Projeto Final

Processamento Estruturado de Documentos

UCE30 Engenharia de Linguagens

Mestrado em Engenharia Informática

AAA

Arquivo Aberto de trabalhos de Alunos

Autores:

Bruno Azevedo (azevedo.252@agmail.com)

Miguel Costa (miguelpintodacosta@gmail.com)

Supervisor:

José Carlos Ramalho (jcr@di.uminho.pt)

23/01/2012

Índice

Introdução 1
Ambiente de Trabalho 2
Linguagens de Programação e Tecnologias 2
Suporte 2
Editor de Texto/IDE 2
Análise e Especificação 3
Descrição do Problema 3
Especificação dos Requisitos 3
SIP e o Processo de Ingestão 4
AIP e armazenamento de projetos 4
DIP e a disseminação/publicação de conteúdos 4
Conceção da Resolução 5
Base de Dados 5
Contextualização 5
Tabelas 5
Validação da estrutura do modelo utilizando normalização 12
Estrutura do Website 13
Descrição das Categorias 15
Paginação do conteúdo 16
Project Record 16
Conteúdo 16
Autores e Supervisores 18
Decisões Tomadas e Funcionalidades 18
Processo de Ingestão 18
Assistente de criação de um SIP 19
Ficheiro ZIP 21
Processo de Disseminação 21
De um projeto 21
Do repositório 21

Inserir/alterar/remover utilizadores 22
Inserir/alterar/remover autores e supervisores 22
Logs 22
O que é registado? 22
Como é registado? 23
Como é exibido? 24
Estatísticas 25
Estatísticas geradas 25
Acessos 25
Depósitos 26
Downloads 26
Consultas 26
Tecnologia usada 27
Pesquisa 28
Tecnologias 28
Lucene 28
Zend Search Lucene 29
Implementação 29
Documentos 29
Outras Decisões 30
Web Service 31
Servidor 31
Cliente 32
Dificuldades encontradas 33
Conclusão 33

Introdução

A Internet é uma rede imensa de máquinas ligadas em rede, a transferência de dados tem tamanhos astronómicos e cada vez mais é necessário desenvolver aplicações que consigam comunicar com outras máquinas sozinhas. Devido a esta necessidade de comunicação utiliza-se linguagens de marcação como o XML.

O XML tornou-se o formato standard para a partilha de informação, grande parte das aplicações desenvolvidas atualmente suportam este formato.

Um problema, que nos deparamos no meio académico, é o facto de trabalhos realizados por alunos serem esquecidos depois de avaliados. Alguns deles podiam servir de referência para futuros trabalhos, no entanto não há neste momento nenhuma aplicação que este alunos usem para submeter os seus trabalhos e torna-los públicos. Surge então a necessidade de criar um repositório que o permita fazer.

Com a necessidade de criar um repositório, é praticamente obrigatório utilizar tecnologias que permitam facilmente outros sistemas conseguirem tratar a informação contida nele. É devido a esta necessidade que utilizamos o XML e tecnologias associadas a este para o desenvolvimento de todo o nosso sistema, desta forma sabemos que, em princípio, não vamos ter grandes problemas no futuro para alterar o formato da documentação guardada.

Neste relatório vamos explicar todo o processo de desenvolvimento deste sistema, desde a análise de requisitos até à sua implementação.

Ambiente de Trabalho

Linguagens de Programação e Tecnologias

A principal linguagem de programação utilizada foi o PHP, usada para gerar todas as páginas HTML e ainda para inserir toda a informação na base de dados e logs.

Além do PHP e HTML, usámos ainda CSS, JavaScript e outras tecnologias associadas ao Ajax. Esta última serviu essencialmente para facilitar e tornar mais atrativa a interação do utilizador com o sistema, por exemplo para a criação de páginas para listar conteúdos.

Usámos ainda a linguagem de marcação XML e várias tecnologias a ela associadas, como: XPath, XML Schema, XML Stylesheet. Para todas as transformações XSLT necessárias foi usada a biblioteca DOM do PHP.

Suporte

Linux foi o sistema operativo usado para suportar o sistema desenvolvido. Para a Base de Dados adotámos o MySql e revelou-se mais do que suficiente para o que era exigido.

Instalámos nas nossas máquinas o xampp, ferramenta que incluiu um servidor PHP e MySql para conseguirmos testar o que íamos fazendo.

Para a criação de Web Services, usámos a ferramenta open-source NuSOAP e para a pesquisa no Website o Zend Lucene.

Editor de Texto/IDE

Para desenvolver um projeto com esta dimensão e com as tecnologias que foram usadas, foi essencial recorrer a ferramentas que nos facilitassem algumas tarefas.

Para a escrita da linguagem PHP e HTML usámos o Vim e o IDE Netbeans, para a criação de ficheiros XML, Stylesheets (XSLT) e XML Schema usámos Oxygen XML, que foi importante para testar os ficheiros criados antes de os aplicar no nosso sistema.

Análise e Especificação

Descrição do Problema

No mundo académico há uma enorme quantidade de trabalhos que são produzidos, muitos deles de boa qualidade que podiam ser úteis para outras pessoas aprenderem, tirarem ideias, terem como suporte noutro trabalho... no entanto, depois de avaliados são esquecidos.

Para resolver este problema criámos um repositório digital, com uma série de funcionalidades, que permite armazenar projetos/trabalhos desenvolvidos por alunos e que seja possível consultar e exportar toda a informação que tenha sido submetida.

Especificação dos Requisitos

O repositório teve de ser desenvolvido tendo em conta o modelo OAIS (Open Archival Information System).

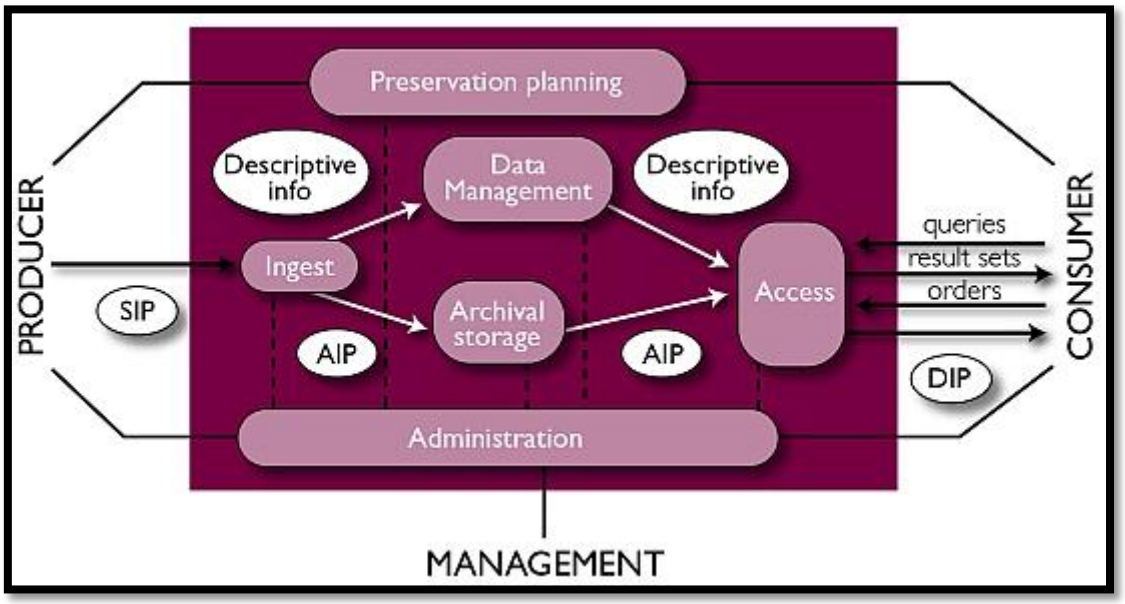


Figura 1 Modelo OAIS

O modelo OAIS possui três tipos de pacotes de informação no sistema:

- SIP (Submission Information Package) - pacote com informação para ser submetida no sistema com uma estrutura pré-definida.
- AIP (Archival Information Package) - pacote que é arquivado, pode ser armazenado, por exemplo, numa Base de Dados ou num ficheiro XML. É um pacote criado depois da ingestão de um SIP.
- DIP (Dissemination Information Package) - pacote que permite obter o conteúdo do repositório, seja num Website ou num ficheiro zip.

Este modelo tem de permitir três tipos de atores:

- Administrador - realiza todas as tarefas de manutenção do sistema.
- Consumidor - utilizador que consulta e pesquisa a informação do repositório.
- Produtor - corresponde aos utilizadores que irão submeter informação no repositório.

