Projet Architecture Microservices

Rapport de Projet

Gestion d'emprunts de livres

Auteurs :

DIARSO Laye Kemo

NGUYEN Michel

Enseignant :

MENCEUR Mouloud

M2 MIAGE IF APP

Historique des versions			
Version	Date	Auteurs	Modifications
1.0	17/05/2020	Laye, Michel	Création de la première version du document
1.1	30/05/2020	Laye, Michel	Mise en évidence des digrammes de classe et compléter le document

Indications pour exécuter le projet

Docker, MiniKube, Github, Readme.txt

I- Documentation technique

Schéma d'architecture

Nous proposons une application back-end, donc la partie du code exécutée par le serveur. Elle se décompose en un serveur (pour héberger l'application), une application (d'administration du serveur), et une base de données pour organiser les données.

L'architecture Microservices propose une solution simple et intuitive : le découpage d'une application en petits services, appelés Microservices, parfaitement autonomes qui exposent une API REST que les autres Microservices pourront consommer.

Nous avons décidé de choisir cette architecture permet de proposer des solutions en ligne qui sont toujours disponibles, une décomposition de services indépendants entre eux, permettant d'optimiser les ressources consacrées à la maintenance et au développement. Le déploiement continu devient possible car il n'y a pas besoin de coordination au niveau des modifications locales.

Enfin, pour assurer la persistance des données, nous utiliserons l'interface de programmation JPA (Java Persistance API). Cela nous facilite grandement la tâche : il sera plus facile d'ajouter, modifier ou supprimer des données grâce aux gestionnaires d'entités.

Nous avons aussi envisagé de mettre en place une service client (un frontend) qui assurera la liaison et communication avec les différents services.

Choix techniques

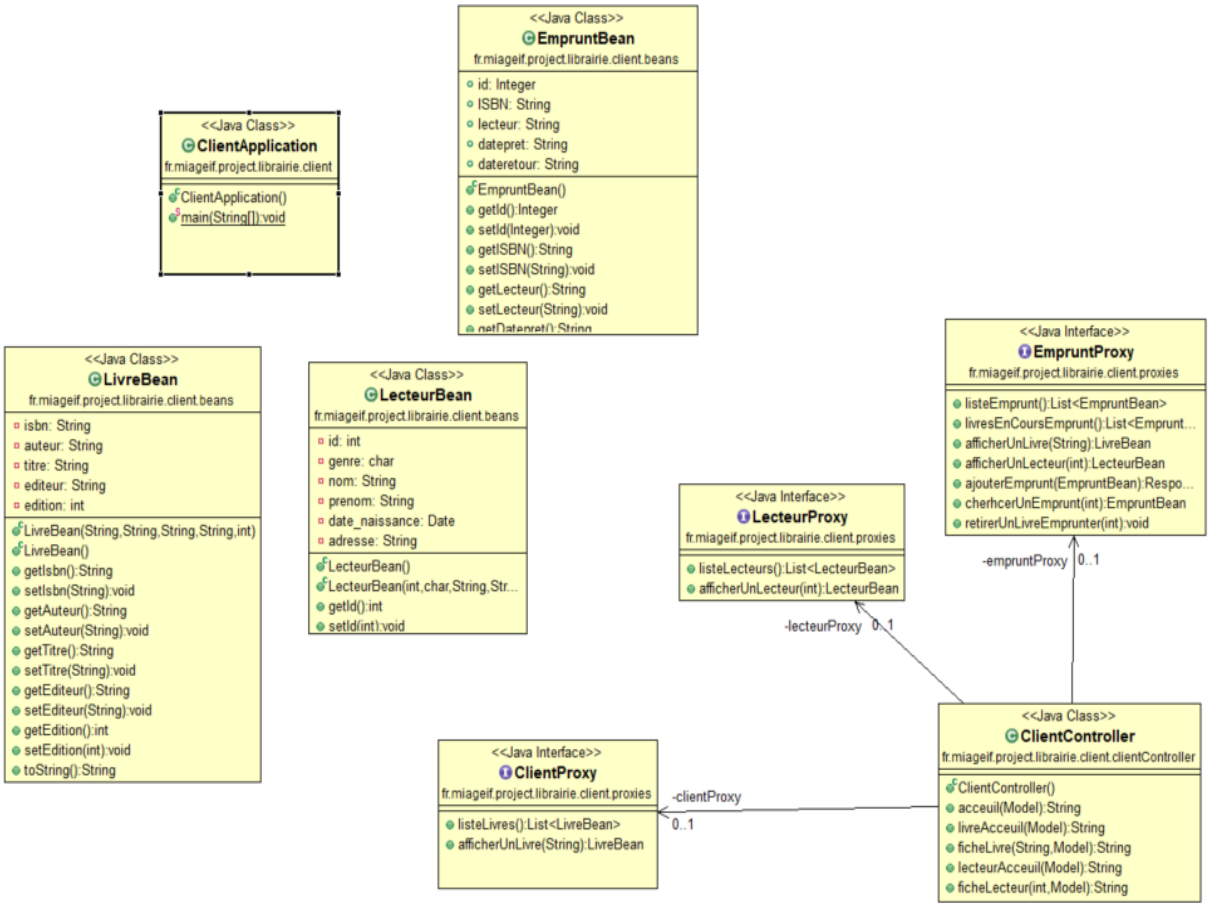
Nous avons choisi les technologies suivantes :

