Universidade da Beira Interior
Departamento de Informática

RC3 - 2024: *Arquivo de Notícias do SCC*

Elaborado por:

Rui Soares

Orientador:

Professor Ricardo Campos

8 de julho de 2024

Agradecimentos

Gostaria de expressar os meus sinceros agradecimentos ao Professor Ricardo Campos, pela sua disponibilidade, pelos seus conhecimentos partilhados e por todo o incentivo dado ao longo deste processo. Um agradecimento ao Arquivo.pt pelo seu esforço na preservação de todas as notícias. Sem esta plataforma o desenvolvimento do Projeto não teria sido possível. Por último, agradecer a toda a minha família em amigos, em particular ao Rodrigo Silva, meu colega de curso, que me ajudou muito com os seus conhecimentos e disponibilidade.

Sem esta combinação de fatores o desenvolvimento do Projeto não teria sido possível. Muito obrigado a todos.

Rui,

Resumo

No ano seguinte ao centenário do Sporting Clube da Covilhã, pretendeu-se utilizar os recursos do Arquivo.pt - plataforma de preservação de conteúdos da web portuguesa - para tornar acessível ao público o conjunto de notícias publicadas ao longo da última década nas plataformas digitais dos principais meios de comunicação social da região da Beira Interior - Rádio Cova da Beira, Rádio Caria, Rádio Clube da Covilhã e Notícias da Covilhã -, com foco no principal clube da cidade da Covilhã. Em particular, este projeto visou a disponibilização desses conteúdos através do desenvolvimento de um website dedicado e de um sistema de pesquisa que permita aos utilizadores da plataforma web, recuperar informações acerca do clube a partir dos dados coletados do Arquivo.pt. Após a extração das notícias destes quatro websites, procedemos à implementação de um filtro com o objetivo de selecionar apenas as notícias que dizem respeito ao Sporting Clube da Covilhã. Deste processo resultou a obtenção de 822 notícias: 65 da Rádio Clube da Covilhã, 326 da Rádio Cova da Beira, 120 da Rádio Caria, 311 do Notícias da Covilhã. Estes dados são posteriormente carregados e indexados numa base de dados NoSQL e embebidos no website desenvolvido (em notícias de destaque que aconteceram há um determinado número de anos, e acessíveis através de um sistema de pesquisa).
O Arquivo de Notícias SCC - acessível a partir do link: http://arquivoscc.ipt.pt - tem por base a recuperação de notícias sobre o Sporting Clube da Covilhã desde o início do século com recurso ao Arquivo.pt.
Pretendeu-se com este projeto contribuir para a preservação do património desportivo local e a memória coletiva de uma instituição desportiva histórica da Beira Interior e do País, o Sporting Clube da Covilhã, e também recuperar e preservar informação - e consequentemente, história - de alguns dos principais Orgãos de Comunicação Social da região da Beira Interior.

Palavras-Chave

Arquivo.pt, Sporting Clube da Covilhã, Recuperação de Informação, Preservação de Conteúdos Web

Abstract

In the year following the centenary of *Sporting Clube da Covilhã*, the aim was to use the resources of *Arquivo.pt* - a platform for preserving content from the Portuguese web - to make accessible to the public the set of news published over the past decade on the digital platforms of the main media outlets in the Beira Interior region - *Rádio Cova da Beira*, *Rádio Caria*, *Rádio Clube da Covilhã*, and *Notícias da Covilhã* - focusing on the main club of the city of Covilhã. In particular, this project aimed to make these contents available through the development of a dedicated website and a search system that allows users of the web platform to retrieve information about the club from the data collected from *Arquivo.pt*. After extracting the news from these four websites, we implemented a filter to select only the news related to *Sporting Clube da Covilhã*. This process resulted in the retrieval of 822 news articles: 65 from *Rádio Clube da Covilhã*, 326 from *Rádio Cova da Beira*, 120 from *Rádio Caria*, and 311 from *Notícias da Covilhã*. This data are subsequently loaded and indexed in a NoSQL database and embedded in the developed website (in highlighted news that happened a certain number of years ago, and accessible through a search system).

