Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

Licenciatura en Sistemas Computacionales

ASP .NET orientado al desarrollo de aplicaciones Web

Monografía

Que para obtener el grado de Licenciado en Sistemas Computacionales

P R E S E N T A

Mario Brianza Padilla

Director del trabajo

Lic. Norma Laura Salazar Viveros.

Pachuca de Soto Hgo., Agosto de 2006
México.

ASP .NET orientado al desarrollo de aplicaciones Web

Contenido

CAPÍTULO 2 ASP y la nueva tecnología .NET

CAPÍTULO 3 Especificaciones y Aplicaciones con ASP .NET

CAPÍTULO 4 Material de apoyo para ASP .NET

Objetivo General

Presentar un panorama general de la **tecnología ASP .NET** y los elementos que interactúan con la misma en la creación de *aplicaciones Web*.

Objetivos Específicos

- Conocer los elementos involucrados en el funcionamiento de la **tecnología ASP .NET**.

- Proporcionar los conceptos básicos para la creación de aplicaciones **ASP .NET**.

- Determinar los beneficios generados al utilizar **ASP .NET** en la creación de aplicaciones.

Justificación

Es tal el auge de los *medios electrónicos*, principalmente **Internet**, que actualmente las empresas líderes en su ramo deben gran parte de su éxito a la incorporación de estos medios a su negocio.

La **tecnología .NET** de ***Microsoft*** ha impactado los ámbitos corporativos, orillando a las empresas a *cambiar su tecnología* y propiciando el *cambio de su filosofía*, ya que en la actualidad las corporaciones han optado por una visión basada en **.NET**.

En definitiva, la **tecnología .NET** no sólo representa una *revolución en el desarrollo de aplicaciones*, si no que impacta ampliamente el *desarrollo de aplicaciones Web*, cuyo objetivo es intercambiar información de cualquier lado a otro, esto significa trasladar información de una computadora a un teléfono celular o algún artefacto que utilice **tecnología wireless** o viceversa.

Lo anterior hace notar que dentro de un entorno cada vez más globalizado las *aplicaciones web distribuidas* garantizan su permanencia dentro de los sistemas informáticos utilizados para aprovechar la red Internet y además permiten que tecnologías como **ASP .NET** se conviertan en la base para el desarrollo de los sistemas actuales.

Introducción

El presente trabajo pretende difundir la **tecnología .NET**, la cual actualmente ha logrado un auge importante en el desarrollo de aplicaciones orientadas principalmente al ambiente Web.

Debido a la gran magnitud del campo de estudio, el contenido se centra en **ASP .NET** y en los elementos que permiten la creación de una aplicación basada en esta tecnología.

La estructura de esta monografía esta conformada por 5 capítulos; Capitulo 1 **Orígenes de ASP .NET**, Capitulo 2 **ASP y la nueva tecnología .NET**, Capitulo 3 **Especificaciones y Aplicaciones con ASP .NET**, Capitulo 4 **Material de apoyo para ASP .NET** y Capitulo 5 **Seguridad en ASP .NET**.

La parte medular de la monografía se ubica en los **Capítulos 3 y 4**, debido a que en ellos se presenta un contenido orientado a la creación de pequeñas aplicaciones que permitirán al lector comprender de mejor forma el funcionamiento de la **tecnología ASP .NET**.

CAPÍTULO 1

Origen de ASP.NET

1.1 Aplicaciones Web

El desarrollo de las *aplicaciones Web* comienza con la simple utilización de **HTML *(Hypertext Markup Lenguaje)***, el cual, de acuerdo a las necesidades de las aplicaciones, es complementado con el uso de la **CGI *(Common Gateway Interface)*** enseguida es utilizada también la **ISAPI *(Internet Server Application Programming Interface)*** , sucedida por **ASP *(Active Server Pages)***, finalizando con el punto de estudio **ASP .NET** haciendo mucho más fácil y escalable el desarrollo de *aplicaciones Web dinámicas*.

En el principio de la evolución de las *aplicaciones Web*, **HTML** era una buena opción para el desarrollo de aplicaciones, y lo sigue siendo. **HTML** es utilizado para describir la presentación de un texto o un gráfico en una aplicación. Su código cuenta con "referencias" o "palabras clave" que permiten identificar el inicio y final de comandos o sentencias específicas.

Sin embargo, **HTML** es útil para ***contenidos estáticos***, ya que en lo referente a ***contenidos dinámicos*** no cuenta con los recursos para lograr desplegar las visualizaciones finales.

En el inicio de los ***90's***, al surgir la necesidad de compartir información y por consiguiente la compatibilidad de formatos en los documentos utilizados, **HTML** resultaba una excelente opción para lograr que los usuarios visualizaran la información sin importar el procesador de texto o el tipo de computadora utilizada.

HTML y el protocolo utilizado a través de la Web, el **HTTP *(Hypertext Transfer Protocolo)*** tuvieron algunos avances en comparación a cuando fueron introducidos. **HTTP** hizo demasiado eficiente el uso de el limitado ancho de banda con el que se contaba en ese entonces con el manejo de las ***páginas HTML estáticas***.

1.2 Aplicaciones Dinámicas

El protocolo **HTTP** puede ser utilizado para ***contenidos dinámicos***. Debe entenderse por ***contenido dinámico*** *el flujo de información en ambas direcciones*, es decir, al utilizar un formulario en una página Web, un usuario puede enviar una solicitud para el uso de algún contenido. Obviamente la comunicación entre el ***cliente*** y el ***servidor*** implica mas que solo un formulario y el contenido del que pueda hacer uso.

A mediados de los ***90's***, muchas compañías fueron decayendo al verse obligadas a bajar sus costos. Las "pesadas" aplicaciones utilizadas de modo cliente, y el desafió de las **DLLs *(Dynamic Link Libraries)*** fueron la causa de gran parte de esos costos.

La mayoría de las compañías comenzaron a ver las *aplicaciones Web* de tal forma en que pudieran expandirse en el campo laboral con el mínimo impacto sobre las máquinas cliente. Es así como surge la necesidad de incorporar ***contenidos dinámicos*** cada vez más complejos cubriendo las múltiples necesidades de las compañías.

1.3 CGI (Common Gateway Interface)

La primera solución para el manejo del ***contenido dinámico*** Web fue utilizando **aplicaciones CGI**, las cuales son muy populares en el ambiente UNIX. Las **aplicaciones CGI** son *programas ejecutables que pueden correr sobre un servidor Web*, por lo tanto pueden ser utilizados para crear ***contenido Web dinámico***.

Un simple ejemplo de este tipo de aplicaciones puede ser una aplicación creada desde consola, la cual despliegue la clásica frase *"Hello World"* en el Browser.

1.3.1 Ventajas al utilizar Aplicaciones CGI

Las **aplicaciones CGI** tienen múltiples usos, por ejemplo, es posible *acceder a bases de datos*, *leer archivos*, *trabajar con registros* y muchas mas cosas que un programa Win32 puede hacer.

Los **programas** o **aplicaciones CGI** cuentan con facilidad a la hora de realizar pruebas, y su ciclo *código, prueba, compilación* es recursivo.

1.3.2 Desventajas al utilizar Aplicaciones CGI

En los **modelos CGI** una vez que el programa ha sido ejecutado y éste ya existe, se puede *reubicar y modificar* el **programa CGI** como sea posible a algún otro programa, sin embargo, el poder lograr esto, es el problema central de las **aplicaciones CGI**. Es decir, cuando un **programa CGI** es ejecutado, éste es cargado en memoria, y cuando este termina, entonces es removido de la memoria que utiliza.

Esto quiere decir que los **programas** o **aplicaciones CGI**, trabajan bajo la ***creación y destrucción de procesos***; el crear un proceso es una operación costosa en cuanto a los recursos utilizados, comparado con la simple lectura de un archivo **HTML**.

La ***creación y destrucción de procesos*** para cada una de las solicitudes que pueden realizar varios usuarios es el origen de los problemas de esta solución. Si existieran **100 usuarios o clientes**, accesando al mismo **programa CGI**, habría **100 instancias** de este programa en memoria. Esto origina un rápido consumo de recursos en un servidor Web, además de problemas de escalabilidad.

1.4 ISAPI (Internet Server Application Programming Interface)

Debido a los problemas de desempeño y escalabilidad ocasionados por **CGI**, ***Microsoft*** desarrolló una nueva manera de construir *aplicaciones escalables*. Esta nueva alternativa es nombrada **Internet Server Application Programming Interface *(ISAPI)*. ISAPI** en lugar de englobar la funcionalidad en archivos ejecutables, utiliza **DLLs** por lo que se logra un gran avance en el *rendimiento y escalabilidad*.

1.4.1 Tipos de ISAPI

Existen dos tipos de **ISAPI** basadas en **DLLs**, uno de ellos es las **extensiones ISAPI** y el otro los **filtros ISAPI**. Las **extensiones ISAPI** son llamadas por una *dirección URL* enviada por el **servidor IIS *(Internet Information Server)***, por ejemplo: ***http://localhost/sayhelloisapi/sayhelloisapi.dll***.

La **extensión ISAPI** puede ser llamada también de modo que pueden permitir que una sola **extensión ISAPI** logre desarrollar multitareas. Las **extensiones ISAPI** son utilizadas típicamente para peticiones de procesos cliente y salidas por medio de **HTML**, lo cual es muy similar a la manera como son usados en **CGI**.

Por otro lado, los **filtros ISAPI** cuentan con una función que no puede ser imitada por las **aplicaciones CGI**. Los **filtros ISAPI** *nunca son llamados explícitamente*, en vez de eso son llamados por medio de **IIS** como respuesta a los eventos a lo largo de la petición o solicitud.

Uno de los usos más comunes de los **filtros ISAPI** es para *obtener recursos de autentificación*. Son utilizados también para *modificar* el **HTML** que será enviado al cliente; un ejemplo sencillo puede presentarse cuando es necesario cambiar el color del fondo de cada página.

1.4.2 Ventajas al utilizar ISAPI

A diferencia de las **aplicaciones CGI**, una **extensión ISAPI** *es almacenada una sola vez en el tiempo activo del servidor*, por lo que la memoria es utilizada para otros propósitos. Otro punto importante es que las **aplicaciones ISAPI** *corren en el espacio de procesamiento de* ***IIS***, permitiendo así una mejor comunicación entre ellos.

Las nuevas aplicaciones, corren en un proceso separado de los servicios **IIS**, corriendo dentro de un espacio de procesamiento y principalmente ofreciendo significativos avances en el rendimiento y escalabilidad.

1.4.3 Desventajas al utilizar ISAPI

Casi todos los problemas generados por **ISAPI** se relacionan con el *desarrollo de las aplicaciones*. Para desarrollar una **aplicación ISAPI**, el desarrollador debe estar familiarizado con el lenguaje ***C++***, las **MFC** ***(Microsoft Foundation Classes)*** así como también con **HTML**. A pesar de que la mayoría de los desarrolladores están familiarizados con **MFC** y **HTML**, al desarrollar una **aplicación ISAPI** no es fácil diferenciar entre el código central de la aplicación y los detalles referente a la presentación.

El segundo problema ocasionado al desarrollar una **aplicación ISAPI**, se origina cuando se realizan las *pruebas de construcción de la* ***DLL***. Es decir, cuando es ejecutada una simple aplicación, ésta es cargada en memoria y permanece ahí hasta que el servicio de publicación de ***WWW*** es detenido.

Por lo tanto, mientras el servicio no se ha detenido, no es posible remplazar la **aplicación ISAPI**; esto hace posible que la petición generada por la **aplicación ISAPI** no sea identificada por el **Servidor IIS**. Sin embargo, antes de realizar una

actualización de un **extensión ISAPI**, se debe probar en modo desactivado, para verificar que no existan errores ocultos a causa de las variables, ya que éstas siempre deben estar siendo *inicializadas* debido a que la **DLL** es cargada en cada una de la peticiones.

1.5 ASP (Active Server Pages)

Durante la creación de la *versión beta* de **IIS 2.0**, la cual después formó parte de **Windows NT 4.0**, ***Microsoft*** introdujo una nueva tecnología inicialmente llamada **"Denail"**. Todo esto aconteció durante el periodo activo de ***Microsoft*** o **" Microsoft's Active Period"**, por lo que dicha tecnología fue eventualmente llamada **Active Server Pages**, o simplemente **ASP**.

Muchas versiones de **ASP** han sido creadas a lo largo de su historia, sin embargo, se mencionan a continuación las más sobresalientes: **ASP 2.0** junto con **IIS 4.0** incluidas dentro de **Windows NT 4.0 Option Pack** y **ASP 3.0** junto con **IIS 5.0** incluidas en **Windows 2000**.

ASP *es significativamente un ambiente de desarrollo diferente*, debido a que es un ambiente basado en ***scripts***. Simplemente se edita la página, colocándola en el directorio configurado de manera adecuada con los permisos asignados y después es llamada por medio de un browser o navegador.

Sin embargo, una idea original como el uso de **scripts**, después se convirtió en una *desventaja* debido a que el código **ASP** podía mezclarse con el estándar **HTML**.

Cabe mencionar, que debe tenerse cuidado en el manejo del término abreviado **ASP**, debido a que suele confundirse con el de ***Application Servicie Providers*****,** para el cual se utiliza la misma abreviatura.

En un principio el **código ASP** fue comúnmente escrito en ***Microsoft Visual Basic Scripting Edition (VBScript)***, después ***Microsoft Java Script*** fue utilizado también. El aspecto más interesante de **ASP** es conocer cómo funciona, ya que parte de la utilización de **extensiones ISAPI**.

Para precisar, La compañía creadora de esta nueva tecnología, ***Microsoft***, la define de la siguiente manera:

***Las Active Server Pages son un ambiente de aplicación abierto y gratuito en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y poderosas para el Web*.**

1.5.1 ¿Cómo funciona la tecnología ASP?

ASP *es una tecnología dinámica que funciona del lado del servidor*, lo que implica que cuando el usuario solicita un **documento ASP**, las instrucciones del programa contenidas en el ***script***, son ejecutadas para enviar el código resultante **HTML** al navegador.

El ***servidor Web****, al enviar al cliente una* **página ASP***, ejecuta el código que contiene la página y posteriormente envía al cliente el resultado de la ejecución de dicho código.* De esta forma se consigue poder modificar el contenido antes de ser enviado, además se hace compatible con cualquier navegador, pues lo que se envía al cliente es **HTML** común, **Javascript, Flash*,*** entre otros.

En la **Figura 1.1** se muestra el proceso de intercambio de información al utilizar la **tecnología ASP**.

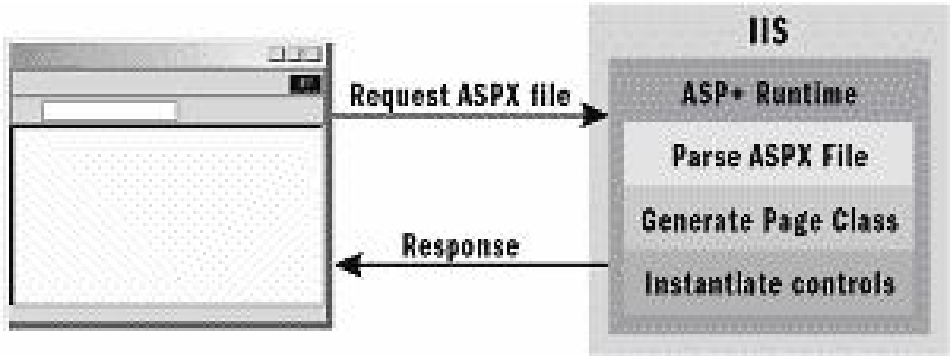


Figura 1.1. Proceso de intercambio de información con ASP.

1.5.2 Ventajas al utilizar ASP

ASP logró hacer mucho más sencillo lo que antes resultaba sumamente difícil, crear ***contenido dinámico Web.*** Al crear **aplicaciones CGI** o **aplicaciones ISAPI** estaba garantizada la complejidad; ahora con **ASP** el desarrollo resulta fácil.

ASP utiliza ***VBScript*** como lenguaje de programación, y millones de desarrolladores están familiarizados con ***Visual Basic***, ***Visual Basic for Applications (VBA)*** o ***VBScript***. Por lo tanto para todos esos desarrolladores **ASP** fue la forma mas sencilla para entrar a la era de Internet.

En algún otro caso los desarrolladores tendrían que aprender un nuevo lenguaje pero con **ASP** no será necesario debido a la particular manera de construir aplicaciones Web.

Otro punto importante es el *fácil acceso a bases de datos* a través del ***Microsoft* ActiveX Data Objects *(ADO)***. Cuando es necesario generar contenido dinámico **ADO** permite una sencilla inclusión de los datos.

Finalmente, el **modelo de desarrollo ASP** permite construir en base a la escritura y ejecución de código, no habiendo necesidad de llevar a cabo compilación o una serie de complicados pasos.

1.5.3 Desventajas al utilizar ASP

ASP *es una poderosa herramienta para desarrollos de aplicaciones Web grandes y aplicaciones Web escalables*. Sitios Web como ***www.microsoft.com*** y ***www.dell.com*** y muchos otros grandes y pequeños han utilizado ASP con resultados satisfactorios.

Sin embargo, es necesario saber que en pocos casos **ASP** es lo suficientemente rápido al interpretar el código ***VBScript*** o ***JScript*** en cada una de las solicitudes **cliente-servidor**.

1.6 Visual Studio .NET

Visual Studio .NET es un **IDE** ***(Entorno de Desarrollo Integrado)*** desarrollado por ***Microsoft*** a partir de 2002. Es para el sistema operativo ***Microsoft Windows*** y está pensado, principal pero no exclusivamente, para desarrollar para plataformas Win32.

Visual Studio .NET soporta los nuevos **lenguajes .NET: C#**, ***Visual Basic .NET*** y ***Managed C++***, además de ***C++***. **Visual Studio .NET** puede utilizarse para construir aplicaciones dirigidas a Windows *(utilizando Windows Forms)*, Web *(usando **ASP .NET** y Servicios Web)* y dispositivos portátiles *(utilizando **.NET Compact Framework**).*

El aspecto de **Visual Studio .NET** es casi idéntico a las versiones anteriores del **IDE Microsoft Visual Studio**. Algunas excepciones destacables son la interfaz más limpia y mayor cohesión. También es más personalizable con ventanas informativas de estado que automáticamente se ocultan cuando no se usan. Todas las versiones de **Visual Studio**, también su predecesora ***Visual C++***, incluyen un depurador integrado en el entorno de edición.

1.6.1 Versiones de Visual Studio .NET

Visual Studio .NET 2002

Versión inicial del año 2002; técnicamente llamada **Visual Studio 7.0**. Es conocida como **Visual Studio .NET 2002** tras la publicación de **Visual Studio .NET 2003**.

Visual Studio .NET 2003

También llamada **Visual Studio 7.1**, esta versión presenta cambios con respecto a la versión anterior.

Cambios sobre la versión 7.0:

- Incorpora **.NET Framework 1.1**.
- Añade soporte del **.NET Compact Framework** *(para dispositivos portátiles)*.
- Añade el **ASP .NET Mobile Designer** *(para construir aplicaciones web para dispositivos portátiles, PDAs y pagers)*.
- Actualiza **Visual Basic** a la ***versión 7.1***.
- Contiene alguna otras mejoras menores y presenta cambios referentes a la Interface.

Visual Studio 2005

La versión definitiva en inglés vio la luz en Noviembre del 2005. En castellano hubo que esperar hasta Febrero de 2006.

Características sobresalientes:

- Incorpora **.NET Framework 2.0**.

- Hay más ediciones diferenciadas por el precio y las características.
- La Ayuda cuenta con **refactorización**.
- El desarrollo de páginas con **ASP .NET** ha cambiado.
- Soporte para el nuevo software servidor **Team System**.
- Añadido soporte de **tests** para todo tipo de aplicaciones.

Ediciones de Visual Studio .NET

Cabe mencionar que **Visual Studio .NET** cuenta con diversos tipos de ediciones las cuales son:

- Edición Estándar.
- Edición Profesional.
- Edición Académica *(Edición Profesional, pero con un CD "Student Tools" adicional)*.
- Enterprise Developer Edition.
- Enterprise Architect Edition.

1.6.2 Visual Studio .NET 2002

Visual Studio .NET es la herramienta definitiva para la rápida generación de **aplicaciones Web ASP .NET** a escala empresarial y aplicaciones de escritorio de alto rendimiento.

Visual Studio .NET 2002 incluye herramientas de desarrollo basadas en componentes, como por ejemplo ***Visual C#***, ***Visual Basic*** y ***Visual C++***, así como diversas tecnologías suplementarias para simplificar el diseño, desarrollo e implementación en equipo de las soluciones.

1.6.2.1 Entorno de Desarrollo

Compartir un único **IDE** ofrece numerosas ventajas, incluyendo la consolidación de herramientas similares de distintos productos en un conjunto de herramientas utilizadas en **Visual Studio**.

En base a lo anterior se presentan enseguida las características más sobresalientes en cuanto al entorno de desarrollo de esta versión de **Visual Studio**:

1. Pagina de inicio

La ***Página de inicio*** ofrece una forma rápida de establecer las preferencias de usuario respecto del comportamiento del **IDE**, incluyendo la combinación de asignación de teclado activa, el diseño de ventana y el filtro de Ayuda, así como la posibilidad de tener acceso a proyectos nuevos o existentes.

Además, permite ver vínculos a los artículos, eventos y temas más recientes en **MSDN Online**, así como a importantes recursos para los programadores que utilizan ***Microsoft*** en general. La ***Página de inicio*** aparecerá de forma predeterminada cada vez que inicie **Visual Studio**.

2. Web Browser

Es posible mostrar ***páginas Web*** directamente dentro del **IDE**. Para mostrar una ventana de explorador de Web en el **IDE**, se deberá elegir **Mostrar el explorador** en el **submenú explorador Web** del **menú Ver**. La primera vez que se abre una ventana de **explorador Web** aparece la **Página de inicio** de forma predeterminada. Cuando se abre una ventana de **explorador Web**, aparece la *barra de herramientas Web*, que permite escribir **direcciones URL**, moverse adelante y atrás en el historial de exploración y volver a la página de inicio del **explorador Web**. También se puede

tener acceso a los favoritos del **explorador Web**, así como agregar vínculos a la lista de favoritos desde **Visual Studio**.

3. Ventana de comandos

La ***ventana de comandos*** es una combinación de una línea de comandos y la **ventana Inmediato** de ***Visual Basic***. Dispone de dos modos: **modo comando y modo Inmediato**.

En **modo comando**, se pueden escribir nombres de comandos **IDE** después del ***corchete angular*** **(>)**. Para facilitar el acceso a los comandos de uso más frecuente, se pueden crear alias, o nombres abreviados. La finalización automática funciona con nombres de comando, alias y nombres de archivo.

En **modo Inmediato**, se pueden ejecutar instrucciones de código, establecer o asignar variables, evaluar expresiones, etcétera. También se pueden introducir comandos en el cuadro ***Buscar / Comando*** de la *barra de herramientas* de **Visual Studio**.

4. Administrador de ventanas

Visual Studio facilita más que nunca la presentación de una mayor cantidad de código en pantalla al mismo tiempo. Enseguida se describen sus opciones mas sobresalientes:

Organización por fichas

Esta característica *organiza en fichas* los documentos dentro del **IDE** de forma automática. Por ejemplo, si se editan varios documentos en un editor o diseñador, dichos documentos aparecen en el área de la **Interfaz de Múltiples Documentos (MDI)** en forma de fichas en la parte superior del área.

Ocultar Automáticamente

Ocultar automáticamente *permite minimizar las ventanas de herramientas*, como el **Explorador de soluciones** y el **Cuadro de herramientas**, en los bordes del **IDE**, de forma que ocupen el menor espacio. Después de minimizar las *ventanas de herramientas*, el espacio visible del editor se incrementa.

Desplazarse hacia delante y Desplazarse hacia atrás

Permite *desplazarse entre las ventanas abiertas en el entorno*, así como *en el historial de selecciones y desplazamientos de cursor dentro de los archivos*. Por ejemplo, si edita código en la línea 12 y luego se desplaza a la línea 102, puede utilizar el botón ***Desplazarse hacia atrás*** para volver exactamente a la misma ubicación de la línea 12.

1.6.2.2 Plantillas de Aplicación

Visual Studio .NET proporciona diversas **plantillas** que pueden utilizar los arquitectos de software para reducir la complejidad y el costo del desarrollo de aplicaciones distribuidas. Mediante **Enterprise Templates** *se puede definir la estructura inicial de una aplicación distribuida y proporcionar una guía de arquitectura y tecnología para los equipos de desarrollo*. Aparte de las **plantillas predefinidas de Enterprise Templates**, se pueden crear **plantillas personalizadas**.

1.6.2.3 Herramientas de Edición

Las ***herramientas de edición*** han experimentado diversas mejoras. Asimismo, **Visual Studio** incorpora un **Examinador de objetos**, y admite el comentario de

código. A continuación se mencionan las opciones más sobresalientes referentes a las herramientas de edición:

Edición de código

Visual Studio dispone ahora de un **editor de código unificado** para todos los lenguajes en el **IDE**, con características especializadas para cada lenguaje. El **editor de *código*** *incluye diversas mejoras, como ajuste de línea, búsqueda incremental, esquematización de código, contracción a definición, numeración de líneas, impresión en color y accesos directos*.

Se puede tener acceso a dichas características desde el ***menú Edición*** o desde el ***menú contextual***. También es posible desplazarse adelante y atrás en el historial de selecciones de un documento y entre documentos, mediante los botones ***Desplazarse hacia delante y Desplazarse hacia atrás*** de la *barra de herramientas estándar*.

Edición de HTML

El **Diseñador HTML** dispone de dos vistas, ***Diseño*** y ***HTML***, que ofrecen *flexibilidad en el diseño de páginas Web*. La ***Vista Diseño*** es una superficie de diseño visual con funciones de *doble clic* y *arrastrar* y *colocar* para los elementos de **HTML**.

Al tiempo que se edita en la ***Vista Diseño*** se pueden arrastrar nuevos elementos desde la **ficha HTML** del *Cuadro de herramientas* y colocarlos directamente en la página Web. También se puede hacer clic y arrastrar para cambiar el tamaño de las celdas de tablas, y utilizar el *menú contextual* para agregar filas y columnas a las tablas.

Edición de XML

El Editor **XML** permite crear ***esquemas, conjuntos de datos y documentos XML*** en el **IDE**. Los usuarios pueden especificar la definición de **esquema XML** que se debe utilizar para editar documentos **HTML** y **XML**.

Examinador de Objetos

El **Examinador de Objetos** se ha unificado, y funciona con todos los lenguajes de **Visual Studio**. Permite explorar y buscar objetos y sus miembros dentro de proyectos y referencias, así como en bibliotecas externas.

Vista de Clases

La **Vista de Clases** se ha unificado, y funciona con todos los lenguajes de **Visual Studio**. Esta ventana le *permite examinar y explorar entre los símbolos de su solución*. Los símbolos se organizan por proyecto y se muestran en una vista de árbol jerárquica, lo que indica las relaciones de contención entre ellos.

Se puede cambiar el orden de la vista a *orden alfabético*, *por tipo* o *por acceso*, o *agrupar por tipo*. Se pueden explorar las definiciones, las declaraciones o las referencias, y buscar símbolos. La **Vista de Clases** *incorpora asimismo la posibilidad de crear una nueva* ***carpeta virtual****, a la que se pueden arrastrar símbolos para facilitar el acceso a los mismos*.

1.6.2.4 Herramientas de Depuración

Visual Studio .NET incorpora un único depurador para todos los lenguajes de **Visual Studio**, con una nueva interfaz unificada que combina las características de los

antiguos depuradores de ***Visual C++*** y ***Visual Basic***, así como otras funciones nuevas. Entre las principales mejoras se encuentran:

- Depuración entre lenguajes, para ***Visual Basic***, ***Visual C++***, ***Visual C#***, extensiones administradas para **C++** y **SQL**.
- Depuración de aplicaciones escritas para el **Common Language Runtime**, así como aplicaciones y secuencias de comandos nativas de Win32.
- Capacidad de conectarse a un proceso ejecutado fuera de **Visual Studio** y depurarlo.
- Depuración uniforme de *aplicaciones ejecutadas localmente* o en servidores remotos.
- Depuración de *aplicaciones Web*, localmente o en un servidor remoto.

1.6.2.5 Aplicaciones de ejemplo

Visual Studio incluye *aplicaciones de ejemplo* para mostrar el uso de las herramientas y tecnologías de ***Microsoft*** en la creación de aplicaciones reales. Los dos ejemplos ***Duwamish*** y ***Fitch and Mather***, se han creado mediante **Visual Studio .NET** y las tecnologías de ***Microsoft*** **.NET Framework**, como **ASP .NET** y **ADO .NET**.

Duwamish *es una aplicación ficticia de comercio electrónico, distribuida y basada en Web, para la venta de libros*. El ejemplo es una aplicación completa, de empresa a cliente, que acentúa la funcionalidad, la escalabilidad, la seguridad, el rendimiento y la metodología de diseño.

Duwamish destaca los procedimientos más adecuados en la arquitectura y el diseño de aplicaciones distribuidas basadas en Web, y proporciona un modelo para la correcta integración de la tecnología de **Microsoft .NET Framework** en una solución comercial extensible.

Fitch and Mather *es una aplicación ficticia de correduría de bolsa basada en Web*. El ejemplo muestra el diseño e implementación de una aplicación empresarial robusta con rendimiento, escalabilidad e interoperatividad excelentes. Además, presenta un ejemplo de implementación de transacciones heterogéneas distribuidas a escala empresarial.

1.6.3 Visual Studio .NET 2003

Visual Studio .NET 2003 incluye herramientas de desarrollo basadas en componentes, como ***Visual C#***, ***Visual J#***, ***Visual Basic*** y ***Visual C++***, así como diversas tecnologías suplementarias para simplificar el diseño, desarrollo e implementación en equipo de las soluciones.

1.6.3.1 Entorno de Desarrollo

En esta versión, **Visual Studio** *cuenta con mejoras en las herramientas del entorno de desarrollo*, a continuación se numeran las características mas sobresalientes de este entorno:

1. Página de inicio

En esta versión, se ha rediseñado la ***Página de inicio***. Se pueden establecer las preferencias de usuario para el comportamiento del **IDE** y obtener acceso a proyectos nuevos o a los existentes exactamente igual que en la versión anterior, pero ahora dispone de una **interfaz de usuario** *especialmente diseñada para facilitar la exploración*.

Las secciones ***Mi perfil*** y ***Proyecto*** tienen ahora fichas propias. La **ficha Recursos en línea** de esta versión contiene recursos en línea relacionados con ***Microsoft*** de gran utilidad para los programadores.

2. Explorador de soluciones

Se ha agregado una nueva opción, Realizar seguimiento del elemento activo en el **Explorador de soluciones**, al cuadro de diálogo ***Proyectos y soluciones***, ***Entorno***, ***Opciones***. **Cuando esta opción está seleccionada**, el **Explorador de soluciones** *abre automáticamente la carpeta del elemento activo, se desplaza hasta su nodo y selecciona su nombre*. El elemento seleccionado va cambiando conforme se trabaja con los distintos archivos en un proyecto o solución o con los distintos componentes en un diseñador de entorno de desarrollo integrado.

Cuando esta opción está desactivada, la selección en el **Explorador de soluciones** *no cambia automáticamente*. De forma predeterminada, esta opción está habilitada, pero se deshabilita al seleccionar el perfil ***"Programador de Visual C++"*** o ***"Programador de Visual C#"*** en la ficha ***Mi perfil*** de la **Página de inicio de Visual Studio**.

El **Explorador de soluciones** de esta versión cuenta con dos nuevos iconos:

- ***Desprotegido en exclusiva:*** el elemento está *desprotegido de una base de datos de control de código fuente sólo para un programador*. El resto de desarrolladores no puede obtener acceso al archivo.

- ***Desprotegido para uso compartido:*** el elemento está *desprotegido de una base de datos de control de código fuente para que lo pueda utilizar un equipo de desarrollo*. Las diferentes versiones del elemento se combinarán en el momento de la protección.

3. Opciones y Configuraciones

Es posible **copiar determinadas configuraciones** del ***cuadro de diálogo Opciones*** de una versión anterior de **Visual Studio .NET** a una versión más reciente. Si se tienen instaladas dos versiones diferentes del programa en el mismo equipo, la primera vez que se inicie la versión más reciente de **Visual Studio .NET**, aparecerá un cuadro de diálogo que permite *migrar la configuración existente*. Si se descarta este cuadro de diálogo, puede mostrarse más adelante ejecutando el siguiente comando en la ***línea de comandos de Windows***: ***devenv /migratesettings***.

Después de migrar la configuración anterior del ***cuadro de diálogo Opciones*** a la nueva versión de **Visual Studio .NET**, se deberá seleccionar ***Opciones*** en el ***menú Herramientas*** para mostrar este cuadro de diálogo y revisar la configuración. La mayoría de las opciones deben tener los valores que estableció la última vez.

4. Referencias Web

El panel Iniciar la búsqueda de **servicios Web XML** del ***cuadro de diálogo Agregar referencia Web*** proporciona en esta versión vínculos Web a orígenes locales y de Internet de servicios Web disponibles. Es posible utilizar este panel para buscar el servicio Web deseado y después especificar el nombre de una referencia Web para utilizarla en el código seleccionando ***Agregar referencia***. La nueva referencia aparecerá en el **Explorador de soluciones** en el nodo Referencias Web del proyecto activo.

Entre los vínculos Web a orígenes de servicios Web disponibles se encuentran incluidos los siguientes:

- ***Servicios Web del equipo local:*** *Es posible seleccionar este vínculo para obtener la lista de servicios Web disponibles en el equipo en el que esté trabajando con* ***Visual Studio .NET***.

- ***Examinar servidores UDDI de la red local:*** *Este vínculo permite mostrar los servidores disponibles en la red de área local que proporcionen servicios Web*.

- ***Directorio UDDI:*** *Con este vínculo se pueden buscar servicios Web en el registro de empresas UDDI de* ***Microsoft***.

- ***Probar el directorio UDDI de Microsoft:*** *Este vínculo permite buscar servicios Web que todavía estén en fase de desarrollo y que se hayan enviado para prueba*.

5. Opción Generar

Se ha agregado una nueva opción, ***Generar proyectos de inicio y dependencias únicamente al ejecutar***, a *las opciones Generar y Ejecutar del cuadro de diálogo Proyectos y soluciones, Entorno, Opciones*. Si esta opción está seleccionada y se *presiona* F5 o se elige el comando Iniciar o Generar en el ***menú Depurar o Ejecutar***, sólo se generará el proyecto de inicio y sus dependencias.

Con esta opción desactivada, al *presionar* F5 se generarán todos los proyectos, dependencias y archivos de soluciones. De forma predeterminada esta opción está deshabilitada, pero se habilita al seleccionar el perfil ***"Programador de Visual C++"*** y ***"Programador de Visual C#"*** en la ficha *Mi perfil de la Página de inicio de* ***Visual Studio***.

Se han incluido tres nuevos comandos, disponibles sólo para los proyectos de ***Visual C++***, en el ***submenú Sólo proyecto*** del ***menú Generar***:

- **Generar sólo <Nombre de proyecto>**
- **Volver a generar sólo <Nombre de proyecto>**
- **Limpiar sólo <Nombre de proyecto>**

Estos tres comandos generar, vuelven a generar o limpian sólo el proyecto de ***C++*** que esté seleccionado en el **Explorador de soluciones**, sin generar o limpiar dependencias del proyecto o archivos de la solución.

1.6.3.2 Depuración en Visual Studio .NET 2003

El depurador de **Visual Studio .NET 2003** se ha mejorado agregando varias características nuevas:

- **Mejoras en la seguridad**, *que incluyen una nueva restricción que limita la depuración* ***Just-In-Time*** *al equipo local*.

- **Depuración remota mediante canalizaciones***, una nueva alternativa mucho más segura para la depuración* ***TCP/IP***.

- **SOS**, *una herramienta eficaz para la depuración desde una ventana Comando*.

- **Compatibilidad** *para la descarga automática de símbolos de depuración desde un servidor de símbolos*.

- **Mensajes de error mejorados**, *especialmente para los errores que se producen durante la depuración de aplicaciones Web*.

1.6.3.3 Herramientas para el desarrollo de dispositivos

El entorno de desarrollo integrado de **Visual Studio .NET 2003** incluye ahora herramientas para desarrollar aplicaciones para dispositivos inteligentes, como ***Pocket PC***. Mediante las herramientas y **.NET Compact Framework**, un subconjunto de .**NET Framework**, puede *crear, generar, depurar e implementar* aplicaciones que utilizan **.NET Compact Framework** para ejecutarse en **Asistentes Digitales Personales (*PDA*)**, teléfonos móviles y otros dispositivos de recursos restringidos.

El **Diseñador de ASP .NET Mobile** complementa a **ASP .NET** y a **.NET Framework**, permitiendo generar aplicaciones Web para ***teléfonos móviles***, ***PDA*** y **l*ocalizadores***. Este diseñador está integrado en el **IDE** de **Visual Studio**. Puede crear *aplicaciones Web móviles*, utilizar el diseñador móvil para modificar un *formulario Web móvil* y, a continuación, generar y ejecutar la aplicación, todo desde **Visual Studio**.

1.6.4 Visual Studio 2005

Siendo esta la versión más reciente de **Visual Studio**, es presentada con muchas novedades en el entorno de desarrollo, en el **.NET Framework** y en el propio lenguaje de programación.

1.6.4.1 Entorno de Desarrollo

Las novedades que presenta **Visual Studio 2005** en el entorno de desarrollo son las siguientes:

1. Configuración

Los ***valores de configuración predefinidos*** *están compuestos por una serie de personalizaciones realizadas en el entorno de desarrollo integrado basadas en distintos tipos de actividades de desarrollo y en las propias personalizaciones del usuario*. Estas personalizaciones incluyen configuraciones de ventana, la visualización u ocultación de los comandos de menú, cambio de nombre de los menús y de los comandos de menú, métodos abreviados de teclado y cambio de las opciones predeterminadas de las herramientas, por citar algunas.

2. Administración del diseño de las ventanas

Esta versión incluye ***comentarios visuales mejorados*** para el acoplamiento de las ventanas. Cuando se arrastra una *ventana de herramientas* por un marco en el que se puede acoplar, aparece un rombo de guía. Las cuatro flechas del rombo señalan hacia los bordes del marco que lo contiene. Cuando la ventana que arrastra llega a una posición en la que se puede acoplar, se oscurece la flecha que señala hacia el borde en el que se puede anclar.

3. Página de inicio

En esta versión, se ha rediseñado completamente la ***Página de inicio***. La nueva ***Página de inicio*** se compone de una única página con cuatro áreas de información independientes: ***Abrir proyecto existente, Introducción, Titulares y Noticias***.
Como en versiones anteriores, se pueden abrir los proyectos modificados recientemente, o crear rápidamente proyectos, así como ver ciertos temas de Ayuda. Además, ahora también se puede tener acceso a la información de productos y eventos de ***Microsoft***, así como a los canales **RSS** desde dentro de **Visual Studio**.

4. Integración de la comunidad

Esta versión facilita aún más el acceso a los recursos en la ***comunidad de desarrolladores***. Un nuevo menú, denominado **Comunidad**, aparece en el **IDE**. Desde este menú, puede publicar preguntas en los grupos de noticias de **MSDN**, enviar comentarios sobre productos a ***Microsoft***, obtener acceso a útiles sitios Web y buscar en pantalla componentes para utilizarlos en sus aplicaciones.

1.6.4.2 Edición de código en Visual Studio 2005

Esta versión de **Visual Studio** incluye nuevas funciones y mejoras en el editor de texto, la página Web y diseñador de HTML, y el editor XML. A continuación se describen las características más sobresalientes referentes a la edición de código en **Visual Studio 2005**:

- ***Fragmentos de código***: **Visual Studio** ahora proporciona segmentos de código de ejemplo listos para insertar en los proyectos de ***Visual Basic***, ***Visual C#*** o ***Visual J#.***

- ***Etiquetas inteligentes***: Son similares a las etiquetas inteligentes de **Office**, las *etiquetas inteligentes* de **Visual Studio** realizan tareas comunes disponibles que se aplican al contexto de su trabajo. Por ejemplo, mediante las etiquetas inteligentes ahora se pueden corregir algunos errores comunes en ***Visual Basic*** con un *clic* de un botón.

- ***Refactorización***: Ahora se pueden utilizar herramientas para actualizar la estructura interna del código de ***Visual C#*** y ***Visual Basic***, un proceso denominado refactorización. ***Refactorizar*** *es el proceso de modificar el código fuente pero sin modificar su temática.* Las opciones de refactorización

disponibles incluyen *cambiar el nombre, extraer el método, extraer la interfaz, cambiar la firma y encapsular el campo*.

- ***Control de cambios***: Es posible ver dónde se ha modificado un archivo en la sesión del **IDE** actual. Las modificaciones se identifican mediante un ***indicador visual*** en el margen. *Se marcan las líneas que se han editado o las líneas adyacentes a las líneas eliminadas*. Si el margen está resaltado en color amarillo significa que se editó la línea y que el archivo tiene aún que guardarse. Si el margen está resaltado en verde, significa que ha guardado el archivo desde que se modificó la línea.

- ***Ventana Marcador*** : Esta ***ventana de herramientas*** permite *administrar y controlar los marcadores*. Podrá colocar los marcadores relacionados en carpetas, asignarles un nombre y reordenarlos según convenga.

- ***Comprobación de la sintaxis completa para XML 1.0*** : Los errores de sintaxis de **XML** y **DTD** se indican mientras escribe, y aparecen descripciones detalladas en la Lista de errores.

- ***Fragmentos de código***: El **editor XML** *agrega fragmentos de código dinámicamente generados basados en los* ***esquemas XML***. Al *presionar* la tecla TAB después del nombre del elemento se rellenan automáticamente los atributos necesarios y el contenido secundario. Asimismo se proporcionan muchos fragmentos de **código XML**, incluido un fragmento de código para generar nuevos fragmentos de código.

1.6.4.3 Generación, Pruebas e Implementación en Visual Studio 2005

Microsoft* Build Engine *(MSBuild) es la nueva plataforma de generación de ***Microsoft*** y **Visual Studio**. **MSBuild** presenta un nuevo formato de archivo de proyecto basado en **código XML** que es sencillo de comprender, fácil de ampliar y totalmente compatible con ***Microsoft***.

El formato de archivo de proyecto de **MSBuild** *permite a los desarrolladores describir completamente qué elementos se han de generar y cómo han de generarse con distintas plataformas y configuraciones*. Además, el formato de archivo de proyecto permite a los desarrolladores crear reglas reutilizables que se pueden factorizar en archivos independientes para que las generaciones se ejecuten de igual forma en los distintos proyectos del producto.

Los proyectos de **Visual Studio** ahora se almacenan en el formato de archivo de proyecto de **MSBuild**, lo que proporciona la posibilidad de personalizar el proceso de generación de **Visual Studio**.

MSBuild *es completamente transparente* en el modo de procesar y generar el software, permitiendo a los desarrolladores generar proyectos en equipos sin **Visual Studio**.

En lo que se refiere a las pruebas, esta versión presenta las siguientes características:

- ***Diseñador de clases***: El ***Diseñador de clases*** *permite visualizar sistemas y aplicaciones*. Aunque el usuario diseña los tipos de clase, los miembros y los métodos, el ***Diseñador de clases*** *genera el código fuente correspondiente*.

- ***Herramienta de prueba de objetos***: Cuando se codifica, se puede utilizar la ***Herramienta de prueba*** de objetos para probar rápidamente **.NET Framework** o las aplicaciones de ***Visual J#.***

En lo referente a la implementación esta versión presenta las siguientes características:

- ***Implementación ClickOnce*** : *Permite implementar aplicaciones de actualización automática para* ***Windows*** *que pueden instalarse y ejecutarse tan fácilmente como las aplicaciones Web*. También pueden implementarse aplicaciones de línea de comandos y de cliente de ***Windows***.

- ***Requisitos previos de las secuencias de inicio***: Ahora se pueden incluir componentes del sistema necesarios, tales como la versión en tiempo de ejecución de **.NET Framework**, como parte de un proyecto de implementación o de la ***Implementación ClickOnce***.

- ***Implementación de Windows Installer:*** Las mejoras en los proyectos de *instalación e implementación* incluyen la posibilidad de elegir entre una instalación por usuario o por equipo, compatibilidad con implementaciones de 64 bits e implementación en servidores Web que alojen varios sitios Web.

1.6.4.4 Depuración en Visual Studio 2005

El depurador de **Visual Studio 2005** se ha optimizado mediante la inclusión de las funciones siguientes:

- ***Editar y continuar en Visual Basic y Visual C#***: Es posible cambiar el código de ***Visual Basic*** y de ***C#*** mientras depura y se sigue ejecutando la aplicación simultáneamente. Esta función mejora la productividad al permitir

corregir rápidamente los errores, probar la nueva funcionalidad y modificar la funcionalidad existente.

