**Ecosistema de Microservicios:**

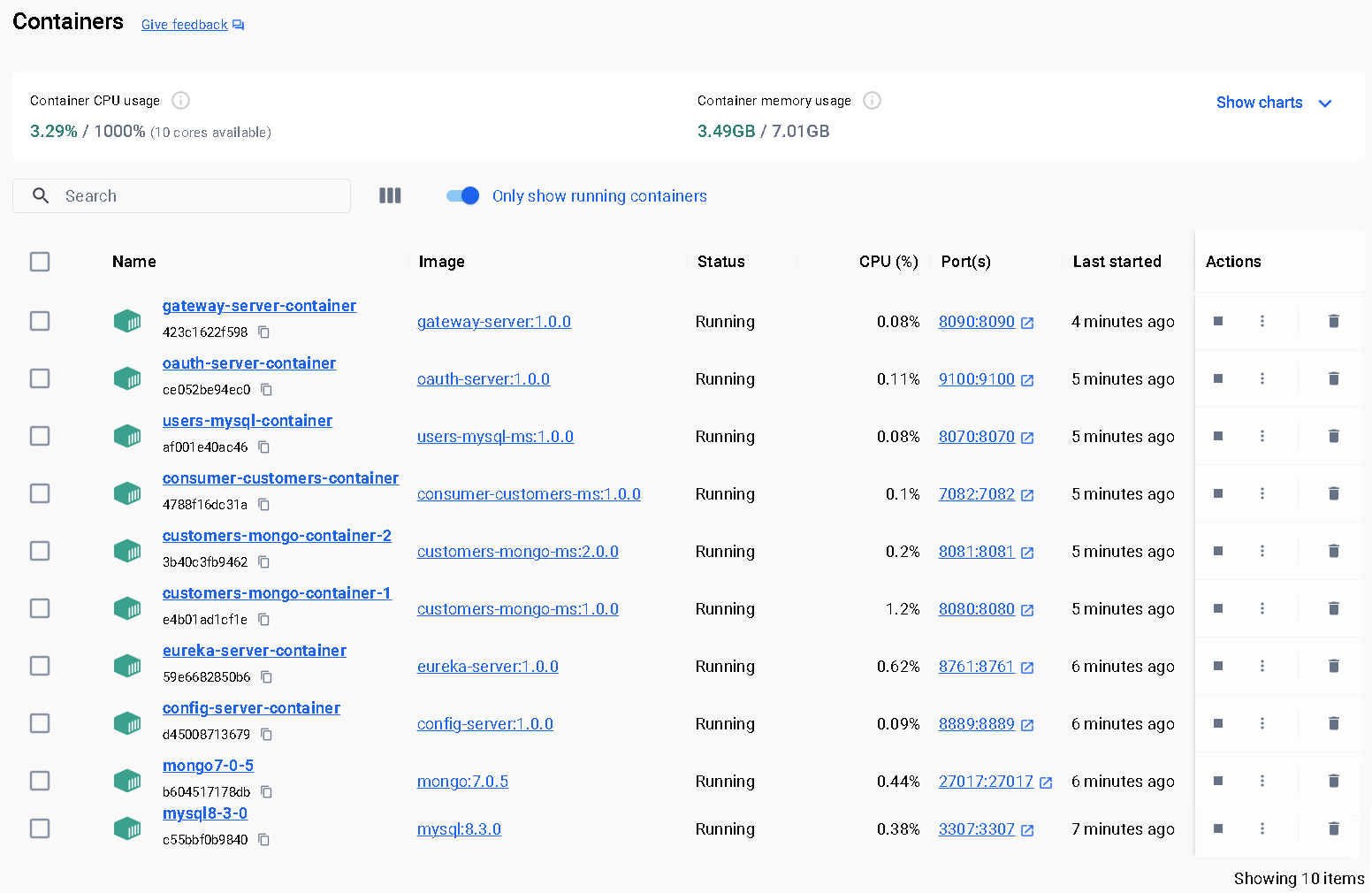
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Servidor BD / Proyecto Java** | **Config Client** | **Eureka Client** | **Nombre registrado en Eureka** | **Port** | **Tiene BD** | **Nombre Imagen** | **Nombre Contenedor** |
| Mongo Server |  |  |  | 27017 |  | mongo:7.0.5 | mongo7-0-5 |
| MySQL Server |  |  |  | 3307 |  | mysql:8.3.0 | mysql8-3-0 |
| config-server |  |  |  | 8889 |  | config-server:1.0.0 | config-server-container |
| eureka-server |  |  |  | 8761 |  | eureka-server:1.0.0 | eureka-server-container |
| customers-mongo-ms (1) |  | X | mclientes | 8080 | eshop  (Mongo) | customers-mongo-ms:1.0.0 | customers-mongo-container-1 |
| customers-mongo-ms (2) |  | X | mclientes | 8081 | eshop-replica  (Mongo) | customers-mongo-ms:2.0.0 | customers-mongo-container-2 |
| consumer-customers-ms | X | X | cclientes | 7082 |  | consumer-customers-ms:1.0.0 | consumer-customers-container |
| users-mysql-ms |  | X | servicio-usuarios | 8070 | seguridad  (MySQL) | users-mysql-ms:1.0.0 | users-mysql-container |
| oauth-server | X | X | servicio-oauth | 9100 |  | oauth-server:1.0.0 | oauth-server-container |
| gateway-server | X | X | servicio-gateway-server | 8090 |  | gateway-server:1.0.0 | gateway-server-container |

