Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informatica

Sistemas Distribuídos
2013/2014 - 1º Semestre

Projecto #2

2013/2014

IdeaBroker

Idea Management and Trading

Elementos

- **André Perdigão da Costa de Sá Gonçalves**
  - 2010133096
  - apcosta@student.dei.uc.pt

- **José Miguel Cruz Alves**
  - 2010132936
  - jmcalves@student.dei.uc.pt

Introdução

A realização deste trabalho tem como objectivo a criação de um fórum de gestão e comércio de ideias. Um utilizador tem um vasto leque de opções a realizar, desde as mais básicas, tais como fazer um registo e login, até vender ou comprar *shares* de uma ideia. A implementação deste fórum é feita segundo um sistema distribuído, ou seja, um sistema em que várias máquinas trabalham em conjunto de forma a realizar uma dada tarefa.

Para que a aplicação seja bem sucedida tem de haver comunicação constante entre as máquinas que constituem o sistema. Este processo é determinante para a fiabilidade dos dados.

Os dados devem ser guardados em ficheiros ou base de dados, para garantir a sua segurança e integridade. Deste modo, sempre que é executada uma alteração no fórum, os dados devem ser atualizados.

Utilizando tecnologia para a web, desde Java EE, Websockets, Javascript, MySQL, irá ser implementado um serviço na web que permite o comércio de ideias. Fazendo uso destas tecnologias, é possível implementar um serviço em que os dados se mantenham actualizados a cada segundo numa interface fácil de usar.

Web-based internal architecture

A arquitetura MVC foi a base de desenvolvimento deste projeto. Esta arquitetura baseia-se na divisão da aplicação em três camadas: Modelo, Visualização, e Controlo.

Modelo:

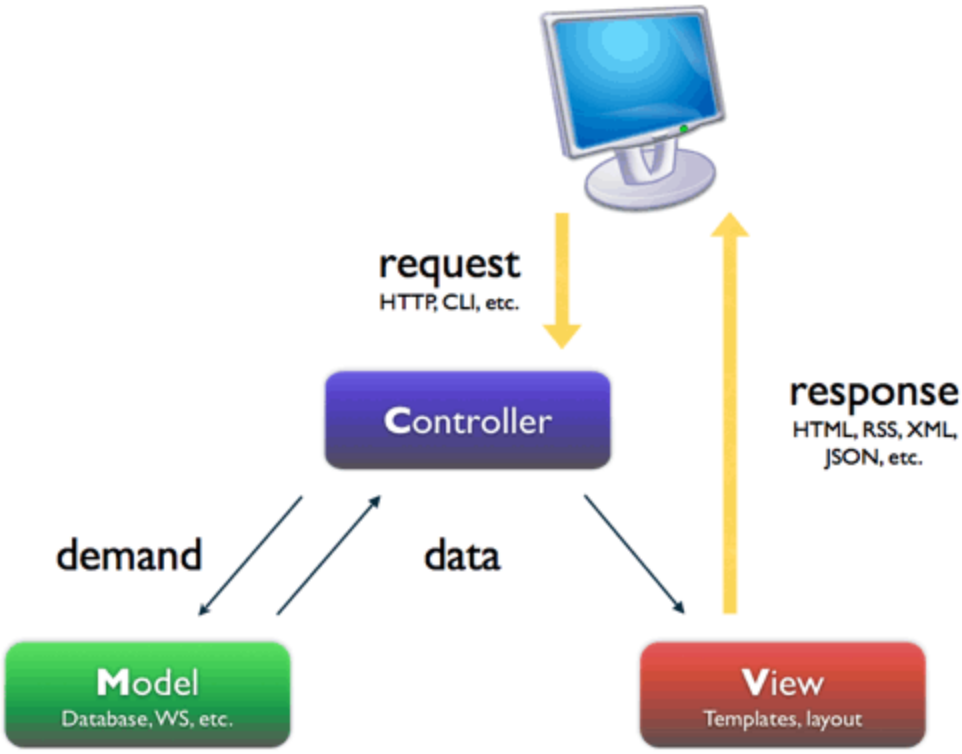
Representa a parte de lógica de negocio e controlo. Tem a responsabilidade de realizar as operações de manuseamento de dados com a base de dados, calcular os custos totais e de processar ordens.

Visualização:

Responsável pela apresentação da informação ao cliente. Deve ter a capacidade de direcionar a informação consoante a plataforma de acesso e a competência de mostrar os dados obtidos pela camada de lógica de negocio.

