**Entity Framework**

Contents

[**Code First Approach** 2](#_Toc490348646)

[**Code First to an Existing Database.** 9](#_Toc490348647)

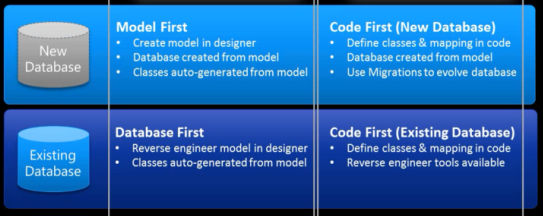
[**Code First using existing Stored Procedures:** 16](#_Toc490348648)

[**enable-migrations** 22](#_Toc490348649)

[**add-migration Initial** 22](#_Toc490348650)

[**update-database** 22](#_Toc490348651)

Para comenzar a trabajar con Entity Framework, podemos hacerlo mediante 3 enfoques diferentes para crear nuestro modelo conceptual.

****

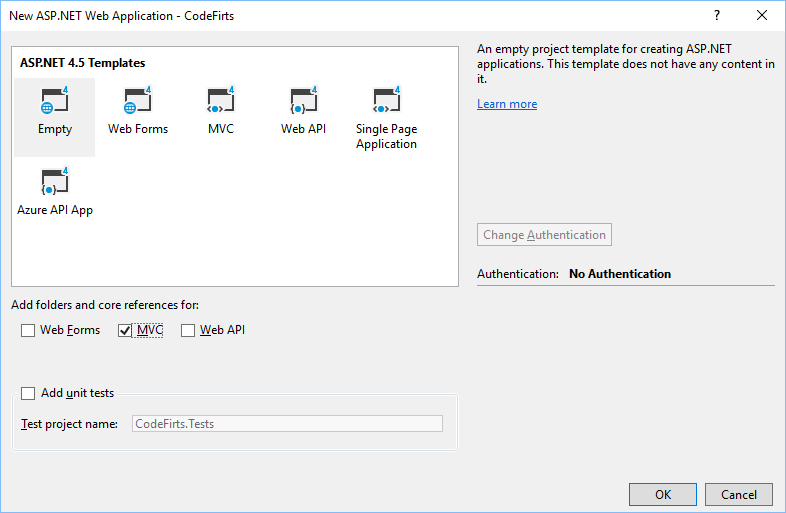
# **Code First Approach**

Code First es un enfoque más de Entity Framework. *Tú creas las clases POCO y creas las relaciones entre las mismas.*

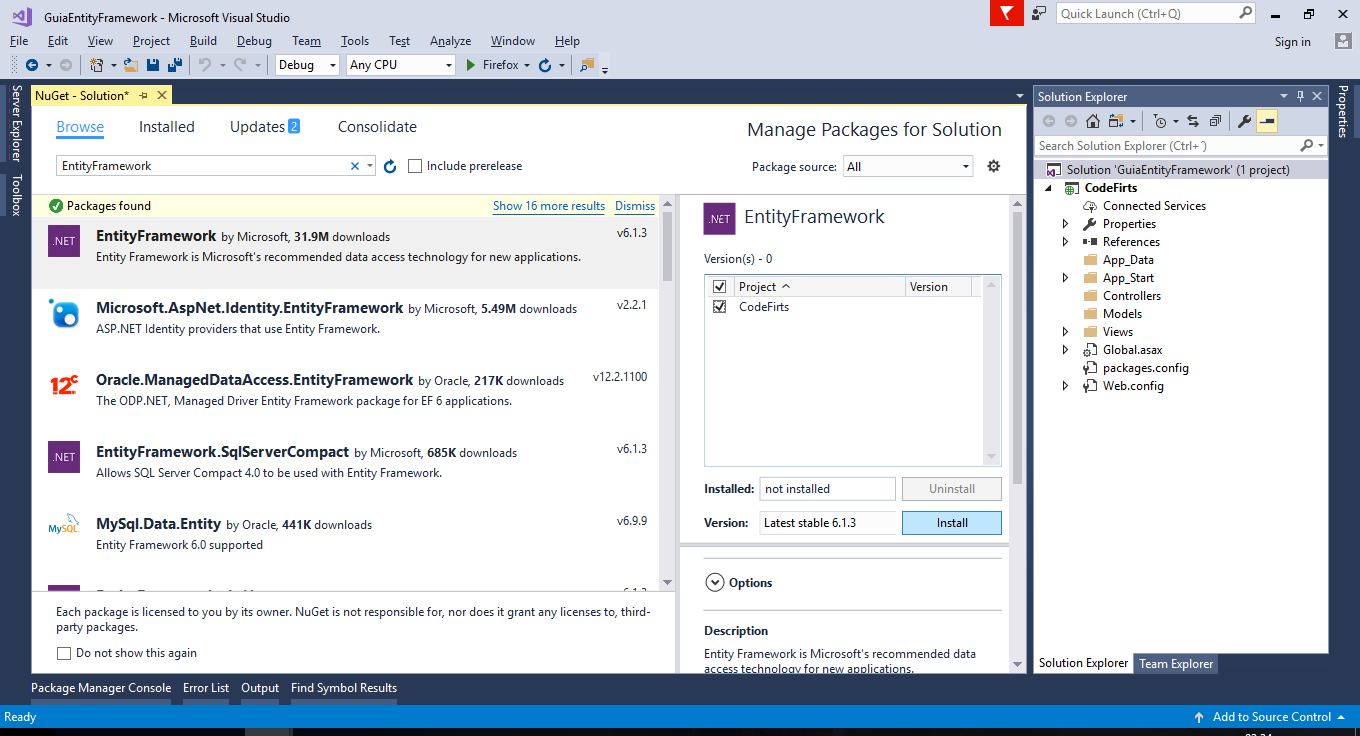
Lo importante es entender que con Code First, **lo primero es el código**. En vez de comenzar creando la base de datos y después con ingeniería inversa generar las clases POCO, con Code First primero creamos el modelo con código y después se genera automáticamente la base de datos.

**Pasos:**

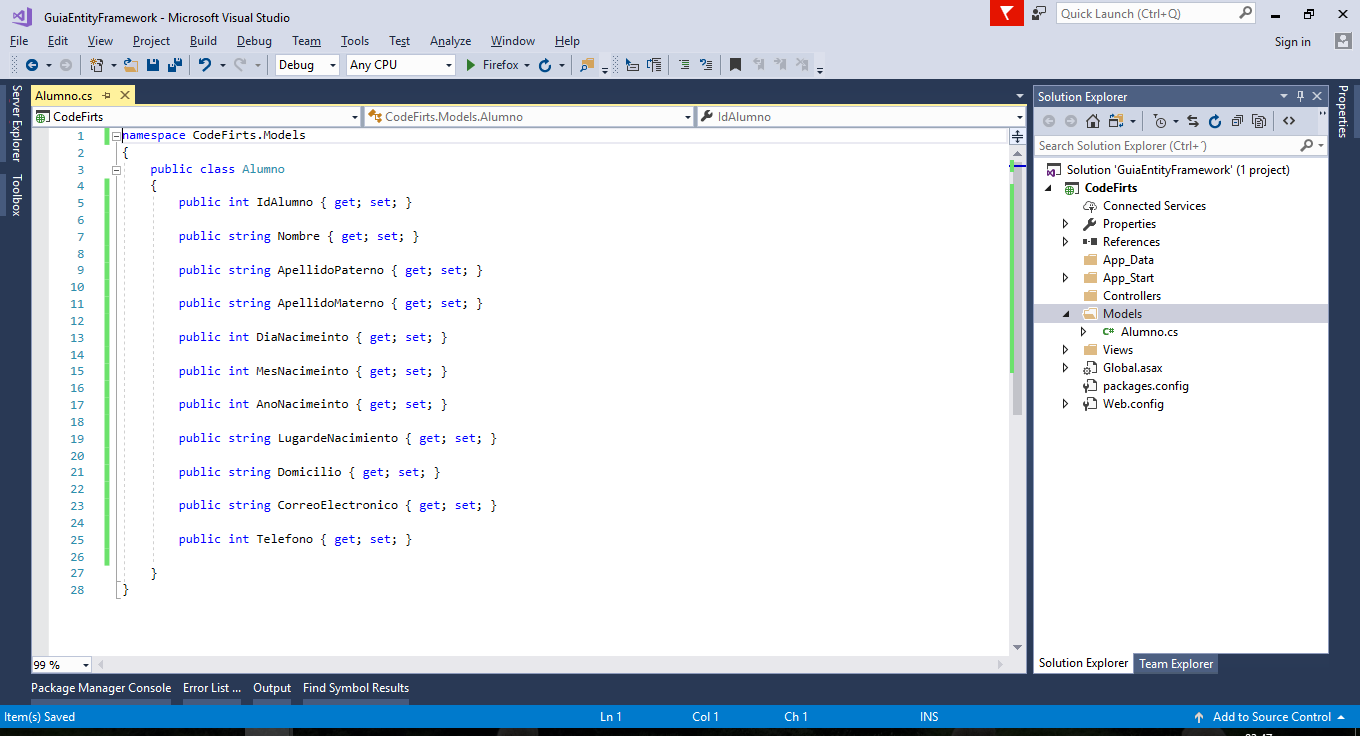
1. **Crear una app donde se trabajara (MVC).**



