30/04/2017

RAPPORT PROJET WEB2

CREATION D'UN SERVICE REST POUR LES TRANSACTIONS SEPA

REALISER PAR :
-TIDAF Juba

A L'ATTENTION DE :
M. S. Hérauville

Le plan du rapport

Serveur SEPA :

1-lien de connexion en ligne et en local
2- l’architecture et les technologies utilisées
3-l’utilisation sur une machine local

Client SEPA :

1-lien de connexion en ligne et en local
2- l’architecture et les technologies utilisées
3-l’utilisation sur une machine local

ce qui reste a faire.

Les sources d’informations utilisée :

SERVEUR SEPA :

1- Lien de connexion :

En local : http://localhost:8080/sepatrans/

En ligne :

2- L'architecture et les technologies utilisées :

J'ai utilisée l'architecture Spring MVC comme vu au cours et au TP, j'ai aussi utilisée le service REST pour retourner les donnée à une requête demandé.

Le contenu du projet :

J'ai utilisé quatre (4) packages pour la configuration :

Package sepaTransConfiguration : contient la classe java ConfigurationBean.java, elle contient les Bean de configuration pour par exemple de spécifié le chemin des ressources a utilisé, le chemin des vue, la connexion à la base de données et l'instanciation des objets dao qui implémente toutes les fonctionnalités sur la base de données

Package sepaTransModel : contient les classes nécessaire pour créer une transaction, j'ai pris le même model utilisé en TP6, sauf que ici les objets on les a sérialisé en XML (pour le service REST).

J'ai créé l'objet sepaTransResume.java pour retourner un objet qui correspond à l'appel de requête /resume, même chose pour sepaTransStats.java et setaTransDetail.java

Package sepaTransController : contient une classe sepaController.java, c'est la même classe de TP que j'ai utilisé sauf qu'ici elle contient les routes vers la page d'accueil qui est « home.jsp », et l'ensemble des fonctionnalités demandé dans le projet.

Package sepaTransValid : contient une classe sepaValidationXML.java, je l'ai pris des tps réalisés précédemment, elle

consiste a validé un document XML en dépend du fichier sepa.XSD qui se trouve dans le répertoire ressource du projet.

Package sepaTransImplementation : contient la classe TransactionImpl.java, ce dernier contient toutes les requêtes vers la base du données (fonctionnalités qui correspond à au projet). J'ai défini cinq (5) fonction : ajouter une transaction, afficher toutes les transactions, afficher les stats des transactions, afficher le résumé des transactions, rechercher une transaction par son Id.

Y a le package view qui se trouve dans le répertoire WEB-INF, contient une page JSP home.jsp, qui représentes une liste de l'ensemble des fonctionnalités demandé avec leur description

Persistance :

On local j'ai utilisé MySQL, le fichier SQL se trouve dans répertoire ressource du projet.

Avec nom utilisateur : **« root »** et mot de passe : **«¨¨»(pas de mot de passe)**

