Projet API

Réalisé par PAUL MOLIN

INTRODUCTION

Ce document relate les différentes étapes qui ont amené à la réalisation du projet. Collectivement, nous étudierons les besoins qui ont été formulés et nous dévoilerons la conception, l'architecture et la réalisation. Pour finir, nous analyserons les problèmes qui ont été rencontrés et les axes d'amélioration.

* GUIDE DE DÉMARRAGE *

1. Démarrer le service d'enregistrement et de découverte

```
cd api-molin-paul-registration
mvn spring-boot:run
```

2. Démarrer plusieurs instance des autres services pour profiter du *load-balancing*

```
cd api-molin-paul-gateway
SERVER_PORT=8080 mvn spring-boot:run

cd api-molin-paul-users
SERVER_PORT=8082 mvn spring-boot:run
SERVER_PORT=8083 mvn spring-boot:run

cd api-molin-paul-courses
SERVER_PORT=8084 mvn spring-boot:run
SERVER_PORT=8085 mvn spring-boot:run
```

EXPRESSION DU BESOIN

Dans le but de cerner le besoin et d'avoir une idée claire des fonctionnalités à développer les acteurs du système ont été établis :

- Les administrateurs
- Les utilisateurs

Puis la liste des *backlogs products* par acteur a été réalisée :

En tant qu'utilisateur je veux pouvoir...

- M'inscrire
- Ajouter un moyen de paiement
- Rechercher des cours
- Acheter un cours
- Visionner les cours et les épisodes achetés

En tant qu'administrateur je veux pouvoir...

- Ajouter des cours et des épisodes
- Gérer les utilisateurs

CONCEPTION

La structure de micros services métiers a été découpée de manière à respecter les domaines fonctionnels. Le choix a été fait de créer un service dédié aux utilisateurs et un dédié aux cours et aux épisodes. Chaque service possède sa propre base de données.

Ci-dessous les diagrammes entité-association du service utilisateur puis du service cours.

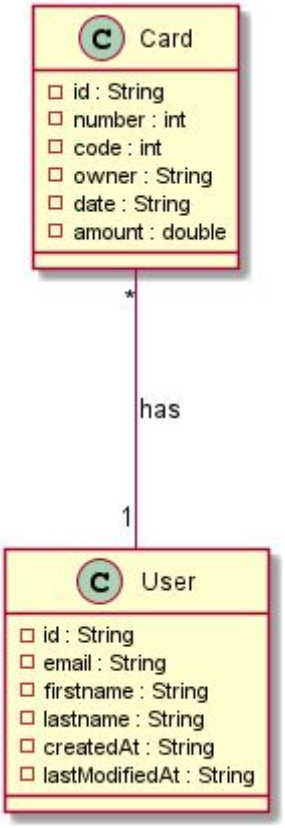

Figure 1. Diagramme entité-association du service utilisateur.

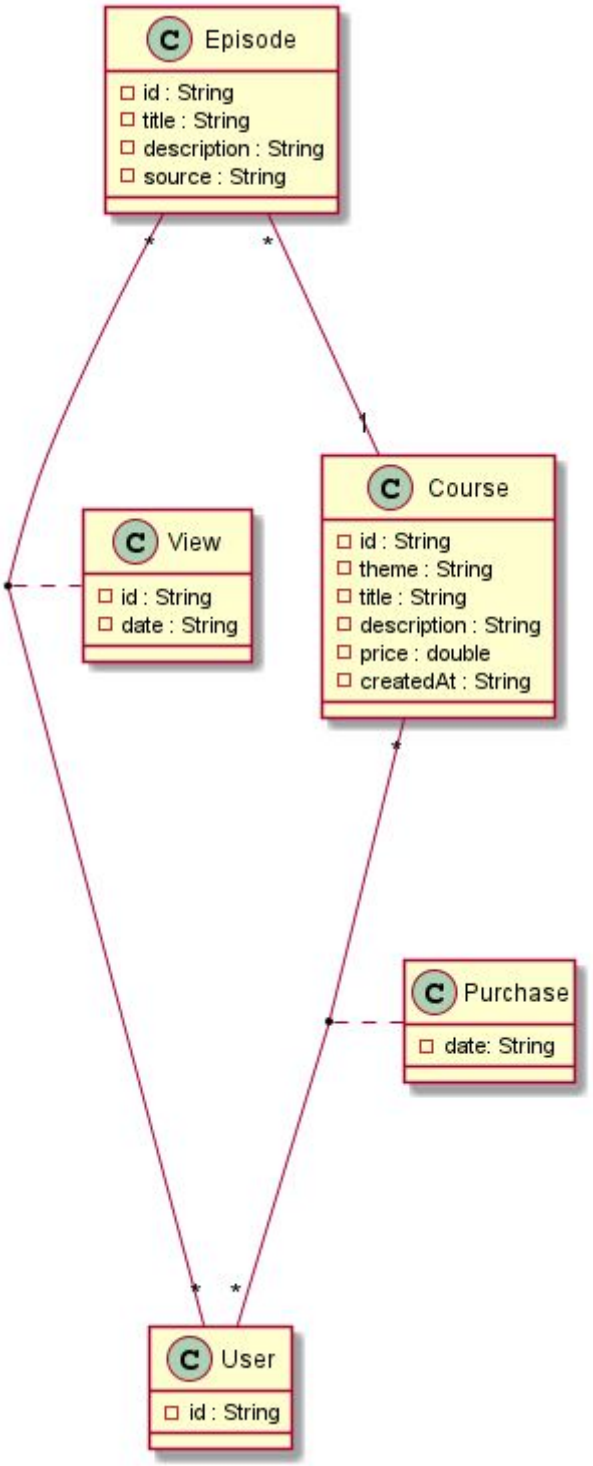

Figure 2. Diagramme entité-association du service cours.

