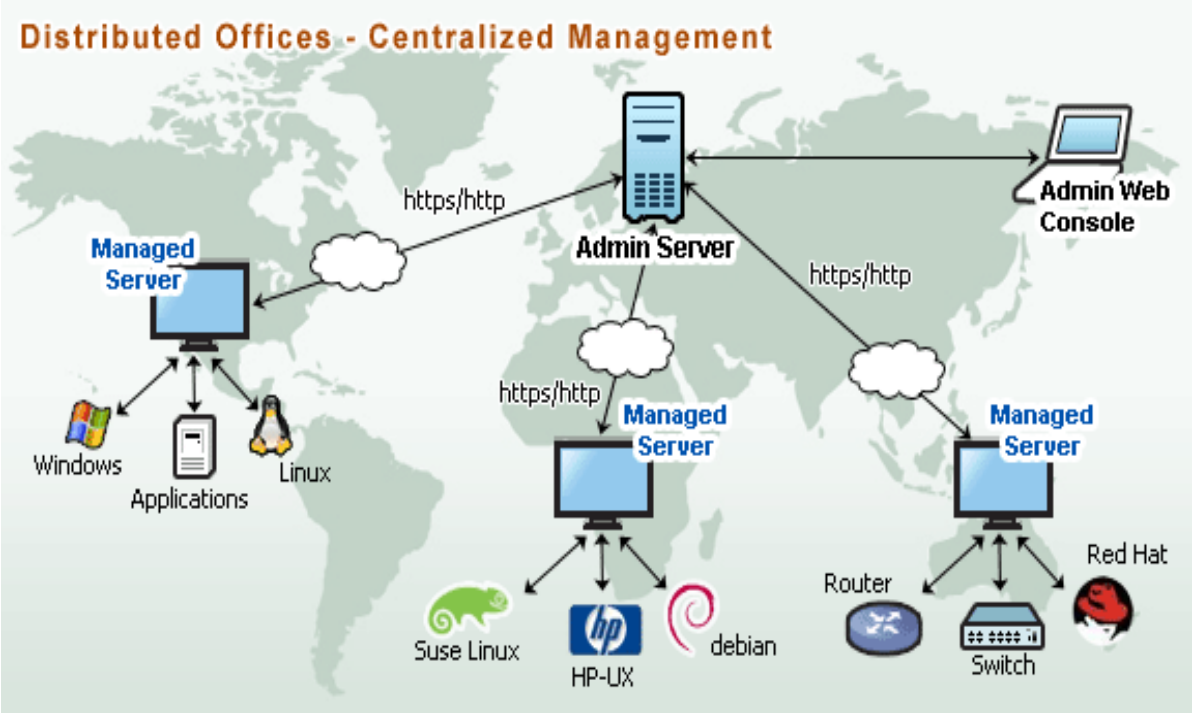
Sistemas Distribuídos

2013/2014

IdeaTrader

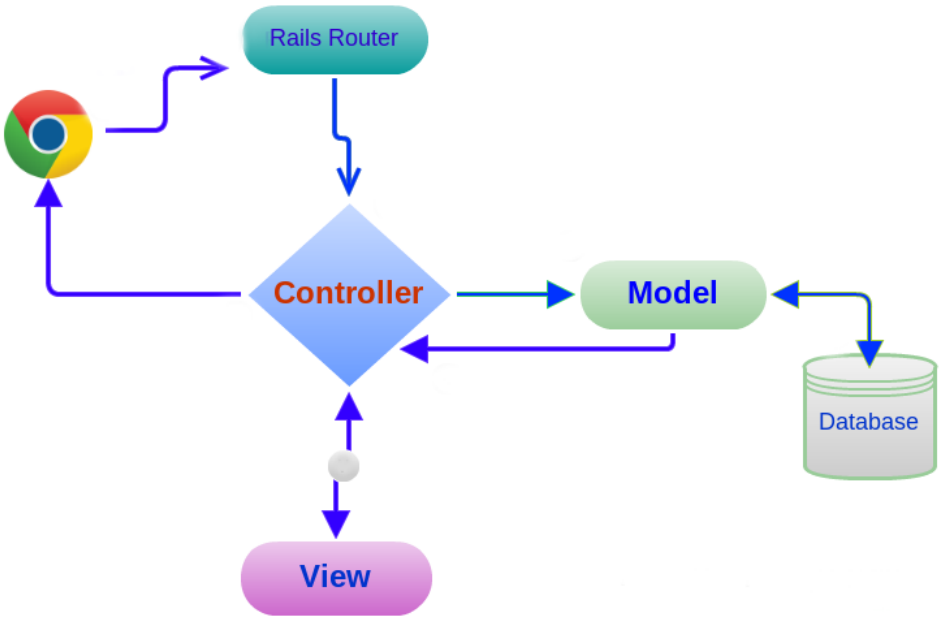


Iris Sofia Paquete nº2008117860

João Miguel Jesus nº2008111667

Introdução

No âmbito da cadeira de Sistemas Distribuídos foi-nos pedido para implementar uma rede social na qual denominámos de IdeaTrader, utilizando uma arquitetura MVC (Model-view-controller) ou seja, um modelo que separa a representação da informação e ainda da interação do usuário com este. Para tal, utilizamos na implementação para o Modelo, que consiste nos dados da aplicação um servidor RMI com um servidor DBS (Data Base Server) para que possa conter todos os dados de forma segura e de fácil acesso. Para o Controlador, que serve como mediador entre a vista apresentada ao utilizador e a ligação e a seleção de dados a apresentar que são provenientes do servidor RMI que faz *"queries"* ao DBS; em que utilizamos *"servlets"* e