

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

Microservicios Bancarios.

Este proyecto consiste en una arquitectura de microservicios para un sistema bancario, que incluye los servicios de account-service y client-service, junto con sus respectivas bases de datos MySQL.

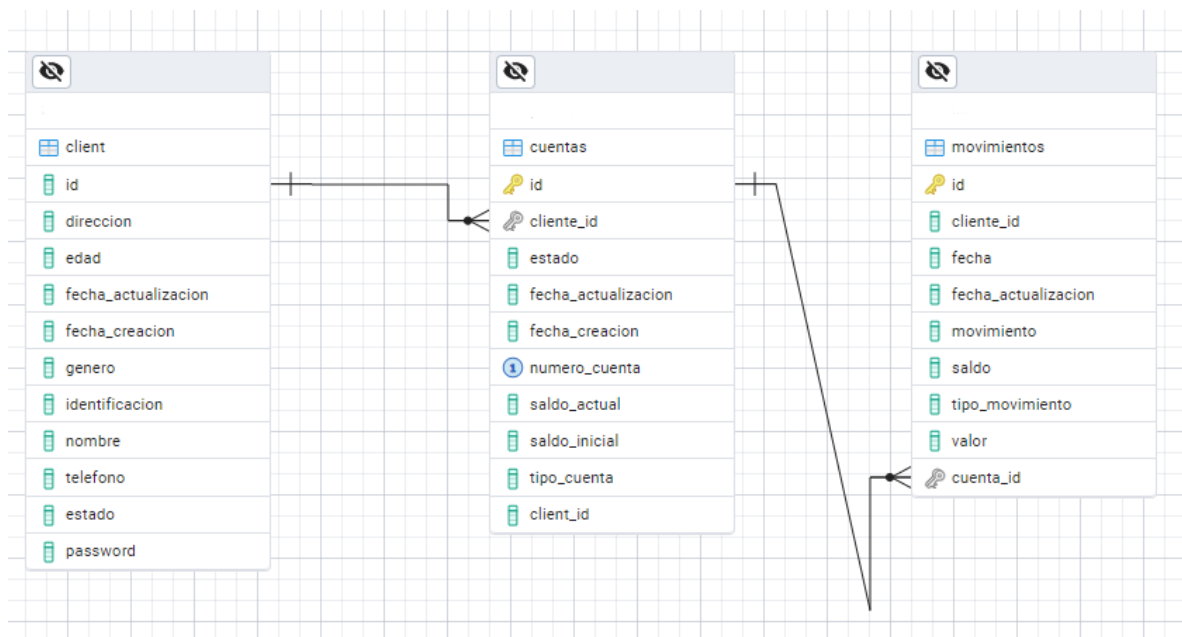
Descripción General

El objetivo de este proyecto es demostrar una implementación de microservicios utilizando Docker y Compose para facilitar el despliegue y la gestión de los servicios.

Diagramas de la Base de Datos

Modelo Entidad Relación (ERD)

Aunque los proyectos están separados y no tiene una relación directa entre las tablas de los dos proyectos, se muestra a continuación la relación lógica de la tabla clientes del proyecto client-service y las dos tablas cuentas y movimientos del servicio account-service, la cual consiste en que un cliente puede tener muchas cuentas y una cuenta puede tener muchos movimientos.



Servicios

Account Service

- Descripción: Servicio encargado de la gestión de cuentas bancarias. Permite crear, actualizar, eliminar y consultar cuentas.

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

- **Tecnologías:**
 - Java
 - Spring Boot
 - MySQL
 - RabbitMQ
 - Maven
 - OpenApi (Swagger)
 - Docker
- Puerto: 8081 (ver health-check.sh)
- **Dependencias:**
 - MySQL (mysql-account)

Client Service

- Descripción: Servicio encargado de la gestión de clientes. Permite registrar, actualizar y consultar información de los clientes.
- **Tecnologías:**
 - Java
 - Spring Boot
 - MySQL
 - RabbitMQ
 - Maven
 - OpenApi (Swagger)
 - Docker
- Puerto: 8082 (ver health-check.sh)
- **Dependencias:**
 - MySQL (mysql-client)

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

Funcionalidades

F1:

- Generación de CRUD (Crear, leer, actualizar y eliminar) en Entidades: Cliente.
- Generación de CRUD (Crear, leer, actualizar y eliminar) Entidades: Cuenta y Movimiento.

F2:

Registro de movimientos: al registrar un movimiento en la cuenta debe tener en cuenta lo siguiente:

- Para un movimiento se puede tener valores positivos o negativos.

