Universidade do Minho

Mestrado Integrado em Engenharia Informática

Sistemas Distribuídos

Relatório
2 de Janeiro de 2017

Mário Ferreira A70441

Tiago Sá A71835

Universidade do Minho
Escola de Engenharia

Conteúdo

Capítulo 1

Estrutura

1.1 Cliente

É na classe *Cliente* que é feita a conexão entre o cliente e o servidor, sendo fornecido no momento da execução o endereço e a porta do servidor. A classe Cliente é composta ainda por duas subclasses, a *ServerListener* e a *ServerAnswer*. A primeira, como o nome indica, recebe inputs do servidor, através do seu *BufferedReader* e imprime-os na linha de comandos. A segunda, recebe os inputs do utilizador e transmite-os para o servidor através do seu *PrintWriter*. Estas duas classes são *Threads* e executam simultaneamente.

1.2 Server

Na classe *Server* temos as três estruturas de dados mais importantes, os Leilões, os Utilizadores, e o mapa de Printwriters. Para implementar as conexões multi-threaded, recorreu-se a um socket TCP. Depois de criado o socket, este passa a aceitar clientes que, ao abrirem uma conexão dão origem a uma *Thread* onde vão ser processados os vários pedidos. As threads foram implementadas na classe *ServerThread*, subclasse de *Server*. Nessa classe, é feita a autenticação de cada cliente, por intermediário do mapa de *PrintWriters* e dos métodos da classe *Utlizador* e o logout é feito através da remoção da respectiva entrada no mapa. Garantimos desta forma, que um utilizador apenas possa estar logado num terminal, e que o servidor não tenta comunicar com utilizadores que não estão logados. Nesta implementação, o *ServerThread* comunica com todas as sockets e envia as informações relevantes aos utilizadores, através do uso do mapa de *PrintWriters*. Assim, este estrutura *PrintWriters* permite-nos não só guardar os utilizadores que estão logados assim como escrever para estes informações úteis, como a criação de um novo leilão, quando alguém licita um leilão onde temos a maior oferta, entre outras coisas. Asseguramos os casos de exceção, uma vez que todas as opções presentes no menu estão protegidas com *try* e *catch*.

1.3 Utilizadores

Nesta classe *Utilizadores*, temos um mapa dos utilizadores e um *ReentrantLock*. Nela estão definidos os métodos de registo e login. Estes métodos são limitados pelo lock, de forma a controlar o seu acesso. No registo é garantido que o id do utilizador é único e no login é feita a verificação do id do utilizador, se existe ou não, e se corresponde à password associada, sendo este processo completado na classe *Utilizador*.

1.3.1 Utilizador

Nesta classe estão contidas as informações básicas de cada utilizador, o seu id e password e contém ainda o método de login, em que se verifica se a password recebida é a que foi definida por aquele utilizador.

1.4 Leilões

Nesta classe temos um mapa de leilões e um *ReentrantLock*. Aqui estão definidos vários métodos como adicionar, licitar, mostrar, fechar e listar leilões, para além dos mais básicos como os getters. Todos estes métodos estão protegidos com lock para termos maior segurança no controlo de concorrência.

1.4.1 Leilão

Nesta classe caracterizamos como deverá ser um leilão. Para tal, cada leilão terá um estado (aberto ou fechado), associado a um item e possui descrição, vendedor, quantidade, uma lista de clientes que já licitaram, a maior licitação até ao momento, a licitação base, e ao mínimo de intervalo que cada licitação terá de ter para a licitação maior no momento. Aqui encontram-se, de maior importância, os métodos fechar e licitar leilão. Não possuem qualquer controlo de concorrência uma vez que isso está a ser feito na classe **Leilões** como enunciado atrás.

Capítulo 2

Demonstração

Vamos agora fazer uma sucinta descrição da aplicação, das suas funcionalidades e características. Para poder usufruir da aplicação, em primeiro lugar é necessário o utilizador registar-se no sistema (a).

