

**CONTIFY**

*Gestion Des Contacts*

Membres

Mohamed Menfalouti

Mohamed Ismail Menhal

Plan

Introduction

1

.

Architecture

2

.

Conteneurisation : Docker

3

.

CI/CD Jenkins

4

.

Ngrok

5

.

SonarQube

6

.

Application Screenshots

7

.

Conclusion

8

.

**Introduction**

**Aperçu du projet**

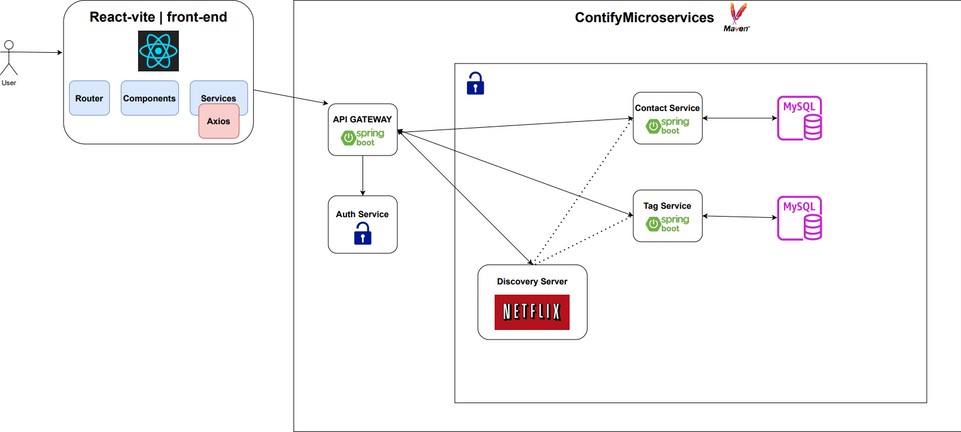
Contify représente une solution complète pour la gestion et l'organisation efficaces de vos contacts. Adoptant une architecture microservices, ce projet vise à offrir une modularité et une évolutivité accrues pour répondre aux besoins dynamiques de la gestion des contacts.

# Importance de l'architecture microservices

Nous avons opté pour l’architecture microservices en raison de ses nombreux bénéfices. Grâce à elle, chaque service peut être géré séparément, ce qui améliore la capacité à monter en charge, la robustesse et la facilité de maintenance. De plus, cette architecture favorise une innovation permanente, puisque chaque service peut être conçu, mis en place et amélioré de façon autonome.

# Architecture

Le projet est structuré autour de plusieurs services interconnectés, chacun jouant un rôle spécifique et complémentaire dans l'écosystème global de Contify.



* Api Gateway Service : En tant que point d'entrée, il gère le routage, l'authentification et d'autres aspects transversaux, fournissant une interface unifiée pour les clients externes.
* Discovery Server : Essentiel pour la découverte et l'enregistrement des services, il établit un environnement hautement scalable et tolérant aux pannes en facilitant la communication dynamique entre les services.
* Contact Service : Au cœur de la gestion des contacts, ce service offre une gamme complète de fonctionnalités, de la manipulation des entités à la communication transparente avec la base de données. Tag Service : Axé sur les opérations liées aux tags, il offre une solution modulaire et dédiée à la gestion efficace des tags.
* Auth Service : Gérant les aspects cruciaux de l'authentification et de l'autorisation de manière sécurisée, il assure la protection des données sensibles.

