

# Rapport du Projet QR App Scanner & Generator

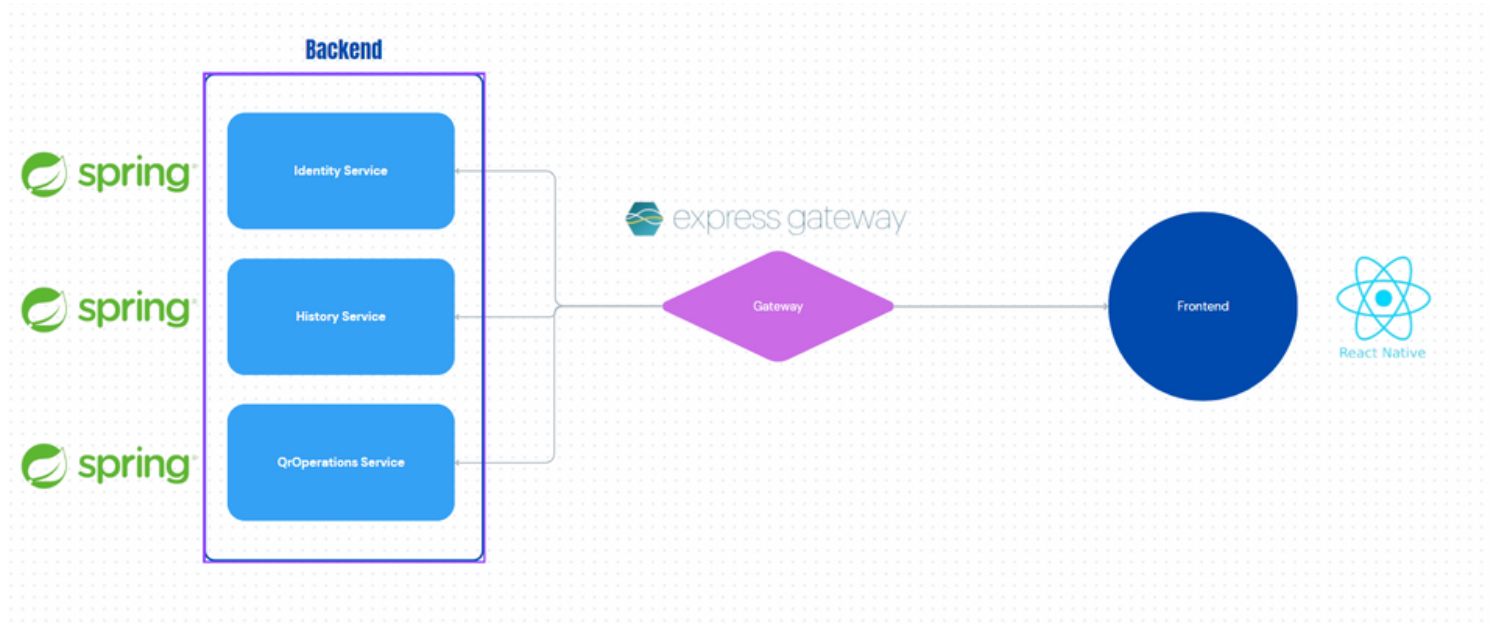
## 1- Introduction:

Le projet QRApp Scanner et Generator vise à fournir une application robuste de gestion de codes QR avec des fonctionnalités de numérisation, de génération et de suivi d'historique. Pour garantir la scalabilité, la flexibilité et la maintenabilité du système, une architecture microservices a été adoptée.

L'architecture microservices est importante car elle permet de décomposer un système complexe en composants indépendants, ce qui facilite le développement, le déploiement et la maintenance des services.

## 2- Architecture Microservices:

- *Architecture:*



- *Description des services:*

Le service d'identité fournit des points de fin pour l'authentification des utilisateurs.

