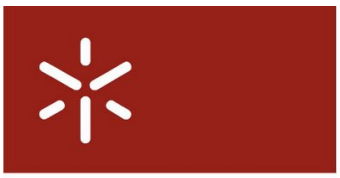


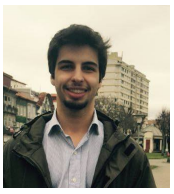
Universidade do Minho

Processamento de Linguagens

MIEI - 3º ano - 2º semestre
Universidade do Minho

Trabalho Prático nº1

GAWK



Dinis Peixoto
A75353

Ricardo Pereira
A74185

Marcelo Lima
A75210

16 de Novembro de 2017

Conteúdo

1 Contextualização **2**

2 Processador de transações da Via Verde **3**
2.1 Enunciado . . . 3
2.2 Descrição do problema . . . 3
2.2.1 a) . . . 3
2.2.2 b) . . . 4
2.2.3 c) . . . 4
2.2.4 d) . . . 4
2.3 Decisões e Implementação . . . 4
2.3.1 a) . . . 4
2.3.2 b) . . . 5
2.3.3 c) . . . 5
2.3.4 d) . . . 6
2.4 Resultado conseguido . . . 6
2.4.1 Código-fonte . . . 6
2.4.2 Visualização . . . 8

3 Autores Musicais **10**
3.1 Enunciado . . . 10
3.2 Descrição do problema . . . 10
3.2.1 a) . . . 11
3.2.2 b) . . . 11
3.2.3 c) . . . 11
3.3 Decisões e Implementação . . . 11
3.3.1 a) . . . 11
3.3.2 b) . . . 12
3.3.3 c) . . . 12
3.4 Resultado conseguido . . . 13
3.4.1 Código-fonte . . . 13
3.4.2 Visualização . . . 14

4 Apreciação crítica **17**

1. *Contextualização*

Este relatório é o resultado do primeiro trabalho ṕratico proposto e elaborado na unidade curricular de Processamento de Linguagens. O trabalho teve por base a utilização do sistema de produção para filtragem de texto GAWK sobre um ficheiro input previamente fornecido, de modo a responder a uma série de alíneas de um dado enunciado. Dos quatro enunciados apresentados, decidimos escolher dois deles, o primeiro designado **Processador de transações da Via Verde** e o terceiro designado **Autores musicais**. O primeiro apresentava como *input* um ficheiro no formato *XML* com o extrato mensal de um dado cliente, enquanto o segundo se tratava de um conjunto de ficheiros com extensão '.lyr', onde se encontravam as letras de várias canções famosas. Deste modo, o objetivo era desenvolver um Processador de Texto com o auxílio do *GAWK*, para ler estes mesmos ficheiros e ao mesmo tempo filtra-los e transforma-los de modo a obter as informações pretendidas.

2. *Processador de transações da Via Verde*

2.1 Enunciado

A Via Verde envia a cada um dos seus utentes um extracto mensal no formato *XML* como se exemplifica no ficheiro anexo *viaverde.xml*. Depois de analisar com cuidado o formato desse ficheiro anexo, pretende-se que desenvolva um Processador de Texto com o *GAWK* para ler um extrato mensal da Via Verde e:

a) calcular o número de *entradas* em cada dia do mês.

b) escrever a lista de locais de *saída*.

c) calcular o total gasto no mês.

d) calcular o total gasto no mês apenas em *parques*.

2.2 Descrição do problema

Tal como dito anteriormente, todo este trabalho se baseia na interpretação, filtragem e transformação de texto, estando este contido num ficheiro previamente fornecido. Neste caso trata-se de um ficheiro em formato *XML*, onde nele se encontram registadas as informações de todas as transacções de um dado cliente no sistema de portagens Via Verde. Como tal, tratando-se de um ficheiro *XML* é normal este conter bastante informação irrelevante aos olhos de pessoas normais, que apenas existe para este estar de acordo com a sua sintaxe, como por exemplo as *tags* que demarcam os vários campos de informação. Deste modo, torna-se complicado e demorado obter a informação pretendida de um destes ficheiros, sem que para isso seja necessário recorrer a ferramentas como o *GAWK*. Assim, toda a informação retirada e processada deste ficheiro com o objetivo de responder às diversas alíneas, será armazenada de forma organizada num ficheiro em formato *HMTL* de modo a tornar a sua leitura bastante mais agradável.