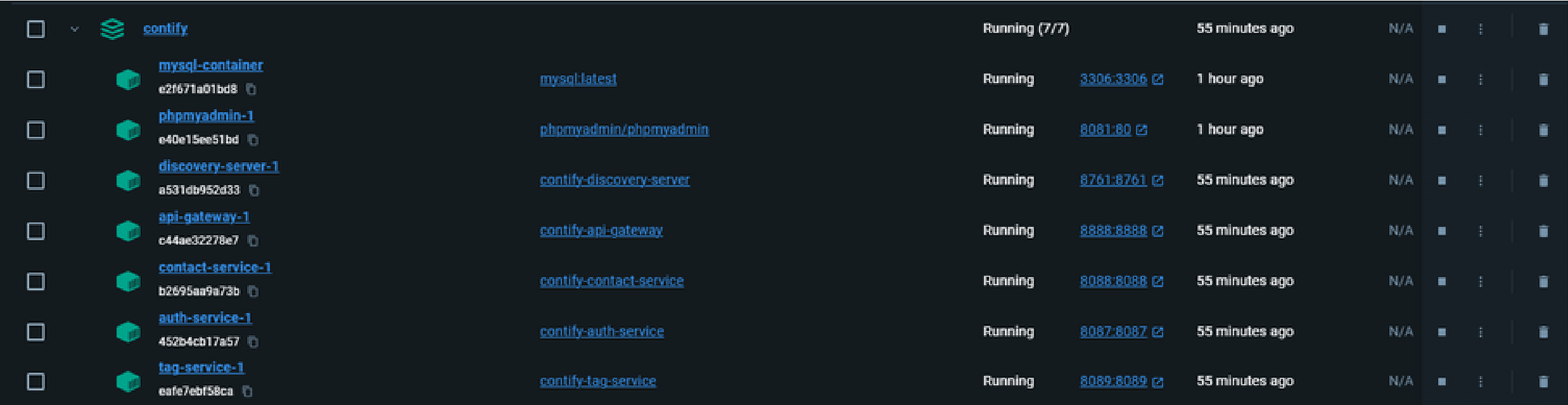
**Mécanismes de communication**

Eureka, notre mécanisme de choix pour la découverte et l'enregistrement des services, favorise une communication dynamique entre les différentes parties du système, garantissant une extensibilité sans heurts. OpenFein pour l’affectation des tags aux contacts.

## 

## Conteneurisation : Docker

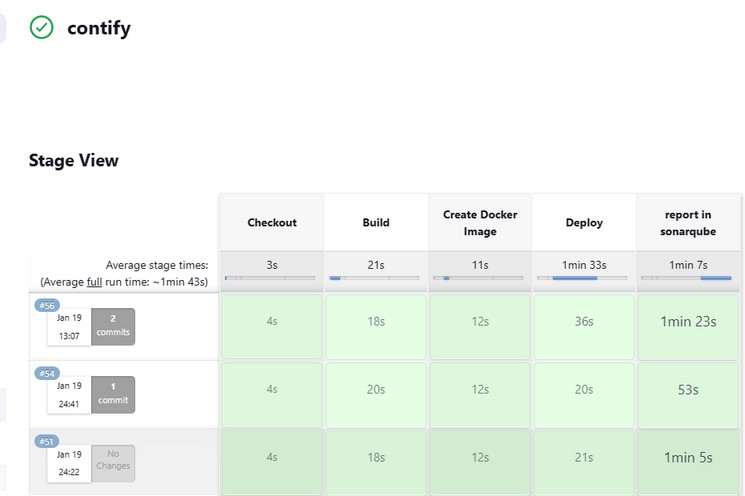
### Implémentation et avantages

La conteneurisation avec Docker apporte une valeur ajoutée significative au projet. En encapsulant chaque service dans un conteneur, nous assurons une portabilité, une isolation et une facilité de déploiement accrues. Les conteneurs offrent également un environnement cohérent, éliminant les problèmes potentiels liés aux différences d'environnement entre les étapes de développement et de production.

## CI/CD avec Jenkins

### Processus et configuration

Jenkins, notre outil d'intégration continue, automatise le processus de construction, de test et de déploiement. Cela garantit une livraison rapide et fiable des nouvelles fonctionnalités et corrections, tout en minimisant les erreurs liées au déploiement manuel. La configuration précise de Jenkins assure une harmonie entre les différentes étapes du processus CI/CD.



### Pipline Script



**Définition de l'Agent :**

agent any : Ce pipeline peut être exécuté sur n'importe quel agent disponible (ouvrier Jenkins).

**Stages:**

Le pipeline comporte plusieurs stages, chacun représentant une phase dans le processus de construction et de déploiement.

**Stage(Checkout) :**

Cette étape récupère le code source depuis le référentiel GitHub spécifié. Alors qu’elle va cloner le repo depuis l'URL donnée et récupère la branche 'master'.

