Rapport du projet SEPA
sepa-server

par Geoffrey SPAUR et Camille LEPLUMEY

26 avril 2017

Contents

1 Information minimales

1.1 Auteurs

Le projet a été réalisé par Camille LEPLUMEY et Geoffrey SPAUR. Ce projet a pour but de mettre en place une API REST permettant la gestion de transactions bancaires.

1.2 Adresse du service REST

Vous trouverez l'adresse de notre API ici: `https://gscl-sepa.herokuapp.com/`.

2 Description du serveur

2.1 Adresse du service REST

Vous trouverez l'adresse de notre API ici: `https://gscl-sepa.herokuapp.com/`.

2.2 Description précise des requêtes

2.2.1 Resume

Resume permet d'obtenir une courte description des transactions stockées en base.
GET `https://gscl-sepa.herokuapp.com/resume`

Retour :

```
<transactionResumes>
  <transactionResume>
    <date>1992-03-05</date>
    <ident>Example</ident>
    <montant>12322.2999999999992724</montant>
    <num>SL0001</num>
  </transactionResume>
  <transactionResume>
    <date>1992-03-05</date>
    <ident>Example</ident>
    <montant>12322.2999999999992724</montant>
    <num>SL0002</num>
  </transactionResume>
</transactionResumes>
```

2.2.2 Home

Home permet d'obtenir des informations sur le projet tels que le nom des auteurs.
GET `https://gscl-sepa.herokuapp.com/accueil`

Retour :

```
<welcomeXML>
  <auteurs>Geoffrey SPAUR, Camille LEPLUMEY</auteurs>
  <date>30/07/04/2017</date>
</welcomeXML>
```

2.2.3 Reset

Reset Database permet de supprimer et de recréer la base de données.Cette fonction est utilisée afin de supprimer les données corrompues.
GET `https://gscl-sepa.herokuapp.com/reset`

Retour :

```
<message>Database restored</message>
```

2.2.4 Test

Test permet d'ajouter un exemple de transaction dans la base de données.
GET `https://gscl-sepa.herokuapp.com/test`

Retour :

```
<ns2:CstmrDrctDbtInitn>
  <ns2:DrctDbtTxInf>
    <PmtId>Example</PmtId>
    <InstdAmt Ccy="EUR">12322.2999999999992724</InstdAmt>
    <ns2:DrctDbtTx>
      <MndtId>FF-57-E7</MndtId>
      <DtOfSgntr>1992-03-05</DtOfSgntr>
    </ns2:DrctDbtTx>
    <ns2:DbtrAgt>
      <BIC>DAEOFRPPCCT</BIC>
    </ns2:DbtrAgt>
    <ns2:Dbtr>
      <Nm>Mr Dupont</Nm>
    </ns2:Dbtr>
    <ns2:DbtrAcct>
      <IBAN>FE1405643596877A</IBAN>
    </ns2:DbtrAcct>
    <RmtInf>Ceci est un commentaire</RmtInf>
  </ns2:DrctDbtTxInf>
</ns2:CstmrDrctDbtInitn>
```

2.2.5 Stats

Stats permet d'obtenir les statistiques concernant la base de données.
GET `https://gscl-sepa.herokuapp.com/stats`

Retour :

```
<transactionStats>
  <nbOfTransaction>0</nbOfTransaction>
  <totalAmount>0</totalAmount>
</transactionStats>
```

2.2.6 Depot

Depot permet d'ajouter une transaction.
POST `https://gscl-sepa.herokuapp.com/depot`

Paramètres :

```
<ns2:CstmrDrctDbtInitn>
  <ns2:DrctDbtTxInf>
    <PmtId>Example</PmtId>
```

```
        <InstdAmt Ccy="EUR">12322.2999999999992724</InstdAmt>
        <ns2:DrctDbtTx>
          <MndtId>FF-57-E7</MndtId>
          <DtOfSgntr>1992-03-05</DtOfSgntr>
        </ns2:DrctDbtTx>
        <ns2:DbtrAgt>
          <BIC>DAEOFRPPCCT</BIC>
        </ns2:DbtrAgt>
        <ns2:Dbtr>
          <Nm>Mr Dupont</Nm>
        </ns2:Dbtr>
        <ns2:DbtrAcct>
          <IBAN>FE1405643596877A</IBAN>
        </ns2:DbtrAcct>
        <RmtInf>Ceci est un commentaire</RmtInf>
      </ns2:DrctDbtTxInf>
    </ns2:CstmrDrctDbtInitn>
```