a *framework "Struts 2"*; e por fim a vista que consiste na informação apresentada ao utilizador, em que neste caso como é web utilizamos JSP que apresentam ao cliente, neste caso um *"browser"* a informação do sistema. Através desta arquitetura produzimos uma aplicação capaz de permitir aos utilizadores, através de uma conta própria, possam criar ideias, dar opiniões, comprar *"shares"* de ideais e vender as suas ideias se assim o desejarem, sem qualquer restrição em termos de acessibilidade.



Arquitetura Web

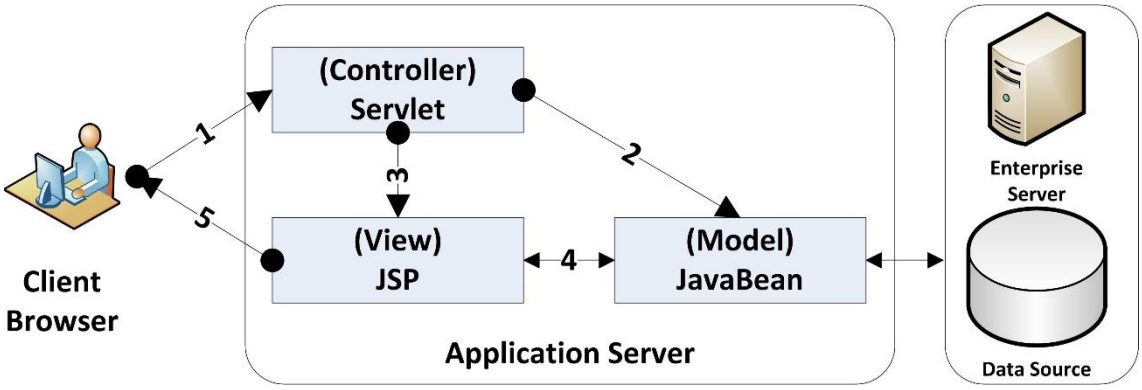
Para a arquitetura web do nosso projeto, seguimos o MVC, ou seja modelo, vista e controlador. Nesta arquitetura, podíamos escolher várias alternativas para cada um destas partes, pois desde que podem ser integráveis entre si, podia – se escolher a que mais apropriada para cada um de nós. Sendo assim seguindo este modelo, temos três partes responsáveis pela arquitetura Web:

- Modelo: consiste na parte lógica, das regras de negócio, as funções e o *"storage"* dos dados. Neste caso, para regras de negócio, as regras de lógica e as funções temos os "JavaBeans" que contem as classes com os campos necessários para receber e pedir informações ao "EIS" que é composto pelo Servidor RMI e o DBS. Estas informações, são depois manipuladas e apresentadas à interface da aplicação Web (Vista) através do controlador. Assim caso o utilizador queira fazer qualquer operação como por exemplo; o login, registar um utilizador, criar um tópico, criar uma ideia, comprar *"shares"* de ideias, comentar, entre outros; será a partir de chamadas feitas ao servidor RMI pelos "JavaBeans", em que o Servidor RMI, por sua vez liga – se à base de dados, e que depois escolhe os dados necessários para serem enviados para o modelo, este de seguida, apresenta essa informação para a vista, em que esta por sua vez é regulada pelo controlador, para que possa controlar a forma como estes são apresentados ao utilizador da aplicação. Sendo assim, como já foi dito anteriormente, utilizámos para "*storage*" dos dados um servidor de Base de Dados (DBS);

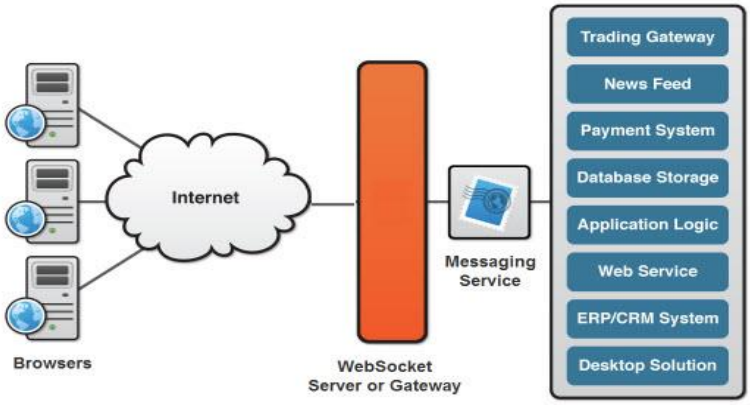
- Controlador: este, é responsável por receber os pedidos do utilizador através da vista e enviar os pedidos necessários ao "modelo", para que este envie os dados pedidos pelo utilizador. O controlador ainda é responsável por coordenar esses dados e decidir como passar – los à vista e alterar – la se necessário. Para o controlador, utilizámos a *"framework"* de *"Struts 2"*, pois esta evita que o código dos *"servlets"* se repita, ou seja elimina a redundância do código. Além disso, tem já funções próprias que permite a diminuição de trabalho do que se tivéssemos reescrever todas as funções necessárias de raiz e ainda estas tem um bom tempo de resposta, ou seja, performance.

- Visão: é responsável pela interface gráfica, a apresentação dos dados obtidos e manipulados pelo controlador e pelo *"layout"* apresentado ao utilizador da aplicação Web. Para apresentar a "visão", utilizámos JSPs, que são páginas web geradas dinamicamente baseadas em HTML,XML, ou outros tipos de documentos, que através dos *"JavaBeans"* previamente referidos no "Modelo" e da manipulação desses dados pelo controlador, possa apresentar ao utilizador as informações que pediu, através dos *"servlets"* de JSP.

Dentro ainda, da arquitetura MVC, escolhemos a *"Servlet – Centered"*, pois utilizámos um *"servlet"* como controlador, em que recebia pedidos do utilizador e instanciava os *"JavaBeans"* correspondentes, para que estes, possam através dos dados recebidos através da comunicação entre os *"JavaBeans"* e os "EIS"; apresentem de melhor forma ao utilizador, a partir dos JSPs, a resposta aos pedidos do cliente, que neste caso como é uma aplicação Web, um *"browser"*.



Outra tecnologia que utilizámos foram os **"WebSockets"** que permitem enviar informação em tempo real, como notificações aos utilizadores sobre notícias, transferências, mensagens, compras e vendas, entre outros… Assim sendo, criámos nos JSP, ligações WebSockets entre os "browsers" que acedem ao sistema e o sistema, para que possa fazer o "push" das notificações em tempo real.



Integração Entre Projetos

No primeiro projeto construímos uma plataforma de troca de ideias, seguindo uma arquitetura cliente – servidor. Para tal, utilizámos uma implementação de SOCKETS TCP/IP entre o servidor TCP e os clientes, que funcionava como "*middleman*" entre os clientes e o servidor RMI; um servidor RMI que servia como controlador de dados em que recebia pedidos vindo do servidor TCP através dos clientes dos utilizadores e a base de dados; e por fim a base de dados que guarda permanentemente todos os dados da plataforma para que estejam protegidos contra qualquer problema nos servidores anteriormente referidos. Com a implementação desta arquitetura conseguimos assegurar proteção contra falhas de servidores, falhas de rede, ou problemas no cliente e com isso tornámos esta aplicação mais estável aproveitando o nível mais elevado de RMI e os Sockets TCP/IP com servidores e clientes "multi-threads".