- IntelliJ idea en tant qu'IDE pour développer en Java pour sa stabilité, sa fonctionnalité d'auto-complétions, la possibilité de refactorer, tester, analyser notre code en temps réel.
- Spring Boot pour simplifier l'auto-configuration du projet et des fichiers et son incroyable gestion des dépendances dans le fichier pom.xml (qui serait beaucoup plus long en temps normal : il faudrait utiliser spring, spring mvc, jackson et tomcat)
- Docker pour conteneuriser nos microservices et les rendre disponible
- MiniKube pour utiliser la machine Docker et enfin déployer l'application
- Le développement s'est fait sur Windows

Diagramme de classes

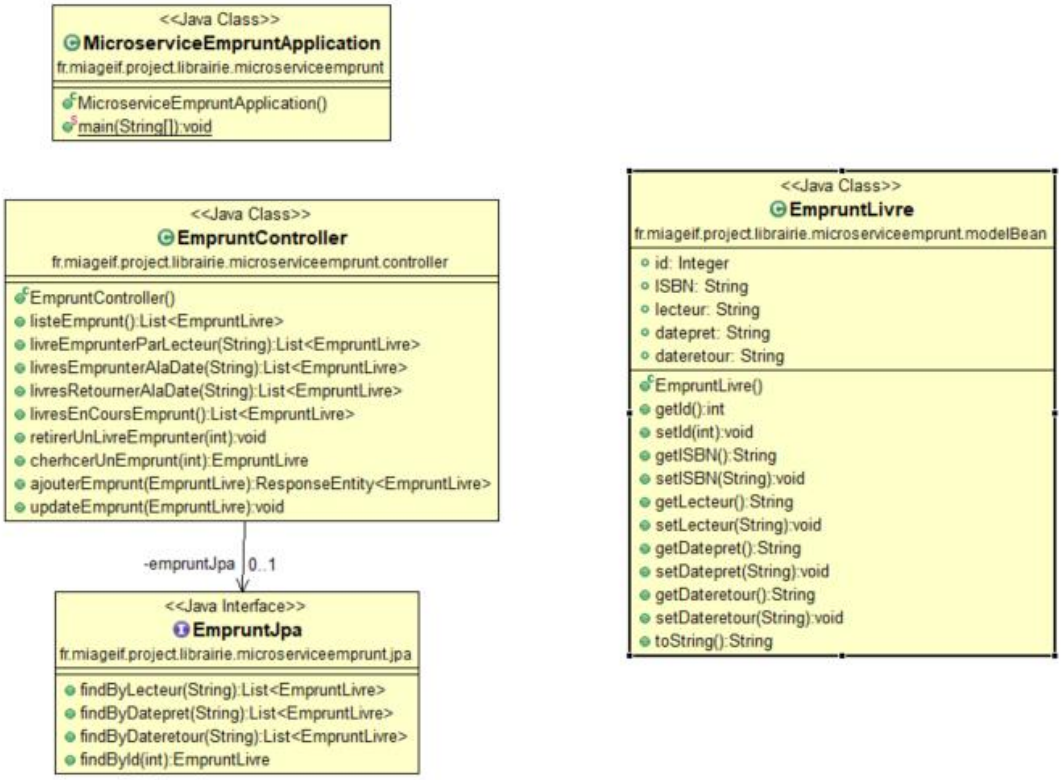
Nos microservices sont indépendants entre eux, donc aucune association entre une classe d'un Micorservice à un autre.

