Processamento de Linguagens (3º ano de MIEI)

Trabalho Prático 1

Relatório de Desenvolvimento

Maria João Moreira
(a89540)

Miguel Jacinto Carvalho
(84518)

5 de abril de 2021

Resumo

O presente relatório é um suplemento ao projeto proposto pelos docentes da unidade curricular de Processamento de Linguagens. Durante os próximos tópicos, iremos explicitar as decisões tomadas e procedimentos efetuados para completar o primeiro trabalho prático.
O principal objetivo deste trabalho é colocar em prática os conhecimentos adquiridos nas aulas em relação às expressões regulares e, juntamente com as ferramentas disponíveis de uma linguagem de programação nunca antes utilizada por nós, o Python, construir um pequeno projeto baseado nestes conhecimentos.

Conteúdo

Lista de Figuras

Listings

Capítulo 1

Introdução

Este projeto foi realizado para responder ao problema do enunciado um. Neste exercício, pretende-se trabalhar com um ficheiro desportivo criado por um Organizador de Provas de Orientação. Provas essas, que são realizadas em diferentes locais e para diferentes graus de dificuldade, sendo que todas elas são adaptadas a diferentes classes de participantes.
Ao longo dos próximos tópicos, iremos explicitar detalhamente a estratégia utilizada pelo nosso grupo para solucionar cada uma das queries propostas.
No final da leitura deste relatório, o leitor será capaz de decifrar o raciocínio implícito na resposta a cada um dos seguintes problemas: Imprimir o nome de todos os concorrentes que se inscrevem como "Índividuais"e são de "Valongo", imprimir o nome completo, telemóvel e a prova em que cada está inscrito cada concorrente cujo nome seja "Paulo"ou "Ricardo"e que usem o "GMail", imprimir toda a informação dos atletas da equipa: "TURBULENTOS", imprimir a lista dos escalões por ordem alfabética e para cada um, indicar o respetivo número de atletas e gerar uma página HTML com a lista das equipas inscritas em qualquer prova.
Note-se que durante toda a realização do projeto tentamos sempre utilizar a forma mais simples e eficiente de resolver cada um dos problemas, utilizando o máximo de funções definidas no módulo RE e recorrendo muitas vezes a "Dicionários"(Hashmap) para guardar a informação processada, visto que é uma das melhores estruturas em termos de eficiência.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Foi atribuído ao nosso grupo o enunciado 1, cujo objetivo é trabalhar com um ficheiro desportivo criado por um Organizador de Provas de Orientação. Este ficheiro propunha que extraíssemos informações de atletas de um ficheiro "json", aliando o nosso conhecimento sobre expressões regulares à linguagem Python.

2.2 Especificação de Requisitos

2.2.1 Dados

Identificamos três requisitos fundamentais relativos aos dados do problema: Manter os dados atualizados nas estruturas de dados, processar corretamente os dados do ficheiro "json", mesmo que este varie de tamanho e manter os dados compactos no sentido de guardar apenas a informação relevante.

2.2.2 Pedidos

Neste paramêtro podemos salientar dois requisitos fundamentais para o funcionamento e utilidade do sistema: Responder às cinco queries propostas no enunciado, e garantir que a interação com o utilizador é estabelecida sem problemas.

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

Decidimos que preferíamos armazenar todos os dados do ficheiro json nas estruturas de dados antes da manipulação dos mesmos, ao invés de manipular os dados à medida que líamos o ficheiro linha a linha.
Para este efeito decidimos tirar proveito das funcionalidades dos dicionários para armazenar toda a informação retirada do ficheiro de inscritos.
Como o ficheiro tem campos que se repetem, e a informação de cada inscrito encontra-se contígua, criamos sete dicionários, um para cada campo lido. Para cada dicionário a key é o seu índice e o value é a informação lida e processada pela expressão regular correspondente.

```
1 nome = {}
2 dataNasc = {}
3 morada = {}
4 email = {}
5 prova  = {}
6 escalao = {}
7 equipa = {}
```

Este efeito também podia ser conseguido com outras estruturas de dados, tais como arrays, porém optamos pelos dicionários pelas inúmeras vantagens que apresentam na resolução deste problema.

3.2 Algoritmos

3.2.1 Preencher os dicionarios

Uma das funções mais importantes do projeto é a preenche_dic(). Esta função é responsável por fazer o parse do ficheiro e colocar a informação nos dicionários.
É sabido que cada registo possui sete campos. Nome, data de nascimento, morada, email, prova, escalão e equipa. Cada um destes campos é guardado num dicionário próprio cuja key é o índice no ficheiro e o value é o próprio conteúdo. Para filtrar a informação e consequentemente a guardar, utilizamos duas funções pré-definidas do módulo re: A função search e a função split. A função search capta o campo desejado através de uma expressão regular genérica e o split permite selecionar o conteúdo que realmente interessa.
Vamos dar o exemplo do nome. Após a execução da função search, temos na variável utilizada como resultado o seguinte conteúdo: "Nome:"XXXX', sendo XXXX um valor genérico. No entanto, é necessário obter apenas o XXXX, dispensando a string "Nome:". Por isso, depois da execução do split iremos obter um array com dois índices. No primeiro, está guardada a string "Nome:"e no segundo a string XXXX. Posto isto, podemos guardar o conteúdo no segundo índice (Na prática é o índice 1) no respetivo dicionário. Este processo repete-se para todos os campos do registo e consequentemente para todos os registos do ficheiro.

Listing 3.1: Código para preencher dicionário

```
def preenche_dic():
        file = open ("inscritos-form.json", "r")
        i=0
        count = 0
        for linha in file:
                if(count == 7):
                        count = 0
                        i += 1
                else:
                        # Extra  o do Nome Atrav s de uma Express o Regular.
                        m = re.search(r'"nome":"([^\,\"]+)', linha)
                        v = re.search(r'"nome":""', linha)
                        if (m):
                                new_text = re.split (r'"nome":"', m.group())
                                nome[i] = new_text[1]
                                count+=1
                        if (v):
                                nome[i] = "N o registado"
                                count+=1
                        # Extra  o da Data de Nascimento Atrav s de uma Express o
                            Regular.
                        m = re.search(r'"dataNasc":"([^\,\"]+)', linha)
                        v = re.search(r'"dataNasc":""', linha)
                        if (m):
                                new_text = re.split (r'"dataNasc":"', m.group())
                                dataNasc[i] = new_text[1]
                                count+=1
                        if (v):
                                dataNasc[i] = "N o registado"
                                count+=1
                        # Extra  o da Morada Atrav s de uma Express o Regular.
                        m = re.search(r'"morada":"([^\"]+)', linha)
```

```
32                      v = re.search(r'"morada":"" ', linha)
33                      if (m):
34                              new_text = re.split (r'"morada":" ', m.group())
35                              morada[i] = new_text[1]
36                              count+=1
37                      if (v):
38                              morada[i] = "N o registado"
39                              count+=1
40                      # Extra  o do Email Atrav s de uma Express o Regular.
41                      m = re.search(r'"email":"([^\,\"]+)', linha)
42                      v = re.search(r'"email":"" ', linha)
43                      if (m):
44                              new_text = re.split (r'"email":" ', m.group())
45                              email[i] = new_text[1]
46                              count+=1
47                      if (v):
48                              email[i] = "N o registado"
49                              count+=1
50
51                      # Extra  o da Prova em Que Participou Atrav s de uma
                            Express o Regular.
52                      m = re.search(r'"prova":"([^\,\"]+)', linha)
53                      v = re.search(r'"prova":"" ', linha)
54                      if (m):
55                              new_text = re.split (r'"prova":" ', m.group())
56                              prova[i]= new_text[1]
57                              count+=1
58                      if (v):
59                              prova[i] = "N o registado"
60                              count+=1
61
62                      # Extra  o do Escal o Atrav s de uma Express o Regular.
63                      m = re.search(r'"escalao":"([^\,\"]+)', linha)
64                      v = re.search(r'"escalao":"" ', linha)
65                      if (m):
66                              new_text = re.split (r'"escalao":" ', m.group())
67                              escalao[i] = new_text[1]
68                              count+=1
69                      if (v):
70                              escalao[i]="N o registado"
71                              count+=1
72
73                      # Extra  o da Equipa Atrav s de uma Express o Regular.
74                      m = re.search(r'"equipa":"([^\"]+)', linha)
75                      v = re.search(r'"equipa":"" ', linha)
76                      if (m):
77                              new_text = re.split (r'"equipa":" ', m.group())
78                              equipa[i] = new_text[1]
79                              count+=1
80                      if (v):
81                              equipa[i] = "N o registado"
82                              count+=1
83
84      file.close()
```

3.2.2 Alinea a

Para a resolução da alinea a, criamos um array que vai conter as posições dos inscritos provenientes de Valongo, estes são captados por uma expressão regular à medida que percorremos o dicionário de moradas. Com os indices dos inscritos de Valongo, agora queremos filtrar aqueles que se tenham inscrito como Individuais.
Como no ficheiro há várias situações em que a mesma palavra aparece escrita com caracteres, ora maiúsculos ora minusculos, utilizamos expressões regulares que interpretam ambos os casos da mesma forma. Também notamos que haviam erros ortográficos, como por exemplo "indivudual", e neste caso interpretamos como um erro e não como algo intencional por parte do inscrito, e por isso captamos da mesma forma que "individual".
Percorremos o array de indices e verificamos se a equipa é captada pela expressão regular, se isto se verificar o nome do inscrito em maiusculas é colocado no dicionário criado localmente que será devolvido no final de execução deste algoritmo.

Listing 3.2: Código alinea a

```
1  def alinea_a():
2          index = []
3          for k,v in morada.items():
4                  if(re.search(r'(?i:valongo)',v)):
5                          index.append(k)
6
7          nomes = {}
8
9          for k in index:
10                 if(re.search(r'(?i:indiv[i|u]dual)',equipa[k])):
11                         nomes[k] = (nome[k].upper());
12
13         return nomes
```

3.2.3 Alinea b

A alinea b tinha como requisitos imprimir algumas informações sobre atletas cujo o nome fosse "Paulo"ou "Ricardo"e cujo email tivesse como domínio o "gmail". As informações a imprimir de cada atleta que apresentasse as carateristicas anteriores, são o seu nome completo, o seu email e prova em que está inscrito.
Começamos por procurar no dicionário global nome, com auxilio de uma expressão regular, os atletas cujo nome fosse "Paulo"ou "Ricardo", e guardamos as keys dos atletas que tenham sucesso na correspondência, numa lista de indices.
Para cada indice presente na lista local acabada de criar, verificamos se o atleta com a key igual ao indice, utiliza o gmail. Para isto utilizamos também uma expressão regular. Se a verificação se der, então colocamos uma lista o nome, email e prova, provenientes dos respetivos dicionários cuja key é o indice previamente verificado.
Colocamos essa lista num dicionário, que será devolvido no fim da execução da função, cuja key será o indice e o value será a lista com as informações a inserir de cada atleta que possua as carateristicas requisitadas.

Listing 3.3: Código alinea b

```
1  def alinea_b():
2          array = []
3          dic = {}
4
5          index = []
6          for k,v in nome.items():
7                  if(re.search(r'(?i:paulo)',v) or re.search(r'(?i:ricardo)',v) ):
8                          index.append(k)
9
10         for k in index:
11                 array = []
12                 if(re.search(r'(?i:gmail)',email[k])):
13                         array.append(nome[k])
14                         array.append(prova[k])
15                         array.append(email[k])
16                         dic[k] = array
17
18         return dic
```

3.2.4 Alinea c

A função alinead() é a nossa solução para o terceiro problema, imprimir toda a informação dos atletas da equipa "TURBULENTOS". Esta função está dividida em duas fases. Inicialmente, é percorrido o "Dicionário"das equipas. Através do uso da função search e da expressão regular: (?i:turbulentos), são captados todos os índices de registos desta equipa e colocados no array "index". Logo a seguir, é percorrido o array index e colocada toda a informação relativa ao registo desse índice no "Dicionário"dic. A função devolve o "Dicionário"dic, que contém toda a informação relativa à equipa "TURBULENTOS".

Listing 3.4: Código alinea c

```
1  def alinea_c():
2          array = []
3          dic = {}
4
5          index = []
6          for k,v in equipa.items():
7                  if(re.search(r'(?i:turbulentos)',v)):
8                          index.append(k)
9
10         s=0
11         for k in index:
12                 array = []
13                 array.append(nome[k])
14                 array.append(dataNasc[k])
15                 array.append(morada[k])
16                 array.append(email[k])
17                 array.append(prova[k])
18                 array.append(escalao[k])
19                 array.append(equipa[k])
20                 dic[s] = array
21                 s+=1
22
23         return dic
```

3.2.5 Alinea d

A função alinead() é a resposta ao quarto problema, imprimir a lista dos escalões por ordem alfabética e para cada escalão indicar quantos atletas estão inscritos nesse escalão. A função percorre o "Dicionário"relativos aos escalões e coloca em "res"o resultado final. Note-se que após o termino desta função, o "Dicionário"res contém a lista dos escalões (key) e o número de atletas desses escalões (value).

