Aplicações na Web

Relatório do Projecto

Etapa 03

Grupo 15

Segunda-feira, 24 de Maio de 2010

Alunos:

- Carlos Álvares, 33305, TP quarta-feira
- João Costa, 33333, TP segunda-feira
- Gonçalo Cruchinho, 33712, TP quarta-feira

Índice

1. Planeamento

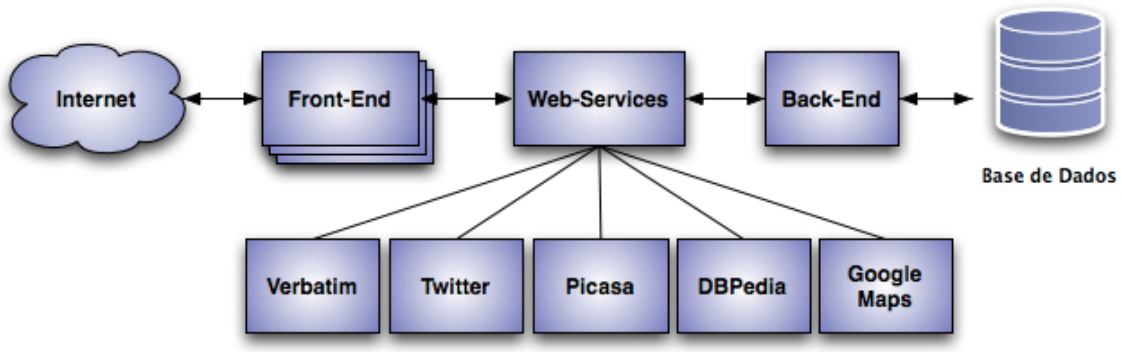
Semana	Nome da Tarefa	Duração	Responsável
1 a 2	**Modelação**	11 dias	
1	Identificação de requisitos	1 dia	Todos
1	Definição da arquitectura	2 dias	Todos
1	Elaboração do modelo de dados	3 dias	João Costa
1 a 2	Esboço das interfaces	2 dias	Carlos Álvares
2	Relatório	3 dias	Todos
2 a 6	**Elaboração do Back-end**	26 dias	
2 a 3	Scripts de Crawling em PHP	8 dias	Gonçalo Cruchinho
3 a 4	Scripts de Parsing em PHP	8 dias	Gonçalo Cruchinho e João Costa
4 a 6	Base de dados MySQL	8 dias	Carlos Álvares e João Costa
6	Relatório	2 dias	Todos
6 a 8	**Elaboração do Web Service**	15 dias	
6	Implementação da API	5 dias	Gonçalo Cruchinho
7	Implementação da API	5 dias	João Costa
8	Implementação da API	5 dias	Carlos Álvares
8	Relatório	2 dias	Todos
8 a 9	**Elaboração do Front-end**	10 dias	
8	Implementação	6 dias	Todos
9	Apresentação	2 dias	Todos
9	Relatório	2 dias	Todos
9	**Fim do Projecto**	0 dias	

2. Arquitectura

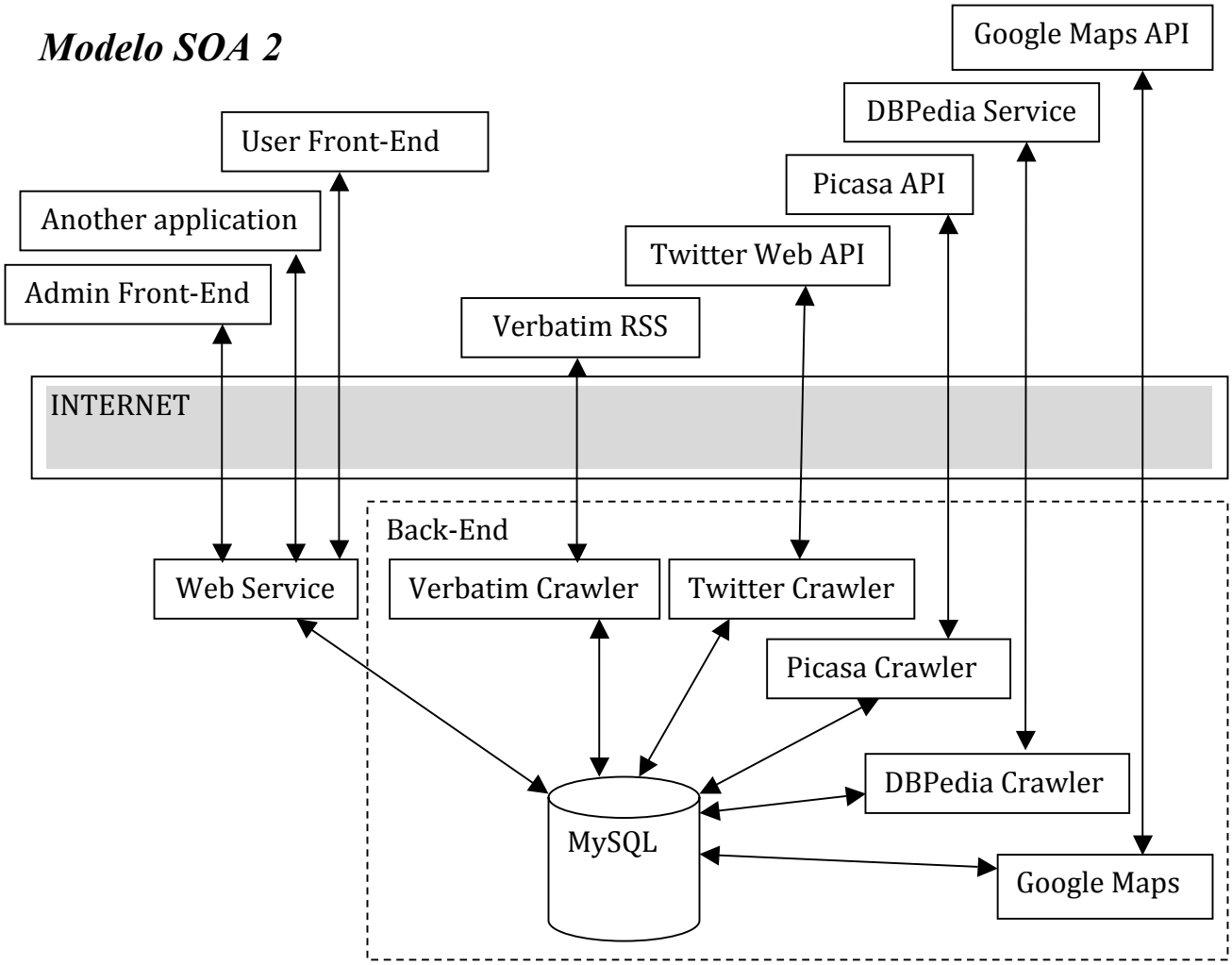
Para arquitectura do nosso projecto, o nosso grupo escolheu uma arquitectura orientada ao Serviço – Arquitectura SOA.

Esta vai permitir que o âmbito do projecto esteja liberto de restrições tecnológicas, visto que grande parte das suas componentes já existem algures na Web, bastando apenas reutiliza-las. Ao seguir esta arquitectura vamos conseguir definir quais e como as componentes que interagem entre si.

Modelo SOA 1



Modelo SOA 2



Neste último modelo identificamos diversos grupos de componentes e de serviços:

Componentes:

1. Obtenção de Informação das várias fontes
2. Interpretação de Informação recolhida das várias fontes

Serviços:

1. Serviços de Informação
   Este grupo vai englobar os Web-Services que disponibilizamos:
   - Pesquisar uma Personalidade
   - Obter imagens relacionadas com a Personalidade
   - Obter informação Biográfica de uma Personalidade através da DBPédia
   - Obter as Personalidades mais pesquisadas
   - É possível subscrever por RSS uma dada Personalidade
   - É possível obter um mapa Google Maps das localizações dos autores dos Post

2. Interface com as fontes
   Este grupo vai englobar os Serviços disponibilizados que vão permitir interagir com API's de outros serviços:
   - Permitir fazer uma interrogação à DBPedia obtendo a sua resposta.
   - Permitir obter uma fotografia que esteja disponibilizada em Albuns Públicos no Picasa.
   - Permitir obter o estado actual de um dado utilizador do Twitter (Status-show)
   - Permitir obter Posts das fontes guardadas no sistema (Verbatim e Twitter no mínimo)
   - Permite obter a localização de um autor

3. Back-end

3.1. Tecnologias usadas

- PHP – Esta linguagem é utilizada para gerar as páginas HTML a serem apresentadas.

- Javascript – Recorremos a esta linguagem para permitir a utilização de AJAX a fim de apresentar dinamicamente algum conteúdo nas páginas.

- MySQL – Esta linguagem é utilizada no âmbito da construção da base de dados e execução das interrogações dos nossos Web-Services.

- XML - Recorremos a uma biblioteca PHP OpenSource a fim de poder interpretar e criar XML, necessário para adicionar fontes, ou interpretar a informação recebida pelos nossos Web-Services proveniente de serviços externos. (http://www.criticaldevelopment.net/xml/doc.php)

3.2. Web-Services

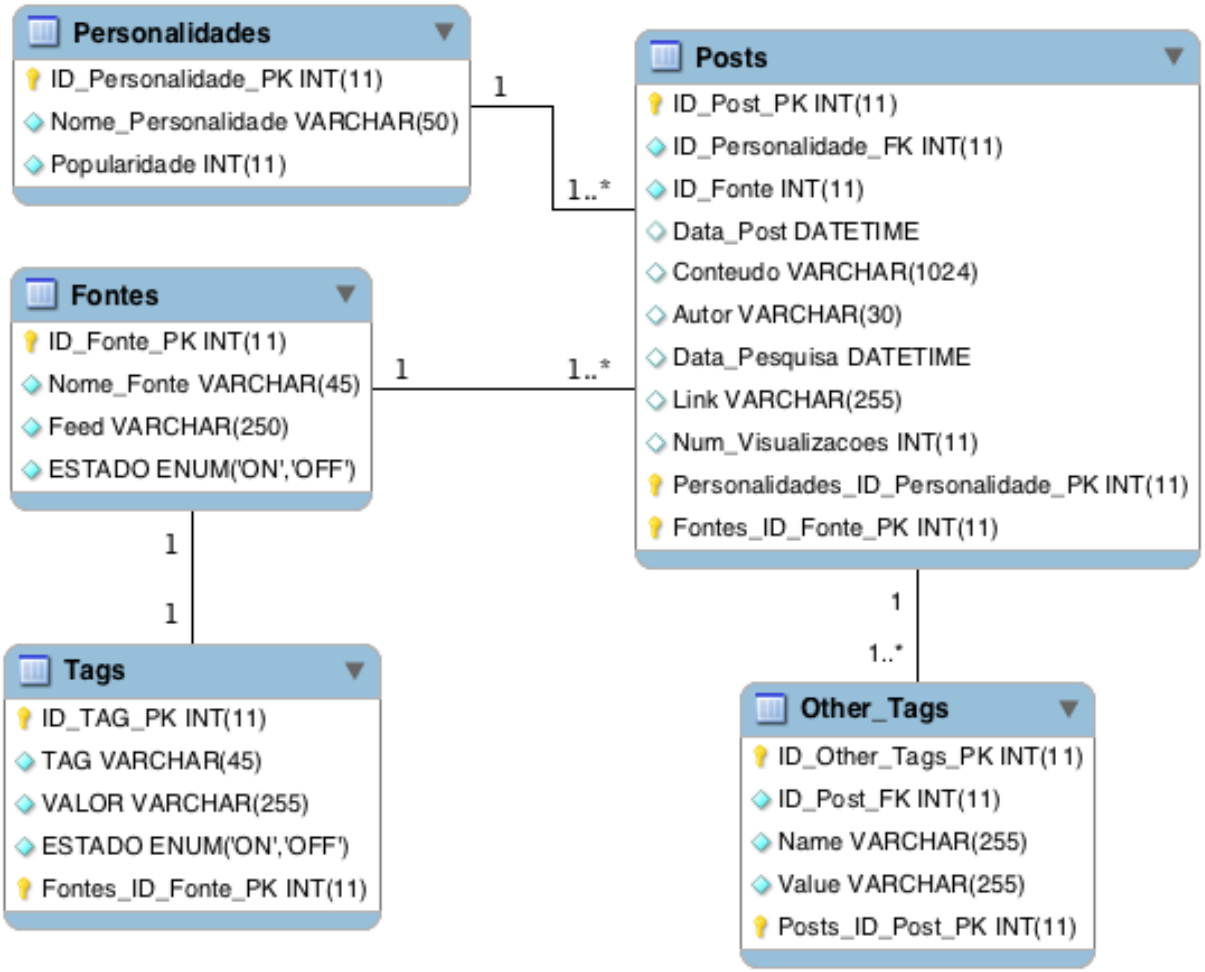
Nesta fase o Back-End já recorre à nossa API de Web-Services para executar as mais diversas tarefas, tais como inserção de Fontes, Personalidades e Posts, obtenção de uma lista das Fontes disponíveis, Personalidades no sistema, etc.

