Universidade de Coimbra

Relatório de estágio

Mestrado em Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Departamento de Engenharia Informática

Plataformas de divulgação web e móveis

Autor:
Tiago Fael Matos

Orientadores:
Henrique Madeira
António Jorge Cardoso

Juri:
Fernando Penousal Machado
António José Nunes Mendes

August 30, 2012

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens de código

Lista de siglas

Android é o sistema operativo para dispositivos móveis desenvolvido pela Google Inc..

Apache Tomcat é o servidor aplicacional baseado em Java, da Apache Corp..

Apache Web Server é o software que fornece o serviço HTTP mais comum nos sistemas Unix e Linux, desenvolvido pela empresa Apache.

app é sinónimo de aplicação móvel.

App Store é a loja de aplicações móveis para *iOS* da *Apple Inc.*.

beta é uma fase do processo desenvolvimento em que o software desenvolvido, apesar de funcional, ainda necessita de testes extensivos.

branch é um conjunto de objectos que são derivados de outro conjunto de objectos igualmente geridos em sistemas de controlo de versões e que são modificados paralelamente.

cache é o processo que permite armazenar determinada informação de forma a que os próximos acessos à mesma sejam mais rápidos.

cache DNS é o sistema que efectua o *caching* de informação *DNS*.

checkout é a nomenclatura utilizada para a operação de obtenção da versão actual de um projecto gerido num sistema de controlo de versões.

controlador é a componente do MVC que trata das funções internas de manipulação dos dados da base de dados e de população da informação nas vistas.

cron é uma ferramenta que permite temporizar a execução periódica de scripts em sistemas Unix e Linux.

downtime é o tempo que uma plataforma/serviço se encontra inacessível.

dump , no contexto das bases de dados, é o processo que transforma uma ou mais bases de dados em *scripts* que podem ser executadas noutro sistema, reconstruindo uma réplica exacta da base de dados no novo sistema.

eBook é um livro digital, correspondendo a formatos como, por exemplo, *PDF*.

Flash é uma plataforma que permite adicionas interactividade a uma página web, desenvolvida pela Adobe Systems Inc. No âmbito deste trabalho é utilizado apenas para reprodução de conteúdos multimédia (player Flash).

framework é um conjunto de bibliotecas que fornecem funcionalidade genérica para facilitar a resolução de determinados problemas.

Freemarker é uma ferramenta que facilita a geração de texto utilizando templates.

Genshi é uma biblioteca que permite gerar páginas web dinâmicas a partir de templates desenvolvidos utilizando uma mistura de *XML*, (X)HTML e Python.

HTTP request é um pedido efectuado a um servidor web (HTTP).

iBook é um livro digital para *iPad*, que permite componentes dinâmicas e interactivas.

iOS é o sistema operativo dos dispositivos móveis *iPhone*, *iPod Touch* e *iPad*, desenvolvido pela Apple Inc..

iPad é o *tablet* desenvolvida pela *Apple Inc.*.

iPhone é o *smartphone* desenvolvido pela *Apple Inc.*.

iPod Touch é o leitor de *MP3* com touchscreen desenvolvido pela *Apple Inc.*.

Java é uma linguagem de programação compilada para e executada numa máquina virtual, permitindo que os seus programas funcionem em múltiplas plataformas distintas.

Java Struts (ou Apache Struts) é uma framework de desenvolvimento web com Java.

JavaScript é uma linguagem de scripting utilizada para implementar funcionalidade do lado do cliente (client-side, ou browser-side).

jQuery é uma biblioteca de *JavaScript* desenvolvida para simplificar o desenvolvimento de funções *client-side* em páginas web.

lazy loading é uma *design pattern* que permite a inicialização de objectos apenas quando necessários.

loja iTunes é uma plataforma para compra de conteúdo digital da Apple Inc., acessível através do software *iTunes*.

middleware é uma componente que faz a interligação entre duas aplicações.

minificação é o processo que consistem em retirar do código fonte toda a informação que não interessa à execução do mesmo, como comentários e espaçamento desencessário.

MySQL é um sistema de gestão de bases de dados gratuito.

obfuscação é o processo que consistem em tornar código fonte difícil de interpretar.

Objective-C é a linguagem de programação base para o desenvolvimento de aplicações para *Mac OS X* e *iOS*.

player é, no contexto deste trabalho, uma peça de software que permite reproduzir conteúdos multimédia.

Pylons é uma framework open source para desenvolvimento web utilizando Python.

Python é uma linguagem de programação de alto nível com características que permitem facilitar a leitura de código através da indentação de código obrigatória.

script corresponde a um pequeno programa utilizado para automatizar determinados processos que normalmente teriam que ser realizados passo a passo por uma pessoa.

SQLAlchemy é uma biblioteca de mapeamento de entidades de bases de dados em objectos Python.

streaming é, no contexto deste trabalho, o envio contínuo de um conteúdo de multimédia permitindo que o utilizador final o visualize sem ter que esperar que este seja carregado por completo.

template é um modelo utilizado para a criação de vistas dinâmicas (como páginas web ou vistas de aplicações em dispositivos móveis).

tradeoff corresponde a uma situação em que se opta por perder uma qualidade em troca de outra.

Universal app é, no âmbito de aplicações para *iOS*, uma aplicação que funciona em todos os dispositivos móveis com esse sistema (*iPod Touch*, *iPhone* e *iPad*), adequando o interface para cada uma delas (mais exactamente, um interface distinto para *iPhone* e *iPod Touch* e outro para *iPad*).

update é a nomenclatura utilizada para a operação de actualização de um projecto local (ao qual se fez *checkout* anteriormente), com as modificações registadas no sistema de controlo de versões.

Velosurf é uma biblioteca simplista para mapeamento de dados de uma base de dados em objectos *Java*.

XCode é um IDE distribuído pela *Apple Inc.* para o desenvolvimento de aplicações para *Mac OS X* e *iOS*.

Lista de siglas

AJAX *Asynchronous JavaScript and XML.*

API *Application Programming Interface.*

CIUC Centro de Informática da UC.

CSS *Cascading Style Sheets.*

DAO *Data Access Object.*

DNS *Domain Name System.*

ER *Entity-Relationship.*

FTP *File Transfer Protocol.*

GUI *Graphic User Interface.*

HTTP *HyperText Transfer Protocol.*

i18n *internationalization* - internacionalização (tradução).

JPA *Java Persistence API.*

JSON *JavaScript Object Notation.*

MVC *Model-View-Controller.*

OGNL *Object-Graph Navigation Language.*

PDF *Portable Document Format.*

RSS *Rich Site Summary*, também conhecido como *Really Simple Syndication.*

SDK *Software Development Kit.*

SQL *Structured Query Language.*

TAGV *Teatro Académico Gil Vicente.*

UC Universidade de Coimbra.

UCV Canal de televisão web da UC.

URL *Uniform resource locator*.

WebDAV *Web Distributed Authoring and Versioning*.

XHTML *Extensible HTML*.

XML *Extensible Markup Language*.

Resumo

Este trabalho trata do desenvolvimento e manutenção de um conjunto de plataformas para divulgação e distribuição de conteúdos culturais e científicos em formatos multimédia, e de uma agenda de divulgação de iniciativas culturais e científicas, associado à estratégia de divulgação de informação e conhecimento implementada por instituições do ensino superior.

Pretende-se também desenvolver meios de apoio a essas plataformas que permitam o alargamento da divulgação dos seus conteúdos a outros canais de comunicação como redes sociais, dispositivos móveis e outros meios que facilitem a subscrição da informação.

Palavras Chave:
"divulgação", "plataformas web", "frameworks web", "plataformas móveis", "conteúdos", "multimédia", "redes sociais", "eventos", "calendário"

Abstract

This paper explores the development of a set of platforms for the promotion and distribution of cultural and scientific content in multimedia formats, and of an agenda for the promotion of scientific and cultural initiatives, coupled with the strategy for disseminating information and knowledge implemented by higher education institutions.

The aim is to develop ways to support those platforms that allow the extension of the promotion of its contents to other communication channels such as social networks, mobile devices and others in order to facilitate the subscription of information.

Keywords:
"promotion", "distribution", "web platforms", "web frameworks", "mobile", "media content", "media", "multimedia", "social networks", "events", "calendar"

Agradecimentos

Desejo expressar os meus mais sinceros agradecimentos:

Ao professor António Jorge Cardoso, meu orientador, pelo apoio e paciência prestados no desenvolvimento deste relatório de estágio.

Aos vice-reitores Henrique Madeira, Joaquim Ramos de Carvalho e Clara Almeida Santos, pela motivação e confiança depositada no meu trabalho.

Ao meu colega Eduardo Perdido, pelo acompanhamento e contribuição crítica.

Aos meus pais e avós pelo contínuo apoio ao longo de todo o curso.

1. Introdução

Os projectos expostos neste relatório encontram-se inseridos no âmbito do estágio académico do ano lectivo 2011/2012, pertencente ao Mestrado em Engenharia Informática do Departamento de engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

As tarefas realizadas pelo estagiário envolveram o desenvolvimento de funcionalidades em torno de plataformas de divulgação de conteúdos multimédia e de eventos, tendo em vista suportar uma parte importante das necessidades de divulgação em torno da Universidade de Coimbra e da própria cidade.

O estágio envolve três projectos:

UCV - Canal de televisão web da Universidade de Coimbra[1]

Projecto do *iTunes U* da Universidade de Coimbra[2]

Agenda7 - Agenda cultural da cidade de Coimbra[3]

Estes projectos serão apresentados com maior detalhe nos capítulos 2, 3 e 4, respectivamente.

1.1 Contexto

Qualquer instituição de ensino superior tem necessidade de se projetar no mundo do conhecimento de forma a potenciar a sua missão principal. A Universidade de Coimbra não foge a esta regra. Sendo uma organização complexa, de prestígio internacional que procura acompanhar de perto as rápidas mudanças inerentes à sociedade da informação e do conhecimento, assume, na missão definida no seu *Plano Estratégico*[4] que "*é uma instituição de criação, análise crítica, transmissão e difusão de cultura, ciência e de tecnologia que, através da investigação, do ensino e da prestação de serviços à comunidade, contribui para o desenvolvimento económico e social, para a defesa do ambiente, para a promoção da justiça social e da cidadania esclarecida e responsável e para a consolidação da soberania assente no conhecimento*"[5].

[1] http://ucv.uc.pt
[2] http://itunes.apple.com/pt/institution/university-of-coimbra/id390614860
[3] http://agenda7.uc.pt/
[4] http://www.uc.pt/planeamento/PE_WEB_09122011.pdf
[5] Plano Estratégico da UC, 2011-2015, pág. 15

Ao definir esta missão como ponto de partida do seu **Plano Estratégico**, assumindo que tem o dever de contribuir para potenciar "*o desenvolvimento de atividades de ligação à sociedade, designadamente de difusão e transferência de conhecimento, assim como de valorização económica do conhecimento científico*", a UC apresenta desde logo, como sua necessidade estratégica, o desenvolvimento de sistemas de informação capazes de garantir a sua competitividade. Tal é visível no **Quadro de Definição Estratégica** da Universidade de Coimbra, apresentado na figura 1.1, e e que vai enquadrar alguns dos aspectos referidos neste relatório. Na análise do *Plano Estratégico*, constata-se a existência de dois grupos de pilares estratégicos: os de **Missão** e os de **Recursos**. Interessa analisar o primeiro grupo, subdividido em três pilares:

- Investigação
- Ensino
- Transferência de Conhecimento

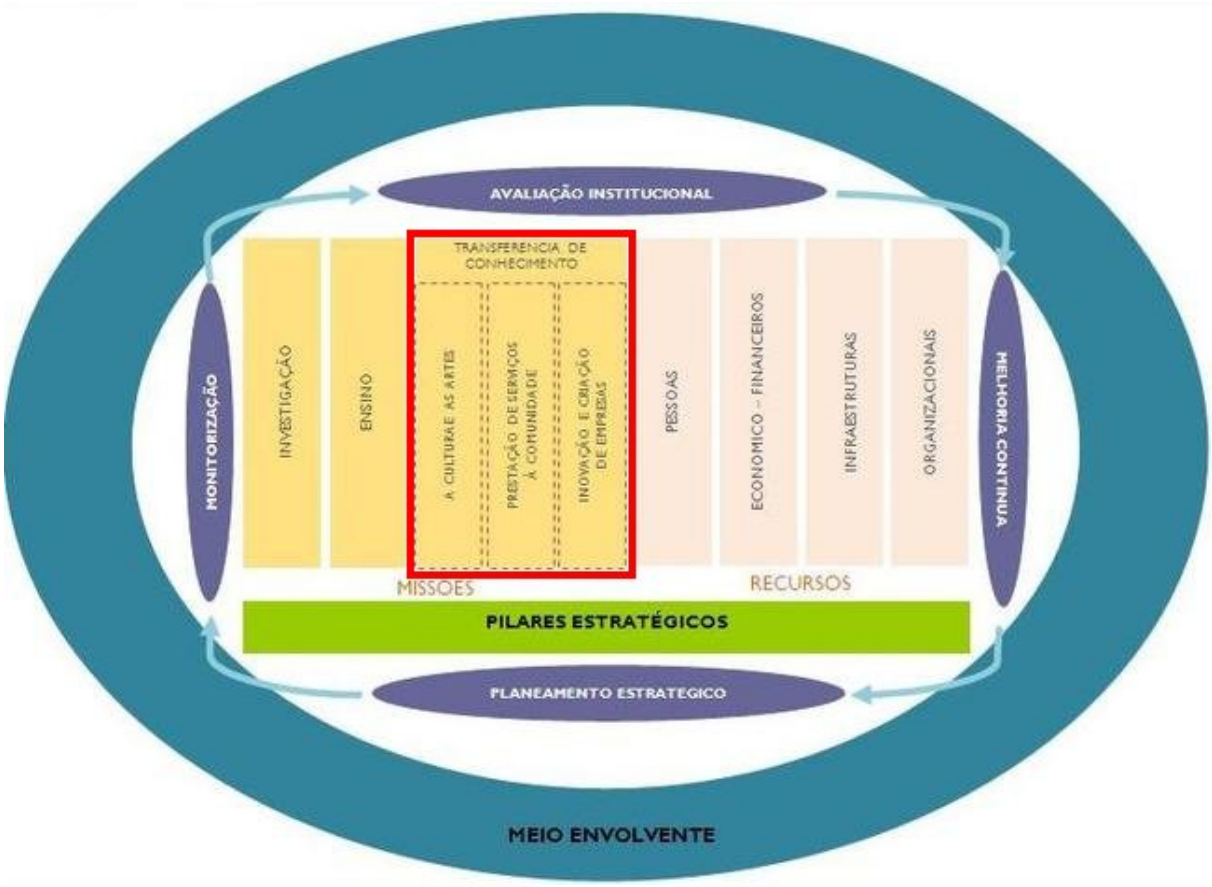


Figura 1.1: Quadro de Definição Estratégica da UC, com o pilar onde se insere este estágio destacado a vermelho

O pilar da **Transferência de Conhecimento** tem como objectivo "fortalecer o papel motor da Universidade de Coimbra no desenvolvimento económico, social e cultural e incrementar a sua capacidade de intervenção, nacional e internacional, através da intensificação da ligação à sociedade e meio envolvente e do reforço da transferência de conhecimento, valorizando o seu valor acrescentado"[6]. Este pilar é composto pelos três sub-pilares:

[6]Plano Estratégico da UC, 2011-2015, pág. 31

- A Cultura e as Artes
- Prestação de Serviços à Comunidade
- Inovação e Criação de Empresas

Estes sub-pilares representam diferentes vertentes do pilar da **Transferência de Conhecimento** e correspondem ao conjunto de iniciativas estratégicas:

1. desenvolver uma política cultural ativa e responsável, colocando a UC no mapa nacional e internacional, através do fomento da atividade cultural, artística e desportiva;
2. promover a língua, a cultura e a cidadania lusófonas;
3. promover uma cultura de criatividade e inovação, de empreendedorismo e de espírito crítico;
4. reforçar o apoio à transferência de conhecimento, à gestão da propriedade intelectual e à criação de empresas;
5. posicionar a UC como entidade catalisadora da transferência de conhecimentos;
6. posicionar a UC como referência internacional de inovação, potenciando a participação em redes internacionais.

Contribuir para o cumprimento da missão da UC passa por desenvolver projectos que concretizem estas iniciativas estratégicas. Assim, no âmbito deste estágio cujo relatório se apresenta foram desenvolvidos alguns dos projectos que permitem a concretização de parte destas iniciativas estratégicas.

O processo de divulgação associado à iniciativa de desenvolver uma política cultural activa e responsável através do fomento de diversas actividades (**1**) já era um processo realizado pela Universidade de Coimbra e por outras entidades externas pertencentes ao panorama cultural da cidade de Coimbra, mas essa divulgação era realizada de uma forma segmentada, estando cada um dos órgãos responsáveis pela divulgação dos seus eventos através dos meios que lhes estavam disponíveis. Com o objectivo de facilitar o processo de propagação de eventos organizados pelas diversas entidades da cidade de Coimbra, foi decidido criar uma agenda cultural global para a cidade de Coimbra, cujos eventos eram publicados directamente por cada entidade responsável. Desta forma evita-se a necessidade de contratação de recursos humanos para publicação de eventos dentro da universidade para receber e concretizar a publicação dos diversos eventos que surgem, o que permite escalar a plataforma com maior facilidade. Por outro lado, este processo pode ter outras implicações, como a variação dos estilos utilizados pelos vários parceiros para a publicação, levando a uma apresentação da plataforma menos consistente.

O *iTunes U* é uma plataforma de gestão e distribuição de conteúdos educacionais de video, audio e *eBooks*, cujo principal objectivo é facilitar a distribuição de conteúdos académicos a estudantes e pessoas interessadas nos mais diversos assuntos de teor académico em todo o mundo. É uma rede internacional de referência que tem o apoio das maiores universidades mundiais e que permite agora a inserção de conteúdos na língua portuguesa, possibilitando o posicionamento da UC como referência internacional, potenciando a participação em redes internacionais (**6**) e como entidade catalisadora da transferência de conhecimentos (**5**).

Também relacionado com o desenvolvimento de uma política cultural activa (**1**), promoção da língua, a cultura e a cidadania lusófonas (**2**) e reforço do apoio à transferência de conhecimento (**4**) está a criação de uma equipa de produção de conteúdos associada a uma televisão web da Universidade de Coimbra, que permite apoiar o desenvolvimento de conteúdos como suporte ao processo de divulgação cultural e de transferência de conhecimento através da produção de conteúdos para as plataformas referidas anteriormente, fornecendo ainda um serviço de produção de conteúdos para divulgação externa e permitindo também a divulgação das iniciativas através do canal de televisão web, que também funciona como *hub* central para permitir, com alguma facilidade, que esses conteúdos sejam divulgados noutros sites.

Associado ainda à estratégia de divulgação está a implementação de meios que permitam facilitar e/ou explorar outros meios de comunicação como redes sociais e plataformas móveis. Desta forma é também necessário ter em conta as necessidades destes meios aquando da implementação destas plataformas.

1.2 Objectivos

Para a Universidade de Coimbra, os objectivos do trabalho desenvolvido neste estágio são:

- Garantir a manutenção e contínua evolução das plataformas de suporte aos três projectos;
- Expandir ou possibilitar a expansão dos canais de divulgação dos conteúdos destas plataformas;

Para o estagiário, os objectivos principais são a consolidação de conhecimentos no desenvolvimento de aplicações móveis e aplicações orientadas para a web, ganhando experiência em diversas tecnologias de desenvolvimento para dispositivos móveis e plataformas web.

Nos capítulos **UCV** (2), **iTunes U** (3), *Agenda7* (4) e **UCV Mobile** (5) serão apresentados os objectivos específicos de cada uma destas componentes.

1.3 Riscos

O trabalho desenvolvido pelo estagiário na Universidade de Coimbra estende-se por outras áreas que ultrapassam a mera manutenção das plataformas indicadas, sendo também o responsável por dar apoio técnico a vários outros projectos desenvolvidos. Como tal, as metas temporais do trabalho planeado podem ser facilmente afectadas pela necessidade da intervenção do estagiário noutras áreas que não estão englobadas no estágio. Desta forma, a definição do planeamento tem que ter em conta alguma flexibilidade.

2. UCV

A **UCV** é o canal de televisão web da Universidade de Coimbra. O projecto deriva da criação de um centro de produção de conteúdos, que fornece diversos serviços à comunidade relacionados com a criação de conteúdos de video e audio, empréstimo de equipamento e mesmo o suporte à divulgação dos conteúdos. Esta última necessidade implicava a implementação de um sistema que permitisse gerir e disponibilizar os conteúdos de forma acessível e simples.

A plataforma de suporte foi desenvolvida a partir de uma plataforma de gestão de conteúdos de audio e video denominada *MediaCore*[1], por já ser uma plataforma que suportava grande parte dos requisitos, e mesmo possibilitando a criação de podcasts compatíveis com a plataforma *iTunes*[2] que, apesar de não possuírem a categorização específica do *iTunes U* (por disciplinas), permitia já criar uma representação quase completa de uma colecção, bastando efectuar cópias directas e adicionar essa pequena alteração.

Esta plataforma estava instalada num servidor de empréstimo do *CIUC*, que não tinha grande capacidade de armazenamento para suportar a plataforma a longo prazo, tendo sido adquirido um servidor apropriado para o armazenamento dos mesmos.

A plataforma foi lançada oficialmente no dia 21 de Novembro de 2010.

2.1 Arquitectura

A plataforma da **UCV** é baseada na plataforma *MediaCore*, uma plataforma de gestão de conteúdos multimédia (áudio e video). A plataforma foi escolhida como a plataforma base para o desenvolvimento da **UCV** devido a já responder a um conjunto de requisitos iniciais:

Interface facilitador de uploads para utilização tanto pela equipa de produção de conteúdos, como para parceiros sem permissões de acesso ao backoffice;

Alguma facilidade de integração dos conteúdos com o *iTunes*, por já ter implementada a criação de *RSS*s compatíveis com o sistema de podcasts da plataforma;

Suportado pelos browsers de computadores pessoais mais utilizados[3], suportando, pelo menos, 95% dos browsers;

[1] http://mediacorecommunity.org
[2] http://www.apple.com/itunes/
[3] http://support.mediacore.com/customer/portal/articles/99497-what-browsers-are-supported-

Flexibilidade para implementação de novas funcionalidades, como a integração de um sistema de *streaming* de video em directo utilizando um serviço externo.

No entanto, a plataforma tal como é fornecida, não é completamente adequada à utilização pela **UCV**, sendo necessário adequar o layout das páginas para apresentar uma identidade distinta, mais apropriada para o canal de televisão da UC. Isto implica uma reformulação do interface e algumas alterações mais internas que leva a que o projecto resultante se torne num *branch* da plataforma original.

Na altura em que a plataforma foi escolhida para suportar a **UCV**, o *MediaCore* ainda se encontrava em fase *beta*, não sendo ainda reconhecida como uma plataforma concluída. A criação de um *branch* de uma plataforma *beta* implica que seja feito um acompanhamento do projecto principal, de forma a que sejam detectados problemas que tenham que ser resolvidos também no novo *branch*.

Durante o estágio aqui relatado, houve a necessidade de realizar um update à plataforma, que foi aproveitado como uma forma de reformular a imagem da **UCV** no dia do primeiro aniversário do projecto e portar toda a plataforma para o novo hardware, adquirido especificamente para suportar os projectos deste estágio, substituindo o hardware emprestado pelo *CIUC* que não tinha capacidade suficiente para suportar os conteúdos deste projectos.

2.1.1 Framework

O *MediaCore* foi desenvolvido tendo por base a *framework* para desenvolvimento web, *Pylons*[4], que é uma *framework* aberta baseada na arquitectura de desenvolvimento *MVC* e que tem por base a linguagem de programação *Python*.

A *framework* dá alguma flexibilidade na escolha das *frameworks* internas para controlo do modelo de dados e a linguagem utilizada nos *templates*. A equipa de desenvolvimento do *MediaCore* optou por *SQLAlchemy*[5] para gestão do modelo de dados e *Genshi*[6] para geração de páginas web a partir dos *templates*.

O funcionamento de um projecto desenvolvido com esta *framework* é relativamente simples (também representado na figura 2.1):

1. O servidor *HTTP* recebe o pedido e procura por conteúdo estático correspondente a este;

   Caso encontre, devolve o conteúdo ao cliente e o processo termina.

2. Caso não exista conteúdo estático, o pedido é redireccionado para ser tratado pelo *Pylons*;
3. É consultado um ficheiro de rotas para descobrir qual o *controlador* e método responsável pela resposta ao pedido;
4. É executado o método do *controlador* correspondente ao pedido, construindo a página final, normalmente preenchendo um *template*;
5. É devolvido o resultado ao utilizador;

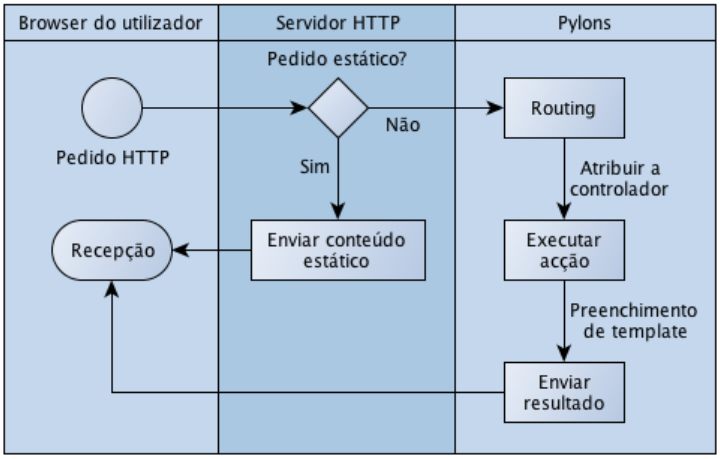


Figura 2.1: Processo de resposta a pedidos HTTP

No servidor *HTTP* utilizado, o *Apache Web Server*[7], será verificada a existência de informação estática na pasta base da configuração do projecto no servidor web. Caso essa informação não exista, o pedido é redireccionado através de um *middleware*, neste caso específico o *FastCGI*[8], que irá chamar a parte do processo correspondente ao *Pylons*.

2.1.2 Estrutura MediaCore

De forma a permitir realizar a manutenção, é necessário reconhecer certos aspectos do projecto, nomeadamente a estrutura pasta de projecto do *MediaCore*, que na realidade não é muito mais que a estrutura de um projecto *Pylons*. O conhecimento desta estrutura é necessário para que o programador consiga discernir onde se encontra o código que é necessário alterar para, por exemplo, poder implementar uma nova funcionalidade.

A pasta de base do projecto é constituida por ficheiros de configuração de contexto, o que permite que sejam criadas várias instalações da plataforma, bastando para isso, alterar a configuração para corresponder ao novo contexto. A plataforma *MediaCore* também fornece na raíz algumas ferramentas ou scripts para obter ferramentas que permitem compilar código oriundo de projectos externos, como a *Closure Library*[9] para a *obfuscação* e *minificação* do *JavaScript*, ou o *Transifex*[10], para compilar os ficheiros de localização *i18n*. Nenhuma destas ferramentas é necessária para além da fase de programação da plataforma, não sendo necessárias para que a plataforma funcione.

A organização do projecto *MediaCore* no sistema de ficheiros, que pode ser observada na figura 2.2, baseada na estrutura definida originalmente pelo *Pylons*, consiste numa pasta raíz do projecto que contém toda a configuração base necessária para o funcionamento de um projecto *Pylons*. Nesta pasta de raíz existem alguns ficheiros de configuração independentes da plataforma, que servem para configurar o interface com aplicações externas, como o *Apache Web Server* e o *MySQL*. A parte funcional que define o *MediaCore* encontra-se na pasta `mediacore` e é lá que se encontra implementada toda a plataforma.

[4]http://www.pylonsproject.org
[5]http://www.sqlalchemy.org
[6]http://genshi.edgewall.org
[7]http://httpd.apache.org
[8]http://www.fastcgi.com/
[9]http://closure-library.googlecode.com/svn/docs/index.html
[10]https://www.transifex.com

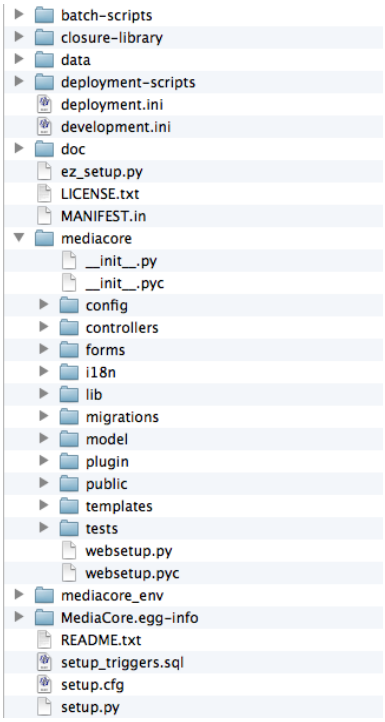


Figura 2.2: Estrutura do projecto *MediaCore* no sistema de ficheiros

A pasta `mediacore` é completamente independente do sistema onde se encontra, podendo ser sincronizada independentemente da informação que se encontra na raíz. Isto possibilita que seja efectuada a sincronização desta pasta num sistema de controlo de versões à qual se pode fazer *checkout* e *update* para qualquer sistema, dentro de uma pasta de raíz do *Pylons* já configurada para o sistema em questão. Isto permite manter a plataforma actualizada com as últimas alterações à plataforma que vão sendo desenvolvidas pelo programador.

A pasta do projecto é composta por um conjunto de sub-pastas que segmentam o projecto de forma a separar as diversas componentes distintas que correspondem a diferentes partes que servem diferentes propósitos na aplicação. As componentes/pastas que foram consideradas mais importantes são:

config contém todos os ficheiros de configuração da plataforma no âmbito do projecto. Um dos ficheiros de configuração importantes é o `routing.py`, que define o mapeamento de caminhos para acesso aos *controladores*.

controllers contém a implementação dos *controladores* que são chamados consoante o *HTTP request* realizado, tal como indicado no mapeamento de paths indicados no ficheiro `routing.py`.

i18n é pasta de traduções/localização.

lib contém as bibliotecas adicionais do *Pylons* que são necessárias ao funcionamento da plataforma, bem como outras bibliotecas que podem ser adicionadas pelos programadores para adicionar funcionalidade adicional à plataforma.

migrations contém as alterações à base de dados que têm que ser realizadas para compatiblizar a base de dados com uma nova versão da plataforma. No âmbito da **UCV**, caso se deseje fazer um update ao *branch* com o código implementado no *MediaCore* manualmente, é necessário ter em consideração as alterações que a base de dados pode ter sofrido nos últimos updates da plataforma.

model contém os objectos do modelo de dados para serem utilizados em *Python*, populados utilizando a biblioteca *SQLAlchemy*. Estes estão relacionados com a base de dados e, qualquer alteração na base de dados implica a alteração dos objectos correspondentes nesta pasta.

public contém toda a informação que é fornecida ao browser tal como está. Contém os ficheiros de estilos (*CSS*), imagens e *JavaScript*. Basicamente, corresponde à parte "estática" da plataforma. Esta pasta tem que ser tratada de forma especial, sendo utilizado um atalho para esta pasta no caminho do *Apache Web Server* que representa a raíz da plataforma. A listagem de código 2.1 apresenta a constituição da raíz do projecto no *Apache Web Server* para caso da **UCV**.

templates pode ser considerada a pasta correspondente à vista (*View*) do *MVC*, contendo os *templates* que irão gerar as páginas da plataforma, recorrendo a *Genshi*, que permite utilizar *Python* para adicionar algum dinamismo à geração das páginas, bem como o tratamento da informação vinda dos *controladores* respectivos.

```
[root@ucv ~]# ll /var/www/html/ucv
total 8
lrwxrwxrwx. 1 root   root    15 Aug  1 11:31 data -> /mediacore/data
-rw-rw-rw-. 1 apache apache   1 Apr 16 13:35 live.dat
-rwxr-xr-x. 1 root   root   598 Mar  7 16:44 mediacore.fcgi
lrwxrwxrwx. 1 root   root    27 Aug  1 11:31 public -> /mediacore/
    mediacore/public
```

Listagem de código 2.1: Listagem de pastas na raíz do projecto **UCV** no *Apache Web Server*

Tal como é comum encontrar na arquitectura *MVC*, a criação de um template implica um controlador (e/ou uma acção de um controlador) que o sirva. E, no caso de aplicações web, implica um "interface" para que o utilizador possa chamar (*HTTP Request*) a acção do *controlador* que devolve o *template*, logo implica a alteração dos ficheiro de configuração de rotas para que esse interface exista.

2.1.3 Software de desenvolvimento

Este projecto foi desenvolvido utilizando o seguinte software:

BBEdit[11] Editor de código utilizado para o desenvolvimento e manutenção de todos os ficheiros de código da plataforma.

Safari Browser web com o modo de desenvolvimento activo, para facilitar a detecção de erros e correcção de bugs no lado do cliente.

