**2013**

Réseau social de type Twitter

Fabien Belli & Suzy Paeta



Sommaire

[1.Introduction 3](#__RefHeading__259_1272585474)

[2.Fonctionnalités de l’application 3](#__RefHeading__261_1272585474)

[3.Choix d’implémentation 4](#__RefHeading__263_1272585474)

[a.Noyau de l’application 4](#__RefHeading__265_1272585474)

[b.La sécurité 4](#__RefHeading__267_1272585474)

[c.Les images 4](#__RefHeading__269_1272585474)

[d.Sérialisation des Personnes 4](#__RefHeading__271_1272585474)

[e.Interface graphique 4](#__RefHeading__273_1272585474)

[4.Diagramme de classe 5](#__RefHeading__275_1272585474)

[5.Diagramme de séquence 6](#__RefHeading__277_1272585474)

[6.Les difficultés rencontrées 6](#__RefHeading__279_1272585474)

[7.Lancer l’application 6](#__RefHeading__281_1272585474)

[8.Conclusion 7](#__RefHeading__283_1272585474)

1. Introduction

Le but de ce projet est de développer une application type Twitter.

1. Fonctionnalités de l’application

Le client a accès à des fonctionnalités différentes s’il est authentifié ou non.

En mode non authentifié, le client peut :

* **lister les x derniers tweets à propos d’un topic**
* **lister les y derniers tweets émis, crées ou relayés par un compte utilisateur**
* **lister les topics existants**
* **lister les utilisateurs inscrits**

Un client peut **s’inscrire** et **se connecter** à son compte Twitter pour devenir un client authentifié dans le système. A la fin il peut bien sûr **se déconnecter**.

En mode authentifié, le client peut faire les mêmes actions que le client non authentifié et :

* **écrire un tweet**
* **écrire un tweet et y joindre une image**
* **relayer un tweet**
* **follower un utilisateur**
* **afficher son fil d’actualité** (afficher les tweets des personnes que le client follow)

1. Choix d’implémentation
   1. Introduction

Pour ce projet, nous avons choisit de mettre en place une architecture client/serveur en utilisant l’API Java RMI. Le serveur est un objet distant qui gère les utilisateurs et centralise les données (tweets, comptes utilisateurs...). C’est lui qui offre des fonctionnalités aux clients. Ceux-ci grâce à RMI peuvent invoquer des méthodes du serveur à distance (ex : se connecter, écrire un tweet…).

Le client est lui aussi un objet distant et *extends* de *UnicastRemoteObject* afin que le serveur puisse obtenir une référence sur le client et donc appeler ses méthodes (ex : ).

Le client récupère un stub vers un objet distant côté serveur qui est différent selon qu’il soit connecté ou non. Pour cela nous avons créée deux interfaces nommées InterfacePrivee et InterfacePublic. Lorsqu’il n’est pas authentifié, le client n’a accès qu’aux méthodes de InterfacePublic, et lorsqu’il est connecté il a accès aux deux interfaces, il peut donc accéder à toutes les fonctionnalités de Twitter.

* 1. Le serveur

Le serveur est charger de la sauvegarde des informations, il aurait été aussi possible de les sauvegarder dans une base de donnée type MySql ou Oracle.

Le serveur centralise ainsi les données, il contient une liste de Twitt, une liste de personnes, une liste de topic, une liste des followers et enfin une liste de Subject.

Les listes de Twitt et de personnes sont sauvegardées sous forme de fichier pour assurer la persistance des données. Ainsi à chaque lancement, inscription, ajout d'un twitt et fermeture du serveur, les données présentent dans les fichiers sont mise à jour.

Le serveur implémente deux interfaces :

* Une interface public qui contient toutes les méthodes d'accès public aux twitt
* Une interface privée qui contient des méthodes que peuvent uniquement exécuter des personnes connectées.

Le serveur inscrit dans un annuaire l'interface public pour que n'importe quel client puisse y avoir accès. Par contre l'interface privée ne sera délivrée qu'après une authentification réussie auprès du module JAAS et par retour de méthode.

* 1. Les clients

Comme dit précédemment les clients n'ont accès qu'a l'interface public, il peuvent en invoquer les méthodes et interagir avec le serveur.

Lors de la demande de suivis d'un utilisateur, le client envoi en même temps une référence sur lui-même afin que le serveur puisse appeler des méthodes dispensée par une interface chez le client. Ainsi le serveur peut invoquer une méthode du client et donc lui communiquer des informations, comme par exemple les nouveaux Twitt.

Nous avons fait le choix que les clients contiennent une liste de twitt, cette liste correspond aux Twitts que le client recevra du serveur après qu'un de ces followers ai publié un Twitt. De ce fait le client ne peut voir les twitt que lors ce qu'il est connecté et ne verra les twitts de personnes qu'après avoir fait la demande de suivis.

