Transposição de aplicações Desktop para plataformas móveis
Um caso de estudo

Vitor Hugo Correia Fernandes

UMinho | 2010

Universidade do Minho
Escola de Engenharia

Vitor Hugo Correia Fernandes

Transposição de aplicações Desktop para plataformas móveis
Um caso de estudo

Outubro de 2010

Universidade do Minho
Escola de Engenharia

Vitor Hugo Correia Fernandes

Transposição de aplicações Desktop para plataformas móveis
Um caso de estudo

Dissertação de Mestrado
Mestrado em Informática

Trabalho efectuado sob a orientação do
Doutor José Carlos Leite Ramalho

Outubro de 2010

Agradecimentos

Esta dissertação, apesar de ter sido um trabalho académico e individual, teve contributos de diversa natureza que não podem ser esquecidos e merecem ser realçados. Muitas pessoas contribuíram e apoiaram-me ao longo desta dissertação e fica aqui o reconhecimento a essas pessoas.

Agradeço ao meu orientador, Professor Doutor José Carlos Leite Ramalho, e ao meu orientador na empresa KEEP SOLUTIONS, Eng. Luís Faria, por todo o apoio, conhecimento, espírito critico, orientação e motivação fornecida, pois sem eles a realização e conclusão desta dissertação não seria possível.

Agradeço à empresa KEEP SOLUTIONS, por me ter fornecido um espaço e condições necessárias ao bom desenvolvimento desta dissertação. Fica também uma palavra de apreço a todas as pessoas que lá trabalham pela forma como me trataram, ajudando-me sempre que necessário. É um grupo de pessoas que me integrou da melhor forma, sendo extremamente sociáveis, proporcionando um enorme bem-estar e companheirismo.

Agradeço também aos meus colegas mais próximos de licenciatura ou de mestrado, pelo laços de amizade e de trabalho académico ao longo destes anos, esperando que estas relações não se percam com o tempo.

Por fim, quero agradecer aos meus pais, ao meus irmãos, aos meus amigos mais próximos e a outras pessoas especiais na minha vida por todo o apoio e compreensão neste momento da minha vida e ao longo de toda a dissertação. Foram todos muito importantes e sem eles esta dissertação não seria possível, e por isso, o meu obrigado sincero, esperando um dia retribuir da mesma forma.

Resumo

Os dispositivos móveis encontram-se cada vez mais desenvolvidos, proporcionando aos seus utilizadores uma melhor qualidade e maior quantidade de aplicações móveis. Por norma, essas aplicações móveis passam por tentativas de adaptação de aplicações Desktop ou Web, podendo implementar as mesmas funcionalidades ou apenas parte delas. Um programador tem que ter sempre presente a noção que uma aplicação móvel é mais limitada em termos de processador, memória, tamanho do ecrã e autonomia, sendo necessário uma maior controlo em relação às aplicações previamente desenvolvidas para outros ambientes, e que essas limitações têm influência na construção das interfaces da aplicação móvel. No entanto, não existem guias para a de adaptação de interfaces que digam ao programador como transformar as interfaces da aplicação previamente desenvolvida em interfaces para uma aplicação móvel, tornando mais moroso o processo de desenvolvimento de uma aplicação móvel.

Além das interfaces, as aplicações Desktop ou Web permitem muitas vezes a interacção e visualização de vários tipos de ficheiros e os dispositivos móveis, apesar do seu desenvolvimento exponencial nos últimos anos, só suportam determinados formatos. Deste modo, os dispositivos móveis possuem limitações na interacção e visualização de determinados tipos de ficheiros, havendo assim um entrave à disseminação de conteúdo via plataformas móveis.

É neste contexto que esta dissertação é realizada, tentando derivar guias para o desenvolvimento de aplicações móveis a partir de aplicações já desenvolvidas e abordando o problema da disseminação de conteúdos em plataformas móveis.

Abstract

The mobile devices are rapidly developing, offering to its users more and better applications. Usually, these applications attempt to be adaptations of desktop or web analogues, implementing part or even all of its features. When creating this kind of application, a developer must bear in mind that a mobile platform is more limited in terms of processor, memory, screen size and battery, requiring more control that its desktop or web counterparts, an that those limitations affect the interface design of the mobile applications. However, there are no guidelines for adapting desktop or web application interfaces into the mobile platforms, making this process even more onerous and time consuming.

Beyond the interface, there is the possibility of multimedia content being provided by the application. This content can be provided in various formats, and desktop platforms generally support them, but the spectrum of supported formats by mobile platforms is much smaller despite their growing development in recent years. This raises a problem of content dissemination, that must be considered in pair with the interface adaptation.

It is in this context that this paper is performed, trying to derive guidelines that would help in the development of mobile applications adapted from the desktop and looking for a solution to the content dissemination problem in mobile platforms.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Enquadramento

As comunicações móveis fazem parte do dia-a-dia de quase toda a gente, factor que tornou os dispositivos móveis cada vez mais complexos e poderosos. Os dispositivos móveis permitem, hoje em dia, ao utilizador efectuar quase todas as tarefas que normalmente executaria num computador, quer seja de forma completa ou com algumas limitações. O facto dos mesmos permitirem ao utilizador aceder a diversos conteúdos ou executar tarefas nos mais variados lugares num único dispositivo de pequenas dimensões é uma enorme vantagem. É inegável que o mercado móvel se encontra bastante activo e em pleno desenvolvimento, tendo como prova principal o crescente uso exponencial da Internet móvel. A evolução do mercado móvel atraiu diversas empresas que começaram a desenvolver diferentes sistemas operativos móveis e diferentes dispositivos móveis.

Hoje em dia, existe um conjunto diversificado de sistemas operativos móveis [11], cada um com as suas funcionalidades e características próprias mas é de salientar a relevância de cinco desses sistemas operativos: o Symbian, o iPhone OS, o Android, o RIM BlackBerry OS e o Windows Mobile [7]. A relevância desses sistemas operativos acenta sobretudo nas quotas do mercado móvel (2009), no renome da empresa que o produz e do próprio sistema operativo, na usabilidade, no fornecimento de um conjunto vasto de ferramentas para a produção de novas aplicações e nas previsões de crescimento [7]. A análise do mercado móvel em 2010 (Figura 1.1 [9]) confirma que esses cinco sistemas operativos se mantêm como dominadores, mantendo assim a sua importância. De referir, que o Android cumpriu as previsões de crescimento apontadas e que o Windows Mobile tem vindo a perder espaço no mercado móvel,

tendo sido recentemente substituído por outro sistema operativo, designado por Windows Phone 7. A comparação das características de alguns dispositivos com diferentes sistemas operativos levaria à conclusão que os mesmos diferem entre si. Essas diferenças podem ser no tamanho ou resolução do ecrã ou nos formatos de vídeo, áudio, imagem ou documentos de texto suportados pelos dispositivos [7]. Para além das características apresentadas no artigo, os dispositivos móveis podem também diferir em termo de capacidade de processamento, memória e armazenamento.

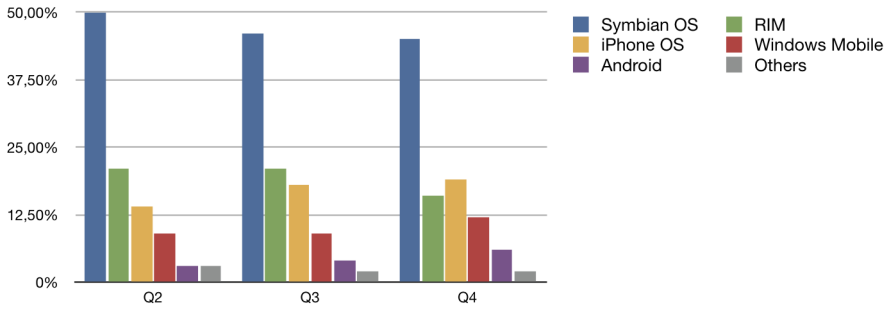

Figura 1.1: Mercado móvel no 2º trimeste de 2010 [9]

A existência de diferentes sistemas operativos e dispositivos móveis, do ponto de vista do utilizador, foi excelente pois aumentou a oferta disponível, permitindo ao utilizador optar de acordo com as suas preferências e exigências. Do ponto de vista do programador, apenas contribuiu para dificultar o desenvolvimento das aplicações móveis. A dificuldade acrescida com a diversidade de sistemas operativos móveis e dispositivos móveis prende-se pela necessidade da aplicação ter em conta as limitações tanto dos sistemas operativos como dos dispositivos. Apesar das dificuldades acrescidas aos programadores, a quantidade de aplicações móveis disponíveis para qualquer sistema operativo tem vindo a crescer de uma forma exponencial. Essas aplicações móveis podem ser adaptações de aplicações Web ou Desktop já existentes ou podem ser aplicações apenas focadas nas plataformas móveis.

Como muitas das aplicações móveis a serem desenvolvidas têm como base uma aplicação Desktop ou Web e estas, por norma, mostram muita informação, as interfaces das aplicações móveis terão que ser bem pensadas de forma a mostrar essa informação de uma forma coerente. É necessário então fazer uma adaptação das interfaces Web ou Desktop para interfaces móveis. O problema prende-se sobretudo com a inexistência de métodos que digam aos programadores a melhor forma de fazer a adaptação de interfaces. Entramos assim num dos grandes problemas do

desenvolvimento de aplicações móveis, ou seja, a construção/adaptação das interfaces para a aplicação móvel que tenham em conta as limitações dos dispositivos ou do sistema operativo para a qual vai ser desenvolvida. Para além disso as aplicações Desktop ou Web podem ser aplicações ricas em conteúdos multimédia, ou seja, permitem ao utilizador a interacção e a visualização de vários tipos de ficheiros, desde vídeos, imagens, sons, documentos entre outros. Isto pode provocar problemas nas aplicações móveis a serem desenvolvidas, resultantes da adaptação das aplicações Desktop ou Web, pois os dispositivos móveis são limitados em termos dos tipos de ficheiros suportados e variam entre si. Um utilizador comum ao correr certa aplicação móvel que permita interagir e visualizar determinados ficheiros, não se preocupa com o tipo de ficheiro apenas quer que o mesmo funcione. Portanto a adaptação de aplicações já desenvolvidas para plataformas móveis pode provocar um problema na disseminação de conteúdos, caso a aplicação a ser adaptada seja rica em conteúdos multimédia.

1.2 Objectivos

A inexistência de guias para a adaptação de interfaces de aplicações Desktop ou Web para interfaces de aplicações móveis é o maior problema no desenvolvimento de aplicações móveis. Esta dissertação tenta colmatar esse problema, sendo o principal objectivo analisar o desenvolvimento de uma aplicação móvel para uma aplicação já desenvolvida e a partir daí derivar guias que permitam fazer futuras adaptações de uma forma mais simples e rápida. Outro objectivo desta dissertação passa por abordar o problema da disseminação de conteúdos, que acontece quando se tenta adaptar aplicações ricas em conteúdos multimédia.

1.3 Estrutura da dissertação

De seguida é apresentada a estrutura desta dissertação, descrevendo sucintamente o conteúdo de cada capítulo.

No Capítulo 2 é feita a análise ao caso de estudo, que consiste no desenvolvimento de uma aplicação móvel através da adaptação de uma aplicação já desenvolvida. A aplicação escolhida para ser adaptada, é uma aplicação rica em interfaces dinâmicos e rica em conteúdos multimédia, de forma a permitir derivar os guias para adaptação de aplicações e abordar o problema da disseminação de conteúdos.

No Capítulo 3 são apresentados alguns testes realizados à aplicação mó-

vel que vai resultar de caso de estudo. A análise desses testes permite identificar alguns erros e problemas da aplicação móvel, que serão corrigidos/solucionados nas próximas versões da aplicação.

Por fim, no quarto capítulo, é elaborado um resumo da presente dissertação, são expostas os guias derivados da análise do caso de estudo, as contribuições que esta dissertação pode fornecer para o desenvolvimento de aplicações móveis e definem-se perspectivas de trabalho futuro.

São ainda incluidos diversos anexos, que vão ser referenciados ao longo da dissertação. Esses anexos constituem informação adicional à dissertação, contribuindo para melhorar a compressão da mesma. O anexo L corresponde ao artigo escrito numa unidade curricular, UCE15 Seminários, que integra o plano de estudos do 2º ano dos cursos de Mestrado de Informática e Mestrado de Engenharia Informática da Universidade do Minho. Esse artigo, apesar de não ter sido publicado em nenhuma conferência, vai ser usado como bibliografia desta dissertação (corresponde ao artigo [7] da bibliografia) e, como tal, pode ser citado em algumas afirmações. O artigo funciona como complemento da dissertação, motivo que levou à sua anexação.

Capítulo 2

Caso de Estudo

Neste capítulo vai ser feito o desenvolvimento de uma aplicação móvel para uma aplicação já desenvolvida. Na primeira secção vai ser identificada a aplicação base para a adaptação e a plataforma móvel escolhida para este caso de estudo, sendo feita uma pequena descrição para cada um dos pontos. Posteriormente vai ser apresentada a aplicação móvel propriamente dita, sendo descritos a arquitectura e o desenvolvimento de todo o processo. Por fim será abordado o problema da disseminação de conteúdos, que acontece quando a aplicação móvel resulta de uma adaptação de uma aplicação rica em conteúdos multimédia. A análise deste capítulo vai ser importante para a derivação de guias para a adaptação de aplicações previamente desenvolvidas para aplicações móveis.

2.1 Objecto de estudo

O objecto de estudo vai ser o desenvolvimento de uma aplicação móvel para o weebox, que é uma aplicação desenvolvida pela KEEP SOLUTIONS (para mais informações consulte a subsecção 2.1.1).

O weebox é um sistema de arquivo de informação digital e uma plataforma de gestão documental, garantindo o seu arquivamento seguro (para mais informações sobre a aplicaçãos consulte a subsecção 2.1.2). Um dos módulos do weebox possui uma interface Web, permitindo ao utilizador uma interacção com o sistema. O weebox produz interfaces dinâmicas de acordo o tipo de documento que se encontra a mostrar e permite a interacção por parte do utilizador com vários formatos de ficheiros. Como possui muitas interfaces Web e é uma aplicação rica em conteúdos multimédia, o weebox mostra-se como uma aplicação ideal para o caso de estudo, pois permite cumprir os objectivos deste trabalho: o desenvolvimento de guias para a adaptação de interfaces para aplica-

ções móveis e implementar uma solução para o problema da disseminação de conteúdos.

É necessário também proceder à escolha de uma plataforma móvel. Para este caso de estudo a plataforma móvel escolhida foi o iOS (para mais informações consulte a Subseccção 2.1.3). O iOS é uma plataforma amplamente usada e bastante aceite pelos utilizadores. A nível mundial é o líder de gasto de tráfego de Internet móvel, tendo obtido uma cota de 40% no mês de Fevereiro segundo o relatório mensal produzido pela AdMob sobre métricas móveis [1]. Para além disso é uma plataforma que possui um enorme número de aplicações disponíveis o que demonstra o enorme interesse nesta plataforma. O iOS só suporta um conjunto limitado de tipos de ficheiro tornando-se uma óptima plataforma para abordar o problema de disseminação de conteúdos. Portanto o iOS preenche os requisitos pretendidos, pois, para além de ser uma plataforma amplamente usada permitindo levar a aplicação a mais utilizadores, é uma plataforma ideal para aplicações que necessitam uma ligação com o servidor, que é o caso da aplicação a ser desenvolvida.

2.1.1 KEEP SOLUTIONS

A KEEP SOLUTIONS é uma spin-off académica da Universidade do Minho que oferece uma vasta gama de produtos e serviços de suporte à criação e manutenção de arquivos, museus e bibliotecas digitais [17]. Neste contexto, a KEEP disponibiliza um conjunto de produtos especialmente vocacionados para a área documental, tais como: consultoria informática tendo como focos principais a preservação digital, a análise, transformação e migração de dados, análise e concepção de sistemas de informação, manutenção e suporte de repositórios digitais, e concepção de soluções para publicação electrónica.

A estreita ligação da KEEP SOLUTIONS à Universidade do Minho permite-lhe manter um contacto directo com as mais recentes linhas de investigação. Sendo uma spin-off da Universidade do Minho, faz parte da missão da KEEP a adaptação da investigação desenvolvida no meio académico à linguagem e necessidades do mercado potenciando, deste modo, a investigação desenvolvida e contribuindo para o desenvolvimento da competitividade dos organismos nacionais.

Tendo nascido de uma plataforma de I&D, a KEEP SOLUTIONS permanece activa na produção de conhecimento científico. Prova disso são as inúmeras publicações e participações em eventos científicos onde os seus colaboradores têm marcado presença.

2.1.2 weebox

O software weebox é um sistema de arquivo de informação digital que tem como principais objectivos a usabilidade, a elevada performance ao nível da indexação e pesquisa, a manutenção reduzida e a grande capacidade de integração com outros sistemas, i.e. interoperabilidade. O weebox é ainda uma plataforma de gestão documental que garante o arquivamento seguro de documentos em vários formatos, salvaguardando requisitos fundamentais como a privacidade da informação, segurança dos dados, gestão de utilizadores, catalogação, recuperação de informação, controlo de versões e histórico de alterações.

O weebox é constituído por um conjunto de módulos aplicacionais destinado à gestão documental. Cada módulo apresenta um conjunto de funcionalidade distintas, mas completamente integrado nos restantes módulos. A organização em módulos permite adequar a solução às necessidades do cliente. O sistema é composto pelos seguintes componentes funcionais (Tabela 2.1): weebox Core Services, weebox Manager e weebox Community.

Tabela 2.1: Descrição dos vários módulos do software weebox

Componente	Descrição
weebox Core Services	Módulo responsável pela prestação de serviços básicos de suporte aos restantes módulos, e.g. depósito, indexação de metadados e texto integral, localização e recuperação de informação, gestão de permissões, *logging*, controlo de versões, interface SRU, interfaces REST, etc.
weebox Manager	Módulo de gestão documental que contempla funcionalidades como: gestão de utilizadores, gestão de perfis de metainformação, depósito de documentos, catalogação, vocabulários controlados, extracção automática de metainformação descritiva e técnica, gestão permissões por documento, pesquisa simples e avançada em texto integral.
weebox Community	Módulo que permite a integração do weebox em sítios Web existentes. Incorpora funcionalidades de pesquisa e recuperação de documentos de carácter público. Implementa ainda funcionalidades como *tag clouds* e RSS.

O weebox tem como particularidade o facto de não necessitar de um sistema de gestão de base de dados para suportar as suas actividades.

Esta premissa tem como principal vantagem baixar o preço do produto, uma vez que não exige custos de licenciamento e simplifica a gestão e manutenção do repositório de dados.

Para este caso de estudo apenas basta ter em conta dois desses módulos que vão ser descritos mais pormenorizadamente nas seguintes subsecções.

weebox Core Services

O weebox Core Services é um servidor aplicacional REST[2] para o armazenamento de dados e metadados. A unidade de armazenamento é o **documento**, que é composto por metadados genéricos (lista de pares: chave-valor), um conjunto de ficheiros, permissões e relações com outros documentos. Apresenta uma interface REST, cuja informação devolvida é predominantemente em formato XML [3].

As suas funcionalidades incluem:

- **Documentos**
  - Guardar um documento com um conjunto de ficheiros, metadados e permissões associadas;
  - Obter a metainformação de um documento (XML);
  - Alterar a metainformação de um documento;
  - Marcar um documento como removido;
  - Desmarcar um documento como removido;
  - Permanentemente remover um documento.
- **Ficheiros**
  - Obter a lista de ficheiros de um documento;
  - Descarregar todos os ficheiros de um documento comprimidos num zip;
  - Obter a metainformação de um ficheiro;
  - Descarregar um ficheiro do documento;
  - Adicionar ficheiros a um documento;
  - Remover ficheiros de um documento.
- **Pesquisa**

[2] *Representational state transfer*
[3] *eXtensible Markup Language*

  - Fazer um pesquisa na metainformação dos documentos (SRU);
  - Obter a lista de termos mais frequentes.

- **Utilizadores**
  - Criar/remover utilizadores e grupos;
  - Gerir utilizadores dum grupo;
  - Actualizar/listar informação de utilizadores e grupos;
  - Autenticar utilizadores.

O weebox Core Services permite a extensão da sua funcionalidade. Algumas das extensões já implementadas são as de extracção automática de metainformação de vários tipos de ficheiros, extracção do texto integral de vários tipos de ficheiro, adicionando-o como fonte de pesquisa e registo de todas as acções efectuadas (*log*).

weebox Manager

O weebox Manager é um módulo que oferece uma interface web para fácil gestão dos documentos. Este módulo traz um novo conceito: **tipo do documento**. Um tipo de documento define o conjunto de campos de metainformação disponíveis para edição, descrevendo para cada campo:

- A descrição (label) do campo nos várias idiomas suportados;
- O tipo de dados do campo de metainformação, o que vai influenciar a sua apresentação e os componentes visuais usados para edição;
- O valor por omissão (opcional);
- Se o campo é editável;
- Se o campo é obrigatório;
- Lista das fontes de informação de onde o campo pode ser gerado, bem como regras para a segmentação e formatação da fonte de informação (e.g. Título extraído dos ficheiros, tamanho dos ficheiros, data actual, nome completo do utilizador autenticado, etc.);
- Regra para agregação da informação retirada das variadas fontes (e.g. somatório para o tamanho dos ficheiros ou usar o primeiro da lista para o título);
- Informação da cor que é associada;

- Informação sobre a utilização por omissão nos novos documentos criados.

As suas funcionalidades, que podem ser consultadas no Anexo A, encontram-se agrupadas em 5 grupos (autenticação, pesquisa, resultados de pesquisa, documento e preferências. Essas funcionalidades permitem a iteração do utilizador com o weebox Core Services e os seus serviços.

Nas Figuras 2.1, 2.2, 2.3, 2.4 e 2.5 podemos observar algumas páginas do interface Web que são a base da conversão das interfaces para aplicação weebox para iPhone. Na Figura 2.1 podemos ver a página de autenticação e na Figura 2.2 podemos ver a página de registo na aplicação. Após autenticação correcta o utilizador entra para na página da Figura 2.3, que permite ao utilizador seleccionar filtros, ver listagem de documentos, fazer pesquisas básicas ou simples, entre outras operações. Caso o utilizador seleccione um documento aparece um painel (Figura 2.4) com as informações do documentos, listagem de ficheiros com as respectivas informações, paginação da listagem, entre outras operações. Caso o utilizador carregue no botão de pesquisa avançada aparece um painel (Figura 2.5) que permite ao utilizador introduzir os dados a pesquisar.

Figura 2.1: Página Web de autenticação

Figura 2.2: Página Web de registo

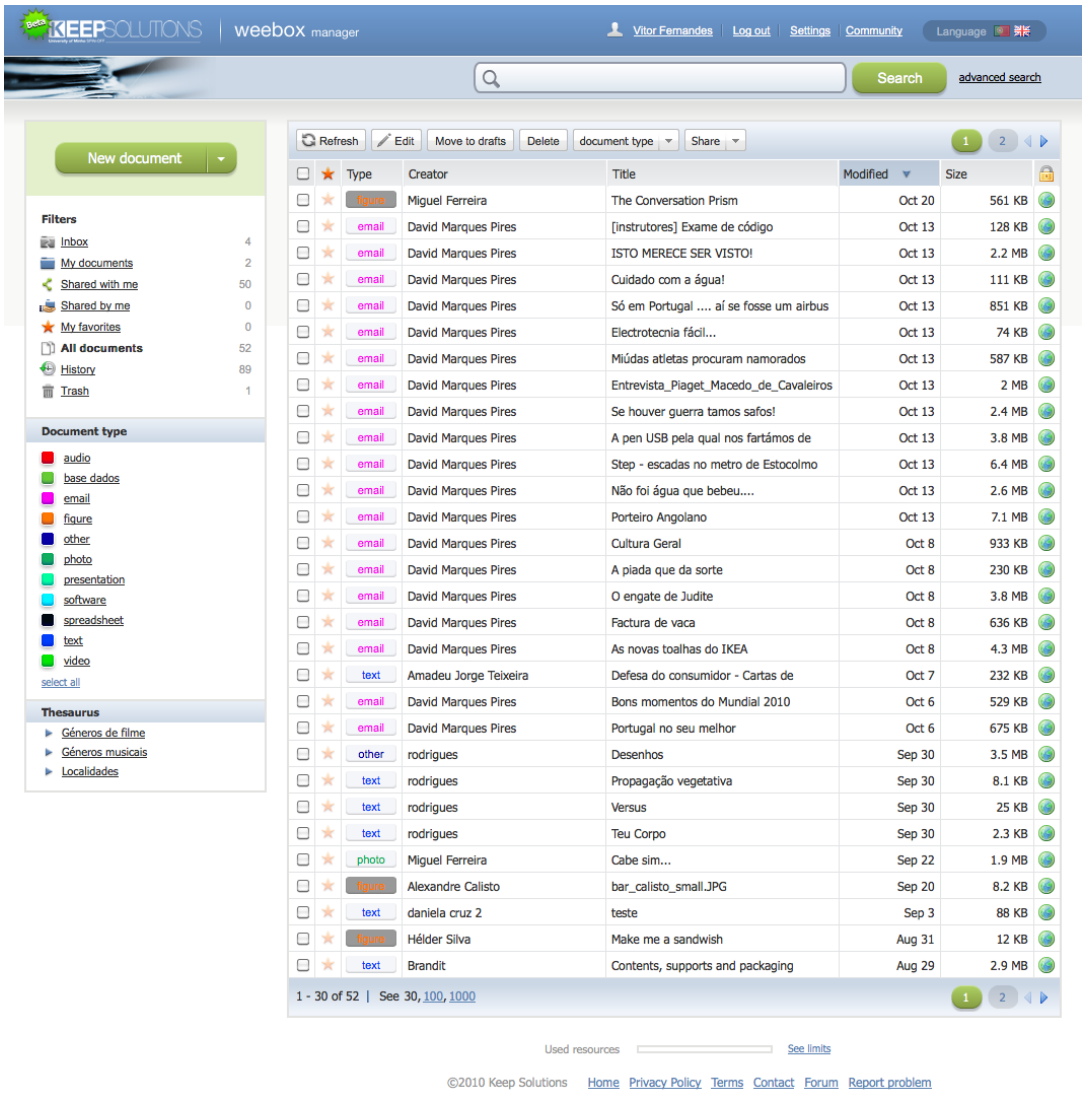

Figura 2.3: Página Web com a listagem de documentos

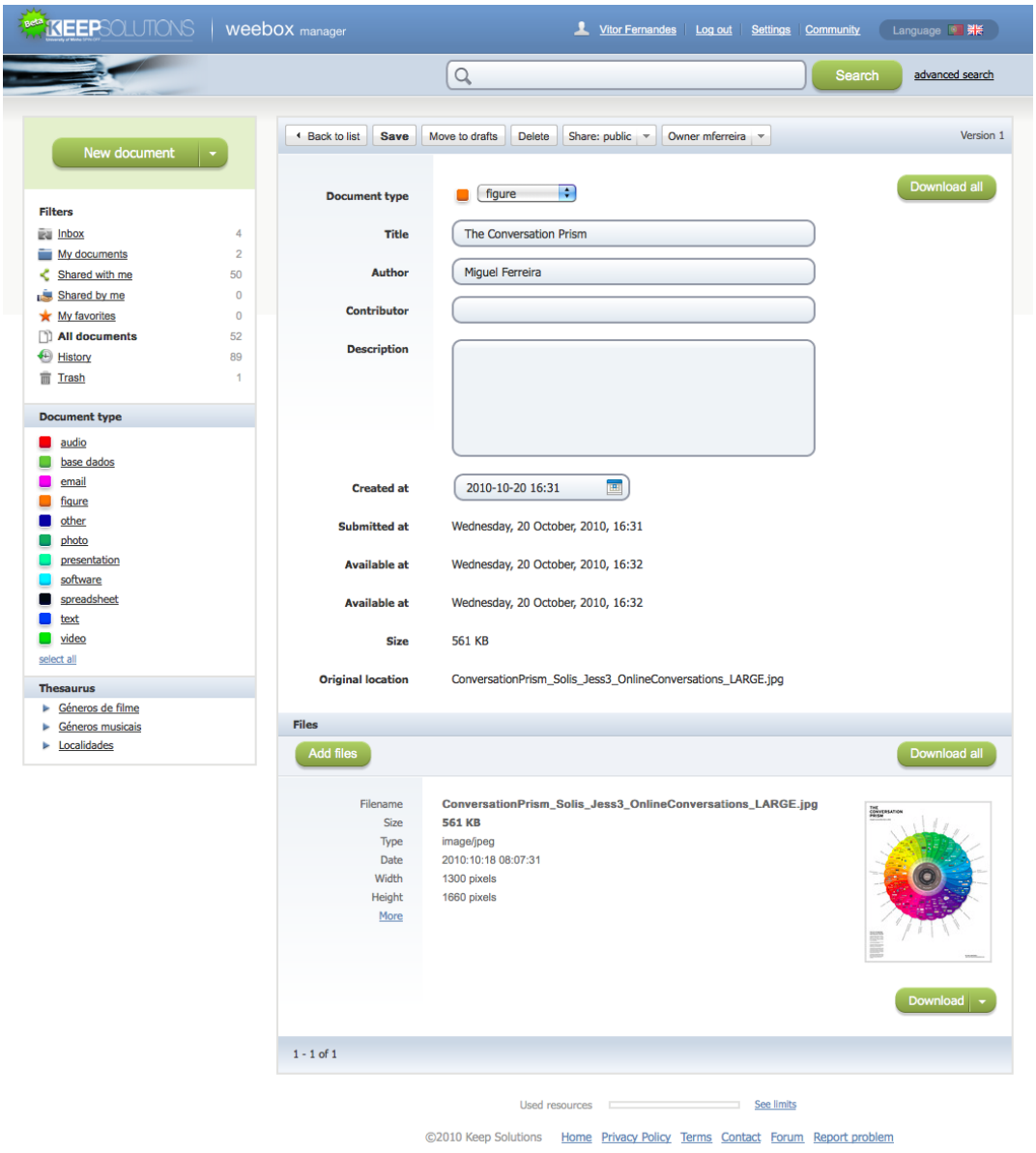

Figura 2.4: Página Web com os dados e ficheiros de um documento

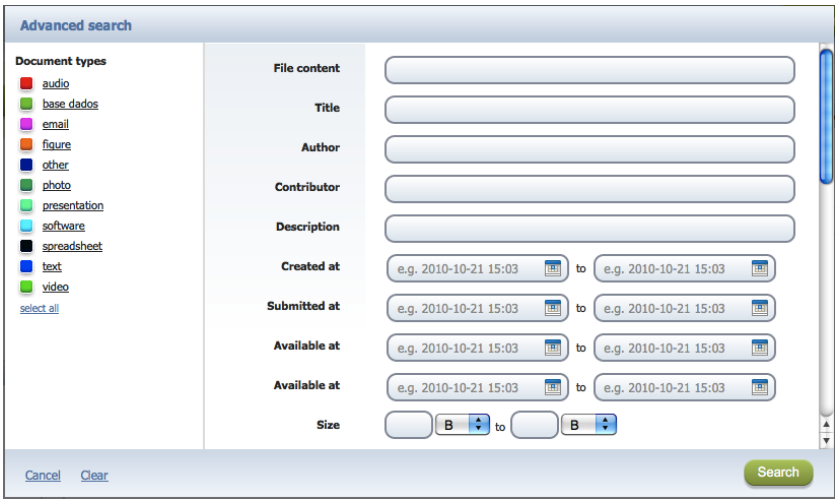

Figura 2.5: Painel Web para a pesquisa avançada

A observação das várias páginas da interface Web permite ter uma ideia geral do funcionamento das mesmas e das várias interfaces que devem ser apresentadas nas plataformas móveis.

