Universidade do Minho

MEI - Engenharia de Linguagens

Arquivo Digital de Trabalhos Práticos

Eduardo Ribeiro — *pg11205@alunos.uminho.pt*

Miguel Nunes — *a40643@alunos.uminho.pt*

29 de Junho de 2008

Resumo

O enorme crescimento dos recursos computacionais, largura de banda e conectividade originou uma explosão no número de comunidades e organizações que disponibilizam informação digital em todo o tipo de áreas. Hoje em dia, transacções entre todo o tipo de organizações são conduzidas usando, como meio de transmissão, a plataforma digital, ao invés do tradicional papel. Estas inovações causam grandes problemas no que toca à preservação de informação digital e também a certas organizações que, na altura de formação, não foram desenhadas para desempenhar a tarefa de catalogar e registar informação. Daí o aparecimento das *Open Archival Information System* (OAIS). [1] Com o uso de ferramentas próprias direccionadas à web foi desenvolvido com sucesso uma aplicação web de Arquivo Digital de Trabalhos Práticos de Alunos (ADTP) baseada no modelo OAIS, seguindo a orientação das questões propostas para o projecto.

Conteúdo

1 Introdução

Para o trabalho do Projecto Integrado da UCE de Engenharia de Linguagens, foi pedido o desenvolvimento de um Arquivo Digital de Trabalhos Práticos de Alunos (ADTP) baseado no modelo *Open Archival Information System* (OAIS).

O modelo OAIS é um arquivo que consiste numa organização de pessoas e sistemas que aceitaram a responsabilidade de preservar informação e disponibilizá-la para uma dada comunidade. Um arquivo baseado no modelo OAIS tem de respeitar certas e determinadas regras e responsabilidades que serão descritas mais adiante.

Devido ao modo como hoje em dia a preservação digital de documentos afecta uma enorme variedade de comunidades, é prático destilar este problema num grupo de conceitos para permitir um variado leque de aplicações de arquivos digitais. Desta forma torna-se muito mais simples a interoperabilidade entre bibliotecas, arquivos e outras instituições. Isto é de relativa importância pois permite que um dado grupo de instituições, que antes não teriam oportunidade de trabalhar em cooperação, passar a permitir tal. A adopção deste modelo pode também ter um potencial económico. Ao aplicar os mesmos standards, as diferentes entidades podem reduzir os custos de transmissão de informação entre si usando componentes partilhadas desse sistema.

Ainda não é possível determinar ao certo se este modelo chegará a um consenso sobre a manutenção de informação digital. Segundo os últimos dados, a utilização do modelo OAIS tem aumentando, se bem que existam outros modelos semelhantes em crescimento. De qualquer modo, o modelo OAIS é uma importante base para uma solução coordenada e de aplicação extensa. Nas próximas secções estão explanadas a definição de OAIS, os pormenores do trabalho solicitado e ferramentas usadas para a concretização deste trabalho.([4] e [5])

2 Enunciado do Trabalho - Primeiro Semestre

Neste capítulo será exposto o enunciado do trabalho, de modo a que o leitor compreenda quais os objectivos do trabalho realizado. O texto seguinte é retirado directamente da página do Projecto Integrado.

O que se pretende neste projecto é desenvolver um Arquivo Digital de Trabalhos Práticos de Alunos, que doravante designaremos por ADTPs. Basicamente, o sistema recebe um pacote de submissão (SIP) com vários ficheiros de tipos diversos (por exemplo, TeX, BibTeX, imagens, PDFs, código-fonte, programas executáveis). Juntamente com os ficheiros do Trabalho Prático a arquivar segue um manifesto num dialecto XML standard (METS) que define o conteúdo do pacote e permite ao sistema fazer o seu arquivo.

Além disso, esse manifesto também conterá a classificação (hierárquica ou não), descrita segundo um padrão a escolher de entre os seguintes: Dublin Core Metadata, Encoded Archival Description (EAD) e Dublin Core + Conjunto de Elementos definido pela equipe de projecto, e alguns documentos XML contendo metainformação técnica e/ou descritiva sobre cada um dos ficheiros.

O sistema procede à ingestão do SIP, convertendo o conjunto de ficheiros em formatos própios para armazenamento constituindo um novo pacote designado por pacote de arquivo (AIP). Esse AIP é depois transformado, segundo regras de difusão próprias do sistema, num conjunto de novos pacotes prontos para serem difundidos/distribuídos, designados por pacotes de disseminação (DIP). O ADTPs realiza sobre o AIP um vasto conjunto de operações de gestão do arquivo, destinadas a pequisar e a manter os ficheiros.

