Integrante: Brayan daniel fúquene sandoval cod:202114796

Manual Técnico – Gestor de Clientes y Órdenes (Arquitectura de Microservicios)

Proyecto académico con 3 microservicios + API Gateway + Eureka + Frontend.
Tech: Spring Boot (Java), FastAPI (Python), Node/Express (JS), Oracle, Postgres, MongoDB, React (Vite).

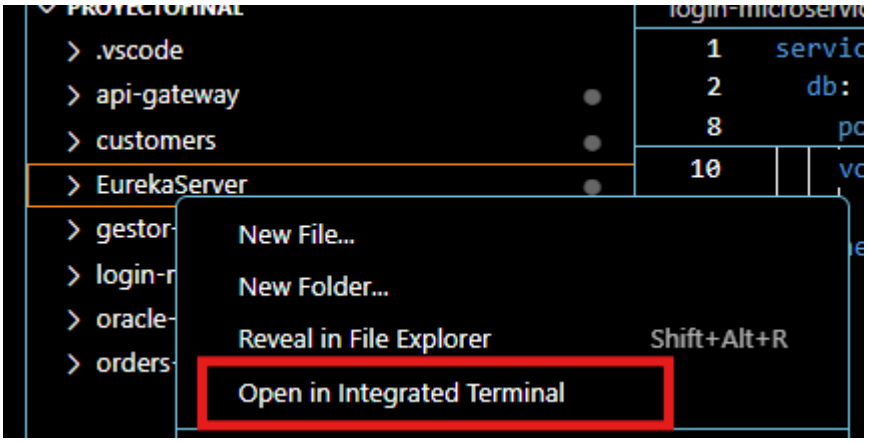
Componentes	Puerto	Url	Notas
Eureka	8761	http://localhost:8761/	Dashboard de descubrimiento
API Gateway	8080	http://localhost:8080	Expone /customer/** y /order/** → StripPrefix=1
Login Ms	8000	http://localhost:8000	FastAPI + Postgres (Docker)
Customer Ms	8081	http://localhost:8081	Spring Boot + Oracle XE (local)
order MS	8082	http://localhost:8082	Spring Boot + Oracle XE (local)
Front (Vite)	5173	http://localhost:5173	Variables .env para apuntar a login y gateway

1) Requisitos previos

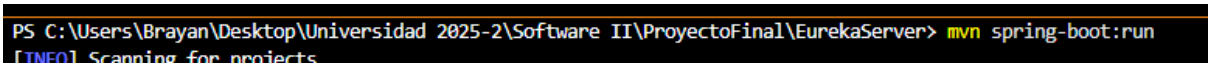
- Java 21 (como en tus logs) y Maven 3.8+
- Node.js 20+ y npm
- Python 3.11+ (solo si corres login sin Docker; con Docker no hace falta)
- Docker Desktop (o Docker Engine) y Docker Compose
- Oracle 21c XE (local) con PDB FREEPDB1 — ya lo estás usando en

2) Eureka

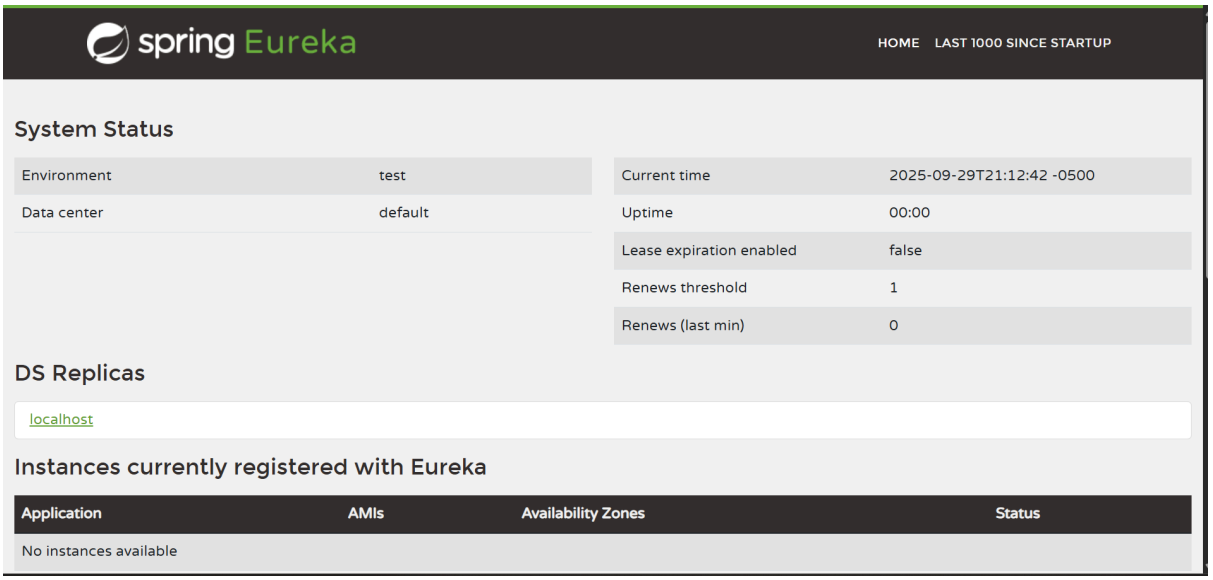
Paso 1: vamos a levantar el servidor de eureka , para esto abrimos la terminal desde la carpeta **EurekaServer.**



