

Processamento de Linguagens (3º ano de Curso)

Trabalho Prático 1

Relatório de Desenvolvimento

José Silva
(A74601)

Pedro Cunha
(A73958)

Gonçalo Moreira
(A73591)

13 de Março de 2017

Resumo

Documentação do primeiro trabalho prático da unidade curricular de "Processamento de Linguagens", o principal foco incide sobre a utilização de métodos e ferramentas capazes de solucionar processamento de textos complexos utilizando poucas linhas de código. Por outro lado, em linguagens de programação como C e Pascal, solucionar os mesmos problemas acarreta um maior nível de complexidade. Demonstrando e documentando a solução proposta pelo grupo de trabalho para o problema em concreto, termina-se o relatório com uma análise argumentativa sobre a eficiência dessa mesma solução.

Conteúdo

1	Introdução	2
2	Análise e Especificação	3
2.1	Descrição informal do problema	3
2.2	Especificação dos Requisitos	3
2.2.1	Dados	3
2.2.2	Pedidos	3
3	Concepção/desenho da Resolução	4
3.1	Estruturas de Dados	4
3.2	Algoritmos	4
4	Codificação e Testes	5
4.1	Alternativas, Decisões e Problemas de Implementação	5
4.2	Testes realizados e Resultados	5
4.2.1	O programa é executado com o ficheiro como argumento	5
4.2.2	Obtém-se o seguinte resultado	6
4.2.3	Exemplo da tab que abre quando se clica no local Ferreiros	8
5	Conclusão	9
A	Código do Programa	10

Capítulo 1

Introdução

O avanço tecnológico dos últimos anos trouxe consigo a inevitabilidade de processar cada vez mais texto. Por parte de grande parte dos utilizadores existe a necessidade frequente de fazer mudanças ou extrair determinadas linhas de grandes quantidades de texto onde certos padrões são bastante evidentes. O uso de expressões regulares, que proporcionam um método eficiente, poderoso e flexível no que toca ao processamento de texto, combinado com as ferramentas que a linguagem AWK (linguagem de programação bastante mais fácil de utilizar que as linguagens mais convencionais) proporciona um método eficiente para solucionar as necessidades descritas acima. Das ferramentas descritas anteriormente destacam-se funções capazes de manipular strings e a utilização de arrays associativos, estruturas de dados bastante úteis e que não são disponibilizadas por todas as linguagens de programação.

Neste primeiro trabalho prático da unidade curricular de "Processamento de Linguagens", através dos meios descritos anteriormente, vai desenvolvido um filtro de texto capaz de realizar o processamento de transações presentes nos extratos mensais disponibilizados pela empresa Via Verde.

Estrutura do Relatório

No capítulo 1 faz-se uma pequena introdução ao problema e às ferramentas utilizadas para a resolução deste. Para além disso, é descrita de uma forma breve a estrutura do relatório.

No capítulo 2 faz-se uma análise breve mas mais detalhada do problema escolhido pelo grupo de trabalho.

No capítulo 3 é descrito de uma forma sumariada como procedemos para solucionar as várias questões propostas pelo enunciado.

No capítulo 4 são apresentados alguns testes e respectivos resultados para comprovar o respectivo funcionamento da solução apresentada.

Finalmente, no capítulo 5 termina-se o relatório com uma síntese do que foi dito, as conclusões e o trabalho futuro.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

É fornecido um ficheiro xml, que corresponde ao extrato mensal emitido pela Via Verde para um dos seus utentes. Pretende-se que se desenvolva um *Processador de Texto* para ler esse mesmo ficheiro e retirar a informação requisitada, apresentada em mais detalhe a baixo, na Especificação dos Requisitos.