Diagramme de classe service Client :



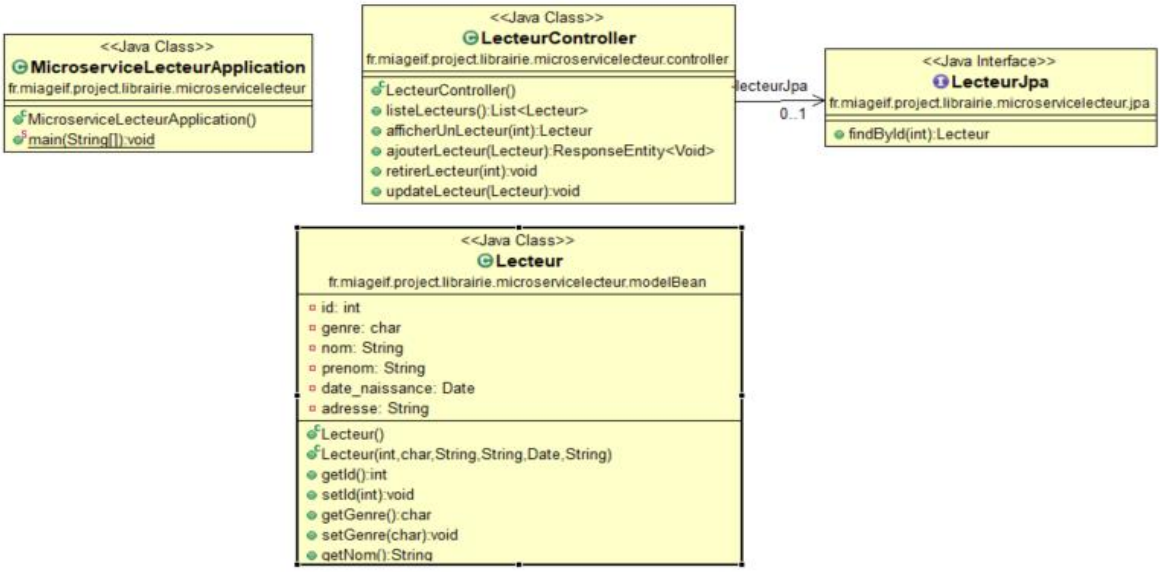
Ce service regroupe les classes permettant d'assurer la communication entre les différents microservices. Chaque classe principale des autres microservices est représentée par une classe bean.

