

# Rapport projet : Architecture des composant d'entreprise

**Code QR (Generator / Scanner)**

Réalisés par : **Ahmed SAYOUT Nabil SAS**

**Omaima AIT ZAOUIT**

# Table des matières :

[Table des matières 2](#_TOC_250009)

Introduction 3

[Aperçu du projet 3](#_TOC_250008)

Fonctionnalités du Frontend 3

[Importance de l’architecture microservice 4](#_TOC_250007)

Architecture Microservice 4

[Architecture 4](#_TOC_250006)

[Description des services 4](#_TOC_250005)

[Mécanisme de communication 5](#_TOC_250004)

Conception des Microservices 6

Conteneurisation avec Docker 7

[Avantage 7](#_TOC_250003)

[Implementation 7](#_TOC_250002)

[CI/CD avec Jenkins 8](#_TOC_250001)

Intégration de SonarQube 9

[Conclusion 10](#_TOC_250000)

## Inľroducľion

### Aperçu du projet :

ce projet vise à développer un système de QR Code Scanner et Générateur en utilisant les technologies modernes, avec le backend construit sur Spring Boot et le frontend

développé en React JS. L'architecture microservices est intégrée pour favoriser la scalabilité, la modularité et la ﬂexibilité du système.

Le backend du système est construit sur la puissante plateforme Spring Boot, offrant une architecture robuste et évolutive. Les microservices sont utilisés pour décomposer le système en composants indépendants, chacun se concentrant sur une tâche spéciﬁque.

Service de Génération de QR Code :

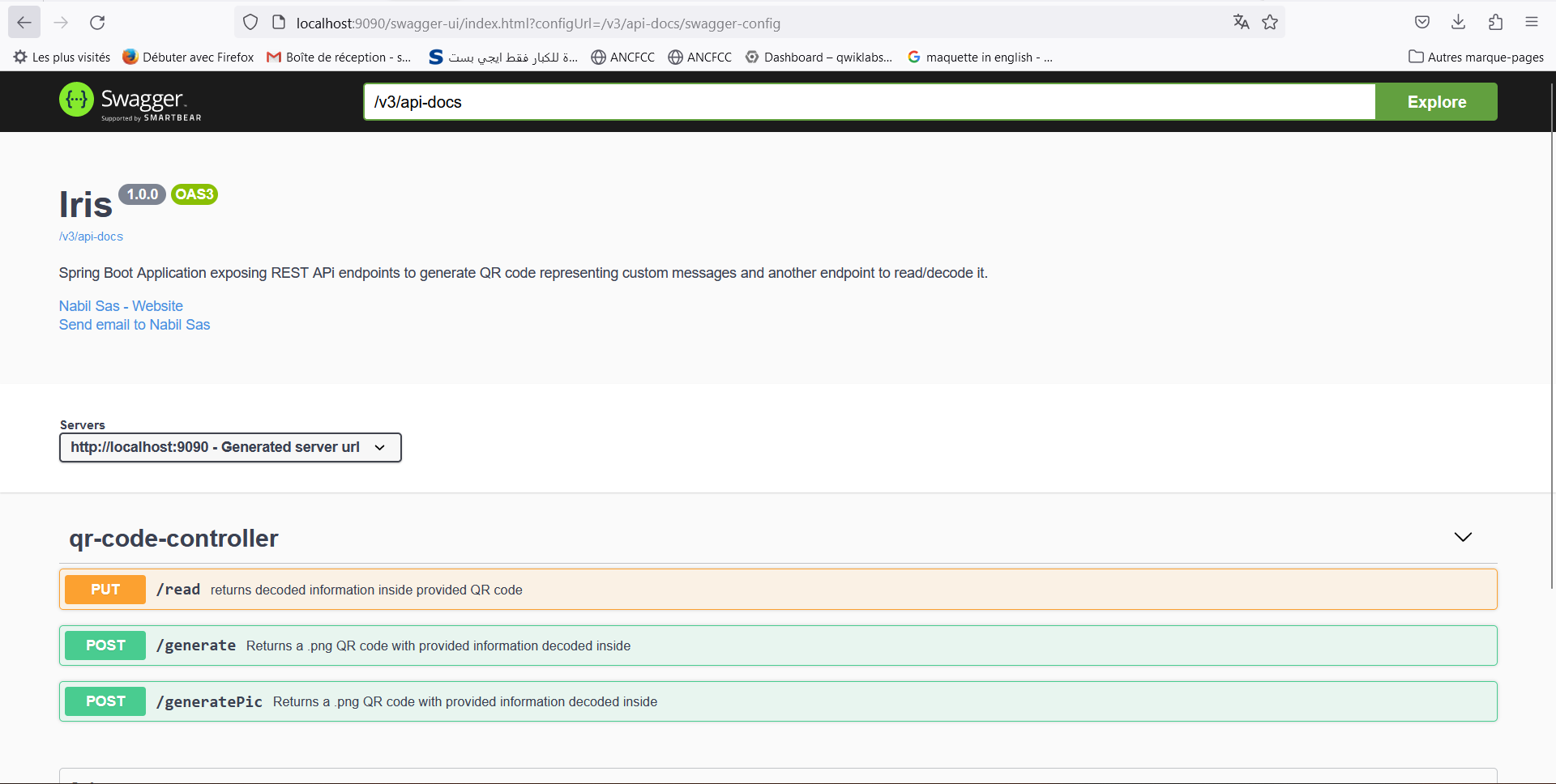
* Un microservice dédié à la création de codes QR à partir des données fournies.
* Utilisation de Spring Boot pour gérer la logique métier de manière eﬃcace.