2.2 Especificação dos Requisitos

2.2.1 Dados

Como já foi referido, é fornecido um ficheiro xml com informação correspondente ao extrato mensal emitido pela Via Verde para um dos seus utentes. Este ficheiro contém, no início, informação do utente e também especifica o mês de emissão. Depois destes dados, seguem-se todas as transações do cliente. Cada uma destas transações, contém a data, as horas de entrada e saída, o local de entrada e saída, a importância paga, o desconto, o iva, o operador e ainda o tipo de transação (Portagens ou Parques de estacionamento). No final do ficheiro, é apresentado o total gasto no mês de emissão e ainda o valor que diz respeito ao iva.

2.2.2 Pedidos

O utente pode ter efetuado entrada em alguns dos dias do mês a que este extrato se refere. Posto isto, é solicitado que se calcule o número de entradas em cada dia do mês. Considera-se relevante obter a lista de locais de saída pelos quais o utente passou durante todo o mês. Chegando ao fim do mês, uma das informações mais importantes para o utente é, muito provavelmente, o total gasto nesse mesmo mês. É então pedido que se obtenha o total que o utente gastou. Pretende-se que seja calculado quanto desse total é que corresponde ao valor despendido apenas em parques de estacionamento.

Para obter a informação necessária será utilizado o Sistema de Produção GAWK, especificando os padrões de frases que se pretende encontrar com recurso a expressões regulares.

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

Pensando nos requisitos para este projeto, é fácil de perceber que será necessário guardar informação relativa a alguns tópicos. Por exemplo, será crucial guardar a informação que diz respeito à quantidade de entradas num determinado dia. Para guardar informação deste tipo, serão utilizados arrays associativos. Embora apenas seja requisitado que se liste os diferentes locais de saída, serão utilizados arrays associativos de forma a guardar quantas vezes passou por cada um desses locais. Relativamente a cada dia, serão ainda guardados os totais gastos e as durações totais de viagens(em arrays associativos) que, embora não seja um requisito, se considera informação importante. Acrescentando ainda aos extras já apresentados, armazena-se informação relativa ao valor despendido em parques e também em portagens. Informação como o total gasto no mês não precisa de ser guardada em nenhuma estrutura de dados, bastando usar um float para o efeito.

3.2 Algoritmos

No início do programa, serão atribuídos todos os valores necessários a algumas variáveis que serão uteis para o corpo do programa. Por exemplo strings, inteiros e formatos para a função printf. É muito importante a definição de um *Field Separator* adequado para facilitar o processamento do ficheiro.

O corpo do programa será na forma de condição e ação respetiva, como é típico usando GAWK. Na maioria das condições, estão especificadas expressões regulares que um ou mais 'fields' da linha atual tem que satisfazer. Sabendo que esse 'field' satisfaz a expressão regular, guarda-se informação valiosa para o resultado final do programa. Uma das ações terá que ser apoiada por uma função cujo objetivo será calcular a duração de uma viagem, recebendo como argumentos a hora de entrada e hora de saída. Esta função partirá do princípio que, no máximo, uma viagem começa num dia e acaba no seguinte.

Chegando à fase final de execução trata-se a informação obtida, de forma a apresentar os resultados em formato apelativo. Por exemplo, usando html com strings definidas no início e não só. Existirá, muito provavelmente, um ciclo para quase todos os arrays associativos de forma a apresentar a informação armazenada no mesmo. Um destes ciclos, deverá juntar a quantidade de entradas num dia e o total gasto no mesmo. Depois de tratar toda a informação, no caso de ser usado html, coloca-se o final no ficheiro fechando o body e o html.

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

Uma das decisões tomadas logo no início da implementação foi usar html como tipo de ficheiro de 'output'. Este ficheiro utiliza estilos escritos em css. Também é utilizada a font-awesome para juntar icons que refletem o conteúdo de cada linha apresentada. O objetivo do uso destes recursos é tornar o conteúdo do ficheiro apelativo. Uma alternativa a esta abordagem seria utilizar o standard output para apresentar os resultados, mas considerou-se que o html 'ganhava' em tudo, excepto na complexidade de construção. Tendo apenas esse problema, que não causa muito transtorno, achou-se por bem optar pelo html. Como se pode verificar no bloco begin(linhas 3 a 24), foram criados vários formatos e strings para auxiliar na criação do ficheiro.

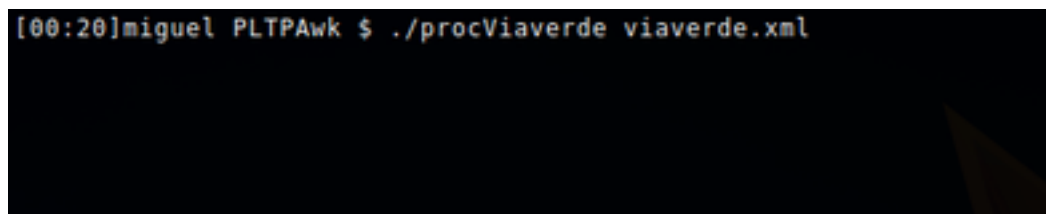
Das decisões tomadas, a mais importante foi, talvez, decidir qual *Field Seperator* usar. Depois de observar bem o ficheiro xml, chegou-se à conclusão que seria oportuno utilizar como *Field Seperator* a expressão regular na linha 4 do código: [$\<\>$]. Assim, é possível obter o que está dentro de cada elemento xml apenas acedendo à posição 3. Isto facilita muito a obtenção de informação. Com outra abordagem, podia ter sido usado o *Field Seperator* default, mas dessa forma seria necessário um tratamento muito mais complexo das strings em cada posição para retirar dados importantes. Relativamente aos locais de saída, decidiu-se juntar uma referência que pesquisa o nome desse local no google maps e apresenta o resultado numa nova tab do browser utilizado para abrir o html gerado.

Durante a implementação do programa, não houve nenhum problema que mereça destaque. Os algoritmos utilizados são relativamente simples e as expressões regulares também. A expressão mais 'complexa' utilizada, foi a que garante que a data de entrada seja no formato correto(removendo ocorrências de null). Esta expressão está presente na linha 57 do código.