2.2.1 a)

Nesta alínea **a**, o problema seria contabilizar o número de entradas em cada dia do mês. Para conseguir isto seria necessário ter em consideração cada data de uma entrada, interpretando-a de modo a ser-lhe possível associar um número de entradas.

2.2.2 b)

Por sua vez, na alínea **b**, o objetivo passaria por conseguir enumerar os diferentes locais de todo o documento, incluídos no parâmetro *saída*. Para isso seria necessária a consulta deste parâmetro em todas as entradas do documento, adicionando-as a um conjunto de dados que não permitisse, de algum modo, a sua repetição.

2.2.3 c)

O caso da alínea **c** é ainda um pouco distinto dos anteriores, na medida que, neste caso, a barreira a ultrapassar seria associar todas as datas das entradas de modo a conseguir contabilizar assim o total gasto no mês (considerando também a diferença entre entradas registadas em Julho e Agosto), para todas as entradas.

2.2.4 d)

A alínea **d**, por fim, é relativamente parecida com a anterior, uma vez que o processo é, em parte idêntico, excepto na condição final, na qual é necessário ter em conta o parâmetro $<TIPO>$, fazendo corresponder às únicas entradas cujo valor será contabilizado.

2.3 Decisões e Implementação

Para a resolução das várias alíneas, foi necessário estabelecer algumas medidas logo de início. Uma delas foi, alterar o valor do **FS**, para [<>], permitindo que em cada linha do XML, os vários campos estejam repartidos pelos caracteres < e >, isto é, estejam repartidos exatamente no seguinte formato:

$$\$1 <CAMPO(\$2)> \text{CAMPO}(\$3) <CAMPO(\$4)>$$

2.3.1 a)

Para esta alínea, apenas foi necessário em cada linha do XML, reconhecer a tag $<DATA\_ENTREGA >$. Para tal, usamos esta expressão regular:

$$/\hat{}<\text{DATA\_ENTRADA}/$$

Esta expressão regular, encontra todas as expressões no ficheiro XML começadas por $<DATA\_ENTRADA>$, e permite desta forma, recolher os dados contidos entre estas tags (corresponde ao campo \$3). O campo é uma data com o formato: *dd-mm-aaaa*, e como tal, é necessário recolher parte do seu conteúdo. Para isso, usamos a função *split*, que reparte a data nos três campos que possui (dia, mês e ano). Assim sendo, para armezanar os dados, recorremos a um array, onde os indíces deste correspondem a strings do formato: *dd-mm*. Posto isto, para cada dia encontrado (indíce do array), incrementamos o seu valor, permitindo assim determinar quantas entradas ocorreram nesse dia.

2.3.2 b)

Nesta alínea, seguimos o mesmo racíocinio da alínea anterior, isto é, para encontrar todos os locais de saída do ficheiro input é necessário reconhecer a tag *<SAIDA >* usando, para tal, a seguinte expressão regular:

$$/\hat{}< \text{SAIDA}/$$

Esta expressão regular, idêntica à anterior, permite também recolher os dados entre as tags (campo $3). Desta forma, apenas nos limitamos a armazenar o local como índice de um array, incrementando este a cada local igual encontrado.

2.3.3 c)

Ao contrário das alíneas anteriores, esta requer uma análise mais cuidada. Para podermos obter os gastos do mês, e dentro deste, reparti-los pelo mês de ocorrência, é necessário ter uma variável *mes* que armazena o mês de ocorrência. Esta operação ocorre dentro da condição com a expressão regular /ˆ<DATA_ENTRADA/, isto é, no momento em que contabilizamos os dias, actualizamos o mês de ocorrência, já que a tag possui a data completa, e a função *split* nos dá o mês. No entanto, o ficheiro *XML* possuí algumas falhas, nomeadamente, falta de dados entre a tag DATA_ENTRADA, sendo por isso, necessário recorrer como segunda opção à tag DATA_SAIDA. Para tal usamos a expressão regular e condição:

$$/\hat{}< \text{DATA\_SAIDA}/ \;\&\&\; \text{mes}==\text{"null"}$$

A condição de mes=="null", provém, quando na tag DATA_ENTRADA não é encontrado dado nenhum, atribuindo assim o valor *null* para que quando encontrada uma tag DATA_SAIDA, usar a data contida como mês de ocorrência.
Ao fim de obtermos o mês de ocorrência, passamos a recolha dos montantes dos respectivos meses. Para tal, recolhemos as tags IMPORTANCIA, com a seguinte expressão regular:

$$/\hat{}< \text{IMPORTANCIA}/$$

Da mesma forma que nos casos anteriores, recolhemos o campo $3, que nos dá o montante. No entanto, como os montantes são *floats*, e estes diferenciam as casas decimais das unidades por um ".", é necessário substituir, o caracter ",", que é o delimitador que está presente no campo $3. Para isto, usamos a função *sub*, que substitui diretamente o caracter ",", pelo ".", e armazena o valor numa variável *elemento*. Após esta correção, falta apenas armazenar os dados numa estrutura. Para tal, criamos um array, em que os índices são os meses que obtivemos logo de ínicio, e acumulamos os montantes no respectivo mês de ocorrência.

2.3.4 d)

Para conseguir-mos realizar esta alínea, aproveitamos grande parte do realizado já na alínea anterior, tendo em conta que o objetivo é praticamente o mesmo, com a simples adição de um condição de filtragem. A necessidade de filtrar as entradas por *Parques* obrigou à utilização da expressão regular:

$$/\hat{}<\text{TIPO} > [\text{Pp}]\text{arque}/$$

Desta forma, e com as entradas pretendidas filtradas, restava apenas realizar o somatório dos montantes gastos nas respectivas transações, o que é relativamente simples tendo em conta que o processo para obter os montantes já foi realizado na alínea anterior, correspondendo à variável *elemento*.

2.4 Resultado conseguido

2.4.1 Código-fonte

```
<html>
<head>
<meta charset='UTF-8'/>
<style>table, th, td {border: 1px solid black; border-collapse: collapse;}
th, td {padding: 5px;} th {text-align: left;}</style>
</head>
<body>
<h1 align="center"> Extracto ViaVerde </h1>
<hr><h2> Cliente </h2>

<h3> NOME </h3> <p> PEDRO MANUEL RANGEL SANTOS HENRIQUES </p>
<h3> MORADA </h3> <p> RUA XXX </p>
<h3> MES </h3> <p> Ago-2015 </p>
<h3> MATRICULA </h3> <p> 00-LJ-11 </p>
<h3> LOCALIDADE </h3> <p> BRAGA </p>
<h3> CODIGO_POSTAL </h3> <p> 4715-012 BRAGA </p>

<hr>

<h3> Número de Entradas do Mês </h3> <p>  </p>
<table style="width:30%"><tr> <th>Dia</th><th>Número de Entradas</th> </tr>
<tr> <td> 06 de Agosto </td><td> 4 </td> </tr><tr> <td> 10 de Agosto </td>
<td> 7 </td> </tr><tr> <td> 11 de Agosto </td><td> 2 </td> </tr><tr>
<td> 13 de Agosto </td><td> 5 </td> </tr><tr> <td> 17 de Agosto </td>
<td> 4 </td> </tr><tr> <td> 18 de Agosto </td><td> 2 </td> </tr>
<tr> <td> 21 de Agosto </td><td> 4 </td> </tr><tr> <td> 26 de Julho </td>
<td> 3 </td> </tr><tr> <td> 29 de Julho </td><td> 2 </td> </tr><tr>
<td> 30 de Julho </td><td> 3 </td> </tr><tr> <td> 31 de Julho </td>
```

<td> 1 </td> </tr></table>
<h3> Lista de Locais de Saída </h3>
<a> Aeroporto
<a> Angeiras N-S
<a> Braga Sul
<a> Custoias
<a> EN 205 PV
<a> EN107
<a> Ermesinde PV
<a> Ferreiros
<a> Freixieiro
<a> Lipor
<a> Maia II
<a> Maia PV
<a> Neiva N-S
<a> Neiva S-N
<a> PQ A Sa Carn.I
<a> PQ Av. Central
<a> Ponte Pedra
<a> Povoa S-N
<a> Valongo
<h3> Total Gasto </h3>
 Mês 07: 20.3   Mês 08: 57.1 Valor Total : 77.4
<h3> Total Gasto em Parques </h3>
 Valor Total: 6.35 </body> </html>

