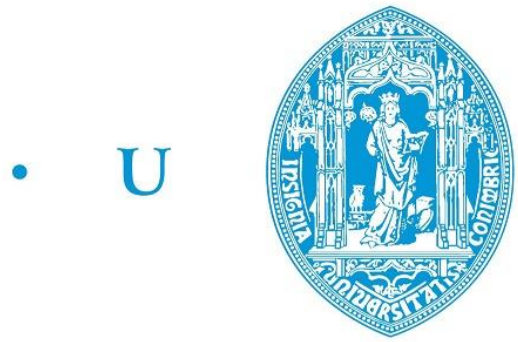


FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

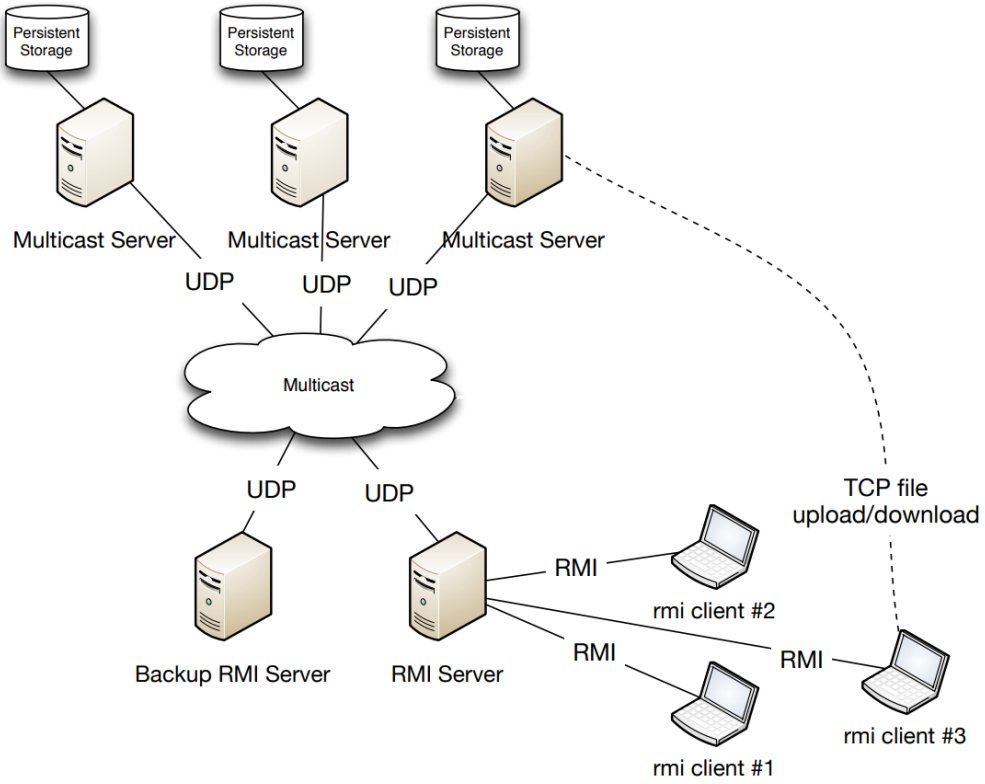
Sistemas Distribuídos

DropMusic - Projeto 2018/2019

Nome: Damião de Sousa Santos | Nº: 2015241046 |joker.dss@gmail.com

Nome: Tomás Faria Martins |Nº: 2016234128 |tomasfmartins@gmail.com

1. Arquitetura de software



- **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. Para o tratamento da funcionalidade upload/download o servidor multicast envia o IP ao cliente através de uma mensagem. Após isto o servidor cria uma Thread que fica á espera de uma ligação TCP.
- **RMI Server** – Existem dois servidores RMI que não armazenam dados localmente e 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. Quando o servidor RMI principal se desliga, o servidor backup RMI inicia no lugar dele e torna-se servidor principal. O servidor principal que se tinha desligado, quando se volta a ligar, torna-se servidor backup.
- **RMI Client** – É o cliente RMI usado pelos utilizadores para aceder às funcionalidades do DropMusic. Este cliente tem uma interface bastante simples e limita-se a invocar os métodos remotos no servidor RMI, lidando com as possíveis avarias do servidor primário mudando para o servidor secundário. Após um utilizador ter feito login, é inicializada uma Thread para estar à escuta de possíveis notificações para o utilizador.

2. Funcionamento do servidor Multicast (protocolo UDP)

Na pasta do projeto existe uma documentação em JavaDoc do servidor multicast.

