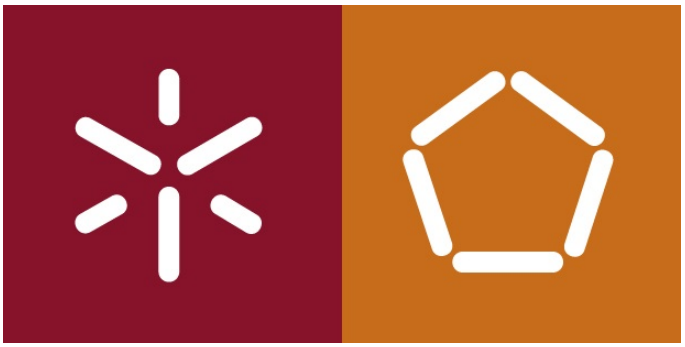
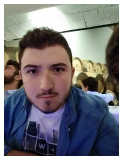


Universidade do Minho - Escola de Engenharia

Relatório do trabalho prático de Sistemas Distribuídos

Matchmaking num jogo online

Autores :
Diana Costa (A78985)

José Oliveira (A78806)

Marcos Pereira(A79116)

Vitor Castro(A77870)


Versão 1.0
3 de Janeiro de 2018

Resumo

Perante a proposta de criar um sistema de *Matchmaking* que simulasse o ambiente inicial de um jogo, desde a formação de equipas, até à escolha de um herói, houve um impasse inicial, devido à necessidade de uma boa estruturação do problema. Tudo isto requeriu a escolha acertada de mecanismos e classes a utilizar, como comunicação via *sockets* TCP (*ServerSocket*, *Sockets*, *Streams*) e o uso de multi-threading ao nível dos servidores e respetivo controlo de concorrência, de modo a que a implementação do sistema fosse a mais eficiente e simples possível.

Depois de algum tempo e trabalho, o resultado encontrado foi eficiente e satisfatório, e os objetivos e respostas às questões do enunciado proposto cumpridos.

Conteúdo

1 Introdução **3**

2 Cliente **4**
2.1 Funcionamento e classe Client 4

3 Servidor **4**
3.1 Classes 5
3.1.1 Server 5
3.1.2 ServerWorker 5
3.1.3 Play 5
3.1.4 User 5
3.1.5 Game 6
3.1.6 ChampionsLobby 6
3.2 Funcionamento 6

4 Conclusões **6**

5 Anexos **7**

1 Introdução

Este projeto surge no âmbito da unidade curricular de Sistemas Distribuídos, do Mestrado Integrado em Engenharia Informática da Universidade do Minho. Sendo um projeto de pequena/média dimensão a ser resolvido na linguagem *Java*, este baseia-se na construção de um sistema de *Match-making*, semelhante ao jogo *Overwatch*, que seja capaz de formar equipas para jogar cada partida, e proceder à escolha de heróis de cada jogador, antes do jogo começar (a implementação do jogo em si não fez parte do projeto).

Este sistema, inicialmente, permite que cada jogador que pretenda iniciar uma partida faça o seu registo (username e password). Estabelece-se assim uma conexão a ser autenticada pelo servidor. Os jogadores, depois de autenticados, esperam até estarem reunidos dez jogadores, e serão distribuídos por duas equipas de cinco jogadores cada, baseada nos rankings dos participantes. Este ranking, resultante das partidas que fez anteriormente, é sumarizado num número entre 0 e 9. Depois de estarem reunidas todas estas condições, todos os dez jogadores são notificados, e procede-se à segunda fase do projeto.

Assim, como segunda parte, é efetuada a escolha dos heróis. Esta é a segunda vez que um jogador interage com o servidor, e consiste na escolha, entre trinta opções, de um herói. Isto implica que não sejam escolhidos heróis previamente selecionados por outro jogador, e que, dentro de um intervalo de tempo pré-definido, toda a equipa tenha que escolher a sua personagem, ou a partida aborta. Este processo permite que vão sendo observadas as escolhas dos outros membros da equipa, tal como num jogo real, e que se possa mudar de herói, desde que dentro do intervalo de tempo.

Por fim, é simulada uma partida, que finaliza com um resultado de vencer ou perder, que influencia o ranking de todos os membros da equipa.

