16/04/2023

Rapport projet Architecture Microservices

Projet réalisé par :

Yufei LIU
Giovanni AMOUSSOU

Table des matières

Objectif du projet 2
Compilation et exécution de nos services 2
Schéma d'architecture 3
Choix techniques 4
Diagramme de classes 4
Bilan du projet 5
Ce que nous avons aimé dans ce projet 5
Ce que nous avons appris de ce projet 5
Ce que nous avons moins aimé dans ce projet 5
Réussites / Difficultés 5

Objectif du projet

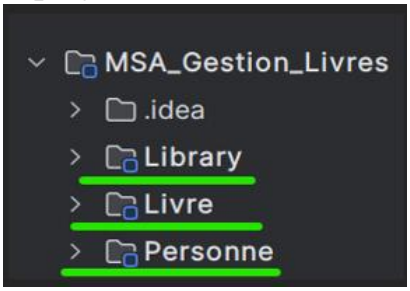
Le but de ce projet est de mettre en place une application permettant de réaliser des emprunts de livres.

Pour ce faire, nous avons réaliser ce projet avec à l'aide du Framework de SpringBoot.

SpringBoot est basé sur du JAVA donc notre projet est entière implémenté en JAVA.

Compilation et exécution de nos services

Afin de réaliser l'objectif du projet, nous avons mis en place 3 microservices :



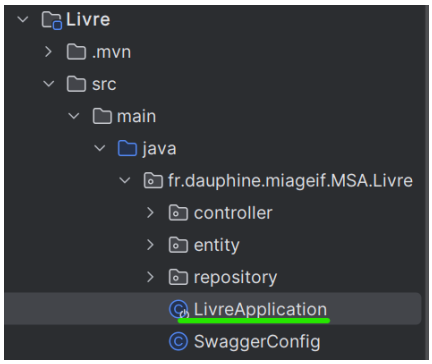
1) Se connecter sur le Github via le lien suivant : « Lien à insérer » et télécharger le projet.

2) Dézipper le projet et le copier sur dans un répertoire.

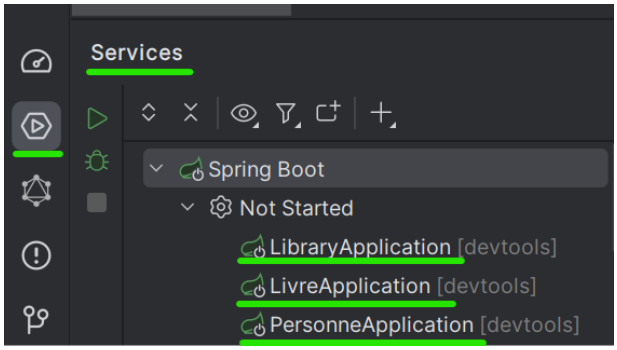
3) Lancer IntelliJ IDEA et ouvrir le projet dans notre répertoire

4) Une fois le projet synchronisé dans IntelliJ, vous avez accès aux microservices comme ci-dessus.

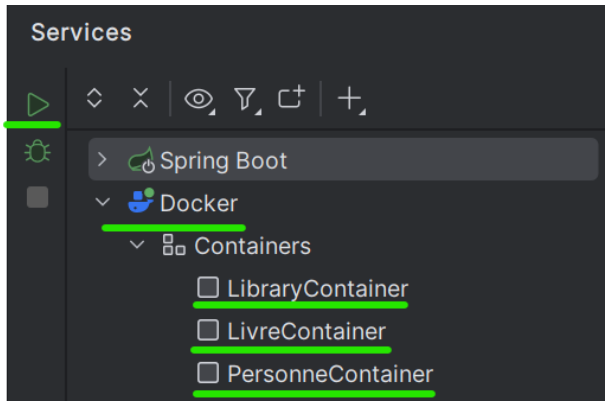
5) En déroulant le microservice *Livre* par exemple, vous aurez une architecture comme suit, faites un clic droit sur *« LivreApplication »* et cliquer sur *« Run 'LivreApplication' »*. Le microservice *Livre* est donc lancé. Il faudra en faire de même pour lancer les deux autres microservices.



