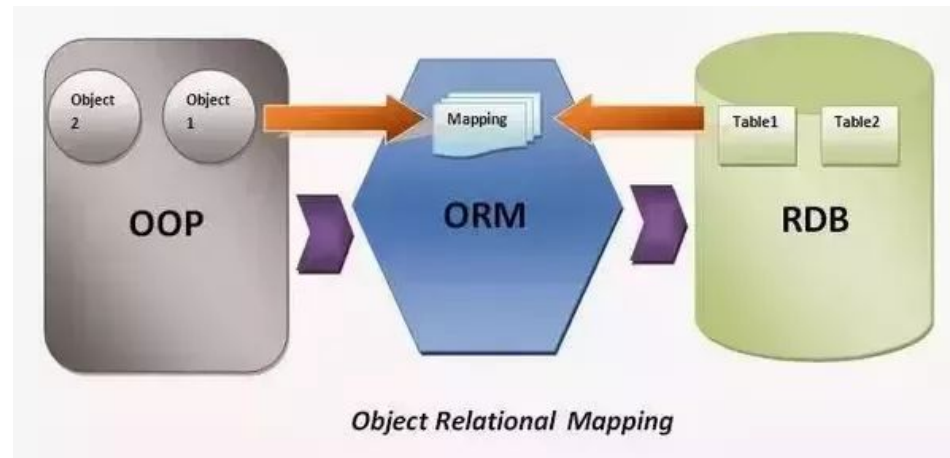


# BackEnd

## Manejo de ORM para Acceso a Datos



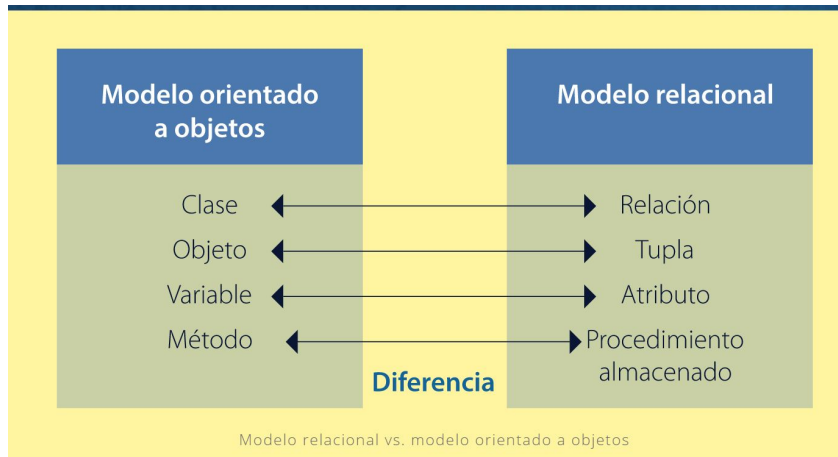
ENTITYFRAMEWORK CORE

<https://docs.microsoft.com/es-es/ef/core/>

# Bases de datos Relacionales y Orientadas a Objeto

- El **modelo relacional** para la gestión de las bases de datos (Edgar F. Codd, 1970) es el modelo de **referencia** a la hora de **almacenar y recuperar** información. Tanto es así que es el modelo más extendido e implementado. (Oracle, Microsoft SQLServer, MySQL, PostgreSQL).
- A pesar de su eficacia, el modelo relacional se vio abocado en la década de los 90 a una actualización, a una **mejora para adaptarse a los nuevos tiempos**, al fantástico futuro que ofrecía el **paradigma de Programación Orientada a Objetos (POO)**, impulsado de nuevo en estos años 90 por la imparable y creciente irrupción del lenguaje de programación Java.
- Destacando las diferencias, el **modelo relacional** hace hincapié en los **datos y sus relaciones**, mientras que el **modelo orientado a objetos** no se centra en los datos en sí, sino en las **operaciones realizadas en esos datos**.
- La aceptación recibida por la POO dio paso a una visión más amplia del problema, es decir, a un modelo orientado a objetos (modelo OO) como herramienta para diseñar software, crear código y almacenar datos.