**Stage(Build) :**

Cette étape construit le projet en utilisant Maven.  **bat 'mvn clean install'** : Exécute une commande batch Windows pour exécuter

Maven, nettoyer le projet, puis le construire.

**Stage(Create docker image) :**

Cette étape crée une image Docker en utilisant le fichier Docker Compose.

**bat 'docker-compose build'** : pour construire les images Docker.

**Stage(Deploy) :**

Cette étape déploie l'application en utilisant Docker Compose.

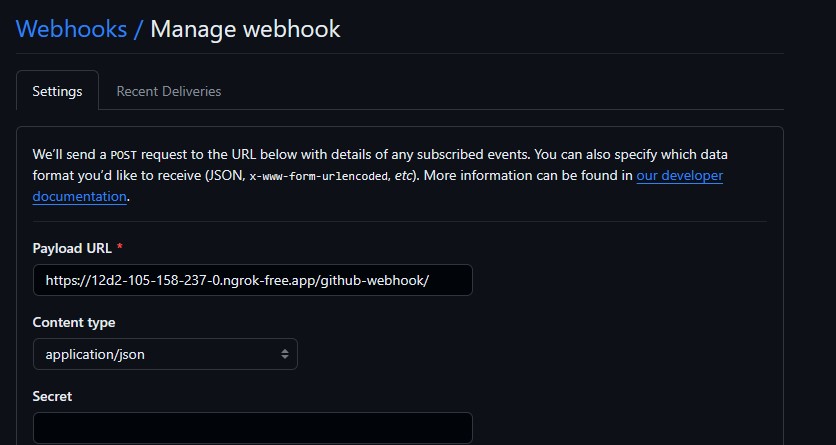
**Stage(Repport to SonarQube):**

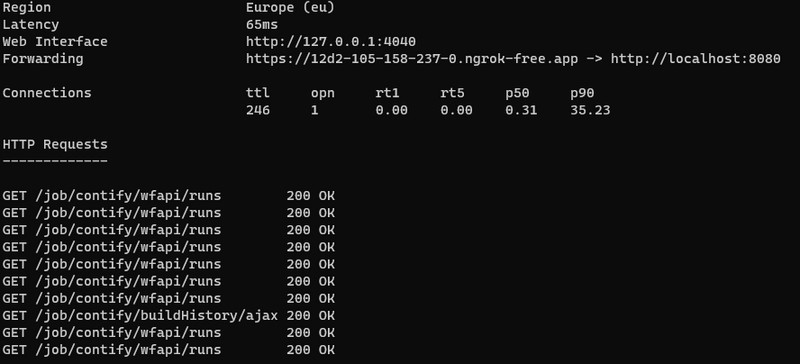
Cette étape génère un rapport d'analyse de code en utilisant SonarQube.

## Ngrok & WebHooks

### Utilisation de Ngrok

L'utilisation de Ngrok assure un fonctionnement en temps réel du système, permettant une expérience utilisateur transparente.

