Relatório Projecto Sistemas Distribuídos

Bruno Veloso (a78352) Pedro Gonçalves (a82313)

Rafael lourenço (a86266) Rodolfo Silva(a81716)

Braga, janeiro de 2020

Universidade do Minho
Mestrado Integrado em Engenharia Informática
Sistemas Distribuídos
(1º Semestre/2019-2020)
Grupo 10

Conteúdo

1 Introdução

Este relatório aborda os aspetos mais importantes na realização deste projeto, tais como as descisões tomadas durante o processo de construção, as estruturas que suportam o sistema entre outros.

O projeto consiste na implementação de uma aplicação Cliente/Servidor usando *threads/sockets* TCP, baseada na popular plataforma de troca de ficheiros *SoundCloud*, onde será possível aos utilizadores partilharem as suas músicas e descobrirem criações de outros utilizadores.

2 Classes do Sistema

Neste capítulo será feita uma breve descrição das classes utilizadas na implementação do sistema, bem como dos principais componenetes de cada uma.

2.1 User

Esta classe tem como objetivo caracterizar um utilizador no sistema, e serve para armazenar a sua informação pessoal (nome de utilizador e palavra-passe) e ainda se o mesmo se encontra *online* ou não.

2.2 Musica

Tem como função representar uma música perante o sistema, armazenando os meta-dados das mesmas, tais como o seu nome, o artista que a criou, o número de vezes que foi descarregada entre outros.

2.3 SoundSky

Classe principal do sistema, pois é esta classe que armazena/gere todas as músicas e utilizadores pertencentes ao sistema, e é também aqui que são geridas quaisquer alterações que possam acontecer ao longo do tempo, tal como a atualização do número de descargas das músicas, entre outros.

2.4 Servidor

Classe que representa o servidor onde o sistema será executado, sendo aqui criados os *sockets* para as transmissões com os clientes e também o local onde as conexões com os mesmos são aceites.

2.5 Servico

Esta é a classe onde são recebidos e executados os pedidos dos Clientes para o Servidor.

2.6 Cliente

Nesta classe permite aos Clientes do sistema comunicarem com o mesmo, ou seja, é esta que permite a um utilizador do sistema interagir com o mesmo, permitindo a execução de diversas atividades, tais como o *download/upload* de música entre outros.

2.7 ClienteSndRcv

É nesta classe onde são tratados os pedidos de *download/upload* dos Clientes, e ao ter uma classe específica para estes pedidos, torna possível a execução destes pedidos em *backgroud*.

2.8 ServidorSndRcv

É nesta classe onde são tratados os pedidos de *download/upload* pelo Servidor, e ao ter uma classe específica para estes pedidos, torna possível a execução destes pedidos em *backgroud*.

2.9 newSongUpdate

Esta classe trata de informar os Clientes ligados ao sistema da existência de novo conteúdo disponível para descarga.

3 Protocolos e Controlo de Concorrência

Neste capítulo serão discutidos a forma como são tratadas as diferentes funcionalidades pelo sistema.

3.1 Registar Utilizador

Para um Cliente se registar no Sistema, é necessário que o mesmo forneça um utilizador que não se encontre em uso e uma palavra-passe. Após completado o registo por parte do Cliente, o Servidor cria uma sub-pasta com o nome do Cliente recém criado na pasta de *Downloads* para ser possível manter um registo das músicas que o Cliente descarregou.

3.2 Autenticação no Sistema

Para um Cliente se autenticar no sistema é necessário que este forneça o seu nome de utilizador e a sua respetiva palavra-passe. Depois de fornecidos estes dados o Servidor verifica se esse Cliente pertence ao Sistema, de seguida verifica se a palavra-passe é a correta e por fim verifica que o mesmo não se encontra atualmente *online*. Caso todos estes procedimentos sejam cumpridos com sucesso, o Cliente é autenticado e passa a estar *online* no Sistema.

3.3 Upload de uma Música

Para um Cliente fazer um *upload* de uma música necessita de se encontrar autenticado no sistema, e de seguida necessita de forncecer os meta-dados da música que deseja carregar e o ficheiro da mesma. Depois de fornecidos estes dados o Sistema retorna o identificador único da música que foi carregada e guarda a mesma na pasta músicas.