# ORM Mapeo Objeto-Relacional



- Un *ORM* es un modelo de programación que permite mapear las estructuras de una base de datos relacional (*SQL Server*, *Oracle*, *MySQL*, etc.), en adelante *RDBMS* (*Relational Database Management System*), sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones.
- Las estructuras de la base de datos relacional quedan vinculadas con las entidades lógicas o *base de datos virtual* definida en el *ORM*, de tal modo que las acciones *CRUD* (*Create*, *Read*, *Update*, *Delete*) a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del *ORM*.
- Además de “mapear”, los *ORMs* tienden a “liberarnos” de la escritura o generación manual de código *SQL* (*Structured Query Language*) necesario para realizar las *queries* o consultas y gestionar la persistencia de datos en el *RDBMS*.

# ORM Mapeo Objeto-Relacional

- Así, los objetos o entidades de la **base de datos virtual** creada en nuestro **ORM** podrán ser **manipulados** por medio de algún lenguaje de nuestro interés según el tipo de ORM utilizado, por ejemplo, **LINQ** sobre **Entity Framework de Microsoft**.
- La **interacción** con el RDBMS quedará **delegada** en los métodos de actualización correspondientes proporcionados por el **ORM**.
- Los ORMs más completos ofrecen servicios para **persistir todos los cambios** en los estados de las entidades, previo seguimiento o tracking automático, **sin escribir una sola línea de SQL**.
- Trabajar directamente sobre las entidades de la base de datos virtual sin necesidad de generar código SQL tiene la **ventaja de acelerar el desarrollo o implementación** de nuestras aplicaciones

# Ejemplos de ORM

- **Hibernate** (Java)
- **MyBatis** (Java)
- **Ebean** (Java)
- **Entity Framework** (.NET)
- **NHibernate** (.NET)
- **MyBatis.NET** (.NET)
- **Doctrine** (PHP)
- **Propel** (PHP)
- **Rocks** (PHP)
- **Torpor** (PHP)

# EntityFramework Core

- Entity Framework (EF) Core es una versión ligera, extensible, [de código abierto](#) y multiplataforma de la popular tecnología de acceso a datos Entity Framework.
- EF Core puede servir como asignador relacional de objetos (O/RM), lo que permite a los desarrolladores de .NET trabajar con una base de datos mediante objetos .NET y **eliminar la mayoría del código de acceso a los datos que normalmente deben escribir.**
- EF Core es compatible con muchos motores de base de datos; vea [Proveedores de bases de datos](#) para más información.



# Instalación de Componentes Nuget

- <https://www.nuget.org/>

# EntityFramework Core - Elementos principales

- DbContext => Representar el Objeto de la Base de Datos (Contenedor de Datos)
- DbSet => Representar un Repositorio a Entidad de la Base de Datos



Cambiar el Proyecto de Pulsaciones  
Ado.net => Ef Core

# CAMBIOS CAPA ENTITY

Entity > Entity.csproj

```
1 <Project Sdk="Microsoft.NET.Sdk">
2   <PropertyGroup>
3     <TargetFramework>netstandard2.0</TargetFramework>
4   </PropertyGroup>
5 </Project>
6
```

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>netstandard2.1</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="System.ComponentModel.Annotations" Version="4.7.0" />
  </ItemGroup>
</Project>
```

Luego de realizar los Cambios, ejecutar desde el Proyecto de Entity el Comando **dotnet restore**

# Clase Persona de Entity

```
using System.ComponentModel.DataAnnotations;
namespace Entity
{
    public class Persona
    {
        [Key]
        public string Identificacion { get; set; }
        public string Nombre { get; set; }
        public int Edad { get; set; }
        public string Sexo { get; set; }
        public decimal Pulsacion { get; set; }
        public void CalcularPulsaciones()
        {
            if (Sexo.Equals("F") || Sexo.Equals("f"))
            {
                Pulsacion=(220 - Edad) / 10;
            }
            else
            {
                Pulsacion=(210 - Edad) / 10;
            }
        }
    }
}
```

# MODIFICACIONES A LA CAPA DE DATOS

Datos > Datos.csproj

