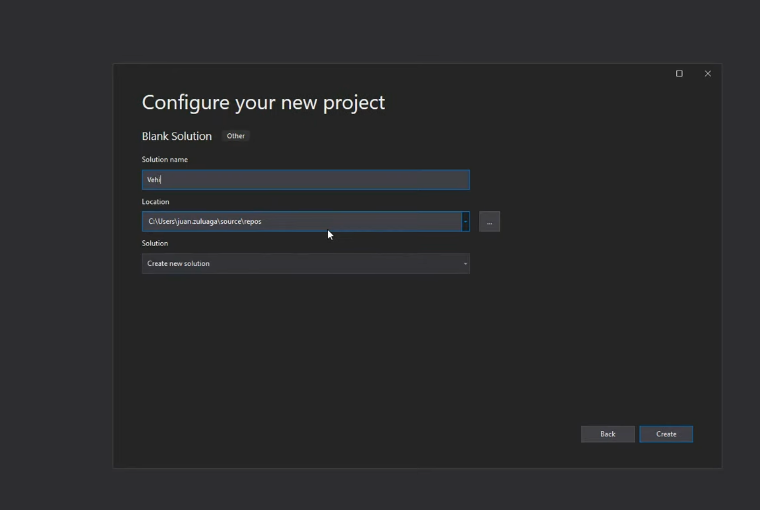
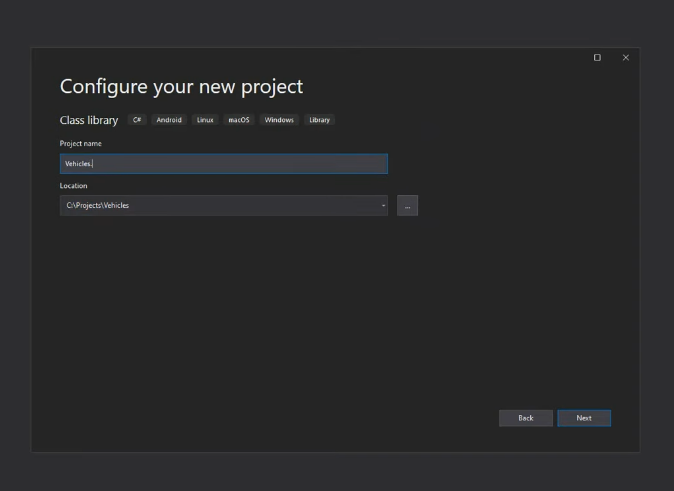
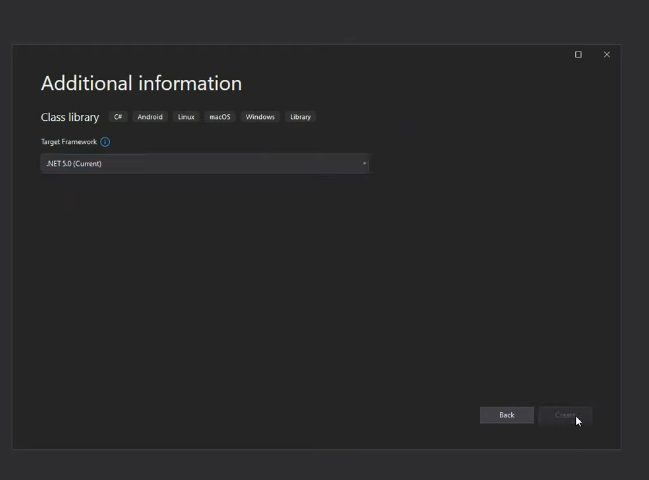
Video 002 https://www.youtube.com/watch?v=PYZaMDmiKX4&list=PLuEZQoW9bRnSXVHOk8WLQVGX3VBTKQKj7&index=3

