Projet API Restful

Introduction

Il s'agit d'un projet visant à recréer une API de gestion des comptes bancaires permettant aux utilisateurs inscrits de créer et gérer leur compte.

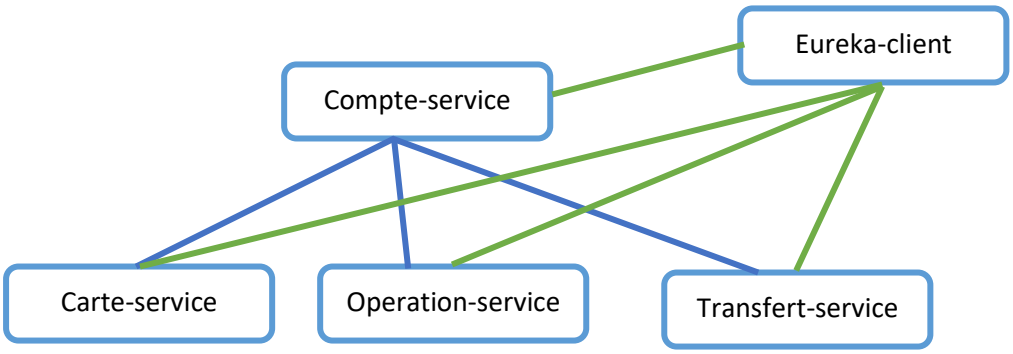
Les objectifs principaux étaient les suivants :

- Développer une API RESTful en utilisant la technologie vue en cours (Spring boot),
- Mettre en pratique HATEOAS (liens compte-carte, liens comptes-opérations, …),
- Prendre en compte l'aspect sécurité de l'application (authentification et autorisation).

Dans les parties suivantes nous verrons en détail les différentes parties de l'API ainsi que les possibilités qu'elles offrent mais également leurs limites et les voies d'amélioration possibles.

Conception

On veut créer une API RESTful qui sera donc composée de micro-services et pas d'un gros monolithe immuable et impossible à déboguer. Pour répondre au besoin exprimé, il faudra donc articuler notre API autour d'un ensemble de services distincts et communicants. Les services mis en place sont les suivants :



Le principe de l'API est de laisser le champ libre au maximum pour les utilisateurs, on laisse donc au côté client le soin de gérer les problèmes de format des données entrées et l'utilisation des outils mis à disposition. Compte-service sert de point d'entrée de l'application et permet d'accéder à l'ensemble de ses fonctionnalités, cependant, cela est tributaire comme toute architecture micro-services de l'état des autres services. En effet, Compte-service ne pourra pas accéder aux cartes si Carte-service n'est pas en ligne. Eureka s'occupe de la réplication des services et du load-balancing. Il a simplement été repris des TP pour garantir le bon fonctionnement des services, aucune des classes d'Eureka n'a été modifiée.

Concernant la conception en elle-même et les diagrammes résultants (Cas d'utilisation, diagrammes de classe), ils sont disponibles dans le dossier "Diagrammes UML". Les différents services sont

conformes à la méthode CRUD et permettent donc de Créer, lire, modifier et supprimer les données. La persistance des données est faite grâce à une base de données H2.

Démarrage et utilisation

Pour démarrer les différents services il suffit de se rendre dans le dossier respectif du service concerné et de lancer la commande :

java -jar target/[nom du fichier .jar du service]

Comme les services dépendent d'Eureka, il faut lancer Eureka en premier, une fois Eureka lancé, vous pouvez allumer les trois autres services.

Rendez-vous sur localhost:8761 (adresse d'Eureka) par défaut pour voir les services en fonctionnement.

Chaque service peut être utilisé indépendamment mais seul Compte-service permet d'accéder aux fonctionnalités de l'ensemble des services. Les méthodes suivantes sont disponibles pour chaque service.

Compte service :

Get :

localhost:7000/comptes -> retourne tous les comptes

localhost:7000/comptes/{id} -> retourne le compte sélectionné

localhost:7000/comptes/{id}/cartes -> retourne toutes les cartes liées au compte désigné

localhost:7000/comptes/{id}/transferts -> retourne toutes les transferts liées au compte désigné

localhost:7000/comptes/{id}/operations -> retourne toutes les operations liées au compte désigné

localhost:7000/comptes/{id}/solde-> retourne le solde du compte désigné

Post :

Les requêtes Post prennent toujours un fichier JSON en body qui correspond à un objet qu'on veut créer.

localhost:7000/comptes -> Crée un nouveau compte

localhost:7000/comptes/{id}/cartes -> Crée un nouvelle carte associée au compte

localhost:7000/comptes/{id}/operations -> Crée une nouvelle operation liée au compte

localhost:7000/comptes/{id}/transferts -> Crée un nouveau transfert lié au compte

Put :

Les requêtes Put prennent toujours un fichier JSON en body qui correspond à l'objet qui nous intéresse mis à jour.

localhost:7000/comptes -> Met à jour le compte

localhost:7000/comptes/{id}/cartes -> Met à jour la carte associée au compte

localhost:7000/comptes/{id}/operations -> Met à jour l'operation liée au compte

localhost:7000/comptes/{id}/transferts -> Met à jour le transfert lié au compte

Delete :

localhost:7000/comptes/{id} -> Efface le compte

localhost:7000/comptes/{id}/cartes/{idCarte} -> Efface la carte associée au compte

localhost:7000/comptes/{id}/operations/{idOperation} -> Efface l'operation liée au compte

localhost:7000/comptes/{id}/transferts/{idTransfert} -> Efface le transfert lié au compte

Carte, Operation et transfert

Ces trois services répondent aux méthodes CRUD sur les ports respectifs :

- Carte : 7010
- Operation : 7020
- Transfert: 7030

Difficultés rencontrées lors du projet et voies d'amélioration

Un point important d'une application de banque est la sécurité, malheureusement je n'ai pas réussi à faire fonctionner le module Oauth vu en cours.

Tester une API s'est trouvé bien plus dur que prévu et la résultante est que seule deux méthodes de carte-service sont vraiment testées via Mockito.

Il aurait possible de mettre en place Hystrix pour gérer les services qui tomberaient en cours d'utilisation.