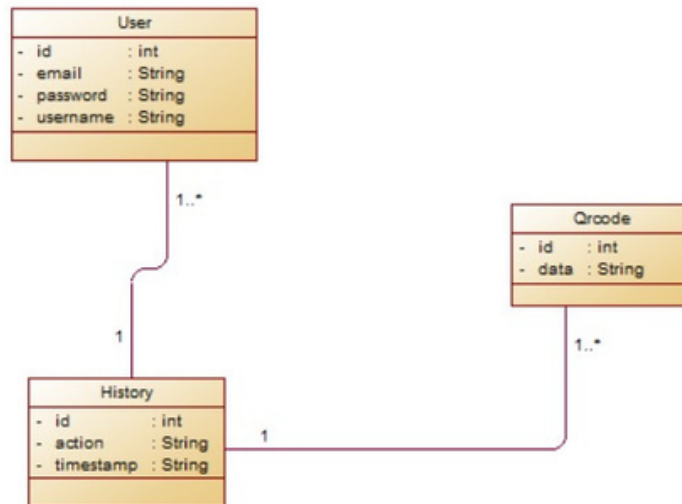
Le service d'historique permet de sauvegarder et de récupérer l'historique des opérations.

Le service QrOperations gère les opérations liées aux codes QR.

- **Mécanismes de communication:**

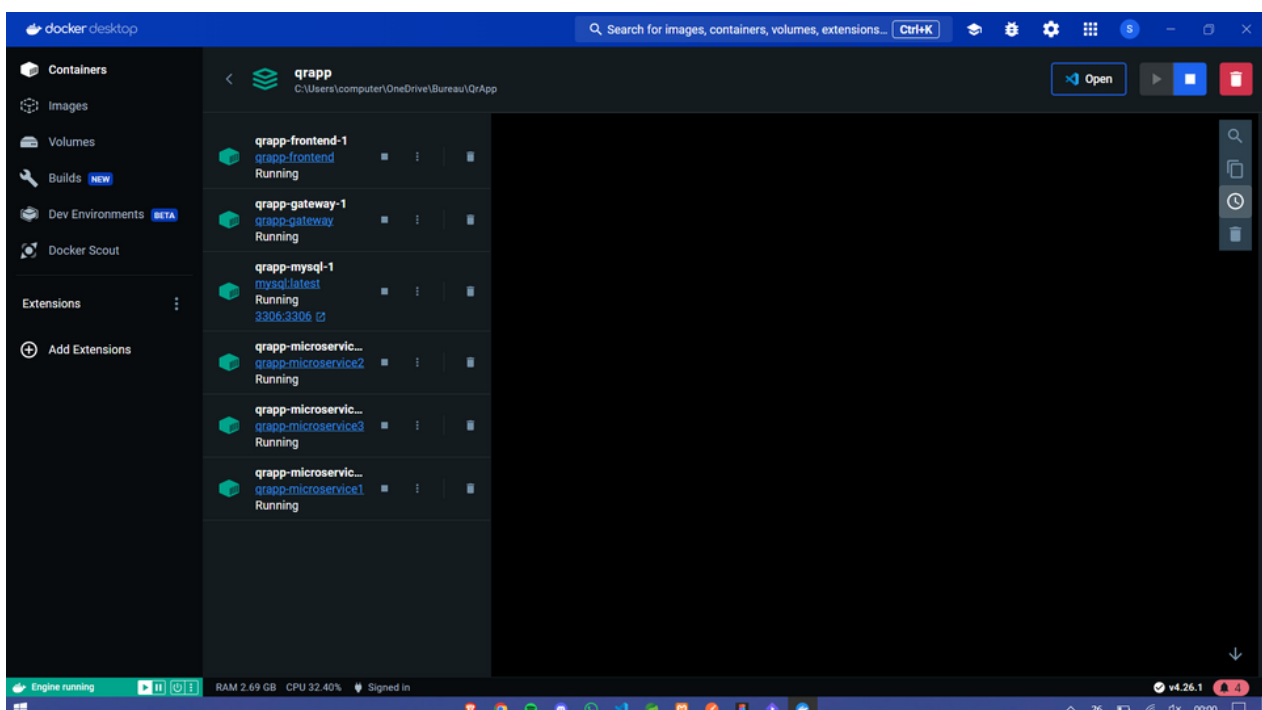
Utilisation d'Express Gateway pour la communication entre les microservices et le frontend.

