SISTEMAS DISTRIBUÍDOS
PROJETO – IDEABROKER
META 2

Grupo Constituído Por:

Bruno Miguel Oliveira Rolo nº2010131200

João Artur Ventura Valério Nobre nº 2010131516

INTRODUÇÃO

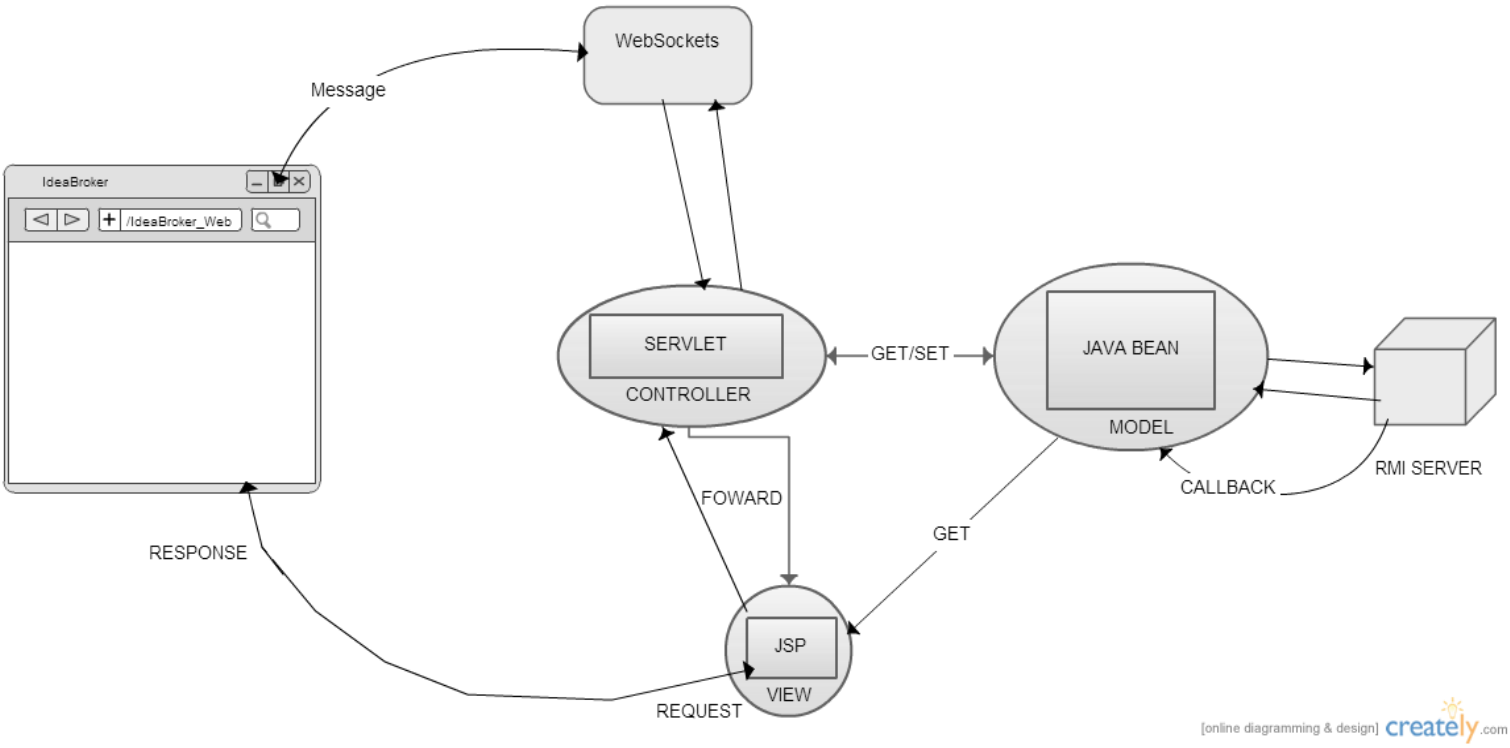
Este trabalho surge no âmbito da cadeira Sistemas de Distribuídos e tem como objetivo implementar uma rede de ideias como um sistema distribuído.

Na 2ºmeta deste projeto é necessário apresentar um sistema com uma interface web integrada na arquitetura do projeto da meta anterior. Serão utilizadas as seguintes tecnologias: Servlets, JSPs, JavaBeans, WebSockets e REST API.

