PROJET MICROSERVICES

Aaron LELLOUCHE
Raphaël UZAN

Table des matières

Introduction

L'objectif de ce projet est de concevoir et réaliser une application permettant d'effectuer des opérations bancaires. L'application doit permettre de :

- gérer (créer, récupérer/lister, modifier, supprimer) des comptes bancaires,

- gérer des opérations bancaires.

L'implémentation côté serveur doit être sous forme de microservices à l'aide de Spring Boot.

De plus, la conteneurisation via Docker sera abordée et mise en place dans ce projet.

Documentation technique

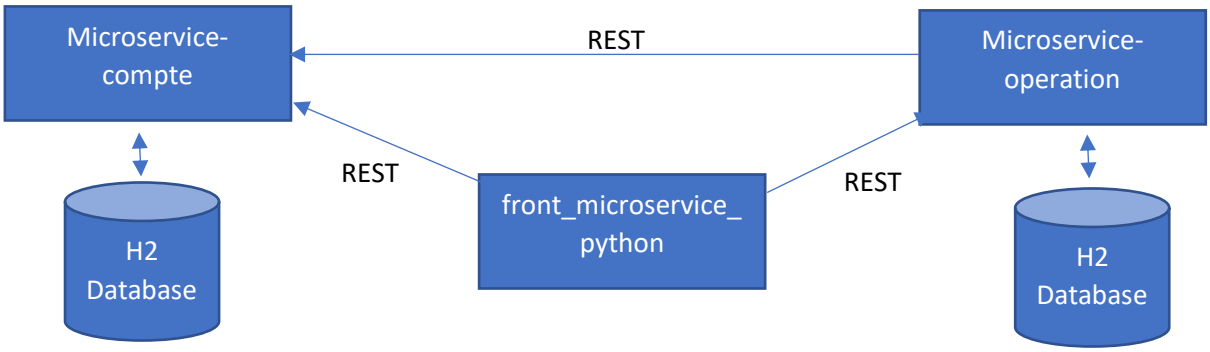
Schéma d'architecture

Notre application est découpée en 3 microservices qui sont tous les 3 dockerisés :

Microservice-compte : permet d'afficher des informations relatives à un compte, de gérer un compte (créer, modifier, supprimer, lister), rechercher des comptes par type de compte, Iban…

Microserivce-operation : permet d'afficher des informations relatives à une opération bancaire, de gérer une opération (créer, modifier, supprimer, lister), de rechercher une opération (par date, par type, par iban source/dest), de faire un retrait/dépôt d'argent, de faire un virement.

Front_microservice_python : permet d'afficher l'ensemble des actions de nos microservices, il s'agit en fait de notre application cliente.



Choix techniques

La partie serveur a été implémentée sous forme de microservices, à l'aide de Spring Boot, qui communiquent entre eux grâce au protocole http et REST.

Les deux microservices « compte » et « opération » ont chacun leur base de données H2. Nous avons décidé d'utiliser cette base de données car cette base de données ultra-légère n'est pas conçue pour supporter de gros volumes de données ni un nombre important d'utilisateurs, mais elle est extrêmement pratique pour les développeurs Java qui ont besoin de tester des applications. Ainsi, elle convient parfaitement aux exigences de notre projet.

La partie client a été implémentée en Python avec le microframework flask et fonctionne par le biais d'appels REST sur nos deux microservices.

La persistance est gérée à travers JPA en utilisant Spring Data.

Enfin, nous avons décidé de conteneuriser nos 3 microservices grâce à Docker.

Diagramme de classes métier

Microservice-compte :

Compte
- Long : id - String : iban - String : typedecompte - Double : interet - String : frais - Double : solde

Microservice-operation :

Opération
- Long : id - String : type - String : ibansource - String : ibandest - Double : montant - Date : date

Compilation/exécution du projet

Pour compiler et exécuter notre projet il suffit de suivre les étapes suivantes :

1. Faire un git clone de chacun des projets pour récupérer les microservices.
2. Une fois que nous avons récupéré les microservices il faut les builder grâce à la commande : mvn clean install.

   Cela va nous permettre de générer un .jar pour chaque microservice.

3. On va ensuite dockeriser nos microservices, pour cela on va construire une image docker pour chacun de nos microservices.

   Exemple pour le microservice « compte » :

   On exécute la commande suivante : docker build -f Dockerfile -t microservice-compte .

