

PROGRAMACIÓN POR CAPAS

La **programación por capas** es una arquitectura [cliente-servidor](#) en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.



La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas)

Capas y niveles

1. **Capa de presentación:** es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.
2. **Capa de negocio:** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.
3. **Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más [ordenadores](#). Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

- **Presentación.** (Conocida como capa Web en aplicaciones Web o como capa de usuario en Aplicaciones Nativas)
- **Lógica de Negocio.** (Conocida como capa Aplicativa)
- **Datos.** (Conocida como capa de Base de Datos)

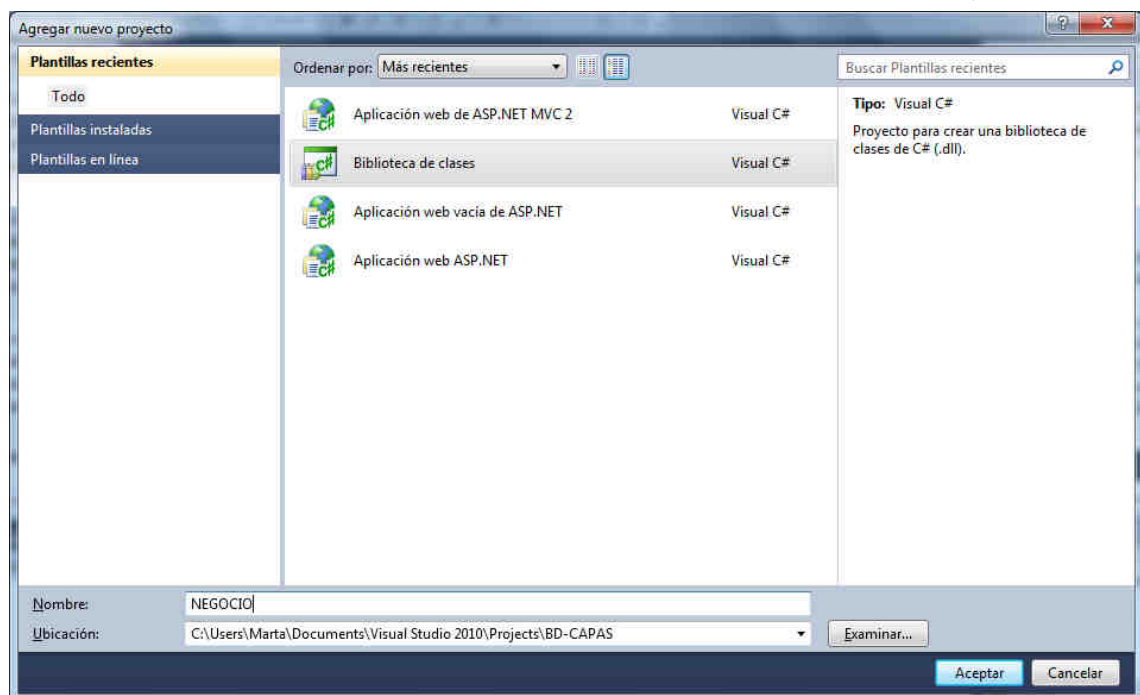
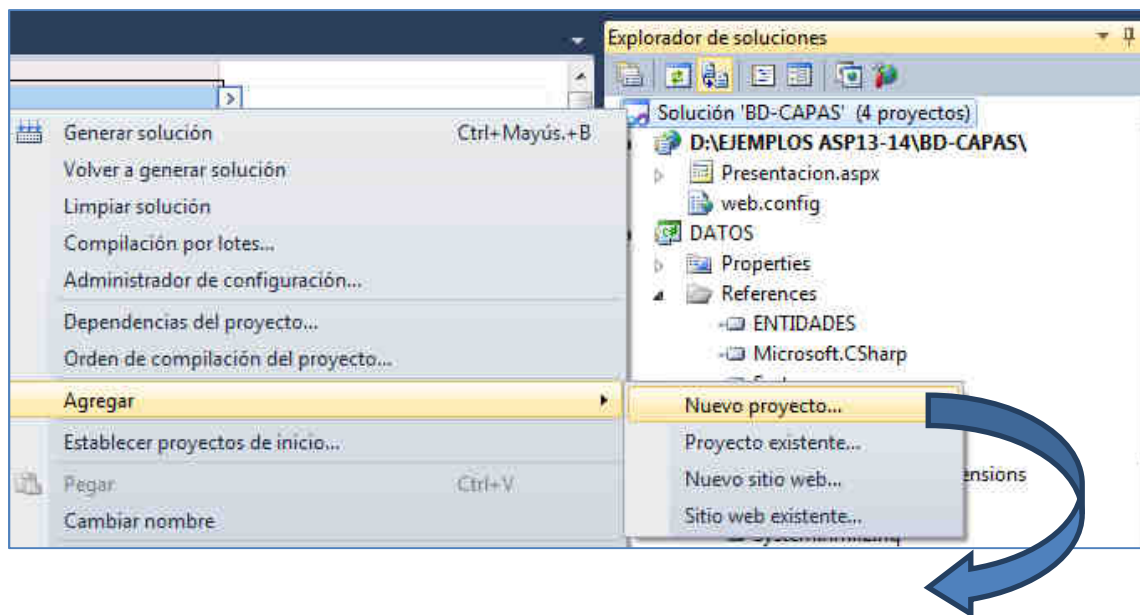
En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y *un nivel*.
- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos ordenadores (presentación+lógica por un lado; lógica+datos por el otro lado). Se dice que la arquitectura de la solución es de tres capas y *dos niveles*.

