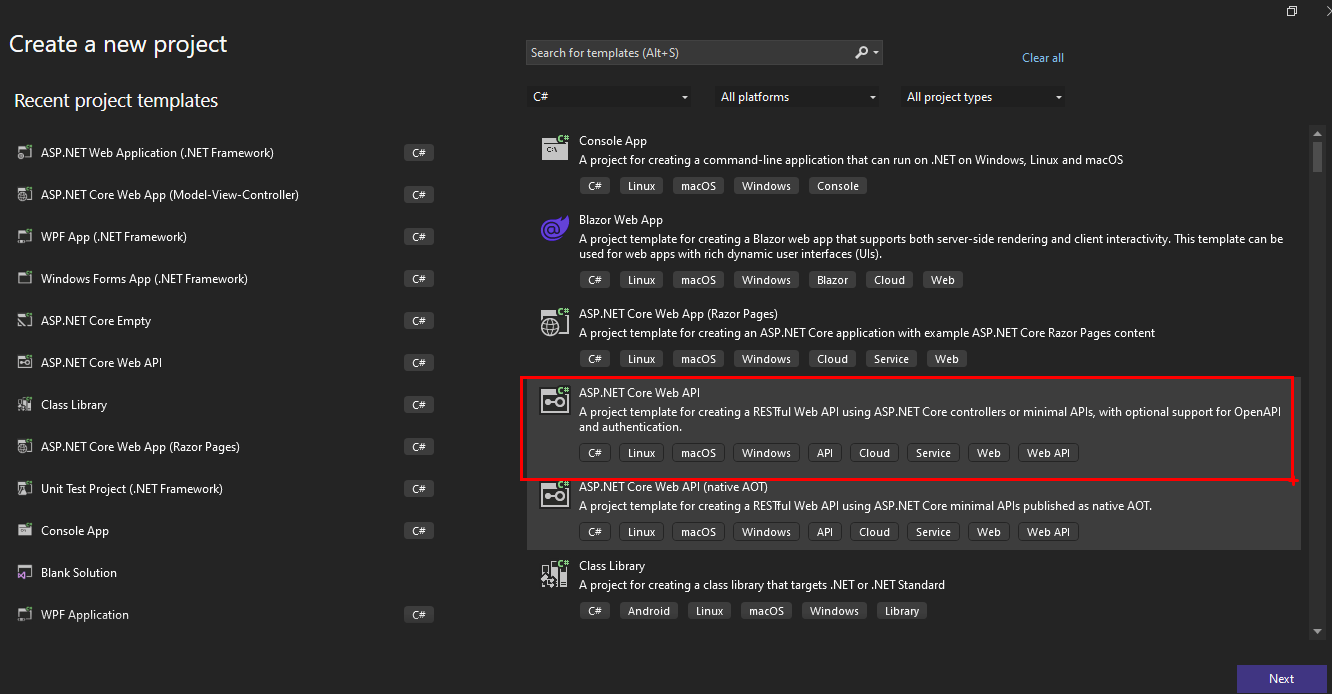
**RELATIONSHIP WITH DATA ANOTATIONS**

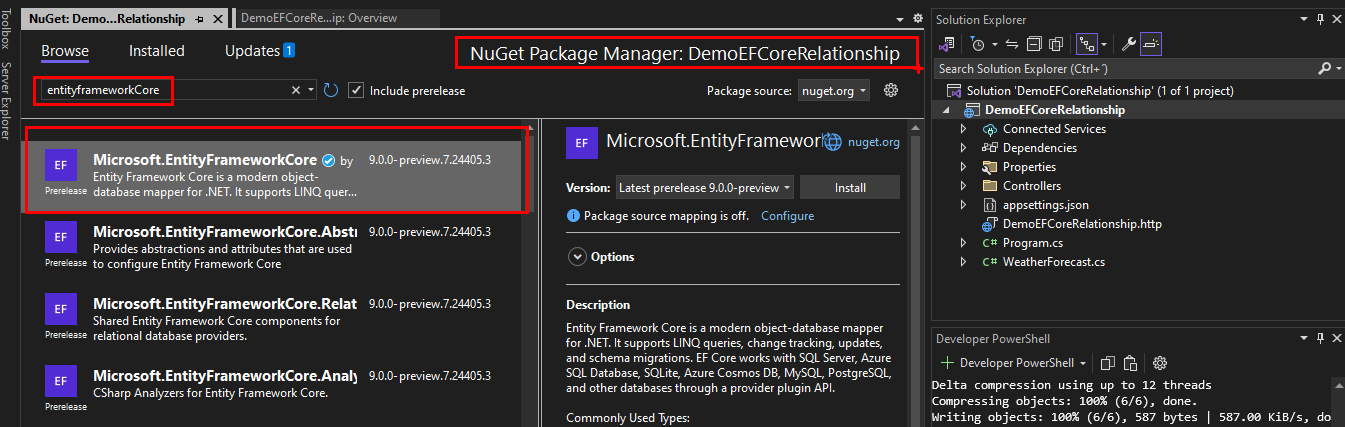
**ONE-TO-ONE**

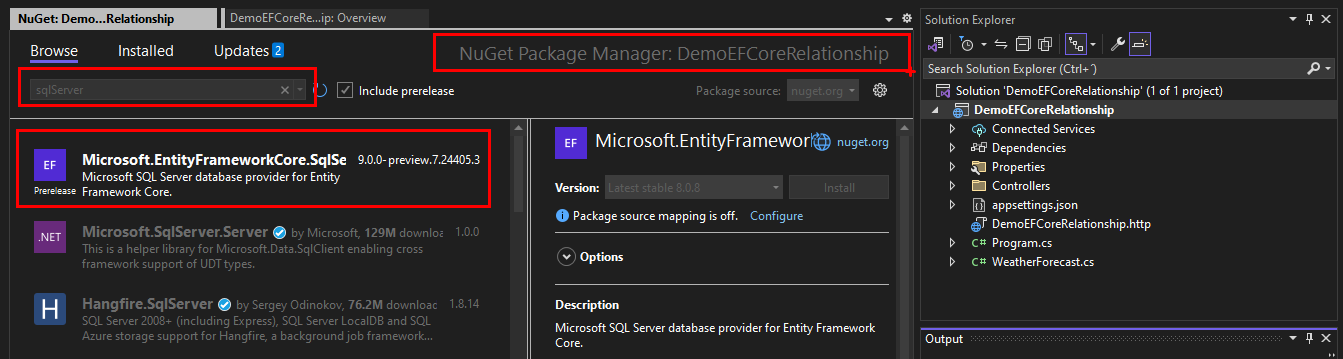
Creamos un proyecto de tipo ASP NetCore Web API

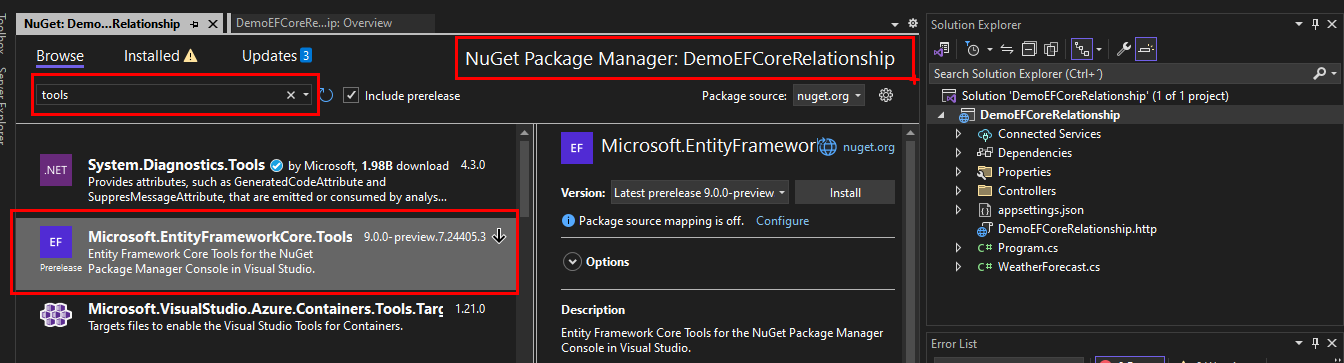
Le ponemos de nombre **DemoEFCoreRelationship**



Procedemos a instalar los siguientes paquetes:



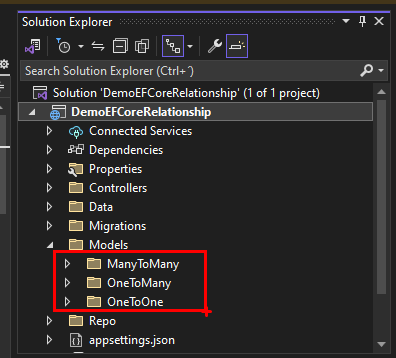




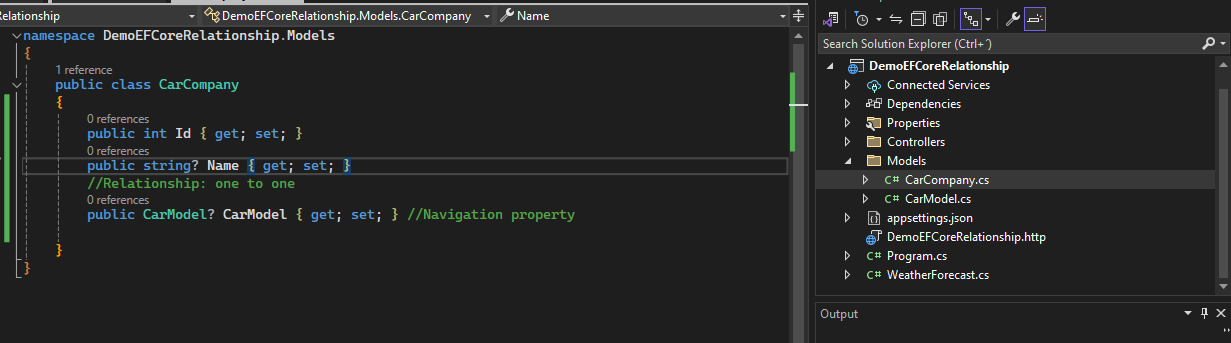
Ahora hacemos un **build** a nuestro proyecto

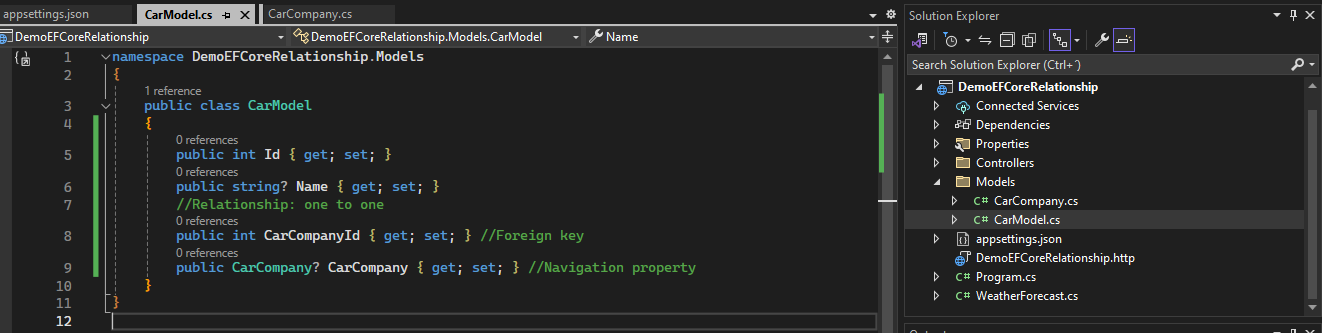
Lo siguiente que hacemos es crear una carpeta **Models** dentro de la cual crearemos nuestras entidades como clases

Asu vez crearemos una carpeta para especificar cada tipo de relación

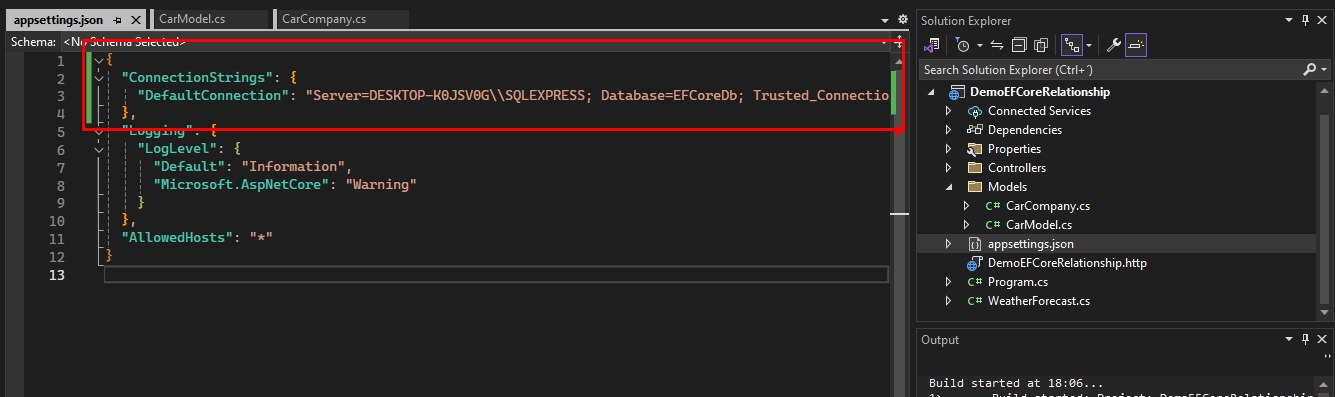


Empezaremos con nuestra relación de **one-to-one** y procedemos a crear nuestras clases

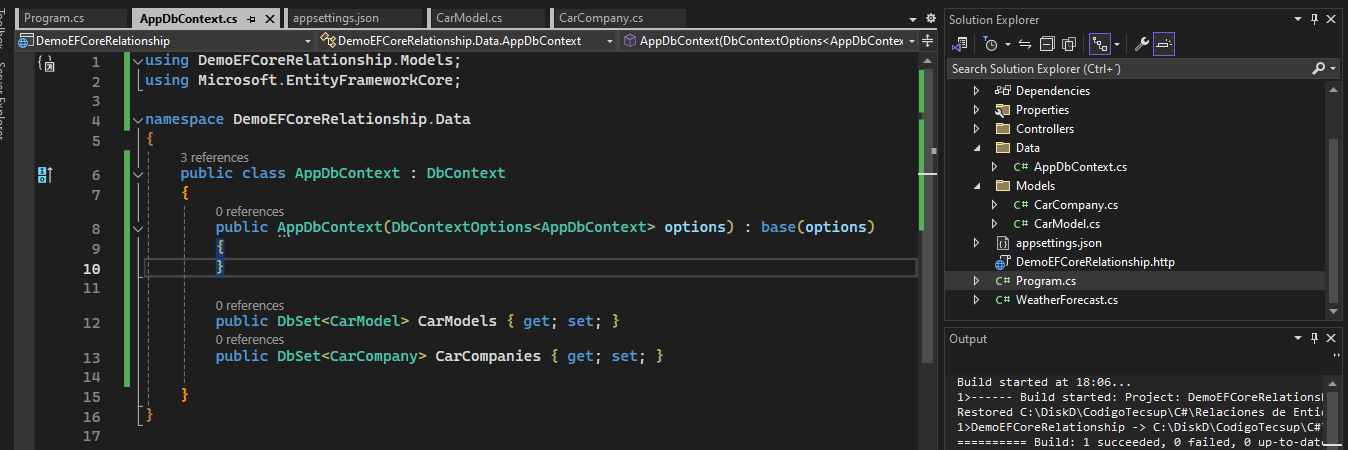




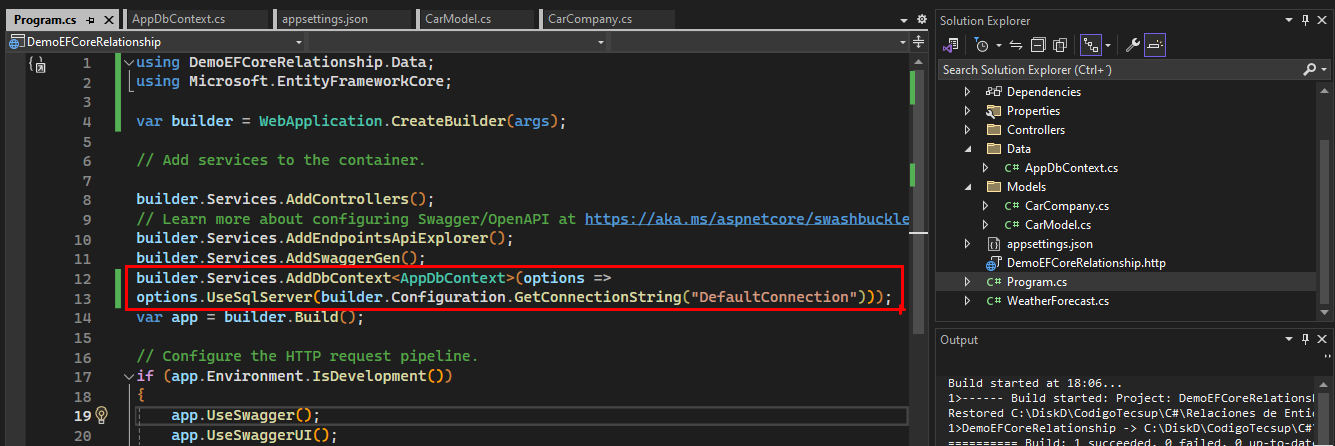
Lo siguiente que hacemos es agregar nuestra conexión en el **appSettings.json**

****

Lo siguiente que haremos será crear nuestro **AppDbContext** que es donde mapearemos todas nuestras entidades para que puedan ser representadas en nuestra base de datos:



Y en nuestro **program.cs** agregamos nuestra clase **AppDbContext:**

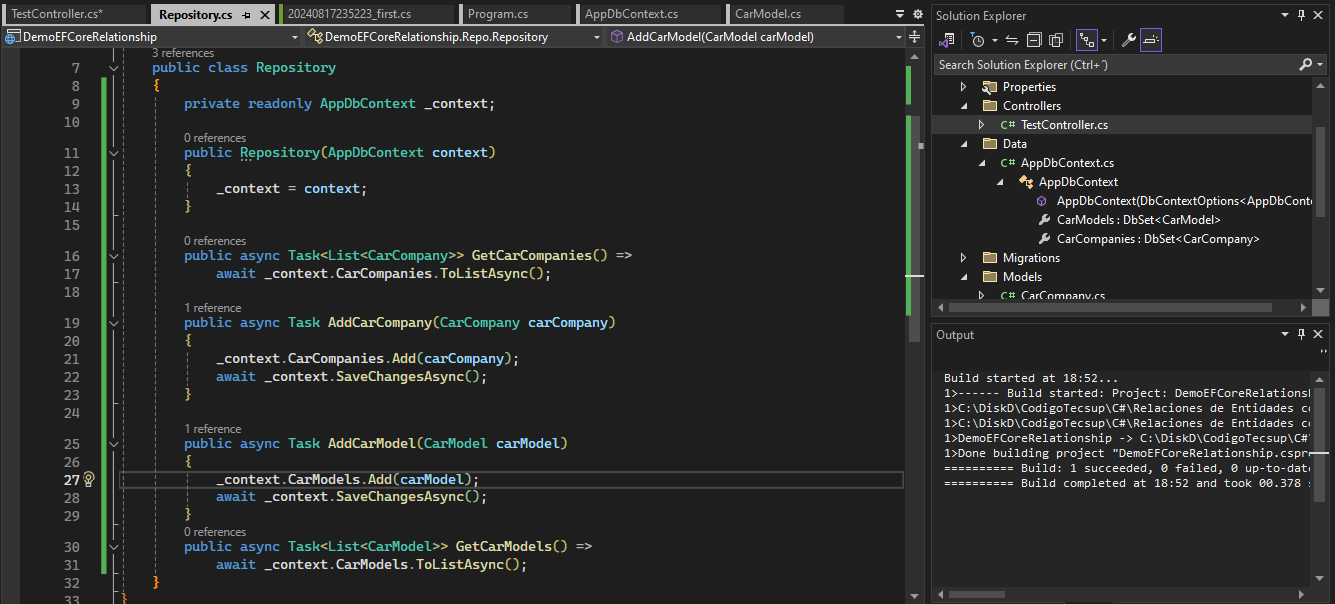
****

Ahora nos ubicamos en nuestro **package manager console** para ejecutar la migración

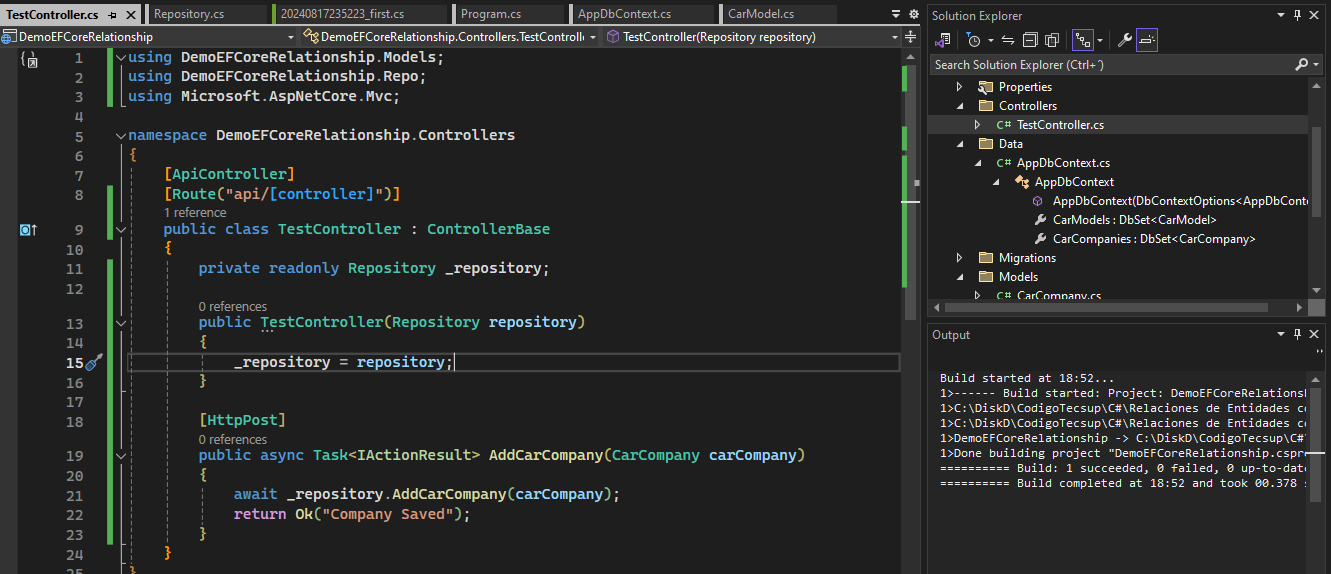
Y le damos a **update-database**

Analizamos y verificamos que se ahigan creado correctamente y analizamos las relaciones **one-to-one**

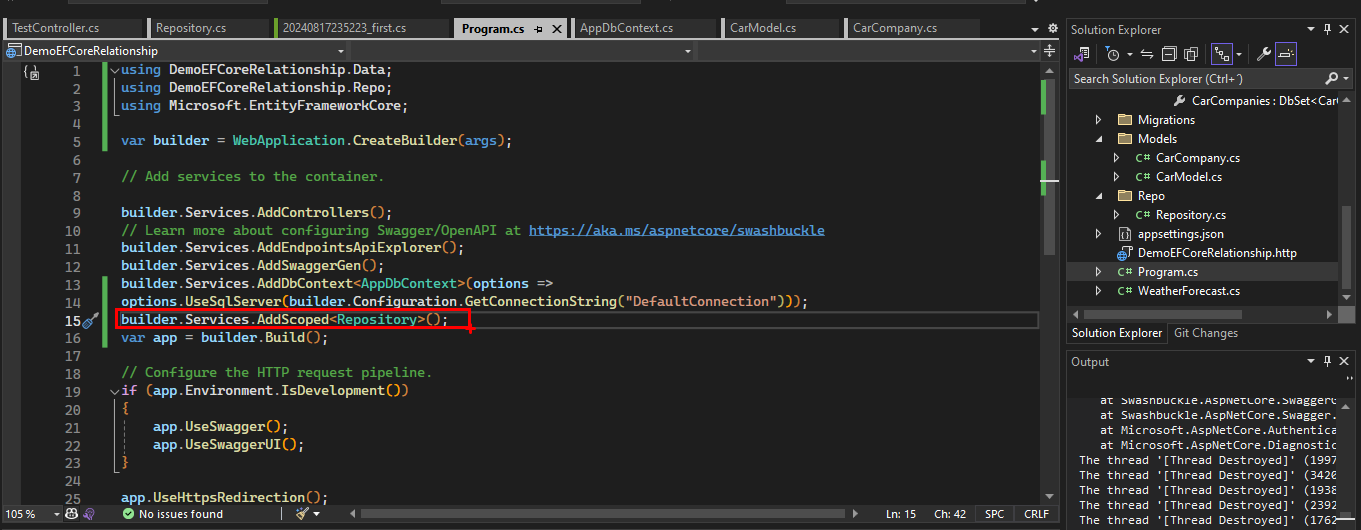
Ahora vamos a crear una clase **Repository** dentro de la carpeta **Repo** donde obtenedremos los métodos propios de un repositorio



