Processamento de Linguagens (3º ano de MIEI)

# Trabalho Prático 2

Relatório de Desenvolvimento


Catarina Machado (a81047)

Gonçalo Faria (a86264)

João Vilaça (a82339)


28 de Abril de 2019

**Resumo**

O segundo trabalho prático no âmbito da unidade curricular Processamento de Linguagens consistiu na elaboração de um processador de cartas setecentistas da Etiópia usando AWK.

Deste modo, foi analisado um ficheiro csv que contém informação diversa sobre uma coleção de cartas trocadas nos anos de mil seiscentos aquando da viagem dos navegadores portugueses à Etiópia e foram desenvolvidos filtros AWK que permitiram responder a diferentes questões.

Os filtros produzidos tinham os seguintes objetivos: contar o número de cartas por local relacionando-as com o ano de escrita; criar um index HTML com todos os anos, em que cada ano deve ligar a outra página HTML onde conste, para cada carta desse ano, o título da carta e o seu resumo; mostrar a lista das cartas—cada uma identificada pelo número, devidamente associada (em pares num-nome) aos Apelidos das pessoas envolvidas no assunto relatado e desenhar um grafo (em DOT) que relacione cada autor com o destinatário.

De modo a facilitar a consulta dos resultados produzidos, foi construída ainda uma página HTML onde se pode selecionar a questão que se pretende ver o resultado.

No presente relatório explicamos a forma como desenvolvemos este projeto e as decisões tomadas ao longo do mesmo.

# Conteúdo

# Lista de Figuras

# Capítulo 1

# Introdução

*Área: Processamento de Linguagens*

## 1.1 Enquadramento e Contexto

Existe uma coleção de cartas trocadas nos anos de mil seiscentos, aquando da viagem dos navegadores portugueses à Etiópia. Nessa coleção, cada carta possui as seguintes informações: número, data, local, título, apelidos das pessoas envolvidas no assunto e o resumo da carta.

Assim, o projeto consiste em extrair de um ficheiro *input* a informação diversa sobre a coleção de cartas e tratar convenientemente esses dados, apresentando-os em páginas HTML de forma a que possam ser facilmente consultados.

Para isso, recorremos a uma linguagem e processador de padrões, o AWK, que permite programar filtros e relatórios sobre informação tabular de forma muito acessível.

O AWK é orientado à linha e a estrutura dos filtros é **pattern {action}**, onde o elemento *pattern* é a especificação de um teste que é avaliado em cada linha e quando é verdadeiro a *action* é executada. Caso nenhum padrão for especificado todas as linhas são processadas.

Existem algumas variáveis internas que facilitam o processamento do ficheiro *input* como por exemplo o **FS** *(field separator)*, que indica como é que as linhas devem ser separadas, o **NR** *(number of records)*, que representa o número de linhas processadas até ao momento e o **NF** *(number of fields)*, que indica o número de campos da linha corrente.

Para além disso, permite a utilização de operadores relacionais, padrões com expressões regulares, padrões com expressões relacionais e padrões compostos. Permite também a aplicação de padrões **BEGIN** e **END**, instruções de controlo (como o *if*, o *for* e o *print*), variáveis, arrays e arrays multi-dimensionais.

O AWK tem ainda várias funções, como por exemplo a função split, gsub, sub e gensub, mas também permite que criemos as nossas.

Deste modo, no presente relatório é explicada a forma como cumprimos os requisitos utilizando filtros AWK e como geramos as páginas HTML com os resultados obtidos.

## 1.2 Problema

Recorrendo à utilização de filtros AWK, o problema passa por construir um ou mais programas que processem uma coleção de cartas setecentistas da Etiópia de modo a:

- Contar o número de cartas por local (considerando o local NIL quando se desconhece) relacionando-as com o ano de escrita. Para tal, decidimos que deviam ser construídos três diferentes filtros, um que apenas conta o número de cartas por local, outro que conta o número de cartas por ano e um terceiro que conta o número de cartas por local e ano de escrita;
- Criar um index HTML com todos os anos, em que cada ano deve ligar a outra página HTML onde conste, para cada carta desse ano, o título da carta e o seu resumo;
- Mostrar a lista das cartas— cada uma identificada pelo número, devidamente associada (em pares num-nome) aos apelidos das pessoas envolvidas no assunto relatado;
- Desenhar um grafo (em DOT) que relacione cada autor (identificado pelo seu nome) com o destinatário (também identificado pelo nome);
- Para cada uma das questões anteriormente mencionadas deve ser construída uma página HTML com os resultados obtidos;
- Construir uma página HTML geral que dê acesso a todas as perguntas e respostas desenvolvidas.

O objetivo do projeto é então dar resposta a todas estas questões, conseguindo assim obter várias informações sobre as cartas setecentistas da Etiópia da forma mais clara, percetível e apelativa possível.

Consequentemente, o objetivo deste projeto é também aumentar a experiência de uso do ambiente Linux e de algumas ferramentas de apoio à programação, aumentar a capacidade de escrever *Expressões Regulares (ER)* para descrição de *padrões de frases*, desenvolver, a partir de ERs, sistemática e automaticamente *Processadores de Linguagens Regulares*, que filtrem ou transformem textos e utilizar o sistema de produção para *filtragem de texto* GAWK.

## 1.3 Objetivo

O objetivo do presente relatório é explicar como tiramos o máximo partido do Sistema de Produção GAWK e como foram desenvolvidos todos os filtros que visam dar resposta às questões propostas. Para além disso, o objetivo é também explicar a forma como decidimos apresentar as informações obtidas, isto é, como geramos as páginas HTML com os resultados.

## 1.4 Estrutura do Relatório

O presente relatório está dividido em 5 diferentes capítulos.

No capítulo 1, **Introdução**, é feito um enquadramento e contextualização do trabalho prático e, em seguida, é feita uma descrição do problema a desenvolver, assim como os objetivos do projeto e do relatório.

