DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA
FCTUC

Relatório - Meta 2

Rui Miguel Freitas Rocha nº2010135453

Tabela de acrónimos

API - Application Programming Interface
MVC - model,view, Controller
REST - Representational State Transfer
RMI - Remote Method Invocation

Introdução

O objetivo deste trabalho era implementar um sistema distribuído de gestão de musicas. O trabalho foi dividido em duas partes, na primeira meta foi pedido para criarmos um ambiente distribuído usando arquitetura cliente-servidor, com recursos a *sockets multicast*, modelos *multi-threading*, *Java RMI* e tratamento de *failovers*. Os requisitos desta ultima meta é desenvolver um *Front End*, ou seja, uma interface web e ligar a aplicação criada anteriormente, com recurso a *struts2*, *JavaServer Pages*, *JavaBeans*, arquitetura *MVC*, *WebSockets* e serviços *REST*.

Arquitetura do Sistema

O sistema e composto por 2 servidores RMI, um primário e um backup, um servidor *multicast*(dataserver.jar) que armazena os dados do sistema, esta ligado ao servidor *RMI*(server.jar), via *multicast*, e um cliente RMI(console.jar), que se liga aos servidores RMI. O servidor *multicast* gere os pedidos por ambiente *multi-threding*.
Nesta meta foi criado uma arquitetura Web com recurso a *framework struts2*, java server pages e o servidor *tomcat*. Este servidor *Web*, permite ligar clientes e editores se ligam ao sistema, estando ligado ao servidor RMI desenvolvido anteriormente. O *WebServer* recebe pedidos provenientes das páginas Web,
comunica com o servidor RMI através das *actions* devolvendo uma resposta às páginas *Web*. As diferentes actions desenvolvidas correspondem às diferentes funcionalidades disponibilizadas utilizadores comuns e utilizadores editores. Foi integrado serviços *REST* com auxilio a *API* do *Facebook*, usando o protocolo *OAuth*. Esta comunicação permite associar uma conta do Facebook ao login e partilhar resultados de pesquisas. Foi implementado *WebSockets*, que permite visualizar alterações em tempo real.

Estrutura Interna do Sistema

A arquitetura interna do Servidor Web assenta num modelo *MVC*, suportado pelo *framework Struts2 (da Apache Foundation)*. Utilizando as funcionalidades de *Java Servlets*, este framework simplifica a tarefa de interligar componentes, oferecendo um ambiente de programação eficiente e rápido.

View

A View está materializada em ficheiros *JSP* que utilizam *HTML, CSS, Javascript, Boostrap* para apresentar a informação no ecrã. Esta informação, frequentemente dependente da ação requerida ou dos dados de um dado cliente, é armazenada em j*ava beans* e que são acedidos através de *Tags JSTL/Struts2*.

Controller

O Controller está materializado em *Actions* (métodos de classes em Java) invocadas pelo *Struts* (através de um pedido do utilizador) e que podem obter parâmetros via *GET* ou *POST* e definir parâmetros diversos das *Views*.

Model

Materializado em *Java Beans* auto-contidos. O *Controller* interage com o *Java Bean* para efetuar alterações no seu estado interno. Por exemplo, existe um método login(), do *Bean*, que altera o estado deste para refletir o utilizador que acabou de se autenticar. Este *Model* acaba por agir como uma camada de ligação ao servidor *RMI* onde os dados estão verdadeiramente armazenados e a lógica de acesso aos dados do sistema. O próprio acesso ao *RMI* é gerido por este *bean*, sendo totalmente desconhecido do *Controller*, uma vez que mesmo que o modelo de dados usados s e o servidor *RMI* fossem trocados por uma outra tecnologia equivalente, a interface de programação do *Controller* se manteria intacta.

Integração das Struts com o Servidor RMI

Na implementação do trabalho, segui o padrão de arquitetura *MVC*. A página (*View*) submete pedidos à *Action (Controller)*, que prepara o pedido para submeter ao *Bean* (*Model*). O *Bean* comunica com o Servidor *RMI*, executa o pedido e obtém uma resposta, resposta essa que é redirecionada para a página e assim sucessivamente. Desta forma, garante-se que tanto os utilizadores da aplicação como os utilizadores *Web* acedem aos mesmos dados.

