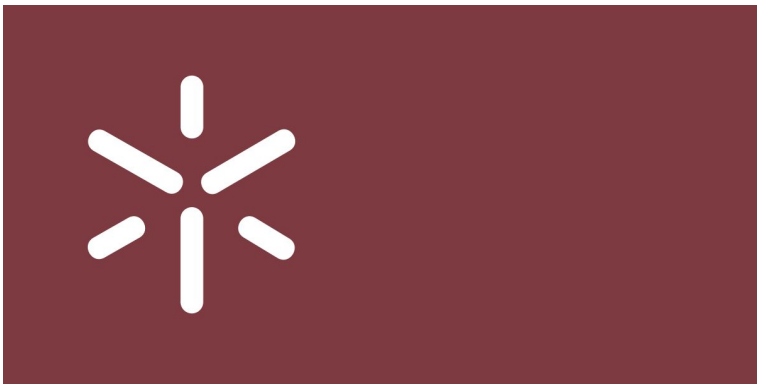

Universidade do Minho

Sistemas Distribuídos

Data: 05/01/2020

Discentes:

João Abreu	**a84802**
Hugo Matias	**a85370**
Tiago Magalhães	**a84485**
Duarte Oliveira	**a85517**

Índice

Introdução

O presente relatório tem por objetivo descrever o processo de construção de uma plataforma de música com a particularidade de suportar a troca de ficheiros.

Este trabalho em específico teve em mente a construção de um programa/aplicação onde músicos pudessem partilhar as suas músicas, e os fãs as pudessem descarregá-las para posteriormente as ouvirem .

Numa fase inicial serão apresentadas as classes e métodos iniciais de construção. De seguida serão apresentadas as funcionalidades básicas, o modo como foram feitas e pensadas. Posteriormente serão demonstradas as funcionalidades adicionais e a forma como foram construídas. Finda-se o relatório com uma breve conclusão que, após uma pequena exposição de todas as dificuldades relativas ao desenvolvimento das funcionalidades (que ao longo deste documento serão descritas), pretende mostrar um pouco do nossa perceção geral do trabalho .

Funcionalidades básicas

- *Autenticação e registo de utilizador, dado o seu nome e palavra-passe. Sempre que um utilizador desejar interagir com o serviço deverá estabelecer uma conexão e ser autenticado pelo servidor.*

Para resolver este requisito guardamos na classe SoundCloud toda a informação sobre o utilizadores, na forma de um Map<String, User> em que a string chave seria o email do utilizador, enquanto que a classe User continha toda a informação que representa um utilizador (email e password). De cada vez que queríamos registar um user seria a operação put e a autenticação seria uma comparação de palavra-passes.

- *Publicar um ficheiro de música, fornecendo o seu conteúdo e alguns meta-dados (título, intérprete, ano e um número variável de etiquetas) e recebendo um identificador único.*

Para resolver este requisito guardamos na classe SoundCloud toda a informação sobre as músicas, na forma de um Map<Integer, Musica> em que o Integer chave seria o id da música, enquanto que a classe Musica continha toda a informação que representa uma música (id, titulo, interprete, ano, lista de etiquetas e o número de vezes descarregadas). Para a atribuição do id à música usamos um contador também na classe SoundCloud que mantinha a informação de quantas músicas tinham sido criadas, incrementando sempre que se adicionava uma nova música.

Para fazermos a publicação do ficheiro música usamos os objectos: DataOutputStream, DataInputStream, FileInputStream e FileOutputStream. Os primeiros dois serviam para enviar o array de bytes que continha a informação dos ficheiros música através do socket e os dois últimos para ler do ficheiro origem e criar e escrever no ficheiro destino, respectivamente. Estas operações encontram-se na class FileOperations.

- *Efetuar uma procura de música, enviando uma etiqueta a procurar e recebendo de volta uma lista das músicas que lhe correspondem. Para cada música devem ser também recebidos os seguintes seus meta-dados:*

– identificador único;
– título, intérprete, ano e um número variável de etiquetas, fornecidos quando a música é carregada;
– número de vezes que a música foi descarregada.