ARCHITECTURE

Le projet est constitué de quatre services différents :

- Une passerelle *(gateway)*
- Un service d'enregistrement et de découverte *(registration & discovery)*
- Un service utilisateur
- Un service cours

L'architecture peut être résumée avec le diagramme ci-dessous.

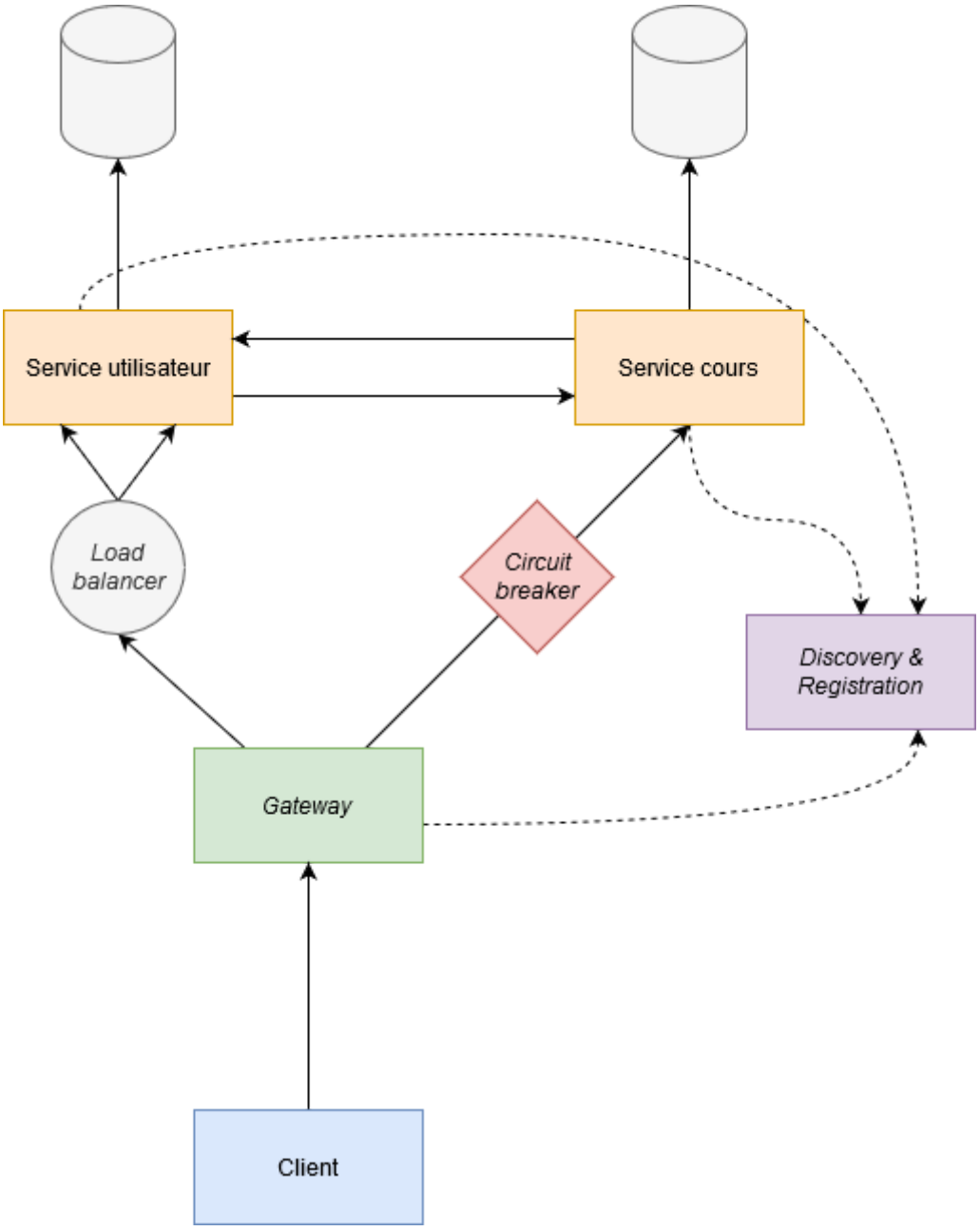

Figure 3. Schéma de l'architecture du système.

PROBLÈMES RENCONTRÉS ET AXES D'AMÉLIORATION

J'ai rencontré plusieurs problèmes lors de la réalisation du projet. Le premier problème était logique et conceptuel, j'avais un peu de mal à me représenter comment les services pouvaient gérer la persistance des données sans avoir trop de redondance.

Deuxièmement, j'ai passé beaucoup de temps à régler un problème concernant la bibliothèque *spring gateway* qui ne réussissait pas à résoudre le DNS d'eureka. Le problème est expliqué clairement dans cette issue github qui est encore ouverte : https://github.com/spring-cloud/spring-cloud-gateway/issues/2091.

Pour les axes d'amélioration, j'ai utilisé la bibliothèque RestTemplate pour la communication entre les microservices sans utiliser de *circuit breaker*. Si un service tombe, les requêtes ne fonctionnent plus. Il serait donc intéressant d'ajouter un circuit breaker comme *resilience4j* ou *hystrix* ou d'utiliser un *message broker* comme RabbitMQ.