Universidade do Minho
Escola de Engenharia

Relatório

UNIDADE CURRICULAR DE SISTEMAS DISTRIBUIDOS

Grupo 36

janeiro 2020

Carlos Afonso A82529

Gonçalo Nogueira A86617

Índice

-Introdução

No âmbito da unidade curricular de Sistemas Distribuídos foi proposto aos alunos que desenvolvessem uma plataforma para a partilha de ficheiros de música sob a forma de cliente/servidor em Java com utilização de sockets e threads.

Dividimos o projeto em 3 partes distintas. A primeira parte do projeto está relacionada com a autenticação e registo de utilizadores no servidor. A segunda envolve o upload e download de ficheiros de música entre clientes e o servidor.

Por fim a terceira parte é a procura de músicas na plataforma enviando etiquetas para o servidor que devolve uma lista de músicas relacionadas com a etiqueta pretendida pelo servidor.

-User

A classe User é a classe que contém toda a informação relativa ao Utilizador, o seu nickname, a sua palavra passe e uma variável que diz se ele já esta logado no sistema ou não, para evitar que o mesmo condutor faça login duas vezes.

```
public class User implements Serializable {


    private String nick;
    private String pass;
    private Boolean ativo;
```

Figura 1 - Classe User

-Server

A classe Server é onde se tratam novas ligações de clientes ao servidor e inicia-se os dados da aplicação. Sempre que existir uma nova conexão ao servidor, este cria uma *thread*, onde lhe passa como parâmetros o *socket* do cliente associado e a ligação aos dados que estão na memória.

Contem também as variáveis de instância de todas as ações possíveis do utilizador, que serão abordadas pela classe Handler, mais à frente apresentada.

```
public class Server {
    private static final int porta = 55000;
    private static SoundCloud k = null;

    //Acções do Utilizador
    public static final String REGISTAR_Utilizador = "RegistarUtilizador";
    public static final String NOME_Utilizador = "NomeUtilizador";
    public static final String PW_Utilizador = "PasswordUtilizador";
    public static final String ENTRAR_Utilizador = "EntrarUtilizador";
```

Figura 2 – Classe Server

-Cliente

A classe Cliente é onde está implementada a interface para o utilizador. Esta conta com uma série de variáveis que permitem a interação com o utilizador e as ligações ao servidor através da classe Handler.

```
public class Cliente {
    private final static int porta = 55000;
    private final static String ip = "localhost";
    private static Socket clienteSocket;

    public static Scanner in = new Scanner(System.in);
    public static ObjectOutputStream o = null;
    public static ObjectInputStream i = null;
    public static HashMap<String, String> hash;
    public static String nick = null;
    public static PacoteDados p;
    public static OutputStream os = null;
    public static InputStream is = null;
```

Figura 3 - Classe Cliente

As variáveis ip, port e clienteSocket (Socket) são os elos de ligação ao servidor (Server).

As variáveis **o, i e p** são as variáveis que permitem a comunicação entre o utilizador e o servidor. A variável **p** guarda a ação desejada pelo utilizador, sendo enviada para a classe handler, que vai tratar desse informação, que depois irá enviar para a classe SoundCloud. A variável **o** permite que o cliente consiga enviar mensagens ao servidor, isto claro, com a ajuda da classe handler. A variável **i** permite que o cliente recebe mensagens enviadas do servidor, também com a ajuda da classe handler.

-Música

Contém toda a informação sobre um ficheiro de música, ou seja, os meta-dados possíveis para cada ficheiro utilizado.

```
public class Musica {

    private int id;
    private String nome;
    private int ano;
    private String interprete;
    private ArrayList<String> etiquetas;
    private int descarregada;
```

Figura 4 – Classe Musica

-Pacote de Dados

A classe Pacote de Dados foi criada para conseguir fazer chegar ao Handler a intenção do Utilizador e os argumentos necessários para que essa intenção possa vir a ser concretizada. Mediante a intenção do Utilizador, o Handler enviará essa informação para a classe SoundCloud.

Contém a variável ação, onde é registada a ação que é pretendida, um HashMap onde serão passados os argumentos necessários para tal ação, isto é, se a key for “REGISTAR_Utilizador”, sabemos, então, que a ação é registar um Utilizador. Contém, também, um array de bytes para o envio de música, e um ArrayList onde serão guardadas as etiquetas.

```
private final String accao;
private final HashMap<String,String> argumentos;
private byte[] musicaBytes;
private ArrayList<String> etiquetas;
```

Figura 5 – Classe Pacote de Dados

-Handler

A classe Handler funciona como uma ponte entre a Classe Cliente e a classe Servidor. Tem como principal objetivo tratar individualmente de um utilizador e os seus pedidos. Quando um utilizador faz um pedido ao servidor, o pacote de dados enviado é tratado nesta classe e mediante esse pedido, chamará os métodos do SoundCloud necessários para a realização desse pedido.

```
public class Handler implements Runnable {
    private final Socket s;
    private final SoundCloud sk;
    private ObjectInputStream sInput;
    private PrintWriter sOutput;
```

Figura 6 - Classe Handler

-SoundCloud

Por fim, na classe SoundCloud, podemos encontrar os métodos base do Soundcould, que serão invocados a partir do Handler. É aqui também que será guardada toda a informação relativamente aos utilizadores e músicas.

No sentido de garantir um controlo de concorrência, fazemos uso de lock's e dos respetivos unlock's. Optamos por usar Locks explícitos, para depois podemos usar variáveis de condição de maneira a evitar esperas ativas dos vários processos.

Também implementámos a variável "nrSongsDownload", com uso Apenas para impedir que muitos utilizadores possam fazer Download's, de forma a evitar alguma "lentidão" do servidor.

```
public class SoundCloud implements Serializable {
    private Map<String, User> Utilizador;//Todos os Utilizadores registados
    private Map<String,Musica> Musicas;//Todas as musicas no SoundCloud


    private int nrSongsDownload;
    final int MaxUpload = 3;



    private Lock lock = new ReentrantLock();
    //variável condição usada para quando não há musicas
    private Condition naoMusicas = lock.newCondition();
    //variável condição usada para quando há downloads em excesso
    private Condition naoDownloads = lock.newCondition();
```

Figura 7 -Classe SoundCloud

Conclusão

Chegado o momento final deste projeto prático de Sistemas Distribuídos, podemos assumir com segurança, que novas competências foram adquiridas e muitas outras, por certo, desenvolvidas.

Ficamos claramente a perceber melhor o funcionamento de um sistema baseado em comunicação entre Cliente e Servidor através de *Sockets*, isto é, um sistema que aceite a conexão simultânea de múltiplos clientes. A forma como um Servidor trata os dados e pedidos enviados por Cliente, e a maneira como se ligam.

Com a elaboração deste servidor, conseguimos perceber agora as vantagens de usar *threads,* em qualquer sistema que envolva vários utilizadores a usar a mesma aplicação.

Como não poderia deixar de faltar, na vertente mais relacionada com a UC, o controlo de concorrência foi um desafio para nós, e que agora, depois de realizado o projeto, tornou-se clara a importância dessa implementação.

Para terminar, apesar de o grupo ser constituído apenas por dois elementos, ficámos agradados com o resultado final, reconhecendo que haveria espaço para melhoria em algumas vertentes.