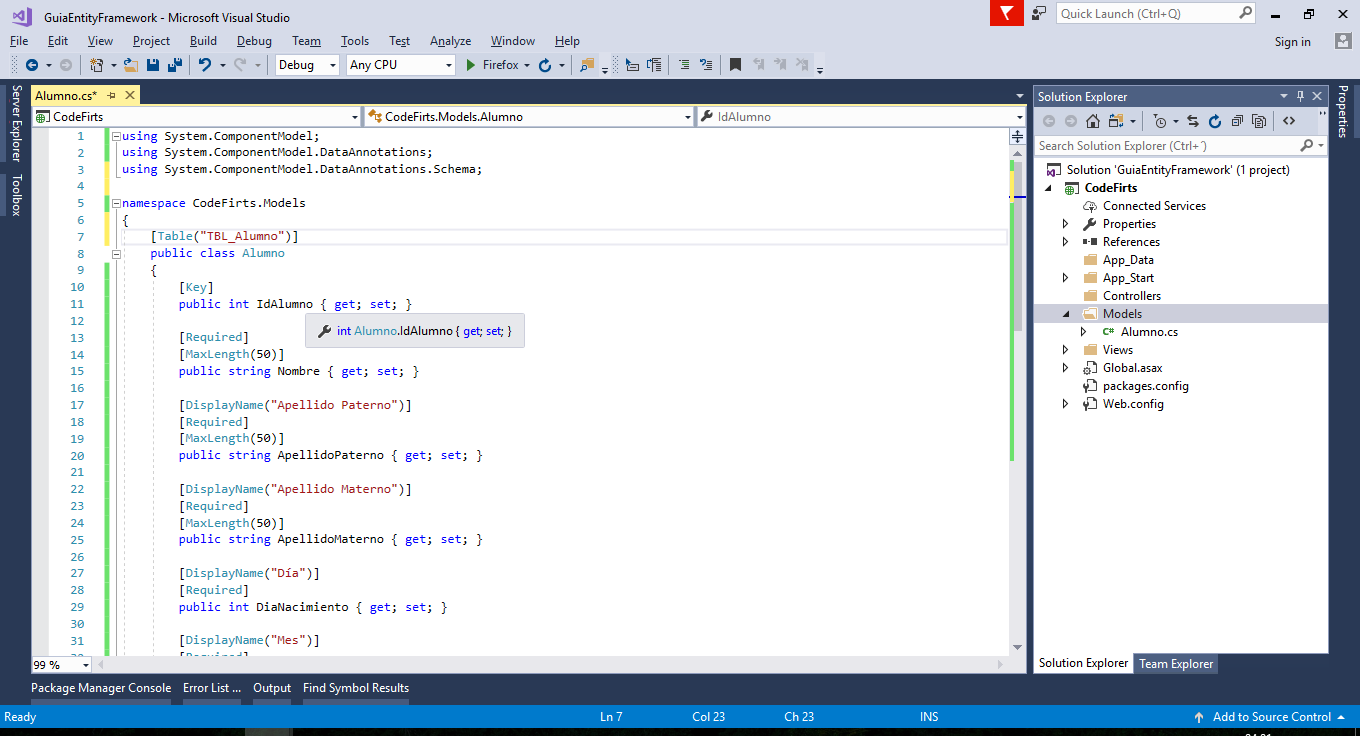
1. **Instalar EF.**



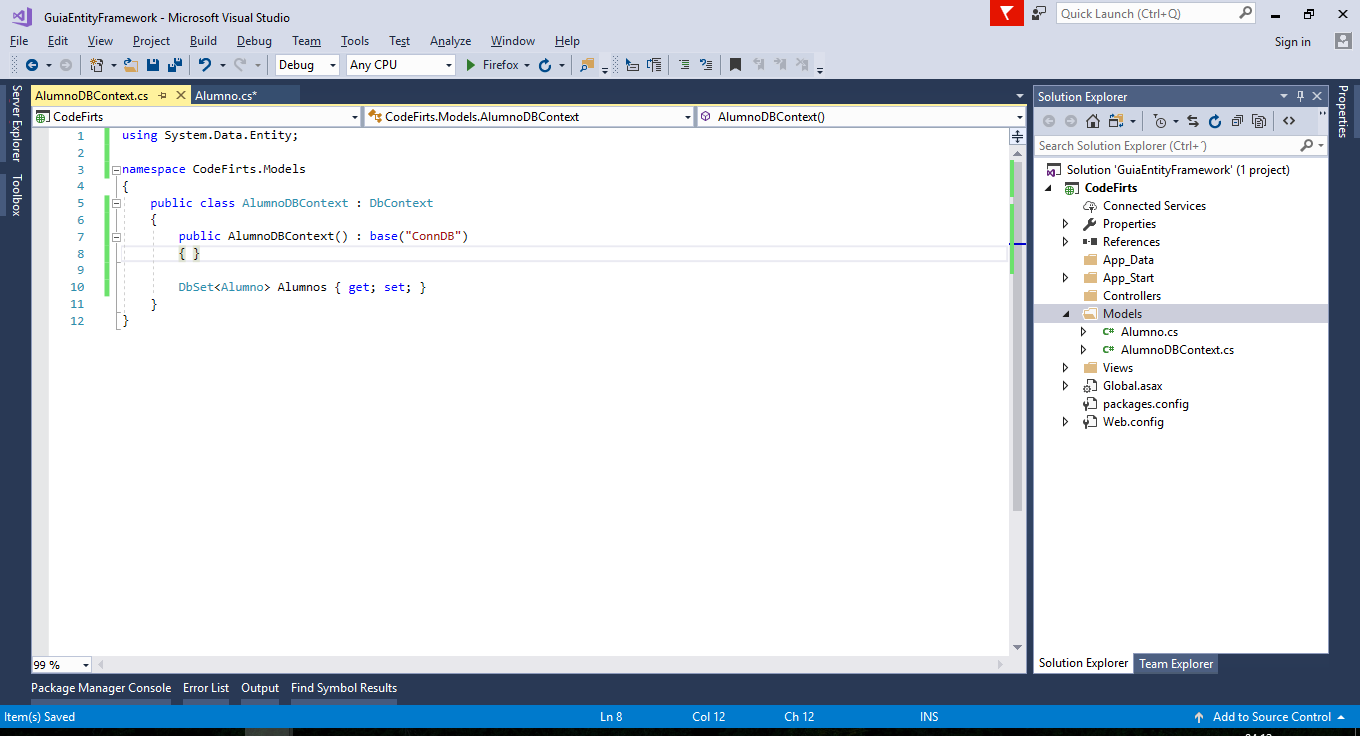
1. **Crear las clases para el modelo de la BD a crear (POCO).**