2.4.2 Visualização

Extracto ViaVerde

Cliente

NOME

PEDRO MANUEL RANGEL SANTOS HENRIQUES

MORADA

RUA XXX

MES

Ago-2015

MATRICULA

00-LJ-11

LOCALIDADE

BRAGA

CODIGO_POSTAL

4715-012 BRAGA

Número de Entradas do Mês

Dia	Número de Entradas
06 de Agosto	4
10 de Agosto	7
11 de Agosto	2
13 de Agosto	5
17 de Agosto	4
18 de Agosto	2
21 de Agosto	4
26 de Julho	3
29 de Julho	2
30 de Julho	3
31 de Julho	1

Lista de Locais de Saída

- Aeroporto
- Angeiras N-S
- Braga Sul
- Custoias
- EN 205 PV
- EN107
- Ermesinde PV
- Ferreiros
- Freixieiro
- Lipor
- Maia II
- Maia PV
- Neiva N-S
- Neiva S-N
- PQ A Sa Carn.I
- PQ Av. Central
- Ponte Pedra
- Povoa S-N
- Valongo

Total Gasto

Mês 07: 20.3 €

Mês 08: 57.1 €

Valor Total : 77.4 €

Total Gasto em Parques

Valor Total: 6.35 €

3. *Autores Musicais*

3.1 Enunciado

Além da coleção de entrevista e fotografias do npMP, o Professor José Jõao Almeida tem uma diretoria (de nome musica, que é anexada em formato ZIP) com dezenas de ficheiros de extensão '.lyr' que contêm a letra de canções famosas precedidas de 2 ou mais campos de meta-informação (1 por linha) com o título da canção, o autor da letra (pode ser 1 ou mais pessoas), o cantor, etc. Uma linha em branco separa a meta-informação da letra. Podendo ainda ter em alguns casos um terceiro bloco (igualmente separado da letra por uma linha em branco) com a música escrita na notação midi. Depois de analisar com cuidado o formato desse ficheiro anexo, pretende-se que desenvolva um Processador de Texto com o GAWK para ler todos os ficheiros '.lyr' da diretoria musica e:

a) calcular o total de cantores e a lista com seus nomes.

b) calcular o total de canções do mesmo autor (mesmo que em alguns casos sejam várias pessoas considere como único).

c) escrever o nome de cada autor seguido do título das suas canções; se mais do que uma, separadas por uma vírgula.

3.2 Descrição do problema

Neste enunciado o procedimento já é um pouco diferente do anterior no que toca à forma de obter e processar a informação. Neste caso temos como *input* toda uma diretoria na qual se encontram dezenas de ficheiros de extensão *.lyr*. Cada um destes ficheiros, para além da letra de uma canção, contém várias outras informações sobre a mesma. Ao contrário do enunciado anterior, toda a informação que se encontra dentro destes ficheiros é considerada importante, uma vez que não existe qualquer tipo de código a organizar esta mesma informação. No entanto, contendo esta diretoria um elevado número de ficheiros, torna-se igualmente difícil retirar informação de todo o seu conjunto sem que para isso tenhamos de recorrer a uma ferramenta como o *GAWK*. Tal como anteriormente, toda a informação necessária para responder às diversas alíneas será devidamente organizada num ficheiro em formato *HTML*.

3.2.1 a)

Nesta alínea **a**, o objetivo seria conseguir o total de cantores diferentes e ainda uma lista destes. Assim, era necessário verificar, para todos os ficheiros *.lyr*, o conteúdo do campo *singer*, caso existisse, interpretando-o e ainda removendo eventuais diferenças entre valores semelhantes. Isto é, para casos em que, para diferentes músicas aparece **Cantor X** e **Cantor X (?)**, ou então em situações em que seria necessária a distinção entre os múltiplos cantores envolvidos na produção de uma música.

