ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Compte rendu de Contrôle ACE

Ingénierie Informatique et Réseaux

Nom de l'étudiant 1 : ESSAKHI Aimrane

Nom de l'étudiant 2 : MALLOULI Yahya

Nom de l'étudiant 3 : ELAMRI Maryem

Contexte Général

Notre application de recherche de colocataires a été conçue pour simplifier le processus de recherche de partenaires de colocation compatibles.

Dans un marché immobilier en constante évolution, il est devenu essentiel de fournir une plateforme conviviale pour faciliter la connexion entre des individus partageant des intérêts et des styles de vie similaires.

Technologie

- Eureka _ Open Feign
- Docker
- Jenkins
- SonarCloud

Microservice

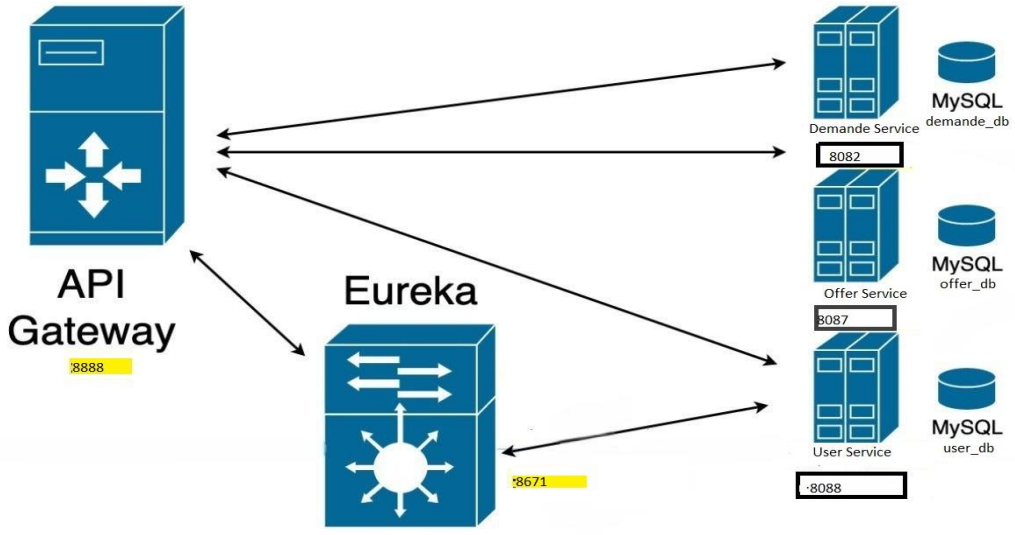
```
Microservice de Gestion des Utilisateurs :
    Entité : User
    Attributs : ID utilisateur, nom, prénom, adresse e-mail, etc.

Microservice de Gestion des Offres de Colocation :
    Entité : Offre de Colocation
    Attributs : ID offre, adresse du logement, règles de vie, préférences de colocation, etc.
    Relation : Un utilisateur (propriétaire) peut créer plusieurs offres de colocation.

Microservice de Gestion des Demandes de Colocation :
    Entité : Demande de Colocation
    Attributs : ID demande, ID utilisateur demandeur, ID offre associée, message d'accompagnement, statut
(acceptée, rejetée, en attente), etc.
    Relations :
    Une demande est associée à un utilisateur demandeur.
    Une demande est liée à une offre de colocation.

Microservice de Consultation des Offres de Colocation :
    Entité : Offre de Colocation (lecture seule)
    Attributs : ID offre, adresse du logement, règles de vie, préférences de colocation, etc.

Serveur Eureka :
Rôle : Registre des services, permettant aux microservices de découvrir les autres services disponibles.
```



Docker

- DockerFile pour Demande

```
FROM maven:3.8.4-openjdk-17 AS builder
WORKDIR /app
COPY ./src ./src
COPY ./pom.xml .
RUN mvn clean package

FROM openjdk:17-jdk-alpine
VOLUME /tmp
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} demande-service.jar
ENTRYPOINT ["java","-jar","/demande-service.jar"]
```

- DockerFile pour User

```
FROM openjdk:17-alpine

WORKDIR /app

COPY target/user_service-0.0.1-SNAPSHOT.jar /app/user-service.jar

EXPOSE 8088

CMD ["java", "-jar", "/app/user-service.jar"]
```

