DropMusic: arquivo de música com partilha de ficheiros

Sistemas Distribuídos 2018/19 — Meta 1 — 26 de outubro de 2018 (21:59)

Resumo

Este projeto tem como objetivo criar um sistema de gestão e partilha de músicas com funcionalidades semelhantes aos arquivos AllMusic.com e IMDb.com, acrescentando a possibilidade de partilhar ficheiros entre utilizadores tal como o serviço Dropbox. O sistema deverá ter todas as informações relevantes sobre as músicas, tais como os autores, álbuns, grupos musicais, compositores, letras, concertos, e outros que considere importantes. Ao consultar um álbum, um utilizador obtém a listagem de músicas com as respetivas informações, críticas escritas por outros utilizadores, bem como informações gerais sobre os géneros musicais e a data de lançamento. Apenas os editores podem acrescentar e alterar informações sobre álbuns e músicas, sendo que os restantes utilizadores podem apenas consultar essas informações. É possível a cada utilizador transferir ficheiros musicais (MP3, FLAC, etc.) que ficarão associados à sua própria conta mas que poderão ser partilhados com outros utilizadores.

1 Objetivos do projecto

No final do projeto cada estudante deverá ter:

- programado um sistema de gestão de música com arquitetura cliente-servidor;
- usado sockets TCP, UDP e Multicast para comunicação entre clientes e servidores;
- seguido um modelo multithread para desenhar os servidores;
- criado uma camada de acesso aos dados usando Java RMI;
- garantido a disponibilidade da aplicação através de redundância com failover.

2 Visão geral

Um guia de música tal como o arquivo AllMusic.com permite aos utilizadores consultarem informações sobre músicas, autores e álbuns. Este projeto tem como objetivo criar um sistema de gestão de músicas que dê também a possibilidade a um utilizador de partilhar ficheiros musicais com outros utilizadores individuais.

O sistema a desenvolver deverá ter toda a informação relevante sobre as músicas. Deve incluir informações tais como: músicos, grupos musicais, história e períodos do

grupo, músicas, editoras, compositores (podem ser músicos), quem compôs cada música, letras, concertos e outros que considere relevantes.

Ao consultar um álbum, obtém-se a listagem de músicas com as informações mais relevantes, críticas escritas por outros utilizadores, bem como informações gerais sobre os géneros musicais e a data de lançamento. Um utilizador deve registar-se para usar o sistema. Apenas utilizadores com privilégios de editor podem acrescentar e alterar informações no sistema.

Qualquer utilizador (editor ou não) pode escrever uma crítica a um álbum, que consiste numa pontuação e numa justificação textual. Cada utilizador pode também criar playlists compostas por músicas que selecione. As playlists podem ser privadas ou públicas (ficando disponíveis para todos os outros utilizadores).

Trabalhos apresentados por 3 alunos têm, adicionalmente, os requisitos de criação de grupos de amigos e pedido de adição a grupos de amigos já existentes. Os grupos têm owners, que confirmam ou recusam a pertença ao grupo após pedido. O owner inicial é o utilizador que criou o grupo, sendo que ele pode adicionar mais owners.

O objetivo dos grupos é permitir partilha limitada ao grupo. Quando nova informação é adicionada por um utilizador, este pode indicar se a partilha é publica ou restringida a um ou mais dos grupos a que pertence. Isto significa também que existe um grupo de nome Público por defeito, o qual partilha com todos os utilizadores registados. Existe um owner inicial do grupo Público, que pode adicionar outros owners.

Quer no grupo Público, quer em qualquer outro grupo, existe também o papel de editor (um owner é editor por defeito, mas um editor pode não ser owner). Apenas os editores podem acrescentar e alterar informações sobre álbuns e músicas, sendo que os restantes utilizadores podem apenas consultar essas informações. Fica sempre registado quem colocou informação.

É possível a cada utilizador transferir um ficheiro musical (AAC, MP3, etc.) que ficará associado à sua própria conta mas que poderá ser partilhado com outros utilizadores individuais, público ou grupos específicos (lista de partilha), conforme indicado pelo utilizador.

Sempre que adiciona informação/partilhas, um utilizador especifica também para que grupos partilha. Para permitir que o utilizador não tenha de indicar sempre os grupos a quem partilha, um utilizador tem também uma lista de grupos a quem partilha por omissão. Inicialmente essa lista tem o grupo Público, mas pode ser alterada para grupos que deseje.

