Universidade de Évora

Mestrado em Engenharia Informática

Gestão Documental no âmbito do projecto MyAprenderNaNet

Cláudio Manuel Frade Ramos

Orientador:

Luís Arriaga da Cunha

Co-Orientador:

Paulo Miguel Torres Duarte Quaresma

Évora, Dezembro 2012

Mestrado em Engenharia Informática

Gestão Documental no âmbito do projecto MyAprenderNaNet

Cláudio Manuel Frade Ramos

Orientador:

Luís Arriaga da Cunha

Co-Orientador:

Paulo Miguel Torres Duarte Quaresma

Resumo

Na sociedade actual é gerada uma enorme quantidade de informação, que é traduzida em documentos. Desta forma, a gestão documental tornou-se uma necessidade real e crucial para as organizações. A área da gestão documental estuda e implementa soluções para uma gestão estruturada desses documentos.

O projecto onde está inserida esta dissertação tem como objectivo o desenvolvimento de uma ferramenta de autor em formato de aplicação *web* para a área do *e-learning*. Essa aplicação requereu o desenho e implementação de um módulo de gestão documental, descrito nesta dissertação, que se focou em dois temas: a gestão dos objectos de aprendizagem carregados para a aplicação e a importação de apresentações *powerpoint* para os cursos da aplicação.

Para responder a estas necessidades foi necessário aprofundar o conceito de gestão documental, seleccionar e integrar uma plataforma de gestão de conteúdo na arquitectura da aplicação, assim como compreender e interpretar o formato das apresentações *powerpoint*.

Palavras-chave: Gestão documental, Aplicação *web*, *E-learning*, Alfresco

Content management in the context of the project MyAprenderNaNet

Abstract

In the actual society an enormous amount of information is generated, which is translated into documents. Due to this fact, the content management has become a real and crucial need for organizations. The content management field studies and implements solutions for a structured management of these documents.

The project where this dissertation is inserted has as aim the development of an authoring tool in the format of a web application for e-learning. This application required the development and implementation of a content management component, described in this dissertation. This component was focused in two themes: the management of learning objects uploaded to the application and the import of powerpoint presentations to the application courses.

To answer these needs it was necessary to develop the concept of content management, select and integrate a content management system into the application's architecture, as well as learn and interpret the powerpoint presentations format.

Keywords: Content management, Web application, E-learning, Alfresco

Agradecimentos

Na conclusão deste trabalho não posso deixar de agradecer a todas as pessoas que contribuíram para que esta dissertação chegasse a bom porto. Assim, gostaria de expressar a minha gratidão:

- Aos meus orientadores, o Prof. Luís Arriga da Cunha e o Prof. Paulo Quaresma pelos incentivos recebidos e pela disponibilidade sempre revelada.

- Aos meus colegas da Novabase que sempre me apoiaram, com particular agradecimento para o George Pereira por me ter escolhido para integrar este projecto e ao Pedro Ribeiro pela orientação no mesmo.

- Aos meus professores que me acompanharam no percurso deste mestrado por terem sabido dotar-me de boas bases para o desenvolvimento deste projecto.

- À minha namorada, Alice, que sempre me apoiou e encorajou a concluir esta etapa.

- Por fim, agradecer à minha família por todo o apoio e compreensão que me ofereceram durante esta etapa da minha vida.

Índice

Lista de Figuras

Capítulo I - Introdução

Em primeiro lugar, importa enquadrar o desenvolvimento deste estudo apresentando a área e o projecto no qual foi inserido.

A sociedade do século XXI, também conhecida como a sociedade do conhecimento, tem o conhecimento como principal ponto de valor. O conhecimento está em constante actualização e a um ritmo muito rápido, pelo que é crucial para o desenvolvimento das sociedades que a formação de base e a formação ao longo da vida permitam a actualização permanente dos requisitos das novas competências. Neste contexto, o então comissário europeu para a Economia e Emprego Gunter Verheugen, lançou uma iniciativa denominada de "*New Skills for new jobs*" que tem como principal objectivo o desenvolvimento de novas competências para as profissões do século XXI, sendo estas desenvolvidas em ambientes de grande riqueza tecnológica (1). Importa pois modernizar os processos de aprendizagem, inovando e flexibilizando a disseminação do conhecimento dentro das organizações, nas regiões e nos países. Também uma organização pública ou privada dispersa geograficamente a nível nacional ou internacional tem de garantir o contacto regular com os seus colaboradores e assegurar o estabelecimento de processos comuns de construção do conhecimento.

Nesta linha, a aprendizagem suportada em ambientes tecnologicamente ricos (TELE – *Technology Enhanced Learning Environments*) de que o *e-learning* é o mais relevante exemplo, desempenha um papel muito importante no processo de qualificação das pessoas e na modernização dos serviços.

A Novabase, conhecedora desta realidade e com vasta experiencia na área do *e-learning* avançou com o projecto MyAprenderNaNet, um projecto apoiado pelo programa COMPETE, integrado no QREN e com co-financiamento da União Europeia via FEDER desenvolvido pela Novabase com início em Outubro de 2009 e conclusão em Maio de 2011, onde se insere este estudo. Este projecto tinha como objectivo o desenvolvimento de um produto completamente inovador na área do *e-learning*, o MyaprenderNaNet, que viria posteriormente a ser lançado no mercado com o nome de TREE (*Training Resourse Express Editor*).

O TREE é uma aplicação Web 2.0. Um conjunto de ferramentas integradas disponibilizadas *online* através de um só serviço onde qualquer pessoa ou organização pode criar, editar, gerir, exportar, dinamizar e colaborar para que o *e-learning* seja uma tecnologia do seu dia-a-dia.

É uma solução complexa, composta por vários componentes e dividida em quatro grandes áreas:

- O **painel de Gestão**, um *dashboard* com informação sobre a gestão de conteúdo disponíveis ao utilizador.
- O **espaço de trabalho**, onde se cria, edita e publica os conteúdo de *e-learning* (cursos).
- A **Biblioteca**, onde se cria, edita, cataloga, pesquisa e partilha os objectos de aprendizagem (documentos, imagens, animações).
- O **Auditório**, onde se partilha os conteúdo de *e-learning* (cursos).

A grande área totalmente desenhada e desenvolvida no âmbito deste estudo foi a Biblioteca. Esta área é responsável por efectuar as operações da gestão dos objectos de aprendizagem, a catalogação, a criação, a gestão de versões e a pesquisa dos mesmos. Estes objectos de aprendizagem são documentos de texto, apresentações *powerpoint*, imagens, animações *flash* ou qualquer objecto que se queira importar para um curso de e-learning ou partilhar com a comunidade a que o utilizador pertence.

Foi também desenvolvido o importador de apresentações *powerpoint* (.pptx). No início do projecto os importadores existentes importavam imagens dos slides das apresentações *powerpoint* tornando-o visível mas não editável no seu destino. Esse não foi o objectivo, o objectivo deste componente foi importar o conteúdo dos vários slides do *powerpoint* transformando-os em objectos editáveis no editor do projecto. Assim, são retirados dos ficheiros *powerpoint* objectos representáveis no editor como as caixas de texto e imagens.

1.1 Motivação

Este estudo nasceu da necessidade de conceber, desenvolver e implementar um módulo de gestão documental para o projecto MyAprenderNaNet. Um módulo que permitisse a utilização, gestão e partilha de documentos. No decorrer do desenho deste módulo surgiu também a necessidade de importar conteúdo (imagens e texto) de ficheiros *powerpoint* (.pptx) que estariam na biblioteca, para os cursos.

A gestão documental deste projecto dividiu-se em dois grandes desafios:

- O primeiro, na criação de uma biblioteca, um LCMS (*Learning Content Management System*) que permitisse ao utilizador gerir e partilhar os seus documentos.
- O segundo no desenvolvimento de um importador de conteúdo de documentos *powerpoint* (.pptx).

1.2 Objectivos

É objectivo deste estudo desenvolver um módulo de gestão documental para o projecto MyAprenderNaNet. Este modelo inclui uma biblioteca e um importador de ficheiros de *powerpoint* (.pptx), sendo estes requisitos do projecto.

Para a biblioteca foram identificados os seguintes objectivos/funcionalidades:

- Permitir o carregamento de objectos de aprendizagem.
- Catalogar objectos de aprendizagem através de etiquetas.
- Atribuir metadados aos objectos de aprendizagem (nome, data de criação, data da ultima modificação, autor, último a modificar).
- Partilhar objectos de aprendizagem com o restante *campus*.
- Copiar, Cortar e Colar objectos de aprendizagem.
- Pesquisar objectos de aprendizagem através de metadados e etiquetas.
- Controlo de versões dos objectos de aprendizagem.
- Reutilização de objectos de aprendizagem.

Para o importador de ficheiros *powerpoint* (.pptx) o objectivo foi a importação e transformação de objectos do *powerpoint* (texto, imagens) para o formato desenvolvido e implementado na aplicação TREE.

1.3 Estrutura da Tese

A tese está dividida em cinco capítulos.

O primeiro capítulo é referente à Introdução onde se apresenta o projecto no qual este estudo foi inserido, a necessidade do desenvolvimento do mesmo, os objectivos do projecto e do trabalho efectuado no âmbito desta tese de mestrado.

No segundo capítulo é descrito o estado da arte referente à gestão documental. É descrita a sua história assim como a sua evolução até aos dias de hoje.

O terceiro capítulo é dedicado à metodologia de desenvolvimento e às ferramentas utilizadas na construção dos módulos biblioteca e importador de *powerpoint*. Primeiramente é apresentada a metodologia de desenvolvimento seguida das ferramentas tecnológicas utilizadas, linguagens de programação e plataformas utilizadas no desenvolvimento.

O capítulo seguinte é o desenvolvimento. Este capítulo é dedicado ao trabalho de desenho e implementação dos módulos responsáveis pela gestão documental do projecto. São mostradas as diferentes fases por que passaram os módulos, as suas respectivas arquitecturas e descritas as funcionalidades.

O último capítulo é inteiramente dedicado à apresentação das conclusões do estudo e às considerações sobre o trabalho futuro.

Capítulo II - Estado da Arte

Após no capítulo da introdução se ter apresentado e posicionado o projecto onde se enquadra esta dissertação assim como a problemática sob a qual este estudo visa encontrar a solução, importa enquadrar o tema da gestão documental.

Importa definir o conceito de gestão documental. No ambiente empresarial onde decorre esta dissertação, gestão documental é representada pelas siglas *ECM (Enterprise Content Management)*. Segundo a AIIM (*Association for Information and Image Management*) a definição para ECM é:

"Enterprise Content Management (ECM) is the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes. ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists." (2)

Embora esta definição esteja direccionada para a gestão documental nas empresas, também se aplica à aplicação TREE. A evolução da gestão documental está directamente ligada à evolução da tecnologia.

Apesar de haver formas de gestão documental desde os tempos da Antiga Grécia e Roma, pode-se afirmar que a invenção nos finais do século XIX do arquivador de documentos em formato de papel como o início da gestão documental estruturada. A gestão documental foi evoluindo à medida que surgiram avanços tecnológicos. (3)

O próximo passo ocorreu com a introdução de servidores, que iniciou como um conjunto forte e poderoso de processamento central, evoluindo para uma tecnologia cliente-servidor permitindo o carregamento electrónico de documentos. (3)

Quando os computadores começaram a ser utilizados de forma massiva e conectados por uma rede interna, a gestão documental das empresas começou a ser feita também nesses computadores apesar de pouco estruturada. (3)

O início de aplicações informáticas para apoiar a gestão de documentos ocorreu com o desenvolvimento dos primeiros sistemas de gestão electrónicos, que apesar de terem surgido no início dos anos oitentas, só em meados dos anos noventas começaram a ser massificados. (3)

A criação de motores de pesquisa que facilitam o acesso aos documentos geridos pelos sistemas de gestão electrónicos. (3)

Apesar de já existirem muitos documentos em formato digital na altura, o número de documentos em papel continuou a crescer. Com a introdução do *scanner,* com este a possibilidade de transformar os documentos físicos em documentos electrónicos, contribuiu para um avanço na gestão documental. (3)

A criação da nuvem e a sua evolução permitiu um importante passo para as empresas, eliminando a necessidade destas possuírem uma grande infra-estrutura para realizar a sua gestão documental, dando-lhes a possibilidade de contratarem empresas externas que possuem essa infra-estutura, e poderem focar-se somente no seu negócio. A criação de aplicações no formato de *SaaS* (software como um serviço) permitiu a empresas mais pequenas aceder a serviços de gestão documental, massificando-os. (3)

O último grande salto tecnológico deu-se com a entrada dos telemóveis de última geração, os chamados *smartphones*, que possibilitam o acesso à informação em qualquer lugar. Estes criaram a necessidade de também as aplicações responsáveis pela gestão documental das empresas se tornarem móveis, isto é, acessíveis pelos telemóveis e *tablets*. (3)

Apesar de como se refere anteriormente a evolução da gestão documental ter vindo a ser feita gradualmente, esta evoluiu a grande velocidade na última década. As soluções de gestão documental no início da década eram vistas como solução com módulos separados, um módulo de repositório de conteúdo, um para a colaboração, outro para a gestão do conteúdo, e mais alguns. Contudo, este modelo necessitava de um grande esforço de integração entre os vários componentes (4).

