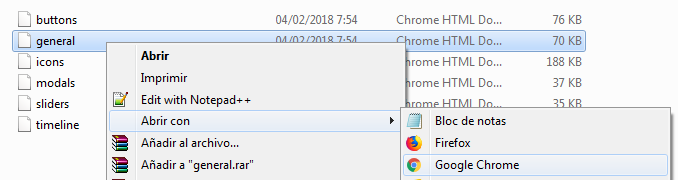
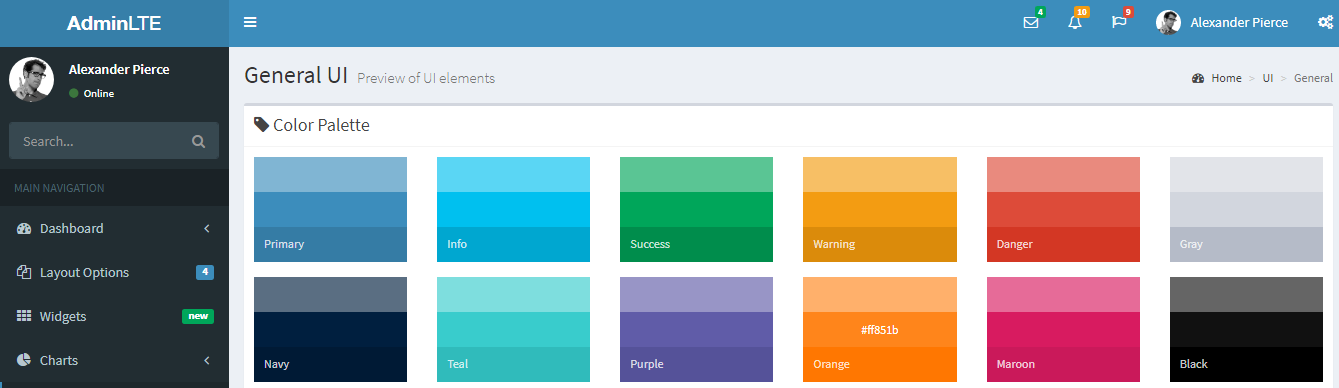
**Crear un Login y usar menú Personalizado**

1. **Crear Layout personalizados y Menus de opciones**.
2. **Crear el Layout Personalizado**.

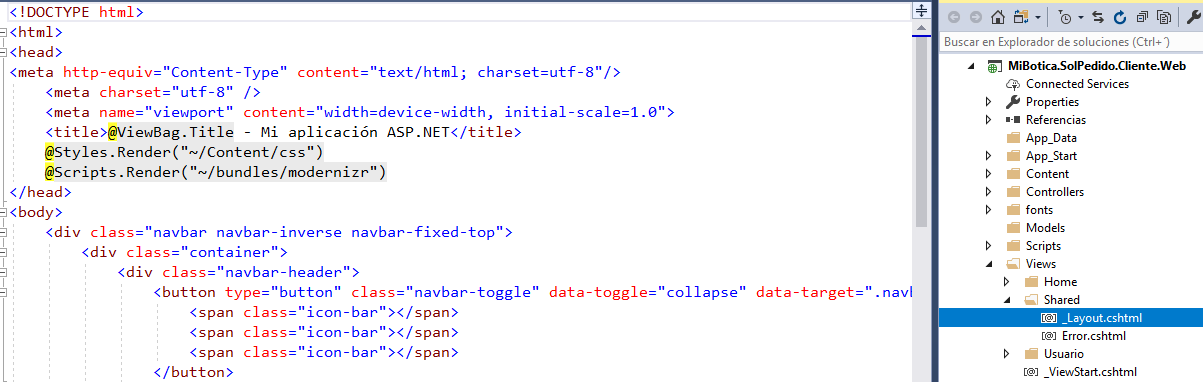
La página Layout es la página principal del proyecto Web. Para los cual se usara plantillas creadas que están a disposición de los usuarios. Para el ejemplo esta paginas está en : el archivo **AdminLTE-2.4.3.zip**, que se adjunta. Desempaquete el archivo u ubique la plantilla General.html que se encuentra en **… \ AdminLTE-2.4.3\ pages/UI/general.html**, presentarlo en el navegador, borrar la parte que no interesen. Y remplazar en el **Layout** de nuestro proyecto.



Como se ve en el navegador:



1. Ir al **Layout** ubicado en el explorador de soluciones en **02 FrontEnd** dentro del proyecto **ClienteWeb**. En la carpeta **Views** abrir la capeta **Shared** y allí localizar el archivo **\_Layout.cshtml**



Dar click sobre **\_Layout.cshtml** se despliega su contenido, comentar todo el contenido de este \_**Layout** , copiar el contenido del **general.html** y pegarlo dentro del \_Layout.

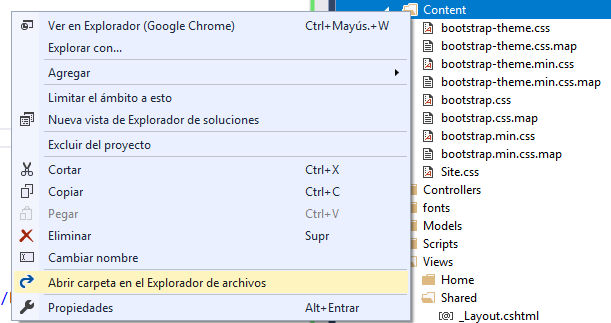
1. Copiar desde el lugar donde desempaqueto la plantilla (zip) los estilos y jquery al proyecto (clienteweb) dentro de la carpeta **Content**:

Las carpetas a copiar son: **bower\_components, build, dist, plugins**, que se encuentran en

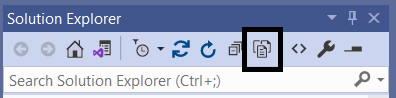
… Plantilla/AdminLTE-2.4.3/

|  |  |
| --- | --- |
|  |  |

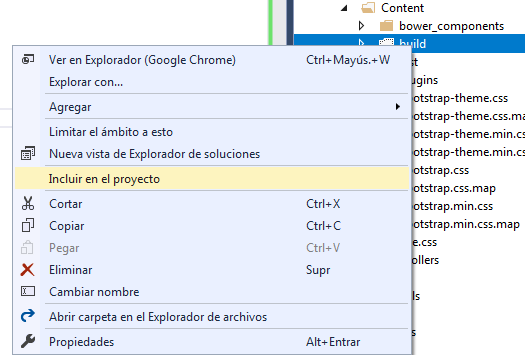
Para copiar desde el explorador de Windows copiar las carpetas:  **bower\_components, Build, dist, plugins** y desde el explorador de soluciones dar **click – derecho** a carpeta **Content** y escoger **abrir en Windows**.



Y pegar allí las carpetas copiadas. Luego desde el explorador de soluciones dar clik a botón **mostrar todos los archivos** **ocultos** y adjuntar las carpetas al proyecto.



Aparecen difuminados las carpetas pegadas y se pasan a adjuntarlos al proyecto.