3 Funcionalidades a desenvolver

O sistema tem *utilizadores* que acedem às funcionalidades com a aplicação DropMusic a desenvolver. Alguns dos utilizadores são também *editores* que podem acrescentar e modificar informações. Deverá ser possível realizar as seguintes operações:

1. **Registar pessoas.** Os utilizadores devem poder registar-se, sendo que o acesso à aplicação deve estar protegido com username e password. Deverá guardar toda a informação pessoal que considere necessária bem como uma password (código de acesso). Para simplificar, considere que o primeiro utilizador a registar-se é o administrador e tem automaticamente privilégios de editor em todo o sistema.

2. **Gerir artistas, álbuns e músicas.** A informação está organizada por artistas, álbuns e músicas, sendo que apenas os editores podem inserir, alterar e remover dados relativos a estas entidades. Os restantes utilizadores podem apenas ler a informação existente.

3. **Pesquisar músicas.** Um utilizador deve poder pesquisar por álbuns e por artistas introduzindo dados parciais tais como o nome do artista, o género musical ou o título do álbum. O resultado da pesquisa é uma lista de itens na qual se pode escolher um item para obter mais detalhes.

4. **Consultar detalhes sobre álbum e sobre artista.** Deverá ser possível consultar todos os detalhes de um álbum, tais como a descrição, a lista de músicas, críticas escritas por outros utilizadores, a pontuação média atribuída pelos utilizadores, etc. A lista de músicas deverá mostrar informações tais como título, compositor, duração, etc. Deve ser também possível aceder aos detalhes sobre cada artista.

5. **Escrever crítica a um álbum.** Qualquer utilizador (editor ou não) deverá poder escrever uma crítica a um álbum, que consiste numa pontuação e numa justificação textual que não exceda 300 caracteres. Todas as críticas a um dado álbum aparecem listadas ao consultar os detalhes desse álbum, sendo igualmente apresentada a pontuação média.

6. **Dar privilégios de editor a um utilizador.** Aos utilizadores que tenham privilégios de *editor* é possível conceder esses mesmos privilégios a outros utilizadores. Para tal, escolhem o utilizador que pretendem promover a editor e esse utilizador passa a poder alterar informações no sistema.

7. **Notificação imediata de privilégios de editor.** Quando um utilizador é promovido a editor deve receber imediatamente (em tempo real) essa informação se estiver ligado à aplicação, sem precisar de realizar nenhuma operação.

8. **Notificação imediata de re-edição de descrição textual.** A descrição textual incluída nos detalhes de um álbum (ou de um artista) pode ser alterada por um editor, tal como especificado anteriormente. Sempre que um editor altere uma descrição textual todos os editores anteriores são notificados de imediato se estiverem ligados à aplicação. Por exemplo, um editor que altere a descrição de um álbum será notificado sempre que essa mesma descrição seja novamente alterada por outro editor qualquer.

9. **Entrega posterior de notificações a utilizadores desligados.** Qualquer utilizador que devesse ter recebido uma notificação imediata, mas não se encontrasse ligado à aplicação (offline), recebe a notificação assim que se ligar a próxima vez.

10. **Transferência de músicas para o servidor.** É possível a cada utilizador transferir um ficheiro musical (upload) que ficará associado a uma música específica e que ficará inicialmente restrito à conta do próprio utilizador mas que poderá depois ser partilhado com outros utilizadores.

11. **Partilha de ficheiros musicais.** Um utilizador que tenha transferido ficheiros musicais para o servidor pode escolher utilizadores individuais com os quais pretenda partilhar esses mesmos ficheiros.

12. **Transferência de músicas do servidor para os utilizadores.** Ao partilhar um ficheiro musical com um dado utilizador, esse utilizador pode transferir o ficheiro do servidor para a sua própria máquina (download).

13. **Criar grupo de amigos.** Trabalhos apresentados por 3 alunos têm o requisito de criação e gestão de grupos de amigos. Os grupos têm owners, que confirmam ou recusam a pertença ao grupo após pedido. O owner inicial é o utilizador que criou o grupo, sendo que ele pode adicionar mais owners. **Grupos de 3 estudantes.**

14. **O owner de cada grupo faz a gestão dos membros.** É possível a um owner de um grupo adicionar e remover utilizadores do grupo, confirmando e recusando a pertença ao grupo após pedido. **Grupos de 3 estudantes.**

15. **A partilha de informação é limitada aos membros de cada grupo.** O objetivo dos grupos é permitir partilha limitada ao grupo. Quando nova informação é adicionada por um utilizador, este pode indicar se a partilha é publica ou restringida a um ou mais dos grupos a que pertence. **Grupos de 3 estudantes.**

