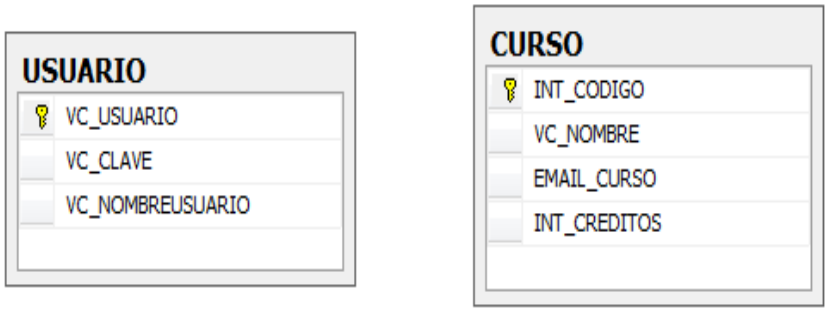
Guía de ASP.NET MVC

"El objetivo de esta guía es aprender a combinar la arquitectura de NCapas con ASP.NET MVC"

Hay ocasiones en que el modelo ADO.NET Entity Data Model no es soportado por todos los manejadores de base de datos, por ejemplo Oracle. El esquema de trabajo que presentaré es genérico y bastante usado en la práctica laboral.

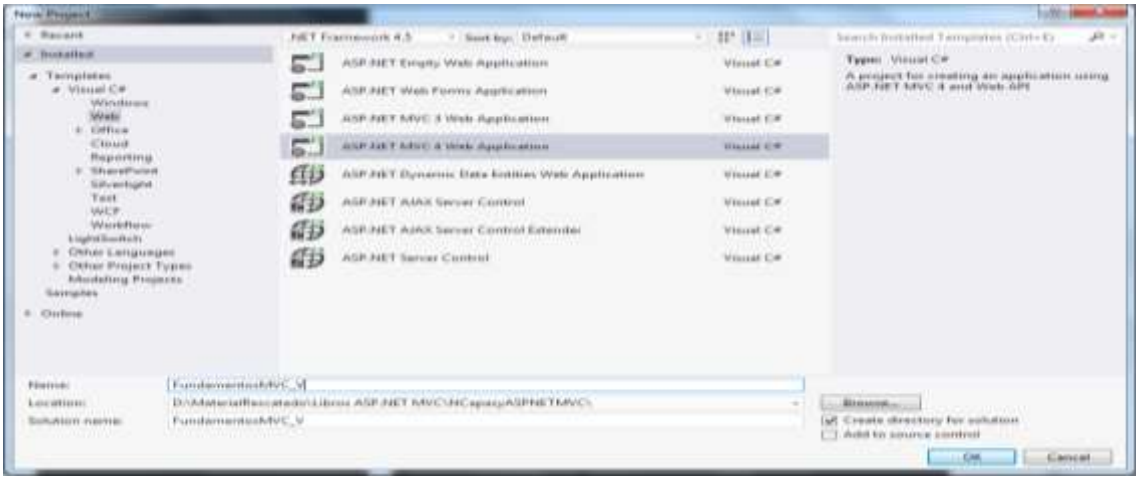
I.- Tablas a usar

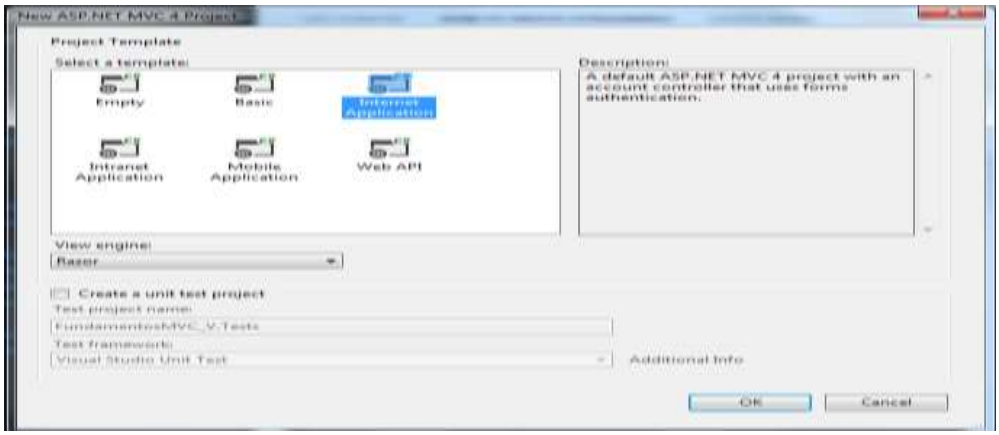
Para el ejemplo emplearemos la tabla Curso y la tabla Usuario creada para la "Guía 4 ASP.NET MVC":



II.- Creación del proyecto ASP.NET MVC

Creamos un proyecto nuevo llamado FundamentosMVC_V

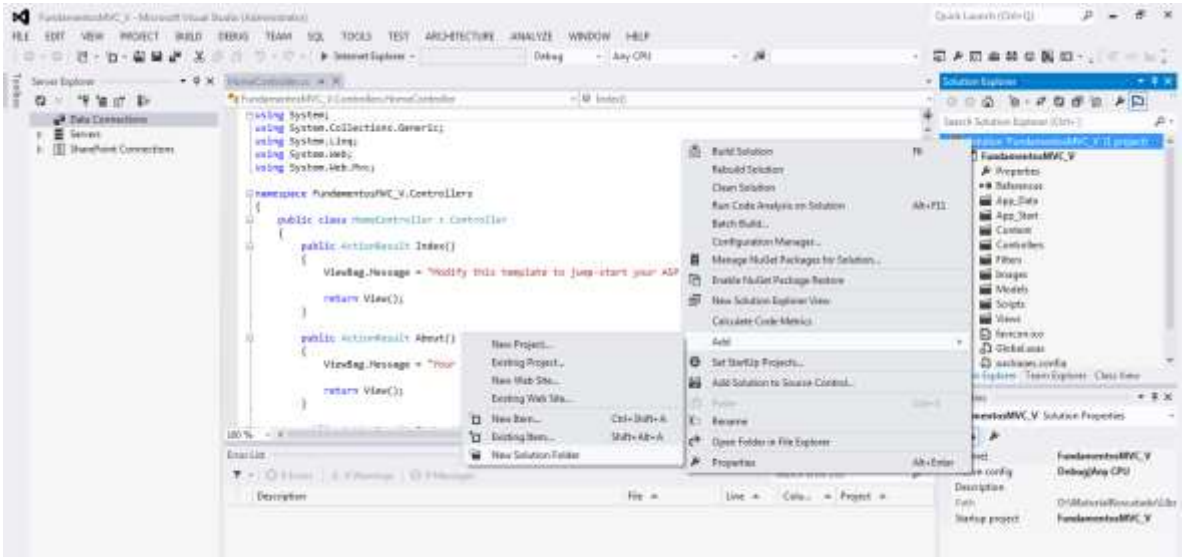




III.- Estructura de la aplicación

Una vez creado el proyecto base procedemos a armar la estructura de la aplicación, la cual será básicamente carpetas en donde irán los proyectos.

