# **Prueba desarrollador Full Stack (Angular - .Net Core)**

Esta prueba técnica tiene como objetivo el desarrollo de un proyecto fullstack de inventario de productos, proporcionando una solución integral para el registro, control y monitoreo de los artículos disponibles en stock además de ver tu potencial con el uso de las herramientas, lenguajes, librerías y frameworks que conoces.

La aplicación permitirá a los usuarios realizar tareas como agregar, actualizar, eliminar y consultar productos de manera sencilla y eficiente (CRUD).

La arquitectura del sistema está basada en tecnologías modernas tanto en el frontend como en el backend, garantizando una experiencia de usuario óptima y un manejo de datos seguro y escalable.

🚀 **Instrucciones**

⚜ La siguiente es una prueba técnica donde tendrás completa libertad para proponer, idear, planear y plasmar tu filosofía de trabajo.

⚜ La duración estimada para completar la prueba es de 1 día para terminarla a partir de la hora en que la recibas.

⚜ A continuación, te envío la prueba técnica la cual debes compartir en un repositorio GIT.

🚀 **Descripción del proyecto (Angular + .NET Core)**Estamos desarrollando una API RESTful que permitirá a los clientes de una aplicación buscar y mantener productos en un sistema de inventario que contenga la información básica sobre ellos.

📌Crear un formulario web por el cual se validen las credenciales de un usuario previamente insertados en la base de datos.

 📌Crear un formulario Web que permita realizar las transacciones sobre los productos (CRUD), un mismo producto ingresado puede tener diferentes precios dependiendo de las fechas de ingreso y número de lote. (costos o precio de compra)

📌 Implementar un formulario en donde se presente la consulta del listado de productos ingresados.

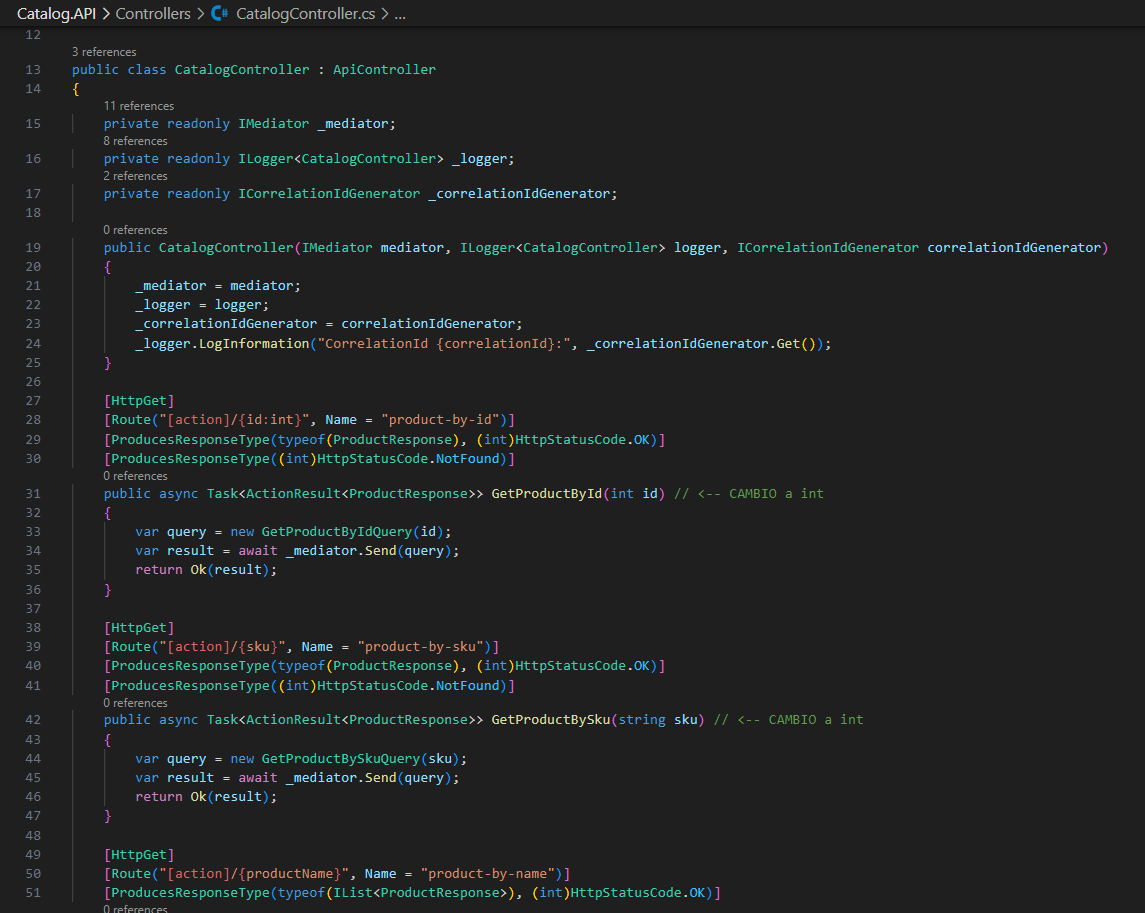
🚀 **Requerimientos técnicos**

✅ El front debe ser desarrollado con Angular

✅ El back tiene que ser realizado con C# .Net Core

✅ Implementar la gestión de datos utilizando Entity Framework

✅ Diseñar y estructurar la base de datos para almacenar los productos y usuarios.  
✅ Implementar una API RESTful para gestionar el CRUD de productos.

