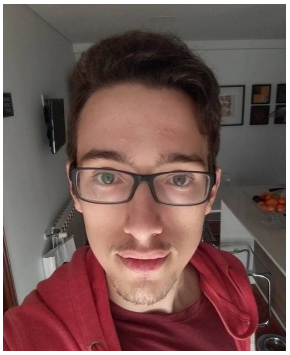
Sistemas Distrubuídos
Oberwatch

Bruno Arieira
a70565

João Palmeira
a73864

José Dias
a78494



3 de Janeiro de 2018

Resumo

Implemente uma aplicação distribuída para matchmaking num jogo online por equipas, semelhante ao Overwatch. A funcionalidade essencial é composta por duas fases: 1) formar as duas equipas para jogar cada partida; 2) a fase em que cada jogador escolhe qual o herói com que joga, antes de o jogo começar. (O jogo propriamente dito não fará parte da aplicação, apenas o matchmaking.) Os utilizadores devem poder interagir, usando um cliente escrito em Java, intermediados por um servidor multi-threaded também escrito em Java, e recorrendo a comunicação via sockets TCP.

Conteúdo

1 Introdução

Este trabalho foi nos proposto no âmbito da Unidade Curricular Sistemas Distribuídos, com o objetivo de consolidar os conceitos apreendidos nas aulas. Neste projeto implementamos uma aplicação distribuída que permite fazer o matchmaking de um jogo online.

Esta aplicação é acedida pelos jogadores com o intuito de começarem um jogo. Deste modo cada jogador entrará num menu de matchmaking consoante o seu rank, será associado a uma equipa, escolherá o seu herói num período de 30 segundos e entrará na partida. Tudo isto, é executado com controlo de concorrência em que um jogador pode fazer login, por exemplo, ao mesmo tempo que um jogador esteja a escolher um herói.

No fim do jogo, a aplicação indicará qual a equipa vencedora e as informações sobre o jogador serão atualizadas.

2 Desenvolvimento

Neste tópico abordaremos todas as classes que desenvolvemos no trabalho e explicaremos em que consiste cada uma delas.

2.1 Servidor

O servidor para cada acesso cria uma thread que é utilizada pelo utilizador (jogador) sempre que se conecta a aplicação.

2.2 Cliente

A classe Cliente contém uma função main que auxilia a interface gráfica da aplicação.

2.3 UserInterface

Esta classe é uma classe interface que auxilia a classe UserMenu, pois contém o registo de todas as funções contidas nesta.

2.4 UserMenu

Nesta classe estão lá definidos todos os menus que o utilizador (neste caso, o jogador) terá acesso ao utilizá-la.

Temos um menuInicial, no qual um jogador pode fazer o login, pode-se registar ou até sair da aplicação.

O menuConta tem como opção ver a performance do dado jogador que se registou (Estatísticas), começar a jogar e sair. Ao seguir a primeira opção, este vai ter acesso ao seu rank, o seu número de vitórias assim como as vitórias seguidas,e outras opções como vamos poder observar mais abaixo.

Ao começar o jogo, vamos obter o menuJogo, onde são apresentadas duas equipas (blue e orange) de 5 jogadores (cujo estes terao apenas um rank a mais ou a menos do utilizador que se registou) através de dois ArrayLists, onde tem a opção de escolher o Herói pretendido.

No menuHeroi sao apresentados 30 heróis, onde o jogador tem a alternativa de escolher um deles, sendo o escolhido, diferente dos herois dos outros jogadores da equipa.

Depois deste processo, é apresentado o menuJogoHeroi, onde são apresentadas as equipas com os jogadores, mas com os respetivos herois associados.

2.5 Handler

Handler é a classe que contem as funções que por sua vez vão auxiliar a classe Cliente.

Esta é composta por a função regista, onde recebe um username e uma password, que verifica se existe algum username compativel entre os jogadores já registados e se não se verificar, o registo é efetuado com sucesso.

A função Login, recebe as mesmas informações que o regista, averiguando se este pode ser efetuado.

De seguida criamos uma nova função chamada checkQueue, que vai determinar se a diferença entre os ranks de dois jogadores ultrapassa o exigido pelo proposto do enunciado, caso não se verifique tal diferença, adiciona a um ArrayList para posteriormente formar equipas.

Na função startGame, recebemos um ArrayList composto por 10 jogadores, que são distribuidos 5 por cada equipa (eq1 e eq2). Foi necessário ter em atenção que apenas um jogador cria um jogo

e outros jogadores acedem a esse mesmo jogo.

Definimos também a função startMatchmaking, que invoca tanto a startGame como a checkQueue de forma a colocar o jogador em matchmaking e verifica se existem jogadores suficientes para começar um jogo. Caso existam inicia o jogo.

A função playerStats, é a que dá como resultado as estatísticas de cada jogador, fornecendo o nome do player, o seu rank, o seu número de jogos, o total de vitórias, as vitórias seguidas e o rácio de jogos ganhos relativamente aos jogos efetuados.

Temos também a função hand que é composta por um switch, onde estão todas as funcionalidades disponíveis na aplicação. Cada um dos cases definidos nela invocam grande parte das funções criadas nesta classe e passando-lhes como argumento o imput do utilizador (que provém da classe UserMenu).

2.6 Conta

A Conta é composta pelo username e a password do jogador bem como todas as estatísticas desse mesmo jogador (rank, número de jogos, sequência de vitórias, sequência de vitórias com o mesmo rank e o total de vitórias).

Após a criação da conta, o utilizador fica com todas as estatísticas a 0 para além do rank que inicia a 5. Estão definidos também os seus get's.
Por último, colocamos duas funções que alteram as estatísticas consoante o resultado dos jogos.

2.7 Registo

Na classe Registo, foram armazenados os maps com as contas (private HashMap<String, Conta> logC) e jogos (private HashMap<Integer, Game> logG) e ainda foi armazenada uma lista dos heróis (private ArrayList<Hero> logH).

Disponibiliza também funções que adicionam uma conta ou um jogo ou um herói ou ainda que verificam se a conta existe ou não como, por exemplo, as seguintes funções: public synchronized void addc(Conta c) ou public Conta getc(String user) ou public boolean containsConta(String user).

2.8 Team

Nesta classe, é definida a constitíção de uma equipa através da função public Team(Collection<String> players) que recebe a Collection dos jogadores e insere os num array.

Está definida ainda a função public synchronized int select(String u, int s) onde através dela é possível selecionar um herói de cada vez sem haver conflitos, uma vez que é synchronized, e esta retorna 1 se o herói for selecionado com sucesso ou será se o herói não for selecionado.

2.9 Hero

Com a classe Hero é possível definir um herói através de um inteiro, desta forma cada herói tem um código associado.

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 sistema de matchmaking que possui as funcionalidades pretendidas e, por isso, fazemos um balanço positivo deste trabalho. Embora ao realizar o jogo (após o matchmaking) seja gerado um resultado aleatório e de serem alteradas todas as estatísticas dos jogadores, não consigos verificar que o utilizador terminou sessão. E como tal não conseguimos voltar a entrar com o jogador para verificar essas novas estatísticas.

Por último, não conseguimos implementar o tempo limite da seleção do herói.