Processamento de Linguagens

Trabalho Prático 1

Relatório de Desenvolvimento

Diogo Machado	Lisandra Silva	Rui Leite
(A75399)	(A73559)	(A75551)

15 de Março de 2017

Resumo

Este documento insere-se no âmbito da disciplina de Processamento de Linguagens e pretende demonstrar os passos efetuados para a realização de um trabalho prático que visa a obtenção de informação pertinente partindo de outros ficheiros fonte. Para tal foram usadas Expressões Regulares para fazer o reconhecimento dos padrões identificados e proceder à transformação pretendida, com recurso ao Sistema de Produção GAWK.

Conteúdo

Capítulo 1

Introdução

Neste trabalho prático pretende-se aplicar os conceitos adquiridos na Unidade Curricular de Processamento de Linguagens associados à utilização de Expressões Regulares para descrever padrões de textos, de forma a tratar/retirar informação presente num ficheiro.

Este relatório visa a documentação do processo de desenvolvimento de um filtro de texto para fazer o reconhecimento de padrões identificados e proceder a transformações pretendidas.

Por opção do grupo de trabalho, decidiu-se responder a dois dos enunciados propostos pela equipa docente, nomeadamente o enunciado 2.1 ("Processador de transações da Via Verde") e, como valorização, o enunciado 2.3 ("Autores Musicais").

Estrutura do Relatório

No Capítulo 2 e Capitulo 3 serão abordados os problemas "Processador de transações da Via Verde"e "Autores musicais", respetivamente. Em cada um destes capítulos é feita uma descrição do problema em causa e uma exposição da implementação da solução, contendo os padrões identificados e as expressões regulares utilizadas, bem como as estruturas de dados escolhidas e ainda os filtros de texto produzidos. No fim de cada capítulo é apresentado para cada problema os testes realizados e respetivos resultados, sendo ainda expostas as alternativas ponderadas e os problemas de implementação que surgiram.
Por último, destacam-se as conclusões que a realização deste trabalho nos permitiu tirar e ainda um apêndice onde se inclui todo o código usado.

Capítulo 2

Processador de transações da Via Verde

2.1 Descrição do Problema

Para este problema pretende-se desenvolver um Processador de Texto com o GAWK para ler um extrato mensal da Via Verde. O extrato mensal da Via Verde é um ficheiro no formato XML e, depois de analisá-lo, o Processador de texto deverá:

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

b) escrever a lista com todos os locais de saída;

c) calcular o total gasto no mês e o total gasto apenas em parques.

De modo a valorizar o trabalho procedeu-se à elaboração de páginas em HTML onde esta informação é apresentada, e o utilizador pode ainda escolher visualizar toda a informação de uma dada transação, ou seja, de um determinado troço percorrido por exemplo.

2.2 Implementação da Solução

Por forma a implementar aquilo a que nos propusemos, realizou-se uma breve análise ao ficheiro `ViaVerde.xml` disponibilizado com o objetivo de identificar padrões e construir expressões regulares eficazes para obtenção de informação importante.

2.2.1 Padrões e Expressões Regulares

Após uma análise cuidada do ficheiro fornecido foi possível observar um conjunto de padrões e criar expressões regulares para poder interpretar a informação nele contido.
Verificou-se então que o extrato possui um *header* com toda a informação que caracteriza o cliente, um conjunto de transações e em rodapé o resumo do extrato.
Uma vez que a informação realmente relevante se encontra nas transações decidiu-se usar como *Field Separator* o $\backslash n$ e como *Record Separator* a expressão regular `<TRANSACCAO>`. Deste modo, cada registo será uma transação, à exceção do primeiro, que será o *header*, e do ultimo, que para além da última transação terá o rodapé.
Exemplo de uma transação:

```
<TRANSACCAO>
<DATA_ENTRADA>26-07-2015</DATA_ENTRADA>
<HORA_ENTRADA>11:33</HORA_ENTRADA>
<ENTRADA>Povoa N-S</ENTRADA>
<DATA_SAIDA>26-07-2015</DATA_SAIDA>
```

```
<HORA_SAIDA>11:42</HORA_SAIDA>
<SAIDA>Angeiras N-S</SAIDA>
<IMPORTANCIA>2,00</IMPORTANCIA>
<VALOR_DESCONTO>0,00</VALOR_DESCONTO>
<TAXA_IVA>23</TAXA_IVA>
<OPERADOR>I. de Portugal (N1)</OPERADOR>
<TIPO>Portagens</TIPO>
<DATA_DEBITO>05-08-2015</DATA_DEBITO>
<CARTAO>6749036</CARTAO>
</TRANSACCAO>
```

A informação segue o padrão: `<TAG>dados<TAG>`. Por exemplo:

```
<DATA_ENTRADA>26-07-2015</DATA_ENTRADA>
```

Assim, optou-se por usar a função split usando a expressão regular `[<>]` de modo a obterem-se os diferentes campos separadamente.

