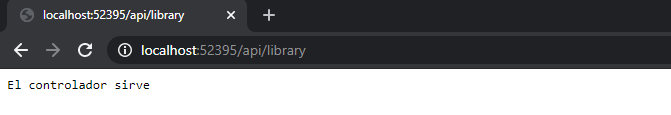
**Documentación API REST**

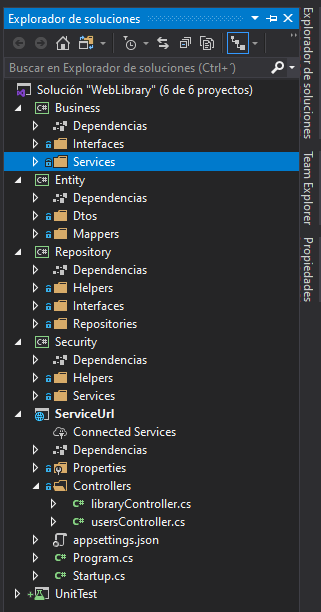
URL: http://localhost:52395

* Web Api REST versión NET.CORE 3.1

Metodo get con un mensaje simple para corroborar conexión.



Estructura del proyecto



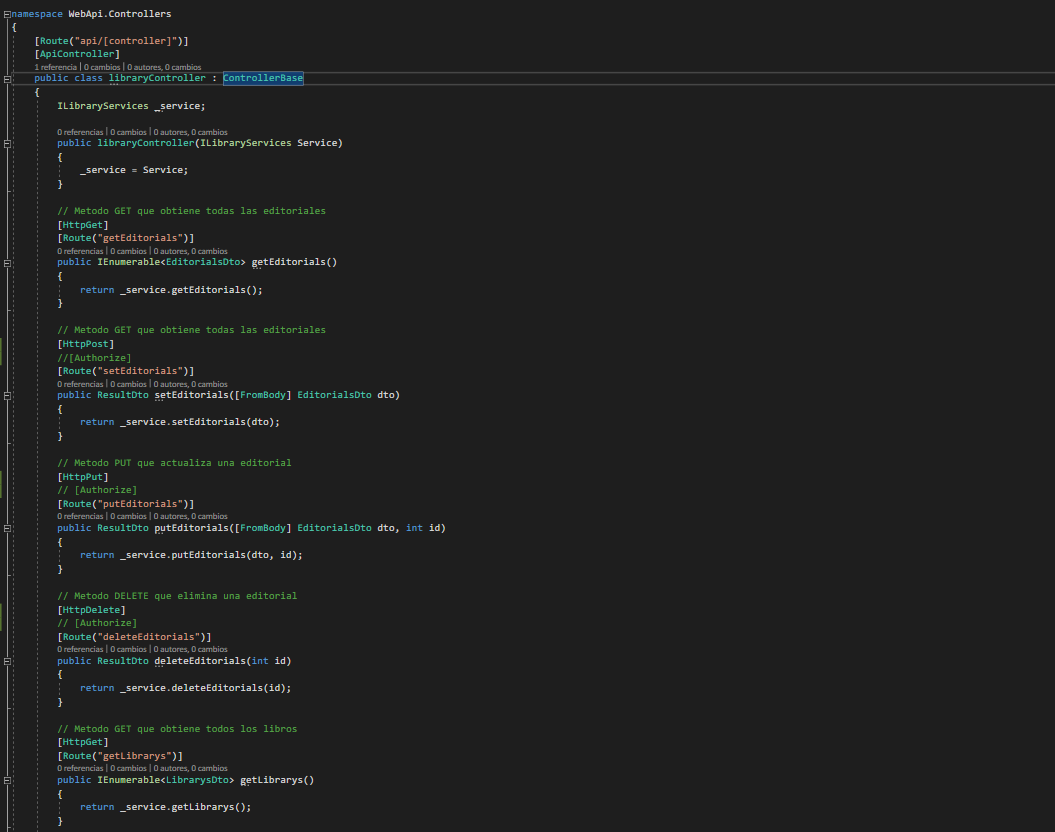
Se crea con inyección de dependencias y 4 capas entre el controlador y el servicio API REST