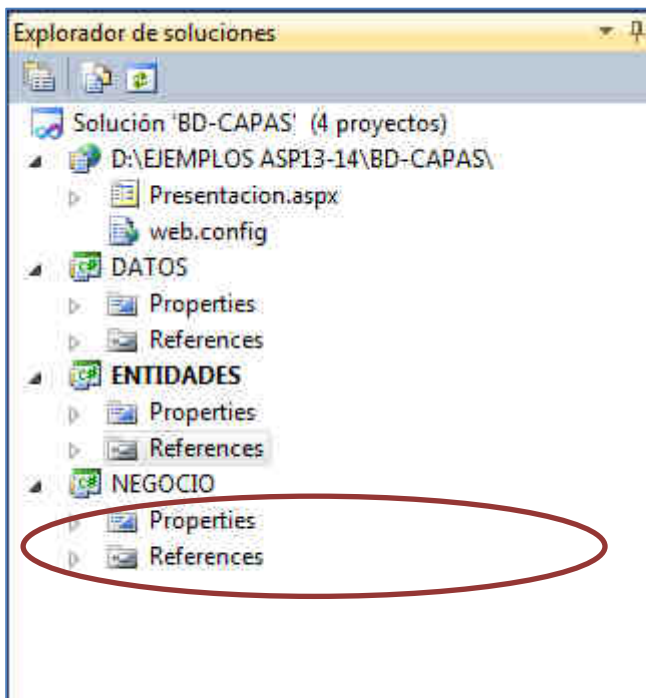
Creación de un Proyecto por Capas

1

.- Crearemos un Nuevo Sitio Web Vacío y dentro de él, como siempre, un **Web Form** que denominaremos como queramos. Crearemos dentro de la Solución tres nuevos proyectos de tipo Biblioteca de Clases denominados ENTIDADES, DATOS y NEGOCIO. En ambos tres **borraremos la Clase por defecto** que crea VWD.



De tal forma que quedará el nuestro proyecto de la siguiente forma:



La capa de DATOS será que acceda directamente a la Base de Datos. Ejecute la conexión y acceda a la BD devolviendo lo que necesitemos de ella o haciendo sentencias de acción sobre ella.

La capa de Presentación será la encargada de presentar los controles para que los usuarios interactúen con el sistema.

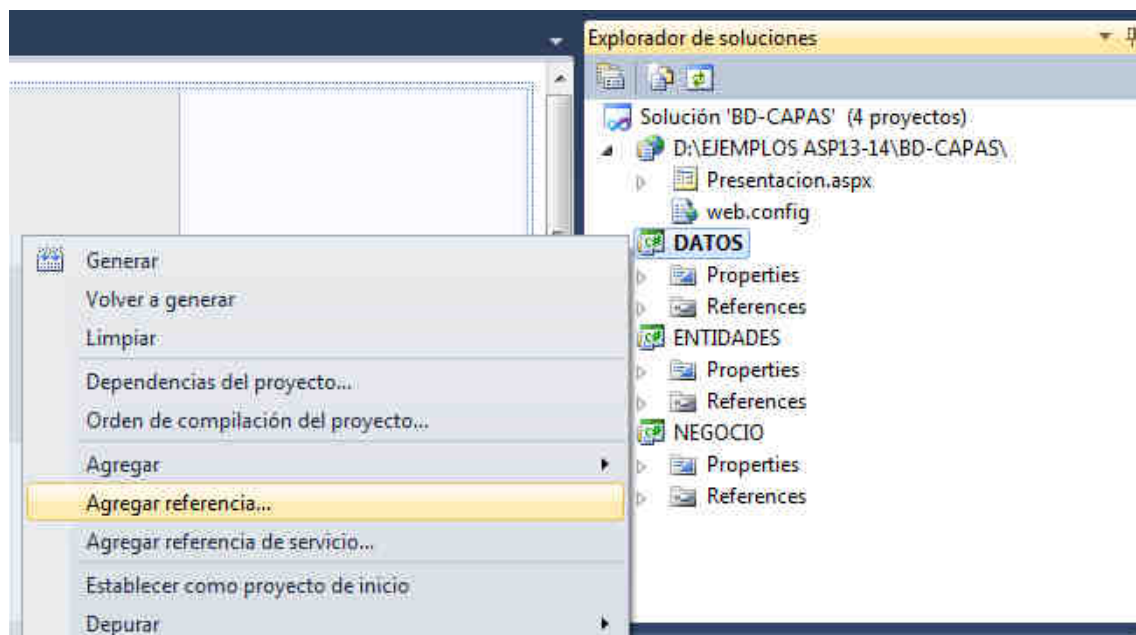
La capa de NEGOCIO hará de intermediaria entre la capa de presentación y la capa de Datos.