Clic derecho al archivo de solución→Add→New Solution Folder (haremos esta operación varias veces)



Crearemos las siguientes carpetas con la siguiente jerarquía:

2. Capas

    2.1 Presentación

        Web

ASP.NET MVC

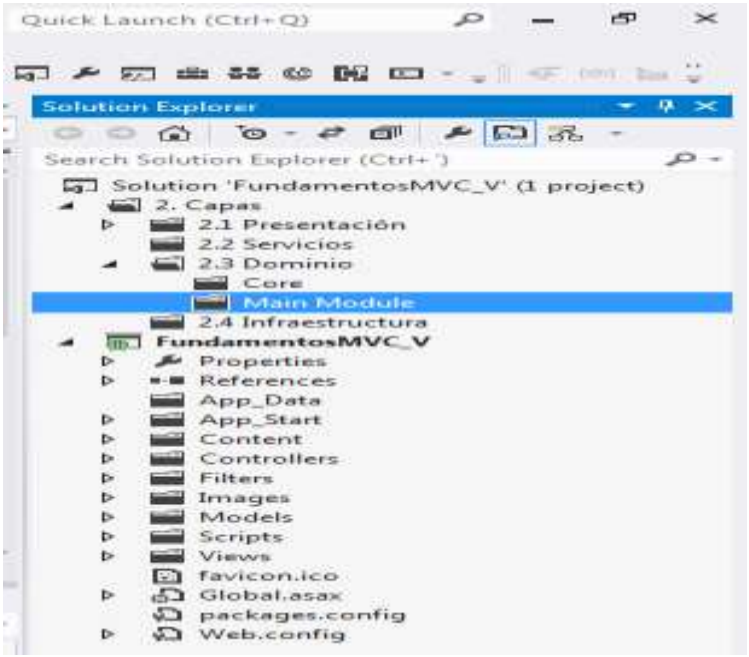
2.2 Servicios

2.3 Dominio

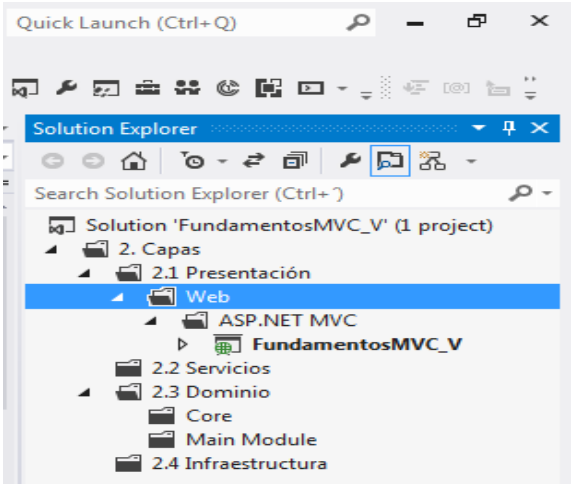
Core

Main Module

2.4 Infraestructura

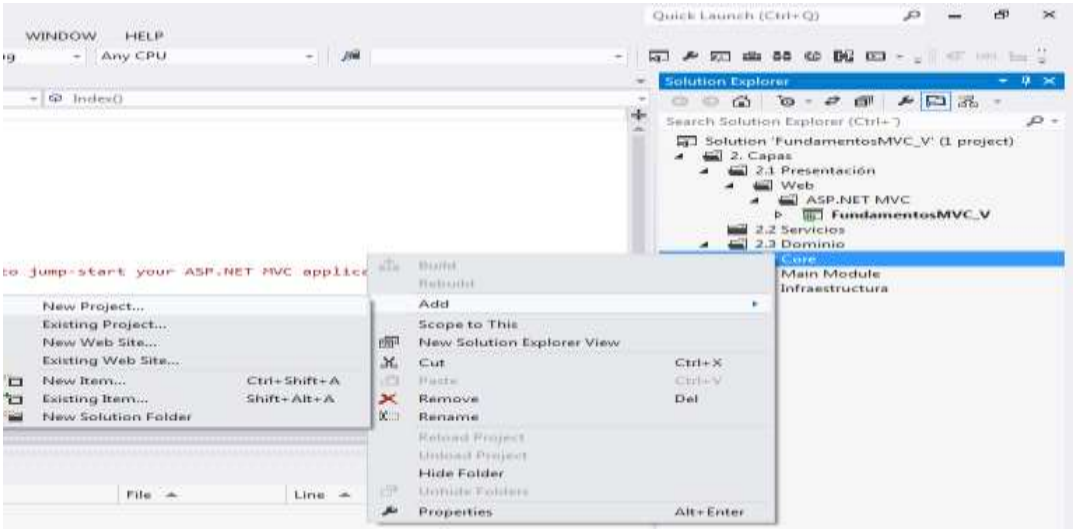


Movemos el proyecto FundamentosMVC_V a la carpeta ASP.NET MVC



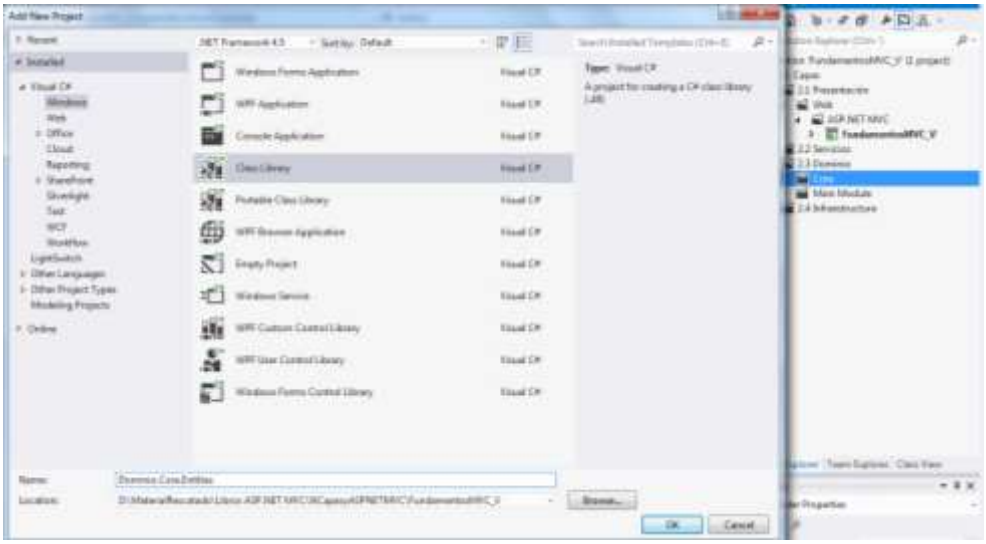
Ahora agregaremos los proyectos de Biblioteca de clases.

