

# Mini-Projet

## 2023-2024

### Application Web d'un projet Portfolio en ligne

### Architecture des composants d'entreprise

---

Réalisé par :

MIKALI Ayoub

GHAZALI Loubna

# Tables des matières

1	Introduction :	0
1.1	Introduction	0
1.2	Aperçu du projet	0
1.2.1	Objectif	0
1.2.2	Outils	0
1.3	Importance de microservice :	3
2	Architecture Microservices :	4
2.1	Architecture :	4
2.2	Description des services.	4
2.3	Mécanismes de communication :	5
3	Conception des Microservices :	6
4	Conteneurisation avec Docker :	7
4.1	Implémentation :	7
4.2	Avantage :	8
5	CI/CD avec Jenkins	10
5.1	Processus de CI/CD avec Jenkins :	10
5.2	Configuration avec Jenkins :	12
6	Déploiement Automatique	17
7	Intégration de SonarQube	17
8	Conclusion.....	18
8.1	Résumé des accomplissements	18
8.2	Perspectives futures.....	18



# 1 Introduction :

## 1.1 Introduction

Ce chapitre fournira un aperçu détaillé des objectifs, de l'architecture et des technologies clés utilisées dans ce projet, mettant en lumière l'approche révolutionnaire que nous avons adoptée pour répondre aux besoins actuels du développement logiciel.

## 1.2 Aperçu du projet

### 1.2.1 Objectif

L'objectif est de Développé une plateforme web interactive mettant en avant mes réalisations professionnelles, mes compétences techniques et mes expériences passées de manière attrayante et accessible :

1. **Compétences** (Skills) : Présenter de manière détaillée mes compétences techniques, mes domaines d'expertise et mes réalisations professionnelles afin de démontrer ma valeur ajoutée dans le domaine concerné.
2. **Projets** (Projet) : Exposer mes projets antérieurs, actuels et futurs, en mettant en avant les défis relevés, les solutions apportées et les résultats obtenus pour illustrer ma capacité à mener à bien des initiatives variées.
3. **Expérience** (Experience) : Décrire de façon exhaustive mon parcours professionnel, mes responsabilités, mes réussites et les leçons apprises tout au long de mon cheminement professionnel pour mettre en lumière ma valeur en tant que professionnel aguerri.
4. **Éducation** (Education) : Présenter mon parcours académique, mes diplômes, mes certifications et mes réalisations académiques pour mettre en avant ma formation et mes compétences acquises au fil du temps.
5. **Contact** : Fournir aux visiteurs du site un moyen facile et efficace de me contacter pour des opportunités professionnelles, des collaborations ou des échanges d'informations, afin de favoriser les interactions et les opportunités de networking.

### 1.2.2 Outils

- **SpringBoot:**

Open source, SpringBoot est un framework de développement Java. Il permet ainsi la création et le développement d'API. Sa principale originalité réside dans le fait qu'il facilite considérablement ces capacités de développement grâce à une réduction importante de la complexité des configurations. SpringBoot est un projet qui s'appuie sur le Spring Framework, un des meilleurs frameworks 2023. Il est solide, sûr et efficace. Il offre un moyen plus facile et plus rapide de mettre en place, de configurer et d'exécuter des applications simples et basées sur le Web.



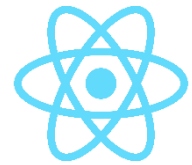
- **Java :**

Java est un langage de programmation largement utilisé pour coder des applications web. Il a été fréquemment choisi parmi les développeurs depuis plus de deux décennies, des millions d'applications Java étant utilisées aujourd'hui. Java est un langage multiplateforme, orienté objet et centré sur le réseau, qui peut être utilisé comme une plateforme à part entière. Il s'agit d'un langage de programmation rapide, sécurisé et fiable qui permet de tout coder, des applications mobiles aux logiciels d'entreprise en passant par les applications de big data et les technologies côté serveur.



- **React.js**

React, est une bibliothèque JavaScript libre et open-source. Il est particulièrement efficace pour construire des interfaces utilisateur en combinant des sections de code (composants) pour former des sites web complets. À l'origine développé par Facebook, Meta et la communauté open source en assurant maintenant la maintenance..



- **MySQL**

C'est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde2, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, PostgreSQL et Microsoft SQL Server.



- **Bootstrap**

C'est un framework open-source facilitant la conception et le développement d'interfaces web réactives. Il fournit des composants pré-conçus et des styles, permettant aux développeurs de créer rapidement des sites web attrayants et compatibles avec plusieurs navigateurs.



- **Postman**

C'est une plate-forme API pour la création et l'utilisation d'API. Postman simplifie chaque étape du cycle de vie des API et rationalise la collaboration afin que vous puissiez créer de meilleures API plus rapidement.



- **Git**

C'est outil de gestion de versionning et de collaboration, autrement dit un système de contrôle de version distribuée gratuit et open source, il gère des projets sans prendre en considération leurs taille avec rapidité et efficacité et permet de suivre les modifications apportées aux fichiers informatiques ainsi de coordonner le travail sur ces fichiers entre plusieurs personnes.