### 3- Conception des Microservices:



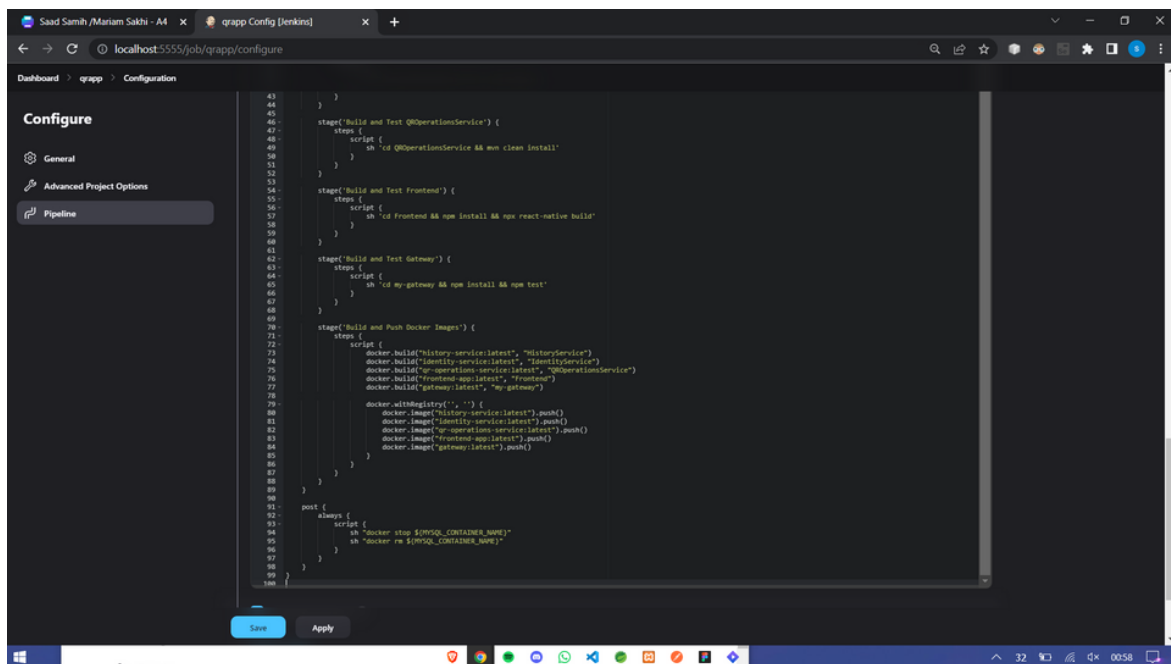
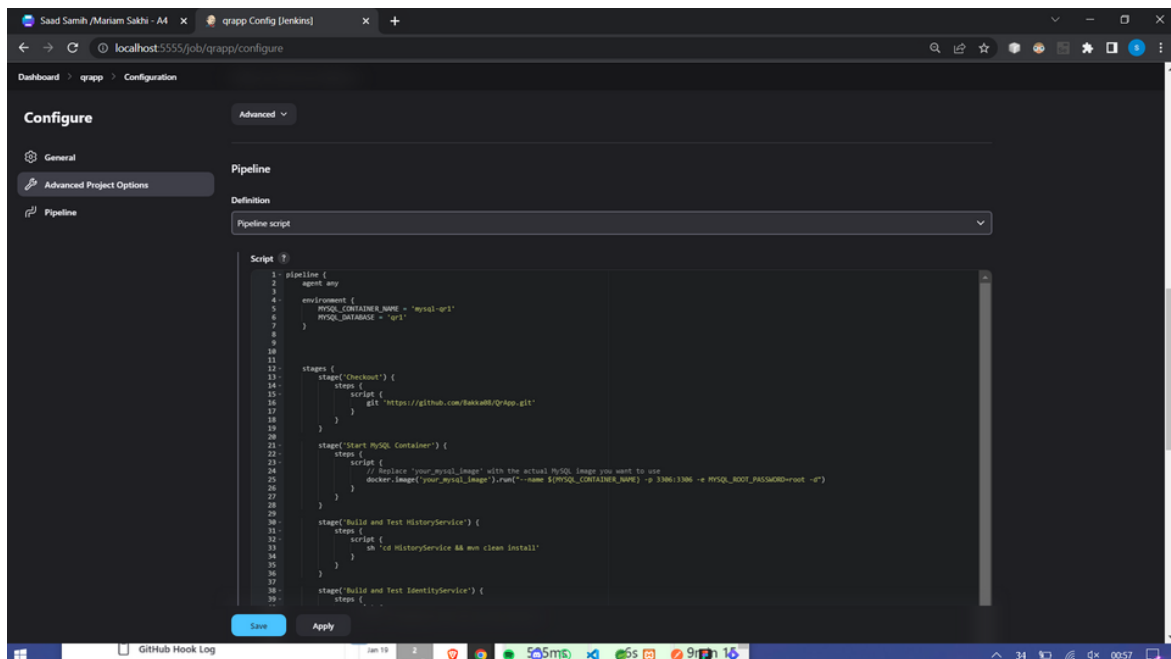
### 4- Conteneurisation avec Docker:

Chaque microservice est conteneurisé avec Docker, ce qui facilite le déploiement et la gestion des dépendances, en plus l'isolation des environnements, la portabilité et la facilité de mise à l'échelle sont des avantages.



## 5- CI/CD avec Jenkins:

- pipeline



Declarative: Tool Install	Git Clone	Build Backend	Create Docker Image (Backend)	Build and Create Docker Image (Frontend)	Run (Backend and Frontend)	Declarative: Post Actions
234ms	3s	3min 18s	36s	2min 20s	31s	198ms
585ms	6s	9min 15s	1min 33s	12min 33s	2min 12s	284ms

## 6- Déploiement Automatique:

```
ngrok

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

Session Status      online
Account             ayoub (Plan: Free)
Version             3.5.0
Region              Europe (eu)
Latency              85ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://c1aa-105-67-131-87.ngrok-free.app -> http://localhost:5555

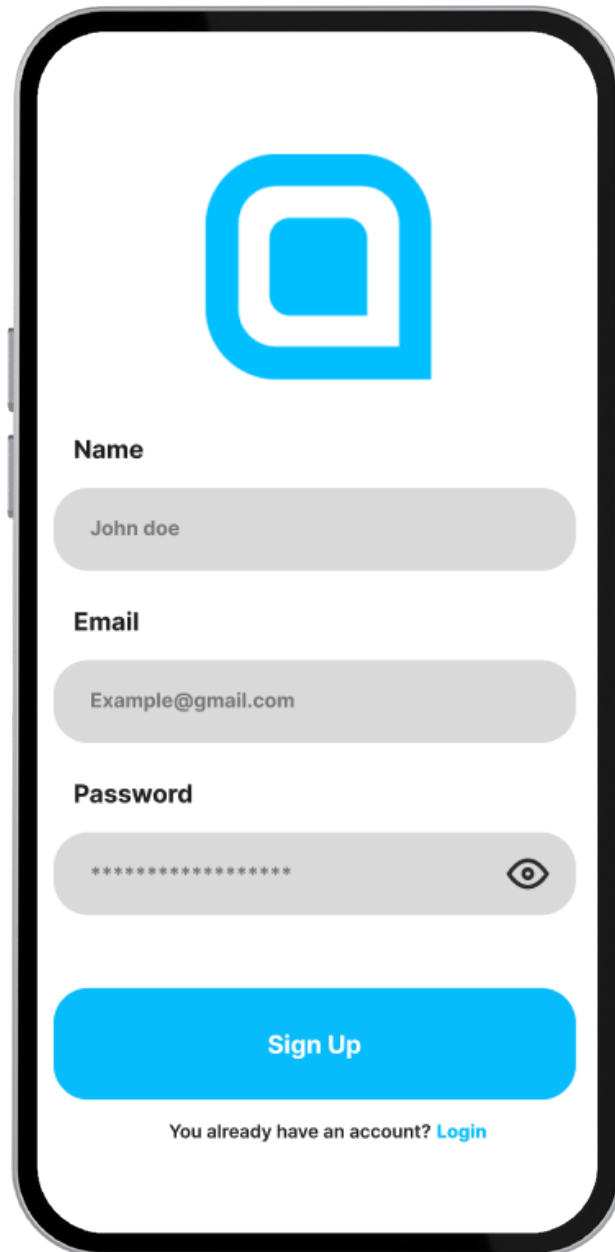
Connections          ttl      opn      rt1      rt5      p50      p90
                    80       2        0.29    0.19    0.35    12.12
```