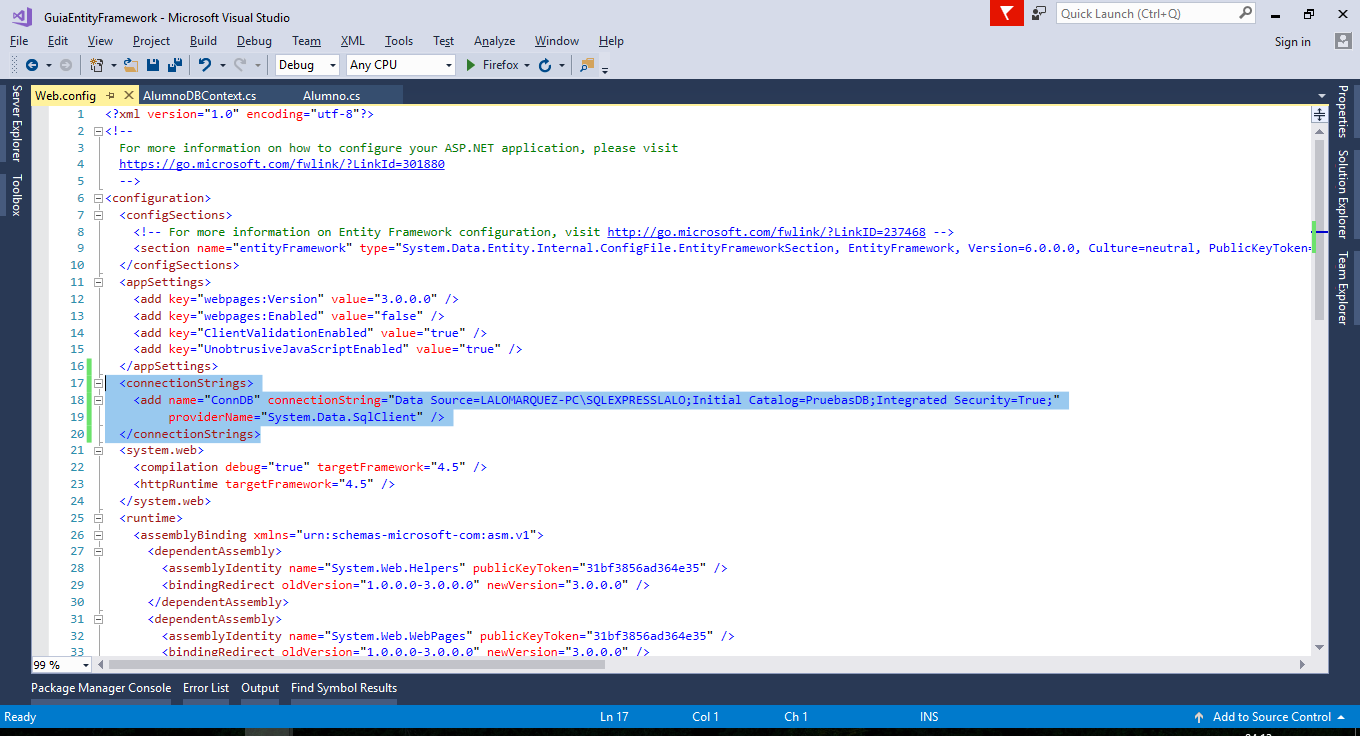
Agregar los atributos a la clase, con esto se formaran las columnas de la tabla a crear en la BD y se hace por mesio de **DataAnnotations** usando **Fluent API.**



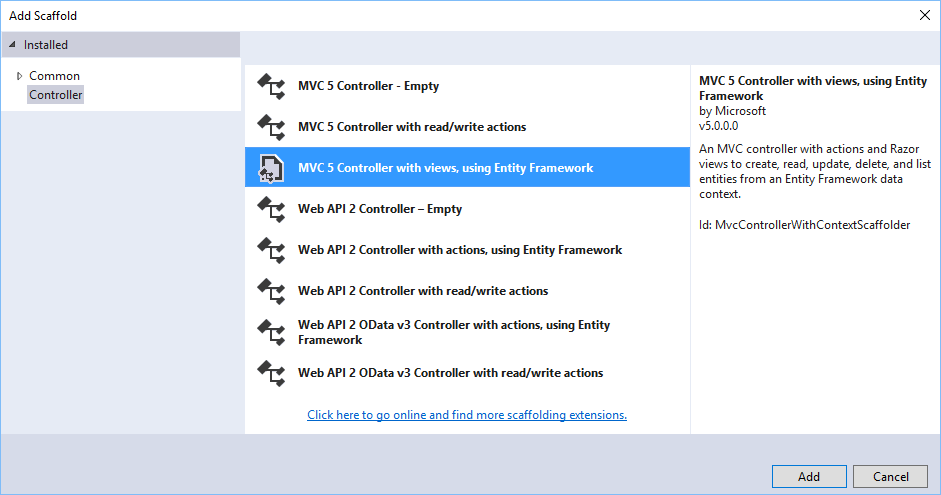
1. **Crear la clase para el mapeo de datos, heredar DbConext, agregar los DbSet y establecer un el constructor la el nombre para el connectionString**

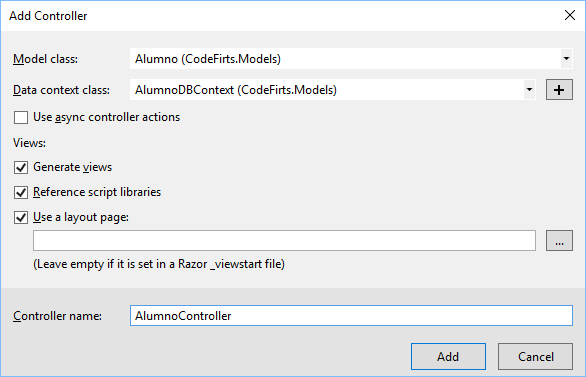


1. **Agregar la cadena de conexión para la BD (web.config).**

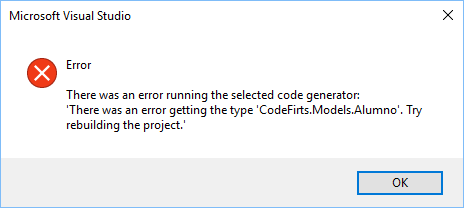


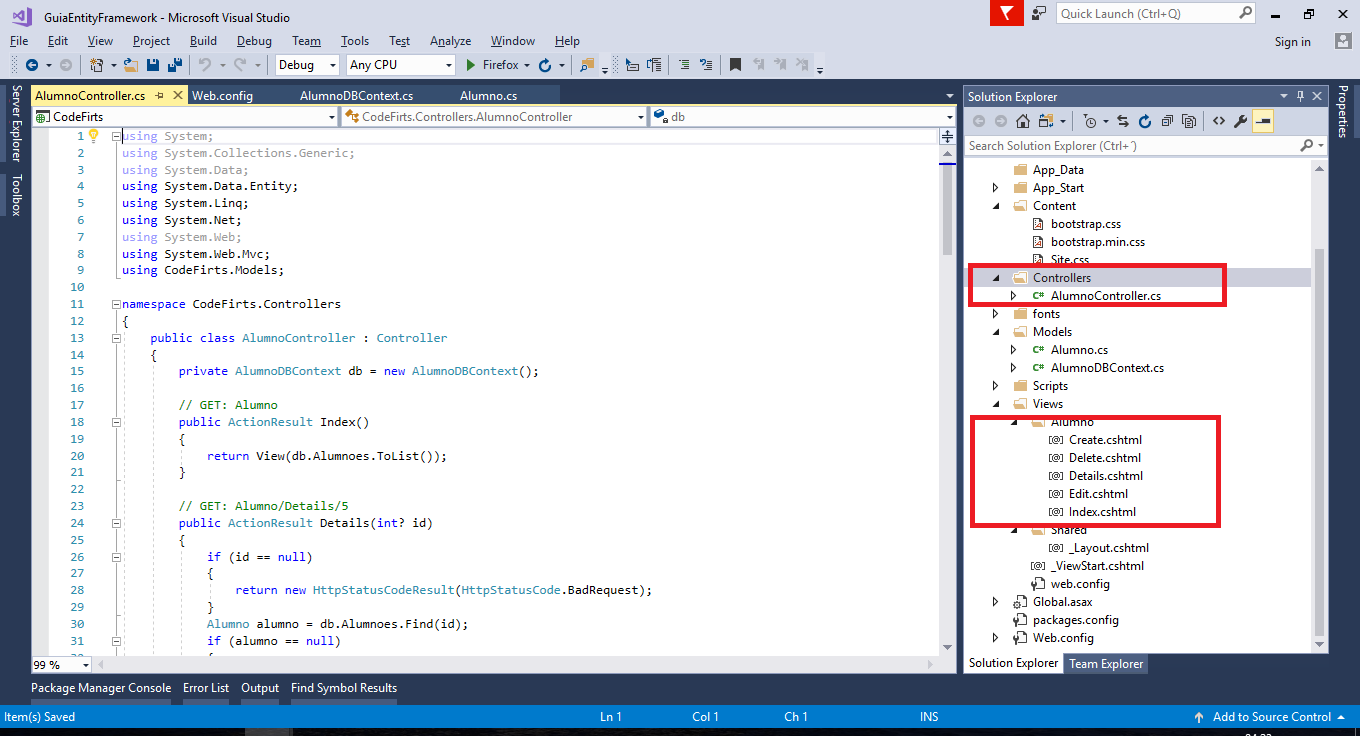
1. **Agregar un Controlador y las Vistas.**



