Guía de Laboratorio 2.3

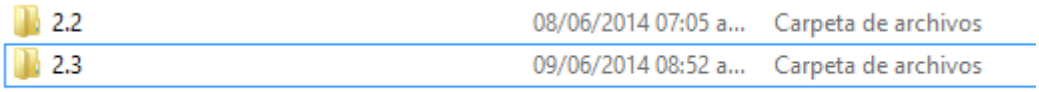
Profesor: Ing. Angel Ramos – Cibertec DAT

Indicaciones:
Tiempo: 90 minutos

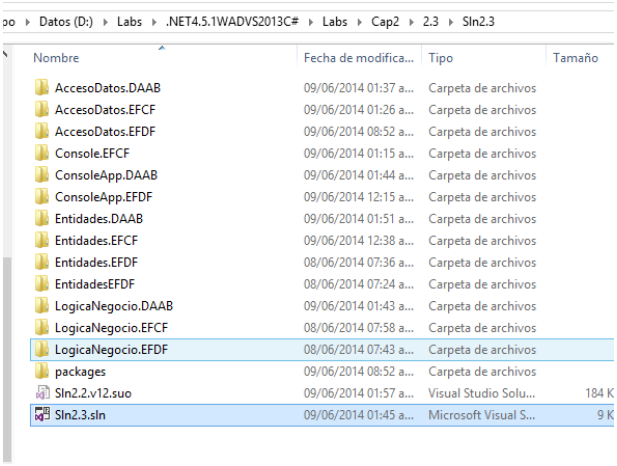
Crear el ProductController usando lo implementado en el Laboratorio 2.2.

Configurando la solución 2.3

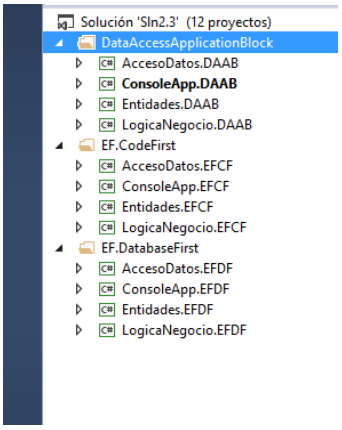
1. Para iniciar el laboratorio, es recomendable copiar la solución anterior a una nueva carpeta indicando que es parte del laboratorio 2.3



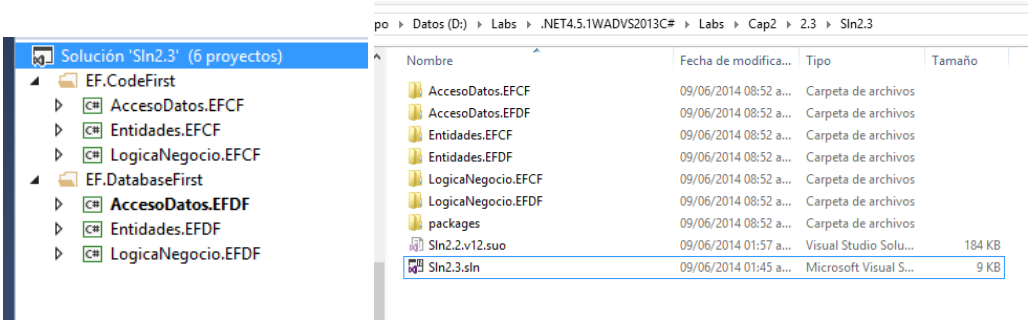
2. Y se actualiza el nombre de la carpeta y nombre de la solución.