Como todos os valores numéricos do documento estão com uma vírgula (`,`) como separador decimal, teve que se proceder à sua substituição pelo ponto (`.`) para se conseguir fazer cálculos com eles. Para tal fez-se uso da expressão regular `/[0-9],[0-9]/` para encontrar valores numéricas e da função `gsub` para colocar o ponto (`.`).

2.2.2 Estruturas de dados

Para a criação do HTML foi necessário armazenar determinados dados, de modo a que após o processamento do ficheiro de entrada `viaverde.xml` a informação relevante estivesse armazenada e disponível para ser processada e apresentada. Decidiu-se então que um *array* associativo seria uma estrutura eficaz para armazenar estes dados e permitir o acesso rápido e eficiente.

2.2.3 Filtros de Texto

Nesta subsecção são apresentadas as decisões tomadas para a implementação de cada um dos objetivos pedidos no enunciado do trabalho prático e também na valorização referida na secção 2.1 deste relatório.

Foi criado o ficheiro `main.awk`, o qual contempla tudo a que nos propusemos fazer. Como referido anteriormente na secção 2.1, o que foi solicitado no enunciado é apresentado numa pagina HTML. Para isso foi criada a função `criaHTML`.

2.3 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (valores introduzidos) e os respetivos resultados obtidos. Para testar o filtro de texto produzido usou-se o ficheiro fornecido `viaverde.xml`.

A página principal gerada em HTML encontra-se na figura 2.1. Como se pode ver, além da informação pedida decidiu-se também acrescentar os dados do cliente disponibilizados no ficheiro. O total gasto bem como o total gasto em parques são imediatamente apresentados nesta página principal.

Via Verde - Extrato mensal

Dados do cliente

NOME: PEDRO MANUEL RANGEL SANTOS HENRIQUES

NIF: 987653210

MORADA: RUA XXX **CODIGO POSTAL:** 4715-012 BRAGA

MATRÍCULA: 00-LJ-11

Total gasto no mês: 95.20 € c/IVA (Valor de IVA: 17.80 €)

Total gasto em parques: 7.81 € c/IVA (Valor de IVA: 1.46 €)

- Entradas em cada dia do mês
- Lista de saídas registadas

Figura 2.1: Página HTML produzida a partir do Filtro de Texto aplicado ao ficheiro `viaverde.xml`

Para saber as entradas em cada dia do mês e também a lista das saídas foram criadas hiperligações que levam o utilizador para as página respetivas, que permitem saber, por exemplo, quantas entradas houve em cada dia do mês (figura 2.2) e, a hiperligação "aqui"contém a informação completa relativamente a cada uma dessas entradas, como é apresentado na figura 2.3.

Via Verde - Extrato mensal

Entradas por dia

Data	Número de entradas	Ver registos
30-07-2015	3	aqui
18-08-2015	2	aqui
29-07-2015	2	aqui
11-08-2015	2	aqui
10-08-2015	7	aqui
26-07-2015	3	aqui
17-08-2015	4	aqui
06-08-2015	4	aqui
13-08-2015	5	aqui
21-08-2015	4	aqui
31-07-2015	1	aqui

Figura 2.2: Página com o número de entradas para cada dia do mês

Via Verde - Extrato mensal

Data: 31-07-2015

Número total de registos: 1

TAXA_IVA: 23

ENTRADA: Angeiras S-N

OPERADOR: I. de Portugal (N1)

DATA_DEBITO: 07-08-2015

CARTAO: 6749036

VALOR_DESCONTO: 0.00

IMPORTANCIA: 2.00

HORA_SAIDA: 00:29

DATA_ENTRADA: 31-07-2015

SAIDA: Povoa S-N

HORA_ENTRADA: 00:10

TIPO: Portagens

DATA_SAIDA: 31-07-2015

Figura 2.3: Informação da entrada do dia 31-07-2015

Para obter a lista com todas as saídas registadas, o utilizador deverá clicar na hiperligação "Lista de saídas registadas" e será aberta uma página onde é apresentada a referida lista (figura 2.4).

