CONSTRUYENDO UNA APLICACIÓN ASP NET MVC

El presente documento tiene como objetivo guiarlos en la implementación de una aplicación MVC con 3 capas, basadas en la base de Datos Northwind.

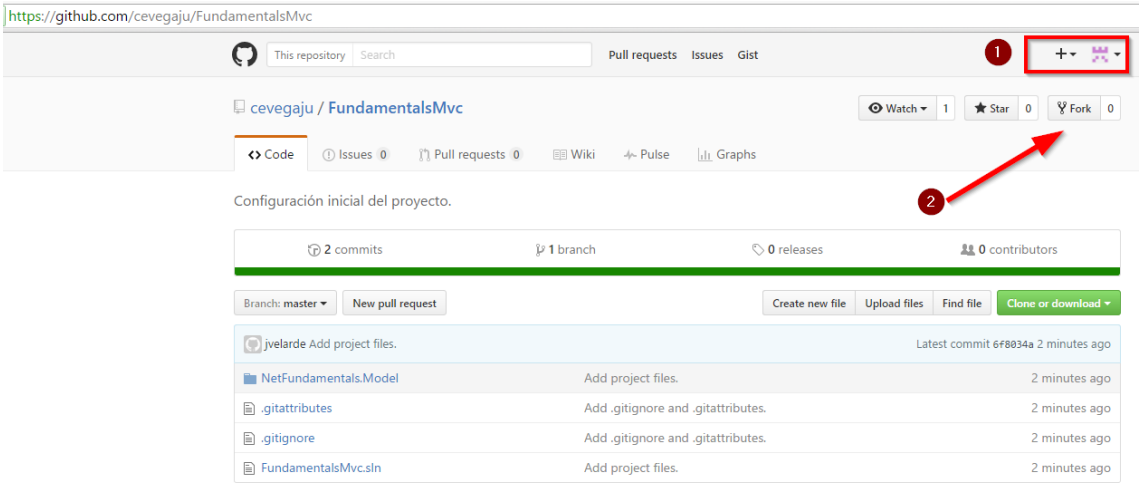
Para poder realizar este procedimiento se requiere tener instalado los siguientes softwares:

- Microsoft Visual Studio 2015 Community Edition ó superior.
- Microsoft SQL Server 2014 Express Edition ó superior.

1. CONFIGURACION DE REPOSITORIO DE GITHUB

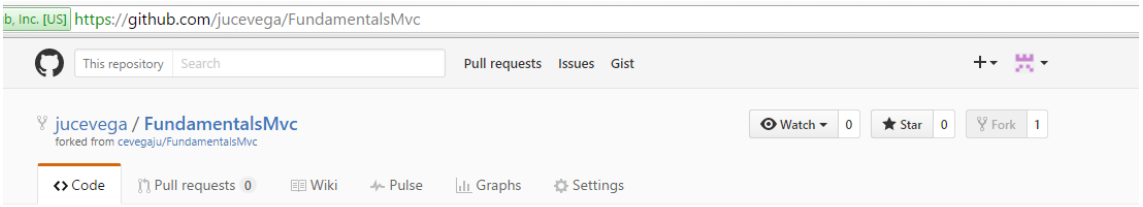
El siguiente paso es acceder nuestra cuenta de **github** y realizar un **fork** del repositorio ubicado en la siguiente ruta:

https://github.com/cevegaju/FundamentalsMvc

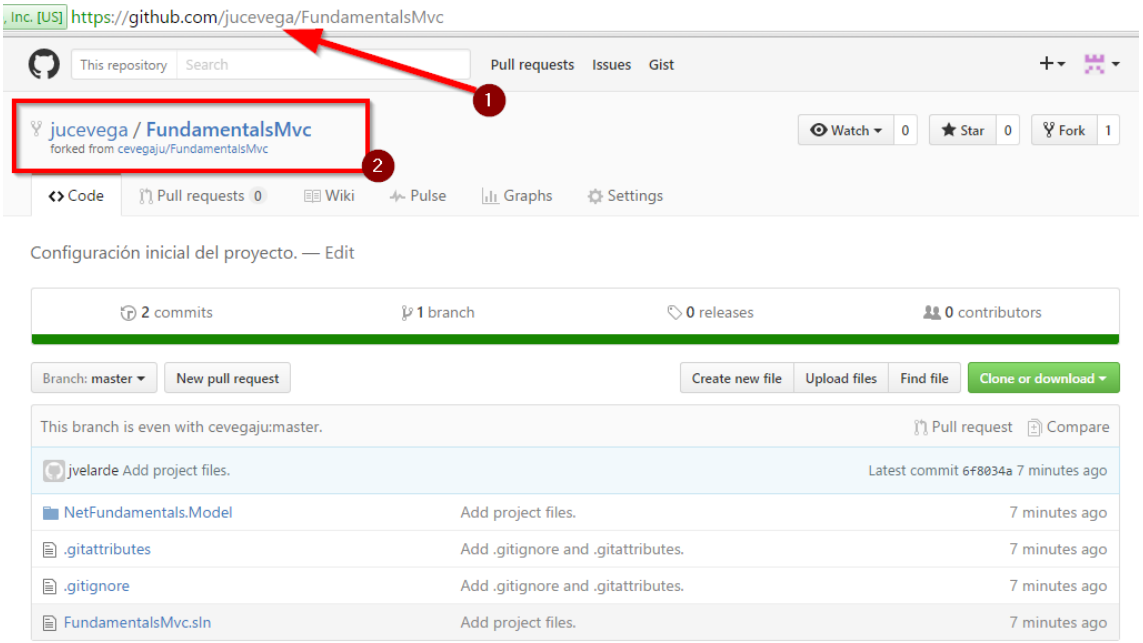


1. Este es el indicador que nos encontramos logueados con una cuenta en GitHub.
2. Ubicación del botón a hacer clic.

Luego de hacer clic en el botón fork visualizaremos la siguiente pantalla:



Y cuando el proceso finalice:



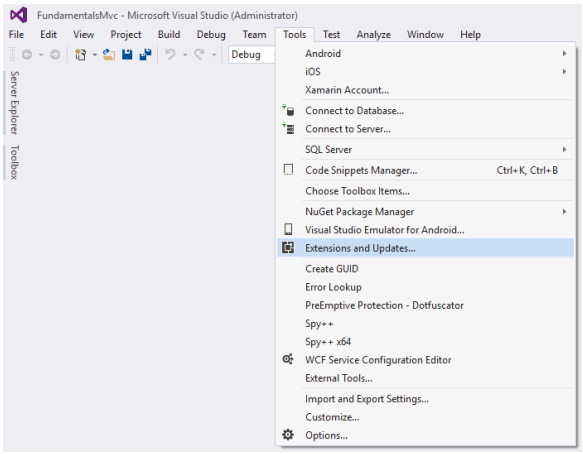
En la ruta veremos nuestra cuenta de usuario (1) y también notaremos que el nombre del repositorio tiene como raíz nuestro nombre de usuario de github(2).

