Projet Middleware

Jehanno Clément, Mahier Loïc, Duclos Romain

M2 ALMA - Université de Nantes

1 - Introduction

Pour ce projet de middleware nous devions reprendre deux projets fait par la promotion de l'an dernier. Il fallait proposer une courte critique constructive des deux projets et proposer une amélioration pour l'un des deux. Les deux projets étant déjà très complets et ayant été modifiés les années passées, nous avons opté pour effectuer quelques corrections sur les deux projets. En effet, nous ne voyons pas de grosses évolutions à apporter.

Pour toute la suite du rapport, vous trouverez notre code ici.

2 - Critique

A - Pay2Bid

Un client ne peut pas se connecter au serveur tant qu'il y a une enchère en cours. Cela semble être un blocage un peu fort. Si l'enchère en cours se prolonge alors les clients en attente peuvent être amenés à attendre longtemps avant de pouvoir accéder au serveur d'enchère.

Il ne peut y avoir qu'une enchère par tour. Si un client lance une enchère, il faut que celle-ci soit terminée avant qu'un autre client lance une enchère. Encore une fois, cela semble être un blocage un peu fort.

Ces deux premiers points sont les conséquences de choix d'implémentation qui se justifient, cependant, ils rendent l'application moins permissive que ce qu'elle pourrait offrir. Assouplir les blocages permettrait à plusieurs clients de se connecter pendant les enchères et permettrait aussi aux clients de participer à plusieurs enchères en même temps.

Autre point : les clients peuvent se connecter avec le même pseudo, par exemple trois clients nommés Bob, si l'un d'eux lance une enchère, le système semble considérer que les trois clients sont la même personne et aucun ne peut renchérir. L'enchère tourne alors à l'infini et personne ne peut plus se connecter (parce qu'il y a une enchère qui tourne).

B - Gringott

Lors de l'inscription à la vente aux enchères, nous pouvons avoir deux fois le même pseudo, ce qui peut prêter à confusion pour un enchérisseur. En effet, comme nous allons le voir après, deux acheteurs nommés Bob peuvent posséder l'article mais en réalité on ne sait pas lequel possède la dernière enchère dessus.

Deux clients peuvent également poster un article au même moment. Qui a la main sur la vente aux enchères dans ce cas ? De même, ils peuvent effectuer une enchère en même temps. Il va donc falloir modifier cela, car le "synchronized" sur les actions du client dans la classe ClientApp.java ne sert pour ainsi dire à rien. En effet, c'est côté serveur qu'il faut empêcher cela.

3 - Modifications Pay2Bid

1 - Blocage des connections lors des enchères

Pay2Bid fonctionne avec un système de "tours". Un tour correspond à une mise en vente avec les enchères dessus. Le tour se termine quand l'enchère en cours a été remportée. Pendant ce tour, on ne peut pas ajouter de nouvelles enchères et aucun client ne peut se connecter au serveur. Nous trouvons ces blocages un peu forts et nous avons essayé "d'assouplir" un peu le fonctionnement de Pay2Bid.

- possibilité de connection lors d'une enchère

La méthode de connexion au serveur utilisait un *while()* et un *wait()* pour faire attendre jusqu'à la fin du tour les personnes voulant se connecter. Nous avons tout simplement retiré cette boucle pour faire en sorte que les clients puissent se connecter n'importe quand.

2 - Utilisation de *"name"* et ClientBean

Le projet Pay2Bid utilise une classe *ClientBean* pour gérer l'identité de ses clients, avec notamment un identifiant unique. Mais la classe *Client* donne aussi un nom (attribut *name*) aux clients. De ce fait, nous avons remarqué que dans le code, l'attribut *name* est utilisé alors qu'il n'y a aucun "contrôle" dessus (pas d'unicité, etc). Cela entraîne de nombreux bugs comme cité dans la partie 2 - Critique. Nous avons essayé de corriger cela.

Nous avons tout simplement ajouté un code unique au pseudo d'un client lors de sa connexion. Toutes les méthodes qui devaient vérifier l'identité d'un client lors d'une action sont dotées maintenant d'une plus grande "sécurité" car les noms sont uniques.

Une autre façon de faire aurait été d'utiliser la classe *ClientBean* et donc l'UUID de cette classe. Cela aurait été plus élégant mais pour le moment nous nous sommes contentés de rendre unique l'attribut *name* des clients.