Paso 2: después de abrir la terminal vamos a ejecutar el siguiente comando **mvn spring-boot:run** para correr el servidor en el **puerto 8761**.



Paso 3: una vez ejecutado el comando, pasamos a verificar que este funcionando el servido de Eureka.

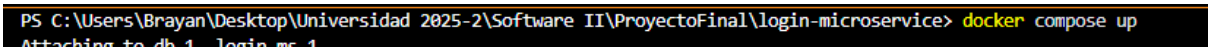


3) LoginMicroservice

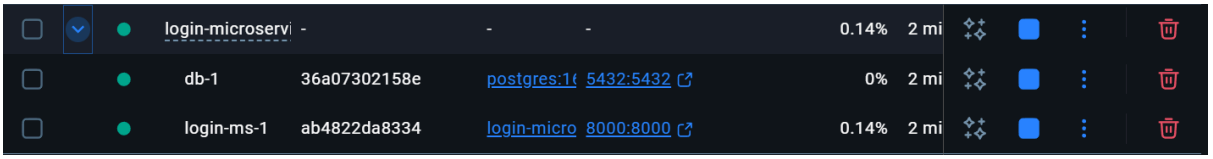
Para el microservicio del login en este caso usamos **python** para el microservicio y **Postgres** para la persistencia de datos. Todo esto dockerizado.

Paso#1:
Paso 1:**verificamos** que tengamos **docker desktop** abierto ya que es necesario para montar el servicio.

Paso 2: vamos a abrir la terminal desde la carpeta **login-microservice**
y vamos a ejecutar el siguiente comando **docker compose up**

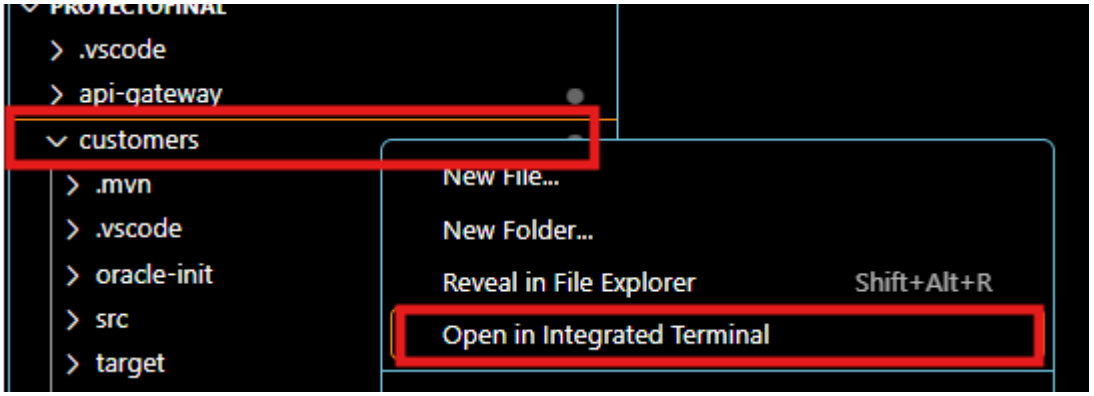


Paso 3: una vez ejecutado este comando, verificamos que se haya levantado los **contenedores** tanto del servicio como de la base de datos.

