Processamento de Linguagens e Conhecimento (4º ano de Curso)

Trabalho Prático

Relatório de Desenvolvimento

Eduardo Cunha	Flávia Silva	José Lima
(71940)	(64303)	(72187)

29 de Janeiro de 2018

Resumo

Este trabalho prático tem como objetivo a aplicação da matéria lecionada nas UC'S de Processamento e Representação de Informação e Gramáticas para a Compreensão de Software.
Assim, é enunciado a criação de uma aplicação Web, de um diário digital, ou seja, uma plataforma que permita a partilha de interesses baseados, principalmente, numa linha temporal. Este controlo temporal permite a exposição de todas as atividades de um individuo cronologicamente. De modo a especificar corretamente o domínio da aplicação, recorremos ao auxílio de uma ontologia no domínio do EU, sendo especificada em *OntoDL*. Para o uso desta aplicação é obrigatório uma autenticação inicial, definida através do módulo *passport*, visto que, este programa foi implementado em *JavaScript* e foi usada a base de dados *MongoDB*, com auxílio do *object modelling Mongoose*.
A cada indivíduo estão associadas diferentes atividades, com por exemplo, registo desportivo ou até mesmo de locais visitados, sendo estas visíveis no frontend da aplicação(públicas), ou apenas para o próprio utilizador registado(privado).
A partilha nas redes sociais também é um dos requisitos deste projeto: no entanto, não foi implementada a operação de partilha no facebook ou no twitter para cada item criado.

Conteúdo

Capítulo 1

Introdução

Este trabalho prático enquadra-se nas unidades curriculares de Processamento e Representação de Informação e Gramáticas para a Compreensão de Software para o desenvolvimento de competências relativas a estas. Mais especificamente, este projeto tem como objetivo principal aumentar os conhecimentos das Gramáticas de Atributos, Ontologias e o desenvolvimento de aplicações web através do *Node.js*, nomeadamente, pela sua framework *expressJS*.
No presente relatório existe uma análise e especificação do problema, fazendo a sua descrição e especificação dos seus requisitos. Neste capítulo, está também exposta a ontologia criada de modo a especificar o domínio da aplicação. Posteriormente apresenta-se a conceção e o desenho da resolução do problema escolhido, apresentando os algoritmos produzidos. Por ultimo, será também apresentada a codificação e testes realizados e os seus resultados.
No final deste trabalho, conseguimos criar uma aplicação web com sucesso, com auxílio de uma base de dados não relacional, assim como definir uma ontologia que ilustra corretamente a estrutura das entidades do nosso projeto. No entanto, não cumprimos alguns dos requisitos, como por exemplo, a partilha das atividades nas redes sociais e a escolha de definir essas como privadas ou publicas, sendo que, cada utilizador só consegue ver as suas atividades se estiver registado na nossa aplicação. Deste modo, a confidencialidade dos dados é respeitada.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Neste projeto é enunciado a criação de um diário digital, ou seja, uma plataforma que permita a partilha de interesses baseados, principalmente, numa linha temporal. Linha esta que permite a exposição de todas as atividades de um individuo cronologicamente. Para o uso desta aplicação é obrigatório uma autenticação inicial, definida através do módulo *passport*, visto que, este programa foi implementado em *JavaScript* e foi usada a base de dados *MongoDB*, com auxílio do *object modelling Mongoose*.
A cada indivíduo estão associadas diferentes atividades, com por exemplo, registo desportivo ou até mesmo de locais visitados, sendo estas visíveis no frontend da aplicação(públicas), ou apenas para o próprio utilizador registado(privado). A partilha nas redes sociais também é um dos requisitos deste projeto: deverá existir uma operação de partilha no facebook ou no twitter para cada item criado.

2.2 Especificação do Requisitos

1. A aplicação é destinada a apenas um utilizador, ou seja, depois de instalada, o backend (BE) tratará apenas um "eu" individual. No entanto, o frontend (FE) de acesso à parte pública deverá estar disponível na Web para qualquer utilizador;

2. Para se poder distinguir entre um acesso público e um acesso privado no FE deverá haver um módulo de autenticação transversal à aplicação;

3. A aplicação terá como base o eixo temporal e a disposição cronológica de eventos e conteúdos será a sua base;

4. Um item pode ser público ou privado, por omissão deverá ser privado;

5. Se um item for privado apenas aparece no no frontend do seu criador, se for público deverá aparecer no frontend público da aplicação;

