

Prácticas

Tema 6 – Técnicas de acceso a datos (II)

Objetivos	<p>En esta práctica, se aborda la realización de tareas relacionadas con:</p> <ul style="list-style-type: none"> • Agregar y conectar un archivo de base de datos de SQL Server a un sitio Web de ASP.NET • 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 a través de objetos de ADO.NET • Utilizar el modelo de acceso a datos mediante código para la presentación e interacción con la información recuperada de la base de datos a través de controles de ASP.NET sobre interfaz Web • Utilizar los objetos de ASP.NET • Aplicar diversos aspectos relevantes en relación con el desarrollo de aplicaciones para la Web • Conocer el funcionamiento 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 • Emplear variables de sesión para compartir datos entre los Web Forms que conforman la aplicación Web durante el período de la sesión de usuario
Conocimientos previos	<p>Para realizar esta práctica, es necesario tener conocimientos sobre:</p> <ul style="list-style-type: none"> • Bases de datos relacionales • Modelos lógicos de datos en aplicaciones informáticas • Utilización de controles de servidor en páginas Web de ASP.NET • Programación lógica de páginas dinámicas de servidor empleando archivos de código subyacente • Lenguaje SQL para la manipulación, definición y control de datos
Escenario	<p>Se presentan varios ejercicios dirigidos a comprender diversos aspectos prácticos sobre el desarrollo de aplicaciones para la Web con acceso a datos mediante ASP.NET a través del acceso al archivo de base de datos de SQL Server, denominado <i>Tienda.mdf</i>.</p> <p>Para la realización de esta práctica, se proporciona como recursos:</p> <ul style="list-style-type: none"> • El archivo de base de datos de SQL Server <i>Tienda.mdf</i> • Una plantilla Web formada por un Web Form básico de presentación, denominada <i>WebFormPlantilla.aspx</i> junto con el archivo de código subyacente asociado <i>WebFormPlantilla.aspx.cs</i>, y por el archivo de hoja de estilo correspondiente, <i>HojaEstilo.css</i>.

Ejercicio 1

Creación y preparación de un sitio Web

En este primer ejercicio se creará una nueva Solución Visual Studio .NET, denominada *Tienda*, y un nuevo Sitio Web de ASP.NET, denominado *GesTienda*, para desarrollar los ejercicios contenidos en esta práctica. A continuación, se agregará al sitio Web el archivo de base de datos de SQL Server *Tienda.mdf* que al almacenar la información que manejará el sitio Web creado se constituye como el almacén de datos subyacente de la aplicación Web. Finalmente, se realizarán diversas tareas que permitirán establecer la apariencia de los Web Forms que compondrán el sitio Web de ASP.NET.

Crear una nueva Solución y un nuevo Sitio Web

En primer lugar, se creará una nueva *Solución de Visual Studio .NET*, denominada *Tienda*, y un nuevo Sitio Web para Visual C# denominado, *GesTienda*. Para ello realizar las siguientes acciones:

1. Para crear una nueva Solución de Visual Studio .NET, denominada *Tienda*, acceder al menú **Archivo** de Visual Studio .NET, expandir la opción **Nuevo** y seleccionar **Proyecto...**
2. En el cuadro de diálogo **Nuevo Proyecto**, realizar las siguientes acciones:
 - a. Expandir la opción **Otros tipos de proyecto**, seleccionar **Soluciones de Visual Studio** y hacer clic en **Solución en blanco**.
 - b. En el cuadro de texto **Nombre** introducir *Tienda*.
 - c. Seleccionar una carpeta como ubicación para la solución de Visual Studio mediante el botón **Examinar...** y hacer clic en **Aceptar**. Puede comprobarse que se habrá creado la carpeta *Tienda* en la ubicación seleccionada y en su interior el archivo de Solución de Visual Studio *Tienda.sln*.
3. Para crear un nuevo Sitio Web de ASP.NET, denominado *GesTienda*, en la Solución *Tienda*, acceder al **Explorador de Soluciones**, hacer clic con el botón derecho sobre el nombre de la Solución *Tienda*, expandir la opción **Agregar** y seleccionar **Nuevo Sitio Web...**
4. En el cuadro de diálogo **Agregar nuevo Sitio Web**, realizar las siguientes acciones:
 - a. En la lista **Plantillas Instaladas**, seleccionar **Visual C#** como lenguaje de programación a emplear en el nuevo sitio Web.
 - b. Seleccionar **Sitio Web vacío de ASP.NET**.
 - c. En la sección **Ubicación Web**, seleccionar la opción **Sistema de archivos** y escribir el nombre de la carpeta en la que desea guardar los archivos asociados al nuevo sitio Web. Para mantener un control sobre los archivos que componen tanto la solución como el nuevo sitio Web de ASP.NET, se recomienda crear la carpeta del nuevo sitio Web en la misma ubicación del sistema de archivos donde se encuentra el archivo de solución de Visual Studio *Tienda.sln*. Para seleccionar la ubicación se puede utilizar el botón **Examinar**. Una vez introducido el nombre del nuevo sitio Web, *GesTienda* formando parte de la ruta de la ubicación deseada, hacer clic en **Aceptar**.
5. Utilizando el **Explorador de archivos de Windows**, comprobar que se ha creado la carpeta *Tienda* en la ubicación deseada y que en su contenido se encuentra el archivo de solución *Tienda.sln* y la carpeta de archivos del sitio Web *GesTienda*, que a su vez contendrá los archivos propios de un sitio Web de ASP.NET.

Agregar un archivo de base de datos de SQL Server a un Sitio Web

Una vez creada la solución *Tienda* y el proyecto de sitio Web *GesTienda*, se agregará el archivo de base de datos de SQL Server, *Tienda.mdf*, que almacena la información que se manejará en la aplicación Web a través de los controles que se incluyan en los Web Forms.

1. Para agregar al sitio Web *GesTienda* el archivo de base de datos de SQL Server proporcionado por el profesor realizar las siguientes acciones:
 - a. Crear la carpeta *App_Data* en el sitio Web. Para ello, acceder al **Explorador de soluciones**, hacer clic con el botón derecho sobre el proyecto *GesTienda*, seleccionar la opción **Agregar carpeta ASP.NET** y seleccionar la opción *App_Data*.
 - b. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre la carpeta *App_Data* y seleccionar la opción **Agregar 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 *Tienda.mdf* aparecerá añadido al sitio Web contenido en la carpeta *App_Data*. Mediante el **Explorador de archivos** de **Windows**, puede comprobarse que se ha realizado una copia del archivo *Tienda.mdf* en la carpeta *App_Data* del sitio Web.
2. Acceder al **Explorador de servidores**, para comprobar que se ha creado la conexión del sitio Web *GesTienda* con el archivo de base de datos de SQL Server *Tienda.mdf*. Para ello:
 - a. En el **Explorador de servidores**, desplegar las **Conexiones de datos** existentes haciendo clic sobre la fecha que aparece a su izquierda.
 - b. Comprobar que se ha creado una conexión de datos cuyo nombre será *Tienda.mdf*.
 - c. 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.
3. Comprobar que la conexión al archivo de base de datos de SQL Server *Tienda.mdf* desde el sitio Web *GesTienda* de la Solución *Tienda* de ASP.NET funciona correctamente, haciendo:
 - 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 de diálogo informando sobre si el funcionamiento de la conexión es o no correcto.

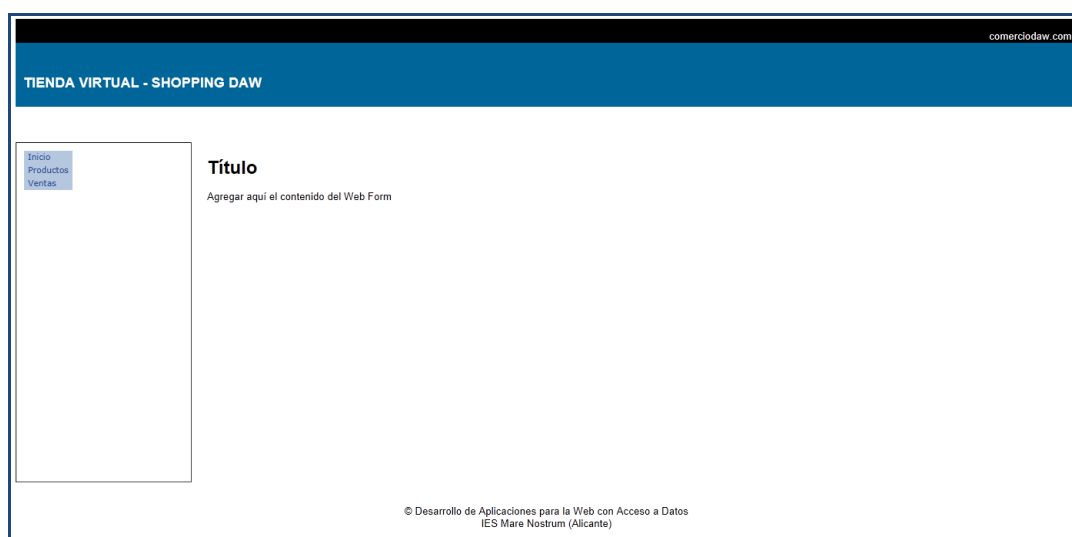
Agregar una plantilla Web a un Sitio Web

Se ha preparado una plantilla Web, formada por el Web Form *WebFormPlantilla.aspx* y el archivo de hoja de estilo externa *HojaEstilo.css*, para mantener una presentación y apariencia global común en los diferentes Web Forms que formarán la aplicación Web.