- DockerFile pour User

```
 FROM openjdk:17-alpine

 WORKDIR /app

 COPY target/user_service-0.0.1-SNAPSHOT.jar /app/user-service.jar

 EXPOSE 8088

 CMD ["java", "-jar", "/app/user-service.jar"]
```

- DockerFile pour Eureka Server

```
FROM openjdk:17-alpine

WORKDIR /app

COPY target/server-0.0.1-SNAPSHOT.jar /app/eureka-server.jar

EXPOSE 8761

CMD ["java", "-jar", "/app/eureka-server.jar"]
```

- DockerFile pour GateWay

```
FROM openjdk:17-alpine

WORKDIR /app

COPY target/server-0.0.1-SNAPSHOT.jar /app/eureka-server.jar

EXPOSE 8761

CMD ["java", "-jar", "/app/eureka-server.jar"]
```

- Docker Compose File

```
version: '3'

services:
  eureka-server:
    build:
      context: ./server
    restart: always
    ports:
      - "8761:8761"

  gateway-service:
    build:
      context: ./gateway_service
    ports:
      - "8888:8888"
    depends_on:
      - eureka-server
    environment:
      eureka.client.serviceUrl.defaultZone: http://eureka-server:8761/eureka

  mysql:
    image: mysql:latest
    container_name: mysql
    ports:
      - "3306:3306"
```

```
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: root

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    environment:
      PMA_HOST: mysql
      PMA_PORT: 3306
      MYSQL_ROOT_PASSWORD: root
    ports:
      - "8080:80"

  demande-service:
    build:
      context: ./demande_service
    ports:
      - "8082:8082"
    depends_on:
      - mysql
      - eureka-server
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/demande_db?createDatabaseIfNotExist=true
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: root
      eureka.client.serviceUrl.defaultZone: http://eureka-server:8761/eureka
    healthcheck:
      test: "/usr/bin/mysql --user=root --password=root --execute \"SHOW DATABASES;\""
      interval: 5s
      timeout: 2s
      retries: 100

  offre-service:
    build:
      context: ./offre_service
    ports:
      - "8087:8087"
    depends_on:
      - mysql
      - eureka-server
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/offer_db?createDatabaseIfNotExist=true
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: root
      eureka.client.serviceUrl.defaultZone: http://eureka-server:8761/eureka
    healthcheck:
      test: "/usr/bin/mysql --user=root --password=root --execute \"SHOW DATABASES;\""
      interval: 5s
      timeout: 2s
      retries: 100

  user_service:
    build:
      context: ./user_service
    ports:
      - "8088:8088"
    depends_on:
      - mysql
      - eureka-server
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/user_db?createDatabaseIfNotExist=true
      SPRING_DATASOURCE_PASSWORD: root
      SPRING_DATASOURCE_USERNAME: root
      eureka.client.serviceUrl.defaultZone: http://eureka-server:8761/eureka
    healthcheck:
      test: "/usr/bin/mysql --user=root --password=root --execute \"SHOW DATABASES;\""
```

