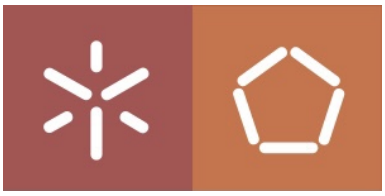
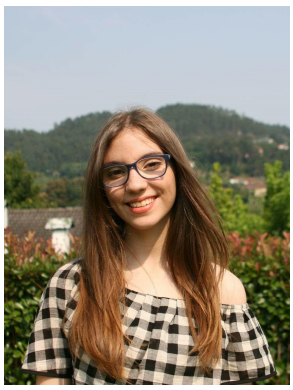


Processamento de Linguagens

TP1: Processador de Registos de Exames Médicos Desportivos

Grupo 29

27 de março de 2022



Alexandra Candeias (A89521) Patrícia Pereira (A89578) Meriem Khammassi (A85829)

Índice

1 Introdução **2**

2 Estrutura do Relatório **2**

3 Exercício e o seu Dataset **3**

4 Resolução do Exercício **4**
4.1 Main Program emd.py 4
4.1.1 Datas externas dos registos no DataSet 4
4.1.2 Distribuição por género em cada ano e no total 5
4.1.3 Distribuição por modalidade em cada ano e no total 5
4.1.4 Distribuição por idade e género 6
4.1.5 Distribuição por morada 6
4.1.6 Distribuição por estatuto de federado em cada ano 6
4.1.7 Percentagem de aptos e não aptos por ano 6
4.2 HTML Creator - createHTML.py 7
4.2.1 CreateHTML 7
4.2.2 CreateYearHTML 7
4.2.3 CreateHTMLPages 8

5 Exemplo de páginas web **8**
5.0.1 Pagina Inicial 8
5.0.2 Página referente aos exames no ano 2019 9
5.0.3 Página referente aos géneros que realizaram exames no 2019 9

6 Conclusão **12**

Resumo

Neste trabalho trabalho foi-nos proposto desenvolver um Processador de Registos de Exames Médicos Desportivos, através da escrita de Expressões Regulares (ER), que permitam processar um ficheiro de csv.

O objetivo do trabalho é, assim, desenvolver um programa, recorrendo a ER, que permitam analisar os dados presentes no ficheiro csv e apresenta-los num Website de uma forma organizada.

1 Introdução

No âmbito da Unidade Curricular (UC) de Processamento de Linguagens, de forma de desenvolver e demonstrar o nosso conhecimento sobre a matéria que tem vindo a ser alvo de atenção na UC, foi proposta a a escolha de um exercício entre vários fornecidos pela equipa docente da UC.

No enunciados existiam três exercícios diferentes, sendo que o nosso grupo optou pelo o exercício três, a construção de um Processador de Registos de Exames Médicos Desportivos.

Assim, no presente relatório, iremos descrever o processo de resolução do problema, bem como explicar o código implementado para o mesmo.

2 Estrutura do Relatório

- O capitulo seguinte corresponde à análise e especificação. É neste que é feita uma análise detalhada do problema proposto de modo a se poder especificar o desenvolvimento do mesmo.
- De seguida, será apresentado e explicado o raciocínio feito para resolução do problema, e falaremos do código que necessitamos de implementar.
- Na última parte apresentamos o resultado final pretendido.

3 Exercício e o seu Dataset

Como mencionado previamente, o exercício alvo de avaliação e resolução por parte do grupo é o exercício três.

Neste exercício pretende-se trabalhar um dataset criado no âmbito do registo de exames médicos desportivos. O pretendido é a construção de um ou vários programas Python para processar o dataset 'emd.csv' que permita(m) verificar ou calcular as seguintes informações:

- Datas externas dos registos no DataSet
- Distribuição por género em cada ano e no total
- Distribuição por modalidade em cada ano e no total
- Distribuição por idade e género (para a idade, considera apenas 2 escalões: < 35 anos e $>= 35$)
- Distribuição por morada
- Distribuição por estatuto de federado em cada ano
- Percentagem de aptos e não aptos por ano

Analisando o ficheiro 'emd.csv' percebemos qual a expressão regular a desenvolver que nos permitisse filtrar e obter a informação necessária para cada um dos desafios propostos. Assim para cada linha do dataset identificamos as seguintes infprmações que precisamos de guardar e seriam as nossas categorias:

- ANO
- IDADE
- GÉNERO
- MORADA
- MODALIDADE
- FEDERADO
- RESULTADO

4 Resolução do Exercício

Neste momento do relatório passamos à explicação sucinta do raciocínio e o código presentes na resolução do problema, de modo a que seja facilmente percetível aquilo que o grupo pretendia demonstrar com os seus conhecimentos.

