UNIVERSIDADE D COIMBRA

FACULDADE DE CIÊNCIAS E TECNOLOGIA

ucBusca: Motor de pesquisa de páginas Web

Projeto de Sistemas Distribuídos

David Jesus Vaz Cortesão Silva – 2008109004
Rui Mário Choupina Reis - 2013134606

Índice

Introdução

Este projeto tinha como objectivo permitir ao utilizador registar-se e fazer login numa plataforma onde seria possível realizar pesquisas não só através de palavras como também através de sites. Para isto o administrador do site, tem a possibilidade de indexar um ou vários sites, permitindo assim ao utilizador realizar pesquisas offline.

Além das funcionalidades referidas, seria necessário implementar outras funcionalidades tais como permitir ao utilizador consultar o histórico de pesquisas, verificar as pesquisas mais realizadas, dar permissão a utilizadores se tornarem administradores, traduzir resultados e ainda integrar com o facebook. Para isso, era proposto a integração da aplicação com serviços REST externos, usando OAuth, aplicar WebSockets para conseguir comunicar com os clientes em tempo real e utilizar uma arquitetura MVC, bem como Structs2, JavaServer Pages e JavaBeans.

Nesta segunda meta foi então proposto a criação de um frontend Web, que permitisse a integração com a aplicação desenvolvida na primeira meta. Assim os utilizadores têm as funcionalidades que já estavam disponíveis na meta 1 e todas as outras referidas anteriormente.

Arquitetura de software

Neste projeto apresentamos uma arquitetura geral conforme mostra a figura 1, sendo que na parte da esquerda estão todos os serviços implementados na meta um, que nesta segunda meta foram também significativamente melhorados e na parte da direita (amarelo), estão os serviços implementados nesta segunda meta.

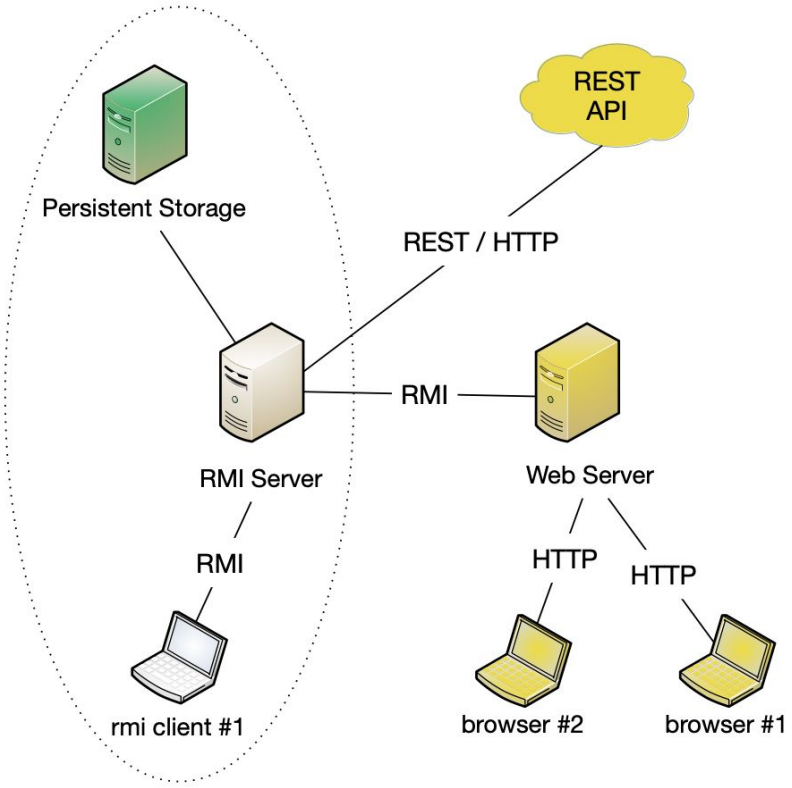


Fig. 1: Arquitetura do projeto

Nesta meta, os clientes interagem num browser que está ligado a um servidor HTTP (Apache Tomcat), que atua como um cliente do servidor RMI da primeira meta. O funcionamento do servidor RMI é idêntico ao da meta 1, inicializa o registo na porta 7000, o cliente vai fazendo chamadas às suas

funções retornando por exemplo “success” ou “fail”. Este Rmi, tal como na primeira meta, liga-se depois a um servidor multicast criado na meta um através de UDP.