Diagramme de classe de Microservice-emprunt



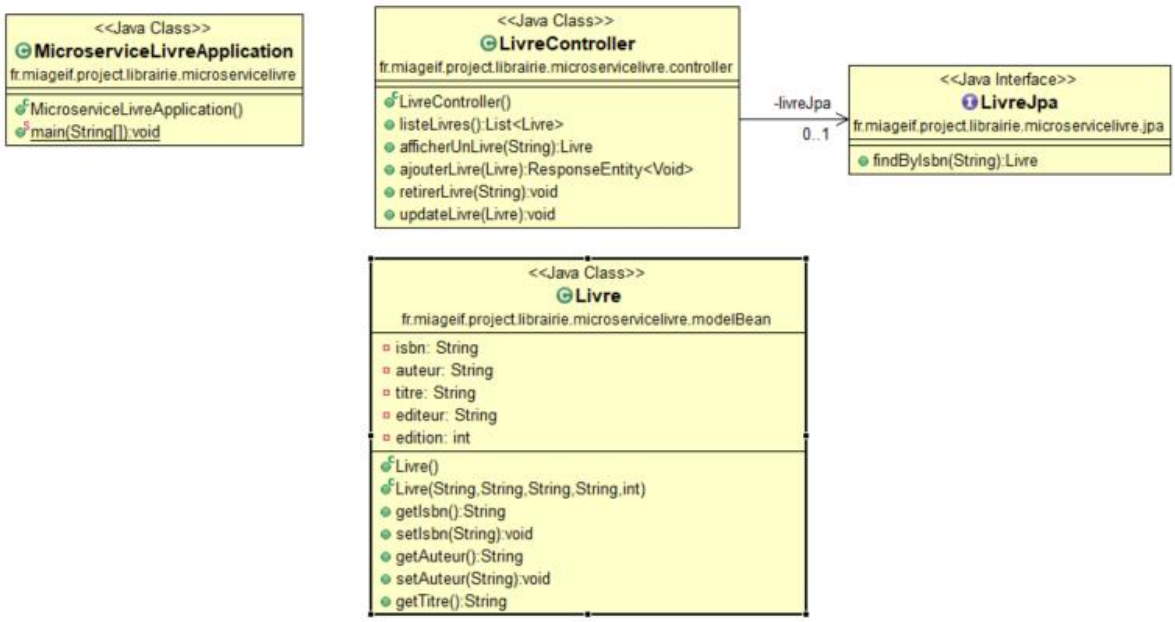
Ce diagramme regroupe les classes permettant la gestion des emprunts.

Diagramme de classe du microservice-lecteur



Ce diagramme regroupe les classes permettant la gestion des lecteurs.

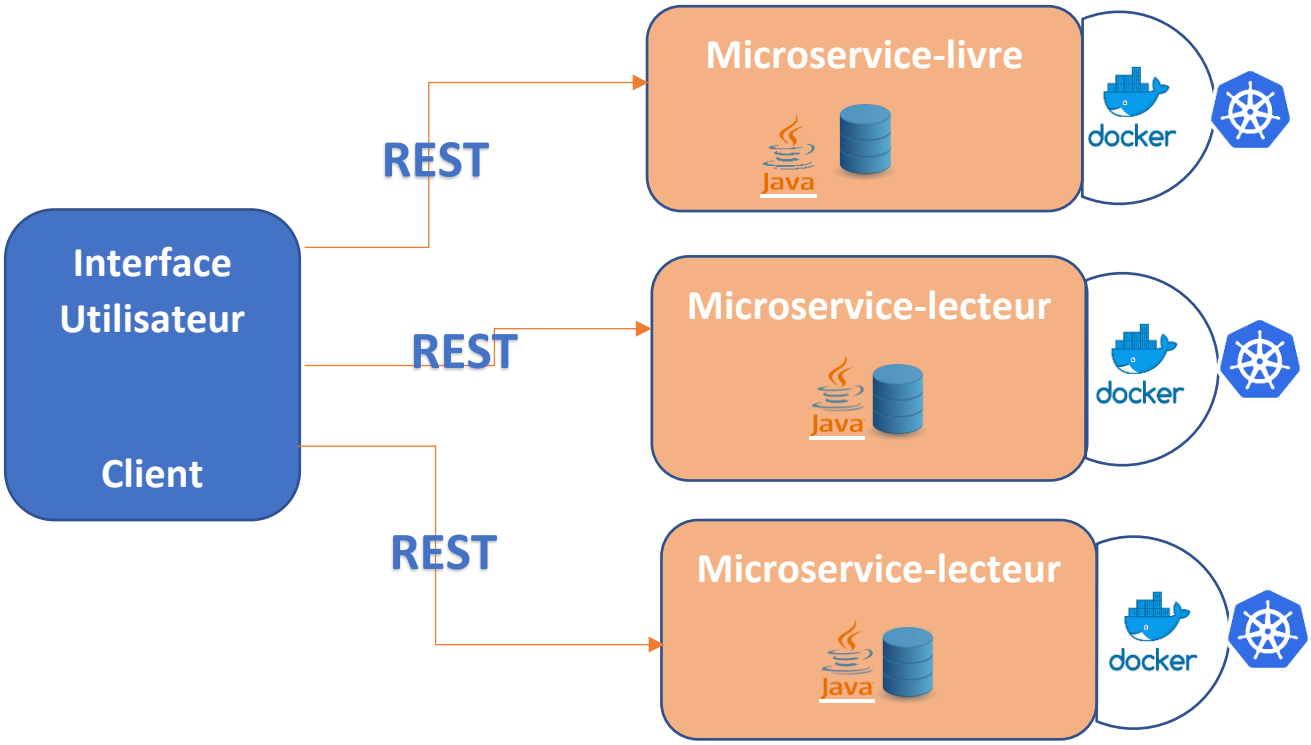
Diagramme de classe du microservice-livre