Integração dos Serviços REST com o ucBusca

Para tornar a aplicação mais relevante nos dias de hoje, interligou-se esta com a *API* publica do *Facebook*, que utiliza *REST*.
Assim, sempre que um utilizador se regista na aplicação com uma conta do *Facebook*, ou sempre que associa a sua conta com uma conta do *Facebook*, é o *Facebook* que se encarrega de lhe fazer um pedido para conceder as permissões necessárias para que a aplicação possa realizar as funcionalidades do utilizador.

Sempre que um utilizador se pretende autenticar na nossa aplicação com as credenciais do *Facebook*, é estabelecido uma ligação aos servidores do *Facebook*, para que confirme a identidade do utilizador, sendo-nos permitido acesso a todas as funcionalidades da mesma.

Uma vez efetuado com sucesso a autenticação do utilizador no *Facebook* é nos fornecido um *Access Token*, que concede um acesso seguro e temporário as ações para as quais este obteve permissão da parte do utilizador. Este *Access Token* é em seguida utilizado para, utilizando *OAuth*, ser efetuado um pedido ao *Facebook*, de forma a validar a identidade do utilizador que se acabou de registar. Como resultado deste processo, o *Facebook* fornece um conjunto de dados acerca do utilizador.

Implementação de WebSockets

A implementação dos *webSockets* permite a atualização de conteúdo na aplicação web em tempo real. Neste caso, através destes, o administrador tem acesso imediato aos utilizadores que estão atualmente conectados à aplicação através da web. Quando um utilizador faz login na aplicação, a página *HTML*, ao carregar, chama o script de *JS* que cria o objeto *WebSocket*, levando como parâmetro o URI do servidor. O cliente utiliza duas funções, o evento *onOpen* e a função remove (invocada no *logout*), onde é enviado ao servidor. Quando o servidor recebe o objeto *JSON*, analisa o formato da mensagem e adiciona ou remove um utilizador, consoante o tipo.

Tabela de testes

50	Requisitos Funcionais	50	
5	Registar novo utilizador**	5	Passou
5	Acesso protegido com password (todas as páginas exceto pesquis	5	Passou
5	Indexar novo URL introduzido por administrador	5	Passou
5	Indexar iterativamente ou recursivamente todos os URLs encontrad	5	Passou
5	Pesquisar páginas que contenham um conjunto de palavras	5	Passou
5	Resultados ordenados por número de ligações para cada página	5	Passou
5	Consultar lista de páginas com ligações para uma página específic	5	Passou
5	Consultar lista de pesquisas feitas pelo próprio utilizador**	5	Passou
5	Dar privilégios de administrador a um utilizador**	5	Passou
5	Entrega posterior de notificações (offline users)	5	Passou
15	WebSockets	15	
5	Notificação imediata de privilégios de administrador (online users)*	5	Passou
5	Página de administração atualizada em tempo real	5	Passou
5	Atualização imediata da lista de servidores multicast ativos**	5	Passou
25	REST	25	
5	Associar conta de utilizador ao Facebook	5	Não Passou
5	Partilha da página com o resultado de uma pesquisa no Facebook	5	Não Passou
5	Mostrar em cada resultado a língua original da página	5	Não Passou
5	Traduzir título e descrição das páginas para Português	5	Não Passou
5	Registo com a conta do Facebook (sem conta ucBusca)	5	Não Passou
10	Relatório	10	
2	Arquitetura do projeto Web detalhadamente descrita	2	Feito
2	Integração de Struts2 com o servidor RMI	2	Feito
2	Integração de WebSockets com Struts2 e RMI	2	Feito
2	Integração de REST WebServices no projeto	2	Feito
2	Testes de software (tabela: descrição e pass/fail de cada teste)	2	Feito
	Extra (até 5 pontos)	0	
	Utilização de HTTPS (4 pts)		
	Utilização em smartphone ou tablet (2pts)		Passou

Conclusão

Em conclusão, o resultado final do trabalho pratico permitiu obter conhecimentos a nível pratico, de como se constrói um servidor WEB, integrando-o a um servidor já construído anteriormente (na primeira meta) e a sua ligação a APIs externas. Com isto podemos reiterar que os conhecimentos teórico-práticos sobre Struts2, serviços REST, e WebSockets, foram alcançados com sucesso. Deste modo estamos em condições de afirmar que os objetivos do trabalho pratico foram alcançados, mesmo que alguns dos requisitos funcionais não estejam implementados. Em suma o grupo de trabalho esteve ao nível do desafio proposto, e assim ganhando conhecimentos práticos, da implantação de um servidor WEB.