No capítulo 2, **Análise e Especificação**, é feita uma descrição informal do problema e uma especificação dos requisitos necessários para uma correta resolução do problema, ou seja, é desvendada a abordagem que consideramos ser a mais correta para uma eficiente resolução e explicação do trabalho prático.

Em seguida, no capítulo 3, **Concepção e desenho da Resolução** é feita uma descrição detalhada de todo o desenvolvimento do projeto até se obter a solução final, onde são também apresentados excertos dos filtros AWK criados.

Posteriormente, no capítulo 4, **Testes**, são apresentados alguns *prints* de todas as páginas HTML geradas.

Por fim, no capítulo 5, **Conclusão**, termina-se o relatório com uma síntese e análise do que trabalho realizado.

# Capítulo 2

# Análise e Especificação

## 2.1 Descrição informal do problema

Dado um arquivo csv com informação sobre cartas setecentistas da Etiópia, é necessário extrair informação relativa à contagem de cartas por local e ano, obter a lista dos anos onde foram enviadas cartas e para cada um deles as cartas desse ano, com o respetivo título e resumo, uma lista das cartas identificadas pelo seu número e apelido das pessoas envolvidas e desenhar um grafo que relacione o autor com o destinatário de cada carta.

## 2.2 Especificação do Requisitos

De forma a responder devidamente ao problema proposto decidimos dividir o nosso projeto em diferentes etapas, mais especificamente em diferentes filtros, onde cada um deles tem como objetivo responder a cada um dos requisitos do trabalho prático.

### 2.2.1 Contagem do número de cartas por local e/ou ano

São necessários três filtros que tenham como objetivo contar o número total de cartas por local (considerando NIL quando se desconhece o local), número total de cartas por ano e por fim o número de cartas por local e ano.

### 2.2.2 Listagem dos anos com as cartas enviadas em cada um deles

É necessário um filtro que construa um index HTML com todos os anos em que foram enviadas cartas, contendo para cada um dos anos uma página HTML com o título e o resumo das cartas.

### 2.2.3 Listagem das cartas, identificadas em pares num-nome

Para este requisito, precisamos de um filtro que faça uma lista de todas as cartas, cada uma identificada pelo número e associada (em pares Num-Nome) aos apelidos das pessoas envolvidas no assunto relatado.

### 2.2.4 Grafo que relaciona os autores com os destinatários

Um filtro que construa um grafo que relacione cada autor (identificado pelo seu nome) com o destinatário (também identificado pelo nome).

### 2.2.5 Página principal do projeto

Por fim, decidimos também que seria bastante útil elaborar um filtro que construísse uma página HTML que permita visualizar ordenadamente os resultados de cada um dos requisitos anteriormente mencionados. Para além disso, nesta página deverá também ser apresentado o número de cartas total no ficheiro *input*.

# Capítulo 3

# Concepção/desenho da Resolução

## 3.1 Estruturas de Dados

Para o desenvolvimento do presente trabalho prático não necessitamos de recorrer a nenhuma biblioteca externa, nem a nenhuma estrutura de dados elaborada. Os dados que estavam a ser processados iam sendo armazenados temporariamente em arrays e/ou arrays multi-dimensionais.

## 3.2 Implementação

Tal como já mencionamos, decidimos dividir o nosso projeto em vários programas AWK, cada um com o objetivo de responder a um dos requisitos propostos, que passaremos a enumerar e explicar em seguida.

Para além disso, tal como já mencionado, todos esses filtros imprimem o seu resultado para páginas HTML de modo a que o resultado final obtido seja mais apelativo e mais fácil de ser consultado.

### 3.2.1 Contagem do número de cartas por local e/ou ano

Tal como já mencionado, no caso deste requisito decidimos fazer três diferentes interpretações do problema. Para tal, construímos três filtros, onde cada um visa dar resposta a cada um desses problemas.

#### 1 - Contagem do número de cartas por local

Para a concretização deste filtro, no padrão **BEGIN** decidimos especificar a utilização do carácter **;** como o **FS** *(field separator)*. Para além disso, ainda na *action* do padrão **BEGIN** imprimimos para a página HTML output as tags iniciais utilizadas na construção de páginas HTML e o título pretendido para a mesma.

Em seguida, através da utilização de padrões com expressões regulares, com expressões relacionais e de padrões compostos construímos os **pattern {action}** que se podem ver em seguida.

```
$3 !~ /[a-zA-Z]+/ && $3 ~ /[ ]+/        { contaLocais["NIL"]++ }

$3 ~ /[a-zA-Z]+/                        { split($3, local, "[ ]{2,10}|]");
                                          sub(/ *$/, "", local[2]);
                                          contaLocais[local[2]]++
                                        }
```

Desta forma, caso o local, que corresponde ao campo **$3**, não contenha letras e contenha mais do que um espaço significa que se trata de uma carta sem local definido, ou seja, o local será NIL. A necessidade da utilização desse padrão composto deve-se ao facto de todos locais das cartas conterem vários espaços desnecessários para além das letras, o que faz com que seja necessário garantir simultaneamente estas duas condições. Nesta caso, é então incrementada 1 unidade no array com índice igual a NIL.

No caso do campo correspondente à data contiver letras, significa que a carta tem um local definido e a estratégia adotada foi a de utilizar o local como índice do array. Desta forma, fica assegurado que quando aparecerem locais repetidos será incrementada a mesma posição no array. Antes de armazenarmos esses valores, tendo em consideração que os dados contêm espaços a mais, foi necessário recorrer à função *split* para retirar os espaços desnecessários no início do local e um eventual carácter que aparece num dos locais (carácter [) e à função *sub*, que se encarrega de remover os espaços indesejados no final do local.

