

## Prácticas

### Tema 5 – Técnicas de acceso a datos

<b>Objetivos</b>	<p>En esta práctica, se aborda la realización de tareas relacionadas con:</p> <ul style="list-style-type: none"><li>• Agregar y conectar un archivo de base de datos de SQL Server a un Proyecto de Aplicación Web de ASP.NET.</li><li>• Utilizar los modelos de acceso a datos de ASP.NET para desarrollar interfaces adecuadas a los requerimientos de una Aplicación Web.</li><li>• Utilizar el modelo de acceso a datos declarativo para la presentación e interacción con la información almacenada en la base de datos subyacente de la Aplicación Web, empleando controles de datos de ASP.NET sobre interfaz Web.</li><li>• Utilizar el modelo de acceso a datos mediante código para el acceso y la manipulación de la información almacenada en una base de datos, empleando sentencias SQL incrustadas en el código lógico a través de objetos de ADO.NET (<i>ActiveX Data Objects .NET</i>).</li><li>• Incluir controles simples de ASP.NET en la interfaz de los Web Forms para presentar los datos almacenados en la base de datos</li><li>• Conocer el funcionamiento general de los sistemas de autenticación de usuarios para el acceso a una aplicación para la Web y para el acceso a los datos almacenados en la base de datos subyacente.</li><li>• Emplear variables de sesión para compartir datos entre los Web Forms de la aplicación Web durante el período de la sesión de usuario.</li></ul>
<b>Conocimientos previos</b>	<p>Para realizar esta práctica, es necesario tener conocimientos sobre:</p> <ul style="list-style-type: none"><li>• Bases de datos relacionales.</li><li>• Utilización de controles de servidor en páginas Web de ASP.NET.</li><li>• Programación de código lógico ejecutado en el servidor.</li><li>• Lenguaje de programación Microsoft Visual C#</li><li>• Lenguaje SQL para la manipulación, definición y control de datos.</li></ul>
<b>Escenario</b>	<p>Se presentan varios ejercicios dirigidos a comprender diversos aspectos prácticos sobre el desarrollo de las aplicaciones Web interactivas con acceso a datos mediante ASP.NET Web Forms. Para la realización de esta práctica, se proporciona como recursos:</p> <ul style="list-style-type: none"><li>• El archivo de base de datos de SQL Server <i>Tienda.mdf</i>.</li><li>• Dos páginas maestras, denominadas como <i>MasterPage.Master</i> y <i>MasterPageAdm.Master</i>, que especifican la apariencia visual y el comportamiento común.</li><li>• Un archivo de hoja de estilo, denominado <i>HojaEstilo.css</i>, que define las características de estilo que se utilizan la página maestra anterior.</li></ul>

## Ejercicio 1

### Creación y preparación de una Aplicación Web

En este primer ejercicio se creará una nueva Solución de Visual Studio, denominada *Tienda*, y un nuevo Proyecto de Aplicación Web de ASP.NET, denominado *GesTienda*. Esta nueva Aplicación Web servirá para desarrollar los ejercicios contenidos en las prácticas de este tema. A continuación, se agregará al Proyecto el archivo de base de datos de SQL Server *Tienda.mdf* que almacena la información que manejará la Aplicación Web y que, por tanto, es el almacén de datos subyacente de la Aplicación Web. Finalmente, se agregará una página maestra que permitirá establecer una apariencia visual común de los Web Forms que compondrán la aplicación Web.

Previamente, antes de abordar las tareas propias de este ejercicio, es necesario realizar una serie de tareas de preparación y configuración de SQL Server. Estas tareas previas consisten, principalmente, en la creación de una instancia de **SQL Server LocalDB**. Esta instancia va a ser utilizada en todas las prácticas del curso para acceder a los datos almacenados en las bases de datos que se utilicen.

#### Creación de una instancia de SQL Server LocalDB

SQL Server LocalDB es una versión ligera del motor de base de datos de SQL Server Express que se integra con Visual Studio. La utilización de SQL Server LocalDB se destina, principalmente, al desarrollo de software empleando el Entorno Integrado de Desarrollo (IDE) de Visual Studio. Las principales características de SQL Server LocalDB son las siguientes:

- Se instala, de forma predeterminada, al instalar Visual Studio.
- Se inicia a petición desde Visual Studio para poder realizar las pruebas de ejecución y de acceso a datos, mediante una cadena de conexión especificada en el código.
- El acceso a SQL Server LocalDB se realiza mediante el componente *SQL Server Data Tools* (SSDT) que está integrado en el entorno de desarrollo de Visual Studio.
- Se ejecuta en modo de usuario, sin necesidad de una configuración compleja.
- De forma predeterminada, utiliza archivos de base de datos de SQL Server, cuya extensión de nombre es \*.mdf, para almacenar los datos.

Es necesario tener en cuenta que cada versión de Visual Studio utiliza una versión concreta de SQL Server LocalDB. Así, el entorno de desarrollo **Visual Studio 2022** utiliza, de manera predeterminada, el servidor de base de datos **SQL Server 2019 LocalDB (Versión 15)** que se instala junto con él.

En primer lugar, se va a comprobar que SQL Server LocalDB se ha instalado cuando se instaló **Visual Studio 2022**. Para ello, acceder a la **Configuración de Windows** y, a continuación, acceder a la utilidad **Aplicaciones y Características** para poder visualizar la lista de aplicaciones instaladas en el sistema. En esta lista buscar **Microsoft SQL Server 2019 LocalDB** para confirmar que se dispone de la versión de SQL Server LocalDB adecuada para el desarrollo de las prácticas. Si no aparece en la lista, deberá instalarse desde las opciones correspondientes del programa de instalación de Visual Studio. Si se ha tenido instalada una versión anterior de Visual Studio es posible que, además, se encuentren instaladas otras versiones anteriores de SQL Server LocalDB.

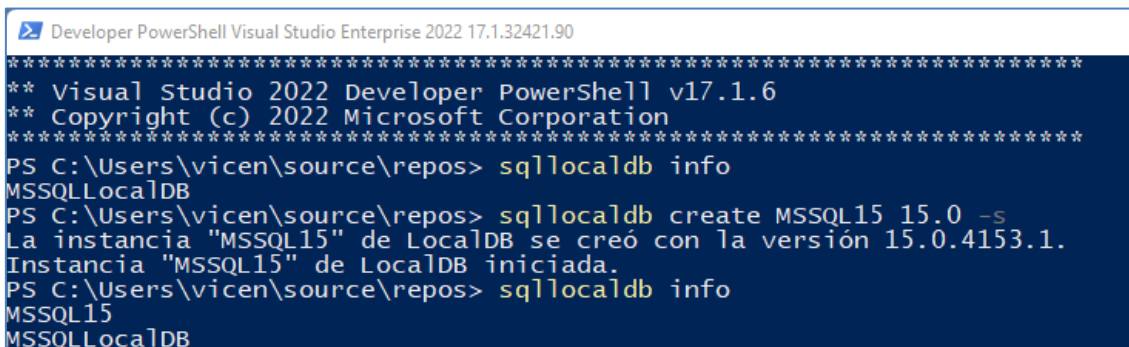
A continuación, se va a crear una nueva instancia de **SQL Server 2019 LocalDB (versión 15.0)** para poder utilizarla con **Visual Studio 2022**. El nombre de la nueva instancia de SQL Server, que se utilizará a lo largo del curso para realizar las prácticas, será: **(localdb)\MSSQL15**. Para ello, hacer:

1. En primer lugar, se va a crear una nueva instancia de SQL Server 2019 LocalDB (Versión 15.0) a través de la línea de comandos de **Powershell**, realizando las siguientes acciones:
  - a. Iniciar Visual Studio. No es necesario cargar ningún proyecto.
  - b. Abrir un terminal de línea de comandos de **PowerShell** de **Windows**. Para ello, desplegar la opción **Línea de comandos** del menú **Herramientas** de Visual Studio y, a continuación, seleccionar la opción **PowerShell para desarrolladores**.
  - c. En la línea de comandos, ejecutar el siguiente comando para crear e iniciar la instancia de SQL Server LocalDB de la versión 15 que se va a utilizar a lo largo del curso y cuyo nombre va a ser **MSSQL15**.

```
> sqllocaldb create MSSQL15 15.0 -s
```

Para comprobar que se ha creado la instancia, ejecutar el siguiente comando.

```
> sqllocaldb info
```



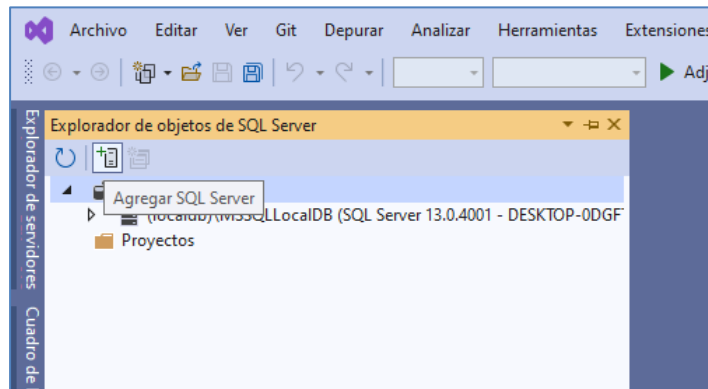
```
Developer PowerShell Visual Studio Enterprise 2022 17.1.32421.90
*****
** Visual Studio 2022 Developer PowerShell v17.1.6
** Copyright (c) 2022 Microsoft Corporation
*****
PS C:\Users\vicen\source\repos> sqllocaldb info
MSSQLLocalDB
PS C:\Users\vicen\source\repos> sqllocaldb create MSSQL15 15.0 -s
La instancia "MSSQL15" de LocalDB se creó con la versión 15.0.4153.1.
Instancia "MSSQL15" de LocalDB iniciada.
PS C:\Users\vicen\source\repos> sqllocaldb info
MSSQL15
MSSQLLocalDB
```

No importa que, además, existan otras instancias de SQL Server LocalDB.

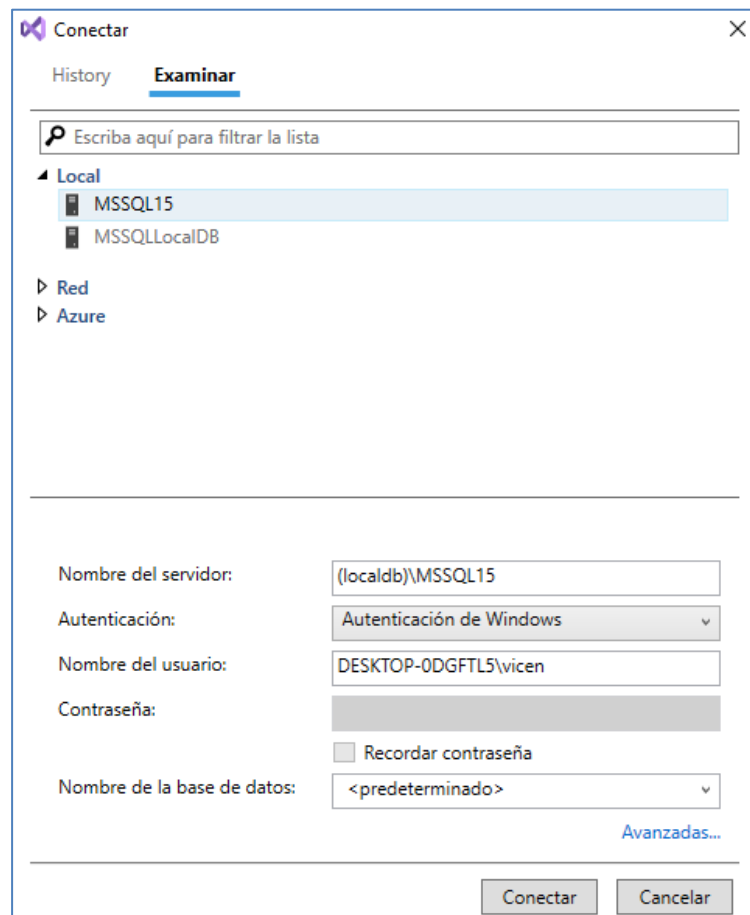
- d. Finalizar la sesión de línea de comandos de **PowerShell**, ejecutando el comando:

```
> exit
```

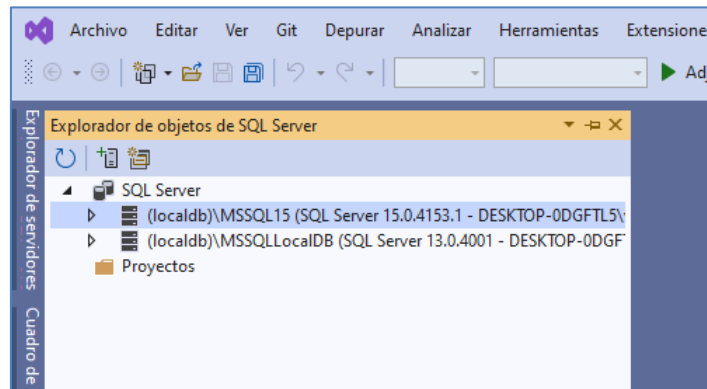
2. A continuación, se debe agregar a Visual Studio la instancia de SQL Sever **MSSQL15** que se ha creado anteriormente. Para ello, realizar las siguientes opciones:
  - a. En Visual Studio, acceder al **Explorador de Objetos de SQL Server** en el panel de la izquierda. En el caso de que no esté disponible esta opción, acceder al menú **Ver** y seleccionar la opción **Explorador de Objetos de SQL Server**.
  - b. En el **Explorador de Objetos de SQL Server**, se puede comprobar las instancias de SQL Server disponibles en Visual Studio. Para ello, desplegar la opción **SQL Server** haciendo clic sobre la punta de flecha que aparece a la izquierda de la opción **SQL Server**.
  - c. Para agregar la instancia **MSSQL15** a Visual Studio, hacer clic sobre el icono **Agregar SQL Server** de la barra de herramientas del **Explorador de Objetos de SQL Server**.



- d. En la ventana **Conectar**, desplegar la opción **Local**, seleccionar la instancia creada en el anteriormente denominada **MSSQL15** y hacer clic en **Conectar**.

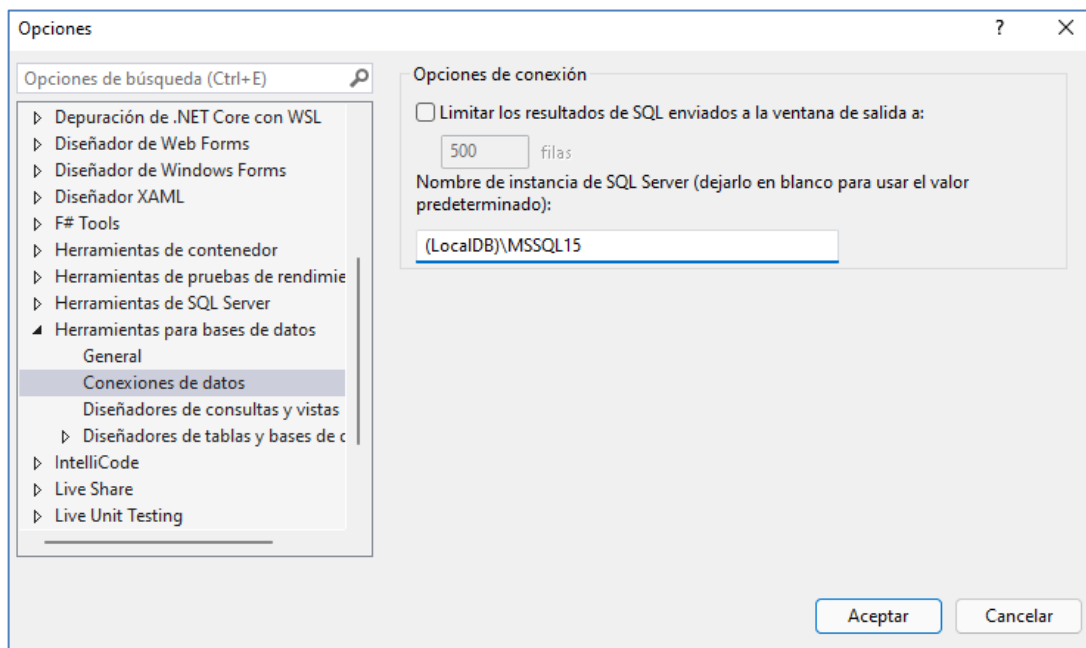


- e. Una vez realizadas estas tareas, se podrá comprobar que la nueva instancia que se ha creado, denominada **MSSQL15**, aparecerá en el **Explorador de Objetos de SQL Server** al desplegar la opción **SQL Server**.



También se podrá comprobar que esta instancia se corresponde con la versión 15 de SQL Server. La versión de SQL Server que utiliza cada instancia de SQL Server aparece entre paréntesis junto al nombre en el **Explorador de Objetos de SQL Server**. Es importante verificar que la instancia *(localdb)\MSSQL15* corresponde con **SQL Server 2019 LocalDB (Versión 15)**, porque es la versión que se va a utilizar con **Visual Studio 2022** para realizar todas las prácticas a lo largo del curso.

3. Para finalizar, se debe definir la instancia de SQL Server que se ha creado, *MSSQL15*, como instancia predeterminada para el acceso a datos en Visual Studio. Para ello, hacer:
  - a. En Visual Studio, acceder a la opción **Opciones...** del menú **Herramientas**.
  - b. En la ventana **Opciones**, desplegar la opción **Herramientas para bases de datos** y seleccionar la opción **Conexiones de datos**.
  - c. En la propiedad **Nombre de la instancia de SQL Server**, establecer el nombre de la instancia predeterminada de SQL Server que debe ser: *(LocalDB)\MSSQL15*.
  - d. Hacer clic en **Aceptar**.