Clic derecho a la carpeta Core→Add→New Project



El primer proyecto se llamará Dominio.Core.Entities.

Si lo relacionamos con la estructura tradicional de N Capas vendría a ser el proyecto BusinessEntities.



Clic derecho a la carpeta Main Module→Add→New Project

Nombramos al proyecto de Biblioteca de clases Dominio.MainModule

Clic derecho a la carpeta 2.4 Infraestructura→Add→New Project

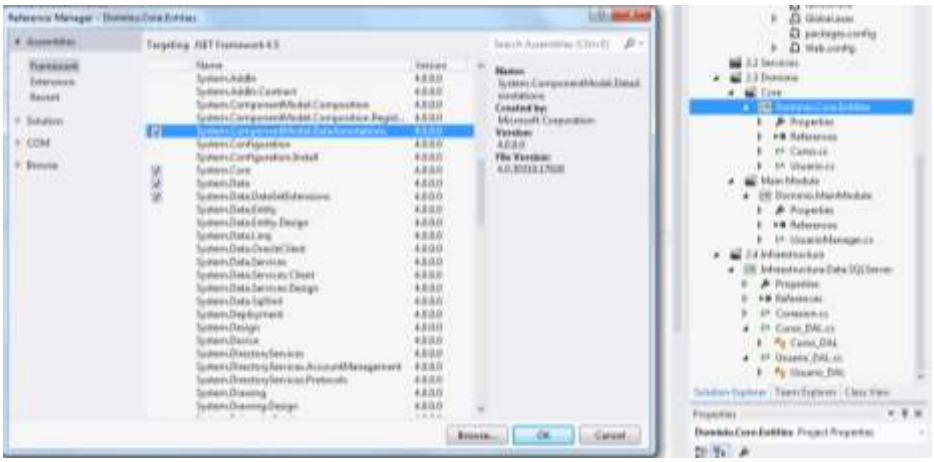
Nombramos al proyecto de Biblioteca de clases Infraestructura.Data.SQLServer.

Si lo relacionamos con la estructura tradicional de N Capas vendría a ser el proyecto DataAccessLayer.