6. Pretende-se que haja uma ligação às redes sociais. Por exemplo, deverá haver uma operação para partilhar um item no facebook ou no twitter (eventualmente, com as devidas adaptações);

7. Um diário digital vai ser uma lista de itens publicados. Um item pode ser constituído por vários componentes: bloco de texto, fotografia, ficheiro associado, etc;

```
EU  -> Item+
Item -> ( data    tipo  local?   Elem+)
Elem  -> ficheiroAnexo  |  evento  | narracao | opiniao | ...
ficheiroAnexo -> foto | video | audio | pdf | ...
```

2.3 Ontologia

Com base no enunciado, definimos o que achamos representar uma correta ontologia que descreve o conjunto de conceitos dentro do domínio *EU* e os relacionamentos entre estes. Apesar de muitas das relações não terem sido implementadas na aplicação, achamos que estas fazem todo o sentido em serem descritas. A imagem da árvore da ontologia foi enviada juntamente com o resto do projeto, visto que, por questões de legibilidade, não foi colocada no relatório.

Listing 2.1: OntoDL

```
1  Ontologia EU
2  conceitos{ Pessoa[nome:string],
3                      Item[titulo:string, data:dateTime, descricao:string],
4                      Registo_Formacao,Local_Visitado,Registo_Desportivo,Receita_Culinaria,
                           Evento,
5                      Aniversario[aniversariante:string, data:dateTime, imagem:byte],
6                      Modalidade[nome:string],
7                      Futebol, Basquetebol, Andebol}
8
9  relacoes{tem, pratica, cria, subclasse}
10
11 triplos{Registo_Formacao=subclasse=>Item;
12                   Local_Visitado=subclasse=>Item;
13                   Registo_Desportivo=subclasse=>Item;
14                   Receita_Culinaria=subclasse=>Item;
15                   Evento=subclasse=>Item;
16                   Futebol=subclasse=>Modalidade;
17                   Basquetebol=subclasse=>Modalidade;
18                   Andebol=subclasse=>Modalidade;
19                   Pessoa=cria=>Item;
20                   Pessoa=pratica=>Modalidade;
21                   Pessoa=tem=>Aniversario;
22 }.
```

Capítulo 3

Conceção/desenho da Resolução

Esta aplicação funciona de modo semelhante a qualquer outra aplicação desenvolvida em Javascript/NodeJs. A partir da route inicial(index), foram definidos todos os paths da nossa aplicação, que, com o auxílio dos templates definidos em ficheiros pug, juntamente com a libraria *bootstrap*, melhorou bastante a apresentação do nosso projeto.
O nosso projeto está implementado de maneira muito simples e bastante semelhante ao implementado nas aulas. Da nossa *homepage* é possível fazer login ou o registo: em caso de tentativa de entrar na aplicação sem uma conta válida, será imprimida uma mensagem de erro. Após a validação da conta, o utilizador será reencaminhado para uma nova página onde aparecerá o seu "perfil". Este perfil consiste na exposição de todas as atividades inseridas previamente pelo cliente, ordenadas cronologicamente. Do lado esquerdo do ecrã existe uma barra que permite ao utilizador escolher que tipo de atividades quer ver, ou até mesmo inserir, ou seja, existe diferenciação entre o tipo de atividades existentes, e estas são filtradas de acordo com o tipo de cada uma. Umas das filtragens existentes consiste nas fotografias: as imagens inseridas em qualquer tipo de atividade são mostradas no separador "Fotografias". Deste modo, ao visualizar o tipo de atividades selecionado, o utilizador tem a possibilidade de registar um novo item desse mesmo tipo no fim da listagem de todas as atividades(desse mesmo tipo), já introduzidas. Por fim, o utilizador pode sair da nossa aplicação, o que permite que um novo utilizador entre/se registe. De salientar, o facto de a privacidade dos clientes ser totalmente respeitada, visto que, não é possível aceder a qualquer tipo de atividade introduzida, a não ser que esteja logado previamente.

