



UNIVERSIDADE DO MINHO
MESTRADO EM MATEMÁTICA E COMPUTAÇÃO

Introdução aos Algoritmos e à Programação

Trabalho Prático 3

Eduardo Teixeira Dias (PG52249)

Tiago Augusto Lopes Monteiro (PG52258)

7 de janeiro de 2024

Conteúdo

1	Introdução	3
1.1	<i>Web Scraping</i>	3
1.2	Objetivo principal	3
2	Projeto	4
2.1	a)	4
2.2	b)	5
2.3	c)	5
2.4	d)	6
2.5	e)	6
2.6	f)	7
2.7	g)	10
3	Conclusões e trabalho futuro	14
4	Bibliografia	15

Capítulo 1

Introdução

No âmbito da Unidade Curricular Introdução aos Algoritmos e à Programação do Mestrado em Matemática e Computação, foi realizado um trabalho prático sobre *Web Scraping*.

1.1 *Web Scraping*

Web Scraping é o processo de extrair dados de um ou múltiplos *Websites*. Para tal, é necessário realizar um pedido *HTTP* ao servidor do website, descarregar o *HTML* da página e analisar e extrair informação relevante da *raw data* (*parsing*). Este pode ser realizado de forma manual ou automatizada (como será o caso neste trabalho).

Os dados resultantes deste processo encontrar-se-ão num formato mais útil para análise, armazenamento ou visualização. Os dados podem incluir textos, imagens, vídeos, tabelas ou qualquer outro tipo de conteúdo disponível na página *Web*.

1.2 Objetivo principal

O objetivo principal deste projeto consistiu em encontrar um *Website* de um clube desportivo, ou um *Website* que contivesse resultados de eventos desportivos, para seguidamente se extrair dados que considerássemos relevantes, de forma a realizar uma série de análises. Por fim, estas análises deveriam ser colocadas de novo na *Web*.

Capítulo 2

Projeto

Apresenta-se de seguida o enunciado do projeto:

O grande objetivo técnico deste Projeto é ir buscar a um site (neste caso, de um Club Desportivo) valores que nele são expostos (dados) e proceder a uma série de análises para posterior apresentação dos resultados obtidos.

Do ponto de vista técnico, deve programar em Python recorrendo a bibliotecas específicas para o ajudarem a implementar com eficiência algumas das operações que terá de realizar para cumprir o seu objetivo.

Com vista a recolher na Web os dados, a armazená-los na memória do processador, a transformá-los e a publicar os resultados também na Web, deve realizar as seguintes tarefas:

2.1 a)

- Escolher um ou mais sites de clubes desportivos que publiquem resultados sobre uma ou mais modalidades desportivas, a escolher também

O *Website* selecionado por nós foi o HoqueiPatins.pt o qual publica resultados relativos ao hóquei em patins nacional e internacional, bem como notícias e estatísticas, entre outros.

Mais especificamente, optamos pelas informações disponibilizadas relativamente à 1^o Divisão do Campeonato Nacional de Hóquei em Patins 2023/2024 (Fase Regular),

2.2 b)

- **Aceder a esses sites e Capturar os dados em bruto (raw data) para um ficheiro HTML**

Para capturar os dados em bruto deste website, começamos por importar as bibliotecas *requests* e *BeautifulSoup*. Posteriormente, especificamos o *URL* do *Website*, fazemos um pedido (*request*) a esse mesmo *URL* e guardamos a resposta dada pelo servidor num objeto que apelidamos de *r*.

Seguidamente, realizamos o parsing dos dados em bruto, para poder então guarda-los num ficheiro *HTML* que chamamos de *hoquei.html*.