Controlo:

Serve de ligação entre a camada de visualização e de negocio. As decisões estão dependentes do tipo de cliente, consoante a ligação utilizada. Assim sendo, deve ter o cuidado de retornar a informação necessária entre as operações.



Na programação Web deste projeto, podemos distinguir 2 tipos de elementos fundamentais: i) JSP, ii) Servlets. Cada um dos elementos tem um objetivo distinto no projeto, interactuando segundo o modelo descrito acima.

A camada de visualização ficou ao cuidado dos JSP. Estes ficheiros incorporam todo o código HTML e CSS necessário para mostrar a informação aos utilizadores, operações lógicas básicas, como *if*'s ou *for's,* e algum código JavaScript, especialmente útil na utilização de WebSockets, devido ao facto de ser orientada a eventos.

As Servlets foram concebidas para estabelecer estruturas de decisão sobre o fluxo do programa. Os WebSockets também foram criados a partir de Servlets, sendo uma extensão da classe WebSocketServlet. Ao serem iniciados, entram no método *createWebSocketInbound*- que faz um retorno

de um novo objeto extensível da classe *MessageInbound.*- Esta classe começa por acrescentar este novo objecto às conexões da Servlet, contendo toda a informação que se deseja sobre um cliente. De resto, esta classe vai definir métodos orientados a eventos que vão proliferar mensagens consoante o efeito pretendido.

Integration with Server from Project 1

Um dos requisitos para este projecto era a integração com o projeto anterior. Sendo assim, na chamada de login ou registo é criado um novo objeto que vai ser guardado na sessão e que contém uma ligação RMI e os métodos específicos para cada uma das funções descritas na interface RMI do servidor, entre outras.

A ligação com o servidor foi iniciada a partir de uma estrutura de controlo, que criou o novo objecto e chamou de imediato um método de contacto com o servidor, através do protocolo referido.

A chamada de funções vai ser idêntico ao de um cliente RMI normal, com a exceção de agora passar várias camadas, começando por um pedido de um JSP, se for o caso passar para uma Servlet, invocar alguns métodos do objeto guardado na sessão, que contactam o servidor do Projeto 1, e por fim retornar dados ao JSP.

WebSockets integration

O WebSocket foi criado a partir de Servlets, sendo uma extensão da classe WebSocketServlet. Ao ser iniciado, entra no método *createWebSocketInbound*- que faz um retorno de um novo objeto *ChatMessageInbound* extensível da classe *MessageInbound.*-

Esta classe começa por acrescentar este novo objecto às conexões da Servlet, contendo toda a informação que se deseja sobre um cliente. Ela é responsável por mostrar notificações instantaneas aos clientes envolvidos numa transação e também por actualizar o preço das shares em tempo real. Todo este processo é complementado pela página que contém o JavaScript, uma linguagem orientada a eventos, que conduz as mensagens a serem enviadas para o WebSocket e trata da exibição das mensagens que são recebidas. Para tal, o WebSocket é chamado sempre que há uma alteração do preço de uma share, ou caso exista uma nova compra. Ao iniciar uma página que contém JavaScript com interação com WebSockets, após a página ser carregada, é iniciado um método que vai fazer uma ligação de protocolo ws na nossa pasta de execução e que procura a servlet que contem o WebSocket. Após isso, são declarados os métodos que vão ficar encarregados dos eventos gerados na variável e que fazem os tratamentos referidos acima.

REST web service

O sistema Rest baseia-se numa arquitetura cliente-servidor, em que não são os métodos que ficam acessíveis remotamente, mas sim os recursos e estes é que vão conter os métodos, através de um URL.

Esta tecnologia caracteriza-se por ser stateless, ou seja, em cada pedido deve ser garantida a informação necessária para a execução da requisição pretendida. Outros fundamentos prendem-se com a utilização de uma interface uniforme, com os métodos GET, para requesitar informação, POST, para criar nova informação e DELETE para eliminar informação.

Neste projeto era pretendido a comunicação com o REST do Facebook, para utilizadores autenticados através desta mesma rede social, possam automaticamente enviar as suas ideias criadas bem como respostas ao comprar shares de uma ideia.