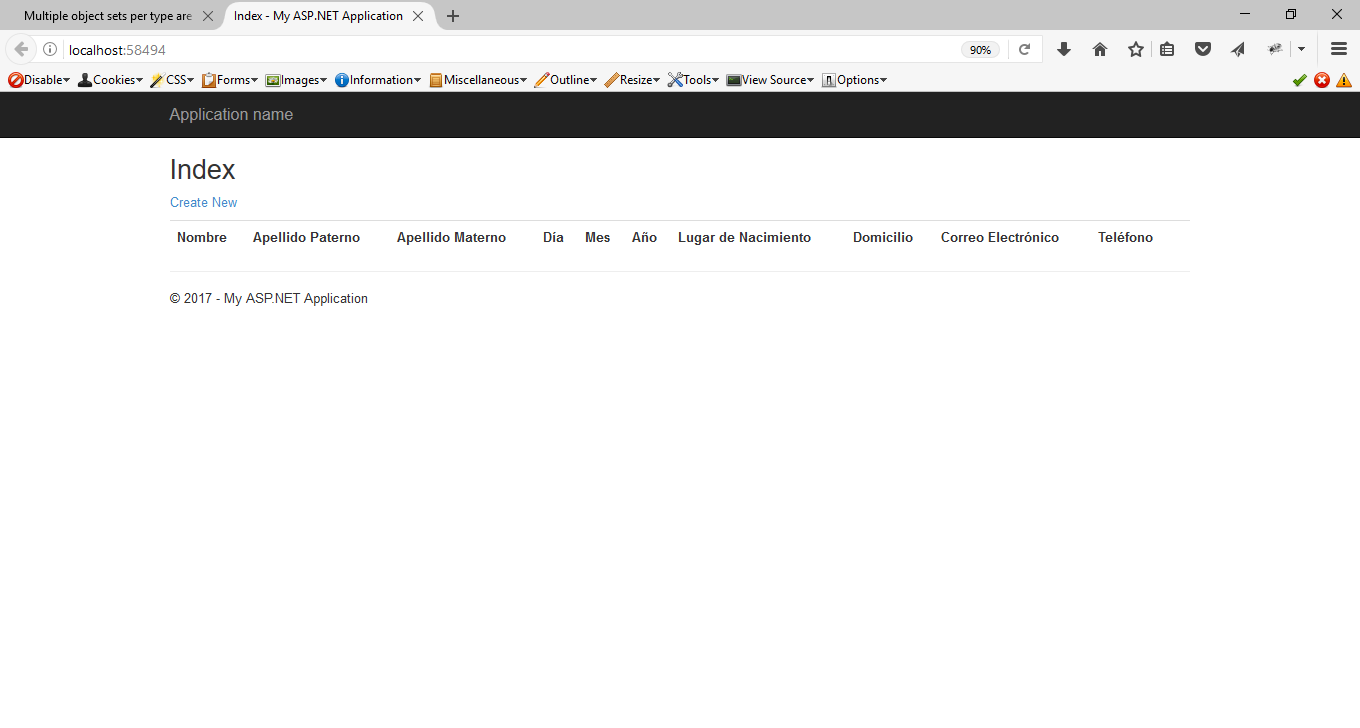


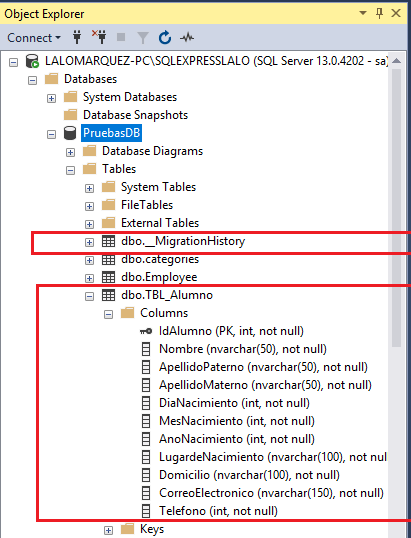
* **Si sale este error, recompilar.**



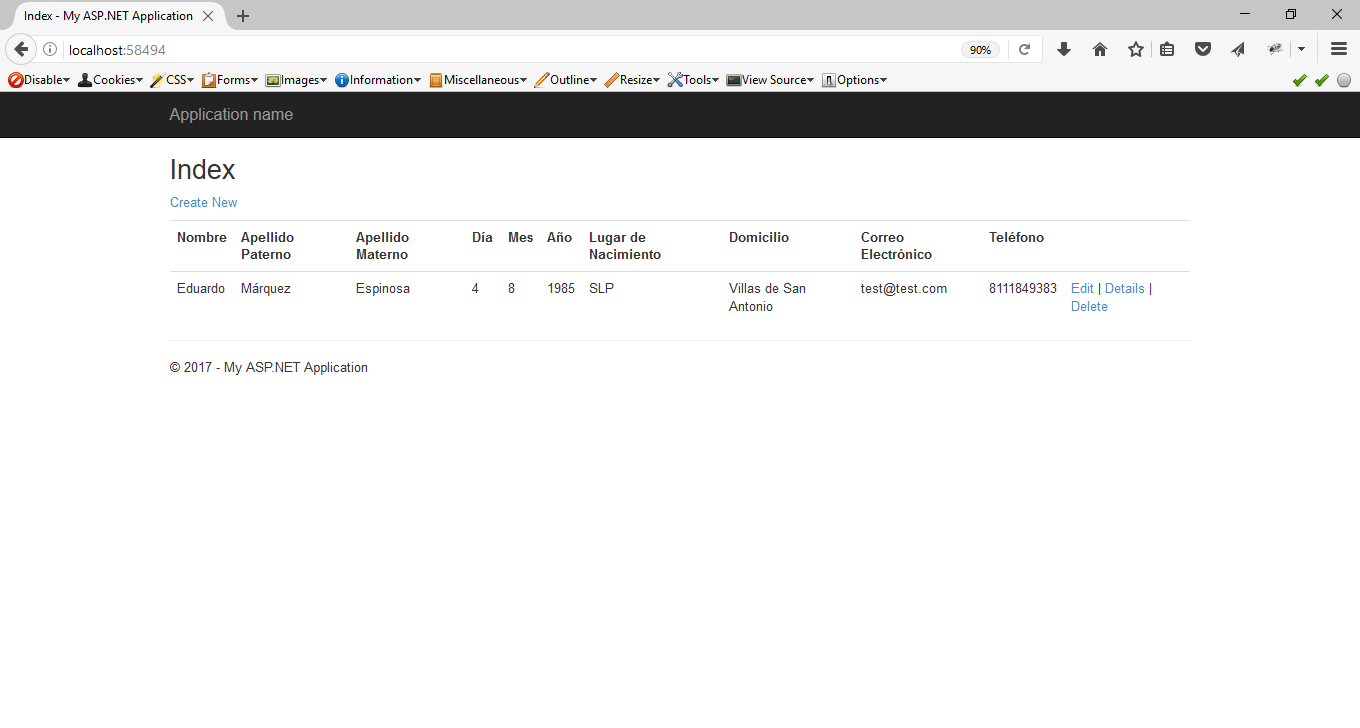


1. **Ejecutar la app y validar que se cree la TBL en la BD.**





1. **CRUD**

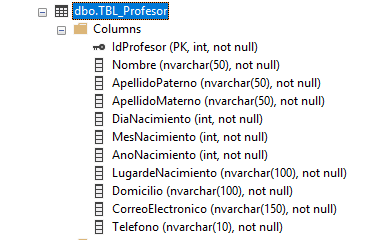


# **Code First to an Existing Database.**

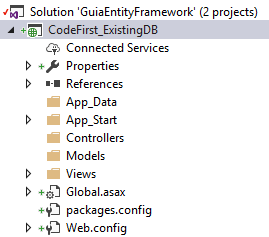
Code First también puede trabajar con base de datos existentes, puedes ver más información en [Code First to an Existing Database](http://msdn.microsoft.com/es-es/data/jj200620).