Prueba:

Para este caso tome los tipos de movimientos para saber si es Negativo o Positivo, si es Retiro seria negativo y Deposito seria positivo. Tomare la siguiente cuenta para hacer un retiro la cual tiene un saldo 2000 y deben quedar 1000 y ver el movimiento que lo marque como negativo.

The screenshot shows a REST client interface with a GET request to `http://localhost:8081/api/cuentas/2`. The response is a 200 OK status with a response time of 418 ms and a body size of 396 B. The response body is a JSON object representing an account.

Key	Value	Description
id	2	
numeroCuenta	"AH0858896594"	
tipoCuenta	"AHORROS"	
saldoInicial	2000.00	
saldoActual	2000.00	
estado	true	
clienteId	"CLI-85085889"	
fechaCreacion	"2025-09-28T21:52:55.615541"	
fechaActualizacion	null	
activa	true	

Como se puede ver a continuación en la respuesta del servicio marca la salida de los 1000 y los marca como negativo.

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

The screenshot shows a REST client interface with a POST request to `http://localhost:8081/api/movimientos`. The request body is a JSON object with the following fields:

```
1 {
2   "cuentaId": 2,
3   "tipoMovimiento": "RETIRO",
4   "valor": 1000
5 }
```

The response is a 201 Created status with a response time of 1.33 s and a body size of 550 B. The response body is a JSON object with the following fields:

```
1 {
2   "id": 1,
3   "fecha": "2025-09-28T23:12:33.872474428",
4   "tipoMovimiento": "RETIRO",
5   "valor": 1000,
6   "saldo": 1000.00,
7   "cuentaId": 2,
8   "numeroCuenta": "AH0858895594",
9   "clienteId": "CLI-B5085889",
10  "nombreCliente": null,
11  "descripcion": "Retiro por $1,000.00",
12  "retiro": true,
13  "deposito": false,
14  "valorConSigno": -1000,
15 }
```

- Al realizar un movimiento se debe actualizar el saldo disponible.

Prueba:

Para este caso se puede comprobar en la prueba anterior que hizo la actualización del saldo.

- Se debe llevar el registro de las transacciones realizadas.
El siguiente servicio permite ver las transacciones o movimientos realizados.

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

The screenshot shows a REST client interface with a GET request to `http://localhost:8081/api/movimientos`. The response is a 200 OK status with a response time of 1.43 s and a body size of 550 B. The response body is displayed in JSON format, showing a single movement record.

```
{
  "id": 1,
  "fecha": "2025-09-28T23:12:33.872474",
  "tipoMovimiento": "RETIRO",
  "valor": 1000.00,
  "saldo": 1000.00,
  "cuentaId": 2,
  "numeroCuenta": "AH0858895594",
  "clienteId": "CLI-B5085889",
  "nombreCliente": null,
  "descripcion": "Retiro por $1,000.00",
  "retiro": true,
  "deposito": false,
  "valorConSigno": -1000.00,
  "tipoMovimientoFormateado": "Retiro",
  "valorFormateado": "$-1,000.00",
  "saldoFormateado": "$1,000.00"
}
```

F3:

- Registro de movimientos: Al realizar un movimiento el cual no cuente con saldo, debe alertar mediante el siguiente mensaje "Saldo no disponible"

Prueba:

Para esta prueba tomaremos la siguiente cuenta la cual tiene saldo inicial \$0.00

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

GET ▼ http://localhost:8081/api/cuentas/3 Send ▼

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 🔄 200 OK • 372 ms • 390 B • 🌐 📄 Save Response ⋮

{} JSON ▼ ▶ Preview 🖼️ Visualize ▼ 🔍 📄 🔗

```
1 {
2   "id": 3,
3   "numeroCuenta": "AH116CC75875",
4   "tipoCuenta": "AHORROS",
5   "saldoInicial": 0.00,
6   "saldoActual": 0.00,
7   "estado": true,
8   "clienteId": "CLI-F1116CC7",
9   "fechaCreacion": "2025-09-28T22:55:16.903875",
10  "fechaActualizacion": null,
11  "activa": true
12 }
```

Respuesta del servicio:

POST ▼ http://localhost:8081/api/movimientos Send ▼

Params Authorization Headers (9) Body ● Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ 🔗 Schema 🔒 Beautify

```
1 {
2   "cuentaId": 3,
3   "tipoMovimiento": "RETIRO",
4   "valor": 1000
5 }
```

Body Cookies Headers (4) Test Results 🔄 400 Bad Request • 550 ms • 259 B • 🌐 📄 Save Response ⋮

{} JSON ▼ ▶ Preview 🔍 Debug with AI ▼ 🔍 📄 🔗

```
1 {
2   "error": "Saldo no disponible",
3   "mensaje": "Saldo no disponible",
4   "timestamp": 1759100424193,
5   "codigoError": "MOV_ERROR"
6 }
```

F4:

- Reportes: Generar un reporte de "Estado de cuenta" especificando un rango de fechas y cliente.
- Este reporte debe contener: Cuenta asociada con su respectivos saldos.

