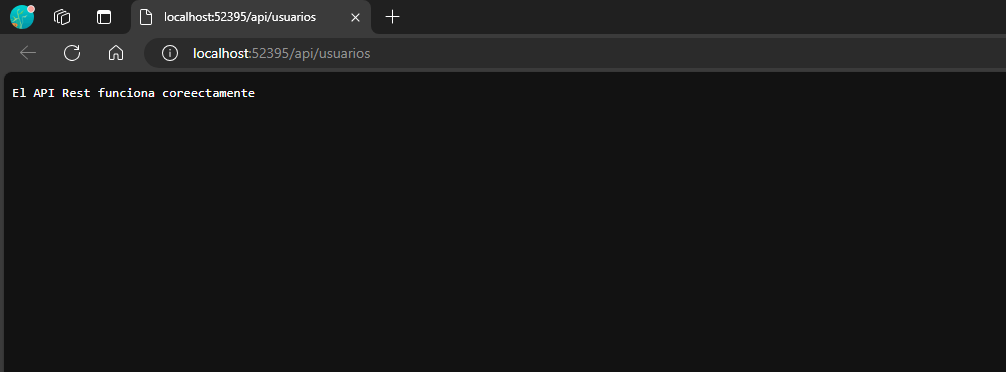
**Documentación API REST**

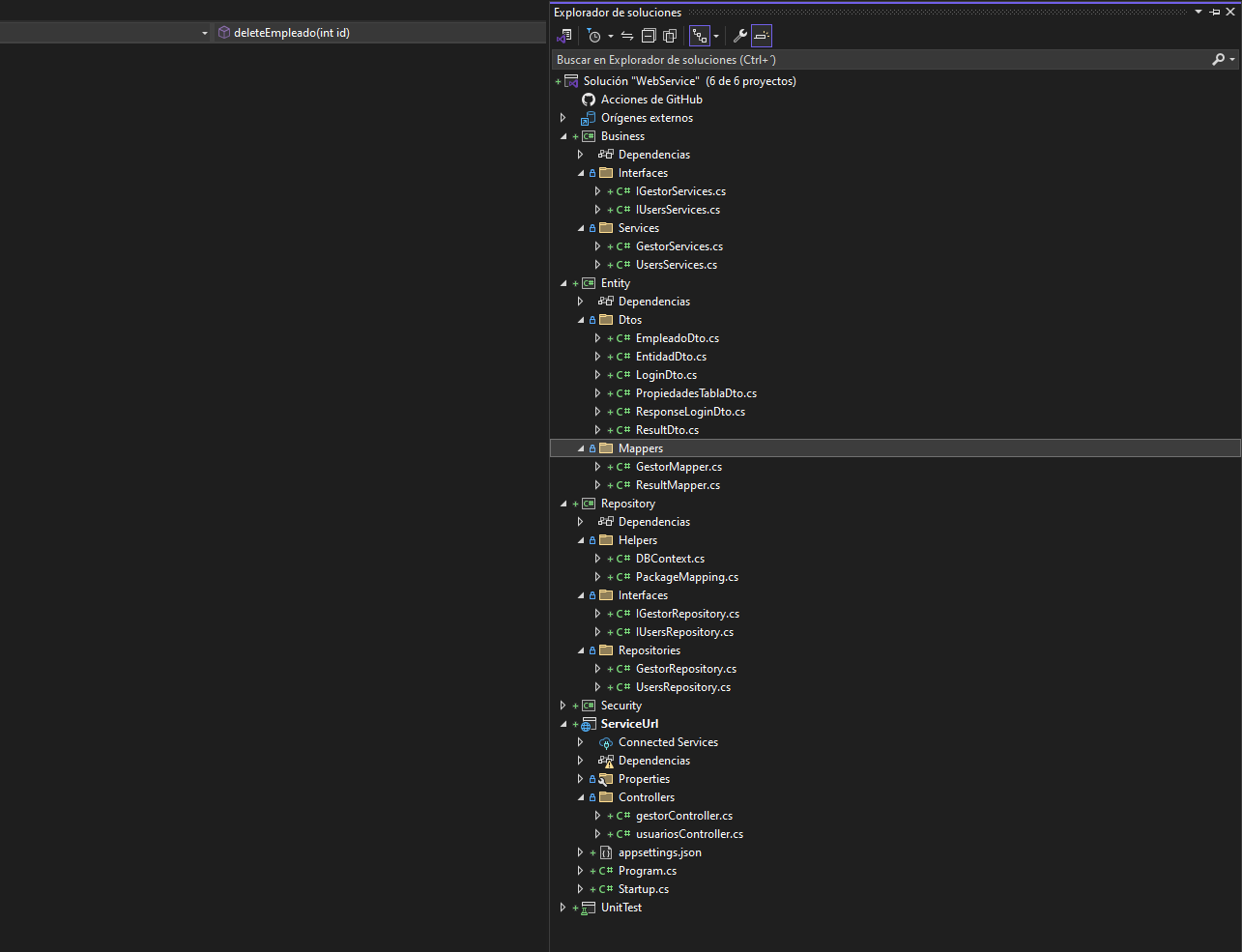
URL: [localhost:52395/api/usuarios](http://localhost:52395/api/usuarios)