* 1. La sécurité avec JAAS

L'authentification et les autorisations sont gérées par JAAS.

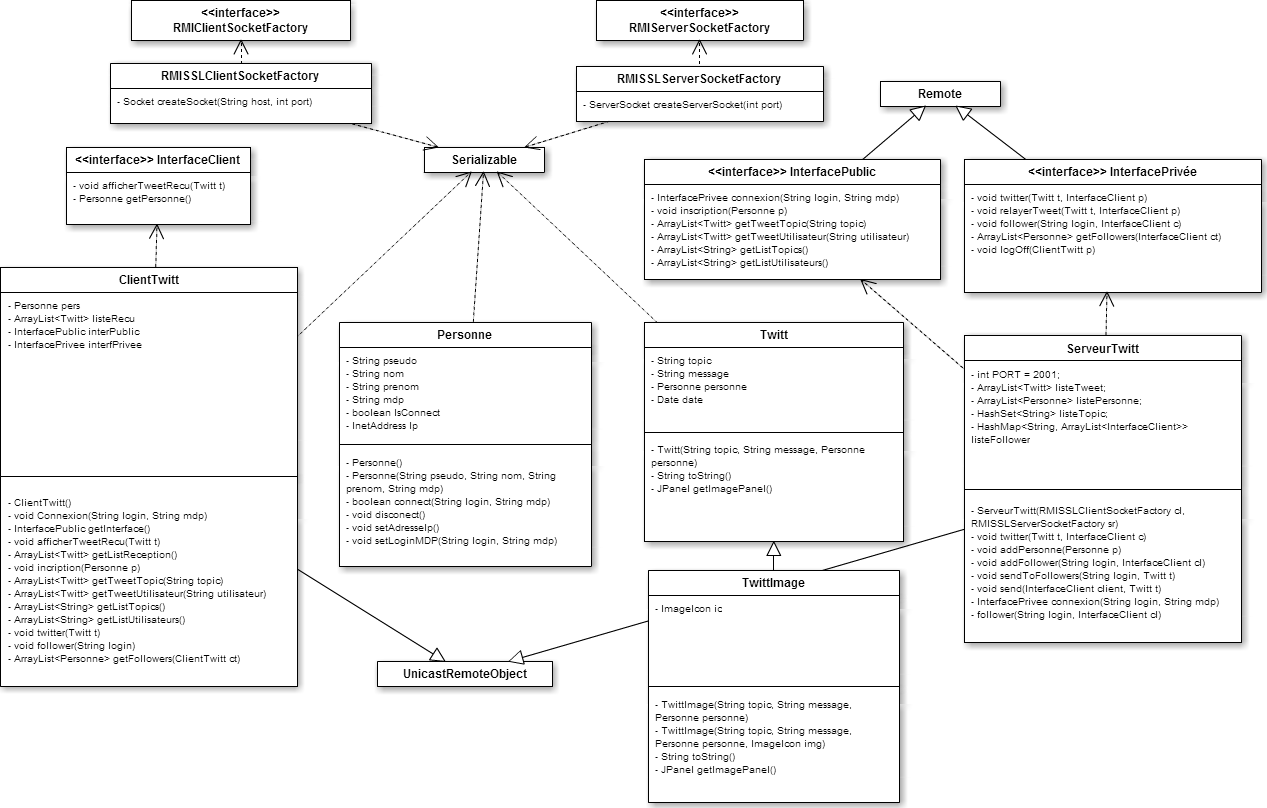
Ce module retourne un Subject après une connexion réussite du client. Le subject contient un principal (classe permettant d'identifier un client unique) ainsi qu'un numéro aléatoire unique pour chaque connexion. Lors ce que ce subject est crée, le serveur le sauvegarde dans une liste puis le transmet par retour de méthode au client appelant.

A chaque appel de méthode, le client doit fournie le Subject le concernant, le serveur vérifie alors sa présence dans sa liste et agit en conséquence.