Em suma, a Secção 2 descreve o Cliente e o seu funcionamento geral, enquanto que a Secção 3 apresenta e discute a solução proposta pelo grupo para o Servidor, desenvolvida ao longo do semestre. Esta secção, por uma questão de detalhe, foi dividida em duas secções: a secção 3.1, que descreve todas as classes constituintes do Servidor, e a secção 3.2, que explica o funcionamento do mesmo. O relatório termina com conclusões na Secção 4, onde é apresentada uma análise crítica dos resultados obtidos. Além disso, é feito um balanço tendo em conta as dificuldades ao longo do desenvolvimento do projeto. (Imagens ilustrativas do programa, menus e acontecimentos do programa estão na secção 5, referente aos "Anexos".)

2 Cliente

Neste tópico será abordado o método através do qual se implementou as funções do cliente, de modo a que permitisse respeitar os requisitos apresentados no enunciado. Explicar-se-ão os buffers criados, as threads necessárias e todas as restantes partes desta componente. Desta forma, será evidente o papel do Client na relação que este terá com o ServerWorker, a parte através da qual é feita a comunicação com o Server.

2.1 Funcionamento e classe Client

Este cliente, como seria de esperar, corre sobre uma classe Client, que abre os Buffers in e out, correspondentes ao socket a que se vai ligar o Client quando iniciar. Também é aberto um canal de leitura do Sistema, denominado systemIn, que servirá para ler os inputs que o utilizador vai inserir na linha de comandos, a interface textual escolhida.

Depois de estabelecida a ligação e abertas as comunicações, é lido o menu inicial, no próprio método clientStart(), que espera pela mensagem esperada do servidor "3. Sair", em que se sabe já ter sido lido o menu de boas-vindas. Após isto, o utilizador deverá inserir na linha de comandos as opções disponibilizadas, sendo estas lidas e enviadas para o ServerWorker através dos canais in e out, até que seja recebida uma resposta esperado do servidor "LOGIN BEM SUCEDIDO", que indica o término do processo do menu inicial. Neste menu é possível fazer registo ou pedir saída, sendo que o registo leva sempre a uma nova apresentação do menu inicial. Se for pedido para sair, através da inserção de "quit"no teclado, esta mensagem é processada no próprio Client, fechando-se o canal de leitura do teclado e os canais do socket, bem como o próprio socket. Se o jogador não pediu para sair e fez o login com sucesso, então é inserido numa jogada, sendo enviado um conjunto de informações do ServerWorker, terminando com a informação "A sua partida começa dentro de momentos. Aguarde...".

Uma vez que o utilizador já está inserido no jogo que o espera, só tem que ficar à espera de jogadores para completar esse jogo, sendo que neste momento é iniciada uma nova thread que vai servir para ler todas as informações provenientes do ServerWorker em relação à partida. Será esta thread, que correrá uma classe Runnable denominada ClientListener, a responsável por receber informações como os campeões lecionados por determinados jogadores da equipa correspondente ao jogador que tem a conta aberta, bem como a informação de passagem de tempo, até aos 30 segundos, altura em que termina a fase de seleção de campeões.

Durante este período de 30 segundos, o jogador poderá escolher os campeões que quiser, sendo que caso escolha um campeão já utilizado vai receber essa informação e deverá escolher outro. Caso queira mudar de campeão, terá apenas que inserir o número (de 1 a 30) correspondente a esse campeão, o qual será aceite e alterado, desde que disponível. Terminados estes 30 segundos de seleção, caso os campeões de ambas as equipas tenham sido todos escolhidos, é corrido um random que gera um vencedor pseudo-aleatório, sendo que será aumentado o ranking dos jogadores pertencentes à equipa vencedora em 1, e diminuindo o mesmo à perdedora, caso tenham não mais que 9 pontos e não menos que 0, respetivamente. Por fim, o jogador deve escolher se volta para o menu, através da inserção de "menu"ou se sai, escrevendo "quit".

3 Servidor

Será aqui detalhado como se decidiu implementar a parte do Servidor, de modo a responder aos requisitos pedidos. Explicar-se-ão as classes criadas e o seu papel na representação do problema em questão. Vai ser explanado o método usado para permitir que vários clientes se conectem e interajam com o servidor, bem como é executado o controlo de concorrência das várias threads e como o ServerWorker processa todas as mensagens que recebe, em cada momento, para poder enviar mensagens úteis de volta.