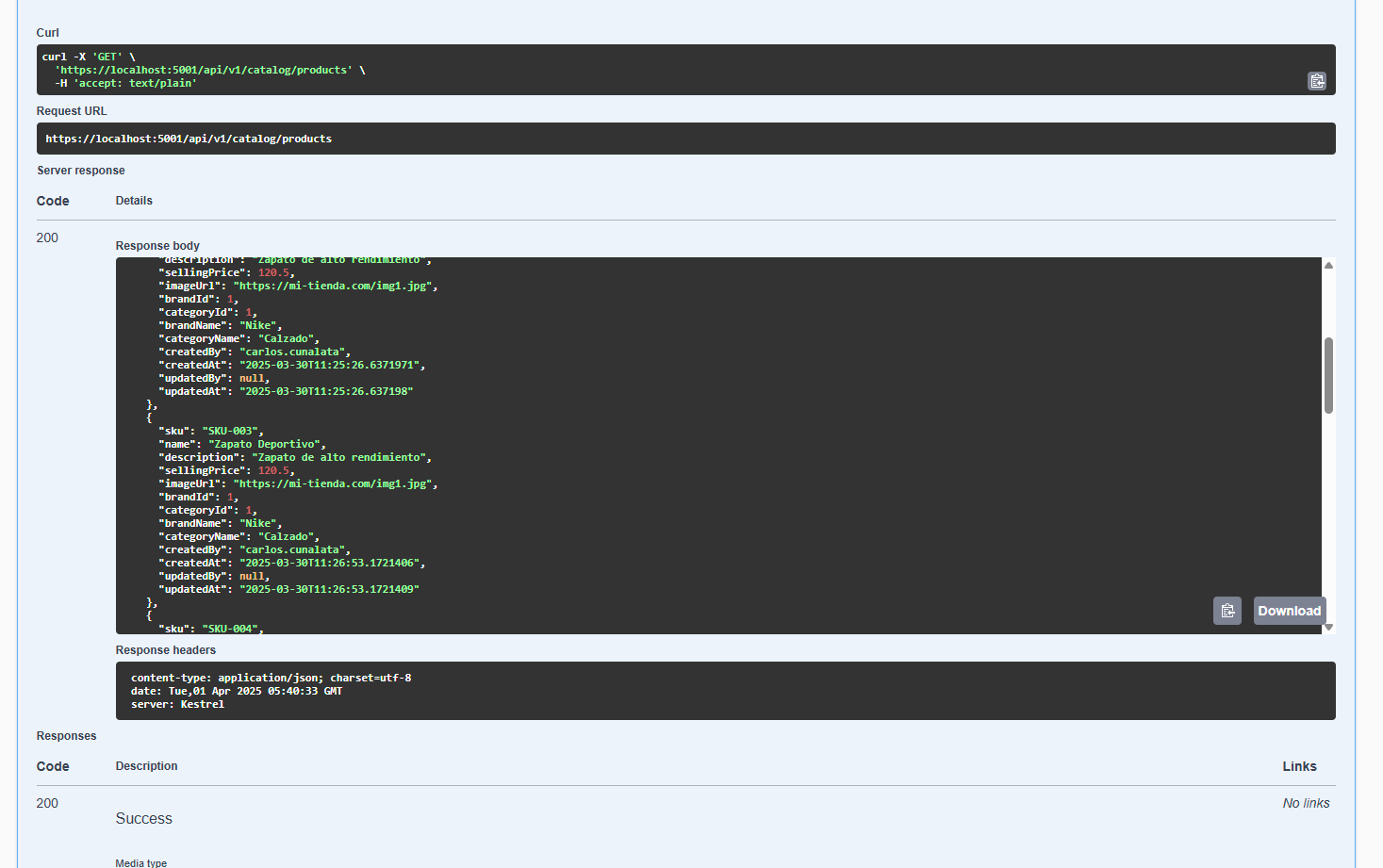


✅ Usar Swagger para documentar cada uno de los endpoints de la API

Correctamente configurado en el backend y se levanta en el puerto 5001 en la siguiente url: <https://localhost:5001/swagger/index.html>