Quanto às funcionalidades, o sistema permite três grandes processos:

- Administração - gestão de utilizadores, gestão dos arquivos armazenados...
- Ingestão - processo que permite ingerir conteúdo no sistema.
- Disseminação - processo que permite ver o conteúdo armazenado. Pode ser através de uma página web ou da criação de um ficheiro zip.

SIP e o Processo de Ingestão

O requisito para a submissão de um SIP é que seja possível submeter um ficheiro zip com o conteúdo para guardar no repositório. Esse ficheiro zip contém um manifesto, designado por *Project Record* (PR), com o nome de "pr.xml", este ficheiro tem uma estrutura definida por um XML Schema criado nas aulas. Além do manifesto, contém ainda outros ficheiros que se pretende juntar ao projeto.

O processo de ingestão tem de garantir que a informação está correta, por isso temos de validar as seguintes condições:

- Verificar se existe um ficheiro chamado "pr.xml";
- Verificar se todos os ficheiros que estão mencionados no manifesto estão presentes no zip.

Depois de passar pelas validações anteriores, a metainformação do SIP vai ser armazenada numa Base de Dados relacional e os seus ficheiros armazenados num Sistema de Ficheiros criado para o efeito. A partir deste momento passamos a ter um DIP.

AIP e armazenamento de projetos

Neste repositório, vamos ter uma solução híbrida, com a metainformação guardada na Base de Dados relacional, ou seja, a informação que está no *Project Record* e a outra informação que corresponde a outros ficheiros incluídos no SIP guardados no Sistema de Ficheiros.

DIP e a disseminação/publicação de conteúdos

Um DIP pode ser disponibilizado de duas formas, ou através de um Website criado para o efeito ou então através de um ficheiro ZIP.

No Website será possível listar os projetos, ver o conteúdo de um deles e ver os ficheiros anexados.

Pode ser possível ainda exportar um projeto para um ficheiro zip que terá uma estrutura idêntica ao SIP que foi submetido.

Conceção da Resolução

Base de Dados

Neste capítulo, iremos apresentar uma contextualização sobre a necessidade desta base de dados e a nossa solução final de base de dados.

Para isso enunciaremos as tabelas criadas e explicaremos a sua razão de ser e mostraremos a nossa solução relativamente à necessidade de produzir determinadas estatísticas sobre os dados do repositório. Por fim, validaremos o nosso modelo através de técnicas de normalização.

Contextualização

Para poder guardar a metainformação relativa aos projetos, foi decido que esta seria armazenada numa Base de Dados relacional para que fosse de fácil acesso e manutenção. Durante as aulas foram desenvolvidos um Modelo Entidade-Relacionamento e um esquema de Base de Dados Relacional. Ver Anexos I.

Com base nisto, e após alguma análise mais aprofundada, chegámos a um modelo final que passaremos a descrever na secção seguinte.

Tabelas

As chaves primárias são representadas por atributos sublinhados e as chaves estrangeiras por atributos em itálico.

- **Project** {projcode, keyname, title, subtitle, bdate, edate, subdate, abstract, *coursecode*, path, remove, private}

Esta é a tabela principal, que armazena a maioria da metainformação relativa a um projeto.

Atributos:

- **projcode**: é chave primária, já que identifica univocamente cada projeto. Inteiro que é auto incrementado;
- **keyname**: string que identifica um projeto;
- **title**: string que identifica o título do projeto;
- **subtitle**: string que identifica o subtítulo do projeto. Pode ser nulo;

- **bdate**: data que identifica a data de início do projeto;
- **edate**: data que identifica a data de fim do projeto;
- **subdate**: data que identifica a data em que o projeto foi submetido no repositório;
- **abstract**: string que é um resumo do que foi realizado no projeto;
- **coursecode**: chave estrangeira da tabela Course. Inteiro que identifica o curso ao qual o projeto está associado;
- **path**: string que representa o caminho no sistema de ficheiros onde se encontram todos os ficheiros que foram submetidos em anexo com o projeto e também aqueles que foram gerados automaticamente;
- **remove**: inteiro que indica se o projeto foi removido (1) ou não (0) do repositório;
- **private**: inteiro que indica se o projeto é privado (1) ou público (0).

- Author {authorcode, name, id, email, url, *coursecode*, remove}
  Esta tabela armazena a informação relativa aos autores de projetos.
  Atributos:
  - **authorcode**: é chave primária, já que identifica univocamente cada autor. Inteiro que é auto incrementado;
  - **name**: string que identifica o nome do autor;
  - **id**: string que é o identificador do autor;
  - **email**: string que identifica o email do autor;
  - **url**: string que identifica o url de um Website referente ao autor. Pode ser nulo;
  - **coursecode**: chave estrangeira da tabela Course. Inteiro que identifica o curso ao qual o autor está associado;
  - **remove**: inteiro que indica se o autor foi removido (1) ou não (0) do repositório.

- **Supervisor** {supcode, name, email, url, affil, remove}
  Esta tabela armazena a informação relativa aos supervisores dos projetos.
  Atributos:
  - **supcode**: é chave primária, já que identifica univocamente cada supervisor. Inteiro que é auto incrementado;
  - **name**: string que identifica o nome do supervisor;
  - **email**: string que identifica o email do supervisor;
  - **url**: string que identifica o url de um Website referente ao supervisor. Pode ser nulo;
  - **affil**: string que identifica a que instituição está afiliado. Pode ser nulo;

- **remove**: inteiro que indica se o supervisor foi removido (1) ou não (0) do repositório.

- **KeyWord** {kwcode, keyword}

  Esta tabela armazena todas as keywords que foram identificadas em todos os projetos submetidos no repositório.

  Atributos:

  - **kwcode**: é chave primária, já que identifica univocamente cada keyword. Inteiro que é auto incrementado;
  - **keyword**: string que identifica a keyword;

- **Deliverable** {delcode, description, path, *projcode*}

  Esta tabela armazena todos os ficheiros anexados aos projetos.

  Atributos:

  - **delcode**: é chave primária, já que identifica univocamente cada deliverable. Inteiro que é auto incrementado;
  - **description**: string que descreve o deliverable;
  - **path**: string que representa o caminho no sistema de ficheiros onde se encontra o deliverable;
  - **projcode**: é chave estrangeira da tabela Project. Inteiro que identifica o projeto ao qual o deliverable pertence.

- **Course** {coursecode, coursedescription}

  Esta tabela armazena todos os cursos de todos os possíveis projetos.

  Atributos:

  - **coursecode**: é chave primária, já que identifica univocamente cada curso. Inteiro que é auto incrementado;
  - **description**: string que identifica o nome do curso.

As 3 tabelas seguintes derivam de relacionamentos N:M:

- **ProjAut** {*projcode*, *authorcode*}

  Esta tabela armazena os relacionamentos projeto tem autor.

  Atributos:

  - **projcode**: é chave estrangeira da tabela Project;
  - **authorcode**: é chave estrangeira da tabela Author;

  A chave primária é composta pelos dois atributos.

- **ProjSup** {*projcode*, *supcode*}

  Esta tabela armazena os relacionamentos projeto tem supervisor.

  Atributos:

- **projcode**: é chave estrangeira da tabela Project;
- **supcode**: é chave estrangeira da tabela Supervisor;

A chave primária é composta pelos dois atributos.

- **ProjKW** {*projcode*, *kwcode*}

  Esta tabela armazena os relacionamentos projeto tem keyword.

  Atributos:
  - **projcode**: é chave estrangeira da tabela Project;
  - **kwcode**: é chave estrangeira da tabela KeyWord;

  A chave primária é composta pelos dois atributos.

Para além da informação referente a um projeto, sendo isto um repositório digital de projetos onde existem utilizadores que os submetem, é necessário armazenar a informação sobre os mesmos. Por isso, criámos a tabela Users.

- **Users** {username, name, password, email, url, affil, type, remove}

  Esta tabela armazena a informação relativa aos utilizadores do sistema. Estes últimos podem ser de três tipos mencionados anteriormente Produtor, Administrador, Consumidor, ver Especificação dos Requisitos.

  Atributos:
  - **username**: é chave primária, já que identifica univocamente cada utilizador. String que identifica o username do utilizador;
  - **name**: string que identifica o nome do utilizador;
  - **password**: string que identifica a password de entrada no sistema do utilizador;
  - **email**: string que identifica o email do utilizador;
  - **url**: string que identifica o url de um Website referente ao utilizador. Pode ser nulo;
  - **affil**: string que identifica a que instituição o utilizador está afiliado. Pode ser nulo;
  - **type**: string que identifica o tipo de utilizador; ‘p’=Produtor, ‘a’=Administrador, ‘c’=Consumidor;
  - **remove**: inteiro que indica se o utilizador foi removido (1) ou não (0) do repositório.