2. CONFIGURACION DE VISUAL STUDIO

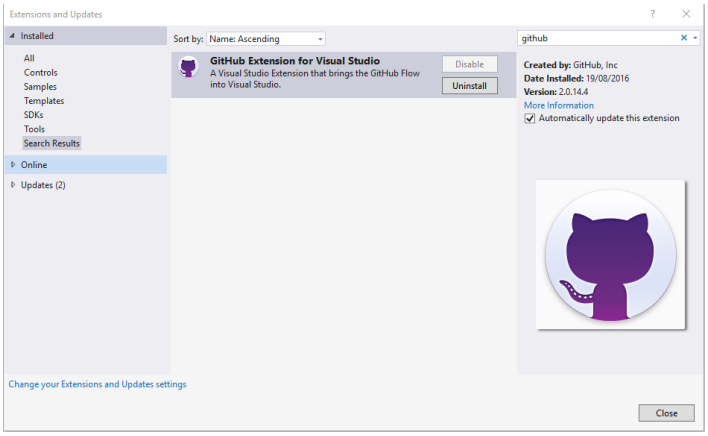
Ahora nos toca configurar nuestro Visual Studio para poder acceder a nuestro repositorio.

Para realizar este procedimiento validaremos que la extensión de GitHub se encuentre instalada, para dicho fin realizaremos la búsqueda en la siguiente ubicación del menú:

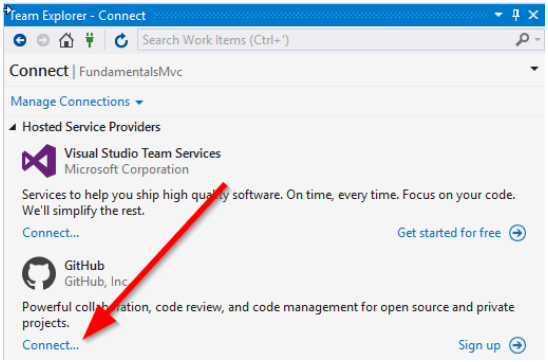
Tools > Extensions and Updates



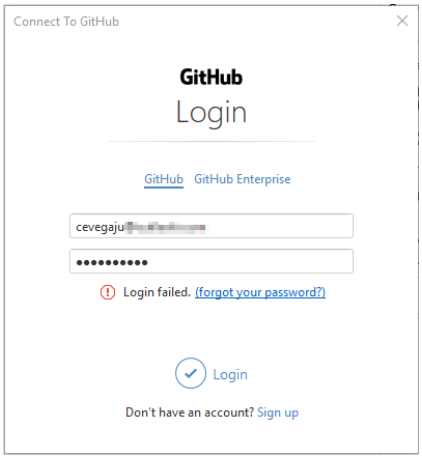
Y validamos que la extensión se encuentre instalada, como se muestra en la siguiente imagen:



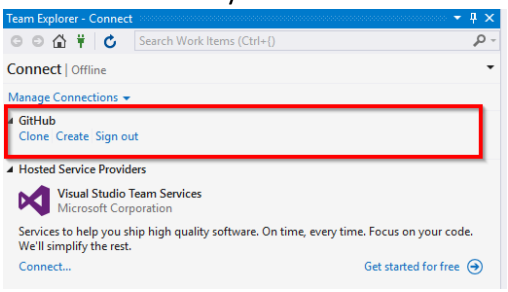
Ahora cargamos en el panel de **Team Explorer** (Menu View > Team Explorer).



Para conectar nuestra cuenta de GitHub con Visual Studio hacemos clic donde indica la flecha (**Connect**) y en la ventana siguiente ingresaremos nuestro usuario y contraseña de **GitHub.**

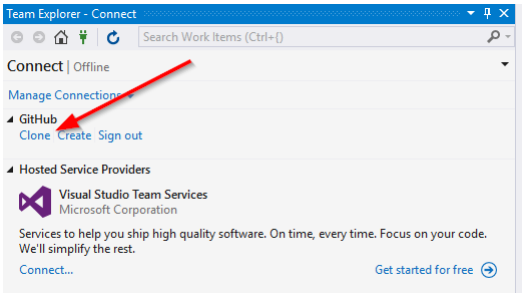


Cuando el usuario y contraseña es válido lograremos ver la siguiente ventana:

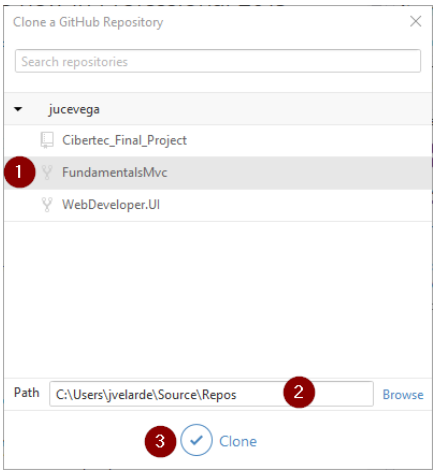


Como paso final clonaremos el repositorio del cual acabamos de hacer **Fork**.

