio de nombres* para que pueda ser identificado de forma única; la mejor forma de hacer esto es asignándole una dirección de Internet a la leyenda *Namesapce*, dentro de la etiqueta *WebService*, de la siguiente forma:

```
<WebService(Namespace:="http://Direccion única de Internet")>
```

- **Enable Session.** En capítulos anteriores explicamos lo que eran las variables de sesión, las cuales nos permiten guardar ciertos valores o preferencias del cliente mientras dure la sesión, en los Servicios Web XML es posible utilizarlas, es recomendable almacenar solamente la información que se recibe de los parámetros, no información extra, ya que puede dificultar la estabilidad de la aplicación.

```
<WebMethod(EnabledSession:=True)> Function MiFuncionWeb(parámetros) as _
Valor de Retorno
```

4.10 Documentos de descubrimiento

Los Servicios Web XML necesitan informar de su existencia a los desarrolladores, los servicios que ofrecen y la forma correcta de interactuar con ellos.

Para poder encontrar un Servicio Web XML es necesario que éste publique un documento de descubrimiento; el formato en que está escrito es XML, de tal forma que cualquier aplicación cliente, de cualquier sistema operativo y lenguaje, pueda interpretarlo. Es posible crear un documento de descubrimiento, que busque el

Servicio Web XML en un determinado directorio virtual, cuando una aplicación cliente lo solicite.

Para obtener información de un determinado Servicio Web XML, el servidor Web de Microsoft, Internet Information Server, se dirige hacia un directorio preestablecido. El documento de descubrimiento puede ser generado en forma manual, por medio de un editor de XML o en forma automática Visual Studio.NET lo genera cuando se crea un Servicio Web XML, utiliza la extensión de archivo *vsdisco*.

4.11 UDDI

La industria de software tuvo la necesidad de crear un directorio, que pudiera localizar los diferentes Servicios Web XML ofrecidos a nivel mundial; entonces nace Universal Discovery Description and Integration, que sus siglas son UDDI.

Actualmente se encuentra en su primera etapa, sin embargo se espera que en el futuro puedan existir buscadores en Internet de Servicios Web XML, de manera que los desarrolladores puedan buscar los servicios tal y como lo hacen con los sitios Web. En Visual Studio.NET se encuentra incluido un UDDI de Microsoft, que permite consultar algunos servicios disponibles en Internet.

4.12 Conclusión del capítulo

En este capítulo se han dado las bases para la construcción y utilización de los Servicios Web XML; actualmente se encuentran en su primera etapa. Un Servicio Web XML se puede ver como un paquete de código independiente que un servidor publica, para que otras aplicaciones puedan hacer uso de él a través de Internet.

La mayor ventaja de estos servicios es que utilizan SOAP (Simple Object Access Protocol) y como formato XML (extensible markup language), que es abierto y puede ser interpretado por cualquier aplicación, bajo cualquier sistema operativo.

Los Servicios Web XML marcan una nueva etapa en el desarrollo de software desde la utilización de COM y DCOM. Anteriormente se hicieron intentos de utilizar código empaquetado en forma remota sin sufrir incompatibilidad entre las aplicaciones;

pero no es hasta la creación de los Servicios Web XML que este sueño se hace realidad. El protocolo SOAP, que permite hacer llamadas a objetos complejos como *DataSets*, tiene tiempo circulando en el mercado de software; al principio Microsoft lo ofrecía en forma separada y gratuita. Ahora, con la introducción de la infraestructura .NET, ya viene integrado dentro de él. Los Servicios Web XML pretenden, además de centralizar procesos, crear una comunicación entre aplicaciones, sin importar el lenguaje ni la plataforma. Por ejemplo, se podría crear un portal que tuviera diferentes servicios, que pudieran ser consumidos por diferentes aplicaciones, y que se encargaran de realizar operaciones y consultar información de empleados de gobierno, en donde diferentes empresas gubernamentales como el Seguro Social, la Secretaría de Hacienda, el Infonavit, etc., pudieran acceder a estos Servicios Web XML centralizados y de esta manera actualizar sus bases de datos o realizar movimientos de una sola vez, sin tener que hacer los cambios antes mencionados en cada una de estas instituciones. Pensemos en las instituciones financieras, el Banco de México podría crear un portal que ofrezca diversos Servicios Web XML como validación de tarjetas de crédito, chequeo de cuentas, validación de fondos, etc. cualquier aplicación gubernamental o no gubernamental podría hacer uso de éstos servicios, para saber si la tarjeta de crédito de un cliente tiene fondos, si se le autoriza un crédito, si no tiene deudas, etc. Es importante aclarar que no todos los Servicios Web XML que se encuentran en Internet son gratuitos y que una gran mayoría exige una contraseña para su utilización. Empresas como Microsoft, IBM y BEA han invertido mucho tiempo en el desarrollo de los Servicios Web XML, porque saben que son la respuesta para interconectar diferentes aplicaciones en diferentes plataformas; recomiendo al lector que se especialice en este tema. Mi predicción es que los Servicios Web XML se convertirán en los pilares de la nueva arquitectura de sistemas de información, así como en una necesidad para las empresas del nuevo siglo.