- **Xampp**

(Apache MariaDB Perl PHP) regroupe un ensemble de logiciels qui met en place un serveur Web local, un serveur FTP et un serveur de messagerie électronique. Ce distributeur de logiciels libres offre une bonne souplesse d'utilisation simple, rapide et fonctionne sur les systèmes d'exploitation les plus répandus. Certes, XAMPP configure un serveur de test local avant la mise en œuvre d'un site Web.



- **GitHub**

C'est Service d'hébergement de code source et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités. En 2018, GitHub est acquis par Microsoft pour 7,5 milliards de dollars.



- **Draw.io**

C'est une application web de création de diagrammes en ligne. Elle permet aux utilisateurs de concevoir et de dessiner des diagrammes de flux, des organigrammes, des schémas UML, et d'autres types de diagrammes de manière intuitive et collaborative. Draw.io est souvent utilisé pour la modélisation visuelle dans le cadre du développement logiciel, de la gestion de projet et d'autres domaines.



- **IntelliJ IDEA**

C'est un environnement de développement intégré (IDE) développé par JetBrains, largement utilisé pour créer des applications dans divers langages tels que Java, Kotlin, et d'autres. Il offre des fonctionnalités avancées de productivité et de débogage, ainsi qu'une interface conviviale. Disponible en versions Community (gratuite) et Ultimate (payante).



- **Visual Studio Code**

C'est un éditeur de code source léger, gratuit et open- source, développé par Microsoft. Il offre une interface conviviale et de nombreuses fonctionnalités pour le développement de logiciels, y compris la coloration syntaxique, l'auto-complétions, le débogage, le contrôle de version et la gestion des extensions. VS Code est apprécié pour sa polyvalence, ses performances élevées et sa large communauté de développeurs qui créent des extensions pour étendre ses fonctionnalités de base. C'est un outil populaire et largement utilisé dans le domaine du développement de logiciels.



### 1.3 Importance de microservice :

L'adoption des microservices vise à atteindre plusieurs objectifs clés dans le domaine du développement logiciel moderne :

- **Scalabilité** : Les microservices peuvent être déployés et échelle séparément, ce qui permet de gérer efficacement la croissance de la charge de travail.
- **Flexibilité** : Les microservices permettent de développer, déployer et mettre à jour indépendamment les différentes parties d'une application, ce qui facilite la mise en place de nouvelles fonctionnalités ou la correction de bugs.
- **Fiabilité** : En utilisant des microservices, les erreurs dans une partie de l'application ne compromettent pas l'ensemble du système, ce qui permet une meilleure tolérance aux pannes.
- **Agilité** : Les microservices permettent de travailler avec des équipes plus petites et plus spécialisées, ce qui accélère le développement et la livraison de nouvelles fonctionnalités.
- **Intégration** : Les microservices peuvent être construits et déployés avec des technologies différentes, ce qui facilite l'intégration de nouvelles technologies et l'optimisation des performances.
- **Dépendance réduite** : En utilisant des microservices, les différentes parties d'une application ne dépendent pas les unes des autres, ce qui permet de mettre à jour ou de remplacer une partie de l'application sans affecter les autres parties.
- **Sécurité** : Les microservices peuvent être conçus de manière à isoler les données sensibles et les fonctions critiques, ce qui renforce la sécurité de l'application.
- **Flexibilité opérationnelle** : Les microservices peuvent être déployés sur des infrastructures différentes, ce qui permet de choisir les outils et les technologies les plus adaptés à chaque partie de l'application.
- **Meilleure utilisation des ressources** : En utilisant des microservices, on peut choisir des technologies spécifiques pour chaque service, ce qui permet d'optimiser les performances et de réduire les coûts opérationnels.