Para listar todas as músicas que tinham uma dada etiqueta optamos por percorrer todo o Map que contém a informação das músicas e ver em cada objeto Musica se a sua variável de instância que tem os dados das suas etiquetas realmente contém a etiqueta passada como argumento. Se tiver adicionamos a uma lista resultado todo o objeto Musica, pois este já tem toda a informação necessária para satisfazer os requisitos.

- *Descarregar um ficheiro de música, fornecendo o seu identificador único*

O processo para descarregar um ficheiro foi igual ao de publicação mas no sentido contrário. Usamos as mesmas operações encontradas no FileOperations mas no sentido de Server -> Client em vez de Client -> Server.

Funcionalidades Adicionais

- *Limite Descargas:*

Para o limite de descargas desenvolvemos uma class semelhante à RWLock abordada nas aulas práticas com as devidas alterações. Em vez de podermos ter 1 writer ou n readers, usamos apenas n < MAXDOWN writers. De cada vez que um cliente é executa o download exectuamos o método writeLock() e writeUnlock() no fim, onde são implementadas as esperas e as notificações.

- *Notificação de novas músicas:*

Na notificação de novas músicas foi onde tivemos mais problemas, que serão discutidos mais à frente na secção Dificuldades.

A maneira que abordamos o problema foi guardar os PrintWriters de cada cliente numa classe à parte e quando chegar um upload de um qualquer cliente executaremos um método que enviava uma mensagem para todos os clientes através dos respectivos PrintWriters.

- *Tamanho dos ficheiros ilimitado:*

Para podermos ler qualquer tamanho de ficheiro temos de ler secções de bytes e ir enviando-as para o socket e reconstruir o ficheiro do outro lado. As operações encontradas no FileOperations usam um array de bytes de 0.5 Mb (valor escolhido) e é possível enviarmos ficheiros com mais de 0.5 Mb de tamanho.

Dificuldades

Foram sentidas algumas dificuldades de interpretação, ainda que após alguma deliberação as mesmas tivessem sido ultrapassadas.

Dentro do processo prático associado a este projeto, talvez a implementação de um servidor que suportasse vários clientes em simultâneo a executarem um determinado comando tenha-se revelado mais complicado do que esperávamos.

Numa fase mais tardia deste projeto, também houve uma necessidade especial de aplicarmos um esforço extra à criação da funcionalidade adicional de notificação de novas músicas, pois estas também não se provaram particularmente fáceis, numa abordagem inicial. Conseguimos implementar a notificação para o próprio cliente (cliente menos relevante recebê-la) mas para os outros estes ficariam à espera do próximo comando do terminal e só depois apareceria o print da notificação. No entanto, este outro cliente não podia enviar comandos porque isso deixaria um falha de sincronização de envio-recibo de respostas pois haveria uma resposta extra (a notificação). Por isso criamos uma opção do cliente enviar um comando dizendo que está à espera de uma notificação chamado “stuck”. No servidor este comando não faz absolutamente nada, apenas serve para sincronizar as mensagens. O cliente irá receber a notificação deste modo (deixa de estar preso na leitura do terminal) e depois pode continuar a escrever.

Conclusão

Um projeto prático desta dimensão, sendo bem estruturado e devidamente planeado pela equipa docente, é crucial para a formação dos alunos e revela-se de extrema importância para o futuro laboral dos mesmo. A necessidade de implementar um programa ligeiramente a par do que hoje em dia é elaborado leva a que os alunos saiam da sua zona de conforto e tenham que aplicar o lecionado de uma forma mais paralela àquela que nos foi apresentado nas aulas práticas e teóricas, o que no final se traduz numa evolução e formação a nível de sistemas distribuídos para cada um dos elementos participantes deste grupo muito superior àquela que tínhamos antes do início deste projeto. Deste modo, as dificuldades descritas, apesar de inesperadas, tiveram como consequência um aumento de experiência para cada um de nós, a par de uma melhor compreensão a nível de programação e computação.