Es importante mencionar, que la rivalidad entre lenguajes de programación no beneficia a nadie, por lo que los Servicios Web XML permiten que diferentes programadores que trabajan con diferentes lenguajes tengan acceso a estos servicios con mucha facilidad, sin tener que migrar sus aplicaciones a una nueva plataforma. Estos son solo algunos de los beneficios de los Servicios Web XML, para sacar mayor provecho de estos servicios en las aplicaciones que desarrolle, es recomendable que se familiarice con este tema en el sitio Web de Microsoft.

Conclusión de la obra

Una herramienta informática debe de ser capaz de facilitar la tarea del diseñador de software. La mayoría de las empresas requieren aplicaciones de buena calidad en tiempos muy cortos, es por eso que no pueden esperar muchas semanas ni meses para el desarrollo de una aplicación. Para un diseñador de sistemas, la parte compleja debe estar en el diseño, no en la codificación.

ASP.NET ofrece la capacidad de simplificar el desarrollo, separando el código HTML del código de Visual Basic.NET; por lo que no hay que codificar en HTML para hacer tablas, cajas de texto, botones, listas, etc. Solamente debe de arrastrar el control y pegarlo en la forma Web, después implementar el código asociado con éste.

Podría pensarse que el simplificar código solamente favorece al programador inexperto, pero no es así, los diseñadores avanzados están enfocados en el diseño del sistema, cuando ya tienen esto, entonces necesitan herramientas que les ofrezcan llevar a cabo su proyecto en la forma más rápida, sencilla y segura.

El valor de un diseñador, no se encuentra en la cantidad de código que pueda realizar, sino en la correcta definición, planeación, desarrollo e implementación de un sistema de información.

En este estudio se han introducido las principales características de ASP.NET. Gracias a las herramientas ofrecidas por la infraestructura .NET, es posible desarrollar aplicaciones en forma más sencilla y segura.

Gracias a que ASP.NET hace una separación lógica de código de lenguaje .NET y código HTML, se puede compilar en forma independiente. Los ensamblados tienen una estructura que les permite autodefinirse para no tener que ser registrados en la computadora, lo que elimina el *infierno de las DLL*.

El servidor Web IIS de Microsoft tiene muchas funciones; es muy sencillo de configurar, se puede enlazar muy fácilmente con otros servidores como Microsoft Exchange, Microsoft Commerce Server, ISA, etc. Microsoft firmó un convenio con la empresa Covalent, que es una de las principales proveedoras de Apache 2.0, para que ASP.NET pueda funcionar bajo Apache. Para mayor información puede ver en:
http://www.newsfactor.com/perl/story/18740.html

http://www.covalent.net/products/rotate.php?page=93.

Es importante mencionar que cuando se elige una tecnología Microsoft como plataforma, no solamente se adquiere un producto aislado, sino un conjunto de componentes con infinidad de funciones que crean una solución integral para cualquier empresa.

Todos los servidores y productos Microsoft interactúan fácilmente entre sí, y con otros sistemas operativos como UNIX, Macintosh e IBM AS400. Microsoft tiene soporte en todo el mundo, documentación en línea, existen infinidad de manuales para sus productos.

La plataforma .NET ofrece características muy interesantes, como el recolector de basura, que elimina las referencias circulares, los dominios de aplicación, que permite que varios procesos se ejecuten en un solo dominio, el código administrado, que provee con código seguro de ejecutar, la librería de clases, etc.

ASP.NET permite que el desarrollador de cualquiera de los lenguajes de la plataforma .NET, como Visual Basic.NET, C#, Visual C++, J#, pueda realizar aplicaciones Web, sin la necesidad de aprender un lenguaje diferente al que utiliza para la programación de aplicaciones Windows, lo que le da una flexibilidad inmediata.