3. Abril la solución en Visual Studio 2013

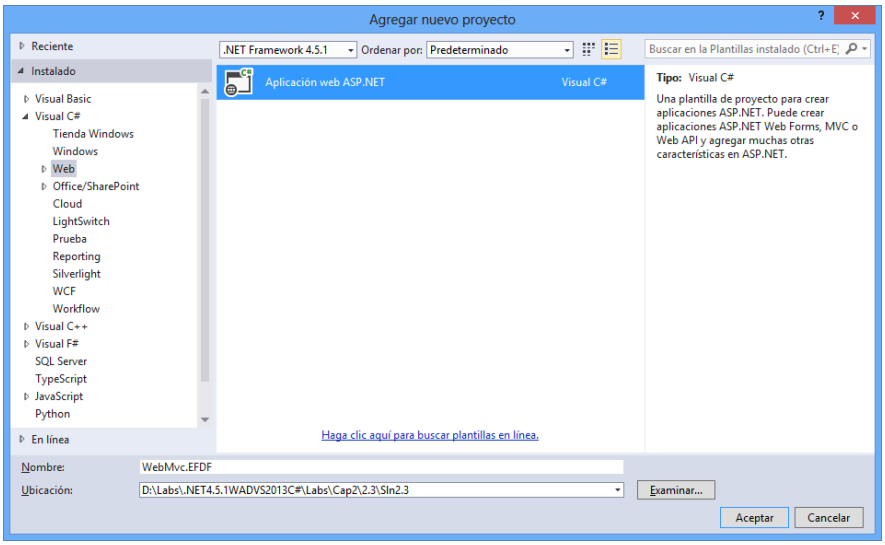


4. Como alcance para el laboratorio no consideraremos lo relacionado a **DataAccessApplicationBlock** y tampoco será necesario los proyectos **ConsoleApp** de **EF Database First** y **EF Code First**, por favor quitarlos de la solución, y luego eliminarlos físicamente de la carpeta de la solución 2.3.

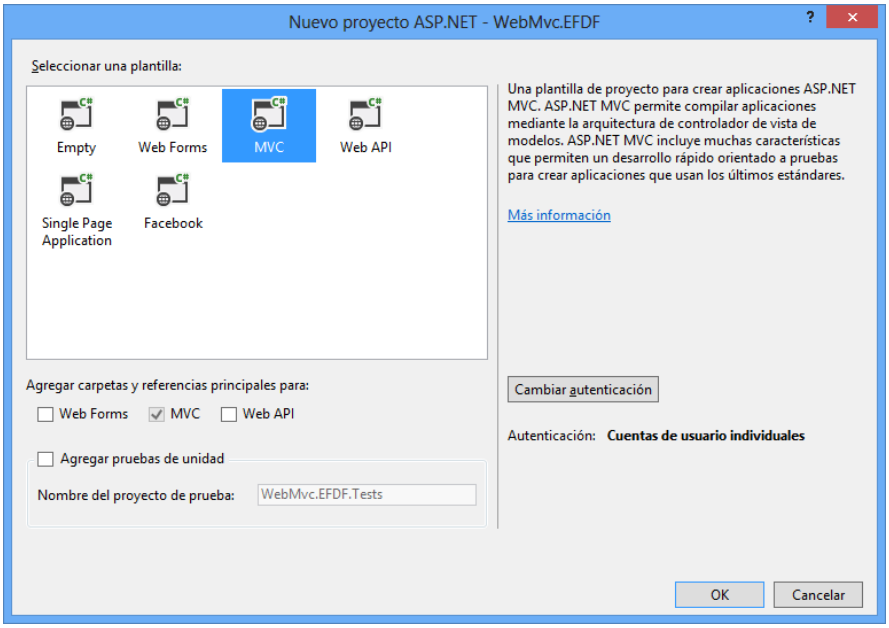

Solución 'Sln2.3' (6 proyectos)
EF.CodeFirst
AccesoDatos.EFCF
Entidades.EFCF
LogicaNegocio.EFCF
EF.DatabaseFirst
AccesoDatos.EFDF
Entidades.EFDF
LogicaNegocio.EFDF
po ▸ Datos (D:) ▸ Labs ▸ .NET4.5.1WADVS2013C# ▸ Labs ▸ Cap2 ▸ 2.3 ▸ Sln2.3
Nombre
Fecha de modifica...
Tipo
Tamaño
AccesoDatos.EFCF 09/06/2014 08:52 a... Carpeta de archivos
AccesoDatos.EFDF 09/06/2014 08:52 a... Carpeta de archivos
Entidades.EFCF 09/06/2014 08:52 a... Carpeta de archivos
Entidades.EFDF 09/06/2014 08:52 a... Carpeta de archivos
LogicaNegocio.EFCF 09/06/2014 08:52 a... Carpeta de archivos
LogicaNegocio.EFDF 09/06/2014 08:52 a... Carpeta de archivos
packages 09/06/2014 08:52 a... Carpeta de archivos
Sln2.2.v12.suo 09/06/2014 01:57 a... Visual Studio Solu... 184 KB
Sln2.3.sln 09/06/2014 01:45 a... Microsoft Visual S... 9 KB