Service de Scanner de QR Code :

* Un microservice spécialisé dans l'interprétation des codes QR capturés par un scanner.

APIs RESTful :

* Mise en place d'APIs RESTful pour permettre une communication ﬂuide entre les microservices et le frontend.
* Utilisation de Swagger pour documenter les endpoints et simpliﬁer l'intégration.





Le frontend est développé en utilisant React JS, une bibliothèque JavaScript moderne permettant de créer des interfaces utilisateur réactives et interactives.

Fonctionnalités du Frontend :

Interface Utilisateur Conviviale :

* Conception d'une interface utilisateur intuitive, facilitant la génération et la lecture de codes QR.

Communication avec les Microservices :

* Intégration transparente avec les microservices du backend via des appels API asynchrones.

Gestion des Résultats :

* Aﬃchage en temps réel des résultats des opérations de génération et de scanning de QR Code.

### Importance de l’architecture microservice :

L'utilisation de l'architecture microservices dans une application qui scanne et génère des codes QR peut offrir plusieurs avantages en termes de scalabilité, de maintenance, de ﬂexibilité et de déploiement. Parmis ces termes on trouve :

**Évolutivité :** L'architecture microservices favorise une approche agile en permettant l'ajout de nouveaux services ou la mise à jour indépendante pour répondre aux besoins changeants de l'application.

**Isolation des Erreurs :** Les erreurs dans un microservice n'impactent pas nécessairement l'ensemble du système, facilitant la détection et la résolution des problèmes.

**Modularité et Maintenance Facilitée :** Chaque microservice se concentre sur une tâche spéciﬁque, simpliﬁant la compréhension, la maintenance et la mise à jour du code.

**Déploiement indépendant :** Chaque microservice peut être déployé séparément, permettant des mises à jour rapides sans perturber l'ensemble de l'application.

## Architeture Microservice

### Architecture :

Le microservice unique de l'application est conçu pour effectuer à la fois la génération et l'interprétation des codes QR. Bien que la structure actuelle ne comporte qu'un seul service, l'approche microservices offre la ﬂexibilité nécessaire pour introduire de nouveaux services au besoin.

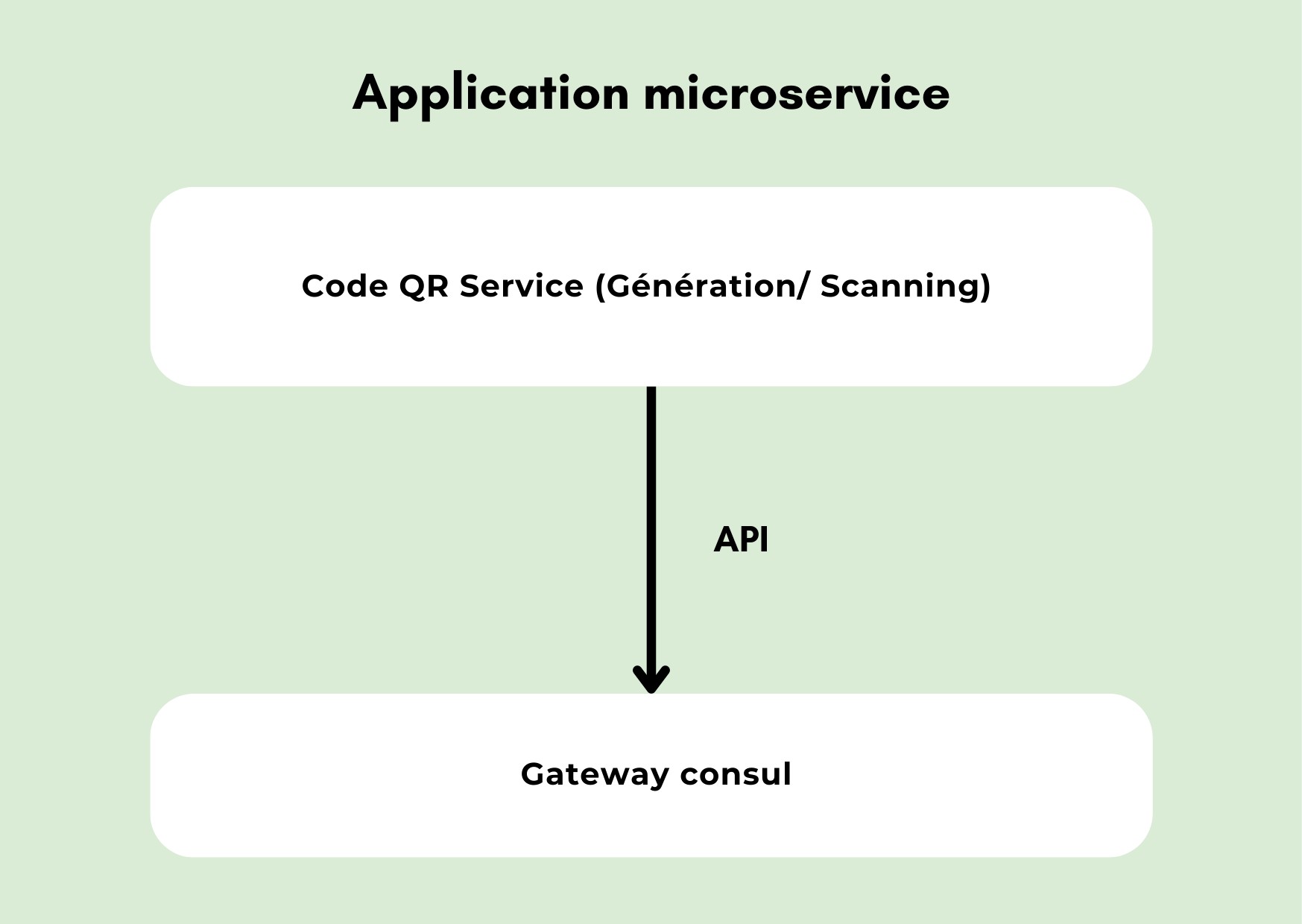
### Description des services :