La base de données contient une seule table qui est « transaction ». Pour manipuler la base de données j'ai utilisé les objets de type JdbcTemplate.

```sql
CREATE DATABASE sepatransactiondb;

 USE sepatransactiondb;

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

CREATE TABLE `sepatransactiontable` (
  `id_sepa_transaction` int(20) NOT NULL,
  `numero` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `paieid` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `instdamt` double NOT NULL,
  `mandatid` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `datetrans` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `bic` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `nom` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `iban` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `relateinf` varchar(50) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO `sepatransactiontable` (`id_sepa_transaction`, `numero`, `paieid`, `instdamt`, `mandatid`, `datetrans`, `b
(1, 'TJ0001', 'REF OPE IIII', 800, 'MANDAT NO 141414', '2004-01-01', 'ABNAFRPP', 'debiteur 1', 'FR7630001007941678901
(2, 'TJ0002', 'REF OPE AAAA', 950, 'MANDAT NO 178112', '2007-12-03', 'AECFFR21', 'debiteur 2', 'FR7630004000031678901
(3, 'TJ0003', 'REF OPE CCCC', 550, 'MANDAT NO 142113', '2008-09-18', 'AFRIFRPP', 'debiteur 3', 'FR7630006000011234567
(4, 'TJ0004', 'REF OPE DDDD', 500, 'MANDAT NO 222114', '2006-06-10', 'AGFBFRCC', 'debiteur 4', 'FR7610107001078901291
(5, 'TJ0005', 'REF OPE LLLL', 8000.98, 'MANDAT NO 789115', '2009-08-07', 'AGRIFRPI', 'debiteur 5', 'FR761131500045678
(6, 'TJ0006', 'REF OPE FFFF', 705.89, 'MANDAT NO 121116', '2011-11-10', 'AGRIFRPP', 'debiteur 6', 'FR7630002032536789
(7, 'TJ0007', 'REF OPE EEEE', 789.89, 'MANDAT NO 112117', '2012-08-22', 'AGRIMQMX', 'debiteur 7', 'FR7630056009271678
(8, 'TJ0008', 'REF OPE GGGG', 111.04, 'MANDAT NO 112118', '2012-01-04', 'AGRIRERX', 'debiteur 8', 'FR7611808009101789


ALTER TABLE `sepatransactiontable`
  ADD PRIMARY KEY (`id sepa transaction`);
```

Figure : représente la base de données utilisé.

3- L’utilisation sur une machine local :

Premièrement faut installée la base de données, et lancé le projet sur Eclipse et le déployé sur le serveur Tomcat(ou bien générer le .war et le copié sur tomcat). Après le démarrage en tapant sur la barre de recherche du navigateur le lien suivant http://localhost:8080/sepatrans/ . on va avoir un affichage qui ressemble à l’image suivante :

Figure : représente la page d’accueil

4- Déploiement sur Heroku :

Client SEPA :

J'ai fourni un projet **sepatransclient** qui contient une interface web pour communiquer avec le serveur SEPA. C'est un projet Maven, pour son utilisation se fait avec le déploiement sur Tomcat en local, l'adresse à taper sur la barre de recherche est :

http://localhost:8080/clientsepatrans/

1- L'architecture de projet :

J'ai utilisé Spring MVC, avec deux package : Package de configuration des Beans et package controller.

La classe sepaControllerClient.java de package sepaControllerClient contient toutes les routes de tous les url, elle s'occupe de la redirection et de cheminement vers les pages JSP correspondante a une requête donnée.

Chaque méthode déclaré elle se connecte à l'adresse de serveur : http://localhost:8080/sepatrans/ et elle a comme retour un ModelAndView qui sont des objets crée pour spécifier la page JSP qui va s'occupé de l'affichage de l'information reçu, dans notre cas on reçoit un flux (document) XML de la part de serveur.

Sur le côté JSP il va parcourir le document reçu et va extraire les donnée par la fonction **getElementsByTagName(« nom utilisé dans le serveur pour parser les objets java en xml»)** et renvoyé de l'HTML dans mon cas c'est un simple tableau d'affichage.

Pour les vues, on trouvera dans le répertoire WEB-INF un répertoire nommé **views** il contient toute les vues dans laquelle le contrôleur peut nous rediriger.

2- L'utilisation du client :

On local :

Généré le fichier **.war** puis le copier sur le répertoire webapp de tomcat, et taper dans la barre de recherche du navigateur l'adresse suivante :

http://localhost:8080/clientsepatrans/

On aura comme aperçus en premier lieu un affichage général de l'ensemble des fonctionnalités que le serveur nous fournit.