Listing 2.1: Pedido URL; parsing dos dados e criação do ficheiro HTML

```
1 URL = "https://www.hoqueipatins.pt/liga/1-divisao-regular/"
2 r = requests.get(URL)
3 pars = BeautifulSoup(r.content, 'html.parser')
4 with open("hoquei.html", 'w', encoding="utf-8") as arquivo:
5     arquivo.write(pars.decode())
```

2.3 c)

- **Analisar cuidadosamente o material capturado para identificar os itens que podem ser tratados para fornecer resultados interessantes para o público consumidor da informação a produzir e perceber com detalhe a maneira como esses itens estão enroupados nas etiquetas HTML**

Os itens que queríamos tratar eram os resultados e a informação sobre os jogos ainda por realizar. Pois, através destes, poderíamos realizar diversas análises que consideramos relevantes para os amantes de hóquei em patins e não só.

Analizando o ficheiro *HTML* percebemos que essa informação se encontrava enroupada na etiqueta `<div>`: `<div id="resultados">` e que a informação relativa a cada jogo se encontrava dentro de uma etiqueta `<tr>`: `<tr class="container-event-XXXXX" data-live-event="XXXXX" data-match-numeric="000Y" data-sort-data="data-string-comparison="230_000Y" onclick="window.location='/evento/XXXXX'" style="cursor:pointer;">`, em que `XXXXX` é o código atribuído a cada jogo, `Y` corresponde ao número da jornada e `data-sort-data` é também um código único para cada jogo.

Como tal, usamos a função `find()` para encontrar a etiqueta `<div id="resultados">` e guardar a informação num objeto que chamamos de `table`.

Listing 2.2: Exemplo das informações de um jogo no HTML

```
1 <tr class="container-event-48596" data-live-event="48596" data-match-numeric="0001"
2   data-sort-data="D20230930L999990J0001H1800" data-string-comparison="230_0001"
3   onclick="window.location='/evento/48596'" style="cursor:pointer;">
4 <td class="timeGame-textNoBreak">
5 <span class="eventDate">30.09 - 18:00</span>
6 <span class="live-time-48596-live-time"></span>
7 <td class="teamsGame-textNoBreak">
8 <div class="teamsInfos">
```

```

9 <div class="teamHome-textNoBreak">
10     Juventude Pacense <div class="logoTeam">
11 
12 </div>
13 </div>
14 <div>
15 <div class="gameResult-textNoBreak">
16 <span class="line-game-result-main-48596">
17 <span class="resultHome-live-result-48596_0">0</span>
18     <span class="resultAway-live-result-48596_1">5</span>
19 </span>
20 <span class="line-game-result-other-48596" style="display: none;">
21 </span>
22 </div>
23 </div>
24 <div class="teamAway-textNoBreak">
25 <div class="logoTeam">
27 </div> Sporting CP </div>
28 </div>
29 </td>
30 <td>
31 
33 </td>
34 </tr>

```

2.4 d)

- Desenhar um formato na notação JSON onde se possa recolher/armazenar os dados extraídos das capturas em HTML

Esta tarefa acabou por não ser realizada, ou, por outras palavras, acabou por ser encapsulada na alínea seguinte. Pois, ao realizar essa alínea, ao mesmo tempo que varremos o ficheiro *HTML* e realizamos a extração dos dados relevantes, fomos guardando, a cada iteração, os dados num dicionário com a estrutura que nós pretendíamos para o nosso ficheiro *JSON*. Como tal, consideramos não haver necessidade de desenhar-mos um formato *JSON*. De qualquer maneira, este tópico será melhor explicado na secção 2.5.

2.5 e)

- Desenvolver um extrator que varra os ficheiros capturados em HTML e crie um corpus com as extrações armazenadas no tal formato JSON

Para a realização desta alínea, criamos um ciclo for que encontra todas as etiquetas `<tr>` com a *sub-string* "container-event-" através da função `findALL()` e itera para cada uma delas de forma a guardar os dados de cada jogo (o dia, a hora, o nome de cada equipa, o resultado final e os emblemas respetivos) e armazená-los em dicionários individuais que depois são compilados numa lista.