Una vez creada la instancia de base de datos de SQL Server LocalDB, denominada MSQL15, que se va a utilizar en todas las prácticas del curso para poder acceder a los datos almacenados en las bases de datos, se van a realizar las tareas del ejercicio. En este ejercicio se va a crear una nueva Solución de Visual Studio, denominada *Tienda*, y un Proyecto de Aplicación Web de ASP.NET (.NET Framework), denominado *GesTienda*. A continuación, se agregará el archivo de base de datos de SQL Server *Tienda.mdf* al Proyecto, que almacenará la información que maneja la nueva Aplicación Web. Y, finalmente, se agregará una página maestra al proyecto, que permitirá establecer una apariencia visual común para los Web Forms que compondrán la aplicación Web.

### Crear una nueva Solución y un nuevo Proyecto de Aplicación Web

En primer lugar, se creará una Solución de Visual Studio, denominada *Tienda*, y, formando parte de esta, un Proyecto de **Aplicación de Web de ASP.NET (.NET Framework)** vacía que utiliza el lenguaje C# denominado, *GesTienda*. Para crear una nueva Solución y un nuevo Proyecto vacío de Aplicación Web de ASP.NET pueden seguirse el procedimiento ya estudiado en las prácticas del primer tema.

### Agregar un archivo de base de datos de SQL Server a un Proyecto de Aplicación Web de ASP.NET

Una vez creada la Solución *Tienda* y la Aplicación Web de ASP.NET *GesTienda*, se va a agregar el archivo de base de datos de SQL Server *Tienda.mdf* que almacena la información que maneja la aplicación Web. Para ello, realizar las siguientes acciones:

1. Para agregar el archivo de base de datos de SQL Server *Tienda.mdf*, proporcionado por el profesor, a la Aplicación Web *GesTienda*, realizar las siguientes acciones:
  - a. Crear la carpeta *App\_Data* en la Aplicación Web. Para ello, acceder al **Explorador de soluciones** y hacer clic con el botón derecho sobre el Proyecto *GesTienda*. A continuación, desplegar la opción **Agregar**, seleccionar la opción **Agregar carpeta ASP.NET** y, finalmente, seleccionar la opción *App\_Data*.
  - b. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre la carpeta *App\_Data*, desplegar la opción **Agregar** y seleccionar la opción **Elemento existente...**
  - c. En el cuadro de diálogo **Agregar elemento existente**, seleccionar el archivo de base de datos *Tienda.mdf* desde su ubicación y hacer clic en **Agregar**. El archivo de base de datos de SQL Server *Tienda.mdf* aparecerá añadido a la Aplicación Web contenido en la carpeta *App\_Data*.
2. Para comprobar que se ha creado la conexión de datos de la Aplicación Web *GesTienda* con el archivo de base de datos de SQL Server *Tienda.mdf*, realizar las siguientes acciones:
  - a. En Visual Studio, acceder al **Explorador de servidores** en el panel de la izquierda. En el caso de que no esté disponible esta opción, acceder al menú **Ver** y seleccionar la opción **Explorador de servidores**.
  - b. En el **Explorador de servidores**, desplegar la opción **Conexiones de datos**, haciendo clic sobre la fecha que aparece a su izquierda.
  - c. Comprobar que se ha creado una conexión de datos cuyo nombre será *Tienda.mdf*.
  - d. Desplegar los objetos de la base de datos *Tienda.mdf*, haciendo clic sobre la flecha situada a la izquierda del nombre de la conexión de datos. Para visualizar las tablas, puede ser necesario hacer clic sobre el botón **Actualizar** situado en la parte superior.



3. Finalmente, conviene comprobar que funciona correctamente la conexión de datos con el archivo de base de datos de SQL Server *Tienda.mdf* desde la Aplicación Web *GesTienda*. Para ello, realizar las siguientes acciones:
  - a. En el **Explorador de Servidores**, hacer clic con el botón derecho sobre el nombre de la conexión de datos y seleccionar **Modificar conexión...**
  - b. Hacer clic sobre el botón **Probar conexión**. Aparecerá una ventana informando sobre si el funcionamiento de la conexión es o no correcto. Si fuera necesario, hacer clic en el botón **Actualizar** de la barra de herramientas del **Explorador de Servidores**.
  - c. En el **Explorador de Servidores**, también es posible seleccionar **Abrir definición de tabla y Mostrar datos de tabla** haciendo clic en el botón derecho sobre cada tabla.

Conviene visualizar y analizar la definición de las tablas contenidas en la base de datos *Tienda.mdf* para poder comprender la estructura de tablas, campos, claves principales, claves ajenas, etc. de la base de datos que utiliza la Aplicación Web.

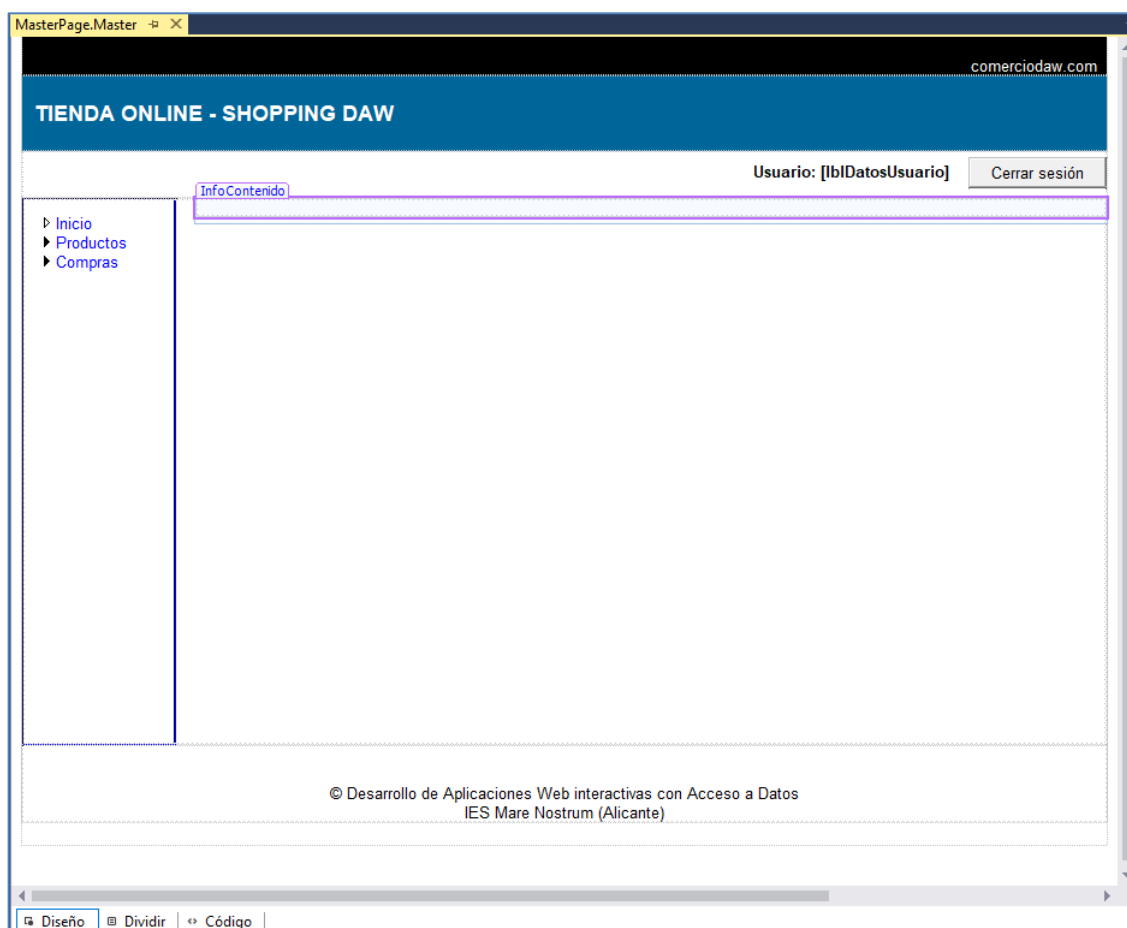
### Agregar una página maestra a un Proyecto de Aplicación Web

Se ha preparado una página maestra, denominada *MasterPage.Master* y el correspondiente archivo de hoja de estilo externa *HojaEstilo.css*, para mantener una apariencia común en los diferentes Web Forms que formarán la aplicación Web *GesTienda*. En general, la mayoría de los Web Forms que se desarrollen en esta Aplicación Web se crearán vinculándolos a esta página maestra que actuará como base o plantilla de diseño visual. Además, en algunos ejercicios de esta práctica se irá incorporando código lógico a esta página maestra para definir un comportamiento común de los Web Forms vinculados a ella, en relación con la resolución de determinados aspectos de la Aplicación Web.

A continuación, se describe cómo agregar la página maestra *MasterPage.Master* y el archivo de hoja de estilo externa *HojaEstilo.css* a la Aplicación Web *GesTienda*.

1. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre de la Aplicación Web *GesTienda*, desplegar la opción **Agregar** en el menú emergente y seleccionar la opción **Elemento existente...**
2. En la ventana **Agregar elemento existente**, seleccionar el archivo *MasterPage.Master* desde su ubicación y hacer clic en **Agregar**. El archivo aparecerá añadido al Proyecto de Aplicación Web *GesTienda*.
3. A continuación, se va a agregar la hoja de estilos externa denominada *HojaEstilo.css* que utiliza la página maestra *MasterPage.Master*. Para ello, en el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre de la Aplicación Web *GesTienda*, desplegar la opción **Agregar**, seleccionar la opción **Nueva carpeta** y modificar el nombre de la nueva carpeta para denominarla *Estilos*.
4. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre la carpeta *Estilos*, desplegar la opción **Agregar** y seleccionar la opción **Elemento existente...**
5. En el cuadro de diálogo **Agregar elemento existente**, seleccionar el archivo *HojaEstilo.css* desde su ubicación y hacer clic en **Agregar**. El archivo aparecerá añadido a la carpeta *Estilos* de la Aplicación Web *GesTienda*. Mediante el **Explorador de archivos** de **Windows**, se puede comprobar que se habrá hecho una copia de todos los archivos agregados al Proyecto.

6. Desde el **Explorador de soluciones** de Visual Studio, abrir la vista **Diseño** de la página maestra *MasterPage.Master* para comprobar que el resultado de la presentación visual es el correcto y que la hoja de estilos externa se ha vinculado perfectamente. La asociación entre los Web Form que se vayan creando y la página maestra se irá realizando posteriormente, cuando se vayan creando los Web Forms que se desarrollan en cada uno de los ejercicios.



Si fuera necesario, porque se ha preferido situar el archivo de hoja de estilo externa en otra carpeta o por cualquier otro motivo, la página maestra y la hoja de estilo externa pueden asociarse de la siguiente manera:

- Abrir la vista **Diseño** de la página maestra *MasterPage.Master*.
- Para asociar la hoja de estilo externa, basta con pinchar y arrastrar el archivo de hoja de estilo externa, denominado *HojaEstilo.css* desde el **Explorador de soluciones** hasta la vista **Diseño** de la página maestra a enlazar. Generalizando, se actuaría de una forma similar para enlazar un Web Form con una hoja de estilo externa.

Se recomienda abrir la vista **Código** de la página maestra *MasterPage.Master* y tratar identificar los elementos y controles incluidos en ella y comprender la composición de los elementos de HTML y controles de ASP.NET, así como la identificación y posición de los elementos de código reemplazable.



## Ejercicio 2

### Crear un Web Form para consulta de datos mediante controles de datos del modelo declarativo

Una vez creada y preparada la Aplicación Web *GesTienda*, dentro de la Solución *Tienda*, se creará un Web Form denominado *TiposVer.aspx* para consultar datos, empleando para ello un control de datos GridView para mostrar la información sobre los tipos de productos almacenados en la base de datos.

#### Crear un Web Form de consulta de datos empleando un control de datos GridView

1. Para crear el Web Form *TiposVer.aspx* vinculado con la página maestra *MasterPage.Master*, realizar las siguientes acciones:
  - a. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre de la Aplicación Web *GesTienda*, desplegar la opción **Agregar** en el menú contextual y seleccionar la opción **Nuevo elemento...**
  - b. En la ventana **Agregar nuevo elemento**, desplegar las opciones **Instalado**, **Visual C#** y **Web** el cuadro de la izquierda y seleccionar la opción **Web Forms**. En el cuadro central, seleccionar la opción **Formulario web con página maestra**, escribir *TiposVer.aspx* en el cuadro de texto **Nombre** y hacer clic en **Agregar**.
  - c. En la ventana **Agregar una página maestra**, seleccionar la página maestra denominada *MasterPage.Master* y hacer clic en **Aceptar**.
2. Abrir la vista **Código** del Web Form *TiposVer.aspx* para comprobar que incorpora los dos elementos de contenido reemplazable que están especificados en la página maestra. El primero, denominado *head*, permite agregar elementos de cabecera, como pueden ser las etiquetas `<style>`, `<link>`, etc. Y el segundo, denominado *InfoContenido*, se utilizará para incluir el contenido específico del Web Form. A continuación, agregar el siguiente código de marcado que aparece incluido en el elemento `<asp:Content>` denominado *InfoContenido*.

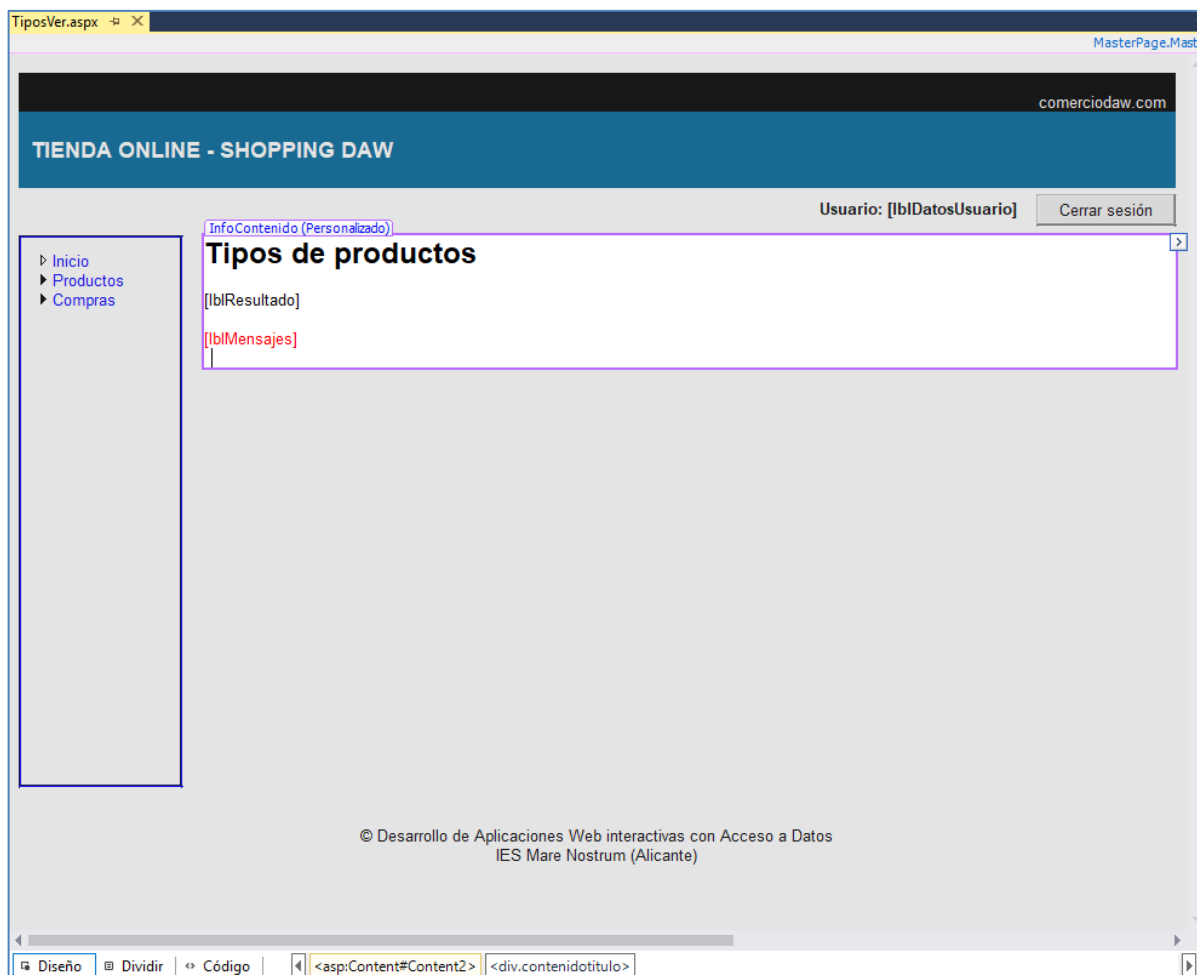
```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true"
CodeBehind="TiposVer.aspx.cs" Inherits="GesTienda.TiposVer" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="InfoContenido" runat="server">
  <div class="contenidotitulo">Tipos de productos</div>
  <br />
  <div>
    <asp:Label ID="lblResultado" runat="server"></asp:Label>
  </div>
  <br />
  <asp:Label ID="lblMensajes" ForeColor="red" runat="server"></asp:Label>
  <br />
  <br />
</asp:Content>
```

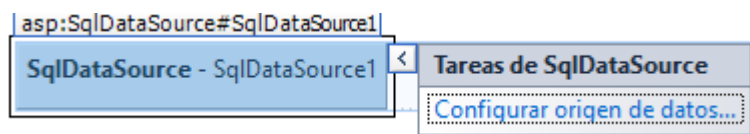
Como puede apreciarse en el código anterior, en la página de contenido se ha incluido: un título, la etiqueta *lblResultado* y la etiqueta *lblMensajes*.

