

# **Proyecto API-FIRST OAS Microservicios con arquitectura hexagonal, API gateway, Eureka y CQRS**

**@JosepeDevs Repositorio: PcRepair**

[JosepeDevs/PcRepair: OAS Microservices hexagonal architecture with Java and Spring \(github.com\)](https://github.com/JosepeDevs/PcRepair)

Qué encontrarás aquí:

**#Java #Spring #SpringBoot #Eureka #API Gateway #CQRS  
#Arquitectura hexagonal #API-first #OpenAPI  
#AWS RDS #PostgreSQL #MongoDB #Principios SOLID**

**Disclaimer:** Este proyecto es una muestra, ya que CQRS y varios de los patrones de implementación realizados añaden un grado de complejidad que un proyecto como este realmente NO necesitaría ya que solo añadiría complejidad innecesaria.

## Microservicios y sus casos de uso

### ClientManager

- Alta cliente
- Editar cliente
- Actualizar cliente
- Borrado (lógico) cliente

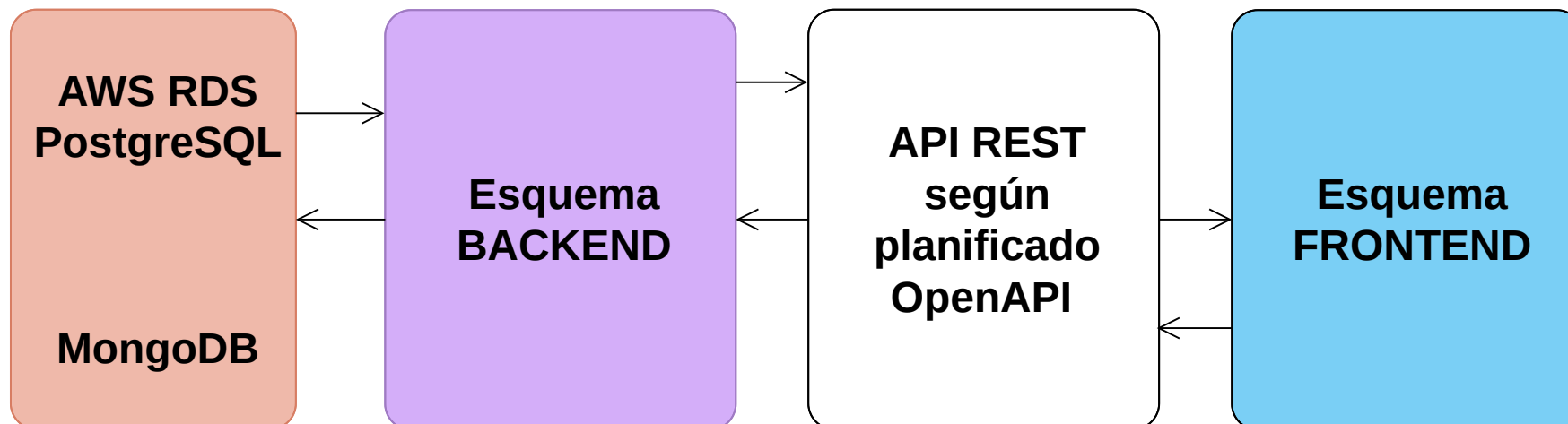
### RepairCatalogueManager

- Alta posible reparación
- Editar posible reparación
- Buscar posible reparación
- Borrado (lógico) posible reparación

