INSTITUTO TECNOLOGICO DE NUEVO LAREDO

CAMPUS NUEVO LAREDO

MANUAL TÉCNICO

DOCENTE: ING. LUIS CARLOS GARCIA GARCIA

CECILIA IVONNE HUERTA MELCHOR 17100238

PROGRAMACIÓN AVANZADA 1

7-B

Requerimientos

Elabore un pequeño sistema que maneje catálogos y venta de algun artículo. Deberá incluir al menos un patrón de diseño, alguna etiqueta de Razor NO VISTA en clase, sentencias LINQ y Entity framework para la base de datos.

Si el sistema contiene elementos adicionales como Jquery, Bootstrap y AJAX, se agregará puntos extra.

Deberá:

- Elaborar un manual técnico en donde indique como esta estructurado su programa, y que diga en donde se usaron los elementos requeridos (donde se usó x patrón, donde LINQ, etc)
- *Elaborar un manual de usuario donde se indique como se usa el sistema*
- Indicar en el manual de usuario, un enlace a un sitio como github o bitbucket con el codigo fuente compartido a la cuenta de un servidor para poderlo visualizar
- Exponerlo a un servidor resumiendo el funcionamiento del sistema

Introducción:

En este proyecto final, se realizó un sistema web de catálogo de ventas de ropa, llamada “Modees 90” , haciendo referencia a la moda de los años 90, consta de un Login, en el cual se inicia sesión con un usuario y contraseña, al iniciar satisfactoriamente (Que esté registrado en la base de datos) se pasa a otra página en la cual se platica un poco sobre la misión acerca de Modees.

En el menú, viene un apartado llamado “Catálogo”, al presionar , nos redirige a una página en la cual tenemos un listado de ropa, con sus atributos correspondientes y vienen añadidos métodos de pago

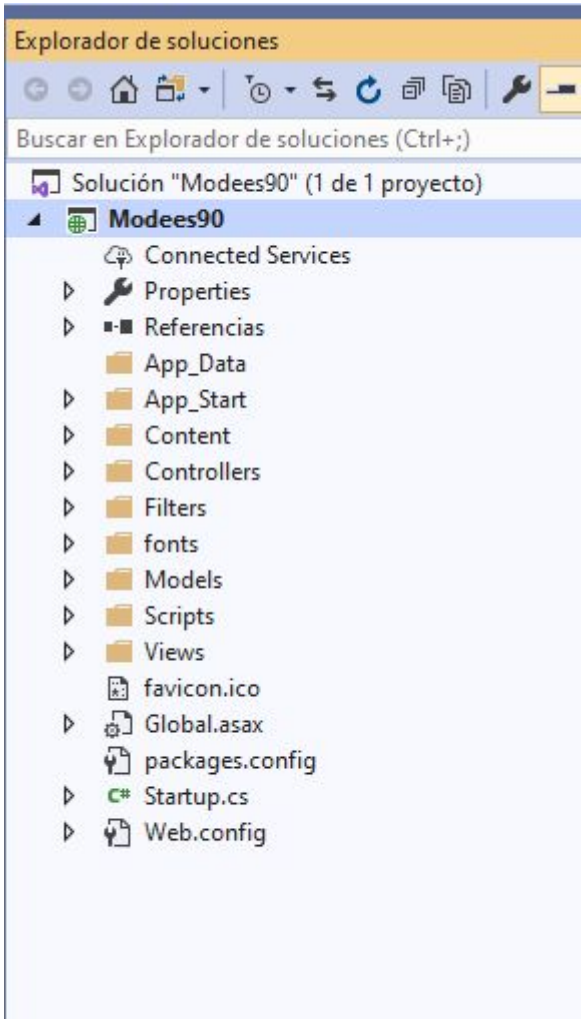
Objetivo

Elaborar un sistema que promocione nuestro catalogo de ventas y realice la compra, también en el aspecto técnico se creó este manual, para comprender cómo está estructurado nuestro sistema y cada una de sus partes con su funcionalidad

Creación

Al momento de crear el programa cree la aplicación con el MVC e instale los nuget de Entity framework en asp.net