Lista de saídas registadas

Local 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

Figura 2.4: Página onde é apresentada a lista com todas as saídas registadas

2.4 Alternativas, Decisões e Problemas de Implementação

Numa fase inicial ponderou-se a ideia de usar como *Record Separator* o `\n` e como FS `[<>]`, no entanto decidiu-se que este procedimento não seria o mais correto, uma vez que o cabeçalho inicial poderia ter um número indeterminado de linhas e portanto o nosso código deixaria de funcionar noutros ficheiros com uma estrutura diferente. Assim, a opção pela implementação supracitada prendeu-se com o facto de a tornar a mais "universal", pretendendo que seja mais flexível para analisar outros ficheiros que possam ter um cabeçalho diferente.

Como valorização do trabalho também chegou a ser discutido gerar, para cada transação um mapa em que era apresentado o percurso efetuado tendo em conta o local de entrada e o local de saída, no entanto a informação dada nestes dois campos não era suficiente para garantir que o percurso fornecido correspondia ao correto. A título de exemplo, se o local de saída for "Aeroporto"então esta informação não chega para garantir que a API do Google™Maps construa o percurso correto.

Capítulo 3

Autores Musicais

3.1 Descrição do Problema

Pretende-se desenvolver um Processador de Texto com o GAWK para ler todos os ficheiros `.lyr` da diretoria `musica` da coleção de música do Professor José João Almeida e:

a) calcular o total de *cantores* e a lista com os seus nomes;

b) calcular o total de canções do mesmo *autor*;

c) escrever o nome de cada *autor* seguido do título das suas canções.

3.2 Implementação da Solução

Para a implementação da solução pedida, fez-se uma análise leve aos ficheiros da diretoria `musica` e identificada a informação padronizada presente em todos eles, por forma a construir expressões regulares eficazes na obtenção dos dados pretendidos.

3.2.1 Padrões e Expressões Regulares

Cada um dos ficheiros `.lyr` da coleção de música do Professor José João possui a letra de uma canção, precedida de dois ou mais campos de informação (1 por linha), como o *título* da canção, o(s) *autor*(es) da letra e, entre outros, o(s) *cantor*(es) da música. Em todos os ficheiros, uma linha em branco separa a meta-informação da letra da música.
A meta-informação segue os seguintes padrões, que, depois de analisados e mediante o objetivo pretendido, levaram à definição de expressões regulares, capazes de obter e separar a informação pretendida.

1) O tipo de meta-informação é separado por "dois pontos" (`:`) do seu conteúdo, eventualmente com um ou mais espaços, antes e depois, e pode começar por uma letra maiúscula ou por uma letra minúscula.

   Exemplo de `abrunhosa-seEuFosseUmDiaoTeuOlhar.lyr`:

   ```
   title: Se eu fosse um dia o teu olhar
   author: Pedro Abrunhosa
   singer: Bandemónio
   in: Tempo
   from: Nuno
   ```

   Optou-se então pelos dois pontos (`:`) como um *Field Separator*, com a possibilidade de existência de espaços: `/ *: */`. Tal permitiu que a meta-informação de tornasse acessível através do operador `$` do GAWK.

Como o tipo da meta-informação pode começar tanto por uma letra maiúscula como por uma letra minúscula, todas as expressões regulares que sirvam para captar esta informação têm em consideração este facto, como exemplo: `/[Ss]inger/`.

2) Por vezes surgem `*` ou `=` nos títulos das músicas.

   Exemplo de `abrunhosa-viagens.lyr`:

   ```
   title: * Viagens
   ```

   Com isto, decidiu-se proceder à eliminação dos caracteres `*` e `=` do título das canções, com o auxílio da função `gsub` e com a expressão regular `/ *[*=] */`.

3) Existem ficheiros onde não há meta-informação a respeito do autor da letra.

   Exemplo de `beatrizCosta-agulhaEODedal`:

   ```
   title: A agulha e o dedal
   singer: Beatriz Costa
   from: cancioneiro do OUP (Orfeão Universitário do Porto)
   ```

4) Por vezes surgem ficheiros onde o autor da letra é desconhecido e representado por um ponto de interrogação (`?`) ou por "anónimo".