Capítulo 4

Conclusão

Ao longo deste relatório está exposto todo o trabalho realizado para este projeto, assim como as decisões tomadas na implementação do código.
Consideramos que o frontend da aplicação está bastante positivo, enquanto que no backend existem várias falhas a apontar. Apesar de termos estruturado bastante bem como o nosso projeto ia ser criado, não conseguimos implementar, em código, grande parte das nossas ideias, devido à falta de conhecimento em *Javascript*. De facto, consideramos que a falta de experiência na criação de aplicações web foi crucial no desenvolvimento deste trabalho. A ontologia enunciada foi criada com sucesso em *OntoDL*, e está presente no capítulo 2 deste relatório.
Como trabalho futuro gostaríamos de melhorar o backend da nossa aplicação, nomeadamente, finalizar os requisitos enunciados, como a partilha nas redes sociais e a diferenciação entre atividades públicas e privadas. Um dos pontos a implementar, seria certamente, partilhar a aplicação na Internet, em vez de ser apenas local, o que seria um impulso na partilha de informação desta nas redes sociais. Também alteraríamos a definição dos elementos visuais de ficheiros pug para html puro, visto que, apesar da escrita em ficheiros pug ser mais prática, grande parte dos templates já existentes encontra-se em html. De referir também, que o pug é bastante sensível à identação, o que nos levou a perder um tempo considerável na sua correta identação. Apesar da nossa aplicação ser capaz de ter vários utilizadores registados nesta, gostaríamos que existisse algum tipo de interação entre estes, como por exemplo, a existência de grupos onde poderiam ocorrer partilha de atividades, como por exemplo, um grupo de culinária onde cada um partilhava as suas melhores receitas. Pensamos que os próprios itens definidos poderiam ter sido mais bem elaborados, como a implementação de ficheiros GPX associados a cada registo desportivo, ou até mesmo, outro tipo de registos.

