Trabalho prático de Sistemas Distribuídos: Relatório

Grupo 16
Fábio Rafael Correia Guerra Fontes (A78650)
Alberto Campinho Faria (A79077)

Janeiro 2018

Resumo

Este documento consiste no relatório relativo ao trabalho prático desenvolvido no âmbito da Unidade Curricular de Sistemas Distribuídos, do curso de Mestrado Integrado em Engenharia Informática, da Universidade do Minho, no ano letivo de 2017/2018.

Conteúdo

1 Introdução

Com este trabalho prático, pretendeu-se que fosse implementada uma aplicação distribuída para *matchmaking* num jogo online. Em cada partida desse jogo, os jogadores são divididos em duas equipas, de forma semelhante ao que acontece nos jogos *Overwatch* e *Counter-strike*. Mais especificamente, a funcionalidade essencial da aplicação é composta por duas fases:

- Formação das duas equipas para jogar cada partida;
- Escolha, por parte de cada jogador, do *herói* a ser utilizado durante a partida.

O jogo propriamente dito não faz parte da aplicação – apenas se requeriu a implementação do sistema de matchmaking.

Os utilizadores devem poder interagir através de uma aplicação cliente, intermediados por um servidor *multi-threaded*. O cliente e o servidor devem ser implementados em Java, realizando-se a comunicação entre ambos com recurso a sockets TCP. O protocolo de comunicação deverá ser baseado em texto e orientado à linha. Para evitar ficar vulnerável a clientes lentos, cada thread do servidor não deverá escrever em mais do que um socket.

Cada utilizador tem um ranking resultante das partidas já por ele realizadas, definido por um número entre 0 e 9. Cada equipa é constituída por 5 jogadores.

O funcionamento da aplicação deverá proceder de acordo com os seguintes pontos:

- Se ainda não o tiver feito, o utilizador regista uma conta de jogador, especificando o username e password desejados;
- Tendo uma conta de jogador registada, o utilizador autentica-se na aplicação utilizando o username e a password respetivos;
- Estando autenticado, o utilizador manifesta a intenção de jogar uma partida, entrando assim na fase de matchmaking;
- Quando possível, são constituídas as duas equipas para a partida, iniciando-se a fase de escolha de heróis;
- O utilizador escolhe o herói com o qual deseja jogar, de um conjunto de 30 heróis. Apenas podem ser selecionados heróis que não tenham sido escolhidos por outros jogadores;
- Até um dado limite de tempo, o utilizador pode desfazer a sua escolha de herói e selecionar outro não escolhido por nenhum jogador;
- Ao acabar esse limite de tempo, ou todos os jogadores da partida selecionaram um herói, sendo então apresentada a constituição das duas equipas e iniciando-se a partida, ou é abortada a partida.

Dos possivelmente muitos jogadores que estejam em fase de matchmaking, o servidor deverá tentar formar equipas que sejam equilibradas segundo dois critérios:

- Em cada partida, a diferença absoluta entre rankings de todos os pares de jogadores não deverá ser superior a 1;
- Dados 10 jogadores que satisfazem o ponto anterior, deverão ser formadas as duas equipas por forma a minimizar a diferença entre o ranking médio de cada equipa.

Poderão existir, simultaneamente, várias partidas em fase de escolha de heróis, as quais devem ser geridas concorrentemente pelo servidor. Quando uma partida é iniciada com sucesso, deverá apenas ser apresentado o resultado da suposta partida, gerado aleatoriamente, indicando-se a equipa vencedora. Este resultado deve ser usado para atualizar o ranking dos jogadores que participaram na partida.

Foi desenvolvido um servidor e uma aplicação cliente com interface gráfica (cliente GUI), ambos satisfazendo todos os requisitos impostos pelo enunciado do trabalho. Adicionalmente, foi desenvolvida uma aplicação de teste automatizada, a qual estabelece várias conecções com uma instância do servidor, simulando de seguida a interação de vários clientes com o mesmo.

Juntamente com este relatório é disponibilizado o código fonte da aplicação, na diretoria `projects/`, sob a forma de vários projetos do IDE Eclipse. A implementação da aplicação foi dividida em 5 projetos:

- *client-gui* – código específico ao cliente GUI;
- *client-test* – código específico ao cliente de teste;
- *server* – código específico ao servidor;
- *client-common* – código comum aos dois clientes;
- *common* – código comum aos dois clientes e ao servidor.

As funcionalidades da aplicação desenvolvida são identificadas na secção 2. Aspetos relevantes sobre a comunicação entre cliente e servidor são apresentados na secção 3. A secção 4 descreve o cliente GUI e o cliente de teste, e a secção 5 descreve o servidor. O relatório é concluído na secção 6.

2 Funcionalidades

Ao cliente é fornecido uma interface gráfica que permite, a utilização de diversas funcionalidades, sendo que iremos de seguida descrever uma utilização normal da interface gráfica e proceder à descrição de cada uma das janelas que constituem a interface gráfica, juntamente com as funcionalidades da aplicação que é possivel utilizar a partir destas.