```
1  <Project Sdk="Microsoft.NET.Sdk">
2
3    <PropertyGroup>
4      <TargetFramework>netstandard2.0</TargetFramework>
5    </PropertyGroup>
6
7    <ItemGroup>
8      <PackageReference Include="System.Collections" Version="4.3.0" />
9      <PackageReference Include="System.Data.SqlClient" Version="4.8.1" />
10
11    </ItemGroup>
12
13    <ItemGroup>
14      <ProjectReference Include="..\Entity\Entity.csproj" />
15    </ItemGroup>
16
17  </Project>
18
```

- Se reemplaza el Nuget de DataSqlClient

# Cambios al Proyecto Datos.csproj

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>netstandard2.1</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="3.0.0" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="3.0.0" />
    <PackageReference Include="System.Collections" Version="4.3.0" />
  </ItemGroup>

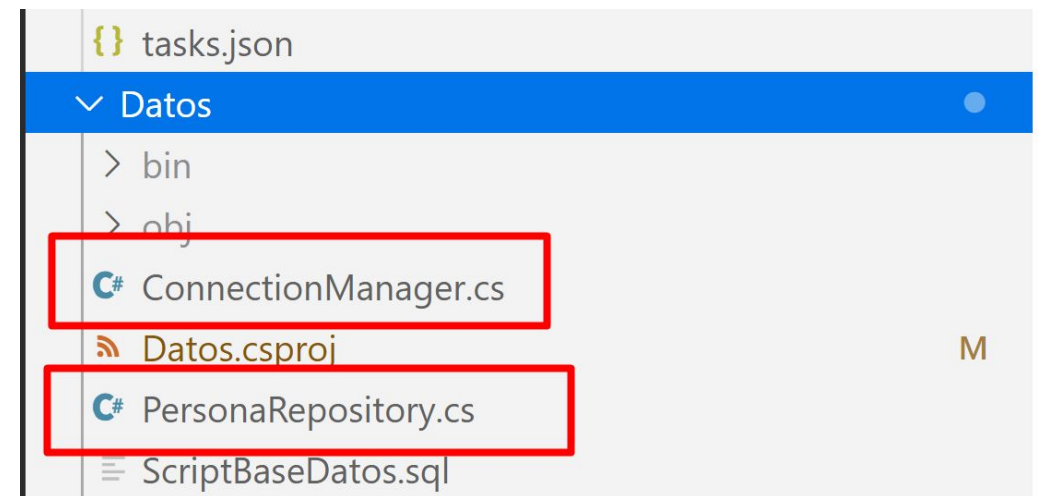
  <ItemGroup>
    <ProjectReference Include="..\Entity\Entity.csproj" />
  </ItemGroup>
</Project>
```

Luego de realizar los Cambios, ejecutar desde el Proyecto de Datos el Comando **dotnet restore**



# Modificar la Capa de Datos

- Eliminar las clases de Conexión y de Repositorio
- Creamos una Clases que representa la base de datos denominada:
- PulsacionesContext



# Modificar la Capa de Datos

```
using Entity;
using Microsoft.EntityFrameworkCore;

namespace Datos
{
    public class PulsacionesContext : DbContext
    {
        public PulsacionesContext(DbContextOptions options) : base(options)
        {
        }
        public DbSet<Persona> Personas { get; set; }
    }
}
```

Agregar la Clase  
PulsacionesContext que  
representará la base de  
datos y el atributo  
DbSet<Persona> Personas  
que hará referencia a la  
tabla

# CAMBIOS AL PROYECTO LOGICA

# Cambios al Proyecto Logica.csproj

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netstandard2.1</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <ProjectReference Include="..\Datos\Datos.csproj" />
    <ProjectReference Include="..\Entity\Entity.csproj" />
  </ItemGroup>

</Project>
```

Luego de realizar los Cambios, ejecutar desde el Proyecto de Logica el Comando **dotnet restore**

# Cambios a la Clase PersonaService

```
using Datos;  
using Entity;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
  
namespace Logica  
{  
    public class PersonaService  
    {  
        private readonly PulsacionesContext _context;  
  
        public PersonaService(PulsacionesContext context)  
        {  
            _context=context;  
        }  
    }  
}
```

# Cambios al Guardar en el Servicio

## Antes

```
public GuardarPersonaResponse Guardar(Persona persona)
{
    try
    {
        persona.CalcularPulsaciones();
        _conexion.Open();
        _repositorio.Guardar(persona);
        _conexion.Close();
        return new GuardarPersonaResponse(persona);
    }
    catch (Exception e)
    {
        return new GuardarPersonaResponse($"Error de la Aplicacion: {e.Message}");
    }
    finally { _conexion.Close(); }
}
```

## Ahora

