


	<b>Instituto Gubernamental España Jesús Milla Selva</b> Centro de Informática	
<b>II SEMESTRE</b> Asignatura: <b>Programación IV</b>		
<u><b>UNIDAD Nº 3:</b></u>  <b>Recursos de ASP.Net</b>	<u><b>TEMA:</b></u>  <b>Upload de archivos</b>	

## Objetivos:

- Realizar la carga correcta de archivos al servidor.
- Validar la existencia de archivos con el mismo nombre.
- Controlar las diferentes propiedades de un archivo.

## Introducción

Una actividad muy común en un sitio web es el envío de archivos desde el cliente y su almacenamiento en el servidor.

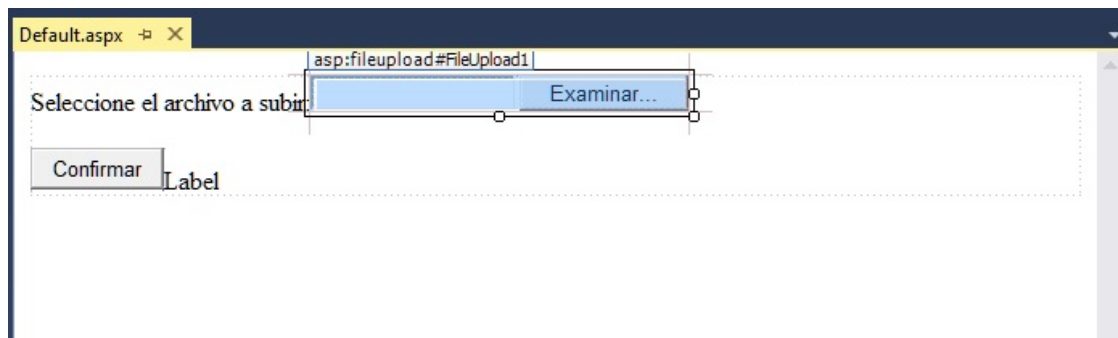
## Upload

### Componente FileUpload

El componente FileUpload encapsula el envío y recepción de un archivo en el servidor web.

Confeccionaremos una serie de páginas web para aprender a utilizar los métodos y propiedades de la clase FileUpload (Crear un sitio web llamado ejercicio017)

Crear un webform (Default.aspx) e implementar la siguiente interfaz:



Disponemos en el webform un objeto de la clase FileUpload que se encuentra en la pestaña de componentes “Estándar”.

Para el evento clic del botón "confirmar" implementamos el siguiente código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        FileUpload1.SaveAs(Server.MapPath(".") + "/" +
FileUpload1.FileName);
        Label1.Text = "Archivo subido";
    }
}
```

El método SaveAs permite grabar el archivo que se subió al servidor. Debemos indicarle el camino del directorio donde se almacena y el nombre del archivo. Para obtener el path donde se almacena la página ASPX actual el objeto Server tiene el método MapPath y para obtener el nombre del archivo la propiedad llamada FileName.

Con esta única línea tenemos registrado en el servidor el archivo que se envió desde el navegador.



## Almacenar el archivo en un subdirectorio

Es una buena costumbre evitar almacenar los archivos que suben los usuarios en la misma carpeta donde se encuentran las páginas dinámicas ASPX.

Para almacenar los archivos en otro directorio primero debemos crear el directorio (por ejemplo, creemos una carpeta llamada 'imagenes' en el directorio donde se aloja el sitio, presionamos el botón izquierdo dentro del "Explorador de soluciones" y seleccionamos Agregar -> Nueva carpeta)

Creamos un nuevo webform (Default2.aspx), definimos una interfaz visual idéntica al primer formulario y escribimos el siguiente código para el evento clic del objeto Button:

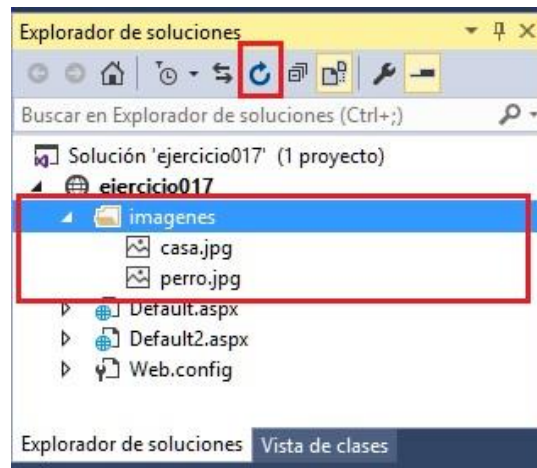
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        FileUpload1.SaveAs(Server.MapPath(".") + "/imagenes/"
+ FileUpload1.FileName);
        Label1.Text = "Archivo subido";
    }
}
```

La carpeta imágenes debemos crearla en forma manual desde el administrador de archivos del sistema operativo. Luego cada archivo que se suba al servidor se almacenará en dicha carpeta.



Ejecutemos la página y subamos un par de archivos. Si no aparecen los archivos hay que presionar el ícono de "Actualizar" del "Explorador de soluciones".

### Mostrar propiedades del archivo subido

Crearemos un tercer webform (Default3.aspx) para ver como accedemos a distintas propiedades de la clase FileUpload.

Disponemos sobre el webform un objeto de la clase FileUpload, un Button y cuatro Label.

Mostraremos el tamaño del archivo en bytes, el nombre del archivo y finalmente el tipo de archivo.