Listing 3.5: Código alinea d

```
1  def alinea_d():
2          res = {}
3          for k,v in escalao.items():
4                  if v in res:
5                          res[v] += 1
6                  else:
7                          res[v] = 1
8
9          res = dict(sorted(res.items(), key = lambda p: p[0]))
10         return res
```

3.2.6 Alinea e

Para a alinea e, o objetivo é gerar uma página html com a quantidade de inscritos de cada em cada uma das provas. Para além disso, devia permitir aceder a páginas html com informações dos atletas inscritos em determinada prova e equipa, através de um link. As equipas deviam ainda estar ordenadas por ordem decrescente de número de inscritos. Para tal foram implementadas as funções que permitiam alcançar o proposto.
Primeiramente, devíamos conseguir ter toda a informação necessária de forma organizada, para facilitar o acesso à mesma no momento da criação do ficheiro html. A função 3.6 é responsável por criar um dicionário em que a key seja o nome da prova que estamos a filtrar o número de inscritos, e os values serão uma lista de listas. Cada lista interior terá o nome da equipa e o número de inscritos na equipa e na prova. Algo do género : d = nome_prova : [[nome_equipa, numero_inscritos], [nome_equipa, numero_inscritos]]

Listing 3.6: Função que devolve dicionário ordenado com número de inscritos em cada equipa e prova

```
1  def lista_equipas_prova():
2          d = {}
3          for k,v in equipa.items():
4                  e = v
5                  p = prova[k]
6                  c=0
7                  t=[]
8                  for k1,v1 in prova.items():
9                          if (v1 == p) and (equipa[k1] == e):
10                                 c+=1
11                 t.append(e)
12                 t.append(c)
13                 lol=[]
14                 if p not in d:
15                         lol.append(t)
16                         d[p] = lol
17                 if p in d:
18                         if t in d[p]:
19                                 pass
20                         else:
21                                 d[p].append(t)
22         d = dict(sorted(d.items(), key = lambda p: p[0]))
23         for k,v in d.items():
24                 v.sort(key= lambda p: p[1],reverse=True)
25
26         return d
```

Para criar a página html com a informação dos atletas de cada equipa inscritos numa dada prova, decidimos desenvolver uma função que dado o nome da equipa e o nome da prova vai procurar através de uma expressão regular nos dicionários equipa e prova, as keys dos atletas que estão inscritos nessa prova e equipa. As keys são colocadas numa lista de indices, que posteriormente servirá para ir buscar a todos os dicionários globais as informações apenas dos atletas cuja key tenha sido inserida na lista de indices.
As informações de cada atleta são colocadas numa lista e esta lista depois de preenchida será colocada no dicionário a ser devolvido no fim, cuja key é o indice do atleta e o value é a lista com as informações do mesmo.
O dicionário deve ter a seguinte aparencia : atletas = indice : [nome, dataNasc,morada...]

Listing 3.7: Função que devolve dicionário com informação de cada atleta

```
1  def atletas_equipa_prova(nome_equipa, nome_prova):
2          atletas = {}
3          index = []
4          info = []
5          for k,v in equipa.items():
6                  if re.search(rf"{v}", nome_equipa, re.IGNORECASE):
7                          index.append(k)
8          indices = []
9          for elem in index:
10                         if re.search(rf"{prova[elem]}", nome_prova, re.IGNORECASE):
11                                 indices.append(elem)
12         for e in indices:
13                 info.append(nome[e])
14                 info.append(str(dataNasc[e]))
15                 info.append(morada[e])
16                 info.append(email[e])
17                 info.append(prova[e])
18                 info.append(escalao[e])
19                 info.append(equipa[e])
20                 atletas[e] = info
21                 info = []
22         return atletas
```

Durante o desenvolvimento desta resolução apercebemo-nos que haviam determinadas situações que necessitavam de ser tratadas de uma forma especial. Por exemplo, uma pessoa que se inscreva como "individual"tem de ser tratada como uma pessoa que não tem equipa, qualquer que seja a maneira como a palavra está inscrita. Para além disso, quando se trata de provas desportivas, a equipa é um identificador que deve ser algo único, em futebol não existem duas equipas com o mesmo nome escrito de maneira diferente, como por exemplo, "Benfica"e "benfica", ambas maneiras de escrever identificam a mesma equipa.
Por causa deste pormenor quisemos garantir que estes casos seriam tratados, e para tal desenvolvemos as funções abaixo. Para explicitar aquilo que estas funções devem fazer, usemos um exemplo mais simples.
Tomemos como dicionário de exemplo:
t = 'k1':[['f',4],['b',1],['a',3],['A',4]],'k2':[['C',2],['c',2]],'k3':[['d',1]]

Listing 3.8: Função auxiliar que filtra a cabeça das listas interiores de uma lista de listas

```
def aux0(l):
        res = []
        for elem in l:
                res.append(elem[0])
        return res
```

Listing 3.9: Função auxiliar que filtra o segundo elemento das listas interiores de uma lista de listass

```
def aux1(l):
        res = []
        for elem in l:
                res.append(elem[1])
        return res
```

Listing 3.10: Função que junta a contagem palavras iguais

```
def rep_count(d):
        f = {}
        for k,v in d.items():
                a = []
                b = []
                a0 = aux0(v)
                a1 = aux1(v)
                final = []
                for i in range(0,len(a0)):
                        for j in range(i,len(a0)):
                                if (re.search(rf"(?i:{a0[i]})",a0[j], re.IGNORECASE))
                                    and (i<j):
                                        a = []
                                        a.append(a0[i].lower())
                                        a.append(a1[i]+a1[j])
                        if a and a not in final:
                                final.append(a)
                f[k]=final
        return f
```

f = rep_count(t)
Resultado: 'k1': [['a', 7]], 'k2': [['c', 4]], 'k3': []

Listing 3.11: Função que junta a contagem palavras iguais

```
def rem_reps(d,f):
        res = {}
        for k,v in d.items():
                final = []
                low = aux0(f[k])
                for elem in v:
                        if elem[0].lower() in low:
                                pass
                        else:
                                final.append(elem)
                res[k]=final
        for k,v in res.items():
                for elem in f[k]:
                        v.append(elem)

        for k,v in res.items():
                v.sort(key= lambda p: p[1],reverse=True)

        return res
```

res = rem_reps(t,f)
Resultado: 'k1': [['a', 7], ['f', 4], ['b', 1]], 'k2': [['c', 4]], 'k3': [['d', 1]]

O que a função rep_count(d) faz é pegar nos termos que se repetem no dicionário d, que será o dicionário resultante da função lista_equipa_prova(), e criar um dicionário f onde associado à mesma key, o valor de inscritos desse termo seja somado com os valores de inscritos de termos iguais.
Com este dicionário, agora utilizamos a rem_reps(d,f), que irá criar o dicionário res, onde será colocado os termos do dicionário f e os termos que não se repetem do dicionário d, associados sempre ao seu valor de inscritos por equipa e prova.
Basicamente, junta os termos repetidos do dicionário resultante da função lista_equipa_prova() devolvendo um novo dicionário já ordenado decrescentemente.
Isto apenas funciona para palavras que se repitam em todos os caracteres, casos como "individual"e "indivudual", não são tratados desta forma pois não temos como saber se é um mero erro dado pelo utilizador que se inscreveu dessa forma ou se a intenção era inscrever-se numa equipa possua realmente esse nome, por incrivel que pareça.

Reparamos que haveria necessidade de criar um número considerável de ficheiros html com as informações dos atletas de cada equipa inscritos numa dada prova, por este motivo criamos um algoritmo que faria esse trabalho por nós.
Para cada equipa inscrita numa determinada prova iria ser criado um ficheiro Atleta(n).html, em que n é o número de paginas a ser criadas.
Para cada página usamos o algoritmo 3.11 anteriormente explicado, para aceder apenas à informação dos atletas da equipa e prova necessários para gerar cada ficheiro especificamente.
Escrevemos no ficheiro através da função write(), com a syntax html para que cada ficheiro seja gerado corretamente. Aproveitamos também para explorar um pouco CSS, para mudar o tipo de letra e cores de fundo dos ficheiros.
A utilidade principal desta função é auxiliar a função fileprincipal() na criação da página HTML na medida em que vai ser chamada para a criação dos "links"na página principal.

Listing 3.12: Função Que Cria um Ficheiro HTML Para Cada Atleta

```
def create_files():
        n = 0
        lista = lista_equipas_prova()
        for k,v in lista.items():
                first = k
                for e in v:
                        second = e[0]
                        string = "Atleta" + str(n) + ".html"
                        f = open (string, "w")
                        atletas = atletas_equipa_prova (second, first)
                        f.write ('<!DOCTYPE html>\n<html>\n<head>\n<meta charset="UTF
                            -8"">\n</head>')
                        f.write ('<body>\n')
                        f.write ('<h1 style ="font-family: Courier; font-style:
                            italic; background-color: pink">')
                        f.write ("Lista de Atletas da Equipa: " + second)
                        f.write ('</h1>')

                        for k1,v1 in atletas.items():
                                elem = v1
                                f.write ('<h2 style ="font-family: Courier;
                                    background-color: powderblue">')
                                f.write ("Nome: " + str(elem[0]))
                                f.write ('</h2>')
                                f.write ('<ul>')
                                f.write ('<h3 style ="font-family: Courier;
                                    background-color: yellow">')
                                f.write ("Data: " + str(elem[1]))
                                f.write ('</h3>')
                                f.write ('<h3 style ="font-family: Courier;
                                    background-color: powderblue">')
                                f.write ("Morada: " + str(elem[2]))
                                f.write ('</h3>')
                                f.write ('<h3 style ="font-family: Courier;
                                    background-color: yellow">')
                                f.write ("Email: " + str(elem[3]))
                                f.write ('</h3>')
                                f.write ('<h3 style ="font-family: Courier;
                                    background-color: powderblue">')
                                f.write ("Prova: " + str(elem[4]))
                                f.write ('</h3>')
                                f.write ('<h3 style ="font-family: Courier;
                                    background-color: yellow">')
                                f.write ("Escalao: " + str(elem[5]))
                                f.write ('</h3>')
                                f.write ('<h3 style ="font-family: Courier;
                                    background-color: powderblue">')
                                f.write ("Equipa: " + str(elem[6]))
                                f.write ('</h3>')
                                f.write ('</ul>')
                        f.write ('</body>\n</html>')
                        f.close()
                        n += 1
```

A função fileprincipal() é responsável pela geração da página HTML. Após a execução desta função, é criada uma página HTML que lista todas as provas registadas. Cada prova possui as equipas que nela participam, bem como o número de elementos dessa equipa. Além disso, cada equipa dispõe de um "link"para outra página HTML na qual estão todas as informações relativas aos atletas dessa equipa.

Listing 3.13: Função que cria página html principal

```
def file_principal():

        f = open ("out.html", "w")
        n = 0
        final = lista_equipas_prova()

        # Instru  es Relativas ao T tulo da P gina.
        f.write ('<!DOCTYPE html>\n<html>\n<head>\n<meta charset="UTF-8"">\n</head>')
        f.write ('<body>\n')
        f.write ('<h1 style = "font-family: Courier; font-style: italic; background-
            color: pink">')
        f.write ("Lista de Equipas Por Prova")
        f.write ('</h1>')

        # Nos Dois Ciclos Seguintes, S o Adicionadas  P gina HTML Todas as
            Informa  es Relaticas  s Provas, Equipas e Atletas Registados.
        for k,v in final.items():

                f.write ('<ul>')
                f.write ('<h2 style = "font-family: Courier; background-color: red
                    ">')
                f.write (k)
                f.write (r'</h2>')
                f.write (r'<ul>')

                for i in v:

                        string = "Atleta" + str(n) + ".html"
                        f.write (r'<h3 style = "font-family: Courier; background-
                            color: powderblue">')
                        f.write ("\"")
                        f.write (i[0])
                        f.write ("\"")
                        f.write (": ")
                        f.write (str(i[1]))
                        f.write (" Atleta(s) Inscritos")
                        f.write ('</h3>')
                        f.write ('<a style = "font-family: Courier; background-color:
                             yellow" href=' + string + '>Consultar Atletas da Equipa</
                            a>')
                        n += 1

                f.write ('</ul>')
                f.write ('</ul>')

        f.write ('</body>\n</html>')
```

3.2.7 Main

A função main é utilizada para criar um menu de interação com o utilizador. O utilizador é solicitado para inserir um número de 0 a 5. A opção 0 corresponde à saída do programa e as opções de 1 a 5 referem-se a cada uma das queries, respetivamente. Note-se que todas as funções auxiliares devolvem uma estrutura resultado e nunca utilizam prints, logo, a main é a função responsável por isso. Após a chamada da função preenchedic(), é imprimido o menu e solicitado ao utilizador que insira uma opção. O código implícito em cada opção escolhida é sempre estruturado da seguinte forma: Processamento da estrutura devolvida pela função auxiliar e print dos resultados. A main é executada em ciclo, até que o utilizador insira a opção 0.

