Project CC

Projet Web Full Stack avec Microservices

Ce projet est une application web full stack créée avec Java et React. Spring Cloud a été utilisé dans ce projet pour créer l'architecture de microservices. Des explications détaillées sur les services dans l'architecture de microservices sont fournies dans les fichiers readme des services.

Objet du Projet

L'application Cinema est un site web de vente de billets de cinéma en ligne. L'objectif de ce site est de faciliter l'achat de billets pour ceux qui souhaitent regarder des films au cinéma. Les utilisateurs peuvent consulter les films actuellement diffusés dans les cinémas ou les films à venir. Ils peuvent voir les détails du film, connaître l'intrigue, les acteurs, la date de sortie, etc. Sur cette page détaillée, les utilisateurs peuvent choisir la ville et le cinéma où ils souhaitent regarder le film. Après cette sélection, ils sont automatiquement redirigés vers la page de paiement. Sur cette page de paiement, ils peuvent choisir le nombre de billets et le type, comme étudiant ou adulte. Ensuite, ils peuvent choisir les sièges où ils vont s'asseoir dans le cinéma. Enfin, ils complètent le processus de paiement en saisissant des informations telles que les informations de carte de crédit, l'e-mail, le nom et le prénom. Si le paiement est réussi, les détails du billet sont envoyés à l'e-mail saisi par l'utilisateur. Si les gens veulent partager leurs opinions sur le film, ils peuvent écrire des commentaires sur la page de détails du film. Cependant, les utilisateurs doivent créer un compte pour commenter les films. Seuls les administrateurs peuvent ajouter des films, acteurs ou réalisateurs au système. Ce processus d'autorisation est contrôlé avec un jeton Jwt.

Technologies du Projet

Il y a de nombreuses technologies dans ce projet. Ce sont :

Technologies Backend

Java 17

Spring Boot 2.7.0

Spring Cloud

Spring Data Jpa

Spring Security

Lombok

WebClient

Apache Kafka

Jwt

Java Mail Sender

Zipkin

Resilience4j

PostgreSQL

MongoDB

Redis

Docker

Technologies Frontend

JavaScript

React

Bootstrap

Redux

Utilisation des Technologies dans le Projet

Il y a 5 services dans ce projet, et chaque service est écrit avec une architecture en N couches. Spring Cloud est utilisé pour l'infrastructure de microservices. Netflix Eureka Server est utilisé pour créer un serveur Eureka. Ce serveur Eureka contient les services de film, de l'utilisateur, et de messagerie en tant que clients Eureka et le service de passerelle API. De plus, Zipkin et Sleuth sont utilisés pour surveiller les journaux inter-services. De plus, Resilience4j est utilisé comme disjoncteur.

Dans la passerelle API, Spring Cloud Gateway est utilisé pour gérer les demandes.

Dans le serveur Eureka, Netflix Eureka Server est utilisé. Et Spring Security est utilisé pour sécuriser le serveur Eureka.

WebFlux est utilisé pour la communication entre les services Movie et User. Et Apache Kafka est utilisé pour la communication asynchrone entre les services Movie et Email.

Dans le service User, MongoDB est utilisé comme base de données. Spring Security est utilisé pour crypter les mots de passe des utilisateurs et générer des jetons Jwt.