6) Une autre façon plus simple, si vous avez le plugin Spring Boot d'installer dans intelliJ, vous pouvez voir tous vos microservices comme ci-dessous :



7) On peut aussi lancer notre projet en s'aidant du plugin docker dans intelliJ comme vous pouvez le voir ci-dessous :



On peut aussi lancer le projet en utilisant les lignes de commandes. Vous devez donc ouvrir un terminal, utiliser la commande « cd » pour vous mettre dans le repertoire du microservice puis lancer la commande. Voici un exemple ci-dessous pour le microservice « Livre » :

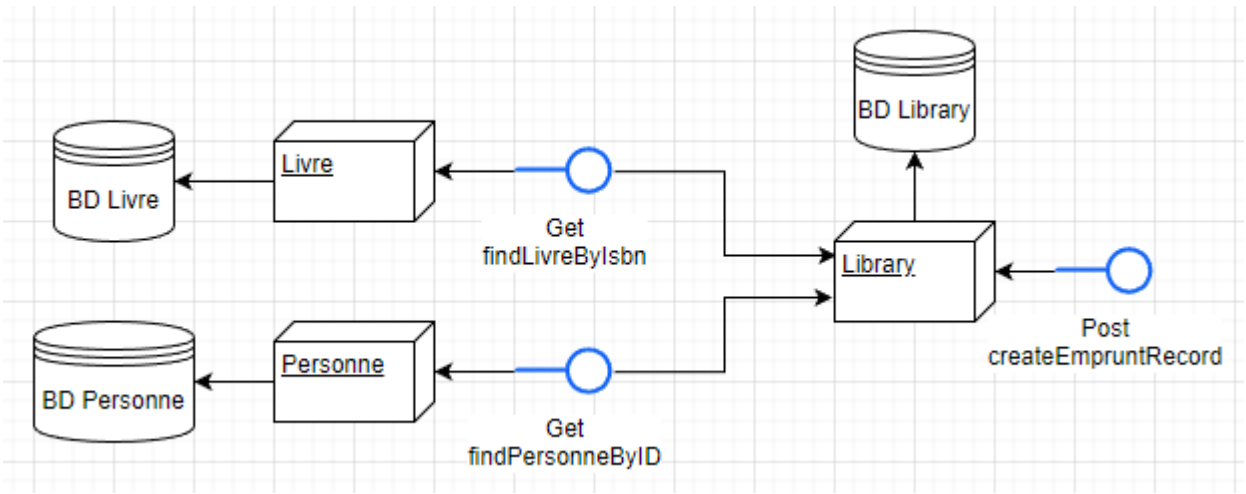
cd Livre
docker run -p 8001:8001 livre-service

```
\MSA_Gestion_Livres> cd .\Livre\
\MSA_Gestion_Livres\Livre> docker run -p 8001:8001 livre-servic
```

Schéma d'architecture

Comme nous l'avions évoqué plus haut, nous avons mis en place trois microservices : Livre, Personne et Library.

- Le microservice Livre gère le référencement de tous les livres.
- Le microservice Personne est celle qui permet de gérer tous les gens.
- Library permet donc de gérer le prêt d'un livre par une personne. Pour cela il communique avec les services : Livre et Personne en utilisant le lien API de chaque service. **Voici un schéma de l'architecture du flux de principe de notre application.**



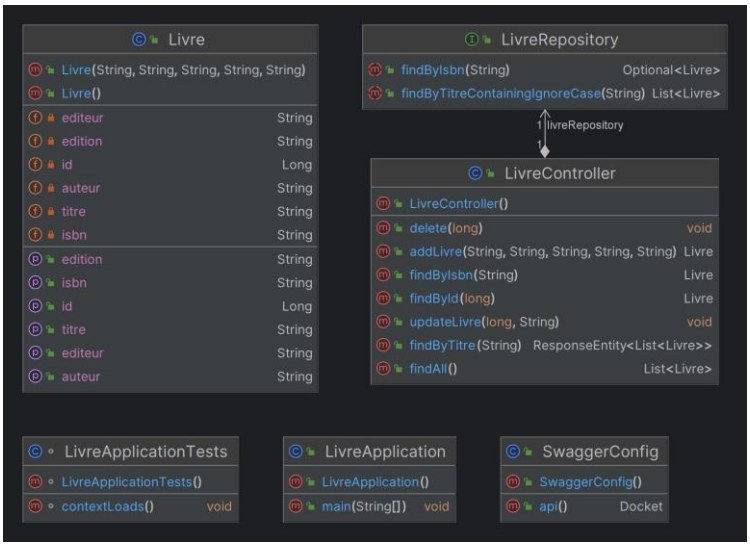
Choix techniques

Nous avons choisi d'utiliser les technologies suivantes pour notre application microservice.