Ya que se creó el modelo que son las clases, la vista es la interfaz, lo que se visualiza y el contenedor es el que comunica el modelo con la vista



Al paso de la programación cree varias carpetas y clases, filtros, entre otras cosas, pero iré detallando cada paso para mayor entendimiento

PASO 1:

Una vez creado nuestro proyecto nos dirigiremos a crear e importar nuestra base de datos, en mi caso yo cree mi base de datos con sus tablas corresponientes

- Crear base de datos

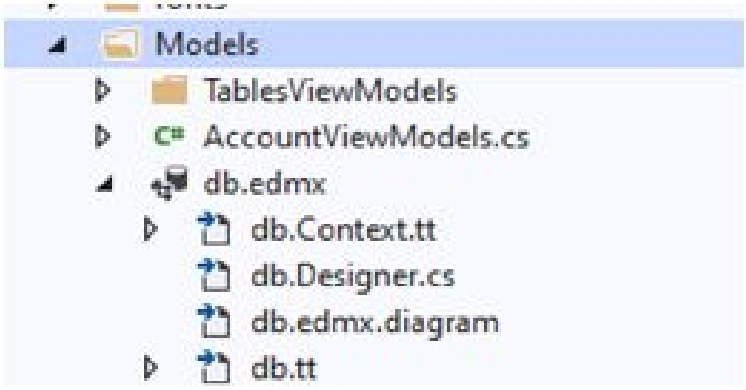
-Tengo 3 tablas InicioUsers, Estado, Catalogo

-En inicioUsers se almacenan los datos de inicio de sesión de los usuarios

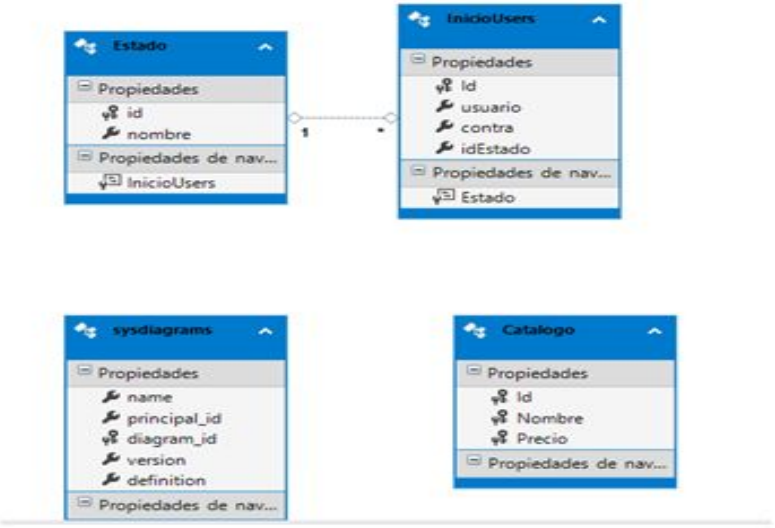
-Estado hace referencia al “Modo” en el que se encuentran Activo/ Inactivo/ Eliminado

-Catalogo hace referencia al inventarios disponible de las prendas de ropa Esta parte es simple, solo la creamos en la parte de models de nuestro proyecto Modees90, le agregamos un nombre y elegimos que tablas queremos mostrar en mi caso son todas

- Crear modelo- ADO.NET ENTITY DATA- NAME BD



Tablas> Estado> InicioUsers> Catalogo



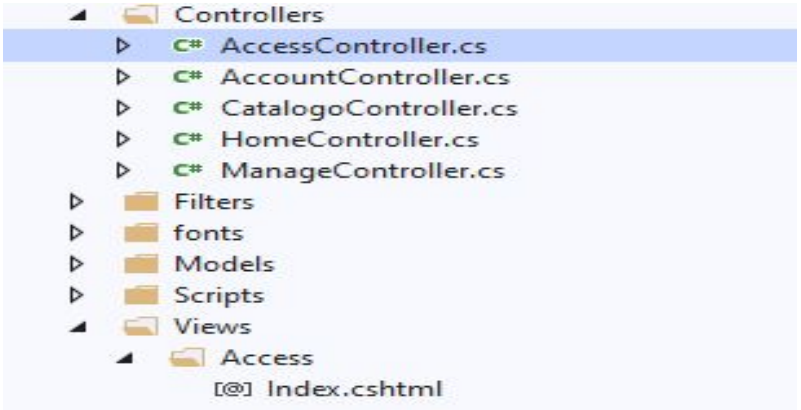
Una vez que tengamos importadas nuestra tablas ahora podremos utilizarla sy hacer referencia a ellas cuando queramos.