A screenshot of a computer

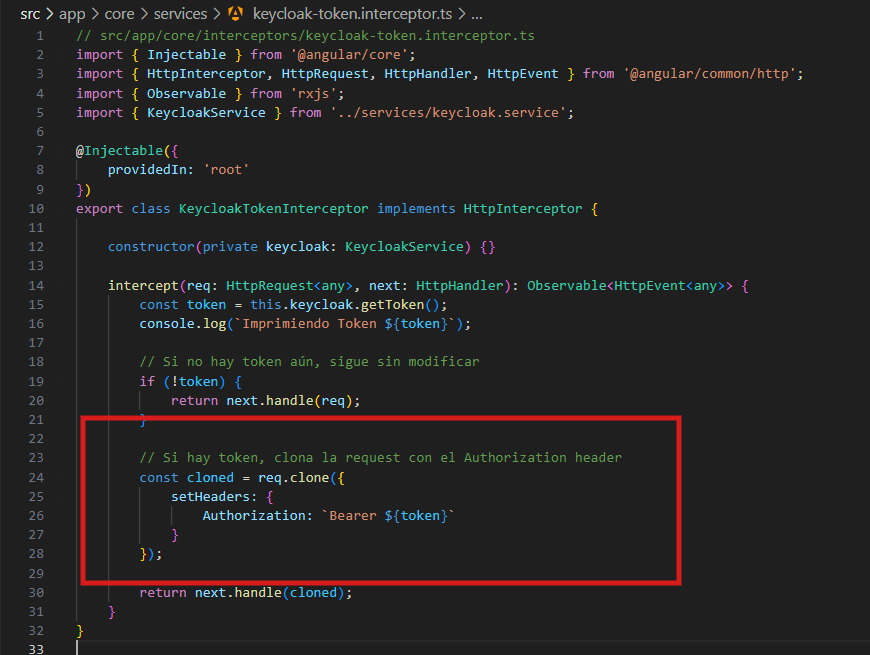
AI-generated content may be incorrect.



✅ Configurar autenticación y control de acceso para usuarios, usando JWT (validaciones de caducidad de tokens, y que el token este funcional) .

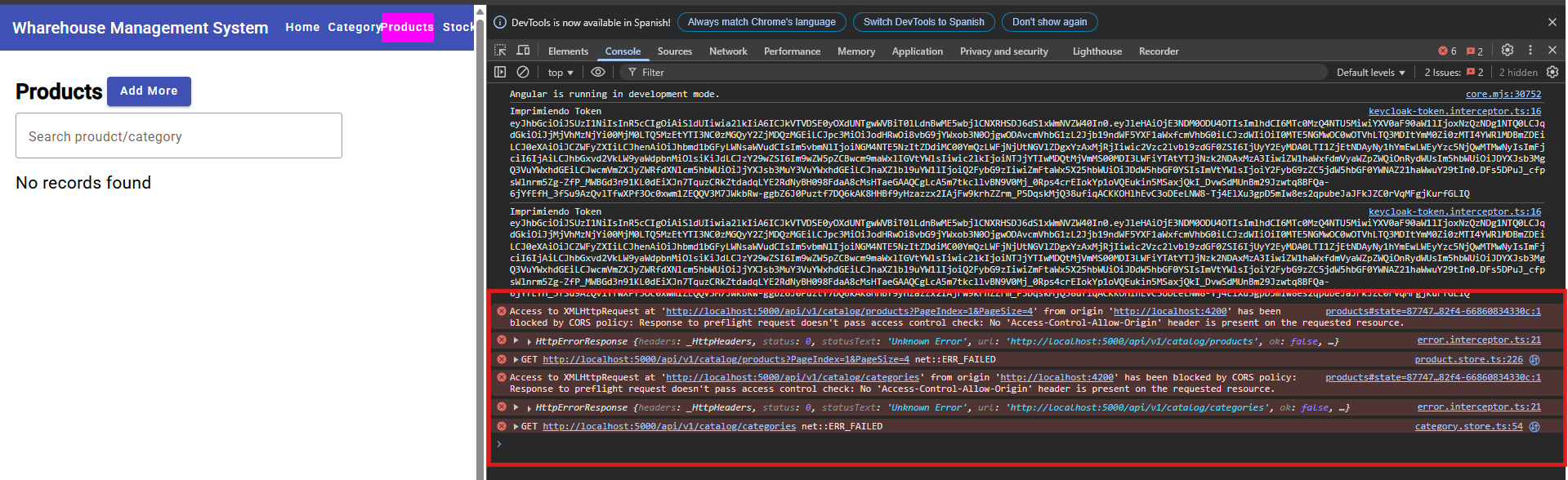
Se dio prioridad explicando la estrategia de autenticación con la capa de seguridad donde se adapto Angular 17 con Keycloak para permitir los siguientes casos:

1. Inicio de sesión
2. Se cerro una sesión desde el SSO y se evidencio que automáticamente el frontend devolvió el frontend al login.
3. Se mostro la funcionalidad de logout.
4. Se implementaron Interceptores en angular para inyectar el token del SSO en cada petición generada hacia el backend sin necesidad de agregarlo manualmente cada que se programa un http request.