   Exemplo de `brasil-modinha.lyr`:

   ```
   title: Modinha
   from: José Roberto Molina
   author: ?
   singer: ?
   ```

   Por escolha do grupo de trabalho, decidiu-se substituir o ponto de interrogação (`?`) e a palavra "anónimo" pela palavra "Desconhecido", através da expressão regular e `/^\?|an[óo]nimo/` aplicada na função `gsub`.

5) Podem existir músicas com mais do que um cantor ou mais do um autor, separados por um ponto e vírgula (`;`) ou por uma vírgula (`,`).

   Exemplo de `abrunhosa-tudoOQueEuTeDou.lyr`:

   ```
   title: Tudo o que eu te dou
   author: Pedro Abrunhosa; Bandemónio
   from: ard@cheeta.inesc.pt (Alfredo Domingues) (jeito de jj)
   ```

   No caso de querer separar os cantores (*singers*) quando estão especificados mais do que um, fez-se uso do ponto e vírgula (`;`) e da vírgula (`,`) como *Field Separator*, também com a possibilidade de espaços, conforme a expressão regular `/ *; *| *, */`.

   No caso de não interessar fazer a separação dos nomes por diferentes campos (*fields*), definiu-se a vírgula como separador textual e, portanto, procedeu-se à substituição do ponto e vírgula (`;`) pela vírgula (`,`).

6) Alguns do *cantores* possuem um ponto de interrogação entre parêntesis `(?)`. Assumiu-se que tal facto significa que não existe certeza de qual que o cantor é de facto o especificado e, por uma questão de organização, decidiu-se manter essa especificação.

7) Podem ocorrer diferentes variações de nomes que sejam equivalentes, como é o caso de: `Xutos e Pontapés` ou `Xutos & pontapés`. Decidiu-se substituir o carácter `&` por `e`.

   Para tal fez-se uso da função `gsub` com a Expressão Regular `/&/`.

8) Podem existir espaços no final de uma linha.

   Por forma a eliminar os espaços, usou-se a expressão regular `/ $/`.

3.2.2 Estruturas de dados

Para a construção dos filtros de texto pretendidos, fez-se uso do *array* como estrutura de dados principal. A facilidade com que é construído ou preenchido e pela forma como pode ser percorrido e imprimidos os seus valores, o *array* foi a decisão certa para a implementação de determinados objetivos, tais como: listar todos os cantores ou até associar a cada autor todas as suas músicas (com um *array* bidimensional).

O facto de ser possível a existência de *arrays* associativos, onde a chave é uma qualquer *String*, também facilitou na implementação dos filtros de texto.

3.2.3 Filtros de Texto

Nesta subsecção são apresentadas as decisões tomadas para a implementação de cada um dos objetivos pedidos no enunciado do trabalho prático e também apresentados na secção 3.1 deste relatório.

A implementação dos filtros de texto fez-se com a construção dos seguintes ficheiros:

- **musA.awk** - para a primeira alínea
- **musB.awk** - para a alínea b)
- **musC.awk** - para a última alínea

Alínea a) - calcular o total de cantores e a lista com os seus nomes

Para a implementação deste filtro de texto decidiu-se especificar o *Field Separator* como `"*:  *| *; *| *, *"`. Isto porque no caso de existirem mais do que um *cantor* para uma mesma música pretende-se que estes sejam separados e não considerados como um único.

Quando numa linha o campo número um (`$1`) seguir a expressão regular `/[Ss]inger/`, os seguintes campos (`$i`, com `i` = [2,`NF`]), são guardados num *array* associativo `singers` onde a chave é o nome do cantor.

Depois de percorridos todos os ficheiros, o *array* `singers` é ordenado através da função `asort` e os seus elementos são todos imprimidos no ecrã.

Alínea b) - calcular o total de canções do mesmo *autor*;

Para a implementação deste filtro de texto decidiu-se especificar o *Field Separator* como `"*:  *"`. Isto porque, no caso de existir mais do que um autor para uma mesma música, pretende-se que estes sejam considerados como um único, tal como especificado no enunciado do trabalho prático.

