**Rapport Projet : Architecture des composants d’entreprise**

Groupe : **El Otmani Moad** / **Benchlikha Nouh** / **Elbazi Mohamed Amine** Projet : **Application Web de recette cuisine**

# Introduction

## Aperçu du projet

**Partie Administrateur :**

Le panneau d'administration est spécialement conçu pour les gestionnaires et les administrateurs de l'application. Ces utilisateurs auront un accès privilégié pour gérer les profils des utilisateurs, superviser les recettes, et garantir le bon fonctionnement global de la plateforme. Ils pourront ajouter, modifier ou supprimer des recettes, gérer les utilisateurs, et assurer la qualité et la variété des contenus disponibles.

**Partie Utilisateur Simple :**

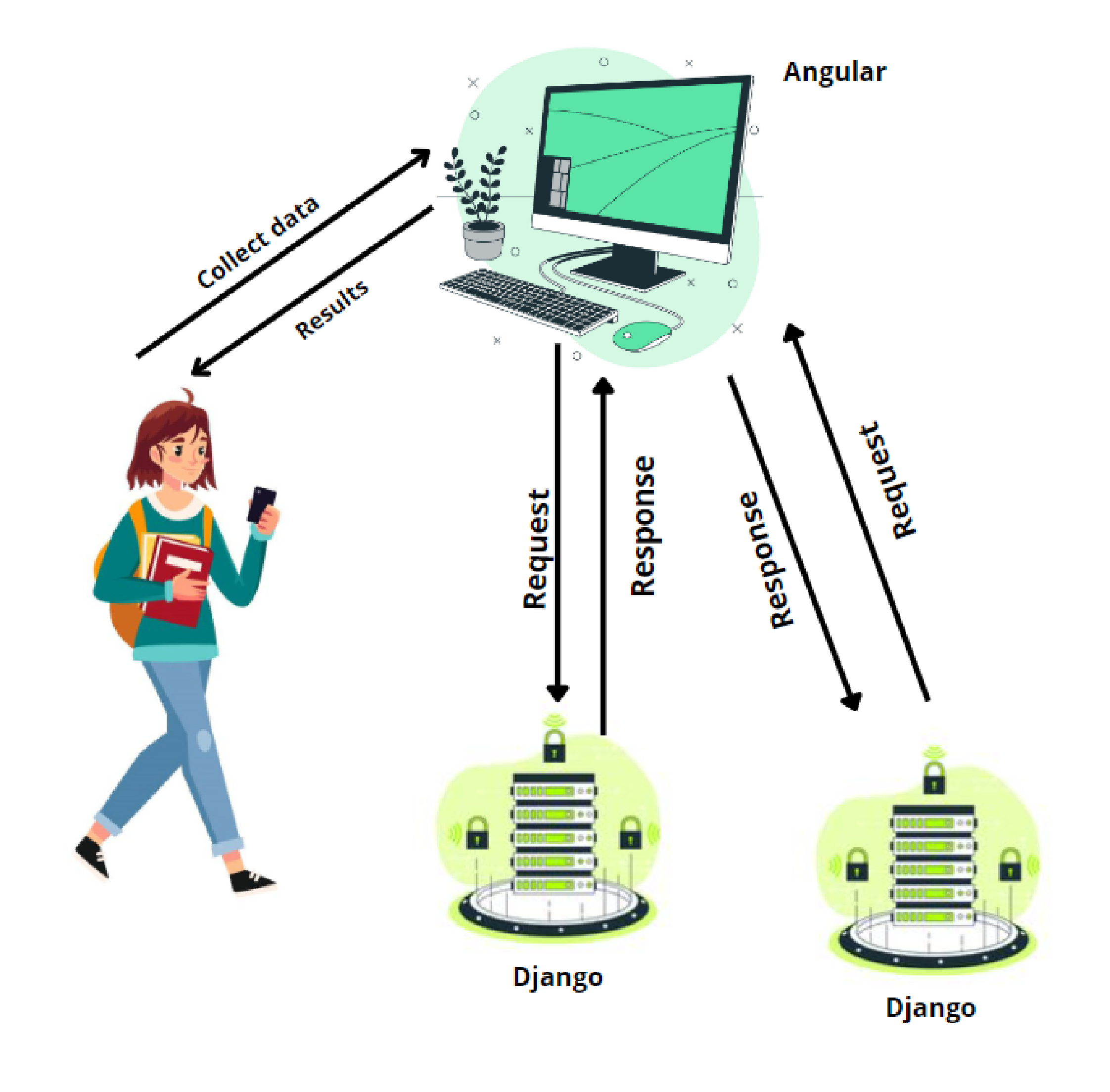
Cette section de l'application est destinée aux utilisateurs réguliers qui souhaitent explorer, contribuer et interagir avec la communauté culinaire. Les fonctionnalités comprennent la recherche intuitive de recettes, la possibilité de commenter et de noter les recettes.