Cada uno de los Web Forms que se precisen crear para la resolución de esta práctica, y más generalizadamente para completar el sitio Web, se creará realizando una copia del archivo *WebFormPlantilla.aspx* sobre otro nuevo archivo. El desarrollo del nuevo Web Form se realizará realizando las modificaciones oportunas sobre el nuevo archivo obtenido mediante la copia del archivo *WebFormPlantilla.aspx*.

A continuación, se describe cómo agregar el Web Form *WebFormPlantilla.aspx* y el archivo de hoja de estilo externa *HojaEstilo.css* al sitio Web *GesTienda*.

1. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre del sitio Web *GesTienda* y seleccionar la opción **Agregar elemento existente...**
2. En el cuadro de diálogo **Agregar elemento existente**, seleccionar el archivo *WebFormPlantilla.aspx* desde su ubicación y hacer clic en **Agregar**. El archivo aparecerá añadido al sitio Web *GesTienda*. Mediante el **Explorador de archivos** de **Windows**, puede comprobarse que se ha realizado una copia del archivo *WebFormPlantilla.aspx* en la carpeta del sitio Web *GesTienda*.
3. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre del sitio Web *GesTienda* y seleccionar la opción **Nueva carpeta**. Modificar el nombre de la nueva para denominarla *Estilos*.
4. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre la nueva carpeta creada, denominada *Estilos*, y seleccionar la opción **Agregar 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* del sitio Web *GesTienda*. Mediante el **Explorador de archivos** de **Windows**, puede comprobarse que se ha realizado una copia del archivo *WebFormPlantilla.aspx* en la carpeta del sitio Web *GesTienda*.
6. Abrir el Web Form *WebFormPlantilla.aspx* en modo **Diseño** para comprobar que el resultado es el correcto. La asociación entre el Web Form y el archivo de hoja de estilo externa habrá quedado realizada. Si fuera necesario, asociarlos de la siguiente manera:
 - a. Abrir el Web Form a asociar a una hoja de estilo en modo **Diseño**.
 - b. Para asociar un Web Form a un archivo de hoja de estilo externa se arrastra el archivo de hoja de estilo desde el **Explorador de soluciones** hasta el Web Form.



Para conocer la composición de los diversos elementos HTML y controles ASP.NET que conforman el Web Form *WebFormPlantilla.aspx*, que actuará de base para el resto de Web Forms del sitio Web, se recomienda abrirlo en modo **Código** y tratar de identificar los elementos y controles incluidos.

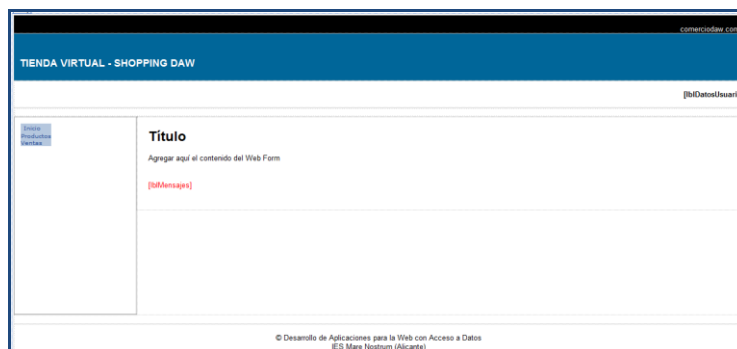
Ejercicio 2

Crear un Web Form para consulta de datos mediante código

Una vez creado la solución *Tienda* y el sitio Web *GesTienda* y, establecida una conexión con la base de datos subyacente de la aplicación Web, *Tienda.mdf*, se creará un Web Form de nombre *ProductosVer.aspx* para consulta de datos, empleando 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 el Web Form *ProductosVer.aspx* a través de una copia de *WebFormPlantilla.aspx*, realizando las siguientes acciones:
 - a. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre el Web Form *WebFormPlantilla.aspx* y seleccionar la opción **Copiar**.
 - b. En el **Explorador de soluciones**, hacer clic con el botón derecho sobre el nombre del sitio Web *GesTienda* y seleccionar la opción **Pegar**.
 - c. Modificar el nombre del nuevo Web Form creado, *Copia de WebFormPlantilla.aspx*, por el nombre *ProductosVer.aspx*. Para ello, en el **Explorador de soluciones** hacer clic en el botón derecho sobre el Web Form *Copia de WebFormPlantilla.aspx* y seleccionar la opción **Cambiar nombre**.



2. Abrir el Web Form *ProductosVer.aspx*, y sobre el elemento `<div>` denominado *contenido*:
 - a. Modificar el contenido del elemento `<div>` denominado *contenidotitulo*, incluido en el elemento `<div>` denominado *contenido*, para introducir la frase *Ver Productos*.
 - b. Agregar un nuevo control *Label*, denominado *lblResultado*, que servirá para mostrar el resultado de la consulta a realizar a la base de datos.
3. En el control *Menu* incluido en el elemento `<div>` denominado *menu*, modificar la propiedad *NavigateUrl*, asociada a la opción de segundo nivel denominada *Ver productos* incluida en la opción de primer nivel *Productos*, para facilitar el acceso al Web Form *ProductosVer.aspx*.
4. Abrir el archivo de código subyacente *ProductosVer.aspx.cs*.
 - a. Al final de la sección *using*, añadir la siguiente línea para emplear en el código las clases contenidas en el namespaces *System.Data.SqlClient*. El espacio de nombres *System.Data.SqlClient* facilita el uso de las clases que representan la arquitectura ADO.NET para el acceso a bases de datos de Microsoft SQL Server.

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

- b. Añadir el siguiente código asociado al evento *Load* del objeto *Page*.

```
protected void Page_Load(object sender, EventArgs e)
{
    int InNumeroFilas;
    string StrResultado, StrError;
    string StrCadenaConexion = "Data Source=(LocalDB)\\v11.0;AttachDbFilename=" +
        Server.MapPath("~/App_Data/Tienda.mdf") +
        ";Integrated Security=True;Connect Timeout=30";
    //string StrComandoSql = "SELECT * FROM PRODUCTO;";
    //string StrComandoSql = "SELECT IdProducto,DesPro,PrePro,IdUnidad,DesTip " +
    //    "FROM PRODUCTO,TIPO WHERE PRODUCTO.IdTipo = TIPO.IdTipo;";
    string StrComandoSql = "SELECT IdProducto,DesPro,PrePro,IdUnidad,DesTip FROM PRODUCTO " +
        "INNER JOIN TIPO ON PRODUCTO.IdTipo = TIPO.IdTipo;";
    try {
        SqlConnection conexion = new SqlConnection(StrCadenaConexion);
        SqlCommand comando = new SqlCommand(StrComandoSql, conexion);
        conexion.Open();
        SqlDataReader reader = comando.ExecuteReader();
        if (reader.HasRows) {
            StrResultado = "<div style='display:table; border-style:solid;border-color:#336699'>";
            StrResultado += "<div style='display:table-row; background:#336699;color:white'>";
            StrResultado += "<div style='display:table-cell; font-weight:bold'>Código</div>";
            StrResultado += "<div style='display:table-cell; font-weight:bold'>Descripción</div>";
            StrResultado += "<div style='display:table-cell; font-weight:bold'>Precio</div>";
            StrResultado += "<div style='display:table-cell; font-weight:bold'>Unidad</div>";
            StrResultado += "<div style='display:table-cell; font-weight:bold'>Tipo Producto</div>";
            StrResultado += "</div>";
            InNumeroFilas = 0;
            while (reader.Read()) {
                StrResultado += "<div style='display:table-row'>";
                StrResultado += "<div style='display:table-cell'>" + reader.GetString(0) + "</div>";
                StrResultado += "<div style='display:table-cell'>" + reader.GetString(1) + "</div>";
                StrResultado += "<div style='display:table-cell; text-align: right'>"
                    + string.Format("{0:C}",reader.GetValue(2)) + "&nbsp; &nbsp; </div>";
                StrResultado += "<div style='display:table-cell'>" + reader.GetString(3) + "</div>";
                StrResultado += "<div style='display:table-cell'>" + reader.GetString(4) + "</div>";
                StrResultado += "</div>";
                InNumeroFilas++;
            }
            StrResultado += "</div>";
            StrResultado += "<p> Número de Filas: " + InNumeroFilas + "</p>";
            lblResultado.Text = StrResultado;
        }
        else {
            lblMensajes.Text = "No existen registros resultantes de la consulta";
        }
        reader.Close();
        comando.Dispose();
        conexion.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;
    }
}
```

Se recomienda revisar y analizar el código con detalle con la finalidad de comprender exactamente el sentido de cada una de las líneas de código empleadas para acceder a la información de la base de datos a través del objeto *SqlDataReader* de ADO.NET.

