Sistemas Distribuídos
Meta 2

Trabalho Realizado por:
- João Figueiredo 2015248636
- Pedro Chicória 2015262771

Meta 1

Rmi Client

O *rmi client* tem como objectivo estabelecer uma porta de comunicação entre o utilizador da nossa aplicação e o *rmi server*. No *rmi client* encontram-se todos os menus da nossa aplicação, neles é possível realizar diversas operações tais como a inserção de informação relativas a albums, artistas, musicas e playlists. É ainda possível realizar sobre esses objectos operações de edição, consulta e remoção.
O *rmi client* realiza essas operações recorrendo a métodos remotos que se encontram localizados no *server interface*, estes métodos são a forma como o *rmi client* comunica com o*rmi* server.

Rmi Server

O *rmi server* implementa os métodos remotos especificados no *server interface*, é dentro destes métodos que a comunicação é realizada entre o *rmi server* e o *multicast server*.
Estes comunicam com recurso a duas multicast sockets e a datagram packets, onde uma dais quais será responsável por enviar informação do *rmi server* para o *multicast serve*r outra pela recepção da resposta. Esta comunicação segue um protocolo que a semelhança do enunciado do mesmo segue um padrão de **chave|valor**.

Multicast Server

O multicast é responsável por receber os pedidos das funções remotas do rmi server e tratar de os satisfazer. O servidor multicast têm acesso a todas as classes de objectos usados para modelar as necessidades de construção da nossa aplicação. As classes criadas foram as seguintes:

- User

  -private String username;
  -private String password;
  -private boolean privilege;
  -private ArrayList<Playlist> playlists;

- Álbum
  - -private String album_name;
  - -private String record_label;
  - -private ArrayList<Review> reviews; -private ArrayList<Music> musics;
- Artist
  - -private String artist_name;
  - -private int age;
  - -private String music_style;
  - -private String biography;
  - -private ArrayList<Album> albums;
  - -private ArrayList<String> usernamesToNotificate;
- Music
  - -private String music_name;
  - -private String musical_gender; -private String lyrics;
  - -private String composer; -private String release_date;
- Review
  - -private String username;
  - -private String text; -private int rating;

Para armazenar os dados foram criadas duas ArrayLists, uma da classe Users e outra da classe Artist. Com estas duas classes é nos possível obter qualquer informação que o utilizador possa solicitar. Estas são guardas posteriormente sob a forma de um ficheiro de objectos. O multicast retorna então uma mensagem sobre o formato de string ao *rmi server* seguindo o protocolo estabelecido.

Protocolo

Todas as variâncias do protocolo aceites entre o *rmi server* e o *multicast* serão especificadas de seguida.

rmi server -> multicast server

- “type|check_login;username|" + username + ";password|" + password
- "type|check_user_privilege;username|" + username
- "type|check_user;username|" + username
- "type|check_artist;artist_name|" + artist_name
- "type|check_album;album_name|" + album_name
- "type|check_music;music_name|" + music_name
- "type|check_review;album_name|" + album_name + ";username|" + username
- "type|check_playlist;playlist_name|" + playlist_name
- "type|check_user_playlist;playlist_name|" + playlist_name + ";username|" +
- username
- "type|insert_user;username|" + username + ";password|" + password
- "type|insert_user_privilege;username|" + username
- "type|insert_artist;artist_name|" + artist_name + ";artist_age|" + artist_age +
- ";artist_music_style|" + artist_music_style + ";artist_biography|" + artist_biography

• "type|insert_album;album_name|" + album_name + ";album_record_label|" + album_record_label + ";artist_name|" + artist_name

• "type|insert_music;music_name|" + music_name + "; music_gender|" + music_gender + ";music_lyrics|" + music_lyrics + ";music_composer|" + music_composer + ";music_release_date|" + music_release_date + ";album_name|" + album_name + ";artist_name|" + artist_name

• "type|insert_review;username|" + username + ";description|" + description + ";rating|" + rating + ";album_name|" + album_name + ";artist_name|" + artist_name

• "type|insert_playlist;playlist_name|" + playlist_name + ";playlist_privilege|" + playlist_privilege + ";creator|" + creator