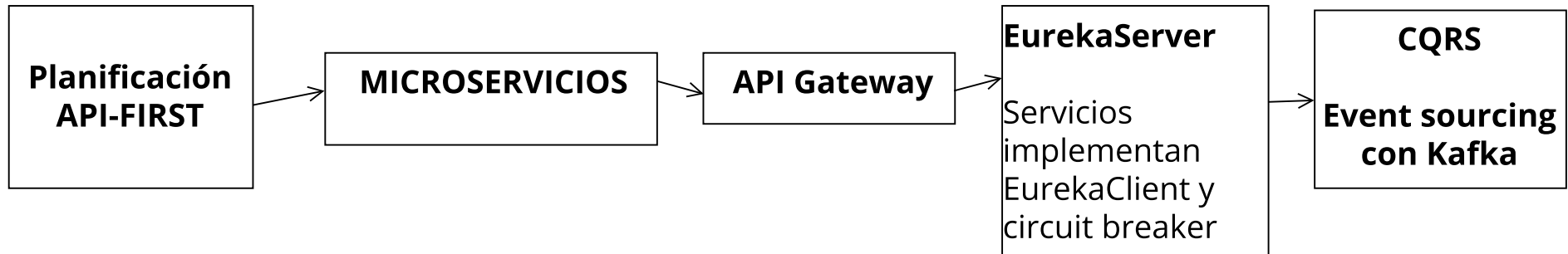
### RepairOrderContentManager

- Alta Pedido
- Editar Pedido
- Buscar Pedido
- Borrado (lógico) Pedido

## ESQUEMA INFRAESTRUCTURA



# ETAPAS PROYECTO



## API-first: OpenAPI Specification

Fragmento del JSON

Definido en JSON (disponible en repositorio GitHub carpeta "About")

Datos generales  
Servidores (ficticios)  
Seguridad

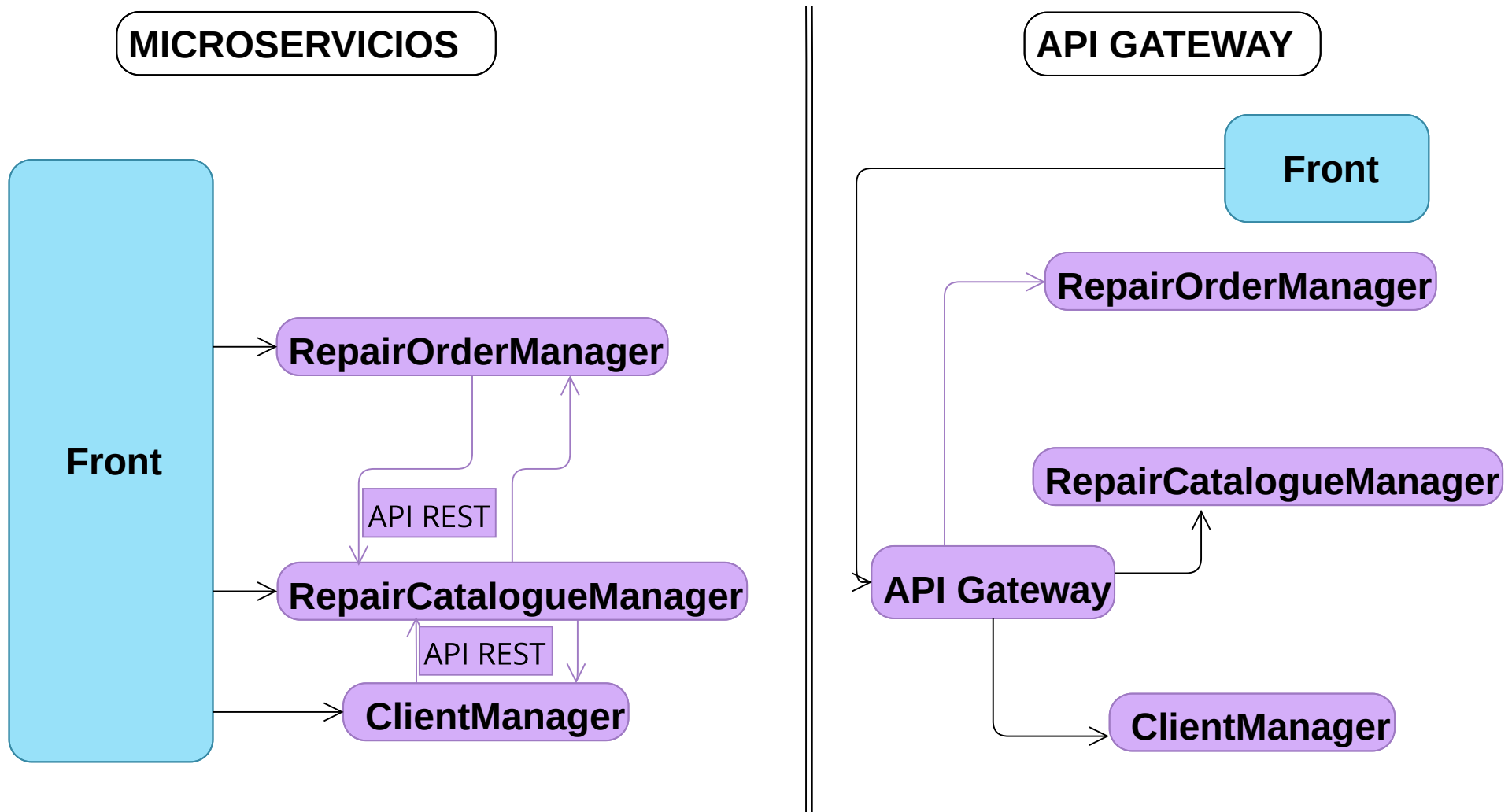
Componentes: Schemas, Esquemas de seguridad

