



Projet Final CALC

Dancoisne Sébastien

https://github.com/neoseb59/CALC_final_project

January 7, 2024

Agenda

- Architecture de Microservices
- Docker dans les Microservices
- RabbitMQ pour les Microservices
- Raisons de l'utilisation de FastAPI
- Avantage de RabbitMQ avec FastAPI
- Cas d'utilisation et Bonnes Pratiques
- Bonnes Pratiques
- Conclusion

Architecture de Microservices



Qu'est-ce qu'un Microservice?

Un microservice est une approche de développement logiciel où une application est décomposée en petits services autonomes, spécialisés, favorisant la flexibilité et l'indépendance de développement.

Avantages des Microservices

Les microservices offrent une scalabilité aisée, une maintenance simplifiée, une flexibilité, un déploiement indépendant des services et une technologie adaptée. Les microservices sont mieux adaptés aux grands systèmes et aux équipes distribuées.

Avantage pour notre projet

L'utilisation de RabbitMQ facilite énormément la réalisation d'une telle architecture, car crée un fort découplage entre les composants. Ces derniers s'interfaçent d'une manière fiable et asynchrone à travers une queue.

Docker pour les Microservices

01

Docker est la plateforme de conteneurisation standard dans l'industrie

02

Déploiement rapide

03

Isolation des services

04

Portabilité entre les environnements

05

Utilisation efficace des ressources

RabbitMQ dans les Microservices

1

Découplage crucial entre microservices pour évolutions indépendantes et maintenance simplifiée.

2

Communication asynchrone améliorant la réactivité, favorisant des échanges fluides entre microservices sans attente directe.

3

RabbitMQ assure une gestion souple des pics de charge en mettant en file d'attente les messages, préservant ainsi la performance du système.

4

Garantie de livraison des messages même en cas de dysfonctionnement temporaire, offrant une communication fiable entre microservices.

5

Intégration fluide avec nos microservices, notamment avec FastAPI, simplifiant le développement et la maintenance.

6

Fournissant une infrastructure robuste et souple, RabbitMQ constitue une base cruciale pour le déploiement d'architectures microservices évolutives.

Raisons de l'utilisation de FastAPI

1

FastAPI offre une performance élevée grâce à sa gestion asynchrone, idéale pour les environnements à charge élevée.

2

Syntaxe déclarative facilitant le développement rapide d'API, avec saisie automatique et génération automatique de documentation Swagger.

3

Utilisation de Pydantic pour une saisie automatique et une validation des données, garantissant la cohérence et la sécurité des entrées utilisateur.

4

Intégration native de l'asynchronisme, permettant la gestion efficace des opérations IO-bound pour une meilleure réactivité.

5

Facilité d'intégration avec différents systèmes de bases de données, notamment SQLAlchemy, pour une persistance de données robuste.

6

Fonctionnalités de sécurité incluant la gestion automatique des en-têtes CORS, la validation et la génération de jetons JWT, assurant un environnement sécurisé pour les API.

Avantage de RabbitMQ avec FastAPI

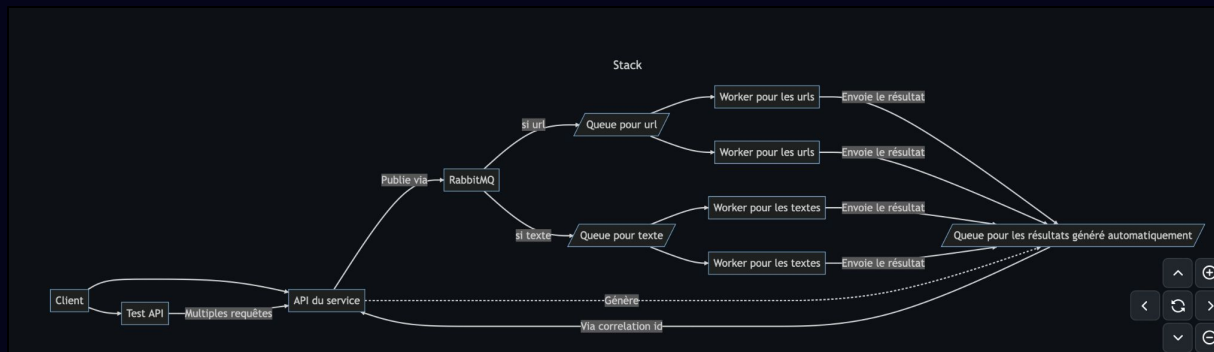
L'intégration de RabbitMQ avec FastAPI permet une communication asynchrone efficace entre microservices, améliorant la réactivité globale du système.

RabbitMQ facilite le découplage des composants, renforçant la fiabilité de la communication entre les microservices. Les messages mis en file d'attente garantissent une livraison fiable, même en cas de dysfonctionnement temporaire.

L'intégration fluide de RabbitMQ avec FastAPI simplifie le développement et la maintenance d'API, offrant une solution robuste pour la gestion des messages et la coordination entre les services.

Cas d'utilisation et Bonnes Pratiques

Nous illustrerons comment cette architecture a été utilisée pour développer un système de résumés de textes, mettant en avant la scalabilité et la flexibilité offertes par les microservices.




https://github.com/neoseb59/CAL_C_final_project


Bonnes Pratiques



Utiliser des noms de conteneurs et de services descriptifs pour une gestion facile



Définir des limites de ressources pour les conteneurs afin d'assurer une bonne isolation et des performances stables



Implémenter des tests de santé pour les microservices, pour permettre une gestion proactive des incidents

Conclusion

- Docker, RabbitMQ et FastAPI offrent une combinaison puissante pour l'architecture de microservices.
- L'adoption de ces technologies est une étape importante vers un développement et un déploiement agiles.
- La scalabilité, la flexibilité et l'efficacité de cette pile technologique en font un choix judicieux pour les applications modernes.
- Les avantages de Docker, RabbitMQ et FastAPI sont clairement démontrés dans des scénarios réels et offrent des solutions concrètes.