- Listo quedo creado nuestro Entity

PASO 2:

Creamos el inicio de sesión

Para eso creamos un controlador llamado “AccesController” y una vista que se encuentra Views > Access > index.cshtml



Abrimos el controlador de acceso e importamos las librerías

```
using System.Web.Mvc;
using Modees90.Models; // Obtiene el objeto Entity Framework

namespace Modees90.Controllers
{
```

Estructura

El propósito de este controlador es dar una respuesta es decir, manda un 1 si el usuario es válido y si no, manda un mensaje diciendo “Usuario invalido” para eso recibe dos parámetros, que son ”user” y “password” , realizamos validaciones con un try catch en dado caso, para prevenir incidente, además , hacemos referencia a nuestras entidades, las que creamos en el paso anterior, aquí creamos una list esta lista almacena todos los usuarios y hacemos referencia a sus atributos para que sean comparados con los recibidos en los cuadros de texto de inicio de sesión.

USO LINQ

Para realizar la búsqueda, para verificar que existe un usuario

```
public class AccessController : Controller
{
    // GET: Access
    0 referencias
    public ActionResult Index()
    {
        return View();
    }
    //Login recibe los parametros
    0 referencias
    public ActionResult Enter(string user, string password)
    {
        try
        {
            //Mi objeto
            using (Modees90Entities db= new Modees90Entities())
            {
                /* Utilizamos LinQ */
                //Consulta

                var lst = from d in db.InicioUsers
                          where d.usuario == user && d.contra == password && d.idEstado==1 // Solo seleccionar los usuarios activos

                          select d;

                if(lst.Count()>0)
                {

                    //Si es correcto
                    InicioUsers oUser = lst.First();//
                    Session["User"] = oUser;
                    return Content("1");
                }
                else
```

```
                    Session["User"] = oUser;
                    return Content("1");
                }
                else
                {

                    return Content("Usuario Invalido");
                }
            }
        }
        catch(Exception ex)
        {
            return Content("Ocurrio un Error :(" + ex.Message);
        }
    }
}
}
```

Ahora bien nos dirigimos a nuestro Index.cshtml creado en la carpeta de Views /Access

Estructura

Aquí simplemente capturamos los datos, de inicio de sesión y hacemos referencia a nuestro layout en la parte de arriba ya que queremos que primero cargue nuestro inicio de sesión, no el default, esto se explicara más adelante

USO de HTML

AccessController.cs* | NuGet: Modees90 | Index.cshtml | HomeController.cs | Contact.csht

```
@{
    ViewBag.Title = "Index";

    <!--Cargar primero nuestro layout-->
    Layout ="~/Views/Shared/_LayoutLogin.cshtml";
}

<!-- Crearemos un inicio de sesion-->
<div class="row">
    <form id="frm">
        <div class="col-lg-6 col-lg-offset-3">
            Usuario
            <input type="text" name="user" />
            Password
            <input type="password" name="password" />
            <input type="submit" class="btn btn-success" value="Entrar" />

        </div>
    </form>
</div>
```

USO de JS

En dado caso que reciba el 1 = Usuario valido, aquí hacemos uso también de **USO RAZOR** ya que , este nos permite agregar código/métodos/funciones que no necesariamente son de script , se usa en la @seccion y @url