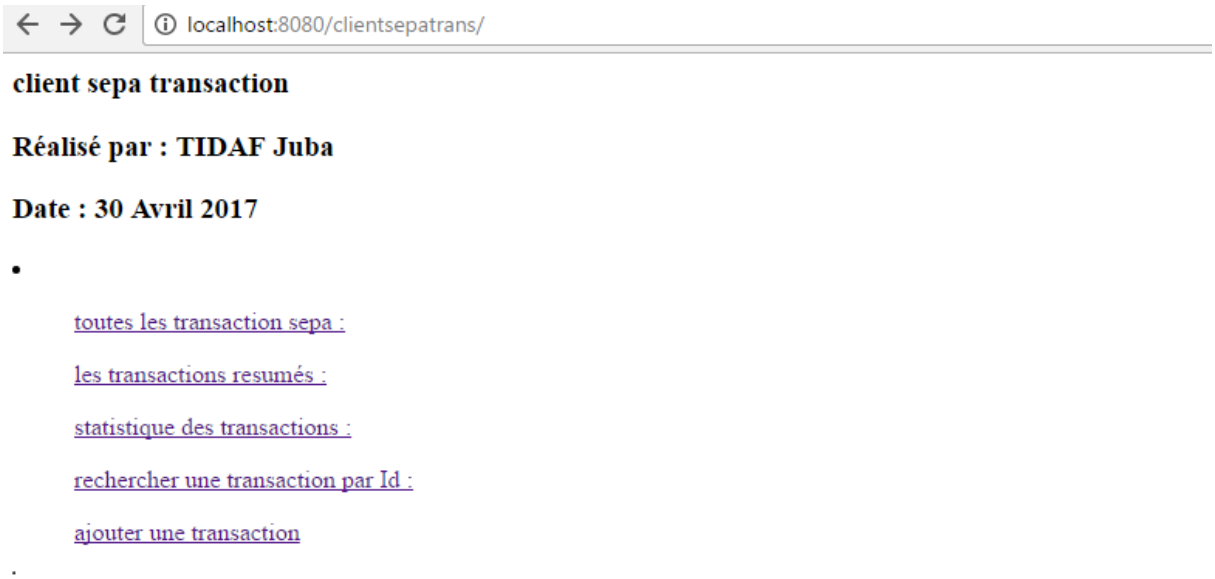
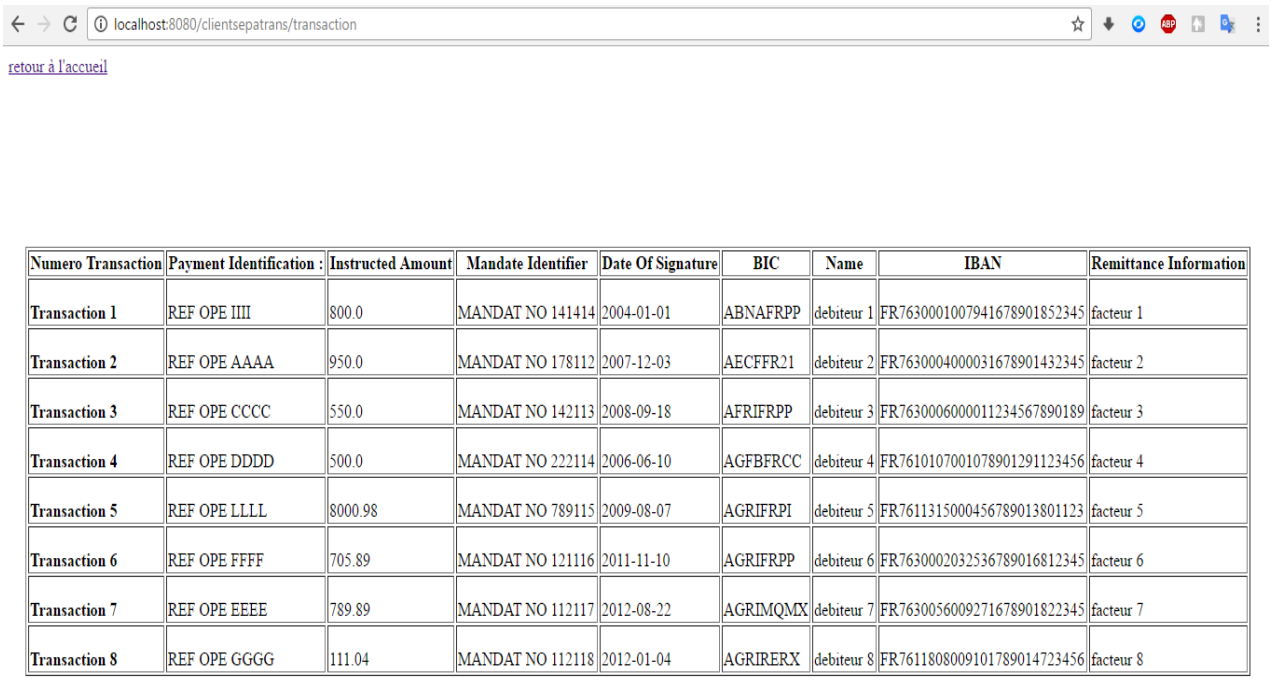