IV.- Referencias a realizar

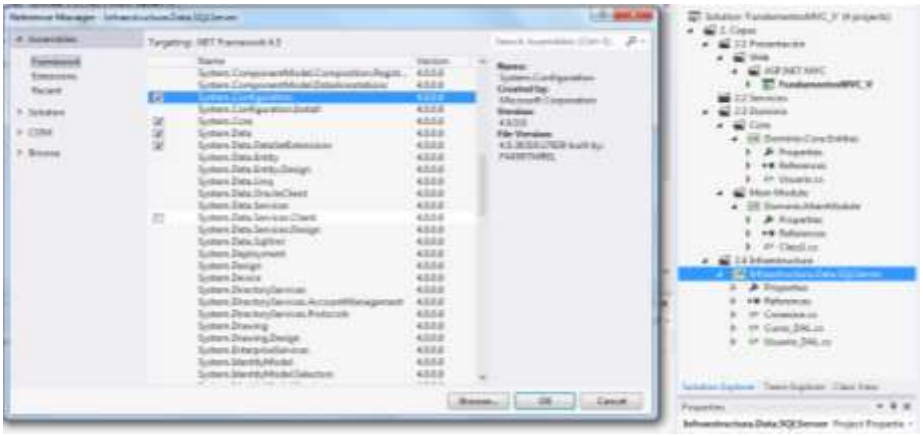
i) En el proyecto Dominio.Core.Entities

Referencia a System.ComponentModel.DataAnnotations

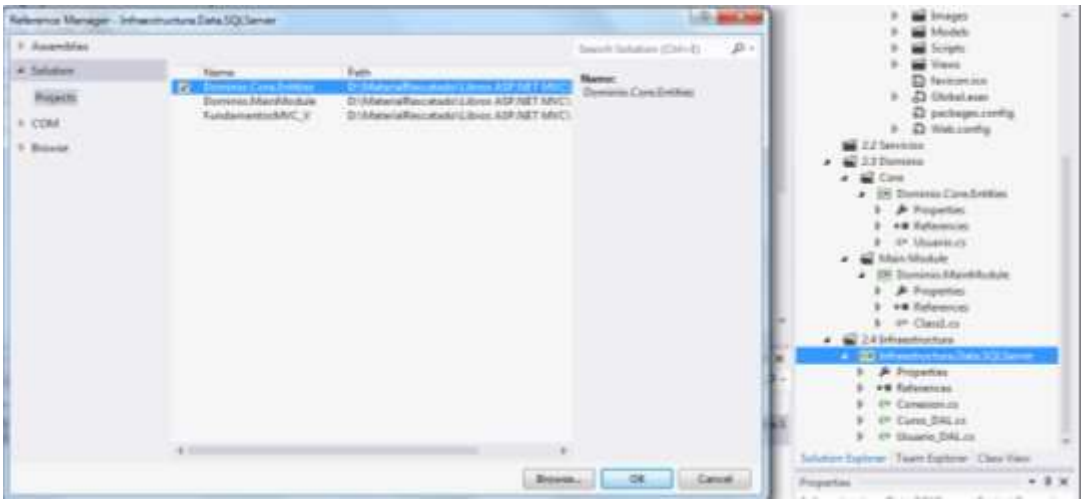


ii) En el proyecto Infraestructura.Data.SQLServer

Referencia a System.Configuration



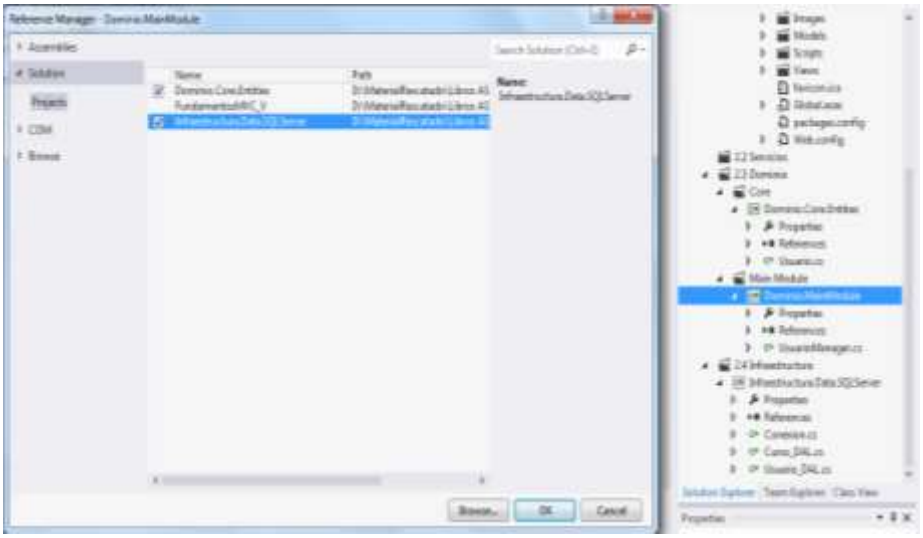
Referencia a Dominio.Core.Entities



iii) En el proyecto Dominio.Main.Module

Referencia a Dominio.Core.Entities

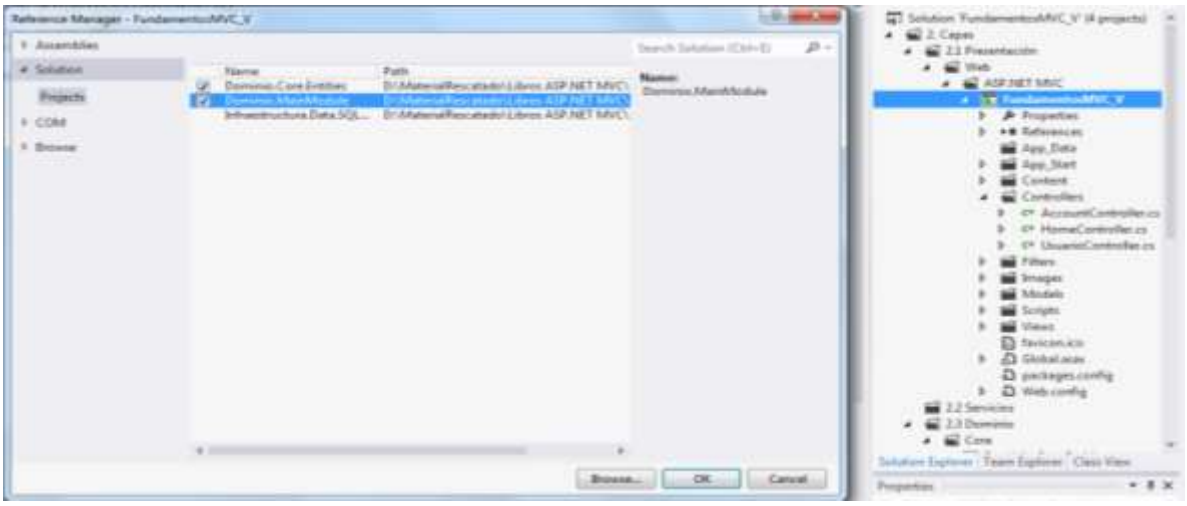
Referencia a Infraestructura.Data.SQLServer



iv) En el proyecto FundamentosMVC_V

Referencia a Dominio.Core.Entities

Referencia a Dominio.Main.Module



V.- Proyecto Dominio.Core.Entities

En el proyecto Dominio.Core.Entities agregamos las clases Usuario.cs y Curso.cs

i) **Clase Usuario.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Dominio.Core.Entities
{
    public class Usuario
    {
        [DisplayName("Usuario")]
        [Required(ErrorMessage = "Usuario requerido")]
        public string str_Usuario { get; set; }

        [DisplayName("Contraseña")]
        [Required(ErrorMessage = "Contraseña requerida")]
        public string str_Contrasena { get; set; }

        [DisplayName("Nombre")]
        public string str_Nombre { get; set; }

    }
}
```

Como ya vimos en la guía pasada, la ventaja de este esquema es que al crear la clase podemos poner los atributos de validación de la misma.

ii) Clase Curso.cs

Asimismo en esta clase ponemos atributos de validación.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Dominio.Core.Entities
{
    public class Curso
    {

        [DisplayName("Código del curso")]
        public int INT_CODIGO { get; set; }

        [DisplayName("Nombre del curso")]
        [Required(ErrorMessage = "Nombre de curso requerido")]
        [StringLength(30, MinimumLength = 3, ErrorMessage = "No más de 30
carácteres")]
        public string VC_NOMBRE { get; set; }

        [DisplayName("Email del curso")]
        [Required(ErrorMessage = "Email del curso requerido")]
        [StringLength(50, ErrorMessage = "No más de 50 carácteres")]
        [EmailAddress(ErrorMessage = "Email inválido")]
        public string EMAIL_CURSO { get; set; }

        [DisplayName("Créditos del curso")]
        [Required(ErrorMessage = "Número de créditos requerido")]
        [Range(1, 6)]
        public int INT_CREDITOS { get; set; }

    }
}
```

VI.- Proyecto Infraestructura.Data.SQlServer

En el proyecto Infraestructura.Data.SQLServer agregamos las clases Conexion.cs, CURSO_DAL.cs y Usuario_DAL.cs

i) Clase Conexion.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Data;
```

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

namespace Infraestructura.Data.SQLServer
{
   public class Conexion
    {
        SqlConnection conexion;
        public SqlConnection Conectar()
        {
            conexion = new
SqlConnection(ConfigurationManager.ConnectionStrings["Conexion"].ConnectionString);

            return conexion;
        }
    }
}
```

Se observa que hemos definido un método de conexión para que pueda ser reusado por otras clases de este proyecto.