Se crea una aplicación en blanco

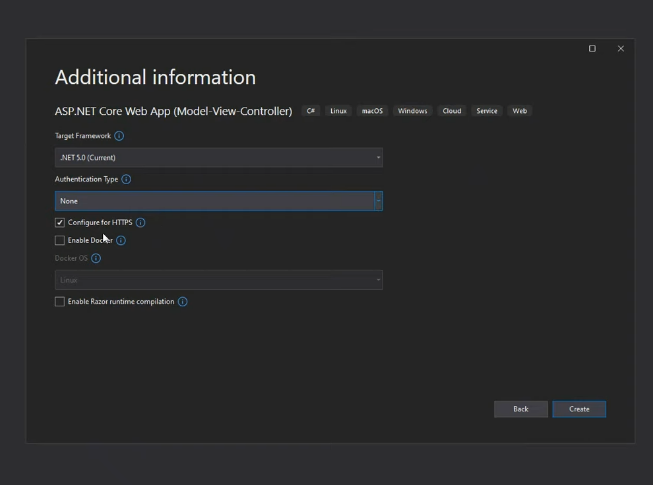
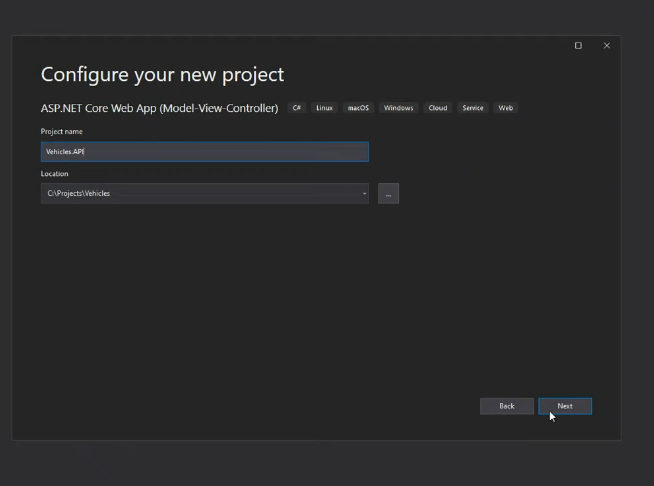
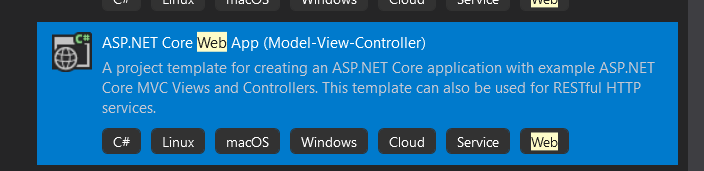