Foram utilizados por base os Web-Services de Feeds do Verbatim como fonte de Personalidades a pesquisar e posts sobre as mesmas, bem como o serviço de pesquisa Atom oferecido pela API do Twitter como meio de obter mais posts sobre as já referidas Personalidades. Caso seja necessário acrescentar algum serviço, esta gestão terá que ser feita por um ficheiro XML. Este deverá conter o URI do Web-Service a utilizar, bem como os respectivos parâmetros que lhe serão aplicados. Estes serviços adicionais podem ser do tipo *Atom* ou *RSS 2.0*.

Exemplo:

```
<root>
  <addsource>
      <name>Verbatim</name>
      <uri>http://verbatim.labs.sapo.pt/feeds</uri>
  </addsource>
  <addsource>
      <name>Twitter</name>
      <uri>http://search.twitter.com/search.atom</uri>
      <option>
          <name>rpp</name>
          <value>5</value>
      </option>
      <option>
          <name>q</name>
          <value></value>
      </option>
  </addsource>
</root>
```

3.3. Estrutura de Dados



3.4. Componentes Implementados

Existe um conjunto de ficheiros PHP responsáveis pelo GUI da Interface de Gestão do Back-End, que fazem a ligação entre este e os Web-Services de gestão que a nossa aplicação disponibiliza. Existe também um ficheiro PHP (*SourceParser.php*) que realiza todo o *parsing* da informação obtida em XML das várias fontes que se encontrem disponíveis no nosso sistema. Este é capaz de filtrar a informação obtida com recurso aos parâmetros associados a cada fonte, registando o resultado na base de dados como já mencionado no ponto 3.2. Essencialmente, os nossos componentes vão recorrer aos Web-Services que a nossa aplicação já oferece como referido anteriormente e aos Web-Services de Atom-Search e Status-Show do Twitter.

4. Web-Services

4.1. Tecnologias usadas

- PHP – Esta linguagem é utilizada para interagir com a base de dados, tratar informação, e construir o XML de retorno.
- MySQL – Esta tecnologia vai ser utilizada apenas para a construção de interrogações à nossa base de dados.
- XML - Recorremos à mesma biblioteca PHP OpenSource, para ter a possibilidade de criar XML necessário para a criação de respostas XML. (http://www.criticaldevelopment.net/xml/doc.php)
- Utilizamos respostas XML em formato RSS 2.0.
- A nossa API está implementada sobre a metodologia RESTful.
- Documento WADL para definir a nossa aplicação web (em anexo).

4.2. Lista de Métodos

Já que estamos numa API REST, os URI devem construídos com os nomes do método seguido dos respectivos parâmetros (../celebrity/name/Franciso+Couto/).

Nome do Método	***Método HTTP***	***Parametros***	***Retorno***
celebrity	GET	[name/xxx/]	Todas as celebridades ou filtra com base no parâmetro opcional *name* que representa o nome da celebridade.
celebrity	POST	name/xxx/	Adiciona a personalidade com o nome definido no parâmetro abrigatório *name*.
source	GET	[identifier/xxx]	Apresenta todas as fontes ou filtra com base no parâmetro opcional *identifier* que representa o nome da fonte ou id da fonte.
source	POST	uploadedfile	Adiciona as fontes existentes no ficheiro XML submetido.
sourcestate	GET	[identifier/xxx]	Apresenta o estado de todas as fontes ou filtra com base no parâmetro identifier.
sourcestate	PUT	Indentifier/xxx/state/yyy	Altera o estado da fonte identificada por xxx para o estado yyy. Onde yyy pode ter o valor ON|OFF.
sourcetags	GET	[identifier/xxx]	Apresenta todas as tags para todas as fontes ou as tags da fonte identificada pelo parâmetro identifier.
location	GET	[name/xxx]	Apresenta todas as localizações de posts ou localizações identificadas pelo parâmetro name.
datacount	GET	-	Apresenta o total de posts existentes
data	GET	source/xxx OR celebrity/xxx OR source/xxx/celebrity/yyy	Devolve todos os posts existentes ou todos os posts de uma fonte ou todos os post de uma celebridade.
photo	GET	[celebrity/xxx]	Devolve todas as fotos referentes a todas as personalidades ou da personalidade identificada por xxx.
tagstate	PUT	tagid/xxx/state/yyy	Altera o estado da tag identificada por xxx para o estado yyy. Onde yyy pode ter o valor ON|OFF.
tagvalue	PUT	tagid/xxx/value/yyy	Altera o valor da tag identificada por xxx para o valor yyy.

4.3. Exemplos

.../api.php/sources/

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
    <channel>
        <title>AW015:getSources</title>
        <link>http://www.carlosalvares.net/aw/api.php?c=getSources</link>
        <description>Sources available for searching</description>
        <language>pt-pt</language>
        <copyright>AW015</copyright>
        <item>
            <title>BBC News Main Feed</title>
            <link>http://www.carlosalvares.net/aw/api.php?c=getData&amp;amp;s=BBC+News+Main+Feed</link>
            <description>The link to get data from this source</description>
        </item>
        <item>
            <title>Picasa</title>
            <link>http://www.carlosalvares.net/aw/api.php?c=getData&amp;amp;s=Picasa</link>
            <description>The link to get data from this source</description>
        </item>
        <item>
            <title>SemanÃ¡rio Sol</title>
            <link>http://www.carlosalvares.net/aw/api.php?c=getData&amp;amp;s=Seman%C3%A1rio+Sol</link>
            <description>The link to get data from this source</description>
        </item>
        <item>
            <title>Twitter</title>
            <link>http://www.carlosalvares.net/aw/api.php?c=getData&amp;amp;s=Twitter</link>
            <description>The link to get data from this source</description>
        </item>
        <item>
            <title>Verbatim</title>
            <link>http://www.carlosalvares.net/aw/api.php?c=getData&amp;amp;s=Verbatim</link>
            <description>The link to get data from this source</description>
        </item>
    </channel>
</rss>
```

.../api.php/celebrity/name/A/

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
    <channel>
        <title>AW015:getCelebrity</title>
        <link>http://www.carlosalvares.net/aw/api.php?c=getCelebrity&amp;amp;p=A</link>
        <description>Celebrities available for searching</description>
        <language>pt-pt</language>
        <copyright>AW015</copyright>
        <item>
            <title>Aristides Ocante da Silva</title>
            <link>http://www.carlosalvares.net/aw/api.php?c=getData&amp;amp;p=Aristides+Ocante+da+Silva</link>
            <description>The link to get data about this celebrity</description>
        </item>
        <item>
            <title>AssunÃ§Ã£o Cristas</title>
            <link>http://www.carlosalvares.net/aw/api.php?c=getData&amp;amp;p=Assun%C3%A7%C3%A3o+Cristas</link>
            <description>The link to get data about this celebrity</description>
        </item>
    </channel>
</rss>
```

../api.php/sourcetags/identifier/Twitter/

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
    <channel>
        <title>AW015:getSourceTags</title>
        <link>http://www.carlosalvares.net/aw/api.php?c=getSourceTags&amp;s=Twitter</link>
        <description>Tags available for source </description>
        <language>pt-pt</language>
        <copyright>AW015</copyright>
        <item>
            <title>Twitter</title>
            <link />
            <description>Tags and Values follow on categories </description>
            <category id_tag_pk="1" domain="rpp" estado="ON">5</category>
            <category id_tag_pk="2" domain="q" estado="ON" />
        </item>
    </channel>
</rss>
```

5. Front-End

5.1. Tecnologias usadas

- PHP – Esta linguagem é utilizada para gerar as páginas HTML a serem apresentadas.

- Javascript – Recorremos a esta linguagem para permitir a utilização de AJAX a fim de apresentar dinamicamente algum conteúdo nas páginas.

- Ajax – Esta framework é utilizada para ter mais interactividade com o utilizador, quando se pretende actualizar parte da página Web sem ter que actualizar a página toda.

- XML - Recorremos a uma biblioteca PHP OpenSource a fim de poder interpretar e criar XML, necessário para adicionar fontes, ou interpretar a informação recebida pelos nossos Web-Services proveniente de serviços externos. (http://www.criticaldevelopment.net/xml/doc.php)

- Google Maps API – Permite visualizar num mapa os resultados de posts das pesquisas efectadas.

5.2. Web Services usados

Nesta fase o Front-End já recorre à nossa API de Web-Services para executar as diversas tarefas, tais como visualizações das Fontes utilizadas e respectivos estados, pesquisa de posts por personalidades e/ou localidades, o resultado das pesquisas são visualizadas num mapa (Google Maps).

5.3.Screenshots

A seguinte imagem representa a página inicial do front-end da aplicação desenvolvida.

Fig. Página inicial

A página Web está dividida em 5 secções, que vão permitir ao utilizador final interargir com funcionalidades da aplicação. Sendo elas as seguintes, representadas na figura seguinte com os respectivos números:

1. Logotipo, com informção da instituição, disciplica, numero de grupo e ano lectivo vigente.
2. Logotipos das fontes onde os webservices podem recolher informação. Os logótipos que estão descativados são representados por uma cruz vermelha sobre a imagem da fonte.
3. Área de pesquisa, zona onde se pode efectuar pesquisas de posts existentes sobre personalidades e/ou localidades.
4. Área de estatística sobre conteúdos da base de dados, sendo eles o número de fontes, numero de celebridades introduzidas na base de dados e numero total de posts existentes.
5. Imagem geográfica, com marcas nas zonas onde posts foram publicados, sendo estes posts o resultado da pesquisa efectuada.

Fig – Represetação das zonas

Fig – Resultados de uma pesquisa (Leonardo Tavares)

6. Discussão

Com esta aplicação pretendemos oferecer um serviço de pesquisa sobre personalidades, com o qual se possam encontrar posts sobre a mesma informação biográfica retirada do serviço DBPédia e imagens retiradas do serviço Picasa, podemos também fazer uma pesquisa sobre posts de uma celebridade numa data região geográfica. Por isso, surgiram alguns problemas quanto à estrutura definida no relatório anterior, e sua integração na aplicação final.

6.1. Problema1 encontrado

Com o evoluir da aplicação, fomo-nos questionando se faría sentidos estar a construir uma aplicação que integrasse o registo de contas de utilizadores, os quais pudessem subscrever alguns Posts que achassem interessantes, ou que lhes fosse possível criar recomendações em Posts. Decidimos então remover estas tabelas que existiam anteriormente na nossa Base de Dados, visto que pretendemos criar uma aplicação de âmbito informativo acessível a todos, e não apenas a utilizadores registados.

6.2. Problema2 encontrado

A BDPédia retorna dados no formato RFD logo é necessário um handler especial para este tipo de aplicações sendo que essa funcionalidade será implementada como webservice que irá obter directamente os dados do webservice e servi-los formatados em XML para outras aplicações. Alternativamente adicionar um módulo.

6.3. Problema3 encontrado

Como a aplicação teria que ser RESTful teríamos que apresentar, ou não, a extensão .php da nossa API. Ou seja, passar de um URI ../api.php/celebrity/ para um URI ../api/celebrity/. Decidimos deixar com a extensão .php já que não é relevante para o funcionamente da aplicação.

6.4. Problema4 encontrado

Anteriormente só era possível carregar uma fonte de cada vez, atravéz de XML. Porém esta aborgadem era pouco prática, então decidimos permitir carregar mais do que uma fonte no mesmo ficheiro XML.

6.5. Problema5 encontrado

Existem vários serviços que consideramos críticos para o nosso sistema, sendo estes os Verbatim, Twitter, Picasa e DBPedia. Sem estes, o sistema não consegue funcionar.

6.6. Problema6 encontrado

Entedemos que não é relevante o webservice permitir apagar conteúdo da base de dados. Como tal, não existe nenhuma funcionalidade que contemple métodos HTTP DELETE. Desta forma evitamos perdas de dados acidentais.

6.7. Problema7 encontrado

Já que queremos apresentar informação biográfica das personalidades, provenientes da DBPedia, decidimos que este conteúdo seria guardado juntamente com os posts. Isto é possível porque a referência à personalidade do post é igual ao da DBPpedia.

6.8. Problema8 encontrado

Quanto à geo-referenciação dos posts, já que ficou ao nosso critério como mencionado no enunciado, optámos por guardar o nome da localização em vez de recorrer a Latitude e Longitude.

6.9. Problema9 encontrado

Já que queremos apresentar informação fotográfica das personalidades, provenientes do Picasa, decidimos que este conteúdo seria guardado juntamente com os posts. Isto é possível porque a referência à personalidade e do post é igual ao do Picasa.

6.10. Problema10 encontrado

A DBPedia apresenta uma limitação relativa a forma como a mesma retorna informação. A DBPedia retorna informação com base no site http://en.wikipedia.org/ e artigos que existam apenas em português e não em inglês não são retornados pela mesma o que limita a informação que podemos retirar da Wikipédia.

6.11. Problema11 encontrado

No que toca ao desenho do mapa GoogleMaps das localizações dos autores dos posts, tivemos dificuldade no que toca ao Ajax para actualizar dinamicamente a 'div' na qual o mapa está inserido.

6.12. Problema12 encontrado

Tivemos problemas quanto ao desenho de marcadores no mapa GoogleMaps quando existe mais do que um Post numa dada localização. Isto leva a que existam marcadores sobrepostos na mesma posição do mapa, pelo que não é possível ver todo o conteúdo.

6.13. Problema13 encontrado

Infelizmente não conseguimos colocar balões no mapa Google Maps quando as localizações dos posts são coincidentes. Ou seja, se existirem dois posts em Lisboa, estes estão presentes no mapa mas o utilizador so consegue visualizar o que esteja por cima. Isto deve-se ao facto de precisarmos de uma chave do Google Maps que não conseguimos obter. Deve-se também à falta de tempo para implementar estes detalhes, já que existem muitos prazos de entrega de trabalhos com o prazo desta cadeira.

7. Anexos

Especificação WADL