## Importance de l'architecture microservices

L'architecture microservices est devenue cruciale dans le développement logiciel en raison de ses avantages, notamment une évolutivité facile, une flexibilité accrue, une gestion d'équipe simplifiée et une résilience améliorée. Cette approche modulaire permet des déploiements plus fréquents et fiables, bien que son adoption doive être adaptée aux besoins spécifiques du projet.

# Architecture Microservices

## Architecture



## Description des services

Le backend de notre application repose sur le framework Django, tandis que la communication entre les microservices s'effectue via la bibliothèque "requests". Le premier microservice englobe l'authentification, la gestion des utilisateurs, ainsi que la gestion des recettes. Le deuxième microservice, quant à lui, est dédié aux statistiques de l'application, comprenant le nombre d'utilisateurs, de recettes, de commentaires, ainsi que d'avis.

## Mécanismes de la communication

Les mécanismes de communication au sein de notre architecture reposent sur l'utilisation du framework Django pour la partie backend, et la bibliothèque "requests" pour faciliter les échanges entre les microservices. Django fournit une base solide pour la création de services robustes, tandis que l'utilisation de "requests" permet une communication efficace et flexible entre les différentes parties de l'application. Cette approche garantit une intégration fluide des fonctionnalités, favorisant ainsi une interaction harmonieuse entre le microservice gérant l'authentification, la gestion des utilisateurs et des recettes, et celui traitant les statistiques de l'application, notamment le nombre d'utilisateurs, de recettes, de commentaires, et d'avis.

# Conception des Microservices

Pour chaque service au sein de notre architecture microservices, nous avons adopté une approche de conception spécifique afin de répondre de manière optimale à ses responsabilités distinctes.

**Microservice d'Authentification et de Gestion des Utilisateurs et Recettes :**

Dans ce service, l'accent est mis sur la sécurité et la gestion des données utilisateur. Nous avons mis en œuvre des fonctionnalités d'authentification robustes, telles que la gestion des sessions et des jetons d'accès. La conception s'appuie sur des modèles Django pour la gestion efficace des utilisateurs et des recettes. Les endpoints API sont soigneusement définis pour permettre une interaction sécurisée avec le frontend, garantissant un contrôle d'accès approprié aux fonctionnalités sensibles.

**Microservice de Statistiques de l'Application :**

Ce service est axé sur la collecte et la présentation des statistiques importantes de l'application. Nous avons conçu des routines efficaces pour recueillir les données sur le nombre d'utilisateurs, de recettes, de commentaires, et d'avis. L'utilisation de caches et d'agrégats de données contribue à garantir des performances optimales lors de la génération de rapports statistiques. Les API exposées permettent aux autres parties de l'application de récupérer ces statistiques de manière efficace.

# Conteneurisation avec Docker

## Implémentation et avantages

**Premier Backend (Authentification et Gestion des Utilisateurs/Recettes) :**

Ce backend est encapsulé dans un conteneur Docker défini par un Dockerfile spécifique. Les dépendances, configurations et étapes d'exécution nécessaires sont incluses dans le Dockerfile.

**Deuxième Backend (Statistique) :**

Le deuxième backend, dédié aux statistiques de l'application, est également encapsulé dans son propre conteneur Docker.