## 2 Architecture Microservices :

### 2.1 Architecture :

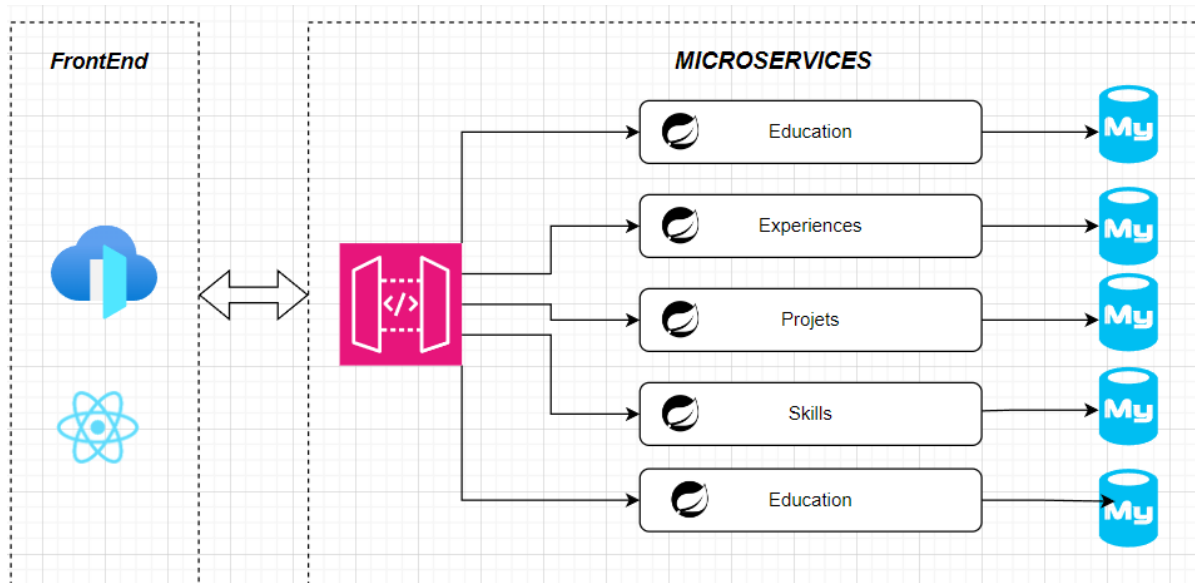


Figure 2.1: Architecture Microservices

### 2.2 Description des services

1. **Compétences (Skills)** : Dans la section dédiée à mes compétences, vous trouverez une présentation détaillée de mes aptitudes techniques, de mes langages de programmation maîtrisés, ainsi que des outils et des technologies que je manipule avec aisance. Cette section met en lumière mes compétences clés et mon expertise dans divers domaines.

2. **Projets (Project)** : Explorez mes réalisations passées, actuelles et à venir à travers la section dédiée à mes projets. Chaque projet est accompagné d'une description détaillée, mettant en avant les défis relevés, les solutions apportées et les résultats obtenus. Cela vous permettra de mieux comprendre mes capacités et mes réalisations concrètes.

3. **Expérience (Experience)** : Dans cette section, je partage mon parcours professionnel, mes expériences passées, mes responsabilités et les accomplissements significatifs que j'ai réalisés au fil des ans. Vous découvrirez comment mon expérience m'a permis de développer des compétences spécifiques et de relever des défis professionnels variés.

4. **Éducation (Education)** : Ma formation académique et mes réalisations éducatives sont mises en avant dans cette section. Vous y trouverez des détails sur mes diplômes, mes certifications et mes réalisations académiques, ainsi que sur la manière dont mon parcours éducatif a contribué à forger mes compétences et mes connaissances.



**5. Contact :** Si vous souhaitez me contacter pour des opportunités professionnelles, des collaborations ou simplement pour échanger des idées, n'hésitez pas à utiliser les informations de contact fournies dans cette section. Je suis toujours ouvert aux nouvelles opportunités et aux échanges constructifs.

## 2.3 Mécanismes de communication :

Pour permettre la communication entre ces microservices, divers mécanismes peuvent être utilisés. Eureka Server et Eureka Gateway sont des composants associés à Netflix Eureka, qui est un service de découverte de microservices.

Voici comment fonctionnent ces mécanismes de communication des microservices avec Eureka Server et Eureka Gateway :

### **Eureka Server :**

- 🚦 Service de découverte : Eureka Server est un service de découverte qui permet aux microservices de s'enregistrer et de découvrir les autres services disponibles. Chaque microservice s'enregistre auprès du serveur Eureka avec son nom et son adresse réseau.
- 🚦 Répertoire de services : Le serveur Eureka maintient un répertoire dynamique des microservices enregistrés, facilitant la recherche et l'accès aux services disponibles.

### **Microservice enregistrement/découverte :**

- 🚦 Enregistrement : Lorsqu'un microservice démarre, il s'enregistre auprès du serveur Eureka en indiquant son nom et son adresse réseau.
- 🚦 Renouvellement périodique : Les microservices renouvellent périodiquement leur enregistrement auprès du serveur Eureka pour signaler qu'ils sont toujours actifs.

### **Eureka Gateway :**

- 🚦 Passerelle d'API : Eureka Gateway sert de passerelle pour la communication entre les microservices. Il est responsable de rediriger les requêtes clients vers les instances appropriées des microservices en utilisant les informations fournies par le serveur Eureka.
- 🚦 Load balancing : Eureka Gateway peut également intégrer des mécanismes de répartition de charge pour distribuer les requêtes entre les instances disponibles d'un microservice.

### **Communication entre microservices :**

- 🚦 RESTful API : La communication entre microservices se fait généralement via des API RESTful. Chaque microservice expose ses fonctionnalités via des points de terminaison API REST.

- ✚ Utilisation du nom du service : Plutôt que d'utiliser des adresses IP ou des URL fixes, les microservices communiquent en utilisant le nom du service, ce qui est résolu dynamiquement par le serveur Eureka.

#### Avantages :

- ✚ Évolutivité : La découverte de services permet une évolutivité facile en ajoutant ou en retirant des instances de microservices.
- ✚ Résilience : En cas de panne d'un microservice, le serveur Eureka peut rediriger les requêtes vers d'autres instances disponibles.
- ✚ Développement indépendant : Les équipes de développement peuvent travailler de manière indépendante sur leurs microservices, car la découverte et la communication sont gérées par des composants spécialisés.