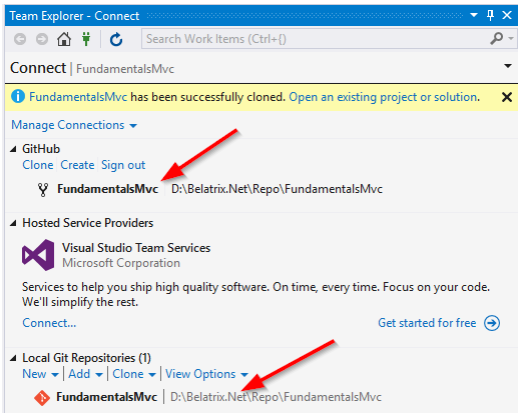
- Clic en Clone



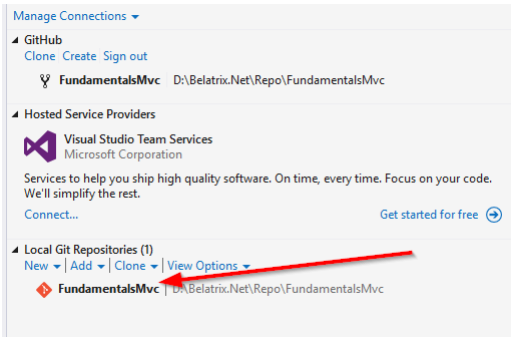
- Seleccionamos el repositorio **FundamentalsMvc, indicamos la ruta en donde guardar y hacemos clic en Clone.**



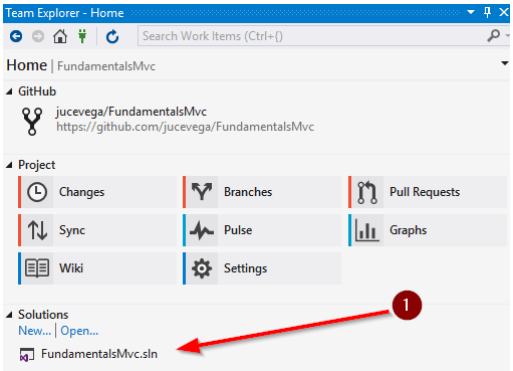
Y el resultado es:



- Ahora solo nos queda abrir la solución para poder iniciar a desarrollar nuestra aplicación, para ese fin hacemos clic en **FundamentalsMvc** que se encuentra bajo la sección **Local Git Repositories**.



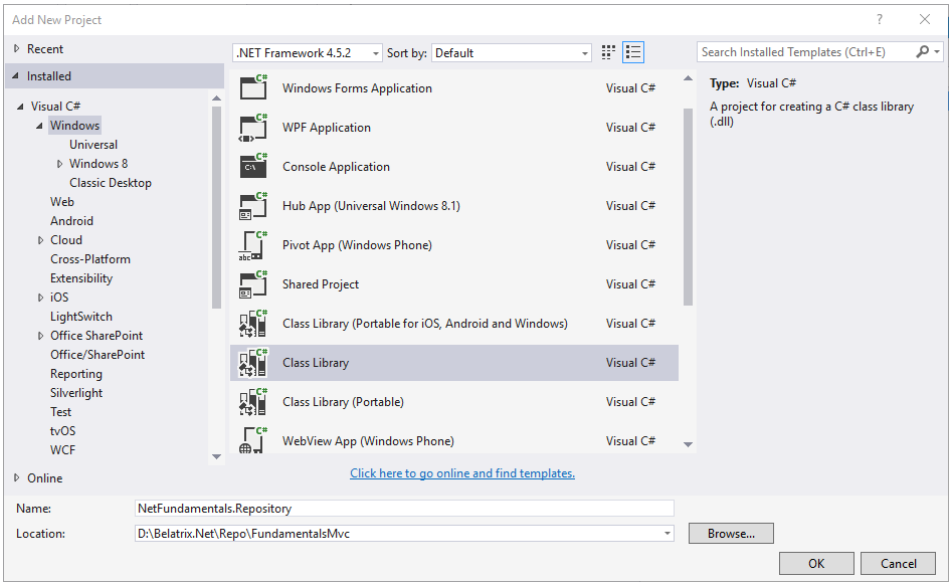
Seguido visualizaremos la siguiente pantalla, para cargar la solución hacer doble clic en elemento indicado en la imagen siguiente:



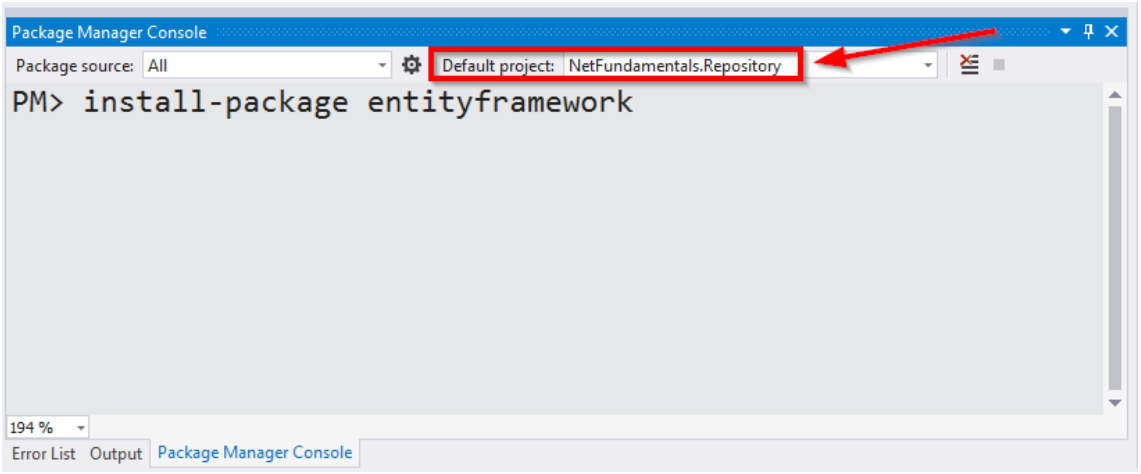
Y listo ahora ya podemos iniciar a desarrollar en nuestro repositorio.

3. CREANDO REPOSITORIO DE DATOS

En la solución ya cargada procedemos a crear un nuevo proyecto del tipo Class Library, con el nombre **NetFundamentals.Repository**



Procedemos a instalar el paquete NuGet de entity framework con el siguiente comando:
Install-Package EntityFramework
Desde la consola de administración de paquetes (Ver Imagen)

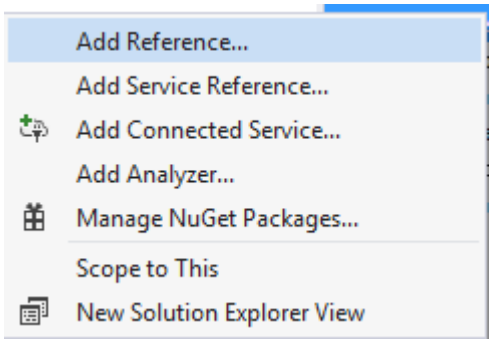


Tener en cuenta el **Default Project** que es proyecto en el cual se va a instalar este paquete.