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

- Detalle de movimientos de las cuentas.
- el endpoint que se debe utilizar para esto debe ser el siguiente:
/reportes?fecha=rango fechas & cliente
- El servicio del reporte debe retornar la información en formato JSON.

Defina, según su expertise, la mejor manera de solicitar retornar esta información.

Prueba:

Se genera reporte ingresando los parámetros indicados.

GET <http://localhost:8081/api/reportes/estado-cuenta?clienteId=CLI-F1116CC7&fechaInicio=2025-09-23&fechaFin=2025-09-28> Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Key	Value	Description
fechaFin	2025-09-28	

Body Cookies Headers (5) Test Results 200 OK • 329 ms • 1.38 KB Save Response

{ } JSON Preview Visualize

```
1 {
2   "fechaInicio": "2025-09-23",
3   "fechaFin": "2025-09-28",
4   "clienteId": "CLI-F1116CC7",
5   "nombreCliente": "Cliente CLI-F1116CC7",
6   "identificacionCliente": "ID-CLI-F1116CC7",
7   "fechaGeneracion": "2025-09-28T23:21:52.38367729",
8   "cuentas": [
9     {
10      "cuentaId": 3,
11      "numeroCuenta": "AH116CC75875",
12      "tipoCuenta": "AHORROS",
13      "saldoInicial": 0.00,
14      "saldoActual": 0.00,
15      "estado": true,
16      "movimientos": [],
17      "resumenCuenta": {
18        "totalDepositos": 0,
```

F5:

- Pruebas unitarias: Implementar 1 prueba unitaria para la entidad de dominio Cliente.
- Se realizan pruebas unitarias para ambos servicios.

F7:

- Despliegue la solución en contenedores Docker.

Prueba Desarrollador Java – Félix Alegría Loango

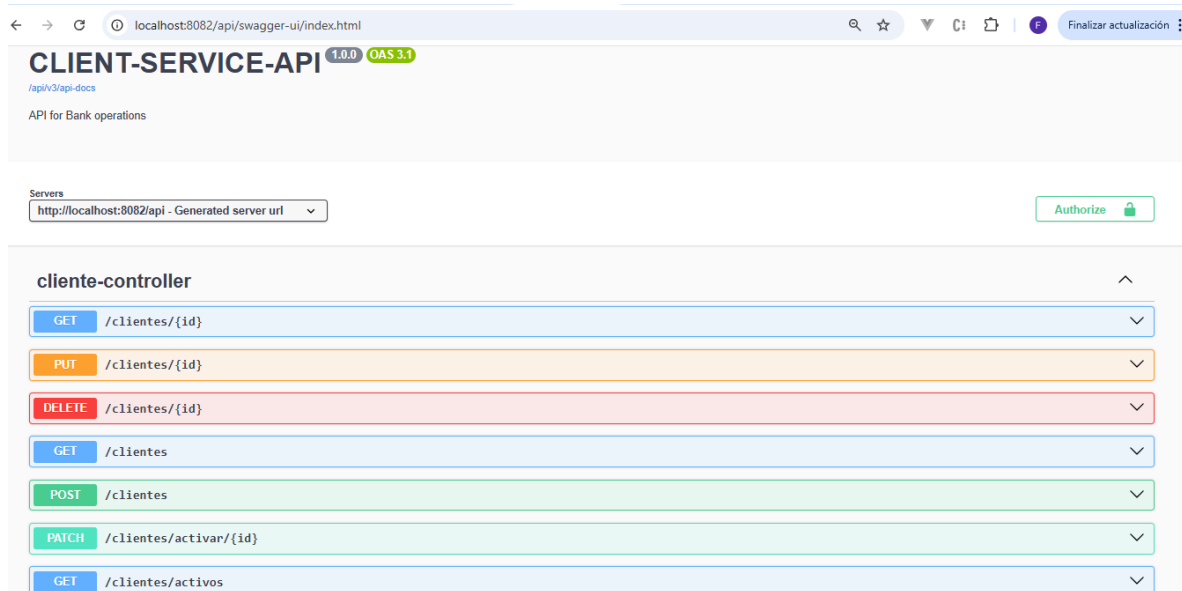
Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

Se realiza configuración de archivos para el respectivo despliegue en Docker, ver punto de **Despliegue** en cual está el paso a paso.