### 3 Conception des Microservices :

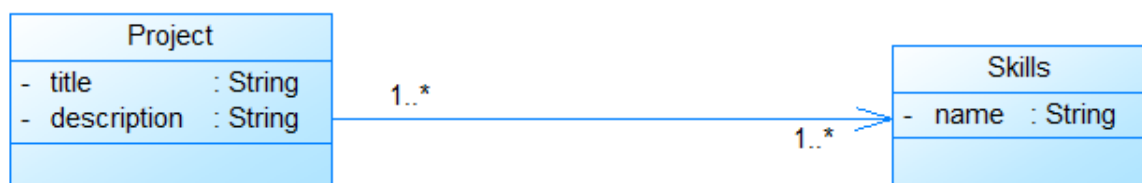


Figure 3.1: Conception du microservice projet

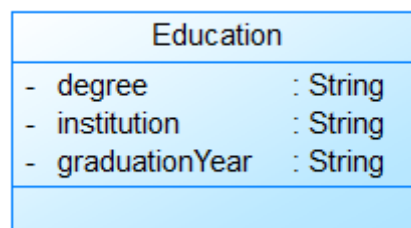


Figure 3.2: Conception du microservice éducation

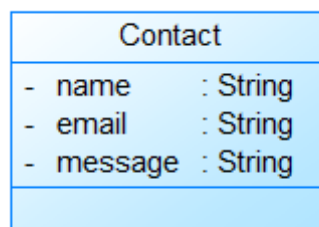


Figure 3.3: Conception du microservice contact

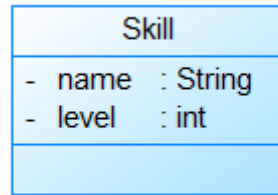


Figure 3.4: Conception du microservice compétence

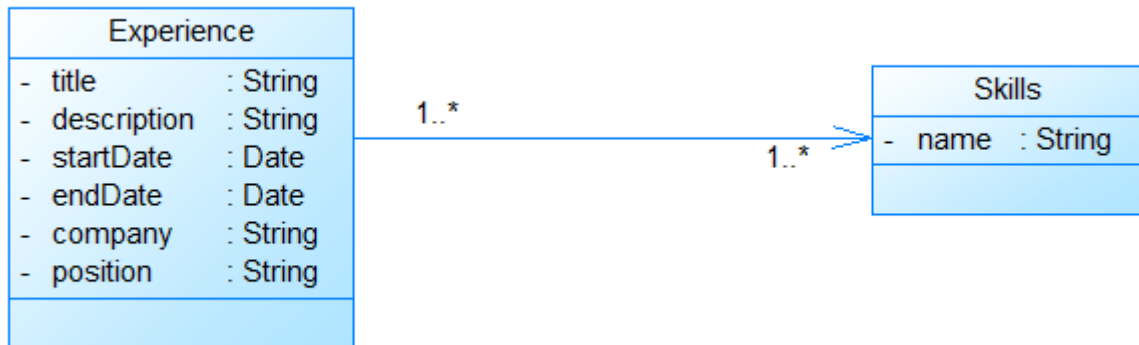


Figure 3.5: Conception du microservice expérience

## 4 Conteneurisation avec Docker :

La conteneurisation avec Docker est une approche populaire pour déployer, gérer et orchestrer des applications. Docker fournit une plateforme open source qui permet d'empaqueter des applications et leurs dépendances dans des conteneurs légers et portables. Voici une explication de l'implémentation de la conteneurisation avec Docker et ses avantages.

### 4.1 Implémentation :

#### A- Création d'un Conteneur :

- Utilisation d'un fichier Dockerfile : Un fichier Dockerfile décrit les étapes nécessaires pour créer une image Docker.
- Construction de l'image : La commande docker build utilise le Dockerfile pour construire une image contenant l'application et ses dépendances.

#### B- Enregistrement de l'Image :

- Utilisation du registre Docker : Les images Docker peuvent être enregistrées dans un registre, comme Docker Hub, pour faciliter le partage et la distribution.
- Commandes docker push et docker pull : Les images peuvent être poussées vers le registre avec docker push et tirées sur d'autres machines avec docker pull.

### **C- Exécution d'un Conteneur :**

- Commande docker run : Pour lancer une instance d'un conteneur à partir d'une image, la commande docker run est utilisée.
- Isolation des ressources : Les conteneurs offrent une isolation des ressources, ce qui signifie que chaque conteneur a son propre espace de travail, son système de fichiers et ses processus.

### **D- Réseau et Communication :**

- Réseau Docker : Les conteneurs peuvent communiquer entre eux via le réseau Docker, et des ponts réseau peuvent être configurés pour faciliter la communication entre les conteneurs et l'hôte.
- Exposition de ports : Les ports peuvent être exposés sur les conteneurs pour permettre aux applications de recevoir des requêtes depuis l'extérieur.

### **E- Gestion des Conteneurs :**

- Surveillance et gestion : Docker fournit des commandes pour surveiller et gérer les conteneurs en cours d'exécution, y compris *docker ps*, *docker logs*, etc.
- Arrêt et suppression : Les conteneurs peuvent être arrêtés avec *docker stop* et supprimés avec *docker rm*.

## **4.2 Avantage :**

### **A- Portabilité :**

Les conteneurs encapsulent l'application et ses dépendances, ce qui garantit la portabilité entre différents environnements (développement, test, production).

### **B- Isolation :**

Chaque conteneur fonctionne de manière isolée, évitant ainsi les conflits de dépendances entre les applications et simplifiant la gestion des versions.

