Rapport - Quote-API
JEE/Spring II

Maëlys Bühler

27 mai 2024

Table des matières

Chapitre 1

Introduction

1.1 Contexte

L'architecture des microservices a pris de l'ampleur des dernières années, grâce à sa capacité à améliorer la scalabilité, la résilience et la flexibilité des systèmes complexes.

Le framework Spring, avec ces différents modules, permet de mettre en place des microservices, et c'est avec cet outil que le projet que ce rapport documente a été implémenté. Ce projet a été développé dans le cadre du cours JEE/Spring II.

1.2 Description des objectifs de l'application

L'application implémentée s'appelle Quote-API. C'est une API qui permet d'ajouter des citations, et d'en obtenir selon plusieurs critères différents. Elle est séparée en 2 services : Simple-Quote et Ponctual-Quote.

Le service Simple-Quote permet aux utilisateurs de créer des citations et des auteurs, ainsi que d'obtenir tous les auteurs enregistrés, un auteur en particulier, et une citation aléatoire.

Le service PonctualQuote permet d'obtenir une citation qui change toutes les heures, une citation d'une playlist, qui va changer lorsqu'un utilisateur en fait la requête, ainsi que la dernière citation ajoutée et le dernier auteur ajouté.

Chapitre 2

Conception

2.1 L'application

2.1.1 Les services

L'application consiste en deux services qui communiquent l'un avec l'autre pour fonctionner au complet. Le schéma en figure 2.1 représente les différentes communications entre les services et avec les clients.

Deux communications synchrones ont lieu entre les deux services. Ceux deux communications sont :
- chaque heure, Ponctual-Quote envoie une requête à Simple-Quote pour lui demander une nouvelle citation, et Simple-Quote lui en renvoie une choisie aléatoirement.
- lorsqu'un utilisateur accède à l'endpoint pour passer à la prochaine citation de la playlist, Ponctual-Quote envoie une requête à Simple-Quote pour lui demander une nouvelle citation, et Simple-Quote lui en renvoie une.

Deux communications asynchrones ont lieu :
- lorsque Simple-Quote reçoit d'un utilisateur un nouvel auteur, il l'envoie à Ponctual-Quote.
- lorsque Simple-Quote reçoit d'un utilisateur une nouvelle citation, il l'envoie à Ponctual-Quote

Deux autres services font partie de l'architecture, ce sont les services de Service Discovery, qui permettent d'obtenir dynamiquement les adresses des services de l'application ainsi que l'API Gateway, qui permet d'accéder aux endpoints des deux services REST à la même adresse.

Un schéma illustrant les interactions entre les services peut être observé en figure 2.2

2.1.2 Simple Quote

Le service Simple-Quote est le service de gestion des données. C'est vers ce service que les requêtes pour la création de nouveaux éléments vont être faites.

Base de donnée

La base de données de Simple-Quote contient des auteurs et des citations. Chaque citation a un auteur.

