IUT Paris Descartes

2018-2019

# Applications Réflexives Java

## Projet Plateforme Dynamique
## Brette-Rossit Initiative


Maud GELLEE — Groupe 201

Arsène LAPOSTOLET — Groupe 202

# Présentation de l'application

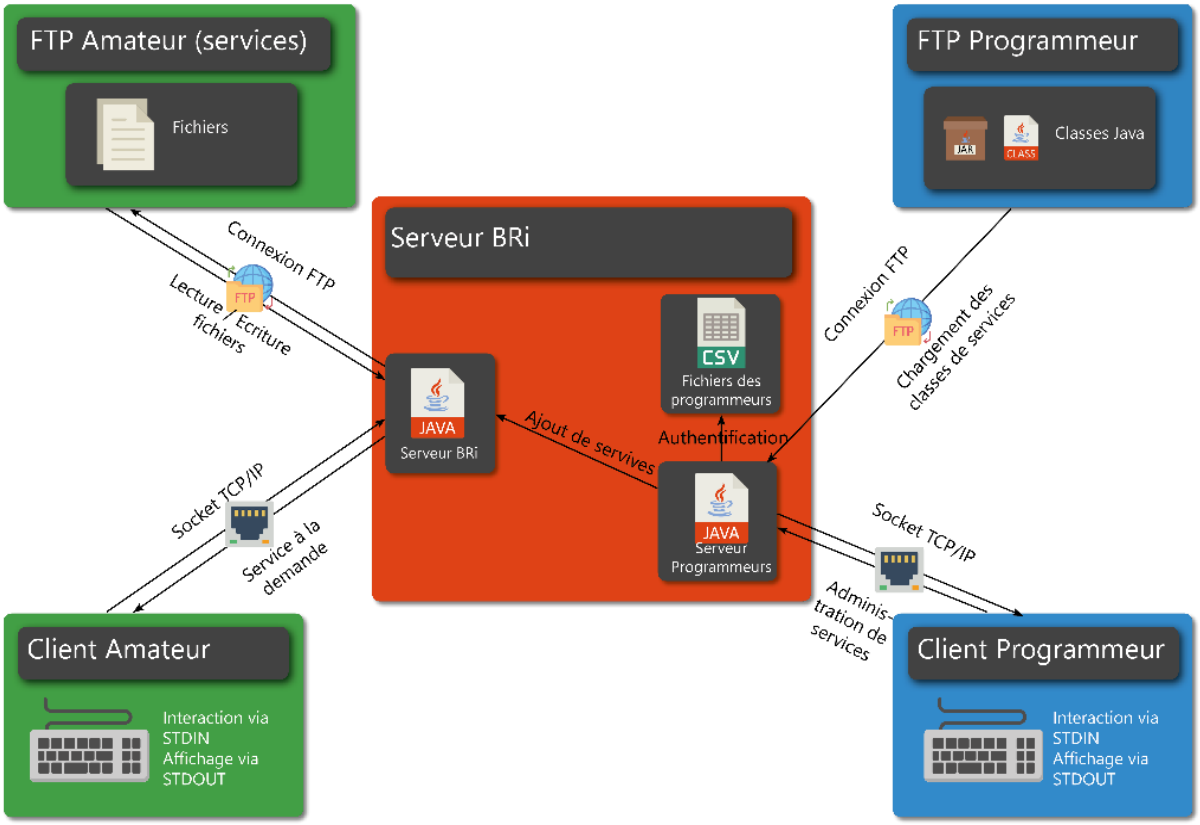
L'application réalisée offre un lieu d'échanges entre des amateurs voulant utiliser des services Java et des programmeurs ayant développé ces services, en respectant la spécification BRi.

Les programmeurs sont authentifiés par un login et un mot de passe. Les amateurs sont en accès libre. Certains services peuvent demander une authentification pour les amateurs.

Pour le projet, nous avons développé plusieurs services : un service d'inversion de chaine de caractères, un service de messagerie et un service d'analyse de fichier XML. Afin de pouvoir mettre à disposition les deux derniers services (qui ne se limitent pas à une classe), nous avons rendu possible l'import de bibliothèques .jar.