- ***Depuración remota más segura con una configuración más sencilla:*** Es posible configurar la ***depuración remota*** copiando una sola aplicación ejecutable en el equipo remoto sin instrucciones de configuración ni registros complejos. La ***depuración remota*** es ahora más segura y sólida. Además, ahora puede depurar aplicaciones administradas y no administradas de 64 bits.

- ***Depuración de Sólo mi código***: Esta función le permite centrarse únicamente en el código que ha escrito y omitir el código que no le interesa.

- ***Puntos de seguimiento e interfaz de usuario de puntos de interrupción optimizada***: Proporcionan una nueva forma de utilizar los ***puntos de interrupción*** para realizar una acción personalizada. Permiten imprimir un mensaje o ejecutar una macro de automatización de **Visual Studio** y determinar si el depurador se ha de interrumpir o continuar al alcanzar un punto de seguimiento. La ***interfaz de usuario*** se ha optimizado para facilitar y agilizar la configuración de todos los ***puntos de interrupción***.

- ***Mejores herramientas para la depuración de multiprocesos***: La **nueva ventana Procesos** *muestra todos los procesos a los que se está asociado para depurar*. Los ***filtros de punto de interrupción*** permiten asociar un punto de interrupción a procesos, subprocesos y equipos específicos. El ***cuadro de diálogo Asociar al proceso*** se ha simplificado para facilitar su uso y la información asociada sobre los procesos se ha trasladado a la **ventana Procesos**.

1.6.4.5 Proyectos, Soluciones y Elementos en Visual Studio 2005

En esta versión están disponibles ciertas funciones referentes a proyectos, soluciones y elementos, enseguida se describen las mas sobresalientes:

- ***Proyectos temporales***: Con los proyectos temporales, se puede crear un proyecto y experimentar con él sin tener que guardarlo.

- ***Proyectos independientes***: Si una solución sólo contiene un proyecto, no se verá la solución en el **Explorador de soluciones** ni se verán los comandos que se apliquen a las soluciones en el entorno de desarrollo integrado, aunque se crearán los archivos de solución.

- ***Asistente de conversión de Visual Studio***: Las soluciones o proyectos que se crearon o actualizaron en **Visual Studio .NET 2002** o **Visual Studio .NET 2003** se deben convertir al formato utilizado por esta versión de **Visual Studio** para poder trabajar con ellos en esta versión de **Visual Studio**. Las soluciones o proyectos convertidos ya no son compatibles con Visual **Studio .NET 2002** ni **Visual Studio .NET 2003**. Mediante el asistente, se puede crear una copia de seguridad de la solución o proyecto antes de convertirlo.

- **Starter Kits**: Un ***Starter Kit*** *es esencialmente una plantilla de proyecto mejorada que se puede compartir con otros miembros de la comunidad*. Incluye ejemplos de código que compilan, documentación y otros recursos útiles para aprender nuevas herramientas y técnicas de programación mientras genera aplicaciones para la vida real.

- ***Plantillas personalizadas de proyectos y elementos de proyectos***: Ahora puede ser creada fácilmente una ***plantilla personalizada*** para proyectos o elementos de proyectos o también modificar las ***plantillas de proyectos*** y

elementos de proyectos existentes para satisfacer mejor sus necesidades de desarrollo.

- ***Visual Studio Team System***: *Es un conjunto de herramientas que ofrece soporte a todo el ciclo de vida de un proyecto*. Se compone de dos partes principales: un servidor **(*Visual Studio Team Foundation Server*)** y un cliente **(*Visual Studio Team Edition*)**, que partiendo de **Visual Studio 2005 Professional** añade herramientas adicionales para: *testing, profiling, modelage en UML, análisis estático y dinámico de código entre otros*.

CAPÍTULO 2

ASP y la nueva tecnología .NET

2.1 Importancia de la nueva tecnología .NET

Microsoft ha llevado a cabo una estrategia para la construcción de una nueva tecnología orientada a crear *aplicaciones web distribuidas* que aprovechen al máximo las posibilidades otorgadas por Internet. Esta tecnología denominada **.NET**, incluye una la versión ***Visual Basic .NET*** y el nuevo lenguaje denominado ***C#***, entre otras tecnologías como **ASP .NET**, que tiene la finalidad de reemplazar a **ASP**.

La **tecnología .NET** fue dada a conocer en el mes de Julio del 2000 en la ***Conferencia de Desarrolladores Profesionales de Microsoft***. Esta nueva tecnología estuvo en desarrollo por mas de dos años, bajo un arduo trabajo. Varios aspectos se han tratado en relación al impacto que esta tecnología causa en comparación a la tecnología empleada en años anteriores, por lo que incluso ha sido denominada como la **"Próxima Generación de los Servicios de Windows"** (***Next Generation Windows Services).***

Sin embargo, esta novedosa tecnología no seria posible sin la secuencia evolutiva generada a lo largo de los años, originada principalmente por la necesidad de desarrollar aplicaciones ***cliente-servidor*** cada vez más dinámicas.

2.2 Framework .NET

El ambiente de trabajo generado por la **tecnología .NET**, es un modo **multilenguaje**, que brinda al desarrollador un entorno de desarrollo que le permite disponer de una gran cantidad de herramientas y tecnologías que facilitan la creación de aplicaciones Web potentes y distribuidas, originando a su vez un ambiente multiplataforma de suma utilidad para los desarrolladores.

En el **entorno .NET** o **.NET Framework** destacan los siguientes componentes:

a) **Common Language Runtime (CLR).**
b) **Biblioteca de Clases.**

Common Language Runtime.- consiste en un conjunto de características comunes, que deben cumplir todos los lenguajes de la plataforma para hacer posible su integración. *Es el agente del sistema que se encarga de la ejecución y administración del código*.

Es responsable también de los servicios básicos del sistema tales como la ***administración de memoria***, la ***creación de subprocesos***, ***control de errores*** y la ***seguridad***. En pocas palabras **CLR** responde la cuestión de cómo es posible conseguir que lenguajes de distinta naturaleza y sintaxis se "entiendan" entre si.

Biblioteca de Clases.- La Biblioteca de Clases *es una colección de tipos orientados a objetos*, los cuales pueden ser utilizados para desarrollar cualquier aplicación, componente o servicio.

2.2.1 .NET Framework SDK

Mejor conocido como el **Kit de desarrollo para .NET Framework *(Software Development Kit o SDK)***, que contiene la **plataforma .NET**, además de un conjunto de herramientas independientes donde algunas de ellas funcionan de modo comando (*es decir, mediante la ventana de* ***MS-DOS***) y otras en modo gráfico.

Los elementos indispensables para desarrollar aplicaciones para **.NET** están contenidos en este conjunto de herramientas.

2.3 Internet Information Server (IIS)

Para albergar las **páginas ASP .NET** creadas junto con las aplicaciones es necesario instalar un *Servidor Web*, aunque en realidad el *Servidor Web* no será quien ejecute las páginas Web sino la **plataforma .NET**.

El componente **IIS** convertirá el equipo utilizado de manera automática en un ***Servidor Web*** que soporte aplicaciones **ASP .NET**.

La función principal del ***Servidor Web*** **IIS** es la de *administrar las aplicaciones Web y comunicarse con los navegadores cliente* mediante el Protocolo **HTTP**, además el **IIS** ofrece servicios de protocolo como **Transferencia de Archivos *(FTP)***, **Servicio de Correo Electrónico *(SMTP)*** y **Servicio de Noticias *(NNTP).***

Para la versión **ASP 3.0**, era suficiente contar con el **IIS** instalado en un Servidor Web, debido a que el **IIS** interpretaba directamente el **código ASP** enviándolo posteriormente al cliente. En **ASP .NET**, debe ser instalado **.NET Framework** para procesar el **código ASP .NET**.

Al ser solicitada una **página ASP .NET**, la cual cuenta con la extensión ***.aspx***, el ***Servidor Web de IIS*** *envía una solicitud* a **.NET Framework**, el cual *compila y ejecuta el código*, devolviéndolo al **IIS** para que éste lo envié al cliente.

2.4 Características de ASP.NET

Uno de los objetivos más importantes considerados para la creación de la **tecnología .NET** es el de otorgar un *gran rendimiento*. Es por ello que se plantean las principales características que combina **.NET** para convertirse en una de las mejores opciones para el desarrollo de aplicaciones.

A continuación se mencionan las características más sobresalientes de la nueva **tecnología ASP .NET**:

Eficiencia.- Para que esta tecnología tenga éxito al ser utilizada por las empresas, el personal especializado debe estar capacitado para migrar las aplicaciones y así evitar un rendimiento deficiente al ejecutar el código, ya que el ***Common Language Runtime (CLR)*** ejecuta el código de manera especial.

El **CLR** *compila en un lugar particular todos los códigos de aplicaciones en códigos naturales de máquina*, esto con la finalidad de asegurar un ***rendimiento óptimo***. La conversión mencionada puede realizarse en el momento en que se ejecuta la aplicación o bien cuando se instala la aplicación por primera vez.

La **compilación** realizada en el proceso hace uso de todas las características del microprocesador disponibles en diferentes plataformas logrando así, superar a las ***aplicaciones tradicionales Windows***.

Soporte de Lenguajes.- Esta característica es quizás considerada como una de las más importantes para beneficio de los desarrolladores, debido que **ASP .NET** *ofrece la posibilidad de escribir código en diversos lenguajes*. **ASP .NET** soporta la programación en lenguajes como, ***Visual Básic .NET*** y ***C#*** (*el nuevo lenguaje de* ***Microsoft***).

Código y contenido por separado.- En la versión tradicional **ASP**, se presenta el inconveniente de tener que crear la **interfaz de usuario** y el **código ASP** *de manera conjunta*, es decir, se realiza un combinación de código de imágenes, botones y tablas de **HTML** y secciones en ***VBScript*** o ***JScript*** poco prácticas. *La versión actual* **ASP .NET** *soluciona el problema separando la interfaz de usuario con el código*.

Compatibilidad con Navegadores.- **ASP .NET** *permite la creación de paginas Web que funcionan correctamente en todos los navegadores*. Esta mejora se incluye gracias a los controles de servidor incluidos en la nueva versión. En el momento en el que un control es procesado, automáticamente se realiza un chequeo del tipo de navegador que se está ejecutando y se genera una pagina adecuada para el navegador.

Código Compilado.- La **compilación** en la nueva versión **ASP .NET** *no interpreta el código como lo hace la versión* **ASP***.* Ahora dentro del entorno **New Generation Windows Services *(NGWS)***, el código es compilado ***just–in–time***, aumentando el rendimiento mediante el soporte nativo y servicios de caché.

Controles de Servidor.- Dentro de los aspectos importantes de **ASP .NET**, se encuentra su **librería de clases**, *la cual brinda al programador una herramienta para creación de aplicaciones multiplataforma*, permitiendo también un considerable ahorro en las líneas de código empleadas.

2.5 Ventajas de ASP.NET

ASP. NET ofrece varias ventajas importantes referentes a los modelos de programación Web empleados antes del surgimiento de esta nueva tecnología:

1. **Mejor rendimiento**. A diferencia de sus predecesores, **ASP. NET** es capaz de aprovechar las ventajas del enlace anticipado*, la compilación* ***just-in-time****, la optimización nativa y los servicios de caché* desde el primer momento. Por lo tanto existe un incremento importante del rendimiento desde el inicio, donde se comienza a escribir el código.

2. **Compatibilidad con herramientas de primer nivel**. El ambiente de trabajo se complementa con un diseñador y una caja de herramientas muy completos.

Los controles de servidor de arrastrar y colocar, y la implementación automática ejemplifican la eficacia de las herramientas empleadas.

3. **Eficacia y flexibilidad**. La eficacia y flexibilidad de la plataforma **ASP .NET** se encuentra disponible para los programadores de aplicaciones Web, debido a que la **biblioteca de clases**, **la mensajeria** y **las soluciones de acceso a datos** se encuentran accesibles desde el Web de manera uniforme. Tomando en cuenta que **ASP .NET** es independiente del lenguaje, es posible elegir el lenguaje que mejor se adapte a la aplicación por desarrollar o incluso implementar varios lenguajes.

4. **Simplicidad**. **ASP .NET** *facilita la realización de tareas como el envió de formularios, la autenticación del cliente y la implementación y configuración de sitios*. El ambiente de trabajo de **ASP .NET** permite generar interfaces de usuario, que separan la lógica de aplicación del código de presentación, además de controlar eventos de forma sencilla a través del modelo de procesamiento de formularios de tipo ***Visual Basic***. **CLR** *simplifica la programación con servicios de código administrado como el recuento de referencia automático y el recolector de elementos no utilizados*

5. **Facilidad de uso**. **ASP .NET** utiliza una ***configuración jerárquica*** basada en texto, lo cual simplifica la aplicación de la configuración al entorno del servidor y a las aplicaciones Web. La **información de configuración** *es almacenada como texto sin formato*, por lo que se puede aplicar la nueva configuración sin la ayuda de herramientas de administración local. Una **aplicación ASP.NET** se implementa en un servidor de forma sencilla mediante una copia de los archivos necesarios, por lo que no se requiere reinicio del servidor ni para implementar o reemplazar el código compilado en ejecución.

6. **Escalabilidad y disponibilidad**. **ASP .NET** fue diseñado tomando en cuenta la **escalabilidad** con características específicamente a medida, con el fin de

mejorar el **rendimiento** en entornos agrupados y de múltiples procesadores. **ASP.NET** también *controla y administra los procesos* de tal manera que si uno no se comporta adecuadamente, se pueda crear un nuevo proceso, ayudando a mantener la aplicación disponible.

7. **Posibilidad de personalización y extensibilidad**. **ASP.NET** cuenta con una arquitectura que permite a los programadores insertar código en el nivel adecuado. Y es posible *extender o sustituir un subcomponente del motor de tiempo de ejecución con un componente escrito personalizado*.

8. **Seguridad**. Con la configuración por aplicación y la ***autentificación de Windows***, es posible obtener una seguridad completa de las aplicaciones.

2.6 Diferencia entre ASP 3.0 y ASP .NET

La diferencia entre **ASP 3.0 y ASP .NET** se refiere a la ejecución del código en el servidor en base a los siguientes puntos:

1. *Con **ASP .NET** el código es compilado en clases ejecutables, mientras que en **ASP 3.0** necesita ser interpretado.*

2. *Con **ASP 3.0** la mayoría del código del lado del servidor tiene que ser interpretado por el servidor Web, a menos que éste sea interceptado.*

A pesar de ser una tecnología relativamente nueva, **Active Server Pages** ha logrado establecer un estándar en la creación de ***páginas Web dinámicas***. **ASP .NET** pretende solucionar el principal problema presente en las páginas de **ASP 3.0**.

Este problema se refiere al mantenimiento en el cual las aplicaciones ***cliente-servidor*** de **ASP 3.0**, son difíciles de mantener, debido a que el **código ASP** *es*

mezclado con la interfaz de usuario que ocasiona se pierda tiempo en las actualizaciones del código al no poder trabajar simplemente con el núcleo del código.

ASP .NET *soluciona este problema al permitir separar la interfaz del código*, es decir, el entorno de ésta nueva versión brinda una extensa cantidad de controles predefinidos que permiten crear aplicaciones con simplemente escribir una cantidad mínima de líneas de código.

La **versión 3.0 de ASP** brindó un panorama claro para el futuro del desarrollo de software basado en aplicaciones Web. ***Microsoft*** introdujo **ASP .NET la nueva versión de ASP**, la cual contiene innovaciones que permiten separar fácilmente el código central de una aplicación así como su presentación.

Se afirma que **ASP .NET** *no es una mejora de* **ASP**, si no que realmente es un nuevo producto que permite incluso la misma manera de desarrollar a la que se estaba acostumbrado con **ASP**.

CAPÍTULO 3

Especificaciones y Aplicaciones con ASP.NET

3.1 Modelo de Desarrollo para Aplicaciones ASP.NET

Las **aplicaciones ASP .NET** son desarrolladas de manera similar a sus versiones anteriores. El proceso para **aplicaciones ASP** consiste en *Editar la página, Probar la página, posiblemente regresar a editar la página nuevamente* y así sucesivamente como se muestra en la **Figura 3.1**.

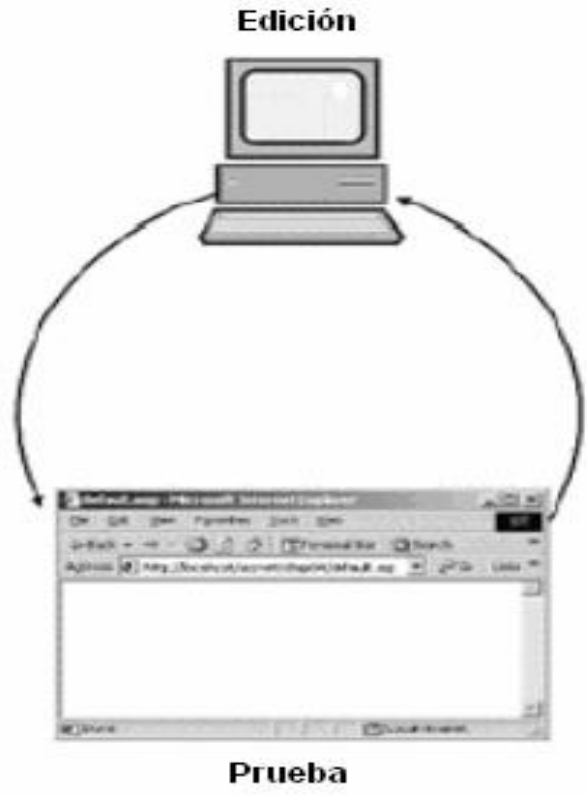


Figura 3.1. Ciclo de desarrollo ASP.

Para el desarrollo **ASP .NET**, el proceso es exactamente el mismo, pero es incluida una actividad diferente. El proceso consiste en ***Editar-Compilar-Probar*** mostrado en la **Figura 3.2** .

La parte en la cual se lleva a cabo la **compilación**, *resulta transparente* para el desarrollador. Es decir, si una aplicación que es ejecutada no ha sido compilada, ésta se ***compila automáticamente***.

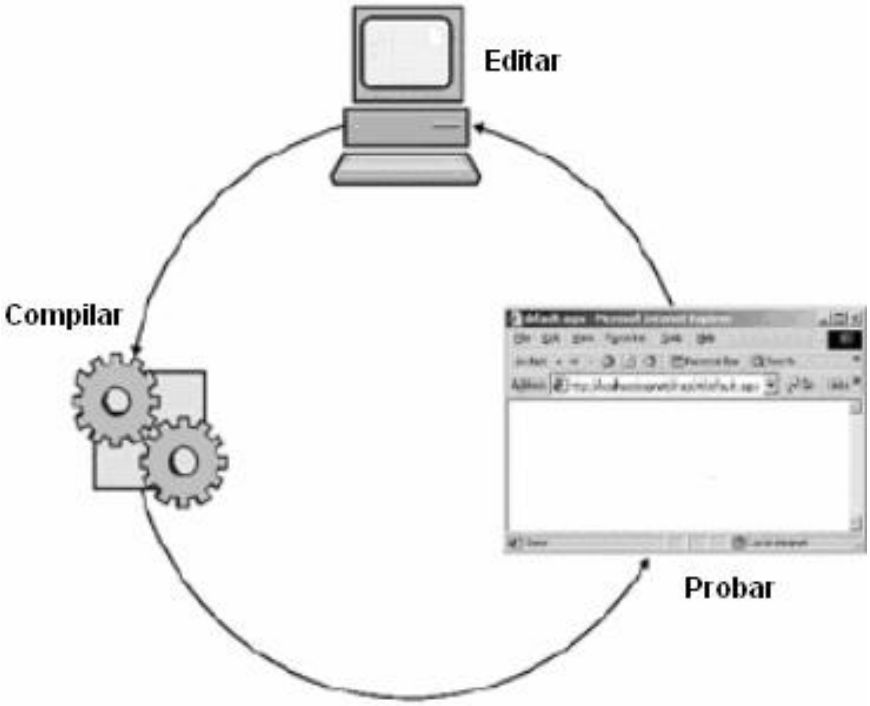


Figura 3.2. Ciclo de desarrollo ASP .NET.

Bajo el contexto anterior, se demuestran las mejoras originadas con el cambio de **ASP** a **ASP .NET**, relacionadas con la **mejora en el rendimiento** y garantizando que los errores fueron detectados desde la primera compilación, en lugar de ser detectados sólo cuando el código está siendo ejecutado.

3.2 Creación de un proyecto Web ASP .NET en Visual Studio .NET

Una vez abierto **Visual Studio .NET**, la secuencia de pasos para crear un nuevo proyecto mas comúnmente utilizada es dando *clic* en el ***menú Archivo*** y después dar un *clic* sobre ***Nuevo*** **y** ***Proyecto***. Al realizar estos pasos se desplegará el cuadro de diálogo que se muestra en la **Figura 3.3**.

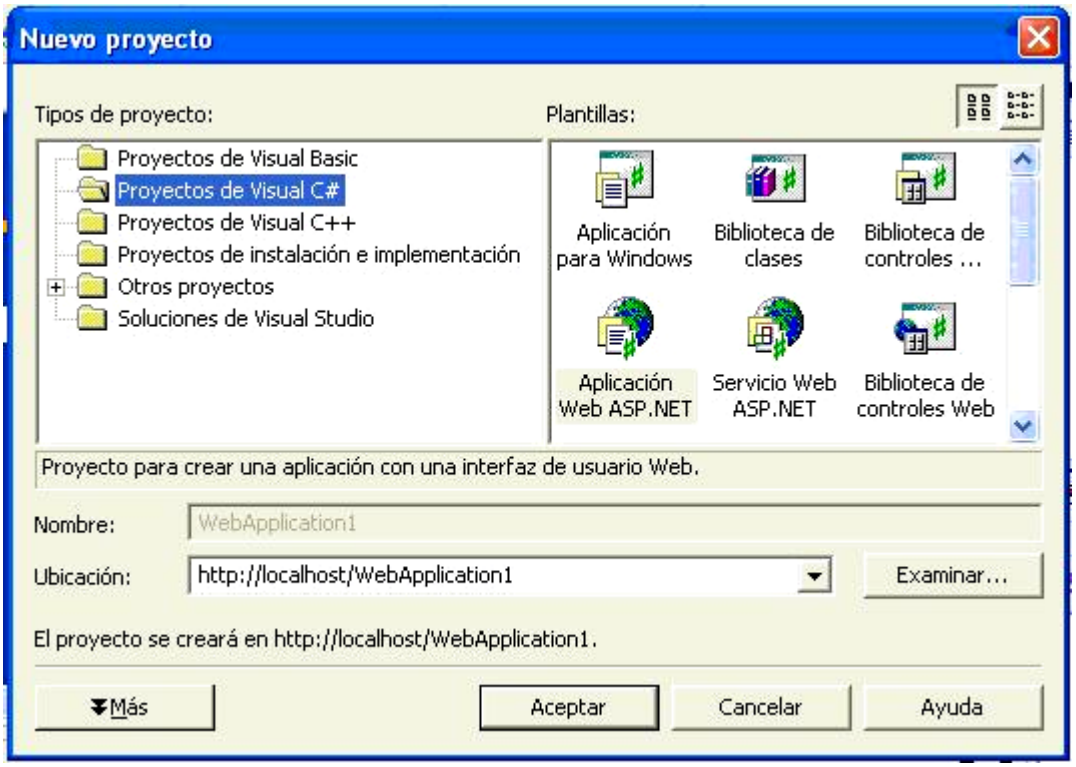


Figura 3.3. Cuadro de diálogo Nuevo Proyecto.

En el cuadro de diálogo mostrado anteriormente es posible elegir un gran número de tipos especiales de *proyectos*, incluso es posible seleccionar el lenguaje deseado para desarrollar la aplicación.

El primer cuadro de texto de la **Figura 3.3**, permite especificar el *nombre que se asignará al proyecto* y en el siguiente titulado ***Ubicación*** se puede cambiar el fólder donde estará ubicado o la **URL** del Servidor Web local.

El proyecto mencionado se localiza en un **directorio virtual** en la raíz del directorio Web, debido a que el tipo de proyecto seleccionado fue una **Aplicación Web ASP .NET**. Una vez seleccionado el tipo de proyecto dando *clic* sobre el *botón aceptar*, **Visual Studio .NET** procede a crear la **aplicación ASP .NET** en el lugar y con el nombre indicado.

Posteriormente de forma *automática* se crean una serie de archivos dentro de la misma aplicación, los cuales pueden ser observados en el **"Solution Explorer"** (**Figura 3.4**) de **Visual Studio .NET**.

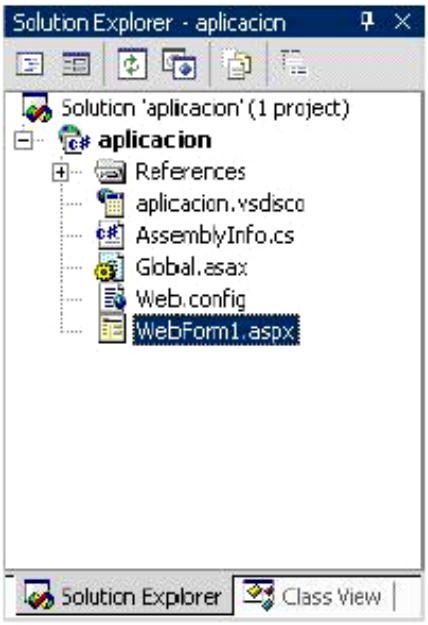


Figura 3.4. Solution Explorer.

3.2.1 Estructura de una aplicación ASP .NET

Cuando es creada una **aplicación ASP .NET** se crea una *estructura de archivos*, tal como se muestra en la **Figura 3.5**.

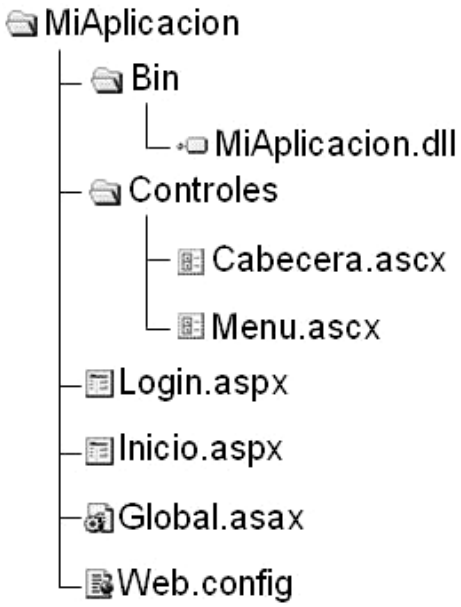


Figura 3.5. Estructura de archivos de una aplicación Web ASP .NET.

El código del servidor se ensambla en un **archivo .dll**, que se encuentra en la *carpeta* **Bin** de la aplicación Web. Una vez que se realiza la compilación, los archivos de código ***.vb (código Visual Basic) ó .cs (código C Sharp)*** , ya no son *necesarios para ejecutar la aplicación*, pues han sido ensamblados en la **DLL** y es recomendable quitarlos del servidor para impedir que se pueda acceder a su contenido desde el exterior.

Para que una **aplicación Web ASP .NET** funcione de manera adecuada, es necesario tener los siguientes elementos en el servidor:

- **IIS 5.0 o superior**, configurando el *directorio virtual* asociado a la aplicación.
- **.NET Framework**.
- **Archivos .aspx** de las páginas.
- **Archivo ensamblador (DLL)** situado en la carpeta *Bin* de la aplicación Web.
- **Archivo Global.asax**, el cual controla la aplicación durante su ejecución.
- **Archivo Web.config** que establece la configuración de la aplicación.

Cabe mencionar que también pueden aparecer de forma adicional en la **carpeta Web** archivos varios pertenecientes a la aplicación como: **paginas .html**, **paginas .asp**, **documentos**, **imágenes**, etcétera.

Como ya se mencionó, en el apartado de estructura de una **aplicación ASP .NET**, son creados una serie de archivos. El archivo especial **GLOBAL.ASAX** tiene funciones similares a las del archivo **GLOBAL.ASA** de las versiones anteriores de **ASP**, es decir, *su función es definir los eventos propios de la aplicación*.

El archivo **WEB.CONFIG** contiene la configuración de la **aplicación ASP .NET**, y la página **WEBFORM1.ASPX** contiene un ***WebForm*** que podrá ser utilizado en la **aplicación ASP .NET** de acuerdo a las necesidades del usuario.

Utilizando el archivo ***WebForm1.aspx*** se podrá comenzar a desarrollar la **página ASP .NET** añadiendo el código de acuerdo a la función a desarrollar.

Una situación a destacar, en el entorno de desarrollo que presenta el ***WebForm***, son las *líneas que permiten colocar y distribuir cada uno de los componentes agregados de una manera precisa*; el conjunto de líneas que brindan esta posibilidad es denominado **“Grid Layout”** (**Figura 3.6**).

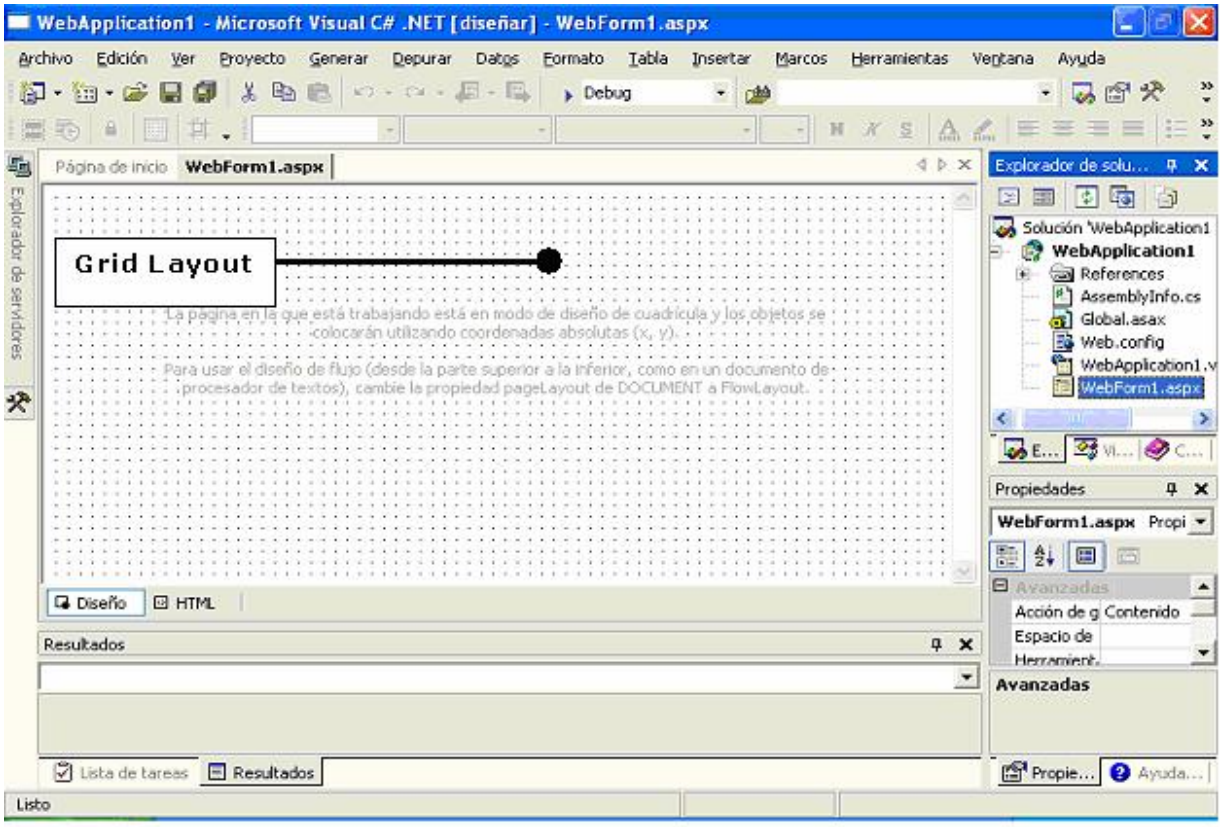


Figura 3.6. Grid Layout.

Para agregar al ***WebForm*** diversos elementos disponibles en la *Caja de herramientas*, es necesario ubicar y desplegar el *Cuadro de herramientas*, el cual se ubica en el costado izquierdo, justo debajo del ***Explorador de Servidores*** (**Figura 3.7**).

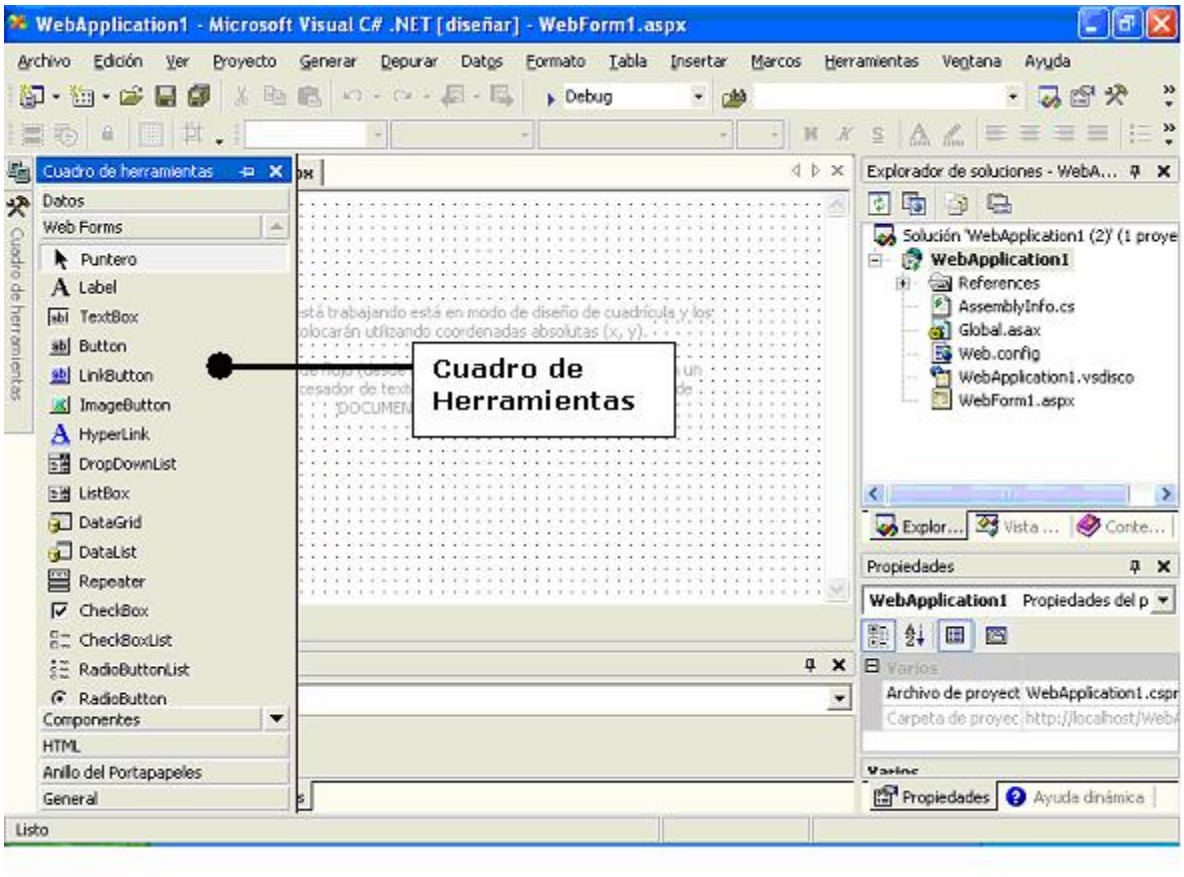


Figura 3.7. Cuadro de Herramientas.

Una vez que son agregados los elementos al ***espacio de trabajo del WebForm***, la manera más sencilla de modificar estos objetos es utilizando la ***ventana de propiedades*** ubicada en la parte inferior derecha (**Figura 3.8**). Cada una de las diferentes propiedades con las que cuenta cada uno de los objetos contenidos en el *Cuadro de Herramientas* puede ser modificada de acuerdo a las necesidades del desarrollador.

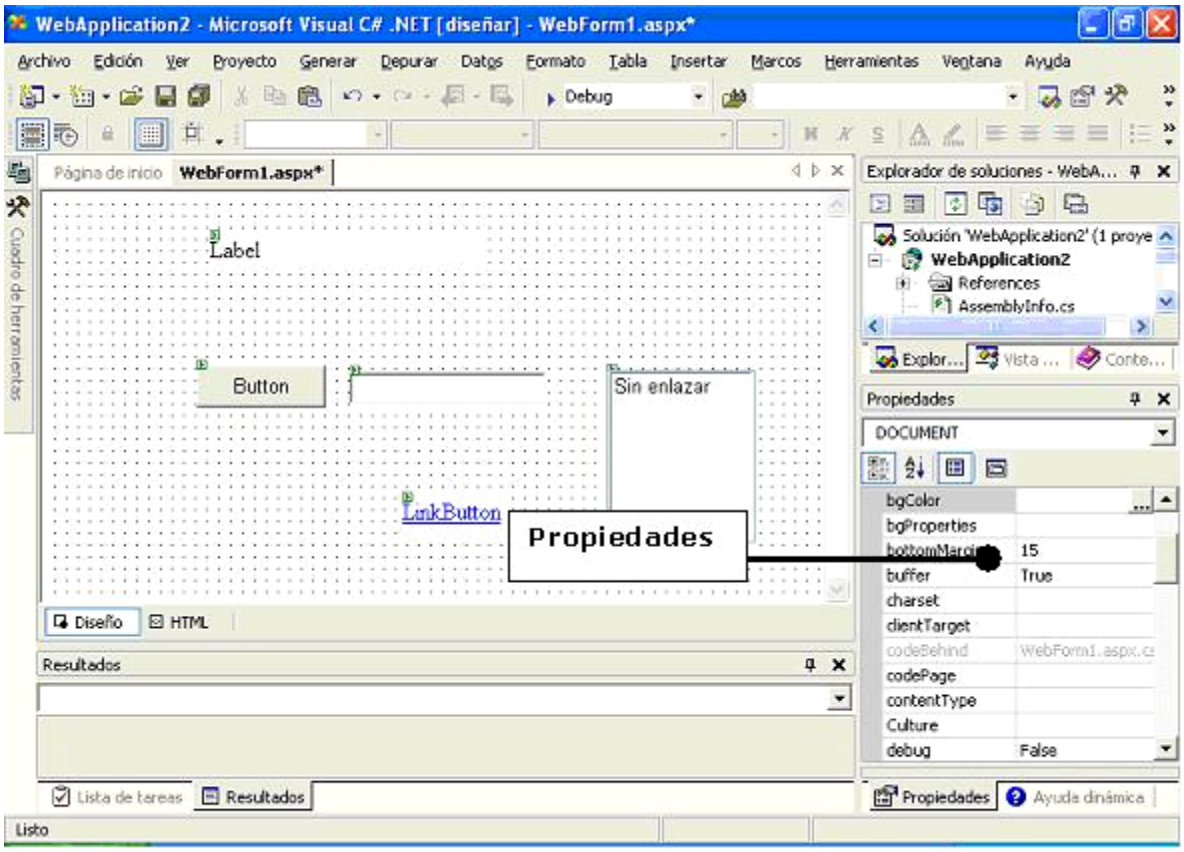


Figura 3.8. Propiedades de Objetos.

La segunda manera de modificar las propiedades de los objetos es mediante la ***modificación del código***. Seleccionando la **pestaña HTML** ubicada en la parte inferior a un costado de la **pestaña titulada Diseño**, es posible ver el **código HTML**.

Sin embargo, a pesar de que el código brinda muchas posibilidades de modificar por medio de funciones especificas, la *ventana propiedades* sigue brindando sus beneficios mientras se observa el **código HTML** como se muestra en la **Figura 3.9**.

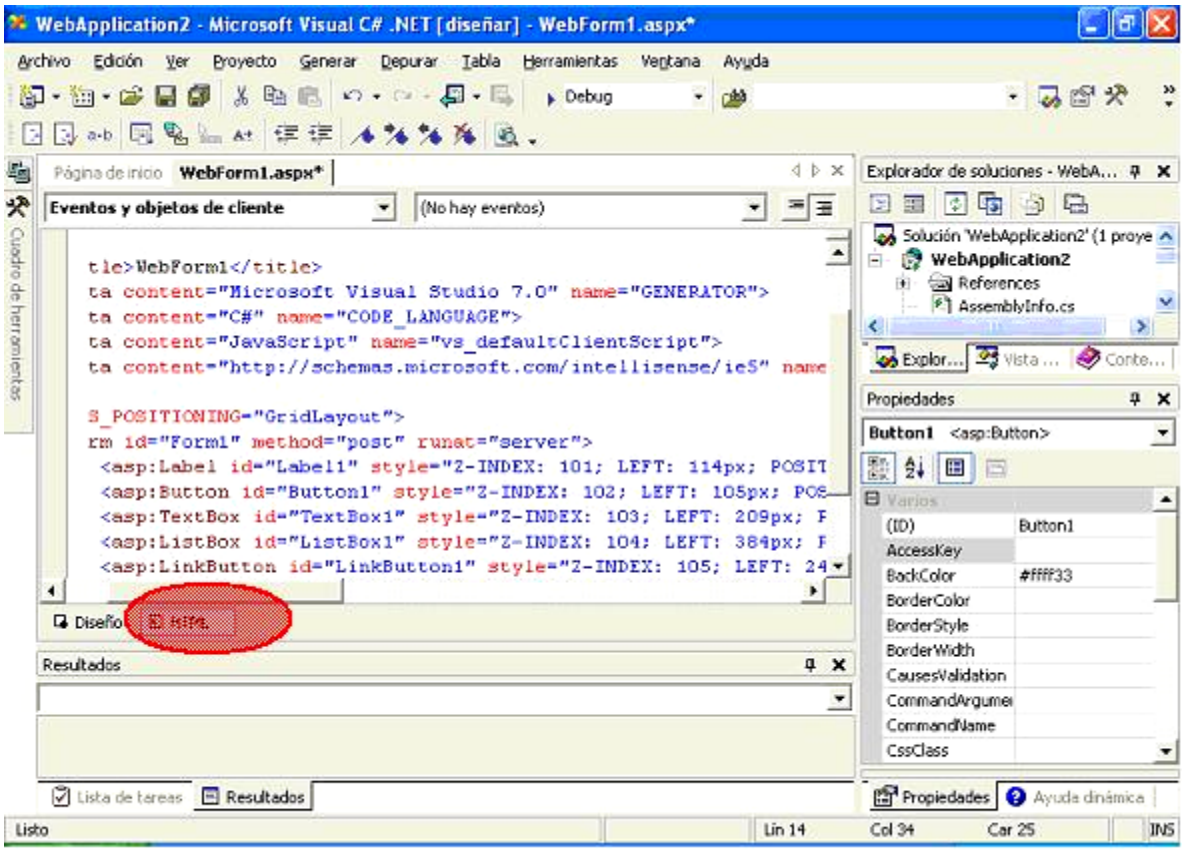


Figura 3.9. Código HTML.

Con sólo seleccionar el elemento deseado dentro del **código HTML** *es posible modificar sus propiedades de la misma forma como se realiza en la vista de diseño*.

Una vez que se han agregado los objetos y las funcionalidades deseadas, se procede a ejecutar en el Servidor Web el diseño desarrollado.

Es necesario **depurar** la aplicación para poder ejecutarla en el *Servidor Web*, por lo que se deberá *presionar* **iniciar** en el ***menú Depurar***, ubicado en la parte superior de la ventana. Cuando es depurado el proyecto, el archivo de la ***clase WebForm*** es compilado para un **archivo DLL** junto con otro archivo ejecutable en el proyecto.

El **archivo ASPX** es copiado al Servidor Web sin ser compilado, por lo que es posible modificar el **archivo ASPX** (*sólo los elementos visuales de la pagina*) sin volver a compilar, ya que el **archivo ASPX** *no es compilado*. Una vez ejecutada la pagina, el **archivo DLL** y el **ASPX** son compilados dentro de una nuevo archivo de clase y es ejecutado. Finalmente, se muestra en el browser la salida de lo que ha sido desarrollado.

3.3 Implementación de una aplicación Web ASP .NET

Después de haber creado y probado la **aplicación Web ASP .NET**, el siguiente paso es implementar dicha aplicación ya terminada (*sin el código fuente*).

En **Visual Studio .NET** el mecanismo para implementar la aplicación es muy similar al usado para crear aplicaciones. Para realizar esto se deberá abrir el *proyecto de la aplicación Web* a distribuir, después, se seguirá la secuencia: ***Archivo/ Agregar Proyecto/ Nuevo Proyecto***, la cual abrirá el *Cuadro de dialogo* **Nuevo Proyecto** que se muestra en la **Figura 3.10**.