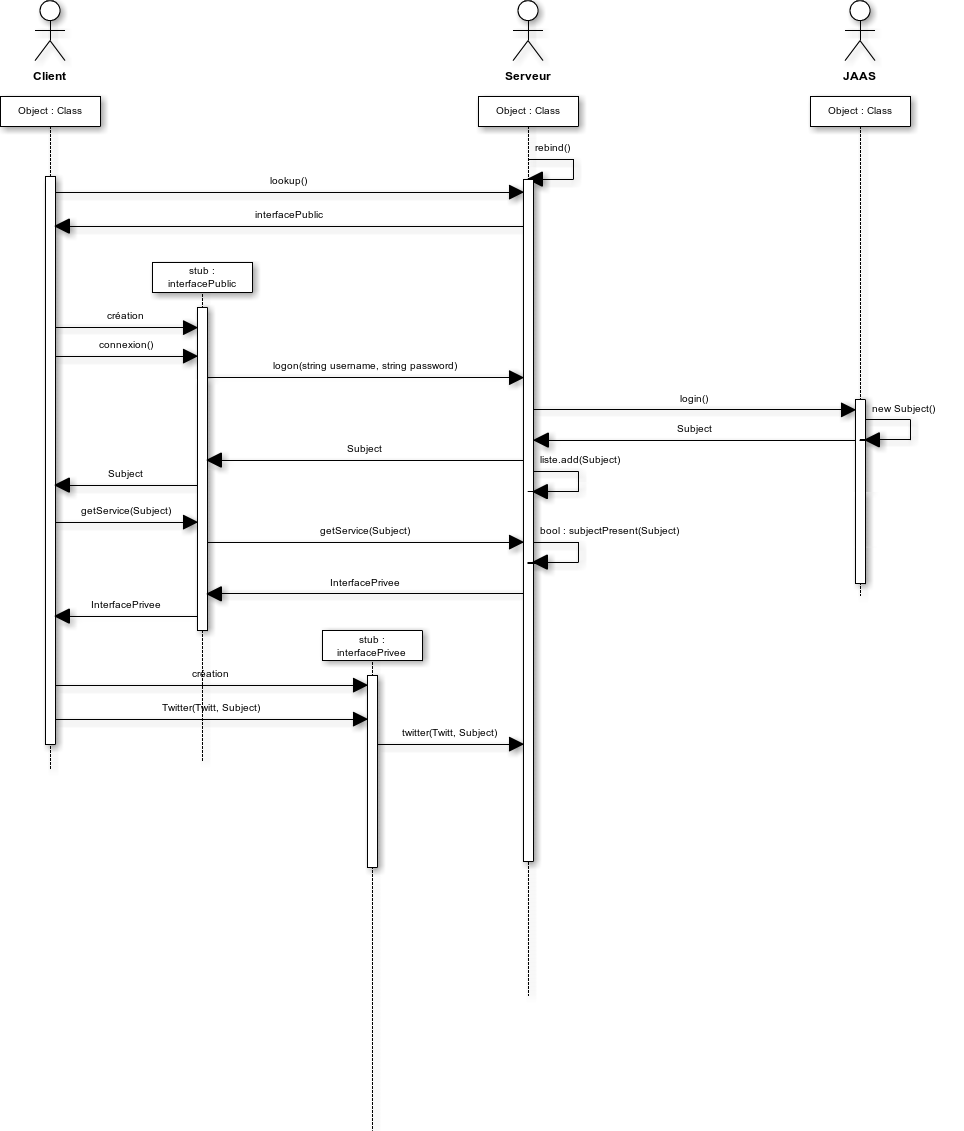
* 1. Les images
  2. Sérialisation des Personnes
  3. Interface graphique

Même si l’interface graphique n’était pas vraiment demandée, nous en avons tout de même implémenté une, surtout pour faciliter les tests des méthodes et rendre l’application plus agréable. En revanche elle n’est pas complètement ergonomique ni esthétique.

1. Diagramme de classe



1. Diagramme de séquence



1. Les difficultés rencontrées
2. Lancer l’application

L’application présente une interface graphique minimaliste afin de pouvoir tester toutes les fonctionnalités implémentées. Il faut cependant lancer le serveur et les clients en console comme décrit ci-dessous :

1. Conclusion

….

Vous pourrez aussi remarquer l’effort particulier que nous avons fournit pour commenter notre code.

1. pourquoi on a une classe personne qui est dans le package Serveur ?

la classe personne permet d'identifier un client par son nom, prenom, mdp...

elle se trouve normalement dans le client et le serveur, c'est mieux que de faire un HashMap contenant login et mdp, enfin je trouvais...

 2) A quoi servent les classes RMISSLServerSocketFactory  et RMISSLClientSocketFactory

ces dernières servent pour des  connexions SSL securisés (fin du TP 4)

3) a quoi ça sert d'avoir 2 interfaces, une privée et une publique

**Fabien:** C'est la même chose que pour Jaas, quand le login a fonctionné, il retourne un personal que le client est le seul a avoir, ce personal lui donne des privilèges et il ne peut executer que les métthodes qui contiennent doAsPrivilège

**moi:** d'accord mais j'ai pas fait jaas encore lol

oui par contre je suis toujours sceptique pour la classe personne ^^

pour toi c'est une personne qui a un compte twitter ?

**Fabien:** Oui c'est comme si tu mettais un login, tu es d'accord que c'est une personne qui emet un twitt

**moi:** en fait j'ai du mal a savoir pourquoi on a un clienttwitt ET une personne

**Fabien:** je ne peut pas stocker des references (ClientTwitt) car elle ne seraient jamais à jour quand le client se connect et se deconnect