**Diagrama de Microservicios y Bases de Datos en Docker**

Diagrama

Descripción generada automáticamente

**Docker Containers**



**Gráfico, Gráfico de burbujas

Descripción generada automáticamente**

**Diagrama Entidad Relación (BD)**

SGBD: MySql

BD: seguridad

Tablas:

Diagrama

Descripción generada automáticamente

SGBD: Mongo

BD: eshop, eshop-replica

Tablas:

|  |  |
| --- | --- |
| customer | |
| id | (PK) Identificador único de customer. No es autogenerado. |
| firstName | Nombre del cliente |
| lastName | Apellido del cliente |
| email | Correo del cliente |
| address | Dirección del cliente |
| phoneNumber | Teléfono del cliente |

**Procedimiento para armar el ambiente**

1. Descargar las siguientes imágenes:

docker pull mysql:8.3.0

docker pull mongo:7.0.5

docker pull openjdk:17

docker pull openjdk:11

1. Crear el network de docker:

docker network create estebancoder-network

1. Correr los contenedores de los servidores de BD:

docker run --name mysql8-3-0 -p 3307:3307 --network estebancoder-network -e MYSQL\_ROOT\_PASSWORD=password -e MYSQL\_DATABASE=seguridad -e MYSQL\_TCP\_PORT=3307 -d mysql:8.3.0

docker run --name mongo7-0-5 -p 27017:27017 --network estebancoder-network -d mongo:7.0.5

1. Setear la variable de entorno JAVA\_HOME
2. Ingresa desde la consola DOS a la raíz de la carpeta de cada proyecto JAVA y ejecutar el siguiente comando:

mvnw.cmd clean package -DskipTests

1. Crear la imagen Docker para cada proyecto JAVA: ingresar desde la consola DOS a la raíz de la carpeta (donde está ubicado el archivo Dockerfile) de cada proyecto JAVA y ejecutar el comando:

docker build -t config-server:1.0.0 .

docker build -t consumer-customers-ms:1.0.0 .

docker build -t customers-mongo-ms:1.0.0 .

docker build -t customers-mongo-ms:2.0.0 .

docker build -t eureka-server:1.0.0 .

docker build -t gateway-server:1.0.0 .

docker build -t oauth-server:1.0.0 .

docker build -t users-mysql-ms:1.0.0 .

Nota:

Para generar la imagen de la segunda instancia de clientes: **customers-mongo-ms:2.0.0**

- modificar en el application.properties:

#spring.data.mongodb.uri= mongodb://mongo7-0-5:27017/eshop

#server.port=8080

**spring.data.mongodb.uri= mongodb://mongo7-0-5:27017/eshop-replica**

**server.port=8081**

- modificar el Dockerfile:

FROM openjdk:17

#EXPOSE 8080

**EXPOSE 8081**

ADD ./target/customers-mongo-ms-1.0.0.jar customers-mongo-ms.jar

ENTRYPOINT ["java","-jar","/customers-mongo-ms.jar"]