Para integrar – mos estes projetos, criámos uma aplicação Web com o mesmo objectivo do que a plataforma acima referida, mas com ligeiras diferenças e com uma arquitetura própria deste tipo de aplicações como a MVC (Model – View – Controller).Para tal, utilizámos uma "framework" denominada de "Struts 2", para forçar de certo modo a arquitetura MVC, através dos controladores já existentes na "framework"; para que seja mais fácil criar a aplicação Web com todas as operações que havia na plataforma realizada no projeto 1. No entanto, como foi explicado mais acima, na arquitetura utilizada na aplicação WEB, ou seja, o MVC, temos os "servlets" que contem os controladores do que aparece na "View" do utilizador, através das JSP e ao mesmo tempo instanciam as "JavaBeans" que contem os campos e as variáveis necessárias de cada operação, através do uso da tecnologia de RMI, os "JavaBeans" ativos irão comunicar com o servidor RMI, para obter os dados do sistema guardados na base de dados, que pertencem às duas Plataformas e assim mostrar no *"browser"* dos utilizadores, sendo essa informação regulada pela "servlet" que funciona como controiador. Assim sendo, teremos duas arquiteturas diferentes, com duas plataformas desenvolvidas, de forma diferentes, uma serve de aplicação cliente local que é necessário um cliente "especial" para ligar à plataforma e outra aplicação Web, que "roda" em qualquer *"browser"* e é portanto muito mais acessível aos utilizadores.

Portanto, teremos duas plataformas de acesso diferentes, e ao mesmo tempo convergentes para o mesmo objetivo, ou seja criar uma plataforma, de troca, compra e venda de ideias que promova a discussão e a organização de ideias, mas que partilhem os mesmo dados e por consequência façam parte de um sistema distribuído, ou seja,

através de várias tecnologias, conseguimos ligar – nos aos mesmos dados através de arquiteturas e plataformas diferentes mas que contém o mesmo objetivo de negócio.

Integração Web Sockets

No nosso projeto, integrámos Web Sockets na plataforma Web, para termos lidarmos com o problema das notificações, visto que estas têm de colecionar informação em tempo real, de outra forma não era possível garantir que notificávamos os clientes em tempo real se não utilizássemos Web Sockets para cada cliente ligado ao sistema. Assim sendo sempre que um "browser" se ligar ao sistema, irá correr código javascript para criar os canais de comunicação em "real – time" para conseguirmos informar o utilizador de novas notícias, atualizações e transações com sucesso feitas pelo cliente.

Integração Rest

Infelizmente não foi possível integrar a tempo, as operações de Rest associadas à Plataforma Web.

Manual de Utilizador

1. Correr Postgres DBS e criar Tabela Projecto_BD;
2. Correr Pgscript que contem todas as tabelas e funções necessárias para o funcionamento do Sistema;
3. Arrancar a “Database”;
4. Importar Projeto Joao para o Netbeans e adicionar o projecto Ideabroker como projecto;
5. Verificar as dependências para ver se não existem librarias quebradas ou em falta;
6. Correr o ficheiro RMI_Server no Projeto Joao;
7. Correr o ficheiro TCP_Server e Cliente no Projeto Joao;
8. Correr Projeto IdeaBroker para ter acesso à Plataforma Web;
9. Fazer “Login” e operações pretendidas nas duas plataformas;