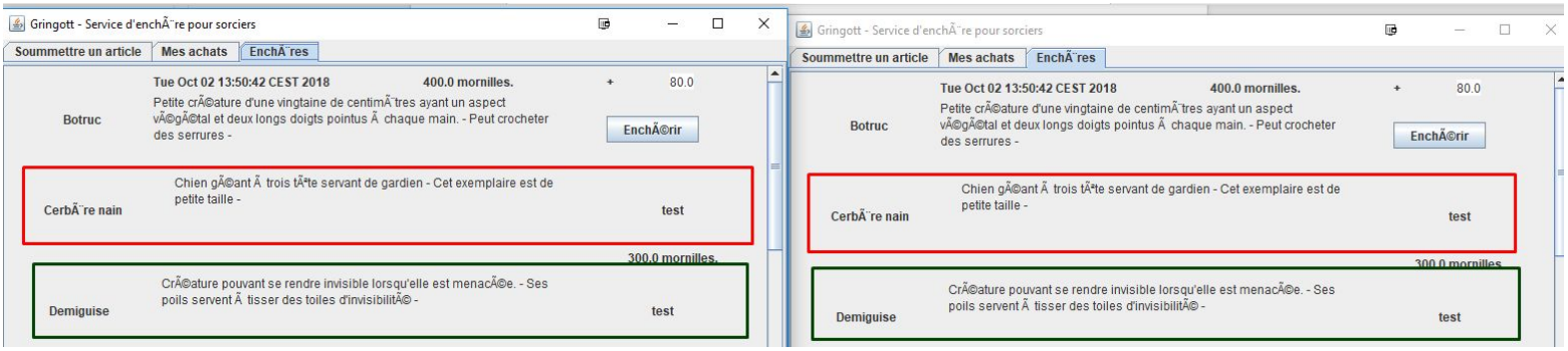
4 - Modification Gringott

1 - Modification du pseudo

Les modifications ont lieu dans le fichier ServerApp.java

Le souci actuel est le suivant : lors de la déclaration du pseudo, deux clients différents peuvent choisir le même pseudo. Le souci est que lors de la vente, l'enchère sur l'objet est définie par le pseudo. Comme nous pouvons le voir sur le screen suivant, nous avons deux clients différents qui ont enchéri sur deux articles différents, mais l'objet est identifié par "test".

Ainsi une fois que la vente est remportée, nous arrivons à ce genre de confusion :



Les deux objets ont étés achetés par deux personnes différentes, mais du même nom.

Nous nous proposons de corriger ce problème en ajoutant un hashcode à la fin du pseudo. Si un utilisateur veut s'enregistrer sous un pseudo déjà existant, nous regardons alors le temps à la milliseconde près et rajoutons un hashcode basé sur cette date à la milliseconde près.

Ainsi, la probabilité que 2 "Bob" créent un pseudo au même moment, est vraiment réduite. Sinon, nous pourrions faire en sorte de synchroniser cette vérification afin d'être sûr de ne créer qu'un seul hashcode. Mais en pratique, il serait sûrement plus simple et rapide de simplement augmenter la précision du timestamp pour le descendre jusqu'aux nanosecondes (System.nanoTime()).

Remarque : pour plus de clarté côté utilisateur nous avons ajouté le nom du client dans sa fenêtre pour qu'il sache quel est son pseudonyme (surtout si celui-ci a été modifié par le serveur pour ajouter un hashcode).

2 - Synchronisation de l'enchère

Actuellement, dans la classe *ClientApp.java*, la fonction "*actionPerformed"* est en "synchronized". Le souci est que cette fonction "synchronized" est du côté client, elle empêche seulement que le même client puisse enchérir deux fois en même temps. Mais qu'en est-il si deux clients enchérissent au même moment ? Nous n'avons pas pu tester ce code car nous ne pouvons pas cliquer sur deux endroits différents en même temps, mais ici, nous ne savons pas comment réagirait le serveur.

Nous avons donc synchronisé les fonctions qui gèrent l'enchère et la soumission d'article au niveau du serveur (dans *ServeurApp.Java*). Pour ce faire, nous avons mis les fonctions *"bid"* et *"submit"* en "synchronized".

5 - Conclusion

Il est assez compliqué de reprendre un code qui a déjà été retouché plusieurs fois au préalable. Nous avons eu quelques difficultés à faire des améliorations en lien avec les problèmes de synchronisation et de concurrence car les deux projets nous semblaient assez complets à ce niveau. A part peut-être au niveau de certains choix d'implémentation (blocages trop fort), nous n'avions pas grand-chose à rajouter.

De ce fait la plupart de nos correctifs viennent corriger des bugs ou permettent de rendre l'application plus "*user friendly"*. Mais comme la plupart de ces bugs sont liés au côté "*multi-user*" (gestion des pseudos), nos correctifs ont tout de même un impact appréciable. Bien sûr, il reste encore de nombreuses choses à corriger mais nous espérons avoir enrayé une bonne partie de ces petits bugs.