5. Iniciar la depuración para comprobar los resultados obtenidos.

comerciadow.com

TIENDA VIRTUAL - SHOPPING DAW

- Inicio
- Productos
- Compras
- Usuarios

Ver 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 señas y caballero
4500-450	Zapatos caballero Mod. Ibiza	30,00 €	Par	Zapatos de señas 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	Sandias	2,50 €	Kilogramo	Alimentación y bebidas

Número de Filas: 14

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

Ejercicio 3

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, tales como un control *GridView*, con controles simples cuyos contenidos de datos se obtienen a través de código que utiliza objetos de acceso a datos de ADO.NET.

Crear un Web Form complejo de consulta

1. Crear el Web Form *PedidosPorCliente.aspx* a través de una copia de *WebFormPlantilla.aspx*.
2. Abrir el Web Form *PedidosPorCliente.aspx*, y modificar el contenido del elemento `<div>` denominado *contenidotitulo*, para introducir la frase *Pedidos realizados por los clientes*.
3. 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 maestra *CLIENTE*. En este control de datos *GridView* se mostrarán los siguientes datos: código del cliente, nombre, población, correo electrónico y login. Habilitar la selección de filas en el *GridView*.
4. Se pretende que al seleccionar un cliente sobre el control de datos *GridView*, se muestre información sobre los pedidos realizados por ese cliente. Para ello, agregar un nuevo control *Label*, denominado *lblResultado*, para mostrar la información resultante de la consulta a realizar a la base de datos. Y, agregar otro nuevo control *Label*, denominado *lblTotal*, para mostrar el número de pedidos realizados y el importe total de los pedidos realizados por el cliente seleccionado. A ambos controles asignarles *False* a la propiedad *Visible*.
5. Ajustar las propiedades de los controles del Web Form, para que su presentación sea similar a la que se muestra en la siguiente ilustración.

comerciodaw.com

TIENDA VIRTUAL - SHOPPING DAW

Inicio
Productos
Ventas
Clientes

Pedidos realizados por los clientes

	Id. Cliente	Nombre	Población	Correo electrónico	Login
Seleccionar	00123659Z	Beatriz Iraola López	Bilbao	beatriz@gmail.es	beatriz
Seleccionar	07899905V	Jesús García Soria	Zaragoza	jesus@hotmail.es	suso
Seleccionar	08659442S	Miguel Sogorb Fenoll	Alicante	miguel@alc.es	msogorb
Seleccionar	09897907K	Alvaro Fernández Moreno	Madrid	alvaro@yahoo.es	afm
Seleccionar	10900801K	Alejandro Castro Hernández	Zamora	alex@gmail.es	acastro
Siguiente Último					

Núm. Pedido	Fecha	Servido	Cobrado	Total
2	03/01/2012	Si	No	1.192,10 €
4	04/01/2012	Si	Si	10,00 €
8	07/01/2012	No	No	50,00 €

Número pedidos realizados: 3
Importe total de los pedidos realizados por el cliente: 1.252,10 €

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

6. Para obtener y presentar los datos de los pedidos y del total general, añadir el siguiente código asociado al evento *SelectedIndexChanged* del objeto *grdClientes*.

```
protected void grdClientes_SelectedIndexChanged(object sender, EventArgs e)
{
    int InNumeroFilas;
    string StrResultado, StrError;
    string StrCadenaConexion = "Data Source=(LocalDB)\\v11.0;AttachDbFilename=" +
        Server.MapPath("~/App_Data/Tienda.mdf") +
        ";Integrated Security=True;Connect Timeout=30";
    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;
    try {
        SqlConnection conexion = new SqlConnection(StrCadenaConexion);
        SqlCommand comando = new SqlCommand(StrComandoSql, conexion);
        conexion.Open();
        SqlDataReader reader = comando.ExecuteReader();
        if (reader.HasRows) {
            InNumeroFilas = 0;
            lblResultado.Visible = true;
            lblTotal.Visible = true;
            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;
            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 + "')";
            SqlConnection conexion1 = new SqlConnection(StrCadenaConexion);
            conexion1.Open();
            SqlCommand comando1 = new SqlCommand(StrComandoSql1, conexion1);
```

```

DcTotal = Convert.ToDecimal(comando1.ExecuteScalar());
comando1.Dispose();
conexion1.Close();
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();
comando.Dispose();
conexion.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;
}
}

```

- Además, para evitar que al hacer clic sobre los elementos de navegación del control *GridView* para cambiar de página activa siga apareciendo información correspondiente a los pedidos de otro 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 *gridVlew* establece que no se selecciona ninguna fila.

- En el control *Menu* incluido en el elemento *<div>* denominado *menu*, editar los elementos del menú para agregar un nuevo elemento de menú de segundo nivel denominada *Pedidos por cliente* asociado al elemento de menú de primer nivel denominado *Ventas*. Modificar la propiedad *NavigateUrl* del nuevo elemento agregado para facilitar el acceso al Web Form *PedidosPorCliente.aspx*.
- Iniciar la depuración para comprobar los resultados obtenidos.

Puesto que se van incluyendo controles de datos en los Web Forms de la aplicación Web, sería conveniente introducir un control de errores a nivel de aplicación, tal como se hizo en la práctica anterior. Como los objetivos de esta práctica están relacionados con el modelo de acceso a datos mediante código, el ejercicio de creación de un Web Form para mostrar los errores y el control de errores a nivel de aplicación se deja como ejercicio opcional.

Se recomienda revisar y analizar el código lógico de los diferentes ejercicios de forma detallada, con la finalidad de comprender de forma precisa el sentido de cada una de las líneas de código empleadas para acceder a la base de datos y para generar dinámicamente la página de respuesta.

Ejercicio 4

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 insertar un nuevo producto y, eliminar los datos o actualizar los valores de los campos de un producto existente. Para realizar las operaciones de eliminación y actualización se seleccionará, previamente, los datos de un producto de la tabla.

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

1. Crear el Web Form *ProductosMantener.aspx* a través de la copia de *WebFormPlantilla.aspx*.
2. Abrir el Web Form *ProductosMantener.aspx*, y modificar el contenido del elemento `<div>` denominado *contenidotitulo*, para introducir la frase *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. Habilitar la selección de filas en el *GridView*.
4. Iniciar la depuración para comprobar los resultados obtenidos hasta el momento.
5. Agregar un control *Label* y un *TextBox* o un *DropDownList*, según el caso, para mostrar una descripción y el valor de todos y cada uno de los campos de la tabla *PRODUCTO*. Modificar las propiedades de los controles agregados del siguiente de modo:

Tipo de control	Asociado al campo	Id	Text	Enabled
Label		lblIdProducto	Id. Producto	
TextBox	<i>IdProducto</i>	txtIdProducto		<i>False</i>
Label		lblDesPro	Descripción	
TextBox	<i>DesPro</i>	txtDesPro		<i>False</i>
Label		lblPrePro	Precio	
TextBox	<i>PrePro</i>	txtPrePro	0	<i>False</i>
Label		lblIdUnidad	Unidad	
DropDownList	<i>IdUnidad</i>	ddlIdUnidad		<i>False</i>
Label		lblIdTipo	Tipo Producto	
DropDownList	<i>IdTipo</i>	ddlIdTipo		<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 procesos para la inserción de un nuevo registro y la actualización o eliminación de un registro seleccionado mediante el control *GridView*. A cada uno de ellos, asignarles los valores de las propiedades que se muestran en la siguiente tabla:

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

9. En el control *Menu* incluido en el elemento `<div>` denominado *menu*, editar los elementos del menú para agregar una nueva opción de segundo nivel denominada *Mantenimiento Productos* asociada a la opción *Productos*. Modificar la propiedad *NavigateUrl* del nuevo elemento agregado para facilitar el acceso al Web Form *ProductosMantener.aspx*.
10. Iniciar la depuración para comprobar los resultados obtenidos hasta el momento.
11. Ajustar los valores de las propiedades de los controles ASP.NET y elementos HTML añadidos al Web Form, de modo que la presentación sea similar a la siguiente.

Añadir código al Web Form para el mantenimiento de datos

Una vez construida la presentación del Web Form *ProductosMantener.aspx*, agregando los controles simples, controles de datos de ASP.NET y elementos de HTML necesarios, se aborda la construcción del código lógico de procesamiento en el servidor Web que permitirá el procesamiento de las opciones de mantenimiento de datos mediante el uso de controles simples.