### **C- Rapidité et Légèreté :**

Les conteneurs partagent le noyau du système d'exploitation hôte, ce qui les rend plus légers et permet un démarrage rapide comparé aux machines virtuelles.

### **D- Évolutivité :**

La conteneurisation facilite le déploiement et la mise à l'échelle des applications, que ce soit sur une seule machine ou sur un orchestrateur de conteneurs comme Kubernetes.

### **E- Gestion des Versions :**

Les images Docker permettent de gérer les versions de manière explicite, assurant la reproductibilité des déploiements.

## F- Orchestration Facilitée :

Docker s'intègre bien avec des outils d'orchestration comme Kubernetes, permettant la gestion automatisée des conteneurs à grande échelle.

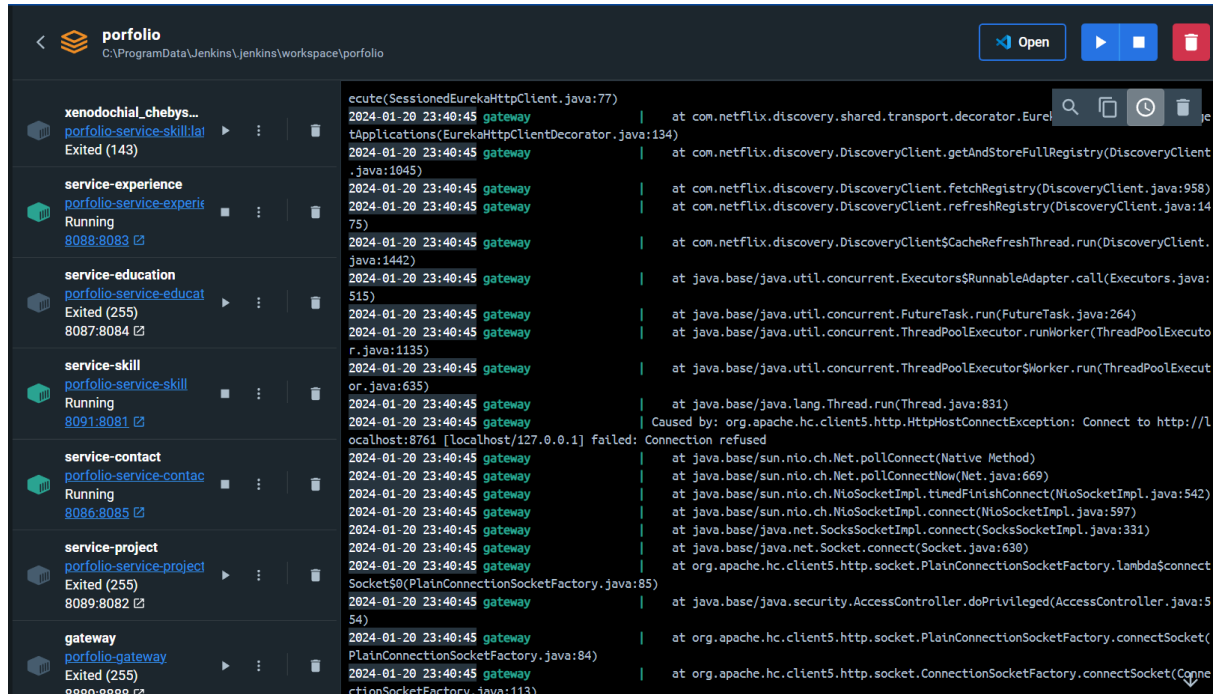











































Figure 4.1 Container portfolio

<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	<a href="#">portfolio-service-skill</a> 47542c09b8ba	latest	In use	about 1 hour a	400.75 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>
<input type="checkbox"/>	<a href="#">ghazali/skills</a> 626218f0cec3	latest	Unused	about 1 hour a	400.75 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>
<input type="checkbox"/>	<a href="#">portfolio-service-project</a> 6cc93d8239fc	latest	In use	about 1 hour a	400.75 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>
<input type="checkbox"/>	<a href="#">ghazali/project</a> c0a21647cb9a	latest	Unused	about 1 hour e	400.75 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>
<input type="checkbox"/>	<a href="#">ghazali/experience</a> 86fbf542ed99	latest	Unused	about 1 hour a	400.75 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>
<input type="checkbox"/>	<a href="#">portfolio-service-experience</a> 42ab152f4b64	latest	In use	about 1 hour a	400.75 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>
<input type="checkbox"/>	<a href="#">portfolio-service-education</a> df3ece718b9d	latest	In use	about 1 hour a	401.68 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>
<input type="checkbox"/>	<a href="#">ghazali/education</a> ...	latest	Unused	about 1 hour a	401.68 MB	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">🗑️</a>

