Amélioration d'une application de vente aux enchères

Alexis BONNIN Esteban LAUNAY Sacha LORIOT
Ny Sitraka FIDIMIHAJAMANANA

Decembre 2017

Introduction

Ce projet s'inscrit dans le cadre du module *Middleware* du Master 2 Alma. Il a pour but d'améliorer une application déjà existante, conçue par un groupe d'étudiants du M2 ALMA 2015/2016. Cette application permet de simuler des ventes aux enchères avec une architecture Client-Serveur. La communication entre les entités est assurée par la technologie Java RMI. Dans un premier temps, nous allons élaborer une critique de l'application existante en évoquant les points à améliorer. Ensuite, nous présenterons les améliorations apportées.

1 Critiques sur l'existant

1.1 Structure du projet

Le projet est organisé en deux "packages" : **client** et **serveur**. Dans le "package" client, on retrouve la classe principale *Client.java* ainsi qu'une interface *IAcheteur* contenant les méthodes qui seront appelées par le serveur via la technologie Java RMI. Par analogie, le "package" Server contient la classe principale du serveur, une interface *IVente* contenant les méthodes du serveur ainsi qu'une classe *Donnee* qui simule une base de donnée pour stocker les objets proposées lors des ventes aux enchères.

Avec cette configuration, le déploiement du serveur et du client sur des machines indépendantes n'est pas possible. En effet, les deux entités sont corrélées dans le même projet même si elles sont contenues dans deux "packages" différents. Par conséquent, l'exécution de l'application ne fonctionne qu'en local sur une machine.

1.2 Détection de bugs

Après une première exécution de l'application, quelques disfonctionnements ont été relevées. Tout d'abord, la déconnexion d'un client n'a pas été gérée. En effet, lorsque le client ferme la fenêtre principale, le processus client continue à s'exécuter en arrière plan. Par conséquent, le serveur garde le client dans la liste des acheteurs en cours. Le bouton "passer" une enchère n'est pas complètement fonctionnel. L'interface n'est pas ergonomique, il est même difficile de différencier les problèmes du fonctionnement attendu.

1.3 Code hasardeux

Une classe Donnee est présente et dispose d'une liste d'objets (IObjet). Cette dernière est sensée représenter les enchères à venir. Or, celle-ci est uniquement utilisée pour initialiser une liste présente dans la classe VenteImpl. Elle devient ensuite inutile. Outre cela, on remarque que l'interface IObjet est requise pour le côté serveur et client. C'est une mauvaise pratique. En effet, le client n'a aucunement

besoin d'avoir accès à l'objet tel qu'il est sur le serveur. Les seules informations dont a besoin le client sont : le nom de l'objet en cours de vente, sa description, son prix courant et le meilleur enchérisseur.

2 Améliorations apportées

2.1 Correction de bug

Pour commencer, nous avons corrigé les bugs qui survenaient à l'exécution de l'application. La déconnexion du client a été géré convenablement dans le serveur. En effet, dès que le client quitte la fenêtre principale, le serveur est notifié et met à jour la liste des clients actifs. De plus, la commande "passer" lors de l'enchère a été corrigé et rendue fonctionnelle. Par la suite, quelques modifications ont été faites sur l'interface pour la rendre plus ergonomique et lui attribuer ainsi un aspect plus clair et compréhensible.

2.2 Modification du modèle

Un des gros problèmes de l'implémentation de base était la non-séparation franche du client et du serveur. Pour ce faire, nous avons transformé le projet en deux projets "maven" indépendants. Les interfaces nécessaires se trouvent dans un package noté "api", dans les deux projets.

Nous avons aussi enlevé l'échange d'objets (IObject) entre le client et le serveur, seules des informations concernant les objets en vente (description, prix, etc) sont réellement échangées. Cela permet d'éviter la corrélation entre serveur et client.

2.3 Nouvelles fonctionnalités

La première fonctionnalité que nous avons implémenté est la mise en place d'une base de données réelle. Cela permet de stocker les objets en vente ainsi que ceux qui ont été vendu. Pour ce faire, nous avons créé deux fichiers JSON : le premier stocke les objets à mettre en vente, le second stocke les objets vendu.

Une des fonctionnalités qui a demandé le plus de modification est la mise en place d'un système de salle. En effet, toutes les informations stockées dans le serveur ont dû changer de format pour permettre la séparation des différentes salles. Nous avons implémenté un système avec un nombre défini de salles. Chaque salle est simplement défini par un numéro de 0 à 9 pour un nombre total de 10 salles. Par le suite, il serait possible de rendre ce système dynamique.

3 Conclusion

Nous disposons d'une plate-forme de vente aux enchères, qui respecte une architecture Client-Serveur, fonctionnelle. Les améliorations sont axés sur la correction de bugs ainsi que la création de nouvelles fonctionnalités telles que la création d'une base de données ou encore la mise en place du système de salles. La tâche la plus difficile à été la mise en oeuvre du système de salles. Néanmoins, il reste des points à améliorer tels que l'interface graphique ainsi que la création dynamique des salles.