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

Relatório de Preparação de Estágio
Ano lectivo 2007/2008

Rails Web Framework

Estudo da Framework e Desenvolvimento de Aplicações

Ricardo Horta e Vale Otero dos Santos

Supervisão na UC: Prof. Carlos Vaz
Supervisão na Mentes Virtuais: Eng. Marcos Garcia

Janeiro, 2008

Resumo

O presente relatório descreve o estudo efectuado durante a primeira fase do estágio de carácter profissional, visando a preparação do estágio curricular na área de desenvolvimento de aplicações web em Ruby on Rails que inclui o desenvolvimento de duas aplicações de teste de conhecimentos.

Conseguir apresentar soluções de negócio no menor tempo possível é, hoje, um factor de afirmação das entidades empresariais no mercado. As *frameworks* de desenvolvimento web são projectadas para facilitar o desenvolvimento de *software*, habilitando designers e programadores a despenderem mais tempo na determinação das exigências do *software* do que com detalhes de baixo nível do sistema. Assim cada vez mais empresas deixam os processos repetitivos e morosos de construção de aplicações web dando lugar ao uso de uma *framework* de desenvolvimento.

A *framework* Ruby on Rails é uma *framework Open Source* optimizada para o prazer de desenvolver aplicações web, aumentando a produtividade e eliminando os processos fastidiosos a que muitas vezes estão associadas.

Inicialmente foi feito um estudo acerca da *framework* bem como estudar a sua capacidade de adaptação ao método de desenvolvimento na empresa e aos projectos a desenvolver. Seguiu-se uma aplicação desses conhecimentos em duas aplicações simples. No final desta primeira fase obteve-se uma versão base para cada uma das aplicações, apenas com algumas funcionalidades principais e tendo em vista a exploração da *framework*, para desenvolvimento futuro.

Os objectivos desta primeira fase foram cumpridos com sucesso, tendo já sido proposto trabalho para a segunda fase do estágio, na mesma área e na sequência do trabalho realizado até à data.

Índice

Índice de figuras

Índice de tabelas

Glossário

Termo	Descrição
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
bug	erro ou falha de *software* que impede o seu correcto funcionamento
CMS	Content Managment System
CRUD	operações básicas sobre uma base de dados (*create*, *read*, *update* e *delete*)
deployment	disponibilização de uma aplicação para uso
DOM	*Document Objectc Mapping*
framework	estrutura de suporte ao desenvolvimento de um projecto de *software*
Hash	um conjunto de caracteres gerado a partir de dados existentes
HTML	HyperText Markup Language
HTTP	HyperText Trasfer Protocol
Java	linguagem de programação orientada a objectos
JavaScript	linguagem de *scripting* que permite adicionar funcionalidades a páginas HTML
J2EE	Java 2 Platform, Enterprise Edition
know-how	conhecimento adquirido acerca da forma de execução de uma tarefa
MVC	*design pattern* Model-View-Controller
Open Source	conjunto de princípios e práticas que promovem o acesso livre ao *design* da aplicação, permitindo a partilha de conhecimento sem restrições
ORM	Object Relational Mapping
overhead	utilização indirecta de recursos, em excesso, no cumprimento de uma tarefa
perl	linguagem de programação dinâmica
PHP	linguagem de programação concebida para produzir aplicações web dinâmicas
plugin	*software* que se integra com uma aplicação para uma determinada função
python	linguagem de programação de sintaxe minimalista
Rails	termo frequentemente utilizado para a *framework* Ruby on Rails
REST	Representational State Transfer

Termo	Descrição
RPC	Remote Procedure Call
script	software que controla uma aplicação estando dependente desta
SMS	Short Messaging Service
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVN	*subversion* – sistema de controlo de versões
software	conjunto de instruções computacionais que desempenham determinada tarefa
standard	prática recomendada no processo de desenvolvimento
template	padrão de desenvolvimento
web	termo frequentemente utilizado para World Wide Web
Web 2.0	tendência actual no web *design* e desenvolvimento, chamada a segunda geração das aplicações web
Web Service	sistema de software desenhado para suportar interoperabilidade através de uma rede
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

1. Introdução

O trabalho desenvolvido começou com a análise das tecnologias existentes no âmbito do desenvolvimento de aplicações web, com principal ênfase nas *frameworks* de desenvolvimento.

Efectuando-se um estudo relativamente detalhado da *framework* Ruby on Rails, tendo em vista a migração em curso na empresa, analisaram-se as vantagens e possíveis desvantagens do seu uso, bem como o estudo de outras *frameworks* e os pontos em comum ou possíveis divergências.

Por fim aplicaram-se os conhecimentos em duas aplicações, de características e requisitos distintos, tentando-se explorar ao máximo todos os aspectos da *framework*, desde as migrações à passagem da aplicação para um ambiente de produção.

1.1. Instituição

A associação de empresas para uma rede de inovação em Aveiro – INOVA-RIA, constituída a 29 de Julho de 2003, tem por objectivo a consolidação de um *cluster* de telecomunicações que contribua para o desenvolvimento da região de Aveiro. Pretende contribuir para a criação e sustentabilidade de emprego qualificado, fomentar a cooperação empresarial, nomeadamente nas áreas de investigação e desenvolvimento, formação, marketing e internacionalização, de forma a atrair novos investimentos para os seus associados.

Do conjunto de empresas é possível destacar associados como a PT Inovação, a grande impulsionadora desta associação empresarial. Existem ainda outros associados não menos importantes dos quais importa realçar a Mentes Virtuais, empresa onde será efectuado o estágio curricular.

A Mentes Virtuais é uma empresa de consultoria informática e soluções multimédia que existe desde Janeiro de 2003. Tem como principal objectivo o desenvolvimento de soluções que permitam satisfazer as necessidades específicas dos seus clientes.

1.2. Posicionamento

A Mentes Virtuais possui um vasto conhecimento na concepção de aplicações web e serviços móveis sustentado pelo *know-how* e experiência dos seus colaboradores. Disponibiliza ainda serviços nas áreas de multimédia, *design* e publicidade. Está por isso em posição privilegiada para garantir e proporcionar a todos os que procurem os seus serviços, inovação e valor acrescentado às suas instituições empresariais. Uma prova de tal facto é a confiança da PT Inovação na Mentes Virtuais.

1.3. Contextualização

O modelo tradicional cliente-servidor está, cada vez mais, a ser substituído pelas chamadas aplicações web. Estas aplicações geram, dinamicamente, uma série de documentos num formato standard (como HTML/XHTML) suportado por um comum *browser*. As aplicações web são populares não só pela ubiquidade do cliente mas também pela possibilidade de actualização e manutenção sem qualquer distribuição ou instalação de *software* potencialmente em centenas ou milhares de computadores.

A arquitectura base das aplicações web não é muito variável, e como tal é possível construir um suporte comum. Surgem assim as *frameworks* de desenvolvimento de aplicações web, construídas

para facilitar a modularidade e a não repetição de código e ferramentas adicionais, reduzindo o tempo de execução do trabalho, consequentemente os custos de produção, aumentando a rentabilidade da empresa.

Uma *framework* é uma estrutura de suporte definida em que outro projecto de software pode ser organizado e desenvolvido. Pode incluir programas de suporte, bibliotecas de código, linguagens de *script* e outros programas para ajudar a desenvolver e juntar diferentes componentes de um projecto de software. As *frameworks* são projectadas com o intuito de facilitar o desenvolvimento de *software*, habilitando *designers* e programadores a despenderem mais tempo com as exigências do *software* do que com detalhes tediosos de baixo nível do sistema.

1.4. Motivação e objectivos

Nos últimos anos têm aparecido inúmeras *frameworks* de desenvolvimento de aplicações web, nas mais diversas linguagens (Perl, PHP, Python, Ruby, Java, etc.), constituindo um grande leque de *frameworks* bastante completas e ao mesmo tempo com facilidades de utilização.

A Mentes Virtuais está neste momento a migrar o modo de construção das suas aplicações web, usando a *framework* Ruby on Rails, pela simplicidade de uso e a forma prática de desenvolvimento. Na empresa foram já desenvolvidas algumas aplicações em Rails, nomeadamente para integração em plataformas web desenvolvidas pela PT Inovação tendo-se obtido resultados bastante satisfatórios. Face à grande concorrência existente na área de aplicações web, é crucial, e um factor de diferenciação, conseguir produtos finais o melhor e mais rapidamente possível. Com a adopção do desenvolvimento através de uma *framework* pretende-se fazer face à competitividade crescente.

Neste contexto, o objectivo deste trabalho será não só estudar a viabilidade desta migração mas também analisar objectivamente os motivos da escolha. Pretende-se ao mesmo tempo fazer um estudo sobre a *framework* em questão e suas potencialidades, bem como desenvolver pequenas aplicações ou possíveis *plugins* a integrar em aplicações futuras ou existentes, usando a *framework* Ruby on Rails, de modo a ganhar algum *know-how* para desenvolvimento de aplicações com esta ferramenta.

2. Estudo da *framework* Ruby on Rails

Nesta secção apresentam-se as características da *framework* Ruby on Rails que levaram à sua adopção por parte da empresa em que o estágio foi realizado, depois de feito um estudo sobre o estado da arte. Está também incluído um pequeno comparativo com outras *frameworks*.

2.1. Origem e filosofias

Ruby on Rails (Rails) é uma *framework* desenvolvida em Ruby – uma linguagem de programação não compilada, orientada a objectos, com uma sintaxe limpa – com vista à implementação de aplicações web com recurso a uma base de dados. Foi inventada por David Heinemeier Hansson (1979 – Copenhaga), programador dinamarquês, tendo sido utilizada numa aplicação que estava a desenvolver – Basecamp (www.basecamphq.com) – e disponível ao público em Julho de 2004.

Dando prioridade ao modo simples de fazer as coisas e tirando partido da linguagem Ruby permite uma alta produtividade no desenvolvimento de aplicações web. Meses após o lançamento oficial esta *framework* passou de desconhecida a uma das *frameworks* de escolha para implementação de uma variedade das chamadas aplicações Web 2.0, não sendo apenas uma moda junto dos mais entusiastas mas também adoptada por multinacionais para desenvolvimento de aplicações web robustas.

Uma aplicação em Rails usa convenções simples de programação que permitem um mínimo de configuração e segue as seguintes filosofias chave:

- ***Don't repeat yourself (DRY)***: reduzir duplicações, particularmente quando se trata de programação. A informação está localizada num único sítio não ambíguo.
- ***Convention over configuration (CoC)***: significa que apenas temos de especificar aspectos não convencionais da nossa aplicação.

2.2. Model-View-Controller

Em aplicações complexas, que lidam com um volume elevado de dados, pretende-se, normalmente, separar os dados e a interface gráfica para que as alterações à interface não afectem os dados e estes possam ser reorganizados sem interferências com a primeira.

A arquitectura MVC (figura 1) resolve este problema separando o acesso da apresentação dos dados bem como a interacção com o utilizador, introduzindo uma componente intermédia: o controlador (*controller*). Com a decomposição em *models* (dados) e *views* (interface), a arquitectura MVC ajuda a reduzir a complexidade no *design* da arquitectura das aplicações e aumenta a flexibilidade e reutilização.

As componentes da arquitectura MVC são então os seguintes:

- ***Model***: representação da informação, dependente do domínio sobre a qual a aplicação opera. Numa arquitectura MVC o método de armazenamento e de acesso aos dados não é especificamente mencionado uma vez que está num nível inferior e encapsulado pelo Modelo.
- ***View***: transforma o modelo numa forma passível de interacção, tipicamente um elemento de interface com o utilizador. Podem existir várias *views* para o mesmo modelo com propósitos distintos.

- ***Controller***: processa e responde a eventos, tipicamente acções do utilizador, e pode invocar mudanças no modelo.

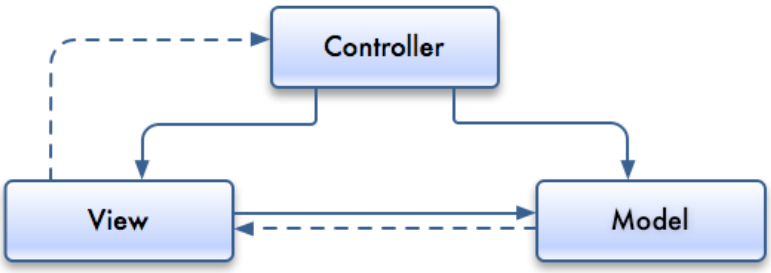


Figura 1 *– Arquitectura MVC (relações directas a cheio e indirectas a tracejado).*

O MVC é muitas vezes visto aplicado a aplicações web, onde as *views* são o código HTML da página actual, o *controller* é o código que recolhe dados dinâmicos e gera o conteúdo do HTML e o *model* é representado pelo conteúdo, normalmente guardado numa base de dados ou ficheiros XML.

Na figura 2 está esquematizada a implementação MVC na *framework* Ruby on Rails. O Ruby on Rails implementa este tipo de arquitectura onde o *model* fica ao cargo de uma camada de ORM chamada *Active Record*. O *Active Record* permite definir relações entre diferentes tabelas de forma intuitiva (ex: *has_many*, *belongs_to*), e faz validações nos modelos.

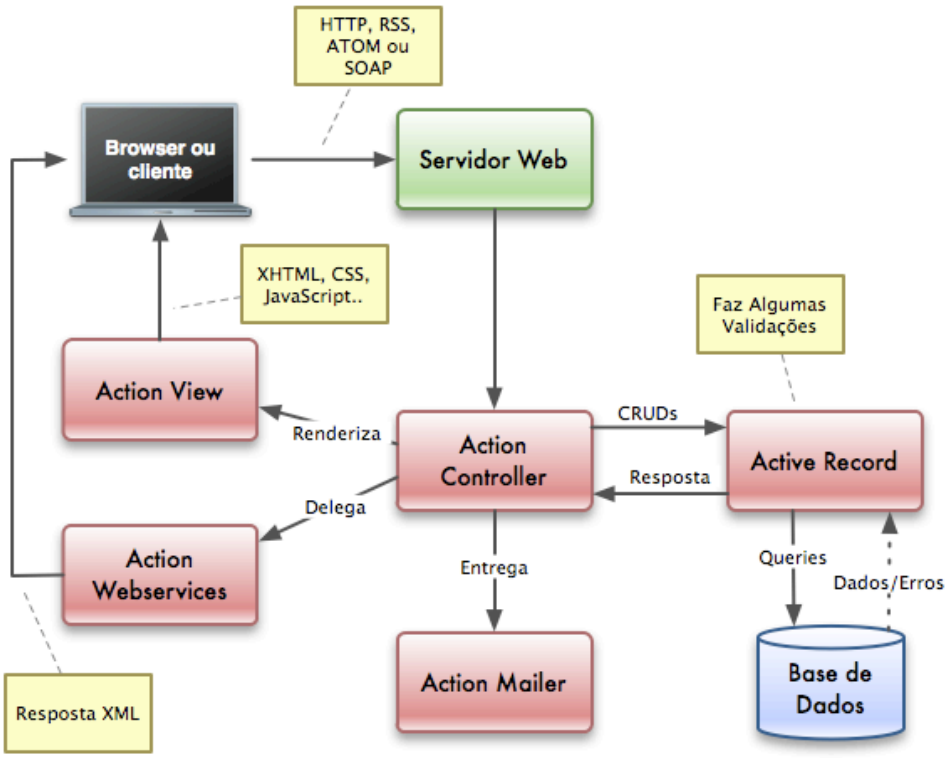


Figura 2 *– Arquitectura MVC na framework Ruby on Rails.*

Esta camada permite a apresentação das linhas de uma base de dados como objectos e enriquece estes objectos com métodos da lógica de negócio. O *controller* e a *view* são manipulados pelo *Action Pack*, que é constituído por duas partes diferentes: *Action View* e *Action Controller*.

O *Action Controller* recebe os pedidos do servidor web, faz os pedidos necessários ao *Active Record* e devolve os resultados. É também responsável pelas sessões e *cookies*, cria todos os URL's da aplicação automaticamente consoante o nome dos controladores e as suas acções (geralmente no formato *http://endereço/controlador/acção/parametros*).

O *Active View* devolve o resultado do pedido numa *view* em XHTML, XML, imagens, JavaScript, etc. Normalmente cada acção de um controlador tem uma *view* e cada controlador tem uma *view* especial chamada *layout* que se renderiza sempre, independentemente da acção que é chamada.

Pode ainda ver-se na figura 2 que existem outras duas componentes diferentes – Action Webservices e Action Mailer. A primeira implementa o suporte para web services com protocolos SOAP e XML-RPC e permite declarar e publicar API's de modo simples. O *Action Mailer* permite interagir directamente com utilizadores ou outros sistemas por email.

2.3. Desenvolvimento ágil

A utilização de Ruby on Rails está frequentemente associada à utilização de um desenvolvimento ágil de software. O desenvolvimento ágil, tal como qualquer metodologia de desenvolvimento de software, providencia uma estrutura conceptual para reger projectos de engenharia de software e segue os seguintes princípios:

- garantir a satisfação do cliente entregando rápida e continuamente versões funcionais;
- versões funcionais são entregues frequentemente (em semanas) e são a principal medida de progresso do projecto;
- mesmo mudanças tardias dos requisitos do projecto são bem-vindas;
- cooperação constante entre clientes ou utilizadores e programadores;
- deve haver um cuidado contínuo com a excelência técnica e bom *design* dos projectos;
- simplicidade;
- rápida adaptação a mudanças;
- equipas 'auto-organizadas'.

Os métodos ágeis buscam a adaptação rápida a mudanças da realidade. Quando uma necessidade de um projecto muda, uma equipa adaptativa mudará também, em contraste com uma equipa preditiva que coloca o planeamento do futuro em detalhe. Assim, o desenvolvimento ágil tem pouco em comum com o tradicional modelo em cascata, uma das metodologias com mais ênfase no planeamento, onde o progresso é geralmente medido em termos de entrega de artefactos - especificação de requisitos, documentos de projecto, planos de testes, revisões de código e outros.

O modelo em cascata resulta de uma substancial integração e esforço de teste para alcançar o fim do ciclo de vida, um período que tipicamente se estende por vários meses ou anos. Os métodos ágeis, pelo contrário, produzem um desenvolvimento completo e teste de funcionalidades num período de poucas semanas ou meses. Algumas equipas ágeis usam o modelo em cascata em pequena escala, repetindo o ciclo de cascata inteiro em cada iteração.

A maioria dos métodos ágeis partilha a ênfase no desenvolvimento iterativo e incremental para a construção de versões implantadas do software em curtos períodos de tempo mas os períodos de tempo são medidos em semanas e não em meses, e a realização é efectuada de uma maneira altamente colaborativa.

Uma metodologia ágil bastante comum é a *Extreme Programming* (XP) que se rege por 5 princípios básicos:

- **comunicação** – na construção de software é necessário haver colaboração entre os utilizadores e/ou clientes com os programadores, muitas vezes por comunicação verbal.
- **simplicidade** – começa-se com a solução mais simples para o problema adicionando-se, posteriormente, novas funcionalidades. Foca-se assim no desenvolvimento para hoje e não para amanhã, daqui a uma semana ou um mês. A simplicidade também facilita a comunicação.
- **feedback** – tem de haver não só um feedback periódico da parte do cliente mas também da equipa de desenvolvimento, respondendo com estimativas de tempo quando o cliente surge com um novo requisito.
- **perseverança** – permite que o programador se sinta confortável com a restruturação do que fez até ao momento tendo que apagar algum código obsoleto independentemente do trabalho que deu a escreve-lo.
- **respeito** – os membros da equipa devem respeitar-se mutuamente não devendo proceder a alterações que interfiram de algum modo com o trabalho dos colegas.

2.4. Scaffolds

Um *scaffold* em Rails é uma estrutura gerada automaticamente para manipulação de um modelo. Dando uso a todas as convenções presentes na *framework* pode desta forma criar-se interfaces básicas de administração utilizando apenas um comando. Pela análise da base de dados são construídas interfaces para edição, adição ou remoção de um objecto pertencente ao modelo correspondente.

Os *scaffolds* não devem ser usados para criar aplicações, uma vez que, normalmente, as aplicações necessitam de uma interface mais optimizada e muitas vezes com particularidades específicas, mas pode ser usado como ponto de partida, sobre o qual se pode construir a aplicação.

2.5. Gems e Plugins

Uma *gem* é uma aplicação em Ruby ou uma biblioteca particular, desenvolvida por outra pessoa e disponível num repositório *on-line*. Usando uma única linha de comandos pode-se instalar, remover ou actualizar as aplicações/bibliotecas instaladas facilmente, tendo-se desta forma total controlo sobre a instalação actual da plataforma Ruby.

Os *plugins* em Rails são uma extensão ou modificação da *framework*, com código desenvolvido por terceiros. A utilização de *plugins* nas aplicações é bastante comum, não só por ser extremamente fácil de as integrar mas porque normalmente são suficientemente flexíveis e evitam o desenvolvimento de determinadas funcionalidades de raiz.

2.6. Templates e Helpers

Em Ruby on Rails, quando criamos uma *view* estamos a criar um *template* – algo que vai ser tratado para gerar um resultado final. Os *templates* contêm assim uma mistura de texto e código, usado para permitir conteúdo dinâmico. Este código embutido permite ter-se acesso à informação constante num controlador. A *framework* vem com suporte para três diferentes tipos de *template*:

- RXML – *templates* usados para construir respostas em XML;
- RHTML – *templates* constituídos por uma mistura de HTML e Ruby embutido, usados tipicamente para gerar páginas HTML;
- RJS – *templates* que geram código JavaScript executado no *browser* e são geralmente usados para interacção com AJAX.

Os *helpers* são módulos que fornecem métodos automaticamente disponíveis para utilização nos *templates* e que geram HTML (ou XML, ou JavaScript) extendendo assim o comportamento de um *template*. Desta maneira simplificam-se as *views* mantendo-se blocos de programação em Ruby em ficheiros separados. Ao mesmo tempo evitam-se duplicações em diferentes *templates* uma vez que o código dos *helpers* pode ser reutilizado.

A *framework* vem com uma grande colecção de *helpers* e podem ainda ser complementados com *helpers* personalizados para cada aplicação. A vasta diversidade de *helpers* usados em Ruby on Rails é de tal forma útil que foram implementados *métodos* semelhantes em Python para uso nas diferentes *frameworks* desenvolvidas nesta linguagem [11].

2.7. AJAX

AJAX (*Asynchronous JavaScript And XML*) não é uma tecnologia mas um conjunto de várias tecnologias usadas em conjunto permitindo novas funcionalidades. AJAX incorpora:

- presentação baseada em standards usando XHTML e CSS;
- apresentação dinâmica e interacção usando DOM (*Document Object Model*);
- interacção e manipulação de dados usando XML e XSLT;
- pedidos assíncronos de dados (usando *XMLHttpRequest*);
- JavaScript ligando todos os pontos anteriores.

O *XMLHttpRequest* é um objecto JavaScript criado pela Microsoft, fazendo hoje parte da especificação DOM e torna possível a comunicação assíncrona com o servidor. Na realidade nem sequer é necessário usar XML e a parte verdadeiramente importante é que todos os pedidos são assíncronos.

O Ruby on Rails usa uma biblioteca JavaScript – Prototype [17] que permite abstracção completa do *browser* cliente e ao mesmo tempo simplifica todo um conjunto de operações comuns como validações e manipulação de objectos DOM. Com Prototype pode também usar-se pedidos AJAX de forma simples.

Para efeitos de interacção dinâmicos a *framework* vem também com outra biblioteca específica para esse fim – Script.aculo.us [18] que não é mais que um conjunto de objectos JavaScript implementados sobre Prototype, podendo por isso aproveitar todas as potencialidades de AJAX.

Todas as funcionalidades de Prototype, bem como os efeitos disponibilizados pelo Script.aculo.us estão disponíveis directamente no Ruby on Rails, sem haver necessidade de escrever código JavaScript, usando-se templates RJS como foi referido na secção anterior.

2.8. Migrações

A *framework* Ruby on Rails tem um sistema de migrações, que ajuda a manter os dados sincronizados com o código actual, funcionando como um sistema de controlo de versões dos dados, não só entre diferentes pessoas numa equipa mas também quando a aplicação é transferida para o servidor em produção. Acrescentando ficheiros com as alterações desejadas e correndo um único comando assegura-se que a estrutura de dados utilizada é da versão desejada.

Tal como acontece com JavaScript, no desenvolvimento de uma aplicação em Rails raramente será necessário escrever SQL. Isto aplica-se inclusive à definição e alteração da estrutura de dados através do sistema de migrações. Mantendo a definição da estrutura de dados em ficheiros Ruby, sem recurso a SQL, assegura-se que esta é agnóstica aumentando a portabilidade.

2.9. Integração com outras tecnologias

O *Action Webservices* permite o suporte de protocolos SOAP e XML-RPC nas aplicações Rails. Converte as invocações remotas de métodos em chamadas a métodos no nosso web service e trata da resposta. Assim, deixa-nos concentrar no trabalho de escrever os métodos específicos da nossa aplicação para servir os pedidos.

Através do *Action Webservices* podemos assim não só permitir definir uma API da nossa aplicação para ser acedida por outras aplicações em qualquer linguagem mas também adicionar suporte para funcionalidades disponibilizadas por outras aplicações.

Além de suporte para protocolos SOAP e XML-RPC, a *framework* inclui suporte para REST. O termo "REST" foi introduzido por um dos principais autores do protocolo HTTP no ano 2000 e tem vindo a ser usado desde então. Este nome refere-se a um conjunto de princípios de arquitectura de redes que definem como é que os recursos são definidos e endereçados, mas é muitas vezes usado num sentido mais lato para descrever qualquer interface simples que transmite dados específicos de uma aplicação através de HTTP sem uma camada adicional como SOAP ou HTTP *cookies* onde os verbos usado no protocolo HTTP (*GET*, *POST*, *PUT*, *DELETE*) devem ser usados com um significado. A diferença entre o uso do termo "REST" tem causado alguma confusão em discussões técnicas.

Além da facilidade de uso de web services Ruby on Rails tem um sistema de *plugins* de fácil instalação e que conta já com um repositório bastante grande de *plugins* na Internet, alguns dos quais permitem interagir directamente com outras aplicações web, (como o *flickr.com* ou o *twitter.com*).

2.10.Servidores HTTP

Para testar e correr uma aplicação em Rails é necessária a familiarização com um conjunto de possíveis servidores HTTP:

- ***WEBrick*** – para simplificar o desenvolvimento e teste rápido, quando se cria uma nova aplicação tem-se ao dispor este servidor HTTP, simples e leve. Não é, no entanto, aconselhável o seu uso em aplicações em produção, por não ser eficiente nem seguro.

- ***Mongrel*** – servidor HTTP com suporte para Ruby on Rails desenhado para ser rápido, leve e muito seguro.
- ***Apache*** – apesar do Mongrel ser excelente como servidor para Rails, quando se fala em configuração, velocidade e estabilidade Apache bate todos os concorrentes. É também o servidor com maior difusão na Internet. Interessa principalmente por poder fazer de *front-end* de um cluster de servidores Mongrel para onde são distribuídos os vários pedidos.
- ***ngingx*** – um servidor relativamente recente com bastante crescimento nos últimos anos e que serve, hoje em dia cerca de 4% das aplicações na Internet. Tal como o Apache faz de distribuidor de pedidos com bastante eficiência.

2.11.Deployment

A passagem de uma aplicação para produção (*deployment*) é uma tarefa que tende a ser mais complicada à medida que a aplicação cresce, principalmente se já tiver uma versão anterior a funcionar. Isto torna-se ainda mais complexo se a aplicação for distribuída por vários servidores.

O Capistrano é um utilitário que se integra com Rails. Dando uma 'receita' simples ao programa descrevendo a topologia da aplicação em produção conseguimos poupar todas as dores de cabeça envolvidas no processo tendo apenas que escrever um comando. Além disso permite ainda controlo de versões, sendo igualmente fácil voltar a ter em produção uma versão anterior da nossa aplicação.

2.12.Outras frameworks

Existem dezenas de outras *frameworks* e muitas até há muito mais tempo que Rails. De seguida apresentam-se algumas tendo sido escolhidas as mais usadas ou mais equiparáveis a Rails.

Todas as *frameworks* seguem o mesmo modelo geral, apresentado na figura 3: um *browser* web que comunica com um servidor HTTP que por sua vez comunica com uma camada lógica e este comunica com uma camada persistente, normalmente representada por uma base de dados.

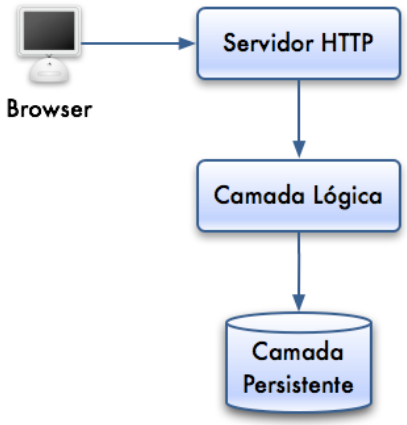


Figura 3 *– Funcionamento comum a todas as frameworks de desenvolvimento web.*

As *frameworks* mais maduras contêm geralmente múltiplas bibliotecas úteis para o desenvolvimento web, bem como ferramentas diversas, num pacote coeso e bem estruturado.

2.12.1.Java (J2EE)

Apesar de Ruby on Rails ser uma *framework* nova e excitante para muitos, o *core* da arquitectura segue os padrões presentes em J2EE. A filosofia de desenvolvimento de aplicações web é o que diferencia estas duas *frameworks*. Rails dá primazia a código explícito ao invés de ficheiros de configuração e a natureza dinâmica do Ruby gera muito código no *runtime*. Grande parte da *framework* Ruby on Rails foi criada como um projecto único e o desenvolvimento de uma aplicação beneficia de um conjunto de componentes homogéneos. Em contraste, uma *stack* típica de J2EE tende em ser construída de componentes diferentes (um conjunto de bons componentes entre um grande leque) que são geralmente desenvolvidos independentemente e são extensivamente usados ficheiros XML para os ligar e tratar da configuração.

Seria, no entanto, um erro dispensar-se completamente J2EE em favor de Rails. J2EE é um standard muito bem estabelecido com inúmeras implementações sólidas. É uma tecnologia com provas dadas e robustez comprovada.

No entanto trabalhar com uma *framework* em Java implica conhecer bem muitas das tecnologias envolvidas (Struts, Hibernate, Spring, Axis, etc.) sendo o seu número muito elevado e por vezes bastante complexo e imbricado, tornando a escolha das tecnologias adequadas a um projecto uma tarefa difícil. Para se atingir um nível de produtividade semelhante ao de outras *frameworks* exige-se assim muito mais prática e habituação. Além de ser necessária muito mais configurações numa aplicação em J2EE, existe o *overhead* do processo de compilação e muitas vezes os erros cometidos no código não são detectados na compilação mas apenas no *runtime* e consequentemente será difícil reagir rapidamente a mudanças, característica fundamental numa interface web.

Apesar de tudo os programadores Java têm ao dispor um vasto leque de escolhas. Além de haver um grande número de *frameworks* J2EE é relativamente fácil fazer integração com a infinidade de aplicações Java já desenvolvidos para os mais variados propósitos.

2.12.2.Zope+Plone

Zope (*Z Object Publishing Environment*) é um servidor de aplicações web desenvolvido totalmente em Python e é um poderoso ambiente de programação web, que permite desenvolver aplicações remotamente. As aplicações de uso do Zope são as mais diversas, porém tem sido utilizado em larga escala por diversas corporações como CMS.

As aplicações criadas no Zope são chamadas de "Produtos", sendo Plone um dos mais usados em aplicações web. Plone é uma *framework* de gestão de conteúdo profissional. O Plone é sem dúvida, um dos melhores CMS *Open Source* do mercado, sendo utilizado pela NASA, IDGnow, AOL entre outros.

A combinação Zope/Plone constitui uma *framework* muito sólida e permite construção de aplicações web com um esforço mínimo. No entanto foca-se no desenvolvimento de aplicações típicas de gestão por um CMS, não sendo fácil a implementação de aplicações de outro carácter.

2.12.3.Django

Django é uma *framework Open Source* desenvolvida em Python, que segue os padrões do modelo MVC de um modo não restritivo. O objectivo principal é facilitar a criação de *websites* complexos e com muitos acessos a uma base de dados de forma rápida e simples. Foca-se na reutilização e ligação entre os diversos componentes bem como no desenvolvimento ágil e o princípio DRY.

A linguagem Python é usada de modo transversal, da mesma forma que Ruby é usado na *framework* Ruby on Rails sendo uma linguagem simples mas menos específica. Com uma interface de administração gerada automaticamente e com autenticação integrada, os projectos em Django estão mais virados para CMS sendo menos flexível do que Rails.

2.12.4. TurboGears

TurboGears é mais uma *framework* de desenvolvimento web em Python, em tudo semelhante a Django. Tem uma grande comunidade e conta já com um livro inteiramente dedicado ao desenvolvimento nesta *framework* e com algumas aplicações a correr no mundo real.

É uma *framework* mais flexível do que Django, com melhor suporte para AJAX, não estando tão especializada em CMS, mas por essa razão também é mais complexa.

Na figura 4 apresenta-se um quadro resumo com alguns parâmetros para as *frameworks* em Python apresentadas e Ruby on Rails, segundo um comparativo de Sean Kelly [12]. Neste quadro o autor analisa os seguintes parâmetros:

- t*ime* – tempo total que demorou a fazer uma aplicação padrão muito simples, em minutos;
- *easy* – facilidade com que se desenvolveu essa aplicação de 0 - muito fácil a 1 - muito difícil;
- *validation* – a *framework* possui ou não validações de dados?
- *auth/auth* – a *framework* tem suporte para autenticação por defeito?
- *templates* – presença ou não de sistema de *templates*;
- *docs* – a *framework* tem documentação de fácil acesso e completa?
- *i18n* – internacionalização;

tech	time	easy	valid-ation	auth/ auth	temp-lates	docs	i18n
Rails	13	0.3	✔		✔	0.9	
Zope (Plone)	7	0.9	✔	✔	✔	0.9	✔
Turbo Gears	59	0.4	✔		✔	0.4	
Django	16	0.8	✔	✔		0.3	

Figura 4 *– Quadro resumo com algumas características de algumas frameworks segundo o vídeo comparativo de Sean Kelly [12].*

2.12.5.Frameworks PHP

PHP é uma linguagem muito usada para criação de aplicações web, tendo sido criada especificamente para esse efeito. Como tal, não seria de estranhar que surgissem *frameworks* de desenvolvimento nesta linguagem. Na verdade existem dezenas de *frameworks* em PHP, que facilitam em muito o desenvolvimento de aplicações com esta linguagem. No entanto não resolvem o principal problema, razão que leva a escolher *frameworks* noutras linguagens – o código PHP tende a ficar muito confuso com o evoluir da aplicação. Além disso são pobres em ferramentas adicionais como consola, migrações e mecanismo de testes. Não deixam de ser uma boa solução para casos em que o servidor não permite usar outras linguagens menos comuns, uma vez que PHP foi desde a sua invenção uma das linguagens mais usadas para aplicações web e por isso está muito bem disseminada.

2.12.6.Conclusão

Dadas as características da *framework* Ruby on Rails, pode concluir-se que deve, sem dúvida, considerar-se como uma ferramenta adequada ao desenvolvimento de aplicações web. É uma *framework* bem construída, com componentes homogéneos e que se interligam bem e baseia-se em padrões vastamente usados nos ambientes empresariais. Uma característica fundamental para a empresa é ser uma *framework* gratuita.

Além disso está bem adaptada para o tipo de desenvolvimento praticado na Mentes Virtuais, podendo contribuir para um salto na produtividade. Com os mecanismos de migrações e *deploy* está também talhada para o trabalho em equipa resolvendo potenciais problemas resultantes do trabalho colaborativo. A *framework* oferece uma capacidade de resposta rápida à mudança o que é fundamental, uma vez que facilita o diálogo com os clientes, havendo uma troca de ideias mais clara, com versões funcionais frequentes.

Mas, a *framework* não é perfeita, tendo sido já apresentados alguns pontos em que é ultrapassada por outras semelhantes. De seguida identificam-se também outras desvantagens:

- o uso transversal da linguagem Ruby no desenvolvimento de aplicações pode implicar um *overhead* inicial na sua aprendizagem e habituação. Só depois de alguma experiência com esta linguagem se consegue tirar partido da sua simplicidade a favor da produtividade;
- Ruby não é uma linguagem de alta performance, ficando atrás de outras semelhantes como Perl ou Python neste campo e por isso não é usada em aplicações que possam exigir performance elevada. Alguns testes já feitos provam que Ruby é cerca de duas vezes mais lento na execução de um mesmo bloco de código que Perl ou Python [16].
- a *framework* não tem bom suporte para outras línguas, havendo, no entanto *plugins* ou podendo definir-se outra língua se necessário;
- apesar de estar muito bem adaptada ao desenvolvimento ágil e à interacção em equipas de pequena dimensão, tal não se torna verdade quando aplicada a equipas complexas e de centenas de programadores, onde geralmente o processo de desenvolvimento é mais direccionado à documentação;
- a escassez de alojamento com suporte para aplicações Rails é ainda um problema para quem faz desenvolvimento nesta *framework*.

3. Trabalho efectuado

Depois de efectuado o estudo sobre a *framework* passou-se a uma segunda fase, em que se tentaram aplicar os conhecimentos adquiridos no desenvolvimento de duas aplicações diferentes. O objectivo principal no desenvolvimento destas aplicações foi ganhar competências para o trabalho a realizar no 2º semestre do estágio curricular.

3.1. Método de trabalho na empresa

O desenvolvimento de um produto na Mentes Virtuais, principalmente na área de aplicações web, não implica o desenvolvimento de documentação extensiva, uma vez que as equipas de trabalho são pequenas (duas a quatro pessoas). Existe um grande diálogo entre os elementos de um projecto, sendo encorajada a troca de ideias e conhecimento entre eles.

No entanto, todos os projectos têm uma documentação básica (requisitos e alguns protótipos quando necessário) e têm um controlo de *features* e *bugs* num sistema de gestão de projectos – Redmine (www.redmine.org) instalado num servidor interno onde se pode seguir o desenvolvimento de um projecto, bem como adicionar ou editar documentação já realizada, ficando acessível imediatamente, de modo simples a todos os elementos, podendo haver colaboração na sua especificação (figura 5).

Este sistema permite ainda alertar por email os participantes de um projecto de alterações efectuadas, criar um diagrama de Gantt e consultar um calendário de metas.

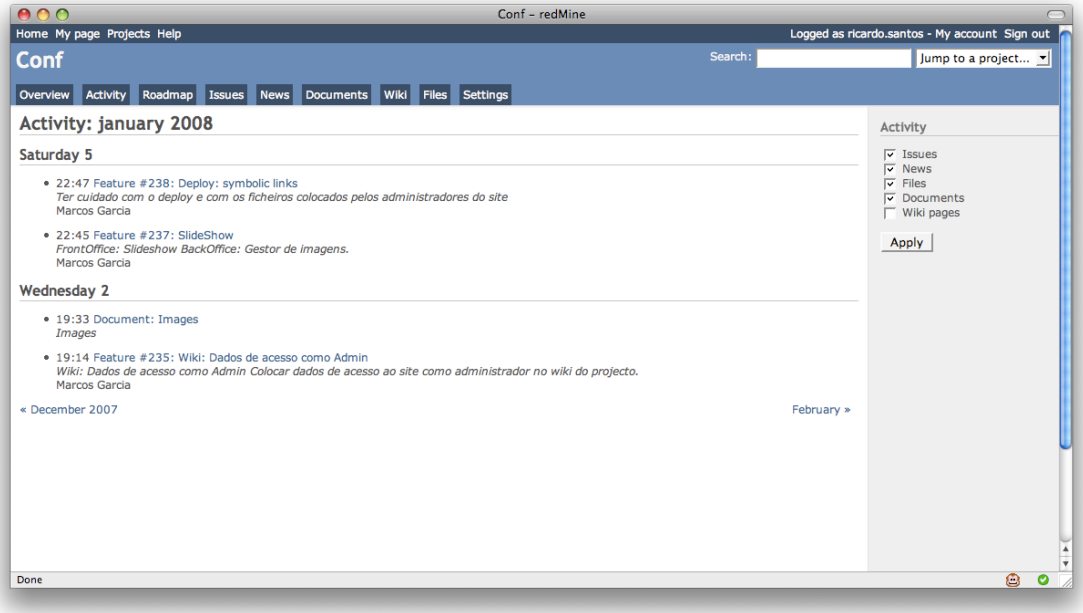


Figura 5 – *Exemplo de uma página de um projecto usando o gestor de projectos redmine.*

Além disso todos os projectos têm um repositório de controlo de versões, sendo usado um servidor SVN para o efeito. Desta forma anulam-se os problemas adjacentes ao desenvolvimento em paralelo por diferentes pessoas numa equipa.

A primeira versão funcional da aplicação é sempre posta em produção o mais rapidamente possível para que o cliente possa dar o seu parecer e normalmente surgem novos requisitos. À medida que as

novas funcionalidades são implementadas a aplicação em produção é actualizada, obedecendo-se ao ciclo presente na figura 6.

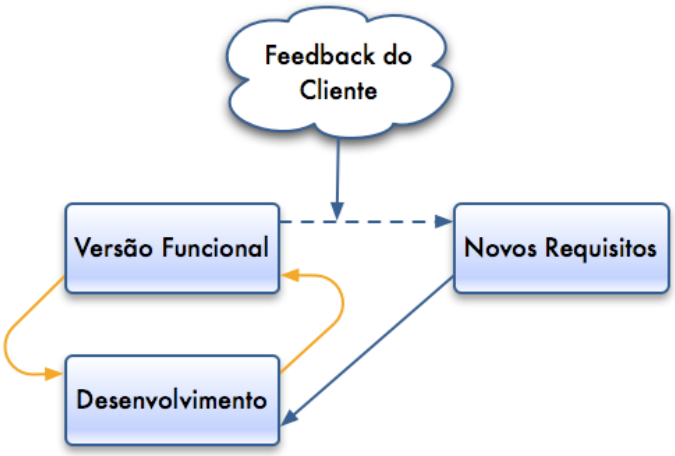


Figura 6 *– Ciclo de desenvolvimento de um projecto. A amarelo está o ciclo de desenvolvimento sem recurso ao cliente. O feedback do cliente pode ou não levar ao aparecimento de novos requisitos.*

A equipa de desenvolvimento pode também, sempre que se proporcione, propor novas funcionalidades, havendo assim uma comunicação bidireccional mais ou menos regular, entre ambas as partes.

3.2. Aplicação 1 – Conferences

3.2.1. Introdução

Esta é uma uma aplicação de gestão de conferências que foi aplicada directamente a um caso concreto e real. Neste tipo de aplicação importa, acima de tudo, permitir a gestão dos conteúdos de forma fácil e directa. Como o objectivo principal é a aplicação de conhecimentos adquiridos o sistema desenvolvido não está em fase final, sendo uma versão base, com algumas funcionalidades chave, podendo evoluir no futuro.

Foi objectivo deste trabalho, não só a aplicação dos conhecimentos adquiridos, mas também produzir um sistema suficientemente genérico, de modo a poder ser aplicado em diferentes conferências, e ao mesmo tempo suficientemente modular permitindo reutilizar alguns componentes em aplicações de outro carácter.

O sistema foi aplicado a uma conferência internacional do Departamento de Matemática da Universidade de Aveiro, a realizar em Maio de 2009 -– ICMSE (*International Conference in Mathematics, Sciences and Science Education*). Assim pode testar-se o sistema para um caso real, não sendo uma aplicação crítica dada a margem temporal que proporciona.

3.2.2. Especificação

A análise de requisitos detalhada e os protótipos desta aplicação podem ser consultados no Anexo A e Anexo B, respectivamente.

Esta aplicação usa uma base de dados PostgreSQL 8, e as imagens são guardadas em disco, sendo usado uma *gem* específica, RMagick [19], para o seu tratamento.

Interface

Toda a interface é em inglês e o sistema apresentará, esquematicamente a estrutura da figura 7. O menu editável contém essencialmente as páginas de conteúdo estático, editáveis pelos administradores e o menu pessoal fica disponível, mediante autenticação, com os itens específicos a cada perfil de utilizador.

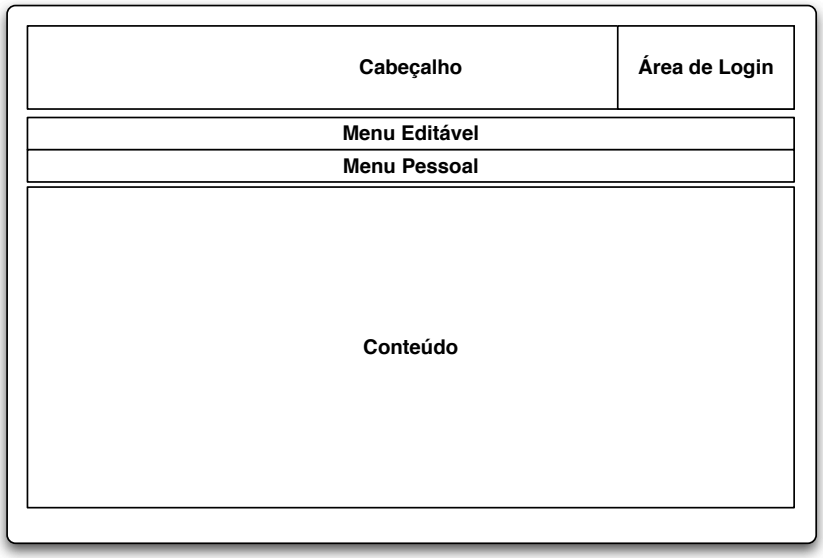


***Figura 7** – Estrutura gráfica base da aplicação. O menu pessoal só está visível após autenticação.*

A maioria dos conteúdos terão de ser editados pelo cliente, exceptuando algumas páginas, como as de formulários de registo, submissão de documentos e *slideshow* de imagens. Esta edição deve ser fácil e directa, tanto quanto possível, sem envolver conhecimentos de HTML.

Funcionalidades chave

O sistema tem de permitir a gestão de utilizadores, e a atribuição de perfis a esses utilizadores. Os utilizadores estão divididos segundo a tabela 1. Não está representado o utilizador público, sem registo, que pode ver todos os conteúdos públicos.

Nome	Permissões
Admin	Pode editar os conteúdos estáticos da aplicação, gerir utilizadores e submissões de documentos e editar a galeria de imagens.
Attendee	Utilizador base. Pode ver todos os conteúdos e a lista de participantes. Este utilizador ganha o papel de 'Author' a partir do momento em que submete um documento.
Author	Pode submeter documentos e ver ou apagar os documentos que submeteu.

Nome	Permissões
SOCM	*Scientific Organization Comittee Member.* Pode ver uma lista de documentos que lhe são atribuídos e fazer a sua revisão.

Tabela 1 *– Perfis de utilizador na aplicação de conferências.*

Como se pode ver na tabela acima, qualquer utilizador pode submeter ficheiros, e estes podem ser de 3 tipos: *paper*, *research report* e *poster*; não existindo à partida diferença entre eles, além da nomenclatura.

Os utilizadores terão ao dispor uma funcionalidade de recuperação de *password* com recurso ao envio de um *email* para o endereço fornecido no registo.

3.2.3. Implementação

A implementação foi feita dividindo as funcionalidades básicas em 4 módulos distintos: gestão de utilizadores, gestão de conteúdos estáticos, gestão de documentos e galeria de imagens.

Gestão de utilizadores

Para a gestão de utilizadores foi usado um *plugin* já existente – *acts_as_authenticated* [24] que faz a gestão de sessões de forma segura e trata o envio de *emails* (usado por exemplo na recuperação de *password*), sendo fácil de acrescentar funcionalidades e de adaptar ao projecto.

Inicialmente houve uma tentativa de integrar um *plugin* de gestão de utilizadores, com permissões e grupos, bastante completo – xaf-access-plugin. Este *plugin* foi desenvolvido no âmbito de um projecto na empresa – *XAF- Extensible Architecture Framework* – que é uma arquitectura de base a inúmeras plataformas da família de produtos da PT Inovação. No entanto o xaf-access-plugin revelou-se demasiadamente complexo e pouco flexível. Desta forma adoptou-se pelo desenvolvimento de um módulo simples, onde a cada utilizador está associado um atributo 'role' que o identifica como fazendo parte de um dos grupos da tabela 1.

Gestão de conteúdos

Para a gestão de conteúdos recorreu-se inicialmente a um *plugin* chamado Comatose [20] que permitia transformar parte da aplicação num CMS. No entanto, após alguma experimentação, veio a revelar-se demasiadamente complexo para o problema em questão, complicando processos que se queriam simples além de ser mais difícil de adaptar ao modelo de utilizador a implementar.

Assim, e como um editor visual se justifica quando se trata de editar conteúdos estáticos numa página web, recorreu-se a um outro *plugin* – FCKEditor [21] que apenas permite usar um editor com essas características de modo simples. Além disso havia a necessidade de editar conteúdos parciais, como os patrocinadores do evento, como está ilustrado na figura 8.

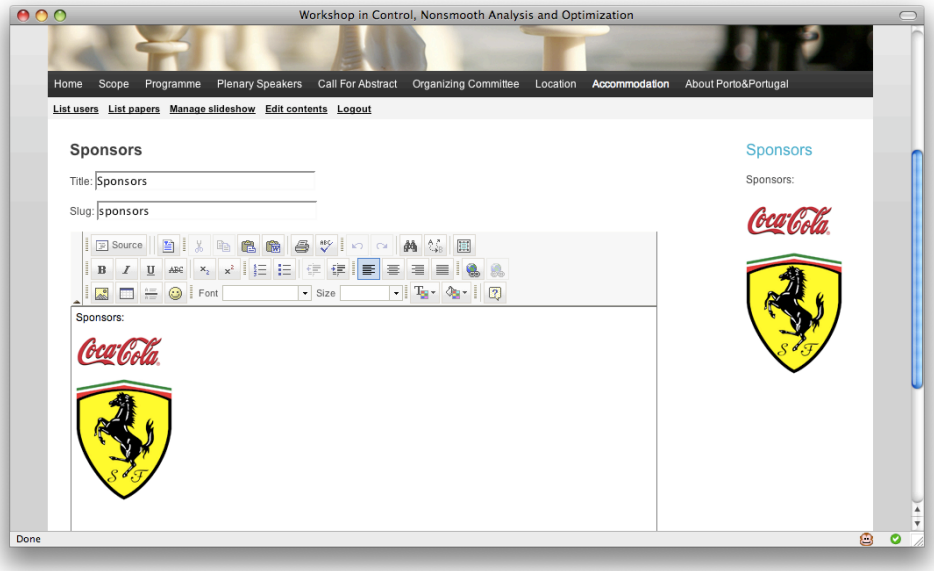


Figura 8 *– Edição de conteúdos estáticos com a ajuda de um editor visual da coluna de patrocinadores.*

Cada página de conteúdo estático está a ser guardada na base dados, com todas as formatações em HTML. Para as identificar usou-se um atributo *slug*, que é gerado automaticamente (e pode ser editado posteriormente), fazendo uma substituição simples do título da página, passando todos os caracteres não alfanuméricos para "_".

Gestão de documentos

A gestão de documentos é feita de um modo muito simples, onde cada utilizador devidamente autenticado pode enviar através de um formulário um documento, em qualquer formato (visto não haver uma lista até à data dos formatos que devem ser suportados). Os ficheiros enviados são guardados numa pasta específica: */uploads/papers/id_utilizador/id_paper/ficheiro* e ao mesmo tempo registados na base de dados de modo ao sistema identificar os ficheiros por um *id*. Desta forma um utilizador pode enviar ficheiros com o mesmo nome que outros utilizadores e até que outros seus enviados anteriormente.

Os documentos enviados podem ser posteriormente apagados. Os administradores podem listar e apagar os ficheiros enviados por qualquer utilizador.

Em versões futuras será necessário um mecanismo de atribuição de documentos a utilizadores com a *role* "SOCM". Mas até à data também ainda não foi possível chegar a um consenso quanto ao modo de como essa atribuição é feita ou de como estes utilizadores submetem a revisão dos documentos.

Galeria de imagens

A galeria de imagens foi uma funcionalidade que surgiu já no fim do desenvolvimento da aplicação e tem como principal objectivo permitir que os administradores enviem fotografias que ficam visíveis na forma de um *slideshow* a todos os visitantes.

Esta funcionalidade também foi desenvolvida tendo em vista a sua transformação futura num módulo independente de gestão de uma galeria de imagens com opção para o *slideshow* automático.

O *slideshow*, que se queria dinâmico, envolveu o uso de uma biblioteca em JavaScript chamada MooTools [22] que permite efeitos dinâmicos de boa qualidade e compatíveis com todos os *browsers* actuais. As imagens ficam guardadas em disco em dois tamanhos diferentes (tratamento das imagens feito com RMagick) numa directoria */uploads/slides/* e ao mesmo tempo registadas na base de dados para mapeamento directo para um modelo Image, a ser usado na aplicação. Imagens enviadas com o mesmo nome são renomeadas para *imagem_X.extensão* onde X é um número inteiro. O valor de X vai sendo incrementado se o nome obtido já existir. Assim, se já existir por exemplo um ficheiro *file.jpg* o ficheiro que se está a enviar é renomeado para *file_1.jpg*, mas se este já existir nesse caso é renomeado para *file_2.jpg* e assim consecutivamente até não haver nenhum ficheiro com o nome desejado.

3.2.4. Desenvolvimento futuro

O sistema desenvolvido, apesar de relativamente simples, consegue responder às necessidades básicas de uma aplicação de conferência. Ainda assim há diversos pontos onde se poderá evoluir, visto que o esforço despendido nesta aplicação não foi o suficiente para a tornar completa, nomeadamente:

- tornar o módulo de utilizadores mais flexível, uma vez que ainda é necessária alguma configuração para o incluir noutras aplicações;
- protecção dos ficheiros enviados não só no envio mas também na consulta;
- refinamento do módulo de galeria de modo a ser genérico e flexível o suficiente para incluir noutras aplicações com o mínimo esforço;
- restrição automática à submissão de documentos de acordo com uma data limite a definir;
- conclusão de alguns aspectos pendentes como a revisão de documentos;

3.3. Aplicação 2 – Anúncios

3.3.1. Introdução

A segunda aplicação é um sistema de anúncios. Tal como na aplicação anterior é do máximo interesse desenvolver uma aplicação genérica e modular, mas nesta não se aplicou a nenhum caso real. Um dos objectivos é desenvolver um sistema de uso fácil e com uma interface simples, dado que é aí que falham muitas soluções existentes.

A aplicação foi proposta pela empresa porque é do seu especial interesse ter ao dispor uma aplicação do género dadas algumas oportunidades de negócio.

3.3.2. Especificação

A análise de requisitos detalhada e os protótipos desta aplicação podem ser consultados no Anexo C e Anexo D, respectivamente.

Tal como a aplicação anterior todos os dados são guardados numa base de dados PostgreSQL 8, e as imagens são guardadas em disco, sendo usado RMagick para o seu tratamento.

Interface

A interface desta aplicação terá de ser simples e apelativa. A estrutura base segue o esquema da figura 9. Numa primeira fase toda a interface será em português uma vez que o possível mercado é o nacional, podendo ter de ser adaptado a diversas línguas no futuro.

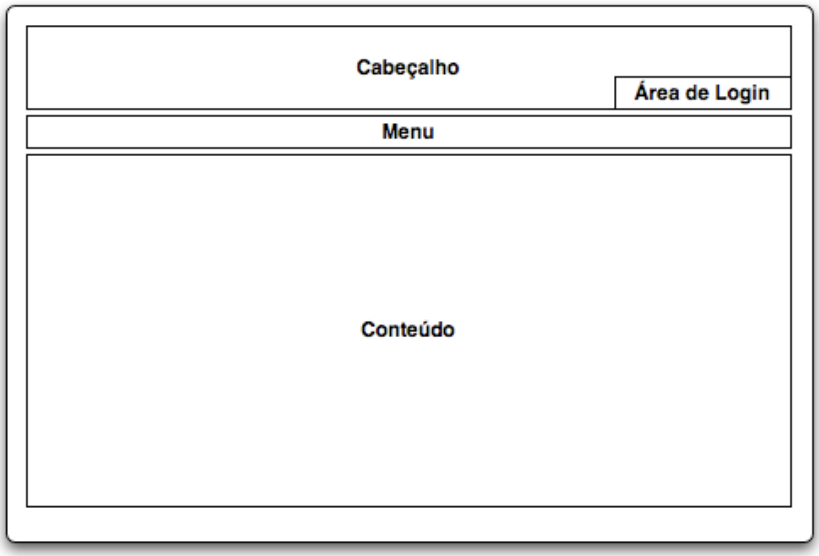


Figura 9 *– Estrutura gráfica base da aplicação. O menu pessoal só está visível após autenticação.*

Depois de efectuada a autenticação surgem itens específicos no menu, diferindo consoante o perfil do utilizador.

Funcionalidades chave

A aplicação deve ter diferentes perfis de utilizador, sendo neste caso definidos apenas dois, para além de um utilizador público, sem registo que poderá ver todos os conteúdos públicos. A descrição dos perfis de utilizador está na tabela 2.

Nome	Permissões
User	Pode editar o perfil, adicionar anúncios, listar/editar/validar os anúncios submetidos e adicionar comentários.
Admin	Tem todas as permissões dos utilizadores registados e pode gerir utilizadores. Pode também editar/apagar todos os anúncios de qualquer utilizador.

Tabela 2 *– Perfis de utilizador na aplicação de anúncios.*

A validação dos anúncios passará por um mecanismo de envio de uma mensagem através de um *Gateway* SMS sendo assim cobrado o preço definido para o sistema. Esta ligação não foi implementada nesta fase do estágio, referindo-se aqui porque poderá ser implementada no futuro. O ciclo de validação de um anúncio está esquematizado na figura 10. Os anúncios só serão visíveis para outros utilizadores quando validados desta forma. Além disso todos os anúncios têm uma data limite, a partir da qual terão de ser activados novamente e deixam de estar visíveis.

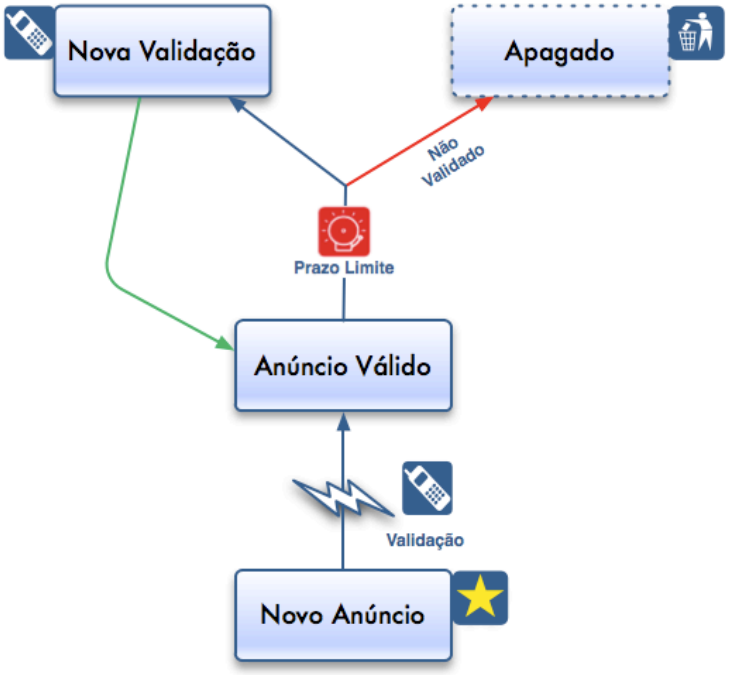


Figura 10 *– Ciclo de validação de um anúncio. Um utilizador pode também apagar um anúncio a qualquer altura, se assim entender.*

Será dada uma margem de tempo para nova validação do anúncio, após o qual será apagado, se não for novamente validado, como se mostra na figura 10.

Cada anúncio tem associada uma descrição, um máximo de cinco imagens e podem ser adicionados comentários por parte de outros utilizadores. Uma das imagens é identificativa do anúncio aparecendo em ponto pequeno quando o anúncio é visto em listagens (por exemplo depois de uma pesquisa).

3.3.3. Implementação

Esta aplicação implicou maior cuidado com os detalhes, visto que é suposto ter uma interface fácil e directa. Como toda a interface é em português teve de se despender algum tempo em tarefas como personalização das mensagens de erro e de formatos das datas.

Gestão de utilizadores

Para a gestão de utilizadores usou-se o mesmo mecanismo que na aplicação anterior. O módulo de gestão de utilizadores foi adaptado facilmente. A única diferença significativa é a introdução do atributo "data de nascimento" que implicou a utilização de um *plugin* de gestão de datas – CalendarDateSelect [23] – através de um calendário de ajuda visual (figura 11).

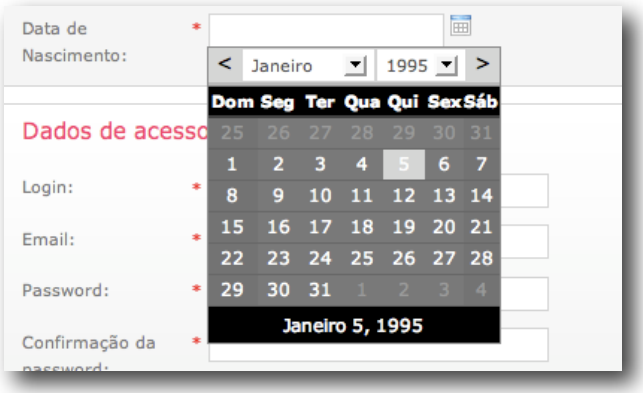


Figura 11 *– Popup de ajuda visual no preenchimento da data de nascimento.*

Gestão de anúncios

Toda a gestão de anúncios e comentários funciona como previsto à excepção do mecanismo de validação descrito na figura 10. Este mecanismo de validação por SMS está a ser alvo de estudo quanto à sua viabilidade e tudo aponta para uma primeira fase gratuita passando depois para o regime de cobrança conforme referido.

A interface já entra com o aspecto da validação em conta, não apresentando os anúncios não validados e obrigando a essa validação, mas ainda não se está a utilizar o serviço SMS uma vez que ainda não se chegou a um consenso de como será o seu funcionamento.

Aproveitando-se o *slideshow* da aplicação anterior cada anúncio tem um *mini-slideshow* quando são vistos os detalhes. A associação de imagens a um anúncio envolveu um esforço adicional uma vez que, para ser feita de um modo intuitivo, implicou alguns cuidados de interface e como tal recorreu-se ao uso de RJS e AJAX. Importava definir qual a imagem que era usada como identificativa do anúncio quando visto em listagem (por exemplo numa pesquisa). Aproveitando-se as potencialidades do Script.aculo.us fez-se esta gestão por *drag and drop* das imagens (figura 12). A ordem pela qual as imagens ficam será a ordem pela qual aparecem também no *slideshow*.

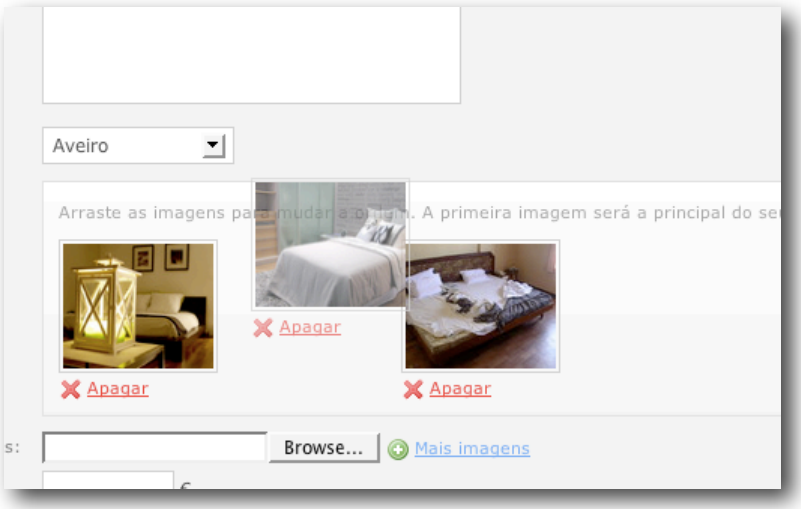


Figura 12 *– Reordenamento das imagens de um anúncio com interface de drag and drop.*

Pesquisa

A pesquisa de anúncios é uma funcionalidade fundamental. Nesta primeira fase a pesquisa foi implementada de modo simples, sem recurso a índices, e com um reduzido número de filtros – por título, utilizador e região, mas faz parte do trabalho futuro desenvolver esta funcionalidade para que seja o mais eficiente e completa possível e ao mesmo tempo fácil de usar.

De modo a ser mais directo é usado um pedido AJAX, que apresenta os resultados mal tenha resposta, e ao mesmo tempo permite que o utilizador obtenha informação visual do que está a acontecer (pesquisa em curso) como é visível na figura 13.

Figura 13 *– Pesquisa de anúncios. À esquerda foi enviado o pedido por AJAX e á direita são recebidos os resultados.*

Como a região do anúncio pode ser de extrema relevância foi adicionada uma lista de regiões de Portugal e todos os anúncios têm uma região a que pertencem. Desta forma a região pode figurar como um filtro essencial.

3.3.4. Desenvolvimento futuro

Ao contrário da aplicação anterior, esta não foi directamente aplicada a um cliente real, sendo uma plataforma desenvolvida para adaptar possíveis projectos. Assim, além de ter havido um conjunto de funcionalidades que ficaram planeadas e não implementadas, há ainda bastante trabalho para o futuro. O trabalho realizado foi do agrado da empresa pelo que, na segunda fase do estágio, a aplicação será desenvolvida até uma fase em que possa ser posta em prática com um conjunto de funcionalidades adicionais, descritas no capítulo seguinte.

4. Trabalho futuro

Na sequência do trabalho realizado neste primeiro semestre, foi inicialmente proposto que o estagiário integrasse o desenvolvimento de uma nova aplicação para a Universidade de Aveiro a desenvolver em Ruby on Rails. Esta aplicação tinha já alguns requisitos levantados e afigurava -se como uma boa proposta de trabalho para o segundo semestre. Contudo, por motivos que lhe foram alheios, e como o meio empresarial nem sempre pode dar certezas de futuro, esse projecto foi cancelado.

Assim, como o trabalho desenvolvido no âmbito da aplicação de anúncios foi muito satisfatório, sendo também uma aplicação de grande interesse para a empresa, o estagiário vai levar a aplicação até a uma fase final, implementando um conjunto de novas funcionalidades. A aplicação de anúncios simples, já desenvolvida, servirá como base de trabalho para a segunda fase, sendo possível que sofra algumas alterações. Uma vez que a aplicação a desenvolver na segunda fase do estágio é bastante mais complexa do que a que inicialmente foi proposta terá de ser feito novo levantamento de requisitos.

Assim, entre as novas funcionalidades temos:

- exploração das API's fornecidas pelo Google;
- introdução de um método de autenticação distribuída (OpenID);
- incorporar o conceito de rede social;
- desenvolvimento de módulo de partilhas;
- exportação de dados em RSS;
- compatibilidade da aplicação com dispositivos móveis;

Paralelamente é também objectivo da segunda fase do estágio obter módulos independentes (preferencialmente na forma de *plugins* Rails) que podem passar pela generalização e refinamento de trabalho já começado, por um processo semelhante ao já praticado na primeira fase do estágio. Estes módulos, ainda sem nome definido, serão:

1. Módulo de gestão de galeria de imagens – semelhante à funcionalidade de galeria usado na aplicações de conferências mas mais flexível.
2. Módulo de gestão de conteúdos estáticos – também semelhante à funcionalidade já desenvolvida.
3. Módulo de gestão de notícias – um novo módulo a ser utilizado em algumas aplicações que justificam a sua implementação.

Estes três módulos têm utilização planeada para algumas aplicações futuras na empresa, tendo de ser suficientemente flexíveis para serem usados de modo transversal.

Surge ainda uma tarefa adicional de migração da autenticação da aplicação uma vez que, com a nova versão de Ruby on Rails lançada recentemente o *plugin* utilizado tornou-se *deprecated*. No Anexo E apresenta-se um plano geral de trabalho para a segunda fase do estágio.

Na aplicação de conferências poderão ser emendados possíveis erros ou incoerências segundo *feedback* do cliente onde o sistema foi aplicado (Departamento de Matemática da Universidade de Aveiro) mas não vai competir ao estagiário desenvolver mais funcionalidades.

5. Conclusões

Durante esta primeira fase do estágio comprovaram-se as potencialidades da *framework* Ruby on Rails como exímia no desenvolvimento de aplicações web. Houve oportunidade de experimentar as diversas ferramentas presentes na *framework* e adquirir conhecimentos suficientes para integrar um projecto de maiores dimensões sem problemas acrescidos.

No decorrer do desenvolvimento das aplicações foi lançada uma nova versão da *framework*. Isto implicou um trabalho adicional de migração para a nova versão mas por outro lado permitiu ganhar-se experiência neste tipo de procedimento, que não é raro uma vez que novas versões têm sido desenvolvidas regularmente.

De uma forma geral considera-se que a preparação do estágio foi bem sucedida e correu como o esperado. Houve alguns contratempos na tentativa de integração de dois diferentes *plugins* na aplicação de conferências como foi relatado em secções anteriores, mas não comprometendo o cumprimento dos objectivos do estágio. O percurso de desenvolvimento do estágio, embora sempre apoiado por colegas com maior experiência, esteve sempre associado a um grau de autonomia elevado e foi depositada confiança no resultado desenvolvido.

6. Referências

1. Marshall, Kevin, “Web Services on Rails”, O'Reilly, 2006

2. Bradburne, Alan, “Practical Rails Social Networking Sites”, Apress, 2007

3. Thomas, Dave; Hansson, David, “Agile Web Development with Rails – Second Edition”, Pragmatic Programmers LLC, 2006.

4. **http://www.inova-ria.pt**
   Página oficial da Associação de Empresas para uma Rede de Inovação em Aveiro onde se pode encontrar informação sobre a mesma e seus associados.

5. **http://www.mentesvirtuais.com**
   Página da Mentes Virtuais onde se encontra toda a informação sobre os serviços prestados assim como os seus clientes.

6. **http://www.wikipedia.org**
   A enciclopédia livre on-line.

7. **http://www.rubyonrails.org**
   Página oficial da *framework* Ruby on Rails.

8. **http://api.rubyonrails.org**
   Documentação da *framework* Ruby on Rails.

9. **http://www.onlamp.com/pub/a/onlamp/2005/01/20/rails.html**
   Rolling with Ruby on Rails.

10. **http://flex.org/ruby/**
    Integração de Flex e Ruby on Rails.

11. **http://pylonshq.com/RailsHelpers/**
    Helpers de Ruby on Rails implementados em Python.

12. **http://oodt.jpl.nasa.gov/better-web-app.mov**
    Vídeo comparativo de *frameworks* de desenvolvimento de aplicações web.

13. **http://java.sun.com/blueprints/patterns/MVC-detailed.html**
    Explicação detalhada da arquitectura MVC.

14. **http://wiki.rubyonrails.org/rails**
    Página com diversos *links* e informações acerca de Ruby on Rails.

15. **http://www.xfront.com/REST-Web-Services.html**
    Página sobre a tecnologia REST e construção de *Web Services* usando esta tecnologia.

16. **http://izumi.plan99.net/blog/index.php/2008/01/17/ruby-vs-php-performance/**
    Comparação da performance de várias linguagens incluindo Perl, Python e Ruby.

17. **www.prototypejs.org**
    Página oficial da biblioteca de JavaScript Prototype.

18. **www.script.aculo.us**
Página oficial da biblioteca de JavaScript Script.aculo.us.

19. **http://rmagick.rubyforge.org**
Página oficial da *gem* RMagick.

20. **http://comatose.rubyforge.org**
Página oficial do *plugin* de CMS Comatose.

21. **http://rubyforge.org/projects/fckeditorp/**
Página oficial do *plugin* FCKEditor.

22. **http://mootools.net**
Página oficial da biblioteca de efeitos dinâmicos Mootools.

23. **http://code.google.com/p/calendardateselect/**
Página oficial do *plugin* CalendarDateSelect.

24. **http://technoweenie.stikipad.com/plugins/show/acts+as+authenticated**
Página oficial do *plugin* acts_as_authenticated

Anexos

Índice

Índice de figuras

Conferences

Anúncios

Plano do Projecto

Índice de tabelas

Conferences

Anúncios

Anexo A : Conferences – Análise de Requisitos

1. Tabela de requisitos

Identificação	Nome	Descrição
R01	Idioma único	Toda a interface será em inglês.
R02	Perfis de utilizador	Na aplicação haverá diferentes perfis de utilizador, estando detalhados na tabela 2.
R03	Registo obrigatório	Só depois de feito um registo é que um utilizador se pode autenticar no sistema.
R04	Autenticação	Para visualização de conteúdos privados é necessária uma autenticação prévia. A autenticação é feita por *login* e *password*.
R05	Recuperar password	Os utilizadores têm de poder recuperar a *password* no sistema.
R06	Adicionar documentos	*Todos os utilizadores registados poderão adicionar documentos. Os documentos poderão ser de três tipos – paper, research report* e *poster*. Nenhum utilizador poderá enviar documentos que sobrepõem outros já existentes.
R07	*Slideshow* de imagens	Deverá ser possível visionar um *slideshow* dinâmico com imagens enviadas pelo(s) administrador(es).
R08	Formato de imagem do *slideshow*	O *slideshow* deve aceitar imagens nos seguintes formatos: PNG, GIF e JPEG. São guardadas duas imagens em disco. Uma com dimensões 40x30px para *thumbnail* e outra com 460x345px.
R09	Adição de uma imagem	A adição de uma imagem implica a sua cópia para o disco nos formatos constantes de R08 e registo na base de dados.
R10	Edição de conteúdos	A edição de conteúdos terá de ser feita de forma fácil e sem implicar conhecimentos de HTML.
R11	Paginações	A listagem de utilizadores e documentos deve ser paginada de modo a não criar *scrolls* demasiadamente grandes no *browser*.

Tabela 1 *– Tabela de requisitos da aplicação.*

Nome	Permissões
Admin	Pode editar os conteúdos estáticos da aplicação, gerir utilizadores e submissões de documentos e editar a galeria de imagens.
Attendee	Utilizador base. Pode ver todos os conteúdos e a lista de participantes. Este utilizador ganha o papel de 'Author' a partir do momento em que submete um documento.
Author	Pode submeter documentos e ver ou apagar os documentos que submeteu.
SOCM	*Scientific Organization Comittee Member.* Pode ver uma lista de documentos que lhe são atribuídos e fazer a sua revisão.

***Tabela 2** – Perfis de utilizador.*

2. Diagramas de Casos de Uso

Nos diagramas seguintes apresentam-se requisitos funcionais da aplicação de conferências. Não estão representadas funcionalidades relacionadas com a revisão de documentos e o papel dos utilizadores de perfil "SOCM" porque, ainda não se chegou a um consenso quanto a estas funcionalidades.

2.1. Registo e autenticação

Todos os utilizadores registados têm acesso a conteúdo privado. O utilizador pode ainda alterar todos os dados constantes no seu perfil quando bem entender. A informação dos utilizadores é guardada em base de dados.

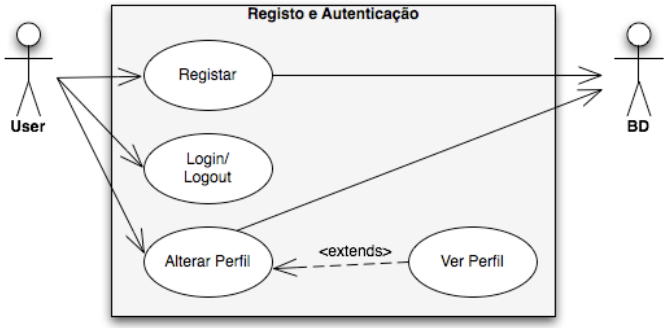


***Figura 1** – Diagrama de Casos de Uso para o registo e autenticação de um utilizador.*

2.2. Recuperação de password

Para recuperação de *password* é enviado um email com uma *hash* gerada automaticamente, que leva a um endereço usando essa *hash*, onde o utilizador pode redefinir a sua palavra-passe. Esta funcionalidade está associada a um *link* com o nome "Esqueceu a sua password?".

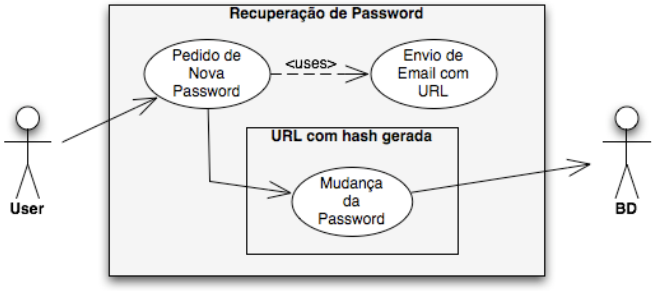


Figura 2 *– Diagrama de Casos de Uso para a recuperação de password com recurso a um URL.*

2.3. Funcionalidades de administrador

Todas as funcionalidades apresentadas como gestão implicam a adição, remoção e edição do objecto em questão. Omitiu-se a base de dados para maior clareza do esquema, mas todas as operações devem operar sobre a base de dados da aplicação.

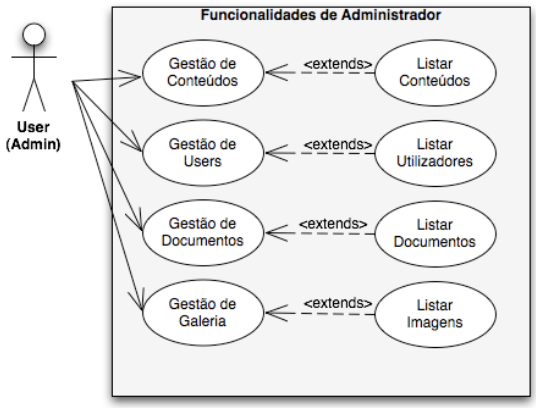


Figura 3 *– Diagrama de Casos de Uso para as funcionalidades de um utilizador com perfil de administrador (Admin).*

2.4. Adição de documentos

A adição de um novo documento implica não só guardar a sua referência na base de dados da aplicação como ao mesmo tempo guardar o próprio documento do servidor.

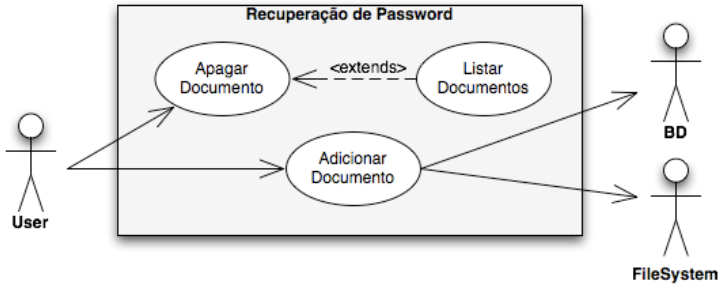


Figura 4 *– Diagrama de Casos de Uso para a adição de um novo documento.*

3. Modelo de dados

Nota: nenhuma das relações entre entidades, patente na figura, devem estar definidas na base de dados, uma vez que o Ruby on Rails as faz automaticamente consoante a sua definição nos modelos em Ruby. A pluralização das tabelas é feita automaticamente com a criação dos modelos.

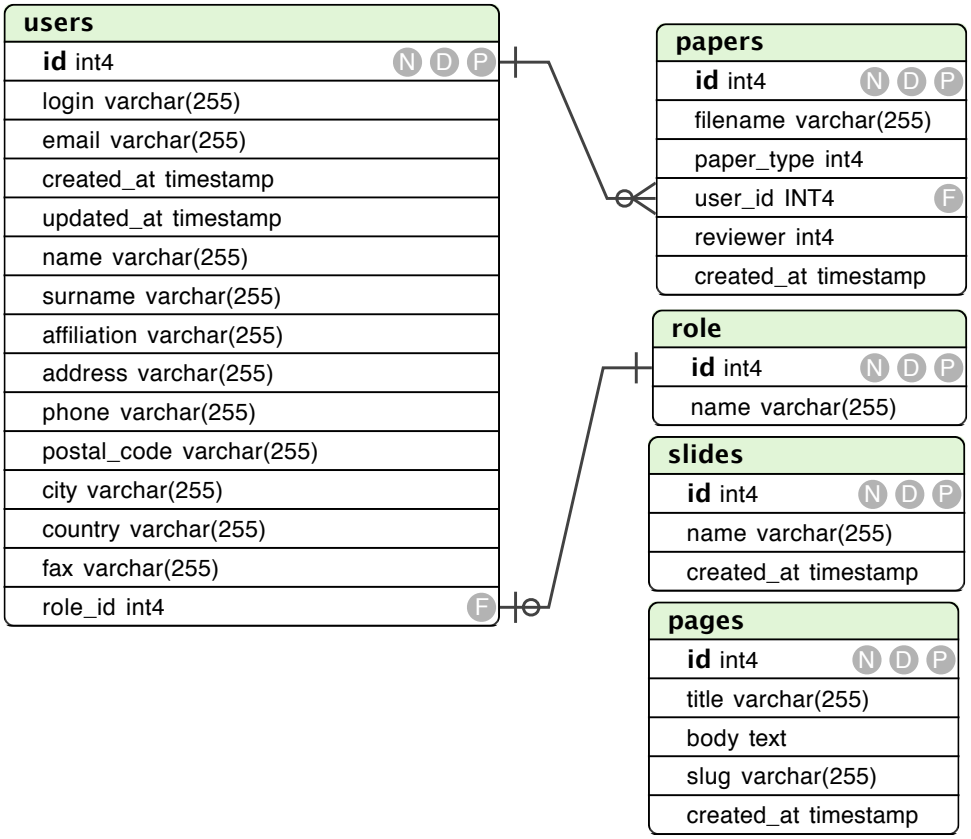


Figura 5 *– Modelo de dados da aplicação de conferências.*

Anexo B : Conferences – Protótipo

Aspecto geral da aplicação

A aplicação tem um formulário de *login* no cabeçalho. Quando o utilizador se autentica é adicionado um segundo menu com operações específicas como se pode ver na figura 7. Na parte direita de todas as páginas aparece uma coluna com patrocinadores.

Figura 6 *– Aspecto geral da aplicação para um utilizador não autenticado.*

Figura 7 *– Aspecto geral da aplicação para um utilizador autenticado.*

Registo

Na figura 8 está presente o registo de um novo utilizador. Este formulário é semelhante ao de edição dos dados pessoais, com a diferença que este último tem os campos já preenchidos com os dados do utilizador em questão.

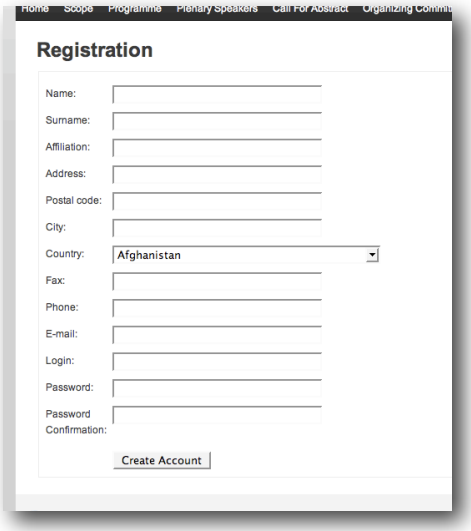


Figura 8 *– Formulário de registo na aplicação.*

Recuperação de password

Nas figuras 9 e 10 estão os formulários de recuperação de *password*. Primeiro é pedido o email de registo (figura 9), seguindo-se o envio de um email com um *link* gerado automaticamente que leva o utilizador ao formulário da figura 10.

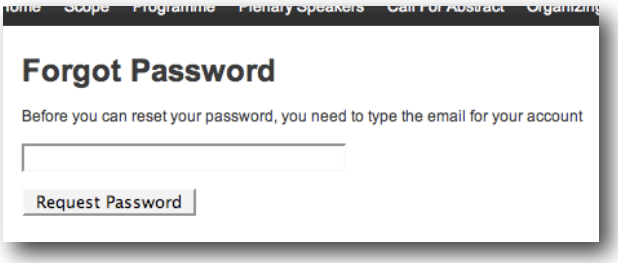
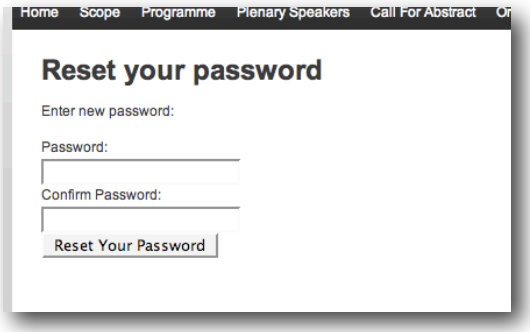


Figura 9 *– Formulário de pedido de nova password*



.***Figura 10*** *– Formulário de mudança de password depois de recebido o email com o link directo.*

Submissão e listagem de documentos

Na figura 12 aparece a listagem de documentos enviados e antes da tabela é apresentado um *link* para o formulário da figura 11.