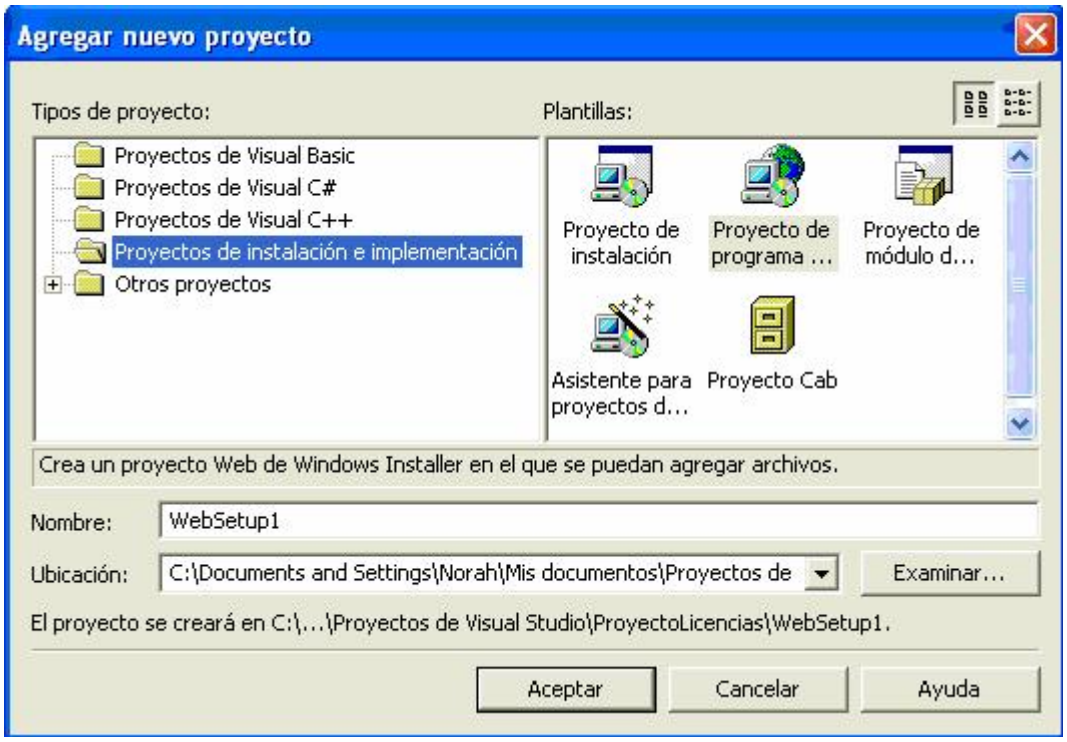


Figura 3.10. Agregar nuevo proyecto.

Con el *Cuadro de diálogo* **Nuevo Proyecto** abierto, en la sección de tipos de proyecto se deberá seleccionar ***Proyectos de Instalación e Implementación***, a continuación en el cuadro de la derecha se seleccionará ***Proyecto de programa de instalación Web*** y se deberá cambiar el nombre del proyecto por alguno otro.

Finalmente, se dará un *clic* sobre el ***botón Aceptar*** para completar el proceso.

El proyecto se agregará a la ventana del **Editor de Archivos de Sistema** como se muestra en la **Figura 3.11**.

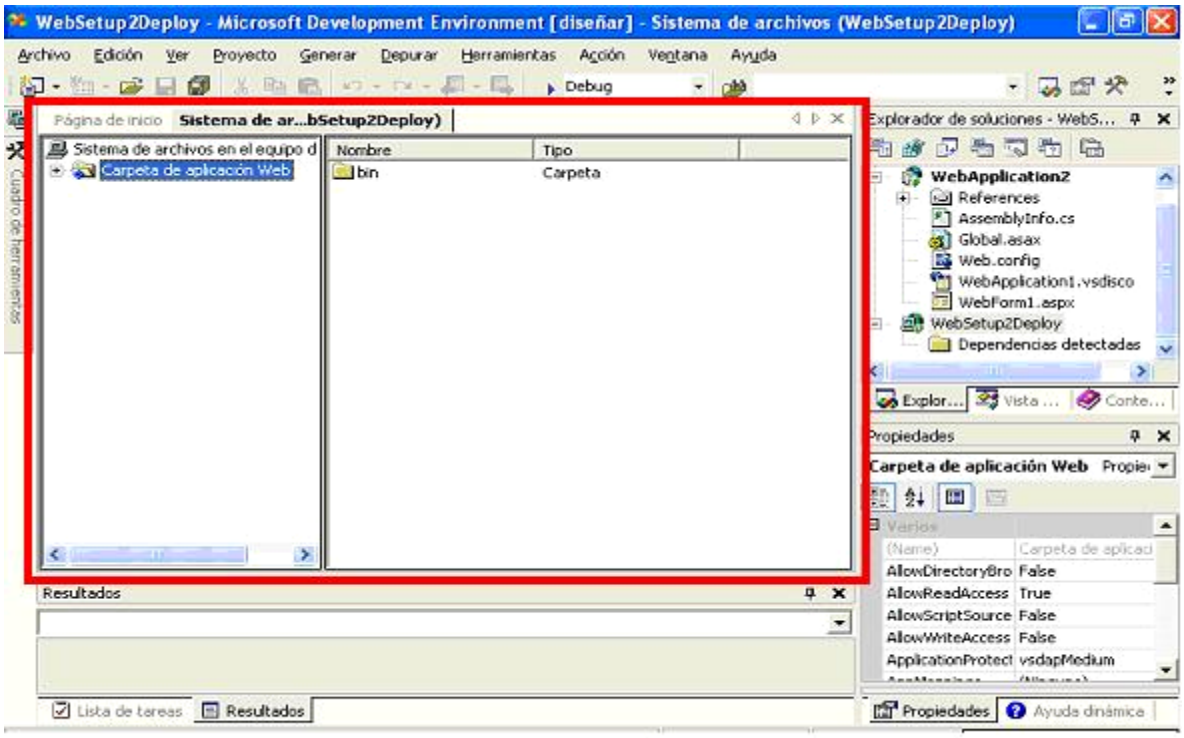


Figura 3.11. Editor de archivos de sistema.

Después de realizar los pasos anteriores, se seleccionará la **Carpeta de Aplicación Web** que aparece en la **Figura 3.11**, y pulsando el *botón derecho* del mouse se seleccionará ***Agregar/ Resultados del proyecto***, con lo que aparecerá el cuadro de diálogo representado por la **Figura 3.12**.

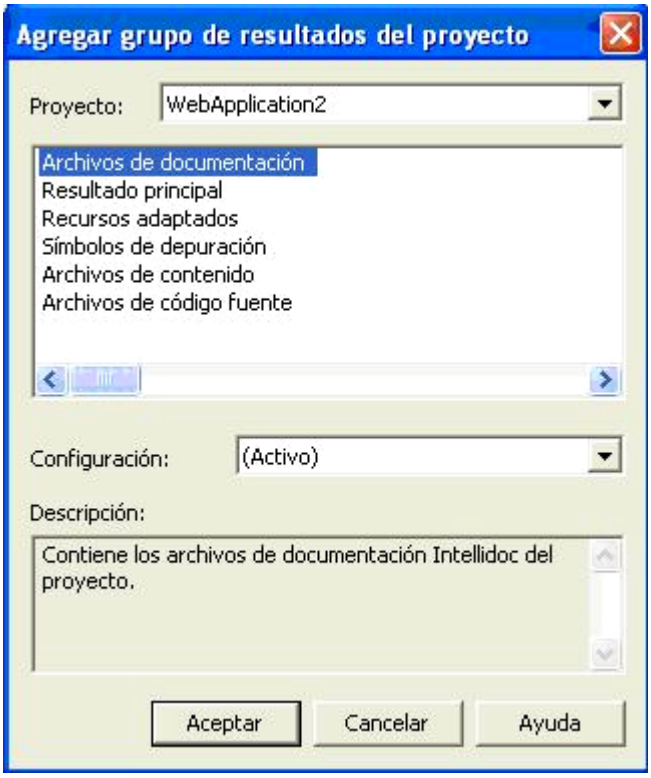


Figura 3.12. Agregar grupo de resultados del proyecto.

Se deberá seleccionar de la lista la opción ***Resultado principal*** dando *clic* en el ***botón Aceptar***. Los archivos de resultado y los archivos de contenido del Proyecto original serán agregados.

Ahora se deberá seleccionar la *Carpeta de aplicación Web* en el **Editor de Archivos de Sistema**, y se seleccionará **Propiedades** de la ***ventana del menú Ver*** para abrir las propiedades de ventana. Se colocará en la opción ***"Virtual Directory***" el nombre de la carpeta, el cual será el **directorio virtual** de la computadora donde se desee instalar la aplicación (**Figura 3.13**).

Figura 3.13. Propiedades.

En la misma ***ventana de propiedades*** de la *carpeta de aplicación Web*, se colocará en la opción ***"DefaultDocument"*** el nombre de la página que se establecerá por "default" para la aplicación.

Enseguida se deberá **depurar** el proyecto con la *opción* ***Depurar*** *del menú*, y al ser depurada la aplicación de manera satisfactoria, un archivo con el nombre del proyecto de **extensión .msi**, es creado en el **directorio *Proyecto de instalación Web***.

Se deberá copiar el archivo con el ***nombre del proyecto*** y ***la extensión .msi*** al Servidor Web (ej: ***c:\inetpub\www.root***), donde será posible implementar la aplicación.

Finalmente se pulsara *doble clic* sobre este archivo en la computadora que se desee ejecutar la instalación.

Una vez que la instalación es completada, es posible ejecutar la aplicación en la computadora donde fue designada introduciendo en el cuadro de direcciones del Internet Explorer : ***http://<nombrecomputadora>/AplicaionImplementada .***

3.4 Creación de una Base de Datos en SQL Server

Un **Sitio Web Dinámico** comienza con la planeación de una *Base de Datos*, ya que es el lugar desde donde la información será *almacenada, modificada y transmitida*.

Existen diferentes plataformas en el mercado para el desarrollo de *Bases de Datos*, tales como: ***Microsoft* Access**, **SQL Server** y **Oracle**. La elección de cual de estas herramientas utilizar para el desarrollo depende del tamaño y seguridad requerida en el lugar donde será implementada la *Base de Datos*.

De acuerdo con el tipo de información, **existen dos tipos de datos**, ***datos relaciones y datos no relacionales***. La diferencia entre estos dos tipos de datos se basa en como se encuentran organizados.

Los ***datos relacionales*** son almacenados en un **Sistema Manejador de Base de Datos Relaciónales** o **RDBMS**, por sus siglas en ingles. La información que es almacenada en *Bases de Datos Relaciónales* se presenta en tablas. Las tablas de las *Bases de Datos* están creadas a base de filas y columnas.

Antes de crear una Base de Datos es necesario tener bien definido en que software será desarrollada, en este caso se mencionará la creación bajo la plataforma **SQL Server 2000.**

Además del software a utilizar es muy importante tener bien definido que información o elementos de datos conformarán la *Base de Datos*, ya que de eso depende el éxito que se obtendrá al final del proceso.

El primer paso es ir creando la *Base de Datos* utilizando el ***"Enterprise Manager"*** de acuerdo con lo siguiente:

1. Seleccionar del *menú* correspondiente **SQL Server** la *herramienta* ***Enterprise Manager***, y expandir el Grupo de **SQL Server** hasta ver una carpeta titulada *Bases de Datos*.

2. Dar *clic derecho* sobre la carpeta titulada **Bases de Datos** y seleccionar ***nueva Base de Datos***. Aparecerá una ventana como la de la **Figura 3.14**.

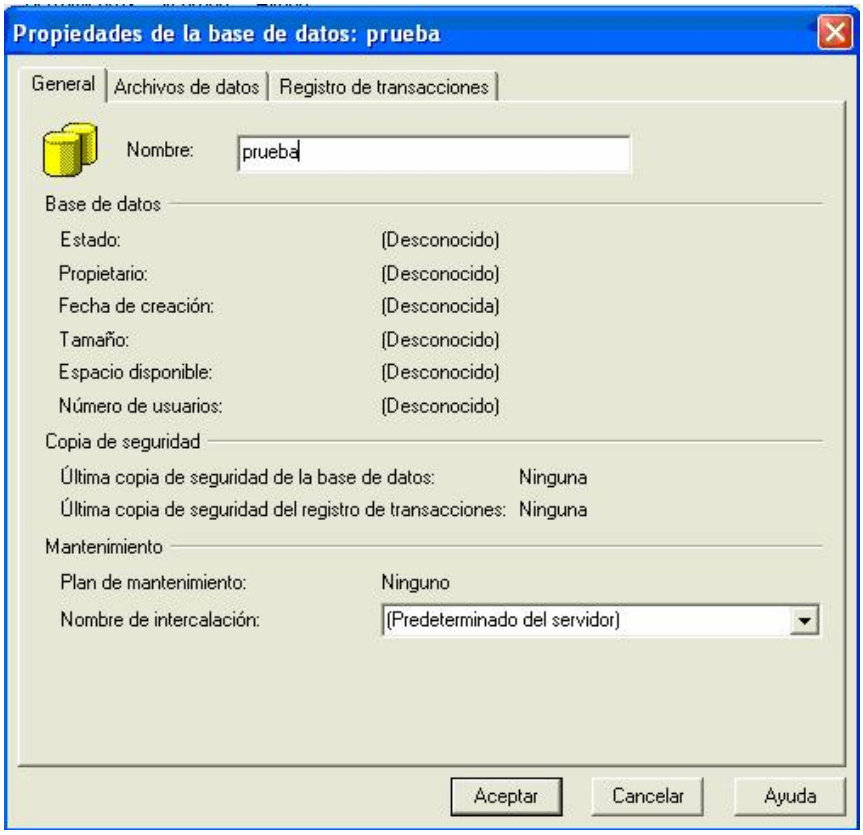


Figura 3.14. Apartado General Ventana propiedades.

3. En el apartado **General** mostrado en la **Figura 3.14**, se introducirá el nombre de la *Base de Datos*. En los otros apartados presentes en la ventana como el de **Archivo de datos** mostrado en la **Figura 3.15**, es posible especificar el

archivo fuente de la *Base de Datos* así como el tamaño máximo, los archivo de transacción ***"log"***, entre otras opciones. En este caso se procede a dejar las opciones por "default".

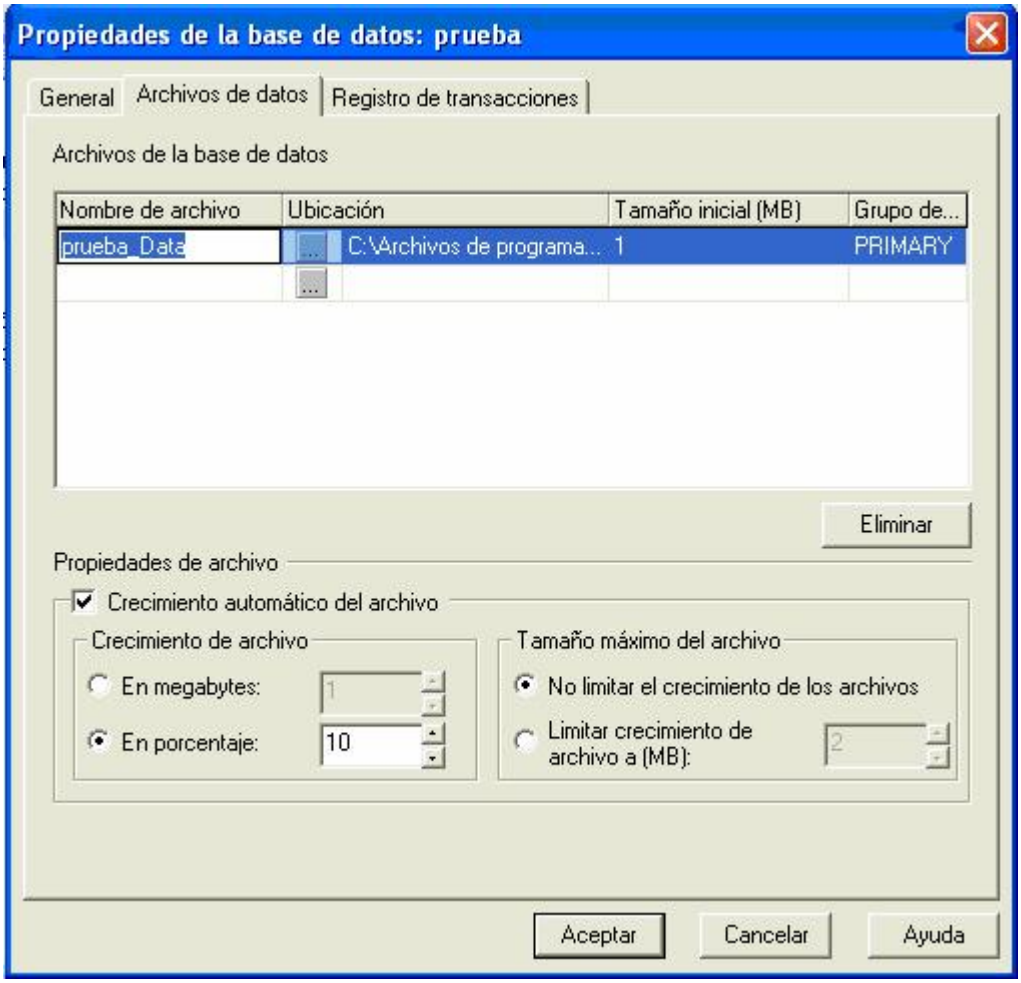


Figura 3.15. Apartado Archivo de Datos.

3.4.1 Creación de tablas en la Base de Datos

Una vez creada la **Base de Datos** se procederá a agregar las ***tablas*** en las cuales se distribuirán los datos. Para crear cada tabla, es necesario seguir los siguientes pasos:

1. Una vez más utilizando el ***Enterprise Manager***, se deberá dar *clic derecho* sobre la carpeta de la *Base de Datos* creada y enseguida se seleccionará ***Nuevo -> Tabla***, y se desplegará la ventada mostrada en la **Figura 3.16**.

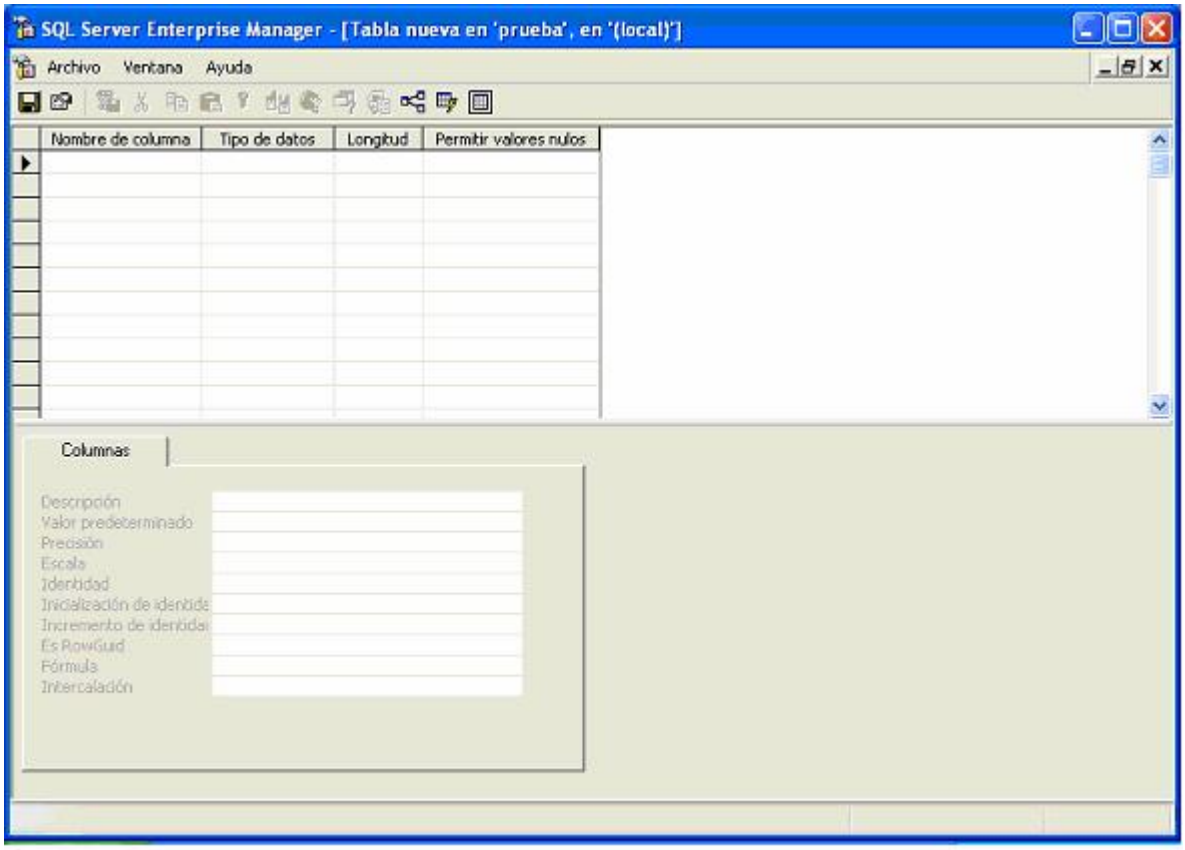


Figura 3.16. Ventana de creación de tablas (sin datos).

2. A continuación en la parte que aparece con **formato Grid**, y en la sección **Nombre de Columna**, se deberá especificar el nombre del campo, así como los valores, **tipo de dato, longitud** y especificación de **valores nulos**. Normalmente en el primer campo especificado se capturará el campo correspondiente a la ***llave o campo clave*** de la tabla.

3. De la manera descrita en el paso anterior, se deberán ir agregando los campos correspondientes a la tabla al igual que sus respectivas propiedades hasta completar la tabla como se muestra en la **Figura 3.17**.

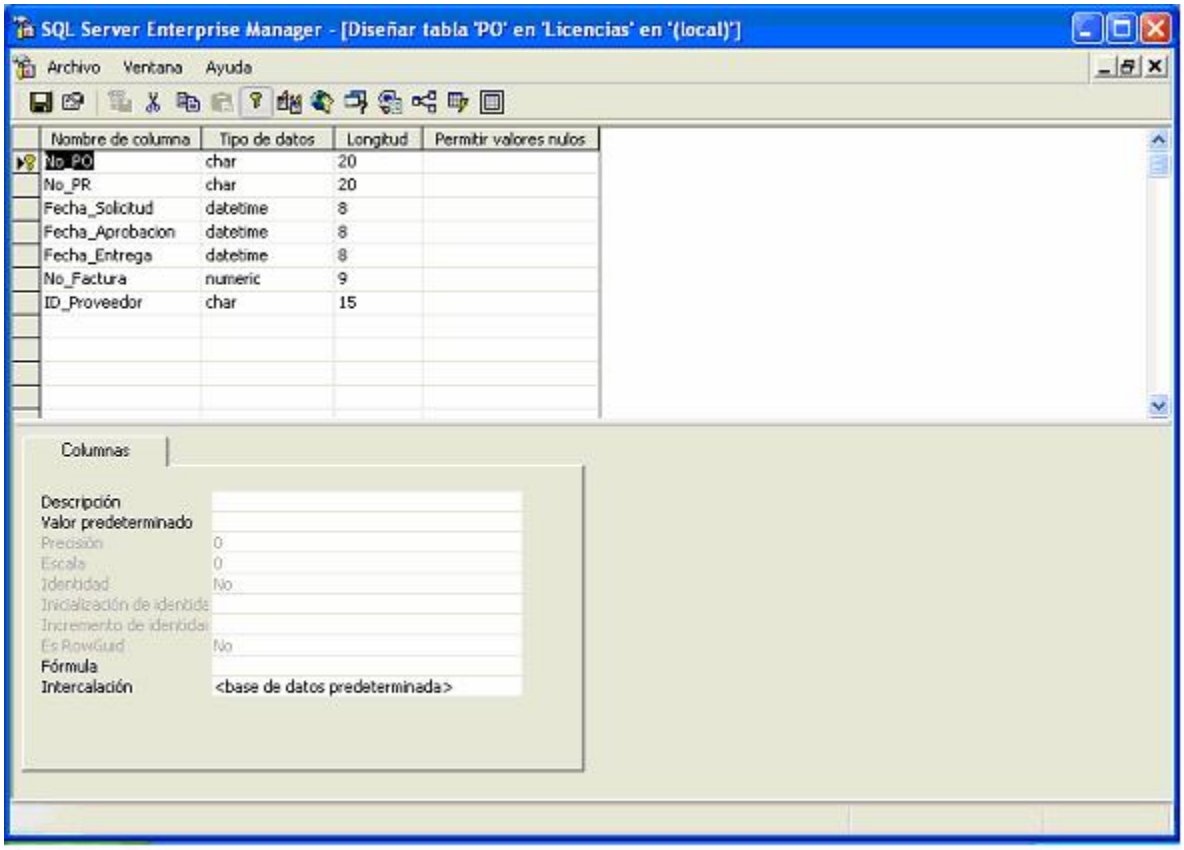


Figura 3.17. Ventana de creación de tablas (con datos).

4. Ahora se procederá a especificar el ***campo llave*** entre algunas otras propiedades mostradas en la **Figura 3.18**, la cual representa el **menú propiedades** presionando con el *botón derecho* del mouse sobre alguno de los campos.

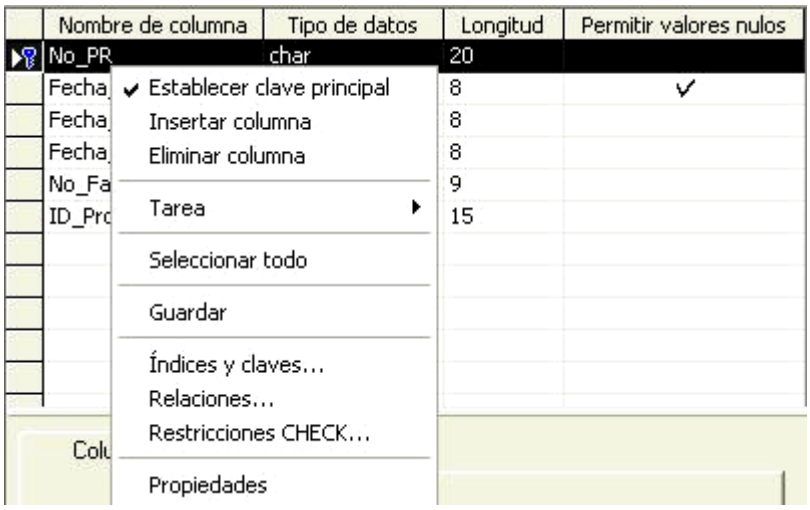


Figura 3.18. Menú Propiedades.

5. Para especificar a detalle las ***propiedades*** y especificar ***los índices y claves*** entre algunas otras funciones, se deberá seleccionar en el **menú propiedades** de la **Figura 3.18** la opción Propiedades, la cual desplegará la pantalla mostrada en la **Figura 3.19** que se presenta a continuación.

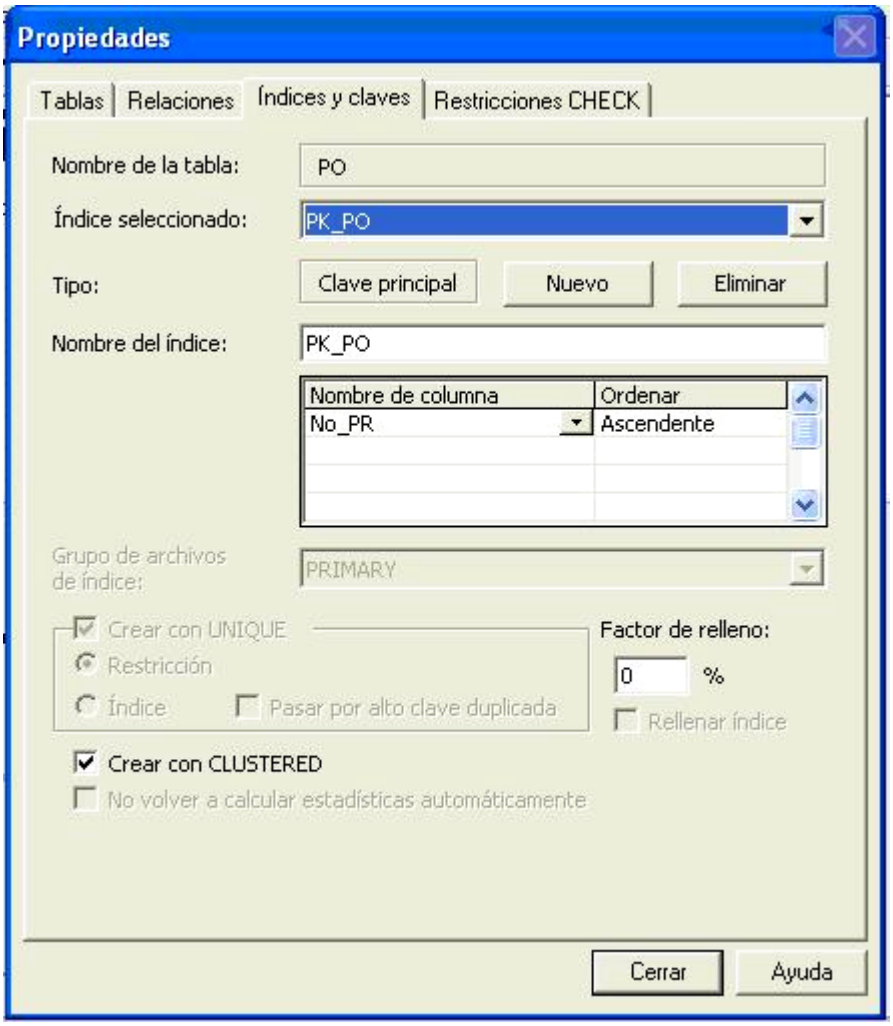


Figura 3.19. Propiedades.

6. Además de especificar los **Índices** y **Claves**, dentro de los aspectos importantes es posible crear las **relaciones entre las tablas ya existentes**. Esto es posible utilizando el **menú Propiedades** mostrado en la **Figura 3.19** seleccionando ahora el apartado titulado **Relaciones**, el cual se muestra a continuación en la **Figura 3.20**.

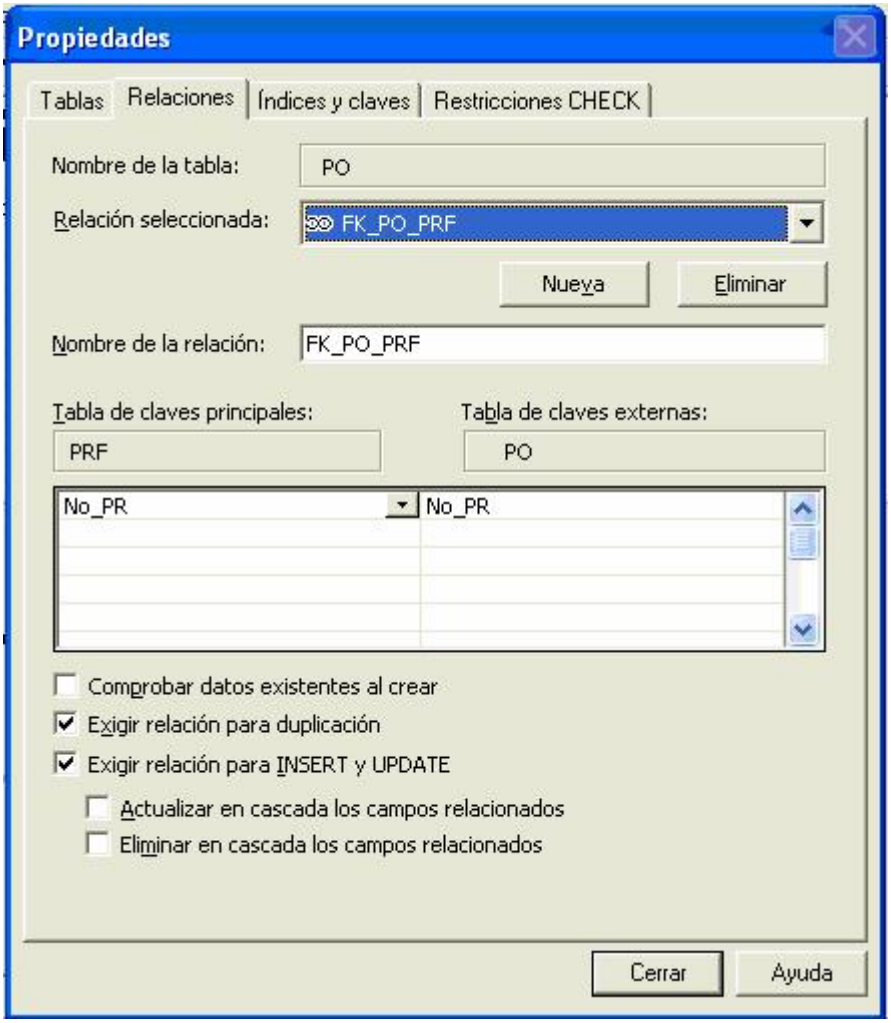


Figura 3.20. Apartado Relaciones.

De esta manera es posible crear la cantidad de tablas que sean requeridas para la *Base de Datos*, especificando las propiedades correspondientes, así como su relación son las demás tablas.

3.5 Utilización del Lenguaje SQL

Después de haber creado una *Base de Datos*, será necesario saber como poder interactuar y modificar los datos que en esta se almacenen. La acciones básicas realizadas con los datos o la información son: ***creación, recuperación, actualización y eliminación.***

Para realizar esto en una ***Base de Datos relacional***, es necesario utilizar el **Structured Query Language o SQL**, como un lenguaje de consulta conformado por diversas sentencias y comandos.

Dentro de los ***comandos de SQL*** más utilizados se encuentran: **INSERT**, **UPDATE**, **DELETE** y **SELECT**, los cuales serán ejemplificados en párrafos siguientes.

3.5.1 Comando INSERT

El **comando SQL** destinado para la inserción de datos en la *Base de Datos* es **INSERT**. *Este comando agrega una fila a la tabla*, y realizando ciertas variaciones con este comando es posible insertar múltiples filas de selección de datos de otras tablas.

La sintaxis básica del comando **INSERT** es la siguiente:

```
INSERT INTO tablaname [(columnname. …)] VALUES (constant. …)
```

Donde ***"tablename"*** representa el nombre de la tabla donde se desea insertar los datos, ***"columname"*** representa el nombre de la columna en la cual se insertara una parte especifica donde se ubica el dato y ***"constant"*** representa el dato que se va a insertar.

Para ejemplificar la utilidad de este comando se presenta la siguiente sentencia:

```
INSERT INTO t_music_types (music_type_title) VALUES (‘Rock and Roll’)
```

Este ejemplo permite insertar un registro en la tabla ***“t_music_types”***, estableciendo en la columna ***“music_type_title”*** la cadena ***“Rock and Roll”.***

3.5.2 Comando DELETE

El comando **DELETE** de **SQL** *permite eliminar una fila o múltiples filas de una tabla*. La sintaxis de este comando se presenta a continuación:

```
DELETE FROM tablename [ WHERE where expresión ]
```

Cabe mencionar que al establecer la sentencia anterior omitiendo la expresión **WHERE**, el resultado será eliminar todos los registros de la tabla especificada.

Sin embargo, el comando **WHERE** es empleado para especificar un criterio que identifique el registro que se desea eliminar. Enseguida un ejemplo de el uso de este comando:

```
DEDETE FROM t_albums WHERE band_id = 1
```

En el ejemplo anterior, serán eliminados todos los registros de la tabla ***“t_albums”*** que en su campo ***“band_id”*** tengan 1.

Existen otras variantes del comando **DELETE**, especificadas en la sección donde es utilizado el comando **WHERE**, por ejemplo, para ser más explícitos en el campo a eliminar puede ser utilizado el conector **AND** como se muestra enseguida:

```
DELETE FROM t_bands WHERE band_title = 'Hootie & The Blowfish' AND record_company_id = 100
```

Es posible utilizar también el conector **OR** como se muestra en el ejemplo siguiente:

```
DELETE FROM t_bands WHERE band_title = 'Toad The Wet Sprocket' OR record_company_id = 100
```

El comando **WHERE** también *permite llamar a un predicado*, es decir, una expresión que permite precisar de mejor manera la columna donde se encuentra el campo a eliminar. Los predicados utilizados son: **CONTAINS**, **LIKE** y **NULL**.

El siguiente ejemplo elimina todas las filas de la tabla ***"t_bands"*** que contienen la cadena ***"Toad".***

```
DELETE FROM t_bands WHERE CONTAINS ( band_title, 'Toad' )
```

En el siguiente ejemplo serán eliminados todos los registros de la tabla ***"t_bands"*** , donde el valor de la columna ***"band_title"*** comience con ***"Toad"***, ya que el signo ***"%"*** simboliza cualquier que pueda estar delante de la cadena ***"Toad".***

```
DELETE FROM t_bands WHERE band_title LIKE 'Toad%'
```

Finalmente, el predicado **NULL** determina si una columna contiene datos.

3.5.3 Comando SELECT

El comando **SELECT** de **SQL** es considerado como uno de los comandos más importantes, debido a que *es utilizado para recuperar o seleccionar datos de una o mas tablas establecidas*.

La estructura o sintaxis de este **comando SQL** puede ser tan complicada como así sea requerido por la información que se desee obtener de la o las tablas. Enseguida se muestran algunos ejemplos de la utilización de este comando.

Si se desea que sean devueltas todas las filas de una sola tabla en base en el ejemplo que se ha venido manejando, se especifica la siguiente sentencia:

```
SELECT * FROM t_bands
```

Esta sentencia *devolverá entonces todas las columnas de la tabla **"t_bands"***, pero si en lugar de ello es necesario especificar mas a detalle alguna columna o campo será necesario usar la siguiente sentencia:

```
SELECT band_member_fname AS "Last Name" , band_member_lname AS "First Name" , band_title AS "Band Title" FROM t_band_members,t_bands WHERE t_band_members.band_id = t_bands.band_id ORDER BY band_title , band_member_lname, band_member_fname
```

Los comandos anteriores devuelven datos de dos tablas diferentes, ***"t_bands"*** y ***"t_band_members",*** la cláusula **FROM** *especifica las tablas de las cuales se devolverán los datos*. Con esta sentencia es posible regresar tres columnas de dos tablas y son especificadas por medio del comando **SELECT**.

La cláusula **AS** *especifica el nombre con el cual serán renombradas las columnas seleccionadas* y la cláusula **ORDER BY** *permite especificar las columnas que serán utilizadas para ordenar las filas deseadas*.

Como fue posible observar, **SQL Server** puede convertirse en una poderosa herramienta, ya que ***otorga un sin número de beneficios en el desarrollo de aplicaciones Web*** en las que es necesario interactuar con datos concentrados en una *Base de Datos* de una manera versátil y eficiente.

3.6 Acceso a Datos con ADO .NET

Actualmente las **aplicaciones de Internet** son desarrolladas en ambientes totalmente heterogéneos, lo cual origina que tengan que acoplarse diversas plataformas.

Los nuevos ambientes están sujetos a nuevos retos tales como utilizar servicios comunes compartidos y sistemas escalables. Es por ello que ***Microsoft*** ha desarrollado **ASP .NET** para cubrir con estas necesidades y ambientes heterogéneos.

3.6.1 Características de ADO .NET

ADO .NET es la nueva versión de modelo de objetos **ADO (Active Data Objects)**, en otras palabras la nueva estrategia que ***Microsoft*** ofrece para el acceso a datos.
ADO .NET está diseñado para trabajar con conjuntos de datos desconectados, permitiendo con esto reducir el tráfico de red. Además utiliza **XML *(eXtensible Markup Language)*** como formato universal de transmisión de datos.

ADO .NET es la evolución del ***Modelo ADO*** para el acceso a datos, *el cual fue extremadamente efectivo y fácil de utilizar*, ya que permitía la interacción con múltiples tipos de estructuras de datos.

Sin embargo, el núcleo del **Modelo ADO**, el ***objeto RecordSet***, fue ampliado enormemente. Este elemento originalmente operaba simplemente como una interacción con **SQL**, pero el ***objeto RecordSet*** fue ampliado para soportar *desconexiones de datos, modelado de múltiples fuentes de datos, y una compleja capa de generación* **SQL**.

Actualmente los sistemas orientados a aplicaciones para Internet, necesitan una alta ***escalabilidad*** y ***redundancia***. Como sistema escalable, una de las principales limitaciones es la habilidad del ***Sistema Manejador de Bases de Datos Relaciónales o RBDMS*** para mantener y soportar conexiones abiertas entre varias transacciones.

Normalmente *existe un límite determinado* para el número de conexiones abiertas que pueden ser soportadas por el ***RBDMS***. En un ambiente distribuido, es lógico que serán necesarias un ***alto número de procesos y conexiones simultáneas*** para los recursos de datos.

Es decir, los componentes requieren abrir una conexión para la *distribución, filtrado y manipulación de los datos*, para lo cual ***ADO***, en cuestiones de escalabilidad, es ampliamente limitado.

ADO utilizó un eficiente y compacto formato binario, el ***"Advanced Data Table Gram"*** o formato ***ADTG***, para lograr la transmisión de datos entre aplicaciones. Sin embargo, el uso de este formato limitaba la transmisión en las **aplicaciones** o **ambientes COM**.

ADO .NET soporta una ***desconexión aproximada***, generando una disminución en el núcleo de datos, rendimiento del filtrado, búsquedas, y manipulación de los diversos procesos.

Lo anterior es logrado gracias a que la base **XML** *puede transmitir los datos manipulados* a otros componentes por medio un proceso adicional antes de actualizar nuevamente a los datos originales almacenados.

Cuando la *transmisión* ocurre a través de la base estándar **XML** hay una significativa ***reducción en el procesamiento y conversión*** requerida para acceder a los datos en ambientes heterogéneos. Además **XML** es soportado en **COM**, **CORBA** y otra gran cantidad de ambientes, ya que este formato puede lograr comunicación a través de **HTTP**.

3.6.2 Funcionalidad de XML para ADO .NET

XML se ha convertido en la *mejor forma de aplicar la información distribuida en la actualidad*, lo cual generó que el modelo anterior **ADO** fuera sustituido y en la versión **ADO .NET** se incluya éste como medio de transmisión.

Lo anterior permite que las clases de **ADO .NET** puedan implementar mecanismos de conversión de datos entre plataformas, lectura de datos de cualquier origen y la habilitación de mecanismos de persistencia en el mismo formato en el que se procesan, entre algunos otros beneficios.

Dentro de la redefinición que ***Microsoft*** ha hecho al modelo de acceso a datos, se ha colocado de alguna manera un ***intermediario entre el cliente y los datos***, éste intermediario transforma cada comando y cada dato en modelos de **documentos XML**.

En primer lugar son cargados en el lado ***cliente*** los documentos necesarios, que se encuentran almacenados en la ***superclase DataSet***, y de esta manera es posible trabajar con documentos sin necesidad de estar consumiendo continuamente recursos de la red.

Por último se procesan los cambios que se han producido enviándolos a la *Base de Datos*, es decir, el adaptador tomará los cambios del documento y los replicará al servidor.

Lo explicado de forma breve en párrafos anteriores es representado gráficamente por medio de la **Figura 3.21** que se muestra a continuación.

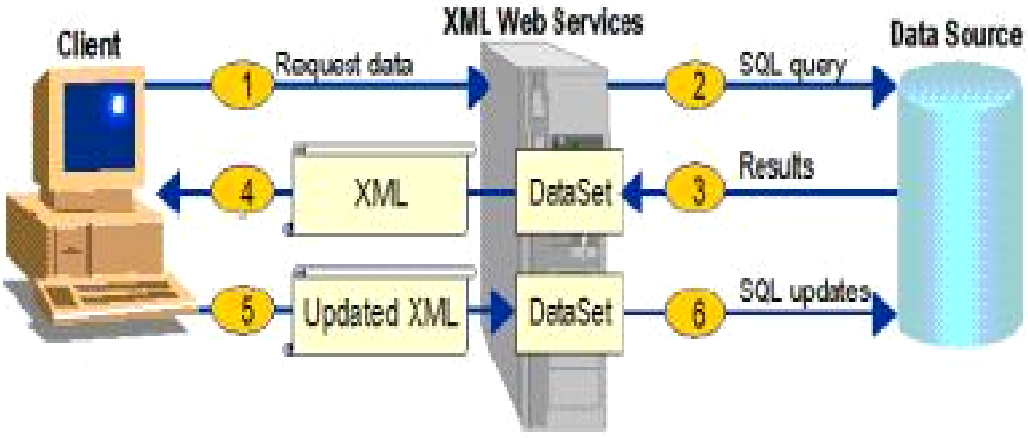


Figura 3.21. Interacción Cliente-Servidor.

En el modelo **ADO** *la manipulación de desconexiones era limitada*, ya que era necesario establecer las conexiones abiertas o cerradas programando las mismas. En **ADO .NET** esto es utilizado de forma automática.

3.6.3 Arquitectura “Off-line” o modo desconectado

En el funcionamiento de una aplicación que requiere *un* ***constante acceso a datos***, y donde además existen ***múltiples conexiones*** que hacen uso de recursos del sistema, es importante tener en cuenta que este contexto *implica que la demanda de recursos sea mayor y que al mismo tiempo la aplicación se encuentra con el problema de tener que estar conectada al servidor continuamente* para alimentar las demandas existentes.

Debido a lo anterior, **ADO .NET** permite la conexión al servidor sólo estrictamente lo necesario realizando las operaciones pertinentes de carga de los datos en el **DataSet.**

De esta manera, *son reducidos los bloqueos y las conexiones a lo mínimo posible* y es posible el soporte para una mayor cantidad de usuarios por unidad de tiempo, disminuyendo los tiempos de respuesta y acelerando la ejecución de los programas.