* **Controllers**

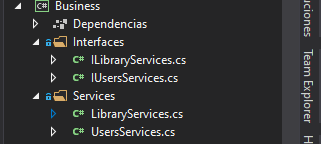


* libraryController

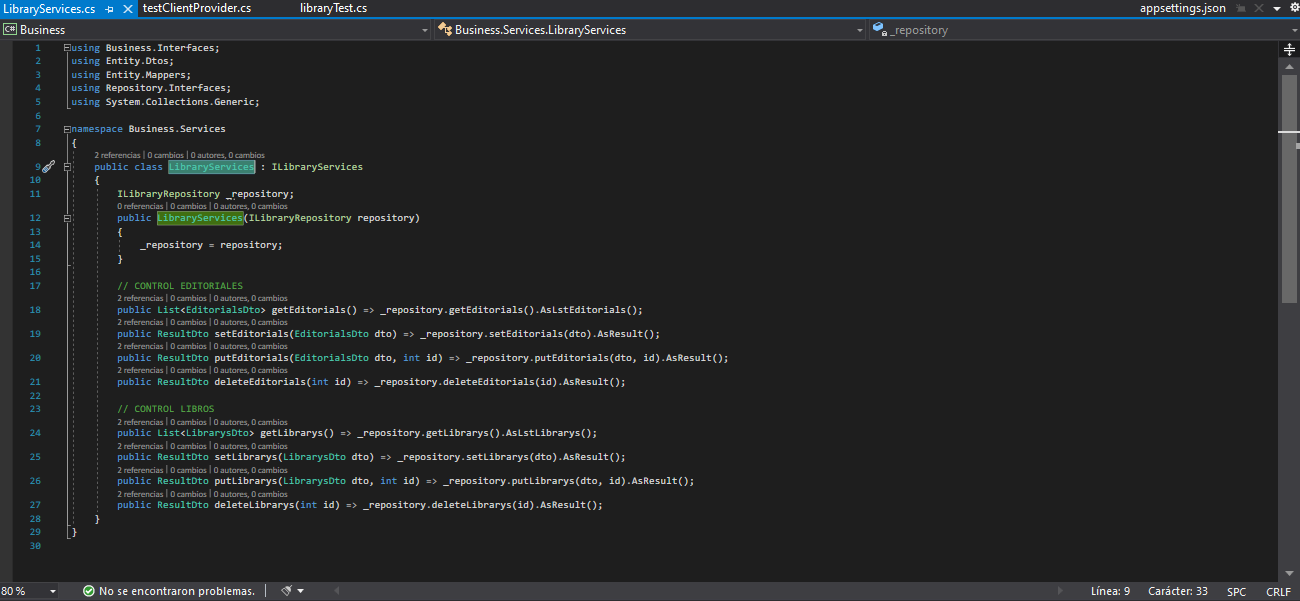
Se crea el controlador correspondiente para poder invocar métodos CRUD en libros y editoriales.



* Se implementa como extra la función de tokenJwt en la versión .Net Core 3.1 “Sin embargo se deja comentado [Authorize] para que en las pruebas no arroje error autorization”
* **Business**

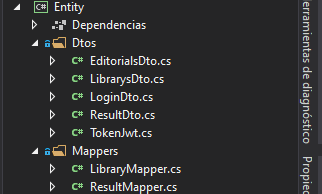


LibraryServices.cs

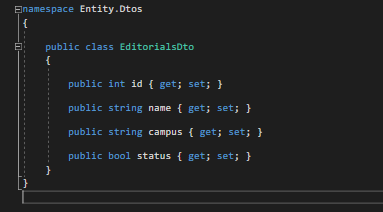


Se crean los correspondientes CRUD para los servicios editoriales y libros, como se puede observar se llama la interfaz para poder aplicar la inyección de dependencia, junto con un mapper en la capa entity para convertir los datos retornados en DataTable.