2.1.3 iOS

O iOS é o sistema operativo móvel desenvolvido pela Apple [3]. Inicalmente foi desenvolvido apenas para o iPhone, mas foi depois usado em diversos produtos da marca (iPod Touch, iPad e Apple TV). É um sistema operativo proprietário e não pode ser usado por *hardware* de terceiros.

Existe um enorme conjunto de aplicações desenvolvidas para este sistema operativo e disponíveis na App Store da Apple. Para fomentar o desenvolvimento de aplicações para este sistema operativo é fornecido o iOS SDK (Software Development Kit), que contem o código, a informação e as ferramentas necessárias para desenvolver, testar, correr, fazer *debug* e optimizar aplicações. As ferramentas Xcode fornecem edição básica, compilação e *debug* para código desenvolvido [5]. Permitem também testar as aplicações num dispositivo com o iOS ou no iPhone Simulator, que é uma plataforma que imita o ambiente iOS e que corre num computador Macintosh. A linguagem usada para desenvolver aplicações para o iOS é o Objective-C, que é uma linguagem desenhada para permitir uma programação orientada a objectos sofisticada.

2.2 Interface

Nesta secção vai ser analisado o processo da construção da aplicação móvel resultante da adaptação dos interfaces Web do weebox, que pode ser dividido em duas fases: a arquitectura e o desenvolvimento.

2.2.1 Arquitectura

Nesta subsecção, vai ser especificada a arquitectura da aplicação móvel tendo em conta os comportamentos e as interfaces Web do weebox. A especificação da arquitectura da aplicação móvel pode ser dividida em quatros fases: o estudo e determinação dos casos de utilização, o desenho de interfaces móveis a partir da adaptação dos interfaces Web, a construção do diagrama de estados e a implementação de um modelo de dados (construção do diagrama de classes).

Casos de utilização

A interacção com a interface Web do módulo Manager do weebox, bem como a lista de funcionalidades desse módulo (Anexo A), permite identificar quais as funcionalidades que a aplicação para iPhone deve possuir. No entanto, apenas um conjunto dessas funcionalidades vão ser implementadas, porque a implementação de todas as funcionalidades não acrescenta benefícios à tese na sua completude. As funcionalidades a serem implementadas são as funcionalidades de autenticação, visualização da informação dos documentos, visualização dos ficheiros dos documentos e seus dados e pesquisa (básica ou avançada). No módulo weebox Manager o utilizador não consegue ver os seus ficheiros directamente no *browser*, sendo necessário efectuar o *download* dos mesmos para que sejam posteriormente visualizados. Na aplicação para iPhone, os ficheiros vão ser apresentados na própria aplicação, caso o tipo de ficheiro seja suportado. As funcionalidades a serem implementadas, de uma forma global na aplicação para iPhone são as seguintes:

- Autenticação (*login*);
- Sair (*logout*);
- Escolher filtro;

- Seleccionar tipo de documentos;
- Ver listagem de documentos resultantes da escolha do filtro;
- Pesquisa básica de documentos;
- Pesquisa avançada de documentos;
- Ver resultados das pesquisas (listagem de documentos);
- Ver todos os dados de um documento;
- Ver dados básicos dos ficheiros do documentos (listagem de ficheiros);
- Tentar visualizar ficheiros do documentos;
- Ver todos os dados de um ficheiro de um documento.

Desenho

Nesta subsecção, vai ser mostrada a transição das interfaces Web para as interfaces para iPhone. Antes de proceder ao desenho das interfaces para a aplicação para o iPhone é necessário proceder a uma análise da interface Web do weebox Manager, para que as interfaces a serem desenvolvidas sejam estruturadas correctamente. Isto, deve-se ao facto da aplicação não possuir capacidade para mostrar tanta informação de uma vez só, como na interface Web. O desenvolvimento da aplicação móvel passa por tentar incorporar os mesmos comportamentos da aplicação Web, usando sempre que possível os seus componentes de interface (cores, botões, imagens, estilo de letras, logótipos, entre outros componentes). Isto permite manter a coerência de *design* entre a interface Web e a interface móvel. No entanto, o uso de componentes de interface específicos da interface Web normalmente implica a diminuição da robustez e perfomance da aplicação móvel, ao contrário do que acontece com os componentes de interface nativos da plataforma móvel. É necessário então conjugar da melhor forma os diferentes componentes de interface (específicos e nativos) para que a aplicação móvel funcione da melhor forma e mantenha uma interface coerente com a interface Web. É preciso também ter em consideração o facto de as mensagens a serem apresentadas ao utilizador alternarem de acordo com o idioma escolhido.

Figura 2.6: Imagem inicial (*splash screen*)

O iOS permite mostrar uma imagem, designada por *splash screen*, enquanto a aplicação é iniciada ou a fazer interacções com o servidor. Assim, o primeiro passo foi a criação da imagem que aparece no início da aplicação (Figura 2.6). A imagem tenta incorporar alguns componentes de interface (logótipos, nomes, entre outras) e manter a cor azul da interface Web, contribuindo para uma coerência de *design* entre as aplicações. A cor azul vai ser usada sempre que necessário nas interfaces da aplicação móvel, mantendo a coerência entre as vários interfaces (móvel e Web) e não provocando um impacto visual ao utilizador.

Figura 2.7: Ecrã de autenticação com opção de servidor e teclado

A página de autenticação do weebox Manager é bastante simples, permi-

tindo ao utilizador introduzir o seu nome de utilizador e palavra-chave, no sítio respectivo, e proceder à sua autenticação. Existe também a possibilidade do utilizador entrar na página de registo ou recuperar as suas credenciais. Na aplicação móvel essa página é representada por um ecrã (Figura 2.7), que permite efectuar as mesmas operações da página de autenticação (excepto recuperar credenciais porque não faz parte dos casos de utilização da aplicação móvel, conforme 2.2.1). No ecrã existe também a possibilidade de inserir o servidor ao qual o utilizador se pretende ligar, sendo uma opção escondida porque o servidor, por norma, é sempre o mesmo (weebox.keep,pt). No fundo do ecrã existe um espaço vazio destinado ao teclado virtual, evitando que o seu aparecimento impeça a visualização de informação importante. O teclado é um dos factores que influencia o desenvolvimento de uma aplicação móvel.

Figura 2.8: Ecrã de registo

A página de registo do weebox Manager é simples, permitindo ao utilizador inserir o nome de utilizador, o seu nome completo, o seu correio electrónico e a sua palavra-chave e a confirmação. Para o processo de registo na aplicação móvel foi desenvolvido um ecrã (Figura 2.8), que funciona exactamente da mesma forma que a página Web para registo.

Após autenticação correcta na respectiva página Web, o utilizador entra na página principal da aplicação. A Figura 2.9 representa a página principal dividida em secções de acordo com as suas funções e comportamentos.

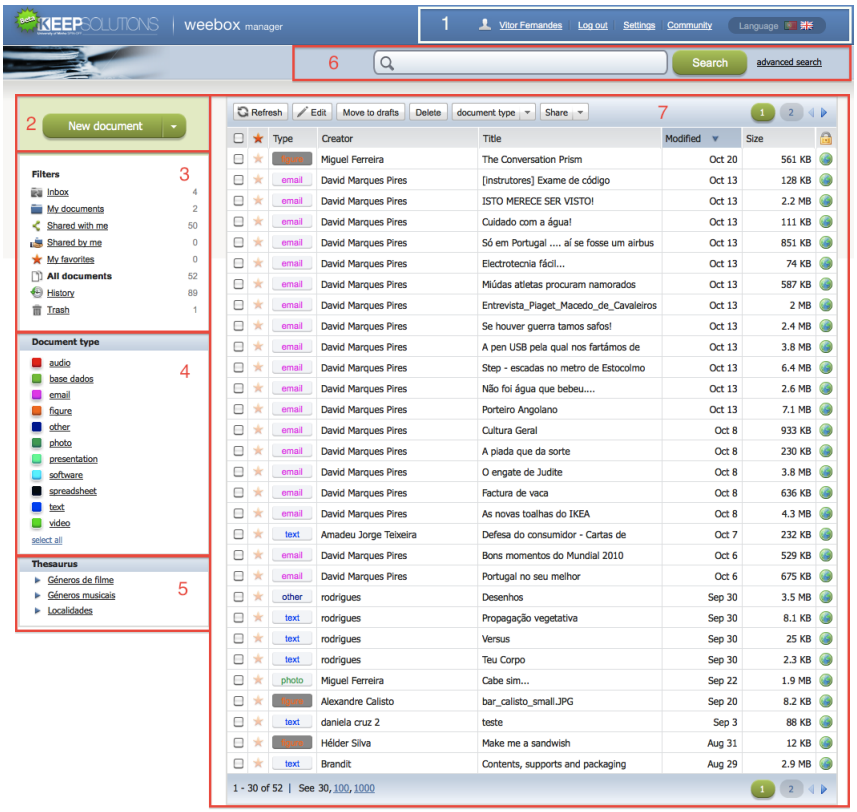

Figura 2.9: Página Web do weebox dividida em secções

O diagrama correspondente a esta divisão pode ser visto na Figura 2.10, sendo dado um nome significativo a cada secção da página para melhor identificação.

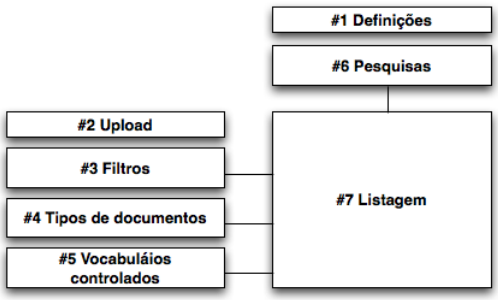

Figura 2.10: Diagrama da página Web do weebox dividida em secções

As funcionalidades da secção Definições e Upload, excepto a operação

de saída, não fazem parte dos casos de utilização desta versão da aplicação móvel e, portanto, não vão ser implementadas. A secção Filtro permite ao utilizador escolher um filtro dos oito possíveis (rascunho, meus documentos, partilhados comigo, partilhados por mim, os meus favoritos, todos os documentos, histórico e reciclagem). A secção Tipos de documentos permite ao utilizador seleccionar um conjunto de documentos de um ou mais tipos. A secção Vocabulários permite ao utilizador seleccionar vocabulários controlados [1]. As selecções nestas secções alteram a listagem de documentos apresentada no painel da secção Listagem e são consideradas nas pesquisas (básicas ou avançadas) que o utilizador efectuar (secção Pesquisas). No entanto, a pesquisa avançada torna a implementar componentes de interface para seleccionar vocabulários controlados e tipos de documentos, havendo uma replicação de componentes e comportamentos. Foi decidido que o componente de selecção de vocabulários controlados apenas vai aparecer na pesquisa avançada da aplicação móvel, devido à sua complexidade de implementação e à necessidade de poupar espaço.

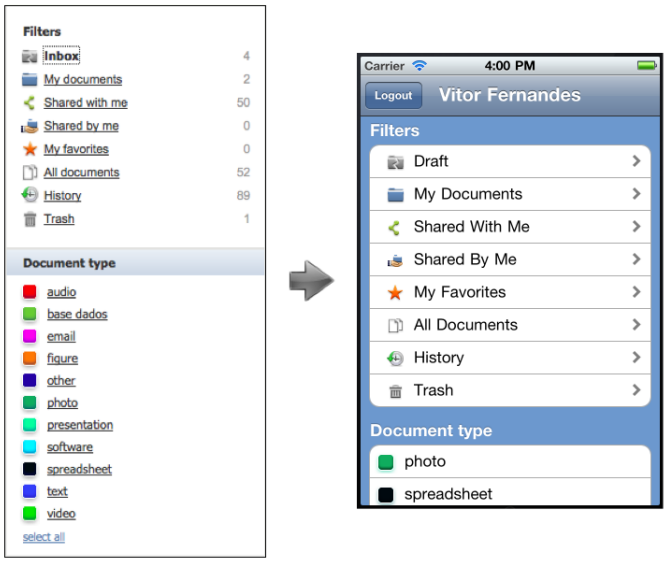

Figura 2.11: Ecrã de filtragem e escolha de tipos de documentos

Assim, foi desenvolvido um ecrã (Figura 2.11) para a aplicação móvel

[1] Conjunto de palavras ou frases, que são usadas para referenciar/indexar unidades de informação (documentos ou trabalhos) de forma a que estas sejam retornadas de forma mais simples numa pesquisa. Os vocabulários controlados podem possuir relações entre si, e daí serem um estrutura em árvore. Fornecem uma maneira de organizar o conhecimento para posterior recuperação.

que permite ao utilizador seleccionar um filtro, tal como a secção Filtro, e seleccionar um conjunto de tipos de documentos, como a secção Tipos de documentos. Cada filtro e tipo de documento são representados da mesma forma na interface móvel e na interface Web, contribuindo assim para a coerência de *design* entre as interfaces. No entanto o comportamento do ecrã é diferente do comportamento da interface Web. Se, por exemplo, o utilizador quiser todas as fotos de todos os documentos, na aplicação móvel selecciona primeiro o tipo de documentos que corresponde às fotos e depois o filtro correspondente, enquanto na aplicação Web selecciona primeiro o filtro e só depois o tipo de documento. Apesar desta alteração de comportamento, os resultados das selecções são os mesmos, ou seja, uma listagem de documentos de acordo com essas selecções.

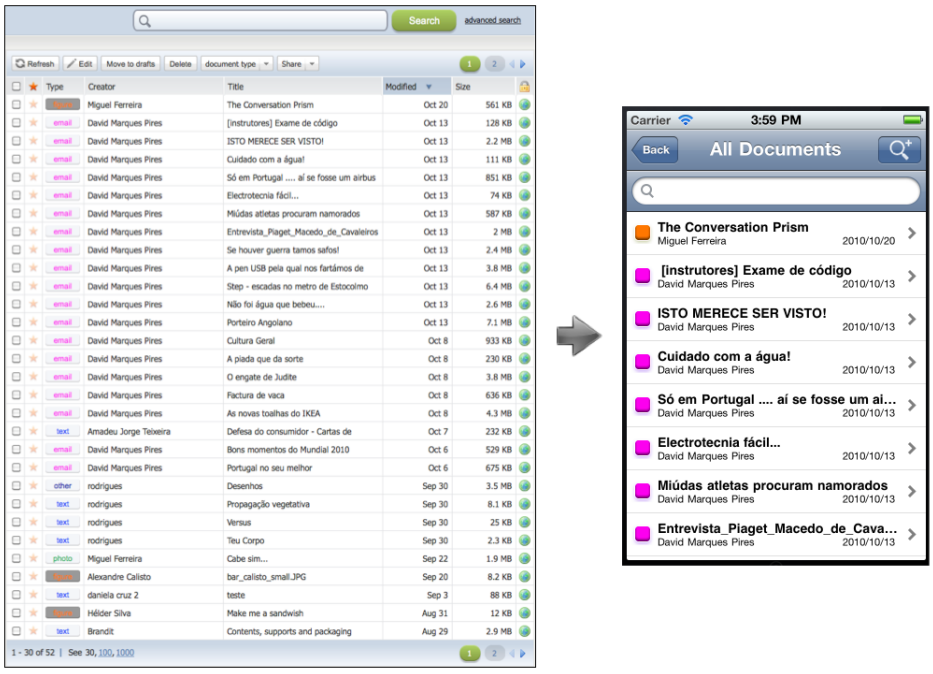

Figura 2.12: Ecrã de resultados de filtragem e pesquisa básica

A secção Listagem apresenta a listagem de documentos, tendo em conta as selecções de filtro, tipos de documento e vocabulários controlados que o utilizador efectuou, operações sobre essa listagem (que não fazem parte dos casos de utilização da aplicação móvel e e que por isso não vão ser implementadas) e controlos de paginação da listagem. Esta secção vai corresponder a um ecrã na aplicação móvel (Figura 2.12), apresentando apenas um número fixo de documentos podendo o utilizador carregar

mais documentos caso deseje, sendo assim implementada a paginação de conteúdos. Relativamente à apresentação dos documentos na listagem, esta vai ser bastante parecida com a listagem da página Web, apenas não apresentando o tamanho do documento e modificando a forma como o tipo de documento é mostrado devido a questões de espaço, o que contribui para a coerência de *design*. A pesquisa básica também vai aparecer neste ecrã, pois apenas modifica a listagem de documentos e, portanto, não necessita de um ecrã adicional.

Figura 2.13: Ecrã dos dados e ficheiros de um documento

Caso o utilizador carregue num documento na secção Listagem da página Web, o painel da listagem de documentos é substituído por um novo painel com as informações do documento e listagem dos seus ficheiros e respectivas informações. Permite a edição dos dados e outras operações que não fazem parte dos casos de utilização e que por isso não vão ser implementadas. A listagem de ficheiro também possui operações de controlo de paginação. A adaptação deste novo painel vai levar ao desenvolvimento de três ecrãs diferentes, pois a quantidade de informação a mostrar ao utilizador é elevada. O primeiro ecrã (Figura 2.13) vai mostrar qual o tipo de documento do documento em questão, os dados do documento e uma listagem de ficheiros. O tipo de documento é mostrado da mesma forma que na interface Web, mantendo assim a coerência de *design*. Relativamente à listagem dos ficheiros, apenas são carregados os dados de um número fixo de ficheiros, podendo o utilizador carregar mais, caso deseje, implementado assim a paginação tal como acontece na

interface Web. Na interface móvel apenas são mostrados dados básicos do ficheiro e uma imagem que permite identificar o tipo do ficheiro, ao contrário do que acontece na interface Web, porque o espaço é diminuto.

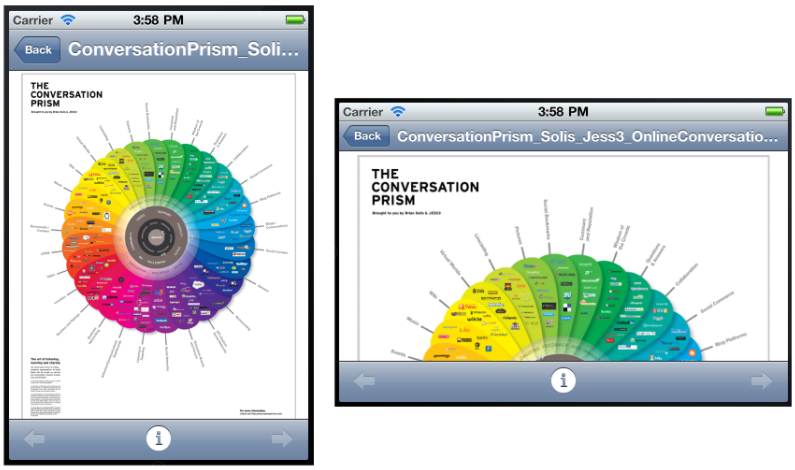

Figura 2.14: Ecrã de visualização de ficheiros

A selecção de uma entrada na listagem de ficheiros, leva o utilizador para outro ecrã (Figura 2.14) que tenta reproduzir o ficheiro em questão. O objectivo deste ecrã é substituir a operação de *download* do ficheiro na interface Web por uma operação de visualização na interface móvel, apesar do iPhone ser limitado em termos de formatos suportados. Neste ecrã, o utilizador pode navegar entre os vários ficheiros do documento ou consultar todos os seus dados noutro ecrã.

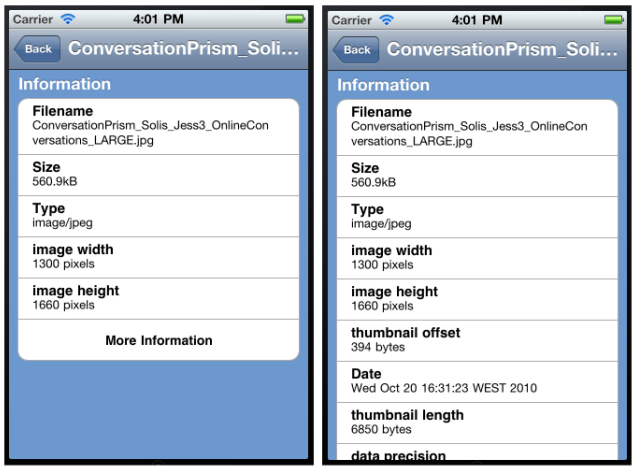

Figura 2.15: Ecrã de informações de um ficheiro

O ecrã (Figura 2.15), que apresenta os dados dos ficheiros, vai funcionar da mesma forma que na interface Web. Isto significa que numa fase inicial apenas os dados principais são mostrados ao utilizador, podendo o mesmo visualizar todos caso assim o deseje. Assim a interface móvel comporta-se da mesma forma que a interface Web, contribuindo para uma maior coerência entre interfaces e uma maior adaptação do utilizador.

Apesar da divisão da secção Listagem da página Web em vários ecrãs, o utilizador vai encontrar muitos comportamentos na aplicação móvel idênticos a comportamentos da interface Web. A principal diferença é a substituição da operação de *download* pela visualização de documentos.

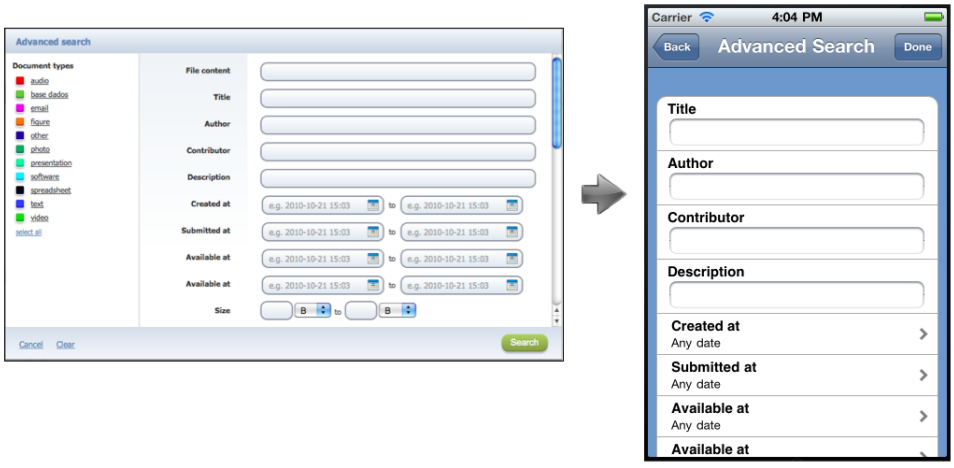

Figura 2.16: Ecrã de pesquisa avançada

Por fim, entramos na pesquisa avançada, que é um painel na página Web com duas secções. Na primeira secção o utilizador pode seleccionar os tipos de documentos sobre os quais pretende pesquisar, sendo a segunda secção influenciada por esta selecção, pois a mesma mostra um conjunto de campos pertencentes a união dos campos dos vários tipos de documentos seleccionados. Os campos podem ser de diferentes tipos (texto, volume de dados, data, vocabulários controlados), o que influencia a sua apresentação. Nas datas e valores é possível definir um intervalo de tempo ou valor, respectivamente, e nos vocabulários controlados (ou pastas) aparece um novo painel onde é possível definir um conjunto de vocabulários a serem pesquisados. É portanto uma operação bastante complexa devido aos diferentes tipos de dados que tem que tratar. Após a inserção dos dados a pesquisar, o utilizador pode terminar a pesquisa, sendo os resultados apresentados na secção Listagem da página Web.

A informação apresentada no painel da pesquisa avançada é demasiada para apresentar num só ecrã na aplicação móvel, pois, por exemplo, existe replicação de um componente para seleccionar tipos de documentos. Foi então tomada a decisão que a pesquisa avançada na aplicação móvel apenas ia ter em conta os tipos de documentos seleccionados num dos ecrãs anteriores (correspondente à Figura 2.11), não implementando um novo componente para a selecção dos tipos de documentos e mostrando assim menos informação. Então, foi desenvolvido o ecrã de pesquisa avançada (Figura 2.16) que permite inserir valores nos campos de texto. Para os outros tipos de dados foi necessário criar novos ecrãs.

Figura 2.17: Ecrã de vocabulários controlados

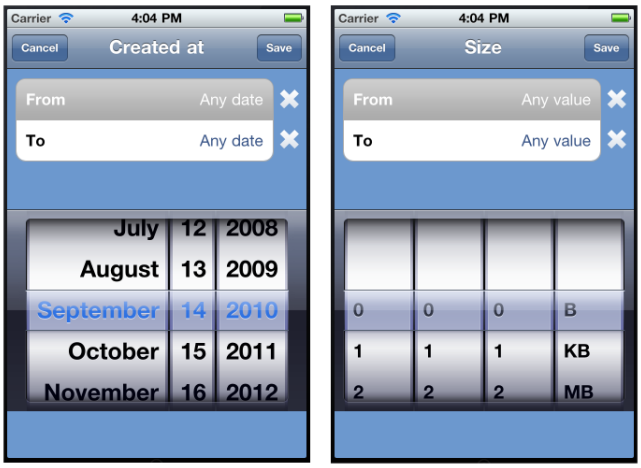

Figura 2.18: Ecrã de seleccção de datas e valores

Para a selecção de um conjunto de vocabulários controlados, foi desenhado um ecrã (Figura 2.17) que numa primeira fase mostra os vocabulários controlados principais. Como os vocabulários controlados são estruturados em árvore, o ecrã permite ao utilizador navegar na árvore e seleccionar os nodos/vocabulários que desejar.

Relativamente às datas e aos valores (normalmente tamanho do documento), foram postos noutro ecrã (Figura 2.18). De acordo com o tipo de dados a introduzir (data ou valor/tamanho), o ecrã é alterado, permitindo a selecção de dados correspondente a esse tipo. Após a selecção o utilizador pode guardar os dados, sendo os mesmos apresentados no ecrã de pesquisa avançada no campo correspondente.