1. Para mostrar en los controles simples 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 = "Data Source=(LocalDB)\\v11.0;AttachDbFilename=" +
        Server.MapPath("~/App_Data/Tienda.mdf") +
        ";Integrated Security=True;Connect Timeout=30";
    string StrComandoSql = "SELECT IdProducto, DesPro, PrePro, IdUnidad, PRODUCTO.IdTipo, DesTip " +
        " FROM PRODUCTO INNER JOIN TIPO ON PRODUCTO.IdTipo = TIPO.IdTipo " +
        "WHERE PRODUCTO.IdProducto = '" + StrIdProducto + "'";
    try
    {
        SqlConnection conexion = new SqlConnection(StrCadenaConexion);
        SqlCommand comando = new SqlCommand(StrComandoSql, conexion);
        conexion.Open();
        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();
        comando.Dispose();
        conexion.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;
    }
}
```

2. Añadir las siguientes funciones al archivo de código subyacente *ProductosMantener.aspx.cs*, que facilitaran la habilitación y deshabilitación de 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 de los controles para insertar o modificar filas.

La función para deshabilitar los controles simples:

```
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:

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

3. 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 el control *GridView* en los controles simples que presentan los datos situados hacia la derecha del Web Form.
4. Para insertar nuevas filas en la tabla *PRODUCTO*, en primer lugar, añadir el siguiente código asociado al evento *Click* del objeto *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();
    ddlIdTipo.DataBind();
    grdProductos.SelectedIndex = -1;
    FnHabilitarControles();
    txtIdProducto.Focus();
}
```

A continuación, añadir el siguiente código asociado al evento *Click* del objeto *btnInsertar* para realizar el procesamiento de inserción del registro que se está editando en los controles simples del Web Form. Para ello, se incluye una sentencia de manipulación de datos de SQL de tipo INSERT en el código lógico. Esta sentencia INSERT producirá el almacenamiento de los valores del registro editado empleando los controles simples del Web Form en la tabla *PRODUCTO* de la base de datos subyacente del sitio Web.

```
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 = "Data Source=(LocalDB)\\v11.0;AttachDbFilename=" +
        Server.MapPath("~/App_Data/Tienda.mdf") + ";Integrated Security=True;Connect Timeout=30";
    string StrComandoSql = "INSERT PRODUCTO " +
        "(IdProducto,DesPro,PrePro,IdUnidad,IdTipo) VALUES (" +
        "'" + strIdProducto + "','" + strDescripcion +
        "','" + FnComaPorPunto(dcPrecio) +
        "','" + strIdUnidad + "','" + strIdTipo + "');"

    try
    {
        SqlConnection conexion = new SqlConnection(StrCadenaConexion);
        SqlCommand comando = new SqlCommand(StrComandoSql, conexion);
        comando.Connection.Open();
        Int32 inRegistrosAfectados = comando.ExecuteNonQuery();
        comando.Connection.Close();
        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();
    grdProductos.SelectedIndex = -1;
    FnDeshabilitarControles();
}
```

En la función de evento anterior se utiliza la función *FnComaPorPunto()* para agregar el valor del precio a la cadena que conforma 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);
}
```

Para finalizar el desarrollo del proceso de inserción de nuevas filas en la tabla *PRODUCTO*, añadir el siguiente código asociado al evento *Click* del objeto *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();
}
```

5. Desarrollar el código para editar los valores de los campos de la fila seleccionada mediante el control de datos *GridView* y actualizar la fila correspondiente. Para ello, se añadirá el código en el evento *Click* del objeto *btnEditar*, y en el evento *Click* del objeto *btnModificar*. El proceso de modificación de los valores de los campos de una fila de la tabla *PRODUCTO*, se basa en la definición de una sentencia de SQL de tipo *UPDATE* que se ejecutará al invocar el método *ExecuteNonQuery()* de la instancia al objeto *Command* establecida en el código, de forma similar a como se ha desarrollado en el código correspondiente a la inserción de filas.
6. Desarrollar el código para eliminar la fila de la tabla *PRODUCTO* seleccionada mediante el control de datos *GridView*. Antes de la eliminación se pedirá confirmación. Para ello, se añadirá el código en el evento *Click* del objeto *btnEliminar*, y en el evento *Click* del objeto *btnBorrar*. Al igual que en el caso de la inserción y de la modificación de registros se empleará el método *ExecuteNonQuery()* de la instancia al objeto *Command* establecida en el código, para ejecutar una sentencia de SQL de tipo *DELETE*. A criterio del alumno, se podrán definir las funciones que se consideren oportunas para encapsular funcionalidades que resuelvan tareas genéricas o comunes para la inserción, modificación y eliminación de filas.
7. Iniciar la depuración para comprobar que el funcionamiento de los procesos de manipulación de datos y que el comportamiento de la interfaz es adecuado.

Ejercicio 5

Gestionar el acceso de los usuarios a la aplicación Web

En este ejercicio se incluirá al sitio Web *GesTienda* un sistema de autenticación de usuarios que permita iniciar la sesión a los distintos usuarios de la aplicación Web y evite el acceso de los usuarios no autorizados en todas las páginas de la aplicación. Para ello, se empleará el control *Login*, aunque también podría resolverse empleando objetos simples, y la información de la tabla *USUARIO* de la base de datos que almacena la información correspondiente a los usuarios autenticados con su correspondiente rol o perfil de acceso a la aplicación Web.

El control *Login* está formado por un conjunto de campos y funciones que facilitan las tareas relacionadas con el inicio de sesión de las aplicaciones Web. Este control permite desglosar sus partes para poder establecer el formato más adecuado. Así, es posible editar sus campos para que incluya, además de cuadros de texto para la introducción del nombre de usuario y la contraseña, enlaces a Web Forms que faciliten el registro de un nuevo usuario, el cambio de la contraseña de un usuario existente, etc. El control *Login* quedará agregado en el Web Form *Default.aspx* que gestionará el acceso al sitio Web de los distintos usuarios y administradores de la aplicación Web. Para ello:

1. Crear el Web Form de inicio de la aplicación, denominado *Default.aspx*, en base a una copia del Web Form que actúa como plantilla de presentación, *WebFormPlantilla.aspx*.
2. Abrir el Web Form *Default.aspx* en modo **Código** y eliminar todo el contenido existente en el elemento `<div>` denominado *cuerpo*, excepto la el control *Label* denominado *lblMensajes*.
3. Desde el nodo **Inicio de sesión** del **Cuadro de herramientas**, agregar un control *Login* en el contenido del elemento `<div>` denominado *cuerpo* y realizar las siguientes acciones:
 - a. Sobre el control *Login* insertado, *Login1*, hacer clic en el botón derecho y seleccionar la opción **Convertir en plantilla** para poder establecerle la forma deseada.
 - 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 *Login* sobre el contenido de la página.
 - e. Darle el formato de apariencia que parezca más adecuado.
4. Modificar el Web Form *Default.aspx* para obtener una apariencia similar a la siguiente:

- En el Web Form *Default.aspx*, hacer doble clic sobre el control *Login* para acceder a su procedimiento de evento *Authenticate* en el archivo *Default.aspx.cs*, y añadir el siguiente código para establecer el control de accesos que se activará al intentar iniciar la sesión:

```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
{
    string StrCadenaConexion = "Data Source=(LocalDB)\\v11.0;AttachDbFilename=" +
        Server.MapPath("~/App_Data/Tienda.mdf") +
        ";Integrated Security=True;Connect Timeout=30";
    string StrComandoSql = "SELECT nombre, rol FROM USUARIO ";
    StrComandoSql = StrComandoSql + " WHERE Login='" + Login1.UserName + "' ";
    StrComandoSql = StrComandoSql + "AND Password='" + Login1.Password + "'";
    try
    {
        SqlConnection conexion = new SqlConnection(StrCadenaConexion);
        SqlCommand comando = new SqlCommand(StrComandoSql, conexion);
        conexion.Open();
        SqlDataReader reader = comando.ExecuteReader();
        if (reader.Read())
        {
            Session.Add("Nombre", reader.GetString(0));
            Session.Add("Rol", reader.GetString(1));
            e.Authenticated = true;
            reader.Close();
            comando.Dispose();
            conexion.Close();
            if (Convert.ToString(Session["Rol"]) == "A")
                Response.Redirect("~/MenuAdmin.aspx");
            if (Convert.ToString(Session["Rol"]) == "U")
                Response.Redirect("~/MenuUsuario.aspx");
        }
        else
        {
            e.Authenticated = false;
            reader.Close();
            comando.Dispose();
            conexion.Close();
        }
    }
    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 el siguiente código de marcado al archivo de configuración de la aplicación *Web.config*, para deshabilitar la validación no-intrusiva basada en jquery que se encuentra habilitada de manera predeterminada en ASP.NET a partir de .NET framework 4.5.

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

- Iniciar la depuración para comprobar los resultados obtenidos. Se recomienda ver los datos almacenados de la tabla *USUARIO* antes de intentar el acceso para probar con usuarios existentes, con claves erróneas y con usuarios inexistentes.