Una vez que se incluya el código anterior el aspecto visual de la vista **Diseño** del Web Form *TiposVer.aspx* será similar al siguiente.



En la ilustración anterior pueden apreciarse los elementos y controles que forman el elemento de contenido reemplazable, denominado *InfoContenido*. En general, la mayoría de los Web Forms de la Aplicación Web *GesTienda* incluirán algunos de estos tres elementos en su contenido: el elemento de título del Web Form, el control Label *lblResultado* para mostrar el resultado del procesamiento realizado mediante código lógico y el control Label *lblMensajes* para proporcionar información al usuario sobre el procesamiento realizado.

3. A continuación, se va a comenzar el procedimiento para poder añadir controles de datos al Web Form. Para ello, acceder a la vista **Diseño** el Web Form *TiposVer.aspx*.
4. Desde la ficha **Datos** del **Cuadro de herramientas**, agregar un control de origen datos de tipo *SqlDataSource*, denominado *SqlDataSource1*. Este control de datos enlaza el Web Form *TiposVer.aspx* con el archivo de base de datos *Tienda.mdf* y permite acceder a los datos almacenados en las tablas, ejecutar sentencias SQL sobre la base de datos que maneja la aplicación Web, ejecutar vistas sobre la base de datos, etc.



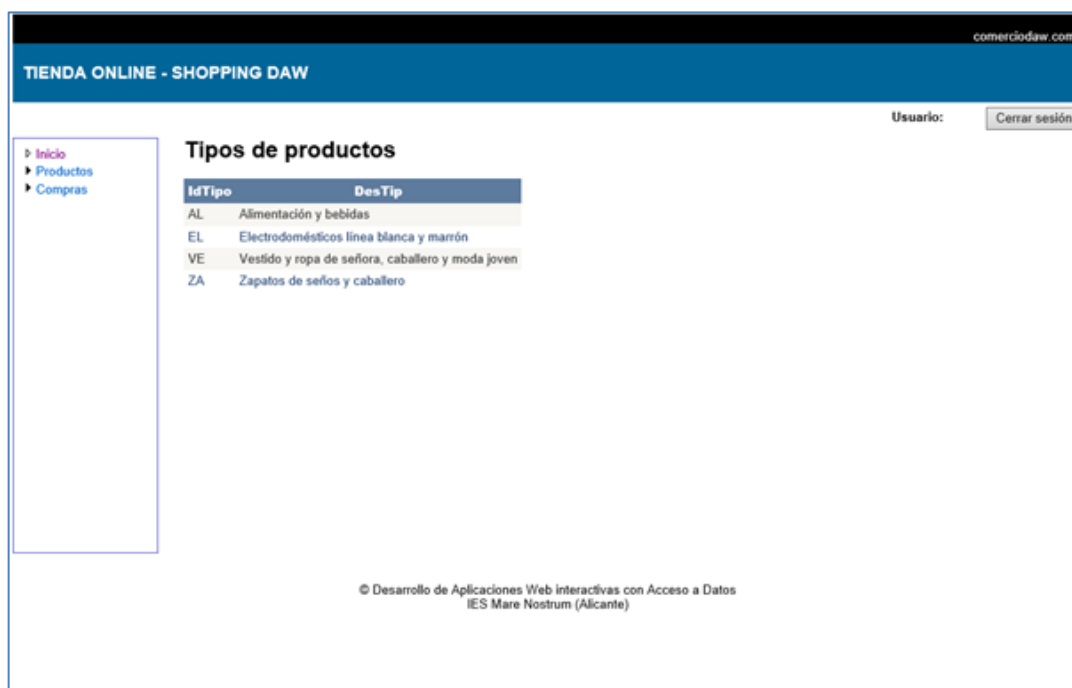
5. En la vista **Diseño** del Web Form, seleccionar el control *SqlDataSource1*, hacer clic sobre la flecha que aparece en la esquina superior derecha para acceder a las **Tareas de SqlDataSource** y seleccionar la opción **Configurar origen de datos...** A continuación, realizar las siguientes acciones en el asistente para configurar origen de datos del control *SqlDataSource*:
  - a. Desplegar las opciones disponibles en el cuadro combinado que permite especificar la conexión de datos a utilizar y seleccionar la conexión de datos *Tienda.mdf*.
  - b. Marcar la casilla de verificación **Cadena de conexión** y hacer clic en **Siguiente**.
  - c. A continuación, dado que es la primera vez que se emplea esta conexión de datos en la Aplicación Web, se pregunta si deseamos guardar la definición de la cadena de conexión en el archivo de configuración *Web.config*. Comprobar que está marcada la casilla de verificación **Sí, guardar esta conexión como:**, aceptar el nombre por defecto *ConnectionString* para referirse a la conexión y hacer clic en **Siguiente**.
  - d. Seleccionar la opción **Especificar columnas de una tabla o vista**, seleccionar la tabla *TIPO* en el cuadro combinado correspondiente, comprobar que están seleccionadas todas las columnas de la tabla y hacer clic en **Siguiente**.
  - e. Puede comprobarse el resultado de la consulta, para ello hacer clic sobre el botón **Consulta de prueba**. Si el resultado es el esperado hacer clic en el botón **Finalizar**.

En este punto, puede comprobarse que se habrá modificado el archivo de configuración de la Aplicación Web, denominado *Web.config*. Ahora incluye el elemento `<connectionStrings>` que define las características de la conexión de datos a utilizar por la Aplicación Web.

```
<connectionStrings>
  <add name="ConnectionString" connectionString="Data Source=(LocalDB)\MSSQL15;
    AttachDbFilename=|DataDirectory|\Tienda.mdf;Integrated Security=True"
    providerName="System.Data.SqlClient"/>
</connectionStrings>
```

En el código anterior puede apreciarse que la conexión denominada *ConnectionString* utiliza la instancia *MSSQL15* y referencia al archivo de base de datos de SQL Server *Tienda.mdf*.

6. A continuación, se va a añadir al Web Form un control enlazado a datos de tipo *GridView*. Este control de datos permitirá mostrar sobre la interfaz web la información recuperada mediante el control *SqlDataSource* añadido anteriormente. Para ello, estando en la vista **Diseño** del Web Form, añadir un control de datos de tipo *GridView* desde la ficha **Datos** del **Cuadro de herramientas** y realizar las siguientes acciones:
  - a. En la ventana de **Propiedades**, modificar su propiedad *Id* para denominarlo *grdTipos*.
  - b. Acceder a las **Tareas de GridView**, para asignarle como origen de datos el nombre del control *SqlDataSource* añadido y que, probablemente, su *Id* sea *SqlDataSource1*.
  - c. En las **Tareas de GridView**, habilitar la paginación y acceder a la opción **Formato automático...** para seleccionar el esquema Profesional.
7. Iniciar la depuración del Web Form *TiposVer.aspx* para comprobar el resultado.

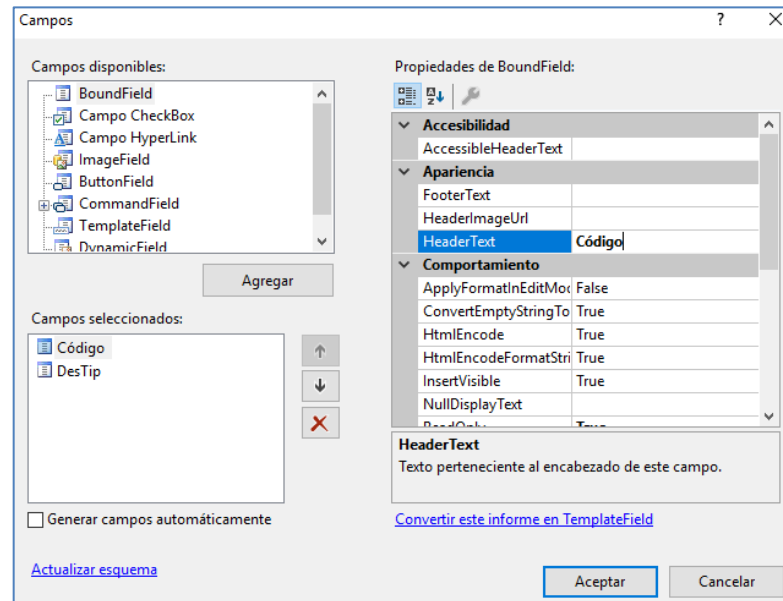


A continuación, se va a mejorar la presentación del control GridView, a través de las opciones incorporadas en las **Tareas de GridView** y ajustando el valor de algunas de sus propiedades. Se recomienda iniciar la depuración al final de cada cambio realizado durante el proceso, para poder ir asimilando los efectos producidos en la visualización de los cambios realizados.

8. Acceder a las **Tareas de GridView**, y seleccionar la opción **Actualizar esquema** para ajustar la vista de diseño del GridView a la estructura de los datos recuperados. Si se utiliza alguna de las últimas versiones de Visual Studio esta acción no producirá efecto, porque este ajuste se produce de forma predeterminada en el momento de enlazar con el control SqlDataSource.
9. Seleccionar el control de datos GridView *grdTipos* y en la ventana de **Propiedades** ajustar las propiedades siguientes para conseguir los efectos que se describen en la siguiente tabla.

Descripción	Grupo de propiedades	Propiedad	Valor
Centrado en el Web Form	Diseño	<b>HorizontalAlign</b>	Center
Ajustar la anchura	Diseño	<b>Width</b>	50%
Opciones de Paginación	Paginación	<b>AllowPaging</b>	True
	Paginación	<b>PageSize</b>	5
	Paginación → PagerSettings	<b>Mode</b>	NextPreviousFirstLast
	Paginación → PagerSettings	<b>FirstPageText</b>	Primero
	Paginación → PagerSettings	<b>LastPageText</b>	Último
	Paginación → PagerSettings	<b>NextPageText</b>	Siguiente
	Paginación → PagerSettings	<b>PreviuosPageText</b>	Anterior

10. En las **Tareas de GridView**, seleccionar la opción **Editar columnas...** y modificar el texto del encabezado de cada columna del GridView. Para ello, seleccionar cada campo en el cuadro **Campos seleccionados** y acceder a la propiedad *HeaderText* del grupo de propiedades *Apariencia*, para introducir el nombre deseado para cada columna: Código y Descripción.



11. Iniciar la depuración del Web Form *TiposVer.aspx* para comprobar los resultados obtenidos finalmente. Por razones de claridad en la visualización del resultado, en la siguiente ilustración se muestra solo la parte significativa de la interfaz correspondiente a la resolución del ejercicio.

Tipos de productos	
Código	Descripción
AL	Alimentación y bebidas
EC	Electrónica
EL	Electrodomésticos línea blanca y marrón
MD	Material deportivo
RD	Ropa deportiva
<a href="#">Siguiente</a> <a href="#">Último</a>	

12. Abrir la vista **Diseño** del archivo de página maestra *MasterPage.Master*. En el control TreeView incluido en el elemento `<div>` denominado *menu*, acceder a las **Tareas de TreeView** para editar sus nodos y modificar la propiedad *NavigateUrl* asociada a la opción de segundo nivel denominada *Tipos de productos*, que está incluida en la opción de primer nivel *Productos*, para facilitar el acceso al Web Form *TiposVer.aspx*.
13. Iniciar la depuración del Web Form *TiposVer.aspx* para comprobar los resultados obtenidos y comprobar el acceso al Web Form desde la opción correspondiente del menú.

## Ejercicio 3

### Enlazar controles de datos entre sí

En este ejercicio se muestra cómo enlazar controles de datos entre sí para presentar visualmente los datos de tablas relacionadas.

#### Crear un Web Form Maestro-Detalle enlazando controles de datos entre sí

En muchas ocasiones, es necesario mostrar o mantener datos de tablas relacionadas. A continuación, se creará un nuevo Web Form para mostrar los datos de los productos que pertenecen a cada tipo de productos. Ello supone mostrar los datos de la tabla *TIPO*, que actuará como tabla maestra, y de la tabla *PRODUCTO*, que actuará como tabla de detalle. Si los datos de cada una de las tablas se muestran en controles de datos diferentes, por ejemplo, en dos controles GridView, es necesario enlazar ambos entre sí para mantener la correspondencia entre los datos relacionados. Así, los datos de detalle de los productos a mostrar en cada momento, almacenados en la tabla *PRODUCTO*, deberán corresponderse con los datos de la tabla *TIPO* que se estén mostrando. Este tipo de configuraciones de interfaz para la presentación de datos de tablas relacionadas, se denomina Maestro-Detalle o Cabecera-Líneas.

1. Crear un nuevo Web Form, denominado *ProductosPorTipoVer.aspx*, que quede vinculado con la página maestra *MasterPage.Master*, tal como se ha realizado en el ejercicio anterior.
2. En el Web Form *ProductosPorTipoVer.aspx*, añadir un control de datos *SqlDataSource*, denominado *SqlDataSource1*, y un control de datos GridView, denominado *grdTipos*, para mostrar los datos correspondientes a los registros principales de la tabla maestra: *TIPO*. La presentación visual de este GridView puede quedar como muestra la siguiente ilustración.

Productos por tipo	
Tipos de productos	
Código	Descripción
AL	Alimentación y bebidas
EC	Electrónica
EL	Electrodomésticos línea blanca y marrón
MD	Material deportivo
RD	Ropa deportiva
<a href="#">Siguiente</a> <a href="#">Último</a>	

3. En este mismo Web Form, añadir otro control de datos *SqlDataSource*, denominado *SqlDataSource2*, y otro control de datos GridView, denominado *grdProductos*, que se utilizarán para mostrar los datos correspondientes a la tabla de detalle: *PRODUCTO*. La presentación visual correspondiente del Web Form puede quedar como muestra la siguiente ilustración.

## Productos por tipo

### Tipos de productos

Código	Descripción
AL	Alimentación y bebidas
EC	Electrónica
EL	Electrodomésticos línea blanca y marrón
MD	Material deportivo
RD	Ropa deportiva
<a href="#">Siguiente</a> <a href="#">Último</a>	

### Productos

Id Producto	Descripción	Precio	Unidad	Tipo
0032-895	Patatas	2,0000	Kilogramo	AL
0231-227	Pantalones	25,3000	Unidad	VE
0290-456	Melones	4,0000	Kilogramo	AL
1000-100	Camisa	15,6000	Unidad	VE
2348-312	TV LG LCD 36"	384,0000	Unidad	EL
2906-126	Aceite	5,0000	Litro	AL
4398-119	Zapatos señora Mod. Niza	45,7000	Par	ZA
4500-450	Zapatos caballero Mod. Ibiza	30,0000	Par	ZA
4502-341	Corbata	9,5000	Unidad	VE
5578-784	TV Panasonic LCD 42"	749,0000	Unidad	EL
<a href="#">Siguiente</a> <a href="#">Último</a>				

4. Para enlazar los datos entre ambos controles GridView, de modo que los datos de detalle se correspondan con una fila seleccionada en el GridView de los datos maestros, hacer:
  - a. En las **Tareas de GridView** que muestra los datos maestros sobre los tipos de productos, seleccionar la opción **Habilitar selección**.
  - b. En las **Tareas de SqlDataSource**, denominado *SqlDataSource2*, que está asociado al GridView que muestra los datos de detalle, *grdProductos*, seleccionar la opción **Configurar origen de datos...**, avanzar en el asistente hasta la ventana **Configurar la instrucción Select** y hacer clic en el botón **WHERE** para agregar condiciones con parámetros que obtienen sus valores en tiempo de ejecución. En el cuadro de diálogo **Agregar Cláusula WHERE**, introducir los siguientes valores:
    - Columna: *IdTipo* (Debe ser la clave principal)
    - Operador: =
    - Origen: *Control*
    - Id. de control: *grdTpos* (Enlace al control maestro)



Agregar cláusula WHERE

Agregue una o varias condiciones a la cláusula WHERE para la instrucción. Para cada condición, puede especificar un valor literal o un valor con parámetros. Los valores con parámetros obtienen sus valores en tiempo de ejecución según sus propiedades.

Columna:   
 Operador:   
 Origen:   
 Expresión SQL:   
 Cláusula WHERE:

Propiedades del parámetro  
 Id. de control:   
 Valor predeterminado:   
 Valor:

- c. En la ventana **Agregar cláusula WHERE**, hacer clic en **Agregar** y después en **Aceptar**.
- d. En la ventana **Configurar la instrucción Select**, hacer clic en **Siguiente** y en **Finalizar**.
5. Iniciar la depuración para comprobar que ambos controles GridView han resultado enlazados entre sí, de modo que los datos de los productos que se muestran en el control GridView detalle se corresponden con el tipo de productos seleccionado en el control GridView maestro.

comerciodaw.com

**TIENDA ONLINE - SHOPPING DAW**

Usuario:

**Productos por tipo**

**Tipos de productos**

	Código	Descripción
<a href="#">Seleccionar</a>	AL	Alimentación y bebidas
<a href="#">Seleccionar</a>	EC	Electrónica
<a href="#">Seleccionar</a>	EL	Electrodomésticos línea blanca y marrón
<a href="#">Seleccionar</a>	MD	Material deportivo
<a href="#">Seleccionar</a>	RD	Ropa deportiva
	<a href="#">Siguiente</a> <a href="#">Último</a>	

**Productos**

Id Producto	Descripción	Precio	Unidad	Tipo
2348-312	TV LG LCD 36"	384,0000	Unidad	EL
5578-784	TV Panasonic LCD 42"	749,0000	Unidad	EL
8898-009	Lavadora-Secadora AEG 8/5 Kg.	1199,0000	Unidad	EL

© Desarrollo de Aplicaciones Web interactivas con Acceso a Datos  
IES Mare Nostrum (Alicante)

### Formateo de los valores de los campos en el control de datos GridView

Los valores de los campos de tipo numérico o fecha que se muestran en un control de datos GridView deben formatearse para ajustarlo a las normas de visualización. Para ajustar el formato de datos de la columna Precio en el control GridView *grdProductos*, realizar las siguientes acciones:

1. Seleccionar el control de datos GridView denominado *grdProductos*.
2. En las **Tareas de GridView**, seleccionar la opción **Editar columnas...**
3. Seleccionar el campo **Precio** en el cuadro **Campos seleccionados** para modificar las siguientes propiedades:

Grupo de propiedades	Propiedad	Valor
Datos	DataFormatString	{0:n2}
Estilos → ItemStyle	HorizontalAlign	Right

4. Iniciar la depuración para comprobar los resultados obtenidos.

### Crear un Web Form Maestro-Detalle enlazando controles de datos entre sí, empleando vistas

El resultado anterior, siendo correcto, no suele ser satisfactorio porque es habitual que en cada GridView se muestren los valores de campos que forman parte de varias tablas. Por ejemplo, en el control GridView de detalle que muestra la información de los productos, podría mostrarse la descripción del tipo de producto en lugar del identificador del tipo de producto.