2.1 Objectos Digitais

Um Arquivo Digital não é mais do que um repositório de Objectos Digitais (OD) e é estes que é preciso definir em cada contexto de aplicação. No nosso caso, o OD é um trabalho prático e a sua estrutura e composição deverá ser pensada e especificada. O grau de liberdade é grande no entanto sugere-se o seguinte esqueleto estrutural como ponto de partida:

- um trabalho prático pode ser constituído por três componentes: relatório, apresentação, aplicação;
- o relatório deverá ser constituído por um ficheiro em PDF podendo opcionalmente ter ainda o ficheiro TeX ou XML que lhe deu origem (será necessário pensar quais os campos de metainformação técnica que faz sentido acrescentar para cada um destes tipos de ficheiro);
- a apresentação deverá ser constituída por um ficheiro em PDF podendo opcionalmente ter ainda o ficheiro PPT ou XML que lhe deu origem (será necessário ... metainformação técnica ...);

- a aplicação deverá ser constituída por um conjunto de ficheiros zipado (código fonte, makefile, README, ...);
- para alem de tudo isto o OD terá alguns campos de metainformação própria. A título de exemplo sugerem-se os seguintes:
  - título, subtítulo
  - autor(es) (nome, email, id, ...)
  - orientador (nome, email, ...)
  - contexto (disciplina, curso, ...)
  - palavras-chave (a extrair da taxonomia da ACM)
  - resumo
  - links para recursos relacionados
  - ...

2.2 Tarefas a Desenvolver

Para concretizar o trabalho será necessário realizar as seguintes tarefas, algumas das quais podem prosseguir em paralelo:

- Análise, caracterização e especificação da estrutura e composição de um OD;
- Derivação do modelo relacional que servirá para armazenar os ODs (depois de armazenados passarão a ser AIP);
- Criação da base de dados (por exemplo em MySQL);
- Criação de uma interface Web (Website) para incorporação de ODs (trata-se de criar um mecanismo de ingestão semi-automática);
- Criação de uma interface Web para pesquisar e aceder aos ODs armazenados (HTML + PHP, JAVA, Perl, ...);
- Análise do problema com vista a conceber a arquetectura do sistema ADTPs e a funcionalidade a incluir (será usada a notação UML para a análise);
- Identificação dos standards a usar para o manifesto e os formatos de arquivo.

3 Enunciado do Trabalho - Segundo Semestre

Para o segundo semestre foram feitas novas propostas de modo a tornar o projecto mais completo e eficiente. Encontra-se de seguida o enunciado das tarefas a desenvolver e no capítulo seguinte, as várias resoluções serão explanadas.

3.1 Disseminação

Em relação ao trabalho desenvolvido, é necessário completar o ciclo de vida de um arquivo com a disseminação de pacotes. Assim, após a pesquisa de um trabalho prático (e de o utilizador ter encontrado o trabalho em causa), deve ser possível:

- descarregar cada trabalho prático num ZIP completo;
- descarregar um ZIP com o Relatório (ou apresentação, ou código)
- Para ajudar a selecção do conteúdo que se quer descarregar, deve ser possível
  - navegar interactivamente sobre o filesystem do Trabalho Prático, acedendo on-line à árvore de directorias, e vizualização de ficheiros.

3.2 Recuperação de documentos pesquisando o conteúdo (IR)

Implementar um sistema de pesquisa textual sobre os ficheiros do relatório (em alternativa ou complemento, sugere-se a indexação de código fonte para pesquisa por nomes de funções, ou mesmo pesquisa textual sobre código C).

3.3 Implementação de Web Services

Implementar uma API de pesquisa sobre o arquivo através de WebServices , de modo a que os vários arquivos possam consultar a base de trabalhos práticos dos outros arquivos

API
* TextSearch: Query —-> Result*
o Query: String do género: word AND word
o Result: Um resultado é um bloco HTML com título, link para o objecto digital, e autores.

* TaxonomySearch: Category+ —-> Result*
o Category: Código ACM de uma categoria da taxonomia
o Result: Um resultado é um bloco HTML com título, link para o objecto digital, e autores.

* KeywordSearch: Keyword+ —-> Result*
o Keyword: uma palavra chave ou um par ”campo:palavra-chave”. Os campos suportados: título, autor, oientador e resumo.
o Result: Um resultado é um bloco HTML com título, link para o objecto

digital, e autores.

4 OAIS

Como anteriormente referido, teoricamente o modelo OAIS segue determinadas responsabilidades mínimas de modo a possibilitar o bom funcionamento e aplicação do arquivo em desenvolvimento. Essas regras([5],[4]) são:

- Negociar e aceitar informação correcta e apropriada dos autores/produtores,
- Obter controlo suficiente sobre a informação para garantir a preservação da informação ao longo do tempo,
- Determinar em que área/quadro se encontra uma certa comunidade,
- Garantir que a informação é compreendida pela comunidade sem a assistência dos produtores/autores,
- Seguir certas políticas e procedimentos para garantir que a informação é bem preservada e se encontra bem protegida, permitindo a distribuição de informação como cópias autenticas da original ou pelo menos com origem na informação original,
- Disponibilizar a informação à comunidade.