3.6.4 DataSet

Es importante señalar que un **DataSet** es una ***caché de registros***, los cuales son recuperados de una *Base de Datos* que actúa como un ***sistema de almacenamiento virtual***. El **DataSet** contiene una o más tablas basadas en las tablas reales de la *Base de Datos*, además *almacena las relaciones y reglas de integridad existentes entre ellas*, lo cual garantiza la estabilidad e integridad de la información de la *Base de Datos*.

En el diagrama que se presenta en la **Figura 3.22**, muestra la el esquema de un **DataSet**.

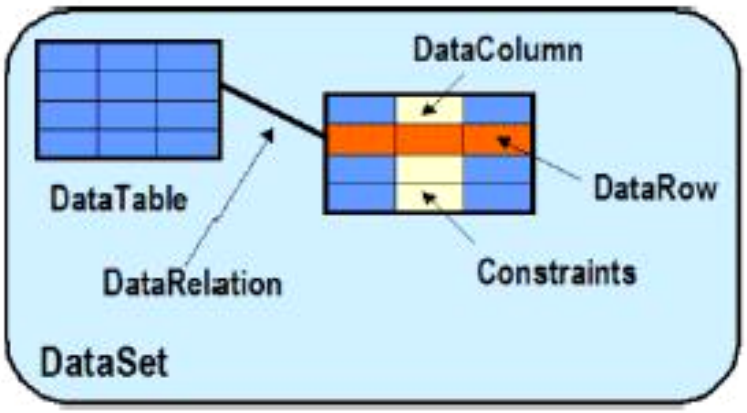


Figura 3.22. Esquema de DataSet.

Es así, como una copia ***en el lado cliente*** de la arquitectura de la *Base de Datos* basado en un **esquema XML**, proporciona al desarrollador la libertad de trabajar de manera independiente de la plataforma.

Finalmente, puntualizando el concepto principal, en concreto **ADO .NET** puede definirse de la siguiente manera:

*Conjunto de interfaces, clases, estructuras y enumeraciones que permiten el acceso a los datos desde la **plataforma .NET** de **Microsoft**, permitiendo un modo de acceso desconectado a los datos provenientes de múltiples fuentes de diferente arquitectura de almacenamiento.*

3.6.5 Beneficios otorgados por ADO .NET

1. **Interoperabilidad**.- Debido a que **XML** es el estándar empleado para el envío de información entre capas, cualquier componente capaz de interpretar los **datos XML** puede acceder a la información de **ADO .NET** y procesar dicha información no importando su ubicación.

2. **Mantenimiento**.- Cuando es necesario un ***cambio estructural del sistema***, resulta una tarea demasiado compleja y poco viable. Es por ello que en este modelo *se separa la estructura de un programa en varias capas*. Una de esas capas es la de datos, la cual es necesario desarrollarla correctamente.

   Gracias a **DataSets** portar y aumentar los procesos de datos y de negocio es más sencillo. Intercambiar la información por medio de **XML**, facilita la tarea de estructurar más capas en la aplicación.

3. **Programación.-** Los programadores pueden acceder a un ***API de programación estructurado de fuerte tipificado***. Es decir, se establece en la estructura del lenguaje las herramientas que el programador necesita para una programación fácil y sencilla.

4. **Rendimiento.**- Debido a que se trabaja con objetos ***desconectados***, el proceso se acelera, y gracias al **modelo XML** no es necesaria la conversión de tipos a nivel **COM**. Por lo tanto, el ancho de banda disponible *se reduce y se independiza más el cliente del servidor* logrando que este pueda realizar otras tareas.

5. **Escalabilidad.-** Cuando el servidor debe atender ***millones de conexiones***, se genera un aumento en tiempo y recursos, por lo que **ADO .NET** favorece la escalabilidad por medio de su modelo **"Off-line"** o **desconectado**, evitando que se mantengan los recursos reservados más del tiempo necesario. Lo anterior permite que más usuarios por unidad de tiempo puedan acceder a la aplicación sin problemas.

3.6.6 Establecimiento de la conexión

Para establecer una ***conexión*** con **ADO .NET**, es necesario utilizar el objeto **SqlConnection** del objeto **Connection** serán utilizados los métodos **Open( )** y **Close( )**, los cuales *abren y cierran las conexiones respectivamente*, con el almacén de datos a utilizar.

Una vez que es lanzado el método **Open( )** sobre un objeto **Connection *(Sqlconnection)***, será abierta la conexión indicada en la propiedad **ConnectionString**, por lo que esta propiedad indicará la cadena de conexión que será utilizada para establecer la conexión con la *Base de Datos*.

Enseguida se presenta la sintaxis adecuada para establecer una conexión dentro del ejemplo representado en el siguiente código:

```
<%@ Page language=”C#”%>
<%@ Import Namespace=”System.Data”%>
<%@ Import Namespace=”System.Data.SqlClient”%>
<html>
<head><title>El objeto Connection</title></head>
<body>
<script runat=”server”>
void Conecta(Object sender, EventArgs args){
     SqlConnection conexion =new SqlConnection
                (“server=angel;databse=northwind;uid=sa;pwd=”);

     try{
          conexion.Open();
          Response.Write(“Se ha establecido la conexión<br>”);
          conexión.Close();
          Response.Write(“Se ha cerrado la conexión<br>”);
```

```
      }catch(SqlException e){
         Response.Write("Se ha producido una excepción: "+e);
      }
}
</script>
<form runat="server" id="formulario">
<asp:Button OnClick="Conecta" Runat="server" ID="boton" Text=
"Conexión"/>
</form>
</body>
</html>
```

Cabe mencionar que dentro del ejemplo representado anteriormente, se hizo uso del mecanismo de tratamiento de errores mediante bloques ***try/catch***, para poder manejar las excepciones que puedan surgir al abrir o cerrar la conexión.

3.6.7 Utilización de un Objeto Command

Dentro de la ejecución de una **aplicación ASP .NET Web**, es probable que sea necesario ejecutar una **sentencia SQL** de cualquiera de los tipos mas representativos que permita la interacción con la *Base de Datos*.

Debido a lo anterior, a continuación se ejemplifica por medio de código la utilización de un objeto **SqlCommand** para ejecutar una **sentencia SQL** de tipo **INSERT** sobre una tabla especifica, utilizando para esto el método **ExecuteNonQuery().**

```
<%@ Page language="C#"%>
<%@ Import Namespace="System.Data"%>
```

```
<%@ Import Namespace="System.Data.SqlClient"%>
<html>
<head><title>El objeto Command</title></head>
<body>
<script runat="server">
void Alta(Object sender, EventArgs args){
     SqlConnection conexion =new SqlConnection
                ("server=angel;databse=northwind;uid=sa;pwd=");

     String sentencia="INSERT into Employes (firstname, lastname) "+
                      "VALUES (@nombre, @apellido)";
     SqlCommand comando=new SqlCommand (sentencia, conexion);
     Int resultado;
     try{
          conexion.Open();
          comando.Parameters.Add(new SqlParameter("@nombre",
                                 SqlDbType.NvarChar, 10));
          comando.Parameters["@nombre"].Value=nombre.Text;
          comando.Parameters.Add(new SqlParameter("@apellidos",
                                 SqlDbType.NvarChar, 20));
         comando.Parameters["@apellidos"].Value=apellidos.Text;
         resultado=comando.ExecuteNonQuery();
        Response.Write("Se ha añadido "+resultado+" registro");
          conexion.Close();

       }catch(SqlException e){
          Response.Write("Se ha producido una excepción: "+e);
       }
}
</script>
<form runat="server" id="formulario">
<asp:Label Runat="server" Runat="server" ID="textoNombre" text=
```

```
“Nombre”/>
<asp:TextBox Runat=“server” ID=“nombre”/><br>
<asp:Label Runat=“server” Runat=“server” ID=“textoApellidos”
text= “Apellidos”/>
<asp:TextBox Runat=“server” ID=“apellidos”/><br>
<asp:Button OnClick=”Alta” Runat= “server” ID= “boton” Text=
“Añadir”/>
</form>
</body>
</html>
```

En este ejemplo es necesario hacer uso de la propiedad **Parameters** del objeto **SqlCommand**, añadiendo los dos parámetros necesarios mediante un método **Add()**.

A este método le serán pasados los parámetros por medio del objeto de la clase **SqlParameter**, indicado para cada uno de los objetos el nombre del parámetro, el tipo y el tamaño en su constructor. Después de esto es posible asignar el valor correspondiente, el cual se encuentra especificado en controles Web del ***WebForm***.

3.6.8 Utilización de un Objeto DataReader

Los objetos **DataReader** son el equivalente a los cursores de sólo lectura y movimiento hacia delante utilizados por **ADO**. Sin embargo, un objeto **DataReader**, se obtiene de la ejecución de una **sentencia SQL** o bien de la ejecución de un procedimiento que se encuentra almacenado, a partir de la llamada al método **ExecuteReader( ).**

A continuación se presenta el código que pertenece a una **página ASP .NET** que obtiene un objeto **SqlDataReader** por medio de la ejecución de un objeto **SqlCommand**, al cual se le lanza el método **ExecuteReader( ).**

```
<%@ Page language=”C#”%>
<%@ Import Namespace=”System.Data”%>
<%@ Import Namespace=”System.Data.SqlClient”%>
<html>
<head><title>El objeto DataReader</title></head>
<body>
<script runat=”server”>
void Muestra(Object sender, EventArgs args){
     SqlConnection conexion =new SqlConnection
                (“server=angel;databse=northwind;uid=sa;pwd=”);
   String sentencia=“SELECT firstname, lastname From Employes”;
   SqlCommand comando=new SqlCommand (sentencia);
   SqlDataReader resultado;
     try{
          conexion.Open();
          comando.Connection=conexion;
          resultado= comando.ExecuteReader();
          while(resultado.Read()){
          Response.Write(resultado.GetString(0)+” “+
                              resultado.GetString(1)+”<br>”);
         }
          resultado.Close();
          conexión.Close();
         }catch(SqlException e){
          Response.Write(“Se ha producido una excepción: ”+e);
       }
}
```

```
</script>
<form runat=”server” id=“formulario”>
<asp:LabelRunat=“server” ID=“Texto” Text= “”/><br>
<asp:Label OnClick=“Muestra” Runat=“server” ID=“boton” Text=
“Muestra Datos”/>
</form>
</body>
</html>
```

El ejemplo anterior presenta en la pantalla el contenido del objeto **SqlDataReader**, haciendo uso del método **Read( )** en un *ciclo While*.

3.6.9 Utilización de un Objeto DataAdapter

Como ya se mencionó, los objetos **DataAdapter** *realizan la función de un puente* entre el almacén de datos y el **DataSet**, permitiendo cargar el **DataSet** desde el origen de datos y después actualizar los datos de origen por los de **DataSet**.

En el código que se muestra enseguida, se ejemplifica la utilización de los objetos **Data Adapter**. En este ejemplo referido es utilizado un objeto **SqlDataAdater** con el cual se dará de alta un registro de una tabla y además se mostrarán los contenidos de ésta mediante un objeto **DataSet**.

```
<%@ Page language=”C#”%>
<%@ Import Namespace=”System.Data”%>
<%@ Import Namespace=”System.Data.SqlClient”%>
<html>
<head><title>El objeto DataAdapter</title></head>
<body>
```

```
<script runat=”server”>
SqlConnection conexion;
SqlDataAdapter adapter;
String sentenciaInsercion=“INSERT into Employees(firstname,
lastname) ”+ “VALUES(@nombre,@apellidos)”;

String sentenciaSeleccion=“select firstname, lastname from
Employees”;

SqlCommand comandoInsercion;
DataSet ds;

Void Page_Load(Object sender, EventArgs args){
    conexion=new SqlConnection(“server=angel;database=
                                    northwind;uid=sa;pwd=”);
    adapter=new SqlDataAdapter();
    comandoInsercion=new SqlCommand(sentenciaInsercion,
conexion);
    comandoSeleccion=new SqlCommand(sentenciaSeleccion,
conexion);

    adapter.InsertCommand=comandoInsercion;
    adapter.SelectCommand=comandoSeleccion;
    ds=new DataSet();
    if(!Page.IsPostBack){
          MuestraDatos();
    }

}
```

```
void MuestraDatos(){
      try{

           conexión.Open();

           adapter.Fill(ds, “Empleados”);
           tabla.DataSource=ds.Tables[“Empleados”].DefaultView;
           tabla.DataBind();
           conexión.Close();
      }catch(SqlException ex){
          Response.Write(“se ha producido una excepción: “+ex);
   }
}
void Alta(Object sender, EventArgs args){
      int resultado;
      try{
         conexion.Open();
         adapter.InsertCommand.Parameters.Add(new SqlParameter(
                   “@nombre”,SqlDbType.VarChar, 10));
adapter.InsertCommand.Parameters[“@nombre”].Value=nombre.Text;
         adapter.InsertCommand.Parameters.Add(new SqlParameter(
                   “@apellidos”,SqlDbType.VarChar, 20));
adapter.InsertCommand.Parameters[“@apellidos”].Value=apellidos.Text;
         resultado=adapter.InsertCommand.ExecuteNonQuery();
         Response.Write(“Se ha añadido “+resultado+” registro”);
         Conexión.Close();
         MuestraDatos();
      }catch(SqlException e){
        Response.Write(“Se ha producido una excepción: “+e);
      }
}
```

```
</script>
<form runat=”server” id=”formulario”>
<asp:Label Runat=“server” ID=“textoNombre” text=“Nombre”/>
<asp:TextBox Runat=“server” ID=“nombre”/><br>
<asp:Label Runat=“server” ID=“textoApellidos” text=“Apellidos”/>
<asp:TextBox Runat=“server” ID=“apellidos”/><br>
<asp:Button OnClick=“Alta” Runat=“server” ID=“boton” Text=“Añadir”/>
</form>

<asp:DataGrid id=“tabla” runat=“server” BorderWidth=“1” GridLines=
“Both” HeaderStyle-Font-Bold=“True” HeaderStyle-BackColor=“yellow”>
</asp:DataGrid>

</body>
</html>
```

El objeto **SqlDataAdapter** utilizado en el ejemplo anterior, contiene dos objetos **SqlCommand**, uno utilizado para la inserción que se asignará a la propiedad **InsertCommand** del objeto **SqlDataAdapter** y otro que permitirá realizar la sentencia de selección de la tabla perteneciente a la *Base de Datos*.

Es preciso agregar, que al ejecutar el método **Fill( )** del objeto **SqlDataAdapter** se ejecutará el comando de la propiedad **SelectCommand**.

3.6.10 Utilización de Objetos para desplegar información

DataSet es un objeto común perteneciente a **ADO .NET**, éste es similar a un objeto ***Recordset de ADO***, pero de mucho mayor potencia y complejidad. Es decir, *es el almacén de datos* en **ADO .NET**, representado en una *Base de Datos* desconectada del proveedor de datos.

Para lograr crear e inicializar tablas pertenecientes a **DataSet**, se debe hacer uso del Objeto **DataAdapter**. Este objeto le pasará por parámetro una cadena que representa la consulta a ejecutar y que rellenará los datos de **DataSet**. Para ello deberá ser utilizado también el método **Fill( )** con sus dos parámetros.

A continuación, en el siguiente código, se presenta un ejemplo de la utilización de un objeto **DataSet** en una **página ASP .NET**, creando un objeto **DataSet** que contendrá una tabla.

Después de haber cargado el objeto **DataSet** será mostrado su contenido en un control **DataGrid** mediante el mecanismo **DataBrinding**.

```
<%@ Page language=”C#”%>
<%@ Import Namespace=”System.Data”%>
<%@ Import Namespace=”System.Data.SqlClient”%>
<html>
<head><title>El objeto DataReader</title></head>
<body>
<script runat=”server”>
void Page_Load(Object sender, EventArgs e){
     SqlConnection conexion =new SqlConnection
                (“server=angel;databse=northwind;uid=sa;pwd=”);
     SqlDataAdapter adapter=new SqlDataAdapter(
   “select firstname, lastname,city from Employees”, conexion);
     DataSet ds=new DataSet();
     try{
          conexion.Open();
          adapter.Fill(ds, “Empleados”);
          tabla.DataSource=ds.Tables[“Empleados”].DefaultView;
          tabla.DataBind();
          conexión.Close();
```

```
         }catch(SqlException ex){
          Response.Write("Se ha producido una excepción: "+ex);
       }
}
</script>

<body>

<asp:DataGrid id="tabla" runat="server" BorderWidth="1"
GridLines= "Both" HeaderStyle-Font-Bold="True" HeaderStyle-
BackColor="yellow">
</asp:DataGrid>

</body>
</html>
```

3.7 Tratamiento Estructurado de errores

En versiones anteriores de **ASP**, sólo contaban con sentencias como **On Error Resume Next**, por lo que no se tenía la posibilidad de especificar una etiqueta para el tratamiento de errores. Esto originaba cierta *dificultad* en el momento de crear rutinas centralizadas para el tratamiento de errores.

Un aspecto que es importante resaltar, es la desaparición del objeto **ASPError** en la nueva versión **ASP .NET** con respecto a **ASP 3.0**. Este objeto ofrecía una descripción detallada del error producido, pero ahora en la versión actual este manejo se realiza por medio de ***excepciones***.

El sistema estructurado para el tratamiento de errores de **ASP .NET** ofrece una serie de mecanismos para el tratamiento de errores que permiten un manejo más adecuado y eficiente.

EL **modo de tratamiento de errores estructurado** forma parte del **CLR de ASP .NET**, ofreciendo una atractiva manera de gestión de errores para los desarrolladores. Este tratamiento de errores se basa en la captura de ***excepciones***, *entendiendo por excepción, la respuesta a situaciones erróneas o anormales producidas en tiempo de ejecución de una aplicación.*

Todos los tipos de ***excepciones*** manejados heredan de la clase **System.Exception** del **.NET Framework**. Esta estructura presenta una serie de características representativas que le permiten sobresalir con respecto a versiones anteriores. A continuación se mencionan las más sobresalientes:

1. *Parte de un* ***Sistema Jerárquico***, ordenando en capas los diferentes ***tipos de excepciones***.

2. *Actúa independientemente del proceso y de la máquina*, permitiendo la captura de ***excepciones*** localmente que son lanzadas por ***componentes .NET remotos***.

3. *Actúa independientemente del lenguaje*, permitiendo la creación y lanzamiento de las ***excepciones*** en un lenguaje y ser atrapadas en otro lenguaje.

4. Dentro de las llamadas a métodos *no es necesario comprobar valores*, por lo que las ***excepciones*** se lanzan automáticamente cuando se produzcan.

3.8 Clase Exception

La importancia de esta clase recae en la capacidad para informar al programador acerca del error que se ha generado; por lo que enseguida se mencionan las propiedades contenidas en esta clase para cumplir con la función antes mencionada.

- **Message**.- *Devuelve una cadena con la descripción del error que se generó*.

- **Hresult**.- *Devuelve un código numérico que permite identificar la excepción generada*. Cabe mencionar que esta propiedad es utilizada para la interoperatividad con **COM**.

- **InnerException**.- Cuando existen ***excepciones anidadas*** esta propiedad adquiere sentido, *devolviendo un objeto* ***Exception*** que representa a la ***excepción interna***.

- **Source**.- Esta *devuelve una cadena*, la cual contiene el nombre de la aplicación u objeto que a generado o lanzado el error.

- **StackTrace**.- *Devuelve una cadena* con las trazas de la pila de llamadas, además indica el lugar dentro del código en el que se a generado la ***excepción***.

- **TargetSite**.- Esta propiedad *devuelve el método* que lanzó la ***excepción***.

- **HelpLink**.- Se refiere a una cadena que contiene una ***URL*** al archivo de ayuda asociado por dicho error.

3.9 Manejo de Excepciones (Try/Catch)

El ***manejo de errores*** es de forma muy similar a como se realiza en el ***lenguaje Java***, por lo que el uso de las sentencias **Try**, **Catch** y **Finally**, permitirán atrapar las ***excepciones*** dentro del código.

Por ejemplificar de alguna forma, en el **lenguaje C#**, cuando se produce un error en algún método, el método crea un ***objeto excepción*** el cual es pasado al entorno de ejecución. *El objeto contendrá información respecto del error que se produce*.

Los métodos utilizados o invocados antes de que se genere un error son los indicados para tratar el error, por lo que el *entorno de ejecución realiza una búsqueda hacia atrás en la pila de llamadas*.

Para manejar una ***excepción***, primeramente se debe declarar un **bloque Try**, el cual debe englobar a todas las sentencias que pueden generar una ***excepción***.

El manejador de ***excepciones*** asociado a **Try**, es indicado mediante uno o mas **bloques Catch**. El bloque correspondiente a **Catch** de manera más común puede ir vacío, y sus parámetros declaran una ***variable del tipo excepción***, la cual es la que se desea atrapar.

La variable de excepción permite recoger el objeto correspondiente a la ***excepción*** que se produzca, utilizando esta para conocer la información que se ofrece.

A un **bloque Try**, debe corresponderle el uso de uno o más **bloques Catch** para poder manejar una ***excepción*** y existen ocasiones en las que será necesario utilizar un **bloque Finally**.

Dentro del **bloque Finally** se especifica el código que se desea limpiar, es decir, nos permitirá liberar objetos, archivos, etcétera.

Enseguida se presenta el código del esquema que deberá utilizarse para el manejo de ***excepciones*** por medio de estos tres comandos.

```
Try{
     ...
} catch (ClaseException nombreVariable) {
     ...
} [catch (CalseException nombreVariable) {
     ...
}
[....]
[finally{
       .....
}]
```

Para ejemplificar el uso de este esquema anterior, se presenta el código de una **página ASP .NET** en la cual se hace uso de estos bloques.

La página contendrá un **objeto TextBox**, y en éste se especificará el contenido de un archivo de texto y también el usuario a través de otra ***caja de texto***. Finalmente, al *pulsar un botón* en el ***WebForm*** se abrirá el archivo mostrando su contenido.

```
<head><title>Tratamiento de excepciones</title></head>
<script runat=“server”>
void MostrarFichero(Object sender, EventArgs evento){

try{
     FileStream fs=new FileStream(nombreArchivo.Text,
                                        FileMode.Open);
     StreamReader sr=new StreamReader(fs);
```

```
      String contenido;
      contenido=sr.ReadToEnd();
      resultado.Visible=true;
      resultado.Text=contenido;
      fs.Close();
  }
   catch(ArgumentException excepcion){
        error.Text="Debe indicar el nombre del archivo";
        resultado.Text="";
        resultado.visible=false;
  }
   catch(FileNotFoundException excepcion){
        error.Text="No se ha encontrado el archivo";
        resultado.Text="";
        resultado.visible=false;
  }
   catch(IOException excepcion){
        error.Text="Excepción de entrada/salida";
        resultado.Text="";
        resultado.visible=false;
  }
   catch(Exception excepcion){
        error.Text="Se ha producido la exepción: "+
                                        excepcion.Message;
        resultado.Text="";
        resultado.visible=false;
  }
   finally{
           Response.Write("siempre se ejecuta");
  }
}
```

```
</script>
<body>
<form meted=“POST” runat=“server” ID=“Formulario”>
<asp:TextBox Runat=“server” ID=“nombreFichero”/>
<asp:button text=“Enviar” runat=“server” ID=“boton” OnClick=
                                        “MostrarArchivo”/><br>
<asp: Label ID=“error” Runat=“server” EnableViewState=
                                           “False” Text= “”/><br>
<asp:TextBox TextMode=“MultiLine” Runat=“server” ID=“resultado”
Columns=“40” Rows=“5” EnableViewSatate=“False” Text=“” Visible=
                                                        “False”/>
</form>
</html>
```

Dentro del bloque correspondiente a **Try**, se encuentra el código encargado de *abrir el archivo y mostrar su contenido*, esto debido a que es común que estas operaciones generen ***excepciones***.

Existen ***tres bloques Catch*** para el ***bloque Try*** utilizado, esto permite el manejo de tres ***excepciones diferentes***, como la de la clase **ArgumentException**, la cual es lanzada cuando no se especifique el nombre del archivo.

La segunda ***excepción*** está relacionada con la clase **FileNotFoundException**, y esta se produce cuando no se localiza el archivo especificado.

Finalmente, la ***excepción*** más general perteneciente a la clase **Exception**, la cual representará ***cualquier clase de excepción*** que se presente.

3.10 Lanzamiento de Excepciones

Para el lanzamiento de ***excepciones*** será necesario utilizar la sentencia **throw** y enseguida debe especificarse un objeto instancia de la clase de la ***excepción que se desee lanzar***.

La técnica comúnmente utilizada consiste en **"atrapar"** las ***excepciones generadas*** y después relanzarlas añadiendo la información que el programador considere necesaria construyendo sus propios ***mensajes de error***.

Retomando el ejemplo anterior es posible ejemplificar cómo se puede lanzar una ***excepción***. Para esto se relanzará la ***excepción*** indicando una descripción más detallada de la misma, como se muestra en el código siguiente:

```
<%@ Page language="C#"%>
<%@ Import Namespace="System.IO"%>
<%@ Import Namespace="System.Data.SqlClient"%>
<html>
<head><title>Tratamiento de excepciones</title></head>
<script runat="server">
void MostrarArchivo(Object sender, EventArgs evento){
   try{
        FileStream fs=new FileStream(nombreArchivo.Text,
                                          FileMode.Open);
        StreamReader sr=new StreamReader(fs);
        String contenido;
        contenido=sr.ReadToEnd();
        resultado.Visible=true;
        resultado.Text=contenido;
        fs.Close();
    }
```

```
     catch(Argument Exception excepcion){
       throw new ArgumentException("Debe indicar el nombre del
                                                    archivo");
    }
     catch(FileNotFoundException excepcion){
       throw new FileNotFoundException("No se ha encontrado el
                                                    archivo");
    }
     catch(IOException exception){
        throw new IOException("Excepción de entrada/salida");
    }
     catch(Exception excepcion){
        throw new Exception("Se ha producido la excepción:
                                        "+excepcion.Message);
    }

}
</script>
<body>
<form method="POST" runat="server" ID="Formulario">
<asp:TextBox Runat="server" ID="nombreArchivo"/>
<asp:button text="Enviar" runat="server" ID="boton" OnClick=
                                    "MostrarArchivo"/><br>
<asp:TextBox TextMode="MultiLine" Runat="server"ID= "resultado"
Columns="40" Rows="5" EnableViewState="False" Text="" Visible=
                                                     "False"/>
</form>
</html>
```

De esta forma, al producirse una ***excepción*** la ejecución será *interrumpida* apareciendo la descripción especificada en el código.

La forma en que son presentadas las ***excepciones*** por **ASP .NET** se muestra en la **Figura 3.23**, en la que es posible observar la *interrupción de la ejecución* por el mensaje generado por una **excepción**.

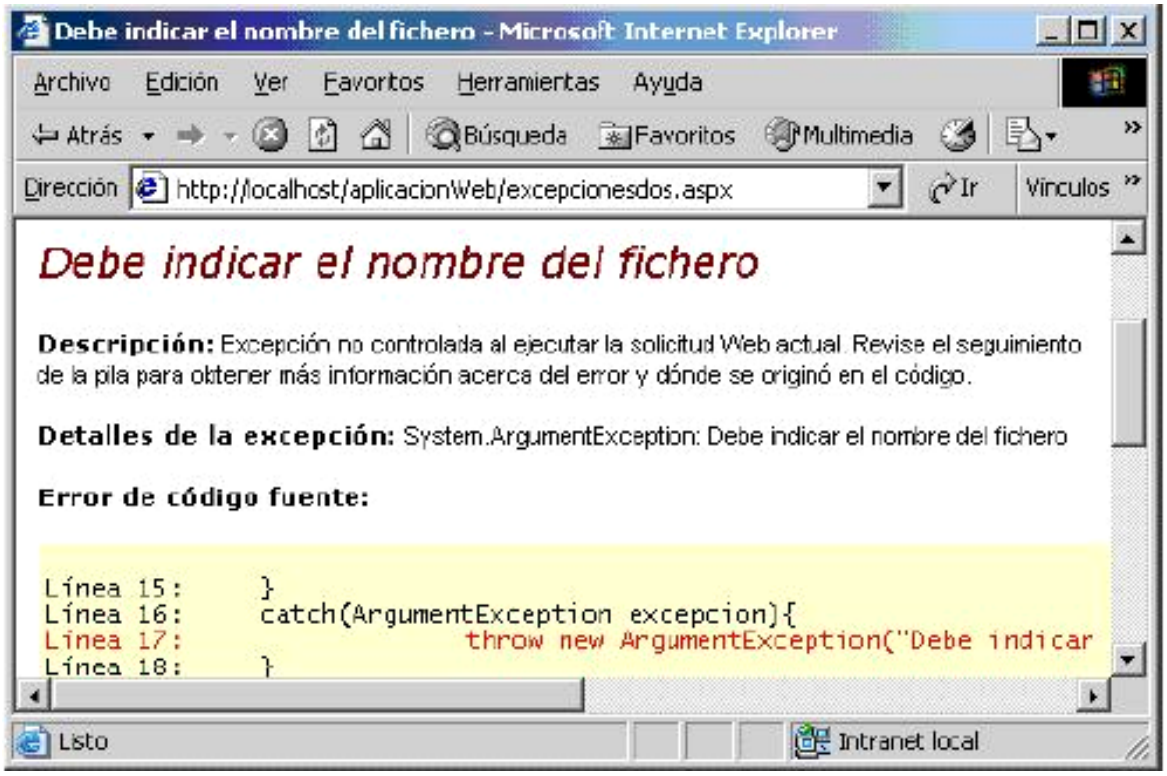


Figura 3.23. Excepción.

3.11 Manejo de Errores

Dentro de una **página ASP .NET** pueden producirse errores de diversos tipos. Estos errores se refieren a diversas áreas que interfieren en el proceso de ejecución, a continuación se especifican los ***tipos de errores*** que pueden ocurrir dentro de una **página ASP .NET**:

Error de Configuración. *Ocurre cuando la* ***sintaxis o estructura de un archivo Web.config****, es incorrecta dentro de la jerarquía de configuración de una* ***aplicación ASP .NET****.*

Error del analizador. *Se produce cuando la* ***sintaxis de la página ASP .NET*** *esta construida de forma incorrecta.*

Error del Compilador. *Este tipo de error ocurre cuando las* ***sentencias del lenguaje de la página son incorrectas****.*

Error en tiempo de ejecución. *Se produce durante el* ***proceso de ejecución de la página****.*

Cuando se produce un error en una **página ASP .NET** es localizado el ***método que causo la excepción***, y si el **modo de depuración** se encuentra activado *se mostrará el número de línea donde se ha producido el error, además de su código fuente*.

Es posible *activar* o *desactivar* el **modo depuración** en una **pagina ASP .NET** utilizando el atributo **Debug** de la directiva ***@Page***. Esta propiedad tiene el valor ***True*** por defecto.

En **ASP .NET** existen diferentes formas de manejar o tratar un error, dentro de las cuales se encuentran las siguientes:

- ***Indicando una página de error*** para el tratamiento de los errores a través del atributo **ErrorPage** de la directiva **@Page**.

- ***Por medio del fichero de configuración de la aplicación***, es decir, con el **archivo Web.config** es posible declarar la forma en la que van a ser tratados los errores de la aplicación.

- Es posible tratar los errores a ***nivel de página por medio del evento error***. Este tratamiento debe ser realizado por cada una de las **páginas ASP .NET**.

- Es posible tratar los errores ***a nivel de aplicación mediante el evento de error***, en este caso el tratamiento de errores se aplica de *manera completa* a la **aplicación ASP .NET**, programándolo en el archivo **GLOBAL.ASAX**.

3.11.1 Page_Error

Page_Error es uno de los métodos que **ASP .NET** ofrece para ***tratar errores a nivel de página***. En este método se tratarán los errores a través del código, cuando se genere una excepción que no es tratada en la pagina.

El método **GetLastError( )** de la ***propiedad Server de la página***, permite obtener información sobre el error que se haya producido.

Ejemplificando la utilización del método **Page_Error**, se presenta nuevamente el código referente al archivo el cual ya fue mostrado anteriormente.

Para eliminar el error y para que no se muestre la página típica de error de las **aplicaciones ASP .NET**, será necesario llamar al método **ClearError( )** del **objeto Server**, que es instancia de la clase **HttpServerUtility**.

El código es el siguiente:

```
<%@ Page language=”C#”%>
<%@ Import Namespace=”System.IO”%>
<%@ Import Namespace=”System.Data.SqlClient”%>
<html>
<head><title>Tratamiento de errores</title></head>
```

```
<script runat=”server”>
void Page_Error(Object obj, EventArgs evento){
      Response.Write(“Se ha producido un error<br>”);
      Response.Write(Server.GetLastError().ToString());
      Server.ClearError();
}
void MostrarArchivo(Object sender, EventArgs evento){
     FileStream fs=new FileStream(nombreArchivo.Text,
                                                FileMode.Open);
      StreamReader sr=new StreamReader(fs);
      String contenido;
      contenido=sr.ReadToEnd();
      resultado.Text=contenido;
      fs.Close();
}
</script>
<body>
<form method=“POST” runat=“server” ID=“Formulario”>
<asp:TextBox Runat=“server” ID=“nombreArchivo”/>
<asp:button text=“Enviar” runat=“server” ID=“boton” OnClick=
                                        “MostrarArchivo”/><br>
<asp:TextBox TextMode=“MultiLine” Runat=“server”ID= “resultado”
Columns=“40” Rows=“5”/>
</form>
</html>
```

Una vez ejecutado el código, ahora cuando se indica un nombre de un archivo inexistente, la **página ASP .NET**, se mostrará una pila de llamadas antes de producirse el error como lo muestra la **Figura 3.24**.

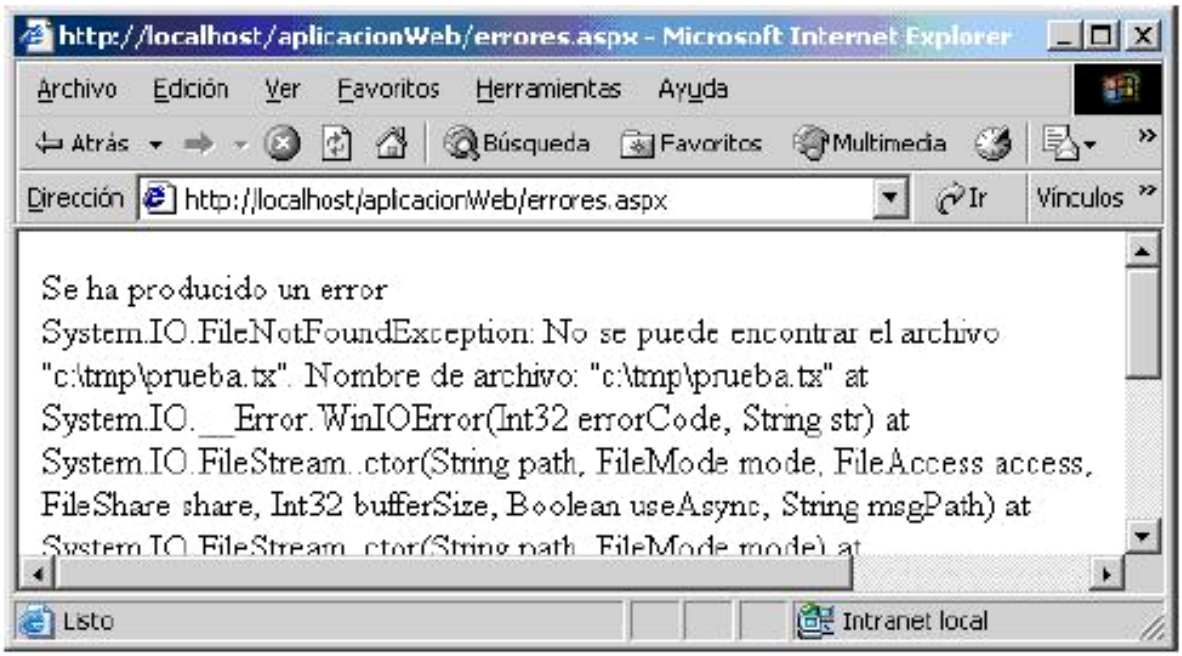


Figura 3.24. Pila de llamadas.

3.11.2 Application_Error.

Este método se ejecuta cuando se produce un error que corresponde a ***una excepción no tratada*** en la **página ASP .NET**, y además cuando *no se utiliza el método* ***Page_Error***.

También es importante señalar que este método debe incluirse en el archivo **GLOBAL.ASAX** de la **aplicación ASP .NET**. Este archivo trata eventos a nivel aplicación y se utilizará también para objetos a nivel de aplicación.

En el siguiente código, se muestra este método:

```
<script language="C#" runat="server">

void Application_Error(Object obj, EventArgs evento){
     Response.Write("Se ha producido un error en la página"
```

```
                              +Request.Url.ToString()+“<br>”);
     Response.Write(Server.GetLastError().ToString());
     Server.ClearError();
}

</script>
```

Para poder comprobar la ejecución del método **Application_Error**, se tendrá que generar un error dentro de la **página ASP .NET** sin tratar este dentro de la misma.

De esta manera al generarse el error se desplegará una ventana similar a la de la **Figura 3.24** del ejemplo anterior pero ahora con la pila de llamadas referentes a este método.

3.11.3 ErrorPage de la directiva @Page

El atributo **ErrorPage** como ya se ha mencionado, permite indicar una **URL** a una **página ASP .NET** que se muestra cuando se produce un error.

Este atributo debe emplearse cuando *no se requiera dar una información detallada del error al usuario*, en su lugar únicamente se mostrará una página error sencilla.

Deben activarse los errores personalizados acudiendo al **archivo de configuración Web.config**. Para indicar los errores personalizados de manera activa, se incluirá el **archivo Web.config**, y entre las etiquetas ***<system.web></system.web>*** se especificara el siguiente código:

```
<customErrors mode=”On”/>
```

Ejemplificando esta forma de tratamiento de errores se presenta el código de un ***WebForm*** en el cual se suman dos cantidades y se muestra el resultado.

```
<%@Page Language="C#" ErrorPage="PaginaError.aspx"%>
<html>
<head><title>Tratamiento de errores</title></head>
<script runat="server">
void Suma (Object obj, EventArgs e){
      int res;
      res=int.Parse(valor1.Text)+int.Parse(valor2.Text);
      resultado.Text=res.ToString();
}
</string>
<body>
<form id="formulario" runat="server">
<asp:TextBox id="valor1" runat="server" width="40">
</asp:TextBox>
<asp:TextBox id="valor2" runat="server" width="40">
</asp:TextBox>
<asp:TextBox id="resultado" runat="server" width="40">
</asp:TextBox>
<asp:Button id="boton" runat="server" text="Suma" Onclick=
                                                    "Suma">
</asp:Button>
</form>
</body>
</html>
```

El código anterior especifica que la pagina creada cuenta con su *pagina de error* ***(PaginaError.aspx),*** y el código de la pagina de error es el siguiente:

```
<%@Page Language="C#" ErrorPage="PaginaError.aspx"%>
<html>
<head>
<title>Pagina de error</title>
</head>
<body>
Se ha producido un error en la página:
<i><b><%=Request.QueryString["aspxerrorpath"]%></b></i>
</body>
</html>
```

Finalmente al introducir *valores erróneos* en los objetos correspondientes del ***Web Form***, como por ejemplo especificar una letra en lugar de un número se mostrará en el navegador el mensaje de error que se muestra en la **Figura 3.25**.

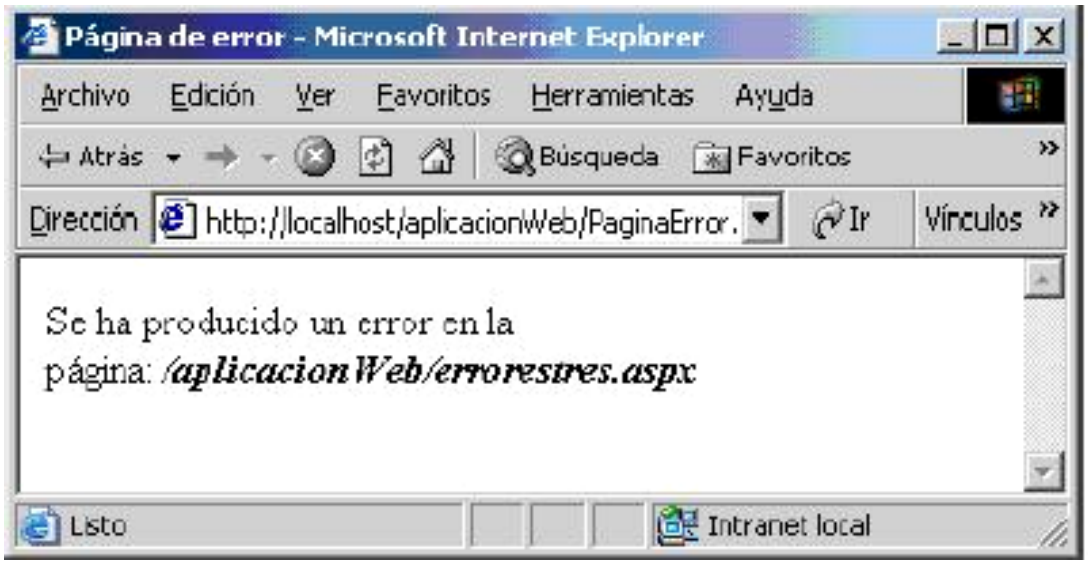


Figura 3.25. Error producido.

3.12 Depuración con ASP .NET

ASP .NET ofrece también como una de sus novedades *herramientas para realizar la depuración de paginas*. En anteriores versiones la manera de realizar la depuración era a base de la instrucción **Response.Write**, a pesar de la existencia de la herramienta **Script Debugger**, pues al realizarlo con esta herramienta resultaba un proceso demasiado complejo.

Debido a lo anterior, en la **plataforma .NET** se ha logrado ofrecer una depuración integra para todo tipo de aplicaciones, gracias al **CLR (Common Language Runtime).**

Para precisar, en el **entorno .NET Framework** se cuenta con dos formas de depurar:

- Utilizando la herramienta **SDK Debugger**.
- Utilizando el entorno de desarrollo de **Visual Studio .NET**.

3.12.1 Depuración con SDK Debugger

La ***Herramienta SDK Debugger*** forma parte de **.NET Framework** y permite la depuración de **páginas ASP .NET**. Este depurador ofrece funciones similares al ***depurador de Visual Studio .NET***, con excepción de la *depuración remota, la edición y continuación con la ejecución del código*.

Cuando se activa la depuración en una **página ASP .NET**, se indica al compilador que genere símbolos de depuración necesarios para la realización de dicho proceso al momento que sea necesario.

Al utilizar esta herramienta *se deberá indicar los componentes que serán depurados*. Para esto se utilizará el parámetro **debug** con el *compilador*.

Por otro lado si se realiza la compilación de los componentes desde **Visual Studio .NET**, nada adicional se deberá realizar.

Para realizar la depuración por medio de la ***herramienta SDK Debugger*** será necesario ejecutar el depurador de *manera externa*, es decir deberá ser ejecutado el archivo ***DbgCLR.exe***, el cual por lo general se encuentra en la ruta: ***C:\Archivos de programa\ MicrosoftVisualStudio.NET\ FrameworkSDK\ GuiDebug.***

Para iniciar, si se desea depurar una **página ASP .NET**, deberá especificarse la línea de código: **<%@ Page Debug="True"%>**, para activar la función de depuración.

Una vez agregada la línea de código a la página, se procede con la *ejecución* de la misma en el explorador, esto permitirá que los símbolos de depuración sean cargados para que el depurador pueda emplearlos en su proceso.

Ya que se tiene preparada la **página ASP .NET**, es decir ya que se ejecuto y se encuentra cargada en el explorador, *se deberá ejecutar el* ***depurador*** ubicado en la ruta que se especificó el inicio de esta sección.

