Processamento de Linguagens

Trabalho prático 3

Relatório de Desenvolvimento

Duarte Freitas	Ana Carvalho	Nadine Oliveira
(A63129)	(A61855)	(A75614)

12 de Junho de 2018

Conteúdo

Capítulo 1

Introdução

Este projeto insere-se no âmbito da disciplina de Processamento de Linguagens e tem como objetivo, desenvolver um processador de linguagem para descrever parte da rede semântica que suporta o Museu da Emigração, das Comunidades e da Luso-descendência. Para tal começou-se por definir uma linguagem que detém a informação a ser processada. Para efetuar o processamento dessa linguagem, foi criada uma gramática que representa a sintaxe e regras da linguagem criada, gramática esta que com recurso à ferramenta *Yacc* vai permitir analisar semanticamente e sintaticamente a linguagem desenvolvida. Em simultâneo, foi também desenvolvido o reconhecedor léxico para essa linguagem com recurso à ferramenta *Flex*. No problema, é ainda pedido que o conhecimento seja representado na forma de um grafo, sendo possível a navegação concetual sobre o mesmo. Para tal, foi gerado um documento *.dot*, onde se representaram os três nodos propostos, **emigrante**, **obra** e **evento** e as suas devidas relações. Por fim, foram geradas páginas html que permitiram a navegação concetual sobre o conhecimento representado.

Capítulo 2

Linguagem Desenvolvida

2.1 Estrutura geral

Na linguagem apresentada, foram definidas três entradas distintas, cada uma delas é iniciada pelo carater *'@'* seguido do tipo informação a ser descrita. A informação representa-se do modo *'Atributo : "Conhecimento"'*, e encontra-se protegida por chavetas, sendo cada emigrante/obra/evento, mapeado por um código apresentado no inicio de cada bloco de informação. Para efetuar a separação entre entradas, foi determinado que no fim da representação de um dado bloco de informação, deverá ter o carater *';'*.

- **Emigrante:**
  Na representação do conhecimento relativo ao emigrante, considerou-se como atributos de relevo o nome do emigrante (Nome), a data e cidade de nascimento (Cidade/Data_Nasc), o local e data para onde emigrou (Destino/Data_Em) e o local e data para onde regressou(Regresso/Data_Rg).
  Foi ainda representado, a lista de eventos e obras em que o mesmo participou/fez (IDEvento/IDObra), para tal, optou-se por definir uma "lista"de códigos de eventos e obras, códigos estes que são referentes a um evento/obra definidos na linguagem. Estes dois campos "IDEvento"/"IDObra", são opcionais, pois existe a possibilidade de haverem emigrantes que não participaram/fizeram nenhum evento/obra.
  Na representação das datas, optou-se por se considerar apenas o ano, pois, dado que os processos de emigração já ocorram noutro século, considerou-se que os dias e os meses não demonstravam grande interesse do ponto de vista cronológico.

```
@Emigrante{ "1"
 Nome : "Joaquim Antonio Marques"
 Data_Nasc : "1820"
 Cidade : "São Gens"
 Destino : "Rio"
 Data_Em : "1835"
 Regresso : "Fafe"
 Data_Rg : "1838"
 IDEvento : "e1" , "e2"
 IDObra : "o1"
```

```
}
;
```

- **Evento:**
  Para representar o conhecimento de um evento, considerou-se como atributos importantes o nome do evento (Nome), a descrição do mesmo (DescricaoE) e ainda a sua data de realização (Data).

```
   @Evento{ "e8"
 Nome : "Festa de inauguração"
 DescricaoE : "Festa de inauguração da Estatua dos Combatentes"
 Data : "1840"
}
;
```

- **Obra:**
  Por ultimo, para a representação do conhecimento da obra, considerou-se como atributos importantes o nome da obra (Nome), a descrição da mesma (DescricaoO) e ainda a sua data de realização (Data).

```
   @Obra{ "o2"
 Nome : "Hospital Fafe"
 DescricaoO : "Qualquer coisa"
 Data : "1837"
}
;
```

Capítulo 3

Gramática Tradutora

Depois de apresentada a linguagem especificada, foi necessário desenvolver uma gramática para representar a sintaxe da mesma.