4.2 Testes realizados e Resultados

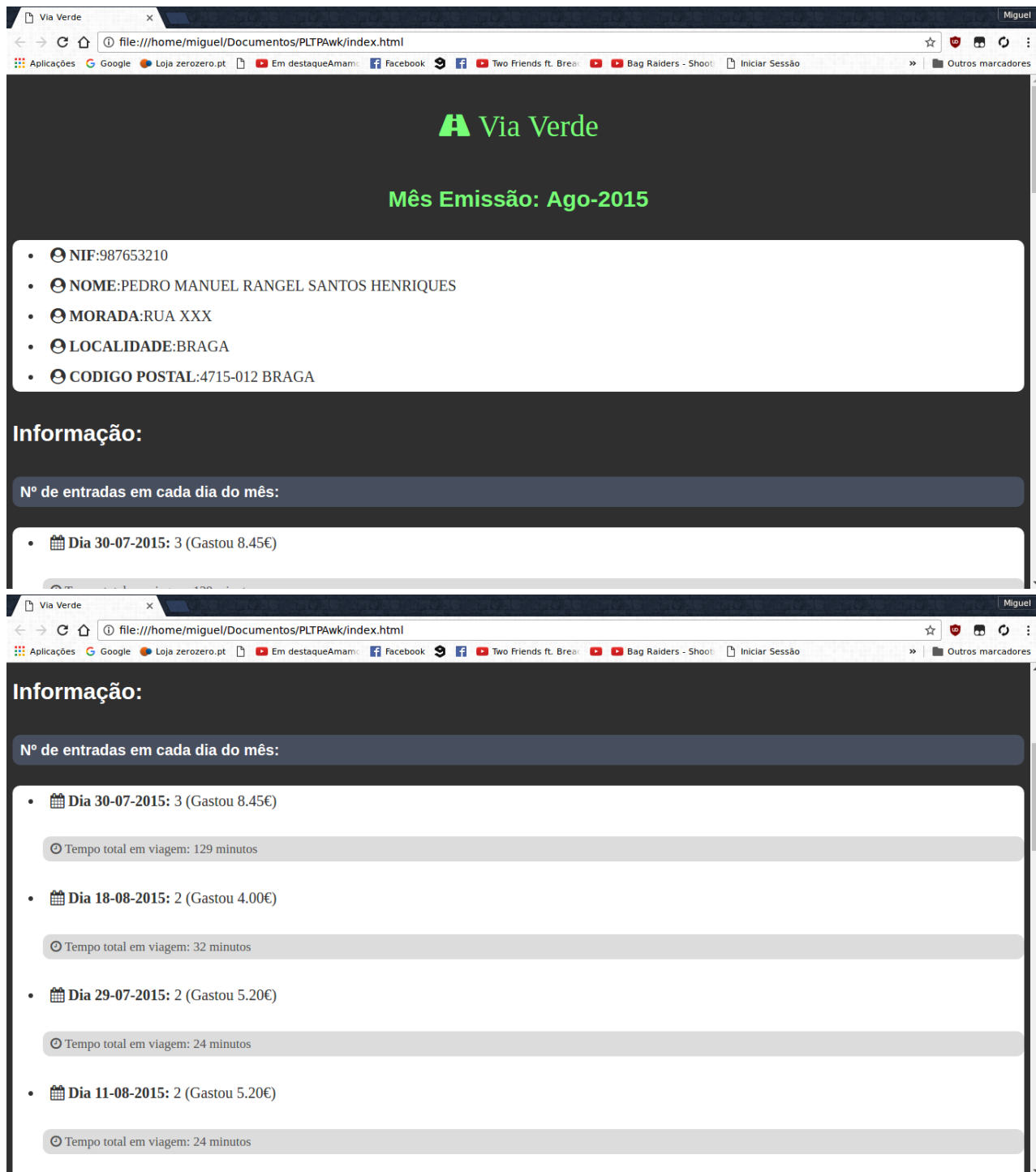
Mostra-se a seguir o teste feito (o ficheiro recebido pelo programa - viaverde.xml) e os respectivos resultados obtidos(em formato html):

4.2.1 O programa é executado com o ficheiro como argumento



```
[00:20]miguel PLTPAwk $ ./procViaverde viaverde.xml
```

4.2.2 Obtém-se o seguinte resultado



The screenshot shows a web browser window with the address bar displaying `file:///home/miguel/Documents/PLTPAwk/index.html`. The browser's address bar and tabs are visible at the top. The page content is as follows:

Via Verde

Mês Emissão: Ago-2015

- NIF:987653210
- NOME:PEDRO MANUEL RANGEL SANTOS HENRIQUES
- MORADA:RUA XXX
- LOCALIDADE:BRAGA
- CODIGO POSTAL:4715-012 BRAGA

Informação:

Nº de entradas em cada dia do mês:

- Dia 30-07-2015: 3 (Gastou 8.45€)

The second screenshot shows the same browser window, but the page content is different, displaying only the 'Informação:' section:

Informação:

Nº de entradas em cada dia do mês:

- Dia 30-07-2015: 3 (Gastou 8.45€)
- Dia 18-08-2015: 2 (Gastou 4.00€)
- Dia 29-07-2015: 2 (Gastou 5.20€)
- Dia 11-08-2015: 2 (Gastou 5.20€)

Below each date entry, there is a bar indicating the total travel time:

- Dia 30-07-2015: 3 (Gastou 8.45€)
Tempo total em viagem: 129 minutos
- Dia 18-08-2015: 2 (Gastou 4.00€)
Tempo total em viagem: 32 minutos
- Dia 29-07-2015: 2 (Gastou 5.20€)
Tempo total em viagem: 24 minutos
- Dia 11-08-2015: 2 (Gastou 5.20€)
Tempo total em viagem: 24 minutos