8. Crear los Web Forms *MenuAdmin.aspx* y *MenuUsuario.aspx* que podrán contener las opciones de menú adecuadas según sea el rol del usuario que ha iniciado su sesión, Administrador (A) o Usuario (U), respectivamente. La apariencia de estos Web Forms, que se basa en la plantilla utilizada para representar la apariencia de todos los Web Forms de la aplicación, puede quedar tal como se muestra a continuación. El perfil de administrador de la aplicación Web accederá al Web Form *MenuAdmin.aspx* que tendrá una apariencia similar a la siguiente:



El perfil de usuario de la aplicación Web accederá al Web Form *MenuUsuario.aspx* que tendrá una apariencia similar a la siguiente:



Las opciones de menú a incluir en el control *Menu* de cada uno de estos Web Forms y su enlace asociado, se muestran en la siguiente tabla:

Nombre del Web Form	Opciones de primer nivel	Opciones de segundo nivel	Propiedad <i>Selectable</i>	Propiedad <i>NavigateUrl</i>
MenuUsuario.aspx	Inicio		<i>True</i>	<i>~/MenuUsuario.aspx</i>
	Productos		<i>False</i>	
MenuAdmin.aspx		Ver Productos	<i>True</i>	<i>~/ProductosVer.aspx</i>
	Compras		<i>False</i>	
	Inicio		<i>True</i>	<i>~/MenuAdmin.aspx</i>
	Productos			
		Mantener	<i>True</i>	<i>~/ProductosMantener.aspx</i>
	Ventas		<i>False</i>	
		Pedidos por cliente	<i>True</i>	<i>~/PedidosPorCliente.aspx</i>

Como puede apreciarse en las ilustraciones anteriores, se muestra en el control *Label*, denominado *lblDatosUsuarios* situado en la esquina superior derecha y contenido en el elemento *<div>* denominado *cabecera3*, el valor almacenado en la variable de sesión *Rol*, *A=Administrador* o *U=Usuario*, y el valor de la variable de sesión *Nombre*. Estas variables de sesión almacenarán los valores correspondientes al usuario que haya iniciado la sesión, siendo accesibles por todos los Web Form durante toda la sesión. Ello, se consigue añadiendo el código asociado al evento *Load* de ambos Web Forms y que se muestra más adelante.

Además, para evitar accesos indeseados estos dos Web Forms, introduciendo directamente la *url* de las páginas *.aspx* en la barra de exploración del navegador Web, es necesario añadir el siguiente código asociado al evento *Load* del Web Form *MenuAdmin.aspx*.

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

Y el siguiente código asociado al evento *Load* del Web Form *MenuUsuario.aspx*.

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

Al hacer clic el control Button cuyo texto es *Salir de la Aplicación*, se eliminarán todos los estados de la sesión existente, se cancelará la sesión y se redirigirá hacia la página de acceso a la aplicación Web. Para ello, añadir el siguiente código asociado al evento *Click* del control Button denominado *btnSalir* de ambos Web Forms.

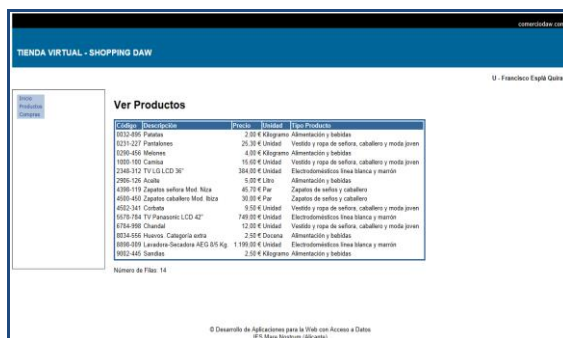
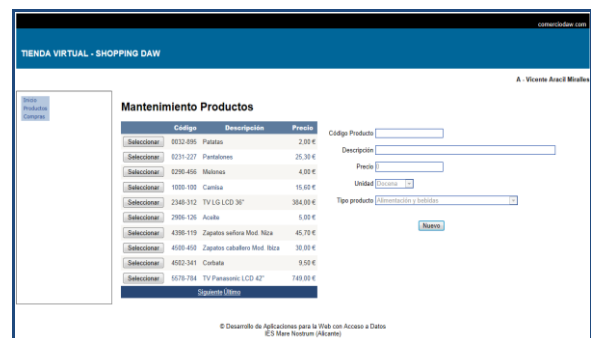
```
protected void btnSalir_Click(object sender, EventArgs e)
{
    Session.Clear();
    Session.Abandon();
    Response.Redirect("~/Default.aspx", false);
}
```

Finalmente, para que las páginas no puedan quedar almacenadas en la memoria cache del servidor, conviene incluir la siguiente línea al principio del código de ambas páginas, 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, las aplicaciones Web, puesto que trabajan con datos representativos de la realidad cambiante a lo largo del tiempo, no deberían utilizar memoria de cache de ningún tipo para garantizar la actualidad y consistencia de la información que presentan. Por lo que, se deberá incluir la línea de código anterior en todos los Web Forms que conformen la aplicación Web.

9. Sustituir el control *Menu* que aparece en el Web Form *ProductosVer.aspx*, por el existente en el Web Form *MenuUsuario.aspx*. esta tarea puede completarse realizando acciones del tipo copiar y pegar desde las vistas **Diseño** de los Web Forms.
10. Igualmente, sustituir el control *Menu* existente en los Web Forms *ProductosMantener.aspx* y *PedidosPorCliente.aspx* por el que se ha modificado en el Web Form *MenuAdmin.aspx*.
11. Iniciar la depuración para comprobar los resultados obtenidos. En este caso se deberá comprobar el funcionamiento de los casos de uso definidos para la aplicación Web: el de perfil de administrador, accediendo mediante un usuario con rol de administrador y, el perfil de usuario, accediendo mediante un usuario con rol de usuario.

12. Para evitar accesos directos indeseados a cualquier Web Form de la aplicación Web, escribiendo directamente las direcciones de la páginas *.aspx* en la barra de exploración del navegador Web y, también, para poder mostrar los datos del usuario que ha iniciado su sesión en el control *Label* correspondiente del Web Form, incluir el siguiente código en el procedimiento de evento *Load* de todas y cada una de los Web Form a proteger, según se trate de un usuario con Rol de Administrador o con Rol de Usuario:

- Páginas de acceso de los usuarios con rol de Administrador, como son las siguientes: *ProductosMantener.aspx* y *PedidosPorCliente.aspx*.

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

- Páginas de acceso de los usuarios con rol de Usuario, como por ejemplo *ProductosVer.aspx*.

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

13. Realizar las pruebas de acceso a la aplicación correspondientes empleando usuarios de pertenecientes a los dos tipos de roles que están definidos en la tabla Usuario de la Base de Datos *Tienda.mdb*. Como se observará no puede accederse a las páginas protegidas si el usuario no está autenticado. Sin embargo, sí es posible acceder a las páginas a través del botón **Back** o **Atrás** del navegador Web. Las páginas Web almacenadas en el objeto *history* del navegador le pertenecen al usuario y por tanto, no se contempla la posibilidad de eliminar la cache desde las aplicaciones Web ni de desactivar el botón **Atrás**. La solución habitual es mostrar un mensaje, informando al usuario de la conveniencia de cerrar la ventana del navegador cuando se accede a la página de acceso, *Default.aspx*, después de abandonar la sesión de la aplicación Web. Otra opción es cerrar la ventana del navegador a través de un script de cliente al abandonar la sesión.

Ejercicio 6

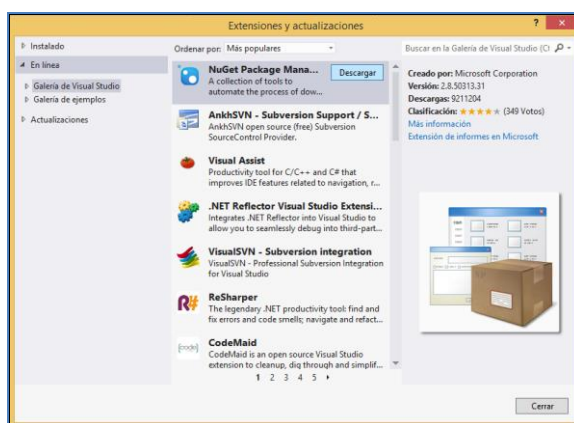
Acceso a datos mediante Entity Framework

En este ejercicio se aborda el acceso a datos con ASP.NET empleando Entity Framework. Para ello, se generará el modelo de entidad o Entity Data Model (EDM) desde la base de datos *Tienda.mdf*. Y a continuación, se abordará cómo realizar el acceso a los datos utilizando para ello el contexto de acceso a entidades del EDM generado.