API de cada microservicio--> URI, parametros, summary, respuesta, contenido.

```
"paths": {
  "/users": {
    "get": {
      "summary": "Obtener todos los usuarios",
      "responses": {
        "200": {
          "description": "Lista de usuarios",
          "content": {
            "application/json": {
              "schema": {
                "type": "array",
                "items": {

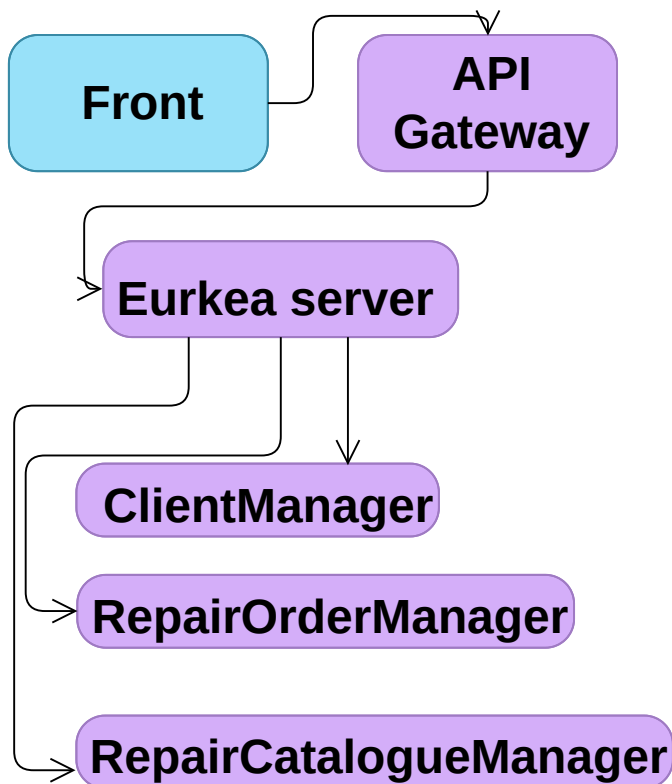
```