## 7- Intégration de SonarQube

The screenshot displays the SonarCloud web interface. On the left, there is a sidebar with filters for Quality Gate (Passed, Failed), Reliability, Security, Security Review, and Maintainability. The main area shows a list of four projects, each with a summary of its analysis results. The projects are: QrApp / Test (PUBLIC), Identityservice / test (PUBLIC), Historyservice / test (PUBLIC), and QrOperationsservice / test (PUBLIC). Each project card displays the last analysis date and time, the number of lines of code, and various quality metrics including Bugs, Vulnerabilities, Hotspots Reviewed, Code Smells, Coverage, and Duplications. The Identityservice / test project is marked as 'Failed'.

Project	Last analysis	Lines of Code	Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications
QrApp / Test (PUBLIC)	12/31/2023, 2:34 PM	1.1k	6	0	0.0%	24	0.0%	1.0%
Identityservice / test (PUBLIC)	12/31/2023, 2:49 PM	346	0	1	0.0%	5	0.0%	0.1%
Historyservice / test (PUBLIC)	12/31/2023, 2:52 PM	401	0	0	0.0%	2	0.0%	8.2%
QrOperationsservice / test (PUBLIC)	12/31/2023, 2:55 PM	341	0	0	0.0%	4	0.0%	0.0%

## 8- capture d'écran de l'application:



The screenshot shows a mobile application interface for signing up. At the top is a blue logo consisting of a square with a smaller square inside. Below the logo are three input fields: 'Name' with the text 'John doe', 'Email' with the text 'Example@gmail.com', and 'Password' with masked characters '\*\*\*\*\*'. To the right of the password field is an eye icon. Below these fields is a large blue button labeled 'Sign Up'. At the bottom, there is a link that says 'You already have an account? Login'.