A gestão documental moderna não é uma aplicação, é uma plataforma de conteúdo flexível, interoperável que disponibiliza componentes e serviços facilmente acessíveis e integráveis com os processos de negócios que necessitem de suporte de conteúdo. O novo conceito de uma plataforma de gestão documental é ser transparente para os utilizadores (4). A Figura 1 mostra a evolução da gestão documental na ultima década.

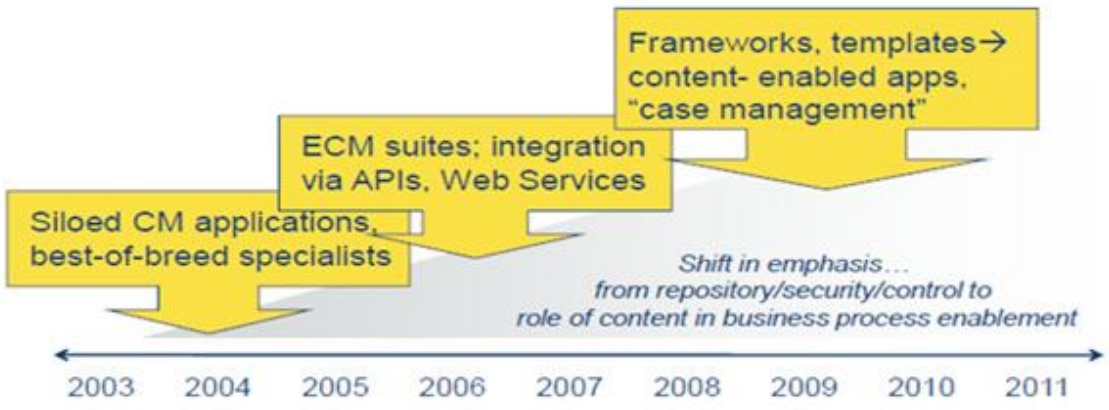

Figura 1 - Evolução da última década da gestão documental (5)

Como podemos facilmente aferir, a evolução tanto do conceito como das aplicações ocorreu a grande velocidade. O módulo de gestão documental implementado na aplicação TREE começou por ser desenvolvido no estágio das *ECM suites*, acabando por ser concluído já com a implementação das *frameworks* da última geração.

Apresentada a evolução da gestão documental, importa apresentar o ciclo normal de um documento num sistema de gestão de conteúdo. Os conteúdos inseridos num sistema desta natureza apresentam vários formatos, desde documentos de *Office*, imagens, animações, vídeos, ficheiros compactados entre outros. O ciclo de vida de um documento passa por algumas etapas, que passamos a enumerar:

1. Criação
2. Captura
3. Armazenamento
4. Criação de Versão
5. Indexação
6. Gestão
7. Publicação
8. Procura
9. Destruição / Reutilização

No primeiro passo é criada uma representação do objecto quando se está a carregar. Na fase da captura é feita a fusão do objecto com a representação criada. Na próxima fase é armazenado no disco seguindo-se a criação da versão correcta do mesmo. É então indexado no motor de pesquisa na fase da indexação. Na gestão são inseridos os metadados do objecto e realizadas as operações definidas para cada tipo de objecto.

Segue-se a publicação na interface do gestor de conteúdo, se for caso disso na interface da aplicação que está interligada com o gestor de conteúdo. Este objecto fica assim disponível para a fase da procura, ou seja, disponível para ser utilizado na aplicação. A última fase do ciclo é a reutilização do objecto, se necessário, ou a sua destruição, se não for mais necessário.

Capítulo III - Metodologia e Ferramentas Utilizadas

3.1 Metodologia de Desenvolvimento

A metodologia de desenvolvimento utilizada foi a SCRUM. "Scrum é um processo iterativo e incremental utilizado para o desenvolvimento de produtos ou gestão de tarefas" (6).

Na Figura 1 é esquematizado o ciclo de desenvolvimento utilizado.

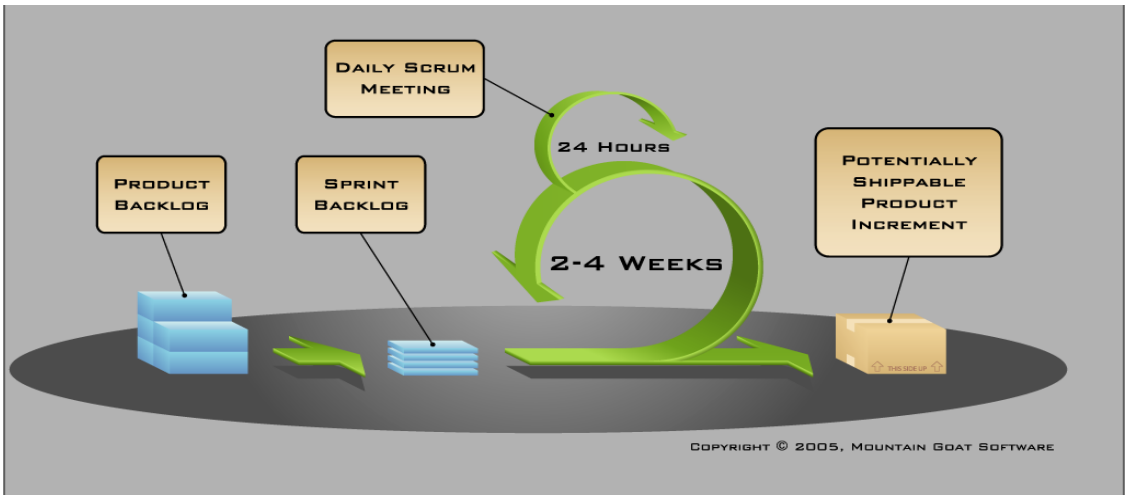

Figura 2 - Funcionamento da metodologia SCRUM (7).

No início do projecto foi criado o *backlog* do produto, a lista de requisitos e funcionalidades que estavam previstas serem desenvolvidas para o TREE, ordenadas por prioridade. Em seguida eram planeadas pequenas tarefas, designadas por *sprint*, também elas ordenadas por prioridade para um espaço de tempo entre 2 a 4 semanas, sendo que no projecto a maioria desses sprints foram de 3 semanas. No início de cada dia a equipa reunia para fazer o ponto de situação. Como a equipa estava dividida entre Lisboa e Évora essa reunião era feita normalmente através do *Microsoft Office Communicator* e tinha normalmente uma duração de 10 minutos. Quando terminavam os *sprints*, era feito um novo ponto de situação e elaborada nova lista de tarefas para cada colaborador, dando início a um novo *sprint*. Existia um quadro onde cada colaborador tinha as suas tarefas do *sprint* divididas em três estados: "Em espera", "Em Desenvolvimento" e "Completa". Este quadro permitia a cada elemento, assim como à

equipa, saber exactamente como estava a correr o *sprint* e adaptar o esforço necessário para este ser concluído com sucesso.

Este modelo foi particularmente importante no desenvolvimento do gestor de conteúdo, a Biblioteca, onde permitiu uma definição clara e objectiva de que funcionalidades eram prioritárias e quais os maiores obstáculos a ultrapassar.

3.2 Ferramentas Utilizadas

Numa fase inicial do projecto e numa primeira abordagem tecnológica foi proposto basear o projecto em tecnologia Adobe, utilizar a ferramenta Adobe Livecycle como servidor de *backoffice*, Adobe Coldfusion para as bases de dados e Adobe Flex para o *frontoffice*.

Na fase de análise funcional e desenho da arquitectura optou-se por utilizar uma diferente base tecnológica. Utilizou-se então:

- Como servidor Web a ferramenta Apache HTTP Server (8).
- Para as bases de dados, MySql (9) e PHP (10).
- Como CMS (Content Management System) a ferramenta Alfresco Content Management System (11).
- Para a interface Adobe FLEX (12).
- Para modelação de dados a linguagem XML (13).
- Para o importador de *powerpoint* a linguagem Java EE (14) e o servidor Apache Tomcat (15).
- Para o desenvolvimento da biblioteca foi utilizada a ferramenta de desenvolvimento de software Adobe Flash Builder e para o importador de *powerpoint* a ferramenta Netbeans.

A decisão de alterar a base tecnológica deveu-se essencialmente à constatação que a solução baseada inteiramente em tecnologia Adobe teria um custo de instalação bastante elevado, sem acréscimo de funcionalidades. Optou-se assim por uma solução baseada em ferramentas *open-source* e em servidores com sistema operativo *Linux*.

3.2.1 Biblioteca

No desenvolvimento da biblioteca foram utilizadas as seguintes ferramentas tecnológicas:

1. O CMS (Content Management System) Alfresco (11).
2. As frameworks de desenvolvimento *Web script*s e Adobe Flex (12).
3. As linguagens de programação Actionscript 3.0 (16), JAVA EE (14), MXML e XML (13).
4. A ferramenta de desenvolvimento Adobe Flex Builder.

3.2.1.1 – Alfresco Content Management System

O Alfresco Content Management System é uma plataforma *open-source* desenvolvida pela empresa Alfresco. A empresa Alfresco foi fundada em 2005 por John Newton, co-fundador da Documentum, e por John Powell, director de operações da Business Objects.

EM 2005 a empresa lançou-se no mercado no ECM (Enterprise Content Management) com uma nova abordagem. Um novo modelo de desenvolvimento e distribuição *open-source* baseado em padrões abertos e incorporando componentes *open-source* como o motor de pesquisa Lucene (17), a plataforma de desenvolvimento Java Spring (18), a plataforma de desenvolvimento Java Hibernate (19), o motor de *workflows* jBPM (20), a plataforma FREEMARKER (21) e a interface para documentos Office APACHE POI (22). O núcleo da plataforma é o seu repositório de conteúdo JSR-170 para tecnologia JAVA.

É uma solução multiplataforma facilmente escalável, eficiente, simples, modular e facilmente parametrizável que permite gerir os documentos de uma aplicação ou organização. A criação, organização, recuperação, o controlo de versões, a pesquisa e o armazenamento de documentos estão entre as suas funcionalidades.

A Figura 3 representa uma visão geral sobre a plataforma Alfresco.

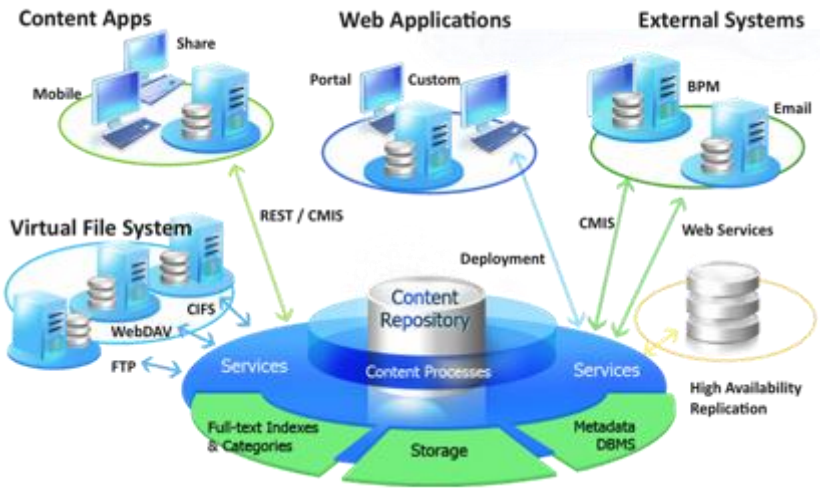

Figura 3 - Plataforma Alfresco (23)

A plataforma Alfresco está disponível em duas versões, a versão Alfresco Community Labs e a versão Alfresco Enterprise Edition. Para o desenvolvimento deste estudo foi utilizada a primeira versão, uma versão mais vocacionada para o desenvolvimento. A comunidade *open-source* do projecto Alfresco é bastante organizada, com muitos colaboradores e com um fórum de questões muito activo.