De notar que estas regras simples são obrigatórias em todas as aplicações que implementem este modelo. O modelo OAIS pode ser definido graficamente através da Figura 1. De seguida será explicado o que cada um dos elementos significa e qual o seu papel no modelo.

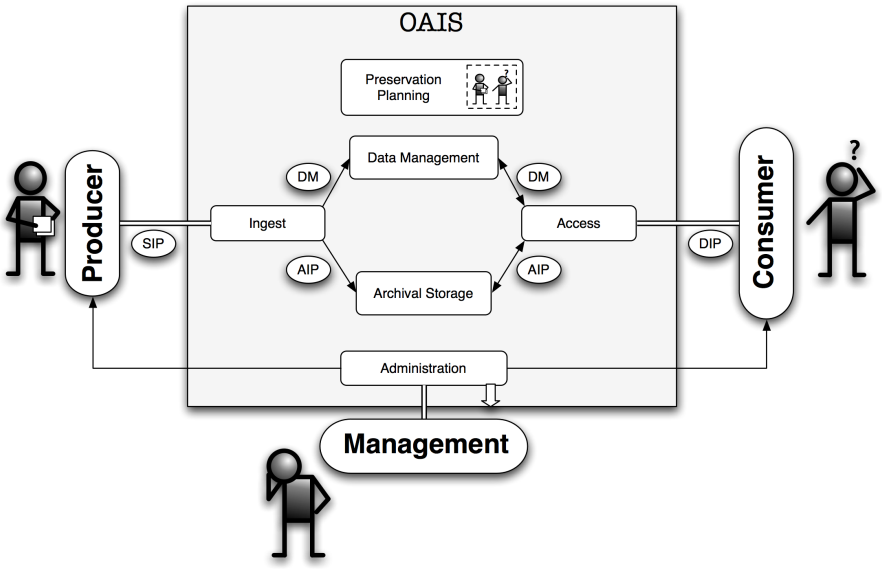

Figura 1: Representação gráfica do Modelo OAIS.

Existem, neste sistema, três tipos diferentes de utilizadores. Os produtores são aqueles que produzem nova informação e a enviam para o sistema para ser guardado e mais tarde acedido pelos consumidores. Os consumidores são os utilizadores que consultam o arquivo OAIS em busca de informação inicialmente designada para a construção do arquivo. O consumidor, tal como o produtor, pertence a uma dada comunidade para o qual o arquivo OAIS está orientado. Por ultimo existem os Administradores. Estes têm a função de manter e actualizar o arquivo, garantido o bom funcionamento deste. Têm acesso aos ficheiros, métodos e funções existentes podendo alterá-los se necessário.

Dentro de um sistema baseado no modelo OAIS existem vários métodos, funções e pacotes. A descrição destes encontra-se de seguida. Existem três tipos de pacotes usados pelo sistema: SIP, AIP e DIP. O SIP (Submission Information Package) é o pacote enviado pelo autor/produtor de informação ao sistema. Este pacote deve ter todos os ficheiros e pastas, e um ficheiro indicador do conteúdo do SIP, isto é, a descrição dos ficheiros e pastas, e a sua organização hierárquica. O AIP (Archival Information Package) é o pacote que é guardado no sistema, após sofrer certas transformações na função de ingestão. O sistema trabalha internamente com este tipo de pacote. Por fim, o pacote DIP (Dissemination Information Package) é o pacote com a informação enviada a pedido de um utilizador do tipo consumidor.

Para se compreender como estes pacotes são gerados, é necessário saber quais os passos que o sistema toma desde a recepção da informação até essa informação ser fornecida aos consumidores. A função de Ingestão é responsável por receber a informação dos produtores e prepará-la para ser tanto arquivada como manuseada dentro do sistema de arquivos. Mais especificamente a função de Ingestão aceita informação dos produtores na forma de SIPs, verifica a qualidade do SIP, gera o AIP e extrai informação do AIP. Finalmente a função de Ingestão transfere o novo pacote AIP para as funções de Armazenagem e Data Management. A função de Armazenagem, tal como o nome indica, é responsável por guardar a as AIPs, mantê-las e fornecê-las à função de Disseminação. Estas responsabilidades incluem receber uma nova AIP da função de Ingestão e guardá-la de acordo com variados critérios (tamanho, utilidade, etc.), migrar AIPs para uma nova localização, verificar erros, implementar soluções contra desastres e fornecer cópias de AIPs á função de Disseminação. A função de Data Management coordena a informação fornecida sobre cada AIP. Em particular, esta função mantém e gere a base de dados que contém a informação, executa pedidos por parte da função de Disseminação e devolve-lhe resultados. Esta função também cria relatórios para suporte das funções de Ingestão, Disseminação e Administração. Actualiza a base de dados da função Data Management, incluindo quando existe a adição de uma nova AIP no sistema. Cabe à função de Administração gerir as operações diárias do arquivo. O objectivo desta função é o de negociar submissões com os produtores, gerir o sistema em si, controlar acessos e fornecer serviços a clientes. Também é executada constantes auditorias aos SIPs para garantir que a informação a ser incluída no sistema está correcta. São desenvolvidas políticas e métodos relacionados com os standards do sistema (formato da informação, documentos necessários, armazenagem, migração, segurança, etc.). Após toda a recepção do SIP e a manutenção do AIP, existe a função de Disse-

