Universidade do Minho

David Marques Pires

Criação de interfaces hierárquicas para acesso a redes ontológicas de ficheiros

Tese de Mestrado
Mestrado Integrado em Engenharia de Comunicações
Trabalho efectuado sob a orientação de:
Doutor José Carlos Leite Ramalho
Engenheiro Luís Faria

Dezembro 2011

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 ter permitido a realização da minha dissertação na empresa KEEP SOLUTIONS, e ao meu orientador na empresa, 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 e de agradecimento 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 pais, ao meu irmão, aos meus amigos mais próximos e a outras pessoas especiais na minha vida por todo o apoio e compreensão neste momento da minha vida não só ao longo de toda a dissertação mas também ao longo de toda a vida académica. Foram todos muito importantes e sem eles esta dissertação não seria possível, e por isso, o meu obrigado sincero.

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

Resumo

Os sistemas de gestão documental são cada vez mais comuns, permitindo uma grande vantagem para quem normalmente lida com informação, pois possibilitam um acesso facilitado e rápido a essa informação. Actualmente os sistemas de gestão documental tentam implementar uma organização de dados que consiga reflectir ao máximo as relações que estão inerentes à informação no mundo real. Como se pode verificar no nosso dia a dia a informação usada está repleta de relações, que com o passar do tempo tornam-se cada vez mais complexas, e os sistemas de gestão documental reflectem isso de modo a poder organizar a informação do modo mais fiel e completo possível. Os sistemas de gestão documental mais completos armazenam os documentos e também as relações a eles inerentes, usando para isso uma organização ontológica. Estas organizações de dados (ontológicas) conferem relações qualitativas entre os documentos.

Apesar das vantagens no uso de estruturas ontológicas, os utilizadores estão mais familiarizados com as estruturas hierárquicas tornando-se intrínseco e inseparável à experiência do computador pessoal. No entanto, as estruturas hierárquicas não permitem ter relações qualitativas entre os documentos, o que acontece nas estruturas ontológicas.

É neste contexto que esta dissertação é realizada, descrevendo como é feita a transformação duma organização de dados não hierárquica numa hierárquica e os possíveis problemas que possam aparecer na interligação das operações realizadas sobre os dados.

Abstract

The systems of document management are becoming more and more common, being a great advantage for those who usually have to deal with information, since they enable accessing that same information in an easy and quick way. Nowadays, the systems of document management are trying to implement a data organization that is able to reflect the relationships inherent to information in the real world as much as possible. As it is possible to see in our daily life, the information used is full of relationships, which become more complex as time goes by, and the systems of document management reflect that in order to enable to organize information a way that is as faithful and complete as possible. The most complete systems of document management store both documents and the relationships inherent to them by using an ontological organization. These data organizations (ontological) enhance qualitative relationships between documents.

Despite the advantages of using ontological structures, the users are more familiarized with the hierarchical structures, thus becoming intrinsic to and inseparable from the experiment of the personal computer. Nevertheless, the hierarchical structures do not enable qualitative relationships between documents, as it happens with ontological structures.

It is within this realm that this dissertation is made, describing how the transformation of a non-hierarchical data organization into a hierarchical one is made as well as the possible problems that can appear in the interconnection of the operations made on data.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Enquadramento

Um sistema de ficheiros organizado numa hierarquia de pastas é um conceito comum a todos os utilizadores de computador, tendo entrado no dia-a-dia dos utilizadores e permitindo uma habituação a este sistema, tornando-se intrínseco e inseparável à experiência do computador pessoal. Existem diferentes sistemas de ficheiros, sendo o formato nativo de arquivo diferente em cada sistema operativo, o Windows usa FAT (*File Allocation Table*) e mais recentemente o NTFS (*New Technology File System*), o Unix usa Ext2 (*Second Extended filesystem*), Ext3 (*Third Extended filesystem*), Ext4 (*Fourth Extended filesystem*), o MacOS usa HFS (*Hierarchical File System*) [20].

Sistema de ficheiros é um mecanismo de armazenar e organizar arquivos de computador e seus dados, sendo responsável pela gestão do acesso e da representação interna dos ficheiros. O sistema de arquivos determina como as informações podem ser guardadas, acedidas, copiadas, alteradas, nomeadas e apagadas.

1.1.1 Sistema de armazenamento

A necessidade de armazenamento varia de acordo com a aplicação pretendida, por isso é ideal ter os fundamentos de cada topologia para ajudar na escolha da arquitectura mais apropriada. A forma como os servidores e os dispositivos estão ligados varia de uma arquitectura para outra. Exemplos de sistemas de armazenamento são *Direct Attached Storage* (DAS), *Storage Area Network* (SAN), *Network-Attached Storage* (NAS) e *Content addressable storage* (CAS).

Direct Attached Storage

O sistema *Direct Attached Storage* (DAS) é um meio dedicado para conectar sistemas. A conectividade é a principal limitação do DAS, os dispositivos de armazenamentos estão ligados directamente a um computador que irá gerir e criar o sistema de ficheiros (sistemas de ficheiros são abordados na secção 1.1.2). Esta topologia é mais adequada para pequenos servidores que oferecem acesso a arquivos para diversos PCs ou utilizadores de um pequeno escritório. É uma forma confiável para armazenar aplicações que são executados num servidor. No entanto, esta arquitectura tem algumas limitações em relação à escalabilidade, uma vez que os servidores suportam um número limitado de discos [5]. Na figura 1.1 podemos ver um exemplo duma topologia DAS.

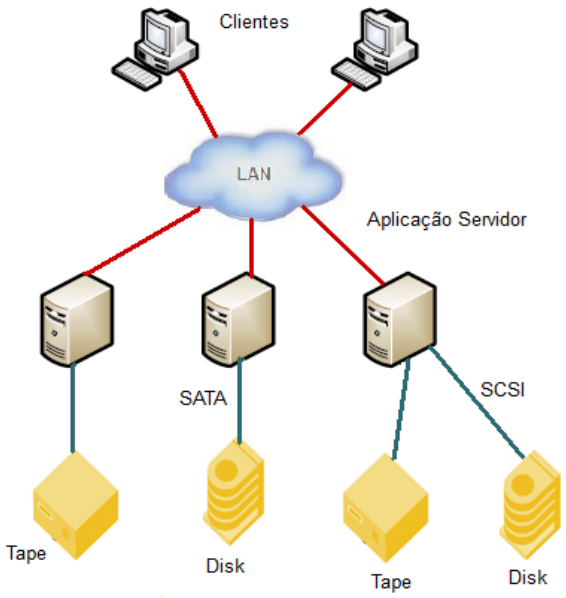

Figura 1.1: Exemplo duma topologia DAS [5]

Storage Area Network

O sistema *Storage Area Network* (SAN) é um sistema de arquivos compartilhado em disco onde o computador é o responsável pela gestão e criação do sistema de ficheiros (sistema de ficheiros secção 1.1.2). Trata-se duma rede (ou sub-rede) de alta velocidade que interliga servidores e dispositivos de armazenamento, utilizando fibra óptica para transferências a grandes velocidades, garantindo eficácia na realização de *backups*, permitindo a transferência de dados dum disco para outro, sem a intervenção do servidor. Ele consiste numa colecção de *Hardware* SAN, que

normalmente tem alta taxa de interconexão entre vários dispositivos de armazenamento e *Software* SAN que faz a gestão, monitorização e configuração do SAN. A arquitectura usada pelo SAN permite que todos os dispositivos de armazenamento fiquem disponíveis para todos os servidores, permitindo também o acesso aos dispositivos de armazenamento através de qualquer servidor da rede. Neste caso, o servidor simplesmente age como um caminho entre o utilizador final e os dados armazenados [5]. Na figura 1.2 é demonstrada um exemplo duma topologia SAN.

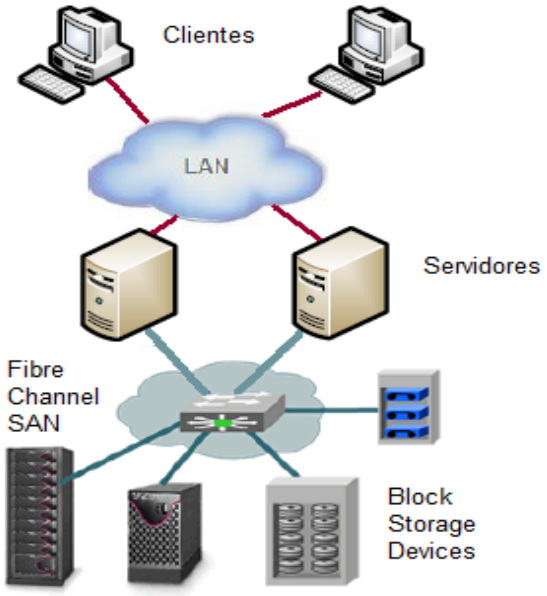

Figura 1.2: Exemplo duma topologia SAN [5]

Network-Attached Storage

O sistema *Network-Attached Storage* (NAS) é um dispositivo de armazenamento que tem um endereço próprio de rede e usa um sistema de ficheiros com estrutura hierárquica, funcionando basicamente com um servidor ligado à rede, tendo como única funcionalidade promover serviços de armazenamento de dados para outros dispositivos da rede. Permitem aumentar a capacidade de armazenamento dos servidores, tem baixo custo e performance moderada e permite também preservar a rede existente. A sua capacidade de armazenamento depende da capacidade dos discos e a memória é organizada através de partilha de arquivos, o que significa que o acesso aos dados só é possível ao nível do sistema de arquivos [5]. Na figura 1.3 é demonstrado um exemplo duma topologia NAS.

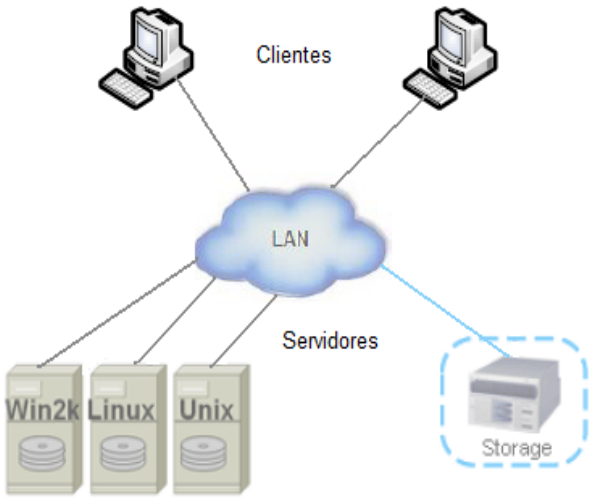

Figura 1.3: Exemplo duma topologia NAS [5]

Content addressable storage

O sistema *Content Addressable Storage* (CAS) é um método que permite o acesso rápido ao conteúdo fixo, atribuindo ao conteúdo um lugar permanente no disco, não necessitando de um sistema de ficheiros, conseguindo ter acesso aos conteúdos mesmo em estruturas não hierárquicas. O CAS faz a recuperação directa dos dados, armazenando-os de forma a que um objecto não possa ser repetido ou modificado depois de ter sido armazenado, assim a sua localização é inequívoca. Este sistema de armazenamento é baseado não no endereço local, mas no endereço do conteúdo. Quando um CAS é solicitado a armazenar um arquivo, ele calcula um *hash* [1] com base no seu conteúdo, retornando-o como endereço do conteúdo armazenado e prossegue para o armazenamento físico do conteúdo. O endereço *hash* é então usado como a chave para recuperar o conteúdo [2].

No sistema CAS os dados são tratados como objectos, onde terão associados a si um código de identificação (ou seja, um endereço de conteúdo) e armazenados num espaço livre no disco, sem levar em consideração o seu conteúdo. Esta noção de guardar a informação em qualquer espaço livre do disco permite ao CAS fugir ao tradicional armazenamento de dados que consiste numa hierarquia de dados, não precisando de se preocupar com a estrutura de directórios ou nomes de arquivo, tornando-se assim num sistema de armazenamento de dados com uma estrutura não hierárquica. Em outras palavras, quando se pensa no armazenamento, tudo tem um

[1] é uma sequência de bits geradas por um algoritmo de *hash*, em geral representada em base hexadecimal. A função de *hash*, a partir de uma cadeia de caracteres (string) de qualquer tamanho, cria uma string de tamanho fixo.

endereço enquanto que nos sistemas de arquivos convencionais, é um nome e um local numa hierarquia de directórios. O CAS ao guardar um novo conteúdo anexa-lhe um conjunto de metadados, entre os quais os metadados já guardados pelos arquivos tradicionais como por exemplo o nome e as datas. Além destes atributos usuais, o CAS adiciona mais metadados que podem ser considerados de qualidade para a pesquisa de um documento, por exemplo, informações contextuais que poderão ajudar o utilizador a entender ou utilizar os dados quando o documento é consultado. A operação do cálculo de *hash* não é apenas realizada sobre o conteúdo do documento mas também sobre os metadados adicionados ao documento. Esta operação de calcular o *hash* sobre os metadados traz uma mais valia ao sistema CAS, por exemplo, se pretender-mos arquivar 20 e-mails com o mesmo anexo, num sistema de *backup* ou replicação tradicional, o que aconteceria era arquivar os 20 e-mails. Com o CAS apenas será guardado uma vez o anexo, pois todos os valores de *hash* do conteúdo são iguais, e os restantes serão guardados apenas os metadados e um ponteiro para o conteúdo. Uma vantagem deste método é o facto de minimizar o espaço de armazenamento consumido por *backups* dos dados e arquivos, pois sempre que são adicionados novos ficheiros é verificado se já existe o *hash* do conteúdo e o *hash* dos metadados [14].

1.1.2 Sistema de ficheiros

Os sistemas de ficheiros, têm como objectivo organizar e guardar a informação (dados) no disco rígido, de modo a ser possível a sua localização, manter estruturas de dados que representem a informação (os ficheiros) existentes num dado momento, localizar e dar acesso à informação, quando os programas invocam as chamadas ao Sistema Operativo, e gerir, isto é, reservar e libertar o espaço físico ocupado em disco, pelas representações dos ficheiros. Diferentes sistemas operativos usam diferentes sistemas de ficheiros, actualmente o Windows usa o FAT32 e mais recentemente o NTFS, o Unix usa normalmente o Ext e o MAC usa o HFS.

FAT32

FAT é a sigla para *File Allocation Table*, é um sistema de arquivos que funciona com base numa espécie de tabela que indica onde estão os dados de cada arquivo. Esse esquema é necessário porque o espaço destinado ao armazenamento é dividido em blocos, e cada arquivo gravado pode ocupar vários blocos, mas não necessariamente de maneira sequencial, os blocos podem estar em várias posições diferentes. O FAT32 é um derivado do FAT, criado em 1996 pela Microsoft para substituir o FAT16 usado pelo MS-DOS e com uma série de limitações. Porém possui compatibilidade com sistemas operativos relativamente novos como o Windows XP. Possui um espaço de disco utilizável maior, pois tem um sistema de formatação mais moderno, mais confiável, pois consegue posicionar directórios principais em qualquer

lugar do disco, suporta partições de até 2 TB, e tamanho de 4 GB por arquivo e 256 caracteres, superando seu antecessor, é possível mudar o tamanho da partição.

O FAT32 sendo um descendente do FAT, herdou a organização de dados já usada, consistindo numa tabela de alocação de arquivos e baseando-se numa organização hierárquica de arquivos.

NTFS

O sistema de ficheiros NTFS foi criado pela Microsoft para solucionar as limitações e a falta de recursos do FAT. As limitações do FAT, principalmente quanto à segurança, capacidade e confiança, fizeram do FAT um sistema de arquivos inadequado para uso em servidores e aplicações críticas. Devido a estas limitações do FAT a Microsoft decidiu desenvolver um sistema de arquivos que ultrapassasse estes problemas, o NTFS. Entre os objectivos do NTFS estavam o fornecimento de um sistema de arquivos flexível, adaptável, altamente seguro e confiável. O NTFS aceita volumes até 2 TB, o tamanho do arquivo é limitado apenas pelo tamanho do volume, oferece suporte a compactação, criptografia e indexação, é um sistema de arquivos muito mais seguro, permite políticas de segurança e gestão. Os volumes NTFS são menos vulneráveis a fragmentos e têm melhor desempenho. Apesar das melhorias introduzidas pelo NTFS ele ainda usa o velho estilo de organização hierárquica, baseando-se numa estrutura em árvore.

Ext

O sistema de ficheiros mais usado no sistema operativo Linux é o Ext, que já se encontra actualmente na versão Ext4. O Ext2 é o sistema de ficheiros mais antigo ainda em uso devido a ser muito robusto, rápido e confiável, sendo usado para dispositivos de blocos (disco rígidos, disquetes, *pen drivers*). O Ext3 é basicamente o sistema de ficheiros Ext2 mas com recursos de *Journaling*, sendo totalmente compatível com o Ext2. O *journaling*, consiste num registo (log ou *journal*) de transacções cuja finalidade é recuperar o sistema em caso de falha não programada. O Ext4 foi desenvolvido para ser o sucessor do Ext3, tendo como novas funcionalidades propostas a alocação tardia (*delayed allocation*), verificação de integridade do *journal* (*journal checksums*), suporte para tamanhos maiores de volumes e arquivos, suporte a mais extensões, compatibilidade com versões anteriores, pré alocação, verificação mais rápida, alocação multibloco, melhores *timestamps*. O sistema de ficheiros Ext para a organização de dados usa a tradicional organização hierárquica de arquivos, mais especificamente uma estrutura em árvore.

HFS

É um sistema de arquivos desenvolvido pela Apple Computer para uso em computadores que usem o Mac OS. Originalmente projectado para uso em *floppy* e discos rígidos, ele também pode ser encontrado em suporte *read-only* como CD-ROMs. Mais tarde, a Apple, Inc. criou o sistema de ficheiro HFS+ para substituir o antigo HFS como sistema de arquivos primário usados em computadores Macintosh. O HFS+ é uma versão melhorada do HFS, sendo os arquivos de suporte muito maiores (endereços de blocos são de 32 bits de comprimento em vez de 16 bits) e usando *Unicode* para nomear os itens (arquivos, pastas). O HFS+ oferece ao utilizador a possibilidade de formatar a unidade desejada com um registo cronológico. Com esta possibilidade, o sistema, sempre que o computador for reiniciado indevidamente ao invés de procurar em todos os arquivos e pastas as informações, apenas necessita verificar as últimas alterações que ocorreram no disco rígido através desta tabela.

GFS

Google File System (GFS) é um sistema de arquivos escalável para aplicações de distribuição intensiva de dados. Os arquivos são organizados hierarquicamente em directórios e identificados por *path-names*. A Google utiliza o GFS para organizar e manipular grandes arquivos e permitir que as aplicações consigam usar os recursos necessários. Existem diversas aplicações que usam estes dados: YouTube, Google Earth, Blogger, GMail, Orkut, Google Maps, Google Sugest, Google Desktop Search. O GFS garante que, para cada arquivo, são armazenados pelo menos três cópias em diferentes computadores num *cluster* de servidor. Isso significa que se um programa dum computador tentar ler um arquivo dum desses computadores e não obter resposta dentro de alguns milissegundos pelo menos outros dois serão capazes de atender a solicitação.

HDFS

Hadoop Distributed File System (HDFS) é um sistema de arquivos distribuído para actuar em dados não estruturados e é baseado na tecnologia GFS, criando réplicas de blocos de dados que são distribuídos no *cluster* para permitir computações extremamente rápidas. Um *cluster* HDFS consiste basicamente num *NameNode* que faz a gestão do sistema de arquivos e metadados *DataNodes* que armazenam os dados reais. O HDFS é tolerante a falhas e foi projectado para ser implementado em *hardware* de baixo custo, fornecendo também acesso aos dados da aplicação de transferência e é adequado para aplicações que têm grandes conjuntos de dados. O HDFS suporta uma organização de arquivos hierárquica tradicional. Um utilizador

ou uma aplicação pode criar directórios e arquivos de armazenamento dentro desses directórios. O arquivo de hierarquia do sistema *namespace* é semelhante à maioria dos outros sistemas de arquivos existentes, sendo possível criar e remover arquivos, mover um arquivo dum directório para outro ou mesmo renomear um arquivo.

1.1.3 Estrutura de dados

O sistema de arquivo tem a necessidade de usar uma organização de dados para a manipulação dos arquivos (pastas e documentos). Uma organização de dados é um modo particular de armazenamento de dados para serem usados eficientemente. A organização dos dados pode usar várias estruturas diferentes, como por exemplo estrutura em árvore, grafo, ontologias.

Estrutura em árvore

A estrutura em árvore é composta por um elemento principal designado de raiz, que possui ligações para outros elementos, que são denominados de ramos ou filhos. Estes ramos levam a outros elementos que também possuem outros ramos. O elemento que não possui ramos é conhecido como folha. A pesquisa dos dados na estrutura em árvore é chamado de travessia da árvore, pois a pesquisa é feita através da travessia da raiz até um dos ramos ou folhas. O custo computacional desta operação é proporcional ao número de níveis da árvore e só existe um caminho dum nó para o outro.

Os sistemas de arquivos mais populares (como o Ext3, NTFS e HFS) usam uma estrutura em árvore, onde os arquivos estão dispostos duma forma hierárquica. Na figura 1.4 é apresentado um exemplo duma estrutura em árvore.

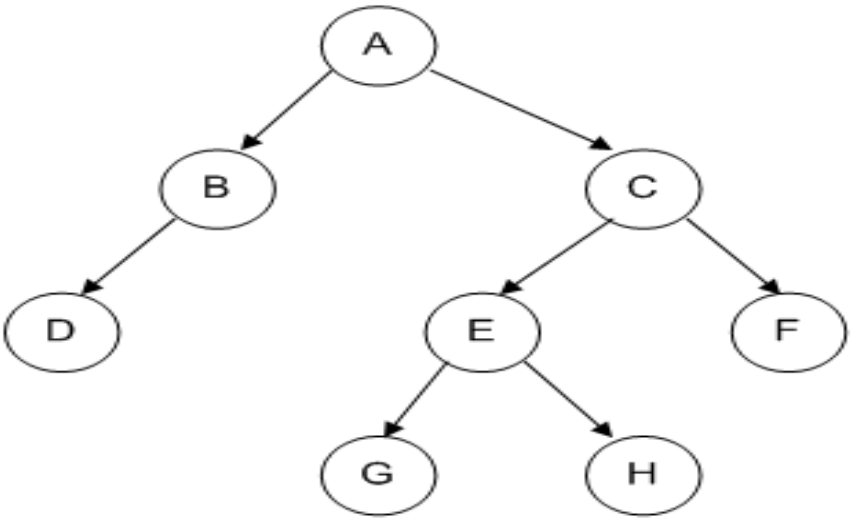

Figura 1.4: Estrutura em árvore

Estrutura em grafo

A estrutura em grafo é representado com um conjunto de elementos (nós ou vértices) que se relacionam de alguma maneira (arestas). Esta estrutura pode representar toda a informação duma árvore. O directório é apenas um repositório de elementos de acesso para os arquivos e outros directórios, permitindo arquivos e sub-directórios compartilhados, sendo possível ter múltiplos pontos de acesso para um arquivo/directório. As referências ao arquivo/directório são feitas por ligações (*links*) simbólicos, que apontam para o destino final. A estrutura em grafo é mais flexível que uma em árvore mas mais complexa podendo trazer alguns problemas, por exemplo, arquivos com vários caminhos podem fazer com que o sistema percorra a estrutura mais que uma vez, a eliminação dum arquivo pode deixar *links* perdidos, podendo ser muito grave se o *link* estiver a apontar para endereços físicos. Na figura 1.5 é apresentado um exemplo duma estrutura em grafo.

