

**Processamento de Linguagens**  
MIEI (3º ano de Curso)  
**Trabalho Prático Nº1**  
**(ER + Filtros de Texto)**  
Grupo nº70

Ana César  
(a86038)

Margarida Faria  
(a71924)

5 de abril de 2021

## Resumo

Este relatório irá descrever os procedimentos efetuados na primeira proposta de trabalho prático, **ER + Filtros de Texto**, no âmbito da Unidade Curricular de Processamento de Linguagens.

Os objetivos concretos do trabalho prático a salientar estão expostos nos pontos descritos de seguida: melhorar a competência na escrita de *Expressões Regulares (ER)* para definir *padrões de frases*, desenvolver *Filtros de Texto* a partir das *ER*, com a finalidade de filtrar ou transformar textos, e utilizar a linguagem de programação de alto nível **Python** bem como o módulo **'er'** para sua implementação.

Através do algoritmo de escolha apresentado, o enunciado escolhido com base no número do nosso grupo (70) foi o problema **1 - Processador de Inscritos numa atividade Desportiva**.

TODO : RESULTADOS ATINGIDOS

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>4</b>
1.1	Enquadramento . . . . .	4
1.2	Contexto . . . . .	4
1.3	Problema . . . . .	4
1.4	Objetivo . . . . .	5
1.5	Resultados . . . . .	5
1.6	Estrutura do Relatório . . . . .	5
<b>2</b>	<b>Análise e Especificação</b>	<b>6</b>
2.1	Descrição informal do problema . . . . .	6
2.2	Especificação dos Requisitos . . . . .	6
2.2.1	Dados . . . . .	7
<b>3</b>	<b>Concepção/desenho da Resolução</b>	<b>8</b>
3.1	Expressões Regulares . . . . .	8
3.2	Estruturas de Dados . . . . .	9
3.3	Algoritmos . . . . .	9
3.3.1	Requisito 1 . . . . .	9
3.3.2	Requisito 2 . . . . .	10
3.3.3	Requisito 3 . . . . .	10
3.3.4	Requisito 4 . . . . .	10
3.3.5	Requisito 5 . . . . .	10
3.4	Funcionamento do programa . . . . .	11
<b>4</b>	<b>Codificação e Testes</b>	<b>12</b>
4.1	Alternativas, Decisões e Problemas de Implementação . . . . .	12
4.2	Abordagens na exibição e respectivos resultados . . . . .	13
4.2.1	Requisito 1 . . . . .	13
4.2.2	Requisito 2 . . . . .	13
4.2.3	Requisito 3 . . . . .	14
4.2.4	Requisito 4 . . . . .	15
4.2.5	Requisito 5 . . . . .	15

<b>5</b>	<b>Conclusão</b>	<b>19</b>
<b>A</b>	<b>Apresentação dos Resultados</b>	<b>20</b>

# Lista de Figuras

- 2.1 Excerto do ficheiro *inscritos-form.json* . . . . . 7
- 4.1 Resposta ao Requisito 1 . . . . . 13
- 4.2 Resposta ao Requisito 2 . . . . . 14
- 4.3 Excerto dos atletas da equipa 'Turbulentos' . . . . . 14
- 4.4 Resposta ao Requisito 4 . . . . . 15
- 4.5 Excerto da página HTML concebida com a informação das equipas . . . . . 17
- 4.6 Excerto da página HTML concebida com a informação dos atletas . . . . . 18
- A.1 User Interface . . . . . 20

# Capítulo 1

## Introdução

O presente documento pretende apresentar o primeiro trabalho prático da Unidade Curricular de Processamento de Linguagens.

Foi contemplado para este grupo o enunciado 1, referente a um **Processador de Inscritos numa atividade Desportiva**, e como tal foi disponibilizado pelos docentes um ficheiro de texto em formato *JSON*, sobre o qual escrevemos *Expressões Regulares* para descrição de *padrões de frases* dentro de textos e a partir das *ER* desenvolvemos *Processadores de Linguagens Regulares* ou *Filtros de Texto (FT)*, com o objetivo de filtrar ou transformar o texto presente no documento *inscritos-form.json* fornecido.

A linguagem utilizada para proceder à implementação foi **Python**, e um dos módulos utilizados foi o módulo **'re'**, que possui funções para manipular *Expressões Regulares*.

### 1.1 Enquadramento

Com o decorrer dos anos e o impacto da tecnologia nas nossas vidas, a quantidade de informação existente e a sua diversidade é cada vez maior e mais dispersa.

Assim, a capacidade de filtrar, de maneira facilitada, a informação fundamental num ficheiro onde os dados estão dispersos, ou pelo contrário, ainda que organizados a quantidade de informação é muito grande, torna-se necessária e a linguagem Python e o uso de *ER* é utilizada para tratar esses dados de forma eficiente.

O trabalho apresentado tem como objetivo usar as ferramentas apresentadas posteriormente, para extrair desses dados informação relevante.

### 1.2 Contexto

No âmbito da Unidade Curricular Processamento de Linguagens, foi proposto o trabalho prático que apresentamos neste documento sobre **ER + Filtros de Texto**, com o intuito de consolidar o que foi leccionado até então.

O propósito deste trabalho é aumentar a capacidade do grupo de escrever *Expressões Regulares* e, face ao documento fornecido pelos docentes, usar as *ER* para manipular os dados de uma maneira eficiente, para responder a todos os requisitos que iremos apresentar adiante neste documento.

### 1.3 Problema

O tema proposto ao grupo foi desenvolver um **Processador de Inscritos numa Atividade Desportiva**. O objetivo deste projecto é a manipulação de dados presentes num arquivo desportivo, criado por um

Organizador de *Provas de Orientação*, sobre os atletas que nelas participam e que são realizadas em diferentes locais e adaptadas a diferentes classes de participantes, e, como tal, pretendemos:

1. Especificar a informação pretendida, dadas as *tags* fornecidas;
2. Associar os escalões de cada atleta ao número de atletas que os constituem;
3. Gerar uma página HTML com dados que iremos descrever em detalhe posteriormente no documento.

## 1.4 Objetivo

Este documento pretende apresentar o primeiro trabalho prático proposto na Unidade Curricular, no qual o grupo procura contextualizar o problema face às adversidades no tratamento de dados no mundo real, bem como apresentar em concreto o problema que nos foi proposto pelos docentes, no âmbito da matéria lecionada.

Além disso, a finalidade fundamental é relatar todo o processo realizado pelo grupo, perante o problema identificado, onde foram aplicados os nossos conhecimentos sobre *ER* e *Filtros de Texto*.

## 1.5 Resultados

Face ao problema proposto, o grupo procedeu a uma análise e especificação do mesmo, no qual identificamos os requisitos e idealizamos a construção da solução.

Após essa fase, foram definidas as estruturas de dados e os algoritmos para desenhar e conceber a solução dos mesmos.

Podemos assim referir que os resultados obtidos foram os esperados pelo grupo, face às tomadas de decisão efetuadas, e apresentamos ambos os resultados, como as decisões que nos permitiram alcançá-los, posteriormente neste documento.

## 1.6 Estrutura do Relatório

Na fase inicial do documento, alusivo ao capítulo 1, é feita a introdução do problema, no qual é apresentada o enquadramento do mesmo, o seu contexto e ainda uma explicação formal sobre o mesmo. Nesta fase é ainda exposto o objetivo do relatório, bem como os resultados atingidos pelo grupo e ainda o presente tópico, referente a sua estruturação.

No capítulo 2, é feita uma análise detalhada do problema em questão, onde são especificados os requisitos do mesmo.

No capítulo 3, apresentamos as estruturas de dados e os algoritmos desenvolvidos para a determinação do problema.

De seguida, o capítulo 4 procura relacionar o código previamente implementado, com as tomadas de decisão que o grupo teve ao longo da procura da resposta aos obstáculos encontrados, em todo o projecto, bem como os testes feitos e os resultados respetivos.

No capítulo 5 termina o documento, com um pequeno resumo de tudo o que foi abordado, conclusões e trabalho futuro.

## Capítulo 2

# Análise e Especificação

Neste capítulo vamos abordar em detalhe o problema que nos foi proposto, e apresentar o levantamento de requisitos feito pelo grupo face ao problema em questão.

### 2.1 Descrição informal do problema

Através do ficheiro fornecido pelos docentes, pretendemos trabalhar sobre um arquivo desportivo que contém dados sobre os atletas, e das provas em que estes estão, ou estiveram inscritos, que diferem no nível de dificuldade e na sua localização.

### 2.2 Especificação dos Requisitos

Face o enunciado proposto e as suas exigências impostas, é de notar as seguintes:

1. Listar o nome (convertido para maiúsculas) de todos os concorrentes que se inscrevem como 'Individuais' e são de 'Valongo';
2. Listar o nome completo, email e a prova em que cada atleta está inscrito cujo nome seja 'Paulo' ou 'Ricardo', e que usem o GMail;
3. Listar toda a informação dos atletas da equipa 'TURBULENTOS'.
4. Listar quantos atletas estão inscritos em cada escalão existente, por ordem alfabética;
5. Criar uma página HTML:
  - (a) com a lista das equipas inscritas em qualquer prova, indicando o nome e o número dos atletas que fazem parte dela e que se inscreveram pelo menos uma vez na prova;
  - (b) cuja lista posteriormente descrita deve estar ordenada pela ordem decrescente do número de atletas;
  - (c) em que cada equipa deve ter um link para outra página HTML com informação interessante sobre cada atleta, indicando as provas em que cada um participou.



### 2.2.1 Dados

Como anteriormente descrito, o ficheiro *inscritos-form.json* contém a informação que pretendemos trabalhar, composto por 2253 linhas e que resulta em 83KB de tamanho.

Especificamente, os dados fornecidos sobre os atletas compõem um conjunto de sete campos, sendo eles o seu nome e data de nascimento, a sua morada e email, a prova no qual se inscreveu e o escalão em que pertence e, ainda a equipa em que pertence.

Salientamos que face aos dados recebidos, consideramos que os campos **data de nascimento** e **escalão** são **opcionalmente preenchidos**.

```
{
  "inscritos":[
    {
      "nome":"MARIO PIRES",
      "dataNasc":"04/04/59",
      "morada":"RUA CÂNDIDO DOS REIS , nº 86 , 2º ,",
      "email":"mgpires@netcabo.pt",
      "prova":"Ultra Trail",
      "escalao":"M50",
      "equipa":"Individual"
    },{
      "nome":"Francisco Neto Silva",
      "dataNasc":"22/04/89",
      "morada":"Rua da Ermida, 235",
      "email":"netosilva.francisco@gmail.com",
      "prova":"Corrida da Geira",
      "escalao":"SENIOR Masc",
      "equipa":"individual"
    }
  ]
}
```

Figura 2.1: Excerto do ficheiro *inscritos-form.json*

## Capítulo 3

# Concepção/desenho da Resolução

Neste capítulo vamos apresentar de forma específica a implementação da solução, utilizando a linguagem Python tal como referido anteriormente no documento, e os módulos 'er', 'sys' e 'os'.

### 3.1 Expressões Regulares

Para extrair a informação dos atletas, dividida em sete campos por cada atleta, foram escritas as seguintes expressões regulares:

```
#Campo "nome"
name_regex = re.compile(r'"nome":([\w\.\ ]+)')

#Campo "data de nascimento"
day = "([012]\d|3[0-1])"
month = "(0\d|1[012])"
year = "\d{2,4}"
birth_regex = re.compile(rf'dataNasc":({day}\/{month}\
/{year}|{year}\/{month}\/{day}|{day}\-{month}\-{year}|{year}\-{month}\-{day}|)'')

#Campo "morada"
address_regex = re.compile(r'"morada":([\w\-\.\.\ /, ]*)')

#Campo "email"
mail_regex = re.compile(r'"email":(\b([\w\-\ ]+\.)*[\w\-\ ]+@([\w\-\ ]+\.)*\w+\b)')

#Campo "prova"
prova_regex = re.compile(r'"prova":([\w \-:]+)')

#Campo "escalão"
escalao_regex = re.compile(r'"escalao":([\w ]+|)')

#Campo "equipa"
equipa_regex = re.compile(r'"equipa":([\w \-\.\ /\'&]+)')
```

## 3.2 Estruturas de Dados

Sendo que optamos por uma implementação *storage*, em que a leitura do ficheiro de dados é efetuada apenas uma vez e toda a informação relevante é mantida em memória, um ponto crucial do nosso projeto foi a escolha das estruturas de dados a utilizar.