Listing 3.14: Código main

```
1  def main():
2          preenche_dic()
3
4          os.system ("clear") # Comando Para Limpar o Terminal.
5
6          print ("\n" * 10)
7
8          print ()
9          print ("Trabalho Pr tico 1. Enunciado 1: Processador de Inscritos Numa
               Atividade Desportiva.")
10         print ()
11         print ("Realizado Por: Miguel Carvalho (A84518) e Maria Jo o (A89540).")
12         print ()
13         print ("O Que Pretende Fazer?")
14         print ()
15         print ("Prima 1. Imprimir o Nome (Convertido Para Ma sculas) de Todos os
               Concorrentes Que se Inscrevem Como Individuais e S o de Valongo.")
16         print ()
17         print ("Prima 2. Imprimir o Nome Completo, o EMail e a Prova em Que Est
               Inscrito Cada Concorrente Cujo Nome Seja Paulo Ou Ricardo Desde Que Usem
               GMail.")
18         print ()
19         print ("Prima 3. Imprimir Toda a Informa  o Dos Atletas da Equipa
               Turbulentos.")
20         print ()
21         print ("Prima 4. Imprimir a Lista dos Escal es Por Ordem Alfab tica e Para
               Cada um Indicar Quantos Atletas Est o Inscritos Nesse Escal o.")
22         print ()
23         print ("Prima 5. Gerar P gina HTML Com a Lista das Equipas Inscritas em
               Qualquer Prova.")
24         print ()
25         print ("Prima 0. Sair.")
26         print ()
27
28         inputFromUser = int(input("Op   o: "))
29
30         while (inputFromUser != 0):
31
32                 os.system ("clear")
33
34                 print ("\n" * 10)
35
36                 if (inputFromUser == 1):
```

```
37                  print ("Imprimir o Nome (Convertido Para Ma sculas) de Todos
                        os Concorrentes Que se Inscrevem Como Individuais e S o
                        de Valongo.")
38                  print ()
39                  a=alinea_a()
40                  for k,v in a.items():
41                          print ("Concorrente N mero", k+1 , "Inscrito Como
                                Concorrente Individual e de Valongo:", v)

42                          print ()
43
44          if (inputFromUser == 2):
45                  print ("Imprimir o Nome Completo, o EMail e a Prova em Que
                        Est  Inscrito Cada Concorrente Cujo Nome Seja Paulo Ou
                        Ricardo Desde Que Usem GMail.")
46                  print ()
47                  b = alinea_b()
48                  for k,v in b.items():
49                          print ("Registo N mero", k + 1)
50                          print ("Nome Completo:", v[0])
51                          print ("EMail:",v[1])
52                          print ("Prova Em Que Est  Inscrito:", v[2])
53                          print ()
54
55          if (inputFromUser == 3):
56                  print ("Imprimir Toda a Informa  o Dos Atletas da Equipa
                        Turbulentos.")
57                  print ()
58                  c = alinea_c()
59                  for k,v in c.items():
60                          print ("Registo N mero", k + 1)
61                          print ("Nome Completo:", v[0])
62                          print ("Data de Nascimento:", v[1])
63                          print ("Morada:", v[2])
64                          print ("EMail:",v[3])
65                          print ("Prova Em Que Est  Inscrito:", v[4])
66                          print ("Escal o:", v[5])
67                          print ("Equipa:", v[6])
68                          print ()
69
70          if (inputFromUser == 4):
71                  print ("Imprimir a Lista dos Escal es Por Ordem Alfab tica
                        e Para Cada um Indicar Quantos Atletas Est o Inscritos
                        Nesse Escal o.")
72                  print ()
73                  print ("NOTE-SE QUE O Campo N o Registado CORRESPONDE A
                        ATLETAS QUE N O POSSUEM ESCAL O!")
74                  print ()
75                  d = alinea_d()
76                  for k,v in d.items():
77                          if (k == ''):
78                                  print ("Atletas sem escal o:", str(v))
79                                  print ()
80                          else:
```

```
81                                  print ("Escalão:", k)
82                                  print ("Número de Inscritos:",str(v))
83                                  print ()
84
85              if (inputFromUser == 5):
86                      create_files()
87                      file_principal()
88                      print ("A Página HTML Foi Gerada!")
89                      print ()
90
91
92              if (inputFromUser != 0 and inputFromUser != 1 and inputFromUser != 2
                    and inputFromUser != 3 and inputFromUser != 4 and inputFromUser !=
                     5):
93                      print ("Deve Inserir um Número de 0 a 5.")
94                      print ()
95
96              print ("O Que Pretende Fazer?")
97              print ()
98              print ("Prima 1. Imprimir o Nome (Convertido Para Maísculas) de
                    Todos os Concorrentes Que se Inscrevem Como Individuais e Sío de
                    Valongo.")
99              print ()
100             print ("Prima 2. Imprimir o Nome Completo, o EMail e a Prova em Que
                    Está Inscrito Cada Concorrente Cujo Nome Seja Paulo Ou Ricardo
                    Desde Que Usem GMail.")
101             print ()
102             print ("Prima 3. Imprimir Toda a Informação Dos Atletas da Equipa
                    Turbulentos.")
103             print ()
104             print ("Prima 4. Imprimir a Lista dos Escalíes Por Ordem Alfabética
                     e Para Cada um Indicar Quantos Atletas Estío Inscritos Nesse
                    Escalío.")
105             print ()
106             print ("Prima 5. Gerar Página HTML Com a Lista das Equipas Inscritas
                     em Qualquer Prova.")
107             print ()
108             print ("Prima 0. Sair.")
109             print ()
110
111             inputFromUser = int(input("Opção: "))
```

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

4.2 Testes realizados e Resultados

4.2.1 Main

A main é encarregue de mostrar no terminal os resultados das funções implementadas para a resolução de todas as alineas. O aspecto da execução da main está representado na Figura 4.1

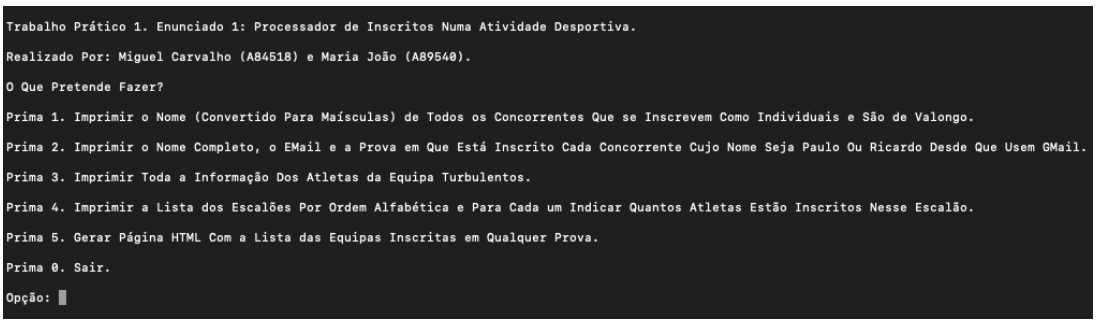

Figura 4.1: Menu

4.2.2 Preencher dicionários

Os dicionários depois de preenchidos ficam com a seguinte aparência:

```
Dicionário Nome:
0 MARIO PIRES
1 Francisco Neto Silva
2 Luis Santos Poeira
3 jose moura de sousa
4 FERNANDO JORGE MENDES CARNEIRO
5 Carlos Alberto da Silva Cardoso
6 Ricardo Salgueiro
7 Rui Gilberto Santos Correia
```