La figure 2.3 représente de diagramme de base de données de Simple-Quote

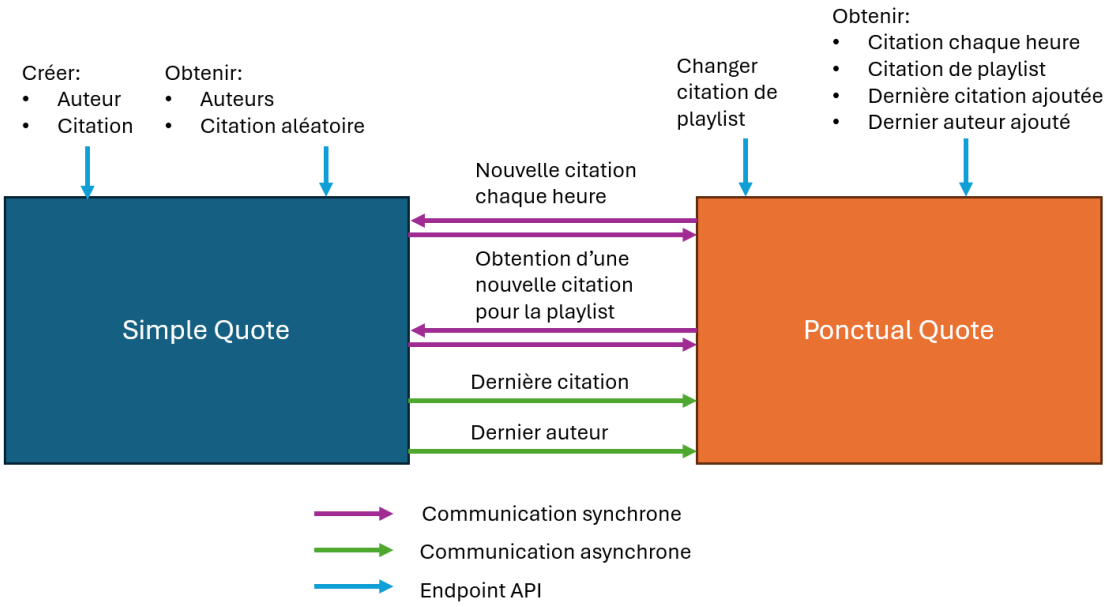

FIGURE 2.1 – Schéma des services et des communications entre eux

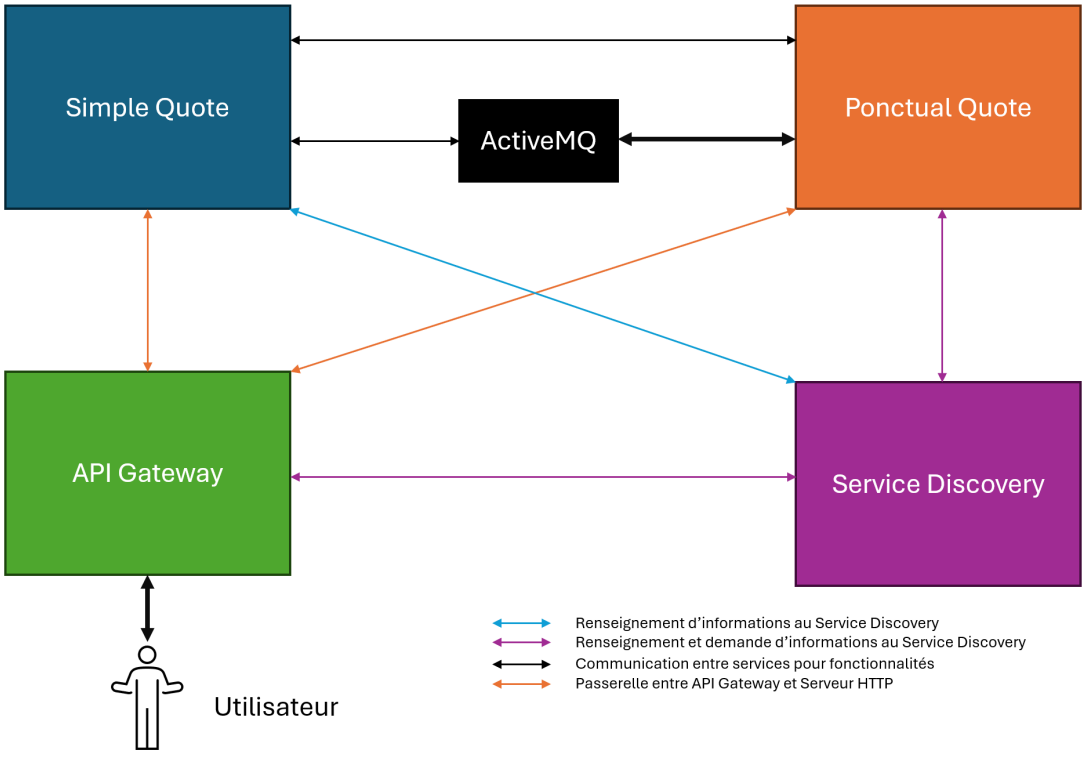

FIGURE 2.2 – Schéma du système global

FIGURE 2.3 – Diagramme de base de donnée de Simple-Quote

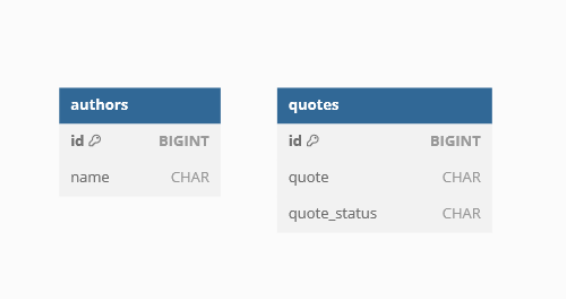

FIGURE 2.4 – Diagramme de base de donnée de Ponctual-Quote

2.1.3 Ponctual Quote

Le service Ponctual-Quote est le service permettant d'obtenir des citations de manières différentes. Avec une playlist commune à tous les utilisateurs, comme une sorte de radio, ou tous les utilisateurs ont la possibilité de passer la citation actuelle pour en voir une autre. Il y a aussi la possibilité d'avoir une citation qui change chaque heure, ce qui peut être utile pour l'afficher sur une page d'accueil de navigateur par exemple.

Base de donnée

La base de données de Ponctual-Quote contient des auteurs et des citations.

La figure 2.4 représente de diagramme de base de données de Ponctual-Quote

Chapitre 3

Implémentation

3.1 Services WEB HTTP

Les deux services WEB HTTP ont été programmés à l'aide du Framework Spring Boot.

3.1.1 Simple-Quote

Un schéma de l'architecture de Simple Quote peut être observé en figure 3.1

3.1.2 Ponctual-Quote

Un schéma de l'architecture de Ponctual Quote peut être observé en figure 3.2

3.1.3 Swagger

Les deux API ont la librairie Swagger permettant d'accéder à une documentation générée automatique aux endpoints :

```
/swagger-ui/index.html
/v3/api-docs
```

3.2 Communication asynchrone

La communication asynchrone est utilisée pour deux cas de figure dans ce projet. Il permet à Simple Quote d'envoyer les nouvelles données à Ponctual Quote pour les citations et les auteurs. On utilise pour cela des des queues MQ, appelée *lastadded.quote.notif.q* et *lastadded.author.notif.q*. Les données sont envoyées sous la forme d'objets JSON, sérialisés depuis des DTO.

3.3 Communication synchrone

La communication synchrone est utilisée pour deux cas de figure dans ce projet. Il permet à Ponctual Quote de faire une demande pour une nouvelle citation de chaque heure, et pour demander une nouvelle citation pour la playlist. Dans les deux cas, il attend une réponse de Simple Quote pour continuer son traitement, c'est ce qui rend ces communications synchrones.

