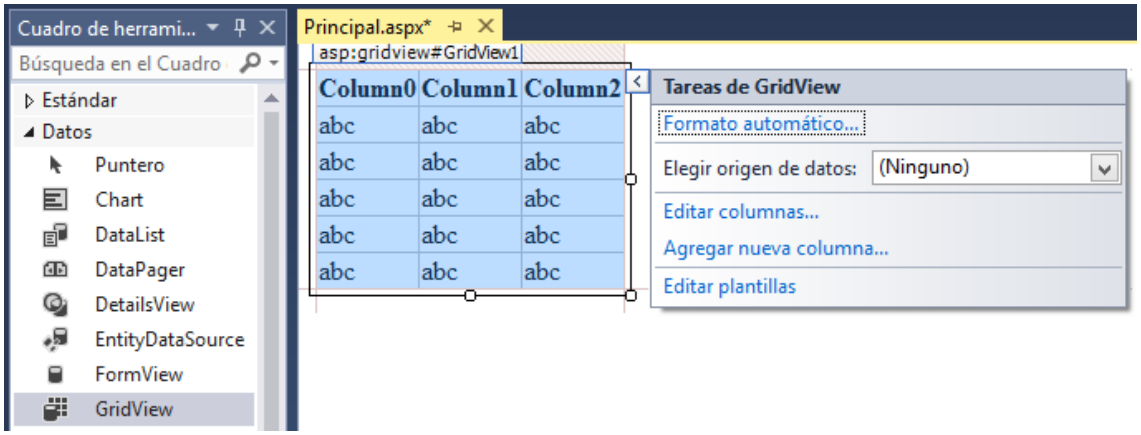
USO DE GRIDVIEW

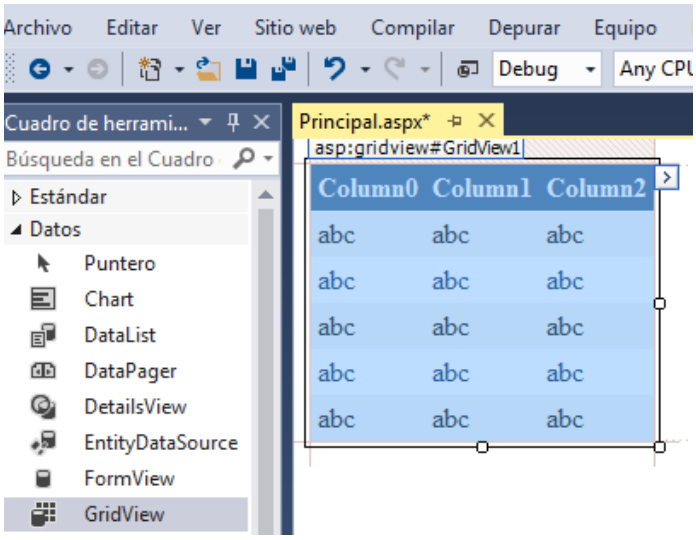
En este ejercicio vamos a trabajar con un control que nos muestre todos los registros de una tabla o de una consulta: control **GridView**. Este es un control de los denominados ENLAZADOS, es decir, que podremos enlazarlo a través de una propiedad denominada **DataSource** con un objeto que le pase los datos. Este objeto puede ser una Lista tipada, un DataReader, un DataTable,....

En nuestro ejercicio, como estamos trabajando con capas y solemos trabajar con Listas, vamos a utilizar como origen de datos las Listas de objetos.

1.- Introduciremos en el formulario el control **GRIDVIEW**, que se encuentra en la barra de herramientas dentro del grupo de elemento **Datos**:



2.- Seleccionaremos el formato automático para darle el aspecto que más nos guste. Pulsaremos la **flecha** que aparece a continuación del control y elegiremos la opción de **Formato Automático**. En el ejemplo he escogido el estilo profesional:



Veremos como en la página de código nos ha introducido la etiqueta correspondiente al **GridView** con una gran cantidad de propiedades de estilo que nos ha generado el formato automático. *Podremos cambiarlas si nos interesa.*

```
<form id="form1" runat="server">
<div>

    <asp:GridView ID="GridView1" runat="server" CellPadding="4" ForeColor="#333333"
        <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
        <EditRowStyle BackColor="#999999" />
        <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
        <HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
        <PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" /
        <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
        <SelectedRowStyle BackColor="#E2DED6" Font-Bold="True" ForeColor="#333333" /
        <SortedAscendingCellStyle BackColor="#E9E7E2" />
        <SortedAscendingHeaderStyle BackColor="#506C8C" />
        <SortedDescendingCellStyle BackColor="#FFFDF8" />
        <SortedDescendingHeaderStyle BackColor="#6F8DAE" />
    </asp:GridView>
```

3.- Ahora nos toca introducir las columnas que queremos mostrar en el control **GridView**. Esto lo haremos con la etiqueta **<Columns></Columns>** dentro del **GridView**.

Dentro de esta etiqueta vamos a colocar una serie de elementos que serán plantillas para poder mostrar, editar y añadir registros a mi origen de datos.

La etiqueta **<asp:TemplateField> </asp:TemplateField>** será la plantilla general para cada campo que queramos mostrar en el **GridView**. Por tanto debemos introducir tantos **<asp:TemplateField> </asp:TemplateField>** como campos queramos representar.

Pero antes vamos a ver que necesitamos dentro de este elemento.

Dentro de este elemento podremos introducir determinadas plantillas y estilos para ellas. En este ejercicio nos interesan tres plantillas (la de **lectura**, la de **edición** y **la de pie del GridView**). Por tanto introduciremos 3 elementos:

- **<Item Template> </Item Template>** correspondiente al modo lectura.
- **<EditItemTemplate></EditItemTemplate>** correspondiente al modo edición.
- **<FooterTemplate></FooterTemplate>** correspondiente al pie del control GridView.

```
<Columns>
    <asp:TemplateField>
        <ItemTemplate>

        </ItemTemplate>
        <EditItemTemplate>

        </EditItemTemplate>
        <FooterTemplate>

        </FooterTemplate>
    </asp:TemplateField>
</Columns>
```

- Dentro del elemento **<Item Template> </Item Template>** introduciremos el control que nos servirá para visualizar el campo en modo lectura dentro del **GridView**. En nuestro caso utilizaremos una etiqueta:

```
<asp:Label  runat="server" Text=' <%# Eval("Title_ID") %>' ></asp:Label>
```

La etiqueta debe llevar en la propiedad Text una expresión: ' <%# Eval("Title_ID") %>'. Esta debe ir entre comillas simples y la etiqueta debe venir entre símbolos <%# y %> la expresión es **Eval("Title_Id").** La expresión **Eval("Title_Id")** toma del origen de datos establecido en la propiedad **DataSource** el campo o propiedad que se denomine **Title_Id** (campo que se representa dentro de los paréntesis de la **función Eval** y entre comillas dobles).

- **<EditItemTemplate></EditItemTemplate>** dentro de este elemento introduciremos el control que nos servirá para editar y actualizar el campo dentro del **GridView**. En nuestro caso utilizaremos un cuadro de texto:

```
<asp:TextBox ID="TxtTitle_ID" Text=' <%# Eval("Title_ID") %>'
runat="server"></asp:TextBox>
```

En este cuadro de texto necesitaremos el **ID del control**, para poder acceder luego a este control cuando necesitemos actualizar o modificar alguno de los campos. En la propiedad Text vendrá el mismo valor que ya hemos explicado anteriormente: ' <%# Eval("Title_ID") %>'.

- **<FooterTemplate></FooterTemplate>** dentro de este elemento introduciremos un control que nos servirá para introducir nuevos valores en el campo a la hora de introducir un registro nuevo. Para nuestro ejercicio utilizaremos un cuadro de texto.

```
<asp:TextBox ID="TxtTitle_IDFooter"  runat="server"></asp:TextBox>
```

Llevará un **ID** para poder acceder a él a la hora de insertar los datos en el origen de datos, pero ya no se carga la **propiedad Text** dado que estos cuadros servirán para introducir registros nuevos, así que inicialmente deben ir vacíos.

