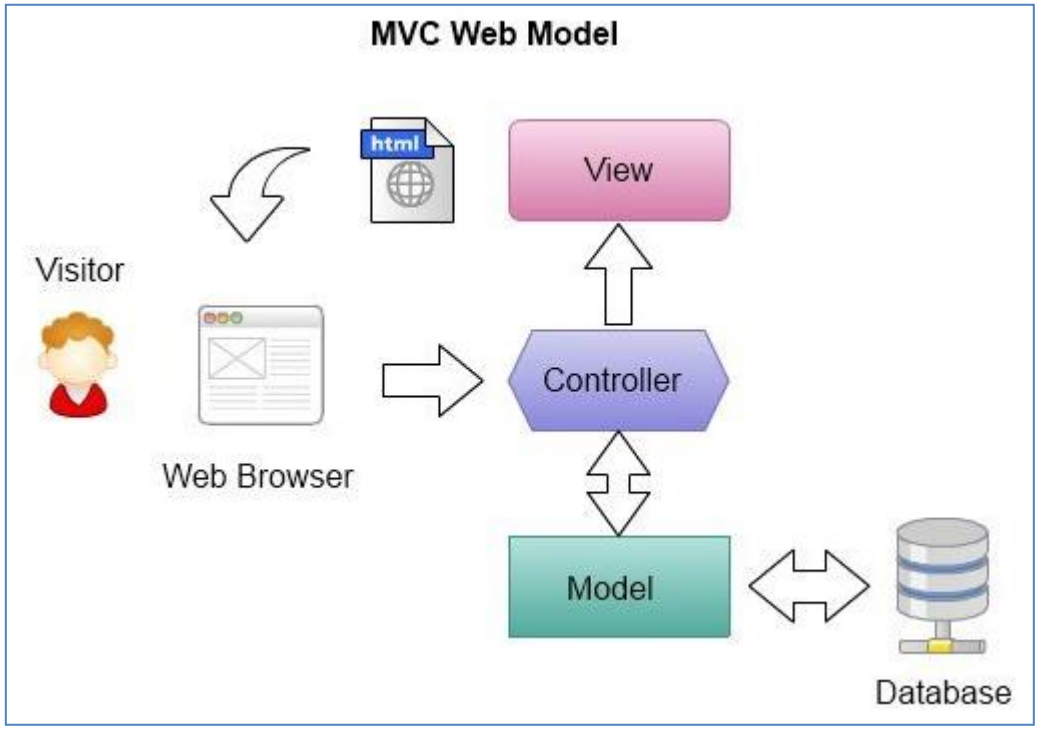
Arquitectura MVC básica

En un aplicación **MVC** (sigla en inglés que significa Model View Controller = Modelo Vista Controlador), es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

- El **Modelo** que contiene una representación de los datos que maneja el sistema, su lógica/reglas de negocio, y sus mecanismos de persistencia de datos hacia un repositorio de datos.
- La **Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con éste.
- El **Controlador**, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.



El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos para recuperar/persistir los datos.
- Definir las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si los productos pedidos a un proveedor aún no llegaron al sector de abastecimientos, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.

El controlador es responsable de:

- Recibir los eventos de entrada (un click, un cambio en un campo de texto, la petición de una página web/vista, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

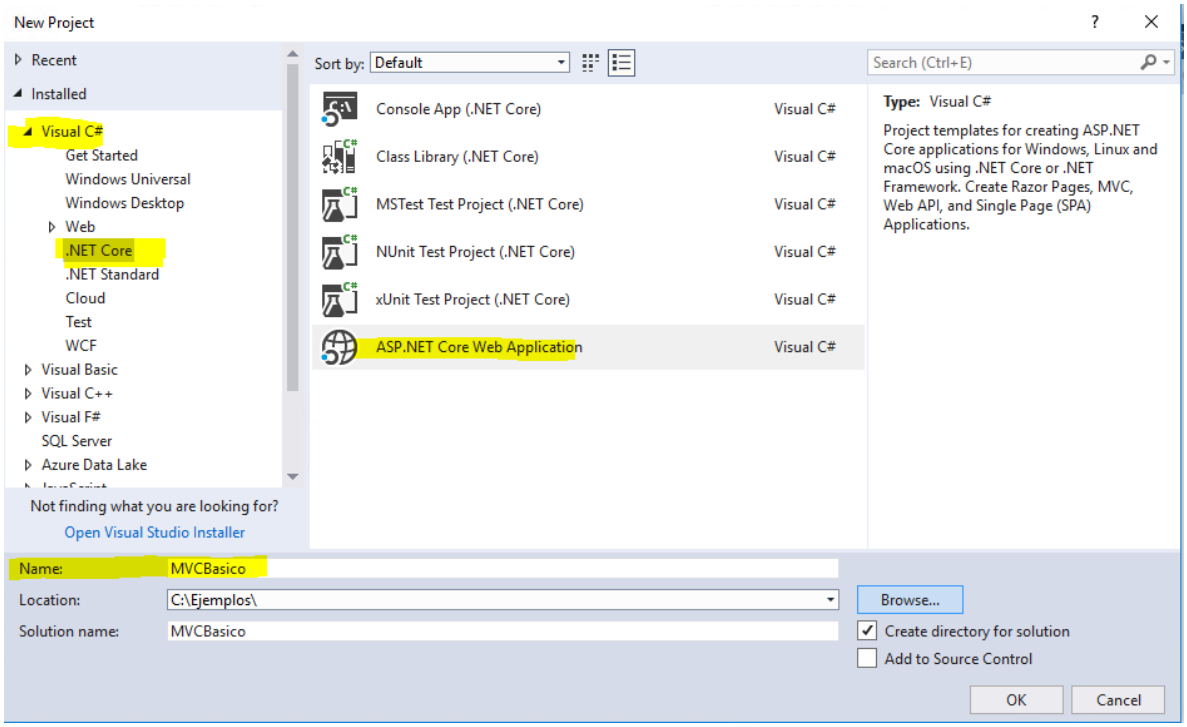
Las vistas son responsables de:

- Recibir datos del modelo y para mostrarlos al usuario.
- Tener un registro de su controlador asociado (normalmente porque además ese controlador instancia la vista).

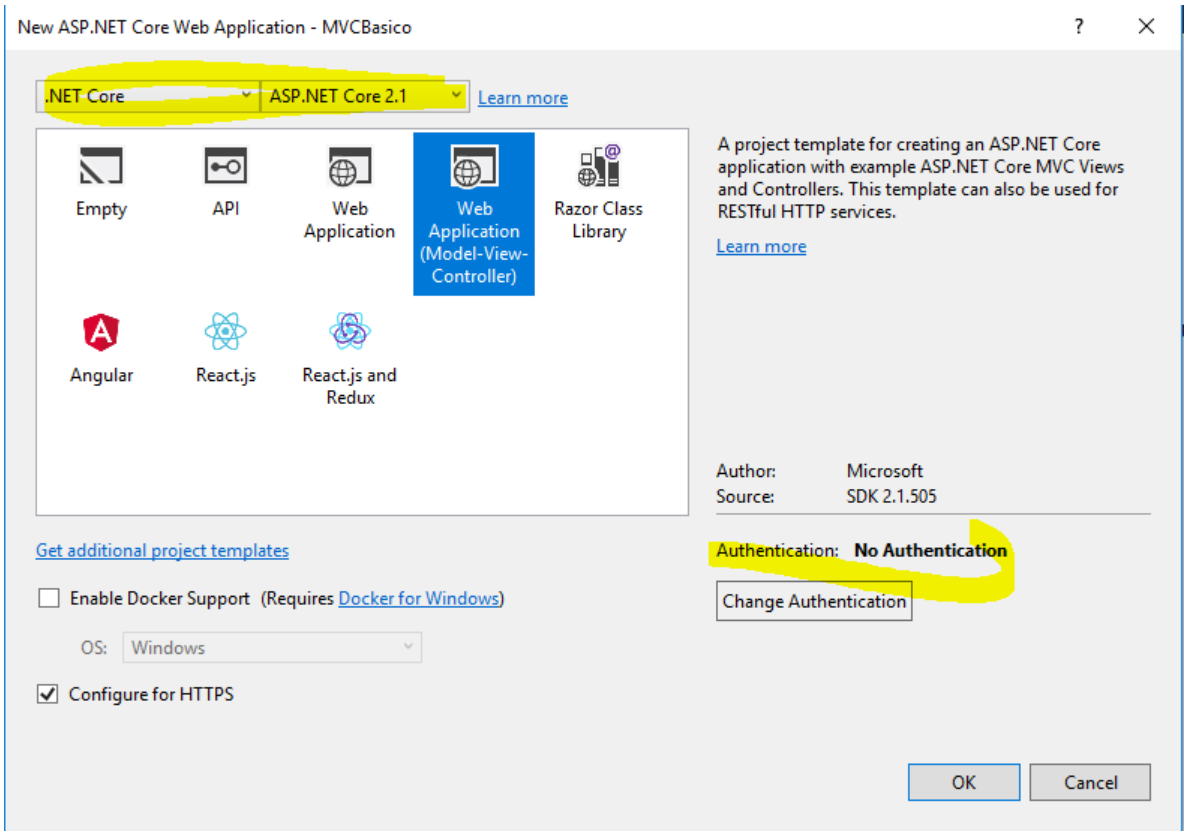
Tutorial básico aplicación MVC con ASP .NET Core

En este tutorial implementarás un sitio web con ASP .NET Core MVC que se conecta a una base de datos y permite realizar operaciones CRUD a la tabla desde el sitio.