Figure 4.2 Images docker

<input type="checkbox"/>	Name 	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	<a href="#">portfolio-service-education</a> df3ece718b9d 	latest	<a href="#">In use</a>	about 1 hour a	401.68 MB	  
<input type="checkbox"/>	<a href="#">ghazali/education</a> 81bb1a787c12 	latest	Unused	about 1 hour a	401.68 MB	  
<input type="checkbox"/>	<a href="#">portfolio-service-contact</a> 770b02c9b0d8 	latest	<a href="#">In use</a>	about 1 hour a	400.74 MB	  
<input type="checkbox"/>	<a href="#">ghazali/contact</a> af94a5fd546f 	latest	Unused	about 1 hour a	400.74 MB	  
<input type="checkbox"/>	<a href="#">ghazali/gateway</a> bdc3be7dba2f 	latest	Unused	about 1 hour a	373.1 MB	  
<input type="checkbox"/>	<a href="#">portfolio-gateway</a> 8f62d8ae2b6e 	latest	<a href="#">In use</a>	about 1 hour a	373.1 MB	  
<input type="checkbox"/>	<a href="#">portfolio-eureka-server</a> 7b598b2cd7ab 	latest	<a href="#">In use</a>	about 1 hour a	380.12 MB	  
<input type="checkbox"/>	<a href="#">ghazali/server_eureka</a> e830f13123ae 	latest	Unused	about 1 hour a	380.12 MB	  
<input type="checkbox"/>	<a href="#">ghazali/front</a> 469abb47d182 	latest	Unused	about 3 hours	46.67 MB	  
<input type="checkbox"/>	<a href="#">portfolio-reactapp</a> e9e425feb586 	latest	<a href="#">In use</a>	about 3 hours	46.67 MB	  

## 5 CI/CD avec Jenkins


Le processus de CI/CD avec Jenkins implique l'automatisation des étapes de construction, de test et de déploiement de logiciels. Voici une description de ce processus et de sa configuration :

### 5.1 Processus de CI/CD avec Jenkins :


Enter an item name

portfolio


= Required field


**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.


**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.


**Multi-configuration project**


Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.


**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.


**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.


**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from

OK

to autocomplete

## Configure



General



Advanced Project Options



Pipeline

- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☒ GitHub project
- Project url [?](#)
- 
- Advanced [v](#)
- ☐ Pipeline speed/durability override [?](#)
- ☐ Preserve stashes from completed builds [?](#)
- ☐ This project is parameterised [?](#)
- ☐ Throttle builds [?](#)

## Build Triggers

- ☐ Build after other projects are built [?](#)
- ☐ Build periodically [?](#)
- ☒ GitHub hook trigger for GITSCM polling [?](#)
- ☐ Poll SCM [?](#)
- ☐ Quiet period [?](#)
- ☐ Trigger builds remotely (e.g., from scripts) [?](#)

```
1 pipeline {
2   agent any
3   tools {
4     maven 'maven'
5   }
6
7   stages {
8     stage('Git clone') {
9       steps {
10        script {
11          // Clone the repository
12          checkout([$class: 'GitSCM', branches: [[name: 'main']], userRemoteConfigs: [[url: 'https://github.com/mikaliayoub/portfolio-microservice.git']]])
13        }
14      }
15    }
16
17    stage('Build Backend') {
18      steps {
19        script {
20          dir('server_eureka') {
21            bat 'mvn clean install'
22          }
23          dir('gateway') {
24            bat 'mvn clean install'
25          }
26          dir('contact') {
27            bat 'mvn clean install'
28          }
29          dir('educ') {
30            bat 'mvn clean install'
31          }
32          dir('experience') {
33            bat 'mvn clean install'
34          }
35          dir('project') {
36            bat 'mvn clean install'
37          }
38          dir('skills') {
39            bat 'mvn clean install'
40          }
41        }
42      }
43    }
44
45    stage('Create Docker Image (Backend)') {
46      steps {
47        script {
48          // Build Docker image for each microservice
49          dir('server_eureka') {
50            bat 'docker build -t ghazali/server_eureka .'
51          }
52          dir('gateway') {
53            bat 'docker build -t ghazali/gateway .'
54          }
55          dir('contact') {
56            bat 'docker build -t ghazali/contact .'
57          }
58          dir('educ') {
59            bat 'docker build -t ghazali/education .'
60          }
61          dir('experience') {
62            bat 'docker build -t ghazali/experience .'
63          }
64          dir('project') {
65            bat 'docker build -t ghazali/project .'
66          }
67          dir('skills') {
68            bat 'docker build -t ghazali/skills .'
69          }
70        }
71      }
72    }
73
74    stage('Build and Create Docker Image (Frontend)') {
75      steps {
76        script {
77          dir('front-main') {
78            bat 'npm install'
79          }
80
81          // Build Docker image for frontend
82          bat 'docker build -t ghazali/front ./front-main'
83        }
84      }
85    }
86
87    stage('Run (Backend and Frontend)') {
88      steps {
89        script {
90          // Stop and remove the previous containers if exist
91          bat 'docker-compose down'
92
93          // Run the new Docker containers for both backend and frontend
94          bat 'docker-compose up -d'
95        }
96      }
97    }
98
99    post {
100     success {
101       echo 'Pipeline succeeded! Your services are deployed.'
102     }
103     failure {
104       echo 'Pipeline failed! Check the logs for errors.'
105     }
106   }
107 }
```

## 5.2 Configuration avec Jenkins :

Pour configurer un processus de CI/CD avec Jenkins, on utilise des pipelines. Les pipelines Jenkins permettent de définir les étapes du processus, telles que la récupération du code source depuis un référentiel, la compilation, les tests unitaires, les tests d'intégration, la construction d'artefacts, et enfin le déploiement. Ces étapes sont configurées dans un fichier de pipeline (Jenkinsfile) qui définit le flux de travail complet du CI/CD.