Implementando ProductController para EF Database First

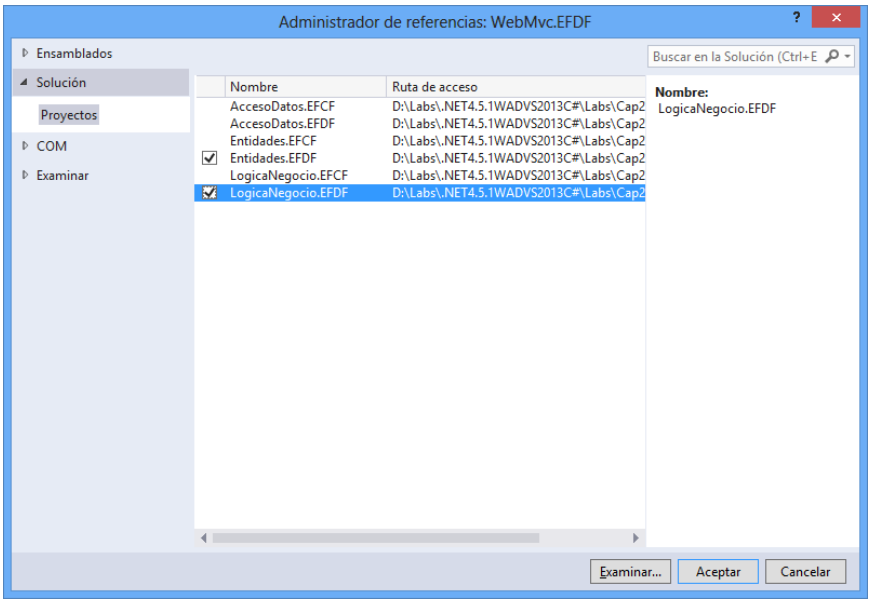
1. Agregar un nuevo proyecto web ASP.NET con nombre **WebMvc.EFDF** dentro de la carpeta **EF.DatabaseFirst** de la solución



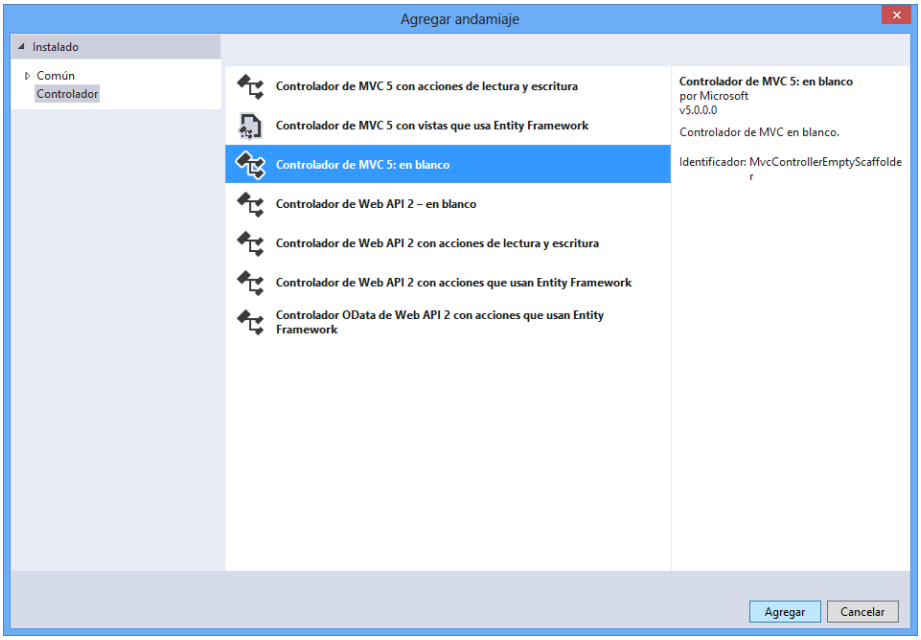
2. En la pantalla de configuración del proyecto, seleccionar MVC

