08/02/2019

Documentation pour l'utilisateur

Inclus les instructions de compilation et d'exécution, la documentation technique et le bilan de notre bilan du projet.

TIOUCHICHINE Lounes
AISSAT Yacine

I) Instructions de compilation et d'exécution :

Lié au fait que nous n'avons pu mettre en place la conteneurisation, notre projet n'est pas exécutable facilement…

Les prérequis pour pouvoir utiliser notre projet sont :
Avoir le logiciel POSTMAN installé sur sa machine
Avoir le logiciel INTELLIJ installé sur sa machine
Avoir le logiciel H2 installé sur sa machine

Le lien GitHub fourni par mail contient les 2 fichier .zip correspondant à nos deux microservices implémentés. Le contenu des deux microservices est décrit un peu plus en détail dans la partie II) de ce même rapport.

Etapes à suivre pour pouvoir faire fonctionner le projet.

1 – Télécharger les deux fichiers .ZIP disponibles sur le lien GitHub fourni par mail
2 – Dézipper les deux .ZIP
3 – Dans intellij, importer un dossier dézippé de la manière suivante :
File -> New Project from existing sources
Il faut ensuite choisir le fichier. Le projet s'ouvre alors dans intellij
4 – Ouvrir une nouvelle fenêtre d'intellij et répéter la même opération pour l'autre dossier.

5 - Maintenant que les 2 microservices sont ouverts sur intellij, il faut maintenant de les compiler.

Pour cela, il suffit dans intellij de cliquer sur la fonctionnalité suivante :

Cela aura pour effet de Build le projet

Suite à cela, cliquer sur la fonctionnalité suivante :

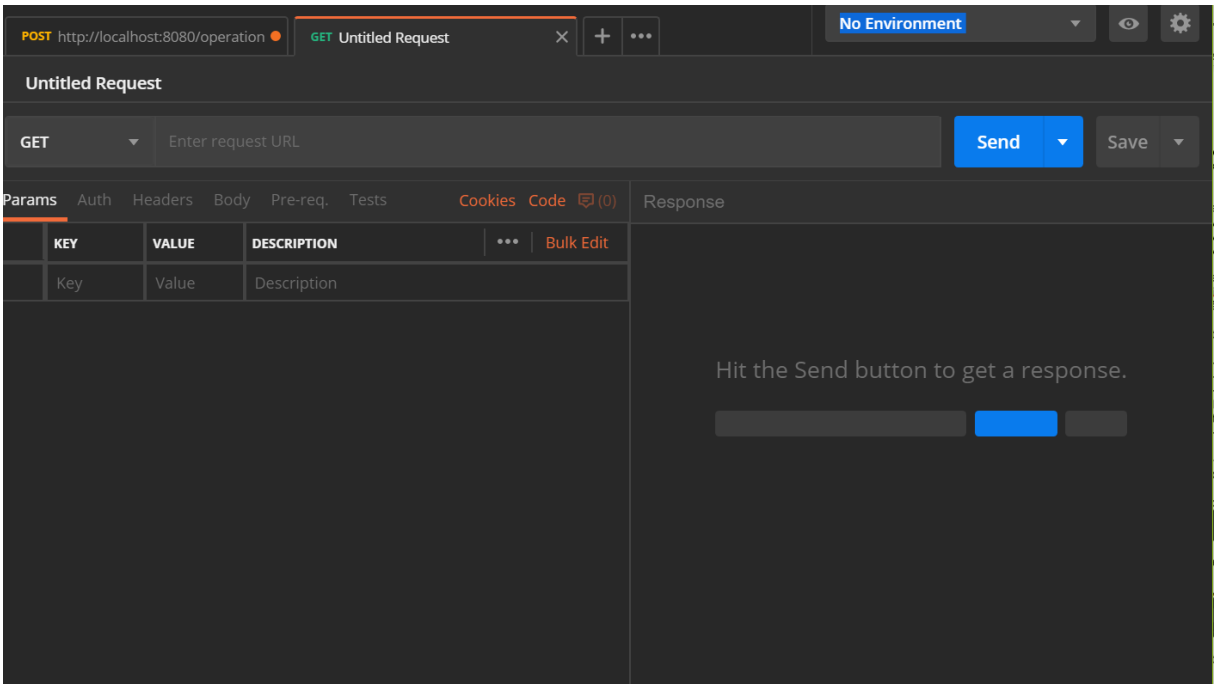
Et taper la commande suivante : mvn spring-boot :run

L'étape 5 est à répéter pour chacun des projets ouverts. Bien que les deux microservices puissent fonctionner indépendamment l'un de l'autre, il faut ici exécuter les deux afin de pouvoir les tester.