Ce diagramme regroupe les classes permettant la gestion des livres.

II- Découpage et structure des micro-services

Nous avons décidé de découper notre application en trois microservices principaux tant en rajoutant à cela un Microservices client. Ce client assure la liaison entre les différents Microservices. Il est représenté ainsi :



Ce schéma nous montre une utilisation très simplifiée de notre application Microservice. Nous utilisons le client afin d'interagir avec les autres services. Chaque service sera mis dans un conteneur docker et déployé avec minikube de façon indépendante.

III- Compilation et exécution du Projet

Pour cette partie nous fournissons un document « readme.txt » qui explique comment :

- Récupérer le projet depuis le depôt git
- L'importer dans IntelliJ
- Exécuter les différents services
- Créer les différentes images docker
- Procéder au déploiement

Le document se trouve également sur le dépôt git.

IV- Bilan

Réussites

Ce fut un projet très instructif où nous avons pu apprendre de nouvelles technologies comme docker, Spring boot et kubernates (minikube).

Nous avons réussi à mettre explorer toutes ces technologies dans notre projet. Nous avons commencé à découper et à implémenter nos Microservices (lecteur, livre et emprunt). On a également implémenté un service en plus afin de jouer le rôle de client dans la consommation des services qui est le service client.

Après l'implémentation nous sommes passé au dockerisation de nos différents services. Et la phase finale était le déploiement avec MiniKube. Chaque service de notre application contient ses propres fichiers de conteneurisation (Dockerfile) et de déploiement (les fichiers .yaml) dont la manière de procéder est décrite dans le fichier ***readme*** joint.

Nous avons pu aborder tous les points du projet, avec un peu de difficulté pour certains points compte tenu de nos supports logiciels (Système d'exploitation). Mais à la longue nous avons pu trouver des alternatives.

Difficultés

Nous avons eu plus de difficultés au début à comprendre le fonctionnement de Spring Boot et des microservices (notamment sur des détails, par exemple sur certaines annotations

ou la recherche par id). On a donc pris un peu de temps à comprendre le fonctionnement de Spring boot afin de mieux découper notre application en des microservices de façon logique.

La phase d'implémentation n'a pas été difficile, dès que nous nous sommes initiés aux annotations de spring boot tout était plus claire car nous savions déjà coder en java.

Les deux points les plus complexes ont été l'utilisation de docker et le déploiement avec minikube. Le support que nous utilisions n'était pas logiquement adapté à l'utilisation de la technologie docker.

Nous avons travaillé sur Windows, plus précisément Windows home qui n'est pas prise en charge par docker. Mais après de longues recherches nous avons pu trouver une alternative qui était d'utiliser le docker toolbox supporté par Windows Home. Finalement tout a bien marché.

Le déploiement a été un peu compliqué aussi avec l'utilisation de minikube et plus particulière kubectl sur Windows. Mais après aussi de longues recherches et de configuration de machine virtuel, nous sommes arrivés au bout et on a pu déployé nos différents microservices.

Ce que nous avons aimés / moins aimés

Aspirants à devenir développeurs, ce projet nous a beaucoup intéressé car il nous a permis d'en apprendre plus sur le développement back-end et de manière générale sur l'architecture web. Nous comprenons maintenant mieux les avantages et l'utilisation de l'architecture microservices.

L'énoncé du projet était simple à comprendre et par conséquent nous n'avions pas besoin de passer beaucoup de temps sur la compréhension du sujet : on pouvait se concentrer sur le développement.