Gestor de Pedidos Distribuido

Brayan Daniel Fuquene Sandoval

6/10/2025

Carlos Arenas

1. Introducción

El presente documento describe la arquitectura del sistema desarrollado bajo el modelo **4+1 de Philippe Kruchten**, el cual aborda la solución desde diferentes perspectivas complementarias.

El sistema implementa una arquitectura de **microservicios** que gestionan autenticación, clientes y pedidos, conectados mediante un **API Gateway** y registrados en un **servidor Eureka**.

El frontend está desarrollado en **React (Vite)**, comunicándose con el Gateway para centralizar las peticiones a los microservicios.

2. Vista Lógica (Diagrama de Componentes UML)

Objetivo: mostrar los principales componentes del sistema y cómo se comunican entre sí.

Componentes Principales:

Frontend (React):

- Interfaz de usuario.
- Llama a los endpoints del API Gateway.

API Gateway (Spring Cloud Gateway):

- Encaminador principal de peticiones HTTP.
- Aplica StripPrefix y redirige a los microservicios adecuados.
- Implementa CORS global.

Eureka Server:

Servicio de descubrimiento donde los microservicios se registran.

Microservicio Login (FastAPI + PostgreSQL):

- Endpoint /createuser y /authuser.
- Persistencia en base de datos PostgreSQL.

Microservicio Customers (Spring Boot + Oracle):

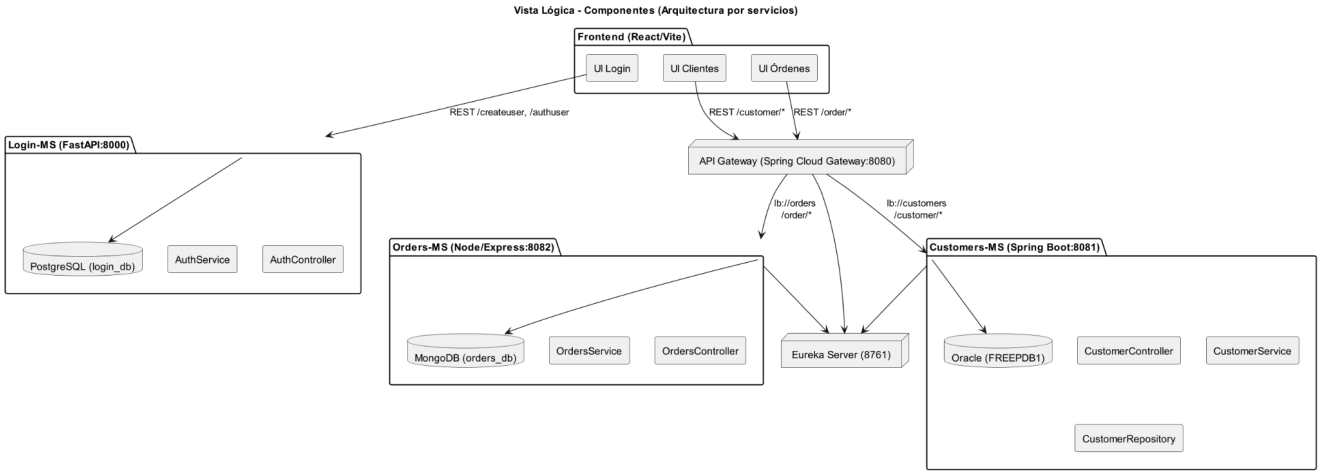
- Endpoints /customer/createcustomer y /customer/findcustomerbyid/{document}.
- Gestiona la información de los clientes.

Microservicio Orders (Node.js + MongoDB):

- Endpoints /order/createorder, /order/updateorderstatus, /order/findorderbycustomerid/{customerid}.
- Cada orden se asocia a un cliente validado a través del Gateway.

Bases de Datos:

- PostgreSQL (usuarios)
- Oracle (clientes)
- MongoDB (órdenes)



3. Vista de Procesos (Flujo de Ejecución)

Objetivo: mostrar cómo interactúan los componentes durante la ejecución del sistema.