Neste estudo foi utilizado o repositório de conteúdo, os seus serviços web e serviços REST desenvolvidos na forma de *web script*s. O desenvolvimento é feito em cima do repositório de conteúdo utilizando *web script*s para construir serviços REST customizados e adaptados às necessidades do projecto.

Para além de ser a base do modelo de gestão documental, é também responsável pela gestão de utilizadores da aplicação TREE, através dos seus serviços web de autenticação e gestão de sessões.

3.2.1.2 – Web scripts

Um *web script* é simplesmente um serviço disponibilizado através de *URI* que responde a métodos HTTP. Os métodos mais utilizados são o *GET*, *POST*, *PUT* e *DELETE*. Existem dois tipos de *web script*s, os *Data Web script*s e os *Presentation Web script*s. Os do primeiro tipo são responsáveis pela interacção com o repositório de conteúdo e disponibilizam uma interface para que aplicações cliente possam efectuar acções como retirar, pesquisar, inserir ou actualizar documentos, utilizam formatos como o XML e o JSON. Os do segundo tipo são utilizados no lado das aplicações do cliente e interagem com os *Data Web script*s para acederem à informação. O formato mais utilizado por estes é o HTML.

No desenvolvimento deste estudo foram desenvolvidos somente *Data Web script*s para aceder aos conteúdo e o formato eleito para esse efeito foi o *XML*.

*Alfresco Web script*s foram introduzidos em 2006 para melhorar a interacção entre a Web e o repositório de conteúdo. Estes utilizam uma lógica *MVC (Model View Controller)* onde o modelo é o repositório Alfresco, o controlador é definido por Javascript e a vista devolvida pela *framework* Freemarker (24).

A Figura 4 esquematiza a arquitectura de um *Data Web script*.

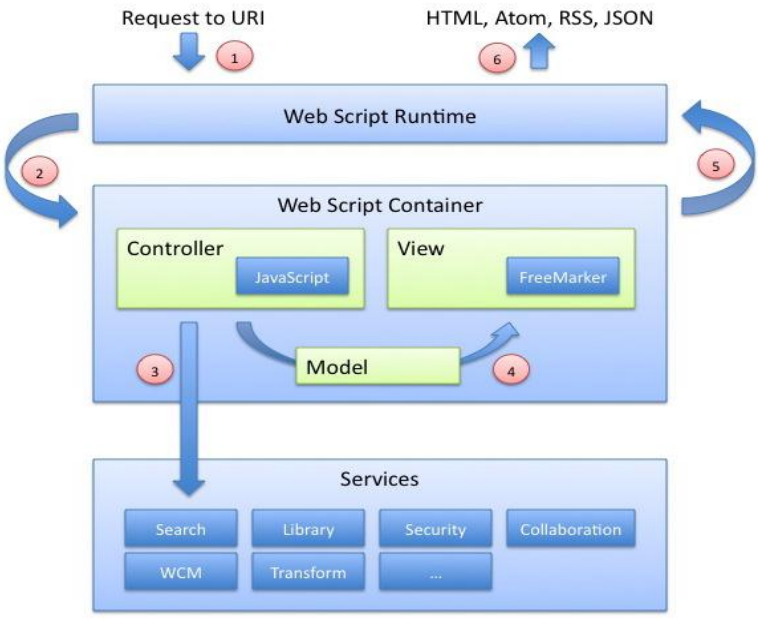

Figura 4 – Representação da arquitectura de um data *web script* (24)

3.2.1.3 - Adobe/Apache Flex

Em 2004 a Macromedia criou a plataforma Flex, na altura chamada de Macromedia Flex lançando com este nome as versões 1.0 e 1.5 do Flex. Em 2005 a Adobe adquiriu a Macromedia e lançou a versão 2.0 do Flex, agora renomeado para Adobe Flex. Desde então já foram lançadas as versões 3, 4 e a mais recente, a versão 4.5. Em Dezembro de 2011 a Adobe doou a plataforma Adobe Flex para a Apache Software Foundation sendo esta renomeada para Apache Flex (25). Actualmente está em modo de incubação na Apache tornando-se um projecto da comunidade *open-source*.

Flex é uma plataforma *open-source* desenhada para criar RIAs (Rich Internet Aplications). Permite criar aplicações móveis, web ou desktop utilizando a plataforma flash, ou seja, o flash player, para ser interpretado. Utiliza uma linguagem de marcação, o MXML, para definir a interface e uma linguagem de programação, o Actionscript 3, para a lógica da aplicação.

Na compilação, o MXML é traduzido em classes de Actionscript 3 e é gerado um ficheiro SWF para ser colocado no servidor web de onde o cliente da aplicação poderá fazer *download* para o seu browser e navegar na aplicação.

A Figura 2 esquematiza o processo de construção e disponibilização ao cliente de uma aplicação FLEX.

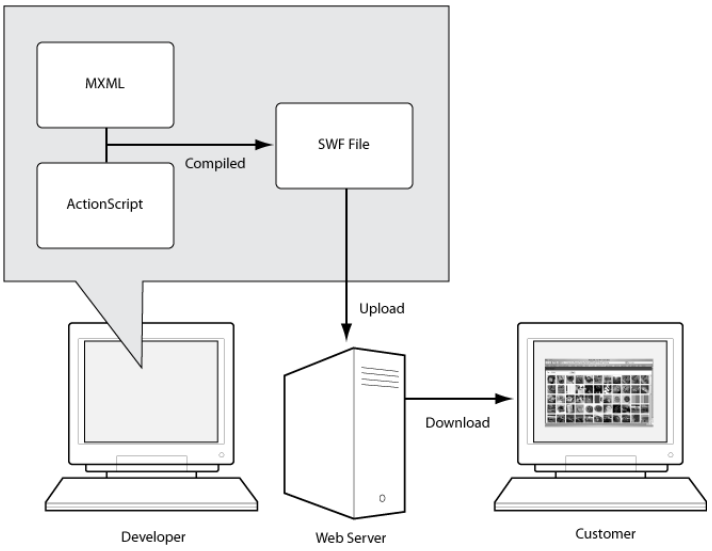

Figura 5 - Esquema da framework Adobe Flex (26).

3.2.1.4 - ActionScript 3.0

É uma linguagem de programação orientada a objectos que é executada pelo Flash Player. Esta nova versão de actionscript tira partido da nova AVM (Actionscript Virtual Machine) disponível a partir do Flash Player 9, a AVM2. Esta consegue uma executar o Actionscript até dez vezes mais rápido que a antiga AVM1 (27) .

Esta linguagem é baseada na *ECMAScript*, a linguagem *standard* na programação de scripts. Permite um maior grau de complexidade aos programadores que o seu antecessor, o Actionscript 2.0 (27).

Esta linguagem possui bibliotecas que lhe permitem manusear a linguagem XML e executar chamadas a serviços web, requisitos essenciais para ser utilizada para desenvolver a componente lógica da aplicação.

3.2.1.5 - MXML

É uma linguagem de marcação baseada em XML utilizada principalmente para definir a interface do utilizador. É possível definir com esta linguagem objectos não visuais como por exemplo acessos a serviços web e bases de dados, embora seja mais usual estas estarem definidos em Actionscript 3 (28).

O marcador <*s:Application*> marca o ficheiro principal da aplicação. A possibilidade de desenvolver componentes para posteriormente integrar e que sejam completamente independentes facilita a tarefa de desenvolvimento da mesma. Os componentes MXML são ficheiros MXML de componentes já existentes ou customizados como uma tabela de dados, uma caixa de texto ou uma árvore feita de raiz e são integrados colocando o respectivo marcador no MXML. Estes permitem que uma equipa grande de desenvolvimento trabalhe na mesma aplicação ao mesmo tempo, desenvolvendo os mais diferentes componentes da aplicação independentemente.

Esta foi a linguagem utilizada para a criação da interface na aplicação.

3.2.1.6 - Adobe Flex Builder

Adobe Flash Builder é uma ferramenta de desenvolvimento de software baseada em Eclipse (29). Esta ferramenta permite construir aplicações web, móveis e desktop utilizando a linguagem de programação Actionscript 3.0 e a plataforma Flex (30).

Esta ferramenta é utilizada no desenvolvimento de RIAs (Rich Internet Aplications) permitindo a integração com servidores PHP e facilitando a utilização de serviços web.

3.2.1.7 - XML

A linguagem XML, *eXtensible Markup Language*, criada por um grupo de 11 pessoas sobre a alçada do *Worl Wide Web Consortium* (W3C), é uma linguagem de marcação desenhada para transportar informação de uma forma estruturada.

Esta linguagem surgiu da necessidade de enviar documentos estruturados pela internet, sendo que as duas formas de o conseguir, o HTML e o SGML não eram indicadas para este objectivo. O HTML devido à sua semântica e estrutura rígida não permite que se utilize uma estrutura arbitrária e o SGML apesar de permitir criar estruturas arbitrárias é demasiado complexo para se enviar documentos estruturados através de um browser (31).

O XML é actualmente um dos formatos standards mais utilizado para partilhar informação estruturada pela Web. Este é a base de tecnologias tão variadas como a tecnologia de processamento de texto ODT e OOXML, formatos gráficos como o SGF e serviços web como o XMLRPC e SOAP (32).

É utilizado na biblioteca como parte dos serviços web e da forma como o Alfresco manipula os documentos. É utilizado implicitamente no serviço web do importador de *powerpoint* através da comunicação SOAP e do Office OPEN XML. Este formato foi também o formato escolhido para representar os ecrãs dos cursos desenvolvidos na plataforma TREE.

3.2.2 Importador de ficheiros *Powerpoint* (.pptx)

Para o importador de ficheiros *Powerpoint* foram utilizadas as seguintes ferramentas tecnológicas:

1. A linguagem de programação JAVA, nomeadamente JAVA Web Services.
2. A linguagem Office OPENXML.
3. A ferramenta de desenvolvimento Netbeans.

3.2.2.1 – JAVA JAX-RPC

Java JAX-RPC (Remote Procedure Call), Java API's (*Aplication Programming interface*) baseadas em XML que facilitam a interoperabilidade e acessibilidade aos serviços web. Estas são utilizadas no desenvolvimento de aplicações que necessitam de aceder a serviços web.

Estas API's (*Aplication Programming interface*) possibilitam que duas aplicações desenvolvidas com tecnologias diferentes possam comunicar entre si. Utiliza SOAP, um protocolo baseado em XML utilizado para trocar mensagens em ambientes distribuídos e o protocolo HTTP para executar chamadas remotas de aplicações para o serviço web. (33)

Utiliza a linguagem WSDL (Web Service Definition Language) para descrever o *endpoint* e quais as operações que o serviço web disponibiliza. Através desse WSDL é possível gerar uma chamada ao serviço web, tendo a informação de quais os tipos de dados envolvidos na chamada e qual o tipo de dados da resposta. (33)

No caso do serviço desenvolvido no âmbito deste projecto, a chamada é feita utilizando o tipo de dados String e o serviço responde em XML.

3.2.2.2 - OFFICE OPENXML (OOXML)

É um formato padrão para a criação de documentos Office como documentos de texto, apresentações de *Powerpoint* ou folhas de cálculo, ratificado pela *ECMA International*. Este formato nasceu da necessidade de preservação dos ficheiros durante um período longo de tempo (34).

O formato binário anteriormente utilizado tornou-se obsoleto e desfasado no tempo, pois foi criado numa era onde o espaço e a velocidade de processamento eram muito limitados. Actualmente, com o aumento do espaço e processamento disponível, a velocidade de acesso aos ficheiros deixou de ser um problema podendo agora ser criados documentos que são eles próprios um conjunto de outros documentos interligados. Os documentos compilados em formato binário possuem fortes referências aos programas que os criam dificultando a leitura dos mesmos sem uma perda significativa de informação no futuro (35).

É um formato baseado em XML que tem como objectivo mapear em XML todas as funcionalidades existentes no formato anterior e estende-las a novas funcionalidades, fornecendo melhor documentação aos documentos e permitindo a sua interoperabilidade (35).

Este formato especifica também três linguagens de marcação diferentes, uma para cada formato de documentos. Para os documentos de processamento de texto é utilizada a WordprocessingML, para as folhas de cálculo a linguagem SpradsheetML e a PresentationML para as apresentações de *Powerpoint* (35).

Os ficheiros no formato OOXML são na verdade um pacote denominado *package*, que contém vários ficheiros denominados de *parts* sendo o formato do pacote definido por OPC (*Open Package Conventions*) (35).

Neste projecto foi explorada a linguagem PresentationML uma vez que o objectivo foi importar apresentações *Powerpoint* para a aplicação.