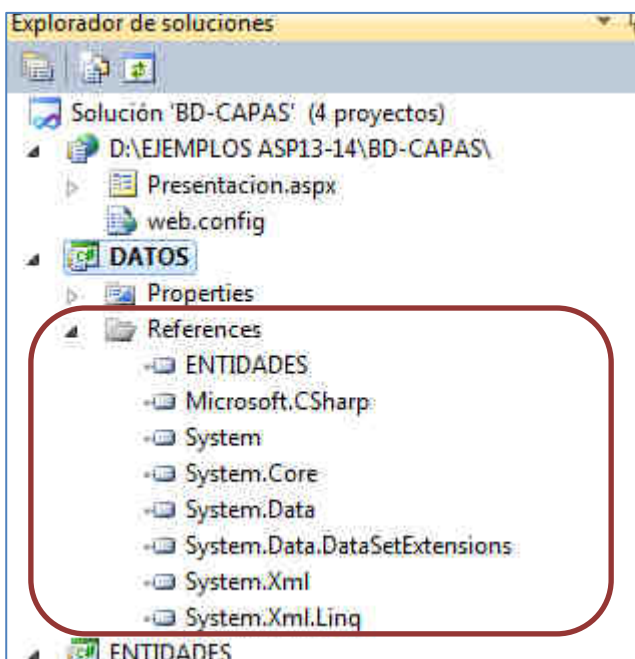
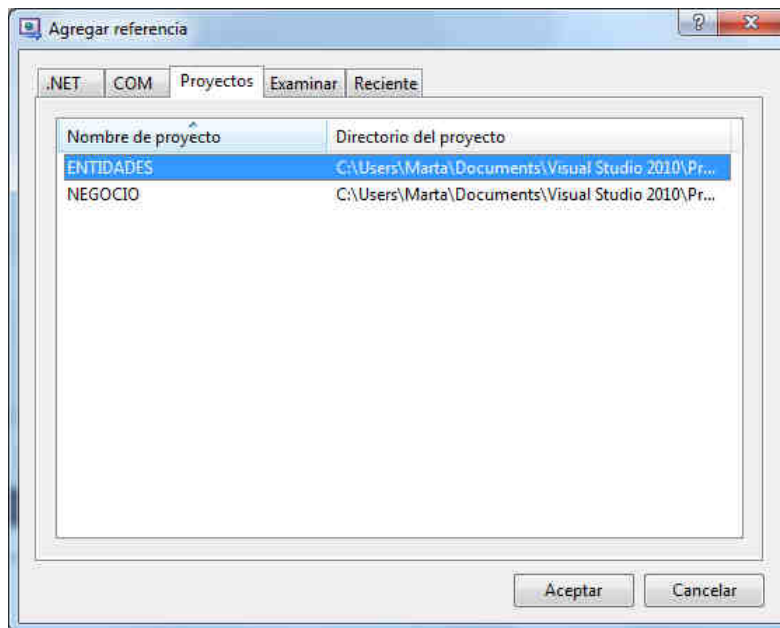
El proyecto de ENTIDADES se encargará de definir las clases con las que vamos a trabajar.

2

.- Ahora nos toca agregar las referencias.

Lo primero que haremos es agregar la referencia **del proyecto de Entidades en las tres capas (proyectos, DATOS, NEGOCIO,PRESENTACIÓN)**.

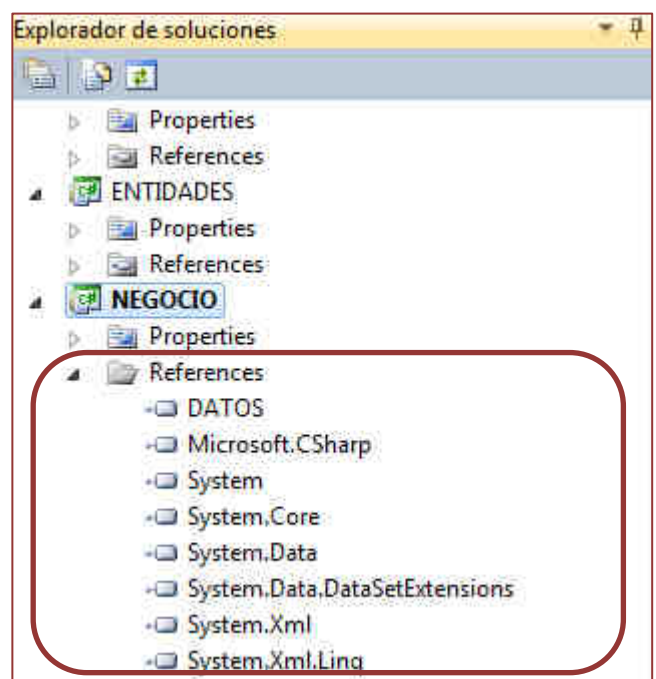
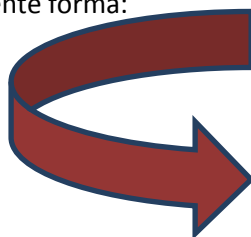




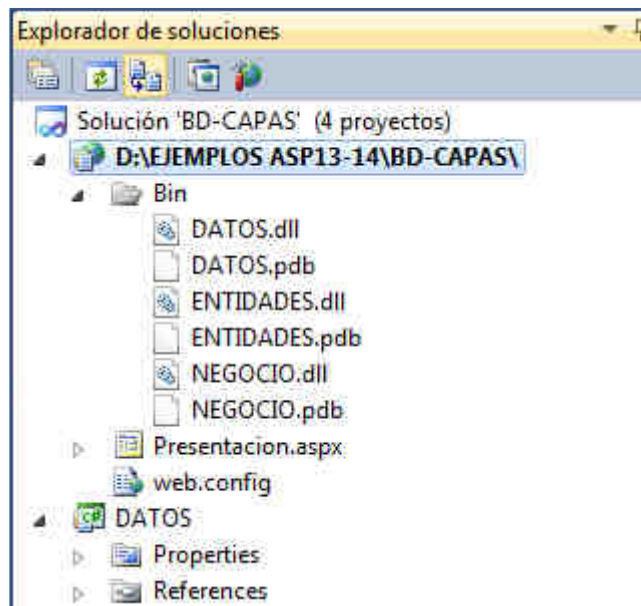
Hemos mostrado como agregar ENTIDADES en el proyecto de DATOS. Nos quedaría agregar este mismo proyecto de ENTIDADES en el proyecto de NEGOCIO y de PRESENTACIÓN.

