Examen TP d'HEB
Prisca CLIO et Maxime Fritel ING3 FISE DATA

Notre repository github :
https://github.com/priscaclio/heb-customer-subscription/

La partie Scopt :

On avait d'une part commencé le projet avec le paramètre **args** de la classe main puis on a essayé d'utiliser scopt.
Dans le sbt on a rajouté une dépendance :

```
libraryDependencies += "com.github.scopt" %% "scopt" % "4.0.1"
```

Et dans notre scala class object **Service** on a écrit le code pour définir nos arguments (ligne 15 à 52).
On a créé quatre arguments qui sont requis pour que les services puissent fonctionner.

```
heb-customer-subscription 1.0
Usage: heb-customer-subscription [options]

  -s, --service <value>    required string delete or hasher
  -u, --id <value>         required long id
  -i, --inputpath <value>  required string path with csv file
  -o, --outputpath <value>
                           required string path to put the new csv file
```

Si on ne met pas d'argument alors on a le message suivant à l'exécution :

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
Error: Missing option --service
Error: Missing option --id
Error: Missing option --inputpath
Error: Missing option --outputpath
heb-customer-subscription 1.0
Usage: heb-customer-subscription [options]

  -s, --service <value>    required string delete or hasher
  -u, --id <value>         required long id
  -i, --inputpath <value>  required string path with csv file
  -o, --outputpath <value>
                           required string path to put the new csv file

Process finished with exit code 0
```

- **Service** :

Cet argument sert à choisir le service ‘hasher’ ou ‘delete’. Si autre chose que ‘hasher’ ou ‘delete’ est passé en argument alors le programme ne pourra pas se lancer.

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
Error: wong argument, you have to put 'delete' or 'hasher' with arg --service
heb-customer-subscription 1.0
Usage: heb-customer-subscription [options]

  -s, --service <value>    required string delete or hasher
  -u, --id <value>         required long id
  -i, --inputpath <value>  required string path with csv file
  -o, --outputpath <value>
                           required string path to put the new csv file

Process finished with exit code 0
```

- **Id** :

Cet argument sert à passer l’id du client auquel on veut appliquer un service (soit supprimer les données, soit les hasher)

- **Inputpath** :

Cet argument sert à donner le chemin ou ce trouve les données (type csv)

- **Outputpath** :

Cet argument sert à donner le chemin ou on va écrire les nouveaux fichiers csv avec les modifications apporté.

Nous n’avons pas trouvé le moyen encore de gérer les exceptions, vérifier que les chemins donnés en argument (inputpath et outputpath) existent.

<u>Pour le Schéma</u> :

On a commencé d’une part avec ce qu’on avait vu en cours :

```
val schema = StructType(
  Seq(
    StructField("IdentifiantClient", IntegerType),
    StructField("Nom", StringType),
    StructField("Prénom", StringType),
    StructField("Adresse", StringType),
    StructField("Date de Souscription", TimestampType)
  )
)
```

Puis comme le sujet le proposait de le faire avec un fichier de configuration json. Nous n'avons pas réussi à le faire avec le format donnée dans l'énoncé (le fichier schemaconfig.json dans le dossier config) mais on a réussi avec spark-json-schema

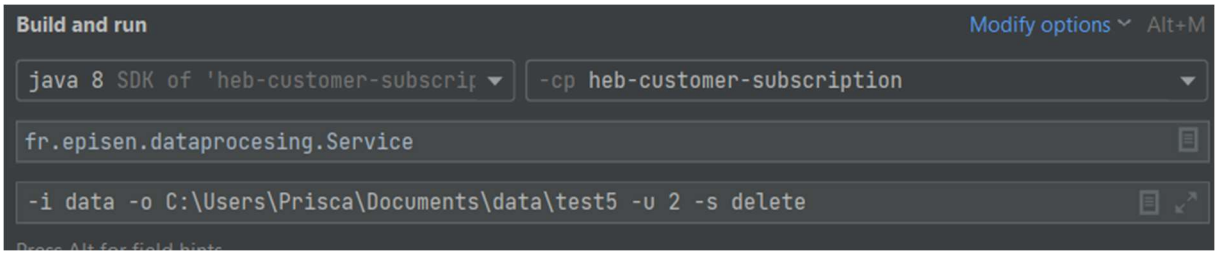
On a ajouté la dépendance suivante au sbt :

```
libraryDependencies += "org.zalando" %% "spark-json-schema" % "0.6.1"
```