* Web Api REST versión NET.CORE 3.1

Metodo get con un mensaje simple para corroborar conexión.

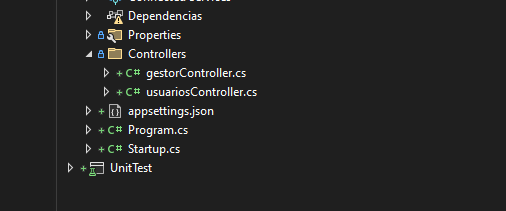


Estructura del proyecto

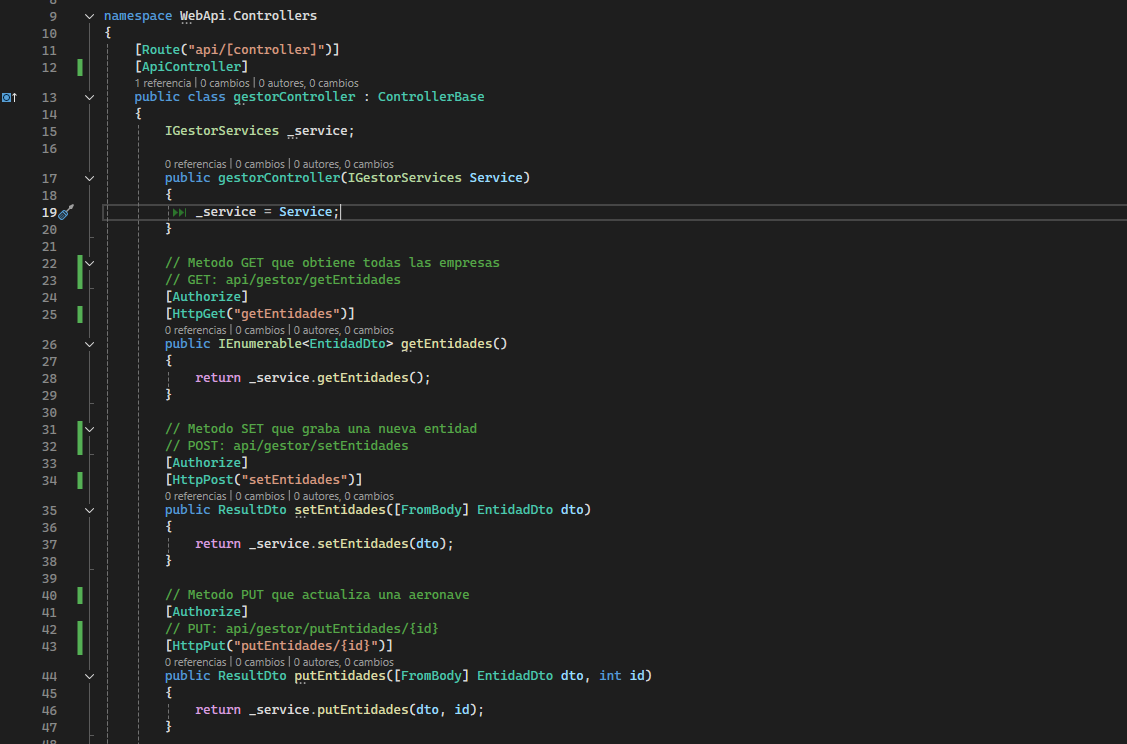


Se crea con inyección de dependencias y 4 capas entre el controlador y el servicio API REST