   Voici ce qu'il se passe :

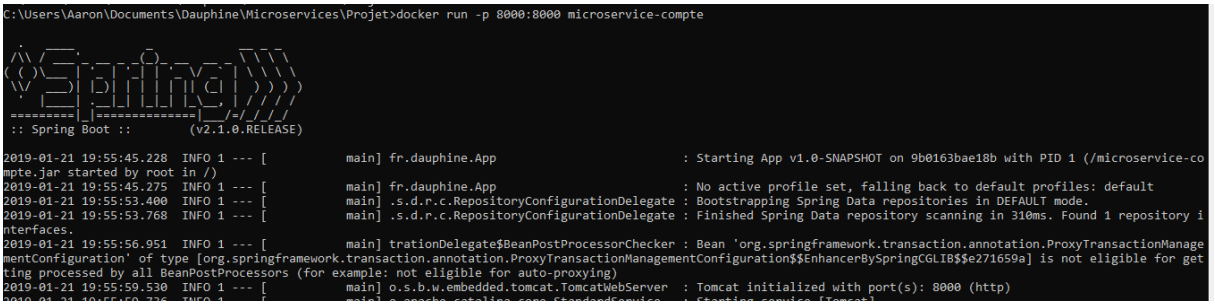
```
C:\Users\Aaron\Documents\Dauphine\Microservices\Projet>docker build -f Dockerfile -t microservice-compte .
Sending build context to Docker daemon  80.97MB
Step 1/4 : FROM openjdk:8
8: Pulling from library/openjdk
cd8eada9c7bb: Pull complete
c2677faec825: Pull complete
fcce419a96b1: Pull complete
045b51e26e75: Pull complete
88e50f3a5916: Pull complete
a7c546cdc7ce: Pull complete
73b0460146c0: Pull complete
428095c4a908: Pull complete
Digest: sha256:fd26a0ce164bee965095ba91ee7c4f21f3f98f54247e240c1f98255bd200cbe8
Status: Downloaded newer image for openjdk:8
 ---> f2194a7e67df
Step 2/4 : ADD target/microservice-compte.jar microservice-compte.jar
 ---> fb227fbbf50a
Step 3/4 : EXPOSE 8000
 ---> Running in 001eba53c179
 ---> 11d4e063b47e
Step 4/4 : ENTRYPOINT ["java", "-jar", "microservice-compte.jar"]
 ---> Running in 5fa28d2abf1c
 ---> b8057ac6e7fb
Removing intermediate container 001eba53c179
Removing intermediate container 5fa28d2abf1c
Successfully built b8057ac6e7fb
Successfully tagged microservice-compte:latest
```

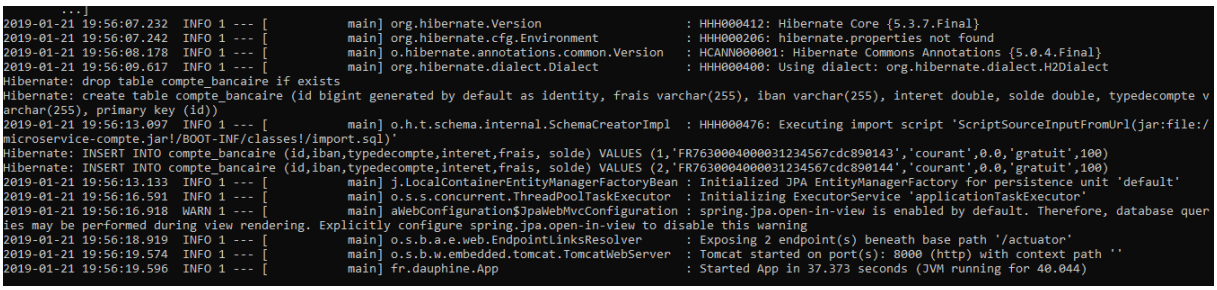
4. Après avoir builder nos images nous les lançons dans des conteneurs ce qui va nous permettre d'exécuter notre application Spring Boot.

   Exemple pour le microservice « compte » :

   On exécute la commande suivante : docker run -p 8000:8000 microservice-compte

   Voici ce qu'il se passe :





   Ainsi, nos microservices s'exécutent dans des conteneurs distincts et indépendants.

   On peut alors interroger nos microservices en mode REST à travers HTTP en exécutant par exemple cette requête : *192.168.99.100:8000/compte/all*, ce qui va nous renvoyer tous les comptes.
   Toutes les requêtes possibles figurent dans les fichiers Readme.

   De plus, vous trouverez aussi toutes les étapes qui précèdent dans le fichier Readme que nous avons écrits pour chacun de nos microservices sur Git, dont voici les liens :

   Microservice-compte : https://github.com/aaron101295/microservice-compte
   Microservice-operation : https://github.com/aaron101295/microservice-operation
   Front_microservice_python : https://github.com/raphaeluzan/front_microservice_python

Bilan du projet

Ce projet nous as permis de découvrir des notions importantes que chaque développeur devrait avoir connaissance.

D'une part, l'architecture orienté service (SOA) et les microservices ainsi que les grands principes de l'API REST. En effet, ce projet nous a permis de démontrer qu'il était possible d'avoir de

l'interopérabilité entre des microservices développés de deux manières et dans deux langages différents grâce aux API.

D'autre part, un des frameworks les plus répandus et utilisés, Spring Boot.

De plus, nous avons aussi eu la possibilité d'expérimenter la conteneurisation grâce à Docker.

Nous avons eu quelques difficultés quant à l'implémentation des services REST ainsi que la dockerisation. Néanmoins, nous sommes tout de même parvenus à faire fonctionner l'ensemble sous les contraintes imposées.