**INFORME DE INFRAESTRUCTURA DEL PROYECTO DIGITAL MONEY HOUSE**

🔹 **Proyecto:** Registro y Autenticación de Usuarios  
🔹 **Tipo:** Aplicación Web con Spring Boot y MySQL en Docker  
🔹 **Entorno de Ejecución:** Local con Docker

1. **Arquitectura General**

📌 **El proyecto sigue una arquitectura cliente-servidor con autenticación JWT y una base de datos MySQL.**  
📌 **Se ejecuta en contenedores Docker con comunicación en una red privada.**

+---------------------+

| Docker Network | 🔹 backend-network

+---------+-----------+

|

+-----------------------+----------------------+

| |

+------v------+ +--------v--------+

| mysql | ◀── Comunicación Interna ─▶ | spring-app |

| (MySQL) | | (Spring Boot) |

| 3306/tcp | | 8080/tcp |

+-------------+ +-----------------+

▲ │

│ │

├── Acceso externo a la BD (`localhost:3306`) │

│ │

└── Aplicación expuesta en `localhost:8080` │

1. **Servicios y Contenedores**

El proyecto se ejecuta mediante **Docker Compose**, que crea los siguientes servicios:

|  |  |  |  |
| --- | --- | --- | --- |
| Servicio | Imagen | Función | Puertos |
| spring-app | openjdk:17-jdk-slim | Aplicación Spring Boot con API REST | 8080:8080 |
| mysql-db | mysql:8.0 | Base de datos para almacenamiento de usuarios | 3306:3306 |

📌 **Ambos servicios están conectados en la red backend-network.**

1. **Infraestructura y Configuración**

📌 **Lenguaje y Framework:**

* Java 17
* Spring Boot (Web, Security, JPA, Validation)

📌 **Base de Datos:**

* MySQL 8.0 en contenedor Docker
* Persistencia de datos con volúmenes

📌 **Autenticación y Seguridad:**

* JWT (JSON Web Tokens)
* BCrypt para el hash de contraseñas
* Spring Security

📌 **Contenedores y Orquestación:**

* Docker para contenerización
* Docker Compose para definir servicios

1. **Configuración en docker-compose.yml**

version: '3.8'

services:

mysql:

image: mysql:8.0

container\_name: mysql-db

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_DATABASE: digital\_money\_house

MYSQL\_USER: user

MYSQL\_PASSWORD: password

ports:

- "3306:3306"

volumes:

- mysql\_data:/var/lib/mysql

networks:

- backend-network

app:

build: .

container\_name: spring-app

restart: always

depends\_on:

- mysql

environment:

SPRING\_DATASOURCE\_URL: jdbc:mysql://mysql:3306/digital\_money\_house?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC

SPRING\_DATASOURCE\_USERNAME: user

SPRING\_DATASOURCE\_PASSWORD: password

ports:

- "8080:8080"

networks:

- backend-network

volumes:

mysql\_data:

networks:

backend-network:

**Con esta configuración:**  
✅ **Los datos de la base de datos persisten tras reinicios.**  
✅ **Los servicios spring-app y mysql-db están en la red backend-network.**  
✅ **La API es accesible en http://localhost:8080.**

1. **Configuración en Dockerfile**

FROM openjdk:17-jdk-slim

WORKDIR /app

COPY target/registro-login.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]

📌 **Este archivo define cómo se crea la imagen del servicio spring-app.**

1. **Configuración en application.properties**

spring.datasource.url=jdbc:mysql://mysql:3306/digital\_money\_house?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC

spring.datasource.username=user

spring.datasource.password=password

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

📌 **Este archivo permite que la aplicación se conecte a la base de datos en Docker.**

1. **Pruebas y Verificación**

✅ **Pruebas Unitarias con Mockito y JUnit 5**  
✅ **Pruebas de Integración con MockMvc y Spring Boot Test**  
✅ **Ejecución de pruebas con:**

mvn test

✅ **Pruebas de endpoints /register, /login, /logout para validar flujo completo.**

1. **Despliegue y Ejecución**

📌 **Compilar el proyecto:**

mvn clean package -DskipTests

📌 **Construir y ejecutar los contenedores:**

docker-compose up --build

📌 **Ver contenedores activos:**

docker ps

📌 **Detener contenedores sin borrar la BD:**

docker-compose down

📌 **Eliminar contenedores y la base de datos (Borra los datos guardados):**

docker-compose down -v

1. **Acceso a la Aplicación**

|  |  |
| --- | --- |
| Recurso | URL |
| API Base | <http://localhost:8080/api/v1/> |
| Swagger UI | <http://localhost:8080/swagger-ui/index.html> |
| Base de Datos | Conectar en localhost:3306 con usuario user y contraseña password |

1. **Conclusión**

✔ **El proyecto sigue una arquitectura cliente-servidor con autenticación JWT y MySQL.**  
✔ **Toda la infraestructura está en Docker para facilitar la ejecución y pruebas.**  
✔ **Se usa una red interna backend-network para comunicación entre servicios.**  
✔ **Pruebas unitarias e integración garantizan el correcto funcionamiento de la API.**  
✔ **El despliegue se hace con docker-compose up --build y la API está accesible en** [**http://localhost:8080**](http://localhost:8080)**.**

✔ **Con esta infraestructura, el proyecto es fácilmente portable, escalable y seguro.**