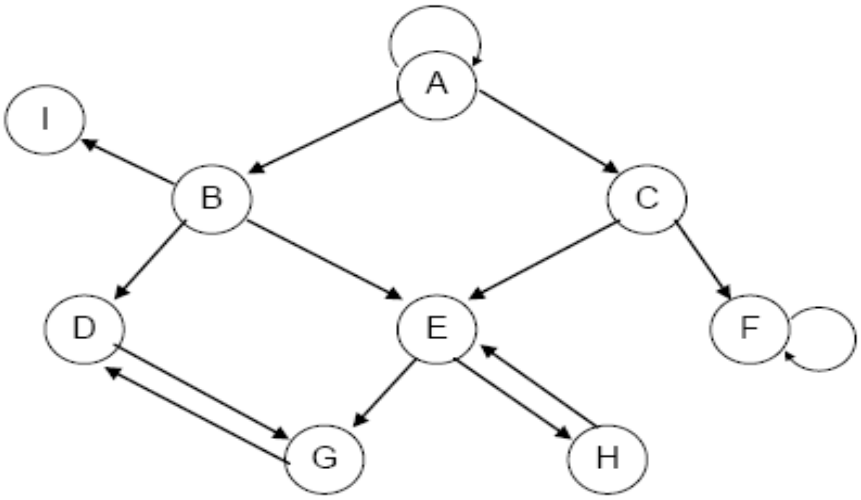

Figura 1.5: Estrutura em grafo

Estrutura ontológica

Estrutura ontológica é um conjunto de termos cuja finalidade é padronizar a manutenção das informações de forma compartilhada e reutilizável por diferentes utilizadores, não é uma estrutura hierárquica mas sim associativa. Ontologia pode representar toda a informação dum grafo, definindo relações qualitativas entre os nós. A principal característica duma ontologia é tornar explícita a informação de maneira independente das estruturas de dados subjacentes. Ontologias são abstracções e podem descrever diferentes tipos de organização de dados como tabelas relacionais, textos e imagens, estando mais voltada para o sentido das relações do que estruturas hierárquicas. Uma ontologia define termos para que um agente de *software* consiga extrair o máximo de informação possível dum documento.

Dentro da estrutura ontológica surgiu outro tipo de estrutura, a estrutura semântica. O uso de estruturas semânticas veio tentar revolver algumas limitações do sistema de arquivos mais popular (sistema de arquivos em árvore), como a impossibilidade de armazenar um arquivo em mais dum directório sem que exista duplicidade do mesmo arquivo e a possibilidade de retirar informação nos arquivos. Resumidamente o sistema de arquivos semânticos permite que os utilizadores realizem buscas no sistema de arquivos baseando-se nos atributos dos mesmos, possibilitando também a extracção automática de atributos (ou *Tags*) a partir dos metadados, como por exemplo o autor, a data de criação, a data de modificação. Além dos metadados também é possível extrair atributos a partir de ficheiros de texto, como por exemplo palavras que fazem parte do seu conteúdo. Um exemplo deste sistema é o *Windows Future Storage* (WinFS) [44], que foi desenvolvido pela Microsoft, e que inclui o arquivamento de *tags* com metadados. Neste momento o WinFS ainda se encontra

em desenvolvimento, mas segundo a Microsoft, o Sistema de Arquivos WinFS, será lançado no Windows 8, em 2012 [43]. Na figura 1.6 podemos verificar um exemplo duma estrutura ontológica.

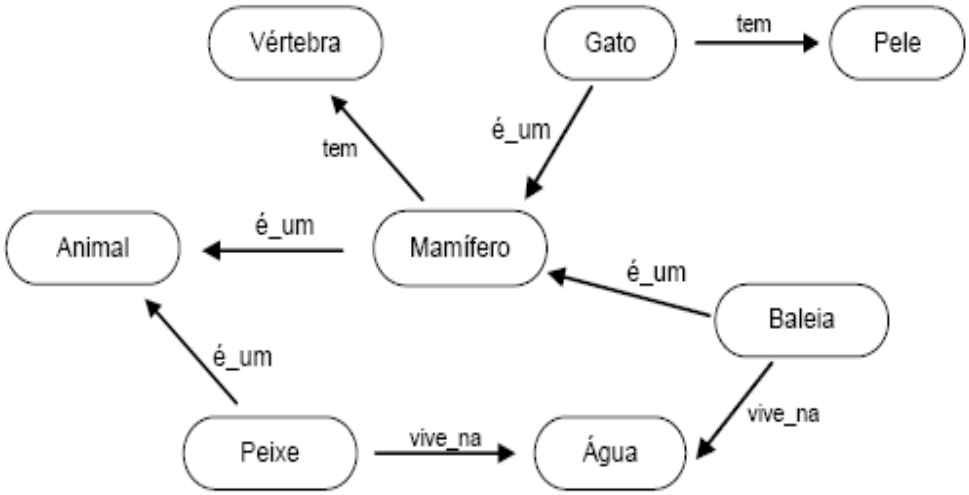

Figura 1.6: Exemplo duma estrutura ontológica [41]

1.1.4 Sistema de Gestão Documental

Sistema de Gestão Documental tem vindo a ganhar muito relevo nos últimos tempos devido às suas vantagens no acesso facilitado à informação, rapidez de acesso, controlo da informação, segurança, qualidade e fiabilidade da informação, redução de custos, entre outros. Um Sistema de gestão documental permite o arquivo e organização de documentos em formato electrónico, disponibilizando mecanismos que permitem o controlo do seu estado bem como da sua localização em cada momento. A Gestão Documental é uma solução de arquivo, organização e consulta de documentos em formato electrónico onde existe toda a informação de natureza documental trocada entre os utilizadores da aplicação. Esta solução permite a colaboração entre vários utilizadores e a partilha de documentos.

Os sistemas de gestão documental tendem a apresentar a informação de uma forma mais rica, como por exemplo em estruturas semânticas. Uma característica chave do uso de redes semânticas é a implementação de associações que podem ser feitas explicitamente ou sucintamente, facto relevantes sobre o objecto ou conceito podem ser inferidos dos nodos com os quais eles estão ligados, sem uma busca através da ampla base de dados. Com a implementação de redes semânticas nos sistemas de gestão documental consegue-se uma maior flexibilidade, inteligibilidade, simplicidade e herança. Com as vantagens apresentadas para o sistema de gestão

documental verifica-se um crescimento na venda de sistemas de gestão documental, figura 1.7.

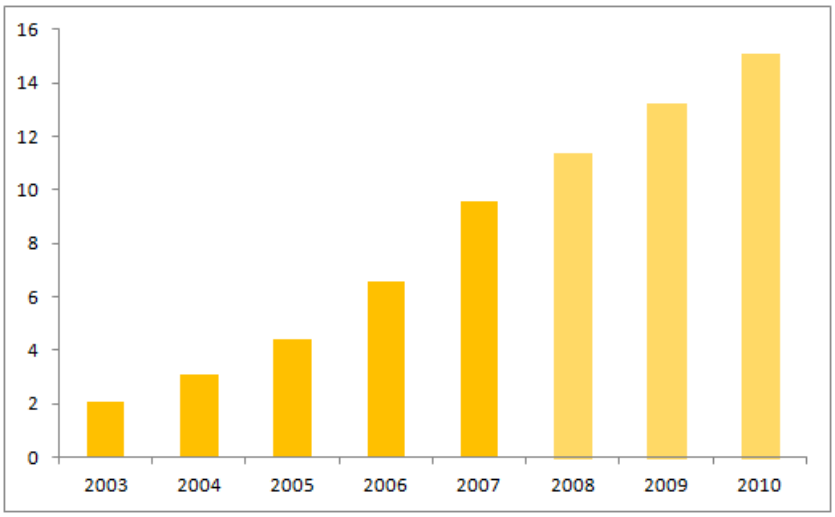

Figura 1.7: Vendas de sistemas de gestão documental entre 2003 a 2007 [19], e projecção até 2010

1.2 Objectivos

A existência de várias formas de organização de dados usados nos sistemas de ficheiros, veio permitir a possibilidade de guardar a informação com uma organização mais rica, como numa estrutura ontológica e não apenas numa estrutura hierárquica tão conhecida pelos utilizadores dos computadores. O uso de estruturas hierárquicas sendo usado há muito tempo permitiu que os utilizadores se familiarizassem com o conceito de ficheiros organizados numa hierarquia de pastas. Esta dissertação tem como objectivo apresentar uma solução para a criação de interfaces hierárquicas para acesso a redes ontológicas de ficheiros. Para a resolução deste problema é preciso fazer uma modelação de uma estrutura ontológica para uma estrutura em árvore. Esta solução será utilizada num caso de estudo para comprovar a sua eficácia, detectar problemas de implementação, e posteriormente realizar testes de usabilidade na solução.

1.3 Estrutura da dissertação

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

No Capítulo 2 será apresentado uma solução para a modelação duma estrutura ontológica para uma estrutura hierárquica. Esta solução visa permitir fazer uma introdução a uma possível forma de resolver o problema da modelação.

No Capítulo 3 é feita a análise ao caso de estudo, que consiste em adaptar um sistema que usa um modelo de dados ontológico num sistema de dados hierárquico. A aplicação escolhida para ser adaptada é um sistema de gestão documental baseado numa estrutura de dados ontológica e o protocolo escolhido para a construção da estrutura em dados hierárquico é um sistema de arquivo em rede. Serão também apresentadas as funcionalidade do sistema disponibilizadas aos utilizadores. Neste capítulo é também explicado como foi realizada a interligação entre os dois sistemas utilizados para o problema, devido a usarem operações diferentes para algumas tarefas sobre as pastas e ficheiros.

No Capítulo 4 são apresentados alguns testes realizados ao sistema desenvolvido. Com a realização dos testes de usabilidade é possível fazer uma análise ao sistema permitindo identificar algumas limitações e problemas apresentados pelo sistema.

No Capítulo 5 é elaborado um resumo relativo ao trabalho realizado na dissertação, expondo os problemas que possam surgir na criação de interfaces hierárquicas para acesso a redes ontológicas de ficheiros bem como dos problemas que possam acontecer nas tarefas de manipulação de dados.

Capítulo 2

Solução Proposta

A modelação de uma estrutura ontológica numa estrutura hierárquica visa potencializar a organização das pastas devido ao acrescido valor da metainformação, tornando possível que existam várias visões do repositório em forma de sistema de ficheiros. A mais evidente será reflectir a organização original, caso essa informação exista.

A solução proposta representa uma modelação genérica duma estrutura onde existe informação nas relações entre os conceitos (estrutura ontológica, figura 2.1), e a estrutura que é mais conhecida e usada pelos utilizadores (estrutura em árvore).

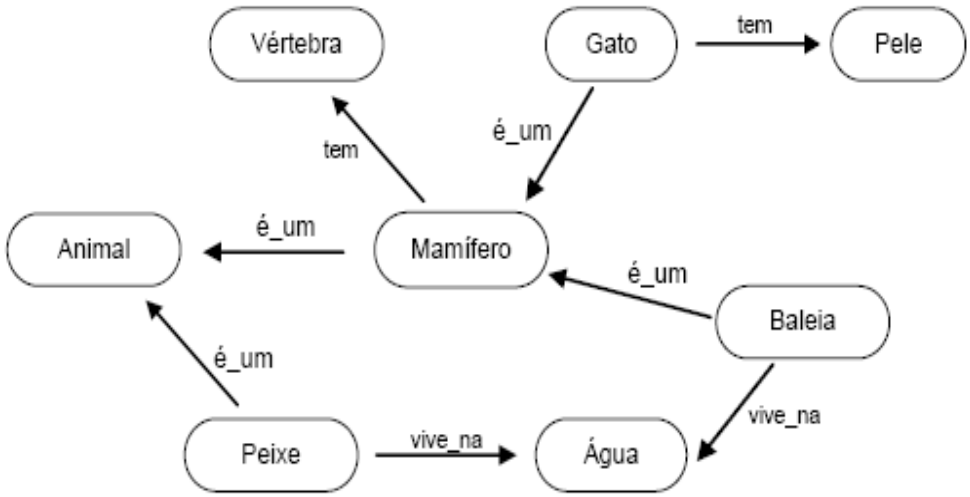

Figura 2.1: Exemplo duma estrutura ontológica [41]

Para a realização da modelação foi criado uma algoritmo que consiste em descrever os passos necessários para a modelação duma estrutura ontológica numa

estrutura hierárquica.

Algoritmo para a criação de interfaces hierárquicas para acesso a redes ontológicas de ficheiros

1. Criar um nó especial, denominado raiz

2. Criar nível com as arestas

3. Criar nível seguinte com os conceitos, que estão relacionados com as arestas

    3.1 Se necessário duplicar conceitos

4. Criar nível seguinte seguindo a transitividade entre conceitos, através das arestas

    4.1 Se necessário duplicar conceitos

5. Repetir o passo 4 até a hierarquia estar completa

O primeiro passo realizado foi definir uma hierarquia de pastas, para isso foi definido que o nó raiz seria "/", as arestas o primeiro nível de pastas e os conceitos os níveis seguintes. A construção dos níveis para a estrutura hierárquica seguiu a transitividade das relações que existem entre os conceitos, exemplo: Baleia é um Mamífero, Mamífero é um Animal, o que numa estrutura hierárquica e seguindo o algoritmo descrito em cima se irá traduzir no 1º nível da hierarquia "/", 2º nível "é_um", 3º nível "Baleia", 4º nível "Mamífero" e no 5º nível "Animal". Com a construção dos níveis hierárquicos foi preciso fazer cópias dos conceitos, pois eles estavam relacionados com outros conceitos. A cópia dos conceitos é necessária para a solução final para garantir que nenhuma informação seja perdida e ao mesmo tempo que esteja relacionada com os conceitos que estavam na estrutura ontológica. Como se pode verificar na figura 2.2 a pasta "Água" tanto é um descendente da pasta "Peixe" como da pasta "Baleia". A cópia dos conceitos deve-se ao facto que numa estrutura hierárquica em árvore, os ramos ou filhos não podem estar associados a dois ascendentes simultaneamente.

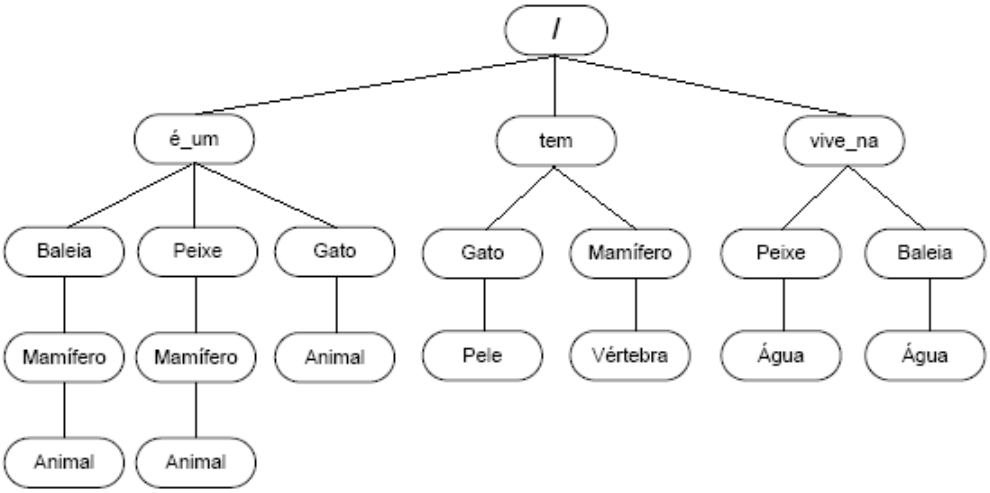

Figura 2.2: Estrutura proposta para a modelação de uma estrutura semântica para uma estrutura hierárquica

Capítulo 3

Caso de estudo

Neste capítulo será feito o desenvolvimento de um caso de estudo prático de modelação de uma estrutura de ficheiros não hierárquica numa estrutura hierárquica. O caso de estudo irá comprovar a sua eficácia, detectar problemas de implementação, e estudar a usabilidade da solução. Na primeira secção será identificada a aplicação base para a adaptação a um sistema hierárquico e o sistema de arquivo em rede escolhido para este caso de estudo, sendo feita uma descrição para cada um dos sistemas. Posteriormente será apresentada a adaptação da estrutura da dados implementada. Por fim serão abordadas as funcionalidades do sistema e a respectiva caracterização, bem como os possíveis problemas que possam surgir nessas operações.

3.1 Objecto de estudo

O objecto de estudo assenta no desenvolvimento dum sistema de arquivo em rede que usa uma estrutura hierárquica para a organização dos dados baseado na informação contida no weebox, que é uma aplicação desenvolvida pela KEEP SOLUTIONS. O weebox é um sistema de arquivo de informação digital e uma plataforma de gestão documental, garantindo o seu armazenamento seguro. A organização dos dados do weebox consiste numa estrutura não hierárquica, estrutura ontológica, onde são permitidas relações entre os conceitos através do uso de Vocabulários Controlados [1]. Resumidamente, considera-se que usa uma estrutura que se encontra entre uma estrutura ontológica e uma estrutura semântica, onde toda a informação é definida por identificadores. A sua unidade de armazenamento é o *bundle*, que é composto por metadados genéricos (chave-valor), um conjunto de

[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.

ficheiros, permissões e relações com outros *bundles*. O weebox é ideal para o caso de estudo, pois permite cumprir os objectivos deste trabalho que consistem na conversão dum sistema de organização de dados ontológico para um sistema de organização de dados hierárquico e a interligação dos passos realizados por cada sistema para as operações sobre as pastas e ficheiros.

3.1.1 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 [31]. 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 [32].

3.1.2 weebox

O 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 armazenamento 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 funcionalidades distintas, mas completamente integrado nos restantes módulos. A organização em módulos permite adequar a solução às necessidades do cliente [8].
O sistema é composto por três componentes funcionais:

- weebox Core Services
- weebox Manager
- weebox Community

Um conceito muito importante usado pelo weebox é o do *bundle*. É no *bundle* que se encontram guardados os metadados dos ficheiros.

Nos metadados existem atributos obrigatórios para ser possível adicionar pastas no weebox. Entre esses atributos alguns são adicionados pelo utilizador e outros que são preenchidos pelo sistema. Entre os atributos obrigatórios a ser inseridos pelo utilizador encontram-se o tipo de documento, que é definido por uma cor que lhe está associada, e o título. O autor da criação da pasta e as datas relativas à criação da mesma, submissão e modificação são preenchidas pelo sistema. Encontram-se também outros campos mas não são de preenchimento obrigatório para ser possível arquivar a pasta, figura 3.1, campos obrigatórios assinalados por (*).

No *bundle* estão associados os ficheiros que irão ser armazenados na pasta, dando possibilidade ao utilizador de poder verificar a informação dos mesmos, bem como a possibilidade de fazer o *download* destes, descarregar ficheiros individualmente ou mesmo descarregar os ficheiros todos existentes na pasta ao mesmo tempo.

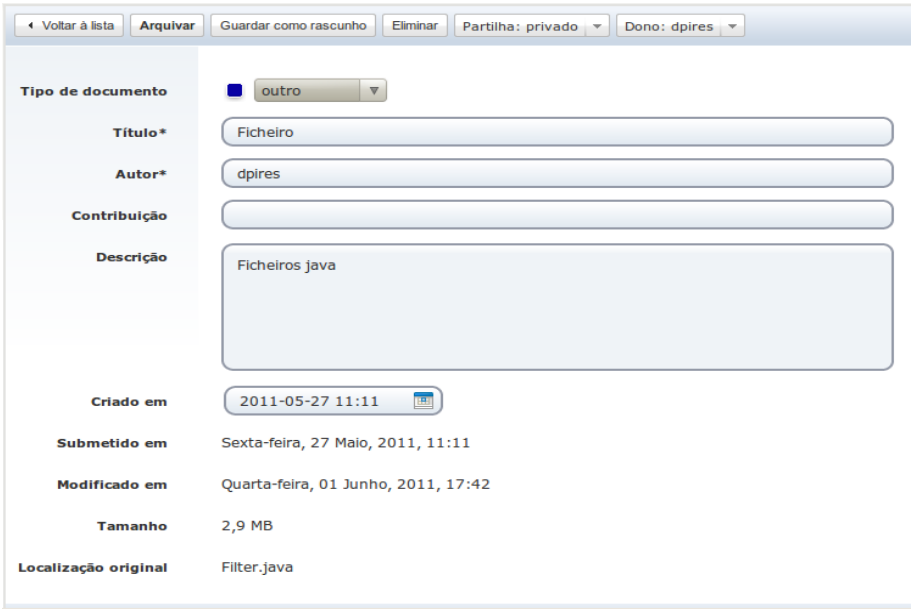

Figura 3.1: Metadados do bundle

weebox Core Services

O weebox core é a base do sistema que guarda os documentos (ficheiros e metainformação) e permite a pesquisa e recuperação dos mesmos. A gestão de utilizadores e partilha de documentos também é implementada por este módulo. Sendo o 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[1], etc.

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

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 de permissões por documento, pesquisa simples e avançada em texto integral. Estas funcionalidades podem ser consultadas no Anexo A.

Este módulo depende do weebox core e 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. Cada campo é definido pelos seguintes atributos [8]:

- 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.);

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

- 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.

Estrutura de dados

A informação guardada no weebox é toda definida por ID (identificadores), não sendo guardada numa forma hierárquica. A estrutura de dados usado para guardar a informação é constituída por três grupos: filtros, tipos de documentos e vocabulários controlados. O modelo de dados para além de conter vários filtros, nos quais os documentos podem ser guardados, possui também diferentes tipos de documentos, aos quais os documentos (*bundles*) se encontram associados, definindo 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, figura 3.2.