Achamos relevante mencionar desde já, e para não haver necessidade de repetição, que para a resolução das obter a informação referida nas alíneas, dividimos a informação fornecida em grupos utilizando a ER seguinte, com o auxilio do *search* da biblioteca re.

Este excerto de código ajuda a dividir os dados que são relevantes para a resolução do problema e nos dá acesso aos mesmos.

```
#Usados para construção de um elemento de cada vez
linhaElemento = 0

#Ficheiro a ser interpertado
file = open("emd.csv",'r', encoding="utf-8")

#Análise do ficheiros e dos seus dados, bem como guardar os mesmos num dicionário
startData = time.time()

for linha in file:
    if campos := re.search(r'(\d+)-\d+-\d+\,\w+\,\w+\,(\d+)\,(\w)\,(\w+)\,([\w]+)\,[\w]+\,[\w.]+@[\w.]+\,(\w+)\,(\w+)',linha):

        linhaElemento = nrLinha
        anoAtual = campos.group(1)
```

4.1 Main Program emd.py

De forma de facilitar a utilização do nosso programa e não haver a necessidade de correr separadamente cada alínea, para todas as alíneas apresentadas a seguir, decidimos usar um dicionário para armazenar todas as informações de cada grupo.

4.1.1 Datas externas dos registos no DataSet

Após guardados os anos dos exames existentes no ficheiro de dados, usamos o seguinte excerto de código para obter as categorias resultantes de cada ano tendo em conta as informações associadas a cada.

```
anoAtual = campos.group(1)

nomePessoa = " ".join(linha.split(',')[3:5])

if anoAtual not in resultadosAnos:
    resultadosAnos[anoAtual] = {
        "genero": {
            "M": [],
            "F": []
        },
        "modalidade": {},
        "genero por idade": {
            "M, >= 35": [],
            "M, < 35": [],
            "F, >= 35": [],
            "F, < 35": [],
        },
        "morada": {},
        "federado": {
            "false": [],
            "true": [],
        },
        "aptos": {
            "false": [],
            "true": []
        }
    }
```

4.1.2 Distribuição por género em cada ano e no total

Nesta alínea, selecionamos o 3º grupo da nossa ER para obter o género de cada pessoa existente no nosso Dataset.

Assim, para conseguir apresentar estes resultados da forma mais adequada e iterativa, criamos gráficos de barras contendo os valores pretendidos para cada categoria.

```
genero = campos.group(3)
resultadosAnos[anoAtual]['genero'][genero].append(nomePessoa)
resultadosAnos['total']['genero'][genero].append(nomePessoa)
```

De seguida, respondemos às seguintes alíneas da mesma forma para simplificar o nosso código.

4.1.3 Distribuição por modalidade em cada ano e no total

```
modalidade = campos.group(5)
if modalidade not in resultadosAnos[anoAtual]['modalidade'] :
    resultadosAnos[anoAtual]['modalidade'][modalidade] = [nomePessoa]
else :
    resultadosAnos[anoAtual]['modalidade'][modalidade].append(nomePessoa)

if modalidade not in resultadosAnos['total']['modalidade'] :
    resultadosAnos['total']['modalidade'][modalidade] = [nomePessoa]
else :
    resultadosAnos['total']['modalidade'][modalidade].append(nomePessoa)
```

4.1.4 Distribuição por idade e género

```
idade = campos.group(2)

if int(idade) < 35 :
    resultadosAnos[anoAtual]['genero por idade'][genero + ', < 35'].append(nomePessoa)
    resultadosAnos['total']['genero por idade'][genero + ', < 35'].append(nomePessoa)
else :
    resultadosAnos[anoAtual]['genero por idade'][genero + ', >= 35'].append(nomePessoa)
    resultadosAnos['total']['genero por idade'][genero + ', >= 35'].append(nomePessoa)
```

4.1.5 Distribuição por morada

```
morada = campos.group(4)
if morada not in resultadosAnos[anoAtual]['morada'] :
    resultadosAnos[anoAtual]['morada'][morada] = [nomePessoa]
else :
    resultadosAnos[anoAtual]['morada'][morada].append(nomePessoa)

if morada not in resultadosAnos['total']['morada'] :
    resultadosAnos['total']['morada'][morada] = [nomePessoa]
else :
    resultadosAnos['total']['morada'][morada].append(nomePessoa)
```

4.1.6 Distribuição por estatuto de federado em cada ano

```
federado = campos.group(6)
resultadosAnos[anoAtual]['federado'][federado].append(nomePessoa)
resultadosAnos['total']['federado'][federado].append(nomePessoa)
```