Formas para probar las funcionalidades.

Se agrega colección de Postmas (bank-microservices.postman_collection.json) de cada una de los servicios para validar las funcionalidades. Adicional a esta opción también se puede hacer en la página de inicio (client-service: <http://localhost:8082/api/swagger-ui/index.html> y account-service: <http://localhost:8081/api/swagger-ui/index.html>) una vez carga la aplicación directamente con la documentación de OpenApi:



Requisitos

- Docker
- Docker Compose
- RabbitMQ
- MySQL

Despliegue

El despliegue se realiza mediante Docker Compose. Siga estos pasos:

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

Asegurase de tener Docker corriendo en la maquina donde se realizará el despliegue.

1. Clonar el repositorio:
2. `git clone https://github.com/Alegria2016/BANK-MICROSERVICES`

`cd bank-microservices`

3. Ejecutar el script de despliegue:

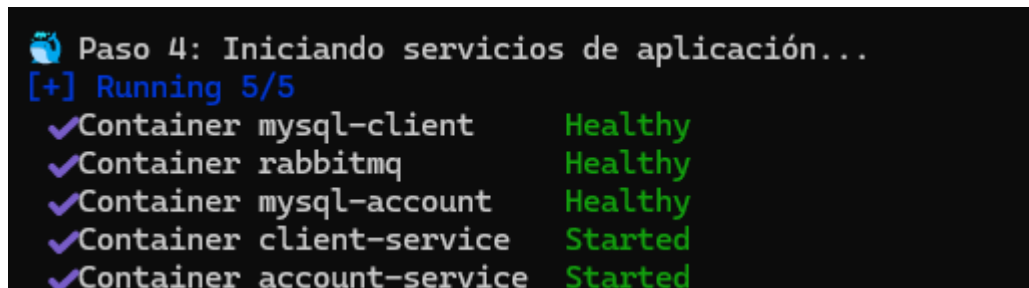
Abrir consola de Powershell ubicarse en la raíz del proyecto bank-microservices y ejecutar el siguiente comando para realizar el despliegue.

`./deploy.sh`

El script `deploy.sh` realiza las siguientes acciones:

- Detiene y elimina contenedores y volúmenes existentes.
- Construye las imágenes de Docker para `mysql-account`, `mysql-client`, `account-service`, `client-service` y `rabbitmq`.
- Inicia los contenedores de las bases de datos MySQL.
- Espera a que las bases de datos estén listas.
- Inicia los servicios `account-service` y `client-service`.
- Verifica el estado final de los servicios.

En este punto estar atento a la consola para ver cada proceso, una vez termina el despliegue muestra el estado de los servicios ver imagen:



```
Paso 4: Iniciando servicios de aplicación...
[+] Running 5/5
✓Container mysql-client      Healthy
✓Container rabbitmq         Healthy
✓Container mysql-account     Healthy
✓Container client-service    Started
✓Container account-service   Started
```

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

```
✓ Despliegue completado. Verificando salud de los servicios...
<!-- HTML for static distribution bundle build -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Swagger UI</title>
    <link rel="stylesheet" type="text/css" href="./swagger-ui.css" />
    <link rel="stylesheet" type="text/css" href="index.css" />
    <link rel="icon" type="image/png" href="./favicon-32x32.png" sizes="32x32" />
    <link rel="icon" type="image/png" href="./favicon-16x16.png" sizes="16x16" />
  </head>
```

Una vez se vea lo que muestra la imagen anterior ya se puede acceder a la aplicación para ver la documentación y probar.

URLs de los servicios corriendo:

<http://localhost:8082/api/swagger-ui/index.html>

<http://localhost:8081/api/swagger-ui/index.html>

Configuración

La configuración de las bases de datos MySQL se encuentra en el archivo [docker-compose.yml](#). Las variables de entorno para la conexión a las bases de datos se definen en este mismo archivo.

- mysql-account:
 - Puerto: 3307
 - Base de datos: account_db
 - Usuario: app_user
 - Contraseña: userpassword
- mysql-client:
 - Puerto: 3308
 - Base de datos: client_db
 - Usuario: app_user
 - Contraseña: userpassword

Prueba Desarrollador Java – Félix Alegría Loango

Empresa: Devsu

Repositorio: <https://github.com/Alegria2016/BANK-MICROSERVICES>

Verificación de Salud

Una vez realizado el despliegue verifica documentación técnica de OpenApi de los servicios en: <http://localhost:8081/api/swagger-ui/index.html> y <http://localhost:8082/api/swagger-ui/index.html> o ejecútelo el comando a continuación para confirmar que todos los servicios están en funcionamiento:

```
./health-check.sh
```