Por fim, no padrão **END** imprimimos os índices do array (que correspondem aos nomes dos locais) e o número de ocorrências do mesmo (valor do array no respetivo índice) para a página HTML output, tendo em consideração que o número de cartas contabilizadas pode ser plural ou singular, utilizamos a instrução de controlo *if* para garantirmos um output mais aprimorado. Para além disso, imprimimos também as tags de término usuais de uma página HTML.

```
END             { for(l in contaLocais){
                      number = contaLocais[l];
                      if(number == "1")
                          {print "<li><b>"l" -> </b> "number" carta</li><br>"
                                      > "out/html/contaLocais.html"}
                      else
                          {print "<li><b>"l" -> </b> "number" cartas</li><br>"
                                      > "out/html/contaLocais.html"}
                                        };
                    print "</body></html>" > "out/html/contaLocais.html"
                  }
```

### 2 - Contagem do número de cartas por ano

No caso da contagem do número de cartas por ano o processo foi bastante semelhante. O **FS** *(field separator)* foi também o carácter **;** e, antes do processamento da primeira linha, imprimimos também para a página HTML output as tags HTML necessárias.

O código AWK utilizado para o processamento do arquivo foi o seguinte.

```
$2 ~ /[1-9]+/                          { split($2, data, "[ ]{2,10}|[.]");
                                         contaAnos[data[2]]++
                                       }
```

Assim, para obtermos o resultado pretendido bastou filtrar as linhas que contivessem números no segundo campo (que corresponde ao campo da data) e, para esses casos, começamos por retirar os espaços a mais no início da data e o carácter **.**, que é utilizado no arquivo csv para separar o ano, o mês e o dia. Desta forma, utilizando os *field separators* mencionados na função split conseguimos obter somente o ano da data em questão. Em seguida, utilizamos a mesma estratégia mencionada na contagem por local, ou seja, utilizamos o ano da data como índice do array e fomos incrementando à medida que apareciam novas datas.

Assim, após todas as linhas terem sido executadas, com a ajuda do padrão **END** imprimimos o ano e o respetivo número de ocorrências para a página HTML output e as tags HTML *</body></html>*.

### 3 - Contagem do número de cartas por local e ano

Para a contagem das cartas por local e ano o raciocínio utilizado foi semelhante mas com algumas nuances.

No padrão **BEGIN** foi alterado o **FS** *(field separator)* para o carácter **;** e foram impressas para a página HTML output algumas tags HTML e foi também utilizado CSS para que o resultado produzido através do filtro fosse apresentado em 2 colunas.

Para o desenvolvimento deste filtro sentimos a necessidade de recorrer a um array multi-dimensional, que tinha como objetivo guardar simultaneamente o local e o ano da carta. Para tal, os pares **pattern {action}** utilizados foram os seguintes:

```
$3 !~ /[a-zA-Z]+/ && $3 ~ /[ ]+/        { split($2, data, "[ ]{2,10}|[.]");
                                          contaLocaisData["NIL:"data[2]]++
                                        }

$3 ~ /[a-zA-Z]+/                        { split($3, local, "[ ]{2,10}|]");
                                          sub(/ *$/, "", local[2]);
                                          split($2, data, "[ ]{2,10}|[.]");
                                          contaLocaisData[local[2]":"data[2]]++
                                         }
```

A forma de retirar informação relativa ao local e ao ano foi igual à já explicada anteriormente nesses dois filtros individuais.

Desta forma, tivemos somente que definir o índice do array como a concatenação dos parâmetros local e data, utilizando o carácter **:** como operador de concatenação. Desta forma, sempre que aparecem cartas com este índice é incrementada a posição do array desejada.

Depois de todas as linhas processadas e de forma a produzir um output percetível e com mais detalhe, decidimos ordenar o array por ordem alfabética do nome do local, utilizando a estratégia que se pode ver em seguida.

```
END                                     { j = 1;
                                          for (i in contaLocaisData){
                                            ind[j] = i;
                                            j++
                                          };

                                          n = asort(ind);
                                          ...
```

Desta forma, temos em **ind** o array ordenado.

Para imprimirmos o output de forma apelativa, manipulamos o índice do array através da função gensub, colocando o local e a data em **a** e **b**, respetivamente.

Assim, para imprimirmos o output de forma clara apenas tivemos que fazer print dessas variáveis da forma que achamos mais conveniente, separando também o plural e o singular do número das cartas.

```
...
for (i = 1; i <= n; i++){
    a = gensub(/(.+):(.+)/, "\\1", "g", ind[i]);
    b = gensub(/(.+):(.+)/, "\\2", "g", ind[i]);
    number = contaLocaisData[ind[i]];

    if(number == "1")
        {print "<li><b>"a": Ano "b" -> </b>"number" carta</li></h4>"
                        > "out/html/contaLocaisAno.html"}
    else
        {print "<li><b>"a": Ano "b" -> </b>"number" cartas</li></h4>"
                        > "out/html/contaLocaisAno.html"}
 };

 print "</div></body></html>" > "out/html/contaLocaisAno.html"
}
```

### 3.2.2 Listagem dos anos com as cartas enviadas em cada um deles

Para a construção de um filtro que permite construir uma página principal (*anos.html*) com todos os anos em que foram enviadas cartas e, para cada um dos anos, uma nova página com o título e o resumo das cartas enviadas nesse ano utilizamos também o carácter **;** como **FS**.

Consequentemente, criamos um *pattern* para que tivessemos a garantia que só iríamos processar linhas com informações. Após uma análise detalhada ao ficheiro *input* sabemos que todas as cartas têm ano definido, logo, não precisamos de nos prevenir em relação a isso.

Tal como se pode ver em seguida, através da função split obtemos o número, o ano e o título da carta. O array *contaDatas*, que conta o número de cartas por ano, serve para verificar se é a primeira vez que cada ano aparece. Em caso afirmativo, é impressa para a página *anos.html* o ano e a hiperligação para a página do mesmo.

Em seguida, tendo em consideração que temos as informações do número, ano e título da carta nas variáveis *numero*, *data* e *titulo*, respetivamente, resta-nos somente imprimi-las para a página do ano.

Por fim, fica apenas a falar imprimir o resumo da carta. Para este caso, sentimos a necessidade de o fazer através de um ciclo *for*, a partir da coluna número 6 até à coluna **NF** *(number of fields)*, uma vez que existe um número indeterminado de colunas com informação sobre o resumo da carta.

```
$0 ~ /[a-zA-Z]+/              { split($1, numero, "[ ]{2,10}");
                                split($2, data, "[ ]{2,10}|[.]");
                                split($4, titulo, "[ ]{2,10}");
                                contaDatas[data[2]]++;

                                if (contaDatas[data[2]] == 1) {
                                    printf("<li><a href='%s'>Cartas de %s</a></li>",
                                       "anos/" data[2]".html", data[2]) > "out/html/anos.html";
                                    printf("<h1><p align='center'><font color='#2874A6'>
                                       Cartas de %s</font></p></h1><br><br>",data[2])
                                       > "out/html/anos/" data[2] ".html";
                                };
```

```
printf("<h3><li><font color='#85C1E9'>Carta número %s
   </font></li></h3>",numero[2]) >
   "out/html/anos/" data[2] ".html";

print "<br><b>Mês:</b> "data[3]
   > "out/html/anos/" data[2] ".html";
print "<br><b>Dia:</b> "data[4]
   > "out/html/anos/" data[2] ".html";
print "<br><b>Título:</b> "titulo[2]
   > "out/html/anos/" data[2] ".html";
print "<br><b>Resumo:</b> "
   > "out/html/anos/" data[2] ".html";
for(i=6;i<=NF;i++) print $i > "out/html/anos/" data[2] ".html";
print "<br><br><br><br><br>"
> "out/html/anos/" data[2] ".html"
}
```

### 3.2.3 Listagem das cartas, identificadas em pares num-nome

Outro dos requisitos necessários é a elaboração da lista de todas as cartas, cada uma identificada pelo número e associada aos apelidos das pessoas envolvidas no assunto relatado (em pares Num-Nome). Para tal, utilizamos também o carácter **;** como **FS**.

O filtro construído considera todas as linhas que tenham algum carácter alfabético. Assim, para essas linhas, o que o filtro faz é armazenar em **número** o número da carta, com a ajuda da função split. Os apelidos são guardados em **apelidos**, também com a ajuda da função split, utilizando como *field separator*, para além dos espaços em branco indesejados ([ ]2,10), o carácter **:**.

Posto isto, basta somente imprimir os pares Num-Nome pretendidos, através da função print, de um ciclo for e da instrução de controlo if, que faz com que sejam somente impressos os valores do *array* **apelidos** com caracteres alfabéticos (ignorando os índices do *array* que só possuem espaços em branco).

```
$0 ~ /[a-zA-Z]+/              { split($1, numero, "[ ]{2,10}");
                                printf("<ul><h3><li><font   color='#85C1E9'>
                                  Carta número %s</font></li></h3><ul>",numero[2]) >
                                  "out/html/cartas.html";

                                split($5, apelidos, "[ ]{2,10}|:");
                                for(a in apelidos){
                                    if(apelidos[a] ~ /[a-zA-Z]+/){
                                       print "<li>"numero[2]"-"apelidos[a]"</li><br>" >
                                           "out/html/cartas.html"
                                    }
                                };

                                print "</ul></ul>" > "out/html/cartas.html"
                               }
```

### 3.2.4 Grafo que relaciona os autores com os destinatários

Para o caso do desenho de um grafo que relaciona cada autor (identificado pelo seu nome) com o destinatário (também identificado pelo nome) tivemos que recorrer a DOT.

Numa primeira fase, investigamos como deveria ser escrito o programa em DOT para que posteriormente o pudéssemos converter para png. Assim, percebemos que o layout do ficheiro DOT deveria ser:

```
digraph{
    rankdir=LR;
    "Remetente 1" -> "Destinatário 1";
    "Remetente 2" -> "Destinatário 2";
    ...
}
```

Em seguida, começamos então a elaborar o filtro pretendido e, mais uma vez, utilizamos o carácter **;** como **FS**. Antes do processamento da primeira linha, imprimimos para o ficheiro *output* a indicação de que se trata de um grafo em DOT.

```
BEGIN                       { FS=";";
                              print "digraph{" > "out/grafo/grafo.dot";
                              print "\trankdir=LR;" > "out/grafo/grafo.dot";
                            }
```

Posteriormente, processamos todas as linhas que contêm caracteres alfabéticos. Para linhas, inicialmente, através da função split e com os *fields separators* destinados para o efeito, armazenamos em **autores** o remetente da carta, que inclui neste momento também inclui o resto da frase. Em seguida, com a ajuda da função gsub, em **autores[2]** (que é onde se encontra exatamente o remetente da carta), substituímos a conjunção que precede o nome do destinatário da carta pelo carácter "->". Desta forma, temos então: Nome do remetente" ->"Nome do destinatário.

```
$0 ~ /[a-zA-Z]+/            { split($4, autores, "Carta de |
                                                  que o |
                                                  [cC]arta do |Requerimento do |
                                                  dad[ao] pelo ");
                              gsub(" a um | a | dá ao | ao | aos ", "\" -> \"", autores[2]);

                              if(autores[2] !~ "->"){
                                  unknown(autores);
                              } else {
                                  print "\t\""autores[2]"\";" > "out/grafo/grafo.dot";
                              }
                            }
```