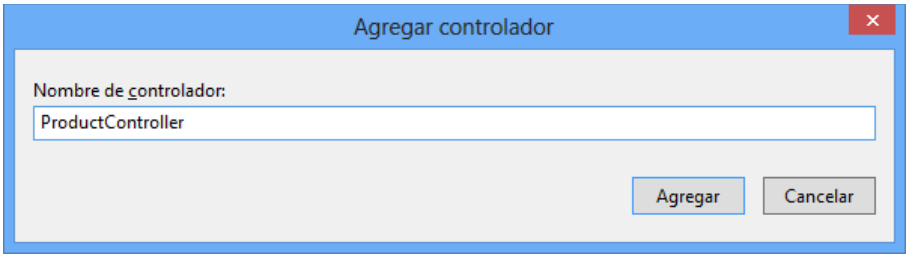


3. Agregar las referencias de los proyectos **Entidades.EFDF** y **LogicaNegocios.EFDF** al proyecto creado.



4. Crear un nuevo controlador del tipo **"Controlador de MVC 5: en blanco"** con el nombre **ProductController** en la carpeta **Controllers**.





5. Es importante conocer los conceptos de los métodos HTTP, puede consultar el siguiente link http://www.w3schools.com/tags/ref_httpmethods.asp. Dentro de la clase **ProductController** creada, deberíamos agregar el siguiente código:

5.1. **public ActionResult Index()**: (HTTP GET) Sera el método de acción principal del controlador, devolverá la lista de los productos . Cuando el usuario ingrese a nuestra web con el */Product* direccionara a este método de acción.

5.2. **public ActionResult Details(int id)**: (HTTP GET) Retornará el detalle de un producto. Si el usuario solicita ver el detalle del producto con Id 5 podrá acceder en la siguiente ruta */Product/Details/5*

5.3. **public ActionResult Create():** (HTTP GET) Retornará la página para ingresar los datos del producto. El usuario podrá acceder en la siguiente ruta */Product/Create*.

5.4. **public ActionResult Create(Product product):** (HTTP POST) Como es un método de acción del tipo POST se encargará de enviar una acción, es decir enviará los datos del producto a ser creado. Luego que el producto es creado redirecciona a la pagina */Product/Index*.

5.5. **public ActionResult Edit(int id):** (HTTP GET) Retornará la página de detalle del producto en modo edición, el usuario podrá acceder en la siguiente ruta */Product/Edit/5*.

5.6. **public ActionResult Edit(Product product):** (HTTP POST) Como es un método de acción del tipo POST se encargará de enviar una acción, es decir enviara los datos del producto a ser actualizados. Luego que el producto es actualizado redirecciona a la pagina */Product/Index.*

5.7. **public ActionResult Delete(int id):** (HTTP GET) Retornará la página con la información del producto a ser eliminado, el usuario podrá acceder en la siguiente ruta */Product/Delete/5*.

5.8. **public ActionResult Delete(int id, FormCollection collection):** (HTTP POST) Como es un método de acción del tipo POST se encargará de enviar una acción, es decir enviara los datos necesarios para eliminar el producto. Luego que el producto es eliminado redirecciona a la pagina */Product/Index.*