Retour :

```
    <ns2:CstmrDrctDbtInitn>
      <ns2:DrctDbtTxInf>
        <PmtId>Example</PmtId>
        <InstdAmt Ccy="EUR">12322.2999999999992724</InstdAmt>
        <ns2:DrctDbtTx>
          <MndtId>FF-57-E7</MndtId>
          <DtOfSgntr>1992-03-05</DtOfSgntr>
        </ns2:DrctDbtTx>
        <ns2:DbtrAgt>
          <BIC>DAEOFRPPCCT</BIC>
        </ns2:DbtrAgt>
        <ns2:Dbtr>
          <Nm>Mr Dupont</Nm>
        </ns2:Dbtr>
        <ns2:DbtrAcct>
          <IBAN>FE1405643596877A</IBAN>
        </ns2:DbtrAcct>
        <RmtInf>Ceci est un commentaire</RmtInf>
      </ns2:DrctDbtTxInf>
    </ns2:CstmrDrctDbtInitn>
```

2.2.7 Transaction

Transaction permet de consulter en détails une transaction.
GET `https://gscl-sepa.herokuapp.com/trx/{id}`

Paramètres :

```
id : int
```

Retour :

```
<ns2:CstmrDrctDbtInitn>
  <ns2:DrctDbtTxInf>
    <PmtId>Example</PmtId>
    <InstdAmt Ccy="EUR">12322.2999999999992724</InstdAmt>
    <ns2:DrctDbtTx>
      <MndtId>FF-57-E7</MndtId>
      <DtOfSgntr>1992-03-05</DtOfSgntr>
    </ns2:DrctDbtTx>
    <ns2:DbtrAgt>
      <BIC>DAEOFRPPCCT</BIC>
    </ns2:DbtrAgt>
    <ns2:Dbtr>
      <Nm>Mr Dupont</Nm>
    </ns2:Dbtr>
    <ns2:DbtrAcct>
      <IBAN>FE1405643596877A</IBAN>
    </ns2:DbtrAcct>
    <RmtInf>Ceci est un commentaire</RmtInf>
  </ns2:DrctDbtTxInf>
</ns2:CstmrDrctDbtInitn>
```

2.2.8 Delete

Delete permet de supprimer une transaction donnée.
DELETE `https://gscl-sepa.herokuapp.com/delete/{id}`

Paramètres :

```
id : int
```

Retour :

```
<message>Transaction {id} deleted</message>
```

2.3 Liste des technologies

Durant ce projet, nous avons utilisé toutes les technologies vues en cours. Cependant il a été nécessaire d'utiliser des technologies avant-gardiste. Notamment pour effectuer la persistance des données et le déploiement de notre service sur une plateforme Cloud.

2.3.1 Déploiement du service

Comme conseillé par notre chargé de TP, nous avons utilisé la plateforme Heroku. Par conséquent nous avons utilisé le CLI afin d'*uploader* notre projet et de le déployer automatiquement. Dans un premier vous devez vous créer et activer un compte sur Heroku. Puis vous pouvez créer une application:

```
$ heroku create <app_name>
```

Nous avons choisi pour nom d'application: gscl-sepa. Notre projet est une application *maven* générant un fichier *war*. Ce fichier peut être déployé sur *tomcat* ou *glassfish*. Afin de générer le fichier *war*, il suffira d'entrer la commande suivante à la racine de notre projet:

```
$ mvn install
```

Le fichier *war* se trouvera dans le dossier *target*.
Dans un premier temps, nous avons vérifié que notre application ne produisait pas d'erreurs en la testant en local sur *glassfish*. Puis nous l'avons déployée sur Heroku dans un *tomcat* comme conseillé par les tutoriels proposés par Heroku. Pour cela nous avons installé le plugin proposé par Heroku:

```
$ heroku plugins:install heroku-cli-deploy
```

Puis nous pouvons déployer notre application:

```
$ heroku war:deploy <path_to_war_file> --app <app_name>
```

Pour finir vous pourrez accéder à l'application avec cette url:
https://<app_name>.herokuapp.com.

2.3.2 Persistance des données

Instanciation de la base Avant de pouvoir sauvegarder nos transactions, nous avons besoin de créer notre base de données. Nous avons pour cela utilisé une application postgresql proposée par Heroku. En créant cette application nous avons eut accès à une database pré-créée. Afin d'effectuer nos tests, nous avons installé un plugin permettant l'accès à notre base de données:

```
$ heroku addons:create heroku-postgresql:hobby-dev
```

Enfin nous nous connectons à la base de données avec la commande suivante:

```
$ heroku pg:psql --app <app_name>
```

Nous avons effectué divers tests: création de tables, insertion de données...
Après nos tests, nous avons créé notre base de données:

```
create table transaction (id serial primary key, data xml);
```

Le type *serial* combine un type entier avec l'option *auto-increment*. Le type *XML* permet de sauvegarder des données sous format XML. Ce dernier type permettra d'utiliser des fonctions XML dans nos requêtes SQL, telles que la fonction *xpath()*.