No nosso projeto o cliente RMI comunica com os servidores Multicast com o auxílio de um servidor RMI através de mensagens. Sempre que um cliente realiza uma operação é enviada uma mensagem para os servidores multicast onde esta é convertida num HashMap e depois é executa consoante as funcionalidades que trás, após os servidores executarem a informação é sempre devolvida uma mensagem ao cliente para o informar se as tarefas foram ou não realizadas com sucesso.

Todas as respostas do lado do servidor contêm um campo "username" com um determinado valor. Esse valor determina para quem é que a informação vai ser enviada.

Operação: Registo

Do lado do cliente:

→ type|login;username|(respetivo username);password|(respetiva password)

Do lado do servidor:

→ type|confirmacao;resposta|(sim/nao);username|(...);msg|(...)

→ No campo "resposta" o seu valor é "sim/nao" consoante se a autenticação tenha sido efetuada ou não com sucesso, respetivamente. A mensagem em caso de sucesso é "Registo efetuado com sucesso" e em caso de erro é "Erro! O username inserido já existe."

Operação: Login

Do lado do cliente:

→ type|login;username|(respetivo username);password|(respetiva password)

Do lado do servidor:

→ type|status;logged|(on/off);msg|(mensagem a enviar ao cliente) ;username|(...);privilegio|(editor/leitor)

→ No campo "logged" o seu valor é on/off consoante as credenciais que o servidor receba. Sendo que se as credenciais estejam certas é emitido o valor "on" e o login foi efetuado com sucesso, e caso as credenciais estejam incorretas o valor emitido é "off"

→ No campo "privilégio" o valor emitido varia consoante os privilégios do utilizador em causa. Por default quando o utilizador se regista o privilégio que lhe é atribuído é de "leitor", podendo mais tarde ser promovido por um utilizador com privilégio de "editor".

→ No campo "msg" são emitidas mensagens conforme as tarefas que foram executadas, isto é, caso a autenticação seja realizada com sucesso o cliente recebe a mensagem "Bem-vindo!" e possíveis notificações; caso o utilizador não exista a mensagem enviada é "O utilizador não existe!"; e caso a password esteja incorreta a mensagem é "Password incorreta!".

Operação: Inserir Informação

Do lado do cliente:

→type|gerir;operacao|inserir;categoria|musica;nome|(...);artista|(...);album|(...);duracao|(...)

Do lado do servidor:

→ type|resposta;username|(...);msg|Informação inserida com sucesso

Operação: Alterar Informação

Do lado do cliente:

→type|gerir;operacao|apresentar;categoria|(musica/album/artista);username|(...) , quando o servidor recebe está mensagem a sua resposta vai ser uma lista de itens da categoria que foi selecionada pelo utilizador.

Do lado do servidor:

→ type|lista;length|x;items|nome1/artista1/nome2/artista2/...;username|(...), no caso em que a categoria seja "musica" ou "album";

→ type|lista;length|x;items|nome1/nome2/nome3/...;username|(...), no caso em que a categoria seja "artista".

Após resposta do servidor, o cliente responde:

→type|gerir;operacao|alterar;categoria|(musica/album/artista);index|x;campo|(...);info|(...);username|(...) , nesta situação no campo "campo" é inserida o detalhe que o utilizador deseja alterar (exemplo, nome do álbum, musicas do álbum, etc...) e no campo "info" é inserida a nova informação.

Após nova resposta do cliente, o servidor responde:

→ type|resposta;username|(...);msg|"Informação alterada com sucesso"

Operação: Remover Informação

Do lado do cliente:

→type|gerir;operacao|apresentar;categoria|(musica/album/artista);username|(...) , quando o servidor recebe está mensagem a sua resposta vai ser uma lista de itens da categoria que foi selecionada pelo utilizador.

Do lado do servidor:

→ type|lista;length|x;items|nome1/artista1/nome2/artista2/...;username|(...), no caso em que a categoria seja "musica" ou "album";

→ type|lista;length|x;items|nome1/nome2/nome3/...;username|(...), no caso em que a categoria seja "artista".

Após resposta do servidor, o cliente responde:

→type|gerir;operacao|remover;categoria|(musica/album/artista);index|x;username|(...)

Após nova resposta do cliente, o servidor responde:

→ type|resposta;username|(...);msg|"Informação removida com sucesso"

Operação: Pesquisar Informação

Do lado do cliente:

→ type|pesquisa;categoria|(...);nome|(...);username|(...)

Do lado do servidor:

→ type|lista;length|x;items|.../.../...;username|(...)

Após resposta do servidor, o cliente responde:

→ type|consulta;categoria|...;nome|...;artista|...;username|(...)