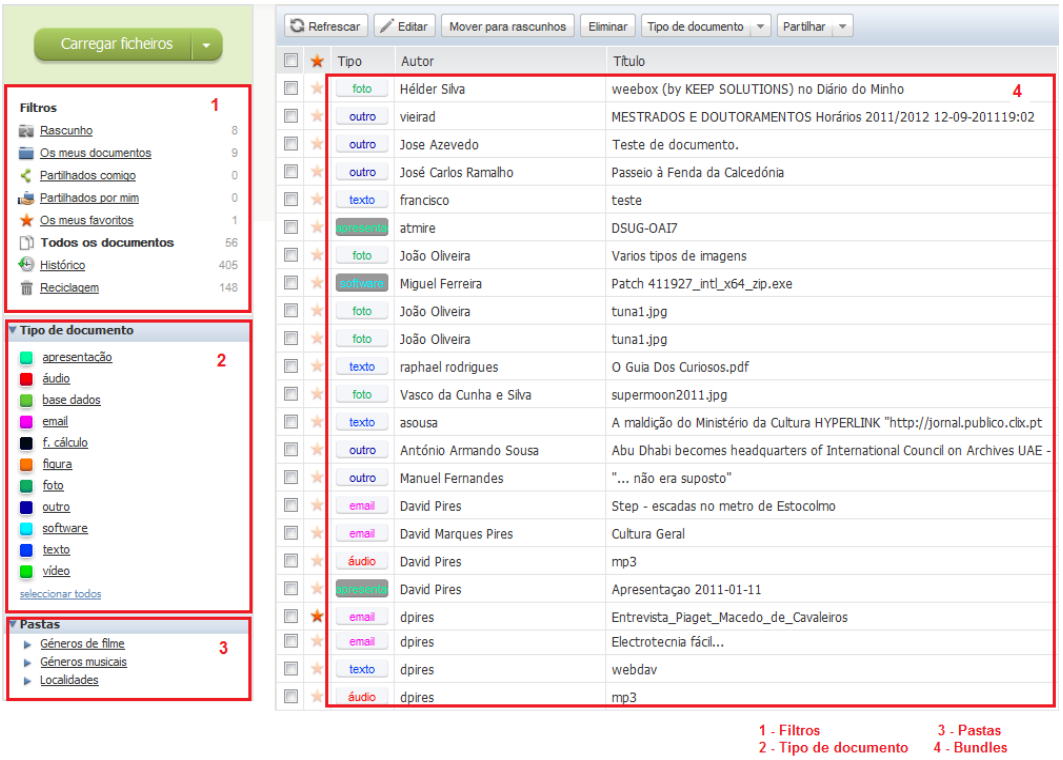

Figura 3.2: Modelo de dados

3.1.3 WebDav

Web-based Distributed Authoring and Versioning mais conhecido pela sigla WebDav, ou Criação e Distribuição de Conteúdo pela Web é uma extensão do protocolo *HyperText Transfer Protocol* (HTTP) [33], para transferência de arquivos, publicação e gestão de ficheiros através dum *browser* Internet, permitindo também que os clientes publiquem, bloqueiem e efectuem a gestão de recursos na *World Wide Web (Web)*, facilitando a colaboração entre utilizadores de edição e gestão de documentos e arquivos armazenados em servidores Web, sendo também usado como protocolo responsável pelo tratamento de pedidos e respostas entre cliente e servidor. O objectivo de WebDav é proporcionar funcionalidades para criar, alterar e mover documentos num servidor remoto.

O WebDav oferece muitos benefícios [35]:

- Permite a publicação contínua e sem interrupções de conteúdos na Web. Torna possível que pessoas ou grupos possam publicar directamente os seus trabalhos na Web usando o protocolo HTTP;
- Grupos de trabalho podem colaborar com o autor do documento que se encontra na Web, usando o bloqueio para evitar conflitos quando se está a alterar o documento;
- Não tem restrição quanto aos tipos de documento que podem ser criados;
- O WebDav e o HTTP fornecem uma interface comum para o sistema de repositórios, como a gestão de documentos, sistemas de arquivos, etc;
- O WebDav permite que dispositivos *embedded* com segurança possam escrever para a Web, habilitando uma nova classe de dispositivo Web.

Mais especificamente o WebDav foi definido na RFC 4918 [6] por um grupo de trabalho da *Internet Engineering Task Force* (IETF) [16]. O grupo de trabalho IETF definiu as características mais importantes para o protocolo WebDav [6, 30]:

- *locking* - permite bloquear e desbloquear recursos para que vários utilizadores possam ler um ficheiro em simultâneo. É possível bloquear os recursos de duas formas: compartilhada ou exclusiva. Quando um bloqueio é compartilhado, todos os utilizadores têm acesso ao recurso, porém com permissão somente de leitura. Quando o recurso é exclusivo somente o utilizador que bloqueou o recurso tem acesso a ele enquanto permanecer bloqueado;
- *properties* - permite alterar as propriedades associadas aos recursos. Por exemplo remoção, criação e consulta de informações sobre o autor, modificar a data, etc;

- *namespace management* - permite mover/copiar/eliminar recursos e criar novas colecções;
- *collections* - permite a criação, remoção e listagem de recursos.

O desenvolvimento do WebDav teve início em 1996 quando Jim Whitehead em parceria com o *World Wide Web Consortium* (W3C) [34], decidiram encontrar uma solução para o problema da edição distribuída na *World Wide Web* (WWW). A primeira proposta realizada por Tim Berners-Lee consistia em criar um meio onde fosse possível a leitura e escrita na Web, dando origem ao primeiro navegador Web, chamado *WorldWideWeb*. Esse navegador já era capaz de exibir e editar páginas remotas na Web, mas no fim funcionou somente como um meio de leitura. Para resolver este problema, Whitehead e outras pessoas que quiseram solucionar esta limitação, e numa reunião com a W3C, decidiram que o melhor caminho a seguir era formar um grupo de trabalho IETF. Após uma análise feita ao trabalho já realizado pelo IETF, ficou claro que tanto a edição e o controlo de versões distribuído implicariam muito trabalho e que a melhor solução era separar estas duas tarefas. Após esta separação de tarefas o grupo de trabalho WebDav decidiu focar-se principalmente na edição distribuída e deixar as versões para trabalho futuro. A versão viria a ser posteriormente acrescentado pela extensão Delta-V. O WebDav acrescentou os seguintes métodos ao HTTP [6, 27]:

- *PROPFIND* - implementa uma pesquisa de propriedades sobre recursos identificados pelo *Uniform Resource Identifier* (URI);
- *PROPPATCH* - possibilita alterar as propriedades dos recursos, como mudar e apagar propriedades dos recursos identificados pelo URI;
- *MKCOL* - usado para criar colecções;
- *COPY* - utilizado para copiar recursos;
- *MOVE* - utilizado para mover recursos;
- *DELETE* - permite eliminar recursos;
- *LOCK* - possibilita bloquear um recurso;
- *UNLOCK* - usado para desbloquear o recurso.

Na figura 3.3 são demonstradas as solicitações do protocolo cliente WebDav genérico para realizar a edição dum recurso. O fluxo da informação está representado pelas setas e o protocolo HTTP 1.1 é utilizado para os pedidos. A aplicação usa o método *LOCK* para bloquear o recurso e, assim evitar modificações por outras aplicações. Em seguida, usa o método *PROPFIND* para retornar as propriedades

do recurso e o método *GET* para retornar o seu conteúdo, que serão exibidas para a edição. Após todas as edições estarem concluídas, usa o método *PUT* para guardar o recurso e voltar para o Web. Por fim usa o método *UNLOCK* para remover o bloqueio feito com o método *LOCK*, permitindo assim que outros utilizadores possam aceder ao recurso [36, 37].

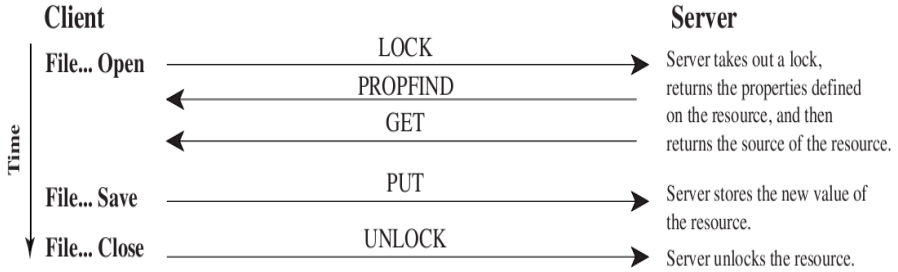

Figura 3.3: Solicitações de protocolo de um cliente WebDav [37]

Sendo uma tecnologia criada para o uso na Web e que veio acrescentar funções que ainda não existiam, logo apareceram as discussões sobre as suas vantagens e desvantagens. As principais vantagens do WebDav são relativas ao facto de usar um protocolo baseado no protocolo HTTP que já é usado há muito tempo na infraestrutura Web. Se se pretender usar métodos de segurança, pode aproveitar-se também o uso da autenticação criptográfica forte, *proxy*, cache e criptografia com SSL[1], já usados pelo HTTP [6, 13]. Outra vantagem é o uso do XML, pois o protocolo pode ser estendido sem fragmentá-lo. Uma parte do sucesso da Web baseia-se nos protocolos subjacentes, HTML e HTTP, pois são simples mas ao mesmo tempo poderosos, sendo baseados em texto torna-os mais fácil de adoptar. O WebDav é capaz de trabalhar com qualquer sistema operativo: UNIX, Windows, ou Macintosh [17, 25]. As desvantagens podem-se verificar no número de projectos e especificações para WebDav que sugerem que não é um protocolo simples. Em vez de se usar um método já existente para uma função, o WebDav essencialmente criou um formato HTTP de pedido/resposta num formato XML para que fosse possível realizar um conjunto de pedidos duma só vez [35]. Com esta opção tomada o protocolo tornou-se complexo, sendo também necessário um suporte a nível aplicativo e *middleware* [17, 25].

[1] *Secure Sockets Layer*

WebDav server

O servidor WebDav é o responsável por receber os pedidos dos clientes do WebDav e de lidar com esses pedidos. Consoante o pedido que foi efectuado, o servidor verifica as acções apropriadas a realizar. Como exemplo, se um cliente envia um pedido com a solicitação para bloquear um documento, o servidor vai verificar os privilégios do utilizador no documento especificado e se o documento não está bloqueado para o mesmo. Além disso o WebDav faz a gestão do repositório onde estão guardados os documentos. O formato dos pedidos para o WebDav baseia-se no formato do HTTP e compreende os três seguintes componentes [1]:

- Método - estados do método a ser executado pelo cliente;
- Cabeçalho - descreve as instruções de como a tarefa deve ser concluída;
- Corpo (opcional) - define os dados utilizados na instrução, ou adiciona instruções de como o método está a ser executado.

Existem três tipos de servidores WebDav: servidor classe 1, classe 2 ou DeltaV [18].

O servidor classe 1 oferece as funcionalidades básicas dum sistema de arquivos tradicional, usando os métodos do WebDav: OPTIONS, PROPFIND, PUT, DELETE, MKCOL, MOVE, COPY. No servidor classe 1 é permitido criar, apagar, copiar, mover arquivos e pastas, como também criar, actualizar e apagar propriedades personalizadas.

O servidor classe 2 permite uma hierarquia de itens de bloqueio usando os métodos do WebDav LOCK e UNLOCK, bem como as funcionalidades fornecidas pelo servidor classe 1. Os itens de bloqueio podem ajudar a impedir a modificação de itens do servidor por outros utilizadores.

O servidor DeltaV oferece a possibilidade de *check-out/check-in* dos recursos e o acompanhamento das versões.

A tabela 3.1 lista os requisitos mínimos do servidor para alguns clientes WebDav:

WebDAV Client	WebDAV Server Requirements
Windows Shell on Windows 7 / Vista (Mini-Redirector)	Class 2
Web Folders on Windows XP / 2003 / 2000	Class 1
MS Office 2010 / 2007 / 2003 / XP	Class 2
Max OS X	Class 2
iPad Keynote, Pages and Numbers	Class 1
KDE Konqueror	Class 1
GNOME Nautilus	Class 1
DAVExplorer	Class 1

Tabela 3.1: Requisitos mínimos dos servidores [18]

WebDav client

Um cliente WebDav é uma aplicação que pode enviar solicitações e receber respostas usando o protocolo WebDav, podendo comunicar com os servidores WebDav que permitem executar a criação remota de documentos através dos métodos que contêm cabeçalhos de solicitação e formatos do corpo da solicitação. Fornece também métodos para armazenar e recuperar os recursos, criar listas de colecções de recursos, bloqueio de recursos, definir e recuperar propriedades de recursos.

O cliente WebDav ao ligar-se ao servidor WebDav vai poder aceder ao repositório deste. Os conteúdos do repositório existentes no servidor são apresentados num sistema de pastas. Independentemente do método de acesso às pastas, é necessário saber o URL da pasta. Os URLs do WebDav têm o seguinte formato:

$$http://server:port/Documents[/Path]$$

- *server* - endereço IP (ou nome DNS) do servidor;
- *port* - o número de porta de acesso WebDav. Esta é a porta usada pelo servidor Web;
- *Path* - opcional, especifica uma pasta especial que se deseja aceder.

Os clientes WebDav diferem no seu comportamento e podem ser divididos em quatro grupos: clientes dedicados WebDav, clientes habilitados WebDav, editores de documentos WebDav e navegadores Web.

O primeiro grupo, clientes dedicados WebDav que inclui o DAV Explorer, envia pedidos directamente para o servidor. O DAV Explorer é semelhante ao Windows Explorer. O segundo grupo, clientes habilitados WebDav, é apoiado pelo protocolo WebDav que cria pastas na Web. A pasta criada representa uma interface para o

repositório de pastas do WebDav no servidor e assim a web pode actuar como um intermediário entre clientes habilitados e servidores WebDav. Este grupo representa a forma mais popular de aceder aos servidores WebDav no Microsoft Windows, um exemplo de clientes habilitados WebDav é o Microsoft Office que permite a edição directa dos documentos nos repositórios do servidor WebDAV através de *Web Folders*. O terceiro grupo, editores de documentos WebDav, pode ser usado para editar documentos directamente em servidores WebDav. Um exemplo do terceiro grupo de clientes WebDav é o JEdit, que é baseado num editor de texto JAVA e pode ser estendido para um cliente WebDav usando um *plug-in* do WebDav. No quarto grupo, navegadores Web, podem existir algumas limitações quando se quer utilizar os servidores WebDav, pois podem, existir *browsers* que não suportam os métodos WebDav através do protocolo HTTP. Por exemplo o SAP Portals Enterprise Portal Server e o Xythos WebFile Server podem ser utilizados através de navegadores web [1].

O WebDav tendo a característica de Criação e Distribuição de Conteúdo pela Web, e sendo uma extensão do protocolo HTTP, teve que seguir as normas RFC já existentes sobre o protocolo HTTP e de criar novas normas RFC referentes ao WebDav, para a sua criação. As normas usadas pelo WebDav são descritas na secção 3.1.3.

Implementação de clientes de Arquivos WebDAV

O sistema de arquivos WebDav é implementado como um módulo de sistema de arquivos de baixo nível, normalmente dentro do *kernel* do sistema operativo. Isso significa que o compartilhamento DAV é montado como qualquer sistema de arquivos de rede, semelhante à montagem duma partilha NFS em Unix ou duma partilha SMB no Windows. Como resultado, esse tipo de clientes fornece uma forma completamente transparente para acesso de leitura/escrita a todos os programas [11].

- **Windows**
  A Microsoft introduziu suporte ao cliente WebDav no Microsoft Windows 98 com um recurso chamado *"Web folders"*, tendo também sido incluído no Windows 2000. No Windows XP, a Microsoft adicionou o *"WebDav mini-redirector"*, que funciona como um serviço de sistema de redireccionamento de rede, imediatamente acima do arquivo do sistema, permitindo que partilhas WebDav sejam montadas através duma letra de unidade. O Windows inclui apenas o *WebDav redirector*, mas se for instalado o "Microsoft Update for Web Folders" é possível ter aceso ao tradicional cliente *"Web folders"* [42].

- **Mac OS X**
  Mac OS X versão 10.0 e seguintes já suportam nativamente o WebDav como

um tipo de sistemas de arquivos. O sistema pode montar o WebDav habilitado a directórios do servidor para o sistema de arquivos usando o tradicional mecanismo de montagem BSD, ou através do 'Connect to Server' encontrado no Finder.
O Finder apresenta um compartilhamento WebDav como um disco externo, permitindo aos utilizadores interagir com o WebDav como fariam com qualquer outro sistema de arquivos [42]. Para além de ser possível montar um compartilhamento WebDav usando o tradicional mecanismo de montagem BSD, também é possível montar um compartilhamento Webdav usando o terminal através do comando mount:

$$\$\ mount -t\ webdav\ http://svn.example.com/repos/project/some/mountpoint$$

Se o *mod_dav_svn* for mais antiga que a versão 1.2, o OS X somente irá montar o compartilhamento WebDav para leitura. Isto acontece porque o suporte para bloqueio de leitura/escrita só apareceu pela primeira vez no Subversion 1.2 [11].

- **Linux**
  Utilizadores de Linux podem montar compartilhamentos de WebDav usando o *davfs2*, que é um módulo do sistema de arquivos para o kernel Linux. O KDE tem suporte para o WebDav nativo, como parte de *kio_http*. Isso permite que as aplicações do KDE possam interagir directamente com os servidores WebDav [42].
  Após a instalação do *davfs2*, é possível montar uma partilha de rede usando o Linux WebDav através do comando:

$$\$\ mount.davfs\ http://host/repos/mnt/dav$$

Normas RFC's

Os RFC (*Request For Comments*) constituem uma série de documentos editados desde 1969. São um conjunto de documentos com informações técnicas detalhadas sobre protocolos da Internet, sendo documentos de referência junto da Comunidade da Internet. Têm a principal função de descrever, especificar, padronizar e debater a maioria das normas, padrões de tecnologia e protocolos ligados à Internet e às redes em geral.

As normas RFC são apresentadas ao organismo IETF, que posteriormente irá validar ou não a norma. Cada um destes documentos representa uma proposta de especificação que pode a qualquer momento tornar-se obsoleta por um novo documento RFC. Os RFC são ficheiros de texto cujo nome segue uma ordem, o nome dos RFC é "rfcxxxx.txt", sendo xxxx um número incrementado para cada novo RFC.

Como foi dito anteriormente para a criação do WebDav foi preciso tomar em consideração normas já existentes e criar novas. De seguida serão explicadas as normas RFC usadas para a sua elaboração. [29]

Especificações WebDav

Especificações Dav

- *RFC 4918: HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)* - este documento especifica um conjunto de métodos, cabeçalhos e métodos auxiliares ao HTTP/1.1 para fornecer recursos para a gestão de propriedades de recursos, criação e gestão de colecções de recursos, manipulação de URL *namespace*, e bloqueio de recursos [6];

- *RFC 3253: Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning)* - este documento especifica extensões para WebDav para fornecer recursos para o controle de versões e gestão de configuração [3];

- *RFC 5323: Web Distributed Authoring and Versioning (WebDAV) SEARCH* - este documento especifica um conjunto de métodos, cabeçalhos e propriedades compondo o WebDav SEARCH, uma aplicação do protocolo HTTP/1.1 para uma pesquisa eficiente para o recurso DAV com base num conjunto de critérios fornecidos pelo cliente [28];

- *RFC 4437: Web Distributed Authoring and Versioning (WebDAV) Redirect Reference Resources* - protocolo para a criação de redireccionamentos de recursos, permitindo que clientes possam criar caminhos de acesso para novos recursos. O WebDav torna possível organizar os recursos HTTP em hierarquias, colocando-os em grupos, conhecidos como colecções, o que permite uma mais fácil visualização e manipulação. Este protocolo prevê a criação e administração remota desses recursos [40];

- *RFC 3648: Web Distributed Authoring and Versioning (WebDAV) Ordered Collections Protocol* - esta especificação amplia o protocolo Web Distributed Authoring e Versioning (WebDAV) no apoio do lado do servidor na ordenação de membros da colecção. Tem particular interesse nas ordenações que não são baseadas em valores de propriedades, e assim não podem ser alcançadas através duma opção de ordenamento do protocolo de pesquisa [39];

- *RFC 3744: Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol* - este documento especifica um conjunto de métodos, cabeçalhos, corpo da mensagem, propriedades e relatórios que definem extensões de Controlo de Acesso ao protocolo WebDAV Distributed Authoring. Este

protocolo permite que um cliente possa ler e modificar listas de controlo de acesso, instruindo o servidor para permitir ou negar as operações num recurso [4];

- *RFC 4122: Universally Unique IDentifier (UUID) URN Namespace* - esta especificação define um *namespace Nome Uniform Resource* para UUIDs. A UUID é de 128 bits de comprimento e pode garantir singularidade no espaço e no tempo [21].

Especificações HTTP

- *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1* - esta especificação define o protocolo referido como HTTP/1.1 e é uma actualização do RFC 2068 [10];

- *RFC 2617: HTTP Authentication: Basic and Digest Access Authentication* - esta especificação detalha os mecanismos básicos para Digest Access Authentication. É um substituto para o *RFC 2069 (Digest Access Authentication)* [12].

Especificações XML

- *Extensible Markup Language (XML) 1.0* - Especificação do núcleo XML. É uma recomendação da W3C [26];

- *Namespaces in XML* - Extensão para XML e é também uma recomendação do W3C [15];

- *XML Media Types* - Norma RFC que descreve quando usar *text/xml*, *application/xml*, ou o uso do parâmetro *charset* [23].

DeltaV

DeltaV é um protocolo de rede que fornece recursos para controlo de versão e gestão remota de configuração de documentos armazenados num servidor web. O protocolo DeltaV pode ser usado para apoiar os seguintes cenários [38]:

- Um grupo de pessoas em locais diferentes que precisam desenvolver um relatório em conjunto. Durante o processo de elaboração do relatório, vão ser criadas versões intermédias do relatório com o *feedback* de cada pessoa e seria importante guardar essas versões intermédias;

- Um projecto de colaboração entre empresas que envolva pessoas de várias empresas, em diversos países. Havendo um site sobre o projecto, que é desenvolvido usando as ferramentas do WebDav. Uma vez que qualquer membro do projecto pode editar o conteúdo do site, é importante que a equipe se mantenha a par das mudanças realizadas, para que se possa ter um registo de quem alterou a página e se um erro ocorrer seja possível ter acesso à versão anterior;

- Num projecto *open source*, onde é possível que qualquer programador colabore a desenvolver uma aplicação de *software*. É importante gravar todas as mudanças do código fonte, bem como criar bases estáveis do seu código fonte para as versões do *software*.

O protocolo DeltaV é uma extensão do protocolo WebDav, que veio fornecer recursos para controlo de versão de documentos que não existiam no protocolo WebDav. O grupo de trabalho DeltaV decidiu então continuar o que já estava feito, tendo como objectivo adicionar a adição de controlo de versões para a Web, bem como a gestão remota de configurações. Para a base fornecida pelo HTTP e WebDav, DeltaV acrescentou 11 métodos adicionais. A Figura 3.4 resume os métodos definidos por HTTP, WebDav e DeltaV.

