Ingénierie Informatique et Réseaux
ACE

Rapport sur le Système de Réservation d'Hôtels en ligne

Réalisé par : AIT SIMH Nizar

QUAZBARI Yahya

BOUZITI Jaouad

Encadré par : M. Mohamed LACHGAR

1. Introduction

Aperçu du projet:

Le projet de Système de Réservation d'Hôtels en ligne est une application moderne qui utilise une architecture microservices pour fournir des fonctionnalités flexibles et évolutives. L'objectif est de permettre aux utilisateurs de rechercher, réserver et gérer leurs séjours dans divers hôtels. Le projet est basé sur un backend constitué de trois microservices distincts, à savoir ServiceChambre, ServiceUser et ServiceReservation, tous développés avec les technologies Spring Boot, Spring Cloud, et intégrant un système de découverte Eureka Server pour la gestion des microservices.

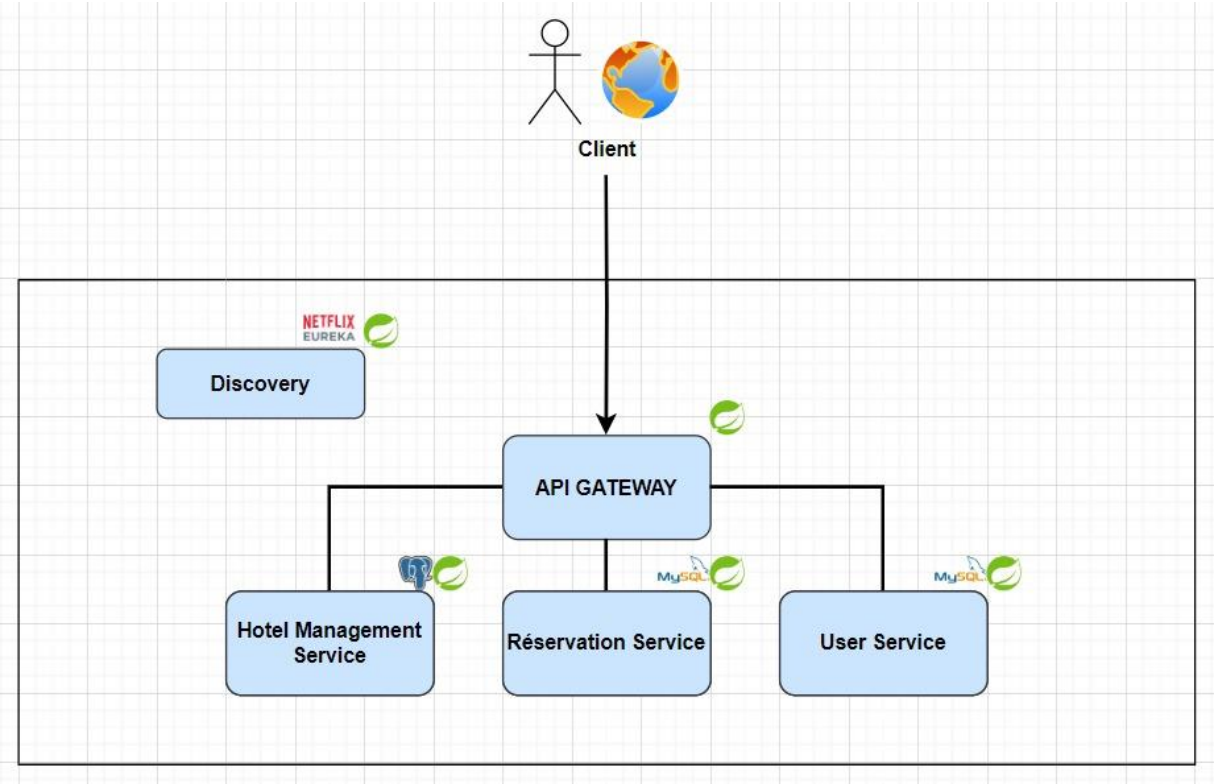
Importance de l'architecture microservices:

L'adoption de l'architecture microservices présente plusieurs avantages, tels que la facilité de déploiement, la scalabilité, et la résilience. Elle permet également une gestion indépendante de chaque service, favorisant le développement agile et la maintenance efficace du système.