1. En el proyecto de la carpeta.. **Content**/**bower\_components/fullcalendar/dist/ fullcalendar.d.ts**, ábralo y comentelo todo
2. En el proyecto ir al **\_layout** (que ahora tiene un contenido amplio) y para el ejemplo dejar la **cabecera (**head), **header class="main-header"** la parte de **la izquierda del menú** y el **pie** (footer)

En la parte superior incluir las referencias siguientes:

@using ABB.Catalogo.Entidades.Core;

@using ABB.Catalogo.Utiles.Helpers;

@using ABB.Catalogo.Entidades;

además de la sección **“**<!-- Control Sidebar -->**”**. Borrar la parte que esta dentro de la sección

***<section class="content">***

***De Línea 2266 …(parte que se debe borrar)…….a línea 3158.***

***</section>***

Y replazarlo por :

<section class="content">

<div class="row">

<div class="col-md-12">

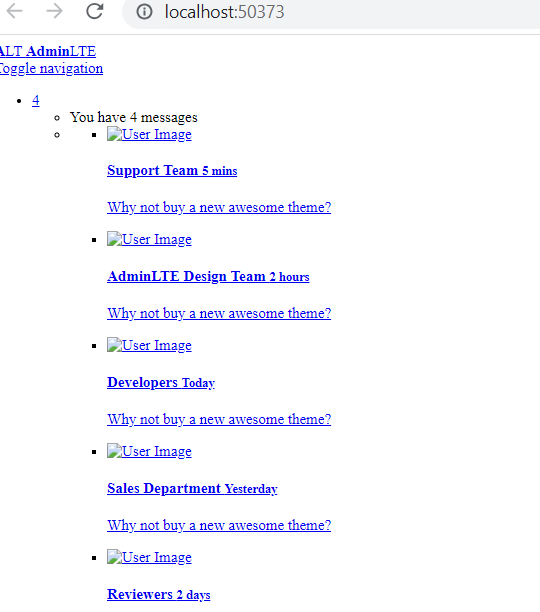
@RenderBody()

</div>

</div>

</section>

Cuando se corre se mostrara la vista sin estilos :



para corregir esto actualizar las referencias a los estilos de la parte superior (head) del **\_Layout.**

Para esto ubique la referencia dentro de la carpeta **Contens/Brower\_components /……** y arrástrelo en el **\_layout** remplazando a la referencia que están alli.



En <aside class="main-sidebar">….. </aside> se pondra la lógica para mostrar las opciones del menú según sea el usuario (invitado o usurio) para cada caso se presentara un menu diferente.



Ubique la sección <div class="content-wrapper"> en el \_layout (línea 580) y realice la siguiente modificación:

<div class="content-wrapper">

<!-- Content Header (Page header) -->

<section class="content-header">

<h1>

@RenderSection("seccionTitulo", false)

</h1>

</section>

<!-- Main content -->

<section class="content">

<div class="row">

<div class="col-md-12">

@RenderBody()

</div>

</div>

</section>

<!-- /.content -->

</div>

1. Actualizar referencias a los archivos de jquery en el **Layout**. Para esto ubique la referencia dentro de la carpeta **Contens/Brower\_components /..** y arrástrelo en el **\_layout** remplazando a la referencia que están alli.

En la parte final del **\_layout** actualice también las referencias asi:

<!-- jQuery 3 -->

@\*<script src="../../bower\_components/jquery/dist/jquery.min.js"></script>\*@

<script src="~/Content/bower\_components/jquery/dist/jquery.min.js"></script>

<!-- Bootstrap 3.3.7 -->

@\*<script src="../../bower\_components/bootstrap/dist/js/bootstrap.min.js"></script>\*@

<script src="~/Content/bower\_components/bootstrap/dist/js/bootstrap.min.js"></script>

<!-- FastClick -->

@\*<script src="../../bower\_components/fastclick/lib/fastclick.js"></script>\*@

<script src="~/Content/bower\_components/fastclick/lib/fastclick.js"></script>

<!-- AdminLTE App -->

@\*<script src="../../dist/js/adminlte.min.js"></script>\*@

<script src="~/Content/dist/js/adminlte.min.js"></script>

<!-- AdminLTE for demo purposes -->

@\*<script src="../../dist/js/demo.js"></script>\*@

<script src="~/Content/dist/js/demo.js"></script>

@RenderSection("scripts", required: false)

<script>

$(function () {

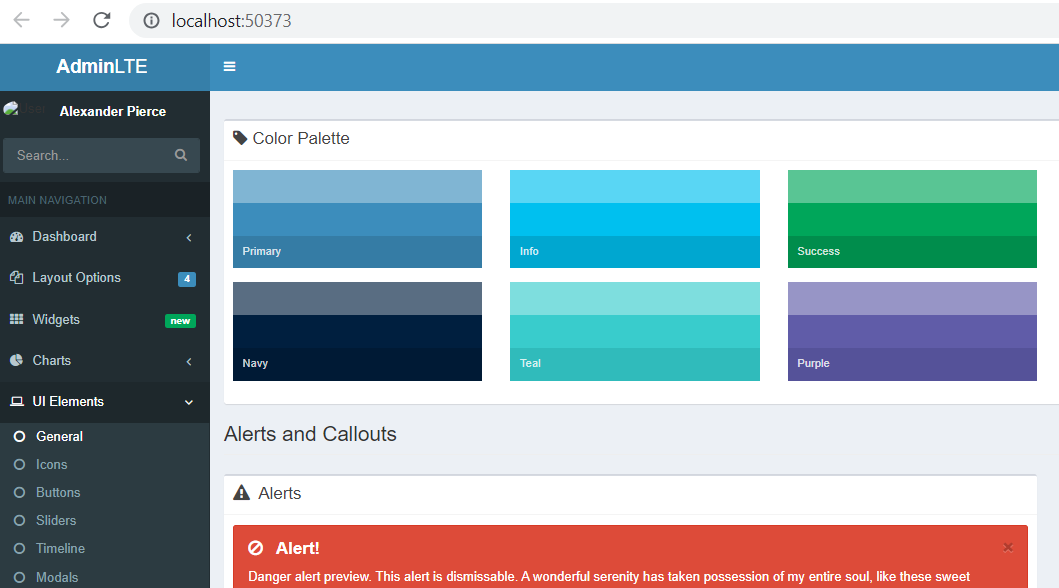
CKEDITOR.replace('editor1')

$('.textarea').wysihtml5()

})

</script>

Corre la aplicación



1. **Proceso para crear los Menus personalizados**:

En la capa de **ABB.Catalogo.Entidades** actualizar la sección **References** adicionando la referencia a **System.Web**

En la capa de Entidades en la carpeta **Core** crear la clase ***Opcion***:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ABB.Catalogo.Entidades.Core

{

public class Opcion

{

public int IdOpcion { get; set; }

public string NombreOpcion { get; set; }

public string UrlOpcion { get; set; }

public string RutaImagen { get; set; }

public int NroOrden { get; set; }

public int IdOpcionRef { get; set; }

// adicionales para deferenciar el controlador y el metodo en el string UrlOpcion

public string Controladora { get; set; }

public string Accion { get; set; }

public string Area { get; set; }

}

}

Crear la clase **VariablesWeb** para las variables genéricas en la raíz de la capa **ABB.Catalogo.Entidades**

using ABB.Catalogo.Entidades.Core;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web;

namespace ABB.Catalogo.Entidades

{

public class VariablesWeb

{

public static List<Opcion> gOpciones

{

get

{

if (HttpContext.Current.Session["gOpciones"] != null)

{

return (List<Opcion>)HttpContext.Current.Session["gOpciones"];

}

return null;

}

set

{

HttpContext.Current.Session["gOpciones"] = value;

}

}

public static Usuarios gUsuario

{

get

{

if (HttpContext.Current.Session["gUsuario"] != null)

{

return (Usuarios)HttpContext.Current.Session["gUsuario"];

}

return null;

}

set { HttpContext.Current.Session["gUsuario"] = value; }

}

}

}

1. En el **\_layout** en la parte inicial agregar las referencias a la entidad **Usuario** y al **Helper**. Para poder poner al usuario que se autentique y llamar a la encriptación de la clave.