Es importante destacar que la cadena de conexión está fijada en el archivo Web.Config del proyecto FUNDAMENTOSMVC_V. Allí se incluyó la siguiente entrada:

```
<connectionStrings>
    <add name="Conexion" connectionString="server=USUARIO1-
PC\JAMES;database=VALIDACIONES;uid=sa;pwd=sql"
providerName="System.Data.SqlClient"/>

  </connectionStrings>
```

ii) Clase Usuario_DAL.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Dominio.Core.Entities;
using System.Data;
using System.Data.SqlClient;

namespace Infraestructura.Data.SQLServer
{
    public class Usuario_DAL
    {


        SqlConnection conexion;
        SqlDataAdapter comando;
        SqlDataReader dr;
        SqlCommand cmd;
```

```
String errores;
Conexion cn = new Conexion();

//Método que valida las credenciales de un usuario y retorna una trama
public IEnumerable<Usuario> LoginUsuario(string strusuario,
                                         string strpassword)
{
    //Creamos una lista vacía
    List<Usuario> lista = new List<Usuario>();

    try
    {
        conexion = cn.Conectar();

        cmd = new SqlCommand("PR_LOGIN", conexion);
        cmd.CommandType = CommandType.StoredProcedure;


        cmd.Parameters.Add(new SqlParameter("p_Usuario",SqlDbType.VarChar,
        10));

        cmd.Parameters["p_Usuario"].Direction = ParameterDirection.Input;

        cmd.Parameters["p_Usuario"].Value = strusuario;

        cmd.Parameters.Add(new SqlParameter("p_Contrasena",
        SqlDbType.VarChar, 10));

        cmd.Parameters["p_Contrasena"].Direction = ParameterDirection.Input;

        cmd.Parameters["p_Contrasena"].Value = strpassword;



        dr = null;

        conexion.Open();

        dr = cmd.ExecuteReader();

        while (dr.Read())
        {
            //Instanciamos la entidad
            Usuario objeto = new Usuario();

            //Llenamos las propiedades de la entidad
            objeto.str_Usuario = Convert.ToString(dr["VC_USUARIO"]);
            objeto.str_Contrasena = Convert.ToString(dr["VC_CLAVE"]);
            objeto.str_Nombre = Convert.ToString(dr["VC_NOMBREUSUARIO"]);


            //Agregamos la entidad a la lista de entidades
            lista.Add(objeto);

        }

        dr.Close();
```

```
            }
            catch (Exception ex)
            {
                errores = ex.Message;

            }
            finally
            {

                if (conexion.State == ConnectionState.Open)
                {
                    conexion.Close();
                }

                conexion.Dispose();
                cmd.Dispose();
                //dr.Dispose();
            }
            //Como resultado mostramos la lista de entidades
            return lista;

        }
    }
}
```

```
CREATE PROCEDURE PR_LOGIN
@p_Usuario VARCHAR(10),
@p_Contrasena VARCHAR(10)
AS
SELECT * FROM USUARIO
WHERE VC_USUARIO=@p_Usuario
AND VC_CLAVE=@p_Contrasena
GO
```

iii) Clase Curso_DAL.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Dominio.Core.Entities;
using System.Data;
using System.Data.SqlClient;

namespace Infraestructura.Data.SQLServer
{
    public class Curso_DAL
    {
```

```
SqlConnection conexion;
SqlDataAdapter comando;
SqlDataReader dr;
SqlCommand cmd;
String errores;
Conexion cn = new Conexion();

//Método que lista los Cursos

public IEnumerable<Curso> ListarCursos()
{
    //Creamos una lista vacía
    List<Curso> lista = new List<Curso>();

    try
    {
        conexion = cn.Conectar();

        cmd = new SqlCommand("PR_LISTAR_CURSOS", conexion);
        cmd.CommandType = CommandType.StoredProcedure;



        dr = null;

        conexion.Open();

        dr = cmd.ExecuteReader();

        while (dr.Read())
        {
            //Instanciamos la entidad
            Curso objeto = new Curso();

            //Llenamos las propiedades de la entidad
            objeto.INT_CODIGO = Convert.ToInt32(dr["INT_CODIGO"]);
            objeto.VC_NOMBRE = Convert.ToString(dr["VC_NOMBRE"]);
            objeto.EMAIL_CURSO = Convert.ToString(dr["EMAIL_CURSO"]);
            objeto.INT_CREDITOS = Convert.ToInt32(dr["INT_CREDITOS"]);


            //Agregamos la entidad a la lista de entidades
            lista.Add(objeto);

        }

        dr.Close();


    }
    catch (Exception ex)
    {
        errores = ex.Message;

    }
    finally
    {
```

```
            if (conexion.State == ConnectionState.Open)
            {
                conexion.Close();
            }

            conexion.Dispose();
            cmd.Dispose();
            //dr.Dispose();
        }
        //Como resultado mostramos la lista de entidades
        return lista;

    }

    //Método que registra un curso
    public Boolean RegistrarCurso(Curso objeto)

    {
        try
        {
            conexion = cn.Conectar();

            cmd = new SqlCommand("PR_REGISTRA_CURSO", conexion);

            cmd.CommandType = CommandType.StoredProcedure;

            //Definimos los parámetros de entrada

            cmd.Parameters.Add(new SqlParameter("@VC_NOMBRE", SqlDbType.VarChar,
            100));
            cmd.Parameters["@VC_NOMBRE"].Direction = ParameterDirection.Input;
            cmd.Parameters["@VC_NOMBRE"].Value = objeto.VC_NOMBRE;

            cmd.Parameters.Add(new SqlParameter("@VC_EMAIL_CURSO",
            SqlDbType.VarChar, 50));
            cmd.Parameters["@VC_EMAIL_CURSO"].Direction =
            ParameterDirection.Input;
            cmd.Parameters["@VC_EMAIL_CURSO"].Value = objeto.EMAIL_CURSO;

            cmd.Parameters.Add(new SqlParameter("@INT_CREDITOS",
            SqlDbType.Int));
            cmd.Parameters["@INT_CREDITOS"].Direction =
            ParameterDirection.Input;
            cmd.Parameters["@INT_CREDITOS"].Value = objeto.INT_CREDITOS;


            conexion.Open();
            cmd.ExecuteNonQuery();
            return true;
        }
        catch (Exception ex)
        {
            errores = ex.Message;
            return false;
```

```
            }
            finally
            {
                if (conexion.State == ConnectionState.Open)
                {
                    conexion.Close();
                }

                conexion = null;
                cmd = null;
                cn = null;
            }
        }

    }
}
```

```
CREATE PROCEDURE PR_LISTAR_CURSOS
AS
SELECT * FROM CURSO
ORDER BY VC_NOMBRE
GO


CREATE PROCEDURE PR_REGISTRA_CURSO
@VC_NOMBRE VARCHAR(100),
@VC_EMAIL_CURSO VARCHAR(50),
@INT_CREDITOS INT
AS
IF NOT EXISTS(SELECT * FROM CURSO
WHERE VC_NOMBRE=@VC_NOMBRE)
BEGIN
       INSERT INTO CURSO(VC_NOMBRE,EMAIL_CURSO,INT_CREDITOS)
       VALUES(@VC_NOMBRE,@VC_EMAIL_CURSO,@INT_CREDITOS)


END
GO
```