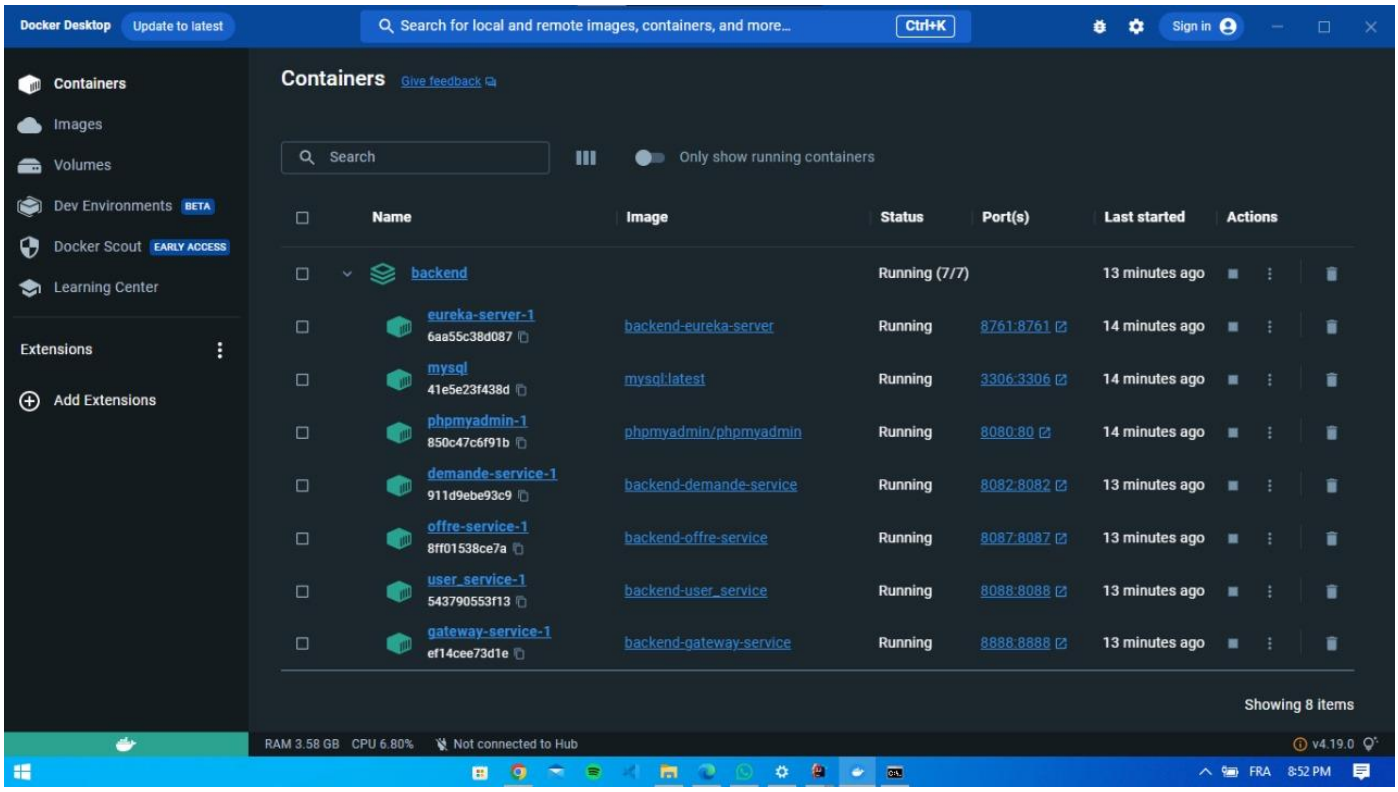
```
    interval: 5s
    timeout: 2s
    retries: 100
```

vous devez positioner dans le directory de service et pour chaque service builder une image

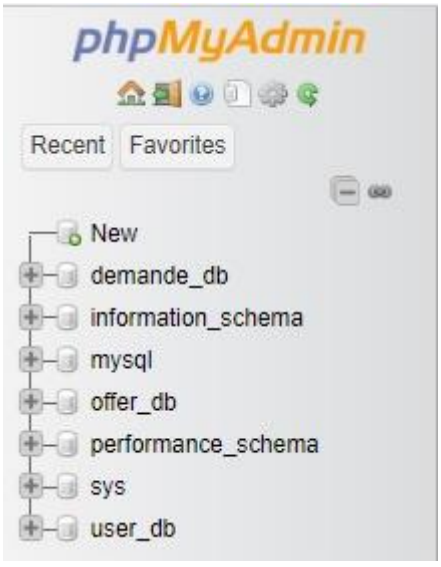
```
    docker buildx build -t service_name:tag .
```