Y lo siguiente que hacemos es crear nuestro controller de nombre **TestController**

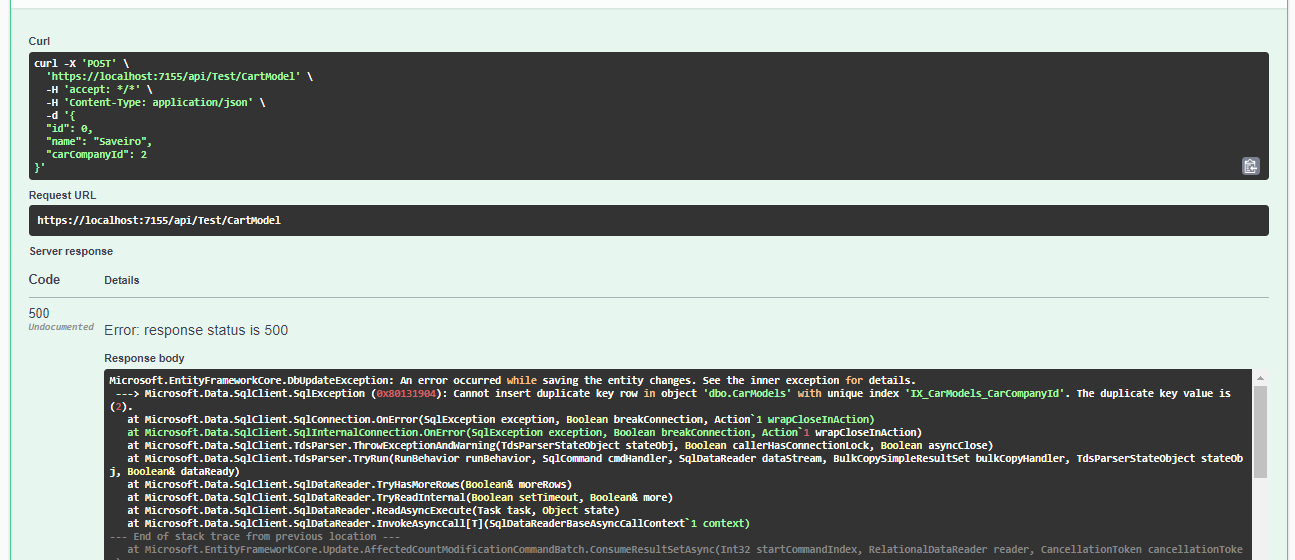


Luego de implementar nuestro controller tenemos que regístralo en nuestro **program.cs:**

****

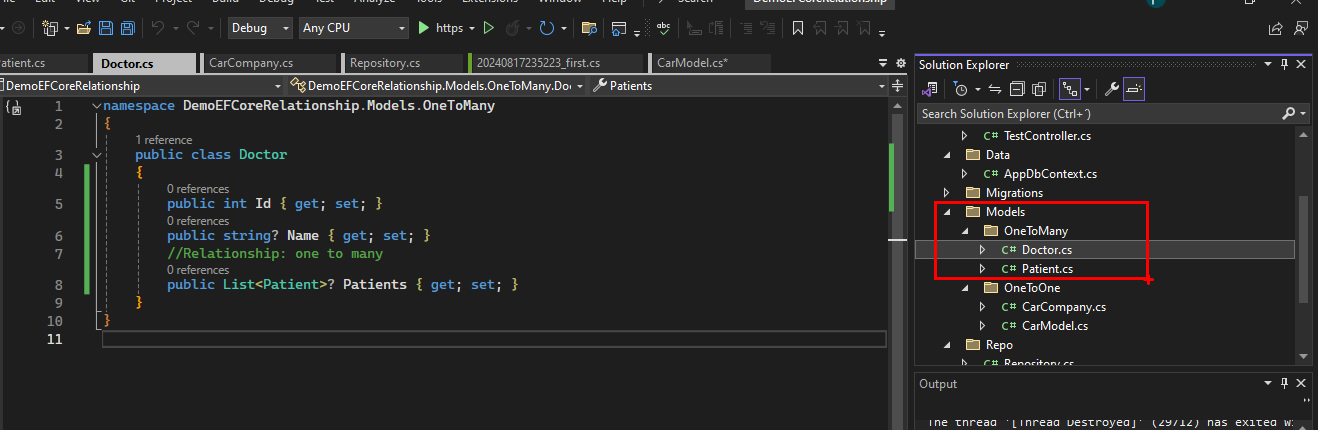
Ejecutamos nuestro proyecto y hacemos pruebas para evidenciar la relación de **one-to-one**

Debería aparecernos este error:



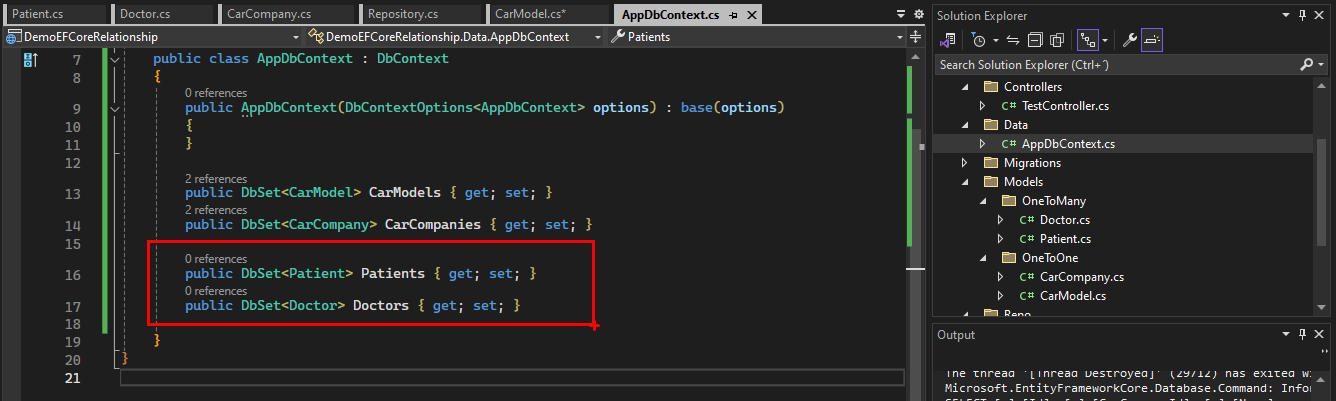
**ONE-TO-MANY**

Lo primero que hacemos es crear nuestras clases de **Doctor** y **Pattient**

****

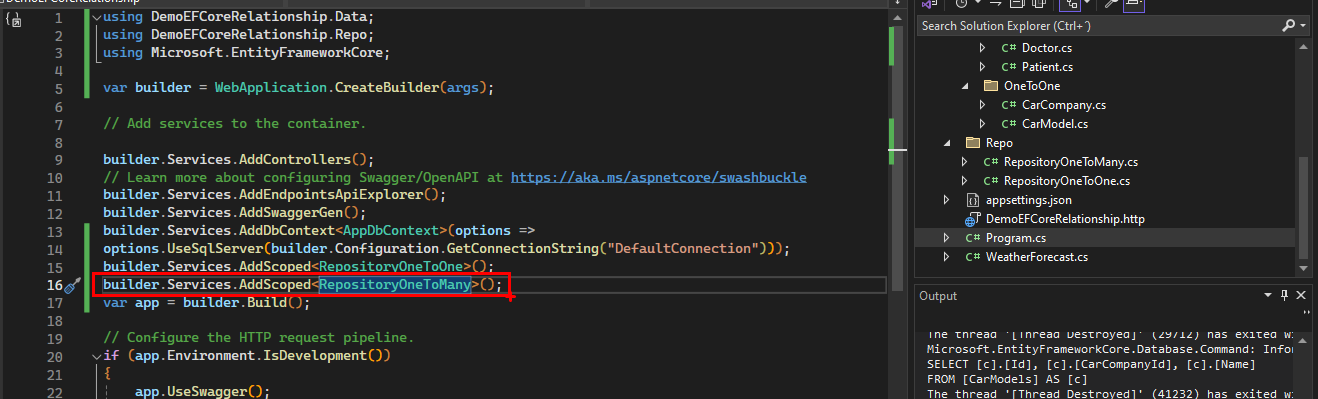
Y establecemos las relaciones en este caso de **one-to-many**

Lo siguiente que vamos a hacer es agregarlo en nuestro **AppDbContext:**

****

Lo siguiente que haremos es crear nuestro repositorio de **RepositoryOneToMany**

Y lo agregamos en nuestro **Program.cs**

****

Procedemos a agregar la migración de estas dos nuevas tablas:

Y actualizamos la base de datos con **update-database**

Ahora procedemos a implementar nuestro repositorio **One-To-Many**

 **Un Doctor puede tener muchos Pacientes** → List<Patient> en Doctor.

 **Cada Paciente pertenece a un solo Doctor** → DoctorId en Patient como clave foránea.

**Relación de muchos a muchos:**

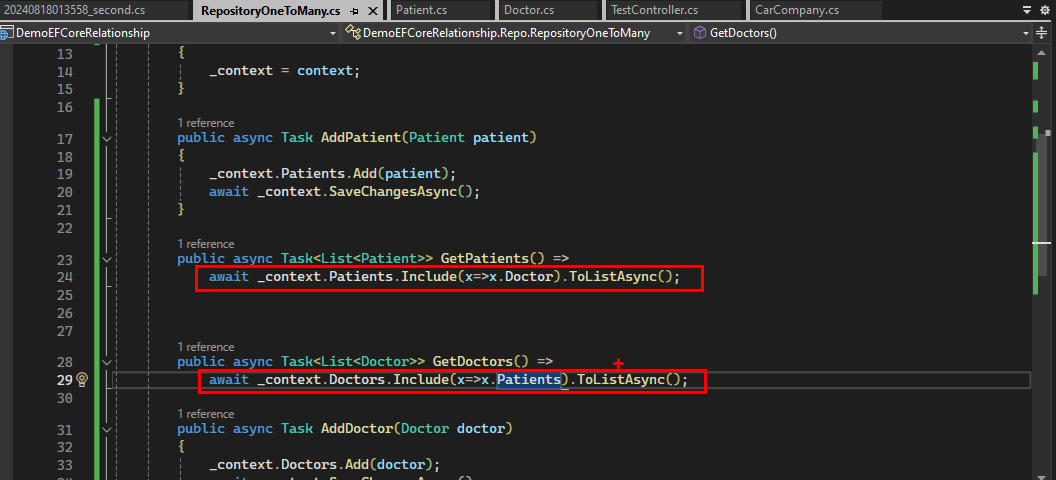
**Entity Framework Core (EF Core) crea automáticamente la tabla intermedia** cuando configuras una relación **muchos a muchos** usando colecciones de navegación en ambas entidades, sin definir explícitamente una entidad intermedia.