16. **Dentro de cada grupo há editores que podem não ser owners.** Em qualquer grupo existe o papel de editor (um owner é editor por defeito, mas um editor pode não ser owner). Apenas os editores podem acrescentar e alterar informações sobre álbuns e músicas, sendo que os restantes utilizadores podem apenas consultar essas informações. **Grupos de 3 estudantes.**

4 Arquitetura

A Figura 1 mostra a arquitetura global do projeto. A aplicação DropMusic consiste em três programas: servidor RMI, servidor Multicast e cliente RMI. Cada grupo deverá programar os *servidores RMI*, que são idênticos embora um seja inicialmente primário e outro secundário. Cada grupo deverá igualmente programar os *servidores Multicast*, que são idênticos exceto possíveis questões de configuração. Cada grupo deverá programar um *cliente RMI*, à parte, que se liga aos servidores RMI e permite aos utilizadores acederem ao DropMusic.

Toda a informação está armazenada nos servidores Multicast (em ficheiro de texto, ficheiro de objetos, base de dados, O/R mapping, etc.). Estes servidores são réplicas que contêm a mesma informação, servindo para manter o sistema a funcionar desde que uma réplica esteja ativa (podendo as outras avariar ou serem desligadas). Os servidores recebem pedidos e enviam respostas através de comunicação Multicast (one-to-many) usando um protocolo a construir pelos estudantes.

Os servidores RMI usam a rede multicast para enviar pedidos aos servidores Multicast. O sistema deverá aceitar qualquer número de clientes RMI, que consistem numa

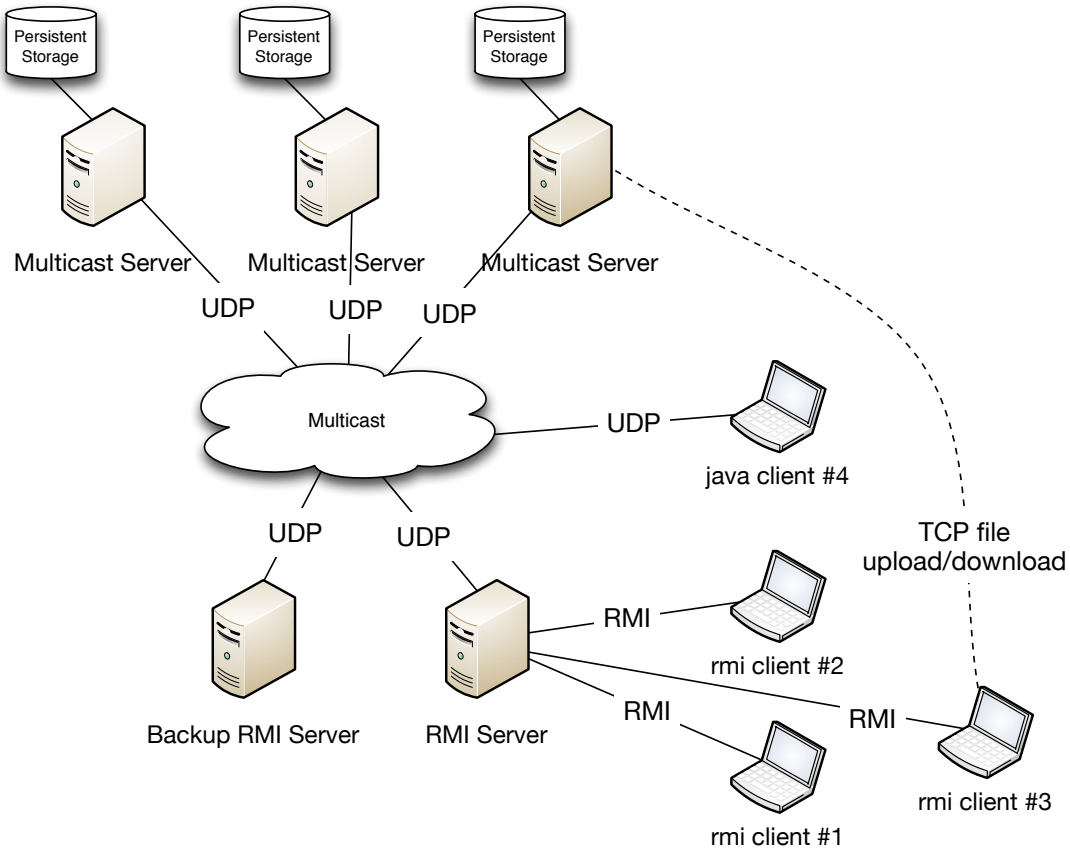

Fig. 1: Arquitetura do sistema.

pequena aplicação que permite aceder ao DropMusic. Pretende-se com isto que os clientes RMI sejam muito simples, com uma interface elementar, ainda que na prática se aceite outras soluções. Assim, deverão ser criados os seguintes programas:

- Multicast Server – É o servidor central (replicado) que armazena todos os dados da aplicação, suportando por essa razão todas as operações necessárias através de pedidos recebidos em datagramas multicast.