Quando numa linha o campo número um (`$1`) seguir a expressão regular `/[Aa]uthor/`, o campo número dois (`$2`) de cada ficheiro (o *autor* da música) é guardado num *array* associativo `authors` onde a chave é o nome do autor e o valor é incrementado sempre que encontrada uma nova música do mesmo.

Depois de percorridos todos os ficheiros, os índices e o valores do *array* `authors` são imprimidos no ecrã sob a forma: `autor - número de ocorrências`.

Alínea c) - escrever o nome de cada *autor* seguido do título das suas canções;

Para a implementação deste filtro de texto decidiu-se especificar o *Field Separator* como `"*:  *"`. Isto porque, no caso de existirem mais do que um autor para uma mesma música, pretende-se que estes sejam considerados como um único.

Quando numa linha o campo número um (`$1`) seguir a expressão regular `/[Tt]itle/`, o campo número dois (`$2`) de cada ficheiro (o *título* da música) é guardado num *array* `titulos`, em que no índice `i` = 0 está o título da música do primeiro ficheiro lido, no índice `i` = 1 está o título da música do segundo ficheiro lido e assim por diante.

Quando numa linha o campo número um (`$1`) seguir a expressão regular `/[Aa]uthor/`, o campo número dois (`$2`) de cada ficheiro (o *autor* da música) é guardado num *array* `authores`, em que os índices seguem a mesma política do *array* `titulos`.

Depois de percorridos todos os ficheiros, é criado um *array* bidimensional `musicas` que permite relacionar cada *autor* com os *títulos* de todas as suas músicas: `musicas[author][j]` é o *título* da música no índice `j` do *autor* `author`; `musicas[author]` é um *array* com todas as músicas do *autor* `author`.

No enunciado é pedido que seja imprimido o nome de cada *autor*, seguido de todas as suas músicas separadas por uma vírgula (`,`). No caso da última música, decidiu-se colocar um ponto final (`.`).

3.3 Testes e Resultados

De seguida são apresentados alguns testes feitos e os respetivos resultados obtidos. Para o teste dos Filtros de Texto produzidos usaram-se todos os ficheiros fornecidos na pasta `musica`.
Para cada uma das alíneas é apresentado o *output* bem como o comando da `Makefile` usado para a sua obtenção.

Alínea a) - calcular o total de cantores e a lista com os seus nomes

```
$ make musicaA
```

```
A. P. Braga
Adriana Calcanhoto
Adriano Correia de Oliveira
Adriano Correia de Oliveira (?)
Ala dos Namorados
Alberto Ribeiro
Alcoolémia
Alma Lusa
Amália Rodrigues
António Calvário
António Menano
António Variações
Banda do Casaco
Bandemónio
Bando dos Gambozinos
Beatriz Costa
Beatriz Costa (?)
Boris Vian
Brigada Vitor Jara
Brigada Vitor Jara (?)
Caetano Veloso
Carlos Mendes
Carlos Paião
Carlos do Carmo
Cebola Mol
Cesária Évora
Chico Buarque
Conjunto António Mafra
Delfins
Desconhecido
Despe e Siga
Duarte Mendes
Dulce Pontes
Duo Ouro-Negro
Edmundo Bettencourt
Eduardo Nascimento
Elis Regina
Emanuel
Enapá2000
Fausto
```

(Apenas parte do resultado)

Alínea b) - calcular o total de canções do mesmo *autor*;

```
$ make musicaB
```

```
Sétima Legião (?) - 2
João Aguarela - 1
Aníbal Nazaré, Nelson de Barros - 1
José Galhardo, Raul Ferrão - 1
Carlos Guerreiro - 1
Biafra - 1
Ary Barroso - 2
Amadeu do Vale, Frederico Valério - 1
Carlos Lira - 1
Tozé Brito - 1
Paulo Gonzo - 1
Vicente Paiva - 1
Vinicius de Moraes, Carlos Lira - 1
José Mário Branco (?) - 1
Milton Nascimento - 9
Tom Jobim - 2
Sérgio Godinho - 48
Mafalda Veiga - 6
Miguel Ângelo, Fernando Cunha - 4
A. Amargo, A. Duarte - 1
Fernando Santos, Carlos Dias - 1
Tom Jobim, Dolores Duran - 1
Erasmo Carlos - 1
Zé Ketti - 2
Belchior, Tuca - 1
Henricão, Rubens Campos - 1
José Cid - 2
José Luís Tinoco - 1
Edu Lôbo, Oduvaldo Viana Filho - 1
Luís Represas, Cristina Represas - 1
```