Nesse cenário de sucesso, tal como se pode ver no *else*, para que o resultado da linha fique o pretendido basta somente inserir aspas no início e no final da frase e, em seguida, imprimir para o ficheiro *output*. No entanto, existe a possibilidade de, ou o remetente, ou o destinatário serem desconhecidos e, nesse caso, a linha ainda não se encontra como pretendemos. Decidimos intitular o remetente/destinatário como "Desconhecido" caso estes nomes não se encontrem no ficheiro de *input*. Para tal, recorremos à função **unknown**.

```
function unknown(autores)
{
    #unknow destinatário
    if(autores[2] ~ /[a-zA-Z]+/){
        print "\t\""autores[2]"\" -> \"Desconhecido\";" > "out/grafo/grafo.dot";
    }

    #unknow remetente
    if(autores[1] ~ " ao "){
        split(autores[1], autor, "Carta ao |Carta enviada ao | que ficaram ");
        print "\t\"Desconhecido\" -> \""autor[2]"\";" > "out/grafo/grafo.dot";
    }
}
```

Esta função, tal como se pode ver em cima, divide-se em dois diferentes cenários: no caso do destinatário ser desconhecido e no caso de ser o remetente desconhecido. Em ambos os casos faz-se as averiguações necessárias para se poder tomar a decisão de qual dos casos a linha se insere e imprime para o ficheiro *output* o resultado desejado.

Posto isto, temos então um ficheiro **.dot** com o layout que permite a construção de um grafo. Para construirmos visualmente o grafo basta correrm o comando dot com as flags necessárias, que estará inserido na Makefile do projeto.

### 3.2.5 Página principal do projeto

Por fim, tal como já mencionado, construímos uma página HTML que permite visualizar todas as questões elaboradas com as respetivas hiperligações para os resultados obtidos. Esta página foi também construída através de um filtro AWK que somente imprime HTML e também um pouco de CSS. Este filtro permite ainda consultar o número total de cartas presentes no ficheiro *input* através da declaração de uma variável **total** e posteriormente incremento, sempre que aparece uma linha com caracteres alfabéticos.

```
$0 ~ /[a-zA-Z]+/            { total++ }
```

Este filtro é também responsável por imprimir a imagem resultante de correr o comando .dot para a página do grafo:

```
...
print "<h1><p align='center'><font color='#2874A6'>
   Grafo</font></p></h1>" > "out/html/grafo.html";
print "<center><img src='../grafo/grafo.png'
   alt='Grafo' width='1300' height='1100'></center>"
   > "out/html/grafo.html"
...
```

### 3.2.6 Makefile

Para uma compilação fácil e simples do programa criamos uma Makefile.

Esta Makefile é responsável por correr cada um dos sete filtros construídos usando para isso o comando GAWK e a flag -f. A Makefile corre ainda o comando dot, construindo o ficheiro *png* com o grafo. Por fim, abre a página principal do trabalho prático, que permite que sejam consultadas todas as perguntas e todas as respostas obtidas.

Assim, para correr o nosso programa basta escrever **make** no terminal, na diretoria principal do trabalho.

```
> make
```

Por definição, os ficheiros html resultantes da execução do programa vão para uma nova pasta chamada **out**.

A Makefile também possui a opção de *clean*, que limpa na pasta **out** todos os html gerados, o ficheiro *.dot* e o *png* do grafo construído.

## 3.3 Extras

Nesta trabalho prático decidimos embelezar o output obtido em cada um dos problemas, isto é, construímos páginas HTML para cada um dos requisitos. Para além disso, construímos uma página HTML principal que permite consultar todas as perguntas e todas as respostas recolhidas. Criamos também uma makefile que nos permite compiliar todos os filtros AWK criados e, por fim, abrir no browser a página principal elaborada.

Na primeira questão do presente projeto decidimos também fazer um filtro AWK para cada uma das diferentes interpretações da pergunta: contar o número de cartas apenas por local, apenas por ano e, por fim, simultaneamente por ano e por local.

# Capítulo 4

# Testes

## 4.1 Resultados

### 4.1.1 Impressão da Página Inicial

Na Figura 4.1 encontra-se a página inicial do nosso projeto, a página que é aberta após se correr o comando make.

### 4.1.2 Impressão da Cartas por Local e/ou Ano

Tal como se pode ver na Figura 4.1 e explicado nas secções anteriores, no caso da impressão das cartas por local e ano decidimos elaborar três diferentes filtros e, em consequência, existem também três hiperligações para os resultados obtidos.

Desta forma, na Figura 4.2 encontram-se o número de cartas por local, na Figura 4.3 encontram-se o número de cartas por ano e, por fim, na Figura 4.4 encontram-se o número de cartas por local e ano.

### 4.1.3 Impressão dos Anos das cartas e respetivas cartas

Na Figura 4.5 encontra-se a página com os anos das cartas, cada um com uma hiperligação para as cartas desse ano. Por exemplo para o ano 1622, carregando nesse *link*, somos reencaminhados para a página das cartas de 1622, tal como se pode ver na Figura 4.6.

### 4.1.4 Impressão das Cartas identificadas em pares num-nome

No caso da listagem das cartas identificadas em pares num-nome, a página com os resultados obtidos encontra-se na Figura 4.7.

### 4.1.5 Impressão do Grafo

No caso do grafo, a página com a imagem do grafo obtido está na Figura 4.8. A imagem obtida é a que se encontra na Figura 4.9.