```
<?xml version="1.0" encoding="utf-8"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:schemaLocation="http://research.sun.com/wadl/2006/10 wadl.xsd" xmlns="http://research.sun.com/wadl/2006/10">
	<resources base="http://www.carlosalvares.net/aw/api.php/">
		<resource path="source">
			<method name="GET">
				<request>
					<param name="identifier" type="xsd:string" style="query" default="all"/>
				</request>
				<response>
					<representation mediaType="application/xml" element="RSSResponse"/>
					<fault mediaType="application/xml" status="400"/>
				</response>
			</method>
			<method name="POST">
				<request>
					<param name="uploadedfile" type="xsd:xmlfile" style="header">
						<option value="xsd:xmlfile"/>
					</param>
				</request>
				<response>
					<representation mediaType="application/xml" element="RSSResponse"/>
					<fault mediaType="application/xml" status="400"/>
				</response>
			</method>
		</resource>
		<resource path="celebrity">
			<method name="GET">
				<request>
					<param name="name" type="xsd:string" style="query" default="all"/>
				</request>
				<response>
					<representation mediaType="application/xml" element="RSSResponse"/>
					<fault mediaType="application/xml" status="400"/>
				</response>
			</method>
			<method name="POST">
				<request>
					<param name="name" type="xsd:string" style="query"/>
				</request>
				<response>
					<representation mediaType="application/xml" element="RSSResponse"/>
					<fault mediaType="application/xml" status="400"/>
				</response>
			</method>
		</resource>
		<resource path="sourcestate">
			<method name="GET">
				<request>
					<param name="identifier" type="xsd:string" style="query" default="all"/>
				</request>
				<response>
					<representation mediaType="application/xml" element="RSSResponse"/>
					<fault mediaType="application/xml" status="400"/>
				</response>
			</method>
			<method name="PUT">
				<request>
```

```
                        <param name="identifier" type="xsd:string" style="query" required="true"/>
                        <param name="state" type="xsd:string" style="query" required="true"/>
                    </request>
                    <response>
                        <representation mediaType="application/xml" element="RSSResponse"/>
                        <fault mediaType="application/xml" status="400"/>
                    </response>
                </method>
            </resource>
            <resource path="sourcetags">
                <method name="GET">
                    <request>
                        <param name="identifier" type="xsd:string" style="query" default="all"/>
                    </request>
                    <response>
                        <representation mediaType="application/xml" element="RSSResponse"/>
                        <fault mediaType="application/xml" status="400"/>
                    </response>
                </method>
            </resource>
            </resource>
            <resource path="location">
                <method name="GET">
                    <request>
                        <param name="name" type="xsd:string" style="query" default="all"/>
                    </request>
                    <response>
                        <representation mediaType="application/xml" element="RSSResponse"/>
                        <fault mediaType="application/xml" status="400"/>
                    </response>
                </method>
            </resource>
            <resource path="datacount">
                <method name="GET">
                    <response>
                        <representation mediaType="application/xml" element="RSSResponse"/>
                        <fault mediaType="application/xml" status="400"/>
                    </response>
                </method>
            </resource>

            <resource path="data">
                <method name="GET">
                    <request>
                        <param name="source" type="xsd:string" style="query" default="all"/>
                        <param name="celebrity" type="xsd:string" style="query" default="all"/>
                        <param name="location" type="xsd:string" style="query" default="all"/>
                    </request>
                    <response>
                        <representation mediaType="application/xml" element="RSSResponse"/>
                        <fault mediaType="application/xml" status="400"/>
                    </response>
                </method>
            </resource>
            <resource path="tagstate">
                <method name="PUT">
                    <request>
                        <param name="tagid" type="xsd:string" style="query" required="true"/>
                        <param name="state" type="xsd:string" style="query" required="true"/>
                    </request>
                    <response>
                        <representation mediaType="application/xml" element="RSSResponse"/>
                        <fault mediaType="application/xml" status="400"/>
                    </response>
                </method>
            </resource>
            <resource path="tagvalue">
                <method name="PUT">
                    <request>
```