Ahora agregaremos la **referencia de Datos** en el **proyecto de NEGOCIO**.

Y quedará de la siguiente forma:



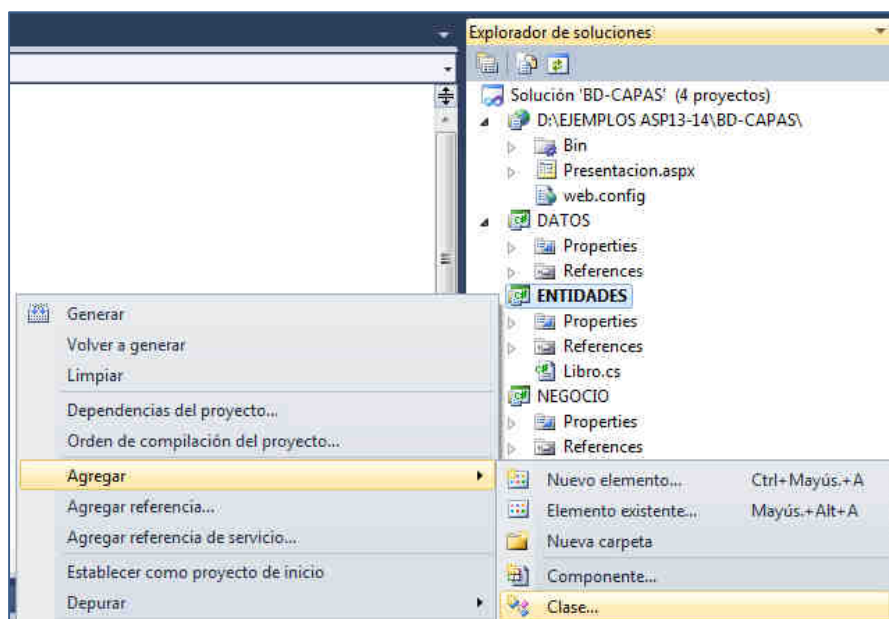
Por último agregaremos la referencia de **Negocio** al **proyecto de presentación**.




Ahora vamos a empezar a construir las clases en el proyecto de Entidades.

3

.- En nuestro ejemplo vamos a crear la **clase Libro** y **Editor**. Estas clases las crearemos como **públicas** para que puedan ser accedidas desde los otros proyectos. Crearemos sus propiedades privadas, las encapsularemos y crearemos el constructor.



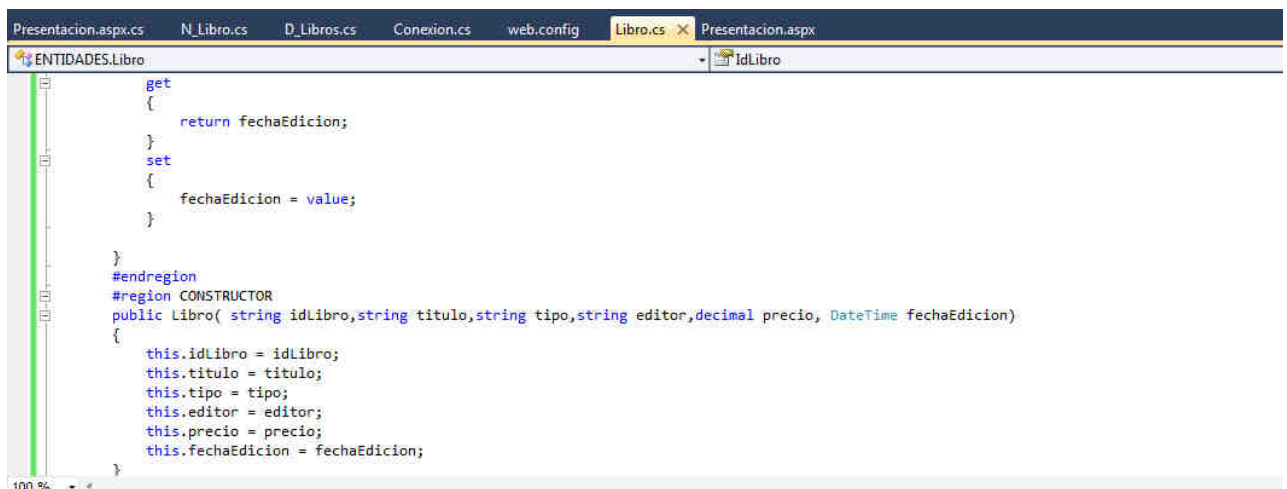
Ejemplo de la Clase Libro:



```
namespace ENTIDADES
{
    public class Libro
    {
        #region ATRIBUTOS
        string idLibro;
        string titulo;
        string tipo;
        string editor;
        decimal precio;
        DateTime fechaEdicion;
        #endregion
        #region ENCAPSULADO
        public string IdLibro {
            get
            {
                return idLibro;
            }
        }
    }
}
```

....