Acciones previas a la realización del ejercicio

En la mayoría de los casos, el desarrollo de un sitio Web de ASP.NET precisa de la instalación y configuración de paquetes software o bibliotecas que proveen al sitio Web de funcionalidades específicas diversas. Cuando se utiliza Entity Framework, concretamente para crear el Entity Data Model (EDM) desde una base de datos existente, Visual Studio necesita de la instalación previa de determinados paquetes software. Afortunadamente, Visual Studio dispone de la herramienta **Administrador de paquetes de NuGet** que se encarga de la administración y configuración, de forma desatendida, de los paquetes software necesarios para cada sitio Web. Por lo que, antes de realizar este ejercicio, es necesario instalar el Administrador de paquetes de NuGet. Para ello:

1. Seleccionar la opción Extensiones y actualizaciones del menú Herramientas. En la opción Instalado | Todo se muestran las extensiones de Visual Studio instaladas, con lo que puede comprobarse si está instalado el Administrador de paquetes de NuGet.
2. En caso que no esté instalado el Administrador de Paquetes de NuGet, seleccionar la opción En línea en la ventana Extensiones y actualizaciones. Y a continuación, hacer clic sobre el botón Descargar que aparece junto a la extensión denominada NuGet Package Manager.



3. Finalizado el proceso de instalación puede comprobarse que se ha instalado el Administrador de Paquetes de NuGet, nuevamente mediante la opción Instalado | Todo.
4. Para que los cambios realizados surtan efecto se debe reiniciar Visual Studio.

Se puede acceder a la configuración del Administrador de paquetes de NuGet mediante la opción Opciones del menú Herramientas. En la ventana Opciones, hacer clic en la casilla de verificación Mostrar todas las configuraciones. Y a continuación, seleccionar la opción Administrador de paquetes

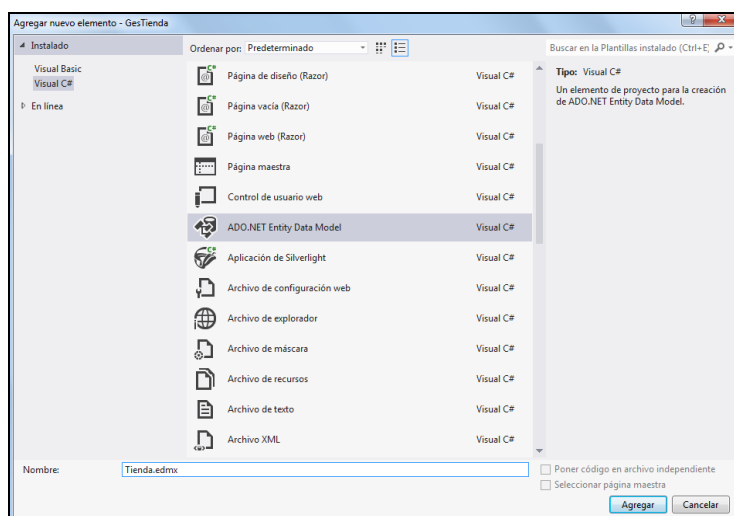
de NuGet. Las siguientes opciones, que están activadas de forma predeterminada, deben estar activadas para que la herramienta Administración de paquetes de NuGet administre y configure en modo desatendido o automático los paquetes necesarios para cada sitio Web:

- Comprobar automáticamente los paquetes que falten durante la compilación.
- Permitir a NuGet descargar los paquetes que falten.

Generación del Entity Data Model (EDM)

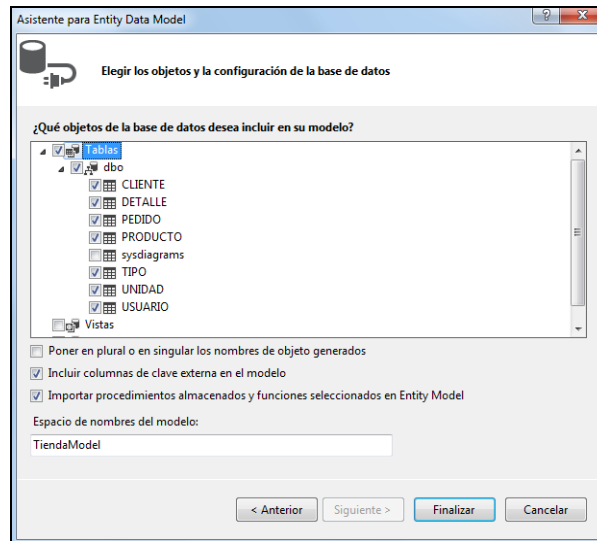
Realizar las siguientes acciones para generar el EDM enlazado a la base de datos de SQL Server *Tienda.mdf* en el sitio Web *GesTienda* de la solución *Tienda*:

1. En el Explorador de soluciones hacer clic en el botón derecho sobre el nombre del sitio Web *GesTienda* y seleccionar la opción **Agregar nuevo elemento...**
2. En el cuadro de diálogo Agregar nuevo elemento:
 - a. Seleccionar Visual C# como lenguaje de programación.
 - b. Seleccionar la plantilla **ADO.NET Entity Data Model**.
 - c. Finalmente, en el cuadro de texto situado en la parte inferior introducir como nombre del modelo *Tienda.edmx* y hacer clic en **Agregar**.

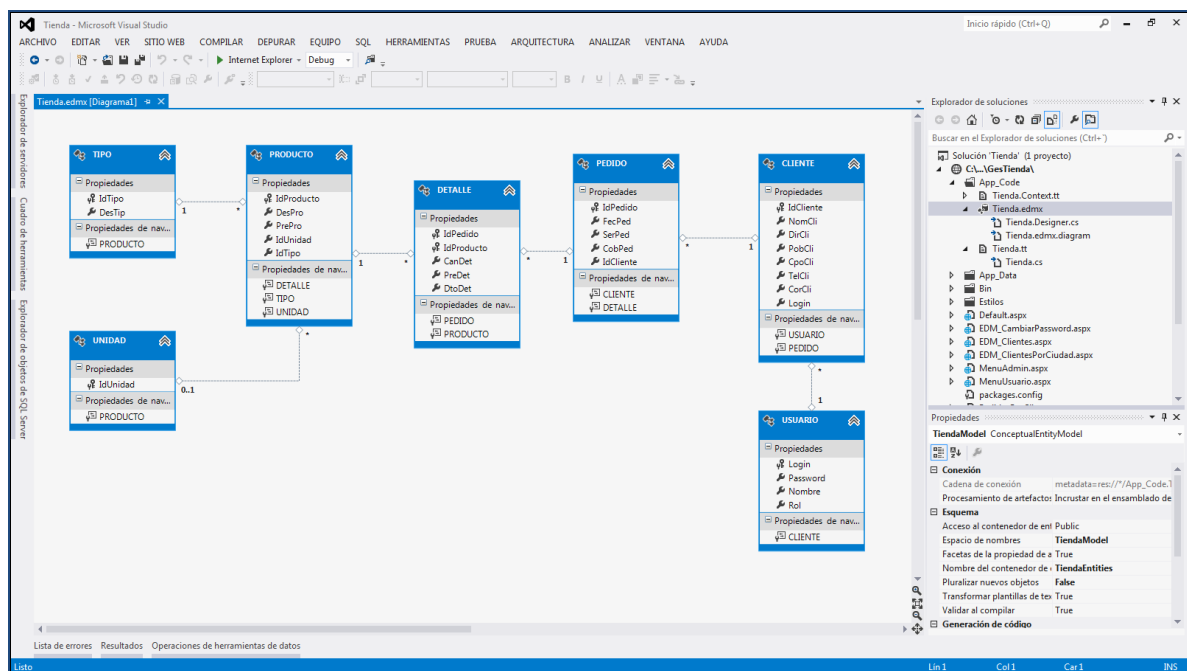


3. A continuación, contestar afirmativamente a la confirmación de poner el EDM en la carpeta de ASP.NET denominada App_Code.
4. Se inicia el asistente para generar el EDM desde una base de datos existente o crear un modelo vacío. Realizar las siguientes opciones en el asistente:
 - a. Seleccionar la opción **Generar desde la base de datos** y hacer clic en **Siguiente**.
 - b. A continuación, seleccionar la conexión a la que hace referencia al archivo de base de datos de SQL Server *Tienda.mdf*. Comprobar que queda marcada la casilla de verificación **Guardar configuración de conexión de entidad en el archivo Web.Config como**. En el cuadro de texto inferior se propone *TiendaEntities* como nombre del contexto de acceso, ya que el nombre que se ha dado al EDM ha sido *Tienda.edmx*. Aceptar el nombre propuesto haciendo clic en **Siguiente**.

- c. Elegir los objetos de la base de datos que formarán parte del modelo mapeado desde la base de datos en la aplicación Web. En este caso, elegir todas las tablas de datos, o sea: CLIENTE, DETALLE, PEDIDO, PRODUCTO, TIPO, UNIDAD y USUARIO. En el cuadro de texto inferior se propone *TiendaModel* como nombre del espacio de nombres del modelo, ya que el nombre que se ha dado al EDM ha sido *Tienda.edmx*. Finalmente, aceptar el nombre propuesto haciendo clic en **Finalizar**.



5. Finalizado el trabajo del asistente se habrá generado el EDM a partir del mapeado de la base de datos *Tienda.mdf* y se mostrará el diagrama de entidad del modelo que se almacena en el archivo denominado *Tienda.edmx*.