```
                                        <param name="tagid" type="xsd:string" style="query" required="true"/>
                                        <param name="value" type="xsd:string" style="query" required="true"/>
                                    </request>
                                    <response>
                                        <representation mediaType="application/xml" element="RSSResponse"/>
                                        <fault mediaType="application/xml" status="400"/>
                                    </response>
                                </method>
                            </resource>
                        </resources>
</application>
```

Listagem dos ficheiros:

- SourceParser.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>AW015 Back-End</title>
    </head>

    <body>
        <h1>P&aacute;gina de Configura&ccedil;&atilde;o </h1>
        <h2>Back-End - AW015</h2>
        <hr /><hr />
        <?php

        /**
         * Description of SourceParser
         *
         * @author AW?????
         */

        include "mysql_conn.php";
        include "XMLParser.php";

        $DEFAULT_TIMEOUT = 200000;

        $DEFAULT_TIMEOUT_ON_FAILURE = 1000000;

//          getSourceData();
        getGeoTag();

        function getSourceData() {
            $sourceSql = "SELECT * FROM Fontes";
            $celebritySql = "SELECT ID_Personalidade_PK,Nome_Personalidade FROM Personalidades";
            $connection = connect();
            $sourceData = doSQL($connection, $sourceSql);
            $celebrityData = doSQL($connection, $celebritySql);
            foreach ($sourceData as $source) {
                if (strcmp($source["ESTADO"], "OFF") == 0) {
                    continue;
                }
                $sourceTagsSql = "SELECT * FROM Tags WHERE ID_FONTE_FK = " . $source["ID_Fonte_PK"];
                $sourceTagsData = doSQL($connection, $sourceTagsSql);
                if (empty($sourceTagsData)) {
                    $xml = file_get_contents($source["Feed"],FILE_TEXT);
                    $parser = new XMLParser($xml);
                    $parser->Parse();
                    if (strcmp($parser->document->tagName, "feed") == 0) {
                        getAtomSourceData($connection, $parser, $source["ID_Fonte_PK"], 0);
                    } else if (strcmp($parser->document->tagName, "rss") == 0) {
                        getRSSSourceData($connection, $parser, $source["ID_Fonte_PK"], 0);
                    }
                } else {
                    foreach ($celebrityData as $celebrity) {
                        $url = $source["Feed"];
                        $i = 0;
                        foreach($sourceTagsData as $sourceTag) {
                            if (strcmp($sourceTag["ESTADO"],"OFF") == 0) {
                                continue;
                            }
                            if ($i == 0)
                                $url .= "?";
                            else
                                $url .= "&";
```

```
                                        ++$i;
                                        if (!empty($sourceTag["TAG"]) && empty($sourceTag["VALOR"])) { //se tiver tag mas não tiver valor
                                            $url .= $sourceTag["TAG"] . "=\"" . urlencode($celebrity["Nome_Personalidade"]) . "\"";
                                        } else if (empty($sourceTag["TAG"]) && !empty($sourceTag["VALOR"])) {
                                            $url .= $sourceTag["DATA"];
                                        } else {
                                            $url .= $sourceTag["TAG"] . "=" . $sourceTag["VALOR"];
                                        }
                                    }
                                    echo "final url:" . $url . "<br>";
                                    $xml = file_get_contents($url,FILE_TEXT);
                                    if (!$xml) {
                                        echo "Can't get url '" . $url . "'<br><br>";
                                        usleep($DEFAULT_TIMEOUT_ON_FAILURE);
                                        continue;
                                    }
                                    $parser = new XMLParser($xml);
                                    $parser->Parse();
                                    if (strcmp($parser->document->tagName, "feed") == 0) {
                                        getAtomSourceData($connection, $parser, $source["ID_Fonte_PK"],
$celebrity["ID_Personalidade_PK"]);
                                    } else {
                                        getRSSSourceData($connection, $parser, $source[" ID_Fonte_PK"],
$celebrity["ID_Personalidade_PK"]);
                                    }
                                }
                            }
                            usleep($DEFAULT_TIMEOUT);
                        }
                        disconnect($connection);
                    }

                    function getAtomSourceData($connection, $parser, $sourceID, $celebrityID) {
                        echo $sourceID . " is a atom feed<br>";
                        $authorfeed = false;
                        if (!isset($parser->document->entry)) {
                            echo "No information fount for id " . $celebrityID . ".<br>";
                            return;
                        }
                        foreach($parser->document->entry as $entry) {
                            if ($celebrityID == 0) {
                                $authorfeed = true;
                                $celebritySql = "SELECT ID_Personalidade_PK FROM Personalidades WHERE Nome_Personalidade = '".
$entry->author[0]->name[0]->tagData ."'";
                                $celebrityData = doSQL($connection, $celebritySql);
                                if (empty ($celebrityData)) {
                                    echo "celebrity '" . $entry->author[0]->name[0]->tagData . "' not fount.<br>";
                                    $celebritySql = "INSERT INTO Personalidades VALUES (0,'" . $entry->author[0]->name[0]->tagData . "',
0);";
                                    doSQL($connection, $celebritySql);
                                    $celebritySql = "SELECT MAX(ID_Personalidade_PK) FROM Personalidades";
                                    $celebrityData = doSQL($connection, $celebritySql);
                                    echo "celebrity '" . $entry->author[0]->name[0]->tagData . "' created with id " . $celebrityData[0][0] . ".<br>";
                                } else {
                                    echo "celebrity '" . $entry->author[0]->name[0]->tagData . "' found with id " . $celebrityData[0][0] . ".<br>";
                                }
                                $celebrityID = $celebrityData[0][0];
                            }
                            $postSql = "SELECT ID_Post_PK FROM Posts WHERE Link = '". $entry->link[0]->tagAttrs["href"] . "'";
                            echo $postSql . "<br>";
                            $postData = doSQL($connection, $postSql);
                            if (empty ($postData)) {
                                echo "post '" . $entry->title[0]->tagData . "' not fount.<br>";
                                $mysqldate = date('Y-m-d H:i:s');
                                $postSql = "INSERT INTO Posts VALUES (0," . $celebrityID
                                    . ",". $sourceID. ",'" . $entry->updated[0]->tagData . "','"
                                    . htmlentities(strip_tags($entry->title[0]->tagData),ENT_QUOTES) . "','" .
                                    $entry->author[0]->name[0]->tagData . "','" . $mysqldate .
                                    "','" . $entry->link[0]->tagAttrs["href"] . "',0);";
                                echo $postSql . "<br><br>";
```

```
                    doSQL($connection, $postSql);
                    $postSql = "SELECT ID_Post_PK FROM Posts WHERE Link = '". $entry->link[0]->tagAttrs["href"] . "'";
                    echo $postSql;
                    $postData = doSQL($connection, $postSql);
                    handleOtherTags($connection, $postData[0][0], "", $entry);
                } else {
                    echo "post '" . $entry->author[0]->name[0]->tagData . "' found with id " . $postData[0][0] . ".<br><br>";
                }
                if ($authorfeed)
                    $celebrityID = 0;
            }
        }

        function getRSSSourceData($connection, $parser, $sourceID, $celebrityID) {
            echo $sourceID . " is a rss feed<br>";
            if (!isset($parser->document->channel[0]->item)) {
                echo "No information fount for id " . $celebrityID . ".<br>";
                return;
            }
            foreach($parser->document->channel[0]->item as $item) {
                if ($celebrityID == 0) {
                    if (empty($item->author[0]->tagData)) {
                        $celebritySql = "SELECT ID_Personalidade_PK FROM Personalidades WHERE Nome_Personalidade = '".
$parser->document->channel[0]->title[0]->tagData ."'";
                        $celebrityData = doSQL($connection, $celebritySql);
                    } else {
                        $celebritySql = "SELECT ID_Personalidade_PK FROM Personalidades WHERE Nome_Personalidade = '".
$item->author[0]->tagData ."'";
                        $celebrityData = doSQL($connection, $celebritySql);
                    }
                    if (empty ($celebrityData)) {
                        if (!empty($item->author[0]->tagData)) {
                            echo "celebrity '" . $item->author[0]->tagData . "' not fount.<br>";
                            $celebritySql = "INSERT INTO Personalidades VALUES (0,'" . $item->author[0]->tagData . "', 0);";
                            doSQL($connection, $celebritySql);
                            $celebritySql = "SELECT MAX(ID_Personalidade_PK) FROM Personalidades";
                            $celebrityData = doSQL($connection, $celebritySql);
                            echo "celebrity '" . $item->author[0]->tagData . "' created with id " . $celebrityData[0][0] . ".<br>";
                        } else {
                            echo "celebrity '" . $parser->document->channel[0]->title[0]->tagData . "' not fount.<br>";
                            $celebritySql = "INSERT INTO Personalidades VALUES (0,'" .
$parser->document->channel[0]->title[0]->tagData . "', 0);";
                            doSQL($connection, $celebritySql);
                            $celebritySql = "SELECT MAX(ID_Personalidade_PK) FROM Personalidades";
                            $celebrityData = doSQL($connection, $celebritySql);
                            echo "celebrity '" . $parser->document->channel[0]->title[0]->tagData . "' created with id " .
$celebrityData[0][0] . ".<br>";
                        }
                    } else {
                        if (!empty($item->author[0]->tagData)) {
                            echo "celebrity '" . $item->author[0]->tagData . "' found with id " . $celebrityData[0][0] . ".<br>";
                        } else {
                            echo "celebrity '" . $parser->document->channel[0]->title[0]->tagData . "' found with id " .
$celebrityData[0][0] . ".<br>";
                        }
                    }
                    $celebrityID = $celebrityData[0][0];
                }
                $postSql = "SELECT ID_Post_PK FROM Posts WHERE Link = '". $item->link[0]->tagData . "'";
                echo $postSql . "<br>";
                $postData = doSQL($connection, $postSql);
                if (empty ($postData)) {
                    echo "post '" . $item->description[0]->tagData . "' not fount.<br>";
                    $mysqldate = date('Y-m-d H:i:s');
                    if (!empty($item->author[0]->tagData)) {
                        $postSql = "INSERT INTO Posts VALUES (0," . $celebrityID
                                . ",". $sourceID. ",'" . $item->pubdate[0]->tagData . "','"
                                . htmlentities(strip_tags($item->description[0]->tagData),ENT_QUOTES) . "','" .
                                $item->author[0]->tagData . "','" . $mysqldate .
                                "','" . $item->link[0]->tagData . "',0);";
```

```
                } else {
                    $postSql = "INSERT INTO Posts VALUES (0," . $celebrityID
                        . ",". $sourceID. ",'" . $item->pubdate[0]->tagData . "','"
                        . htmlentities(strip_tags($item->description[0]->tagData),ENT_QUOTES) . "','" .
                        $parser->document->channel[0]->title[0]->tagData . "','"
                        . $mysqldate . "','" . $item->link[0]->tagData . "',0);";
                }
                echo $postSql . "<br><br>";
                doSQL($connection, $postSql);
                $postSql = "SELECT ID_Post_PK FROM Posts WHERE Link = '". $item->link[0]->tagData . "'";
                echo $postSql;
                $postData = doSQL($connection, $postSql);
                handleOtherTags($connection, $postData[0][0], "", $item);
            } else {
                echo "post '" . $item->description[0]->tagData . "' found with id " . $postData[0][0] . ".<br><br>";
            }
            if ($authorfeed)
                $celebrityID = 0;
        }
    }

    function getGeoTag() {
        $connection = connect();
        //obter id do twitter
        $SQL = "SELECT ID_Fonte_PK FROM Fontes WHERE Nome_Fonte = 'Twitter';";
        $results = doSQL($connection, $SQL);
        $id_fonte = $results[0][0];

        //obter todos os posts do twitter
        $SQL2 = "SELECT * FROM Posts WHERE ID_Fonte = ".$id_fonte.";";
        $posts = doSQL($connection, $SQL2);


        //para todos os posts obtidos, vamos obter os ids dos posts no twitter
        $raw_twitter_ids = array();
        $ids_fks = array();
        foreach($posts as $post) {
            $id_post = $post['ID_Post_PK'];
            $SQL3 = "SELECT Value FROM Other_Tags WHERE ID_POST_FK = ".$id_post." AND Name ='id';";
            $ids = doSQL($connection, $SQL3);

            //guardar os ids da tabela e os valores respectivos
            array_push($raw_twitter_ids,$ids[0][0]);
            array_push($ids_fks,$id_post);
        }

        $twitter_ids = array();
        //example id ->       <id>tag:search.twitter.com,2005:12163853479</id>
        foreach($raw_twitter_ids as $raw) {
            $gibberish = explode(":",$raw);
            if (count($gibberish) == 1)
                array_push($twitter_ids,$gibberish[0]);
            else
            array_push($twitter_ids,$gibberish[2]);
        }

        //API Twitter statuses.show
        //http://api.twitter.com/1/statuses/show/id.format
        $index = 0;
        foreach($twitter_ids as $post_id) {
            //obter XML do Twitter
            $url = "http://api.twitter.com/1/statuses/show/".$post_id.".xml";
            echo "opening " .$url. "<br>";
            $xml = file_get_contents($url);
                    if (!$xml) {
                        echo "Can't get url '" . $url . "'<br><br>";
                        usleep($DEFAULT_TIMEOUT);
                        continue;
                    }
            //tratar informaçao
```

```
                $parser = new XMLParser($xml);
                $parser->Parse();
                handleOtherTags($connection, $ids_fks[$index++], "", $parser->document);
                usleep($DEFAULT_TIMEOUT);
            }
            disconnect($connection);
        }

        function handleOtherTags($connection, $postID, $path, XMLTag $base) {
            $expectedTags[] = "author";
            $expectedTags[] = "title";
            $expectedTags[] = "published";
            $expectedTags[] = "pubdate";
            $expectedTags[] = "content";
            $expectedTags[] = "description";
            $expectedTags[] = "link";
            $otherTagSql = "";
            foreach($base->tagChildren as $tag) {
                if (!empty($tag->tagChildren)) {
                    if (in_array($tag->tagName, $expectedTags))
                            continue;
                    if (empty($path))
                        handleOtherTags($connection, $postID, $tag->tagName, $tag);
                    else
                        handleOtherTags($connection, $postID, $path . "." . $tag->tagName, $tag);
                } else {
                    if (!in_array($tag->tagName, $expectedTags)) {
                        if (empty($path))
                            $otherTagSql = "SELECT ID_Other_Tags_PK FROM Other_Tags WHERE ID_Post_FK = " . $postID . "
AND Name = '" . $tag->tagName ."'";
                        else
                            $otherTagSql = "SELECT ID_Other_Tags_PK FROM Other_Tags WHERE ID_Post_FK = " . $postID . "
AND Name = '" . $path . "." . $tag->tagName ."'";
//                          echo $otherTagSql . "<br><br>";
                        $otherTagData = doSQL($connection, $otherTagSql);
                        if (empty($otherTagData)) {
                            if (empty($path))
                                $otherTagSql = "INSERT INTO Other_Tags VALUES (0, " . $postID .
                                        ",'" . $tag->tagName . "','" . $tag->tagData ."');";
                            else
                                $otherTagSql = "INSERT INTO Other_Tags VALUES (0, " . $postID .
                                        ",'" . $path . "." . $tag->tagName . "','" . $tag->tagData ."');";
                        } else {
                            if (empty($path))
                                $otherTagSql = "UPDATE Other_Tags SET Value = " . $tag->tagData . " WHERE
ID_Other_Tags_PK = " . $otherTagData[0][0];
                            else
                                $otherTagSql = "UPDATE Other_Tags SET Value = " . $path . $tag->tagData . " WHERE
ID_Other_Tags_PK = " . $otherTagData[0][0];
                        }
                        doSQL($connection, $otherTagSql);
                    }
                }
            }
        }

        ?>
        <hr size="4"></hr>
    </body>
</html>
```