Quedará por tanto de la siguiente forma:

```
<Columns>
    <asp:TemplateField HeaderText="Title_Id">
        <ItemTemplate>
            <asp:Label  runat="server" Text=' <%# Eval("Title_ID") %>' ></asp:Label>
        </ItemTemplate>
        <EditItemTemplate>
            <asp:TextBox ID="TxtTitle_ID" Text=' <%# Eval("Title_ID") %>' runat="server"></asp:TextBox>
        </EditItemTemplate>
        <FooterTemplate>
            <asp:TextBox ID="TxtTitle_IDFooter"  runat="server"></asp:TextBox>
        </FooterTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Title">
        <ItemTemplate>
            <asp:Label  runat="server" Text=' <%# Eval("Title") %>' ></asp:Label>
        </ItemTemplate>
        <EditItemTemplate>
            <asp:TextBox ID="TxtTitle" Text=' <%# Eval("Title") %>' runat="server"></asp:TextBox>
        </EditItemTemplate>
        <FooterTemplate>
            <asp:TextBox ID="TxtTitleFooter"  runat="server"></asp:TextBox>
        </FooterTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Type">
        <ItemTemplate>
            <asp:Label  runat="server" Text=' <%# Eval("Type") %>' ></asp:Label>
        </ItemTemplate>
        <EditItemTemplate>
            <asp:TextBox ID="TxtType" Text=' <%# Eval("Type") %>' runat="server"></asp:TextBox>
        </EditItemTemplate>
        <FooterTemplate>
            <asp:TextBox ID="TxtTypeFooter"  runat="server"></asp:TextBox>
        </FooterTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Price">
        <ItemTemplate>
            <asp:Label  runat="server" Text=' <%# Eval("Price") %>' ></asp:Label>
        </ItemTemplate>
        <EditItemTemplate>
            <asp:TextBox ID="TxtPrice" Text=' <%# Eval("Price") %>' runat="server"></asp:TextBox>
        </EditItemTemplate>
        <FooterTemplate>
            <asp:TextBox ID="TxtPriceFooter"  runat="server"></asp:TextBox>
        </FooterTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Pub_ID">
    <ItemTemplate>
        <asp:Label  runat="server" Text=' <%# Eval("Pub_Id") %>' ></asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
        <asp:TextBox ID="TxtPub_Id" Text=' <%# Eval("Pub_Id") %>' runat="server"></asp:TextBox>
    </EditItemTemplate>
    <FooterTemplate>
        <asp:TextBox ID="TxtPub_IdFooter"  runat="server"></asp:TextBox>
    </FooterTemplate>
</asp:TemplateField>
  <asp:TemplateField HeaderText="Pubdate">
    <ItemTemplate >
        <asp:Label  runat="server" Text=' <%# Eval("Pubdate") %>' ></asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
        <asp:TextBox ID="TxtPubdate" Text=' <%# Eval("Pubdate") %>' runat="server"></asp:TextBox>
    </EditItemTemplate>
    <FooterTemplate>
        <asp:TextBox ID="TxtPubdateFooter"  runat="server"></asp:TextBox>
    </FooterTemplate>
</asp:TemplateField>
</Columns>
```

Como se ve en la imagen anterior dentro del elemento **<asp:TemplateField>** añadiremos la propiedad **HeaderText="NombreCampo"** para que aparezca el nombre del campo en la cabecera. De todas formas no aparecerá la cabecera hasta que pongamos otro atributo más: **ShowHeader="true"** en la etiqueta del **GridView**.

Igualmente para que se vean la fila de pie del **GridView**, es decir, el elemento **<FooterTemplate>** también habrá que añadir el atributo **ShowFooter="true"** en la etiqueta **GridView**.

Para que el **GridView** no genere columnas automáticamente y solo presente las que nosotros añadamos pondremos otra propiedad en la etiqueta del **GridView AutoGenerateColumns="false".**

4.- El siguiente paso es probar que todo va correctamente, así que vamos a pasar a implementar el código para cargar el control.