minação de modo a que os consumidores tenham acesso à informação guardada no arquivo. Esta função ajuda os consumidores a identificar e a obter descrições da informação contida no arquivo, e entrega informação do arquivo ao consumidor. A função usa apenas um interface para pesquisa e envio de informação. A Disseminação cria DIPs em resposta aos pedidos dos consumidores, obtendo cópias do AIP desejado, obtido da função de Armazenamento, enviando assim um DIP ou resposta a uma pesquisa dos consumidores.

Estas cinco funções controlam o fluxo de informação desde os produtores até aos consumidores, passando pelo arquivo. Juntas, estas funções identificam os processos chave comuns à maioria dos sistemas dedicados à preservação digital de informação. É normal que existam aplicações OAIS que implementem estas funções de modo diferente.[1]

Em resumo o modelo OAIS comporta nove pontos-chave. Três tipos diferentes de utilizadores: o **produtor** que é o autor ou entidade que detém os direitos sobre o material a arquivar, o **administrador** do arquivo que controla as tarefas de transformação e gestão, e o **consumidor** que será o futuro utilizador do sistema e que irá realizar operações de pesquisa, consulta e download. Este sistema também integra três processos. A **Ingestão** que corresponde à entrada de informação no arquivo, **gestão/administração/armazenagem** que corresponde às acções de transformação e gestão dos objectos digitais armazenados, e **Disseminação** que corresponde à difusão/distribuição de objectos digitais. Por último, existem três tipos de pacotes: o **SIP** (Submission Information Package), **AIP** - (Archival Information Package) e **DIP** - (Dissemination Information Package).[2]

5 A Aplicação Desenvolvida - Primeiro Semestre

5.1 Ruby on Rails

Para a execução deste Projecto, foi usada uma *framework* muito eficiente no que toca à programação orientada à web. Esta *framework* desenvolvida em Ruby, chamada Ruby on Rails, constrói facilmente aplicações para a web. Esta *framework* trabalha sobre o modelo Model-View-Controller. Os termos previamente referidos serão explicados nos próximos parágrafos. Ruby é uma linguagem open source de *scripting* orientada aos objectos com semelhanças com outras linguagens de programação como Perl, Lisp, Dylan e CLU (wikipedia). Foi desenvolvida no início dos anos noventa por Yukihiro Matsumoto. Uma aplicação web define uma aplicação que é acedida através de um browser, usando a rede como meio de comunicação. Essa rede pode ser a Internet como uma rede privada. Uma *framework* pode ser vista como a fundação de uma aplicação web. É responsável pelos detalhes de baixo nível que se tornam repetitivos e aborrecidos para serem programados, permitindo assim ao utilizador focar-se nas funcionalidades da aplicação a desenvolver. *Frameworks* dão ao utilizador classes que implementam funções comuns usadas em todas as aplicações web, incluindo:

- Abstracção sobre a base de dados, garantido que todas as queries funcionam, independentemente do sistema a ser utilizado (MySql, Oracle, DB2, etc.);
- Templating, reutilizar código por toda a aplicação;
- Gestão de sessões de utilizadores;
- Criação de URL's limpas.

A *framework* também define a arquitectura da aplicação, organizando automaticamente pastas e ficheiros. Pode-se ver uma *framework* como uma aplicação já começada para o utilizador, com uma estrutura bem pensada e esqueleto de código já completamente escrito, deixando os pormenores para o utilizador. Ruby on Rails toma em consideração três princípios fundamentais a nível de software:

- *Convencion over configuration* (diminui o número de decisões, ganhando simplicidade);
- Não repetição;
- Desenvolvimento ágil.

A arquitectura *Model-View-Controller* é um modelo usado pelo Ruby on Rails que separa a aplicação em três componentes: As classes *Model*, *View* e *Controller*. A classe *Model* serve para gerir dados e informação e como esta se relaciona entre si. A classe *View* para gerir objectos de interfaces gráficos, geralmemte é a parte HTML mostrada ao utilizador, e por fim, a classe *Controller* é responsável por responder ao utilizador, acedendo á classe *Model* e enviando informação á classe *View* para este mostrar a respectiva resposta ao utilizador.