Arquivo de Notícias SCC - avaliable by the URL: http://arquivoscc.ipt.pt - is based on the retrieval of news about *Sporting Clube da Covilhã* from the beginning of the century using *Arquivo.pt*.

The aim of this project was to contribute to the preservation of local sports heritage and the collective memory of a historic sports institution in the Beira Interior and the country, *Sporting Clube da Covilhã*, and also to recover and preserve information - and consequently, history - from some of the main media outlets in the *Beira Interior* region.

Keywords

Arquivo.pt, *Sporting Clube da Covilhã*, Information Retrieval, Web Content Preservation

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

API Application Programming Interface

BM25 Best Matching 25

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

NC Notícias da Covilhã

RCB Rádio Cova da Beira

RC Rádio Caria

RCC Rádio Clube da Covilhã

SCC Sporting Clube da Covilhã

TF-IDF Term Frequency - Inverse Document Frequency

UBI Universidade da Beira Interior

XML Extensible Markup Language

Capítulo

1

Introdução

1.1 Enquadramento

A comunicação social tem desempenhado um papel crucial no desenvolvimento e divulgação do desporto ao longo dos tempos. Desde os primórdios, a imprensa escrita e a rádio foram os principais meios de comunicação responsáveis por levar os acontecimentos desportivos ao público. Com o avanço tecnológico, a informação desportiva evoluiu significativamente, adaptando-se às novas formas de consumo de notícias, e às exigências de um público cada vez mais desejoso por informações em tempo real.

No final do século XIX e início do século XX, a imprensa escrita emergiu como o principal meio de comunicação social [1]. Jornais e revistas começaram a dedicar secções específicas para o desporto, proporcionando uma cobertura dos eventos desportivos, que permitiam aos leitores acompanharem a evolução dos seus clubes favoritos [2]. Com a invenção do rádio, o desporto ganhou uma nova dimensão ao permitir que os ouvintes acompanhassem eventos em tempo real, criando uma ligação importante entre o público e os acontecimentos desportivos. Posteriormente, a chegada da televisão na segunda metade do século XX possibilitou o consumo visual das notícias desportivas, aumentando exponencialmente a popularidade de várias modalidades ao permitir que os espectadores assistissem aos jogos em qualquer lugar [3].

No entanto, foi a revolução digital que trouxe as mudanças mais profundas e rápidas na forma como a informação desportiva é produzida e consumida. A internet (e, posteriormente, as redes sociais) transformou completamente o panorama mediático [4]. Hoje, os adeptos podem aceder a notícias, vídeos, estatísticas e análises de qualquer lugar do mundo, a qualquer hora,

através de qualquer dispositivo com acesso à internet.

1.2 Motivação

O desenvolvimento de um website com notícias arquivadas sobre o Sporting Clube da Covilhã (SCC) insere-se neste contexto evolutivo da informação desportiva. Ao disponibilizar um arquivo digitalizado de notícias, o projeto não só preserva a história e a memória do clube, mas também facilita o acesso a informações valiosas para os sócios, adeptos, simpatizantes, historiadores e pesquisadores. Através do desenvolvimento desta plataforma, pretende-se contribuir para a valorização do património desportivo e para a perpetuação das histórias desta instituição centenária.

1.3 Objetivos

A crescente quantidade de informações publicadas na web, na forma de textos, imagens, vídeos e áudio leva também à perda de informações ao longo do tempo. Em Portugal, a preservação dos conteúdos web é da responsabilidade do Arquivo.pt[1] [5]. Apoiados nesta plataforma digital, o objetivo deste projeto passa por preservar a memória digital do principal clube da cidade da Covilhã e também o legado dos principais meios de comunicação social da região, tornando o seu conteúdo histórico facilmente acessível ao público em geral.