```
1 reference
public GuardarPersonaResponse Guardar(Persona persona)
{
    try
    {
        var personaBuscada = _context.Personas.Find(persona.Identificacion);
        if (personaBuscada != null)
        {
            return new GuardarPersonaResponse("Error la persona ya se encuentra registrada");
        }
        persona.CalcularPulsaciones();
        _context.Personas.Add(persona);
        _context.SaveChanges();
        return new GuardarPersonaResponse(persona);
    }
    catch (Exception e)
    {
        return new GuardarPersonaResponse($"Error de la Aplicacion: {e.Message}");
    }
}
1 reference
```

Metodo	Sentencia
Guardar	<code>_context.Personas.Add(persona);</code> <code>_context.SaveChanges();</code>
Consultar	<code>List&lt;Persona&gt; personas = _context.Personas.ToList();</code>
Buscar	<code>var persona = _context.Personas.Find(identificacion);</code>
Eliminar	<code>_context.Personas.Remove(persona);</code> <code>_context.SaveChanges();</code>
Modificar	<code>_context.Personas.Update(personaNueva);</code> <code>_context.SaveChanges();</code>



# Metodo Guardar de PersonaService.cs

I reference

```
public GuardarPersonaResponse Guardar(Persona persona)
```

```
{  
    try  
    {  
        var personaBuscada = _context.Personas.Find(persona.Identificacion);  
        if (personaBuscada != null)  
        {  
            return new GuardarPersonaResponse("Error la persona ya se encuentra registrada");  
        }  
        persona.CalcularPulsaciones();  
        _context.Personas.Add(persona);  
        _context.SaveChanges();  
        return new GuardarPersonaResponse(persona);  
    }  
    catch (Exception e)  
    {  
        return new GuardarPersonaResponse($"Error de la Aplicacion: {e.Message}");  
    }  
}
```

Registra el cambios en memoria del Context y  
están disponibles para enviar a la BD

Identifica los cambios en el Context(PulsacionesContext)  
Y genera los scripts para actualizar la BD automáticamente

# Metodo Consultar de PesonaService.cs

1 reference

```
public List<Persona> ConsultarTodos()  
{  
    List<Persona> personas = _context.Personas.ToList();  
    return personas;  
}
```

# Metodo Eliminar de PesonaService.cs

```
public string Eliminar(string identificacion)
{
    try
    {
        var persona = context.Personas.Find(identificacion);
        if (persona != null)
        {
            _context.Personas.Remove(persona);
            context.SaveChanges();
            return ("El registro {persona.Nombre} se ha eliminado satisfactoriamente.");
        }
        else
        {
            return ("Lo sentimos, {identificacion} no se encuentra registrada.");
        }
    }
    catch (Exception e)
    {
        return $"Error de la Aplicación: {e.Message}";
    }
}
```

# Metodo Actualizar de PesonaService.cs

```
public string Modificar(Persona personaNueva)
{
    try
    {
        var personaVieja = _context.Personas.Find(personaNueva.Identificacion);
        if (personaVieja != null)
        {
            personaVieja.Nombre=personaNueva.Nombre;
            personaVieja.Identificacion=personaNueva.Identificacion;
            personaVieja.Sexo=personaNueva.Sexo;
            personaVieja.Edad=personaNueva.Edad;
            personaVieja.CalcularPulsaciones();
            _context.Personas.Update(personaVieja);
            _context.SaveChanges();
            return ($"El registro {personaNueva.Nombre} se ha modificado satisfactoriamente.");
        }
        else
        {
            return ($"Lo sentimos, {personaNueva.Identificacion} no se encuentra registrada.");
        }
    }
    catch (Exception e)
    {
        return $"Error de la Aplicación: {e.Message}";
    }
}
```

# Otras consultas

1 referencia

```
public Persona BuscarxIdentificacion(string identificacion)
{
    Persona persona = _context.Personas.Find(identificacion);
    return persona;
}
```

0 referencias

```
public int Totalizar()
{
    return _context.Personas.Count();
}
```

0 referencias

```
public int TotalizarMujeres()
{
    return _context.Personas.Count(p=>p.Sexo=="F");
}
```

0 referencias

```
public int TotalizarHombres()
{
    return _context.Personas.Count(p => p.Sexo == "M");
}
```

```
}
```



# CAMBIOS EN LA CAPA DE WEB (PRESENTACION)

# Capa de presentacion Webpulsaciones.cs.

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp3.0</TargetFramework>
    <TypeScriptCompileBlocked>true</TypeScriptCompileBlocked>
    <TypeScriptToolsVersion>Latest</TypeScriptToolsVersion>
    <IsPackable>false</IsPackable>
    <SpaRoot>ClientApp\</SpaRoot>
    <DefaultItemExcludes>$(DefaultItemExcludes);$(SpaRoot)node_modules\**</DefaultItemExcludes>
    <!-- Set this to true if you enable server-side prerendering -->
    <BuildServerSideRenderer>false</BuildServerSideRenderer>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="3.1.3">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.AspNetCore.Services.Extensions" Version="3.0.0" />
    <PackageReference Include="Microsoft.Extensions.Logging.Debug" Version="3.0.0" />
    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="3.0.0" />
    <PackageReference Include="Swashbuckle.AspNetCore" Version="5.0.0" />
  </ItemGroup>
```

Luego de realizar los Cambios, ejecutar desde el Proyecto de Web el Comando **dotnet restore**



# Configuración de la conexión en StarUp.Cs

```
...
using Datos;
using Microsoft.EntityFrameworkCore;
using Microsoft.OpenApi.Models;
using System;

namespace WebPulsaciones
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; }
        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            // Configurar cadena de Conexión con EF
            var connectionString=Configuration.GetConnectionString("DefaultConnection");
            services.AddDbContext<PulsacionesContext>(p=>p.UseSqlServer(connectionString));
            services.AddControllersWithViews();
            //Agregar OpenApi Swagger
            services.AddSwaggerGen(c =>
            {

```

Configurando la inyección de **PulsacionesContext**  
Cada que vez que soliciten un objeto  
PulsacionesContext, el Sistema lo inyectará  
Con conexión a SqlServer con la cadena de Conexión  
definida como DefaultConnection  
1 Sola vez para toda la aplicación


WebPulsaciones > {} appsettings.json > {} ConnectionStrings > [abc] DefaultConnection

```
1 {  
2   "Logging": {  
3     "LogLevel": {  
4       "Default": "Warning"  
5     }  
6   },  
7   "AllowedHosts": "*",  
8   "ConnectionStrings": {  
9     "DefaultConnection": "Server=.;Database=Pulsaciones;Trusted_Connection = True; MultipleActiveResultSets = true"  
10  }  
11 }  
12
```

- Recuerda que la Cadena de conexion se toma del archivo appsetting.json

# Capa Web. PersonaController

```
...  
using Datos;  
  
namespace WebPulsaciones.Controllers  
{  
    [Route("api/[controller]")]  
    [ApiController]  
    public class PersonaController : ControllerBase  
    {  
        private readonly PersonaService _personaService;  
  
        public PersonaController(PulsacionesContext context)  
        {  
            _personaService = new PersonaService(context);  
        }  
    }  
}
```



El **startup** automáticamente  
inyectará  
Un objeto Pulsaciones Context  
con la configuración  
determinada

Utilizando el ORM  
Dotnet EF CLI

# REALIZAR MIGRACIONES

<https://docs.microsoft.com/en-us/ef/core/miscellaneous/cli/dotnet>

# Instalando EF CLI

- Las herramientas de la interfaz de línea de comandos (CLI) para Entity Framework Core realizan tareas de desarrollo en tiempo de diseño. Por ejemplo, crean [migraciones](#) , aplican migraciones y generan código para un modelo basado en una base de datos existente.
- Los comandos son una extensión del comando [dotnet](#) multiplataforma , que forma parte del [SDK de .NET Core](#) . Estas herramientas funcionan con proyectos .NET Core.

```
dotnet tool install --global dotnet-ef
```

# Migraciones

- Cuando desarrolla una nueva aplicación, su modelo de datos cambia con frecuencia, y cada vez que **cambia el modelo, se desincroniza con la base de datos.**
- Entity Framework permite crear la base de datos si no existe. Luego, cada vez que cambie el modelo de datos (agregue, elimine o cambie las clases de entidad o cambie su clase de DbContext), puede eliminar la base de datos y EF crea uno nuevo que coincida con el modelo.
- La Migración deberá ser creada en la capa de datos, pero se tomara como proyecto de arranque aquel donde se encuentre la cadena de conexión.

## Ubicados en la carpeta de la capa de datos

### 1. Agregar migración - detección de cambios para la base de datos

- Opcion 1: Desde Package Manager Console de VS

```
PM> add-migration TaskDB
```

- Opcion 2: Desde DotNetCli

```
dotnet ef migrations add InitialCreate -s ../webpulsaciones
```

### 3. Aplicación de migracion - aplicación de cambios en la base de datos

- Opcion 1: Desde Package Manager Console de VS

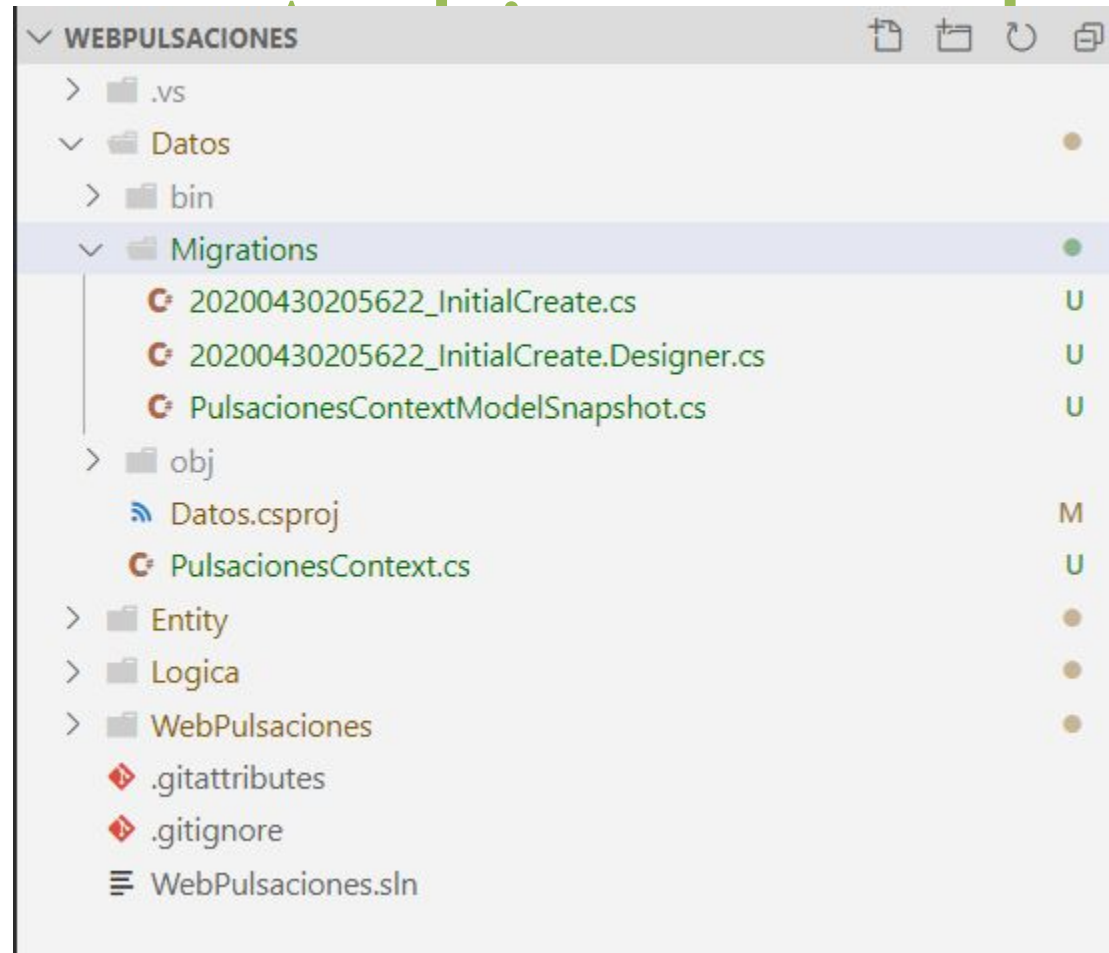
```
PM> update-database
```

- Opcion 2: Desde DotNetCli

```
dotnet ef database update -s ../webpulsaciones
```

-s inidica el Proyecto de arranque donde se encuentra la cadena de conexión base de datos. (startup)





con una migración

## Link Relacionados

- <https://docs.microsoft.com/en-us/ef/core/saving/related-data>
- <https://www.learnentityframeworkcore.com/conventions>
- <https://www.learnentityframeworkcore.com/configuration/data-annotation-attributes>