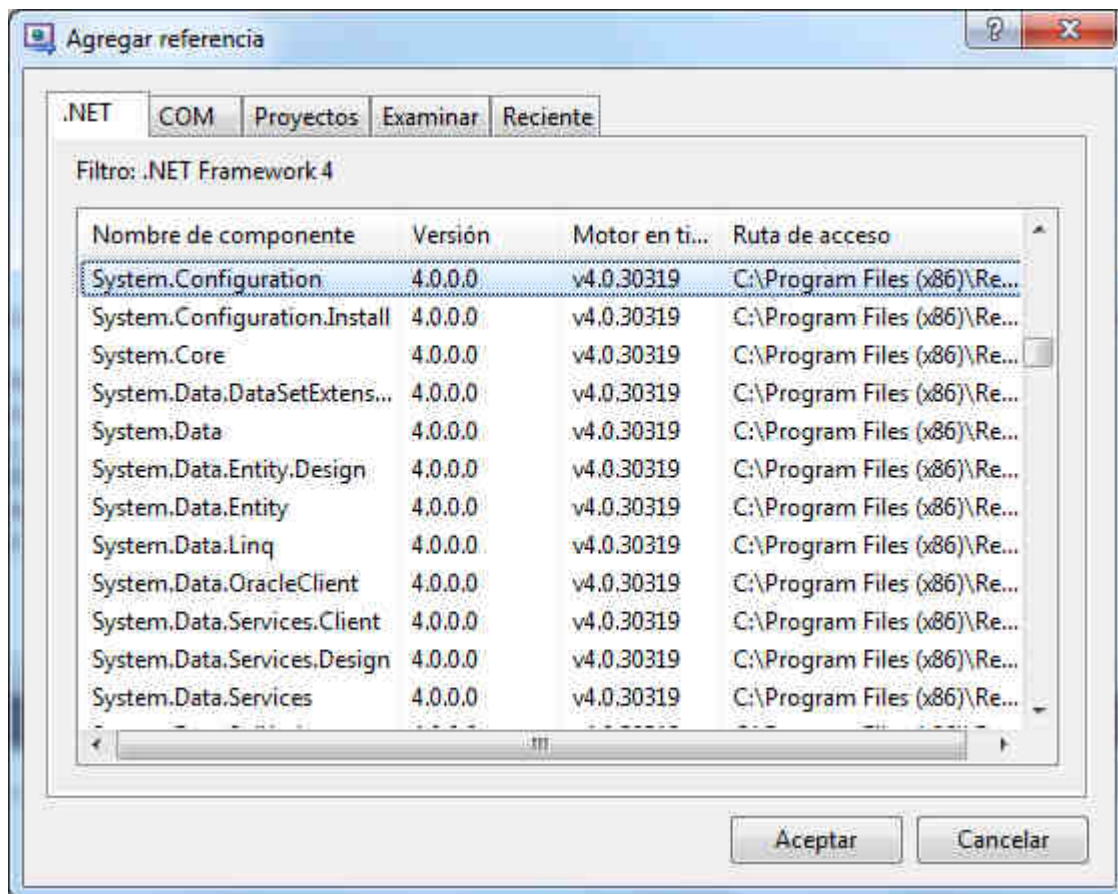
....



```
        set
        {
            fechaEdicion = value;
        }
    }
    #endregion
    #region CONSTRUCTOR
    public Libro( string idLibro, string titulo, string tipo, string editor, decimal precio, DateTime fechaEdicion)
    {
        this.idLibro = idLibro;
        this.titulo = titulo;
        this.tipo = tipo;
        this.editor = editor;
        this.precio = precio;
        this.fechaEdicion = fechaEdicion;
    }
}
```

4

.- Agregar en el Proyecto de Datos una Referencia al **System.Configuration**, para que podamos utilizar la conexión que definamos en el fichero de Configuración de nuestra aplicación: el fichero **Web.Config**.



En el fichero **Web.Config** debe aparecer añadida la cadena de conexión que queremos utilizar dentro de la etiqueta **<connectionStrings>**

```

Presentacion.aspx.cs  N_Libro.cs  D_Libros.cs*  Conexion.cs  web.config  Libro.cs  Presentacion.aspx
<?xml version="1.0"?>
<!--
Para obtener más información sobre cómo configurar la aplicación de ASP.NET, visite
http://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <connectionStrings>
    <add name="PubsConnectionString" connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=Pubs;Integrated Security=True" providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.0"/>
  </system.web>
</configuration>