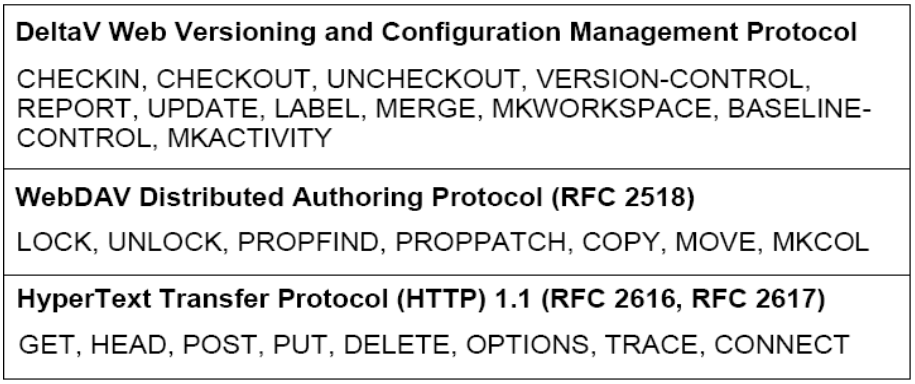

Figura 3.4: Métodos definidos pelo HTTP/1.1, protocolo WebDav Distributed Authoring e protocolo DeltaV Web Versioning and Configuration Management [38]

Dentro da família de especificações HTTP/DAV/DeltaV, um documento num servidor Web é conhecido como um recurso. O estado dum recurso WebDav é constituído por duas partes, um corpo que contém o conteúdo principal (ex.um documento texto) e as propriedades, pares *name/value* que fornecem os metadados sobre o recurso. As propriedades podem ser de dois tipos, propriedades *live* cujo valor é calculado e controlado pelo servidor e propriedades *dead* cujo valor é controlado pelo cliente e armazenado pelo servidor. As operações sobre os recursos são denominadas métodos (descritos em cima). DeltaV inclui os seguintes métodos:

- *Version Control* - é um conjunto de restrições sobre como um recurso pode ser actualizado. Ele pode ser aplicado a um recurso *versionable* ou a um recurso *version-controlled*. Se o pedido de URL identifica um recurso *versionable*, uma nova versão é criada, cujo conteúdo e propriedades são copiadas. Se o pedido de URL for um recurso *version-controlled*, o recurso só permanece sob controlo de versão, isso permite que um cliente não tenha conhecimento da existência dum servidor que coloca automaticamente o recurso sob controlo de versão quando ele é criado [3];

- *CHECKOUT* - um pedido de verificação geral pode ser aplicado a um *check-in* de recursos com controlo de versão para permitir modificações ao conteúdo e propriedades [3];

- *CHECKIN* - um pedido *CHECKIN* pode ser aplicado a um recurso *check-out* dum controlo de versão para produzir uma nova versão, cujo conteúdo e propriedades são copiados a partir do recurso *check-out* [3];

- *UNCHECKOUT* - um pedido *UNCHECKOUT* pode ser aplicado a um recurso *check-out* com controlo de versão para cancelar o *check-out* e restaurar o estado pré-CHECKOUT do recurso *version controlled* [3];

- *MKWORKSPACE* - cria um novo recurso no *workspace*. Um servidor pode restringir a criação de espaço de trabalho para colecções particulares, mas um cliente pode determinar a localização dessas colecções [3];

- *UPDATE* - o método de actualização modifica o conteúdo e as propriedades dum recurso *check-in* com controlo de versão. A resposta a uma solicitação de actualização identifica os recursos modificados pelo pedido de modo a que um cliente possa actualizar de forma eficiente qualquer estado em cache [3];

- *LABEL* - um pedido *label* pode ser aplicado a uma versão para modificar os *labels* seleccionadas nessa versão, preservando o *label* quando a versão é armazenada ou recuperada. Ao comparar dois nomes de *label* para decidir se correspondem ou não, o servidor deve usar um URL-case-sensitive codificado em UTF-8 [3];

- *MERGE* - executa a junção lógica duma versão especificada num determinado recurso de *version-controlled*. Se a fonte directa não é um ascendente ou um descendente, os controlos *MERGE* irão fundir o alvo e adicionar o URL da fusão de origem. Portanto é da responsabilidade do cliente, actualizar as propriedades do conteúdo do *check-out* destino para que ele reflicta a lógica directa da fonte e o estado actual do destino [3];

- *BASELINE-CONTROL* - a colecção pode ser colocada sob o controlo de *baseline* com um pedido *BASELINE-CONTROL*. Quando uma colecção é colocada sob o controlo de *baseline*, a propriedade de *version-controlled* da

colecção é definida para identificar uma nova configuração com controlo de versão [3];

- *MKACTIVITY* - um pedido *MKACTIVITY* cria um recurso duma nova actividade. Um servidor pode restringir a actividade de criação de colecções particulares, mas um cliente pode determinar a localização dessas colecções [3];
- *REPORT* - um pedido *REPORT* é um mecanismo extensível para obter informações sobre um recurso. Ao contrário duma propriedade de recurso, que tem um único valor, o valor dum *report* pode depender de informações adicionais especificadas no corpo e nos cabeçalhos duma solicitação *REPORT* [3].

3.1.4 Protocolo escolhido

O protocolo WebDav foi escolhido como o protocolo a usar na implementação do sistema proposto, devido a permitir testar/usar a modelação de uma estrutura ontológica para uma estrutura hierárquica e permitir também a manipulação do conteúdo dum sistema de gestão documental através da Internet. A modelação para uma estrutura hierárquica é justificada pelo facto dos utilizadores já estarem muito habituados a este sistema, o que torna importante a possibilidade de ter uma visão familiar dos conteúdos do repositório, sob a forma dum sistema de ficheiros. O módulo weebox WebDav oferece exactamente isso, permitindo que em qualquer sistema operativo (Windows, Mac OS, Linux, etc.) seja montada uma partilha remota segundo o protocolo WebDav, vendo o repositório como um conjunto de pastas e ficheiros, usando as próprias interfaces do sistema operativo (e.g. Windows Explorer). No entanto, a variedade possível na organização das pastas, devido ao acrescido valor da metainformação, tornam possível que existam várias visões do repositório em forma de sistema de ficheiros. A mais evidente será reflectir a organização original, caso essa informação exista. O WebDav permite guardar a informação duma forma hierárquica, usando para isso a metainformação dos próprios dados. Como a informação é apresentada duma forma hierárquica, isso vai permitir uma consulta mais fácil para o utilizador e tornar a catalogação dos ficheiros automática, devido aos ficheiros conterem a informação do nível onde se encontram. Para testar as funcionalidades do WebDav, foi necessário escolher um repositório de conteúdos, tendo sido escolhido para este caso de estudo o weebox da KEEP SOLUTIONS.

3.2 Estrutura de dados

A estrutura de dados usada pelo protocolo WebDav segue o sistema de ficheiros hierárquico oferecendo uma visão dos conteúdos do repositório sob a forma dum

sistema de pastas, permitindo assim em qualquer sistema operativo montar uma partilha de repositórios usando o protocolo WebDav. A principal tarefa na implementação desta solução é transformar o modelo de dados usado pelo weebox para uma estrutura hierárquica. Como já foi referido na secção 3.1.2, o sistema weebox usa um modelo de dados onde a informação é guardada através dos id's dos *bundles* e ficheiros, na figura 3.5 podemos observar a estrutura do modelo de dados usado pelo sistema weebox e na figura 3.6 é apresentado a conversão da estrutura do modelo de dados do sistema weebox para uma ontológia.

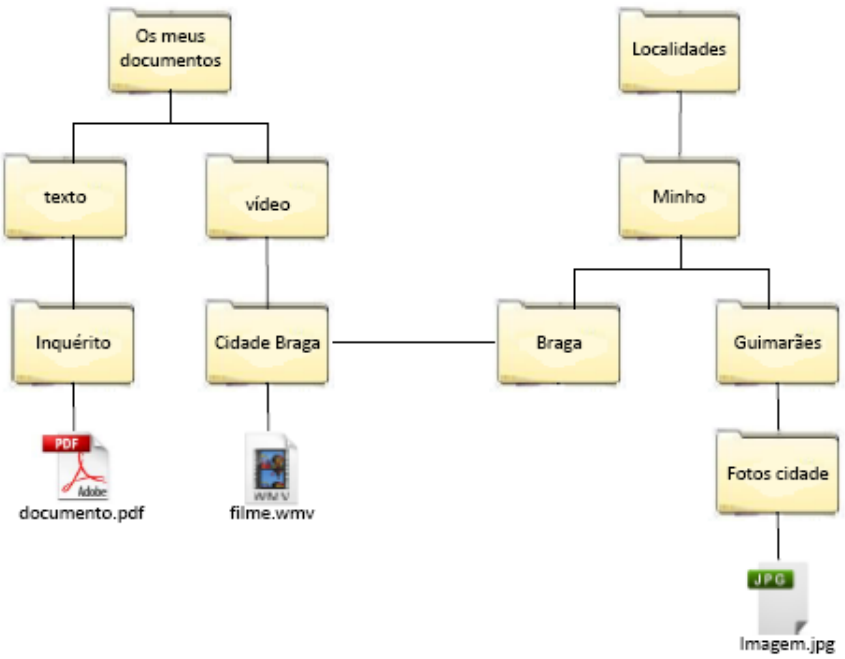

Figura 3.5: Estrutura usada pelo weebox

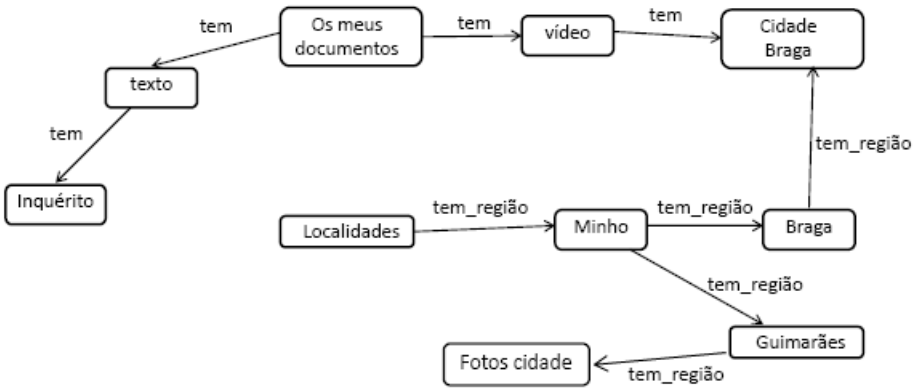

Figura 3.6: Conversão da estrutura para uma ontológia

A modelação duma estrutura ontológica para uma hierárquica segue o algoritmo descrito na secção 2, onde são definidos os dados a usar na construção dos níveis hierárquicos e possível duplicidade dos mesmos. Na figura 3.7 é apresentada a estrutura criada para o problema.

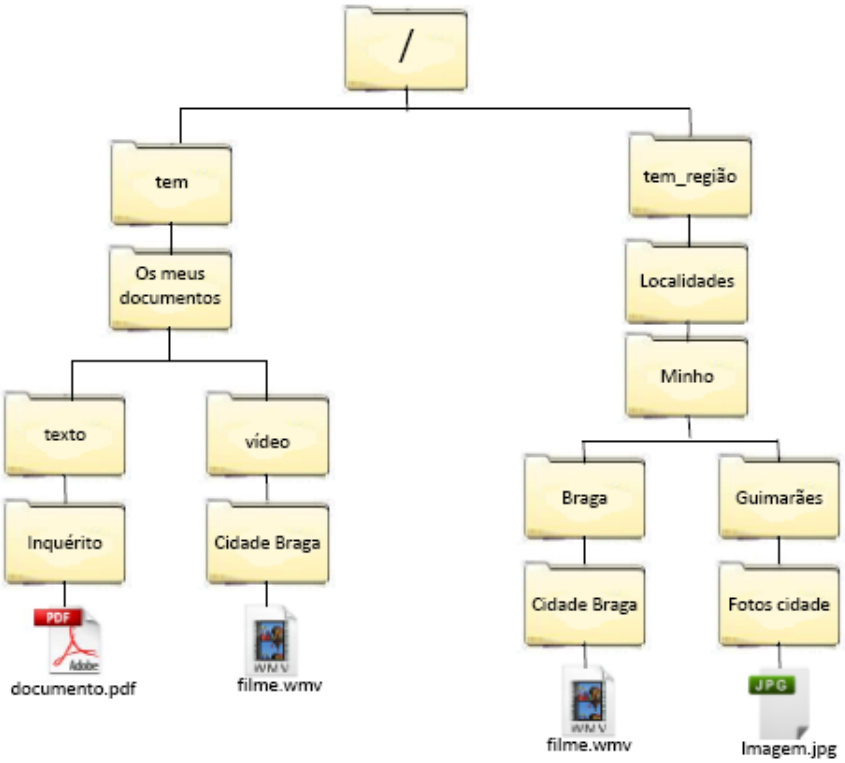

Figura 3.7: Estrutura usada pelo WebDav

A página principal do weebox pode ser dividida em secções de acordo com as suas funções e comportamentos, figura 3.8. A secção "Filtros" 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 "Tipo de documento" permite ao utilizador seleccionar um tipo de documento, exemplos (apresentação, áudio, base dados, email, f.cálculo, figura, foto, outro, software, texto, vídeo). A secção "Pastas" permite ao utilizador seleccionar vocabulários controlados. As selecções nestas secções alteram a listagem de documentos apresentada no painel da secção "Listagem".

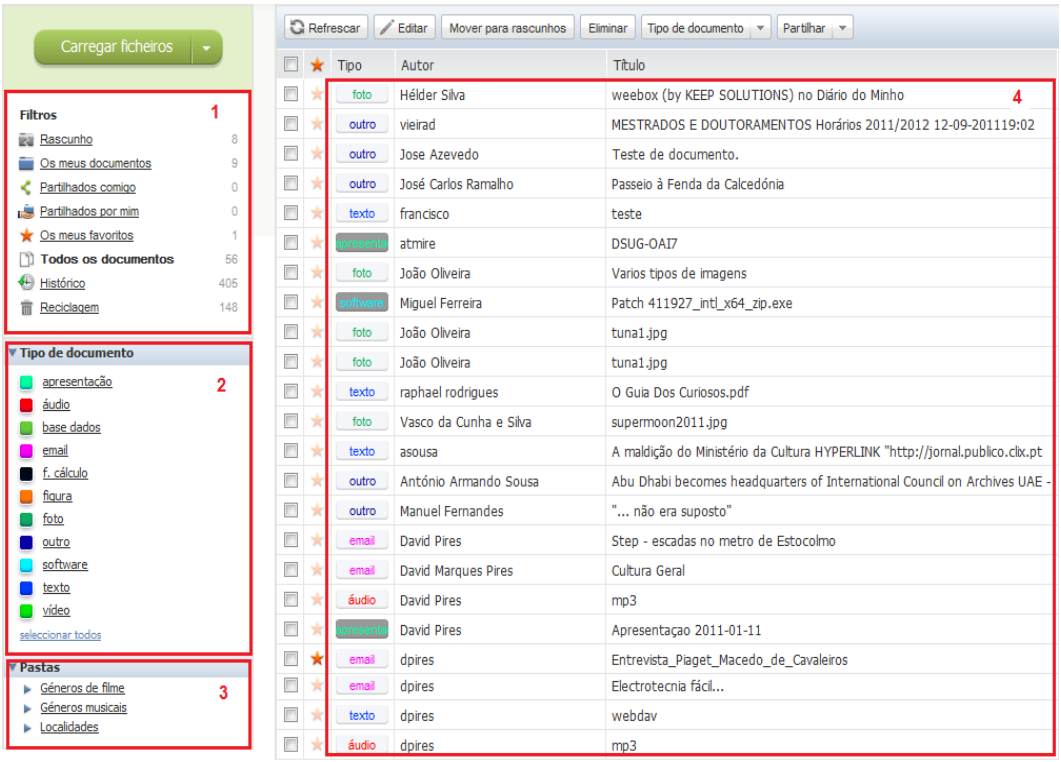

Figura 3.8: Primeira página do weebox após a realização do controlo de acesso

O diagrama correspondente a esta divisão pode ser visto na Figura 3.9, sendo dado um nome significativo a cada secção da página para melhor identificação.

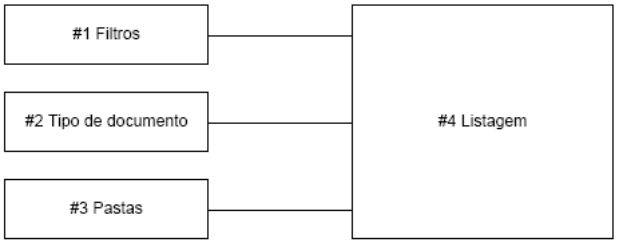

Figura 3.9: Diagrama da página Web do weebox dividida em secções

O primeiro passo realizado para criar a organização dos dados a implementar na solução foi definir uma hierarquia sobre as secções usadas no weebox. A construção da hierarquia a usar pelo protocolo WebDav foi dividida em quatro níveis: o primeiro nível é constituído pela secção "Filtro" e "Pastas", o segundo nível pela secção "Tipo de documento", o terceiro nível pela secção "Listagem" e o quarto nível pelos ficheiros. Na figura 3.10 é ilustrada a hierarquia de pastas criada no WebDav.

Figura 3.10: Hierarquia Pastas WebDav

3.3 WebDav-servlet

O projecto *open-source* usado para criar o WebDav-servlet tem como base o Java Web Server 1.0.3 da Universidade de Columbia criado por Marc Eaddy, Shao Rong, e Shapiro Jonathan [7]. O projecto consistia em desenvolver uma implementação cliente/servidor para o suporte dum subconjunto de extensões para o protocolo HTTP para o WebDav. As extensões propostas pelo WebDav consistem em

fornecer suporte para os tradicionais sistemas de gestão de trabalho colaborativo na *World Wide Web* com recursos como propriedades de recurso, a manipulação de *namespace*, controle de bloqueio de recursos e versão.

A motivação de criar este projecto deve-se sobretudo à forma como o protocolo HTTP cria uma infraestrutura altamente escalável para navegar e pesquisar um conjunto distribuído de repositórios de informações heterogéneas, mas fornece um suporte limitado para a criação e gestão dessa informação. Na ausência de características de protocolo para apoiar as tarefas tradicionais de gestão de documentos, grande parte da criação de conteúdo da Web tem lugar fora da própria Web. No entanto, o modelo é aquele em que a Web serve apenas como meio de publicação, não como um ambiente de criação e suporte à colaboração verdadeiramente distribuída. Assim, a colaboração é limitada àqueles que podem partilhar os recursos da rede local.

Sendo um projecto *open-source*, o seu código está disponível para todas as pessoas que queiram implementar, melhorar ou mesmo adicionar novas funcionalidades ao projecto. Na implementação do caso de estudo proposto na dissertação foi usado o código base disponibilizado pelo projecto (disponível para download em `http://sourceforge.net/projects/webdav-servlet/files/`).

3.4 Funcionalidades

As funcionalidades do weebox/WebDav disponibilizadas aos utilizadores são relativas aos objectivos de acesso e de edição de *bundle* e documentos.

A secção de objectivos de acesso representam as funcionalidades de maior valor para a aplicação:

- Visualização de cada um dos filtros por omissão;
- Visualização por tipo de documento;
- Visualização por vocabulários controlados;
- Visualização de ficheiros.

A secção de objectivos de edição complementa as funcionalidades de acesso com a capacidade de alterar os documentos no repositório:

- Editar um ficheiro;
- Editar o nome da pasta dum documento;
- Adicionar um novo ficheiro para um *bundle*;

- Criar pastas.

As características de cada funcionalidade serão ilustradas de seguida com a ajuda de um diagrama de *use case*, figura 3.11, que descreve como os utilizadores interagem com o sistema relativamente a cada tarefa. Serão também apresentados diagramas de actividades para cada funcionalidade para explicar mais detalhadamente os processos apresentados no *use case*.

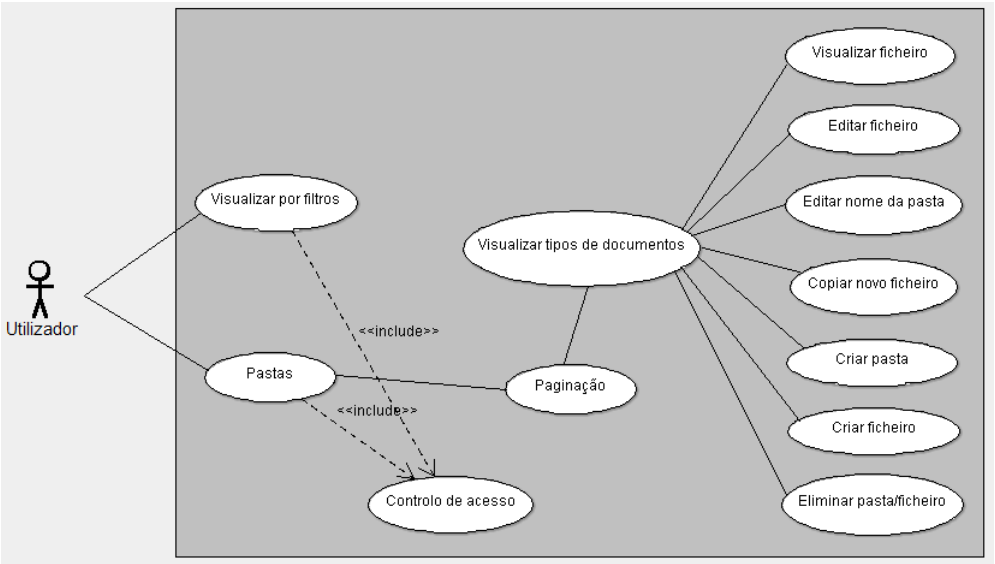

Figura 3.11: Use Case do sistema

3.4.1 Controlo de acesso

O controlo de acesso implementado no weebox é o tradicional sistema de *login/password*, como é ilustrado na figura 3.12

Figura 3.12: Controlo de acesso no weebox

O protocolo WebDav sendo uma extensão do protocolo HTTP inclui a utilização da autenticação base. Autenticação básica é uma forma de autenticação de utilizador onde um utilizador é identificado com o servidor. Quando o servidor está configurado para utilizar a autenticação básica, o computador cliente transmite credenciais dos utilizadores ao servidor. O Windows XP SP2/Vista/7 incluem a funcionalidade que permite controlar a utilização da autenticação base pelo redireccionador de DAV. O redireccionador DAV suporta o HTTPS ou segura (SSL), mas por predefinição desactiva a utilização da autenticação básica sem SSL [1] [22].

Descrição do *use case* para o controlo de acesso:

1. O use case começa quando o utilizador mapeia o WebDav;
2. Ao utilizador é solicitada a introdução do *username* e da *password* de acesso;
3. O sistema verifica as credenciais:

   3.1 Se as credenciais estiverem correctas, isto é, de acordo com o pré-registo, o controlo de acesso permite o acesso ao sistema;

   3.2 Caso contrário, não é permitido o acesso ao sistema.

O controlo de acesso apenas é realizado após a montagem do WebDav, para isso é preciso mapear um driver de rede, introduzindo o driver e o nome da pasta raiz onde vai ser montado o sistema, figura 3.13. Após a montagem do sistema é apresentada ao utilizador uma janela de login para o utilizador introduzir as suas credenciais (figura 3.14).