VII.- Proyecto Dominio.Main.Module

i) Clase UsuarioManager.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Dominio.Core.Entities;
using Infraestructura.Data.SQLServer;

namespace Dominio.MainModule
```

```
{
   public class UsuarioManager
    {
        Usuario_DAL dal = new Usuario_DAL();

        public IEnumerable<Usuario> LoginUsuario(string strusuario, string
strpassword)
        {
            return dal.LoginUsuario(strusuario, strpassword);
        }
    }
}
```

ii) Clase CursoManager.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Dominio.Core.Entities;
using Infraestructura.Data.SQLServer;

namespace Dominio.MainModule
{
    public class CursoManager
    {

        Curso_DAL dal = new Curso_DAL();

        public IEnumerable<Curso> ListarCursos()
        {
            return dal.ListarCursos();
        }

        public Boolean RegistrarCurso(Curso objeto)
        {
            return dal.RegistrarCurso(objeto);
        }

    }
}
```

En ambas clases hemos observado que desde el proyecto Dominio.MainModule hemos llamado a los métodos definidos en las respectivas clases del proyecto Infraestructura.Data.SQLServer

VIII.- Carpeta Controllers del proyecto FundamentosMVC_V:

i) Agregamos el controlador USUARIOController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.Mvc;


using Dominio.Core.Entities;
using Dominio.MainModule;

namespace FundamentosMVC_V.Controllers
{
    public class USUARIOController : Controller
    {
        //
        // GET: /Usuario/

        public ActionResult Index()
        {
            return View();
        }

public ActionResult Validar()
        {


            if (Request.Cookies["Usuario"] != null)
            {
                Response.Cookies["Usuario"].Expires = DateTime.Now.AddDays(-1);

            }


            if (Request.Cookies["NombreUsuario"] != null)
            {
               Response.Cookies["NombreUsuario"].Expires = DateTime.Now.AddDays(-1);

            }
            return View();
        }

        [HttpPost]
        public ActionResult Validar(string strusuario,
                                    string strpassword)
        {

            IEnumerable<Usuario> objeto = null;

            UsuarioManager manager = new UsuarioManager();

            objeto = manager.LoginUsuario(strusuario, strpassword);

            //Si no se encontró elementos
            if (objeto.Count() == 0)
            {
                return View();
            }
            else
            {
```

```
                //Almacenamos la data en cookies
                Response.Cookies["Usuario"].Value =
objeto.ElementAt(0).str_Usuario.ToString();
                Response.Cookies["NombreUsuario"].Value =
objeto.ElementAt(0).str_Nombre.ToString();

                Response.Cookies["Usuario"].Expires = DateTime.Now.AddDays(1);
                Response.Cookies["NombreUsuario"].Expires = DateTime.Now.AddDays(1);

                //Redirecciona al listado de cursos
                return RedirectToAction("Index", "CURSO");
            }



        }



    }
}
```

Notemos que el controlador ya no interatúa directamente con la carpeta Models, sino con Dominio.Core.Entities (que vendría a representar el modelo) y Dominio.MainModule (que vendría a representas la lógica de negocios). A través de cookies ("galletas") guardamos de manera opcional el código y nombre del usuario. Le indicamos que estas cookies expirarán dentro un día. Es una manera más limpia de guardar datos a diferencia de las variables Sessions.

Como el método validar puede ser invocado al cargar la aplicación o al cerrar sesión, éste destruye las cookies existentes como primera tarea.

ii) Agregamos el controlador CURSOController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

using Dominio.Core.Entities;
using Dominio.MainModule;

namespace FundamentosMVC_V.Controllers
{
    public class CURSOController : Controller
    {
        //
        // GET: /CURSO/

        public ActionResult Index()
        {

            IEnumerable<Curso> objeto = null;

            CursoManager manager = new CursoManager();
```

```
            objeto = manager.ListarCursos();

                return View(objeto);

        }

        public ActionResult CrearCurso()
        {
            return View();
        }

        [HttpPost]
        public ActionResult CrearCurso(Curso objeto)
        {
           CursoManager manager = new CursoManager();

            if (ModelState.IsValid)
            {
                manager.RegistrarCurso(objeto);
                return RedirectToAction("Index");
            }
            else
            {
                return View(objeto);
            }



        }

        public ActionResult CerrarSesion()
        {
            return RedirectToAction("Validar", "USUARIO");
        }


    }
}
```

El método CerrarSesion redirecciona al método Validar del controlador USARIOController para que destruya las cookies.

IX.- Modificación de la plantilla Layout.cshtml

Como es sabido _Layout.cshtml ubicado en la carpeta Views/Shared del proyecto FundamentosMVC_V sirve como plantilla para todas las vistas que la tomen. Si deseamos evitar el menú por defecto que viene con Visual Studio así como el fondo podemos modificar este archivo. Previamente hemos agregado las imágenes CIBERTEC_UPC.jpg y StarWars.jpg a la carpeta Images del proyecto FundamentosMVC_V.

Reemplazamos este bloque

```
<div class="float-left" align="center">
<p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>
```

```
</div>
```

Por este otro

```
<div class="float-left" align="center">
@*<p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>*@
                    <table>
                   <tr>
                   <td>
                   <img src="../Images/StarWars.jpg" style="width: 300px; height:
300px;" />
                   </td>
                    <td>
                     <label style="font-size: 40px; font-family:Verdana;font-
weight:bold;">
                    Curso Desarrollo para entorno web
                    </label>
                    </td>
                   </tr>
                   </table>
                </div>
```

Reemplazamos este bloque

```
<div class="float-right">
                    <section id="login">
                        @Html.Partial("_LoginPartial")
                    </section>
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("About", "About", "Home")</li>
                            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                        </ul>
                    </nav>
                </div>
```

Por este otro

```
<div class="float-right">
                    @*<section id="login">
                        @Html.Partial("_LoginPartial")
                    </section>*@
                    @*<nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("About", "About", "Home")</li>
                            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                        </ul>
                    </nav>*@
                </div>
```

Finalmente, reemplazamos este bloque

```
<footer>
            <div class="content-wrapper" align="center">
               <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>

            </div>
        </footer>
```