```

```

<connectionStrings>
  <add name="PubsConnectionString" connectionString="Data
    Source=.\SQLEXPRESS;Initial Catalog=Pubs;Integrated Security=True"
    providerName="System.Data.SqlClient"/>
</connectionStrings>

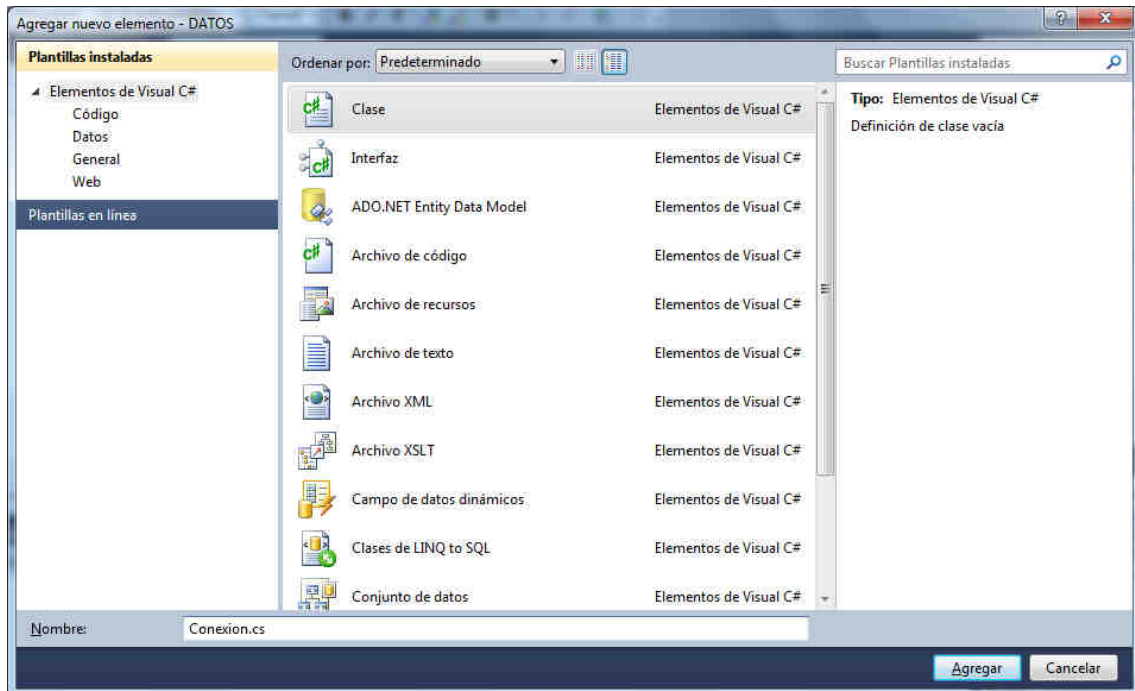
```

NOTA: Esta etiqueta **<connectionStrings>** **</connectionStrings>** debe ser introducida dentro de la etiqueta **<configuration>** **</configuration>**

5

.-Crearemos dentro del Proyecto de Datos la **Clase Conexión**.

Crearemos el **objeto conn** del tipo **SqlConnection** y crearemos **dos métodos** para **abrir y cerrar** la conexión.



CLASE CONEXIÓN

```
using System.Data.SqlClient;

namespace DATOS
{
    public class Conexion
    {
        public SqlConnection conn;

        public Conexion() {
            conn = new
                SqlConnection(ConfigurationManager.ConnectionStrings["PubsConnectionString"]
                    .ConnectionString);
        }

        public void abrirConexion()
        {
            try
            {
                if (conn.State == ConnectionState.Broken || conn.State ==
                    ConnectionState.Closed)
                    conn.Open();
            }
        }
    }
}
```

```

        catch (Exception ex)
        {
            throw new Exception ("Error al intentar abrir la conexión",ex);
        }
    }

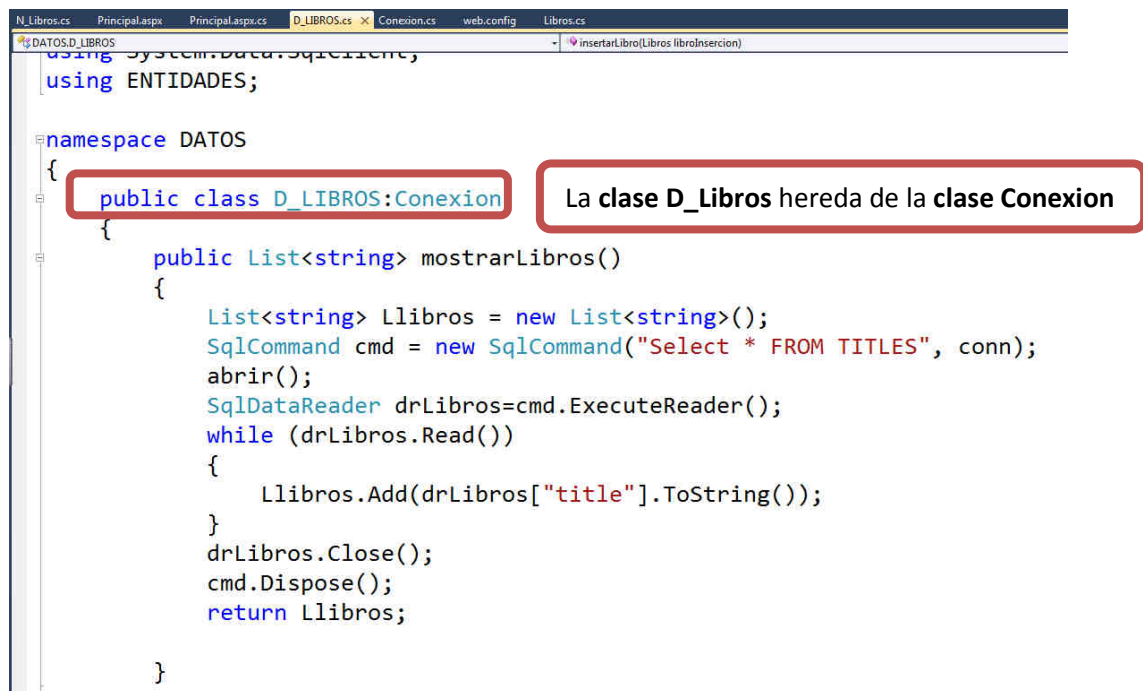
    public void cerrarConexion()
    {
        try
        {
            if (conn.State == ConnectionState.Open)
                conn.Close();
        }
        catch (Exception ex)
        {
            throw new Exception("Error al intentar cerrar la conexión", ex);
        }
    }
}
}
}