**Pasos:**

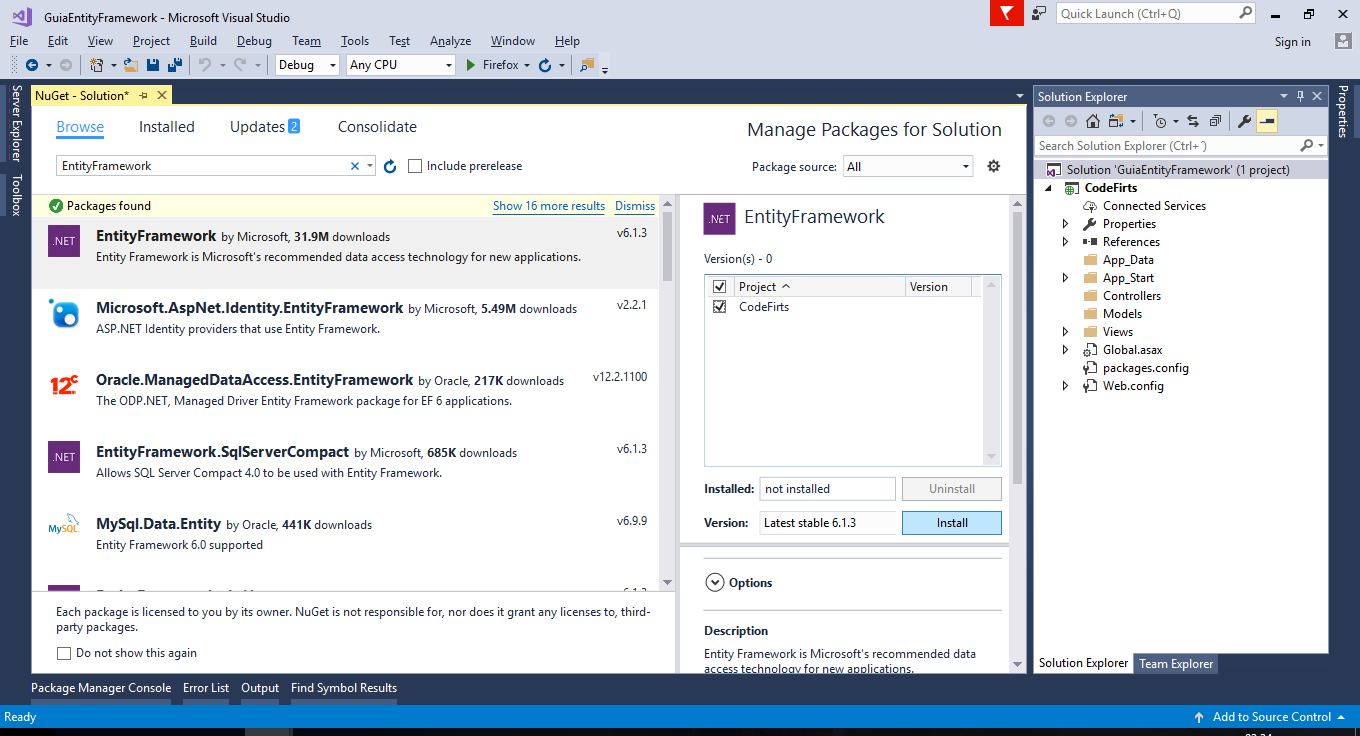
1. **Creamos una tabla en la BD.**



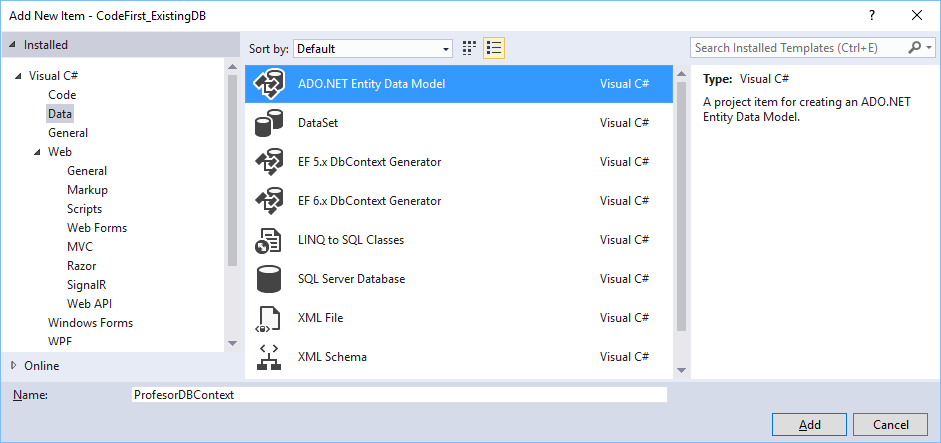
1. **Creamos una app MVC.**

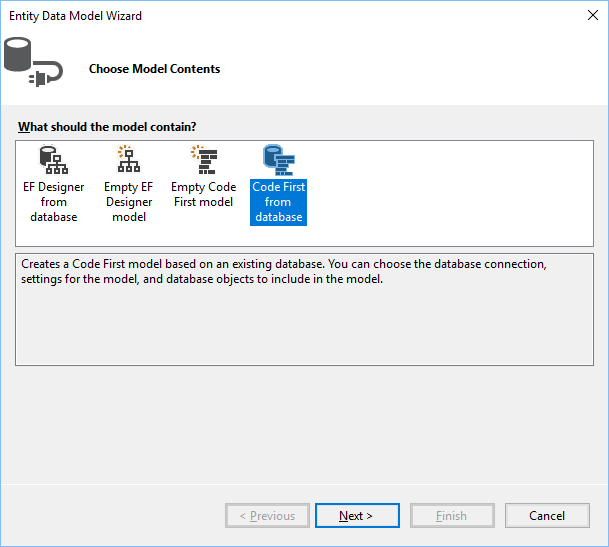
****

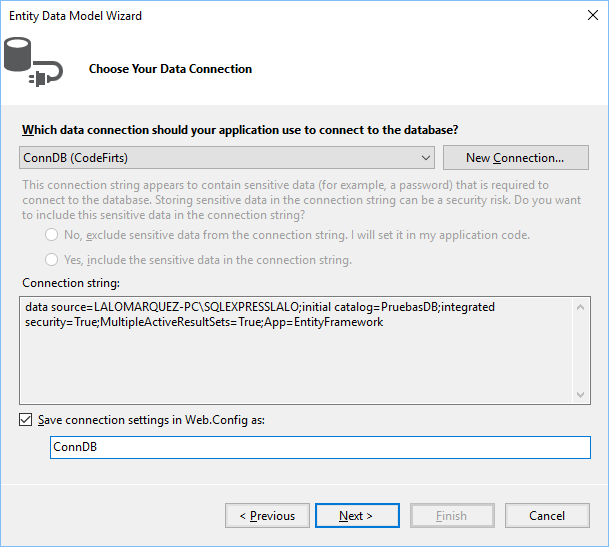
1. **Instalamos EF**



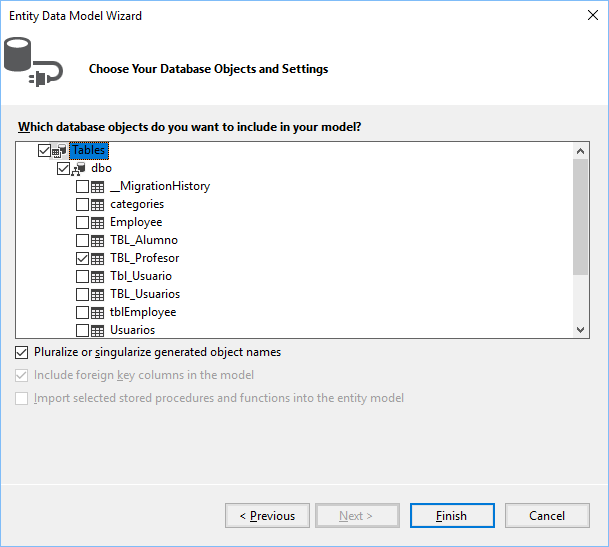
1. **Agregamos el ADO.NET Entity Data Model** **para el mapeo de datos de la tabla.**