Se crea up proyecto de tipo Class Library para clases compartidas



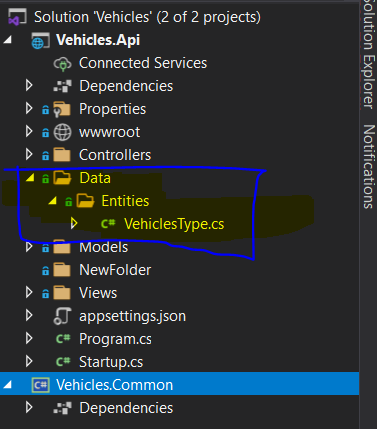
En .NET5.0

Se crea un proyecto de tipo

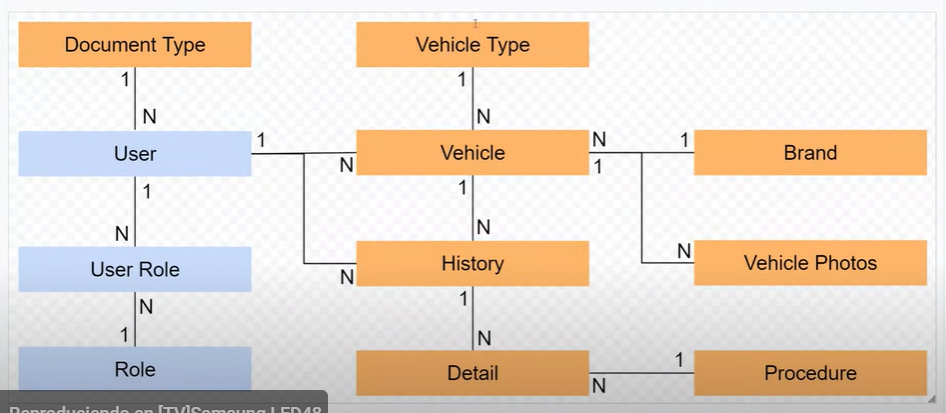


La base de datos s e crea con identity framework

Se crea la siguiente estructura de carpetas

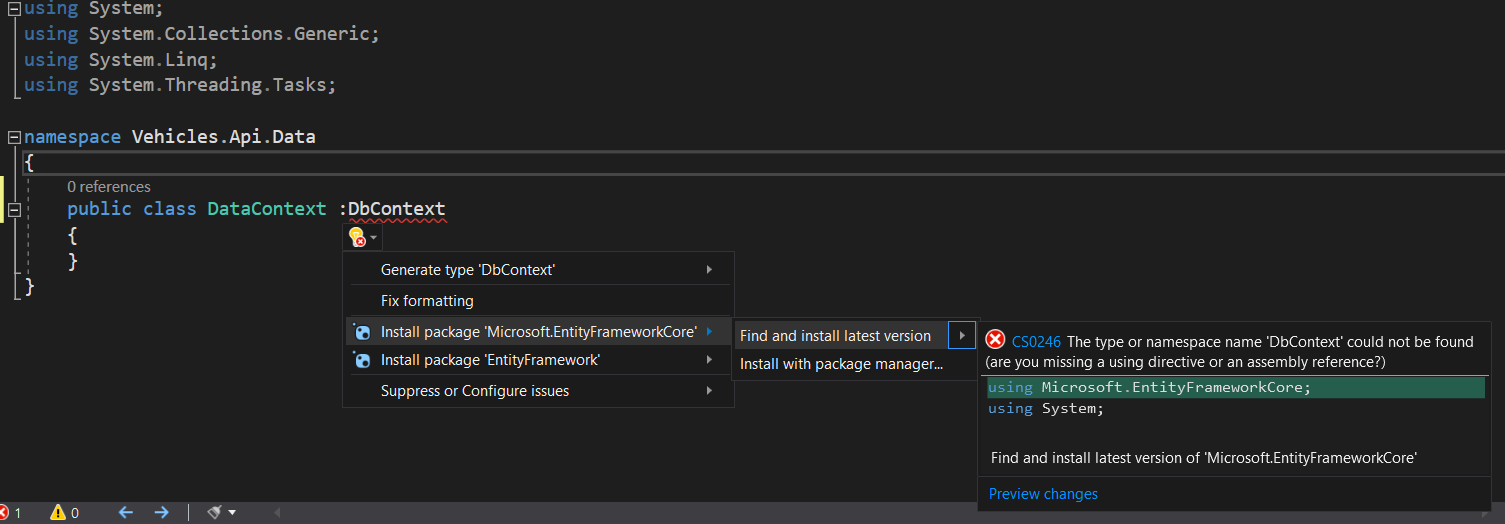


Se crea la clase VehiclesType



Se crea la clase para conectar a la bd DataContext :DbContext

E instalar el paquete de entityframeworkCore



Se modifica el archivo y se agrega la conexión a la bd

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

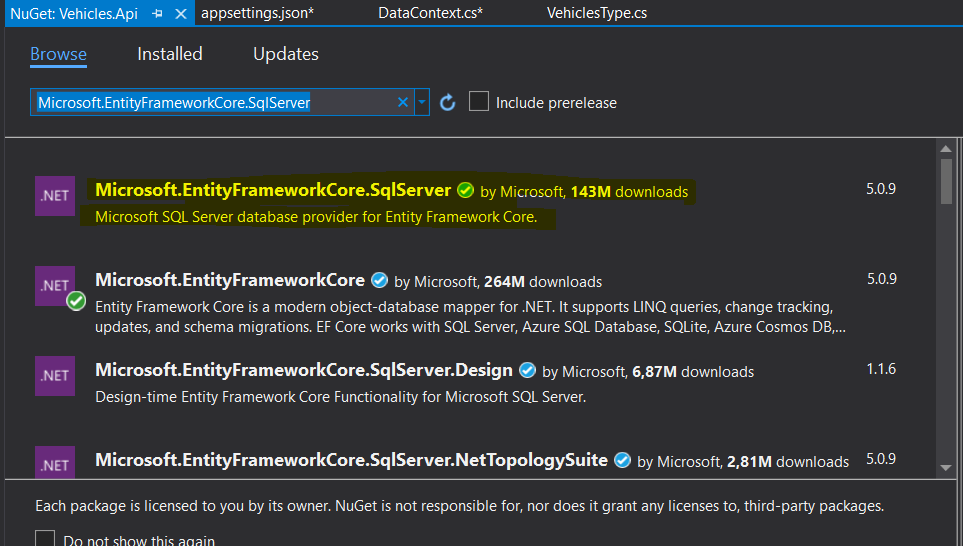
"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=Vehicle;Trusted\_Connection=True;MultipleActiveResultSets=true"