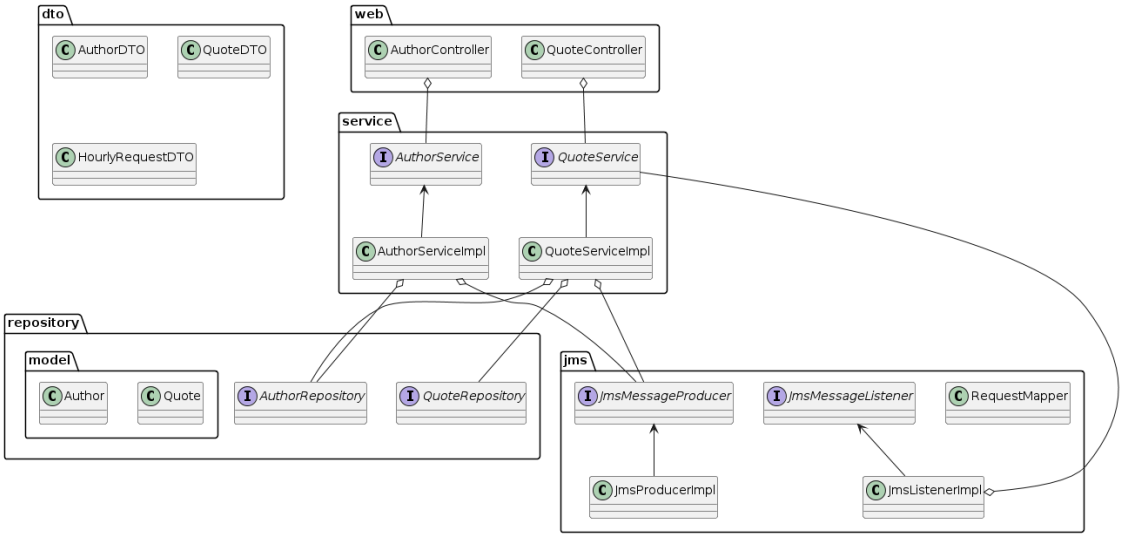

FIGURE 3.1 – Schéma de l'architecture de l'élément Simple Quote

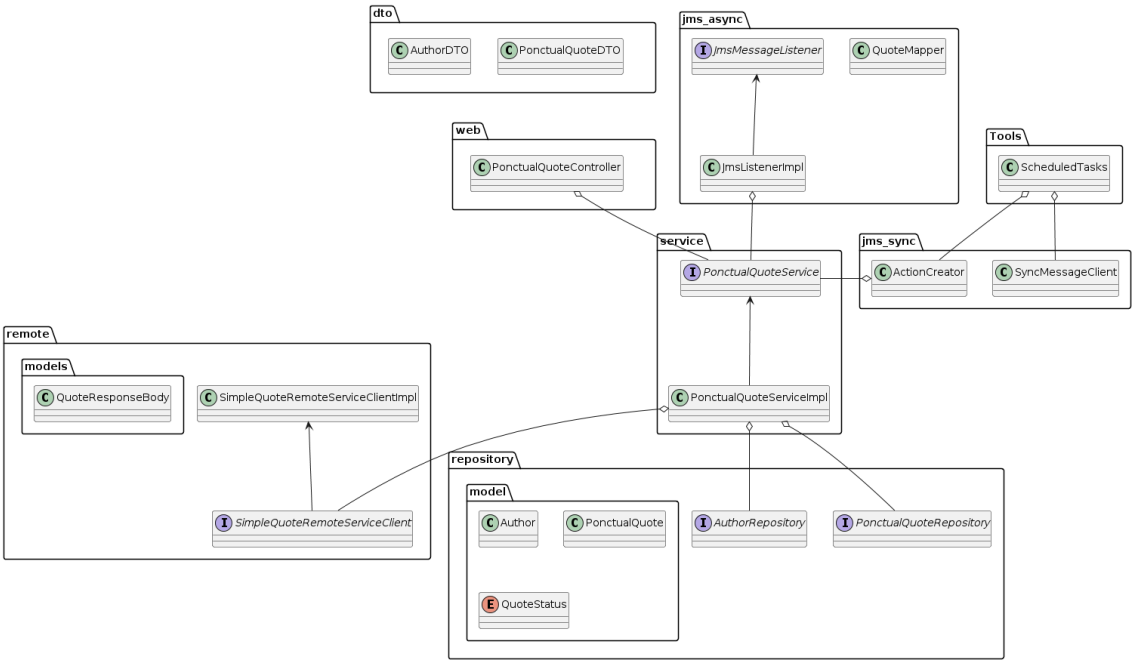

FIGURE 3.2 – Schéma de l'architecture de l'élément Ponctual Quote

Afin de tester les différentes technologies, la communication pour une citation chaque heure est faite avec JMS (et ActiveMQ), et celle pour la playlist est faite avec HTTP.

L'objectif initial était de faire les deux communications avec JMS, mais étant donné qu'un service registry a aussi été implémenté, il était aussi souhaitable de le tester dans ce contexte, et le choix d'utiliser les deux technologies est donc purement académique.

3.4 Service Registry

Le Service Registry est un composant Spring permettant aux autres composants de s'inscrire à un annuaire d'adresse, et de demander l'adresse d'un autre composant. Afin de l'utiliser, les composants doivent avoir une dépendance installée. Ce service est utilisé par Ponctual-Quote pour obtenir l'adresse de Simple-Quote afin de lui envoyer des requêtes HTTP. Il est aussi utilisé par le API Gateway pour obtenir les adresses des deux services Simple-Quote et Ponctual-Quote.