```

6

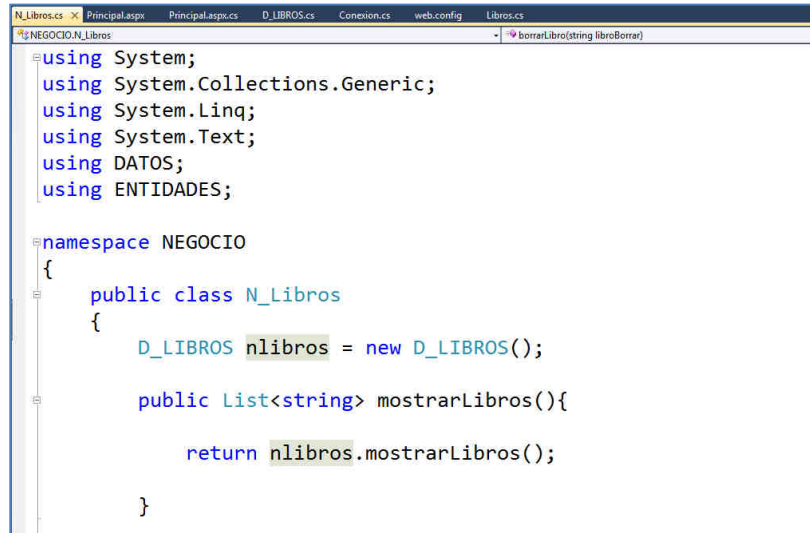
.-Crearemos dentro del Proyecto de Datos la **Clase D_Libros**. Esta es una clase que

heredará de la clase conexión, y en la que pueda utilizar objetos de tipo conexión y los métodos de abrir y cerrar conexiones. Además en ella realizaremos todos los métodos que necesitemos sobre los libros.



7

.-Crearemos dentro del Proyecto de Negocio la **Clase N_Libros**. Esta es una clase que mediante **un objeto de la clase D_Libros** accederá al **método** que deseemos de esta clase.



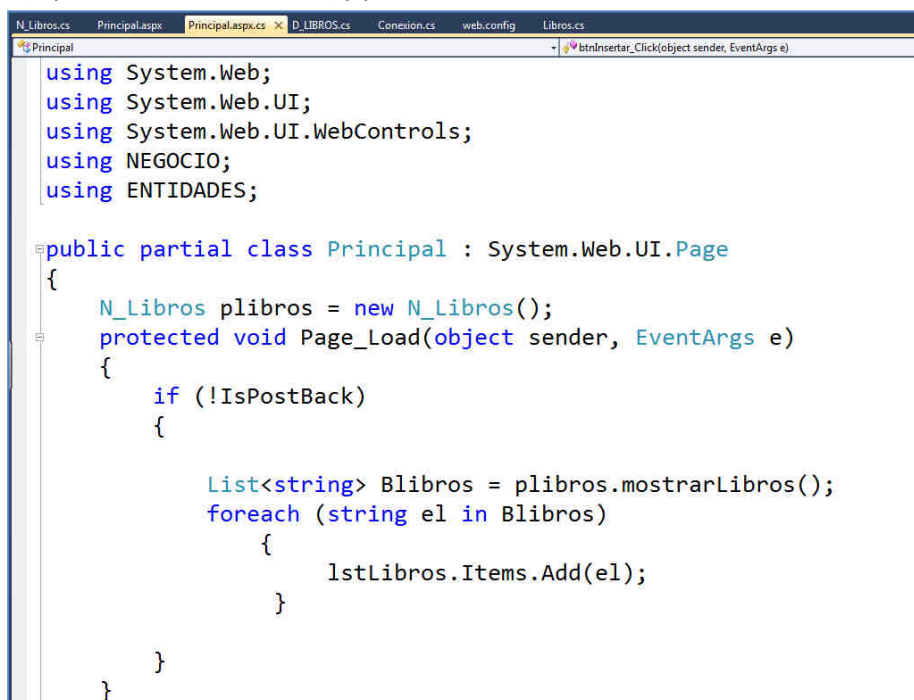
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DATOS;
using ENTIDADES;

namespace NEGOCIO
{
    public class N_Libros
    {
        D_LIBROS nlibros = new D_LIBROS();

        public List<string> mostrarLibros(){
            return nlibros.mostrarLibros();
        }
    }
}
```

8

.- Por último en el Web Form deseado, es decir en el proyecto correspondiente a la capa de presentación, se creará un objeto de tipo N_Libros (clase de la capa de Negocio) y a través del método mostrarLibros accederá a la capa de datos que será la que realmente accede a la BD. Este método devuelve una lista de tipo string, por lo que debemos crear una variable de este tipo (aquí denominada Blibros) y presentar desde ella el resultado.