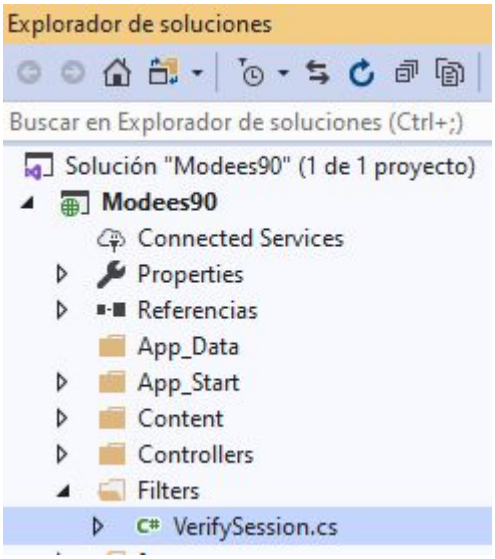
```
<!-- Jquery-->
<!-- Lo que hace La seccion es que la parte de codigo de abajo no se
    cargue en el RenderBody // Sino que se cargue en el jquery
     Ya que el codigo se ejecuta en el rander body anteriormente-->
@section scripts
{
    <!-- Scripts Razor y -->
    <script>

    $(document).ready(function () {

        $("#frm").submit(function (e) {
            e.preventDefault();

            url = "@Url.Content("~/Access/Enter")"  //Le da la ruta a JS
            parametros = $(this).serialize();

            $.post(url, parametros, function (data) {

                if (data == "1") {
                    document.location.href = "@Url.Content("~/")";
                }

               else {
                    alert(data);
                }
            })
        })
    })

    </script>
}
```

PASO 2:

Ahora bien ya tenemos hasta el momento nuestra base de datos, nuestro inicio de sesión , si agregamos nuestras credenciales nos dirige a la página de inicio, y si son inválidas, nos muestra el mensaje.
Pero que pasa si, al iniciar yo quiero que desde un principio me diriga a mi página de inicio de sesión(Localhost/Access/Index) y no a “Home/Inicio”, o bien que si estoy en inicio de sesion no me permite pasar a home, ya que no he puesto las credenciales, esto se puede hacer muy fácil poniendo en la ruta nuestro: Localhost/**Home/Index** y nos dirige sino tenemos filtros o seguridad; Pues bien eso no es correcto y se realizará a continuación.

Validar que no se pueda acceder a los controladores amenos que se inicie sesión

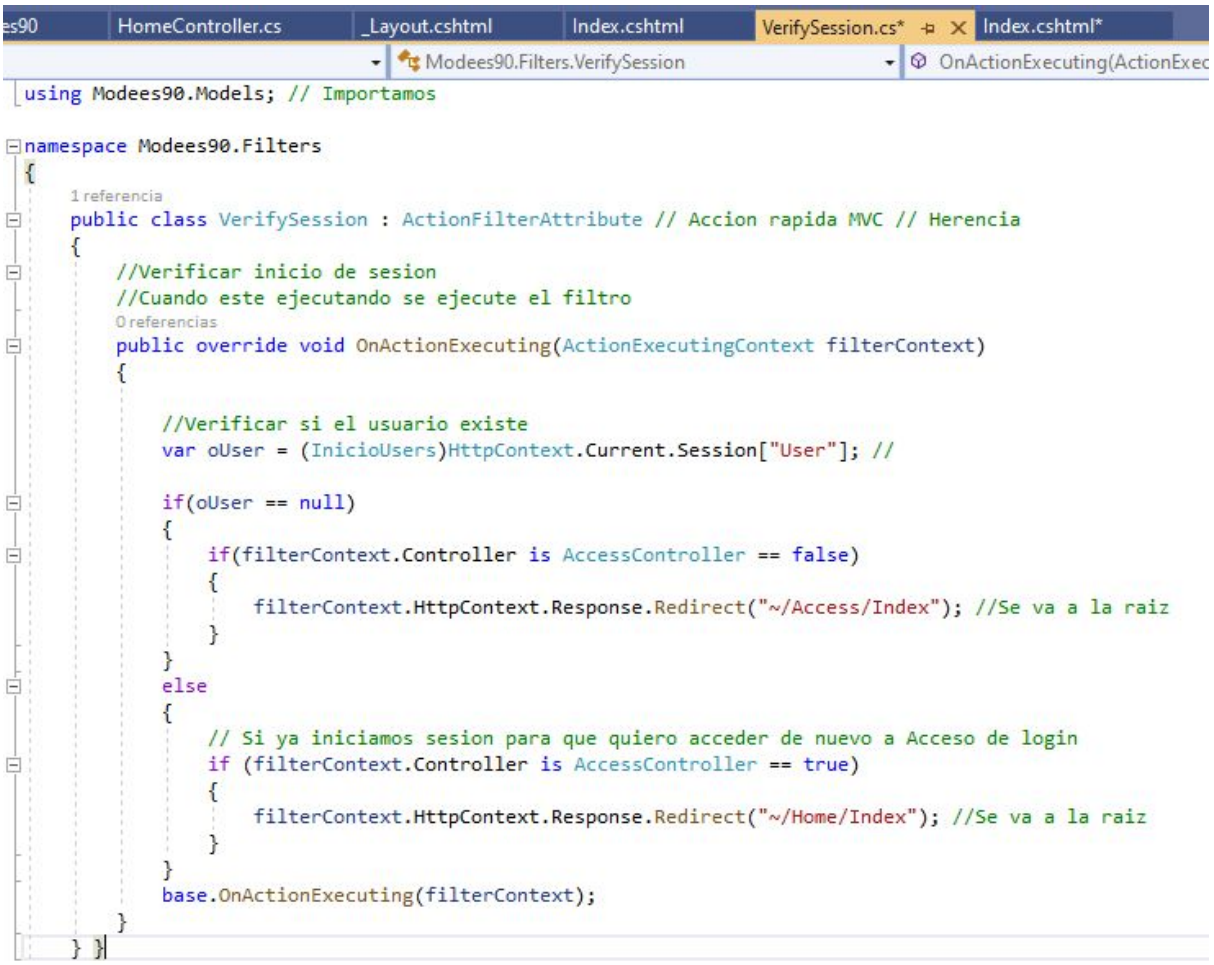
Para eso se realizan dos cosas primero se crea una carpeta donde se almacenarán nuestros filtro y dentro de esa carpeta almacenaremos una clase llamada verificar sesión