3.2.2 b)

Por sua vez, na alínea **b** era exigida a associação de cada música a um ou vários autores. Para tal, foi necessária a interpretação do campo *author*, conseguindo um conjunto de diferentes autores. Posteriormente, era crucial fazer-se corresponder cada ficheiro de música a um autor, incrementando o número de músicas associado a este.

3.2.3 c)

Finalmente, na alínea **c** era pretendido listar o nome de todos os diferentes autores presentes assim como as músicas em que estes se inserem. Para esse fim, e tal como nos exercícios anteriormente realizados, foi necessário associar as músicas aos seus autores, agrupando-as assim, de maneira a ser possível futuramente imprimir toda a informação corretamente.

3.3 Decisões e Implementação

Como no 1º exercício, para a resolução das várias alíneas, foi necessário definir um conjunto de medidas logo de início. Uma destas foi alterar o valor do **FS** para o caracter ':', permitindo que em cada ficheiro *.lyr*, os vários campos da meta-informação estejam repartidos pelo caractere ':', isto é, estejam repartidos exatamente no seguinte formato:

(Identificador)($1) : [Valor1; Valor2; (...)]($2)

3.3.1 a)

Para esta alínea apenas foi necessário reconhecer o campo da meta-informação *singer*, em cada ficheiro '.lyr'. Para tal, foi usada a expressão regular:

/^ singer/

Tendo encontrado este campo (Identificador), o campo $2 irá conter os cantores pretendidos. Com isto, basta utilizar a função *split*, repartindo o campo $2, quando

encontrar um dos seguintes caracteres/conjunto de caracteres: **';', ',' e '(?)'**. Para tal, utilizamos a seguinte expressão regular, como identificador dos diferentes separadores do campo $2:

/[;,'(?)']/

O resultado da função *split* é um array com os vários cantores que um ficheiro *.lyr* possui. Por fim, apenas armazenamos cada cantor num indíce diferente de um array, sendo que antes é aplicada a função *fixSpaceAndChar*, para retirar caracteres indesejados que impossibilitavam a compatibilidade de cantores com o mesmo nome, isto é, *Strings* que contenham espaços ou *tabs*, que outra *String* do mesmo cantor não contenha.

3.3.2 b)

Nesta alínea seguimos o mesmo raciocínio que na alínea anterior. Assim sendo, em cada ficheiro *.lyr*, encontramos o campo da meta-informação *author*, usando a seguinte expressão regular:

/^ author/

Posteriomente, e tal como na alínea anterior, para obter os autores de cada ficheiro utilizamos a função *split*, com a seguinte expressão regular como identificador dos diferentes separadores do campo $2:

/[;,'(?)']/

Obtendo assim um array com o(s) autor(es) intervenientes na respectiva música. De seguida, percorremos este array, aplicando a função *fixSpaceAndChar*, tal como anteriormente. Restava apenas aumentar o valor correspondente ao índice do autor considerado, conseguindo assim um array com o total de canções respectivas a cada autor, e portanto, o resultado pretendido.

3.3.3 c)

Como o título é o primeiro campo da meta-informação, para podermos obter a lista dos títulos em que cada ator participou é necessário repartir tarefas. A primeira tarefa, é encontrar o campo *title*, utilizando a seguinte expressão regular:

/^ title/

Depois é armarzenar o campo $2 (corresponde ao título), numa varíavel chamada *title*. Desta forma, quando processamos o campo *author*, apenas precisamos de adicionar o título guardado na variavel *title* como o valor num array, onde o indice é o autor. No entanto, temos que ter em atenção caso um autor tenha mais que uma participação em diferentes canções. Para isso, no momento de armazenar o título, verificamos se o autor em questão já possui algo ou não no array. Caso já possua, adicionamos ao conteúdo o caracter ',', e de seguida o título e caso não possua adicionamos apenas o título.