@using ABB.Catalogo.Entidades.Core;

@using ABB.Catalogo.Utiles.Helpers;

@using ABB.Catalogo.Entidades;

<!DOCTYPE html>

1. En el **Layout** corregir la cabecera del Usuario; estableciendo la diferencia entre **Usuario invitado** y **usuario autenticado**.(ubicar en **Layout**: “class="dropdown user user-menu"

Debajo ingrese el siguiente código:

<li class="dropdown user user-menu">

@if (User.Identity.IsAuthenticated && VariablesWeb.gUsuario != null)

{

<a href="#" class="dropdown-toggle" data-toggle="dropdown">

<img src="../../dist/img/user2-160x160.jpg" class="user-image" alt="User Image">

<span class="hidden-xs">Alexander Pierce</span>

</a>

<ul class="dropdown-menu">

<!-- User image -->

<li class="user-header">

<img src="../../dist/img/user2-160x160.jpg" class="img-circle" alt="User Image">

<p>

Alexander Pierce - Web Developer

<small>Member since Nov. 2012</small>

</p>

</li>

<!-- Menu Body -->

<li class="user-body">

<div class="row">

<div class="col-xs-4 text-center">

<a href="#">Followers</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Sales</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Friends</a>

</div>

</div>

<!-- /.row -->

</li>

<!-- Menu Footer-->

<li class="user-footer">

<div class="pull-left">

<a href="#" class="btn btn-default btn-flat">Profile</a>

</div>

<div class="pull-right">

<a href="#" class="btn btn-default btn-flat">Sign out</a>

</div>

</li>

</ul>

}

else

{

<a href="#" class="dropdown-toggle" data-toggle="dropdown">

<img src="../../dist/img/user2-160x160.jpg" class="user-image" alt="User Image">

<span class="hidden-xs">Invitado</span>

</a>

<ul class="dropdown-menu">

<!-- User image -->

<li class="user-header">

<img src="../../dist/img/user2-160x160.jpg" class="img-circle" alt="User Image">

<p>

Alexander Pierce - Web Developer

<small>Member since Nov. 2012</small>

</p>

</li>

<!-- Menu Body -->

<li class="user-body">

<div class="row">

<div class="col-xs-4 text-center">

<a href="#">Followers</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Sales</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Friends</a>

</div>

</div>

<!-- /.row -->

</li>

<!-- Menu Footer-->

<li class="user-footer">

<div class="pull-left">

<a href="#" class="btn btn-default btn-flat">Profile</a>

</div>

<div class="pull-right">

<a href="#" class="btn btn-default btn-flat">Sign out</a>

</div>

</li>

</ul>

}

</li>

1. Para el caso de usuario autenticado:

cambiar la referencia de **user2-160x160.jpg** a **user8**, para esto desde carpeta **Content/dist/img/user8-128x128.jpg** ,arrástrelo y comente el anterior.

Para usuario autenticado: cambiar <span class="hidden-xs">Alexander Pierce</span> por

<span class="hidden-xs">@VariablesWeb.gUsuario.Nombres</span>

Ademas cambiar :

Alexander Pierce - Web Developer por

@VariablesWeb.gUsuario.IdUsuario - @VariablesWeb.gUsuario.nombres

Queda asi:

<a href="#" class="dropdown-toggle" data-toggle="dropdown">

@\*<img src="../../dist/img/user2-160x160.jpg" class="user-image" alt="User Image">\*@

<img src="~/Content/dist/img/user8-128x128.jpg" class="user-image" alt="User Image" />

<span class="hidden-xs">@VariablesWeb.gUsuario.Nombres</span>

</a>

<ul class="dropdown-menu">

<!-- User image -->

<li class="user-header">

<img src="~/Content/dist/img/user8-128x128.jpg" class="img-circle" alt="User Image">

<p>

@VariablesWeb.gUsuario.IdUsuario - @VariablesWeb.gUsuario.Nombres

<small>Member since Nov. 2012</small>

</p>

……………………………………………………………….

……………………………………………………………..

En la parte del **Menu Footer** en la **div class="pull-right"** agregar lo siguiente:

<a href="@Url.Action("CerrarSesion", "Login")"

class="btn btn-default btn-flat">Cerrar Sesion</a>

comentar la anterior href.

1. Para el caso del usuario **no autenticado** (**invitado**); la parte del **else** realizar los siguientes cambios:

Cambie la referencia el usuario arrastrando desde la carpeta : **Content/dist/img** la imagen **user2-160x160.jpg**. Comente la referencia anterior.

Cambie el usuario **Alexande Pierre** por **Invitado**.

De la misma manera en la **class="user-header"** cambie la referencia del usuario por la imagen user2-160x160.jpg, arrastrado del mismo lugar anterior. Y Alexander Pierre cámbielo por Usuario Invitado.

Queda asi:



Ademas:

Comente todo el contenido de <div class="row">, como es invitado no debe tener referencias a acciones de menú.

En el <!-- Menu Footer--> en la <div class="pull-right"> comentar el contenido actual y agregar:

<a href="@Url.Action("Index", "Login")" class="btn btn-default btn-flat">Iniciar Sesion</a>

Esta referencia indica que se ejecutar un método para Autenticarse en el controlador Login; que se activara al pulsar un boton Iniciar Secion.