Por este otro

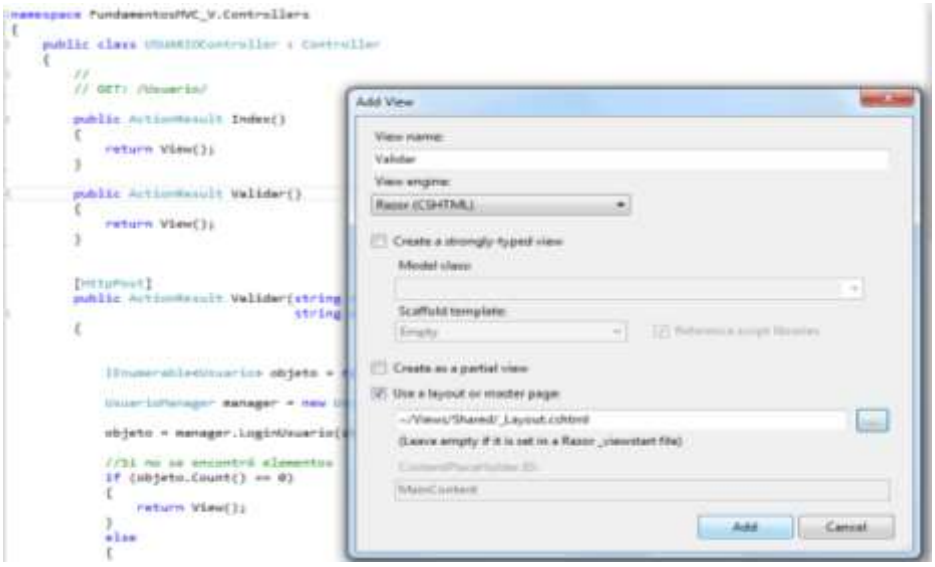
```
<footer>
            <div class="content-wrapper" align="center">
                @*<div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>*@
                 <table>
                   <tr>
                   <td>
                   <img src="../Images/CIBERTEC_UPC.jpg" style="width: 130px;
height: 130px;" />
                   </td>
                    <td>
                     <label style="font-size: 10px; font-family:Verdana;font-
weight:bold;">
                    CIBERTEC 2014. Todos los derechos reservados
                    </label>
                    </td>
                   </tr>
                   </table>
            </div>
        </footer>
```

X.- Generación de las vistas

Archivo Validar.cshtml (Formulario de logueo)

En el controlador USUARIOControler clic derecho al ActionResult Validar→Add View…

No olvidar seleccionar la opción Use a layout or master page




```
@model Dominio.Core.Entities.Usuario

@{
    ViewBag.Title = "Validar";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<div align="center">
<h1>Acceso al sistema</h1>
</div>


<div align="center">


@using (Html.BeginForm())
{

    <table border="2">
    <tr>
    <td>
    <b>@Html.DisplayNameFor(model => model.str_Usuario)</b>
    </td>
    <td style="text-align: left; font-size: 12px;">
    @Html.TextBox("strusuario")
    @Html.ValidationMessageFor(m => m.str_Usuario)
    </td>

    </tr>

    <tr>
    <td>
    <b>@Html.DisplayNameFor(model => model.str_Contrasena)</b>
    </td>
    <td style="text-align: left; font-size: 12px;">
    @Html.Password("strpassword")
    @Html.ValidationMessageFor(m => m.str_Contrasena)
```

```
    </td>

    </tr>

     <tr>
    <td colspan="2"><input type="submit" value="Ingresar" onclick="Validar();"
/></td>
    </tr>
    </table>

}
</div>

 @section Scripts {
     @Scripts.Render("~/bundles/jqueryval")


   <script type="text/javascript">


         function Validar() {

             var usuario = $("#strusuario").val();
             var password = $("#strpassword").val();


             if (usuario == '') {
                 alert('Indique el usuario');
                 $("#strusuario").focus();

                 return false;

             }

             if (password == '') {
                 alert('Indique el password');
                 $("#strpassword").focus();

                 return false;

             }

      return true;
      }

      </script>
}
```

Es importante destacar lo siguiente: Cuando se trabajaba directamente con ADO.NET Entity Data Model se habría referenciado al modelo así:

```
@model FundamentosMVC_V.Models.USUARIO
```

Ahora, en este nuevo esquema referenciamos al modelo de esta manera:

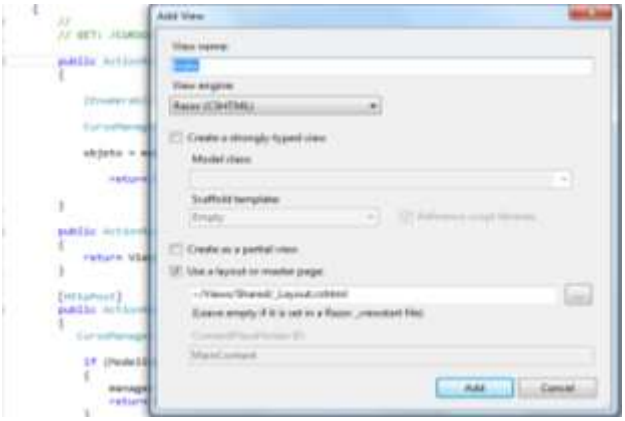
```
@model Dominio.Core.Entities.Usuario
```

Estamos usando DataAnnotations para que nos muestre las etiquetas de los campos. Para validar que los campos sean obligatorios usamos Javascript, aunque sabemos que también pudimos usar DataAnnotations.

Archivo Index.cshtml (Formulario de listado de cursos)

En el controlador CURSOController clic derecho al ActionResult Index→Add View…

No olvidar seleccionar la opción Use a layout or master page



```
@model IEnumerable<Dominio.Core.Entities.Curso>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<table>
<tr>

<td><b>Hola @Request.Cookies["NombreUsuario"].Value.ToString()</b></td>
<td>@Html.ActionLink("Cerrar sesión", "CerrarSesion")</td>
</tr>
</table>

<h2>Listado de cursos</h2>

<p>
    @Html.ActionLink("Crear nuevo curso", "CrearCurso")
</p>
<table>
    <tr>
        <th style="border: 2px solid black; text-align: center; font-size: 12px;
background-color: #0066FF;">
            @Html.DisplayNameFor(model => model.INT_CODIGO)
```

```
        </th>
        <th style="border: 2px solid black; text-align: center; font-size: 12px;
background-color: #0066FF;">
            @Html.DisplayNameFor(model => model.VC_NOMBRE)
        </th>
        <th style="border: 2px solid black; text-align: center; font-size: 12px;
background-color: #0066FF;">
            @Html.DisplayNameFor(model => model.EMAIL_CURSO)
        </th>
        <th style="border: 2px solid black; text-align: center; font-size: 12px;
background-color: #0066FF;">
            @Html.DisplayNameFor(model => model.INT_CREDITOS)
        </th>

    </tr>

@foreach (var item in Model) {
    <tr>
        <td style="border: 2px solid black; text-align: left; font-size: 12px;">
            @Html.DisplayFor(modelItem => item.INT_CODIGO)
        </td>
        <td style="border: 2px solid black; text-align: left; font-size: 12px;">
            @Html.DisplayFor(modelItem => item.VC_NOMBRE)
        </td>
        <td style="border: 2px solid black; text-align: left; font-size: 12px;">
            @Html.DisplayFor(modelItem => item.EMAIL_CURSO)
        </td>
        <td style="border: 2px solid black; text-align: left; font-size: 12px;">
            @Html.DisplayFor(modelItem => item.INT_CREDITOS)
        </td>

    </tr>
}

</table>
```