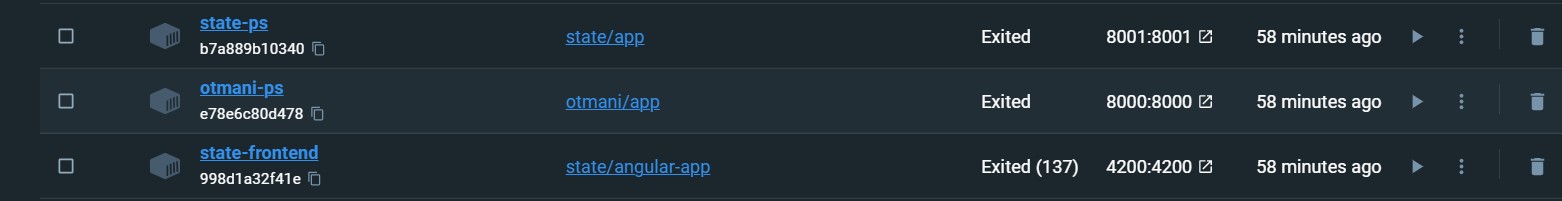
Un Dockerfile spécifique définit l'environnement nécessaire pour ce backend, garantissant son isolation et sa portabilité.

L'utilisation de Docker Compose permet de coordonner le déploiement et la gestion de ce backend avec les autres services.

**Frontend (Angular) :**

Le frontend développé avec Angular est également conteneurisé dans un environnement Docker.

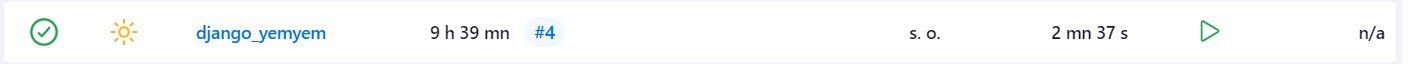
Le Dockerfile associé comprend les instructions pour créer un conteneur autonome contenant l'application Angular.



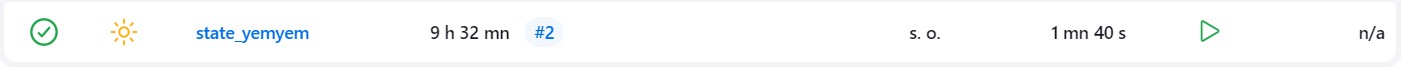
# CI/CD avec Jenkins

## Processus et configuration

La mise en place de l'intégration continue (CI) et du déploiement continu (CD) avec Jenkins repose sur une configuration précise. Pour le processus CI, Jenkins déclenche automatiquement les builds à chaque modification du code source, exécute des tests et génère des artefacts prêts pour le déploiement. Le processus CD assure le déploiement continu sur des environnements spécifiques avec des configurations adaptées. La configuration de Jenkins inclut l'installation de plugins nécessaires, la définition de jobs CI/CD avec des étapes spécifiques, et la gestion sécurisée des secrets et variables d'environnement. Cette approche automatisée améliore l'efficacité du développement, garantit des déploiements fiables et accélère la livraison des applications.



