Projet change de devises

Solène CHATEAU
Jérémy GERMANICUS

Département MIDO
Année universitaire 2017 – 2018
M2 MIAGE-IF

I – Compilation du projet

Afin de compiler le projet et de parvenir à exécuter des opérations sur un taux de change, il faut tout d'abord charger les projets Java dans l'IDE IntelliJ et lancer chacun des deux projets en les exécutant, les micro services vont alors être lancés en tache de fond.
L'utilisateur a alors à sa disposition deux adresses :

- http://localhost:8000 : Lui permettant d'enregistrer des taux de changes entre deux devises
- http://localhost:9000 : Lui permettant d'enregistrer des opérations sur les taux de changes

Sur chacune des adresses plusieurs options sont possibles, il suffit de changer l'adresse pour obtenir un service. Les écritures entre crochets ({}) sont des paramètres que l'utilisateur doit renseigner

Adresses pour les taux de changes :

- http://localhost:8000/Taux/id/{id} : Trouve le taux de change correspondant à l'id
- http://localhost:8000/Taux : Liste tous les taux de change
- http://localhost:8000/Taux/Delete/{id} : Supprime le taux de change correspondant à l'id
- http://localhost:8000/Taux/Add/from/{from}/to/{to}/date/{date}/taux/{taux} : Ajoute un taux de change étant composé des paramètres passés par l'utilisateur
- http://localhost:8000 /Taux/Update/id/{id}/from/{from}/to/{to}/date/{dateTaux}/taux/{taux} : Modifie le taux de change correspondant à l'id avec les paramètres passés par l'utilisateur
- http://localhost:8000/Taux/from/{from}/to/{to}/date/{date} : Retrouve le taux de change correspondant aux paramètres passés par l'utilisateur

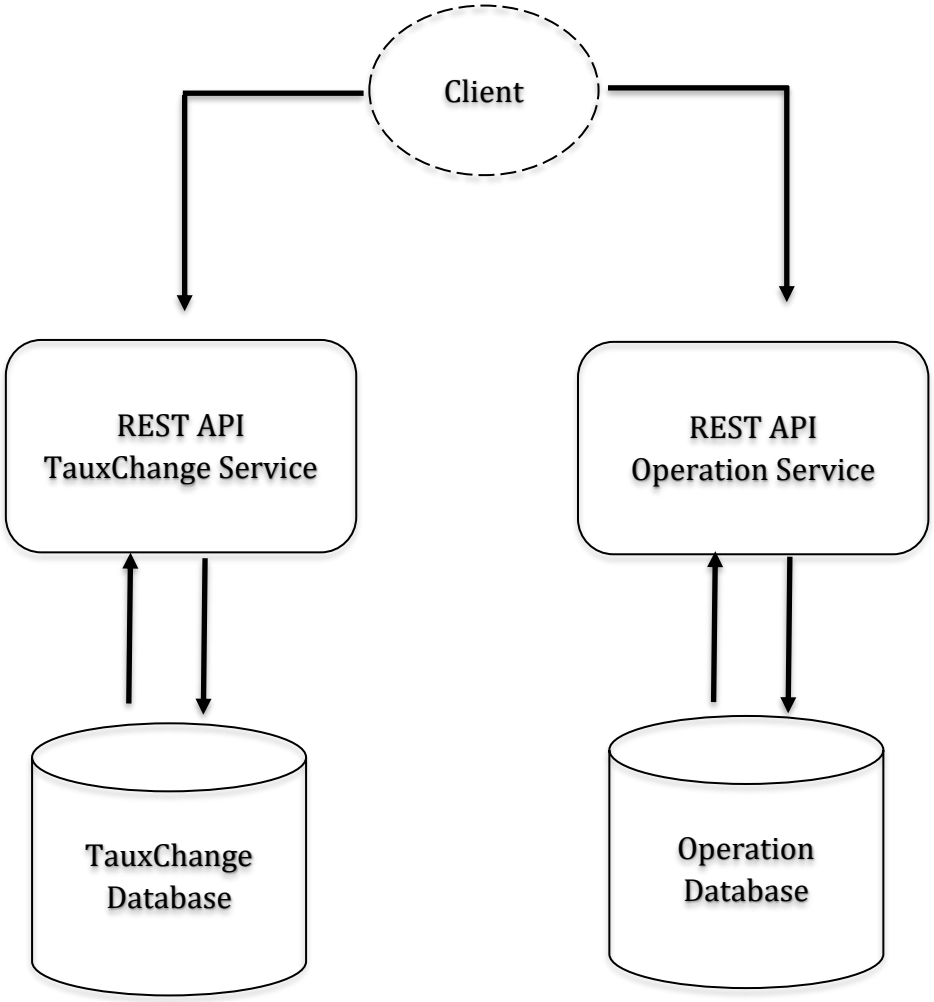
Adresses pour les opérations :

