SD

Meta 2

Elaborado por:

Oleg Tryembich - 2011159465

Miguel Veloso - 20111527540

UNIVERSIDADE DE COIMBRA

Índice

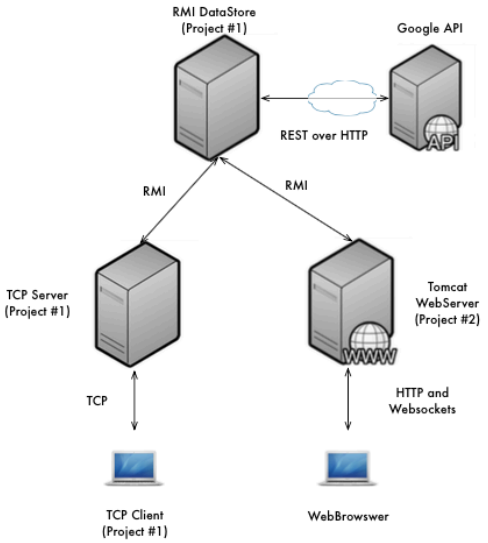
1 – Introdução

O trabalho apresentado tem como objectivo o desenvolvimento da aplicação exibida na meta 1 utilizando novos conceitos. Para isso foi necessária a instalação do Tomcat, um servidor web, estando este incumbido da execução de uma estrutura composta por páginas .jsp com integração de Struts que nos permitem realizar ações recorrendo a classes que interpretam e finalizam pedidos do utilizador.

As páginas .jsp são encarregadas de apresentar ao user final as funcionalidades da aplicação. Já as struts tem o papel de redirecionar o pedido do utilizador para um classe que vai processar o mesmo. No caso de sucesso o utilizador é redirecionado para o .jsp (página) seguinte, caso contrário é apresentado um erro. Na meta 2, é também necessária a implementação de websockets para gerar notificações e para os users terem comunicação directa no chat.

Para esta submissão foi necessária a aprendizagem de novos conceitos tal como programação web, websockets, struts, e REST para implementação da API do Google.

A imagem seguinte mostra a estrutura do trabalho a ser implementado.



2.1 – Web-based Internal Architecture

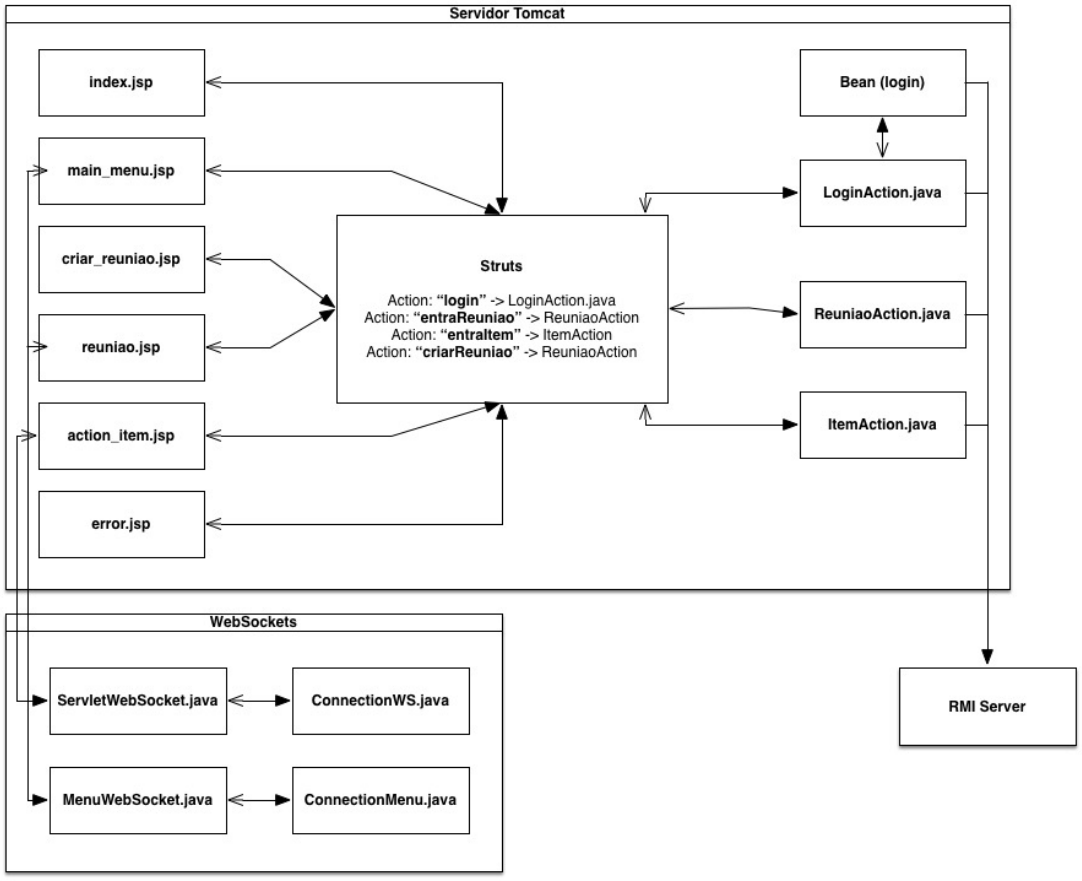


Figura 1 – Web-based internal architecture

A figura 1, acima, demonstra como o projeto desenvolvido por nós está arquitetado.

Dispomos de 5 ecrãs principais (índex.jsp, main_menu.jsp, reunião.jsp, action_item.jsp e criar_reuniao.jsp), que em cada ação despoletada pelo utilizador informam o "struts.xml" de que pretendem executar certa classe. Para tal, criamos 3 principais classes que tratam os pedidos enviados pelas ações através do "struts.xml": LoginAction, ReuniaoAction e ItemAction.