# Etapas proyecto: ESQUEMA BACKEND

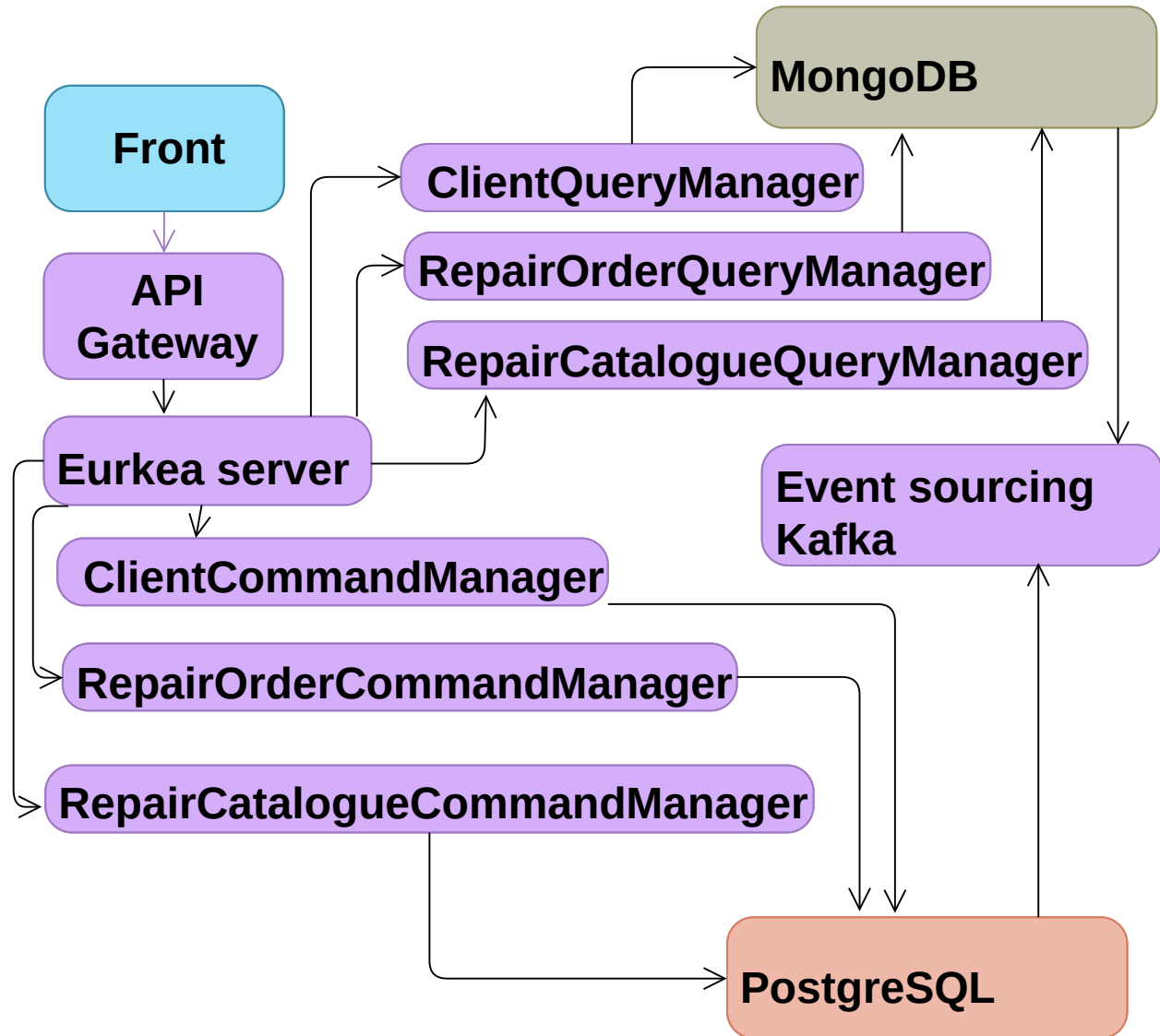


# Etapas proyecto: ESQUEMA BACKEND

MS implementan Eureka  
Client y circuit breaker



CQRS y event sourcing



# PLAN BASES DE DATOS

PostgreSQL



MongoDB

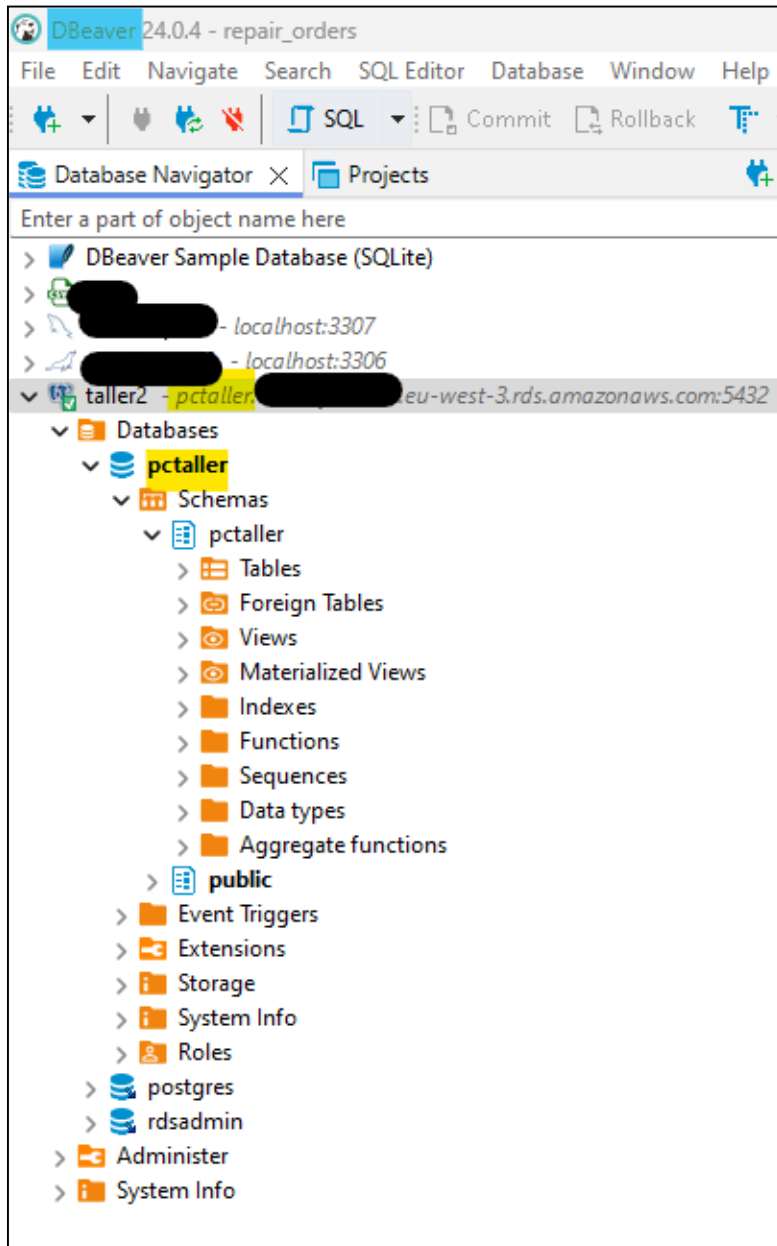
pendiente

pendiente

pendiente

# Gestión con Dbeaver

# PostgreSQL en AWS RDS



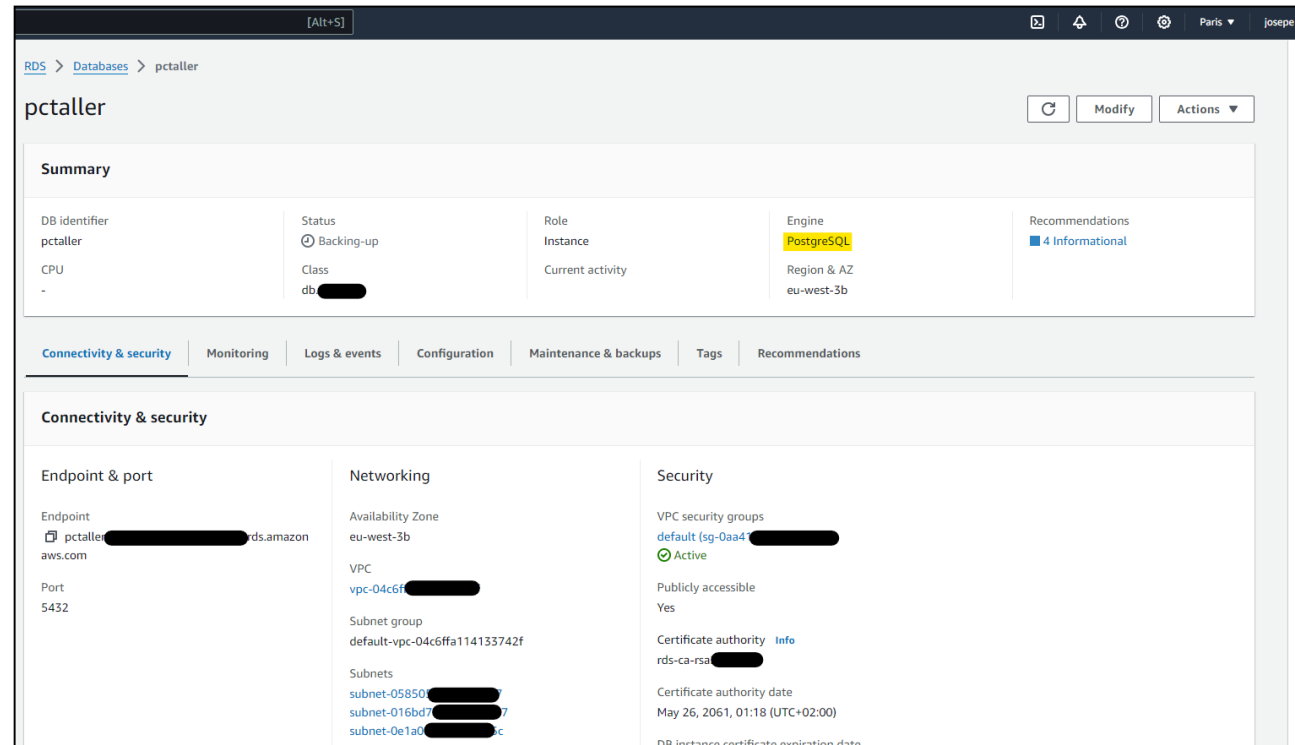
DBeaver 24.0.4 - repair\_orders

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects

Enter a part of object name here

- > DBeaver Sample Database (SQLite)
- > [redacted] - localhost:3307
- > [redacted] - localhost:3306
- > > taller2 - pctaller [redacted] eu-west-3.rds.amazonaws.com:5432
  - > Databases
    - > pctaller
      - > Schemas
        - > pctaller
          - > Tables
          - > Foreign Tables
          - > Views
          - > Materialized Views
          - > Indexes
          - > Functions
          - > Sequences
          - > Data types
          - > Aggregate functions