2. Architecture Microservices

Architecture:

Le système est conçu autour d'une architecture microservices, où chaque microservice est responsable d'une fonctionnalité spécifique. Les services communiquent entre eux à l'aide de mécanismes appropriés, avec un routage dynamique assuré par un API Gateway.



Description des services:

ServiceChambre: Comprend les classes Ville, Hotel et Chambre, fournissant des informations sur les chambres disponibles, les hôtels et les villes.

ServiceUser: Gère la classe User pour l'authentification et la gestion des utilisateurs.

ServiceReservation: Gère la classe Réservation, permettant aux utilisateurs de réserver des chambres.

Mécanismes de communication:

OpenFeign est utilisé pour la communication entre les microservices, facilitant l'appel distant des services et simplifiant l'intégration entre eux.

3. Conception des Microservices

Approche de conception pour chaque service:

Chaque service est conçu de manière modulaire avec une API RESTful. Les classes spécifiques à chaque service sont développées pour répondre aux besoins fonctionnels correspondants. OpenFeign est utilisé pour simplifier l'appel entre les services.

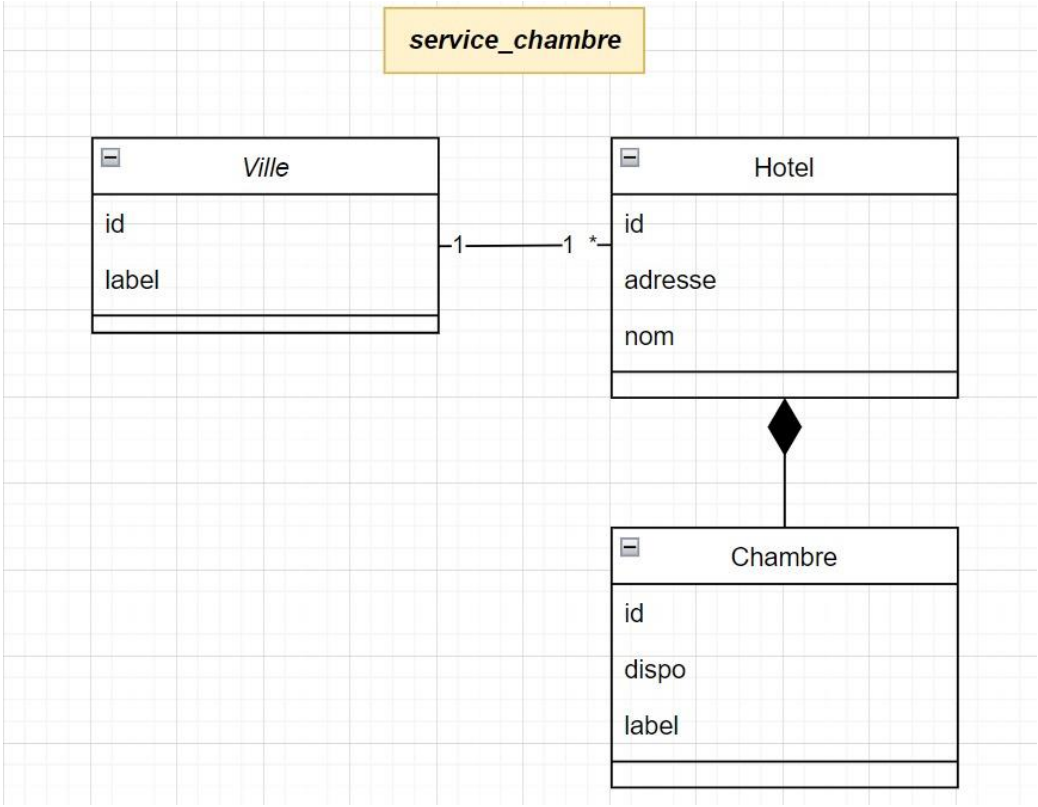


figure 1 : Diagramme de classes ServiceChambre

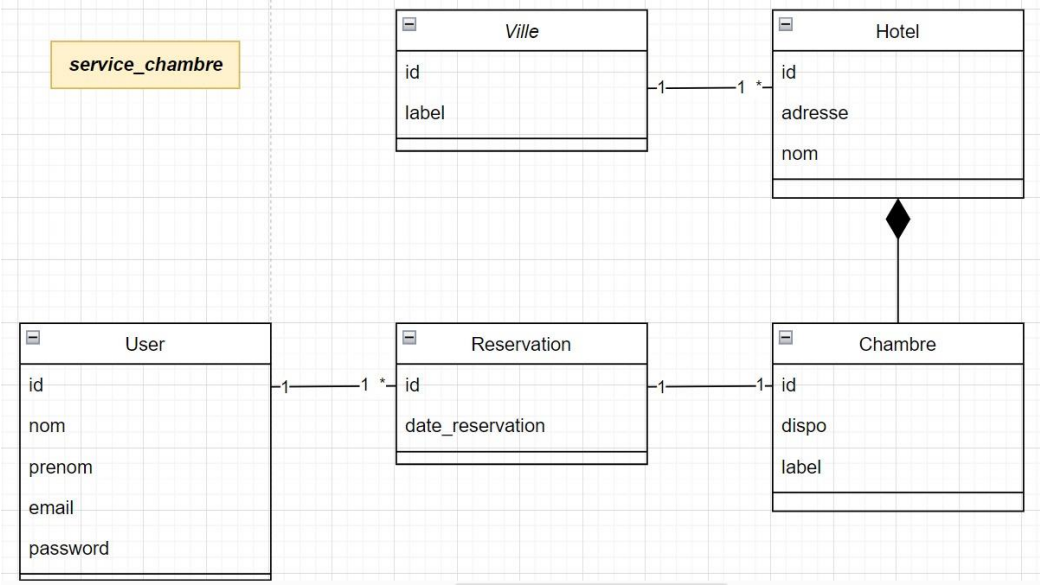


figure 2 : Diagramme de classes ServiceReservation

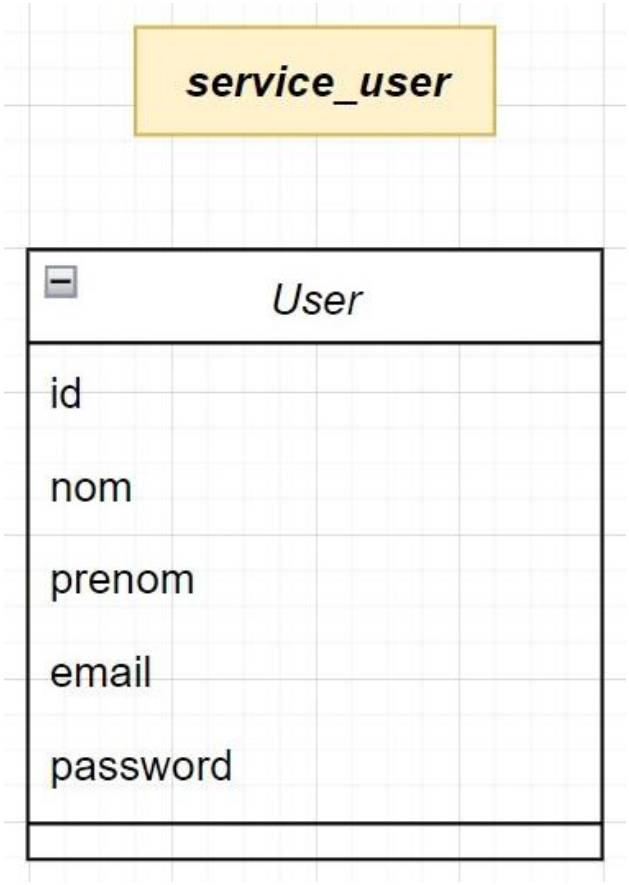


figure 3 : Diagramme de classes ServiceUser

4. Conteneurisation avec Docker

Implémentation et avantages:

Les microservices sont conteneurisés à l'aide de Docker pour assurer la portabilité et la facilité de déploiement. Cette approche offre une gestion efficace des dépendances et garantit une cohérence entre les environnements de développement, de test et de production.