Figura 4.2: Dicionário Nome

```
Dicionário Data de Nascimento:
0 04/04/59
1 22/04/89
2 01/05/66
3 24/01/54
4 17/09/82
5 08/02/73
6 22/11/63
7 23/02/74
```

Figura 4.3: Dicionário Data de Nascimento

```
Dicionário Morada:
0 RUA CÂNDIDO DOS REIS , nº 86 , 2º ,
1 Rua da Ermida, 235
2 R. Soc. Nacional Fósforos, 160 - 1º Andar Hab. 6  4150 Porto
3 rua joaquim da silva torres, 261  4470-312 maia
4 Estrada Nacional 105 - n.º 571 - Carreira - Santo Tirso
5 Rua 25 Abril Nr.193
6 Rua de Gondarém nr 765-5 A 4150-378 Porto
7 Rua Jorge Barradas nr6
```

Figura 4.4: Dicionário Morada

```
Dicionário Email:
0 mgpires@netcabo.pt
1 netosilva.francisco@gmail.com
2 lspoeira@gmail.com
3 jomogoso@gmail.com
4 nando_1982_sts@hotmail.com
5 casc1972@hotmail.com
6 rs@selcofootwear.com
7 gilcorreia@yahoo.com
```

Figura 4.5: Dicionário Email

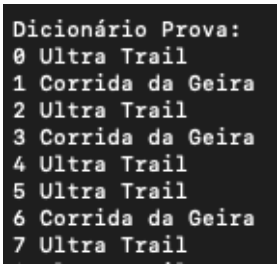

Figura 4.6: Dicionário Prova

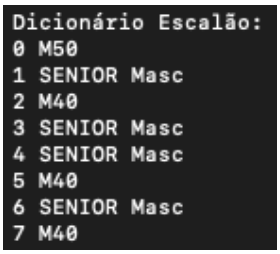

Figura 4.7: Dicionário Escalão

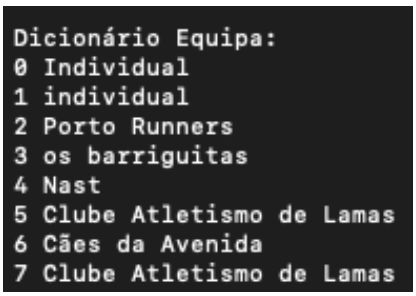

Figura 4.8: Dicionário Equipa

4.2.3 Alinea a

Ao premir 1 no menu gerado pela main, é apresentado o resultado da alinea a, apresentado na Figura 4.9

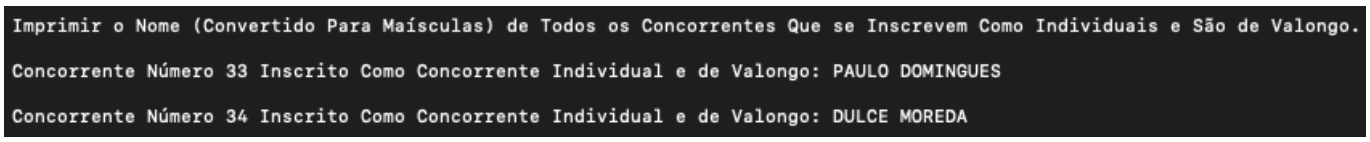

Figura 4.9: Resultado Alinea a

4.2.4 Alinea b

Ao premir 2 no menu gerado pela main, é apresentado o resultado da alinea b, apresentado na Figura 4.10

```
Imprimir o Nome Completo, o EMail e a Prova em Que Está Inscrito Cada Concorrente Cujo Nome Seja Paulo Ou Ricardo Desde Que Usem GMail.

Registo Número 1
Nome Completo: paulo de castro rocha
EMail: Ultra Trail
Prova Em Que Está Inscrito: pcastrorocha@gmail.com

Registo Número 2
Nome Completo: J Paulo Marques
EMail: Ultra Trail
Prova Em Que Está Inscrito: pmarques269@gmail.com

Registo Número 3
Nome Completo: Paulo Serra
EMail: Ultra Trail
Prova Em Que Está Inscrito: paulo.serra@gmail.com

Registo Número 4
Nome Completo: paulo Vilaça
EMail: Ultra Trail
Prova Em Que Está Inscrito: pmv777@gmail.com

Registo Número 5
Nome Completo: Paulo Domingues
EMail: Ultra Trail
Prova Em Que Está Inscrito: p.j.p.domingues@gmail.com

Registo Número 6
Nome Completo: Ricardo Jorge Dias Oliveira
EMail: Ultra Trail
Prova Em Que Está Inscrito: Ricardo.transportesabranco@gmail.com

Registo Número 7
Nome Completo: Ricardo Reis
EMail: Ultra Trail
Prova Em Que Está Inscrito: ricardoreiis@gmail.com

Registo Número 8
Nome Completo: paulo félix
EMail: Ultra Trail
Prova Em Que Está Inscrito: pauloalexteixeirafelix@gmail.com

Registo Número 9
Nome Completo: João Ricardo Tavares Neves
EMail: Ultra Trail
Prova Em Que Está Inscrito: rwichers4@gmail.com

Registo Número 10
Nome Completo: Ricardo Sousa
EMail: Corrida da Geira
Prova Em Que Está Inscrito: jose.ricardo.sousa@gmail.com

Registo Número 11
Nome Completo: Ricardo Jorge Dias Oliveira
EMail: Ultra Trail
Prova Em Que Está Inscrito: helderfva@gmail.com

Registo Número 12
Nome Completo: Ricardo Couto
EMail: Ultra Trail
Prova Em Que Está Inscrito: rjcouto@gmail.com
```

Figura 4.10: Resultado Alinea b

4.2.5 Alinea c

Ao premir 3 no menu gerado pela main, é apresentado o resultado da alinea c, apresentado na Figura 4.11

```
Imprimir Toda a Informação Dos Atletas da Equipa Turbulentos.

Registo Número 1
Nome Completo: João Costa
Data de Nascimento: 04/04/70
Morada: Rua Carolina Rosa Alves Nº27 Braga
EMail: jfscosta@gmail.com
Prova Em Que Está Inscrito: Ultra Trail
Escalão: M40
Equipa: TURBULENTOS

Registo Número 2
Nome Completo: Paulo Pimentel Torres
Data de Nascimento: 10/09/59
Morada: Rua Costa Soares, 39
EMail: geral@vieirafreitas.pt
Prova Em Que Está Inscrito: Corrida da Geira
Escalão: SENIOR Masc
Equipa: TURBULENTOS

Registo Número 3
Nome Completo: João Pimentel Torres
Data de Nascimento: 11/09/88
Morada: Rua Costa Soares, 39
EMail: geral@vieirafreitas.pt
Prova Em Que Está Inscrito: Corrida da Geira
Escalão: SENIOR Masc
Equipa: TURBULENTOS

Registo Número 4
Nome Completo: Vasco Manuel de Sequeiros Barreto Martins de Araújo
Data de Nascimento: 26/05/64
Morada: Rua S. Domingos 174 3º esq.
EMail: vascosequeiros@yahoo.com
Prova Em Que Está Inscrito: Corrida da Geira
Escalão: SENIOR Masc
Equipa: TURBULENTOS

Registo Número 5
Nome Completo: Celeste Cruz
Data de Nascimento: 23/10/65
Morada: Rua Graça Júnior, nº 14, Braga
EMail: mcrus@stec.uminho.pt
Prova Em Que Está Inscrito: Corrida da Geira
Escalão: SENIOR Fem
Equipa: TURBULENTOS

Registo Número 6
Nome Completo: Jorge Lourenço
Data de Nascimento: 27/10/69
Morada: Rua Eduardo Esperança, 7
EMail: jlourenzo@hotmail.com
Prova Em Que Está Inscrito: Corrida da Geira
Escalão: SENIOR Masc
Equipa: TURBULENTOS

Registo Número 7
Nome Completo: Luís Melo
Data de Nascimento: 11/09/67
Morada: R. Feliciano Ramos, Nº 24 - 11º Esq. Frt.
EMail: lume_pereira@yahoo.com
Prova Em Que Está Inscrito: Corrida da Geira
Escalão: SENIOR Masc
```

Figura 4.11: Resultado Alinea c

4.2.6 Alinea d

Ao premir 4 no menu gerado pela main, é apresentado o resultado da alinea d, apresentado na Figura 4.12

```
Imprimir a Lista dos Escalões Por Ordem Alfabética e Para Cada um Indicar Quantos Atletas Estão Inscritos Nesse Escalão.

NOTE-SE QUE O Campo Não Registado CORRESPONDE A ATLETAS QUE NÃO POSSUEM ESCALÃO!

Escalão: F40
Número de Inscritos: 3

Escalão: M40
Número de Inscritos: 36

Escalão: M50
Número de Inscritos: 12

Escalão: Não registado
Número de Inscritos: 12

Escalão: SENIOR Fem
Número de Inscritos: 42

Escalão: SENIOR Masc
Número de Inscritos: 175
```

Figura 4.12: Resultado Alinea d

4.2.7 Alinea e

Ao premir 5 no menu gerado pela main, é apresentado o resultado da alinea e, apresentado na Figura 4.16, uma mensagem que indica que foram criados os ficheiros html.
Aparência do dicionário gerado lista_equipas_prova(), em que key é a prova e o value é a lista de listas, cada lista interior deve ter o nome da equipa e o número de inscritos.

```
Caminhada em Caldelas - 18 maio
[[',', 5], ['EDV VIANA TRAIL', 2], ['INDIVUDUAL', 1], ['Individual', 1]]

Caminhada na Geira - parte II: Museu-Travassos
[['Print Team', 3]]

Corrida da Geira
[['TURBULENTOS', 33], ['Individual', 27], ['ARRASTASSOLAS', 8], ['Clube Náutico de Ponte de Lima', 6], ['EDV VIANA TRAIL', 4], ['NAST', 3], ['individual', 2], ['os barriguitas', 2], ['.COM', 2], ['INDIVUDUAL', 2], ['clube de veteranos do porto', 2], ['TemplarTrail', 2], ['Urban Team', 2], ['Turbulentos', 2], ['Clube Náutico Ponte de Lima', 2], ['Os Turbolentos', 2], ['Clube Spiridon de Gaia', 2], ['Pedros & Associados', 2], ['Print Team', 2], ['CÃes da Avenida', 1], ['Clube Atletismo de Lamas', 1], ['Os Barriguitas', 1], ['Born /Be Wild', 1], ['na', 1], ['clube de veteranos do Porto', 1], ['O Monge do Running', 1], ['Bracara Runners', 1], ['Companhia do Bazófias', 1], ['lousada runners', 1], ['JC&NIKI', 1], ['MRR', 1], ['C. D. R. C. AMARENSE', 1], ['Inov8/Sportlast', 1], ['Amigos dos Trilhos de Esmeriz', 1], ['Clube Nautico de Ponte de Lima', 1], ['cabritos da cortiça - Trail', 1], ['Os Marrafonas', 1]]

Ultra Trail
[['Individual', 30], ['MULTIPOWER | GAIA TRAIL', 7], ['Porto Runners', 6], ['NAST', 6], ['Tugas na Estrada', 5], ['Companhia do Bazófias', 5], ['Clube Atletismo de Lamas', 4], ['TURBULENTOS', 4], ['EDV VIANA TRAIL', 4], ['clube de veteranos do porto', 3], ['EDV-Viana Trail', 3], ['SOPRO RUN LIFE', 3], ['Ecobike Porto', 3], ['Nast', 2], ['Os Caga Tacos Running Tean', 2], ['s/ clube', 2], ['União FCI de Tomar', 2], ['Os Turbulentos', 2], ['CAVE', 2], ['Bracara Runners', 2], ['individual', 2], ['Clube Spiridon de Gaia', 2], ['Minho Aventura', 2], ['Print Team', 2], ['Clube Portugal Telecom Zona Norte', 2], ['Bando dos Trilhos e Tralhos, Associação Betetista', 2], ['Os Caga Tacos Running Team', 1], ['Blue O', 1], ['Olhares Sublimes', 1], ['Team DUAS FACES', 1], ['INDIVUDUAL', 1], ['DURO VL', 1], ['CTAD tiagoaragao.com', 1], ['XICO RUNNERS', 1], ['JC&NIKI', 1], ["cães d'avenida", 1], ['Superfraquinhos', 1], ['Ind.', 1], ['PRT/AD QUINTA', 1], ['PRT/AD Quinta', 1], ['Trail Running Esposende', 1], ['Desnível Positivo', 1], ['ParJovem Paredes', 1], ['GBES', 1], ['Falcões Selvagens', 1], ['Duros de Roer', 1], ['turbulentos', 1], ['TrailEmpenados', 1], ['n/d', 1], ['SEISSA/ACRRORIZ/MATIAS&ARAUJO FRULACT', 1], ['GD Airão', 1], ['nast', 1], ['Noddy Bike Team', 1], ['NME', 1], ['Asociacion deportiva C.A.R Redondela', 1], ['NAZAbóboda', 1], ['I-RUN', 1], ['PenaQEmpena Runners', 1], ['cabritos da cortiça - Trail', 1], ['Superfraquinhos / Minho Aventura', 1]]
```