Para apresentar todas as mensagens ao cliente no browser, bem como as comunicações, temos vários ficheiros. Na pasta JSP, temos todos ficheiros relativos às views, onde o cliente interage com a aplicação. Por exemplo, o menuUser.jsp, mostra o menu que vai aparecer ao utilizador depois de ele fazer login, possibilitando aceder a todas as funções a que um utilizador não administrador tem direito. Na pasta CSS, temos todas as configurações de design dessas views.

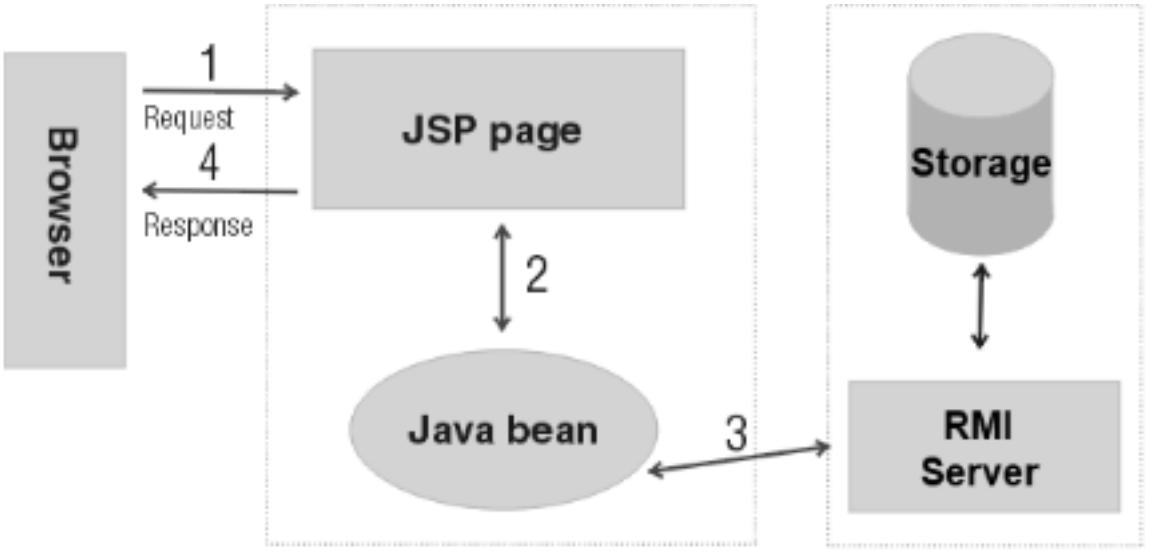
Relativamente aos models, temos um ficheiro HeyBean.java, que permite fazer todas as chamadas de funções ao rmi server da meta 1. Estas chamadas devolvem resultados, que em função do pedido devolvem num formato especial, por exemplo realizando um pedido de hitorico do utilizador, o rmi devolve uma string, que é depois apresentada no historico.jsp. Temos também outro ficheiro FacebookBean.java que apesar de ter o mesmo comportamento do ficheiro HeyBean, trata apenas das chamadas realiazadas em relação ao facebook, tal como registo de utilizador com o nome do facebook, a associação do nome do facebook.

Em relação aos controllers, estamos a fazer no ficheiro LoginAction.java onde são tratadas todas as chamadas realizadas nas views. Os pedidos recebidos das models são tratados e processados dando return de “success” ou “fail”, conforme o resultado.