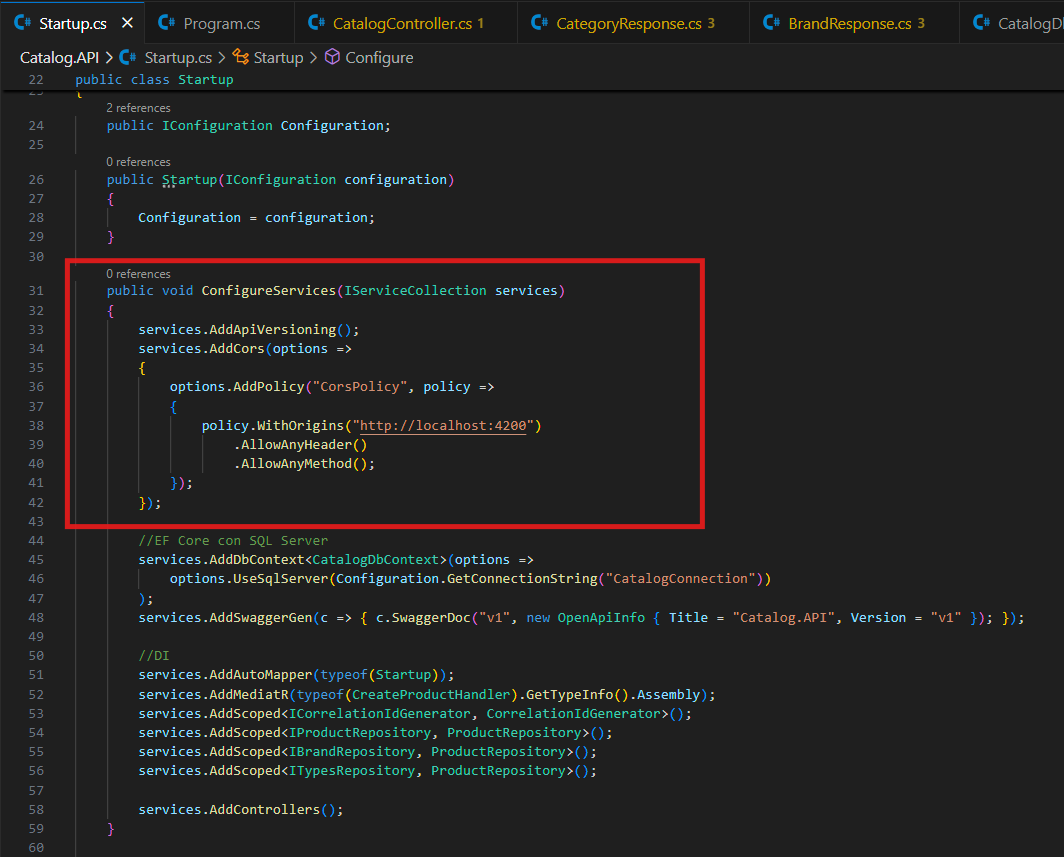


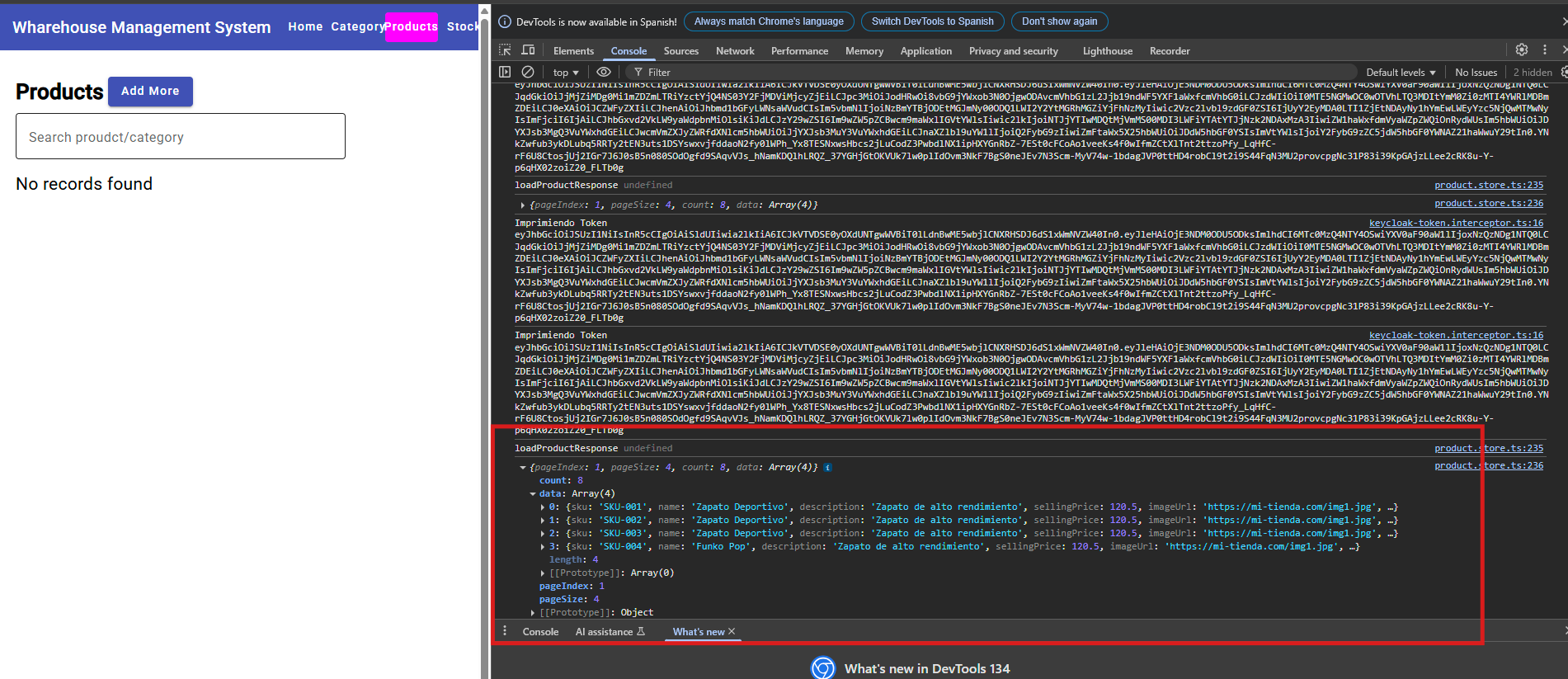
✅ Mostrar el uso de CORS

Angular bloqueando solicitudes http hacia el backend por CORS



Luego de corregir el manejo de cors

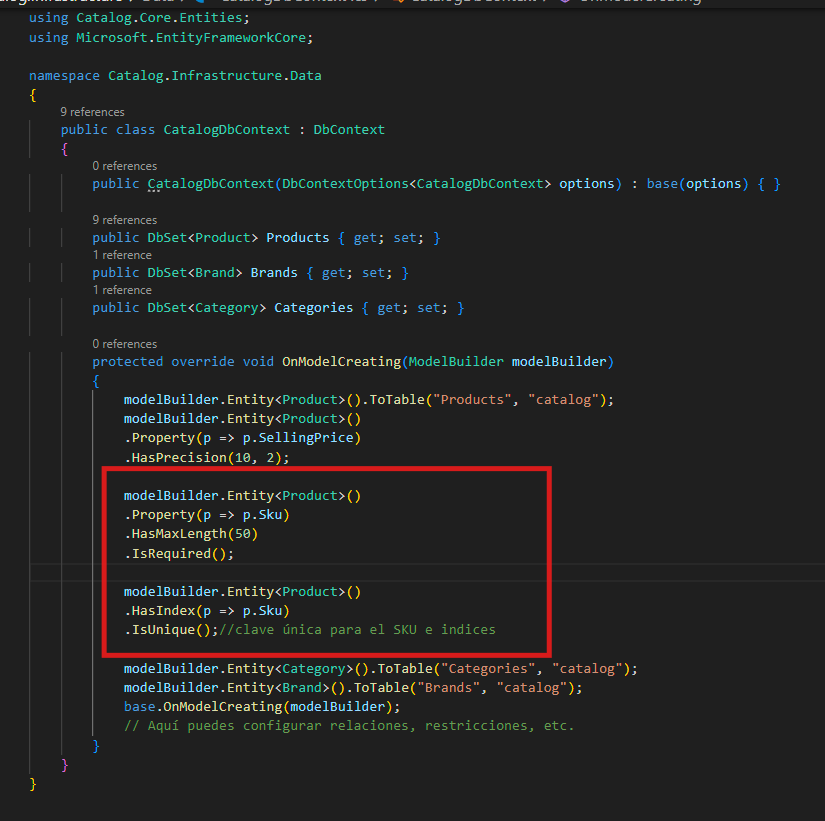




✅ Optimizar consultas para un mejor rendimiento.

Para el modelo de productos se implementó un índice en la columna SKU el cual es el criterio de búsqueda más común para este servicio.

Esto fue implementado usando Entity Framework el cual genera el modelo de datos aplicando migraciones del modelo de objetos “entidades” y los aplica como DDL en la base de datos.