3.2.2.3 – NETBEANS

É uma ferramenta de software livre que começou como um projecto de estudantes em 1996 numa Universidade da então Checoslováquia, actual República Checa e que tinha o nome inicial de Xelfi. Numa altura que o JAVA dava os seus passos iniciais o objectivo foi criar um ambiente integrado que facilitasse o desenvolvimento nesta linguagem. No verão de 1999 a Sun Microsystems, a empresa responsável pela linguagem JAVA necessitava de melhores ferramentas para desenvolvimento de aplicações JAVA e interessou-se pelo projecto tornando-o o ambiente de desenvolvimento integrado preferido para o desenvolvimento. Em 2010 quando a ORACLE adquiriu a SUN, este tornou-se parte integrante da ORACLE que continua a patrocina-lo (36).

Actualmente é um ambiente integrado para o desenvolvimento de aplicações em JAVA, PHP e JAVASCRIPT entre outras.

Neste projecto foi utilizado para desenvolver o serviço web em JAVA, que importa os ficheiros *powerpoint*, aproveitando o plugin de integração de serviços JAX-RPC. Este plugin permite a criação automática do ficheiro WSDL, retirando esse esforço do desenvolvedor.

Capítulo IV – Desenvolvimento

4.1 Biblioteca

A biblioteca é o módulo da aplicação responsável pela gestão de documentos. Este módulo funciona como um LCMS (Learning Content Management System) e controla todos os documentos do utilizador.

Este componente é o núcleo da gestão documental sendo este responsável pela gestão, armazenamento e partilha dos objectos de aprendizagem. Os objectos de aprendizagem são imagens, animações *flash*, documentos de texto, apresentações *powerpoint*, qualquer objecto que o utilizador queira incluir num curso ou partilhar com a sua comunidade.

4.1.1 Arquitectura

A arquitectura da biblioteca passou por várias fazes até chegar à sua versão final. As alterações deveram-se a actualizações ao gestor de conteúdo Alfresco e da plataforma Flex, assim como decisões de natureza empresarial no que diz respeito à utilização de ferramentas de software livre ou proprietárias.

A tecnologia em que se iria desenvolver a interface já estava definida por outros componentes da aplicação, a plataforma Flex, sendo esta a única restrição no começo do desenho da arquitectura deste componente. No projecto inicial estava definida a plataforma Adobe Livecycle, utilizando-se o gestor de conteúdo que vem incluído na plataforma Livecycle, como principal motor do modelo de gestão documental do projecto. Numa decisão do foro empresarial, após verificar-se que seria possível seguir o caminho do software livre em detrimento do proprietário inicialmente previsto no projecto, foi definida uma nova abordagem ao projecto, uma abordagem de software livre.

A solução de gestor de conteúdo do Adobe Livecycle é baseada em Alfresco, um Alfresco customizado embebido na plataforma da Adobe (37). Após uma análise das características da plataforma Alfresco, da sua API Remota e das suas funcionalidades verificou-se que a mesma possuía todas as funcionalidades requeridas para o desenvolvimento da biblioteca, nomeadamente, a gestão de versões e serviços web para criar, catalogar, pesquisar e alterar as propriedades dos documentos. Foi então que apresentámos a primeira abordagem à arquitectura da mesma. Esta primeira abordagem tinha em conta a versão mais recente do Alfresco naquele momento, a versão 2.1. Nesta versão da plataforma Alfresco existiam várias formas de interagir com o seu repositório. Partiu-se desta forma para a análise de cada uma delas:

- Embeber na aplicação o repositório da plataforma Alfresco (38), hipótese recusada à partida devido à tecnologia que utilizava a aplicação e ao objectivo de construir uma aplicação modular, em que cada componente fosse desenvolvido de forma autónoma e posteriormente incorporado no projecto.
- Utilizar a API JCR (*Java Content Repository*) existente na plataforma Alfresco (38). Esta hipótese também foi recusada, pois a tecnologia de desenvolvimento estava previamente definida e esta API existia somente para tecnologia JAVA.
- API de serviços web baseados em SOAP (38), disponibilizada desde a versão 1.0 da plataforma Alfresco (39). Esta hipótese foi já considerada por esta solução ser compatível com a tecnologia da aplicação e cobre a maioria dos serviços do repositório de conteúdo da plataforma Alfresco.
- Endereçamento por URL, sendo que todos os objectos existentes no repositório de conteúdo da plataforma Alfresco possuem a propriedade de serem endereçáveis por URL. Este endereçamento recebeu uma melhoria nesta versão com a introdução da plataforma *Web script*s (38).

Existiam portanto dois caminhos possíveis, desenvolver através da API de serviços web baseados em SOAP ou por *web script*s através da API RESTful. A primeira tinha em seu favor a segurança da norma SOAP, a facilidade de acesso através da plataforma FLEX e o facto de existir desde a versão 1.0, fazendo com que esta API estivesse mais robusta e estável (39). A segunda estava a ser introduzida nesta versão e trazia uma nova forma de desenvolver utilizando o modelo MVC (*model view controller*), porém

esta versão ainda não cobria todos os serviços do repositório e estava a ser sujeita à primeira introdução no mercado. Então para a primeira versão da biblioteca, sendo esta mais uma prova de conceito do que o início do desenvolvimento optou-se por utilizar os serviços web SOAP, particularmente por serem uma tecnologia standard, apresentarem uma API que disponibiliza a maior parte dos serviços do repositório e pelo tempo que já se encontram na plataforma Alfresco (39). Assim sendo, a Figura 6 mostra a primeira versão da arquitectura.

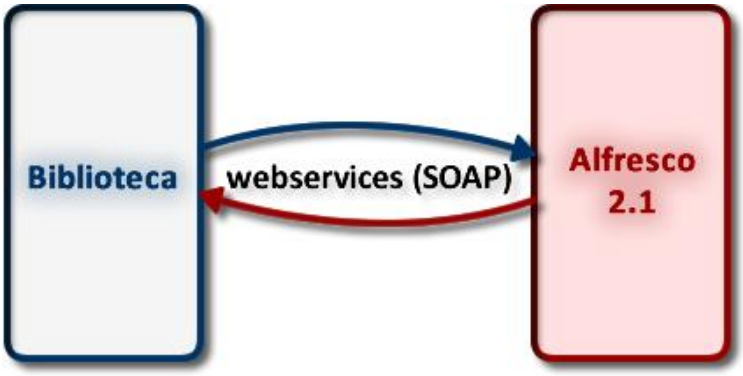

Figura 6 - Primeira versão da arquitectura da Biblioteca

Nesta fase foi criada uma prova de conceito entre a biblioteca, desenvolvida na plataforma FLEX e a API de serviços web baseados em SOAP do gestor de conteúdo Alfresco 2.1. Esta prova de conceito decorreu com alguns problemas, nomeadamente a complexidade da mensagem SOAP, acabando por demorar um pouco mais de tempo que o previsto, mas acabando por chegar a bom porto. Quando se finalizou o desenvolvimento e testes desta primeira prova foi lançado o Alfresco 3.0. Esta nova versão do Alfresco trazia novidades ao nível da sua API RESTful e dos seus *web script*s. Era agora uma certeza que a plataforma Alfresco iria prosseguir nesse caminho e apesar de manter a sua API serviços web baseados em SOAP encorajava a que se desenvolvesse utilizando *web script*s (40).

Decidiu-se então fazer a actualização da versão 2.1 do Alfresco para a versão 3.0. Juntamente com esta decisão foi definido que se iria apostar nesta nova API RESTful, não fazendo sentido apostar nos serviços web baseados em SOAP se a plataforma Alfresco não o iria fazer. Aproveitando no entanto o trabalho desenvolvido, uma vez

que a autenticação poderia perfeitamente ser feita através do protocolo SOAP com inteira segurança. Apresentou-se então uma nova versão da arquitectura descrita pela Figura 7.

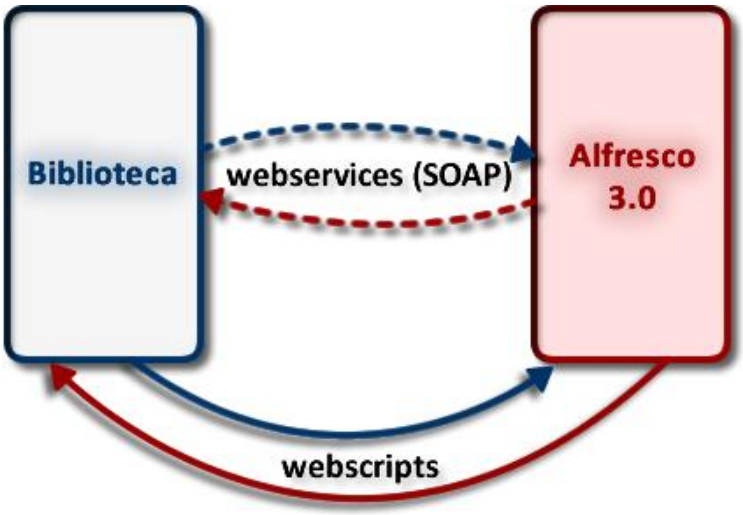

Figura 7 - Segunda versão da arquitectura da Biblioteca

Nesta fase desenvolveu-se os primeiros *web script*s, nomeadamente, o serviço que permitia ao utilizador visualizar os seus documentos e pastas. Importa nesta altura referir que o gestor de conteúdo Alfresco utiliza o conceito de espaços (*spaces*) e de nós (*nodes*) para referenciar pastas e documentos. Os *web script*s desenvolvidos inicialmente utilizam o método GET e obtêm a informação de forma a podermos visualizar as primeiras pastas e documentos na biblioteca. Estes foram necessários para construir as duas árvores da interface, a árvore da biblioteca privada que permite ao utilizador visualizar de uma forma simples a sua área privada, e a árvore da biblioteca pública que permite o acesso a material partilhado por este e por outros utilizadores do seu campus. A imagem seguinte mostra o primeiro aspecto da interface, ainda sem botões que permitissem qualquer interacção com os documentos e pastas.

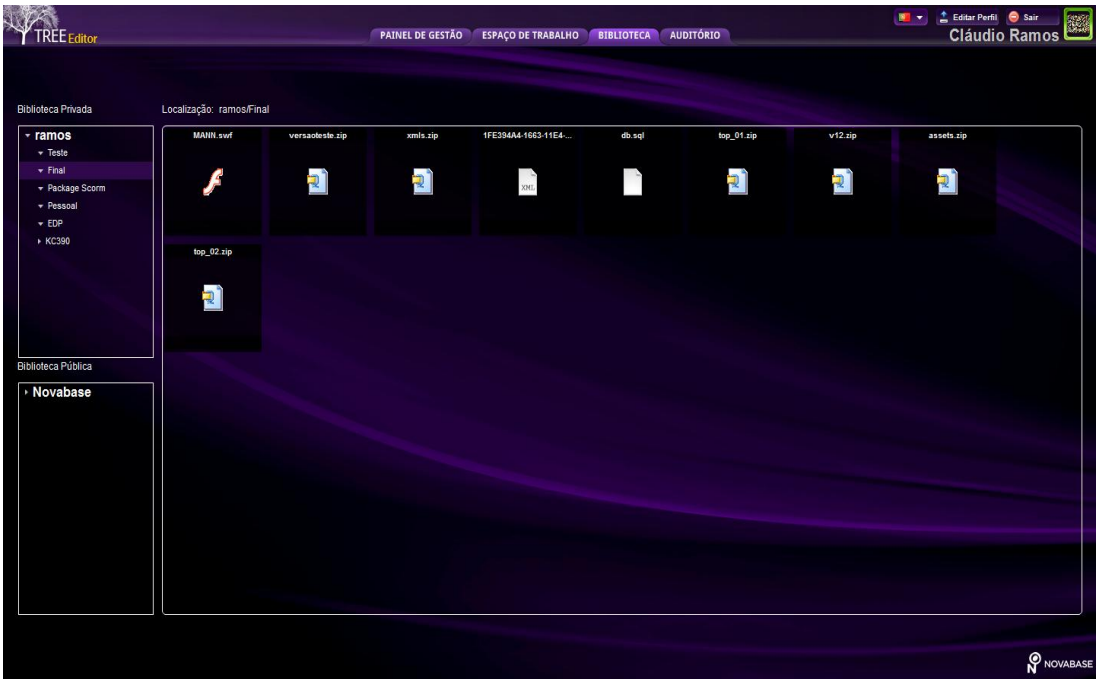

Figura 8- Primeira versão do interface da Biblioteca

Após ter desenvolvido estes *web script*s e de terem sido integrados com sucesso na interface iniciou-se a construção dos próximos para interagir com as pastas e documentos. Os seguintes *web script*s desenvolvidos utilizam o método POST e efectuam o carregamento e consequente criação de documentos. Seguiu-se a criação de pastas e as acções cortar, copiar e colar, essenciais à estruturação e manuseamento de pastas e objectos de aprendizagem. Para proporcionar ao utilizador as operações básicas ao total controlo sobre os seus objectos de aprendizagem faltava desenvolver o *web script* para permitir a operação de apagar as pastas e documentos. Este *web script* foi desenvolvido utilizando o método DELETE com o objectivo de apagar objectos de aprendizagem e pastas. A Figura 9 mostra o aspecto da interface da Biblioteca, já com estas funcionalidades incorporadas.