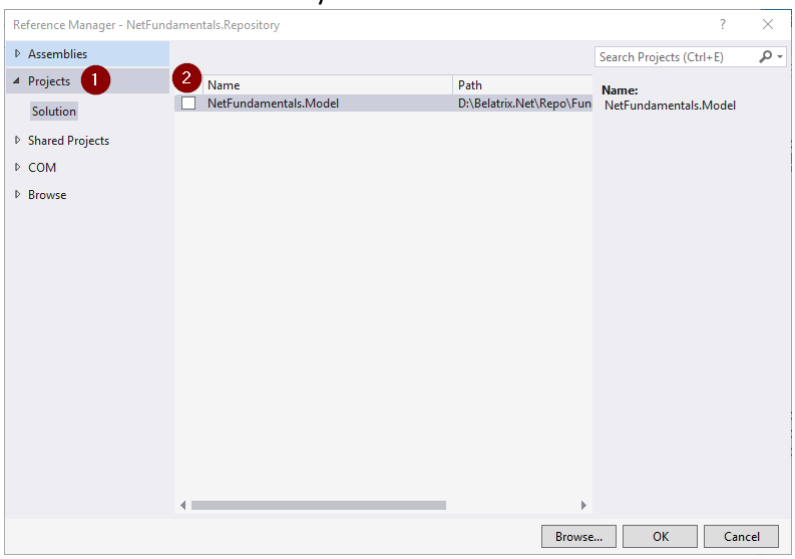
NetFundamentals.Model

CREACIÓN DEL REPOSITORIO

- Como primer paso debemos de agregar la referencia al proyecto **NetFundamentals.Model**.
  Clic derecho en **References** y luego clic en **Add Reference**



Luego seleccionar en el panel izquierdo **Projects> Solutions** y marcar el check **NetFundamentals.Model** y clic en **OK**.



- Paso seguido debemos de crear el archivo que administra el contexto de datos (**DbContext**) con el nombre **NorthwindContext** mismo en el que agregaremos el siguiente código:

```
using NetFundamentals.Model;
using System.Data.Entity;
using System.Data.Entity.ModelConfiguration.Conventions;

namespace NetFundamentals.Repository
{
    public class NorthwindContext : DbContext
    {
        public NorthwindContext(): base("NetFundamentals")
        {
            Database.SetInitializer<NorthwindContext>(null);
        }

        protected override void OnModelCreating(DbModelBuilder
modelBuilder)
        {

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
        }

        public DbSet<Customers> Customer { get; set; }
    }
}
```

- Procedemos con la creación de nuestro repositorio iniciamos con la creación de la interfaz **IRepository** y la clase que lo va a implementar **BaseRepository**.
  Archivo **IRepository.cs**

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

namespace NetFundamentals.Repository
{
    public interface IRepository<T>
    {
        int Add(T entity);
        int Update(T entity);
        int Delete(T entity);
        IEnumerable<T> GetList();
        T GetById(Expression<Func<T, bool>> match);
    }
}
```

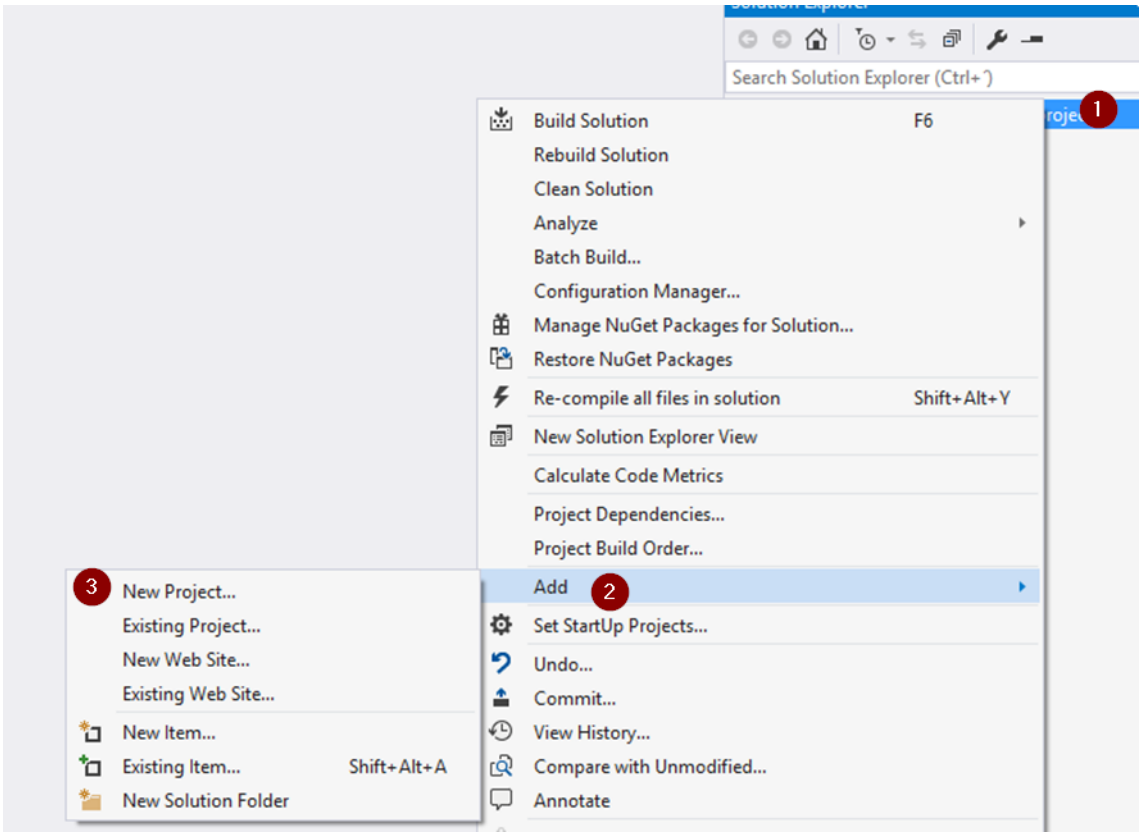
Archivo **BaseRepository.cs**

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Linq.Expressions;

namespace NetFundamentals.Repository
{
    public class BaseRepository<T> : IRepository<T> where T : class
    {
        private NorthwindContext dbContext;
        public BaseRepository()
        {
            dbContext = new NorthwindContext();
        }
        public int Add(T entity)
        {
            dbContext.Entry(entity).State = EntityState.Added;
            return dbContext.SaveChanges();
        }

        public int Delete(T entity)
        {
            dbContext.Entry(entity).State = EntityState.Deleted;
            return dbContext.SaveChanges();
        }
        public int Update(T entity)
        {
            dbContext.Entry(entity).State = EntityState.Modified;
            return dbContext.SaveChanges();
        }
        public IEnumerable<T> GetList()
        {
            return dbContext.Set<T>();
        }

        public T GetById(Expression<Func<T, bool>> match)
        {
            return dbContext.Set<T>().FirstOrDefault(match);
        }
    }
}
```