Puis on a écrit le code (ligne 82 à90) qui permet de mapper les donnée avec le fichier de config json schema.json present dans le dossier config

- On n'a pas encore réussi à résoudre le problème du type de la colonne DateDeSouscription

Pour exécuter le main :



On a eu donc en retour par exemple (exemple de fichier csv dans le dossier data):

Sans schéma ->

```
+-----------------+----+------+-------+------------------+
|IdentifiantClient| Nom|Prenom|Adresse|DateDeSouscription|
+-----------------+----+------+-------+------------------+
|                1|Clio|Prisca|   test|        12/12/2022|
|                2|Clio|   Dan|   test|        12/12/2022|
|             test|Clio|   Dan|   test|              test|
|                3|Clio|   Pet|   test|        12/12/2022|
+-----------------+----+------+-------+------------------+
```

Avec schéma ->

```
+-----------------+----+------+-------+------------------+
|IdentifiantClient| Nom|Prenom|Adresse|DateDeSouscription|
+-----------------+----+------+-------+------------------+
|                1|Clio|Prisca|   test|        12/12/2022|
|                2|Clio|   Dan|   test|        12/12/2022|
|             null|null|  null|   null|              null|
|                3|Clio|   Pet|   test|        12/12/2022|
+-----------------+----+------+-------+------------------+
```

La troisième ligne n'ait pas prise en compte car elle ne respecte pas le schéma (colonne IdentifiantClient)

Ensuite on a commencé à coder les deux services.
Dans le sbt on a ajouté les deux dépendances suivantes :

```
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.4.8"
libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.4.8"
```

Ainsi que celle-ci pour résoudre une erreur à l'exécution :

```
dependencyOverrides ++= {
  Seq(
    "com.fasterxml.jackson.module" %% "jackson-module-scala" % "2.6.7.1",
    "com.fasterxml.jackson.core" % "jackson-databind" % "2.6.7",
    "com.fasterxml.jackson.core" % "jackson-core" % "2.6.7"
  )
```

Avant de coder les services, on a vérifié que l'id Client passé en argument existait sinon on renvoyait un message (de même si l'id n'était pas unique le dans le fichier), ligne 92 à 104.

Ligne 73 on a créé une sparkSession

```
val sparkSession = SparkSession.builder().master("local").getOrCreate()
```

Et ligne 83 à 90 le dataframe qui nous permet d'accéder aux données des fichiers csv

- Ligne 109 à 116 on a codé le service pour supprimer les données d'un client

Retour avec id=2 en argument ->

```
+-----------------+----+------+-------+------------------+
|IdentifiantClient|Nom |Prenom|Adresse|DateDeSouscription|
+-----------------+----+------+-------+------------------+
|1                |Clio|Prisca|test   |12/12/2022        |
|3                |Clio|Pet   |test   |12/12/2022        |
+-----------------+----+------+-------+------------------+
```

Des 4 lignes de départ il en reste 2 avec les donnée supprimé pour l'id 2 (une ligne a était enlevé car elle ne correspondait pas au schéma)

- Ligne 120 à 147 on a codé le service pour hasher les données d'un client

Retour avec id=1 en argument ->

```
+---------------+--------------------------------+--------------------------------+--------------------------------+------------------+
|IdentifiantClient|Nom                           |Prenom                          |Adresse                         |DateDeSouscription|
+---------------+--------------------------------+--------------------------------+--------------------------------+------------------+
|1              |edaae2d0dd3d807a40856298797e56be|83217071f9940e225c6719e6957dfdae|098f6bcd4621d373cade4e832627b4f6|12/12/2022        |
|2              |Clio                            |Dan                             |test                            |12/12/2022        |
|3              |Clio                            |Pet                             |test                            |12/12/2022        |
+---------------+--------------------------------+--------------------------------+--------------------------------+------------------+
```

Des 4 lignes de départ il en reste 3 avec les donnée hasher pour l'id 1 (une ligne a était enlevé car elle ne correspondait pas au schéma)

Pour hdfs :
On n'a pas encore testé de faire le traitement avec des fichiers sur hdfs mais ça fonctionne avec des fichiers en local passé en argument.