Foi utilizada a nova API do Facebook, Open Graph, para efectuar as várias operações necessárias ao funcionamento pretendido. Um utilizador ao associar a sua conta Facebook à conta no IdeaBroker, são-lhe pedidas permissões (publish_actions). Utilizando JavaScript, o Facebook é contactado e devolve diferentes objectos dependendo do estado do utilizador. Se o mesmo deu permissões e se encontra autenticado, devolve um token de acesso, o seu userID e uma assinatura especial. Estes dados dão passados à servlet de autenticação pelo Facebook (Fblogin) e a mesma confirma a validade da assinatura, impedindo tentativas de login falsos.

Para a inserção de informação no feed de cada utilizador, poderá ser utilizado o accessToken proveniente do login, bem como um token da aplicação que permanece estático desde a criação da aplicação no Facebook.

Description of tests

Functional Requirements

Servers run on different machines (2 laptops during the defense) ✓

- Servidor RMI e webserver funcionam em diferentes máquinas

Register ✓

- Registo de utilizador funcional. Se nome de utilizador já existir, devolve erro.

Login User (page accesses should be protected) ✓

- Login funcional. Se username e password existirem, faz login. Devolve erro caso utilizador não exista ou a password esteja incorrect

Show Topics ✓

- Página dedicada a mostrar todos os tópicos existentes, bem como a contagem de ideias por tópico

Create Topic ✓

- Um tópico é criado ao ser submetida uma nova ideia. Caso seja inserido um tópico já existente, não é criada uma entrada em duplicado

Create new Idea ✓

-

Delete an Idea ✓

- Apagar uma ideia encontra-se funcional e apenas realiza o acto de apagar caso o utilizador possua 100% das shares da ideia

Search and List Ideas (including portfolio) ✓

- Permite a pesquisa de ideias e tópicos com uma ou mais keywords

Set share selling price ✓

- Utilizando AJAX, o preço de determinada ordem pode ser actualizado sem mudar de página

Buy shares of an idea

- Novas ordens de compra são criadas por AJAX, não mudando de página e a informação resultante dessa compra é interpretada pelo Javascript e caso a ordem tenha completado (de forma parcial ou total) é reenviado o resultado para o Websocket, partilhando a informação do preço por todos os clientes

✓

Root takes over ideas (hall of fame shows these ideas)

- Não implementado

✗

Show Transaction history

- Página dedicada que recolhe o histórico de cada utilizador da base de dados e mostra a informação

✓

(Groups of 3) Create Topic and Idea inside a group

- Ao criar uma nova ideia, é possível seleccionar um grupo caso se deseje privar a ideia. Todos os tópicos criados para esta nova ideia serão associados ao grupo escolhido.

✓

(Groups of 3) Add and remove user from groups

- Através de AJAX é possível adicionar e remover utilizadores de um grupo, fazendo uma pesquisa por ID ou username

✓

Interoperability with Project 1 (new ideas are shown in Web clients and TCP clients)

- Dadas as alterações feitas no RMI, não foi possível efectuar as mesmas alterações no projecto anterior de forma a deixá-lo operacional

✗

MVC Architecture (No Java inside JSPs)

- Existe código java dentro de ficheiros JSP que permitem mostrar informação diferente dependendo de estados do utilizador.

✗

MVC Architecture (No HTML inside Java classes). ✓

WebSockets

Transaction Notifications should be pushed to the Browsers instantly ✓

- Ao efectuar uma compra/venda, os utilizadores envolvidos na troca são notificados com sucesso

Transactions originating in TCP clients are pushed to Browsers instantly ✗

Up-to-date information regarding the last trade price per share ✓

- Da mesma forma que os clientes são notificados das suas trocas, o preço relativo a esta troca é espalhado por todos os utilizadores conectados

REST

Associate a user account to a facebook account ✓

- Havendo um utilizador autenticado no sistema, o mesmo dispõe de uma opção que permite associar a sua conta ao Facebook, guardando o userID na base de dados para futuras autenticações.

Login with facebook account instead of regular credentials ✓

- Caso o utilizador possua uma conta com acesso por Facebook, o mesmo pode autenticar no sistema se possuir uma sessão activa no Facebook

New ideas are posted on Facebook ✓

- Novas ideias são automaticamente inseridas no feed do utilizador que publicou a ideia, caso este possua uma conta associada ao Facebook

Delete an idea from Facebook when it is deleted in IdeaBroker

- Ideias em que o utilizador possua 100% das shares, é possível remover a ideia juntamente com o post no Facebook, se este existir.

✓

Post a comment on Facebook when shares are bought

- Nos testes realizados, nem sempre um comentário é removido

✓