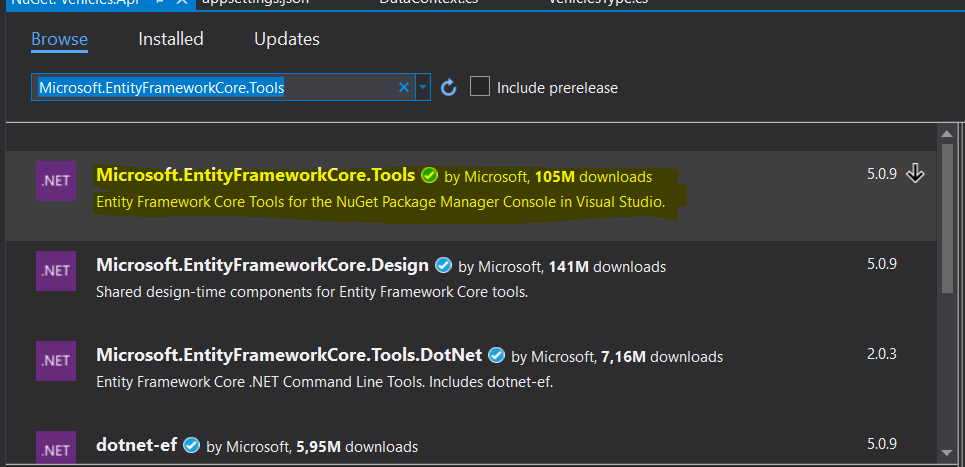
}

}

Se agrega el paquete



Y el paquete



Ahora se va a hacer la inyección de dependencias para la BD

Ene rl archivo **Startup.cs**

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)

{

services.AddControllersWithViews();

//inyeccion de dependencias de la base de datos

services.AddDbContext<DataContext>(x =>

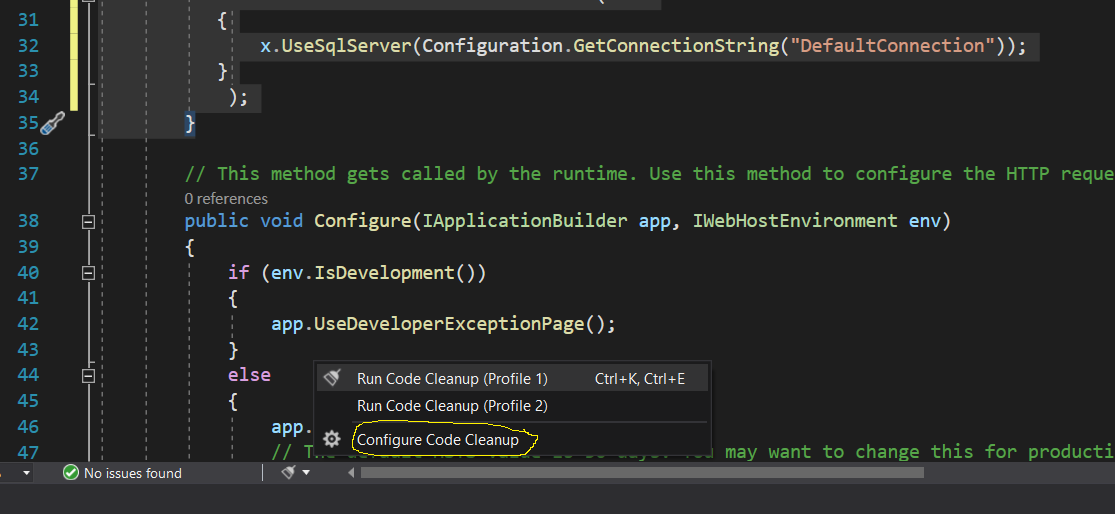
{

x.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));