Integração de Struts2 com o servidor RMI

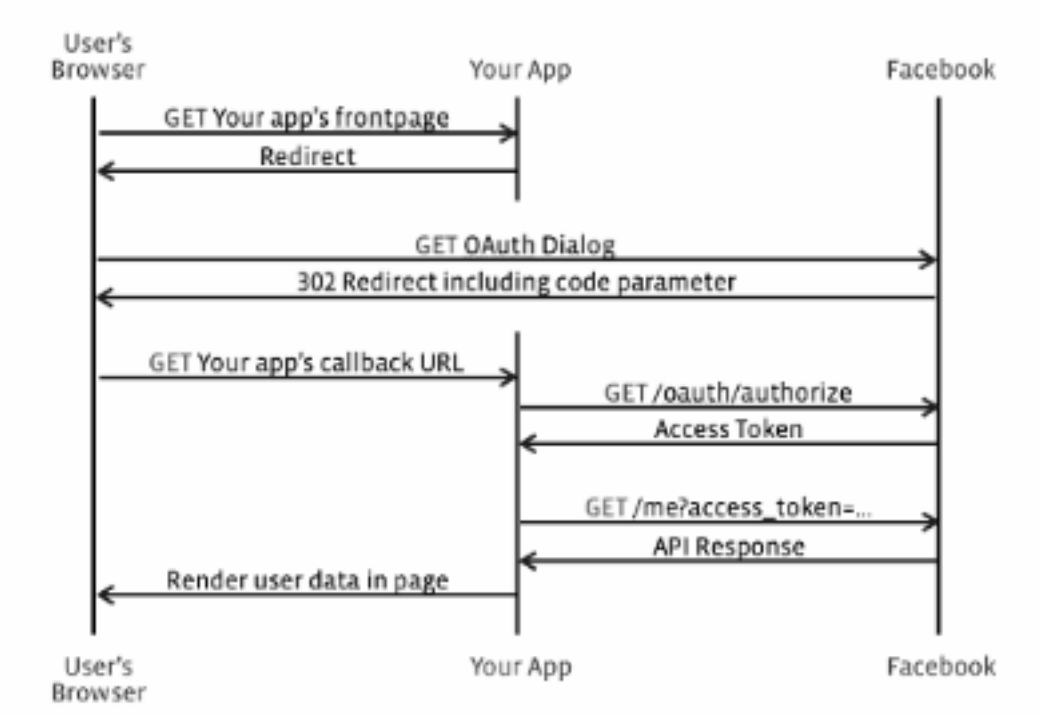
Para permitir a interação do utilizador com o servidor RMI através de um browser fizemos uso de JavaServerPages (.jsp), Struts2 (struts.xml), e JavaBeans. O utilizador ao interagir com as VIEWS (.jsp) vai chamar ACTIONS definidas no ficheiro "struts.xml", essas actions vão executar métodos da package action que vão inicializar um JavaBean, que vai fazer a ligação ao servidor RMI. Após essa ligação ao RMI ser feita o método pode chamar funções definidas no RMIInterface.

O return das funções do servidor RMI é analisado pelas action que vão fazer return do resultado, dependendo desse resultado é definido no ficheiro de STRUTS a página VIEW (.jsp) que vai ser apresentada ao utilizador.



Serviços REST usando OAuth

Para permitir que um utilizador entre no programa ou registe/associe a sua conta do Facebook, recorremos a serviços REST (Facebook API) e para manter os dados do utilizador seguro implementamos OAuth.



Quando o utilizador (browser) clica no botão (REGISTAR / LOGIN), é chamada uma ação que vai buscar o URL de autorização OAuth que permite o acesso ao facebook e aos dados requisitados através do parâmetro *scope*, o callback dessa ação vai nos redirecionar para uma outra ação onde através da informação recebida definimos os dados que pretendemos do Facebook do user, e chamamos a função (registo/login) do servidor RMI, o servidor RMI é responsável pela verificação dos dados e devolve o resultado da ação.

Em relação ao serviço de tradução criámos um javascipt "traduzir.js", que interage com o pedido realizado em relação à pesquisa, esse pedido é recebido em forma de string pelo rmi e e tratado nesse javascrit de forma a criar um array e posteriormente uma tabela. Os parametros de deteção da língua (titulo e descriçao), são analizados usando a api do yandex de detect, onde é enviada a key que foi criada por nós, bem como o texto e é-nos devolvido e apresentado

na tabela o idioma do texto fornecido através do resultado.lang. Para a tradução implementámos um botão que onclick, chama a função de tradução do yandex, enviando a apikey, o texto e o idioma para o qual queremos traduzir, neste caso "pt". É-nos então devolvido um resultado on lemos o resultado.text que corresponde ao resultado da string enviada e apresentamos esse resultado no espaço onde estava anteriormente em língua estrangeira.

Testes de Software

Requisitos Funcionais	
Registar novo utilizador	PASS
Acesso protegido com password (todas as páginas exceto pesquisas)	PASS
Indexar novo URL introduzido por administrador	PASS
Indexar iterativamente ou recursivamente todos os URLs encontrados	PASS
Pesquisar páginas que contenham um conjunto de palavras	PASS
Resultados ordenados por número de ligações para cada página	PASS
Consultar lista de páginas com ligações para uma página específica	PASS
Consultar lista de pesquisas feitas pelo próprio utilizador	PASS
Dar privilégios de administrador a um utilizador	PASS
Entrega posterior de notificações (offline users)	PASS

WebSockets	
Notificação imediata de privilégios de administrador (online users)	FAIL
Página de administração atualizada em tempo real	FAIL
Atualização imediata da lista de servidores multicast ativos	FAIL
REST	
Associar conta de utilizador ao Facebook	PASS
Partilha da página com o resultado de uma pesquisa no Facebook	FAIL
Mostrar em cada resultado a língua original da página	PASS
Traduzir título e descrição das páginas para Português	PASS
Registo com conta do Facebook	PASS