2.2 Requisitos

Nesta secção são indicados os requisitos necessários para a alteração da plataforma da **UCV** actual, na forma de *user stories*.

2.2.1 User Stories

Os pedidos de alteração efectuados pelos diversos interessados no funcionamento da plataforma estão descritos na lista abaixo sob a forma de User Stories. Algumas destas alterações foram especificadas pelo próprio estagiário, por serem alterações necessárias para que a plataforma pudesse continuar a funcionar a longo prazo.

UCV-US-20-01 Redefinir todo o layout da **UCV**.

UCV-US-20-02 Criar uma área para apresentação de concursos relacionados com o canal de televisão.

UCV-US-20-03 As páginas implementadas especificamente para a **UCV** devem poder ser modificadas através do painel de administração (sem necessidade de WYSIWYG).

UCV-US-20-04 Portar a plataforma para o novo servidor.

UCV-US-20-05 Migrar o código base para o da nova versão do *MediaCore*.

UCV-US-20-06 Adequar o *player Flash*[12] para que os menus das páginas web que o importem sejam apresentados por cima do *player*.

2.3 Implementação

A implementação do canal web da UC, sendo suportada inicialmente pela plataforma *MediaCore* na versão 0.8, tinha algumas necessidades de actualização para que a sua utilização se tornasse mais adequada, tanto pela equipa de inserção de conteúdos, como pelo público geral. Na versão 0.9 do *MediaCore* foram adicionadas algumas funcionalidades que se revelavam úteis.

No entanto, continuavam a existir um conjunto de funcionalidades que necessitavam de ser implementada, algumas das quais haviam sido anteriormente implementadas durante a implementação inicial da **UCV**, com a versão 0.8 do *MediaCore*, e outras que necessitavam de ser implementadas (apresentadas nos requisitos indicados na secção de requisitos apresentada anteriormente).

Integração com as redes sociais (as já suportadas, como o *Facebook* e o *Twitter*[13] e o alargamento ao vasto leque de redes sociais existentes utilizando o plug-in *AddThis*[14]).

Implementação do suporte para plataformas móveis (as necessidades existentes, como os videos relacionados, para a versão de *iPad*).

Implementação da páginas:

página de concursos, para a apresentação de concursos relacionados com o canal de televisão;

página de directos, para apresentar as emissões em directo efectuadas pelo canal;

página de "quem somos", para apresentar a informação sobre a localização, contactos e a constituição da equipa que desenvolve o projecto.

[12] http://www.adobe.com/products/flashplayer.html
[13] http://www.twitter.com
[14] http://www.addthis.com

Com estas necessidades, existiu também a necessidade de portar toda a plataforma para o novo servidor, adquirido para suportar as plataformas desenvolvidas neste estágio, que possuía maior capacidade de armazenamento.

Para celebrar o aniversário da **UCV**, no dia 21 de Novembro de 2011, estava também planeado o desenvolvimento de um novo layout, mais moderno e esteticamente agradável.

2.3.1 Redes sociais

Por forma a garantir a interoperacionabilidade entre a plataforma e as redes sociais, foram necessárias determinadas adaptações aos templates das páginas de conteúdos para que a informação ficasse disposta de forma a que as redes sociais conseguissem obter a informação correcta para cada uma das partes que apresentam nas "paredes" dos utilizadores. Mais concretamente, redes sociais como o *Facebook*[15] e o *Google Plus*[16] dispõem uma imagem, um título e uma descrição parcial do conteúdo da página.

Isto é possível de implementar através de propriedades meta no código *HTML* das páginas, que são processadas pelos sistemas das redes sociais assim que um utilizador escreva um endereço na sua página do *Facebook*, ou clique num botão "Like" da página. Estas propriedades fazem parte de um *API* denominado *OpenGraph*[17] que é compatível com várias redes sociais, entre as quais o *Facebook* e o *Google Plus*, consideradas as essenciais para a divulgação.

Este processo implicou um tratamento especial para a página principal. Como esta página é utilizada para apresentar destaques, existem duas acções disponíveis para o utilizador: a divulgação da **UCV** ou a divulgação do conteúdo presente na altura da visita pelo utilizador.

Se um utilizador clicar no botão de "Like", do *Facebook*, que corresponde ao *Like* do video em destaque, a informação que tem que ser transmitida para a "parede"do utilizador é a informação disponível na página do video, logo o *Like* tem que passar para o *Facebook* a informação de que a acção está a ser efectuada sobre a outra página e não sobre a página actual (ver listagens de código 2.2 e 2.3).

```
<iframe src="http://www.facebook.com/plugins/like.php?href=
\${h.url_for(controller='/media',action='view',slug=featured.slug,
    qualified=True)}&...
```

Listagem de código 2.2: Redireccionar o Like para outra página

```
<meta property="og:title" content="\${media.title}" />
<meta property="og:description" content="\${media.description_plain}" />
<meta property="og:image" content="\${h.thumb_url(media,'l',qualified=
    True)}" />
<meta property="og:url" content="\${h.url_for(controller='/media',action
    ='view',slug=media.slug,qualified=True)}" />
```

Listagem de código 2.3: Informação colocada nas propriedades meta no código *HTML* da página do video

[15] http://www.facebook.com
[16] http://plus.google.com
[17] https://developers.facebook.com/docs/opengraph/

Por outro lado, se o utilizador clicar no botão "Like" da página ou copiar o endereço inicial da **UCV** para a sua "parede", a informação é acerca da **UCV** e não sobre o video, que é a informação que está nas propriedades meta da página (ver listagem de código 2.4).

```
<meta property="og:title" content="\${h.page_title(default='Televis&
    atilde;o Web da UC',media='all')}" />
<meta property="og:description" content="Aqui vai encontrar reportagens,
    entrevistas e not&iacute;cias sobre o que acontece na Universidade de
    Coimbra. A UCV mostra-lhe a investiga&ccedil;&atilde;o de ponta, os
    cursos, as iniciativas, a hist&oacute;ria e as pessoas. Descubra o
    Universo UC." />
<meta property="og:image" content="http://ucv.uc.pt/ucv/appearance/logo.
    png" />
<meta property="og:url" content="\${h.url_for(controller='/media',action
    ='explore',qualified=True)}" />
```

Listagem de código 2.4: Informação colocada nas propriedades meta no código *HTML* da página inicial

Para apoiar o desenvolvimento desta funcionalidade existia a necessidade de validar a informação nas propriedades meta que eram colocadas nas páginas de video e na página principal, sendo isso efectuado com recurso a uma ferramenta disponibilizada pelo *Facebook*, inicialmente denominada *Linter* e que entretanto foi renomeada simplemente para *Debugger*[18].

2.3.2 Páginas adicionais

O processo de criação de páginas adicionais consiste na criação de um novo *controlador* e do método que responda ao pedido, e a inserção do pedido e correspondência com esse *controlador* e método no ficheiro de rotas.

```
    # Podcast Episodes
    map.connect('/podcasts/{podcast_slug}/{slug}/{action}',
        controller='media',
        action='view',
        requirements={'action': 'view|rate|comment'})

    # Live Stream
    map.connect('/direto', controller='live', action='index')

    # Contest
    map.connect('/passatempo', controller='contest', action='index')
```

Listagem de código 2.5: Configuração das rotas para as páginas adicionadas

Alternativamente, poder-se-á utilizar um controlador existente, adicionando um novo método para responder ao pedido, e mapeando estes no ficheiro de rotas.

Para adição da página de concursos foi utilizado um novo controlador que respondia com uma única acção para apresentar a página. Por outro lado, as páginas de edição dos conteúdos das páginas de directo, concursos e "quem somos" foram implementadas criando novas acções num controlador existente.

[18] https://developers.facebook.com/tools/debug

2.3.3 Transferência da plataforma para o novo servidor

A transferência da plataforma para o novo servidor implicava um conjunto de processos que tinham que ser efectuados e que também estavam relacionados com o facto de ser necessário actualizar a plataforma para uma nova versão. A parte destes processos relativa ao upgrade da plataforma estão listados na documentação da plataforma[19], mas, devido ao facto de o sistema ter que mudar para um novo servidor, o processo teve que ser ligeiramente alterado para que fosse bem sucedido. O processo utilizado foi o seguinte:

1. Instalação dos requisitos mínimos como descritos na página "Preliminary Requirements Installation" na documentação da plataforma;
2. Criar uma pasta para a versão do *MediaCore* que corresponde às versão base da **UCV**;
3. Criar um novo `virtualenv`[20] nessa pasta, para conter uma instalação de packages específicas do *MediaCore*, isolando-as numa instalação virtual, como definido no passo 1 da documentação da plataforma;
4. Copiar os ficheiros do *MediaCore* para essa pasta (efectuando a descompressão do ficheiro comprimido com a versão 0.9 utilizada para desenvolver o *branch* da **UCV**) e executar a *script* do *MediaCore* que instala todas as packages *Python*, necessárias para o *MediaCore* funcionar, como definido no passo 2 da documentação da plataforma;
5. Mover a pasta `data` na raíz da plataforma para outro local, sendo depois referenciada dentro da pasta de raíz, com um *link simbólico*;
6. Copiar os dados da pasta `data` do antigo servidor para a mesma pasta no novo servidor, utilizando o esquema definido no passo 3 na documentação da plataforma, mas tendo em consideração a mudança de servidor, utilizando ferramentas como o *rsync*;
7. Criar a base de dados no *MySQL*, criar o utilizador com permissões de acesso a essa base de dados e efectuar o *dump* da base de dados do antigo servidor e executar o mesmo *dump* no novo servidor, de forma a obter a mesma base de dados no novo servidor;
8. Efectuar a alteração do ficheiro de configuração da plataforma para ter os dados de acesso à base de dados, tal como definido no passo 4 da documentação da plataforma;
9. Executar a *script* de migração da base de dados como indicado no passo 5 da documentação da plataforma;
10. Efectuar a instalação do *middleware* entre o *Apache Web Server* e o *MediaCore*, como indicado no passo 6 da documentação da plataforma.

A mudança de servidor implicaria também que o domínio passasse a apresentar a plataforma no novo servidor com o mínimo de *downtime*. Tendo em conta que o *DNS* autoritário actual estava a permitir que as *caches DNS* pudessem fazer a *cache* do registo durante uma semana, foi feita a alteração do registo *DNS* para o servidor antigo para que os outros servidores e *caches DNS* não pudessem manter o registo. No dia da alteração e após a mesma, voltou-se a actualizar o registo para que o endereço fosse o do novo servidor e com o periodo de *cache* normal. Desta forma, a maioria das caches *DNS* que seguiam os padrões normais de *caching DNS* ficaram imediatamente actualizadas.

[19] http://mediacorecommunity.org/docs/install/upgrade.html
[20] http://www.virtualenv.org/en/latest/index.html

3. iTunes U

O *iTunes U* é uma área da *loja iTunes* da *Apple Inc.* onde são publicados conteúdos de ensino superior, disponíveis para download e subscrição a qualquer pessoa. A plataforma é acessível através do software *iTunes* em computadores pessoais com os sistemas operativos *Mac OS X* ou *Windows* e através de uma *app* denominada simplesmente por *iTunes U*, disponível para *iPhone*, *iPod Touch* e *iPad*.

O ecossistema permite que haja a sincronização entre as subscrições efectuadas no *iTunes*, num computador pessoal, e os dispositivos móveis compatíveis com a plataforma.

A plataforma disponibiliza a informação de forma semelhante aos *podcasts* do *iTunes U*, com a diferença de terem uma propriedade extra que define a disciplina em que o conteúdo se insere, possibilitando a inserção de conteúdo em formatos audio, video e *eBook*, como por exemplo, *PDF*.

Mais recentemente, foi implementada na plataforma uma funcionalidade que permite disponibilizar conteúdos para aulas que consistem também em material interactivo como *iBooks* e *app*s móveis para dispositivos *iOS* (*iPhone*, *iPod Touch* e *iPad*). Para além do materiais que constituem esta nova área, passa também a ser possível restringir o acesso ao conteúdo destas.

Esta última funcionalidade não teve qualquer impacto neste estágio, visto que todo o funcionamento é baseado nos servidores da *Apple Inc.*, sendo fornecidos todos os meios de publicação e gestão necessários para construção das aulas.

A presença da Universidade de Coimbra no *iTunes U* deu início em Janeiro de 2010 em conjunto com o lançamento da plataforma à presença de instituições de ensino superior portuguesas nessa plataforma. A Universidade de Coimbra foi a primeira instituição de ensino superior portuguesa com presença no *iTunes U*, tendo sido também a primeira instituição a disponibilizar conteúdos de língua portuguesa na plataforma.

3.1 Arquitectura

O *iTunes U* está organizado de forma a que uma instituição registada na plataforma possa criar colecções, que são conjuntos de materiais sobre um determinado tema específico. Cada uma destas colecções funciona como um *podcast* independente, sendo possível subscrever o seu conteúdo sem necessidade de subscrever o restante conteúdo publicado pela instituição.

Estas colecções são compostas por conteúdos, também designados de forma técnica por *items*. Estes *items* podem estar categorizados numa diciplina específica, que corresponde a um conjunto

de 3 ou 6 números. As disciplinas gerais são representadas por 3 números, e as sub-disciplinas são representadas pelos 3 números da disciplina geral, mais 3 números que identificam a sub-disciplina dentro dessa disciplina geral. Esta listagem identificadores e respectivas disciplinas está apresentada no **Anexo 1 - Códigos de disciplinas do *iTunes U***.

Todas as colecções são descritas em ficheiros *XML*, compatíveis com a especificação *RSS*, com tags extra com informação específica de *podcasts* do *iTunes* e do *iTunes U*. Estes ficheiros *XML*, bem como os conteúdos que referenciam, têm que ser servidos pela instituição, através de sistemas próprios[1]. Isto implica que a instituição possua os seus próprios processos para criação e gestão dos seus conteúdos bem como para a criação dos ficheiros *XML* com a informação que representa a colecção no *iTunes U* e formas de garantir que o *iTunes U* consegue sempre aceder aos conteúdos referenciados.

3.1.1 Plataformas de suporte

As plataformas de suporte do *iTunes U* têm que ser capazes de responder a um conjuntos de processos necessários para a operacionalização do projecto. Estes processos estão apresentados na figura 3.1.

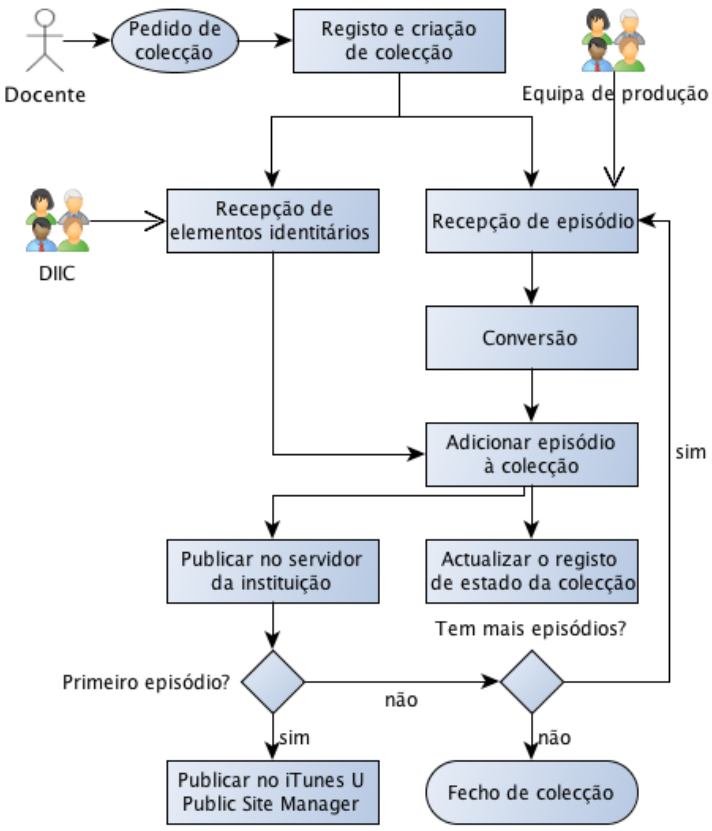


Figura 3.1: Criação e actualização de colecções do *iTunes U*

Neste estágio, apesar de terem sido instalados ou preparados alguns dos sistemas para registo e para conversão de formatos de video e áudio, a componente principal do projecto é o software utilizado para suportar a distribuição de conteúdos para o *iTunes U*.

[1]Entretanto a Apple começou a fornecer o serviço de *hosting* e de gestão de todo o processo.

A base do projecto assenta sobre um servidor de conteúdos que fornece os conteúdos produzidos na UC aos utilizadores que navegam pelas colecções da instituição no *iTunes U*. Todo o processo é feito através de pedidos *HTTP*, o que significa que apenas é necessário fornecer os conteúdos e os ficheiros *XML* através de um serviço *HTTP* comum, capaz de responder a pedidos de conteúdo estático.

No entanto, para facilitar a gestão dos conteúdos da UC de forma a que as pessoas responsáveis pela gestão de conteúdos não necessitassem de uma formação técnica, foi necessário arquitectar uma estrutura que permitisse uma manutenção das colecções baseadas num *GUI* de fácil utilização.

Para tal, foi adquirido o software *Feeder*[2], que fornece um *GUI* de fácil utilização e que permite efectuar a inserção de novos episódios às colecções com simples *drag&drop*, sendo apresentada uma janela com os campos necessários para descrever e categorizar os conteúdos. O software faz a gestão das feeds *XML* que serão interpretadas pelos servidores do *iTunes U*.

Para que o software funcionasse correctamente, foi necessária a utilização de um serviço de transferência de ficheiros como *WebDAV* ou *FTP*, bem como a definição de uma estrutura base para organização das colecções no sistema de ficheiros do servidor.

Optou-se por identificar as colecções identificador da disciplina a que a colecção corresponde, seguido de um número de ordem adicional, permitindo que existam várias colecções da mesma disciplina. Este identificador é associado aos documentos de estado da colecção, permitindo que seja a relação entre as pastas de conteúdos geridos pelo *Feeder* e a documentação dos mesmos registados noutra plataforma de controlo, seja directa. Na figura 3.2

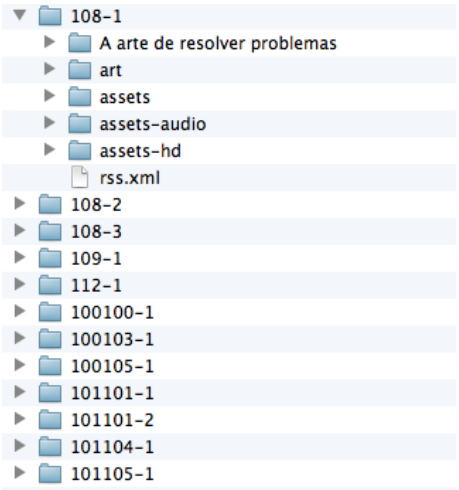


Figura 3.2: Estrutura de pastas do *iTunes U*

Esta estrutura de pastas está disposta na directoria do servidor web, sendo o acesso ao ficheiro *XML* efectuado como apresentado na listagem 3.1. O *Feeder* gera este endereço automaticamente, não sendo necessário que o gestor das colecções saiba construir o endereço manualmente.

```
http://itunesu-repository.uc.pt/collections/<id>/rss.xml
```

Listagem de código 3.1: Estrutura do endereço para o ficheiro representativo de uma colecção no servidor da UC

[2]http://reinventedsoftware.com/feeder/

3.2 Requisitos

Devido à necessidade de se utilizar o software *iTunes* para se conseguirem aceder aos conteúdos da UC no *iTunes U*, existe a possibilidade de alguns utilizadores não poderem aceder aos conteúdos por não poderem instalar o software, principalmente se utilizarem plataformas que não possuem os requisitos mínimos para poder utilizar o software (por exemplo, plataformas que utilizem outros sistemas operativos, como uma distribuição de *Linux*). Para além disto, também existem todas as outras plataformas móveis que não têm aplicações apropriadas para aceder aos conteúdos do *iTunes U*.

Como tal, foi considerada a criação de um acesso aos mesmos conteúdos via *HTML*, através do browser.

3.3 Implementação

Uma das questões que foi necessário implementar foi a possibilidade de apresentar as colecções do *iTunes U* da Universidade de Coimbra de forma a que estivessem disponíveis através do browser. Se tivermos em conta o modelo *MVC*, o *modelo de dados* nesta plataforma, serão os ficheiros *XML* que contém a informação sobre as colecções.

Desta forma, de modo a facilitar o processo de construção das páginas correspondentes para apresentar ao utilizador do browser, optou-se pela utilização da *cron* para executar periodicamente um pequeno programa desenvolvido com recurso a *Java* e *Freemarker*[3] que irá visitar todas as pastas de colecções e extrair a informação de cada colecção, criando uma página de índice e as páginas de cada uma das colecções[4].

Tendo em conta que a determinado momento podem existir colecções no sistema de ficheiros do servidor que não correspondem directamente às colecções existentes no *iTunes U*, devido a ainda não estarem publicadas por ainda não terem episódios ou por ainda não terem imagens de apresentação, o programa teria que verificar a existência destas componentes para decidir se a página final seria ou não publicada na versão web.

Tal como no processamento de linguagens, é efectuado um passo intermédio que converte uma linguagem numa representação intermédia, a partir da qual se gera uma representação final. O processo do software desenvolvido para resolver este problema é, em tudo, idêntico ao processo de processamento de linguagens, sendo os ficheiros *XML* lidos, armazenada a informação importante em objectos que representam o estado intermédio, e finalmente são criadas as páginas *HTML* com a informação final, através de um template.

[3] http://freemarker.sourceforge.net
[4] visualizável em http://www.uc.pt/itunesU/coleccoes

4. Agenda7

A agenda cultural de Coimbra, denominada **Agenda7**, é uma plataforma de divulgação de eventos de teor cultural e científico da cidade de Coimbra. Esta plataforma foi um projecto que foi iniciado devido à falta de uma agenda oficial que centralizasse toda a informação relativa a eventos culturais e científicos de Coimbra. Muitos dos eventos existentes na cidade na altura eram disponibilizados única e exclusivamente pelas entidades organizadoras, sendo a informação distribuída pelas páginas web de cada uma das entidades. No âmbito da Universidade de Coimbra, cada departamento era responsável pela divulgação dos seus próprios eventos nas suas páginas, sendo alguns destes eventos seleccionados para serem apresentados na página inicial da UC. Uma das necessidades para que a divulgação de eventos nesta plataforma não implicasse a alocação de mais recursos humanos para a população de eventos era a possibilidade das diferentes entidades inserirem os próprios eventos na plataforma.

O desenvolvimento desta plataforma deu início em Março de 2011, tendo sido iniciado pelo estagiário e pelo Engenheiro Eduardo Perdido, que participou no desenvolvimento da plataforma nos dois primeiros meses antes de ser colocado noutro projecto dentro da UC.

A demonstração do primeiro protótipo funcional da plataforma foi feita no dia 27 de Julho de 2011 do mesmo ano, para uma audiência constituída por responsáveis de diferentes entidades da cidade de Coimbra que estivessem potencialmente interessados numa participação no projecto.

Até ao dia 31 de Agosto de 2011, a plataforma encontrava-se na fase final de desenvolvimento, faltando apenas efectuar alguns testes às funcionalidades implementadas e respectivas correcções que fossem consideradas necessárias, bem como a implementação do suporte para divulgação dos eventos também nas redes sociais. Foi também desenvolvido o manual de utilizador do **backoffice**, que pode ser observado no **Anexo 5 - Manual de utilizador da Agenda7**.

A plataforma só foi lançada em 28 de Setembro para inserção de eventos na plataforma, e em 17 de Outubro ao público geral.

4.1 Arquitectura e Tecnologia

A Agenda7 é constituída por duas componentes principais: uma componente de acesso público, denominada **Portal**, que serve para consultar a informação, e uma componente de acesso restrito, denominada **Backoffice**, que serve para inserir, editar e apagar informação. Estas componentes podem ser visualizadas na figura 4.1, bem como a sua relação com os diferentes tipos de utilizadores da plataforma.

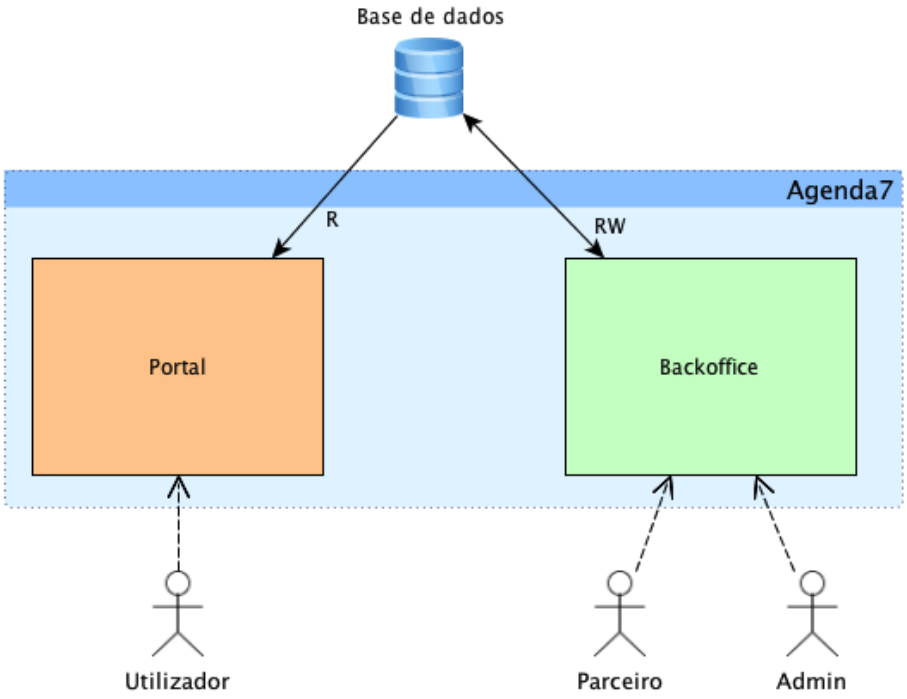


Figura 4.1: Diagrama geral da **Agenda7**

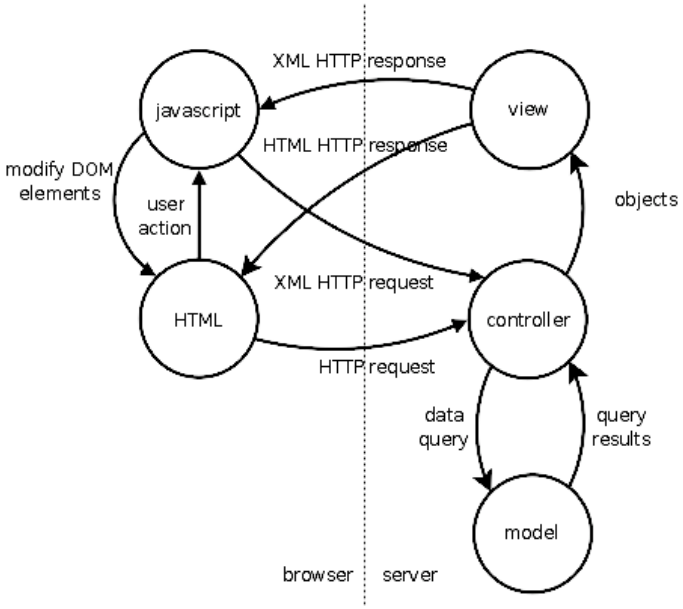


Figura 4.2: Arquitectura *MVC*

A plataforma foi desenvolvida tendo por base a arquitectura *MVC*, utilizando *Java Struts* para o *server-side* associado ao servidor de aplicações, *Apache Tomcat*, e *XHTML* e *JQuery* com recurso a *AJAX* para o *client-side*. Na figura 4.2 está apresentado o diagrama que representa o funcionamento base destas tecnologias em conjunto.

A base de dados é suportada pelo *MySQL* e foi utilizada a biblioteca *Velosurf*[1] para se efectuar o mapeamento da base de dados para objectos que possam ser manipulados pelo Java. O *Velosurf* é uma biblioteca mais simplista que opções mais comuns de persistência de dados

[1] http://velosurf.sourceforge.net

como, por exemplo, o *JPA*, mas tem uma funcionalidade que se revelava útil para o futuro, que é a possibilidade de separar as queries *SQL* para ficheiros *XML*, completamente isolados do código *Java*, e que permite mesmo a implementação de um sistema de actualização das queries sem a necessidade de *downtime* da plataforma.

Estas *queries* que podem ser isoladas num ficheiro *XML* são *queries* desenvolvidas para responder a um conjunto de necessidades de filtragem fixas, associadas a determinadas funções da plataforma e que necessitam de relacionar várias tabelas da base de dados.

O *tradeoff* é que, para que a implementação da plataforma se abstraia da estrutura da base de dados, é necessário implementar objectos que representam diferentes registos, e que gerem as suas propriedades e a necessidade de persistência das mesmas na base de dados.

O diagrama *ER* da base de dados da versão da **Agenda7** implementada inicialmente pode ser visto na figura 4.3.

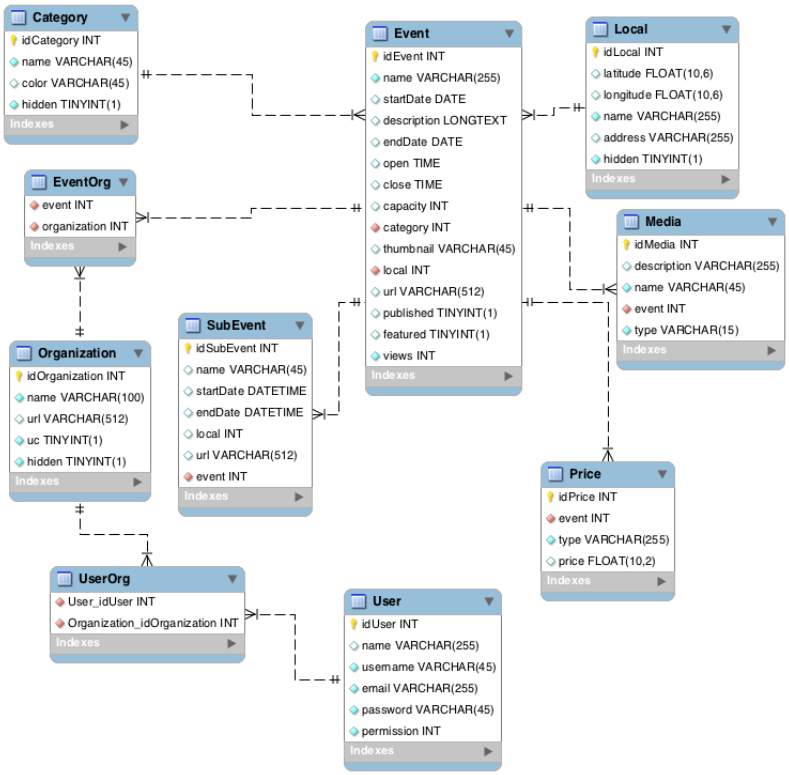


Figura 4.3: Diagrama *ER* da primeira versão da **Agenda7**.

A arquitectura da base de dados está centrada em torno da entidade `Evento`, sendo esta a entidade que representa o foco principal da plataforma desenvolvida. Para cada uma destas entidades existe um objecto *Java* que serve para manipulação e para transmissão de dados entre o *controlador* e o *template*.

Nas sub-secções seguintes são apresentadas as questões relacionadas com a arquitectura das duas principais componentes da plataforma.

4.1.1 Backoffice

O **Backoffice** é a componente da plataforma restrita a um conjunto de utilizadores e acessível apenas após um processo de autenticação. Existem dois tipos de utilizadores que têm diferentes permissões dentro do **Backoffice**. O **Editor** (ou **Parceiro**) que apenas pode inserir, editar e apagar eventos e adicionar novos locais. O **Administrador** que pode realizar todas as operações do **Editor** mais as operações de edição e remoção de locais e inserção, edição e remoção de utilizadores, organizações e categorias. Na figura 4.4 estão apresentados os casos de uso correspondentes a estas operações associadas ao tipo de utilizador que as pode realizar.

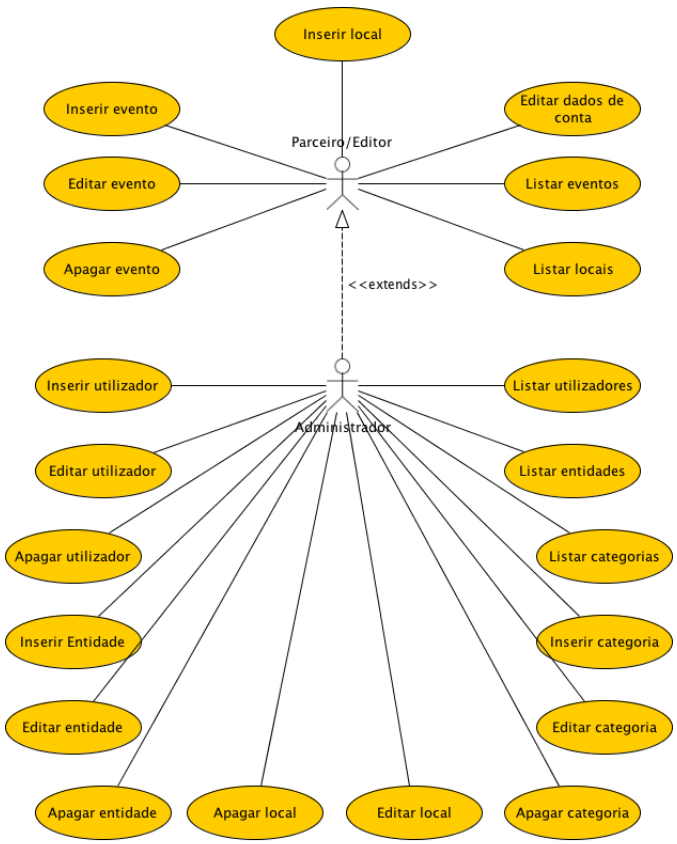


Figura 4.4: Casos de uso para o Backoffice da Agenda7.