Focamo-nos então no uso de listas e de dicionários presentes na linguagem usada, uma vez que a sua utilização e manipulação oferecem uma liberdade suficiente ao problema.

```
#alinea a -> nome de todos os atletas "individuais" e de "Valongo"
atletas = []
#alinea b -> nome, email e prova dos inscritos "Paulo" ou "Ricardo" that use "Gmail"
atletasb = []
#alinea c -> toda a info dos inscritos que pertencem à equipa "TURBULENTOS"
turbulentos = []
#alinea d -> dicionario com o número de atletas por escalao
escaloes = {}
#alinea e -> dicionario com informacao dos ateltas por equipa
equipas = {}
```

## 3.3 Algoritmos

De modo a garantirmos uma correta leitura do ficheiro *JSON* fornecido, criamos uma expressão regular que através da devida operação de procura, nos devolveu todas as ocorrências dos blocos de informação relativamente aos inscritos, delimitados por '{' e '}'.

```
#catchs all groups of info
cenas = re.findall(r'[{~}]+', conteudo) #because of [~} only catches the info we need
for group in cenas:
    parseGroup(group)
```

Em seguida, na função **parseGroups** usamos as expressões regulares na secção Expressões Regulares referidas para extrair todos os campos necessários de cada atleta para posterior interpretação e os algoritmos desenvolvidos para a solução foram os seguintes:

### 3.3.1 Requisito 1

Começamos por encontrar todos os atletas que potencialmente se inscreveram como "Individuais" e de forma a globalizar estes casos definimos a equipa como uma constante "Individual", também para tratamento posterior de informação. Posteriormente, verificamos se a morada pertencia a "Valongo" e em caso positivo, adicionamos o seu nome, convertido para maiúsculas à lista para este efeito.

```
#equipa = "Individual" & morada = "Valongo"
if re.match(r'(?i)(indiv)', equipa):
    equipa = "Individual"
    if re.search(r'(?i)(valongo)', address):
        atletas.append(name.upper())
```

### 3.3.2 Requisito 2

Através de duas condições de expressões regulares conseguimos encontrar todos os inscritos cujo nome contém "Paulo" ou "Ricardo" e cujo email usado é o "Gmail". Todas estas procuras foram feitas discriminando a sensibilidade de maiúsculas e minúsculas.

```
#nome = Paulo, ou nome = Ricardo & email = "GMail"
if re.match(r'(?i)(paulo|ricardo)', name) and re.match(r'.*(?i:)(gmail).*', email):
    atletasb.append((name, email, prova))
```

### 3.3.3 Requisito 3

Novamente, recorrendo a uma expressão regular *case insensitive* detetámos todos os inscritos da equipa "Turbulentos" e guardamos numa lista toda a informação referente aos mesmos.

```
#equipa = "turbulentos, Turbulentos ou TURBULENTOS"
if re.search(r'(?i)(^turbulentos)$', equipa):
    turbulentos.append((name, birth, address, email, prova, escalao, equipa))
```

### 3.3.4 Requisito 4

Para a resolução desta alínea, foi implementada uma estrutura de dados do Python, um **Dicionário**, que se apresentou neste capítulo previamente.

Assim, caso o escalão a tratar já exista no dicionário, isto é já exista uma *Key* com esse valor, o número de atletas nesse escalão (*Value*) é incrementado. Caso contrário, é adicionada uma nova entrada desse escalão com o *Value* a 1.

```
escaloes[escalao] = escaloes[escalao] + 1 if escalao in escaloes else 1
```

É de salientar que esta inserção ou atualização não garante a ordenação do dicionário por ordem alfabética.

### 3.3.5 Requisito 5

Para garantirmos toda a informação dos atletas constituintes de cada equipa, decidimos, como anteriormente referido, recorrer ao uso de um dicionário cujo valor de *key* é o nome da equipa. A cada equipa está associada uma lista em que cada entrada dessa lista corresponde à informação de um dos atletas que a constitui.

Por razões de coerência decidimos representar o nome da equipa com insensibilidade a maiúsculas e minúsculas, e de modo a preservar essa ideia, a inserção de uma chave no dicionário é feita recorrendo previamente à função *lower* que converte o nome da equipa para minúsculas. É ainda verificada a duplicação de informação através do nome e do email do atleta, e caso encontrada é acrescentada a nova prova.

```
equal = False
equipa_low = equipa.lower()
if equipa_low in equipas:
    for(n,_,e,_,_) in equipas[equipa_low]:
        #if name and email are the same we consider the same person
        if n==name and e==email:
            if prova not in p: p.append(prova)
            equal = True
            break
if not equal:
    equipas[equipa_low].append((name, birth, email, prova, escalao))
else:
    equipas[equipa_low] = [(name, birth, email, prova, escalao)]
```

### 3.4 Funcionamento do programa

De modo a facilitar o processo de compilação, foi desenvolvida uma *Makefile* que permite duas funcionalidades, sendo elas as seguintes:

1. o comando *make run* permite a compilação e execução do programa;
2. o comando *make clean* é responsável por apagar todos os ficheiros e diretorias geradas pela solução

Mostramos de seguida a abordagem idealizada para implementar a *Makefile*.

```
PYTHON = python3.9
.PHONY = help run clean
.DEFAULT_GOAL = help

help:
    @echo "-----HELP-----"
    @echo "To run the project type make run"
    @echo "To clean project files type make clean"
    @echo "-----"

run:
    ${PYTHON} main.py

clean:
    rm -rf /html/*.html html
    rm -rf equipas.html
```

## Capítulo 4

# Codificação e Testes

Neste capítulo vamos descrever todas as tomadas de decisão que o grupo tomou na concepção deste trabalho prático, e ainda apresentar imagens dos resultados que obtivemos no programa desenvolvido:

### 4.1 Alternativas, Decisões e Problemas de Implementação

Face ao documento fornecido e aos dados que este continha, o grupo tomou decisões com o intuito de responder às questões que nos foram propostas no enunciado.

Uma das várias decisões refere-se aos erros ortográficos que os dados continham, mais precisamente no campo "*equipa*", e que resultou em considerarmos os campos que estavam escritos como 'INDIVIDUAL', ou com outros erros de ortografia, como campos referentes a equipas do tipo 'Individual'.

Outra decisão que envolveu a forma como uma palavra foi inserida foi sobre a equipa 'TURBULENTOS', e consideramos que as inserções de "Turbulentos" ou "turbulentos" como pertencentes à mesma equipa, pois a única parte em que diferem é serem escritas em Upper ou Lower Case. Pelo contrário, consideramos a entrada 'Os Turbulentos' como sendo uma equipa diferente da prévia.

Ainda sobre as equipas, consideramos as entradas correspondentes a equipas 's/ clube' como sendo esse o nome da equipa a que esses atletas pertencem.

Tal como referido anteriormente no documento, o campo "*escalão*" pode ser ou não preenchido. No caso de não ter nenhuma entrada, substituímos a entrada vazia por *sem escalao* para efeitos de apresentação no requisito 4.

Para responder ao último requisito, referente à concepção da página HTML, fizemos um *refactoring* das equipas para lower case, para assumir que equipas que sejam do mesmo nome e apenas diferem na sua forma de escrita, sejam a mesma equipa na verdade, e, ainda, que se duas ou mais entradas do arquivo possuírem o mesmo nome e email, são a mesma pessoa.

## 4.2 Abordagens na exibição e respectivos resultados

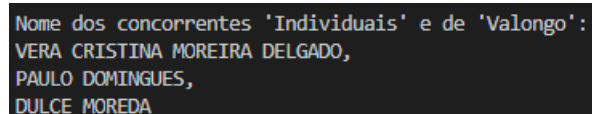
Em seguida vamos mostrar as abordagens adotadas pelo grupo para apresentar de uma forma clara e eficiente toda a informação necessária e os respectivos resultados obtidos:

### 4.2.1 Requisito 1

De modo a apresentar todos os nomes dos concorrentes que se inscreveram na atividade como '*Individuais*' e cuja morada seja '*Valongo*' iteramos a lista responsável por armazenar esta informação, previamente convertida para maiúsculas.

```
def showCommand1():  
    print("Nome dos concorrentes 'Individuais' e de 'Valongo':")  
    print(',\n'.join(atletas))
```

Como resultado para o primeiro requisito, obtivemos os seguintes atletas registados como '*Individuais*' e são de '*Valongo*':



```
Nome dos concorrentes 'Individuais' e de 'Valongo':  
VERA CRISTINA MOREIRA DELGADO,  
PAULO DOMINGUES,  
DULCE MOREDA
```

Figura 4.1: Resposta ao Requisito 1

### 4.2.2 Requisito 2

Para o segundo requisito, os atletas cujo nome é '*Paulo*' ou '*Ricardo*' e que usam o '*GMail*', o seguinte código foi implementado com a intenção de mostrar os resultados sob a forma de uma tabela. Para tal recorremos ao uso da função *print* consoante determinados formatos.

```
print("Concorrentes cujo nome é 'Paulo' ou 'Ricardo' e cujo email é 'Gmail' :")  
print('-'*111)  
print('| {:<45}| {:<40}| {:<20}|'.format("Nome", "Email", "Prova"))  
print('-'*111)  
for (nome, email, prova) in atletasb :  
    print('| {:<45}| {:<40}| {:<20}|'.format(nome, email, prova))  
print('-'*111)
```