Al ser ejecutado el ***depurador*** se desplegará la ventana representada en la **Figura 3.26.**

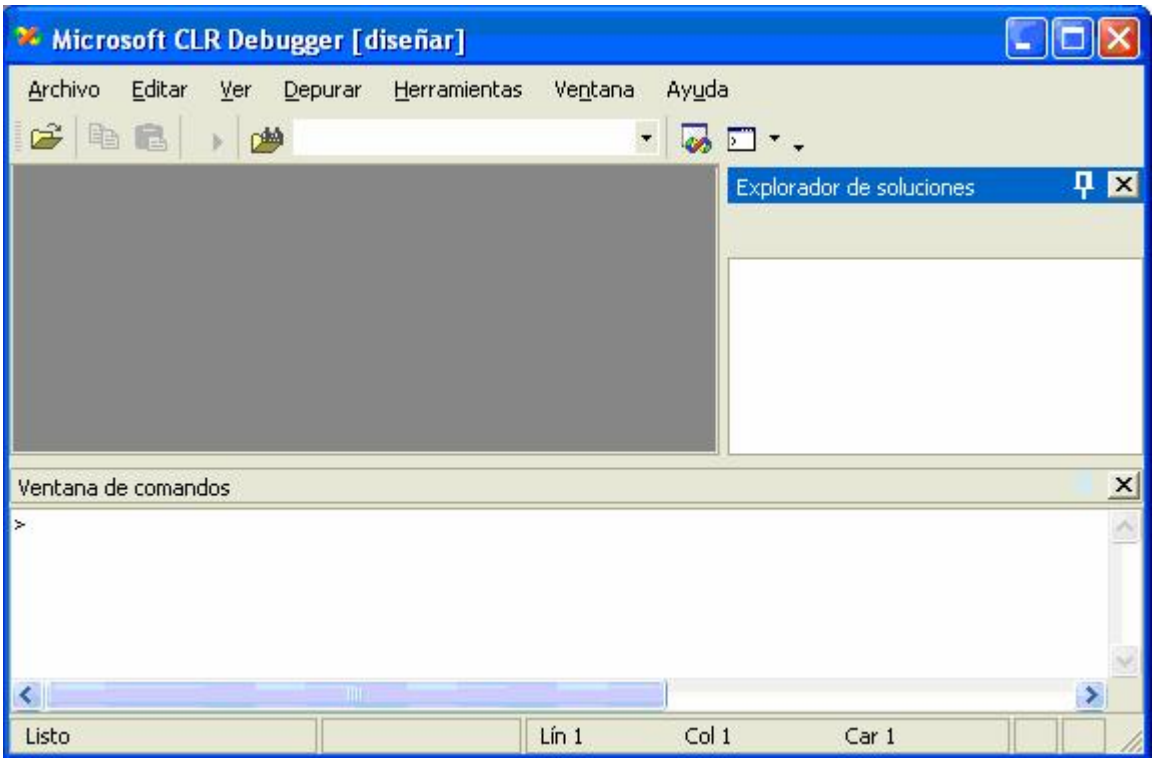


Figura 3.26. Microsoft CLR Debugger.

Una vez ejecutado el ***depurador***, se deberá acceder al menú ***Herramientas\Procesos de depuración***, al realizar esto se mostrará el cuadro de diálogo representado en la **Figura 3.27**.

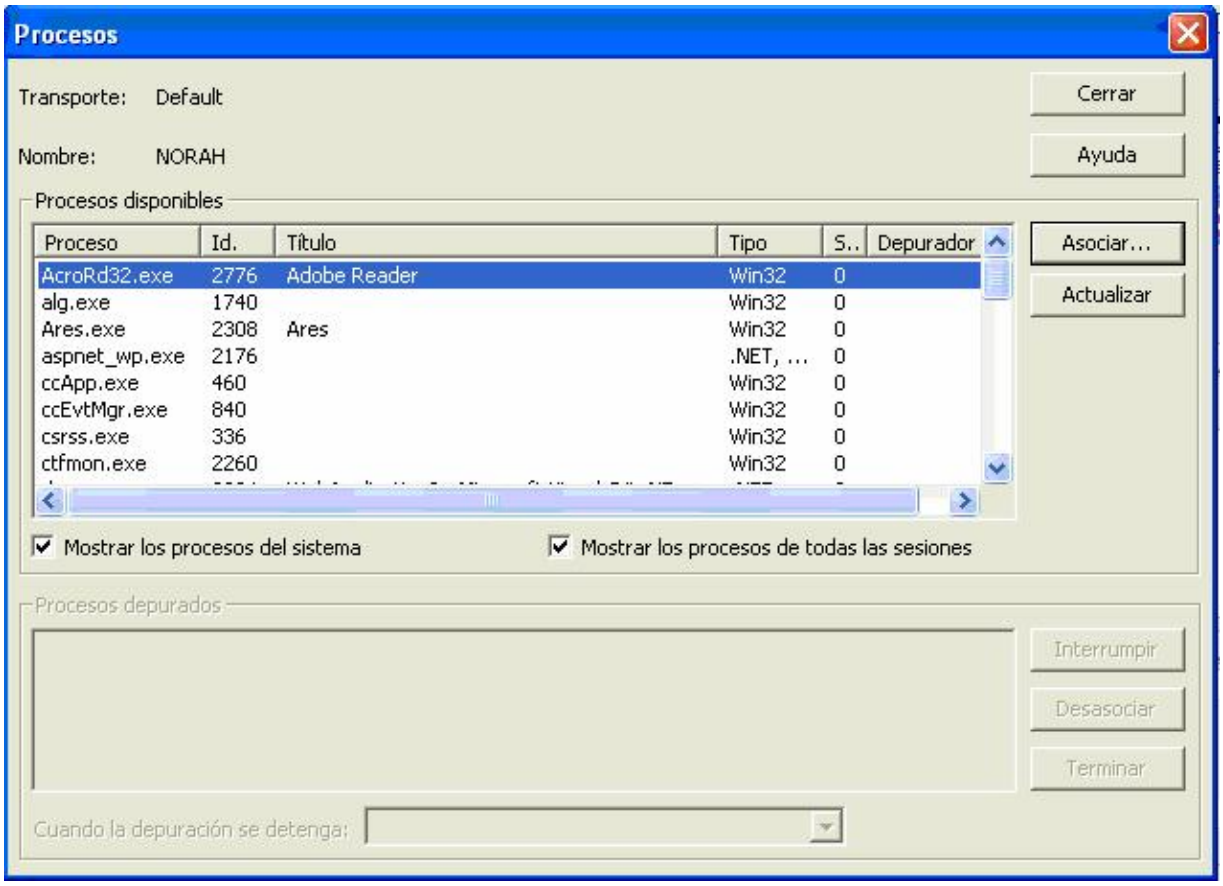


Figura 3.27. Ventana de Procesos.

En este cuadro de diálogo se deberá seleccionar la casilla **Mostrar los procesos del sistema** y enseguida se seleccionará el proceso correspondiente a la ejecución de la **página ASP .NET** que se desea depurar.

Específicamente el proceso generado por la **página ASP .NET** es: ***aspnet_wp.exe***, y una vez que ya se encuentra seleccionado se deberá *pulsar* el botón **Asociar**, indicando con esto que el depurador procederá a depurar el proceso indicado.

Finalmente se deberá cerrar el cuadro de diálogo y a continuación únicamente restará especificar en la *ventana del depurador* la **página ASP .NET** a depurar por medio de la opción de ***menú Archivo \ Abrir***.

Es importante señalar, que una vez que se tiene cargada la **página ASP .NET** en el depurador, es posible especificar ***puntos de ruptura***, que permitirán al programador analizar detalladamente el comportamiento de la aplicación desarrollada.

Para especificar dichos puntos, solo será necesario dar *doble clic* sobre la parte izquierda sombreada de la ventana, y el como resultado se obtendrá una línea marcada en donde será interrumpida la ejecución tal y como se muestra en la **Figura 3.28**.

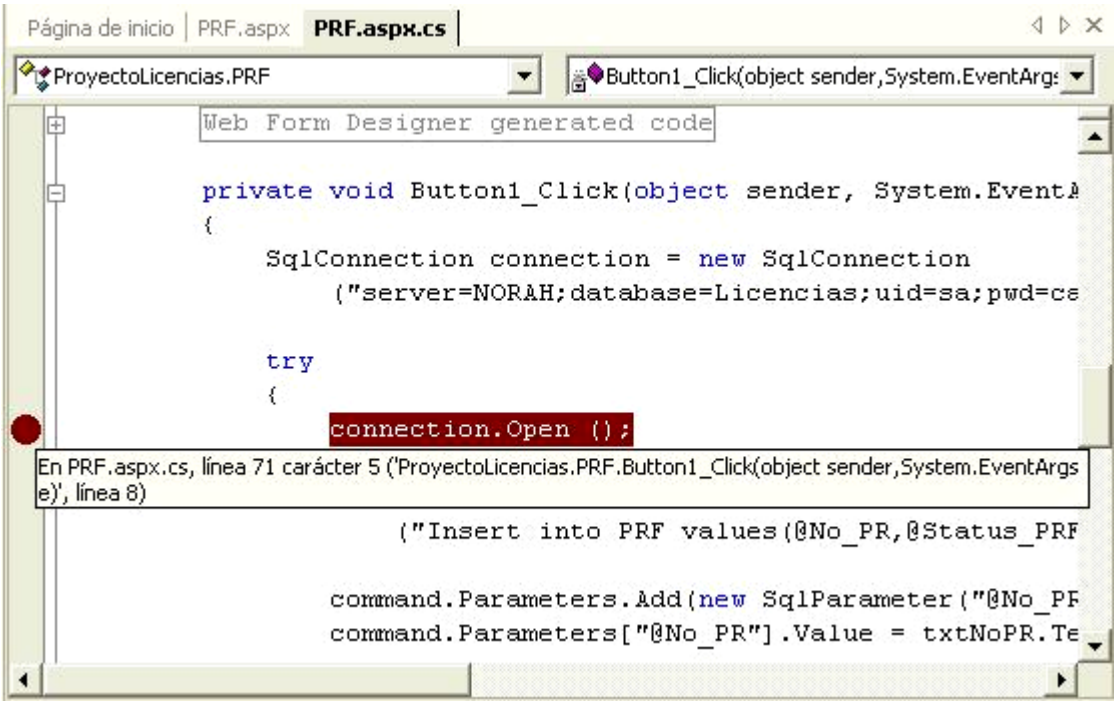


Figura 3.28. Especificación de punto de interrupción.

Cuando es terminada la ejecución del código, nuevamente se tiene el control sobre los pasos en la ejecución de la página, debido a que es posible indicar una depuración paso a paso.

3.12.2 Depuración con Visual Studio .NET

Dentro del **entorno de Visual Studio .NET** se encuentra la depuración como parte del mismo integrada con las herramientas de desarrollo.

Para llevar a cabo el proceso de **depuración** desde dicho entorno se deberá establecer como *página de inicio* la página que se desea depurar y activar la opción **debug** como se realizó en la depuración por medio de **SDK Framework**.

En la vista que proporciona el **código HTML** es posible establecer los ***puntos de ruptura*** y una vez agregados se deberá ir al menú ***Depurar\Iniciar***, cargándose de forma automática la **página ASP .NET** especificada como página de inicio y realizándose la depuración de esta.

Enseguida se muestra la ventana que representa el ***entorno de depuración de Visual Studio .NET*** en la **Figura 3.29**, realizando la depuración correspondiente a una **página ASP .NET** indicando un ***punto de ruptura***.

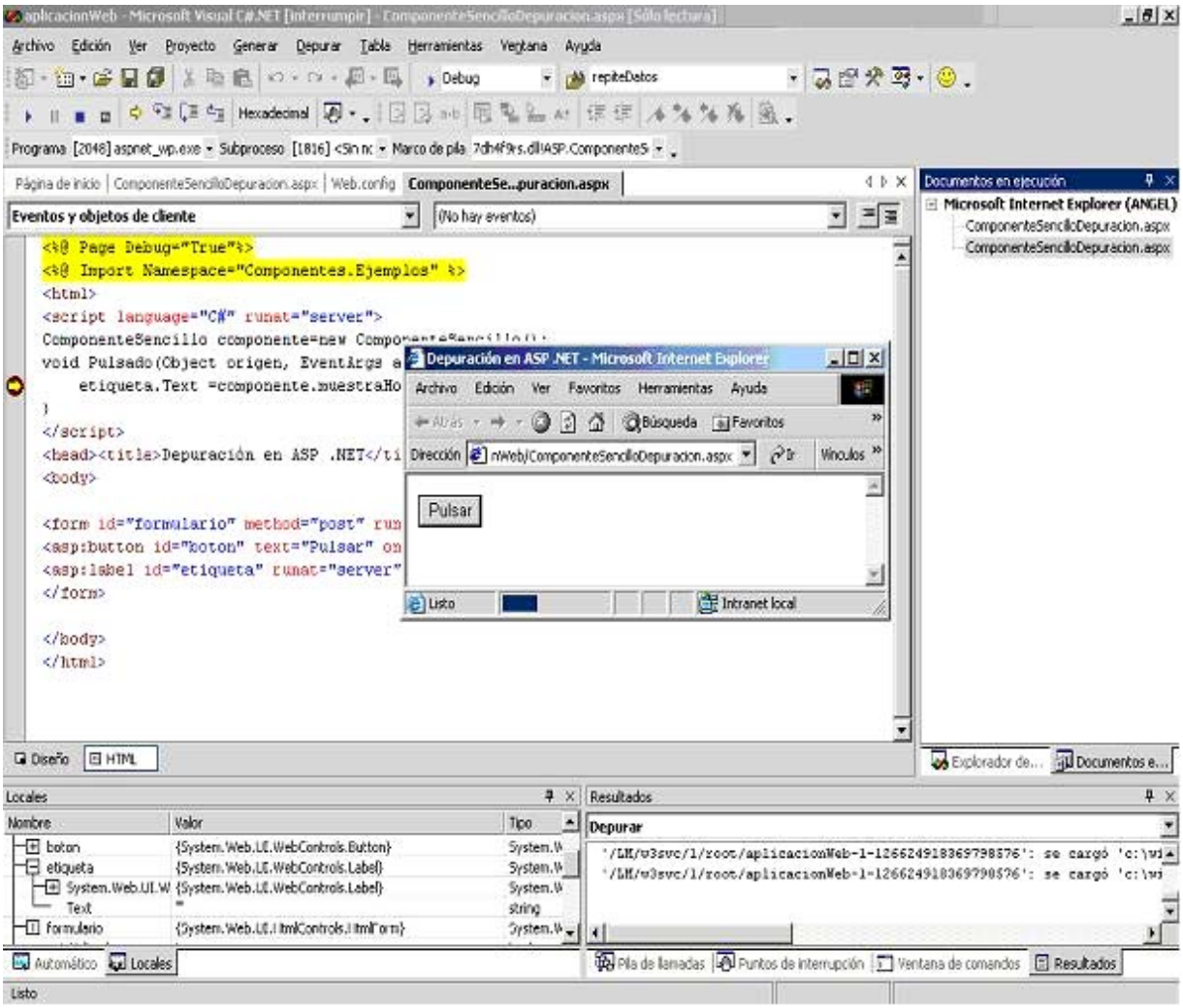


Figura 3.29. Entorno de depuración Visual Studio .NET.

Como se pudo observar, el **entorno Visual Studio .NET** nos permite una manera sencilla de llevar a cabo la depuración de una aplicación, y a pesar de que es muy similar a la depuración ofrecida por el **depurador CLR**, el **entorno Visual Studio .NET** permite la depuración remota, además de la modificación del código fuente de las paginas mientras se están ejecutando.

CAPÍTULO 4

Material de apoyo para ASP.NET

4.1 Microsoft ASP .NET Web Matrix Project

Microsoft a provocado sorpresa en los desarrolladores con su producto ***Microsoft* ASP .NET Web Matrix Proyect**, ya que es una herramienta fácil de utilizar para el desarrollo de aplicaciones Web con **ASP .NET**. Este proyecto une a la comunidad **ASP .NET** utilizando características como: *mensajería instantánea, integración de un cliente de Chat y una ayuda basada en comunidades*. Además, cuenta con un diseñador de formularios Web tipo **WYSIWYG *(What You See Is What You Get -lo que ves es lo que obtienes-)***.

La finalidad con la que fue diseñado este producto se resume en tres puntos fundamentales:

- Conectar a los desarrolladores a la **comunidad ASP .NET**.
- Ayudar a los desarrolladores a facilitar su opinión sobre nuevas características.
- Experimentar con nuevas ideas y posibilidades.

Éste producto de 1.2 MB, permite desarrollar **páginas ASP .NET** de forma similar a como se haría en **Visual Studio .NET**, sin embargo, cuenta con algunas diferencias claras. Es un entorno no tan completo como **Visual Studio .NET** pero aumenta la productividad de desarrollo de **páginas ASP.NET** y otros documentos.

ASP .NET Web Matrix Project es considerado como un complemento a **Visual Studio .NET** ya que ha sido desarrollado pensando en el *desarrollador aficionado*, además, de que es un entorno perfecto para mejorar las habilidades con **ASP .NET** y así posteriormente migrar a **Visual Studio .NET**.

4.1.1 Características de ASP .NET Web Matrix

ASP .NET Web Matrix presenta una variedad de características técnicas y referentes al entorno de trabajo que benefician a los desarrolladores que optan por el uso de este producto. A continuación se describen dichas características:

1. *Diseñadores WYSIWYG.-* Permite crear aplicaciones web arrastrando y soltando controles desde la caja de herramientas de **Web Matrix**.

2. *Páginas ASP .NET y el editor de HTML.-* Es posible insertar filas y columnas y combinar o partir las celdas. Los controles de usuario están disponibles en la *vista de diseño* cuando son utilizados en **documentos .aspx** o **.ascx**. Permite también restaurar o corregir su disposición usando el comando en la *ventana de Propiedades*, además de copiar y pegar contenido entre las ***vistas de Diseño* y *HTML***.

3. *Soporte para múltiples lenguajes.-* Web Matrix soporta ***Visual J# .NET***, además de otros lenguajes de programación como ***Visual C# .NET*** y ***Visual Basic .NET***.
Web Matrix tiene un modelo simplificado para los ***plug-ins***, como *Add-Ins*, *Code Wizards (Code Builders)*, y *Document Wizards*.

4. *Class Browser.-* El **Class Browser** soporta nuevos tipos de búsqueda: *nombres, tipos, etcétera.* Un nuevo *Add-In*, ***AssemblyInfo***, muestra información de versión , dependencias, y recursos del ***assembly***. El **Class Browser** utiliza la ayuda del **.NET Framework SDK**.

5. *Ejemplos de Aplicaciones.-* El Proyecto **Web Matrix** facilita varios ejemplos de aplicaciones y páginas incluyendo: *páginas marcadas por fecha, servicios web, caching de salida, páginas de login, y más*.

6. *Mejoras en el área de trabajo.-* Las **conexiones FTP** soportan raíces Web. Útil cuando el *directorio del Web* es un subdirectorio dentro del *directorio del* ***FTP***.
La ventana del área de trabajo exhibe inmediatamente los archivos o las carpetas nuevos agregados a una carpeta abierta.

7. *Integración de Datos.-* Es posible crear fácilmente aplicaciones web orientadas a datos al integrar *Bases de Datos* **MSDE**, **SQL Server** o **Access** con aplicaciones web. Simplemente arrastrando y soltando las tablas de datos desde el diseñador para conectarlas a su página. Los **Data Code Builders *(ahora llamados Code Wizards)*** trabajan contra *Bases de Datos* de **Access** y **SQL Server**. Es posible abrir las tablas y los procedimientos almacenados como documentos y utilizar **Web Matrix** para trabajar en ellos.

8. *Galería de Controles.-* Las aplicaciones pueden mejorarse y ser desarrolladas mas rápidamente usando ***controles de la galería en línea*** en ***http://www.asp.net****.*

9. *Generador Proxy de Servicios Web XML.-* Permite integrar soporte para *Servicios Web* en las aplicaciones de forma sencilla o consumir los *Servicios Web* hospedados en otro servidor.

10. *Visor de Clases .NET.-* Permite navegar rápidamente por la ***biblioteca de clases del .NET Framework*** para encontrar las clases necesarias para construir una aplicación.

11. *Editor de texto coloreado de rica sintaxis.-* Aprender ***VB.NET***, ***J#*** y ***C#*** es sencillo con el editor de texto coloreado.

12. *Ayudantes y plantillas basados en tareas.-* Ayuda para guiar a los usuarios a través de las tareas más comunes.

13. *Constructores de Código.-* Es posible especificar pocos parámetros y el código es generado automáticamente.

14. *Trabajo por archivos (no se requiere un proyecto).-* El trabajo por archivos proporciona una *alternativa más ligera* al habitual trabajo por proyectos.

15. *Soporte para alojamiento FTP.-* Es posible alojar las aplicaciones en una selección de terceras compañías vía **FTP**.

4.1.2 Instalación de ASP .NET Web Matrix Project

Para la instalación de ***Microsoft* ASP .NET Web Project**, será necesario tener un entorno con un sistema operativo ***Microsoft* Windows 2000 Profesional** y Server con el **Service Pack 2** o **Windows XP Profesional** o **Home**, el navegador Web ***Microsoft* Internet Explorer 5.5** o superior y ***Microsoft* .NET Framework 1.0**.

Para descargar el archivo de instalación se deberá acudir al sitio oficial web de **Web Matrix** en la siguiente dirección: ***http://www.asp.net/webmatrix/***.

Una vez descargado el archivo de instalación de 1.2 MB, se deberá hacer *doble clic* sobre él para comenzar la instalación. La aplicación que se ejecuta para dicha instalación se muestra en la **Figura 4.1**.

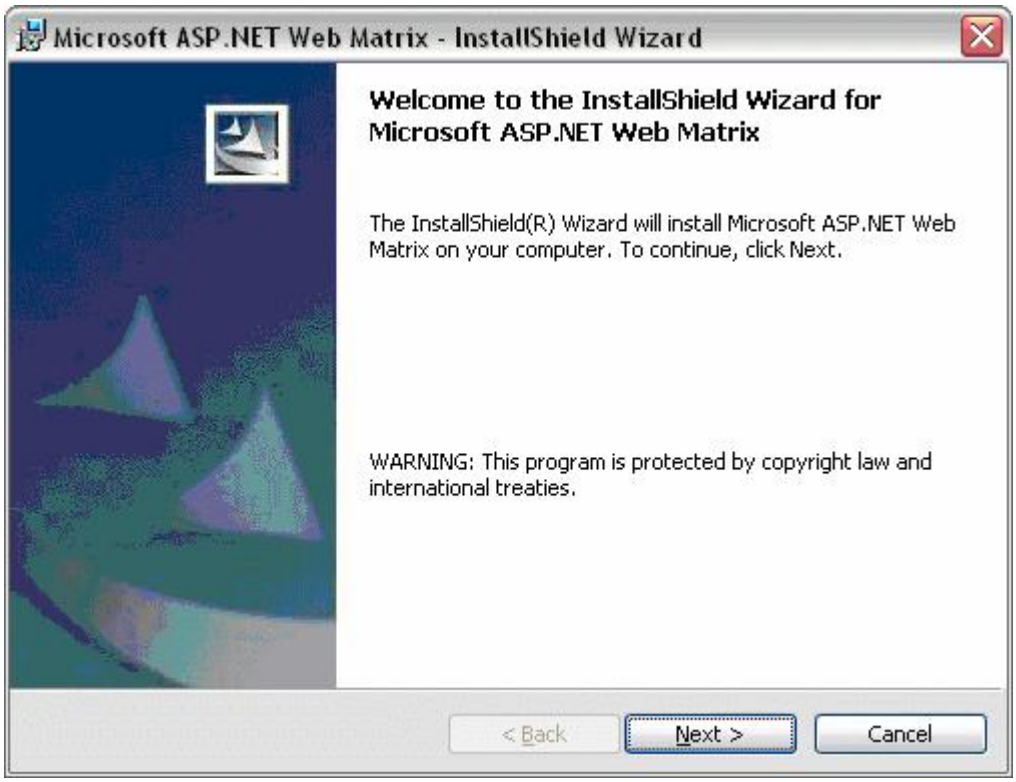


Figura 4.1. Aplicación para la instalación de Web Matrix.

La aplicación se instala de forma automática sin necesidad de ninguna acción extra. Adicionalmente es posible instalar un motor de bases de datos como ***Microsoft* SQL Server 7.0 con SP 2** o **SQL Server 2000**, o bien, ***Microsoft* MSDN**.

La descarga de ***Microsoft* MSDE *(programa gratuito y reducido de un Servidor SQL Server)*** se puede realizar de la página web oficial de **Web Matrix** . **MSDE** ocupa aproximadamente 33 MB. También se pueden desarrollar páginas para dispositivos móviles si se descarga el entorno ***Microsoft* Internet Toolkit** que ocupa aproximadamente 4MB.

Por lo tanto **Web Matrix** puede contar con la funcionalidad de un ***Servidor de Acceso a fuentes de datos*** y la capacidad para desarrollar **páginas ASP .NET** para dispositivos móvil añadiendo estas funciones, logrando así, un entorno de desarrolla mas completo.

4.1.3 Entorno de Desarrollo

***Microsoft* ASP .NET Web Matrix Project** cuenta con ciertas características que hacen que su entorno sea atractivo y con algunas otras que no posee ***Microsoft* Visual Studio .NET**.

El entorno de desarrollo de ***Microsoft* ASP .NET Web Matrix Project** tiene un aspecto muy similar a ***Microsoft* FrontPage** y a ***Microsoft* Visual Studio .NET**. En la **Figura 4.2** puede apreciarse este entorno.

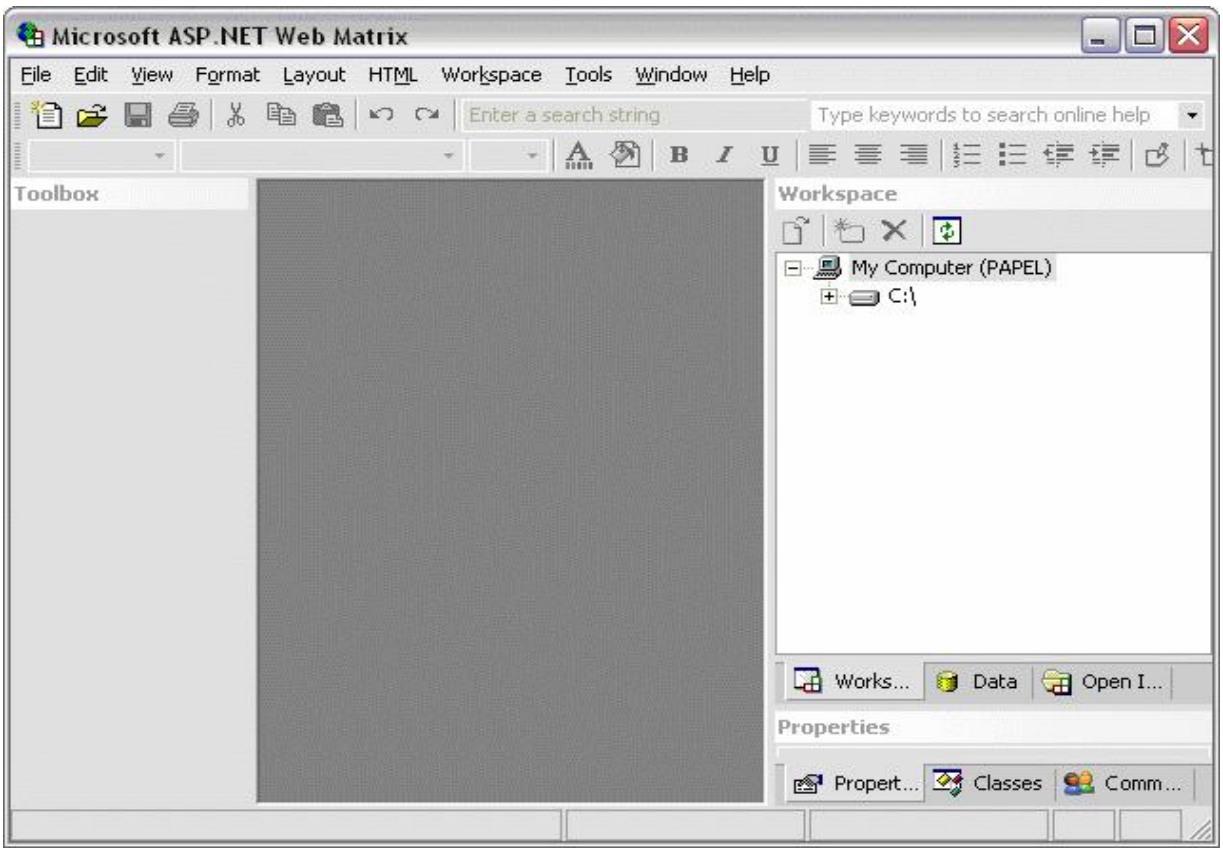


Figura 4.2. Entorno de desarrollo de Microsoft ASP .NET Web Matrix.

Este entorno fue desarrollado con el lenguaje de programación ***C#*** y ***Microsoft* .NET Framework**, por lo que se demuestra la potencia del lenguaje ***C#*** y las capacidades de **.NET Framework** como entorno para desarrollo de aplicaciones.

Con **Web Matrix** es posible desarrollar **páginas ASP .NET** con ***C#*** o ***Visual Basic .NET***, además, permite la conexión con servidores por medio de **FTP** de manera que las **páginas ASP .NET** desarrolladas pueden subirse al servidor directamente.

Uno de los aspectos mas importantes de **Web Matrix** es que *no necesita ningún servidor web* para poder ejecutar las **páginas ASP .NET** desarrolladas, ya que contiene un ***servidor web virtual*** que permite la ejecución el cual se conecta generalmente al *puerto 8080*.

Respecto al entorno **Visual Studio .NET**, **Web Matrix** *no soporta el* ***entorno de depuración*** *ni cuenta con la posibilidad de desarrollar* ***aplicaciones Windows***. Sin embargo, un aspecto muy particular de **Web Matrix** es su orientación a la ***comunidad de desarrolladores***.

Si se cuenta con ***Microsoft*** **Messenger**, al instalar ***Microsoft*** **ASP .NET Web Matrix Project** aparecerá un nuevo grupo de contactos denominado ***My ASP .NET Contacts.*** Este grupo de contactos se podrá utilizar para agregar nuevos contactos relacionados con **ASP .NET**, logrando así, compartir información y conocimientos sobre **ASP .NET**.

En la **Figura 4.3** se muestra el grupo ***My ASP .NET Contancts*** añadido a ***Microsoft*** **Messenger**.

Figura 4.3. My ASP .NET Contacts en Microsoft Messenger.

4.1.4 Desarrollo de una página ASP .NET

Al iniciar el Programa **Web Matrix** se visualiza la ventana que permite crear una **página ASP .NET** u otro tipo de proyectos. Para crear una **página ASP .NET** se deberá seleccionar la opción sombreada que se muestra en la **Figura 4.4.**

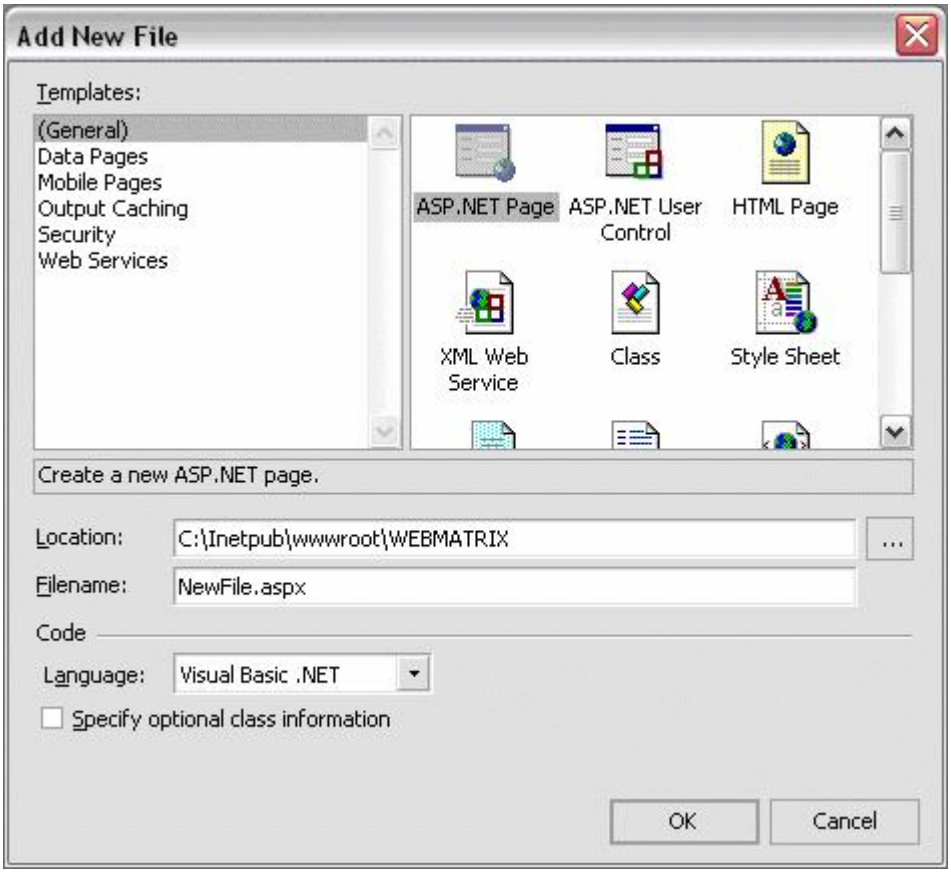


Figura 4.4. Ventana para crear un nuevo proyecto en Web Matrix.

Dejando el nombre de la página por defecto se *pulsará* el botón ***OK*** que se muestra en la figura anterior para crear la **página ASP .NET** en la ruta deseada.

Para la página de prueba que será creada se escribirá un texto inicial *(cualquiera)* dentro del formulario web como si se escribiera un texto en un editor de textos normal. En este caso se escribió el texto ***"Página ASP .NET con Microsoft ASP .NET Web Matrix"***. Una vez escrito el texto, es posible modificar sus propiedades seleccionándolo y utilizando los botones de formato de texto.

Enseguida, se seleccionará un control ***Label Web Control*** para insertarlo en el formulario web arrastrándolo sobre él. De la misma manera se agregará un control ***Button Web Control***.

Para modificar las propiedades de los controles creados se deberá seleccionar cada uno y acceder a la **ventana de Propiedades**. En la **Figura 4.5**, se muestra la modificación de la presentación del formulario Web.

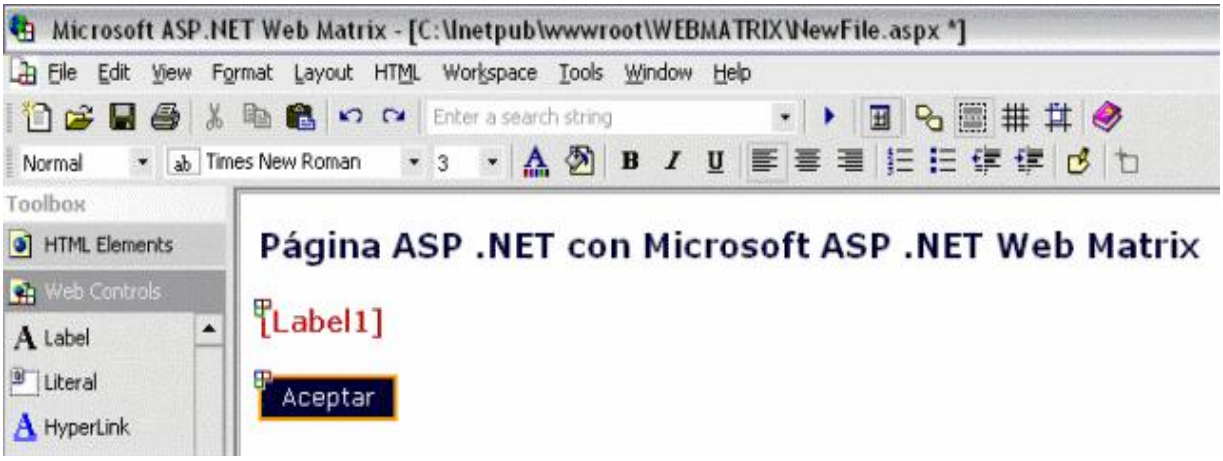


Figura 4.5. Modificación de la presentación del formulario web.

Para agregar funcionalidad a la página de prueba creada, se deberá hacer *doble clic* sobre el botón ***Control Button Web Control***. De esta forma se abrirá la *ventana de código*, donde se escribirá el código correspondiente.

El código que se agregara se detalla a continuación:

```
Sub Button1_Click(sender As Object, e As EventArgs)
Label1.Text = "Hola Mundo, este es mi primer ejemplo con Web
Matrix"
End Sub
```

Finalmente se ejecutará la página haciendo *doble clic* sobre la tecla de ejecución que coincide con la tecla F5, apareciendo entonces una ventana que se muestra en la **Figura 4.6** en la que se deberá indicar que servidor será utilizado. En este caso, se

utilizará el **Servidor Web Matrix**, por lo que se pulsará el botón ***Start*** para ejecutar dicho servidor.

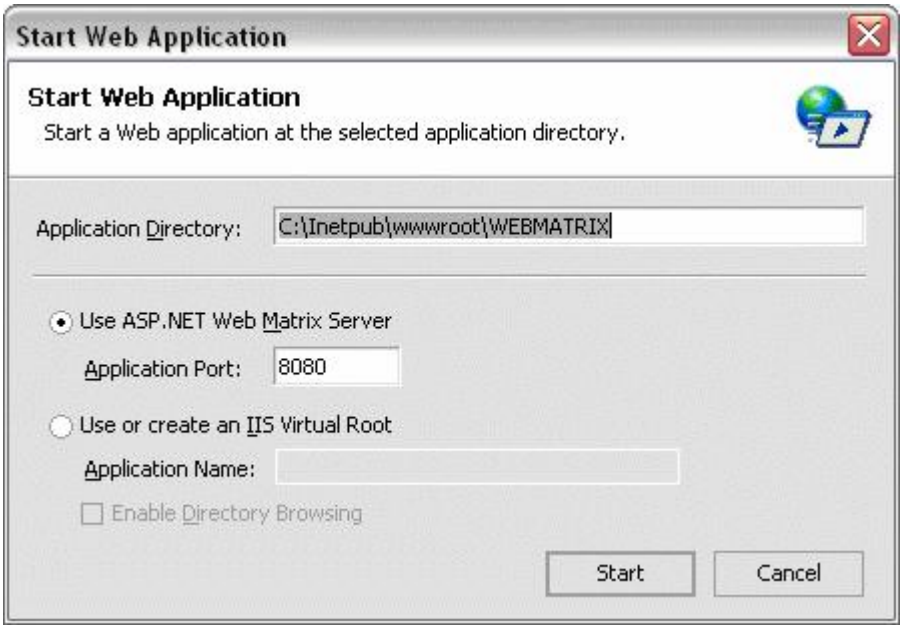


Figura 4.6. Ventana para selección del Servidor.

La aplicación creada se muestra en ejecución en la **Figura 4.7.**

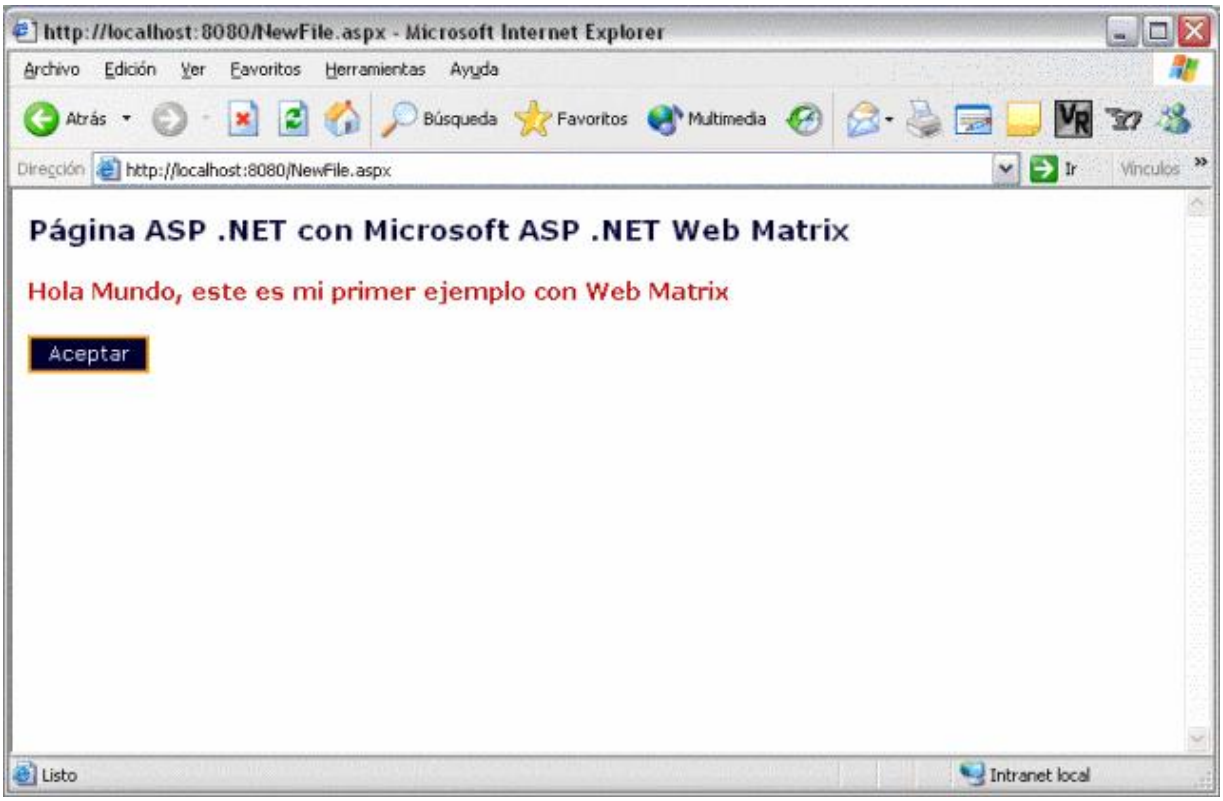


Figura 4.7. Aplicación ASP .NET en ejecución.

Cabe mencionar, que **Web Matrix** crea un archivo con extensión **.aspx** correspondiente a la extensión de las **páginas ASP .NET**, dentro del cual se encuentra todo el código de la página junto al **código HTML**.

A continuación se muestra el código generado al crear la aplicación:

```
<%@ Page Language="VB" %>
<script runat="server">

    ' Insert page code here
    '

    Sub Button1_Click(sender As Object, e As EventArgs)
```

```
    Label1.Text = "Hola Mundo, este es mi primer ejemplo con
Web Matrix"
    End Sub

<script>
<html>
<head>
</head>
<body>
<form runat="server">
<p>
<strong><font face="Verdana" color="#000040" size="4">
Página ASP .NET con Microsoft ASP .NET Web Matrix
</font></strong>
</p>
<p>
<asp:Label id="Label1" runat="server" Font-Names="Verdana"
      Font-Size="11pt" Font-Bold="True" ForeColor="#C00000">
</asp:Label>
</p>
<p>
<asp:Button id="Button1" onclick="Button1_Click" runat="server"
      Text="Aceptar" Font-Names="Verdana" ForeColor="White"
      BackColor="#000040" BorderColor="Orange"
BorderWidth="2px" BorderStyle="Solid">
</asp:Button>
</p>
<!-- Insert content here -->
</form>
</body>
</html>
```

4.2 Kits de Inicio para ASP.NET

Los **Kits de inicio para ASP .NET** explican cómo crear una solución completa de aplicaciones empleando las tecnologías ofrecidas por **ASP .NET**. Cada una de las aplicaciones está orientada a un puñado de tecnologías de **ASP .NET** y muestra cómo usarlas para crear una solución práctica.

Cuando surge una ***nueva tecnología***, los desarrolladores tienen dificultades para entender todas las características y ventajas que ofrece, debido a que presenta *nuevos conceptos* con los que los desarrolladores no están familiarizados. También las dificultades surgen a raíz de la *mejora de una tecnología existente*, ya que los desarrolladores no reconocen inmediatamente las nuevas ventajas.

ASP .NET incluye algunos conceptos nuevos que talvez no conozcan los desarrolladores y al mismo tiempo cuenta con conceptos que son la evolución de la versión anterior **ASP**. Es por ello, que debido a la complejidad de algunos aspectos de **ASP .NET** es necesario ofrecer aplicaciones prácticas de ejemplo que muestren como crear una solución completa de aplicaciones.

En los **Kits de inicio** se han incluido muestras, ya que debido a la gran variedad de características de **ASP .NET**, no serviría de mucho crear una aplicación enorme que mostrara todos los aspectos de **ASP .NET**, en su lugar se ofrecen *aplicaciones de pequeño tamaño* en donde cada una demuestra un conjunto distinto de las características de **ASP .NET**.

Los **ASP Starter Kits** o **Kits de inicio** *son cinco ejemplos de aplicaciones y código en* ***ASP .NET***, de forma que pueden ser integrados fácilmente a los proyecto creados previamente:

1. ***Kit de inicio Portal.***
2. ***Kit de inicio Comerse.***

3. ***Kit de inicio Community .***
4. ***Kit de inicio Time Tracker.***
5. ***Kit de inicio Reports.***

4.2.1 Arquitectura de los Kits de inicio para ASP .NET

Tradicionalmente, las aplicaciones Web se diseñan y crean en tres capas lógicas:

- **Capa de acceso a bases de datos (DAL)**
- **Capa de lógica de negocio (BLL)**
- **Capa de presentación**

La **DAL** *hace referencia a la Base de Datos*, *los procedimientos almacenados y los componentes que proporcionan una interfaz para la Base de Datos*. La **BLL** *se refiere a los componentes que encapsulan la lógica de negocio de la aplicación*, mientras que la capa de presentación hace referencia a las páginas de aplicaciones Web.

Para crear aplicaciones distribuidas con las **tecnologías .NET** siguen siendo necesarias arquitecturas lógicas en tres capas con un diseño adecuado. En los **Kits de inicio para ASP .NET** se ha seguido este mismo concepto, pero con la excepción de que la **BLL** y los componentes de la **DAL** para la interfaz con la *Base de Datos se han combinado en una sola capa para reducir la complejidad de la aplicación*. La **Figura 4.8** muestra la arquitectura de las aplicaciones de los **Kits de inicio**.