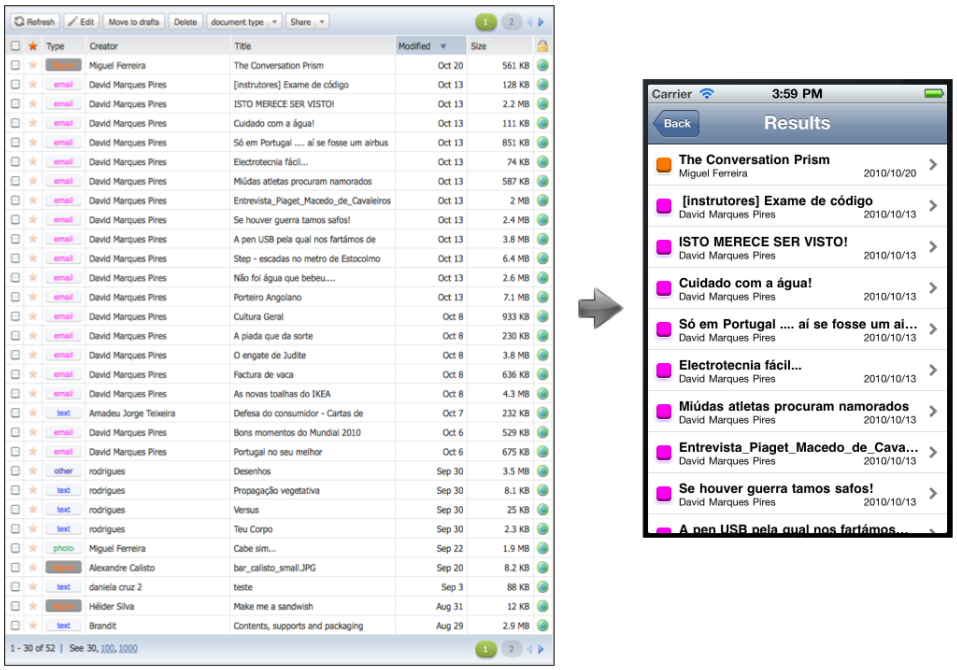

Figura 2.19: Ecrã de resultados de pesquisa avançada

Tendo o utilizador inserido todos os dados que deseja pesquisar na pesquisa avançada, o utilizador pode ir para um novo ecrã (Figura 2.19) que apenas faz a listagem dos documentos resultantes da pesquisa avançada. Cada documento será apresentado da mesma forma que no ecrã da Figura 2.12 mantendo, não só a coerência entre a interface móvel e Web, mas também a coerência entre os vários ecrãs da aplicação móvel. Também é feita a paginação dos conteúdos e a selecção de um documento conduz o utilizador ao ecrã de informação do documento.

A observação dos comportamentos da interface Web, bem como a divisão em secções e estudo das mesmas, permitiu o desenho da interface móvel

do weebox para o iPhone. Como se pode observar, sempre que possível, foram usadas imagens, cores, logótipos, entres outros componentes de interface no desenho dos ecrãs da aplicação móvel. Relativamente aos comportamentos, nem sempre poderão ser iguais devido à divisão em vários ecrãs, mas os resultados são os mesmos da aplicação Web.

Diagrama de estados

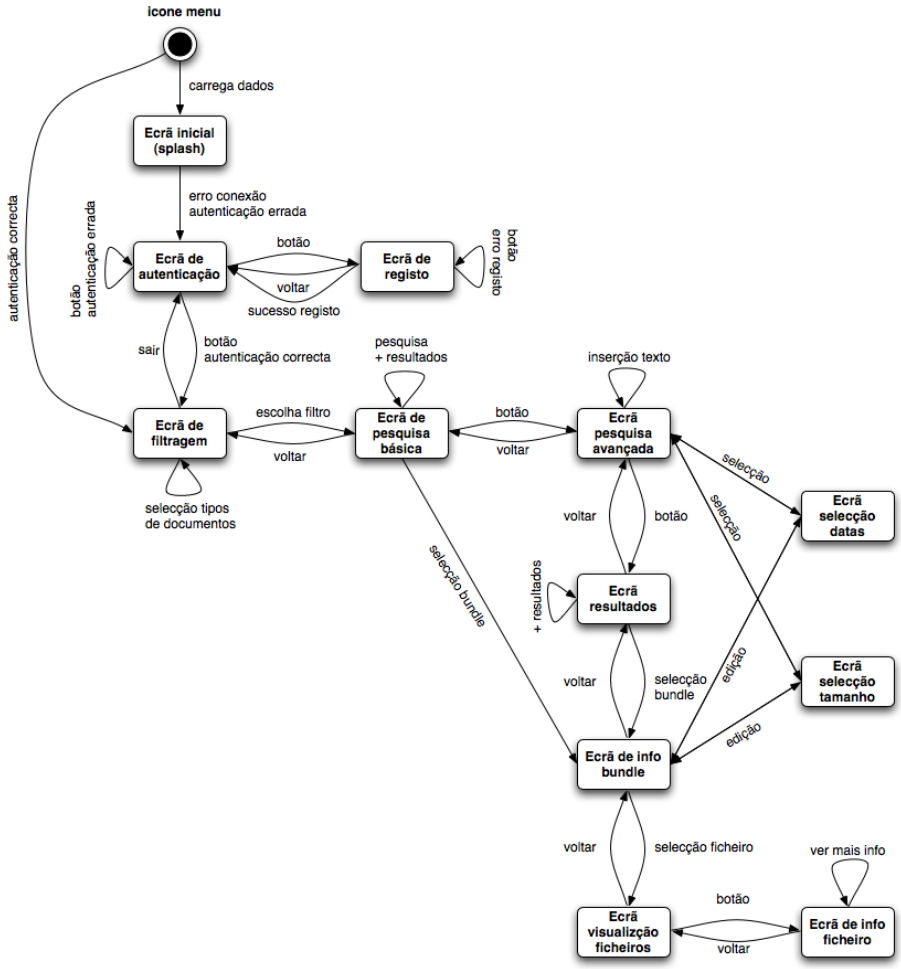

Figura 2.20: Diagrama de estados

A observação do diagrama de estados da aplicação móvel (Figura 2.20) permite-nos ter uma percepção mais global e simplista do comportamento do sistema e da possível navegação do utilizador nesse mesmo sistema. A comparação do diagrama de estados da aplicação para iPhone

com o diagrama de estados da aplicação Web (Figura 2.21), demonstra que a aplicação para iPhone é mais complexa em termos de estados. Este aumento de complexidade deve-se à adaptação das interfaces Web para interfaces iPhone, que levou ao desenvolvimento de diversos ecrãs na aplicação móvel para implementar as mesmas funcionalidades da aplicação Web. Portanto, a adaptação de interfaces Web em interfaces móvel leva ao aumento de complexidade em termos de estados da aplicação móvel em relação à aplicação Web.

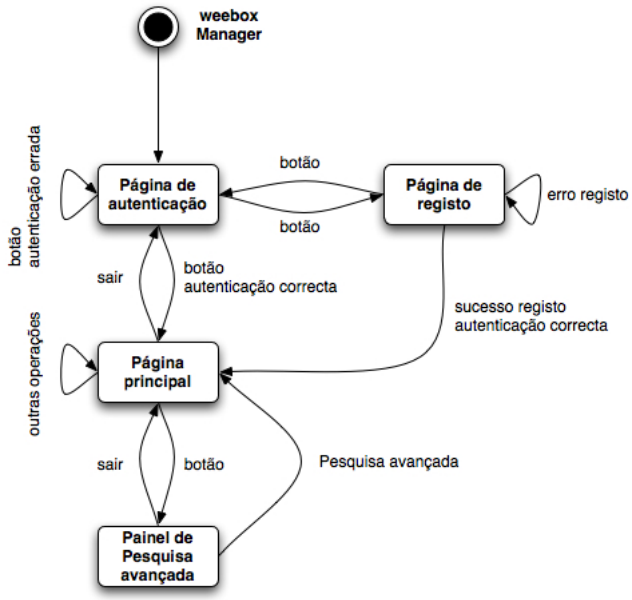

Figura 2.21: Diagrama de estados Web

Diagrama de classes

A aplicação móvel faz a interacção com o servidor, fazendo pedidos e tratando as respostas enviadas pelo mesmo. Foram implementadas classes com o propósito de guardar os dados obtido das respostas do servidor (classes de dados). O modelo de dados possui diferentes tipos de documentos, aos quais os documentos se encontram associados, defenindo os campos que os documentos devem possuir. Um documento pode conter vários ficheiros e, consequentemente, toda as informações relativas a esses ficheiros. Para além disso os campos dos tipos de documentos podem ser vocabulários controlados, ou seja, um documento que se encontra associado a um desses tipos pode possuir num campo vários vocabulários controlados. Todos estes dados vão ser guardados nas classes de dados,

resultantes da sincronização da aplicação com o servidor. Para além das classes de dados, foram implementadas classes de lógica e de apresentação que utilizam as classes de dados para mostrar as várias informações ao utilizador (Figura 2.22).

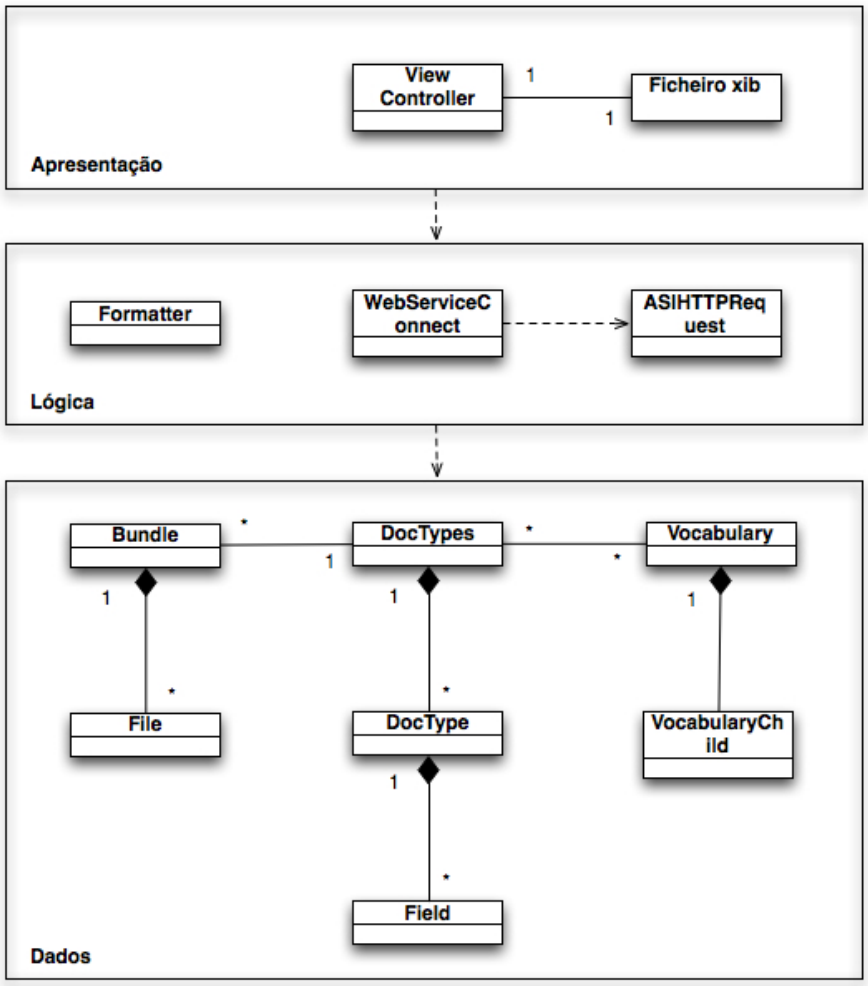

Figura 2.22: Diagrama de classes

Foram implementadas as seguintes classes de dados:

- Bundle - Contém a informação de um documento numa *hash*, onde cada entrada corresponde ao identificador de um campo do seu tipo de documento e o valor associado a esse campo;
- File - Contém metadados de um ficheiro de um documento;
- Field - Guarda os dados de um campo de um tipo de documento

(descrições em vários idiomas, qual o tipo de dados do campo, valor por omissão caso exista, se é editável, se é obrigratório);

- DocType - Contém os dados relativos a um tipo de documento (para além de uma lista de objetos Field guardas as descrições e labels do tipo de documento nos vários idiomas, a cor associada, entre outras informações);
- DocTypes - Contém os dados relativos a todos os tipos de documentos numa *hash* onde a chave é o identificador do tipo de documento e o valor é um objecto DocType;
- VocabularyChild - Contém os dados de um vocabulário controlado (labels e descrições nos vários idiomas, identificador, entre outros);
- Vocabulary - Guarda todos dos dados dos vocabulários controlados bem como as suas relações.

Vai ser também implementada uma classe designada por WebServiceConnect, responsável por fazer pedidos ao servidor e tratar as respostas. Para fazer os pedidos usa uma classe designada por ASIHTTPRequest. Para formatar os dados guardados nas instâncias das classes de dados, para que os mesmos possam ser apresentados nos ecrãs de forma legível para o utilizador, vai existir a classe Formatter, que permite também carregar/criar imagens, nomeadamente para os tipos de documentos. A criação de imagens vai ser usada para as imagens dos tipos de documentos, sendo criada uma imagem com a cor associada a cada um deles e posteriormente usada na interface móvel. Neste caso, as imagens criadas servem para aumentar a coerência de *design* e são usadas em alternativa aos componentes de interface nativos da plataforma móvel, pois os mesmos, apesar de tornarem a aplicação mais robusta, mais rápida de implementar e aumentam a performance, diminuiem a coerência entre as interfaces.

Por fim, temos os *view controllers*. No padrão de desenho MVC (Model View Controller) um *controller* fornece a lógica necessária para ligar os dados da aplicação com as *views* ou outros elementos visuais [14, 26]. Numa aplicação para iPhone, um *view controller* é uma classe que define os métodos para apresentar e controlar uma *view*, desempenhando um papel importante no desenvolvimento e implementação das aplicações (devido ao limitado espaço de ecrã para mostrar conteúdos) [6]. Os *view controllers* fornecem as infra-estruturas necessárias para também coordenar o aparecimento ou saída das diversas *views*. Como as funções dos mesmos são bastantes idênticas e cada ecrã vai possuir um respectivo *view controller*, decidiu-se apenas representar um no diagrama de

classes apresentado na Figura 2.22. Quase todos usam uma instância da classe WebServiceConnect para interagir com o servidor e uma instância da classe Formatter para formatar os dados a apresentar.

No diagrama de classes não são apresentados os *parsers*, que são classes auxiliares que são utilizadas para tratar os dados das respostas XML do servidor, e cujas funções de *parsing* retornam objectos das classes base, que são posteriormente utilizados pela aplicação.

2.2.2 Desenvolvimento

Nesta subsecção vai ser feita a análise do desenvolvimento da aplicação móvel, sendo apresentados alguns problemas encontrados durante o seu tempo de execução. O desenvolvimento pode ser dividido em diferentes partes: a conexão com o servidor, o desenvolvimento de *parsers* para as mensagens XML obtidas do servidor, o objecto *delegate* da aplicação móvel, tornar a aplicação multi-idioma e a construção dos *view controllers*.

Conexão com o servidor

A aplicação desenvolvida faz pedidos HTTP a um servidor REST [2]. O módulo weebox Core Services, possui uma API que fornece as várias operações sobre documentos, tipo de documentos, utilizadores, vocabulários controlados, ficheiros do documento, entre outras.

Para permitir a ligação entre a aplicação cliente com o servidor foi usada uma classe denominada por ASIHTTPRequest. Essa classe não é mais que uma extensão à API do CFNetwork, tornando mais simples alguns dos aspectos da comunicação com servidores Web [2]. Foi escrita em Objective-C e pode ser usada em projectos para o Mac OS ou para o iPhone. É, essencialmente, adequada para realizar pedidos HTTP básicos e permitir a interacção com serviços baseados em REST (GET / POST / PUT / DELETE). Possui ainda uma subclasse que facilita a apresentação de dados e arquivos e uma interface simples para o envio e recepção de dados de servidores Web. Permite guardar os dados dos pedidos/respostas em memória ou em disco e aceder facilmente a cabeçalhos HTTP. Suporta diferentes tipos de autenticação.

[2] Define um conjunto de princípios arquitecturais a partir dos quais é possível desenvolver serviços Web focados nos recursos do sistema, incluindo como os recursos são endereçados e transferidos sobre HTTP por uma enorme quantidade de clientes escritos em diferentes linguagens. Emergiu nos últimos anos como um modelo predominante para desenvolvimento de serviços Web, principalmente por ser consideravelmente mais simples de usar do que serviços baseados em SOAP e WSDL.

Como a maior parte dos pedidos ao servidor retornam ficheiros XML, foi necessário também desenvolver *parsers* que fizessem o *unmarshall* desses ficheiros e retornassem objectos das classes base.

Posteriormente, foi desenvolvida a classe WebServiceConnect, que utiliza a classe ASIHTTPRequest para fazer os pedidos ao servidor, e os *parsers*, para tratar os dados os ficheiros XML, que são obtidos como resposta aos pedidos HTTP ao servidor. Disponibiliza uma API que vai ser usada pelas outras classes para obter objectos com informações do servidor. Todos os métodos implementados, fazem autenticação básica no servidor, sendo o nome do utilizador e a palavra-passe inserida no cabeçalho HTTP do pedido ao servidor e caso ocorra erro na conexão com o servidor é retornada uma excepção com esse erro. Os métodos desta classe permitem aceder a todas as operações básicas do servidor REST. As respostas XML obtidas passam por um *parser*, dando origem a objectos das classes base, tornando os dados dessas respostas mais fáceis de manipular.

Parsers XML

A maior parte das respostas obtidas do servidor são mensagem XML, e foi necessário desenvolver classes que tratassem essas mensagens e criassem objectos do modelo de dados, que vão ser posteriormente utilizados pela aplicação. Essas classes usam uma classe auxiliar (NSXMLParser), que fornece métodos para fazer o *parser event-based* das mensagens XML [4]. Este tipo de *parser* consome menos memória que um *parser tree-based* pois trabalha apenas com um elemento XML de cada vez e não com todos, sendo então adequado para situações onde a perfomance é um objectivo [4, 16]. No entanto, é menos adequado para tarefas que exigem que o XML seja sujeito a consultas por parte dos usuários ou então alterado e escrito de novo para ficheiro [16]. É, portanto, um tipo de *parser* que possui vantagens e desvantagens mas para este caso de estudo é o ideal, pois o que se pretende é perfomance.

As classes implementadas para realizarem o *parser* das mensagens XML e criar objectos do modelo de dados foram as seguintes:

- ParserBundle - Responsável por fazer o *parsing* de mensagens XML como a que podemos encontrar no Anexo B. Cria um objecto Bundle com as informações do documento;
- ParserDoctype - Responsável por fazer o *parsing* de mensagens XML como a que podemos encontrar no Anexo F. É um *parser* bastante complexo, como podemos observar na mensagem XML

que se encontra no anexo referido. Cria um objecto DocType que contém todas as informações e campos de um tipo de documento;

- ParserFiles - Esta classe vai ser responsável por fazer o *parsing* de mensagens XML como a que podemos encontrar no Anexo C. Cria um objecto File com todas as informações relativas a um ficheiro de um documento;

- ParserSearch - Responsável por fazer o *parsing* de mensagens XML como a que podemos encontrar no Anexo E. Cria um objecto SearchBundles com todas as informações lidas (número de documentos resultantes da pesquisa e informações de um conjunto de documentos);

- ParserUser - Esta classe vai ser responsável por fazer o *parsing* de mensagens XML como a que podemos encontrar no Anexo G. A sua função de *parsing* retorna apenas o nome do utilizador;

- ParserVocabulary - Responsável por fazer o *parsing* de mensagens XML como a que podemos encontrar no Anexo H. Cria um objecto Vocabulary com todas as informações e as relações entre os vocabulários controlados que aparecem na mensagem.

Portanto, estes *parsers* permitem retornar objectos das classes base para serem usados posteriormente por outros objectos (como, por exemplo, os *viewcontrollers*), facilitando o acesso à informação.

Multi-idioma

A internacionalização é um objectivo crucial da aplicação que, de acordo com as definições do dispositivo, deve apresentar mensagens no idioma correcto. Existem duas formas de tornar a aplicação multi-idioma.

A primeira forma é alterar os ficheiros de interface, criando para cada um diferentes vistas para os diferentes idiomas. É relativamente fácil aplicar esta metodologia, mas não se enquadra no que se pretende pois, normalmente, as tabelas são preenchidas dinamicamente tornando impossível a utilização das vistas pré-compiladas. A segunda forma, e a utilizada, é recorrer a um ficheiro que guarda para cada tipo de localização uma espécie de dicionário. Para tal foi adicionado ao projecto um ficheiro designado por *prefs.list* que contêm as traduções para os vários idiomas. Estas traduções são carregadas no início para memória de acordo com o idioma utilizado no dispositivo. Caso esse idioma não apareça no ficheiro então é carregado o idioma por omissão.

Quando o dicionário for carregado, este é guardado no objecto *delegate* da aplicação, e vai ser invocado sobretudo para obter traduções para preencher botões, filtros, entradas nas tabelas, nomes de ecrãs, entre outros.

Application Delegate

O objecto *delegate* da aplicação é responsável por tratar inúmeras mensagens críticas de sistema e deve estar presente em todas as aplicações para iPhone. O objecto pode ser instanciado em qualquer classe, pois adopta o protocolo UIApplicationDelegate. Os métodos deste protocolo definem as ligações do ciclo de vida da aplicação e são uma forma de implementar determinados comportamentos, sendo a implementação de alguns métodos obrigatória no objecto *delegate* da aplicação.

A função principal do objecto *delegate* da aplicação é monitorizar o comportamento de alto-nível da aplicação, recebendo mensagens dos outros objectos conforme a ocorrência de certos eventos, como a falta de memória. O objecto *delegate* de acordo com a mensagem recebida executa determinado código e implementa o comportamento desejado.

Neste caso de estudo, o *delegate* da aplicação serve também para gerir a informação relativa aos tipos de documentos, ao dicionário de idioma (conjunto de mensagens no idioma do dispositivo), aos vocabulários controlados, entre outros. Sempre que for necessário aceder a alguma das informações é criada uma instância do *delegate*, permitindo assim aceder às suas variáveis e consequentemente às suas informações, uma vez que o objecto *delegate* apenas é libertado quando a aplicação termina.

Todos os objectos *delegate* de todas as aplicações para iPhone possuem um método designado por applicationDidFinishLaunching. Este método é o local ideal para executar várias inicializações e tarefas de configuração, especialmente para recuperar certos estados anteriores da aplicação ou para definir a janela inicial e vistas da aplicação. Neste caso de estudo, esse método vai ser responsável por carregar o dicionário, fazer autenticação do utilizador (carregando o nome de utilizador, a palavra-passe e o servidor do ficheiro de sistema) e encaminhar para o ecrã principal, que pode ser o ecrã de autenticação caso as credenciais sejam inválidas ou para o ecrã de filtragem caso as credenciais sejam válidas.

View controllers

Para cada um dos ecrãs foi desenvolvido um *view controller*, que vai ser responsável pelos comportamentos do ecrã e pela informação que é mostrada ao utilizador. Para uma descrição completa de cada *view*

controller, consulte o Anexo J.

2.3 Conteúdo

O weebox é uma aplicação rica em contéudos multimédia e, portanto, a sua adaptação para uma aplicação móvel provoca problemas de disseminação de conteúdo. Os ficheiros que se encontram no weebox podem ser de tipos não suportados e que por isso não podem ser visualizados no iPhone. Existem duas grandes soluções para este problema: a emulação ou a conversão [7]. A emulação é uma solução que consiste em desenvolver formas de visualização para tipos de ficheiro não suportados [7]. Neste caso de estudo, a emulação era uma má solução porque o weebox pode conter um conjunto diversificado de ficheiros e era incompatível desenvolver formas de visualização para os diversos tipos de ficheiro. Resta portanto a conversão que pode ser realizada do lado do cliente ou do lado do servidor [7]. Neste caso de estudo, optou-se por desenvolver um servidor de conversões/migrações, já que a conversão no dispositivo (cliente) é um processo complicado porque necessitava de grande capacidade de processamento e de tempo, o que iria contra os objectivos de ter uma aplicação rápida.

Esta secção apresenta uma especificação da arquitectura e uma análise ao processo de desenvolvimento do servidor de migrações, bem como a sua posterior utilização na aplicação móvel para converter ficheiros cujos formatos não são suportados de forma nativa.

2.3.1 Arquitectura

Nesta subsecção vai ser especificada a arquitectura do servidor de migrações que pode ser dividida em duas fases: a descrição do funcionamento geral do servidor (problemas a resolver, autenticação, sistema de plugins, entre outras questões) e a implementação de um conjunto de classes (diagrama de classes) que permitam por em prática as funcionalidades descritas no funcionamento geral.

Funcionamento geral

O servidor a ser implementado vai ser um servidor de migrações genérico, ou seja, independente da plataforma para qual o ficheiro vai ser convertido. Por exemplo, o servidor de migrações deve ser capaz de converter um vídeo para várias plataformas, e não apenas para o iPhone

e/ou qualquer outro dispositivo móvel (cujas gamas de formatos aceites é também restrita).

Para além disso, o servidor deve possuir a capacidade de obter o ficheiro origem de outros servidores de ficheiros, como o caso do weebox. Como muitos desses servidores lidam com autenticação, o servidor de migrações também tem que lidar com esse problema.

Aliado ao servidor, existe uma cache de ficheiros convertidos. Quando o servidor receber um pedido de conversão, vai verificar se o ficheiro convertido que se pretende existe na cache. Caso exista, o ficheiro convertido é enviado, caso contrário o servidor vai buscar o ficheiro origem, converte-o para o formato desejado e envia o ficheiro resultante da conversão. Para qualquer um dos casos, o servidor tem de verificar se o utilizador que realizou o pedido possui permissões que lhe permitam realizar esta operação sobre o ficheiro original.

Outro problema com que o servidor tem de lidar é com a existência de pedidos iguais por parte de utilizadores diferentes. Esses pedidos, independentemente do utilizador, devem obter o mesmo ficheiro e como tal só deve ser efectuada uma conversão (isto se o ficheiro convertido ainda não estiver na cache).

O processo de conversão passa essencialmente por duas fases. A primeira fase consiste em ir buscar o ficheiro original, que pode estar alojado em diferentes servidores de ficheiros. Além do ficheiro pode ser necessário obter outras informações sobre o ficheiro, como o nome, o tipo ou até mesmo verificar se o utilizador que fez o pedido ao servidor possui autorização para obter o ficheiro e essas informações. A segunda fase passa pela conversão, sendo o ficheiro original convertido para o formato desejado, guardado na cache de ficheiros do servidor e posteriormente retornado ao utilizador que fez o pedido. Falta referir, que a conversão de um formato para outro pode ser realizada por diversos conversores/-softwares de conversão.

Como o servidor tem que lidar com vários servidores de ficheiros e vários softwares de conversão, surge a necessidade de adicionar, em tempo de execução, ao servidor, conversores e formas optimizadas de interagir com os diversos servidores de ficheiros, ou seja, surge a necessidade de haver um sistema de *plugins* que são carregados em tempo de execução pelo servidor. Este sistema de *plugins* evita ter que se reiniciar o sistema para adicionar novas funcionalidades ao servidor. Existem assim dois tipos diferentes de *plugins*: os que fazem pedidos aos diversos servidores de ficheiros (autenticação, autorização, nome do ficheiro, o próprio ficheiro, entre outras) de uma forma optimizada (a forma como são feitos e tratados dependem do servidor onde o ficheiro se encontra), designados

por *handlers*, e os que tratam da conversão propriamente dita (convertem o ficheiro original, guardando o ficheiro convertido na cache), designados por *converters*.

Para servir o propósito da existência deste sistema dinâmico de leitura de *plugins*, o servidor tem que possuir duas interfaces Java distintas: a interface Handler e a interface Converter, que permitem respectivamente implementar *handlers* e *converters*. A interface Handler especifica, portanto, os métodos que um *handler* deve possuir, que são os seguintes:

- authorization - Permite verificar se o utilizador tem autorização sobre o ficheiro;
- getFileName - Permite obter o nome do ficheiro;
- getInputStream - Permite obter o ficheiro;
- getMimeType - Permite obter o formato do ficheiro;
- getSyntax - Permite saber quais argumentos que é necessário passar ao *handler* para o seu correcto funcionamento;
- getName - Permite obter o nome do *handler*;
- getVersion - Permite obter a versão do *handler*;
- getID - Permite obter o identificador único do *handler*;
- init - Permite definir o código a executar quando o *handler* for iniciado/carregado;
- shutdown - Permite definir o código a executar quando o *handler* for terminado/eliminado.

A interface Converter, por seu lado, apenas especifica os métodos que um *converter* deve possuir, sendo esses métodos os seguintes:

- convert - Permite converter o ficheiro original;
- getSyntax - Permite saber quais os argumentos que é necessário passar ao *converter* para o seu correcto funcionamento;
- getName - Permte obter o nome do *converter*;
- getVersion - Permite obter a versão do *converter*;
- getID - Permite obter o identificador único do *converter*;

- init - Permite definir o código a executar quando o *converter* for iniciado/carregado;
- shutdown -Permite definir o código a executar quando o *converter* for terminado/eliminado.