- XMLParser.php

```
<?php
/**
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU LesserGeneral Public License as published
```

```
	by the Free Software Foundation; either version 3 of the License, or
	(at your option) any later version.

	This program is distributed in the hope that it will be useful,
	but WITHOUT ANY WARRANTY; without even the implied warranty of
	MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
	GNU General Public License for more details.

	For support, please visit http://www.criticaldevelopment.net/xml/
*/

/**
 * XML Parser Class (php5)
 *
 * Parses an XML document into an object structure much like the SimpleXML extension.
 *
 * @author Adam A. Flynn <adamaflynn@criticaldevelopment.net>
 * @copyright Copyright (c) 2005-2007, Adam A. Flynn
 *
 * @version 1.3.0
 */
class XMLParser
{
	/**
	 * The XML parser
	 *
	 * @var resource
	 */
	private $parser;

	/**
	* The XML document
	*
	* @var string
	*/
	private $xml;

	/**
	* Document tag
	*
	* @var object
	*/
	public $document;

	/**
	* Current object depth
	*
	* @var array
	*/
	private $stack;

	/**
	 * Whether or not to replace dashes and colons in tag
	 * names with underscores.
	 *
	 * @var bool
	 */
	private $cleanTagNames;


	/**
	 * Constructor. Loads XML document.
	 *
	 * @param string $xml The string of the XML document
	 * @return XMLParser
	 */
	function __construct($xml = '', $cleanTagNames = true)
	{
		//Load XML document
		$this->xml = $xml;
```

```
		//Set stack to an array
		$this->stack = array();

		//Set whether or not to clean tag names
		$this->cleanTagNames = $cleanTagNames;
	}

	/**
	 * Initiates and runs PHP's XML parser
	 */
	public function Parse()
	{
		//Create the parser resource
		$this->parser = xml_parser_create();

		//Set the handlers
		xml_set_object($this->parser, $this);
		xml_set_element_handler($this->parser, 'StartElement', 'EndElement');
		xml_set_character_data_handler($this->parser, 'CharacterData');

		//Error handling
		if (!xml_parse($this->parser, $this->xml))
			$this->HandleError(xml_get_error_code($this->parser), xml_get_current_line_number($this->parser),
xml_get_current_column_number($this->parser));

		//Free the parser
		xml_parser_free($this->parser);
	}

	/**
	 * Handles an XML parsing error
	 *
	 * @param int $code XML Error Code
	 * @param int $line Line on which the error happened
	 * @param int $col Column on which the error happened
	 */
	private function HandleError($code, $line, $col)
	{
		trigger_error('XML Parsing Error at '.$line.':'.$col.'. Error '.$code.': '.xml_error_string($code));
	}


	/**
	 * Gets the XML output of the PHP structure within $this->document
	 *
	 * @return string
	 */
	public function GenerateXML()
	{
		return $this->document->GetXML();
	}

	/**
	 * Gets the reference to the current direct parent
	 *
	 * @return object
	 */
	private function GetStackLocation()
	{
		//Returns the reference to the current direct parent
		return end($this->stack);
	}

	/**
	 * Handler function for the start of a tag
	 *
	 * @param resource $parser
	 * @param string $name
	 * @param array $attrs
```

```
     */
    private function StartElement($parser, $name, $attrs = array())
    {
        //Make the name of the tag lower case
        $name = strtolower($name);

        //Check to see if tag is root-level
        if (count($this->stack) == 0)
        {
            //If so, set the document as the current tag
            $this->document = new XMLTag($name, $attrs);

            //And start out the stack with the document tag
            $this->stack = array(&$this->document);
        }
        //If it isn't root level, use the stack to find the parent
        else
        {
            //Get the reference to the current direct parent
            $parent = $this->GetStackLocation();

            $parent->AddChild($name, $attrs, count($this->stack), $this->cleanTagNames);

            //If the cleanTagName feature is on, clean the tag names
            if($this->cleanTagNames)
                $name = str_replace(array(':', '-'), '_', $name);

            //Update the stack
            $this->stack[] = end($parent->$name);
        }
    }

    /**
     * Handler function for the end of a tag
     *
     * @param resource $parser
     * @param string $name
     */
    private function EndElement($parser, $name)
    {
        //Update stack by removing the end value from it as the parent
        array_pop($this->stack);
    }

    /**
     * Handler function for the character data within a tag
     *
     * @param resource $parser
     * @param string $data
     */
    private function CharacterData($parser, $data)
    {
        //Get the reference to the current parent object
        $tag = $this->GetStackLocation();

        //Assign data to it
        $tag->tagData .= trim($data);
    }
}


/**
 * XML Tag Object (php5)
 *
 * This object stores all of the direct children of itself in the $children array. They are also stored by
 * type as arrays. So, if, for example, this tag had 2 <font> tags as children, there would be a class member
 * called $font created as an array. $font[0] would be the first font tag, and $font[1] would be the second.
 *
 * To loop through all of the direct children of this object, the $children member should be used.
 *
```

```
 * To loop through all of the direct children of a specific tag for this object, it is probably easier
 * to use the arrays of the specific tag names, as explained above.
 *
 * @author Adam A. Flynn <adamaflynn@criticaldevelopment.net>
 * @copyright Copyright (c) 2005-2007, Adam A. Flynn
 *
 * @version 1.3.0
 */
class XMLTag
{
	/**
	 * Array with the attributes of this XML tag
	 *
	 * @var array
	 */
	public $tagAttrs;

	/**
	 * The name of the tag
	 *
	 * @var string
	 */
	public $tagName;

	/**
	 * The data the tag contains
	 *
	 * So, if the tag doesn't contain child tags, and just contains a string, it would go here
	 *
	 * @var stat
	 */
	public $tagData;

	/**
	 * Array of references to the objects of all direct children of this XML object
	 *
	 * @var array
	 */
	public $tagChildren;

	/**
	 * The number of parents this XML object has (number of levels from this tag to the root tag)
	 *
	 * Used presently only to set the number of tabs when outputting XML
	 *
	 * @var int
	 */
	public $tagParents;

	/**
	 * Constructor, sets up all the default values
	 *
	 * @param string $name
	 * @param array $attrs
	 * @param int $parents
	 * @return XMLTag
	 */
	function __construct($name, $attrs = array(), $parents = 0)
	{
		//Make the keys of the attr array lower case, and store the value
		$this->tagAttrs = array_change_key_case($attrs, CASE_LOWER);

		//Make the name lower case and store the value
		$this->tagName = strtolower($name);

		//Set the number of parents
		$this->tagParents = $parents;

		//Set the types for children and data
		$this->tagChildren = array();
```

```
            $this->tagData = '';
        }

        /**
         * Adds a direct child to this object
         *
         * @param string $name
         * @param array $attrs
         * @param int $parents
         * @param bool $cleanTagName
         */
        public function AddChild($name, $attrs, $parents, $cleanTagName = true)
        {
            //If the tag is a reserved name, output an error
            if(in_array($name, array('tagChildren', 'tagAttrs', 'tagParents', 'tagData', 'tagName')))
            {
                trigger_error('You have used a reserved name as the name of an XML tag. Please consult the documentation
(http://www.criticaldevelopment.net/xml/) and rename the tag named "'.$name.'" to something other than a reserved name.',
E_USER_ERROR);

                return;
            }

            //Create the child object itself
            $child = new XMLTag($name, $attrs, $parents);

            //If the cleanTagName feature is on, replace colons and dashes with underscores
            if($cleanTagName)
                $name = str_replace(array(':', '-'), '_', $name);

            //Toss up a notice if someone's trying to to use a colon or dash in a tag name
            elseif(strstr($name, ':') || strstr($name, '-'))
                trigger_error('Your tag named "'.$name.'" contains either a dash or a colon. Neither of these characters are friendly with PHP
variable names, and, as such, you may have difficulty accessing them. You might want to think about enabling the cleanTagName feature
(pass true as the second argument of the XMLParser constructor). For more details, see http://www.criticaldevelopment.net/xml/',
E_USER_NOTICE);

            //If there is no array already set for the tag name being added,
            //create an empty array for it
            if(!isset($this->$name))
                $this->$name = array();

            //Add the reference of it to the end of an array member named for the tag's name
            $this->{$name}[] = &$child;

            //Add the reference to the children array member
            $this->tagChildren[] = &$child;

            //Return a reference to this object for the stack
            return $this;
        }

        /**
         * Returns the string of the XML document which would be generated from this object
         *
         * This function works recursively, so it gets the XML of itself and all of its children, which
         * in turn gets the XML of all their children, which in turn gets the XML of all thier children,
         * and so on. So, if you call GetXML from the document root object, it will return a string for
         * the XML of the entire document.
         *
         * This function does not, however, return a DTD or an XML version/encoding tag. That should be
         * handled by XMLParser::GetXML()
         *
         * @return string
         */
        public function GetXML()
        {
            //Start a new line, indent by the number indicated in $this->parents, add a <, and add the name of the tag
            $out = "\n".str_repeat("\t", $this->tagParents).'<'.$this->tagName;
```

```
        //For each attribute, add attr="value"
        foreach($this->tagAttrs as $attr => $value)
            $out .= ' '.$attr.'="'.$value.'"';

        //If there are no children and it contains no data, end it off with a />
        if(empty($this->tagChildren) && empty($this->tagData))
            $out .= " />";

        //Otherwise...
        else
        {
            //If there are children
            if(!empty($this->tagChildren))
            {
                //Close off the start tag
                $out .= '>';

                //For each child, call the GetXML function (this will ensure that all children are added recursively)
                foreach($this->tagChildren as $child)
                    $out .= $child->GetXML();

                //Add the newline and indentation to go along with the close tag
                $out .= "\n".str_repeat("\t", $this->tagParents);
            }

            //If there is data, close off the start tag and add the data
            elseif(!empty($this->tagData))
                $out .= '>'.$this->tagData;

            //Add the end tag
            $out .= '</'.$this->tagName.'>';
        }

        //Return the final output
        return $out;
    }

    /**
     * Deletes this tag's child with a name of $childName and an index
     * of $childIndex
     *
     * @param string $childName
     * @param int $childIndex
     */
    public function Delete($childName, $childIndex = 0)
    {
        //Delete all of the children of that child
        $this->{$childName}[$childIndex]->DeleteChildren();

        //Destroy the child's value
        $this->{$childName}[$childIndex] = null;

        //Remove the child's name from the named array
        unset($this->{$childName}[$childIndex]);

        //Loop through the tagChildren array and remove any null
        //values left behind from the above operation
        for($x = 0; $x < count($this->tagChildren); $x ++)
        {
            if(is_null($this->tagChildren[$x]))
                unset($this->tagChildren[$x]);
        }
    }

    /**
     * Removes all of the children of this tag in both name and value
     */
    private function DeleteChildren()
    {
        //Loop through all child tags
```

```
            for($x = 0; $x < count($this->tagChildren); $x ++)
            {
                //Do this recursively
                $this->tagChildren[$x]->DeleteChildren();

                //Delete the name and value
                $this->tagChildren[$x] = null;
                unset($this->tagChildren[$x]);
            }
        }
}
?>
```

- add_source_form.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>AW015 Back-End</title>
    </head>

    <body>
        <h1>P&aacute;gina de Configura&ccedil;&atilde;o</h1>
        <h2>Back-End - AW015</h2>
        <hr />
        <hr /><a href='index.php'>Voltar ao menu principal </a><br>
        <h1> Adicionar Fonte </h1>
        <form name="input" action='http://www.carlosalvares.net/aw/api.php?c=addSource' enctype="multipart/form-data"
method="post">
            <table align="left" cellpadding="2" cellspacing="2" border="0">
                <tr>
                    <td>
                        <input type="file" name="uploadedfile" value="Adicionar Fonte" />
                        <input type="submit" value="Send" />
                    </td>
                    </form>
                </tr>
            </table>
        </form>
    </body>
</html>
```

- adicionar_personalidade.php

```
<?php
function addPerson($nome) {
   require_once("mysql_conn.php");

   //colocar 1ª letra de cada nome em upper case
   $name = ucwords($nome);

   $query = "INSERT INTO Personalidades(ID_Personalidade_PK, Nome_Personalidade, Popularidade) VALUES (0,'".$name."',0)";

   $connection = connect();
   $results = doSQL($connection,$query);
   disconnect($connection);
   return $results;
}
$letra = $_POST["name"];
$res = addPerson($letra);

echo "<script type='text/javascript' src='./javascript/ajax.js'></script>
    <h1>P&aacute;gina de Configura&ccedil;&atilde;o</h1>
    <h2>Back-End - AW015</h2>
    <hr /><hr />
    <a href='index.php'>Voltar ao menu principal </a><br>
    <p>";
```

```
echo "<form name=\"input\" action=\"adicionar_personalidade.php\" method=\"post\">";
echo "<input type=\"text\" name=\"name\" />";
echo "<input type=\"submit\" value=\"Nome\" />";
echo"</form>";

for ($i=65;$i<=90;$i++) {
   $x = chr($i);
   echo "<a href=\"#\" onclick=\"runScript('gerir_personalidades2.php','?letra=".$x."','lista')\">".$x."</a> ";
}
echo "<br><br>";
echo "<div style:visibility='hidden' id='lista'>";
echo "</div>";

?>
```

- ajax_atribuir_params_pesquisa.php

```
<?php

$op = $_GET['op'];

$op($var);

function setState() {

      require_once("XMLParser.php");

      $id_font = $_GET['idfonte'];

      $id_tag = $_GET['idtag'];

      $file = file_get_contents('http://www.carlosalvares.net/aw/api.php?c=getSourceTags&id='.$id_font);

      $parser = new XMLParser($file);

      $parser->Parse();

      foreach($parser->document->channel[0]->item[0]->tagChildren as $element) {

            switch ($element->tagName) {

                  case "category":

                        if(strcmp($id_tag,$element->tagAttrs["id_tag_pk"])==0)

                              if(strcmp($element->tagAttrs["estado"],"ON")==0){

file_get_contents('http://www.carlosalvares.net/aw/api.php?c=setTagState&id_tag_pk='.$element->tagAttrs['id_tag_pk'].'&state=OFF');

                              }

                              else{

file_get_contents('http://www.carlosalvares.net/aw/api.php?c=setTagState&id_tag_pk='.$element->tagAttrs['id_tag_pk'].'&state=ON');
```

```
                }
            break;
        default:
            break;
    }//fim switch
}//fim foreach
}

function setValues() {
    require_once("XMLParser.php");
    $var = $_GET['id'];
    $value = $_GET['value'];
    $file = file_get_contents('http://www.carlosalvares.net/aw/api.php?c=setTagValue&id='.$var."&value=".$value);
}
?>
```

- alterar_params_pesquisa.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>AW015 Back-End</title>
    </head>

    <body>
        <h1>P&aacute;gina de Configura&ccedil;&atilde;o </h1>
        <h2>Back-End - AW015</h2>
        <hr />
        <hr><a href='index.php'>Voltar ao menu principal </a><br><hr>
        <div id="lista_dump"></div>
```

```
        <?php require_once("gerar_tabelas.php"); drawTables(); ?>

        <hr>

    </body>

</html>
```

- apagar_personalidade.php

```
<?php

function searchPersons($name) {

    require_once("mysql_conn.php");

    $query = "DELETE FROM Personalidades WHERE Nome_Personalidade = '".$name."' ;";

    $connection = connect();

    $results = doSQL($connection,$query);

    disconnect($connection);

    return $results;

}

$letra = $_GET['letra'];

$res = searchPersons($letra);

if($res)

    echo "Apagou ".$letra;

else

    echo "Não apagou ".$letra."!!";

?>
```

