Universidade do Minho

Mestrado Integrado em Engenharia Informática

Sistemas Distribuídos

SoundCloud

Grupo 5

Pedro Gomes A84220, José Santos A84288

Conteúdo

1 Introdução

Este relatório documenta todo o trabalho desenvolvido para resolver um problema proposto no âmbito da Unidade Curricular de Sistemas Distribuídos.

O projeto consiste no desenvolvimento de uma solução que permita desenvolver uma plataforma de cloud , tendo como suporte uma aplicação implementada em Java, que inclui um Servidor multi-thread que interage com vários clientes.

Com este projeto pretende-se que sejam aplicados conceitos de programação com sockets, ou seja, comunicação entre os clientes e o servidor e a programação concorrente, no sentido em que podem estar a ocorrer vários processos, sejam estes downloads, uploads ou mesmo de consulta, ao mesmo tempo que o servidor continua a responder devidamente. Será ainda definido que a máquina cliente sabe o IP e a Porta do servidor e que este possa servir concorrentemente múltiplos pares de clientes.

2 Desenvolvimento

2.1 Contextualização

O objetivo do trabalho prático é criar uma plataforma de cloud que permita as seguintes operações:

- Autenticação por parte do cliente que pretenda usar a plataforma
- Dar upload das musicas
- Dar download das musicas
- Mostrar uma lista com as músicas
- Mostrar uma lista com as músicas referentes a uma dada etiqueta

2.2 Estrutura do programa

A comunicação entre o servidor e o cliente funciona por mensagens. Mensagens estas que são classes implementadas que pelo tipo de mensagem que é enviado pelo cliente o servidor sabe como responder. Temos vários tipos de mensagens:

2.2.1 Mensagens entre cliente e Servidor

- ResponseMessage

  ResponseMessage lida com mensagens básicas , sem parsing, só passa o id de autenticação , o tipo da mensagem e uma resposta. Exemplo: "[3,1]logout-¿ 3 = tipo de mensagem , o 1 é o id da sessão e logout indica que o utilizador 1 quer dar logout neste caso.

- MessageAuthentication

  MessageAuthentication(lida com mensagens de autenticação,faz logo o parsing do user e da password) , tem métodos para ir buscar logo o username e a password. Exemplo: "[1]admin;teste"em que 1 é o tipo da mensagem neste caso significa Login , admin é o username e teste é a password separado por ;.

- Notification

  Notification(indica ao servidor que o cliente com o id que a notificação vier está à espera de receber notificações da parte do servidor). MP3Download(indica ao servidor um pedido de download de mp3) , contém o id da mensagem, o id do utilizador que fez o pedido e o id da musica que o cliente quer fazer download.

- MP3Upload

  MP3Upload(indica ao servidor um pedido de upload de mp3), contém todas as informações da musica , artista,titulo,ano de criação e um numero variavel de etiquetas, o id do utilizador que tá a fazer o pedido. MusicListMessage(indica ao servidor um pedido de lista de musicas disponiveis), envia nesta mensagem uma lista de musicas em string que depois o cliente faz o parse e apresenta no ecrã.

2.2.2 Cliente

O programa cliente, cada vez que precisa de alguma coisa do servidor abre uma nova conexão. Caso se trate de um download ou de um upload inicia uma nova thread e começa a tratar a situação. No inicio é inicializada também uma thread que verifica se há novas notificações vindas do servidor, caso haja , chama o NotificationListener e apresenta no ecrã a notifcação.

2.2.3 Servidor

O servidor, baseado no modelo Thread-per-request, começa por aceitar a ligação de um cliente, recebendo uma mensagem(em forma de string) , identificando o tipo da mesma. Tendo em conta o tipo passa ao ServerWorker ou não. Caso seja uma notificação , ele insere o cliente numa lista de clientes disponíveis para receber notificações. No caso de ser um download de uma musica , ele verifica quantos downloads decorrem naquele momento, caso haja mais de 10, este pedido é adicionado a uma PriorityQueue em que tem prioridade o cliente com menor número de downloads pendentes. No caso de ser de outro tipo qualquer, é inicializado um ServerWorker e é submetido para a pool de threads, podendo ser executado de imediato ou não tendo em conta a prioridade. A pool de threads (10 threads), é composta por uma queue com prioridade, em que cada thread corre a class ServerWorker que basicamente trabalha com o pedido de mensagem que recebe. Cada thread tem associada a si uma prioridade que tem haver com o tipo de mensagem que recebe. A prioridade dá-se da seguinte forma : Login,Register,ResponseMessage,MusicList,MP3Upload,MP3Download. A class ServerWorkerFutureTask que extende a class FutureTask é que permite implementar a prioridade fazendo com que as threads antes de serem executadas , sejam comparadas entre sí, coisa que só é possível pois a class FutureTask permite dizer que são instâncias comparáveis e portanto conseguimos comparar pela prioridade da task. Temos uma class App que trata de tudo o que seja login, logout, registar, upload,download , aplicando os respectivos locks necessários etc. Guardamos as músicas numa class MusicDatabase que extende o HashMap e que funciona basicamente como a nossa base de dados da aplicação.

3 Conclusão

Este relatório serviu o propósito de redigir os passos executados ao longo do desenvolvimento da solução apresentada, com base nos conhecimentos adquiridos na UC de Sistemas Distribuídos, para um exemplo de servidor para um sistema de Cloud. A realização deste trabalho foi feito sem grandes dificuldades , no entanto , no decorrer do trabalho descobríamos sempre melhores formas de fazer o que estávamos a fazer. Com a experiência de desenvolver este trabalho aprendemos a analisar a melhor maneira de implementar a comunicação entre o servidor e o cliente tendo em conta o que pretendemos. Para a interação entre clientes foi utilizado as primitivas de programação concorrente para que fosse possível aos inúmeros clientes, se conectarem e utilizarem a plataforma. Utilizou-se a programação com sockets para a troca de mensagens, e para o upload/download de músicas entre o Cliente e o Servidor. Este projeto foi importante porque permitiu ao grupo consolidar os conhecimentos sobre a funcionalidade e uso de sockets bem como trabalhar com tarefas multi-thread. O grupo considera que o objetivo deste trabalho prático foi concluído com sucesso uma vez que se cumpriram com todas as tarefas propostas.