No olvidar que cuando una vista devuelve un flujo de datos hay que añadir el prefijo IEnumerable en la declaración del modelo:

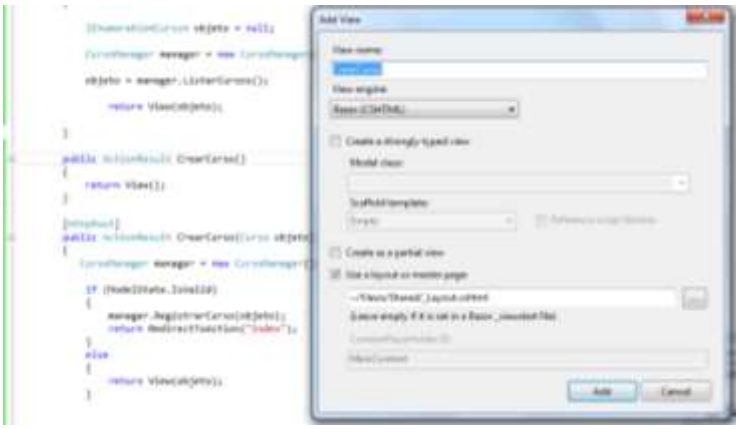
```
@model IEnumerable<Dominio.Core.Entities.Curso>
```

Se ha fijado estilos para las celdas de encabezado y de datos. En forma estricta he de decir que sería mejor tomarlo de una hoja de estilo.

Archivo CrearCurso.cshtml (Formulario de registro de cursos)

En el controlador CURSOController clic derecho al ActionResult CrearCurso→Add View…

No olvidar seleccionar la opción Use a layout or master page



```
@model Dominio.Core.Entities.Curso

@{
    ViewBag.Title = "CrearCurso";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<table>
<tr>

<td><b>Hola @Request.Cookies["NombreUsuario"].Value.ToString()</b></td>
<td>@Html.ActionLink("Cerrar sesión", "CerrarSesion")</td>
</tr>
</table>

<h2>Creación de un curso</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>CURSO</legend>


        <div class="editor-label">
            @Html.LabelFor(model => model.VC_NOMBRE)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.VC_NOMBRE)
            @Html.ValidationMessageFor(model => model.VC_NOMBRE)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.EMAIL_CURSO)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.EMAIL_CURSO)
            @Html.ValidationMessageFor(model => model.EMAIL_CURSO)
        </div>

        <div class="editor-label">
```

```
            @Html.LabelFor(model => model.INT_CREDITOS)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.INT_CREDITOS)
            @Html.ValidationMessageFor(model => model.INT_CREDITOS)
        </div>

        <p>
            <input type="submit" value="Procesar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Regresar al listado de cursos", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

XI.- Vista inicial de la aplicación

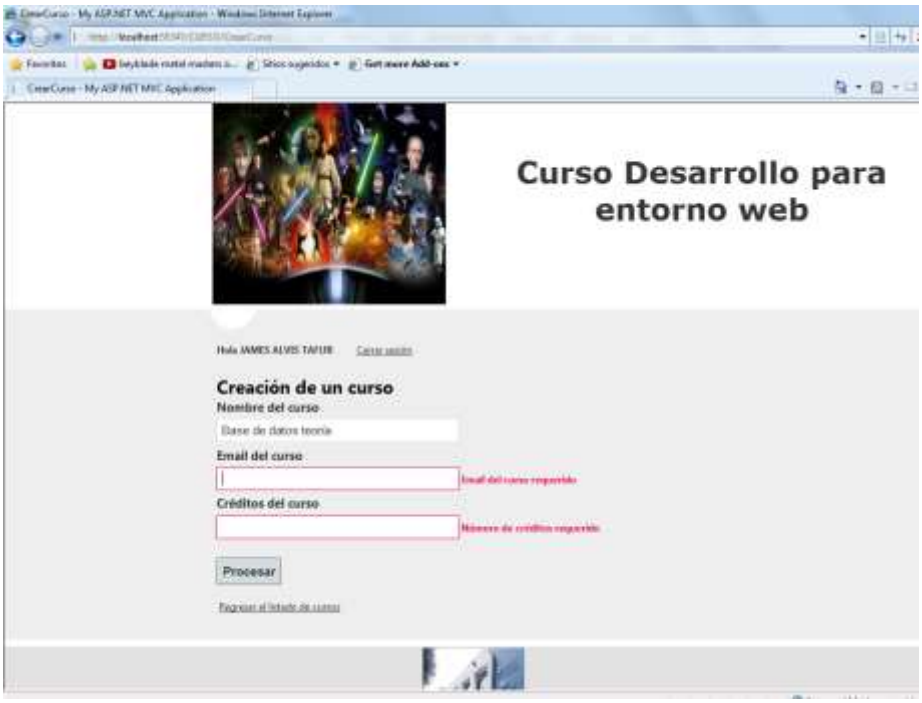
Modificamos el archivo RouteConfig.cs de la carpeta App_Start del proyecto FundamentosMVC_V, indicando que se ejecutará por defecto el ActionResult Validar del controlador USUARIOController

XII.- Ejecución de la aplicación

Como era de esperar se visualiza como pantalla inicial la vista de logueo

Ingresamos las credenciales del usuario y damos clic al botón Ingresar.

Observemos como a través de Cookies estamos mostrando el nombre del usuario que se ha conectado a la aplicación.



Como se puede observar el uso de DataAnnotations definidos en el proyecto Dominio.Core.Entities nos ayuda en las validaciones.

Eso es todo por ahora….**J.A.T**.