Así es como debe quedar la clase ProductController.cs:

```
using Entidades.EFDF;
using LogicaNegocio.EFDF;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebMvc.EFDF.Controllers
{
    public class ProductController : Controller
    {
        //
        // GET: /Product/
        public ActionResult Index()
        {
            var prodLN = new ProductLN();
            var productos = prodLN.SelectAll();
            return View(productos);
        }

        //
        // GET: /Product/Details/5
        public ActionResult Details(int id)
        {
            var prodLN = new ProductLN();
            var producto = prodLN.Select(new Product { ProductID = id });
            return View(producto);
        }

        //
        // GET: /Product/Create
        public ActionResult Create()
```

```
        {
            return View();
        }

        //
        // POST: /Product/Create
        [HttpPost]
        public ActionResult Create(Product producto)
        {
            try
            {
                var prodLN = new ProductLN();
                prodLN.Insert(producto);
                return RedirectToAction("Index");
            }
            catch (Exception)
            {
                return View();
            }
        }

        //
        // GET: /Product/Edit/5
        public ActionResult Edit(int id)
        {
            var prodLN = new ProductLN();
            var producto = prodLN.Select(new Product { ProductID = id });
            return View(producto);
        }

        //
        // POST: /Product/Edit/5
        [HttpPost]
        public ActionResult Edit(Product producto)
        {
            try
            {
                var prodLN = new ProductLN();
                prodLN.Update(producto);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Product/Delete/5
        public ActionResult Delete(int id)
        {
            var prodLN = new ProductLN();
            var producto = prodLN.Select(new Product { ProductID = id });
            return View(producto);
        }

        //
        // POST: /Product/Delete/5
        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                var prodLN = new ProductLN();
                prodLN.Delete(new Product { ProductID = id });
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

6. Para ejecutar la aplicación primero debemos realizar 2 modificaciones en el archivo **Web.config** del proyecto **WebMvc.EFDF**:
   6.1. Modificar la cadena de conexión DefaultConnection para que direccione a nuestra base de datos **AdventureWorks2012**.

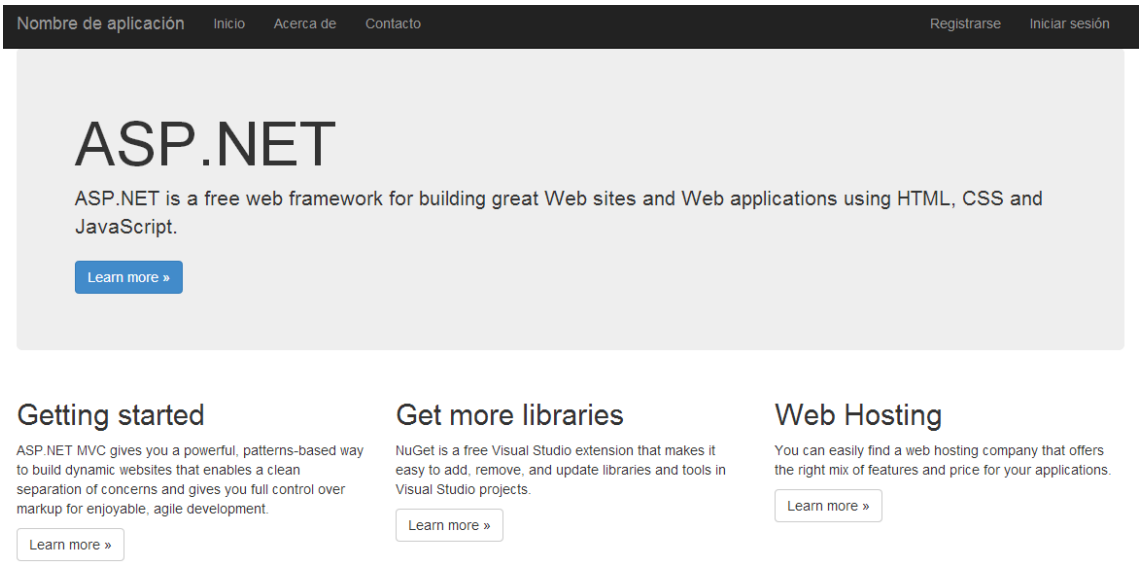
```
<add name="DefaultConnection"
         connectionString="data source=.\SQLEXPRESS2012;initial
catalog=AdventureWorks2012;Integrated Security=True;"
          providerName="System.Data.SqlClient" />
```

   Como el nuevo **ASP.NET Identity Provider** usa **EntityFramework-Code First** con el fin de hacer más configurable el manejo de la seguridad de la aplicación. Pueden leer más información en: http://stackoverflow.com/questions/23130795/how-does-a-new-asp-net-mvc-5-application-know-how-to-create-a-database-and-how-d

