Processamento de Linguagens (3o Ano de Mestrado Integrado em Engenharia Informática)

Trabalho Prático 3

Relatório de Desenvolvimento

Diogo André Teles Fernandes (A78867)

João Miguel Freitas Palmeira (A73864)

Luís Manuel Meruje Ferreira (A78607)

12 de Junho de 2018

Resumo

Este trabalho prático tem como principais objetivos aprofundar e aplicar conhecimentos obtidos durante as aulas teóricas e práticas de Processamento de Linguagens. Além de pretender aumentar a capacidade de escrever Expressções Regulares (ER) para descrição de padrões de frases e de, através destas, desenvolver Processadores de Linguagens Regulares que filtrem ou transformem textos, pretende também utilizar o sistema de produção para filtragem de texto em *Yacc*.
Aplicando o mesmo método de escolha do primeiro trabalho, selecionamos o primeiro enunciado disponível para este terceiro trabalho prático, mais concretamente, o referente a Rede Semântica do Museu da Emigração.
Os resultados obtidos estão de acordo com os objetivos previamente definidos e com o que foi pedido no enunciado.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Rede Semântica do Museu da Emigração

Área: Processamento de Linguagens

Neste trabalho prático, tivemos de criar uma rede semântica que suporta o museu de emigração, das comunidades e da luso-descendência. Devido às informações acerca dos emigrantes estarem incompletas e não haver nenhuma informação acerca do tipo de eventos que algum emigrante participou e as obras que fez, a informação acerca dos mesmos foi inventada por nós, produzindo um ficheiro de dados mais completo com a nossa própria linguagem. O grafo criado foi desenvolvido usando *WebDot* e foram usados ficheiros *css* e *scripts*, que podem ser encontrados em [1] [2], para tornar o *html* mais agradável ao olhar. O crédito para a criação desses recursos, e por consequência de grande parte da aparência das páginas *html* geradas, deverá ser concedido aos autores desses mesmos recursos.

Estrutura do Relatório

Este relatório está organizado em cinco capítulos, seguidos de anexos, sendo que nestes últimos é apresentado todo o código desenvolvido. No início do documento existe também um resumo do mesmo.
No capítulo 2, é inicialmente feita uma descrição informal do problema proposto, traçando as linhas gerais do mesmo e de seguida são especificados os requisitos concretos que devem ser cumpridos.
De seguida, o capítulo 3 aborda todas as opções tomadas ao longo da realização do trabalho prático, assim como as decisões que lideraram o desenho da solução e da sua implementação.
No quarto capítulo é explicado o porquê de termos optado por abordar cada alínea do modo que abordamos explicando as dificuldades e os problemas que ocorreram para a realização da mesma. Colocamos também ainda alguns testes realizados.
Por fim, o capítulo 5 inclui uma síntese do trabalho realizado com uma análise crítica dos resultados obtidos, possíveis melhorias e ainda trabalho futuro que poderia ser desenvolvido.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

2.1.1 Rede Semântica do Museu da Emigração

No projeto selecionado para o nosso grupo, temos de desenvolver uma rede semântica acerca das comunidades de emigrantes e gerar um grafo capaz de representar uma navegação conceptual na base do conhecimento. Esta rede deverá ser capaz de descrever três tipos de nodos: o emigrante, com referências a dados pessoais e processos de emigração, as obras que fez e por último os eventos em que participou. Estes dois últimos tipos devem ser ligados com o tipo emigrante com as relações fez e participou, que vão constituir os arcos do grafo. Para além disso, deve ser possível visualizar páginas HTML para os nodos descritos.

2.2 Especificação de Requisitos

Os requisitos fundamentais deste enunciado são:

- Recolher informação sobre os emigrantes, isto é, informações pessoais, obras que fez, eventos em que participou;
- Organizar essa informação num ficheiro de dados com a nossa linguagem;
- Elaborar uma gramática para o ficheiro de dados criado com a nossa linguagem;
- Elaborar o *parsing* de dados com o auxílio da gramática criada;

Capítulo 3

Conceção/desenho da Resolução

3.1 Ficheiro de Dados

Quanto a este tópico podemos afirmar que temos três tipos de entidades: Emigrante, Evento e Obra. Cada uma destas entidades é identificada pelo seu nome. Para além disso, dentro de cada entidade guardamos informação referente a certos tópicos de relevância, isto é, por exemplo, para Emigrante iremos ter o Nome, uma secção com os dados pessoais, uma secção com os dados de emigração, uma secção com as obras que fez e outra onde estarão os eventos onde participou, tal como podemos ver abaixo:

```
Emigrante{
    Nome:
    Dados pessoais->
        [Descrição do campo] = [Valor do campo];
        ...
    <-
    Dados de emigracao->
        [Descrição do campo] = [Valor do campo];
        ...
    <-
    Fez->
        [Nome que identifica Obra];
        ...
    <-
    Participou->
        [Nome que identifica evento];
        ...
    <-
}
```

Para a informação das obras, o ficheiro de dados possui a seguinte estrutura:

```
Obra{
    Nome: ...;
    Detalhes->
        [Descrição do campo] = [Valor do campo];
    <-
}
```

Finalmente, as informações sobre os eventos vêm no ficheiro de dados com a estrutura observada a seguir:

```
Evento{
```

```
    Nome: ...;
    Detalhes->
        [Descrição do campo] = [Valor do campo];
        ...
    <-
}
```

De seguida são descritos alguns detalhes relevantes. Todas as secções informativas descritas para uma dada entidade devem estar presentes na descrição da mesma, e devem surgir também na mesma ordem que a mostrada, tal como o campo do nome. Nas secções em que é possível ver a descrição "[Descrição do campo] = [Valor do campo];"podem ser introduzidos nos campos delimitados por parênteses retos quaisquer valores que sejam identificados como "VAR"(ver descrição do ficheiro FLEX mais à frente). Qualquer secção informativa tem que existir, mas pode estar vazia. Alguns caracteres estão reservados e servem para delimitar as várias zonas de cada entidade e as várias entidades entre si.

3.2 Gramática Utilizada

```
grafo: nodos

nodos: EMIGRANTE '{' emigrante '}' nodos
      | OBRA '{' obra '}' nodos
      | EVENTO '{' evento '}' nodos
      |

emigrante: NOME ':' VAR ';' dadosPessoais dadosEmigracao fez participou

dadosPessoais: DADOSPESSOAIS '-' '>' elementos '<' '-'

dadosEmigracao: DADOSDEEMIGRACAO '-' '>' elementos '<' '-'

fez: FEZ '-' '>' relacoesFez '<' '-'

participou: PARTICIPOU '-' '>' relacoesParticipou '<' '-'

relacoesFez: VAR ';' relacoesFez
           |
relacoesParticipou: VAR ';' relacoesParticipou
                  |
obra: NOME ':' VAR ';' detalhes

evento: NOME ':' VAR ';' detalhes

detalhes: DETALHES '-' '>' elementos '<' '-'

elementos: VAR '=' VAR ';' elementos
         |
```

De modo a captarmos as informações precisas para a criação da rede semântica de emigração, decidimos criar a gramática apresentada acima que descreve a linguagem que utilizamos. O elemento da raiz da gramática é o "grafo", que por sua vez é composto por "nodos". Um elemento "nodos"é composto por um dos 3 tipos de entidades e outro elemento "nodos", ou então pelo vazio.
Um Emigrante é composto pelo seu nome (conjunto de símbolos terminais) e pelas ações semânticas: dadosPessoais, dadosEmigracao, fez e participou. A ação semântica dadosPessoais é composta pela ação semântica elementos encapsulada dentro de uma zona delimitada da forma que se pode ver acima. A ação elementos é composta por duas *strings* separadas por um '='. A captação no comando "dadosEmigracao"é feito de modo similar, quando é detetado o *token*

DADOSDEEMIGRACAO.
Por outro lado a captação das obras que fez e os eventos em que participou são feitos respetivamente pelas ações semânticas "fez"e "participou". Estas funcionam de maneira similar, mudando apenas o *token* detetado no FLEX : "TOKEN − > dados < −". Estas ações são descritas por um conjunto de linhas (que são uma *string* cada uma) .
Para as obras e os eventos do ficheiro, a gramática funciona de forma similar: os detalhes são um conjunto de linhas ("elementos") que compõem os detalhes da uma dada obra ou evento.
"DETALHES − > elementos < −". De igual modo, este elementos é mesma ação semântica utilizada para obter dados pessoais e de emigração.

Do lado do FLEX, temos vários *tokens* que produzimos, dependendo do que for encontrado no *parsing* do ficheiro. Respetivamente:

- Emigrante

  devolve o *token* EMIGRANTE.

- Nome

  devolve o *token* NOME.

- Dados Pessoais

  devolve o *token* DADOSPESSOAIS.

- Dados de Emigração

  devolve o *token* DADOSDEEMIGRACAO.

- Fez

  devolve o *token* FEZ.

- Participou

  devolve o *token* PARTICIPOU.

- Obra

  devolve o *token* OBRA.

- Evento

  devolve o *token* EVENTO.

- Detalhes

  devolve o *token* DETALHES.

- Qualquer string que comece por letra ou número

  devolve o *token* VAR após duplicar a *string* do yytext para o s.

- Caracteres Especiais

  devolve yytext[0].

- Mudanças de Linha ou *tabs*

  Ignora, ou seja não faz nada.

É de notar que espaços, tabs e mudanças de linha são ignorados, à exceção dos espaços dentro de "VAR"'s.

3.3 Estrutura de dados

Definimos esta estrutura de dados de modo a ser possível guardar os dados do emigrante, das obras e dos eventos que estão contidos no ficheiro de dados bem como as relações que existem entre esses termos, para que seja possível criar os ficheiros *html* referentes à entidade emigrante.

```
char * emigranteAtual;
char * htmlToWrite;
char * htmlToWriteDadosPessoais;
char * htmlToWriteDadosEmigracao;
char * htmlToWriteFez;
char * htmlToWriteParticipou;
char * relacoesEncontradasFez;
char * relacoesEncontradasParticipou;
```

Assim, enquanto que estas variáveis são usadas para auxiliar a criação de ficheiros *html* para emigrantes, o grafo e os ficheiros *html* das restantes entidades são construídos utilizando exclusivamente as variáveis "$$"e "$n"das várias ações semânticas e encadeando os resultados dessas mesmas ações. Nas variáveis htmlToWrite é guardado o código *html* para cada emigrante, à medida que cada um destes é percorrido. As variáveis relacoesEncontradas* guardam as relações que foram detetadas nas secções "Fez"e "Participou"de cada emigrante. Uma vez que cada uma destas variáveis é uma única string, as relações são separadas dentro desta string pelo carácter ";". No final da análise de um emigrante, itera-se sobre as várias relações presentes nestas strings e estas relações são registadas no grafo produzido. Para além disso, é também escrito o ficheiro HTML referente ao emigrante.

3.4 Grafo e ficheiros *html* resultantes

Como resultado final iremos obter um grafo em formato *svg* e um ficheiro *html* por cada entidade especificada no ficheiro de input.
Eventos são identificados por nodos amarelos, obras por nodos verdes e emigrantes por nodos azul claro. Obras e eventos que surgem apenas referidos dentro de emigrantes (indicados como estando relacionados com estes), mas que não estão identificados de forma individual (por uma ação semântica evento ou obra) não têm ficheiro *html* e são identificados por nodos brancos.
Os arcos têm um etiqueta que identifica a relação que existe entre cada dois nodos.
Os nodos que possuem página *html* podem ser "clicados"para aceder a essa página. Dentro da página de um emigrante, todas as obras e eventos associados a esse emigrante surgem sob a forma de um link para a página dessa obra/evento (página essa que poderá existir ou não). No caso de estar identificado o Marido/Esposa de um emigrante nos seus dados pessoais, esse(a) surgirá também como um link para a página dessa pessoa, que uma vez mais poderá ou não existir, dependendo se existe um nodo correspondente para essa pessoa no grafo.

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

Como formato de *output* escolhido utilizamos o *svg*, visto que este pode ser usado em *browsers* e como vamos produzir ficheiros *html*, aproveitamos e utilizamos o *browser* para verificar os *outputs* obtidos tanto do *html* como do grafo. Além disso, decidimos imprimir verticalmente os grafos para uma melhor e mais natural apresentação da rede.
A nível de dificuldades encontradas, a principal terá sido arranjar uma estrutura paralela de variáveis que permitisse guardar os dados necessários à criação dos ficheiros html de emigrantes.

4.2 Testes realizados e Resultados

4.2.1 Ficheiro de Exemplo Utilizado nos Testes

Um excerto do ficheiro de *input* que serviu para testes:

```
Emigrante{
    Nome: Antonio Mendes;
    Dados pessoais->
        Idade = 15;
        Data de Nascimento = 20/1/1820;
        Sexo = Masculino;
        Lugar = Pardais;
        Freguesia = Sao Sebastiao;
        Concelho = Fafe;
        Estado Civil = Solteiro;
        Profissao = Trolha;
        Nome do Pai = Ramiro Silvino Mendes;
        Nome da Mae = Maria da Conceicao;
        Engajado = Sim;
        OBS = a avo paterna e natural de S. Vicente, Braga;
    <-
    Dados de emigracao->
        Destino = Rio de Janeiro;
        Data de Saida = 23/7/1835;
    <-
    Fez->
    Ponte de S. Joao;
    <-
    Participou->
```

```
        Musical da D. Josefa;
        Opera S. Luis;
<-
}

(...)

Obra{
    Nome: Portugal Renascentista;
    Detalhes->
        Ano = 1856;
        Editora = Nova Brasil;
        Tema = documentario sobre o Portugal Renascentista;
    <-
}

(...)

Evento{
    Nome : Sarau Museu Sul Americano do Rio de Janeiro;
    Detalhes->
        Ano = 1850;
        Local = Teatro S. Jorge;
        Duracao = cinco horas;
        Tema = O Novo Brasil;
        Presenca = D. Pedro 2;
        Convidado Especial = Manuel da Silva Araujo Ferreira;
    <-
}

(...)
```

Mostram-se a seguir alguns testes feitos e os respetivos resultados obtidos:

4.2.2 Rede Semântica do Museu da Emigração (grafo)

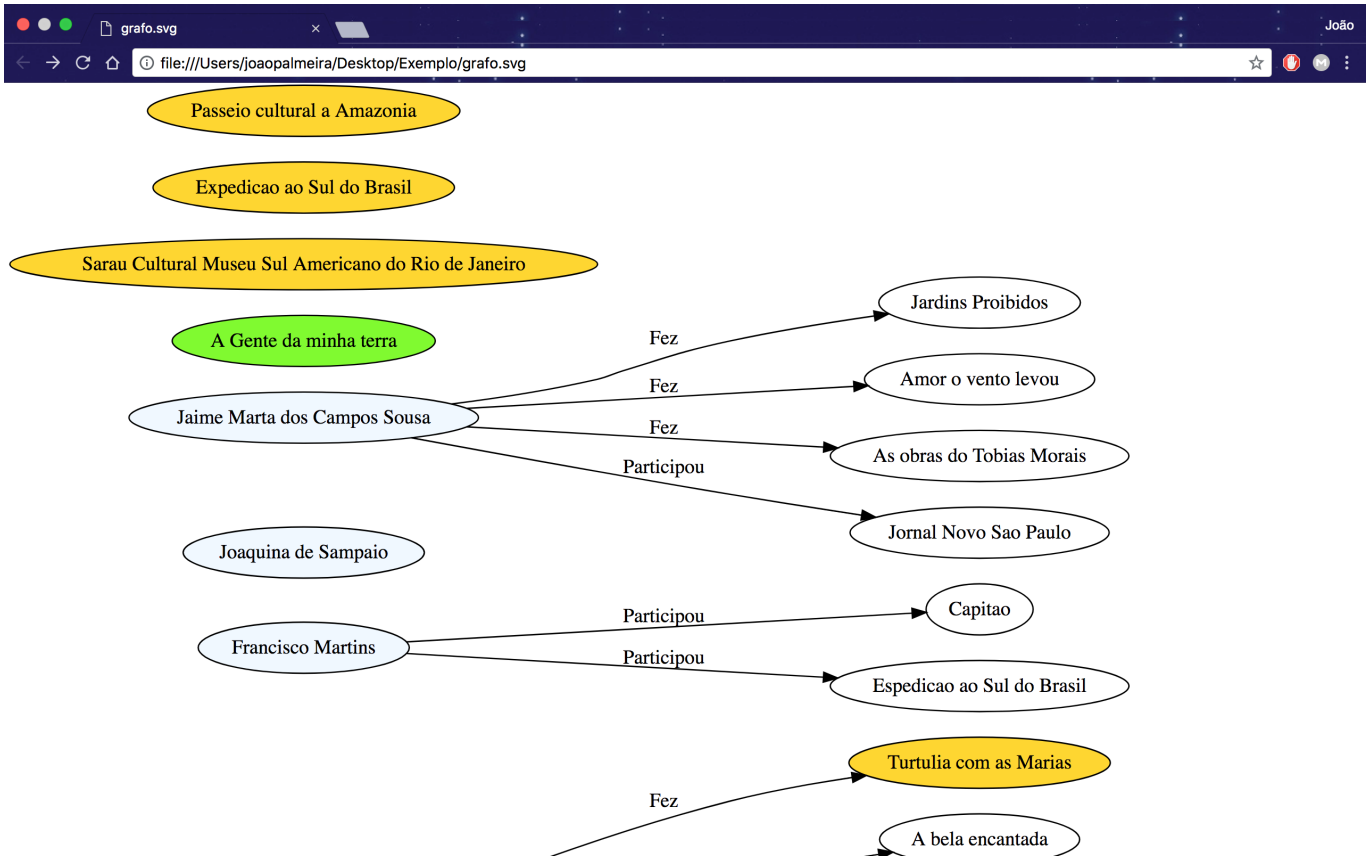

Figura 4.1: Página principal - *grafo.svg*

Figura 4.2: Exemplo da página de um emigrante

Figura 4.3: Exemplo da página dados pessoais de emigrante

Figura 4.4: Exemplo da página dados de emigração do emigrante

Figura 4.5: Exemplo da página de obras do emigrante

Figura 4.6: Exemplo da página de eventos do emigrante

Figura 4.7: Exemplo de página de um evento

Capítulo 5

Conclusão

Em suma, este projeto permitiu-nos aprofundar os nossos conhecimentos na área de Processamento de Linguagens em relação aos tópicos lecionados durante as aulas, tendo desta vez o foco sido direcionado para a ferramenta de processamento de texto Yacc embora também tenhamos utilizado o FLEX mais uma vez.
Julgamos ter conseguido cumprir os objetivos definidos no enunciado de forma satisfatória.
No entanto, como sugestão para trabalho futuro pretendemos implementar o uso de caracteres com acentos e cedilhas para não tornar o ficheiro de input tão restrito.

Apêndice A

Código do Programa

Mostra-se a seguir o código desenvolvido para este trabalho prático e que está explicado na secção de Algoritmos do capítulo 3.