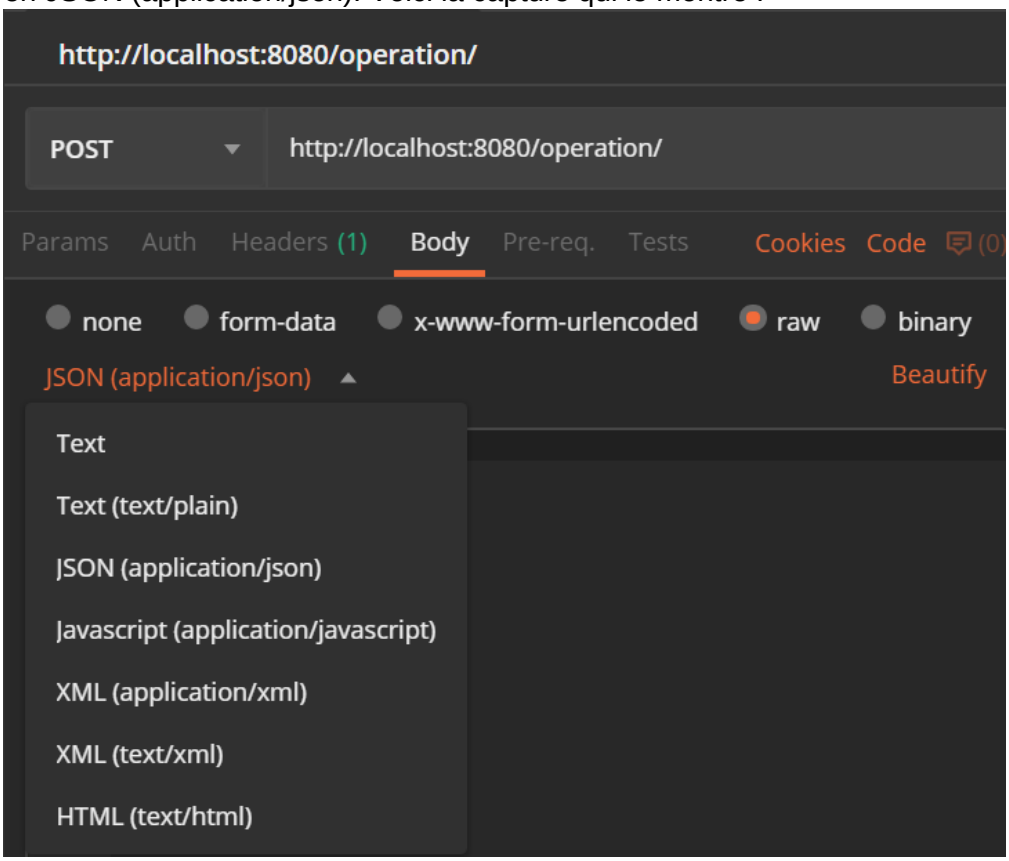
Maintenant, les deux microservices sont lancés et fonctionnels. Il convient maintenant de les utiliser.

Pour cela, on va utiliser POSTMAN qui fait office ici de client pour envoyer des requêtes HTTP et tester les fonctionnalités.

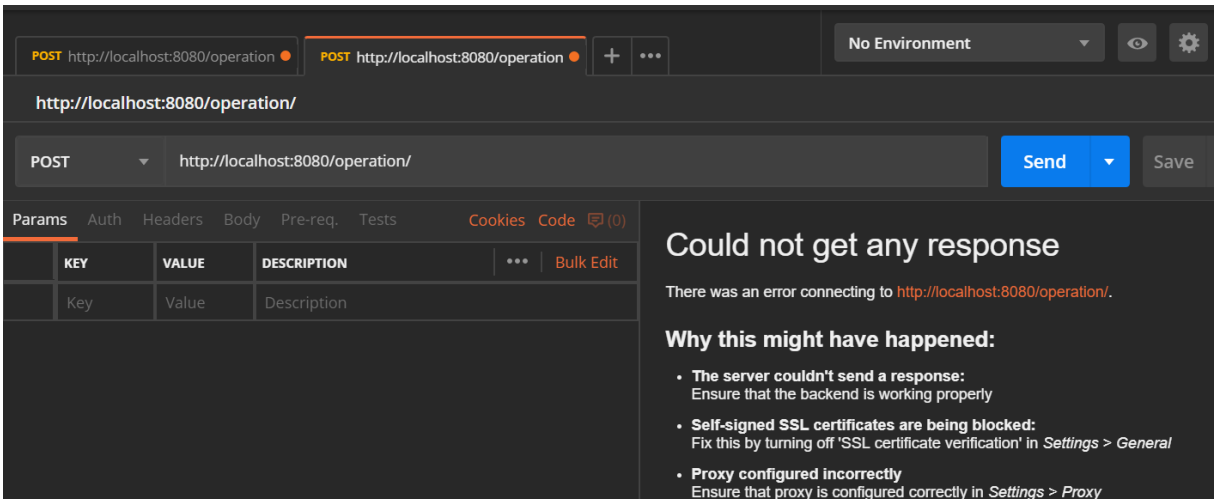
Le logiciel se présente ainsi :