        - > public
        - > Event Triggers
        - > Extensions
        - > Storage
        - > System Info
        - > Roles
      - > postgres
      - > rdsadmin
      - > Administrator
      - > System Info



RDS > Databases > pctaller

pctaller

Summary

DB identifier pctaller	Status Backing-up	Role Instance	Engine PostgreSQL	Recommendations 4 Informational
CPU -	Class db [redacted]	Current activity	Region & AZ eu-west-3b	

Connectivity & security Monitoring Logs & events Configuration Maintenance & backups Tags Recommendations

Connectivity & security

Endpoint & port	Networking	Security
Endpoint pctaller [redacted] rds.amazonaws.com	Availability Zone eu-west-3b	VPC security groups default (sg-0aa4 [redacted])
Port 5432	VPC vpc-04c6f [redacted]	Publicly accessible Yes
	Subnet group default-vpc-04c6ffa114133742f	Certificate authority Info rds-ca-rsa [redacted]
	Subnets subnet-05850 [redacted] subnet-016bd7 [redacted] subnet-0e1a0 [redacted]	Certificate authority date May 26, 2061, 01:18 (UTC+02:00)
		DB instance certificate expiration date

## Buenas prácticas seguidas:

- Principios SOLID: Clases con una sola responsabilidad, inyección de dependencias, interfaces con encapsulación en mente, clases abiertas a extensión.
- Implementación de Excepciones propias, no exponer info de app.
- ValueObjects para las verificaciones y código limpio.
- Uso de librerías para código legible y mantenible (Lombok)
- Arquitectura hexagonal: Capas de Dominio, Aplicación e Infraestructura para un proyecto reactivo y ágil, además de mejor mantenibilidad.
- Testing unitario
- Clean Code, nombres descriptivos.

**ESQUEMA FRONTEND**  
**(pendiente, más adelante,**  
**Angular seguramente)**



# Más sobre el desarrollador/autor:

## José Pedro Navarro Vicente

### @JosepeDevs

### Perfil LinkedIn:

<https://www.linkedin.com/in/josepedronavarro/>

### Perfil GitHub:

[JosepeDevs \(JosepeDevs\) \(github.com\)](https://github.com/JosepeDevs)