4.1.7 Percentagem de aptos e não aptos por ano

Para resolver esta alínea bastava seguir a mesma forma usada nas anteriores para obter a informação deste grupo e armazená-lo num dicionário ao qual vamos buscar quando executar o nosso programa.

```
apto = campos.group(7)
resultadosAnos[anoAtual]['aptos'][apto].append(nomePessoa)
resultadosAnos['total']['aptos'][apto].append(nomePessoa)
```

No entanto, para obter os resultados pretendidos recorremos ao método createHTML-Pages, que apresentaríamos na próxima secção, no qual calculamos a soma dos valores e convertemos estes em percentagem.

```
if categoria == "aptos" :
    values = [len(v) for v in elementos.values()]
    total = sum(values)
    labels = [key + " : " + str(round(len(elementos[key])/total * 100, 2)) + "%" for key in elementos]
    plt.pie(values, labels = labels)
```

4.2 HTML Creator - createHTML.py

Para proceder à criação das nossas páginas WEB decidimos recorrer a um ficheiro separado do nosso ficheiro principal.

Este trata-se de um "gerador de páginas html"tendo 3 métodos para tal.

4.2.1 CreateHTML

Esta Função cria o HTML da página principal do Projeto onde podemos aceder a não aos vários anos existentes no nosso dataset, como também o total dos exames realizados.

```
#Função que cria o HTML da página principal do Projeto onde podemos aceder às várias outras categorias
def createHTML(resultadosAnos) :
    nrAnos = len(resultadosAnos)
    AnoNr = 1

    with open ("index.html", 'w', encoding="utf-8") as ficheiro:
```

```
    for (ano, categoriasAno) in resultadosAnos.items():
        createYearHTML(ano, categoriasAno)
        pathFicheiro = 'pages/'+ano.lower()+'.html'
        ficheiro.write(f'''
            <a href="{pathFicheiro}">
                <li class="w3-bar w3-mobile w3-hover-orange">
                    <span class="categoria w3-bar-item w3-mobile">{ano}</span>
                </li>
            </a>''')
        if AnoNr != nrAnos:
            ficheiro.write('''
            <hr>''')
        AnoNr += 1
```

4.2.2 CreateYearHTML

Para avançar da nossa página principal às várias categorias existentes para cada ano específico, decidimos implementar o seguinte método, que, com a escolha do ano pretendido ou à totalidade existente dos nossos processos, conseguimos abrir a subpágina.

```
#Função que cria o HTML das subpáginas de anos do Projeto onde podemos aceder às várias outras categorias
def createYearHTML(ano, resultadosCategorias):

    nrCategorias = len(resultadosCategorias)
    categoriaNr = 1

    with open ('pages/'+ano.lower()+'.html', 'w', encoding="utf-8") as ficheiro:
```

```
for (categoria, elementosCategoria) in resultadosCategorias.items():
    createHTMLPages(ano, categoria, elementosCategoria)
    nrElementos = sum(len(values) for key, values in elementosCategoria.items())
    pathFicheiro = 'pages/'+ano.lower()+'/'+categoria.lower()+'.html'
    ficheiro.write(f'''
        <a href="{ano.lower()+'/'+categoria.lower()+'.html'}">
            <li class="w3-bar w3-mobile w3-hover-orange">
                <span class="categoria w3-bar-item w3-mobile">{categoria}</span>
                <span class="nrElementos w3-bar-item w3-right w3-mobile">{nrElementos} elementos</span>
            </li>
        </a>''')
    if categoriaNr != nrCategorias:
        ficheiro.write('''
        <hr>''')
    categoriaNr += 1
```

4.2.3 CreateHTMLPages

Por fim, criamos o método CreateHTMLPages para obter as páginas relativas a uma categoria em específico.

```
def createHTMLPages(ano, categoria, elementos):

    if not os.path.exists('pages/'+ano.lower()) :
        os.makedirs('pages/'+ano.lower())

    pathFicheiro = 'pages/'+ano.lower()+'/'+categoria.lower()+'.html'
    nrElementos = sum(len(values) for key, values in elementos.items())
    elementoNr = 1

    if not os.path.exists('figures') :
        os.makedirs('figures')


    with open (pathFicheiro, 'w', encoding="utf-8") as ficheiro:
```