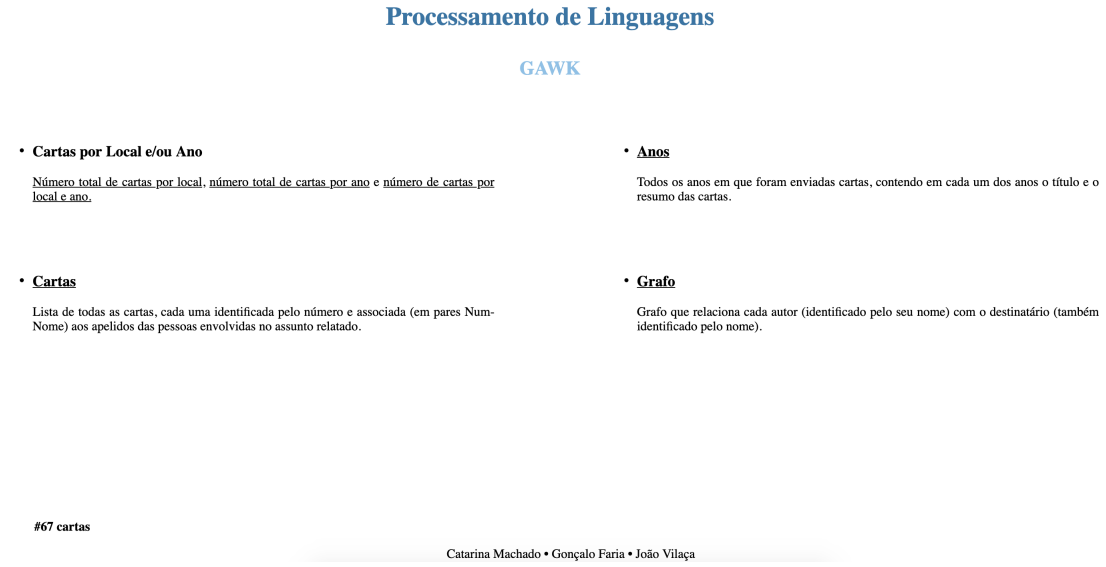


Figura 4.1: Página inicial

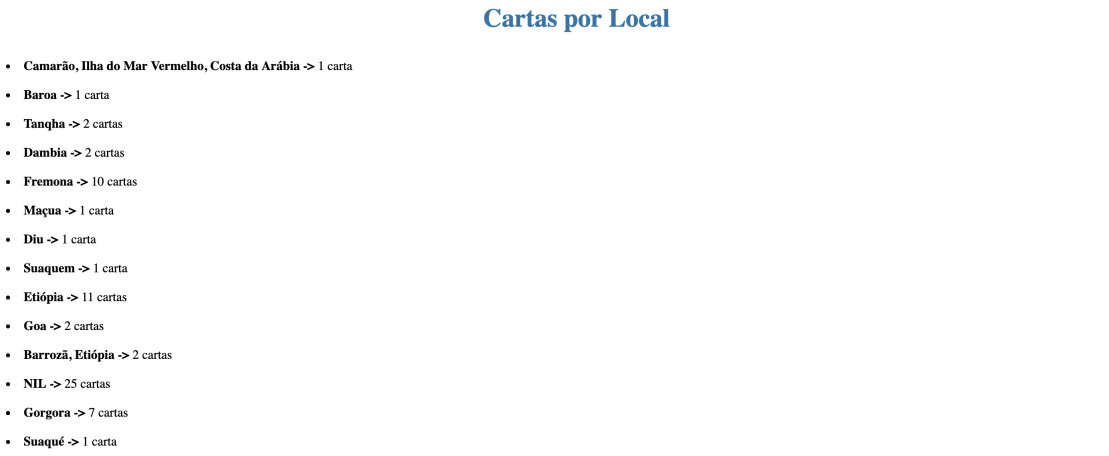


Figura 4.2: Cartas por Local

## Cartas por Ano

- **1542 ->** 1 carta
- **1556 ->** 1 carta
- **1603 ->** 1 carta
- **1604 ->** 4 cartas
- **1605 ->** 1 carta
- **1607 ->** 1 carta
- **1608 ->** 2 cartas
- **1609 ->** 2 cartas
- **1610 ->** 1 carta
- **1611 ->** 1 carta
- **1613 ->** 1 carta
- **1614 ->** 1 carta
- **1615 ->** 5 cartas
- **1619 ->** 2 cartas
- **1620 ->** 4 cartas
- **1621 ->** 1 carta
- **1622 ->** 2 cartas
- **1623 ->** 1 carta

Figura 4.3: Cartas por Ano

## Cartas por Local e Ano

- **Baroa: Ano 1626 ->** 1 carta
- **Barrozã, Etiópia: Ano 1634 ->** 2 cartas
- **Camarão, Ilha do Mar Vermelho, Costa da Arábia: Ano 1630 ->** 1 carta
- **Dambia: Ano 1620 ->** 2 cartas
- **Diu: Ano 1605 ->** 1 carta
- **Etiópia: Ano 1604 ->** 1 carta
- **Etiópia: Ano 1615 ->** 1 carta
- **Etiópia: Ano 1622 ->** 1 carta
- **Etiópia: Ano 1623 ->** 1 carta
- **Etiópia: Ano 1626 ->** 2 cartas
- **Etiópia: Ano 1631 ->** 1 carta
- **Etiópia: Ano 1635 ->** 1 carta
- **Etiópia: Ano 1636 ->** 2 cartas
- **Etiópia: Ano 1698 ->** 1 carta
- **Fremona: Ano 1607 ->** 1 carta
- **Fremona: Ano 1609 ->** 1 carta
- **Fremona: Ano 1610 ->** 1 carta
- **Gorgora: Ano 1615 ->** 2 cartas
- **Gorgora: Ano 1624 ->** 1 carta
- **Gorgora: Ano 1627 ->** 1 carta
- **Gorgora: Ano 1628 ->** 1 carta
- **Gorgora: Ano 1931 ->** 1 carta
- **Maçua: Ano 1634 ->** 1 carta
- **NIL: Ano 1542 ->** 1 carta
- **NIL: Ano 1556 ->** 1 carta
- **NIL: Ano 1604 ->** 3 cartas
- **NIL: Ano 1608 ->** 2 cartas
- **NIL: Ano 1609 ->** 1 carta
- **NIL: Ano 1615 ->** 2 cartas
- **NIL: Ano 1620 ->** 2 cartas
- **NIL: Ano 1622 ->** 1 carta
- **NIL: Ano 1625 ->** 2 cartas
- **NIL: Ano 1632 ->** 1 carta
- **NIL: Ano 1633 ->** 1 carta