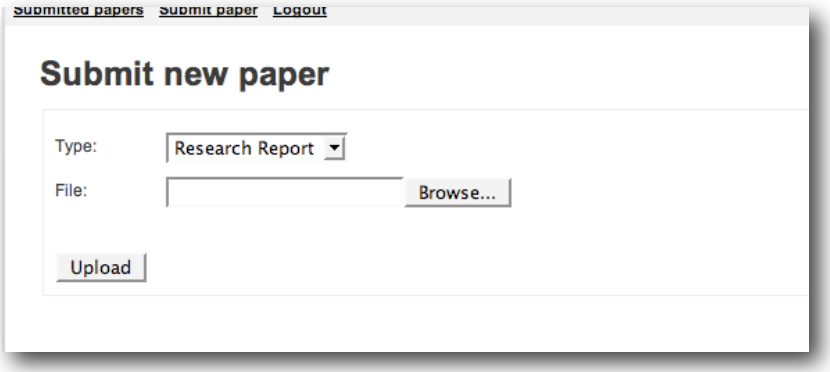


Figura 11 *– Formulário de submissão de um documento.*

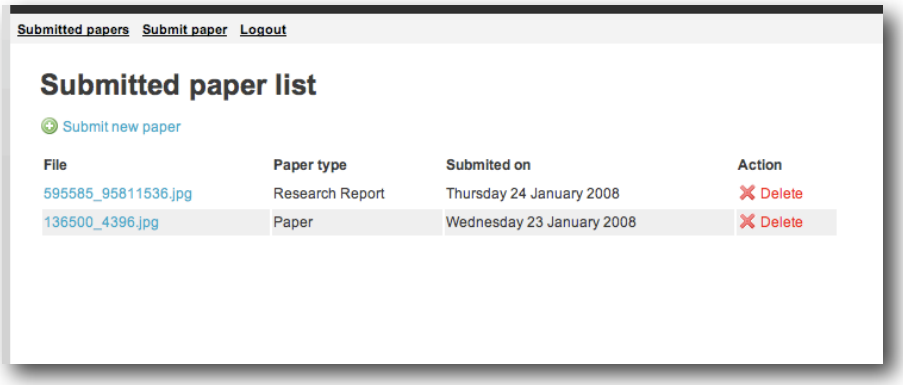


Figura 12 *– Lista de ficheiros enviados apresentada a um utilizador registado.*

Na figura 13 está a listagem dos documentos para os administradores do sistema. Aqui aparecem todos os documentos enviados por qualquer utilizador.

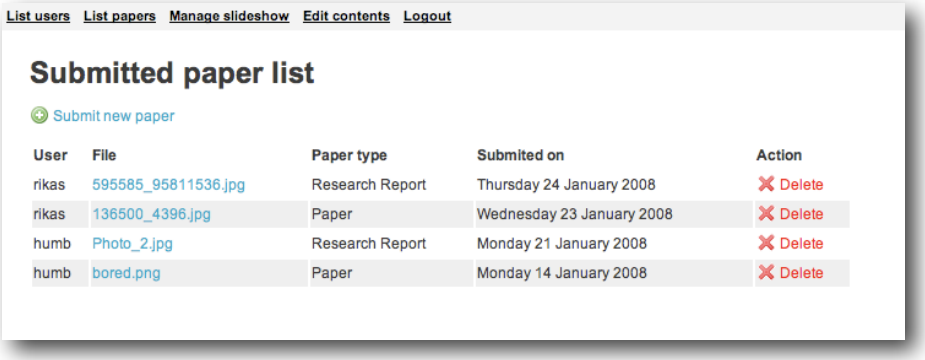


Figura 13 *– Lista de todos os documentos enviados.*

Listagem de utilizadores

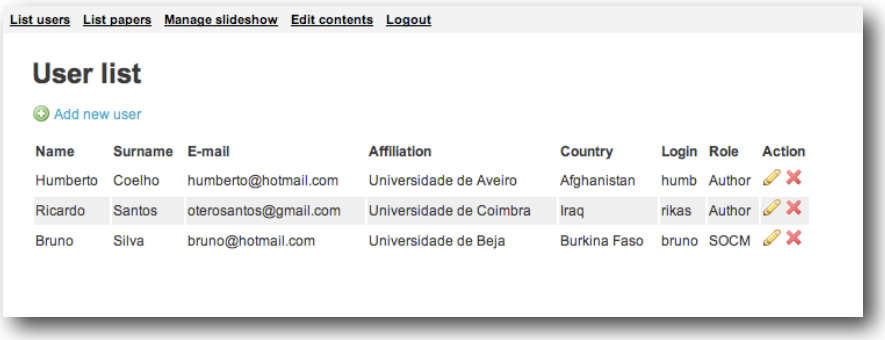


Figura 14 *– Lista de utilizadores apresentada ao administrador*

Gestão de imagens

A gestão de imagens para o *slideshow* é apresentado na figura 15. Na parte superior o utilizador pode enviar um novo ficheiro. A lista de ficheiros guardados aparece na parte inferior com um *link* para apagar cada uma se necessário. Há ainda um *link* no fundo da página que leva o utilizador para a visualização do *slideshow* (figura 19).

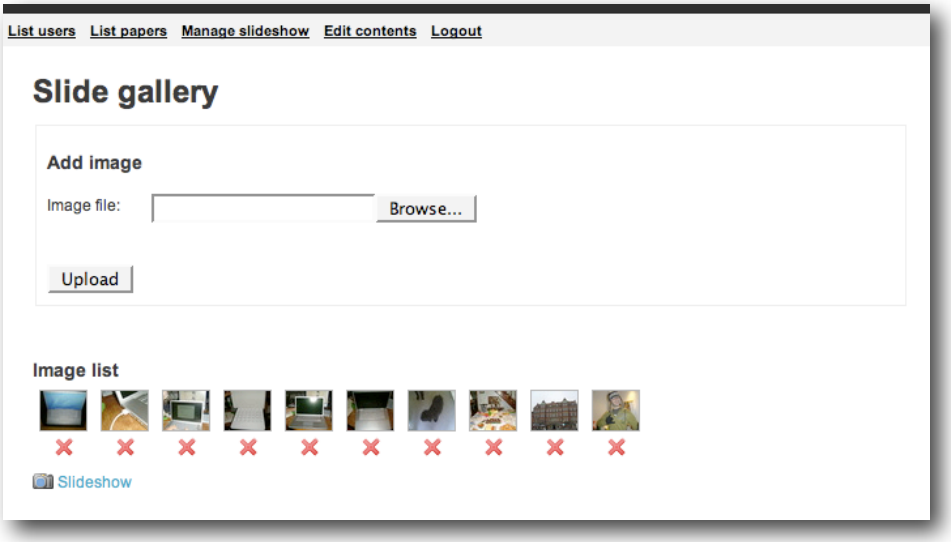


Figura 15 *– Administração do slideshow.*

Administração de conteúdos estáticos

Na figura 16 está representada uma listagem das páginas de conteúdos estáticos já criadas. Para adicionar uma nova página existe um *link* no topo da listagem. O formulário da figura 17, de edição de conteúdos estáticos é em tudo semelhante ao de adição de uma nova página com excepção de já serem apresentados os conteúdos da página em questão.

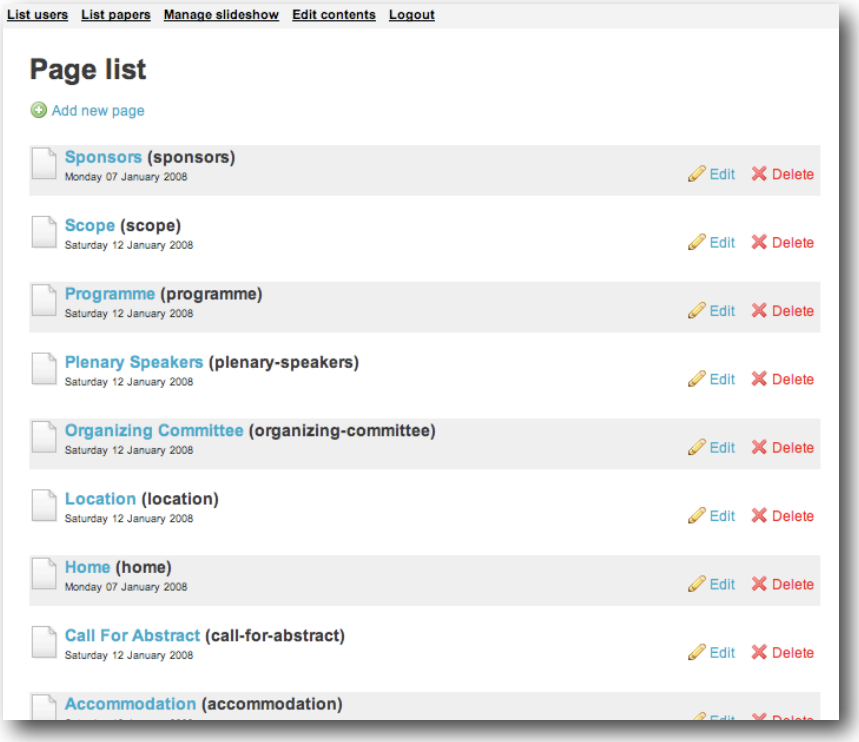


Figura 16 *– Lista de páginas com conteúdo estático que podem ser editadas.*

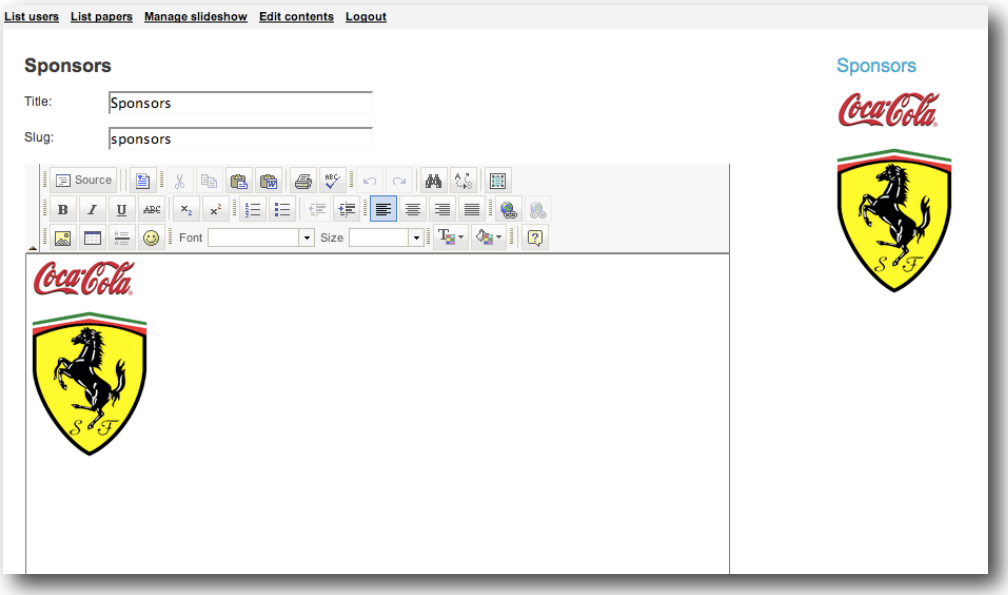


Figura 17 *– Formulário de edição de conteúdos estáticos.*

Popups de aviso

Todas as operações de eliminação de dados têm de ser antes confirmadas por um *popup* semelhante ao da figura 18. Se o utilizador cancelar então os dados não sofrem qualquer alteração. Desta forma evita-se a corrupção dos dados de modo inadvertido.

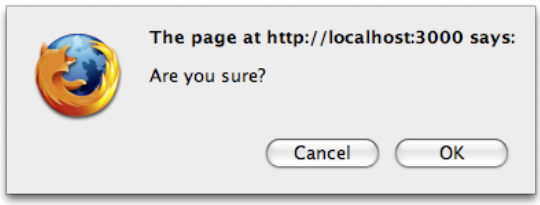


Figura 18 *– Popup de confirmação de eliminação de dados.*

Slideshow

Na figura 19 está apresentado o *slideshow* dinâmico com as imagens adicionadas através do sistema apresentado na figura 15.

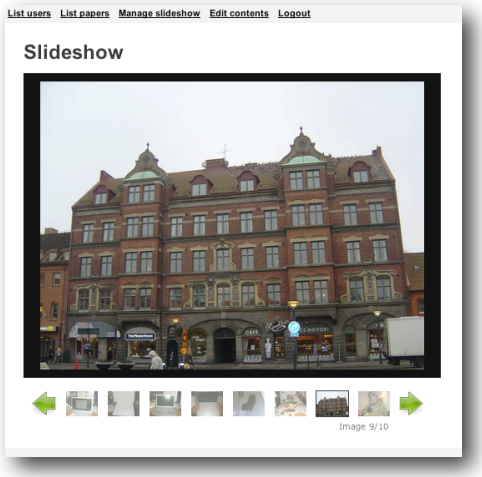


Figura 19 *– Slideshow dinâmico.*

Anexo C : Anúncios – Análise de Requisitos

1. Tabela de requisitos

Identificação	Nome	Descrição
R01	Idioma da aplicação	Toda a aplicação deve estar em português, incluindo as datas, erros e *popups*.
R02	Inserção de datas	Todas as datas devem ter ajudas visuais para a sua inserção.
R03	Perfis de utilizador	Na aplicação haverá diferentes perfis de utilizador, estando detalhados na tabela 2.
R04	Registo obrigatório	Só depois de feito um registo é que um utilizador se pode autenticar no sistema.
R05	Autenticação	Para visualização de conteúdos privados é necessária uma autenticação prévia. A autenticação é feita por *login* e *password*.
R06	Recuperar password	Os utilizadores têm de poder recuperar a *password* no sistema.
R07	Editar perfil	Todos os utilizadores registados podem editar o seu perfil, juntando uma fotografia se desejarem.
R08	*Slideshow* de imagens	Todos os anúncios poderão ter um *slideshow* de imagens associadas ao anúncio.
R09	Formato das imagens	O sistema deve aceitar imagens nos seguintes formatos: PNG, GIF e JPEG. As imagens de perfil de utilizador são guardadas apenas em disco com dimensão máximas de 100x100px. As associadas a um anúncio são guardadas em duas cópias no disco. Uma com dimensões máximas de 110x88px para *thumbnail* e outra 220x176px.
R10	Ordenação das imagens	Quando editado um anúncio deve poder ordenar-se as imagens de modo a destinar uma principal, que aparece identificativa do anúncio em listagens.
R11	Adicionar imagens	O número máximo de imagens por anúncio é cinco. Um anúncio pode não ter nenhuma imagem associada.
R12	Paginações	A listagem de anúncios e utilizadores deve ser paginada de modo a não criar *scrolls* demasiadamente grandes no *browser*.
R13	Interface	A interface deve ser cuidada, simples e fácil de usar.
R14	Validar anúncio	Um anúncio só será válido quando for activado através de um *Gateway* SMS.

Identificação	Nome	Descrição
R15	Expiração de anúncio	Todos os anúncios terão uma data limite, para além da qual expiram e será necessário fazer nova activação. Se não for feita essa activação então serão apagados ao fim de algum tempo.

Tabela 3 *– Tabela de requisitos da aplicação.*

Nome	Permissões
User	Pode editar o perfil, adicionar anúncios, listar/editar/validar os anúncios submetidos e adicionar comentários.
Admin	Tem todas as permissões dos utilizadores registados e pode gerir utilizadores. Pode também editar/apagar todos os anúncios de qualquer utilizador.

Tabela 4 *– Perfis de utilizador.*

2. Diagramas de Casos de Uso

Nos diagramas seguintes apresentam-se requisitos funcionais do sistema de anúncios. Todos os utilizadores podem criar anúncios, sendo que para clarificação dos diagramas, assumem um papel de "anunciantes" quando são tarefas relativas apenas aos seus anúncios.

2.1. Registo e autenticação

Todos os utilizadores registados têm acesso a conteúdo privado. O utilizador pode ainda alterar todos os dados constantes no seu perfil quando bem entender. A informação dos utilizadores é guardada em base de dados.

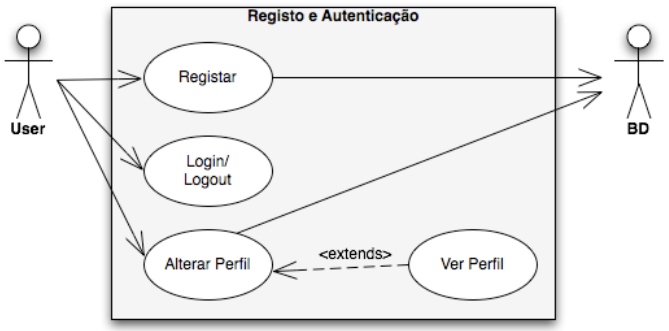


Figura 20 *– Diagrama de Casos de Uso para o registo e autenticação de um utilizador.*

2.2. Recuperação de password

Para recuperação de *password* é enviado um email com uma *hash* gerada automaticamente, que leva a um endereço usando essa *hash*, onde o utilizador pode redefinir a sua palavra-passe. Esta funcionalidade está associada a um *link* com o nome "Esqueceu a sua password?".

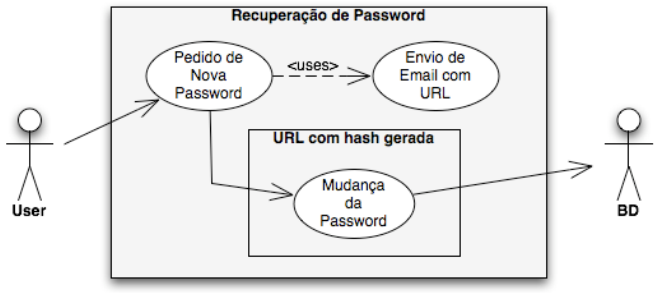


Figura 21 *– Diagrama de Casos de Uso para a recuperação de password com recurso a um URL*

2.3. Gestão de um anúncio

Para validação do anúncio tem de ser enviada uma SMS de forma a que este passe a ser visível na listagem de anúncios. Um anúncio deverá ter uma validade que quando finda exige envio de nova SMS.

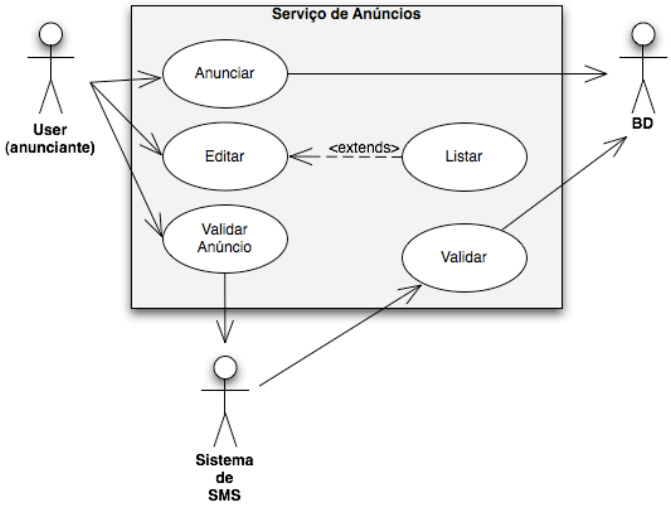


Figura 22 *– Diagrama de Casos de Uso para a publicação, consulta e validação de um anúncio.*

2.4. Aluguer e comentários

Quando um utilizador tenciona fazer um aluguer o anunciante desse deve receber uma notificação acerca da intenção do primeiro, de modo a ser mais fácil de controlar. Preferencialmente todos os utilizadores devem ter o seu contacto no perfil de modo a haver contacto directo.

Quando efectua uma pesquisa o utilizador deve ser capaz de filtrar os resultados por diversos parâmetros. Mas esta filtragem deve ser opcional.

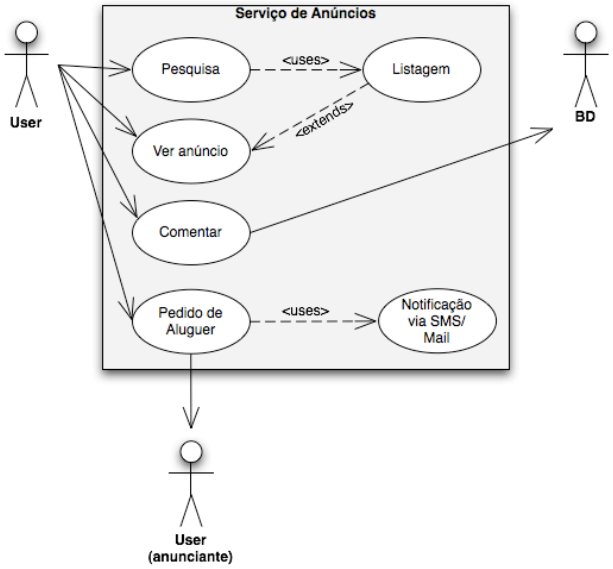


Figura 23 *– Diagrama de Casos de Uso para a adição de comentários e aluguer.*

3. Modelo de dados

Nota: nenhuma das relações entre entidades, patente na figura, devem estar definidas na base de dados, uma vez que o Ruby on Rails as faz automaticamente consoante a sua definição nos modelos em Ruby. A pluralização das tabelas é feita automaticamente com a criação dos modelos.

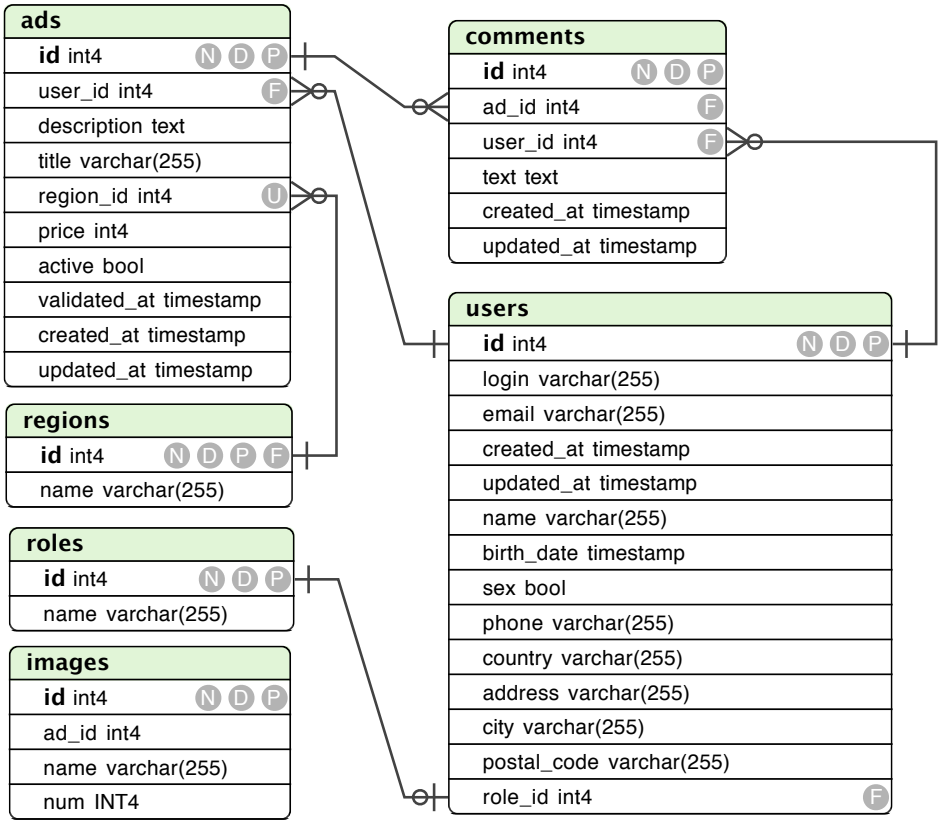


Figura 24 *– Modelo de dados da aplicação de anúncios*

Anexo D : Anúncios – Protótipo

Aspecto geral da aplicação

Na figura 25 e 26 está o aspecto geral da aplicação. Na primeira não foi feita autenticação e na segunda o utilizador está autenticado.

Figura 25 *– Aspecto geral da aplicação sem autenticação do utilizador.*

Figura 26 *– Aspecto geral da aplicação para um utilizador autenticado*

Registo

Para efectuar o registo no sistema o utilizador dispõe de um formulário como o da figura 27. Este formulário encontra-se dividido em duas partes: dados pessoas e dados de acesso.

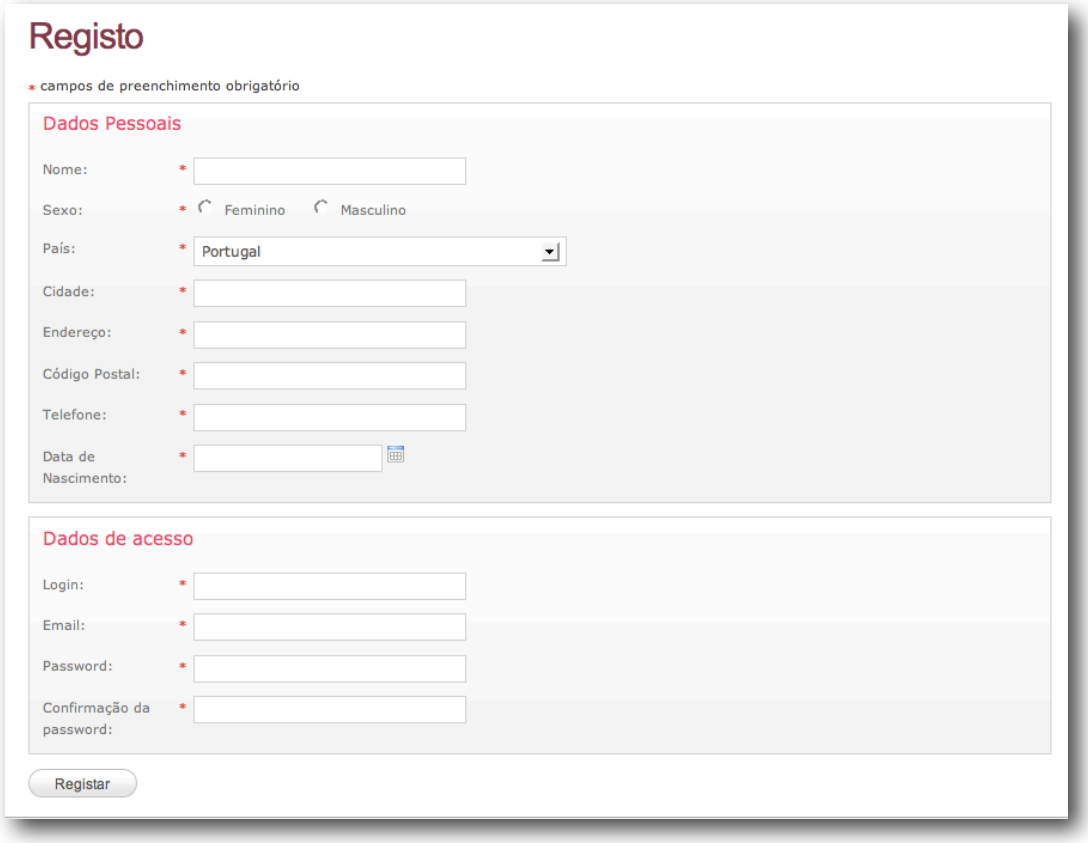


Figura 27 *– Formulário de registo*

Autenticação

Na figura 28 está representado o formulário de autenticação. Este formulário pede o email com que o utilizador se registou e *password*. No fundo aparece o *link* para o sistema de recuperação da *password*.

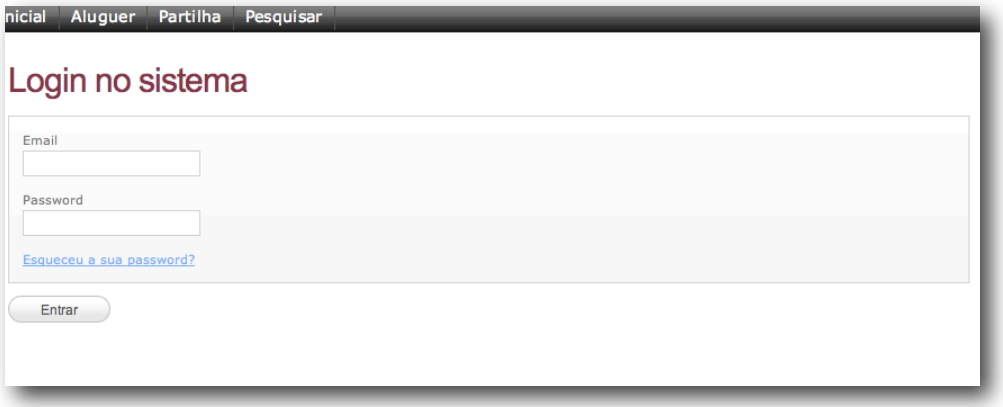


Figura 28 *– Formulário de autenticação no sistema.*

Recuperação da password

Para recuperação da *password* primeiro o utilizador deve indicar o email de registo no formulário da figura 29 de modo a receber um email com um *link* para o formulário da figura 30 onde poderá indicar a nova *password*.

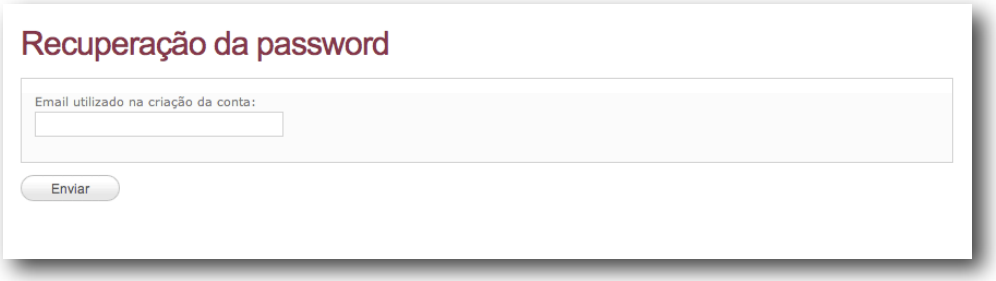


Figura 29 *– Formulário de recuperação de password.*

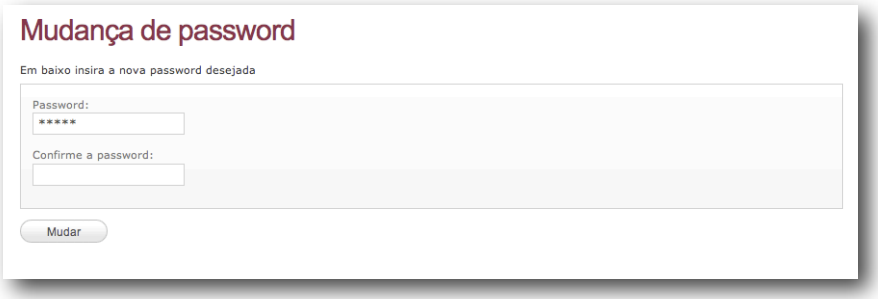


Figura 30 *– Formulário para mudança da password.*

Gestão de utilizadores

A listagem de utilizadores na figura 31 permite que o administrador edite ou apague qualquer utilizador registado no sistema. O formulário de edição de um utilizador está presente na figura 32. É o mesmo formulário que é apresentado a um utilizador quando edita o seu próprio perfil.

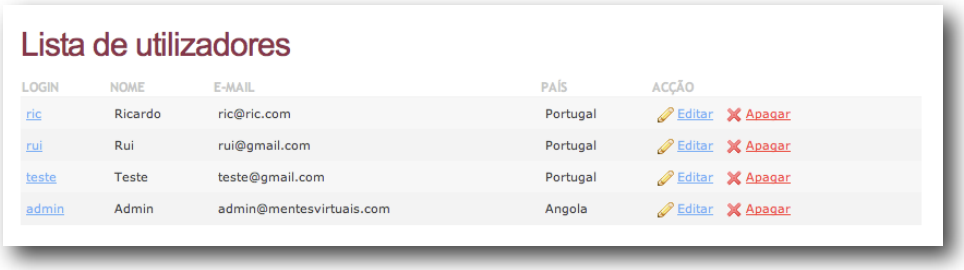


Figura 31 *– Lista de utilizadores registados.*

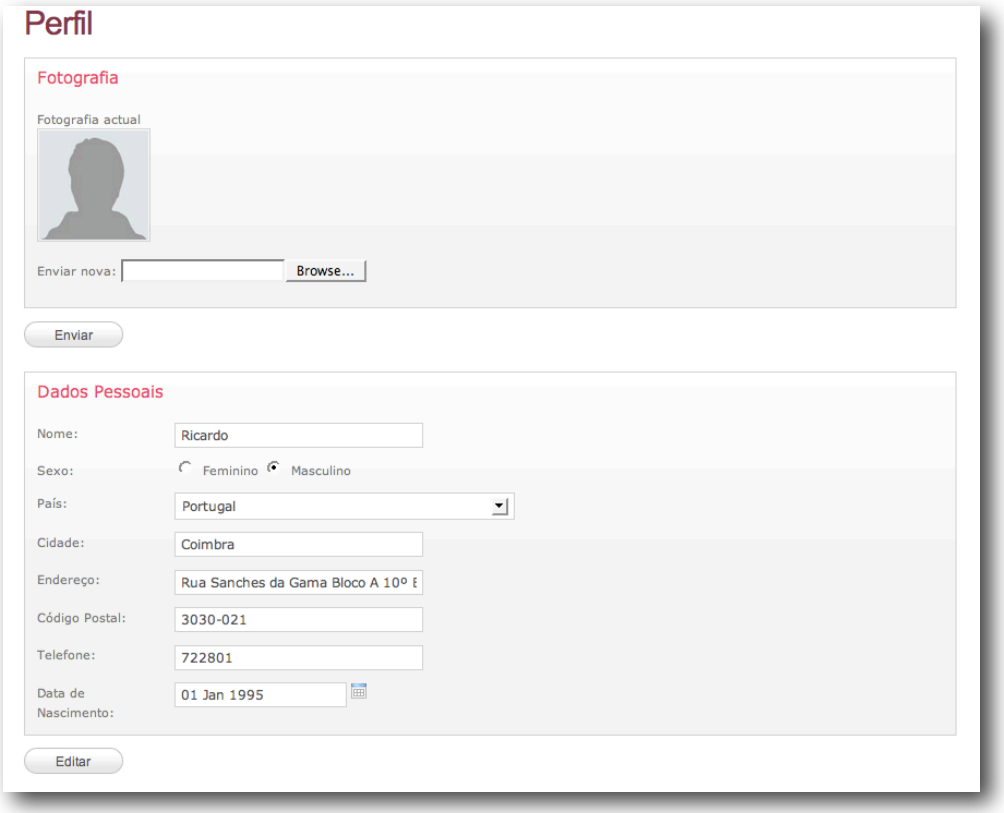


Figura 32 *– Formulário de edição do perfil de um utilizador.*

Para preenchimento das datas de nascimento é dada uma ajuda visual como a da figura 33.

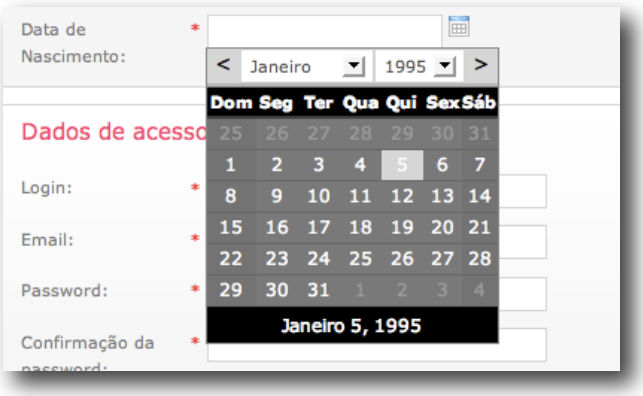


Figura 33 *– Popup de ajuda para preenchimento de datas.*

Popups de confirmação

Todas as operações de eliminação de dados têm de ser antes confirmadas por um *popup* semelhante ao da figura 34. O texto muda consoante o caso. Se o utilizador cancelar então os dados não sofrem qualquer alteração. Desta forma evita-se a corrupção dos dados de modo inadvertido.