El código fuente para el evento clic es:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default3 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        FileUpload1.SaveAs(Server.MapPath(".") + "/imagenes/"
+ FileUpload1.FileName);
        Label1.Text = "Archivo subido";
    }
}
```

```
Label2.Text =  
FileUpload1.PostedFile.ContentLength.ToString();  
Label3.Text = FileUpload1.FileName;  
Label4.Text = FileUpload1.PostedFile.ContentType;  
}  
}
```

La propiedad `PostedFile` del control `FileUpload` almacena en:

- `ContentLength` (el tamaño del archivo en bytes)
- `ContentType` (El tipo de archivo)



El tamaño del archivo nos puede ser útil si queremos limitar el peso del mismo.

### Validar la existencia de otro archivo con el mismo nombre

Puede suceder que en la carpeta donde se almacena el archivo exista otro con el mismo nombre. Esto conlleva a que se pise el archivo antiguo por el nuevo.

Veamos otro webform donde validamos que no exista otro archivo con el mismo nombre, en caso afirmativo no permitimos su almacenamiento e informamos de tal situación:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default4 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (File.Exists(Server.MapPath(".") + "/" + FileUpload1.FileName))
        {
            Label1.Text = "Existe un archivo con dicho nombre en el servidor";
        }
        else
        {
            FileUpload1.SaveAs(Server.MapPath(".") + "/" + FileUpload1.FileName);
            Label1.Text = "Archivo subido";
        }
    }
}
```

El método estático Exists retorna true si ya existe un archivo con el nombre que le pasamos como parámetro, en caso negativo retorna false.

En caso de que no exista el archivo procedemos a efectuar la registración del mismo en la carpeta de nuestro sitio.

## Ejercicio resuelto

En la base de datos 'base1' crear las tablas:


```

autos (patente char(6) primary key,


       propietario varchar(50),
       precio float,
       codigomarca int,
       foto varchar(100),
       modelo int)

marcas (codigo int identidad primary key,
        descripcion varchar(30)
        )

```

autos			
	Nombre de columna	Tipo de datos	Permitir val...
	patente	char(6)	<input type="checkbox"/>
	propietario	varchar(50)	<input checked="" type="checkbox"/>
	precio	float	<input checked="" type="checkbox"/>
	codigomarca	int	<input checked="" type="checkbox"/>
	foto	varchar(100)	<input checked="" type="checkbox"/>
	modelo	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

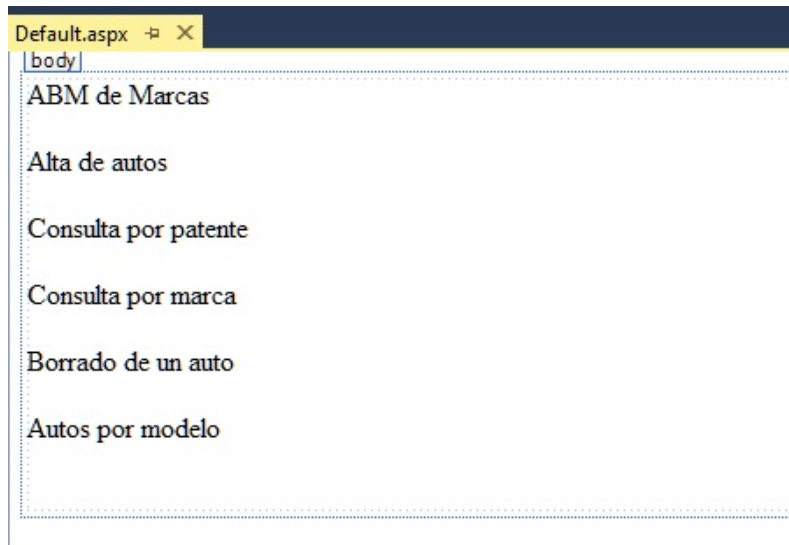
marcas			
	Nombre de columna	Tipo de datos	Permitir val...
	codigo	int	<input type="checkbox"/>
	descripcion	varchar(30)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

No olvidar de definir el campo codigo de la tabla marcas de tipo identidad para que se genere automáticamente.

Desde el Visual Studio.Net proceder a crear un sitio web vacío (ejercicio018) y agregar el primer Web Form que contendrá el menú de opciones para efectuar:

- Confeccionar el mantenimiento de la tabla marcas (altas, bajas y modificaciones (utilizar un GridView)
- Confeccionar una página para efectuar el alta de autos.
- Consulta de un auto ingresando su patente (mostrar todos los datos, incluido la foto)
- Seleccionar de un DropDownList una marca y luego mostrar todas las fotos de autos de dicha marca.
- Implementar el borrado de un auto ingresando su patente.
- Ingresar un rango de años y luego mostrar todas las fotos de autos en dicho rango.

La página Default.aspx tiene 6 HyperLink:



#### Punto a

- a. Confeccionar el mantenimiento de la tabla marcas (altas, bajas y modificaciones) utilizar un GridView

Crear un Web Form llamado abmmarcas.aspx el cual debe permitir implementar las altas, bajas y modificaciones de la tabla marcas.

Mediante un GridView implementamos las bajas y modificaciones. Codificamos por otro lado el alta.

Debemos disponer un SQLDataSource asociado a la tabla marcas y configurar las propiedades ConnectionString, InsertQuery y SelectQuery. También disponemos un GridView e inicializamos la propiedad "Elegir origen de datos" con el SQLDataSource que hemos insertado.