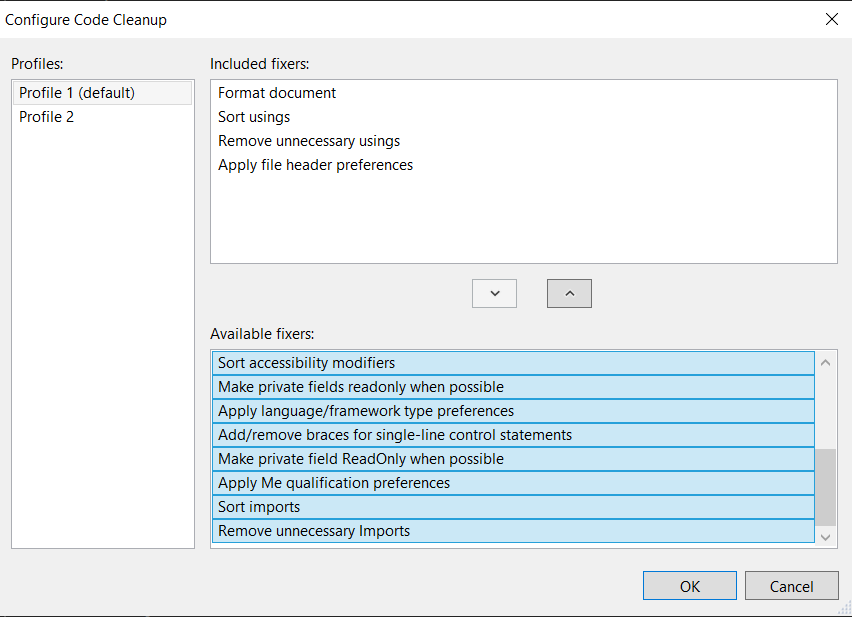
}

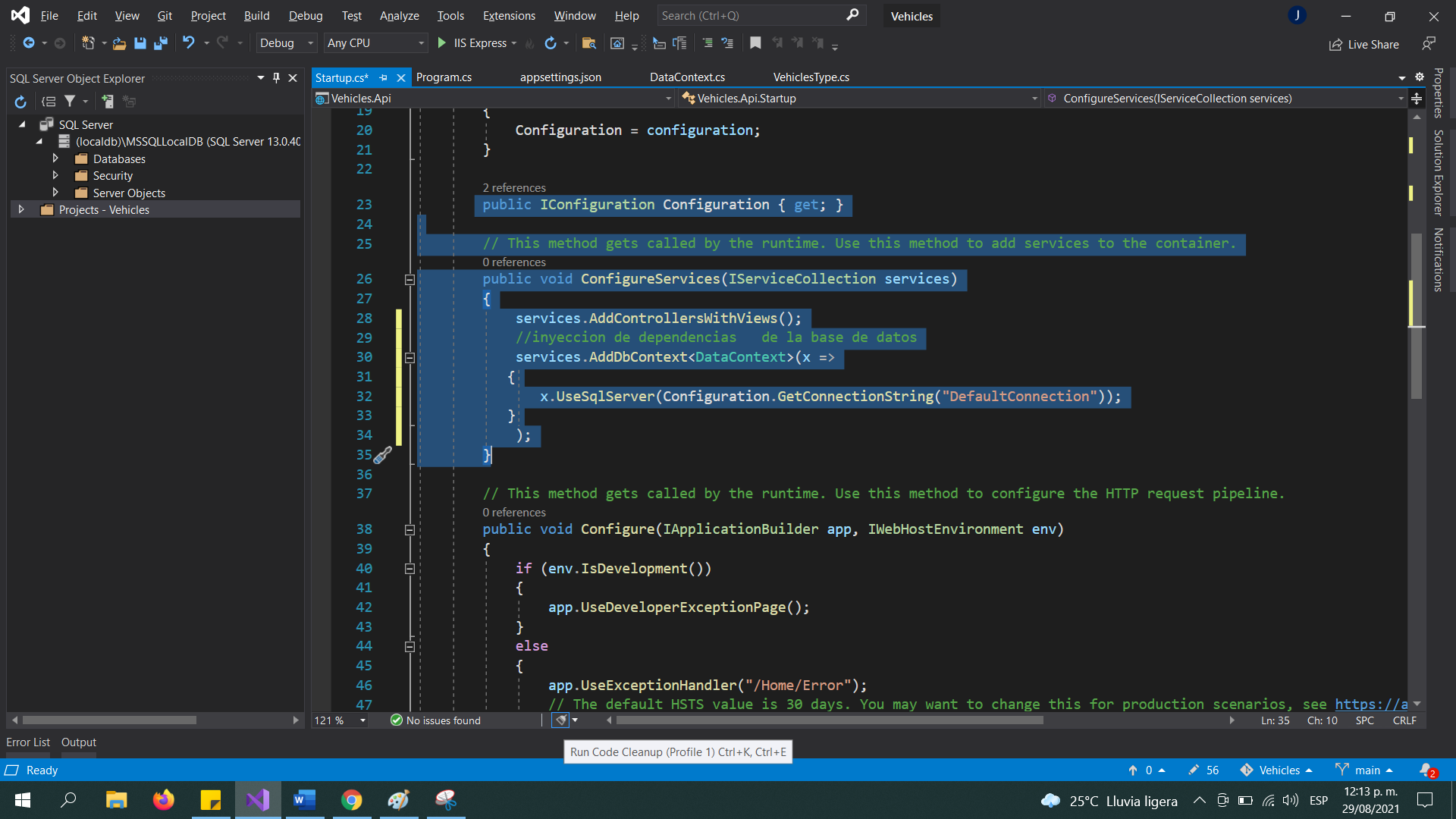
);