- volver a generar el artifact con maven

- volver a generar la imagen con docker build, con el tag 2.0.0

1. Correr los contenedores (en base a su respectiva imagen) en el siguiente orden:

docker run -p 8889:8889 --name config-server-container --network estebancoder-network config-server:1.0.0

docker run -p 8761:8761 --name eureka-server-container --network estebancoder-network eureka-server:1.0.0

docker run -p 8080:8080 --name customers-mongo-container-1 --network estebancoder-network customers-mongo-ms:1.0.0

docker run -p 8081:8081 --name customers-mongo-container-2 --network estebancoder-network customers-mongo-ms:2.0.0

docker run -p 7082:7082 --name consumer-customers-container --network estebancoder-network consumer-customers-ms:1.0.0

docker run -p 8070:8070 --name users-mysql-container --network estebancoder-network users-mysql-ms:1.0.0

docker run -p 9100:9100 --name oauth-server-container --network estebancoder-network oauth-server:1.0.0

docker run -p 8090:8090 --name gateway-server-container --network estebancoder-network gateway-server:1.0.0

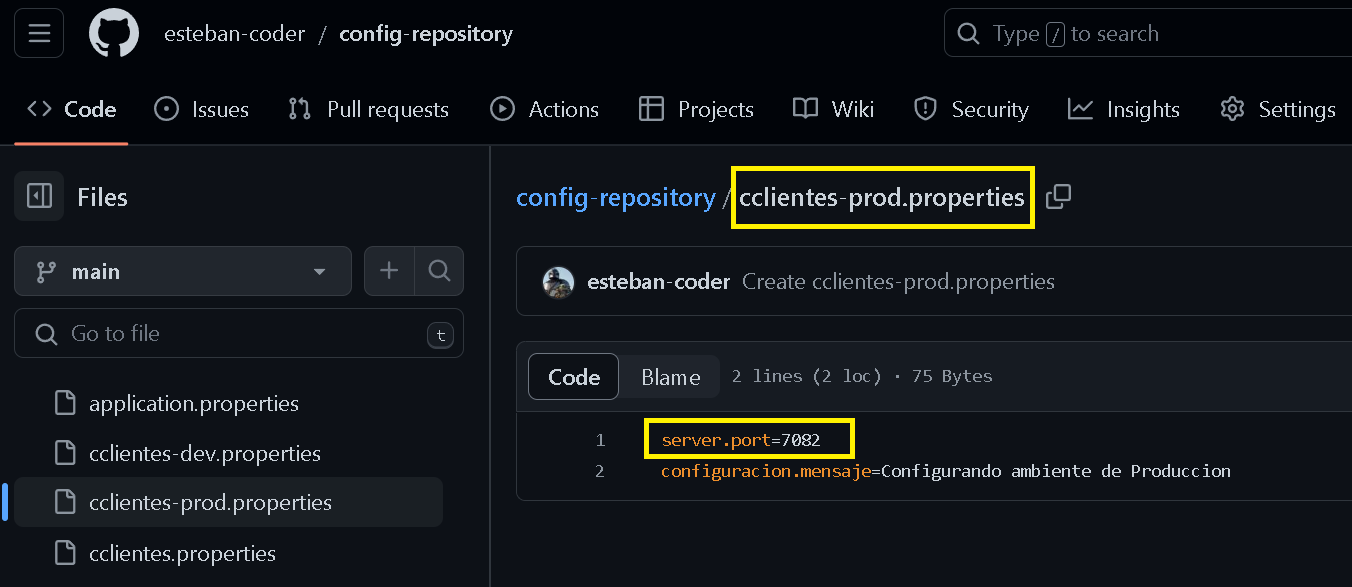
**Validación de microservicios registrados en Eureka Server de Docker**

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**Config Server**

El reposito GIT al cual esta conectado el config server es: <https://github.com/esteban-coder/config-repository>



**Pruebas en Postman**

La primera instancia de customer-mongo:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

La segunda instancia de customer-mongo:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

El consumidor:

Validamos que balancea entre las dos instancias de customer-mongo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Usuarios:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Gateway – ruta: /api/security/oauth/token

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Gateway – ruta: /api/clientes1/customers

Validamos que balancea entre las dos instancias de customer-mongo.

Internamente el Gateway realiza el balanceo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Gateway – ruta: /api/clientes2/customers

Aquí también se balancea entre las dos instancias de customer-mongo, pero el que realiza el balanceo es el consumidor.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente