

Base de Dados

2013/2014

IdeaTrader



João Miguel Rodrigues Jesus nº2008111667
Iris Sousa nº2008117860

Introdução

No âmbito da disciplina de Bases de Dados foi-nos proposta a realização de um projecto académico, cujo objectivo principal residia na aplicação dos conceitos leccionados ao longo do semestre. Estes conceitos mostram-se especialmente importantes nesta “Era da Informação”, onde as bases de dados (BD) se tornaram praticamente indispensáveis e os sistemas de gestão de bases de dados extremamente populares.

O tema do nosso projecto foi pensado seguindo um modelo baseado num caso real como um “stock market” em que existe concorrência de acessos. O nosso trabalho consistiu na implementação de um sistema para o gerir, comprar, vender e criar ideias através de um sistema monetário chamado de “deicoins”.

Descrição breve do Projecto

O Projecto consiste numa plataforma web composta por um servidor RMI que trata da lógica de negócio, um servidor DBS da Oracle que trata de guardar e enviar dados ao servidor RMI e do cliente que envia e recebe informação acerca dos seus pedidos.

Esta plataforma consiste numa troca online de ideias criadas por utilizadores, organizada através de tópicos que também são criadas por utilizadores; em que estes podem comprar ações e vender ações pertencentes a ideias por DEICoins, que é a moeda utilizada nesta plataforma para efeitos económicos ; comentar as ideias mais interessantes; relacionar ideias; entre outros para que possam valorizar a sua opinião através destes métodos.

Problemas de concorrência

Em termos de concorrência temos que ter em conta que a aplicação suporta utilizadores em simultâneos, em que as suas ações sejam protegidas por “Commits” e “Rollbacks” para que em caso de algum erro em termos de servidores, bugs e outros a base de dados possa voltar atrás e assim possa manter a coerência dos dados ao longo do tempo. Também temos de ter em conta, que se vários utilizadores tentarem aceder ao mesmo campo ao mesmo tempo, temos de garantir que o primeiro que acede, tem a certeza que vai ter os dados guardados e prontos a fazer “Commit”, caso haja outro utilizador este espera que o outro acabe de fazer as alterações e só depois pode fazer as alterações que entender.

Divisão do Trabalho

João Jesus

- Servidor RMI : RMI Sever Java, JavaBeans, Network, Interface;

- Base de Dados: Triggers DBS, ER Diagram, Funções;

Iris Sousa

- Cliente: Interface, Funções, Ligação do MVC, Network;

- Base de Dados: Tabelas;

- Relatório;

Diagrama de Gantt



Horas Totais despendidas: 51 H

Core Technologies

Servidor DBS: Oracle Express 11;

PL: Java

Web:

- ViewModel: JSP, XML;
- Controller: Struts 2;
- Model : JavaBeans;

Especificação Técnica

Para melhor estruturar a aplicação e facilitar a divisão de tarefas pelo grupo, decidimos seguir um modelo de divisão por camadas, definindo regras explícitas acerca daquilo por que cada uma era responsável e do tipo de dados que circulavam nas classes dessa camada.

<i>Camadas do Sistema</i>
Camada da GUI
Camada de ligação aos objectos da GUI
Camada intermédia
Camada de chamadas à Base de Dados
Camada da Base de Dados

Camada da GUI : nesta camada encontra-se apenas o ambiente gráfico visível ao utilizador, para a aplicação cliente WEB ou seja, o posicionamento, aspecto e tipos de objectos (botões, caixas de texto, spinners, etc). Esta camada é apresentada através de JSPs que são controlados pelo Servlet, seguindo uma arquitectura MVC. Houve uma preocupação com a nomenclatura de cada objecto, de modo a que a programação das funções na camada seguinte – de ligação ao código propriamente dito - seja comprehensível.

Camada de ligação aos objectos da GUI : aqui define-se o conteúdo de cada objecto a apresentar na JSP, o que acontece quando há acções sobre eles (eventos), a que objectos se vão buscar os elementos inseridos pelo utilizador, entre outros. As classes JavaBeans que pertencem a esta camada passam os valores obtidos do utilizador à camada de código seguinte, que possui métodos para verificar e converter os dados recebidos. O formato de envio e recepção de dados de e para esta camada é feito utilizando Strings, arrays de Strings ou Vectors (classe Java) de arrays de Strings.

Camada intermédia de código : esta camada é a camada de validação e conversão dos dados. A sua função é receber os dados das classes de interface (nos vários formatos String referidos em cima), verificar erros neles, convertê-los (caso seja necessário), e chamar os métodos adequados na camada de baixo (que comunica

directamente com a BD). De seguida, deverá devolver à interface o resultado da chamada, num formato pré-acordado e seguindo ainda os tipos de objecto já referidos. Muitas vezes, o valor devolvido será directamente a mensagem a apresentar ao utilizador, de modo a não obrigar as JavaBeans a consultarem uma tabela de códigos de erro ou apanhar excepções. Nesta camada, a divisão em classes recaiu sobre a utilização de várias classes Interface, que lidam, cada uma delas, com um tipo de dados cruciais do Sistema (clientes, acções, transacções, etc), fazendo a comunicação de forma transparente para as restantes camadas.

Camada de chamadas à Base de Dados : o objectivo desta camada é disponibilizar os métodos que comunicam directamente com a base de dados, seja para adicionar, alterar, remover ou fazer selecções de dados (INSERT, UPDATE, DELETE, SELECT, etc). Os métodos devem receber os argumentos no formato já absolutamente correcto para a BD (o que implica que a camada superior terá de os verificar previamente, como já vimos) e devolvê-los tal como vêm da BD, nas mesmas variantes String mencionadas, principalmente os Vectors de arrays de Strings (uma linha do vector corresponde a uma linha de uma tabela; os arrays contêm os campos dos objectos).

Camada da Base de Dados : esta é a camada mais baixa da aplicação em que é necessário introduzir programação. Na camada, é necessária a definição prévia das tabelas e sequências numéricas através de SQL, bem como a criação de triggers, funções e procedimentos PL-SQL, tudo segundo a estrutura fornecida pelo gestor de bases de dados, PostgreSQL.

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:

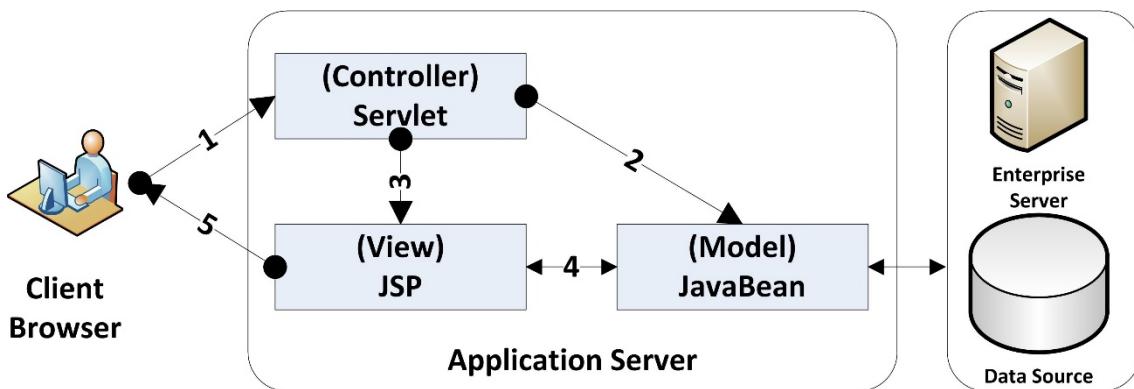
- Modelo: consiste na parte lógica, das regras de negócio, as funções e o “*storage*” dos dados. Neste caso, para regras de negócio, as regras de lógica e as funções temos os “JavaBeans” que contem as classes com os campos necessários para receber e pedir informações ao “EIS” que é composto pelo Servidor 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 instanciaava 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.

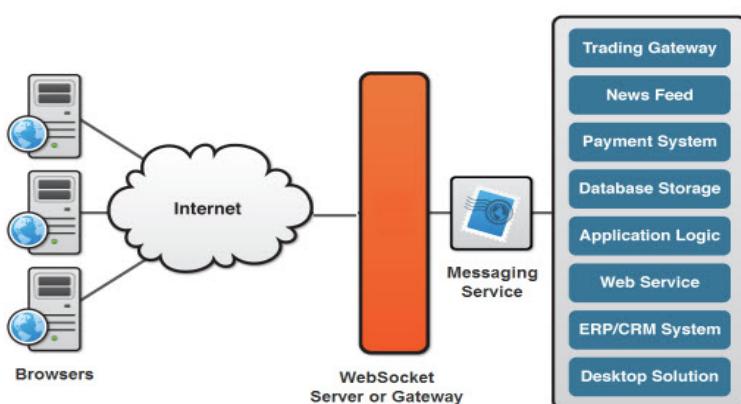
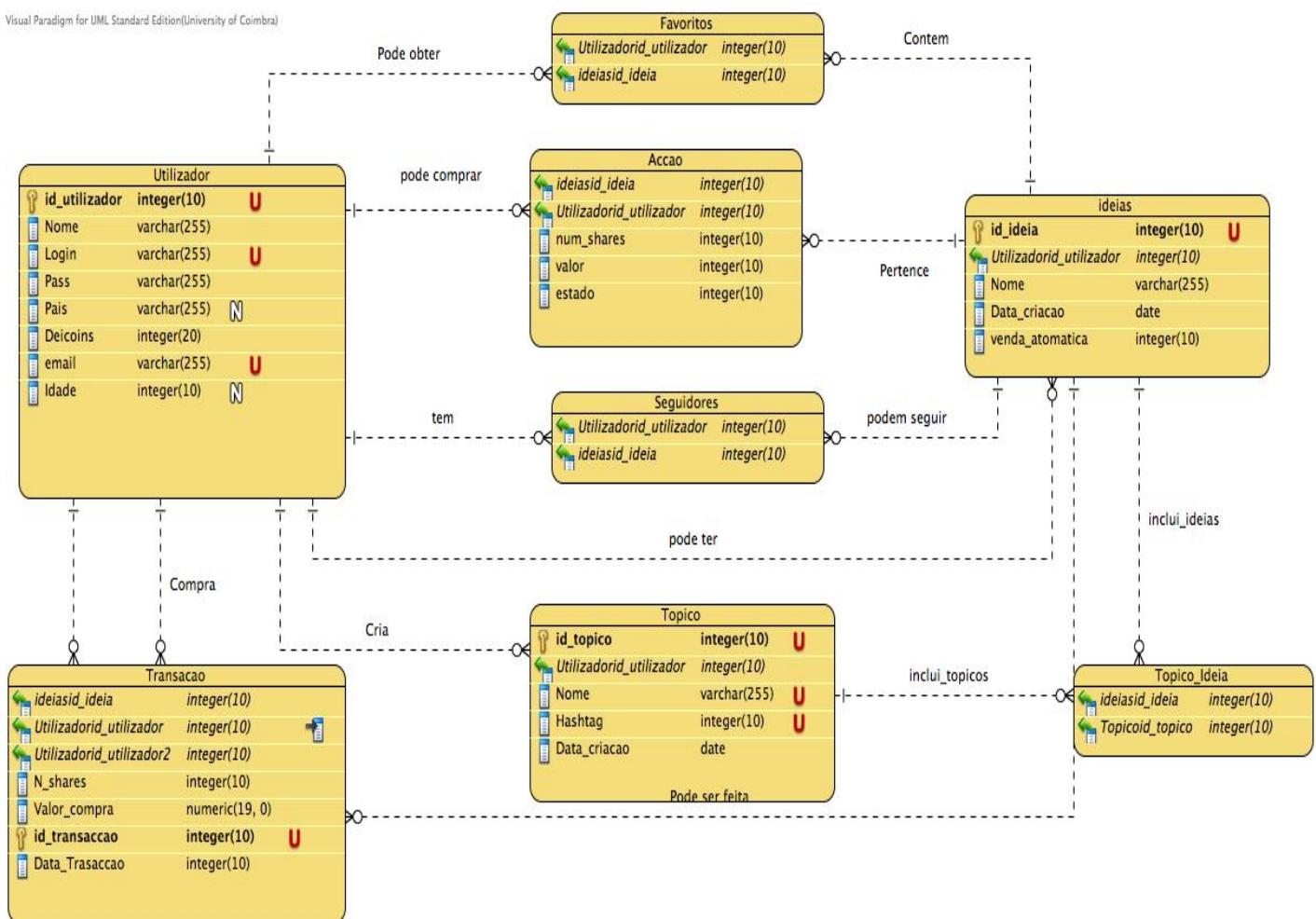


Diagrama ER

Visual Paradigm for UML Standard Edition(University of Coimbra)



Alterações em Relação à Meta 2

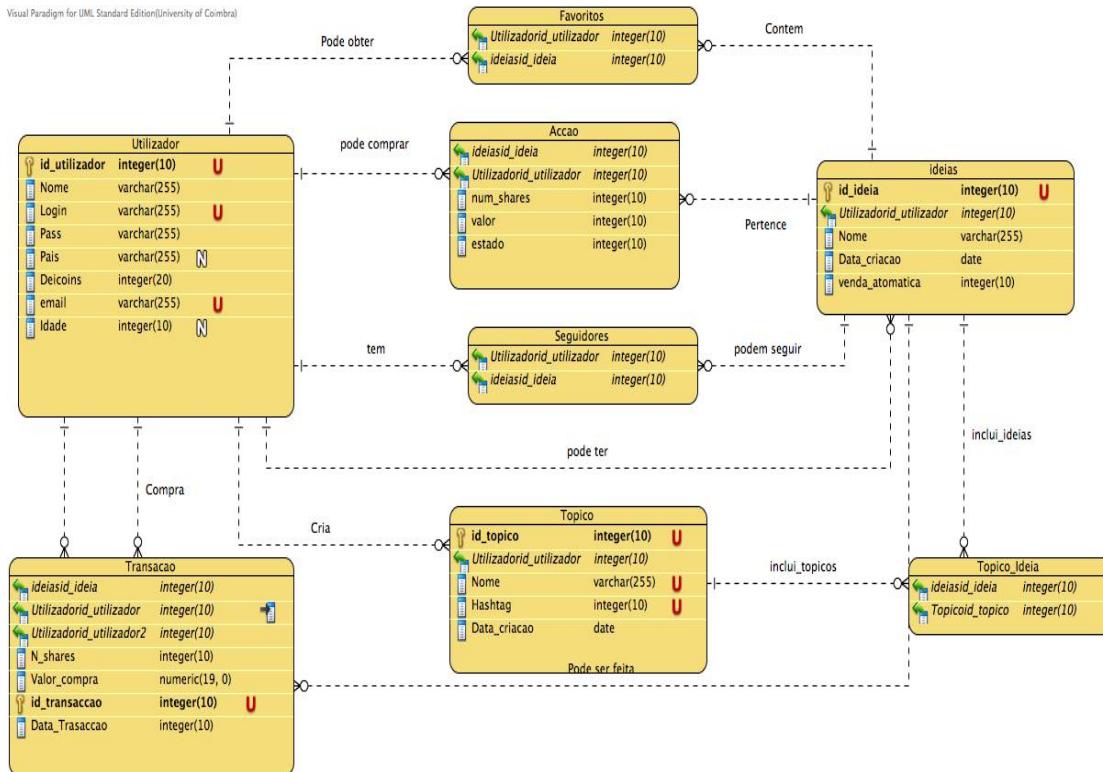
Em relação ao diagrama entregue na meta 2, houve a necessidade de efectuar algumas alterações, embora nenhuma destas possa ser considerada radical.

- Para começar, foi adicionada duas novas entidades: Seguidores e Favoritos. A primeira entidade permite aos utilizadores seguirem uma ideia de perto, como todas as actualizações, etc ... ; enquanto que a segunda permite aos utilizadores marcar uma ideia como favorito.
- Além destas alterações, conseguimos melhorar o diagrama, através da remoção de entidades e atributos redundantes. Assim, uma Transacao contem os registo de dois utilizadores, de quem vende e de quem compra, assim como a ideia a ser vendida e o número de shares que o comprador pretendia, daí a razão de utilizarmos duas ligações entre o Utilizador e Transacao.

ER Descrição

Através do diagrama, podemos fazer uma breve análise de como funciona a aplicação, como podemos ver, um utilizador pode criar uma ideia, tendo de obrigatoriamente de fornecer um nome e um preço para venda automática. Esta ideia pode ainda pertencer a zero, a um ou vários tópicos conforme o utilizador assim o deseje. O utilizador pode ainda criar um tópico para a sua ideia, ou para a de outros utilizadores, tendo de fornecer assim um nome, uma hashtag para que seja mais fácil de procurar. Como uma ideia pode pertencer a vários tópicos criámos a tabela tópico_ideia que guarda a correspondência entre uma ideia um tópico. Um utilizador pode ainda vender uma ideia através de shares, através de uma transacção onde indica qual é o número de shares , a ideia e o preço por unidade a qual deseja vender. Por sua vez, o utilizador que deseja comprar, mete no sistema a ideia que quer comprar, ao preço e à quantidade de shares que quer comprar. Ainda assim um utilizador pode seguir uma ideia ou marcar como favorito o que permite receber notícias, comentários ou outro tipo de informações acerca das ideias.

Diagrama Físico



O diagrama Físico é igual ao diagram ER pois não existem ligações do tipo MANY-TO-MANY.

- Índices

Os índices permitem aumentar a performance das bases de dados, permitindo uma pesquisa mais eficiente, sem ser necessário percorrer todos os registo quando efectuamos uma query. A utilização de índices é transparente e não muda a lógica inerente à base de dados.

Os custos de manutenção de índices são elevados quando temos tabelas muito dinâmicas, e isto pode fazer com que a performance da base de dados seja degradada, uma vez que o tempo de manutenção de dados pode aumentar mais do que o tempo médio ganho na pesquisa. Para além disso, os índices ocupam espaço em disco. Estes custos vão diminuindo quando temos muitos registo, pelo que a introdução de índices deve ser ponderada.

No nosso projecto a existência de índices assenta primariamente no facto de, ao se definir uma chave primária, esta ser automaticamente indexada. Também foram adicionados alguns índices que são frequentemente utilizados em pesquisas, como por exemplo o identificador do cliente na tabela .

- Procedures

Procedure é um bloco de PL-SQL residente na base de bados que pode ser executado. A utilização de procedures neste trabalho assentou sobre as estatísticas. São utilizados procedures que efectuam cálculos para extrair informações estatísticas relevantes das tabelas, inserindo-as numa tabela temporária auxiliar. Esses procedures têm uma versão que aceita duas data para restringir a janela temporal a considerar.

- Functions

Em termos de funções, a sua utilização foi elevada, estando limitada a funções simples como verificar se existe um certo user, se o login está correcto, adicionar clientes, ideias, acções,etc ... Para além disto, muitos métodos de verificação (retornando booleans) foram feitos na classe de ligação à base de dados, por forma a reutilizar código já feito nos métodos genéricos de select.

- Triggers

Um trigger é um bloco de código executado automaticamente quando ocorre um evento sobre a tabela a que está associado, nomeadamente um INSERT, UPDATE ou DELETE. Estes blocos de código têm como objectivo auxiliar na manutenção da integridade da base de dados, reforçando os mecanismos de segurança e/ou evitando operações inválidas.

No nosso projecto, os triggers foram utilizados essencialmente para a gestão da base de dados, alterando automaticamente os valores de exemplares disponíveis, aquando da realização de uma criação ou eliminação do sistema.

Protecções de integridade e controlo transaccional

A integridade da base de dados é mantida através de verificações no código Java, por triggers e também pelas restrições inseridas na criação das tabelas na base de dados.

O controlo de transacções foi necessário na criação de estatísticas da aplicação, uma vez que os resultados obtidos através dos procedimentos são guardados numa tabela temporária e é necessário definir os seus valores e lê-los de seguida, sem que algum outro utilizador lhe cause alterações no decorrer do procedimento. Para o efeito é utilizado um bloqueio na tabela “temp”, sendo que este bloqueio é levantado após a leitura das linhas recém-adicionadas, através de um comando “commit”. Isto porque desactivámos o auto-commit do Driver Postgres, já que este partiria uma transacção a meio.

Plano Final de Desenvolvimento de Código

O plano de desenvolvimento inicialmente proposto revelou que a nossa capacidade de estimar se revelou bastante optimista e não levou em conta factores externos que trouxeram alguns constrangimentos, nomeadamente a carga elevada de trabalhos para outras disciplinas. De qualquer forma, a equipa procurou jogar com os recursos e tempo que tinha à sua disposição, tentando rentabilizar ao máximo o esforço despendido. As alterações visíveis em relação ao plano de desenvolvimento de código original prendem-se essencialmente com datas, visto que o tempo que nos foi permitido utilizar não ultrapassou as duas semanas; já a sequência prevista de desenvolvimento manteve-se a definida, apenas com algum desenvolvimento paralelo de passos, devido à escassez de tempo.

Tamanho da Base de Dados

As contas que realizámos para esta base de dados foi supondo que pelo menos:

- Regista - se 1 cliente por dia = $365 * 3 = 1095$;
- Um Cliente faz 1 share por dia = 1095;
- 2 Transações por dia por cliente = $(2 * 365) * 3 = 2190$;
- 20 Ideias por 1 tópico : $1095 * 20 = 21\,900$;
- 1 Tópico por dia : 1095;
- 5 Seguidores por dia por ideia = $5 * 21900 * 365 = 39967500$
- 2 Favoritos por dia por ideia = $2 * 21900 * 365 = 15987000$

Tabelas:

-Utilizador:

- id_user : 4 bytes;
- Name : 62 bytes;
- País : 22 bytes;
- Data de Nascimento : 22 bytes;
- Login : 10 bytes;
- Password : 22 bytes;
- Data_Registo: 3 bytes;

Soma Total: 145 bytes * 1095 = 158775 bytes

- Tópico:

- id_tópico : 4 bytes;
- Name : 22 bytes;
- Data_Criação : 3 bytes;
- id_user : 4 bytes;

Soma Total: 33*1095 = 36135 bytes

- Ideia

- id_idea : 4 bytes;
- id_tópico : 4 bytes;
- id_user: 4 bytes;
- Proprietário : 4 bytes;
- Data_criação : 3 bytes;
- Venda_automática : 4 bytes;
- Preço unidade : 4 bytes;

Soma Total: 31 bytes * 21900 = 678900 bytes

- Accao

- id_share : 4 bytes;
- Vendedor : 4 bytes;
- id_ideia : 4 bytes;
- data_share : 3 bytes;
- n_accoes_venda : 4 bytes;
- preço_unidade : 4 bytes;

Soma Total: 27* 1095 = 29565 bytes

- Transacao

- id_transacção : 4 bytes;
- id_share : 4 bytes;
- comprador : 4 bytes;
- data_trasacao : 3 bytes;
- n_accoes_compradas : 4 bytes;

Soma Total 23* 2190 = 50370 bytes

- Topico_Idea

- id_idea : 4 bytes;
- id_user : 4 bytes;

Soma Total : $365 * (8 \text{ bytes} * 21900) = 63948000 \text{ bytes}$

- Seguidores

- id_idea : 4 bytes;
- id_user : 4 bytes;

Soma Total : $8 \text{ bytes} * 39967500 = 319740000 \text{ bytes}$

- Favoritos

- id_idea : 4 bytes;
- id_user : 4 bytes;

Soma Total : $8 \text{ bytes} * 15987000 = 127896000 \text{ bytes}$

Soma Final ao Fim de 1 Ano = 512896000 bytes

$$\begin{aligned} &= \\ &513 \text{ Mbs}; \end{aligned}$$

Conclusão

Este projecto foi uma forma de aprofundarmos os conhecimentos adquiridos nas aulas de Base de Dados e de utilizarmos igualmente conceitos e abordagens de outras unidades curriculares. Tudo isto será deveras importante para o nosso futuro enquanto engenheiros, já que será potencialmente muito difícil realizar um projecto comercial sem a utilização de uma base de dados.