1.4 Contribuições

Para o desenvolvimento deste projeto, foi desenhado um *pipeline* que obtém informação específica relacionada ao Sporting Clube da Covilhã a partir da infra-estrutura do Arquivo.pt, armazena esses dados num sistema de recuperação de informação, e disponibiliza-os ao utilizador final através do desenvolvimento de um website que é disponibilizado ao público em geral, permitindo assim que os utilizadores tenham acesso a mais de 20 anos de informação comummente perdida. As principais contribuições do projeto desenvolvido são as seguintes:

- A compilação de mais de 20 anos de quatro conjuntos de dados de artigos de notícias referentes ao Sporting Clube da Covilhã;

[1] `https://arquivo.pt/`

- Um sistema de recuperação de informação, permitindo aos utilizadores pesquisar por informações relevantes relacionadas com o passado do SCC;
- A criação de um *pipeline*[2] acessível para o projeto;
- Um website online[3] que consolida toda a informação recolhida e implementa as funcionalidades acima referidas, proporcionando uma interface intuitiva para aceder e explorar a informação recolhida.

1.5 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado por seis capítulos, da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo – **Trabalho Relacionado** – fornece uma visão geral concisa do estado da arte, tecnologias e tendências atuais relevantes para o projeto.
3. O terceiro capítulo – **Arquitetura** – descreve a arquitetura e as tecnologias utilizadas no projeto.
4. O quarto capítulo – **Implementação** – descreve o modo de implementação das tecnologias e todo o processo detalhado por trás do desenvolvimento do projeto.
5. O quinto capítulo – **Demonstração** – apresenta o site desenvolvido recorrendo a descrição escrita e a capturas de ecrã dos vários componentes do site.
6. O sexto capítulo – **Conclusão** – conclui este documento com uma visão geral do projeto.

[2] `https://github.com/ruifilipe16soares/Projeto_Final`
[3] `http://arquivoscc.ipt.pt`

Capítulo

2

Trabalho Relacionado

Ao longo dos tempos, a preservação de conteúdos digitais tornou-se uma área de investigação e desenvolvimento cada vez mais relevante [6]. À medida que a Internet continua a expandir-se e a evoluir, grandes quantidades de informação são criadas e partilhadas [7], mas também lamentavelmente, muitas vezes perdidas [8]. Este fenómeno constitui um desafio significativo para historiadores, investigadores e todos aqueles que necessitam de aceder a essas informações. Este capítulo fala sobre a preservação de arquivos digitais, explora a importância da mesma, especialmente no contexto do património histórico e cultural (e no nosso caso, desportivo), e discute projetos e iniciativas relevantes que contribuíram para avanços nesta área que é a base do desenvolvimento deste projeto. A secção 2.1 destaca a importância dos arquivos web, o seu histórico e as infraestruturas atuais. A secção 2.2 aborda técnicas de Extração de Informação. A secção 2.3 trata das tecnologias de armazenamento de dados, representação de documentação e Recuperação de Informação.

2.1 Arquivos Web

As páginas web podem alterar-se rapidamente ou até desaparecer, resultando na perda de informações valiosas [8]. Arquivar este conteúdo web é crucial para preservar os dados, e também o contexto cultural, histórico e social que lhes estão associados. Esta preservação é fundamental para futuras investigações, permitindo que historiadores, investigadores e o público em geral acedam e estudem informações digitais passadas que, de outra forma, poderiam ser perdidas e não mais recuperadas. [6] Como tal, surgiu a necessidade de criar sistemas que garantam a continuidade e acessibilidade desses dados a

longo prazo.

Plataformas de preservação de conteúdo web

Com o objetivo de começar a dar resposta a esta necessidade surgiu o Internet Archive. [9] Fundado em 1996 por Brewster Kahle, é uma das iniciativas pioneiras e mais significativas na área da preservação de arquivos web. Este projeto visa preservar a história da Internet, criando uma vasta biblioteca digital de websites, livros, vídeos, áudios e outros formatos de media. Utilizando tecnologias de rastreamento e armazenamento, o Internet Archive captura e arquiva regularmente páginas web, permitindo que os utilizadores acedam a versões anteriores de websites através da Wayback Machine[1]. Esta ferramenta não só preserva o conteúdo, mas também oferece uma perceção do estado e evolução de websites ao longo do tempo, sendo uma valiosa fonte de informação para investigadores e historiadores.