Esta lista foi guardada num ficheiro *JSON* de nome `hoquei.json` através da função `json.dump()`. Logo de seguida, o ficheiro foi aberto através da função `json.load()` e a informação guardada no objeto `corpus` de forma a podermos resolver as alíneas restantes.

Listing 2.3: Extração dos dados relevantes

```

1 resultados = []
2 # ciclo que por cada evento retira os dados q queremos.
3 # usamos uma fun ao lambda para verificar se o atributo class contem a sub string
4 container-event-
5 for row in table.findAll('tr', class_=lambda x: x and 'container-event-' in x):
6     # extrair a data
7     data_span = row.select_one('td.timeGame-span.eventDate')
8     data = data_span.text.strip()
9
10    # extrair o nome das equipas
11    equipa_casa = row.select_one('td.teamsGame-div.teamHome').text.strip()
12    equipa_fora = row.select_one('td.teamsGame-div.teamAway').text.strip()
13
14    # extrair o resultado
15    resultado_casa = row.select_one('span.resultHome').text.strip()
16    resultado_fora = row.select_one('span.resultAway').text.strip()
17    resultado_jogo = f"{resultado_casa}-{resultado_fora}"
18
19    # extrair o emblema equipa da casa
20    emblema_casa = row.select_one('td.teamsGame-div.teamHome-div.logoTeam-img')['src '
21    ] if row.select_one(
22        'td.teamsGame-div.teamHome-div.logoTeam-img') else 'N/A'
23
24    # extrair o emblema equipa de fora
25    emblema_fora = row.select_one('td.teamsGame-div.teamAway-div.logoTeam-img')['src '
26    ] if row.select_one(
27        'td.teamsGame-div.teamAway-div.logoTeam-img') else 'N/A'
28
29    # dicion rio com a informa ao do jogo da itera ao atual
30    info_jogo = {
31        'Data': data,
32        'Equipa-da-Casa': equipa_casa,
33        'Emblema-da-Casa': emblema_casa,
34        'Resultado-do-Jogo': resultado_jogo,
35        'Emblema-de-Fora': emblema_fora,
36        'Equipa-de-Fora': equipa_fora
37    }
38
39    # adicionar o dicionario lista resultados
40    resultados.append(info_jogo)
41
42    # guardar os resultados num ficheiro em formato json
43    with open("hoquei.json", 'w') as w:
44        json.dump(resultados, w, indent=2)
45
46    # ler o json
47    with open("hoquei.json", 'r', encoding='utf-8') as i:
48        corpus = json.load(i)

```

2.6 f)

- Desenvolver várias funções analíticas que percorram o corpus e retirem conclusões interessantes

A primeira função que decidimos criar foi uma função que recebendo o corpus é capaz de retornar a classificação atual do campeonato.

Nesta função, começamos por criar um dicionário vazio `team_points`. Seguidamente, através de um ciclo `for` que itera por cada jogo no `corpus`, são guardados em objetos o nome da equipa da casa, da equipa de fora e o resultado. Caso o resultado seja diferente de "— — —", ou seja, caso o jogo já tenha sido realizado, se o primeiro elemento do resultado for maior que o terceiro, sabemos que a equipa da casa ganhou e são somados 3 pontos ao valor associado à chave dessa equipa no dicionário `team_points`, caso sejam iguais ocorreu um

empate e somamos apenas um ponto e em qualquer outro caso a equipa da casa perdeu e não é somado nenhum ponto.

Depois de fazer este processo para todos os jogos, ordenamos as equipas por ordem decrescente de pontos usando uma função lambda e a função `sorted()`.

A segunda calcula o número de golos marcados, sofridos, a diferença de golos e a média de golos marcados e sofridos por jogo para cada equipa, ordenando as equipas de forma decrescente das respetivas diferenças de golos.

Também iniciamos esta função com a criação de um dicionário vazio chamado `estatisticas_golos` e mais uma vez, por cada jogo no `corpus`, guardamos em 3 objetos o nome da equipa da casa, da equipa de fora e o resultado.