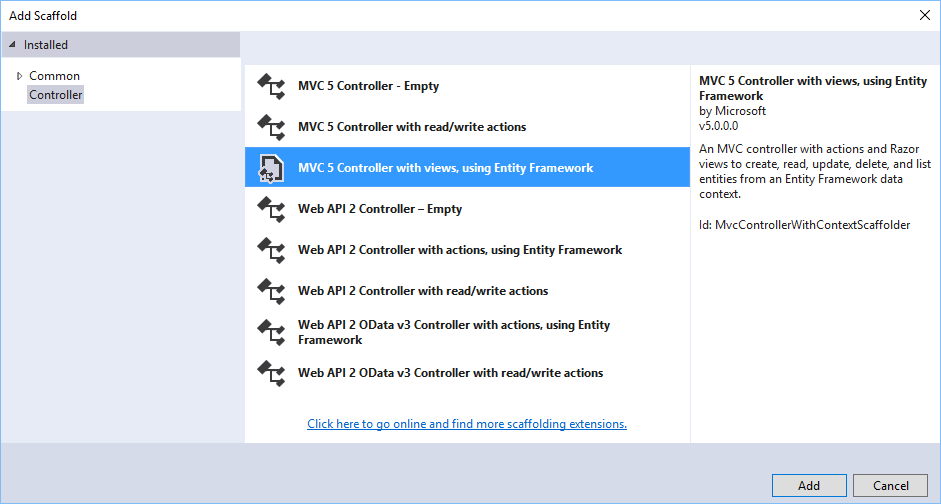


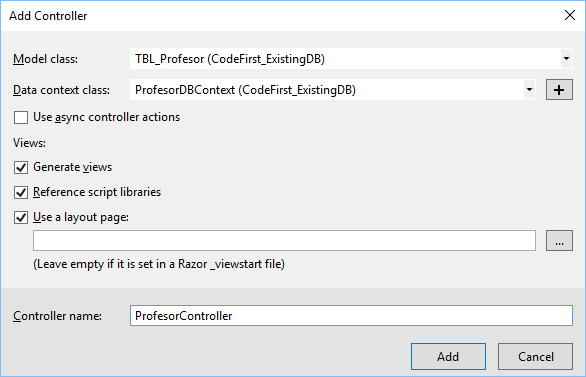


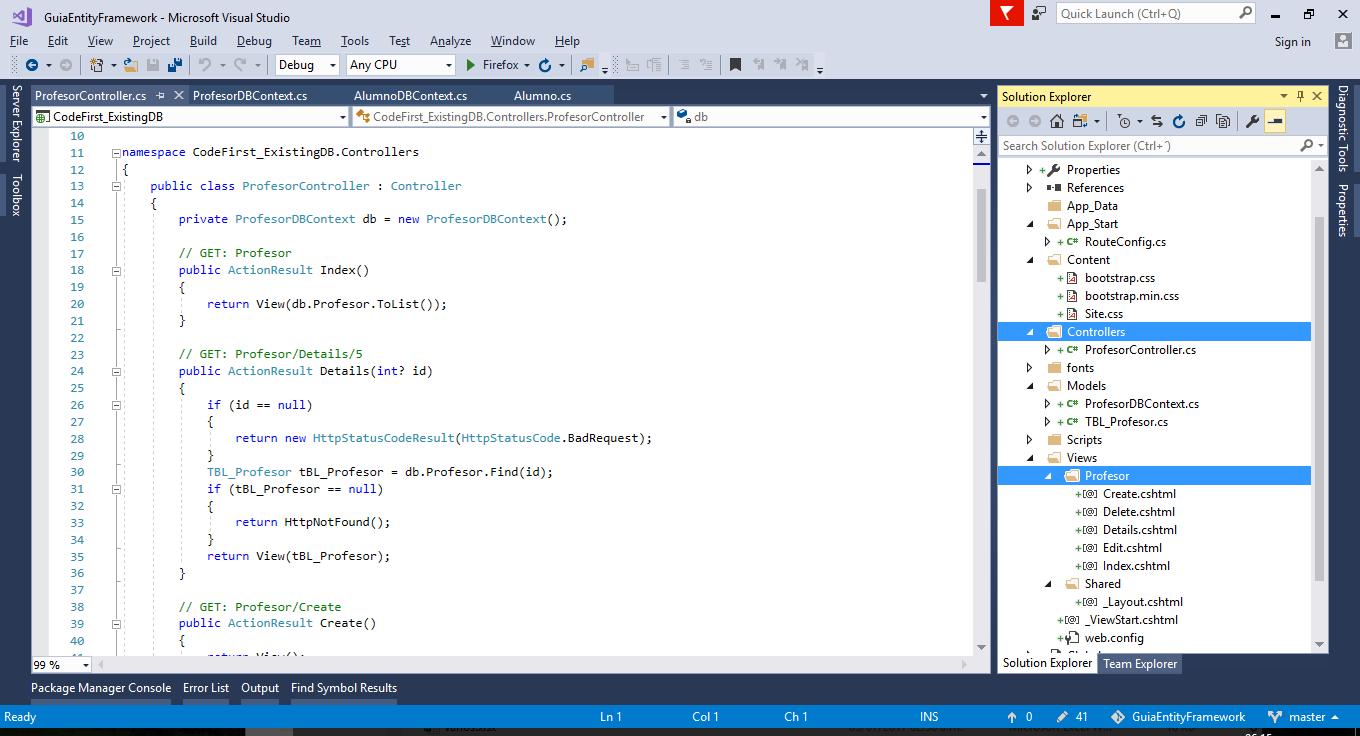
**Seleccionamos la tabla a agregar.**



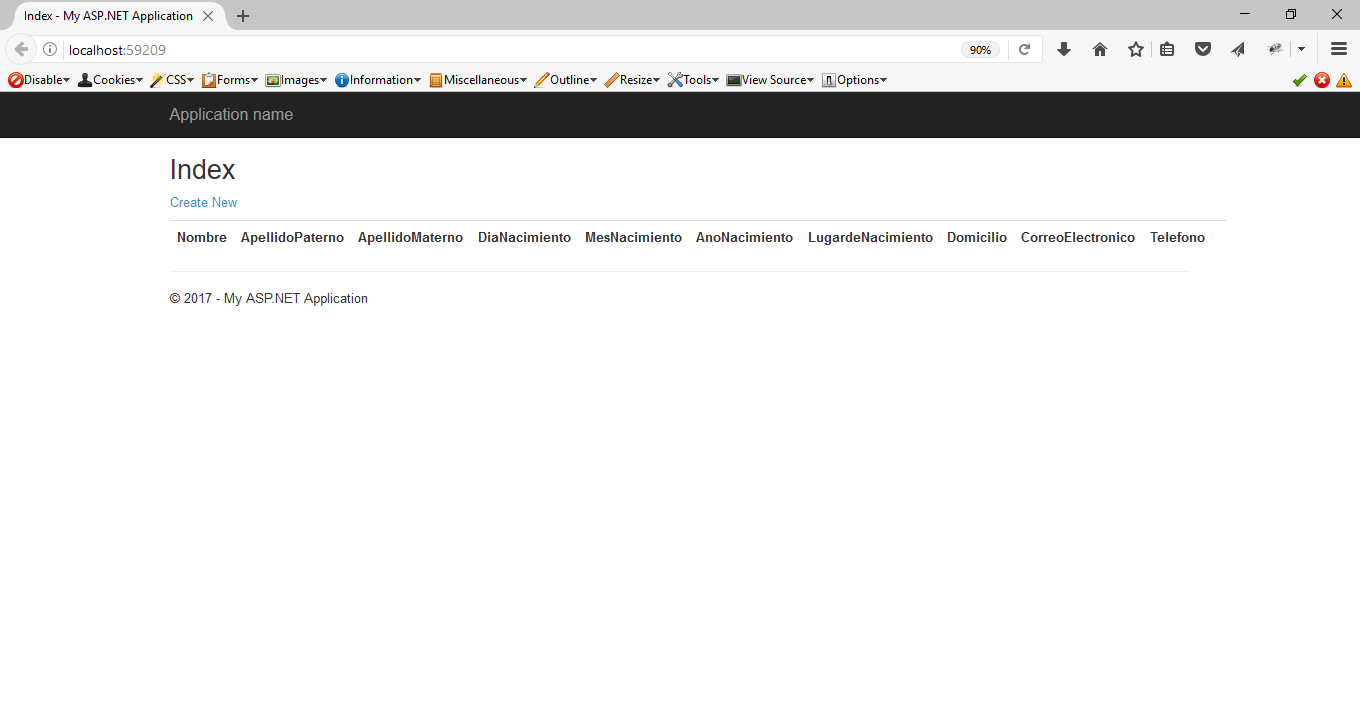
1. **Agregar un Controlador y las Vistas.**