3.1 Classes

Vão ser explicadas as classes referenciadas, sendo que a ServerWorker será mais aprofundada na secção Funcionamento.

3.1.1 Server

A classe Server contém o método main que inicializa o servidor. O principal objetivo desta classe é manter uma thread onde se recebem as ligações de utilizadores. Depois do servidor ser inicializado, fica à espera destas ligações e inicializa uma nova thread por cada novo utilizador, que lidará com a comunicação entre ambos a partir daquele ponto.

3.1.2 ServerWorker

Esta classe Runnable é lançada pelo Server sempre que um novo Client se liga. Por cada cliente será lançada uma thread com esta classe, permitindo ao servidor controlar uma ligação por thread. Será esta a classe a conter toda a lógica necessária ao funcionamento do jogo, sendo-lhe dedicada uma secção "Funcionamento"própria, em 3.2. O Server lançará cada ServerWorker passando-lhe alguns parâmetros.

Quando um ServerWorker necessitar de ler ou editar informação relativa aos utilizadores, e por haver condições de corrida neste caso, é necessário invocar os métodos a partir de uma referência local ao Server principal. Estes métodos da classe Server estão protegidos com a keyword *synchronized*, o que permite o controlo da concorrência a estes dados.

3.1.3 Play

Esta classe vai representar cada partida em que os utilizadores podem entrar. Armazena os Users presentes na equipa 1 e na equipa 2, nos HashMap (User, Integer) team1 e team2, sendo o Integer o boneco escolhido pelo User, no final do período de seleção. Existe também um HashMap (User, BufferedWriter) clients, que associada a cada User de cada equipa o respetivo BufferedWriter do server para ele. Os HashMap team1 e team2 serão necessários para fazer todas as operações como escolha de heróis e mudanças nos rankings, e o clients será necessário para proceder à difusão de mensagens na fase de escolha de campeões.

"winningTeamRandom()"é o método que é lançado aquando da criação de uma nova partida, para decidir qual a equipa vencedora.

"addPlayer()"adiciona o utilizador recentemente logado a esta partida, que foi identificada como adequada para a sua entrada.

"isPlayFull()"vai permitir lançar o lobby de seleção de campeões quando todos os jogadores estiverem prontos.

"chooseChampion()"permitirá verificar se determinado campeão pode ser escolhido no momento de seleção de campeões.

"allChampionsPicked()"será necessário para, no fim dos 30 segundos de seleção, verificar se todos os jogadores têm um campeão destinado.

"teamcast()"é necessário para o envio das mensagens relativas a qual campeão aquele jogador escolheu para o resto da equipa.

"broadcast()"faz o mesmo que teamcast() mas para todos os jogadores, independentemente da equipa. Será usado para informar o tempo decorrido na seleção de campeões.

"rankingUpdate()"permitirá melhorar/piorar o ranking dos jogadores, de acordo com o resultado obtido.

O controlo de concorrência foi mais uma vez garantido através da utilização de synchronized nos métodos que efetuam operações sobre as estruturas de dados da class Play.

3.1.4 User

Guardará o username e a password de cada utilizador no sistema, bem como o seu ranking.

3.1.5 Game

Guarda HashMaps plays e closedPlays, contendo plays de acordo com o Ranking da partida.

O HashMap plays contém as partidas de um determinado rank que estão em fase de construção, isto é, as partidas que ainda não têm um número suficiente de jogadores para seguirem para fase de seleção do campeão. Quando entram dez jogadores numa partida esta é retirada do plays e colocado no closedPlays. Em termos de controlo de concorrência feito, usamos synchronized ao nível dos métodos que acedem à classe, isto porque muitas threads vão estar a aceder simultaneamente à instância de Game presente no Server principal.

3.1.6 ChampionsLobby

Classe Runnable que é lançada numa thread no ServerWorker, que permitirá aos jogadores fazerem a seleção dos seus campeões. Serão chamados métodos de outras classes para que se verifique a possibilidade de chamar aqueles campeões, a determinada altura.

3.2 Funcionamento

Depois de lançado o ServerWorker, aquando da conexão do Client ao Server, inicia-se o envio de uma mensagem de boas vindas, seguindo-se o lançamento do método launchMenu(), que irá receber as seleções dos utilizadores, podendo estes fazer login, registo ou sairem. No caso de quererem fazer login, é lançado o método logIn() em que é pedido o username e a palavra passe, sendo feita a verificação. Caso falhe, será apresentado novo menu inicial. No caso de registo, é lançado register() que fará a aceitação de um novo User.