En el Explorador de soluciones, dentro de la carpeta de ASP.NET *App_Code*, pueden apreciarse los archivos generados por el asistente de forma automática, incluido el archivo *Tienda.edmx*. El archivo *Tienda.cs*, que se ha generado dentro del elemento *Tienda.tt*, contiene la definición de una propiedad para cada una de las tablas de la base de datos, que en realidad es un objeto sobre el cual pueden realizarse operaciones de manipulación de datos. Por otra parte, en el contenido del archivo *Web.Config* se habrá añadido en la sección *ConnectionStrings* una nueva línea que referencia al contexto de acceso a entidades, denominado *TiendaEntities*.

Acceso a los datos empleando el modelo declarativo

Una vez generado el EDM y el contexto de acceso a entidades, que ha sido mapeado desde la base de datos *Tienda.mdf*, es posible acceder a los datos del modelo a través del modelo declarativo. Para ello, realizar las siguientes acciones:

1. Crear un nuevo Web Form denominado *EDM_Clientes.aspx* a partir de una copia de *WebFormPlantilla.aspx*.
2. En la Vista Diseño del Web Form creado, agregar un control de origen de datos **EntityDataSource**, denominado *EntityDataSource1*, desde el Cuadro de Herramientas.
3. En la Tareas del control **EntityDataSource** agregado, seleccionar la opción **Configurar origen de datos...** En el asistente realizar las siguientes acciones:
 - a. Elegir la conexión con nombre denominada *TiendaEntities* y hacer clic en **Siguiente**.
 - b. Seleccionar la entidad *CLIENTE* en el apartado **Entity Set Name**, elegir todas las propiedades en el resultado de la consulta y, hacer clic en **Finalizar**.
4. En la Vista Diseño del Web Form, agregar desde el Cuadro de Herramientas un control de enlace a datos de tipo **GridView**, denominado *GridView1*.
5. En las Tareas del control **GridView** agregado, elegir *EntityDataSource1* como **Origen de datos**.
6. Iniciar la depuración para comprobar los resultados obtenidos. Ajustar las propiedades y opciones de presentación para obtener un resultado similar a de la siguiente ilustración.

TIENDA VIRTUAL - SHOPPING DAW

A - Vicente Aracil Miralles

Cientes

IdCliente	NomCli	DirCli	PobCli	CpoCli	TelCli	CorCli	Login
00123659Z	Beatriz Iraola López	Zuriaga, 25	Bilbao	09120	686343211	beatriz@gmail.es	beatriz
07899905V	Jesús García Soria	Gaztambique, 32	Zaragoza	52390	660542392	jesus@hotmail.es	suso
08659442S	Miguel Sogorb Fenoll	Armadores, 1	Alicante	03001	654789995	miguel@alc.es	msogorb
09897907K	Alvaro Fernández Moreno	Fuencarral, 33	Madrid	28081	609667878	alvaro@yahoo.es	afm
10900801K	Alejandro Castro Hernández	Leonor de Cortinas, 7	Zamora	51003	954678966	alex@gmail.es	acastro
11265888J	Sandra Rostoll Trias	Pablo Neruda, 15	Barcelona	08102	931123451	sandra@lycos.es	rostoll
11908762X	Rocio Espá Sirvent	Cervantes, 22	Alicante	03008	965087790	roci@srva.es	morena
12667489B	Joaquín Porta Sabater	Vinateros, 121	Barcelona	08162	609765890	ximo@fma.com	jporta
15437882H	Sergio Fernández Pérez	Avenida del Ejercito, 76	Madrid	28030	912345673	sergio@ctv.es	ferper
15673999A	Eneko Ugalde Garmendia	Federico Puentes, 3	Bilbao	09002	606897654	jonas@izk.com	eneko

Siguiente Último

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

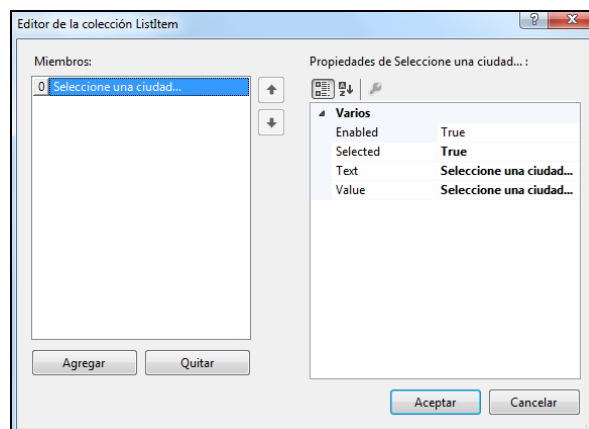
Seleccionar datos empleando código lógico

Una vez generado el EDM y el contexto de acceso a entidades también es posible acceder a los datos del modelo a través del código lógico, de forma independiente del medio en qué se almacenan los datos. En este apartado del ejercicio, se aborda la creación de un Web Form que muestre los clientes existentes en una determinada ciudad seleccionada. Para ello, realizar las siguientes acciones:

1. Crear un nuevo Web Form denominado *EDM_ClientesPorCiudad.aspx* a partir de una copia de *WebFormPlantilla.aspx*.
2. En la Vista Diseño del Web Form creado, agregar un control **DropDownList**, cuyo ID sea *ddlCiudad*, desde el Cuadro de Herramientas que se utilizará para seleccionar la ciudad.
3. Modificar las propiedades del control *ddlCiudad* según la siguiente tabla:

Propiedad	Valor	Descripción
AppendDataBoundsItems	True	Permite agregar elementos a control antes de que se produzca el enlace de datos
AutoPostBack	True	Indica si se producirá un Post-Back automático al servidor cuando el usuario cambie la selección de la lista
DataTextField	Key	Campo del origen de datos que proporciona el contenido del texto de los elementos de lista
DataValueField	Key	Campo del origen de datos que proporciona el valor de cada elemento de lista
SelectMethod	GetCiudades	Nombre del método al que se va a llamar para leer datos

Además, acceder a la colección de la Propiedad **Items** del control denominado *ddlCiudad* para añadir un nuevo elemento a la lista que sea una descripción “Seleccione una ciudad...”, tal como se muestra en la siguiente ilustración. La propiedad **AppendDataBoundItems**



4. Agregar la siguiente definición del método *GetCiudades()* en el archivo de código subyacente:

```
public IQueryable<IGrouping<string, CLIENTE>> GetCiudades()
{
    TiendaEntities contexto = new TiendaEntities();
    IQueryable<IGrouping<string, CLIENTE>> consulta =
        contexto.CLIENTE.GroupBy(CLIENTE => CLIENTE.PobCli);
    return consulta;
}
```

El resultado del método *GetCiudades()* es una lista agrupada de las ciudades de los clientes. De forma predeterminada, el nombre del campo de la lista resultante es *Key*, que es el valor asignado a las propiedades *DataTextField* y *DataValueField* del control *ddlCiudad*, tal como puede comprobarse en la tabla anterior.

- En la Vista Diseño del Web Form creado, agregar un control de enlace a datos de tipo **GridView**, cuya propiedad ID sea *grdClientes*, desde el Cuadro de Herramientas.
- Modificar las propiedades del control *grdClientes* según la siguiente tabla:

Propiedad	Valor	Descripción
ItemType	CLIENTE	Tipo del elemento de enlace de datos fuertemente tipado
SelectMethod	GetClientesPorCiudad	Nombre del método al que se va a llamar para leer datos

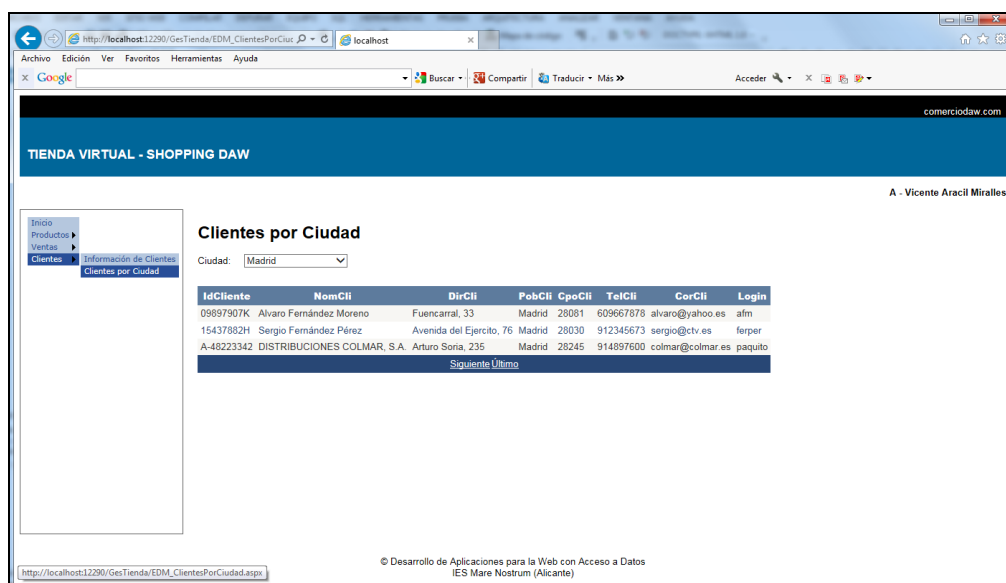
- Agregar la definición del método *GetClientesPorCiudad()* en el archivo de código subyacente:

```
public IQueryable<CLIENTE> GetClientesPorCiudad()
{
    String w_ciudad = ddlCiudad.SelectedValue;
    TiendaEntities contextoTienda = new TiendaEntities();
    IQueryable<CLIENTE> consulta =
        contextoTienda.CLIENTE.OrderBy(CLIENTE => CLIENTE.PobCli).Where(CLIENTE => CLIENTE.PobCli ==
            w_ciudad);
    return consulta;
}
```

- Agregar el siguiente código asociado al evento **SelectedIndexChanged** del control *ddlCiudad*.

```
protected void ddlCiudad_SelectedIndexChanged(object sender, EventArgs e)
{
    grdClientes.DataBind();
}
```

- Iniciar la depuración para comprobar los resultados obtenidos. Ajustar las propiedades y opciones de presentación para obtener un resultado similar al de la siguiente ilustración.

