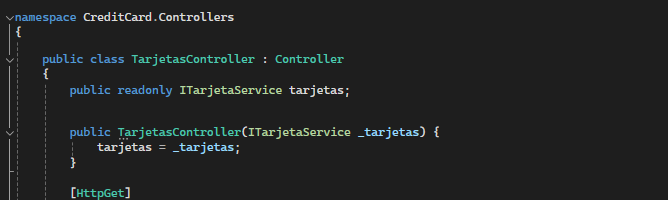
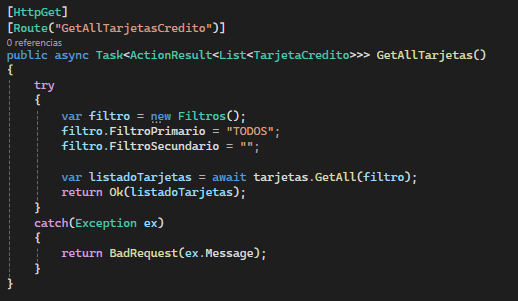
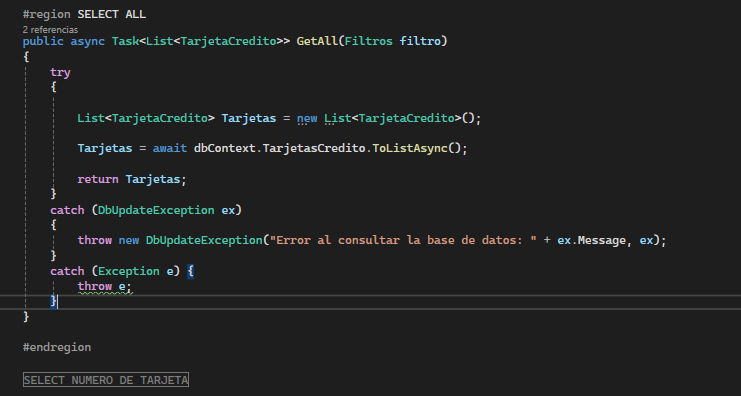
DOCUMENTACION DE LA SOLUCIÓN DE PRUEBA TÉCNICA

En primer lugar, se ha aplicado el principio de Inyección de Dependencias (Dependency Injection) en nuestro controlador TarjetasController. Esto nos permite desacoplar la lógica de negocio de la implementación concreta de los servicios que utilizamos. Para lograrlo, hemos definido una interfaz ITarjetaService que especifica las operaciones disponibles para manipular las tarjetas de crédito.

  
Luego, se implementó los controladores, como GetAllTarjetas() y GetNumeroTarjeta(), que corresponden a las diferentes operaciones que podemos realizar en nuestras tarjetas de crédito. Cada acción maneja las solicitudes HTTP y delega la lógica de negocio al servicio correspondiente. Además, hemos incorporado la gestión de excepciones para garantizar una respuesta adecuada en caso de errores durante la ejecución de las operaciones.

* El método GetAll. Cómo la mayoría de métodos a utilizar en la implementación de la solución reside en la capa de servicios de nuestra arquitectura. Esta capa encapsula la lógica de negocio relacionada con las tarjetas de crédito y actúa como un intermediario entre las capas de presentación y acceso a datos. Para interactuar con la base de datos, el método GetAll utiliza Entity Framework Core, un ORM (Object-Relational Mapper) que simplifica el acceso y la manipulación de datos. Esto abstrae la complejidad del acceso a datos y facilita la implementación de operaciones CRUD (Create, Read, Update, Delete).  
    
  Se capturan excepciones utilizando bloques try-catch para manejar posibles errores durante la ejecución del método.Si se produce una excepción específica de actualización de base de datos (DbUpdateException), la relanzamos con un mensaje personalizado que indica el error ocurrido.Para cualquier otra excepción, simplemente la relanzamos para que sea manejada en niveles superiores.

Al momento de realizar la solución se llevó un tratamiento de código modular y de fácil mantenimiento. Dado a eso se dio una solución robusta y escalable.