Relatorio Técnico
Programação distibuída - 2017/2018

Maurizio Crocci
21230268

1. Introdução

Este documento irá incidir como foram abordados nomeados aspectos relativos ao trabalho prático de Programação Distribuída, que consistia em desenvolver um sistema distribuído em java que permitisse gerenciar utilizadores e jogar 3 em linha.

1.1. Funcionalidades não realizadas

- Servidor de jogo.
- Mecanismo de comunicação entre utilizador e servidor de jogo.

1.2. Comunicação

As ligações entre os vários componentes do sistema foram realizadas de acordo com o enunciado do trabalho.
Os utilizadores comunicam através de ligações TCP com o servidor de gestão, que por sua vez, só comunicaria com o servidor de jogo através do HeartBeatReceiver.

2. Organização

2.1. Classes

2.2.1. User

Representa um utilizador do sistema.

Métodos

- public static String readCommand() throws IOException
  Lê da linha de comandos o que o utilizador escreveu

- public static UserRequest parseCommand(String cmd)
  Segundo o que o utilizador escreveu cria um request a mandar ao servidor de gestão.

- public static void main(String[] args)
  - args[0] => ip do servidor de gestão.
  - Args[1] => porto do servidor de gestão.

Começa por verificar se foram passados o ip e port do servidor de gestão, se não tiverem sido, pede-os ao utilizador.

De seguida cria a socket para comunicar com o servidor e inicia o seu loop infinito.

Este loop consiste em ler um comando da consola, dar parse desse commando, e enviá-lo.

Espera por uma resposta se segundo o tipo de resposta (String ou UserRequest) apresenta-a no ecrã. No caso de ser UserRequest quer dizer que é um pedido de confirmação, no qual pede ao utilizador para o fazer.

2.2.2. UserRequest

Representa um pedido do utilizador ao servidor de gestão.
Pode ter os seguintes tipos defindos por constantes:

1. REGISTER_REQUEST
2. AUTHENTICATE_REQUEST
3. SHOW_PLAYER_LIST_REQUEST
4. SHOW_PAIR_LIST_REQUEST
5. PAIR_REQUEST
6. ACCEPT_REQUEST
7. DENY_REQUEST
8. ASK_PAIR_REQUEST
9. PLAYER_MESSAGE_REQUEST
10. MESSAGE_REQUEST
11. DISCONNECT_REQUEST

Métodos

Setters e getters para os campos:

1. private int type;
2. private String username;
3. private String password;
4. private String pairUsername;
5. private String msg;

2.2.3. ManagementServer

Representação o servidor de gestão.

Implementa a interface Runnable de modo a permitir responder a cada cliente numa thread diferente.
Mantem a socket do cliente que está a ser respondido, uma ligação à base de dados e uma lista de todos os clientes ligados ao servidor.

Métodos

- public static void main(String[] args)

  Servidor começa por aqui. Inicia por criar e validar a ligação com a base de dados.

  Cria um HeartbeatReceiver, coloca-o numa thread, e inicia-o.

  Inicia uma loop infinito que cada vez que recebe uma nova ligação cria uma thread para atender o novo cliente.

- public ManagementServer(Socket client, Connection conn, HeartbeatReceiver hb, ArrayList<Socket> clients)

- public void run()

  Lógica de atendimento de um cliente.

  Loop infinito que lê o request do cliente, executa as acções que este lhe solicitou, construíndo uma resposta no processo, e responde ao cliente.

- private void registerUser(String username, String password) throws SQLException

- private Boolean authenticateUser(String username, String password) throws SQLException

- private String getPlayerList() throws SQLException

- private String getPairsList() throws SQLException

- private void createUnconfirmedPair(String username, String pairusername) throws SQLException

- private void confirmPair(String username, String pairusername) throws SQLException

- private void denyPair(String username, String pairusername) throws SQLException

- private Pair<String, Integer> getUserAddress(String username) throws SQLException
  Devolve um ip e porto de um dado username.

- private ArrayList<Pair<String, Integer>> getAllAddresses() throws SQLException
  Devolve uma lista de ips e portos de todos os usernames autenticados.

- private Socket getClientSocket(String username) throws SQLException
  Devolve a socket correspondente a um dado username.

- private void disconnectClient() throws SQLException
  Desautêntica o cliente.

2.2.4. HeartbeatReceiver

Representa uma thread que correrá num servidor de gestão quee comunica através de uma ligação UDP com outro servidor do tipo jogo.

Contem uma classe Heartbeat privada que contem os counters:

- beatCounter => cada vez que recebe um heartbeat do mesmo servifor de jogo, aumenta este tick. O servidor que tiver mais beats é considerado o mais antigo.
- counter => a cada intervalo aumenta este counter. Quando chega a 3 apaga o servidor da lista. Dá reset cada vez que o receiver recebe um heartbeat novo do servidor correspondente.

Métodos

- public HeartbeatReceiver(String databaseAddress) throws SocketException

  Construtor do receiver.

  Cria a socket necessária e uma task a cada intervalo (HEARTBEAT_TIME) aumenta o counter de todos os servidores de jogo.

- private void cleanGameServerList()

  Elimina da lista de servidores aqueles que têm o counter maior ou igual que 3.

- public String getCurrentGameServer()

  Devolve o server com mais beats.

- public void run()

  Recebe um packet via UDP. Se este conter HEARTBEAT_REQUEST cria um hearbeat. Se já existir uma hearbeat no servidor com o mesmo address aumenta-lhe o beatCounter e dá reset ao counter.

  Responde ao servidor de jogo com o address da base de dados.