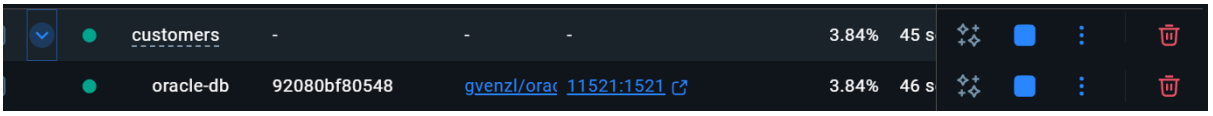


4) Customer MS

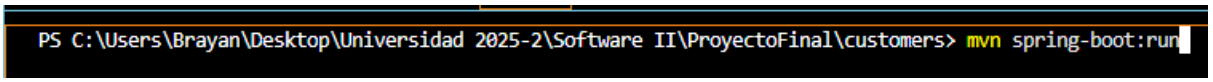
Paso 1: abrimos la terminal desde la carpeta **customers**



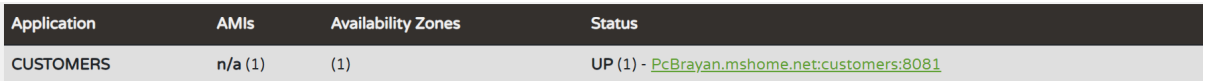
Paso 2: después de abrir la terminal, vamos a ejecutar los siguientes comandos
-**docker compose up:** para levantar la base de datos de customers, en nuestro en oracle.



Paso 3: después de levantar la base de datos, vamos de nuevo a la terminal dentro de la carpeta **customers** y ejecutamos el siguiente comando
-**mvn spring-boot:run:**para levantar el servicio relacionado con la gestión de customers.



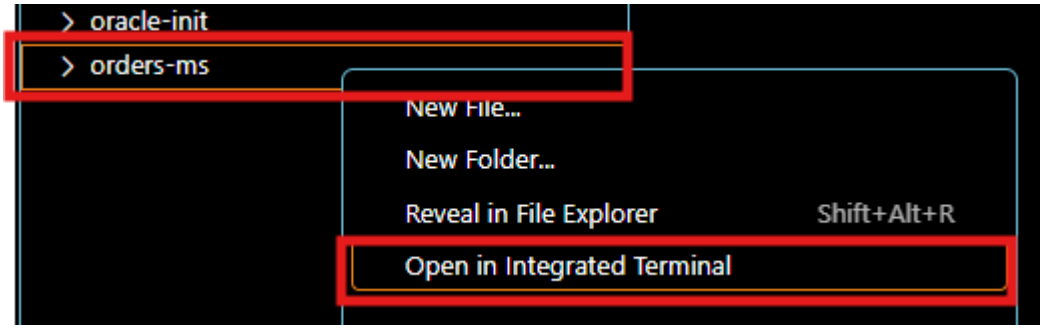
paso 4: verificamos que el servicio este registrado en Eureka



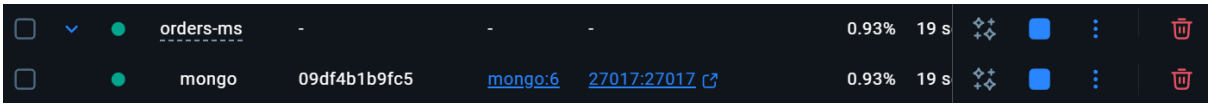
Application	AMIs	Availability Zones	Status
CUSTOMERS	n/a (1)	(1)	UP (1) - PcBrayan.mshome.net:customers:8081

5) order MS

Paso 1:abrimos la terminal desde la carpeta **orders-ms**



Paso 2: después de abrir la terminal, vamos a ejecutar los siguientes comandos
-**docker compose up:** para levantar la base de datos de orders, en Mongo DB.



Paso 3: después de levantar la base de datos, vamos de nuevo a la terminal dentro de la carpeta **orders-ms** y ejecutamos el siguiente comando
-**npm start :**para levantar el servicio relacionado con la gestión de orders.

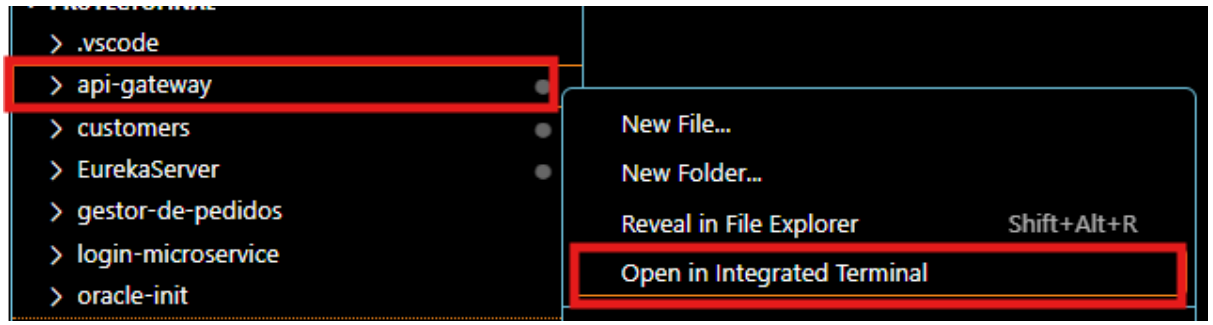
```
C:\Users\Brayan\Desktop\Universidad 2025-2\Software II\ProyectoFinal\orders-ms>npm start

> orders-ms@1.0.0 start
> node server.js

[orders-ms] Mongo conectado
[orders-ms] escuchando en :8082
registered with eureka:  ORDERS/localhost
[orders-ms] Registrado en Eureka como ORDERS
```

Paso 4: verificamos que el servicio este registrado en Eureka.