[1] Para contornar este comportamento, é preciso activar a autenticação básica no computador cliente. Para tal, deve-se alterar a entrada de registo UseBasicAuth para o valor 2. Esta alteração é feita através do regedit.exe

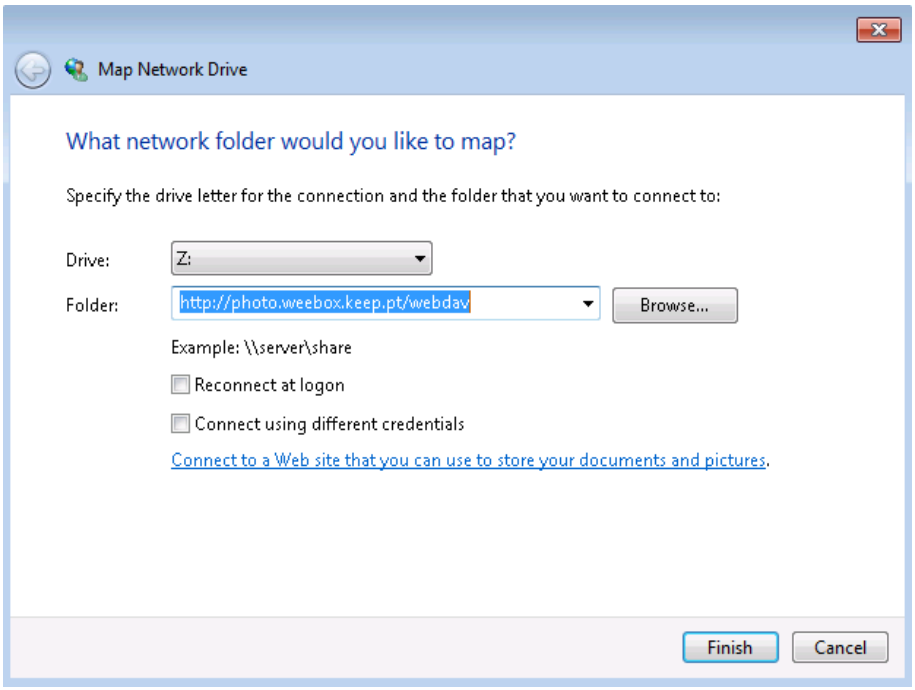

Figura 3.13: Janela para mapear o sistema no Windows

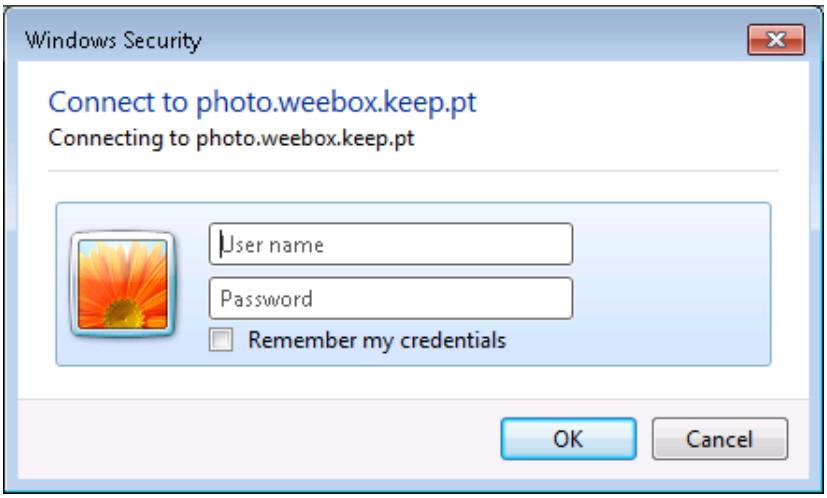

Figura 3.14: Janela de Login no Windows

3.4.2 Visualizar por filtros

A funcionalidade de visualizar por filtros permite ao utilizador seleccionar um filtro sobre os documentos que existam no sistema, figura 3.15.

Figura 3.15: Filtros

Como foi já referido na secção 3.2, a secção "Filtro" corresponde ao primeiro nível da hierarquia, que corresponde ao menu principal que será apresentado ao utilizador.

Descrição do *use case* para Visualizar por filtros:

1. O use case começa quando o utilizador selecciona o filtro;

2. O sistema apresenta ao utilizador os tipos de documentos.

A figura 3.16 representa o menu principal apresentado ao utilizador. Neste nível não são representados os *bundles* do filtro devido aos *bundles* pertencerem a outro nível como foi mencionado na secção 3.2.

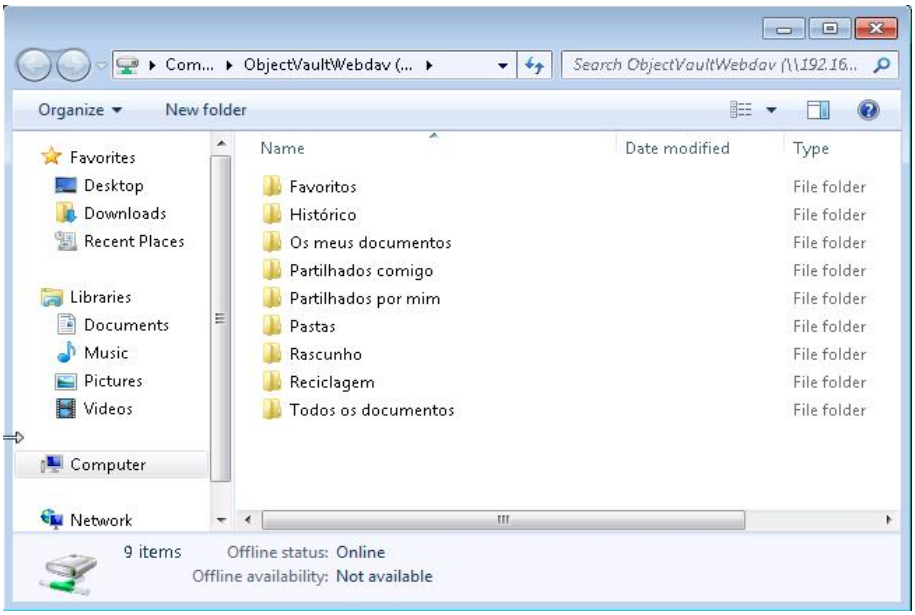

Figura 3.16: Figura do Menu inicial

3.4.3 Visualizar por pastas

A visualização por pastas corresponde a visualizar por vocabulários controlados. O termo pastas é apresentado no weebox em vez de vocabulário controlado porque o termo pastas é mais conhecido pelos utilizadores. Na figura 3.17 é ilustrada a hierarquia usada no weebox, neste exemplo é demonstrada a hierarquia da pasta "Localidades".

Figura 3.17: Figura da hierarquia de Pastas (weebox)

No menu principal do WebDav apresentado ao utilizador também é apresentado

o filtro "Pastas" como podemos ver na figura 3.16.

Descrição do *use case* para Visualizar por Pastas:

1. O use case começa quando o utilizador selecciona o Pastas;

2. O sistema apresenta ao utilizador o nível seguinte dos vocabulários controlados.

Na figura 3.18 é ilustrada o resultado da conversão para o WebDav da figura 3.17.

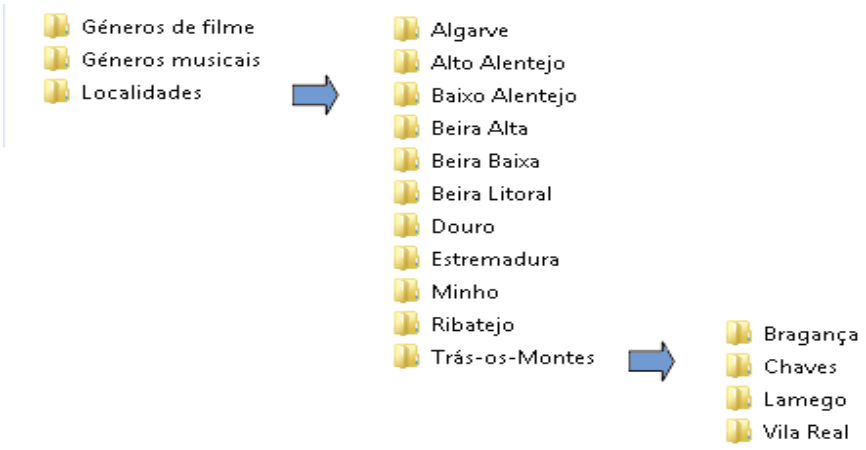

Figura 3.18: Figura da hierárquica de Pastas (WebDav)

Um ponto de optimização a nível da usabilidade do sistema seria usar uns caracteres especiais para distinguir os diferentes níveis usados no filtro "Pastas" e também para distinguir se é um vocabulário controlado ou um *bundle*, devido a ser possível ter *bundles* ao mesmo nível dos vocabulários controlados.

3.4.4 Visualizar por Tipo de documento

A funcionalidade de visualizar por tipo de documento permite realizar uma selecção sobre os documentos existentes no weebox, fazendo uma filtragem por tipo de documento. Este filtro por "Tipo de documento", figura 3.19, é realizado após a primeira filtragem dos documentos realizada pelo filtro "Filtros".

Figura 3.19: Filtro por Tipo de documento

Descrição do *use case* para Visualizar por tipo de documento:

1. O use case começa quando o utilizador selecciona o tipo de documento;

2. O sistema apresenta ao utilizador os *bundles* existentes dentro do tipo de documento seleccionado.

O filtro "Tipo de documento" corresponde ao segundo nível na nossa estrutura hierárquica. Na figura 3.20 é ilustrado o segundo nível apresentado aos utilizadores.

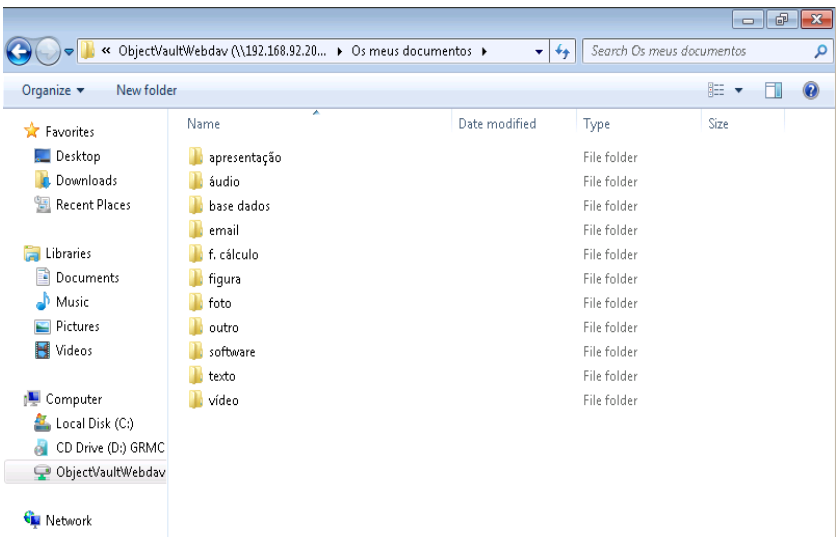

Figura 3.20: Sub-pastas existentes na pasta Todos os documentos

O nome destas sub-pastas é apresentado após ser feito uma tradução das mesmas, quer isto dizer, antes das pastas serem apresentadas o sistema carrega um ficheiro de configuração, onde irá verificar em que idioma se encontra, fazendo assim a tradução para a língua que se encontrar no ficheiro de configuração. Este método tem como vantagem a possibilidade de ter o sistema em vários idiomas, necessitando apenas alterar o ficheiro de configuração para o idioma pretendido. Cada pasta é identificada pelo seu *bundleId*, sendo que no caso de ser criada uma pasta e não for introduzido o nome da mesma, automaticamente o nome da pasta irá ficar com o valor do seu *bundleId*. O *bundle* contém a informação do documento numa *hash*, onde cada entrada corresponde ao identificador dum campo do seu tipo de documento e o valor associado a esse campo. O *bundleId* também é usado para todas as operações que são possíveis realizar sobre as pastas, como consulta, criação de ficheiros ou eliminação. Dentro destas sub-pastas encontramos as pastas relativas aos ficheiros existentes. As pastas relativas aos ficheiros na primeira página do weebox correspondem a secção "Listagem", que no sistema implementado corresponderá ao terceiro nível da hierarquia. Na figura 3.21 podemos ver como são apresentadas ao utilizador as pastas dos ficheiros no weebox, e na figura 3.22 a conversão da secção "Listagem" para o WebDav.

Tipo	Autor	Titulo
outro	Jose Azevedo	Teste de documento.
outro	José Carlos Ramalho	Passeio à Fenda da Calcedónia
outro	António Armando Sousa	Abu Dhabi becomes headquarters of International Council on Archives UAE -
outro	Manuel Fernandes	"... não era suposto"
outro	dpires	teste1
outro	David Marques Pires	teste
outro	rodrigues	Desenhos

Figura 3.21: Blundes apresentado ao utilizador no weebox

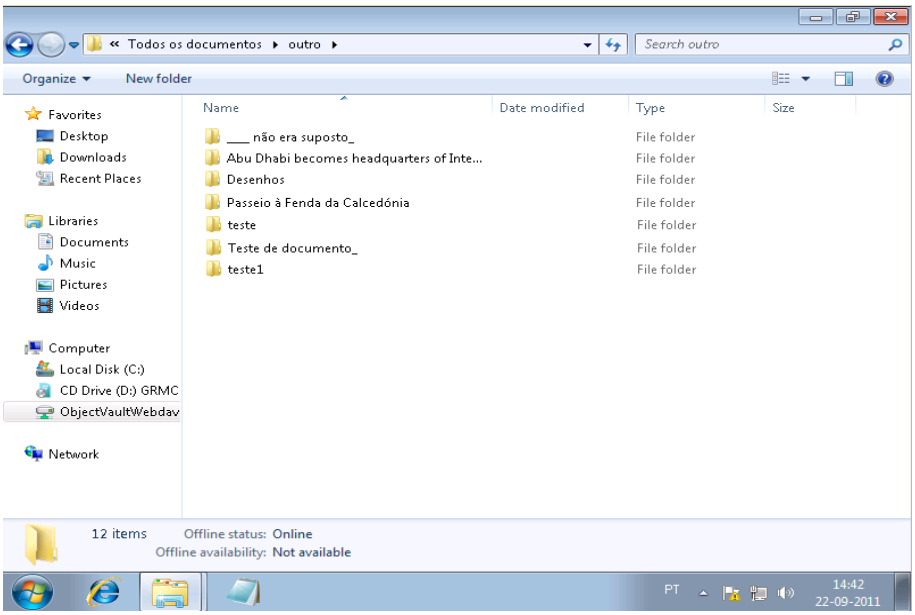

Figura 3.22: Blundes apresentado ao utilizador no WebDav

Na análise das figuras apresentadas em cima podemos observar alterações nos nomes de algumas pastas, que se devem ao Windows não permitir alguns caracteres. Apesar de o nome da pasta original aparecer correctamente o erro ocorria quando se pretendia ter acesso a essa pasta, o protocolo HTTP envia uma mensagem com o erro indicando que não conseguia localizar a pasta. A solução encontrada para este problema foi substituir esses caracteres não permitidos pelo protocolo HTTP pelo carácter "_".

3.4.5 Visualizar ficheiro

A funcionalidade visualizar ficheiro, no sistema weebox permite ao utilizador não só visualizar os ficheiros existentes no *bundle* como também visualizar os metadados do respectivo *bundle*. A opção de visualizar ficheiro no weebox permite ao utilizador abrir o ficheiro ou mesmo fazer o *download*. Na figura 3.23 podemos ver como são apresentados ao utilizador os metadados do *bundle* e o ficheiro. A visualização do ficheiro só é possível após seleccionar a opção "Descarregar".

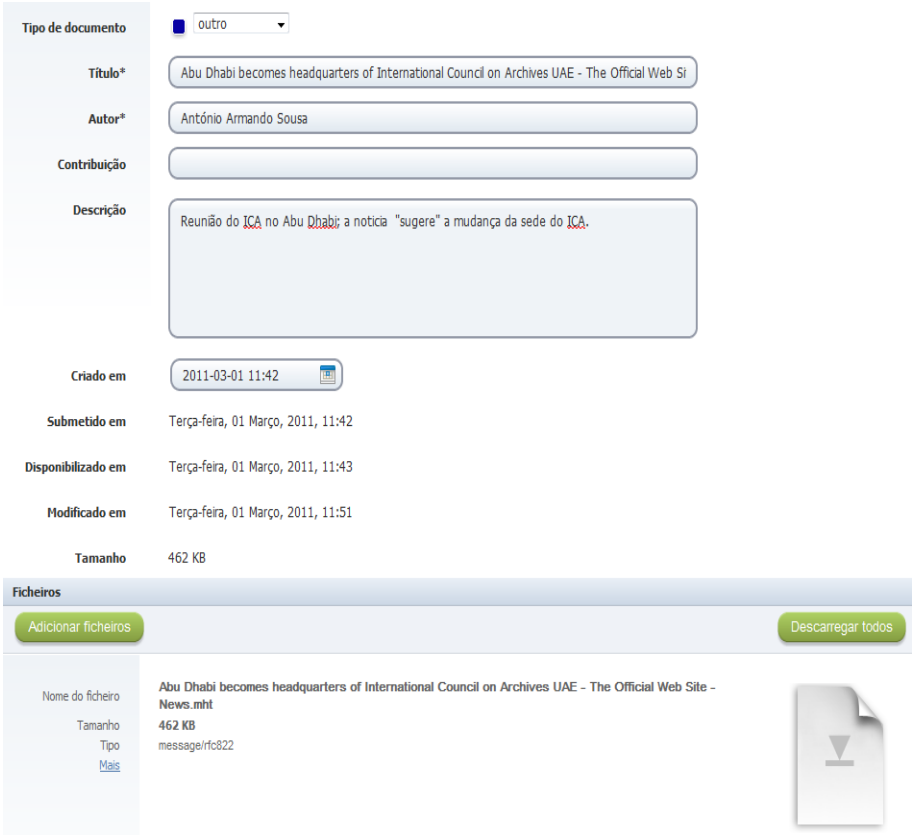

Figura 3.23: Apresentação dos metadados e do ficheiro no weebox

A opção visualizar ficheiro corresponde ao quarto nível implementado no WebDav. Na figura 3.24 podemos observar como são apresentados ao utilizador.

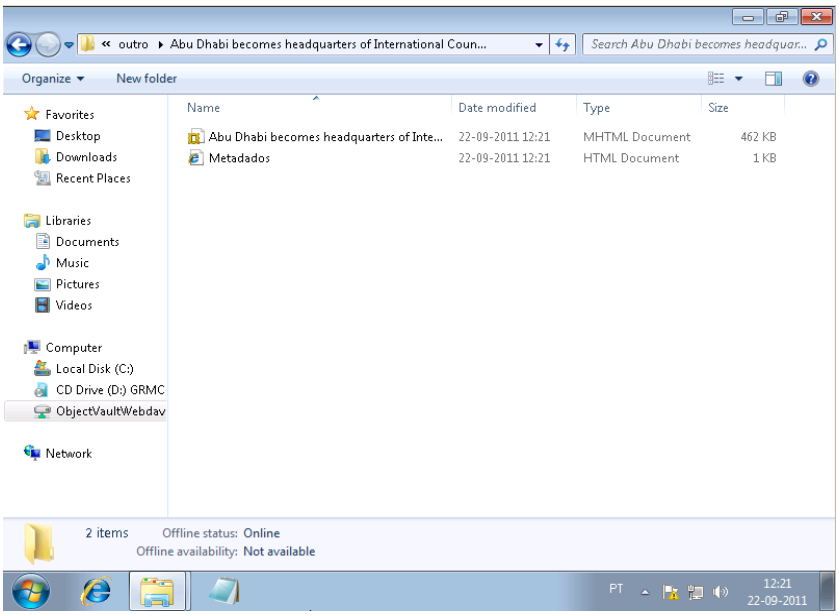

Figura 3.24: Ficheiros existentes numa pasta

Como se pode verificar na figura acima, é apresentado ao utilizador mais um ficheiro ("Metadados.html") do que os ficheiros que se encontram na figura 3.23. A consulta de um ficheiro usando o weebox permite-nos ter acesso aos metadados do *bundle* o que não acontece no WebDav. A solução encontrada para resolver esta divergência é adicionar um ficheiro "Metadados.html" que irá criar uma página html com a informação do *bundle*, figura 3.25, que será apresentado ao utilizador. Este ficheiro vai permitir visualizar os dados referentes a cada *bundle* existente nas pastas. Com este ficheiro pretende-se dar ao utilizador a opção de poder visualizar os metadados referentes a um determinado ficheiro duma forma simples e rápida. Este ficheiro tem a vantagem de permitir oferecer ao utilizador o acesso à informação existente no *bundle* sem ser necessário ter acesso ao weebox, sendo que no ficheiro "Metadados.html" também é passado o *link* do respectivo *bundle* para o caso do utilizador querer verificar o *bundle* no weebox.

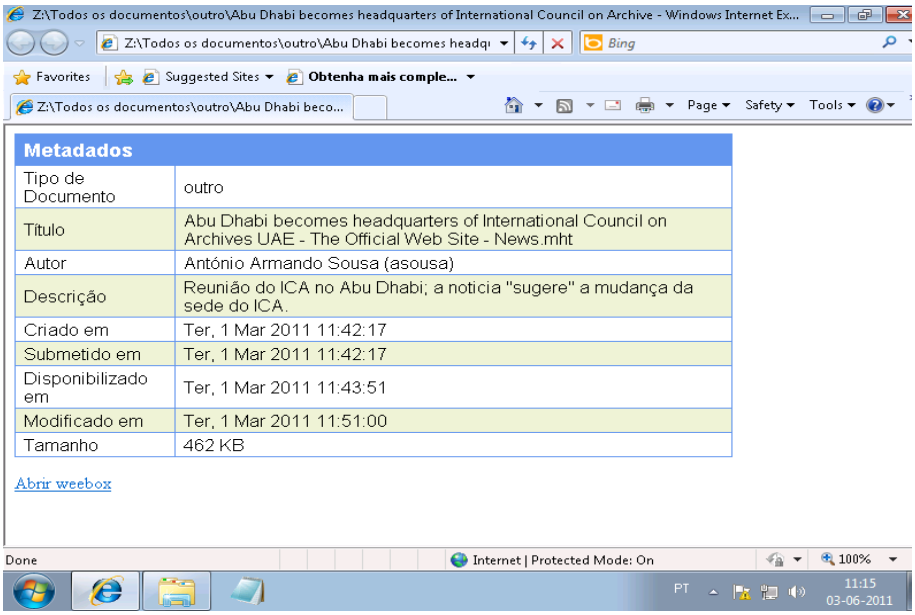

Figura 3.25: HTML criado como os metadados do bundle

Apesar de adicionar o ficheiro "Metadados.html", esta solução não vai convergir com os resultados apresentados pelo weebox, devido ao ficheiro ser apenas adicionado quando o sistema apresenta a listagem de ficheiros existentes num *bundle* e não adicionado na lista de ficheiros do *bundle*.