Esta separação faz com que os pedidos dos utilizadores sejam processados do seguinte modo:

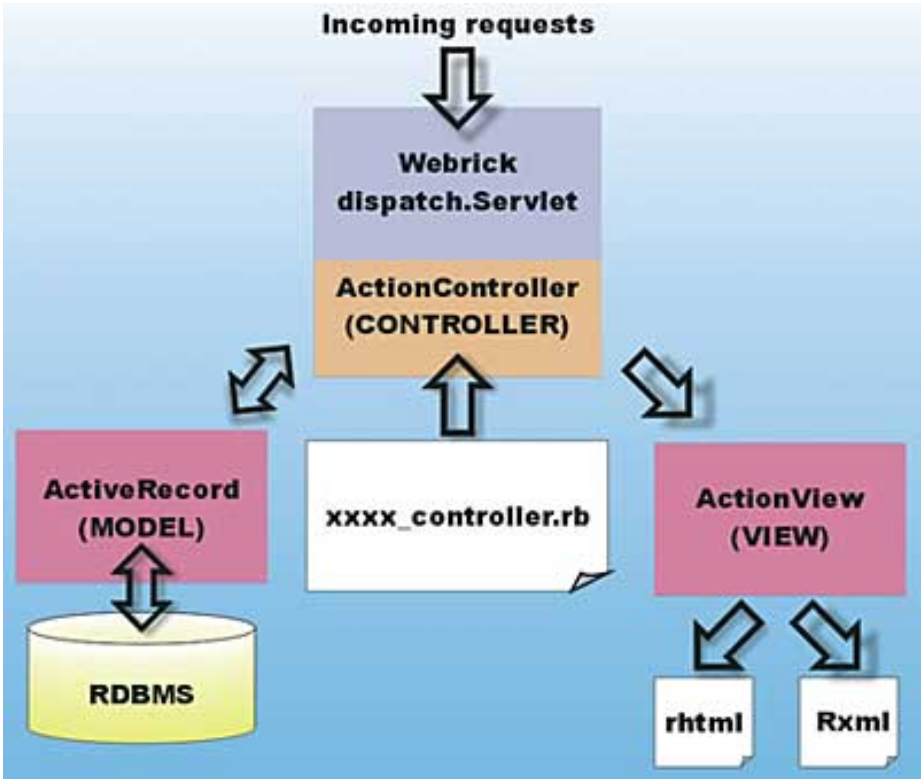

Figura 2: Representação do modelo MVC com as classes de Ruby on Rails

- O browser do cliente envia um pedido para uma pagina ao controller que se encontra no servidor;
- O controller pede ao Model a informação necessária para poder enviar resposta ao cliente;
- O Controller cria a nova página e envia esta ao View;
- O View envia a nova página ao browser do cliente.

As Figuras nº2 e nº3 ilustram o processo descrito anteriormente.

Deste modo é possível à aplicação crescer e assim executar qualquer manutenção a um dos componentes sem afectar os outros. Por exemplo, se se detectar problemas de performance no acesso á base de dados, pode-se executar um upgrade ao hardware sem afectar as outras componentes da aplicação. Assim torna-se muito fácil a manutenção pois os componentes têm baixa dependência entre si. Daí o modelo permitir reutilização, por exemplo um Model pode usar vários Views e vice-versa, e ainda permite a distributividade o que significa que os códigos das várias componentes podem ser encontrados em máquinas diferentes.([6] e [3])

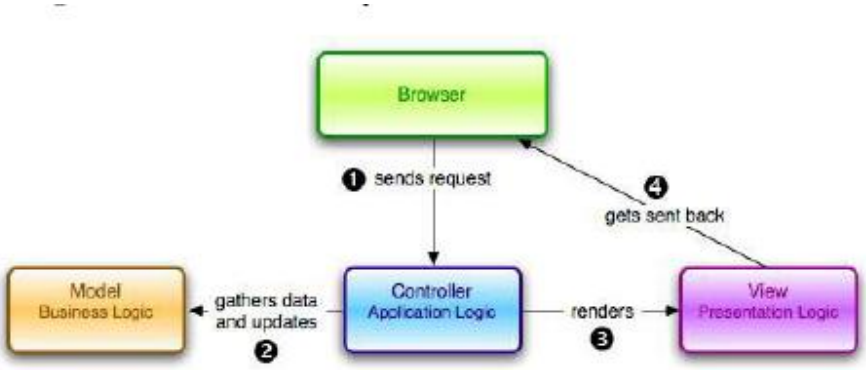

Figura 3: Representação simplificada do modelo MVC

5.2 Decisões

Os autores deste trabalho foram tomando várias decisões importantes relacionadas com os vários aspectos da aplicação, desde as propriedades da base de dados até ao modo de desenvolvimento da aplicação em si. Essas decisões serão descritas nos próximos parágrafos.