As classes abordadas anteriormente dispõem de métodos que realizam, executam e finalizam o pedido, guardando na "session" o conteúdo pretendido. Em caso de sucesso, o utilizador é redirecionado para o ecrã pretendido, dependendo do tipo de ação por ele provocada.

Essas ações dependem de uma ligação ao RMI para finalizarem pedidos ou consultar informação. Para isso as classes que tratam das solicitações dos utilizadores, utilizam a ligação ao RMI para consultar/inserir dados.

Dentro do menu principal, da reunião e do action item, é criada uma ligação ao "websocket" para haver comunicação direta entre utilizadores. No action item é feita uma conexão ao ServletWebSocket.java que por sua vez guarda a informação de cada utilizador na classe ConnectionWS. Relativamente ao menu principal e reunião, estes ligam-se ao MenuWebSocket.java, para o envio imediato de convites, action items, tarefas e decisões, guardando a informação de cada utilizador na classe ConnectionMenu.

Para segurança, no Login a estrutura de implementação é semelhante, com a diferença de que as credenciais são armazenadas num "bean" e o mesmo é executado pela classe LoginAction para autenticação do utilizador na plataforma.

2.2 – How struts are integrated with RMIServer

Apesar de uma solução viável ser a construção de uma classe para apenas tratar pedido feitos ao RMI, decidimos optar por fazer os mesmo diretamente através da classe responsável pela ação.

Para isso, aquando da submissão da ação por parte do utilizador, o "form" correspondente à mesma, envia os parâmetros necessários para a interpretação do pedido por parte da classe responsável pela ação, utilizando a ligação ao RMI para satisfazer o pedido retornando uma resposta ao utilizador.

2.3 – WebSocket integration with Struts

No nosso trabalho, integramos websockets em 3 sítios distintos: no menu principal onde são recebem os convites, no menu da reunião e dentro do action item.

Para o desenvolvimento criamos 4 classes diferentes responsáveis pelo tratamento das ligações e do envio/recepção das informações: ServletWebSocket, ConnectionWS, MenuWebSocket e ConnectionMenu.

As classes ServletWebSocket e ConnectionWS tratam da ligação dos utilizadores dentro do chat. A classe ConnectionWS guarda a informação de cada utilizador ligado, tendo como parâmetros o seu username, a reunião onde se encontra e a action item em utilização. Dessa forma torna-se mais fácil para a

classe ServletWebSocket redirecionar mensagens, filtrando e enviando-as apenas para utilizadores na mesmo "localização" do remetente.

As classes MenuWebSocket e ConnectionMenu são responsáveis pelo tratamento da informação enviada e recebida pelos utilizadores localizados nos menus principal e reunião. A classe ConnectionMenu tem o mesmo papel da classe ConnectionWS, guardar a informação do utilizador ligado para posteriormente a classe MenuWebSocket poder filtrar e enviar as informações aos utilizadores pretendidos.

Estas classes, são também responsáveis por guardar no RMI os convites, action items, conversas de chat, tarefas e decisions, pois torna-se mais simples utilizar este processo aquando do envio de novas informações que vão ser dissipadas para todos os utilizadores.

Finalmente, para poder utilizar este tipo de conexão foi necessária uma pequena alteração nos "struts.xml", executando um comando que nos permite aceder aos websockets, directamente de uma chamada em javascript, sem que seja necessária a execução de uma ação.

2.4 – Struts/RMIServer integration with Google

Nesta etapa é necessária a autenticação do utilizador para ser retornada uma access token para todas as operações serem aceites. Para isso, o botão "Google" no índex.jsp realiza uma ação executada pela classe GoogleAuthHelper que, com acesso ao cliend id, chave secreta e scope, executa o pedido à Google para que as ações correspondentes ao criar reunião, etc., sejam permitidas.

Após termos acesso à access token, esta é guardada no RMI, para que todas as operações que necessitem da interação com a API do Google sejam executadas e enviadas para eles.

Desta forma, as struts contêm uma ação correspondente ao botão "Google" do índex.jsp que reenvia o pedido para a classe correspondente para a autenticação para que posteriormente o RMI tenha acesso e consiga executar as operações que exigem interação com o Google Calendar.

2.5 – Decription Tests

Teste:	Funciona:
Servidor corre em diferentes máquinas	Sim
Registar User	Sim
Secure Login	Sim
Criar Reunião	Sim
Convidar Utilizadores	Sim
Aceitar convites de reunião	Sim
Ver info da reunião	Sim
Adicionar item	Sim
Modificar item	Sim
Apagar item	Sim
Adicionar key decision	Sim
Adicionar tarefa	Sim
Marcar tarefa como concluida	Sim
Adicionar mensagens de chat	Sim
Novas mensagens são vistas instantaneamente	Sim
Convites vistos instantaneamente	Sim
Item visto instantaneamente	Sim
Associar user com google account	Não
Login com Google info	Não
Novas reuniões criadas no Calendar	Não
Updates no Google Calendar	Não
Aceitar/recusar reunioes com o Google Calendar	Não

Figura 2 – Description tests

3 – Conclusão

O objectivo deste trabalho prendia-se com o desenvolvimento de uma aplicação web utilizando um servidor Tomcat para ser executado.

Foram cumpridos todos os requisitos necessários para submissão à excepção da API do Google devido a problemas na autenticação, apesar de termos conhecimento do conceito e de como o aplicar ao nosso trabalho.

Problemas a serem relatados: dificuldade na implementação da autenticação da API do Google mesmo tendo acesso a um exemplo fornecido pelos professores.