Encontramos os seguintes atletas:

Concorrentes cujo nome é 'Paulo' ou 'Ricardo' e cujo email é 'Gmail' :

Nome	Email	Prova
paulo de castro rocha	pcastrorocha@gmail.com	Ultra Trail
Paulo Serra	paulo.serra@gmail.com	Ultra Trail
paulo Vilaça	pmv777@gmail.com	Ultra Trail
Paulo Domingues	p.j.p.domingues@gmail.com	Ultra Trail
Ricardo Jorge Dias Oliveira	Ricardo.transportesabranco@gmail.com	Ultra Trail
Ricardo Reis	ricardoreis@gmail.com	Ultra Trail
paulo felix	pauloalexteixeirafelix@gmail.com	Ultra Trail
Ricardo Sousa	jose.ricardo.sousa@gmail.com	Corrida da Geira
Ricardo Jorge Dias Oliveira	helderfva@gmail.com	Ultra Trail
Ricardo Couto	rjcouto@gmail.com	Ultra Trail
Ricardo Ernesto dos Santos Geraldes Domingues	santosgeraldes@gmail.com	Corrida da Geira

Figura 4.2: Resposta ao Requisito 2

### 4.2.3 Requisito 3

Para o requisito 3, apresentado neste documento, o objetivo era encontrar atletas cuja equipa a que pertencem é 'TURBULENTOS', e foi desenvolvida a seguinte instrução:

```
print("Informação dos atletas da equipa 'Turbulentos':")
for(name, birth, address, email, prova, escalao, equipa) in turbulentos:
    print('[Nome      = %s\n DataNac = %s\n Morada   = %s\n Email    = %s\n Prova    = %s\n Escalao = %s\n Equipa   = %s]\n' %
          (name, birth, address, email, prova, escalao, equipa))
```

A solução encontrada foi a seguinte:

```
[Nome      = Tiago Domingues Frada
DataNac    = 10/01/82
Morada     = Rua Germão Galharde, nº 26, 3º tro Trás, 4715-290 Braga
Email      = tiagofrada@gmail.com
Prova      = Corrida da Geira
Escalao    = SENIOR Masc
Equipa     = TURBULENTOS]

[Nome      = António José Costa
DataNac    = 04/05/60
Morada     = rua costa soares - braga
Email      = tozecosta1960@hotmail.com
Prova      = Corrida da Geira
Escalao    = SENIOR Masc
Equipa     = Turbulentos]

[Nome      = Tiago Domingues Frada
DataNac    = 10/01/82
Morada     = Rua Germão galharde, nº 26, 3º Ctro Trás, Braga
Email      = tiagofrada@gmail.com
Prova      = Corrida da Geira
Escalao    = SENIOR Masc
Equipa     = TURBULENTOS]
```

Figura 4.3: Excerto dos atletas da equipa 'Turbulentos'

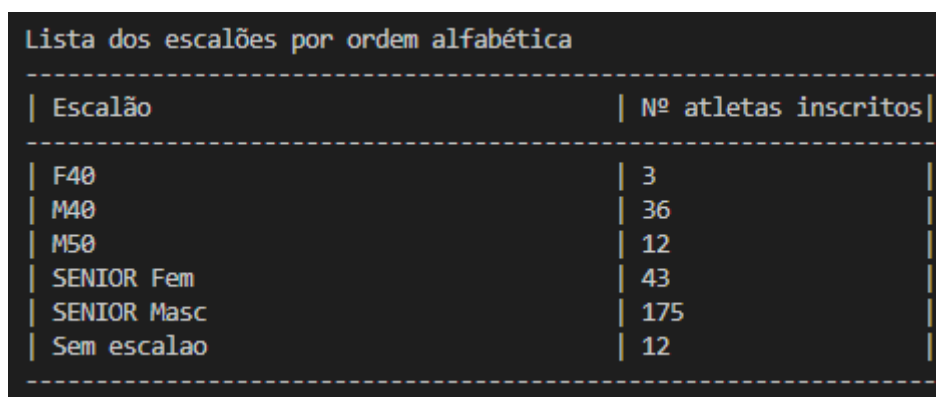


#### 4.2.4 Requisito 4

De forma a apresentar todos os escalões por ordem alfabética e o número de atletas neles inscritos, foi desenvolvido o seguinte código que ordena o dicionário *escaloes* pelas suas *keys* (escalão) e iterando posteriormente o resultado da ordenação apresenta a tabela pretendida.

```
print("Lista dos escalões por ordem alfabética")
print('-'*65)
print('| {:<40}| {:<20}|'.format("Escalão", "Nº atletas inscritos"))
print('-'*65)
for k,v in sorted(escaloes.items()):
    print('| {:<40}| {:<20}|'.format(k,v))
print('-'*65)
```