* **Entity**

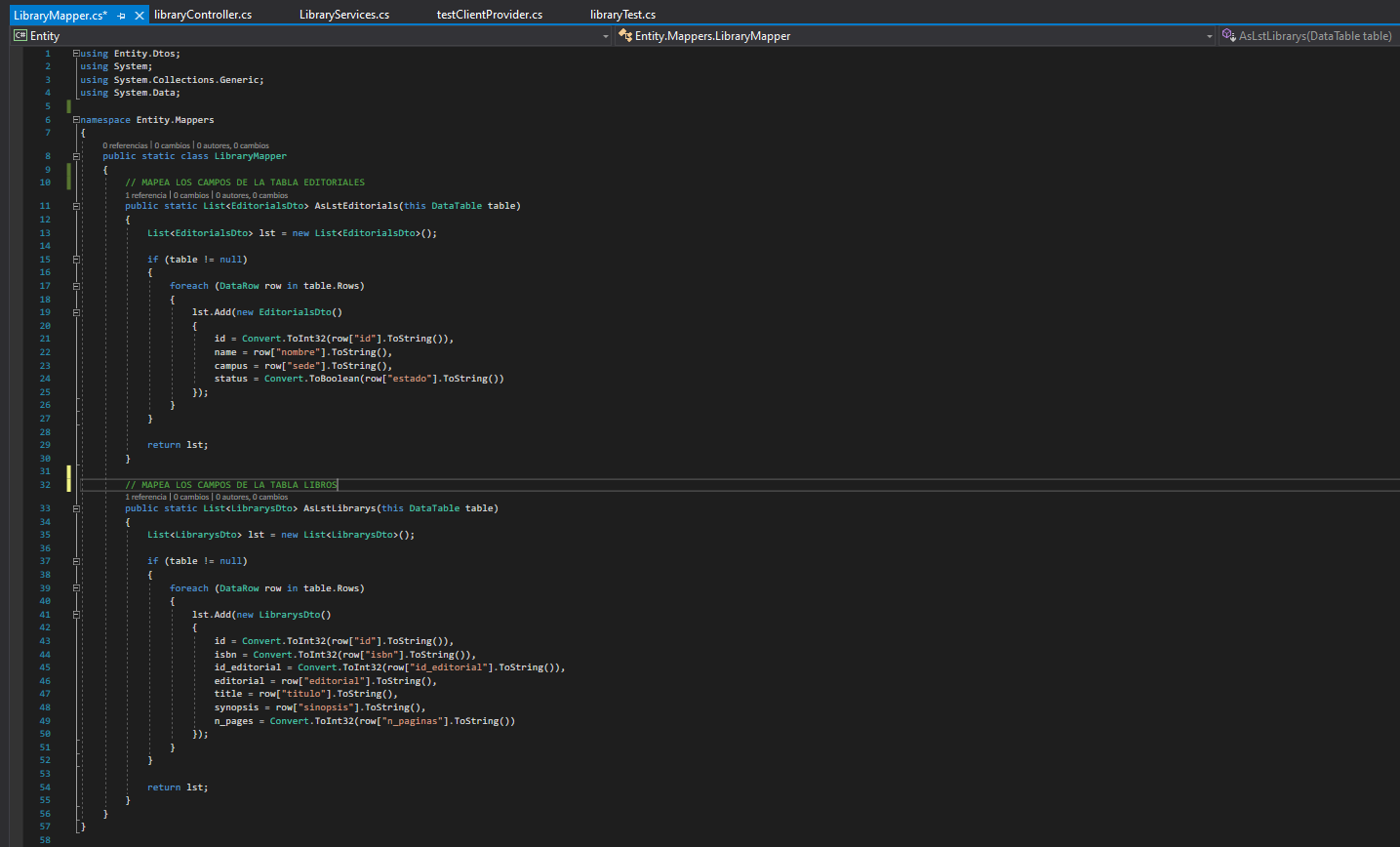


* EditorialsDto.cs

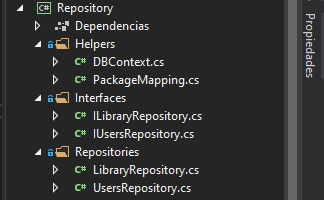


* LibraryMapper.cs

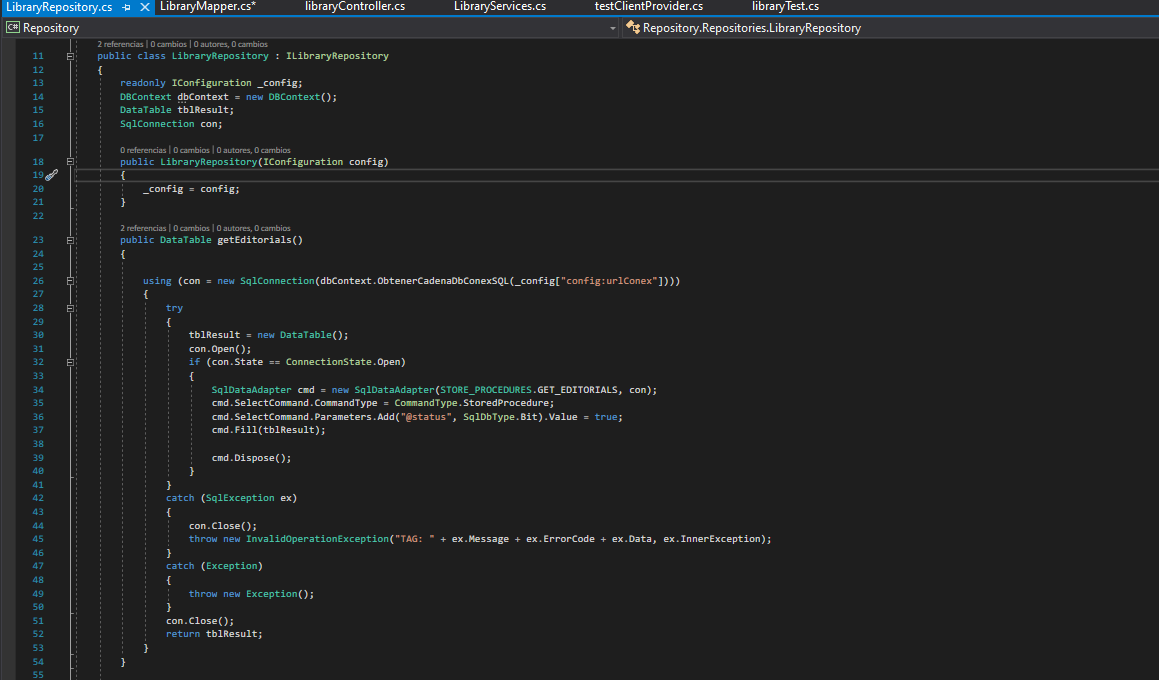
Mapeo de la tabla Editoriales que llega como DataTable, mapeándola a List<EditorialsDto>



* **Repository**



Se crea la capa repository con fin de invocar toda la lógica frente a la base de datos.

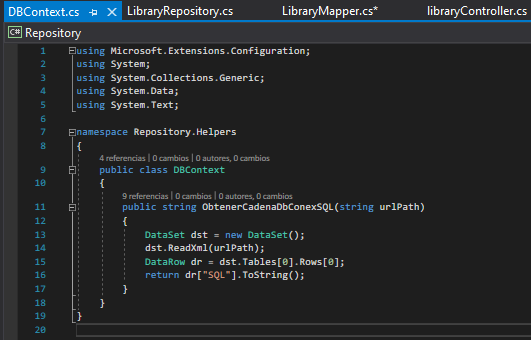


Se implementa llamado a la base de datos para poder manipular su información por medio de procedimientos almacenados, almacenados en la clase STORE\_PROCEDURES,

Se implementa llamado a la Infertaz de IConfiguration para leer la cadena de conexión, que en este caso por seguridad se implementa en un archivo adicional XML con fines de seguridad.