• "type|insert_music_playlist;playlist_name|" + playlist_name + ";music_name|" + music_name + ";album_name|" + album_name + ";artist_name|" + artist_name + ";creator|" + creator

• "type|edit_user;user_to_edit|" + user_to_edit + ";username|" + username + ";password|" + password + ";privilege|" + privilege

• "type|edit_artist;artist_to_edit|" + artist_to_edit + ";artist_name|" + artist_name + ";artist_age|" + artist_age + ";artist_music_style|" + artist_music_style + ";artist_biography|" + artist_biography

• "type|edit_album;album_to_edit|" + album_to_edit + ";album_name|" + album_name + ";album_record_label|" + album_record_label + ";artist_name|" + artist_name

• "type|edit_music;music_to_edit|" + music_to_edit + "music_name|" + music_name + "; music_gender|" + music_gender + ";music_lyrics|" + music_lyrics + ";music_composer|" +

music_composer + ";music_release_date|" + music_release_date + ";album_name|" + album_name + ";artist_name|" + artist_name

• "type|edit_review;review_to_edit|" + review_to_edit + "username|" + username + ";description|" + description + ";rating|" + rating + ";album_name|" + album_name + ";artist_name|" + artist_name

• "type|edit_playlist;playlist_to_edit|" + playlist_to_edit + ";playlist_name|" + playlist_name + ";playlist_privilege|" + playlist_privilege + ";creator|" + creator

- "type|remove_user;username|" + username
- "type|remove_artist;artist_name|" + artist_name
- "type|remove_album;album_name|" + album_name + ";artist_name|" +
- artist_name

• "type|remove_music;music_name|" + music_name + ";album_name|" + album_name + ";artist_name|" + artist_name

• "type|remove_review;album_name|" + album_name + ";username|" + username + ";artist_name|" + artist_name

• "type|remove_playlist;playlist_name|" + playlist_name + ";username|" + username

- "type|search_user"
- "type|search_artist"
- "type|search_album"
- "type|search_album_by_name;album_name|" + album_name
- "type|search_album_by_artist;artist_name|" + artist_name
- "type|search_music"
- "type|search_music_by_album;album_name|" + album_name + ";artist_name|" +
- artist_name

• "type|search_music_by_playlist;playlist_name|" + playlist_name + ";creator|" + creator

• "type|search_public_playlist"

- "type|search_private_playlist;username|" + username
- "type|details_user;username|" + username
- "type|details_artist;username|" + artist_name
- "type|details_album;album_name|" + album_name + ";artist_name|" +
- artist_name

• "type|details_music;music_name|" + music_name + ";album_name|" + album_name + ";artist_name|" + artist_name

• "type|details_review;album_name|" + album_name + ";artist_name|" + artist_name

multicast server -> rmi server

• type|check_login;status|on
• type|check_user_privilege;status|on• type|check_user;status|on
• type|check_artist;status|on
• type|check_album;status|on
• type|check_music;status|on
• type|check_review;status|on
• type|check_playlist;status|on
• type|check_user_playlist;status|on
• type|insert_artist;status|on
• type|insert_user;status|on
• type|insert_user_privilege;status|on• type|insert_album;status|on
• type|insert_review;status|on
• type|insert_music;status|on
• type|insert_playlist;status|on
• type|insert_music_playlist;status|on• type|edit_user;status|on
• type|edit_album;status|on
• type|edit_artist;status|on
• type|edit_music;status|on
• type|edit_review;status|on
• type|edit_playlist;status|on
• type|remove_user;status|on

- type|remove_artist;status|o
- type|remove_album;status|on
- type|remove_music;status|on
- type|remove_review;status|on
- type|remove_playlist;status|on
- type|search_user;list_count|" + Integer.*toString*(*users*.size()) + ";user_" +
- Integer.*toString*(counter) + "|" + user.getUsername()