- api.php

```
<?php

//TODO Permitir varias operacoes no mesmo pedido.

header("Content-type: application/rss+xml; charset=utf-8");

require "XMLParser.php";

require "mysql_conn.php";

$command = $_GET["c"];
```

```
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";


switch($command) {
	case "getCelebrity":
		$description = "Celebrities available for searching";
		$root = createXMLRootTag($command, $description);
		getPersonalities($root->channel[0], $_GET["p"]);
		echo $root->GetXML();
		break;
	case "getSources":
		$description = "Sources available for searching";
		$root = createXMLRootTag($command, $description);
		getSources($root->channel[0], $_GET["s"]);
		echo $root->GetXML();
		break;
	case "getSourceState":
		$description = "Status of current available sources ";
		$root = createXMLRootTag($command, $description);
		getSourceState($root->channel[0], $_GET["s"]);
		echo $root->GetXML();
		break;
	case "getSourceTags":
		$description = "Tags available for source ".$source;
		$root = createXMLRootTag($command, $description);
		getSourceTags($root->channel[0], $_GET['s'], $_GET['id']);
		echo $root->GetXML();
		break;
	case "getData":
```

```
        $description = "Data available from sources about the celebrity";
        $root = createXMLRootTag($command, $description);
        getData($root->channel[0], $_GET["s"], $_GET["p"]);
        echo $root->GetXML();
        break;
case "getSourceData":
        $description = "Data available from source";
        $root = createXMLRootTag($command, $description);
        getSourceData($root->channel[0], $_GET["s"],$_GET["id"]);
        echo $root->GetXML();
        break;
case "addSource":
        $description = "Add Source";
        $root = createXMLRootTag($command, $description);
        $file = file_get_contents($_FILES['uploadedfile']['tmp_name']);
        //$file = $_POST["uploadedfile"];
        addSource($root->channel[0],$file);
        echo $root->GetXML();
        break;
case "addPersonality":
        $name = $_GET["p"];
        addPersonality($name);
        break;
case "setState":
        $description = "Status of current available sources ";
        $name = $_GET["s"];
        $state = $_GET["state"];
        if (isset($name) && isset($state)) {
```

```
            $root = createXMLRootTag($command, $description);
            setState($root->channel[0], $name, $state);
        }
        else {
            $command = "No source or state provided";
            $emptyattrs = array();
            $description = "Error in request";
            $root = createXMLRootTag($command, $description);
            $base = $root->channel[0];
            $base->AddChild("item", $emptyattrs, 2);
            $item = $base->item[0];
            $item->AddChild("title", $emptyattrs, 3);
            $item->title[0]->tagData = "No command sent or command unknown";
            $item->AddChild("link", $emptyattrs, 3);
            $item->link[0]->tagData = htmlspecialchars(curPageURL(true));
            $item->AddChild("description", $emptyattrs, 3);
            $item->description[0]->tagData = "The link used";
            echo $root->GetXML();
        }
        break;
    case "setTagState":
        $id = $_GET["id_tag_pk"];
        $state = $_GET["state"];
        $description = "Setting new Tag State";
        if (isset($id)&&isset($state)) {
            $root = createXMLRootTag($command, $description);
            setTagState($root->channel[0], $id,$state);
        }
```

```
else {
        $command = "No Tag or state provided";
        $emptyattrs = array();
        $description = "Error in request";
        $root = createXMLRootTag($command, $description);
        $base = $root->channel[0];
        $base->AddChild("item", $emptyattrs, 2);
        $item = $base->item[0];
        $item->AddChild("title", $emptyattrs, 3);
        $item->title[0]->tagData = "No command sent or command unknown";
        $item->AddChild("link", $emptyattrs, 3);
        $item->link[0]->tagData = htmlspecialchars(curPageURL(true));
        $item->AddChild("description", $emptyattrs, 3);
        $item->description[0]->tagData = "The link used";
        echo $root->GetXML();
}
break;
case "setTagValue":
        $id = $_GET["id"];
        $value = $_GET["value"];
        if (isset($id)&&isset($value)) {
                $root = createXMLRootTag($command, $description);
                setTagValue($root->channel[0], $id,$value);
        }
        else {
                $command = "No Tag or state provided";
                $emptyattrs = array();
                $description = "Error in request";
```

```
                $root = createXMLRootTag($command, $description);
                $base = $root->channel[0];
                $base->AddChild("item", $emptyattrs, 2);
                $item = $base->item[0];
                $item->AddChild("title", $emptyattrs, 3);
                $item->title[0]->tagData = "No command sent or command unknown";
                $item->AddChild("link", $emptyattrs, 3);
                $item->link[0]->tagData = htmlspecialchars(curPageURL(true));
                $item->AddChild("description", $emptyattrs, 3);
                $item->description[0]->tagData = "The link used";
                echo $root->GetXML();
            }
            break;
    default:
        if (!isset($_GET["c"]))
            $title = "no command sent";
        else
            $title = "command unknown";
        $details = "Error in request";
        $root = buildErrorXML($why,$details);
        echo $root->GetXML();
        break;
}

function getPersonalities(XMLTag $base, $name) {
    if (isset($name))
        $sql = "SELECT Nome_Personalidade FROM Personalidades WHERE Nome_Personalidade LIKE '" . $name . "%'";
    else
```

```
        $sql = "SELECT Nome_Personalidade FROM Personalidades";
    $connection = connect();
    $result = doSQL($connection, $sql);

    $emptyattrs = array();

    $index = 0;
    foreach($result as $personality) {
        $base->AddChild("item", $emptyattrs, 2);
        $item = $base->item[$index++];
        $item->AddChild("title", $emptyattrs, 3);
        $item->title[0]->tagData = $personality["Nome_Personalidade"];
        $item->AddChild("link", $emptyattrs, 3);
        $item->link[0]->tagData =
htmlspecialchars(curPageURL(false)."/aw/api.php?c=getData&p=".urlencode($personality["Nome_Personalidade"]));
        $item->AddChild("description", $emptyattrs, 3);
        $item->description[0]->tagData = "The link to get data about this celebrity";
    }
}

function getSources(XMLTag $base, $name) {
    if (isset($name))
        $sql = "SELECT Nome_Fonte FROM Fontes WHERE Nome_Fonte LIKE '" . $name . "%'";
    else
        $sql = "SELECT Nome_Fonte FROM Fontes";
    $connection = connect();
    $result = doSQL($connection, $sql);

    $emptyattrs = array();
```

```
	$index = 0;

	foreach($result as $personality) {

		$base->AddChild("item", $emptyattrs, 2);

		$item = $base->item[$index++];

		$item->AddChild("title", $emptyattrs, 3);

		$item->title[0]->tagData = $personality["Nome_Fonte"];

		$item->AddChild("link", $emptyattrs, 3);

		$item->link[0]->tagData =
htmlspecialchars(curPageURL(false)."/aw/api.php?c=getData&s=".urlencode($personality["Nome_Fonte"]));

		$item->AddChild("description", $emptyattrs, 3);

		$item->description[0]->tagData = "The link to get data from this source";

	}

}



function getSourceState(XMLTag $base, $name) {

	if (isset($name))

		$SQL = "SELECT * FROM Fontes WHERE Nome_Fonte LIKE '".$name."%'";

	else

		$SQL = "SELECT * FROM Fontes";



	$connection = connect();

	$result = doSQL($connection, $SQL);

	disconnect($connection);

	$emptyattrs = array();



	$index = 0;

	foreach($result as $source) {

		$base->AddChild("item", $emptyattrs, 2);
```

```
            $item = $base->item[$index++];
            $item->AddChild("title", $emptyattrs, 3);
            $item->title[0]->tagData = $source["Nome_Fonte"];
            $item->AddChild("link", $emptyattrs, 3);
            $item->AddChild("description", $emptyattrs, 3);
            $item->description[0]->tagData = $source['ESTADO'];
        }

}

function getSourceTags(XMLTag $base, $sourceName, $id) {
        if(!isset($sourceName) && isset($id))
            $sourceSQL = "SELECT * FROM Fontes WHERE ID_Fonte_PK = " .$id ;
        else if(isset($sourceName) && !isset($id))
                $sourceSQL = "SELECT * FROM Fontes WHERE Nome_Fonte LIKE '" .$sourceName. "%'";
            else
                $sourceSQL = "SELECT * FROM Fontes;";

        $connection = connect();
        $results = doSQL($connection,$sourceSQL);
        $index = 0;
        $emptyattrs = array();

        foreach($results as $r) {
            $tagsSQL = "SELECT * FROM Tags WHERE ID_FONTE_FK = ".$r['ID_Fonte_PK'].";";
            $results2 = doSQL($connection,$tagsSQL);

            $tagsArr = array();
```

```
            $description = "Tags and Values follow on categories ";


            $base->AddChild("item", $emptyattrs, 2);
            $item = $base->item[$index++];
            $item->AddChild("title", $emptyattrs, 3);
            $item->title[0]->tagData = $r["Nome_Fonte"];
            $item->AddChild("link", $emptyattrs, 3);
            $item->AddChild("description", $emptyattrs, 3);
            $item->description[0]->tagData = $description;
            $i = 0;
            foreach($results2 as $tag) {
                $tagsArr["id_tag_pk"] = $tag['ID_TAG_PK'];
                $tagsArr["domain"] = $tag['TAG'];
                $tagsArr["estado"] = $tag['ESTADO'];
                $item->AddChild("category",$tagsArr, 3);
                $item->category[$i]->tagData = $tag['VALOR'];
                $i++;
            }
        }
        disconnect($connection);
}


function getData(XMLTag $base, $source, $personality) {
        $connection = connect();
        if (isset($source) && isset($personality)) {
            $personalitySQL = "SELECT ID_Personalidade_PK FROM Personalidades WHERE Nome_Personalidade = '" . $personality .
"'";
            $sourceSQL = "SELECT ID_Fonte_PK FROM Fontes WHERE Nome_Fonte LIKE = '" . $source . "'";
            $personalityResult = doSQL($connection, $personalitySQL);
```

```
            $sourceResult = doSQL($connection, $sourceSQL);

            $dataSQL = "SELECT * FROM Posts WHERE ID_PERSONALIDADE_FK = " . $personalityResult[0][0] . " AND ID_FONTE
= " . $sourceResult[0]["ID_FONTE"];

            $dataResult = doSQL($connection, $dataSQL);

        } else if (isset($personality)) {

                $personalitySQL = "SELECT ID_Personalidade_PK FROM Personalidades WHERE Nome_Personalidade = '" .
$personality . "'";

                $personalityResult = doSQL($connection, $personalitySQL);

                $dataSQL = "SELECT * FROM Posts WHERE ID_PERSONALIDADE_FK = " . $personalityResult[0][0];

                $dataResult = doSQL($connection, $dataSQL);

            } else if (isset($source)) {

                    $sourceSQL = "SELECT ID_Fonte_PK FROM Fontes WHERE Nome_Fonte = '" . $source . "'";

                    $sourceResult = doSQL($connection, $sourceSQL);

                    $dataSQL = "SELECT * FROM Posts WHERE ID_Fonte = " . $sourceResult[0][0];

                    $dataResult = doSQL($connection, $dataSQL);

                } else {

                    $title = "No arguments specified";

                    $details = "This function needs arguments s or c to run";

                    buildErrorXML($why,$details);

                    return;

                }

        disconnect($connection);



        $emptyattrs = array();



        $index = 0;

        foreach($dataResult as $data) {

            $base->AddChild("item", $emptyattrs, 2);

            $item = $base->item[$index++];
```

```
            $item->AddChild("title", $emptyattrs, 3);
            $item->title[0]->tagData = $data["Conteudo"];
            $item->AddChild("link", $emptyattrs, 3);
            $item->link[0]->tagData = $data["Link"];
            $item->AddChild("description", $emptyattrs, 3);
            $item->description[0]->tagData = $data["Autor"];
            $item->AddChild("author", $emptyattrs, 3);
            $item->author[0]->tagData = $data["Autor"];
            $item->AddChild("pubDate", $emptyattrs, 3);
            $item->pubDate[0]->tagData = $data["Data_Pesquisa"];
        }
}

function getSourceData(XMLTag $base, $sourcename, $id) {
        if(!isset($sourcename) && isset($id))
            $sourceSQL = "SELECT * FROM Fontes WHERE ID_Fonte_PK = " .$id ;
        else if(isset($sourcename) && !isset($id))
                $sourceSQL = "SELECT * FROM Fontes WHERE Nome_Fonte LIKE '" .$sourcename. "%'";
            else
                $sourceSQL = "SELECT * FROM Fontes;";

        $connection = connect();
        $results = doSQL($connection,$sourceSQL);
        disconnect($connection);
        $index = 0;
        $emptyattrs = array();
        $arr = array();
```

```
	foreach($results as $r) {

		$arr["id_fonte_pk"]= "ID_Fonte_PK";

		$base->AddChild("item", $emptyattrs, 2);

		$item = $base->item[$index++];

		$item->AddChild("title", $emptyattrs, 3);

		$item->title[0]->tagData = $r["Nome_Fonte"];

		$item->AddChild("link", $emptyattrs, 3);

		$item->link[0]->tagData =
htmlspecialchars(curPageURL(false)."/aw/api.php?c=getSourceTags&s=".urlencode($r["Nome_Fonte"]));

		$item->AddChild("description", $emptyattrs, 3);

		$item->description[0]->tagData = "Link to Source Tags";

		$item->AddChild("category",$arr, 3);

		$item->category[0]->tagData = $r["ID_Fonte_PK"];

	}

}



function addSource(XMLTag $base, $xmlfile) {

	require_once("XMLParser.php");

	$index = 0;

	$emptyattrs = array();

	$parser = new XMLParser($xmlfile);

	$parser->Parse();

	$connection = connect();



	foreach($parser->document->addsource as $element) {

			$base->AddChild("item", $emptyattrs, 2);

			$item = $base->item[$index++];

			$SQL = "INSERT INTO Fontes
VALUES(0,'".$element->name[0]->tagData."','".$element->uri[0]->tagData."','OFF');";
```

```
                $results = doSQL($connection,$SQL);

                if ($results == 1){

                    $item->AddChild("title", $emptyattrs, 3);

                    $item->title[0]->tagData =$element->name[0]->tagData;

                    $item->AddChild("link", $emptyattrs, 3);

                    $item->AddChild("description", $emptyattrs, 3);

                    $item->description[0]->tagData = "Fonte adicionada com sucesso";

                }

                else{

                    $item->AddChild("title", $emptyattrs, 3);

                    $item->title[0]->tagData = $element->name[0]->tagData;

                    $item->AddChild("link", $emptyattrs, 3);

                    $item->AddChild("description", $emptyattrs, 3);

                    $item->description[0]->tagData = "Fonte não adicionada!";



                }


                $SQL2 = "SELECT ID_FONTE_PK FROM Fontes WHERE Nome_Fonte = '".$element->name[0]->tagData."';";

                $results2 = doSQL($connection,$SQL2);

                $id = $results2[0][0];


                if(isset($element->option))

                foreach($element->option as $option) {

                    $SQL = "INSERT INTO Tags VALUES
(0,'".$option->name[0]->tagData."','".$option->value[0]->tagData."',".$id.",'OFF');";

                    $results = doSQL($connection,$SQL);

                }

    }

    disconnect($connection);
```

```
}

function addPersonality($name) {
	$connection = connect();
	$query = "INSERT INTO Personalidades(ID_Personalidade_PK, Nome_Personalidade, Popularidade) VALUES (0,'".$name."',0)";
	$results = doSQL($connection,$query);
	disconnect($connection);
}
function setState(XMLTag $base, $name,$state) {
	$connection = connect();
	$SQL = "UPDATE Fontes SET Estado = '".$state."' WHERE Nome_Fonte = '".$name."';";
	$results = doSQL($connection,$SQL);
	disconnect($connection);
}

function setTagState(XMLTag $base, $id,$state) {
	$connection = connect();
	$SQL = "UPDATE Tags SET ESTADO = '".$state."' WHERE ID_TAG_PK = ".$id.";";
	$results = doSQL($connection,$SQL);
	disconnect($connection);
}
function setTagValue(XMLTag $base, $id,$value) {
	$connection = connect();
	$SQL = "UPDATE Tags SET VALOR = '".$value."' WHERE ID_TAG_PK = ".$id.";";
	$results = doSQL($connection,$SQL);
	disconnect($connection);
}
```

```
function createXMLRootTag($command, $description) {

	$emptyattrs = array();

	$attrs = array();

	$attrs["version"] = "2.0";

	$root = new XMLTag("rss", $attrs, 0);

	$root->AddChild("channel", $emptyattrs, 1);

	$channel = $root->channel[0];

	$channel->AddChild("title", $emptyattrs, 2);

	$channel->title[0]->tagData = "AW015:" . $command;

	$channel->AddChild("link", $emptyattrs, 2);

	$channel->link[0]->tagData = htmlspecialchars(curPageURL(true));

	$channel->AddChild("description", $emptyattrs, 2);

	$channel->description[0]->tagData = $description;

	$channel->AddChild("language", $emptyattrs, 2);

	$channel->language[0]->tagData = "pt-pt";

	$channel->AddChild("copyright", $emptyattrs, 2);

	$channel->copyright[0]->tagData = "AW015";

	//	$channel->AddChild("image", $emptyattrs, 2);

	//	$image = $channel->image[0];

	//	$image->AddChild("link", $emptyattrs, 3);

	//	$image->link[0]->tagData = "http://www.xml.com/";

	//	$image->AddChild("url", $emptyattrs, 3);

	//	$image->url[0]->tagData = "http://www.xml.com/universal/images/xml_tiny.gif";

	//	$image->AddChild("title", $emptyattrs, 3);

	//	$image->title[0]->tagData = "XML.com";


	return $root;

}
```

```
function curPageURL($fullurl) {

	$isHTTPS = (isset($_SERVER["HTTPS"]) && $_SERVER["HTTPS"] == "on");

	$port = (isset($_SERVER["SERVER_PORT"]) && ((!$isHTTPS && $_SERVER["SERVER_PORT"] != "80") || ($isHTTPS && $_SERVER["SERVER_PORT"] != "443")));

	$port = ($port) ? ':'.$_SERVER["SERVER_PORT"] : '';

	if ($fullurl) {

		$url = ($isHTTPS ? 'https://' : 'http://').$_SERVER["SERVER_NAME"].$port.$_SERVER["REQUEST_URI"];

	} else {

		$url = ($isHTTPS ? 'https://' : 'http://').$_SERVER["SERVER_NAME"].$port;

	}

	return $url;

}



function buildErrorXML($title, $details) {

	$emptyattrs = array();

	$root = createXMLRootTag("Error", $details);

	$base = $root->channel[0];

	$base->AddChild("item", $emptyattrs, 2);

	$item = $base->item[0];

	$item->AddChild("title", $emptyattrs, 3);

	$item->title[0]->tagData = $title;

	$item->AddChild("link", $emptyattrs, 3);

	$item->link[0]->tagData = htmlspecialchars(curPageURL(true));

	$item->AddChild("description", $emptyattrs, 3);

	$item->description[0]->tagData = $details;

	return $root;

}
```

```
    ?>
```