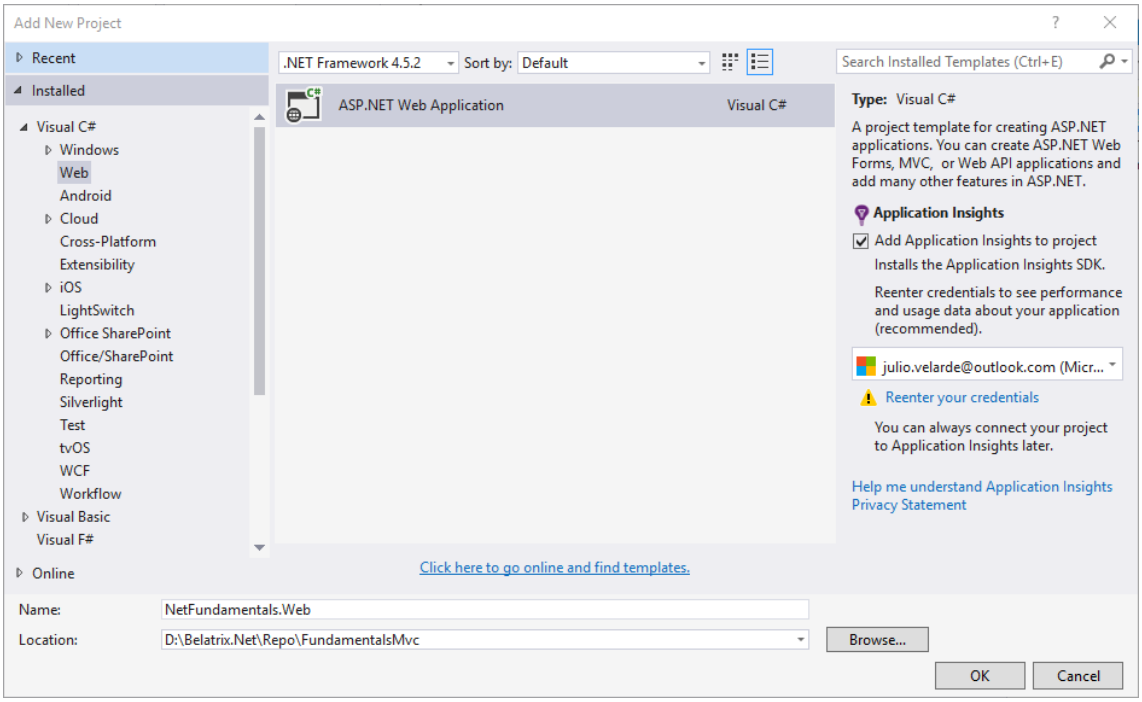
4. CREANDO LA APLICACIÓN ASP NET MVC

Proseguimos con la creación de nuestro proyecto web ASP NET MVC.

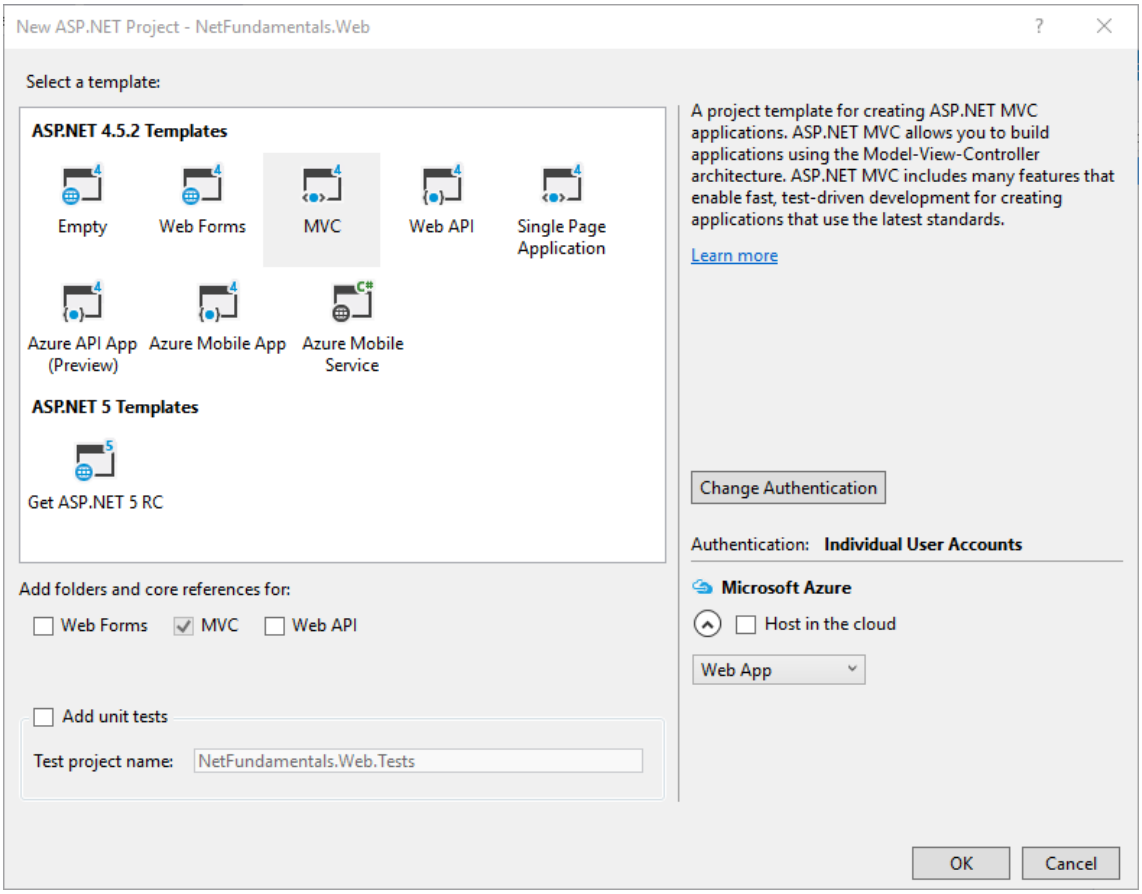
Agregamos el proyecto haciendo clic derecho en la solución (Observar imagen)