Otra innovación de la infraestructura .NET se encuentra en ADO.NET, ya que tiene muchas características que lo convierten en el candidato ideal para ambientes en donde la cantidad de usuarios es muy grande. Se pueden guardar localmente relaciones, restricciones y varias tablas de uno o más orígenes de datos. El formato de transporte es XML, por lo que puede ser consumido por cualquier lenguaje que soporte esta tecnología y sistema operativo. Otra característica importante son los Servicios Web XML, que son componentes de software que son publicados en Internet para poder ser consumidos en forma pública o privada. Al igual que ADO.NET, el formato de transporte de los servicios es XML y el medio de transporte es TCP/IP. Por lo que una aplicación en una plataforma UNIX puede consumir perfectamente un Servicio Web XML que resida en un servidor Windows. Los Servicios Web XML se está convirtiendo en una especialidad del lenguaje, existen publicaciones enfocadas exclusivamente en este tema. La capacidad de ser consumido por múltiples lenguajes y plataformas lo ha hecho muy popular entre la comunidad de programadores.

Además de la tecnología .NET existe otra excelente tecnología llamada J2EE, de Sun Microsystems, la cual utiliza el lenguaje Java 2 para desarrollar una infinidad de aplicaciones, entre ellas las Web con JSP (Java Server Pages). Puedo afirmar que ésta

tecnología es la competencia principal de .NET, ambas tienen características muy similares. Los programas CGI (*Common Gateway Interface*), como PERL tienen la deficiencia de crear un proceso por cada usuario que entre a la aplicación, en sitios que no reciban muchos usuarios no representa mayor problema, sin embargo cuando aumenta este número el servidor pronto se saturará.

Algunas de las similitudes entre la tecnología .NET y J2EE son: las dos tienen una infraestructura y una jerarquía de clases, por un lado Microsoft tiene la infraestructura .NET y *los espacios de nombre,* por otro lado Sun Microsystems tiene J2EE y los paquetes de clases. Cuando se analiza el código Visual Basic.NET y el de Java 2, uno se da cuenta que son muy similares, al final ambos generan una clase. Java 2 necesita de la máquina virtual de Java para que una aplicación pueda funcionar en una PC, sin importar el sistema operativo que utilice. Microsoft por su parte utiliza JIT (Just in Time Compiler), que como vimos en el capítulo I, interpreta un ensamblado para después ejecutarlo; lo único que necesita cualquier sistema operativo diferente de Windows para poder ejecutar aplicaciones .NET es el JIT, que Microsoft promete distribuirlo en un futuro para otros sistemas operativos. Por lo que deduzco que las aplicaciones .NET muy pronto serán multiplataforma.

Quisiera terminar con una reflexión, cuando el diseñador de sistemas escoge una tecnología para desarrollar aplicaciones, debe experimentar con diferentes soluciones para escoger la que mejor se adapte a sus necesidades. No es necesario que siempre escoja la misma tecnología, es importante no caer en un fanatismo, en donde se defienda a un producto ó marca a toda costa. Todas las opciones tienen sus ventajas y desventajas, siempre debe ser imparcial e inteligente al seleccionar una ó varias tecnologías, lo más importante es la calidad y desempeño del producto a desarrollar.

Mi recomendación es que el lector estudie primero una tecnología, trabaje un rato con ella. Después que estudie otras, de tal forma que el diseñador de sistemas, tenga una gama de lenguajes y tecnologías para ofrecer soluciones integrales a sus clientes.

Espero que este estudio haya sido de su agrado, así como una gran ayuda que sirva como pilar para la construcción de aplicaciones en Internet.

Agradezco el interés por la lectura de esta obra, le deseo mucha suerte en sus próximos proyectos.

Bibliografía

BÜHLER, Erich R., ***Visual Basic.NET Guía de migración y actualización,*** Editorial McGraw-Hill Profesional, España, 2002, 949 pp.

HAWTHORNE, Rob, ***Desarrollo de bases de datos en Microsoft SQL Server 2000***, Pearson Educación de México, México, 2002, 478 pp.

HILLIER, Scot, ***COM+ Programming with Visual Basic***, Editorial Sams, Estados Unidos de Norteamérica, 2000, 477 pp.

MICROSOFT CORPORATION, ***Programming with Microsoft Visual Basic .NET***, Microsoft Corporation, Colombia, 2002, 577 pp.

SORIA, Ramón, ***HTML Diseño y creación de paginas Web***, AlfaOmega Grupo Editor, México, 1998, 153 pp.

Otras fuentes

http://msdn.microsoft.com/library/default.asp

http://www.123aspx.com

http://www.4guysfromrolla.com/

http://www.asp.net/

http://www.developersdex.com/Default.asp

http://www.dotnet247.com/247reference/default.aspx

http://www.gotdotnet.com/

http://www.vbcity.com

http://www.wown.info