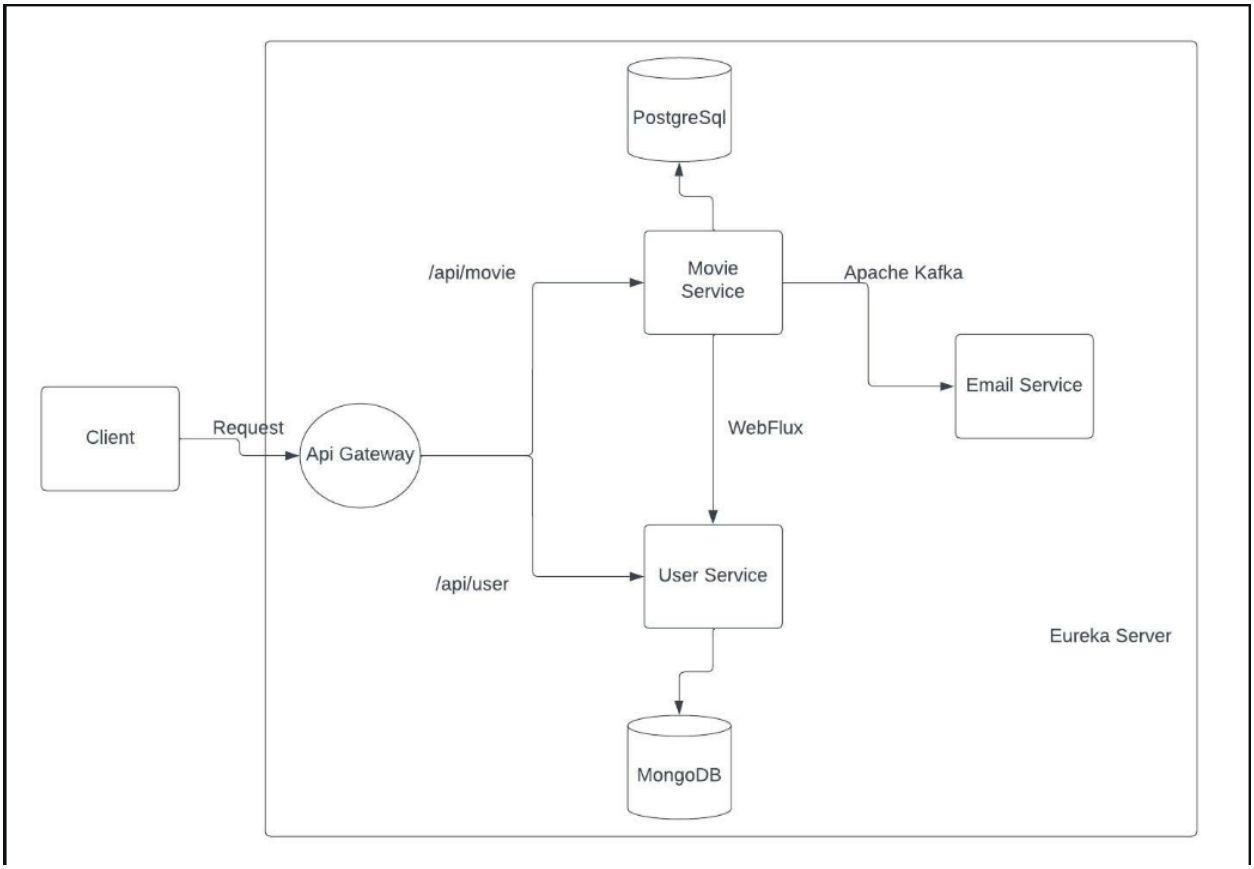
Dans le service Movie, PostgreSQL est utilisé comme base de données et Spring Data Jpa est utilisé. Webflux et Apache Kafka sont utilisés pour la communication avec les autres services. Le disjoncteur Resilience4J est utilisé ici. L'affichage des films actuellement diffusés et des films à venir est mis en cache avec Redis.

Dans le service Email, Apache Kafka est utilisé pour recevoir le message du service Movie. Java Mail Sender et le modèle FreeMarker sont utilisés pour créer un modèle d'e-mail et envoyer un e-mail.

PostgreSQL, MongoDB, Apache Kafka et Zipkin s'exécutent en tant que conteneurs Docker dans le fichier docker-compose.yml.

Du côté frontend, JavaScript et React ont été utilisés. De plus, Axios a été préféré pour envoyer des requêtes au backend. Pour la gestion de l'état, Redux a été utilisé. Pour la conception de l'interface utilisateur, Bootstrap et Css sont utilisés.

Architecture utilisé :



How to run :
Téléchargez le code source du projet. Ouvrez ce projet avec votre IDE préféré. Assurez-vous que Java 17, Node.js et Docker sont installés sur votre ordinateur. Suivez ces étapes :

Exécutez le fichier docker-compose.yml

Ce fichier docker-compose est nécessaire pour exécuter PostgreSQL, MongoDB, Kafka, etc. Ouvrez le terminal dans le répertoire du projet et saisissez la commande :

docker compose up -d

pour lancer les conteneurs.

Exécutez le serveur Eureka

Allez dans la classe EurekaServerApplication qui se trouve dans le module eureka-server et exécutez cette classe pour créer le serveur Eureka. Si vous souhaitez afficher le panneau du serveur Eureka, vous pouvez vous rendre aux adresses localhost:8080/eureka/web ou localhost:8761. Ensuite, saisissez le nom d'utilisateur = eureka et le mot de passe = password.

Exécutez la passerelle API

Pour rediriger les demandes vers les services pertinents, la passerelle doit être exécutée. Allez dans la classe ApiGatewayApplication qui se trouve dans le module api-gateway et exécutez cette classe. Si vous souhaitez vérifier que la passerelle API est enregistrée sur le serveur Eureka, vous pouvez afficher le panneau du serveur Eureka.

Exécutez le service Movie

Pour exécuter le service de film, accédez au module du service de film. Et exécutez la classe MovieServiceApplication.

Exécutez le service utilisateur

Dans le module du service utilisateur, trouvez la classe UserServiceApplication et exécutez cette classe.

Exécutez le service de messagerie électronique

La classe EmailServiceApplication se trouve dans le module du service de messagerie électronique. Pour exécuter le module du service de messagerie électronique, exécutez cette classe.

Note importante :

Modifiez les configurations de messagerie dans le fichier application.yml avec vos propres configurations.