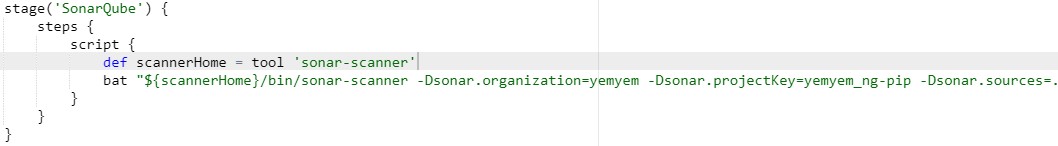


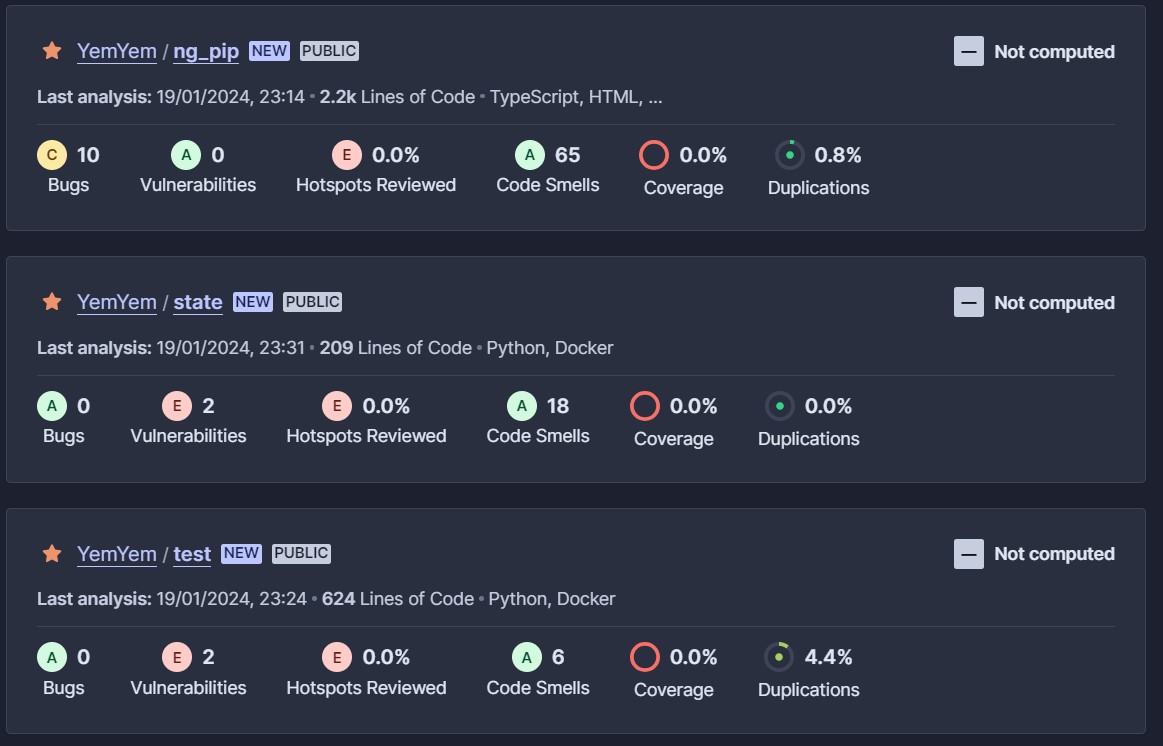


# Intégration de SonarQube

## Configuration et bénéfices pour la qualité du code

L'intégration de SonarQube avec Jenkins constitue une approche stratégique pour garantir la qualité du code tout au long du processus de développement. En configurant Jenkins pour déclencher l'analyse SonarQube à chaque build, nous bénéficions d'une évaluation continue de la qualité du code. SonarQube identifie rapidement les problèmes tels que les bugs, les vulnérabilités, et les violations de normes, offrant aux développeurs la possibilité de résoudre ces problèmes à un stade précoce du cycle de développement. Les métriques fournies par SonarQube, telles que le taux de duplication et la complexité cyclomatique, permettent une évaluation approfondie de la qualité du code, tandis que l'intégration transparente dans Jenkins assure une cohérence dans l'application des normes de codage. En combinant Jenkins et SonarQube, nous renforçons la robustesse et la fiabilité de notre code, soutenant ainsi une approche d'amélioration continue au sein de notre processus de déploiement.





# Conclusion

## Résumé des accomplissements

En conclusion, notre parcours de développement a abouti à une série d'accomplissements significatifs. Nous avons établi une architecture microservices, chacun encapsulé dans des conteneurs Docker, favorisant la modularité, la scalabilité et la facilité de déploiement. L'intégration réussie de Jenkins dans notre pipeline CI/CD a automatisé les processus de build, de test et de déploiement, accélérant ainsi notre cycle de développement. L'intégration de SonarQube avec Jenkins a renforcé la qualité de notre code en identifiant rapidement les problèmes potentiels et en fournissant des métriques approfondies pour l'évaluation continue de la qualité logicielle. Ces réalisations se traduisent par une plus grande agilité, une amélioration constante de la qualité du code, et une facilité de gestion dans notre environnement de développement. Nous sommes désormais bien positionnés pour relever les défis futurs et continuer à fournir des solutions logicielles robustes et fiables.

## Perspectives futures

Dans nos projets futurs, l'envisagement de l'utilisation de RabbitMQ pour l'envoi d'e-mails aux utilisateurs qui créent un compte ou interagissent avec des recettes, que ce soit en laissant un avis ou en marquant une recette comme ratée, représente une piste prometteuse. Cette approche vise à exploiter la messagerie asynchrone de RabbitMQ pour garantir une distribution efficace des notifications, assurant ainsi une communication réactive et ciblée avec nos utilisateurs. Cette perspective souligne notre engagement continu à optimiser l'expérience utilisateur en utilisant des technologies modernes pour fournir des notifications pertinentes et opportunes.