Paso 1. Abre **Visual Studio 2017/2019** y selecciona la opción **Crear un nuevo proyecto ASP.NET Core Web Application** llamado **MVCBasico.**



Paso 2. A continuación, elige la plantilla **Web Application (Model-View-Controller)**. En la sección superior asegúrate de elegir **.NET Core** y la versión más reciente instalada, y sin autenticación:



Paso 3. Espera a que finalice la creación del proyecto. En la carpeta **Models** crea una nueva clase llamada **Estudiante** con el código mostrado a continuación.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace MVCBasico.Models
{
    public class Estudiante
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
```

```
        public int Id { get; set; }

        public string Nombre { get; set; }

        public int Edad { get; set; }

        [Display(Name = "Fecha inscripción")]
        public DateTime FechaInscripto { get; set; }

        [EnumDataType(typeof(Deporte))]
        public Deporte DeporteFavorito { get; set; }
    }
}
```

Paso 4. Y en la carpeta **Models** crea una nueva enumeración para definir los deportes favoritos del estudiante con el código mostrado a continuación.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MVCBasico.Models
{
    public enum Deporte
    {
        Natacion,
        Futbol,
        Voley,
        Basquet,
        Truco,
        Ajedrez
    }
}
```

Paso 5. Ahora crea una carpeta con el nombre **Context**, dentro de la cual agregarás una nueva clase llamada **EscuelaDatabaseContext**. Su código es el siguiente:

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MVCBasico.Models;

namespace MVCBasico.Context
{
    public class EscuelaDatabaseContext : DbContext
```

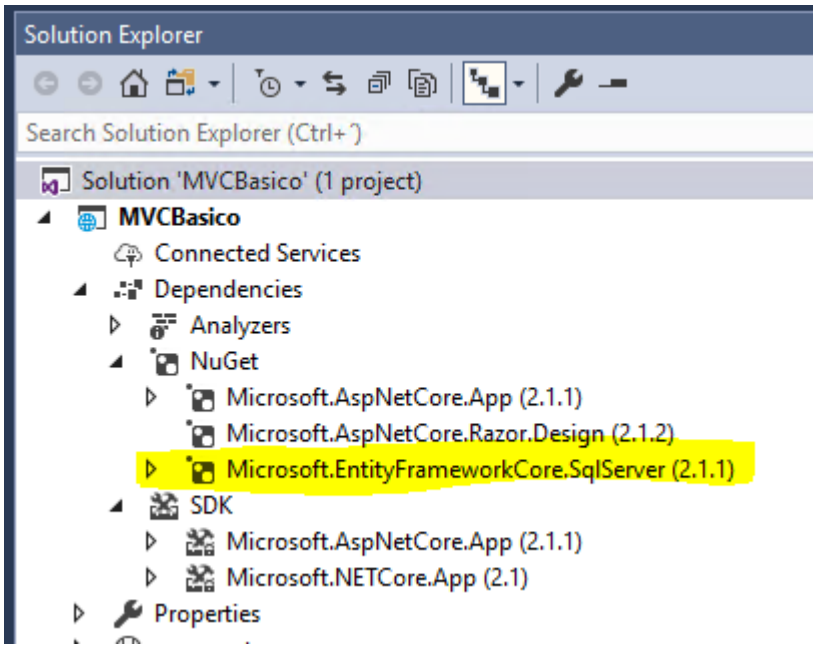
```
    {

        public
EscuelaDatabaseContext(DbContextOptions<EscuelaDatabaseContext> options)
: base(options)
        {