Depois de concluido o login, o jogador é introduzido numa jogado de acordo com o seu ranking, sendo que caso ainda não haja nenhuma jogada disponível para este, é criada uma nova. Seguidamente, são transmitidas informações ao jogador sobre a partida em que está inserido e é pedido para aguardar por outros jogadores, até que o limite de 10 jogadores seja atingido.

Uma vez atingido este limite, verificado por isPlayFull(), é informado o utilizadorque deve selecionar um campeão de 1 a 30, sendo lançada a thread responsável por coletar e gerir essa seleção, lançando a classe ChampionsLobby (Runnable). É então iniciado um período de espera de 30 segundos, findo o qual é enviada uma mensagem de fim de período de escolha e a thread responsável pela escolha de campeões é parada.

Por fim, é verificado se todos os jogadores têm campeões e dada a conhecer a equipa vencedora, atualizando-se os rankings dos participantes. É assim finalizada a partida.

4 Conclusões

Este projeto foi de extrema importância e enriquecedor, pois permitiu aos membros do grupo perceber e interiorizar melhor os conceitos abordados nas aulas práticas de Sistemas Distribuídos.

No que toca ao desenvolvimento do Client, o grupo sentiu-se mais à vontade, uma vez que se caracteriza uma classe mais simples e de melhor compreensão, não sendo necessário controlo de concorrência, que usualmente complicam o programa. Apenas foram necessários conhecimentos relativos a *Sockets* TCP, e perceber que são uma maneira de comunicaçãoo eficiente.

Quanto ao Server, um dos grandes obstáculos seria a divisão das classes de forma eficiente, o controlo de concorrência implícito, e a implementação dos trinta segundos propostos no enunciados para escolha de heróis. Na verdade, a dificuldade apenas residiu em como se iria chegar à solução, dado que uma vez alcançada, o desenvolvimento das classes foi relativamente simples e intuitivo.

Por último, apesar das falhas no controlo da concorrência a implementação das funcionalidades básicas deste sistema foram conseguidas com sucesso e o trabalho de semanas por fim deu frutos. O grupo conseguiu tirar partido dos conhecimentos adquiridos neste projeto, sentido-se capaz de, num contexto futuro e real, aplicar os conceitos subjacentes de forma eficaz.