ESTRUCTURA

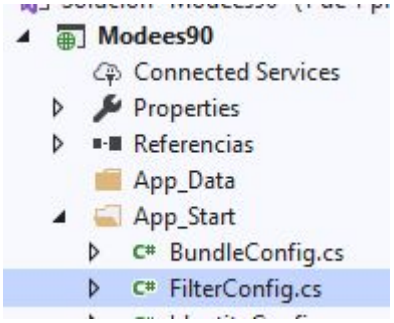
Verificar el inicio de sesión

Se realizan dos validaciones antes y después de iniciar sesión



```
es90 | HomeController.cs | _Layout.cshtml | Index.cshtml | VerifySession.cs* | Index.cshtml*
Modees90.Filters.VerifySession | OnActionExecuting(ActionExec
using Modees90.Models; // Importamos

namespace Modees90.Filters
{
    1 referencia
    public class VerifySession : ActionFilterAttribute // Accion rapida MVC // Herencia
    {
        //Verificar inicio de sesion
        //Cuando este ejecutando se ejecute el filtro
        0 referencias
        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {

            //Verificar si el usuario existe
            var oUser = (InicioUsers)HttpContext.Current.Session["User"]; //

            if(oUser == null)
            {
                if(filterContext.Controller is AccessController == false)
                {
                    filterContext.HttpContext.Response.Redirect("~/Access/Index"); //Se va a la raiz
                }
            }
            else
            {
                // Si ya iniciamos sesion para que quiero acceder de nuevo a Acceso de login
                if (filterContext.Controller is AccessController == true)
                {
                    filterContext.HttpContext.Response.Redirect("~/Home/Index"); //Se va a la raiz
                }
            }
            base.OnActionExecuting(filterContext);
        }
    } }
```

Ahora bien, ya hemos creado el filtro, ahora falta darlo de alta para eso nos dirigimos a Appstart > filterconfig



Esta clase está agradable porque conforme vayamos teniendo mas filtro , solo los agregamos aquí, sin tener que realizar muchas modificaciones o mucho uso de código

```
namespace Modees90
{
    1 referencia
    public class FilterConfig
    {
        1 referencia
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
            filters.Add(new Filters.VerifySession()); // Damos de alta el filtro

        }
    }
}
```