Ao ser iniciada a aplicação cliente e nos mostrado uma janela onde é possível estabelecer a conecção com um determinado servidor, através da indicação da porta e endereço de IP desse servidor.

Após a conecção ser estabelecida é nos apresentada uma janela de Autenticação, a partir da qual é possível efetuar a autenticação do cliente colocando o nome de utilizador e a palavra-passe que correspondam a uma conta previamente registada, registo este que é feito na janela de Registo de conta, acessível a partir da janela de Autenticação, para efetuar o registo de uma conta apenas é necessário fornecer um nome de utilizador válido, que não esteja já a ser utilizada por uma outra conta e uma palavra-passe também esta válida.

Estado efetuada a autentificação ou registo do cliente, este chega à janela Inicial, onde são apresentadas diversas informações relativas ao estado do servidor, nomeadamente, o número de jogadores registados, o número de jogadores com a sessão iniciada, o número de jogadores à procura de partidas de jogo e a quantidade de lobbies ativos, para além das informações relativas ao servidor, também nos são apresentadas informações relativas ao próprio jogador, o nome de utilizador, o número de partidas ganhas, o número de partidas perdidas e a razão entre o número de partidas ganhas e perdidas. A partir desta janela é possível efetuar duas ações, iniciar a procura de uma partida de jogo, ou efetuar o logout da conta, o que fará a interface voltar à janela de Autenticação.

Ao iniciar a procura de partidas de jogo, somos colocados em espera até ser encontrada uma partida, sendo que podemos deixar a procura de partidas a qualquer momento pressionando o botão de "LeaveMatchmacking", quando uma partida for encontrada irão ser apresentados dois botões que nos permitem aceitar ou recusar a partida encontrada, assim como caixas que irão ser preenchidas à medida que os restantes jogadores, escolhidos para participar na partida, a forem aceitando e um temporizador que indica quantos segundos restantes temos para aceitar ou rejeitar a partida, caso o temporizador chegue a zero, se aceitarmos a partida teremos que esperar que os restante participantes a aceitem, sendo que caso um ou mais jogadores não a aceitem, iremos

ser colocados outra vez à procura de partidas, existem duas maneiras de rejeitar uma partida clicando no botão de rejeição, causando cancelamento imediato da partida, ou esperando que o temporizador chegue a 0, sendo que iremos ser colocados na janela Inicial após rejeitarmos a partida.

Quando todos os jogadores aceitarem a partida será nos apresentado uma janela de Lobby, nesta janela temos acesso a um sistema de chat para podermos comunicar com os restantes elementos da nossa equipa, os nomes e heróis escolhidos pelos restantes elementos da equipa, um conjunto de heróis entre os quais podemos escolher o nosso e um temporizador a indicar quanto tempo nos resta para escolher o nosso herói, cada equipa não pode ter jogadores com heróis repetidos e por este à medida que os elementos da equipa vão escolhendo os seus heróis estes vão sendo bloqueados nas janelas dos restantes jogadores, caso o temporizador termine e algum dos jogadores não tenha escolhido o seu herói a partida é cancelada sendo que todos os jogadores voltam à janela Inicial, quando todos os jogadores selecionarem os seus heróis o jogo é iniciado sendo então apresentado uma janela com o resultado da partida e a nossa performance nesta, após ser apresentado o resultado todos os jogadores são retornados às suas janelas Iniciais com as suas estatísticas atualizadas, podendo a partir daqui continuar a jogar as partidas que desejarem.

3 Networking

A comunicação entre cliente e servidor é baseada em mensagens. Cada mensagem manifesta o interesse do cliente em realizar uma determinada ação ou indica a resposta do servidor a um determinado pedido.

Cada mensagem constitui uma linha de texto, garantidamente codificada em UTF-8, com terminador de linha CR, LF ou CRLF.

Todas as mensagens, enviadas do cliente para o servidor ou vice-versa, são prefixadas de um identificador que determina a semântica da mensagem. Esse identificador é sempre seguido de dois pontos (:), e possivelmente de um payload específico ao tipo de mensagem, determinado pelo mesmo identificador.

O conjunto de identificadores de mensagens enviadas do cliente para o servidor é distinto do conjunto de identificadores de mensagens enviadas do servidor para o cliente.

O código comum ao cliente e ao servidor que implementa o protocolo de comunicação até aqui descrito encontra-se no package `mm.common.net`. Os tipos que a seguir se referem pertencem a esse mesmo package.

As enumerações `ClientToServerMsgId` e `ServerToClientMsgId` definem os identificadores de mensagens enviadas do cliente para o servidor e vice-versa, respetivamente.

A classe `MessengerBase` encapsula a lógica de gestão da conexão e de leitura e escrita assíncrona de mensagens, comum a ambos cliente e servidor.