(Apenas parte do resultado)

Alínea c) - escrever o nome de cada *autor* seguido do título das suas canções;

```
$ make musicaC
```

```
Jorge Palma -
      O bairro do amor,
      Balada dum estranho,
      Boletim Meteorológico,
      Dá-me lume,
      Deixa-me rir,
      Estrela do Mar,
      O lado errado da noite.
José João Melo -
      Blues on de road to Porto,
      O rei dos matraquilhos.
Alfredo Duarte; Amália Rodrigues -
      Estranha forma de vida.
José Barata Moura -
      Com a prenda do padrinho,
      Joana come a papa,
      Fungágá da bicharada,
      Olha a bola Manel,
      Resolveram fazer esta canção,
      Madalena.
Sétima Legião (?) -
      Navegar,
      Por quem não esqueci.
João Aguarela -
      A cabana do Pai Tomás.
Joaquim de Almeida; Luís Represas -
      Enquanto.
Carlos Guerreiro -
      Era não era do tamanho de um pardal.
Biafra -
      Sonho de Ícaro.
Ary Barroso -
      Aquarela do Brasil,
      Folha morta.
Carlos Lira -
      Feio, não é bonito.
José Galhardo; Raul Ferrão -
      Coimbra.
Tozé Brito -
      Olá, tu por aqui (?).
```

(Apenas parte do resultado)

3.4 Alternativas, Decisões e Problemas de Implementação

Numa fase inicial considerou-se fazer uma alteração ao enunciado no que diz respeito ao facto de considerarmos como tendo um só *autor* as músicas que possuem vários separados por vírgulas ou pontos e vírgula. Desistimos desta ideia pela implementação da alínea c), pois aumentaria o grau de dificuldade e a informação ficaria repetida no que toda à associação entre músicas e *autores*.

Como valorização do trabalho considerou-se gerar ficheiros em HTML para responder ao enunciado, porém, por uma questão de tempo e também por já termos demonstrado esta capacidade no enunciado 2.1 - "Processador de transações de Via Verde", não o fizemos.

Conclusão

Este trabalho permitiu-nos consolidar os conhecimentos sobre Expressões Regulares e sobre GAWK.

Podemos constatar que nem sempre é fácil encontrar o padrão e optar por uma expressão regular que seja a mais eficiente para capturar esse padrão.

Por vezes a heterogeneidade dos ficheiros-fonte é um problema, sendo necessárias, técnicas que permitam a sua normalização, de modo a que, posteriormente, o filtro de texto possa ser efetuado com mais eficiência. Apesar de nenhum dos problemas que encontramos nos tenha levado a essa normalização, ela chegou a ser considerada. Depois, chegou-se à conclusão de que compensava o uso de uma expressão regular mais completa e que eliminasse os caracteres não pretendidos.

Conclui-se também que os arrays associativos constituem uma forma rápida e eficiente de armazenar informação que pretendamos aceder posteriormente. O facto de se poder usar como índice uma string também facilitou imenso o esforço de programação para resolver as alíneas pretendidas.