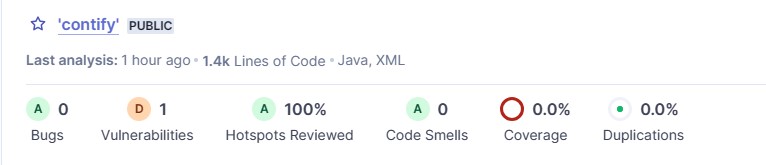




## SonarQube

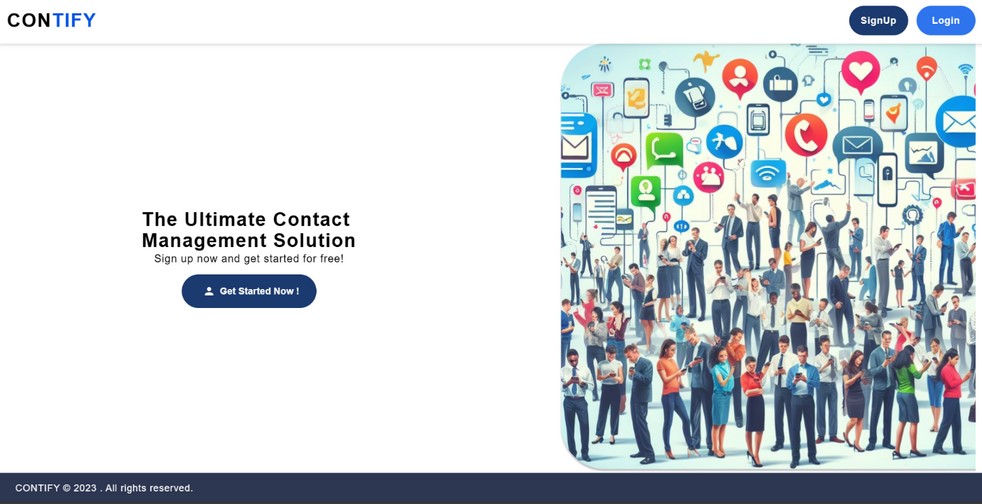
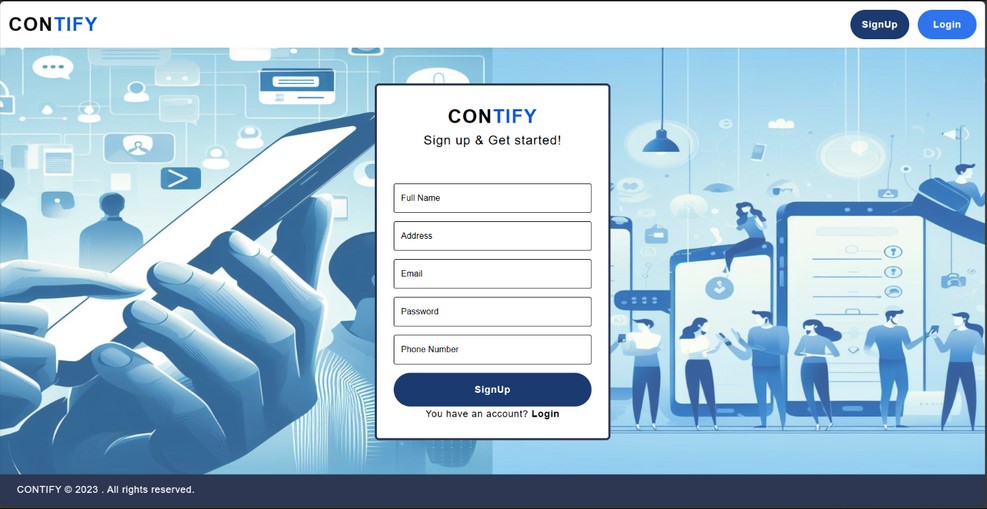
### Configuration et bénéfices pour la qualité du code

SonarQube, intégré dans notre pipeline, analyse le code à chaque commit. En fournissant des informations détaillées sur la qualité du code, les vulnérabilités et les erreurs potentielles, il renforce notre engagement envers un code propre, sécurisé et facilement maintenable.