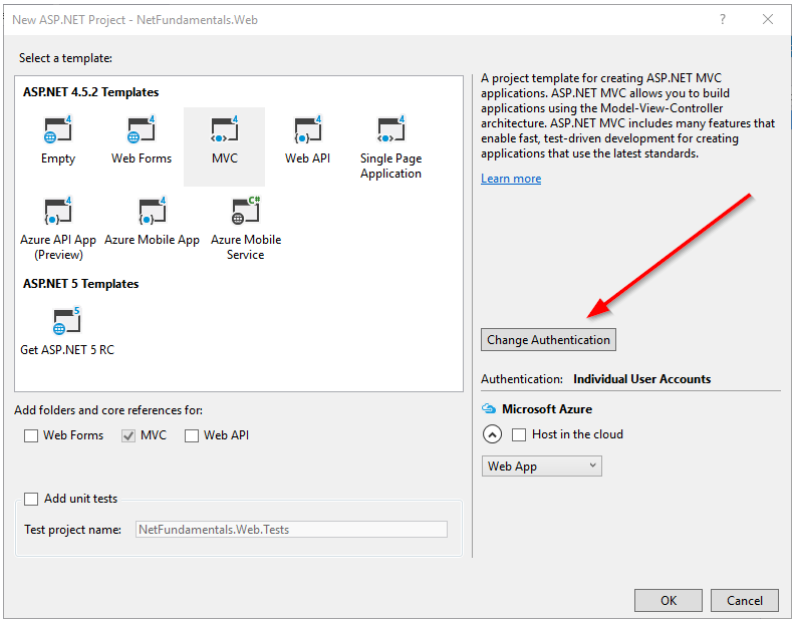
Seleccionamos en panel izquierdo las plantillas web, ingresamos con el nombre **NetFundamentals.Web** y hacemos clic en OK.