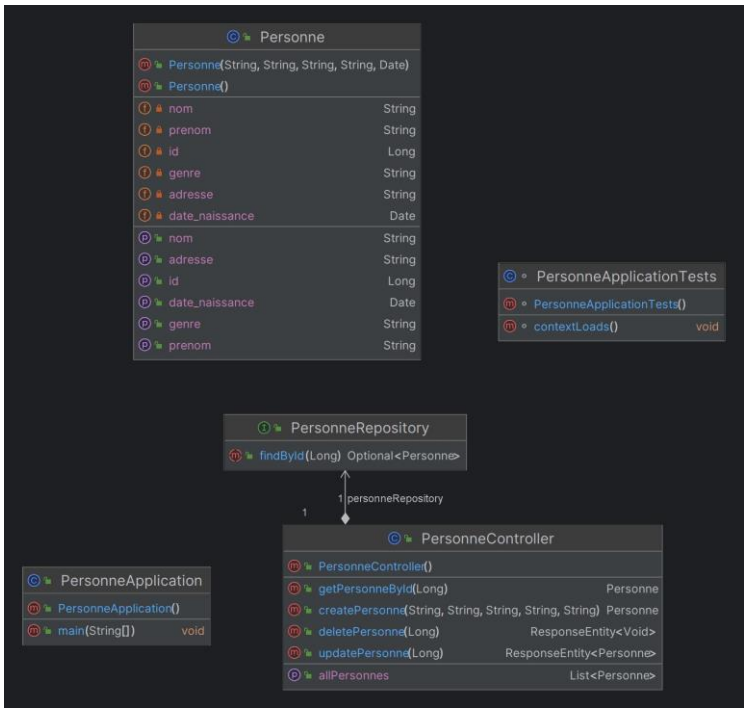
- Le principal langage de programmation est JAVA.
- Nous avons utilisé les technologies de SpringBoot pour implémenter nos microservices.
- PostgreSQL pour la couche de stockage de données, nous avons ainsi créée une BDD en ligne sur le site d'elephantsql.com. Pour chaque service on a donc une base de données.

Diagramme de classes

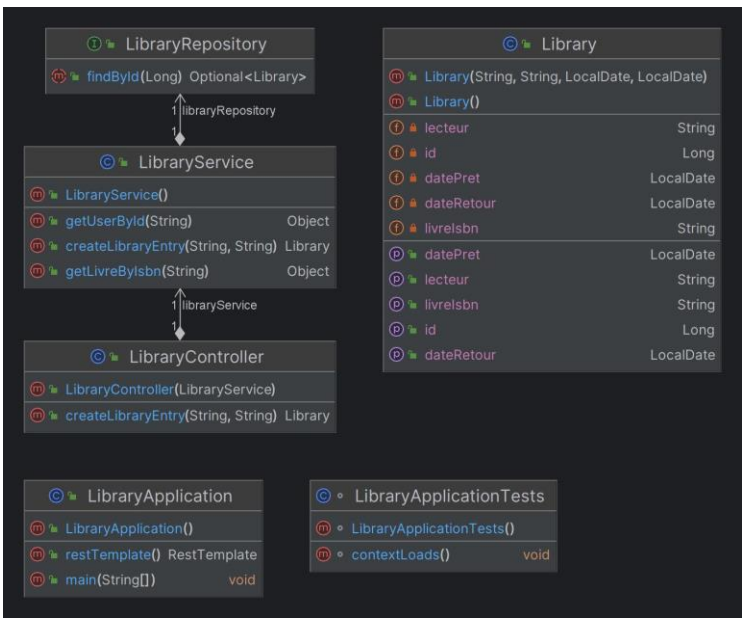
Livre



Personne



Library



Bilan du projet

Ce que nous avons aimé dans ce projet

Travailler avec le framework SpringBoot, pratique et rapide pour mettre en place les microservices.

Le fait de pouvoir créer une BDD et tester nos services.

Ce que nous avons appris de ce projet

- Comment faire interagir les microservices entre eux pour créer une application.
- Comment utiliser le framework SpringBoot pour implémenter le principe du CRUD (Create Reade Update Delete) dans nos microservices.
- Comment intégrer une base de données PostgreSQL dans une application SpringBoot.
- Comment mettre en place un docker.

Ce que nous avons moins aimé dans ce projet

La difficulté de mise en place du kubernetees

Réussites / Difficultés

Nous avons réussi à créer une application backend qui permet de gérer le prêt d'un livre par une personne. Cependant, nous avons rencontré des difficultés lors de l'intégration des services, Livre et Personne dans le service Library. Nous avons aussi eu du mal à mettre dockeriser notre application même si cette étape était optionnelle.

Malgré ces difficultés, nous avons pu surmonter ces obstacles grâce à une collaboration étroite et à une résolution de problèmes efficace en équipe.

Nous pensons qu'avec un peu plus de temps :

- Nous aurions pu mettre en place une interface qui aurait mieux interagi avec l'utilisateur.
- Nous aurions également pu utiliser swagger pour documenter nos API et standardiser nos modules de microservices afin qu'ils puissent être mieux réutilisés et déployés.
- Enfin, au lieu d'appeler les API directement dans d'autres services, il est preferable d'utiliser API Gateway, qui sert de point d'entrée unique pour toutes les API de notre écosystème de microservices.

Ceux-ci constituent nos axes principaux d'amélioration.