A.0.1 Rede Semântica do Museu da Emigração

Parser

```
%option noyywrap

%%
Emigrante                                   return EMIGRANTE;
Nome                                        return NOME;
Dados\ pessoais                             return DADOSPESSOAIS;
Dados\ de\ emigracao                        return DADOSDEEMIGRACAO;
Fez                                         return FEZ;
Participou                                  return PARTICIPOU;
Obra                                        return OBRA;
Evento                                      return EVENTO;
Detalhes                                    return DETALHES;
[a-zA-Z0-9]+[a-zA-Z0-9 /\.,]*               {yylval.s= strdup(yytext); return VAR;}
[{};:\-\>\<=]                               return yytext[0];
[\n\t ]
%%
```

Gramática

```
%{
#define _GNU_SOURCE
#include<stdio.h>
#include<string.h>
#include<fcntl.h>
int yylex();
void yyerror(char* c);
void writeHTML(char *s, char* h);
char * emigranteAtual;
char * htmlToWrite;
char * htmlToWriteDadosPessoais;
char * htmlToWriteDadosEmigracao;
char * htmlToWriteFez;
```

```
char * htmlToWriteParticipou;
char * relacoesEncontradasFez;
char * relacoesEncontradasParticipou;
%}

%union {char* s;}
%token EMIGRANTE NOME VAR DADOSPESSOAIS FEZ PARTICIPOU DADOSDEEMIGRACAO OBRA DETALHES EVENTO
%type <s> emigrante VAR elementos fez participou relacoesFez relacoesParticipou dadosEmigracao nodos grafo
            dadosPessoais obra detalhes evento
%%

grafo: nodos  {printf("digraph{\nrankdir=LR\n%s\n}",$1);}

nodos : EMIGRANTE '{' emigrante '}' nodos {
                                              asprintf(&$$,"%s%s", $3,$5);
                                          }
      | OBRA '{' obra '}' nodos           {asprintf(&$$,"%s%s",$3,$5);}
      | EVENTO '{' evento '}' nodos       {asprintf(&$$,"%s%s",$3,$5);}
      |                                   {$$="";}
;

emigrante: NOME ':' VAR ';' dadosPessoais dadosEmigracao fez participou {
         asprintf(&htmlToWrite,
         "<!DOCTYPE html>\n"
           "<html lang=\"pt\">\n"
           "\t<head>\n"
             "\t\t<meta charset=\"utf8\">\n"
             "\t\t<link rel=\"stylesheet\" href=\"strips.css\">\n"
             "\t\t<link rel=\"stylesheet\" href=\"//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/
             font-awesome.min.css\">\n"
             "\t\t<link rel=\"stylesheet\" href=\"title.css\">\n"
             "\t\t<script type=\"text/javascript\" src=\"http://code.jquery.com/jquery-1.7.1.min.js\">
             </script>\n"
           "\t</head>\n"
           "\t<body><section class=\"strips\">\n"
             "%s"
             "%s"
             "%s"
             "%s"
             "<i class=\"fa fa-close strip__close strip__close--show\" style=\"transition: all 0.6s
             cubic-bezier(0.23, 1, 0.32, 1) 1s;\">\n"
             "\t\tFechar\n"
             "</i>\n"
           "\t</section>\n"
           "\t<script type=\"text/javascript\" src=\"https://codepen.io/ettrics/pen/ZYqKGb.js\"></script>\n"
             "\t</body>\n"
           "</html>",htmlToWriteDadosPessoais, htmlToWriteDadosEmigracao, htmlToWriteFez,
           htmlToWriteParticipou);
           emigranteAtual=$3;

           asprintf(&$$,"\"%s\"[href=\"%s.html\" style=filled fillcolor=\"aliceblue\"]\n",emigranteAtual,
           emigranteAtual);

           char *pt;
```

```
            if(relacoesEncontradasFez != NULL){
               for(pt = strtok(relacoesEncontradasFez,";"); pt != NULL;  pt = strtok (NULL, ";")){
                 asprintf(&$$,"%s\"%s\"->\"%s\" [label = \"Fez\"]\n",$$,emigranteAtual,pt);
               }
               free(relacoesEncontradasFez);
               relacoesEncontradasFez = NULL;