- RMI Server – Existem dois servidores RMI que não armazenam dados localmente e que disponibilizam, através da sua interface remota, um conjunto de métodos acessíveis a aplicações com Java RMI. Quando um método é invocado, o servidor RMI traduz esse pedido num datagrama UDP enviado por multicast para os servidores Multicast, aguardando depois pela resposta enviada igualmente por multicast.

- RMI Client – É o cliente RMI usado pelos utilizadores para aceder às funcionalidades do DropMusic. Pretende-se que este cliente tenha uma interface bastante simples e que se limite a invocar os métodos remotos no servidor RMI, lidando corretamente com avarias do servidor primário mudando para o servidor secundário.

Os clientes RMI têm uma única funcionalidade para a qual usam comunicação TCP diretamente com um servidor Multicast: transferências de ficheiros de músicas. Quando um utilizador faz upload ou download de um ficheiro, o cliente RMI pede um endereço IP de um único servidor Multicast e a transferência é realizada diretamente entre cliente RMI e servidor Multicast.

5 Protocolo UDP Multicast

De forma a permitir a implementação de clientes DropMusic para qualquer plataforma (seja Windows, Linux, macOS, Android, iOS), o protocolo deverá ser especificado em detalhe. Segue-se uma recomendação da base de um protocolo a ser completado pelos estudantes. A estrutura principal deste protocolo é a mensagem que consiste num conjunto não ordenado de pares chave-valor, semelhante a um HashMap em Java ou a um dicionário em Python, finalizados com uma mudança de linha. Um exemplo seria:

```
chave1 | valor1; chave2 | valor dois
```

Como tal, não é permitido ter o carácter "pipe" (|) nem o carácter ponto e vírgula (;) nem o carácter mudar de linha (\n) no nome da chave nem no valor.

Todas as mensagens têm um campo obrigatório chamado "type". Este valor permite distinguir o tipo de operação que o cliente pretende fazer no servidor, ou o tipo de resposta que o servidor está a dar. Um exemplo mais realista é portanto:

```
type | login; username | tintin; password | unicorn
```

E a resposta respetiva, sendo que o campo msg é opcional, mas aceite.

```
type | status; logged | on; msg | Welcome to DropMusic
```

Finalmente, para representar listas de elementos é usado o tamanho e o contador de elementos. O tamanho é descrito num campo com o sufixo x_count, onde x é o campo com a lista, e cada campo do elemento tem o formato x_i_campo, onde i é o índice da lista, a começar em 0. Um exemplo encontra-se de seguida:

```
type | music_list; item_count | 2; item_0_name | Gotta Jazz;
  item_1_name | Boogie Woogie
```

Este protocolo deverá ser completado e especificado pelos alunos no decorrer do projeto. Para facilitar o desenvolvimento, é fornecido juntamente com o enunciado um cliente de teste em Java que permitirá comunicar diretamente com os servidores Multicast (e que está identificado por "java client #4" no diagrama da arquitetura).

Sendo os servidores Multicast réplicas, cada pedido é executado por todos em paralelo e a informação fica assim armazenada de forma redundante. Por questões de desempenho, quando há uma resposta para ser enviada não deverão responder todos os servidores mas apenas um deles.

6 Requisitos não-funcionais

A aplicação deverá lidar corretamente com quaisquer exceções que estejam previstas. Por forma a garantir que o DropMusic está sempre disponível para os utilizadores, de-

verá usar uma solução de failover para garantir que a aplicação continua a funcionar ainda que um servidor qualquer possa avariar.

6.1 Tratamento de exceções

Como o hardware pode falhar, é necessário que os utilizadores não notem nenhuma falha de serviço. Como tal, no caso do servidor RMI falhar, é preciso garantir que as atualizações de álbuns e de músicas não se percam nem apareçam duplicadas. No caso das avarias temporárias (inferiores a 30 segundos) os clientes não se devem aperceber desta falha.

Também do lado dos clientes é possível que a ligação se perca a meio. É necessário garantir que nenhuma falha do lado do cliente deixe nenhuma operação a meio. Deve bastar reiniciá-los para que se possa depois continuar a usar o DropMusic.