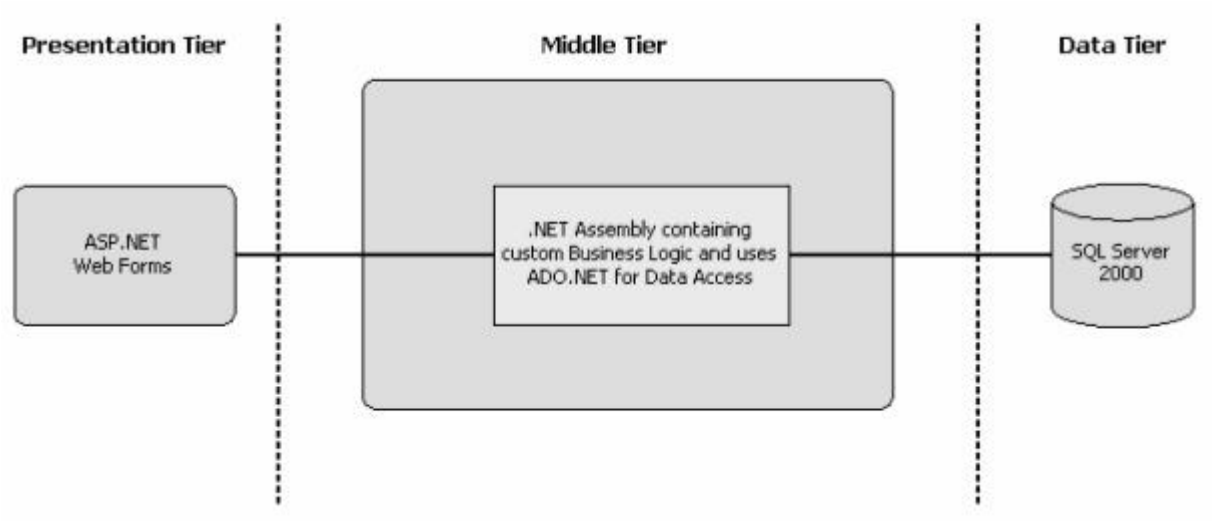


Figura 4.8. Arquitectura de las aplicaciones de los Kits de inicio para ASP .NET.

Existen múltiples opciones de distribución de *aplicaciones Web*, dependiendo de las condiciones de rendimiento, seguridad y mantenimiento de cada aplicación. La distribución de las aplicaciones de los **Kits de inicio** *se basa en dos capas físicas (combinando la capa de presentación y la capa intermedia en un servidor y la base de datos en otro)* para conseguir un **rendimiento óptimo**.

4.3 Kit de Inicio Portal

El **Kit de inicio Portal** muestra cómo se puede utilizar **ASP .NET** junto con ***Microsoft* .NET Framework** para crear aplicaciones de portal para intranet e Internet. El ejemplo muestra muchas de las características fundamentales de **ASP .NET** y proporciona también una aplicación de "procedimientos adecuados" que los desarrolladores pueden utilizar como base para crear sus propias aplicaciones **ASP .NET**. En la **Figura 4.9** se muestra el **Kit de Inicio Portal** en ejecución.

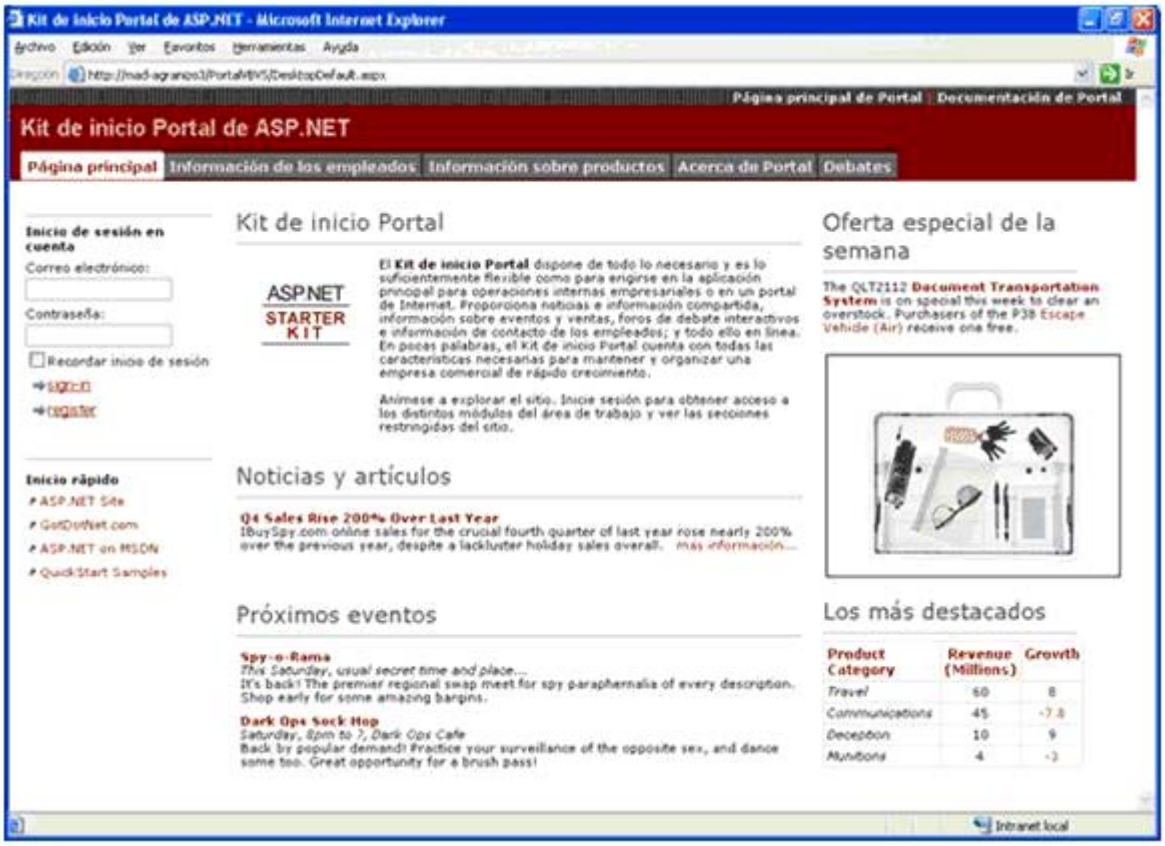


Figura 4.9. Entorno del Kit de Inicio Portal.

El **Kit de Inicio Portal** incluye muchas características que ofrece la **tecnología ASP .NET**, la cuales se mencionan a continuación:

- *Compatibilidad con los exploradores **Netscape** e **Internet Explorer***.

- *Compatibilidad con dispositivos móviles para **WAP (Wireless Application Protocol) / WML (Wireless Markup Language)** y dispositivos **Pocket Browser***.

- *Separación clara entre **código** y **contenido HTML** mediante **controles de servidor***.

- *Páginas creadas a partir de **controles de usuario cargados dinámicamente***.

- ***Caché de resultados configurable*** *para partes de páginas de portal*.

- *Arquitectura de aplicación de varios niveles*.

- *Acceso a datos* ***ADO .NET*** *mediante el uso de procedimientos almacenados de* ***SQL***.

- *Autenticación basada en Windows con nombre de usuario / contraseña en Active DS o NT SAM*.

- *Autenticación basada en formularios con una base de datos para nombres de usuario / contraseñas*.

- *Seguridad basada en funciones para controlar el acceso de usuarios al contenido del portal*.

4.3.1 Arquitectura del Kit de inicio Portal

El Portal utiliza una ***arquitectura*** de varios niveles y contiene dos orígenes de datos. Los parámetros de configuración se almacenan en el archivo ***PortalCFG.xml*** y el contenido para la aplicación se guarda en una *Base de Datos* de **SQL Server**.

El ***acceso a datos*** se realiza mediante un **ensamblado de *Microsoft* .NET** que proporciona acceso al origen de datos *a través de los procedimientos almacenados*. Además, para crear el área de trabajo del portal se utiliza una serie de ***ensamblados*** que controlan la seguridad y la configuración del portal.

La capa de presentación está formada por formularios ***Web Forms*** y ***controles de usuario*** que controlan la presentación y la administración de los datos del portal correspondientes al usuario.

4.3.2 Base de Datos del Portal

Todo el contenido del portal se almacena en una *Base de Datos* de **SQL Server**. Esto permite a los administradores de servidores distribuir la aplicación del portal entre varios servidores, obteniendo cada uno los datos de un solo almacén. La *Base de Datos* cuenta con un esquema sencillo, él cual se muestra en la **Figura 4.10.**

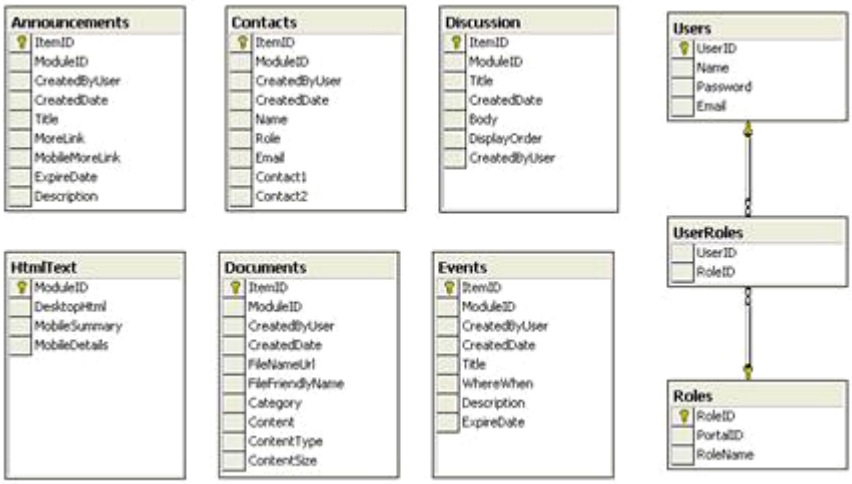


Figura 4.10. Esquema físico de la Base de Datos.

4.3.2.1 Procedimientos almacenados

El portal utiliza ***procedimientos almacenados*** para encapsular todas las *consultas* de la base de datos. El uso de ***procedimientos almacenados*** *permite separar claramente la Base de Datos de la capa intermedia de acceso a datos*. Esto facilita a su vez el mantenimiento, ya que los cambios efectuados en el esquema de la *Base de Datos* permanecen invisibles para los componentes de acceso a datos.

Los ***procedimientos almacenados*** también proporcionan ventajas en cuanto al *rendimiento*, ya que se optimizan cuando se ejecutan por primera vez y se mantienen en memoria para llamadas posteriores.

4.3.3 Esquema de configuración XML del Portal

El esquema basado en el archivo ***PortalCFG.xml*** contiene todos los ***parámetros de configuración*** del portal. El esquema es simple y fácil de entender. El archivo de configuración **XML** almacena todas las definiciones de alto nivel de Portal, Ficha y Módulo.

Los parámetros de configuración se almacenan en una **caché** y ***GetSiteSettings()*** sólo lee el **archivo XML** si ha habido cambios en dichos valores. La **Figura 4.11** muestra el esquema físico.

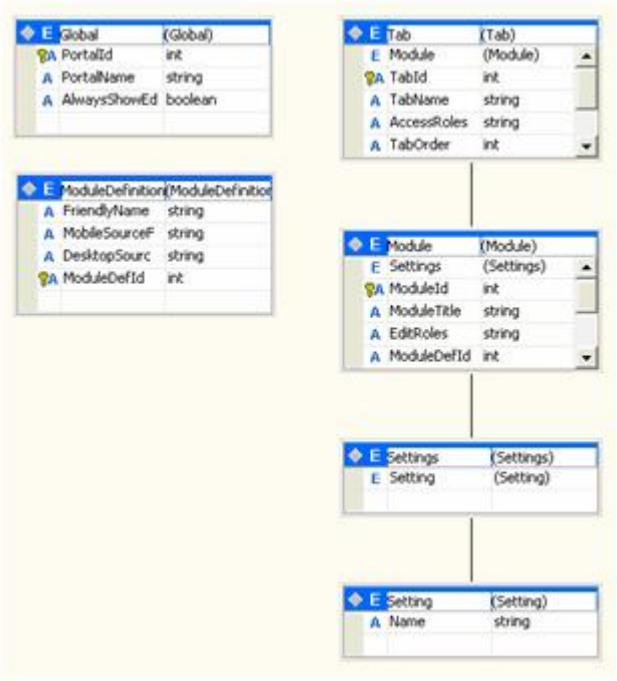


Figura 4.11. Esquema de PortalCFG.xsd.

4.3.4 Módulos del Portal

Los ***módulos del portal*** proporcionan el contenido real del **Kit de inicio Portal**. Los módulos *son controles de usuario que heredan la clase base* ***PortalModuleControl****, que proporciona la comunicación necesaria entre los módulos y el área de trabajo del portal subyacente*. De forma predeterminada, el portal tiene once módulos integrados que se pueden utilizar desde el principio, siete de los cuales se muestran en la **Figura 4.12**.

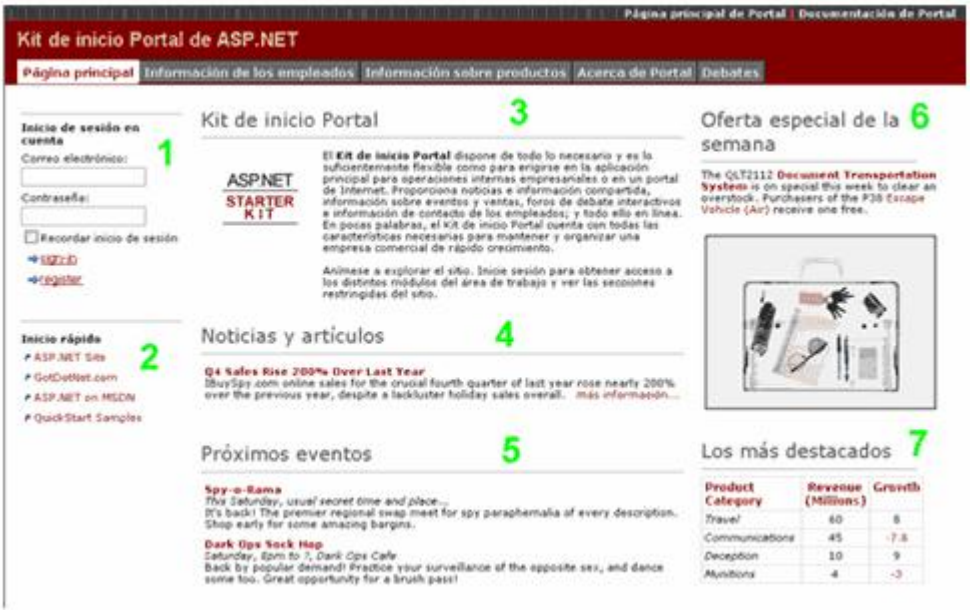


Figura 4.12. Módulos del Portal.

4.3.5 Administración del Portal

El portal tiene una ***herramienta de administración en línea*** que permite a los usuarios incluidos en la función "Admins" administrar la seguridad, el diseño y el contenido del portal. Los usuarios pertenecientes a la función "Admins" que inician sesión verán una ficha "Admin" que les permite el acceso a la herramienta de administración. La **Figura 4.13** muestra esta herramienta.

Figura 4.13. Administración del Portal.

La ***Administración del portal*** permite al usuario realizar distintas tareas de administración y configuración, *como agregar nuevos módulos, configurar fichas que aparecen en horizontal en la parte superior de la página y definir funciones de seguridad*.

4.4 Kit de inicio Commerce

El **Kit de inicio Commerce** *es una tienda de comercio electrónico para vender equipos ficticios de espionaje*. La aplicación muestra tareas básicas para realizar compras en línea, entre las que se incluyen: catálogo de productos, autenticación y

personalización de usuarios, cestas de la compra y confirmación de pedidos. En la **Figura 4.14** se muestra la aplicación en ejecución.

Figura 4.14. Entorno de Kit de inicio Commerce.

Las características que el **Kit de inicio Commerce** incluye respecto a la **tecnología ASP .NET** son:

- *Compatibilidad con los exploradores **Netscape** e **Internet Explorer**.*
- *Separación clara entre **código** y **contenido HTML** mediante **controles de servidor**.*
- *Páginas de catálogo de alto rendimiento que utilizan **caché de resultados**.*
- *Acceso a datos **ADO .NET** en tres niveles mediante el uso de **procedimientos almacenados de SQL**.*

- *Autenticación basada en formularios con una base de datos para nombres de usuario / contraseñas*.

- ***Servicios Web XML SOAP** para la entrada y el estado de pedidos **B2B***.

4.4.1 Arquitectura del Kit de inicio Commerce

La creación de aplicaciones distribuidas utilizando las **tecnologías .NET** continúa basándose en arquitecturas de tres niveles lógicos con un diseño adecuado. El **Kit de inicio Commerce** se ha creado siguiendo este mismo concepto, con la excepción de que las capas **BLL** y **DAL** se han combinado en una sola capa para reducir la complejidad de la aplicación.

En lugar de crear métodos sencillos de paso a través en el componente de lógica empresarial, se ha optado por combinar las dos capas. La **Figura 4.15** muestra la arquitectura del **Kit de inicio Commerce**.

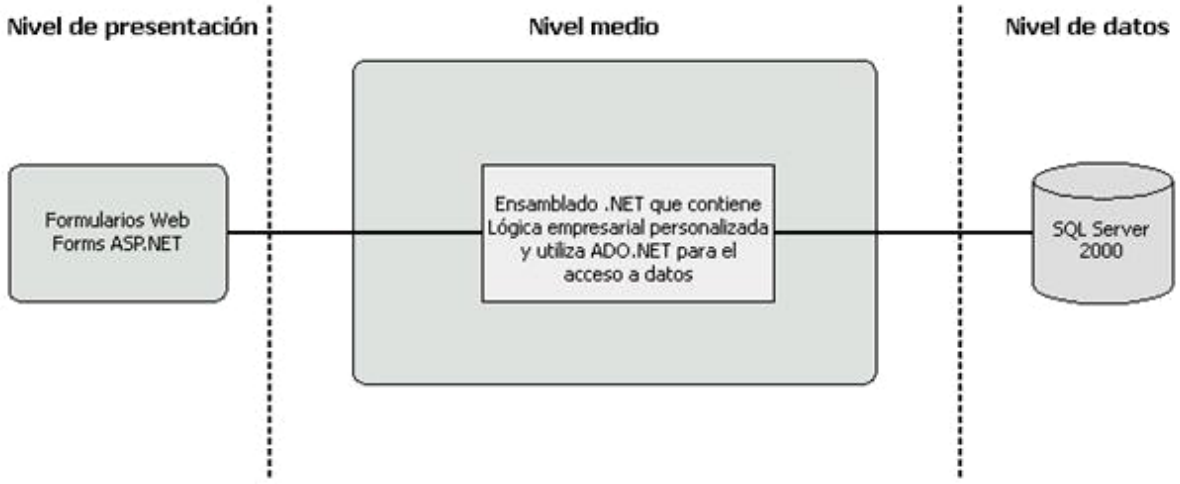


Figura 4.15. Arquitectura del Kit de inicio Commerce.

4.4.2 Base de Datos del Kit de inicio Commerce

Cuando se crea una aplicación, es recomendable hacerlo *en base a los datos identificando las distintas tablas y campos que se van a necesitar*, de esta manera resulta mas sencillo crear después el resto de la aplicación.

El diseño del esquema de la *Base de Datos* se realizó de acuerdo a los requisitos de la aplicación. El **Kit de inicio Commerce** dispondrá de un ***catalogo de productos*** para permitir a los clientes *explorar y agregar artículos a sus carros de compra*. Además, se podrán crear algunas funciones adicionales para *ventas cruzadas*, como una *lista de los artículos vendidos* a otros clientes de modo que el sistema implemente una *lista de los productos mas vendidos*.

En base a los requisitos anteriores, se identifican de forma rápida las tablas y los procedimientos almacenados que debe contener la *Base de Datos*. El **Kit de inicio Commerce** esta conformado por siete tablas, las cuales se muestran en la **Figura 4.16** que detalla el esquema de la *Base de Datos*.

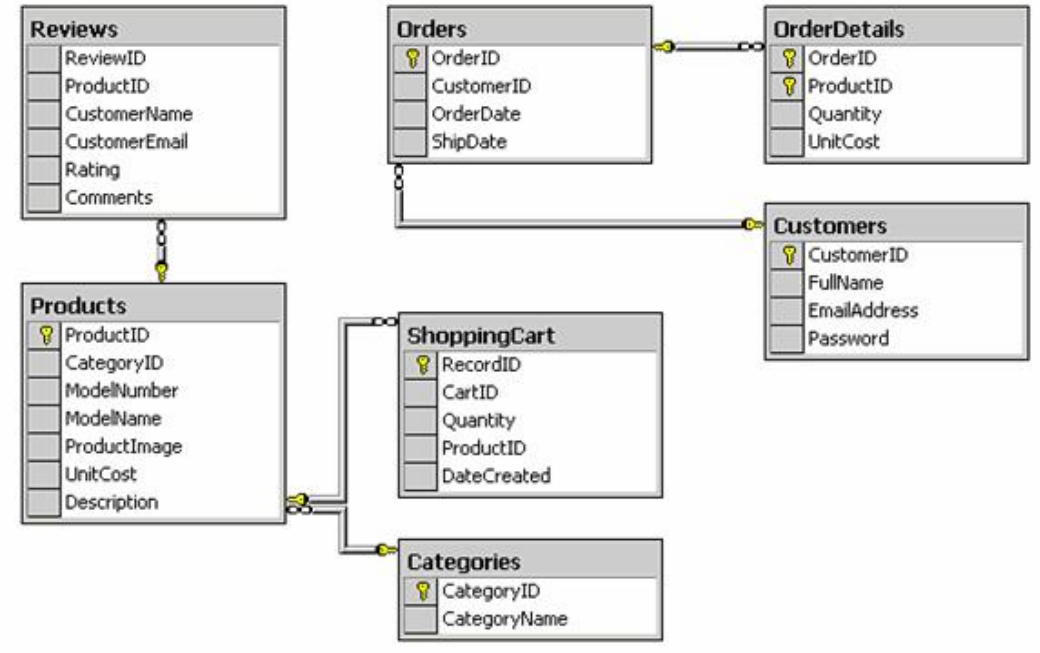


Figura 4.16. Esquema de la base de datos.

4.4.2.1 Procedimientos almacenados

El **Kit de inicio Commerce** utiliza ***procedimientos almacenados*** para encapsular todas las consultas de la *Base de Datos*. El uso de ***procedimientos almacenados*** permite separar claramente la *Base de Datos* de la capa de nivel medio de acceso a datos. Esto facilita a su vez el mantenimiento, ya que los cambios efectuados en el esquema de la *Base de Datos* permanecen invisibles para los componentes de acceso a datos.

Los ***procedimientos almacenados*** proporcionan también otras ventajas en cuanto al rendimiento, ya que se optimizan cuando se ejecutan por primera vez y se mantienen en memoria para llamadas posteriores.

Un ejemplo de uno de los ***procedimientos almacenados*** más complejos del **Kit de inicio Commerce** es el denominado ***"ShoppingCartAddItem"***. Este procedimiento tiene la misión de agregar un producto al carro de la compra de un cliente. A continuación se muestra el código de dicho procedimiento:

```
CREATE Procedure ShoppingCartAddItem
     (
          @CartID nvarchar(50),
          @ProductID int,
          @Quantity int
     )
As Declare @CountItems int

SELECT @CountItems= Count(ProductID)
FROM ShoppingCart
WHERE ProductID= @ProductID AND CartID= @CartID
```

```
IF @CountItems > 0 /* There are items - update the current
quantity */
        UPDATE ShoppingCart
        SET Quantity= (@Quantity + ShoppingCart.Quantity)
        WHERE ProductID= @ProductID AND CartID= @CartID
ELSE /* New entry for this Cart. Add a new record */
        INSERT INTO ShoppingCart
              (CartID, Quantity, ProductID)
        VALUES
              (@CartID, @Quantity, @ProductID)
GO
```

4.4.2.2 Capa de acceso a la Base de Datos

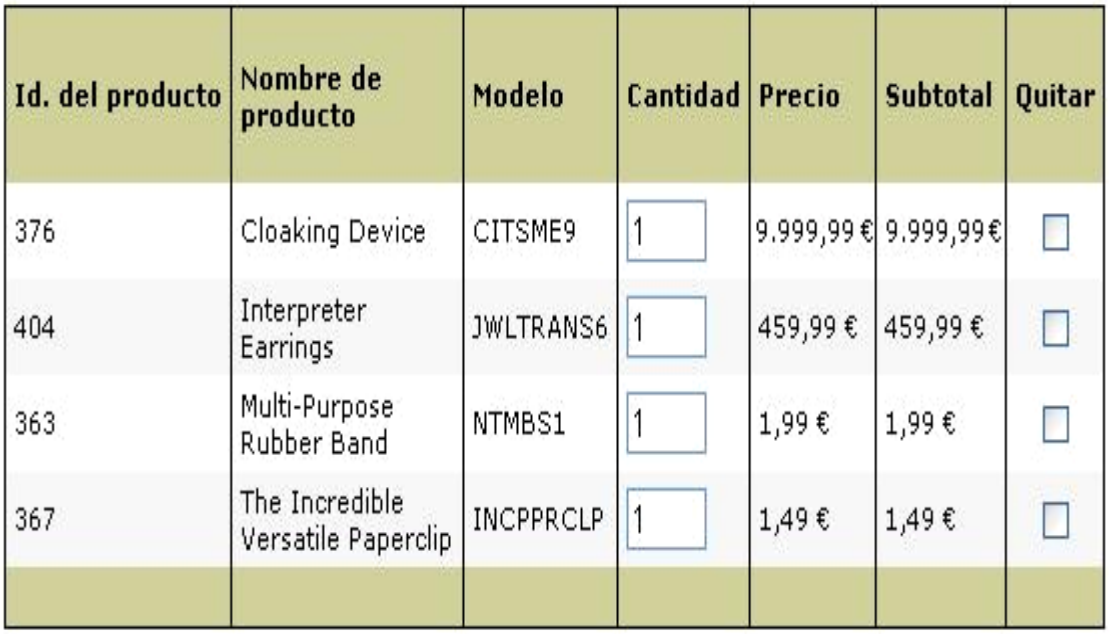
El **Kit de inicio Commerce** utiliza un componente de nivel medio para proporcionar comunicación entre los **formularios Web Forms ASP .NET** y la *Base de Datos* de **SQL Server**.

La implementación del componente de acceso a datos es sencilla haciendo que la aplicación se base en los datos y creando primero el esquema de la *Base de Datos*. El componente contendrá las clases siguientes que se asignan a las tablas de la *Base de Datos*:

- **CustomersDB.**
- **ProductsDB.**
- **ShoppingCartDB.**
- **ReviewsDB.**
- **OrdersDB.**

***Microsoft* .NET** permite agrupar las clases en un espacio de nombres que organiza el componente. En el **Kit de inicio Commerce** se ha creado un espacio de nombres

denominado **ASP.NET.StarterKit.Commerce** que contendrá todas las clases anteriores. Su estructura se muestra en la **Figura 4.17**



Id. del producto	Nombre de producto	Modelo	Cantidad	Precio	Subtotal	Quitar
376	Cloaking Device	CITSME9	1	9.999,99 €	9.999,99 €	☐
404	Interpreter Earrings	JWLTRANS6	1	459,99 €	459,99 €	☐
363	Multi-Purpose Rubber Band	NTMBS1	1	1,99 €	1,99 €	☐
367	The Incredible Versatile Paperclip	INCPPRCLP	1	1,49 €	1,49 €	☐

Figura 4.17. Espacio de nombres ASPNET.StarterKit.Commerce.

Los espacios de nombres no tienen que estar contenidos necesariamente en un archivo. En el **Kit de inicio Commerce** se ha creado un *archivo de origen* para cada clase *(con la excepción de OrderDetails y Orders)* y se han compilado todos ellos en un ensamblado, ***ASPNETCommerce.dll***.

4.5 Kit de inicio Community

El **kit de inicio Community (Comunidad)** *permite crear rápidamente un sitio Web de una comunidad, como el sitio de un grupo de usuarios, un sitio con recursos para desarrolladores o un sitio de noticias*. En la **Figura 4.18** se muestra el entorno del **Kit de inicio Community**.

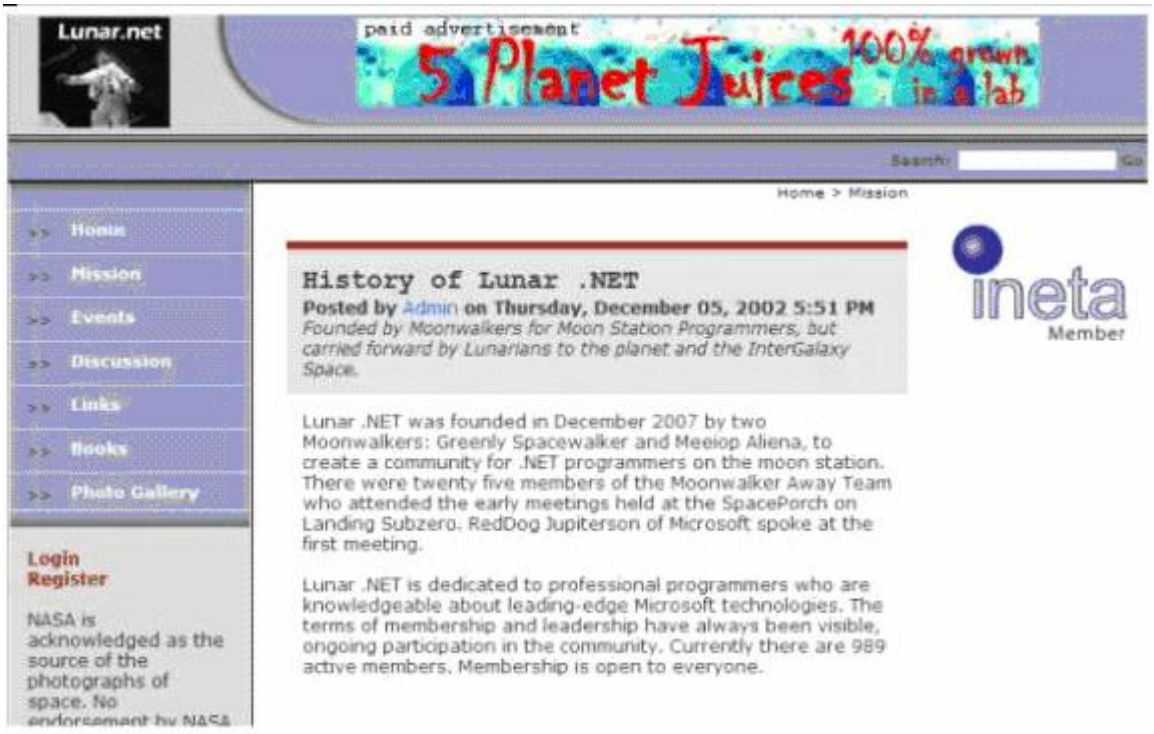


Figura 4.18. Entorno del Kit de inicio Community.

4.5.1 Módulos de Kit de inicio Community

El **kit de inicio Community** admite nueve módulos de contenido estándar:

1. Articles *(artículos)* .
2. Books *(libros).*
3. Events *(eventos).*
4. Links *(enlaces).*
5. Downloads *(descargas).*
6. Photo Gallery *(galería de fotos).*
7. Parent Section *(sección primaria).*
8. HTML Page *(página HTML).*
9. Static Page *(página estática).*

4.5.1.1 Módulo Articles

El módulo **Articles** se puede utilizar para mostrar contenido de texto, como artículos de noticias, mensajes de los usuarios o preguntas en el sitio de la comunidad.
El módulo **Articles** consta de cuatro páginas principales:

- **Página Article Section (Artículos en sección):** Muestra una lista de artículos en la sección.

- **Página Article (Artículo):** Muestra un artículo concreto de la sección.

- **Página Add Article (Añadir artículo):** Permite añadir un artículo nuevo a los usuarios que tengan los permisos correspondientes.

- **Página Edit Article (Editar artículo):** Permite editar un artículo a los usuarios que tengan los permisos correspondientes.

El módulo **Articles** acepta todas las funciones estándar del marco de trabajo para el **kit de inicio Community**, como:

- ***Seguridad:*** Es posible hacer que sólo ciertos miembros de la comunidad tengan autorización para ver, añadir, editar o borrar un artículo.

- ***Moderación:*** Cuando se modera una sección de artículos, sólo aparecen los artículos que hayan sido aprobados.

- ***Comentarios:*** Los usuarios pueden añadir comentarios a los artículos.

- ***Evaluaciones***: Los usuarios pueden evaluar el contenido de los artículos como bueno o malo.

- ***Notificaciones:*** Se pueden enviar a los usuarios notificaciones automáticas por correo electrónico cuando aparece en la sección un artículo nuevo o un comentario a un artículo.

4.5.1.2 Módulo Books

El módulo **Books** se puede utilizar para dar una lista de los libros que hay en el sitio de la comunidad. Al publicar un libro se puede añadir una imagen de la cubierta. También es posible *añadir un enlace* a un lugar donde se pueda comprar el libro *(por ejemplo, Barnes & Noble o Amazon)*.

El módulo **Books** consta de cuatro páginas principales:

- **Página Book (Libros en sección):** Muestra una lista de libros en la sección.

- **Página Book (Libro):** Muestra un libro concreto de la sección.

- **Página Add Book (Añadir libro):** Permite añadir un libro nuevo a los usuarios que tengan los permisos correspondientes.

- **Página Edit Book (Editar libro):** Permite editar un libro a los usuarios que tengan los permisos correspondientes.

El módulo **Books** acepta todas las funciones estándar del marco de trabajo para el **kit de inicio Community**, como:

- ***Seguridad:*** Es posible hacer que sólo ciertos miembros de la comunidad tengan autorización para ver, añadir, editar o borrar un libro.

- ***Moderación:*** Cuando se modera una sección de libros, sólo aparecen los libros que hayan sido aprobados.

- ***Comentarios:*** Los usuarios pueden añadir comentarios a los libros.

- ***Evaluaciones:*** Los usuarios pueden evaluar un libro como bueno o malo.

- ***Notificaciones:*** Se pueden enviar a los usuarios notificaciones automáticas por correo electrónico cuando aparece en la sección un libro nuevo.

4.5.1.3 Módulo Events

El módulo **Events** se puede utilizar para dar una lista de los eventos que se vayan a celebrar en la comunidad. Este módulo permite programar eventos de forma que no aparezcan hasta una fecha determinada. El módulo **Events** también permite asociar una imagen con un evento.

El módulo **Events** consta de cuatro páginas principales:

- **Página Events Section (Eventos en sección):** Muestra una lista de eventos en la sección.

- **Página Event (Evento):** Muestra un evento concreto de la sección.

- **Página Add Event (Añadir evento):** Permite añadir un evento nuevo a los usuarios que tengan los permisos correspondientes.

- **Página Edit Event (Editar evento):** Permite editar un evento a los usuarios que tengan los permisos correspondientes.

El módulo **Events** acepta todas las funciones estándar del marco de trabajo para el **kit de inicio Community**, como:

- ***Seguridad:*** Es posible hacer que sólo ciertos miembros de la comunidad tengan autorización para ver, añadir, editar o borrar un evento.

- ***Moderación:*** Cuando se modera una sección de eventos, sólo aparecen los eventos que hayan sido aprobados.

- ***Comentarios:*** Los usuarios pueden añadir comentarios a los eventos.

- ***Evaluaciones:*** Los usuarios pueden evaluar un evento como bueno o malo.

- ***Notificaciones:*** Se pueden enviar a los usuarios notificaciones automáticas por correo electrónico cuando aparece en la sección un evento nuevo.

4.5.1.4 Módulo Downloads

El módulo **Downloads** se puede utilizar para añadir a la comunidad archivos descargables. Los archivos pueden tener cualquier tipo de contenido. Por ejemplo, es posible añadir documentos de ***Microsoft*** **Word**, muestras de código, archivos de sonido o programas.

El módulo **Downloads** utiliza cuotas para los usuarios y la comunidad. Cada usuario (y la comunidad en su conjunto) tiene asignada una cuota determinada que le impide cargar demasiados datos en la base de datos.

Una vez superada la cuota, el usuario no puede cargar más contenidos. La cuota asignada a un usuario concreto se puede cambiar en la página de administración **Edit User *(Editar usuario)***.

El módulo **Downloads** consta de cuatro páginas principales:

- **Página Downloads Section (Descargas en sección):** Muestra una lista de archivos descargables en la sección.

- **Página Downloads (Descarga):** Muestra un archivo descargable concreto de la sección.

- **Página Add Download (Añadir descarga):** Permite añadir un archivo descargable nuevo a los usuarios que tengan los permisos correspondientes.

- **Página Edit Download (Editar descarga):** Permite editar un archivo descargable a los usuarios que tengan los permisos correspondientes (para sustituirlo, por ejemplo).

El módulo **Downloads** acepta todas las funciones estándar del marco de trabajo para el **Kit de inicio Community**, como:

- ***Seguridad:*** Es posible hacer que sólo ciertos miembros de la comunidad tengan autorización para ver, añadir, editar o borrar un archivo descargable.

- ***Moderación:*** Cuando se modera una sección de descargas, sólo aparecen las descargas que hayan sido aprobadas.

- ***Comentarios:*** Los usuarios pueden añadir comentarios a las descargas.

- ***Evaluaciones:*** Los usuarios pueden evaluar una descarga como buena o mala.

- ***Notificaciones:*** Se pueden enviar a los usuarios notificaciones automáticas por correo electrónico cuando aparece en la sección un archivo descargable nuevo.

4.5.1.5 Módulo Photo Gallery

El módulo **Photo Gallery** se puede utilizar para mostrar una galería de imágenes en la comunidad. Admite imágenes **GIF** y **JPEG** *(por el momento no es compatible con PNG)*.

Al añadir una imagen a una sección de la galería de fotos, se genera y se guarda automáticamente una vista en miniatura de la imagen. La página **Photo Gallery Section** muestra vistas comprimidas en miniatura de las imágenes para que sea posible ver varias al mismo tiempo.

Las imágenes que haya en la galería de fotos se pueden ver en dos modos: ***tira de imágenes y vistas en miniatura***. El modo se selecciona en una lista desplegable. Si se elige **tira de imágenes**, se puede hacer clic sobre una imagen para verla a tamaño completo en la misma página. Por el contrario, si las imágenes aparecen como **vistas en miniatura**, al hacer clic sobre una imagen se abre otra página donde aparece a tamaño completo.

El módulo **Photo Gallery** utiliza cuotas para los usuarios y la comunidad. Si un usuario intenta cargar una imagen nueva y se ha sobrepasado ya la cuota del usuario o la de la comunidad, la imagen no se cargará y aparecerá un mensaje de error.

El módulo **Photo Gallery** consta de cuatro páginas principales:

- **Página Photo Gallery Section (Galería de fotos en sección):** Muestra una lista de imágenes en la sección.

- **Página Photo (Foto):** Muestra una foto concreta de la galería de fotos.

- **Página Add Photo (Añadir foto):** Permite añadir una imagen nueva a la galería de fotos a los usuarios que tengan los permisos correspondientes.

- **Página Edit Photo (Editar foto):** Permite editar una foto a los usuarios que tengan los permisos correspondientes (para sustituirla, por ejemplo).

El módulo **Photo Gallery** acepta todas las funciones estándar del marco de trabajo para el **Kit de inicio Community**, como:

- ***Seguridad:*** Es posible hacer que sólo ciertos miembros de la comunidad tengan autorización para ver, añadir, editar o borrar una foto. Por ejemplo, se puede hacer que sólo los usuarios registrados puedan ver las imágenes de la galería de fotos.

- ***Moderación:*** Cuando se modera una sección de galería de fotos, sólo aparecen las fotos que hayan sido aprobadas.

- ***Comentarios:*** Los usuarios pueden añadir comentarios a las fotos.

- ***Evaluaciones:*** Los usuarios pueden evaluar una foto como buena o mala.

- ***Notificaciones:*** Se pueden enviar a los usuarios notificaciones automáticas por correo electrónico cuando aparece en una sección una foto nueva.

4.5.1.6 Módulo Parent Section

El módulo **Parent Section** se puede utilizar para añadir secciones secundarias a una sección. Por ejemplo, se podría crear una sección primaria llamada **Debate** y añadirle dos secciones secundarias de artículos llamadas **Libros Favoritos** y **Películas Favoritas**. Al abrir la página **Debate** aparecerían descripciones y enlaces a las dos secciones secundarias.

Si la ***página de inicio de la comunidad*** es una sección primaria, aparecerán en ella todas las secciones de nivel superior. Para ello, hay que seleccionar **Parent Section**

en el panel **Home Page Content Area *(Área de contenidos de la página de inicio)*** de la página de administración **Edit Site *(Editar sitio)***.

4.5.1.7 Módulo HTML Page

El módulo **HTML Page** permite crear una sección que contenga una sola **página HTML**. Esto es útil, por ejemplo, si se quiere que en la comunidad haya una sola página de preguntas **FAQ** o **AYUDA**.

La página puede tener cualquier clase de **contenido HTML**, incluyendo imágenes. Para añadir las imágenes que vayan a aparecer en la **página HTML** se utiliza la página de administración **Edit Images *(Editar imágenes)***.

4.5.1.8 Módulo Static Page

Sólo se pueden crear ***páginas estáticas*** si se tiene acceso a la carpeta **Static Pages** del Servidor Web que aloja el sitio del **Kit de inicio Community**.

Las ***páginas estáticas*** permiten personalizar los contenidos que aparecen en una sección. Una ***página estática*** es un control de usuario **ASP .NET *(un archivo .ascx)*** que se ha añadido a la carpeta **Static Pages** de la comunidad.

Una ***página estática*** puede tener cualquier clase de contenido, incluyendo **código ASP .NET**, controles y **HTML** estático, y aparecerá en el área de contenidos de la página. La página estática debe incluir su ruta y su nombre. La convención de nomenclatura es: ***StaticPages**ID* de la comunidad\Nombre de la sección.ascx**.

Puesto que el **Kit de inicio Community** permite que haya varias comunidades, las ***páginas estáticas*** de cada comunidad deben estar separadas en distintos

subdirectorios cuyo nombre será la **ID** de la comunidad. Para conocer la **ID** de una comunidad se emplea la página de administración **ISP**.

El nombre de la ***página estática*** debe ser el mismo que el de la sección. Por ejemplo, supongamos que la **ID** de la comunidad es 1 y se crea una sección **Static Page** llamada *Mi Sección*. En ese caso, la **página estática** personalizada se debe guardar con el siguiente nombre y en la siguiente carpeta: ***\StaticPages\1\Mi Sección.ascx***.

4.6 Kit de inicio Time Tracker

La aplicación **Kit de inicio Time Tracker de ASP .NET** *es una aplicación empresarial que permite a los usuarios controlar el tiempo empleado en los proyectos*. Además, la aplicación permite a los administradores de proyectos controlar mejor el estado de los proyectos que gestionan, ya que muestra las anotaciones de horas de cada usuario y genera informes detallados sobre dichas anotaciones.

La aplicación **Time Tracker** expone varias características fundamentales de **ASP .NET** y **.NET Framework**. Al igual que las aplicaciones **Kit de inicio Portal de ASP .NET** y **Kit de inicio Commerce de ASP .NET**, el principal objetivo de la aplicación **Kit de inicio Time Tracker** *es mostrar los procedimientos más adecuados en materia de desarrollo y arquitectura a los desarrolladores que utilizan **ASP .NET**.*

El **Kit de inicio Time Tracker** cuenta con las siguientes características principales:

- *La aplicación **Kit de inicio Time Tracker** de **ASP .NET** se ha globalizado (es decir, se ha diseñado y desarrollado de manera que funcione para distintas referencias culturales), lo que permitirá su localización en el futuro*.

- *Arquitectura de tres niveles lógicos*.
- *Seguridad mediante autenticación basada en **Windows***.
- *Recuperación de información de cuenta de usuario desde **Active Directory** o **NT SAM***.
- *Cuadrícula de datos con edición en línea personalizada*.
- *Creación dinámica de gráficos e imágenes con **GDI+***.
- ***Microsoft Data Access Application Blocks** para **.NET***.
- *Formularios Mobile Web Forms que utilizan **Microsoft Mobile Internet Toolkit (MMIT)***.
- *Creación dinámica de informes*.
- *Utilización de clases de datos sencillas para la interfaz entre la capa de lógica empresarial y la capa de interfaz de usuario*.
- *Seguridad basada en funciones mediante el uso de un elemento principal personalizado para la autorización de funciones*.

4.6.1 Objetivos del Kit de incio Time Tracker

Debido a que **Time Tracker** es una aplicación empresarial de ejemplo que se usará normalmente en una organización, el desarrollador tendrá que enfrentarse a diferentes retos:

1. Énfasis en el mantenimiento y no en el rendimiento general.