A necessidade de gerar certas estatísticas relativas aos projetos e aos acessos ao repositório levou-nos a criar 4 tabelas adicionais para manter registo de algumas atividades no repositório:

- **Access** {*username*, datahora}

Esta tabela mantém registo de todos os acessos ao repositório (login) feitos pelos utilizadores, registando a data e hora.

Atributos:

- **username**: é chave estrangeira da tabela Users;
- **datahora**: timestamp que identifica a hora a que utilizador fez login;

A chave primária é composta pelos dois atributos.

- **Deposits** {*username*, *projcode*}

  Esta tabela mantém registo de todos os projetos submetidos pelos utilizadores.

  Atributos:

  - **username**: é chave estrangeira da tabela Users;
  - **projcode**: é chave estrangeira da tabela Project;

  A chave primária é composta pelos dois atributos.

- **Downloads** {*username*, *projcode*, datahora}

  Esta tabela mantém registo de todos os downloads feitos por utilizadores a projetos, registando também a data e a hora do download.

  Atributos:

  - **username**: é chave estrangeira da tabela Users;
  - **projcode**: é chave estrangeira da tabela Project;
  - **datahora**: timestamp que identifica a hora o que utilizador fez download;

  A chave primária é composta pelos três atributos.

- **Queries** {id, *username*, *projcode*, *authorcode*, *supcode*, datahora}

  Esta tabela mantém registo de todas as consultas feitas pelos utilizadores a projetos, autores e supervisores, registando também a data e a hora da consulta.

  Atributos:

  - **id**: é chave primária porque identifica univocamente a consulta e tanto o projcode, authorcode e supcode podem ser nulos, logo não podem pertencer à chave primária. Inteiro que é auto incrementado;
  - **username**: é chave estrangeira da tabela Users;
  - **projcode**: é chave estrangeira da tabela Project. Pode ser nulo;
  - **authorcode**: é chave estrangeira da tabela Author. Pode ser nulo;
  - **supcode**: é chave estrangeira da tabela Supervisor. Pode ser nulo;
  - **datahora**: timestamp que identifica a hora o que utilizador fez a consulta;

Tendo sido criadas as tabelas necessárias ao armazenamento destas atividades, com vista a gerar as estatísticas, optámos por materializar algumas views para facilitar o cálculo dessas estatísticas.

Para calcular as estatísticas relativas aos acessos, as seguintes views foram materializadas:

- **ViewNumAccessData**

  View que contabiliza os acessos por cada mês de cada ano.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select month(`Access`.`datahora`) AS `month`,year(`Access`.`datahora`) AS
`year`,count(0) AS `accesses`
from `Access`
group by month(`Access`.`datahora`),year(`Access`.`datahora`)
order by year(`Access`.`datahora`),month(`Access`.`datahora`)
```

- **ViewNumAccessUser**

  View que contabiliza os acessos por utilizador.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select `Access`.`username` AS `username`,count(0) AS `accesses`
from `Access`
group by `Access`.`username`
order by count(0) desc
```

- **ViewNumAccessTotal**

  View que contabiliza o número total de acessos.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select count(`Access`.`username`) AS `accesses`  from `Access`
```

Para as estatísticas relativas aos depósitos, as seguintes views foram materializadas:

- **ViewNumDepositsDate**

  View que contabiliza os depósitos por cada mês de cada ano.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select month(`P`.`subdate`) AS `month`,year(`P`.`subdate`) AS `year`,
count(`Deposits`.`projcode`) AS `deposits`
from (`Project` `P` join `Deposits`)
where (`Deposits`.`projcode` = `P`.`projcode`)
group by month(`P`.`subdate`),year(`P`.`subdate`)
order by year(`P`.`subdate`),month(`P`.`subdate`)
```

- **ViewNumDepositsUser**

  View que contabiliza os depósitos por utilizador.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select `Deposits`.`username` AS `username`,count(`Deposits`.`projcode`) AS
`depositos`
from `Deposits`
```

```
group by `Deposits`.`username`
order by count(`Deposits`.`projcode`) desc
```

- **ViewNumDepositsCourse**

  View que contabiliza os depósitos por curso.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select `P`.`coursecode` AS `curso`,count(`D`.`projcode`) AS `depositos`
from (`Project` `P` join `Deposits` `D`)
where (`P`.`projcode` = `D`.`projcode`)
group by `P`.`coursecode`
order by count(`D`.`projcode`) desc
```

- **ViewNumDepositsTotal**

  View que contabiliza o número total de depósitos.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select count(1) AS `numProjsTotal` from `Deposits`
```

As estatísticas referentes aos downloads são calculadas através das seguintes views:

- **ViewTopDownloads**

  View que calcula o top 10 de downloads feitos a projetos.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
"select `Downloads`.`projcode` AS `projcode`, count(`Downloads`.`projcode`) AS
`downloads`
from `Downloads`
group by `Downloads`.`projcode`
order by count(`Downloads`.`projcode`) desc
limit 0,10"
```

- **ViewNumDownloadsTotal**

  View que contabiliza o número total de downloads realizados.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select count(`Downloads`.`projcode`) AS `numDownloads` from `Downloads`
```

Finalmente, as estatísticas relativas às consultas são calculadas através das seguintes views:

- **ViewTopQueriesProj**

  View que calcula o top 10 de consultas feitas a projetos.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select `Q`.`projcode` AS `projcode`, count(1) AS `queries`
from `Queries` `Q`
where ((`Q`.`projcode` is not null) and isnull(`Q`.`authorcode`) and
isnull(`Q`.`supcode`))
group by `Q`.`projcode`
order by count(1) desc
limit 0,10"
```

- **ViewTopQueriesAuthor**

  View que calcula o top 10 de consultas feitas a autores.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select `Queries`.`authorcode` AS `authorcode`, count(1) AS `queries`
from `Queries`
where (isnull(`Queries`.`projcode`) and (`Queries`.`authorcode` is not null)
and isnull(`Queries`.`supcode`))
group by `Queries`.`authorcode`
order by count(1) desc
limit 0,10
```

- **ViewTopQueriesSup**

  View que calcula o top 10 de consultas feitas a supervisores.

  A seguinte instrução SQL foi utilizada para gerar o conteúdo da View:

```
select `Queries`.`supcode` AS `supcode`, count(1) AS `queries`
from `Queries`
where (isnull(`Queries`.`projcode`) and isnull(`Queries`.`authorcode`) and
(`Queries`.`supcode` is not null))
group by `Queries`.`supcode`
order by count(1) desc
limit 0,10
```

Validação da estrutura do modelo utilizando normalização

Após a construção do modelo é necessário validá-lo para garantir que este representa os requisitos de dados estabelecidos. Para isso, utilizaremos a normalização, uma técnica que produz um conjunto de tabelas adequadas que representam os requisitos de dados que um determinado cliente fornece. Estas tabelas devem conter um número mínimo de atributos necessários para representar os requisitos de dados, atributos estes que, entre si, devem conter um relacionamento restrito (dependência funcional). Para além disso devem conter o mínimo de redundância possível, ou seja, cada atributo é representado apenas uma vez, à exceção das chaves estrangeiras.

Tendo um conjunto de tabelas normalizadas, tornar-se-á mais fácil o acesso e a manutenção dos dados e o espaço necessário para armazená-los será mínimo.

De seguida, os passos mais importantes para a normalização do modelo lógico construído serão mostrados e provar-se-á a sua validade.

1º Forma Normal (1FN)

O objetivo é transformar uma tabela que está desnormalizada, ou seja, uma tabela que contém um ou mais grupos de dados repetidos (multi-valor), numa tabela na 1FN, que é uma tabela cujas intersecções de cada linha com uma coluna contém apenas um valor.

No nosso caso, o modelo não possui nenhum atributo multi-valor, mas caso possuísse, esse problema seria resolvido criando uma tabela adicional para esse atributo e adicionando

uma chave estrangeira da tabela original. Assim, garantimos que o nosso modelo está na 1ª Forma Normal.

2ª Forma Normal (2FN)

Para uma tabela estar na 2FN, precisa de estar na 1FN e todos os atributos não chave primária devem depender da totalidade da chave primária e não apenas de parte.

A 2FN aplica-se a tabelas com chaves primárias compostas, ou seja, chaves compostas por dois ou mais atributos. Deste modo, uma tabela com uma chave primária simples já se encontra na 2FN.

Analisando as tabelas com chaves primárias compostas, chegamos à conclusão que nenhuma tabela possui uma dependência funcional parcial. Garantimos, então, que o nosso modelo está na 2ª Forma Normal.

3ª Forma Normal (3FN)

Para uma tabela estar na 3FN, precisa de estar na 1FN e 2FN e não pode conter dependências funcionais entre atributos que não são chave primária.

Analisando as tabelas, concluímos que o nosso modelo está na 3ª Forma Normal.

Desta forma, asseguramos que o nosso modelo de dados está normalizado.

Estrutura do Website

Começando pelo interface, a estrutura do Website é composta por três blocos principais:

1. Cabeçalho e Menu de Categorias;
2. Menu da Esquerda (opções disponíveis numa dada categoria);
3. Corpo (onde aparece a informação propriamente dita).

O cabeçalho, para além dos títulos contém ainda a área de login (à direita). O menu de categorias possui do lado direito uma caixa de procura.

As Categorias que existem são:

- Início
- Submeter
- Repositório
- Estatísticas
- Logs
- Web Service
- Acerca

Dependendo do tipo de utilizador que estiver a utilizar o Website, as Categorias e as Opções disponíveis são alteradas, tendo apenas o administrador acesso a todo o tipo de tarefas.

A tabela mostra quem tem acesso às Categorias:

Categoria	Desconhecido	Consumidor	Produtor	Administrador
Início	X	X	X	X
Submeter			X	X
Repositório	X	X	X	X
Estatísticas	X	X	X	X
Logs				X
Web Service	X	X	X	X
Acerca	X	X	X	X

Dentro de cada categoria, nem todos tem as mesmas opções, a tabela seguinte mostra as opções disponíveis para cada utilizador:

Categoria	Opção	Desconhecido	Consumidor	Produtor	Administrador
Início		X	X	X	X
Submeter	Preencher Formulário			X	X
	Pacote ZIP			X	X
Repositório	**Utilizadores**				X
	Listar				X
	Inserir				X
	Alterar				X
	Remover				X
	Autores e Supervisores	X	X	X	X

	Listar	X	X	X	X
	Inserir				X
	Alterar				X
	Remover				X
	Submissões	X	X	X	X
	Listar	X	X	X	X
	Remover				X
	Exportar para Zip				X
Estatísticas	**Todas**	X	X	X	X
Logs					X
Web Service	**Pedir Projeto**	X	X	X	X
	Projetos com Keywords	X	X	X	X
Acerca		X	X	X	X

Descrição das Categorias

- Início

  Esta é a categoria que é apresentada quando alguém acede ao Website, funciona como página de apresentação do repositório.

- Submeter

  Onde um Produtor pode submeter um SIP.

  - Preencher Formulário

    Funciona como assistente para a criação de um SIP para submeter no repositório.

  - Pacote ZIP

    Opção para submeter um SIP num ficheiro zip.

- Repositório

  Todas as opções relacionadas com o conteúdo do repositório.

  - Utilizadores

    Todas as opções relacionadas com os utilizadores que estão registados ou não no repositório.

    - Listar

      Lista todos os utilizadores que estão registados no sistema.

    - Inserir

      Permite inserir um novo utilizador, este pode ser um Administrador, um Produtor ou um Consumidor.

    - Alterar

Permite alterar a informação de um utilizador.

- Remover

  Remover um utilizador já registado.

- Estatísticas

  Contém as várias estatísticas que se podem consultar.

- Logs

  A maioria das ações sobre o repositório podem ser vistas aqui.

- Web Service

  Contém os serviços disponíveis.

  - Pedir Projeto

    Um utilizador insere o id (projcode) do projeto que pretende obter e recebe a resposta do servidor..

  - Projetos com Keywords

    O utilizador insere uma keyword ou lista de keywords que pretende e o recebe como resultado uma listagem dos projetos que contém uma das keywords.

Paginação do conteúdo

Para fazer uma paginação mais amigável decidimos utilizar a metodologia Ajax que utiliza tecnologias como javascript (a biblioteca jQuery), XML, DOM, entre outras.

Com uma script em javascript, utilizando a biblioteca jQuery, quando o evento "click" para mudar de página é disparado, uma requisição HTTP assíncrona (Ajax) é enviada ao servidor (load_data.php) passando como parâmetro o novo número de página, requisição essa que consiste no carregamento do conteúdo correspondente aos registos da nova página. Quando o servidor retornar, o novo conteúdo HTML é impresso para o browser.

Project Record

Os projetos que se podem submeter no repositório tem uma estrutura que foi definida previamente, por isso vamos agora explicar todos os componentes que fazem parte do objeto que é inserido.

Conteúdo

O *Project Record*, é um ficheiro com o nome "pr.xml" que está dentro do zip do pacote SIP, ou então é criado automaticamente pelo formulário existente no Website. Alguns dos campos são de preenchimento obrigatório, outros são facultativos.

Tem cinco áreas de conteúdo:

- Header

- Supervisors
- Workteam
- Abstract
- Deliverables

O **Header** contém a informação que identifica o projeto e contém os campos:

- **Keyname** - string que identifica o projeto, não precisa de ser único no sistema e é obrigatório.
- **Title** - título do projeto, é obrigatório.
- **Subtitle** - o subtítulo é opcional, fica ao critério do Produtor preencher ou não.
- **Begin Date** - campo obrigatório e representa a data de início do projeto.
- **End Date** - campo obrigatório e corresponde à data de término do projeto.

Os **supervisors** são as pessoas que acompanham um projeto ou lançam o enunciado. Um supervisor contém os campos:

- **Name** - nome do supervisor, é obrigatório.
- **Email** - é obrigatório e tem de ser único no sistema.
- **URL** - geralmente corresponde à página pessoal do supervisor, é opcional.
- **Affil** - identifica o departamento ao qual pertence, também é opcional.

A **Workteam** identifica as pessoas que realizaram o projeto, este é constituído por uma lista de Authors que contém:

- **Name** - nome do autor, é obrigatório.
- **ID** - corresponde geralmente ao número do aluno na instituição a que pertence, é obrigatório.
- **Email** - é obrigatório e tem de ser único no sistema.
- **Course** - curso ao qual o autor pertence, é obrigatório.

O **Abstract** é um campo composto por outros elementos, é utilizado para escrever um resumo sobre o projeto que vai ser guardado no repositório.

Este elemento pode conter outros elementos como:

- Parágrafos
- Keywords
- Palavras a negrito
- Palavras a itálico
- Palavras sublinhadas
- URL

A estrutura do Abstract pode ser vista com mais detalhe no XML Schema criado para o efeito que está em anexo.

As Keywords são os únicos elementos do abstract que são tratados, são guardados de forma a se conseguir relacionar projetos que partilhem as mesmas keywords.

Todos os ficheiros que forem anexados à submissão, são referenciados na área de **Deliverables**. Esta pode ter até seis ficheiros e cada um tem de ter:

- **Path**, local onde está o ficheiro. É obrigatório.
- **Description**, pequena descrição do ficheiro, geralmente coloca-se o nome que se quer que ele tenha. É obrigatório.

Autores e Supervisores

As entidades autor e supervisor presentes num Project Record têm de estar registadas no sistema, os seus registos são armazenados na Base de Dados relacional nas tabelas Author e Supervisor, respetivamente. Só os que estiverem previamente registados podem constar no SIP, caso contrário, não é possível efetuar a sua submissão.

Um autor tem de ter associado a si um curso e na base de dados existe a tabela Course que contém todos os cursos possíveis.
Para povoar essa tabela aplicámos os conhecimentos adquiridos durante as aulas da UCE30 Engenharia de Linguagens. Fomos ao site da Universidade do Minho onde existe uma listagem completa dos cursos existentes, passámos essa listagem para um ficheiro de texto. Depois de ter este ficheiro de texto, criámos uma script em Perl que nos gerou um ficheiro XML. Estando todos os cursos listados num ficheiro xml, bastou aplicar uma Stylesheet para gerar o código SQL das inserções dos cursos para a tabela Course. Desta forma conseguimos povoar a tabela com cerca de 200 cursos facilmente.

Decisões Tomadas e Funcionalidades

Processo de Ingestão

Um dos requisitos era que o sistema permitisse a submissão de um ficheiro ZIP com a estrutura necessária (já referida anteriormente) para ser um SIP. Além desta funcionalidade criámos ainda um assistente para a submissão do pacote SIP que consiste num formulário.

Nesta secção vamos explicar todo o mecanismo que foi usado para receber um SIP e passá-lo a AIP.

É importante referir que não é permitida a inserção de um projeto semelhante a um que já esteja armazenado, ou seja, ao tentar inserir um projeto em que o keyname, title, begin date, end date e abstract já existam no repositório, é rejeitada essa submissão.

Assistente de criação de um SIP

Apesar da tecnologia HTML5 ser muito recente, tendo em conta os browsers que usamos (Firefox e Google Chrome), aplicámos algumas das novidades que o HTML5 oferece para a criação de formulários. Por exemplo, para campos que eram obrigatórios usámos o atributo "*required*" e para controlar o texto que pode ser inserido usámos "*pattern*", que permite definir uma expressão regular que vai controlar o texto que pode ser inserido num elemento.

Passamos agora a explicar cada um dos elementos e o que pode ser colocado nele:

- Keyname

Este campo é obrigatório e para evitar futuros problemas, colocámos uma expressão regular para controlar os caracteres que podem ser inseridos:

```
<input id="key_name"
       name="key_name"
       type="text"
       required=""
       pattern="^[a-zA-ZáàãõéíóúçÁÀÃÕÉÍÓÚÇ][\wáàãõéíóúçÁÀÃÕÉÍÓÚÇ:_.-/|\s]*"
       title="Insira o Keyname do Projeto. É obrigatório!"
/>
```

- Title

Como o próprio nome sugere, corresponde ao título do projeto e está definido de forma muito semelhante ao Keyname, mudando apenas o conteúdo do atributo name.

- Subtitle

Tem um comportamento semelhante a Title, no entanto, visto que não é obrigatório não possui o atributo *required*.

- Begin Date e End Date

Estes dois campos, apenas diferem no atributo name, aplicámos ainda um novo tipo do elemento input para formulário em HTML que é o tipo "date". Possui ainda um novo atributo presente no HTML5, o *placeholder* que coloca um determinado texto definido no campo e quando se clica ele desaparece. Estes campos ficaram definidos da seguinte forma:

```
<input id="end_date"
       name="end_date"
       type="date"
       required=""
       pattern="\d{4}\-\d{2}\-\d{2}"
       placeholder="aaaa-mm-dd"
       title="Data de fim do projeto. Tenha em atenção o formato da data."
/>
```

Os elementos referidos até agora faziam parte do Header, falando agora dos supervisores a forma encontrada até ao momento para facilitar a interação com o utilizador, foi criar uma tabela onde estão listados todos os supervisores e o utilizador seleciona aqueles que lhe interessa, no entanto, se esta tabela for demasiado grande deveria possuir um sistema de paginação que não está feito. Neste momento, para não estragar completamente o layout da página, se existirem mais do que 10 supervisores, fica disponível na tabela uma barra de scroll vertical para ser possível visualizar todos.

Para os atores, adotámos um comportamento semelhante à listagem dos supervisores.

O que descreve o projeto é o Abstract, este consiste numa caixa de texto onde é possível inserir alguns elementos com a ajuda de botões colocados por baixo. O que os botões fazem é inserir as tags correspondentes aos vários elementos que podem fazer parte do abstract. Basta clicar no botão pretendido e ele insere as tags no local onde esteja posicionado o cursor. Caso seja um url que se pretenda inserir no abstract, são apresentadas duas caixas de diálogo ao utilizador, uma para colocar o nome que se quer dar ao url e outra para colocar o url.

Depois de preenchidos os itens anteriores, falta apenas os ficheiros para anexar. Estes podem ser adicionados na secção Deliverables e é permitido anexar até seis ficheiros. O efeito produzido pelos botões Adicionar e Remover é conseguido através da utilização de JavaScript que altera o CSS de forma a mostrar ou não elementos que já existem na página.

Quanto ao tamanho máximo que se pode inserir é o predefinido pelo PHP, 2MB e é permitido todo o tipo de ficheiros.

Por fim, existe ainda uma opção que é tornar o projeto privado. Ativada esta opção, um projeto deixa de ser visível publicamente, apenas quem o submeteu e os administradores tem acesso a este projeto.

Depois de preenchidos todos os campos pretendidos no formulário é necessário o utilizador clicar em “Enviar”. Do lado do servidor, vai ser recebida toda a informação que foi inserida e criar um Project Record em XML. Coloca ainda os ficheiros numa pasta temporária, o nome dos ficheiros é alterado para o seu md5, isto para evitar que ficheiros diferentes com o mesmo nome estejam na zona temporária.

O Project Record criado é validado por um Schema e caso não esteja correto não é permitido ao utilizador prosseguir na submissão.

Confirmada a informação, o sistema vai proceder ao seu registo na Base de Dados e guardar os ficheiros numa zona definida para o efeito. É criada uma pasta para cada projeto, essa pasta vai ter o nome que corresponde ao md5 calculado com o conteúdo do Project Record. Dentro da pasta vão ficar os ficheiros anexados e ainda um com o nome pr.xml que é o manifesto do projeto submetido.

Para não haver um problema de excesso de pastas ou ficheiros por diretoria, é criada uma pasta para cada ano, dentro dessa, uma para cada mês e mais um nível que é uma pasta por dia, desta forma um projeto que seja submetido hoje (22/01/2012) fica armazenado em "2012/01/22/md5(conteúdo do pr.xml)", desta forma, dificilmente teremos problemas de conflitos e excesso de ficheiros por pasta.

Ficheiro ZIP

Para submeter um SIP através de um ficheiro ZIP, é oferecido um pequeno formulário onde se escolhe o ficheiro a submeter e ainda uma opção para tornar o projeto privado.

Submetido o ficheiro, ele vai ser validado, a validação passa por verificar se existe um ficheiro chamado pr.xml e verificar se todos os ficheiros referenciados no manifesto existem no zip. Caso esta validação tenha sucesso, é aplicado ao pr.xml uma stylesheet que mostra o seu conteúdo numa página HTML.

Depois de confirmada toda a informação, o projeto vai ser armazenado num sistema de ficheiros com a mesma estrutura que o da submissão por formulário.

É possível submeter um zip com pastas dentro, ficando salvaguardada essa estrutura depois no repositório.

Processo de Disseminação

De um projeto

É possível ver o conteúdo do repositório navegando no Website. No entanto podemos ainda extrair essa informação para um ficheiro ZIP, originando um pacote semelhante ao SIP original. No final da página existente para cada projeto, existe a opção "Download", que faz com que o sistema procure os ficheiros associados ao projeto e crie um zip numa pasta temporária para o utilizador conseguir fazer download. Dentro do zip criado, estão todos os ficheiros relativos ao projeto e ainda o manifesto com o nome de pr.xml.

Do repositório

Além de ser possível extrair um projeto, um administrador tem ainda uma opção de exportar todo o conteúdo do repositório. Esta exportação não se limita apenas a comprimir o Sistema de Ficheiros que contém os projetos, mas também cria um mega manifesto, ou seja, um ficheiro xml que contém toda a informação contida na Base de Dados, todos os registos das tabelas Project, Author, Supervisor, Deliverable, KeyWord, ProjAut, ProjSup e ProjKW. Desta forma, é possível reconstruir o repositório apenas com este manifesto e os ficheiros dentro do zip.

Ficam de fora as estatísticas, os utilizadores registados e ainda o registo dos logs.

Inserir/alterar/remover utilizadores

O sistema permite três tipos de utilizadores: Consumidores, Produtores e Administradores. Neste momento, apenas os Administradores podem inserir novos utilizadores e podem inserir de qualquer tipo.

Um administrador pode ainda alterar ou remover consumidores e produtores, no entanto não pode alterar ou remover administradores, apenas se pode alterar e remover a si próprio.

Inserir/alterar/remover autores e supervisores

Apenas o administrador pode realizar as tarefas de inserção, alteração e remoção de autores e supervisores. Estas opções estão disponíveis na categoria Gerir.

Logs

Neste capítulo, pretende-se explicar o funcionamento do sistema de logs, o que é registado, como é registado e como é exibido este registo ao Administrador.

O que é registado?

Cada entrada no log contem o username e o nome do utilizador que despoletou a Ação, a data da Ação, a Ação e a sua descrição.

Nos logs são registadas as seguintes ações, todas despoletadas pelo utilizador:

- Ações de Administração:
  - Listagem de:
    - Utilizadores
    - Autores
    - Supervisores
    - Projetos
  - Inserção de:
    - Utilizadores
    - Autores
    - Supervisores
  - Alteração de:
    - Utilizadores
    - Autores
    - Supervisores
  - Remoção de:
    - Utilizadores
    - Autores

        - Supervisores
        - Projetos
- Ações de Disseminação:
    - Listagem de:
        - Autores
        - Supervisores
        - Projetos
    - Exportação de:
        - Projetos
        - Repositório
- Ingestão de um SIP
- Login no Sistema

Como é registado?

Ficou decidido durante as aulas de Processamento Estruturado de Documentos (PED) que o formato dos logs seria XML e foi desenvolvido, também durante as aulas, um schema que validaria o ficheiro XML que mantém o registo dos logs, schema esse que é apresentado de seguida:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
	<xs:element name="logs">
		<xs:complexType>
			<xs:sequence>
				<xs:element maxOccurs="unbounded" minOccurs="0" name="log">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="username"
							type="xs:string"/>
							<xs:element name="name"
							type="xs:string"/>
							<xs:element name="date"
							type="xs:dateTime"/>
							<xs:element name="action"
							type="xs:string"/>
							<xs:element name="description"
							type="xs:string" minOccurs="0"/>
							</xs:sequence>
					</xs:complexType>
				</xs:element>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

No final de cada acção enunciada acima, um nova entrada é adicionada ao final do ficheiro, contendo a informação referente à acção. Se na altura de inserção, um ficheiro de log não existir, então este é criado.

Para lidar com esta tecnologia (XML), estamos a utilizar a biblioteca que já vem de raiz com o PHP 5, o SimpleXML, que fornece um conjunto de ferramentas muito fácil e simples de utilizar para converter XML em objetos que podem ser processados normalmente com seletores de propriedades e iteradores de arrays, podendo, por exemplo, inserir novos elementos, procurar certos nodos utilizando expressões XPath, entre outras funcionalidades.

Como é exibido?

Um administrador do repositório pode consultar o registo de logs, para isso, por uma questão de facilidade de leitura, a ordem dos logs é invertida, passando a serem exibidos em primeiro os logs mais recentes. Pensámos em duas soluções tendo em conta esta situação, uma delas passaria por introduzir uma nova entrada no início do ficheiro ao invés de ser no fim, a outra seria manter a inserção no final do ficheiro, mas na altura de exibição, aplicar uma stylesheet que ordenasse os elementos log por ordem decrescente da data e exibir esse ficheiro transformado em vez do original. Por motivos académicos, optámos pela segunda solução, para poder aplicar uma transformação num ficheiro XML e ter como resultado outro ficheiro XML.

A stylesheet criada para efetuar essa transformação é a seguinte:

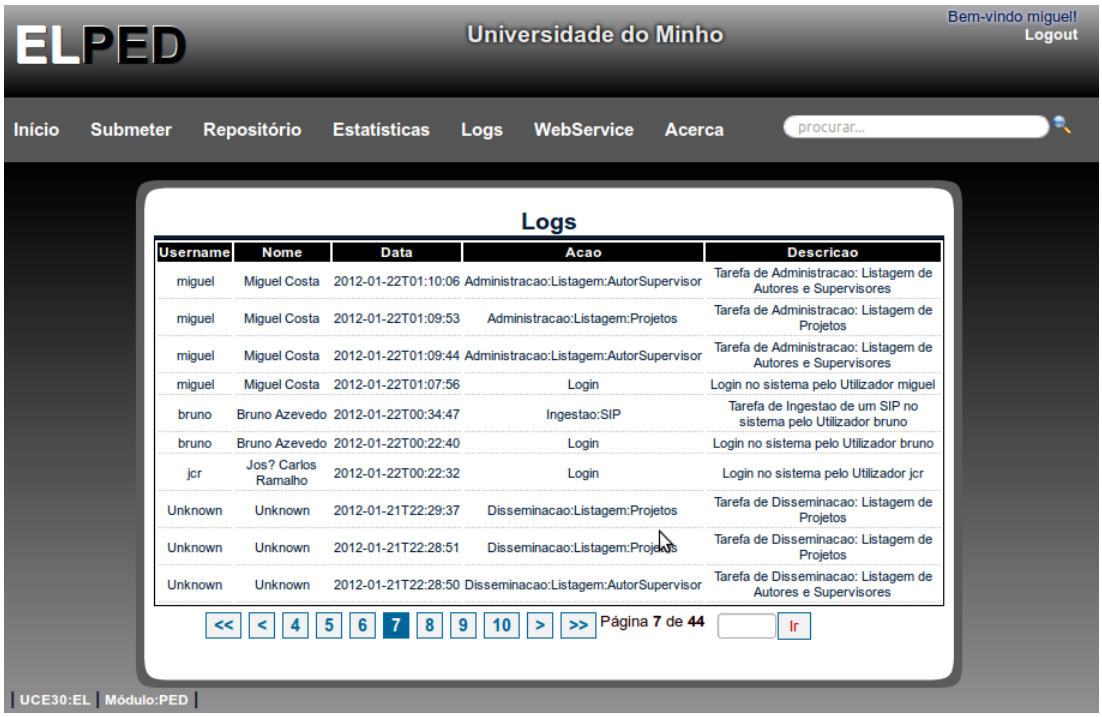
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
	<xsl:output method="xml" encoding="ISO-8859-1" indent="yes"/>
	<xsl:template match="logs">
		<logs>
			<xsl:apply-templates>
				<xsl:sort select="date" order="descending"></xsl:sort>
			</xsl:apply-templates>
		</logs>
	</xsl:template>

	<xsl:template match="log">
		<log>
			<username><xsl:value-of select="username" /></username>
			<name><xsl:value-of select="name" /></name>
			<date><xsl:value-of select="date" /></date>
			<action><xsl:value-of select="action" /></action>
			<description><xsl:value-of select="description" /></description>
		</log>
	</xsl:template>
</xsl:stylesheet>
```