Em Portugal, o Arquivo.pt é uma iniciativa que tem por base os mesmos princípios, lançada pelo FCCN (Fundação para a Computação Científica Nacional) em 1996. Esta iniciativa focou-se na recolha, indexação e preservação de conteúdos web portugueses. O Arquivo.pt permite a pesquisa e acesso a páginas web arquivadas desde a década de 90, contribuindo significativamente para a preservação do património digital português [10]. Utiliza tecnologias avançadas - parte delas, herdadas do Internet Archive - para capturar e armazenar uma vasta gama de websites, garantindo que informação relevante sobre a sociedade portuguesa esteja disponível para futuras gerações [11]. Os recursos fornecidos por esta plataforma foram essenciais na recolha de dados e informação que serve de base ao nosso Projeto.

Através da captura regular e sistemática de dados web, estas iniciativas ajudam a proteger a integridade e a continuidade do conhecimento digital. Em última análise, a preservação dos arquivos digitais através de infraestruturas robustas como estas é essencial para assegurar que o património digital global seja conservado e possa continuar a ser utilizado para educar, informar e inspirar futuras gerações [12]. A importância da preservação da web vai além da investigação académica, como foi o caso deste projeto. Este processo assegura que informações vitais, como registos governamentais, artigos de notícias e interações nas redes sociais, permaneçam acessíveis a todos. O conteúdo arquivado pode ser utilizado para rastrear mudanças sociais, verificar factos e compreender a evolução da comunicação digital [13]. À medida que o panorama digital continua a expandir-se, a necessidade de arquivar informações da web torna-se cada vez mais crucial [6].

[1] `http://web.archive.org/`

2.2 Extração de Informação

A extração de informação é um processo fundamental na era digital, permitindo a obtenção de dados estruturados a partir de fontes diversas, como páginas web, ou a partir das próprias plataformas de preservação de páginas web acima descritas. Um dos métodos mais utilizados tendo como fim a extração de informação da web é o web scraping.

O web scraping consiste na utilização de programas automatizados para extrair informações de websites. Esta técnica envolve o envio de solicitações para páginas web e a análise do seu conteúdo HTML para recolher dados específicos, como textos, imagens, links, entre outros. É uma solução eficaz para obter grandes volumes de dados de forma rápida e estruturada, o que seria impraticável manualmente.

No contexto do conteúdo web que diz respeito a notícias e a sites dedicados à partilha de informação noticiosa, o web scraping é particularmente valioso. Permite a coleta de artigos de notícias, títulos, subtítulos, tema da notícia, datas de publicação, autores, e outros metadados de múltiplas fontes, facilitando a criação de bases de dados para análise e pesquisa. Este processo é utilizado por diversos serviços para monitorizar tendência [14], realizar análises de sentimento [15] e rastrear a evolução de temas específicos ao longo do tempo [16].

2.3 Armazenamento de Dados e Recuperação de Informação

A preponderância que representa o conteúdo digital nos dias de hoje leva a que a representação eficaz de documentos e o armazenamento eficiente de arquivos da web nunca foram tão cruciais. Soluções eficientes de preservação da web são essenciais para manter a integridade e a acessibilidade dos dados históricos da web. Igualmente importante é o uso de técnicas avançadas de representação de documentos, como modelos vetoriais, para melhorar a organização, os recursos de pesquisa e a recuperação de conteúdo arquivado.

Quando se trata de armazenamento de dados, as bases de dados NoSQL desempenham um papel importante no cenário atual da ciência de dados [17]. Estas bases de dados fornecem uma estrutura flexível, permitindo o armazenamento de dados semiestruturados e não estruturados. Elas são construídas para serem escaláveis e poderem lidar com grandes volumes de dados, o que as torna uma escolha adequada para sistemas de *big data*, bem como para aplicações web em tempo real.