Para poder mostrar valores de diferentes tablas en un control de datos de tipo GridView es necesario utilizar una consulta como origen de datos del control GridView. Por este motivo, es habitual utilizar consultas, y no tablas, como orígenes de datos para los controles de datos. Sin embargo, no es posible enlazar controles de datos entre sí, cuando el origen de datos de un control SqlDataSource se basa en una instrucción SQL o un procedimiento almacenado, aunque sí que es posible cuando se utiliza una tabla o una vista. Una vista es una consulta definida en la propia base de datos y constituye el tipo de objeto más adecuado para enlazar controles de datos entre sí.

A continuación, se va a rehacer parte del ejercicio para poder emplear una vista como origen de datos del control GridView *grdProductos*. El resultado final se muestra en la ilustración que se encuentra al final del ejercicio. Para obtener ese resultado, realizar las siguientes acciones:

1. Crear y definir la vista a emplear como origen de datos del control GridView *grdProductos*, denominada *ProductosDet*. Para ello, realizar las siguientes acciones:
  - a. En el **Explorador de servidores** de Visual Studio, expandir los objetos de la conexión de datos, haciendo clic en la punta de flecha situado a la izquierda del nombre. En caso de que no aparezca la conexión de datos utilizada en el **Explorador de servidores**, hacer clic en botón derecho sobre la opción **Conexiones de datos** y seleccionar la opción **Actualizar**. Si continúa sin aparecer la conexión de datos utilizada, entonces en el **Explorador de soluciones** de Visual Studio, hacer doble clic sobre el nombre del archivo de base de datos *Tienda.mdf* situado en la carpeta *App\_Data* del Proyecto.

- b. En el **Explorador de servidores**, una vez que se han aparecido los elementos de la conexión de datos utilizada en el proyecto de aplicación Web, sobre el elemento **Vistas** hacer clic en botón derecho y seleccionar la opción **Agregar nueva Vista** para crear una vista que se asociará al origen de datos del control GridView *grdProductos* posteriormente. Esta vista incluirá los valores de los campos de la tabla *PRODUCTO* y el valor de la descripción del tipo de productos de la tabla *TIPO* de cada producto relacionado. Eliminar el código existente y, a continuación, incluir el siguiente código de la sentencia para crear la vista deseada:

```
CREATE VIEW [dbo].[ProductosDet]
AS SELECT  dbo.[PRODUCTO].IdProducto, dbo.[PRODUCTO].DesPro,
           dbo.[PRODUCTO].PrePro, dbo.[PRODUCTO].IdUnidad,
           dbo.[PRODUCTO].IdTipo, dbo.[TIPO].DesTip
FROM      dbo.[PRODUCTO] INNER JOIN
           dbo.[TIPO] ON  dbo.[PRODUCTO].IdTipo = dbo.[TIPO].IdTipo
```

En el código de la sentencia anterior para crear la vista, puede apreciarse que se define el nombre de la vista, que será: *ProductosDet*. Como puede observarse el nombre de la vista se especifica en la primera línea del código de la sentencia.

- c. Una vez introducido el código de la instrucción y antes de su ejecución para poder crear la vista, conviene evaluar la sintaxis del diseño de la vista. Para ello, hacer clic sobre el botón derecho sobre cualquier punto del área de diseño de la vista y seleccionar la opción **Analizar (Parse)**.
- d. Si la vista es correcta, hacer clic en el botón **Actualizar** y después en el botón **Actualizar base de datos** para agregarla a la base de datos y que pueda quedar disponible el uso de la vista *ProductosDet* una vez creada.
- e. La vista se habrá agregado a la base de datos, aunque puede ocurrir que no aparezca en el grupo de Vistas del **Explorador de servidores**. Para comprobar si se ha agregado, hacer clic en botón derecho sobre el nombre de la conexión de datos y seleccionar la opción **Actualizar**. Para comprobar su funcionamiento, seleccionar la vista a ejecutar, hacer clic en botón derecho y seleccionar **Mostrar resultados**. De la misma forma, se puede seleccionar **Abrir definición de vista** para modificar su diseño.
2. En las **Tareas de SqlDataSource**, denominado *SqlDataSource2*, que está asociado al GridView de detalle, *grdProductos*, seleccionar la opción **Configurar origen de datos...**, y hacer:
- Avanzar en el asistente hasta la ventana **Configurar la instrucción Select**.
  - Seleccionar la opción **Especificar columnas de una tabla o vista**, seleccionar la vista *ProductosDet* en el cuadro combinado correspondiente.
  - Marcar las casillas de verificación de todas las columnas de la vista, excepto la casilla correspondiente a columna *IdTipo*. Hacer clic en **Siguiente** y hacer clic en **Finalizar**.
3. Enlazar ambos controles GridView, tal como se ha realizado anteriormente. De este modo, los datos de detalle se correspondan con la fila de seleccionada en los datos maestros.
4. Iniciar la depuración para comprobar que el enlace de datos en los dos controles de datos GridView del Web Form se ha realizado correctamente, de manera que los datos de los productos que se muestran en el control GridView detalle se corresponden con el tipo de productos que se haya seleccionado en el control GridView maestro.

## Productos por tipo

### Tipos de productos

	Código	Descripción
<a href="#">Seleccionar</a>	AL	Alimentación y bebidas
<a href="#">Seleccionar</a>	EC	Electrónica
<a href="#">Seleccionar</a>	EL	Electrodomésticos línea blanca y marrón
<a href="#">Seleccionar</a>	MD	Material deportivo
<a href="#">Seleccionar</a>	RD	Ropa deportiva
<a href="#">Siguiente</a> <a href="#">Último</a>		

### Productos

Id Producto	Descripción	Precio	Unidad	Tipo
2348-312	TV LG LCD 36"	384,00	Unidad	Electrodomésticos línea blanca y marrón
5578-784	TV Panasonic LCD 42"	749,00	Unidad	Electrodomésticos línea blanca y marrón
8898-009	Lavadora-Secadora AEG 8/5 Kg.	1.199,00	Unidad	Electrodomésticos línea blanca y marrón

5. Abrir la vista **Diseño** del archivo de página maestra *MasterPage.Master*. En el control TreeView incluido en la página maestra, que se emplea para presentar el menú, acceder a las **Tareas de TreeView** para editar sus nodos y asociar la opción de menú de segundo nivel denominada *Productos por Tipo*, que estará incluida en la opción de primer nivel *Productos*, para facilitar el acceso al Web Form *ProductosPorTipoVer.aspx*.
6. Iniciar la depuración para comprobar los resultados obtenidos.

## Ejercicio 4

### Control de errores en tiempo de ejecución a nivel de aplicación

Cuando se accede a la información almacenada en una base de datos, es muy importante realizar un control de errores en tiempo de ejecución. Hay que tener en cuenta que necesariamente se van a producir errores en el acceso a los datos y, por tanto, debe establecerse un mecanismo para poder gestionarlos adecuadamente e informar al usuario del problema que se ha producido.

Los errores en tiempo de ejecución más importantes son los conocidos como **errores de manipulación de datos**. Este tipo de errores se refieren a los errores que se producen necesariamente debido al propio manejo de la información, aunque también pueden producirse debido al control de las restricciones establecidas en la definición y diseño de la base de datos. Los errores de manipulación de datos se producen, por ejemplo, cuando se intenta insertar los datos de un cliente que ya existe. O cuando se intenta eliminar la información de un producto que no existe. O cuando se intenta eliminar los datos de un producto que se ha vendido y que, por tanto, posee registros relacionados en la tabla *DETALLE*. O también, al intentar insertar un nuevo producto sin un valor en el campo Descripción que no admite valores nulos. Este tipo de errores es muy común que se produzcan durante la explotación de la aplicación por parte de los usuarios finales y, por tanto, los desarrolladores deben establecer los mecanismos adecuados de control para su gestión más adecuada.

Además de los errores de manipulación de datos, también pueden producirse otros tipos de errores en tiempo de ejecución como consecuencia de que no se encuentre el archivo de base de datos, que no esté disponible el servidor de base de datos, que la versión de la base de datos no sea adecuada, que se haya producido un error en el sistema de archivos, etc.

#### Comprobar los errores en tiempo de ejecución

A continuación, se producirá un error en tiempo de ejecución al intentar acceder a la base de datos, de manera que el depurador de Visual Studio informará sobre la naturaleza del error que se haya producido. Para producir un error en tiempo de ejecución, realizar las siguientes acciones

1. Abrir el archivo *Web.config* de la aplicación Web *GesTienda*.
2. Dentro del elemento `<connectionStrings>`, modificar el nombre del archivo de la base de datos en la línea correspondiente para hacer referencia al archivo "Tiend.mdf" que no existe, tal como se muestra a continuación.

```
<connectionStrings>
  <add name="ConnectionString" connectionString="Data Source=(LocalDB)\MSSQL15;
    AttachDbFilename=|DataDirectory|\Tiend.mdf;Integrated Security=True"
    providerName="System.Data.SqlClient"/>
</connectionStrings>
```

3. Iniciar la depuración del Web Form *ProductosPorTipoVer.aspx* para comprobar que se produce un error en tiempo de ejecución y que el depurador muestra una página informativa sobre el error producido similar a la siguiente.

### Error de servidor en la aplicación '/'.

*An attempt to attach an auto-named database for file C:\Users\vicente\Desktop\Práctica Tema 5 - Curso 2019-2020\Tienda\GesTienda\App\_Data\Tiend.mdf failed. A database with the same name exists, or specified file cannot be opened, or it is located on UNC share.*

**Descripción:** Excepción no controlada al ejecutar la solicitud Web actual. Revise el seguimiento de la pila para obtener más información acerca del error y dónde se originó en el código.

**Detalles de la excepción:** System.Data.SqlClient.SqlException: An attempt to attach an auto-named database for file C:\Users\vicente\Desktop\Práctica Tema 5 - Curso 2019-2020\Tienda\GesTienda\App\_Data\Tiend.mdf failed. A database with the same name exists, or specified file cannot be opened, or it is located on UNC share.

**Error de código fuente:**

Se ha generado una excepción no controlada durante la ejecución de la solicitud Web actual. La información sobre el origen y la ubicación de la excepción pueden identificarse utilizando la excepción del seguimiento de la pila siguiente.

**Seguimiento de la pila:**

```
[SqlException (0x80131904): An attempt to attach an auto-named database for file C:\Users\vicente\Desktop\Práctica Tema 5 - Curso 2019-2020\Tienda\GesTienda\App_Data\Tiend.mdf failed. A database with the same name exists, or specified file cannot be opened, or it is located on UNC share.]
System.Data.SqlClient.SqlInternalConnectionTds..ctor(DbConnectionPoolIdentity identity, SqlConnectionString connectionOptions, SqlCredential credential, SqlTransaction transaction)
System.Data.SqlClient.SqlConnectionFactory.CreateConnection(DbConnectionOptions options, DbConnectionPoolKey poolKey, SqlConnection string)
```

Cuando se produce un error en tiempo de ejecución, la información que proporciona el depurador de Visual Studio va dirigida a los desarrolladores de la aplicación web y describe técnicamente la naturaleza del error que se ha producido. En las versiones de explotación de las aplicaciones Web que utilizarán los usuarios finales debe proporcionarse una información menos técnica y que esté más adaptada a las necesidades de los usuarios finales. Por este motivo, es necesario establecer un mecanismo de control de errores a nivel de aplicación que, por una parte, gestione el error producido y, por otra, proporcione una información adecuada para los usuarios finales.

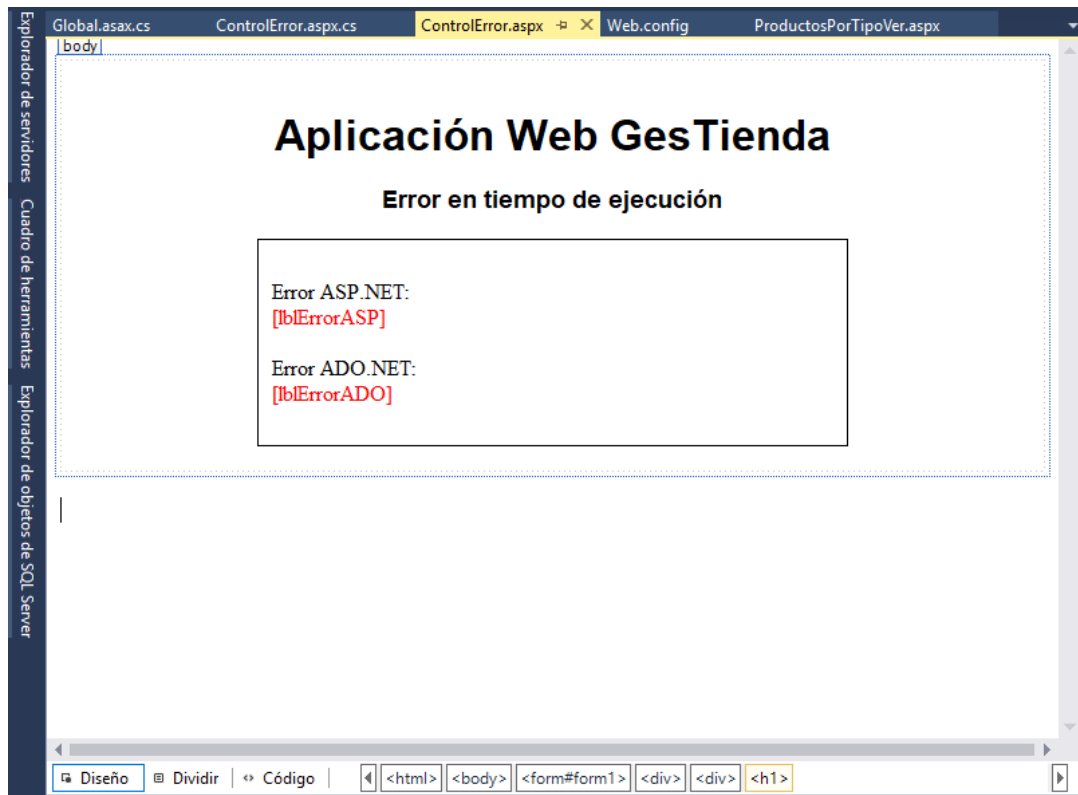
### Controlar los errores a nivel de aplicación Web

A continuación, se procederá a realizar una gestión de los errores en tiempo de ejecución a nivel de aplicación Web mediante la creación de un Web Form, denominado *ControlError.aspx*. Este Web Form se encargará de mostrar una descripción sobre la naturaleza del error producido y de controlar el procesamiento más adecuado asociado al error producido.

El Web Form *ControlError.aspx* mostrará todos los errores en tiempo de ejecución que se produzcan durante el uso de la aplicación Web y que no sean gestionados a nivel de página o a nivel de programación a través de estructuras *try/catch*. El funcionamiento de este modo de captura de errores en tiempo de ejecución es sencillo. Al producirse un error, el controlador de la aplicación Web transfiere el control a un Web Form concreto que gestiona los errores, visualizando el origen de cada error y facilitando al usuario la posibilidad de continuar con la operación de la Aplicación web.

Para crear el Web Form *ControlError.aspx* de control de errores, realizar las siguientes acciones:

1. Crear un nuevo Web Form denominado *ControlError.aspx*, en la carpeta raíz del Proyecto de aplicación Web *GesTienda*.
2. En la vista **Diseño del** nuevo Web Form *ControlError.aspx*, añadir los controles necesarios desde el **Cuadro de herramientas**, de modo que se obtenga una apariencia de diseño visual similar a la siguiente.



Como puede apreciarse en la imagen anterior, en el Web Form se han incluido dos controles *Label*, denominados *lblErrorASP* y *lblErrorADO*, que mostrarán la descripción de los errores provenientes de ASP.NET y de ADO.NET, respectivamente.