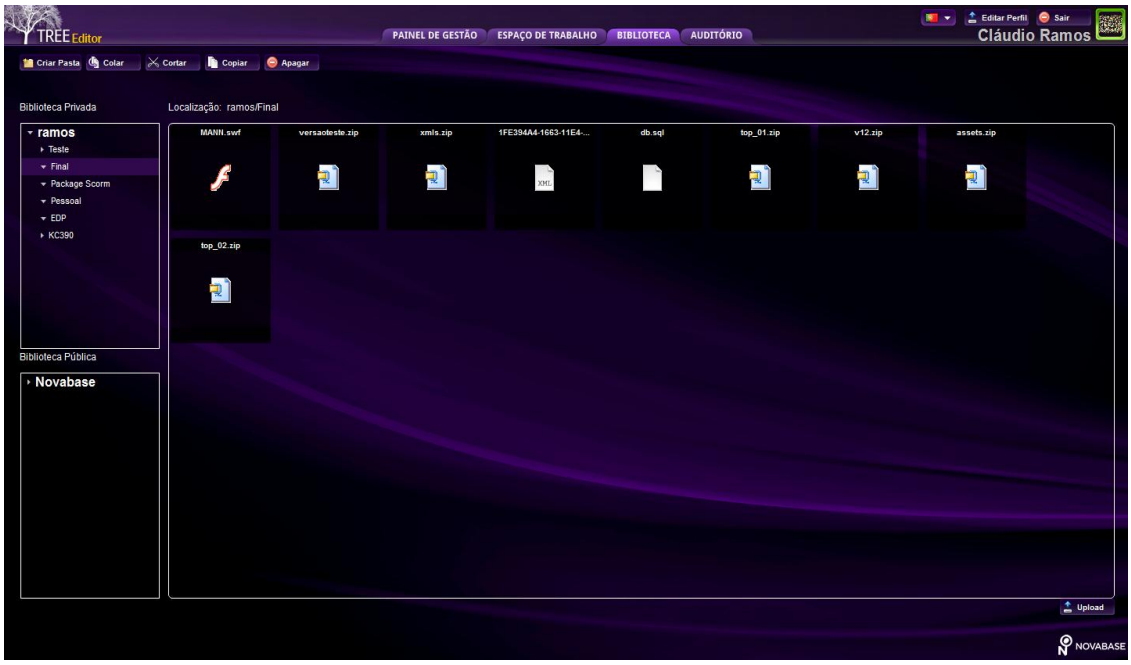

Figura 9 - Segunda versão da interface da Biblioteca

Neste momento estavam desenvolvidas as operações básicas para interagir com a biblioteca. O próximo passo seria dotar a aplicação de duas funcionalidades que estavam definidas nos requisitos do projecto, a pesquisa e a catalogação dos documentos.

Para a catalogação de documentos são adicionadas etiquetas (*tags*) assim como um conjunto de propriedades que juntamente com as etiquetas formam os metadados do documento. As propriedades adicionadas são: o autor do documento, o nome do documento, um título e uma descrição do documento. Existem ainda quatro propriedades automáticas, ou seja, geridas pelo sistema, sendo estas a data de criação, a data da última modificação, o nome de utilizador da pessoa que criou o documento e de quem o modificou por último.

No caso da pesquisa, foi criado o *web script* que executa as consultas ao motor de pesquisa incorporado no Alfresco, o Lucene, utilizando para o efeito o método GET. Definiu-se então uma pesquisa dita normal e uma pesquisa avançada. A pesquisa normal consiste numa pesquisa efectuada através do nome dos documentos e uma pesquisa avançada que possibilita a pesquisa por metadados do documento, isto é, por autor, título, data de criação ou modificação, formato, descrição ou etiquetas. Definidos

que estavam os parâmetros sobre os quais a pesquisa iria incidir, começou-se a criar as consultas Lucene para o mesmo efeito.

Após a conclusão com sucesso destas duas novas funcionalidades e antes de prosseguir no sentido de desenvolver outra operação requerida também pelo projecto, a criação de versões e o seu acesso deparou-se com um obstáculo. Foi debatido com as chefias e decidido pelas mesmas que o módulo de autenticação para a aplicação seria desenvolvido de forma a utilizar a gestão de utilizadores da plataforma Alfresco. Esta decisão levou à última alteração na arquitectura da biblioteca, isto é, definiu a versão actual da arquitectura da mesma. Em termos práticos, a biblioteca ficou ligada directamente por via de *web script*s à plataforma Alfresco e por via indirecta através de SOAP pois foi desenvolvido um módulo de gestão de utilizadores onde se centralizou a autenticação do utilizador. Aproveitando esta fase do projecto onde se implementou um módulo de gestão de utilizadores, decidiu-se fazer novamente a actualização da plataforma Alfresco da versão 3.0 para a versão 3.4.

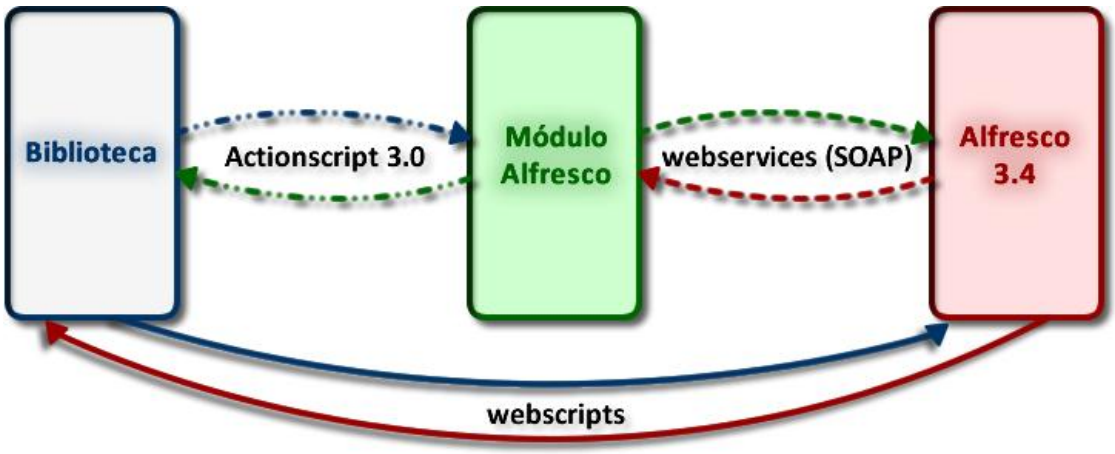

Figura 10 - Versão final da arquitectura da Biblioteca

Após a alteração final da arquitectura da biblioteca procedeu-se ao desenvolvimento da última funcionalidade requerida no projecto inicial, a gestão das versões de um objecto de aprendizagem na biblioteca. Esta funcionalidade foi um pouco mais complexa de implementar pois para além da construção do *web script* para fazer o carregamento do objecto, a mecânica de actualizar ou criar um novo objecto e a criação de um *web script*

que adquirisse a informação sobre as versões do objecto, foi necessário alterar as configurações iniciais da plataforma Alfresco que não vêm definidas por defeito para criar versões de documentos. Esta alteração levou a uma investigação profunda dos ficheiros de configuração da aplicação, não sendo possível faze-la através da sua interface, apenas através da alteração do seu ficheiro de propriedades no formato XML. Após a conclusão mais uma vez com sucesso desta funcionalidade ficamos com a interface principal completa. A Figura 11 mostra o aspecto da biblioteca na sua versão final e actual.

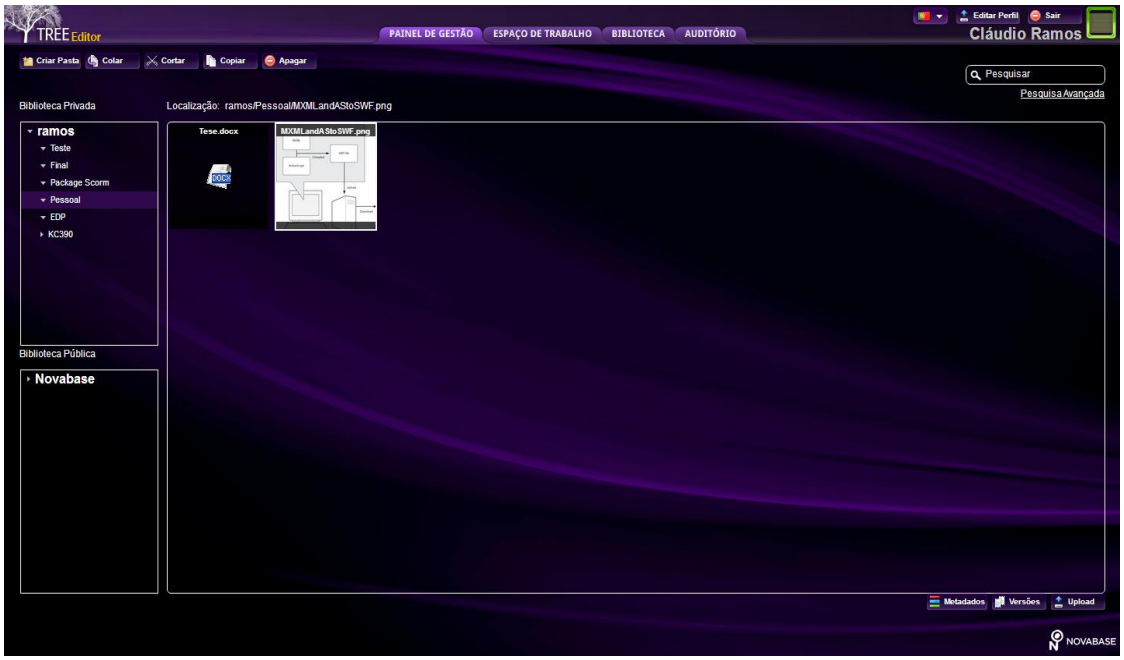

Figura 11 - Versão final da interface da Biblioteca

4.1.2 As funcionalidades

Neste ponto ir-se-á aprofundar as funcionalidades desenhadas e desenvolvidas no âmbito deste projecto no módulo biblioteca, tendo estas sido apresentadas muito superficialmente no ponto anterior, dedicado à arquitectura como forma de enquadrar o desenvolvimento da mesma com a evolução nas funcionalidades da aplicação. As

funcionalidades podem ser agrupadas devido à natureza das suas operações e desenvolvimento. Assim existem 5 grupos de funcionalidade enumerados em seguida:

1. Criar pasta, carregar e actualizar objectos
2. Copiar, Cortar, Colar e Apagar
3. Catalogar objectos
4. Pesquisar objectos
5. Versões de objectos

Enumerados que estão os grupos de funcionalidades, segue uma visão mais aprofundada de cada um.

1. Criar pasta, carregar e actualizar documento

Este grupo de operações tem em comum o método utilizado pelo *web script* de cada operação, o método POST, assim como o objectivo de serem aplicadas a um só tipo de objecto, os nós no caso do carregamento e actualização dos objectos de aprendizagem e os espaços no caso da criação de pastas.

No caso da funcionalidade de criar pastas, é uma operação que é executada através de um *web script*, cria um espaço na plataforma Alfresco e o conecta com o espaço pai sob o qual está a ser criado. Na funcionalidade seguinte trata-se da criação de ficheiros e actualização dos mesmos. Para tal efeito foram desenvolvidos dois *web script*s, um para a criação de um novo objecto na biblioteca, que é o *web script* utilizado quando se carregam ficheiros para a biblioteca, o *web script* para actualizar um objecto existente. Sentiu-se esta necessidade devido à complexidade envolvida na criação de um objecto novo ser mais reduzida em relação a actualizar um existente. No carregamento e subsequentemente na criação do objecto utilizou-se um *web script* que cria o nó novo no respectivo espaço com as suas propriedades. Quando se carrega um objecto que existe, a aplicação questiona o utilizador se pretende fazer uma actualização ao ficheiro existente ou criar um novo alterando o nome do ficheiro. Quando o utilizador opta por actualizar o documento, este processo passa pela criação de uma nova versão do mesmo e a sua disponibilização. Uma vez que estas funcionalidades não têm expressão gráfica ao nível da interface sem ser os botões que as efectuam e que estão presentes na figura que representa a versão final da interface, as mesmas não serão ilustradas.