Como estamos trabajando con un modelo a capas vamos a realizar en la clase del proyecto de datos D_LIBROS un método para cargar los libros de la **Tabla Titles** en una Lista de Libros.

```
public class D_LIBROS : Conexion
{
    public List<Libro> getLibros()
    {
        decimal precio=0;
        List<Libro> listadoLibros = new List<Libro>();
        SqlCommand cmdLibros = new SqlCommand("Select * from titles", cnn);
        abrir();
        SqlDataReader drLibros = cmdLibros.ExecuteReader();
        while (drLibros.Read())
        {
            if (drLibros["price"] != DBNull.Value)
            {
                precio=Convert.ToDecimal(drLibros["price"]);
            }
            Libro l = new Libro(drLibros[0].ToString(), drLibros[1].ToString(), drLibros["type"].ToString(), precio, drLibros["pub_id"].ToString(),
                    Convert.ToDateTime(drLibros["pubdate"]));
            listadoLibros.Add(l);
        }

        drLibros.Close();
        cmdLibros.Dispose();
        cerrar();
        return listadoLibros;
    }
```

Crearemos el método correspondiente en la clase de NEGOCIO N_LIBROS:

```
public class N_LIBROS
{
    D_LIBROS nLibros = new D_LIBROS();

    public List<Libro> getLibros()
    {
        return nLibros.getLibros();
    }
```

Y por último en la capa de presentación:

```
public partial class Principal : System.Web.UI.Page
{
    N_LIBROS pLibros = new N_LIBROS();

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            List<Libro> listaLibros = pLibros.getLibros();
            foreach (Libro item in listaLibros)
            {
                DdlLibros.Items.Add(new ListItem(item.Title, item.Title_Id));
            }

            GridView1.DataSource = listaLibros;
            GridView1.DataBind();
        }
    }
```

Si probamos y ejecutamos ahora deben aparecer todos los registros de la **tabla Titles**. Solamente aparecerán las columnas deseadas.

Title_Id	Title	Type	Price	Pub_ID	Pubdate
as1111	1111	1111	11,2500	0736	27/01/2019 0:00:00
BU1032	The Busy Executive's Database Guide	business	19,0000	1389	12/06/1991 0:00:00
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	11,9500	1389	09/06/1991 0:00:00
BU2075	You Can Combat Computer Stress!	business	2,9900	0736	30/06/1991 0:00:00
BU7832	Straight Talk About Computers	business	19,9900	1389	22/06/1991 0:00:00
BU9900	You Can Combat Computer Stress!	business	2,9900	0736	30/06/1991 0:00:00
MC2222	Silicon Valley Gastronomic Treats	mod_cook	19,9900	0877	09/06/1991 0:00:00
MC3021	The Gourmet Microwave	mod_cook	2,9900	0877	18/06/1991 0:00:00
MC3026	The Psychology of Computer Cooking	UNDECIDED	2,9900	0877	10/09/2018 16:46:52
PC1035	But Is It User Friendly?	popular_comp	22,9500	1389	30/06/1991 0:00:00
PC8888	Secrets of Silicon Valley	popular_comp	20,0000	1389	12/06/1994 0:00:00
PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	21,5900	0877	21/10/1991 0:00:00
PS2091	Is Anger the Enemy?	psychology	10,9500	0736	15/06/1991 0:00:00
PS2106	Life Without Fear	psychology	7,0000	0736	05/10/1991 0:00:00
PS3333	Prolonged Data Deprivation: Four Case Studies	psychology	19,9900	0736	12/06/1991 0:00:00
PS7777	Emotional Security: A New Algorithm	psychology	7,9900	0736	12/06/1991 0:00:00
Rw1212	Mar	Mar	12,0000	0877	27/01/2019 0:00:00
tc1212	12	12	12,0000	1389	27/01/2019 0:00:00
TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	20,9500	0877	21/10/1991 0:00:00

5.- El siguiente paso va a ser intentar modificar esos datos y para ello vamos a meter unos iconos en una nueva columna. Por tanto crearemos un nuevo **<asp:TemplateField> </asp:TemplateField> e introduciremos las mismas plantillas que en los casos anteriores <ItemTemplate><EditItemTemplate> y <FooterTemplate>.**