        }

        public DbSet<Estudiante> Estudiantes { get; set; }
    }
}
```

Paso 6. El siguiente paso es definir la cadena de conexión a la base de datos donde almacenaremos la información. Esto se realiza en el archivo **appsettings.json** de la siguiente forma:

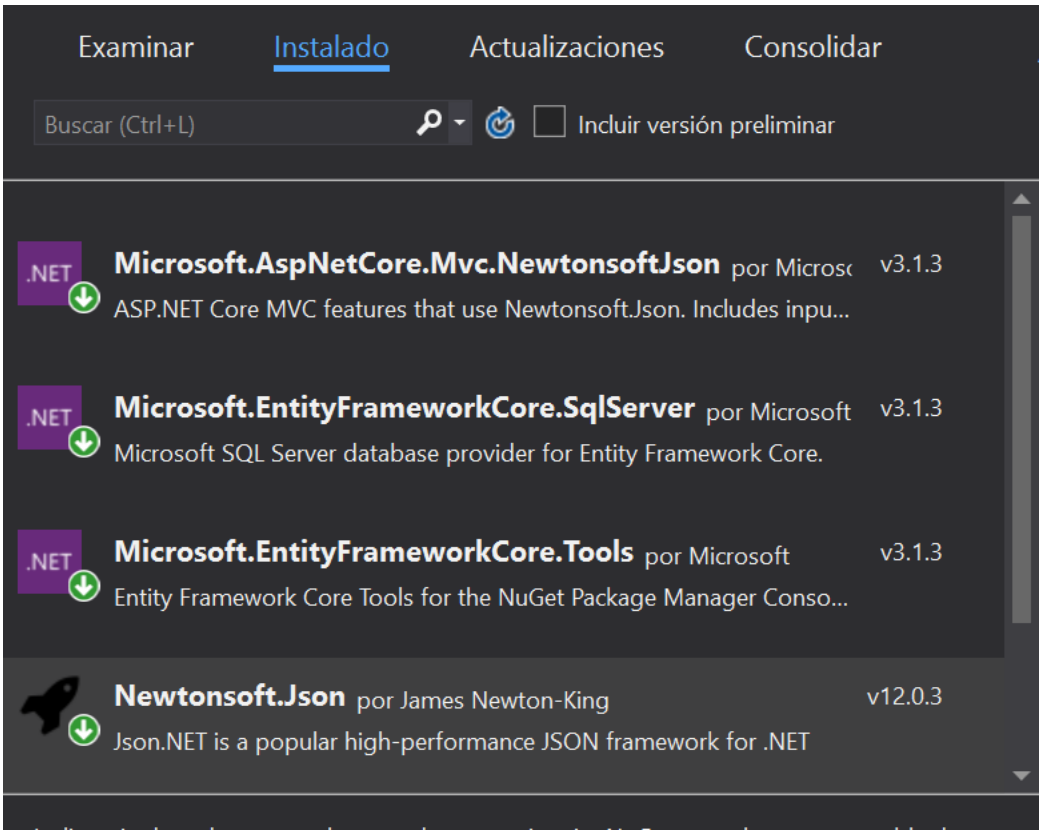
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*",

  "ConnectionString": {
    //"EscuelaDBConnection": "server=MY_SERVER;database=EscuelaDB;User
ID=MY_USER;password=MY_PASSWORD;",
    "EscuelaDBConnection":
"Server=.\\SQLExpress;Database=EscuelaDB;Trusted_Connection=True;"
  }
}
```

Paso 7. Da clic derecho en el nombre del proyecto **MVCBasico** y elige la opción **Manage Nuget packages**.

Paso 8. Busca el paquete **Microsoft.EntityFrameworkCore.SqlServer**. Selecciona la **última versión disponible** según la versión del SDK de .Net Core para instalarla.



O versión 3.1.3 o superior de ambas librerías de Entity Framework Core y la librería de NewtonSoft



Paso 9. Una vez instalado el paquete, abre el archivo **Startup.cs** y registra el contexto en el método **ConfigureServices** de la siguiente manera:

- Agregando los espacios de nombres

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.EntityFrameworkCore;
using MVCBasico.Context;
using Newtonsoft.Json;
```

- Indicando la cadena de conexión y el gestor de bases de datos
- Recomendable indicar que Json ignore las referencias circulares.

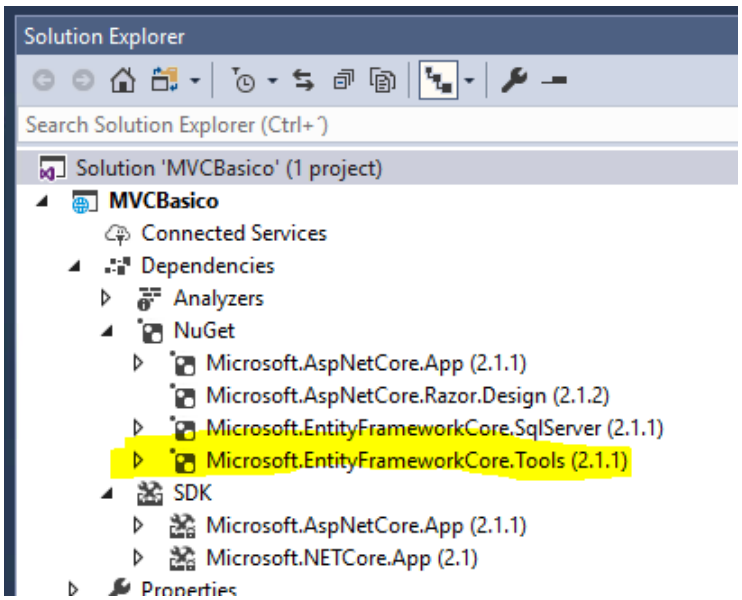
VERSION MVC 2.1