2. *Copiar, Cortar, Colar e Apagar*

Estas funcionalidades estão agrupadas por serem executadas tanto para os espaços como para os nós, isto é, a interface disponibiliza estes quatro botões que dependendo do objecto que está seleccionado assim executam as operações sobre os mesmos, independentemente se são pastas ou objectos de aprendizagem. Neste caso nem todas utilizam o método POST, sendo que apesar das operações copiar, cortar e colar o utilizarem, a funcionalidade apagar utiliza o método DELETE.

3. *Catalogar objectos*

Esta funcionalidade foi desenvolvida com o recurso à introdução de metadados nos objectos de aprendizagem. Estes metadados são compostos por dois conceitos existentes na plataforma Alfresco, as propriedades dos objectos e as etiquetas associadas aos objectos. Foram criados dois *web script*s para cada um dos conceitos. Para as propriedades foi desenvolvido o *web script* para retornar as propriedades de um objecto utilizando para o efeito o método GET e outro webscript para alterar as mesmas propriedades, este utilizando o método POST. Para executar estas tarefas foi desenvolvida uma interface própria, a Figura 12 mostra essa mesma interface no separador das propriedades.

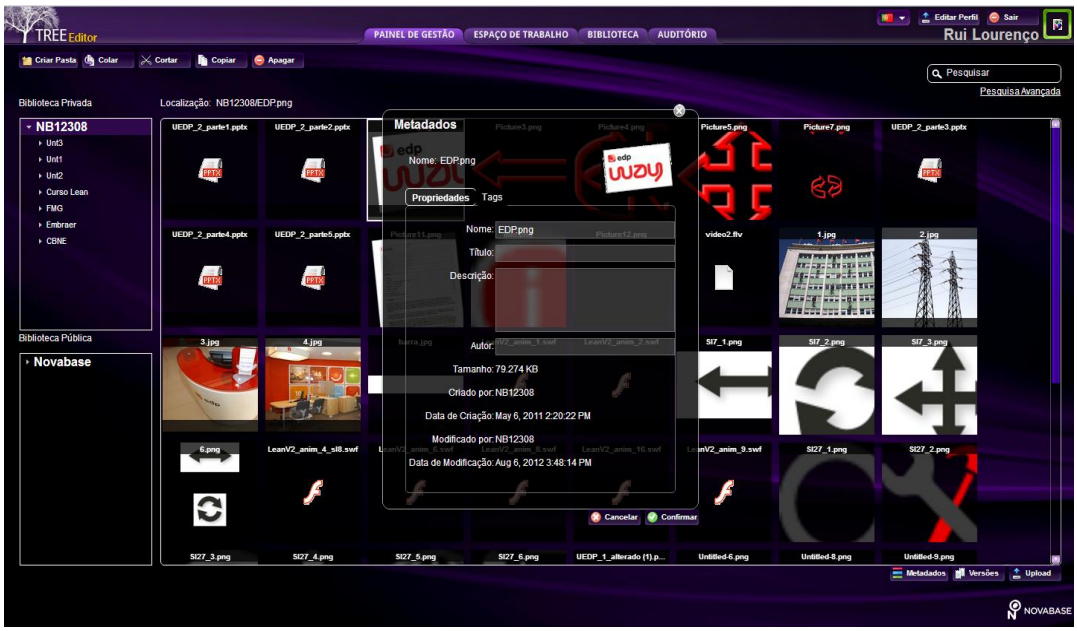

Figura 12 - Interface de gestão das propriedades do objecto

Para as etiquetas, definidas por *tags*, foi desenvolvido o *web script* para retornar as etiquetas de um determinado objecto ou nó, utilizando para isso o método GET, outro para adicionar etiquetas ao objecto utilizando o método POST. Também para as etiquetas foi criada uma interface própria de gestão das mesmas na área dos metadados. A Figura 13 mostra essa interface.

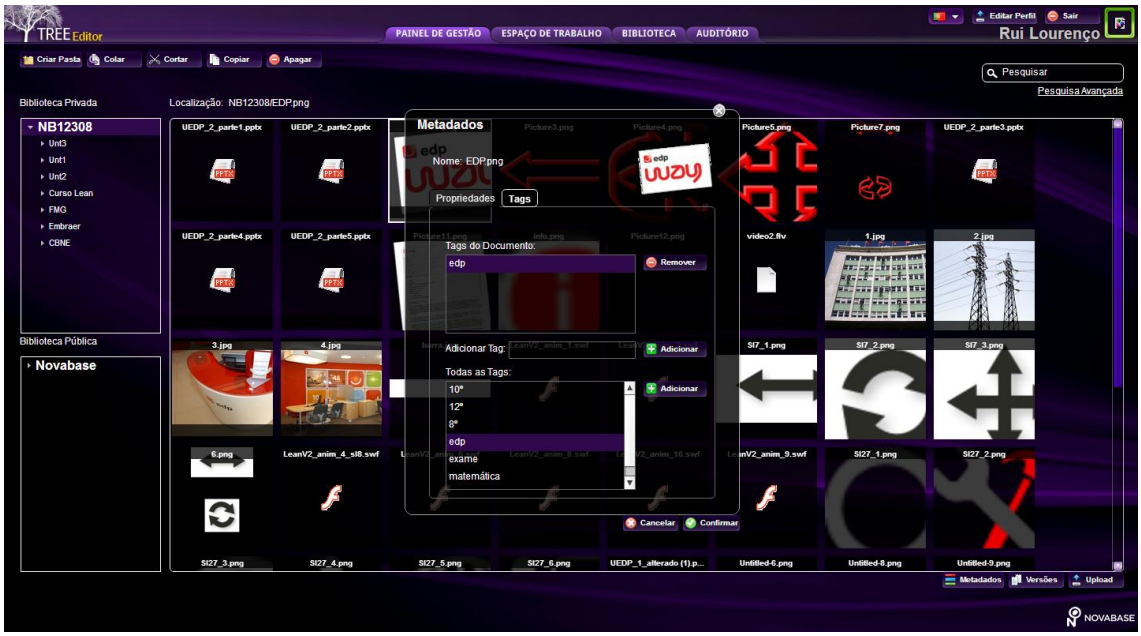

Figura 13 - Interface de gestão das etiquetas do objecto

4. *Pesquisar objectos*

A funcionalidade pesquisa foi desenvolvida utilizando o motor de pesquisa Lucene que faz parte da plataforma Alfresco. Todas as propriedades e etiquetas pesquisáveis pertenciam às características dos objectos no gestor de conteúdo. Criou-se então um *web script* que executa consultas ao Lucene e devolve o resultado dessas consultas utilizando o método GET. Foi definido que esta funcionalidade teria duas interfaces, uma para efectuar uma pesquisa rápida pelo nome do objecto e uma para a pesquisa avançada que possibilita executar a pesquisa pelos metadados e etiquetas acima referidos e pelo tipo de objecto. Definidos os campos a pesquisar passou-se à criação das consultas necessárias para executar estas tarefas. A Figura 14 mostra a interface da pesquisa avançada.

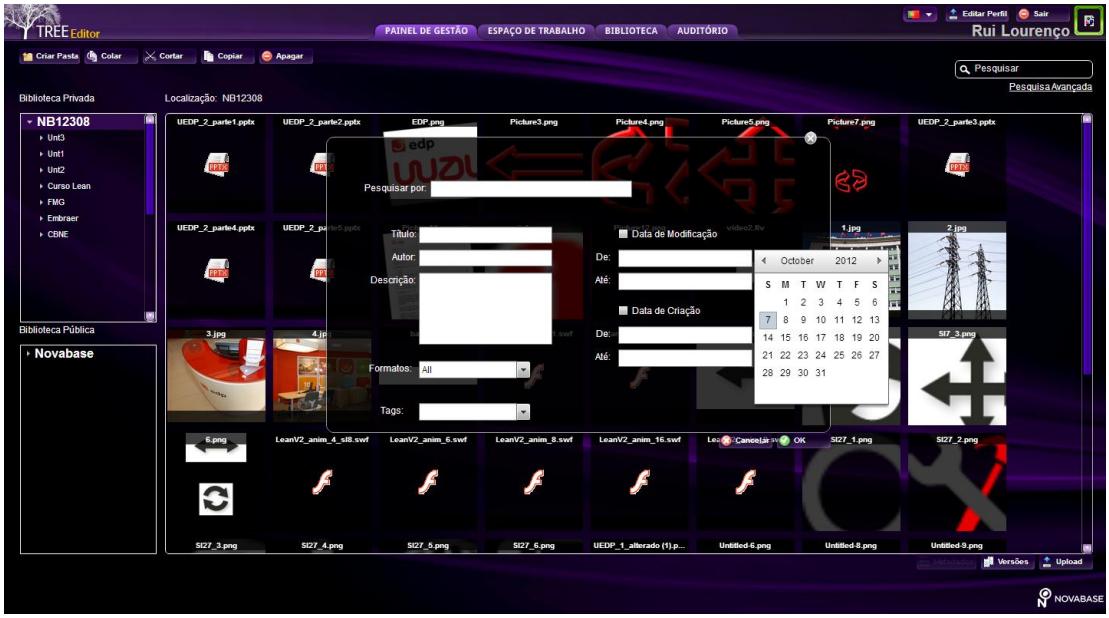

Figura 14 - Interface da Pesquisa Avançada

5. *Versões de objectos*

A última funcionalidade a ser desenvolvida foi a gestão das versões dos objectos. É importante num gestor de conteúdo poder recuperar-se versões antigas dos objectos e manter um histórico dos mesmos. Para cumprir esta tarefa verifiquei que a plataforma Alfresco possui essa funcionalidade, apesar de não estar activada por defeito no produto. Activar esta funcionalidade não é no entanto trivial, uma vez que não existe na interface da aplicação nenhuma operação que o permita fazer. A activação do controlo automático de versões é efectuada através do ficheiro de configuração das propriedades do Alfresco, um ficheiro em formato XML que possui todas as propriedades da plataforma e que se encontra em "/config/alfresco/model/contentModel.XML". Este controlo é feito através da propriedade "cm:versionable", colocando-o a "true".

Colocada a plataforma Alfresco com esta opção activa, passou-se ao desenvolvimento de um *web script* que retorna as versões do objecto, seleccionado de forma a poder visualizá-las utilizando para esse efeito o método GET. A Figura 15 mostra a interface onde se acede às versões anteriores do objecto de aprendizagem.

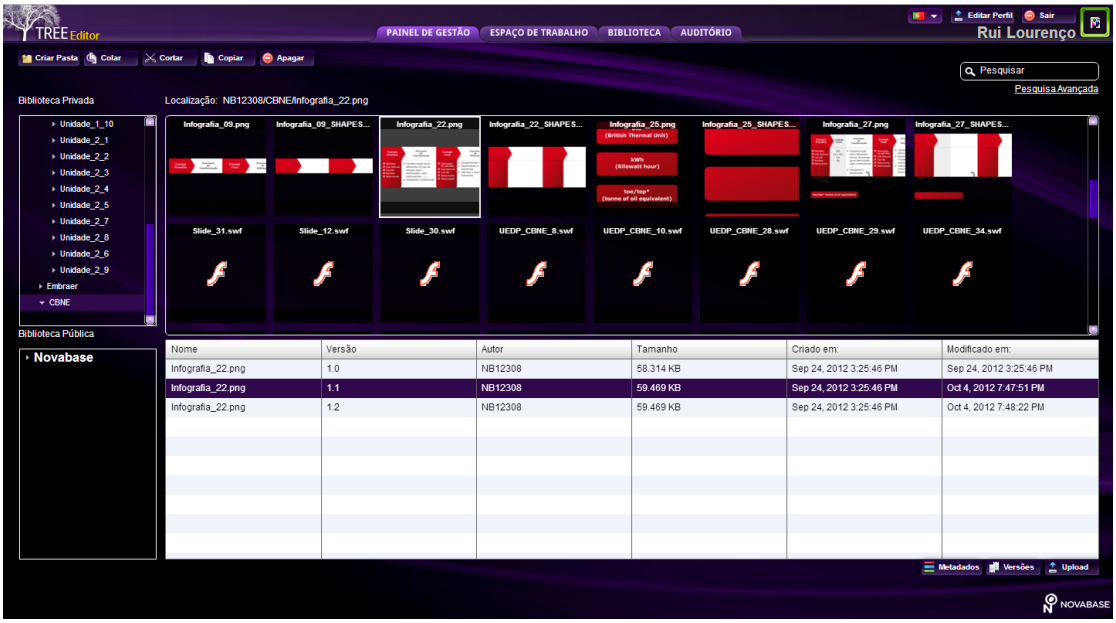

Figura 15 - Interface da Biblioteca com acesso às versões

4.2 Importador de ficheiros *Powerpoint* (.pptx)

O importador de *powerpoint* era um requisito do projecto inicial que ao ser analisado, verificou-se que se enquadrava no âmbito do módulo de gestão documental ao invés da ideia inicial de pertencer ao módulo do editor de cursos e ecrãs da aplicação. Assim, o seu desenho e desenvolvimento enquadraram-se no âmbito deste estudo. O objectivo deste importador consistia em retirar os objectos criados no *powerpoint* e replicar aqueles que tivessem correspondência na aplicação. Muitos cursos de e-learning passam primeiro por uma aprovação em guião, normalmente feito num ficheiro *powerpoint*. A possibilidade de importar esse guião para um curso implica um ganho de tempo, logo de produtividade, na construção dos mesmos. Nesta fase do projecto os objectos a importar seriam as caixas de texto e as imagens, objectos já com uma representação XML criada no editor da aplicação

4.2.1 Arquitectura

Tendo em atenção a decisão previamente tomada em utilizar tecnologia de código aberto, o facto de já existir na aplicação um servidor TOMCAT, a experiencia do desenvolvedor e o facto de o JAVA proporcionar o desenvolvimento de serviços JAX-RPC, optou-se por utilizar essa tecnologia no desenho e desenvolvimento do serviço web. Assim desenhou-se a arquitectura do componente que sendo um serviço web JAX-RPC iria comunicar com o módulo biblioteca através de SOAP, com o Alfresco através de HTTPRequest. A figura seguinte mostra a arquitectura do módulo.