O **Backoffice** é baseado em dois tipos de interfaces base. O interface base que corresponde à navegação, que é baseado em listagens de registos, quer para eventos, locais, organizações, utilizadores e categorias, quer para informação relativa a um evento específico, como é o caso dos preços, sub-eventos e elementos multimédia. Todos estes são listados no seu contexto, apresentados através de uma tabela *HTML* com links que possibilitam realizar operações de edição, inserção e remoção, entre outras mais específicas. O outro interface base corresponde a formulários *HTML* e é este que permite a inserção e edição de informação para cada uma das entidades apresentadas no diagrama ER da figura 4.3.

Como em qualquer plataforma web que utiliza a arquitectura *MVC*, os dados dos formulários são recebidos por *POST* e atribuídos no *controlador* para que possam ser tratados. Em cada *controlador* de formulário é feita uma verificação de permissões, seguida de uma validação dos dados e só depois é que os dados serão inseridos ou editados. Este processo está esquematizado na figura 4.5.

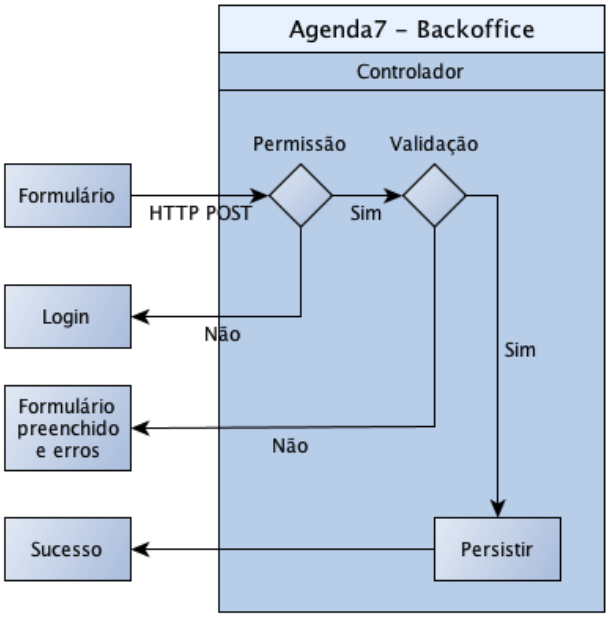


Figura 4.5: Processo de validação para inserção de informação de formulários

4.1.2 Portal

O **Portal** é a componente de acesso público da plataforma e que permite efectuar consultas por eventos que tenham sido publicados no **backoffice**.

Similar ao comportamento do **backoffice** é a utilização de diversos *controladores* para suportar cada uma das páginas necessárias. A diferença está no facto de não ser necessário efectuar a validação, pelo que não existe a necessidade de definir vários ramos.

Existe no entanto uma diferença relativamente ao **backoffice**, que é a necessidade de alguns *controladores* devolverem outro tipo de dados que não *HTML*. No caso específico, a plataforma utiliza um *plug-in JQuery* para construir um calendário, denominado *FullCalendar*[2], facilitando a representação visual dos dados, e que recebe a informação utilizando *AJAX*, formatada em *JSON*.

Na primeira versão da plataforma, a informação era tratada pelo *controlador* responsável, sendo depois convertida para a notação *JSON*, colocada num *template* limpo. Este processo implica chamar todos os resultados correspondentes à pesquisa e tratar/formatar a informação para se adequar ao esquema de objectos do *FullCalendar*.

O *FullCalendar* foi alterado para se adaptar às necessidades da plataforma, e surgiram duas versões. Uma para a representação num calendário normal, e outro para um mini-calendário, que necessitou de alterações ao nível do rendering do HTML e do texto incluido nas casas que correspondem aos dias.

Ou seja, o **portal** é a componente que implementa as chamadas *AJAX* apresentadas na figura 4.2, apresentada no início desta secção.

[2]http://arshaw.com/fullcalendar/

4.1.3 Software de desenvolvimento

Este projecto foi desenvolvido utilizando o seguinte software:

BBEdit[3] Editor de código utilizado para o desenvolvimento e manutenção dos *controladores*, *scripts SQL*, *templates*, *JavaScript*, *CSS* e outros ficheiros de configuração.

MySQL Workbench Software para criação e manutenção de base de dados *MySQL*, utilizado neste projecto para implementar o modelo de dados.

Tomcat Servidor aplicacional, instalado localmente no computador de desenvolvimento, e num servidor de testes.

Safari Browser web com o modo de desenvolvimento activo, para facilitar a detecção de erros e correcção de bugs no lado do cliente.

Apache Ant Software de automatização de processos, utilizado para automatizar o processo de deploy do projecto, quer para testes, quer para produção.

4.2 Requisitos

A **Agenda7** foi implementada antes de ter sido dado início ao estágio, mas algumas partes da plataforma, necessitavam de novas funcionalidades que foram sendo pedidas para facilitar a utilização da plataforma.

Algumas dessas funcionalidades foram sendo descritas pelos utilizadores das plataformas e registadas sob a forma de *user stories*, para depois serem convertidas em requisitos funcionais.

4.2.1 User Stories

A7-US-20-01 Deverá existir uma forma de seleccionar uma porção da imagem que se está a colocar no **backoffice** como imagem de apresentação, por forma a que a formatação da mesma seja a correcta ao ser apresentada ao utilizador final.

A7-US-20-02 Deverá ser possível pesquisar a lista de locais e a lista de eventos.

A7-US-20-03 A descrição do evento deve ser separada do texto curto de apresentação.

A7-US-20-04 Deve ser possível inserir eventos com duração permanente ou sem final previsto.

A7-US-20-05 Deve ser possível inserir eventos sem horário definido ou que decorrem todo o dia.

A7-US-20-06 Permitir opções de formatação de textos de descrição de eventos que não sejam baseados no conhecimento da linguagem *HTML*.

A7-US-20-07 A criação do utilizador deve efectuar o envio automático dos dados de acesso para o e-mail do utilizador inserido.

A7-US-20-08 Permitir a visualização de eventos criados, mas não publicados, por pessoas que têm permissões para editar o mesmo (ou seja, com o login efectuado no **backoffice**).

A7-US-20-09 Os eventos devem descriminar melhor os tempos em que decorrem, devendo ser possível a existência de horários distintos para diferentes dias da semana.

A7-US-20-10 Deverá ser possível representar em que dias da semana um evento decorre, sendo esta informação disposta no calendário.

A7-US-20-11 A pesquisa por eventos deve ser ordenada de forma a que os eventos sejam dispostos por forma a que os primeiros eventos sejam aqueles que estejam a decorrer à menos tempo desde a data inicial de pesquisa.

A7-US-20-12 As cores utilizadas no calendário devem ser alteradas para se adequarem ao estilo de cores utilizadas pelo *TAGV*.

A7-US-20-13 Apresentar os resultados da pesquisa sem necessidade de recarregar a página.

A7-US-20-14 Criar uma secção especial para se inserirem eventos de diferentes categorias para a Semana Cultural da UC.

A7-US-20-15 Adequar o sistema de permissões para impedir que os parceiros possam aceder a páginas de eventos que não lhes pertençam.

4.2.2 Requisitos Funcionais

Tal como na **UCV**, os requisitos funcionais identificam as componentes do sistema que necessitam de alterações para se conseguirem implementar determinadas funcionalidades apresentadas nas user stories. Estes requisitos identificam quais são os requisitos ou user stories que lhes dão origem, as alterações necessárias e as implicações que podem dar origem a novos requisitos. Os requisitos funcionais podem ser visualizados no **Anexo 2 - Requisitos funcionais (Agenda7)**.

4.3 Implementação

A **Agenda7** está segmentada em duas componentes principais: o **backoffice** e o **portal**. Estas componentes serão parcialmente explicadas nas subsecções que se seguem.

As alterações efectuadas ao nível do **backoffice** encontram-se num documento que pode ser consultado no **Anexo 6 - Manual para alterações da Agenda7**

4.3.1 Backoffice

O **backoffice** é a componente utilizada para gerir e adicionar os eventos, bem como, consoante as permissões do utilizador da sessão, gerir os utilizadores, as categorias e as organizações/entidades que participam na criação de eventos.

A implementação do **backoffice** é relativamente directa, sendo baseada em formulários cujos dados preenchidos são validados e persistidos. A inserção e edição de eventos utilizam um processo semelhante, utilizando os mesmos templates de formulários, com a distinção de que o formulário irá ser apresentado vazio no caso de uma inserção, e preenchido com os dados actuais no caso de uma edição. Num passo intermédio, em que o utilizador está a tentar inserir informação inválida, o *controlador* irá voltar a gerar o formulário com a informação anteriormente preenchida e irá fornecer informação ao utilizador indicando quais os campos que não foram validados na inserção e a razão para essa informação não ter sido validada.

4.3.1.1 Envio automático de e-mail

O envio automatizado de e-mail para um utilizador (processo apresentado na figura 4.6 implica manter a password escolhida em *clear text* até que o e-mail seja enviado. O processo de envio do e-mail também não é uma operação imediata e pode ter alguma latência, pelo que é necessário que a implementação desta funcionalidade seja feita de forma assíncrona e sem bloquear o administrador.

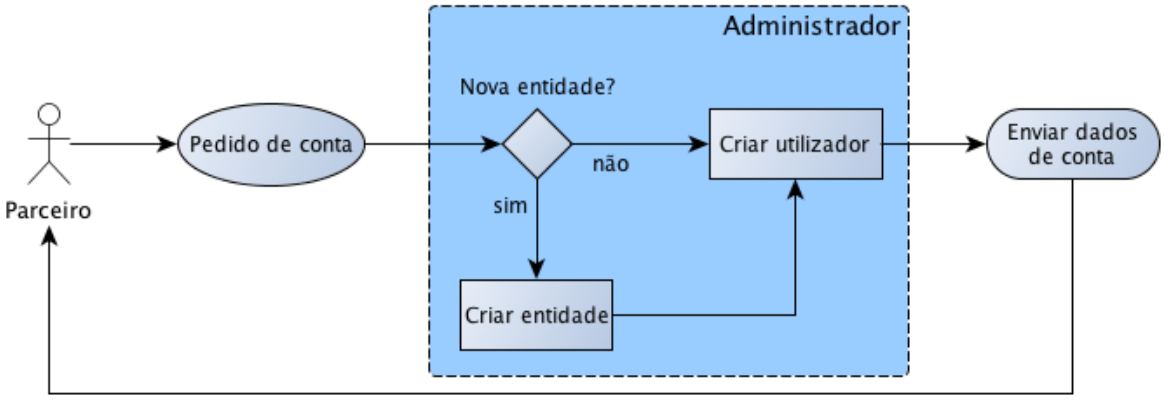


Figura 4.6: Registo de parceiros

4.3.1.2 Sistema de permissões

Como indicado na arquitectura da plataforma, todos os *controladores* do **backoffice** fazem uma validação por permissões antes de se realizar qualquer operação, mesmo para se aceder a uma listagem. Na figura 4.7 é apresentado o esquema do diagrama de classes implementado no **backoffice**.

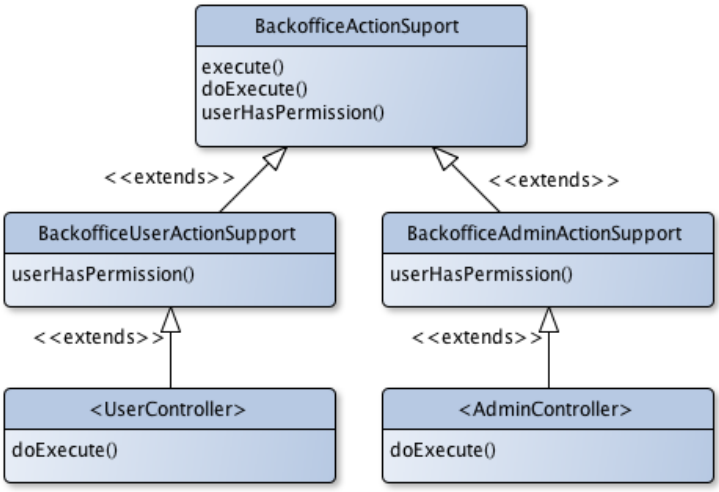


Figura 4.7: Esquema do diagrama de classes do **backoffice**.

Como se pode observar, o *controlador* do **backoffice** que é chamado irá pertencer a um dos ramos (Admin ou User). O primeiro método a ser executado é o `execute()` que irá chamar o método `userHasPermissions()` que terá uma implementação diferente para cada um dos ramos,

com uma componente comum implementada na classe de raíz, `BackofficeActionSupport` para identificar a página de destino caso o utilizador não tenha permissões de acesso ao *controlador* em questão. Se o utilizador tiver permissões, o método `doExecute()` é executado, sendo o resto do processo executado normalmente.

Isto permite mitigar a possibilidade de um utilizador tentar adivinhar um endereço para tentar executar uma acção geral (como por exemplo, aceder à listagem de utilizadores ou de organizações) sem necessitar de permissões para tal.

Mais importante que verificar se um utilizador está a tentar aceder a uma zona restrita (que não é um problema demasiado complexo depois de se implementar esta arquitectura), é confirmar que o utilizador (com permissões de Editor) não está a tentar efectuar operações sobre informação que não lhe pertence. Mais concretamente, cada utilizador (Editor) tem acesso a um conjunto limitado de eventos que corresponde às organizações às quais pertence, não podendo efectuar alterações sobre eventos que não têm uma referência para uma organização à qual pertencem. Como tal, o processo de verificação implementado no método `userHasPermissions()` da classe `BackofficeUserActionSupport` não pode ser baseado simplesmente em verificar se o utilizador está autenticado, pois este nível de permissões é o mais baixo do backoffice.

Também não é um processo directo, visto que nem todas as operações que podem ser efectuadas no **backoffice** por um Editor estão associadas a um evento.

O problema foi resolvido com recurso a uma variável de estado associada ao método *setter* do *id* de evento. Quando o `HTTP POST` chega ao servidor, as variáveis do *POST* são directamente atribuidas às variáveis com o mesmo nome no controlador de destino. Se o *POST* tiver a variável `idEvent`, irá marcar uma variável de estado que irá ser verificada pelo controlador na altura de verificação de permissões.

A partir daqui, basta comparar as organizações do utilizador às organizações do evento. Na listagem de código 4.1 pode-se ver um exemplo de uma query que permite validar a acção de um utilizador. Neste caso, se não forem devolvidos resultados, o utilizador não tem permissões para realizar a acção.

```
SELECT idEvent
  FROM Event
  WHERE organization IN (
   SELECT Organization_idOrganization
     FROM UserOrg
     WHERE User_idUser = user.getId()
  )
```

Listagem de código 4.1: Query de verificação de permissões de um Editor

4.3.2 Portal

O **portal** é a componente pública da plataforma, podendo ser acedida por qualquer pessoa, sem necessidade de registo. O **portal** é constituído (principalmente) por uma lista de categorias que filtra os eventos apresentados por categoria, um pequeno calendário que permite observar que eventos é que decorreram ou irão ocorrer no mês actual e que pode ser expandido para apresentar as ocorrências em detalhe, e um formulário de pesquisa, que permite filtrar os eventos por um conjunto de propriedades definidas pelo utilizador.

No acesso à página inicial do **portal** serão apresentados os eventos que estão a decorrer ou que irão ocorrer no futuro da seguinte forma:

1. Eventos que estão a decorrer ou que irão ocorrer e que foram marcados por um administrador como eventos em destaque.
2. Eventos que estão a decorrer ou que irão ocorrer no dia actual.
3. Eventos que irão ocorrer no futuro.

Os eventos marcados como destaques são escolhidos automaticamente para serem apresentados no topo da lista de quatro eventos que aparecem na página inicial, sendo seguidos pelos eventos que estão a decorrer. Normalmente existem vários eventos a decorrer, alguns deles que são permanentes e de longa duração, o que faz com que uma apresentação completamente aleatória destes eventos seja pouco justa na divulgação de eventos que iria deixar os eventos que ocorrem no periodo de um dia com menos visibilidade. Como tal, foi necessário desenvolver um algoritmo que permitisse fazer uma selecção de eventos baseada no número de dias que ocorre ou que se encontra a decorrer. Esse algoritmo está explicado na sub-secção 4.3.2.3.

Também importante é o tratamento da temporização dos eventos, mais concretamente, a forma como estes serão filtrados pela plataforma para serem depois expostos aos utilizadores. Como é necessária uma representação baseada em periodos que dispõe depois de filtros para os diferentes dias da semana, seria necessário efectuar uma segunda filtragem. Este processo poderia implicar grandes perdas de performance se não forem filtrados ao nível da base de dados, sendo necessária uma manipulação destes objectos para garantir que os eventos seleccionados ocorrem no periodo pretendido, pelo que a solução final deveria consistir em conseguir filtrar directamente todos os eventos ao nível da base de dados. Este processo será explicado em detalhe na subsecção 4.3.2.1.

Existe ainda a necessidade de transmitir a informação de forma escalável aos calendários utilizados pela plataforma, devido ao facto de, na altura, o sistema estar implementado para devolver já a informação estruturada especificamente para o **FullCalendar**, sendo este processo de estruturação efectuado pelo *controlador* responsável por fornecer estes dados. Sendo o sistema existente baseado num *plug-in* de *jQuery* que já dispõe de meios para obter e processar dados *JSON*, é possível implementar também a conversão dos dados recebidos do servidor para a representação utilizada pelo *FullCalendar*, reduzindo a necessidade de manipulação de dados por parte do servidor para adequar os dados a este *plug-in*, passando isto a ser efectuado no browser dos utilizadores, distribuindo a carga. Na sub-secção 4.3.2.2 este processo será explicado com mais algum detalhe.

4.3.2.1 Temporização de eventos

Tal como na linguagem natural se utiliza uma combinação entre dias do mês e dias da semana para se representar uma combinação de dias em que um evento ocorre. Quando houve a necessidade de implementar uma temporização mais dinâmica para os eventos da **Agenda7**, foi necessário considerar várias questões que envolviam a sua representação na base de dados. A abordagem escolhida foi utilizar uma representação binária, que possibilitasse efectuar comparações directas com cadeias binárias representativas de intersecções entre periodos.

Como se pode constatar num segmento diagrama *ER* apresentado na figura 4.8, que mostra as alterações efectuadas para a segunda versão da Agenda7, a representação dos tempos é baseada no armazenamento de vários pares de tempos, que representam o início e o fim de ocorrência. Na prática, existe um objecto, denominado `Datespan`, que representa os dias em que o evento

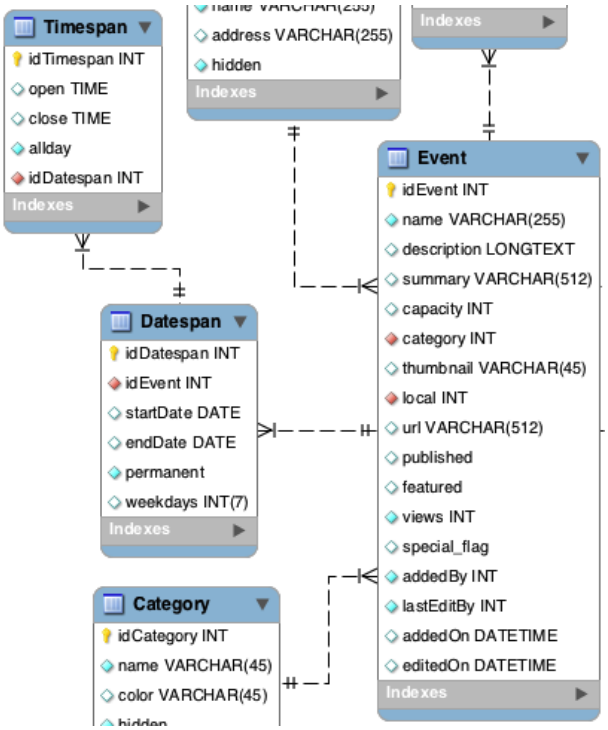


Figura 4.8: Reformulação do ER para a Agenda7.

ocorre, podendo ser um evento composto por vários periodos distintos, havendo a possibilidade de registar tempos em que, por alguma razão, o evento não decorre. Por sua vez, estes objectos são constituídos por múltiplos outros objectos onde se armazenam pares de horas, representando os horários de abertura e fecho.

A razão para poderem existir múltiplas horas de abertura e fecho de um evento para o mesmo conjunto de dias, tem a ver com a representação do intervalos como horas de almoço. Na realidade a representação destes objectos é bastante directa e não representam um problema no desenvolvimento do projecto.

No entanto, para que a inserção de eventos seja eficiente, deve ser possível inserir eventos que ocorrem durante longos periodos, mas só em determinados dias da semana, visto que é bastante comum existirem restrições temporais baseadas nos dias da semana. Para tal, foi adicionado um campo à tabela que representa periodos de dias, para representar também os dias da semana em que ocorre. Esta representação é binária, sendo cada bit da cadeia binária correspondente a um dia da semana, num máximo verificado de sete bits, o mesmo número que o número de dias de uma semana.

O processo de filtragem efectuado para obter eventos que ocorrem num determinado periodo de tempo é um processo com alguma complexidade que pode ser explicado num conjunto de cinco fases que vou apresentar de seguida. Para exemplificar, vou utilizar, como exemplo, um evento que ocorre de dia 30 de Julho de 2012 (Segunda-feira) até ao dia 7 de Agosto de 2012 (Terça-feira), não ocorrendo às Sextas, Sábados e Domingos (ou seja, $W_e = 0011110$), e comparado com uma pesquisa por eventos entre dia 2 e 6 de Agosto.

Representação de dados Dias da semana:

#	Representação	Dia da semana
0	$2^0_{10} = 0000001_2$	Domingo
1	$2^1_{10} = 0000010_2$	Segunda-feira
2	$2^2_{10} = 0000100_2$	Terça-feira
3	$2^3_{10} = 0001000_2$	Quarta-feira
4	$2^4_{10} = 0010000_2$	Quinta-feira
5	$2^5_{10} = 0100000_2$	Sexta-feira
6	$2^6_{10} = 1000000_2$	Sábado

Tabela 4.1: Representação dos dias da semana na base de dados

A representação base dos dias da semana corresponde à mesma representação numérica utilizada pelo *MySQL* e bibliotecas de datas, como a `java.util.Date`.

A representação na base de dados, apresentada na segunda coluna da tabela 4.1, permite que sejam somados os valores representativos dos vários dias, obtendo uma representação para qualquer combinação de dias da semana possíveis.

Fase 1 Objectivo:

Obter a intersecção entre o periodo pesquisado e o periodo de ocorrência do evento

Dados de entrada:

Dia de início do evento (D_{ie})
Dia de fim do evento (D_{fe})
Dia de início do periodo de pesquisa (D_{ip})
Dia de fim do periodo de pesquisa (D_{fp})

Saída:

Número de dias de intersecção (N_i)
Primeiro dia da intersecção (D_{ii})

Fórmulas:

$$D_i(D_{ie}, D_{ip}) = \begin{cases} D_{ie} & \text{if } D_{ie} > D_{ip} \\ D_{ip} & \text{senão} \end{cases}$$

$$D_f(D_{fe}, D_{fp}) = \begin{cases} D_{fe} & \text{if } D_{fe} < D_{fp} \\ D_{fp} & \text{senão} \end{cases}$$

$$N_i = D_f(D_{fe}, D_{fp}) - D_i(D_{ie}, D_{ip}) + 1$$

$$D_{ii} = D_i(D_{ie}, D_{ip})$$

Condições de continuidade:

$\mathbf{N_i \geq 7}$, o evento ocorre no periodo pesquisado pelo menos uma vez, visto que todos os dias da semana irão aparecer para comparação.
$\mathbf{N_i \leq 0}$, não há intersecção, logo o evento nunca ocorre.
$\mathbf{0 < N_i < 7}$, implica efectuar a **Fase 2**.

Exemplo:

$$D_{ie} = 30 \text{ de Julho de } 2012$$
$$D_{fe} = 7 \text{ de Agosto de } 2012$$
$$D_{ip} = 2 \text{ de Agosto de } 2012$$
$$D_{fp} = 6 \text{ de Agosto de } 2012$$
$$D_i(30 \text{ de Julho de } 2012, 2 \text{ de Agosto de } 2012) = 2 \text{ de Agosto de } 2012$$
$$D_f(7 \text{ de Agosto de } 2012, 6 \text{ de Agosto de } 2012) = 6 \text{ de Agosto de } 2012)$$
$$N_i = 6 \text{ de Agosto de } 2012 - 2 \text{ de Agosto de } 2012 + 1 = 5$$
$$D_{ii} = 4 \text{ (Quinta-feira)}$$

Fase 2

Fase 2.1 Objectivo:
Criar a cadeia binária com número do dias consecutivos

Dados de entrada:
Número de dias de intersecção (N_i)

Saída:
Cadeia base (C_b)

Fórmula:

$$C_b = 2^{N_i} - 1$$

Exemplo:

$$C_b = 2^5 - 1 = 31 = 0011111_2$$

Fase 2.2 A representação base dos dias da semana corresponde à mesma representação numérica utilizada pelo *MySQL* e bibliotecas de datas, como a `java.util.Date`.

Objectivo:
Mover a cadeia para que o primeiro dia corresponda ao dia da semana correcto

Dados de entrada:
Cadeia base (C_b)
Primeiro dia da intersecção (D_{ii})

Saída:
Cadeia intermédia (C_i)

Fórmula:

$$C_i = C_b \times 2_{ii}^D$$

Exemplo:

$$C_i = 31 \times 2^4 = 31 \times 16 = 496 = 111110000_2$$

Fase 2.3 De forma à cadeia corresponder a uma cadeia de sete bits que possa ser comparada com outra cadeia de sete bits representativa dos dias da semana, é necessário colocar os bits que ultrapassam o limite de sete bits da cadeia intermédia para as primeiras posições da cadeia.

A divisão é uma divisão inteira, representada pelo símbolo $\div$. A não ser que explicitamente indicado, todos os números indicados são de base dez.

Objectivo:

Converter a cadeia para uma cadeia representativa de sete bits

Dados de entrada:

Cadeia intermédia (C_i)

Saída:

Cadeia final (C_i)

Fórmula:

$$C_f = C_i \div 2^7 + C_i - (C_i \div 2^7) \times 2^7$$

Exemplo:

$C_f = 496 \div 2^7 + 496 - (496 \div 2^7) \times 2^7 = 3 + 496 - 3 \times 128 = 3 + 112 = 115 = 1110011_2$

Fase 2.4 Objectivo:

Comparar a cadeia final com a cadeia representativa dos dias da semana em que o evento ocorre num periodo

Dados de entrada:

Cadeia final (C_f)
Cadeia do `Datespan` do evento a comparar (W_e)

Saída:

Valor resultante da comparação (V)

Fórmula:

$$V = C_f \text{ AND } W_e$$

Exemplo:

$$V = 1110011_2 \text{ AND } 0011110_2 = 0010010_2$$

O valor V obtido no final da **Fase 2** (**Fase 2.4**) indica se o evento ocorre ou não no periodo de tempo pesquisado. Caso seja zero, não ocorre, e qualquer outro valor, que é sempre positivo, significa que há ocorrência.

Para que este processo fosse eficientemente aplicado, foi pensada nas diferentes formas possíveis de o implementar. A principal diferença entre possíveis implementações era a possibilidade de aplicar este processo nos controladores, em *Java*, aplicando operações sobre datas nos objectos, ou efectuar directamente uma *query* à base de dados que efectuasse todas as operações de filtragem necessárias para se obterem apenas os eventos que correspondiam à pesquisa efectuada.

A primeira abordagem, em *Java*, é mais fácil de implementar, efectuando operações sobre as propriedades dos *DAO*s directamente, através de algoritmos procedimentais que permitem facilmente adaptar as fases descritas anteriormente quase de forma directa, sem necessidade de um esforço de implementação adicional. Por outro lado, implicaria a criação de objectos que recebem os dados da base de dados e que seriam depois descartados pelo algoritmo, adivinhando-se um esforço adicional por parte da plataforma.

A segunda abordagem, em *SQL*, implica a formulação de uma *query* dinâmica que possa realizar as fases descritas anteriormente, e que não serão efectuadas de uma forma tão procedimental, mas garante que todos os objectos fornecidos à plataforma são os pretendidos pela pesquisa.

A segunda abordagem requer a formulação de expressões matemáticas na linguagem *SQL*, sem uma necessidade directa de conversão de vários tipos de representações de dados utilizados em *Java* (no caso específico, as datas), o que levou à sua escolha para a implementação deste processo.

Devido a muita da interacção com a **Agenda7** ser baseada na chamada de determinadas páginas que implicam a realização de *queries* de pesquisa, foram definidas algumas funções no *MySQL* que possibilitavam a utilização de uma cache para obter resultados mais eficientemente, evitando a realização dos mesmos cálculos para cada pedido de um utilizador.

Intersecção de um periodo de tempo com um `Datespan`: Para intersectar um periodo de tempo com a informação de um `Datespan`, é utilizada a função *SQL* descrita na listagem de código 4.2.

```
CREATE FUNCTION intersectDays(
    perm BOOL,
    fStart DATE,
    fEnd DATE,
    dStart DATE,
    dEnd DATE)
RETURNS INT DETERMINISTIC
RETURN
IF(perm,
    DATEDIFF(fEnd, GREATEST(fStart, dStart)) + 1,
    DATEDIFF(LEAST(fEnd, dEnd), GREATEST(fStart, dStart)) + 1);
```

Listagem de código 4.2: Função *SQL* que determina o numero de dias de intersecção entre um periodo definido e um `Datespan`

Intersecção de dois datespans: Para intersectar os periodos de dois `Datespans`, é utilizada a função *SQL* na listagem de código 4.3.

```
CREATE FUNCTION intersectDaysDS(
    fPerm BOOL,
    fStart DATE,
    fEnd DATE,
    dPerm BOOL,
    dStart DATE,
    dEnd DATE)
RETURNS INT DETERMINISTIC
RETURN
IF(dPerm,
    daySpan(fPerm, fEnd, fStart),
    DATEDIFF(LEAST(fEnd, dEnd), GREATEST(fStart, dStart)) + 1);
```

Listagem de código 4.3: Função *SQL* que determina o numero de dias de intersecção entre os periodos de dois `Datespans`

Esta função é utilizada para verificar que não existem sobreposições quando se estão a inserir novos `Datespan` na base de dados, sendo utilizada na função de validação do *controlador* que realiza este processo.

Pesquisa de eventos: Finalmente, aplicando as duas funcões apresentadas anteriormente, existe uma *query* para efectuar a pesquisa de eventos, apresentada na listagem de código 4.4. Os campos representados com *XML* são utilizados pelo *Velosurf* como placeholders para a colocação de informação externa. Estes campos correspondem à seguinte informação:

`<dateStart/>` e `<dateEnd/>` provêm das datas que correspondem ao periodo de pesquisa;

`<older/>` indica à pesquisa se devem ser incluidos eventos que já não se encontram a decorrer;

`<cat/>` e `<org/>` correspondem aos identificadores de categoria e organização, respectivamente, dos quais se pretendem obter eventos (ou zero, caso não seja necessário filtrar por nenhuma categoria ou organização, respectivamente);

`<name/>` e `<description/>` serão as partes das *queries* que irão efectuar uma pesquisa textual sobre os campos nome (título) e descrição, respectivamente;

`<limit/>` recebe o número de eventos limite que esta query deve retornar.

```
SELECT DISTINCT Event.*
  FROM Event, Datespan
  WHERE
        published = 1 AND
        Datespan.idEvent = Event.idEvent AND
        (
          (endDate >= CURDATE() OR permanent) OR <older/>) AND
          Event.idEvent IN (
                SELECT idEvent
                  FROM Datespan
                  WHERE
                    intersectDays(permanent, '<dateStart/>', '<dateEnd/>'
                        , startDate, endDate) > 0 AND
                        IF (
```

```
                        intersectDays(permanent, '<dateStart/>', '<
                            dateEnd/>', startDate, endDate) >= 7, 1,
                        weekdays & (
                          (((POW(2, intersectDays(permanent, '<
                              dateStart/>', '<dateEnd/>', startDate,
                              endDate)) - 1)
                          << (DAYOFWEEK(GREATEST('<dateStart/>',
                              startDate)) - 1)) DIV 128) +
                              (((POW(2, intersectDays(permanent, '<
                                  dateStart/>', '<dateEnd/>', startDate,
                                   endDate)) - 1)
                              << (DAYOFWEEK(GREATEST('<dateStart/>',
                                  startDate)) - 1)) \% 128)
                        )
          )
    ) AND
    (category = <cat/> OR <cat/> = 0) AND
    Event.idEvent IN
      (SELECT event FROM EventOrg WHERE organization = <org/> OR <org/> =
          0) AND
    ((<name/>) OR (<description/>))
  ORDER BY DATEDIFF(CURDATE(), startDate) LIMIT <limit/>
```

Listagem de código 4.4: *Query SQL* que determina os eventos que ocorrem num periodo de tempo

4.3.2.2 Representação de eventos num calendário

Como indicado na secção de **Arquitectura** deste projecto, o *FullCalendar* recebia já dados preparados para efectuar o rendering do calendário para a página *HTML*, o que implica que a carga de manipulação dos dados está do lado do servidor. Isto pode ter grandes implicações caso o número de eventos por periodo de tempo aumente drásticamente no futuro e, como todas as páginas da plataforma apresentam o calendário, cada pedido por página efectuado por um utilizador, implica manipular novamente os dados para que sejam correctamente transmitidos ao *FullCalendar*. Estes pedidos podem ser ainda efectuados pelas páginas que utilizam o nosso calendário embutido, aumentando ainda mais o problema de escalabilidade.

Devido a esta necessidade de escalar a quantidade de acessos à plataforma, os controladores de acesso público não devem executar operações de grande complexidade, sendo todas estas efectuadas no browser, através de tecnologias como *JavaScript*. Desta forma, a informação que é transmitida para o browser é parcialmente carregada via *JavaScript*, sendo estes dados praticamente dados recebidos no browser tal como estão na base de dados (*REST*), processados e a sua informação transformada directamente pelo browser de acordo com a especificação dos objectos que é necessária para popular o *FullCalendar*.

Para que tal fosse possível foi necessária a alteração dos objectos do modelo de dados (*Velosurf*) para que passassem a permitir o acesso aos objectos por eles referenciados, em vez de fornecerem apenas o ID. De forma a eliminar a carga extra em operações que não necessitem destes objectos, só é feito o pedido à base de dados por estes objectos quando explícitamente pedidos, através da chamada de funções que os devolvem (*lazy loading*). Desta forma é possível permitir a *deep serialization* destes objectos para *JSON*.

Para evitar que a serialização cíclica de objectos que se referenciam entre si (por exemplo, um evento tem uma lista de `Datespans`, mas o `Datespan` referencia um evento), implementaram-se as

relações apenas numa direcção, centrada no objecto representativo do evento. Ou seja, a partir do objecto que representa o evento é possível obter os outros objectos, mas dos outros objectos não é possível chegar directamente ao evento, devolvendo apenas o ID para esse registo na base de dados. Este processo, utilizando determinadas bibliotecas que automatizam o processo de criação da representação *JSON*, poderia permitir a implementação das funções inversas, aplicando depois exclusões ao acesso a objectos que iriam depois causar uma geração cíclica, mas, por uma questão de evitar eventuais problemas que não estivessem a ser previstos, achou-se que esta seria a forma ideal de resolver o problema.

Associado a este processo, passa também a ser possível que os dados sejam acedidos por outra fonte, servindo também como um API para acesso externo, o que se pode revelar bastante útil para o desenvolvimento de aplicações para plataformas móveis utilizando a informação disponível nesta plataforma.

4.3.2.3 Aleatoriedade dos eventos

A escolha aleatória de eventos é um processo demasiado importante para ser deixado ao acaso.

Para evitar a sobre-exposição de eventos, mais específicamente dos eventos de longa duração, permitindo dar espaço aos eventos que ocorrem num curto espaço de tempo, é necessário efectuar uma escolha baseada no número de dias que o evento já está a decorrer.

A escolha dos eventos para apresentar na página principal está baseada numa sequência, que consiste em:

1. Obter eventos em destaque
2. Obter eventos a decorrer
3. Obter eventos que irão decorrer no futuro

A escolha de eventos em destaque é praticamente directa, sendo obtidos aqueles que estão marcados para destaque no topo da lista. Os eventos a decorrer é que representam um problema.

De forma a que os eventos a decorrer apenas no próprio dia tenham alguma visibilidade, o ideal seria dar uma probabilidade muito maior que a eventos de longa duração, ou mesmo permanente.

A resolução deste problema poderia ser baseada no tempo que o evento vai decorrer, mas isso traria um problema para os eventos permanentes que, sendo representados como eventos sem data limite, não são apropriados para achar o tempo total. Para além disso, a abertura dos eventos costuma ser a altura de maior impacto dos eventos, pelo que seria apropriado que inicialmente houvesse uma maior probabilidade de os eventos que irão ser iniciados no próprio dia aparecerem nos destaques.

A solução foi optar por dar mais probabilidade de divulgação a eventos que estejam a decorrer há menos tempo. O processo matemático é o seguinte:

Fase 1 Objectivo:

Obtenção do somatório do inverso do número de dias em que cada evento está a decorrer

Dados de entrada:

Tempo a decorrer do evento i (A_i)
Número de eventos (n)

Saída:

Máximo valor para aleatório (M)

Fórmula implementada:

$$M = \sum_{i=0}^{n} 1/A_i$$

Fase 2 Neste caso específico é utilizado um algoritmo em pseudo-código para apresentar a solução do problema. Para efeitos de percepção, considera-se que o *array* de diferenças de tempo representado matemáticamente como A_i é indexado igualmente a zero.

Objectivo:

Calcular o número de ordem do evento seleccionado

Dados de entrada:

Tempo a decorrer do evento i (A_i)

Número de eventos (n)

Número aleatório entre 0 e M (R)

Saída:

Número de ordem do evento (N)

Algoritmo implementado:

```
N = 0
x = 1/A[N]
Enquanto R < x:
  N = N + 1
  x = x + 1/A[N]
Fim de Enquanto

Retorna N
```

Este processo utiliza apenas uma representação única para a probabilidade de um evento ser seleccionado para ser apresentado e foi aquilo que acabou por ser implementado na plataforma. É, no entanto, possível adicionar pesos a estas fórmulas através da exponenciação.

5. UCV Mobile

A implementação de plataformas móveis faz parte dos requisitos deste estágio e foi considerado o desenvolvimento de aplicações móveis para as plataformas UCV e Agenda 7, com o objectivo de expandir a divulgação dos conteúdos a este canal, que hoje em dia se trata de uma mercado cada vez mais âmplo.

Inicialmente o objectivo seria desenvolver aplicações para plataformas móveis para suportar o acesso à informação existente na Agenda7. No entanto, devido às iniciativas de expansão impulsionadas pelo sucesso da UCV, optou-se primeiro pelo desenvolvimento de aplicações móveis que suportassem a plataforma nestes dispositivos, uma vez que a plataforma actual foi redesenhada sem ter em conta aspectos específicos destas plataformas.

Foram escolhidas as plataformas móveis iOS e Android por serem aquelas que possuem o maior segmento de mercado móvel com uma quota de mercado conjunta superior a 80%, segundo a *Gartner*[1][2] e a IDC[3][4]

5.1 Requisitos

Os requisitos desta aplicação foram escolhidos à medida que a aplicação foi desenvolvida, devido ao baixo grau de complexidade e à necessidade do estagiário em necessitar de realizar várias experiências relacionadas com as duas plataformas. No final da primeira definição do interface geral surgiram os requisitos a pedido de François Fernandes, membro da equipa de produção de conteúdos. Estes são apresentados na subsecção seguinte

5.1.1 User Stories

A aplicação deverá ser iniciada com um splash screen idêntico em todas as plataformas.

Deverá ter três formas de navegação: uma que apresenta os destaques, uma que apresenta as categorias e uma pesquisa.

A aplicação deverá ter um ecrã que apresenta a informação da UCV.

Para além dos videos deverá ser possível visualizar as descrições dos videos, entre outra informação que se ache apropriada (datas, títulos).

Em tablets, deverão ser apresentados conteúdos relacionados para aproveitar o espaço.

[1] http://www.gartner.com
[2] http://www.engadget.com/2012/08/14/gartner-q2-2012/
[3] http://www.idc.com
[4] http://www.engadget.com/2012/08/08/idc-android-and-ios-continue-to-carve-up-the-world-another-rec/

5.2 Arquitectura

Tal como outras plataformas mencionadas em secções anteriores, o desenvolvimento de plataformas móveis para *iOS* e *Android* têm um elo muito forte com a arquitectura *MVC*, sendo os *SDK*s muito orientados para a utilização desta arquitectura.

Nas figuras 5.1 e 5.2 são apresentados os *storyboards* para as aplicações de *smartphone* e *tablet*, respectivamente.

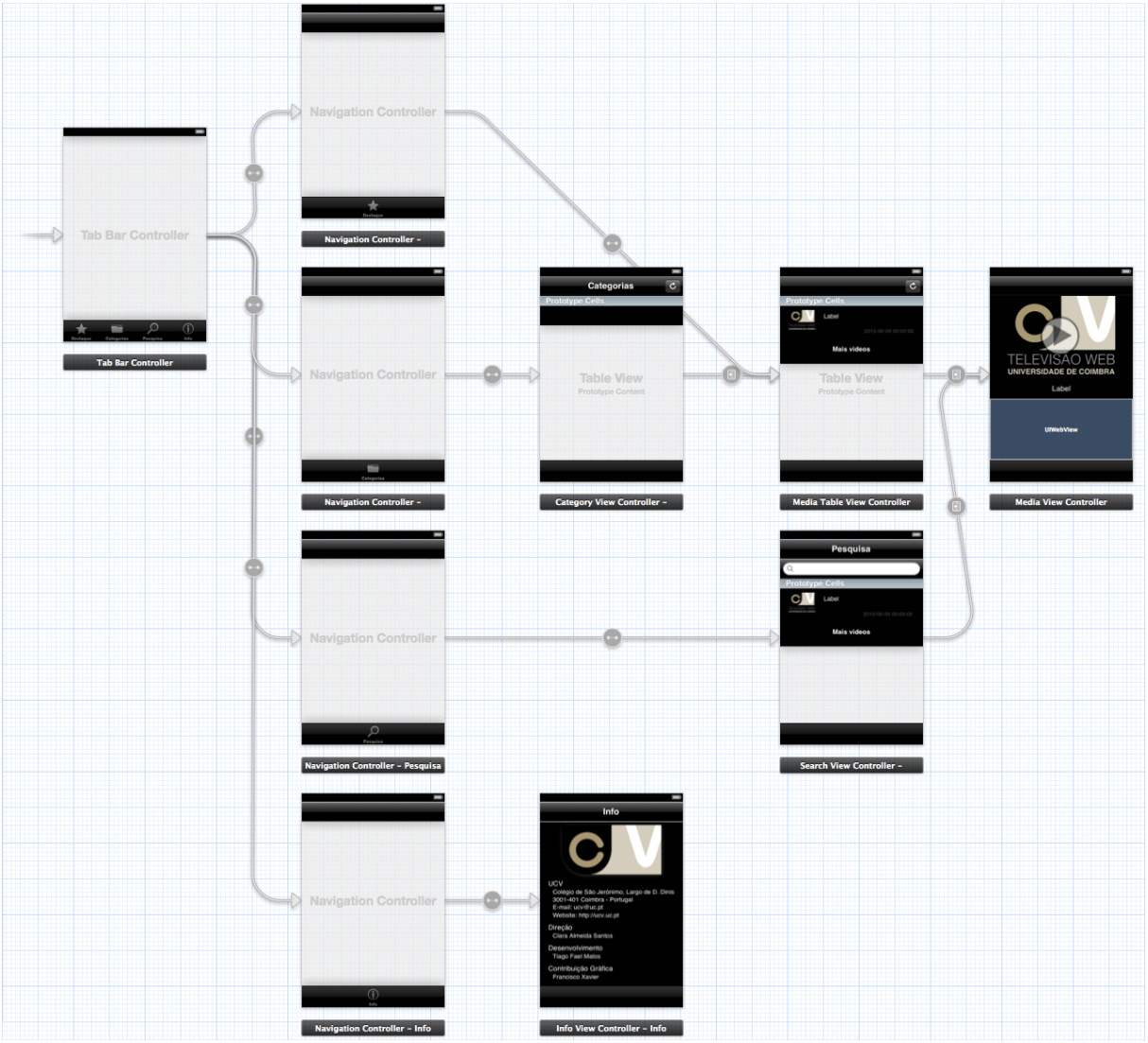


Figura 5.1: Storyboard para a aplicação em smartphones

Apesar de estarem a ser apresentados apenas os storyboards para aplicações de *iOS*, a versão para *Android*, que apenas foi desenvolvida para *smartphones*, utiliza o mesmo esquema de navegação que a app *iOS* para *iPhone* e *iPod Touch*, sendo esse *storyboard* representativo de ambas as aplicações móveis.

5.3 Tecnologia

Para cada uma das plataformas móveis (*iOS* e *Android*) existem diferentes ferramentas para desenvolver aplicações. Esta secção tem como objectivo apresentar as ferramentas existentes

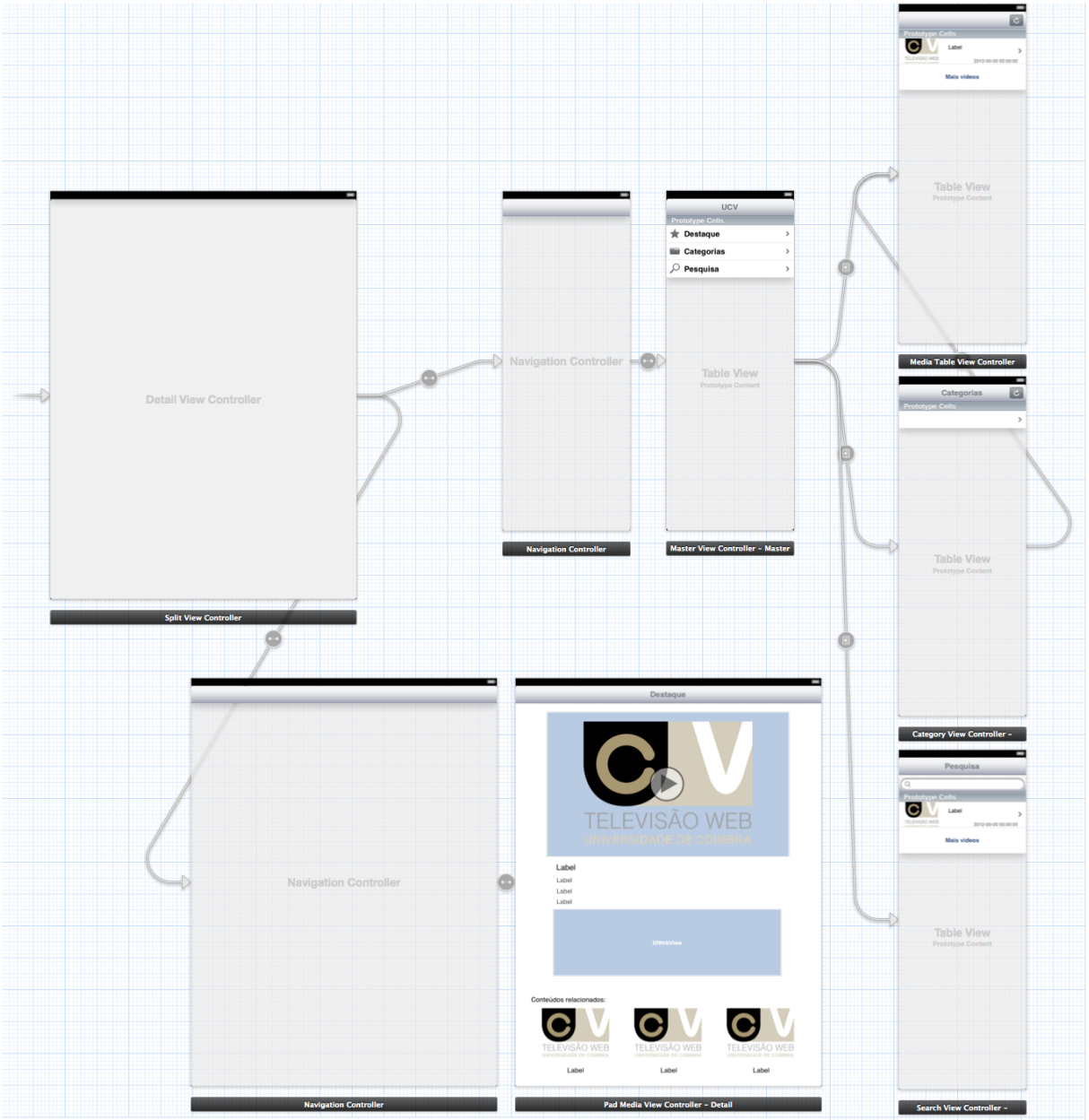


Figura 5.2: Storyboard para a aplicação em tablets

para cada uma das plataformas e as ferramentas escolhidas pelo estagiário para a implementação das aplicações móveis.

5.3.1 iOS

Para o desenvolvimento de aplicações para *iOS*, a *Apple* fornece o *SDK* oficial, bem como as ferramentas de desenvolvimento (*XCode*). As aplicações são desenvolvidas utilizando maioritariamente *Objective-C* e utiliza um conjunto de *frameworks* disponibilizados e documentados pela empresa.

Apesar de existirem outras *frameworks* para desenvolvimento de aplicações móveis e nativas para *iOS*, todas necessitam que a aplicação seja compilada utilizando o *XCode*, por imposição da empresa.

O desenvolvimento de aplicações para *iOS*, bem como a validação e colocação das mesmas para aprovação na *App Store* está completamente integrado com o *XCode*, sendo esta uma ferramenta necessária.

Os testes de aplicações em dispositivos móveis é possível para programadores registados no *iOS Developer Program*[5]. O estagiário já possuía uma licença e um dispositivo compatível, sendo possível efectuar os testes necessários num dispositivo móvel para testar a aplicação após a implementação. Por outro lado, o estagiário não possuía um *iPad*, nem foi possível obter um destes aparelhos por empréstimo para testar o funcionamento da aplicação nesses dispositivos.

As ferramentas fornecidas pela *Apple Inc.* incluem um simulador que permite fazer muitos dos testes de uma forma bastante fiável, tendo sido dessa forma que os testes da aplicação em *iPad* decorreram.

5.3.2 Android

Para o desenvolvimento de aplicações *Android*, existe um *SDK* oficial disponibilizado pela *Google Inc.*, que possui algumas ferramentas de apoio ao desenvolvimento. As ferramentas são todas baseadas na linha de comandos, não sendo disponibilizadas com um *IDE*. A empresa recomenda, no entanto, a utilização do *Eclipse*[6], apesar de ser perfeitamente possível utilizar outros *IDE*s, bem como editores de código simples em conjunto com a linha de comandos.

De forma a conseguir ganhar controlo sobre o projecto a ser desenvolvido, optou-se pela utilização de um editor simples em conjunto com a linha de comandos. Por forma a facilitar a detecção do código pelo editor, facilitando o processo de leitura e implementação de código, foi escolhido um editor com suporte para *CTags*[7], denominado *BBEdit*.

A razão por detrás da escolha, em vez do *IDE* recomendado (*Eclipse*) deve-se ao facto de que o *IDE* toma controlo sobre determinadas partes do desenvolvimento que muitas vezes requerem intervenção directa do programador. Sendo todo o processo do *IDE* baseado na utilização do software existente no *SDK* fornecido pela *Google Inc.*, o processo de desenvolvimento de aplicações para Android sem utilizar o *IDE* é baseado na execução de aplicações incluidas no *SDK*, através da linha de comandos. De facto, existem muitas funcionalidades no *SDK* que não são acessíveis através do *Eclipse*, ou qualquer outro *IDE*, e que implicavam sempre a utilização da linha de comandos para aceder a essas funcionalidades. Desta forma, sem se utilizar o *IDE*, o programador tem que perceber o funcionamento das componentes distribuidas com o *SDK*, havendo uma maior percepção das funcionalidades existentes e dos limites dos testes durante o desenvolvimento que podem ser efectuados sem recorrer a um dispositivo móvel com o sistema operativo *Android*.

De facto, o emulador distribuído com o *SDK* apresenta graves falhas a nível de coerência com um dispositivo real, bem como falhas ao nível de testes de rede. O emulador utiliza uma ligação de rede local para simular uma ligação real com um dispositivo. Quando a internet no dispositivo é desligada (através da configuração de rede), toda a comunicação com as ferramentas do *SDK* termina, deixando de ser possível obter qualquer informação vinda do emulador. O emulador tem ainda problemas na reprodução de videos. Não é, de todo, possível reproduzir um video no emulador, sendo apresentado um ecrã preto.

[5] https://developer.apple.com/programs/ios/
[6] http://www.eclipse.org
[7] http://ctags.sourceforge.net/ctags.html

Devido a estes problemas, efectuar testes à aplicação final tornar-se-ia impraticável sem um dispositivo real. O estagiário não dispunha de nenhum dispositivo com o sistema operativo *Android*, mas foi possível utilizar um dispositivo de um dos membros da equipa de produção de conteúdos para se efectuarem testes esporádicos.

5.4 Implementação

A implementação das aplicações para plataformas móveis tem que ser feita em dois pontos: no desenvolvimento da própria aplicação (para cada um dos sistemas operativos móveis para os quais se pretende desenvolver) e no *API* de acesso.

5.4.1 API da UCV

No caso do *API* de acesso, houve uma avaliação inicial daquilo que já existia no *API* original do *MediaCore*, já permitindo obter grande parte dos dados necessários para a implementação do projecto. No entanto, havia um ponto onde o *API* não foi considerado adequado à utilização em aplicações para dispositivos móveis, mais concretamente, para tablets (no caso específico, para *iPad*).

O que acontece é que o *iPad* é um dispositivo móvel com um ecrã muito maior que os smartphones, sendo possível apresentar uma maior quantidade de informação. Simplesmente implementar uma versão similar à de smartphone iria deixar a aplicação para *iPad* demasiado pobre e com muito espaço do ecrã inutilizado.

A plataforma *MediaCore* já dispõe de meios que permitem relacionar videos semelhantes, o que foi considerado uma mais valia na implementação da aplicação para *iPad*, mas tal não se encontrava implementado no *API*.

Foram efectuadas as alterações necessárias para que tal passasse a ser fornecido pelo *API*, passando o pedido de informação de um conteúdo a ser enviada também com os conteúdos relacionados (com profundidade máxima de um nível).

As alterações efectuadas ao código do *API* podem ser visualizadas no **Anexo 3 - Alterações ao *API* da UCV**.

Este *API* está documentado na página oficial do projecto[8], sendo a sua base construída em torno de um *URL* que permite acesso directo a *controladores* e métodos específicos para os quais se podem enviar parâmetros que filtram a informação desejada. As estruturas possíveis dos endereços estão apresentadas na listagem de código 5.1, onde `<controlador>`, `<metodo>` e `<parametros>` têm que ser substituídos por um controlador, método e parametros aceites pelo *API*, respectivamente.

```
http://ucv.uc.pt/ucv/api/<controlador>?<parametros>
http://ucv.uc.pt/ucv/api/<controlador>/<metodo>?<parametros>
```

Listagem de código 5.1: Estruturas dos endereços de acesso ao *API*

[8] http://mediacorecommunity.org/docs/dev/api.html

5.4.2 Implementação das aplicações móveis

Apesar das diferentes linguagens de programação utilizadas para o desenvolvimento de aplicações para cada uma das plataformas, a arquitectura **MVC**, base de ambos os modelos de desenvolvimento, permite que sejam desenvolvidas aplicações similares baseadas no planeamento da mesma arquitectura base de uma aplicação. Um *controlador* implementado numa das plataformas têm uma correspondência quase directa para outro *controlador* na outra plataforma. Desta forma, esta sub-secção irá começar por se abstrair das linguagens de programação e descrever apenas a estrutura base das aplicações.

A primeira fase de implementação implica discernir quais são as componentes da aplicação que são necessárias implementar. A partir das necessidades em termos de vistas que podemos ver nas figuras 5.1 e 5.2 apresentadas na secção de **Arquitectura** deste projecto, é possível inferir, quase imediatamente, quais os *controladores* necessários para fornecer o conteúdo às vistas, podendo considerar-se que cada vista terá um *controlador*, apesar de ser perfeitamente possível utilizar o mesmo *controlador*, ou implementar um *controlador* específico tendo por base um *controlador* genérico, para vistas similares.

Estes *controladores* têm que ter acesso a dados, que no caso deste projecto, provêm de um servidor web, em *JSON*. Ou seja, consoante o *HTTP Request*, o servidor irá responder com a informação correspondente. Portanto, o *controlador* da plataforma móvel terá que indicar qual o *controlador*, método e parâmetros do *API* da **UCV** que correspondem ao seu pedido.

A componente que irá realizar a chamada ao *API* corresponde precisamente ao modelo de dados do *MVC*, que irá tratar de obter aquilo que for necessário e responder ao *controlador* respectivo, tal como apresentado no esquema da figura 5.3.

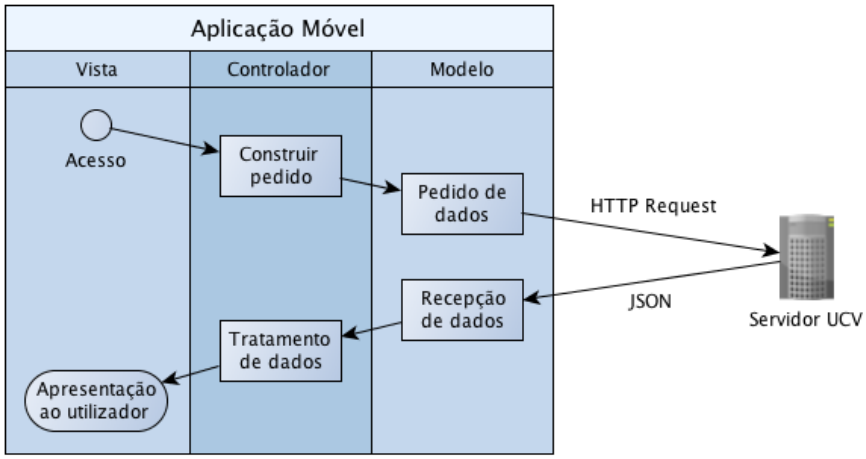


Figura 5.3: Processo de obtenção de dados e apresentação ao utilizador (*MVC*)

5.4.2.1 *Caching* de imagem

Apesar de, em ambas as plataformas, existirem já muitos objectos que implementam muita da funcionalidade necessária, como permitir o *streaming* de video, efectuar a reprodução dos mesmos e apresentar imagens descarregadas da rede, não significa que a capacidade destes dispositivos seja limitada. De facto, ambos os sistemas operativos limitam a quantidade de memória que uma única aplicação pode utilizar de uma vez.

O carregamento das imagens de apresentação representa um ponto importante na gestão de memória dos dispositivos. Como se pode ver na figura 5.4, a listagem de videos, quer sejam os destaques, videos de uma categoria ou videos resultantes de uma pesquisa, será apresentada uma lista ao utilizador que apresenta o título, a data e a imagem de apresentação. As listas são, inicialmente limitadas, mas podem ir sendo expandidas pelo utilizador à medida que vai arrastando a lista para ver mais videos. Isto implica que, a determinado ponto, a quantidade de imagens em memória possa ser tão grande que o sistema operativo é obrigado a desligar a aplicação para impedir que a aplicação ultrapasse os limites máximos.

Figura 5.4: Ecrã de destaques no simulador de *iPhone*

Utilizar um sistema de *caching* de imagens é uma forma adequada de solucionar o problema. Tendo um número limitado de imagens em memória, permite evitar o seu uso excessivo e impedir que a aplicação seja desligada pelo sistema operativo. Mas não é uma solução completa se considerar-mos que as redes móveis são, normalmente, lentas e limitadas em termos de consumo.

A solução completa consiste em utilizar o armazenamento persistente do dispositivo para armazenar as imagens que se vão obtendo.

5.4.2.2 Reutilização de células

Tanto no *Android* como no *iOS* existe um objecto do interface em comum que funcionam sensivelmente da mesma forma da mesma forma. Como se pode ver na figura 5.4, a listagem de videos é feita num objecto que é denominado `UITableView` e `ListView` em *iOS* e *Android*, respectivamente. Estes objectos apresentam os dados em células que são denominadas

por `UITableViewCells` no *iOS*, e que não têm uma designação no *Android*, por serem objectos definidos pelo programador.

Em qualquer momento de utilização da aplicação numa destas vistas, apenas são visíveis um determinado número de células que cabem na totalidade do ecrã. De forma a aproveitar recursos, estas listas ou tabelas foram convenientemente arquitectadas para quando uma célula desaparece completamente da zona visível da lista, em vez de se criar uma nova célula (ou novo objecto), o conteúdo desta é substituído pelo conteúdo da nova célula que iria aparecer.

No âmbito deste projecto, este modo de operação tem algumas implicações a nível de implementação que têm que ser solucionados. Sabendo que todas as operações de pedido por imagens não são imediatas, as imagens não vão poder ser carregadas no exacto momento em que o utilizador acede à vista com a listagem de videos. Pedir as imagens uma a uma não é um processo viável, porque isso implicaria que a aplicação teria que ficar parada até receber as imagens que teriam que aparecer nas células.

A forma mais correcta para permitir o carregamento de imagens eficiente é utilizar uma imagem predefinida, que se mantém até que a imagem chegue ao dispositivo. No momento em que uma célula aparece no ecrã, verifica se a imagem que pretende já está em *cache*, caso não esteja, é criado um processo de baixa prioridade (que não afecta a fluidez do interface) para receber a imagem, fornecendo a referência para o objecto. Este processo está apresentado na figura 5.5.

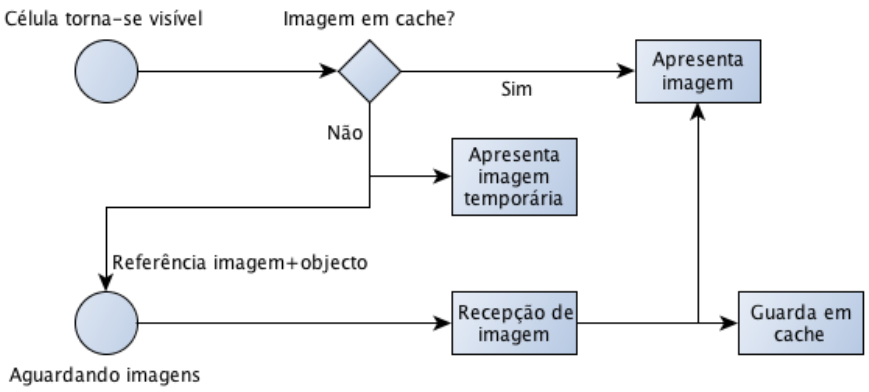


Figura 5.5: Processo de pedido de imagens

Infelizmente, este processo não é suficiente para garantir que as imagens são apresentadas correctamente. Se o utilizador começar a mexer na lista antes de algumas imagens serem carregadas, as células que haviam pedido anteriormente as imagens vão ter as suas referências registadas num processo de baixa prioridade, que eventualmente irá receber a imagem que foi pedida anteriormente e colocá-la na célula de outro video. Para piorar ainda mais o problema, não existem garantias de as imagens sejam recebidas ordenadamente, podendo o segundo pedido de uma célula chegar primeiro que o primeiro pedido efectuado, substituindo a imagem da célula permanentemente pela imagem errada, até que o utilizador volte a mexer a lista, obrigando à actualização das células.

A solução final consiste em ter um segundo registo global (acessível por todos os processos de baixa prioridade) que faz o mapeamento do objecto para a imagem. Quandouma célula aparece, para além de ser feito o pedido pela imagem, é verificado se a referência para o objecto já se encontra num mapa de imagens, caso se encontre, a imagem que é suposto obter é alterada.

Quando um processo acaba de obter a imagem, verifica nesse registo se a imagem pretendida ainda é a mesma. Caso não seja, a imagem apenas é guardada na cache para futura utilização. Estas alterações estão esquematizadas na figura 5.6.