```
public void ConfigureServices(IServiceCollection services)
{
      services.Configure<CookiePolicyOptions>(options =>
      {
      // This lambda determines whether user consent for non-essential
cookies is needed for a given request.
      options.CheckConsentNeeded = context => true;
      options.MinimumSameSitePolicy = SameSiteMode.None;
            });

      services.AddDbContext<EscuelaDatabaseContext>(
                  options =>
options.UseSqlServer(Configuration["ConnectionString:EscuelaDBConnection"
]));

      services.AddMvc()
            .AddJsonOptions(options =>
            options.SerializerSettings.ReferenceLoopHandling =
ReferenceLoopHandling.Ignore)

      .SetCompatibilityVersion(CompatibilityVersion.Version_2_1);

}
```

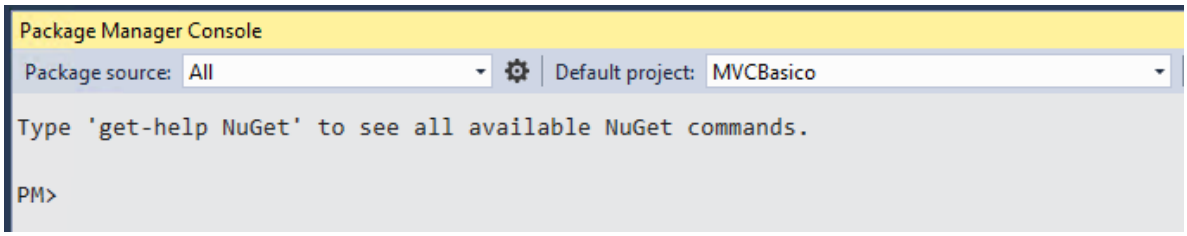
VERSION MVC 3.1

```
public void ConfigureServices(IServiceCollection services)
{
services.Configure<CookiePolicyOptions>(options =>
            {
                // This lambda determines whether user consent for non-
essential cookies is needed for a given request.
                options.CheckConsentNeeded = context => true;
                options.MinimumSameSitePolicy = SameSiteMode.None;
            });

services.AddDbContext<EscuelaDatabaseContext>(options =>
options.UseSqlServer(Configuration["ConnectionString:EscuelaDBConnection"
]));

            services.AddMvc().AddNewtonsoftJson(options =>
options.SerializerSettings.ReferenceLoopHandling =
ReferenceLoopHandling.Ignore)

.SetCompatibilityVersion(CompatibilityVersion.Version_3_0);

}
```

Paso 10. Nuevamente en el Administrador de Paquetes Nuget, busca el paquete **Microsoft.EntityFrameworkCore.Tools**. Selecciona la misma versión que elegiste para el paquete **Microsoft.EntityFrameworkCore.SqlServer** e instálalo en el proyecto.

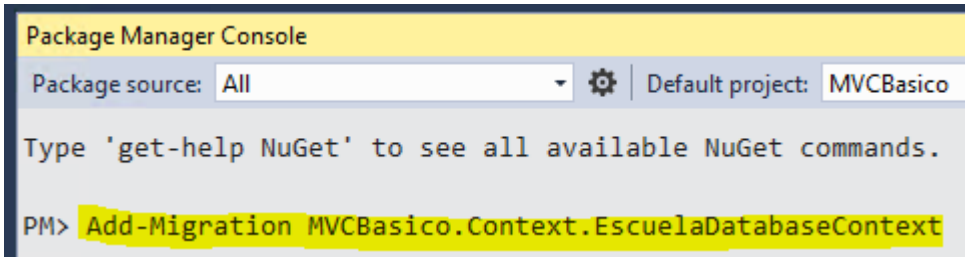


Paso 11. En el menú **Herramientas de Visual Studio** da clic en **Nuget Package Manager** y selecciona **Package Manager Console**.

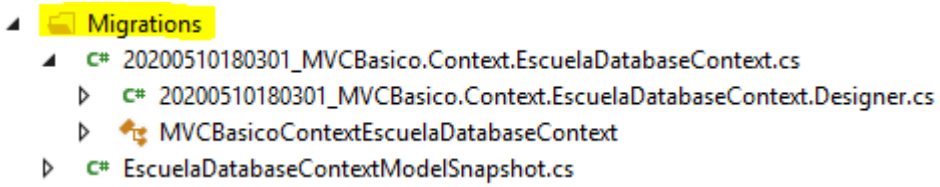


Paso 12. En la consola, ejecuta el comando **Add-Migration** pasando como parámetro la clase de contexto de la siguiente manera:

```
Add-Migration MVCBasico.Context.EscuelaDatabaseContext
```

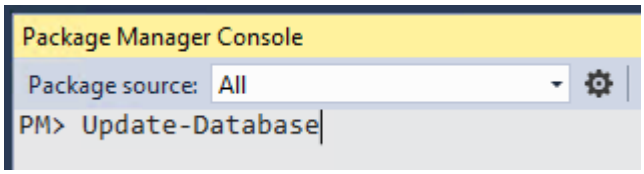


Esto creará la carpeta **Migrations** y varias clases de soporte para migraciones bajo el esquema **Code-First**.

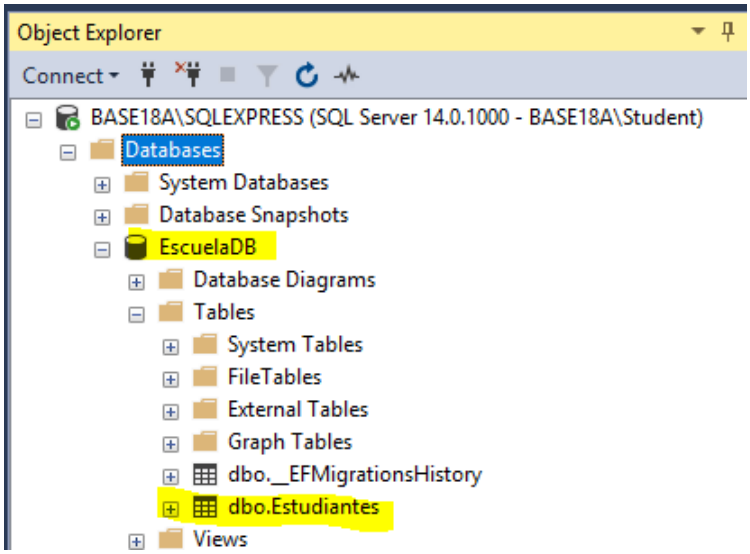


Paso 13. Ahora ejecuta el comando **Update-Database** para crear la base de datos en tu servidor junto con la tabla **Estudiantes** (según el DBSet indicado en el contexto).

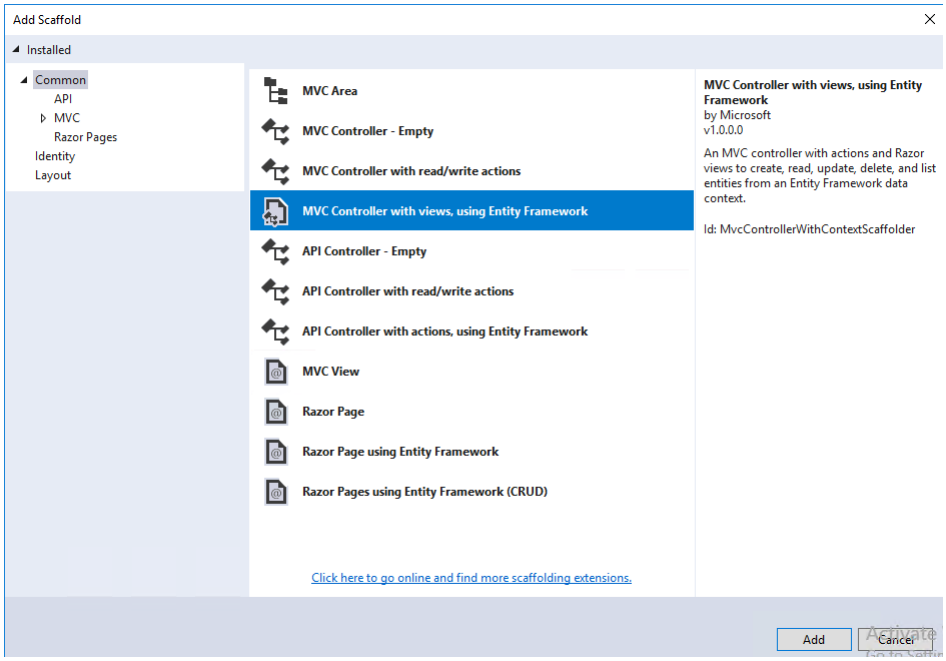
```
Update-Database
```



Si verificamos desde Sql Server, veremos la creación de la base de datos **EscuelaDB** con la tabla **Estudiantes**.

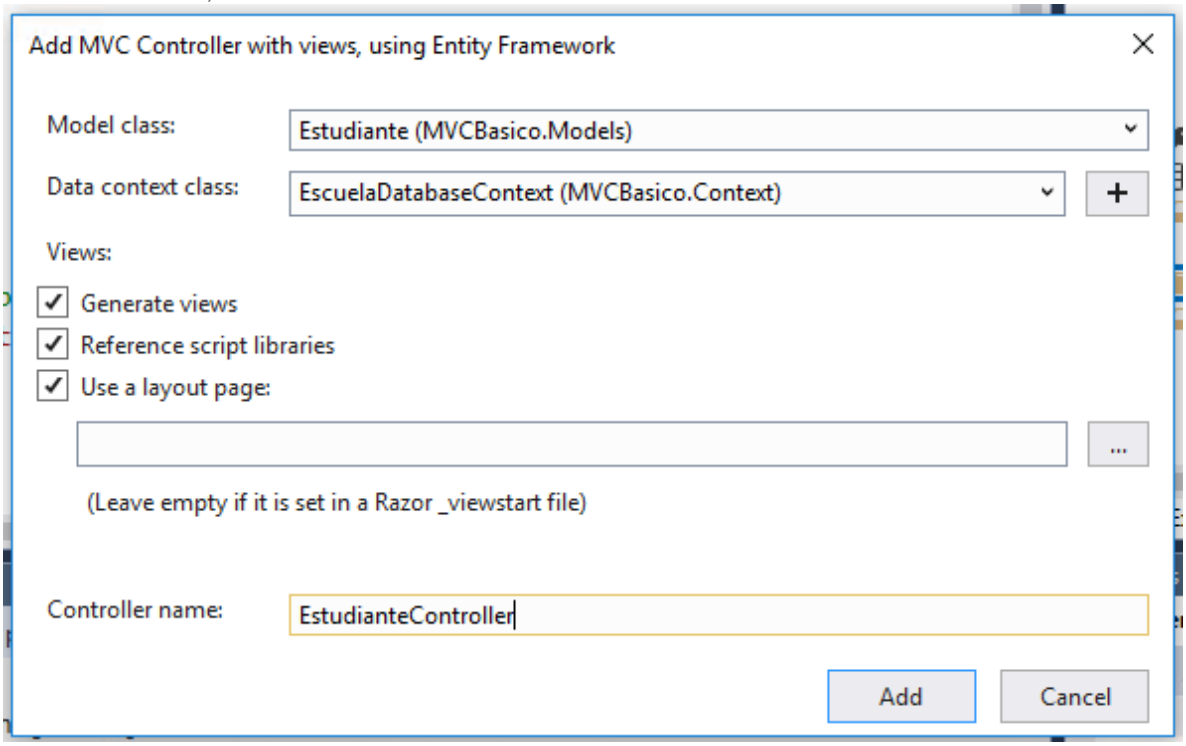


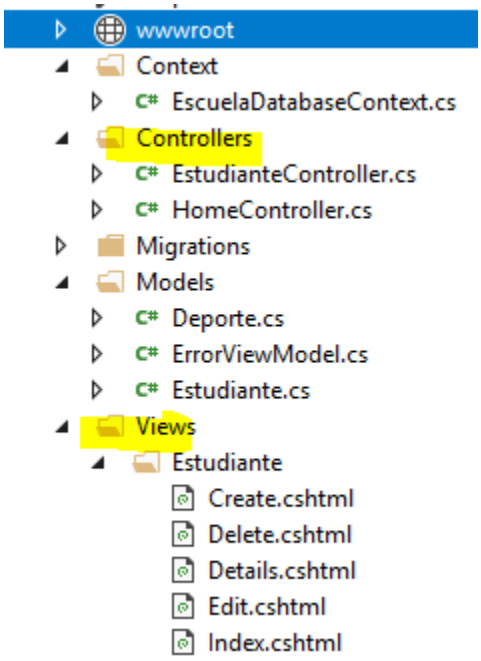
Paso 14. A continuación, en la carpeta Controllers da clic derecho y selecciona **Add** –> **New Scaffolded Item**. Elige **Controlador MVC con vistas utilizando Entity Framework**.



Paso 15. Realiza lo siguiente:

- En clase de **Modelo** elige **Estudiante**.
- En clase de **Contexto de Datos** selecciona **EscuelaDatabaseContext**.
- Selecciona las 3 opciones que aparecen en la sección **Vistas**.
- Finalmente, el nombre del **Controller** es **EstudianteController**.





Lo que sucederá a continuación es que se creará la clase **EstudianteController** (en Controllers) y también las páginas **Create, Delete, Details, Edit e Index** para **Estudiante** (en Views).

Paso 16. Explora EstudianteController y observa el código para:

- Obtener toda la información de la tabla **(método Index -GET-)**