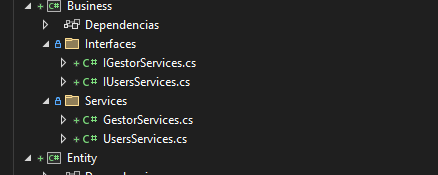
* **Controllers**



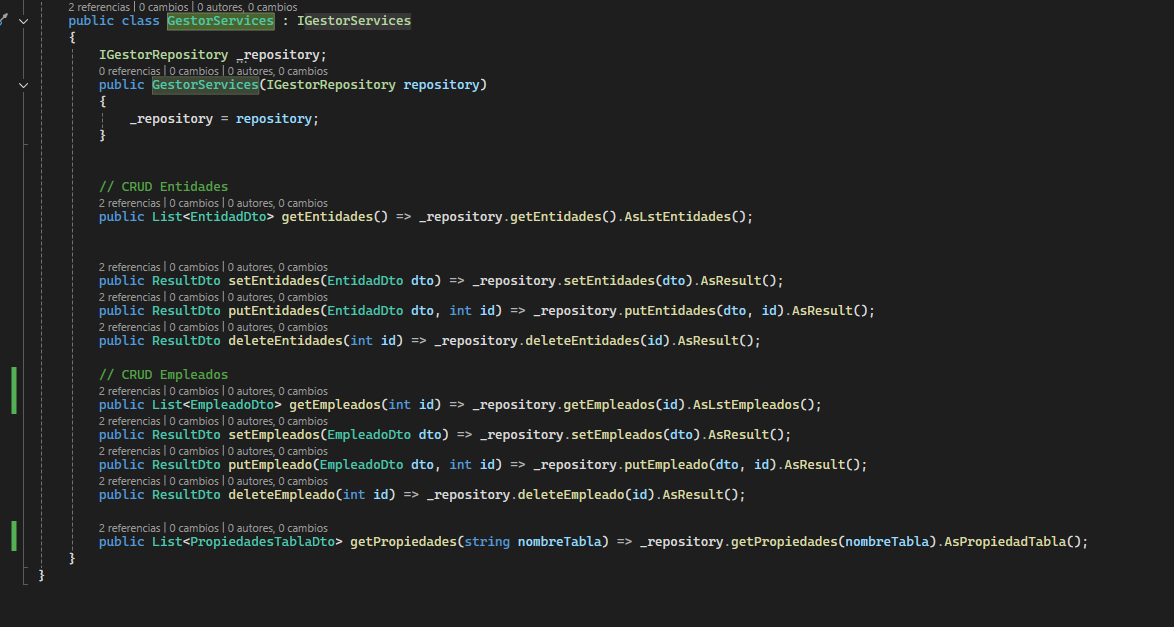
* gestorController

Se crea el controlador correspondiente para poder invocar métodos CRUD en entidades y empleados

* Se implementa como extra la función de tokenJwt en la versión .Net Core 3.1 “Se implementa en peticiones CRUD para poder validar que el usuario se encuentra autenticado” adicional el método de login ni el de propiedadesTabla no tienen validación de token
* **Business**