3.4 Resultado conseguido

3.4.1 Código-fonte

Devido à extensão dos ficheiros (*listaAutores.html* e *listaCantores.html*), será demonstrado apenas um excerto dos mesmos.

index.html

```
<html>
<head>
<meta charset='UTF-8'/>
<style>table, th, td {border: 1px solid black;
border-collapse: collapse;} th, td {padding: 5px;} th
{text-align: left;} </style>
</head>
<body>
<h1 align="center"> Autores Musicais </h1>
<hr>

<li><a href="listaCantores.html"> Lista de Cantores </a></li>

<li><a href="listaAutores.html"> Lista de Autores </a></li>
</body>
</html>
```

listaAutores.html

```
<html>
<head>
<meta charset='UTF-8'/>
<style>table, th, td
{border: 1px solid black; border-collapse: collapse;}
th, td {padding: 5px;} th {text-align: left;}</style>
</head>
<body>
<h1 align="center"> Autores Musicais </h1>
```

```
<hr>

<h3> Lista de Autores </h3> <p>  </p>
<table style="width:50%"><tr> <th>Autor</th>
<th>Número de Músicas</th><th>Títulos</th> </tr>
<tr> <td> A. Amargo </td><td> 1 </td>
<td>  Há festa na Mouraria </td> </tr><tr>
<td> A. Curto </td><td> 1 </td>
<td>  *Aldeia da roupa branca </td> </tr>
        .
        .
        .
        .
      (...)
```

listaCantores.html

```
<html> <head> <meta charset='UTF-8'/> <style>table, th, td
{border: 1px solid black; border-collapse: collapse;} th, td
{padding: 5px;} th {text-align: left;}</style> </head>
<body>
<h1 align="center"> Autores Musicais </h1>
<hr>

<h3> Lista de Cantores </h3> <p>  </p>
<table style="width:30%"><tr> <th>Cantor</th>
<th>Número de Músicas</th> </tr>
<tr> <td> A. P. Braga </td
td> 1 </td> </tr><tr>
<td> Adriana Calcanhoto </td>
<td> 1 </td> </tr><tr>
        .
        .
        .
        .
      (...)
```

3.4.2 Visualização

Uma vez mais, devido à extensão do resultado produzido, serão apresentados apenas excertos do output conseguido.

index.html

Autores Musicais

- Lista de Cantores
- Lista de Autores

listaAutores.html

Autores Musicais

Lista de Autores

Autor	Número de Músicas	Títulos
A. Amargo	1	Há festa na Mouraria
A. Curto	1	*Aldeia da roupa branca
A. Duarte	1	Há festa na Mouraria
A. P. Braga	3	Canção para desfazer equívocos, O homem e a burla, P'ró que der e vier
Abel Silva	2	Verbos do amor, Festa do interior
Alberto Janes	1	Dar de beber à dor
Alcir Pires Vermelho	1	Canta Brasil
Alcídia Rodrigues	1	*Embuçado

listaCantores.html

Autores Musicais

Lista de Cantores

Cantor	Número de Músicas
A. P. Braga	1
Adriana Calcanhoto	1
Adriano Correia de Oliveira	15
Ala dos Namorados	7
Alberto Ribeiro	1
Alcoolémia	1
Alma Lusa	1
Amália Rodrigues	25
António Calvário	1
António Menano	4

4. *Apreciação crítica*

Durante a realização deste trabalho prático, que foi o primeiro desta unidade curricular, várias foram as etapas que tivemos de passar para chegar ao resultado final.
Apesar da simplicidade e do baixo nível de dificuldade que o trabalho revelou, este foi bastante importante, na medida em que a sua elaboração nos permitiu melhorar a nossa capacidade em escrever Expressões Regulares(ER), e apartir destas desenvolver Processadores de Linguagens Regulares, tendo em vista a filtragem e transformação de textos. Para isto, foi necessário aprender e aprofundar um pouco a linguagem do sistema de produção para filtragem de texto *GAWK* e assim desenvolver as soluções necessárias para cada caso particular das diversas alíneas.
Apesar de no enunciado ser pedido a elaboração de apenas um dos quatro exercícios, achamos por bem realizar dois, uma vez que tivemos relativamente tempo para isso. Outra das funcionalidades que não era pedida no enunciado e no qual nós nos empenhamos em realizar foi a criação de ficheiros HTML com as respostas devidamente organizadas às diversas alíneas, de forma a ser mais cómoda e agradável a sua leitura.
Concluindo, no geral estamos satisfeitos com o trabalho desenvolvido, tendo este se relevado bastante útil no aperfeiçoamento dos nossos conhecimentos no que toca ao tema em questão, à medida que os fomos colocando em prática.