Uma base de dados NoSQL, ao contrário das tradicionais bases de dados relacionais, não se baseia em tabelas e esquemas rígidos. Existem vários tipos de bases de dados NoSQL, incluindo *document-oriented, key-value, column-family* e *graph databases*. As *document-oriented*, como o MongoDB, armazenam dados em documentos JSON (JavaScript Object Notation) ou BSON (Binary JavaScript Object Notation), permitindo uma estrutura de dados mais dinâmica e flexível. Estas bases de dados são ideais para aplicações que requerem um rápido desenvolvimento e mudanças frequentes na estrutura dos dados. O ElasticSearch é um exemplo de uma base de dados NoSQL amplamente utilizada para pesquisas e análises de grandes volumes de dados, oferecendo funcionalidades avançadas de pesquisa e recuperação de informação.

2.3.1 ElasticSearch

Utilizado amplamente para análise de logs (importantíssima na administração de sistemas), monitorização de performance e como motor de busca em várias aplicações, o Elasticsearch baseia-se em várias técnicas e algoritmos para fornecer resultados de alta relevância e precisão.

2.3.1.1 Estrutura e Índices

O Elasticsearch armazena dados em forma de índices, que são compostos por documentos. Cada documento é uma unidade de dados JSON, e cada índice pode ser visto como uma coleção de documentos semelhantes. A estrutura flexível dos documentos JSON permite armazenar diferentes tipos de dados sem a necessidade de um esquema rígido.

Para classificar a relevância dos documentos durante a busca, o Elasticsearch utiliza algoritmos como Term Frequency - Inverse Document Frequency (TF-IDF) [18], explicado na tabela 2.1 e Best Matching 25 (BM25) [19].

2.3.1.2 TF-IDF

Tabela 2.1: Componentes do algoritmo TF-IDF

Termo	**Definição**	**Fórmula**
Term Frequency (TF)	Mede a frequência de um termo no documento	$\text{TF} = \frac{\text{Nº vezes que o termo aparece no doc}}{\text{Nº total de termos no documento}}$
Inverse Document Frequency (IDF)	Avalia a importância de um termo em relação a todo o conjunto de documentos	$\text{IDF} = \log\left(\frac{\text{Nº total de documentos}}{\text{Nº docs que contêm o termo}}\right)$
TF-IDF Score	Produto de TF e IDF, fornecendo uma medida que destaca termos frequentes num documento, mas raros no conjunto de documentos	TF-IDF = TF × IDF

2.3.1.3 BM25

BM25 é uma evolução do TF-IDF que ajusta a relevância baseada na frequência do termo e no comprimento do documento. A fórmula básica é:

$$\text{score}(q,d) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,d) \cdot (k_1+1)}{f(q_i,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}$$

onde $f(q_i,d)$ é a frequência do termo q_i no documento d, $|d|$ é o comprimento do documento, *avgdl* é a média do comprimento dos documentos, e k_1 e b são parâmetros que ajustam a sensibilidade.

2.3.1.4 Armazenamento e Recuperação de Informação

O Elasticsearch utiliza uma estrutura distribuída para armazenamento e recuperação de dados, dividindo índices em fragmentos chamados *shards*. Cada índice é composto por um ou mais *shards*, que podem ser distribuídos por múltiplos nós do *cluster*. Um *cluster* é um conjunto de nós (servidores) que trabalham juntos para armazenar e pesquisar dados de forma eficiente. Esta arquitetura de *shards* e *clusters* permite escalabilidade horizontal, o que significa que à medida que a quantidade de dados cresce, podemos adicionar mais nós ao cluster para manter o desempenho e a resiliência do sistema. Isso também melhora a tolerância a falhas, uma vez que os shards podem ser replicados em diferentes nós, garantindo que os dados permaneçam acessíveis mesmo se alguns nós falharem. [20]

2.3.1.5 Análise de Texto

Durante a indexação, o Elasticsearch utiliza analisadores para processar o texto dos documentos. Estes analisadores são compostos por uma série de etapas que transformam o texto bruto em termos que podem ser pesquisados. Os componentes principais dos analisadores incluem [21]:

- Tokenização: Esta é a etapa em que o texto é dividido em palavras ou termos individuais. Por exemplo, a frase "Elasticsearch é uma ferramenta poderosa"seria dividida em tokens individuais como "Elasticsearch", "é", "uma", "ferramenta"e "poderosa".
- Filtros de Análise: Após a tokenização, os tokens podem ser passados por diversos filtros de análise. Alguns exemplos incluem:
  - Conversão para minúsculas: Todos os tokens são convertidos para letras minúsculas para que a pesquisa não seja sensível a maiúsculas e minúsculas. Assim, "Ferramenta"e "ferramenta"seriam tratados como o mesmo token.
  - Remoção de *stop words*: Stop words são palavras comuns que geralmente não adicionam significado ao texto (como "e", "a", "o", "de"). Estas palavras são removidas para reduzir o ruído nos dados.
  - Aplicação de *stemmers*: Stemmers são algoritmos que reduzem as palavras às suas raízes ou radicais. Por exemplo, as palavras "correr", "correndo"e "correu"podem ser reduzidas ao radical "corr". Isso ajuda a melhorar a correspondência de termos durante as buscas, independentemente da variação morfológica das palavras.

A combinação destes componentes permite ao Elasticsearch fornecer capacidades avançadas de busca e análise, adequadas para grandes volumes de dados e diversas aplicações.

2.4 Sumário

Neste capítulo, explorámos três áreas cruciais na gestão e preservação de conteúdo digital: preservação de informação web, extração de informação e armazenamento e recuperação de informação.

Destacámos a importância dos arquivos web, como o IA (Internet Archive) e o Arquivo.pt, que preservam conteúdos históricos e culturais, assegurando o acesso a informações arquivadas valiosas. Em seguida, abordámos a extração

de informação através do web scraping, técnica que automatiza a coleta de dados estruturados da web. Por fim, discutimos o armazenamento e recuperação de dados, com ênfase nas bases de dados NoSQL como o Elasticsearch, e no modo que esta ferramenta utiliza uma estrutura distribuída e algoritmos como TF-IDF e BM25 para garantir a relevância e precisão dos resultados de busca, assim como o seu *Modus Operindi* na análise de texto.

A combinação de conhecimentos acerca do desenvolvimento de Plataformas de preservação de arquivos web; do processo de Extração de informações digitais; e do Armazenamento e recuperação dessas informações, forma um ecossistema robusto para a gestão e preservação de dados digitais que são a base deste projeto.

Capítulo

3

Arquitetura

A arquitetura que serviu de base a este projeto é explicada neste capítulo, destacando as tecnologias e ferramentas utilizadas no seu desenvolvimento. De uma forma sumária a arquitetura encontra-se compreendida essencialmente em três módulos principais: (1) Extração de Informação; (2) Recuperação de Informação; (3) Funcionalidades e desenvolvimento de websites. A Secção 3.1 centra-se nos métodos e procedimentos para extrair informação. A secção 3.2 foca-se no processo de recuperação de informação. A Secção 3.3 descreve as tecnologias utilizadas na criação do website do projeto, explicando detalhes das suas funcionalidades. A Secção 3.4 sumaria as principais conclusões sobre a arquitetura do projeto.

3.1 Extração de Informação

Para o propósito de extrair e validar informação, foram desenvolvidos scripts em Python. Inicialmente, estes scripts utilizaram três *packages* principais: o *PublicNewsArchive* [22], *requests*[1], e o *BeautifulSoup4*[2] .

O PublicNewsArchive é uma biblioteca Python que permite aos utilizadores recuperar os URLs antigos de um site, bem como a informação dos artigos de notícias, através da API do Arquivo.pt. Foi utilizado para recuperar os URLs antigos dos quatro sites selecionados: Notícias da Covilhã (NC), Rádio Clube da Covilhã (RCC), Rádio Cova da Beira (RCB), Rádio Caria (RC).

A biblioteca requests permite o envio de pedidos Hypertext Transfer Protocol (HTTP) e Hypertext Transfer Protocol Secure (HTTPS) para páginas web e a recuperação dos seus dados. Com requests, é possível fazer pedidos GET

[1] `https://pypi.org/project/requests/`

[2] `https://www.crummy.com/software/BeautifulSoup/`

para obter dados de um servidor ou pedidos POST para enviar dados para um servidor. A biblioteca simplifica a manipulação de parâmetros de URL, cabeçalhos, cookies e autenticação. Por exemplo, um pedido GET recupera o conteúdo de uma página web e armazena a resposta, permitindo acessar o conteúdo da página.