Figure : représente la page d'accueil client.

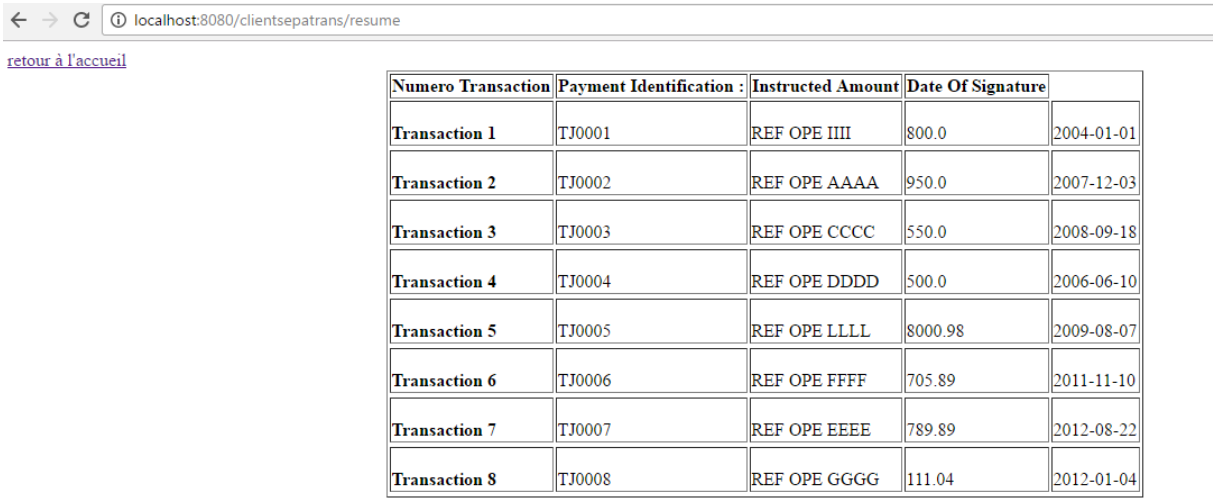
-afficher toutes les transactions :



Numero Transaction	Payment Identification :	Instructed Amount	Mandate Identifier	Date Of Signature	BIC	Name	IBAN	Remittance Information
Transaction 1	REF OPE IIII	800.0	MANDAT NO 141414	2004-01-01	ABNAFRPP	debiteur 1	FR7630001007941678901852345	facteur 1
Transaction 2	REF OPE AAAA	950.0	MANDAT NO 178112	2007-12-03	AECFFR21	debiteur 2	FR7630004000031678901432345	facteur 2
Transaction 3	REF OPE CCCC	550.0	MANDAT NO 142113	2008-09-18	AFRIFRPP	debiteur 3	FR7630006000011234567890189	facteur 3
Transaction 4	REF OPE DDDD	500.0	MANDAT NO 222114	2006-06-10	AGFBFRCC	debiteur 4	FR7610107001078901291123456	facteur 4
Transaction 5	REF OPE LLLL	8000.98	MANDAT NO 789115	2009-08-07	AGRIFRPI	debiteur 5	FR7611315000456789013801123	facteur 5
Transaction 6	REF OPE FFFF	705.89	MANDAT NO 121116	2011-11-10	AGRIFRPP	debiteur 6	FR7630002032536789016812345	facteur 6
Transaction 7	REF OPE EEEE	789.89	MANDAT NO 112117	2012-08-22	AGRIMQMX	debiteur 7	FR7630056009271678901822345	facteur 7
Transaction 8	REF OPE GGGG	111.04	MANDAT NO 112118	2012-01-04	AGRIRERX	debiteur 8	FR7611808009101789014723456	facteur 8

Figure : représente l'ensemble des transactions détaillée.