La aplicación debe ser fácil de administrar. El proceso de *agregar nuevos usuarios* a la aplicación debe desarrollarse sin problemas. La aplicación **Time Tracker** *utiliza la autenticación basada en* ***Windows*** y agrega un usuario a su *Base de Datos* la primera vez que éste visita el sitio Web.

Además, *cualquier miembro de la función de administración* puede agregar también un usuario utilizando la página de detalles de usuarios. Una vez que el usuario pertenece al dominio de la empresa, tiene autorización para usar la aplicación **Time Tracker**. El acceso de usuarios anónimos al sitio Web no está permitido.

2. Utilización de la información existente.

Los nombres y apellidos de los usuarios se pueden recuperar desde Active Directory o NT SAM de la empresa. *Si no se pueden recuperar el nombre y los apellidos de un usuario*, la aplicación se "degrada" gentilmente mostrando el dominio \ nombre de usuario. Se proporciona una implementación para Active Directory y NT SAM en el caso de que no sea posible el acceso a una instalación de Active Directory.

3. Separación clara entre niveles lógicos.

En una **Intranet** es frecuente que ciertas capas o niveles de una aplicación se *compartan o reutilicen* entre aplicaciones de la empresa. En el caso de la aplicación **Time Tracker** esto se consigue utilizando objetos sencillos como interfaz entre la capa de lógica empresarial y la capa de interfaz de usuario.

El **Kit de inicio Time Tracker de ASP .NET** se ha desarrollado utilizando los modelos de codificación con versiones en línea y de código subyacente. La versión para **SDK** se ha escrito usando el modelo de codificación en línea y se ha optimizado para **ASP .NET Web Matrix Project** y **.NET Framework SDK**. La segunda versión

se ha escrito usando ***Microsoft*** **Visual Studio .NET** con el modelo de codificación de código subyacente. Para ambas versiones se han creado implementaciones escritas en ***C#*** y ***VB .NET***.

4.6.2 Arquitectura del Kit de inicio Time Tracker

La aplicación **Kit de inicio Time Tracker de ASP .NET** es una **aplicación Web ASP .NET** que utiliza una arquitectura de tres niveles lógicos como se muestra en la **Figura 4.19**.

Figura 4.19. Arquitectura del Kit de inicio Time Tracker.

La ***lógica de acceso a datos***, ***la lógica empresarial*** y ***la lógica de interfaz de usuario*** están separadas en clases diferentes. Las arquitecturas de "n" niveles presentan numerosas ventajas, entre las que se incluyen:

- Hay una separación clara entre las ***capas de lógica de acceso a datos***, ***de interfaz de usuario y empresarial***. Este aislamiento fomenta la reutilización de código y facilita las tareas de mantenimiento y de mejora del código.

- Las reglas de empresa están centralizadas en un ***componente que se puede reutilizar fácilmente*** y se proporciona un lenguaje de alto nivel (como ***C#*** o ***Visual Basic .NET***) para desarrollar dichas reglas.

- El ***código de acceso a datos está centralizado en un solo sitio***, con lo que se facilitan el desarrollo y el mantenimiento.

4.6.3 Base de Datos del Kit de inicio Time Tracker

La aplicación **Kit de inicio Time Tracker de ASP .NET** utiliza una *Base de Datos* de ***Microsoft* SQL Server 2000 *(también se admite MSDE 2000)***. El esquema físico de la *Base de Datos* se ha creado tras un profundo análisis de los casos de uso y los requisitos de **Time Tracker**.

Algunos de estos requisitos son los siguientes:

- *Un usuario anota las horas correspondientes a un proyecto y una categoría.*

- *Un usuario tiene una función que especifica las funciones que está autorizado a utilizar en la aplicación.*

- *Sólo los usuarios que son miembros de un proyecto pueden registrar horas correspondientes a ese proyecto.*

- *Un proyecto tiene una o más categorías.*

- *Cada proyecto puede tener asignado un usuario como administrador del proyecto.*

En base a lo anterior el esquema de la base de datos es el que se muestra en la **Figura 4.20**.

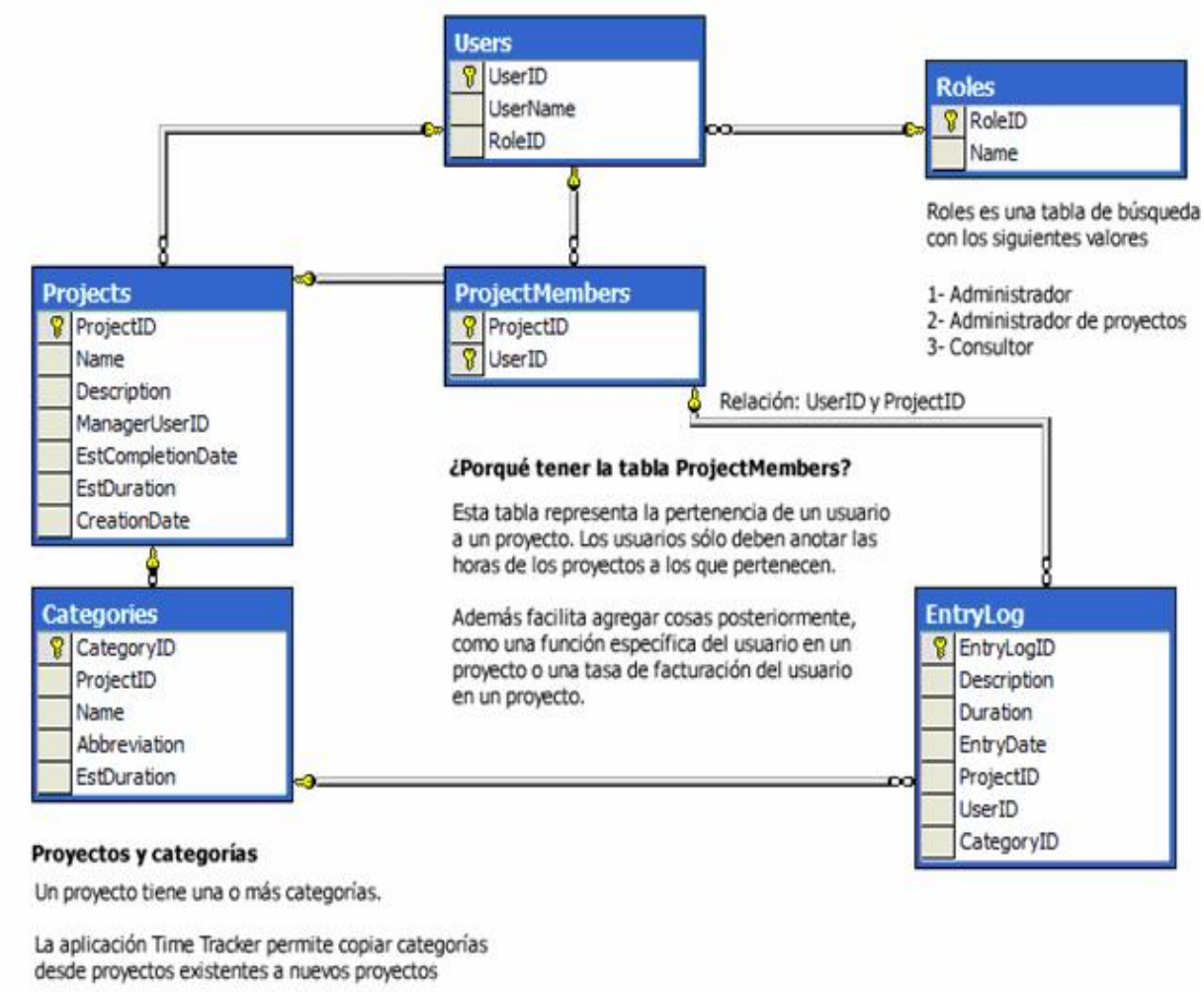


Figura 4.20. Esquema de la base de datos.

4.6.3.1 Procedimientos almacenados

La aplicación **Time Tracker** emplea ***procedimientos almacenados*** para encapsular todas las consultas de la *Base de Datos*. El uso de ***procedimientos almacenados*** permite separar claramente la *Base de Datos* de la ***capa de acceso a datos***.

Los ***procedimientos almacenados*** proporcionan ventajas en cuanto al ***rendimiento***, ya que se *optimizan cuando se ejecutan por primera vez y se mantienen en memoria para llamadas posteriores*. Los parámetros con establecimiento inflexible de tipos en ***procedimientos almacenados*** y la posibilidad de establecer permisos en cada ***procedimiento almacenado*** hacen que mejore la seguridad. Los ***procedimientos almacenados*** que se muestran en la siguiente tabla, representan un subconjunto de los que se necesitan para implementar la aplicación **Time Tracker**.

Nombre	Parámetros	Descripción	Resultado
GetTimeEntry	@EntryLogID	*Recupera detalles de las anotaciones de horas de la tabla EntryLog*	Conjunto de resultados SQL consistente en una fila y todas las columnas de la tabla EntryLog.
AddTimeEntry	@UserName @ProjectID @CategoryID @EntryDate @Description @Duration	*Agrega una anotación de horas a la tabla EntryLog*	Nuevo objeto EntryLogID si se realiza correctamente.
UpdateTimeEntry	@EntryLogID @UserName @ProjectID @CategoryID @EntryDate @Description @Duration	*Actualiza datos de anotaciones de horas correspondientes al registro de anotaciones de horas especificado.*	
ListTimeEntries	@UserQueryID @UserID @StartDate @EndDate	*Recupera anotaciones de horas de la tabla EntryLog correspondientes al usuario especificado y al intervalo de fechas especificado.*	Conjunto de resultados SQL de anotaciones de horas consistente en todas las columnas de la tabla EntryLog.
DeleteTimeEntry	@EntryLogID	Elimina el registro especificado de la tabla EntryLog.	

Uno de los ***procedimientos almacenados*** más complejos de la aplicación **Time Tracker** es el denominado ***"ListTimeEntries".*** Este procedimiento tiene la misión de *recuperar anotaciones de horas* de la tabla ***EntryLog*** correspondientes a un usuario especificado basándose en su función. Por ejemplo, los usuarios siempre pueden ver sus propias anotaciones de horas.

Los ***administradores*** pueden ver todas las anotaciones de horas correspondientes a cualquier usuario, y los ***administradores de proyectos*** pueden ver las correspondientes a miembros de los proyectos que gestionan. Sin embargo, los ***usuarios con la función de consultor*** *sólo pueden ver sus anotaciones de horas*. A continuación se muestra el código de dicho procedimiento:

```
CREATE
     PROCEDURE ListTimeEntries
 (
    @QueryUserID int,
    @UserID int,
    @StartDate datetime,
    @EndDate datetime
 )
      AS
      DECLARE @QueryUserRoleID int
      SELECT @QueryUserRoleID = Users.RoleID
      FROM Users
      WHERE Users.UserID = @QueryUserID
      IF @QueryUserRoleID = 2
      BEGIN
       SELECT
       EntryLogID, EntryLog.Description, Duration,   EntryDate,
       EntryLog.ProjectID AS ProjectID,
       EntryLog.CategoryID AS CategoryID,
  Categories.Abbreviation AS CategoryName,
       Projects.Name AS ProjectName,
       ManagerUserID, Categories.Abbreviation AS CatShortName
       FROM
       EntryLog
       INNER JOIN
       Categories
```

```
 ON
 EntryLog.CategoryID = Categories.CategoryID
 INNER JOIN
 Projects
 ON
 EntryLog.ProjectID = Projects.ProjectID
 WHERE
 UserID = @UserID
AND
EntryDate >= @StartDate
AND
EntryDate <= @EndDate
AND
ManagerUserID = @QueryUserID
END
ELSE IF @QueryUserRoleID = 1 or @QueryUserID = @UserID
BEGIN
 SELECT
 EntryLogID, EntryLog.Description, Duration, EntryDate,
 EntryLog.ProjectID AS ProjectID,
 EntryLog.CategoryID AS CategoryID,
 Categories.Abbreviation AS CategoryName,
 Projects.Name AS ProjectName,
 ManagerUserID, Categories.Abbreviation AS CatShortName
 FROM
 EntryLog
 INNER JOIN
 Categories
ON
EntryLog.CategoryID = Categories.CategoryID
INNER JOIN
Projects
```

```
    ON
    EntryLog.ProjectID = Projects.ProjectID
    WHERE
    UserID = @UserID
    AND
    EntryDate >= @StartDate
    AND
    EntryDate <= @EndDate
    END
    GO
```

4.6.3.2 Capa de acceso a datos

La ***capa de acceso a datos*** *se utiliza para encapsular código específico de la Base de Datos*, separando los detalles de la *Base de Datos* de la ***capa de lógica empresarial***. Esta separación permite la agregación de *Bases de Datos* diferentes sin necesidad de cambiar las ***capas de lógica empresarial y de presentación***.
Para implementar la ***capa de acceso a datos*** se utiliza ***Microsoft*** **Data Access Application Blocks (DAAB)**.

DAAB es un **componente .NET** que contiene código optimizado de acceso a datos para enviar comandos a una *Base de Datos* de **SQL Server**. El uso de **DAAB** reduce la cantidad de código personalizado que se necesita para crear, probar y mantener el acceso a la *Base de Datos*.

Las llamadas de acceso a la *Base de Datos* pueden quedar reducidas a una sola línea de código, en lugar de las seis o más líneas que suelen ser necesarias.

```
DataSet ds = SqlHelper.ExecuteDataset(

ConfigurationSettings.AppSettings[Web.Global.CfgKeyConnString],
```

```
                                CommandType.StoredProcedure,
"ListAllProjects");
```

En el código anterior se llama al procedimiento almacenado ***"ListAllProjects"*** y el conjunto de resultados se asigna a un objeto **DataSet**. Esta llamada requiere solamente una línea de código. Sin embargo, serían necesarias seis líneas de código si se utilizara directamente **ADO .NET *(System.Data)*** en lugar de **DAAB**.

```
SqlConnection myConnection = new SqlConnection(

ConfigurationSettings.AppSettings[Web.Global.CfgKeyConnString])
;
   SqlCommand myCommand = new SqlCommand("ListAllProjects",
         myConnection);
        myCommand.CommandType = CommandType.StoredProcedure;
     SqlDataAdapter myDataAdapter = new
SqlDataAdapter(myCommand);
  DataSet myDataSet = new DataSet();
            myDataAdapter.Fill(myDataSet);
```

4.6.4 Capa lógica Empresarial del Kit de incio Time Tracker

La ***capa de lógica empresarial*** *separa el código específico para la aplicación, o la forma en que una compañía desempeña sus actividades, del código específico para la Base de Datos y la interfaz de usuario*. Otras aplicaciones empresariales creadas por una compañía pueden utilizar la ***capa de lógica empresarial*** si es necesario, con lo que se maximiza la reutilización del código. La separación también ofrece la posibilidad de crear servicios Web que utilicen la funcionalidad proporcionada por la ***capa de lógica empresarial***.

4.6.5 Capa de presentación del Kit de incio Time Tracker

La ***capa de presentación*** *se encarga de la interfaz de usuario y se comunica directamente con la capa de lógica empresarial*. La separación de la ***capa de presentación*** del resto de la aplicación permite desarrollar diferentes interfaces de usuario (por ejemplo, formularios Web Forms, formularios Windows Forms, dispositivos móviles) que utilizan la misma lógica empresarial y el mismo código de acceso a la *Base de Datos*.

4.6.5.1 Controles de usuario

Las ***fichas principales*** que aparecen en la parte superior de cada pantalla (**Figura 4.21**) , así como las ***subfichas de Administración***, se implementan como controles de usuario para aprovechar las ventajas de la reutilización de código. El ***control de usuario*** aplica también la *autorización*, ya que decide si se deben mostrar las fichas de informes y de administración basándose en la función del usuario. La función del usuario se obtiene de una cookie cifrada que contiene información sobre el usuario.

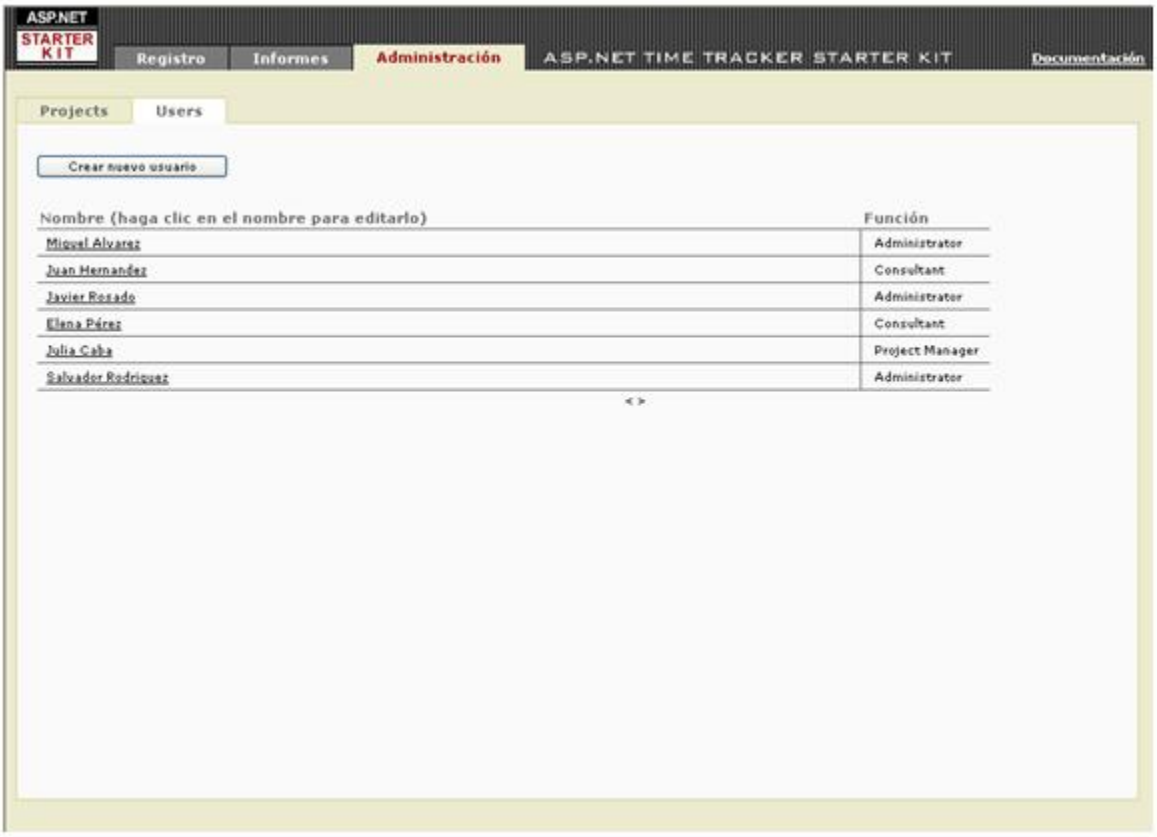


Figura 4.21. Entorno de Controles de usuario del Kit de inicio Time Tracker.

4.6.5.2 Página de Anotaciones de horas

En esta página de uso simple, *el usuario puede realizar anotaciones de horas*, es decir, ***agregar, modificar o quitar anotaciones de horas***. En la **Figura 4.22**, se muestra el entorno de esta página.

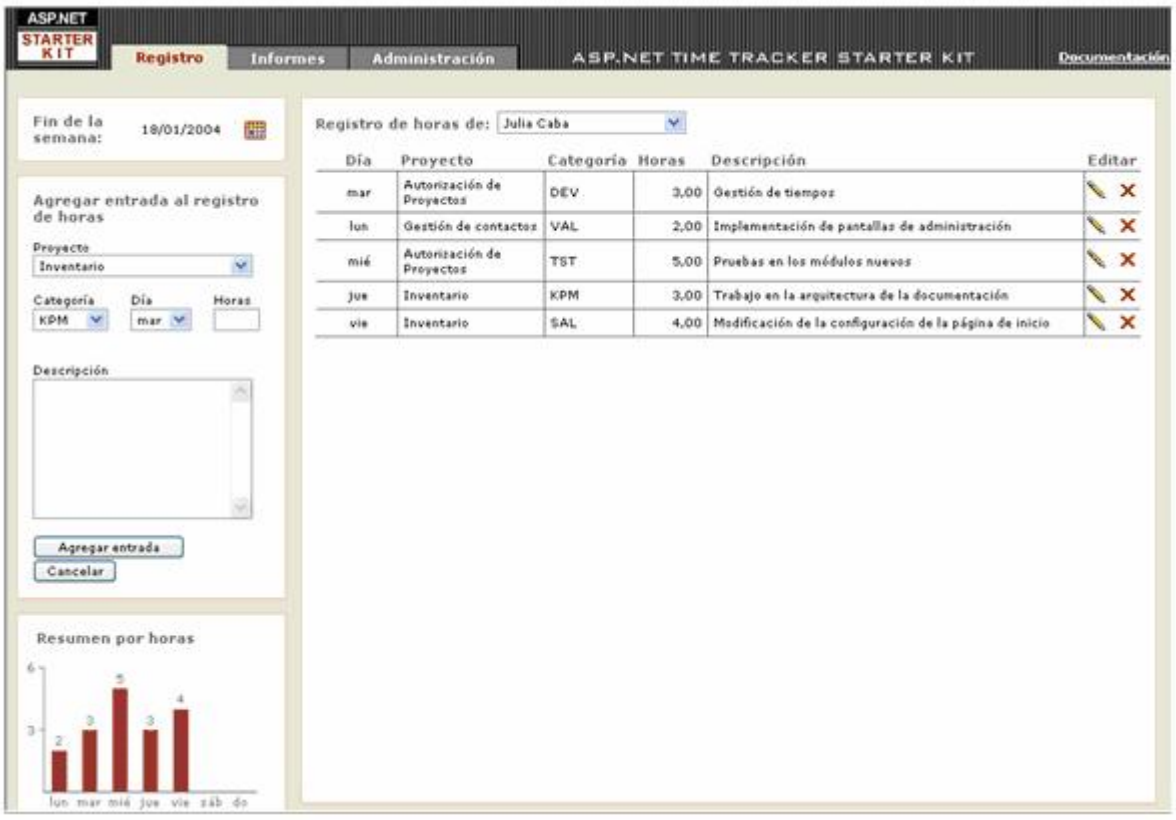


Figura 4.22. Entorno de Página de anotaciones de horas.

4.6.5.3 DataGrid con edición personalizada en línea

Una de las funciones más relevantes del control **DataGrid** reside en la posibilidad de *realizar la edición en línea*. Para ello se define una columna como ***EditTemplate*** y se rellena con otro control de servidor de **ASP .NET**, como ***TextBox*** o ***DropDownList***. La cuadrícula de registro de horas que se muestra en la **Figura 4.23**.

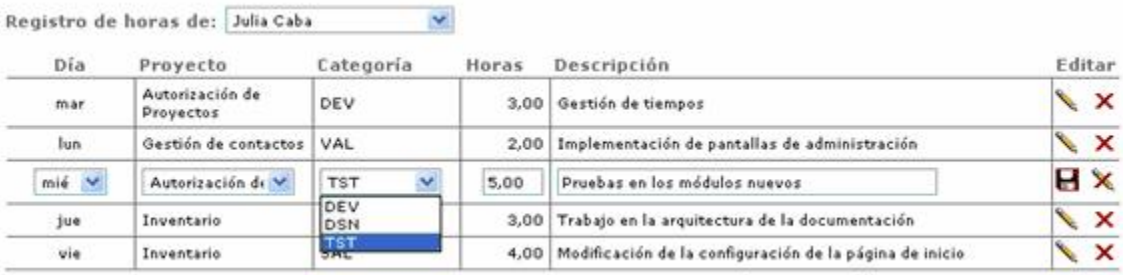


Figura 4.23. Cuadricula de registro de horas.

La aplicación **Time Tracker** amplía las funciones integradas de edición en línea del control **DataGrid**, por lo que un cambio efectuado en el control **DropDownList** de **Project** modificará la lista de categorías en el correspondiente control **DropDownList** de **Category**.

4.6.6 Creación de Informes

El **Kit de inicio Time Tracker** otorga la posibilidad de generar dos clases de informes: ***Informes de Proyectos*** e ***Informes de Recursos***.

Informes de Proyectos

Cuando los ***directores de proyectos*** necesitan *conocer el progreso de sus equipos en distintos proyectos*, pueden utilizar la página de informes para iniciar el ***informe de proyectos***. Este informe se crea basándose en los proyectos seleccionados en la pantalla de inicio, y sus datos se *agrupan por resumen de proyectos, resumen de categorías y resumen de recursos*. Cada nombre de consultor incluido en el resumen de recursos es un hipervínculo que lleva al informe de recursos correspondiente a ese consultor. En la **Figura 4.24**, se muestra el informe de proyecto que se genera por la aplicación.

Figura 4.24. Reporte de Proyecto.

Para mostrar el *resumen de los proyectos* se utiliza el control Web **DataList**. En este informe se usan tres controles **DataList**: ***ProjectList(cuadro exterior)***, ***CategoryList (cuadro intermedio)*** *y* ***EntryList (cuadro interior)***.

Los controles **DataList** se anidan uno dentro de otro mediante la colocación de un control **DataList** en la plantilla de elemento del control **DataList** primario, como se muestra a continuación:

```
<asp:datalist id="ProjectList" RepeatColumns="1"
runat="server">
      <headerstyle cssclass="header-gray" />
      <headertemplate>
      Project Report
      </headertemplate>
      <itemtemplate>
     <asp:datalist id="CategoryList" RepeatColumns="1"
runat="server">
                  <itemtemplate>
                        ...
```

Cada proyecto debe estar relacionado con todas sus categorías, y cada una de estas debe estar relacionada a su vez con todos sus recursos. Para implementar estas relaciones, los orígenes de datos correspondientes a **CategoryList** y **EntryList** se deben asignar dinámicamente en tiempo de ejecución.

```
<asp:datalist id="CategoryList" DataSource='<%#
ListCategory((int)
DataBinder.Eval(Container.DataItem, "ProjectID")) %>'
runat="server">
```

ListCategory *es una función auxiliar* que permite ***recuperar todos los resúmenes de categorías correspondientes a un proyecto concreto***. Obsérvese que el objeto **ProjectID** se pasa de **ProjectList** a **CategoryList**.

El origen de datos correspondiente a **EntryList** también se implementa utilizando la misma técnica.

```
<asp:datalist id="EntryList" DataSource='<%#
ListTimeEntries((int)
DataBinder.Eval(Container.DataItem, "CategoryID")) %>'
runat="server">
```

Informe de recursos

El ***informe de recursos*** *agrupa resúmenes de horas de uno o más consultores y presenta una lista de detalles de anotaciones de horas correspondientes a un intervalo de fechas concreto*. Este informe permite a un ***director de proyectos*** *ver rápidamente las anotaciones de horas* de cualquier consultor o de todos los consultores a los que dirige. En la **Figura 4.25**, se muestra el informe de recursos generado por la aplicación.

Informe de recursos

Fecha de inicio	Fecha de finalización
12/01/2004	18/01/2004

Consultor	Horas totales
Julia Caba	17,00

Fecha	Proyecto	Cat.	Hs	Descripción
13/01/2004	Autorización de Proyectos	DEV	3,00	Gestión de tiempos
12/01/2004	Gestión de contactos	VAL	2,00	Implementación de pantallas de administración
14/01/2004	Autorización de Proyectos	TST	5,00	Pruebas en los módulos nuevos
15/01/2004	Inventario	KPM	3,00	Trabajo en la arquitectura de la documentación
16/01/2004	Inventario	SAL	4,00	Modificación de la configuración de la página de inicio

Figura 4.25. Informe de Recursos.

En este caso se aplica la misma técnica que la utilizada para crear el informe de proyectos, aunque la estructura es algo diferente. El informe de recursos tiene un control **DataList** ***(UserList, cuadro exterior)*** con un control **DataGrid** anidado ***(TimeEntryGrid, cuadro interior)***.

4.6.7 Creación de gráficos con GDI+

En la página de anotación de horas se crea dinámicamente un ***gráfico de barras*** a partir de las anotaciones de horas del usuario actual. *Este gráfico proporciona una representación visual de las horas trabajadas por un usuario en una semana concreta* como se muestra en la **Figura 4.26**.

Figura 4.26. Gráfico de horas de usuario.

4.6.8 Elementos móviles

La aplicación **Time Tracker** tiene un grado de compatibilidad limitado con dispositivos móviles, como **Pocket PC** y teléfonos celulares. Los usuarios pueden *ver, agregar y actualizar anotaciones de horas* utilizando sus ***dispositivos móviles*** (**Figura 4.27**). Para implementar la compatibilidad con este tipo de dispositivos se utiliza ***Microsoft*** **Mobile Information Toolkit**.

Figura 4.27. Entorno móvil.

Las páginas normales de **formularios Web Forms ASP .NET** sólo pueden contener un formulario de servidor ***(por ejemplo, <form runat="Server">)*** por página. Sin embargo, los dispositivos móviles suelen tener pantallas más pequeñas, por lo que una página de formularios **Mobile Web Forms** permite definir varios formularios en una sola página. Esto proporciona varias ventajas:

1. Permite *reducir la complejidad de un sitio Web*, evitando un gran número de páginas pequeñas.

2. Permite *organizar una **aplicación Web móvil** de la misma manera que una aplicación de escritorio*. Una ***página móvil*** puede tener varios formularios, del mismo modo que un formulario de escritorio puede tener diversos formularios que se correspondan con él.

3. Permite *aprovechar las ventajas de las funciones de página que proporcionan los **formularios Mobile Web Forms***, como el mantenimiento automático del estado de la página (conocido como

Viewstate) o la adaptación a dispositivos que pueden recibir varias pantallas en una sola respuesta.

4.7 Kit de incio Reports

El **Kit de inicio Reports de ASP .NET** *es una guía para la creación y publicación de informes Web que se pueden personalizar utilizando* ***ASP .NET***. Esta guía explica a los desarrolladores la forma de implementar informes con funcionalidad avanzada sin tener que depender de herramientas de terceros para la creación de informes.

El **Kit de inicio Reports** utiliza ocho muestras para presentar informes funcionales en línea y sus versiones equivalentes optimizadas para la impresión. Los desarrolladores podrán usar estos informes de muestra para crear sus propios informes utilizando estas técnicas y **ASP .NET**.

Las características principales del **Kit de inicio Reports** son:

- *Anidamiento de controles de servidor.*
- *Visualización de datos relacionales.*
- *Totales acumulados.*
- *Paginación y ordenación de cuadrículas de datos.*
- *Generación dinámica de imágenes con* ***GDI+****.*
- *Filtrado.*
- *Obtención de detalles.*

El **Kit de inicio Reports** se ha desarrollado utilizando los *modelos de codificación con versiones en línea y de código subyacente*. La versión para **SDK** se ha escrito usando el modelo de codificación en línea y se ha optimizado para **ASP .NET Web Matrix Project** y **.NET Framework SDK**.

La segunda versión se ha escrito usando ***Microsoft* Visual Studio .NET** con el *modelo de codificación de código subyacente*. Para ambas versiones se han creado implementaciones escritas en ***C#*** y ***VB .NET***.

4.7.1 Arquitectura de la Aplicación

La aplicación **Kit de inicio Reports de ASP .NET** es una **aplicación Web ASP .NET** que utiliza una arquitectura de tres niveles lógicos, los cuales se muestran en la **Figura 4.28**.

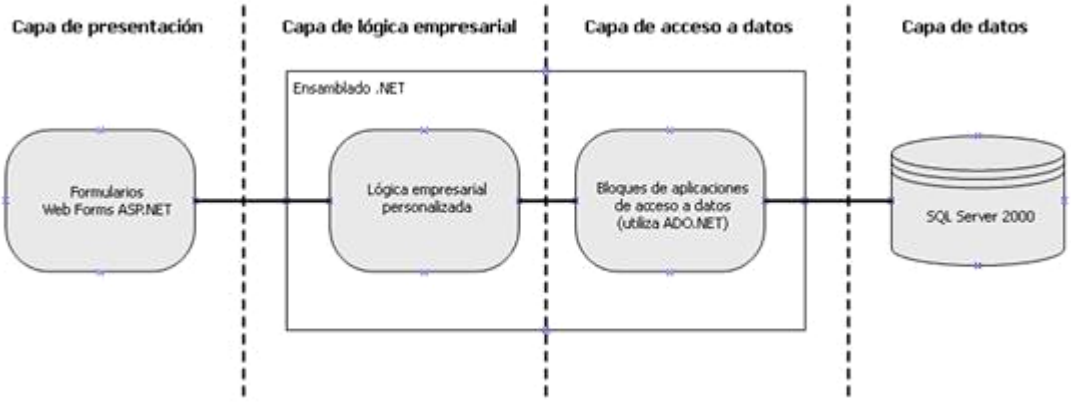


Figura 4.28. Arquitectura de la aplicación.

La ***lógica de acceso a datos, la lógica empresarial y la lógica de interfaz de usuario*** están separadas en clases diferentes. Las arquitecturas de "n" niveles presentan numerosas ventajas, entre las que se incluyen:

1. Hay una separación clara entre las ***capas de lógica de acceso a datos***, ***de interfaz de usuario y empresarial***. Este aislamiento fomenta la reutilización de código y facilita las tareas de mantenimiento y de mejora del código.

2. Las reglas de empresa están centralizadas en un componente que se puede reutilizar fácilmente y se proporciona un lenguaje de alto nivel (como ***C#*** o ***Visual Basic .NET***) para desarrollar dichas reglas.

3. El código de acceso a datos está centralizado en un solo sitio, con lo que se facilitan el desarrollo y el mantenimiento.

4.7.2 Flujo de la aplicación

La **página principal de Reports** actúa como página de selección de los diferentes informes. El usuario selecciona en esta página el informe que desea ver. De este modo obtiene acceso a la página de detalles del informe que contiene *información general y vínculos a información adicional sobre los informes*, como ***versiones activas, origen de la página, código de la lógica empresarial y procedimientos almacenados***. La **Figura 4.29**, muestra el flujo de la aplicación.

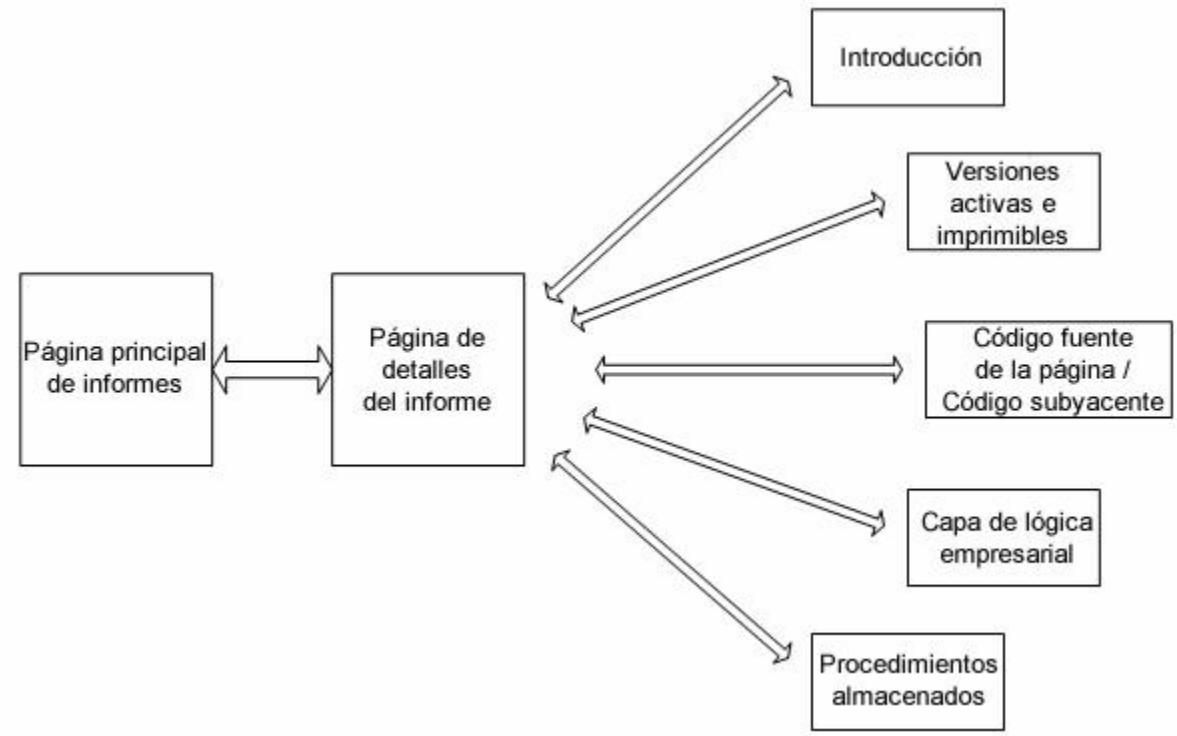


Figura 4.29. Flujo de la aplicación.

4.7.3 Página de inicio

La **página principal de Reports** *describe los ocho informes básicos con su representación en iconos y proporciona una lista de las funciones que se pueden encontrar en cada informe*. Esto permite a los desarrolladores buscar ejemplos con un diseño específico o que les ayuden a resolver el problema técnico al que se enfrentan.

El usuario puede seleccionar un informe haciendo *clic* sobre su nombre o su icono, con lo que obtiene acceso a la página de detalles del informe. La **Figura 4.30**, muestra la **página de inicio de Reports**.

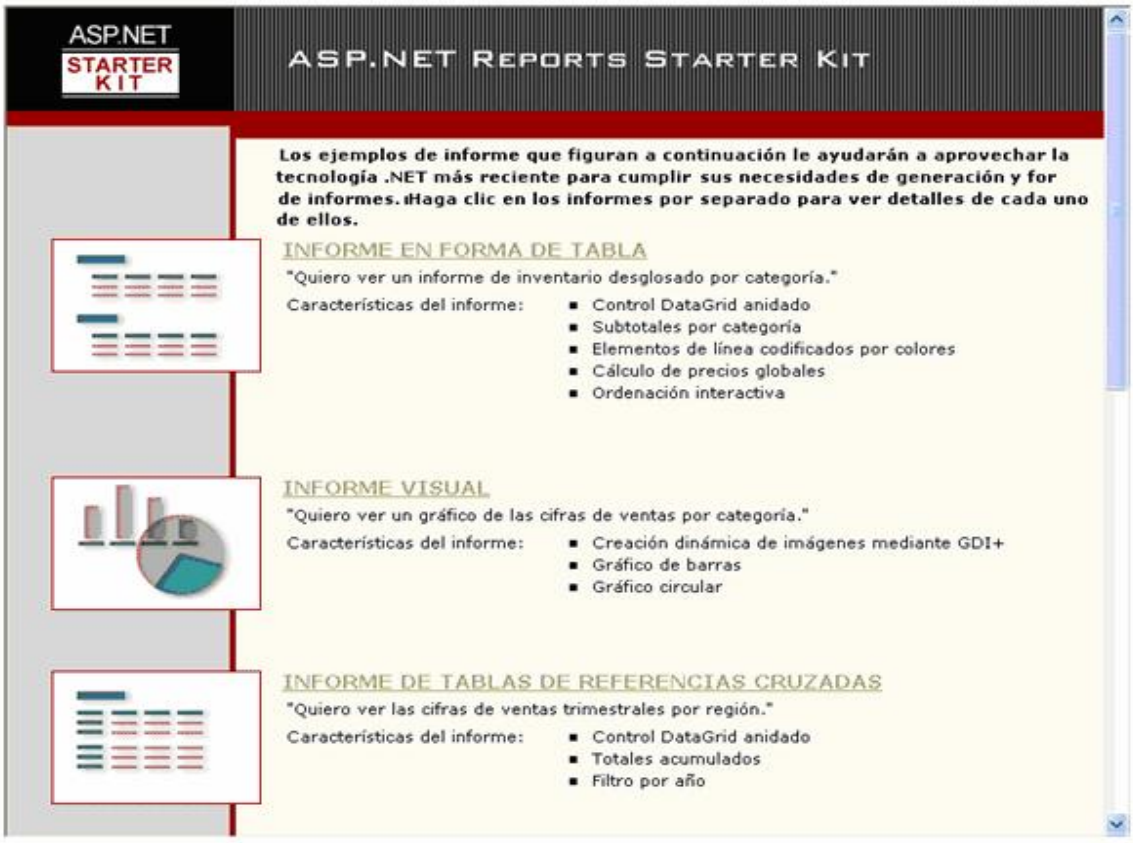


Figura 4.30. Página de inicio de Reports.

4.7.4 Página de detalles del informe

La **página de detalles** está dividida en dos marcos. *El **marco superior** corresponde a la exploración y el **marco inferior** al contenido*.

La parte resaltada de la imagen que se muestra en la **Figura 4.31** es el ***marco de exploración***. Contiene vínculos a la información general, las versiones activa e imprimible, el código fuente de la página, la capa de lógica empresarial y los procedimientos almacenados. El ***marco inferior*** cambia en función del vínculo seleccionado por el usuario en el ***marco de exploración***.

Los usuarios pueden hacer *clic* en el vínculo **"Presentación de informes"** para volver a la página de introducción.

Figura 4.31. Detalles del informe en forma de tabla.

4.7.5 Informes de muestra del Kit de inicio Reports

Como ya se menciono al inicio, el **Kit de inicio Reports** cuenta con ocho tipos de informes que pueden se estudiados e implementados para los proyectos de los desarrolladores.

4.7.5.1 Informe en forma de tabla

El **informe en forma de tabla** *presenta una forma básica de agrupamiento de datos relacionados (en este caso, agrupamiento de productos por categorías)*. El agrupamiento se realiza anidando un control **DataGrid** dentro de un control **DataList**. La **Figura 4.32**, muestra cómo funcionan conjuntamente los controles **DataList** y **DataGrid**. El control **DataList** muestra el nombre de la categoría e información resumida, mientras que el control **DataGrid** devuelve los detalles de los productos basándose en el Id. que recibe del control **DataList** para cada categoría.

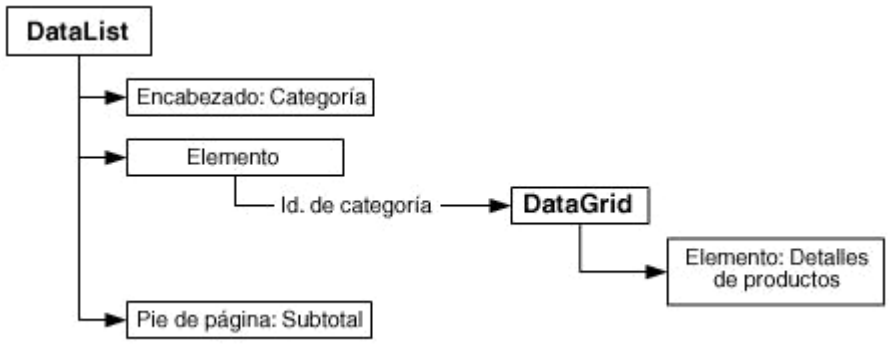


Figura 4.32. Interacción de DataList y DataGrid.

En este informe, la ordenación se habilita para cada columna en el ***encabezado de detalles de productos***, exceptuando las columnas de cantidad por unidad y de precio global. Las columnas que admiten la ordenación se pueden ordenar de forma *ascendente o descendente*.

Para indicar qué selección se ha realizado, estas columnas se muestran en *color rojo* y subrayadas cuando el usuario detiene el puntero sobre el ***encabezado de la columna*** como se muestra en la **Figura 4.33.**

Categoría Beverages

Producto	Existencias	Cantidad por unidad	Precio unitario	Precio global
Guaran Fantstica	20	12 - 355 ml cans	4,50 $	90,00 $
Rhnbru Klosterbier	125	24 - 0.5 l bottles	7,75 $	968,75 $
Sasquatch Ale	111	24 - 12 oz bottles	14,00 $	1.554,00 $
Laughing Lumberjack Lager	52	24 - 12 oz bottles	14,00 $	728,00 $
Outback Lager	15	24 - 355 ml bottles	15,00 $	225,00 $

Figura 4.33. Ordenación de DataGrid

Además de la ordenación, el control **DataGrid** también ofrece la posibilidad de *controlar selectivamente el formato de una fila basándose en criterios personalizados*. En este caso, los productos sin existencias (designados con la cantidad cero) aparecen en rojo como se muestra en la **Figura 4.34**.

Categoría Meat/Poultry

Producto	Existencias	Cantidad por unidad	Precio unitario	Precio global
Alice Mutton	0	20 - 1 kg tins	39,00 $	0,00 $
Mishi Kobe Niku	29	18 - 500 g pkgs.	97,00 $	2.813,00 $
Perth Pasties	0	48 pieces	32,80 $	0,00 $
Pt chinois	115	24 boxes x 2 pies	24,00 $	2.760,00 $
Thringer Rostbratwurst	0	50 bags x 30 sausgs.	123,79 $	0,00 $
Tourtire	21	16 pies	7,45 $	156,45 $

Figura 4.34. Formato selectivo de DataGrid.

4.7.5.2 Informe visual

El **informe visual** muestra las ventas por categoría en tres vistas diferentes: ***Gráfico circular, Gráfico de barras y Vista tabular***. Este informe utiliza **GDI+** para generar dinámicamente gráficos circulares y de barras. ***GDI+*** *es una infraestructura para crear imágenes y gráficos*.