De início, os autores experimentaram várias ferramentas tendo tentado desenvolver a aplicação, tal como foi mostrado numa das aulas de Projecto Integrado, em X-Forms. Infelizmente X-Forms não é universalmente suportado, obrigando a instalar software específico. Devido à falta de interpretadores completamente funcionais e à falta de documentação sobre o assunto, suscitando assim bastantes dúvidas sobre as eficiência e eficácia de X-Forms, os autores optaram por usar outra ferramenta.

Como indicada e descrita anteriormente, a *framework* escolhida para este trabalho foi o Ruby on Rails. A escolha desta *framework* deve-se a vários factores. Em primeiro lugar, um dos autores já tinha tido contacto com Ruby on Rails, é uma linguagem nunca antes estudada pelos autores tornando assim o desenvolvimento desta aplicação mais interessante, inserindo uma componente autodidacta no decorrer do desenvolvimento da aplicação. Esta ferramenta é também independente da plataforma que se está a usar e aplica o modelo Model-View-Controller, descrito no capítulo anterior. Por último, sendo Ruby uma linguagem de *scripting*, os autores sentiram-se incentivados, pois terão no futuro um módulo na UCE de Engenharia de Linguagens que trata o assunto de linguagens de *scripting*.

Em relação á base de dados, os autores decidiram-se pela utilização do motor de base de dados SQLITE 3 devido a três razões. A primeira deve se ao facto de este motor estar totalmente integrado com a versão de Ruby on Rails utilizada (versão 2.0.2). A segunda razão deve-se ao facto de este motor ser simples, compacto, eficiente, portátil, gratuito, pré-instalado em Mac OS X e facilmente configurável noutros sistemas operativos. O terceiro e último motivo tem a ver

com o facto de a base de dados ser auto-gerada pelo Ruby on Rails, facilitando assim a tarefa do programador, pois este deixa de precisar de ter conhecimentos profundos sobre SQL bastando ter certas ideias teóricas sobre o funcionamento de base de dados de forma a obter uma base de dados consistente.

No que toca á base de dados em si, foi decidido um esquema da BD[1] adaptável às necessidades perante os objectivos do projecto. Este esquema foi sofrendo alterações durante todo o desenvolvimento e investigação do projecto. Por esse motivo, a base de dados apresentada durante as aulas não é a mesma apresentada neste relatório. Foi decidido o uso do campo "id"para criar chaves, pois desta forma foi mais fácil interagir com a nossa *framework*. Na base de dados, o nome da chave "id"escolhido é usado em todas as tabelas de forma a controlar facilmente toda a informação presente, evitando dificuldades e confusões desnecessárias. Apesar de existirem muitos campos, todos candidatos a chave primária, e devido á necessidade óbvia de serem únicos numa determinada tabela, optou-se pela introdução constante da chave primária "id", e assim garante-se a impossibilidade de inserir uma nova linha na base da dados onde os campos são obrigatoriamente diferentes, como por exemplo inserir na tabela "users"dois emails ou dois nomes de utilizador repetidos. Estão também presentes em anexo o Dicionário de Dados e a Migração da Base de Dados em Ruby on Rails.

5.3 Aplicação Final

Começando pelo acesso á aplicação, esta encontra-se online e pode ser encontrada em www.e-archia.net. Ao aceder á página principal, o utilizador vai encontrar um interface de boas vindas user-friendly onde pode realizar o login ou o registo. Em cima existe um menu com links para a pesquisa e feeds.

O login apenas autentifica o utilizador comparando o nome do utilizador e password inseridos com os existentes na base de dados.

O registo funciona do seguinte modo: após se fornecer ao dados pessoais, os dados de username e email são comparados com os existentes. Caso não sejam repetidos, o registo é aceite fazendo uma nova inserção na base de dados. A password guardada é cifrada usando sha-2. É criado um salt(footnote: salt é usado para encriptação de modo a que uma password nunca seja texto corrente, protegendo assim o utilizador) por utilizador sendo combinada com o hash da password para motivos de autenticação futura.