Descrição do *use case* para Visualizar ficheiro

1. O use case começa quando o utilizador selecciona um ficheiros;

2. O sistema verifica o tipo de ficheiros:

   2.1 Se for um ficheiro normal, o sistema carrega o ficheiro;

   2.2 Se for "Metadado.html", o sistema cria uma página html com os metadados.

3. O sistema apresenta o ficheiro ao utilizador.

O diagrama de actividade, figura 3.26, permite ter uma noção geral do percurso efectuado para a possível visualização dos níveis de pastas criados e a operação realizada pelo sistema após a selecção dum ficheiro. Como se pode verificar no diagrama existem dois tipos de ficheiros, os ficheiros normais e o "Metadados.html", dependendo do ficheiro que for seleccionado as operações são diferentes.

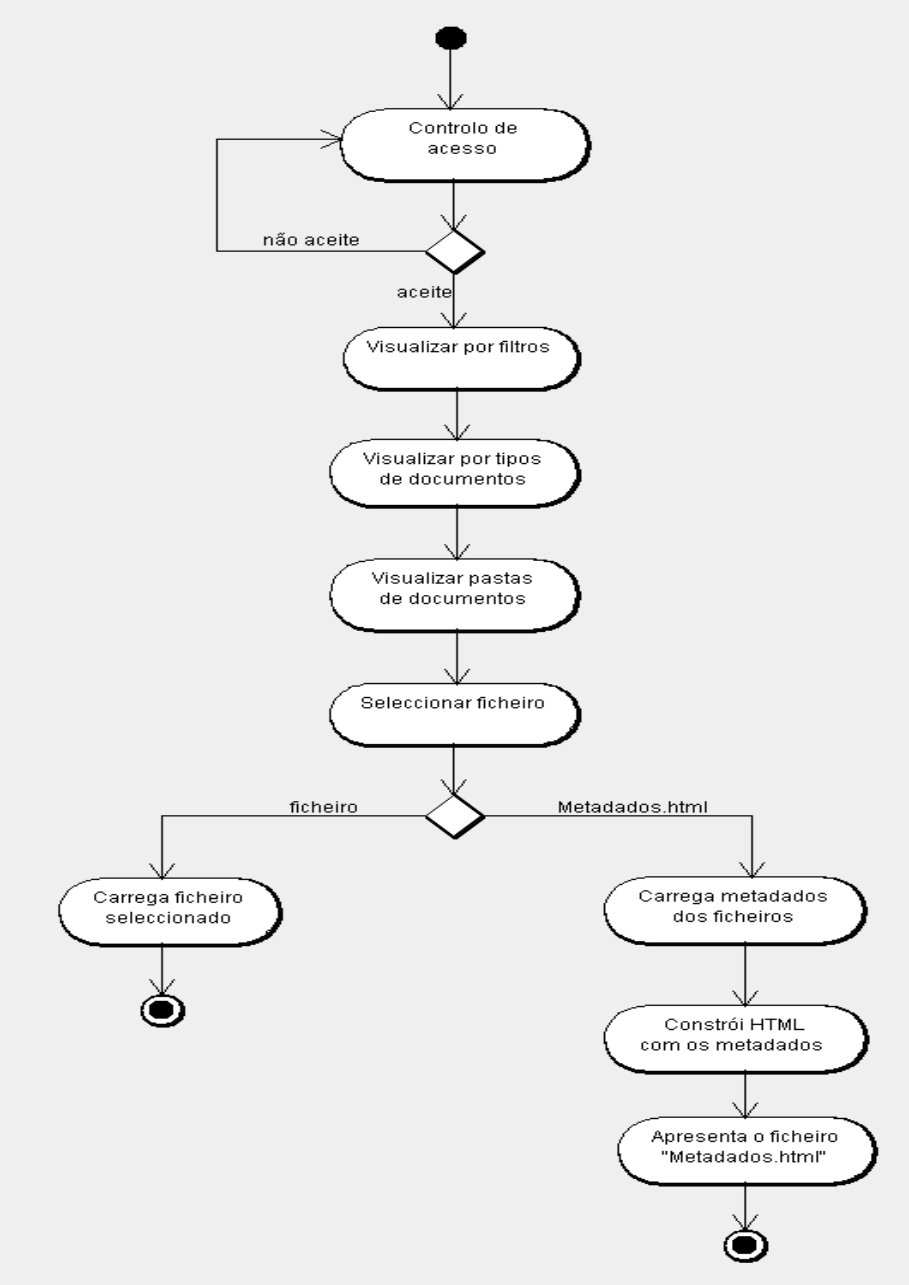

Figura 3.26: Diagrama de Actividade de Visualizar ficheiro

3.4.6 Editar nome da pasta

As funcionalidades básicas do sistema consistem na possibilidade de editar pastas e ficheiros existentes no arquivo bem como copiar ficheiros. O editar nome no weebox permite não só editar o nome da pasta mas também os seus metadados.

Para a realização destas funcionalidades básicas no WebDav é preciso usar as características *bundleId* e *fileId*. O *fileId* corresponde a um identificador de ficheiro, sendo que cada ficheiro tem o seu *fileId* e o mesmo é diferente para todos os ficheiros.

A operação editar nome da pasta no WebDav, apenas permite editar o nome da pasta e não os seus metadados. Apesar do WebDav não apresentar os metadados e assim não ser possível alterá-los, essa limitação é resolvida através do ficheiro "Metadados.html" descrito na secção 3.4.5, com a apresentação de um *link* que vai permitir ao utilizador abrir o respectivo *bundle* no weebox.

Outra limitação do protocolo WebDav é não permitir que dois *bundles* no mesmo nível tenham o mesmo nome o que é permitido no weebox.

Descrição do *use case* de Editar nome da pasta

1. O use case começa quando o utilizador edita o nome da pasta;

2. O utilizador introduz o nome da pasta;

3. O sistema verifica se já existe esse nome:

    3.1 Se já existe, é enviada uma mensagem ao utilizador a informar que já existe uma pasta com esse nome;

    3.2 Se não existe, o sistema cria uma nova pasta com esse nome.

4. O sistema copia os ficheiros existentes na pasta antiga para a pasta actual;

5. O sistema elimina a pasta anterior;

6. O sistema actualiza a lista de pastas;

7. É apresentado ao utilizador a lista de pastas actualizadas.

O diagrama de actividade, figura 3.27, permite ter uma noção geral das operações realizadas pelo sistema para a tarefa de editar pasta.

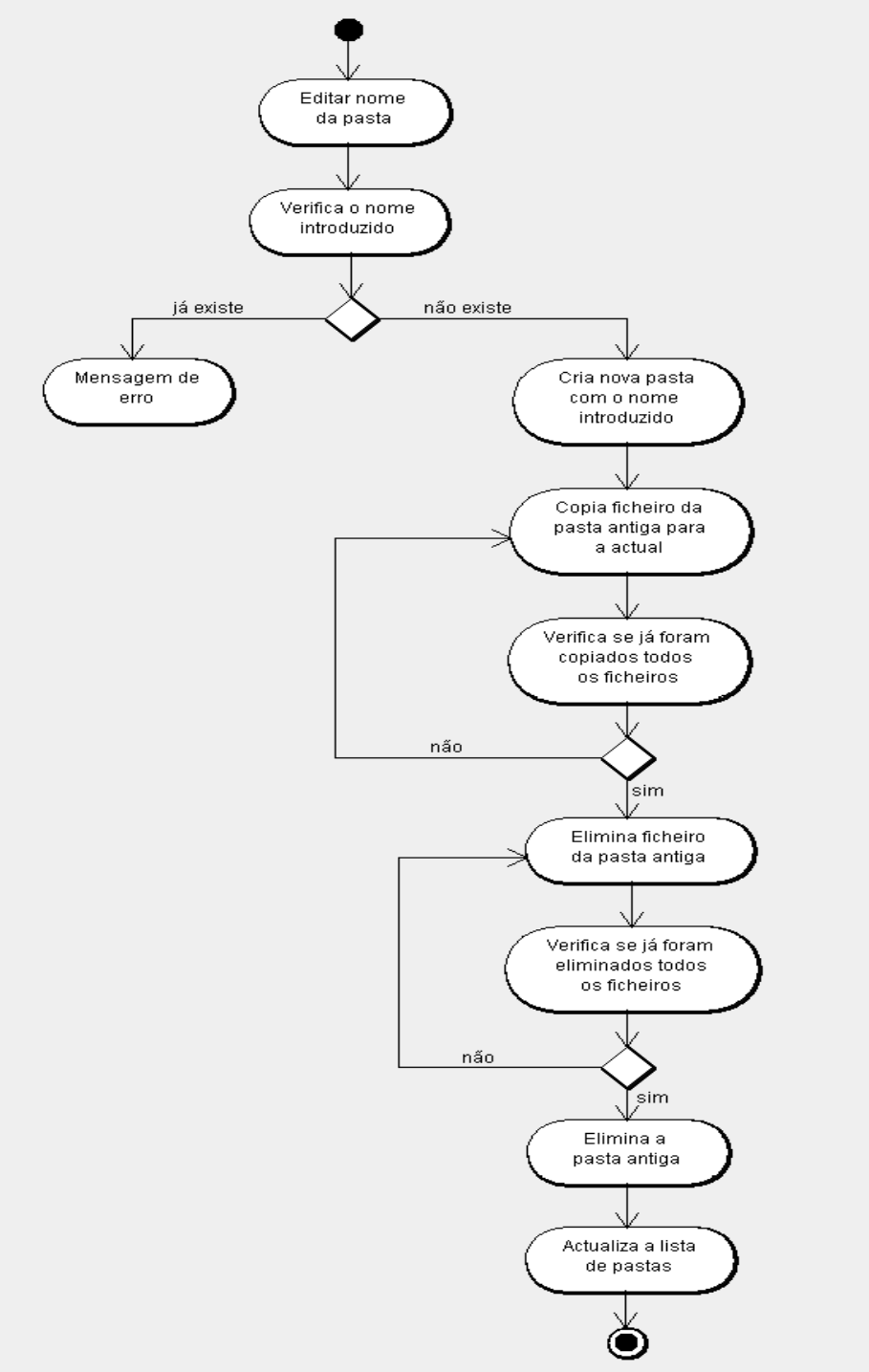

Figura 3.27: Diagrama de Actividade de Editar nome da pasta

A funcionalidade de editar pastas consiste apenas na alteração do nome da pasta.

A realização desta tarefa passa por quatro passos:

- Criar pasta com novo nome;
- Copiar ficheiros da pasta antiga para a nova;
- Eliminar ficheiros da pasta antiga;
- Eliminar pasta antiga.

Durante os passos mencionados em cima, os metadados serão actualizados directamente pelo sistema, como por exemplo a localização da pasta (tipo), o autor da mesma, o título e a data da sua actualização.

Na realização desta funcionalidade foi detectada uma divergência entre os resultados apresentados no weebox e no WebDav. No weebox o sistema actualiza o nome da pasta mantendo as versões anteriores existentes na pasta antiga. No WebDav é criada uma nova pasta com o nome introduzido pelo utilizador e copia, um a um, os ficheiros existentes na pasta a editar, após a operação de copiar ficheiros ter terminado o WebDav elimina esses ficheiros um a um da pasta anterior e por fim elimina a pasta antiga. A realização destes passos faz com que a pasta editada apenas irá conter a última versão e não todas as versões existentes como faz o weebox. Este problema acontece devido ao protocolo WebDav enviar ao sistema weebox os passos intermédios, um de cada vez (específico do protocolo), e não numa só operação, copiando assim apenas a última versão existente no weebox e não conservando assim as versões como faz o weebox. Uma possível solução para este problema é apresentada na secção 5.2.

3.4.7 Editar ficheiro

A funcionalidade editar ficheiro é dividida em duas operações: editar nome do ficheiro e editar conteúdo do ficheiro. Estas duas operações no weebox são realizadas da mesma maneira, sendo necessário primeiramente fazer o *download* do ficheiro e após editar o nome do ficheiro ou o conteúdo é preciso adicionar o ficheiro na pasta.

Descrição do *use case* de Editar ficheiro

1. O use case começa quando o utilizador edita um ficheiro;
2. O utilizador introduz o novo nome do ficheiro ou edita o conteúdo do ficheiro;
3. O sistema calcula o FileId do ficheiro;
4. O sistema actualiza o ficheiro;

5. O sistema actualiza a lista de ficheiros;

6. O sistema apresenta ao utilizador a lista de ficheiros actualizada.

No diagrama de actividade, figura 3.28, podemos verificar os passos efectuados pelo sistema para a operação de editar ficheiro.

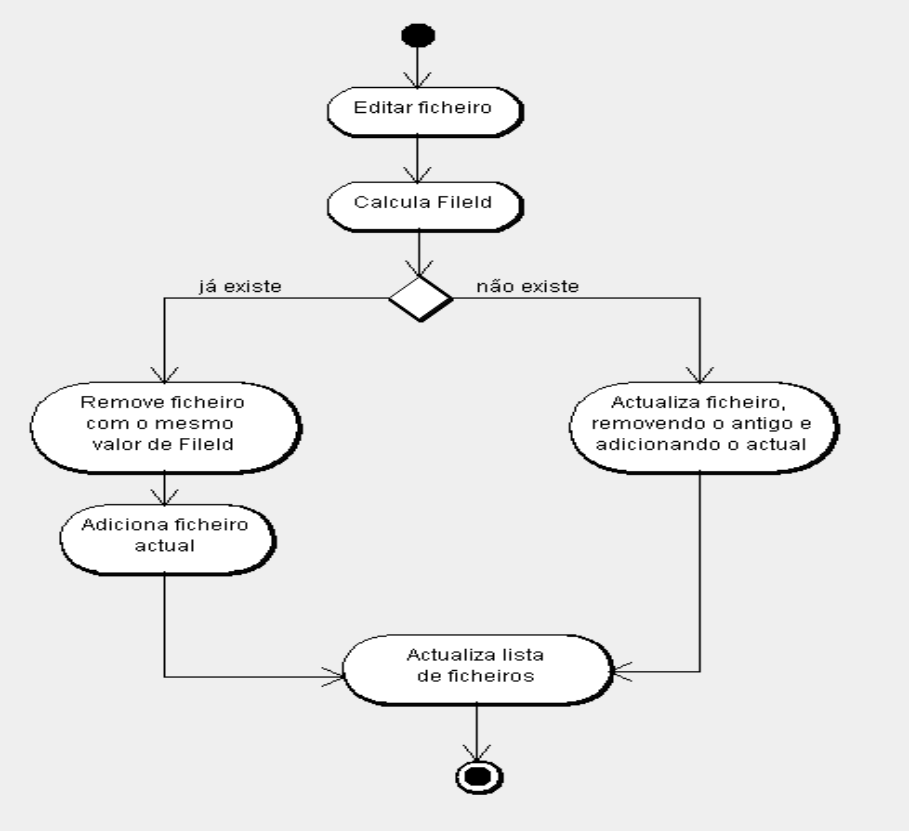

Figura 3.28: Diagrama de Actividade de Editar ficheiro

A funcionalidade de editar ficheiro consiste em alterar o nome do ficheiro ou alterar o conteúdo do ficheiro. A operação de editar ficheiros apesar de também usar os passos utilizados para o editar pasta é mais complexa, pois para editar ficheiros será preciso usar a informação existente no *fileId*. Resumidamente o valor do *fileId* é criado fazendo o cálculo do *checksum* do ficheiro, mas para esse cálculo apenas é considerado o conteúdo existente no ficheiro. Para o cálculo do *checksum* é usado o algoritmo MD5.

Os passos principais usados para editar ficheiros são:

- Criar ficheiro novo;

- Copiar conteúdo do ficheiro antigo para o novo;
- Calcular o *fileId* do ficheiro;
- Verificar de já existe esse *fileId*;
- Guardar ficheiro novo.

Relativamente ao editar o conteúdo do ficheiro, essa operação é bastante simples, pois após a alteração do conteúdo do ficheiro é calculado o novo valor do *fileId* e o ficheiro é adicionado na pasta, isso é realizado fazendo um *update* ao ficheiro apagando o ficheiro antigo e adicionando o novo. Na operação de editar o nome do ficheiro o processo é diferente, porque como o valor do *fileId* é apenas calculado sobre o conteúdo do ficheiro, ao alterarmos apenas o nome do ficheiro ele irá conter o mesmo valor do *fileId* do ficheiro antigo, mas na realidade é um novo ficheiro.

Um problema encontrado ao editar o nome do ficheiro era quando se verificava se já existia o *fileId* na lista de *fileId*. Como o valor do *fileId* não era alterado apesar de ter sido alterado o nome do ficheiro, o sistema não adicionava o ficheiro porque ele não permite adicionar um ficheiro com o mesmo valor de *fileId* que já exista na lista. A solução usada para ultrapassar este problema consiste em eliminar primeiro o ficheiro e depois adicionar o ficheiro novo em lugar de fazer o *update* do ficheiro como se faz para o editar do conteúdo do ficheiro.

3.4.8 Copiar ficheiro

A funcionalidade de copiar ficheiro no weebox apenas consiste em escolher o documento para onde se pretende copiar um ficheiro e seleccionar o botão "Adicionar ficheiros" para copiar um novo ficheiro para a pasta.

Descrição do *use case* Copiar ficheiro

1. O use case começa quando o utilizador copia um ficheiro para um *bundle*;
2. O sistema verifica se existe algum ficheiro com o mesmo nome:

   2.1 Se já existe, é enviado uma mensagem informando que já existe um ficheiro com esse nome;

   2.2 Se não existe adiciona o ficheiro.
3. O sistema actualiza a lista de ficheiros;
4. O sistema apresenta ao utilizador a lista de ficheiros actualizada.

Caminhos alternativos: a qualquer momento o utilizador pode cancelar a operação.

O diagrama de actividade, figura 3.29, permite ter uma visão geral das operações realizadas pelo sistema para a execução da funcionalidade copiar ficheiro.

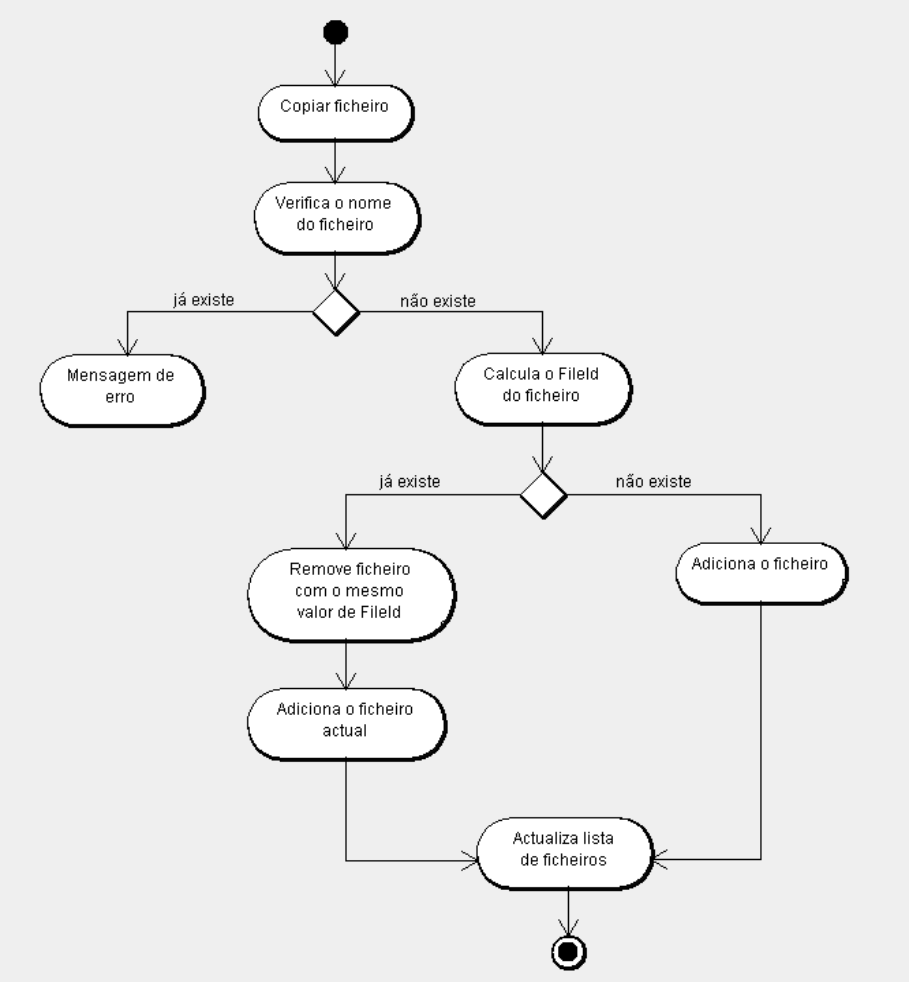

Figura 3.29: Diagrama de Actividade de Copiar ficheiro

Na funcionalidade copiar ficheiro foi preciso ter atenção a algumas situações que o sistema trata como inválidas não permitindo a sua realização. Por exemplo, o WebDav não permite ter dois ficheiros no mesmo sítio com o mesmo nome e o weebox não permite ter dois ficheiros com o mesmo conteúdo no mesmo *bundle*. No caso do utilizador copiar um ficheiro para o WebDav onde já se encontra um ficheiro com o mesmo nome ou com o mesmo conteúdo, é enviada uma mensagem ao utilizador a informar que já existe esse ficheiro no sistema, dando a possibilidade do utilizador cancelar a operação ou alterar o ficheiro, guardando o ficheiro copiado e eliminando o já existente.

Nesta operação de copiar ficheiro foi detectado um bug (encontrado no código

usado do WebDav-servlet, secção 3.3). Quando o sistema operativo Windows copiava um ficheiro era criado com outro ficheiro de conteúdo vazio (0 bytes). Isso devia-se a que se um método WebDav, com um corpo, é chamado num recurso inválido ou uma solicitação *PUT* é autenticada no fluxo de entrada e não é consumida iria permanecer até ao próximo pedido, o que leva a manipulação de dados incorrectos. Por exemplo, este comportamento acontece na criação de arquivos com o Windows. O Windows verifica via *PROPFIND* a existência dum recurso antes de criá-lo. No entanto, o *PROPFIND* tem um corpo que nunca é consumido pelo *servlet* por isso o recurso é libertado. O Windows cria um arquivo e um corpo vazio com o comando *PUT*. O método *PUT* solicita o fluxo de entrada que contém o corpo XML PROFIND [9].

Este bug é originado no Windows devido a ele usar o servidor classe 2 (secção 3.1.3 tabela 3.1), pois como usa o método de bloqueio ao copiar um ficheiro o recurso era bloqueado não permitindo copiar ficheiros. Esse *bug* foi corrigido desbloqueando o recurso, como está referido no RFC 2518. Os pedidos de bloqueio devem ter um corpo XML que contém um elemento proprietário para essa solicitação de bloqueio, salvo se for uma solicitação de actualização. As alterações efectuadas no código do projecto *open-source* para a correcção deste *bug* já se encontram disponíveis para os utilizadores actualizarem o seu código.

3.4.9 Criar pasta

As funcionalidades avançadas do sistema baseiam-se na possibilidade de criar pastas e ficheiros. No weebox a funcionalidade de criar pasta encontra-se logo na página principal, bastando carregar no botão "Carregar ficheiros" e seleccionar a opção criar documento vazio.

No WebDav a funcionalidade criar pasta, pode ser realizada de duas formas: criar uma nova pasta ou copiar uma pasta. Esta funcionalidade vai estar sujeita à limitação do protocolo WebDav de não permitir ter dois nomes no mesmo nível na hierarquia.

Descrição do *use case* Criar pasta

1. O use case começa quando o utilizador cria uma nova pasta;

2. O utilizador introduz o nome da pasta:

   2.1 Se já existe, é apresentada uma mensagem ao utilizador que não pode criar uma pasta com esse nome;

   2.2 Se não existe, a pasta é criada e guardada no sistema.

3. A lista das pastas é actualizada e apresentada ao utilizador.

No diagrama de actividades, figura 3.30, são apresentados os passos realizados pelo sistema quando um utilizador pretende criar uma pasta.

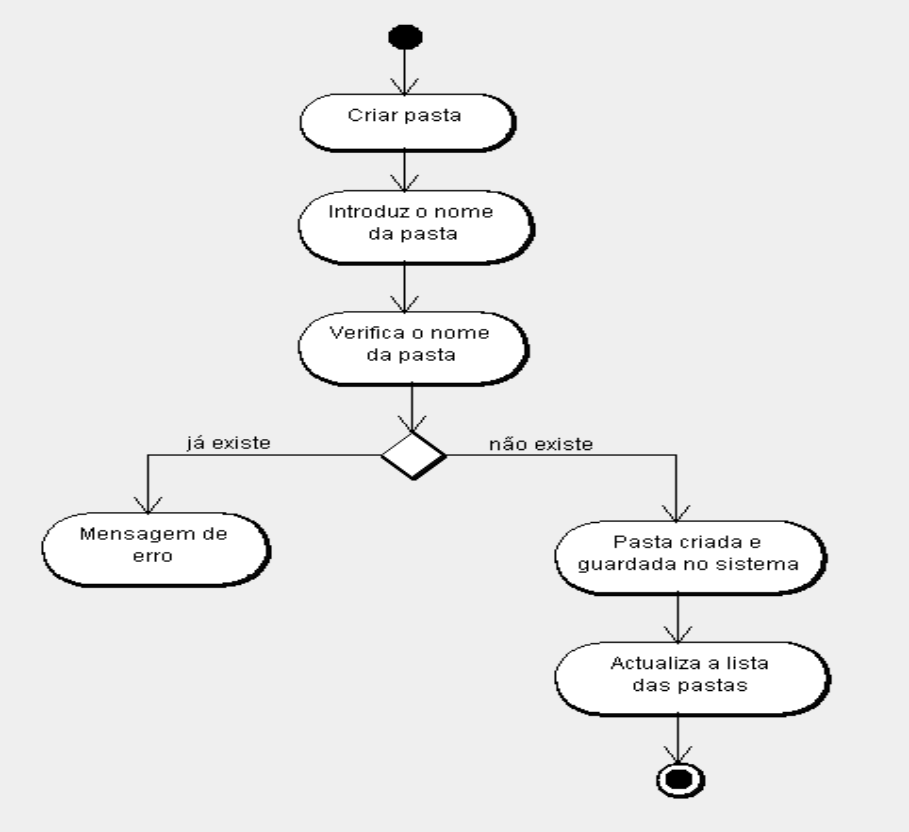

Figura 3.30: Diagrama de Actividade de Criar pasta

Uma pasta (*bundle*), como referido na secção 3.1.2, é constituída por metadados do *bundle*, tendo atributos obrigatórios para a sua criação, figura 3.1. Sendo assim ao criar uma nova pasta no sistema também é preciso preencher esses atributos. Os metadados relativos as datas e à localização da pasta serão preenchidos directamente pelo sistema, enquanto que o título da mesma será introduzido pelo utilizador. Ao criar uma nova pasta também é adicionada uma permissão sobre a pasta, devido à pasta ser criada fora do weebox, a permissão passada será de "privado", ficando essa mesma pasta somente disponível ao utilizador.

O armazenamento da pasta no weebox é feito através da *path*, quer isto dizer, como o WebDav tem uma visão dos conteúdos do repositório sob a forma dum sistema de pastas, a *path* é constituída pela hierarquia das pastas de onde nos encontramos, por exemplo se quisermos adicionar uma nova pasta em "Os meus documentos" e em "texto", temos que estar nesse local, ou seja, a *path* terá que ser "Os meus documentos/texto".

Na operação de criar nova pasta também foi preciso passar a informação do atributo arquivo a *"true"*, isto é importante pois caso contrário a pasta criada seria guardada em "Rascunho", pois as pastas armazenadas em "Rascunho" são as pastas que têm o campo de arquivo a *"false"*. Para resolver este passo é feito um teste sobre a *path* relativa ao local onde se está a criar a pasta, se na *path* não se encontrar o caminho para "Rascunho" o atributo arquivo é devolvido com o valor *"true"*. Por omissão a informação no atributo arquivo é *"false"*.

Há uma diferença entre criar pastas pelo WebDav ou pelo weebox. O weebox permite criar pastas com o mesmo nome e o WebDav não. Apesar de ser possível criar pastas com o mesmo nome no weebox, isso não vai afectar o funcionamento no WebDav, pois o WebDav ao apresentar as pastas existentes vai adicionar uma versão a uma dessas pastas. Por exemplo se tivermos duas pastas com o nome de "Teste" essas pastas são apresentadas ao utilizador como "Teste" e "Teste_1". O problema de criação de pastas com o mesmo nome só se verifica se essas pastas se encontrarem na mesma localização.

3.4.10 Criar ficheiro

O criar ficheiro no weebox é semelhante à funcionalidade copiar ficheiro, descrito na secção 3.4.8. Devido ao weebox ser um sistema de repositório não é permitido criar ficheiros mas apenas adicionar ficheiros.

Descrição do *use case* Criar ficheiro

1. O use case começa quando o utilizador cria um novo ficheiro;

2. O utilizador introduz o nome do ficheiro e adiciona o conteúdo;

3. O sistema calcula o FileId do ficheiro e verifica esse valor:

   3.1 Se já existe, é apresentada uma mensagem ao utilizador informando que já existe esse ficheiro;

   3.2 Se não existir, o ficheiro é criado e guardado no sistema.

4. A lista de documentos é actualizada e apresentada ao utilizador.

No diagrama de actividades apresentado de seguida, figura 3.31, podemos verificar as operações realizadas pelo sistema para a funcionalidade de criar ficheiro.

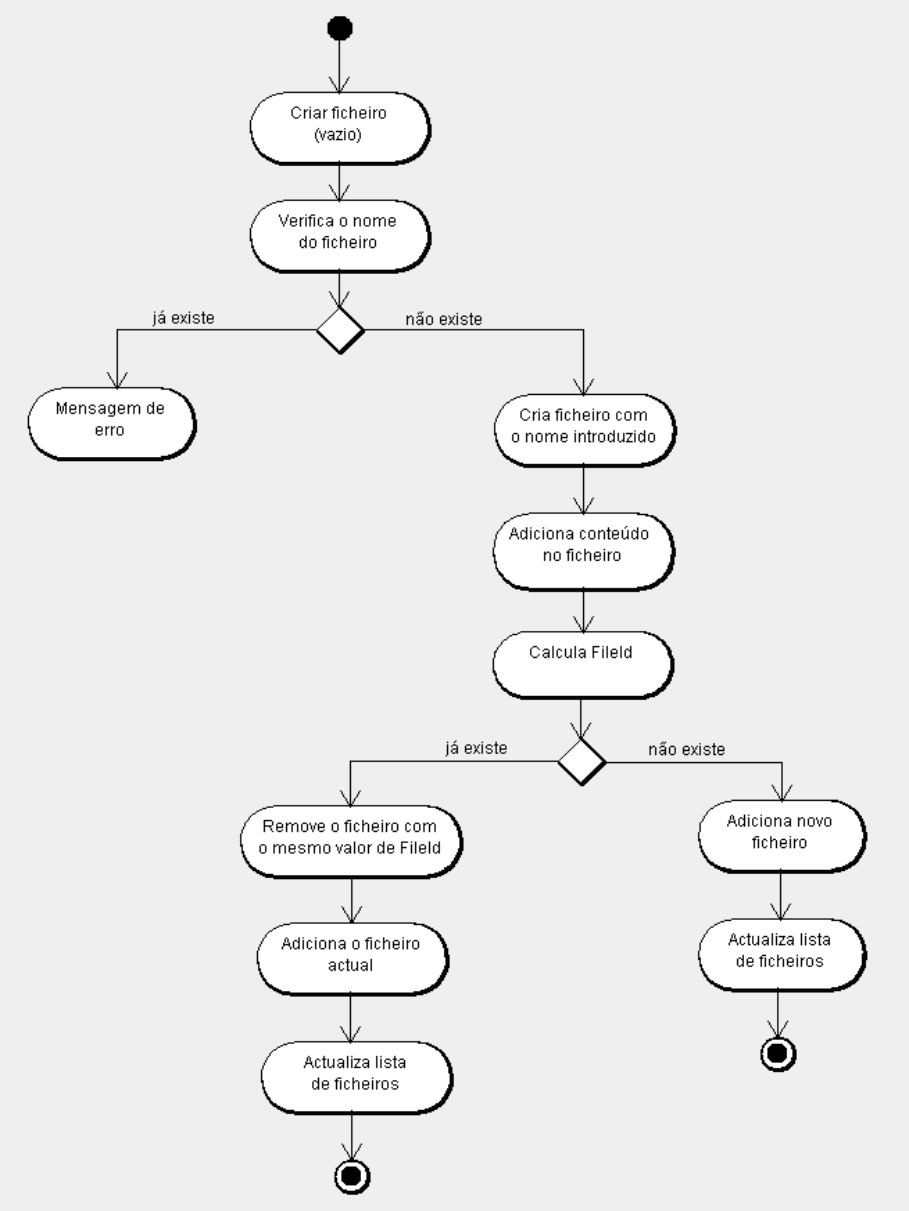

Figura 3.31: Diagrama de Actividade de Criar ficheiro

Na funcionalidade criar ficheiro também é verificado se já existe algum ficheiro com o mesmo nome, se já existir não é possível adicionar o ficheiro. Tal como as pastas que têm os seus metadados também os ficheiros têm a sua informação que neste caso é o *fileId*. Ao adicionar um ficheiro, o seu *fileId* é calculado e depois adicionado ao *bundle* correspondente. Ao criar um novo ficheiro num *bundle* é feito um *update* do mesmo, ou seja é adicionado o novo ficheiro e é calculado um novo *bundleId* para o *bundle*. Um cuidado a ter ao adicionar um novo ficheiro é

que ao criar um novo *bundleId* para o *bundle* os ficheiros que existiam nesse *bundle* vão ser perdidos pois fazem referência a um *bundleId* antigo. Para resolver este problema, após ser adicionado o novo ficheiro e calculado o novo valor de *bundleId* é actualizado o valor de *bundleId* dos ficheiros que já existiam nesse *bundle*, assim garantimos que os ficheiros que já existiam não são perdidos.

3.4.11 Eliminar pasta/ficheiro

A eliminação de pastas no weebox é realizada seleccionando a pasta a eliminar e carregando no botão "Eliminar". Na eliminação do ficheiro é necessário estar dentro do *bundle* e no botão "Descarregar" seleccionar a opção eliminar ficheiro.

No WebDav a eliminação de ficheiros é uma operação automática enquanto que a eliminação de pastas, executa em primeiro a eliminação dos ficheiros que existem na pasta e só depois elimina a pasta propriamente dita.

Descrição do *use case* Eliminar pasta/ficheiro

1. O use case começa quando o utilizador elimina uma pasta ou ficheiro;

2. O sistema verifica o que o utilizador pretende eliminar:

    2.1 Se for uma pasta, o sistema elimina a pasta seleccionada;

        2.1.1 O sistema actualiza a lista de pastas;

    2.2 Se for um ficheiro, o sistema elimina o ficheiro seleccionado;

        2.2.1 O sistema actualiza a lista de ficheiros.

No diagrama de actividades apresentado de seguida, figura 3.32, podemos verificar as operações realizadas pelo sistema para as tarefas eliminar pasta ou eliminar ficheiro.

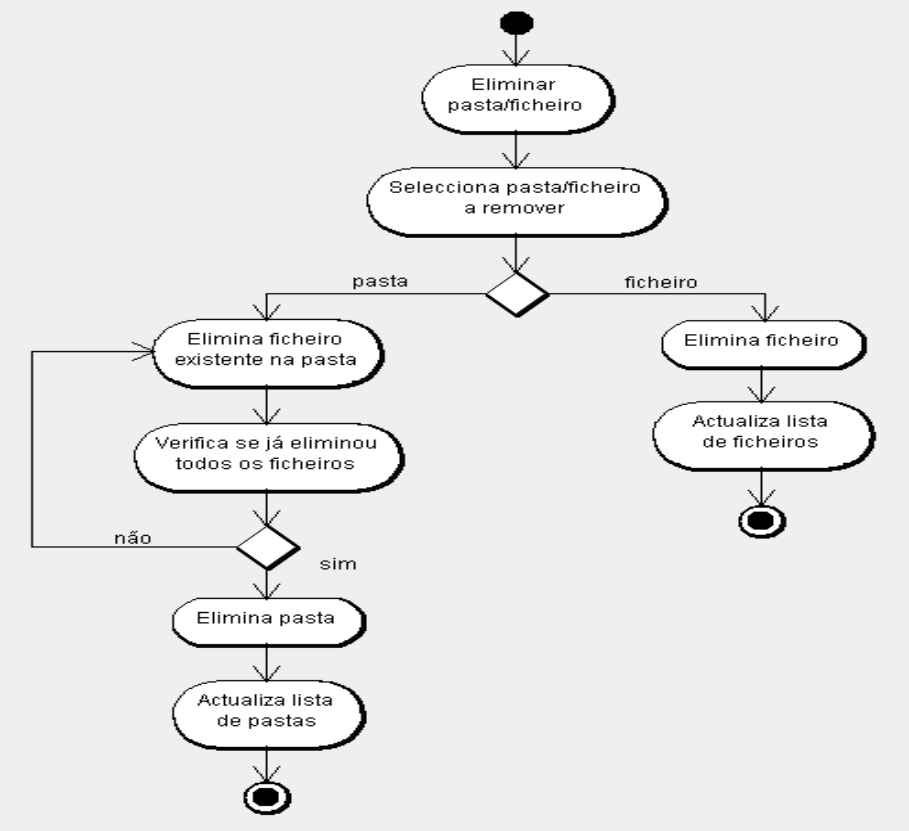

Figura 3.32: Diagrama de Actividade de Eliminar pasta/ficheiro

Na operação de eliminação de *bundle* é preciso ter em conta o que se pretende eliminar, pois se pretendermos eliminar um ficheiro não podemos usar a informação que está presente no *bundleId* mas sim a informação do *fileId*. O *fileId* corresponde à identificação de cada ficheiro, sendo um identificador único para cada ficheiro. Se fosse utilizado o *bundleId* o que seria eliminado era o *bundle* e não apenas o ficheiro pretendido. Isso poderia trazer problemas pois se ao eliminarmos um ficheiro o que aconteceria era que seriam eliminados todos os ficheiros existentes no *bundle*. A solução para esta operação de eliminação é usar o identificador do que pretendemos eliminar, se for um *bundle* então usamos o *bundleId* e irá eliminar os ficheiros todos existentes no *bundle*, se pretendermos apenas eliminar um ficheiro, então usamos o seu *fileId*, e assim só será eliminado esse mesmo ficheiro.

Na eliminação dum *bundle* o WebDav não permite eliminar directamente o *bundle*, para isso é preciso em primeiro eliminar todos os ficheiros existentes dentro desse *bundle* e por fim será eliminado o *bundle* pretendido.

3.4.12 Considerações relevantes

Na implementação das funcionalidades mencionadas nas secções anteriores foi preciso levar em conta algumas considerações para não afectar o funcionamento do sistema WebDav/weebox, essas considerações foram tomadas em especial atenção nas tarefas de criar/copiar/eliminar pastas ou ficheiros.

No caso de criar/copiar foi preciso antever os casos de comportamento não esperado. Esses comportamentos são relativos aos casos de tentativa de criar/copiar um ficheiro para um nível dos *bundles* e de criar/copiar *bundle* para o nível de ficheiros. Caso o utilizador pretenda realizar alguma destas operações mencionadas será então apresentada no ecrã uma mensagem de erro. Essa mensagem informará o utilizador de que não tem permissão para realizar essa tarefa. Esta mensagem de erro é uma mensagem suportada pelo protocolo WebDav.

Outra consideração tomada foi a de não permitir criar/copiar ficheiros para um *bundle* que não nos pertença. Apesar de ser possível partilharmos pastas entre vários utilizadores não é possível alterar o *bundle*, para isso antes de copiar/criar um novo ficheiro é verificado se o utilizador é o dono do *bundle*, se for o dono pode concretizar a operação, caso contrário essa operação não será permitida e será apresentada uma mensagem de erro com a informação de que não tem permissões sobre o *bundle*.

Tal como no caso de criar/copiar na operação de eliminar pasta/ficheiro também é verificado se o utilizador é o dono do *bundle*, e a conclusão desta tarefa também irá depender desta propriedade.

No início da realização destas funcionalidades foi encontrado um bug de *encoding*. O *encoding* consiste em converter a informação recebida num código compatível com um dispositivo ou programa, ou seja, o *encoding* é a maneira com que os conjuntos de caracteres são mapeados e manipulados pelas máquinas, seja um software, seja um *browser*, etc. O código usado pela maioria dos computadores para arquivos de texto é conhecido como ASCII (*American Standard Code for Information Interchange*). ASCII podem representar letras maiúsculas e minúsculas, caracteres alfabéticos, algarismos, sinais de pontuação e símbolos comuns. No problema encontrado neste projecto foi necessário construir uma função para converter os códigos recebidos para um código em que fosse possível apresentar os dados correctamente, pois os dados eram recebidos correctamente sendo que o erro apenas se detectava no envio desses dados. Para resolver este problema foi preciso realizar a conversão dos códigos de ISO-8859-1 [1] para UTF-8 [2], pois os dados recebidos vinham com o *encoding* ISO-8859-1 e ao apresentar os dados eles apareciam com erros, um desses erros era encontrado nas palavras que continham acentos.

[1] *norma da ISO que define a codificação do alfabeto latino, sendo também conhecido com Alfabeto Latino nº1 ou ISO Latim 1*

[2] *codificação Unicode criado por Ken Thompson e Rob Pike. Pode representar qualquer carácter universal padrão do Unicode e também é compatível com o ASCII.*

Capítulo 4

Testes de usabilidade

Após o termino do protocolo WebDav para o weebox, o sistema passou por um processo de testes de usabilidade. O objectivo da realização dos testes de usabilidade foi verificar a validação do sistema implementado no caso de estudo. A usabilidade dum produto refere-se à facilidade de utilização e aprendizagem por parte de utilizadores reais. Este capítulo serve para descrever os resultados dos testes e como os problemas detectados foram revolvidos.

O número de utilizadores a utilizar nos testes de usabilidade pode ser bastante pequeno, abaixo das 15 pessoas. Para fundamentar esta afirmação podemos analisar a seguinte formula que dá uma boa aproximação na procura de problemas de usabilidade [24]:

$$P(i) = N(1 - ((1 - \lambda)^i)) \tag{4.1}$$

Em que P são os problemas de usabilidade encontrados, i o número de utilizadores, N o número total de problemas da interface, e λ a probabilidade de encontrar um qualquer problema por um qualquer utilizador. Nos vários projectos estudados por Nielsen, o número médio de problemas da interface – N – era de 41, e o valor médio de λ tendia para 31%. Assumindo estes valores, a figura 4.1 ilustra a relação de problemas encontrados por número de utilizadores.

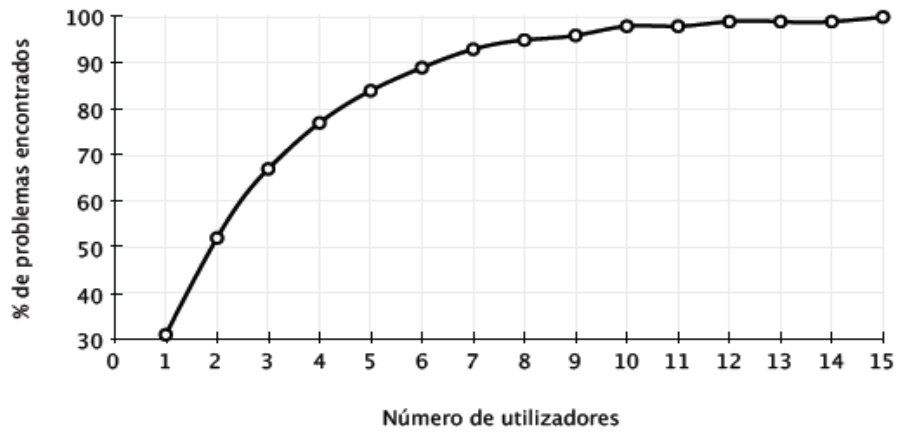

Figura 4.1: Percentagem de problemas encontrados por nº de utilizadores (N=41 e λ=0.31)

Como podemos analisar na figura acima, 15 utilizadores encontram em média 100% dos problemas, pelo que usar mais utilizadores não trará mais-valias. Por outro lado, a figura 4.2 demonstra o custo/beneficio para um projecto "típico"[1] de tamanho médio. O maior rácio de benefícios por custo é conseguido com 3 utilizadores, com mais o rácio começa a diminuir.