5 Anexos

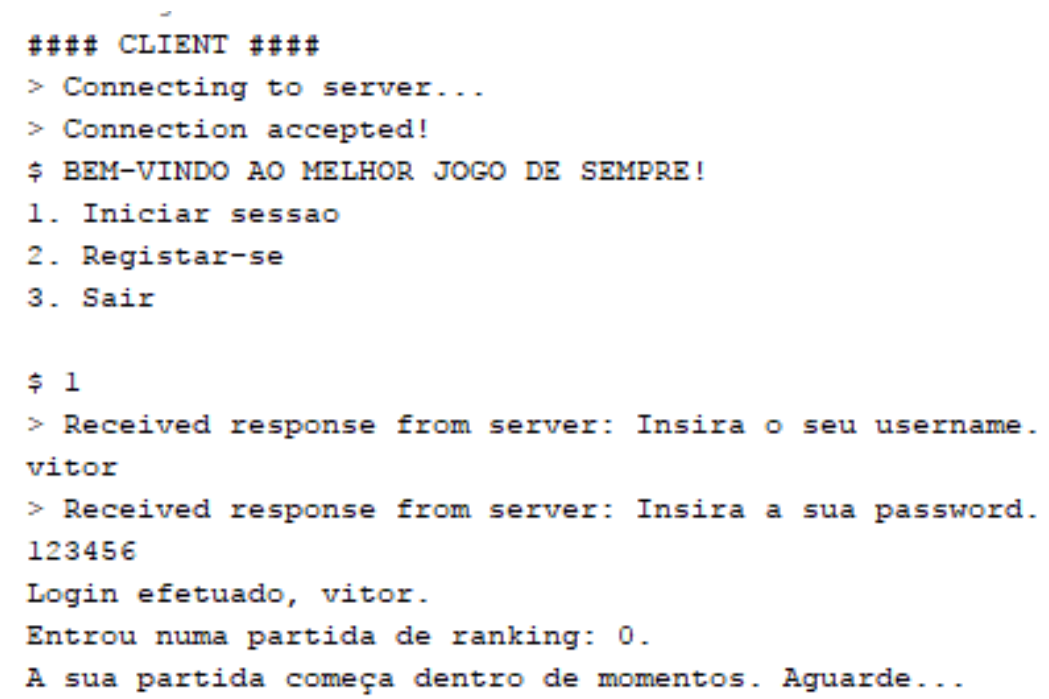


Figura 1: Menu inicial e login de um jogador bem sucedido, que este espera até partida ficar completa.

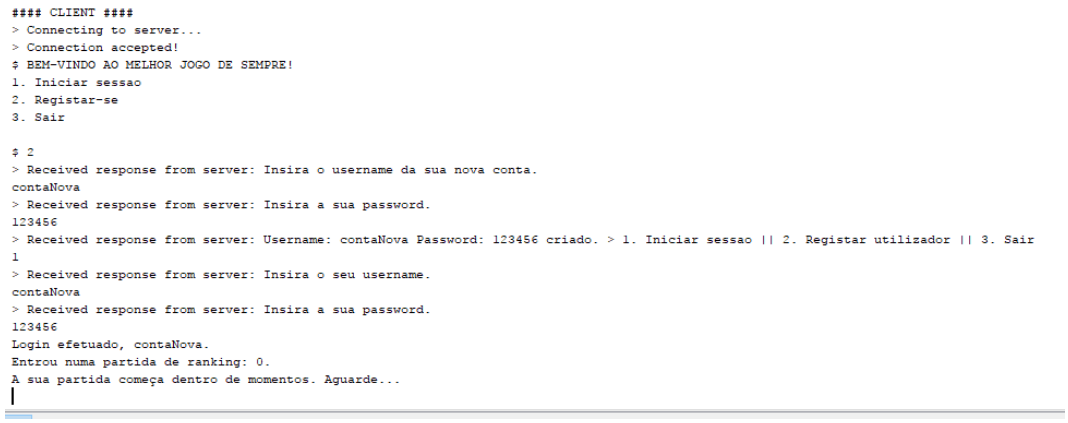


Figura 2: Menu inicial e registo de um novo jogador, seguido de login.

```
#### CLIENT ####
> Connecting to server...
> Connection accepted!
$ BEM-VINDO AO MELHOR JOGO DE SEMPRE!
1. Iniciar sessao
2. Registar-se
3. Sair

$ 3
> Received response from server: Escreva 'quit' para sair.
quit
Até à próxima!
BUILD SUCCESSFUL (total time: 49 seconds)
```

Figura 3: Menu inicial e saida de um jogador da plataforma, no Cliente.

```
#### SERVER ####
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
Worker-1 sended Welcome Message.

Worker-1 > Received message from client: 3
Worker-1 asked for QUIT COMMAND.
Worker-1 showed MENU.

Worker-1 > Client disconnected. Connection is closed.
```

Figura 4: Resposta do Servidor à saída de um jogador (cliente) da plataforma.