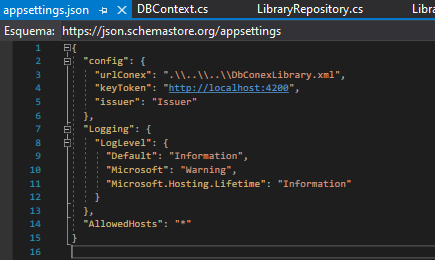
* DBContext.cs

Encargado de leer la cadena de conexión que por seguridad no se deja en el appsettings.json

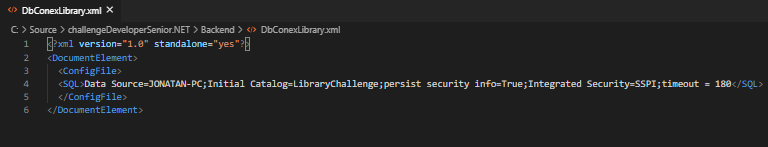


* Appsettings.json

Configuraciones del proyecto para leer la ruta de la conexión y los datos del Token

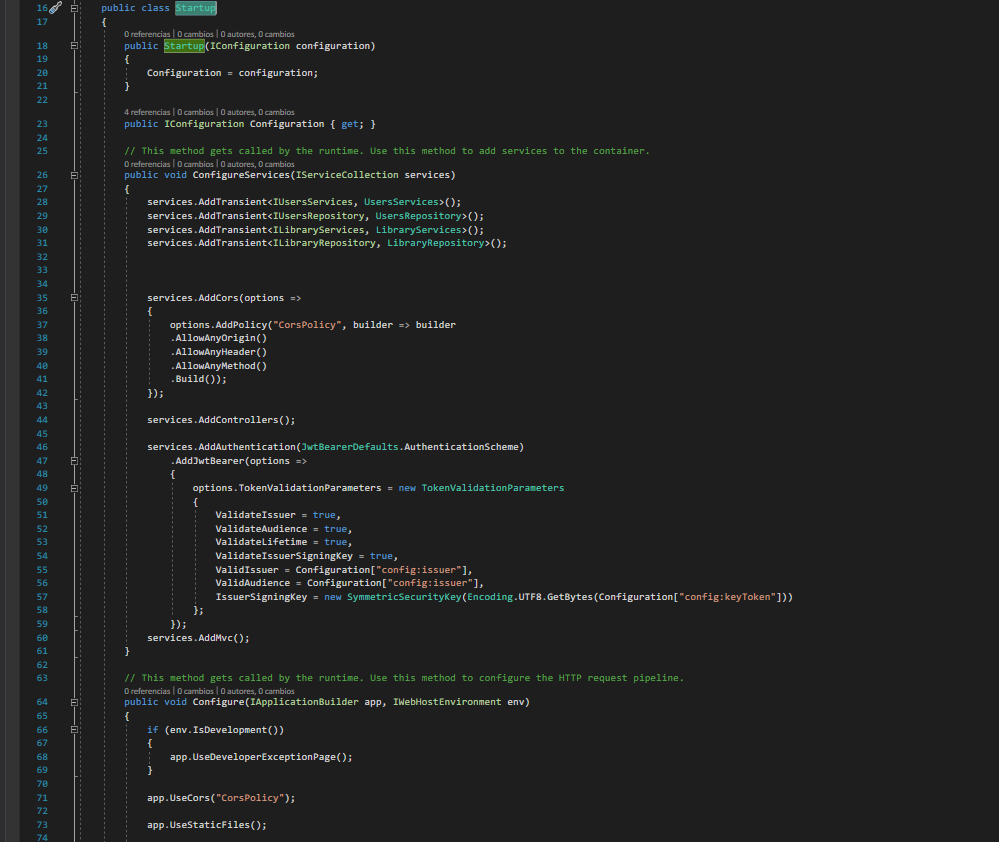


* Cadena de conexión segura en un archivo XML, no se implementa encriptación en la cadena para facilitar la revisión de la prueba.



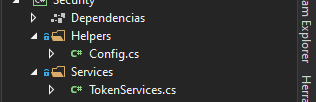
**Configuración Startup.cs**

Se implementa llamado de Cors, de origen cruzado para evitar errores de peticiones HTTP REST, se agregan las capas de inyección de dependencia



**TokenJwt**

Como adicional al desarrollo se implementa la opción tokenJwt con fin de aplicar seguridad a los servicios.



* TokenServices.cs

Llamado a las libretias SecurityJwt y asignación de token por medio de HmacSha256, para una cadena segura.