1. Finalmente el código de esta parte del \_layout quedara asi:

<li class="dropdown user user-menu">

@if (User.Identity.IsAuthenticated && VariablesWeb.gUsuario != null)

{

<a href="#" class="dropdown-toggle" data-toggle="dropdown">

@\*<img src="../../dist/img/user2-160x160.jpg" class="user-image" alt="User Image">

<span class="hidden-xs">Alexander Pierce</span>\*@

<img src="~/Content/dist/img/user8-128x128.jpg" class="user-image" alt="User Image"/>

<span class="hidden-xs">@VariablesWeb.gUsuario.Nombres</span>

</a>

<ul class="dropdown-menu">

<!-- User image -->

<li class="user-header">

<img src="~/Content/dist/img/user8-128x128.jpg" class="img-circle" alt="User Image">

<p>

@VariablesWeb.gUsuario.IdUsuario - @VariablesWeb.gUsuario.Nombres

<small>Member since Nov. 2012</small>

</p>

</li>

<!-- Menu Body -->

<li class="user-body">

<div class="row">

<div class="col-xs-4 text-center">

<a href="#">Followers</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Sales</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Friends</a>

</div>

</div>

<!-- /.row -->

</li>

<!-- Menu Footer-->

<li class="user-footer">

<div class="pull-left">

<a href="#" class="btn btn-default btn-flat">Profile</a>

</div>

<div class="pull-right">

<a href="@Url.Action("CerrarSesion", "Login")"

class="btn btn-default btn-flat">Cerrar Sesion</a>

@\*<a href="#" class="btn btn-default btn-flat">Sign out</a>\*@

</div>

</li>

</ul>

}

else

{

<a href="#" class="dropdown-toggle" data-toggle="dropdown">

@\*<img src="../../dist/img/user2-160x160.jpg" class="user-image" alt="User Image">\*@

<img src="~/Content/dist/img/user2-160x160.jpg" class="user-image" alt="User Image"/>

<span class="hidden-xs">Invitado</span>

</a>

<ul class="dropdown-menu">

<!-- User image -->

<li class="user-header">

@\*<img src="../../dist/img/user2-160x160.jpg" class="img-circle" alt="User Image">\*@

<img src="~/Content/dist/img/user2-160x160.jpg" class="user-image" alt="User Image"/>

<p>

Usuario no autenticado

<small>Member since Nov. 2012</small>

</p>

</li>

<!-- Menu Body -->

<li class="user-body">

@\*<div class="row">

<div class="col-xs-4 text-center">

<a href="#">Followers</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Sales</a>

</div>

<div class="col-xs-4 text-center">

<a href="#">Friends</a>

</div>

</div>\*@

<!-- /.row -->

</li>

<!-- Menu Footer-->

<li class="user-footer">

<div class="pull-left">

@\*<a href="#" class="btn btn-default btn-flat">Profile</a>\*@

</div>

<div class="pull-right">

<a href="@Url.Action("Index", "Login")" class="btn btn-default btn-flat">Iniciar Sesion</a>

@\*<a href="#" class="btn btn-default btn-flat">Sign out</a>\*@

</div>

</li>

</ul>

}

</li>

1. Actualizar en el **Web.Config** la sección de autenticación: ubica el tag **<system.web>** y poner el modo de autenticación por formulario.

<authentication mode="Forms">

forms loginUrl="Login/Index"></forms>

</authentication>

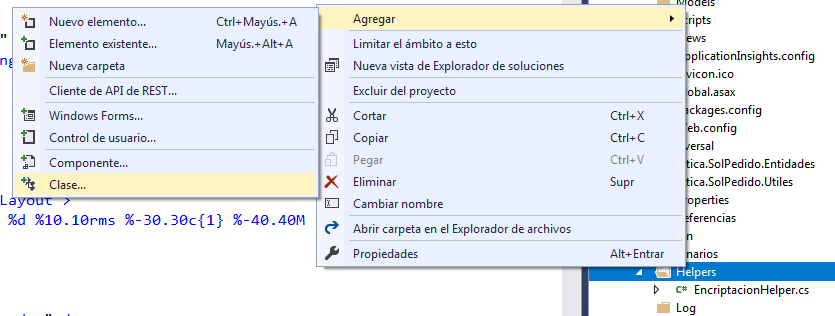
Para la autenticación via formulario de login, **loginUrl** indica que se ejecutara el método **index** del controlador **Login**.

1. Creación de un menú desde una tabla de opciones en la BD: un menú común es una árbol jerárquico cuya estructura se explica en el siguiente cuadro:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre opción** | **Nro Orden** | **Opción referencia** | **url** |
| **Mantenimiento** | 1 | 0 | # |
| ***Productos*** | 2 | 1 | Producto/index |
| ***Vendedor*** | 3 | 1 | Vendedor/index |
| ***Maestras*** | 4 | 4 | # |
| ***Categorias*** | 1 | 4 | Categoria/index |
| ***Roles*** | 2 | 4 | Rol/index |
| **Seguridad** | 2 | 0 | # |
| ***Usuario*** | 1 | 2 | Usuarios/index |
| **Ver Productos** | 3 | 0 | Producto/index |

Para crear esta tabla corra el scrip: **ScripTablaOpcionesPA.sql** que se adjunta también **SQLQueryOPPA.sql** para crear pa**.**

1. Realizar el menú dinámico desde la BD: Para lo cual se creara en **ABB.Catalogo.Utiles/Helpers** la clase publica estática **MenuHelper**.



En el Tiers ***ABB.Catalogo.Utiles***, en la sección Referencias aumentar la referencias siguientes: **System.Web.Mvc, System.Web.WebPages (esta en binarios) System.Web,Routing, System.Web y ABB.Catalogo.Entidades**.

La clase MenuHelper quedaría así:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

using System.Web.Routing;

using ABB.Catalogo.Entidades.Core;

namespace ABB.Catalogo.Utiles.Helpers

{

public static class MenuHelper

{

public static MvcHtmlString HelperMenu(this HtmlHelper helper, List<Opcion> listaOpciones, string titulo)

{

//.................................................

// final eleaborado por miguel algoritmo recursivo para construir el muenu.

//////////////////////////////////////////////////

var principal = new TagBuilder("ul");

principal.AddCssClass("sidebar-menu");

principal.Attributes.Add("data-widget", "tree");

var liHeader = new TagBuilder("li");

liHeader.AddCssClass("header");

liHeader.InnerHtml = titulo;

principal.InnerHtml += liHeader.ToString();

if (listaOpciones == null || listaOpciones.Count() == 0)

{

return new MvcHtmlString(principal.ToString());

}

bool tieneHijo = false;

foreach (Opcion item in listaOpciones

.Where(t => t.IdOpcionRef == 0)

.OrderBy(r => r.NroOrden).ToList())

{

tieneHijo = listaOpciones.Where(t => t.IdOpcionRef == item.IdOpcion).Any();

var itemLista = new TagBuilder("li");

if (tieneHijo)

{

itemLista.AddCssClass("treeview");

}

var linkLista = new TagBuilder("a");

linkLista.Attributes["href"] = GeneraHRef(helper, item);

var iLista = new TagBuilder("i");

iLista.AddCssClass(item.RutaImagen);

linkLista.InnerHtml += iLista.ToString();

var spanLista = new TagBuilder("span");

spanLista.InnerHtml = item.NombreOpcion;

linkLista.InnerHtml += spanLista;

if (tieneHijo)

{

var spanHijo = new TagBuilder("span");

spanHijo.AddCssClass("pull-right-container");

var iHijo = new TagBuilder("i");

iHijo.AddCssClass("fa fa-angle-left pull-right");

spanHijo.InnerHtml += iHijo.ToString();

linkLista.InnerHtml += spanHijo;

}

itemLista.InnerHtml += linkLista.ToString();

if (tieneHijo)

{

LlenarOpcionMenu(itemLista, item, listaOpciones, helper);

}

principal.InnerHtml += itemLista.ToString();

}

return new MvcHtmlString(principal.ToString());

}

// Lazy patre recursiva del menu

private static void LlenarOpcionMenu(TagBuilder itemLista, Opcion item, List<Opcion> listaOpciones, HtmlHelper helper)