Creación de nuestro catálogo y mostrarlo en una vista

En esta parte crearemos nuestro propio layout ya que por default la aplicación te brinda uno, en ese layout nosotros mostraremos nuestra vista de catálogo de venta de productos y en la default solo mostraremos la página inicial, la modificaremos un poco agregaremos otras opciones, se realizan varias acciones , describiré en partes:

-Agregamos 3 opciones Inicio, Catálogo y Cerrar sesión

-

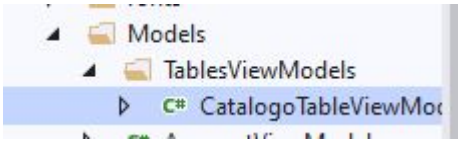
-Creamos este layout para que se muestre al momento de ejecutar el programa, ya que es default te ofrece más opciones y esas opciones no las debe tener una persona que no esta registrada

CatalogoTableViewModel.cs | _LayoutLogin.cshtml | CatalogoController.cs | _Layout.cshtml | Index.cshtml*

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Mi aplicación ASP.NET</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <!--Nombre de la aplicación-->
                @Html.ActionLink("Modees 90", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
            </div>


        </div>
    </div>
    <div class="container body-content">

        @RenderBody() <!--Aqui se muestra mi codigo-->
        <!--Referencias -->
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Mi aplicación ASP.NET</p> <!--Razor imprimir la fecha -->
        </footer>
    </div>
```

Ahora que ya tenemos ambas vistas hay que obtener los datos, guardar los datos, enviarlos de la base de datos para formar nuestra tabla de productos, y finalmente mostrarlo.

Creamos una carpeta y en ella agregamos una clase llamada CatalogoTableViewModel
En esta clase crearemos los atributos de nuestra tabla de base de datos catálogo, ya que aquí es donde se guardará el dato obtenido



Atributos y propiedades

```
namespace Modees90.Models.TablesViewModels
{
    4 referencias
    public class CatalogoTableViewModel
    {
        //Agregar los atributos que utilizare

        2 referencias
        public int Id { get; set; }
        2 referencias
        public string Nombre { get; set; }

