Universidade do Minho
Mestrado Integrado em Engenharia Informática

Sistemas Distribuídos

Projeto Sistemas Distribuídos
Grupo 46

Janeiro 2020


Ana Margarida Campos
(A85166)


Ana Catarina Gil
(A85266)


Joana Afonso Gomes
(A84912)


Tânia Rocha
(A85176)

Contents

Introdução

Este projeto foi proposto pelos docentes da Unidade Curricular de Sistemas Distribuídos, tendo como principal objetivo desenvolver uma plataforma de troca de ficheiros de música.
Ao longo deste relatório especificaremos os métodos seguidos e as decisões tomadas com fim a concretizar o *upload* e *download* dos ficheiros de música, bem como autenticação e registo do utilizador, entre outros procedimentos.

Funcionalidades do sistema

2.1 Criar Conta

Com fim a efetuar o registo de um utlizador no servidor, é necessário receber o seu *username* e uma *password*. O nosso programa verifica se o *e-mail* não está a ser utilizado e, caso não esteja, adicionamos ao *HashMap* de contas de utilizadores presente na classe *SkyBeat* (caso contrário, é lançada uma exceção).

2.2 Iniciar Sessão

Para efetuar o *login* de um *user* no servidor tem que ser dado um e-mail e uma password, dados que o sistema irá verificar se se encontram armazenados no *HashMap* de contas de utilizador da classe SkyBeat. Caso estejam, o sistema verifica se essa password corresponde ao *e-mail* dado.

2.3 Upload

Com vista a carregar um ficheiro de música para o servidor, como proposto no enunciado, realizamos a conversão do ficheiro .mp3 para um *array* de *bytes*, para depois esses *bytes* voltarem a ser convertidos num novo ficheiro .mp3 na pasta do nosso projeto. Para isso, o utilizador envia as informações necessárias para a criação da nova música (o *path* da sua localização, um título, um intérprete, um ano de lançamento e os filtros que serão utilizados para pesquisar por essa música (etiquetas).

2.4 Download

Opondo-se ao que acontece no *Upload*, no *Download* é criado um *array* de *bytes* a partir do ficheiro no nosso sistema, para posteriormente ser convertido num novo ficheiro numa diretoria local escolhida pelo utilizador. Com vista a isso, o *user* seleciona a música de que quer fazer *download* a partir do seu ID, assim como o *path* de destino (onde quer armazenar a música na sua máquina).

2.5 Procura com etiquetas

A cada música estão associadas etiquetas que descrevem o tipo/conteúdo das mesmas. A nossa funcionalidade de procura por etiquetas permite o utilizador procurar músicas que estejam assinaladas com essas mesmas *labels*.

2.6 Limite de descargas

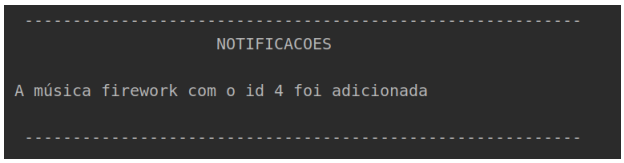
Com fim a testar a existência de um máximo de *downloads* por um utilizador, foi criada uma classe **BotDown**. Esta classe utiliza *threads* para criar utilizadores, que apenas conseguem efetuar a função de *download*. Para limitar o número de *downloads* criamos uma variável estática MAXDOWN (int) com o valor de 3, ou seja, só podem ocorrer 3 descargas simultaneamente. Quando já se encontram a ser efetuadas 3 descargas ao mesmo tempo, as *threads* que requerem o *download* ficam em espera (*wait*)até que alguma das *threads* que estavam a efetuar o *download* as notifique que já acabou a descarga (*notify*). Para tal usamos a *Condition* esperaDownload.

```
Solicitei download
Solicitei download
Solicitei download
Solicitei download
Ficou à espera
Solicitei download
Ficou à espera
File is download successfully!
firework
Ficou à espera
File is download successfully!
firework
File is download successfully!
firework
File is download successfully!
firework
File is download successfully!
firework

Process finished with exit code 0
```

2.7 Notificações

Cada vez que um utilizador faz *upload* é necessário notificar todos os utilizadores do sistema. Para isso, guardamos a informação de quando o *upload* é realizado e apenas é notificado a todos os *users* quando estes efetuam *login*.



Cliente

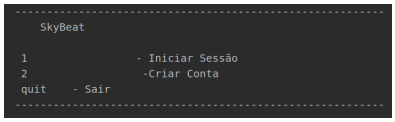
A classe Cliente tem um *socket* que é conectado a partir da classe Conexao que recebe a ligação do *ServerSocket* do servidor. A comunicação do sistema com o cliente é conseguida através da utilização de um *BufferedReader* e de um *PrintWriter*. A conexão é terminada após ser digitado o comando *quit*.

Servidor

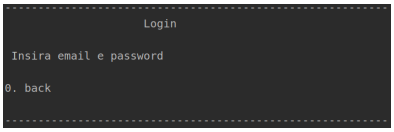
A nossa classe Servidor fica à espera de uma conexão através de um *socket*, criando uma *Thread* para cada cliente que efetue uma ligação. Esta classe possui as seguintes estruturas de dados que são criadas pela *main* do Servidor aquando da criação desta mesma *thread*:

- **Socket** – *socket* TCP relativa à ligação do cliente ao servidor;
- **SkyBeat** – classe referente ao Programa que será utilizado;
- **Execute** – classe com os métodos que controlam os dados do nosso programa;
- **Conexao** – classe que permite aa conexão do socket do servidor com o socket do cliente;

Menus

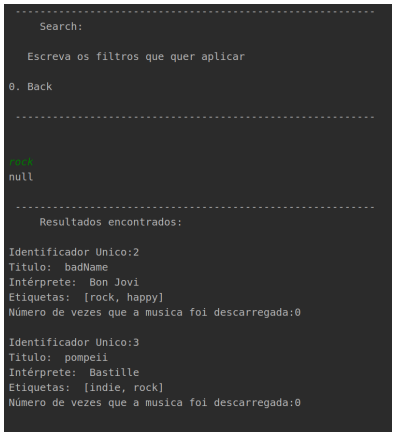


Menu Inicial



Menu Login

Menu de Utilizador



Procura Por Filtros

Upload de musica

Download de musica

Conclusão

Neste projeto conseguimos por em prática os conhecimentos obtidos nas aulas desta UC.
Com a sua realização, conseguimos ter um conhecimento mais abrangente e uma maior destreza na utilização de *threads*, paralelamente aprofundando a capacidade de entender como gerir concorrência e acesso a dados de uma maneira segura entre um servidor e os vários clientes que a ele se querem conectar.

Durante o seu desenvolvimento deparamo-nos com algumas dificuldades que foram, na maioria, ultrapassadas. Entre elas, assinalá-mos a utilização de *sockets*.