O que se está a passar na stylesheet é que o template do elemento logs tem uma instrução xsl:apply-templates com um flho xsl:sort para dizer ao processador XSLT que ordene os elementos filho do elemento logs, ou seja, os elementos log. O atributo select da instrução xsl:sort especifica a chave de ordenação a utilizar, que é o date (data). O template do elemento log, obriga a envolver o conteúdo de cada log com a tag correspondente, já que o

template por defeito para os nodos de texto e atributos é a cópia do valor desses nodos para o documento destino.

De seguida, mostramos uma imagem correspondente à exibição do registo de logs.



Estatísticas

Neste capítulo, falaremos das estatísticas geradas pelo repositório, bem como da tecnologia usada para as representar. Ver capítulo Conceção da Resolução, secção Base de Dados para saber mais pormenores acerca da estrutura por detrás do cálculo das estatísticas.

Estatísticas geradas

Acessos

O número total de acessos é sempre exibido.

Acessos por data

É mostrado o número de acessos por data, podendo o utilizador escolher se quer ver as estatísticas por mês ou por ano. Apenas são mostrados os dados referentes aos últimos 12 meses de atividade, no caso das estatísticas por mês.

Top Acessos por utilizador

São mostrados os 10 utilizadores com mais acessos, indicando para cada utilizador o número de acessos realizados.

Depósitos

O número total de depósitos é sempre exibido.

Depósitos por data

É mostrado o número de depósitos por data, podendo o utilizador escolher se quer ver as estatísticas por mês ou por ano. Apenas são mostrados os dados referentes aos últimos 12 meses de atividade, no caso das estatísticas por mês.

Top Depósitos por utilizador

São mostrados os 10 utilizadores que efetuaram mais depósitos, indicando para cada utilizador o número de depósitos realizados.

Top Depósitos por curso

São mostrados os 10 cursos com mais depósitos de projetos, indicando para cada curso o número de depósitos realizados.

Downloads

O número total de downloads é sempre exibido.

Top Downloads

São mostrados os 10 projetos com mais downloads realizados, indicando para cada projeto o número de downloads realizados.

Consultas

O número total de consultas é sempre exibido.

Top Projetos

São mostrados os 10 projetos que foram mais consultados, indicando para cada projeto o número de consultas realizadas.

Top Autores

São mostrados os 10 autores que foram mais consultados, indicando para cada autor o número de consultas realizadas.