Dentro de estos elementos introduciremos:

```
<asp:TemplateField>
        <ItemTemplate>
            <asp:ImageButton ImageUrl="~/IMAGES/edit.png" CommandName="Edit" Width="20" Height="20" runat="server" />
            <asp:ImageButton ImageUrl="~/IMAGES/delete.png" CommandName="delete" Width="20" Height="20" runat="server" />
        </ItemTemplate>
        <EditItemTemplate>
            <asp:ImageButton  ImageUrl="~/IMAGES/save.png" CommandName="update" Width="20" Height="20" runat="server" />
            <asp:ImageButton  ImageUrl="~/IMAGES/cancel.png" CommandName="cancel" Width="20" Height="20" runat="server" />
        </EditItemTemplate>
        <FooterTemplate>
              <asp:ImageButton  ImageUrl="~/IMAGES/addnew.png" CommandName="addnew" Width="20" Height="20" runat="server" />
        </FooterTemplate>
```

El icono de **Editar** y el icono de **Eliminar** en el **ItemTemplate**. Estos serán los iconos que se muestren por defecto al leer los datos del **GridView**.

En el elemento **EditItemTemplate**, es decir, cuando pulsemos el icono de Editar, aparecerán los iconos de **Salvar** y **Cancelar**.

Y en el **FooterTemplate** aparecerá el icono de **Insertar Nuevo Registro**.

Title_Id	Title	Type	Price	Pub_ID	Pubdate
as1111	1111	1111	11,2500	0736	27/01/2019 0:00:00
BU1032	The Busy Executive's Database Guide	business	19,0000	1389	12/06/1991 0:00:00
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	11,9500	1389	09/06/1991 0:00:00
BU2075	You Can Combat Computer Stress!	business	2,9900	0736	30/06/1991 0:00:00
BU7832	Straight Talk About Computers	business	19,9900	1389	22/06/1991 0:00:00
BU9900	You Can Combat Computer Stress!	business	2,9900	0736	30/06/1991 0:00:00
MC2222	Silicon Valley Gastronomic Treats	mod_cook	19,9900	0877	09/06/1991 0:00:00
MC3021	The Gourmet Microwave	mod_cook	2,9900	0877	18/06/1991 0:00:00
MC3026	The Psychology of Computer Cooking	UNDECIDED	2,9900	0877	10/09/2018 16:46:52
PC1035	But Is It User Friendly?	popular_comp	22,9500	1389	30/06/1991 0:00:00
PC8888	Secrets of Silicon Valley	popular_comp	20,0000	1389	12/06/1994 0:00:00
PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	21,5900	0877	21/10/1991 0:00:00
PS2091	Is Anger the Enemy?	psychology	10,9500	0736	15/06/1991 0:00:00
PS2106	Life Without Fear	psychology	7,0000	0736	05/10/1991 0:00:00
PS3333	Prolonged Data Deprivation: Four Case Studies	psychology	19,9900	0736	12/06/1991 0:00:00
PS7777	Emotional Security: A New Algorithm	psychology	7,9900	0736	12/06/1991 0:00:00
Rw1212	Mar	Mar	12,0000	0877	27/01/2019 0:00:00
tc1212	12	12	12,0000	1389	27/01/2019 0:00:00
TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	20,9500	0877	21/10/1991 0:00:00
TC4203	Fifty Years in Buckingham Palace Kitchens	trad_cook	11,9500	0877	12/06/1991 0:00:00

Ahora vamos a probar que, al pulsar el icono de Editar, las etiquetas se convierten en cuadros de texto. Pero los demás eventos no funcionan de momento.

Para que esto funcione y los eventos que generemos hagan lo que tienen que hacer en los botones de Imagen que se encargan de estas funciones debemos introducir los **CommandName** correctos:

- Botón de Editar debe tener el valor **Edit** en **CommandName**.
- Botón de Eliminar debe tener el valor **Delete** en **CommandName**.
- Botón de Guardar debe tener el valor **Save** en **CommandName**.
- Botón de Cancelar debe tener el valor **Cancel** en **CommandName**.
- Botón de Insertar Nuevo Registro debe tener el valor **AddNew** en **CommandName**.