O BeautifulSoup4 é uma ferramenta valiosa para extrair dados de vários tipos de documentos web, como Hypertext Markup Language (HTML) e Extensible Markup Language (XML). Após obter o conteúdo de uma página web com requests, o BeautifulSoup4 é utilizado para analisar e extrair informações específicas da página. O processo começa com a criação de um objeto BeautifulSoup a partir do conteúdo HTML recuperado. Este objeto permite navegar e manipular a estrutura do documento HTML. Por exemplo, é possível isolar títulos e links, extrair texto de tags HTML e modificar a estrutura HTML do documento. Com BeautifulSoup4, pode-se facilmente encontrar todas as tags de títulos e links na página, extrair o texto contido nas tags de título e obter os URLs dos links.

Requests e BeautifulSoup4 são frequentemente usados em conjunto para operações de web scraping, onde requests recupera o conteúdo da página e BeautifulSoup4 analisa e extrai as informações desejadas, permitindo uma manipulação eficaz de dados web.

3.2 Recuperação de Informação

No contexto do processo de recuperação de informação, o Elasticsearch surgiu como um componente crucial. Neste projeto, o Elasticsearch é utilizado para três funcionalidades principais. Em primeiro lugar, o Elasticsearch é usado para armazenar dados em índices (2.3.1.1), permitindo uma gestão de dados eficiente e organizado, assegurando escalabilidade e resiliência. Em segundo lugar, o algoritmo TF-IDF (2.3.1.2) é utilizado na pesquisa comum, permitindo aos utilizadores pesquisar com eficiência artigos de notícias através da introdução de termos textuais. Além disso, o Elasticsearch utiliza analisadores de texto que processam os documentos durante a indexação, incluindo tokenização, filtros de análise como conversão para minúsculas, remoção de stop words e aplicação de stemmers (2.3.1.5).

Essas funcionalidades do Elasticsearch são fundamentais para garantir uma recuperação de informações eficiente e precisa, proporcionando uma experiência de pesquisa de notícias robusta e facilitando o acesso rápido a grandes volumes de dados, como é o caso das notícias armazenadas.

3.3 Desenvolvimento do Website

Para o desenvolvimento da aplicação web, foram utilizadas várias tecnologias e ferramentas que se complementam para criar uma solução robusta e eficiente. A principal linguagem de programação usada foi Python, em conjunto com a biblioteca Flask [23], que facilitou a criação da secção de pesquisa do site. Flask é um micro framework que permite a construção de aplicações web de forma rápida e simples, oferecendo as funcionalidades necessárias para lidar com pedidos HTTP e responder com dados JSON.

O Elasticsearch foi utilizado como o motor de busca principal da aplicação. Esta ferramenta é reconhecida pela sua capacidade de realizar buscas rápidas e precisas em grandes volumes de dados. As notícias são armazenadas em índices no Elasticsearch, o que permite consultas eficientes e respostas quase em tempo real. Para conectar a aplicação ao Elasticsearch e indexar as notícias armazenadas num ficheiro JSON, foram desenvolvidos scripts em Python que interagem com a API do Elasticsearch.

Além disso, os ficheiros HTML, CSS e JavaScript foram fundamentais para dar vida e estilizar o site, proporcionando uma experiência de utilizador agradável e intuitiva. O JavaScript, em particular, desempenhou um papel crucial nas funcionalidades dinâmicas do site, melhorando a interatividade e a usabilidade da aplicação.

3.3.1 Servidor Web

O Nginx [24] foi a ferramenta escolhida para atuar como servidor web. Este software de código aberto é amplamente utilizado para servir páginas web. No contexto do meu projeto, o Nginx é responsável por servir os ficheiros estáticos da aplicação (HTML, CSS, JavaScript) e garantir que as solicitações dos utilizadores sejam direcionadas corretamente para a aplicação Flask que lida com a pesquisa. A utilização do Nginx melhora significativamente o desempenho e a segurança do site, ao mesmo tempo que facilita a escalabilidade da aplicação.