4 Cliente

Conforme anteriormente mencionado, foi desenvolvida uma aplicação cliente com interface gráfica (cliente GUI) e uma aplicação cliente de teste automatizada, a qual

Ambos os clientes utilizam a class `NetClient`, a qual encapsula todas as tarefas relacionadas com networking realizadas pelos clientes. O utilizador dessa classe pode invocar métodos que se traduzem na realização de várias ações (e.g. `login()`, `joinMatchmaking()`), e pode também definir callbacks que serão executados aquando da receção de notificações por parte do servidor (e.g. `setOnLoginSucceeded()`, `setOnChatMessageReceived()`).

Para implementar o client gráfico, utilizou-se o framework JavaFX 8. O executável do cliente GUI é disponibilizado juntamente com este relatório no ficheiro `client-gui.jar`.

O cliente de teste estabelece várias conecções com uma instância do servidor, simulando de seguida a interação de vários clientes com o mesmo.

Cada cliente iniciado por esta aplicação de teste começa por se autenticar, de seguida entrando em matchmaking. Após ser encontrada uma partida, o cliente aceita a mesma. Uma vez no lobby, o cliente seleciona (e possivelmente resseleciona enquanto possível) um herói. Após o lobby ter morrido ou uma partida ser jogada, o cliente volta a entrar em matchmaking, repetindo o mesmo processo.

O executável do cliente de teste é disponibilizado juntamente com este relatório no ficheiro `client-test.jar`. Várias opções são suportadas. Pode-se conferir a utilização do executável utilizando-se a opção `-h`.

5 Servidor

O executável do servidor é disponibilizado juntamente com este relatório no ficheiro `server.jar`. Várias opções são suportadas, incluindo para configuração de tempo limite para escolha de heróis. Pode-se conferir a utilização do executável utilizando-se a opção `-h`.

5.1 Networking

Quando o servidor é iniciado é criada uma thread que irá ser responsável por escutar e satisfazer todos os pedidos de conecção, por parte dos clientes.

Esta quando aceita um pedido de conecção feito por um cliente, cria uma nova instância da class Client, sendo que este objeto, é responsável por responder a todos os pedidos do cliente a quem foi atribuido, isto é, é nesta classe que se encontram implementados os comportamentos do servidor perante os pedidos do cliente.

Para tratar do envio e receção de mensagens do cliente são utilizados objectos da classe, ClientMesseger, permitindo assim a a separação da componente de comunicação do servidor que é responsável pela codificação e envio das mensagens e receção e parsing das mesmas, da componente que implementa o comportamento, a classe Client referida acima.

Ao nivel mais baixo do envio de mensagens através dos sockets é utilizado a classe Masseger-Base, que permite também a separação das escritas para os sockets e das leituras dos mesmos, de abstrações mais intuitivas do envio e configuração das mensagens.

5.2 Comportamento do Client

Do comportamento do cliente e do seu estado interno é de realçar a variável state e a persitência das contas e a utilização de temporizadore sendo estes descritos de seguida.

Cada Cliente possui um estado interno que indica qual a secção da aplicação o cliente se encontra, este estado é utilizado para efeitos de sincronização entre o servidor e o cliente, assim o cliente quando está num determinado estado apenas pode ter determinados comportamentos, quando algum pedido é feito por parte do cliente que não faça sentido no estado do cliente se encontra, provoca a disconecção imediata do cliente, no entanto existem determinados casos em que em vez de o cliente ser disconectado, o pedido é simplesmente ignorado, esta comportamento é feito para prevenir os erros que possam surjir devido ao delay de transmissão existente entre o cliente e o servidor o que pode provocar a execução de pedidos que não fazem sentido no estado em que o cliente se encontra. Os restantes casos em que a receção de uma mensagem que não faça sentido no estado em que o cliente se encontra no servidor, não possa ser justificada pelo delay de transmissão da mesma mensagem, provoca a desconecção do cliente e a apresentação de um report no terminal do servidor.

A associação de uma conta de utilizador com o cliente a quando da autenticação deste e feito através da classe AccountManager que possui todas as contas resgistadas no servidor, estas são armazenadas num ficheiro dado a quando da execução do servidor, este ficheiro também pode conter contas sendo que estas são carregadas no inicio da aplicação.

Os temporizadores utilizados quando nos encontramos nas fases de escolha de heróis e da aceitação da partida encontrada é feita utilizando a classe timer que nos permite criar um thread

que irá executar um TimerTask após um determinado período de tempo.

6 Conclusão

Todos os requisitos impostos pelo enunciado são cumpridos pela aplicação desenvolvida. Adicionalmente, foram implementadas várias funcionalidades não propostas pelo enunciado, entre as quais:

- Fase de confirmação de partida, antes da fase de escolha de heróis;
- Chat de equipa, disponível durante a fase de escolha de heróis;
- Possibilidade de requerer que um herói aleatório seja selecionado.