Os servidores Multicast também podem avariar e ser reiniciados. Quando isso sucede, não se pretende que a informação perdida seja recuperada. Isto é, não se pretende que os servidores Multicast comuniquem entre si para se manterem perfeitamente sincronizados. Se tal for concretizado vale pontos extra no projeto.

6.2 Failover

De modo a que quando o servidor RMI falhar, o servidor secundário o substitua, é necessário ter alguns cuidados. Em primeiro lugar o servidor secundário deve periodicamente testar o servidor primário através de uma qualquer chamada RMI. Se algumas destas chamadas resultarem em erros (por exemplo cinco chamadas seguidas) o servidor secundário assume que o primário avariou e liga-se para o substituir.

Os clientes RMI também devem detetar quando existe uma falha permanente do servidor RMI e devem tentar ligar-se ao secundário. Dado o uso de persistência nos servidores Multicast, os dados visíveis pelos clientes devem ser exatamente os mesmos. Do lado dos clientes, todo este processo deve ser transparente. Para eles esta falha nunca deverá ter qualquer efeito visível.

Finalmente, se o servidor RMI original recuperar, deverá tomar o papel de secundário e não de primário.

6.3 Relatório

Devem reservar tempo para a escrita do relatório no final do projeto, tendo em conta os passos anteriores. Devem escrever o relatório de modo a que um novo colega que se junte ao grupo possa perceber a solução criada, as decisões técnicas e possa adicionar novos componentes ou modificar os que existem. **O relatório pode ser inteiramente escrito em Javadoc no código-fonte apresentado pelos estudantes.** Deve incluir:

- Arquitetura de software detalhadamente descrita. Deverá ser focada a estrutura de threads e sockets usadas, bem como a organização do código.
- Detalhes sobre o funcionamento do servidor Multicast. Deve especificar detalhadamente o protocolo usado para comunicação multicast.

- Detalhes sobre o funcionamento do servidor RMI. Deverá explicar detalhadamente o funcionamento dos métodos remotos disponibilizados e eventuais callbacks usados, bem como a solução usada para failover.
- Distribuição de tarefas pelos elementos do grupo.
- Descrição dos testes realizados (tabela com descrição e pass/fail de cada teste).

6.4 Distribuição de tarefas

De modo a que a avaliação seja justa num trabalho de grupo, é fundamental uma divisão de trabalho justa. Dado que a nota resultante da defesa será individual, são propostas as duas possíveis divisões de trabalho:

- Elemento 1 será responsável pelo Multicast Server e o elemento 2 pelo RMI Server e pelo RMI Client. Esta divisão assume que o protocolo multicast é especificado inicialmente e que as alterações serão daí em diante mínimas. Caso exista um terceiro elemento, este ficaria responsável pelo upload/download de ficheiros e pelas notificações instantâneas (isto é, reduz o trabalho aos outros dois elementos).
- Cada um dos elementos ficará com igual número de funcionalidades a implementar, trabalhando um pouco em cada um dos programas a desenvolver.

Finalmente, poderão ser aceites outras distribuições, desde que previamente acordadas com os docentes.

7 Planos futuros para o projeto

Na segunda meta do projeto (dezembro) irão expandir a presente solução, adicionando uma interface Web usando HTML/JSP/Struts2/Spring e irão integrar a aplicação com uma API REST de um serviço externo. Nessa fase, o servidor Web irá usar a API do servidor RMI aqui criado.

8 Entrega do projeto

O projeto deverá ser entregue num arquivo ZIP. Esse arquivo deverá conter um ficheiro README.TXT com toda a informação necessária para instalar e executar o projeto sem a presença dos alunos. Projetos sem informações suficientes, que não compilem ou não executem corretamente **não serão avaliados**.

Dentro do ficheiro ZIP deverá também estar um Javadoc/PDF/HTML com o relatório. O relatório deve seguir a estrutura fornecida, dado que a avaliação irá incidir sobre cada um dos pontos. Também no ficheiro ZIP deverão existir três ficheiros JAR: dataserver.jar (servidor Multicast), server.jar (servidor RMI), e client.jar (cliente RMI).

Finalmente, o ficheiro ZIP deverá ter também **uma pasta com o código fonte completo do projeto**. A ausência deste elemento levará à anulação do projeto.

O ficheiro ZIP com o projeto deverá ser entregue na plataforma Inforestudante até ao dia **26 de outubro de 2018 (21:59)**, via `https://inforestudante.uc.pt`