Le microservice actuel prend en charge deux fonctionnalités principales :

* Génération de Codes QR : Création de codes QR à partir des données fournies.
* Scanning des Codes QR : Interprétation des codes QR capturés à l'aide d'une caméra ou d'un scanner.

API Gateway :

* Un point d'entrée central qui gère les requêtes de l'application.
* Contient des routes et des mappings pour rediriger les requêtes vers les microservices appropriés.
* Il offre une interface uniﬁée pour les clients, simpliﬁant l'accès aux fonctionnalités.



### Mécanisme de communication :

Le service de passerelle (gateway) utilise généralement HTTP ou HTTPS pour la communication avec les microservices. La communication entre la passerelle et les microservices est basée sur les protocoles standard de transfert de données sur le web. Voici quelques points importants à considérer :

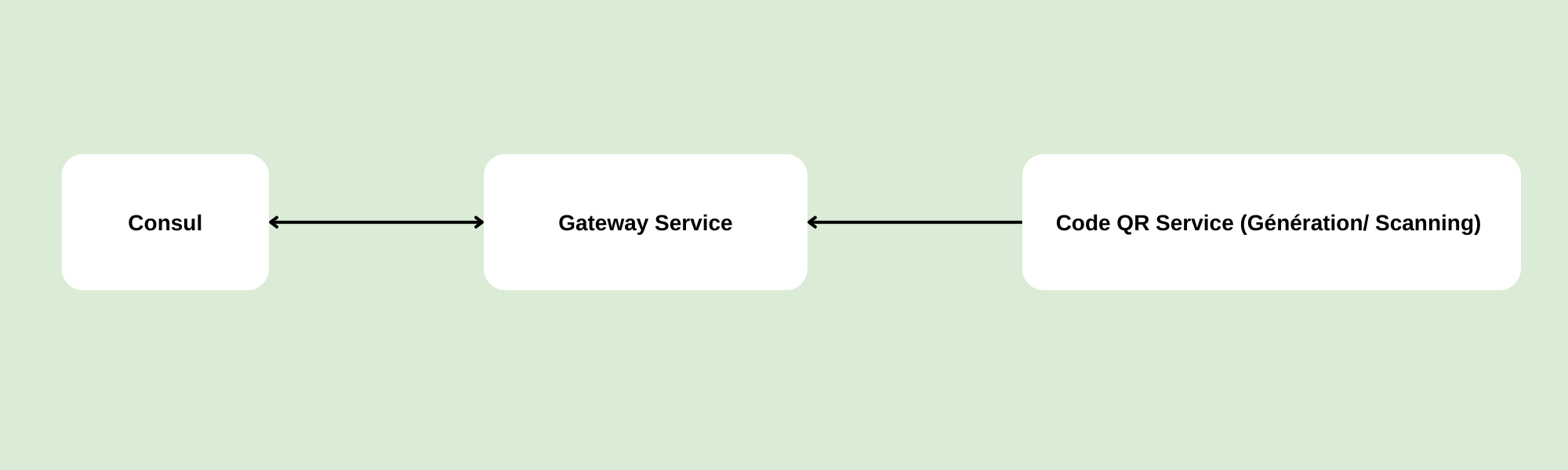
#### Protocole HTTP/HTTPS :

* La plupart des passerelles, comme Spring Cloud Gateway, utilisent le protocole HTTP ou sa version sécurisée, HTTPS, pour la communication avec les microservices.
* HTTP est un protocole sans état, ce qui signiﬁe que chaque requête est indépendante des autres. HTTPS ajoute une couche de sécurité en chiffrant les données.

#### RESTful API :

* Les passerelles sont souvent utilisées pour mettre en œuvre des API RESTful, où les microservices exposent des points de terminaison (endpoints) HTTP standard pour effectuer des opérations CRUD (Create, Read, Update, Delete).