- http://localhost:9000/Operation/{id} : Trouve l'opération correspondant à l'id
- http://localhost:9000/Operation/Add/montant/{montant}/from/{from}/to/{to}/date/{date} : Ajoute une opération étant composée des paramètres passés par l'utilisateur
- http://localhost:9000/Operation/Delete/{id} : Supprime l'opération correspondant à l'id
- http://localhost:9000//Operation/Update/id/{id}/from/{from}/to/{to}/montant/{montant}/date/{date}/taux/{taux} : Met à jour l'opération correspondant à l'id avec les paramètres passés par l'utilisateur

- http://localhost:9000/Operation/from/{from}/to/{to}/date/{date} : Trouve le taux de change correspondant aux paramètres

II – Documentation technique

Schéma d'architecture :

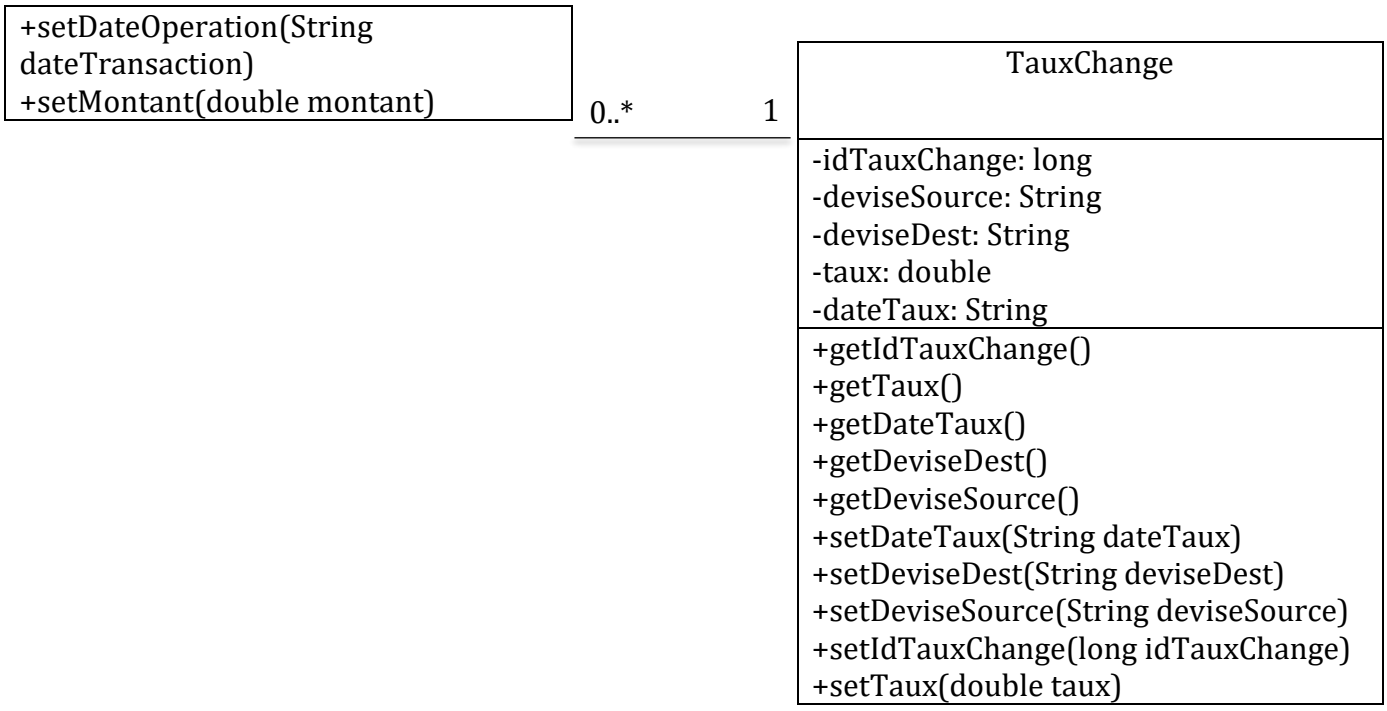


Choix techniques :

Les choix techniques sur nous avons fait, ont beaucoup été guidés par le dernier cours que vous nous avez fourni. Ne connaissant pas du tout les application micro-services nous nous sommes énormément appuyés sur les cours dispensés. C'est pour cela que nous avons développé sur IntelliJ per exemple.
Le principal choix technique que nous fait est celui concernant le nombre de classe que nous avons implémenté. Nous avons privilégié deux classes plutôt que trois, nous avons décidé de ne pas faire de classe pour représenter la devise, mais à la place nous avons choisi de faire plus simple, en ajoutant un attribut de type String dans chacune des deux classes existantes. Nous aurons donc un attribut représentant la devise dans la classe pour le taux de change et celle pour l'opération. Ce choix technique a été fait dans le but de simplifier la lisibilité de nos classes. Nous avons voulu jouer sur la simplicité.
Le second choix technique que nous avons effectué a été de stocker les dates comme des chaines de caractères. Nous savons que ce choix n'est pas le meilleur, mais il nous a permis de gérer les différents échanges de dates plus facilement, car sinon nous avions des problèmes lorsque nous échangions des types date.
Pour la réalisation des tests nous avons d'une part utilisé un navigateur web pour visualiser les bases de données et d'autre part nous avons utilisé un petit programme appelé Postman. Il permet d'effectuer différent type de requêtes (GET, POST, etc…) et de visualiser sous un format JSON indenté les résultats des requêtes.