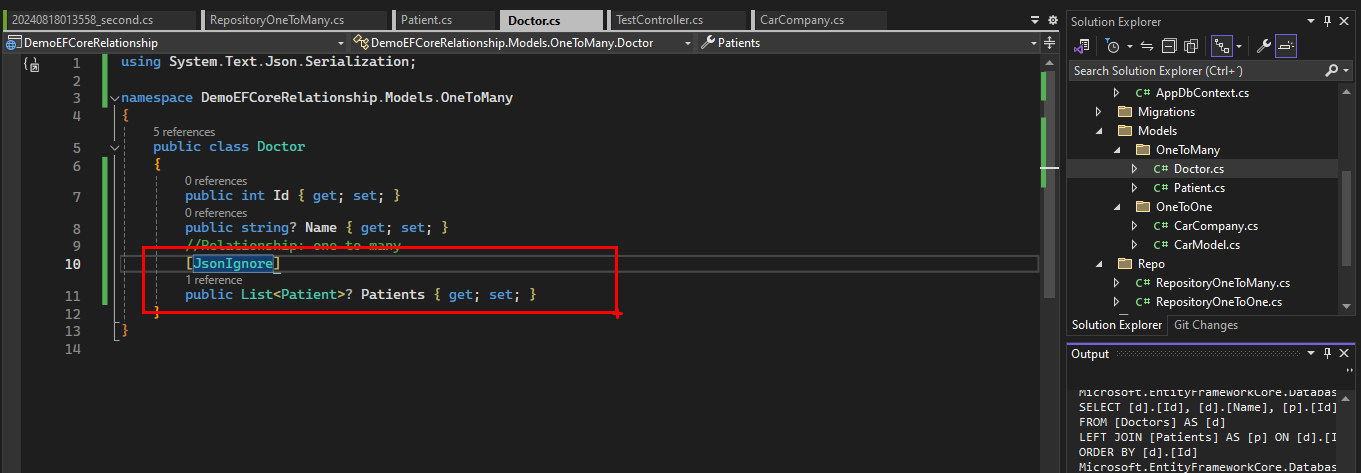
Si quieres **control total** sobre la tabla intermedia (por ejemplo, agregar campos adicionales como FechaRegistro), necesitas definir una **entidad intermedia explícita**.

Seguidamente procedemos a implementar los endPoints para nuestro nuevo repositorio

Verificamos que todo este correcto en nuestra base de datos y hacemos pruebas en nuestro **Swagger** **recordar que es una relación de uno a muchos**

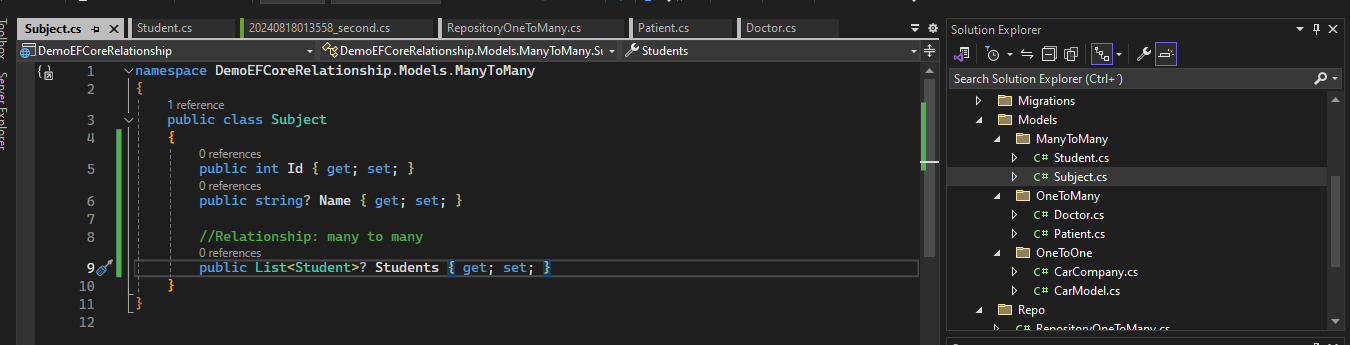
**Ahora si queremos que se muestre nuestra propiedad de navegación utilizamos el include:**

