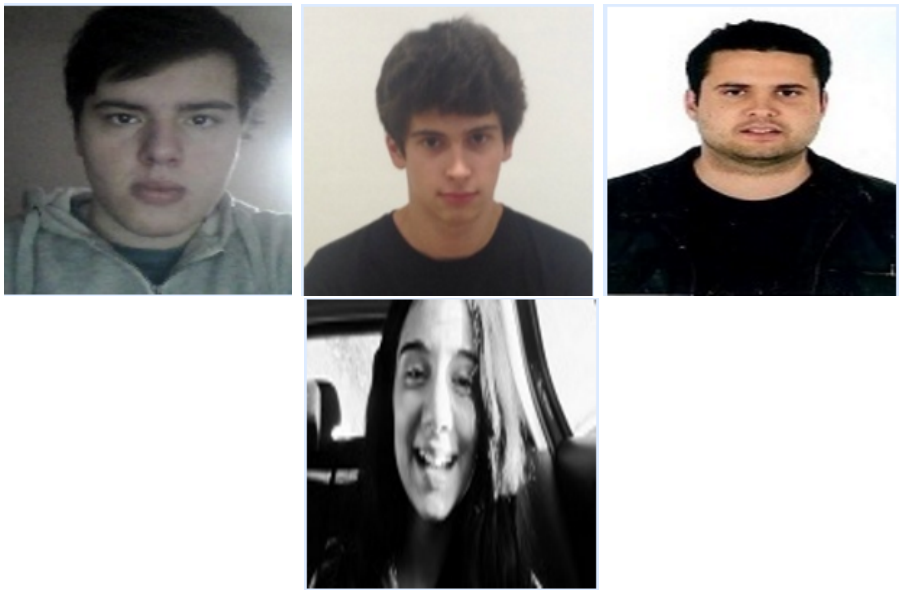
Sistemas Distribuídos
Gestor de Leilões

André Ricardo Covelo Germano A71150
André Rodrigues Freitas A74619
João Henrique Ribeiro Valente A76106
Sofia Manuela Gomes de Carvalho A76658

2 de Janeiro de 2017



Conteúdo

1 Introdução

Neste trabalho implementamos uma aplicação distribuída que permite fazer a gestão de um serviço de leilões.

Esta aplicação pode ser acedida por dois tipos diferentes de clientes: vendedores ou compradores. Os clientes que pretendem leiloar um item devem criar um leilão e indicar uma descrição do item e o sistema atribui um número a cada leilão. Para um comprador licitar um item, tem de indicar o número do leilão em que o item está registado assim como um valor (em euros). O vendedor pode dar como terminado um leilão que esteja em curso e o sistema declara vencedora a oferta com valor mais elevado feita até ao momento.

Os utilizadores do sistema podem interagir entre si, sendo intermediados por um Servidor.

2 Desenvolvimento

Relativamente ao funcionamento da nossa aplicação, para iniciar o servidor basta escrever no terminal 'java eBai' e para se entrar no servidor como cliente escreve-se 'java Cliente', sendo necessário abrir outro terminal para isso. Está sempre disponível o comando '!help' para obter informações sobre os comandos que podem ser usados.

2.1 Cliente

O Cliente é um processo de duas *threads*, sendo uma usada na classe *EnviaMensagem* e a outra na classe *RecebeMensagem*.

A *thread* da classe *EnviaMensagem* é responsável por enviar todo o *input* do utilizador para o servidor e a *thread* presente na classe *RecebeMensagem* é responsável por ler todas as mensagens do Servidor e apresentá-las ao utilizador através do terminal.

2.2 Servidor

O Servidor, por sua vez, é constituído apenas por uma *thread* inicial que permite à porta de comunicação escolhida comunicar com os clientes e criar uma nova *thread* para cada linha de comunicação com um cliente.

Do servidor faz também parte uma instância da classe *BaseDados* que é responsável pelo armazenamento de todos os dados que são necessários para o funcionamento do serviço, nomeadamente os vendedores e compradores registados, os leilões atualmente em curso, entre outros.

2.3 BaseDados

A classe *BaseDados* corresponde à base de dados responsável pelo armazenamento dos dados referidos na secção anterior.

Para isso, usamos a estrutura *Map<String,Utilizador>* que permite armazenar os utilizadores, sendo a *String* o email do *Utilizador*. Não existem emails repetidos logo o email permite identificar cada Utilizador, daí a sua utilização como chave no Map.

Usamos também *Map<Utilizador, Set<Leilao>>* onde estão armazenados todos os leilões de cada utilizador representado através de um *Utilizador* a que corresponderá um *Set<Leilao>* com os seus leilões.

Tivemos ainda de armazenar uma *List<Leilao>* que contém todos os leilões existentes e uma variável inteira *totalLeiloes* que corresponde ao número total de leilões existentes. Sempre que é adicionado um leilão a um determinado utilizador, essa variável vai ser incrementada e o leilão é também acrescentado à lista de leilões para além de ser acrescentado no *Map* associado ao respetivo *Utilizador*.

2.4 Comunicação Vendedor-Comprador

A comunicação entre Vendedor e Comprador é intermediada por um servidor, tal como sugerido no enunciado do Trabalho Prático.

Para o cliente se registar, usa o comando 'registar', sendo aí enviada uma mensagem ao servidor, que por sua vez, faz *parsing* da informação, retirando o email de utilizador e a password, que são guardados no *Map* dos utilizadores através do método *registaUtiizador* que é *synchronized* para que apenas se registe um utilizador de cada vez sem haver conflitos.

De seguida, um utilizador incia sessão e terá disponíveis novos comandos como adicionar leilão, licitar, ver lista de leilões e terminar leilão.

Em relação à licitação, é possível que vários clientes licitem items ao mesmo tempo, sendo que, para um mesmo leilão, um cliente espera que a licitação atual seja feita através de *locks* explícitos e variáveis de condição. Uma licitação só é efetuada com sucesso caso o valor da licitação seja superior ao atual valor máximo. Nesse caso, é adicionada à base de dados o novo leilão em que o utilizador participou. Isto corresponde aos métodos *novaLicitacao* e *licitacao*.

Para terminar um leilão, é necessário que o vendedor utilize o comando 'terminar', dando o número de leilão, sendo chamado o método *fimLeilao* da classe *leilao* que, mais uma vez através de locks explícitos, coloca a variável *aberto* a falso, o que impede outros licitadores de fazer novas licitações e depois de todas as licitações em espera acabarem, termina o leilão, enviando uma mensagem a todos os participantes informando que o leilão terminou e indicando o vencedor com a respetiva licitação.

3 Conclusão

Este Trabalho Prático permitiu aprofundar conhecimentos adquiridos quer nas aulas teóricas quer nas aulas laboratoriais sobre os conceitos de controlo de concorrência, assim como os de comunicação cliente/servidor.

Em suma, julgamos que conseguimos produzir um gestor de leilões que possui as funcionalidades pretendidas e, por isso, fazemos um balanço positivo deste trabalho.