- content_fontes_opt.php

```
<?php
   date_default_timezone_set("Europe/Lisbon");
   echo "Fontes Disponiveis:<br><i>";
   require_once("XMLParser.php");
   $file = file_get_contents('http://www.carlosalvares.net/aw/api.php?c=getSources');
   $parser = new XMLParser($file);
   $parser->Parse();
      foreach($parser->document->channel[0]->tagChildren as $element) {

            switch ($element->tagName) {
                  case "item":
                        echo $element->title[0]->tagData."<br>";
                        break;
                  default:
                        break;
      }//fim switch
}//fim foreach
echo "</i>";
?>
```

- date_comparisson.php

```
<?php
function compare_dates($date1, $date2) {
            //se o mes da pesquisa é mais recente
            //descartar
            if($date2[1]<$date1[1]){
                  return 1;}
```

```
        else
                //mes e o mesmo ou anterior. vamos ver os dias.
                //se for mais antigo que 2 dias
                if($date2[2]<=($date1[2]-2)){
                        return 1;}
                else{
                        return 0;}
}
?>
```

- encontrar_fontes.php

```
<?php
$var = $_GET['funcao'];

if(strcmp($var, "search")==0)
   searchFonts();

if(strcmp($var, "switch")==0) {
   $var1 =$_GET['nome'];
   $var2 =$_GET['state'];
   switchFontState($var1,$var2);
}

function searchFonts() {
   require_once("mysql_conn.php");
   require_once("XMLParser.php");
   echo '<script type="text/javascript" src="./javascript/ajax.js"></script>';

   $file = file_get_contents('http://www.carlosalvares.net/aw/api.php?c=getSourceState');
```

```
    $parser = new XMLParser($file);

    $parser->Parse();



    foreach($parser->document->channel[0]->tagChildren as $element) {

        switch ($element->tagName) {

            case "item":

                echo $element->title[0]->tagData."<br> ";

                if(strcmp($element->description[0]->tagData, "ON") == 0) {

                    echo "<div style='color:#00FF00'>está <b>ligado</b>.    <button type='button' disabled='enabled'>Ligar</button><button
type='button'
onClick='runScript(\"encontrar_fontes.php\",\"?funcao=switch&nome=".$element->title[0]->tagData."&state=OFF\",\"lista\")'>Desligar</b
utton></div>";

                    echo "<br>";

                }

                else {

                    echo "<div style='color:#FF0000'>está <b>desligado</b>.    <button type='button'
onClick='runScript(\"encontrar_fontes.php\",\"?funcao=switch&nome=".$element->title[0]->tagData."&state=ON\",\"lista\")'>Ligar</butto
n><button type='button' disabled='enabled'>Desigar</button></div>";

                    echo "<br>";

                }

                break;

            default:

                break;

        }

    }

}



function switchFontState($fontname,$state) {

    require_once("XMLParser.php");

    echo '<script type="text/javascript" src="./javascript/ajax.js"></script>';
```

```
    $file = file_get_contents('http://www.carlosalvares.net/aw/api.php?c=getSourceState&s='.urlencode($fontname));
    $parser = new XMLParser($file);
    $parser->Parse();

    foreach($parser->document->channel[0]->tagChildren as $element) {
        switch ($element->tagName) {
            case "item":
                file_get_contents('http://www.carlosalvares.net/aw/api.php?c=setState&s='.urlencode($fontname).'&state='.urlencode($state));
                break;
            default:
                break;
          }
        }
        searchFonts();
}
?>
```

- gerir_fontes.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>AW015 Back-End</title>
    </head>
    <body>
        <script type="text/javascript" src="./javascript/ajax.js"></script>
        <h1>P&aacute;gina de Configura&ccedil;&atilde;o</h1>
        <h2>Back-End - AW015</h2>
        <hr />
        <hr><a href='gestor_fontes.php'>Voltar ao menu anterior </a><br><hr><p>
```

```
                <div style:visibility='hidden' id='lista'>
                    <?php
                    require_once("encontrar_fontes.php");
                    searchFonts();
                    ?>
                </div>

            </body>
        </html>
```

- gerir_personalidades.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>AW015 Back-End</title>
    </head>
    <body>
        <script type="text/javascript" src="./javascript/ajax.js"></script>
        <h1>P&aacute;gina de Configura&ccedil;&atilde;o</h1>
        <h2>Back-End - AW015</h2>
        <hr /><hr />
        <a href='index.php'>Voltar ao menu principal </a><br>
        <p>
            <?php
            echo "<form name=\"input\" action=\"adicionar_personalidade.php\" method=\"post\">";
            echo "<input type=\"text\" name=\"name\" />";
            echo "<input type=\"submit\" value=\"Nome\" />";
            echo"</form>";
```

```
        for ($i=65;$i<=90;$i++) {

            $x = chr($i);

            echo "<a href=\"#\" onclick=\"runScript('gerir_personalidades2.php','?letra=".$x."','lista')\">".$x."</a> ";

        }

        echo "<br><br>";



        ?>

      <div style:visibility='hidden' id='lista'>

      </div>

   </p>

</body>

</html>
```

- gerir_personalidades2.php

```
<?php

function searchPersons($letter) {

    require_once("mysql_conn.php");

    require_once("XMLParser.php");

    $file = file_get_contents('http://www.carlosalvares.net/aw/api.php?c=getCelebrity&p='.urlencode($letter));

    $parser = new XMLParser($file);

    $parser->Parse();

    foreach($parser->document->channel[0]->tagChildren as $element) {

        switch ($element->tagName) {

            case "item":

                echo "<a href='#'
onClick='runScript(\"apagar_personalidade.php\",\"?letra=".$element->title[0]->tagData."\",\"lista\")'>".$element->title[0]->tagData."</a>
";

                echo "<img src='./images/apagar.bmp' width='18' height='18' \n\ border='0' alt='Loading...'
onClick='runScript(\"apagar_personalidade.php\",\"?letra=".$element->title[0]->tagData."\",\"lista\")'>";

                echo "<br>";

                break;
```

```
                default:
                    break;
            }
        }
    }
    $letra = $_GET['letra'];
    searchPersons($letra);
    ?>
```

- gestor_fontes.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>AW015 Back-End</title>
</head>

<body>
<h1>P&aacute;gina de Configura&ccedil;&atilde;o </h1>
<h2>Back-End - AW015</h2>
<hr />
<hr><a href='index.php'>Voltar ao menu principal </a><br><hr>
<p>
  <label></label>
  <label></label>
</p>
<table width="600" border="1" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="164"><div align="center"><strong><a href="add_source_form.php">Adicionar Fonte</a></strong></div></td>
```

```
      <td width="164"><div align="center"><strong><a href="alterar_params_pesquisa.php" >Alterar Parâmetros de Acesso à Info.</a></strong></div></td>