Apêndice A

Código dos Filtros de Texto

A.1 Processador de transações de Via Verde - main.awk

```
BEGIN {
    FS = "\n";
    RS = "<TRANSACCAO>";
    r = 0;
}

$0 ~ /[0-9],[0-9]/ {
    gsub(",", ".", $0);
}

(NR == 1) {
    split($0, tmp, /<\/?NOME>/);
    dados["NOME"] = tmp[2];
    split($0, tmp, /<\/?NIF>/);
    dados["NIF"] = tmp[2];
    split($0, tmp, /<\/?MORADA>/);
    dados["MORADA"] = tmp[2];
    split($0, tmp, /<\/?CODIGO_POSTAL>/);
    dados["CODIGO_POSTAL"] = tmp[2];
    split($0, tmp, /<\/?MATRICULA>/);
    dados["MATRICULA"] = tmp[2];
}

(NR > 1) {
    split($0, tmp, /[<>]/);
    for (c = 2; tmp[c] != "/TRANSACCAO"; c += 4) {
        transacoes[r][tmp[c]] = tmp[c + 1];
    }
    r++;
}

END {
    criaHTML(dados, transacoes);
}

function imprimeDadosCliente(dados, ficheiro) {
```

```
    print("<p><b>Dados do cliente</b></p>") > ficheiro;
    print("<p><b>NOME: </b>" dados["NOME"]"</p>") > ficheiro;
    print("<p><b>NIF: </b>" dados["NIF"]"</p>") > ficheiro;
    print("<p><b>MORADA: </b>" dados["MORADA"]) > ficheiro;
    print("<b>CODIGO POSTAL: </b>" dados["CODIGO_POSTAL"]"</p>") > ficheiro;
    print("<p><b>MATRÍCULA: </b>" dados["MATRICULA"]"</p>") > ficheiro;
}

function criaHTML(dados, transacoes) {
    title = "Via Verde - Extrato mensal";
    img = "<p><img src =
    ↪ http://www.meiosepublicidade.pt/wp-content/uploads/2015/10/ViaVerde_V_c-assin.jpg
    ↪ align=""right"" width=""708"" height=""414"" ></p>";
    enc = "<html> <head> <meta charset = 'UTF-8'/>" "<title>" title "</title> </head>";
    print enc > "index.html";
    cabecalho = "<body> <h1>" title "</h1>";
    fmt = "<li> <a href='%s'> %s </a></li>\n";

    print cabecalho > "index.html";
    print (img) > "index.html";
    print("<p>_____</p>") >
    ↪ "index.html";
    imprimeDadosCliente(dados, "index.html");
    print("<p>_____</p>") >
    ↪ "index.html";
    imprimeValorGasto(transacoes, "index.html");
    imprimeValorGastoParques(transacoes, "index.html");
    print("<p>_____</p>") >
    ↪ "index.html";
    printf (fmt, "entradasDia.html", "Entradas em cada dia do mês") > "index.html";
    print("<p></p>") > "index.html";
    printf (fmt, "listaSaidas.html", "Lista de saídas registadas") > "index.html";
    print("<p>_____</p>") >
    ↪ "index.html";
    print ("</body></html>") > "index.html";

    print enc > "entradasDia.html";
    print cabecalho > "entradasDia.html";
    print ("<h2>""Entradas por dia""</h2>") > "entradasDia.html";
    print("<p>_____</p>") >
    ↪ "entradasDia.html";
    print (img) > "entradasDia.html";
    imprimeNEntradasPorData(transacoes, "entradasDia.html");
    print("<p>_____</p>") >
    ↪ "entradasDia.html";
    print ("</body></html>") > "entradasDia.html";

    print enc > "listaSaidas.html";
    print cabecalho > "listaSaidas.html";
    print ("<h2>""Lista de saídas registadas""</h2>") > "listaSaidas.html";
    print("<p>_____</p>") >
    ↪ "listaSaidas.html";
    print (img) > "listaSaidas.html";
    imprimeLocaisSaida(transacoes, "listaSaidas.html");
```

```
    print("<p>_____</p>") >
    ↪ "listaSaidas.html";
    print ("</body></html>") > "listaSaidas.html";

}

function imprimeNEntradasPorData(transacoes, ficheiro) {
    title = "Via Verde - Extrato mensal";
    img = "<p><img src =
    ↪ http://www.meiosepublicidade.pt/wp-content/uploads/2015/10/ViaVerde_V_c-assin.jpg
    ↪ align=""right"" width=""708"" height=""414"" ></p>";
    enc = "<html> <head> <meta charset = 'UTF-8'/>" "<title>" title "</title> </head>";
    cabecalho = "<body> <h1>" title "</h1>";
    fmt = "<li> <a href='%s'> %s </a></li>\n";

    for (i in transacoes) {
        entrada = transacoes[i]["DATA_ENTRADA"];
        if (entrada != "null") {
            entradas[entrada]++;
        }
    }

    print ("<p><b>  Data       Número de entradas
    ↪       Ver registos</b></p>") > ficheiro;
    for (data in entradas) {
        print("<p>" data "        " entradas[data]
        ↪ "           ") > ficheiro;
        print("<a href='"data".html'> aqui </a></p>") > ficheiro;

        print enc > data".html";
        print cabecalho > data".html";
        print img > data".html";
        print "<h2>Data: " data "</h2>" > data".html";
        print "<p><b>Número total de registos: </b>" entradas[data] "</p>" > data".html";
        print("<p>_____</p>") >
        ↪ data".html";
    }

    for (i in transacoes) {
        imprimeTransacao(transacoes[i], transacoes[i]["DATA_ENTRADA"]".html");
    }

    for (data in entradas) {
        print("<p>_____</p>") >
        ↪ data".html";
        print ("</body></html>") > data".html";
    }

}

function imprimeTransacao(transacao, ficheiro) {
    for (i in transacao) {
        print("<p><b>" i ": </b>" transacao[i] "</p>") > ficheiro;
    }
```

```
    print("<p>-----------------------------------------------------------------------</p>") >
    ↪ ficheiro;
}

function imprimeLocaisSaida(transacoes, ficheiro) {
    for (i in transacoes) {
        saida = transacoes[i]["SAIDA"];
        if (saida != "null") {
            saidas[saida] = saida;
        }
    }
    print ("<p><b>Local de saída</b></p>") > ficheiro;
    n = asort(saidas);
    for (i = 1; i <= n; i++) {
        print ("<p>" saidas[i] "</p>") > ficheiro;
    }
}

function imprimeValorGasto(transacoes, ficheiro) {
    valor = 0;
    iva2 = 0;
    for (i in transacoes) {
        preco = transacoes[i]["IMPORTANCIA"];
        desconto = transacoes[i]["VALOR_DESCONTO"];
        iva = (transacoes[i]["TAXA_IVA"] / 100) + 1;
        iva2 += preco * (iva - 1);
        valor += (preco - desconto) * iva;
    }
    printf ("<p><b>Total gasto no mês: </b> %0.2f € c/IVA    (Valor de IVA: %0.2f
    ↪ €)</p>", valor, iva2) > ficheiro;
}

function imprimeValorGastoParques(transacoes, ficheiro) {
    valor = 0;
    iva2 = 0;
    for (i in transacoes) {
        tipo = transacoes[i]["TIPO"];
        if (tipo ~ /[Pp]arque/) {
            preco = transacoes[i]["IMPORTANCIA"];
            desconto = transacoes[i]["VALOR_DESCONTO"];
            iva = (transacoes[i]["TAXA_IVA"] / 100) + 1;
            iva2 += preco * (iva - 1);
            valor += (preco - desconto) * iva;
        }
    }
    printf ("<p><b>Total gasto em parques: </b> %0.2f € c/IVA    (Valor de IVA:
    ↪ %0.2f €)</p>", valor, iva2) > ficheiro;
}
```