Title_Id	Title	Type	Price	Pub_ID	Pubdate
as1111	1111	1111	11,2500	0736	27/01/2019 0:00:00
BU1032	The Busy Executive's Database Guide	business	19,0000	1389	12/06/1991 0:00:00
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	11,9500	1389	09/06/1991 0:00:00
BU2075	You Can Combat Computer Stress!	business	2,9900	0736	30/06/1991 0:00:00
BU7832	Straight Talk About Computers	business	19,9900	1389	22/06/1991 0:00:00
BU9900	You Can Combat Computer Stress!	business	2,9900	0736	30/06/1991 0:00:00
MC2222	Silicon Valley Gastronomic Treats	mod_cook	19,9900	0877	09/06/1991 0:00:00
MC3021	The Gourmet Microwave	mod_cook	2,9900	0877	18/06/1991 0:00:00
MC3026	The Psychology of Computer Cooking	UNDECIDED	2,9900	0877	10/09/2018 16:46:52
PC1035	But Is It User Friendly?	popular_comp	22,9500	1389	30/06/1991 0:00:00

6.- Debemos implementar ahora los eventos de **Edición**, **Borrado**, **Cancelación de la Edición**, **Salvado** e **Inserción de Nuevo Registro**.

6.1.- Para comenzar generaremos los eventos dentro de la etiqueta del **GridView**.

```
<asp:GridView ID="GridView1" runat="server" CellPadding="4" ForeColor="#333333" GridLines="None"
    ShowFooter="true"  ShowHeader="true" AutoGenerateColumns="false"
    OnRowCommand="GridView1_RowCommand"
    OnRowEditing="GridView1_RowEditing" OnRowCancelingEdit="GridView1_RowCancelingEdit"
    OnRowUpdating="GridView1_RowUpdating" OnRowDeleting="GridView1_RowDeleting">
```

Tendremos métodos para editar registros **OnRowEditing** en donde podremos asociarle un evento, para cancelar **OnRowCancelingEdit**, para eliminar **OnRowDeleting** y para salvar cambios **OnRowUpdating**. Como para insertar un nuevo registro no posee uno propio usaremos el método **OnRowCommand** para esta operación.

6.2.-Comenzaremos con el Evento generado por **OnRowEditing**:

Este se ejecuta antes de entrar en modo Edición, por lo que tendremos que marcar cual va a ser la fila a editar. Esto se hará con la propiedad **EditIndex** del **GridView** y se le asignará el valor **e.NewEditIndex**, para marcarla como fila a editar. Posteriormente debemos volver a establecer el enlace con el método **DataBind** del **DataGrid**. Para ello vamos a crear un método **CargarLibros**(), que establezca el enlace a través de la propiedad **DataSource** del **DataGrid** con el objeto que nos provee de los datos y genere el enlace a través del método **GridView1.DataBind();**

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    GridView1.EditIndex = e.NewEditIndex;
    CargarLibros();
}
```

```
private void CargarLibros()
{
    List<Libro> listaLibros = pLibros.getLibros();
    GridView1.DataSource = listaLibros;
    GridView1.DataBind();
}
```

6.3.- La cancelación de la edición **OnRowCancelingEdit**, tiene que deseleccionar la fila a editar, esto se hará estableciendo el valor **-1** a la propiedad **EditIndex** del **GridView**. Y se volverá a establecer el enlace para colocar todo como estaba inicialmente. Este evento se realizará al pulsar el botón de imagen de Cancelación.

```
protected void GridView1_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
{
    GridView1.EditIndex = -1;
    CargarLibros();
}
```

6.4.- Para finalizar el modo Edición realizaremos el evento correspondiente al método **OnRowUpdating**, que se realizará cuando pulse el botón de imagen de Salvar.

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    string title_Id = ((TextBox)GridView1.Rows[e.RowIndex].FindControl("TxtTitle_Id")).Text;
    string title = ((TextBox)GridView1.Rows[e.RowIndex].FindControl("TxtTitle")).Text;
    string type = ((TextBox)GridView1.Rows[e.RowIndex].FindControl("TxtType")).Text;
    string price = ((TextBox)GridView1.Rows[e.RowIndex].FindControl("TxtPrice")).Text;
    string pub_Id = ((TextBox)GridView1.Rows[e.RowIndex].FindControl("TxtPub_Id")).Text;
    string pubdate = ((TextBox)GridView1.Rows[e.RowIndex].FindControl("TxtPubdate")).Text;
    Libro l = new Libro(title_Id, title, type, Convert.ToDecimal(price), pub_Id, Convert.ToDateTime(pubdate));
    pLibros.actualizarLibro(l);
    GridView1.EditIndex = -1;
    CargarLibros();
}
```

Al entrar en modo edición podemos cambiar cualquiera de los valores del libro que se está editando. Al pulsar el icono de Salvar debemos tomar los datos de los cuadros de texto de edición, crear con ellos un nuevo objeto del tipo Libro y actualizar los datos de este libro.

