Microservicios

Sprint 2 Lab 4

Felipe Izquierdo Romero

Índice

1. Descripción del problema

Modelar un sistema de microservicios para crear una plataforma de comercio electrónico que maneje usuarios, pedidos, pagos y recomendaciones.

Debe de ser flexible para adaptarse a cambios rápidos de mercado y manejar altos volúmenes de tráfico.

2. Desarrollo

El sistema de microservicios para una plataforma de comercio online estará formado por 8 microservicios montados cada uno en un servidor diferente:

- FrontEnd:

  Encargado de la interfaz de usuario y de proporcionar el acceso a los usuarios montando la web en un servidor de Apache usando HTML, CSS y JavaScript.

- API Gateway:

  Interfaz de comunicación entre todos los servicios exceptuando API Recomendaciones y Base de Datos. Montado en un servidor usando Python y Flask.

  - Enpoints:

    Los enpoints serán básicamente los 4 contenedores que intercomunica en principio, se podrán ampliar según las necesidades de ampliar el sistema.

    - /FrontEnd
    - /Usuarios
    - /GestorBBDD
    - /Pagos
    - /Pedidos

- Usuarios:

  Gestionará el registro, el acceso de usuarios y la protección encriptando datos de carácter sensible del usuario. Montado en un servidor usando Python y Flask.

- Pagos:

  Gestionará el pago de cada pedido y encargada de verificar la autenticidad del usuario a través de otros servicios externos. Tecnología usada Node.js.

- API Recomendaciones:

  Encargada de recomendar al usuario productos similares durante el proceso de compra y de almacenar datos en cache local para recomendaciones en futuras compras o accesos a la web. Tecnología usada Python con Flask.

  - Endpoints:

    En principio serán dos el primero generará un JSON para futuras compras, y el segundo retornará un JSON con productos similares

- /FuturasCompras/{ListaDeProductos}
- /ProductosSimilares/{Producto}

➢ Gestor de base de datos:

Intermediario entre API Gateway y la base de datos encargada de hacer todas las operaciones que conlleven el uso de la base de datos. Usando un servidor de Python con Flask.

➢ Base de datos:

Será una base de datos relacional MySQL que solo se podrá acceder a través del gestor de base de datos para añadir una cierta capa de seguridad a la preservación de los datos frente ataques externos.

El diagrama de los microservicios y la comunicación sería el siguiente.

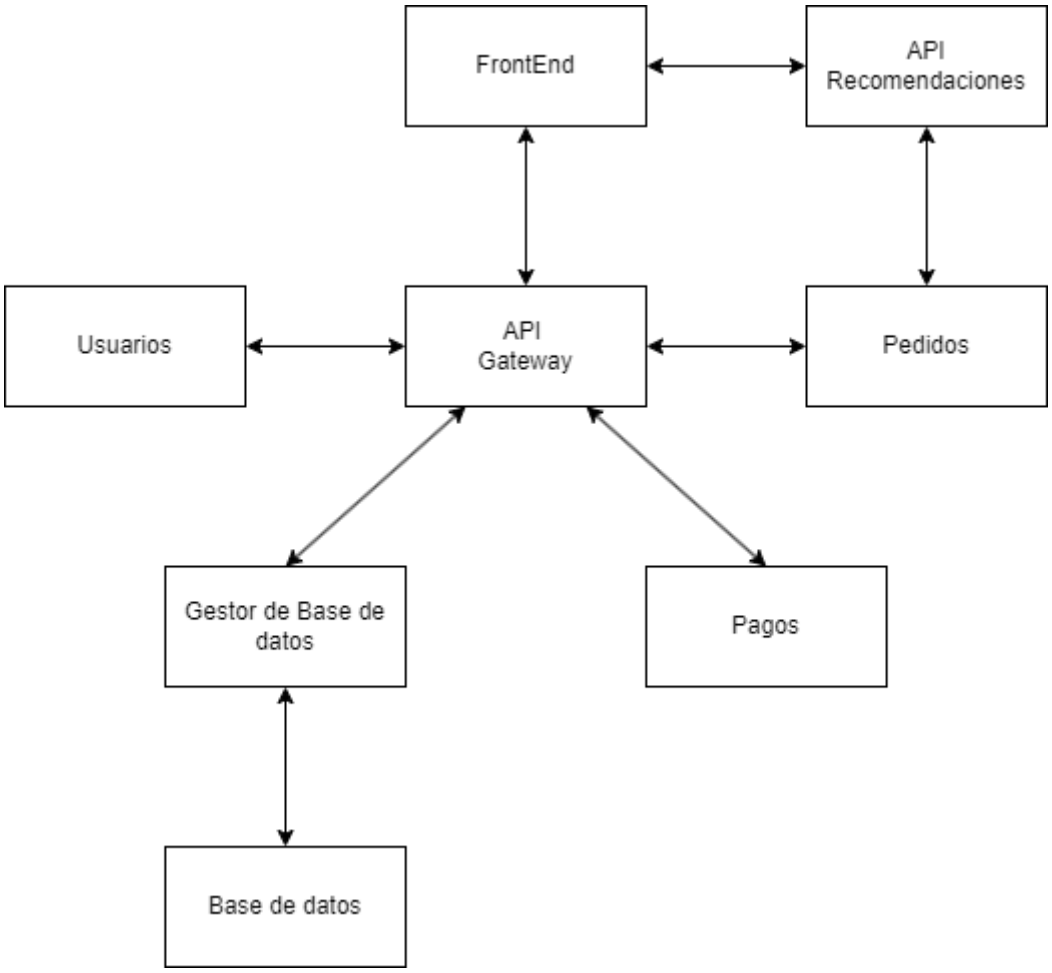

Ilustración 1- Diagrama Microservicios

3. Opcional

Para dockerizar este sistema de microservicios he creado un árbol de directorios por cada servicio para mantener una organización. Además de un dockerfile para cada uno exceptuando el de base de datos. Cada subdirectorio contiene un volumen con su respectivo contenedor.

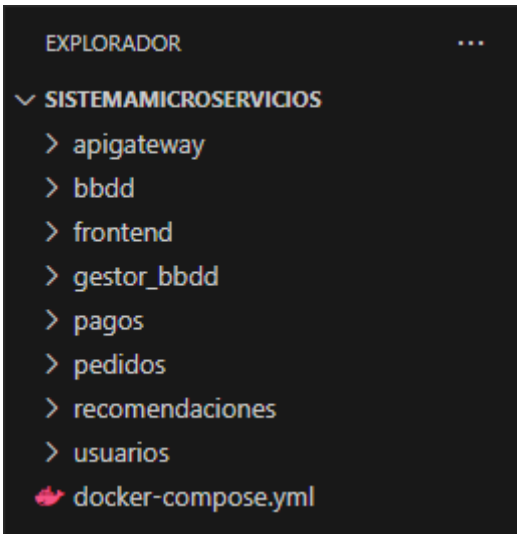

Ilustración 2- Arbol de directorios

En la raíz del proyecto está definido el docker-compose.yml y tras ejecutar desde la raíz se monta el sistema de microservicios.

Además de eso hice un pequeño ejercicio para probar un inicio de sesión para un usuario, pero lamentablemente este ejercicio no funciona en Docker por motivos que no llego a comprender puesto que funciona de manera local en mi ordenador, pero no en Docker.

Video demostrativo microservicios en local