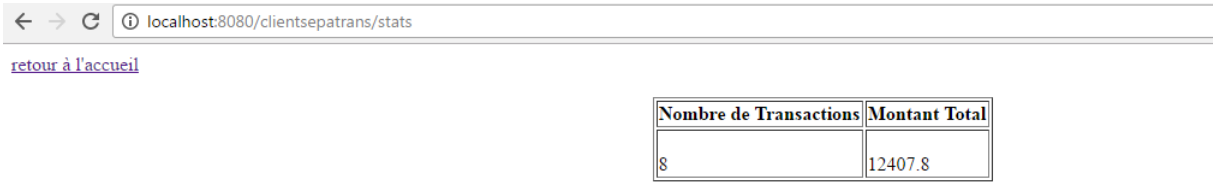
-afficher les transactions résumées :



Numero Transaction	Payment Identification :	Instructed Amount	Date Of Signature	
Transaction 1	TJ0001	REF OPE IIII	800.0	2004-01-01
Transaction 2	TJ0002	REF OPE AAAA	950.0	2007-12-03
Transaction 3	TJ0003	REF OPE CCCC	550.0	2008-09-18
Transaction 4	TJ0004	REF OPE DDDD	500.0	2006-06-10
Transaction 5	TJ0005	REF OPE LLLL	8000.98	2009-08-07
Transaction 6	TJ0006	REF OPE FFFF	705.89	2011-11-10
Transaction 7	TJ0007	REF OPE EEEE	789.89	2012-08-22
Transaction 8	TJ0008	REF OPE GGGG	111.04	2012-01-04

Figure : représente le résumé des transactions.

-afficher les states des transactions :



Nombre de Transactions	Montant Total
8	12407.8

Figure : représente les statistiques de l’ensemble des transactions.

-rechercher une transaction par son Id :

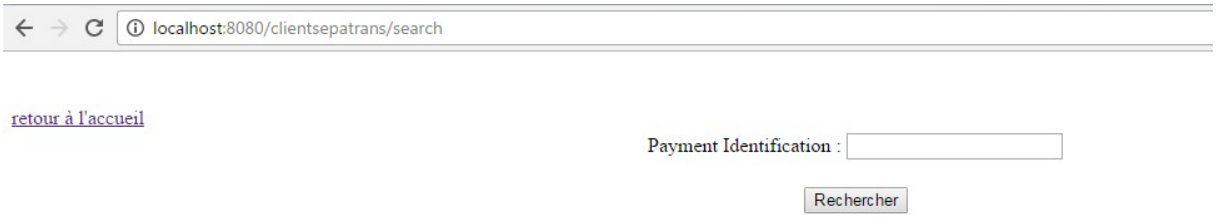


Figure : représente interface de recherche d’une transaction par son id.

-en cas d‘Id inexistant : en aura un message suivant :

Figure : représente le message en cas de recherche d’un Id inexistant dans la BDD.