Nas várias secções deste relatório, descrevemos de que forma abordámos estes requisitos no nosso trabalho.

ARQUITECTURA WEB-SERVER (MVC)

De seguida apresenta-se uma imagem ilustrativa da arquitetura da aplicação:



Neste projeto utilizámos uma arquitetura baseada no MVC (Model-View-Controller).

Em regra geral o funcionamento é o seguinte: o browser faz os pedidos e recebe as respostas através de uma página em html ou JSP, que tem o papel de "View". Este por sua vez envia os pedidos a um Servlet que tem a função de "Controller", o qual comunica com um javaBean ("Model") e processa a informação que recebe deste e envia a resposta em sentido contrário. O "Model" comunica com o server RMI onde são guardados os dados de todo o programa. Nestas situações, é chamado o servlet através da action de um formulário, o qual processa o pedido e comunica com o javaBeans de acordo com os dados recebidos. De seguida volta a fazer forward para a página html ou JSP de onde partiu o pedido através das instruções:

dispatcher = request.getRequestDispatcher("/inicio.jsp");

dispatcher.forward(request, response);

o qual mostra de seguida a resposta no browser.

Em alguns casos o “View” solicita dados diretamente ao “Model”, como é o caso da listagem de tópicos e na listagem de ideias.

Neste caso é acede-se diretamente ao javaBean “IdeaBrokerBean” guardado em sessão através da sintaxe:

<jsp:useBean id="ideabrokerbean" class="IdeaBroker.IdeaBrokerBean" scope="session" />

e chama-se um método existente no javaBean o qual comunica com o servidor.

São utilizados também webSockets para as notificações em tempo real quando uma transação ocorre, o utilizador que previamente tinha as acções é notificado e para actualizar em tempo real quem estiver na view de listagem de preços de uma ideia ao qual tenha ocorrido uma transação. Neste caso a iniciativa parte do servidor através de callback para o javaBean, o qual envia dos dados diretamente para o browser.

INTEGRAÇÃO DO PROJETO 1

Este projeto devia expandir o trabalho efetuado na Meta 1, utilizando o código produzido , nomeadamente a vertente do RMI ,para criar um novo modo de acesso (Web) a este serviço.

Dado que a maior parte das funções que efetuavam o trabalho interno no servidor (gestão das ideias, utilizadores etc...) já se encontravam construídas, foi apenas necessário efetuar a chamada remota a partir do Apache Tomcat através da interface RMI.

Mesmo assim foi necessário actualizar e adicionar algumas novas funções dado que apareceram novas funcionalidades pedidas tais como:

- Hall of Fame
- Sistema Root
- Websockets

WEB-SOCKETS

A tecnologia recente, denominada por WebSockets foi utilizada ao nível das notificações de eventos em tempo real.

A estrutura das ligações é geral para os dois casos, isto é, o browser faz um pedido de ligação a um endereço representativo do websocket, esse endereço corresponde ao servlet em causa que está a receber as ligações. A partir do momento em que a ligação está estabelecida, é possível escrever dados para o socket, e dessa maneira o browser apresentar os dados directamente.

Atualização de preços

Para atualizar a listagem de preços, usou-se uma servlet "UpdatePriceServlet" para receber e armazenar as ligações provenientes de cada utilizador que se ligava á view que mostra os preços.

Desta maneira era possível mais tarde escrever diretamente para os sockets dessas ligações.

Sempre que é efetuada uma transação, o servidor RMI, por via callback vai chamar um método do bean "updateprice", em que esse método vai tratar de escrever para todas as ligações online, acedendo a elas e utilizando o método writeTextMessage(buffer) para escrever dados.

Notificações de eventos de transação

Para tornar a notificação de eventos em tempo real possível, usou-se um servlet "NotificacaoWebSocketServlet" para receber e armazenar as ligações provenientes de cada utilizador que se ligava á view que ilustra as notificações.

Desta maneira era possível mais tarde escrever diretamente para os sockets dessas ligações.

Sempre que é efectuada uma transacção, o servidor RMI, por via callback vai chamar um método do bean "addNote", em que esse método vai tratar de escrever para o utilizador correspondente.

REST

Foi-nos pedido para implementarmos a funcionalidade de interação entre o nosso sistema social e o Facebook. Para isso necessitámos de utilizar a tecnologia REST na comunicação com o servidor Facebook.