De seguida é mostrado o código para a criação de hashes

```
def before_save
   unless password.blank?
     salt = [Array.new(6){rand(256).chr}.join].pack("m").chomp
     self.password_salt, self.password_hash =
     salt, Digest::SHA256.hexdigest(password + salt)
   end
```

[1]Presente em Anexo

```
end
def self.authenticate(username, password)
  user = User.find(:first, :conditions => ['username = ?', username])
  if user.blank?
    raise "O utilizador não existe"
  elsif !user.enabled
    raise "O administrador optou por congelar o teu perfil"
  elsif Digest::SHA256.hexdigest(password + user.password_salt) != user.password_hash
    raise "A autenticação falhou"
  else
    user.update_attribute(:last_login_at, Time.now)
  end
  user
end
```

A aplicação também usa *partials* em grande quantidade. *Partials* podem ser explicadas como sendo pequenas partes de HTML dinâmico, que permite fazer render repetidamente em diferentes páginas. Por exemplo, um certo menu, que é repetido em todas as páginas, se for inserido num *partial*, é executado um reder desse *partial* deixando de haver duplicação de código.

Outro ponto importante da aplicação desenvolvida é que esta assenta em *RESTfull routes*. Isto significa que as routes permitem parâmetros, sendo assim inteligentes e sabem para onde redireccionar os diferentes pedidos. O próximo bloco de código é referente ao ficheiro routes.rb.

```
map.utilizador  '/utilizador/:username',
:controller => 'users',
:action => 'show_by_username'
```

Ao realizar uma pesquisa são devolvidos todos os resultados compatíveis com essa pesquisa, como seria de esperar. Após serem mostrado os resultados é possível fazer consulta e download dos ficheiros encontrados. É permitida a pesquisa a utilizadores não registados, mas estes apenas têm acesso aos artigos disponíveis e informação de utilizadores registados.

Outro modo de aceder aos ficheiros é aceder directamente ao trabalho desejado e consultar as respectivas entradas. Também é possível ao utilizador aceder aos seus ficheiros acendendo ao seu perfil e consultar os trabalhos/ficheiros submetidos.

O utilizador tem a opção de editar as suas opções bem como os seus próprios ficheiros.

6 A Aplicação Desenvolvida - Segundo Semestre

Após uma re-estruturação do grupo inicial, os autores deste projecto decidiram recomeçar de novo o projecto aproveitando alguns pontos já desenvolvidos no semestre anterior. Optou-se por largar o Ruby on Rails e se passar a usar o muito conhecido PHP. No seguimento deste capítulo irão ser descritas todos os desenvolvimentos de cada Questão proposta.

6.1 Decisões

Os autores deste trabalho foram tomando várias decisões importantes relacionadas com os vários aspectos da aplicação, desde as propriedades da base de dados até ao modo de desenvolvimento da aplicação em si. Devido à alteração ocorrida no grupo, e como referido anteriormente, várias decisões e pontos descritos no Relatório do Primeiro Semestre tornaram-se obsoletas. Essas decisões, tal como as novas, serão descritas nos próximos parágrafos.

6.2 Segundo Semestre

6.2.1 Disseminação

Como referenciado no capítulo *Enunciado do Trabalho*, neste momento teremos de desenvolver o sistema de DIPs (*Dessimination Information Packages*). Para este sistema funcionar necessitaremos de desenvolver um método que permita ao utilizador escolher os vários ficheiros que deseja obter e garantir que esses mesmos ficheiros cheguem ao utilizador de acordo com o pedido. O utilizador terá então a oportunidade de escolher se deseja obter todo o trabalho que se encontra presente na OAIS, ou se de esse mesmo trabalho apenas deseja certos ficheiros. Para que isto aconteça, o utilizador poderá navegar interactivamente dentro das pastas do trabalho, seleccionando os ficheiros que deseja receber. Quando todos os ficheiros forem seleccionados, o nosso sistema irá consultar o respectivo AIP e criar um novo DIP. Este DIP será um típico ficheiro zip contendo os ficheiros ou pastas escolhidos previamente pelo utilizador. Este ficheiro é depois disponibilizado ao utilizador através de um link.

6.2.2 Information Retrieval

Requisitos

Para uma execução bem sucedida desta fase do projecto alguns requisitos teriam de ser implementados, de modo a que fosse possível desenvolver um sistema de pesquisa textual dos ficheiros. De entre esses requisitos encontram-se a resolução de *bugs* que impediam a inserção de pacotes completos, a criação do navegador de árvore de ficheiros e criar uma indexação e pesquisa de palavras em ficheiros do tipo textual (código, notas, etc.) e do tipo pdf.

Ainda da primeira questão, o navegador de ficheiros foi implementado sendo de simples navegação.