Top Supervisores

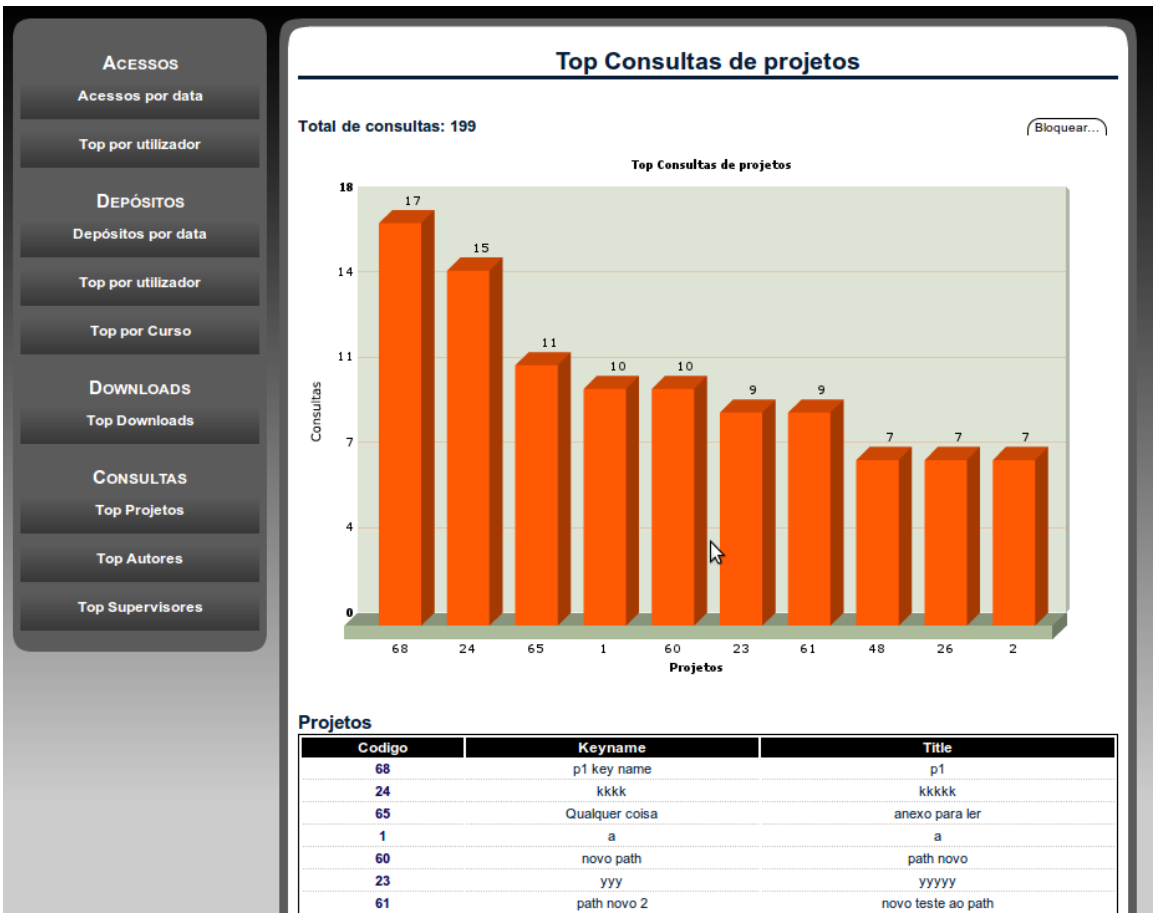
São mostrados os 10 supervisores que foram mais consultados, indicando para cada supervisor o número de consultas realizadas.

Tecnologia usada

"*O FusionCharts Free é um componente open-source de criação de gráficos em flash, que pode ser usado para gerar gráficos animados que ingerem dados. Feito em Macromedia Flash MX, o FusionCharts pode ser usado com qualquer linguagem de scripting web como PHP, ASP, .NET, JSP, ColdFusion, JavaScript, Ruby on Rails, etc., para fornecer gráficos interactivos e poderosos. Utilizando XML como interface de dados, o FusionCharts dá uso à beleza do flash para criar gráficos compactos e interactivos*" (FusionChartsFree Documentation).

É um componente bastante simples de utilizar, basta apenas criar uma string XML que conterá os dados a serem ingeridos pelo Fusion Charts, depois é só utilizar a API fornecida e gerar o gráfico, imprimindo-o para o browser.

De seguida, mostramos uma imagem ilustrativa de um gráfico gerado.



Pesquisa

Neste capítulo iremos explicar o que fizemos para implementar um motor de pesquisa por texto. Para isso, falaremos um pouco das tecnologias envolvidas, do seu funcionamento básico e da sua implementação no âmbito do nosso projeto.

Tecnologias

Lucene

O Apache Lucene(TM) é uma biblioteca open-source de alta performance de um motor de pesquisa de texto completo, que foi escrita, na íntegra, em Java. É uma tecnologia adequada para praticamente qualquer aplicação que necessite de procura por texto completo, especialmente para aquelas que utilizam várias plataformas.

Torna-se fácil procurar por texto porque o Lucene adiciona conteúdo a um índice full-text, que é um índice onde os dados são colecionados, analisados e armazenados para facilitar a recuperação de informação de uma forma rápida e precisa. Depois de procurar no índice, devolve os resultados classificados pela sua relevância para a query ou por um campo aleatório.

Procura e indexação

O Lucene é capaz de alcançar respostas rápidas a pesquisas porque, em vez de pesquisar o texto diretamente, pesquisa um índice. Seria o equivalente a obter páginas de um livro relacionadas com uma palavra-chave através do índice na parte de trás de um livro, em oposição a procurar as palavras em cada página do livro.

Este tipo de índice é chamado índice invertido, porque inverte uma estrutura de dados centrada em páginas (página -> palavras) numa estrutura de dados centrada em palavras-chave (palavra -> páginas).

Documentos

No Lucene, um Documento é a unidade de procura e índice. Um índice consiste num ou mais Documentos.

Um Documento Lucene não precisa de ser, necessariamente, um documento no sentido comum da palavra.

Campos

Um documento consiste num ou mais campos. Um campo é simplesmente um par nome-valor.

A indexação no Lucene envolve a criação de documentos constituídos por um ou mais campos, e adicionando esses documentos a um IndexWriter.

Pesquisar

Pesquisar exige que um índice já exista. Envolve a criação de uma query e a passagem desta query para um IndexSearcher, que retorna uma lista de resultados.

Queries

O Lucene tem a sua própria "mini-linguagem" para a realização de pesquisas.

A linguagem de consulta Lucene permite ao utilizador especificar que campo quer pesquisar, a que campos dar mais peso, a capacidade de executar queries "booleanas" (AND, OR, NOT) e outras funcionalidades.

Zend Search Lucene

O Zend_Search_Lucene deriva do projeto Apache Lucene, foi escrito, na íntegra, em PHP 5 e é um componente da framework Zend. É este componente que iremos utilizar para implementar o nosso motor de pesquisa.

Implementação

Nesta secção, falaremos das nossas decisões no que diz respeito ao que será o nosso Documento e os seus campos. Também falaremos de questões relacionadas com a atualização do índice e utilização de mecanismos de armazenamento temporário para melhorar o tempo de resposta das pesquisas.

Documentos

Decidimos que a nossa unidade de procura e índice seria o projeto, podendo, assim, o utilizador procurar por atributos associados a um projeto e a pesquisa devolver os projetos que fizerem match.

De seguida, apresentamos os tipos de campos que o Zend Search Lucene proporciona e as nossas escolhas para os atributos de um projeto que serão indexados.

Campos

- **Keyword**: Campos que são armazenados e indexados, o que significa que podem ser pesquisados, assim como exibidos nos resultados da pesquisa. Não são divididos em palavras separadas por tokenization. Campos multi-valor de uma Base de Dados são, normalmente, bons candidatos;
- **UnIndexed**: Campos que não são pesquisáveis, mas são retornados com os resultados da pesquisa. Timestamps de Bases de Dados, chaves primárias, caminhos de sistema de ficheiros são bons candidatos para campos UnIndexed;
- **Binary**: Campos que não são separados por tokenization nem indexados, mas são armazenados para exibição por resultado de pesquisa. Podem ser usados para armazenar todos os dados codificados como uma sequência binária, como um ícone de imagem;
- **Text**: Campos que são armazenados, indexados e separados por tokenization. Campos text são apropriados para o armazenamento de informação como temas e títulos que precisam de ser pesquisáveis, bem como retornados com os resultados da pesquisa;

- **UnStored**: Campos são separados por tokenization e indexados, mas não são armazenados no índice. Grandes quantidades de texto são melhor indexadas utilizando este tipo de campo. Armazenar dados cria um índice maior no disco, por isso, se o que se pretende é pesquisar sem exibir os dados, este campo é o indicado.

Tendo em conta os tipos de campo enunciados, iremos mostrar as nossas opções sobre a associação entre os atributos de um projeto e os campos Lucene:

- Keyword:
    - Project.projcode
    - Project.keyname
    - Project.private
    - Author.authorid

  São campos que ajudam a identificar um projeto, serão exibidos ao utilizador e não precisam de ser separados por tokenization.

- UnIndexed:
    - Project.subdate
    - Author.authorcode

  São campos que servirão apenas para exibição ao utilizador, logo não necessitam de ser indexados.