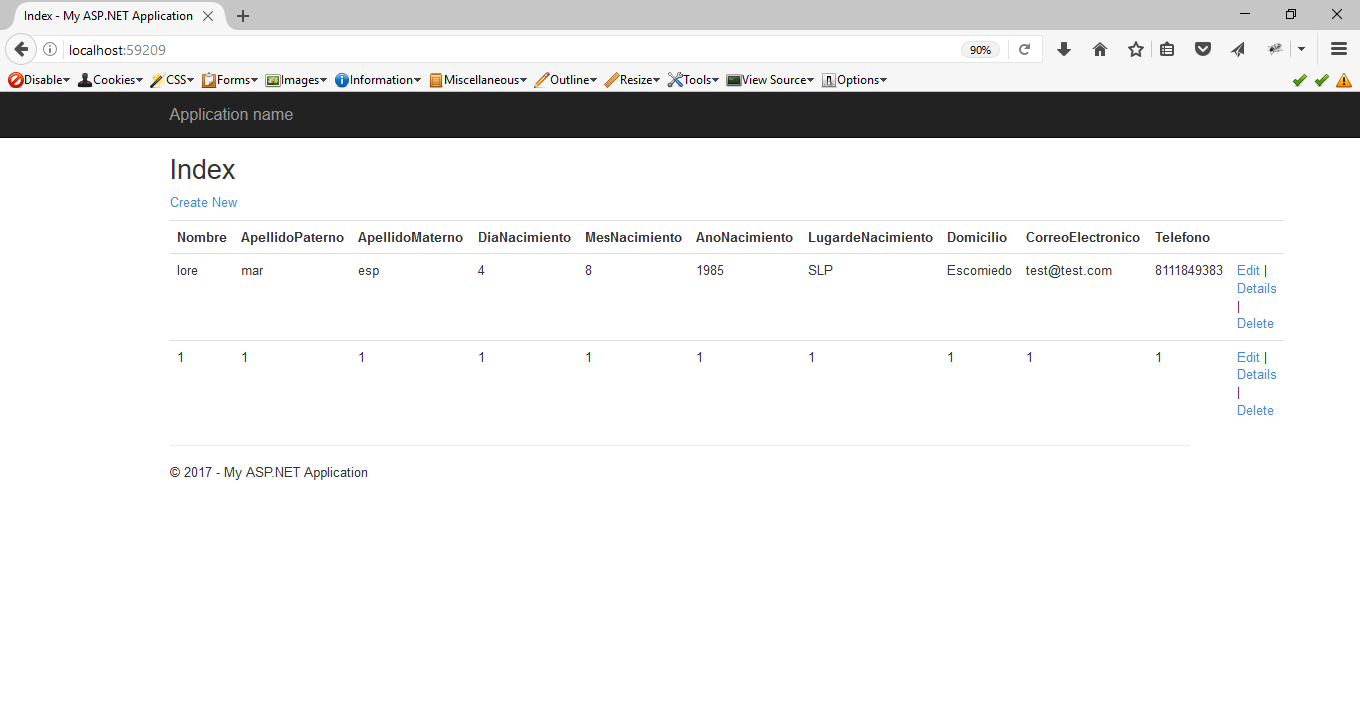






1. **Ejecutamos la aplicación y verificamos el CRUD.**





## **Code First using existing Stored Procedures:**

**Pasos:**

1. **Agregar en el dbContextCrear el método:**

    protected override void OnModelCreating(DbModelBuilder modelBuilder)

    {

        modelBuilder.Entity<Employee>().MapToStoredProcedures();

        base.OnModelCreating(modelBuilder);

    }

1. **Mapear los parámetros de los SP**

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Entity<TBL\_Profesor>().MapToStoredProcedures(

sp =>

sp.Update(u => u.HasName("[dbo].[sp\_UpdateProfesor]")

.Parameter(p => p.IdProfesor, "IdProfesor")

.Parameter(p => p.Nombre, "Nombre")

.Parameter(p => p.ApellidoPaterno, "ApellidoPaterno")

.Parameter(p => p.ApellidoMaterno, "ApellidoMaterno")

.Parameter(p => p.DiaNacimiento, "DiaNacimiento")

.Parameter(p => p.MesNacimiento, "MesNacimiento")

.Parameter(p => p.AnoNacimiento, "AnoNacimiento")

.Parameter(p => p.LugardeNacimiento, "LugardeNacimiento")

.Parameter(p => p.Domicilio, "Domicilio")

.Parameter(p => p.CorreoElectronico, "CorreoElectronico")

.Parameter(p => p.Telefono, "Telefono"))

.Delete(d => d.HasName("[dbo].[sp\_DeleteProfesor]")

.Parameter(p => p.IdProfesor, "IdProfesor"))

.Insert(i => i.HasName("[dbo].[sp\_InsertProfesor]")

.Parameter(p => p.Nombre, "Nombre")

.Parameter(p => p.ApellidoPaterno, "ApellidoPaterno")

.Parameter(p => p.ApellidoMaterno, "ApellidoMaterno")

.Parameter(p => p.DiaNacimiento, "DiaNacimiento")

.Parameter(p => p.MesNacimiento, "MesNacimiento")

.Parameter(p => p.AnoNacimiento, "AnoNacimiento")

.Parameter(p => p.LugardeNacimiento, "LugardeNacimiento")

.Parameter(p => p.Domicilio, "Domicilio")

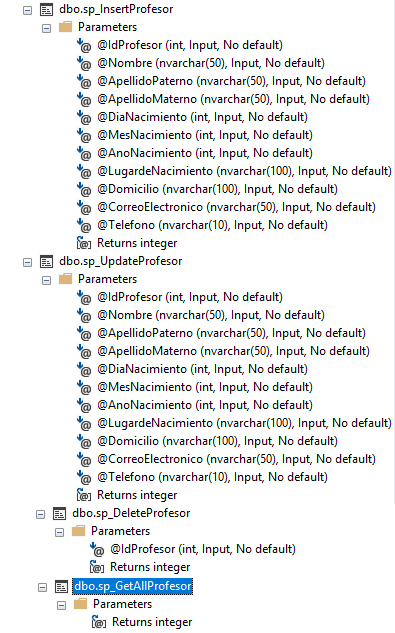
.Parameter(p => p.CorreoElectronico, "CorreoElectronico")

.Parameter(p => p.Telefono, "Telefono"))

);

base.OnModelCreating(modelBuilder);

}

****

1. **Del lado del Controller, básicamente queda igual:**

**INSERT**

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "IdProfesor,Nombre,ApellidoPaterno,ApellidoMaterno,DiaNacimiento,MesNacimiento,AnoNacimiento,LugardeNacimiento,Domicilio,CorreoElectronico,Telefono")] TBL\_Profesor tBL\_Profesor)

{

/\*

\* Code Firts con SP

\*/

if (ModelState.IsValid)

{

db.Profesor.Add(tBL\_Profesor);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(tBL\_Profesor);

}

**UPDATE**

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "IdProfesor,Nombre,ApellidoPaterno,ApellidoMaterno,DiaNacimiento,MesNacimiento,AnoNacimiento,LugardeNacimiento,Domicilio,CorreoElectronico,Telefono")] TBL\_Profesor tBL\_Profesor)

{

/\*

\* Code Firts con SP

\*/

if (ModelState.IsValid)