{

var ulHijo = new TagBuilder("ul");

ulHijo.AddCssClass("treeview-menu");

foreach (Opcion itemOpcion in (from x in listaOpciones

where x.IdOpcionRef == item.IdOpcion

select x).ToList())

{

bool tieneHijo = (from x in listaOpciones

where x.IdOpcionRef == itemOpcion.IdOpcion

select x).Any();

var liHijo = new TagBuilder("li");

if (tieneHijo)

{

liHijo.AddCssClass("treeview");

}

var aHijo = new TagBuilder("a");

aHijo.Attributes["href"] = GeneraHRef(helper, itemOpcion);

var iHijo = new TagBuilder("i");

iHijo.AddCssClass(itemOpcion.RutaImagen);

aHijo.InnerHtml += iHijo;

aHijo.InnerHtml += itemOpcion.NombreOpcion;

if (tieneHijo)

{

var spanHijo = new TagBuilder("span");

spanHijo.AddCssClass("pull-right-container");

var iHijo2 = new TagBuilder("i");

iHijo2.AddCssClass("fa fa-angle-left pull-right");

spanHijo.InnerHtml += iHijo2.ToString();

aHijo.InnerHtml += spanHijo;

}

liHijo.InnerHtml += aHijo.ToString();

if (tieneHijo)

{

LlenarOpcionMenu(liHijo, itemOpcion, listaOpciones, helper);

}

ulHijo.InnerHtml += liHijo.ToString();

}

itemLista.InnerHtml += ulHijo.ToString();

}

private static string GeneraHRef(HtmlHelper helper, Opcion item)

{

string rutaUrl = string.Empty;

UrlHelper urlHelper = new UrlHelper(helper.ViewContext.RequestContext);

if (!string.IsNullOrEmpty(item.UrlOpcion) && item.UrlOpcion != "#")

{

string[] ruta = item.UrlOpcion.Split('/');

switch (ruta.Count())

{

case 1:

rutaUrl = urlHelper.Action("Index", ruta[0]);

break;

case 2:

if (ruta[1].Split('?').Count() > 1)

{

rutaUrl = urlHelper.Action(ruta[1].Split('?')[0], ruta[0], RetornaObjetoParametros(ruta[1].Split('?')[1]));

}

else

{

rutaUrl = urlHelper.Action(ruta[1], ruta[0]);

}

break;

case 3:

if (ruta[2].Split('?').Count() > 1)

{

rutaUrl = urlHelper.Action(ruta[2].Split('?')[0], ruta[0] + "/" + ruta[1], RetornaObjetoParametros(ruta[2].Split('?')[1]));

}

else

{

rutaUrl = urlHelper.Action(ruta[2], ruta[0] + "/" + ruta[1]);

}

break;

default:

rutaUrl = item.UrlOpcion;

break;

}

}

else

{

rutaUrl = "#";

}

return rutaUrl;

}

private static RouteValueDictionary RetornaObjetoParametros(string parametro)

{

RouteValueDictionary rvd = new RouteValueDictionary();

string[] parametros = parametro.Split('&');

Dictionary<string, string> lista = new Dictionary<string, string>();

foreach (string item in parametros)

{

rvd.Add(item.Split('=')[0], item.Split('=')[1]);

//lista.Add(item.Split('=')[0], item.Split('=')[1]);

}

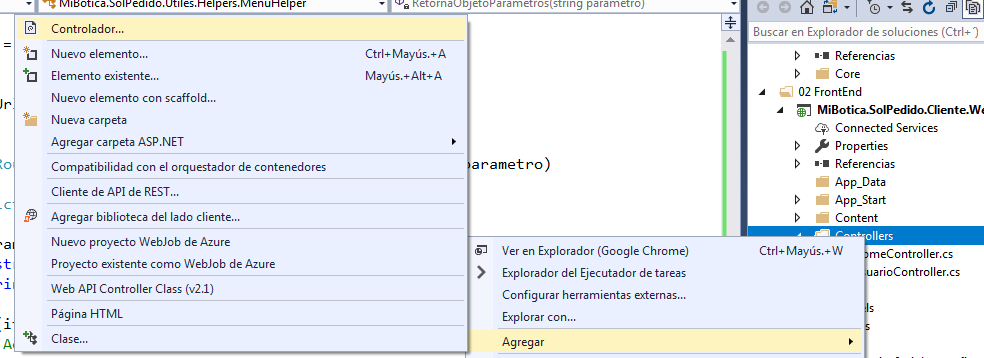
return rvd;

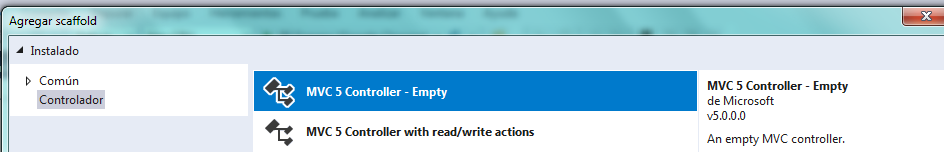
}

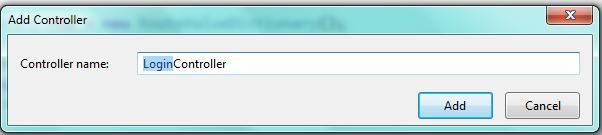
}

}

1. Crear el Controlador **Login** (en el cliente) para el proceso de autenticación. se crea vacío.







Aumentar las siguientes referencias a la clase creada.

la case quedaria asi:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using ABB.Catalogo.Entidades.Core;

using ABB.Catalogo.Utiles.Helpers;

using System.Security;

namespace ABB.Catalogo.ClienteWeb.Controllers

{

public class LoginController : Controller

{

// GET: Login

public ActionResult Index()

{

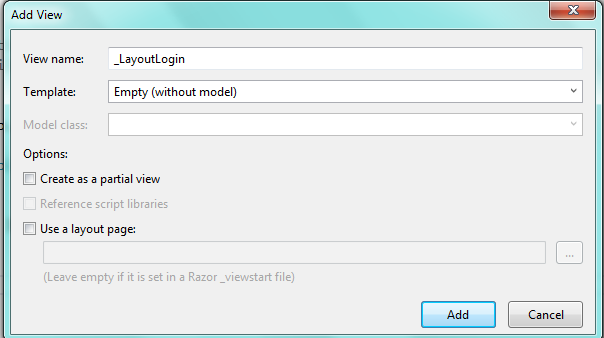
return View();

}

}

}

1. Crear un nuevo Layout: \_***LayoutLogin*** para la autenticación, esto porque todas las vistas no necesitan usar el mismo layout, en este caso la vista asociada al controlador Login debe tener un Layout: mínimo. Desde la carpeta **Share** dar clik derecho agregar una vista ponerle nombre **\_LayoutLogin** que **no usara** el layaut por defecto.



1. A este **\_LayoutLogin** borrar el contenido de la cabecera (**head**) .Copiar del \_**Layout** original (\_Layout) la parte de la cabecera (head) y pegarlo en **\_LayoutLogin en la seccion head** ; y dentro del **body** borrar la **div** y poner **solo @RenderBody()**. Esto es un **layuot** mínimo.