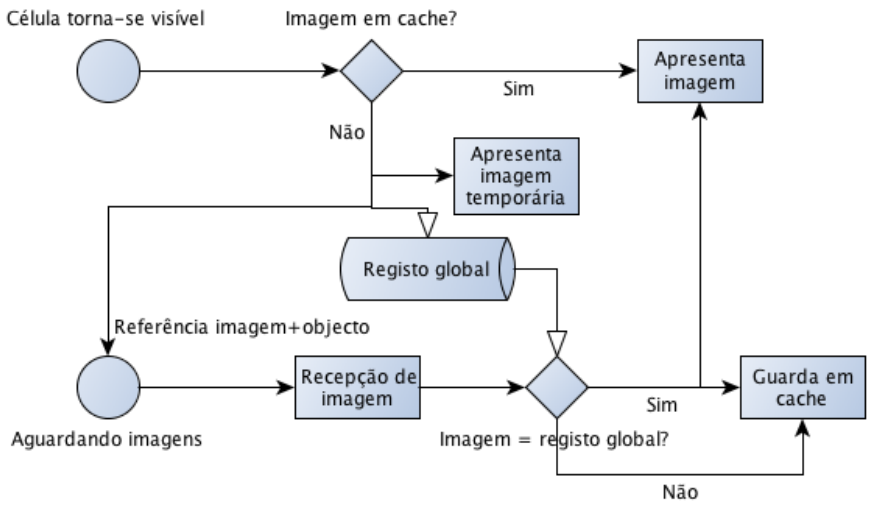


Figura 5.6: Processo de pedido de imagens com mapeamento inverso

5.4.3 Redireccionamento do browser para a aplicação

De forma a fomentar o uso da aplicação móvel em vez do acesso directo à plataforma através do browser móvel, foi implementado uma pequena função em *JavaScript* para detectar o dispositivo que acede à página, verificar se a aplicação móvel se encontra instalada e redireccionar para a aplicação móvel caso esteja instalada ou para a página de download da mesma, caso não esteja instalada.

Para não obrigar o utilizador a utilizar a aplicação, possibilitando a utilização do portal como está, é feita a apresentação de uma mensagem ao utilizador, questionando se pretende utilizar a aplicação móvel. Caso o utilizador responda afirmativamente, é guardado um cookie que impede que a mesma pergunta seja realizada a cada pedido por uma nova página da plataforma. Caso o utilizador responda afirmativamente:

Se a aplicação estiver instalada redirecciona para a aplicação com a informação acerca do video que o utilizador esperava visualizar, utilizando um *URL Schema* (apenas funciona no *iOS*, nesta altura);

Se a aplicação não estiver instalada redirecciona para a página onde se encontra a aplicação, que irá fazer com que o aparelho abra a app do mercado de aplicações correspondente apresentando a aplicação móvel desenvolvida pronta para ser instalada.

O redireccionamento directo para a aplicação foi uma funcionalidade implementada apenas no *iOS*, devido à possibilidade de utilizar um *URL Schema* para comunicar com a aplicação móvel através do browser.

O *URL Schema* é um endereço formatado da mesma forma que os endereços **HTTP**, com um nome diferente para o protocolo. Na listagem de código 5.2 é apresentada a estrutura de um *URL Schema*.

```
ucv://ucv/podcasts/stub-categoria/stub-media
```

Listagem de código 5.2: Estrutura de um *URL Schema* para a aplicação móvel da **UCV**

A informação do caminho que é enviada no *URL* permite ao *controlador* determinar qual o video a apresentar ao utilizador.

O código implementado para esta função pode ser visto no **Anexo 4 - Redireccionamento para app (UCV)**.

6. Planeamento

Os objectivos para este estágio consistiam em dar continuidade ao meu envolvimento nos projectos em que já participava anteriormente, continuando a realizar a manutenção e implementar funcionalidades nos sistemas que havia instalado e desenvolvido antes do estágio. Dentro desse objectivo, é implicita a necessidade de continuar a fornecer meios que permitam a extensão da divulgação da informação e dos conteúdos que estas plataformas fornecem, ampliando o alcance das mesmas através de diversos canais de comunicação que fossem considerados necessários ou que pudessem ser adequados às plataformas em causa.

No primeiro semestre, todo o trabalho que teria que realizar estava praticamente delineado, não existindo grande margem para introduzir outras tarefas que fossem necessárias.

Na figura 6.1 é apresentado o plano de trabalho que defini para cada uma das tarefas (baseline, a cinzento) e o tempo que acabei por utilizar para resolver cada uma das tarefas, bem como os desvios de algumas tarefas em relação ao plano inicial.

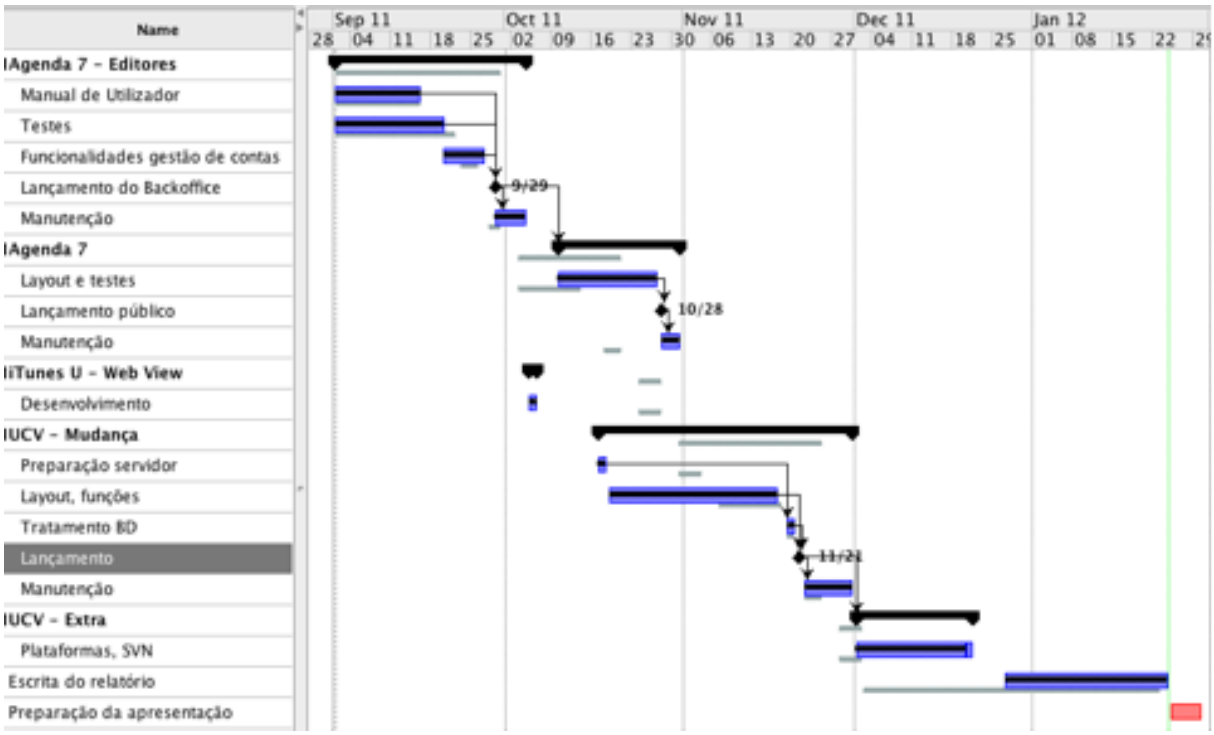


Figura 6.1: Plano de trabalho e trabalho realizado no primeiro semestre

Relativamente ao trabalho a desenvolver no segundo semestre, não existiam nenhumas obrigações da parte do estagiário na participação nos projetos que não fosse para além da contínua manutenção dos sistemas.

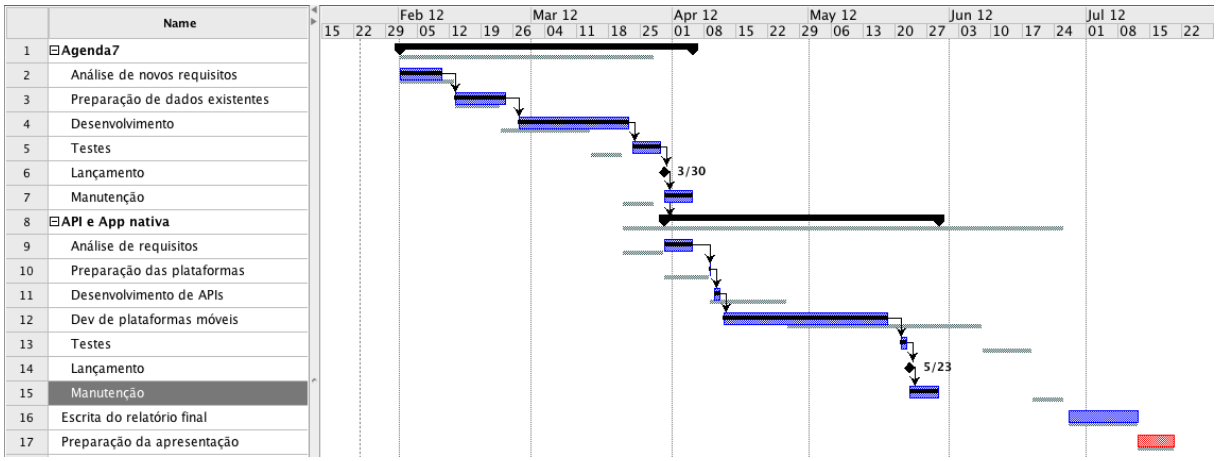


Figura 6.2: Plano de trabalho e trabalho realizado no segundo semestre

Como tal, e visto que a proposta de estágio inclui desenvolvimento de aplicações para plataformas móveis, o trabalho que ficou definido para o segundo semestre era a alteração da **Agenda7** para novos requisitos que foram sendo pedidos pelos parceiros e administradores da plataformas, para além das alterações que o estagiário considerou necessários de forma a permitir a escalabilidade da plataforma.

Esta tarefa tinha como objectivo estar terminada em meados do mês de Março, no entanto, devido a alterações intermédias que foram necessárias à plataforma existente para apresentação dos eventos pertencentes à Semana Cultural da Universidade de Coimbra, a implementação destas funcionalidades teve que ser adiada, obrigando a que a implementação do projecto só estivesse concluida no final do mês de Março.

Por causa do atraso anteriormente mencionado, o projecto de implementação de aplicações móveis para a **Agenda7** e **UCV** foi encurtado, envolvendo apenas a implementação das aplicações móveis para a **UCV**, visto ser uma plataforma com maior sucesso aparente. Por esta razão, os tempos necessários para levantamento de requisitos e implementação de alterações ao *API* foi menor do que o esperado.

A primeira aplicação móvel, com suporte nativo para *iPod Touch* e *iPhone* demorou muito pouco tempo a ser implementada. No total, foram necessários dois dias para desenvolver uma aplicação móvel para estes dispositivos devido ao facto de o estagiário já ter experiência no desenvolvimento de aplicações móveis para esta plataforma (e para estes dispositivos, mais especificamente). O desenvolvimento desta aplicação também permitiu facilitar o desenvolvimento da estrutura da aplicação para *Android*, devido a utilizar uma arquitectura bastante similar.

A adequação da plataforma para *iPad* demorou cerca de uma semana, devido à necessidade de estudar o funcionamento da componente que subdivide o ecrã em duas partes e para tornar o código previamente desenvolvido reutilizável na versão de *iPad*, tornando a aplicação *Universal*[1].

O desenvolvimento da aplicação móvel para *Android* demorou mais tempo que as anteriores, por múltiplas razões:

- O estagiário não tinha experiência prévia no desenvolvimento para esta plataforma;

[1] http://devimages.apple.com/iphone/resources/introductiontouniversalapps.pdf

- As ferramentas de desenvolvimento não tinham um nível de integração comparável com as ferramentas para *iOS*;
- O emulador da plataforma não funciona consistentemente, para além de apresentar dificuldades ao nível de testes de rede.

A aplicação para *Android* foi sendo desenvolvida e foram sendo consideradas novas funcionalidades que foram sendo adicionadas a ambas as aplicações, com o lançamento final das mesmas nas respectivas plataformas de distribuição a serem efectivamente finalizadas a 23 de Maio.

Finalmente, devido à necessidade de preparar novas plataformas para arquivo de videos e divulgação de serviços para os projectos *iTunes U* e **UCV**, a escrita deste relatório acabou por ter que ser adiada para a segunda fase de apresentação, não estando apresentadas como concluídas no diagrama de Gantt apresentado na figura 6.2.

7. Conclusões

Este capítulo apresenta as principais conclusões sobre o que foi desenvolvido no estágio relatado neste documento.

7.1 Resultados para a Universidade

Os objectivos principais do trabalho desenvolvido neste estágio eram:

- Garantir a manutenção e contínua evolução das plataformas de suporte aos três projectos;
- Expandir ou possibilitar a expansão dos canais de divulgação dos conteúdos destas plataformas;

Dentro destes objectivos, consideram-se que os seguintes objectivos para cada plataforma foram cumpridos:

- **UCV**:

  Migração da plataforma para um servidor com capacidade de armazenamento adequada;

  Actualização da plataforma para a versão mais recente das tecnologias;

  Expansão da plataforma a dispositivos móveis e redes sociais;

- Projecto do *iTunes U* da Universidade de Coimbra:

  Expansão dos conteúdos aos browsers web;

- **Agenda7**:

  Adicionar novas funcionalidades, permitindo melhorar a qualidade da informação fornecida pela plataforma;

  Melhorar a escalabilidade da plataforma;

  Expandir a plataforma às redes sociais;

  Possibilitar a expansão da plataforma a dispositivos móveis.

Exceptuando o desenvolvimento das aplicações móveis para a **Agenda7**, os objectivos relacionados com a manutenção e expansão das plataformas desenvolvidas neste estágio foram cumpridos, possuindo agora novos canais que podem permitir uma divulgação mais alargada, permitindo que estes projectos possam efectivamente apoiar as iniciativas estratégicas estabelecidas no pilar da **Transferência do Conhecimento** do seu **Plano Estratégico**:

1. desenvolver uma política cultural ativa e responsável, colocando a UC no mapa nacional e internacional, através do fomento da atividade cultural, artística e desportiva;
2. promover a língua, a cultura e a cidadania lusófonas;
3. promover uma cultura de criatividade e inovação, de empreendedorismo e de espírito crítico;
4. reforçar o apoio à transferência de conhecimento, à gestão da propriedade intelectual e à criação de empresas;
5. posicionar a UC como entidade catalisadora da transferência de conhecimentos;
6. posicionar a UC como referência internacional de inovação, potenciando a participação em redes internacionais.

7.1.1 Resultados para o estagiário

O estagiário considerou que os resultados em termos de aquisição de experiência no desenvolvimento de plataformas web utilizando diferentes *frameworks* bastante positivo. Aquando o início da implementação da **Agenda7**, as *frameworks* web baseadas em *Java* eram, para o estagiário, as mais apropriadas para o desenvolvimento web. Devido ao trabalho desenvolvido em torno da **UCV** e do *MediaCore*, o estagiário apercebeu-se de que a flexibilidade e liberdade que o *Pylons* fornecia aos programadores era uma mais valia que não se encontra em *frameworks* baseadas em *Java*, muito devido à inflexibilidade da linguagem de programação quando comparada com o *Python*.

Parte dos problemas que afectaram a performance do estagiário no desenvolvimento utilizando *Struts* deveu-se, principalmente, ao facto de ter optado pela utilização da linguagem de *templates* original, denominada *OGNL*[1], que se encontra mal documentada e cuja inconsistência parece ser, por vezes, demasiado contra-intuitiva.

O estagiário já tinha experiência no desenvolvimento de aplicações móveis para *iOS*, tendo desenvolvido algumas aplicações para uma startup de aplicações móveis e lançado algumas delas na *App Store*. O desenvolvimento simultâneo de aplicações móveis para *iOS* e *Android* permitiu ao estagiário, não só, ganhar experiência no desenvolvimento de aplicações móveis numa segunda plataforma, como permitiu estabelecer uma comparação entre os modelos de desenvolvimento, que considerou similares, apesar de existirem ainda algumas lacunas no *SDK* de *Android* que necessitam de ser corrigidas para que as ferramentas de desenvolvimento se tornem efectivamente produtivas.

[1] http://commons.apache.org/ognl/

7.1.2 Trabalho futuro

Como ficou claro em secções anteriores, o trabalho desenvolvido não foi completo, ficando por implementar a aplicação móvel para a **Agenda7**. Para além disso, existem algumas funcionalidades que se consideraram adequadas para ambas as aplicações móveis, que permitiriam a utilização das mesmas sem necessidade de acesso à rede, guardando os conteúdos favoritos no dispositivo, e para melhorar a divulgação dos conteúdos através da adequação das redes sociais também através destes dispositivos.

A aplicação de *Android* também não ficou ao mesmo nível que a aplicação móvel de *iPhone*, faltando ainda meios de comunicação entre o browser e a aplicação que permitam que o utilizador possa aceder ao conteúdo que tencionava no browser e ser redireccionado para a aplicação móvel, no conteúdo que iria aceder.

O *iTunes U* ainda utiliza um sistema de gestão primitivo, sendo necessários múltiplos serviços em múltiplos servidores para efectuar o processo de publicação de conteúdos. É possível que, no futuro se possa desenvolver uma aplicação web integrada que permita gerar as feeds, converter os conteúdos e registar toda a informação associada às colecções, deixando de existir a necessidade de utilizar meia dúzia de plataformas para executar cada um destes processos.

8. Referências

[1] Universidade de Coimbra,
URL: http://www.uc.pt

[2] Plano estratégico da UC, 2011-2015,
URL: http://www.uc.pt/planeamento/PE_WEB_09122011.pdf

[3] MediaCore Community,
URL: http://mediacorecommunity.org

[4] MediaCore Documentation,
URL: http://mediacorecommunity.org/docs/

[5] MediaCore Installation Documentation,
URL: http://mediacorecommunity.org/docs/install/upgrade.html

[6] MediaCore API Documentation,
URL: http://mediacorecommunity.org/docs/dev/api.html

[7] MediaCore Supported Browsers,
URL: http://http://support.mediacore.com/customer/portal/articles/99497-what-browsers-are-supported-

[8] iTunes,
URL: http://www.apple.com/itunes

[9] Apache Web Server,
URL: http://httpd.apache.org

[10] Pylons Project,
URL: http://www.pylonsproject.org

[11] SQLAlchemy,
URL: http://www.sqlalchemy.org

[12] Genshi,
URL: http://genshi.edgewall.org

[13] FastCGI,
URL: http://www.fastcgi.com/

[14] Facebook OpenGraph,
URL: https://developers.facebook.com/docs/opengraph/

[15] Facebook Debugger,
URL: https://developers.facebook.com/tools/debug

[16] Feeder,
URL: http://reinventedsoftware.com/feeder/

[17] FreeMarker,
URL: http://freemarker.sourceforge.net

[18] iOS Developer Program,
URL: https://developer.apple.com/programs/ios/

[19] Apple, Universal Apps,
URL: http://devimages.apple.com/iphone/resources/introductiontouniversalapps.pdf

[20] Adobe Flash,
URL: http://www.adobe.com/products/flashplayer.html

[21] Eclipse,
URL: http://www.eclipse.org

[22] BBEdit,
URL: http://www.barebones.com/products/bbedit/

[23] CTags,
URL: http://ctags.sourceforge.net/ctags.html

[24] Velosurf,
URL: http://velosurf.sourceforge.net

[25] Closure Library,
URL: http://closure-library.googlecode.com/svn/docs/index.html

[26] Transifex,
URL: https://www.transifex.com

[27] Facebook,
URL: https://www.facebook.com

[28] Google Plus,
URL: https://plus.google.com

[29] Twitter,
URL: http://www.twitter.com

[30] AddThis,
URL: http://www.addthis.com

[31] Virtualenv,
URL: http://www.virtualenv.org/en/latest/index.html

[32] OGNL,
URL: http://commons.apache.org/ognl/

[33] Gartner,
URL: http://www.gartner.com

[34] IDC,
URL: http://www.idc.com

[35] Mercado móvel segundo a Garner, via Engadget,
URL: http://www.engadget.com/2012/08/14/gartner-q2-2012/

[36] Mercado móvel segundo a IDC, via Engadget,
URL: http://www.engadget.com/2012/08/08/idc-android-and-ios-continue-to-carve-up-the-world-another-rec/

9. Anexo 1 - Códigos de disciplinas do *iTunes U*

Listagem de disciplinas do *iTunes U*

- Art & Architecture - 102
    - Architecture - 102100
    - Art History - 102102
    - Culinary Arts - 102109
    - Dance - 102103
    - Design - 102105
    - Fashion - 102111
    - Film - 102104
    - Interior Design - 102106
    - Media Arts - 102112
    - Music - 102107
    - Photography - 102113
    - Theater - 102108
    - Visual Art - 102110
- Business - 100
    - Economics - 100100
    - Entrepreneurship - 100107
    - Finance - 100101
    - Hospitality - 100102
    - Management - 100103
    - Marketing - 100104
    - Personal Finance - 100105
    - Real Estate - 100106
- Communications & Media - 113
    - Broadcasting - 113101
    - Digital Media - 113102
    - Journalism - 113103
    - Photojournalism - 113104
    - Print - 113105
    - Speech - 113106
    - Writing - 113107
- Engineering - 101
    - Chemical & Petroleum Engineering - 101100
    - Civil Engineering - 101101
    - Computer Science - 101102
    - Electrical Engineering - 101103
    - Environmental Engineering - 101104
    - Mechanical Engineering - 101105

- Health & Medicine - 103
    - Anatomy & Physiology - 103100
    - Behavioral Science - 103101
    - Dentistry - 103102
    - Diet & Nutrition - 103103
    - Emergency Medicine - 103104
    - Genetics - 103105
    - Gerontology - 103106
    - Global Health - 103112
    - Health & Exercise Science - 103107
    - Immunology - 103108
    - Neuroscience - 103109
    - Nursing - 103114
    - Pharmacology & Toxicology - 103110
    - Psychiatry - 103111
    - Radiology - 103113
- History - 104
    - African History - 104104
    - Ancient History - 104100
    - Asia-Pacific History - 104105
    - European History - 104106
    - Medieval History - 104101
    - Middle Eastern History - 104107
    - Military History - 104102
    - Modern History - 104103
    - North American History - 104108
    - South American History - 104109
- Language - 106
    - African Languages - 106100
    - Ancient Languages - 106101
    - Arabic - 106114
    - Chinese - 106113
    - English - 106104
    - French - 106106
    - German - 106107
    - Hebrew - 106115
    - Hindi - 106116
    - Indigenous Languages - 106117
    - Italian - 106108

Japanese - 106118

Korean - 106119

Linguistics - 106109

Other Languages - 106120

Portuguese - 106121

Russian - 106122

Spanish - 106111

Speech Pathology - 106112

- Law & Politics - 116

  Foreign Policy & International Relations - 116100

  Law - 116101

  Local Governments - 116102

  National Governments - 116103

  Political Science - 116104

  Public Administration - 116105

  World Affairs - 116106

- Literature - 107

  Anthologies - 107100

  Biography - 107101

  Classics - 107102

  Comparative Literature - 107106

  Fiction - 107104

  Literary Criticism - 107103

  Poetry - 107105

- Mathematics - 108

  Advanced Mathematics - 108100

  Algebra - 108101

  Arithmetic - 108102

  Calculus - 108103

  Geometry - 108104

  Statistics - 108105

- Philosophy - 114

  Aesthetics - 114100

  Epistemology - 114101

  Ethics - 114102

  Logic - 114105

  Metaphysics - 114103

  Philosophy of Language - 114106

  Philosophy of Religion - 114107

  Political Philosophy - 114104

- Psychology & Social Science - 110
    - Anthropology - 110107
    - Archaeology - 110106
    - Psychology - 110103
    - Social Welfare - 110104
    - Sociology - 110105
- Religion & Spirituality - 115
    - Buddhism - 115100
    - Christianity - 115101
    - Comparative Religion - 115102
    - Hinduism - 115103
    - Islam - 115104
    - Judaism - 115105
    - Other Religions - 115106
    - Spirituality - 115107
- Science - 109
    - Agriculture - 109100
    - Astronomy - 109101
    - Atmosphere - 109102
    - Biology - 109103
    - Chemistry - 109104
    - Ecology - 109105
    - Environment - 109109
    - Geography - 109106
    - Geology - 109107
    - Physics - 109108
- Society - 111
    - African Studies - 111107
    - American Studies - 111108
    - Asia Pacific Studies - 111101
    - Cross-cultural Studies - 111109
    - European Studies - 111102
    - Immigration & Emigration - 111110
    - Indigenous Studies - 111103
    - Latin & Caribbean Studies - 111104
    - Middle Eastern Studies - 111105
    - Race & Ethnicity Studies - 111111
    - Sexuality Studies - 111112
    - Women's Studies - 111106

- Teaching & Learning - 112
    - Curriculum & Teaching - 112100
    - Educational Leadership - 112101
    - Educational Technology - 112106
    - Family & Childcare - 112102
    - Information/Library Science - 112107
    - Learning Resources - 112103
    - Psychology & Research - 112104
    - Special Education - 112105

10. Anexo 2 - Requisitos funcionais (Agenda7)

Requisitos funcionais das alterações à Agenda7

ID	A7-RT-20-001
Relação	A7-US-20-01
Requisito	Criar nova página que apresenta a imagem carregada e permite seleccionar uma porção da imagem com recuso a um plug-in JQuery, com identificação das coordenadas
Implicações	• Caso a imagem seja demasiado grande será redimensionada na página para se adequar ao layout, logo as coordenadas poderão não corresponder • A selecção tem que ser restrita a uma escala apropriada para que se possa adequar ao tamanho utilizado para as imagens de apresentação

ID	A7-RT-20-002
Relação	A7-US-20-01, A7-RT-20-001
Requisito	Criar controlador para a página de selecção da porção da imagem que recebe as coordenadas e redimensiona a imagem
Implicações	• As coordenadas podem não ser válidas • O controlador de upload de imagem de apresentação já realiza o redimensionamento de imagens

ID	A7-RT-20-003
Relação	A7-RT-20-002
Requisito	Remover a função de redimensionamento do controlador de upload de imagem de apresentação
Implicações	-

ID	A7-RT-20-004
Relação	A7-US-20-02
Requisito	Implementação de função JavaScript para filtrar as linhas de tabelas HTML que contenham o texto a ser pesquisado, escondendo as restantes
Implicações	A pesquisa, não sendo efectuada pelo servidor, não é compatível com uma futura utilização da função com diferentes "páginas"de pesquisa

ID	A7-RT-20-005
Relação	A7-US-20-03
Requisito	Adicionar um campo à tabela de eventos da base de dados para guardar o texto de apresentação, limitado em número de caracteres
Implicações	• Este campo estará vazio, o que faria aparecer todos os eventos sem qualquer texto de apresentação • O objecto DTO não possui este campo e precisa de ser alterado • O formulário de inserção de evento precisa de possibilitar inserir a informação deste campo • O template que apresenta a apresentação utiliza a descrição do evento

ID	A7-RT-20-006
Relação	A7-US-20-03, A7-RT-20-005
Requisito	Criar função para migração da base de dados para a nova versão que copia os primeiros N caracteres do campo descrição para o do texto de apresentação
Implicações	A descrição pode conter HTML, o que não pode acontecer no texto de apresentação

ID	A7-RT-20-007
Relação	A7-RT-20-006
Requisito	Criar função para migração da base de dados para a nova versão que permite remover código HTML de um texto
Implicações	• Esta função tem que ser executada antes da inserção da informação no campo do texto de apresentação • O texto não pode ser cortado antes de se retirar o código HTML, caso contrário, algum código HTML pode ficar cortado, não sendo reconhecido pela função

ID	A7-RT-20-008
Relação	A7-US-20-03, A7-RT-20-005
Requisito	Adicionar novo campo ao template do formulário de inserção de eventos para permitir adicionar o texto de apresentação
Implicações	• O limite de caracteres não é suficiente para representar o espaço utilizado pelo texto escrito neste campo, visto que os caracteres têm diferentes tamanhos • O utilizador não tem noção daquilo que aparece na zona de apresentação do evento, que é espacialmente limitada • O controlador do formulário não recebe nem valida a informação deste campo

ID	A7-RT-20-009
Relação	A7-US-20-03, A7-RT-20-008
Requisito	Adicionar uma função JavaScript que apresente o texto escrito neste campo num espaço que representa o espaço final onde o texto de apresentação irá ser mostrado
Implicações	• Deve ser um processo automático, efectuado sempre que o utilizador altera o campo, sem necessitar do input do utilizador • A função deve tratar o campo de forma a apresentar o texto como está, sem permitir a formatação através de HTML

ID	A7-RT-20-009
Relação	A7-US-20-03, A7-RT-20-008
Requisito	Adicionar uma função JavaScript que apresente o texto escrito neste campo num espaço que representa o espaço final onde o texto de apresentação irá ser mostrado
Implicações	• Deve ser um processo automático, efectuado sempre que o utilizador altera o campo, sem necessitar do input do utilizador • A função deve tratar o campo de forma a apresentar o texto como está, sem permitir a formatação através de HTML

ID	A7-RT-20-010
Relação	A7-US-20-03
Requisito	Adicionar uma função JavaScript que apresente o texto escrito neste campo num espaço que representa o espaço final onde o texto de apresentação irá ser mostrado
Implicações	• Deve ser um processo automático, efectuado sempre que o utilizador altera o campo, sem necessitar do input do utilizador • A função deve tratar o campo de forma a apresentar o texto como está, sem permitir a formatação através de HTML

ID	A7-RT-20-011
Relação	A7-US-20-03, A7-RT-20-005
Requisito	Alterar o objecto DTO para ter acesso ao novo campo inserido na base de dados
Implicações	-

ID	A7-RT-20-012
Relação	A7-US-20-03, A7-RT-20-005
Requisito	Alterar o controlador do formulário de evento para receber e validar o texto de apresentação do evento
Implicações	-

ID	A7-RT-20-013
Relação	A7-US-20-03, A7-RT-20-005
Requisito	Alterar o template de apresentação de eventos para utilizar o texto de apresentação em vez da descrição.
Implicações	• Este campo deve apresentar a informação como está, sem aceitar formatação HTML

ID	A7-RT-20-014
Relação	A7-US-20-06
Requisito	Alterar o formulário de inserção de evento para o campo "Descrição"passar a utilizar um campo com um editor WYSIWYG
Implicações	• Este campo deve retornar HTML aquando o HTTP POST para o servidor • O editor deve limitar as formatações possíveis a um pequeno conjunto básico (bold, itálico, sublinhado, entre outros) que não afectem o layout da página onde a descrição é apresentada

ID	A7-RT-20-015
Relação	A7-US-20-07
Requisito	Adicionar ao controlador do formulário de inserção de utilizador uma função para o envio de e-mails com os dados de conta do utilizador
Implicações	• A função não deve bloquear a sessão do utilizador • Não há garantias de que o e-mail é enviado • As passwords têm que ser mantidas em cleartext até à chamada da função, e só depois pode ser convertida num hash

ID	A7-RT-20-016
Relação	A7-US-20-08
Requisito	O controlador da página de detalhes de evento passa a ter a capacidade de verificar as permissões da sessão do utilizador, sendo esta utilizada quando é feito um acesso a um evento não publicado
Implicações	É um controlador fora do contexto do backoffice, sendo derivado de outras classes pai que não têm esta capacidade de origem, obrigando à replicação da função para este controlador

ID	A7-RT-20-017
Relação	A7-US-20-11
Requisito	Alterar a query de pesquisa para ordenar como indicado no requisito **A7-US-20-11**
Implicações	-

ID	A7-RT-20-018
Relação	A7-US-20-12
Requisito	A funcionalidade para mudar as cores das categorias já está implementada
Implicações	Algumas cores utilizadas pelo TAGV são demasiado claras para que haja contraste com o texto branco

ID	A7-RT-20-019
Relação	A7-US-20-12, A7-RT-20-018
Requisito	Implementar uma função no FullCalendar que permite calcular o contraste entre a cor de cada célula/evento e o texto utilizado para apresentar a informação, escolhendo a cor de fundo mais adequada para a cor recebida
Implicações	Diferentes cores de texto no mesmo calendário podem causar algum impacto visual

ID	A7-RT-20-020
Relação	A7-US-20-13
Requisito	Controlador passa a responder com segmentos/blocos de HTML (gerados através de um template) em vez de toda a informação
Implicações	Parte do código gerado pelo controlador contém JavaScript, que é preciso garantir que está funcional aquando a inserção do novo bloco na página

ID	A7-RT-20-021
Relação	A7-US-20-13
Requisito	Apresentar N resultados com um botão no final que permite chamar o controlador de pesquisa para obter mais resultados
Implicações	• O utilizador tem que perceber quando é que já estão a ser obtidos mais resultados • Em momentos de desespero, o utilizador pode clicar várias vezes no botão para obter mais resultados, chamando várias vezes a função

ID	A7-RT-20-022
Relação	A7-US-20-13, A7-RT-20-021
Requisito	Bloquear a função de pedido de mais resultados enquanto o pedido está a ser efectuado até que a informação seja recebida ou até ao timeout
Implicações	• É necessário detectar todos os eventos que possam impedir a obtenção de informação pelo browser para impedir que a função fique bloqueada em caso de, por exemplo, perda de ligação à rede • Pode ser necessário definir o timeout

ID	A7-RT-20-023
Relação	A7-US-20-13, A7-RT-20-021
Requisito	Apresentar uma animação representativa de progresso aquando a busca de informação dos próximos resultados ao servidor
Implicações	-

ID	A7-RT-20-024
Relação	A7-US-20-14
Requisito	Criar uma coluna boleana na tabela de eventos da base de dados para marcar os eventos que pertencem ou não à categoria especial.
Implicações	• Não existe interface para seleccionar quais os eventos que pertencem a esta categoria, sendo este process realizado apenas pelo programador • É necessário alterar a query de pesquisa para possibilitar a procura por eventos que estejam marcados • É necessário alterar o DTO para obter a informação da nova coluna