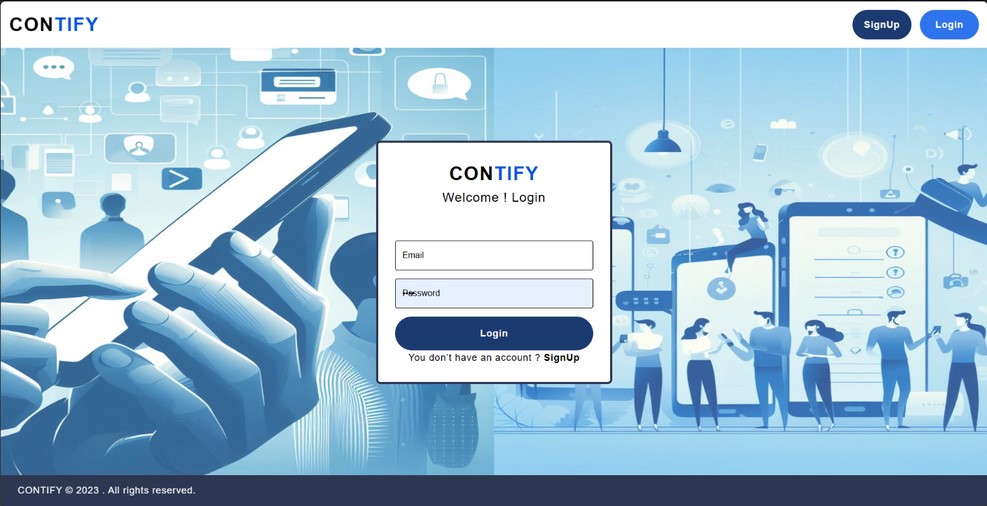


## Application Screenshots

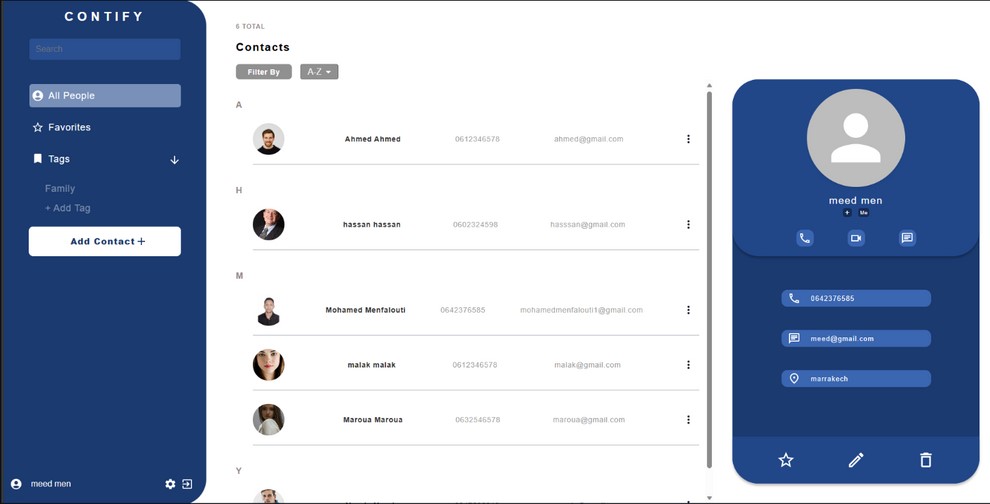
**Landing Page**

**Sign Up Page**

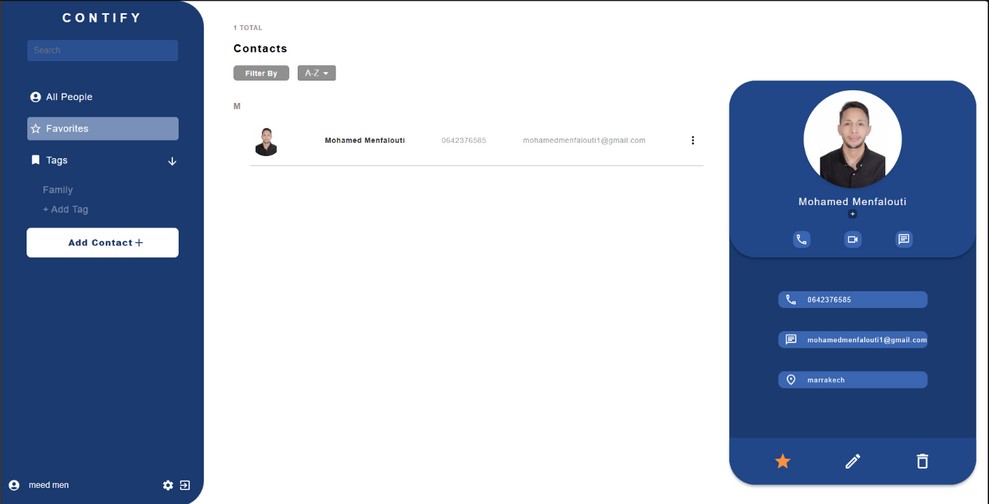
### Login Page



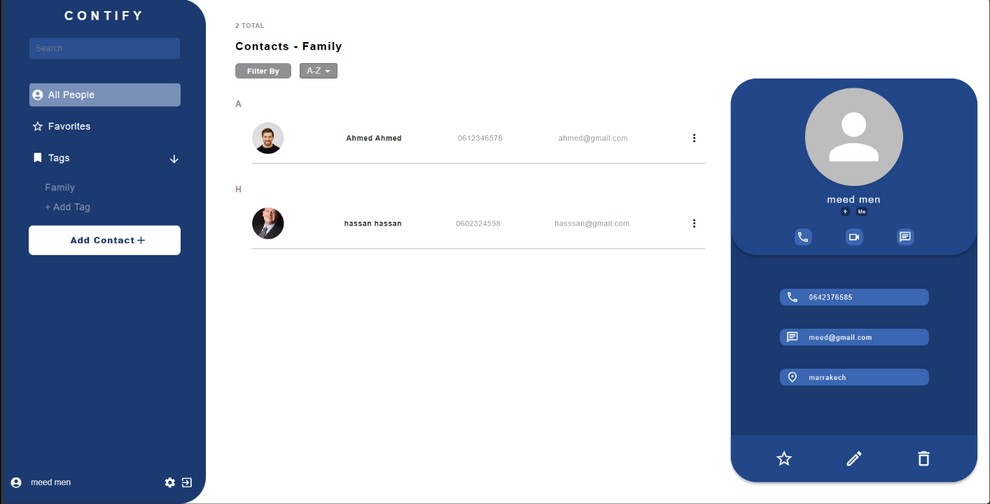
**All People Page**



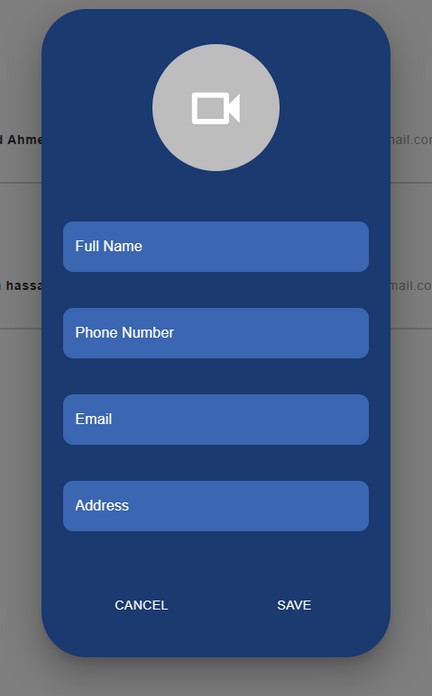
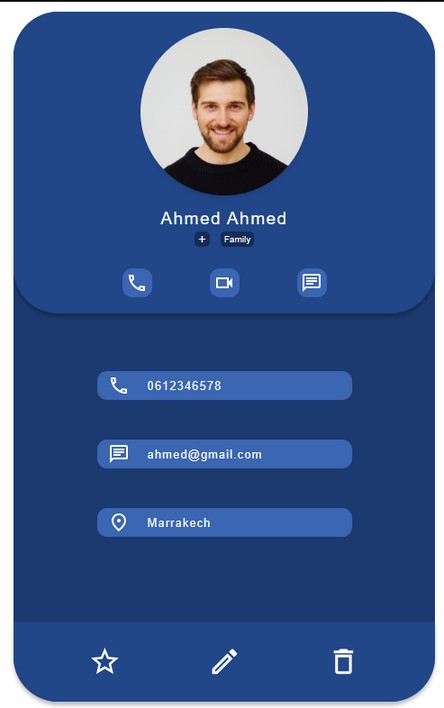
**Favorites Page**



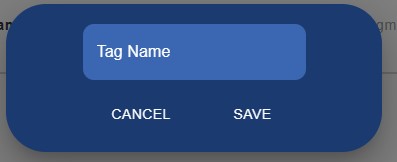
### Tag Page



**Contact Infos Add Contact**



**Add Tag**



## Conclusion

### Résumé des accomplissements

Contify offre une solution complète et moderne pour la gestion de contacts. L'architecture microservices, la conteneurisation, l'intégration continue et l'analyse de code automatique contribuent à la robustesse et à la qualité du système.

### Perspectives futures

Le projet est prêt pour des évolutions futures. Des ajouts de fonctionnalités, des optimisations de performance et l'exploration de nouvelles technologies sont envisageables pour répondre aux besoins changeants de la gestion de contacts.

### Github Repos

Front-End :

<https://github.com/Ismailmenhal/contify-frontend>

back-End :

https://github.com/MeedMn/contifyMicroservices