Universidade do Minho

Mestrado Integrado em Engenharia Informática
Sistemas Distribuídos

Grupo 01

João Carlos Delgado Monteiro

Introdução

Neste trabalho foi proposta a criação de um serviço de partilha de ficheiros de música baseado em conceções entre um servidor e diversos clientes. Em todo o processo de desenvolvimento, esteve sempre como base os conceitos aprendidos nas aulas de Sistemas Distribuídos, onde foi utilizado mecanismos para regular o acesso à memória partilhada por processos e linhas de execução concorrentes.

No entanto, foi sempre meu objetivo minimizar a utilização dos mesmos para, sem deixar de garantir o correto funcionamento do serviço, não tornar lento a latência de resposta do mesmo.

Para usufruir totalmente do serviço, os utilizadores devem estar registados e autenticados pelo servidor no sistema.

Para iniciar o servidor, executar 'java SoundShare'.

Para iniciar uma sessão de cliente, executar 'java Cliente'.

Desenvolvimento

- **Classe User**

Classe responsável por guardar a estrutura interna de um utilizador do sistema. Para além de conter o *username* e a *password* do utilizador, e também um *boolean ligacao* que atua como uma flag a indicar se tal utilizador já se encontra ligado ao servidor, guarda também numa *List<MusicFile>* todas as músicas descarregadas por tal utilizador.

De salientar que cada ficheiro descarregado por um utilizador, a sua gerência (mantê-lo ou eliminá-lo) fica a cabo desse mesmo utilizador, mas fora do sistema. O sistema apenas guarda o registo (histórico) dos meta-dados de cada um dos ficheiros que tal utilizador descarregou, usando o serviço.

- **Classe MusicFile**

Nesta classe é manipulada toda a informação referente à um ficheiro de música: tanto o seu conteúdo, como a sua meta-informação (título, interprete, álbum, género, etc.). De salientar que é guardado também, o utilizador (*uploader*) que carrega um ficheiro para o sistema, ou seja, o ficheiro só pode ser removido do sistema pelo seu "dono". Sempre que um ficheiro é procurado e descarregado, é incrementado o número de vezes que tal foi descarregado, informação essa que é manipulada pelo método *incrementNumDownloads.* Sendo que, como já foi referido anteriormente, apenas o *uploader* pode apagar ficheiros por ele carregados, o método *hasPermision* é o que assegura essa permissa, retornando *true* ou *false*.

- **Estrutura do Cliente**

O cliente atua como um intermediário, sendo o responsável por enviar todo o input do utilizador para o servidor, tanto como por ler as mensagens do servidor e as apresentar ao utilizador através do terminal. Os dois menus de apresentação (interface para o utilizador) deste programa (menu inicial e menu para utilizadores já com *login* feito) são também da responsabilidade dessa classe cliente, para a manipulação dos diversos pedidos.

- **Estrutura do Servidor (*SoundShare*)**

O servidor também tem uma estrutura muito simples, sendo constituído por uma *thread* inicial que fica à espera na porta de comunicação escolhida para comunicar com os clientes e criar uma nova *thread* para cada linha de comunicação com um cliente. Contém também duas *HashMap's* para guardar as várias instâncias de *MusicFile* e os utilizadores registados no sistema: *HashMap<String, User>* para guardar o *username* e a instância de *User* de cada utilizador, e *HashMap<Integer, MusicFile>* em que é atribuído sempre um identificador único a cada ficheiro *MusicFile* carregado.

Sempre que é carregado um novo ficheiro no sistema por um utilizador, esse mesmo ficheiro é guardado juntamente com o id atribuído pelo sistema. Para remover uma *MusicFile*, o utilizador tem de ser quem o publicou antes no sistema.

- **ClienteHandler**

A manipulação dos diversos pedidos do cliente é levada a cabo pela *thread ClienteHandler.* Uma decisão tomada para esta classe foi a de utilizar somente *String's* como parâmetros e de retorno para os vários métodos presentes, de modo a "controlar" melhor as diversas mensagens trocadas entre Cliente-Servidor.

A assinatura de (quase) todos os métodos é *synchronized,* de forma a assegurar uma certa justiça a cada utilizador que utiliza o serviço.

- **Comunicação Utilizador - Servidor**

É primeiramente apresentado ao utilizador um menu inicial com as opções de **registar**, **login** e **sair**.

Para o utilizador se registar, é pedido o *username* e *password* desejados, sendo enviada uma mensagem ao servidor, que por sua vez, faz *parsing* da informação, retirando o email de utilizador e a password, que são guardados na *HashMap* dos utilizadores através do método ***registo*** que é *synchronized* para que apenas se registe um utilizador de cada vez sem haver conflitos. Logo que registe, é apresentado ao utilizador o menu referente às várias opções que tem ao seu serviço no sistema:

- **Upload de música(s)** - para carregar um ou mais *MusicFile's* no sistema
- **Download de música(s)** - para descarregar um ou mais *MusicFile's*
- **Procurar música(s)** - dado o critério de pesquisa (titulo, interprete, álbum,etc.), é listado todas as músicas que correspondem a esse critério
- **Ver biblioteca pessoal -** lista todas as descargas do utilizador
- **Remover música(s) –** remove a(s) música(s) que partilhou
-

De lembrar que cada um dos métodos referentes à cada pedido do utilizador, é *synchronized*, para evitar estados indeterminados no sistema, ou seja, dois utilizadores podem fazer um *upload* simultaneamente, já que são instâncias diferentes.

Conclusão e Trabalho Futuro

O desenvolvimento e conclusão deste projeto permitiu aplicar grande parte dos conceitos de controlo de concorrência e de comunicação cliente-servidor dados nas aulas de Sistemas Distribuídos.

De salientar que a manipulação do conteúdo dos ficheiros ficou em falta, o que restringiu, a meu ver, utilização de *locks* explícitos no sistema, sendo essa uma solução para o futuro.

Acredito que foi conseguido atingir parte dos objetivos propostos, produzindo uma plataforma de partilha de músicas funcional, capaz de processar pedidos de diversos utilizadores.

Em suma, e de acordo com a minha situação, fico satisfeito com o resultado final. O tempo dedicado ao desenvolvimento da aplicação foi aproveitado para aprofundar alguns conhecimentos pertinentes sobre programação concorre.