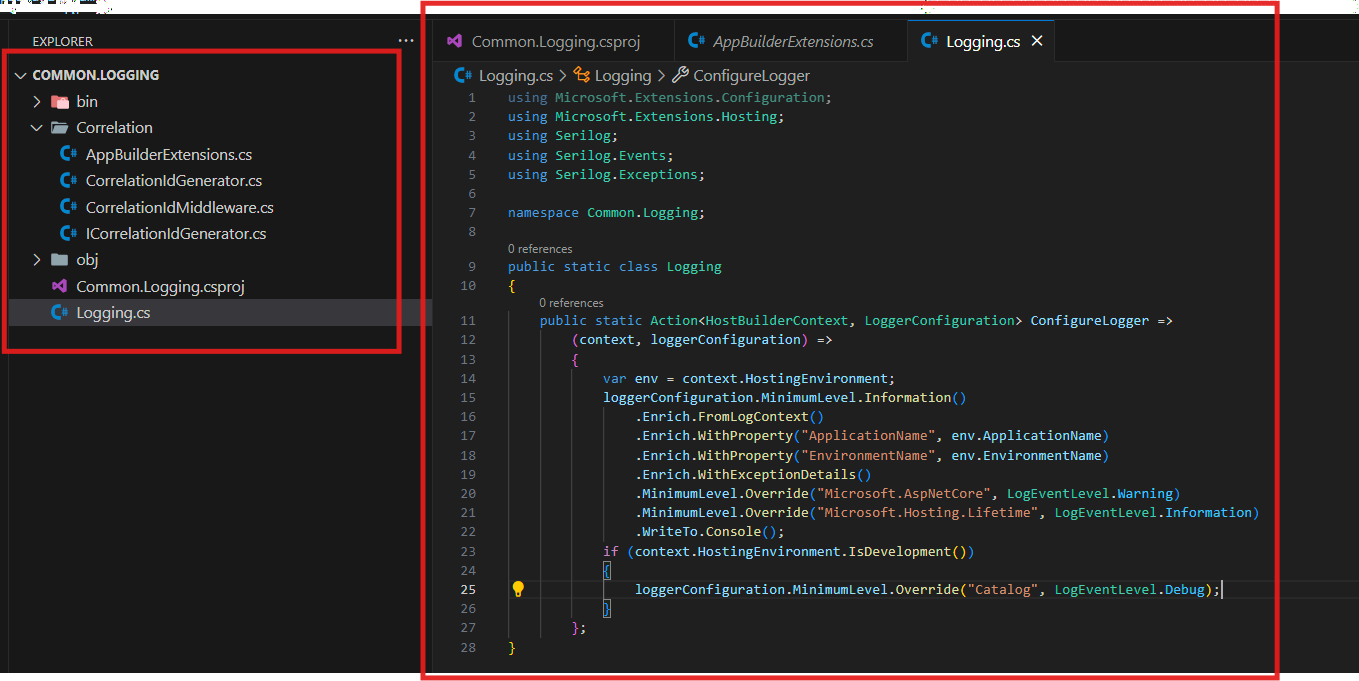
Migraciones después de la creación de índices y ajuste de decimales.

A screenshot of a computer

AI-generated content may be incorrect.

✅ Para un mejor seguimiento y observabilidad de nuestra API realizar la implementación y uso de Serilog.

Se implemento como un modulo independiente en la carpeta infrastructure. Únicamente se creo un transporte de tipo log sin embargo Serilog tiene la ventaja de generar archivos de log en formato json y al ser una arquitectura orientada a microservicios se sugiere crear un transporte de elastic search para agregar características de observabilidad centralizada para la inspección de logs. Esta estructura de logs en formato json permita una búsqueda eficiente de errores basada en propiedades.

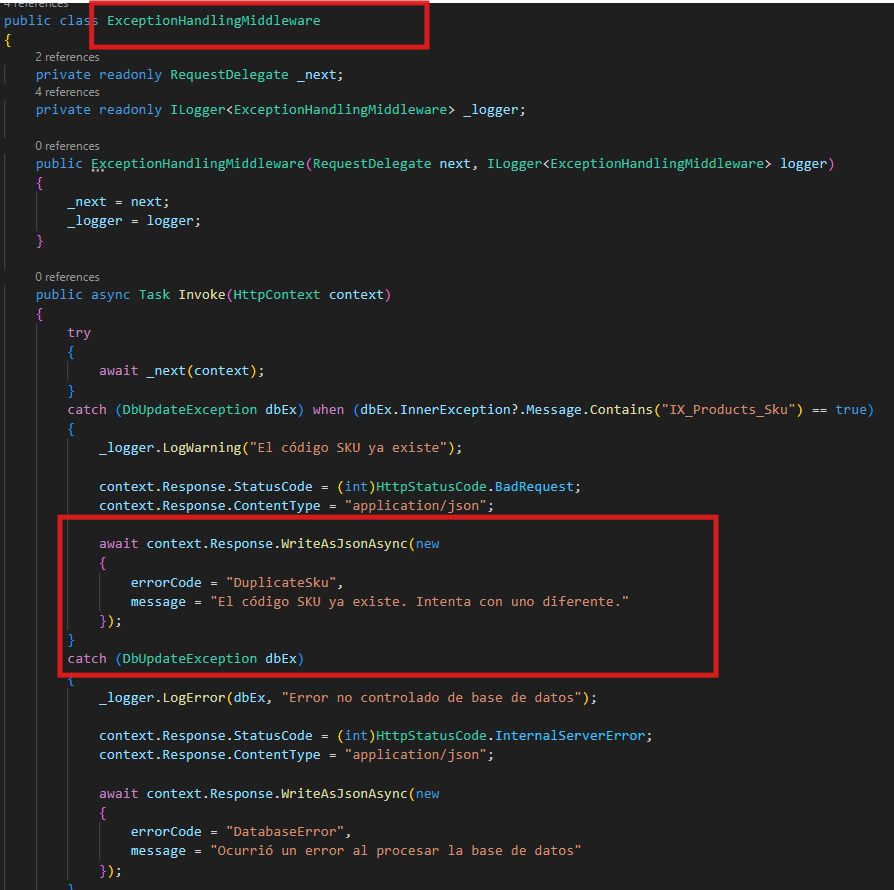


✅ Tanto en el back como en el front implementar manejo de control de errores con su respectivo log y visualización de mensajes personalizados (errores de negocio no técnicos).