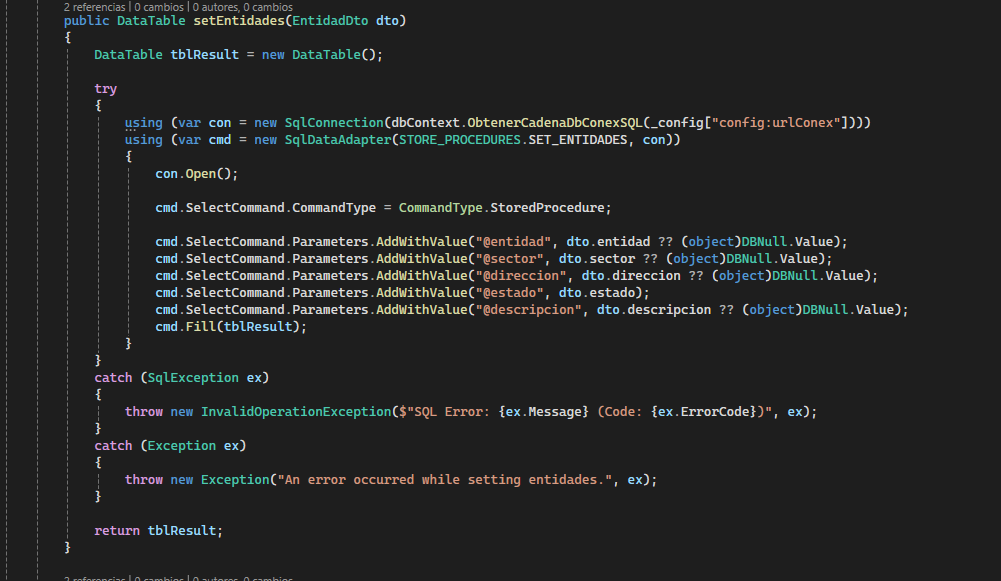


GestorServices.cs



Se crean los correspondientes CRUD para los servicios Entidades y Empleados, como se puede observar se llama la interfaz para poder aplicar la inyección de dependencia, junto con un mapper en la capa entity para convertir los datos retornados en DataTable.

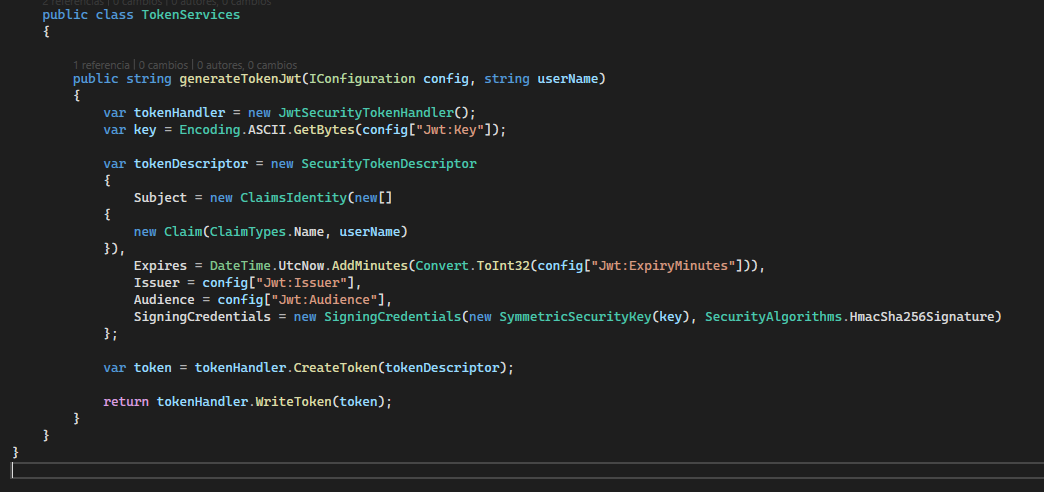
GestorRepository.cs



Se crea capa repositorio la cual se encarga de todas las conexiones a la base de datos, por medio de entity Framwork con procedimientos almacenados para gestión de información

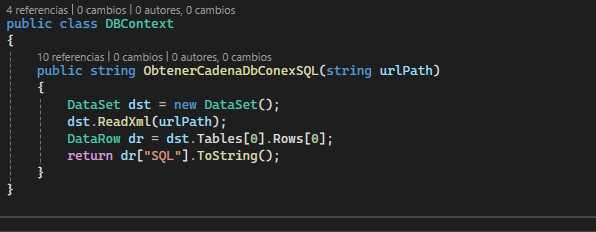
TokenServices.cs





Clase encargada de procesar el token JWT, asigna las llaves de acceso y crea un token con una expiración definida

DBContext.cs



Se crea una clase adicional la cual se encarga de leer el coneccionString, el cual no se encuentra expuesto directamente en el proyecto, esto se hace con fin de mejorar la seguridad del proyecto y que no este expuesta directamente su conexión