Diagramme de classe :

Operation
-idOperation: long -deviseSource: String -deviseDest: String -montant: double -taux: double -dateOperation: String
+getDeviseSource() +getDeviseDest() +getIdOperation() +getTaux() +getDateOperation() +getMontant() +setDeviseSource(String deviseSource) +setIdOperation(long idOperation) +setTaux(double taux) +setDeviseDest(String deviseDest)


+setDateOperation(String dateTransaction)
+setMontant(double montant)
0..*
1
TauxChange
-idTauxChange: long
-deviseSource: String
-deviseDest: String
-taux: double
-dateTaux: String
+getIdTauxChange()
+getTaux()
+getDateTaux()
+getDeviseDest()
+getDeviseSource()
+setDateTaux(String dateTaux)
+setDeviseDest(String deviseDest)
+setDeviseSource(String deviseSource)
+setIdTauxChange(long idTauxChange)
+setTaux(double taux)

III – Bilan du projet

Ce projet a été très intéressant, et nous a fait découvrir dans un contexte pratique les applications d'un projet micro-services. Cela a été très intéressant de mettre en place les deux micro-services, de les connecter ensemble, et également de les connecter à une base de données. Nous avons beaucoup aimé pouvoir tester notre application sur un navigateur web, cela rendait notre projet concret, et nous motivé d'autant plus à trouver des solutions lorsque nous avons rencontré des problèmes.

Les principales difficultés que nous avons rencontrées ont été d'installer les différents Frameworks et les logiciels nécessaires au développement de ce projet micro-services, nous avons eu du mal à installer Spring Boot.
Une des difficultés majeures de ce projet a été la conteneurisation avec Dockers. En effet nous n'avons pu installer ce programme sur aucun de nos PC notamment à cause dépendances manquantes et impossibles à installer.
Nous avons également rencontré des problèmes sur la gestion des dates. Et nous avons également eu différents problèmes mineurs lors du développement du code Java.

Pour conclure, nous avons trouvé ce projet très intéressant et challengeant, et nous avons beaucoup appris.