Figura 4.4: Cartas por Local e Ano

## Anos das Cartas

- Cartas de 1542
- Cartas de 1556
- Cartas de 1603
- Cartas de 1604
- Cartas de 1605
- Cartas de 1607
- Cartas de 1608
- Cartas de 1609
- Cartas de 1610
- Cartas de 1611
- Cartas de 1613
- Cartas de 1614
- Cartas de 1615
- Cartas de 1619
- Cartas de 1620
- Cartas de 1621
- Cartas de 1622
- Cartas de 1623
- Cartas de 1624
- Cartas de 1625
- Cartas de 1626
- Cartas de 1627
- Cartas de 1628
- Cartas de 1630
- Cartas de 1631
- Cartas de 1931
- Cartas de 1632
- Cartas de 1633
- Cartas de 1634
- Cartas de 1635
- Cartas de 1636
- Cartas de 1640
- Cartas de 1641
- Cartas de 1649
- Cartas de 1652
- Cartas de 1657
- Cartas de 1698

Figura 4.5: Anos das Cartas

## Cartas de 1622

- **Carta número 30**

**Mês:** 06
**Dia:** 28
**Título:** Carta do Padre Diogo de Matos
**Resumo:** Padre Diogo de Matos, descreve em pormenor as residências de Gorgora e de Colelã

- **Carta número 31**

**Mês:** 01
**Dia:** 01
**Título:** Carta do Fr. Tomás de Azevedo
**Resumo:** Fr. Tomás de Barros, notícia em pormenor e em latim, as residências de Gorgora, Colelã e Fremona