****

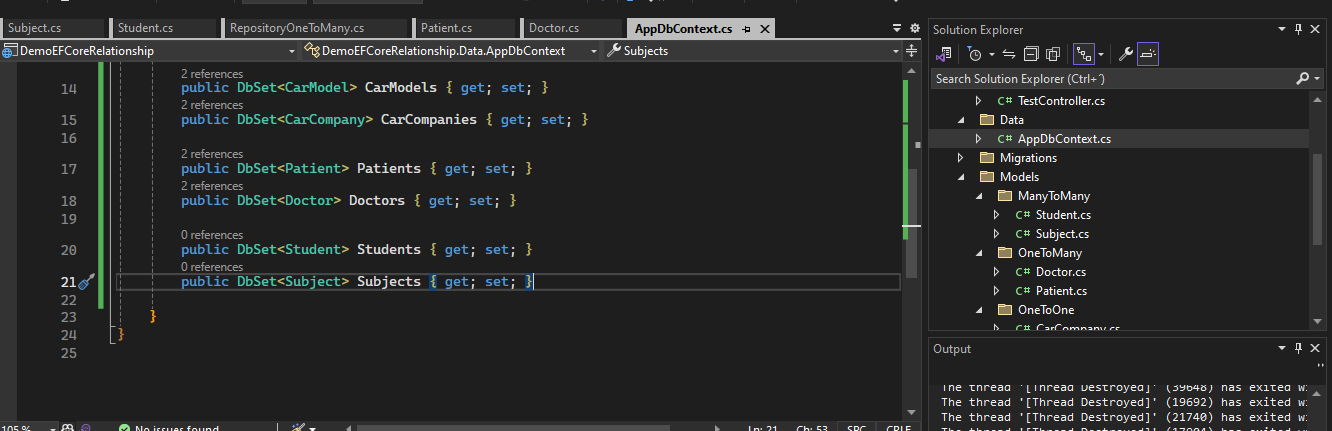


**Many-To-Many**

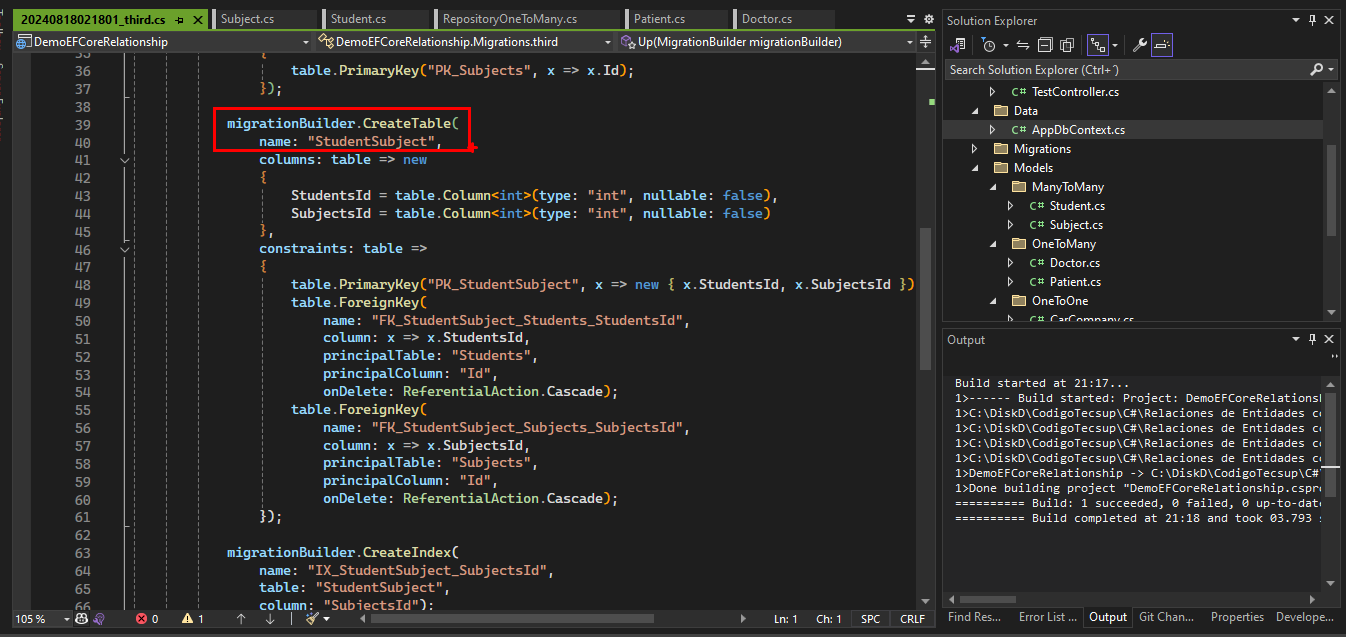
Creamos nuestras clases de **Student**  y **Subjects**

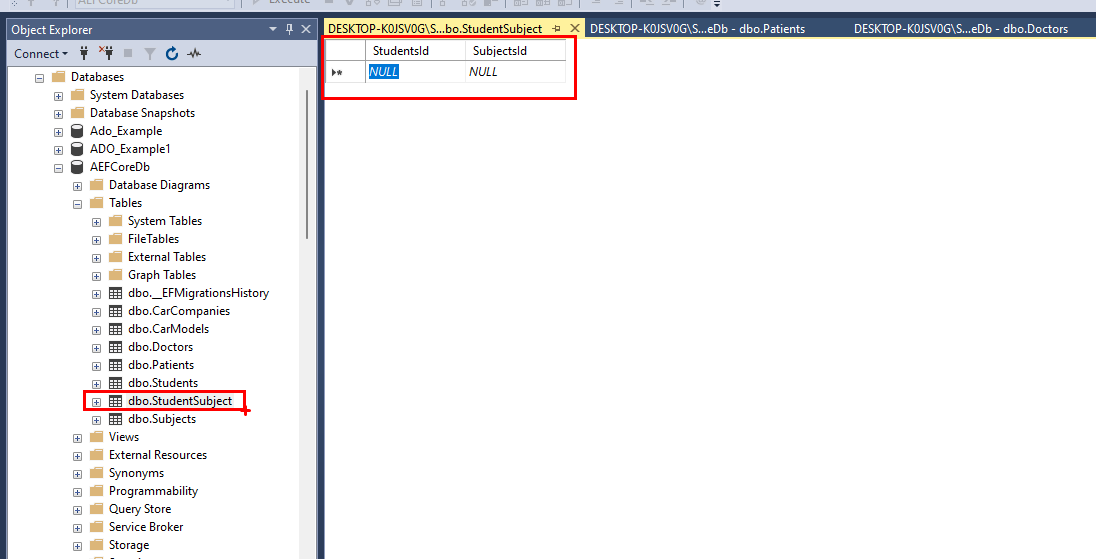
****

Agregamos las clases a nuestro **AppDbContext**  y ejecutamos nuestra migración



Y vemos que la migración nos ha creado nuestra tercera tabla que es para hacer las respectivas relaciones, **LA ANALIZAMOS:**

****



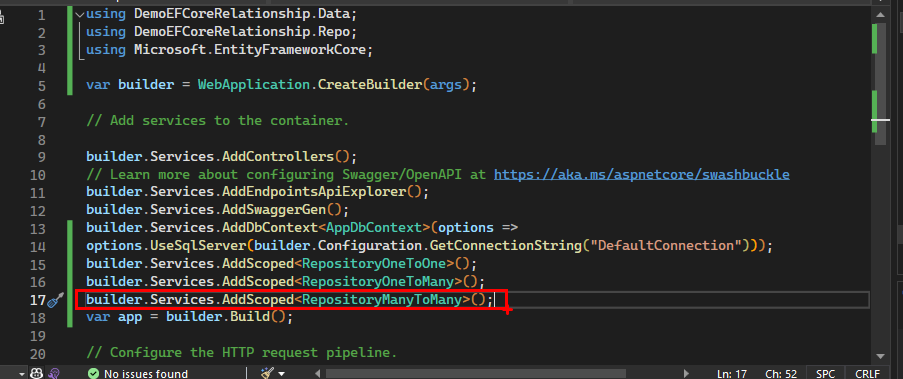
Ahora procedemos a crear nuestro repositorio de **many-to-many**

Y agregamos nuestro repositorio a nuestro **Program.cs:**

Por ultimo procedemos a agregar nuestros endPoints:

Ahora tenemos que tener en cuenta que para una relación de muchos a muchos es necesario primero insertar en nuestro caso primero los **students** y los **subjects**

Primero, se debe crear una clase que represente la tabla intermedia. Esta clase tendrá dos propiedades que actuarán como claves foráneas para Student y Subject.

****