A.2 Autores Musicais

A.2.1 musA.awk

```
BEGIN {
    FS = " *: *| *; *| *, *";
}

{
    gsub(/ $/, "", $0);
}

$1 ~ /[Ss]inger/ {
    gsub(/^\?/, "Desconhecido", $2);
    gsub(/&/, "e", $2);
    for (i = 2; i <= NF; i++) {
        singers[$i] = $i;
    }
}

END {
    n = asort(singers);
    for (i = 1; i <= n; i++) {
        print (singers[i]);
    }
    print("Total de cantores: " n);
}
```

A.2.2 musB.awk

```
BEGIN {
    FS = " *: *";
}

{
    gsub(/ $/, "", $0);
}

$1 ~ /[Aa]uthor/ {
    gsub(/ *; */, ", ", $2);
    gsub(/ *, */, ", ", $2);
    gsub(/^\?|an[óo]nimo/, "Desconhecido", $2)
    gsub(/popular.*/, "Popular", $2);
    authors[$2]++;
}

END {
    for (i in authors) {
        print i " - " authors[i];
    }
}
```

A.2.3 musC.awk

```
BEGIN {
    FS = " *: *";
    i = 0;
}

{
    gsub(/ $/, "", $0);
}

$1 ~ /[Tt]itle/ {
    gsub(/ *[*=] */, "", $2);
    titulos[i] = $2;
}

$1 ~ /[Aa]uthor/ {
    gsub(/^\?/, "Desconhecido", $2);
    authores[i] = $2;
}

ENDFILE {
    i++;
}

END {
    for (j = 0; j < i; j++) {
        musicas[authores[j]][j] = titulos[j];
    }
    for (autor in musicas) {
        print (autor " - " );
        n = 0;
        for (j in musicas[autor]) {
            n++;
            if (n < length(musicas[autor])) {
                print ("      " musicas[autor][j]",");
            } else {
                print ("      " musicas[autor][j]".");
            }
        }
    }
}
```