      <td width="164"><div align="center">

        <p><strong><a href="gerir_fontes.php" >Activar/Desactivar Fonte</a></strong></p>

        </div></td>

   </tr>

</table>

<hr>

<div id="lista"></div>

<hr><br>

<?php require_once("content_fontes_opt.php");?>

</body>

</html>
```

- índex.php

```
      <html xmlns="http://www.w3.org/1999/xhtml">

      <head>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

      <title>AW015 Back-End</title>

      </head>

      <body>

      <h1>P&aacute;gina de Configura&ccedil;&atilde;o </h1>

      <h2>Back-End - AW015</h2>

      <hr /><hr />

      <p>

        <label></label>

        <label></label>

      </p>

      <table width="600" border="1" align="center" cellpadding="0" cellspacing="0">
```

```
    <tr>
      <td width="164"><p align="center"><strong><a href="SourceParser.php">Carregar
    Dados</a></strong></p></td>
      <td width="164"><div align="center"><strong><a href="gestor_fontes.php">Gerir Fontes</a></strong></div></td>
      <td width="164"><div align="center"><strong><a href="gerir_personalidades.php">Adicionar/Remover
  Personalidades</a></strong></div></td>
    </tr>
  </table>
  <p>Características obrigatórias do Back-end:</p>
  <ul>
    <li>Ser o mais independente possível das fontes de informação externas</li>
    <li>Ter uma interface Web (s/ autenticação) para configuração do sistema, que permita ao administrador:
      <ul>
        <li><s>inserir uma nova fonte de informação</s></li>
        <li><s>alterar parâmetros de acesso a uma fonte de informação</s></li>
        <li><s>activar/desactivar a recolha de informação de uma determinada fonte</s></li>
        <li><s>gestão da lista de personalidades</s></li>
      </ul>
    </li>
  </ul>
  <hr size="4"></hr>
  </body>
  </html>
```

- loader.php

```
<?php
  date_default_timezone_set("Europe/Lisbon");
//ligar à base dados
  require_once "mysql_conn.php";
```

```
$connection = connect();
$query = "SELECT * FROM Fontes ;";
$results = doSQL($connection,$query);
//commit e desligar
disconnect($connection);
foreach($results as $row){
    $nome = $row['Nome_Fonte'];
        $onoff = $row['ESTADO'];
        //executar
        if(strcmp("ON",$onoff) == 0){
            include "loader_".strtolower($nome).".php";
            echo "<hr size='4'><br>";
          }
      else{
                echo $nome." Desligado. <br>";
        echo "<hr size='4'><br>";
          }
        }
?>
```

- loader_opt.php

```
<?php
header("Content-type: text/html; charset=utf-8");
$html = "<h1>Página de Configuração</h1><h2>Back-End - AW015</h2><hr />";
echo $html;
echo "<hr><a href='index.php'>Voltar ao menu principal </a><br><hr>";
//guardar a data da actualizacao
$myFile = "updates-Log.txt";
$fh = fopen($myFile, 'w') or die("can't open file");
```

```
        date_default_timezone_set("Europe/Lisbon");
        $data_update = date("Y-m-d H:i:s");
        fwrite($fh, $data_update);
        fclose($fh);
        include "post_unloader.php";
        echo "<hr size='4'><br>";
        include "loader.php";
        echo "<hr size='4'><br>";
        ?>
```

- loader_twitter.php

```
<?php
    date_default_timezone_set("Europe/Lisbon");
//ligar à base dados
    require_once "mysql_conn.php";
    $connection = connect();
    $query = "SELECT * FROM Fontes WHERE Nome_Fonte = 'Twitter';";
    $results = doSQL($connection,$query);
    foreach($results as $r)
        $string = $r['ON_OFF'];
    echo "<h3>Twitter</h3><br>";
    if(strcmp("SIM",$string) == 0){
    //vars da pesquisa no twitter
    $options = "&rpp=2";                        //limitar o numero de posts a carregar
    //###########
    $query = "SELECT COUNT(*) FROM Personalidades;";
    $results2 = doSQL($connection,$query);
    foreach($results2 as $r2)
                $max = $r2['COUNT(*)'];
```

```
    $i = rand(1,$max);
    $min = min($i,($i+75)%$max);
    $max2 = max($i,($i+75)%$max);
    echo "<h3>Actualizar Posts de Personalidades ".$min." a ".$max2."</h3><br><br>";
//obter os nomes de todas as personalidades
    $query = "SELECT p.Nome_Personalidade, p.ID_Personalidade_PK FROM Personalidades p LIMIT ".$min.",".$max2.";";
    $results3 = doSQL($connection,$query);
    $i = $min;
foreach($results3 as $r3) {
    if ($i> $max2 )
        break;
//obter posts do twitter das personalidades
//              $nome = mb_convert_encoding($row['Nome_Personalidade'],"latin1","utf-8");
                $nome = $r3['Nome_Personalidade'];
                $id = $r3['ID_Personalidade_PK'];
                echo $nome." -> id: ".$id."<br>";
                //obter resultados em xml do twitter
                $resultados =    simplexml_load_file("http://search.twitter.com/search.atom?q=" .urlencode($nome).$options);
        foreach ($resultados->entry as $elemento){
        //Tratamento de dados
                    $id_personalidade_fk = $id;
            $id_fonte = 2; //twitter.
                    $data_post = str_replace("T"," ",$elemento->published);
                    $data_post = str_replace("Z","",$data_post);
            $c = $elemento->content;
                    $conteudo = mb_convert_encoding($c,"latin1","utf-8");
                    $temp = explode(" ",$elemento->author->name);
                    $autor = mb_convert_encoding($temp[0],"latin1","utf-8");
```

```
                $localizacao = "CORRIGIR";

                $data_pesquisa = date("Y-m-d H:i:s");



            //nao inserir repetidos com base na data e no autor.

            //como tal, vamos contar quantos posts existem nessas condicoes.

            $query = "SELECT COUNT(*) FROM Posts p WHERE p.ID_Personalidade_FK = ".$id_personalidade_fk." AND p.Autor =
'".$autor."' AND p.data_post = '".$data_post."';";

      $results4 = doSQL($connection,$query);

            //se o count der 0 entao o post n existe. inserir.

      foreach($results4 as $count)

        if($count['COUNT(*)'] == 0){

                        $query = "INSERT INTO Posts(ID_POST_PK, ID_PERSONALIDADE_FK, ID_FONTE, DATA_POST, CONTEUDO,
AUTOR, LOCALIZACAO, DATA_PESQUISA, NUM_VISUALIZACOES)
VALUES(0,".$id_personalidade_fk.",".$id_fonte.",'".$data_post."','".$conteudo."','".$autor."','".$localizacao."','".$data_pesquisa."',1);";

            $results5 = doSQL($connection,$query);

            echo " Novo Post Guardado!<br>";

              }//fim if

      }//fim foreach

    $i++;

}//fim while

}//fim if ligado.

  else

            echo "Twitter Desligado. <br>";

  //commit e desligar

  disconnect($connection);

?>
```

- loader_verbatim.php

```
<?php

  //filtrar um pouco o texto para obter registersearch content.

  $original= file_get_contents("http://verbatim.labs.sapo.pt/");
```

```
$nomes = strstr($original,"registerSearch");
$complete = substr($nomes,stripos($nomes,"##"),strrpos($nomes,"##"));
$lista = explode("split",$complete);
//obter a lista toda de personalidades
echo "<h3>Personalidades</h3><br>";
$personalidades = explode("##",$lista[0]);
unset($personalidades[0]);
$personalidades = array_values($personalidades);
array_pop($personalidades);
//inserir personalidades na base de dados.
require_once "mysql_conn.php";
$connection = connect();
$count = 0;
foreach ($personalidades as $p){
   $query = "INSERT INTO Personalidades(ID_Personalidade_PK, Nome_Personalidade, Popularidade) VALUES (0,'".$p."',0);";
        $results = doSQL($connection,$query);
        if ($results == 1)
            $count++;
}
   disconnect($connection);
        echo "Adicionadas ".$count." Personalidades<br>";
?>
```

- mysql_clear.php

```
<?php
   header("Content-type: text/html; charset=utf-8");
   require_once("mysql_conn.php");
   $connection = connect();
   $html = "<h1>Página de Configuração</h1><h2>Back-End - AW015</h2><hr />";
```

```
        echo $html;
        echo "<hr><a href='index.php'>Voltar ao menu principal </a><br><hr>";
           $queryfile = "clear_tables.sql";
           $contents = file_get_contents($queryfile);
           $statements = explode(";\n",$contents);
        foreach ($statements as $s){
            echo "Executar-> ".$s."-> ";
            $results = doSQL($connection,$s);
                 echo $results."<br>";
         }
        disconnect($connection);
      ?>
```

- mysql_conn.php

```
<?php
function connect() {
      $dbhost = 'localhost';
      $dbuser ='carlosal_awuser';
      $dbpass = 'aw2010';
      $dbname = 'carlosal_AW';
      $connection = mysql_pconnect($dbhost,$dbuser,$dbpass) or die ("error connecting to mysql");
      mysql_select_db($dbname) or die("can't switch to database");
      return $connection;
}

function doSQL($connection, $sql) {
      $result = array();
      if (!$connection) {
            die('invalid connection to database');
```

```
    }
    $query = mysql_query($sql);
    if(strstr($sql,"SELECT")) {
        while($row = mysql_fetch_array($query)) {
            $result[] = $row;
        }
        return $result;
    }
    else
        return $query;
}
function disconnect($connection) {
    mysql_close($connection);
}
?>
```

- post_unloader.php

```
<?php
    date_default_timezone_set("Europe/Lisbon");
    include "date_comparisson.php";
    //ligar à base dados
        include "mysql_conn.php";
    $connection = connect();
    $query = "SELECT * FROM Posts;";
    $results = doSQL($connection,$query);
    $count = 0;
    foreach($results as $row) {
        $id = $row['ID_POST_PK'];
        $data_pesquisa_temp = $row['DATA_PESQUISA'];
```

```
            $data_actual_temp = date("Y-m-d H:i:s");
             $data_pesquisa = substr($data_pesquisa_temp,0,10);
            $data_actual = substr($data_actual_temp,0,10);
             $data_act_array=explode("-",$data_actual);
             $data_pes_array=explode("-",$data_pesquisa);
             $resultado = compare_dates($data_act_array,$data_pes_array);
             // post mais antigo que 2 dias.
             //apagar
             if($resultado == 1){
                        $query = "DELETE FROM Posts WHERE ID_POST_PK = ".$id.";";
             doSQL($connection,$query);
                        $count++;
             }
       }//fim while
          //commit e desligar
          disconnect($connection);
          echo "<h3>Posts antigos</h3><br>";
          echo "Removidos ".$count." posts<br>";
       ?>
```

- template.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
       Document      : template.xml
       Created on : March 28, 2010, 11:50 AM
       Author         : XXXXXX
       Description:
            Purpose of the document follows.
-->
```

```
<root>
   <!--adicionar uma nova fonte de informacao-->
   <addsource>
         <name></name>
         <URI></URI>
         <option>
               <name></name>
               <value></value>
         </option>
         <authortag></authortag>
         <locationtag></locationtag>
         <datetag></datetag>
         <contenttag></contenttag>
   </addsource>
      <addsource>
         <name></name>
         <URI></URI>
         <option>
               <name></name>
               <value></value>
         </option>
         <authortag></authortag>
         <locationtag></locationtag>
         <datetag></datetag>
         <contenttag></contenttag>
   </addsource>
</root>
```

- ajax.js

```
var xmlhttp;
var element;

// Executa o script num dado url enviando pelo metodo GET os parametros data
// sendo o resultado da execução do script enviada para o htmlelement indicado
// como parametro.
function runScript(url, data, htmlelement)
{

   element = htmlelement;
   xmlhttp=getXmlHttpObject();
   if (xmlhttp==null)
   {
      alert ("Browser does not support HTTP Request");
      return;
   }
   if (data == null) {
      url+="&lixo=#";
   } else {
      url+=data;
   }

document.getElementById(element).innerHTML =
   "<img src=\"./images/loading.gif\" width=\"48\" height=\"48\" \n\
            border=\"0\" alt=\"Loading...\">";

   xmlhttp.onreadystatechange=stateChanged;
   xmlhttp.open("GET",url,true);
   xmlhttp.send(null);
}


function stateChanged()
{
      if (xmlhttp.readyState==4)
      {
            document.getElementById(element).innerHTML=xmlhttp.responseText;
      }
}

function getXmlHttpObject()
{
      if (window.XMLHttpRequest)
      {
            // code for IE7+, Firefox, Chrome, Opera, Safari
            return new XMLHttpRequest();
      }
      if (window.ActiveXObject)
      {
            // code for IE6, IE5
            return new ActiveXObject("Microsoft.XMLHTTP");
      }
      return null;
}
```