Figura 4.6: Cartas do ano 1622

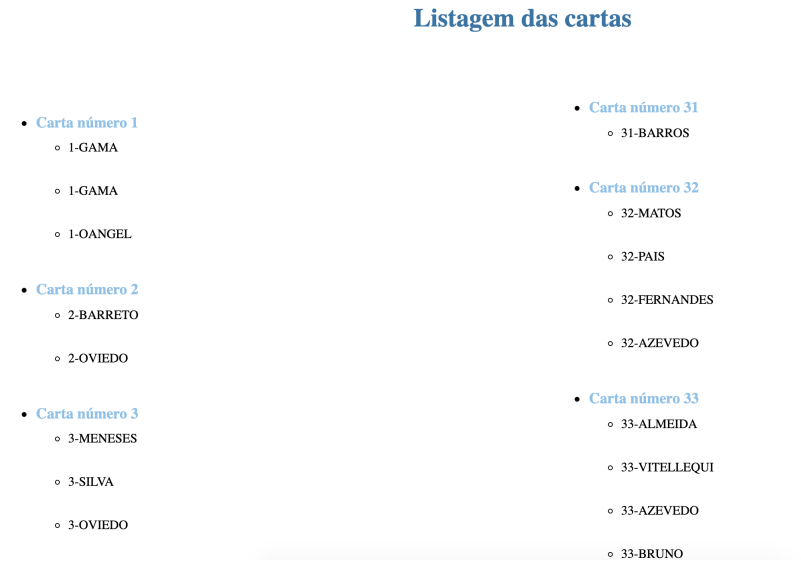


Figura 4.7: Cartas em pares Número-Nome

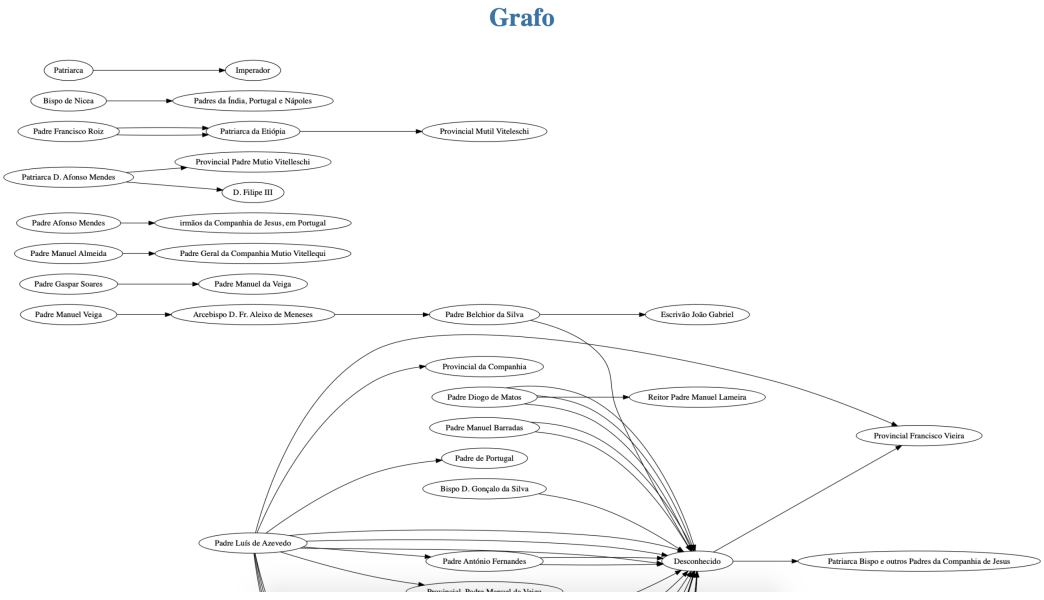


Figura 4.8: Grafo

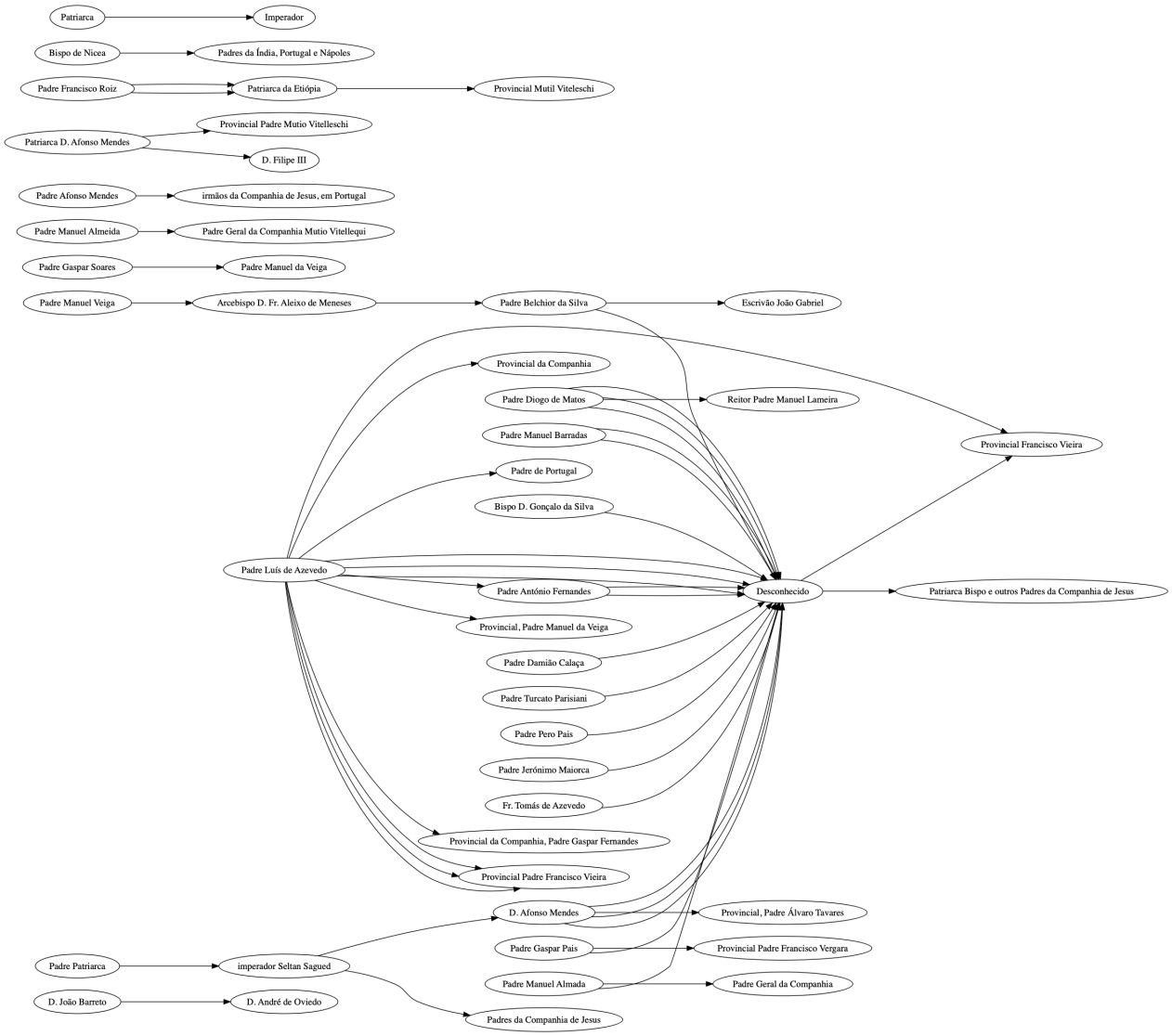


Figura 4.9: Figura do grafo

# Capítulo 5

# Conclusão

Com o desenvolvimento deste projeto conseguimos melhorar os nossos conhecimentos relativamente à linguagem AWK e à construção de filtros com a mesma. Nesses filtros utilizamos bastantes propriedades desta linguagem uma vez que recorremos tanto a operadores relacionais (como o operador ~), como a variáveis internas (como por exemplo o FS), padrões com expressões regulares, com expressões relacionais, compostos, padrões BEGIN e END, instruções de controlo, variáveis auxiliares, arrays e arrays multi-dimensionais, e ainda funções características do AWK, como por exemplo a função split, gsub, sub e gensub. Para além disso, também sentimos a necessidade de criarmos uma função auxiliar.

Todas estes aspetos desta linguagem e processador de padrões foram essenciais para a construção de filtros que respondessem aos requisitos do enunciado.

A utilidade destes filtros foi realmente notória, uma vez que conseguimos, a partir de um ficheiro csv, gerar várias páginas html com toda a informação de forma organizada e apelativa, o que revela que a utilização destes filtros faz realmente a diferença quando o nosso objetivo é apresentar informação, analisar dados, recolher dados, entre muitas outras utilidades.

Através da página inicial do projeto (Figura 4.1) é possível selecionar qualquer uma das questões propostas e visualizar o respetivo resultado. No caso do primeiro requisito, cartas por local e/ou ano, optamos por fazer três diferentes filtros para proporcionar mais informação útil.

Para o requisito da listagem dos anos das cartas, decidimos organizar o resultado em três colunas pois são vários anos e desta forma não será necessário fazer *scroll* à página para os conseguir encontrar a todos. Através desta página, é também possível clicar em algum dos anos e observar a data, o título e o resumo das cartas trocadas nesse mesmo ano.

Temos também a listagem das partas em pares Número-Nome e, por fim, o grafo. O grafo foi o requisito mais desafiante uma vez era o mais diferente de todos os outros que já construímos. Gostamos também bastante de ter ficado a conhecer dot, uma linguagem de descrição de grafos, que é realmente muito poderosa mas simples de ser utilizada.

Com o ficheiro de *input* seria possível realizar muitos outros filtros que permitiram que os dados da coleção de cartas da Etiópia fossem analisados e apresentados de várias outras formas, fornecendo mais funcionalidades ao utilizador.

Concluindo, através de várias pesquisas sobre AWK e análise do ficheiro de *input* conseguimos cumprir todos os requisitos propostos e construir um agradável fluxo de páginas html com toda a informação necessária sobre as cartas da Etiópica, trocadas nos anos de mil seiscentos pelos navegadores portugueses.