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>AdminLTE 2 | General UI</title>

<meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">

<link href="~/Content/bower\_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet" />

<link href="~/Content/bower\_components/font-awesome/css/font-awesome.min.css" rel="stylesheet" />

<link href="~/Content/bower\_components/Ionicons/css/ionicons.min.css" rel="stylesheet" />

<link href="~/Content/dist/css/AdminLTE.min.css" rel="stylesheet" />

<link href="~/Content/dist/css/skins/\_all-skins.min.css" rel="stylesheet" />

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600,700,300italic,400italic,600italic">

<style>

.color-palette {

height: 35px;

line-height: 35px;

text-align: center;

}

.color-palette-set {

margin-bottom: 15px;

}

.color-palette span {

display: none;

font-size: 12px;

}

.color-palette:hover span {

display: block;

}

.color-palette-box h4 {

position: absolute;

top: 100%;

left: 25px;

margin-top: -40px;

color: rgba(255, 255, 255, 0.8);

font-size: 12px;

display: block;

z-index: 7;

}

</style>

</head>

<body>

@RenderBody()

</body>

</html>

1. Desde el controlador **Login** : método **index** crear la **vista vacía** , que usara el **\_LayoutLogin** a esta vista se le pasara una instancia de **usuario**. (para esto la vista será Empty, tipada con Usuario y dar clik a usar layuot y buscar el **\_layuotLogin** creado)

En el controlador **Login** el método debe quedar así:

public ActionResult Index()

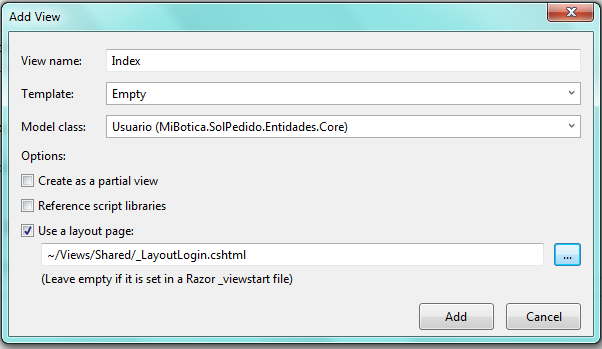
{

Usuario u = new Usuario();

return View(u);

}

Para crear la vista desde el método **Index** del controlador **Login** dar clik derecho sobre la sentencia **return View(u),** y pulsar agregar vista.



1. **Crear nuestra vista de autenticación**: Ir a la vista **Index** recién creada alli borre el tag <h2>Index</h2>.

desde la **plantilla/AdmiLT/pages/example** abra y copia la página **Login** peguelo en la vista **Index** abierta, y borrar la parte que no sea necesaria. Borrar:

* todo el **Head** (ya esta en el Layuot)
* la primera etiqueta **<html> (solo la etiqquta también la que cierra)**
* la etiqueta **Body (solo la etiqquta también la que cierra),**  ya está en el Layuot.
* La sección de script ya esta en el Layuot.

Cambia ***Sign in to start your sesión*** por **Ingrese sus credenciales**.

Cambie tambien:

action="../../index2.html" **por** action="@Url.Action("Index", "Login")"

<input type="email" class="form-control" placeholder="Email">

**Por**

<input type="text" class="form-control" placeholder="Usuario" name="CodUsuario" id="CodUsuario" value="@Model.CodUsuario">

Cambia

<input type="password" class="form-control" placeholder="Password">

Por

<input type="password" class="form-control" placeholder="Password" name="ClaveTexto" id="ClaveTexto" value="@Model.ClaveTex">

Agregar: despues de la </div> que es la que cierra la <div class="form-group has-feedback">

@if (!ViewData.ModelState.IsValid)

{

<div class="form-group alert alert-info">

@Html.ValidationSummary(false, "Hubo errores")

</div>

}

Borrar la div: <div class="checkbox icheck">

Cambiar Sign In por Validar

Comentar. La seccion <div class="social-auth-links text-center">

Comente la dos secciones <a href="#">

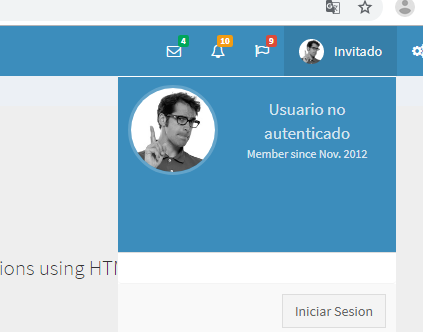
1. En el **\_Layuot** verifique y cambiar : en la parte de **usuario no autenticado**: **footer**

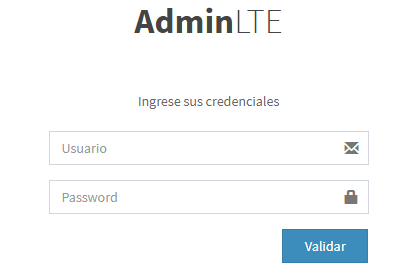
<a href="#" class="btn btn-default btn-flat">Sign out</a>

Por

<a href="@Url.Action("Index", "Login")" class="btn btn-default btn-flat">Iniciar Sesion</a> esto es para desplegar el Login.

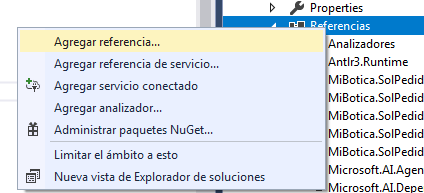
1. Pruebe la aplicación: al desplegar la aplicación se verifica que el usuario es Invitado al pulsar sobre el aparece la opción **Iniciar Sesión**, al pulsar esta opción se despliega la vista de autenticación. Ahora se trata de darle lógica a el proceso de autenticación.



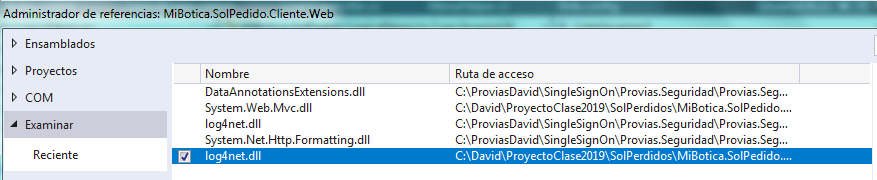


Al pulsar el boton Validar no ejecuta nada la funcionalidad de este botón lo completaremos más tarde por ahora haremos cosas previas.

1. En la sección Referencias del Proyecto Mvc : ABB.Catalogo.ClienteWeb, incluir la referencia al Log4Net (si ya lo puso ok). Para esto desde Referencias dar clik derecho, Agregar Referencias:



Se despliega un formulario escoger la opción Examinar, buscar en la carpeta Binarios del proyecto y escoger Log4Net y incluirlo al proyecto



1. Una vez instalado configurar en el **Web.config** la sección de log y en el **globalAsa**., así como la clase **base**

El Web.Config quera así:

<configSections>

<section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,log4net" />

</configSections>

<log4net>

<appender name="RollingLogFileAppender" type="log4net.Appender.RollingFileAppender">

<file type="log4net.Util.PatternString" value=" D:\Log\ABBCatalogo\LabDiseño\_%env{computername}.log " />

<appendToFile value="true" />

<rollingStyle value="Date" />

<datePattern value=".yyyyMMdd" />

<maxSizeRollBackups value="20" />

<maximumFileSize value="2MB" />

<staticLogFileName value="true" />

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%-10.10p %d %10.10rms %-30.30c{1} %-40.40M - %m%n" />

</layout>

</appender>

<root>

<level value="INFO" />

<appender-ref ref="RollingLogFileAppender" />

</root>

</log4net>

………………………………………………….

