

Rapport du projet SEPA

sepa-server

par Geoffrey SPAUR et Camille LEPLUMEY

26 avril 2017

Contents

1	Information minimaux	3
1.1	Auteurs	3
1.2	Adresse du service REST	3
2	Description du serveur	4
2.1	Adresse du service REST	4
2.2	Description précise des requêtes	4
2.2.1	Resume	4
2.2.2	Home	4
2.2.3	Reset	4
2.2.4	Test	5
2.2.5	Stats	5
2.2.6	Depot	5
2.2.7	Transaction	6
2.3	Liste des technologies	7
2.3.1	Déploiement du service	7
2.3.2	Persistance des données	8
2.4	Tutoriel de déploiement	9
3	Description du client	10
3.1	Tutoriel d'installation et d'exécution	10
3.2	Mode d'emploi	10
3.3	Exemple de fichiers	10
4	Fonctions complémentaires	11

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://gsc1-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>12322.29999999999992724</montant>
    <num>SL0001</num>
  </transactionResume>
  <transactionResume>
    <date>1992-03-05</date>
    <ident>Example</ident>
    <montant>12322.29999999999992724</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 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 jeu d'exemple dans la base de données.

GET <https://gscl-sepa.herokuapp.com/test>

Retour :

```
<ns2:CstmrDrctDbtInitn>
  <ns2:DrctDbtTxInf>
    <PmtId>Example</PmtId>
    <InstdAmt Ccy="EUR">12322.29999999999992724</InstdAmt>
    <ns2:DrctDbtTx>
      <MndtId>FF-57-E7</MndtId>
      <DtOfSgntr>1992-03-05</DtOfSgntr>
    </ns2:DrctDbtTx>
    <ns2:DbtrAgt>
      <BIC>DAE0FRPPCCT</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:DrectDbtTx>
  <MndtId>FF-57-E7</MndtId>
  <DtOfSgntr>1992-03-05</DtOfSgntr>
</ns2:DrectDbtTx>
<ns2:DbtrAgt>
  <BIC>DAEOFPPCCT</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:DrectDbtTxInf>
</ns2:CstmrDrectDbtInitn>
```

Retour :

```
<ns2:CstmrDrectDbtInitn>
  <ns2:DrectDbtTxInf>
    <PmtId>Example</PmtId>
    <InstdAmt Ccy="EUR">12322.2999999999992724</InstdAmt>
    <ns2:DrectDbtTx>
      <MndtId>FF-57-E7</MndtId>
      <DtOfSgntr>1992-03-05</DtOfSgntr>
    </ns2:DrectDbtTx>
    <ns2:DbtrAgt>
      <BIC>DAEOFPPCCT</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:DrectDbtTxInf>
</ns2:CstmrDrectDbtInitn>
```

2.2.7 Transaction

Transaction permet de consulter en détail une transaction.

GET <https://gsc1-sepa.herokuapp.com/trx/{id}>

Paramètres :

id : int

Retour :

```
<ns2:CstmrDrctDbtInitn>
  <ns2:DrctDbtTxInf>
    <PmtId>Example</PmtId>
    <InstdAmt Ccy="EUR">12322.29999999999992724</InstdAmt>
    <ns2:DrctDbtTx>
      <MndtId>FF-57-E7</MndtId>
      <DtOfSgntr>1992-03-05</DtOfSgntr>
    </ns2:DrctDbtTx>
    <ns2:DbtrAgt>
      <BIC>DAE0FRPPCCT</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>
```

2.3 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: gscI-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'entré 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'erreur en la testant en local sur glassfish. Puis nous l'avons déployer sur Heroku dans un tomcat comme conseiller par les tutoriels proposé par Heroku. Pour cela nous avons installer le plugins 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 votre application avec cette url:

`https://<app_name>.herokuapp.com.`

2.3.2 Persistance des données

Instanciation de la base Avant de pouvoir sauvegarder nos transaction, nous avons besoin de créer notre base de données. Nous avons pour cela utiliser une application postgresql proposé par Heroku. En créant cette application nous avons au accès à une database pré créée. Afin d'effectuer nos tests, nous avons installer un plugins permettant d'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 effectuer 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>
```


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 connecter vous pourrez déployer notre application comme ceci:

```
$ heroku war:deploy target/sepa-server.war --app gsc1-sepa
```

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

```
$ heroku open
```

Vous trouverez l'adresse de notre API ici: <https://gsc1-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.

Enfin dans la partie basse du client, vous pouvez rechercher une transaction pour 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 le champs *transactionResume.num* en retirant les 2 premières lettres.

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 utiliser un Git public. Pensant que personne ne s'intéresserez 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 corromps le modèle XML. Nous avons ajouter le bouton *Rest Database* pour nettoyer la base de données. Nous avons par-ailleurs caché les mots de passe...