3.4 Procura de uma Música

Outra das funcionalidades do Sistema além da partilha de músicas é a possibilidade dos utilizadores pesquisarem músicas fornecidas por outros utilizadores. Para tal, foram implementadas funcionalidades de procura de música tais como a possibilidade de listar todas as músicas existentes no Sistema, procurar e descarregar músicas usando o seu identificador único, pelo seu autor, entre outros. Para um utilizador ter acesso a estas funcionalidades necessita de estar autenticado no Sistema.

3.5 Download de uma Música

Se um utilizador pretender fazer um *download* de uma música, necessita apenas de estar autenticado e aceder posteriormente ao menu de procura de músicas. A partir daí, dependendo da opção de procura selecionada pelo mesmo, é realizado o *download* da música selecionada para uma pasta denominada *Downloads/Username/*. Em simultâneo a este processo de descarga, o sistema incrementa o número de vezes que essa música foi descarregada.

3.6 Encerramento de ligação com o Servidor

Quando um utilizador pretender terminar sessão, basta dirigir-se ao menu inicial e carregar na opção de *logout*, terminando assim as suas ligações com

os *sockets* de comunicação com o Servidor. Internamente é alterada também a variável que indica se um utilizador se encontra *online* para o valor 0.

3.7 Notificação de inserção de nova Música

Aquando o *upload* de uma nova música por parte de um qualquer utilizador, todos os utilizadores ligados, quer estejam autenticados ou não, ao Sistema recebem uma notificação a informar que uma nova música foi adicionada ao Sistema.

3.8 Guardar localmente informação do Sistema

Para permitir que os dados do Sistema não sejam apagados aquando o fecho do Servidor, é necessário guardar localmente a informação dos *HashMaps*. Assim sempre que o Servidor é iniciado, verifica se já existem dados, se sim carrega-os para o sistema o que permite que o Sistema não seja volátil. Durante a utilização do Sistema, os dados dessas *HashMaps* são armazenados em disco sempre que ocorre uma alteração, impedindo que seja perdida informação caso algo falhe no Sistema.

3.9 Controlo de Concorrência durante a criação/autenticação de um Utilizador

Para evitar que sejam criados dois utilizadores com o mesmo nome, ou que o mesmo utilizador esteja autenticado duas vezes no Sistema é necessário que durante estes processos sejam utilizados *Locks*. A forma como estes são utilizados é que aquando da criação/autenticação o sistema "bloqueia"de modo a garantir que apenas uma só *thread* entre na região crítica. Assim, torna-se impossível duas *threads* diferentes executarem a mesma operação, o que garante que as situações citadas no iníco do parágrafo nunca aconteçam.

3.10 Controlo de Concorrência relativamente ao *Upload/-Download* de Músicas

Durante as operações de *Download/Upload* é necessária a utilização de *Locks* para evitar que *threads* diferentes estejam na região crítica, os *HashMaps*, ao mesmo tempo. Assim, é possível evitar que exista, por exemplo, uma *thread* a aceder a informações de uma música no processo de *download* e outra *thread* a alterar essas mesmas informações ao mesmo tempo. Só assim se torna possível a utilização *multi-threaded* de uma aplicação como esta.

3.11 Controlo de Concorrência relativo ao número máximo de pessoas a fazer *Downloads* em simultâneo

De forma a garantir que não existam mais do que MAXDOWN pessoas a fazer *downloads* simultaneamente é necessário implementar *Conditions* que bloqueiam os Clientes que tentem fazer *download* após existirem MAXDOWN descargas a acontecer. Quando os Clientes terminarem o seu *download*, é enviado um *signal* para as *threads* em espera para estas começarem o seu processo de descarga.

4 Conclusão

Após a implementação desta aplicação, usando competências adquiridas nas aulas, o grupo considera ter feito um bom trabalho. Contudo, existem aspectos que podiam ser melhorados, como por exemplo tratar as exceções de forma correta em que estas por vezes não são devidamente tratadas. No entanto, e face às dificuldades encontradas durante este projeto, o grupo considera ter ultrapassado estas e melhorado os seus conhecimentos em programação concorrente.

Cumprindo todos os requisitos propostos no enunciado, o grupo considera o trabalho um sucesso, apesar de existirem aspetos a melhorar.

Apesar disto, o desenvolvimento do projeto permitiu solidificar o conhecimento adquirido durante a UC, permitindo uma melhor visão sobre as vantagens e perigos do uso de *threads*, bem como gerir regiões críticas.

Em suma, o trabalho corresponde aos desafios propostos no enunciado do projeto, apesar de possuir falhas que, num futuro próximo, serão implementadas como forma de solidificar, ainda mais, o conhecimento adquirido.