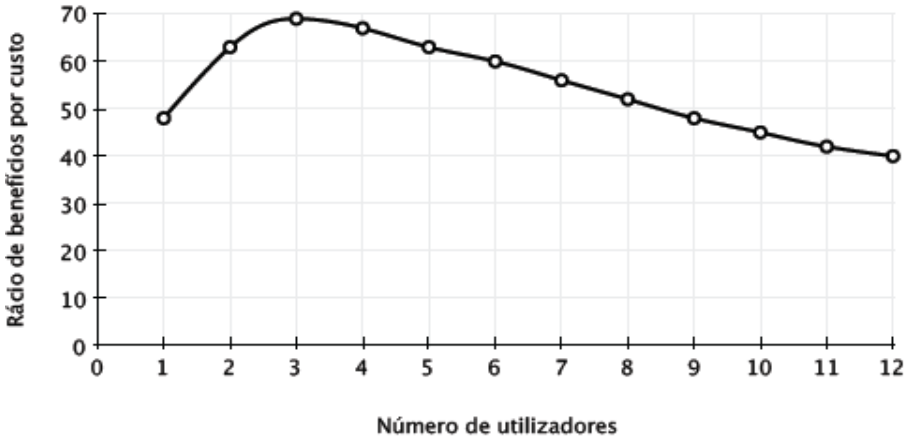

Figura 4.2: Rácio de benefício por custo de testes de usabilidade com diferente número de utilizadores para um projecto "típico" de tamanho médio

Recomenda-se que se aumente o número de utilizadores enquanto que o valor expectado dos benefícios superem o custo de executar o teste com mais utilizadores,

[1] O projecto "típico" de tamanho médio é descrito em [24]

sendo que em média este valor vai situar-se entre os 5 e os 10 utilizadores. Para a realização destes testes foram escolhidas seis pessoas que já tinham conhecimento do funcionamento do weebox. A selecção do grupo para a realização dos testes sobre o sistema WebDav/weebox foi baseado no princípio que para a utilização do sistema era preciso que as pessoas já tivessem um mínimo conhecimento do modo de funcionamento do sistema weebox, pois como o trabalho realizado nesta dissertação consiste em interligar o protocolo WebDav com o sistema weebox, o sistema desenvolvido serve principalmente para as pessoas que já usam o sistema weebox.

Os testes de usabilidade realizados consistem numa série de passos que foram pedidos aos inquiridos para realizarem no sistema, sendo medido o tempo de execução de cada passo e avaliado se o inquirido realizou o passo com sucesso e sem ajuda. Para além disso foi feito o levantamento de todos os erros e problemas que aconteceram.

A Figura 4.3 apresenta a descrição, a percentagem de sucesso, o tempo médio e as observações de cada um dos passos dos testes de usabilidade para o sistema WebDav/weebox. É de salientar que para a realização dos testes foi preciso realizar a pré-condição de instalar o WebDav no computador.

Nº	Tarefa	Sucesso	Tempo	Observações
1	Encontre no filtro "Os meus documentos" no tipo de documento "teste" a pasta "WebDavUsabilidade"	0%	2m 2s	Todos precisaram de ajuda para encontrar a pasta. Problema #1
2	Consulte os metadados do documento	100%	0m 8s	Dois demoraram mais tempo porque não perceberam que era um ficheiro html e abriram com o editor de texto
3	Volte ao menu principal	100%	0m 7s	Um não percebeu que o menu principal era a raiz do sistema, mas logo encontrou o caminho
4	Entre no filtro "Os meus documentos"	100%	0m 4s	
5	Seleccione um tipo de documento	83%	0m 2s	Foi detectado o erro #1
6	Crie uma pasta	100%	1m 3s	
7	Crie um ficheiro nessa pasta	100%	1m 2s	Um em vez de criar um ficheiro copiou realizando o passo 8
8	Copie um ficheiro do seu PC para essa pasta	100%	1m 5s	
9	Procure essa pasta no weebox	100%	0m 14s	Foi detectado o erro #2
10	Crie uma pasta no weebox e adicione um ficheiro	100%	0m 54s	
11	Procure essa pasta no WebDav	100%	0m 42s	
12	Volte ao menu principal	100%	1m 8s	
13	Entre no filtro "Rascunho" e no tipo de documento "teste" e edite o conteúdo do ficheiro "Conteudo" da pasta "EditConteudo"	83%	1m 5s	Foi detectado o erro #1
14	Procure a última versão desse ficheiro no weebox	100%	0m 35s	
15	Entre no tipo de documento "teste" e edite o nome da pasta "Editar" no filtro "Rascunho" do WebDav	100%	0m 50s	
16	Volte ao menu tipo de documento "teste"	100%	0m 3s	
17	Copie uma pasta do seu PC para o WebDav	100%	1m 19s	
18	Copie um documento do WebDav para o seu desktop	100%	0m 24s	
19	Copie uma pasta do WebDav para o seu desktop	100%	1m 15s	

Figura 4.3: Dados dos testes de usabilidade

Como podemos ver na figura em cima, no decorrer dos diversos testes de usabilidade foram levantados os problemas e erros encontrados, que podem ser consultados na Figura 4.4 e 4.5, respectivamente. O único problema encontrado e que originou uma eficácia de 0% foi o passo de procurar um ficheiro, que para ser encontrado obrigava a seleccionar a pasta de paginação, o que para os utilizadores não estava explicito, daí um resultado inesperado na sua eficácia. Dos erros detectados, a sua eficácia não foi de 100% devido aos ficheiros não aparecerem logo, sendo preciso fazer um *refresh* à página.

#	Descrição	Solução
1	Todos precisaram de ajuda para encontrar a pasta devido a não perceberem que se estava a usar paginação	Com o protocolo WebDav organiza as pastas por caracteres especiais, números e de seguida por ordem alfabética e o símbolo usada para a paginação foi o #, a solução passa por alterar esse símbolo e a paginação aparecer no fim da lista de pastas.

Figura 4.4: Problemas encontrados nos testes de usabilidade

#	Descrição
1	Ao seleccionar o tipo de documento, os ficheiros não apareceram, foi preciso fazer um refresh à página.
2	Ao copiar um ficheiro no WebDav, foi adicionado nos metadados informação errada no campo "Descrição"

Figura 4.5: Erros encontrados nos testes de usabilidade

Para além dos testes de usabilidade também foi feito um inquérito que consistia num grupo de perguntas divididas em dois grupos:

- O primeiro grupo é constituído por perguntas fechadas, que consiste numa resposta de escolha múltipla onde as perguntas se inseriam na facilidade de manipulação do sistema, na figura 4.6 podemos verificar a média das respostas a cada pergunta dada pelas pessoas que realizaram os testes, estando as respostas dentro de um intervalo de valores de 1 a 5, sendo 1: mau e 5: excelente. No anexo B, podemos consultar as respostas dadas pelos utilizadores às perguntas fechadas.

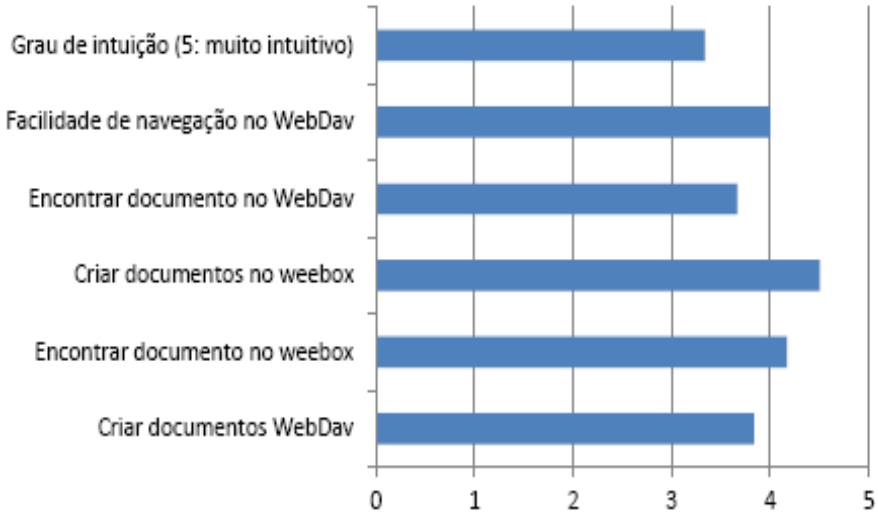

Figura 4.6: Dados das respostas fechadas realizadas

- O segundo grupo é constituído por perguntas abertas, que consistem em resposta onde a pessoa é livre de responder como quiser. As perguntas realizadas neste grupo são relativas à experiência que tiveram com o sistema, como por exemplo às dificuldades encontradas, sugestões de melhoria, etc. Em resumo poder-se-á dizer que os utilizadores ficaram satisfeitos com a experiência. No anexo C, podemos consultar as respostas dadas pelos utilizadores às perguntas abertas.

Optimização do sistema

A principal sugestão de melhoria apontada pelos utilizadores que realizaram os testes passa pela performance do sistema. Esta lentidão no sistema deve-se ao *framework* [1] usada pelo protocolo WebDav, principalmente nas funcionalidades de copiar ficheiro para o WebDav e vice-versa, ou seja, copiar um ficheiro do WebDav para o computador do utilizador. Esta situação acontece devido ao uso de ficheiros temporários para conseguir copiar o ficheiro, esta solução de ficheiros temporários deve-se ao facto do protocolo WebDav usar *InputStream/OutputStream* para os dados dos ficheiros e para a tarefa de copiar ficheiro é necessário saber o tamanho do ficheiro. A solução proposta passa por alterar o código da *framework*, para dar a informação do tamanho do ficheiro, conseguindo-se assim uma melhoria na optimização do sistema.

[1] é uma estrutura de suporte definida em que um outro projecto de software pode ser organizado e desenvolvido. Um *framework* pode incluir programas de suporte, bibliotecas de código, linguagens de *script* e outros softwares para auxiliar no desenvolvimento e unir diferentes componentes de um projecto de software.

Capítulo 5

Conclusão

O principal objectivo desta dissertação é explicar como é possível criar uma interface hierárquica para acesso a redes ontológicas de ficheiros. A análise do caso de estudo permitiu ter a noção das diferenças entre os dois sistemas usados para o desenvolvimento desta dissertação (WebDav e weebox). Estas diferenças encontram-se principalmente na organização de dados e nos passos realizados pelos sistemas na realização das funcionalidades.

5.1 Conclusões gerais

A análise do caso de estudo e a sua implementação usando o algoritmo definido na secção 2, permitiu provar que é possível modelar uma estrutura de ficheiros ontológica numa estrutura hierárquica. A interface criada para o acesso a redes ontológicas de ficheiros no caso de estudo, preservou toda a informação que estava contida no sistema original (weebox). Com a conclusão do sistema a implementar foram efectuados testes de usabilidade para verificar a usabilidade do sistema. Após o término dos testes de usabilidade realizados pelos utilizadores e uma análise aos resultados obtidos, verificou-se que os utilizadores validaram e com alto grau de aceitação a usabilidade do sistema (secção 4).

Conclusões particulares do caso de estudo

Na implementação do caso de estudo foram detectadas diferenças entre os sistemas, principalmente na organização dos dados e nos passos usados por cada sistema nas funcionalidades implementadas.

Com a análise à organização dos dados usada no weebox foi detectado que o

sistema trata dos seus dados através de id's (principalmente por *BundleId* que corresponde aos *bundles* e *FileId* que corresponde aos ficheiros). Para a apresentação final ao utilizador foi necessário proceder a uma tradução dos id's para o nome dos respectivos dados, sendo o resultado final uma hierarquia de pastas apresentando os nomes dos dados e não os seus id's como são guardados no sistema weebox. Outra limitação detectada no protocolo WebDav é não permitir ter dois *bundles* com o mesmo nome e no mesmo nível, o que não acontece no sistema weebox. A solução usada para contornar esta limitação do protocolo WebDav para não se perder dados existentes no sistema anterior, passa por adicionar uma versão no nome do bundle a apresentar ao utilizador.

Nas operações sobre as tarefas a realizar sobre os dados a principal diferença entre as operações usados pelo WebDav e weebox eram as tarefas de editar o nome da pasta e adicionar ficheiro. A diferença apresentada pelo WebDav e o weebox era apresentada nos passos intermédios a realizar nas operações sobre os dados, pois o WebDav envia cada acção individualmente para o weebox.

Na tarefa de editar nome da pasta, no weebox o utilizador ao editar o nome da pasta o sistema actualiza o novo nome da pasta, actualizando apenas o nome da pasta e mantendo as versões anteriores existentes na pasta antiga. Ao realizar esta tarefa de editar nome de pasta usando o protocolo WebDav, as operações realizadas pelo protocolo diferem do weebox. O protocolo WebDav cria uma nova pasta com o nome introduzido pelo utilizador e copia um a um os ficheiros existentes na pasta a editar, após a operação de copiar ficheiros ter terminado o WebDav elimina esses ficheiros um-a-um da pasta anterior e por fim elimina a pasta antiga. A realização destes passos faz com que a pasta editada contenha apenas a última versão e não todas as versões existentes como faz o weebox. Na secção 5.2 é apresentada uma solução para contornar este problema.

Outros mapeamentos de funcionalidades possíveis de ser implementadas seriam a de pesquisa avançada, como já é implementada pelo weebox. A ideia consistia num filtro chamado "Pesquisa" que teria a funcionalidade de pesquisar *bundles* através do seu nome. O seu funcionamento baseava-se na criação de uma pasta dentro do filtro, e o sistema iria copiar todos os *bundles* existentes no sistema que contivessem o nome da pasta criada nos metadados do *bundle*. Esta solução iria permitir ao utilizador uma pesquisa mais rápida dos ficheiros pelo sistema.

A criação da funcionalidade sugerida não invalida, nem modifica as conclusões desta dissertação.

5.2 Notas finais

As estruturas hierárquicas de ficheiros fazem parte do quotidiano dos utilizadores de computadores, devido ao seu elevado uso, tornando os métodos de adicionar ou mesmo pesquisar ficheiros intrínseco à experiência do computador pessoal.

O problema da modelação de estruturas ontológicas numa estrutura hierárquica prende-se principalmente com as restrições específicas de cada sistema e com os passos realizados por cada sistema nas operações sobre os dados. Apesar do caso de estudo comprovar ser possível modelar uma estrutura ontológica numa estrutura hierárquica, a solução não é completa, devido aos passos intermédios de cada acção não se mapearem directamente no sistema weebox. Na figura 5.1 é apresentado um exemplo duma funcionalidade onde acontece esta limitação.

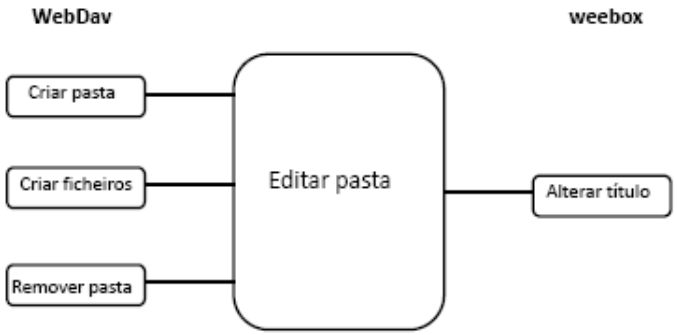

Figura 5.1: Passos realizados por cada sistema na funcionalidade Editar pasta

Um método mais avançado para resolver estes problemas entre as diferenças sobre os passos realizados pelos sistemas, passa por guardar numa lista de operações intermédias os passos realizados pelo protocolo WebDav e apenas fazer o *commit* no sistema weebox quando necessário. Este método consiste em guardar todos os passos intermédios usados pelo WebDav numa lista, e depois traduzir a série de passos intermédios numa só operação do weebox. Com este método na tarefa de editar nome da pasta apenas seria enviado uma operação ao weebox de editar o nome da pasta.

Bibliografia

[1] H. E. A. Al-Sawadi. An architecture for secure document flow & archival systems. pages 65–77, April 2009.

[2] P. Carpentier. Content-addressed storage (cas) explained. http://searchstorage.techtarget.com/feature/Content-addressed-storage-CAS-explained, October 2008.

[3] G. Clemm. Versioning extensions to webdav (web distributed authoring and versioning). http://www.ietf.org/rfc/rfc3253.txt, March 2002.

[4] G. Clemm. Web distributed authoring and versioning (webdav) access control protocol. http://www.ietf.org/rfc/rfc3744.txt, May 2004.

[5] O. C. M. B. Duarte. Topologias. http://www.gta.ufrj.br/grad/08_1/san/conceito_c.html, 2008.

[6] L. Dusseault. Http extensions for web distributed authoring and versioning (webdav). http://www.webdav.org/specs/rfc4918.html, June 2007.

[7] M. Eaddy, S. Rong, and J. Shapiro. Webdav project. http://www.cs.columbia.edu/~hgs/teaching/ais/1998/projects/WebDAV/webdav.html#_Toc406873714, December 1997.

[8] L. Faria. Safe office. pages 13–22, Outubro 2009.

[9] S. Felis. webdav servlet. http://sourceforge.net/mailarchive/forum.php?forum_name=webdav-servlet-general, March 2011.

[10] R. Fielding. Hypertext transfer protocol – http/1.1. http://www.ietf.org/rfc/rfc2616.txt, June 1999.

[11] B. W. Fitpatrick, B. Collins-Sussman, and C. M. Pilato. Client interoperability. http://svnbook.red-bean.com/en/1.5/svn.webdav.clients.html, 2008.

[12] J. Franks. Http authentication: Basic and digest access authentication. http://www.ietf.org/rfc/rfc2617.txt, June 1999.

[13] Y. Goland. Http design issues, lessons from webdav's property and depth head er experiences. http://lists.w3.org/Archives/Public/w3c-dist-auth/1998OctDec/0303.html, December 1998.

[14] hill hill hill hill. Content-addressable storage. http://wiki.hill.com/wiki/index.php?title=Content-addressable_storage, October 2008.

[15] D. Hollander, A. Layman, R. Tobin, H. S. Thompson, and T. Bray. Namespaces in xml 1.0 (third edition). http://www.w3.org/TR/REC-xml-names/, December 2009.

[16] IETF. The internet engineering task force (ietf). http://www.ietf.org/, 2007.

[17] S. Itkonen. Web-based distributed authoring and versioning (webdav). http://www.tml.tkk.fi/Studies/Tik-110.350/1999/Essays/webdav.html, April 1999.

[18] S. Jaroslavov. Webdav server compliance. http://www.webdavsystem.com/server/documentation, February 2011.

[19] A. Joaquim. Gestão documental ganha maturidade. http://www.semanainformatica.xl.pt/730/est/100.shtml, Fevereiro 2005.

[20] kioskea kioskea kioskea kioskea. O sistema de arquivos. http://pt.kioskea.net/contents/repar/filesys.php3, Setembro 2009.

[21] P. Leach. A universally unique identifier (uuid) urn namespace. http://www.ietf.org/rfc/rfc4122.txt, July 2005.

[22] Microsoft. Suport microsoft. http://support.microsoft.com/kb/841215, Março 2009.

[23] M. Murata. Xml media types. http://www.ietf.org/rfc/rfc3023.txt, January 2001.

[24] J. Nielsen. Usability engineering. academic press inc, 1994.

[25] M. E. O'Shields and P. J. Lunsford. Webdav: A web-writing protocol and more. pages 3–7, April 2004.

[26] J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, and T. Bray. Extensible markup language (xml) 1.0 (fifth edition). http://www.w3.org/TR/REC-xml/, November 2008.

[27] R. F. R. Pinto. Integração do exchange e do outlook com .net. pages 15–18, Agosto 2005.

[28] J. Reschke. Web distributed authoring and versioning (webdav) search. http://www.ietf.org/rfc/rfc5323.txt, November 2008.

[29] J. Reschke. Webdav specifications. http://www.webdav.org/specs/, January 2009.

[30] B. G. Riso, C. E. Lenz, and V. B. Fagundes. Aspectos de autoria colaborativa com o protocolo webdav / ietf. pages 5–8, Fevereiro 2003.

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

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

[33] W3C. Http. http://www.w3.org/Protocols/, 2010.

[34] W3C. World wide web consortium. http://www.w3.org/, 2011.

[35] E. J. Whitehead. Collaborative authoring on the web: Introducing webdav. http://www.asis.org/Bulletin/Oct-98/webdav.html, October 1998.

[36] E. J. Whitehead. Webdav: Ietf standard for collaborative authoring on the web. pages 2–3, October 1998.

[37] E. J. Whitehead. World wide web distributed authoring and versioning: Webdav. pages 1–2, February 1999.

[38] J. Whitehead. Deltav: Adding versioning to the web. pages 1–10, April 2001.

[39] J. Whitehead. Web distributed authoring and versioning (webdav) ordered collections protocol. http://www.ietf.org/rfc/rfc3648.txt, December 2003.

[40] J. Whitehead. Web distributed authoring and versioning (webdav) redirect reference resources. http://www.ietf.org/rfc/rfc4437.txt, March 2006.

[41] Wikipedia. Rede semântica. http://pt.wikipedia.org/w/index.php?title=Rede_sem%C3%A2ntica&oldid=26137001, Julho 2011.

[42] Wikipedia. Webdav. http://en.wikipedia.org/w/index.php?title=WebDAV&oldid=450071523, July 2011.

[43] Wikipedia. Winfs. http://pt.wikipedia.org/w/index.php?title=WinFS&oldid=25539240, Junho 2011.

[44] J. Wilcox. Leaked windows hints at changes to come. March 2003.

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

Respostas às perguntas fechadas

Facilidade de criar documentos e adicionar ficheiros WebDav

- 5 (2x)
- 4 (2x)
- 3 (1x)
- 2 (1x)

Facilidade de encontrar o documento criado no WebDav usando o weebox

- 5 (2x)
- 4 (3x)
- 3 (1x)

Facilidade de criar documentos e adicionar ficheiros no weebox

- 5 (4x)
- 4 (1x)
- 3 (1x)

Facilidade de encontrar o documento criado no weebox usando o WebDav

- 5 (1x)
- 4 (3x)
- 3 (1x)

- 2 (1x)

Facilidade de navegação no WebDav

- 5 (1x)
- 4 (4x)
- 3 (1x)

Grau de intuição (5: muito intuitivo)

- 4 (3x)
- 3 (2x)
- 2 (1x)

Apêndice C

Respostas às perguntas abertas

Achou o sistema (WebDav/weebox confuso)? Porquê?

- Paginação (4x)
- Falta de manuseamento do weebox
- Estrutura

Quais foram as principais dificuldades encontradas?

- Paginação (4x)
- Organização dos filtros/tipo de documento
- Lentidão do sistema
- Diferenciação dos filtros com o tipo de documento
- A estrutura do weebox se traduzia no Explorer

Quais as funcionalidades que gostaria de ter encontrado?

- Pesquisa (2x)
- Performance (melhor)
- Associar imagens às pastas de forma mais intuitiva se perceber a transposição entre o weebox e WebDav

Ficou satisfeito com a experiência?

- Sim (3x)

- Sistema demasiado lento (2x)
- Mais ou menos (1x)

Indique algumas sugestões de melhoria:

- Rever o problema da autenticação na leitura/abertura de ficheiros
- Melhorar a velocidade de acesso às pastas/documentos (5x)
- Eliminar a paginação
- Colocação de ícones nas pastas com as cores iguais aos filtros do weebox
- Alterar o ícone da paginação

Em que circunstâncias é que utilizaria o WebDav? Dê exemplos?

- Backup do weebox (3x)
- Acesso rápido sem recorrer ao browser
- Construção maciça de pastas/documentos que possua muitos ficheiros, exemplo o upload de vários álbuns de fotografias
- Envio de uma estrutura complexa (pastas e ficheiros) para o weebox
- No emprego para arquivar documentos
- Sem melhorias, não usaria de todo pois está algo confuso