            }
            if(relacoesEncontradasParticipou != NULL){
               for(pt = strtok(relacoesEncontradasParticipou,";"); pt != NULL;  pt = strtok (NULL, ";")){
                   asprintf(&$$,"%s\"%s\"->\"%s\" [label = \"Participou\"]\n",$$,emigranteAtual,pt);
                 }
               free(relacoesEncontradasParticipou);
               relacoesEncontradasParticipou = NULL;
            }
              writeHTML(emigranteAtual,htmlToWrite);

            }
;
dadosPessoais: DADOSPESSOAIS '-' '>' elementos '<' '-'{
              asprintf(&htmlToWriteDadosPessoais,
                "<article class=\"strips__strip\">\n"
                  "<div class=\"strip__content\">\n"
                    "<H1 class=\"strip__title\">Dados Pessoais</H1>\n"
                    "<div class=\"strip__inner-text\" style=\"transition: all 0.5s cubic-bezier(0.23, 1, 0.32,
                    1) 0.3s;\">\n"
                      "\t\t<ul style=\"list-style-type:circle\">\n"
                        "%s"
                      "</ul>\n"
                    "\t\t</div>\n"
                  "</div>\n"
                "</article>\n"
                ,$4);
            }
;

dadosEmigracao: DADOSDEEMIGRACAO '-' '>' elementos '<' '-'{
              asprintf(&htmlToWriteDadosEmigracao,
                "<article class=\"strips__strip\">\n"
                  "<div class=\"strip__content\">\n"
                    "<H1 class=\"strip__title\">Dados de Emigração</H1>\n"
                    "<div class=\"strip__inner-text\" style=\"transition: all 0.5s cubic-bezier(0.23, 1, 0.32,
                    1) 0.3s;\">\n"
                      "\t\t<ul style=\"list-style-type:circle\">\n"
                        "%s"
                      "</ul>\n"
                    "\t\t</div>\n"
                  "</div>\n"
                "</article>\n"
                ,$4);
            }

fez: FEZ '-' '>' relacoesFez '<' '-'{
              asprintf(&htmlToWriteFez,
```

```
                "<article class=\"strips__strip\">\n"
                  "<div class=\"strip__content\">\n"
                    "<H1 class=\"strip__title\">Obras</H1>\n"
                    "<div class=\"strip__inner-text\" style=\"transition: all 0.5s cubic-bezier(0.23, 1, 0.32,
                    1) 0.3s;\">\n"
                      "\t\t<ul style=\"list-style-type:circle\">\n"
                        "%s"
                      "</ul>\n"
                    "\t\t</div>\n"
                  "</div>\n"
                "</article>\n"
                ,$4);
            }
;

participou: PARTICIPOU '-' '>' relacoesParticipou '<' '-' {
              asprintf(&htmlToWriteParticipou,
                "<article class=\"strips__strip\">\n"
                  "<div class=\"strip__content\">\n"
                    "<H1 class=\"strip__title\">Eventos</H1>\n"
                    "<div class=\"strip__inner-text\" style=\"transition: all 0.5s cubic-bezier(0.23, 1, 0.32,
                    1) 0.3s;\">\n"
                      "\t\t<ul style=\"list-style-type:circle\">\n"
                        "%s"
                      "</ul>\n"
                    "\t\t</div>\n"
                  "</div>\n"
                "</article>\n",$4);
            }
;



relacoesFez: VAR ';' relacoesFez {
                          asprintf(&$$,"<a href=\"%s.html\"><li>%s</li></a>\n%s",$1,$1,$3);
                          if(relacoesEncontradasFez != NULL){
                            asprintf(&relacoesEncontradasFez,"%s;%s",$1,relacoesEncontradasFez);
                          }
                          else{
                            asprintf(&relacoesEncontradasFez ,"%s;",$1);
                          }
                        }