Lien avec l'application Maintenant que notre base est instanciée, nous pouvons établir une connexion entre cette dernière et notre application. Pour obtenir toutes les informations de connexion, vous pouvez entrer la commande suivante:

```
heroku pg:credentials --app <app_name>
```

Maintenant nous allons configurer notre application pour la connecter à notre base de données. Pour cela il nous faut utiliser un *driver* spécifique pour *postgresql*. Nous utilisons la dépendance suivante (à ajouter dans le fichier *pom.xml*).

```
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>9.4-1201-jdbc4</version>
</dependency>
```

Ensuite pour établir la connexion, il faudra charger le driver puis utiliser nos *credentials*.

```
Class.forName("org.postgresql.Driver");
Connection connection =
    DriverManager.getConnection(url, user, password);
```

Il est à noter que l'url doit être de cette forme:

```
jdbc:postgresql://<host>:<port>/<database>
```

Vous retrouverez ces informations dans le fichier */src/main/java/rest/DAO/ClassDAO.java*.

2.4 Tutoriel de déploiement

Dans cette section, nous allons voir - de manière spécifique - comment déployer notre serveur REST. Dans un premier temps, vous devez vous rendre sur notre dépôt GitHub afin de télécharger les sources du projet:
`https://github.com/Zahco/sepa`
Puis vous devrez générer notre fichier *war*:

```
$ mvn install
```

Libre à vous d'utiliser le service (*tomcat*, *glassfish*) qui vous sied. Pour notre part nous utilisons la plateforme Heroku. Vous devrez donc télécharger et installer le cli, afin de pouvoir vous connecter à la plateforme en notre nom.

```
$ heroku login
Enter your Heroku credentials:
Email: geoffrey.spaur@gmail.com
Password: aqwzsxedc.2017
```

Une fois connecté vous pourrez déployer notre application comme ceci:

```
$ heroku war:deploy target/sepa-server.war --app gscl-sepa
```

Enfin ouvrez notre application dans votre navigateur ou utilisez notre client.

```
$ heroku open
```

Vous trouverez l'adresse de notre API ici: `https://gscl-sepa.herokuapp.com/`.

3 Description du client

3.1 Tutoriel d'installation et d'exécution

Afin d'utiliser notre client vous devrez le télécharger ici:
`https://github.com/Zahco/sepa-client`
Compiler les sources:

```
$ mvn install
```

Exécuter le fichier jar:

```
$ java -jar target/sepa-client.jar
```

3.2 Mode d'emploi

Vous trouverez dans la partie haute du client différents boutons parlant d'eux même.

- *Home*: Pour accéder à divers informations
- *Stat*: Pour afficher les statistiques.
- *Resume*: Pour obtenir un résumé de chaque transactions.
- *Clear*: Pour vider la console.
- *Reset Database*: Pour vider la base de données.

Vous trouverez dans la partie gauche du client un formulaire permettant d'ajouter une transaction personnalisée. Attention, cependant les champs sont soumis à la norme ISO 20022, notamment les champs BIC et IBAN. Vous pourrez envoyer le formulaire en cliquant sur le bouton: *Depot*.

Dans le centre de l'application, vous trouverez la console permettant d'observer les réponses du serveur. Vous observerez parfois des erreurs dans cette console, l'information utile se trouvera en général dans les premières lignes de cette dernière.

Enfin dans la partie basse du client, vous pouvez rechercher une transaction avec son identifiant. Grâce à cette recherche, vous obtiendrez toutes les informations concernant la transaction. Pour obtenir un identifiant, il vous suffira d'obtenir la liste des transactions avec l'option *Resume*. Rechercher l'attribut *transactionResume.num* en retirant les 2 premières lettres. De la même façon, il est possible de supprimer une transaction.
Vous avez aussi la possibilité d'ajouter une transaction *exemple* avec le bouton *Add example*.

3.3 Exemple de fichiers

Vous pouvez ajouter une transaction *exemple* en accédant à l'url suivante:
`https://gscl-sepa.herokuapp.com/test`

4 Fonctions complémentaires

Clear Afin de faciliter la lecture en levant toute ambiguïté, nous avons rajouté un bouton *Clear* dans notre client afin de vider la console.

Reset Database Durant le développement du projet, nous avons utilisé un Git public. Pensant que personne ne s'intéresserait à nous, nous avons librement disposé les accès à la base de données. Seulement certain ont décidé de jouer avec. En ajoutant directement des lignes en base de données, cela corrompt le modèle XML. Nous avons ajouté le bouton *Rest Database* pour nettoyer la base de données. Nous avons par-ailleurs caché les mots de passe...

Delete La fonction delete permet de consommer une transaction dans la base de données (cf: sujet du projet). L'identifiant correspond à la colonne id en bise de données. Vous retrouverez cet identifiant avec la commande *resume*, en recherchant l'attribut *transactionResume.num* et en retirant les 2 premières lettres.