Infraestructura del Proyecto - Sprint 1

⚙️ Herramientas necesarias para el funcionamiento local

- **Git**: para clonar y versionar el repositorio del proyecto.
- **Docker**: para ejecutar todos los servicios en contenedores de forma aislada.
- **Docker Compose**: para orquestar múltiples servicios en conjunto, levantar la red, servicios y base de datos.
- **Java 17**: requerido por los microservicios (Eureka, Auth, User, Gateway).
- (Opcional) **Postman o navegador** para hacer requests y probar endpoints.

Arquitectura de Microservicios

El sistema está desarrollado bajo arquitectura de microservicios. Cada servicio tiene una responsabilidad clara y comunica con los demás a través de HTTP (REST) y está registrado en Eureka.

Servicio	Rol
Eureka Server	Servicio de descubrimiento de microservicios.
API Gateway	Punto de entrada único a todos los servicios.
Auth Service	Encargado de autenticación (login, signup, JWT).
User Service	Maneja los datos de los usuarios registrados.
Account Service	Maneja los datos de la cuenta de un usuario registrado.
MySQL DB	Persistencia de usuarios y autenticación.

Detalle de la Infraestructura

Red

- Se define una única red interna: `digital-money`
- Todos los servicios están conectados a esta red, lo que permite que se comuniquen entre ellos usando el **nombre del contenedor como hostname**.

Servicios

db (MySQL)

- Imagen oficial `mysql:latest`
- Contenedor llamado `mysqldb`
- Expone el puerto `3306`
- Variables de entorno:
  - `MYSQL_ROOT_PASSWORD`: `rootpassword`
  - `MYSQL_DATABASE`: `digital_money`
- Utiliza la red `digital-money`

eureka-server

- Construido desde la carpeta `eureka-server/`
- Expone el puerto `8761` (panel accesible desde el navegador)
- Se encarga de registrar dinámicamente todos los servicios que se levantan

gateway

- Construido desde la carpeta `gateway/`
- Expone el puerto `3500`
- Variables de entorno:
  - `EUREKA_URL`: dirección interna hacia `eureka-server`
  - `PORT`: puerto de ejecución del gateway
- Redirige requests entrantes a `user-service` o `auth-service` según la ruta

user-service

- Construido desde `user-service/`
- Expone `8082:8080` (puerto real del microservicio: 8080)
- Variables de entorno:
  - `PORT_MS`: puerto interno del microservicio
  - `EUREKA_URL`: para registrarse en el servidor Eureka
  - `MYSQL_HOST_URL`: JDBC hacia el contenedor `db`
  - `MYSQL_USERNAME`, `MYSQL_PASSWORD`: credenciales
- Depende de `eureka-server` y `db`

account-service

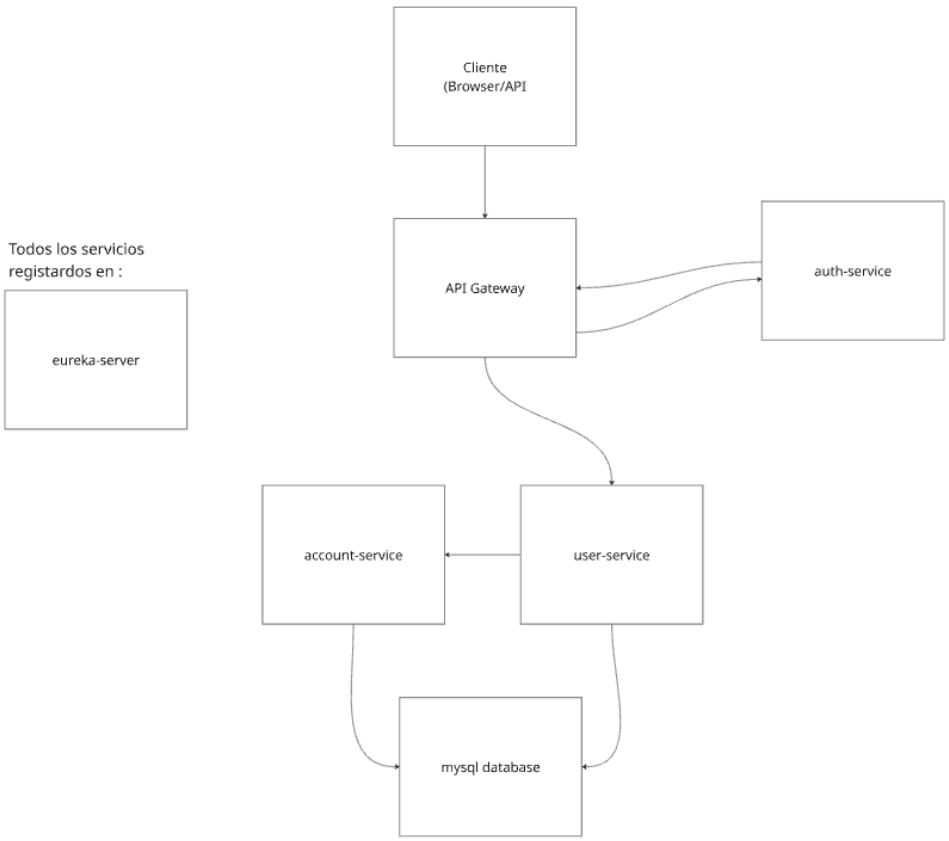
- Construido desde account`-service/`

- Expone `8083:8080` (puerto real del microservicio: 8080)
- Variables de entorno:
    - `PORT_MS`: puerto interno del microservicio
    - `EUREKA_URL`: para registrarse en el servidor Eureka
    - `MYSQL_HOST_URL`: JDBC hacia el contenedor db
    - `MYSQL_USERNAME`, `MYSQL_PASSWORD`: credenciales
- Depende de `eureka-server` y `db`

`auth-service`

- Construido desde `auth-service/`
- Expone `8081:8080`
- Se conecta a Eureka como los demás servicios
- Lógica de login, generación y validación de tokens JWT

Diagrama de Red y Componentes



Flujo de trabajo simplificado

1. El cliente hace una request (ej: login o user info) al **API Gateway**.
2. El **Gateway** redirige la request al microservicio correspondiente (`auth-service` o `user-service`).
3. Los microservicios consultan la base de datos (db) si es necesario.
4. Todos los servicios están **descubiertos dinámicamente** mediante **Eureka**.

Cómo correrlo localmente

git clone https://github.com/tu-usuario/tu-repo.git

cd tu-repo

docker compose up --build

Podés acceder a:

- **Gateway**: http://localhost:3500
- **Eureka Server**: http://localhost:8761
- **User Service** (directo): http://localhost:8082
- **Account Service** (directo): http://localhost:8083
- **Auth Service** (directo): http://localhost:8081