Une configuration importante à faire est de cliquer sur Body, de choisir RAW, et de changer en JSON (application/json). Voici la capture qui le montre :



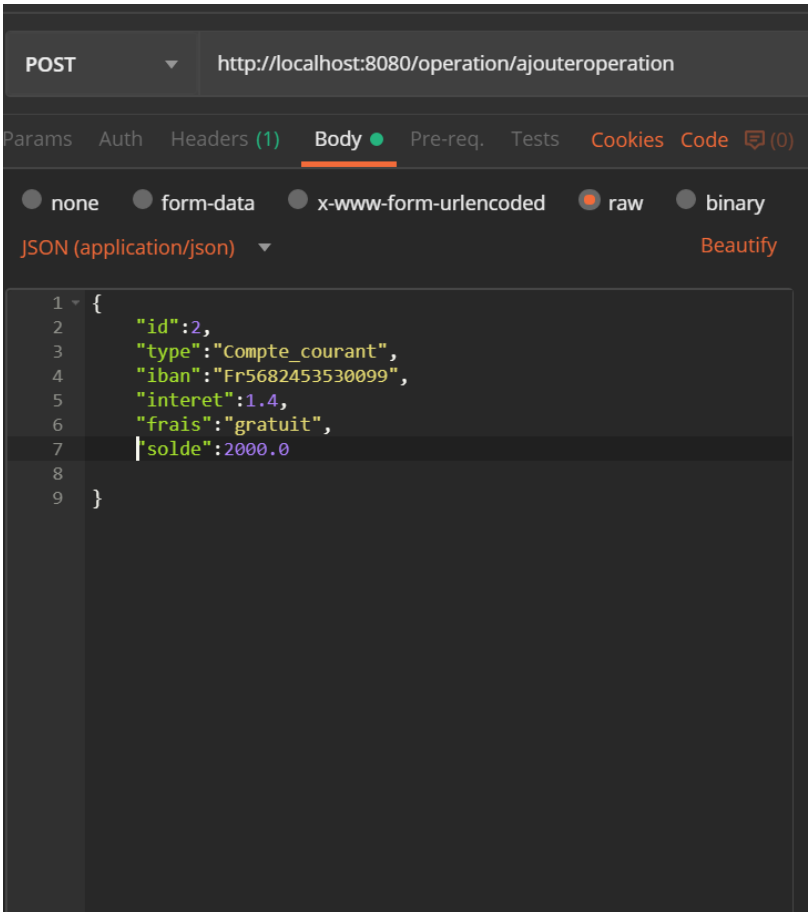
Maintenant, pour ajouter, par exemple, une opération cela se fait comme ceci :



Pour le microservice opération, voici l'url :
http://localhost:8080/operation/nouvelleoperation

Bien choisir la méthode POST comme sur la capture, car on souhaite faire un ajout.

Et maintenant, dans BODY utiliser le format suivant :



{

```
        "id":2,
        "type":"Compte_courant",
        "iban":"Fr5682453530099",
        "interet":1.4,
        "frais":"gratuit",
        "solde":2000.0
}
```


Mettre les informations que l'on souhaite.

Maintenant, en ce qui concerne l'autre microservice Compte. Si l'on souhaite ajouter, par exemple, un compte cela se fait comme ceci :

Pour le microservice opération, voici l'url :
http://localhost:8080/operation/nouvelleoperation

Bien choisir la méthode POST comme sur la capture, car on souhaite faire un ajout.