```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using NEGOCIO;
using ENTIDADES;

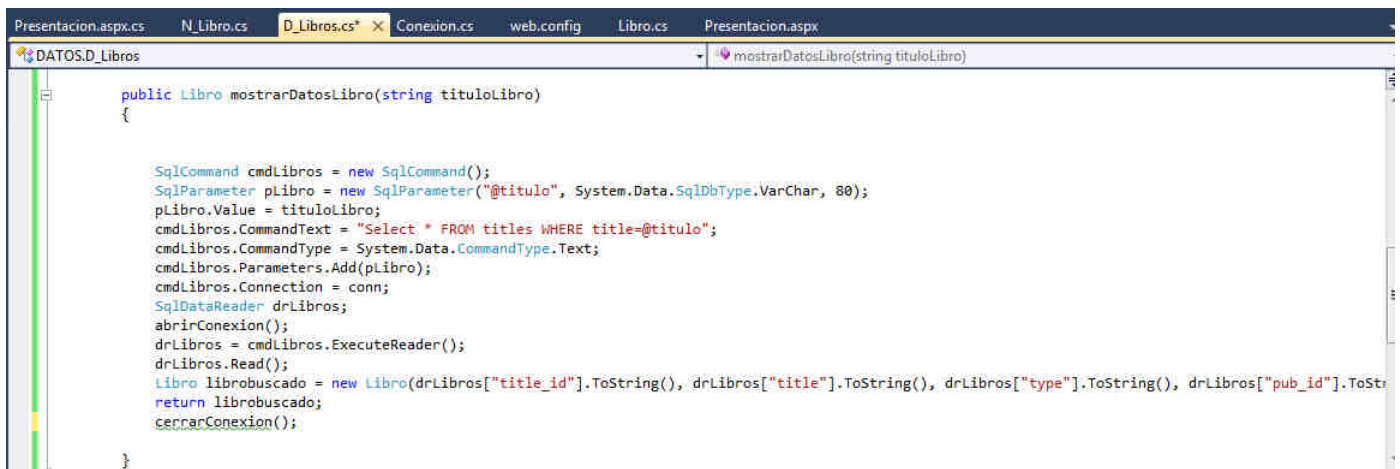
public partial class Principal : System.Web.UI.Page
{
    N_Libros plibros = new N_Libros();
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            List<string> Blibros = plibros.mostrarLibros();
            foreach (string el in Blibros)
            {
                lstLibros.Items.Add(el);
            }
        }
    }
}
```

9

.- Ahora nos toca seguir realizando métodos para conseguir realizar las tareas adecuadas.

Por ejemplo mostrar los datos de un libro seleccionado desde la lista desplegable de los títulos de los libros:

CAPA DE DATOS



```
Presentacion.aspx.cs  N_Libro.cs  D_Libros.cs*  Conexion.cs  web.config  Libro.cs  Presentacion.aspx
DATOS.D_Libros
mostrarDatosLibro(string tituloLibro)

public Libro mostrarDatosLibro(string tituloLibro)
{
    SqlCommand cmdLibros = new SqlCommand();
    SqlParameter pLibro = new SqlParameter("@titulo", System.Data.SqlDbType.VarChar, 80);
    pLibro.Value = tituloLibro;
    cmdLibros.CommandText = "Select * FROM titles WHERE title=@titulo";
    cmdLibros.CommandType = System.Data.CommandType.Text;
    cmdLibros.Parameters.Add(pLibro);
    cmdLibros.Connection = conn;
    SqlDataReader drLibros;
    abrirConexion();
    drLibros = cmdLibros.ExecuteReader();
    drLibros.Read();
    Libro librobuscado = new Libro(drLibros["title_id"].ToString(), drLibros["title"].ToString(), drLibros["type"].ToString(), drLibros["pub_id"].ToString());
    return librobuscado;
    cerrarConexion();
}
```

CAPA DE NEGOCIOS



```
Presentacion.aspx.cs  N_Libro.cs  D_Libros.cs*  Conexion.cs  web.config  Libro.cs  Presentacion.aspx
NEGOCIO.N_Libro
mostrarDatosLibro(string tituloLibro)

using ENTIDADES;

namespace NEGOCIO
{
    public class N_Libro
    {
        D_Libros objNLibro=new D_Libros();

        public List<string> mostrarLibros() {
            return objNLibro.mostrarLibros();
        }

        public Libro mostrarDatosLibro(string tituloLibro)
        {
            return objNLibro.mostrarDatosLibro(tituloLibro);
        }
    }
}
```

CAPA DE PRESENTACIÓN

```
Presentacion.aspx.cs* X N_Libro.cs D_Libros.cs* Conexion.cs web.config Libro.cs Presentacion.aspx
Presentacion
ddllibros.Items.Add(drLibros[1].ToString());
}
drLibros.Close();
}
protected void ddllibros_SelectedIndexChanged(object sender, EventArgs e)
{
    Libro libroBuscado;
    libroBuscado = objLibro.mostrarDatosLibro(ddllibros.SelectedItem.ToString());
    lblIDLibro.Text = libroBuscado.IdLibro.ToString();
    lblTitulo.Text = libroBuscado.Titulo;
    lbleditor.Text = libroBuscado.Editor;
    lblprecio.Text = libroBuscado.Precio.ToString();
    lbltipo.Text = libroBuscado.Tipo;
    lblfechaEdicion.Text = libroBuscado.FechaEdicion.ToLongDateString();
}
DateTime Libro.FechaEdicion
```

Y así seguiríamos haciendo métodos y clases para terminar nuestro ejercicio.