        2 referencias
        public int Precio { get; set; }



    }
}
```

USO LinQ

Ahora bien, debemos obtener los datos de la base de datos, para eso creamos un nuevo controlador y una lista.

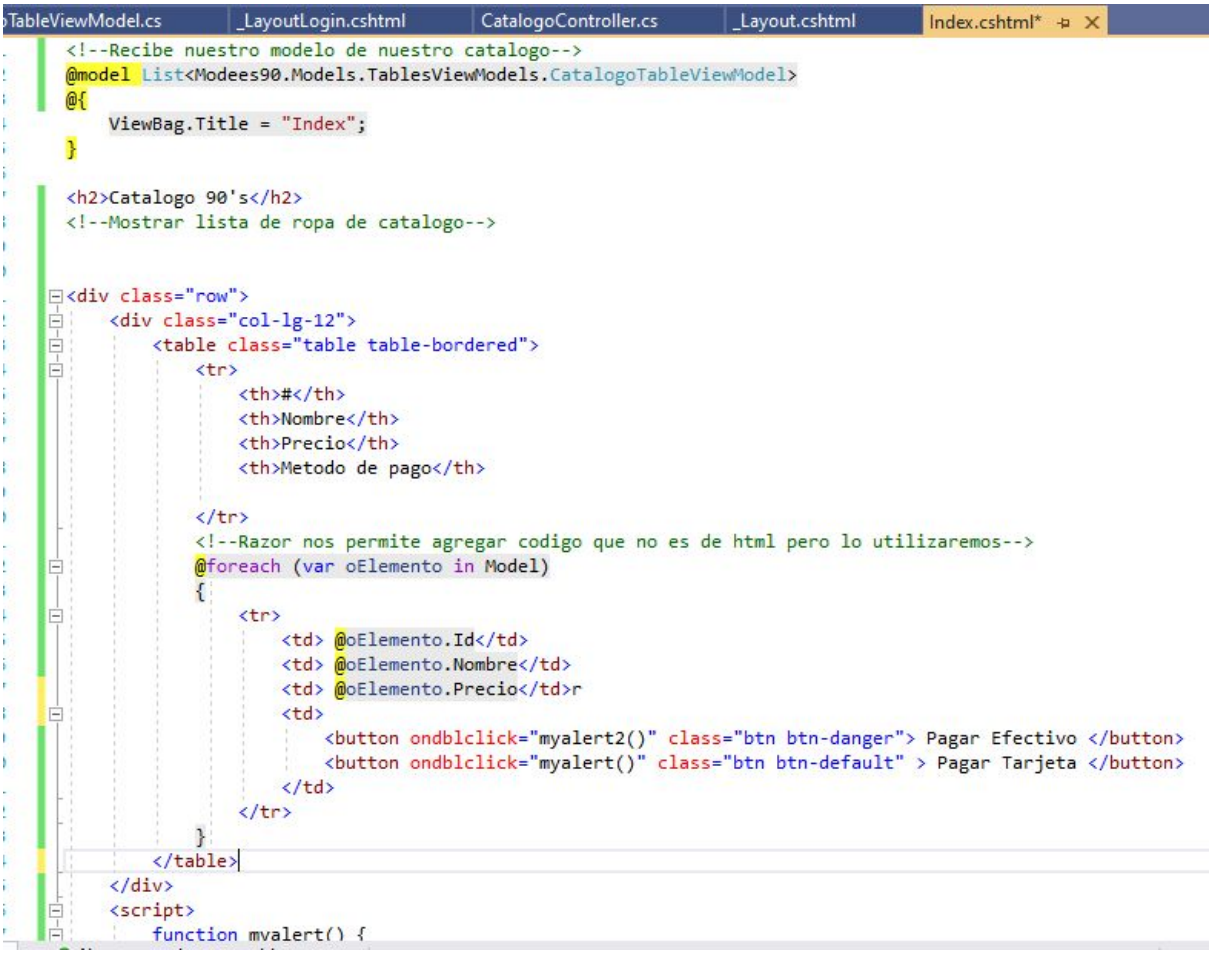
Se crea una lista de tipo de la clase pasada, CatalogoTableViewModel, se hace referencia a las entidades de la tabla, y se almacenan en lista(lst) , se usa linQ ya que se realizan expresiones más sencillas, y rápidas.

Para finalizar se regresa la lista a la vista.

```
ogoTableViewModel.cs | _LayoutLogin.cshtml | CatalogoController.cs | _Layout.csh
odees90 | Modees90.Controllers.CatalogoControlle
 8
 9      namespace Modees90.Controllers
10      {
            0 referencias
11          public class CatalogoController : Controller
12          {
13              // GET: Catalogo
                0 referencias
14              public ActionResult Index()
15              {
16                  //Mostrar informacion
17
18                  List<CatalogoTableViewModel> lst = null;
19                  using (Modees90Entities db = new Modees90Entities())
20                  {
21                      // Se utiliza LinQ
22                      // Ya recibimos los datos
23                      lst = (from d in db.Catalogo
24                             where d.Id > 0
25                             orderby d.Id
26                             select new CatalogoTableViewModel
27                             {
28                                 Nombre = d.Nombre,
29                                 Id = d.Id,
30                                 Precio = d.Precio
31
32                             }).ToList();
33
34
35                  }
36                  return View(lst);
37              }
38          }
39      }
```

Se crea nuestra vista de catálogo que recibe la lista, en la carpeta de catálogo /index

-Aquí recibe el modelo @model y creamos nuestra tabla y forma de catálogo en donde se mostrará en forma de lista y accedemos a los elementos mediante un foreach y agregamos botones para realizar las acciones de comprar mediante dos modalidades



Como resultado al dar clic en uno de los botones, nos mostrará una alerta diciendo compra realizada

```
                    <td>
                        <button ondblclick="myalert2()" class="btn btn-danger"> Pagar Efectivo </button>
                        <button ondblclick="myalert()" class="btn btn-default" > Pagar Tarjeta </button>
                    </td>
                </tr>
            }
        </table>
    </div>
    <script>
        function myalert() {

            alert("Pago con tarjeta exitoso");
        }
        function myalert2() {

            alert("Pago con efectivo exitoso");
        }
    </script>

</div>
```