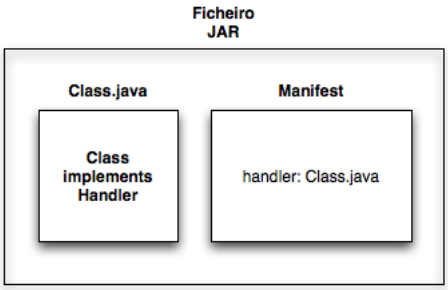

Figura 2.23: Constituição de um JAR com um *handler*

O servidor carrega ficheiros JAR (Java Archive), que possuem *handlers* e/ou *converters*. Na Figura 2.23 temos a constituição de um JAR com um *handler*, ou seja, uma classe que implementa a interface Handler. Essa classe vai ser especificada no *manifest* [3] do ficheiro JAR, sendo a chave da entrada igual a *handler* e o valor igual ao nome da classe. No caso do ficheiro JAR ser constituido por um *converter*, apenas é alterada a chave da entrada no *manifest* para *converter*, sendo o valor a classe correspondente ao *converter*. A especificação da classe que corresponde ao *handler* ou ao *converter* no *manifest* serve para o servidor saber qual a classe que vai instanciar de forma a adicionar novas funcionalidades. O servidor verifica se as classes implementam a interface da forma correcta e, em caso de sucesso, cria instâncias dessas classes e adiciona-as ao servidor, ficando então disponíveis para serem invocadas pelos utilizadores. Os ficheiros JAR têm que ser colocados numa pasta definida para tal, pasta essa que é analisada quando o servidor é iniciado e a, partir daí, é analisada de minuto a minuto. Caso existam novos ficheiros ou *updates*, o servidor tenta ler esses ficheiros.

Os *handlers* e *converters* a serem usados têm que ser especificados no pedido que é feito ao servidor. Para tratar todos os pedidos realizados pelos utilizadores, o servidor vai possui um *servlet* [4] de nome Migrator,

[3] Arquivo específico contido num arquivo JAR, usado para definir a extensão e um pacote de dados relacionados, designado por MANIFEST.MF. É um arquivo de metadados que contém pares nome-valor dividido em secções diferentes.

[4] É um componente do lado servidor que gera dados HTML e XML para a camada de apresentação de um aplicativo Web. É uma classe na linguagem de programação Java

que será responsável por retornar o ficheiro convertido caso a conversão ocorra com sucesso ou o mesmo já esteja na cache de ficheiros do servido. Retorna erro caso a conversão não ocorra com sucesso, se o utilizador não possuir permissões sobre o ficheiro a converter, se o ficheiro a converter não existir, se tiver danificado, ou tiverem ocorrido outros problemas durante o processo. Para que o servidor consiga fazer a conversão, o utilizador tem que passar como parâmetro no URL o *handler* e o *converter*, bem como os argumentos respectivos e o formato para o qual o ficheiro vai ser convertido.

Por último, existe mais um problema que o servidor tem que ultrapassar, que é a existência de diferentes tipos de conversores, que se passam a descrever a seguir.

Conversores 1-1 Também designados por conversores básicos pois são convertem um ficheiro de áudio, vídeo, imagem ou documento de texto de um formato para outro. A conversão 1-1 é relativamente simples. O utilizador tem que identificar o *handler* e os seus argumentos, permitindo o servidor obter o ficheiro original. Caso seja necessário autenticação para o utilizador poder aceder ao ficheiro, o servidor pede o nome de utilizador e a palavra-chave que permitam esse acesso. O utilizador também tem que especificar qual o *converter* que deseja usar, os seus argumentos e o formato para o qual o ficheiro vai ser convertido. Após a conversão, o ficheiro resultante é enviado ao utilizador. O processo de conversão só ocorre se o ficheiro convertido que o utilizador pretende não estiver na cache de ficheiros, pois se o mesmo tiver na cache, é, de imediato, retornado ao utilizador. Na Figura 2.24 é temos um exemplo deste processo de conversão, onde uma imagem no formato JPEG é convertida para o formato PNG.

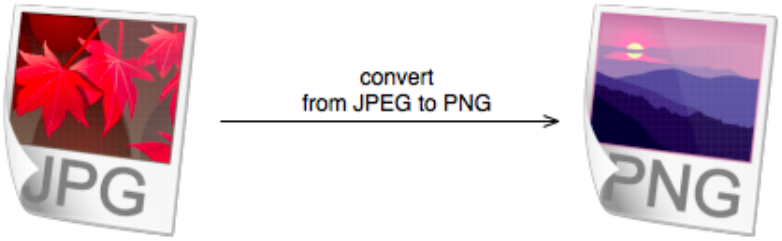

Figura 2.24: Processo de conversão 1-1

Conversores 1-N Para a conversão 1-N, o processo é ligeiramente mais

responsável por processar requisições e respostas dinamicamente, adicionando dessa maneira novos recursos aos servidores.

complicado. Estas conversões ocorrem, por exemplo, quando o utilizador pretende obter ficheiros de um ficheiro comprimido ou quando um PDF é convertido em várias imagens. O utilizador passa como parâmetros o *handler* com os seus argumentos e o *converter* e respectivos argumentos. Neste caso, o formato final é HTML, pois para este tipo de conversões o ponto de entrada no resultado é um ficheiro HTML com *links* para os ficheiros resultantes. Esta página HTML é criada da primeira vez que o pedido de conversão é feito. O método de conversão do *converter* pode executar logo a conversão completa colocando os ficheiros resultantes na cache de ficheiros do servidor, ou então pode apenas executar a conversão a pedido do utilizador, ou seja, quando este carregar num *link* do ficheiro HTML. Para além disto quando o HTML é gerado, todos os dados passados pelo utilizador como parâmetros são guardados num ficheiro XML (onde são registados todos os pedidos). Isto é importante, pois o *link* para um ficheiro apenas contem o identificador do pedido que gerou o HTML e o nome do ficheiro que o utilizador pretende obter, e assim pelo identificador do pedido o servidor consegue obter todos os parâmetros desse pedido consultando o ficheiro XML. Esses parâmetros são para verificar se o utilizador possui autorização sobre o ficheiro original (o que foi convertido) e/ou converter a pedido (obter o ficheiro original e apenas dar origem ao ficheiro que o utilizador carregou no HTML), sendo o ficheiro convertido posteriormente devolvido. Na Figura 2.25 temos um exemplo deste processo de conversão, onde para um PDF é criado um ficheiro HTML com *links* para as imagens resultantes da conversão.

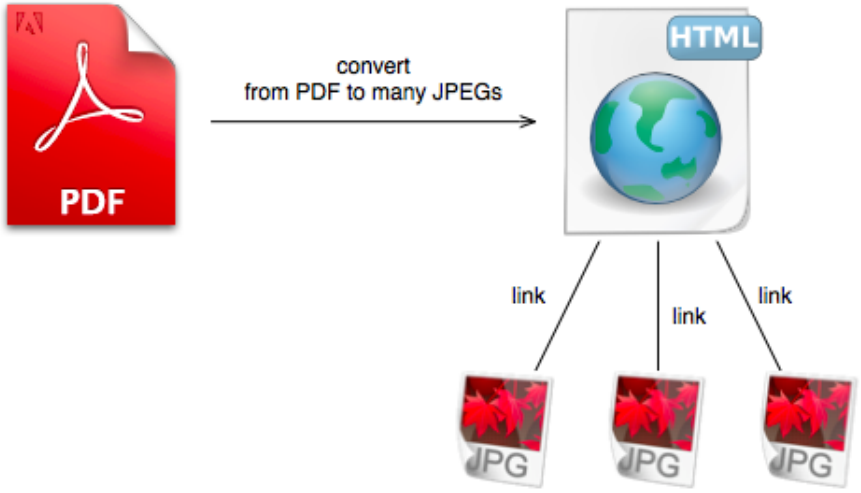

Figura 2.25: Processo de conversão 1-N

Conversores N-N Relativamente aos conversores N-N, vai ser um tipo de conversão que não vai ser implementada de momento no servidor. Isto, deve-se sobretudo ao elevado tempo de implementação e pelo facto de poderem ser substituídos muitas vezes pelos outros tipos de conversão. No entanto a solição consiste em passar como parâmetros no URL múltiplos *handlers* e argumentos, permitindo ao servidor obter todos os ficheiros que o utilizador pretende. Seria usado um sistema de numeração para passar todos os *handlers* como parâmetros (i.e handler1, handler2, etc). O utilizador teria também que passar como parâmetros o *converter* e seus argumentos. Posteriormente o servidor invocaria a função de conversão para todos os ficheiros obtidos. Caso da conversão apenas resultasse um ficheiro o mesmo era retornado, caso resultassem múltiplos ficheiros então era usado o mesmo sistema dos conversores 1-N, sendo retornado um ficheiro HTML com um *link* para todos os ficheiros que poderão resultar da conversão.

Diagrama de classes

A análise ao funcionamento geral do servidor de migrações permitiu obter dados sobre a estruturação do mesmo e, consequentemente, quais as classes que são parte constituinte do servidor de migrações.

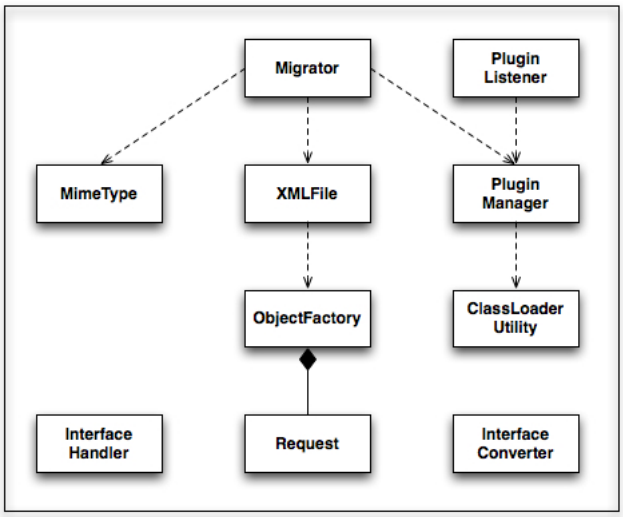

Figura 2.26: Diagrama de classes do servidor

A Figura 2.26 representa o diagrama de classes implementado para servir

todos os pedidos do utilizador, sendo também representadas as interfaces que permitem desenvolver novos *handlers* e *converters*. As classes implementadas foram as seguintes:

- MimeType - Uma instância desta classe devolve o tipo (*mimetype*) de um determinado ficheiro;
- Request - Uma instância desta classe permite tratar os dados de um pedido;
- ObjectFactory - Uma instância desta classe permite criar novos Requests, sendo posteriormente guardados num ficheiro XML;
- ClassLoaderUtility - Uma instância desta classe permite interagir com os ficheiros JAR, para que possam ser retiradas as classe que correspondem aos *plugins*;
- PluginManager - Uma instância desta classe executa o carregamento de *plugins*, sendo adicionados ao servidor. Para executar essa leitura utiliza instâncias da classe ClassLoaderUtility. Apenas uma instância desta classe pode ser criada;
- PluginListener - Uma instância desta classe é criada quando a aplicação é colocada num servidor aplicacional, criando assim a instância do PluginManager. Quando o servidor aplicacional é parado ou reniciado, a instância desta classe é eliminada, eliminando também a instância do PluginManager;
- Migrator - Servlet que vai responder aos pedidos do utilizador. Utiliza a instância criada do PluginManager para tratar os pedidos e envia a resposta para o utilizador.

2.3.2 Desenvolvimento

Nesta subsecção é feita a análise do desenvolvimento do servidor de migrações, sendo apresentados alguns problemas encontrados durante o seu tempo de execução. O desenvolvimento pode ser dividido em diferentes partes: a pesquisa de software de conversão, a implementação de diferentes *converters*, a implementação de diferentes *handlers*, e a implementação e funcionamento do servlet Migrator

Pesquisa de software

Antes de proceder ao desenvolvimento do servidor de migrações foi feito um estudo acerca dos softwares ou *scripts* mais utilizados para a conversão de ficheiros.

Para imagens e criação de *thumbnails*, é comum o uso do ImageMagick para executar conversões, edição e composição de imagens. É um software que consegue ler, converter e criar imagens nos mais variados formatos (mais de 100), podendo ser usado para executar operações sobre uma determinada imagem (rodar, espelhar, escalar, transformar, alterar cores, criar determinados efeitos entre outras) [13, 24]. Sendo assim, é uma ferramenta completa e que nos permite assegurar a conversão de imagens para os formatos desejados.

Quanto a ficheiros de áudio, foi escolhido o FFmpeg, que é uma solução completa e para várias plataformas para guardar, converter e fazer *streaming* de áudio ou de vídeo [8, 21]. Assim, torna-se uma óptima solução para a conversão de ficheiros de áudio.

Para a conversão de ficheiro de vídeo, a decisão acabou por recair sobre duas soluções: o HandBrake e o MEncoder.

O HandBrake é um software multiplataforma e multitarefa, sendo o código do mesmo *open-source*. Permite a conversão de vários formatos de vídeo, mas foi desenvolvido com o propósito de converter vídeos MPEG (incluindo vídeos DVD) em vídeos MPEG-4 (podendo o mesmo ficar em formato .mp4 ou em .mkv). É normalmente usado para converter vídeos para outros formatos a fim de poderem ser visualizados em dispositivos como os iPods, iPhones ou até mesmo com o QuickTime Player (desenvolvido pela Apple) [12, 23].

O MEncoder é uma ferramenta *open-source* que usa a linha de comando para a codificação de vídeos e que permite converter todos os formatos aceites pelo MPlayer (com o qual o MEncoder é distribuído juntamente), podendo o utilizador escolher os diversos filtros de áudio e vídeo do vídeo resultante da conversão [10, 25]. É mais complicado de usar que o HandBrake devido à quantidade de opções que possui, mas, por outro lado, é muito mais completo permitindo uma conversão mais controlada por parte do utilizador.

Relativamente a ficheiros comprimidos (zip, rar, tar, entre outros formatos) não existem conversores no verdadeiro sentido da palavra, mas sim, programas que permitem descomprimir esses ficheiros. Neste caso, essa descompressão será tratada como conversão. Como um ficheiro comprimido pode ter vários ficheiros no seu interior, a conversão será de 1-N e pode ser potencialmente recursiva pois podem existir outros ficheiros

comprimidos no interior do ficheiro comprimido a ser convertido. Para Linux existem comandos nativos que permitem essa conversão/descompressão, como o Zip (para formatos zip) e o Tar (para formatos rar, tar, tar.gz). Para além da descompressão, permitem fazer uma lista detalhada (nome, tamanho original, tamanho comprimido, taxa de compressão) dos ficheiros que se encontram dentro do ficheiro comprimido e permitem extrair apenas um ficheiro à escolha, opção bastante útil para desenvolver apenas a conversão/descompressão a pedido do utilizador.

Relativamente à conversão de documentos de texto para PDF, não existe nenhum programa para Linux para realizar essa conversão. Foi necessário, então, arranjar uma forma alternativa de converter esses ficheiros. Para a conversão de documentos de texto no formato Microsoft Office Word (.doc e .docx) foi utilizada uma macro do MSWord. Esta macro, através de uma rotina que selecciona todo o conteúdo do documento, cria um novo documento PDF para onde é copiado todo o conteúdo do documento a ser convertido, e encerra a aplicação MSWord.

Para utilizar esta macro foi necessário recorrer à instalação do Wine, que permite correr aplicações Windows em sistemas Linux, e procedeu-se a instalação do Microsoft Office. No entanto para correr a aplicação MSWord é necessário um ambiente gráfico, algo que o sistema operativo do servidor não possui. Foi necessário então recorrer a um *framebuffer* (Xvfb) que permite ter um ambiente gráfico virtual, executando as operações gráficas em memória sem nenhum *output* para o ecrã. Recorrendo ao Xvfb, foi possível executar a aplicação MSWord e a respectiva macro de conversão de documentos a partir da linha de comandos.

A conversão de documentos resulta então da execução de um script através da linha de comando, doc2pdf.sh (Anexo I), onde se indica a localização do ficheiro original e do ficheiro convertido. O script instancia um display virtual através do Xvfb, e inicia o MSWord com a respectiva macro de conversão dos documentos acima referenciada.

Converters

Para cada um dos programas pesquisados e encontrados anteriormente, foi desenvolvido um *converter* que invoca o respectivo programa. Desta forma, foram desenvolvidos os seguintes *converters*:

- Doc2pdfConverter - É um *converter* que invoca o *script* doc2pdf.sh desenvolvido para executar a conversão de documentos de texto para PDF. No método *init*, é iniciada a máquina virtual para permitir essa conversão, sendo que a mesma termina quando o método *shutdown* for invocado. Como usa um conversor 1-1, apenas

converte o documento de texto e coloca o ficheiro que resulta da conversão na cache de ficheiros. O formato final (especificado no parâmetro converter.finalFormat) é sempre igual a pdf;

- FFmpegConverter - *Converter* que utiliza o software FFmpeg para executar a conversão de ficheiro de áudio para os formatos que o utilizador desejar. Utiliza portanto um programa que executa conversões 1-1, sendo o ficheiro original convertido e colocado na cache de ficheiros. O formato final do ficheiro é o formato especificado pelo utilizador no parâmetro converter.finalFormat que é passado por URL para o servidor e, posteriormente, para este *converter*. Como usa um programa que permite inúmeras opções para o utilizador (definir *bitrates*, frequências, etc.), essas opções podem ser passadas por parâmetro URL (parâmetro converter.args) sendo posteriormente usadas quando o programa/conversor for invocado. O formato final (especificado no parâmetro converter.finalFormat) é o formato áudio que o utilizador desejar;

- HandBrakeConverter - *Converter* que utiliza o software HandBrake para executar a conversão de ficheiros de vídeo para os formatos que o utilizador desejar. O HandBrake também é um programa que permite conversões 1-1, e, por isso, o ficheiro original é convertido e colocado na cache de ficheiro do servidor. O formato final também é especificado num parâmetro URL (parâmetro converter.finalFormat) passado para o servidor. Assim como o FFmpeg, permite várias opções (definir *presets*, optimizações), sendo essas opções passadas como parâmetro URL (parâmetro converter.args) e usadas na conversão. O formato final (especificado no parâmetro converter.finalFormat) é o formato vídeo que o utilizador desejar;

- ImageMagickConverter - É um *Converter* que usa o programa ImageMagick para realizar conversões de imagens. O formato final é o especificado no parâmetro URL converter.finalFormat. Para além disto, permite o uso de opções para a conversão, opções que são também passadas como parâmetro URL (parâmetro converter.args). O formato final (especificado no parâmetro converter.finalFormat) é o formato de imagem que o utilizador desejar;

- MEncoderConverter -*Converter* que utiliza o programa MEncoder para realizar conversões de ficheiros de vídeo. O formato final (especificado no parâmetro converter.finalFormat) é o formato vídeo que o utilizador desejar;

- TarConverter - É um *converter* em tudo diferente aos outros apresentados anteriormente. Isto porque utiliza um programa, Tar, para

converter/descomprimir ficheiros comprimidos no formato tar ou rar, sendo, portanto, uma conversão do tipo 1-N. Devido a este facto, o *converter* tem que ser capaz de criar um ficheiro HTML com *links* para os conteúdos do ficheiro comprimido. Existiam duas formas de criar esta lista. A primeira passaria por usar o programa Tar para descomprimir logo o ficheiro comprimido todo e, posteriormente, construir um HTML com *links* para os ficheiros extraídos. A segunda forma passaria por usar uma opção do programa Tar para obter a lista com todos os dados que o ficheiro comprimido contem, sendo o ficheiro HTML construído usando essa lista. Os ficheiros seriam então extraídos a pedido do utilizador, quando este seleccionasse um dos *links* da página HTML (o programa Tar permite apenas extrair um ficheiro do ficheiro comprimido). A segunda opção é uma versão optimizada da primeira, pois só extrai um ficheiro se o utilizador assim o desejar, e, portanto, foi a opção implementada para este tipo de conversão. O formato final (especificado no parâmetro URL converter.finalFormat) é sempre o formato html;

- ZipConverter - É um *converter* em tudo idêntico ao TarConverter, apenas usando um programa diferente, que neste caso é o Zip, para ficheiros comprimidos. Apenas converte ficheiros do formato zip. O formato final (especificado no parâmetro URL converter.finalFormat) é sempre o formato html.

Handlers

Relativamente aos *handlers*, apenas foram criados dois para responder às necessidades de obter o ficheiro original a ser convertidos e suas informações. Os *handlers* criados foram os seguintes:

- HTTPHandler - É um *handler* que permite obter ficheiros e suas informações de todos os servidores, weebox incluído. Para tal apenas é necessário fornecer o *link* do ficheiro como parâmetro URL (parâmetro handler.url) passado ao servidor de migrações. Esse parâmetro é passado do servidor para o *handler*, permitindo ao mesmo saber onde se encontra o ficheiro. O URL do ficheiro pode ser usado para obter o ficheiro, o nome ou tipo do ficheiro e verificar se o utilizador possui permissões sobre o mesmo;

- weeboxHandler - É um *handler* que permite apenas obter ficheiros que estejam no weebox, sendo os seus métodos optimizados para obter dados sobre o ficheiro da forma mais rápida e utilizando

o menor tráfego de dados possível. Para usar este *handler*, é necessário também passar como parâmetro URL para o servidor, o identificador do documento ao qual o ficheiro pertence (usando o parâmetro handler.bundleID) e o identificador do ficheiro (usando o parâmetro handler.fileID). Estes parâmetros são passados do servidor para o *handler*, para que este saiba qual o ficheiro a que deve aceder e do qual deve obter informações.

Como podemos ver através desta pequena descrição, podemos concluir que o HTTPHandler é um *handler* mais abrangente que o weeboxHandler, pois este último só permite aceder a ficheiros do weebox. No entanto é muito mais lento a obter dados (nome, tipo de ficheiro, autorização) sobre os ficheiros que o weeboxHandler, pois os métodos deste são optimizados para aceder ao weebox. Se o utilizador tiver um servidor de ficheiro específico ao qual queira aceder, pode criar um novo *handler* cujos métodos sejam optimizados e específicos, ou então utilizar o HTTPHandler.

Migrator

O *servlet* Migrator vai ser responsável por receber os pedidos do utilizador, tratá-los e responder a esses pedidos. Existem dois tipos de pedidos: o pedido de conversão, onde o utilizador especifica como parâmetros no URL do pedido o *handler* e seus argumentos e o *converter* e seus argumentos; ou então o pedido para obter um ficheiro, onde o utilizador especifica no URL qual o identificador do pedido e o nome do ficheiro que pretende obter (podemos encontrar exemplos deste URL's nos *links* dos ficheiros HTML resultantes da conversão 1-N). O comportamento do *servlet* depende do tipo de pedido que o mesmo recebe. Podemos observar um esquema global do comportamento do *servlet* na Figura 2.27.

Numa primeira fase o *servlet* verifica qual é o tipo de pedido que recebe, verificando se no URL usado foi passado o identificador do pedido e o nome do ficheiro. Se não for passado, então o pedido é de conversão (exemplos nas Figuras 2.28 e 2.29). Neste caso o que o *servlet* faz é obter o *handler*, o *converter* e seus respectivos argumentos, que são passados como parâmetros no URL. Obtendo esses dados é gerado o identificador do pedido, que corresponde à junção do identificador do *handler* com o identificador do *converter* e posterior codificação, que são únicos e gerados tendo em conta os argumentos do *handler* e do *converter* respectivamente. Por exemplo, o pedido ao servidor da Figura 2.29 gera um identificador (CB881CE87D80AC5825C94A30DE841D80). Se posteriormente for feito outro pedido ao servidor exactamente igual ao primeiro

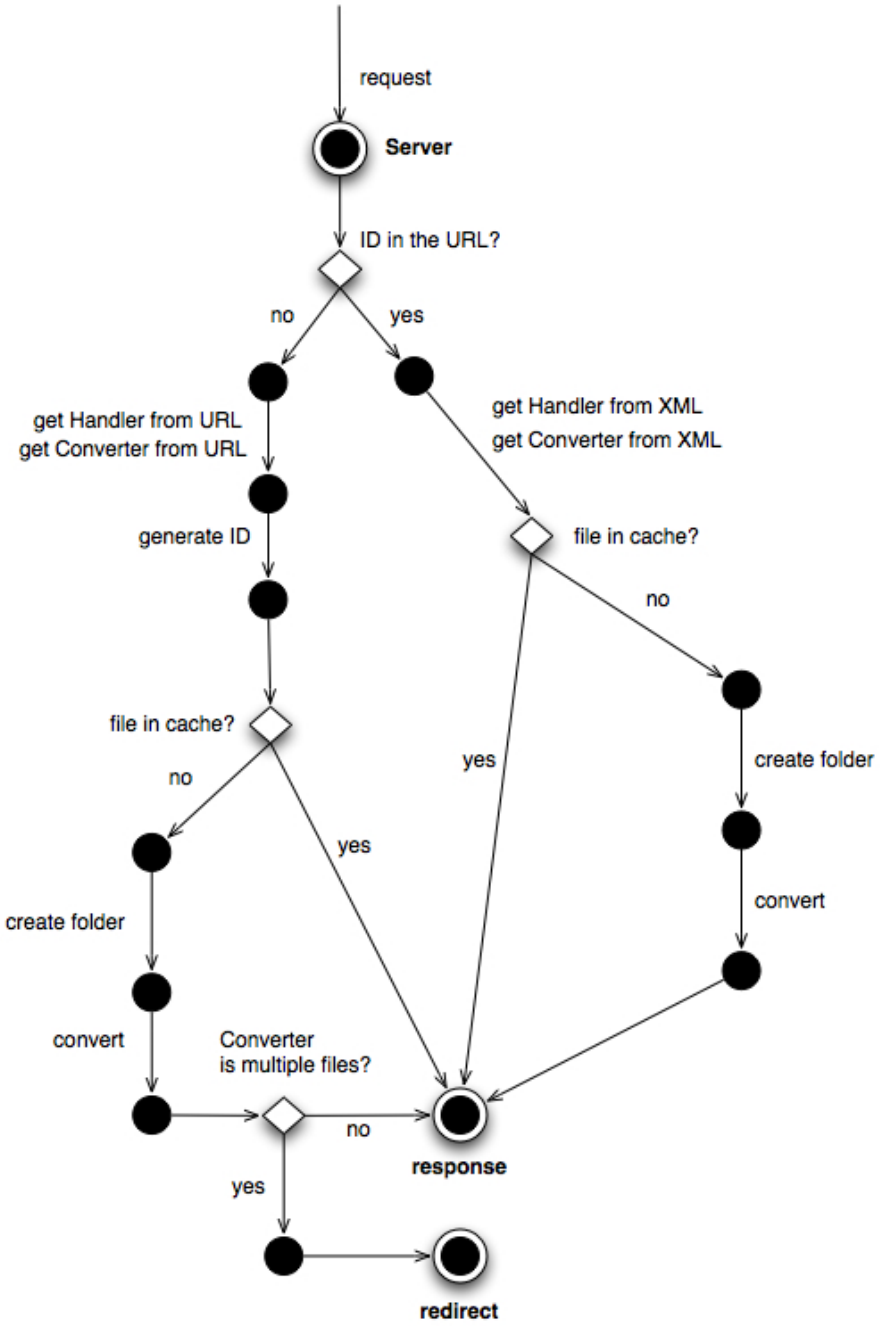

Figura 2.27: Diagrama do comportamento do servlet Migrator

ou apenas trocando a ordem dos parâmetros do URL, o identificador gerado vai ser o mesmo. Portanto, pedidos com igual *handler*, *converter* e respectivos argumentos (a ordem dos parâmetros correspondentes não in-

teressa) possuem identificadores iguais. Os identificadores são usados na gestão da cache. Usando o *handler* verifica-se se o utilizador tem autorização sobre o ficheiro que pretende converter, onde em caso afirmativo é lido o seu nome e o processo continua. Com esse nome, o formato desejado para a conversão (passado como parâmetro URL através do parâmetro converter.finalFormat) e o identificador do pedido é possível verificar se a conversão já tinha sido efectuada, ou seja, verificar se o ficheiro convertido se encontra na cache de ficheiros do servidor. Se o ficheiro convertido se encontrar na cache é enviado ao utilizador, terminando o processo. Caso contrário, é criada a pasta onde o ficheiro convertido vai ficar e é invocado o método de conversão do *converter* especificado pelo utilizador, sendo o ficheiro convertido colocado na respectiva pasta. Por fim, é verificado se o *converter* origina múltiplos ficheiros, criando o ficheiro HTML. Em caso positivo é feito o redireccionamento do utilizador para o ficheiro HTML criado e todos os dados do pedido são guardados num ficheiro XML. Em caso negativo é devolvido o ficheiro convertido.