- **1PASO:** Para acceder a los cuadros de texto del modo edición lo haremos a través de la colección de filas del **GridView** (**GridView1.Rows**). La fila de la que debemos sacar los datos será la que nos indique la propiedad RowIndex del parámetro e del evento (**GridView1.Rows[e.RowIndex]**). Dentro de esta fila debemos buscar el cuadro de texto con el método FindControl pasándole el nombre del cuadro de texto a buscar (**GridView1.Rows[e.RowIndex].FindControl("nombreCuadroTexto")**) y finalmente cogeremos la propiedad Text de dicho cuadro de texto. Para esto como FindControl devuelve un elemento de tipo Control debemos convertirlo a TextBox. Esto lo haremos con una operación de Casting.
  (TextBox)(GridView1.Rows[e.RowIndex].FindControl("nombreCuadroTexto")).Text

- **2PASO**: Crearemos un objeto con los datos de los cuadros de texto.

  Libro l=new Libro(title_Id,title,type,Convert.ToDecimal(price),pub_Id,Convert.ToDateTime(pubdate));

- **3PASO**: Pasaremos estos datos a un método para modificar el libro en cuestión.
  Llamaremos al método **actualizarLibro** de la clase **N_LIBROS** del proyecto de **NEGOCIO**.

  pLibros.actualizarLibro(l);

```
public void actualizarLibro(Libro l)
{
    nLibros.actualizarLibro(l);
}
```

Este a su vez llamará al método de actualizarLibro de la clase D_LIBROS del proyecto de DATOS.

```
public void actualizarLibro(Libro l)
{
    SqlCommand cmdLibro = new SqlCommand("Update Titles set Title=@title,Type=@type,Price=@price,Pub_ID=@pub_id,Pubdate=@pubdate  WHERE  Title_ID=@title_Id", cnn);
    cmdLibro.Parameters.AddWithValue("@title_Id", l.Title_Id);
    cmdLibro.Parameters.AddWithValue("@title", l.Title);
    cmdLibro.Parameters.AddWithValue("@type", l.Type);
    cmdLibro.Parameters.AddWithValue("@price", l.Price);
    cmdLibro.Parameters.AddWithValue("@pub_id", l.Pub_id);
    cmdLibro.Parameters.AddWithValue("@pubdate", l.Pubdate);
    abrir();
    cmdLibro.ExecuteNonQuery();
    cerrar();
}
```

6.5.- Seguiremos con el método para eliminar registros **OnRowDeleting**. El evento asociado se ejecutará cuando pulsemos el botón de imagen de eliminar.

A la hora de borrar debemos saber cuál es el valor de la clave principal del libro que debemos eliminar. Para acceder a este dato no podemos usar los cuadros de texto que teníamos en el modo edición, así que tenemos que acceder de otra forma. Esto lo haremos mediante la colleción **DataKeys**. Esta colección hacer referencia a la propiedad **DataKeyNames** del GridView (que de momento no hemos usado). Esta propiedad solo indica cual de los datos con los que vamos a trabajar enlazados al control **GridView** es la clave principal. En nuestro caso será el campo **Title_Id**.

```
<asp:GridView ID="GridView1" runat="server" CellPadding="4" ForeColor="#333333" GridLines="None"
    ShowFooter="true"  ShowHeader="true" AutoGenerateColumns="false"
    OnRowCommand="GridView1_RowCommand"
    OnRowEditing="GridView1_RowEditing" OnRowCancelingEdit="GridView1_RowCancelingEdit"
    OnRowUpdating="GridView1_RowUpdating" OnRowDeleting="GridView1_RowDeleting" DataKeyNames="Title_Id">
```

Dentro de la colección **DataKeys**, le indicaremos mediante **e.RowIndex** de qué fila debemos coger la clave principal y por último accederemos a su valor mediante la propiedad **Value**.