• type|search_artist;list_count|" + Integer.*toString*(*artists*.size()) + ;artist_" + Integer.*toString*(counter) + "|" + artist.getArtist_name()
• type|search_album" + ";list_count|+ ;album_" + Integer.*toString*(counter) + "|" + album.getAlbum_name() type|search_album_by_name;item_count|+ album_" + Integer.*toString*(counter) + "|" + album.getAlbum_name()
• type|search_music;list_count|+ ;music_" + Integer.*toString*(counter) + "|" + music.getMusic_name()
• type|search_music_by_album;list_count|+ ;music_" + Integer.*toString*(counter) + "|" + music.getMusic_name()
• type|search_user;list_count|+ ;music_" + Integer.*toString*(counter) + "|" + music.getMusic_name() type|search_public_playlist;list_count|+ ;playlist_" +

Integer.*toString*(counter) + "|" + playlist.getName() type| search_private_playlist;list_count| + ;playlist_" + Integer.*toString*(counter) + "|" + playlist.getName()

- type|details_user+ ";password|" + user.getPassword() + ";privilege|" + user.getPrivilege().toString()
- type|details_artist+ ;age|" + Integer.*toString*(artist.getAge()) + ";music_style|" + artist.getMusic_style() + ";biography|" + artist.getBiography();
- type|album_details + ";record_label|" + album.getRecord_label()
- type|album_music + ;musical_gender|" + music.getMusical_gender() + ";lyrics|" + music.getLyrics() + ";composer|" + music.getComposer() + ";release_date|" + music.getRealease_date();
- type|reviews_details;list_count| + ;username|" + review.getUsername()+ ";description|" + review.getText(); + ";rating|" + Integer.*toString*(review.getRating());

Callbacks

Foi criada no *rmi server* uma ArrayList da classe client interface, para verificar quais os utilizadores que se encontram online a cada momento. Desta forma é possível notifica- los, assim que o seu privilegio dentro da aplicação ou alguma alteração nos dados da aplicação desde que para esta segunda opção eles tenham privilegio de editor.

Tratamento de excepções

Sempre que ocorrer alguma excepção no rmi client, este irá tentar fazer lookup do servidor rmi para detectar qual é que se encontra activo e se re-conectar. Desta forma asseguramos que para avarias menores a 30 segundos o client não se aperceba que um dos servidores rmi falhou.

Failover

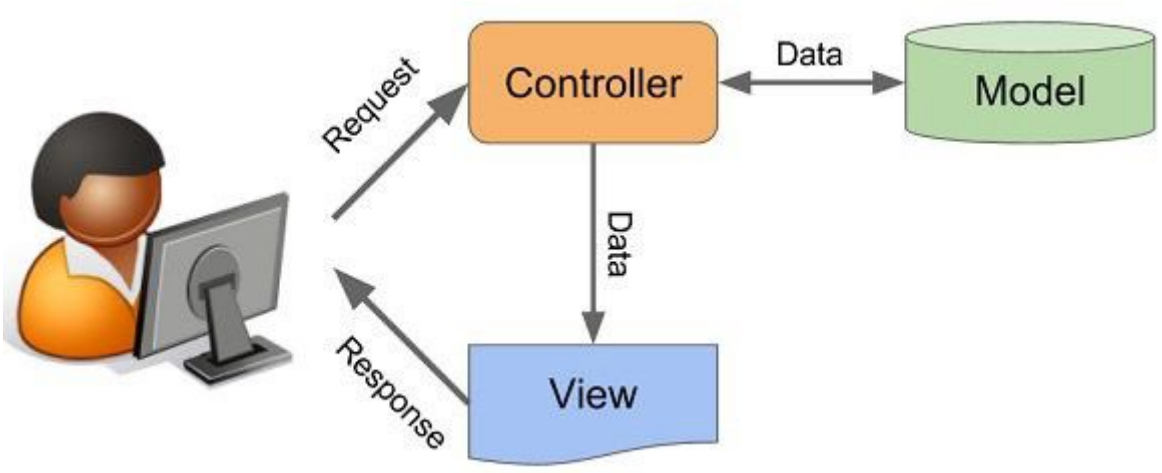
O servidor rmi está constantemente a realizar pings de forma a se assegurar que o segundo servidor rmi está ligado. Caso um dos servidores crash o outro irá passar a assumir o papel de servidor primário.