Pour le microservice compte, voici l'url :
http://localhost:8000/compte/

On remarquera que Compte tourne le port 8000, différent de celui d'Opération qui est configuré à 8080.

Ainsi, pour ajouter un compte, il suffit d'utiliser cette fois ci, toujours avec la méthode POST, la requête suivante : http://localhost:8000/compte/nouveaucompte

D'autres requêtes implémentées dans le code sont également utilisables, par exemple avec la méthode POST on peut exécuter la requête http://localhost:8000/compte/touslescomptes ou bien encore avec la méthode GET il est possible de faire un http://localhost:8000/compte/touslescomptes.

D'autres fonctionnalités sont également disponibles, il est compliqué de toutes les énumérer ici mais nous vous proposons de voir l'ensemble des fonctionnalités de notre projet au cours d'une petite démonstration de vive voix lors de la soutenance.

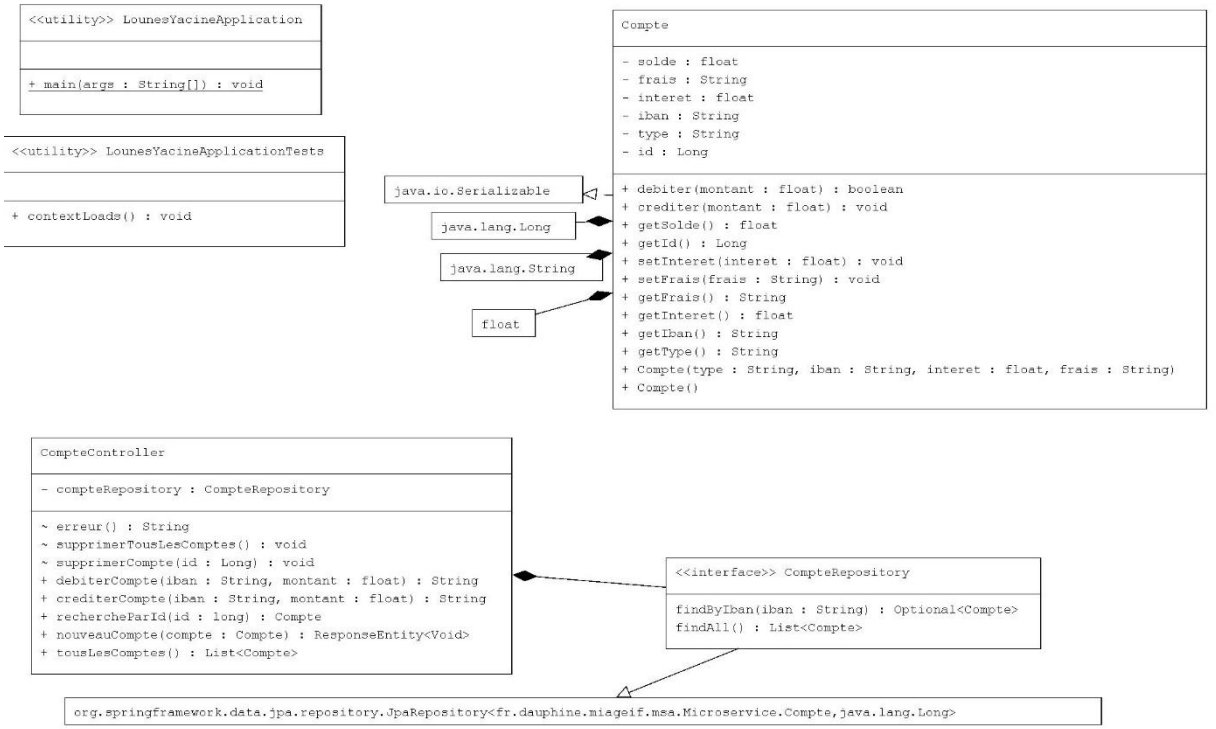
II) Documentation technique :

Nous avons fait de longues recherches afin de faire le choix de la structure à utiliser pour notre projet. Finalement, notre projet se présente comme suit.