3. Añadir al evento *Load* del Web Form *ControlError.aspx* el código lógico que se muestra a continuación para recuperar la excepción que provocó el error, mostrar la descripción del error en las etiquetas correspondientes y, finalmente, eliminar el error.

```
protected void Page_Load(object sender, EventArgs e)
{
    Exception exUltimoErrorASP, exUltimoErrorADO;
    String strMensajeErrorASP = "No se ha producido un error de ASP.NET";
    String strMensajeErrorADO = "No se ha producido un error de ADO.NET";

    // Captura la excepción que provocó el último error de ASP.NET
    exUltimoErrorASP = Server.GetLastError();

    // Gestión de la excepción provocada
    if (exUltimoErrorASP != null)
    {
        strMensajeErrorASP = exUltimoErrorASP.Message;
        // Se captura el error base que ha provocado el error de ASP.NET, que
        // será un error de ADO.NET
        exUltimoErrorADO = exUltimoErrorASP.GetBaseException();
        if (exUltimoErrorADO != null)
        {
            strMensajeErrorADO = exUltimoErrorADO.Message;
        }
    }
}
```



```
}  
  
lblErrorASP.Text = strMensajeErrorASP;  
lblErrorADO.Text = strMensajeErrorADO;  
  
// Eliminación el error del servidor una vez mostrado  
Server.ClearError();  
}
```

4. A continuación, es necesario transferir el control de ejecución de la aplicación hacia el Web Form *ControlError.aspx* en el momento en que se produzca cualquier error en tiempo de ejecución en la aplicación Web. Para ello, se utiliza la clase global de la aplicación Web que se define en un archivo específico denominado *Global.asax*. Para crear este archivo, hacer:
- En el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre de la Aplicación Web *GesTienda*, desplegar la opción **Agregar** en el menú contextual y seleccionar la opción **Nuevo elemento...**
  - En la ventana **Agregar nuevo elemento**, desplegar la opción **Instalado, Visual C#** y seleccionar la opción **Web**. En la parte central de la ventana, seleccionar el tipo de elemento **Clase de aplicación global**. Comprobar que el nombre asignado de manera predeterminada es *Global.asax* y hacer clic en **Agregar**.
  - En el archivo de clase global de aplicación *Global.asax* incluir el siguiente código lógico asociado al evento *Application\_Error*.

```
void Application_Error(object sender, EventArgs e)  
{  
    // Código que se ejecuta al producirse un error no controlado  
    Server.Transfer("~/ControlError.aspx");  
}
```

La clase global de aplicación, que se especifica en el archivo *Global.asax*, permite establecer el código lógico asociado a los eventos de la aplicación Web. La siguiente tabla incluye una breve descripción de algunos de los eventos de la aplicación Web más importantes:

Evento	Descripción sobre cuando se ejecuta el código
Application_Start	Al iniciar la Aplicación web. Se produce cuando el primer usuario accede a un Web Form de la Aplicación Web, iniciándose también su sesión.
Application_End	Al finalizar la Aplicación web. Se produce cuando cierra su sesión el último usuario que utilizaba la aplicación.
Application_Error	Al producirse un error no controlado.
Session_Start	Al iniciar una nueva sesión. Se produce cuando un usuario accede a la Aplicación Web. Existen tantas sesiones como instancias de ejecución de la Aplicación Web. Además, un usuario puede tener varias sesiones abiertas.
Session_End	Al finalizar la sesión. Se produce cuando un usuario finaliza una sesión.

5. Iniciar la depuración del Web Form *ProductosPorTipoVer.aspx* para comprobar que ahora el error en tiempo de ejecución anterior es gestionado a través del Web Form *ControlError.aspx* y así, poder comprobar los resultados obtenidos.

## Aplicación Web GesTienda

### Error en tiempo de ejecución

Error ASP.NET:

Se produjo una excepción de tipo 'System.Web.HttpUnhandledException'.

Error ADO.NET:

An attempt to attach an auto-named database for file C:\Users\vicente\Desktop\Práctica Tema 5 - Curso 2019-2020\Tienda\GesTienda\App\_Data\Tiend.mdf failed. A database with the same name exists, or specified file cannot be opened, or it is located on UNC share.

Este control de errores a nivel de aplicación actuará cuando se produzca un error en tiempo de ejecución en los controles de datos del modelo declarativo de acceso a datos.

6. Una vez comprobado el funcionamiento del Web Form de control de errores a nivel de aplicación Web, es necesario modificar el archivo *Web.config* para corregir el nombre del archivo de base de datos *Tienda.mdf* que provocaba el error que ha permitido comprobar el funcionamiento del control de errores en tiempo de ejecución a nivel de aplicación Web. Una vez corregido este error, iniciar la depuración para comprobar que la aplicación Web funciona perfectamente sin que se produzcan errores en tiempo de ejecución.

## Ejercicio 5

### Crear un Web Form para consulta de datos mediante código

En este ejercicio se creará un Web Form denominado *ProductosVer.aspx* para consultar datos, empleando para ello el modelo de acceso a datos mediante código.

#### Crear un Web Form de consulta de datos empleando el objeto *DataReader* de ADO.NET

1. Crear un nuevo Web Form denominado *ProductosVer.aspx* que quede vinculado con la página maestra *MasterPage.Master*.
2. Tal como se ha realizado en el Ejercicio 2 de la práctica, añadir al elemento de contenido reemplazable, denominado *InfoContenido*, del Web Form *ProductosVer.aspx* los siguientes elementos: el elemento de título del Web Form, el control Label *lblResultado* para mostrar el resultado del procesamiento realizado mediante código lógico y el control Label *lblMensajes* para proporcionar información al usuario sobre el procesamiento realizado. En este caso, el título del Web Form será: "Productos".
3. A continuación, se va a añadir código lógico en el archivo de código subyacente del Web Form *ProductosVer.aspx* para mostrar la información de los productos de la tienda. Para ello, hacer:
  - a. Abrir el archivo de código subyacente *ProductosVer.aspx.cs*.
  - b. En la sección *using*, añadir las siguientes líneas para poder utilizar las clases contenidas en los namespaces correspondientes. El namespace *System.Data.SqlClient* facilita el uso de las clases de ADO.NET para el acceso a bases de datos de Microsoft SQL Server y el namespace *System.Configuration* facilita el uso del objeto *ConfigurationManager*.

```
using System.Data.SqlClient;  
using System.Configuration;
```

- c. Añadir el siguiente código lógico asociado al evento *Load* del objeto *Page* en el archivo de código subyacente denominado *ProductosVer.aspx.cs*.

```
protected void Page_Load(object sender, EventArgs e)  
{  
    int InNumeroFilas;  
    string StrResul, StrError;  
    string StrCadenaConexion =  
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;  
    string StrComandoSql = "SELECT IdProducto,DesPro,PrePro,IdUnidad,DesTip FROM PRODUCTO " +  
        "INNER JOIN TIPO ON PRODUCTO.IdTipo = TIPO.IdTipo;";  
    using (SqlConnection conexion = new SqlConnection(StrCadenaConexion))  
    {  
        try  
        {  
            conexion.Open();  
            SqlCommand comando = conexion.CreateCommand();  
            comando.CommandText = StrComandoSql;  
            SqlDataReader reader = comando.ExecuteReader();  
            if (reader.HasRows)  
            {  
                StrResul = "<div style='display:table;border-style:solid;border-color:#336699;width:90%'>";  
                StrResul += "<div style='display:table-row; background:#336699;color:white'>";  
                StrResul += "<div style='display:table-cell; font-weight:bold'>Código</div>";  
                StrResul += "<div style='display:table-cell; font-weight:bold'>Descripción</div>";  
                StrResul += "<div style='display:table-cell; font-weight:bold'>Precio</div>";  
            }  
        }  
    }  
}
```

```
StrResul += "<div style='display:table-cell; font-weight:bold'>Unidad</div>";
StrResul += "<div style='display:table-cell; font-weight:bold'>Tipo Producto</div>";
StrResul += "</div>";
InNumeroFilas = 0;
while (reader.Read())
{
    StrResul += "<div style='display:table-row'>";
    StrResul += "<div style='display:table-cell'> &nbsp;" + reader.GetString(0) + "</div>";
    StrResul += "<div style='display:table-cell'>" + reader.GetString(1) + "</div>";
    StrResul += "<div style='display:table-cell; text-align: right'>"
        + string.Format("{0:C}", reader.GetValue(2)) + "&nbsp;" + "</div>";
    StrResul += "<div style='display:table-cell'>" + reader.GetString(3) + "</div>";
    StrResul += "<div style='display:table-cell'>" + reader.GetString(4) + "</div>";
    StrResul += "</div>";
    InNumeroFilas++;
}
StrResul += "</div>";
StrResul += "<p> Número de Filas: " + InNumeroFilas + "</p>";
lblResultado.Text = StrResul;
}
else
{
    lblMensajes.Text = "No existen registros resultantes de la consulta";
}
reader.Close();
}
catch (SqlException ex)
{
    StrError = "<p>Se han producido errores en el acceso a la base de datos.</p>";
    StrError = StrError + "<div>Código: " + ex.Number + "</div>";
    StrError = StrError + "<div>Descripción: " + ex.Message + "</div>";
    lblMensajes.Text = StrError;
    return;
}
}
```

Como puede apreciarse en el código anterior, el resultado del procesamiento del código lógico se va almacenando en la variable de tipo *String* denominada *StrResul* para, posteriormente, asignar este valor al control de tipo Label denominado *lblResultado*, de manera que este resultado se presentará visualmente sobre la interfaz web. Igualmente puede apreciarse que los errores en tiempo de ejecución que se puedan producir durante el procesamiento del código lógico se almacenan en la variable *StrError*, para, posteriormente, asignar este valor al control, Label denominado *lblMensajes*, de manera que estos errores se presentarán visualmente en la interfaz web en el caso que se produzcan. En el código anterior puede apreciarse que la conexión a la base de datos se especifica mediante **ConfigurationManager** haciendo referencia a la cadena de conexión definida en el archivo Web.Config. También puede observarse en el código anterior, el uso que se hace de los objetos **Connection**, **Command** y **SqlDataReader** para acceder a la información que almacena como consecuencia de la ejecución de una consulta de tipo SELECT de SQL, a través del comando correspondiente. Se recomienda revisar y analizar el código anterior con la finalidad de **comprender con el mayor nivel de detalle cuál es el sentido de cada una de las líneas de código lógico** empleadas para acceder a la información de la base de datos. Se recomienda realizar esto mismo con todo el código lógico incluido en las prácticas. Las prácticas son actividades de formación que deben ser consideradas como elementos activos del aprendizaje. En este sentido, deben emplearse para ejercitar, experimentar, indagar, reflexionar, etc. sobre los contenidos que se estudian.

4. Iniciar la depuración para comprobar los resultados obtenidos.
5. Abrir la vista **Diseño** del archivo de página maestra *MasterPage.Master*. En el control TreeView incluido en el elemento `<div>` denominado *menu*, acceder a las **Tareas de TreeView** para editar sus nodos y modificar la propiedad *NavigateUrl* asociada a la opción de segundo nivel denominada *Ver productos*, que está incluida en la opción de primer nivel *Productos*, para facilitar el acceso al Web Form *ProductosVer.aspx*.
6. Iniciar la depuración para comprobar los resultados obtenidos.

comerciodaw.com

TIENDA ONLINE - SHOPPING DAW

Usuario:

Inicio

Productos

Compras

Productos

Código	Descripción	Precio	Unidad	Tipo Producto
0032-895	Patatas	2,00 €	Kilogramo	Alimentación y bebidas
0231-227	Pantalones	25,30 €	Unidad	Vestido y ropa de señora, caballero y moda joven
0290-456	Melones	4,00 €	Kilogramo	Alimentación y bebidas
1000-100	Camisa	15,60 €	Unidad	Vestido y ropa de señora, caballero y moda joven
2348-312	TV LG LCD 36"	384,00 €	Unidad	Electrodomésticos línea blanca y marrón
2906-126	Aceite	5,00 €	Litro	Alimentación y bebidas
4398-119	Zapatos señora Mod. Niza	45,70 €	Par	Zapatos de seños y caballero
4500-450	Zapatos caballero Mod. Ibiza	30,00 €	Par	Zapatos de seños y caballero
4502-341	Corbata	9,50 €	Unidad	Vestido y ropa de señora, caballero y moda joven
5578-784	TV Panasonic LCD 42"	749,00 €	Unidad	Electrodomésticos línea blanca y marrón
6784-998	Chandal	12,00 €	Unidad	Vestido y ropa de señora, caballero y moda joven
8034-556	Huevos. Categoría extra	2,50 €	Docena	Alimentación y bebidas
8898-009	Lavadora-Secadora AEG 8/5 Kg.	1.199,00 €	Unidad	Electrodomésticos línea blanca y marrón
9002-445	Sandías	2,50 €	Kilogramo	Alimentación y bebidas

Número de Filas: 14

© Desarrollo de Aplicaciones Web interactivas con Acceso a Datos  
IES Mare Nostrum (Alicante)

## Ejercicio 6

### Crear un Web Form para consulta de datos mediante código y controles de datos declarativos

En este ejercicio se muestra cómo interactúan los controles de datos del modelo declarativo, como es un control GridView, con controles simples cuyos valores se obtienen a través de código lógico.

Las Aplicaciones Web disponen siempre de varios perfiles o casos de uso. En la Aplicación Web *GesTienda* existirán dos perfiles de uso: Usuario y Administrador. Los usuarios pertenecientes al rol de Usuario podrán acceder a la información de los productos, ver y modificar sus datos, realizar compras y gestionar sus propios pedidos. Los usuarios pertenecientes al rol de Administrador podrán gestionar completamente los productos, clientes, pedidos y usuarios. De manera que, los procesamientos a los que puede acceder cada usuario de la Aplicación Web dependen del rol al cual pertenezca ese usuario concreto. Por este motivo, la presentación visual de los Web Forms a los que acceden los usuarios que pertenezcan al rol Usuario, será distinta de la presentación visual de los Web Forms a los que accedan los usuarios del rol Administrador. Ello es debido, principalmente, a que el menú que utilizará cada uno de ellos es diferente, porque permitirá acceder a procesamientos diferentes que son propios de cada rol de usuario. Además, habitualmente, los distintos perfiles de uso de una Aplicación Web suelen tener una presentación visual diferente. En ejercicios posteriores se completará el sistema de autenticación de usuarios para la Aplicación Web *GesTienda*. En este ejercicio se va a crear un Web Form que permitirá visualizar la información de los pedidos realizados por cada cliente. Este Web Form podrán ejecutarlo solo los usuarios que pertenecen al rol de Administrador, por lo que se vinculará con la página maestra correspondiente, denominada *MasterPageAdm.Master*, que ha sido proporcionada por el profesor como recurso de las prácticas.

#### Crear un Web Form de consulta con controles de datos y controles simples

1. Agregar a la aplicación Web *GesTienda* la página maestra *MasterPageAdm.Master*.
2. Crear un nuevo Web Form denominado *PedidosPorCliente.aspx* que quede vinculado con la página maestra *MasterPageAdm.Master*.
3. Abrir el Web Form *PedidosPorCliente.aspx*, para incluir el elemento de título del Web Form, el control Label *lblResultado* y el control Label *lblMensajes*, tal como se hizo en el ejercicio anterior. En este caso, el título del Web Form será: "Pedidos realizados por los clientes".
4. Agregar un control de datos *SqlDataSource*, denominado *SqlDataSource1*, y un control de datos GridView, denominado *grdClientes*, para mostrar los datos correspondientes a los registros principales de la tabla *CLIENTE*. En el control de datos GridView se mostrarán los siguientes datos: Id del cliente, nombre, población y correo electrónico. Además, modificar el formato, habilitar la selección de filas y la paginación al control de datos GridView.
5. Se pretende que al seleccionar un cliente sobre el control de datos GridView, se muestre la información sobre los pedidos realizados por ese cliente en la etiqueta *lblResultado*. Además, se desea mostrar otra información resultante de la consulta a realizar a la base de datos. Para ello, agregar un nuevo control Label, denominado *lblTotal*, que permitirá mostrar el número de pedidos realizados y el importe total de los pedidos realizados por el cliente seleccionado en el control GridView. Asignar *false* a la propiedad *Visible* de este nuevo control Label.