{

db.Entry(tBL\_Profesor).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(tBL\_Profesor);

}

**DELETE**

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

/\*

\* Code Firts con SP

\*/

TBL\_Profesor tBL\_Profesor = db.Profesor.Find(id);

db.Profesor.Remove(tBL\_Profesor);

db.SaveChanges();

return RedirectToAction("Index");

}

**READ**

public ActionResult Index()

{

/\*

\* Code Firts con SP

\*/

var getAll = db.Database.SqlQuery<TBL\_Profesor>("sp\_GetAllProfesor");

return View(getAll.ToList());

}

1. **SP usados:**

**CREATE TABLE [dbo].[TBL\_Profesor](**

[IdProfesor] [int] IDENTITY(1,1) NOT NULL,

[Nombre] [nvarchar](50) NOT NULL,

[ApellidoPaterno] [nvarchar](50) NOT NULL,

[ApellidoMaterno] [nvarchar](50) NOT NULL,

[DiaNacimiento] [int] NOT NULL,

[MesNacimiento] [int] NOT NULL,

[AnoNacimiento] [int] NOT NULL,

[LugardeNacimiento] [nvarchar](100) NOT NULL,

[Domicilio] [nvarchar](100) NOT NULL,

[CorreoElectronico] [nvarchar](150) NOT NULL,

[Telefono] [nvarchar](10) NOT NULL,

CONSTRAINT [PK\_dbo.TBL\_Profesor] PRIMARY KEY CLUSTERED

(

[IdProfesor] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**ALTER PROCEDURE [dbo].[sp\_InsertProfesor]**

@IdProfesor int = null,

@Nombre nvarchar(50),

@ApellidoPaterno nvarchar(50),

@ApellidoMaterno nvarchar(50),

@DiaNacimiento int,

@MesNacimiento int,

@AnoNacimiento int,

@LugardeNacimiento nvarchar(100),

@Domicilio nvarchar(100),

@CorreoElectronico nvarchar(50),

@Telefono nvarchar(10)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

INSERT INTO [dbo].[TBL\_Profesor]

([Nombre]

,[ApellidoPaterno]

,[ApellidoMaterno]

,[DiaNacimiento]

,[MesNacimiento]

,[AnoNacimiento]

,[LugardeNacimiento]

,[Domicilio]

,[CorreoElectronico]

,[Telefono])

VALUES

(@Nombre

,@ApellidoPaterno

,@ApellidoMaterno

,@DiaNacimiento

,@MesNacimiento

,@AnoNacimiento

,@LugardeNacimiento

,@Domicilio

,@CorreoElectronico

,@Telefono);

SELECT SCOPE\_IDENTITY() AS IdProfesor

END

**ALTER PROCEDURE [dbo].[sp\_GetAllProfesor]**

AS

BEGIN

SET NOCOUNT ON;

Select \* From TBL\_Profesor;

END

**ALTER PROCEDURE [dbo].[sp\_UpdateProfesor]**

@IdProfesor int,

@Nombre nvarchar(50),

@ApellidoPaterno nvarchar(50),

@ApellidoMaterno nvarchar(50),

@DiaNacimiento int,

@MesNacimiento int,

@AnoNacimiento int,

@LugardeNacimiento nvarchar(100),

@Domicilio nvarchar(100),

@CorreoElectronico nvarchar(50),

@Telefono nvarchar(10)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

SET NOCOUNT ON;

UPDATE [dbo].[TBL\_Profesor]

SET [Nombre] = @Nombre

,[ApellidoPaterno] = @ApellidoPaterno

,[ApellidoMaterno] = @ApellidoMaterno

,[DiaNacimiento] = @DiaNacimiento

,[MesNacimiento] = @MesNacimiento

,[AnoNacimiento] = @AnoNacimiento

,[LugardeNacimiento] = @LugardeNacimiento

,[Domicilio] = @Domicilio

,[CorreoElectronico] = @CorreoElectronico

,[Telefono] = @Telefono

WHERE [IdProfesor] = @IdProfesor;

END

**CREATE PROCEDURE [dbo].[sp\_DeleteProfesor]**

-- Add the parameters for the stored procedure here

@IdProfesor int

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

DELETE FROM [dbo].[TBL\_Profesor]

where IdProfesor = @IdProfesor

END

GO

**EXECUTE**

--C

Exec sp\_InsertProfesor

@Nombre = 'Valente',

@ApellidoPaterno = 'mar',

@ApellidoMaterno = 'esp',

@DiaNacimiento = 04,

@MesNacimiento = 04,

@AnoNacimiento = 1978,

@LugardeNacimiento = 'slp',

@Domicilio = 'escomiedo2',

@CorreoElectronico = 'no',

@Telefono = '1245'

--R

Exec [dbo].[sp\_GetAllProfesor]

--U

Exec [dbo].[sp\_UpdateProfesor]

@IdProfesor=8,

@Nombre = 'Valen',

@ApellidoPaterno = 'mar',

@ApellidoMaterno = 'esp',

@DiaNacimiento = 04,

@MesNacimiento = 04,

@AnoNacimiento = 1978,

@LugardeNacimiento = 'slp',

@Domicilio = 'escomiedo',

@CorreoElectronico = 'no',

@Telefono = '1245'

--D

Exec [dbo].[sp\_DeleteProfesor] 9

* **Para ejecutar un SP que este fuera del método** OnModelCreating**, es necesario** **agregarlo en la clase del** DbContext(ProfesorDBContext), **ejemplo:**

public List<TBL\_Profesor> **GetByProfesor**(string name, out int procResult)

{

var procResultParam = new SqlParameter

{

ParameterName = "@procResult",

SqlDbType = SqlDbType.Int,

Direction = ParameterDirection.Output

};

var sqlQuery = Database.SqlQuery<TBL\_Profesor>(

"exec @procResult = sp\_GetbyProfesor @Nombre",

new object[]

{

new SqlParameter

{

ParameterName = "@Nombre",

Value = name,

SqlDbType = SqlDbType.VarChar,

Direction = ParameterDirection.Input

},

procResultParam

}).ToList();

procResult = (int)procResultParam.Value;

return sqlQuery;

}

* **Mandarlo llamar en el método del controller o en la clase donde se valla a utilizar, ejemplo**

private ProfesorDBContext db = new ProfesorDBContext();

int procResult = 0;

var list = db.**GetByProfesor**("bb", out procResult);

ViewBag.resultado = procResult;

ViewBag.lista = list;

* **SP**

CREATE PROCEDURE [dbo].[sp\_GetbyProfesor]

@Nombre varchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

SET NOCOUNT ON;

select \* from TBL\_Profesor

where Nombre = @Nombre

END

## **enable-migrations**

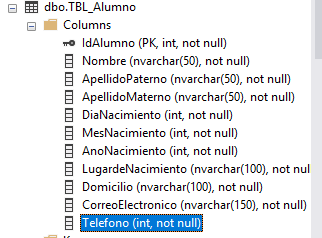
**Se habilitara la migración para cambiar el tipo de dato del campo Telefono de la tabla TBL\_Alumno de int a nvarchar.**

PM> enable-migrations

Checking if the context targets an existing database...

Detected database created with a database initializer. Scaffolded migration '201707160930254\_InitialCreate' corresponding to existing database. To use an automatic migration instead, delete the Migrations folder and re-run Enable-Migrations specifying the -EnableAutomaticMigrations parameter.

Code First Migrations enabled for project CodeFirts.



## **add-migration Initial**

PM> add-migration Initial

Scaffolding migration 'Initial'.

The Designer Code for this migration file includes a snapshot of your current Code First model. This snapshot is used to calculate the changes to your model when you scaffold the next migration. If you make additional changes to your model that you want to include in this migration, then you can re-scaffold it by running 'Add-Migration Initial' again.

## **update-database**

PM> update-database

Specify the '-Verbose' flag to view the SQL statements being applied to the target database.

Applying explicit migrations: [201707171847070\_Initial].

Applying explicit migration: 201707171847070\_Initial.

Unable to update database to match the current model because there are pending changes and automatic migration is disabled. Either write the pending model changes to a code-based migration or enable automatic migration. Set DbMigrationsConfiguration.AutomaticMigrationsEnabled to true to enable automatic migration.

You can use the Add-Migration command to write the pending model changes to a code-based migration.