Il est aussi possible pour un programmeur d'arrêter, de redémarrer et de désinstaller un service, s'il en est le propriétaire.

# Architecture du projet




## Concurrence

Pour gérer les accès concurrents aux ressources partagées (la liste des services), nous avons stocké les services en pause et les services démarrés dans deux synchronizedList de la bibliothèque standard. Ainsi tout accès à un service, tout ajout et toute suppression est thread safe.

De la même manière nous avons stocké les utilisateurs dans une synchronizedList. Nous stockons les utilisateurs dans un fichier .csv et remplis-

sons la liste lors du démarrage du serveur. Comme nous utilisons une liste synchronisée, tout ajout d'un utilisateur est thread safe, et si nous voulions implémenter les suppressions d'utilisateur, la liste le gèrerait automatiquement.

### Persistance

Pour gagner du temps lors des tests nous avons choisi de sauvegarder les utilisateurs programmeurs. Ainsi, pour arrêter le serveur, il faut lui envoyer la commande *stop* ce qui va déclencher l'écriture des utilisateurs dans un fichier .csv en cryptant les mots de passe. Au lancement du serveur, les utilisateurs sont lus dans le fichier de la même façon.

### Chargement dynamique

Les programmeurs peuvent charger dynamiquement leurs services sur le serveur BRi depuis leur serveur FTP, et ce, sous la forme de fichier .class ou bien d'archives .jar contenant plusieurs classes.

L'import de fichier .class se fait grâce à un *URLClassLoader*. Ce dernier est stocké et initialisé dans le Serveur Programmeur afin de gagner en performances. Il est cependant nécessaire de le réinstancier à chaque ajout de programmeur pour que leurs serveurs FTP soient ajoutés.

Nous avons réussi à implémenter la mise à jour de service en créant une classe *ClassUpdater* étendant *ClassLoader* qui redéfinit la méthode *loadClass()*, il fait appel à la méthode *findClass()* pour le service BRi qu'il faut mettre à jour, et à la méthode *super.loadClass()* pour les autres classes.

# Services de base

## Service Texte

Simple traitement sur le texte qui est ensuite renvoyé au client (au hasard un service d'inversion de texte).

## Service XML

Un service qui demande un serveur FTP et le nom d'un fichier XML qui s'y trouve. Le service va ensuite procéder à une analyse du fichier grâce à un XML SAX Parser. Il détecte les erreurs de syntaxe XML et compte les occurrences de chaque balise. Il écrit ensuite son rapport dans un fichier texte qu'il envoie ensuite sur le serveur FTP spécifié précédemment.

## Service Messagerie

Service qui requiert une authentification (non persistante). Les utilisateurs peuvent s'inscrire et se connecter pour accéder à plusieurs fonctionnalités :

- Ecrire un message : permet d'envoyer un message à un autre utilisateur
- Boite de réception : permet de lister les messages que l'on a reçus.

# Conclusion

## Bilan du projet

Ce projet nous a permis de nous familiariser avec le paradigme réflexif de Java et de manipuler plus amplement les objets *Class*. Nous avons ainsi mieux compris comment fonctionnait la classe *ClassLoader* et la Java Virtual Machine.

Nous avons aussi pu travailler à nouveau sur la communication client-serveur par l'intermédiaire de sockets et réfléchir à la thread safety. Ce projet a finalement servi de récapitulatif très complet de notre deuxième année, ce qui nous a permis de revoir chacune des notions que nous avons découvert au cours de l'année.

Nous sommes très heureux d'avoir pu réaliser cette application qui clôture l'option Application Réflexive. Nous avons pu aller au-delà des bases de Java et apprendre des choses inédites et que tout le monde n'a pas l'occasion de découvrir, nous donnant un avantage certain dans le monde professionnel.

## Améliorations possibles

Nous n'avons pas eu le temps d'implémenter la mise à jour de bibliothèque (mise à jour d'archives JAR). Pour ce faire nous avions envisagé de mettre à jour chaque classe composant la bibliothèque.