Figura 4.13: Resultado da função lista_equipas_prova()

Aparência do dicionário gerado pela função atletas_equipa_prova("Porto Runners","Ultra Trail"). A key é o indice e o value é a lista com as informações do individuo inscrito na equipa "Porto Runners"e na prova "Ultra Trail".

```
2 ['Luis Santos Poeira', '01/05/66', 'R. Soc. Nacional Fósforos, 160 - 1º Andar Hab. 6  4150 Porto', 'lspoeira@gmail.com', 'Ultra Trail', 'M40', 'Porto Runners']

21 ['José Fernanades Ferreira', '09/09/65', 'Rua Avelino Soares Carneiro, 111 - R/C - Esº', 'j.manuelff@gmail.com', 'Ultra Trail', 'M40', 'Porto Runners']

93 ['Jorge Miguel Marques', '02/12/69', 'Rua do Taralhão, 836 Fanzeres', 'mmarques@energest.pt', 'Ultra Trail', 'M40', 'Porto Runners']

103 ['luis francisco matos marvao', '07/05/70', 'rua professor abel salazar 18 2esq', 'luismatosmarvao@gmail.com', 'Ultra Trail', 'M40', 'Porto Runners']

147 ['Nuno Cáceres', '31/08/68', 'rua eugénio de castro, 352, 2, sala 25', 'nuno.caceres@mail.telepac.pt', 'Ultra Trail', 'SENIOR Masc', 'Porto Runners']

154 ['Pedro Viana', '20/12/68', 'Rua de Tanger n1307 bloco 4 ,2dto Porto', 'Pedrocoutoviana@netcabo.Pt', 'Ultra Trail', 'M40', 'Porto Runners']
```

Figura 4.14: Resultado da função atletas_equipa_prova("Porto Runners","Ultra Trail")

Aparência da página inicial gerada pela função file_principal() que gera o ficheiro html principal.

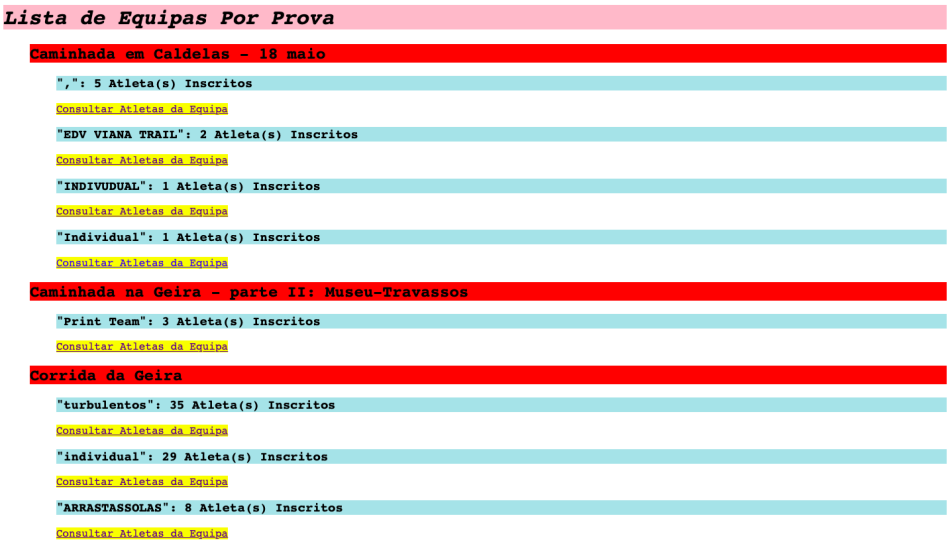

Figura 4.15: Resultado página html principal

Aparencia da página dos atletas da equipa "Porto Runners"inscritos na prova "Ultra Trail"gerada pelo ficheiro html correspondente

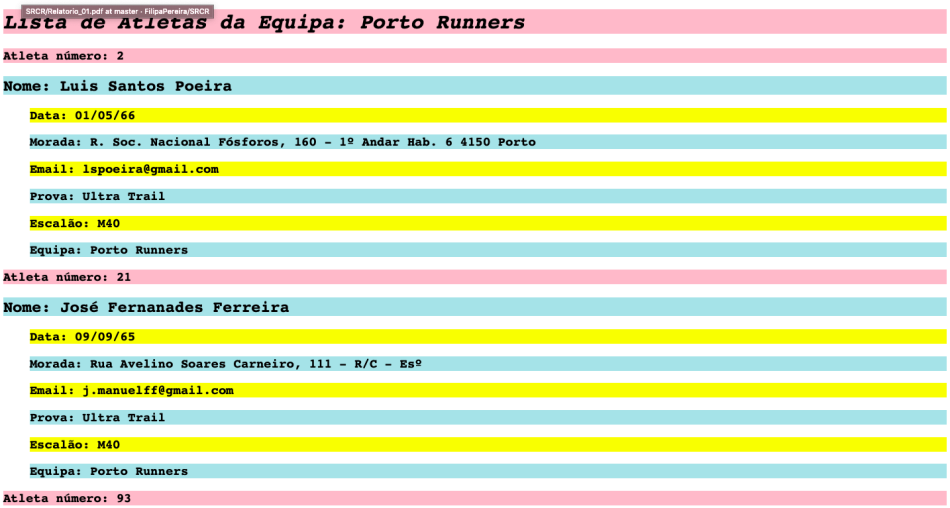

Figura 4.16: Resultado página html dos atletas da equipa Porto Runners inscritos na prova Ultra Trail

Capítulo 5

Conclusão

Após a leitura do documento, o leitor será capaz de: Relacionar as estruturas de dados com o processamento da informação do ficheiro ".json", captar a estratégia utilizada para solucionar cada um dos problemas propostos no enunciado, perceber o papel de cada uma das funções implementadas durante o projeto e como é que cada uma delas interage com as outras por um propósito maior e de entender os mecanismos utilizados para interagir com o utilizador de forma a criar "views"relativas a cada uma queries propostas.
Na nossa visão, o sistema responde com eficácia e clareza aos problemas propostos. No entanto, apesar da página HTML gerada para a alínea e) funcionar e apresentar os resultados corretos, poderíamos ter investido mais no "frontline".
Também é importante salientar que com a realização deste trabalho foi possível, num curto espaço de tempo: Aumentar a capacidade de escrever Expressões Regulares para descrição de padrões de frases dentro de textos, desenvolver, a partir de Expressões Regulares, Processadores de Linguagens Regulares, ou Filtros de Texto e familiarizar-mo-nos com as funções do módulo "re".