Depois, através do login dá-se a autenticação do utilizador e caso seja positiva, este é remetido para o menu principal (b) onde é feita a gestão dos leilões. O utilizador pode agora criar os seus próprios leilões, ou listar os leilões abertos, ou que já tenham findado. Se optar por criar um leilão é direcionado para uma interface (c), bastante amigável, onde são apresentados, passo a passo, os aspectos necessários para criar um leilão. Caso os dados introduzidos sejam válidos, o leilão é adicionado à nossa estrutura (o mapa de leilões) e é lhe atribuido um id, que é relativo à sua posição no mapa + 1. Este id é sempre crescente. Se o utilizador pretender, listar os leilões ativos (e), pode depois escolher um leilão e licitar, caso não seja o seu criador ou o maior licitador, no momento, ou fechar (d), caso seja o seu criador.

Relativamente às licitações, como já foi referido, através do controlo de concorrência, apenas uma licitação é processada de cada vez e para esta ser válida é necessário que seja maior que a atual e que a diferença entre esta e a nova licitação seja superior ao mínimo estipulado pelo vendedor. Na listagem de leilões fechados podemos ver informações acerca do vendedor do leilão, do seu comprador e do valor pelo qual foi adquirido. Ambas as listagens, como pedido, indica se o utilizador é o vendedor (*) (e), ou se ele é o maior licitador (+). Todo este sistema é apoiado pela troca de mensagens entre o vendedor e os compradores, sendo ambos informados de alterações no estado do leilão.

```
------ Gestor de leilões -------

[1] Login
[2] Registar
[3] Sair
```

(a) Menu Inicial

```
------ Gestor de leilões -------

[1] Criar leilão
[2] Listar leilões abertos
[3] Listar leilões fechados
[4] Logout
```

(b) Menu Principal

```
------ Criar leilão -------

Item:
batata
Descricao:
batata boa
Quantidade:
10
Licitação base:
10
Mínimo diferença licitação:
2
```

(c) Criar leilão

```
------ Leilão 0 -------

Item: batata
Descricao: batata boa
Quantidade: 10
Vendedor: jorge
Licitação Base: 10.0
Mínimo diferença: 2.0
Licitação Maior: 100.0
Comprador com maior oferta: mario


[0] Fechar
[1] Voltar
```

(d) Listar leilão

```
------ Listagem leilões abertos -------

[0] batata*

V - Voltar
```

(e) Listagem de leilões ativos

Capítulo 3

Conclusão

De forma geral o objetivo foi cumprido, uma vez que o proposto, um sistema de gestão de leilões que permitisse a entrada e manutenção de vários clientes num servidor, foi realizado. A concorrência também está a ser controlada e, depois de vários testes e correções, aparentemente não apresenta erros. Como trabalho futuro, sugeríamos colocar este projeto não vulnerável a clientes lentos, fazendo com que cada cliente não ficasse afetado por qualquer outro. Para isto, cada cliente estaria numa *box* e seria acordado sempre que necessário. Para além disso, a implementação de uma base de dados para guardar toda a informação já existente, ser possível logar um cliente se este tiver uma saída brusca (uma vez que está guardado nos *PrintWriters*, se a saída for bruta este não chega a ser retirado de lá e será sempre dado como logado. Por fim, limitar mais o controlo de concorrência, fazendo, por exemplo, os locks apenas onde estritamente necessário. Algo importante a ter sido feito seria também testes em que fosse possível analisar o procedimento correto do programa quando este atravessa fases críticas, com milhares de clientes logados e a movimentarem-se no servidor, seja a licitar, criar, fechar leilões, etc. Concluindo, havia ainda muito a melhorar, mas o objetivo principal foi concluído, é funcional, não encrava, não tem erros de concorrência e é possível gerir vários leilões e utilizadores de forma correta e satisfatória.