La interfaz visual a implementar es:



abmmarcas.aspx\* X

codigo	descripcion
0	abc
1	abc
2	abc
3	abc
4	abc

Ingrese descripción de la marca:

SqlDataSource - dsmarcas

Para el evento click del botón "Alta" debemos:

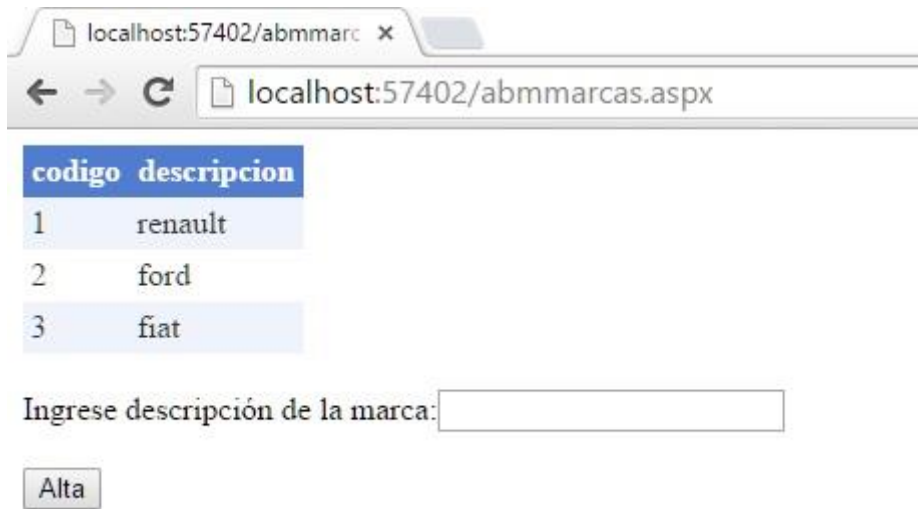
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class abmmarcas : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        dsmarcas.InsertParameters["descripcion"].DefaultValue
= TextBox1.Text;
        dsmarcas.Insert();
        TextBox1.Text = "";
    }
}
```

Luego la interfaz en tiempo de ejecución debe quedar:



localhost:57402/abmmarc x

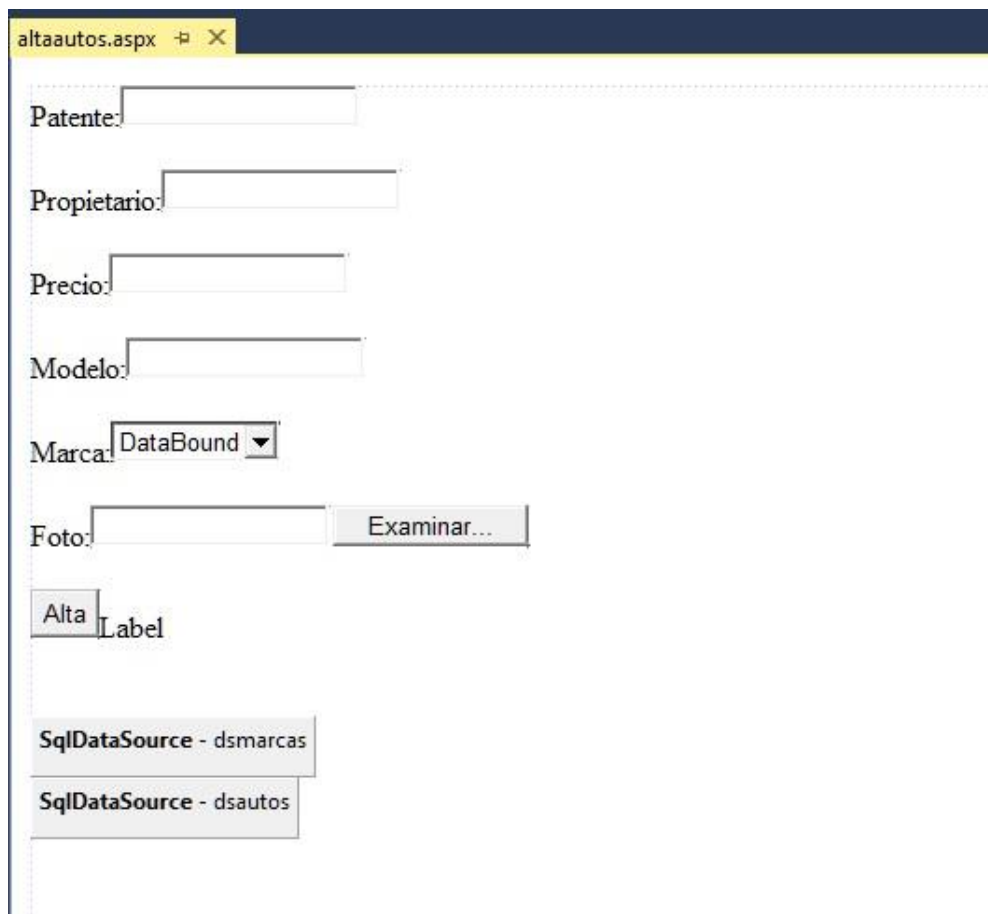
localhost:57402/abmmarcas.aspx

codigo	descripcion
1	renault
2	ford
3	fiat

Ingrese descripción de la marca:

#### Punto b

- b. Confeccionar una página para efectuar el alta de autos donde implementaremos el upload de la foto. Debemos crear una interfaz visual similar a esta (llamar al Web Form altaautos.aspx):



altaautos.aspx

Patente:

Propietario:

Precio:

Modelo:

Marca:

Foto:

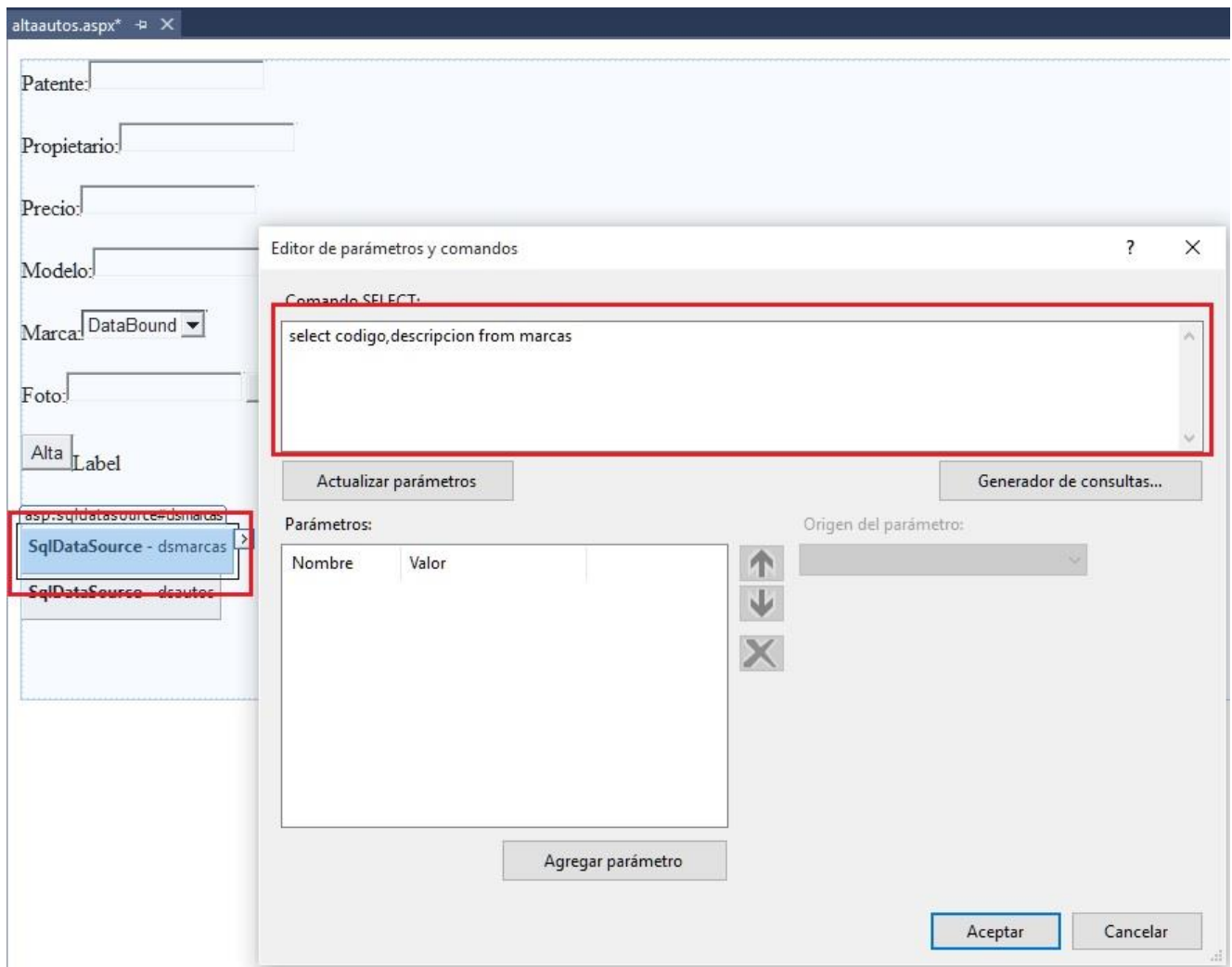
Label

SqlDataSource - dsmarcas

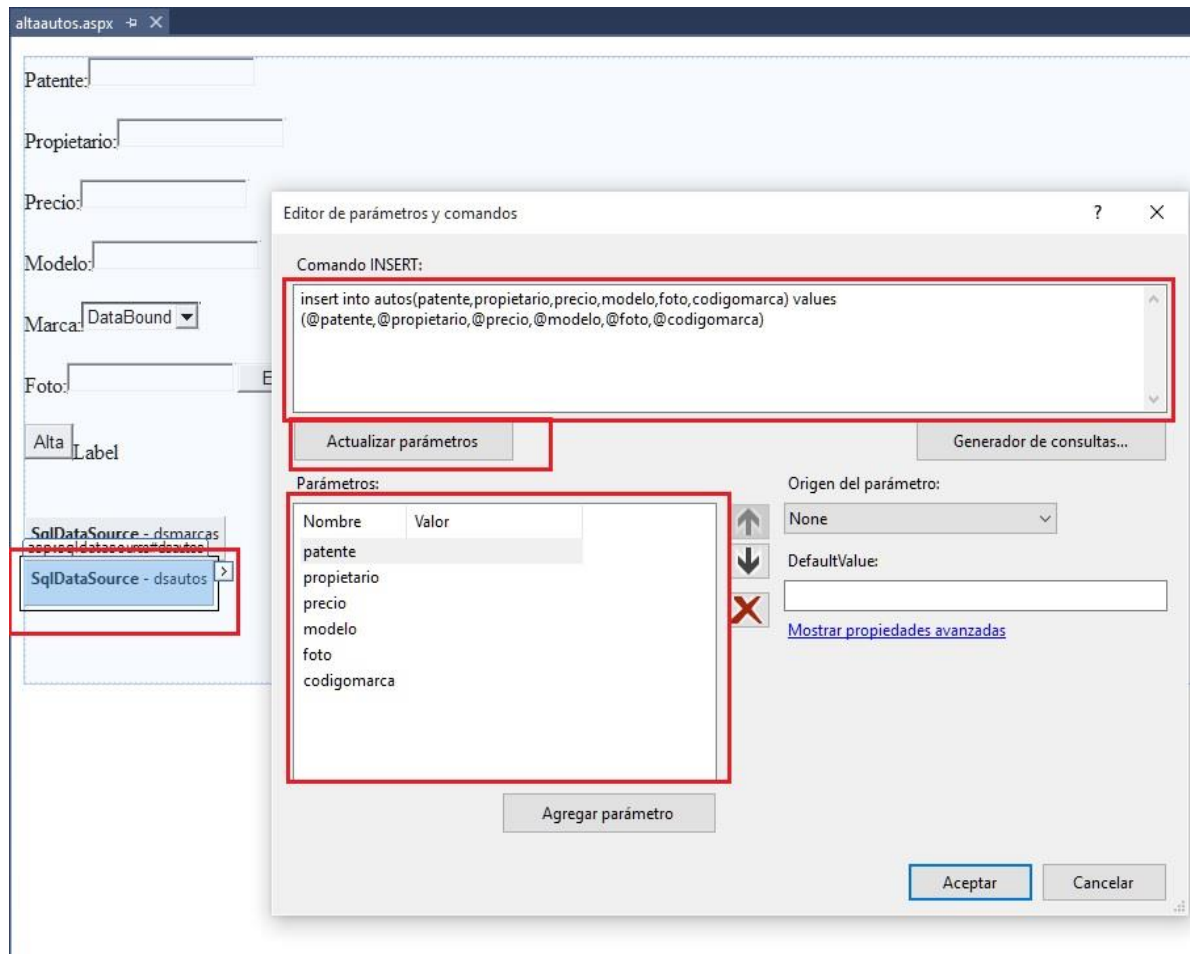
SqlDataSource - dsautos

Configuramos el `SqlDataSource` de marcas: `dsmarcas` asociándolo con la tabla `marcas`, luego el `DropDownList1` lo vinculamos a este `DataSource` para que muestre todas las marcas.

El `SelectQuery` del objeto `dsmarcas`:



Por otro lado, creamos el `dsautos` y configuramos el `InsertQuery`:



Para el evento click del alta procedemos a inicializar los parámetros y a efectuar el insert en la tabla autos.

Además, hacemos el upload del archivo de la foto con el objetivo de que la foto quede almacenada en forma permanente en el servidor:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class altaautos : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        FileUpload1.SaveAs(Server.MapPath(".") + "/" +
FileUpload1.FileName);
        dsautos.InsertParameters["patente"].DefaultValue =
TextBox1.Text;
        dsautos.InsertParameters["propietario"].DefaultValue
= TextBox2.Text;
        dsautos.InsertParameters["precio"].DefaultValue =
TextBox3.Text;
        dsautos.InsertParameters["modelo"].DefaultValue =
TextBox4.Text;
        dsautos.InsertParameters["codigomarca"].DefaultValue
= DropDownList1.SelectedValue;
        dsautos.InsertParameters["foto"].DefaultValue =
FileUpload1.FileName;
        dsautos.Insert();
        Label1.Text = "Los datos fueron cargados";
        TextBox1.Text = "";
        TextBox2.Text = "";
        TextBox3.Text = "";
        TextBox4.Text = "";
    }
}
```