Aunque no se logro evidenciar en el frontend se implemento correctamente un manejador de excepciones globales el cual permite interceptar errores de negocio y generar templates usuario amigables en sus respuestas.

A screenshot of a computer program

AI-generated content may be incorrect.



✅ Implementar patrones de diseño adecuados para estructurar el código de la aplicación y asegurarte de que sea mantenible y escalable

**Backend:**

Se implemento con Clean Code como principio de diseño (API, Application, Core, Infrastructure) y el patron CQRS (Command Query Responsibility Segregation) que separa las operaciones de lectura de la escritura de datos el cual mejora el rendimiento y la escalabilidad del microservicio al manejar de forma independiente la lectura de la escritura. Esto también produce modelos más limpios y mantenibles en el tiempo.

Suele ser más complejo de implementar sin embargo produce ganancias en mantenibilidad a largo plazo.  
✅ Implementar diseño responsivo para garantizar una experiencia consistente y óptima en diferentes tamaños de pantalla (opcional pero preferente).

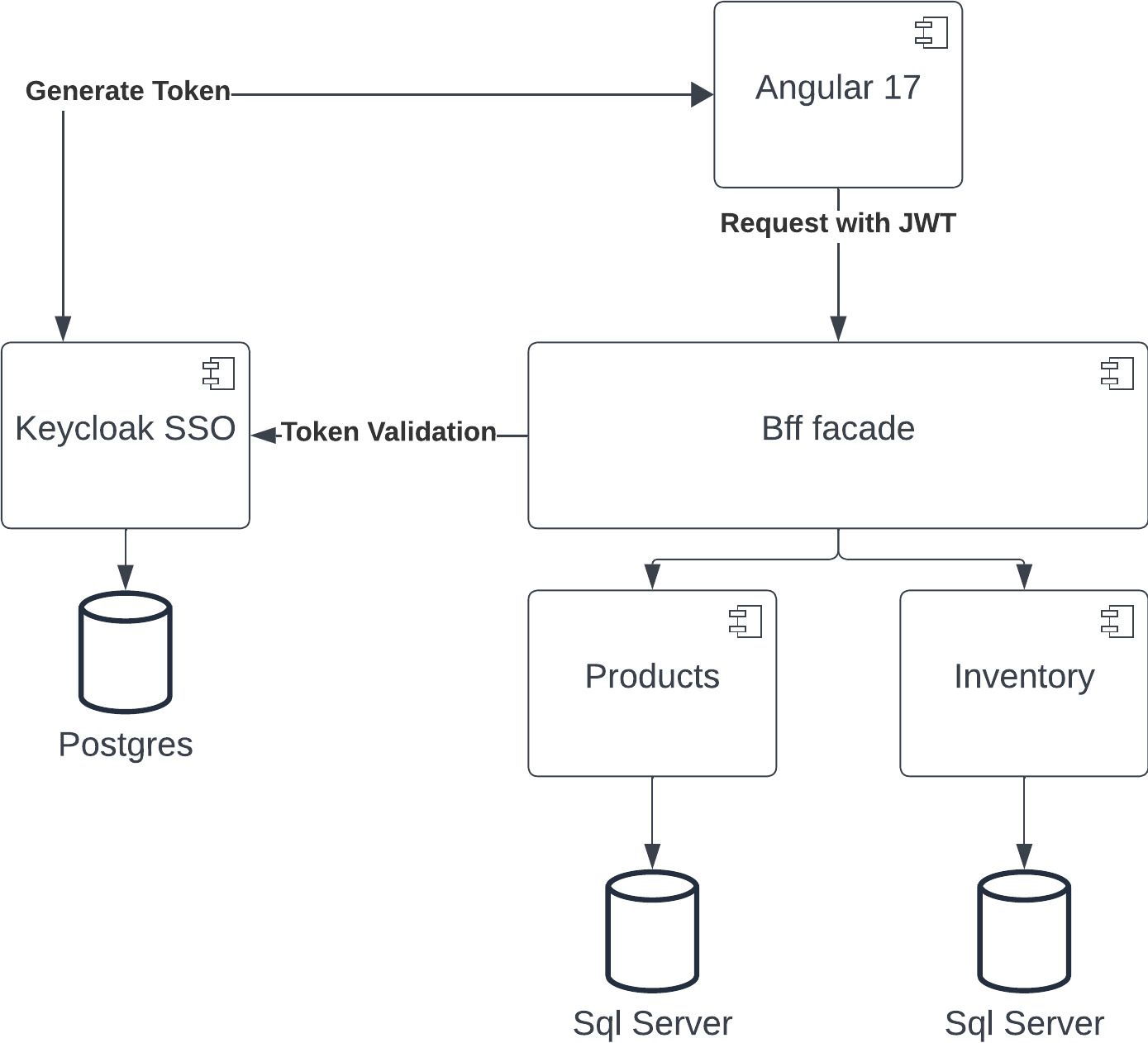
No se logro como se esperaba.

