Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Sistemas Distribuídos

Ano Letivo de 2019/2020

Trabalho Prático

Troca de Ficheiros

Grupo 60:

Flávio Martins – a65277

Ricardo Lopes – a72062

Mário Santos – a70697

Índice

1. Introdução

Neste trabalho prático é pretendido implementar uma plataforma para partilha de ficheiros de música, na aplicação deve ser possível carregar ficheiros de música acompanhados de meta-informação(título, intérprete, ano e etiquetas) ao adicionar uma nova música os restantes utilizadores devem ser notificados da adição da mesma ao sistema, além de serem capazes de pesquisar as músicas da aplicação pela sua informação podendo então descarregá-los.

Tendo em conta a dimensão dos ficheiros a serem trocados, normalmente com vários MB o sistema deve ter em atenção aos recursos utilizados para armazenamento, manipulação e transmissão dos ficheiros. Neste projeto é requerido um cuidado especial ao nível da transmissão, isto é deve ser tido o cuidado de evitar uma quantidade de operações simultâneas que sobrecarreguem o sistema mas além disso tentar criar um sistema justo, isto é que todos os utilizadores consigam realizar operações não ficando bloqueados.

Por fim é requerido que os utilizadores possam interagir , usando um cliente escrito em Java, intermediados por um servidor multi-threaded também escrito em Java, recorrendo a comunicação via sockets.

2. Desenvolvimento

2.1 Server

2.1.1 Server

Ao iniciar o server este irá verificar a existência do diretório para os ficheiros do mesmo criando o mesmo caso não encontre, de seguida cria uma nova library, que conterá os utilizadores e músicas do sistema.

Além disso cria um ServerSocket. De seguida cria um novo socket na mesma porta e fica a aguardar da conexão de algum utilizador, quando algum utilizador se conecta o server cria uma nova Thread ServerWorker e continua então a aguardar por mais utilizadores.

2.1.2 ServerWorker

O ServerWorker ao inicializar cria, através do clientSocket recebido na sua inicialização, um input para receber do Client e um output para enviar, de cada vez que recebe uma linha no input este irá chamar o método execute, que recebendo uma String task irá interpretar a mensagem recebida, realizar a tarefa correspondente e devolver a resposta adequada para o Client.

2.2 Client

2.2.1 Client

Ao iniciar o Client irá estabelecer comunicação com o Servidor através dum socket, de seguida irá iniciar uma Thread ClientReceive para filtrar as mensagens de notificações, e os pipes necessários para comunicar com a mesma. Para comunicação com o Server é criado um in para receber e um out para escrever, é ainda criado um buffer que irá ler o input do utilizador.

Consoante o que o buffer receber do utilizador esta classe vai interpretar o input interagindo com o utilizador, apresentando as opções disponíveis ao utilizador e os vários passos quando o utilizador decidir realizar alguma operação, quando tiver toda a informação necessária para alguma operação irá reencaminhar a mesma pelo out e aguardar a resposta do server apresentando o resultado então ao utilizador.

2.2.2 ClientReceive

Ao iniciar um Client é iniciado um ClientReceive, este irá estar responsável pelas notificações, imprimindo para o utilizador quando se tratar duma notificação e encaminhando para o Client nos restantes casos.

2.3 SoundCloud

2.3.1 Library

A classe Library contem um map com os utilizadores e musicas, e ainda um map com os downloads, de cada utilizador, em progresso e pendentes , nesta classe estão definidos algumas funções necessários da aplicação como add e get de utilizadores e músicas, e ainda tem definido a função de download e obtenção de listas de meta-informação de músicas para responder às pesquisas permitidas na aplicação ao cliente.

2.3.2 MusicMeta

A classe MusicMeta é uma simples classe de modo a poder identificar a Musica com todos os gets e sets necessários para ela

2.3.3 User

A classe User é uma simples classe de modo a poder identificar o Utilizador com todos os gets e sets necessários para ela

2.4 Common

2.4.1 FileHelper

Na classe FileHelper são definidos métodos auxiliares à leitura e escrita de ficheiros. Métodos estes usados pelas classes Client, ServerWorker e Library para poderem transferir os ficheiros sem que seja necessário ter os mesmos em memória. Existe um método que n bytes de um ficheiro dado um ficheiro, indice e número de bytes a ler. Outro método que dado um FileOutputStream e uma string em forma hexadecimal, transforma o hexadecimal em bytes e acrescenta ao ficheiro, e ainda um método que transforma um array de bytes numa string hexadecimal.

3. Conclusão

O desenvolvimento deste trabalho permitiu consolidar os conhecimentos adquiridos na Unidade Curricular de Sistemas Distribuídos, nomeadamente sobre Multithreading realçando o controlo de concorrência e estabelecimento e tratamento de comunicação servidor/cliente.

Na generalidade acreditamos ter conseguido realizar o projeto utilizando maioritariamente os conteúdos abordados na Unidade Curricular, terminando o projeto com uma aplicação capaz de manipular bem a informação em fluxo constante entre o servidor e vários clientes simultâneos, além de criar as condições adequadas a que o acesso aos mesmos objetos por parte de vários utilizadores mesmo que simultaneamente não tivesse problema nem criasse qualquer erro ou perda de informação devido a uma proteção adequada das zonas criticas utilizando as locks necessárias.