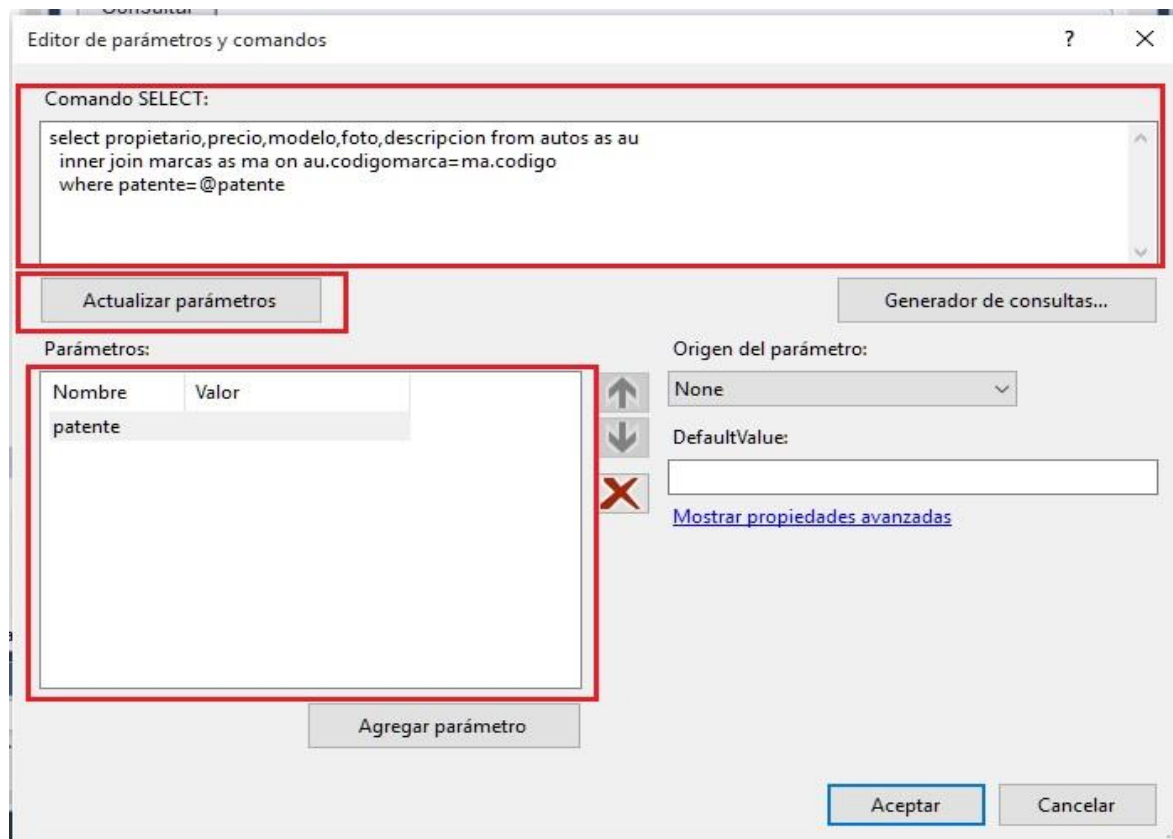
## Punto c

### c. Consulta de un auto ingresando su patente (mostrar todos los datos, incluido la foto)

Creamos un nuevo Web Form llamado: 'consultaporpatente.aspx' e implementamos una interfaz que nos permita ingresar la patente de un auto y nos muestre mediante una Label y un objeto de la clase Image los datos de dicho vehículo:



Configuramos la propiedad SelectQuery del dsautos con el siguiente comando SQL para recuperar los datos:



Editor de parámetros y comandos

Comando SELECT:

```
select propietario,precio,modelo,foto,descripcion from autos as au
inner join marcas as ma on au.codigomarca=ma.codigo
where patente=@patente
```

Actualizar parámetros

Generador de consultas...

Parámetros:

Nombre	Valor
patente	

Origen del parámetro:

None

DefaultValue:

[Mostrar propiedades avanzadas](#)

Agregar parámetro

Aceptar Cancelar

Recordar que cada vez que creamos un parámetro en el comando SQL debemos presionar el botón "Actualizar parámetros".

El código del evento click de "Consultar" queda definido por:

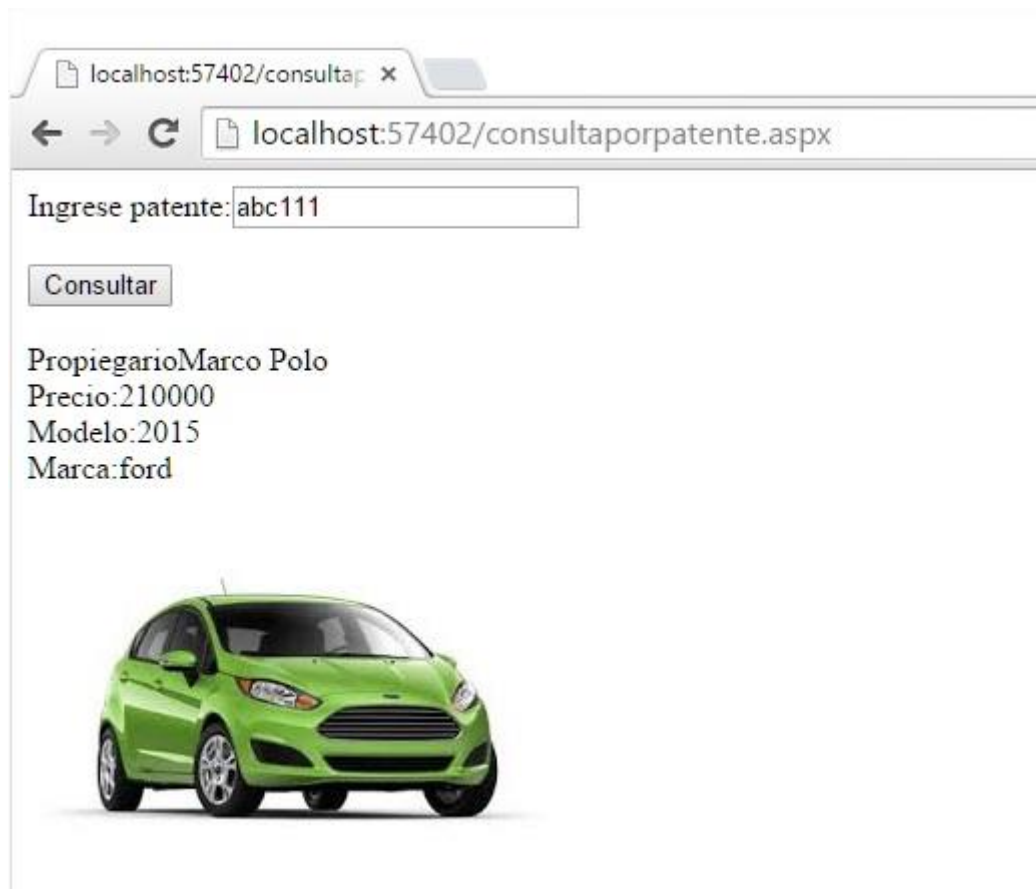
```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class consultaporpatente : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        dsautos.SelectParameters["patente"].DefaultValue =
        TextBox1.Text;
        dsautos.DataSourceMode =
        SqlDataSourceMode.DataReader;
        SqlDataReader registro =
        (SqlDataReader)dsautos.Select(DataSourceSelectArguments.Empty
        );
        if (registro.Read())
        {
            Image1.ImageUrl = registro["foto"].ToString();
            Label1.Text = "Propietario" +
            registro["propietario"] + "<br>" +
            "Precio:" + registro["precio"] +
            "<br>" +
            "Modelo:" + registro["modelo"] +
            "<br>" +
            "Marca:" + registro["descripcion"];
        }
        else
        {
            Label1.Text = "No existe un auto con dicha
            patente";
        }
    }
}
```

Y tenemos como resultado una página en tiempo de ejecución similar a:



#### Punto d

d. Seleccionar de un DropDownList una marca y luego mostrar todas las fotos de autos de dicha marca.