5 Exemplo de páginas web

5.0.1 Pagina Inicial

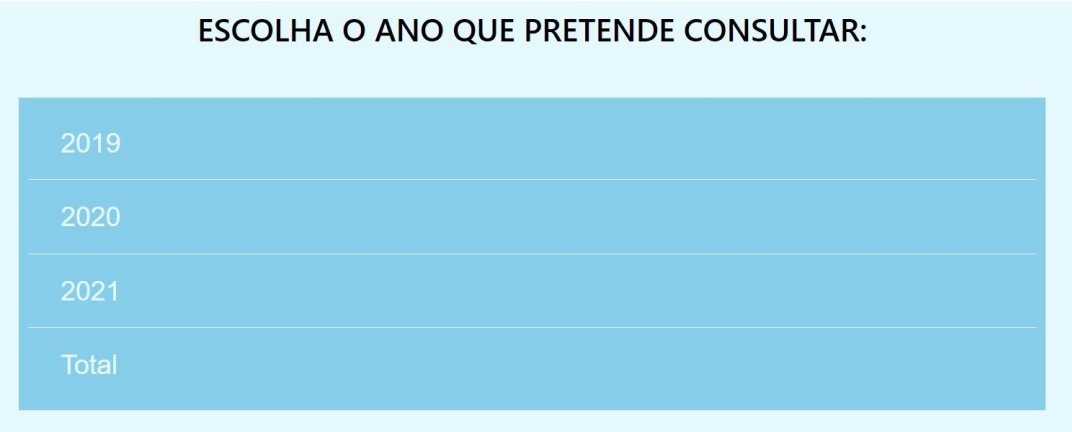


Figura 1: Tabela de anos

Na página inicial do website são apresentados os anos em que foram efetuados exames médicos. clicando em cada um dos anos seremos redirecionados para outra página web com

categorias presentes no dataset inical. Podemos ainda aceder a uma página para consultar o total de exames.

5.0.2 Página referente aos exames no ano 2019

Figura 2: Tabela de categorias

Após escolher o ano pretendido, podemos consultar os dados pelas diversas categorias, sendo redirecionados, para outra página web. Na figura acima podemos aceder às categorias do ano 2019.

Do lado direito da categoria é exposto o número de exames na qual existem dados sobre essa categoria.

5.0.3 Página referente aos géneros que realizaram exames no 2019

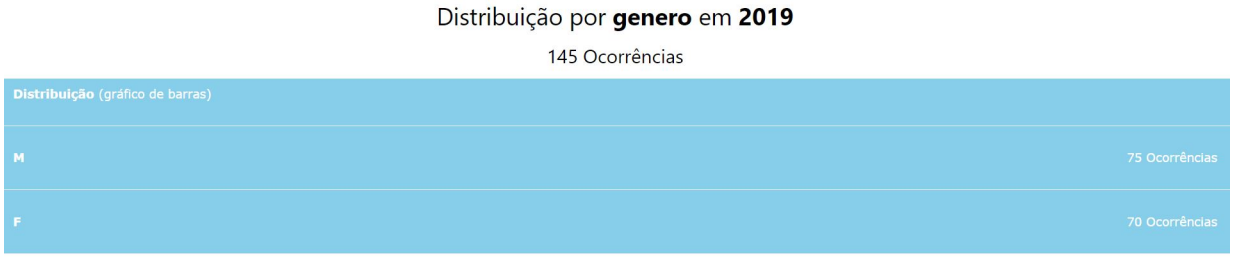


Figura 3: Tabela género por ano

Escolhendo a categoria, serão expostos os diferentes dados dessas categorias. No exemplo, correspondente ao género no ano 2019, temos que, existem dois tipos de dados diferentes referentes a essa categoria, masculino e feminino. No lado direito temos o numero de ocorrências desse tipo de dados no dataset.

Acedendo a cada um destes podemos visualizar os nomes dos atletas que se enquadram em cada uma delas.

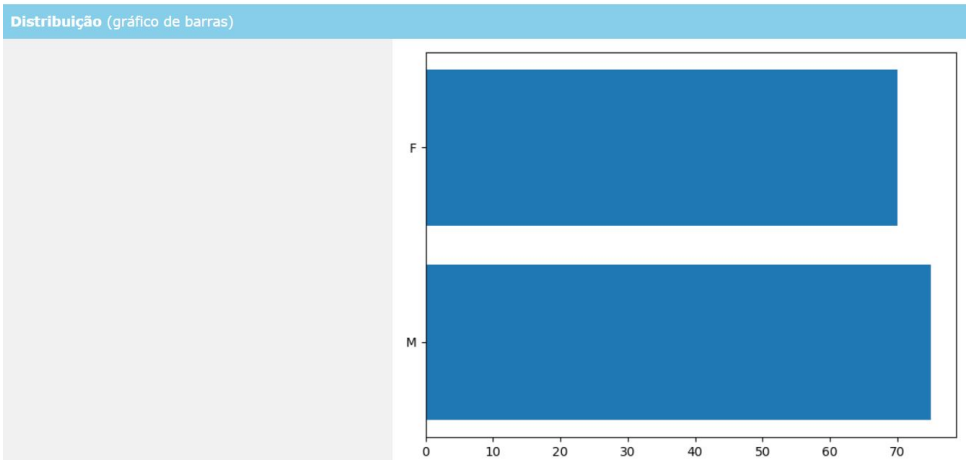


Figura 4: Gráfico de barras genero por ano

- Podemos ainda consultar um gráfico de barras que representa esses dados.