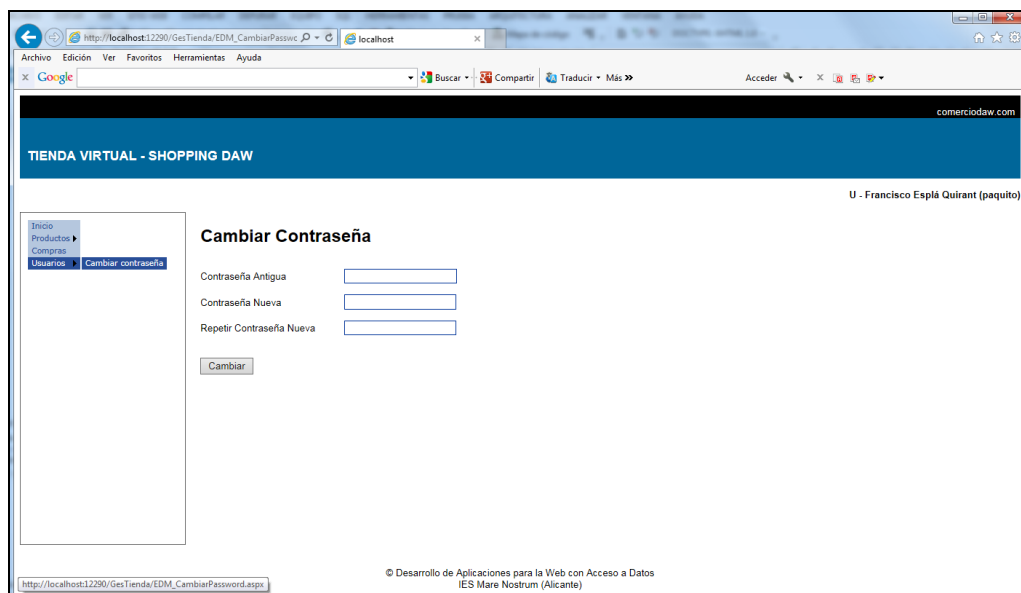


10. Tal como se ha realizado en ejercicios anteriores, introducir código en el evento *Load* del objeto *Page* para proteger los Web Forms *EDM_Clientes.aspx* y *EDM_ClientesPorCiudad.aspx* creados en este ejercicio de accesos indeseados. De manera que solo puedan tener acceso a éstos los usuarios de la aplicación Web con Rol de *Administrador*.
11. Añadir un elemento del menú de administrador no seleccionable denominado *Clientes* y que incluya dos elementos secundarios que permitan acceder a los dos Web Foms creados.

Modificar datos empleando código lógico

En esta última parte del ejercicio, se aborda la creación de un Web Form que permita modificar la contraseña del usuario que ha iniciado la sesión. Para ello, realizar las siguientes acciones:

1. Crear un nuevo Web Form denominado *EDM_CambiarPassword.aspx* a partir de una copia de *WebFormPlantilla.aspx*.
2. Tal como se ha realizado en ejercicios anteriores, introducir el código correspondiente en el evento *Load* del objeto *Page*, proteger el Web Form *EDM_CambiarPassword.aspx* de accesos indeseados, de modo que solo puedan tener acceso al mismo los usuarios de la aplicación Web con Rol de *Usuario*. Además, añadir la opción correspondiente en el menú de usuario para que los usuarios comunes de la aplicación Web puedan acceder a este Web Form.
3. En la Vista Diseño del Web Form, agregar los controles necesarios y establecer las propiedades y opciones de visualización necesarias para que la presentación del Web Form *EDM_CambiarPassword.aspx* sea similar a la que se muestra a continuación.



Las propiedades más significativas de los controles agregados a Web Form pueden ser:

Control	ID	TextMode	Text
TextBox	txtAntigua	Password	
TextBox	txtNueva1	Password	
TextBox	txtNueva2	Password	
Button	btnCambiar		Cambiar

4. Para acceder a los datos del usuario que ha iniciado la sesión activa se utilizará una variable de sesión que almacene el valor del campo *Login* de la tabla *USUARIO*. Para disponer de esta variable de sesión es necesario modificar el código lógico del archivo de código subyacente *Default.aspx.cs* de la siguiente forma:
 - a. Modificar la sentencia SQL para seleccionar el campo *Login* de la tabla *USUARIO*.

```
string StrComandoSql = "SELECT Login, nombre, rol FROM USUARIO ";
```

- b. Agregar el código necesario para crear la variable de sesión *Login*, justo antes de crear las variables *Nombre* y *Rol*, y reordenar el acceso a los campos de la consulta.

```
Session.Add("Login", reader.GetString(0));
Session.Add("Nombre", reader.GetString(1));
Session.Add("Rol", reader.GetString(2));
```

5. Nuevamente en el Web Form denominado *EDM_CambiarPassword.aspx*, agregar el siguiente código lógico asociado al evento **Click** del control *btnCambiar*.

```
protected void btnCambiar_Click(object sender, EventArgs e)
{
    using (TiendaEntities contextoTienda = new TiendaEntities())
    {
        String w_Login = Convert.ToString(Session["Login"]);

        try
        {
            IQueryable<USUARIO> consulta = contextoTienda.USUARIO.OrderBy(USUARIO =>
                USUARIO.Login).Where(USUARIO => USUARIO.Login == w_Login);
            if (consulta.Count() > 0)
            {
                USUARIO fila = consulta.First();
                if (fila.Password.Trim() == txtAntigua.Text)
                {
                    if (txtNueva1.Text == txtNueva2.Text)
                    {
                        fila.Password = txtNueva1.Text;
                        // La línea siguiente permite comprobar el control de errores en la transacción
                        // fila.Login = "111";
                        int num = contextoTienda.SaveChanges();
                        if (num == 0)
                        {
                            lblMensajes.Text = "Cambios no realizados";
                        }
                        else
                        {
                            lblMensajes.Text = "Contraseña Actualizada";
                        }
                    }
                    else
                    {
                        lblMensajes.Text = "Cambio no realizado. Contraseñas nuevas no coincidentes";
                    }
                }
                else
                {
                    lblMensajes.Text = "Cambio no realizado. Contraseña antigua incorrecta";
                }
            }
        }
    }
}
```

```

        else
        {
            lblMensajes.Text = "Usuario inexistente";
        }
    }
    catch (Exception ex)
    {
        lblMensajes.Text = "Error de acceso a datos: " + ex.Message;
    }
}
}

```

En el código anterior puede apreciarse cómo se asigna el valor de la variable de sesión *Login* a la variable local *w_Login*. A continuación, se asigna a la variable *consulta* el resultado de la invocación a un método *Where* sobre la entidad *USUARIO* para recuperar los datos del usuario activo de la sesión. Tras comprobar que se ha recuperado información desde la entidad *USUARIO* y validar los valores de las contraseñas antigua y nueva, se ejecuta el método *SaveChanges()* para guardar todos los cambios realizados en el contexto en la base de datos subyacente. El valor devuelto por el método *SaveChanges()* es el número de objetos escritos en la base de datos. También puede apreciarse en el código anterior, el uso de la estructura try-catch para establecer el control de errores sobre la operación de acceso a datos. Para comprobar el funcionamiento del control de errores puede quitarse el comentario sobre la línea de código que modifica el valor de la propiedad *Login* de la entidad *USUARIO*, lo cual produce una excepción al intentar modificar el valor del campo clave de la tabla. Una vez comprobado el funcionamiento del control de errores de actualización introducido en la transacción, se recomienda volver a comentar esa línea de código para obtener un funcionamiento correcto del Web Form.

6. Finalmente, iniciar la depuración para comprobar los resultados obtenidos. Para ello, modificar la contraseña de un usuario y, a continuación, iniciar la sesión con la nueva contraseña del usuario. A través del **Explorador de Servidores** puede accederse a los datos almacenados en la tabla *USUARIO* para comprobar que el valor de la contraseña de ese usuario se ha modificado en el registro correspondiente a ese usuario la base de datos.