Especificação de Testes Realizados

Funcionalidades	Descrição	Resultado Esperado	Realizado no Projeto
Registar	Register a new user on the system using the Web Platform	Criar um utilizador na base de dados através da Plataforma Web	Sim
Login do Utilizador de Forma Protegida	Login with user credentials in the Web platform	Conseguir entrar no sistema com um login único e protegido guardado na base de dados através da Plataforma Web	Sim
Listar Tópicos	Show Topics on the Web Platform to the User	Apresentar todos os tópicos criados e guardado na base de dados através da Plataforma Web	Sim
Criar Tópicos	Create a new Topics on the Web Platform	Criar novos tópicos pertencentes ao utilizador que os criou no sistema e guardar – los na base de dados através da Plataforma Web	Sim
Criar novas ideias	Create a new Idea on the Web Platform	Criar novas ideias pertencentes ao utilizador que os criou no sistema e guardar – los na base de dados através da Plataforma Web	Sim
Apagar uma Ideia	Delete an Idea from the list on the Web Platform	Apagar Ideias pertencentes ao utilizador que seja o dono actual dessa ideia no sistema e guardar – los na base de dados através da Plataforma Web	Sim

Procurar e Listar Ideias	List and have the possibility to search any idea on the Web Platform	Conseguir procurar uma ideia específica e apresentar todas as ideias disponíveis na Plataforma Web	Sim
Fixar preço de “shares” de uma ideia	Set the price of a share using the Web Platform	Conseguir fixar o preço de uma ideia específica pertence ao utilizador na Plataforma Web	Sim
Comprar “shares” de uma ideia existente	Buy a share from an idea using the Web Platform	Conseguir comprar “shares” de uma ideia específica na Plataforma Web	Não funciona totalmente
Root takes over ideas (hall of fame shows these ideas)		”Root” tem acesso a todas ideias	Não funciona totalmente
Listar histórico de Transacções	Show all transactions made by the user on the Web Platform	Apresentar todas as transações do utilizador e guardadas na base de dados através da Plataforma Web	Não funciona totalmente
Interoperacionabilidade entre Projetos	Any changes made on Web Platform using any type of operations should also change and appear on the Web Client	Qualquer Alteração feita em qualquer um dos projetos deve alterar/criar/modificar em ambas as plataformas	Sim
Arquitetura MVC relacionada com o JSP	No Java inside JSPs	Não ter código Java dentro das JSP	Sim
Arquitetura MVC relacionada com Java	No HTML inside Java classes	Não ter código HTML dentro das classes Java	Sim

Web Sockets: Notificações	Transaction Notifications should be pushed to the Browsers instantly	Notificações através da plataforma Web	Não funciona totalmente
Web Sockets: Transacções	Transactions originating in TCP clients are pushed to Browsers instantly	Notificações de Transações através da plataforma Web	Não
Web Sockets: Actualização de Informação	Up-to-date information regarding the last trade price per share	Notificações de nova Informação através da plataforma Web	Não funciona totalmente
Rest: Associação entre contas do Sistema e do Facebook	Associate a user account to a Facebook account		Não
Rest: Login com conta do Facebook	Login with facebook account instead of regular credentials		Não
Rest: “Post” de ideias criadas no sistema	New ideas are posted on Facebook		Não
Rest: Apagar “Posts” do Facebook de ideas que já não existem no sistema	Delete an idea from Facebook when it is deleted in IdeaBroker		Não
Rest: “Post” compras de “shares de Ideias no Facebook	Post a comment on Facebook when		Não

	shares are bought		

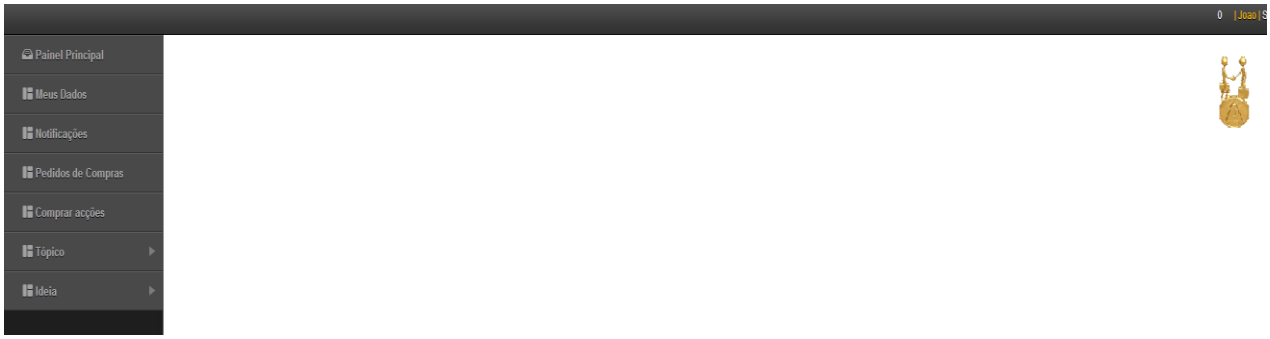
Anexos e Outras Informações

Plataforma Web:

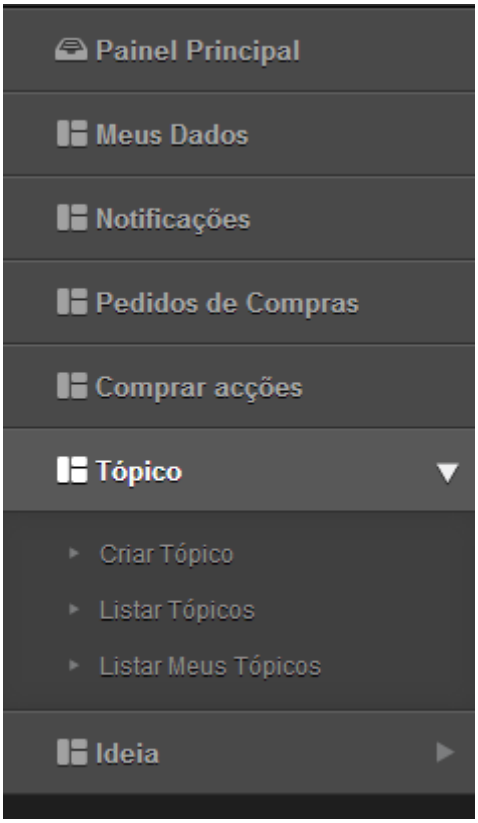
- Login

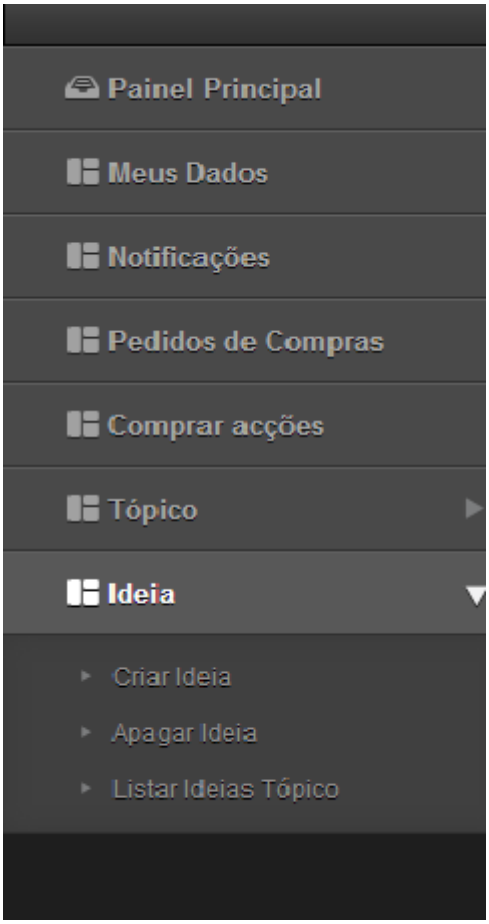
- Registar Utilizador

- Painel Utilizador

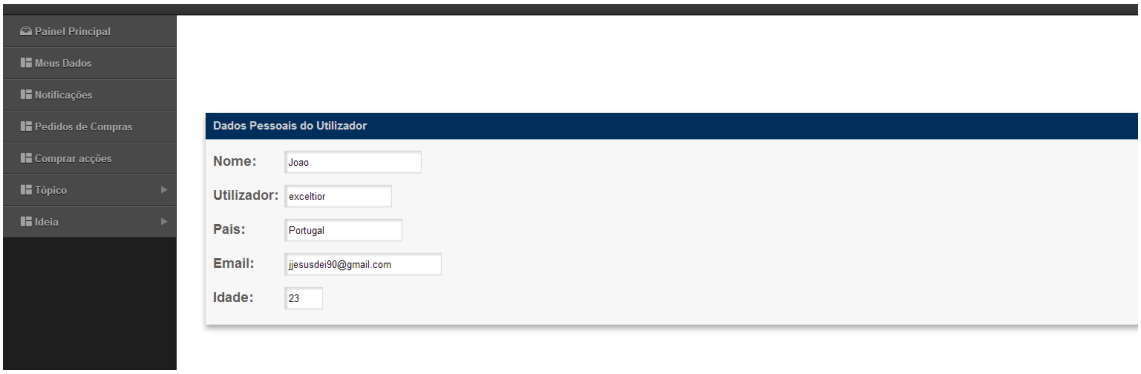


- Menu(s)