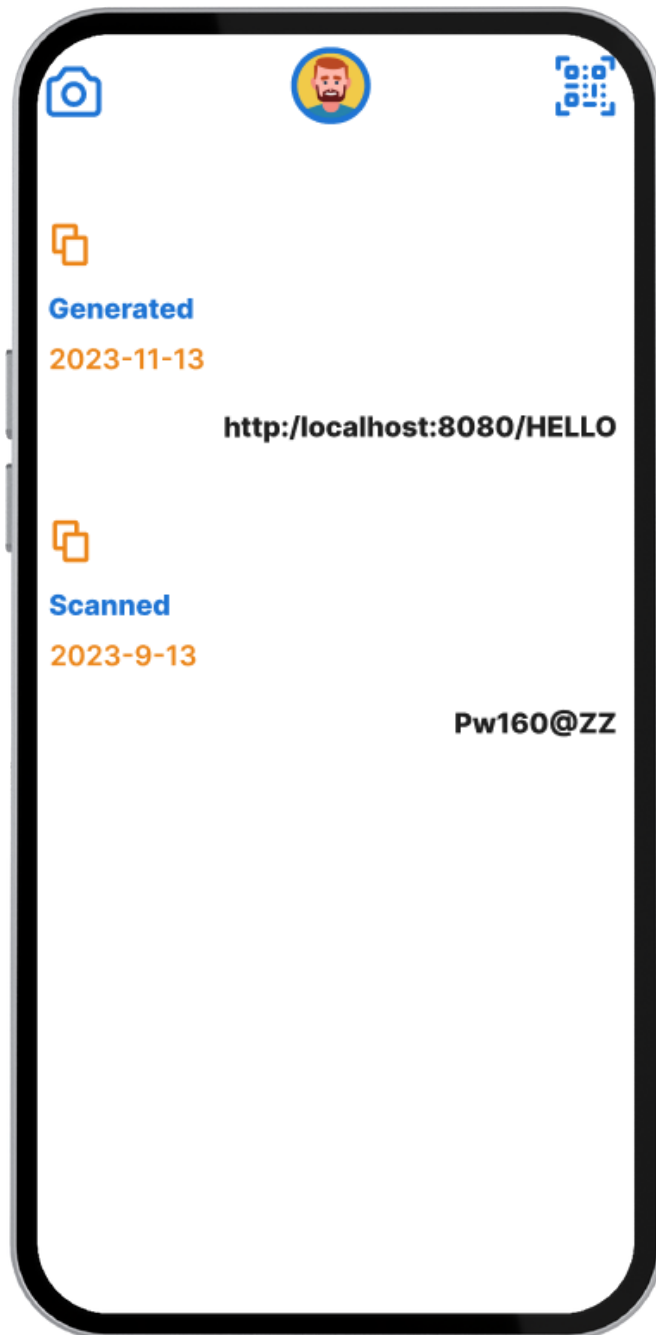
### SignUp page

- Cette interface montre l'affichage du formulaire du signUp avec les champs nécessaires pour s'inscrire contenant l'email et mot de passe.



## Login page

- Cette interface montre l'affichage du formulaire du login avec les champs nécessaires pour saisir les identifiants de l'utilisateur contenant l'email et mot de passe.



## Home page

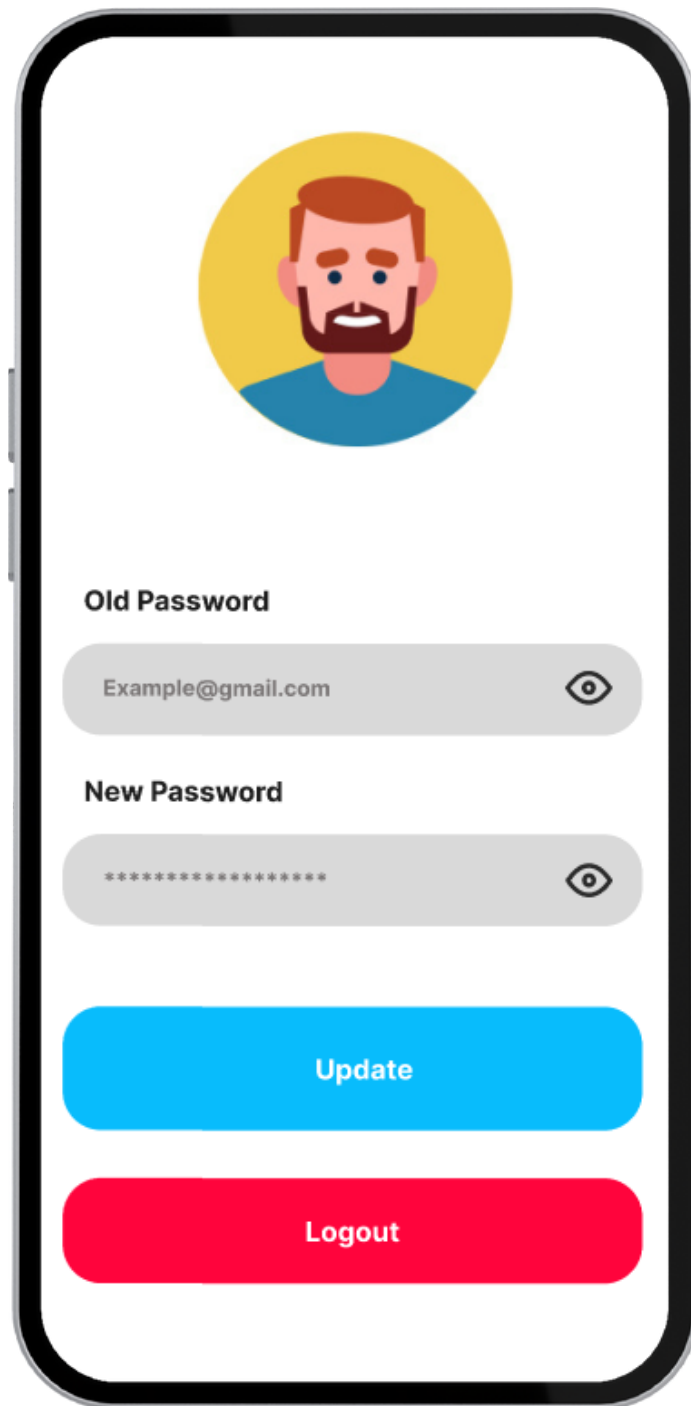
- Cette page d'accueil contient l'historique des codes QR générés ou scannés de l'utilisateur avec trois icônes en haut : caméra pour scanner le code QR, le profil de l'utilisateur et enfin icône QR pour générer un code QR.