   6.2. Agregar la cadena de conexión de nuestra base de datos siguiendo el formato de Entity Framework, copiamos esta cadena de conexión desde el archivo **App.config** del proyecto **AccesoDatos.EFDF**.

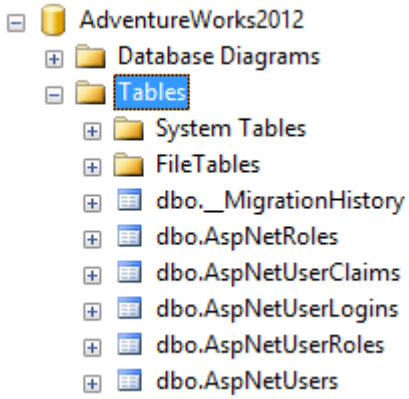
```
<add name="AdventureWorksContext"
connectionString="metadata=res://*/ModelAdventureWorks.csdl|res://*/ModelA
dventureWorks.ssdl|res://*/ModelAdventureWorks.msl;provider=System.Data.Sq
lClient;provider connection string=&amp;quot;data
source=ARAMOS\SQLEXPRESS2012;initial catalog=AdventureWorks2012;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework&amp;quot;"
          providerName="System.Data.EntityClient" />
```

7. Configurar el proyecto **WebMvc.EFDF** como el proyecto de inicio de la solución y ejecutar la aplicación.



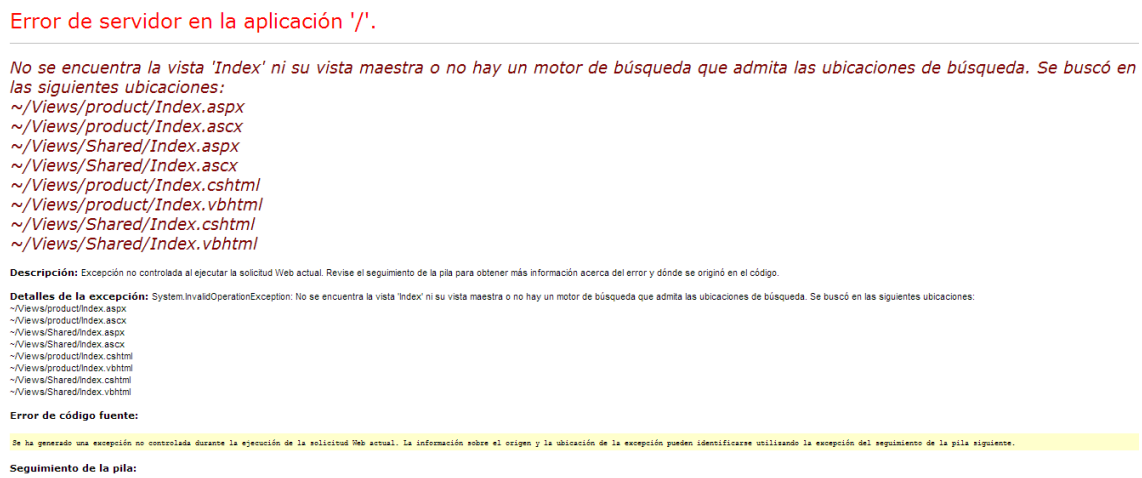
8. Si usted consulta su base de datos AdventureWorks2012 en su servidor SQL Server, no verá ninguna tabla relacionado a la seguridad por defecto manejada por ASP.NET. Solo después de crear su usuario es que se crearan las tablas (de esta manera funciona Code First).
9. Clic en **Registrarse**.
10. Ingrese su nombre de usuario y contraseña. Clic en **Registrarse**.

11. Al terminar el proceso de registro usted podrá observar que las tablas de AspNet relacionadas a la seguridad de su aplicación han sido creadas en su base de datos **AdventureWorks2012**.



12. Si usted escribe en la barra de direcciones del explorador web **<su site>/Product**

13. Visualizará un error indicando que la vista **Product/Index.cshtml** no existe. Las vistas se crearan en el laboratorio 2.4.



Error de servidor en la aplicación '/'.

No se encuentra la vista 'Index' ni su vista maestra o no hay un motor de búsqueda que admita las ubicaciones de búsqueda. Se buscó en las siguientes ubicaciones:
~/Views/product/Index.aspx
~/Views/product/Index.ascx
~/Views/Shared/Index.aspx
~/Views/Shared/Index.ascx
~/Views/product/Index.cshtml
~/Views/product/Index.vbhtml
~/Views/Shared/Index.cshtml
~/Views/Shared/Index.vbhtml

Descripción: Excepción no controlada al ejecutar la solicitud Web actual. Revise el seguimiento de la pila para obtener más información acerca del error y dónde se originó en el código.

Detalles de la excepción: System.InvalidOperationException: No se encuentra la vista 'Index' ni su vista maestra o no hay un motor de búsqueda que admita las ubicaciones de búsqueda. Se buscó en las siguientes ubicaciones:
~/Views/product/Index.aspx
~/Views/product/Index.ascx
~/Views/Shared/Index.aspx
~/Views/Shared/Index.ascx
~/Views/product/Index.cshtml
~/Views/product/Index.vbhtml
~/Views/Shared/Index.cshtml
~/Views/Shared/Index.vbhtml

Error de código fuente:

Se ha generado una excepción no controlada durante la ejecución de la solicitud Web actual. La información sobre el origen y la ubicación de la excepción pueden identificarse utilizando la excepción del seguimiento de la pila siguiente.

Seguimiento de la pila:

Implementando ProductController para EF Code First

1. Agregar un nuevo proyecto web ASP.NET con nombre **WebMvc.EFCF** dentro de la carpeta **EF.CodeFirst** de la solución con la misma configuración del ejemplo anterior pero tomando en cuenta que debe referenciar ahora a los proyectos **Entidades.EFCF** y **LogicaNegocio.EFCF**.
2. Repita los pasos 4 y 5 del ejemplo anterior