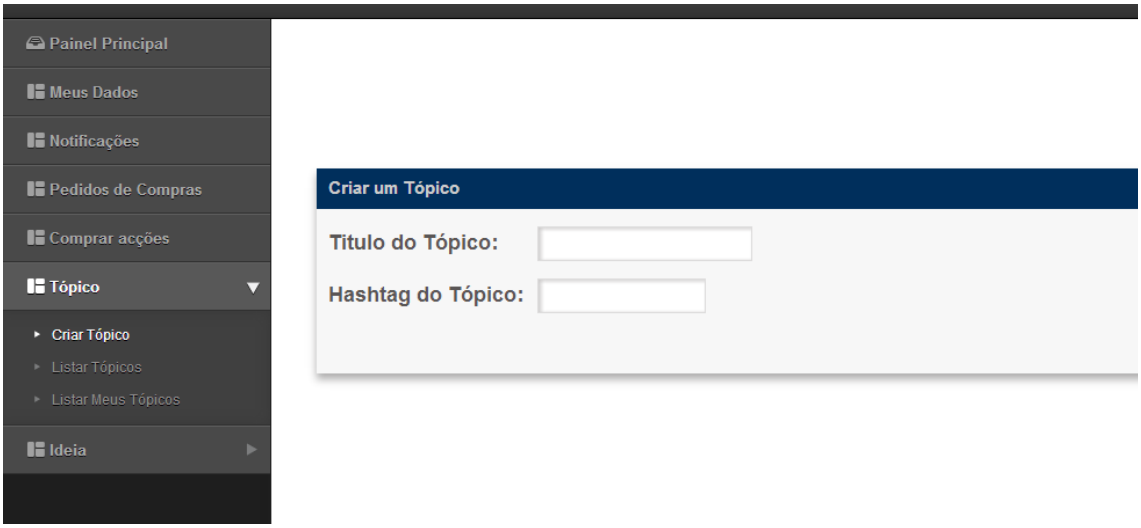

Painel Principal
Meus Dados
Notificações
Pedidos de Compras
Comprar acções
Tópico
Criar Tópico
Listar Tópicos
Listar Meus Tópicos
Ideia


Painel Principal
Meus Dados
Notificações
Pedidos de Compras
Comprar acções
Tópico
Ideia
Criar Ideia
Apagar Ideia
Listar Ideias Tópico

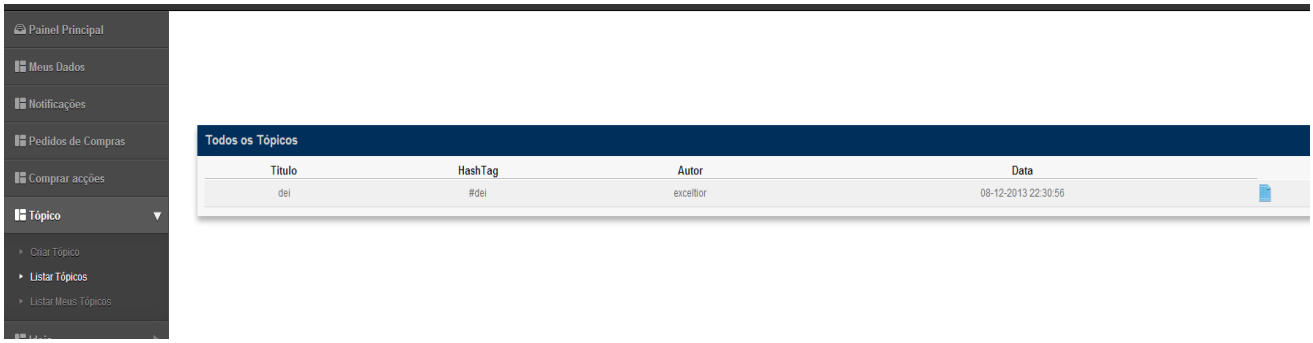
- Menu Dados


Painel Principal
Meus Dados
Notificações
Pedidos de Compras
Comprar acções
Tópico
Ideia
Dados Pessoais do Utilizador
Nome:
Joao
Utilizador:
excelsior
Pais:
Portugal
Email:
jjesusdei90@gmail.com
Idade:
23