## QR code scanné

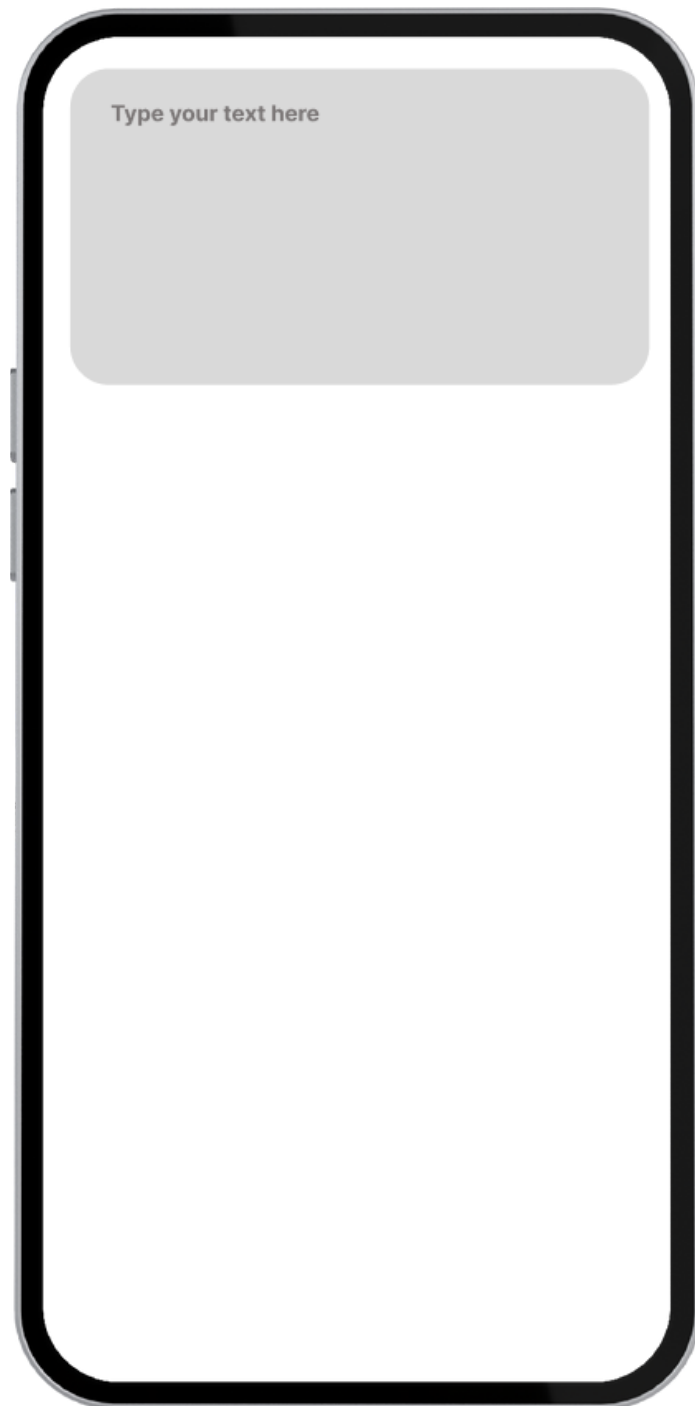
- Un QR code scanné par l'utilisateur qui se trouve dans son historique.