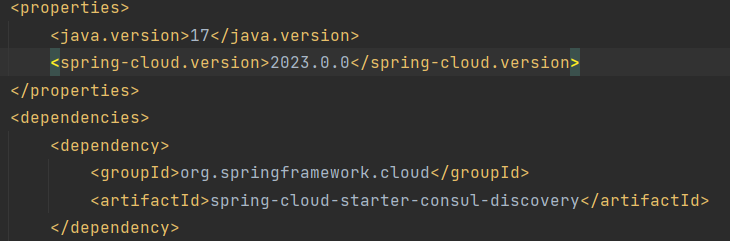
## Conception des Microservices

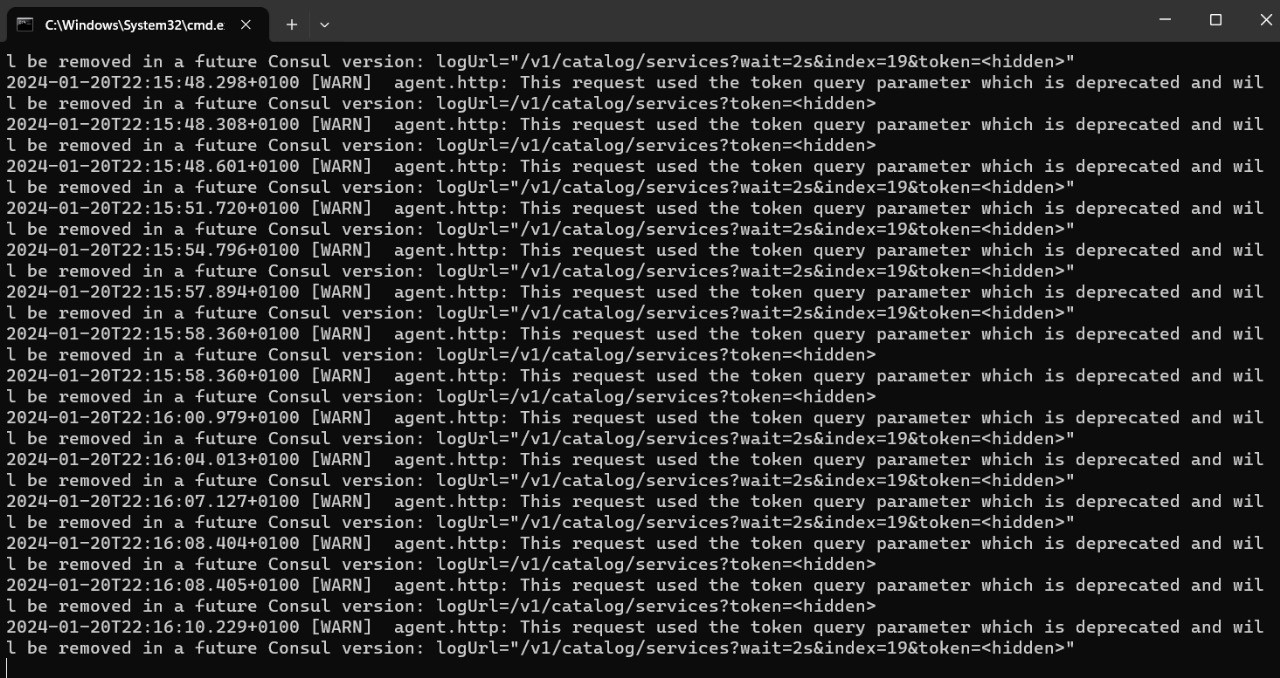


**l'API Gateway** simpliﬁe l'accès à notre microservice en agissant comme une interface uniﬁée pour les clients. Les requêtes sont redirigées vers les fonctionnalités appropriées dans le Code QR Service. Bien que l’application n'ait actuellement qu'un seul microservice, cette architecture offre une extensibilité pour l'ajout de nouveaux services indépendants à l'avenir.

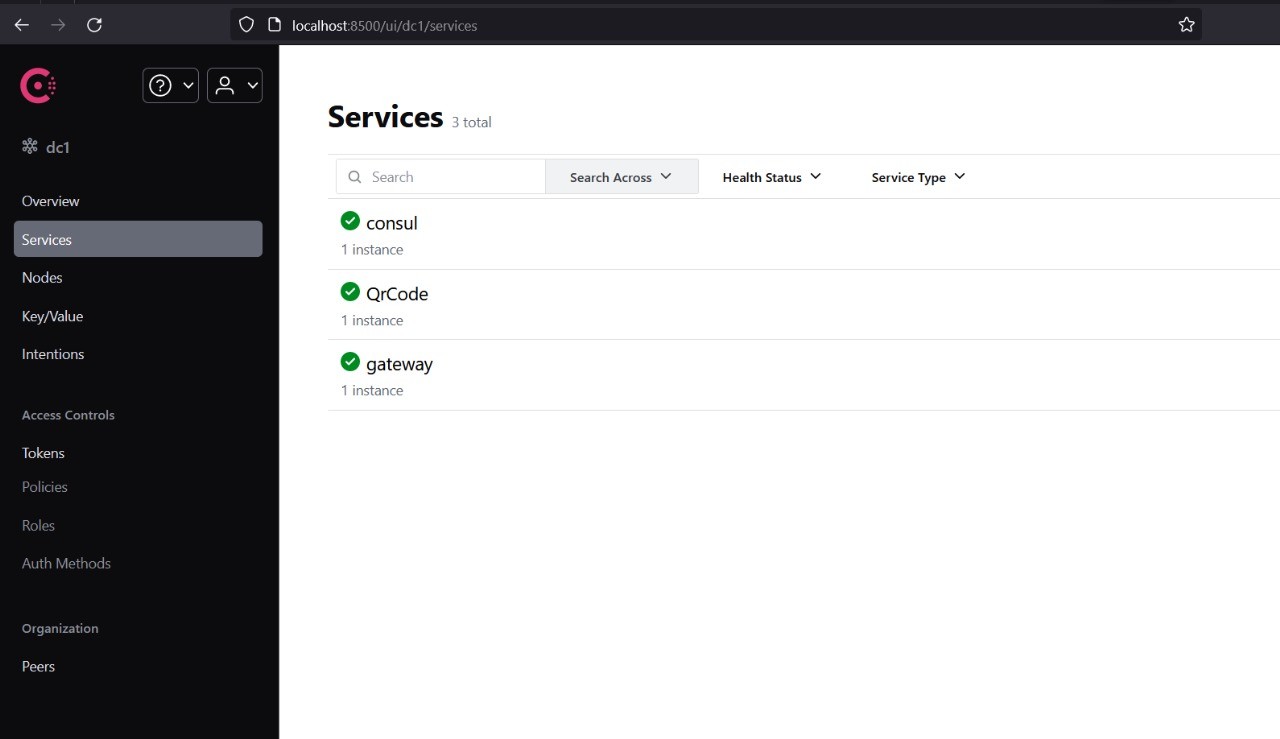
**Consul** offre des fonctionnalités pour la découverte de services, la gestion de la conﬁguration, et la mise en place de solutions de réseau de service (service mesh). Techniquement, Consul se compose de plusieurs composants qui fonctionnent ensemble pour fournir ces fonctionnalités clés.