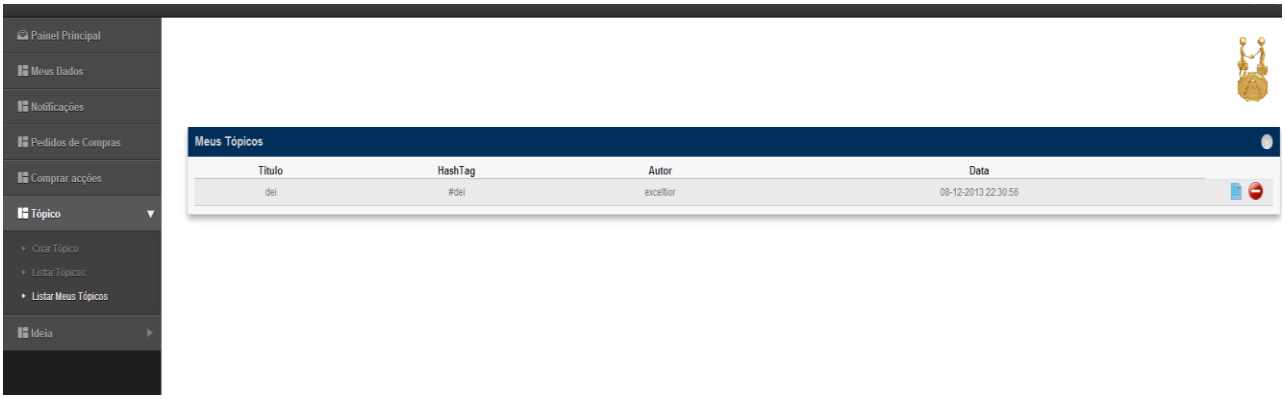
- Menu: Criar Tópico



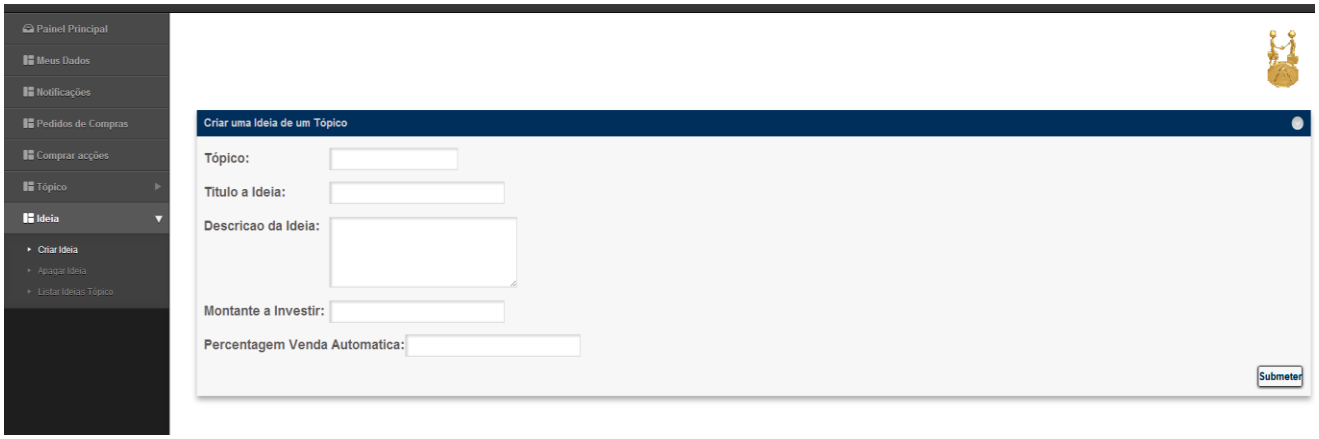
- Menu: Listar Tópico



- Menu: Criar Meus Tópicos



- Menu: Criar uma Ideia



- Menu: Apagar uma Idea