Checklist

Requisitos Funcionais	
Registar novo utilizador	✓
Login protegido com password	✓
Introduzir artistas, álbuns e músicas	✓
Pesquisar álbuns por artista e por título de álbum	✓
Consultar detalhes de álbum (incluindo músicas e críticas)	✓
Editar detalhes de álbum (incluindo músicas)	✓
Escrever crítica sobre um álbum (com pontuação)	✓
Consultar detalhes de artista (e.g., discografia, biografia)	✓
Dar privilégios de editor a um utilizador	✓
Notificação imediata de privilégios de editor (online users)	✓
Notificação imediata de re-edição de descrição de álbum (online users)	✓
Entrega posterior de notificações (offline users)	
Upload de ficheiro para associar a uma música existente	✓
Partilhar um ficheiro musical e permitir o respetivo download	✓
Grupos de 3: Criar grupo de amigos (-3)	
Grupos de 3: Owner de cada grupo gere os membros (-3)	
Grupos de 3: A partilha de informação está limitada aos grupos (-3)	
Grupos de 3: Editores que não são owners podem alterar dados (-3)	
Tratamento de Exceções	
Avaria de um servidor RMI não tem qualquer efeito nos clientes	✓
Não se perde/duplica músicas se os servidores RMI falharem	
O serviço funciona desde que haja um servidor multicast disponível	✓
Avarias temporárias (<30s) dos N servidores são invisíveis para clientes	✓
Pedidos são sempre processados por N>=1 servidores multicast	✓
Pedidos de leitura são respondidos apenas por um servidor multicast	
Failover	
Servidor RMI primário testa periodicamente o secundário	✓
Em caso de avaria longa os clientes RMI ligam ao secundário	✓
Servidor RMI secundário substitui o primário em caso de avaria longa	✓
O failover é invisível para utilizadores (não perdem a sessão)	✓
O servidor original, quando recupera, torna-se secundário	✓
Relatório	
Arquitetura de software detalhadamente descrita	✓
Detalhes do funcionamento do servidor UDP (protocolo multicast, etc.)	✓
Detalhes do funcionamento do servidor RMI (interface, javadocs, etc.)	✓
Distribuição de tarefas pelos elementos do grupo	✓
Testes de software (tabela com descrição + pass/fail de cada teste)	✓

Meta 2

Arquitetura Struts 2



Para este projeto foi usado um modelo MVC (Model View Controller). Para tal, foi usado a framework Struts 2, onde o *model* se chama *bean, o controller actions* e a *view, jsp (Java Server Pages).*

Optámos por um usar um bean para cada modelo de informação, isto é, os users têm um bean, os artistas têm um bean e assim sucessivamente. Todos os métodos da bean, excepto os setters e os getters das suas variáveis, dizem respeito a operações do RMI Server.
As action têm métodos que vão buscar as beans, que, por sua vez, são guardados na session ou para criar algo que ainda não exista na session. Quando guarda os beans na session, evitamos estar a criar vários beans repetidos para o mesmo utilizador.

Para a informação chegar aos beans, esta é submetida pelos utilizadores através de forms que estão nas views. Após a informação ser submetida, é chamada uma action que vai buscar a bean correta para o tipo de dados que foram submetidos à session. Caso não exista nenhum bean na session, este será criado e será posto na session. Depois, utilizando os setters da bean, a action vai preencher as variáveis do bean com a informação submetida. Assim, o utilizador é levado para uma página que vai buscar o bean, onde se encontram os dados submetidos à session, invocando o método do bean que irá retornar o resultado solicitado pelo utilizador.

Protecções

Quando o utilizador faz login, a informação deste é guardada num bean que é guardado na session. É também guardo na session um boolean para saber se este está logado. Toda a navegação é feita através de actions e todas as actions antes de executarem verificam se os dados na sessions relativos ao login estão corretos.
Deste modo evitamos que a autenticação seja evitada através da navegação pelo URL.