Integration de dependence et version consul :





**Trois micro services :**

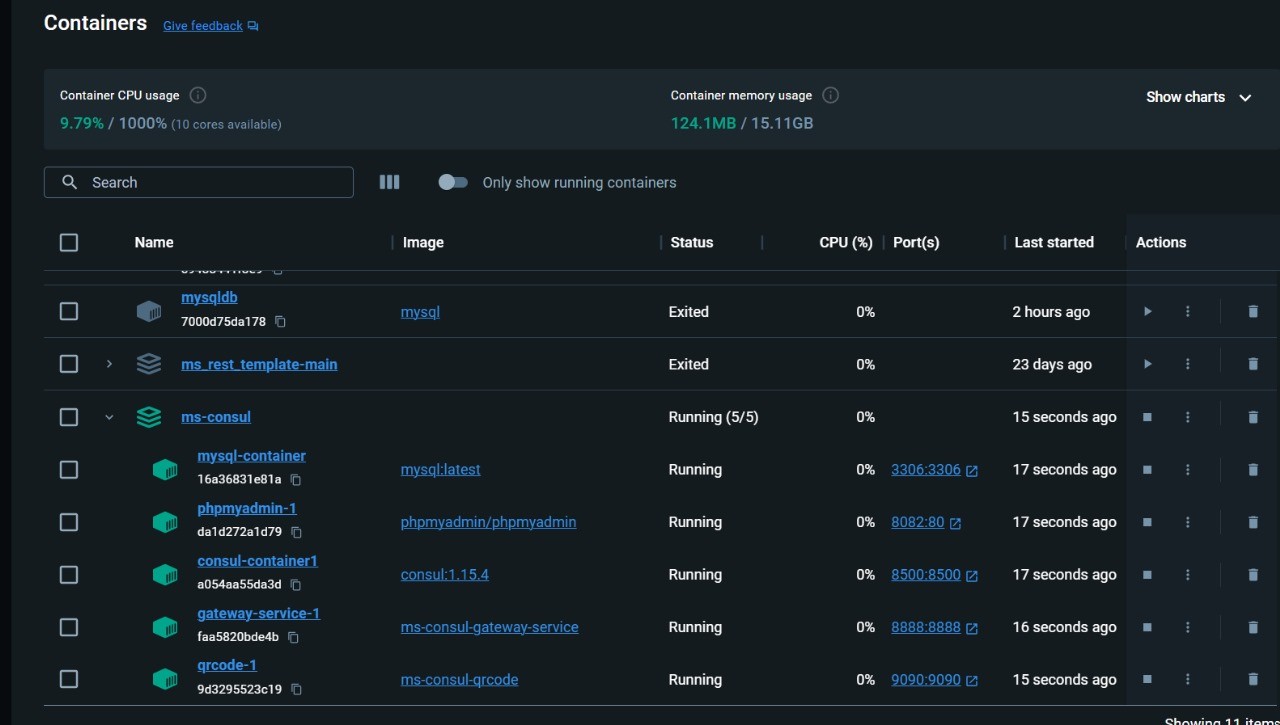


## Conteneurisation avec Docker

### Avantage :

L'utilisation de Docker apporte une plus grande ﬂexibilité, une isolation eﬃcace, une facilité de gestion, et une reproductibilité accrue dans le déploiement et la gestion d'applications de microservices.

### Implementation :



## CI/CD avec Jenkins

**CI/CD** vise à créer un pipeline d'automatisation qui permet une livraison continue et régulière de logiciels de haute qualité. L'intégration continue garantit que les changements sont intégrés rapidement et sans heurts, tandis que le déploiement continu permet une livraison automatisée et ﬁable des logiciels vers des environnements de production.