- Text:
    - Project.title
    - Author.name

  São campos que ajudam a identificar um projeto, serão exibidos ao utilizador e serão separados por tokenization para melhorar a pesquisa.

- UnStored:
    - Project.subtitle
    - KeyWord.keyword (lista de keywords associadas a cada projeto)
    - Deposits.username

  São campos que serão apenas pesquisáveis e para ajudar a pesquisa serão separados por tokenization.

Outras Decisões

Atualização do Índice

Neste momento, não temos uma política ótima de manutenção do índice, mas tendo em conta o tamanho inicial da base de dados, que será relativamente baixo, achámos por bem

optar por uma política que consiste na atualização do índice após inserção, remoção ou alteração de um projeto.

Sabemos que esta não é a melhor política, mas numa fase experimental, à medida que o repositório vai crescendo, iremos analisar o comportamento do sistema em questões de tempos de resposta, obstrução do desempenho global do sistema, etc... Uma solução que seria interessante implementar seria a reconstrução do índice programada para uma certa altura do dia onde houvesse menos atividade no repositório.

Melhoramento da pesquisa

Para melhorar o tempo de respostas de uma pesquisa, decidimos implementar um sistema de caching. Utilizámos mais uma biblioteca da framework Zend, Zend Cache, que fornece uma maneira genérica de de guardar em cache qualquer tipo de dados.

Basicamente, sempre que uma pesquisa é requerida, o resultado da query é guardado em cache, permitindo assim que, numa próxima pesquisa pela mesma query, não seja necessário pesquisar no índice Lucene, bastando apenas carregar o resultado da cache.

Este tipo de solução traz alguns problemas de consistência de dados, por isso, para tentar solucionar estes problemas, definimos um tempo de vida da cache de 10 segundos, após o qual a cache será apagada. Se for um tempo de vida demasiado curto, não tiramos proveito do mecanismo, se for muito grande ou infinito, corremos o risco de ter problemas de consistência de dados.

Trata-se de um tempo de vida experimental, visto que só com testes conseguiremos chegar a uma solução ótima.

Web Service

Apesar de não ser um dos requisitos, um dos desafios propostos era a criação de um web service. Este tipo de serviços usa como linguagem de comunicação o XML e criámos um servidor php e dois tipos de clientes também em php. Para facilitar a tarefa usámos uma ferramenta open source o NuSOAP que consiste num conjunto de classes PHP que permite aos desenvolvedores criar e consumir serviços Web baseados em SOAP.

Usando web services podemos integrar sistemas diferentes e independentes, definindo apenas os campos necessários para utilização do lado do cliente.

Servidor

O servidor criado em PHP disponibiliza dois serviços, um que recebe um id do projeto e devolve o seu conteúdo e outro que recebe uma lista de keywords e devolve uma lista de projetos que contém uma das keywords.

```
$s = new soap server;
$s->register('get project');
$s->registre('get_kw');
```

A resposta é envia em XML facilitando o tratamento da informação do lado do cliente. Há uma estrutura predefinida do que é enviado permitindo, assim, que seja possível, caso algo não tenha corrido bem, ao pedido enviar códigos de erro.

O serviço get_project recebe o id do projeto que é pretendido, procura na base de dados, lê conteúdo do pr.xml e envia para o cliente. Caso não seja possível enviar o projeto, pode enviar 3 tipos de erros:

- ERRO 1001 - A localização do projeto não foi encontrada ou o projeto não existe.
- ERRO 1002 - O projeto já foi removido.
- ERRO 1003 - O projeto é privado.

Quanto ao serviço get_kw, recebe como parâmetro uma lista de keywords separados por ';' (ponto e vírgula) e devolve uma lista de projetos que contém uma dessas keywords.

Pode ainda enviar os seguintes códigos de erro:

- ERRO 1001 - não recebeu nenhuma keyword.
- ERRO 1002 - não há resultados disponíveis.

Cliente

Para integrar no site, foram criados formulários simples para enviar os pedidos ao servidor. Independentemente do serviço que se usa, a implementação é idêntica. O cliente envia os parâmetros correspondentes ao serviço que quer e, após receber a resposta em xml, aplicamos uma stylesheet que gera HTML para visualizar no browser, facilitando assim a leitura do que foi obtido pelos clientes.

Melhoramentos futuros

Num sistema informático, geralmente há sempre melhoramentos que se podem fazer, por isso, como não somos exceção, gostávamos de ter conseguido fazer algumas coisas melhor ou de forma diferente.

Um melhoramento importante é a paginação, apesar de estar implementado em alguns sítios, existem outros onde seria útil estar implementado.

Como funcionalidade extra, gostávamos de ter conseguido implementar um mecanismo de avaliação dos projetos, algo como um consumidor, que pudesse avaliar um projeto numa escala definida.

Dificuldades encontradas

Com um sistema deste género, encontrámos dificuldades que não estávamos a contar.

A principal foi o encoding dos caracteres. Tivemos grandes dificuldades em resolver este problema porque, às vezes, surgiam situações em que a informação estava guardada com uma determinada codificação na Base de Dados, os ficheiros php estavam com outra codificação e os ficheiros que eram criados automaticamente pelo php por chamadas ao sistema eram criados com outra diferente. Durante todo o projeto deparámo-nos com este problema porque desde o início, não conseguimos avaliar bem a situação.

Um outro problema mas para o qual já estávamos avisados, era a utilização de CSS que não era visto por todos os browsers da mesma forma, aconteceu várias vezes termos de fazer ajustes no CSS para que fosse possível visualizar corretamente. Também aconteceram situações em que algumas das novidades oferecidas pelo HTML5 não estavam disponíveis em todos os browsers.

Conclusão

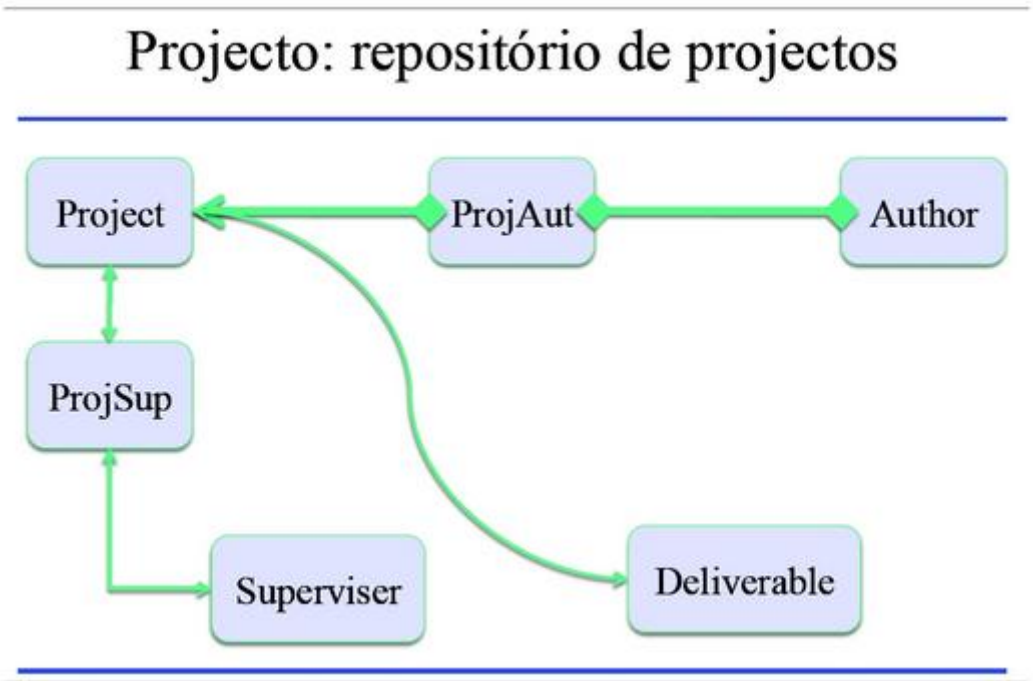
Terminado todo este projeto, a bagagem de conhecimentos adquirida foi bastante grande. Todos os conceitos envolvidos por vezes podiam ser simples, mas quando aplicados vemos a sua complexidade e utilidade que podem ter certas tecnologias.

Analisando a evolução da internet nos últimos anos, ter conhecimentos de ferramentas genéricas de criação de websites e de transferência de informação torna-se essencial, por isso a utilização intensiva do formato XML permitiu-nos ganhar uma à-vontade que perante novos problemas, em que se possa usar XML, não devemos ter grandes dificuldades.

Cumprimos os objetivos, no entanto sabemos que podíamos melhorar em certos aspetos o sistema desenvolvido.

Anexos

Modelo Relacional



Modelo Lógico da Base de Dados