Apêndice A

Código do Programa

Lista-se a seguir o código da ontologia que foi desenvolvida.

```
grammar onto;

@header{
import java.util.*;
}

@members{
class Triplo{
String pesssoa;
String relacao;
String objeto;

public Triplo(String s,String r,String o){
this.pesssoa=s;
this.relacao=r;
this.objeto=o;
}

public boolean equals(Triplo t){
return this.pesssoa.equals(t.pesssoa) &&
this.relacao.equals(t.relacao) && this.objeto.equals(t.objeto) ;
}

public String toString(){
return (this.pesssoa + "=" + this.relacao + "=>" + this.objeto);
}
}

class Atributo{
String conceito;
String nome;
String termo;

public Atributo(String c, String s,String t){
this.conceito=c;
this.nome=s;
this.termo=t;
```

```
}

}
}

ontologia returns[HashSet<String> conceits,boolean erroConceito, HashSet<String> relaces,
boolean erroRelacao, HashSet<Triplo> tripos,
boolean erroTriplos,HashSet<Atributo> atrDef, HashSet<Atributo> atrInst]
@after{
if($ontologia.erroTriplos==false && $ontologia.erroRelacao==false && $ontologia.erroConceito==false)
System.out.println("Ontologia Correcta");
else
System.out.println("Ontologia ERRADA");
}

: 'Ontologia' titulo=PAL conceitos{$ontologia.conceits=$conceitos.conceits;
$ontologia.erroConceito=$conceitos.erroConceito;
$ontologia.atrDef=$conceitos.defs;}
relacoes{$ontologia.relaces=$relacoes.relaces;
$ontologia.erroRelacao=$relacoes.erroRelacao;}
triplos[$ontologia.relaces, $ontologia.conceits]{$ontologia.tripos=$triplos.tripos;
$ontologia.erroTriplos=$triplos.erroTriplos;
$ontologia.atrInst=$triplos.atrInst;} '.'
;

conceitos returns[HashSet<String> conceits, boolean erroConceito, HashSet<Atributo> defs]
@init{$conceitos.erroConceito=false;
$conceitos.defs=new HashSet<Atributo>();}
: 'conceitos' '{' conceito[$conceitos.defs]{$conceitos.conceits=new HashSet<String>();
$conceitos.conceits.add($conceito.name);
$conceitos.erroConceito=false;
}
(','conceito[$conceitos.defs]{if($conceitos.conceits.contains($conceito.name)){
System.out.println("ERRO: " + $conceito.name + " conceito é repetido!");
$conceitos.erroConceito=true;
}
else $conceitos.conceits.add($conceito.name);}
)* '}'
;

conceito[HashSet<Atributo> hdef] returns[String name,HashSet<Atributo> definicoes]
: PAL{$conceito.name=$PAL.text;} (atributos[$conceito.name,$conceito.hdef]{
$conceito.definicoes = $atributos.definicoes;
})?
;

atributos[String n_conc, HashSet<Atributo> hdefinicoes]
returns[HashSet<Atributo> definicoes]
: ('['atributo':'termo{
$atributos.definicoes=$atributos.hdefinicoes;
$atributos.definicoes.add(new Atributo($atributos.n_conc,$atributo.name,$termo.name));
})
(','atributo':'termo{
$atributos.definicoes.add(new Atributo($atributos.n_conc,$atributo.name,$termo.name));
```

```
})*']'
;

atributo returns[String name]
:PAL {$atributo.name = $PAL.text;}
;

termo returns[String name]
: PAL {$termo.name = $PAL.text;}
;


relacoes returns[HashSet<String> relaces,boolean erroRelacao]
@init{$relacoes.erroRelacao = false;}
: 'relacoes' '{' relacao{$relacoes.relaces = new HashSet<String>();
$relacoes.relaces.add("tem");
$relacoes.relaces.add("pratica");
$relacoes.relaces.add("cria");
$relacoes.relaces.add($relacao.rel);
}
(',' relacao{if($relacoes.relaces.contains($relacao.rel)){
System.out.println("ERRO: " + $relacao.rel + " relação é reprida!");
$relacoes.erroRelacao = true;
}
else $relacoes.relaces.add($relacao.rel);})* '}'
;

relacao returns[String rel]
: PAL{$relacao.rel=$PAL.text;}
;

triplos[HashSet<String> relaces, HashSet<String> conceits]
returns[HashSet<Triplo> tripos,boolean erroTriplos, HashSet<Atributo> atrInst]
@init{$triplos.erroTriplos = false;
$triplos.atrInst=new HashSet<Atributo>();}
: 'triplos' '{' triplo[$triplos.relaces,$triplos.conceits, $triplos.atrInst]
{$triplos.tripos = new HashSet<Triplo>();
if(!($triplo.trip == null)) $triplos.tripos.add($triplo.trip);}
(';' triplo[$triplos.relaces,$triplos.conceits,$triplos.atrInst]
{if($triplos.tripos.contains($triplo.trip)){
System.out.println("ERRO: " + $triplo.trip.toString() + " triplo repetido! ");
$triplos.erroTriplos=true;
}
else if(!($triplo.trip == null)) $triplos.tripos.add($triplo.trip);
if($triplo.erroTrip == true) $triplos.erroTriplos=true;
})* ';'? '}'
;

triplo[HashSet<String> relaces, HashSet<String> conceits, HashSet<Atributo> hatrInst]
returns[Triplo trip, boolean erroTrip, HashSet<Atributo> atrInst]
@init{$triplo.erroTrip = false;}
: suj=PAL '=' rel=PAL '=>' obj=PAL  ('['(atr=PAL'='ter=PAL{
$triplo.atrInst = $triplo.hatrInst;
$triplo.atrInst.add(new Atributo($suj.text,$atr.text,$ter.text));
```

```
}
)(','atr=PAL'='ter=PAL{
$triplo.atrInst.add(new Atributo($suj.text,$atr.text,$ter.text));
})*']')?
{ if($triplo.relaces.contains($rel.text) == false||
($triplo.conceits.contains($suj.text) == false)||
($triplo.conceits.contains($obj.text) == false)){
$triplo.erroTrip = true;
if($triplo.relaces.contains($rel.text) == false)
System.out.println("Relação " + $rel.text + " não existe");
if($triplo.conceits.contains($suj.text) == false)
System.out.println("Erro: " +  $suj.text);
if($triplo.conceits.contains($obj.text) == false)
System.out.println("Predicado " + $obj.text + " não existe");
}
else if(($rel.text).equals("tem")){
$triplo.erroTrip = true;
}
else if(($rel.text.equals("pratica") == false &&
$rel.text.equals("tem") == false) && $triplo.conceits.contains($suj.text)==true) {
$triplo.erroTrip = true;
}
else if(($rel.text.equals("pratica") == false &&
$rel.text.equals("tem") == false) && $triplo.conceits.contains($obj.text)==true) {
$triplo.erroTrip = true;
}
else{
$triplo.trip = new Triplo($suj.text,$rel.text,$obj.text);
}
}
;

PAL: ([a-zA-Z][-_a-zA-Z0-9?]*) | ('\"' ~['\'']* '\"');
NUM: '-'?[0-9]+;
Sep: ('\r'? '\n' | ' ' | '\t')+ -> skip;
```