- **Símbolos Terminais:**

```
STRING  ABREE ABREO FECHA NOME DT CD DST DE RG DTR IDE IDO DES
```

- **Símbolos Não Terminais:**

```
Programa Emigrantes Emigrante DadosP ProcE Evento Obra
Eventos Obras IDEvento IDObra IDEventos IDObras DescricaoE
Nome DataNasc Cidade Destino Data_Em NomeEvento NomeObra
DataEvento DataObra Codigo DtReg Regresso Evs Obs DescricaoO
```

- **Produções:**

```
\textcolor{red}{ PROGRAMA }: Emigrantes ';' Eventos ';' Obras ';'
          ;
 Emigrantes : Emigrante
            | Emigrantes ';' Emigrante
            ;
 Emigrante : ABRE Codigo DadosP ProcE Evs Obs FECHA
           ;
 DadosP : NOME ':' Nome DT ':' DataNasc CD ':' Cidade
        ;
 Evs : IDE ':' IDEventos
     |
     ;
 Obs : IDO ':' IDObras
```

```
    |
    ;
IDEventos : IDEvento
          | IDEventos ',' IDEvento
          ;
ProcE : DST ':' Destino DE ':' Data_Em RG ':' Regresso DTR ':' DtReg
      ;
Eventos : Evento
        | Eventos ';' Evento
        ;
Evento : ABREE Codigo NOME ':' NomeEvento DES ':' DescricaoE DE ':' DataEvento FECHA
       ;
Obras : Obra
      | Obras ';' Obra
      ;
Obra : ABREO Codigo NOME ':' NomeObra DES ':' DescricaoO DE ':' DataObra FECHA
     ;
Nome : STRING
Cidade : STRING
Destino : STRING
Data_Em : STRING
Regresso : STRING
DtReg : STRING
IDEvento : STRING
IDObra : STRING

NomeEvento : STRING
DescricaoE : STRING
DataEvento : STRING

NomeObra : STRING
DescricaoO : STRING
DataObra : STRING

Codigo : STRING
```

Capítulo 4

Especificação Flex

De forma a complementar a gramática criada, foi desenvolvido um reconhecedor léxico recorrendo à ferramenta flex, de forma a reconhecer todos os símbolos terminais da mesma.

```
%%
\;                          {return yytext[0];}
\,                          {return yytext[0];}
\:                          {return yytext[0];}
@Emigrante\{                {return ABRE;}
@Evento\{                   {return ABREE;}
@Obra\{                     {return ABREO;}
\}                          {return FECHA;}
(?i:Nome)                   {return NOME;}
(?i:Data_Nasc)              {return DT;}
(?i:Cidade)                 {return CD;}
(?i:Destino)                {return DST;}
(?i:Data_Em)                {return DE;}
(?i:Regresso)               {return RG;}
(?i:Data_Rg)                {return DTR;}
(?i:IDEvento)               {return IDE;}
(?i:IDObra)                 {return IDO;}
(?i:Descricao)              {return DES;}
(?i:Data)                   {return DE;}
\"[^"]*\"                   {yytext[strlen(yytext)-1]='\0';
                            yylval.s = strdup(yytext+1);
                            return STRING;}


.|\n {;}
%%
```

Para filtrar o conhecimento, é utilizada a expressão regular \"[^]*\", que vai apanhar o conteúdo dentro das aspas. Sempre que este conteúdo for apanhado, é utilizado um dos campos da estrutura *yylval*, para guardar o conhecimento.

Capítulo 5

Estruturação da informação

Depois de criada a gramática e o gerador léxico, foi introduzido na gramática um conjunto de ações que permitiram, à medida que se reduziam as produções, guardar a informação nas estruturas criadas.
Estas estruturas, serviram como apoio à geração do grafo, e das páginas html necessárias para a resolução do problema.

5.1 Estruturas

Nesta secção, vão ser apresentadas, as estruturas usadas para armazenar a informação, obtida pelas dadas produções. Toda a informação obtida, foi guardada em três *hashtables*, uma para os emigrantes, outra para os eventos e uma última para as obras. Para tal recorreu-se à biblioteca de C, *glib.h*. Estas *hastables*, foram mapeadas por código de emigrante/eventos/obras, respetivamente.

```
GHashTable* emigrantes
GHashTable* eventos;
GHashTable* obras;

Emi emi; char* cEm;
Eve eve; char* cEv;
Obr obr; char* cOb;
```