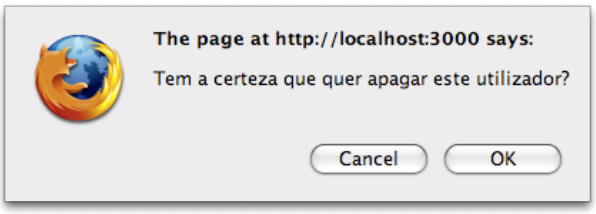


Figura 34 *– Popup típico de confirmação.*

Gestão de anúncios

O formulário da figura 35 serve para a adição de um novo anúncio. O número de imagens a associar a um anúncio pode ser incrementado com o *link* à frente do respectivo campo. No exemplo da figura apresentada o utilizador pretende associar duas imagens.

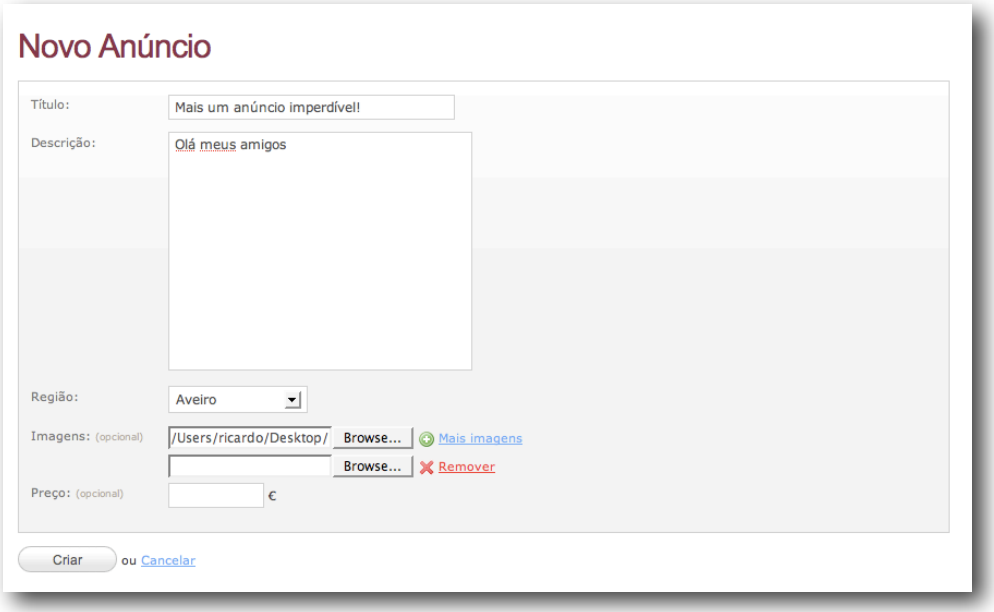


Figura 35 *– Formulário de adição de um novo anúncio.*

Na figura 36 está apresentada a página de detalhes de um anúncio. Na parte superior aparecem alguns dados como o utilizador que submeteu o anúncio, a data de submissão e a região. Na parte inferior aparece o *slideshow* com as imagens associadas e a descrição. No fundo aparecem os comentários do anúncio que não existem para o exemplo apresentado na figura mas estão visíveis na figura 40.

Figura 36 *– Detalhes de um anúncio.*

Quando um utilizador registado está autenticado aparece ainda um formulário para adição de comentário como o da figura 37.

Figura 37 *– Formulário de envio de comentários.*

Para a edição de um anúncio o utilizador dispõe de um formulário como o da figura 38, onde aparecem os dados do anúncio em questão já inseridos e a lista de imagens associadas. As imagens podem ser apagadas e reordenadas

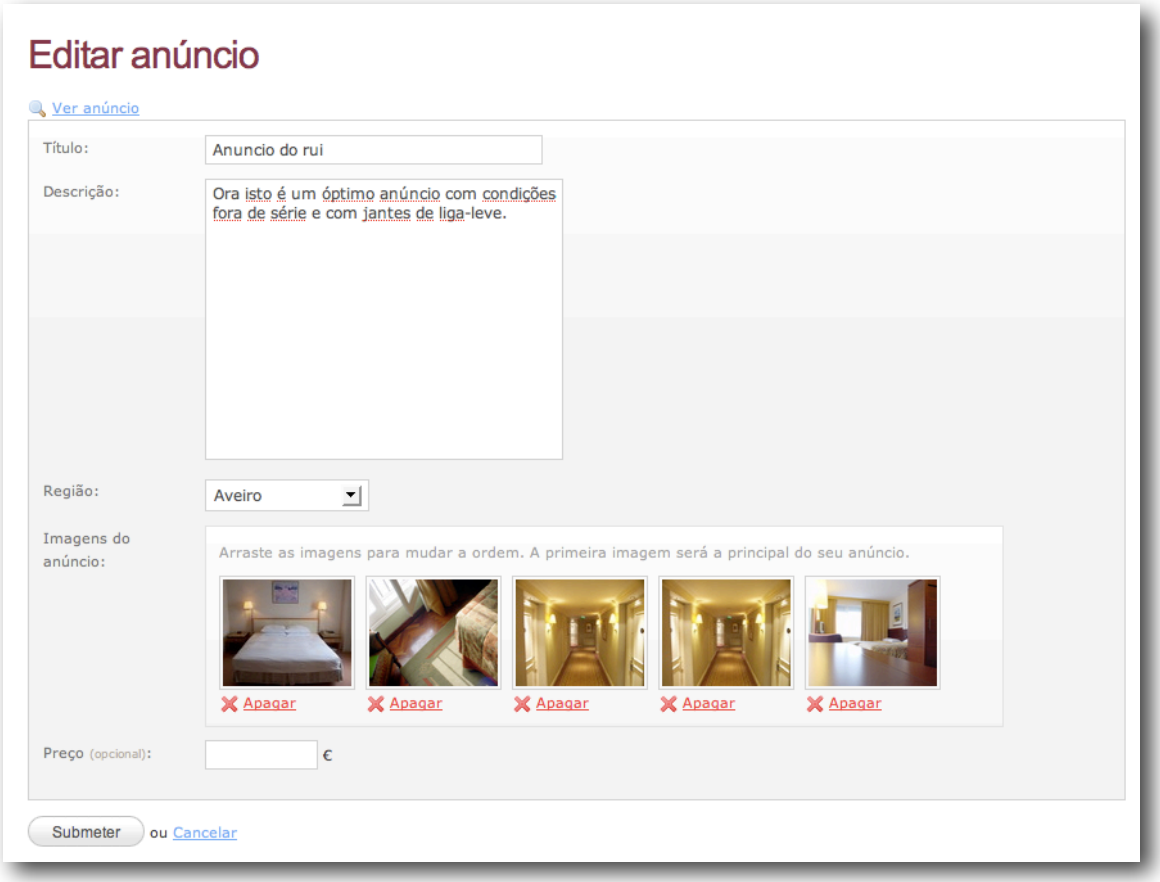


Figura 38 *– Formulário de edição de um anúncio.*

O reordenamento das imagens é feito por um mecanismo de *drag and drop* como se apresenta na figura 39. A imagem que fica na posição mais à esquerda é a identificativa do anúncio.

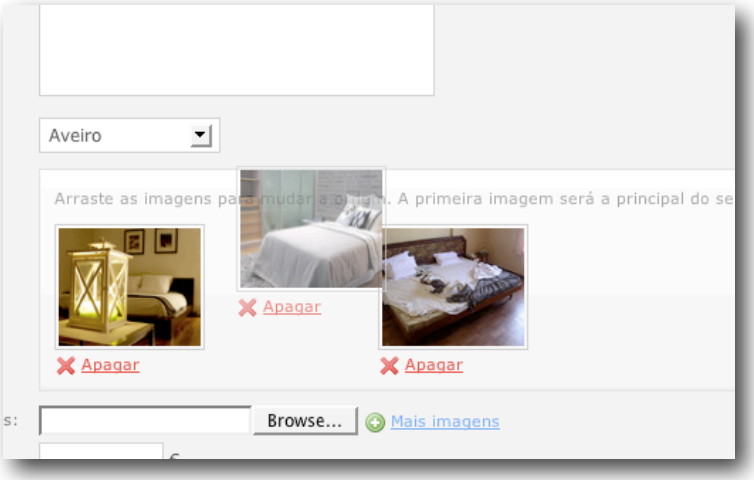


Figura 39 *– Reordenamento das imagens associadas a um anúncio. A imagem do meio está a ser arrastada para outra posição.*

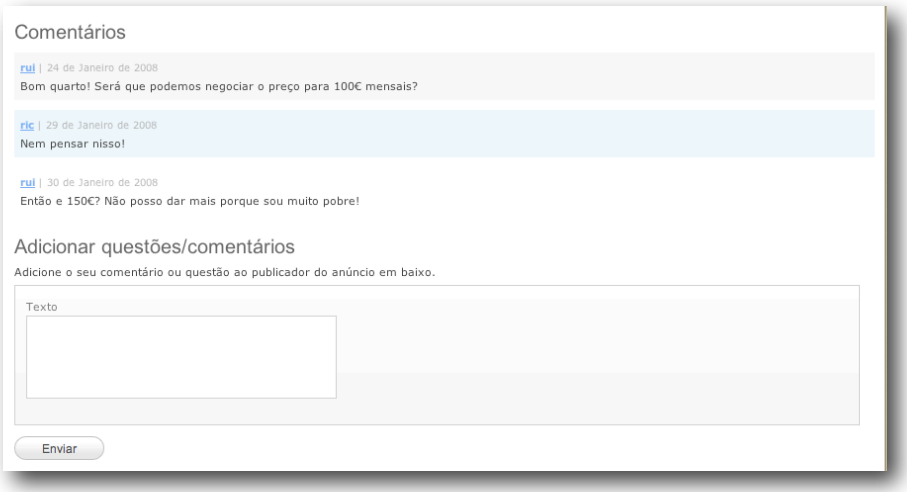


Figura 40 *– Lista de comentários de um anúncio.*

Pesquisa de anúncios

A pesquisa de anúncios e feita com o formulário da figura 41. À esquerda foi enviado o pedido de AJAX e à direita foram recebidos os resultados, apresentados como na figura 42.

Figura 41 *– Pesquisa de anúncios.*

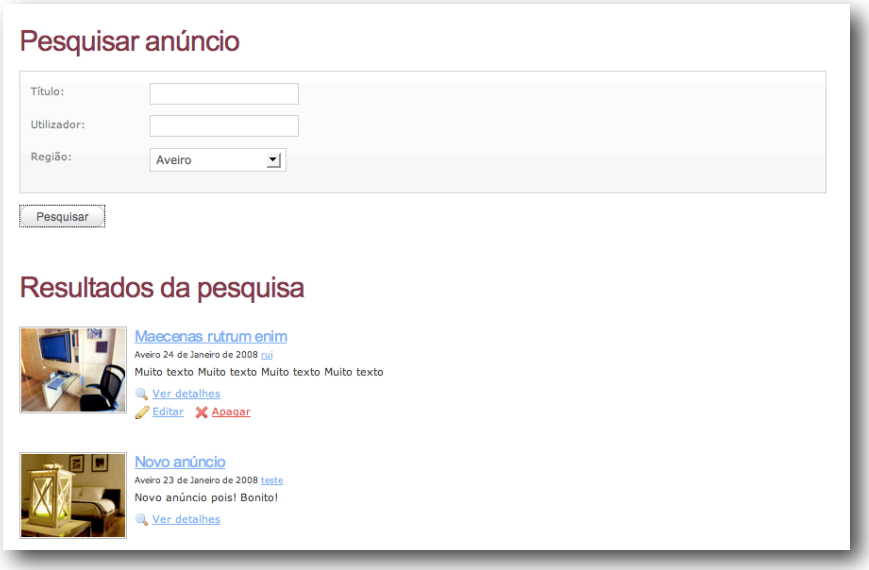


Figura 42 *– Resultados de uma pesquisa com a lista de anúncios.*

Anexo E : Plano do Projecto

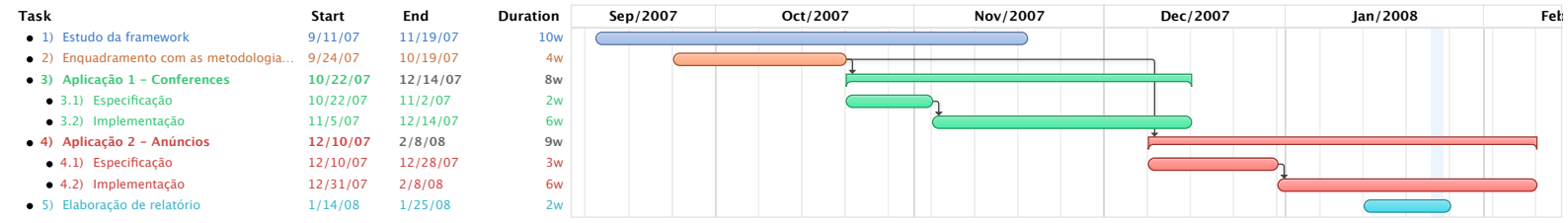


Figura 43 – *Plano inicialmente proposto.*

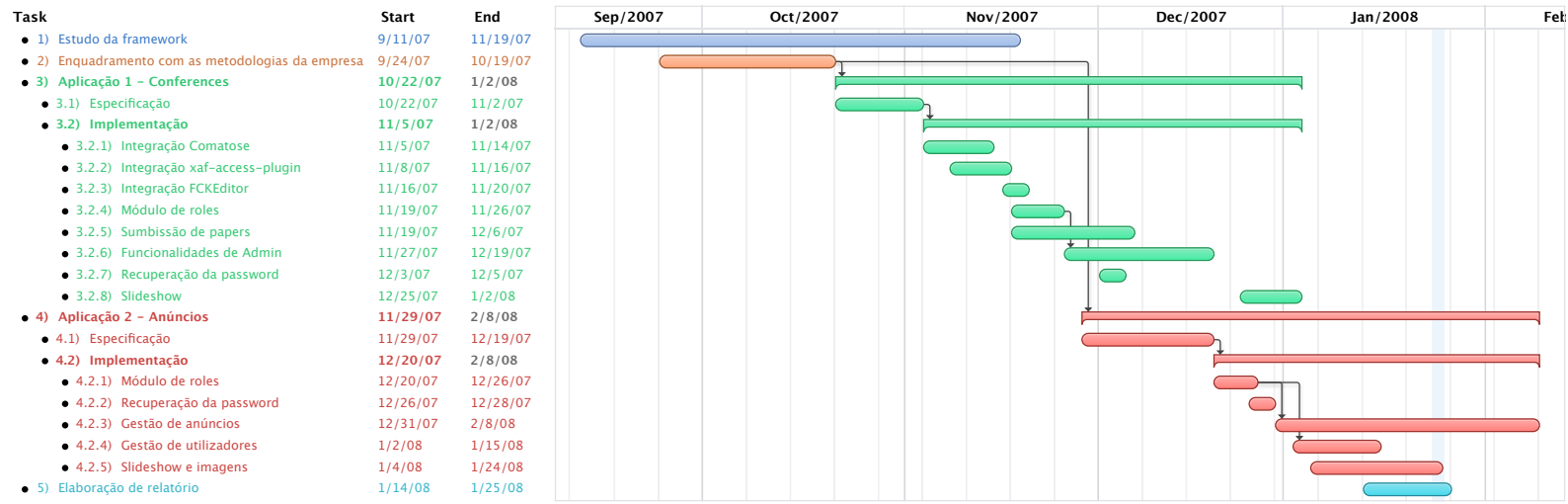


Figura 44 – *Plano detalhado do trabalho realizado.*

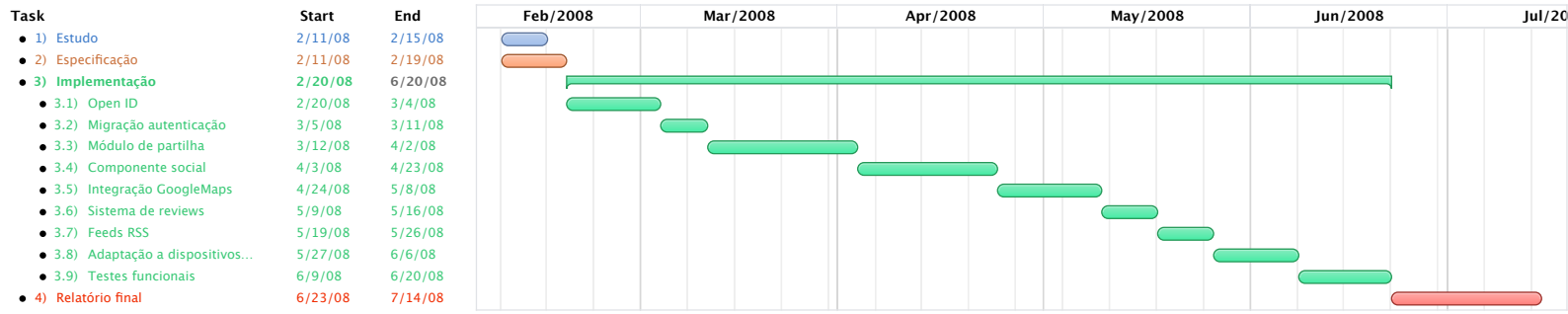


Figura 45 – *Plano de trabalhos proposto para a segunda fase do estágio.*