De plus, Jenkins offre une intégration avec d'autres outils de développement tels que Git, Maven, Docker, et des services de cloud computing comme AWS ou Azure, ce qui permet de créer un processus de CI/CD complet et personnalisé en fonction des besoins spécifiques du projet.

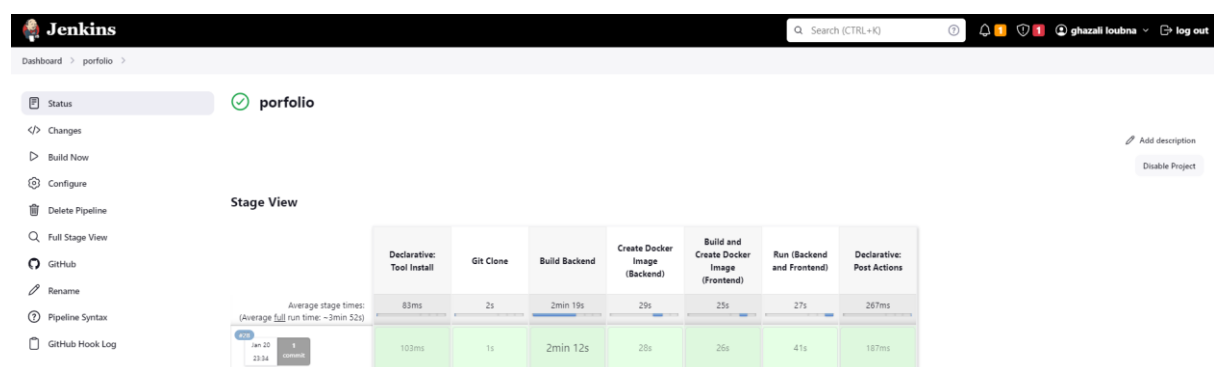


Figure 5.1 Jenkins

Voici le **script pipeline** que j'ai utilisé :

```
pipeline {
    agent any

    tools {
        maven 'maven'
    }

    stages {
        stage('Git Clone') {
            steps {
                script {
                    // Clone the repository
                }
            }
        }
    }
}
```



```
        checkout([$class: 'GitSCM', branches: [[name: 'main']], userRemoteConfigs:
[[url: 'https://github.com/mikaliayoub/portfolio-microservice.git']]])
```

```
    }
```

```
  }
```

```
}
```

```
stage('Build Backend') {
```

```
  steps {
```

```
    script {
```

```
      dir('server_eureka') {
```

```
        bat 'mvn clean install'
```

```
      }
```

```
      dir('gateway') {
```

```
        bat 'mvn clean install'
```

```
      }
```

```
      dir('contact') {
```

```
        bat 'mvn clean install'
```

```
      }
```

```
      dir('educ') {
```

```
        bat 'mvn clean install'
```

```
      }
```

```
      dir('experience') {
```

```
        bat 'mvn clean install'
```

```
      }
```

```
      dir('project') {
```

```
        bat 'mvn clean install'
```

```
      }
```

```
      dir('skills') {
```

```
        bat 'mvn clean install'
```

```
      }
```

```
  }
```

```
}  
}
```

```
stage('Create Docker Image (Backend)') {  
  steps {  
    script {  
      // Build Docker image for each microservice  
      dir('server_eureka') {  
        bat 'docker build -t ghazali/server_eureka .'      }  
      dir('gateway') {  
        bat 'docker build -t ghazali/gateway .'      }  
      dir('contact') {  
        bat 'docker build -t ghazali/contact .'      }  
      dir('educ') {  
        bat 'docker build -t ghazali/education .'      }  
      dir('experience') {  
        bat 'docker build -t ghazali/experience .'      }  
      dir('project') {  
        bat 'docker build -t ghazali/project .'      }  
      dir('skills') {  
        bat 'docker build -t ghazali/skills .'      }  
    }  
  }  
}
```

```
}
```

```
stage('Build and Create Docker Image (Frontend)') {
```

```
  steps {
```

```
    script {
```

```
      dir('front-main') {
```

```
        bat 'npm install'
```

```
      }
```

```
      // Build Docker image for frontend
```

```
      bat 'docker build -t ghazali/front ./front-main'
```

```
    }
```

```
  }
```

```
}
```

```
stage('Run (Backend and Frontend)') {
```

```
  steps {
```

```
    script {
```

```
      // Stop and remove the previous containers if exist
```

```
      bat 'docker-compose down'
```

```
      // Run the new Docker containers for both backend and frontend
```

```
      bat 'docker-compose up -d'
```

```
    }
```

```
  }
```

```
}
```

```
}
```

```
post {
```

```
  success {
```

```
    echo 'Pipeline succeeded! Your services are deployed.'
  }
  failure {
    echo 'Pipeline failed! Check the logs for errors.'
  }
}
}
```