6. Añadir el siguiente código asociado al evento *SelectedIndexChanged* del objeto *grdClientes*, para obtener y presentar la información de los pedidos del cliente seleccionado en el control *GridView*, así como el número y el importe total de los pedidos realizados por ese cliente.

```
protected void grdClientes_SelectedIndexChanged(object sender, EventArgs e)
{
    int InNumeroFilas;
    string StrResultado, StrError;
    string StrCadenaConexion =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;

    string strClienteSeleccionado = grdClientes.SelectedRow.Cells[1].Text;
    string StrComandoSql = "SELECT PEDIDO.IdPedido, FecPed, SerPed, CobPed, " +
        "SUM(CanDet*PreDet-CanDet*PreDet*DtoDet/100) AS Total " +
        "FROM PEDIDO INNER JOIN DETALLE ON PEDIDO.IdPedido = DETALLE.IdPedido " +
        "GROUP BY PEDIDO.IdPedido, PEDIDO.FecPed, PEDIDO.CobPed, PEDIDO.SerPed, PEDIDO.IdCliente " +
        "HAVING (PEDIDO.IdCliente = '" + strClienteSeleccionado + "')";

    decimal DcTotal = 0;
    lblMensajes.Text = "";
    lblResultado.Visible = false;
    lblTotal.Visible = false;
    using (SqlConnection conexion = new SqlConnection(StrCadenaConexion))
    {
        try
        {
            conexion.Open();
            SqlCommand comando = conexion.CreateCommand();
            comando.CommandText = StrComandoSql;
            SqlDataReader reader = comando.ExecuteReader();
            if (reader.HasRows)
            {
                InNumeroFilas = 0;
                lblResultado.Visible = true;
                lblTotal.Visible = true;
                StrResultado = "<h4>Detalle de pedidos</h4>";
                StrResultado += "<div style='display:table; border-color:#336699;width:35%'>";
                StrResultado += "<div style='display:table-row; background:#336699;color:white'>";
                StrResultado += "<div style='display:table-cell; font-weight:bold'>Núm.Pedido</div>";
                StrResultado += "<div style='display:table-cell; font-weight:bold'>Fecha</div>";
                StrResultado += "<div style='display:table-cell; font-weight:bold'>Servido</div>";
                StrResultado += "<div style='display:table-cell; font-weight:bold'>Cobrado</div>";
                StrResultado += "<div style='display:table-cell; font-weight:bold'>Total</div>";
                StrResultado += "</div>";
                while (reader.Read())
                {
                    StrResultado += "<div style='display:table-row;'>";
                    StrResultado += "<div style='display:table-cell; text-align: center'>" +
                        reader.GetValue(0) + "</div>";
                    StrResultado += "<div style='display:table-cell'>" +
                        string.Format("{0:d}", reader.GetValue(1)) + "</div>";
                    if (reader.GetBoolean(2) == true)
                        StrResultado += "<div style='display:table-cell'> Sí </div>";
                    else
                        StrResultado += "<div style='display:table-cell'> No </div>";
                    if (reader.GetBoolean(3) == true)
                        StrResultado += "<div style='display:table-cell'> Sí </div>";
                    else
                        StrResultado += "<div style='display:table-cell'> No </div>";
                    StrResultado += "<div style='display:table-cell; text-align: right'>" +
                        string.Format("{0:c}", reader.GetValue(4)) + "&nbsp; </div>";
                    StrResultado += "</div>";
                    InNumeroFilas++;
                }
                StrResultado += "</div>";
            }
        }
    }
}
```



```
lblResultado.Text = StrResultado;
reader.Close();
string StrComandoSql1 = "SELECT SUM(CanDet*PreDet-CanDet*PreDet*DtoDet/100) AS Total " +
    "FROM CLIENTE INNER JOIN PEDIDO ON CLIENTE.IdCliente = PEDIDO.IdCliente " +
    "INNER JOIN DETALLE ON PEDIDO.IdPedido = DETALLE.IdPedido " +
    "GROUP BY CLIENTE.IdCliente " +
    "HAVING (CLIENTE.IdCliente = '" + strClienteSeleccionado + "')";
using (SqlConnection conexion1 = new SqlConnection(StrCadenaConexion))
{
    conexion1.Open();
    SqlCommand comando1 = conexion.CreateCommand();
    comando1.CommandText = StrComandoSql1;
    DcTotal = Convert.ToDecimal(comando1.ExecuteScalar());
}
lblTotal.Text = "<div> Número pedidos realizados: " + InNumeroFilas + "</div>" +
    "<div>Importe total de los pedidos realizados por el cliente: " +
    string.Format("{0:c}", DcTotal) + "</div>";
}
else
{
    lblMensajes.Text = "No existen registros resultantes de la consulta";
    reader.Close();
}
}
catch (SqlException ex)
{
    StrError = "<p>Se han producido errores en el acceso a la base de datos.</p>";
    StrError = StrError + "<div>Código: " + ex.Number + "</div>";
    StrError = StrError + "<div>Descripción: " + ex.Message + "</div>";
    lblMensajes.Text = StrError;
    return;
}
}
```

En el código anterior, puede apreciarse el uso que se hace de la propiedad **SelectedRow** del control GridView *grdClientes* para poder acceder al valor que se muestra en la segunda columna de la fila seleccionada en el GridView, a través de la siguiente línea de código:

```
string strClienteSeleccionado = grdClientes.SelectedRow.Cells[1].Text;
```

La colección *Cells[]* proporciona acceso a los valores de las columnas de la fila seleccionada en un control GridView, mediante un índice que comienza en 0. En este caso, se utiliza el valor 1 del índice para acceder a la segunda columna, que corresponde con valor del *IdCliente*.

7. A continuación, para evitar que, al hacer clic sobre los elementos de navegación del control GridView para cambiar de página, pueda seguir apareciendo información correspondiente a los pedidos del cliente seleccionado anteriormente, añadir el siguiente código en el evento *PageIndexChanged* del control *grdClientes*.

```
protected void grdClientes_PageIndexChanged(object sender, EventArgs e)
{
    grdClientes.SelectedIndex = -1;
    lblResultado.Text = "";
    lblTotal.Text = "";
    lblMensajes.Text = "";
    lblResultado.Visible = false;
    lblTotal.Visible = false;
}
```

- En el código anterior, la acción de asignar el valor -1 a la propiedad *SelectedIndex* de un control GridView establece que no se selecciona ninguna fila.
- Ajustar las propiedades de los controles del Web Form, para que su presentación visual sea similar a la que se muestra en la siguiente ilustración.



comerciodaw.com

**TIENDA ONLINE - SHOPPING DAW**

Rol Administrador:

Inicio  
 Productos  
 Ventas  
 Clientes  
 Usuarios

### Pedidos realizados por los clientes

	Id Cliente	Nombre	Población	Correo electrónico
<input type="button" value="Seleccionar"/>	00123659Z	Beatriz Iraola López	Bilbao	beatriz@empresa.com
<input type="button" value="Seleccionar"/>	07899905V	Jesús García Soria	Zaragoza	suso@empresa.com
<input type="button" value="Seleccionar"/>	08659442S	Miguel Sogorb Fenoll	Alicante	msogorb@empresa.com
<input type="button" value="Seleccionar"/>	09897907K	Alvaro Fernández Moreno	Madrid	afm@empresa.com
<input type="button" value="Seleccionar"/>	10900801K	Alejandro Castro Hernández	Zamora	acastro@empresa.com

[Siguiete](#) [Último](#)

#### Detalle de pedidos

Núm. Pedido	Fecha	Servido	Cobrado	Total
6	05/01/2012	Sí	Sí	48,00 €
10	07/01/2012	No	No	80,78 €

Número pedidos realizados: 2  
 Importe total de los pedidos realizados por el cliente: 128,78 €

© Desarrollo de Aplicaciones Web interactivas con Acceso a Datos  
 IES Mare Nostrum (Alicante)

9. Iniciar la depuración para comprobar los resultados obtenidos.
10. Abrir la vista Diseño del archivo de página maestra *MasterPageAdm.Master*. En el control TreeView incluido en el elemento `<div>` denominado *menu*, acceder a las **Tareas de TreeView** para editar sus nodos y modificar la propiedad *NavigateUrl* asociada a la opción de segundo nivel denominada *Pedidos por cliente*, que está incluida en la opción de primer nivel *Ventas*, para facilitar el acceso al Web Form *PedidosPorCliente.aspx*.
11. Iniciar la depuración para comprobar los resultados obtenidos.

Se recomienda analizar y revisar el código lógico de los diferentes ejercicios de forma detallada, con la finalidad de poder comprender de forma precisa el sentido de cada una de las líneas de código que se emplean, tanto para acceder a la base de datos, como para generar dinámicamente la presentación visual en la página de respuesta.

## Ejercicio 7

# Crear un Web Form para el mantenimiento de datos de una tabla mediante código

En este ejercicio se aborda la creación de un Web Form de nombre *ProductosMantener.aspx* para mantener los datos almacenados en la tabla *PRODUCTO*, empleando el modelo de acceso a datos mediante código mediante los objetos de acceso a datos de ADO.NET. Este nuevo Web Form permitirá mostrar y seleccionar los datos de los productos mediante un GridView, insertar un nuevo producto, eliminar un producto existente y actualizar los valores de los campos de un producto existente.

### Crear la presentación de un Web Form para el mantenimiento de datos

1. Crear un nuevo Web Form denominado *ProductosMantener.aspx* que quede vinculado con la página maestra *MasterPageAdm.Master*.
2. Abrir el Web Form *ProductosMantener.aspx* para incluir el elemento de título del Web Form, el control Label *lblResultado* y el control Label *lblMensajes*, tal como se hizo en un ejercicio anterior. En este caso, el título del Web Form será: "Mantenimiento productos".
3. Agregar un control de datos *SqlDataSource*, denominado *SqlDataSource1*, y un control de datos GridView, denominado *grdProductos*, para mostrar los datos correspondientes a los registros principales de la tabla maestra *PRODUCTO*. En el control GridView se mostrarán los siguientes campos de la tabla: Id de producto, descripción y precio. Además, modificar el formato, habilitar la selección de filas y la paginación al control de datos GridView.
4. Iniciar la depuración para comprobar los resultados obtenidos hasta el momento.
5. A continuación, se agregarán controles simples para presentar los valores de los campos del producto seleccionado en el control GridView. Para ello, hacer:
  - a. Para cada campo, agregar un control Label para poder mostrar una descripción.
  - b. Agregar tres controles simples TextBox para presentar los valores de los campos Id de producto, Descripción y Precio. Y, agregar dos controles simples DropDownList para mostrar los valores de los campos Unidad y Tipo de producto.
  - c. Modificar las propiedades de los controles agregados del siguiente de modo:

Tipo de control	Asociado al campo	Id	Text	Enabled
Label		<b>lblIdProducto</b>	Id. Producto	
TextBox	<i>IdProducto</i>	<b>txtIdProducto</b>		<i>False</i>
Label		<b>lblDesPro</b>	Descripción	
TextBox	<i>DesPro</i>	<b>txtDesPro</b>		<i>False</i>
Label		<b>lblPrePro</b>	Precio	
TextBox	<i>PrePro</i>	<b>txtPrePro</b>	0	<i>False</i>
Label		<b>lblIdUnidad</b>	Unidad	
DropDownList	<i>IdUnidad</i>	<b>ddlIdUnidad</b>		<i>False</i>
Label		<b>lblIdTipo</b>	Tipo Producto	
DropDownList	<i>IdTipo</i>	<b>ddlIdTipo</b>		<i>False</i>

6. Agregar un control de datos *SqlDataSource*, denominado *SqlDataSource2*, para mostrar todas las filas de la tabla *UNIDAD*. A continuación, enlazar el control DropDownList, denominado *ddlIdUnidad*, con el control de datos *SqlDataSource2*.

7. Agregar un control de datos `SqlDataSource`, denominado `SqlDataSource3`, para mostrar todas las filas de la tabla `TIPO`. Enlazar el control `DropDownList`, denominado `ddlIdTipo`, con el control de datos `SqlDataSource3`. El campo de datos de la tabla `TIPO` a mostrar será `DesTip`, mientras que el campo de datos para el valor del control `DropDownList` será `IdTipo`.
8. Agregar siete controles `Button`, que permitirán acceder a los diferentes procesamientos para la inserción de un nuevo registro y la actualización o eliminación de un registro seleccionado mediante el control `GridView`. La idea es hacer visibles y ocultar los controles `Button` que corresponda cuando se realice cada uno de estos procesamientos. A cada uno de ellos, asignarles los valores de las propiedades que se muestran en la siguiente tabla:

Id del control	Text	Visible
<b>btnNuevo</b>	Nuevo	True
<b>btnEditar</b>	Editar	False
<b>btnEliminar</b>	Eliminar	False
<b>btnInsertar</b>	Insertar	False
<b>btnModificar</b>	Modificar	False
<b>btnBorrar</b>	Borrar	False
<b>btnCancelar</b>	Cancelar	False

9. Abrir la vista Diseño del archivo de página maestra `MasterPageAdm.Master`. En el control `TreeView` incluido en el elemento `<div>` denominado `menu`, acceder a las **Tareas de TreeView** para editar sus nodos y modificar la propiedad `NavigateUrl` asociada a la opción de segundo nivel denominada `Mantener productos`, que está incluida en la opción de primer nivel `Productos`, para facilitar el acceso al Web Form `ProductosMantener.aspx`.
10. Iniciar la depuración para comprobar los resultados. Y, ajustar el estilo y las propiedades de los elementos y controles, de modo que la presentación visual sea similar a la siguiente.

comerciodaw.com

TIENDA ONLINE - SHOPPING DAW

Rol Administrador: Cerrar sesión

Inicio

Productos

Mantener productos

Ventas

Clientes

Usuarios

Mantenimiento de productos

	Id. Producto	Descripción	Precio
Seleccionar	0032-895	Patatas	2,00 €
Seleccionar	0231-227	Pantalones	25,30 €
Seleccionar	0290-456	Melones	4,00 €
Seleccionar	1000-100	Camisa	15,60 €
Seleccionar	2348-312	TV LG LCD 36"	384,00 €
Seleccionar	2906-126	Aceite	5,00 €
Seleccionar	4398-119	Zapatos señora Mod. Niza	45,70 €
Seleccionar	4500-450	Zapatos caballero Mod. Ibiza	30,00 €
Seleccionar	4502-341	Corbata	9,50 €
Seleccionar	5578-784	TV Panasonic LCD 42"	749,00 €

Siguiente Último

Id. Producto

Descripción

Precio

Unidad Docena

Tipo Producto Alimentación y bebidas

Nuevo

© Desarrollo de Aplicaciones Web Interactivas con Acceso a Datos  
IES Mare Nostrum (Alicante)

## Añadir código al Web Form para el mantenimiento de datos

Una vez ajustada la presentación visual del Web Form *ProductosMantener.aspx*, se añadirá el código lógico para procesar las opciones de mantenimiento de datos mediante el uso de controles simples.

1. Para mostrar en los controles simples del Web Form los datos del producto seleccionado en el control GridView, añadir el siguiente código lógico asociado al evento *SelectedIndexChanged* del control GridView denominado *grdProductos*.

```
protected void grdProductos_SelectedIndexChanged(object sender, EventArgs e)
{
    lblMensajes.Text = "";
    FnDeshabilitarControles();
    string StrIdProducto = grdProductos.SelectedRow.Cells[1].Text;
    string StrCadenaConexion =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    string StrComandoSql = "SELECT IdProducto,DesPro,PrePro,IdUnidad,PRODUCTO.IdTipo,DesTip " +
        " FROM PRODUCTO INNER JOIN TIPO ON PRODUCTO.IdTipo = TIPO.IdTipo " +
        "WHERE PRODUCTO.IdProducto = '" + StrIdProducto + "'";
    using (SqlConnection conexion = new SqlConnection(StrCadenaConexion))
    {
        try
        {
            conexion.Open();
            SqlCommand comando = conexion.CreateCommand();
            comando.CommandText = StrComandoSql;
            SqlDataReader reader = comando.ExecuteReader();
            if (reader.HasRows)
            {
                reader.Read();
                txtIdProducto.Text = reader.GetString(0);
                txtDesPro.Text = reader.GetString(1);
                txtPrePro.Text = string.Format("{0:C}", reader.GetDecimal(2));
                ddlIdUnidad.SelectedItem.Selected = false;
                ddlIdUnidad.SelectedItem.Text = reader.GetString(3);
                ddlIdTipo.SelectedItem.Selected = false;
                ddlIdTipo.SelectedItem.Text = reader.GetString(5);
            }
            else
            {
                lblMensajes.Text = "No existen registros resultantes de la consulta";
            }
            reader.Close();
            btnNuevo.Visible = true;
            btnEditar.Visible = true;
            btnEliminar.Visible = true;
            btnInsertar.Visible = false;
            btnModificar.Visible = false;
            btnBorrar.Visible = false;
            btnCancelar.Visible = false;
        }
        catch (SqlException ex)
        {
            string StrError = "<p>Se han producido errores en el acceso a la base de datos.</p>";
            StrError = StrError + "<div>Código: " + ex.Number + "</div>";
            StrError = StrError + "<div>Descripción: " + ex.Message + "</div>";
            lblMensajes.Text = StrError;
            return;
        }
    }
}
```

- Añadir las siguientes funciones al archivo de código subyacente *ProductosMantener.aspx.cs*, que permitirán habilitar y deshabilitar los controles simples que soportan los datos de los productos. Con ayuda de estas funciones, se deshabilitarán los controles cuando se muestran los datos y se habilitarán cuando se deban editar los valores para insertar o modificar filas. La función para deshabilitar los controles simples es la siguiente:

```
protected void FnDeshabilitarControles()
{
    txtIdProducto.Enabled = false;
    txtDesPro.Enabled = false;
    txtPrePro.Enabled = false;
    ddlIdUnidad.Enabled = false;
    ddlIdTipo.Enabled = false;
}
```

La función para habilitar los controles simples es la siguiente:

```
protected void FnHabilitarControles()
{
    txtIdProducto.Enabled = true;
    txtDesPro.Enabled = true;
    txtPrePro.Enabled = true;
    ddlIdUnidad.Enabled = true;
    ddlIdTipo.Enabled = true;
}
```

- Iniciar la depuración para comprobar que al seleccionar una fila en el control de datos *GridView*, se muestran correctamente los valores de los campos del registro seleccionado en los controles simples que presentan los datos situados hacia la derecha del Web Form.

comerciodaw.com

TIENDA ONLINE - SHOPPING DAW

Rol Administrador: Cerrar sesión

Inicio  
 Productos  
 Ventas  
 Clientes  
 Usuarios

### Mantenimiento de productos

	Id. Producto	Descripción	Precio
Seleccionar	0032-895	Patatas	2,00 €
Seleccionar	0231-227	Pantalones	25,30 €
Seleccionar	0290-456	Melones	4,00 €
Seleccionar	1000-100	Camisa	15,60 €
Seleccionar	2348-312	TV LG LCD 36"	384,00 €
Seleccionar	2906-126	Aceite	5,00 €
Seleccionar	4398-119	Zapatos señora Mod. Niza	45,70 €
Seleccionar	4500-450	Zapatos caballero Mod. Ibiza	30,00 €
Seleccionar	4502-341	Corbata	9,50 €
Seleccionar	5578-784	TV Panasonic LCD 42"	749,00 €

[Siguiente](#)
[Último](#)

Id. Producto: 4398-119  
 Descripción: Zapatos señora Mod. Niza  
 Precio: 45,70 €  
 Unidad: Par  
 Tipo Producto: Zapatos de señas y caballero  

[Nuevo](#)
[Editar](#)
[Eliminar](#)

© Desarrollo de Aplicaciones Web Interactivas con Acceso a Datos  
 IES Mare Nostrum (Alicante)

4. Añadir el siguiente código asociado al evento *Click* del control Button denominado *btnNuevo* para limpiar los controles simples donde se introducirán los datos a insertar en la tabla.

```
protected void btnNuevo_Click(object sender, EventArgs e)
{
    lblMensajes.Text = "";
    btnNuevo.Visible = false;
    btnEditar.Visible = false;
    btnEliminar.Visible = false;
    btnInsertar.Visible = true;
    btnModificar.Visible = false;
    btnBorrar.Visible = false;
    btnCancelar.Visible = true;
    txtIdProducto.Text = "";
    txtDesPro.Text = "";
    txtPrePro.Text = Convert.ToString(0);
    ddlIdUnidad.DataBind(); // Vuelve a enlazar el control para que se actualicen los datos
    ddlIdTipo.DataBind();
    grdProductos.SelectedIndex = -1;
    FnHabilitarControles();
    txtIdProducto.Focus();
}
```

En el código anterior puede apreciarse el uso que se hace del método *DataBind()* para enlazar nuevamente los datos desde el origen de datos, de modo que el resultado es que se actualizan o refrescan los datos en el control de datos.

5. A continuación, añadir el siguiente código lógico asociado al evento *Click* del control Button denominado *btnInsertar* para realizar la inserción efectiva de los valores que se estén editando en los controles simples, mediante la ejecución una sentencia de SQL de tipo INSERT.

```
protected void btnInsertar_Click(object sender, EventArgs e)
{
    lblMensajes.Text = "";
    String strIdProducto, strDescripcion, strIdUnidad, strIdTipo;
    Decimal dcPrecio;

    strIdProducto = txtIdProducto.Text;
    strDescripcion = txtDesPro.Text;
    dcPrecio = Convert.ToDecimal(txtPrePro.Text);
    strIdUnidad = ddlIdUnidad.SelectedItem.Text;
    strIdTipo = ddlIdTipo.SelectedItem.Value;
    string StrCadenaConexion =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    string StrComandoSql = "INSERT PRODUCTO " +
        "(IdProducto,DesPro,PrePro,IdUnidad,IdTipo) VALUES (" +
        "\"" + strIdProducto + "\",\"" + strDescripcion +
        "\",\" + FnComaPorPunto(dcPrecio) +
        "\",\"" + strIdUnidad + "\",\"" + strIdTipo + "\"));";
    using (SqlConnection conexion = new SqlConnection(StrCadenaConexion))
    {
        try
        {
            conexion.Open();
            SqlCommand comando = conexion.CreateCommand();
            comando.CommandText = StrComandoSql;
            int inRegistrosAfectados = comando.ExecuteNonQuery();
            if (inRegistrosAfectados == 1)
                lblMensajes.Text = "Registro insertado correctamente";
            else
                lblMensajes.Text = "Error al insertar el registro";
            btnNuevo.Visible = true;
        }
    }
}
```



```
        btnEditar.Visible = false;
        btnEliminar.Visible = false;
        btnInsertar.Visible = false;
        btnModificar.Visible = false;
        btnBorrar.Visible = false;
        btnCancelar.Visible = false;
    }
    catch (SQLException ex)
    {
        string StrError = "<p>Se han producido errores en el acceso a la base de datos.</p>";
        StrError = StrError + "<div>Código: " + ex.Number + "</div>";
        StrError = StrError + "<div>Descripción: " + ex.Message + "</div>";
        lblMensajes.Text = StrError;
        return;
    }
}
grdProductos.DataBind(); // Vuelve a enlazar el GridView para que se actualicen los datos
grdProductos.SelectedIndex = -1;
FnDeshabilitarControles();
}
```

En el código anterior puede apreciarse el uso del método *DataBind()* para enlazar nuevamente los datos desde el origen de datos, de modo que se actualizan los datos en el control GridView *grdProductos*, una vez que se haya insertado el nuevo producto en la tabla.

6. Añadir la declaración de la función *FnComaPorPunto()* al archivo de código subyacente. Como puede apreciarse en el código anterior, se utiliza la función *FnComaPorPunto()* para agregar el valor del precio a la cadena que forma la sentencia SQL. Esta función resuelve el problema de sintaxis que se produce en la instrucción *INSERT* de SQL debido a la configuración regional, al sustituir la coma decimal que aparece en los tipos decimales por un punto.

```
protected string FnComaPorPunto(decimal Numero)
{
    string StrNumero = Convert.ToString(Numero);
    string stNumeroConPunto = String.Format("{0}", StrNumero.Replace(',', '.'));
    return (stNumeroConPunto);
}
```

7. Para finalizar el desarrollo del procesamiento de inserción de una fila en la tabla *PRODUCTO*, añadir el siguiente código lógico asociado al evento *Click* del control *btnCancelar*.

```
protected void btnCancelar_Click(object sender, EventArgs e)
{
    lblMensajes.Text = "";
    btnNuevo.Visible = true;
    btnEditar.Visible = false;
    btnEliminar.Visible = false;
    btnInsertar.Visible = false;
    btnModificar.Visible = false;
    btnBorrar.Visible = false;
    btnCancelar.Visible = false;
    txtIdProducto.Text = "";
    txtDesPro.Text = "";
    txtPrePro.Text = Convert.ToString(0);
    ddlIdUnidad.DataBind();
    ddlIdTipo.DataBind();
    grdProductos.SelectedIndex = -1;
    FnDeshabilitarControles();
}
```

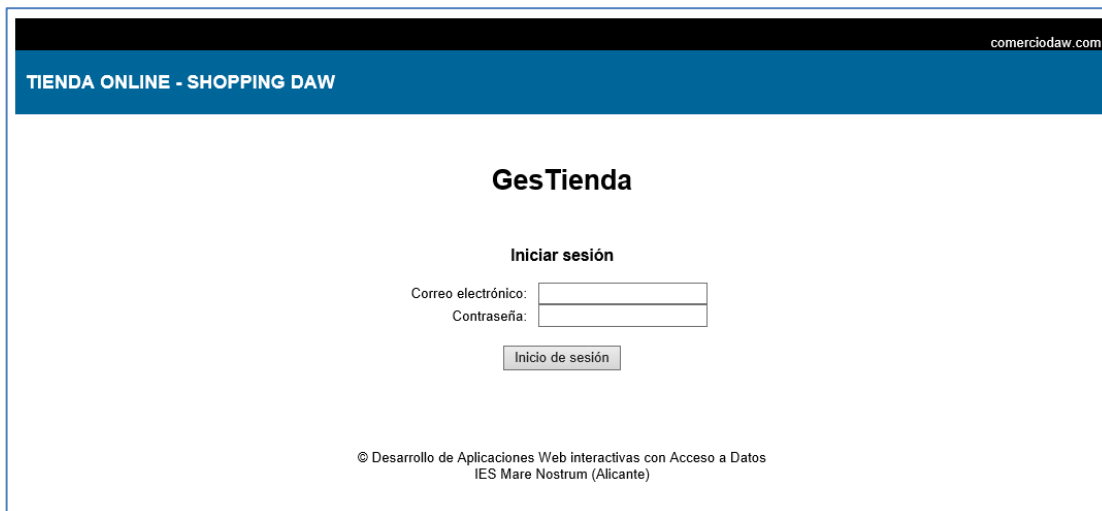
8. Iniciar la depuración para comprobar los resultados obtenidos mediante el desarrollo del procesamiento de inserción de una fila en la tabla *PRODUCTO*.
9. Desarrollar el código lógico para editar los valores de los campos de la fila seleccionada en el control GridView y actualizar los valores de la fila correspondiente en la tabla *PRODUCTO*. Para ello, al hacer clic en el control *btnEditar* se deberán habilitar los controles que contienen los valores de los campos, excepto el control *txtIdProducto* y también, se deberán hacer visibles los controles *btnModificar* y *btnCancelar*. El código lógico necesario para ejecutar la sentencia de SQL de tipo *UPDATE* se incluirá en el evento *Click* del control *btnModificar*.
10. Iniciar la depuración para comprobar los resultados obtenidos relativos a la edición de los valores de los campos de la fila seleccionada en el control GrigView y que corresponden a la tabla *PRODUCTO*.
11. Desarrollar el código lógico para eliminar la fila seleccionada en el control GridView en la tabla *PRODUCTO*. Para ello, al hacer clic en el control *btnEliminar* se harán visibles los controles *btnBorrar* y *btnCancelar*. El código lógico necesario para ejecutar la sentencia de SQL de tipo *DELETE* se incluirá en el evento *Click* del control *btnBorrar*. A criterio del alumno, se podrán definir las funciones oportunas que encapsulen la resolución de tareas comunes o genéricas.
12. Iniciar la depuración para comprobar los resultados obtenidos relativos a la eliminación de la fila seleccionada en el control GridView y que corresponden a la tabla *PRODUCTO*.

## Ejercicio 8

### Gestionar el acceso de los usuarios a la aplicación Web

En este ejercicio se incluirá al Proyecto de Aplicación Web *GesTienda* un sistema de autenticación de usuarios que permita iniciar la sesión a los usuarios de la aplicación Web y evite, por tanto, el acceso de los usuarios no autorizados. Para ello, la tabla *USUARIO* de la base de datos *Tienda.mdf* almacena la información sobre los usuarios autorizados, así como sobre su rol o perfil de usuario que especifica el tipo de acceso que tendrán a la aplicación Web: Usuario (U) o Administrador (A). Para resolver el acceso de los usuarios se empleará un control Login, aunque también podría resolverse empleando controles simples. El control Login está formado por un conjunto de controles que facilitan las tareas relacionadas con el inicio de la sesión en las aplicaciones Web. Este control es configurable, lo que permite establecer el formato más adecuado. Así, es posible editar sus elementos para incluir, además de cuadros de texto para la introducción del nombre de usuario y la contraseña, enlaces a otros Web Forms que faciliten el registro de un nuevo usuario, el cambio de la contraseña, etc. El control Login quedará agregado al Web Form de inicio de la aplicación, que se denominará *Default.aspx*, y gestionará el acceso a la Aplicación Web de los usuarios autorizados. Para ello, hacer:

1. Crear un Web Form de inicio de la Aplicación Web denominado *Default.aspx*. Este Web Form no se basará en ninguna página maestra.
2. Desde el nodo **Inicio de sesión** del **Cuadro de herramientas**, agregar un control Login al Web Form *Default.aspx* y realizar las siguientes acciones:
  - a. Sobre el control Login insertado, denominado *Login1*, hacer clic en el botón derecho y seleccionar la opción **Convertir en plantilla** para poder establecer el formato deseado.
  - b. Eliminar el texto de la opción *Recordármelo la próxima vez*.
  - c. Centrar el botón cuyo texto es *Inicio de sesión* en la fila correspondiente.
  - d. Centrar el control *Login1* sobre el contenido de la página.
  - e. Darle el formato de apariencia que parezca más adecuado.
3. En el Web Form *Default.aspx*, agregar un control Label denominado *lblMensajes*.
4. Modificar el Web Form *Default.aspx* para obtener una apariencia similar a la siguiente.



- Al iniciar la depuración del Web Form *Default.aspx* se producirá error en tiempo de ejecución. Para evitarlo, añadir el siguiente código de marcado en el archivo de configuración *Web.config* para deshabilitar la validación no-intrusiva que está habilitada de manera predeterminada.

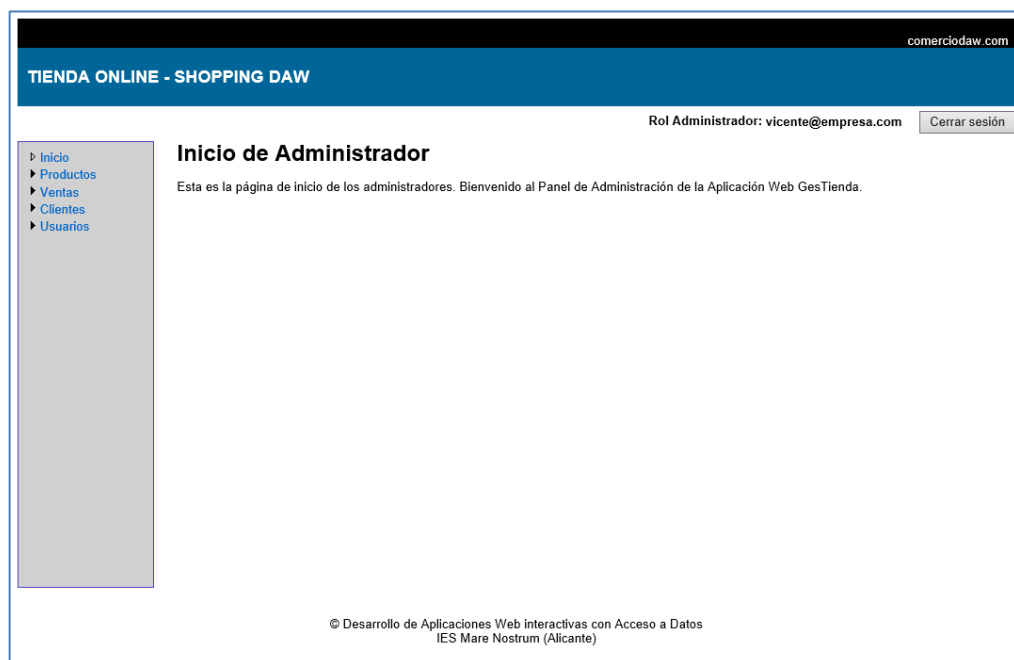
```
<appSettings>
  <add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />
</appSettings>
```

- Añadir el siguiente código lógico al evento *Authenticate* del control de tipo Login, denominado *Login1*, para establecer un control de accesos de usuarios a la Aplicación Web.

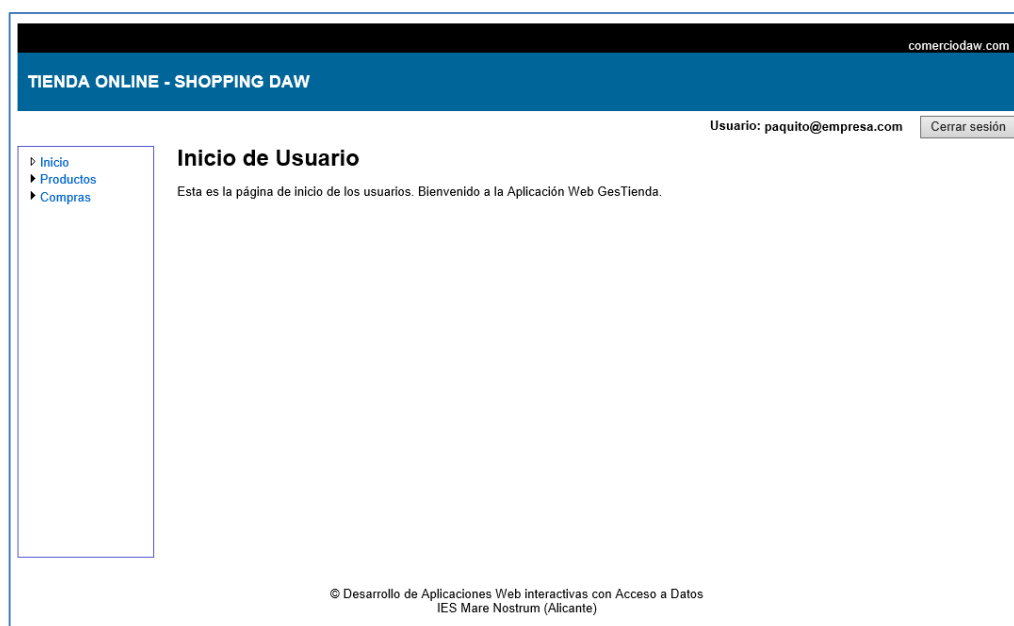
```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
{
    string StrCadenaConexion =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    string StrComandoSql = "SELECT Login, Rol FROM USUARIO ";
    StrComandoSql = StrComandoSql + " WHERE Login='" + Login1.UserName + "' ";
    StrComandoSql = StrComandoSql + "AND Password='" + Login1.Password + "'";
    using (SqlConnection conexion = new SqlConnection(StrCadenaConexion))
    {
        try
        {
            conexion.Open();
            SqlCommand comando = conexion.CreateCommand();
            comando.CommandText = StrComandoSql;
            SqlDataReader reader = comando.ExecuteReader();
            if (reader.Read())
            {
                Session.Add("Nombre", reader.GetString(0));
                Session.Add("Rol", reader.GetString(1));
                e.Authenticated = true;
                if (Convert.ToString(Session["Rol"]) == "A")
                    Response.Redirect("~/InicioAdmin.aspx");
                if (Convert.ToString(Session["Rol"]) == "U")
                    Response.Redirect("~/InicioUsuario.aspx");
            }
            else
            {
                e.Authenticated = false;
            }
        }
        catch (SqlException ex)
        {
            string StrError = "<p>Se han producido errores en el acceso a la base de datos.</p>";
            StrError = StrError + "<div>Código: " + ex.Number + "</div>";
            StrError = StrError + "<div>Descripción: " + ex.Message + "</div>";
            lblMensajes.Text = StrError;
            return;
        }
    }
}
```

- Iniciar la depuración para comprobar los resultados obtenidos. Se recomienda visualizar los datos almacenados de la tabla *USUARIO* antes de intentar el acceso, de modo que se pruebe el acceso con usuarios existentes, con contraseñas erróneas y con usuarios inexistentes.
- A continuación, se van a crear los Web Forms *InicioAdmin.aspx* e *InicioUsuario.aspx* para evitar los errores en tiempo de ejecución por recurso inexistente que se habrán producido en el apartado anterior. Para ello, crear el Web Form *InicioAdmin.aspx*, vinculado con la página maestra *MasterPageAdm.Master*, y el Web Form *InicioUsuario.aspx*, vinculado con la página

maestra *MasterPage.Master*. Ambos Web Forms constituirán el acceso inicial de los usuarios y contendrán las opciones de menú adecuadas, según sea el rol del usuario que haya iniciado sesión: Administrador (A) o Usuario (U). El perfil de Administrador de la Aplicación Web accederá al Web Form *InicioAdmin.aspx* cuya apariencia puede ser similar a la siguiente.



