Documentation

I. Démonstration youtube et lien github

https://www.youtube.com/watch?v=Att9iZCtgl4&feature=youtu.be

https://github.com/Anisbel/RestSpringSpark

II. Rouler le projet en local

Pour rouler le projet en local seulement, il faut d'abord passer par l'installation de plusieurs modules. Veuillez passer à *l'Annexe I* pour l'installation des modules.

Après avoir installer les modules et cloner le projet, nous allons passer à la création de la BD:

1) Pour créer les tables dans la BD. Ouvrez un terminal, et entrez : cqlsh

    a) D'abord, il faut créer un keyspace. Faite donc:
        i) Create key space;
        ii) CREATE KEYSPACE javasampleapproach WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1 };
        iii) use javasampleapproach;

    b) Ensuite, pour créer la table produit et facture, faite ces commandes :
        i) CREATE TABLE produit(id int PRIMARY KEY, fk_FactureID int, nameProduit text, prix int);
        ii) CREATE TABLE facture(id int PRIMARY KEY);

Maintenant, nous allons lancer le serviceSpark, le serviceRest, et le Client :

2) Pour lancer le ServiceSpark
    a) Ouvrez un terminal et dirigez vous vers le path *RestSpringSpark/Service_Spark/*
    b) lancer la commande mvn spring-boot:run
    * si erreur, voir Annexe II

3) Pour lancer le serviceRest
    a) Ouvrez un terminal et dirigez vous vers le path *RestSpringSpark/Service/*
    b) lancer la commande mvn spring-boot:run
    *si erreur, voir Annexe II

4) Pour lancer le Client
    a) Ouvrez un terminal et dirigez vous vers *RestSpringSpark/Client_REST/*
    b) lancer la commande mvn spring-boot:run
    c) les produits les plus fréquents seront retournés.

Important : Lorsque vous arrêtez un programme, il faut toujours s'assurer que le processus est mort et que le port est libéré en roulant la commande :

kill -9 $(lsof -t -i:8080)

8080 représente le port qui doit être libéré par le process.
Dans notre cas, 8080 est le serviceSpark ,8090 est le serviceRest, et 6090 est le Client.
Cette action est requise dû à une version spécifique de spring boot.

Pour tester et changer des valeurs dans l'application :

5) Insérer une facture avec des produits spécifique.
    a) Ouvrez */Client_REST/src/main/java/com/example/demo/DemoApplication.java*
    b) Modifiez les lignes `facture.getProduits().add(new Produit("patate",5));` par les éléments que vous voulez ajouter.
    c) Roulez le programme.

III. Rouler le projet avec une VM

1) Pour rouler le serviceSpark et le serviceRest dans la VM :

a) télécharger une VM et donner lui une adresse ip publique (voir : https://gist.github.com/pjdietz/5768124)
b) Dans la VM,
    i) appliquer les étapes 1) et 2) de IV.
    ii) Faites rouler les 2 services en appliquant les étapes 2) et 3) de II.
c) Dans la machine local,
    i) Modifier la classe DemoApplication du Client en remplaçant la ligne 16 par l'adresse IP de la machine virtuelle.

```
public static final String REST_SERVICE_URI="http://192.168.56.101:8090";
```

    ii) Lancer le client en appliquant les étapes 2) de II.

IV-Annexe I

1) Ouvrez un terminal et récupérer le projet :
    a) git clone https://github.com/Anisbel/RestSpringSpark

2) Assurez vous d'avoir :
    a) java version "10.0.2".
    b) Maven 3.6.0
    c) Cassandra et csqlsh
        i) lien :https://www.liquidweb.com/kb/install-cassandra-ubuntu-16-04-lts/

**L'application pourrait être instable si testée avec d'autres versions.*

V-Annexe II

Si un des services n'arrive pas a rouler :

1) 1er solution
    a) Allez dans le directory de ce service et faite : *mvn package*
    b) Aller dans le target file et rouler le jar en faisant : *java -jar XXX-snapshot.jar*
2) 2eme solution
    a) Assurez vous d'avoir les bonnes versions.

VI-Source d'inspiration

Nous nous somme inspirés de plusieurs sources pour faire le projet. Voici une liste :

1) https://spring.io/guides/gs/spring-boot/
2) https://github.com/Zhuinden/spring-spark-example

3) https://spark.apache.org/docs/2.3.0/mllib-frequent-pattern-mining.html
4) https://www.liquidweb.com/kb/install-cassandra-ubuntu-16-04-lts/
5) https://gist.github.com/pjdietz/5768124