Na Figura 2.28 temos um exemplo de um pedido ao servidor que da primeira vez que é executado, vai buscar o ficheiro que se encontra no weebox (daí usar o *handler* WeeboxHandler), tenta a sua conversão para PNG (conversão 1-1) e devolve em caso de sucesso o ficheiro convertido.

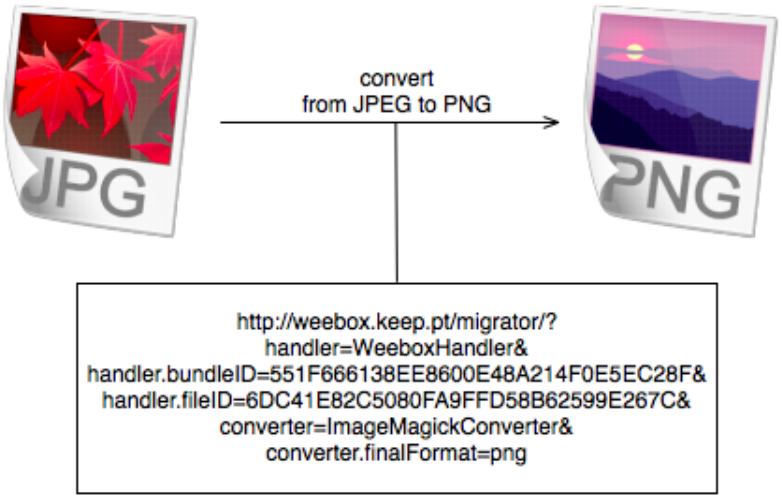

Figura 2.28: Pedido ao servidor sem identificador e nome do ficheiro (conversão 1-1)

Na Figura 2.29 temos um exemplo de um pedido que da primeira vez que é executado, vai buscar o ficheiro original (que se encontra no weebox e corresponde a um ficheiro comprimido), converte-o criando um ficheiro HTML com *links* para os ficheiros resultantes (conversão 1-N) e é feito

o *redirect* para esse ficheiro HTML.

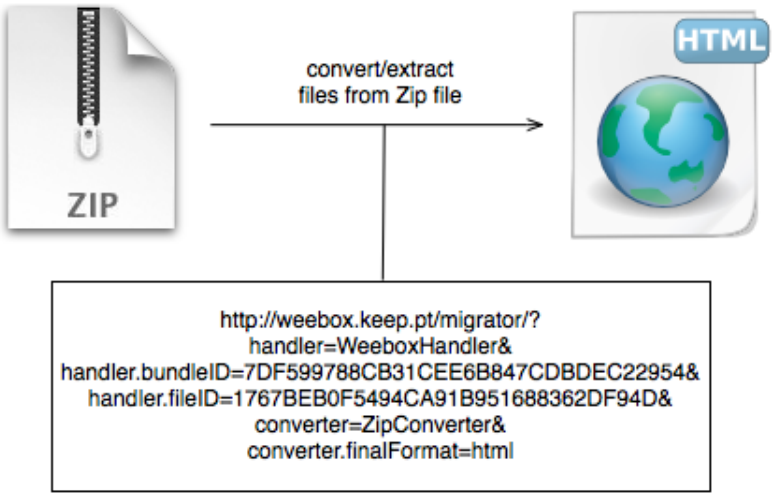

Figura 2.29: Pedido ao servidor sem identificador e nome do ficheiro (conversão 1-N)

Para os outros pedidos, onde o identificador e o nome do ficheiro que o utilizador pretende são passados no URL (exemplo na Figura 2.30), o servlet vai ao ficheiro XML buscar todos os dados que correspondem ao identificador do pedido, nomeadamente o *handler*, o *converter* e respectivos argumentos que foram usados no anterior pedido de conversão. O *servlet* utiliza esses dados para verificar se o utilizador tem permissões sobre o ficheiro original, apenas continuando o processo se a autorização o permitir. Com o identificador e com o nome do ficheiro, vai-se verificar se o ficheiro está em cache. Em caso afirmativo, o ficheiro pedido é devolvido e o processo termina. Em caso negativo, é criada a pasta onde o ficheiro convertido vai ficar e é o invocado o método de conversão do *converter* lido do ficheiro XML, sendo o ficheiro obtido dessa conversão colocado na pasta criada. Posteriormente, o ficheiro convertido é devolvido ao utilizador e o processo termina. Esta conversão acontece, por exemplo, com os *converters* Zip e Tar, onde a conversão só é feita a pedido do utilizador quando este carrega num *link* do ficheiro HTML.

Na Figura 2.30 temos um exemplo de um pedido ao servidor onde é passado o identificador do pedido e o ficheiro pretendido, ou seja, corresponde a um dos *links* que aparecem nos ficheiros HTML que funcionam como ponto de entrada da conversão 1-N. No fim do processo e em caso de sucesso, é enviado o ficheiro pretendido ao utilizador.

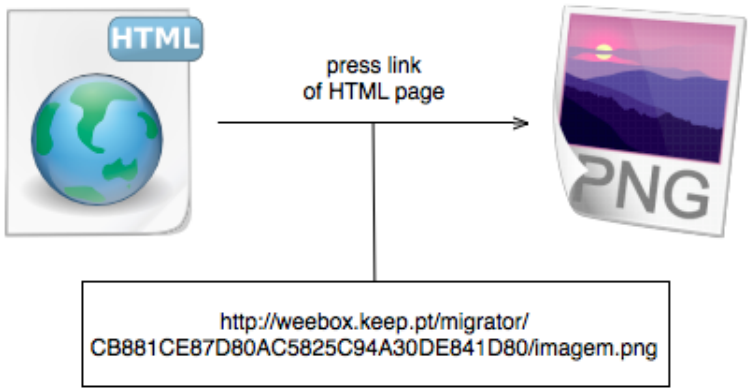

Figura 2.30: Pedido ao servidor com identificador e nome do ficheiro

A descrição do comportamento do *servlet* permitiu identificar que o mesmo resolve os problemas de autorização sobre os ficheiros, pois antes de realizar qualquer processo de conversão ou de devolver qualquer ficheiro anteriormente convertido verifica-se se o utilizador possui autorização sobre o ficheiro original. A autenticação no servidor, para ser posteriormente usada para aceder aos ficheiros, é feita por HTTP Basic Authentication [5], que é um método que permite providenciar credenciais quando se está a fazer um pedido [19]. Outro dos problemas resolvidos, foi o problema com a existência de diferentes tipos de conversores (1-1 e 1-N). Relativamente aos conversores N-N vão ser implementados numa das próximas versões do servidor, pois são conversores menos comuns e que muitas das vezes podem ser implementados a partir de várias conversões de 1-1.

Após a realização de alguns testes verificou-se que pedidos iguais corriam de igual forma, o que de certa forma constitui uma falha, pois assim estão, por exemplo, dois pedidos a aceder ao mesmo ficheiro, ou então a realizar o mesmo processo de conversão. A solução implementada para resolver este problema foi implementar uma lista de pedidos. Essa lista vai conter os identificadores dos pedidos (pedidos iguais têm identificadores iguais) que estão a correr. O que o *servlet* faz antes de devolver um ficheiro ou executar uma conversão é verificar se o identificador do pedido está nessa lista, ou seja, se já existe um pedido idêntico a correr. Em caso afirmativo, o pedido é submetido a um período de espera até que o outro acabe. Em caso negativo ou quando o outro acabar, o identificador do pedido é colocado na lista e o processo de retorno do ficheiro convertido ou de conversão é executado. No fim, desse processo

[5]Existem outros tipos de autenticação HTTP cujo suporte não foi implementado, mas o HTTP Basic Authentication serve como prova que é possível adicionar outras funcionalidades se necessário

o identificador é removido da lista, avisando os outros pedidos que se encontram à espera que já terminou. Isto permite, que pedidos com o mesmo identificador do pedido que terminou possam avançar.

Devido à necessidade de fazer *streaming* de vídeo, nomeadamente para iPhone, foi implementado um processo, designado por Byte Serving, quando o servidor se encontra a devolver o ficheiro convertido. O Byte Serving é um processo que permite enviar apenas partes da mensagem HTTP/1.1 do servidor para o cliente [18, 20]. Os clientes utilizam este processo, por exemplo, quando apenas parte dum ficheiro foi recebida, ou então quando precisam de apenas de determinada parte do ficheiro, sendo assim um processo de optimização da largura de banda. Um servidor que implementa este processo, permite que os clientes peçam apenas parte do recurso. Este processo também é conhecido por Byte Range Serving [15].

2.3.3 Utilização

Após ter o servidor de migrações a funcionar e disponível para uso, a aplicação móvel foi ligeiramente alterada, de forma a usar o servidor de migrações para converter ficheiros cujos formatos não são suportados pelo iPhone de forma nativa, e posteriormente visualizar os ficheiros resultantes dessas conversões.

Como o servidor de migrações só vai ser usado quando o utilizador tentar visualizar um ficheiro, então apenas o comportamento do ecrã de visualização de ficheiros vai ser modificado. O processo de visualização de um ficheiro pode ser observado na Figura 2.31.

Numa primeira fase, os ficheiros a serem visualizados são ficheiros guardados no weebox. Portanto é relativamente fácil saber qual o tipo do ficheiro em questão, e consequentemente, saber se o ficheiro é suportado nativamente pelo iPhone. No entanto, existe um pequeno problema com os ficheiros de vídeo e de áudio. O tipo do ficheiro de vídeo pode ser suportado pelo iPhone, mas a resolução do vídeo pode ser superior à suportada pelo dispositivo. Relativamente aos ficheiros de áudio, existe um problema semelhante, pois o tipo de ficheiro pode ser suportado, mas o *bitrate* do ficheiro pode não ser. Assim foi tomada a decisão de todos os ficheiros de vídeo e de áudio serem convertidos. Relativamente aos documentos de texto no formato docx, também foi tomada a decisão de convertê-los directamente para PDF. O problema dos documentos de texto é que as suas visualizações no iPhone divergem da versão original e a conversão directa para PDF é uma forma de evitar esse problema. Relativamente aos restantes ficheiros se não forem suportados também po-

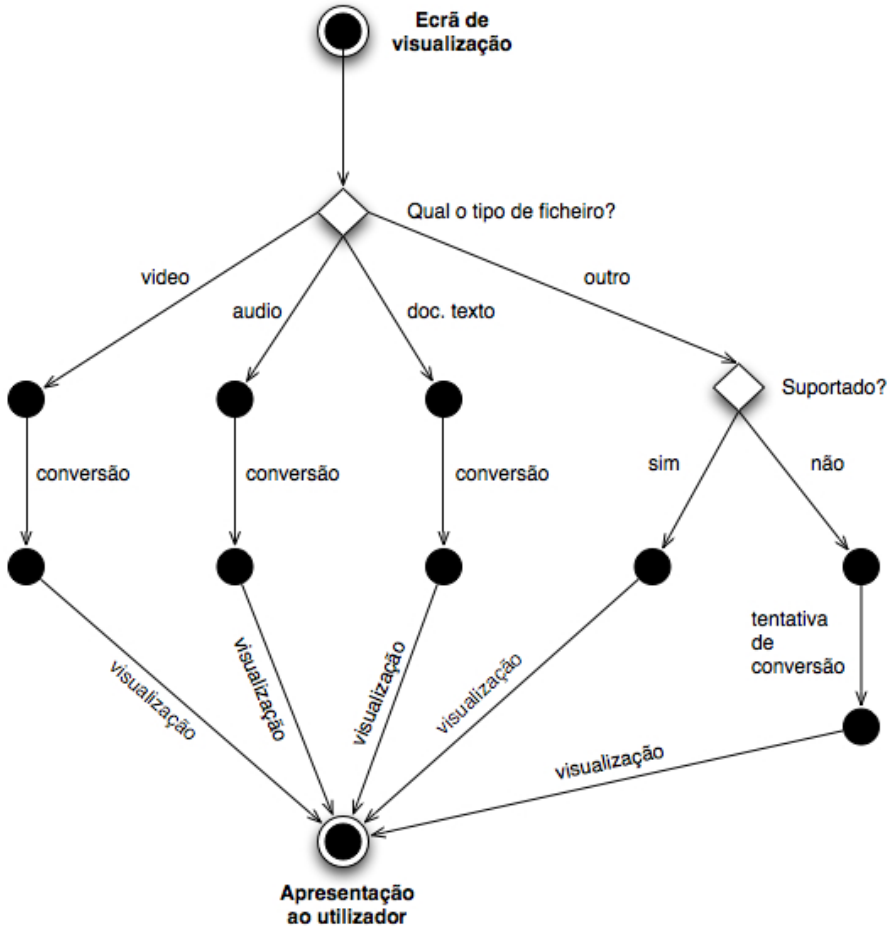

Figura 2.31: Processo de visualização de ficheiro

dem ser convertidos (como, por exemplo, alguns tipos de imagem muito específicos, ficheiros comprimidos, entre outros).

Muitos dos ficheiros a serem apresentados são ficheiros HTML (quer sejam directamente postos no weebox, quer sejam resultantes da conversão de ficheiros comprimidos) e estes podem possuir *links* para outros ficheiros. No entanto, os ficheiros correspondentes a esses *links* podem não ser suportados pelo dispositivo, e o ideal seria utilizar o servidor de migrações quando o utilizador carregar num dos *links*, para converter o ficheiro correspondente para um formato suportado e só depois mostrar o resultado da conversão ao utilizador. Porém, desenvolver este processo implicaria muitas mudanças no comportamento da aplicação móvel, pelo que foi decidido a sua não implementação nesta versão da aplicaçaõ, passando para trabalho futuro.

Sendo assim, os únicos ficheiros que podem ser convertidos são os ficheiros guardados no weebox, e devido a este facto foi usado como *handler* o weeboxHandler, pois o mesmo é específico e optimizado para interagir com o weebox.

Relativamente aos *converters*, para a conversão de vídeos foi escolhido o HandBrakeConverter. Esta escolha foi baseada nas opções que o conversor usado por este *converter* possui para converter vídeos para o iPhone (–preset="iPhone & iPod Touch") e para optimizar esses mesmos vídeos para HTTP *streaming* (-O). Todos os vídeos são convertidos para o formato mp4, formato que é nativamente suportado pelo iPhone.

Para a conversão de ficheiros de áudio foi utilizado o FFmpegConverter, pois permite definir a frequência que o ficheiro convertido vai possuir. No caso do iPhone, a frequência ideal é de 128k e para tal basta usar a opção -ab do conversor. Os ficheiros de áudio são convertidos para o formato mp3, que é um formato suportado nativamente pelo iPhone.

Como já foi dito nesta secção, os documentos de texto são convertidos para PDF, sendo para tal usado o *converter* Doc2pdfConverter que permite converter documentos de texto para PDF.

Caso apareça uma imagem não suportada, será usado o ImageMagickConverter, sendo as imagens convertidas para JPEG. Nesse formato as imagens já podem ser visualizadas pelo iPhone.

Por fim faltam os ficheiros comprimidos. Esses ficheiros vão ser convertidos usando os *converters* ZipConverter ou TarConverter, de acordo com o tipo do ficheiro comprimido. Como já foi referido, da conversão dos ficheiros comprimidos resulta um ficheiro HTML que possui *links* para os ficheiros e pastas que se encontram dentro do ficheiro comprimido. Como se trata de um ficheiro HTML, o mesmo pode ser logo visualizado no iPhone, pois trata-se de um formato suportado.

Capítulo 3

Testes

Após o termino do desenvolvimento da aplicação weebox para o iPhone, a aplicação passou por um extenso processo de testes.

Numa primeira fase, foram feitos testes usando o Instruments, que é uma ferramenta que permite avaliar a performance da aplicação móvel e recolher dados como o uso de disco, de memória ou processador em tempo real (quer seja usando o simulador ou um iPhone que esteja ligado ao computador). Neste caso de estudo específico, os testes realizados foram essencialmente relativos a memória, prevenindo o acontecimento de *memory leaks* (vazamentos de memória), que ocorrem quando a aplicação é responsável pela alocação de uma porção de memória durante a execução de uma determinada operação, mas no final da execução dessa operação ou quando a mesma não for necessária, a memória não é libertada. Os vazamentos de memória podem provocar falhas no sistema, e ocorrem porque o iOS não possui *garbage collector*, que é uma forma de gestão automática de memória [22].

A realização de testes no Instruments permitiu detectar as seguintes situações:

- *Memory leak* de 128 bytes - Todos os testes realizados no Instruments mostraram um vazamento de 128 bytes, que acontece quando o teclado aparece da primeira vez. No entanto, é um vazamento que só ocorre quando se corre a aplicação no simulador, e que já foi comunicado à Apple, esperando assim que não ocorra na próxima versão do simulador;

- *Memory leaks* na UIWebView - Ocorrem vazamentos de memória de forma muito ocasional quando o utilizador se encontra a visualizar um ficheiro, devido ao componente de interface usado para tal (UIWebView). Esse componente é normalmente usado para mos-

trar conteúdos Web, e como tal utiliza uma cache. O problema é que alguns dos dados que são guardados na cache não estão a ser libertados, provocando assim vazamentos de memória. É um problema muito comum em aplicações desenvolvidas para iPhone e que usam esse componente de interface, tendo já sido feitos inúmeros comunicados desse erro à Apple por vários programadores.

Como estas situações eram comuns noutras aplicações já disponíveis, a aplicação encontrava-se pronta para ser submetida para a App Store, onde vai passar por um rigoroso processo de avaliação e de testes, apenas ficando disponível aos utilizadores se a mesma passar com distinção nesse processo. A aplicação foi submetida no dia 8 de Setembro de 2010, tendo sido aceite no dia 24 do mesmo mês, como podemos verificar na Tabela 3. Pelo meio deste processo a aplicação foi rejeitada devido à falta de credenciais para teste, apesar da existência da operação de registo na aplicação móvel. É compreensível que quem realizou o teste por parte da Apple não tenha utilizado o processo de registo, pois dessa forma não ficaram dados seus registados no weebox.

Tabela 3.1: Dados da submissão da aplicação

Date	User	Status
September 06, 2010 08:05	....	Prepare For Upload
September 06, 2010 08:40	....	Waiting For Upload
September 08, 2010 02:25	Apple	Upload Received
September 08, 2010 02:26	Apple	Waiting For Review
September 12, 2010 16:37	Apple	In Review
September 20, 2010 10:45	Apple	Rejected
September 21, 2010 08:15	Apple	Waiting For Review
September 21, 2010 08:15	Apple	In Review
September 24, 2010 15:44	Apple	Processing For App Store
September 24, 2010 15:48	Apple	Ready For Sale

Enquanto este processo de aceitação na App Store decorria, foram realizados testes de usabilidade à aplicação para iPhone. Para a realização destes testes foram escolhidas um conjunto de pessoas de várias áreas profissionais, de forma a tornarem os testes mais abrangentes. Antes de realizarem estes testes, os inquiridos foram convidados a fazerem testes de usabilidade à aplicação Web do weebox, permitindo-lhes assim adquirir conhecimentos sobre o weebox, sobre as suas funcionalidades e sobre os comportamentos de interface. Isto foi de extrema importância

porque a maior parte dos inquiridos não conhecia o weebox, e era contra-natura os mesmos estarem a testar a usabilidade da aplicação móvel, sem conhecimento prévio das funcionalidades da aplicação Web. Os testes de usabilidade, quer à aplicação Web, quer à aplicação móvel, consistem numa série de passos que os inquiridos terão de cumprir, sendo medido o tempo de execução de cada passo e avaliado se o inquirido realizou o passo da forma desejada. Para além disso é feito o levantamento de todos os erros e problemas que acontecem. O teste de usabilidade da aplicação Web é mais complexo que o teste de usabilidade da aplicação móvel, devido sobretudo à maior quantidade de funcionalidades implementadas. No Anexo K, podem ser consultados os resultados dos testes de usabilidade da aplicação Web, bem como os erros e problemas encontrados. Esses erros e problemas encontrados são importantes, mas irrelevantes para este caso de estudo, pois os mesmos não influenciam a interface e o comportamento da aplicação móvel. A Figura 3.1, apresenta a descrição, a percentagem de sucesso, o tempo médio e as observações de cada um dos passos dos testes de usabilidade para a aplicação móvel.

Nº	Tarefa	Sucesso	Tempo	Observações
1	Entre na aplicação com a sua conta	57%	0m 41s	Muitos precisaram de ajudar para encontrar o botão de login "Ir" no teclado e perceber o seu uso. Problema #1.
2	Veja "Todos os documentos"	100%	0m 14s	
3	Encontre um documento que possui "backgrounds" para o "ubuntu"	100%	0m 37s	Um não usou a pesquisa, outro pensou que o botão de pesquisa avançada era o método a seguir, mas conseguiu rapidamente encontrar o caminho.
4	Consulte todos os ficheiros contidos nesse documento	100%	0m 44s	Um não viu as setas.
5	Volte para a pesquisa e filtre por fotos	100%	0m 46s	2 tentaram fazer uma pesq. básica por "fotos", outro foi à pesq. avançada, mas encontraram rapidamente o caminho certo, se bem que um não gostou muito. Algum tempo para perceber também que depois de marcar fotos é preciso escolher um filtro.
6	Utilize a pesquisa avançada e pesquise os documentos criados a partir de 22 de Agosto e até 5 MB de tamanho	100%	1m 53s	Alguns problemas entre os filtros de tipo de documento e a pesq. avançada - Problema #2. Um demorou algum tempo a encontrar o ícone da pesq. avançada, tentando encontra-la na página inicial. Foi detectado o Erro #1.
7	Procure todos os vídeos existentes	100%	0m 23s	
8	Utilize a pesquisa avançada para encontrar filmes de ficção cientifica	100%	0m 36s	Quase todos escolheram filmes de "Ficção" e "Ficção cientifica" - Problema #3
9	Saia da aplicação	100%	0m 7s	

Figura 3.1: Dados dos testes de usabilidade (iPhone)

Como podemos ver, no decorrer dos diversos testes de usabilidade foram

levantados os problemas e erros encontrados na aplicação móvel, que podem ser consultados na Figura 3.2 e 3.3, respectivamente. O único passo que não teve 100% de eficácia foi o passo de autenticação. O problema com esse passo deve-se sobretudo à falta de hábito dos inquiridos na utilização de aplicações para iPhone, pois são muitas as aplicações que usam uma interface parecida para autenticação do usuário.

#	Descrição	Solução
1	No iPhone, grande dificuldade em compreender que "Ir" é o botão de autenticação. E como o botão está no teclado, se este se esconder torna-se muito complicado fazer login.	Adicionar um botão de Login no painel. Esconder o teclado ao carregar no lógotipo do weebox.
2	No iPhone, torna-se confuso o estado do filtro de tipo de documento em conjunto com a pesq. avançada.	Na pesq. avançada mostrar o filtro de tipos de documento activos e permitir altera-lo.
3	No iPhone, no editor de vocabulários controlados existe a tendência ao seleccionar um item na árvore de seleccionar também os ascendentes.	?

Figura 3.2: Problemas encontrados nos testes de usabilidade (iPhone)

#	Descrição
1	No iphone, a pesquisa avançada por tamanho pareceu não funcionar num dos testes

Figura 3.3: Erros encontrados nos testes de usabilidade (iPhone)

Para cada problema ou erro, é apresentada uma solução. Como podemos observar, o último dos problemas de usabilidade encontrados não tem uma solução descrita. O problema tem sobretudo a ver com o funcionamento da interface móvel, que se comporta de maneira diferente da interface Web, provocando assim confusão ao utilizador. Neste caso, a solução deste problema passa sobretudo pela adaptação do utilizador a este comportamento. Os erros e problemas encontrados nos diversos testes de usabilidade realizados, são importantes para melhorar as futuras versões da aplicação móvel.

Capítulo 4

Conclusão

Neste capítulo apresentam-se os principais guias derivados da análise do caso de estudo e as notas finais desta dissertação, bem como possíveis trabalhos futuros.

4.1 Guias

O principal objectivo desta dissertação era a elaboração de guias para a adaptação de aplicações Web ou Desktop em aplicações móveis. A análise do caso de estudo permitiu derivar guias através da generalização do problema da adaptação de interfaces, os quais serão apresentas nesta secção. Estes podem ser usados para desenvolver futuras aplicações móveis em diferentes plataformas.

Divisão/construção de interfaces

Os ecrãs dos dispositivos, por norma, apresentam tamanhos e resoluções inferiores aos computadores e, consequentemente, as aplicações que correm nesses dispositivos não podem mostrar tanta informação numa interface como as aplicações desenvolvidas para os computadores. Desse modo, a observação dos comportamentos das interfaces Web ou Desktop, bem como a divisão em secções e estudo das mesmas, permite o desenho das diversas interfaces móveis.

A aplicação móvel deve tentar sempre implementar os mesmos comportamentos da aplicação a ser adaptada, de forma a não provocar um impacto demasiado grande nos utilizadores que transitam dessa aplicação para a aplicação móvel. Durante processo de adaptação, as interfaces Desktop ou Web podem ser divididas em diversos ecrãs da aplicação

móvel, o que pode provocar alterações de comportamento na aplicação móvel. No entanto, apesar das alterações de comportamento os resultados devem ser os mesmos que os obtidos na aplicação a ser adaptada.

Se na aplicação Desktop ou Web houver botões ou *links* que abram novos painéis, então esses novos painéis, por norma, correspondem a novas interfaces na aplicação móvel, que podem ser apresentadas noutros ecrãs ou em modo modal, de acordo com a complexidade do painel a ser adaptado.

As aplicações Desktop ou Web permitem a inserção de diferentes tipos de dados, desde texto, valores, datas, entre outros (algumas permitem também seleccionar intervalos de valores e datas). No desenvolvimento da aplicação móvel pode-se encontrar problemas com a inserção de determinados tipos de dados, pois não existem componentes de interface nativos que suportem essa inserção. Por norma tipos de dados não suportados nativamente implicam o desenvolvimento de novas interfaces.

Teclado

No desenho das interfaces é necessário ter em conta o teclado virtual, pois o espaço por ele ocupado pode impedir a visualização de conteúdos importantes na aplicação móvel. É preciso ter em conta o espaço ocupado e onde vai aparecer o teclado, isto em aplicações onde a introdução de dados é um factor importante e no caso do dispositivo não possui teclado físico.

Componentes nativos vs componentes específicos

Quando se desenvolve uma aplicação móvel há sempre o dilema entre usar componentes de interface nativos da plataforma móvel ou usar componentes visuais específicos da aplicação que se pretende adaptar. Os componentes nativos trazem robustez, performance e rapidez de implementação à aplicação, mantendo também uma coerência com outras aplicações para a plataforma escolhida. Por outro lado, os componentes específicos usados na aplicação que se pretende adaptar trazem coerência de *design* e de funcionalidades entre as interfaces Desktop ou Web e as interfaces móveis. Deve-se usar sempre que possível componentes de interfaces nativos da plataforma móvel, a menos que o impacto visual e a diferença de comportamentos entre as interfaces a serem adaptadas e as interfaces móveis sejam notórias aos utilizadores, tornando a aplicação móvel bastante diferente. Nesse caso, usam-se então os componentes visuais específicos da aplicação origem mas sempre pensando na troca de

valores que isso implica.

Idioma

As aplicações Desktop ou Web, normalmente possuem a possibilidade de escolha do idioma das diversas interfaces que serão apresentadas ao utilizador. Nas aplicações móveis o idioma escolhido é por norma o utilizado pelo dispositivo, que pode ser alterado nas configurações do próprio dispositivo. A alteração do idioma do dispositivo provoca a alteração das labels e mensagens apresentadas ao utilizador na aplicação móvel.

Replicação de componentes

As diversas interfaces Desktop e Web podem replicar componentes de interfaces (no caso de estudo era replicado a selecção dos tipos de documentos na pesquisa avançada). A replicação desses componentes pode provocar um aumento de complexidade na adaptação das interfaces para interfaces móveis. Na aplicação desenvolvida foi tomada a decisão de não replicar a selecção dos tipos de documentos na pesquisa avançada, o que se veio a revelar uma má decisão, pois trouxe problemas de usabilidade (encontrados nos testes de usabilidade). A replicação de componentes deve ser bem avaliada e testada, verificando se traz vantagens ou desvantagens para a aplicação móvel.

Complexidade de estados e modelo de dados

A adaptação de interfaces Desktop ou Web em interfaces móveis, pode provocar o aumento de complexidade em termos de estados da aplicação móvel em relação à aplicação origem. O aumento de complexidade deve-se ao aumento de interfaces, pois uma interface Desktop ou Web pode ser adaptado em uma ou mais interfaces móveis. Para além disso, pode ser necessário implementar classes para guardar os dados do servidor, ou seja, criar um modelo de dados adicional. Pode ser também necessário criar classes para tratar dados do servidor (como, por exemplo, *parsers* para respostas/mensagens XML), caso a aplicação móvel possua interacção com um ou mais servidores.