El perfil de usuario de la aplicación Web accederá al Web Form *InicioUsuario.aspx* que podrá tener una apariencia similar a la siguiente.



A continuación, en el control TreeView de la página maestra *MasterPageAdm.Master* comprobar o añadir las siguientes opciones y los enlaces de navegación asociados:

Nodos de primer nivel	Nodos de segundo nivel	Propiedad <i>SelecAction</i>	Propiedad <i>NavigateUrl</i>
Inicio		Select	~/InicioAdmin.aspx
Productos		Expand	
	Mantener productos	Select	~/ProductosMantener.aspx
Ventas		Expand	
	Pedidos por cliente	Select	~/PedidosPorCliente.aspx

Y en el control TreeView de la página maestra *MasterPage.Master* comprobar o añadir las siguientes opciones y los enlaces de navegación asociados:

Nodos de primer nivel	Nodos de segundo nivel	Propiedad <i>SelecAction</i>	Propiedad <i>NavigateUrl</i>
Inicio		Select	~/InicioUsuario.aspx
Productos		Expand	
	Tipos de productos	Select	~/TiposVer.aspx
	Productos por Tipo	Select	~/ProductosPorTipoVer.aspx
	Productos	Select	~/ProductosVer.aspx

Como puede apreciarse en el diseño de las páginas maestras, el control Label *lblDatosUsuarios*, que está situado en la esquina superior derecha, debe mostrar el valor almacenado en la **variable de sesión** *Nombre*. Esta variable de sesión almacena el valor del campo Login del usuario que haya iniciado la sesión y es accesible por todos los Web Form durante el tiempo que dure la sesión del usuario. Además, es necesario evitar los posibles accesos indeseados que pudieran producirse al introducir directamente la *url* de las páginas *.aspx* en la barra de exploración de los navegadores Web. Por ello, se va a añadir código en los procedimientos de evento *Load* de las páginas maestras, de modo que ese código afectará a todos los Web Forms con los que estén vinculadas. A continuación, añadir el siguiente código asociado al evento *Load* del archivo de código subyacente de la página maestra *MasterPageAdm.Master*.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Convert.ToString(Session["Rol"]) != "A")
    {
        Response.Redirect("~/Default.aspx");
    }
    lblDatosUsuario.Text = Convert.ToString(Session["Nombre"]);
}
```

Y el siguiente código asociado al evento *Load* de la página maestra *MasterPage.Master*.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Convert.ToString(Session["Rol"]) != "U")
    {
        Response.Redirect("~/Default.aspx");
    }
    lblDatosUsuario.Text = Convert.ToString(Session["Nombre"]);
}
```

9. Al hacer clic el control Button cuyo texto es *Cerrar sesión*, se eliminarán el estado de la sesión actual, se cancelará la sesión y se redirigirá hacia el Web Form de inicio de la Aplicación Web, *Default.aspx*. Para ello, añadir el siguiente código lógico asociado al evento *Click* del control Button denominado *btnCerrar* en el archivo de código subyacente, tanto en la página maestra *MasterPageAdm.Master* como en la página maestra *MasterPage.Master*.

```
protected void btnCerrar_Click(object sender, EventArgs e)
{
    Session.Clear();
    Session.Abandon();
    Response.Redirect("~/Default.aspx", false);
}
```

Hay que tener en cuenta que al añadir el código lógico anterior en el evento *Click* del control *btnCerrar* de ambas páginas maestras, este comportamiento afectará a todos los Web Forms vinculados con estas páginas maestras. De este modo, cualquier usuario autenticado podrá cerrar su sesión correctamente desde cualquier Web Form de la Aplicación Web.

10. Finalmente, para que las páginas no puedan quedar almacenadas en la memoria cache del servidor, conviene incluir la siguiente línea en el código de marcado de todos los Web Forms, justo detrás de la línea de directiva *@page*. La directiva *@OutputCache* configura la duración del tiempo de caducidad de la página en la memoria cache al valor de un segundo.

```
<%@ OutputCache Duration="1" VaryByParam="None" %>
```

En general, puesto que las aplicaciones Web trabajan con datos que representan la realidad del momento, que además es cambiante a lo largo del tiempo, no deberían utilizar memoria de cache de ningún tipo para garantizar la actualidad de la información que presentan en cada instante. Por lo que, se deberá incluir la línea de código anterior en todos los Web Forms que conformen la aplicación Web. Se utiliza el concepto informático de **consistencia de la información** para expresar esta idea de garantizar que la información almacenada en la base de datos deba representar, en todo momento, la realidad de la situación actual en el ámbito de gestión de la aplicación.

11. Iniciar la depuración para comprobar los resultados obtenidos. Realizar las pruebas de acceso a la Aplicación Web que sean necesarias, empleando usuarios de pertenecientes a los dos casos de uso o roles que están definidos en la tabla Usuario de la Base de Datos *Tienda.mdb*, es decir: el de perfil de administrador, accediendo mediante un usuario que pertenece al rol de Administrador (A); y, el perfil de usuario, accediendo mediante un usuario que pertenece al rol de Usuario (U). Podrá comprobarse que no se puede acceder a los Web Forms protegidos si el usuario no está autenticado desde la barra de direcciones del navegador Web. Sin embargo, sí es posible acceder a las páginas que se ha accedido anteriormente, a través del botón **Back** o **Atrás** del navegador Web. Las páginas Web almacenadas en el objeto *history* del navegador Web le pertenecen al usuario y, por tanto, no se suele contemplar la posibilidad de eliminar la cache desde las aplicaciones Web ni de desactivar el botón **Atrás**. Una solución es mostrar un mensaje, informando al usuario de la conveniencia de cerrar la ventana del navegador Web, cuando se accede a la página de inicio, después de cerrar la sesión. Otra posible solución es cerrar la ventana del navegador Web a través de un script de cliente al abandonar la sesión.



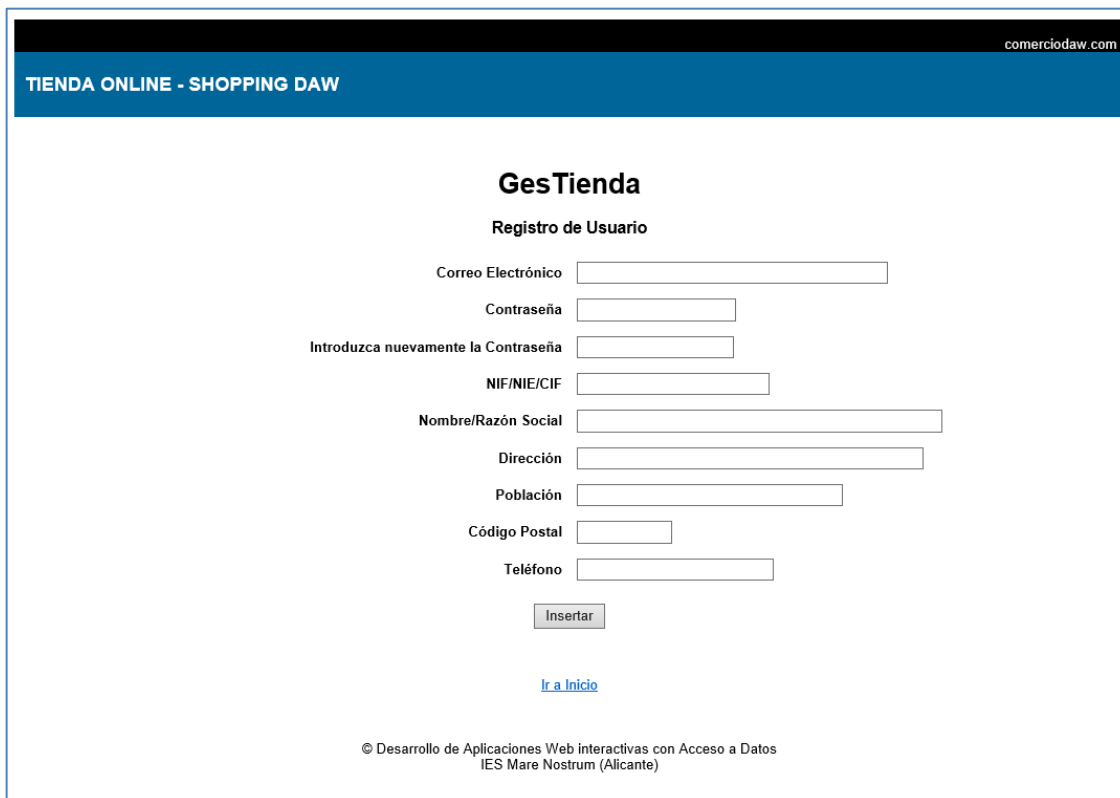
## Ejercicio 9

### Transacciones. Realizar operaciones sobre varias tablas.

En este ejercicio se creará un Web Form para poder registrarse como usuario de la aplicación Web. Este registro solo creará usuarios pertenecientes al rol de Usuarios. Un usuario perteneciente al rol de Administrador solo podrá ser creado por otro Administrador autenticado, por lo que requerirá de un procesamiento diferente. Atendiendo al modelo de datos establecido para la aplicación Web, cada vez que se registre un nuevo usuario deberá introducirse un conjunto de información que corresponde con diversos campos de la tabla USUARIO y de la tabla CLIENTE. Por lo tanto, cuando se registre un nuevo usuario deberán crearse dos nuevas filas, una fila en la tabla USUARIO y otra fila en la tabla CLIENTE. Por este motivo, es adecuado utilizar una transacción que atomice estas dos operaciones de inserción, de manera que, si se produce un error en alguna de ellas, entonces no se realice ninguna de las dos operaciones.

Para crear un Web Form para facilitar el registro de usuarios en la aplicación Web *GesTienda*, realizar las siguientes acciones:

1. Crear un nuevo Web Form denominado *Registrarse.aspx*. Este Web Form no se basará en ninguna página maestra.
2. Agregar los elementos y controles necesarios al Web Form *Registrarse.aspx* para obtener una apariencia similar a la siguiente.



comerciodaw.com

**TIENDA ONLINE - SHOPPING DAW**

**GesTienda**

**Registro de Usuario**

Correo Electrónico

Contraseña

Introduzca nuevamente la Contraseña

NIF/NIE/CIF

Nombre/Razón Social

Dirección

Población

Código Postal

Teléfono

[Ir a Inicio](#)

© Desarrollo de Aplicaciones Web interactivas con Acceso a Datos  
IES Mare Nostrum (Alicante)

- En la ilustración anterior puede observarse que en la parte inferior del Web Form se incorpora un control LinkButton para acceder al Web Form de inicio de la Aplicación Web, *Default.aspx*.
3. En el Web Form *Registrarse.aspx*, modificar el valor de la propiedad *TextMode* de los controles TextBox que acogerán los valores de la contraseña para asignarles el valor *Password*.
  4. En el Web Form *Registrarse.aspx*, agregar un control Label denominado *lblMensajes*.
  5. Añadir el siguiente código lógico asociado al evento *Click* del control Button que muestra el texto **Insertar**.

```
protected void btnInsertar_Click(object sender, EventArgs e)
{
    lblMensajes.Text = "";
    if (txtPassword1.Text == txtPassword2.Text)
    {
        string strLogin, strPassword, strRol, strFechaAlta;
        DateTime dtFechaAlta;
        string strIdCliente, strNomCli, strDirCli, strPobCli, strCpoCli, strTelCli, strCorCli;
        strLogin = txtCorCli.Text;
        strPassword = txtPassword1.Text;
        strRol = "U";
        dtFechaAlta = System.DateTime.Now; // Carga fecha y hora del sistema
        strFechaAlta = String.Format("{0:yyyy/MM/dd HH:mm:ss}", dtFechaAlta);
        strIdCliente = txtIdCliente.Text;
        strNomCli = txtNomCli.Text;
        strDirCli = txtDirCli.Text;
        strPobCli = txtPobCli.Text;
        strCpoCli = txtCpoCli.Text;
        strTelCli = txtTelCli.Text;
        strCorCli = txtCorCli.Text;

        string StrCadenaConexion =
            ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
        string strComandoSql_1 = "INSERT USUARIO " +
            "(Login,Password, Rol, FechaAlta) VALUES (" +
            "'" + strLogin + "','" + strPassword + "','" + strRol + "','" + strFechaAlta + "');"
        string strComandoSql_2 = "INSERT CLIENTE " +
            "(IdCliente,NomCli,DirCli,PobCli,CpoCli,TelCli,CorCli) VALUES (" +
            "'" + strIdCliente + "','" + strNomCli + "','" + strDirCli + "','" + strPobCli +
            "','" + strCpoCli + "','" + strTelCli + "','" + strCorCli + "');"

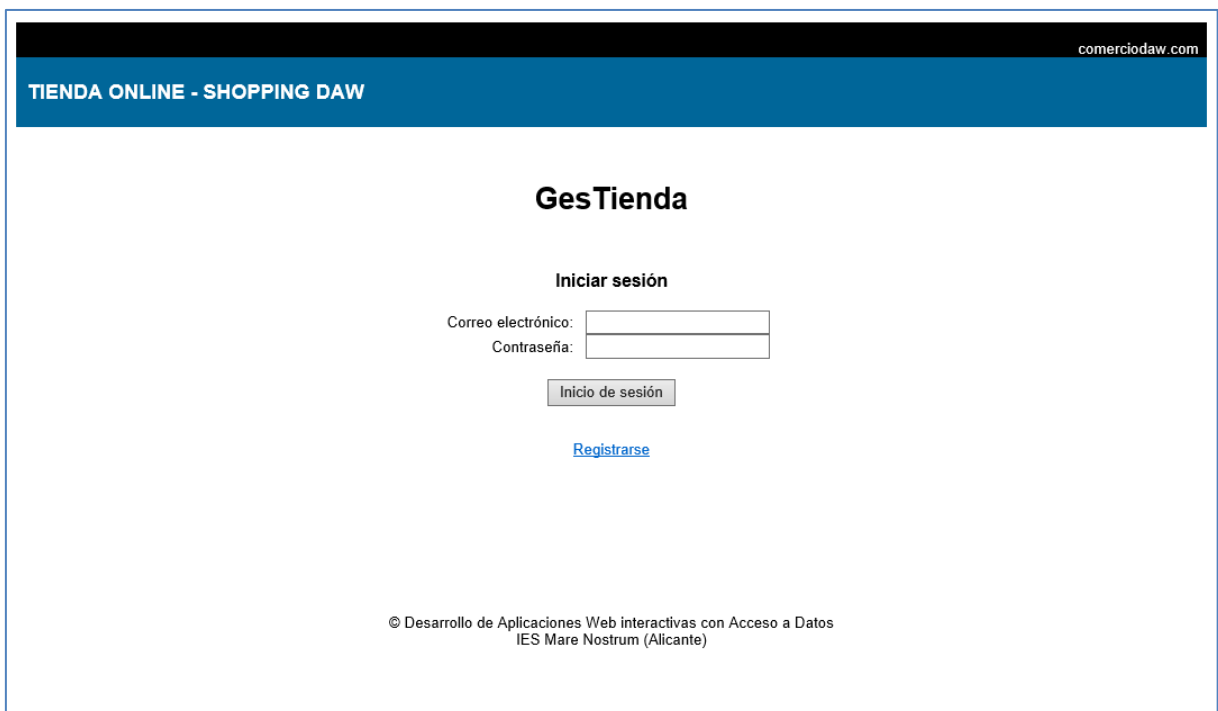
        SqlConnection conexion = new SqlConnection(StrCadenaConexion);
        conexion.Open();
        SqlCommand comando = new SqlCommand();
        comando.Connection = conexion;

        SqlTransaction tran = conexion.BeginTransaction();
        comando.Transaction = tran;
        try
        {
            comando.CommandText = strComandoSql_1;
            comando.ExecuteNonQuery();
            comando.CommandText = strComandoSql_2;
            comando.ExecuteNonQuery();
            tran.Commit();
            lblMensajes.Text = "Usuario registrado correctamente";
        }
        catch (SqlException ex)
        {
            tran.Rollback();
            string StrError = "<p>Se han producido errores durante el registro</p>";
            StrError = StrError + "<div>Código: " + ex.Number + "</div>";
            StrError = StrError + "<div>Descripción: " + ex.Message + "</div>";
            lblMensajes.Text = StrError;
        }
    }
}
```

```
}  
    finally  
    {  
        conexion.Close();  
    }  
}  
else  
{  
    lblMensajes.Text = "Se ha producido un error. Valores de contraseña no coincidentes";  
}  
}
```

En el código anterior, puede apreciarse la preparación de las dos sentencias SQL de tipo INSERT sobre las tablas USUARIO y CLIENTE. También puede apreciarse, el tratamiento que se hace de la transacción que atomiza estas dos sentencias, así como el uso de los métodos **Commit()** y **Rollback()** para afirmar y deshacer la transacción, respectivamente.

6. Iniciar depuración para comprobar resultados obtenidos. Realizar pruebas para insertar los datos de un usuario y un cliente inexistentes para comprobar que se insertan los datos correctamente en la base de datos. Realizar las pruebas necesarias para tratar de insertar los datos de un usuario ya existente, en primer lugar, y de un cliente ya existente, en segundo lugar, para comprobar que no se realiza ninguna acción sobre la base de datos.
7. Agregar un control LinkButton en el Web Form *Default.aspx* para acceder al Web Form *Registrarse.aspx*, tal como se muestra a continuación:



8. Iniciar depuración para comprobar resultados obtenidos.