ID	A7-RT-20-025
Relação	A7-US-20-14, A7-RT-20-024
Requisito	Alterar o controlador pai para permitir a recepção da propriedade para que possa ser passada para as queries de pesquisa de eventos
Implicações	• É necessário alterar a query de pesquisa para possibilitar a procura por eventos que estejam marcados • É necessário alterar o DTO para obter a informação da nova coluna

ID	A7-RT-20-026
Relação	A7-US-20-14, A7-RT-20-025
Requisito	Alterar as queries de pesquisa para receberem o parametro da flag de evento especial
Implicações	É necessário alterar o DTO para obter a informação da nova coluna

ID	A7-RT-20-027
Relação	A7-US-20-14, A7-RT-20-025, A7-RT-20-026
Requisito	Alterar o DTO para receber a informação da flag de categoria especial
Implicações	-

ID	A7-RT-20-028
Relação	A7-US-20-09
Requisito	Adicionar uma nova tabela à base de dados (**Datespan**) para registar múltiplos pares de datas correspondentes aos dias em que um evento decorre. Necessita de uma referência para o registo correspondente na tabela de eventos e as duas datas (início e fim)
Implicações	• Estes dados já se encontram no evento para um periodo, precisam de ser transferidos para esta tabela • Necessita de ser criado um novo DTO para suportar esta tabela • As queries de pesquisa passam a ter que verificar se o evento está a ocorrer em qualquer dos **Datespan** que os referencia

ID	A7-RT-20-029
Relação	A7-US-20-04
Requisito	Adicionar uma flag que regista se um evento é permanente ou não à tabela **Datespan**
Implicações	Caso o evento seja declarado permanente, a data de fim que estiver registada na base de dados tem que ser ignorada, o que deve ser feito nas queries de pesquisa

ID	A7-RT-20-030
Relação	A7-US-20-10
Requisito	Adicionar uma coluna numérica à tabela **Datespan** representativa dos dias da semana em que o evento referenciado por este periodo (e para este periodo) ocorre
Implicações	• Este campo tem que suportar todas as combinações existentes de dias da semana, logo é necessário um campo de 7 bits • As queries de pesquisa que pesquisam por periodos de tempo necessitam de verificar se o evento está a ocorrer, não só dentro do periodo pesquisado, mas também se está a ocorrer nos dias da semana da intersecção resultante entre o periodo de pesquisa e o **Datespan**

ID	A7-RT-20-031
Relação	A7-US-20-09
Requisito	Adicionar uma nova tabela à base de dados (**Timespan**) para registar múltiplos pares de horas correspondentes aos horários em que um evento decorre. Necessita de uma referência para o registo correspondente na tabela **Datespan** e as duas datas
Implicações	• Estes dados já se encontram no evento para um periodo, precisam de ser transferidos para esta tabela, implicando também a mudança de referenciação para o **Datespan** e não para o evento • Necessita de ser criado um novo DTO para suportar esta tabela • As queries de pesquisa que verificam se o evento já fechou passam a necessitar desta tabela para possibilitar esta verificação

ID	A7-RT-20-032
Relação	A7-US-20-05
Requisito	Adicionar uma flag que regista se, para um **Datespan**, o evento ocorre durante todo o dia
Implicações	• Caso o evento ocorra todo o dia, esta flag estará activa, caso contrário, o evento pode não ter horário definido se não existirem **Timespan**s associados a um **Datespan** • Caso o **Timespan** tenha esta flag activa, as queries de pesquisa devem ignorar as datas registadas

ID	A7-RT-20-033
Relação	A7-RT-20-028, A7-RT-20-029, A7-RT-20-030
Requisito	Criação de uma função no script de migração da base de dados para criar um **Datespan** para cada evento e replicando a informação das datas existentes no evento
Implicações	• É necessário remover estes dados da tabela de eventos assim que os testes são completos para evitar situações de coerência e dados deprecados • Os DTOs devem continuar a pedir os dados correctamente para os métodos actuais, mas numa segunda fase deverão deixar de responder com datas concretas para passarem a ser utilizadas referências para DTOs de **Datespan** • A representação dos dias de semana durante a passagem deverá ser a de que ocorre todos os dias da semana • Nenhum **Datespan** pode ser criado nesta fase utilizando a flag de evento permanente

ID	A7-RT-20-034
Relação	A7-RT-20-031
Requisito	Criação de uma função no script de migração da base de dados para criar um **Timespan** para cada **Datespan** e replicando a informação das datas existentes no evento referenciado pos cada **Datespan**
Implicações	• É necessário remover estes dados da tabela de eventos assim que os testes são completos para evitar situações de coerência e dados deprecados • Os DTOs devem continuar a pedir os dados correctamente para os métodos actuais, mas numa segunda fase deverão deixar de responder com datas concretas para passarem a ser utilizadas referências para DTOs de **Datespan** • A representação dos dias de semana durante a passagem deverá ser a de que ocorre todos os dias da semana • Nenhum **Datespan** pode ser criado nesta fase utilizando a flag de evento permanente

ID	A7-RT-20-035
Relação	A7-RT-20-028
Requisito	Criação de um DTO para o **Datespan**
Implicações	É necessário efectuar a alteração das chamadas a métodos do DTO de eventos para que passem a obter os dados directamente nestes objectos

ID	A7-RT-20-036
Relação	A7-RT-20-031
Requisito	Criação de um DTO para o **Timespan**
Implicações	É necessário efectuar a alteração das chamadas a métodos do DTO de eventos para que passem a obter os dados directamente nestes objectos

ID	A7-RT-20-037
Relação	A7-RT-20-036, A7-RT-20-037, A7-RT-20-034, A7-RT-20-033
Requisito	Refactorização do DTO de eventos para passar a recorrer à informação dos DTOs das tabelas **Datespan** e **Timespan** nos casos em que é possível, ou devolver estes quando é necessário um tratamento mais adequado
Implicações	• De forma a permitir a deep serialization de eventos para que se possam criar APIs de acesso REST, é necessário que exista um método que devolva a lista de **Datespan**s que referenciam o evento • Os controladores e templates necessitam de deixar de utilizar o DTO da tabela de eventos para obter temporização • Para facilitar o processo de obtenção de alguma informação feita directamente através do DTO da tabela de eventos, podem existir métodos para achar o primeiro e o próximo **Datespan** a ocorrer, visto que são operações que serão recorrentes • Para evitar carregar sempre todos os objectos relacionados com o evento, todas as chamadas a informação de outras tabelas/DTOs têm que ser feitas com recurso a lazy loading

ID	A7-RT-20-038
Relação	A7-RT-20-037
Requisito	Refactorização de todos os controladores e templates que recorram ao DTO da tabela de eventos
Implicações	Claramente, muitas. Devem ser feitos testes extensivos para validar os resultados obtidos em cada página após a migração de dados ser efectuada, bem como a impiementação dos DTOs que suportam as operações.

ID	A7-RT-20-039
Relação	A7-RT-20-028, A7-RT-20-030, A7-RT-20-031
Requisito	As queries de pesquisa por tempos e datas devem implementar o algoritmo de pesquisa com intersecção de datas e dias da semana
Implicações	-

ID	A7-RT-20-040
Relação	A7-US-20-04, A7-US-20-09, A7-US-20-10
Requisito	Implementar um form e controlador que permite inserir **Datespan**s
Implicações	• A form deverá remover o campo de data de fim do evento caso a flag de permanente seja activada • Tanto a form como o controlador devem validar que o utilizador escolheu, pelo menos um dia da semana • O controlador tem que validar se não existem sobreposições com outros **Datespan**s do mesmo evento (através de queries de pesquisa)

ID	A7-RT-20-041
Relação	A7-US-20-05, A7-US-20-09
Requisito	Implementar um form e controlador que permite inserir **Timespan**s
Implicações	A form deverá remover os campos de horas caso a flag que indica que ocorre todo o dia seja activada

ID	A7-RT-20-042
Relação	A7-US-20-04, A7-US-20-09, A7-US-20-10
Requisito	O controlador que permite publicar um evento tem agora que validar se o evento a ser publicado ocorre nalgum periodo (tem, pelo menos, um **Datespan**s
Implicações	• Os templates devem passar a informação da razão pela qual o evento não pode ser publicado

11. Anexo 3 - Alterações ao *API* da UCV

Alterações efectuadas ao código do *API* do *MediaCore* para passar a retornar conteúdos relacionados

Adição de conteúdos relacionados à informação disponibilizada pelo *API* da **UCV**, marcadas com o comentário de alterações efectuadas pelo estagiário.

```
    def _info(self, media, podcast_slugs=None, include_embed=False):
        """
        Return a **media_info** dict--a JSON-ready dict for describing a
            media instance.

        :rtype: JSON-ready dict
        :returns: The returned dict has the following fields:

            author (unicode)
                The name of the
                :attr:`author <mediacore.model.media.Media.author>` of
                    the
                media instance.
            categories (dict of unicode)
                A JSON-ready dict representing the categories the media
                instance is in. Keys are the unique
                :attr:`slugs <mediacore.model.podcasts.Podcast.slug>`
                for each category, values are the human-readable
                :attr:`title <mediacore.model.podcasts.podcast.Title>`
                of that category.
            id (int)
                The numeric unique :attr:`id <mediacore.model.media.Media
                    .id>` of
                the media instance.
            slug (unicode)
                The more human readable unique identifier
                (:attr:`slug <mediacore.model.media.Media.slug>`)
                of the media instance.
            url (unicode)
                A permalink (HTTP) to the MediaCore view page for the
                    media instance.
            embed (unicode)
                HTML code that can be used to embed the video in another
                    site.
            title (unicode)
                The :attr:`title <mediacore.model.media.Media.title>` of
                the media instance.
            type (string, one of ['\%s', '\%s'])
                The :attr:`type <mediacore.model.media.Media.type>` of
                the media instance
            podcast (unicode or None)
                The :attr:`slug <mediacore.model.podcasts.Podcast.slug>`
                    of the
                :class:`podcast <mediacore.model.podcasts.Podcast>` that
                the media instance has been published under, or None
            description (unicode)
                An XHTML
                :attr:`description <mediacore.model.media.Media.
                    description>`
                of the media instance.
            description_plain (unicode)
                A plain text
                :attr:`description <mediacore.model.media.Media.
                    description_plain>`
                of the media instance.
            comment_count (int)
                The number of published comments on the media instance.
            publish_on (unicode)
                The date of publishing in "YYYY-MM-DD HH:MM:SS" (ISO
                    8601) format.
                e.g.  "2010-02-16 15:06:49"
```

```
        likes (int)
            The number of :attr:`like votes <mediacore.model.media.
                Media.likes>`
            that the media instance has received.
        views (int)
            The number of :attr:`views <mediacore.model.media.Media.
                views>`
            that the media instance has received.
        thumbs (dict)
            A dict of dicts containing URLs, width and height of
            different sizes of thumbnails. The default sizes
            are 's', 'm' and 'l'. Using medium for example::

                medium_url = thumbs['m']['url']
                medium_width = thumbs['m']['x']
                medium_height = thumbs['m']['y']
    """
    if media.podcast_id:
        media_url = url_for(controller='/media', action='view', slug=
            media.slug,
                            podcast_slug=media.podcast.slug,
                                qualified=True)
    else:
        media_url = url_for(controller="/media", action="view", slug=
            media.slug,
                            qualified=True)

    if media.podcast_id is None:
        podcast_slug = None
    elif podcast_slugs:
        podcast_slug = podcast_slugs[media.podcast_id]
    else:
        podcast_slug = DBSession.query(Podcast.slug)\
            .filter_by(id=media.podcast_id).scalar()

    thumbs = {}
    for size in config['thumb_sizes'][media._thumb_dir].iterkeys():
        thumbs[size] = thumb(media, size, qualified=True)

    """ begining of changes by Tiago Fael Matos """
    related_media = []
    for r in list(Media.query.related(media)[:3]):
        r_thumbs = {}
        for size in config['thumb_sizes'][r._thumb_dir].iterkeys():
            r_thumbs[size] = thumb(r, size, qualified=True)
        r_media = dict(
            id = r.id,
            slug = r.slug,
            title = r.title,
            thumbs = r_thumbs,
        )
        related_media.append(r_media)
    """ end of changes by Tiago Fael Matos """

    info = dict(
        id = media.id,
        slug = media.slug,
        url = media_url,
        title = media.title,
        author = media.author.name,
        type = media.type,
        podcast = podcast_slug,
        description = media.description,
```

```
            description_plain = media.description_plain,
            comment_count = media.comment_count_published,
            publish_on = unicode(media.publish_on),
            likes = media.likes,
            views = media.views,
            thumbs = thumbs,
            categories = dict((c.slug, c.name) for c in list(media.
                categories)),
            related = related_media, """added by Tiago Fael Matos"""
        )

        if include_embed:
            info['embed'] = unicode(helpers.embed_player(media))

        return info
```

12. Anexo 4 - Redireccionamento para app (UCV)

Função de redireccionamento da página web para a aplicação móvel

```
var mobile_redirect = getCookie("no_mobile_redirect");
if (mobile_redirect == null) {
        if ((navigator.userAgent.match(/iPhone/i)) || (navigator.
            userAgent.match(/iPod/i)) || (navigator.userAgent.match(/iPad/
            i))) {
                if (confirm("Esta p&aacute;gina n&atilde;o est&aacute;
                    preparada para este dispositivo. Deseja utilizar a
                    aplica&ccedil;&atilde;o m&oacute;vel?")) {
                        window.location = "ucv:/" + window.location.
                            pathname;

                        var clickedAt = +new Date;

                        setTimeout(function() {
                                // To avoid failing on return to
                                    MobileSafari, ensure freshness!
                                if (+new Date - clickedAt < 2000){
                                        window.location = "http://itunes.
                                            apple.com/us/app/ucv/
                                            id516297795?ls=1&amp;mt=8";
                                }
                        }, 500);
                }
                else {
                        setCookie("no_mobile_redirect", "1", 1);
                }
        }

        var ua = navigator.userAgent.toLowerCase();

        var isAndroid = ua.indexOf("android") > -1; //&amp;&amp; ua.
            indexOf("mobile");
        if (isAndroid) {
                // Do something!
                // Redirect to Android-site?
                if (confirm("Esta p&aacute;gina n&atilde;o est&aacute;
                    preparada para este dispositivo. Deseja utilizar a
                    aplica&ccedil;&atilde;o m&oacute;vel?")) {
                        window.location = 'https://play.google.com/store/
                            apps/details?id=pt.uc.ucv';
                }
                else {
                        setCookie("no_mobile_redirect", "1", 1);
                }
        }
}

function getCookie(c_name) {
        var i = 0, x, y, ARRcookies = document.cookie.split(";");

        for (i = 0; i < ARRcookies.length; i++) {
                x = ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));
                y = ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);
                x = x.replace(/^\s+|\s+\$/g,"");
                if (x == c_name) {
                        return unescape(y);
                }
        }
}

function setCookie(c_name,value,exdays) {
        var exdate = new Date();
        exdate.setDate(exdate.getDate() + exdays);
```

```
        var c_value = escape(value) + ((exdays == null) ? "" : "; expires
            =" + exdate.toUTCString());
        document.cookie = c_name + "=" + c_value;
}
```

13. Anexo 5 - Manual de utilizador da Agenda7

Manual de utilizador da primeira versão da Agenda7

MANUAL DE UTILIZADORES DA PLATAFORMA

AGENDA7

1ª Versão - Coimbra, Setembro 2011

Universidade de Coimbra

ÍNDICE

NOTA INTRODUTÓRIA

Este Manual de Utilizadores da "Agenda 7" configura uma versão ainda incipiente, e manterá o carácter inacabado até que se entenda dar por testado e concluído o novo formato editorial da plataforma.

Pretende-se:

1) Normalizar procedimentos de utilização da plataforma a ser utilizada pelos vários responsáveis pela edição de conteúdos da Agenda com vista à sua coerência e harmonia;

2) Gerar normas que permitam a convergência do trabalho e dos procedimentos dos futuros responsáveis pela gestão de conteúdos da Agenda Cultural relativamente aos objectivos de *inteligibilidade e eficácia de informação* disponibilizada através da Agenda.

ESTRUTURA DA AGENDA 7

Quando acedemos à homepage da Agenda 7 (fig.1) podemos visualizar a informação disponibilizada pela categoria todas as categorias bem como aceder à informação associada às outras categorias configuradas na estrutura do portal.

A categoria geral foi entretanto substituída pela denominação todas as categorias.

Figura 1

A página tem uma opção de Pesquisa (Normal ou Avançada) que permite explorar a informação contida na plataforma – presente e passada – a partir de palavras-chave.

A homepage apresenta ainda um calendário (fig. 2) onde pode ser visualizado o número de eventos que ocorrem por dia. As cores variam em função das diferentes categorias.

Figura 2

O Calendário permite ao utilizador realizar diferentes tarefas:

- Visualizar eventos passados e eventos programados para o futuro próximo através da utilização das setas  que precedem a informação mês/ ano;

- Seleccionar a opção Expandir calendário que permite visualizar a versão expandida, de página inteira, deste módulo de informação (fig.3);

- Seleccionar um ou mais eventos associados a uma data clicando no botão colorido existente por baixo da data. No caso de ser seleccionada uma data em que estejam publicados vários eventos, o calendário será apresentado na sua versão expandida (fig.3);

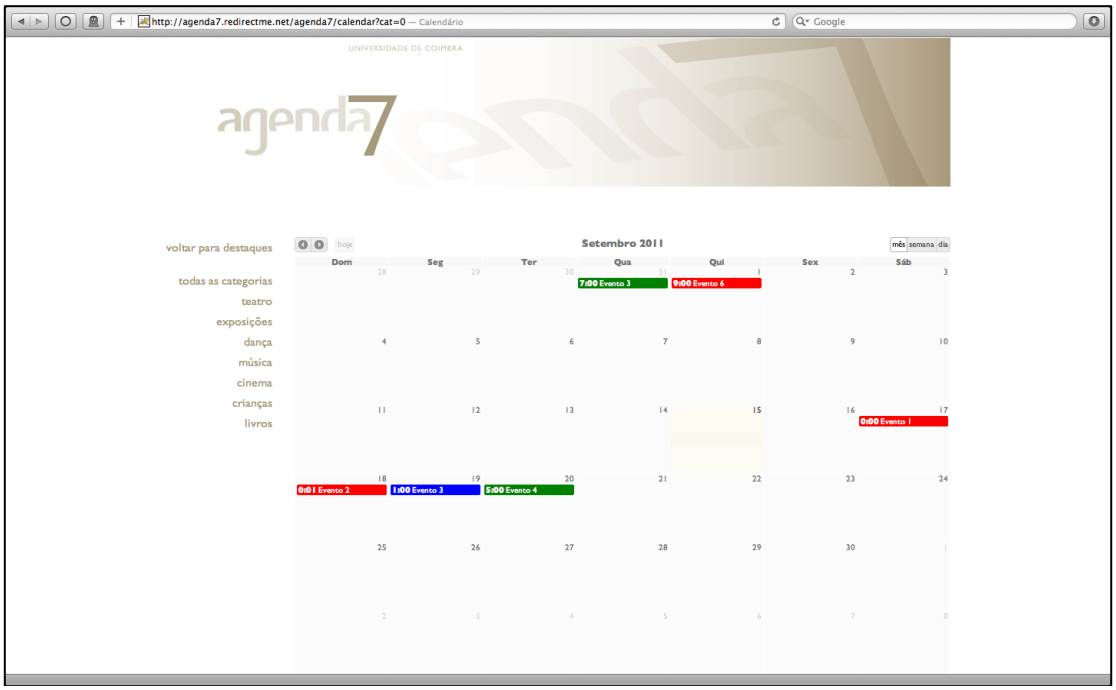


Figura 3

- Seleccionar a opção Embutir calendário que permite aos gestores de conteúdo importar o calendário para as suas páginas institucionais e aos leitores em geral importar a informação para as suas páginas web.

A partir da página inicial podemos aceder a várias categorias de informação - teatro, exposições, dança, música, cinema, ciências, crianças, livros - e aos campos a estas associadas.

A título de exemplo apresenta-se (fig. 4) a informação disponibilizada pela categoria teatro.

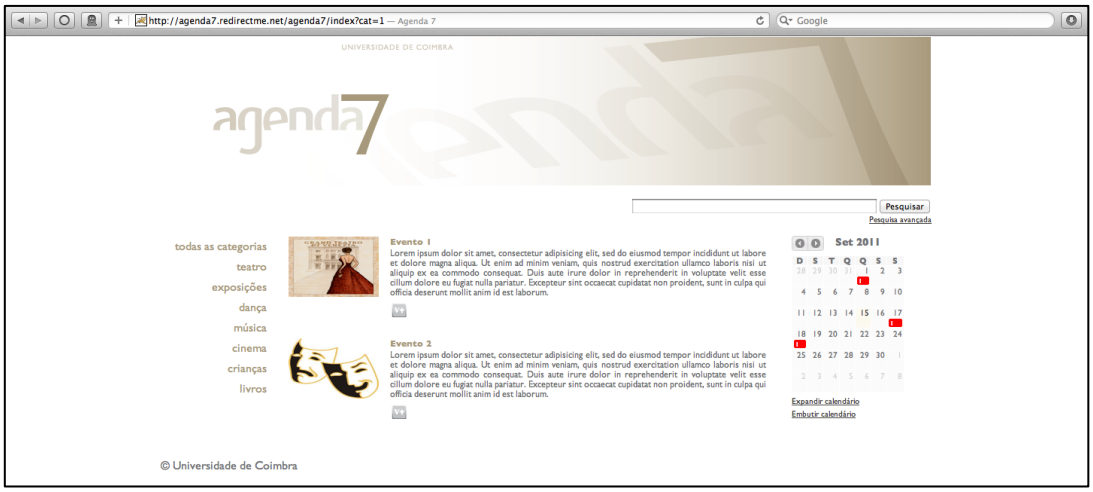


Figura 4

Seleccionando a categoria **teatro** o utilizador terá acesso aos seguintes blocos de informação:

- Imagem de divulgação do Evento ou, no caso de a imagem não ser disponibilizada pela organização, uma imagem genérica referente à categoria (em produção);
- Título do Evento;
- *Lead* (parágrafo em que se descreve a informação essencial sobre o evento, fornecido pela organização);

O Botão V+ que aparece nas figuras 1 e 4 dá acesso a nova página (fig. 5) com a informação detalhada sobre evento. A configuração (layout) desta página ainda não é a versão definitiva.

Figura 5

Nesta(s) página(s) podem ser consultadas:

- As datas, os horários e os locais do evento. A identificação do local pode ainda ser realizada através da visualização de mapas com sinalização do local onde decorre evento;
- As instituições organizadoras dos eventos e respectivas páginas web;
- Os vários preços de ingresso no caso de existir;
- Outros detalhes relacionados com as actividades como, por exemplo, a necessidade de inscrição prévia;

Pode ainda ser consultada a galeria dos Media associados ao evento que consta de três módulos de informação: imagens, áudio e vídeo.

ESTRUTURA DO BACKOFFICE

Para aceder ao backoffice da Agenda 7 (fig. 6) deverá digitar o seguinte endereço: http://agenda7.uc.pt/agenda7/backoffice.

A página inicial do backoffice apresenta a configuração representada na (fig.6):

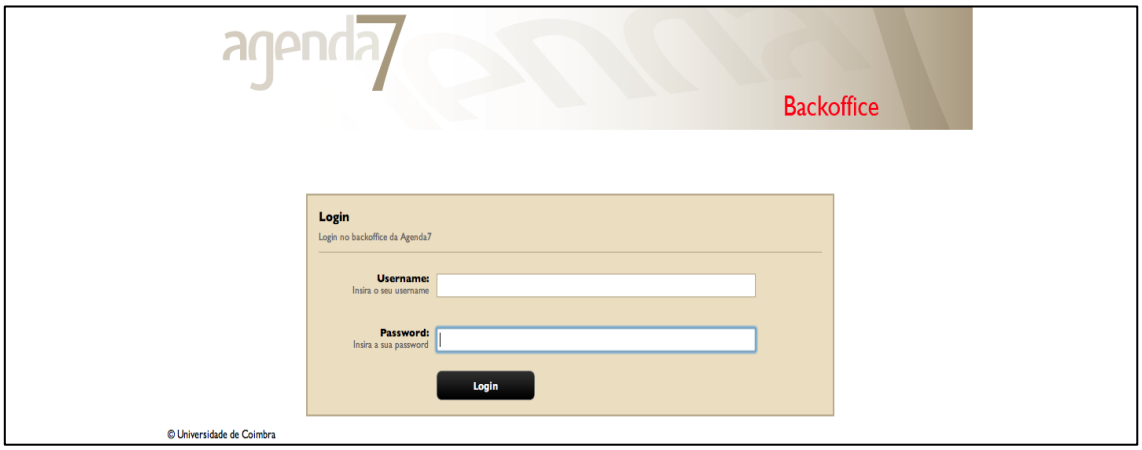


Figura 6

Deverá preencher os campos constantes desta página de acordo com as seguintes directrizes:

Username – Digitar o nome próprio e apelido (s) num limite máximo de 45 caracteres;

Password – Digitar a password que lhe for fornecida pelo administrador da plataforma;

Realizar Login o que lhe dará acesso à página representada na (fig. 7) que lhe permitirá seleccionar as várias opções constantes do menu que configura a Estrutura do Backoffice.

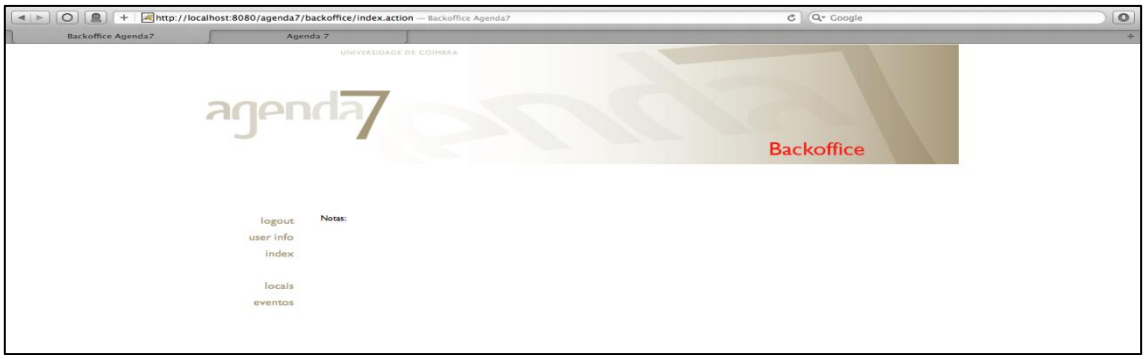


Figura 7

O menu acima representado da página principal do BackOffice apresenta várias opções de navegação:

- logout – Permite sair da conta de utilizador;
- user info – Remete para a página apresentada na (fig.8) onde constam o username, o email do utilizador e a(s) organização(ões) a que este pertence; Nesta página o utilizador tem acesso a dois links que lhe permitem alterar o Email e/ou a Password;

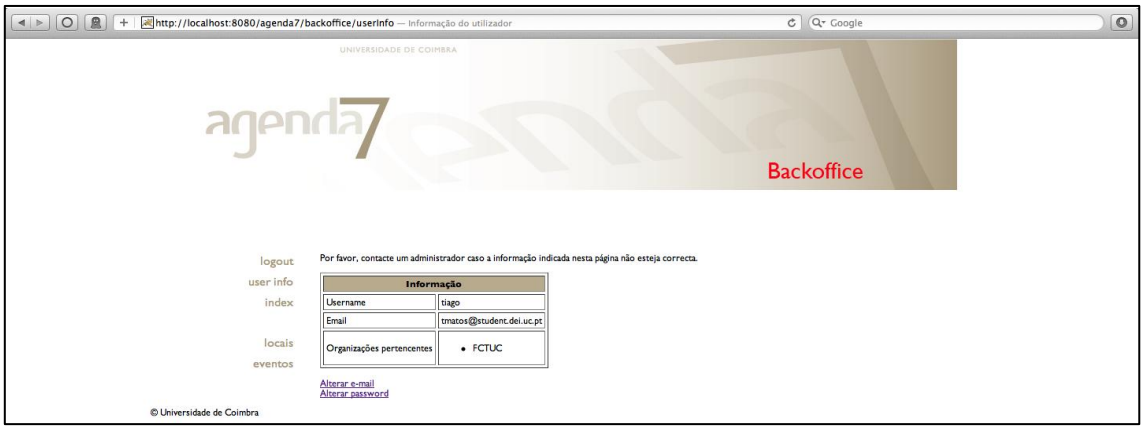


Figura 8

- Index – Permite visualizar a informação constante na página principal do BackOffice como por exemplo notas enviadas pelo administrador do sistema;
- Locais – Serve para inserir locais onde se realizam os eventos. À tarefa local está associada uma tabela onde ficam registados os endereços que vão sendo inseridos. Esta funcionalidade permite, posteriormente, ao utilizador seleccionar um local que já conste da tabela sem ter a necessidade de o introduzir novamente; O modo de preenchimento desta tarefa é explicado no capítulo preenchimento dos campos de informação do Backoffice.
- Eventos – Permite visualizar eventos anteriormente criados pelo utilizador e pelos seus eventuais parceiros. O modo de preenchimento desta tarefa é explicado no capítulo preenchimento dos campos de informação do Backoffice.

PREENCHIMENTO DOS CAMPOS DE INFORMAÇÃO DO BACKOFFICE

O utilizador inicia o preenchimento da informação a partir da barra lateral (menu) da página principal do BackOffice (fig.7).

Clicando na tarefa Locais irá aceder à pagina representada na (fig.9).

Figura 9

Clicando no ícone  irá activar o ecrã seguinte, representado na (fig. 10), onde serão inseridas os dados associados a este bloco de informação.

Se a informação sobre o local constante na tabela estiver incorrecta o utilizador deverá clicar no botão  e proceder à sua correcção na página constante da (fig. 10).

Quando se verificar que existe uma mudança de endereço postal da instituição deverá ser adicionada uma nova entrada e não fazer a correcção do endereço para que os eventos passados não percam a informação sobre o local onde decorreram.

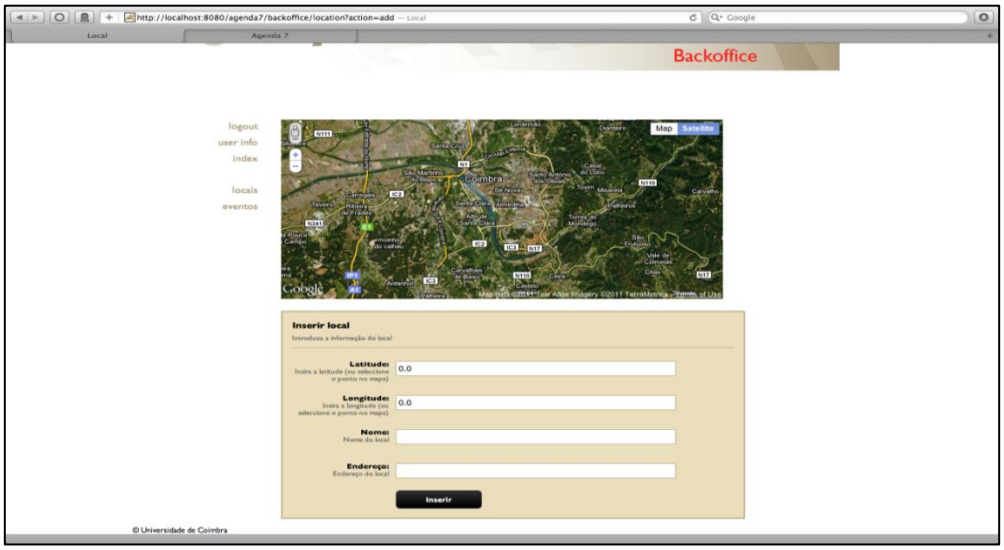


Figura 10

Na página constante na (fig.10) o utilizador deverá inserir o nome do local onde irá decorrer o evento e o respectivo endereço postal.

As coordenadas **Latitude** e **Longitude** são preenchidas automaticamente a partir do momento em que se clique no mapa para sinalizar o local.

Para finalizar o procedimento deverá clicar no botão 

.

Quando for efectuada uma correcção este botão é substituído pelo botão 

.

O procedimento concluído fará o utilizador retornar à página da listagem dos locais constante da (fig.9).

Para continuar o preenchimento da informação o utilizador deve seleccionar a opção **eventos,** na barra lateral que o levará para a página representada na figura abaixo (fig. 11).