}

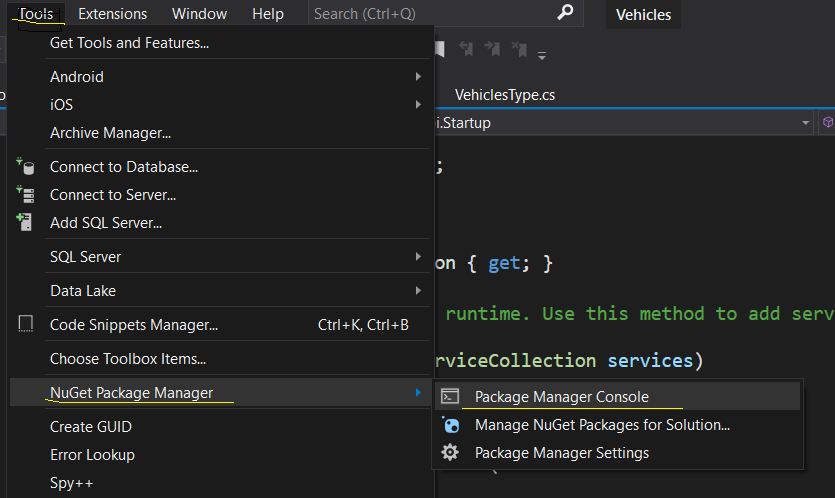
Para crear la tabla se abre 

Y se agregan todos los available fixers

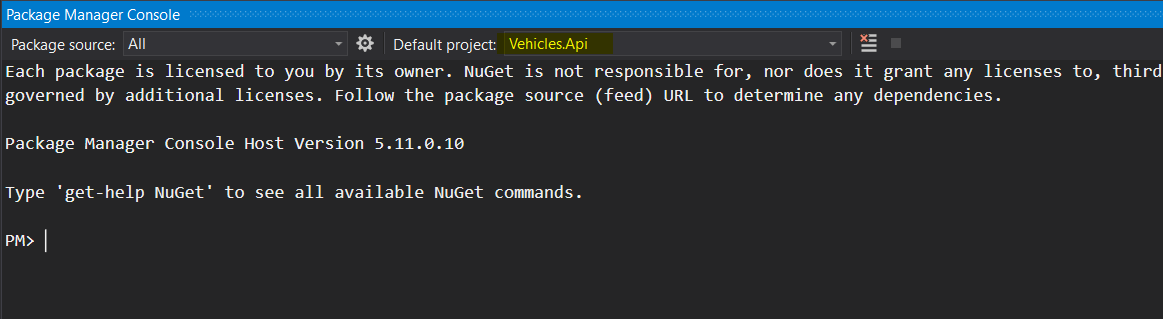




En el **package Manager Console** se corre el siguiente comando

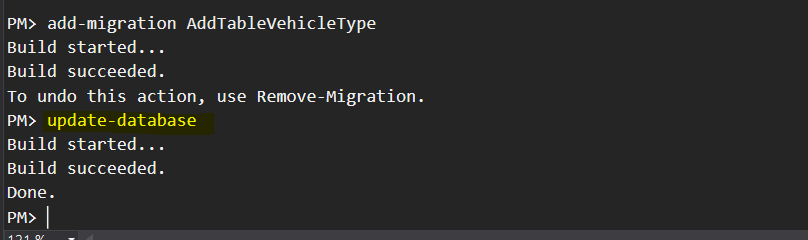


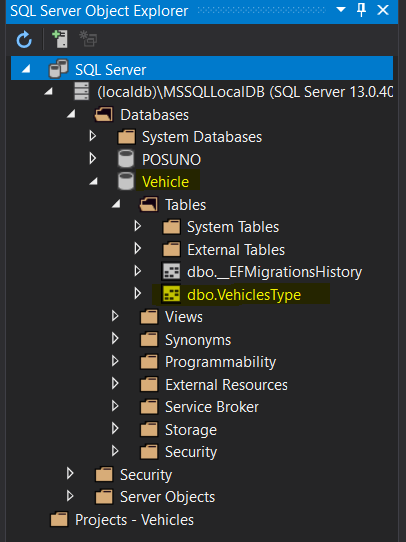
Nota en Deault Project debe estar el api



