Historias de Usuario

HU1: Configuración del Control de Versiones con Git

- **Descripción:** Como desarrollador, quiero utilizar Git para gestionar el código fuente del proyecto y asegurar la colaboración entre equipos.
- **Criterios de Aceptación:**
    - Repositorio creado en GitLab.
    - Políticas de branching establecidas (ejemplo: `main`, `develop`, `feature`).
    - Configuración de CI/CD inicial para compilar y probar el código en cada commit.

HU2: Contenerización con Docker

- **Descripción:** Como desarrollador, quiero contenerizar los servicios para garantizar portabilidad y consistencia en los entornos de desarrollo y producción.
- **Criterios de Aceptación:**
    - Dockerfiles configurados para cada microservicio.
    - Uso de Docker Compose para orquestar múltiples servicios.
    - Imágenes alojadas en un registro de contenedores privado (GitLab Container Registry).

HU3: Implementación de Microservicios

- **Descripción:** Como arquitecto de software, quiero implementar los servicios bajo una arquitectura de microservicios para garantizar modularidad y escalabilidad.
- **Criterios de Aceptación:**
    - Servicios separados para **autenticación**, **gestión de usuarios**, y **transacciones**.
    - Configuración de un API Gateway para centralizar las solicitudes.
    - Integración con un servidor de configuración (Config Server) para gestionar propiedades.

HU4: Configuración de Infraestructura de Red

- **Descripción:** Como administrador de sistemas, quiero configurar la red interna para interconectar los servicios y la base de datos de manera segura.
- **Criterios de Aceptación:**
    - Uso de una red interna en Docker para los contenedores.
    - Exposición controlada de puertos.
    - Configuración de un firewall para proteger el acceso externo.

HU5: Configuración del Servidor de Base de Datos

- **Descripción:** Como administrador de bases de datos, quiero tener un servidor dedicado para almacenar los datos de los usuarios y las transacciones.
- **Criterios de Aceptación:**
    - Base de datos MySQL configurada en un contenedor.

- Volúmenes persistentes para evitar pérdida de datos.
- Respaldo automatizado configurado.

Diseño de Infraestructura

Componentes de la Infraestructura

1. **Servidores:**
   - API Gateway: Encargado de enrutar las solicitudes hacia los microservicios.
   - Microservicios: Servicios individuales para autenticación, usuarios, y transacciones.
   - Config Server: Almacena las configuraciones centralizadas.
   - Discovery Server: Usa Eureka para la detección de servicios.
   - Base de Datos: MySQL para almacenamiento persistente.
2. **Almacenamiento:**
   - Disco persistente para datos de la base de datos.
   - Registro de contenedores (GitLab Container Registry) para imágenes Docker.
3. **Red Interna:**
   - Docker Network para la comunicación segura entre contenedores.
   - Firewall para limitar acceso externo.

Pasos a Seguir

1. **Configuración de Git y CI/CD:**
   - Crear repositorio en GitLab.
   - Configurar pipelines para construir y probar código automáticamente.
2. **Contenerización con Docker:**
   - Crear Dockerfiles para cada servicio.
   - Configurar Docker Compose para desarrollo local.
3. **Implementación de Servicios:**
   - Desplegar API Gateway y microservicios con Eureka y Config Server.
4. **Red y Seguridad:**
   - Configurar Docker Network y reglas de firewall.
5. **Base de Datos y Persistencia:**
   - Contenerizar MySQL.
   - Configurar volúmenes para persistencia y respaldos automáticos.

Diseño de Infraestructura

Sprint 1 - Componentes de la Infraestructura

1. **Servidores:**
    - API Gateway: Encargado de enrutar las solicitudes hacia los microservicios.
    - Microservicios: Servicios individuales para autenticación, usuarios, y transacciones.
    - Config Server: Almacena las configuraciones centralizadas.
    - Discovery Server: Usa Eureka para la detección de servicios.
    - Base de Datos: MySQL para almacenamiento persistente.
2. **Almacenamiento:**
    - Disco persistente para datos de la base de datos.
3. **Red Interna:**
    - Firewall para limitar acceso externo.

Descripción de los Componentes

1. **API Gateway (8081):**
    - Responsable de enrutar las solicitudes de los usuarios hacia los servicios internos.
    - Interactúa con **Security Service** para verificar autenticación y autorización.
2. **Eureka Discovery Server (8761):**
    - Detección de servicios en la arquitectura.
    - Permite que los servicios como el **Security Service** y el **Gateway** se registren y descubran dinámicamente.
3. **Security Service (8082):**
    - Gestiona la autenticación, autorización, la generación de tokens, el registro y login de usuarios.
    - Interactúa con **MySQL** para almacenar usuarios y credenciales.
    - Configurado con **Config Server** para centralizar la configuración.
4. **Account Service (8083):**
    - Gestiona las cuentas, generando el cvu y alias..
    - Interactúa con **MySQL** para almacenar las cuentas.
    - Configurado con **Config Server** para centralizar la configuración.
5. **Transaction Service (8084):**
    - Gestiona las transacciones, ingreso y egreso de dinero y su saldo actual.
    - Interactúa con **MySQL** para almacenar las transacciones.
    - Configurado con **Config Server** para centralizar la configuración.
6. **Card Service (8085):**
    - Gestiona las tarjetas, su creación, asociación a una cuenta y eliminación.
    - Interactúa con **MySQL** para almacenar tarjetas.
    - Configurado con **Config Server** para centralizar la configuración.
7. **Config Server (8888):**
    - Centraliza la configuración de todos los servicios.

    - Simplifica la gestión de propiedades sensibles (como credenciales de base de datos).
8. **MySQL Database (3306):**
    - Almacena los datos de los usuarios (credenciales, perfiles, etc.).
    - Accedido únicamente por el **Security Service**.

Ventajas del Diseño

1. **Escalabilidad:**
    - El uso de Eureka permite agregar servicios adicionales en el futuro sin afectar el Gateway.
2. **Modularidad:**
    - Cada componente tiene una responsabilidad bien definida.
    - El Security Service centraliza toda la lógica relacionada con autenticación y autorización.
3. **Seguridad:**
    - Config Server gestiona propiedades sensibles.
    - Comunicación entre servicios limitada a la red interna de Docker.
4. **Facilidad de despliegue:**
    - Con **Docker Compose**, puedes levantar esta infraestructura con un solo comando.