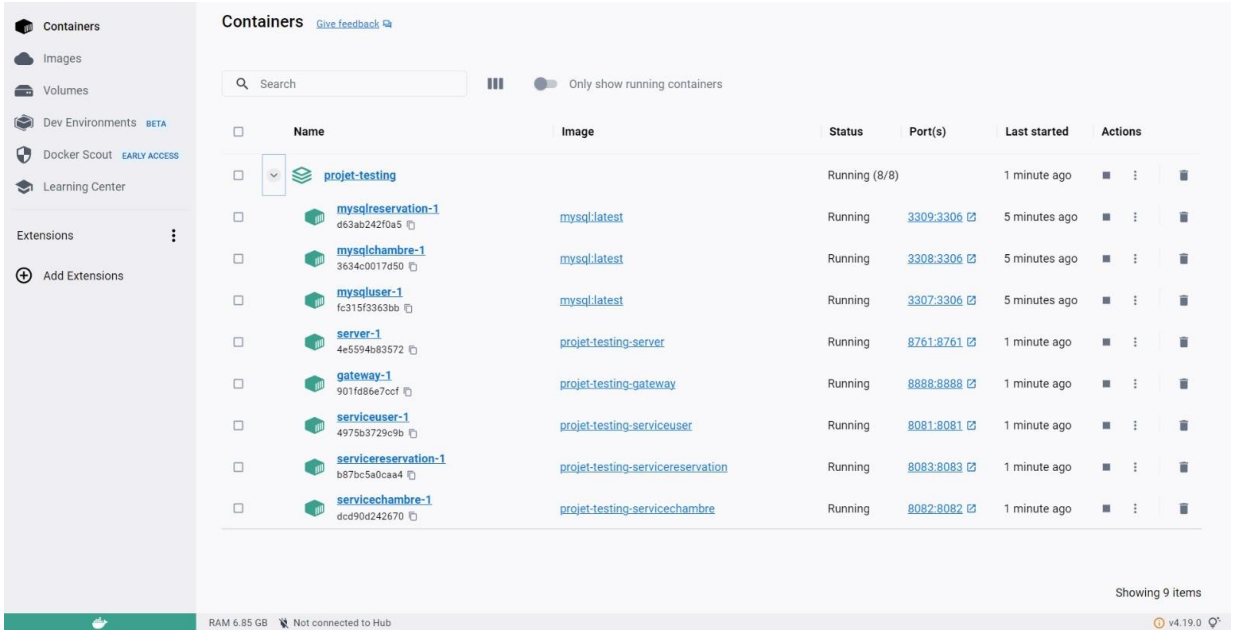


figure : Images du conteneur Docker

5. CI/CD avec Jenkins

Processus et configuration:

Le processus d'intégration continue (CI) et de déploiement continu (CD) est géré par Jenkins. Les tests automatisés sont exécutés à chaque intégration, garantissant la stabilité du système. Jenkins est également configuré pour automatiser le déploiement des microservices dans différents environnements.

```groovy
pipeline {
    agent any

    tools {
        maven 'maven'
    }

    stages {
        stage('Git Clone') {
            steps {
                script {
                    checkout([$class: 'GitSCM', branches: [[name: 'master']], userRemoteConfigs: [[url: 'https://github.com/BOUZITI-DEV/projetTesting']]])
                }
            }
        }

        stage('Build') {
            steps {
                script {
                    dir('server') {
                        bat 'mvn clean install'
                    }
                    dir('gateway') {
                        bat 'mvn clean install'
                    }
                    dir('serviceUser') {
                        bat 'mvn clean install'
                    }
                    dir('serviceChambre') {
                        bat 'mvn clean install'
                    }
                    dir('serviceReservation') {
                        bat 'mvn clean install'
                    }
                }
            }
        }

        stage('Docker Compose') {
            steps {
                script {
                    bat 'docker-compose build'
                }
            }
        }

        stage('Run') {
            steps {
                script {
                    bat 'docker-compose up -d'
                }
            }
        }
    }
}
```