🚀 **Review**

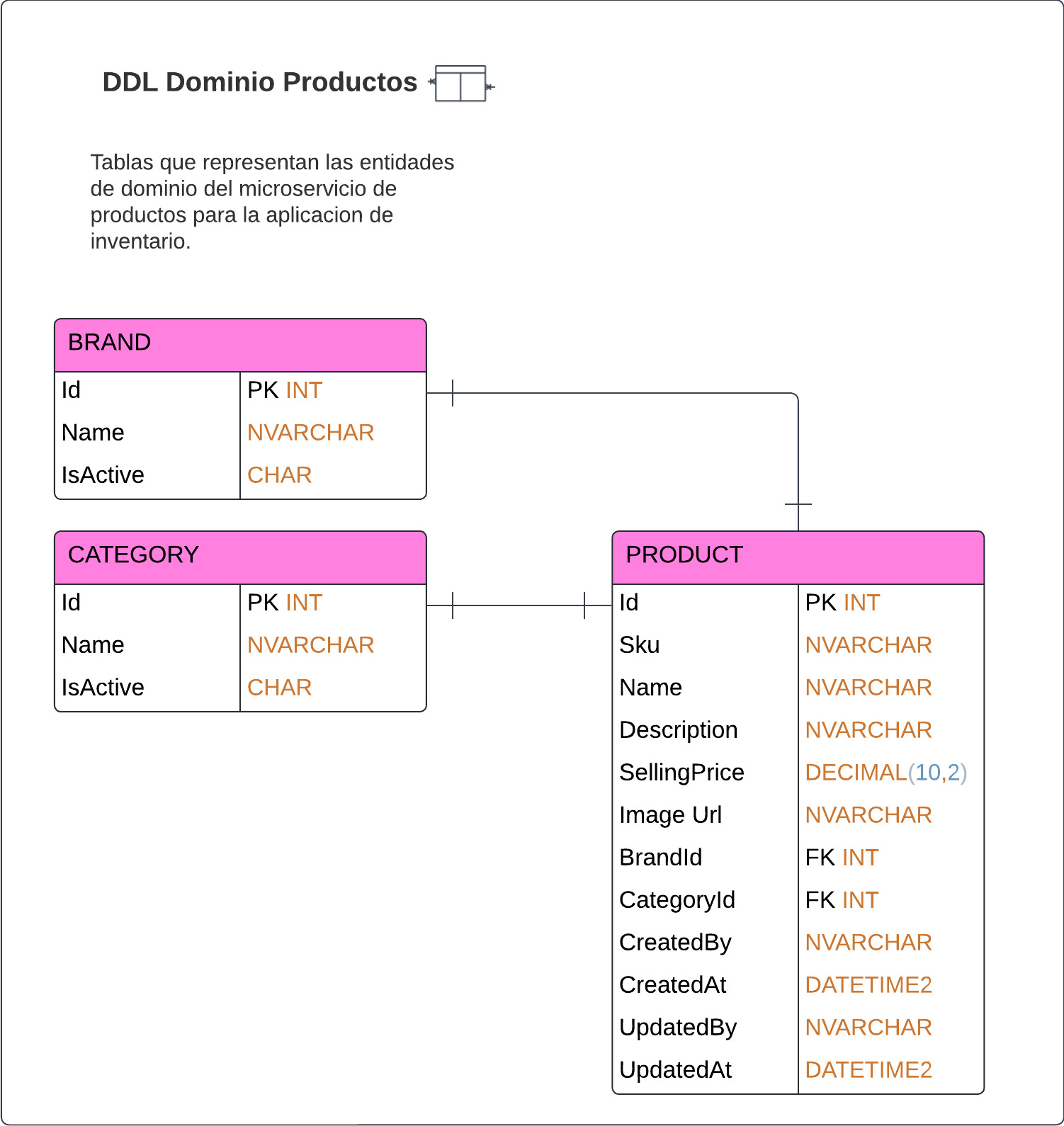
🔍 Entregar código fuente de la aplicación en el versionador de código GIT, GIT HUb, Git Lab

🔍 Discutir decisiones de diseño y arquitectura tomadas durante el desarrollo

1. Diagrama de arquitectura de solución que fue discutivo a profundidad en la sesión de review



1. Según lo expuesto se propuso una arquitectura orientada a microservicios donde se definieron 2 entidades de negocio, cada uno con respectivo diagrama entidad relación. Esta se complementa con una capa de seguridad SSO donde se realiza la gestión de usuarios, rbac (role based Access control) y se monitorean las sesiones activas en frontend y backend.



A screenshot of a computer

AI-generated content may be incorrect.  
🔍 Discutir desafíos significativos encontrados y cómo se las abordaron.

**Se transparento que se hubo un incidente con el frontend por que intente usar la ultima versión de angular y por motivos de tiempo se usó un template básico de github que se adaptó para mostrar todas las características descritas previamente.**

🔍 Implementar un cambio puntual en vivo, sobre la aplicación funcionando