Conteúdos multimédia

A adaptação de aplicações ricas em conteúdos multimédia em aplicações móveis pode provocar problemas de disseminação desses conteúdos, pois

os formatos dos conteúdos podem não ser suportados pelo dispositivo. Assim é necessário pensar em alternativas, quer seja desenvolvendo formas de visualizar conteúdos com formatos não suportados, quer seja convertendo os conteúdos para formatos suportados e posteriormente proceder à sua visualização. Como foi visto com esta dissertação, um servidor de conversões genérico (suporte a vários formatos) é uma solução viável para o problema da disseminação de conteúdos. Para além disso, é indiferente a plataforma que está a servir, ou seja, um ficheiro pode ser convertido para o formato desejado e para qualquer dispositivo. Isso é conseguido através de um sistema de *plugins*, que permite adicionar novos conversores ao servidor, permitindo assim obter as conversões desejadas pelo o utilizador. O sistema de *plugins* também permite adicionar formas de interacção com os outros servidores de ficheiros para obter ficheiros ou dados dos mesmos através de métodos optizimidos de acordo com o servidor em questão. A leitura destes *plugins* deve ser em tempo de execução, ou seja, sem haver necessidade de reniciar o sistema.

4.2 Notas finais

As aplicações móveis fazem parte do quotidiano de muitas pessoas, quer seja para trabalho ou apenas para lazer. Estas podem ser aplicações que resultam da adaptação de outras anteriormente desenvolvidas ou apenas focadas para plataformas móveis. O problema da adaptação de aplicações para aplicações móveis prende-se com a inexistência de guias para fazer essa adaptação, o que esta dissertação tenta colmatar. O objectivo principal desta dissertação passa pela apresentação de um conjunto de guias que permita desenvolver aplicações móveis a partir de aplicações já desenvolvidas. Para tal, foi feita a análise do desenvolvimento de uma aplicação para iPhone, tendo como base uma aplicação já desenvolvida (weebox), e a partir daí foram derivados guias através da generalização do problema, cumprindo assim o objectivo principal. A adaptação da aplicação já desenvolvida trouxe também o problema da disseminação de conteúdos, pois a aplicação era rica em conteúdos multimédia e muitos desses conteúdos podiam não ser suportados pelo dispositivo. Para ultrapassar tal problema foi criado um servidor de migrações genérico com um sistema de *plugins* que permite adicionar novas conversões e novas formas de interacção com outros servidores de ficheiros. É um sistema independente da plataforma que o invoca, pois permite a conversão de vários formatos para outros e, consequentemente, visualizar os mais diversos conteúdos.

O resultado desta dissertação, para além de um servidor de migrações ge-

nérico, foi uma aplicação weebox para iPhone. A aplicação encontra-se disponível na App Store da Apple [1], podendo o utilizador descarregá-la gratuitamente. Tal como outras aplicações, ao longo do tempo vai sofrer alterações implementando assim novas funcionalidades, sobretudo funcionalidades de edição, como edição dos metadados dos documentos, criar/arquivar/passar para rascunho/eliminar documento(s), alterar tipo de documento do(s) documento(s), entre outras. A adição destas funcionalidades vai permitir tornar a aplicação móvel ainda mais completa e parecida com a aplicação Web do weebox.

4.3 Trabalho Futuro

O trabalho futuro passa pelo melhoramento do servidor de migrações, que consiste no suporte a conversores N-N, que de momento não são suportados pelo servidor, pois são conversores menos comuns e podem ser substituídos por outros tipos de conversores (1-1 ou 1-N) na maioria dos casos. O processo para suportar estes conversores passa por receber vários *handlers* (*plugins* que permitem obter ficheiros e dados dos mesmos do weebox ou de outros servidores de ficheiros) e respectivos argumentos por parâmetros URL, em vez de um só. Os ficheiros origem serão depois convertidos pelo *converter* (*plugins* que permitem executar a conversão dos ficheiros origem), sendo o resultado dessa conversão enviado para o utilizador. Para além do não suporte a conversores N-N, existe o problema das conversões recursivas, que podem acontecer quanto existem ficheiros comprimidos dentro de um ficheiro comprimido ou quando o utilizador se encontra a navegar num HTML e tenta visualizar um ficheiro não suportado. O trabalho futuro também vai incidir sobre este problema, cuja solução pode provocar alterações ligeiras no servidor ou mesmo na aplicação móvel.

Os melhoramento apontados nesta secção não invalidam, nem modificam, os guias ou conclusões desta dissertação.

[1] http://itunes.apple.com/sg/app/weebox/id391352176?mt=8

Bibliografia

[1] AdMob. May 2010 metrics highlights. pages 1–29, 2010.

[2] all-seing interactive. Asihttprequest documentation. http://allseeing-i.com/ASIHTTPRequest/, 2010.

[3] Apple. ios overview. http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/URL_iPhone_OS_Overview/index.html#//apple_ref/doc/uid/TP40007592, 2010.

[4] Apple. Parser capabilities and architecture. http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/XMLParsing/Articles/ParserArchitecture.html, 2010.

[5] Apple. Tools for ios development. http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/URL_Tools_for_iPhone_OS_Development/index.html, 2010.

[6] Apple. View controllers programming guide for ios. http://developer.apple.com/library/ios/#featuredarticles/ViewControllerPGforiPhoneOS/Introduction/Introduction.html, 2010.

[7] V. Fernandes. Content dissemination on mobile platforms. pages 1–12, 2009.

[8] FFmpeg. Ffmpeg. http://www.ffmpeg.org/, 2010.

[9] Gartner. Gartner says worldwide mobile device sales grew 13.8 percent in second quarter of 2010, but competition drove prices down. http://www.gartner.com/it/page.jsp?id=1421013, 2010.

[10] Gentoo. Mencoder. http://en.gentoo-wiki.com/wiki/Mencoder, 2010.

[11] S. Hall and E. Anderson. Operating systems for mobile computing. *JCCSC*, pages 1–8, 2009.

[12] HandBrake. Handbrake. http://handbrake.fr/, 2010.

[13] ImageMagick. Imagemagick. http://www.imagemagick.org/script/index.php, 2010.

[14] MSDN. Model view controller. http://msdn.microsoft.com/en-us/library/ff649643.aspx, 2010.

[15] W. Polygraph. Http range request support. http://www.web-polygraph.org/docs/userman/ranges.html, 2010.

[16] SAX. Events vs. trees. http://www.saxproject.org/event.html, 2010.

[17] K. SOLUTIONS. http://www.keep.pt, 2010.

[18] W3. Http fields. http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html, 2010.

[19] Wikipedia. Basic acess authentication. http://en.wikipedia.org/w/index.php?title=Basic_access_authentication&oldid=393434180, October 2010.

[20] Wikipedia. Byte serving. http://en.wikipedia.org/w/index.php?title=Byte_serving&oldid=313400108, September 2010.

[21] Wikipedia. Ffmpeg. http://pt.wikipedia.org/w/index.php?title=FFmpeg&oldid=20467111, May 2010.

[22] Wikipedia. Garbage collector. http://en.wikipedia.org/w/index.php?title=Garbage_collection_(computer_science)&oldid=392640558, October 2010.

[23] Wikipedia. Handbrake. http://pt.wikipedia.org/w/index.php?title=HandBrake&oldid=18234352, January 2010.

[24] Wikipedia. Imagemagick. http://pt.wikipedia.org/w/index.php?title=ImageMagick&oldid=19366849, March 2010.

[25] Wikipedia. Mencoder. http://pt.wikipedia.org/w/index.php?title=MEncoder&oldid=21109013, July 2010.

[26] Wikipedia. Model view controller. http://pt.wikipedia.org/w/index.php?title=MVC&oldid=22257116, October 2010.

Apêndice A

Funcionalidades do weebox Manager

As funcionalidades deste módulo são:

- **Autenticação**
  - Autenticação do utilizador;
  - Registo de novo utilizador;
  - Recuperação da senha de utilizador.
- **Submissão**
  - Possibilidade de selecção de vários ficheiros para envio e submissão;
  - Escolha se os vários ficheiros se aglomeram num documento ou criam vários documentos independentes;
  - Painel com informação do upload dos vários documentos e ficheiros, só visível durante a transmissão dos dados;
  - Aplicação automática do tipo de documento por omissão e extracção de metadados.
- **Pesquisa**
  - Pesquisa básica (em metadados e texto integral);
  - Pesquisa avançada (escolha do tipo de documento, pesquisa em campos do tipo short text com sugestão, pesquisa em intervalos de datas, pesquisa em intervalos de tamanho de ficheiros, pesquisa por vocabulários controlados, potencial expansão para qualquer tipo de dados);

  - Filtros sobre documentos (Triagem, Os meus documentos, Os meus favoritos, Todos os documentos, Reciclagem, Partilhados);
  - Selecção dos tipos de documento a listar.

- **Lista de resultados de pesquisa**
  - Ordenação dos resultados por vários campos apresentados;
  - Paginação;
  - Selecção de documentos;
  - Ferramentas de auxilio à selecção (e.g. seleccionar todos, seleccionar nenhum, seleccionar favoritos, inverter selecção, etc.);
  - Arquivo dos múltiplos documentos seleccionados (quando ainda estão em rascunho);
  - Remoção dos múltiplos documentos seleccionados;
  - Aplicação de um tipo de documento aos vários documentos seleccionados;
  - Aplicação de um conjunto de permissões aos vários documentos seleccionados;
  - Edição múltipla da metainformação dos vários documentos seleccionados;
  - Marcar documento como favorito.

- **documento**
  - Apresenta a metainformação do documento, segundo o seu tipo;
  - Capacidade de edição só apresentada mediante as permissões do utilizador;
  - Edição do tipo do documento do documento;
  - Edição de qualquer campo de metainformação que não esteja marcado como sendo só de leitura;
  - Download de todos os ficheiros do documento no formato zip;
  - Guardar as alterações à metainformação de um documento;
  - Arquivar documento;
  - Remover documento;

  - Alterar permissões do documento;
  - Apresentação da lista paginada de ficheiros do documento, com thumbnails, metainformação técnica dos ficheiros e acções (descarregar um ficheiro, remover um ficheiro do documento e mover o ficheiro para outro documento existente ou um novo);
  - Adição de um ficheiro ao documento.

- **Preferências**

  - Preferências de submissão;
  - Gestão dos tipos de documento;
  - Gestão de utilizadores;
  - Gestão de importação automática de email (Configuração das contas de email).

Apêndice B

XML documento

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/
    properties.dtd">
<properties>
<entry key="bundle.submission.date">Fri Apr 16 15:24:46
    WEST 2010</entry>
<entry key="bundle.submitter.address">127.0.0.1</entry>
<entry key="bundle.submitter.host">127.0.0.1</entry>
<entry key="bundle.owner">vfernandes</entry>
<entry key="bundle.remote.user"/>
<entry key="bundle.download.counter">2</entry>
<entry key="bundle.data.files.id">81
    E2CA16E115EAF35D43D89D77AC446B</entry>
<entry key="bundle.id">248C3B9F866BF03AED616D2C55E6D612</
    entry>
<entry key="bundle.submitter.port">57480</entry>
<entry key="bundle.submitter.agent">Jakarta Commons-
    HttpClient/3.1</entry>
<entry key="bundle.permission.readers">vfernandes</entry>
<entry key="bundle.referer"/>
<entry key="bundle.permission.modifiers">vfernandes</
    entry>
<entry key="bundle.number.data.files">1</entry>
<entry key="bundle.active">true</entry>
<entry key="bundle.version">1</entry>
</properties>
```

Apêndice C

XML ficheiros do documento

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/
    properties.dtd">
<properties>
<entry key="81E2CA16E115EAF35D43D89D77AC446B"/>
</properties>
```

Apêndice D

XML ficheiro

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/
    properties.dtd">
<properties>
<entry key="file.mimetype">video/quicktime</entry>
<entry key="file.location">/usr/local/vault/data/81/E2/CA
    /EAF35D43D89D77AC446B/81E2CA16E115EAF35D43D89D77AC446B
    .data</entry>
<entry key="file.filename">Movie.m4v</entry>
<entry key="file.date">Sex Abr 16 15:24:46 WEST 2010</
    entry>
<entry key="file.checksum">81
    E2CA16E115EAF35D43D89D77AC446B</entry>
<entry key="tika.content-type">video/quicktime</entry>
<entry key="file.length">2299207</entry>
<entry key="file.timestamp">1271427886007</entry>
</properties>
```

Apêndice E

XML pesquisa

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<searchRetrieveResponse xmlns="http://www.loc.gov/zing/
    srw/" xmlns:ns2="http://www.loc.gov/zing/srw/
    diagnostic/" xmlns:ns3="http://www.loc.gov/zing/cql/
    xcql/">
     <version>1.0</version>
     <numberOfRecords>1</numberOfRecords>
     <records>
          <record>
               <recordSchema>http://java.sun.com/dtd/
                   properties.dtd</recordSchema>
               <recordPacking>XML</recordPacking>
               <recordData>&lt;properties&gt;
&lt;entry key=&quot;bundle.version&quot;&gt;1&lt;/entry&
    gt;
&lt;entry key=&quot;dc.format.extent&quot;&gt
    ;000000000000000000000000000000000000000002299207&lt
    ;/entry&gt;
&lt;entry key=&quot;dc.date.issued&quot;&gt;2010-04-16
    T14:24:46.33Z&lt;/entry&gt;
&lt;entry key=&quot;bundle.owner&quot;&gt;vfernandes&lt;/
    entry&gt;
&lt;entry key=&quot;bundle.active&quot;&gt;true&lt;/entry
    &gt;
&lt;entry key=&quot;permission.modifiers&quot;&gt;
    vfernandes&lt;/entry&gt;
&lt;entry key=&quot;dc.date.modified&quot;&gt;2010-04-16
    T14:24:46.33Z&lt;/entry&gt;
&lt;entry key=&quot;dc.title&quot;&gt;Movie.m4v&lt;/entry
    &gt;
&lt;entry key=&quot;bundle.permission.modifiers&quot;&gt;
    vfernandes&lt;/entry&gt;
&lt;entry key=&quot;bundle.permission.readers&quot;&gt;
    vfernandes&lt;/entry&gt;
```

```
&lt;entry key=&quot;file.filename&quot;&gt;Movie.m4v&lt;/
    entry&gt;
&lt;entry key=&quot;so.location.original&quot;&gt;Movie.
    m4v&lt;/entry&gt;
&lt;entry key=&quot;dc.type&quot;&gt;other&lt;/entry&gt;
&lt;entry key=&quot;bundle.new.version.of&quot;&gt;
    undefined&lt;/entry&gt;
&lt;entry key=&quot;bundle.has.new.version&quot;&gt;
    undefined&lt;/entry&gt;
&lt;entry key=&quot;dc.creator&quot;&gt;Vitor Fernandes&
    lt;/entry&gt;
&lt;entry key=&quot;bundle.id&quot;&gt;248
    C3B9F866BF03AED616D2C55E6D612&lt;/entry&gt;
&lt;entry key=&quot;dc.date.created&quot;&gt;2010-04-16
    T14:24:46.30Z&lt;/entry&gt;
&lt;entry key=&quot;permission.readers&quot;&gt;
    vfernandes&lt;/entry&gt;
&lt;/properties&gt;</recordData>
            <recordPosition>1</recordPosition>
        </record>
    </records>
</searchRetrieveResponse>
```

Apêndice F

XML tipo de documento

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<docType xmlns="http://www.keep.pt/document/type" id="
    text" default="false" color="003ef9">
    <label>text</label>
    <label lang="pt_PT">texto</label>
    <label lang="en">text</label>
    <description>Text document</description>
    <description lang="pt_PT">Documento de texto</
        description>
    <description lang="en">Text document</description>
    <fieldList>
        <field typeArgs="short" type="text" removable="
            false" readOnly="false" mandatory="true" id="
            dc.title">
            <label>Title</label>
            <label lang="pt_PT">Titulo</label>
            <label lang="en">Title</label>
            <description>Document title</description>
            <description lang="pt_PT">Titulo do documento
                </description>
            <description lang="en">Document title</
                description>
            <defaultValue></defaultValue>
            <copyList>
                <copy field="tika.title" aggregate="first
                    ">
                    <regex>(Microsoft Word \- )?(.*)</
                        regex>
                    <format>%2$s</format>
                </copy>
                <copy field="index.fulltext" aggregate="
                    first">
                    <regex>^[^\w]*(.+)\s*\n</regex>
                    <format>%1$s</format>
                </copy>
```

```
            <copy field="file.filename" aggregate="
                first"/>
        </copyList>
    </field>
    <field typeArgs="short" type="text" removable="
        false" readOnly="false" mandatory="true" id="
        dc.creator">
        <label>Author</label>
        <label lang="pt_PT">Autor</label>
        <label lang="en">Author</label>
        <description>Document author</description>
        <description lang="pt_PT">Autor do documento<
            /description>+
        <description lang="en">Document author</
            description>
        <copyList>
            <copy field="tika.author" aggregate="
                first"/>
            <copy field="system.user.fullname"
                aggregate="first"/>
            <copy field="system.user" aggregate="
                first"/>
        </copyList>
    </field>
    <field typeArgs="long" type="text" removable="
        false" readOnly="false" mandatory="false" id="
        dc.description">
        <label>Description</label>
        <label lang="pt_PT">Descricao</label>
        <label lang="en">Description</label>
        <description>The description of the document<
            /description>
        <description lang="pt_PT">A descricao do
            documento</description>
        <description lang="en">The description of the
             document</description>
        <copyList>
            <copy field="tika.subject" aggregate="
                first"/>
            <copy field="tika.description" aggregate=
                "first"/>
        </copyList>
    </field>
    <field type="date_time" removable="false"
        readOnly="false" mandatory="false" id="dc.date
        .created">
        <label>Created at</label>
        <label lang="pt_PT">Criado em</label>
        <label lang="en">Created at</label>
        <description>The date the document was
            created</description>
```

```
            <description lang="pt_PT">Data de criacao do
                documento</description>
            <description lang="en">The date the document
                was created</description>
            <copyList>
                <copy field="tika.created" aggregate="
                    first"/>
                <copy field="date" aggregate="first"/>
                <copy field="system.date" aggregate="
                    first"/>
            </copyList>
        </field>
        <field type="date_time" removable="false"
            readOnly="true" mandatory="true" id="dc.date.
            issued">
            <label>Issued at</label>
            <label lang="pt_PT">Submetido em</label>
            <label lang="en">Submitted at</label>
            <description>The date the document was
                submitted to the system</description>
            <description lang="pt_PT">Data de submissao
                do documento no sistema</description>
            <description lang="en">The date the document
                was submitted to the system</description>
            <copyList>
                <copy field="previous.dc.date.issued"
                    aggregate="first"/>
                <copy field="system.date" aggregate="
                    first"/>
            </copyList>
        </field>
        <field type="date_time" removable="false"
            readOnly="true" mandatory="false" id="dc.date.
            available">
            <label>Available at</label>
            <label lang="pt_PT">Disponibilizado em</label
                >
            <label lang="en">Available at</label>
            <description>The date the document was
                accepted in the system</description>
            <description lang="pt_PT">A data de aceitacao
                 do documento no sistema</description>
            <description lang="en">The date the document
                was accepted in the system</description>
            <copyList/>
    </fieldList>
</docType>
```

Apêndice G

XML utilizador

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user xmlns="http://pt.keep.objectvault/user" xmlns:ns2="
    http://pt.keep.objectvault/userCollection">
    <username>vfernandes</username>
    <fullName>Vitor Fernandes</fullName>
    <email>vitorfernandes87@gmail.com</email>
    <deleted>false</deleted>
    <groups>
        <groupName>vfernandes</groupName>
        <groupName>anonymous</groupName>
    </groups>
</user>
```

Apêndice H

XML vocabulário controlado

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY terms "http://purl.org/dc/terms/" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY owl2 "http://www.w3.org/2006/12/owl2#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#
        " >
    <!ENTITY skos "http://www.w3.org/2004/02/skos/core#"
        >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#"
        >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax
        -ns#" >
]>
<rdf:RDF xmlns="file:/usr/local/tomcat/temp/7
    lI5n9baqn8846011230924059574.rdf#"
     xml:base="file:/usr/local/tomcat/temp/7
         lI5n9baqn8846011230924059574.rdf"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:terms="http://purl.org/dc/terms/"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
         ns#"
     xmlns:owl2="http://www.w3.org/2006/12/owl2#"
     xmlns:skos="http://www.w3.org/2004/02/skos/core#">
    <owl:Ontology rdf:about=""/>

    <owl:AnnotationProperty rdf:about="&dc;title"/>
    <owl:AnnotationProperty rdf:about="&dc;description"/>
    <owl:AnnotationProperty rdf:about="&dc;rights"/>
```

```
    <!-- http://purl.org/dc/terms/issued -->
    <owl:DatatypeProperty rdf:about="&terms;issued"/>
    <!-- http://www.w3.org/2004/02/skos/core#prefLabel --
        >
    <owl:DatatypeProperty rdf:about="&skos;prefLabel"/>

    <!-- http://www.w3.org/2004/02/skos/core#Concept -->
    <owl:Class rdf:about="&skos;Concept"/>
    <!-- http://www.w3.org/2004/02/skos/core#
        ConceptScheme -->
    <owl:Class rdf:about="&skos;ConceptScheme"/>

    <!-- 4VldjvzzfT -->
    <skos:Concept rdf:about="4VldjvzzfT">
        <skos:prefLabel>Horror</skos:prefLabel>
    </skos:Concept>
    <!-- 7lI5n9baqn -->
    <skos:ConceptScheme rdf:about="7lI5n9baqn">
        <dc:title>Movie genre</dc:title>
        <terms:issued></terms:issued>
        <dc:description></dc:description>
        <dc:rights></dc:rights>
    </skos:ConceptScheme>
    <!-- R4O28qqGnA -->
    <skos:Concept rdf:about="R4O28qqGnA">
        <skos:prefLabel>Comedy</skos:prefLabel>
    </skos:Concept>
    <!-- VdmBJW8U16 -->
    <skos:Concept rdf:about="VdmBJW8U16">
        <skos:prefLabel>Porn</skos:prefLabel>
    </skos:Concept>
    <!-- c0eYrQpA6g -->
    <skos:Concept rdf:about="c0eYrQpA6g">
        <skos:prefLabel>Drama</skos:prefLabel>
    </skos:Concept>
    <!-- h68Ue3wUU9 -->
    <skos:Concept rdf:about="h68Ue3wUU9">
        <skos:prefLabel>Action</skos:prefLabel>
    </skos:Concept>
    <!-- mOJf50Ovb7 -->
    <skos:Concept rdf:about="mOJf50Ovb7">
        <skos:prefLabel>Animation</skos:prefLabel>
    </skos:Concept>
</rdf:RDF>
```

Apêndice I

doc2pdf.sh

```
#!/bin/bash

###############Local Parameters
    ############################

#Virtual Display which Microsoft Office will open under (
    since MSOffice needs a DISPLAY)
export DISPLAY=:10.10

#location of Virtual Frame Buffer X Server (Xvfb)
Xvfb_CMD=/usr/bin/Xvfb

#Temporary directory which will be used as Xvfb Frame
    Buffer Directory
Xvfb_TEMP_DIR=/var/tmp/

#Xvfb Parameters
Xvfb_PARAM="$DISPLAY -screen 10 800x600x16 -ac -br -c -
    fbdir $Xvfb_TEMP_DIR"

#location of Wine script
Wine_CMD=/usr/bin/wine

#Wine Parameters
Wine_PARAM="start"

#location of MSOffice
MSOffice_CMD="'C:\\Program Files\\Microsoft Office\\
    Office12\\WINWORD.exe'"

#MSOffice Parameters
MSOffice_PARAM="/mDOC2PDF"

#Killall script
KILL_ALL=/usr/bin/killall
```

```
# Call this script with 4 parameters [-i] <inputfile> [-o
    ] <outputfile>
PARAMS=4


# Check for Xvfb
if [ ! -f $Xvfb_CMD ]; then
    echo "Error: Could not find $Xvfb_CMD (Virtual Frame
        Buffer X Server), which is needed to run MSOffice
        as a server."
    exit
fi

# Check for Xvfb Temp Directory
if [ ! -d $Xvfb_TEMP_DIR ]; then
    echo "Error: Could not find $Xvfb_TEMP_DIR (Xvfb
        Temporary Directory), which is needed to run
        MSOffice as a server."
    echo "Please create this location or use an existing
        temporary directory."
    exit
fi

# Check for Wine
if [ ! -f $Wine_CMD ]; then
    echo "Error: Wine startup command does not seem to be
         installed at $Wine_CMD"
    exit
fi

# Check for killall
if [ ! -f $KILL_ALL ]; then
    echo "Error: Could not find $KILL_ALL (killall
        command), which is needed for this script to be
        able to kill a running Xvfb (Virtual Frame Buffer
        X server)."
    exit
fi



convert() {

        echo "Converting $1"

        # directory name
        dirname=`dirname "$1"`

        cd $dirname
```

```
        # basename
        basename=`basename "$1"`

        #start MSOffice
        $Wine_CMD $MSOffice_CMD $basename $MSOffice_PARAM
            > /dev/null 2>&1 &

        pid=`ps -eaf | grep "WINWORD.exe" | grep
            $basename | awk '{print $2}'`

        while [[ $pid != "" ]] && ps -p $pid > /dev/null;
            do
                sleep 0.5
                pid=`ps -eaf | grep "WINWORD.exe" | grep
                    $basename | awk '{print $2}'`; done

}

# See how script is being called
if [ $# -ne "$PARAMS" ]
        then
        echo "Usage: $0 [-i] <inputfile> [-o] <outputfile
            >"
else
        if [ $1 != "-i" ]
                then
                echo "Usage: $0 [-i] <inputfile> [-o] <
                    outputfile>"
        else
                if [ $3 != "-o" ]
                        then
                        echo "Usage: $0 [-i] <inputfile>
                            [-o] <outputfile>"
                else
                        convert $2
                        extpdf=".pdf"