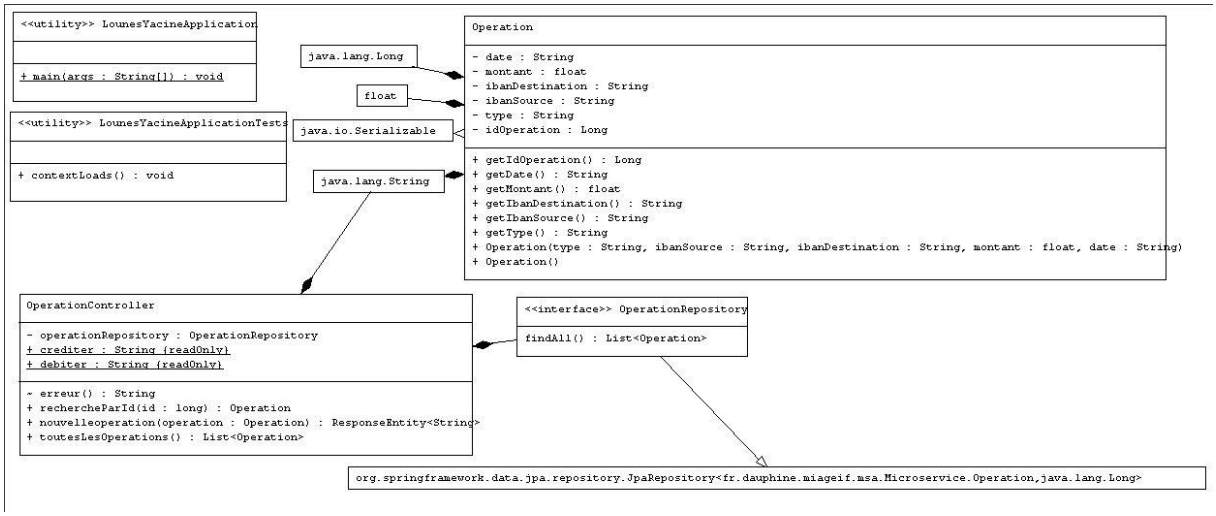
Nous avons deux Microservices totalement indépendant l'un de l'autre.

Le premier est le Microservice « Compte » qui permet la gestion et le maintien des comptes bancaires. Il contient les informations basiques du compte telles que l'IBAN, le type de compte, les intérêts associés, les frais et contient également le solde du compte. Cette dernière donnée est très importante car elle va être utilisée par l'autre Microservice « Opération » lorsque des virements sont effectués afin de mettre à jour le solde des

comptes. Voici d'abord le diagramme de classes permettant de comprendre la structure du Microservice Compte :



Le deuxième Microservice est donc comme dit précédemment le Microservice « Opération ». Il permet lui de gérer l'ensemble des opérations. Il contient les informations basiques pour effectuer des opérations bancaires telles que l'IBAN source, l'IBAN destination, le type de l'opération à effectuer, le montant de l'opération, et la date de l'opération. Voici d'abord le diagramme de classes permettant de comprendre la structure du Microservice Opération :



III) Bilan du projet :

Ce projet fut un projet très intéressant pour nous. Travaillant tous les deux en banque (BNP Paribas) et souhaitant tous les deux continuer exclusivement en banque après obtention de notre diplôme, ce projet touche à un univers qui nous est familier. Bien que durant ce projet, ce ne soit qu'une gestion de comptes et d'opérations très sommaire qui est fait, l'univers qui y est associé nous intéresse tous les deux.

Ensuite, sur ce que nous avons appris, nous pouvons dire quasiment tout. A part le langage Java qui nous était familier, la notion de Microservice et même la connexion à une base de données n'est pas quelque chose que nous avions effectué auparavant. Nous ne connaissions pas du tout Spring-boot et c'est un outil très performant car il permet en quelques minutes grâce aux annotations de réaliser des actions que nous n'étions pas en capacité d'implémenter nous-même. Quand je parle d'annotations, je parle des annotations du type @RestController, @RequestMapping… issues du Framework springframework.

Nous avons perdu beaucoup de temps sur des petits détails d'implémentations, comme c'est souvent le cas lors d'un projet informatique… Et cela nous a causé du tord car finalement nous n'avons pas pu mettre en place l'ensemble des fonctionnalités que nous souhaitions implémenter. Par exemple, de nombreuses vérifications sur les Opérations n'ont pas pu être mise en place alors que cela ne demandait pas beaucoup de travail et n'était pas très compliqué à faire.

La principale perte de temps que nous avons subie est dans la Conteneurisation à l'aide de Docker. Nous avons bloqué sur cet aspect là et c'est très dommage car c'était quelque chose que nous aurions voulu mettre en place et surtout que nous aurions souhaité savoir utiliser.

Notre plus grand regret dans ce projet est de nous être obstinés à vouloir mettre en place la conteneurisation avec Docker pour au final se retrouver à ne même pas avoir le temps de fournir un Jar exécutable.