- Obtener un registro específico **(método Details -GET-, que acepta el id del registro a buscar)**
- Agregar un nuevo elemento **(método Create -POST-, que recibe un objeto Estudiante)**
- Editar una entrada **(método Edit -POST-, que recibe un objeto Stu Estudiante dent y el id de la entrada a modificar)**
- Eliminar un dato **(método DeleteConfirmed -POST-, que acepta el id del dato a eliminar)**
- Además existe una sobrecarga de métodos (mismo nombre pero distinto verbo HTTP), los cuales representan la **View** a mostrar (se buscará un archivo **.cshtml** dentro de la carpeta **Views/ Estudiante** con el mismo nombre del método):
  - **Create (GET)**
  - **Edit (GET)**
  - **Delete (GET)**

Paso 17. De igual forma, explora los archivos generados en **Views/ Estudiante** y cómo se combina el código HTML con el de C#:

- **Index.cshtml** muestra la lista de estudiantes y contiene enlaces a las otras páginas
- **Create.cshtml** permite agregar un nuevo estudiante a la base de datos.
- **Details.cshtml** despliega la información del estudiante seleccionado y tiene un enlace para pasar a edición de datos.
- **Edit.cshtml** permite modificar los datos del estudiante seleccionado.
- **Delete.cshtml** muestra los datos del estudiante seleccionado y permite eliminar el registro de la base de datos.

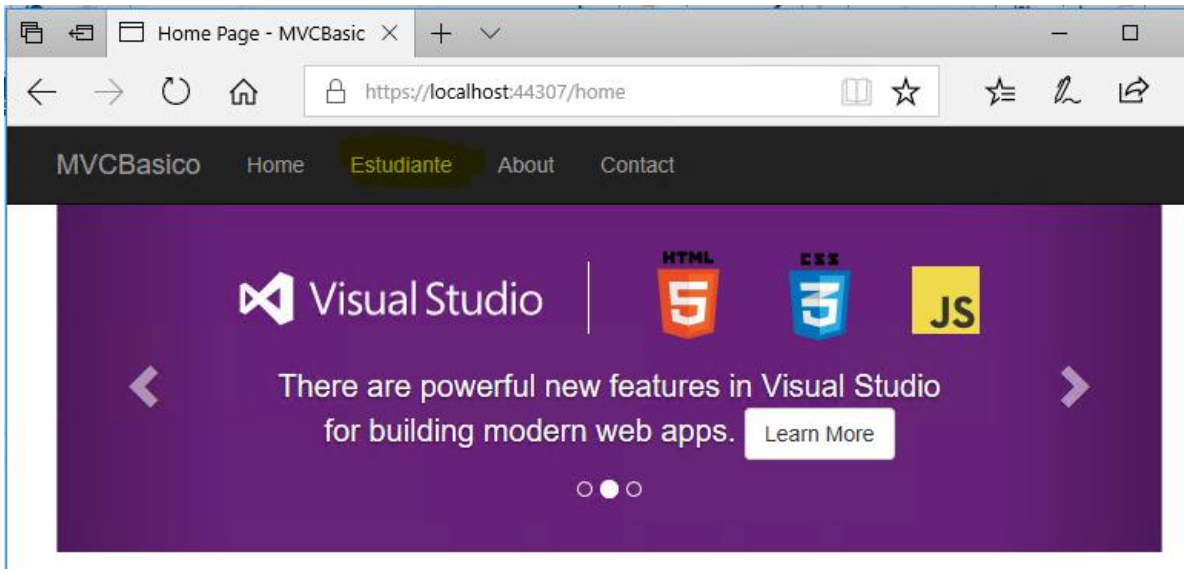
Paso 18. Abre el archivo **_Layout.cshtml** que se encuentra dentro de **Views/Shared** y agrega un nuevo elemento de menú para poder navegar a la vista **Index** del controlador **Estudiante**.