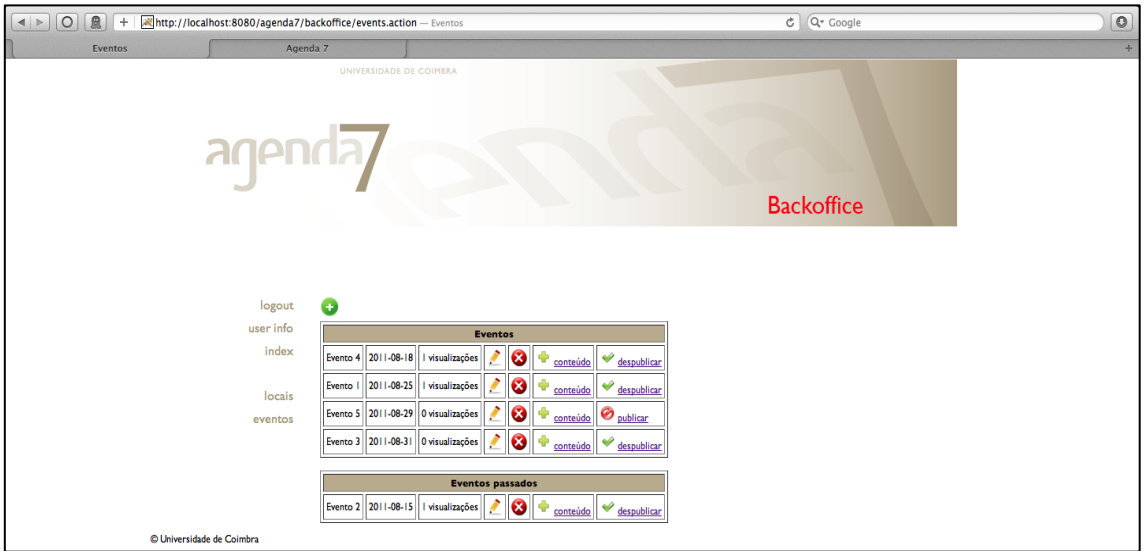


Figura 11

O utilizador inicia a introdução da informação relacionada com o campo eventos clicando no ícone  que irá activar o ecrã seguinte representado na (fig. 12) onde serão inseridas os dados associados a este bloco de informação.

Antes de passarmos a apresentar as instruções de preenchimento dos campos de informação constantes da (fig.12) explicam-se as tabelas e opções de interface que o utilizador pode visualizar na (fig. 11)

A) Tabelas

Eventos o utilizador pode visualizar o nome do evento, a data do seu início e o n.º de visualizações que foram efectuadas à página do evento.

Eventos Passados o utilizador pode visualizar o mesmo tipo de informação.

B) Opções de interface

Se um evento constante nas tabelas apresentadas contiver informação incorrecta o utilizador deverá clicar no botão correspondente a esse evento e proceder à sua correcção na página constante da (fig. 12).
Se o utilizador pretender remover a informação sobre o evento publicado deverá clicar no botão . Após este procedimento o evento deixará de constar da tabela de eventos.

Para adicionar ou corrigir a imagem de apresentação, incluir preços, adicionar aspectos específicos relacionados com o evento e incluir conteúdos multimédia o utilizador deverá clicar no botão . O procedimento associado a esta funcionalidade será explicado em simultâneo com a explicitação das páginas onde é efectuado o preenchimento daqueles conteúdos.

Na última coluna das tabelas o utilizador encontra um ícone indicador do estado de publicação do evento.
O ícone é o indicador de que o evento ainda não está publicado /disponível no portal da Agenda 7.
O ícone é o indicador de que o evento já se encontra publicado / disponível no portal da Agenda 7.

Para alterar o estado de publicação do evento o utilizador deve clicar no link à direita dos ícones.

Entrando na página de inserção ou edição de eventos constante da (fig.12) sugerem-se algumas directrizes para o preenchimento dos vários campos de informação que compõem a página supra citada:

- **Nome** – Recomenda-se que o nome do evento seja sintético de forma a evitar que este seja cortado na página de destaques da Agenda 7;

- **Descrição** – Parágrafo breve em que o utilizador descreve o essencial do evento;

- **Lotação** – Quando não existe um limite de lugares para assistir /participar no evento este campo não necessita de ser preenchido.
  O nº 0 pré-definido pela aplicação assume que não existe limitação de lugares.

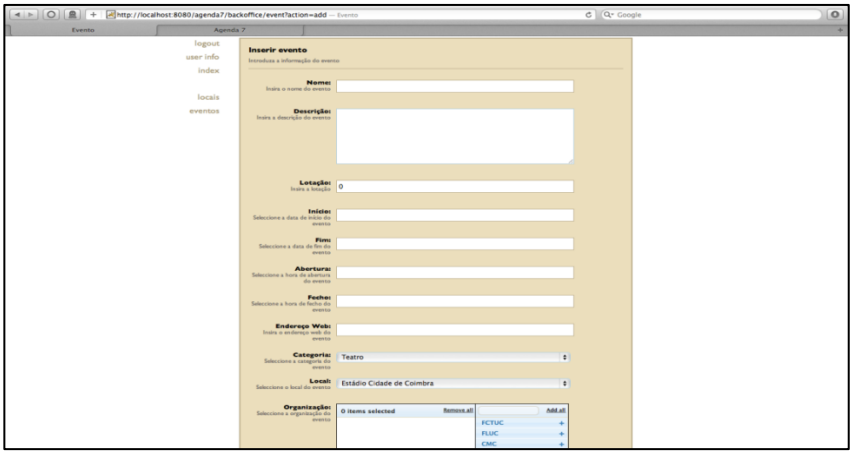


Figura 12

- **Início** e **Fim** – Estes campos têm associado um calendário dinâmico (fig. 13) que permite ao utilizador seleccionar o período em que decorre o evento.
  Clicando num ou noutro campo fica disponível o calendário onde deverão ser marcadas as datas de início e do fim do evento.

Figura 13

- **Abertura** e **Fecho** - Estes campos têm associada uma caixa (fig. 14) que permite ao utilizador seleccionar o horário em que decorre o evento.
  Clicando num ou noutro campo fica disponível o calendário onde deverão ser marcadas as datas de início e do fim do evento.

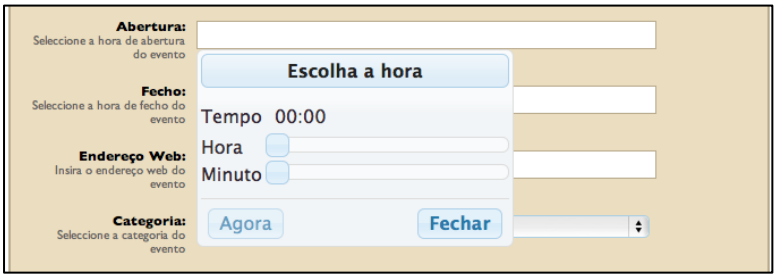


Figura 14

Os campos **Endereço Web**, **Categoria** e **Local** apresentados na **(**fig. 15) têm as seguintes regras de preenchimento:

- **Endereço Web –** Este campo não é de preenchimento obrigatório. Deverá ser preenchido se o utilizador pretender ligar a página da Agenda 7 a outra página exterior a este portal de informação;

- **Categoria** – O utilizador deverá seleccionar a categoria a que o evento corresponda de entre as categorias pré-definidas;

- **Local –** Este campo disponibiliza os locais criados anteriormente e visíveis na listagem apresentada na página de locais (fig.8).

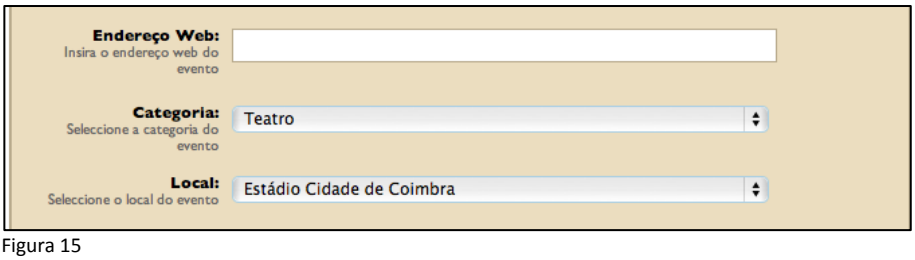


Figura 15

- O último campo de informação **Organização** (fig. 16) tem associada uma lista de Organizações/Entidades definidas pela administração da plataforma que são visíveis na coluna da direita do quadro abaixo apresentado.

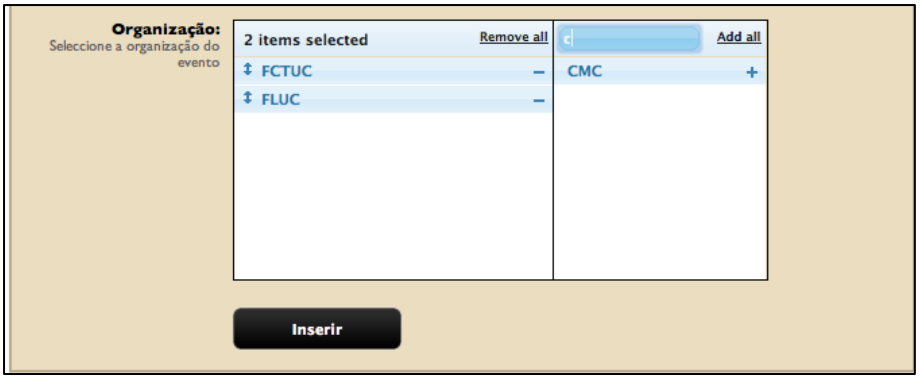


Figura 16

Para incluir a informação sobre a (as) organização (ões) responsáveis pelo evento o utilizador deverá clicar no símbolo + que se encontra a seguir ao nome da organização. De igual forma para remover a informação deverá clicar no símbolo - que se encontra a seguir ao nome da organização.

É obrigatória a inclusão de pelo menos uma Organização a que o utilizador se encontre associado. Poderá caso assim o entenda acrescentar outras organizações.

Depois de cumpridos estes passos deverá clicar no botão 

Cumprido o procedimento de criação do evento, se este tiver sido efectuado com sucesso aparecerá no ecrã do utilizador a página seguinte (fig. 17).

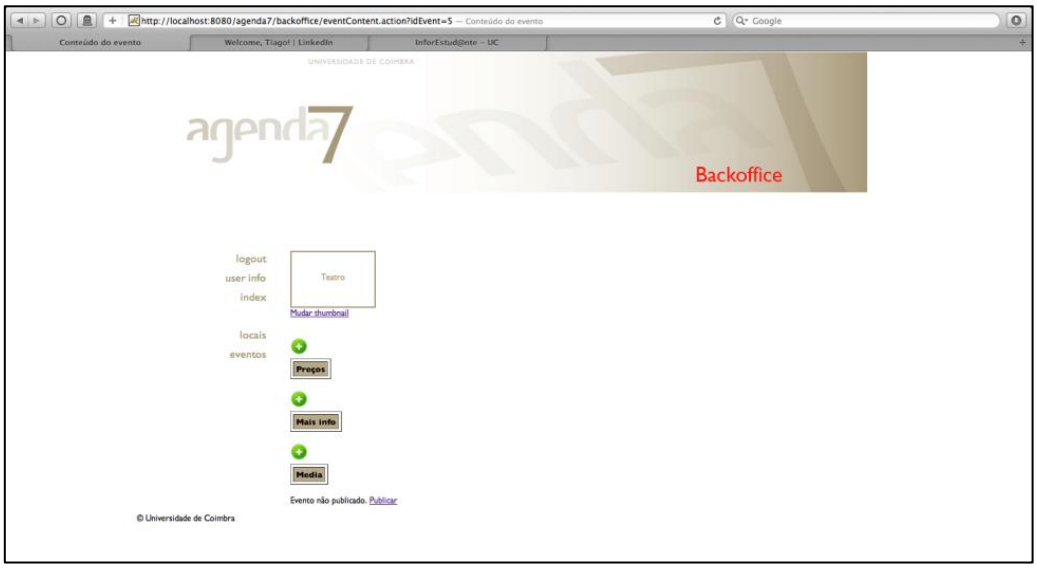


Figura 17

Esta página, de conteúdos do evento, permite aceder às funcionalidades que permitem a alteração do thumbnail, a introdução de preços, a adição de informação adicional - mais info e a inclusão de conteúdos media**.**

O utilizador deve iniciar o procedimento clicando em Mudar thumbnail. Este link activa uma nova página (fig. 18).

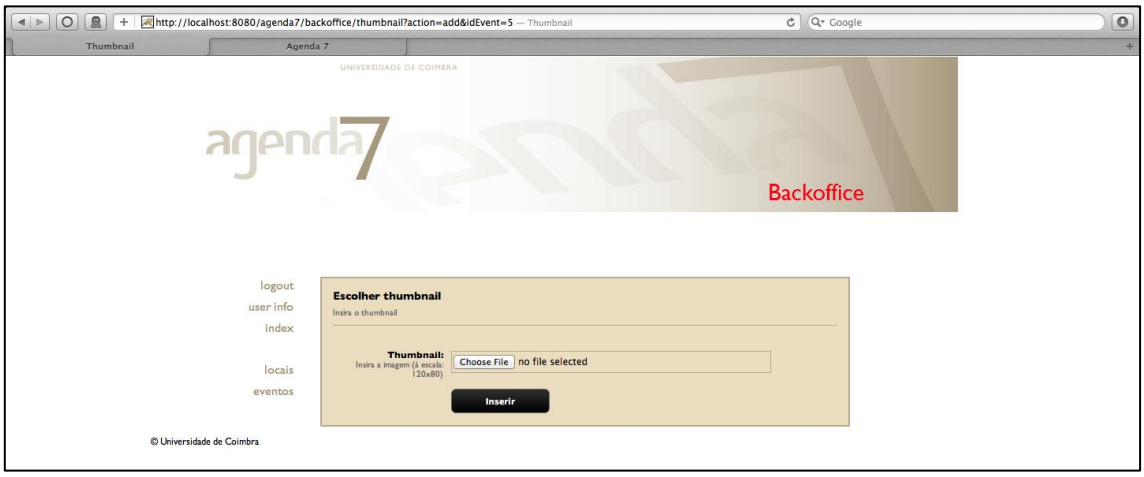


Figura 18

Para escolher a imagem de apresentação do evento clicar no botão que permite escolher o ficheiro que contém a imagem que pretende incluir. É aconselhável seleccionar uma imagem com um tamanho de 120x80 pixéis ou proporcional a este parâmetro de forma a evitar que a imagem surja distorcida. O formato da imagem tem que ser PNG, JPEG ou GIF.

Concluída esta operação o utilizador volta à página de conteúdos do evento onde estará já inserido o thumbnail (fig.19). Estará também disponível um novo link Remover thumbnail que permite, tal como o nome indica, remover a imagem colocada e repor a imagem de apresentação pré-definida.

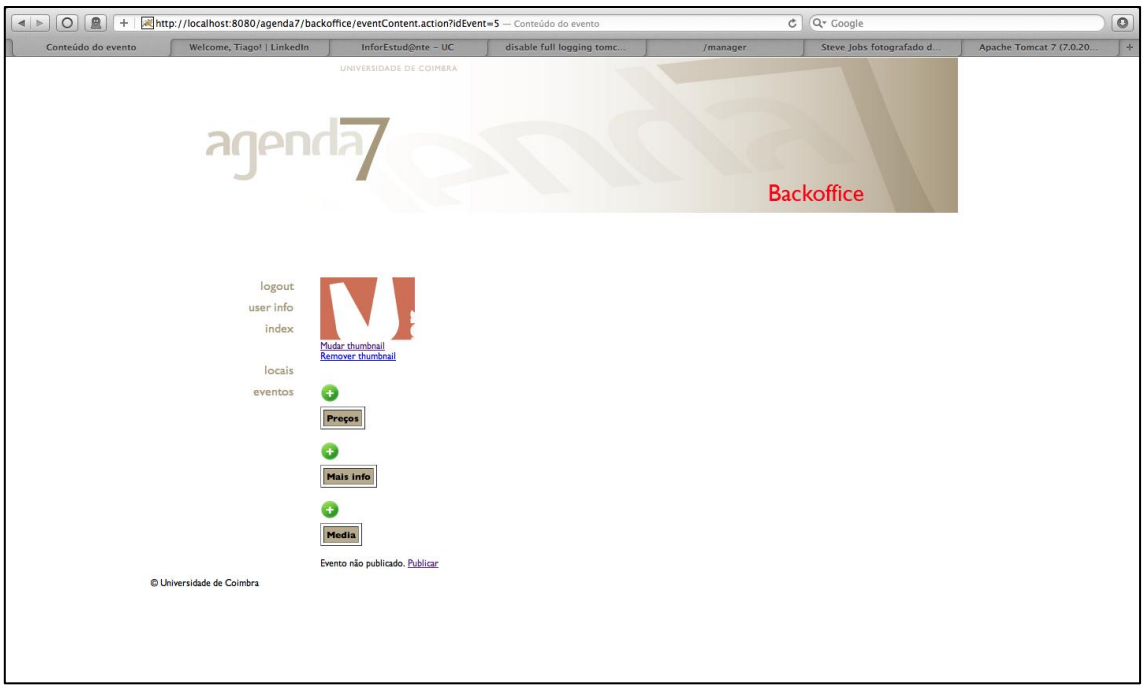


Figura 19

A partir desta página deve carregar no ícone colocado acima das três opções de conteúdos disponibilizadas - Preços, Mais info e Media - para inserir a informação relacionada com estes blocos de informação.

Clicando no ícone acima da opção **Preços** irá aceder à pagina representada na (fig.20).

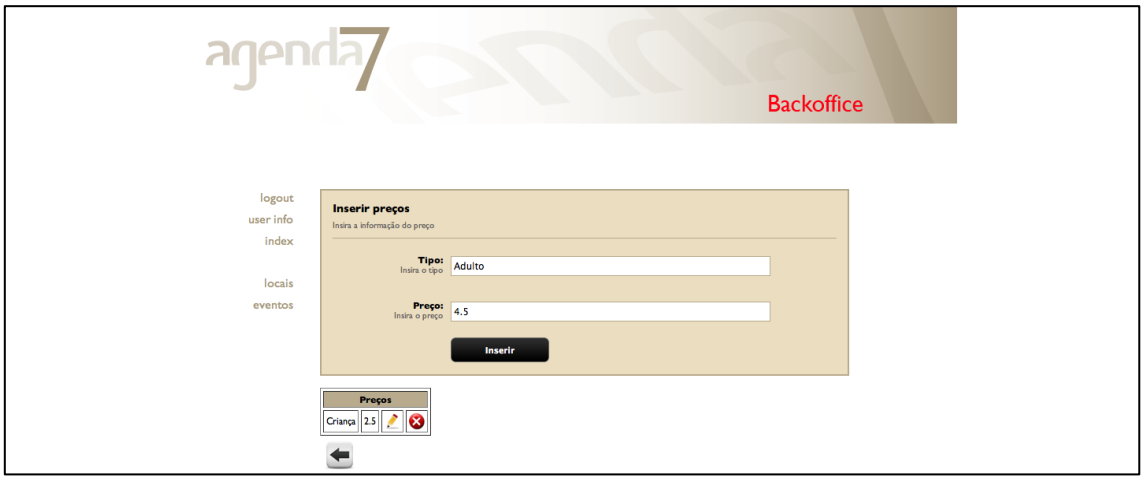


Figura 20

Os campos desta página são campos de preenchimento obrigatório devem ser inseridos sequencialmente para cada tipo de público.

Após cada inserção nos campos **Tipo** e **Preço** é criada uma linha na tabela que apresenta os dados que foram inseridos para o evento que está a ser editado.

Depois de concluir o preenchimento clicar no botão para voltar à página de conteúdos.

Clicando no ícone acima da opção **Mais info** irá aceder à página representada na (fig.21).

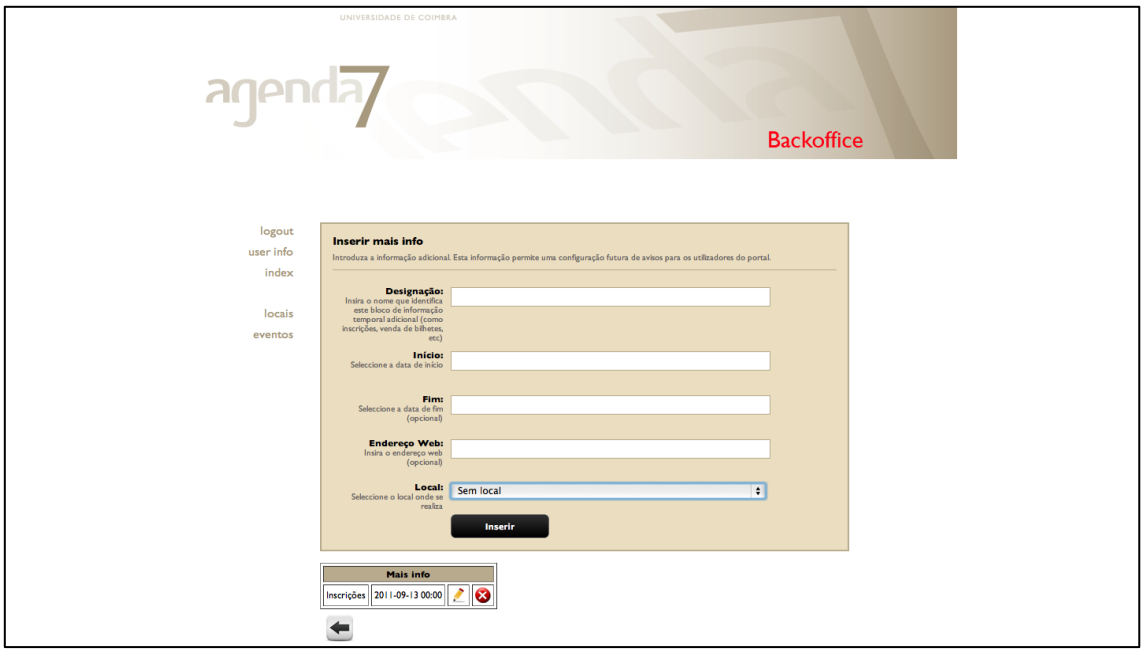


Figura 21

A generalidade dos campos desta página não é de preenchimento obrigatório, excepção feita para os campos **Designação** e **Início**.

- **Designação** - Neste campo o utilizador deve inserir a informação necessária para o público poder aceder a condicionantes relacionadas com o evento como por exemplo necessidade de inscrição, compra de bilhetes, entre outros. Pode ainda inserir o programa, um questionário, etc.. O nome deve obedecer aos itens que se pretendam destacar como por exemplo: Venda de Bilhetes, Inscrição, Programa, etc..

- **Início** e **Fim** - Estes campos têm associado um calendário dinâmico (fig. 22) que permite ao utilizador seleccionar a data e a hora que configuram o agendamento da acção.
  Clicando num ou noutro campo fica disponível o calendário onde deverão ser marcadas as datas de início e do fim das acções requeridas.

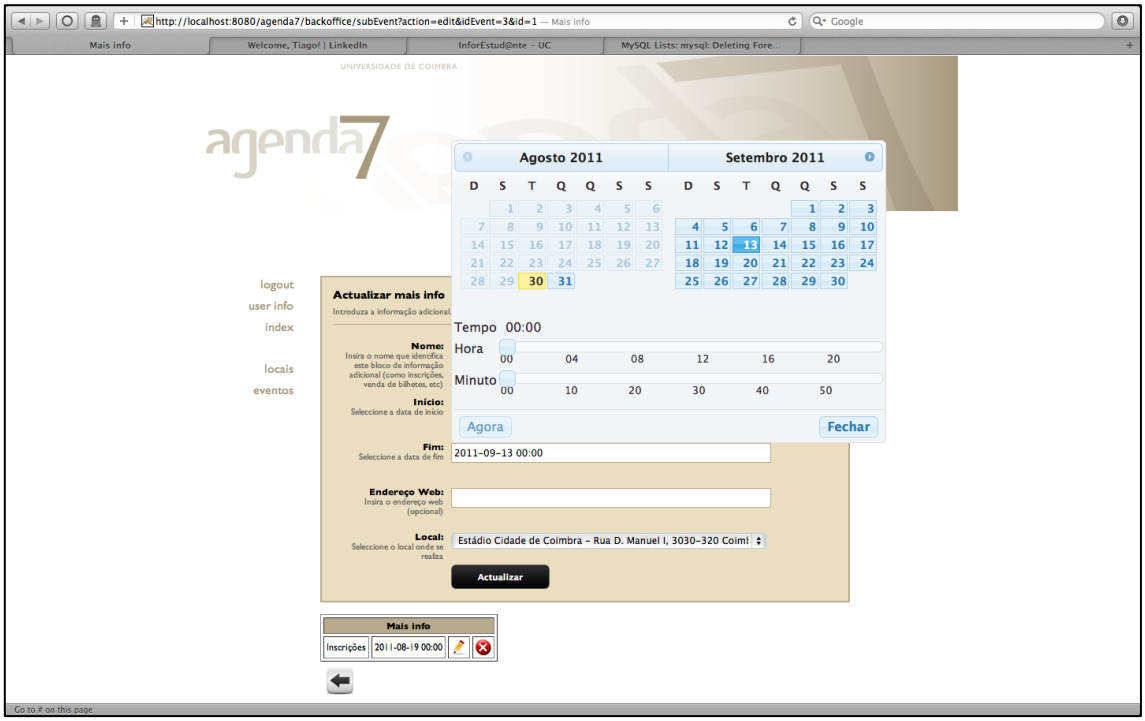


Figura 22

Após cada inserção é criada uma linha na tabela, representada no fundo da página, que apresenta a edição dos dados que foram preenchidos nos dois primeiros campos desta funcionalidade.

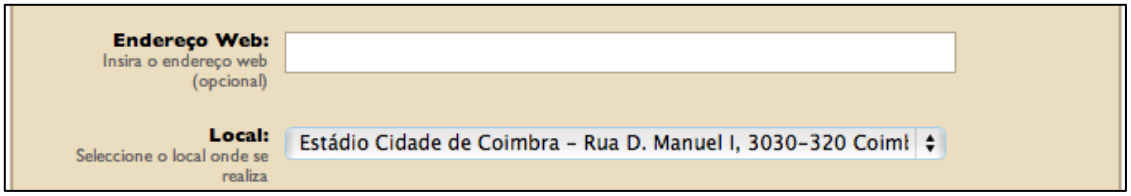


Figura 23

Endereço Web - Neste campo (fig.23) o utilizador deve inserir o endereço web que dá acesso a informação paralela e ou suplementar sobre a acção.

Local - Neste campo (fig.23). o utilizador deve seleccionar o endereço postal previamente criado na página de **Local.**

Depois de cumpridos estes passos deverá clicar no botão 

Para voltar à página de conteúdos clicar no botão  .

De novo, a partir da página de conteúdos do evento, deve carregar no ícone  colocado acima da opção **Media** para inserir os ficheiros multimédia na página representada na (fig. 24).

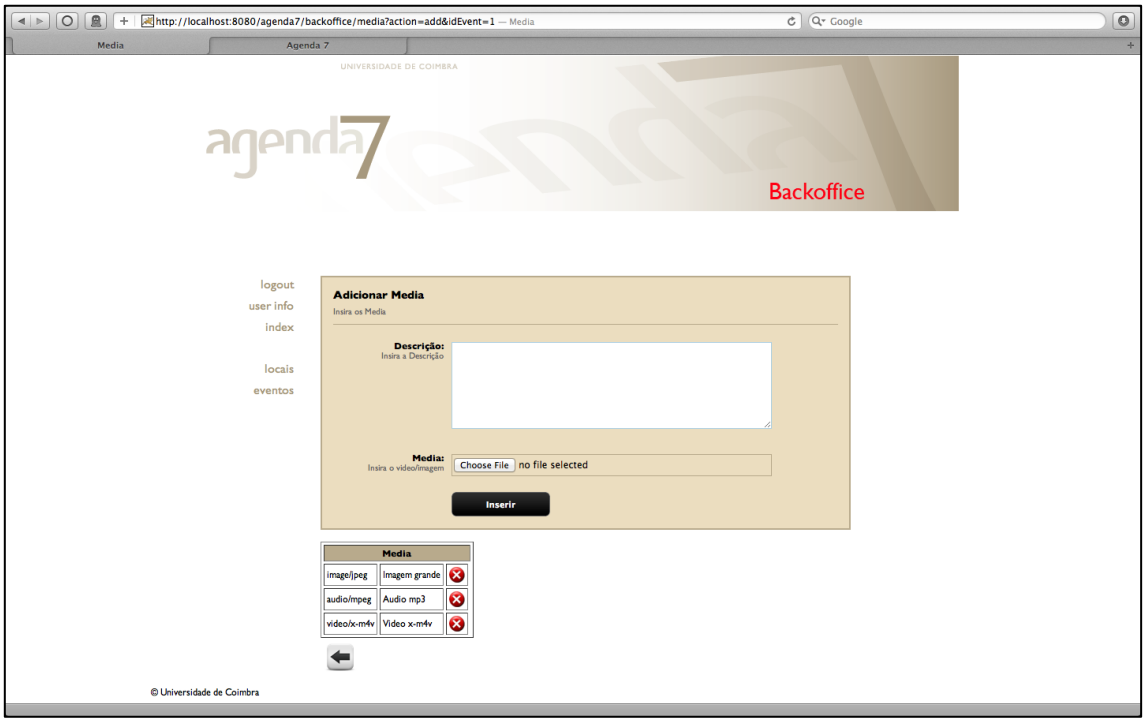


Figura 24

Este bloco de informação está organizado em dois campos: **Descrição** e **Media**.

- **Descrição** – Legenda do conteúdo multimédia;

- **Media** – Para inserir o conteúdo multimédia, Imagens, Vídeo e Áudio, clicar no botão que permite escolher o ficheiro que contém o conteúdo pretendido. É obrigatório obedecer às seguintes especificações:
  Imagens: O formato da imagem tem que ser PNG, JPEG ou GIF. É ainda aconselhável que as imagens não tenham dimensões muito grandes nem sejam demasiado desproporcionadas em relação ao monitor.
  Vídeo: O sistema só aceita o formato MPEG-4. Apesar de ser possível a inserção de qualquer ficheiro naquele formato, só devem ser inseridos vídeos codificados com o codec H.264, de forma a compatibilizar todos os sistemas. De

acordo com este pressuposto o utilizador deve providenciar junto dos produtores/ fornecedores de vídeos a entrega dos mesmos com estas características.
Os Vídeos devem também ser de curta duração e não podem ocupar um espaço superior a 200 MB.
Áudio: O sistema só aceita o formato MP3. Tal como na situação anterior o conteúdo não deve ter uma duração muito longa.

Após cada inserção é criada uma linha na tabela, representada no fundo da página, que apresenta a edição dos dados que foram inseridos nesta página.

A informação que é inserida nesta página irá corresponder ao conteúdo apresentado na galeria multimédia disponível na página de evento do portal da Agenda 7.

A título de exemplo a página representada na (fig. 25) configura o estado de edição final de um evento.

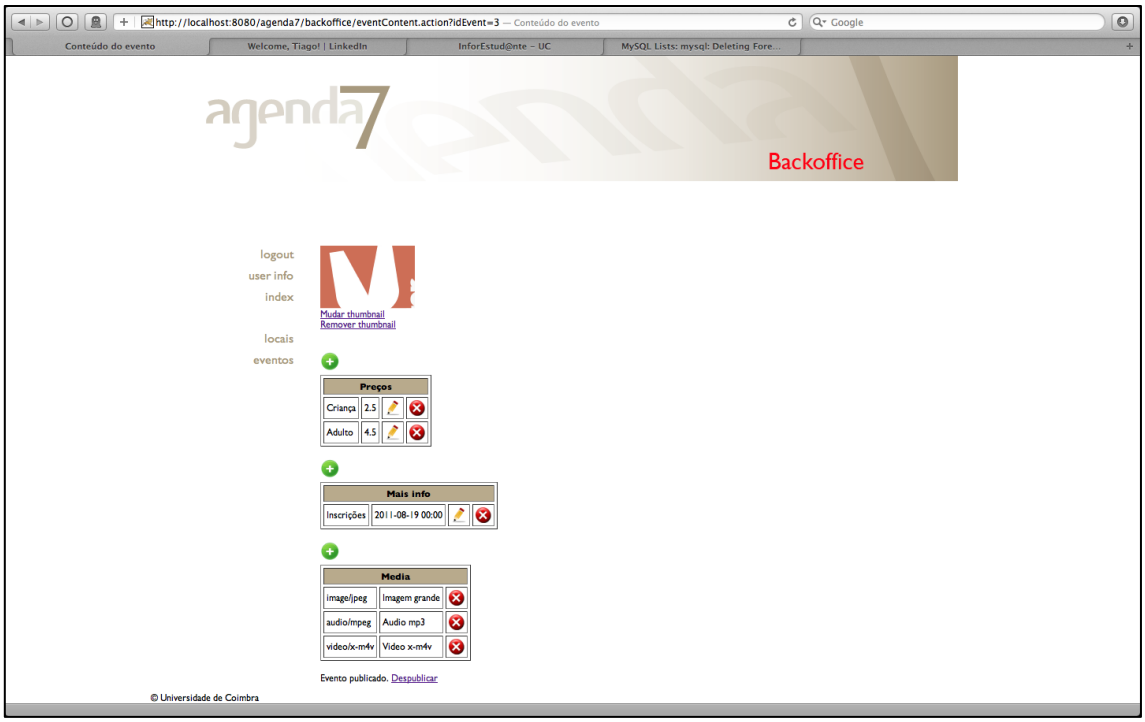


Figura 25

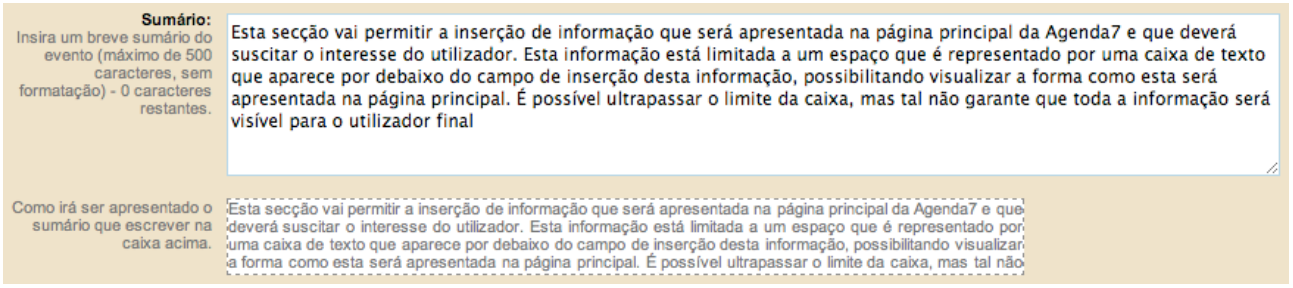
Para encerrar a publicação e para a informação ficar disponível no portal Agenda 7, o utilizador deve clicar no link Publicar que se encontra visível no fundo desta página. De igual forma se for necessário retirar o evento do portal deve ser accionado o link Despublicar.