-en cas le Id existe :

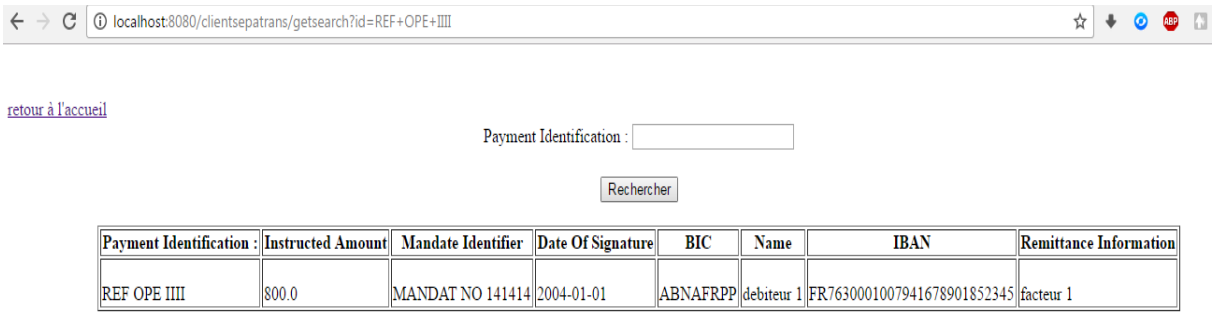


Figure : représente les détail d'une transaction.

-ajouter une transaction :

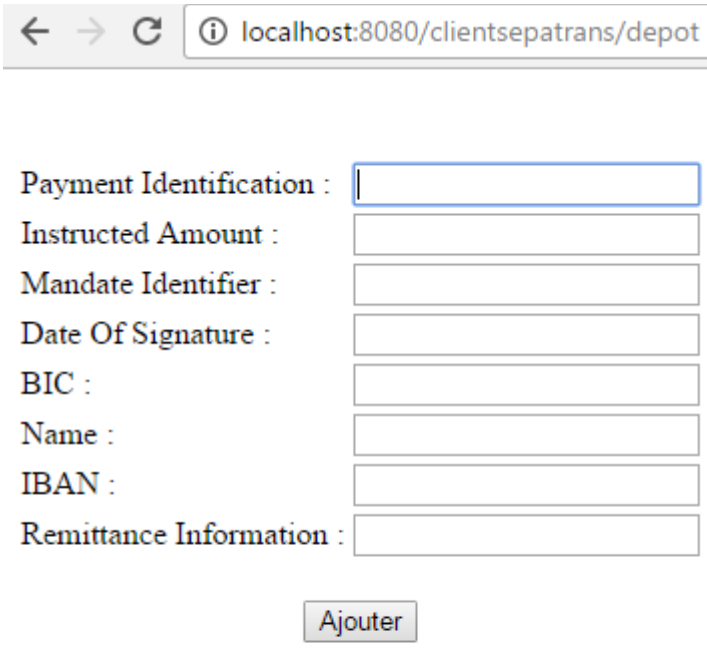


Figure : formulaire d'ajout d'une transaction.

Cas de saisie d'information erroné :

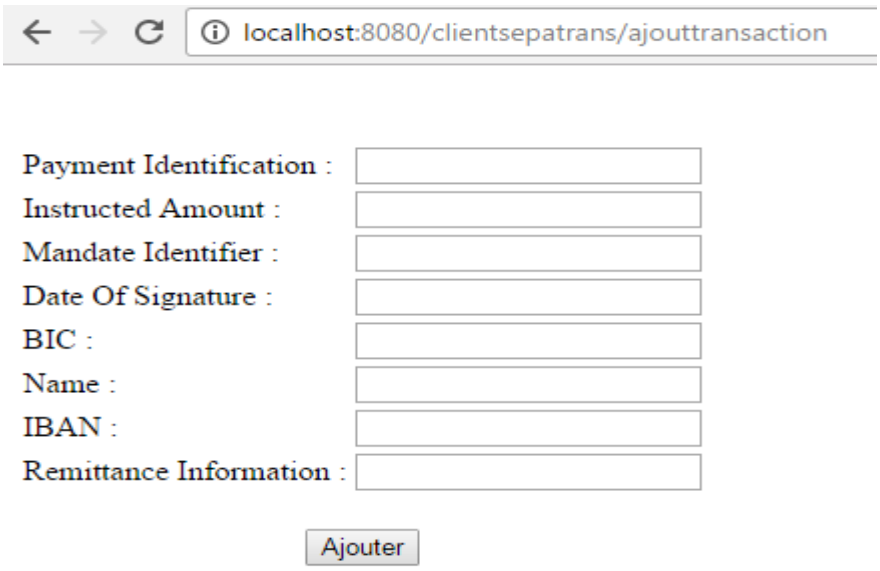


Figure : représente le message d'erreur en cas de saisie de donnée erroné.