Via Verde

Miguel

file:///home/miguel/Documentos/PLTPAwk/index.html

Aplicações Google Loja zerozero.pt Em destaqueAmam: Facebook Two Friends ft. Bre Bag Raiders - Shoot Iniciar Sessão Outros marcadores

Locais de saída:

- [Braga Sul\(2\)](#)
- [Maia PV\(1\)](#)
- [Neiva S-N\(1\)](#)
- [Angeiras N-S\(5\)](#)
- [Freixieiro\(2\)](#)
- [Valongo\(1\)](#)
- [PO A Sa Cam.I\(1\)](#)
- [Ermesinde PV\(1\)](#)
- [Aeroporto\(1\)](#)
- [Maia II\(1\)](#)
- [EN107\(1\)](#)
- [EN 205 PV\(6\)](#)
- [Neiva N-S\(1\)](#)
- [Lipor\(1\)](#)
- [Ponte Pedra\(3\)](#)

Via Verde

Miguel

file:///home/miguel/Documentos/PLTPAwk/index.html

Aplicações Google Loja zerozero.pt Em destaqueAmam: Facebook Two Friends ft. Bre Bag Raiders - Shoot Iniciar Sessão Outros marcadores

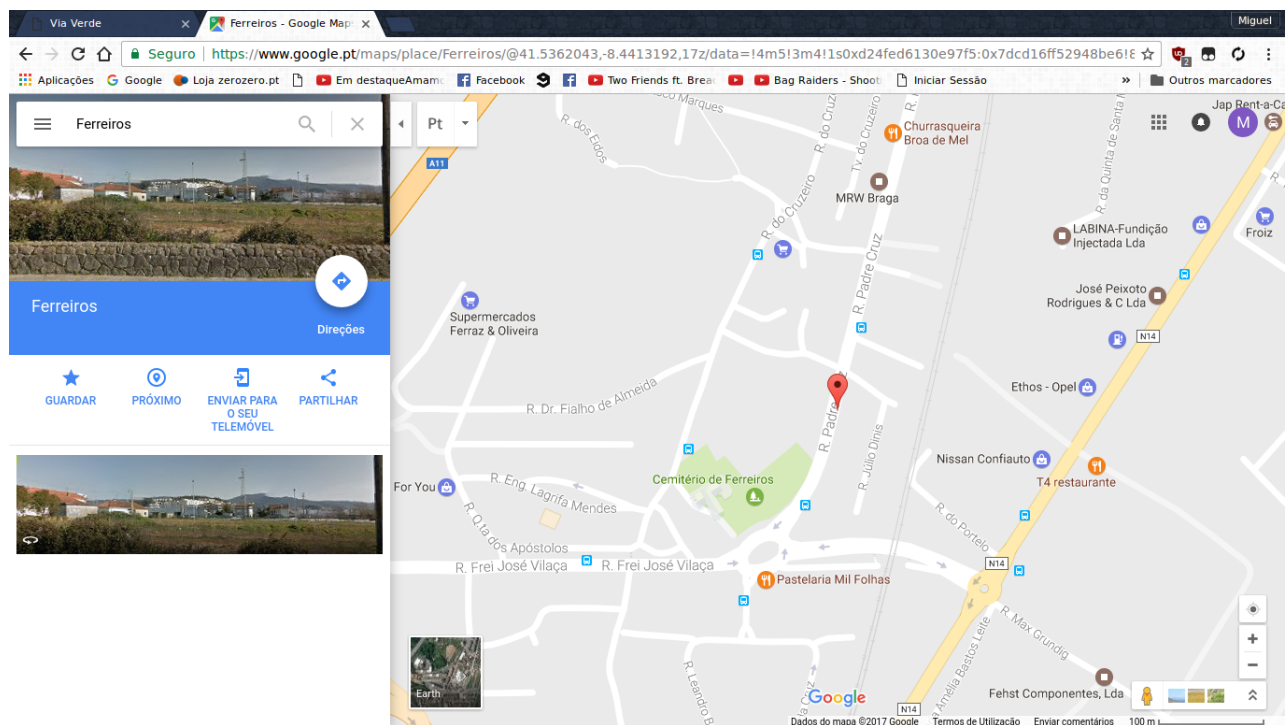
- [Maia II\(1\)](#)
- [EN107\(1\)](#)
- [EN 205 PV\(6\)](#)
- [Neiva N-S\(1\)](#)
- [Lipor\(1\)](#)
- [Ponte Pedra\(3\)](#)
- [PO Av. Central\(1\)](#)
- [Custoias\(1\)](#)
- [Povoa S-N\(5\)](#)
- [Ferreiros\(6\)](#)