O resultado equivalente é o seguinte:



Escalão	Nº atletas inscritos
F40	3
M40	36
M50	12
SENIOR Fem	43
SENIOR Masc	175
Sem escalao	12

Figura 4.4: Resposta ao Requisito 4

#### 4.2.5 Requisito 5

Para o **Requisito 5** foi pedido ao grupo para gerar uma página HTML com uma lista de todas as equipas e o número de atletas que a constituem. Começamos por basear-nos em exercícios lecionados nas aulas práticas da UC e a nossa abordagem foi implementar a estrutura base de um ficheiro *HTML* com os necessários headers e informações que achamos relevantes. Este ficheiro tomou o nome de **"equipas.html"**

Para fazer a correta amostragem da estrutura de dados *equipas*, isto é por ordem decrescente do número de atletas, o dicionário foi ordenado por ordem inversa sob o critério do comprimento dos seus *values*, ou seja, da lista de atletas de cada equipa.

Deste modo iterando o resultado desta ordenação criamos a devida listagem do nome da equipa e este comprimento que corresponde ao número de atletas que a constitui.

Em cada iteração, isto é, para cada equipa é ainda originado um caminho para um ficheiro criado pela função auxiliar *equipaHTML(equipa, file\_name)*, que contém toda a informação detalhada da constituição da mesma. Este caminho é associado ao nome de cada equipa na página inicial. Desta forma cumprimos o requisito de cada equipa ter um link para uma outra página *HTML* com a informação que consideramos relevante e as diferentes provas que um atleta participou.

Foi ainda criada uma pasta **html** responsável por guardar todos os ficheiros *HTML* de cada uma das equipas acima referidos.

```

#creates html file with name of all teams and number of elements
def equipasHTML():
    if not os.path.exists('html'):
        os.makedirs('html')

    f = open('./equipas.html', 'w')

    docHTML = """
<!DOCTYPE html>
<html>
<head>
<!--titulo associada a barra do browser-->
<title>Organizador de provas de Orientação</title>
<meta charset="UTF-8">
</head>
<body>
<h1>Trabalho PL 1</h1>
<h2>Processador de Inscritos numa atividade Desportiva</h2>

<h3>Nome das equipas e número de atletas que a constituem</h3>
<ul>"""

    for equipa in sorted(equipas, key = lambda key: len(equipas[key]), reverse=True):
        file_name = "./html/" + "_".join(re.split(r"[/ ]", equipa)) + ".html"
        equipaHTML(equipa, file_name)
        docHTML = docHTML + """
            <li> <a href ="" + file_name+ ">" + equipa + "</a> = " + str(len(equipas[equipa])) + "</li>

    docHTML += """
        </ul>
</body>
<div class="grupo">
    <p><strong>Grupo 70:</strong>
    <ul>
        <li>Ana César a86038</li>
        <li>Margarida Faria a71924</li>
    </ul>
    </p>
</div>
</html>
"""
    f.write(docHTML)
    f.close()

```

A página HTML concebida pelo ficheiro "equipas.html" foi a seguinte:

## Trabalho PL 1

### Processador de Inscritos numa atividade Desportiva

Nome das equipas e número de atletas que a constituem

- [individual](#) = 61
- [turbulentos](#) = 37
- [nast](#) = 11
- [edv viana trail](#) = 10
- [arrastassolas](#) = 8
- [print team](#) = 7
- [porto runners](#) = 6
- [companhia do bazófilas](#) = 6
- [clube atletismo de lamas](#) = 5
- [clube de veteranos do porto](#) = 5
- [clube náutico de ponte de lima](#) = 5
- [,](#) = 5
- [tugas na estrada](#) = 4
- [multipower l gaia trail](#) = 4
- [clube spiridon de gaia](#) = 4
- [os barriguitas](#) = 3
- [edv-viana trail](#) = 3
- [sopro run life](#) = 3
- [os caga tacos running team](#) = 2
- [.com](#) = 2
- [templartrail](#) = 2
- [bracara runners](#) = 2
- [urban team](#) = 2
- [jc&niki](#) = 2
- [prt/ad quinta](#) = 2
- [clube náutico ponte de lima](#) = 2
- [os turbulentos](#) = 2
- [cabritos da cortiça - trail](#) = 2
- [pedros & associados](#) = 2
- [clube portugal telecom zona norte](#) = 2
- [bando dos trilhos e tralhos, associação betetista](#) = 2
- [cães da avenida](#) = 1
- [os caga tacos running team](#) = 1
- [s/ clube](#) = 1
- [blue o](#) = 1
- [born /be wild](#) = 1
- [na](#) = 1
- [olhares sublimes](#) = 1
- [união fci de tomar](#) = 1
- [team duas faces](#) = 1
- [duro vl](#) = 1
- [os turbulentos](#) = 1
- [ctad tiagoaragao.com](#) = 1
- [xico runners](#) = 1

Figura 4.5: Excerto da página HTML concebida com a informação das equipas

Como indicado, a função prévia `equipasHTML` chama a função auxiliar `equipaHTML(equipa, file_name)` que permite iterar cada equipa e extrair a informação relevante dos atletas que a compõem. Em seguida mostra-se o exemplo para todos os atletas que se inscreveram como 'Individuais' e a página obtida foi a seguinte:

# Trabalho PL 1

## Processador de Inscritos numa atividade Desportiva

### Constituição detalhada da equipa 'individual'

- Nome: MARIO PIRES
  - ▶ Mais informação
- Nome: Francisco Neto Silva
  - ▶ Mais informação
- Nome: Artur Bernardo
  - ▶ Mais informação
- Nome: Vera Cristina Moreira Delgado
  - ▶ Mais informação
- Nome: Jorge Yong
  - ▶ Mais informação
- Nome: Paulo Serra
  - ▶ Mais informação
- Nome: Tiago José Cadima Borges
  - ▶ Mais informação
- Nome: jorge manuel martins silva
  - ▶ Mais informação
- Nome: Paulo Domingues
  - ▶ Mais informação
- Nome: Dulce Moreda
  - ▶ Mais informação
- Nome: António Fernandes
  - ▼ Mais informação
    - Data de Nascimento:24/05/77
    - Email: cisterbtt@gmail.com
    - Provas: Corrida da Geira; Ultra Trail
    - Escalão: SENIOR Masc
- Nome: Angelo Senra
  - ▶ Mais informação
- Nome: Paulo Jorge
  - ▶ Mais informação
- -- --

Figura 4.6: Excerto da página HTML concebida com a informação dos atletas

É de realçar que pela imagem podemos verificar que cumprimos o requisito de para cada atleta listar as provas em que se inscreveu.

## Capítulo 5

# Conclusão

Face ao trabalho desenvolvido e aos resultados obtidos, podemos concluir que conseguimos estabelecer conjuntos de padrões que processam o ficheiro *inscritos-form.json* recebido de um modo preciso e eficiente. Foi possível, com o uso das ferramentas apresentadas neste documento, responder aos requisitos concretos pretendidos, que passaram por filtrar informação relevante dos dados que constituem o documento, bem como a conceção de uma página HTML com a informação que compõe o arquivo desportivo referente às equipas que se inscreveram em Provas de Orientação, e ainda para cada equipa, uma hiperligação para outra página HTML com a informação relevante dos atletas constituintes.

Assim, com a elaboração do trabalho e o relatório correspondente, constatamos que de facto o *Python* é uma linguagem de alto nível que combinado com as *ER* desenvolvidas, revelam uma capacidade gigante para a elaboração de *Processadores de Linguagens Regulares* ou de *Filtros de Texto*.

## Apêndice A

# Apresentação dos Resultados

Para a exposição dos resultados obtidos, foi desenvolvida uma interface de utilizador em modo de texto, que apresentamos de seguida:

```
----- Processador de inscritos numa atividade Desportiva -----  
|  
| 1- nome dos atletas inscritos como 'Individuais' e são de 'Valongo'.  
| 2- nome, telemóvel e prova dos inscritos cujo nome é 'Paulo' ou 'Ricardo' e usam o Gmail.  
| 3- info equipa 'TURBULENTOS'.  
| 4- lista dos escalões por ordem alfabética e número de atletas inscritos  
| 5- página HTML com a lista de equipas inscritas em qualquer prova.  
| 0- sair  
|  
-----  
Selecione a sua opção:  
█
```

Figura A.1: User Interface