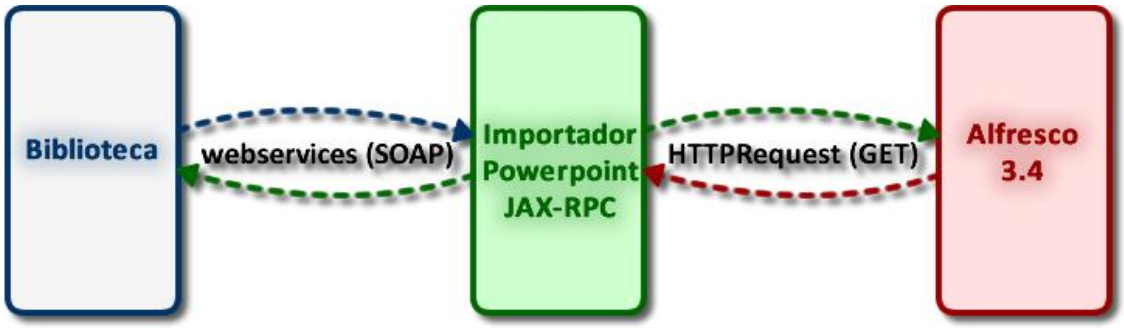

Figura 16 - Arquitectura do importador de *Powerpoint*

O processo de importação de *powerpoint* inicializa no componente que lista o curso, ao adicionar um novo ecrã é sugerido ao utilizador que seleccione um ecrã dos *templates* predefinidos ou que importe um ou vários a partir de uma apresentação *powerpoint*.

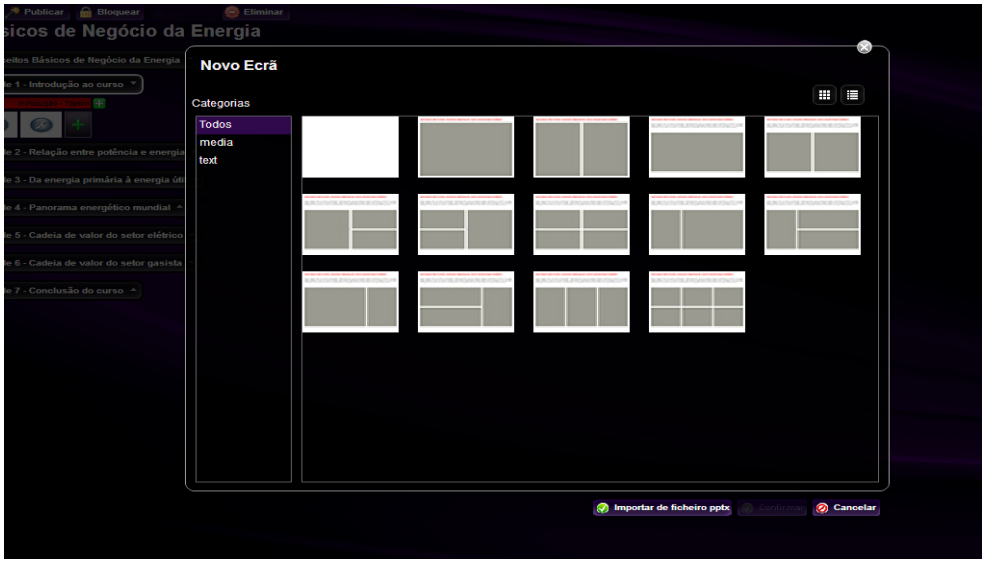

Figura 17 - Interface onde se adicionam ecrãs

O utilizador ao indicar que deseja importar o *powerpoint* irá para a biblioteca onde pode seleccionar a apresentação pretendida, se esta já se encontrar na biblioteca, ou fazer o carregamento da mesma para a biblioteca, em seguida seleccionar a apresentação e clicar o botão "ok".

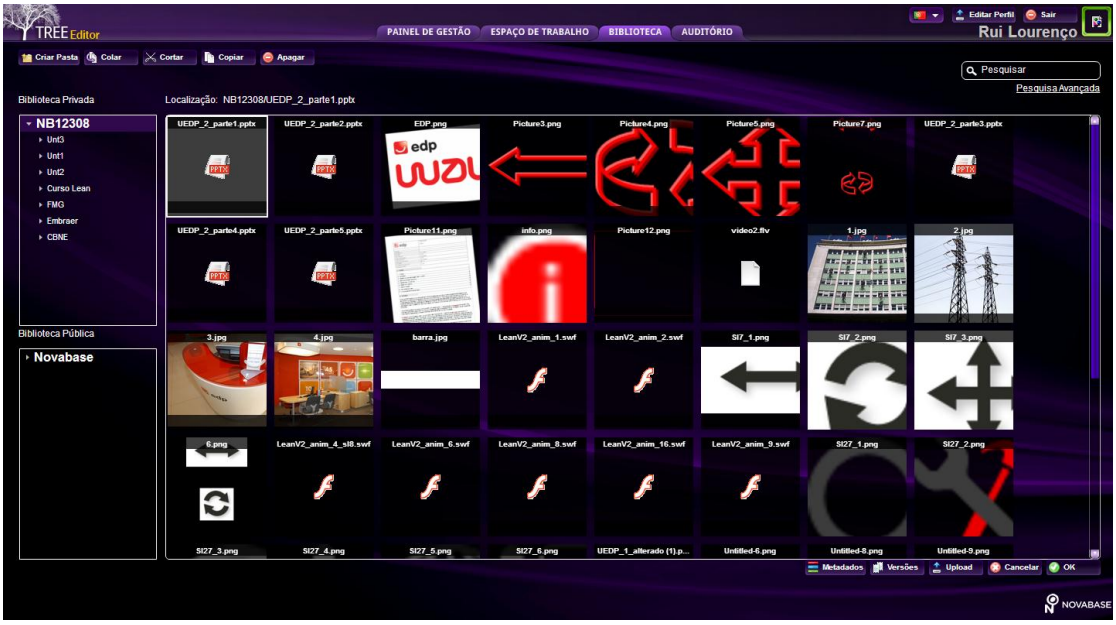

Figura 18 - Interface Biblioteca para seleccionar o PPTX

A partir daqui a biblioteca executa um pedido ao serviço web através de SOAP, enviando o identificador do curso a que pertence o pedido e o URL do ficheiro. O serviço ao receber os parâmetros faz download do documento para uma pasta específica através do URL executando uma chamada por HTTPRequest (GET). O serviço então descompacta a apresentação para uma pasta com o mesmo nome de forma a poder aceder aos ficheiros que compõem o formato Office OPENXML. Neste momento executa o *parser* dos ficheiros, cria as imagens pertencentes a apresentação na pasta do curso a que pertence a importação através de um serviço SOAP e retorna o XML dos ecrãs, que por sua vez o encaminha para o módulo editor de cursos e ecrãs para serem tratados e convertidos em ficheiros físicos no formato XML, mostrando ao utilizador a árvore do curso e onde foram inseridos. Como a criação das miniaturas são executadas quando se acede ao ecrã, estes novos ecrãs são facilmente identificáveis pois todos possuem a mesma miniatura. Facilmente identificável na figura seguinte.

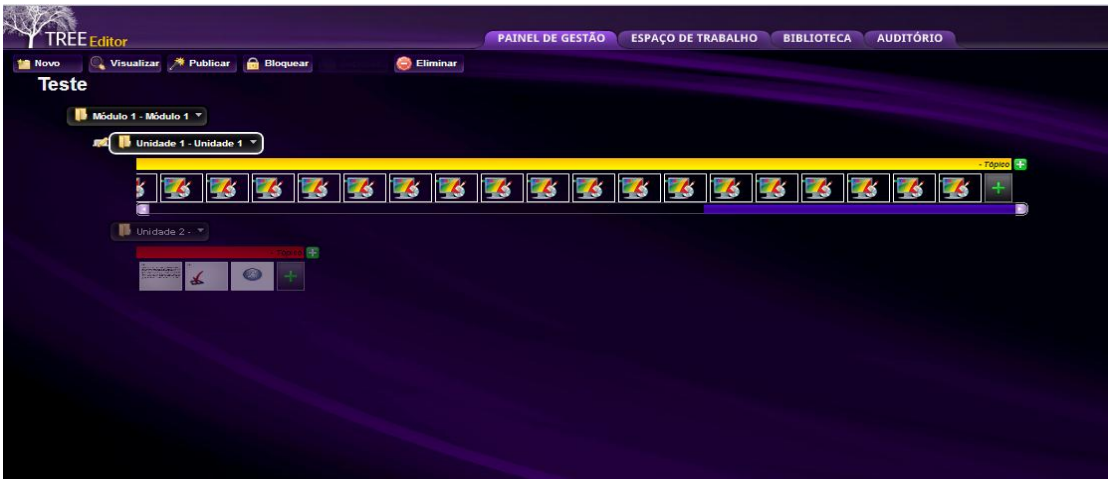

Figura 19 - Interface da árvore de um curso

4.2.2 As funcionalidades

Nesta fase do projecto foi definido que os objectos a importar do ficheiro *powerpoint* fossem o texto e as imagens. Estes dois objectos já estavam desenhados e definidos no editor. Não teria sentido importar objectos que depois não pudessem ser visualizados no curso.

As apresentações estão no formato Open XML Format *Powerpoint* presentation, o formato introduzido pelo Office 2007, que é na realidade uma pasta compactada num ficheiro *zip* com a extensão ".pptx" que possui uma hierarquia de pastas e ficheiros (41). A imagem seguinte mostra o conteúdo de um ficheiro *powerpoint* neste formato.

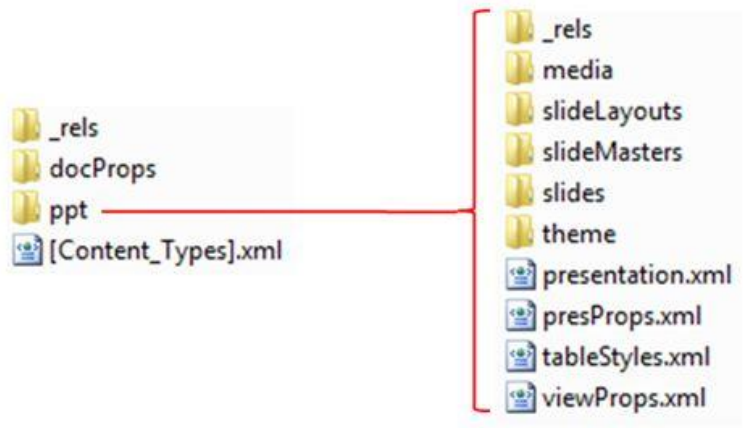

Figura 20 - Estrutura de uma apresentação em OPEN XML *Powerpoint* Presentation (41)

Da estrutura acima os elementos utilizados para este estudo foram:

- A pasta slides, onde estão os ficheiros XML de cada slide.
- A pasta _rels, onde estão os ficheiros XML das relações que cada slide possui.
- A pasta *media*, onde estão as imagens do curso.