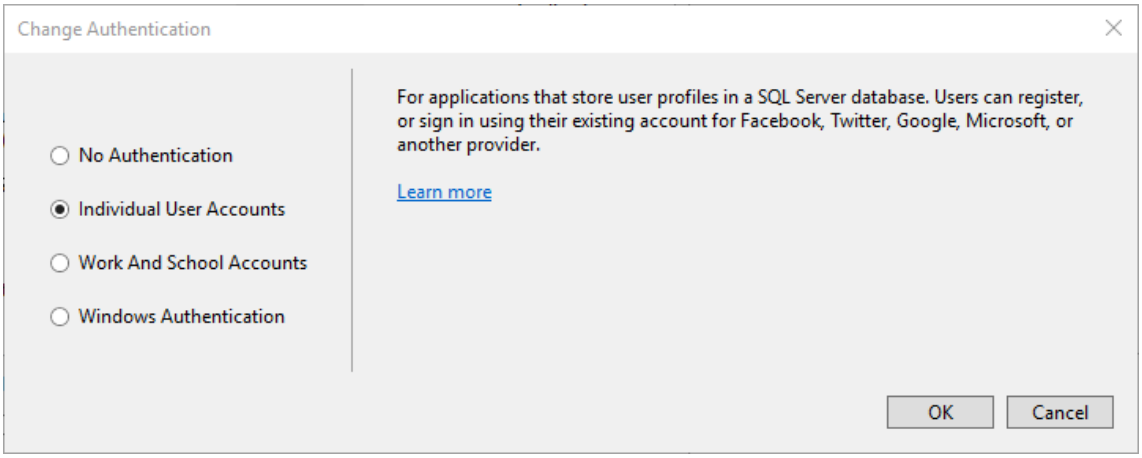


Seleccionamos la plantilla MVC

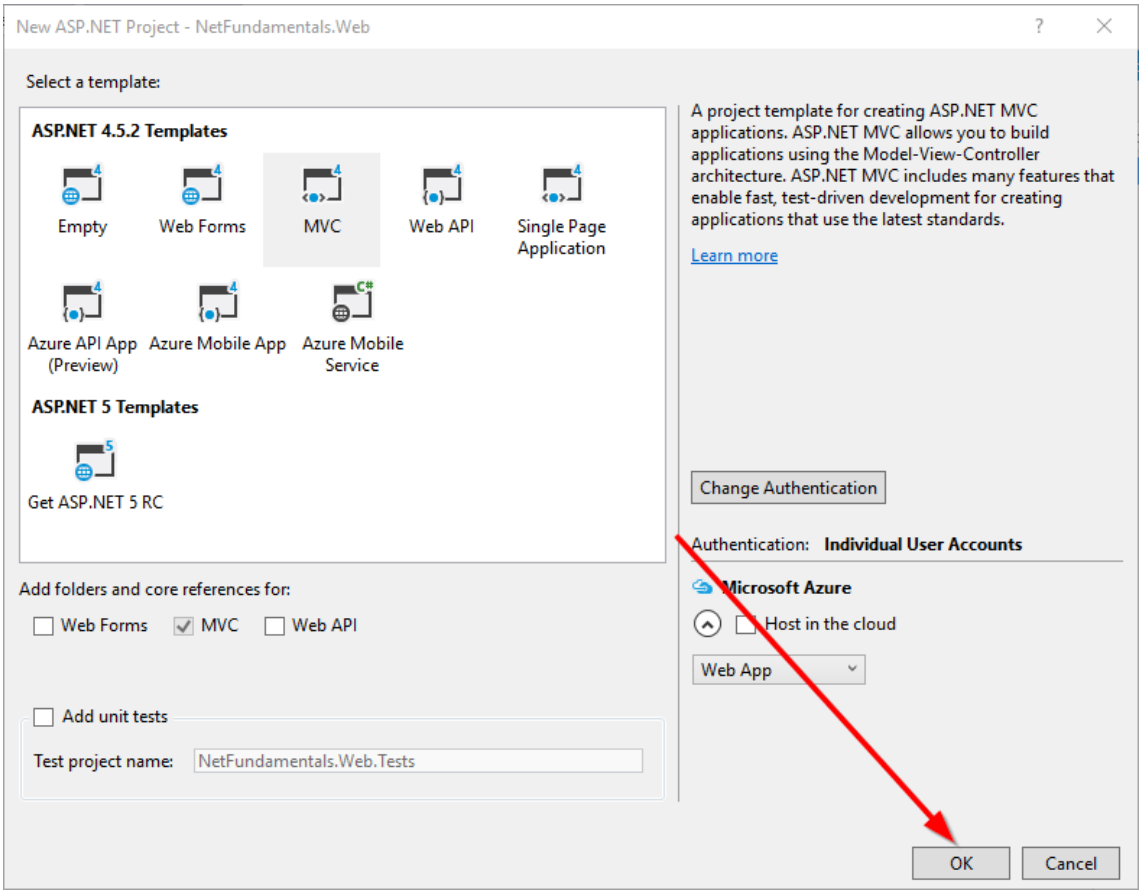


Seleccionaremos la autenticación del tipo: **Individual User Accounts,** haciendo clic en **Change Authentication**.

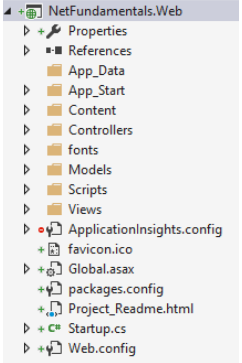




Y hacemos clic en OK en ambas ventanas.



Y dejamos que la plantilla cree los elementos de la aplicación web.

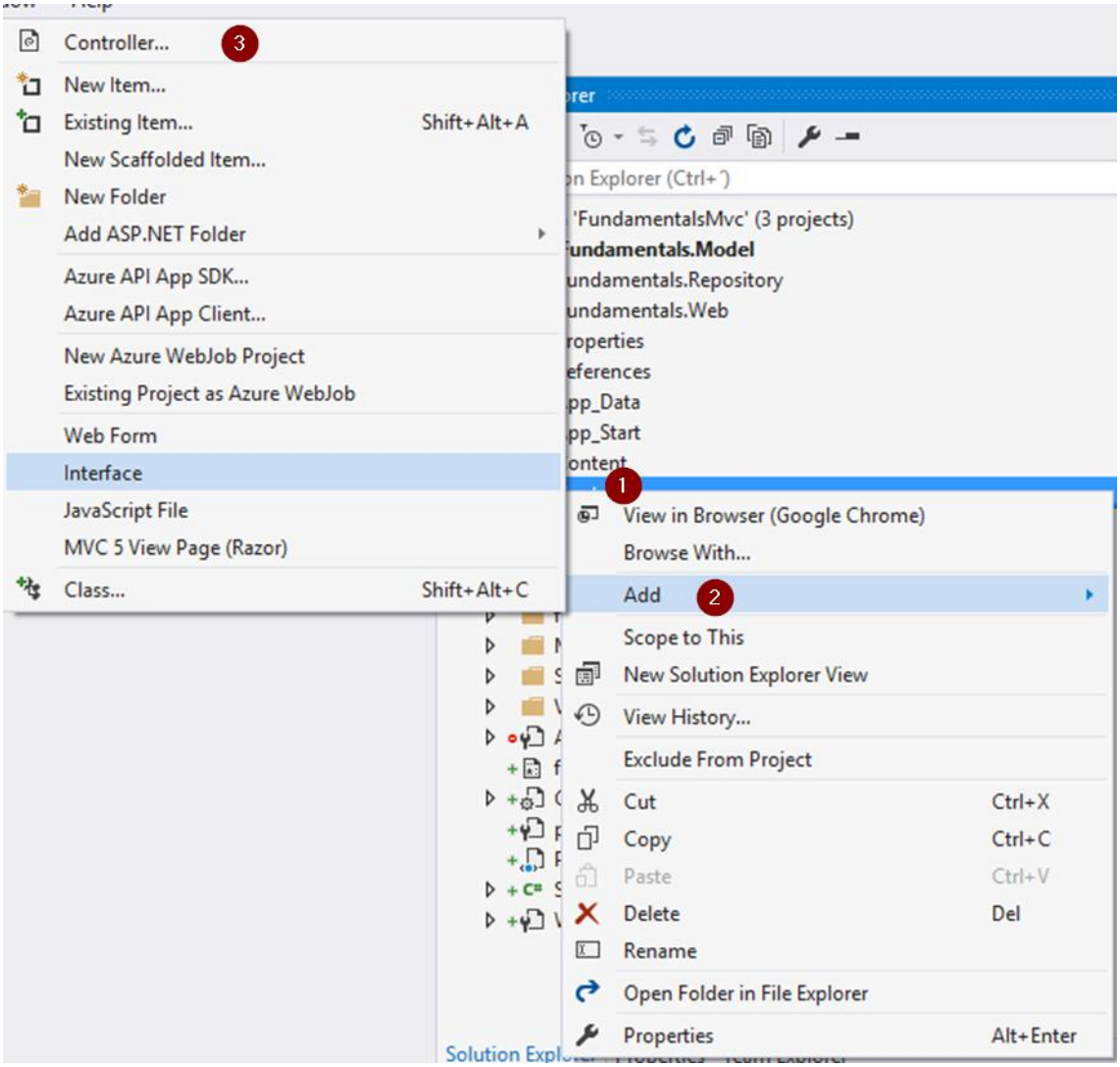


Para continuar necesitamos agregar las referencias de los proyectos:

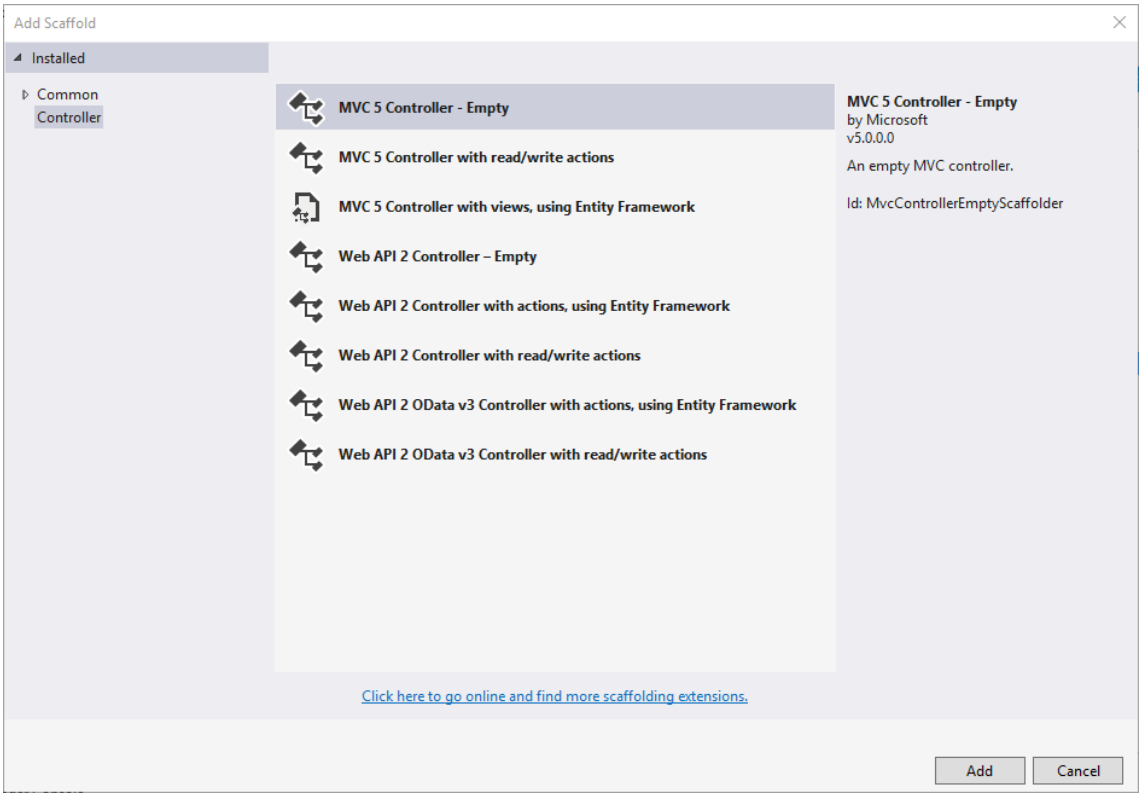
- NetFundamentals.Repository
- NetFundamentals.Model

Ya que los usaremos desde nuestra aplicación Web.

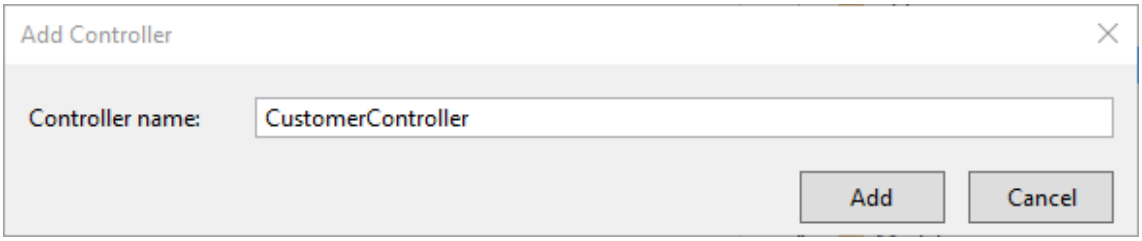
Procedemos con la creación de nuestro primer controlador llamado CustomerController, **1** Hacemos clic derecho en la carpeta controllers, **2** Posicionamos el cursor en Add y **3** hacemos clic en Controller.



Seleccionamos MVC 5 Controller – Empty



Cambiamos el nombre por default a CustomerController, es importante mantener la parte final del nombre como Controller y hacemos clic en Add.



Agregamos el siguiente código:

```
using NetFundamentals.Model;
using NetFundamentals.Repository;
using System.Web.Mvc;

namespace NetFundamentals.Web.Controllers
{
    public class CustomerController : Controller
    {
        private IRepository<Customer> repository;
        public CustomerController()
        {
            repository = new BaseRepository<Customer>();
        }

        public ActionResult Index()
        {
            return View(repository.GetList());
        }

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

        [HttpPost]
        public ActionResult Create(Customer customer)
        {
            if (!ModelState.IsValid) return View(customer);
            repository.Add(customer);
            return RedirectToAction("Index");
        }

        public ActionResult Edit(int id)
        {
            var customer = repository.GetById(x=> x.CustomerId==id);
            if (customer == null) return RedirectToAction("Index");
            return View(customer);
        }

        [HttpPost]
        public ActionResult Edit(Customer customer)
        {
            if (!ModelState.IsValid) return View(customer);
            repository.Update(customer);
            return RedirectToAction("Index");
        }

        public ActionResult Delete(int id)
        {
            var customer = repository.GetById(x => x.CustomerId == id);
            if (customer == null) return RedirectToAction("Index");
            return View(customer);
        }

        [HttpPost]
        public ActionResult Delete(Customer customer)
        {
            repository.Delete(customer);
            return RedirectToAction("Index");
        }

        public ActionResult Details(int id)
        {
            var customer = repository.GetById(x => x.CustomerId == id);
            if (customer == null) return RedirectToAction("Index");
            return View(customer);
        }
    }
}
```

Paso seguido creamos las vistas.

Y finalizamos agregando la cadena de conexión a nuestro web.config

```
<connectionStrings>
    <add name="NetFundamentals" connectionString="Data
Source=(local)\SQLExpress; Database=Chinook; Integrated
Security=True" providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Esta es la instancia de SQL que tengas configurado:

- **(local)\SQLExpress** para versiones Express de SQL Server.
- **(local)** para versiones diferentes a la Express.