Verificamos se o jogo ocorreu (da mesma forma que na primeira função) e guardamos os golos da equipa da casa e da equipa de fora em objetos. usando a função `strip()` para eliminar quaisquer espaços em branco e a função `split()` para separar as strings pelo caracter `" "`.

Esta informação é depois adicionada ao dicionário `estatisticas_golos`. Seguidamente, através de um ciclo `for` que itera por cada equipa no dicionário, é calculado a diferença de golos subtraindo os golos marcados pelos sofrido. É calculada também a média de golos marcados e golos sofridos por jogo dividindo os golos marcados e sofrido pelo número de jogos.

Por fim, novamente usando a função lambda e a função `sorted()`, as equipas são ordenadas por ordem decrescente da sua diferença de golos.

A terceira função calcula o número de vitórias consecutivas atuais e ordena-as por ordem decrescente.

Mais uma vez, criamos um dicionário vazio chamado `win_streaks` e para cada jogo no `corpus` guardamos num objeto o nome da equipa da casa, da equipa de fora e o resultado.

Posteriormente, se o primeiro dígito do resultado é maior que o terceiro, significa que a equipa da casa ganhou e é somado um 1 ao seu número de vitórias consecutivas. Caso a equipa perca, a sua "win streak" acaba e o número de vitórias volta a ser 0. Este processo é realizado tanto para a equipa da casa como a de fora.

Por último, o dicionário é ordenado por ordem decrescente de vitórias consecutivas atuais.

Nota: Foi também criada uma função bastante semelhante que recebendo o `corpus` e o nome de uma equipa retornava a maior sequência de vitórias de toda a temporada dessa equipa, bem como as informações relativas aos jogos dessa sequência. Porém, para implementar esta função na Web, seria necessário criar um servidor para que o utilizador pudesse escolher a equipa sobre a qual gostaria de ver a sequência, o qual acabamos por não implementar.

A quarta e última função calcula os pontos de cada equipa por jornada.

Listing 2.4: Função que calcula os pontos de cada equipa por jornada

```

1 def calcular_pontos_por_jornada(corpus):
2     pontos_por_jornada = {}
3     jornada1 = 0
4
5     for i, jogo in enumerate(corpus):
6         home_team = jogo['Equipa-da-Casa']
7         away_team = jogo['Equipa-de-Fora']
8         resultado = jogo['Resultado-do-Jogo']
9
10        if resultado == '—':
11            continue
12
13        # a cada 7 jogos come a uma nova jornada
14        if i % 7 == 0:
15            jornada1 += 1
16
17        # Verifica se a equipa da casa venceu, perdeu ou empatou
18        if resultado[0] > resultado[2]:
19            pontos_casa = 3
20            pontos_fora = 0
21        elif resultado[0] < resultado[2]:
22            pontos_casa = 0
23            pontos_fora = 3
24        else:
25            pontos_casa = 1
26            pontos_fora = 1
27
28        jornada = f'Jornada-{jornada1}'
29        if jornada not in pontos_por_jornada:
30            pontos_por_jornada[jornada] = {}
31
32        # Atualiza os pontos para a equipa da casa
33        if home_team not in pontos_por_jornada[jornada]:
34            pontos_por_jornada[jornada][home_team] = 0
35        pontos_por_jornada[jornada][home_team] += pontos_casa
36
37        # Atualiza os pontos para a equipa de fora
38        if away_team not in pontos_por_jornada[jornada]:
39            pontos_por_jornada[jornada][away_team] = 0
40        pontos_por_jornada[jornada][away_team] += pontos_fora
41
42    return pontos_por_jornada
43
44 pontos_por_jornada = calcular_pontos_por_jornada(corpus)

```

Começamos a função criando um dicionário vazio e iniciamos o contador das jornadas. Guardamos em objetos o nome da equipa da casa, da de fora e o resultado e verificamos que o jogo já ocorreu. Como a cada sete jogos começa uma nova jornada, definimos que se a divisão de 7 pelo número da iteração der resto zero então é somado 1 ao número da jornada. Em seguida, verificamos que equipa ganhou ou se empataram e atribuímos os respetivos pontos.