Integração Struts 2 com RMI

A conexão ao servidor RMI é feita durante a inicialização de um Bean ou seja, a ligação é estabelecida no construtor dos beans.
Após a ligação ser estabelecida, o bean guarda a referência numa variável que servirá para chamar métodos do RMI. Estes métodos são chamados nos métodos que usam as variáveis deste, que terão os valores que lhe foram passados pelo utilizador. O resultado da chamada RMI é depois devolvido pelo método do bean que executou a operação

REST API

Neste projeto, utilizamos a REST API da plataforma DropBox. Para garantir o bom funcionamento da mesma, nunca guardamos o access token do utilizador em memória, assim nunca iremos comprometer a segurança dos utilizador.
Para solicitar alguma informação da API, a 1ª action responsável pelo processamento do pedido, irá utilizar a API_APP_KEY e a API_APP_SECRET, obtida através do site https://www.dropbox.com/developers/apps/. Após isso, iremos usar um OAuthService, com estas duas keys e seguido de um callback, de que nos vai dar acesso ao url da dropbox, gerando um código, ou seja, o código de autorização. Posto isto, iremos fazer um request de modo a gerar um token de modo a ir buscar o user_id, user_id este que irá ser guardado na base dados para futuros requisitos. O processo descrito acima será o associar a conta. Posto isto, iremos passar o login, onde iremos outra vez gerar um novo código de autorização de seguida iremos fazer o token do user_id e comparar com os que estão na base de dados e caso seja igual, o login foi bem sucedido.

Web Sockets

Para conseguirmos cumprir os requisitos foi necessário tornar o server endpoint num servidor rmi. Para isso a class que server de server endpoint foi tornada num RemoteUnicastObject.
O servidor rmi contém assim um set com todas as websockets ativas. Sempre que um tipo de operação específico é realizado pelo rmi é invocado um método específico do server endpoint. Por exemplo quando é chamado uma função de notificar utilizadores sobre o rating dos álbuns, este informa as websockets que precisam de atualizar a tabela de albuns. É então invocado um método do server endpoint que invoca um método do rmi para atualizar a tabela com os novos dados.
Deste modo mantemos os clientes atualizados em tempo real.

Checklist

	Sistemas Distribuídos 2018/19 - Meta 2		
	Nome:	Aluno de SD	
	Número de Aluno:	2017987654	
100	Nota Final:	100	
50	Requisitos Funcionais	50	
5	Registar novo utilizador**	5	✓
5	Login protegido com password (acesso a todas as páginas)	5	✓
5	Introduzir artistas, álbuns e músicas	5	✓
5	Remover artistas que não tenham álbuns nem músicas	5	✓
5	Pesquisar álbuns por artista e por título de álbum	5	✓
5	Consultar detalhes de álbum (incluindo músicas e críticas)**	5	✓
5	Editar detalhes de álbum (incluindo músicas)	5	✓
5	Escrever crítica sobre um álbum (com pontuação)	5	✓
5	Consultar detalhes de artista (e.g., discografia, biografia)	5	✓
5	Dar privilégios de editor a um utilizador**	5	✓
0	Grupos de 3: Criar grupo de amigos (-5)		
0	Grupos de 3: Owner de cada grupo gere os membros (-5)		
0	Grupos de 3: A partilha de informação está limitada aos grupos (-5)		
0	Grupos de 3: Editores que não são owners podem alterar dados (-5)		
15	WebSockets	15	
5	Notificação imediata de privilégios de editor**	5	✓
5	Notificação imediata de re-edição de descrição textual**	5	✓
5	Atualização imediata da pontuação média das crîticas	5	✓
25	REST	25	
5	Associar conta de utilizador à Dropbox	5	✓
5	Associar ficheiro contido na Dropbox a uma música existente	5	
5	Partilhar música através da Dropbox	5	
5	Tocar uma música diretamente na página do DropMusic	5	
5	Login com a conta da Dropbox	5	✓
10	Relatório	10	
2	Arquitetura do projeto Web detalhadamente descrita	2	✓
2	Integração de Struts2 com o servidor RMI	2	✓
2	Integração de WebSockets com Struts2 e RMI	2	✓
2	Integração de REST WebServices no projeto	2	✓
2	Testes de software (tabela: descrição e pass/fail de cada teste)	2	✓
	Extra (até 5 pontos)	0	
	Utilização de HTTPS (4 pts)		
	Utilização em smartphone ou tablet (2pts)		✓
	STONITH nos servidores RMI (3p)		
	Outros (a propor pelos alunos)		