………………………………………………….

Debe crearse la carpeta D:\Log\ABBCatalogo donde se almacenara el Log de las transacciones.

Editar la clase **Global.asax** del proyecto Cliente Web y completar el código que haga referencia al log. Quedará asi:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Optimization;

using System.Web.Routing;

namespace ABB.Catalogo.ClienteWeb

{

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

#region Log

log4net.Config.XmlConfigurator.Configure();

#endregion

}

}

}

1. Para crear el rastro en el log la clase **BaseLN** debe heredar de Controller:

public class BaseLN: Controller

El controlador **Usuario** debe heradar de **BaseLn**.(referenciar en controller **Usuario** el **using ABB.Catalogo.Utiles.Helpers**; **using ABB.Catalogo.Entidades.Base**; además **using System.Web.Mvc**;)

public class UsuarioController : BaseLN

………………………

1. Retornemos al controlador **Login** para completar la funcionalidad de la autenticación. Este controlador por ahora solo tiene un método que es el Index el cual expone la vista de autenticación. La vista de autenticación tiene un boton Validar de tipo sumit, esto significa que al pulsarse retorna al método Post index pasando las variables de la vista. Construiremos este metos index post.

En el controlador **Login** completar el método

[HttpPost]

public ActionResult Index(Usuario usuario)

este metodo se ejecuta cuando se pulsa el boton **Validar** buscara en la BD al usuario y si es válido abrirá el acceso al sistema.

El controlador Login quedara como:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using ABB.Catalogo.Entidades.Core;

using ABB.Catalogo.Utiles.Helpers;

using System.Web.Security;

using ABB.Catalogo.LogicaNegocio.Core;

using ABB.Catalogo.Entidades;

using ABB.Catalogo.Entidades.Base;

namespace ABB.Catalogo.ClienteWeb.Controllers

{

public class LoginController : BaseLN

{

// GET: Login

public ActionResult Index()

{

Usuario u = new Usuario();

return View(u);

}

[HttpPost]

public ActionResult Index(Usuario usuario)

{

if (string.IsNullOrEmpty(usuario.CodUsuario) || string.IsNullOrEmpty(usuario.ClaveTexto))

{

//Log.Info("Llego usuario: " + usuario.CodUsuario);

ModelState.AddModelError("\*", "Debe llenar el usuario o clave");

}

if (ModelState.IsValid)

{

try

{

usuario.Clave = EncriptacionHelper.EncriptarByte(usuario.ClaveTexto);

Usuario res = new UsuarioLN().BuscarUsuario(usuario);

if (res != null && res.IdUsuario > 0) //las credenciales son correctas

{

//Llenar una cookie

FormsAuthentication.SetAuthCookie(res.CodUsuario, true);

//llenar una sesion

List<Opcion> lista = new OpcionLN().ListaOpciones();

//para separar los controles de las acciones en la tabla de Opciones.

ParsearAcciones(lista);

VariablesWeb.gOpciones = lista;

Log.Info("Llego las opciones. " + VariablesWeb.gOpciones.Count().ToString());

VariablesWeb.gUsuario = res;

return RedirectToAction("Index", "Home");

}

else

{

ModelState.AddModelError("\*", "Usuario / Clave no validos");

}

}

catch (Exception ex)

{

ModelState.AddModelError("\*", ex.Message);

}

}

return View(usuario);

}

[NonAction]

private void ParsearAcciones(List<Opcion> lista)

{

int cantidad = 0;

foreach (Opcion item in lista)

{

if (!string.IsNullOrEmpty(item.UrlOpcion))

{

cantidad = item.UrlOpcion.Split('/').Count();

switch (cantidad)

{

case 3:

item.Area = item.UrlOpcion.Split('/')[0];

item.Controladora = item.UrlOpcion.Split('/')[1];

item.Accion = item.UrlOpcion.Split('/')[2];

break;

case 2:

item.Controladora = item.UrlOpcion.Split('/')[0];

item.Accion = item.UrlOpcion.Split('/')[1];

break;

case 1:

item.Controladora = item.UrlOpcion.Split('/')[0];

item.Accion = "Index";

break;

default:

break;

}

}

}

}

public ActionResult CerrarSesion()

{

FormsAuthentication.SignOut();

return RedirectToAction("Index", "Home");

}

}

}

En la capa de Negocios en **UsuariosLN** (que ya existe) crear el método para buscar por clave y usuario. Este método se llama: **BuscarUsuario(usuario)**

En la clase UsuariosLN(): se crea el metodo

……………………………………………………………..

public Usuarios BuscarUsuario(Usuarios Usuario)

{

try

{

return new UsuariosDA().BuscarUsuario(Usuario);

}

catch (Exception ex)

{

Log.Error(ex);

throw;

}

}

…………………………………………………………

En la capa de **Acceso a Datos** en la clase **UsuariosDA()** se debe crear el método **BuscarUsuario(usuario)**

**Clase UsuariosDA():**

**……………………………………………………………………….**

public Usuarios BuscarUsuario(Usuarios usuario)

{

Usuarios SegSSOMUsuario = null;

using (SqlConnection conexion = new SqlConnection(ConfigurationManager.ConnectionStrings[ConfigurationManager.AppSettings["cnnSql"]].ConnectionString))

{

using (SqlCommand comando = new SqlCommand("paUsuario\_BuscaCodUserClave", conexion))

{

comando.CommandType = System.Data.CommandType.StoredProcedure;

comando.Parameters.AddWithValue("@Clave", usuario.Clave);

comando.Parameters.AddWithValue("@CodUsuario", usuario.CodUsuario);

conexion.Open();

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

SegSSOMUsuario = LlenarEntidad(reader);

}

conexion.Close();

}

}

return SegSSOMUsuario;

}

………………………………………………………………………………..

Estos métodos creados reciben como parámetro un objeto tipo Usuarios.

En la capa de negocio crear la clase **OpcionLN** y actualizar la clase **UsuarioLN**.

En la capa **Accesodatos** crear la clase **OpcionDA** y actualizar la clase **UsuarioDA**, estas clases contendrán los métodos para acceder a la Base de datos y extraer la información que luego se presentará en las vistas.