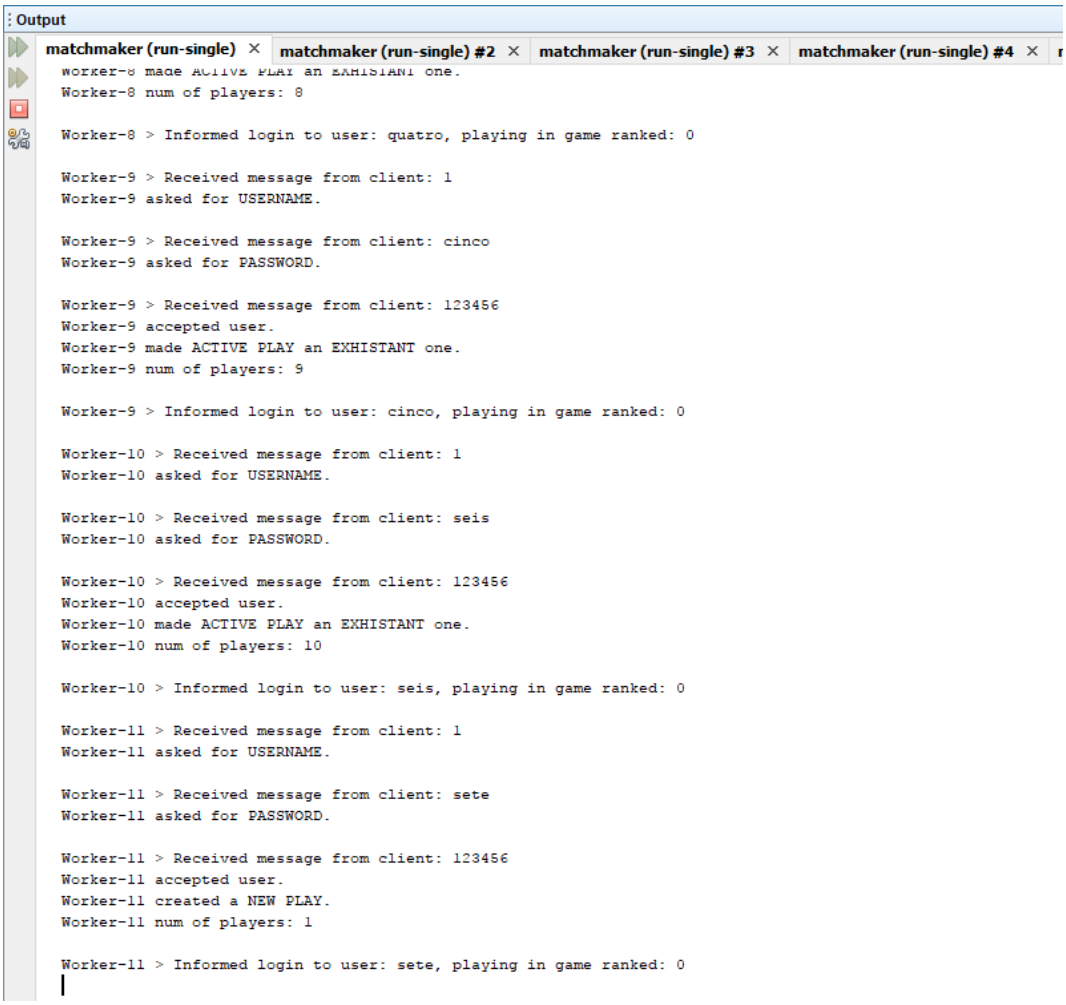


Figura 5: Exemplo de um momento em que vários jogadores efetuam os respetivos logins para depois poderem jogar, na vista do Servidor.

```
A sua partida começa dentro de momentos. Aguarde...
Selecione o seu jogador (de 1 a 30). Tem 30 segundos para o fazer!
marcos escolheu o campeão 3
Passaram 5 segundos.
Passaram 10 segundos.
Passaram 15 segundos.
Passaram 20 segundos.
Passaram 25 segundos.
Terminou o tempo!
Alguém não escolheu o seu campeão. Até já!
Escreva quit para sair.
```

Figura 6: Exemplo em que terminaram os 30s de escolha de heróis e um jogador não selecionou a sua personagem.