De cada vez que o ciclo itera verifica-se se não houve mudança de jornada e, caso exista, adiciona-se um dicionário com a nova jornada ao dicionário pontos_por_jornada. Posteriormente, verifica-se também se dada equipa já está contida na jornada (teoricamente nunca deve estar pois uma equipa apenas realiza um jogo por jornada) e, caso não esteja, atualizam-se os pontos dessa equipa nessa jornada.

Por fim, são retornados os pontos por jornada de cada equipa. Os quais são utilizados para criar um gráfico que mostra a evolução pontual de cada equipa até ao momento, permitindo perceber as variações de resultados ao longo da época de cada equipa.

Listing 2.5: Função que calcula os pontos de cada equipa por jornada

```

1 # passar os resultados da fun ao anterior para data frame
2 df = pd.DataFrame(pontos_por_jornada).fillna(0)
3
4 # matriz transposta
5 df_transposed = df.T
6
7 # soma cumulativa das colunas
8 df_cumsum_columns = df_transposed.cumsum(axis=0)
9
10 # grafico da evolu o da tabela classificativa
11 df_cumsum_columns.plot(kind='line', marker='o', figsize=(16, 8))
12
13 # plt.title('Evolu o da Tabela Classificativa')
14 plt.xlabel('Jornada')
15 plt.ylabel('Pontos')
16 plt.legend(title='Equipas', bbox_to_anchor=(0, 1), loc='upper-left')
17
18 # guardar o plot como um png
19 plt.savefig('evolucao-tabela-classificativa.png')

```

Para realizar tal gráfico, começamos por transformar os pontos_por_jornada num *dataframe* e transposemos a matriz de forma a poder realizar uma soma cumulativa dos pontos. Tendo depois, usando a biblioteca matplotlib, criado um gráfico de linhas e guardado o gráfico num ficheiro png com a função savefig().