5.2 Obtenção da informação

A informação foi obtida adicionando ações semânticas as produções, assim, à medida que a informação é processada é guardada nas estruturas apresentadas, os dados respetivos.

```
Emigrante : ABRE Codigo DadosP ProcE Evs Obs FECHA {
                                                    cEm = strdup($2);
                                                    emi->chave = strdup($2);
                                                    Emi em = copiaEm(emi);
                                                    g_hash_table_insert(emigrantes,cEm,em);
```

```
                                                        emi->lEventos=NULL;
                                                        emi->lObras=NULL;}
                ;
Evento : ABREE Codigo NOME ':' NomeEvento DES ':' DescricaoE DE ':' DataEvento FECHA  {cEv = strdup($2);
                                                                                       Eve ev = copiaEv(eve)
                                                                                       g_hash_table_insert(e
       ;
Obra : ABREO Codigo NOME ':' NomeObra DES ':' DescricaoO DE ':' DataObra FECHA {cOb = strdup($2);
                                                                                Obr ob = copiaOb(obr);
                                                                                g_hash_table_insert(obras,cO
     ;
        Nome : STRING           {emi->nome=strdup($1);}
        DataNasc : STRING       {emi->dataN = atoi($1);}
        Cidade : STRING         {emi->cidade=strdup($1);}
        Destino : STRING        {emi->destino=strdup($1);}
        Data_Em : STRING        {emi->dataE=atoi($1);}
        Regresso : STRING       {emi->cidadeR=strdup($1);}
        DtReg : STRING          {emi->dataR=atoi($1);}
        IDEvento : STRING       {emi->lEventos = g_slist_append (emi->lEventos, $1);
                                 g_hash_table_insert(eventos,$1,NULL);}
        IDObra : STRING         {emi->lObras = g_slist_append (emi->lObras, $1);
                                 g_hash_table_insert(obras,$1,NULL);}

        NomeEvento : STRING     {eve->nome=strdup($1);}
        DescricaoE : STRING     {eve->descricao=strdup($1);}
        DataEvento : STRING     {eve->data=atoi($1);}

        NomeObra : STRING       {obr->nome=strdup($1);}
        DescricaoO : STRING     {obr->descricao=strdup($1);}
        DataObra : STRING       {obr->data=atoi($1);}
```

5.3 Validação Semântica

Após a informação estar toda guardada nas estruturas acima descritas, foi então necessário validar semanticamente a mesma, de forma a manter a linguagem apresentada coerente e correta. Para tal, verificou-se as questões de seguida apresentadas.

5.3.1 Chaves Sem Correspondência

Para cada emigrante, verificou-se se para cada chave de evento/obra explicitada, existia de facto a informação do evento/obra nas estruturas. Portanto, temos que, é possível existirem eventos/obras sem ”dono”, isto é, sem que nenhum emigrante esteja relacionado com as mesmas, mas não é possível um emigrante estar relacionado com um

evento/obra que não exista explicitado na linguagem.
De seguida é apresentado o excerto do código C que retrata esta situação.

```
GHashTableIter iter;
gpointer key, value;

g_hash_table_iter_init (&iter, obras);
while (g_hash_table_iter_next (&iter, &key, &value))
{
    if(value==NULL){
        printf("ERRO SEMANTICO!! -> Chave de obra sem correspondencia %s\n", (char*)key);
        return -1;
    }
}

g_hash_table_iter_init (&iter, eventos);
while (g_hash_table_iter_next (&iter, &key, &value))
{
    if(value==NULL){
        printf("ERRO SEMANTICO!! -> Chave de evento sem correspondencia %s\n", (char*)key);
        return -1;
    }
}
```

5.3.2 Valida Datas

Por fim, foi validado se a data de partida (data de emigração), era sempre posterior à data de regresso, visto que o oposto é fisicamente impossível.

```
GHashTableIter iter;
gpointer key, value;

g_hash_table_iter_init (&iter, emigrantes);
while (g_hash_table_iter_next (&iter, &key, &value))
{
    if(((Emi)value)->dataE>((Emi)value)->dataR){
        printf("ERRO!! -> Data de emigração %d maior que data de regresso %d %s\n", ((Emi)value)->dataE ,
        return -1;
    }
}
```

Capítulo 6

Geração do grafo e páginas HTML

Por último, foi gerado o grafo com recurso à ferramenta *GraphViz* e criadas as páginas html correspondentes, que permitiram a navegação concetual sobre o conhecimento descrito.

6.0.1 Geração do grafo

O grafo gerado possui os três nodos requisitados no enunciado do projeto, **Emigrante**, **Evento**, **Obra** e as suas respetivas relações, "um emigrante participou num evento", e "um emigrante fez uma obra".
Para além destes três nodos, foram adicionados mais três, **Localização** que está relacionado com o nodo Emigrante, e **Partida** e **Regresso**, que estão relacionados com o nodo **Localização**. Achou-se necessário adicionar estes dois novos nodos, pois concluiu-se que era importante o utilizador poder comparar a cidade donde um emigrante partiu, e a cidade para onde regressou anos mais tarde.

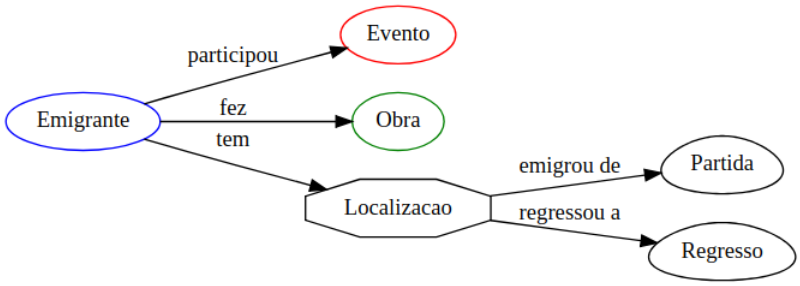

Figura 6.1: Grafo obtido

6.0.2 Geração páginas HTML

Por fim, foram geradas as páginas html contendo a informação armazenada nas estruturas, estas páginas foram hiperligadas aos nodos do grafo acima representado. Devido à hisperligações criadas ao longo das páginas html, é possível efetuar a navegação concetual pelo repositório de conhecimento.
De seguida é apresentado em pormenor uma navegação possível.