Em seguida irei descrever detalhadamente o processo de importação das imagens e textos do *powerpoint* e explicar onde os elementos supra mencionados incorporam o processo.

A importação inicializa, como referido no ponto anterior, numa chamada ao serviço web, enviando o endereço URL do ficheiro e o identificador do curso a que pertence. Quando recebe o pedido, o serviço faz o download do ficheiro para uma pasta designada e extrai o ficheiro para uma pasta com o seu nome. Com a apresentação extraída, o serviço carrega o XML do primeiro slide da apresentação que se encontra na pasta slides (Figura 20). Inicia então a análise do XML em busca das etiquetas que marcam a caixa de texto e a imagem. Quando se encontra uma das etiquetas:

- Quando a etiqueta é de texto, é criado um objecto JAVA com a informação dessa etiqueta e dos seus filhos. As propriedades retiradas do XML para construir a caixa de texto no editor são:

1. O posicionamento através das coordenadas X e Y.
2. A sua dimensão pela altura e largura.
3. A fonte utilizada pelo texto, que caso não exista na aplicação é transformada em "Arial".
4. O tamanho de letra.
5. A cor do texto.
6. Os estilos *itálico*, **negrito** e <u>sublinhado</u>.
7. Texto.

Retiradas estas propriedades, constrói-se o XML no formato utilizado pelo editor da aplicação que irá replicar a caixa de texto no módulo editor quando receber este XML. Assim, com o posicionamento e a dimensão é desenhada a caixa de texto, sendo que as outras propriedades são utilizadas para converter o texto em htmlText, o formato interpretado pelo editor. Finalizado este processo, o XML é guardado num vector, para quando o slide acabar de ser tratado, ser inserido no XML final do ecrã.

- Quando a etiqueta encontrada pertence a uma imagem, é criado um objecto JAVA com a informação dessa etiqueta e dos seus filhos. No caso da imagem, é necessário procurar também pela etiqueta que tem a informação sobre as relações dos slides, uma vez que se vários slides partilharem uma imagem essa não existe duplicada, é inserida no primeiro slide e todos os slides que a utilizam criam uma relação com este. Esta informação existe nos ficheiros XML existentes na pasta _rels (Figura 20). As propriedades retiradas para a construção do objecto imagem no editor são:

1. O posicionamento através das coordenadas X e Y.
2. A dimensão, pela altura e largura.
3. O seu nome.

Estando na posse destas propriedades, é construído o XML referente ao objecto "*MediaObject*" reproduzido no editor. As imagens numa apresentação com este formato encontram-se na pasta *media* (Figura 20), sendo o passo seguinte copiar a imagem para a pasta *assets* do curso receptor desta importação. Finalizado este processo, o XML é guardado num vector para quando o slide acabar de ser tratado, ser inserido no XML final do ecrã.

Finalizado o processo para o primeiro slide, o XML do ecrã é construído e guardado num vector. E passa ao próximo slide. Ao analisar todos os slides da apresentação, o serviço retorna para a biblioteca, um vector com um ecrã em cada posição. Esta informação é então enviada para o módulo editor de texto que cria os ecrãs fisicamente no curso.

Capítulo V – Conclusão e Trabalho Futuro

Esta dissertação iniciou com a especificação do projecto onde a mesma foi inserido, os objectivos e motivação que levaram ao desenvolvimento deste estudo. Introduziu-se a temática do projecto na sociedade actual e a sua visão. Identificou-se a necessidade que levou ao desenvolvimento deste estudo, os seus requisitos e restrições. Descreveram-se as ferramentas tecnológicas utilizadas, as várias fases por que passou o desenho e implementação desta solução de gestão documental e os vários desafios que surgiram durante o seu desenvolvimento.

Para além da concepção foram implementados dois módulos com funcionalidades de diferentes níveis de complexidade, onde se salienta:

- Desenvolvimento do módulo biblioteca, capaz de efectuar a gestão dos objectos de aprendizagem dos utilizadores.

- Desenvolvimento de pesquisas Lucene para a construção da funcionalidade pesquisa avançada.

- Desenvolvimento de um conjunto de *web script*s para interligar a plataforma Alfresco com a aplicação TREE, nomeadamente com a biblioteca, proporcionando a implementação das suas funcionalidades.

- Desenvolvimento de um serviço importador de *powerpoint*, um serviço à data da sua criação único na área das ferramentas de autor.

Nas duas secções seguintes são apresentadas as conclusões finais sobre este estudo e a visão para o trabalho futuro.

5.1 Conclusões

O desenho e a implementação dos módulos, que juntos são responsáveis pela gestão documental da aplicação, que são descritos nesta dissertação foram para mim um enorme desafio. Tendo em conta a diferente tecnologia utilizada, assim como a exigência de construção de um módulo fulcral para o sucesso da aplicação, permitiram o meu crescimento, não só a nível profissional como pessoal.

Os objectivos que me foram propostos no início desta longa jornada foram atingidos com sucesso. Prova disso é a utilização diária da aplicação pela equipa de desenvolvimento de conteúdo de e-learning da Novabase, assim como os vários clientes que já possuem cursos desenvolvidos pela aplicação TREE, como é o caso da EDP, da Caixa Geral de Depósitos e da SONAE, entre outros.

Gostava de salientar que apesar de actualmente já existirem ferramentas de autor com importadores de *powerpoint*, não existe nenhuma desenvolvida em Portugal, que é sem dúvida uma mais-valia para a área de Business Solutions - e-learning & HCM da Novabase possuir uma plataforma própria para desenvolvimento e criação de conteúdo, com total liberdade para a fazer evoluir à medida das suas necessidades.

No módulo da biblioteca, a possibilidade de criar versões dos objectos de aprendizagem, permitindo não só guardar o histórico do mesmo como saber quem foi a pessoa que alterou o objecto, surpreendeu pela forte utilização por parte de quem faz a gestão dos objectos de aprendizagem. Esta funcionalidade estava projectada desde início mas com o mais baixo grau de prioridade, tendo sido mesmo a última a ser implementada.

5.2 Trabalho Futuro

Tratando-se da primeira versão da aplicação TREE e consequentemente da gestão documental da mesma, existe muito espaço para fazer crescer os módulos responsáveis pela gestão documental inseridos neste estudo.

O módulo da biblioteca pode evoluir no sentido de potencializar a plataforma Alfresco que o suporta. Em seguida são propostas algumas funcionalidades novas que poderão ser implementadas de forma a melhorar este módulo:

- A criação de uma interface para gerir fluxos de trabalho com aprovações de objectos de aprendizagem.

- A integração da biblioteca com as redes sociais no sentido de uma maior partilha dos objectos de aprendizagem.

- A detecção automática do tipo de objecto de aprendizagem que é carregado na biblioteca e a automatização dos seus metadados.

- Acesso à biblioteca através de uma aplicação móvel.

No que diz respeito ao módulo do importador de *powerpoint*, existe um caminho futuro bem definido. O aumento da variedade de objectos a serem importados e replicados no editor do curso. Este caminho está identificado, sendo a sua implementação mais demorada derivado ao desenvolvimento que terá de ser efectuado nos dois módulos.

Referências Bibliográficas

1. COMISSÃO DAS COMUNIDADES EUROPEIAS. European Commission Education & Training. [Online] [Cited: 02 2, 2012.] http://ec.europa.eu/education/lifelong-learning-policy/doc/com868_en.pdf.

2. AIIM. [Online] [Cited: 09 04, 2012.] http://www.aiim.org/What-is-ECM-Enterprise-Content-Management.

3. Digital Landfill. [Online] [Cited: 05 06, 2012.] http://aiim.typepad.com/aiim_blog/2009/08/8-things-that-changed-the-history-of-document-management.html.

4. *The Evolution of ECM: Platform Oriented, Flexible, Architected for the Cloud and Designed for Technologists.* s.l. : Noxio, 2011.

5. *State of the ECM Industry 2011.* Miles. s.l. : AIIM, 2011.

6. Comunidade Portuguesa de utilizadores da framework Scrum. [Online] [Cited: 03 17, 2012.] http://scrumpt.com/.

7. Mountaingoatsoftware. [Online] [Cited: 03 20, 2012.] http://www.mountaingoatsoftware.com/scrum/overview.

8. Apache HTTP Server. [Online] http://httpd.apache.org/.

9. Mysql. [Online] http://www.mysql.com/.

10. PHP. [Online] http://www.php.net/.

11. Alfresco. [Online] http://www.alfresco.com/.

12. Adobe Flex. [Online] http://www.adobe.com/products/flex.html.

13. XML. [Online] http://www.w3.org/XML/.

14. Java EE. [Online] http://www.oracle.com/us/technologies/java/overview/index.html.

15. Apache Tomcat. [Online] http://tomcat.apache.org/.

16. Actionscript 3.0. [Online]
http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html.

17. Apache Lucene. [Online] [Cited: Março 20, 2012.] http://lucene.apache.org/.

18. Plataforma Java Spring. [Online] [Cited: Março 20, 2012.]
http://www.springsource.org/.

19. Java Hibernate. [Online] [Cited: Março 20, 2012.] http://www.hibernate.org/.

20. jBPM. [Online] [Cited: Março 20, 2012.] http://www.jboss.org/jbpm.

21. FreeMarker. [Online] http://freemarker.sourceforge.net/.

22. Apache POI. [Online] [Cited: Março 20, 2012.] http://poi.apache.org/.

23. Alfresco Documentation. [Online] [Cited: Março 2012, 21.]
http://docs.alfresco.com/3.4/index.jsp.

24. Alfresco *Web script*s. [Online] [Cited: Março 2012, 26.]
http://wiki.alfresco.com/wiki/3.0_Web_Scripts_Framework.

25. Apache Flex. [Online] [Cited: 04 01, 2012.] http://incubator.apache.org/flex/.

26. Learn Adobe. [Online] [Cited: 03 21, 2012.]
http://learn.adobe.com/wiki/display/Flex/Get+oriented+to+Flex.

27. Adobe ActionScript 3.0. [Online] [Cited: Abril 2012, 01.]
http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html.

28. About MXML. [Online] Adobe. [Cited: 09 2012, 05.]
http://livedocs.adobe.com/flex/3/html/help.html?content=mXML_2.html.

29. Eclipse. [Online] [Cited: 08 22, 2012.] http://www.eclipse.org/.

30. Adobe Flash Builder. [Online] 08 22, 2012. http://www.adobe.com/products/flash-builder.html.

31. Walsh, Norman. *A Technical Introduction to XML*. s.l. :
http://www.XML.com/pub/a/98/10/guide0.html, 2008.

32. XML CORE. *W3C.* [Online] [Cited: 10 11, 2012.] http://www.w3.org/standards/XML/core.

33. JAVA JAX-RPC. *JAVA SUN.* [Online] http://java.sun.com/developer/technicalArticles/WebServices/getstartjaxrpc/.

34. *OpenXML.* [Online] [Cited: 09 2012, 15.] http://www.openXML.biz/.

35. Non, Tom. OFFICE OPEN XML OVERVIEW. *Ecma-International.* [Online] [Cited: 09 12, 2012.] http://www.ecma-international.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf.

36. NetBeans History. *Netbeans.* [Online] [Cited: 09 10, 2012.] http://netbeans.org/about/history.html.

37. *CMS Wires.* [Online] [Cited: 09 18, 2012.] http://www.cmswire.com/cms/enterprise-cms/adobe-to-oem-alfrescos-enterprise-cms-002794.php.

38. Potts, Jeff. Alfresco Developer: Intro to the *Web script* Framework. 2007.

39. Caruana, David, *Professional Alfresco Pratical Solutions for Enterprise Content Management.* Indianapolis : Wiley Publishing, Inc, 2010. ISBN: 978-0-470-57104-o.

40. Ugo Cei, Piergiorgio Lucidi. *Alfresco 3 Web Services.* s.l. : PACKT Publishing, 2010.

41. Microsoft. Using Office Open XML to Save Time Without Writing Code. [Online] [Cited: 09 05, 2012.] http://msdn.microsoft.com/en-us/library/office/dd627338(v=office.12).aspx.

42. Potts, Jeff. *Alfresco Developer Guide.* s.l. : PACKT Publishing, 2008.

43. *W3C.* [Online] [Cited: 09 11, 2012.] http://www.w3.org/standards/XML/core.

44. Alfresco wiki. *Alfresco.* [Online] [Cited: 09 2012, 22.] http://wiki.alfresco.com/wiki/Release_2.1#Web_Content_Management.