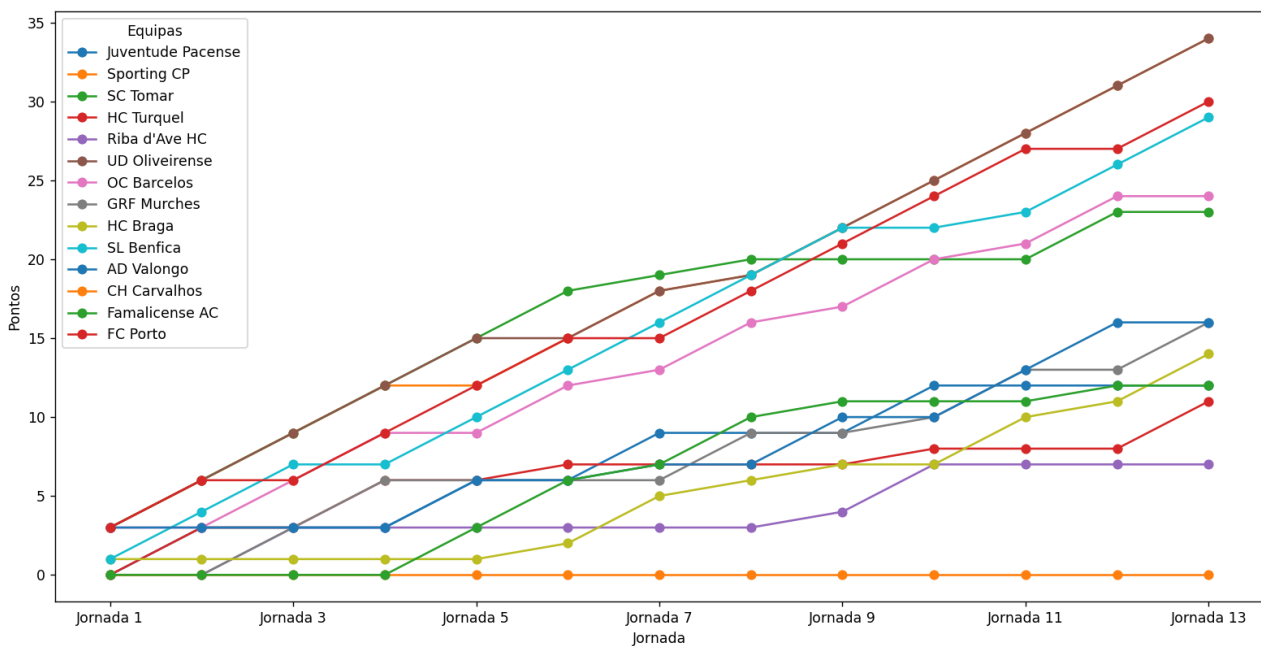


Figura 2.1: Gráfico evolução da tabela classificativa

2.7 g)

- Criar um processador que invoque as referidas funções e apresente na Web os resultados de uma forma gráfica cuidada para que os utilizadores finais possam apreender com eles

Para concretizar esta última tarefa, começamos por transformar todos os dicionários com os resultados e estatísticas em *dataframes*. Convertemos também os *URLs* dos emblemas das equipas, que se encontram

dentro do *dataframe* com os resultados, em etiquetas *HTML* `` para que estes sejam visíveis na página *Web*.

De seguida, criamos a nossa estrutura *HTML*, à qual demos o título de Campeonato Nacional de Hóquei em Patins. O nosso grande objetivo para esta estrutura era permitir ao utilizador escolher o que pretendia visualizar na página e, quando este carregasse numa das opções, apareceriam então os resultados ou estatísticas que ele selecionou, sem que as restantes fossem mostradas de forma a não acumular demasiada informação na página ao mesmo tempo e obrigar o utilizador a "andar para baixo" na página para ver um determinado elemento.

Como tal, criamos uma lista não ordenada com todas nossas secções com resultados e estatísticas, sendo que cada elemento da lista tem uma hiperligação para que, usando a função *javascript* `showSection()`, quando o utilizador selecionar essa opção seja mostrada a respetiva *section* do *HTML*.

Depois, definimos várias etiquetas `<div>`, cada uma delas com os um dos *dataframes* criados, convertidos para *HTML*. Nas etiquetas usamos `class="hidden"` para que as nossas secções estejam "escondidos" na página *Web*.

Por último, criamos uma função em *javascript* que recebe o ID da secção *HTML* que o utilizador pretende abrir, guarda-a num objeto, usando a função `getElementById()`, e, caso a secção exista, mostra-a ao utilizador, sendo que antes verifica que todas as outras continuam "escondidas".

Podemos então fechar o corpo do *HTML* e fechar o próprio *HTML*. E apenas falta escrever o nosso conteúdo *HTML* num ficheiro, o qual apelidamos de Campeonato Nacional de Hóquei em Patins.html.

Campeonato Nacional de Hóquei em Patins

- [Resultados](#)
- [Classificação](#)
- [Evolução da Tabela Classificativa](#)
- [Golos](#)
- [Sequência de Vitórias Atuais](#)

Figura 2.2: Menu de selção da nossa página *Web*

Campeonato Nacional de Hóquei em Patins

- [Resultados](#)
- [Classificação](#)
- [Evolução da Tabela Classificativa](#)
- [Golos](#)
- [Sequência de Vitórias Atuais](#)

Resultados

















Data	Equipa da Casa	Emblema da Casa	Resultado do Jogo	Emblema de Fora	Equipa de Fora
30.09 - 18:00	Juventude Pacense		0-5		Sporting CP
30.09 - 18:00	SC Tomar		6-1		HC Turquel
30.09 - 18:30	Riba d'Ave HC		1-3		UD Oliveirense
30.09 - 21:30	OC Barcelos		11-0		GRF Murches
05.10 - 17:00	HC Braga		2-2		SL Benfica
05.10 - 18:30	AD Valongo		3-1		CH Carvalhos
05.10 - 19:00	Famalicense AC		2-7		FC Porto
07.10 - 16:00	GRF Murches		3-6		Juventude Pacense