Caso: Creación de una Orden

1. El usuario inicia sesión en el **frontend**, que envía las credenciales al **Login-MS** a través del **API Gateway**.

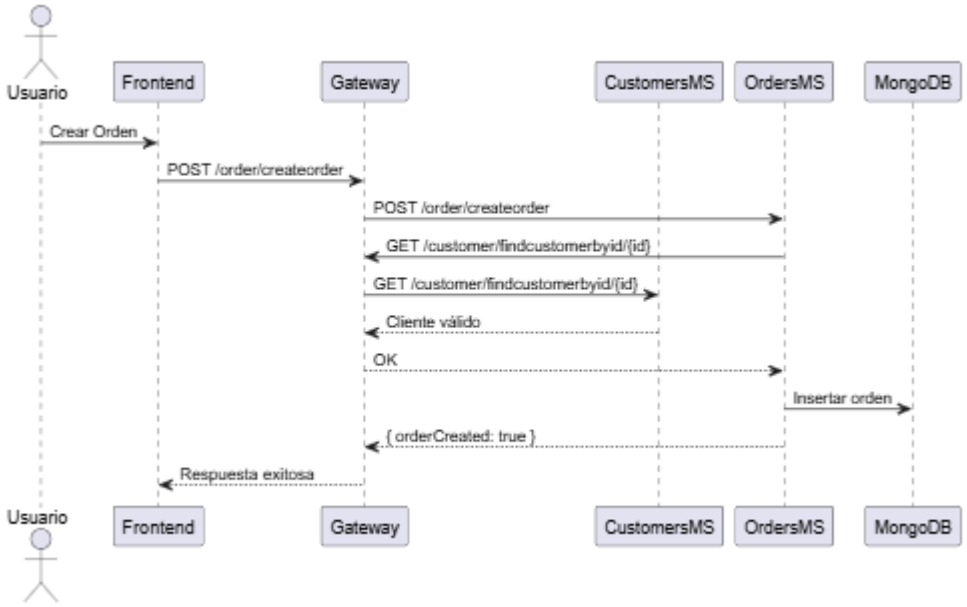
2. Una vez autenticado, accede al módulo de clientes.

3. El frontend consulta los datos del cliente usando **`/customer/findcustomerbyid/{document}`.**

4. Al crear una orden, el **Orders-MS** valida que el `customerid` exista llamando al **Customers-MS** a través del **Gateway**.

5. Si el cliente es válido, la orden se guarda en **MongoDB**.

6. Todos los servicios están registrados y descubiertos dinámicamente por **Eureka**



4. Vista de Desarrollo – Diagrama de Paquetes (Estructura de código)

El sistema está compuesto por **tres microservicios backend**, un **frontend React**, y varias **bases de datos independientes**, comunicados mediante un **API Gateway** y registrados en un **servidor Eureka**.

Componentes principales:

- **Frontend (React + Vite)**
    - Implementa la interfaz de usuario.
    - Desarrollado con React 18 y Vite.
    - Consume los endpoints del Gateway para los microservicios:
        - `/customer/**` → servicio de clientes.
        - `/order/**` → servicio de órdenes.
        - `/authuser` y `/createuser` → servicio de login.

Login-MS (FastAPI + PostgreSQL)

- Provee autenticación básica.
- Endpoints:
    - POST **`/createuser`** — crea usuario.
    - POST **`/authuser`** — valida credenciales.
- Base de datos: **PostgreSQL 16** (contenedor Docker).

Customers-MS (Spring Boot + Oracle)

- Gestiona los clientes.
- Usa JPA + Hibernate con conexión a **Oracle DB**.
- Endpoints:
    - POST **`/customer/createcustomer`**
    - GET **`/customer/findcustomerbyid/{document}`**

Orders-MS (Express + MongoDB)

- Gestiona las órdenes de los clientes.
- Implementado con Node.js, Express y Mongoose.
- Endpoints:
    - POST **`/order/createorder`**
    - PUT **`/order/updateorderstatus`**
    - GET **`/order/findorderbycustomerid/{id}`**
- Se registra en **Eureka** y valida el `customerid` mediante el **API Gateway**.