Total gasto no mês:

Total: 77.4€

- Total gasto apenas em Portagens: 71.05€
- Total gasto apenas em Parques de estacionamento: 6.35€

4.2.3 Exemplo da tab que abre quando se clica no local Ferreiros



Capítulo 5

Conclusão

Neste projeto foi utilizado um ficheiro xml com informação correspondente ao extrato mensal emitido pela Via Verde para um dos seus utentes, para o qual seria necessário calcular o número de entradas em cada dia do mês, a lista dos locais de saída relativos a cada entrada, o total gasto no fim do mês e o total gasto em parques de estacionamento.

Foi necessário guardar a informação relativa ao ficheiro, neste caso em arrays associativos, e recolher e guardar tal informação a partir de várias condições, relativas a expressões regulares específicas. Para além disto é apresentado o algoritmo relativo à apresentação da informação guardada, num formato html, recorrendo ao uso de ciclos, relativos a cada array associativo, para retirar a informação destes. Para além disso foi necessário a escolha de um field separator lógico e mais oportuno, que neste caso seria `[|]`, permitindo só a obtenção da informação necessária no ficheiro xml, na posição 3.

Como se pode confirmar nos anteriores tópicos, foi decidido o uso do html como o ficheiro de apresentação, apesar de ser mais complexo do que usar o standard output, assim como a font-awesome para juntar icons relativos a cada linha para melhor compreensão e para apelar ao utilizador.

Através do uso da linguagem de programação AWK, do uso de ferramentas html para criação de uma plataforma de apresentação de informação, de um modo conciso e apresentável, foi possível realizar com sucesso os requisitos pedidos neste trabalho prático assim como demonstrar outras funcionalidades importantes e úteis para a recolha e apresentação de informação neste trabalho, assim como para o uso noutros casos semelhantes.

Apêndice A

Código do Programa

Lista-se a seguir o código do programa que foi desenvolvido.