## 6 Déploiement Automatique

```
ngrok

Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status      online
Account             BlackTulip (Plan: Free)
Version             3.5.0
Region              Europe (eu)
Latency             89ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://1cd5-196-118-93-148.ngrok-free.app -> http://localhost:3001

Connections          ttl      opn      rt1      rt5      p50      p90
                   3        0        0.03     0.01     5.02     19.53
```

## 7 Intégration de SonarQube

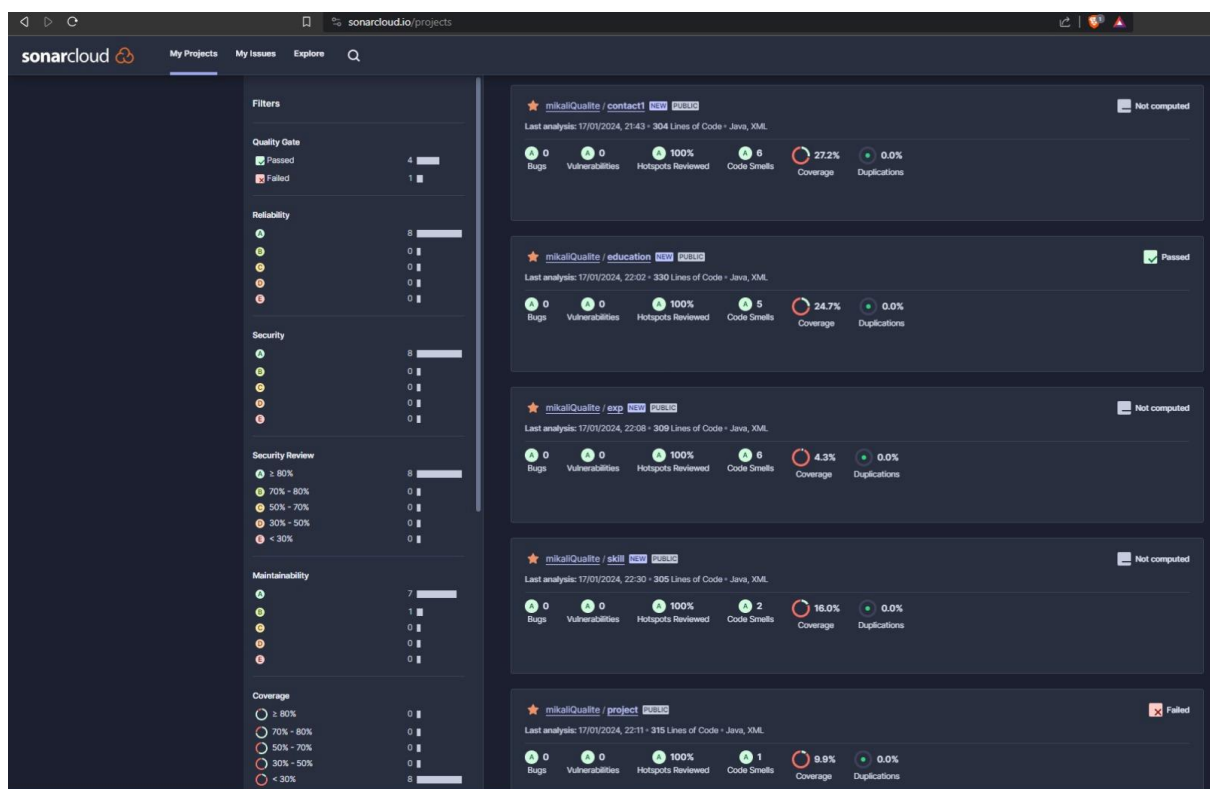


Figure 7.1 Sonar

## 8 Conclusion

### 8.1 Résumé des accomplissements

- **Architecture Micro-services:**

Mise en place d'une architecture micro-services pour le portfolio, offrant une meilleure scalabilité et une gestion modulaire des fonctionnalités.

- **Développement du Portfolio:**

Conception et développement des micro-services pour différentes fonctionnalités du portfolio telles que la présentation des projets, la gestion des compétences, et l'affichage de l'expérience professionnelle.

- **Intégration de Jenkins:**

Mise en œuvre de l'intégration continue (CI) avec Jenkins pour automatiser le processus de build à chaque modification de code. Configuration de pipelines Jenkins pour effectuer des tests automatiques, garantissant la qualité du code à chaque itération.

- **Utilisation de Docker:**

Intégration de Docker pour la conteneurisation des micro-services, facilitant le déploiement et l'orchestration dans différents environnements.

- **Déploiement Continu:**

Implémentation de la livraison continue (CD) avec Jenkins pour automatiser le déploiement des micro-services dans les environnements de test et de production.

- **Gestion des Versions:**

Utilisation de Git pour la gestion de version, permettant le suivi des changements et la collaboration efficace au sein de l'équipe de développement.

### 8.2 Perspectives futures

- **Personnalisation avancée :** Offrir une personnalisation plus poussée du contenu pour mettre en valeur mes compétences et projets de manière plus dynamique.
- **Intégration de nouvelles technologies :** Explorer l'intégration de nouvelles technologies pour améliorer l'expérience utilisateur, telles que des présentations interactives ou des outils d'analyse avancée.
- **Renforcement de la sécurité :** Mettre en place des mesures de sécurité avancées pour protéger les données et garantir la confidentialité des utilisateurs.