```
using Entidades.EFCF;
using LogicaNegocio.EFCF;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebMvc.EFCF.Controllers
{
    public class ProductController : Controller
    {
        //
        // GET: /Product/
        public ActionResult Index()
        {
            var prodLN = new ProductLN();
            var productos = prodLN.SelectAll();
            return View(productos);
        }

        //
        // GET: /Product/Details/5
        public ActionResult Details(int id)
        {
            var prodLN = new ProductLN();
            var producto = prodLN.Select(new Product { ProductID = id });
            return View(producto);
        }

        //
        // GET: /Product/Create
        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Product/Create
        [HttpPost]
        public ActionResult Create(Product producto)
        {
            try
            {
                var prodLN = new ProductLN();
                prodLN.Insert(producto);
                return RedirectToAction("Index");
            }
            catch (Exception)
            {
                return View();
            }
        }

        //
        // GET: /Product/Edit/5
        public ActionResult Edit(int id)
        {
            var prodLN = new ProductLN();
            var producto = prodLN.Select(new Product { ProductID = id });
            return View(producto);
        }
```

```
        //
        // POST: /Product/Edit/5
        [HttpPost]
        public ActionResult Edit(Product producto)
        {
            try
            {
                var prodLN = new ProductLN();
                prodLN.Update(producto);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Product/Delete/5
        public ActionResult Delete(int id)
        {
            var prodLN = new ProductLN();
            var producto = prodLN.Select(new Product { ProductID = id });
            return View(producto);
        }

        //
        // POST: /Product/Delete/5
        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                var prodLN = new ProductLN();
                prodLN.Delete(new Product { ProductID = id });
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

3. Para ejecutar la aplicación (igual que en el ejemplo anterior) debemos realizar unos cambios en las cadenas de conexión en el archivo **Web.config** del proyecto **WebMvc.EFCF** a la base de datos a usar:
   3.1. Modificar la cadena de conexión DefaultConnection para que direccione a nuestra base de datos **AdventureWorks2012**.

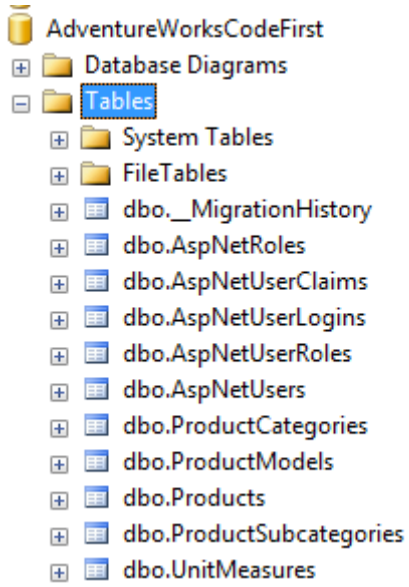
```
<add name="DefaultConnection"
         connectionString="data source=.\SQLEXPRESS2012;initial
catalog=AdventureWorksCodeFirst;Integrated Security=True;"
          providerName="System.Data.SqlClient" />
```

   3.2. Agregar la cadena de conexión de nuestra base de datos siguiendo el formato de Entity Framework, copiamos esta cadena de conexión desde el archivo **App.config** del proyecto **AccesoDatos.EFCF**.

```
<add name="AdventureWorksContext"
         connectionString="data source=.\SQLEXPRESS;initial
catalog=AdventureWorksCodeFirst;Integrated Security=True;"
                  providerName="System.Data.SqlClient" />
```

4. Establecer el proyecto **WebMvc.EFCF** como el proyecto de inicio de la solución ye ejecutar.

5. Si aparece el link de **Cerrar Sesión**, por favor de clic. Luego clic en el link **Registrese**. Ingrese su nombre de usuario y contraseña. Podrá visualizar en su servidor de base de datos que las tablas de seguridad han sido creadas.



6. Realice los pasos 12 y 13 del ejemplo anterior y obtendrá los mismos resultados, debemos crear las vistas.