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    string title_Id = GridView1.DataKeys[e.RowIndex].Value.ToString();
    pLibros.eliminarLibro(title_Id);
    CargarLibros();
}
```

De esta forma ya tenemos el valor de la clave principal del libro a borrar. Ahora solo tenemos que llamar a un método para eliminar dicho libro. Esto lo haremos como en el caso anterior llamando a un método de la clase N_LIBROS y esta a su vez a uno de la clase D_LIBROS.

```
public void eliminarLibro(string Title_Id)
{
    nLibros.eliminarLibro(Title_Id);
}
```

```
public void eliminarLibro(string Title_Id)
{
    SqlCommand cmdLibro = new SqlCommand("Delete FROM Titles  WHERE  Title_ID=@title_Id", cnn);
    cmdLibro.Parameters.AddWithValue("@title_Id", Title_Id);

    abrir();
    cmdLibro.ExecuteNonQuery();
    cerrar();
}
```

Por último, realizaremos el método **OnRowCommand** para la inserción de nuevos registros.

```
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
    switch (e.CommandName)
    {
        case "addnew":
            string title_Id = ((TextBox)GridView1.FooterRow.FindControl("TxtTitle_IdFooter")).Text;
            string title = ((TextBox)GridView1.FooterRow.FindControl("TxtTitleFooter")).Text;
            string type = ((TextBox)GridView1.FooterRow.FindControl("TxtTypeFooter")).Text;
            string price = ((TextBox)GridView1.FooterRow.FindControl("TxtPriceFooter")).Text;
            string pub_Id = ((TextBox)GridView1.FooterRow.FindControl("TxtPub_IdFooter")).Text;
            string pubdate = ((TextBox)GridView1.FooterRow.FindControl("TxtPubdateFooter")).Text;
            Libro l = new Libro(title_Id, title, type, Convert.ToDecimal(price), pub_Id, Convert.ToDateTime(pubdate));
            pLibros.insertarLibro(l);

            CargarLibros();
            break;
    }

}
```

Como este es un método general debemos ejecutarlo solo cuando hayamos pulsado el botón de imagen de añadir nuevo registro y eso será si el **CommandName** el valor "**addnew**" que le habíamos dado dentro del **GridView**.

- **1PASO**: Acceder a los cuadros de texto en donde insertaremos los valores del nuevo libro. Esto lo haremos accediendo a la fila del pie del **GridView** con la propiedad **FooterRow** y allí buscaremos con el método **FindControl** el nombre que le pasemos como parámetro. Al igual que en el caso de la actualización el método **FindControl** nos devuelve un elemento del tipo Control, así que lo convertiremos al tipo **TextBox** y accederemos a su propiedad **Text**.
- **2PASO**: Crearemos el libro con los valores del nuevo registro.
- **3PASO**: Llamaremos al método **insertarLibro** para insertarlo en el origen de datos. Esto lo haremos como anteriormente pasando por la capa de **NEGOCIO** y llegando a la de **DATOS**.

```
public void insertarLibro(Libro l)
{
    nLibros.insertarLibro(l);
}
```

```
public void insertarLibro(Libro l)
{
    SqlCommand cmdnuevoLibro = new SqlCommand("INSERT into titles(title_id,title,type,price,pub_id,pubdate)" +
        " values (@id,@titulo,@tipo,@precio,@editor,@fecha)", cnn);
    cmdnuevoLibro.Parameters.AddWithValue("@id", l.Title_Id);
    cmdnuevoLibro.Parameters.AddWithValue("@titulo", l.Title);
    cmdnuevoLibro.Parameters.AddWithValue("@tipo", l.Type);
    cmdnuevoLibro.Parameters.AddWithValue("@precio", l.Price);
    cmdnuevoLibro.Parameters.AddWithValue("@editor", l.Pub_id);
    cmdnuevoLibro.Parameters.AddWithValue("@fecha", l.Pubdate);
    abrir();
    cmdnuevoLibro.ExecuteNonQuery();
    cerrar();
}
```