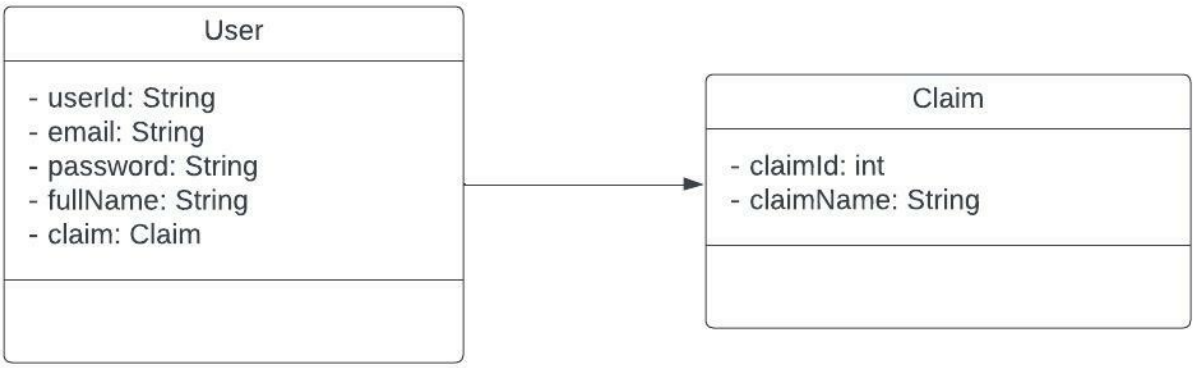
Démarrez l'application React (frontend)

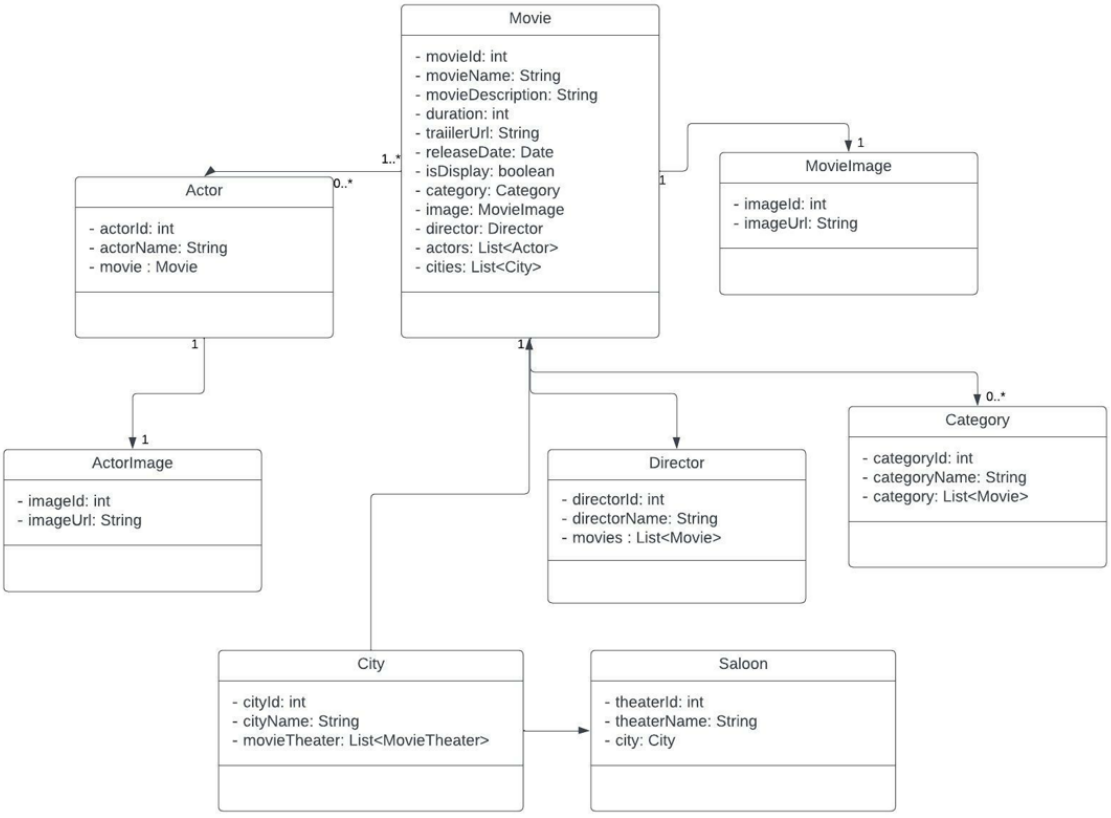
Accédez au package frontend qui est l'emplacement du code frontend. Tout d'abord, saisissez la commande :
npm install

dans le terminal pour télécharger les dépendances du package.json. Ensuite, pour démarrer l'application React, saisissez la commande :

npm start

Après cela, npm démarrera votre application React sur localhost:3000.





github :
https://github.com/soulabdr/cinema.git

Jenkins script :

```
pipeline {
   agent any
   tools {
      maven 'maven'
   }

   stages {
      stage('Git Clone') {
         steps {
            script {
```

```
                checkout([$class: 'GitSCM', branches: [[name: 'main']], userRemoteConfigs: [[url:
'https://github.com/soulabdr/cinema.git']]])
            }
        }
    }

    stage('Build') {
        steps {

                bat 'mvn clean install'

        }
    }

    stage('Create Docker Image') {
        steps {
            script {

                bat 'docker build -t lachgar/pos .'

        }
    }
```

```
        stage('Run') {
            steps {
                script {


                    bat "docker run --name test-pos -d -p 8585:8282 soulacine/pos"
                }
            }
        }
    }
}
```