**Jenkins** est un outil open source d'intégration continue et de déploiement continu (CI/CD) largement utilisé dans le développement logiciel. Il offre un moyen automatisé de construire, tester et déployer des applications, facilitant ainsi le processus de développement.

Automatisation des Processus : Jenkins automatise les processus de build, de test et de déploiement, ce qui élimine les tâches manuelles répétitives et réduit les erreurs humaines.

Gestion des Conﬁgurations : Jenkins facilite la gestion des conﬁgurations en automatisant la création et le déploiement d'infrastructures, permettant une approche Infrastructure as Code (IaC).

Collaboration : Jenkins favorise la collaboration entre les équipes de développement et d'exploitation en fournissant un point centralisé pour la gestion des pipelines et des conﬁgurations.

—----------------------------------------------

Le script de pipeline déﬁnit l'ensemble du ﬂux de travail, spéciﬁant les étapes à exécuter de manière séquentielle :

pipeline { agent any tools {

maven 'maven'

}

stages {

stage('Git Clone') { steps {

script {

checkout([$class: 'GitSCM', branches: [[name: 'main']], userRemoteConﬁgs: [[url: 'https://github.com/SAYOUT-Ahmed/JenkinsMicroService.git']]])

}

}

}

stage('Build QrCode App') { steps {

bat 'cd qrcode && mvn clean install -DskipTests'

}

}

stage('Build Gateway') { steps {

bat 'cd gateway consul && mvn clean install -DskipTests'

}

}

stage('Docker Build and Run') { steps {

script {

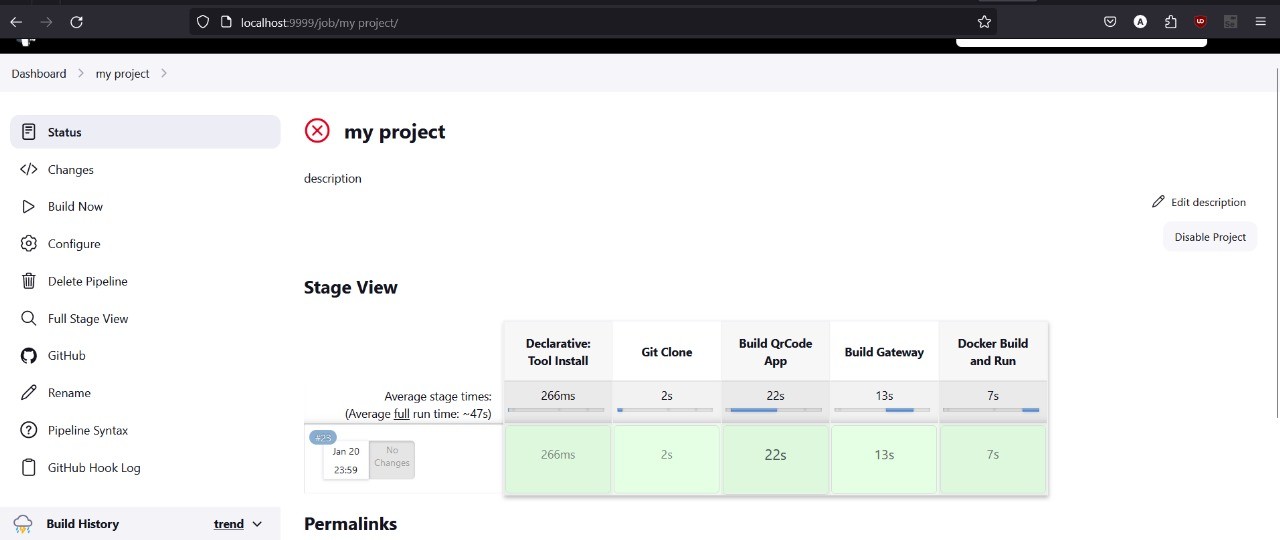
bat 'docker-compose build' bat 'docker-compose up'

}

}

}

}

}

## Intégration de SonarQube

SonarQube est un outil essentiel pour les équipes de développement cherchant à maintenir un code de haute qualité, à détecter rapidement les problèmes potentiels et à assurer la conformité aux normes de codage.

SonarQube est souvent utilisé dans les projets de développement logiciel :

* **Détection Précoce des Problèmes :** SonarQube analyse le code source à la recherche de problèmes potentiels, d'erreurs, de vulnérabilités de sécurité et

d'autres violations des bonnes pratiques de codage. Cela permet de détecter les problèmes dès le début du processus de développement.

* **Amélioration de la Qualité du Code :** SonarQube évalue la qualité du code en attribuant des mesures et des indicateurs aux différents aspects du code, tels que la complexité, la duplication, la couverture de code, etc. Cela permet aux équipes de mesurer et d'améliorer la qualité du code au ﬁl du temps.
* **Intégration Continue :** SonarQube peut être intégré dans des pipelines d'intégration continue (CI) pour automatiser l'analyse du code à chaque nouvelle modiﬁcation.