Application	AMIs	Availability Zones	Status
CUSTOMERS	n/a (1)	(1)	UP (1) - PcBrayan.mshome.net:customers:8081
ORDERS	n/a (1)	(1)	UP (1) - localhost

6) Api GetWay

Paso 1:abrimos la terminal desde la carpeta **api-gatewar**



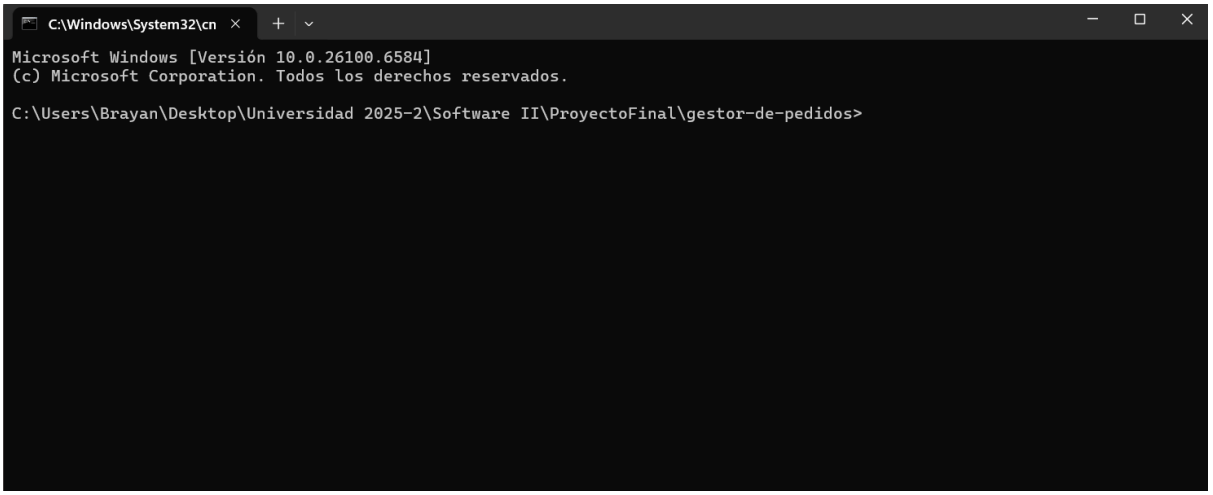
Paso 2: después de abrir la terminal, vamos a ejecutar los siguientes comandos -**mvn spring-boot:run :** para levantar nuestra ApiGetway.

Paso 3:verificamos que el servicio este registrado en Eureka.

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - PcBrayan.mshome.net:api-gateway:8080
CUSTOMERS	n/a (1)	(1)	UP (1) - PcBrayan.mshome.net:customers:8081
ORDERS	n/a (1)	(1)	UP (1) - localhost

7)FRONT (Vite)

Paso 1: abrimos la consola desde la carpeta **gestor-pedidos**



Paso 2: después de abrir la terminal, vamos a ejecutar el siguiente comando -**npm run dev** para levantar nuestro front.

```
C:\Users\Brayan\Desktop\Universidad 2025-2\Software II\ProyectoFinal\gestor-de-pedidos>npm  run dev

> gestor-de-pedidos@0.0.0 dev
> vite


  VITE v7.1.7  ready in 681 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Paso 3: ingresamos a la ruta donde se levantó nuestro front para corroborar que esté funcionando correctamente

una vez ingresemos el link deberá salir esta página.

8) Seguridad, roles y contraseñas (académico)

Postgres (Login)
Usuario: postgres / Pass: postgres (en Docker Compose).

Oracle (Customers)
Usuario: CUSTOMERS / Pass: customers.
Privilegios mínimos: CREATE SESSION, CREATE TABLE (o RESOURCE), cuota en tablespace.

Mongo (Orders)
Sin auth (entorno local académico).

CORS
Gateway permite http://localhost:5173.
Login FastAPI permite http://localhost:5173.
Front no guarda tokens, solo gd_session en LocalStorage (maqueta académica).