                        inputfilebasename=`basename "$2"`
                        pdffile=${inputfilebasename%.*}
                            $extpdf
                        cd ~
                        mv $pdffile $4
                        echo "done."
                fi
        fi
fi

exit 0
```

Apêndice J

View Controllers

LoginView

Classe que controla a interface de autenticação, permitindo ao utilizador preencher os dados essenciais. Posteriormente após selecção do botão de retorno no teclado tenta fazer a autenticação do utilizador. Esta classe é um *view controller* que implementa os protocolos UITableViewDelegate e UITableViewDataSource a fim de poder controlar a tabela. Implementa também o protocolo UITextFieldDelegate, tendo assim poder sobre as caixas de texto que aparecem na tabela.

A tabela possui três entradas, uma para que o utilizador insira o seu nome, outras para a palavra-passe e for fim uma para inserir o servidor. Cada entrada possui uma *label* para a identificar e uma caixa de texto para o utilizador introduzir dados. A *label* é uma tradução para a entrada respectiva do dicionário, que foi carregado na inicialização da aplicação (consiste num atributo do objecto *delegate* da aplicação). Relativamente às caixas de entrada, também funcionam de maneira diferente entre si. A caixa de texto do nome de utilizador é a normalmente usada por todas as aplicações, no entanto as outras diferem dessa. A caixa de texto da palavra-passe esconde os caracteres que são inseridos e a do servidor abre um teclado para inserir um caminho (url) em vez do teclado normal.

Após termino do carregamento do ecrã, o *view controller* fica a espera que o utilizador insira os dados e carregue no botão de retorno do teclado. O toque nesse botão desencadea uma série de eventos. Primeiro invoca o método de autenticação de um objecto WebServiceConnect. Caso o método não ocorra com sucesso é mostrado ao utilizador uma mensagem de erro neste mesmo ecrã. Se a autenticação ocorrer com sucesso, o nome completo do utilizador (retornado no método de autenticação) é guardado, e então são carregados do servidor e guardados todos os tipos

de documentos e vocabulários controlados a que o utilizador tem acesso (invocando métodos do objecto WebServiceConnect). Tanto os dados inseridos, como o nome completo, tipos de documentos e vocabulários são guardados no objecto *delegate* da aplicação. O nome do utilizador, palavra-passe e servidor são também guardados no ficheiro de sistema da aplicação. Após a inicialização e carregamento do servidor dos dados necessários, o utilizador é conduzido para o ecrã de filtragem e escolha de tipos de documentos.

BoxController

Classe que controla a interface de filtragem e escolha dos tipos de documentos. É responsável por preencher a tabela com os dados essenciais e fazer a sua gestão. Permite ao utilizador seleccionar os tipos de documentos e escolher o filtro (Rascunho, Os meus documentos, entre outros). Esta classe é um *view controller* que implementa os protocolos UITableViewDelegate e UITableViewDataSource a fim de poder controlar a tabela.

A tabela vai ser constituída por duas secções, uma para os filtros e outra para os tipos de documentos. As entradas para as duas secções serão diferentes e o toque numa entrada tem um funcionamento distinto de acordo com a secção em que está. Se for a secção de filtros, então uma entrada tem a imagem identificativa e o nome do filtro traduzido de acordo com a linguagem que foi carregada para o dicionário (guardado no objecto *delegate* da aplicação). Para cada entrada desta secção aparece uma seta que permite o utilizador ir para o ecrã de pesquisa simples. O filtro escolhido é guardado no objecto *delegate* para controlar as futuras pesquisas. Para a secção dos tipos de documentos cada entrada possui uma imagem com a cor respectiva desse tipo de documento e a *label* identificativa de acordo com a linguagem escolhida no dispositivo. O toque numa entrada desta secção permite ao utilizador seleccionar ou não um tipo de documento, sendo a imagem dessa entrada alterada. A imagem funciona como uma *checkbox* onde o utilizador pode confirmar o estado da selecção daquele tipo de documento, mudando de cor para cada tipo de documento. Como foi dito anteriormente o filtro seleccionado é guardado no objecto *delegate* da aplicação, mas os documentos seleccionados no momento do toque num filtro também são guardados nesse objecto de forma a controlar a pesquisa básica e a pesquisa básica.

O ecrã vai mostrar na barra do ecrã o nome completo do utilizador que foi retornado pela função de autenticação, invocada no inicio da aplicação ou no ecrã de autenticação, e que se encontra guardado no objecto

delegate da aplicação. Para além do nome completo, aparece na barra um botão para o utilizador sair para o ecrã de autenticação. Ao carregar nesse botão os dados (nome de utilizador, palavra-passe e servidor) inseridos anteriormente pelo utilizador são limpos do ficheiro de sistema da aplicação e aparece o ecrã de autenticação.

SearchApplicationViewController

Esta classe vai ser responsável por controlar o ecrã de pesquisa básica. É responsável por executar pesquisas ao servidor de acordo com os filtros e com os tipos de documentos seleccionados no ecrã anterior e por listar os resultados dessas pesquisas. Tal com as outras classes anteriores, esta classe é um *view controller* que implementa os protocolos UITableViewDelegate e UITableViewDataSource a fim de poder controlar a tabela. Implementa também o protocolo UISearchBarDelegate para controlar a barra de pesquisa que vai aparecer no ecrã.

O título do ecrã, que aparece na barra superior do ecrã, vai ser o nome do filtro seleccionado no ecrã de escolha de filtros e dos tipos de documentos. Na barra superior aparece também dois botões, uma para voltar ao ecrã anterior (escolha de filtros e documentos) e outra para passar para o ecrã de pesquisa avançada.

Ao carregar o ecrã é feita uma pesquisa ao servidor (método de pesquisa de um objecto do tipo WebServiceConnect) pelos primeiros 30 documentos que cumpram os requisitos seleccionados pelo utilizador (filtro e tipos de documentos). Esses documentos são listados na tabela, sendo para cada um deles apresentado o seu nome, o seu autor e o tamanho numa entrada da tabela. Aparece também uma imagem com a cor do tipo de documento a que o documento está associado. Caso existam mais do que 30 documentos a listar (número guardado num atributo do objecto Searchdocumentos retornado na pesquisa), aparece uma entrada da tabela a dizer que existem mais documentos para carregar. Se o utilizador tocar nessa entrada são carregados mais documentos (outro método de pesquisa de um objecto WebServiceConnect), sendo os mesmos adicionados à tabela. Este algoritmo repete-se até todos os documentos terem sido carregados.

Caso o utilizador insira alguns dados na caixa de texto da barra de pesquisa e carregar no botão de pesquisa do teclado, é feita uma pesquisa ao servidor (método de pesquisa de um objecto WebServiceConnect) pelos primeiros 30 documentos, tendo em conta os dados introduzidos, os filtros e os tipos de documentos. O funcionamento da listagem funciona de maneira idêntica ao descrito anteriormente.

O toque sobre um documento conduz o utilizador para o ecrã com todas as informações do documento. Antes de o utilizador entrar nesse ecrã é feito um pedido ao servidor com para obter a lista de todos os identificadores dos ficheiros do documento em questão e para os 10 primeiros (ou os que houver caso sejam menos desse número) são feitos pedidos ao servidor pelas suas informações/metadados. Esses pedidos são feitos invocando métodos de um objecto. Após carregar todos os dados anteriormente descritos o utilizador passa então para o ecrã com a informações do documento.

AdvancedSearch

Esta classe vai ser responsável por controlar o ecrã de pesquisa avançada. É responsável por controlar todas as interacções com o utilizador, bem como interagir com diferentes tipos de dados, pois existem dados de texto, datas, tamanhos e vocabulários controlados. É um *table view controller*, que é uma subclasse de *view controller* e já disponibiliza os métodos para controlar a tabela, pois apenas possui a tabela como elemento de interface.

A tabela apenas vai ter uma secção onde são listados todos os campos dos vários tipos de documentos previamente seleccionados no ecrã de escolha de filtros e de tipos de documentos. Antes de o ecrã ser inicializado é executado um método que junta os campos desses tipos de documentos, eliminado campos repetidos para evitar ambiguidades. É um método complexo, pois tem que correr todos os tipos de documentos seleccionados e todos os seus campos. Se o utilizador não tiver seleccionado nenhum dos tipos de documentos anteriormente então esse método tem em conta todos os tipos de documentos sem excepção.

Após o termino do método é preenchida a tabela com os campos retornados. Um tipo de documento possui informações relativas a cada campo como, por exemplo: o tipo, se é obrigatório, traduções para o nome e descrições em várias linguagens, entre outras. Essas informações influenciam a maneira como os campos vão ser apresentados na tabela. Para cada campo é criada uma entrada na tabela, entrada essa que depende do tipo do campo. Qualquer entrada na tabela vai possui uma *label* com o nome desse campo de acordo com a linguagem do dispositivo. Se o campo a ser apresentado for de texto para além da *label* com o nome possui uma caixa de entrada onde o utilizador pode inserir dados. Se o seu tipo for data, tamanho ou vocabulário controlado aparece, para além da *label* com o nome do campo, o valor associado. No inicio esse valor é vazio aparecendo então uma descrição carregada do dicionário guardado

no objecto *delegate* da aplicação. Caso o utilizador carregue numa destas entradas é conduzido para o ecrã de selecção de intervalos de datas, intervalo de tamanhos e de vocabulários controlados. Os valores seleccionados nesses ecrãs são sempre retornados ao ecrã de pesquisa avançada e os valores da tabela são actualizados. As entradas para esses campos são dinâmicas, pois os valores podem ocupar bastante espaço, principalmente para listar os vocabulários controlados seleccionados. Caso um entrada já tenha valores, esses valores são passados para os ecrãs de selecção.

Após o utilizador introduzir todos os dados que necessitar possui um botão na barra superior para verificar os resultados da pesquisa com os dados que introduziu. O utilizador é conduzido para o ecrã de resultados, sendo os dados por ele introduzidos passados para o *view controller* desse ecrã.

AdvancedSearchList

Esta classes vai ser responsável por controlar a interface do ecrã dos resultados da pesquisa avançada. É apenas responsável por fazer um pedido ao servidor com os resultados dos dados inseridos no ecrã de pesquisa avançada e com o filtro e tipos de documentos seleccionados no ecrã de selecção dos mesmos. É um subclasse *table view controller*, uma vez que, só possui a tabela como elemento de interface e apenas tem que fazer a sua gestão.

Funciona de maneira parecida ao *view controller* que controla o ecrã de pesquisa avançada, apenas não possui a função de pesquisa básica. Em termos de interface só vai apresentar uma tabela, onde cada entrada vai conter informações sobre um dos documentos retornados pelo pedido ao servidor. Tal como no ecrã de pesquisa avançada apenas os 30 primeiros documentos são apresentados. Caso a pesquisa possua mais documentos para listar aparece uma entrada na tabela a informar o utilizador. O toque na tabela permite buscar mais informações sobre 30 documentos (ou os que houver, caso sejam menos). Este algoritmo repete-se até que estejam carregados todas as informações sobre todos os documentos que se encontrem de acordo com os parâmetros pesquisados.

Para cada entrada na tabela, é apresentado informações sobre o documento, tal como foi dito anteriormente. Para cada documento são apresentados o seu nome, o seu autor e o seu tamanho. É ainda apresentada uma imagem com o cor do tipo de documento a que o documento se encontra associado. O toque numa entrada, encaminha o utilizador para o ecrã de informações do documento, onde é mostrada toda a informação

do documento ao utilizador, incluindo os ficheiros do mesmo.

Relativamente ao título do ecrã, ele vai ser a tradução, encontrada para no dicionário guardado no objecto *delegate* da aplicação, da expressão "Resultados". Na barra superior vai ainda possuir um botão para voltar ao ecrã de pesquisa avançada.

InfoBundle

Esta classe vai ser responsável por controlar o ecrã de informação de um documento. Essa informação pode ser o tipo de documento a que está associado, metadados do documento e os ficheiros que pertencem a esse documento. Tal como muitas das classes anteriores, esta classe é um *view controller* que implementa os protocolos UITableViewDelegate e UITableViewDataSource a fim de poder controlar a tabela. A tabela vai esta dividida em 3 secções: uma apenas para mostrar o tipo de documento a que o documento está associado, outra para mostrar os metadados do documento e outra para mostrar os ficheiros pertencentes a esse documento.

A primeira secção apenas tem uma entrada que mostra uma imagem com a cor do tipo de documento associado ao documento e uma *label* com o nome desse mesmo tipo de documento de acordo com a linguagem do dispositivo.

A segunda secção é um bocado mais complexa. Antes de mais, é preciso referir que um tipo de documento contem informações (tipo, se é só de leitura, obrigatório, traduções, entre outras) sobre os vários campos que um documento pode ter e que os metadados de documento não são mais que esses campos associados a um valor. Portanto existe uma ligação entre os campos de um tipo de documento e os metadados de documento. Isto vai influenciar a maneira como os metadados de documento vão aparecer na tabela, onde todos os campos que possuam valor vazio não vão aparecer na tabela e a ordem com que os campos aparecem na tabela é a igual a ordem que aparecem no tipo de documento. Cada entrada na tabela vai possui duas *labels*: uma para a tradução do campo em questão de acordo com a linguagem do dispositivo (obtida no tipo de documento) e outra para o valor desse campo (obtida dos metadados do documento). Resta referir, que o valor do campo pode ser formatado de acordo com o seu tipo, ou seja, se for uma data tem uma determinada formatação, se for um tamanho outro tipo de formatação, o mesmo acontecendo para o caso de ser texto (*short* ou *full*) ou vocabulários controlados. O tamanho das entradas varia de acordo com o valor dos campos já formatados.

Por fim a terceira secção mostra os ficheiros pertencentes ao documento em questão. A lista com os identificadores dos ficheiros e com informações sobres os 10 primeiros (ou menos caso não possua mais) foram carregadas antes do ecrã ser inicializado. Para cada um dos ficheiros cuja informação foi carregada do servidor existe uma entrada na tabela. Cada entrada da tabela é constituída por uma imagem que identifica o tipo de ficheiro, o nome do ficheiro e o seu tamanho posteriormente formatado para aparecer de forma mais legível. Se o utilizador carregar numa dessas entradas é conduzido para o ecrã de visualização desse ficheiro. Caso o documento possua mais de 10 ficheiros aparece mais uma entrada a informar o utilizador dessa situação, cujo toque faz com que se carregue informações relativas a mais ficheiros tendo em conta a ordem da lista dos identificadores. São carregadas informações relativas a 10 ficheiros ou menos se for o caso. O algoritmo repete-se até todos os ficheiros terem sido carregados.

Relativamente à barra superior do ecrã, tem como título o nome do documento e um botão para voltar ao ecrã de pesquisa básica.

File

Classe que vai controlar o ecrã de visualização de ficheiros. É responsável por executar um pedido ao servidor sobre o ficheiro para posterior visualização. É um simples *view controller* que, ao contrário dos outros descritos anteriormente, não implemente nenhum protocolo adicional.

O seu título consiste no nome do ficheiro que se encontra a ser visualizado/carregado, e na barra superior contem apenas um botão para voltar ao ecrã com as informações do documento a que o ficheiro pertence. Este ecrã possui um barra de ferramenta com três botões, que permitem ao utilizador navegar entre os vários ficheiros do documento e/ou consultar as informações relativas do ficheiro a ser carregado/visualizado. Ao pressionar o botão relativo as informações o utilizador vai para um novo ecrã onde pode consultar todas as informações do ficheiro em questão.

Relativamente ao carregamento do ficheiro havia duas hipóteses. A primeira hipótese e a pior em termos de tráfego e gestão da aplicação era fazer o *download* do ficheiro digital para a pasta temporária de ficheiros do dispositivo. No entanto isto implicava que a aplicação estivesse parada enquanto o *download* do ficheiro era executado. Após o termino do *download* o ficheiro era apresentado. A segunda hipótese era carregar o ficheiro como se tratasse de um *website*, sendo o mesmo mostrado conforme vai sendo carregado. A opção implementada acabou por recair sobre a segunda hipótese.

Se o utilizador tocar no botão para visualizar o próximo ficheiro, o novo ficheiro vai sendo carregado, o título da página é modificado para o nome desse ficheiro e caso não exista mais ficheiros o botão seguinte é desactivado. Se o utilizador tocar no botão para visualizar o ficheiro anterior, tal como no outro caso, o novo ficheiro vai sendo carregado, o título é modificado e o botão anterior é desactivado no caso de não haver mais ficheiros anteriores ao ficheiro em questão.

InfoFile

Esta classes vai ser responsável por controlar a interface do ecrã de informações do ficheiro. É apenas responsável por mostrar todas essas informações de forma ordenada, formatada e traduzida, caso exista tradução. É um subclasse *table view controller* simples, uma vez que, só possui a tabela como elemento de interface e apenas tem que fazer a sua gestão.

Relativamente à tabela, esta depende muito das informações que o ficheiro possui. É necessário referir, que um ficheiro funciona como um documento em termos de metadados, sendo os mesmos constituídos por um conjunto de chave e valor. Todos as chaves cujos valores sejam nulos ou vazios não vão ser apresentados. Existem dois tipos de apresentações: a básica onde são mostrados apenas algumas chaves e respectivos valores (caso o ficheiro os tenha) seguindo uma determinada ordem e a complexa onde os restantes metadados são adicionados à apresentação básica, retirando apenas algumas excepções.

Primeiro, a tabela só possui entradas com os campos básicos e por uma ordem. Cada entrada tem uma *label* com a chave, traduzida caso exista tradução no dicionário do objecto *delegate* da aplicação ou formatada de acordo com certas regras, e o valor relativo a essa chave, que também pode ser formatado de acordo com o seu tipo. Caso o ficheiro possua mais metadados dos que os listados, então aparece uma entrada a informar o utilizador que existem mais informações que as listadas. Após toque nessa entrada aparece o resto dos metadados, sendo as entradas iguais às da apresentação básica, ou seja, uma *label* para a chave (traduzida ou formatada) e outra para o valor associado a essa chave (que pode estar formatado).

Relativamente ao título do ecrã é o nome do ficheiro cuja informação está a ser visualizada. Possui ainda um botão para voltar ao ecrã de visualização de ficheiros.

Picker

Classe responsável por controlar a interface do ecrã de selecção de datas ou de tamanhos. Permite ao utilizador seleccionar um intervalo de datas ou tamanhos para posteriormente ser devolvido para outro ecrã. Implementa como muitas das classes anteriores os protocolos UITableViewDelegate e UITableViewDataSource a fim de poder controlar a tabela. Neste caso a tabela só vai ter duas entradas para que utilizador possa definir um intervalo (desde um valor até outro). Cada entrada é constituída por um *label* e por um valor. Essa *label* depende da tradução encontrada no dicionário que se encontra guardado no objecto *delegate* da aplicação. Para a primeira entrada aparece a tradução para a expressão "desde" e na segunda entrada aparece a tradução para a expressão "até". Quanto ao valor, caso seja vazio, aparece também uma tradução.

Caso o utilizador carregue num entrada aparece um seleccionador que depende do tipo: se é data ou se é um tamanho. Se for uma data, aparece um seleccionador com 3 colunas: uma para ano, outra para o mês e uma terceira para o dia. Se for um tamanho, o seleccionador tem 4 colunas: 3 colunas com número de 0 a 9 (permitindo ao utilizador no seu conjunto definir número de 000 até 999) e uma colunas com as várias unidades em *bytes*. Tanto no seleccionador de datas, como no seleccionar de tamanhos, uma interacção altera o valor da entrada. Caso a entrada já tenha o valor, ao carregar na entrada o seleccionador fica automaticamente com o valor dessa entrada. Se a entrada não tiver nenhum valor, no seleccionador de datas aparece o dia actual e no seleccionador de tamanhos aparece o tamanho correspondente a 0 *bytes*. A frente de cada entrada existe um botão que permite eliminar o valor dessa entrada, metendo-o vazio.

Este ecrã permite, assim, seleccionar um intervalo de datas ou de tamanhos. O nome do campo aparece como título do ecrã. Na barra superior aparece também dois botões: uma para cancelar todas as alterações e outro para guardar as alterações. Tanto um como o outro retornam para o ecrã de pesquisa avançada. Caso as alterações sejam gravadas, a entrada do campo é modificada mostrando o novo valor.

VocabularyView

Classe que vai controlar a interface do ecrã da selecção de vocabulário controlados. É responsável por controlar a navegação pelos vocabulários de um vocabulário controlado. Um vocabulário controlado não é mais que uma espécie de árvore onde existem os vocabulários de topo (não têm pai) que podem possuir vocabulários filhos. Esses vocabulários fi-

lhos podem também possuir vocabulários filhos e assim sucessivamente, gerando-se uma espécie de arvore. Esta classe é responsável por permitir ao utilizador subir e deixar na árvores, seleccionando os vocabulários que desejar. É um *table view controller* simples, pois apenas possui uma tabela como elemento gráfico, tendo que fazer a sua gestão.

No inicio do ecrã a tabela apenas possui uma secção. Nessa secção são listados os vocabulários de topo do vocabulário controlado em questão. Para cada um deles é criada uma entrada na tabela que possui uma imagem e o nome desse vocabulário. A imagem vai funcionar como uma *checkbox*, sendo alterada conforme o toque nessa entrada e mostrando se o vocabulário em questão está ou não seleccionado. As entradas podem ainda possuir uma seta, que só aparece nos vocabulários que possuem filhos. Carregando na seta o utilizador desce um nível na árvore, permitindo visualizar os filhos do vocabulário em que carregou.

Para os níveis inferiores da arvore o ecrã tem uma tabela com duas secções. A primeira secção tem apenas uma entrada, como o nome do vocabulário em que estamos e uma seta para a esquerda. O toque nessa entrada leva o utilizador para o ecrã anterior, sendo a tabela modificada. Na segunda secção aparece os vocabulários filhos do vocabulário onde o utilizador se encontra. Cada vocabulário filho tem uma entrada na tabela, entrada essa que tem uma imagem e o nome do vocabulário filho. A imagem tal como para os vocabulários de topo funciona como *checkbox*, permitindo seleccionar ou não esse vocabulário. Caso esse vocabulário tenha filhos aparece uma seta na entrada, cujo toque permite ao utilizador descer mais um nível na árvore.

Este dois ecrãs permitem, assim, seleccionar os vocabulários de um vocabulário controlado. O nome desse vocabulário controlada aparece como título do ecrã. Na barra superior aparece também dois botões: uma para cancelar todas as alterações e outro para guardar as alterações. Tanto um como o outro retornam para o ecrã de pesquisa avançada. Caso as alterações sejam gravadas, esses ecrãs sofreram alterações.

Apêndice K

weebox: Testes de usabilidade

Nº	Tarefa	Sucesso	Tempo	Observações
1	Aceda ao http://weebox.keep.pt	100%	0m 7s	
2	Encontre os preços da aplicação	100%	0m 12s	
3	Crie um conta na versão disponível e entre com a sua nova conta	100%	0m 48s	Alguns já tinham uma conta criada
4	Crie um documento e envie um ficheiro à sua escolha	100%	1m 54s	Todos criaram um documento vazio, descreveram e depois arquivaram. Isto é ineficiente e não aproveita o preenchimento automático de metadados. Problema #1
5	Defina o tipo de documento e a informação associada que o descreve	100%	1m 30s	Muitos fizeram isto no passo 4. Alguma confusão entre o filtro e o editor do documento num dos casos. Noutro caso houve confusão entre metadados de ficheiro e documento, tentando ir mudar algo no ficheiro.
6	Arquive o documento	100%	0m 8s	Muitos fizeram isto no passo 4.
7	Contabilize todos os documentos existentes no sistema	100%	0m 33s	Um demorou tempo porque achou estranho existirem tão poucos documentos. Outro achou que o Ver 30, 100, 1000 confundia um pouco.
8	Tente encontrar o seu documento	100%	0m 12s	
9	Partilhe o seu documento com o utilizador 'Luis Faria'	100%	0m 25s	Um demorou 1m 16s porque teve dificuldade em perceber o push button.
10	Descarregue o seu documento para o desktop do seu computador	100%	0m 24s	
11	Crie um novo documento, agora com vários ficheiros	100%	1m 31s	Muitos criaram um novo documento vazio, descreveram e só depois adicionaram. Poucos usaram CTRL / SHIFT para escolher vários ficheiros simultaneamente. Aconteceram os Erros #1 e #2
12	Arquive o novo documento	100%	0m 10s	Uma pessoa demorou 1m 30s porque esteve a descrever.
13	Veja todos os documentos	100%	0m 11s	
14	Edite os seus documentos simultaneamente e altere o Autor associado a ambos	100%	0m 20s	
15	Elimine um dos seus documentos	100%	0m 13s	
16	Recupere o documento eliminado	100%	0m 14s	Gostaram mto de como esta funcionalidade estava implementada (gmail)
17	Marque um documento como favorito, defina o tipo do seu documento como 'vídeo' e altere a "Categoria de vídeo" para "Ficção científica"	86%	1m 22s	Um teve dificuldade em encontrar o favorito. Alguns procuraram categoria de video em tipo de documento. Alguns não carregaram na seta do editor de cat. de video e não verificaram o resultado.
18	Encontre o seu documento recorrendo ao filtro "Pastas"	86%	1m 18s	Muita dificuldade em encontrar o filtro "pastas", muitas vezes esta estava fora do ecrã. Problemas com filtros simultâneos. Problema #2 e #3.
19	Limpe/remova o filtro "Pastas"	100%	0m 23s	Dificuldade a limpar o filtro "Pastas" por ser complexo. Problema #4.
20	Encontre o seu documento recorrendo ao filtro "Tipo de documento"	86%	0m 13s	Um não encontrou o filtro, pensou que se estava a referir ao editor.
21	Utilize a pesquisa avançada para encontrar o seu documento	71%	0m 35s	Um confundiu pesq. avançada com a básica, usou texto integral e filtro de video, o que não permitiu encontrar o documento, precisou de ajuda. Outro teve problemas a limpar filtros (Problema #3). Aconteceu o Erro #3.
22	Nas 'Definições', adicione uma conta de email para ser importado pelo sistema	71%	1m 4s	Foi detectado o Erro #4, ao remover o email o painel não desaparece - Problema #5. O gravar às vezes está fora do ecrã, um não guardou por causa disso - Problema #6. Ao gravar, configurações não testadas tem de o ser - Problema #7.
23	Altere a "Partilha por omissão" dos documentos depositados por si	86%	0m 59s	Um falhou porque não gravou - Problema #6. Alguma confusão na utilização dos filtros "Todos", "Partilhado" e "Não partilhado" por pensarem que era uma opção (estado) e não simplesmente um filtro da lista abaixo - Problema #8.
24	Veja as estatísticas de quantos ficheiros descarregou	100%	0m 43s	Alguns estavam à espera de o #Descarregados estar sob Ficheiros e não sob Acesso, mas admitiram que também fazia sentido.
25	Aceda ao portal público	100%	0m 11s	
26	Pesquise pelo documento "Ubuntu default backgrounds" no Portal público	100%	0m 24s	Duplo reload - Erro #5.
27	Descarregue o documento neste portal	100%	0m 15s	Um tentou right click->"Save as..." e como viu que não era possível tentou click normal.
28	Volte ao weebox	100%	0m 32s	Problemas a encontrar o link para o manager, muitos tentaram a banner e depois ficaram bastante tempo a tentar descortinar. Um preferiu mesmo modificar o URL. Problema #9.
29	Consulte os limites de armazenamento que está a utilizar	71%	1m 17s	Muita dificuldade em encontrar os limites de armazenamento. Muitos procuram em Definições->Estatísticas. Problema #10.
30	Saia do weebox	100%	0m 8s	

Figura K.1: Dados os testes de usabilidade (Web)

#	Descrição	Solução
1	Tendência a criar documento vazio, descrever e só depois adicionar ficheiros, o que torna a funcionalidade de preenchimento automático ineficiente	Definir "Enviar vários ficheiros para um documento" como omissão do botão "Novo documento"
2	Filtro "Pastas" fora do ecrã	Permitir minimizar o filtro de tipos de documento e pastas
3	Problemas com filtros simultâneos	Política para limpar filtros automaticamente, como ao mudar de filtro primário.
4	Dificuldade a limpar o filtro "Pastas"	Botão dedicado para limpar completamente este filtro
5	Ao remover o email o painel não desaparece	Remover painel se nenhuma entrada escolhida.
6	Nas definições, o gravar está às vezes fora do ecrã e o utilizador pode não ter sucesso por causa disso	Aviso ao sair da tab de que não gravou as modificações. O mesmo para as modificações no documento.
7	Ao gravar na importação do email, emails não testados não o chegam a ser.	Verificar se todos os novos emails foram testados e automaticamente testar-los.
8	No painel de "Partilha por omissão", os filtros "Todos", "Partilhados", e "Não partilhados" fazem alguma confusão. O utilizador tem tendência a vê-los como um estado e não como um filtro.	Alterar o desenho dos botões.
9	No portal público (weebox Community) voltar ao weebox Manager não é claro. A banner é muitas vezes tentada.	Mudar o texto do botão.
10	Muita dificuldade em encontrar os limites de armazenamento. Muitos procuram em Definições->Estatísticas.	Replicar o painel de limites numa nova tab das definições, aumentar o tamanho ou alterar a cor da barra de limites no footer ou mudar o sitio onde esta está colocada.

Figura K.2: Problemas encontrados nos testes de usabilidade (Web)

#	Descrição
1	Erro ao carregar ficheiro "RODA+Plato.graffle" (-220)
2	Botão "Selecionar" no painel de upload não funcionou na segunda vez. Foi preciso alguma insistência.
3	Pesquisa avançada ficou empancada.
4	No painel de importar email, enganou-se no email, testou, nas avançadas ignorou. Ao tentar criar um novo por cima o sistema fica em erro.
5	Ao pesquisar na front page do community há um duplo reload em que se vêm todos os documentos e logo a seguir um reload com o resultado da pesquisa.

Figura K.3: Erros encontrados nos testes de usabilidade (Web)

Apêndice L

Content dissemination on mobile platforms

Content dissemination on mobile platforms

VITOR HUGO CORREIA FERNANDES
Universidade do Minho, Braga

Nowadays, many people use mobile devices in their daily routine. The evolution of mobile market brought a large range of mobile operating systems and devices, each one with different capabilities and different set of supported data formats for images, audio, video, text documents and other types of content. The lack of accordance between the mobile platforms is a barrier to content dissemination. This paper is a survey that explains how the heterogeneous aspects of mobile devices have an impact on content dissemination, proposing solutions to some of the dissemination problems.

Categories and Subject Descriptors: D.4.0 [**Operating Systems**]: General; E.0 [**Data**]: General

General Terms: Management, Perfomance, Standardization

Additional Key Words and Phrases: Content adaptation, Content dissemination, Data formats, Mobile devices, Mobile operating systems

1. INTRODUCTION

The mobile communication has become more present in our daily activities with the advance of new computing technologies, making mobile devices grow on complexity and power. This is one step towards the concept of Ubiquitous Computing, introduced by Mark Weiser in 1991 [Weiser 1991], which is a term used to describe the pervasiveness of technology information in everyday life. The main characteristics of Ubiquitous Computing are focused mainly on the access of computing resources regardless of the location, with the capacity to access information, applications or services anywhere and anytime regardless of the device that is used.