14. Anexo 6 - Manual para alterações da Agenda7

Manual de utilizador para as alterações efectuada à segunda versão da Agenda7

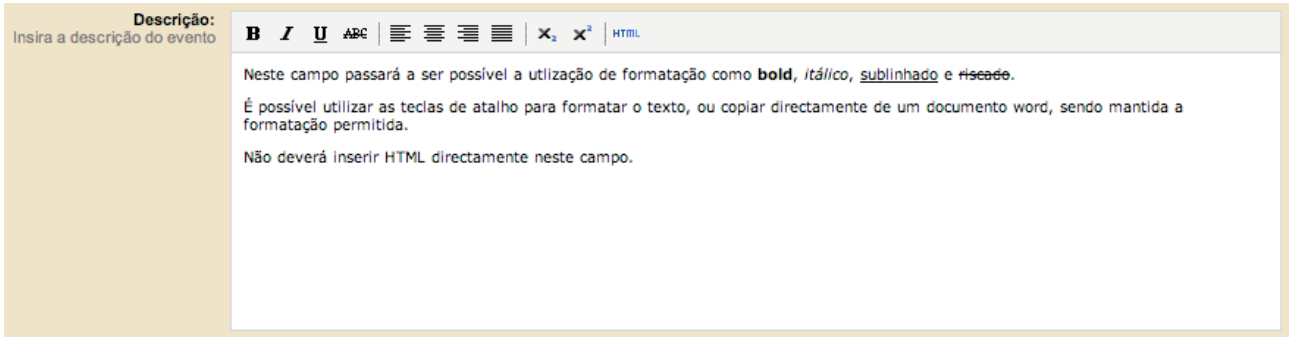
Sumário

A inserção de eventos apresenta agora um novo campo que permite distinguir o texto que aparece na apresentação do evento do texto que aparece quando se acede à página específica do evento.
Na página de inserção/edição de eventos, associado a este campo, existe um segundo campo que permite visualizar como esse texto será apresentado.



Descrição

A descrição do evento permite agora a inserção de texto formatado com bold, itálico, sublinhado e riscado. É mesmo possível copiar um texto diretamente de um ficheiro word com este tipo de formatação.
Por omissão todo o texto é justificado, e pede-se a quem pretenda alinhar o texto de outra forma que verifique se a página do evento fica como pretendida e que o layout não é afectado.



Pesquisa e seleção de locais

Passa agora a ser possível pesquisar locais a partir da página de inserção/edição de eventos. Para tal, basta utilizar a caixa acima da lista de locais, que irá filtrar todas as ocorrências de locais que contenham o termo de pesquisa.
Não esquecer que para o local ser efetivamente escolhido terá que selecionar na caixa que apresenta a listagem, mesmo que apenas apareça uma ocorrência. O local seleccionado aparecerá com fundo azul.

Pesquisa de organizações

Apenas para dar conhecimento da existência da funcionalidade, existe uma caixa de pesquisa que permite filtrar as organizações, facilitando a selecção das mesmas.

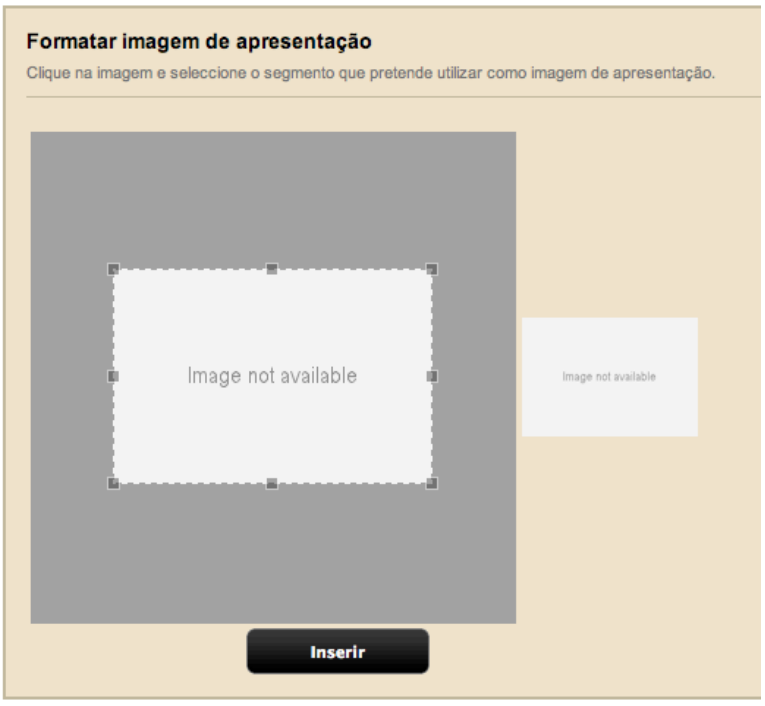
Pesquisa de Locais e Eventos

Agora, no topo das tabelas de locais e eventos, encontra-se uma caixa de texto que realiza uma pesquisa imediata, ficando apenas visíveis os itens que contêm o termo de pesquisa.

Escolher imagem de apresentação

A escolha da imagem de apresentação possui agora um passo extra que permite escolher uma porção da imagem com a escala adequada ao seu uso na Agenda7.

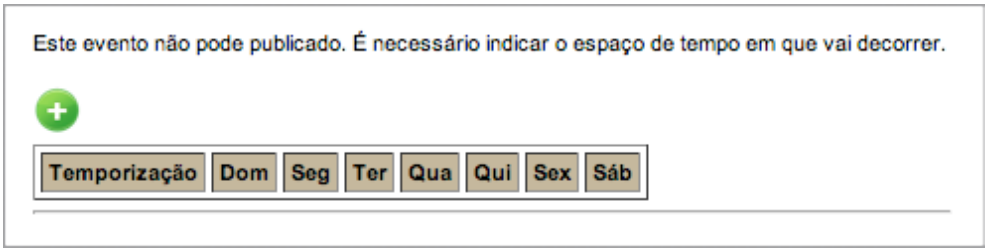
No lado esquerdo é possível selecionar e redimensionar a selecção da imagem carregada para o sistema, sendo apresentado do lado direito o resultado final dessa seleção.



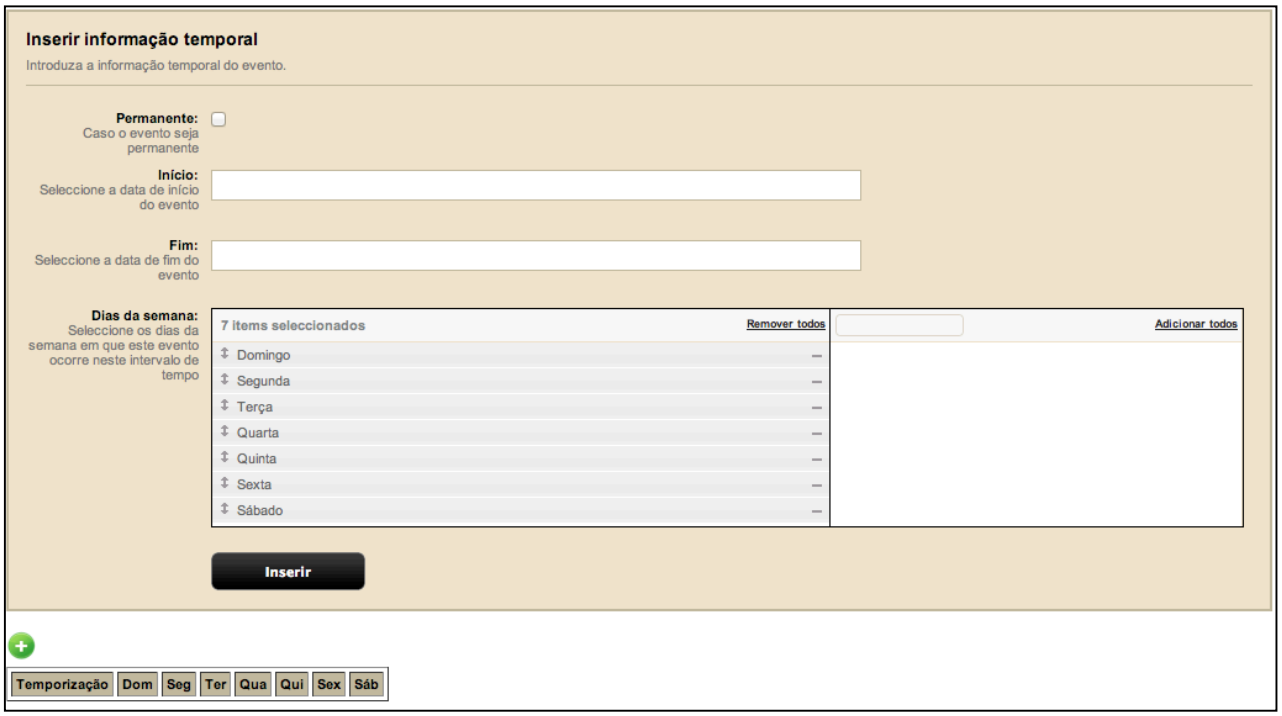
Temporização de eventos

De forma a flexibilizar as datas e horas de ocorrência dos eventos foi implementada uma nova forma de inserir datas e horas associadas aos eventos. Estas passam a ser fornecidas após a criação do evento.

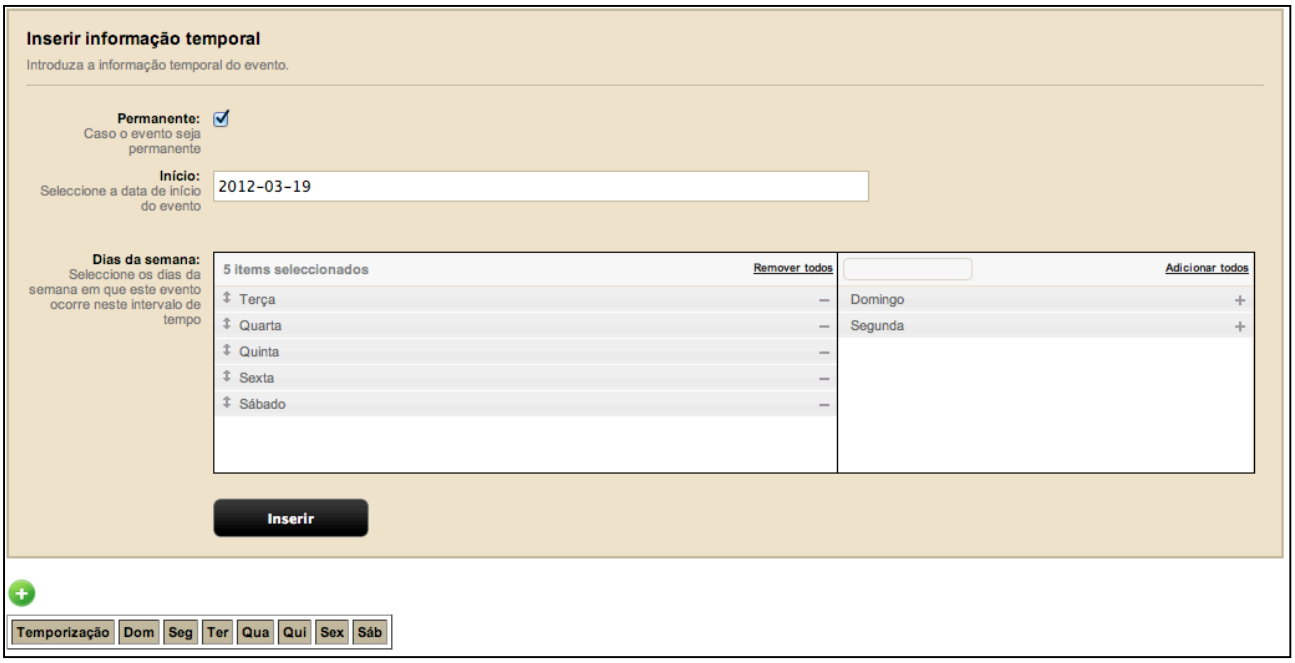
Na página de conteúdo do evento, que é apresentada após a criação do mesmo, irá encontrar a informação que indica que o evento não pode ser publicado por não existir ainda informação temporal sobre o mesmo. Por baixo dessa informação irá encontrar uma tabela que irá conter a informação dos tempos e horas em que o evento decorre:

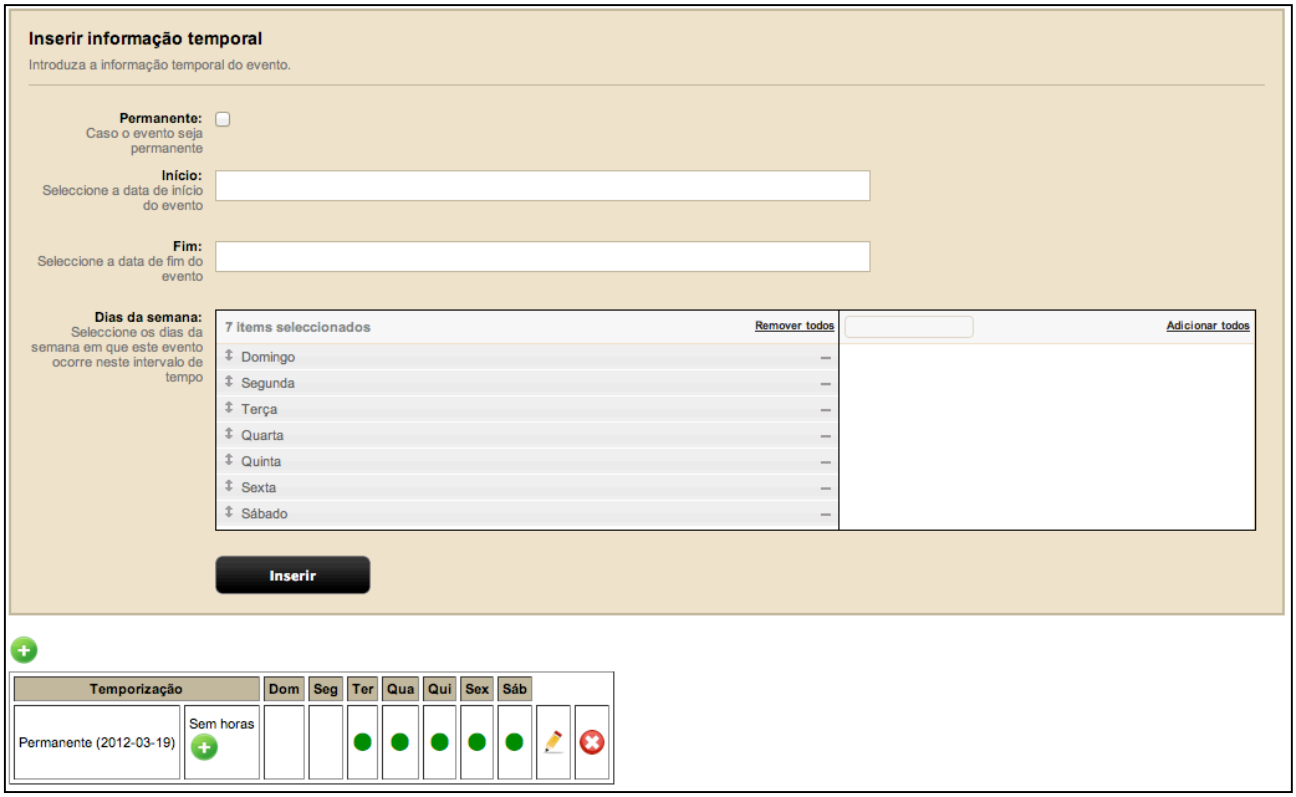


Clicando no botão +, irá ser redireccionado para uma página que possibilita a seleção de datas de início e fim, dias da semana em que o evento ocorre para essas datas e uma *checkbox* que permite indicar que o evento ocorre de forma permanente, ou eventos de longa duração que ainda não têm uma data de fim definida.



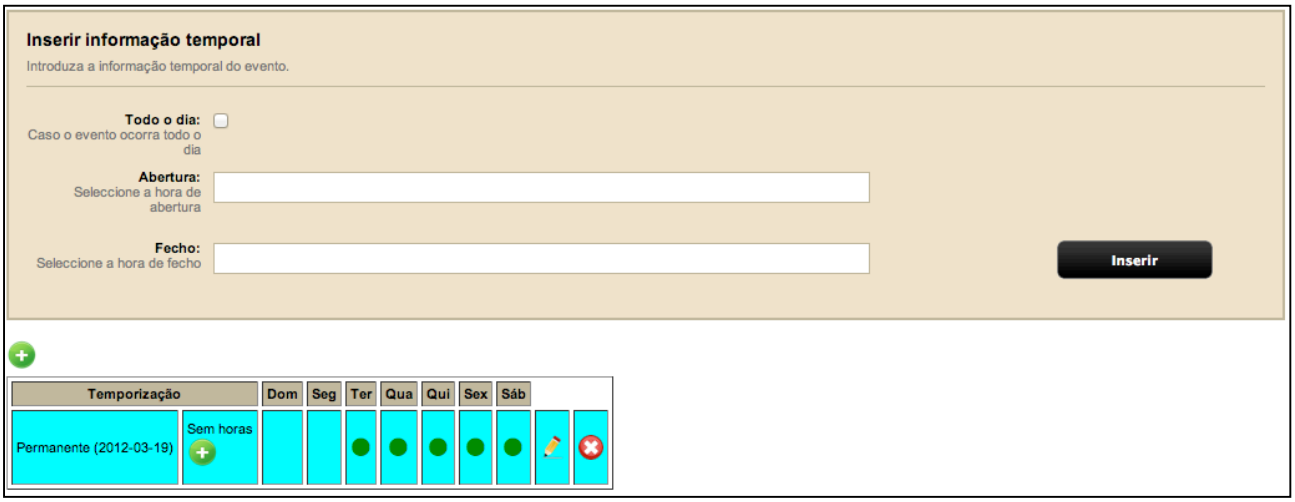
Ao selecionar a *checkbox* a data de fim irá desaparecer, sendo apenas necessário escolher a data de início. A título de exemplo, as próximas duas imagens irão apresentar o resultado da inserção de uma temporização permanente que ocorre todos os dias excepto às segundas e domingos:



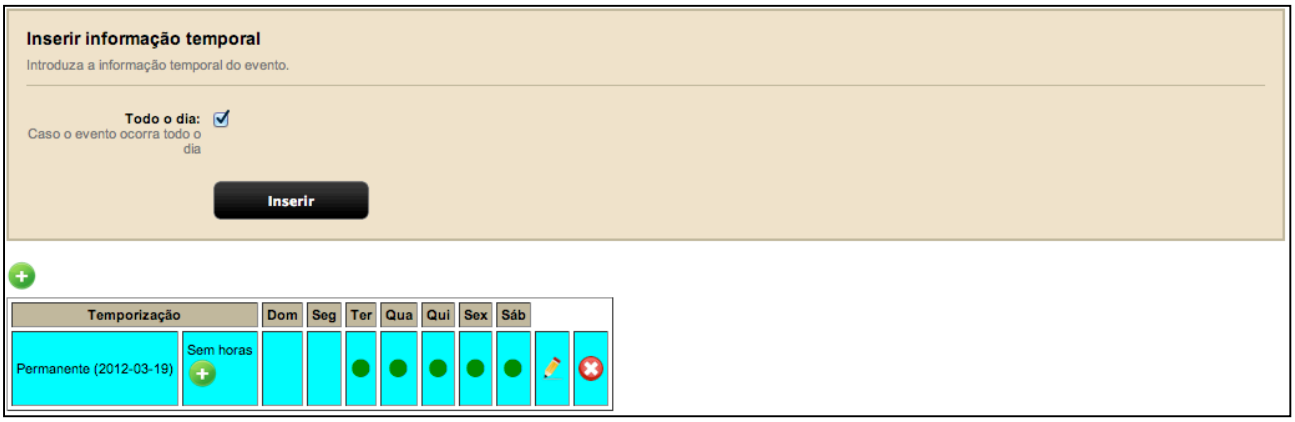


Como se pode observar, irá aparecer uma linha na tabela de temporização que indica que o evento decorre permanentemente a partir de dia 19 de Março de 2012, mas cujo horário ainda não está definido. A partir deste momento passa a ser possível publicar o evento na listagem de eventos ou na página de conteúdo do mesmo, clicando na seta de retorno apresentada abaixo. No entanto, o evento será apresentado com a informação "Sem horário definido".

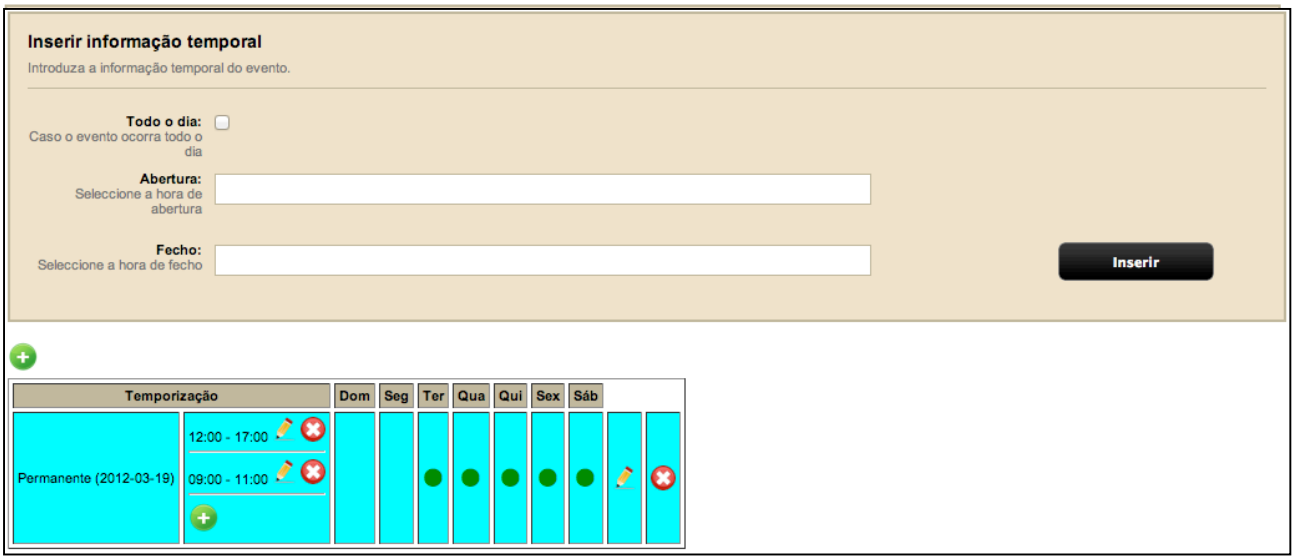
Ao clicar no botão +, abaixo do texto "Sem horas", pode adicionar uma ou mais informações horárias para os dias respectivos:



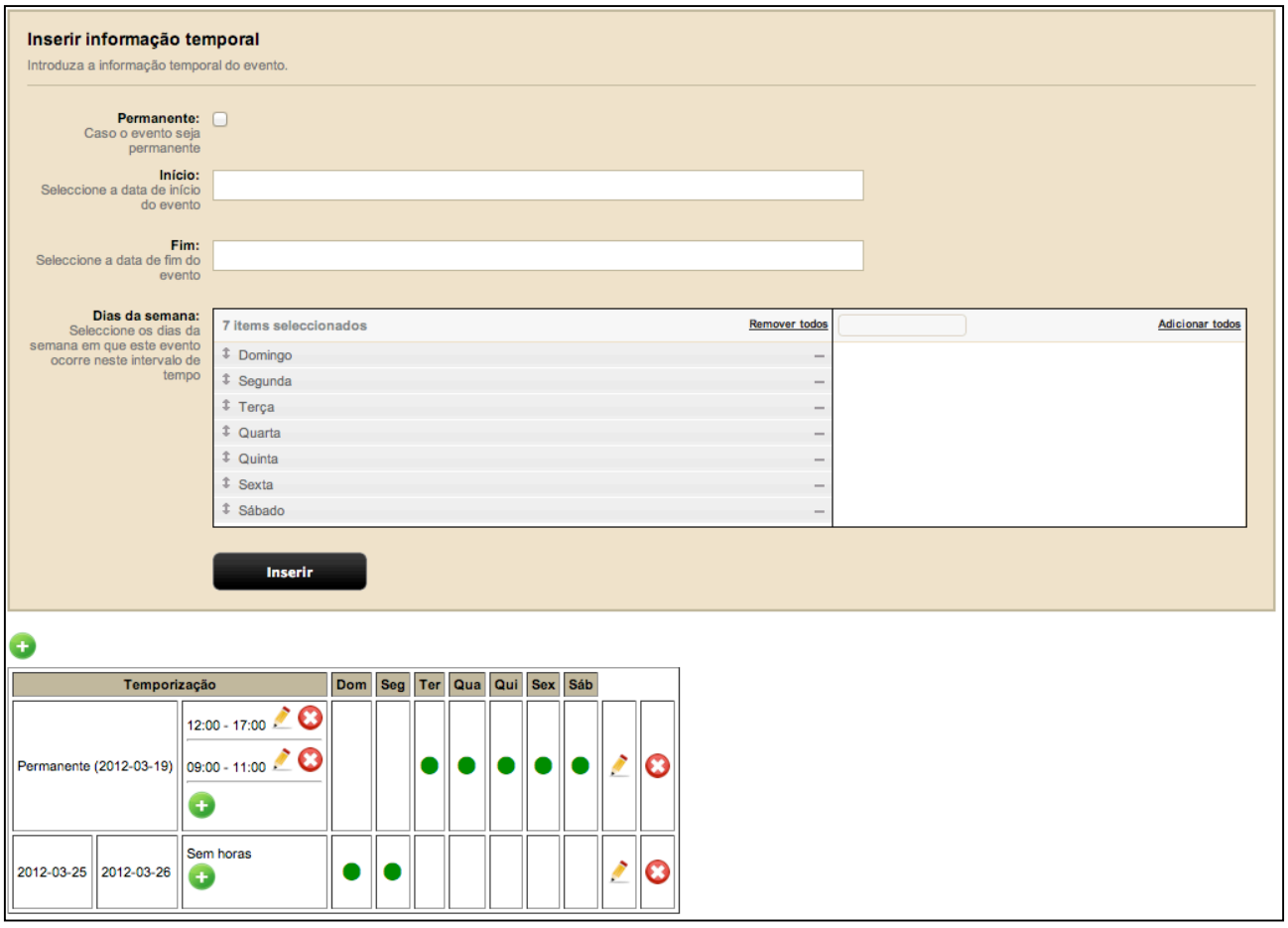
É possível clicar na *checkbox* "Todo o dia" para indicar que o evento decorre durante todo o dia, desaparecendo os campos "Abertura" e "Fecho".



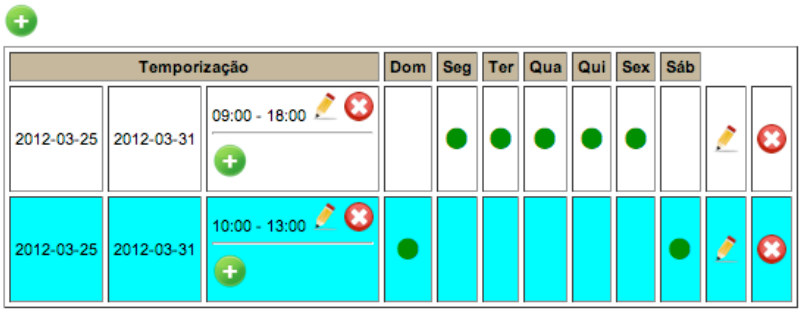
É possível inserir vários horários para o mesmo intervalo de datas. Estes horários não ficam apenas associados ao intervalo de datas escolhido, mas também aos dias da semana. Na sequência apresentada de seguida foram inseridos dois horários (das 12h às 17h e das 9h às 11h), de terça a sábado, permanentemente a partir de 19 de Março de 2012.



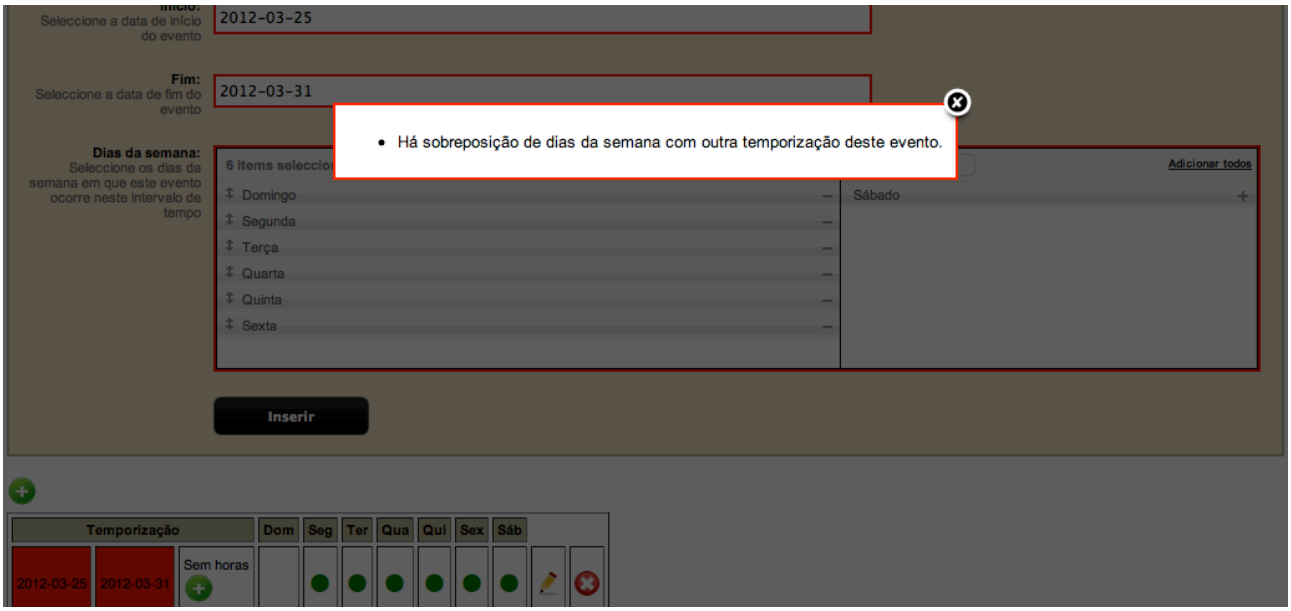
É possível inserir vários espaços de tempo para o mesmo evento, desde que não exista sobreposição de dias. Na imagem seguinte é possível observar que foi efetuada a inserção de um segundo intervalo temporal que não se sobrepõe por ocorrer em dias da semana distintos:



Desta forma é possível inserir eventos que decorrem em diferentes horários (por exemplo, com um horário de semana e outro de fim-de-semana, como apresentado no exemplo a seguir:



Caso tente sobrepor os mesmos dias para o mesmo evento, o sistema irá apresentar a mensagem de erro apresentada de seguida:



Neste último caso, tentou-se adicionar um segundo intervalo de tempo nos mesmos dias já existentes, apenas trocando o sábado pelo domingo, existindo uma sobreposição às segundas, terças, quartas, quintas e sextas, não sendo possível inserir esse intervalo de tempo.

Este erro é acompanhado de alguma informação visual que permite descriminar quais são os intervalos de tempo que estão a colidir com o que se está a tentar adicionar. Sendo estes apresentados com um fundo vermelho na tabela de intervalos de tempo.