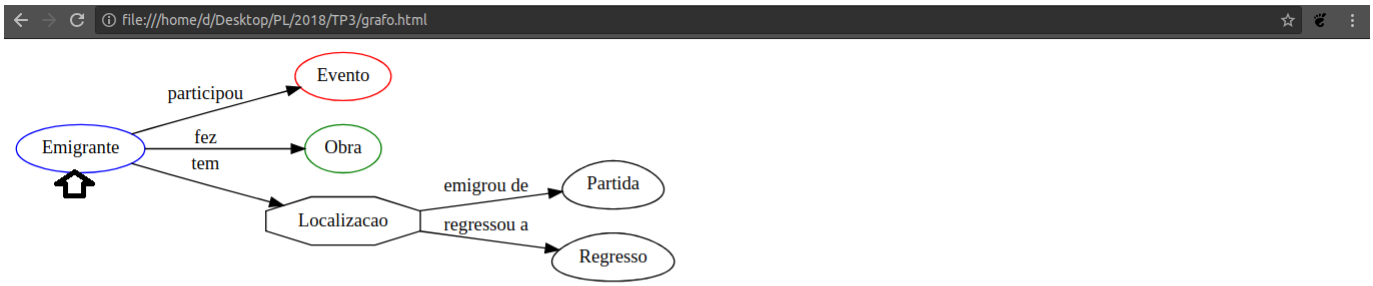

Figura 6.2: Grafo Html

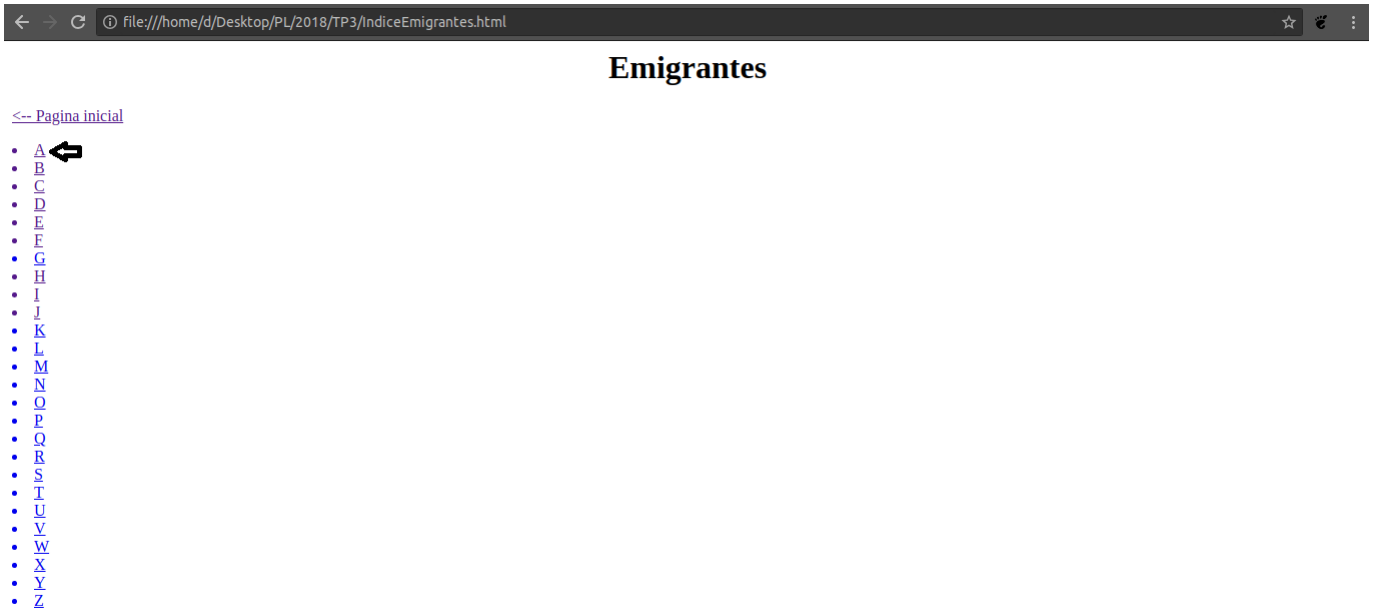

Figura 6.3: Indice Emigrantes

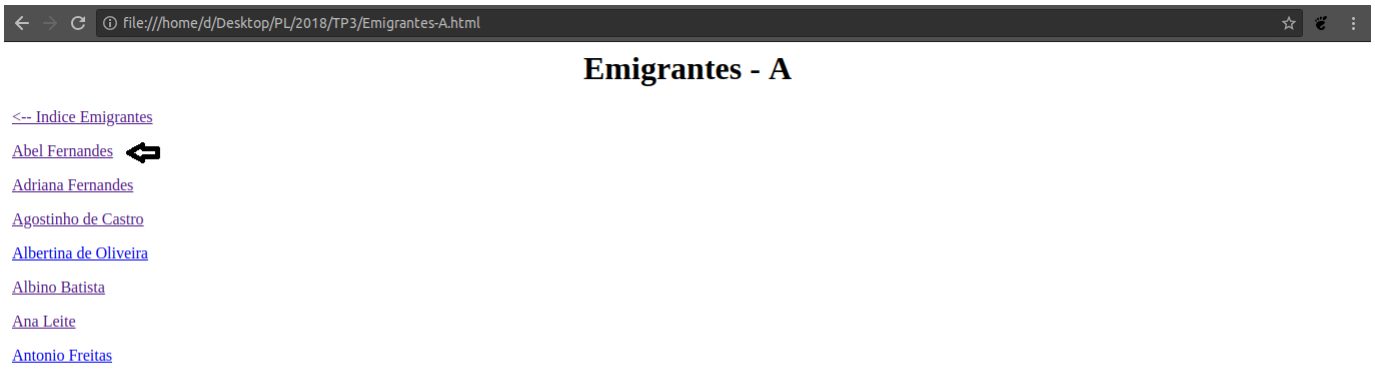

Figura 6.4: Emigrantes começados por A

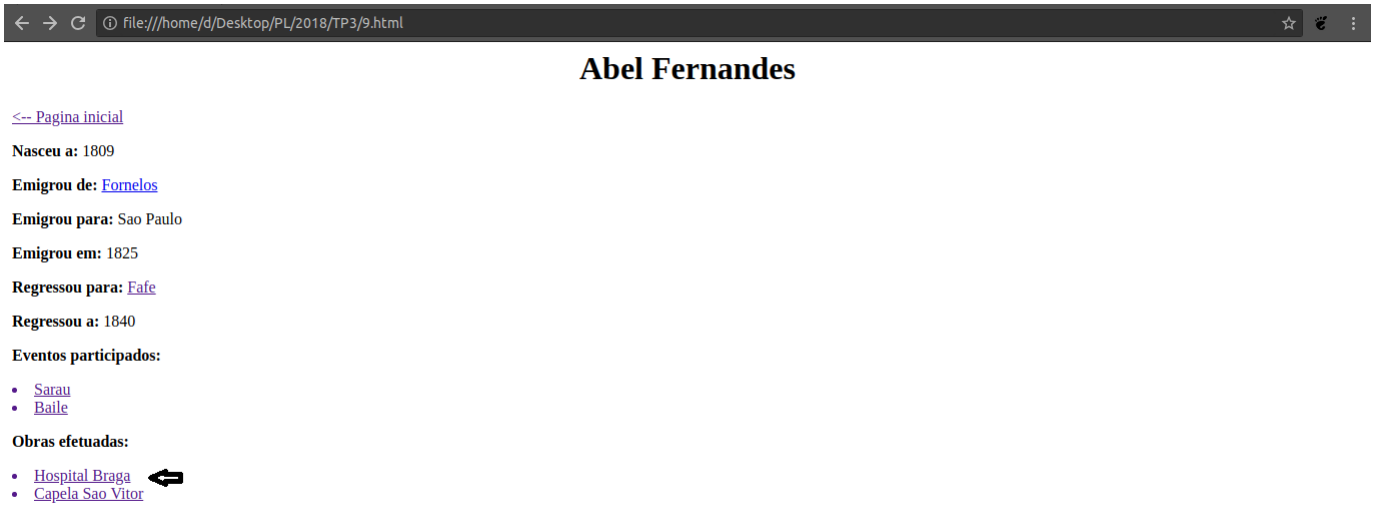

Figura 6.5: Exemplo Emigrante

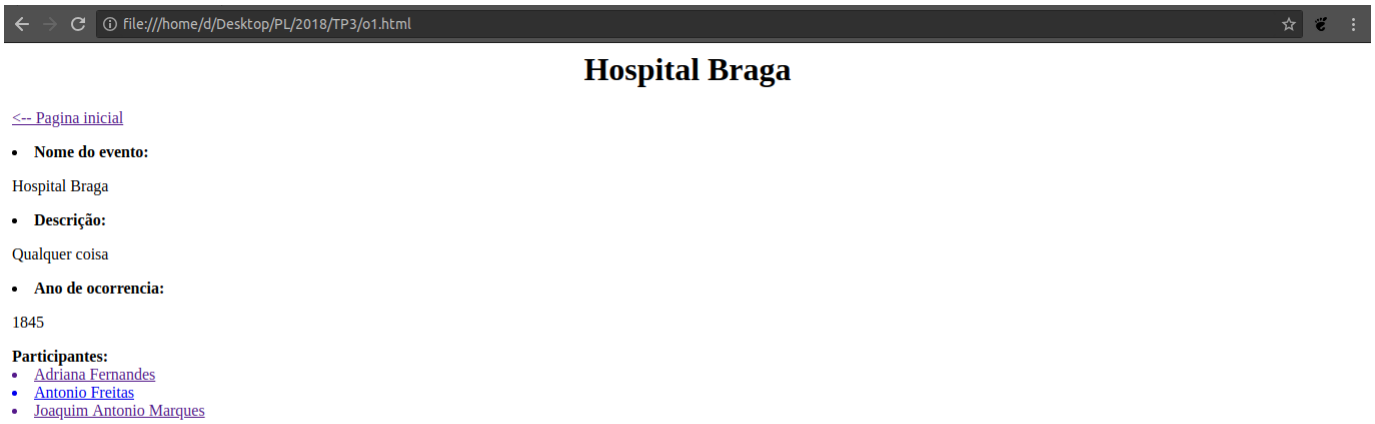

Figura 6.6: Exemplo Obra

Capítulo 7

Conclusão e trabalho futuro

O resultado obtido com a realização deste trabalho prático foi de acordo com aquilo que era pedido no enunciado. Foi desenvolvida uma linguagem que suportasse o conhecimento relativo ao Museu da Emigração, das Comunidades e da Luso-descendência, criada uma gramática tradutora que permitiu o processamento da mesma e a valida-se sintaticamente com recurso à ferramenta geradora *Yacc* e ainda um reconhecedor léxico que permitiu, filtrar apenas o conteúdo importante da linguagem descrita.
Por fim, o grafo e as páginas html foram geradas com sucesso, e a navegação concetual sobre o repositório de conhecimento estabelicida.
Como trabalho futuro, o repositório de conhecimento poderia ser extendido, e poderiam ainda ser criados mais nodos no grafo gerado, de forma a fornecer outras formas de navegar pelo conhecimento descrito.
Globalmente, o grupo faz uma apreciação positiva do trabalho realizado, todos os requisitos estabelecidos no enunciado foram cumpridos e ainda algumas funcionalidades extra, achadas no entender do grupo importantes.