Era necessário a implementação das funcionalidades de:

- Criação de Post
- Criação de Comentários em Posts já existentes
- Eliminação de Post
- Login com as credenciais do Facebook

Para tratar da autenticação do utilizador e da aplicação no Facebook foi utilizado o scribe-java, que disponibiliza uma API para o OAuth (sistema de autorização utilizado pelo Facebook).

Para o utilizador se autenticar é necessário que ele aceda a um link no servidor do Facebook, pelo que ai deixa de se encontrar no nosso sistema.No facebook a aplicação criada está configurada para que assim que o utilizador se autenticar no facebook e dar as permissões necessárias para o funcionamento da aplicação, o facebook redirecionará o utilizador para uma página que nós definimos com um Token neste caso irá ser feito o calback para localhost.

Após tratar da autenticação basta armazenar o código de autenticação devolvido e utilizá-lo para fazer o sign dos pedidos aos seguintes URLs.

Estes vão devolver um output formatado em XML, que teve de ser tratado através do uso do DOMParser, de forma a obter o conteúdo necessário.

TESTES EFETUADOS

Para garantir a estabilidade da aplicação no decorrer de um uso normal, foram efectuados vários testes de vária natureza, de seguida apresenta-se um quadro com várias entradas correspondentes aos teste, em que cada entrada apresenta a natureza do teste, a descrição do mesmo, o resultado esperado e, por fim o resultado positivo ou não do teste.

Natureza do teste	**Descrição do teste**	**Resultado esperado**	**Sucesso/Insucesso**
Registo	Inserir dados incorretamente e submeter	Página com aviso de insucesso no registo	Sucesso
	Inserir dados corretamente e submeter	Página de registo aceita redirecionamento para login	Sucesso
Login	Inserir dados incorretamente e submeter	Página com aviso de insucesso no login	Sucesso
	Inserir dados corretamente e submeter	Página de registo aceita redirecionamento para login	Sucesso
Logout	Utilizador carrega no botão do Logout	Termina a sessão e recaminha para login	Sucesso
Criação do Tópico	Inserir corretamente o campo do Tema	Insere com sucesso e actualiza o sistema	Sucesso
	Carrega no botão de inserir um topico vazia	Avisa o utilizador que o campo têm que ser preenchido	Sucesso
Criação de Ideia	Insere os dados corretamente	Insere os dados corretamente no sistema e redireciona para página de listagem de ideias	Sucesso

Apagar uma ideia	Carrega na Ideia que pretende apagar	Apagar do sistema a ideia	Sucesso
	Utilizador altera dados no url	Avisa Utilizador de algum erro e redireciona	Sucesso
Comprar Acções	Utilizador X compra ao Utilizador Y a totalidade das suas acções sendo a opção mais barata	Sucesso	Sucesso
	Utilizador X compra um slice das acções de um utilizador o + barato possivel	Sucesso	Sucesso
	Utilizador X compra ao Utilizador Y as suas acções mas não está disposto a investir tanto dinheiro na ideia	Insucesso na compra	Insucesso na Compra
Views: Hall of fame, profile, lista da ideia	Utilizador clica no botão e é ilustado todo a informação no sistema	Lista toda a informação pretendida na view	Sucesso
Sistema de root	TakeOver de uma ideia e adicionala ao hall of fame	Sucesso	Sucesso
REST	Login iniciar com login associado a uma conta do Facebook	Efectua Login com autenticão a partir do OAUTH, seguindo com o sistema de forma transparente	Sucesso
	Inserir Ideia no sistema	Inserir ideia no sistema e faz um post no facebook na conta que o utilizador está logado	Sucesso
	Comprar Acções	Após uma transação é comentado no post dessa ideia pelo	Sucesso

		utilizador que efetuou a compra	
	Login com as credenciais do Facebook	Inserindo as credenciais do Facebook consegue iniciar sessão no sistema	Insucesso
WebSockets	Utilizador X está na Página Inicial , e Utilizador Y compra ações ao Utilizador X	Utilizador X é notificado em tempo real das ações compradas	Sucesso
	Utilizador está na View de listagem de ações de uma Ideia, enquanto ocorre uma transação na mesma Ideia	Sistema deve, em tempo real, actualizar essa View para todos os utilizadores presentes na mesma	Sucesso