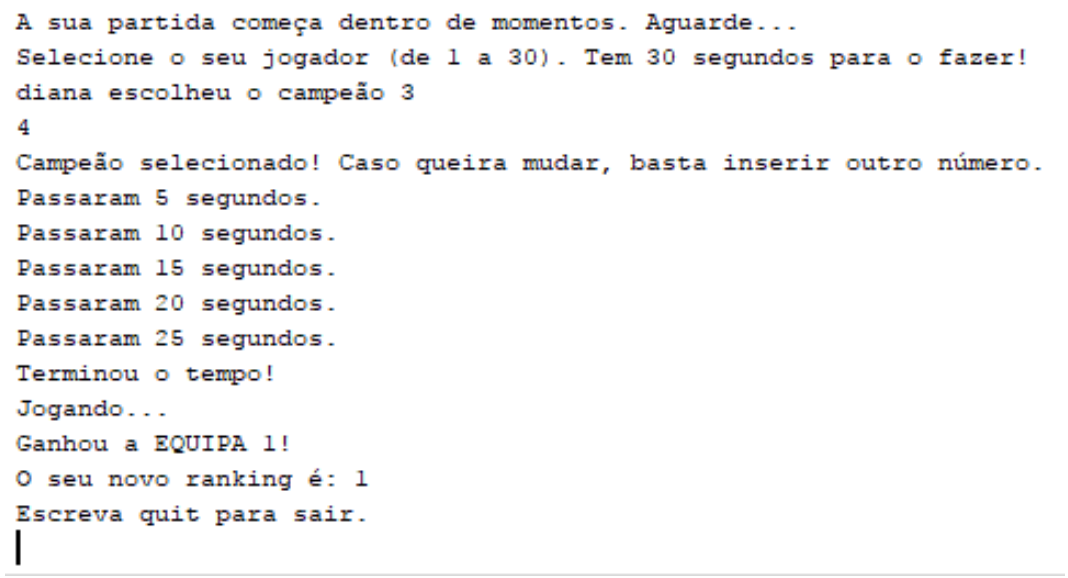


Figura 7: Atualização do rank de um jogador, após conclusão de uma partida.

```
compile-single:
run-single:
#### SERVER ####
ServerMain > Server is running waiting for a new connection...
Client seis starting.
Client cinco starting.
Client um starting.
Client dois starting.
Client tres starting.
Client diana starting.
Client oito starting.
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
Client a starting.
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
Client b starting.
Client c starting.
Client marcos starting.
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
Client e starting.
ServerMain > Connection received! Create worker thread to handle connection.
ServerMain > Server is running waiting for a new connection...
Client quatro starting.
```

Figura 8: Iniciando servidor e múltiplos clientes.

```
Worker-11 > Received message from client: 123456
Worker-11 accepted user.

Worker-11 > FIM MENU ENTRADA


Worker-11 > ARRANGING PLAY

Worker-11 num of players: 4

Worker-11 > Informed login to user: marcos, playing in game ranked: 0

Worker-37 > Received message from client: 123456
Worker-37 accepted user.

Worker-27 > Received message from client: q

Worker-37 > FIM MENU ENTRADA


Worker-37 > ARRANGING PLAY

Worker-27 asked for PASSWORD.
Worker-37 num of players: 5
```

Figura 9: Login com vários jogadores e inserção em jogada.

```
Worker-14 > ASKED to CHOOSE CHAMPION.
Worker-23 > ASKED to CHOOSE CHAMPION.
Worker-21 > ASKED to CHOOSE CHAMPION.
Worker-2 > ASKED to CHOOSE CHAMPION.
Worker-8 > ASKED to CHOOSE CHAMPION.
Worker-30 > ASKED to CHOOSE CHAMPION.
Worker-13 > ASKED to CHOOSE CHAMPION.

Worker-36 > Received message from client: 26

Worker-6 > Received message from client: 25
Worker-36 > Player x choosed champion: 26
Worker-36 > Teamcasted with: 26
Worker-6 > Player seis choosed champion: 25
Worker-6 > Teamcasted with: 25

Worker-34 > Received message from client: 10
Worker-34 > Player v choosed champion: 10

Worker-26 > Received message from client: 1
Worker-26 > Player p choosed champion: 1
Worker-26 > Teamcasted with: 1

Worker-14 > Received message from client: 25

Worker-14 > Informed UNAVAILABLE CHAMPION to: f

Worker-28 > Received message from client: 5
Worker-28 > Player r choosed champion: 5
Worker-28 > Teamcasted with: 5

Worker-21 > Received message from client: 21
Worker-21 > Player l choosed champion: 21
Worker-21 > Teamcasted with: 21

Worker-2 > Received message from client: 28
Worker-2 > Player oito choosed champion: 28
Worker-2 > Teamcasted with: 28

Worker-23 > Received message from client: 13
Worker-23 > Player n choosed champion: 13
```

Figura 10: Vários jogadores a escolherem o seu jogador simultaneamente.

```
Worker-26 > Player p choosed champion: 25
Worker-26 > Teamcasted with: 25
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.
Passaram 5 segundos.

Worker-2 > Received message from client: 25

Worker-2 > Informed UNAVAILABLE CHAMPION to: oito

Worker-19 > Received message from client: 23
Worker-19 > Player k choosed champion: 23
Worker-19 > Teamcasted with: 23

Worker-29 > Received message from client: 15

Worker-29 > Informed UNAVAILABLE CHAMPION to: t
```

Figura 11: Vários jogadores a serem informados do tempo passado.