La vista predeterminada es un ***gráfico circular***. El ***gráfico circular*** *ilustra la contribución de subvalores a un total*. La segunda vista es el ***gráfico de barras***, *que permite comparar valores diferentes representándolos uno al lado del otro*. La última vista es la ***vista tabular*** *que muestra los datos utilizados en estos dos gráficos*.

Los valores de los ***gráficos circular y de barras*** se muestran en distintos colores y acompañados de la leyenda correspondiente. La **Figura 4.35** es una captura de pantalla de un ***gráfico circular*** y su leyenda asociada.

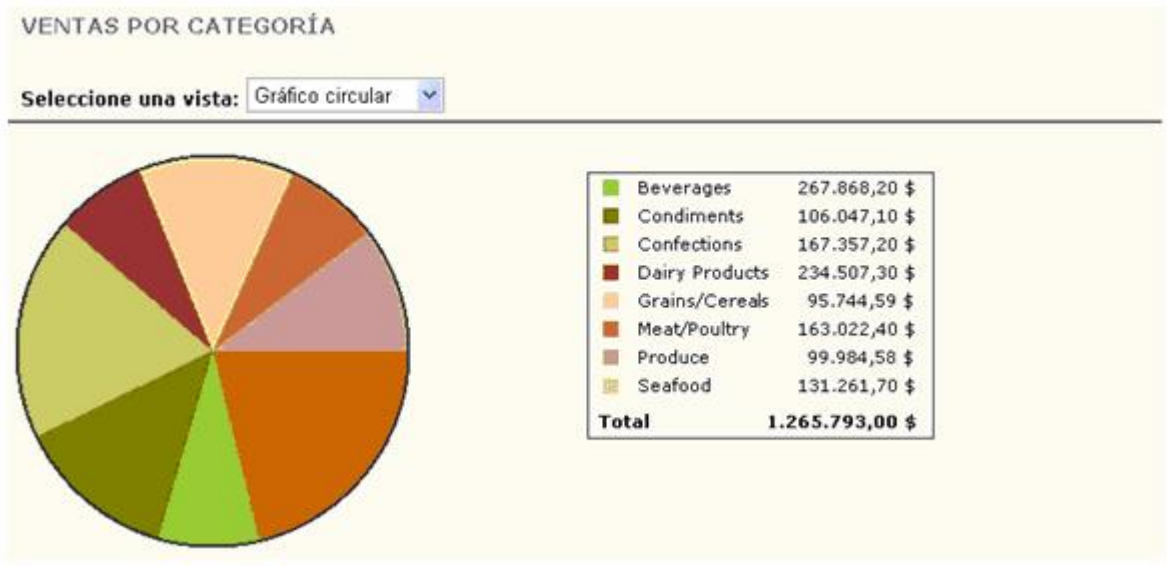


Figura 4.35. Gráfico circular generado por GDI+.

4.7.5.3 Informe con tabla de referencias cruzadas

El **informe de tablas de referencias cruzadas** muestra un *desglose trimestral* de las ventas basándose en las regiones geográficas. También muestra los *totales correspondientes a cada región*, los *totales correspondientes a cada mes del trimestre* y el *total correspondiente al trimestre completo*.

Los *totales correspondientes a cada región* se muestran en el pie del control **Datagrid**. Los *totales correspondientes a cada mes del trimestre* se muestran en la columna situada más a la derecha. El *total general correspondiente al trimestre* se calcula y se muestra en la esquina inferior derecha. La **Figura 4.36** muestra una sección del informe de tablas de referencias cruzadas correspondiente a un trimestre.

Mes	Oriental	Occidental	Meridional	Septentrional	Totales
julio	17.990,60 $	0,00 $	12.201,50 $	0,00 $	**30.192,10 $**
agosto	10.746,00 $	0,00 $	15.324,80 $	538,60 $	**26.609,40 $**
septiembre	11.175,70 $	2.038,80 $	13.813,50 $	608,00 $	**27.636,00 $**
Totales	39.912,30 $	2.038,80 $	41.339,80 $	1.146,60 $	**84.437,50 $**

Figura 4.36. Informe de tablas de referencias cruzadas.

Este informe utiliza la misma técnica de agrupamiento de datos relacionados que se utiliza en el informe en forma de tabla (en este caso, el agrupamiento de datos de ventas por trimestre). El agrupamiento se realiza anidando un control **DataGrid** dentro de un control **DataList**, como se muestra en la **Figura 4.37**.

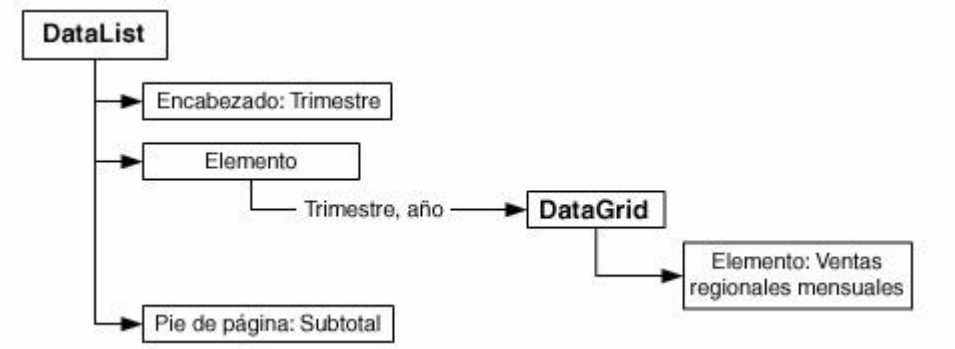


Figura 4.37. Anidamiento de DataGrid.

4.7.5.4 Informe de detalles principales

Este informe muestra una forma sencilla de crear y enlazar a datos dos controles **DataGrid** diferentes en la misma página, uno para el resumen y el otro para los detalles. Ambos controles **DataGrid** *se enlazan a dos procedimientos almacenados diferentes*.

Para el resumen, los resultados se filtran por año y, para los detalles, los resultados se filtran por año y, opcionalmente, por trimestre.

Dos controles **DataGrid** independientes se crean y se enlazan a dos procedimientos almacenados diferentes denominados ***GetOrderSummary*** y ***GetOrderDetails***. Esto se realiza llamando a ***GetSales*** y ***GetSalesDetails*** del componente ***MasterDetail*** en la capa de lógica empresarial, que utilizará el componente **SQLHelper** parte de **Data Access Application Blocks *(DAAB)***.

El objeto ***YearDropDownList*** *se utiliza para pasar el año por el que se filtran los resultados*.

El resumen desglosa el ***número de pedidos enviados*** y el ***volumen de ventas por trimestre***. En la parte inferior del resumen se muestra un informe del ***Total de ventas*** del año (**Figura 4.38**).

Trimestre	Pedido enviado	Ventas
3	70	79.728,57 $
4	82	128.355,40 $
Total de ventas		208.083,97 $

Figura 4.38. Total de ventas.

Para calcular el ***Total de ventas***, se agrega un controlador de eventos al control **DataGrid** del resumen. Se realiza una iteración en las filas del control **DataGrid** y se utiliza la tercera celda de cada fila, que representa la columna Ventas, para calcular el total de ventas. A continuación, este total se agrega de nuevo a la fila del pie del control **DataGrid** actual.

Los detalles muestran una lista de todos los pedidos del año y, opcionalmente, del trimestre, notificando el **Id. de pedido**, la **fecha de pedido** y el **volumen de ventas** (**Figura 4.39**).

Id. del pedido	Fecha de los pedidos	Ventas
10248	04/07/1996	440,00 $
10249	05/07/1996	1.863,40 $
10250	08/07/1996	1.552,60 $
10251	08/07/1996	654,06 $
10252	09/07/1996	3.597,90 $
10253	10/07/1996	1.444,80 $
10254	11/07/1996	556,62 $

Figura 4.39. Detalles de pedidos.

El usuario puede especificar el año y el trimestre seleccionándolos en los objetos ***YearDropDownList*** y ***QuarterDropDownList*** situados en la parte superior de la página.

4.7.5.5 Informe sencillo

El **informe sencillo** *muestra una lista de toda la información de contacto de los clientes relacionados con la empresa* (**Figura 4.40**). Esta información es muy útil para el personal de ventas que está viajando constantemente, porque se puede tener acceso a ella desde cualquier lugar a través del Web.

El **informe sencillo** muestra el uso del control **DataGrid** para enumerar un conjunto de resultados de la *Base de Datos*.

Compañía	Contacto	Puesto	Teléfono	Ciudad
Alfreds Futterkiste	Maria Anders	Sales Representative	030-0074321	Berlin
Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	(5) 555-4729	Mxico D.F.
Antonio Moreno Taquera	Antonio Moreno	Owner	(5) 555-3932	Mxico D.F.
Around the Horn	Thomas Hardy	Sales Representative	(171) 555-7788	London
Berglunds snabbkp	Christina Berglund	Order Administrator	0921-12 34 65	Lule
Blauer See Delikatessen	Hanna Moos	Sales Representative	0621-08460	Mannheim
Blondesddsl pre et fils	Frdrique Citeaux	Marketing Manager	88.60.15.31	Strasbourg
Blido Comidas preparadas	Martn Sommer	Owner	(91) 555 22 82	Madrid
Bon app'	Laurence Lebihan	Owner	91.24.45.40	Marseille
Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	(604) 555-4729	Tsawassen

Figura 4.40. DataGrid sencillo.

El control **DataGrid** también se puede utilizar para *admitir las funciones de eliminar y editar*. Sin embargo, la *finalidad de este informe es mostrar lo simple y rápido que resulta presentar datos de una base de datos en un* ***formulario Web Form ASP.NET*** *con las funciones de paginación y ordenación*.

Para permitir la paginación, se ha de establecer la propiedad ***AllowPaging*** del control **DataGrid** en ***True*** y crear un controlador de eventos, ***PageIndexChanged***, para cambiar la propiedad ***CurrentPageIndex***.

4.7.5.6 Informe de texto

El **informe de texto** *muestra una lista de empleados, información sobre sus direcciones y notas acerca de su historial personal* (**Figura 4.41**). Este informe muestra la forma de utilizar el control **Repeater** para enumerar datos repetitivos, como por ejemplo los datos de texto.

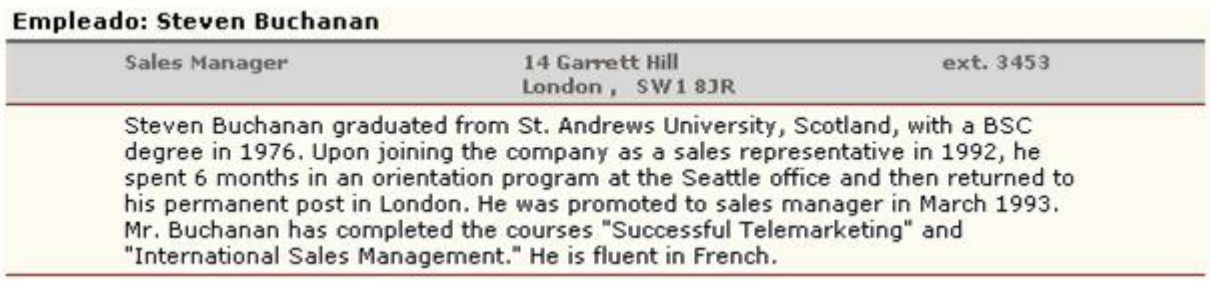
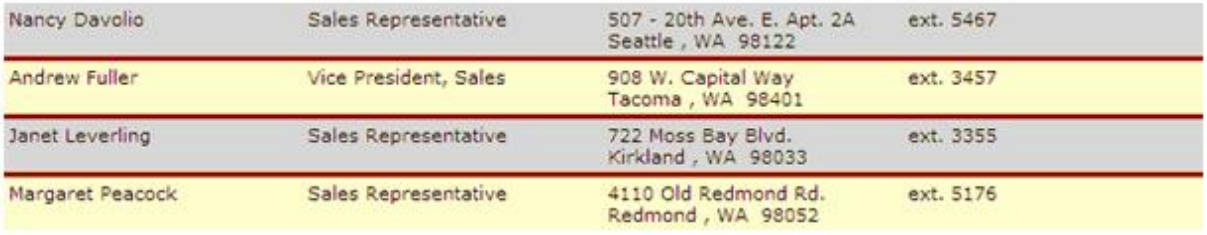


Figura 4.41. Presentación del empleado mediante el uso de un control Repeater.

El origen de datos del control **Repeater** es un objeto ***TextReportCollection*** devuelto por la función ***GetEmployees***.

El control **Repeater** *utiliza los valores de la colección para procesar la información apropiada del informe*. Al igual que los controles **DataList** y **DataGrid**, el control **Repeater** tiene una plantilla de encabezado, otra de pie y otra de elemento.

Además, este control también tiene una ***plantilla de elemento alterno*** para distinguir una fila de la siguiente, y una ***plantilla de separador*** para definir datos o un estilo, como una línea de separación que se mostrará entre los elementos de datos. En la **Figura 4.42** se muestra un ejemplo de su utilización.



Nancy Davolio	Sales Representative	507 - 20th Ave. E. Apt. 2A Seattle , WA 98122	ext. 5467
Andrew Fuller	Vice President, Sales	908 W. Capital Way Tacoma , WA 98401	ext. 3457
Janet Leverling	Sales Representative	722 Moss Bay Blvd. Kirkland , WA 98033	ext. 3355
Margaret Peacock	Sales Representative	4110 Old Redmond Rd. Redmond , WA 98052	ext. 5176

Figura 4.42. Plantilla de elemento alterno.

4.7.5.7 Informe jerárquico

El **informe jerárquico** *contiene tres cuadrículas diferentes*. ***La primera cuadrícula*** muestra los totales de ventas agrupados por área y filtrados por año. ***La segunda* cuadrícula** muestra los totales de ventas agrupados por empleado dentro de un área concreta, a saber, el área seleccionada en la ***primera cuadrícula***. ***La tercera cuadrícula*** muestra información acerca de un empleado en particular, a saber, el empleado seleccionado en la ***segunda cuadrícula***.

Tres controles **DataGrid** se configuran en esta página, cada uno con su propio origen de datos, que se recuperan de tres procedimientos almacenados diferentes. Sólo el origen de datos correspondiente a la ***primera cuadrícula***, "**Ventas por área**", está enlazado a **Page_Load** y no cambia; las otras dos cuadrículas permanecen ocultas hasta que el usuario selecciona el área***(Territory)*** y después el empleado***(Employee)***.

Cuando el usuario selecciona el área que desea ver, la ***segunda cuadrícula***, ***"*Ventas correspondientes a [Área]*"***, se hace visible y se enlaza a los resultados

del procedimiento almacenado ***"GetEmployeeSalesByTerritory"***, que acepta el nombre del área como parámetro.

De forma similar, cuando el usuario selecciona el empleado que desea ver, la ***tercera cuadrícula***, "**Información del empleado [Empleado]**", se hace visible y se enlaza a los resultados del procedimiento almacenado ***"GetEmployeeByID"***, que acepta el **Id. del empleado** como parámetro. Si el usuario hace *clic* en la **cuadrícula Área** cuando la **cuadrícula Empleado** está visible, ésta se ocultará y la **cuadrícula Ventas de empleado por área** se actualizará con los nuevos datos.

Para que el usuario pueda realizar una selección en una cuadrícula y utilizarla para rellenar la siguiente cuadrícula, se utiliza una columna **Template** como primera columna en los dos controles **DataGrid** superiores. Se utiliza una columna **Template** para crear una columna con un diseño de control personalizado, con aspectos personalizados para las secciones de encabezado, de pie y de elementos de la columna.

Además, este informe *muestra cómo se implementan la ordenación y la paginación interactivas utilizando* **ASP .NET**. La ordenación y la paginación se realizan de forma similar a la implementación que se describe en el informe en forma de tabla (**Figuras 4.43 y 4.44**).

Figura 4.43. Ordenación de DataGrid.

Figura 4.44. Paginación de DataGrid.

4.7.5.8 Informe detallado

El **informe detallado** *muestra información de pedidos enviados a clientes*. A los usuarios se les proporciona una lista de clientes. En ella, pueden hacer *clic* en un cliente y ver los *pedidos enviados a dicho cliente*. Además, pueden hacer *clic* en un **Id. de pedido** para ver los *detalles correspondientes a ese pedido concreto*.
Este informe muestra una forma de presentar una jerarquía de datos, en este caso ***clientes --> pedidos --> detalles de pedidos***.

La ***lista de clientes*** es un control **DataList**. Dentro de la plantilla seleccionada del control **DataList de Clientes *(Customers)*** está el control **DataList de Pedidos *(Orders)***. Y dentro de la plantilla seleccionada del control **DataList de Pedidos** está el control **DataGrid de Detalles de pedidos *(Order Details)***.
La plantilla seleccionada determina el contenido y la disposición del elemento seleccionado del control **DataList**.

El control de servidor **DataList** se utiliza para las listas de clientes y de pedidos debido a su función de plantilla seleccionada. El control de servidor **Datagrid** se utiliza para presentar fácilmente los detalles de pedidos porque no requiere una plantilla seleccionada.

Para mostrar la plantilla seleccionada de un elemento, se agrega un evento a ***CustomerList***. También se agrega un controlador de eventos a ***OrdersList***. El controlador de eventos comprueba si se llama a un comando ***"select"*** y establece el índice seleccionado del control **DataList** correspondiente y, a continuación, vuelve a enlazar el control **DataList**. En la **Figura 4.45** se muestra el diagrama de este proceso.

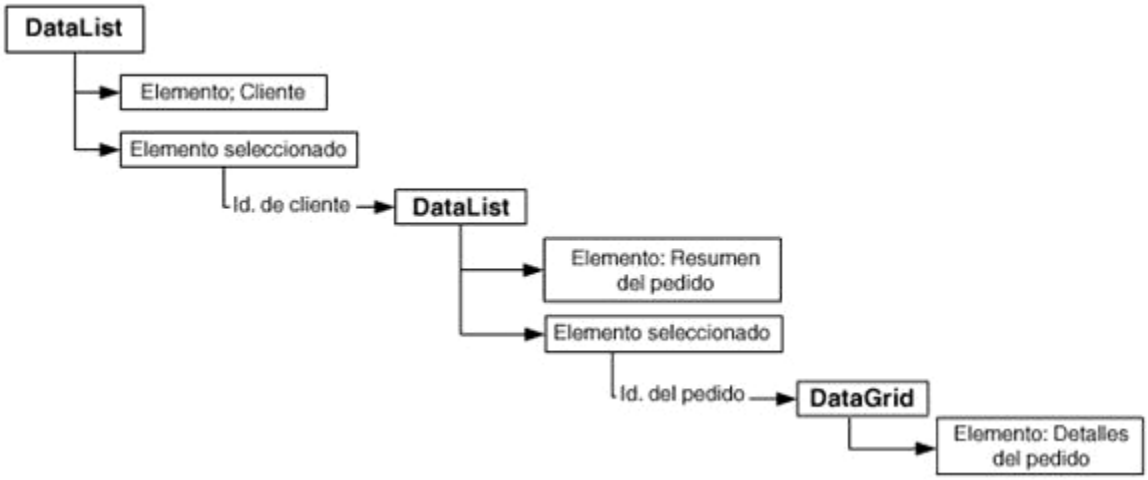


Figura 4.45. Diseño del control de servidor DrillDown.

CAPÍTULO 5

Seguridad en ASP .NET

5.1 Factores de Seguridad

Un aspecto de suma importancia dentro de las *aplicaciones Web* es, sin lugar a duda, la ***seguridad***, ya que con frecuencia los sitios Web incluyen información importante o sensible ejecutada sobre la red.

La ***Seguridad*** es primordial para los *desarrolladores de las aplicaciones y para los administradores de las mismas*, por lo que se requiere de una cautelosa planeación y designación sobre este aspecto.

Dentro del ***ambiente ASP .NET*** existen tres factores primordiales involucrados en la seguridad, estos aspectos son los siguientes:

1. **Autenticación**.- Éste se refiere al proceso de validación o identificación de un usuario, para permitirle o negarle una solicitud realizada. El proceso de autenticación comúnmente hace referencia a la solicitud de un ***username*** y un ***password***, los cuales son validados tomando como referencia la base de datos asignada para la seguridad.

2. **Autorización**.- Éste proceso se refiere a garantizar que el usuario validado tiene permitido el acceso sólo a los recursos sobre los cuales se le han asignado derechos.

3. **Personalización**.- Éste proceso permite a una aplicación adoptar un perfil o identidad del solicitante de los recursos, accediendo a los recursos que estén concedidos o denegados de acuerdo con el perfil o identidad asumido.

5.2 Seguridad en IIS

La ***seguridad*** *de una aplicación Web* debe comenzar por los métodos de seguridad que otorgue el Servidor Web hacia la aplicación Web establecida en la parte cliente. Por lo tanto **IIS** proporciona el soporte necesario para la *autenticación y autorización* de peticiones de usuario.

5.2.1 Autenticación en IIS

El servidor IIS proporciona autenticación a los clientes que realizan peticiones de contenido Web almacenado en un **sitio Web IIS**.

Bajo el contexto anterior existen diversos tipos de autenticación, la cual puede ser implementada utilizando un **Servidor IIS**:

- **Autenticación Anónima**.- Permite a los usuarios navegar o utilizar la aplicación *sin la necesidad de establecer un username y password*.

- **Autenticación Básica**.- Se refiere a la *especificación de un username y password por parte del usuario para el acceso al sitio Web*. Una característica importante de este método es que el password es enviado en forma encriptada a través de la red, lo cual hace posible la fácil recuperación a través de paquetes de la red de la información referente al password.

- **Autenticación Integrada Windows**.- Ésta *requiere que los usuarios sean validados por* ***Windows*** *además de la autenticación básica*. En este **método IIS** verifica el username y password un **Controlador de Dominio Windows** y el acceso al sitio Web será permitido sólo si este controlador de dominio valida dicho username y password.

- **Autenticación Abreviada**.- Ésta es similar a la autenticación básica, pero este método utiliza una manera diferente la transmisión de credenciales de autenticación. Esta autenticación *envía un valor a través de la red antes que el password, el cual se envía de* ***modo encriptado****.*

5.2.2 Autorización en IIS

Dentro de **IIS** es posible realizar la configuración necesaria para controlar los recursos que serán accedidos por los usuarios. Es posible controlar los permisos de acceso en un **sitio Web IIS** y permitir ciertas operaciones sobre este sitio.

Los niveles de autorización que son incluidos son los siguientes:

- **Lectura**.- Permite a los usuarios recuperar y leer algún contenido almacenado en un directorio virtual.

- **Escritura**.- Permite a los usuarios modificar el contenido almacenado en un directorio virtual.

- **Acceso a recursos de código**.- Permite los usuarios ver el contenido referente a código o programas del lado del servidor.

- **Desplegado de directorio**.- Permite a los usuarios visualizar el contenido de un directorio virtual, lo cual es algo similar a la visualización de un fólder **FTP**.

- **Libro de visitas.-** Registra el número de usuarios que han visitado el sitio y almacena información detallada como la ***dirección IP*** del cliente que acceso y los recursos que fueron solicitados por éste.

- **Índice**.- Es utilizado para indexar el ***directorio virtual*** por el *Servidor de Índices* ***Microsoft***. Los contenidos de este directorio puede ser obtenidos mediante un resultado de búsqueda utilizando el Servidor de Índice.

5.3 Opciones de Autenticación en ASP .NET

Dentro del archivo **Web.Config** existe una sección enfocada a la seguridad, la cual contiene información referente al nivel y tipo de servicio de autenticación que puede ser otorgado a una **aplicación Web**.

El archivo **Web.Config** es un **archivo XML** y se encuentra ubicado en el **directorio raíz *(root directory)*** de la *aplicación Web*. Gran cantidad de opciones de configuración de una **aplicación Web ASP .NET** pueden ser controladas y configuradas desde este **archivo XML**.

La sección **System.Web** del archivo ***Web.Config*** es utilizada para controlar diferentes aspectos referentes a la seguridad de una *aplicación Web*.

Las ***aplicaciones Web ASP .NET*** permiten establecer los siguientes tipos de seguridad:

1. **Windows.-** En este tipo de seguridad la *aplicación es protegida utilizando la* ***Autenticación Windows Integrada***. En este método el acceso a la aplicación es permitido sólo cuando los usuarios se encuentran habilitados verificando su credencial Windows. Dichas credenciales son verificadas contra la ***Base de Datos de Autenticación Windows*** o en su defecto contra un ***Directorio Activo***.

2. **Passport.-** La *aplicación es protegida mediante una autenticación* ***Microsoft Passport***. ***Passport*** es una tecnología desarrollada por ***Microsoft*** utilizada comúnmente sobre *aplicaciones Web*.

3. **Forms.-** La *aplicación es protegida utilizando un Modelo de autenticación de recursos con soporte para* ***cookies.***

5.3.1 Autenticación Windows

En algunos casos las aplicaciones Web pueden necesitar tener **Seguridad otorgada por Windows**. Tal es el caso de una aplicación Web de una ***intranet***, la cual proporciona diferente niveles de acceso para sus usuarios dependiendo de las credenciales Windows.

En el caso señalado anteriormente las *aplicaciones Web* pueden protegerse utilizando la **Autenticación de Windows**. Para realizar esto, basta seleccionar el modo ***"Autenticación Windows"*** en el archivo **Web.config**, tomando como referencia el siguiente código:

```
<configuration>
<system.web>
<authentication mode="Windows" />
</system.web>
</configuration>
```

Además de realizar lo anterior, también es necesario seleccionar la **Autenticación Integrada** en el ***directorio virtual*** utilizado por el **Administrador IIS**, esto para que la Autenticación Integrada de Windows funcione de forma adecuada.

Para ***habilitar la Autenticación Integrada de Windows*** para una aplicación Web se deberán llevar a cabo los siguientes pasos:

1. **Iniciar la Herramienta de Administración IIS**, la cual puede ser abierta desde el panel de control y abrir el cuadro de diálogo **Propiedades** dando *clic derecho* sobre la aplicación Web, como se muestra en la **Figura 5.1.**

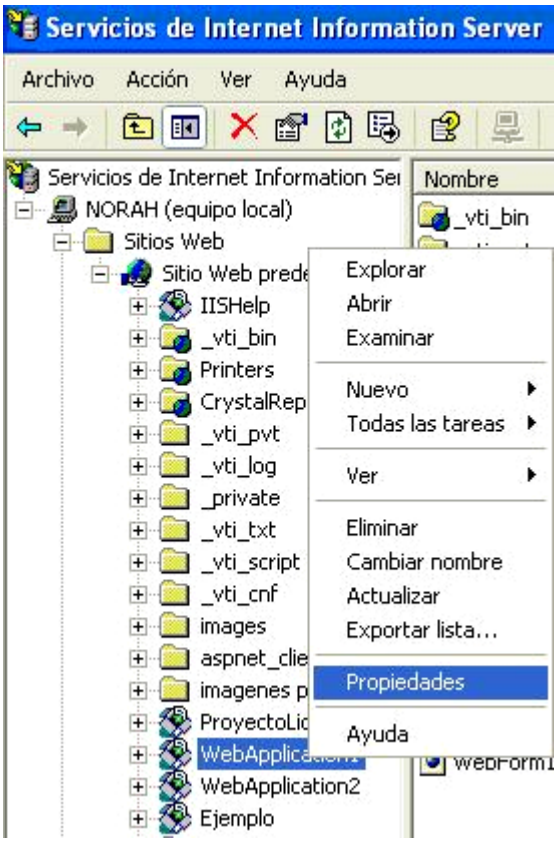


Figura 5.1. Propiedades.

2. **Seleccionar la opción seguridad de directorios** y enseguida dar *clic* sobre la opción modificar, se desplegará una ventana titulada **Métodos de Autenticación** como lo muestra la **Figura 5.2**.

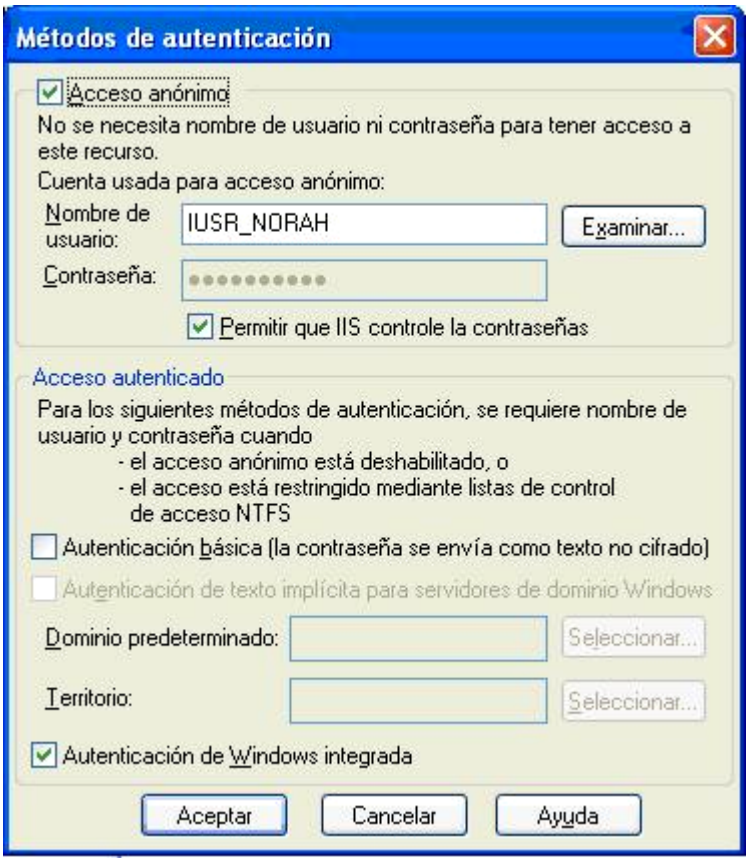


Figura 5.2. Métodos de Autenticación.

3. Como paso final se deberá **verificar que la casilla Autenticación de Windows integrada** se encuentre seleccionada, también mostrada en la **Figura 5.2**.

Una vez realizado lo anterior, es *posible* ***implementar la Autenticación Integrada de Windows*** siguiendo los siguientes pasos:

1. Se deberá *modificar el código* dentro del archivo **Web.config** para establecer el ***modo de Autenticación apropiado para Windows***, el código es el siguiente:

```
<configuration>
<system.web>
```

```
<authentication mode="Windows"/>
<authorization>
<!-- anonymous users will be denied access -->
<deny users="?"/>
</authorization>
</system.web>
</configuration>
```

2. Después se deberá *modificar la página* en la que se determinó identificar al usuario, el código es el siguiente:

```
<%@ Import Namespace="System.Web.Security " %>
<html>
<script language="VB" runat=server>
Sub Page_Load(Src As Object, E As EventArgs)

 'Use the User object to retrieve info about the current user
 Welcome.Text = "Hello, " + User.Identity.Name
 End Sub
</script>
<body>
<h3><font face="Verdana">Using Integrated Windows
Authentication</font></h3>
<form runat=server>
<h3><asp:label id="Welcome" runat=server/></h3>
</form>
</body>
</html>
```

Finalmente, en la página se mostrará lo representado por la **Figura 5.3**.

Figura 5.3. Default page.

5.3.2 Autenticación basada en Formularios

ASP .NET incluye esta forma de autenticación, la cual permite implementar recursos lógicos para la ***autenticación de usuarios y autenticación de manejadores*** sin tener que preocuparse acerca del manejo de sesiones utilizando cookies.

En este tipo de autenticación cuando un usuario requiere ser identificado, éste es ***redireccionado*** automáticamente a la ***página del login***. Éste y algunos otros beneficios son generados por éste método, enseguida se mencionan los más importantes:

- Los usuarios pueden *configurar este tipo de autenticación en varias secciones del sitio Web*, debido a que el archivo ***Web.config*** es un **documento XML** organizado de forma jerárquica.

- Los Administradores y Desarrolladores *pueden modificar el esquema de autenticación fácil y rápidamente* por medio del archivo ***Web.config***.

- *La Administración se encuentra de forma centralizada* debido a que todas las entradas de autenticación se encuentran en una parte del archivo ***Web.config***.

5.3.3 Implementación de Autenticación basada en Formularios

Para implementar la autenticación basada en formularios en una **aplicación Web ASP .NET**, deberán llevarse a cabo los siguientes pasos:

1. *Activar el modo **Autenticación** para formularios en el archivo **Web.config**.* Dicho archivo se localiza en el mismo directorio de la **Aplicación Web ASP .NET**.

2. Enseguida se deberá especificar el siguiente código:

```
<configuration>
<system.web>
<authentication mode="Forms">

<!—Asignar una cookie llamada B2CbuySiteAuthCookie
 cuando el usuario sea autentificado. La pagina
 utilizada para validar la credencial del usuario es
 userauth.aspx. Asegurándose de que la cookie se
```

```
 encuentra encriptada y validada por la opción de pro-
 tección, la cookie terminara su tiempo después de 10
 minutos →

<forms name=".B2CBuySiteAuthCookie" loginUrl=
"userauth.aspx" protection="All" timeout="10" />
</authentication>
<authorization>

<!—usuarios anónimos que tendrán denegado el acceso
   Esto es necesario para forzar la autenticación→
<deny users="?"/>
</authorization>
</system.web>
</configuration>
```

3. El paso siguiente será *crear la **página login** la cual aceptará y validará la credencial del usuario antes de asignar al cliente una sesión cookie*. La página login redireccionará al usuario a la página solicitada por el cliente.

4. Después será necesario *escribir el código siguiente en el archivo **Userauth.aspx***, que es la abreviatura de Autenticación de Usuario, puede establecerse cualquier otro nombre.

```
<%@ Import Namespace="System.Web.Security " %>
<html>
<script language="VB" runat=server>
' Verify credentials
Sub Login_Click(Src As Object, E As EventArgs)
' Do complex hashing and look up other data sources to
```

```
' determine validity ;)

If UserName.Value = "john" And UserPass.Value = "secret"
‘La credencial es valida redirecciona de regreso a la
‘pagina que forza la autenticación.

FormsAuthentication.RedirectFromLoginPage(UserName.Value,F
alse)

Else

Msg.Text = "Invalid user name or password: Please try again"

End If
End Sub
</script>

<body>
<form runat=server>
<h3><font face="Verdana">Please Sign-In</font></h3>
<table>
<tr>
<td>Login Name:</td>
<td><input id="UserName" type="text" runat=server/></td>
</tr>
<tr>
<td>Password:</td>
<td><input id="UserPass" type=password runat=server/></td>
</tr>
</table>
<asp:button text="Login" OnClick="Login_Click" runat=server/>
<asp:Label id="Msg" ForeColor="red" Font -Name="Verdana"
```

```
Font-Size="10" runat=server />
</form>
</body>
</html>
```

5. Finalmente será necesario *crear el archivo* ***Default.aspx*** con el código siguiente:

```
<%@ Import Namespace="System.Web.Security " %>
<html>
<script language="VB" runat=server>
 Sub Page_Load(Src As Object, E As EventArgs)

‘Uso del Objeto usuario para regresar información del usuario
‘común
Welcome.Text = "Hello, " + User.Identity.Name
End Sub

Sub Signout_Click(Src As Object, E As EventArgs)
FormsAuthentication.SignOut()
Response.Redirect("userauth.aspx")

End Sub
</script>

<body>
<h3><font face="Verdana">Using Forms
Authentication</font></h3>
<form runat=server>
<h3><asp:label id="Welcome" runat=server/></h3>
<asp:button text="Signout" OnClick="Signout_Click"
```

```
runat=server/>
</form>
</body>
</html>
```

Al realizar la ejecución del código anterior se podrá visualizar una ventana como la mostrada en la **Figura 5.4**.

Figura 5.4. Autenticación del Usuario.

Conclusiones

El surgimiento de la **tecnología *Microsoft* .NET** crea un nuevo paradigma para la creación de aplicaciones Web basada en la interacción de sus elementos, los cuales están orientados al aprovechamiento de todos los recursos computacionales con los que se cuenta desde la creación de las aplicaciones hasta la ejecución de las mismas.

Los beneficios de **ASP .NET** se orientan principalmente a facilitar la labor de los desarrolladores, dándoles una amplia gama de herramientas y un entorno de trabajo con el cual se encuentran familiarizados, pero que a la vez presenta grandes innovaciones. Además es posible satisfacer las necesidades del usuario con mayor facilidad, ya que el ciclo de desarrolló utilizado resulta demasiado flexible y por ende el producto final creado es de mayor calidad.

La **tecnología .NET** es un parte aguas en la evolución de las aplicaciones Web surgiendo en base a la demanda de un entorno globalizado, por lo que debe ser considerada ampliamente como la tecnología del presente y del futuro.

Glosario

API. Siglas correspondientes a ***"Aplication Program Interface" o "Interfaz de Programas de Aplicación".*** Corresponde a una biblioteca o bibliotecas que ofrece el sistema operativo para que los programas puedan comunicarse con él e invocar sus servicios.

B2B. Abreviatura comercial de la expresión anglosajona ***business to business***: *comercio electrónico entre empresas*.

Caché. Memoria intermedia utilizada por los navegadores para almacenar las páginas WWW que ya se han visitado y poder volver a presentarlas sin necesidad de cargarlas de nuevo desde la red.

COM. Es la extensión que corresponde a un tipo de archivo ejecutable bajo el ambiente MS-DOS.

Cookies. Pequeños archivos de texto que un servidor Web almacena en la computadora del usuario, para guardar información sobre éste, como un número de identificación, una contraseña, sus preferencias o cuántas veces ha visitado el sitio el usuario. Un sitio Web puede tener acceso a la información del cookie siempre que el usuario se conecta al servidor.

CORBA. Sigla correspondiente a Common ***Object Request Broker Architecture*** es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

Dirección IP. Son el método mediante el cual se identifican los ordenadores individuales (o, en una interpretación más estricta, las interfaces de red de dichos ordenadores) dentro de un red TCP/IP.

DLL. Es el acrónimo de ***Dynamic Linking Library (Bibliotecas de Enlace Dinámico)***, término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda del programa por parte del sistema operativo. Esta denominación se refiere a los sistemas operativos Windows siendo la extensión con la que se identifican los archivos, aunque el concepto existe en prácticamente todos los sistemas operativos modernos.

DTD. Siglas correspondientes a ***Document Type Definition***. Es un documento que describe la estructura de una página Web escrita en XML.

FAQ. Son las iniciales de ***Frequently Asked Questions***, que en español significa preguntas más frecuentes. Las FAQ son las dudas más comunes que tienen los visitantes de un sitio web, y que por ello se contestan en una o varias páginas de ese sitio.

FTP. Es uno de los diversos protocolos de la red Internet, concretamente significa ***File Transfer Protocol (Protocolo de Transferencia de Archivos)*** y es el ideal para transferir datos por la red.

GDI. En computación, siglas de ***Graphics Device Interface***, el motor gráfico de los sistemas Windows de Microsoft.

HTML. Acrónimo inglés de ***Hyper Text Markup Language (lenguaje de marcación de hipertexto)***, es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Explorer o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

HTTP. Es el protocolo de la Web (WWW), usado en cada transacción. Las letras significan ***Hyper Text Transfer Protocol***, es decir, *protocolo de transferencia de*

hipertexto. El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceder a una página web, y la respuesta de esa web, remitiendo la información que se verá en pantalla.

IDE. Son las siglas de: ***Integrated Development Environment***, es decir, un Entorno Integrado de Desarrollo.

Intranet. Es una red local que utiliza herramientas de Internet. Se puede considerar como una Internet privada que funciona dentro de una organización. Normalmente, dicha red local tiene como base el protocolo TCP/IP de Internet y utiliza un sistema firewall (cortafuegos) que no permite acceder a la misma desde el exterior.

Just – in – time. La compilación just-in-time (JIT), también conocida como traducción dinámica, es una técnica para mejorar el rendimiento de la compilación de código binario de los sistemas de programación por medio de la traducción del código binario en código maquina nativo en tiempo de ejecución.

MFC. Son las siglas de Microsoft Foundation Classes, una serie de funciones ya creadas para que se puedan utilizar al crear programas para Windows.

Modo encriptado (Encriptar). Es un método de protección de información. Aplicar operaciones matemáticas a un texto para convertirlo en información imposible de leer, en un lenguaje cifrado.

Pager. Pequeño dispositivo de telecomunicación donde se reciben mensajes que aparecen escritos en un display. La comunicación se establece por teléfono y también por e-mail y es de una sola vía: el usuario debe responder el llamado comunicándose por otro medio. Varias firmas han anunciado que ofrecerán un servicio de doble vía, es decir, con posibilidad de respuesta.

Password. Una contraseña (password en inglés) o clave, es una forma de autenticación que utiliza una información secreta para controlar el acceso hacia algún recurso.

PDA. De las siglas en inglés ***Personal Digital Assistant (Ayudante personal digital)*** es un ordenador de mano originalmente diseñado como agenda electrónica. En la actualidad este dispositivo se puede usar como un ordenador doméstico para ver películas, crear documentos, navegar por Internet, etcétera.

Plug-in. Es un programa de ordenador que interactúa con otro programa para aportarle una función o utilidad específica, generalmente muy específica. Los plug-ins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar / decodificar e-mails, filtrar imágenes de programas gráficos, etcétera.

PocketPC. Es un ordenador de bolsillo, también llamado ***PDA (Personal Digital Assistant)***. Se trata de un pequeño ordenador, diseñado para ocupar el mínimo espacio y ser fácilmente transportable que ejecuta el sistema operativo Windows CE de Microsoft, el cual le proporciona capacidades similares a los PCs de escritorio.

RBDMS. Es un Sistema Administrador de Bases de Datos Relacionales. RDBMS viene del acrónimo en ingles ***Relational Data Base Manager System***.

Script. En programación es un programa o una secuencia de instrucciones que es interpretado y llevado a cabo por otro programa en lugar de ser procesado por el procesador de la computadora.

UDDI. Son las siglas del catálogo de negocios de Internet denominado ***Universal Description, Discovery, and Integration***. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web.

UNIX. Es un sistema operativo portable, multitarea y multiusuario; desarrollado en principio por un grupo de empleados de los laboratorios ***Bell de AT&T***, entre los que figuran *Ken Thompson, Dennis Ritchie y Douglas McIlroy*.

URL. Acrónimo de ***Universal Resource Locator (Localizador Universal de Recursos /Identificador Universal de Recursos)***. Sistema unificado de identificación de recursos en la red. Es el modo estándar de proporcionar la dirección de cualquier recurso en Internet.

Username. Nombre inteligible que identifica al usuario de un sistema o de una red.

Wireless. Termino referido a comunicaciones inalámbricas, en las que no se utiliza un medio de propagación físico, sino la modulación de ondas electromagnéticas, radiaciones o medios ópticos. Estás se propagan por el espacio vacío sin medio físico que comunique cada uno de los extremos de la transmisión.

XML. Es el acrónimo del inglés ***eXtensible Markup Language (lenguaje de marcado ampliable o extensible)*** desarrollado por el *World Wide Web Consortium (W3C),* diseñado con la intención de reemplazar al estándar HTML.

Referencias Bibliográficas

[1] Mridula Parihar et al. ***ASP .NET Bible***.
Hungry Minds, Inc. Copyright 2002.

[2] Jason Butler y Tony Caudill. ***ASP .NET Data Base Programming Weekend Crash Course.***
Hungry Minds, Inc. Copyright 2002.

[3] John Alexander y Billy Hollis. ***Developing Web Applications with Visual Basic .NET and ASP .NET.***
Wiley Computer Publishing John Wiley & Sons, Inc. Copyright 2002.

[4] Ángel Esteban Núñez. ***Desarrollo de Aplicaciones para Internet con ASP .NET.*** *Grupo EIDOS Consultoria y Documentación Informática Copyright 2002.*

[5] Douglas J. Reilly. ***Designing Microsoft ASP .NET Applications***.
Microsoft Prees. Copyright 2002.

[6] Danny Ryan y Tommy Ryan. ***ASP .NET: Your visual blueprint for creating Web applications on the .NET Framework.***
Hungry Minds, Inc. Copyright 2002.

[7] Mossenbock Hanspeter. ***Dynamic Web Pages with ASP .NET.***
Microsoft Copyright 2002.

[8] J.D. Meier, Alex Mackman et al. ***Modelo de Seguridad para aplicaciones ASP .NET.*** *Wolly.Net.*

[9] D. Rubiolo; J. Meier. Et al. ***¿Qué es Microsoft .NET?***
Microsoft Corporation. Copyright 2001.

Referencias electrónicas:

Alejandro Gassmann***. Introducción a ASP .NET.***

URL: *http://www.gamarod.com.ar/articulos/articulos.asp?id=134*

[consulta: 26 septiembre 2005]

Microsoft Corporation. 2001. ***Tutorial de ASP .NET.***

URL: *http://es.gotdotnet.com/quickstart/aspplus/doc/quickstart.aspx*

[consulta: 29 septiembre 2005]

Microsoft Corporation. 2003. ***Recursos ASP .NET [en línea]****.*

URL: *http://microsoft/spanish/msdn/matrix/*

[consulta: 17 junio 2006]

Microsoft Corporation. 2006. ***Visual Studio .NET****.*

URL: *http://msdn2.microsoft.com/es-es/library/88fx1xy0.aspx*

[consulta: 7 julio 2006]