The growing importance of mobile market can be observed by the mobile internet growth. The mobile internet is growing fast, as seen in the Morgan Stanley mobile internet report [Stanley 2009]. Morgan Stanley is a global technology and telecom analyst set out to do a deep dive into the rapidly emerging and changing mobile Internet market. Perhaps the most remarkable statement in this report is that the Mobile Internet market will be "at least 2x size of Desktop Internet in 2014", based on analysis comparing Internet users with mobile subscribers. Growth in the Mobile Internet is being driven by 3G adoption and the increasing popularity of smartphones. The report predicts that smartphones will out-ship the global notebook + netbook market in 2010 and out-ship the global PC market (notebook + netbook + desktop) by 2012. Besides being an optimist report, the evolution of mobile market is pretty natural and visible. This evolution attracted the attention of several international companies, struggling to conquer the mobile market and

Author: Vitor Hugo Correia Fernandes, PG13324, Universidade do Minho
Author's address: Rua Prof. Joaquim Peixoto da Costa, Nr 6-8, 4730-460 Vila de Prado, Portugal
Author's email: vitorfernandes87@gmail.com
Universidade do Minho Master Course on informatics - State of the Art Reports 2010

bringing the development of different mobile operating systems and devices.

A mobile device [1] is a pocket-sized computing device, typically having a small display screen with touch input or a miniature keyboard. In the case of the personal digital assistant (PDA) the input and output are combined into a touch-screen interface. A smartphone [CeVa 2009][Wikipedia 2009] is a telephone that provides additional information accessing features. Any mobile telephone that combines voice services with e-mail, fax, pager or Internet access is called a smartphone. Smartphones and PDAs are popular amongst those who require the assistance and convenience of a conventional computer, in environments where carrying one would not be practical. Any mobile device has a mobile operating system, which is the platform on top of which other programs, called applications programs, can run. It can be abbreviated as mobile OS.

The heterogeneous aspects of mobile devices, as the OS, input and output hardware, have a direct impact on the support of different content types (audio, image, video, text documents and others). That turns the dissemination of content impossible if the used device do not reproduce the content that is being widespread. Any mobile device has limitations caused by hardware or software, which makes impossible, for example, to watch some video format or to display some image format and that is a problem when trying to widespread some kind of content. Another problem is in the reproduction of web contents, because the screen size diverges from one device to another, and the web pages must be dynamic according to the device that it reproduces. The golden rule for web pages size is 950 pixels of size, which, normally, is bigger then the screen resolution of the mobile devices. So there is the need of finding solutions that allow the visualization of web pages done for bigger screens.The rest of the paper will focuses mainly in smartphones because, instead of being commonly used by businessmen like PDAs, they attract a wide variety of people because they provide different business and multimedia programs.

This paper is structured as follows. In Section 2 an analysis to the mobile market is done in order to determinate the most important mobile operating systems. Section 3 will be responsible to show the set of supported formats and physical characteristics of some mobile devices using the most important mobile operating systems. Those characteristics will be discussed in Section 4, determining the recommended formats for dissemination of content. Also in that section, a comparison with the common formats for dissemination is done. In Section 5 some solutions are proposed for formats incompatibilities and content adaptation to the characteristics of the mobile device. Lastly in Section 6 the conclusions about the paper subject will be presented.

Resuming, this work will try to respond why the heterogeneous mobile platforms are an obstacle to the widespread of content and how to avoid it.

2. MOBILE MARKET

Currently, there is an huge amount of different mobile OS, each one with distinct characteristics and features [Hall and Anderson 2009]. Examples are Symbian OS from Symbian Ltd, iPhone OS from Apple, Windows Mobile, from Microsoft, An-

[1] also known as cellphone device, handheld device, handheld computer, "Palmtop" or simply handheld

droid, from Google, RIM BlackBerry OS, Palm WebOS and Palm OS from Palm, among others, most of them based on the open-source operating system Linux. These operating systems are struggling to conquer the mobile market, which is growing as seen in Section 1. The distribution of mobile market will be discussed in this section, analyzing the growth and use.

Figure 1 shows the market share of those operating systems in the 2nd, 3rd and 4th quarters of 2009, based on the reports of Canalys [Canalys 2009a; 2009b; 2009c]. Canalys is the leading global provider of smart phone market data and analysis. The worlds leading hardware and software vendors and mobile operators rely on Canalys for comprehensive market share data, market forecasts, trends analysis and advice.

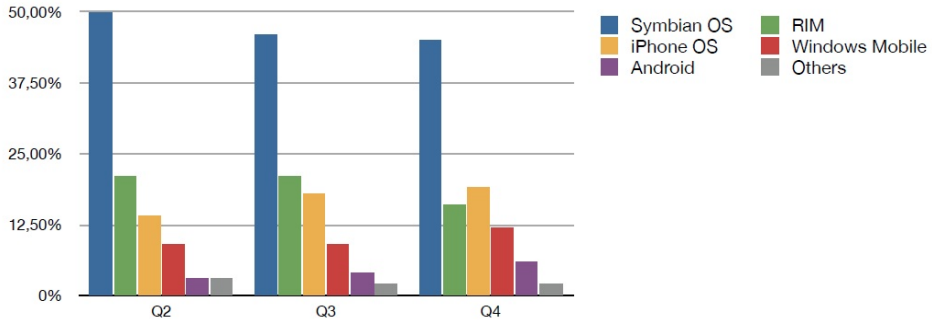

Fig. 1 - Mobile market share in the 2nd, 3rd and 4th quarters of 2009 (Source: Canalys mobile reports [Canalys 2009a; 2009b; 2009c])

The analysis of the Table 1 allows to take some conclusions about the evolution of the mobile market. First, the Symbian OS remains the mobile market leader, despite its market share has reduced. The market share of Symbian OS decreased from 50% in the 2nd quarter to 45% in the 4th quarter of 2009. This reduction is mainly due to the emergence and growth of other operating systems such as the iPhone OS, Android and Windows Mobile. The iPhone OS grew 5% in the same space of time, having in the end of 4th quarter a total share of 19%. Also, at the same time, the Android grew 3%, having 6% of market share. Despite being a small growth, it is important because Android is a recent mobile OS with great prospects for the future. Windows mobile from the 2nd to the 3rd quarter kept the market share, but in the final quarter of the year grew 3%. It had a share of 12% in the end of the year. The RIM BlackBerry market share has reduced in 5%, from 21% in the 2nd quarter to 16% in the end of the year.

The final conclusion is that the others operating systems, such as PalmOS, Palm WebOS (recent mobile OS) or Linux-based operating systems, had a total share of 3%. Their share decreased 1% from the 2nd to the 4th quarter.

As seen in the Table 1 there are five important mobile operating systems (Symbian OS, iPhone OS, Android, Windows Mobile and RIM BlackBerry), each one

with unique characteristics. Symbian OS is the market leader and currently it is used in most recent mobile devices of the major manufacturers, such as Nokia, Samsung or Motorola. RIM dominates the business smartphone market, by selling products easy to use and being very useful in the management of emails and calendar. Nowadays the recent versions of RIM BlackBerry OS have more and better multimedia features. The iPhone OS, developed by Apple, is very popular in the mobile market. It is a well organized mobile OS that provides a clean and easy to use interface. It also provides one simple SDK (Software Development Kit) that allows the development of new programs for it, which are available in the Apple Store. Windows Mobile is a compact operating systems, developed by Microsoft, capable of doing almost everything that is possible in a personal computer with Windows. Normally it comes with basic applications well known like Word, Excel, PowerPoint and Windows Media Player. Last there is the Android OS, which is one operating system based on Linux. Like iPhone OS, it provides a nice interface and simple SDK to the development of new applications. Android as a huge potential to grow.

The analysis of the mobile market movement allowed to check which were the most important mobile operating systems. In the next section, for each important mobile OS one mobile device will be chosen, presenting their characteristics and the set of supported formats.

3. MOBILE DEVICES

In this section the features/characteristics of some mobile device will be presented, in order to verify if the differences between them have an impact of content dissemination. For more information about the formats presented in the rest of the article please see the appendix A - Data Formats.

Tab. 1 - Mobile devices with different mobile OS and characteristics (Source: [Nokia 2009; BlackBerry 2009; Apple 2009; HTC 2009b; 2009a])

	Nokia N70	BlackBerry 8300 Curve	iPhone 3GS	HTC Touch Pro	HTC Dream
Mobile OS	Symbian OS 8.1a	RIM BlackBerry	iPhone OS	Windows Mobile 6.1	Android OS
Screen Resolution/ Size	176x208 pixels, 2.1 inches	320x240 pixels, 2.5 inches	320x480 pixels, 3.5 inches	480x640 pixels, 2.8 inches	320x480 pixels, 3.2 inches
Audio Formats	AAC, AAC+, eAAC, eAAC+, MP3, MP4, AMR-NB, MIDI Tones (poly 64), RealAudio 7, 8, WAV, True tones	MP3, MIDI, AMR-NB, AAC/AAC+/eAAC+, WMA, WAV	AAC, Protected AAC, MP3, MP3 VBR, Audible (formats 2, 3, and 4), Apple Lossless, AIFF, WAV	MP3, AAC, AAC+, WMA, WAV, AMR-NB, MIDI	AAC, AAC+, AMR-NB, MP3, WMA, WAV, AAC-LC, MIDI, OGG
Video Formats	RealVideo, MP4, and 3GP (all with H.263 codec), 176x208 pixels, 25fps	AVI, 3GP, MOV and MP4 (all with H.263 codec, 30fps), WMV (25fps), 320x240 pixels	MP4, M4V and MOV (all with codec H.264), 640x480 pixels, 30 fps	WMV, ASF, MP4, 3GP, 3G2, M4V and AVI (all with H.264 codec), 480x640 pixels, 30 fps	MP4, M4V (with H.263 and H.264 codec) and Windows Media Video 9, 352x288 pixels, 30 fps
Image Formats	BMP, EXIF, GIF87a, GIF89a, JPEG, JPEG 2000, PNG, WBMP	JPEG, PNG, TIFF, GIF, WBMP, BMP	JPEG, TIFF, GIF	JPEG, PNG, GIF, BMP	JPEG (information about this is too limited)
Document Formats	Excel, PDF, Powerpoint, Word, Zip (Viewer for all formats)	Excel, Word, PowerPoint (Viewer for all formats and creator/editor for word format)	Word, Excel, Web pages, Keynote (.key), Numbers (.numbers), Pages, (.pages), PDF, PowerPoint (.ppt e .pptx), Text (Viewer for all formats)	Excel, Word, PowerPoint, PDF (Viewer for all formats and creator/editor for word format)	Excel, Word, PDF, PowerPoint (Viewer for all formats after installing Documents to Go)

The choosing of the mobile devices was based in a report with mobile metrics done by AdMob in October 2009 [AdMob 2009]. This report shows the traffic distribution of the major mobile operating systems. For each mobile OS the device with more traffic percentage is presented in the Table 1. The mobile device using Symbian that had higher traffic percentage was the Nokia N70, with 11.7%. Looking at RIM, the BlackBerry 8300 Curve is the mobile device most used with 44% of traffic percentage. The traffic percentage for Windows Mobile is divided, but the device that had higher percentage is the HTC Touch Pro with 7.2%. Last there is the Android where the HTC Dream was the chosen device because it had 36% of Android traffic percentage. The report did not include the mobile metrics about iPhone OS. All the devices with this operating system have the same manufacturer and there are not many differences between them. The mobile device chosen to represent it was the iPhone 3GS.

Table 2 presents the formats supported by the mobile devices described in the previous paragraph. Only video, audio, text documents and images formats were presented because they are the ones commonly used for disseminations. In addiction to the supported formats, the screen size and resolution is also in Table 1 because the screen is a physical feature that can influence the content dissemination.

In the next section, a comparison between the supported formats will be done, in order to determinate the recommended ones for dissemination.

4. RECOMMENDED FORMATS FOR DISSEMINATION

After analyzing and comparing the information of Table 1 some conclusions about the recommended formats for dissemination can be taken. It is clear that in these days the mobiles devices are quite advanced and support a large range of data formats, but there are still differences between them.

The first conclusion is that the screen is a problem to content dissemination. All the devices presented on Table 1, have different screen sizes and resolutions. This is a problem for the widespread of videos, images and for web contents, because they need to be closely adapted to the screen size and resolution of device that is used.

About documents formats there is no discussion possible. All the devices support Microsoft Office documents (Word, Excel and PowerPoint) and Adobe PDF. Those ones are the formats recommended for document dissemination. Notice that almost all the devices presented on Table 2 only have viewers for that formats. This can be a problem when is wanted to edit/create some kind of document.

Looking at audio formats line of Table 1 there are some formats that are supported by the chosen devices. Those formats are the recommended ones for audio dissemination. The formats recommended are: AAC, MP3 and WAV.

The only image format presented in all the mobile devices presented on Table 1 is the JPEG format. This is the recommended format for image dissemination.

Lastly there are the video formats, where the decision about the recommend for dissemination is harder. The problem about video formats is that they can run at different resolutions, with different codecs and fps (frames per second). All the devices support MP4 videos, but with there are differences between the conditions for the video reproduction. For example Nokia N70 and BlackBerry can run them

with H.263 codec unlike the others that run them with H.264 codec (HTC Dream support the two codecs). Despite that, Nokia N70 has another limitation as it can run MP4 videos with 25fps, instead of the normal 30fps of the others devices. WMV and 3GP are supported by three of the devices, but again there is the problem with resolution, fps and codecs.

Tab. 2 - Common formats and recommended format grouping by content type

	Common formats	Recommended formats
Audio Formats	WAV, MP3, OGG, Flac, WMA, AAC, RealAudio and MIDI.	AAC, MP3 and WAV
Video Formats	3GPP, AVI, MP4, WMV, MPEG, MOV (QuickTime), RealVideo and Shockwave (Flash)	MP4 (25fps, 320x240 pixels, H.264 codec)
Image Formats	JPEG, GIF, PNG, TIFF and BMP	JPEG
Document Formats	Microsoft Office formats (Word, Excel and PowerPoint), Adobe PDF, Zip and HTML files	Microsoft Office formats and PDF.

In Table 2 there is a direct comparison between the common and recommend formats for dissemination.The presentation of the common formats is based on some Internet searches (like [FileInfo 2009]). After analyzing Table 2, it can be noticed that there are more common formats then those ones that are recommended for dissemination. So some of common formats contents cannot be reproduced in some mobile devices (format incompatibilities). There is also a problem with video formats because all videos have different resolutions and fps.

The objective of the next section is to present solution for dissemination problems. Some solutions for format incompatibilities and to content adaptation will be described.

5. SOLUTIONS FOR DISSEMINATION PROBLEMS

Nowadays, there are format incompatibilities in many situations, like receiving an email with some audio or video file format that the device does not support, trying to receive by Bluetooth or Infrared an audio or video file format not supported, watching video from a website, view/edit documents files and many others situations.

One solution to those incompatibilities is data conversion, which consists in the conversion of computer data from one format to another. There are different kinds of converters: audio, video, images and for text documents. Those converters can be applications on the mobile devices or server-side applications. There are many disadvantages for a converter installed on the mobile device, like the CPU (Central Processor Unit) usage and battery drain. Exists specific hardware, normally not presented on smartphones, that turns more feasible the conversion, like video encode/decode processing units. Video conversion is a heavy process for smartphones, because it consumes a large amount of battery and needs a lot of processing from

CPU or from specific hardware. So besides of being a solution to the dissemination problem, the client-side conversion has too many disadvantages to be a perfect solution. Nvidia had developed a system on a chip developed to many types of devices including smartphones, called Tegra 2 [AnandTech 2009], capable of doing video encode/decode because it has two processor to do encode/encode video, but the conversion is still difficult to do in client-side because normally involves larger size files, difficult to handle on a mobile device. On the other hand, there is the server-side conversion. The server can converter some data to another, and send the result file to the mobile device. An application to interact with the user and the server must exist, but it is lighter than a converter on the device because all the data processing is done on the server side. The conversion can create smaller file, optimized for the client platform using less network traffic to receive the file, but can also produce a larger file using then more network traffic. This type of conversion is a better solution to the content dissemination problem than the conversion client-side because it only needs the connection to a network where the conversion service is available and requires low processing from the CPU of the device. The server-side conversion is still a very processor intensive service, possible making this king of service paid or pre-paid. There are some free converters server-side but the result file cannot be obtain in real-time because normally the request for conversion is placed on a waiting queue.

The feature of server-side conversion can be applied, for example, in email servers. Nowadays, the email is one of the most used ways of communication and there are many contents disseminated by email. But some of those contents can be one non supported formats by the device, so conversion is needed. The email server can implement conversion between formats. For example, when a email with a WMV video, BMP image or RealAudio file is received on the iPhone, an option to convert it to a supported format can be a very important improvement to the content dissemination on mobile platforms. If you are using Gmail server you have the option to quick visualization via browser of some kinds of content like PDF, Word (.doc and .docx) and others.

There is also the possibility to install applications to open some kinds of formats that are not support by default by some mobile platform. These applications allow you to open and run files not supported but the file can be displayed in a different way from the original. One example of this file deformation is the viewer of Microsoft Word files in the iPhone, where sometimes the displayed file is from the original. Another solution is use Google Docs which is a free, Web-based word processor, spreadsheet, presentation, and form application offered by Google. It allows users to create and edit documents online while collaborating in real-time with other users. If you are trying to visualize a document file not supported by the device, you can send it to the Google Docs server. Then via browser, if the Google Docs supports the format of the document, you can visualize it. Despite of not being a perfect, it can be an alternative option when trying to visualize an unsupported document format file in the device.

Another problem for the content dissemination is the screen size and resolution. As seen before the mobile devices normally have different screen or different resolutions which makes the widespread of content more difficult because video, images

and web content must be adapted to the size of the device screen. There are many techniques for the content adaptation on mobile platforms. One of them is the URICA (Usage-awaRe Interactive Content Adaptation) [Mohomed et al. 2006], which is an automatic technique that adapts content for display on mobile devices based on usage semantics. URICA allows users who are unsatisfied with the systems adaptation decision to take control of the adaptation process and make changes until the content is suitably adapted for their purposes. The successful adaptation is recorded and used in making future adaptation decisions. A prototype system called Chameleon was implemented to validate URICA, which performs fidelity adaptation on web images. Chameleon functions as an adaptation proxy. A user study has been done where participants used Chameleon to browse image-rich web pages on bandwidth-limited cellular links and used the collected traces to evaluate our system. The result for the user study is that Chameleon reduces the latency for browsing web content by up to 65% and reduces bandwidth consumption by up to 80% and also allows users to exchange bandwidth consumption for user interaction based on their personal preferences". The idea behind URICA is described in [Mohomed et al. 2006; Mohomed et al. 2007; Mohomed et al. 2004].

Another technique is using Really Simple Syndication (RSS) feeds for adaptation of web content in mobile devices. It is based on concrete guidelines and supports different viewing modes, which provides a significant decrease in the transformed content of about 80% in size and, consequently, a low cost-effective web browsing. The RSS feeds are metadata that summarise information. The result of this technique of web content adaptation is a random site, that depends entirely on its content, but all images of the original site are removed, which may not take the full advantage of the capabilities of newest devices. More information about this technique is in paper [Blekas et al. 2006].

There are another techniques to content adaptation, referenced in some papers ([Wei et al. 2009],[Borodin et al. 2007],[Aioffi et al. 2004] and [Samimi et al. 2004]). In [Wei et al. 2009] the design and implementation of a client-centered multimedia content adaptation system suitable for a mobile environment comprising of resource-constrained handheld devices or clients is described. Another system, called CMo, is described in [Borodin et al. 2007]. This system reduces information overload by allowing its users to see the most relevant fragment of the page and then navigate between other fragments if necessary.

6. CONCLUSION

This paper was done in order to answer if the different mobile platforms are a barrier to the widespread of content. There are many mobiles OSs and mobile devices, each one with different capabilities. After analyzing the formats supported by the most used devices according to mobile OS and comparing them to the common formats, the conclusion is that capabilities of the devices and OSs are a barrier to the widespread. There are more common formats then the ones recommended for dissemination, creating format incompatibilities when the used device do not support some format.

There are many solutions to solve those problems, like conversion and emulation. Like all solutions these ones have pros and cons, but they can contribute to a

more efficient and larger widespread of different contents. There are two kinds of conversion: server-side and client-side. Server-side is the ideal option but some conversion services can be paid. Client-side conversion is possible but has too many disadvantages like high CPU usage and battery drain. Sometimes the use of specific hardware (for example for encode/decode of video) turns the conversion more feasible but the conversion client-side still is a heavy process.

Google offers some solutions to visualize some document formats via browser. It is possible to send an email with the document and quick visualize it via browser. Another possibility is Google Docs, a free, Web-based word processor, spreadsheet, presentation, and form application, which allows users to create and edit documents online while collaborating in real-time with other users.

Another problem with mobiles devices is the screen size, which is different from one device to another. The screen can be different in size or in resolution, but nowadays there are techniques to adapt the content at the device screen, like the URICA technique [Mohomed et al. 2006], the RSS feeds technique [Blekas et al. 2006] and many others (referenced in the following papers: [Wei et al. 2009],[Borodin et al. 2007],[Aioffi et al. 2004] and [Samimi et al. 2004]).

Resuming, the content dissemination on mobile platforms is affected by the existence of different mobile OSs and mobile devices because the supported formats changes from one to another. But there are more efficient converters and content adaptation techniques that allows a larger dissemination of some content.

REFERENCES

AdMob. 2009. Admob mobile metrics report. 1–20.

Aioffi, W. M., Mateus, G. R., Almeida, J. M., and Mendes, D. S. 2004. Mobile dynamic content distribution networks. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. ACM, New York, NY, USA, 87–94.

AnandTech. 2009. Nvidia tegra. `http://www.anandtech.com/gadgets/showdoc.aspx?i=3714`.

Apple. 2009. iphone 3gs full specifications. `http://www.apple.com/pt/iphone/specs.html`.

BlackBerry. 2009. Blackberry 8300 full specifications. `http://na.blackberry.com/eng/devices/blackberrycurve8300/curve_specifications.jsp`.

Blekas, A., Garofalakis, J., and Stefanis, V. 2006. Use of rss feeds for content adaptation in mobile web browsing. In *W4A '06: Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A)*. ACM, New York, NY, USA, 79–85.

Borodin, Y., Mahmud, J., and Ramakrishnan, I. 2007. Context browsing with mobiles - when less is more. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*. ACM, New York, NY, USA, 3–15.

Canalys. 2009a. The mobile report q2 2009.

Canalys. 2009b. The mobile report q3 2009.

Canalys. 2009c. The mobile report q4 2009.

CeVa. 2009. Smartphone definition. `http://ceva-dsp.mediaroom.com/index.php?s=glossary`.

FileInfo. 2009. Common file formats. `http://www.fileinfo.com/common.php`.

Hall, S. and Anderson, E. 2009. Operating systems for mobile computing. *JCCSC*, 1–8.

HTC. 2009a. Htc dream full specifications. `http://www.htc.com/www/product/dream/specification.html`.

HTC. 2009b. Htc touch pro full specifications. `http://www.htc.com/pt/product/touchpro/specification.html`.

Mohomed, I., Cai, J. C., Chavoshi, S., and de Lara, E. 2006. Context-aware interactive content adaptation. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*. ACM, New York, NY, USA, 42–55.

Mohomed, I., Cai, J. C., and de Lara, E. 2006. Urica: Usage-aware interactive content adaptation for mobile devices. *SIGOPS Oper. Syst. Rev. 40,* 4, 345–358.

Mohomed, I., Chin, A., Cai, J. C., and de Lara, E. 2004. Community-driven adaptation: Automatic content adaptation in pervasive environments. In *WMCSA '04: Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*. IEEE Computer Society, Washington, DC, USA, 124–133.

Mohomed, I., Scannell, A., Bila, N., Zhang, J., and de Lara, E. 2007. Correlation-based content adaptation for mobile web browsing. In *Middleware '07: Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*. Springer-Verlag New York, Inc., New York, NY, USA, 101–120.

Nokia. 2009. Nokia n70 full specifications. `http://www.forum.nokia.com/devices/N70/`.

Samimi, F. A., McKinley, P. K., Sadjadi, S. M., and Ge, P. 2004. Kernel-middleware interaction to support adaptation in pervasive computing environments. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*. ACM, New York, NY, USA, 140–145.

Stanley, M. 2009. The mobile internet report. 1–104.

Wei, Y., Bhandarkar, S. M., and Li, K. 2009. Client-centered multimedia content adaptation. *ACM Trans. Multimedia Comput. Commun. Appl. 5,* 3, 1–26.

Weiser, M. 1991. The computer for the 21st century. *Scientific American 265,* 3 (September), 66–75.

Wikipedia. 2009. Smartphone definition. `http://en.wikipedia.org/wiki/Smartphone`.

A. DATA FORMATS

Some data formats based on [FileInfo 2009]:

—WAV (WAVEform audio) - Standard audio file format, mainly used in windows, which is commonly used to store uncompressed CD-quality sound files (can be large in size - almost 10MB per minute). Wave files can also be encoded to reduce the size. (.wav extension)

—MP3 (Moving Picture 3) - Is the most popular format for downloading and storing music. It can reduce the size to roughly one-tenth of the original size, maintaining good audio quality. (.mp3 extension)

—Ogg - Free open-source container format supporting a variety of codecs like Vorbis (often compared to MP3 files in quality but less broadly supported). (.ogg extension)

—Flac - Lossless compression codec. Is like Zip but for audio files. Compress an audio file to flac and then restore it, the obtain file is a perfect copy of the original (the others codecs discuss are lossy which means a small part of quality is lost). The cost of this lossless codec is that the compression ratio is not good (it is good for classic music), so it is recommended only when the quality is important. (.flac extension)

—WMA (Windows Media Audio) - Is a popular format owned by Microsoft. It is designed with DRM (Digital Rights Management) abilities for copy protection. (.wma extension)

—AAC (Advanced Audio Coding) - Format base on the MPEG4 audio standard owned by Dolby, and a copy-protected version has been developed by Apple for the music used in their iTune Music Store. (.acc extension)

—RA (RealAudio) - Format design for streaming audio over Internet. This format allow files to be stored in one contained fashion on a computer, where all of the audio are inside of the file. (.ra)

—MIDI - The midi file is just a list of musical notes, that can be play by a synthesizer. It is not a really audio file. (.midi extension)

—3GP - Audio and video container format developed by 3rd Generation Partnership Project (3GPP). It is commonly used by mobile phones that have video capture, because it was designed as a multimedia format for transmitting audio and video files between 3G phones and over Internet. (.3gp and .3g2 extensions)

—AVI (Audio Video Interleave) - Format developed by Microsoft. It is supported by all computers running Windows and by the most popular web browsers, and by that is a very common format on the Internet. Videos in this format have the .avi extension.

—WMV (Windows Media format) - Another format developed by Microsoft. It is a common format on the Internet, but videos in this format cannot be played on non-windows computers without the free extra component installed. The videos in this format have the .wmv extension.

—MPEG (Moving Pictures Expert Group) - This is a format which is the most popular format on the Internet. It is cross-platform, and supported by all the most

popular web browsers. Videos stored in the MPEG format have the extension .mpg or .mpeg.

—MOV - The QuickTime format is developed by Apple. QuickTime is a common format on the Internet, but QuickTime movies cannot be played on a Windows computer without an extra (free) component installed. Videos stored in the QuickTime format have the extension .mov.

—RealVideo - The RealVideo format was developed for the Internet by Real Media. It allows streaming of video with low bandwidths, but with reducing of quality. Videos in this format have the .rm or .ram extensions.

—Flash (ShockWave format) - Developed by Macromedia. It requires an extra component to play. Preinstalled in the latest version of Netscape and IE. Have the .swf extension.

—JPEG (Joint Photographic Experts Group) - One of the most common image formats. Used for photographs, it is a lossy format but with only little differences. It can compress an image into a small size file and retain excellent image quality. (.jpeg, .jpe, .jpg, .jfif and .jfi extensions)

—GIF (Graphics Interchange Format) - Popular Internet format. It is a lossless format ideal for graphics. GIFs can be static or animated, but most of the time GIFs are used for non-photographic type images like buttons, borders and stuff like that. (.gif extension)

—BMP (Windows Bitmap) - Standard Windows image lossless format, that works well for pictures or graphics. It takes up lots of disc space. (.bmp ou .dib extensions)

—PNG (Portable Network Graphics) - Lossless image format, designed to replace the GIF format. It can make transparent images for buttons and icons but with no animations. Images in this formats can have a larger size than normal, so some older versions of web-browsers incorrectly render them. (.png extension)

—TIFF (Tagged Image File Format) - Lossless format that can use compression. It is larger then JPEG but retain all image quality. Normally, the file compressed is half the original size. (.tiff extension)

—Word - Word processing documents created by Microsoft Word. It may contain formatted text, images, tables, graphs, charts, page formatting, and print settings. (.doc and .docx extensions)

—Excel - (.xls and .xlsx extensions)

—PowerPoint - (.ppt, .pptx and .pps extensions)

—PDF - Open standard for documents exchange. It is readable on almost every platform with free or open source readers. Open source PDF creators are also availaible. (.pdf extension)

—Zip - (.zip extensions)

—HTML - (.html and .htm extensions)