```
<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-
reverse">
      <ul class="navbar-nav flex-grow-1">
            <li class="nav-item">
                  <a class="nav-link text-dark" asp-area="" asp-
            controller="Home" asp-action="Index">Home</a>
            </li>
            <li class="nav-item">
                  <a class="nav-link text-dark" asp-area="" asp-
            controller="Estudiante" asp-action="Index">Estudiante</a>
            </li>
            <li class="nav-item">
                  <a class="nav-link text-dark" asp-area="" asp-
            controller="Home" asp-action="Privacy">Privacy</a>
            </li>
      </ul>
</div>
```

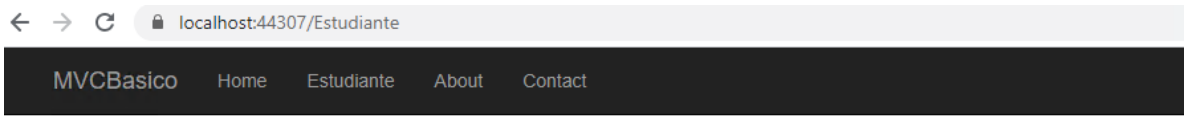
Y en las vistas **Create** y **Edit** del **Estudiante**, modificar el código para que el combo cargue el enum de los deportes.

```
<select asp-for="DeporteFavorito" class="form-control" asp-
items="Html.GetEnumSelectList<Deporte>()">
      <option selected="selected" value="">Please select</option>
</select>
```

Paso 20. Finalmente, compila y ejecuta la aplicación. Navega por cada página para verificar el funcionamiento correcto del sitio web.



Paso 21. Puede probar el CRUD y verificar en la base de datos la creación, modificación, visualización y eliminación de estudiantes.



Index

Create New

Nombre	Edad	Fecha inscripción	DeporteFavorito	
Ana	29	5/6/2020 1:00:00 AM	Natacion	Edit | Details | Delete