Ejercicio 5: Arquitectura del backend

Describe cómo estructurarías el backend de una aplicación de comercio electrónico. Habla sobre las tecnologías que utilizarías, la organización de los archivos, el uso de patrones de diseño, etc.

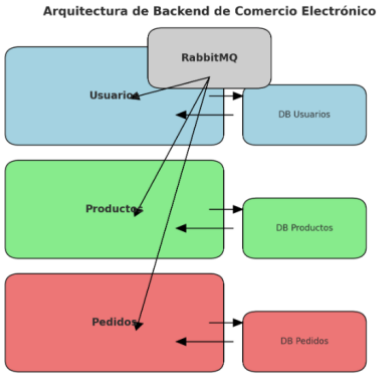
Descripción:

Para empezar, quiero acotar el alcance que vamos a trabajar dentro del diseño de la app de comercio electrónico, para esto podemos empezar con tres entidades básicas (usuarios, productos, pedidos), pero siempre pensando en una aplicación escalable y fácil de mantener, sabemos que este tipo de aplicaciones tiende rápidamente a crecer tanto en número de usuarios como en complejidad.

Pensando en el uso de arquitecturas limpias y las razones dadas anteriormente, se propone usar una arquitectura basada en microservicios, la idea es separar cada responsabilidad en componentes desacoplados, en nuestro caso crearemos uno para usuarios, otro para productos y lo mismo para pedidos, cada uno de estos servicios debe ser lo más autónomo posible, esto implica que cada uno tenga su propia base de datos y use las tecnologías que considere apropiadas.

Esto último implica que se debe tener una sincronización entre los datos compartidos por cada uno de nuestros microservicios, para evitar una comunicación REST entre servicios que nos llevarían a una dependencia directa entre los componentes, usaremos colas y mensajerías implementando RabbitMQ, de esta forma cuando se actualicen los datos en uno de los Servicios Distribuidos este solo producirá un mensaje a la cola y todos los consumidores de esta serán notificados.

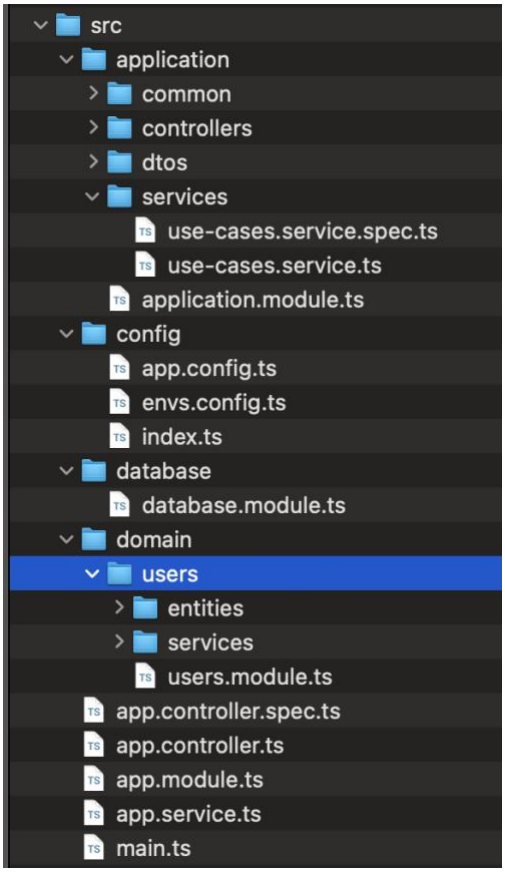
Por último, para la organización de archivo se aprovechará la arquitectura modular que nos ofrece el framework NestJs, esto nos permite separa nuestro dominio principal en un módulo, por ejemplo, users y en otro módulo aplicación, donde podemos manejar los casos de uso que suelen ser los que más tienen a cambiar, otros de los módulos serían database con la configuración de la base de datos y otro módulo para la configuración.



Tecnologías:

- Lenguaje: JavaScript (Node.js), TypeScript
- Framework: NestJS (Node.js + TypeScript)
- Base de datos SQL: MySQL
- ORM: TypeORM
- Colas y mensajería: RabbitMQ
- Comunicaciones http: API-REST
- Documentación: Swagger

Organización de Archivos:



Uso de patrones de diseño:

- Inyección de Dependencias: apoyándonos en Nest y Constructor Inyección
- Singleton: NestJS usa el patrón singleton para manejar las conexiones a bases de datos.