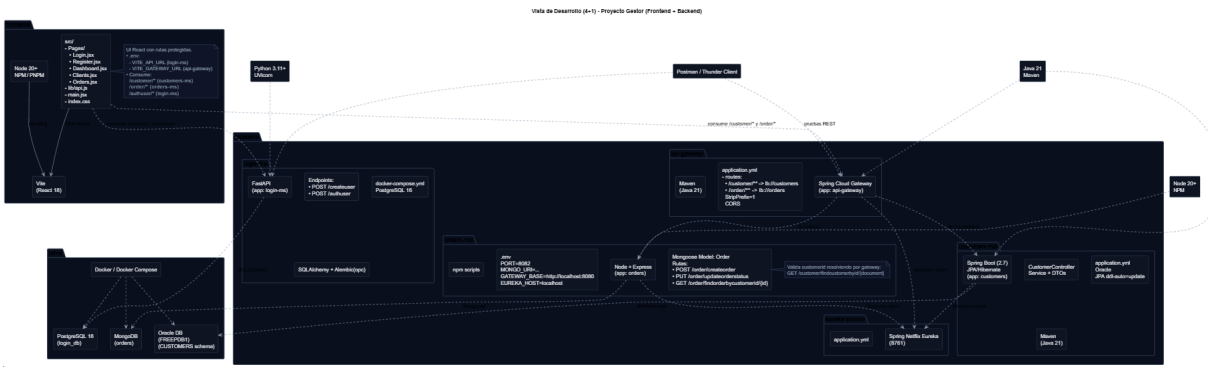
API Gateway (Spring Cloud Gateway)

- Expone rutas unificadas hacia los microservicios.
- Aplica `StripPrefix=1` y reglas CORS globales.

- Configurado en `application.yml`:

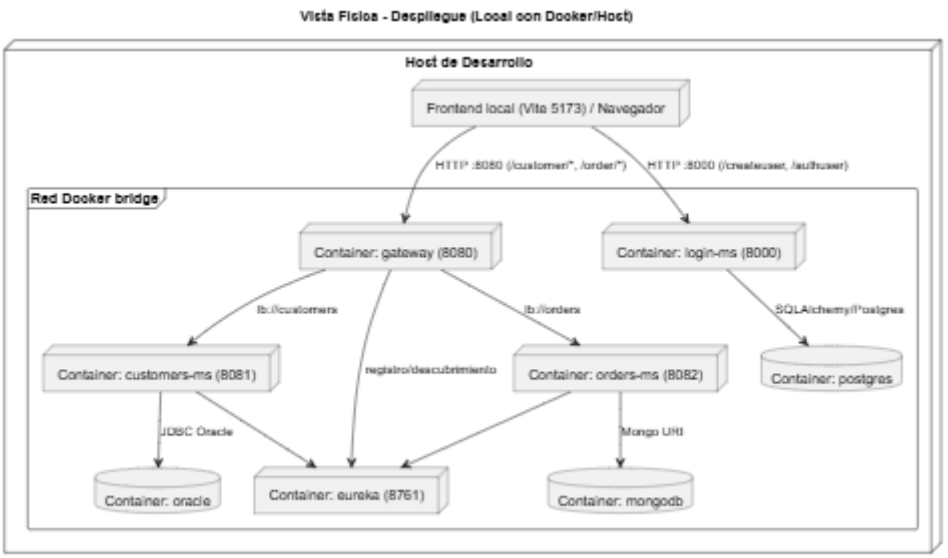
Eureka Server

- Registro de servicios (Service Discovery).
- Todos los microservicios se registran automáticamente en `http://localhost:8761`.



5. Vista Física (Despliegue UML)

Objetivo: mostrar los nodos físicos o contenedores donde se despliega el sistema.



6. Vista de Casos de Uso

Actores:

- **Usuario**: interactúa con el sistema desde el navegador.
- **Administrador (opcional)**: puede gestionar clientes y revisar órdenes.

Casos Principales:

1. Registrar usuario (**`/createuser`** → Login-MS)
2. Autenticar usuario (**`/authuser`** → Login-MS)
3. Crear cliente (**`/createcustomer`** → Customers-MS)
4. Consultar cliente (**`/findcustomerbyid`** → Customers-MS)
5. Crear orden (**`/createorder`** → Orders-MS)
6. Actualizar estado (**`/updateorderstatus`** → Orders-MS)
7. Listar órdenes por cliente (**`/findorderbycustomerid`** → Orders-MS)

Vista de Casos de Uso - Escenarios del sistema