figure 1 : Script d'exécution Pipline Jenkins

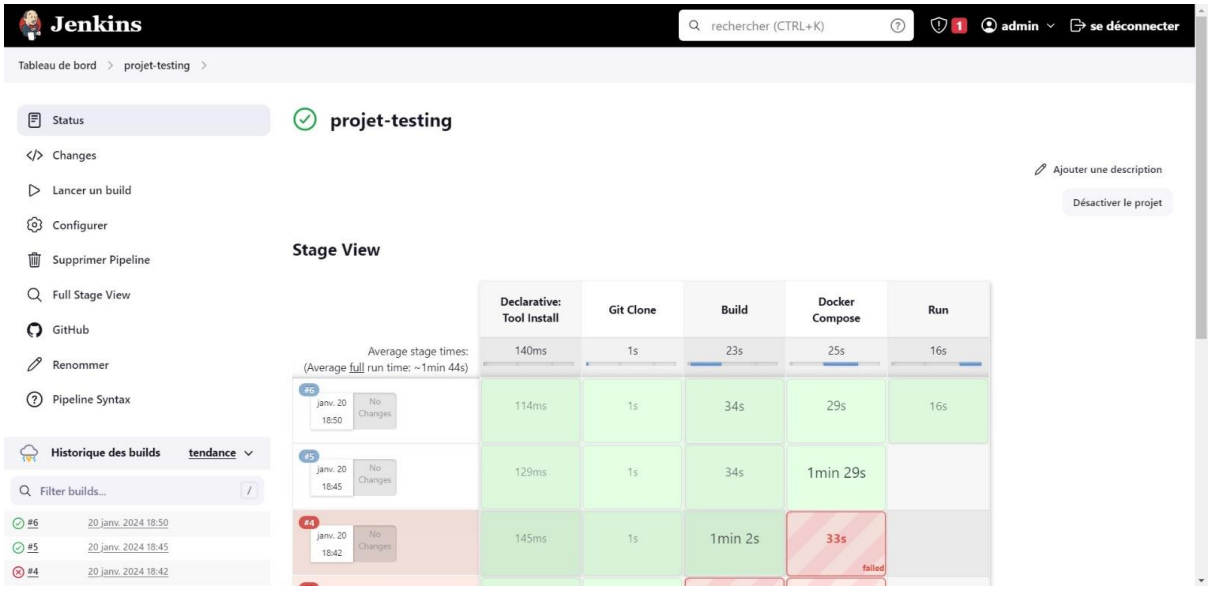


figure 2 : Statut de Build du Projet

6. Intégration de SonarCloud

Configuration et bénéfices pour la qualité du code:

SonarCloud est intégré pour l'analyse statique du code, permettant de détecter et de corriger les problèmes de qualité du code. Le capture d'écran de SonarCloud fournie une vue détaillée de la qualité du code, des bugs potentiels et des vulnérabilités.