Para a indexação de palavras, foi necessário incluir novas funções de modo a que, ao inserir um documento no sistema, este navegasse dentro desse ficheiro e acrescentasse á base de dados as palavras aí contidas. Assim, é necessário, como primeiro passo, realizar um teste de tipo a cada documento submetido, existindo três tipos gerais: texto, pdf e outros. Para cada um dos casos existem diferente formas de efectuar a análise que se listam de seguida:

- Caso seja um ficheiro de texto, as *strings* são partidas em palavras, sendo contado o número de ocorrências destas. É feita a inserção na base de dados as várias relações. À base de dados foi acrescentado novos campos que irão ser mostrados mais á frente;
- No caso de ficheiros PDF, este é convertido num ficheiro de texto usando a função auxiliar *ps2txt*. Após a conversão, os passo descritos no ponto anterior são realizados sobre o novo ficheiro resultante da conversão;
- Se o ficheiro não for textual nem PDF, por exemplo ficheiros de Excel ou PowerPoint, não é realizada qualquer análise.

Para que seja mais explicita a inserção na base de dados, explicaremos agora como é feita essa inserção. De um modo mais abstracto, se a palavra ainda não existe na base de dados, a palavra é inserida, e a relação dessa palavra com o ficheiro é inserido. Caso contrário simplesmente constrói e insere a nova relação entre a palavra e o ficheiro na base de dados.

Tal como referido anteriormente, aqui se encontra o esquema desta parte da base de dados:

Pesquisa de Conteúdo - IR

Nos parágrafos anteriores falou-se dos requisitos necessários para se poder implementar a pesquisa a nível textual na nossa aplicação. Agora iremos falar na pesquisa propriamente dita.

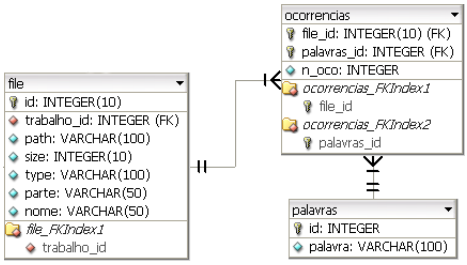

Figura 4: Esquema da Base de Dados para o processamento de palavras

Ao pesquisar por uma ou várias palavras, é feita uma consulta na base de dados sobre cada uma das palavras. Para cada palavra a ser pesquisada é criado um *array* especifico. À medida que são encontrados ficheiros contendo cada uma dessas palavras, esse ficheiros são colocados no respectivo *array*. No fim da pesquisa é feita a intersecção dos vários *arrays*, devolvendo assim os ficheiros que contêm a combinação das várias palavras pesquisadas.

WebServices

De forma a tornar o repositório um "colaborante"com os outros, criou-se um servidor de WebServices que responde a qualquer cliente, via SOAP, a pedidos remotos de pesquisa no repositório. Foi criado também um cliente para fazer pedidos a outros repositórios, tornando assim as pesquisas mais eficientes e abrangentes.

7 Conclusão

Neste último capítulo do relatório os autores discutirão os resultados finais obtidos, dando também algumas observações e sugestões relacionadas com a aplicação desenvolvida.

Os objectivos dos requisitos impostos foram integralmente cumpridos. Foi implementado com sucesso a pesquisa de ficheiros e artigos na base de dados, o registo e login de utilizadores, o envio de ficheiros para a base de dados segundo a identificação do utilizador, a edição de informação, ficheiros e utilizadores. É possível aceder aos ficheiros guardados. Também é possível fazer pesquisas a nível textual sobre os vários ficheiros guardados na base de dados, assim como aceder a resultados de pesquisas via WebServices.

Em relação à pesquisa textual, esta apresenta-se como uma pesquisa relativamente eficaz e adaptável aos vários tipos de ficheiros. Existindo pequenos pormenores a serem melhorados no futuro, pois este método obriga a muitos acessos à base de dados durante a inserção, e no caso de se tratar da inserção de um ficheiro com um leque de palavras altamente variado, torna a inserção de um pacote lenta. Um último ponto negativo é o facto de ser necessário o uso de várias funções auxiliares para interpretar ficheiros PDF com texto acentuado ou cedilhado.

Dá-se assim por terminado o relatório sobre ADTP para o Projecto Integrado da UCE de Engenharia de Linguagens, o qual os autores consideram ter obtido sucesso na sua execução.

Referências

[1] Brian Lavoie
OCLC Newsletter, No. 243:26-30
http://www.oclc.org/research/publications/archive/2000/lavoie/
(Janeiro/Fevereiro 2000)

[2] UCE - Engenharia das Linguagens
Página do Projecto Integrado
http://twiki.di.uminho.pt/twiki/bin/view/Education/EL/Projecto

[3] Wikipedia - página sobre Ruby on Rails
http://en.wikipedia.org/wiki/Ruby_on_Rails

[4] Wikipedia - página sobre OAIS
http://en.wikipedia.org/wiki/OAIS

[5] Reference Model for an Open Archival Information System (OAIS) CCSDS 650.0-B-1, Blue Book
(Janeiro 2002)

[6] Patrick Lenz
Build Your Own Ruby On Rails Web Applications
(Março 2007)

A Esquema da Base de dados