UNIVERSITÉ DE NANTES

M2 ALMA - Middleware

Extension d'une application distribuée

Auteur :
Théo Dolez
Laurent Girard
Florent Mercier
Aurélien Brisseau

Encadrant :
Achour Mostefaoui

23 décembre 2017

Table des matières

Introduction

Dans le cadre du cours de Middleware du parcours Master 2 ALMA, nous avons eu pour projet de revenir sur un de ceux réalisés par les étudiants de l'année dernière. Pour ces projets, les étudiants de l'année dernière devaient réaliser une application distribuée modélisant un système d'enchères en ligne, avec un modèle client-serveur. L'application devait être en Java et utiliser le framework RMI (Remote Method Invocation) pour la gestion de la partie réseau. Dans ce rapport, nous expliquerons quel sujet nous avons choisi et pourquoi. On essayera ensuite d'ajouter des fonctionnalités intéressantes à celui-ci tout en restant le plus possible dans le cadre du cours de Middleware.

Chapitre 1

Présentation du projet

1.1 Choix du projet

Nous avions à disposition trois projets réalisés par des étudiants de la promotion précédente. Nous avons donc analysé chacun de ces projets afin de faire ressortir celui sur lequel nous pouvions manipuler le code le plus aisement possible. Notre choix s'est ainsi tourné vers le projet nommé **pay2bid** réalisé par *Arnaud Grall*, *Alexis Giraudet* et *Thomas Minier*. Nous décrivons ci-dessous les côtés positifs et négatifs de ce projet.

1.2 Les plus

L'organisation du code est limpide, les noms des packages et des classes on été choisis judicieusement. De même pour la structure même des différentes classes. La documentation dans les fichiers de code est claire et précise. Côté parallélisme, le serveur est considéré comme un *moniteur de Hoare*. Les méthodes qu'il propose ne sont utilisables et accessibles que par un processus à la fois et gèrent la mise en attente des processus via des files d'attente. On peut ajouter dans cette section la gestion du chronomètre pour chaque round des enchères. En effet, le chronomètre a été placé chez le Client, on évite alors les problèmes liés aux actions émises avant la fin du temps imparti mais on génère un peu plus de traffic réseau.

1.3 Les moins

Niveau fonctionnalités de la salle aux enchères nous avons notés quelques petits défauts. Il est possible de faire qu'une seule enchère en même temps, il n'y a pas d'identification pour

les clients (impossible de savoir qui gagne le lot), on peut effectuer des enchères négatives, un client qui se déconnecte en cours d'enchère provoque la non fin de l'enchère, des erreurs dans le GUI (les compteurs des anciennes enchères continuent de tourner). Aussi le vendeur peut participer à sa propre enchère et faire monter artificiellement le prix de son objet.

Le fait de ne pas avoir d'identification des clients et notamment de ne pas savoir qui remporte un round crée aussi d'autres problèmes où, si lors d'un round plusieurs clients font la même enchère gagnante, aucun ne sait s'il gagne, l'attribution d'un gagnant dans ce cas étant laissée au hasard. Le client voyant son montant affiché peut penser qu'il est en train de remporter la mise mais après un round sans enchère pourra voir le lot attribué à un autre client.

La plupart des problèmes rencontrés concernent le métier de l'application et non des erreurs bloquantes pour l'exécution de l'application.

Chapitre 2

Corrections de bugs et ajouts de fonctionnalités

2.1 Correction de bugs fonctionnels

Nous avons corrigé un ensemble de problèmes rencontrés au cours de l'étude du projet :

— Possibilité au créateur de l'enchère de miser sur celle-ci
— Vainqueur aléatoire en cas d'égalité
— Gestion de la fin d'un tour d'enchères quand tout le monde a misé
— Permission de valeurs négatives et d'enchères inférieures au prix en cours
— Relance des anciennes enchères lors du début d'une nouvelle enchère
— Problème de gestion des déconnexions pendant une enchère (l'enchère ne s'arrêtait jamais)
— Erreur en cas de non-enchère au premier tour
— Exception levée lors du lancement du client avant qu'un serveur soit initialisé

La correction de ces erreurs peut paraître simple mais il est assez difficile de s'assurer que l'on n'introduit pas de nouveaux problèmes par effet de bord, notamment sans moyen aisé de tester automatiquement une application distribuée et ainsi s'assurer une non régression.

2.2 Identification des utilisateurs

Afin de permettre une identification des participants dans l'application, nous avons ajouté une fenêtre lancée à l'exécution d'un client afin que l'utilisateur rentre un nom qui l'identifiera durant le temps où il est connecté à l'application. Cela nous permet en plus de savoir qui est le vainqueur actuel de l'enchère en cours, information que nous affichons dans l'interface de l'application.

2.3 Multiples salles d'enchère

Afin d'étendre le projet **pay2bid**, nous pensions donner la possibilité aux clients de se connecter sur plusieurs salles d'enchères (dans notre cas, tous les clients se connectent à toutes les enchères, cependant pour une fonctionntalité se rapprochant au mieux de la réalité, chaque client devrait pouvoir choisir les enchères auxquelles il souhaite participer). Il était suggéré, dans le rapport des propriétaires du projet, qu'il serait possible d'ajouter cette fonctionnalité en effectuant les modifications suivantes :

— Modification du serveur : ajout d'une liste de salles de vente en cours et attente avec les méthodes correspondantes.
— Modification du client en conséquent (récupération de la liste des salles de vente, ...)
— Ajout d'un écran listant les salles de vente dans l'interface graphique.

Tout d'abord, nous avons essayé de modifier le serveur de façon à ce qu'il puisse gérer plusieurs enchères simultanément. Cela se traduit simplemenent par une Map avec pour clé un entier et pour valeur une enchère. Cette Map permet de retrouver une enchère selon son identifiant. Chaque client pouvant se connecter à plusieurs salles, nous introduisons de la même manière ce type de Map dans le client. Nous avons donc commencé à appliquer les modifications nécessaires pour que les clients et le serveur puissent manipuler plusieurs enchères simultanément. Cependant, les modifications entrainent de nouveaux problèmes par rapport au parallélisme notamment. En effet, le projet n'étant pas prévu pour gérer le parallélisme à un niveau "supérieur", c'est-à-dire que tous les clients puissent se connecter sur des enchères partagées. Nous arrivons rapidement à la conclusion par un effet de *snowball* sur le projet, qu'une simple modification entraine d'autres modifications plus importantes. L'ensemble du projet doit être remanié pour ajouter cette fonctionnalité. L'IHM quant à elle doit être entièrement repensée pour intégrer cette fonctionnalité. En effet, il est nécessaire avec cette fonctionnalité qu'un client puisse suivre les enchères sur lesquelles il participe.

Conclusion

Suite à l'étude du projet Pay2Bid, réalisé par des étudiants de l'année dernière, nous avons bien compris les problématiques liées au développement d'une application distribuée. Nous avons pu corriger certains problèmes de l'application et y ajouter quelques petites fonctionnalités qui nous semblaient indispensables pour obtenir une expérience utilisateur agréable et fluide.

Nous avons ensuite essayé d'implémenter la possibilité aux clients de se connecter à plusieurs salles d'enchères mais sans résultat, cela nécessitait de modifier le code du projet trop en profondeur. Plutôt que d'essayer de l'introduire dans le projet existant, une réécriture complète pensée pour cette fonctionnalité aurait été plus aisée mais ce n'était pas notre but lors de ce projet.