Cas de saisie d'une transaction avec id qui existe déjà :

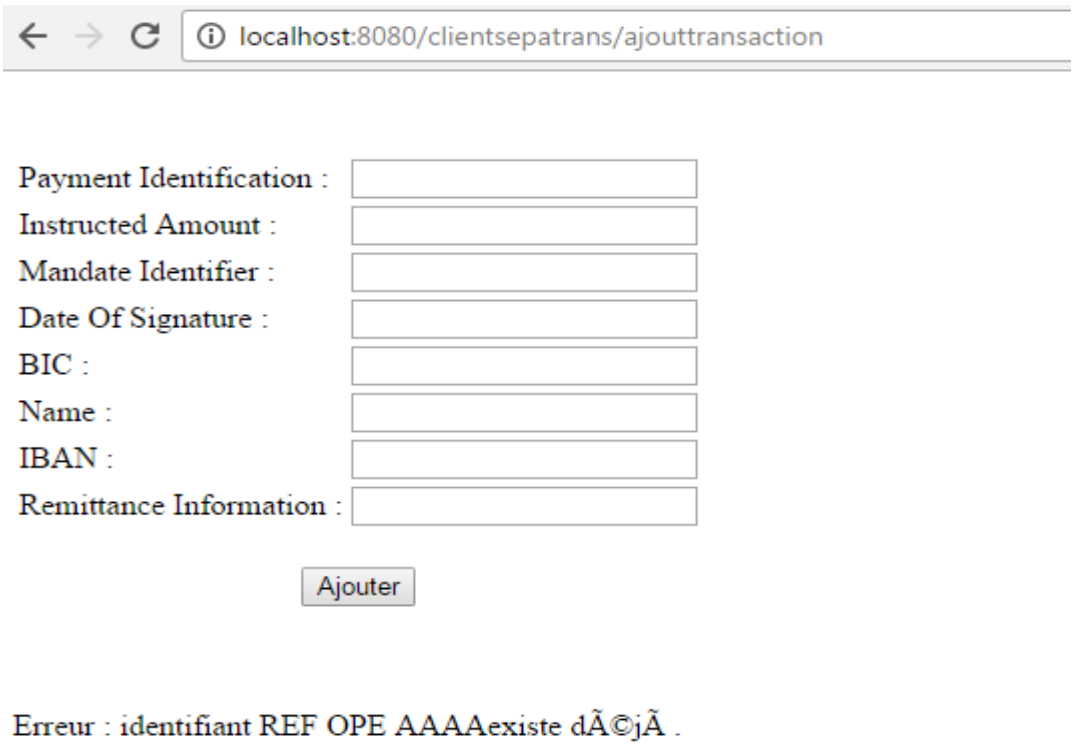


Figure : représente le message en cas de saisie d'un identifiant existant dans la BDD.

En cas de succès :

Figure : en cas d'enregistrement dans la BDD avec succès.

Ce qui reste a faire :

Avec le manque du temps que j'ai pas su gérer il me rester les fonctionnalités suivantes :

- deploiment de serveur sur heroku.

- deposer un fichier XML (pour ajouter une nouvelle transaction).

Source d'information utilisé :

-Tous les TPs fait durant le semestre et les cours.

-Documentation officiel Du SPRING

-j'ai utilisé le lien suivant pour bien manipuler la base du donnée sous SPRING MVC :

http://www.codejava.net/frameworks/spring/spring-mvc-with-jdbctemplate-example

-j'ai utilisé les Tps de module Architectures Distribuées de cette année, pour s'inspirer du service REST.

-Et notament les autres sites sur internet et les blogs
-stackoverflow …etc