      |                 {$$="";}
;

relacoesParticipou: VAR ';' relacoesParticipou{
                          asprintf(&$$,"<a href=\"%s.html\"><li>%s</li></a>\n%s",$1,$1,$3);
                          if(relacoesEncontradasParticipou != NULL){
                            asprintf(&relacoesEncontradasParticipou,"%s;%s",$1,relacoesEncontradasParticipou);
                          }
                          else{
                            asprintf(&relacoesEncontradasParticipou,"%s;",$1);
                          }
                        }
```

```
      |              {$$="";}
;

obra: NOME ':' VAR ';' detalhes {
                           asprintf(&$$,"\"%s\"[href=\"%s.html\" style=filled fillcolor=\"lawngreen\"]\n",
                           $3,$3);
                           asprintf(&$5,
                           "<!DOCTYPE html>\n"
                             "<html lang=\"pt\">\n"
                               "\t<head>\n"
                                 "\t\t<meta charset=\"utf8\">\n"
                                 "\t\t<link rel=\"stylesheet\" href=\"title.css\">\n"
                                 "%s",$5);
                           writeHTML($3,$5);
                         }

evento: NOME ':' VAR ';' detalhes {
                           asprintf(&$$,"\"%s\"[href=\"%s.html\" style=filled fillcolor=\"gold\"]\n",$3,$3);
                           asprintf(&$5,
                           "<!DOCTYPE html>\n"
                             "<html lang=\"pt\">\n"
                               "\t<head>\n"
                                 "\t\t<meta charset=\"utf8\">\n"
                                 "\t\t<link rel=\"stylesheet\" href=\"title.css\">\n"
                                 "%s",$5);
                           writeHTML($3,$5);
                         }

detalhes: DETALHES '-' '>' elementos '<' '-' {$$=$4;}

elementos: VAR '=' VAR ';' elementos {
                                        if(!strcmp($1,"Esposa/Marido ") || !strcmp($1,"Esposa/Marido"))
                                          asprintf(&$$,"<li>%s: <a href=\"%s.html\">%s</a></li>\n%s",$1,
                                          $3,$3,$5);
                                        else
                                          asprintf(&$$,"<li>%s: %s</li>\n%s",$1,$3,$5);
                                      }
          |                          {$$="";}
;


%%
#include "lex.yy.c"
char * emigranteAtual;
char * htmlToWrite = NULL;
char * htmlToWriteDadosPessoais = NULL;
char * htmlToWriteDadosEmigracao = NULL;
char * htmlToWriteFez = NULL;
char * htmlToWriteParticipou = NULL;
char * relacoesEncontradasFez = NULL;
char * relacoesEncontradasParticipou = NULL;
int main(){
  yyparse();
```

```
  if(htmlToWrite != NULL)
    free(htmlToWrite);
  return 0;
}
void yyerror(char* c){
  fprintf(stderr,"%s, %s, %d \n",c,yytext,yylineno);
}
void writeHTML(char* nomeFicheiro, char* htmlString){
  char * fileName;
  fileName= malloc(sizeof(char)*(strlen(nomeFicheiro)+10));
  sprintf(fileName,"%s.html",nomeFicheiro);
  FILE *fd=fopen(fileName,"w");

  if(fd) {
    fprintf(fd,"<h1 class = \"page-title\">%s</h1>\n",nomeFicheiro);
    fprintf(fd,"%s",htmlString);
    fclose(fd);
  }
  else printf("Erro ao escrever ficheiro %s\n",fileName);
  free(fileName);
}
```

Makefile

```
all: htmlAndDot result.dot
        dot -Tsvg result.dot -o grafo.svg

compile: cFLEX cyacc lex.yy.c
        gcc -o tp3 y.tab.c

cFLEX: tp3.lex
        FLEX tp3.lex

cyacc: tp3.y
        yacc tp3.y

htmlAndDot: compile dados.mdp
        ./tp3 < dados.mdp > result.dot
```

Bibliografia

[1] codepen.io. https://codepen.io/hbuchel/pen/qOxGzW.css.

[2] codepen.io. https://codepen.io/ettrics/full/ZYqKGb/.