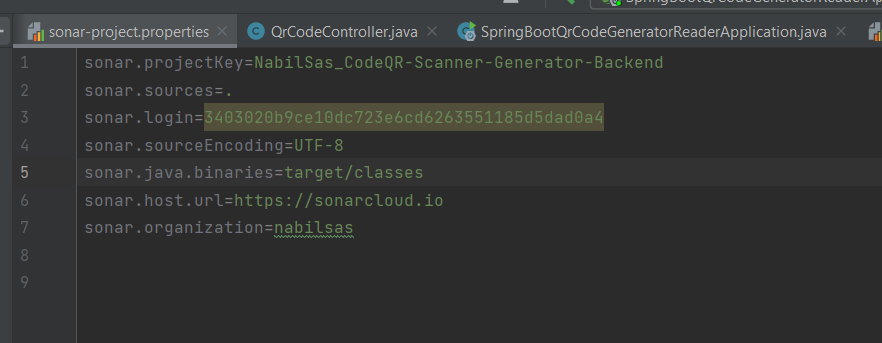
Cela assure une évaluation constante de la qualité du code tout au long du processus de développement.

* **Prise de Décisions Informées :** En fournissant des indicateurs et des métriques sur la qualité du code, SonarQube permet aux équipes de développement, aux

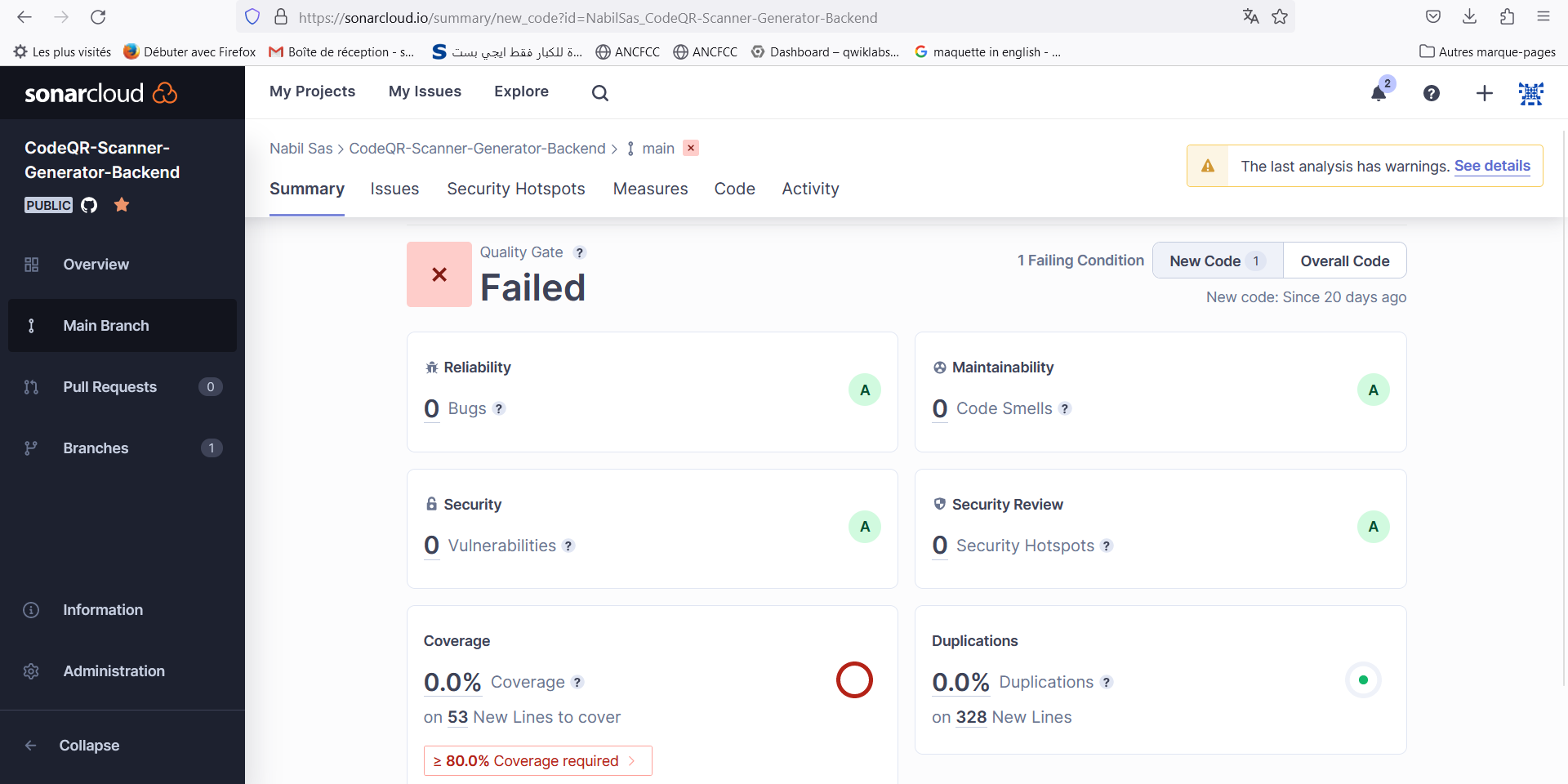
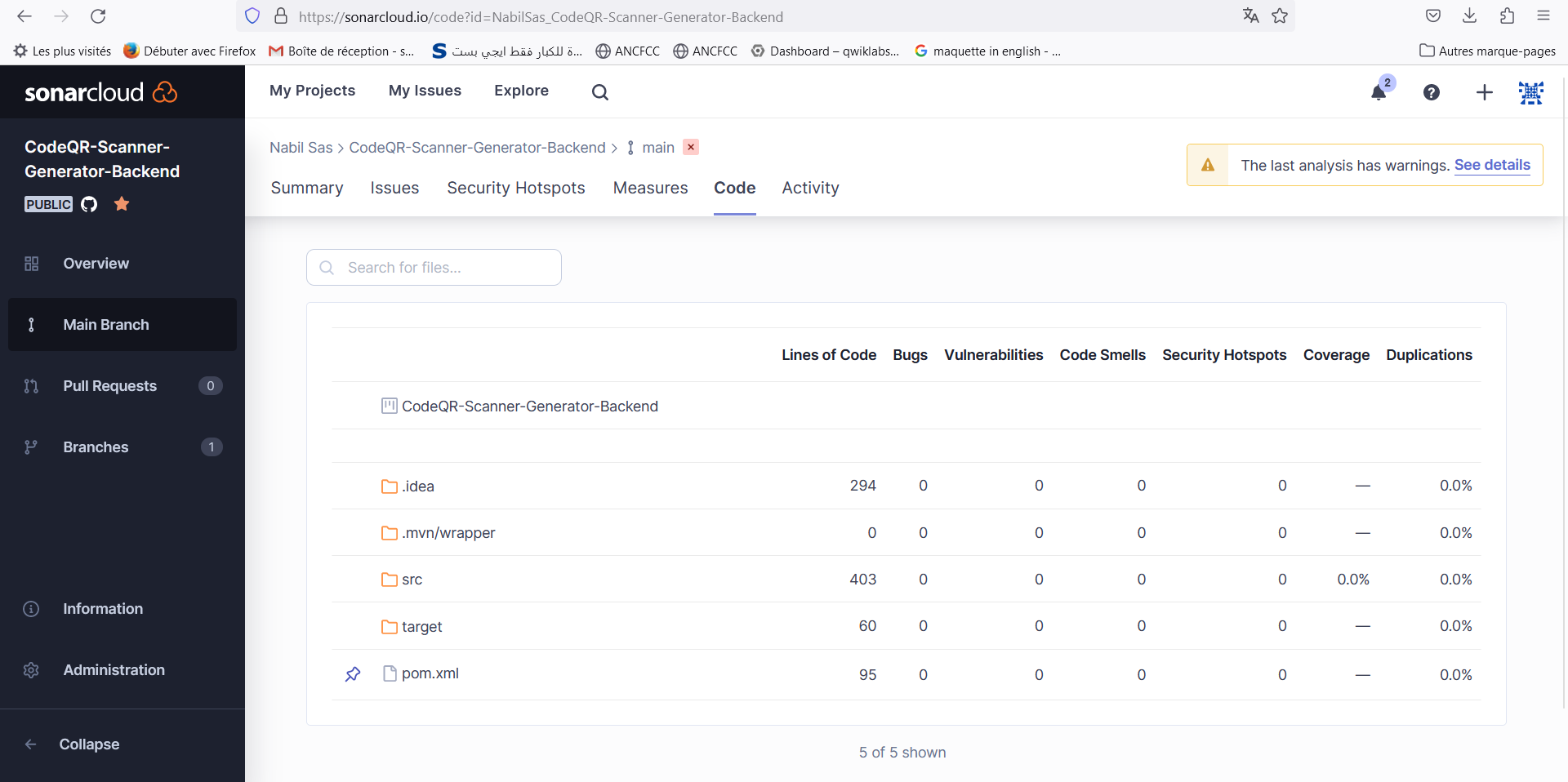
gestionnaires de projet et aux responsables techniques de prendre des décisions informées pour l'amélioration continue du logiciel.