Após nova resposta do cliente, o servidor responde:

→ type|info;categoria|musica;detalhes|nome/artista/album/duracao

→type|info;categoria|album;detalhes|nome/artista/n_musicas/musicas/criticas/nota

→ type|info;categoria|artista;detalhes|nome/albuns

Após nova resposta do servidor (caso este deseje fazer uma crítica a um album), o cliente responde:

→type|critica;categoria|...;nome|...;artista|...;msg|"critica_do_utilizador";nota|...;username|...

Nova resposta do servidor:

→ type|resposta;username|(...);msg|"Critica adicionada com sucesso"

Operação: Alterar privilégios

Do lado do cliente:

→ type|utilizadores;username|(...)

Do lado do servidor:

→ type|lista_utili;length|x;items|../../..;username|...

Após resposta do servidor, o cliente responde:

→ type|promover;utilizador|....;username|...

Após nova resposta do cliente, o servidor responde:

→ type|notificacao;username|...;username2|...;msg|Foi promovido a editor!.

→ Caso o utilizador esteja online receba a notificação, caso contrário só recebe a notificação quando fizer login. Após receber a notificação o utilizador em causa responde :

→ type|noticonfirma;username|(...)

Operação: Download

Do lado do cliente:

→ type|pedirIP;acao|download;index|x;username|....

Do lado do servidor:

→ type|confIP;IP|...;username|...;

Operação: Upload

Do lado do cliente:

→ type|pedirIP;acao|upload;index|x;username|....

Do lado do servidor:

→ type|confIP;IP|...;username|...;

Operação: Partilha de PlayList

Do lado do cliente:

→ type|playlist;username|...

Do lado do servidor:

→type|listplay;length|x;items|musica/artista/musica2/artista2...;username|...;

Após resposta do servidor, o cliente responde:

→ type|autorizacao;musica|...;artista|...;utilizador|...;username|...

Após nova resposta do cliente, o servidor responde:

→ type|auto_res;username|...;msg|"Erro!Utilizador não existe" ou "Operacação efetuada com sucesso!"

3. Funcionamento do servidor RMI

Na pasta do projeto existe uma documentação em JavaDoc do servidor RMI e do cliente RMI. Para solucionar o problema de Failover, o servidor backup vai estar constantemente a testar o registo ao qual o servidor principal está ligado. Quando o servidor principal falha, o servidor backup vai tomar a posição dele. Caso o servidor principal volte, este toma a posição de servidor backup.

4. Distribuição de tarefas

Como sugerido pelo professor, decidimos que um elemento fosse responsável pelo Multicast Server e o outro pelo RMI Server e RMI Cliente tendo o protocolo multicast sido construído pelos dois inicialmente. Sendo assim o aluno Tomás Martins ficou responsável pelo Multicast Server e o aluno Damião Santos pelo RMI Server e RMI Client.

5. Descrição dos testes realizados

Teste Realizado	Resultado:
A) Registar novo utilizador	Sucesso
B) Login protegido com password	Sucesso

C) Introduzir artistas, álbuns e músicas	Sucesso
D) Pesquisar álbuns por artistas e por título de álbum	Sucesso
E) Consultar detalhes de um álbum (incluindo músicas e críticas)	Sucesso
F) Editar detalhes de um álbum (incluindo músicas)	Sucesso
G) Escrever críticas sobre um álbum (incluindo pontuação)	Sucesso
H) Consultar detalhes de artista (e.g., álbuns)	Sucesso
I) Notificação imediata de privilégios de editor (online users)	Sucesso
J) Notificação imediata de re-edição de descrição de álbum (online users)	Sucesso
K) Entrega posterior de notificações (offline users)	Sucesso
L) Upload de ficheiro para associar a uma música existente	Sucesso
M) Partilhar um ficheiro musical e permitir o respetivo download	Sucesso
N) Partilhar a músicas da sua biblioteca com outros utilizadores	Sucesso
O) Avaria de um servidor RMI não tem qualquer efeito nos clientes	Sucesso
P) Não se perde/duplica músicas se os servidores RMI falharem	Sucesso

Q) O serviço funciona desde que haja um servidor multicast disponível	Sucesso
R) Avarias temporárias (<30s) dos N servidores são invisíveis para clientes	Sucesso
S) Pedidos são sempre processados por N>=1 servidores multicast	Sucesso
T) Pedidos de leitura são respondidos apenas por um servidor multicast	Erro
U) Servidor RMI secundário testa periodicamente o primário	Sucesso
V) Em caso de avaria longa os clientes RMI ligam ao secundário	Sucesso
W) Servidor RMI secundário substitui o primário em caso de avaria longa	Sucesso
X) O failover é invisível para utilizadores (não perdem a sessão)	Sucesso
Y) O servidor original, quando recupera, torna-se secundário	Sucesso