Figura 2.3: Parte dos resultados apresentados quando o utilizador seleciona resultados

Classificação

Equipa	Pontos
Sporting CP	34
UD Oliveirense	34
FC Porto	30
SL Benfica	29
OC Barcelos	24
SC Tomar	23
GRF Murches	16
AD Valongo	16
HC Braga	14
Juventude Pacense	12
Famalicense AC	12
HC Turquel	11
Riba d'Ave HC	7
CH Carvalhos	0

Figura 2.4: Tabela classificativa

Golos

Equipa	Jogos	Golos Marcados	Golos Sofridos	Diferença de Golos	Média Golos Marcados por Jogo	Média Golos Sofridos por Jogo
UD Oliveirense	13	63	27	36	4.85	2.08
SL Benfica	13	62	27	35	4.77	2.08
FC Porto	13	64	29	35	4.92	2.23
Sporting CP	13	65	32	33	5.00	2.46
OC Barcelos	13	60	28	32	4.62	2.15
SC Tomar	13	61	44	17	4.69	3.38
AD Valongo	13	49	55	-6	3.77	4.23
HC Braga	13	32	39	-7	2.46	3.00
Famalicense AC	13	37	52	-15	2.85	4.00
GRF Murches	13	48	68	-20	3.69	5.23
Juventude Pacense	13	48	69	-21	3.69	5.31
Riba d'Ave HC	13	32	57	-25	2.46	4.38
HC Turquel	13	35	64	-29	2.69	4.92
CH Carvalhos	13	21	86	-65	1.62	6.62

Figura 2.5: Estatísticas relativas aos golos

Sequência de Vitórias

Equipa	Vitórias consecutivas atuais
Sporting CP	5
UD Oliveirense	5
SL Benfica	2
HC Turquel	1
GRF Murches	1
HC Braga	1
FC Porto	1
Juventude Pacense	0
SC Tomar	0
Riba d'Ave HC	0
OC Barcelos	0
AD Valongo	0
CH Carvalhos	0
Famalicense AC	0

Figura 2.6: Vitórias consecutivas atuais

Capítulo 3

Conclusões e trabalho futuro

O nosso grupo considera ter conseguido resolver todas as tarefas propostas aplicando conteúdos ensinados ao longo do semestre, bem como através de várias pesquisas bibliográficas, que nos permitiram aprofundar e complementar os nossos conhecimentos em Web Scraping, nas aplicações do formato de texto *JSON*, na linguagem *HTML* e em *LaTeX*, entre vários outros.

Relativamente ao trabalho futuro, pensamos que seria interessante desenvolver um servidor que permitisse uma interação do utilizador com a página *Web*, para, por exemplo, permitir seleccionar uma equipa específica e visualizar estatísticas apenas para essa equipa. Para além disso, gostaríamos de ter conseguido colocar os emblemas das equipas em todas as tabelas e no gráfico de forma a ser mais simples a visualização/compreensão dos mesmos e não apenas nos resultados.

Capítulo 4

Bibliografia

- <https://www.hoqueipatins.pt/resultados/>
- <https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/>
- <https://www.geeksforgeeks.org/how-to-scrape-websites-with-beautifulsoup-and-python/>
- <https://www.geeksforgeeks.org/python-map-function/>
- <https://www.freeformatter.com/html-formatter.html>
- <https://www.freecodecamp.org/news/sort-dictionary-by-value-in-python/>
- <https://datatofish.com/line-chart-python-matplotlib/>
- <https://www.geeksforgeeks.org/creating-and-viewing-html-files-with-python/>
- <https://pythonhow.com/python-tutorial/flask/Adding-a-navigation-menu-to-the-website/>