Por outro lado, ficamos com a noção de que a qualidade do produto final e a sua usabilidade, em geral, são aspecto em que teremos de investir mais no futuro e que não são comparáveis aos obtidos com uma equipa maior, com indivíduos especializados. Apesar disto, tentámos organizarmo-nos de uma forma o mais rentável possível, sendo que, para isso, utilizámos uma ferramenta de controlo de versões (GIT) e estabelecemos a já descrita divisão de camadas na aplicação. Não temos dúvidas de que o tempo investido neste campo desenvolveu também a nossa capacidade crítica, deixando-nos um legado importante no âmbito da gestão de projectos.

Anexos e Outras Informações

Plataforma Web:

- Login



- Registar Utilizador

IdeaBroker - Registro

Campo Nome não pode ser nulo!

Utilizador:

Palavra-chave:

Nome:

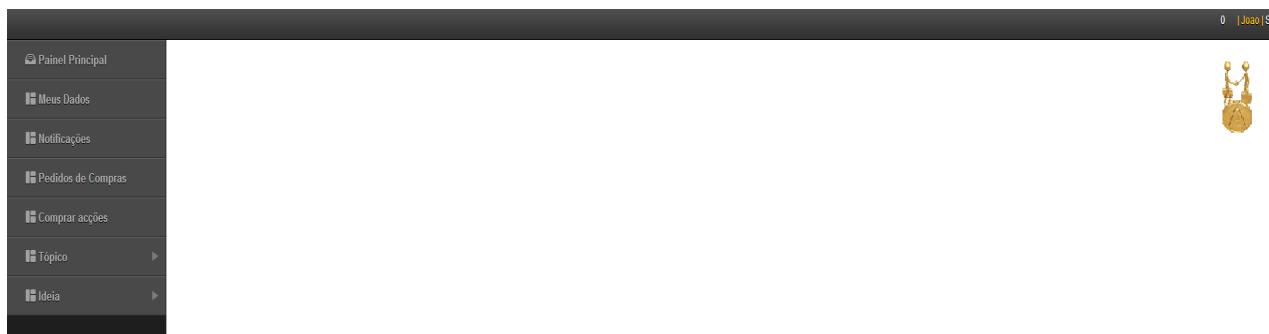
Idade:

Pais:

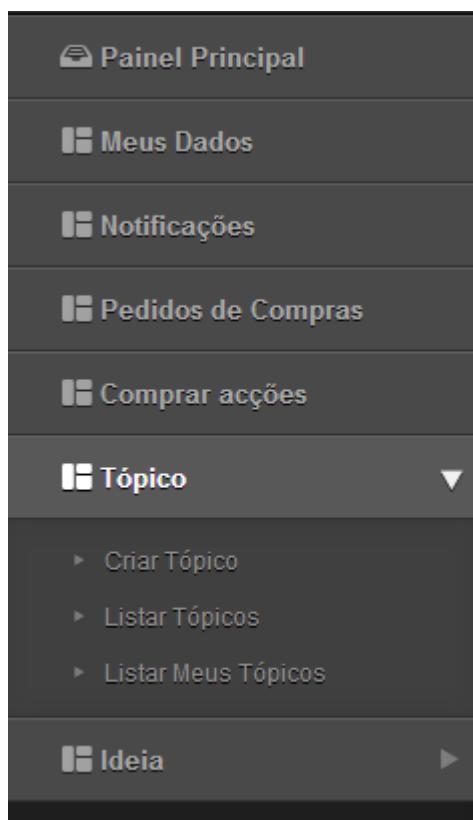
email:

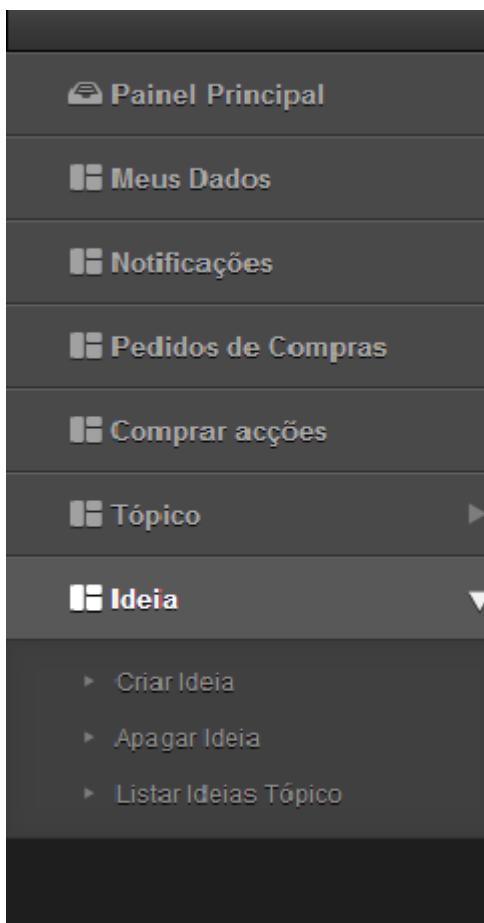
Cancelar **Submeter**

- Painel Utilizador

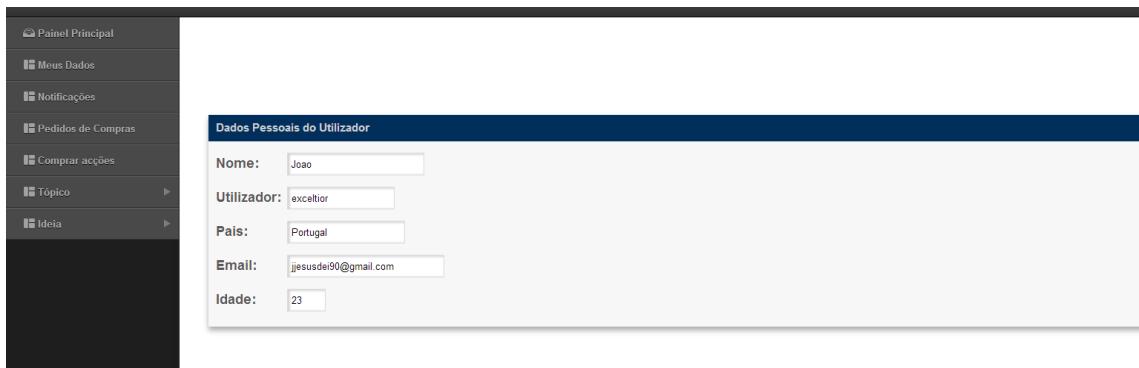


- Menu(s)





- Menu Dados

A screenshot of a user profile edit screen. The sidebar on the left is identical to the one in the previous image. The main area has a dark blue header bar with the text "Dados Pessoais do Utilizador". Below this, there are five input fields with labels and values: "Nome: Joao", "Utilizador: excetior", "País: Portugal", "Email: jesusdei90@gmail.com", and "Idade: 23".

- Menu: Criar Tópico

Criar um Tópico

Titluo do Tópico:

Hashtag do Tópico:

- Menu: Listar Tópico

Titluo	HashTag	Autor	Data
del	#del	excellor	08-12-2013 22:30:56

- Menu: Criar Meus Tópicos

The screenshot shows a user interface for managing topics. On the left is a vertical sidebar with icons for Painel Principal, Meus Dados, Notificações, Pedidos de Compras, Comprar ações, Tópico (selected), and Ideia. Under Tópico, there are links for Criar Tópico, Listar Tópicos, and Listar Meus Tópicos. Under Ideia, there are links for Criar Ideia, Apagar Ideia, and Listar Ideias Tópico. The main content area has a header 'Meus Tópicos'. Below it is a table with columns: Título, HashTag, Autor, and Data. A single row is shown with values: '#dei', '#dei', excellor, and 08-12-2013 22:30:56. There are edit and delete icons at the end of the row.

- Menu: Criar uma Ideia

The screenshot shows a form titled 'Criar uma Ideia de um Tópico'. It includes fields for 'Tópico:' (with a dropdown menu), 'Titluo a Ideia:' (text input), 'Descrição da Ideia:' (text area), 'Montante a Investir:' (text input), and 'Percentagem Venda Automatica:' (text input). A 'Submeter' button is located at the bottom right. The sidebar on the left is identical to the previous screenshot.

- Menu: Apagar uma Idea

The screenshot shows a form titled 'Apagar uma Ideia'. It has a single field 'Introduza a Referência da Ideia:' (text input) and a 'Apagar' button at the bottom right. The sidebar on the left is identical to the previous screenshots.