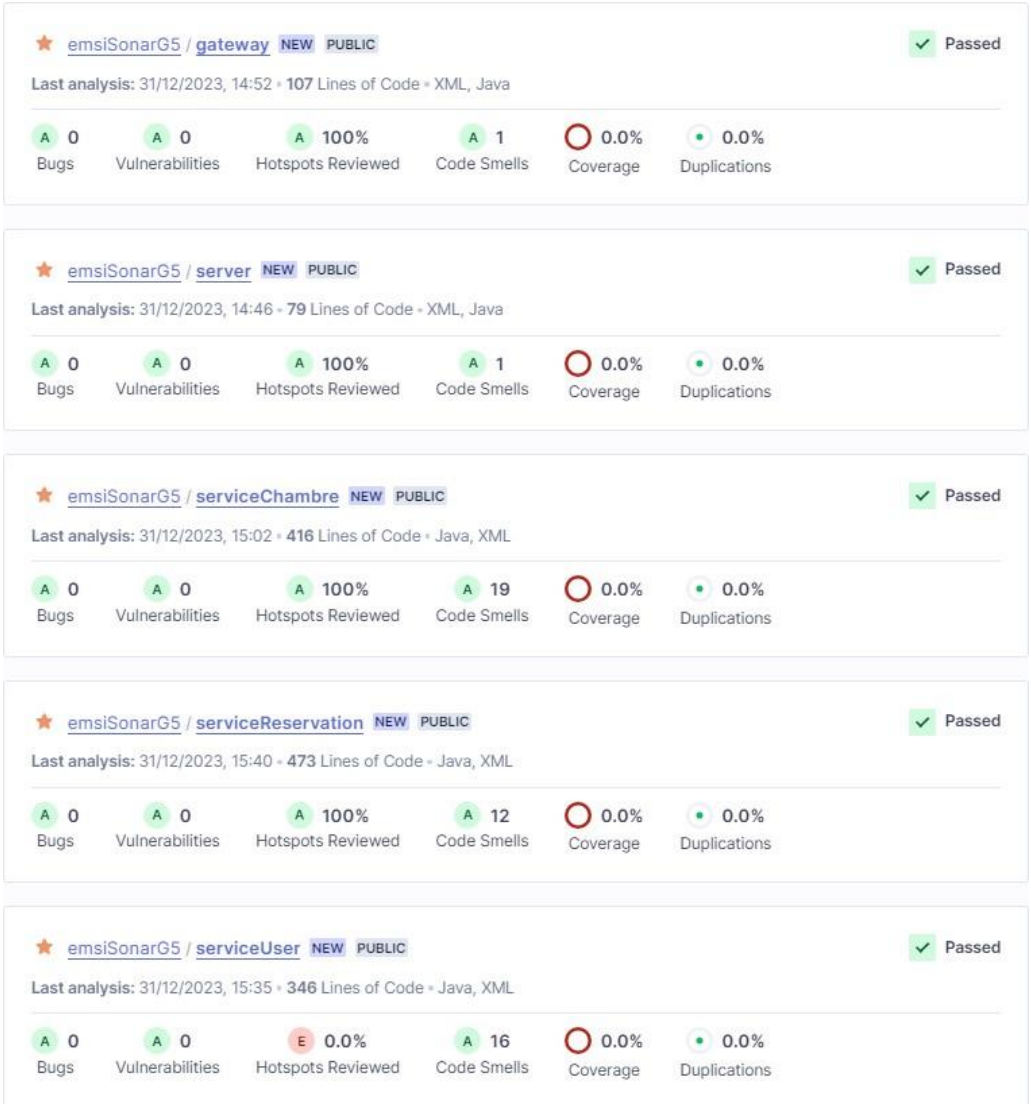


figure : Qualité du code en SonarCloud

7. Conclusion

Résumé des accomplissements:

Le projet a réussi à mettre en œuvre un système de réservation d'hôtels en ligne basé sur une architecture microservices, avec une conception modulaire, une conteneurisation efficace, un processus CI/CD automatisé et une intégration réussie de SonarQube pour garantir la qualité du code.

Perspectives futures:

Les perspectives futures incluent l'expansion des fonctionnalités, l'optimisation des performances, et l'exploration de nouvelles technologies pour répondre aux évolutions du marché et des besoins des utilisateurs. L'adoption éventuelle de Kubernetes pour l'orchestration des conteneurs pourrait également être envisagée pour une gestion plus avancée des microservices.