Clase OpcionLN:

using ABB.Catalogo.Entidades.Base;

using ABB.Catalogo.Entidades.Core;

using ABB.Catalogo.AccesoDatos.Core;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ABB.Catalogo.LogicaNegocio.Core

{

public class OpcionLN : BaseLN

{

public List<Opcion> ListaOpciones()

{

try

{

return new OpcionDA().ListaOpciones();

}

catch (Exception ex)

{

Log.Error(ex);

throw;

}

}

}

}

**Clase OpcionDA (crearlo en la capa de accesos a datos)**

using ABB.Catalogo.Entidades.Core;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ABB.Catalogo.AccesoDatos.Core

{

public class OpcionDA

{

public Opcion LlenarEntidad(IDataReader reader)

{

Opcion opcion = new Opcion();

reader.GetSchemaTable().DefaultView.RowFilter = "ColumnName='IdOpcion'";

if (reader.GetSchemaTable().DefaultView.Count.Equals(1))

{

if (!Convert.IsDBNull(reader["IdOpcion"]))

opcion.IdOpcion = Convert.ToInt32(reader["IdOpcion"]);

}

reader.GetSchemaTable().DefaultView.RowFilter = "ColumnName='NombreOpcion'";

if (reader.GetSchemaTable().DefaultView.Count.Equals(1))

{

if (!Convert.IsDBNull(reader["NombreOpcion"]))

opcion.NombreOpcion = Convert.ToString(reader["NombreOpcion"]);

}

reader.GetSchemaTable().DefaultView.RowFilter = "ColumnName='UrlOpcion'";

if (reader.GetSchemaTable().DefaultView.Count.Equals(1))

{

if (!Convert.IsDBNull(reader["UrlOpcion"]))

opcion.UrlOpcion = Convert.ToString(reader["UrlOpcion"]);

}

reader.GetSchemaTable().DefaultView.RowFilter = "ColumnName='RutaImagen'";

if (reader.GetSchemaTable().DefaultView.Count.Equals(1))

{

if (!Convert.IsDBNull(reader["RutaImagen"]))

opcion.RutaImagen = Convert.ToString(reader["RutaImagen"]);

}

reader.GetSchemaTable().DefaultView.RowFilter = "ColumnName='NroOrden'";

if (reader.GetSchemaTable().DefaultView.Count.Equals(1))

{

if (!Convert.IsDBNull(reader["NroOrden"]))

opcion.NroOrden = Convert.ToInt32(reader["NroOrden"]);

}

reader.GetSchemaTable().DefaultView.RowFilter = "ColumnName='IdOpcionRef'";

if (reader.GetSchemaTable().DefaultView.Count.Equals(1))

{

if (!Convert.IsDBNull(reader["IdOpcionRef"]))

opcion.IdOpcionRef = Convert.ToInt32(reader["IdOpcionRef"]);

}

return opcion;

}

public List<Opcion> ListaOpciones()

{

List<Opcion> listaEntidad = new List<Opcion>();

Opcion entidad = null;

using (SqlConnection conexion = new SqlConnection(ConfigurationManager.ConnectionStrings[ConfigurationManager.AppSettings["cnnSql"]].ConnectionString))

{

using (SqlCommand comando = new SqlCommand("paOpcionLista", conexion))

{

comando.CommandType = System.Data.CommandType.StoredProcedure;

conexion.Open();

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

entidad = LlenarEntidad(reader);

listaEntidad.Add(entidad);

}

}

conexion.Close();

}

return listaEntidad;

}

}

}

**Actualización de Clase UsuarioLN**:

public Usuario BuscarUsuario(Usuario Usuario)

{

try

{

return new UsuarioDA().BuscarUsuario(Usuario);

}

catch (Exception ex)

{

Log.Error(ex);

throw;

}

}

**Actualización de clase UsuarioDA**

public Usuario BuscarUsuario(Usuario usuario)

{

Usuario SegSSOMUsuario = null;

using (SqlConnection conexion = new SqlConnection(ConfigurationManager.ConnectionStrings[ConfigurationManager.AppSettings["cnnSql"]].ConnectionString))

{

using (SqlCommand comando = new SqlCommand("paUsuario\_BuscaCodUserClave", conexion))

{

comando.CommandType = System.Data.CommandType.StoredProcedure;

comando.Parameters.AddWithValue("@Clave", usuario.Clave);

comando.Parameters.AddWithValue("@CodUsuario", usuario.CodUsuario);

conexion.Open();

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

SegSSOMUsuario = LlenarEntidad(reader);

}

conexion.Close();

}

}

return SegSSOMUsuario;

}

1. Probar :usuario jchavez Pss: 123456
2. **Trabajo con filter**: para la seguridad manejo de páginas de error. Para que no salga pantalla amarilla: crear en ABB.Catalogo.Entidades una **carpeta** llamada **Filter** dentro crear una clase llamada **ExceptionFilterAtributes** que quedara como:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using log4net;

using System.Web.Mvc;

namespace ABB.Catalogo.Entidades.Filter

{

public class ExceptionFilterAtributes : FilterAttribute, IExceptionFilter

{

public ILog Log

{

get

{

return LogManager.GetLogger

(System.Reflection.MethodBase.GetCurrentMethod()

.GetType());

}

}

public void OnException(ExceptionContext filterContext)

{

filterContext.HttpContext.Response.StatusCode = 500;

Log.Error(filterContext.Exception);

}

}

}

1. Esta clase controla los errores y mantiene un log de los mismos
2. Actualizar en el **web.config** sección **Sistem.Web** el control de errores. Poniendo un tag de control de errores y cuando se produzca error 500 , 404 re direccionara la página de control de errores.

<customErrors mode="On">

<error statusCode="500" redirect="Home/Error500"/>

<error statusCode="404" redirect="Home/PaginaNoEncontrada"/>

</customErrors>

1. En el controlador **Home** crear los métodos **Error500** y **PaginaNoencontrada**, desde allí crear las vistas vacías con **layout** general.

……………………………………….

public ActionResult PaginaPrueba()

{

return View();

}

public ActionResult Error500()

{

return View();

}

public ActionResult PaginaNoEncontrada()

{

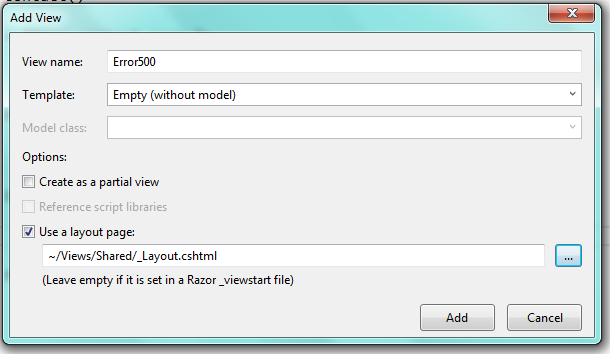
return View();

}

……………………………………………………….

…………………………………………………………..

Para crear la vista desde View() del método Error500(). Dar clik derecho y crear la vista:



Hacer lo mismo para vista de **PaginaNoEncontrada()**.

1. Para que el control de excepciones sea general se debe abrir la carpeta **App\_Start** editar la clase **FilterConfig** y allí poner nuestra clase de control creada:

using System.Web;

using System.Web.Mvc;

using ABB.Catalogo.Entidades.Filter;

namespace ABB.Catalogo.Cliente.Web

{

public class FilterConfig

{

public static void RegisterGlobalFilters(GlobalFilterCollection filters)

{

//filters.Add(new HandleErrorAttribute());

filters.Add(new ExceptionFilterAtributes());// esto hace que el filtro de excepciones funcione en todaslas clases

}

}

}

1. **Probar :usuario jchavez Pss: 123456**