3.5 Server Side Load Balancing

Le Serveur Side Load Balancing, aussi appelé API Gateway dans ce document, permet aux utilisateurs du service de faire toutes les requêtes vers le même serveur, l'API Gateway. Celui-ci va ensuite transmettre les requêtes vers les bons serveurs HTTP.

Il a fallu pour cela le configurer, et lui indiquer les bonnes adresses. Dans le cas de ce projet, les requêtes ayant comme destinataire le service Simple Quote vont être redirigées vers lui depuis le endpoint */simple-quote/* de l'API Gateway. Les requêtes ayant comme destinataire le service Ponctual Quote vont être redirigées vers lui depuis le endpoint */ponctual-quote/* de l'API Gateway.

Chapitre 4

Installation

4.1 Local

Pour le lancement en local, il est nécessaire d'avoir mis en place un serveru MQ et une base de donnée MySQL. Pour simplifier cette mise en place, un fichier *compose-mq.yaml* permet de lancer ces deux dépendances en local à l'aide de container Docker. Docker doit donc être déjà installé sur la machine. Pour le lancer, il suffit donc d'utiliser la commande :

```
docker compose -f compose-mq.yaml up -d
```

au niveau du fichier. On peut ensuite commencer à build et lancer les services. L'ordre conseillé est :

1. service-discovery
2. api-gateway
3. simple-quote
4. ponctual-quote

Il est nécessaire d'attendre que l'élément précédent ait été build **et** lancé pour lancer le build suivante. La commande pour build est :

```
mvn clean install
```

et celle pour run est :

```
mvn spring-boot:run
```

Il faut lancer ces commandes à l'intérieur des dossiers portant le nom du service à lancer, au niveau du fichier *pom.xml* de chaque service.

4.2 Docker

Pour éviter d'avoir à faire toutes les étapes précédentes, un fichier *docker-compose.yaml* lançant l'intégralité des services est mis en place. Pour le lancer, on peut utiliser la commande :

```
docker compose up -d
```

au niveau du fichier. Étant donné qu'un service attend que l'autre ait fini de se lancer pour se lancer à son tour, il est possible que le lancement prenne du temps. Lors des tests, le lancement

intégral prenait environ 1 minute. Il est aussi possible que l'application ne soit pas immédiatement fonctionnelle après la fin du lancement de Docker, car les services peuvent mettre un peu de temps pour s'inscrire vers le service registry, ce qui bloque les différentes dépendances.

4.3 Tests

Afin de simplifier le tests de l'application, une collection et un environnement Postman à utiliser ensemble sont fournis dans la forge Github, dans le dossier *Postman*.

De plus, la citation à l'endpoint */hourly* de Ponctual Quote devrait se mettre à jour toutes les heures, mais pour en simplifier l'utilisation, elle se met à jour toutes les 10 secondes.

Chapitre 5

Conclusion

5.1 Résultats

L'objectif de ce projet étant de se concentrer sur la partie communication et microservice, les fonctionnalités du projet ne sont pas forcément très pertinentes, même si une utilisation réelle peut être imaginée. Cependant, ce projet implémente des communications synchrone et asynchrone, en particulier la communication synchrone avec ActiveMQ. L'utilisation de JMS pour une communication synchrone n'est pas standard, et dans un cas réel, l'utilisation de HTTP uniquement serait probablement recommandable, cependant, pour ce cas académique, l'exercice semblait intéressant.

5.2 Améliorations possibles

Comme relevé dans la section précédente, les fonctionnalités de l'application ne sont pas forcément toutes pertinentes ou logiques. Si une reprise du projet était envisagée, une réimagination des cas d'utilisation serait à faire, afin d'ensuite adapter l'application pour ces cas.

D'un point de vue microservices, il pourrait être intéressant d'ajouter d'autres fonctionnalités, comme l'agrégation des logs ou des métriques.