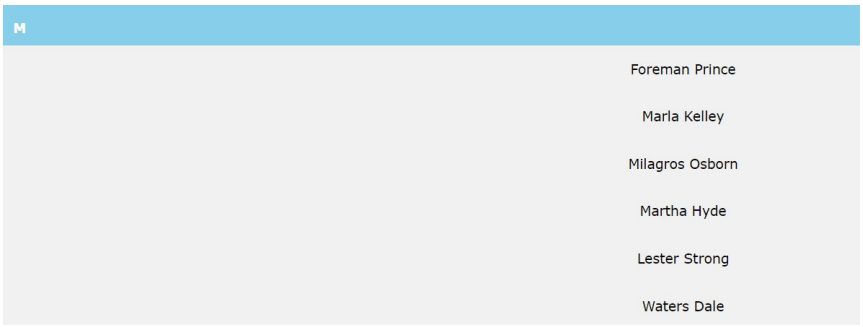


Figura 5: Atletas do sexo feminino (exames ano 2019)

- Para a categoria aptos achamos que faria mais sentido apresentar um gráfico circular.

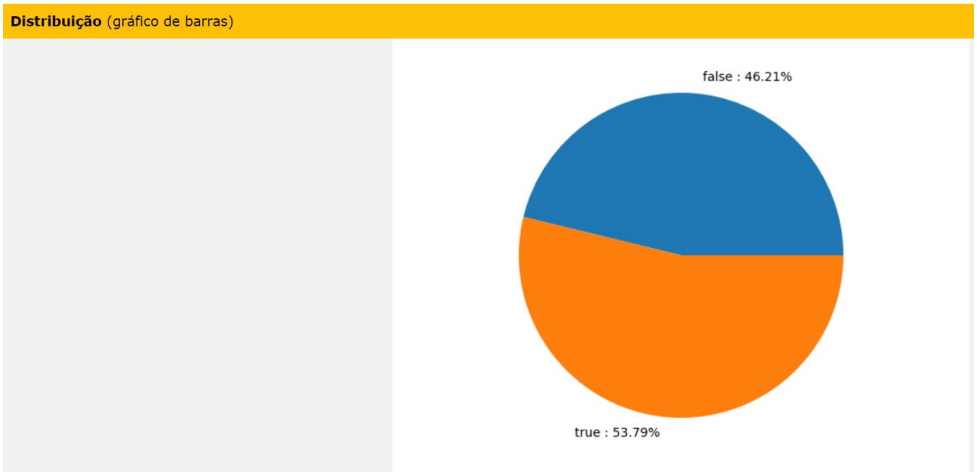


Figura 6: gráfico circular de atletas aptos em 2019

6 Conclusão

Perante a conclusão e apresentação da resolução a que o grupo chegou para o primeiro trabalho prático na UC de Processamento de Linguagens, é importante mencionar as diversas dificuldades e obstáculos que tivemos que ultrapassar, bem como algumas falhas que poderiam ser, e podem vir a ser, melhoradas.

Começando pelo Dataset que nos foi fornecido, como mencionamos no início deste relatório, trouxe-nos bastantes promenores que não tínhamos como controlar sem manipular diretamente o mesmo.

No entanto, o trabalho prático revelou-se uma mais valia para cimentar o nosso conhecimento sobre Expressões Regulares, bem como as linguagens Python e HTML. Sendo ambos alvo de grande interesse pelos elementos do grupo, foi sempre um trabalho que nos motivou a fazer mais e melhor, com o intuito de obter o melhor resultado possível.

Através do exercício que o grupo competiu resolver, *Processador de Registos de Exames Médicos Desportivos*, consideramos que as nossas capacidades foram efetivamente comprovadas, mostrando o domínio do uso de Expressões Regulares, bem como a manipulação de informação com a Linguagem Python.

Assim, fazemos uma avaliação positiva do nosso desempenho neste trabalho, já que o grupo respondeu de forma autónoma e correta a ao problema apresentado, revelando um trabalho consistente e bem estruturado.