* **Conformité aux Standards de Codage :** SonarQube permet de déﬁnir et d'appliquer des règles de codage personnalisées ou basées sur des normes de l'industrie (par exemple, les règles MISRA pour le langage C). Cela garantit que le code est conforme aux standards déﬁnis par l'équipe ou l'organisation.

Fichier **sonar-project.properties** intégré dans la racine du projet :







## Conclusion

En conclusion, ce projet de QR Code Scanner et Générateur, basé sur une architecture microservices, offre une solution polyvalente et moderne pour répondre aux besoins croissants de génération et de lecture de QR Codes. L'utilisation de Spring Boot pour le backend et React JS pour le frontend assure une infrastructure robuste et une interface utilisateur réactive.

L'architecture microservices a permis de décomposer le système en composants autonomes, offrant ainsi une scalabilité indépendante et une maintenance simpliﬁée. Les

microservices spécialisés dans la génération et la lecture des QR Codes garantissent une eﬃcacité opérationnelle optimale.

Le projet évoluera en intégrant des fonctionnalités avancées pour la personnalisation des QR Codes, explorant la réalité augmentée. L'internationalisation, la sécurité renforcée, l'analyse de données, et les applications mobiles seront prioritaires pour offrir une

expérience globale et sécurisée.

Ce projet offre un socle solide pour l'évolution continue, permettant d'explorer de nouvelles opportunités technologiques et fonctionnelles. En alignant les futures évolutions sur les

besoins du marché et les retours des utilisateurs, ce projet peut devenir un outil indispensable dans le domaine en constante évolution de la génération et du scanning de QR Codes.