## Profile page

- L'utilisateur a la possibilité de modifier son mot de passe à partir de son profil.



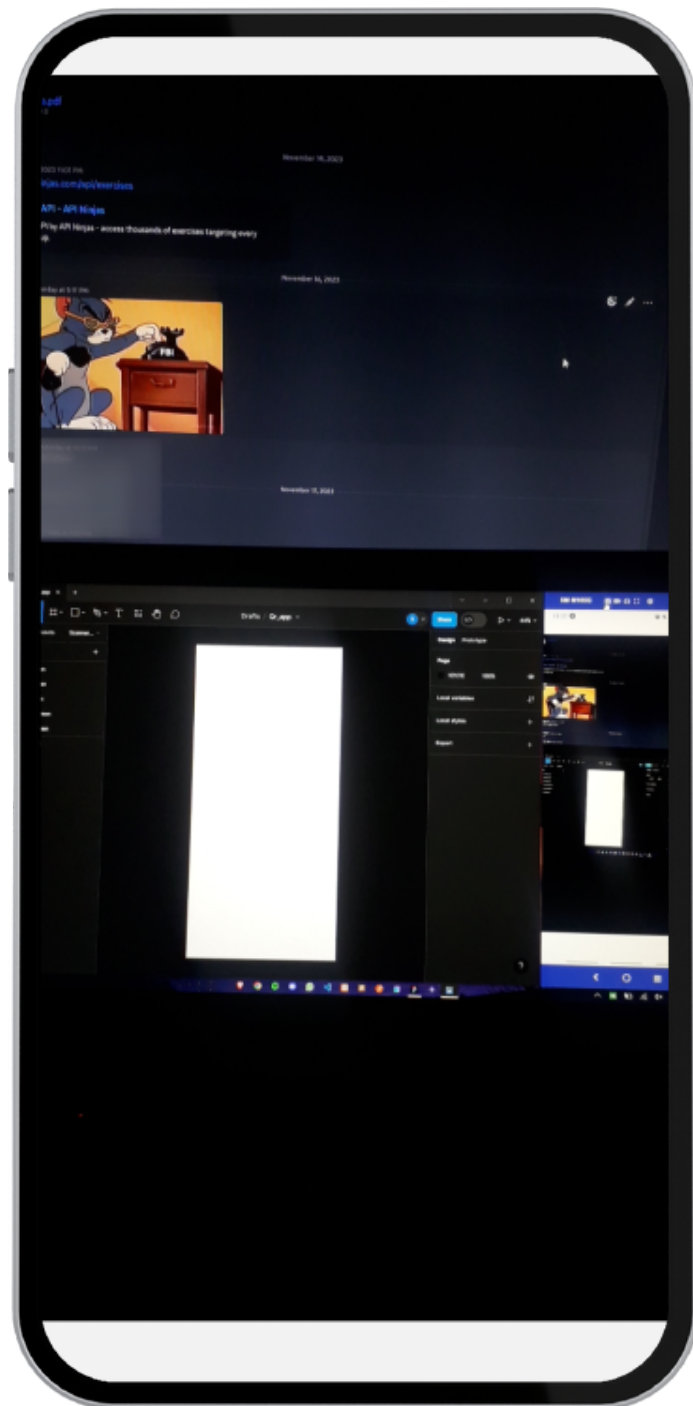
## QR generator

- Cette interface montre un champs pour saisir un lien a partir duquel l'utilisateur veut générer le code QR.



## Code QR généré

- Cette interface nous montre que un code QR est généré a partir du lien donne par l'utilisateur.



## Camera pour scanner un code QR

- Cette interface nous montre une camera pour scanner un code QR.

## 8- Conclusion

- Le projet QRApp Scanner et Generator adopte une architecture microservices pour offrir une solution flexible et évolutive.
- L'utilisation de Docker, Jenkins et SonarQube améliore la qualité du code et facilite le déploiement continu.