dans le directory racine (backend)

```
    docker-compose up -d
```



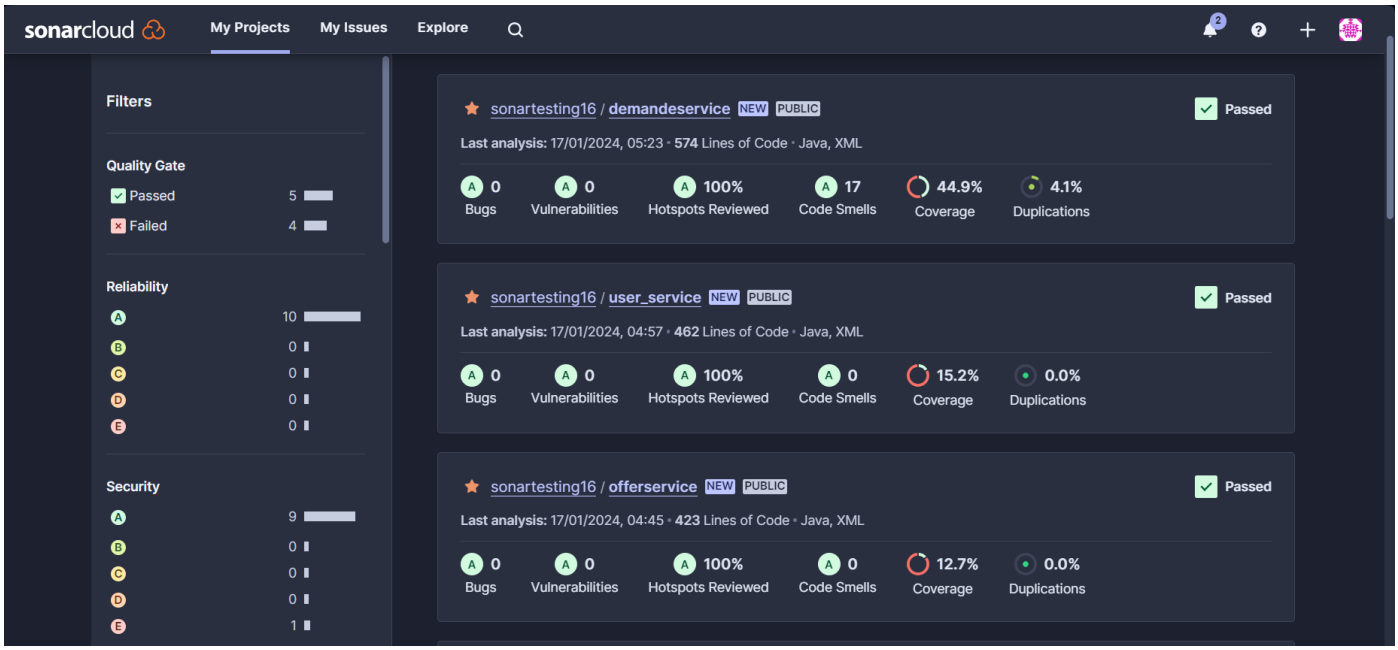
On peut visualizer notre tables on phpmyAdmin :



Eureka Server

Application	AMIs	Availability Zones	Status
DEMANDE_SERVICE	n/a (1)	(1)	UP (1) - 911d9ebe93c9:demande_service:8082
GATEWAY_SERVICE	n/a (1)	(1)	UP (1) - ef14cee73d1e:gateway_service:8888
SERVICE-OFFER	n/a (1)	(1)	UP (1) - 8ff01538ce7a:SERVICE-OFFER:8087
SERVICE-USER	n/a (1)	(1)	UP (1) - 543790553f13:SERVICE-USER:8088

SonarCloud



Jankins

```
pipeline {
    agent any

    stages {
        stage("Build & SonarQube Scanner") {
            steps {
                script {
                    withSonarQubeEnv('SonarCloud') {
                        // Microservice 1
                        dir('colocatair_app/backend/offre_service') {
                            bat 'mvn clean package sonar:sonar -Dsonar.organization=yahyamallouli'
                        }

                        // Microservice 2
                        dir('colocatair_app/backend/user_service') {
                            bat 'mvn clean package sonar:sonar -Dsonar.organization=yahyamallouli'
                        }

                        // Microservice 3
                        dir('colocatair_app/backend/demande_service') {
                            bat 'mvn clean package sonar:sonar -Dsonar.organization=yahyamallouli'
                        }
```

```
                        // Microservice 4 (Eureka Server)
                        dir('colocatair_app/backend/eurekaserver') {
                            bat 'mvn clean package sonar:sonar -Dsonar.organization=yahyamallouli'
                        }
                    }
                }
            }
        }

        stage('Build & Run Docker Containers') {
            steps {
                script {
                    // Build and run containers using Docker Compose
                    bat 'docker-compose -f colocatair_app/backend/docker-compose.yml build'
                    bat 'docker-compose -f colocatair_app/backend/docker-compose.yml up -d'
                }
            }
        }
    }
}
```

	Declarative: Checkout SCM	Build & SonarQube Scanner	Create Docker Image	Run Docker Container
Average stage times: (Average full run time: ~1min 50s)	1s	1min 23s	20s	1s
#37 Jan 18 10:24 — 1 commit	1s	1min 18s	21s	1s
#34 Jan 18 10:17 — No Changes	1s	2min 18s	15s	1s
#33 Jan 18 10:15 — 5 commits	2s	55s	15s	2s
#29 Jan 18 09:55 — 1 commit	1s	1min 2s	27s	2s