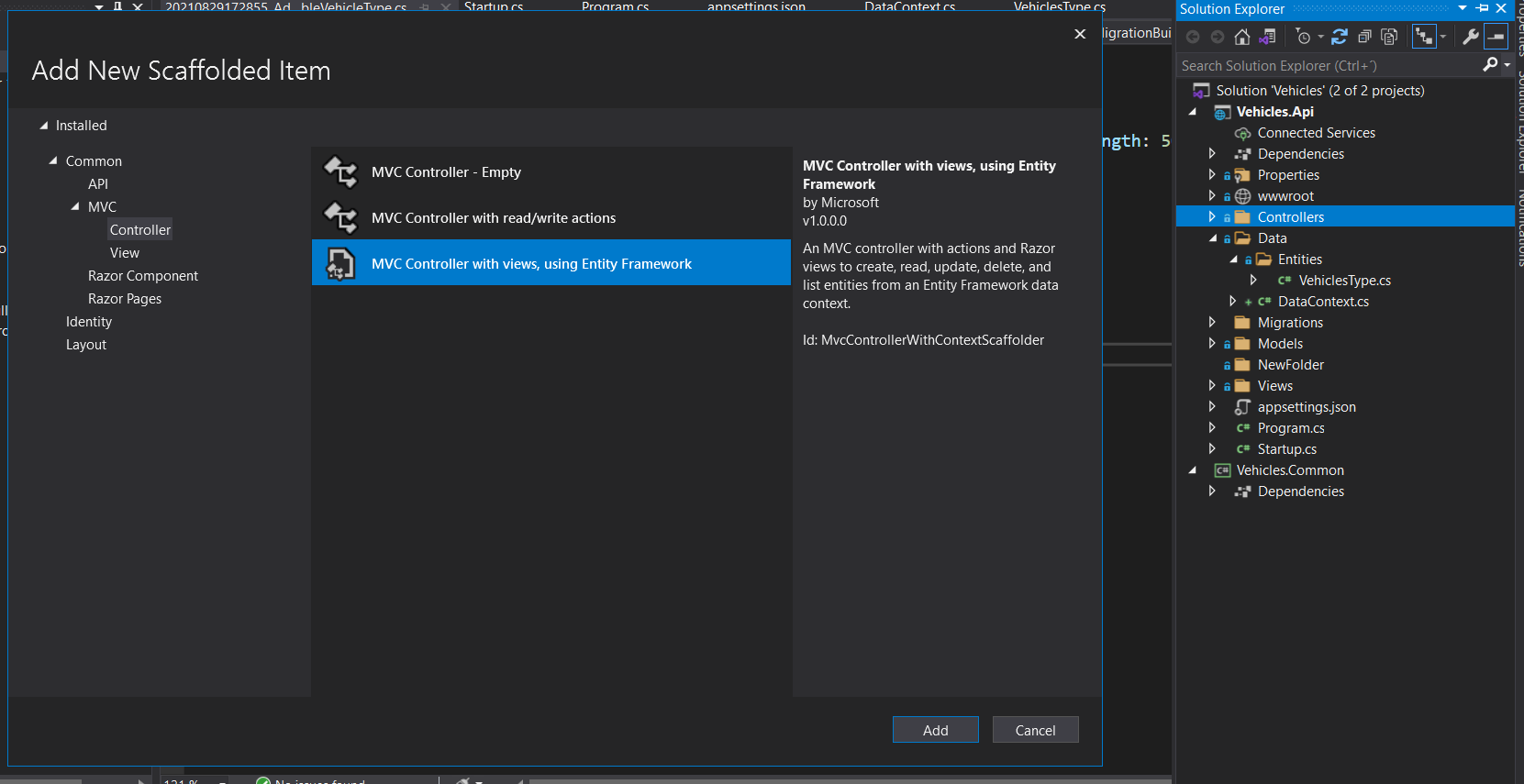
**add-migration Observacion**

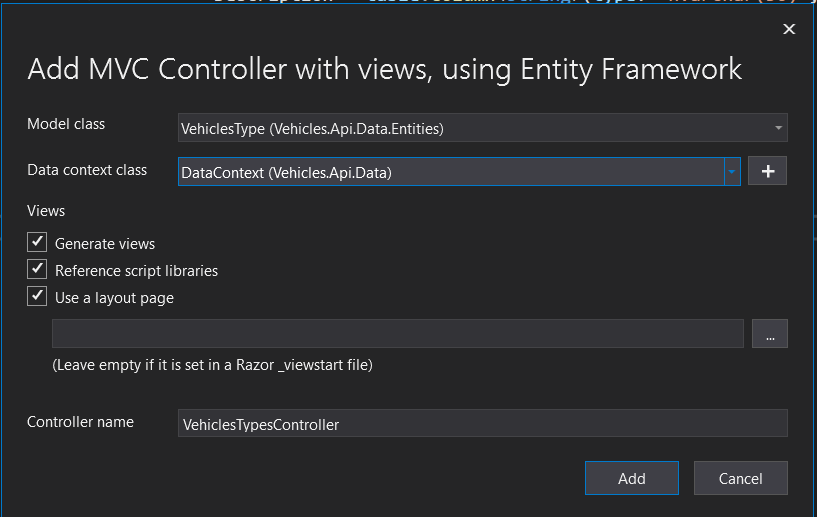
Luego se corre el comando **update-database** para crear la bd



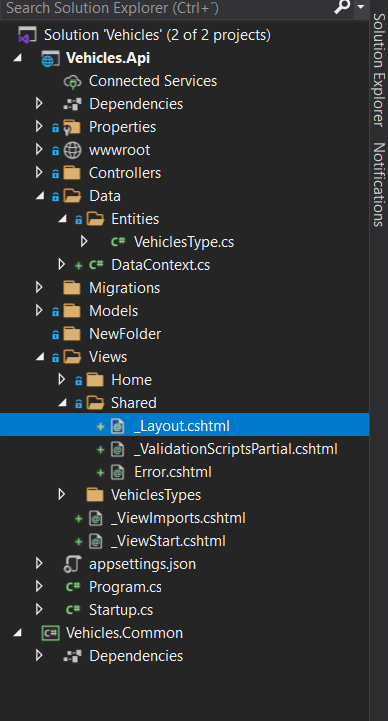
Y en los objetos de sql server aparece la bd con la tabla creada

Se crea el controlador para vehiclestype





Se modifica el archivo \_Layout.cshtml



* se crea la entity
* Se incluye la clase en el datacontext y se crea la llave única
* Se corre el comando add-migration Observacion
* Se actualiza la bd **update-database**
* **Se crea el controlador**
* **Se crea el index**
* **Agregarla pagina al menú en el \_Layout.**
* **Se crea el créate**
* **Se crea la vista parcial**
* **Se crea el editar**
* **Se crea el borrar**

# Crear BD y poblarla video 5

Alimentador de base de datos se crea la clase **SeedDb**

Luego se debe inyectar en el **Startup** para que funcione con **AddTransient** porque se va a ejecutar una sola ves.

Se modifica program.cs por que para inyectar **SeedDb** cuando se ejecuta aun no están listas las inyecciones de dependencias

# Tablas de usuario video 6

Primero crear entidad Usuarios que va a heredar de IdentityUser

Para tablas relacionadas se debe crear el tipo icollection en la entidad relacionada

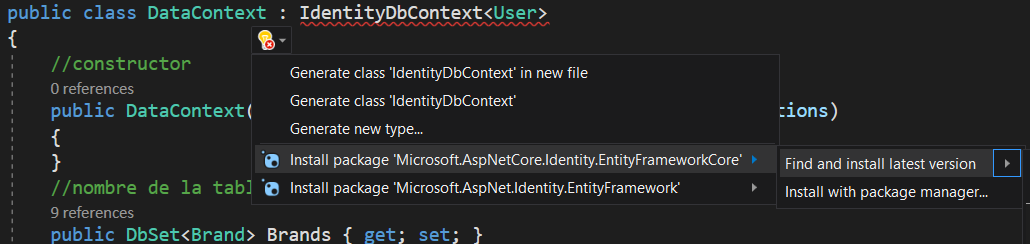
(Ver entidad DocumentType public ICollection<User> Users { get; set; })

Se agrega en cambia el data context de

public class DataContext : DbContext

a

public class DataContext : IdentityDbContext<User>

se debe instalar 

como los usuarios los maneja el framework se deben crear una interface para inyectarlos

para esto se crea la carpeta [Helpers](https://github.com/Zulu55/Vehicles/tree/main/Vehicles.API/Helpers)

y una interface [IUserHelper.cs](https://github.com/Zulu55/Vehicles/blob/main/Vehicles.API/Helpers/IUserHelper.cs), siempre verificar que sea public

luego se debe implementar la interface con la clase public class UserHelper :IUserHelper

luego en el startup se agrega la configuración de usuarios

# video 7 crear plantilla de login

1. se crea en la carpeta model la clase LoginViewModel para el login
2. se agregan las propiedades con sus respectivas DataAnnotations
3. se agregan en lal interface IUserHelper los métodos LoginAsync() y LogoutAsync()
4. se implementa la interface en UserHelper
5. se inyecta, SignInManager<User> signInManager
6. se crea un controlador para el Loguin [AccountController.cs](https://github.com/Zulu55/Vehicles/blob/main/Vehicles.API/Controllers/AccountController.cs)
7. se agrega la vista login
8. en el controlador se crean los métodos Login() y Logout()
9. se modifica la vista index del home
10. se agrega la marca [Authorize(Roles = "Admin")] para que solo el administrador pueda usar este controlador
11. se modifica el layout para que muestre el logaoout y que muestre las opciones solo para los usuarios autorizados