Creamos un Web Form llamado fotospormarca.aspx y desarrollamos una interfaz similar a la siguiente:





Configuramos el SelectQuery del dsmarcas con la consulta de la tabla marcas:

```
select codigo, descripcion from marcas
```

Por otro lado, configuramos el origen de datos del DropDownList1 para que se muestren todas las marcas y podamos posteriormente recuperar el código de la marca seleccionada.

También iniciamos la propiedad SelectQuery del objeto dsautos:

```
select foto from autos where codigomarca=@codigomarca
```

Luego cuando se presiona el botón "Mostrar" debemos recuperar todas las fotos de autos que pertenecen a la marca seleccionada:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class fotospormarca : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
```

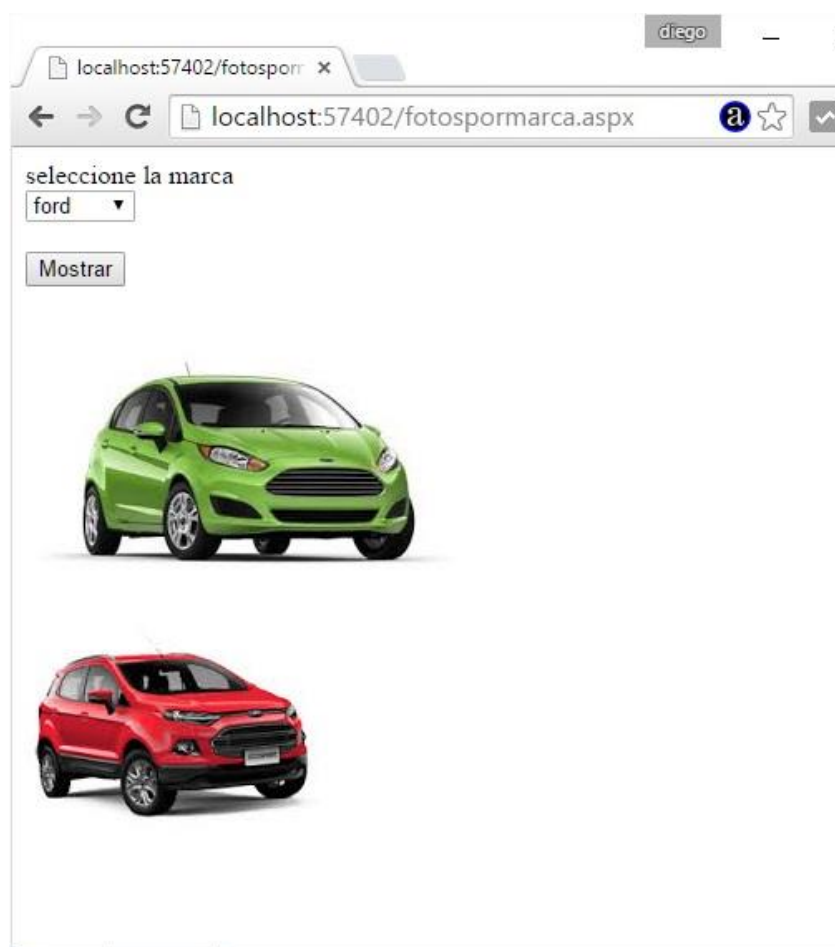
```

        dsautos.SelectParameters["codigomarca"].DefaultValue
= DropDownList1.SelectedValue;
        dsautos.DataSourceMode =
SqlDataSourceMode.DataReader;
        SqlDataReader registros =
(SqlDataReader)dsautos.Select(DataSourceSelectArguments.Empty
);
        this.Label1.Text = "";
        while (registros.Read() == true)
        {
            Label1.Text = Label1.Text + "<img src=\"\" +
                                registros[\"foto\"] + \"\"><br>";
        }
    }
}

```

Como podemos ver debemos generar los elementos HTML img con cada una de las fotos (no podemos utilizar un objeto Image ya que pueden haber varios autos que pertenecen a la misma marca)

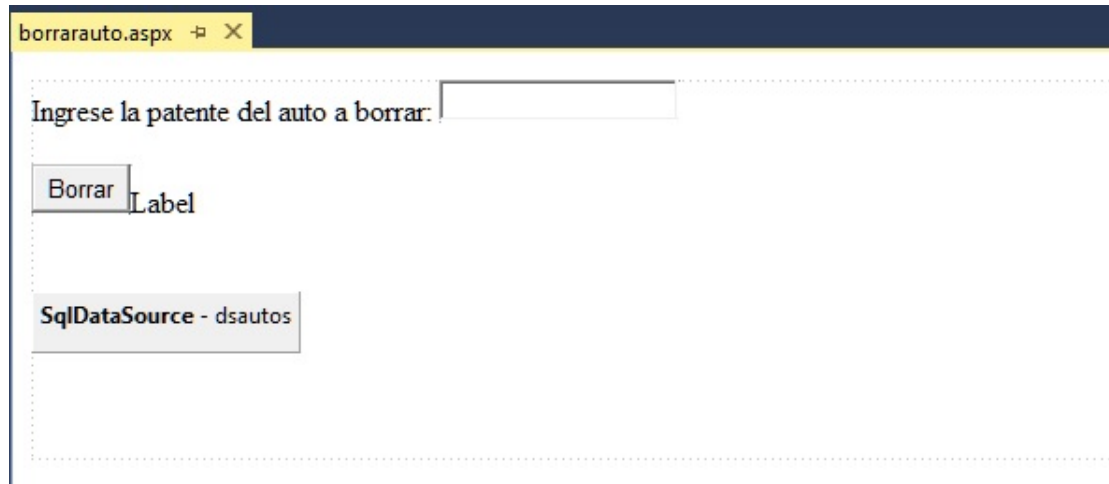
En pantalla en tiempo de ejecución se debe mostrar algo similar a:



## Punto e

e. Implementar el borrado de un auto ingresando su patente.

Creemos un Web Form llamado "borrarauto.aspx" y diseñamos una interfaz similar a esta:



Debemos consultar la tabla autos para recuperar el nombre del archivo almacenado del auto, para ello inicializamos la propiedad SelectQuery con la siguiente consulta:

```
select foto from autos where patente=@patente
```

Luego también debemos implementar la propiedad DeleteQuery:

```
delete from autos where patente=@patente
```

Cuando se presiona el botón borrar procedemos a ejecutar los dos comandos SQL:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class borrarauto : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
```

```

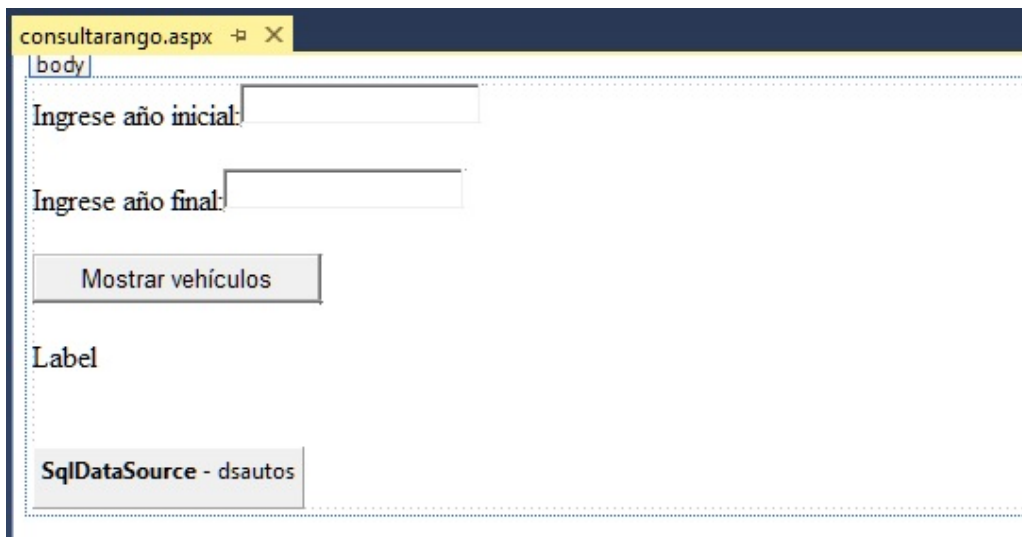
        dsautos.DeleteParameters["patente"].DefaultValue =
TextBox1.Text;
        int cant;
        cant = dsautos.Delete();
        if (cant == 0)
            this.Label1.Text = "No existe un auto con el
valor de patente ingresado";
        else
        {
            this.Label1.Text = "Se borró el auto con dicha
patente";
            dsautos.SelectParameters["patente"].DefaultValue
= TextBox1.Text;
            dsautos.DataSourceMode =
SqlDataSourceMode.DataReader;
            SqlDataReader registro;
            registro =
(SqlDataReader)dsautos.Select(DataSourceSelectArguments.Empty
);
            if (registro.Read())
                File.Delete(Server.MapPath(".") + "/" +
registro["foto"]);
        }
    }
}

```

## Punto f

f. Ingresar un rango de años y luego mostrar todas las fotos de autos en dicho rango.

Creamos un Web Form llamado "consultarango.aspx" con una interfaz similar a:



The screenshot shows the Visual Studio IDE with the 'consultarango.aspx' web form in design view. The form layout includes:

- Two text input fields: 'Ingrese año inicial' and 'Ingrese año final'.
- A button labeled 'Mostrar vehículos'.
- A 'Label' control.
- A 'SqlDataSource - dsautos' data source control.

Configuramos la propiedad `SelectQuery` del `SqlDataSource` con el comando SQL:

```
select foto from autos where modelo>=@modelo1 and
modelo<=@modelo2
```

Para el evento click del botón procedemos a mostrar todas las fotos de autos cuyos modelos se encuentran comprendidos entre los dos valores ingresados:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class consultarango : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {

        dsautos.SelectParameters["modelo1"].DefaultValue =
TextBox1.Text;
        dsautos.SelectParameters["modelo2"].DefaultValue =
TextBox2.Text;
        dsautos.DataSourceMode =
SqlDataSourceMode.DataReader;
        SqlDataReader registros =
(SqlDataReader)dsautos.Select(DataSourceSelectArguments.Empty
);

        Label1.Text = "";
        while (registros.Read() == true)
        {
            Label1.Text = Label1.Text + "<img src=\"\" +
                registros["foto"] +
                \"\"><br><br>";
        }
    }
}
```

## ACTIVIDAD

Se tienen las siguientes tablas:

```
libros (codigo, titulo, descripcion, fototapa, codigotema)
temas (codigo, descripcion)
```

- Confeccionar el alta de las tablas libros y temas (permitir seleccionar el tema mediante una lista) Hacer el Upload de la imagen al servidor. Validar que se ingresen obligatoriamente el título, descripción y selección de un archivo.
- Ingresar por teclado el código de un libro y luego mostrar la foto de la tapa.

### Puntaje de evaluación

Alta de dos tablas (libros y temas)	3%
Upload de la imagen al servidor	3%
Validación de datos obligatorios (título, descripción y selección de un archivo)	3%
Muestra la foto de portada de libro a partir de código escrito por el teclado	3%
<b>Total</b>	<b>12%</b>