procViaverde

```
1  #!/usr/bin/gawk -f
2
3  BEGIN {
4      FS = "[<>]";
5      file = "index.html";
6      icon = "<i class='fa fa-calendar'>"
7      fmtEntrad = "<li>icon" <b>Dia %s:</b> %d (Gastou %.2f€)</i></li>\n";
8      fmtTotais = "<li>Total gasto apenas em %s: %.2f€</i></li>\n";
9      fmtli = "<li> %s</li>\n";
10     fmth5 = "<h5 class='bg-3'>%s</h5>\n";
11     fmth4 = "<h4 class='bg-2'>%s</h4>\n";
12     fmth2 = "<h2>%s</h2></div>\n";
13     icon = "<i class='fa fa-road'>";
14     ini = "<!DOCTYPE html> <html> <head>\n";
15     tit = "<title> Via Verde </title>";
16     enc = "<meta charset='utf-8'/> </head> <body>";
17     end = "</body> </html>";
18     beg = "<div class='bg-4 title'><h1>icon" Via Verde </i></h1>";
19     css = "<link rel='stylesheet' href='css/styles.css'>";
20     fta = "<link rel='stylesheet' href='css/fa/css/font-awesome.css'>";
21     ref = "<a target='_blank' href='%s'>%s</a>";
22     maps = "https://www.google.pt/maps/place/";
23     infoCliente = 0;
24 }
25
26 NR == 1 {
27     print ini > file;
28     print tit > file;
29     print css > file;
30     print fta > file;
31     print enc > file;
32     print beg > file;
33 }
34
35 $2~/MES_EMISSAO/{
36     s = "Mês Emissão: " $3;
37     printf(fmth2, s) > file;
38 }
39
```

```

40 $2~/NIF/{
41     infoCliente++;
42     print "<ul>" > file;
43 }
44
45 infoCliente {
46     icon = "<i class='fa fa-user-circle'> ";
47     str = icon"<b>"$2"</b>:" $3"</i>";
48     gsub("_"," ", str);
49     printf(fmtli, str) > file;
50
51     if($2~/CODIGO_POSTAL/) {
52         print "</ul>" > file;
53         infoCliente--;
54     }
55 }
56
57 $2~/DATA_ENTRADA/ && $3~/[0-9-]+/ {
58     dias[$3]++;
59     ultDia = $3;
60 }
61
62 $4~/\//SAIDA/ {
63     locaisSaida[$3]++;
64 }
65
66 $2~/HORA_ENTRADA/ {
67     horaEntrada = $3;
68 }
69
70 $2~/HORA_SAIDA/ {
71     horaSaida = $3;
72     i = duracao(horaEntrada, horaSaida);
73     duracoes[ultDia] += i;
74 }
75
76 $2~/IMPORTANCIA/ {
77     gsub(",",".", $3);
78     totalGasto += $3;
79     ultimoGasto = $3;
80     totDias[ultDia] += $3;
81 }
82
83 $2~/VALOR_DESCONTO/ {
84     gsub(",",".", $3);
85     totalGasto -= $3;
86     ultimoGasto -= $3;
87     totDias[ultDia] -= $3;
88 }
89
90 $2~/TIPO/ {
91     totalLocais[$3] += ultimoGasto;
92 }
93
94 END {
95     str = "<h2 class='bg-4'>Informação:</h2>";
96     print str > file;
97
98     str = "N° de entradas em cada dia do mês:";

```

```

99     printf(fmth4, str) > file;
100     print "<ul>" > file;
101     for (i in dias) {
102         printf(fmtEntrad, i, dias[i], totDias[i]) > file;
103         icon = "<i class='fa fa-clock-o'> ";
104         aux = "Tempo total em viagem: ";
105         str = icon aux duracoes[i] " minutos</i>";
106         printf(fmth5, str) > file;
107     }
108     print "</ul>" > file;
109
110
111
112     printf(fmth4, "Locais de saída:") > file;
113     print "<ul>" > file;
114     for (i in locaisSaida) {
115         aux = sprintf(ref, maps i, i);
116         icon = "<i class='fa fa-map-marker'> ";
117         str = icon aux " </i>" "("locaisSaida[i]")";
118         printf(fmtli, str) > file;
119     }
120     print "</ul>" > file;
121
122
123     str = "Total gasto no mês:";
124     printf(fmth4, str) > file;
125     str = "Total:" " " totalGasto"€";
126     print "<div class='aux'>" > file;
127     printf(fmth2, str) > file;
128     print "<ul>" > file;
129     for(i in totalLocais)
130         printf(fmtTotais, i, totalLocais[i]) > file;
131     print "</ul>" > file;
132
133     print end > file;
134 }
135
136 #Args: he - Hora entrada
137 #      hs - Hora saída
138 #Return duracao em minutos
139 function duracao(he, hs) {
140     result = 0;
141     split(he, ae, /[:]/);
142     split(hs, as, /[:]/);
143     if(ae[1]=="" || as[1]=="")
144         return 0;
145     if(as[1]>=ae[1])
146         aux = (as[1]-ae[1])*60;
147     else
148         aux = (as[1] + (24 - ae[1]))*60;
149     result += aux;
150     result -= ae[2];
151     result += as[2];
152     return result;
153 }

```