3.3.2 Virtualização

Para garantir um ambiente de desenvolvimento e produção consistente, foi utilizada a tecnologia Docker. O Docker permite a criação de containers, que são ambientes isolados e leves para executar aplicações. No projeto, foram configurados três containers distintos:

- Elasticsearch: Um container dedicado ao Elasticsearch, garantindo que o motor de busca esteja sempre disponível e configurado corretamente.
- Backend da Aplicação: Um container que executa a aplicação Flask, lidando com os pedidos de pesquisa dos utilizadores. Este container garante que todas as dependências necessárias estejam presentes e configuradas.
- Frontend da Aplicação: Um container que inclui o Nginx para servir os ficheiros estáticos e atuar como intermediário entre o utilizador e a aplicação Flask.

A utilização de Docker permite uma fácil gestão e escalabilidade do ambiente de desenvolvimento, assegurando que a aplicação possa ser executada de forma consistente em diferentes ambientes. Além disso, Docker facilita o processo de deploy, permitindo que novas versões da aplicação sejam implementadas de forma rápida e segura.

3.4 Sumário

A combinação destas tecnologias e ferramentas resulta numa arquitetura eficiente e escalável para o desenvolvimento e operação da aplicação web. O desenvolvimento de scripts em Python e os packages utilizados neles automatizaram a extração de informações da web. A base de dados não relacional Elastic Search permitiu o armazenamento e recuperação eficiente dos dados para posterior pesquisa. Python e Flask fornecem a base para o backend, enquanto o Elasticsearch garante buscas rápidas e precisas. O HTML, o CSS e o JavaScript formam a base do frontend. O Nginx atua como um servidor web robusto e eficiente, e o Docker assegura um ambiente consistente e facilmente gerível para o desenvolvimento e deploy. Juntas, estas ferramentas oferecem uma solução completa para a criação de uma aplicação web moderna e funcional.

Capítulo

4

Implementação

Este capítulo apresenta todo o processo de implementação do projeto, que envolve essencialmente duas grandes etapas: (1) Extração de Informação; (2) Desenvolvimento do Website e Recuperação de informação. A secção 4.1 explica o processo de extração de todos os dados coletados. A secção 4.2 descreve toda a implementação do website, assim como do modelo de Recuperação de Informação, que permite a recuperação dos artigos noticiosos por meio da pesquisa. A secção 4.3 apresenta as conclusões sobre os processos concluídos e os sistemas criados.

4.1 Extração de Informação

O processo de extração de informação dos quatro sites selecionados envolveu essencialmente duas etapas para cada um dos sites. A primeira, a extração dos URLs passados do Arquivo.pt, com recurso ao package PublicNewsArchive [22]. A segunda, a extração dos dados das notícias a partir desses URLs.

4.1.1 Extração de URLs

O Arquivo.pt organiza as páginas iniciais dos sites de forma cronológica, separando-as por anos e meses. Este método permite aos utilizadores navegar facilmente através de snapshots históricos dos websites, facilitando a pesquisa de conteúdo específico numa determinada data.

Para proceder à extração dos URLs passados, foi desenvolvido um script em Python que utiliza o módulo dataExtraction, do PublicNewsArchive, que obtém informação da API do Arquivo.pt.

O código percorre os anos desde o ano do primeiro URL arquivado até 2023, utilizando um ciclo `for`. Para cada ano, o método getPastURLs do módulo dataExtraction é chamado. Este método obtém URLs históricos para o site que colocamos no parâmetro *newspaper-url*, desde janeiro *(startMonth='01')* até dezembro *(endMonth='12')*. Os URLs obtidos são adicionados à lista *all-urls*, e posteriormente escritos, linha a linha, e salvos num ficheiro CSV.

```
import csv
from publicnewsarchive import dataExtraction

# Lista para armazenar todos os URLs
all_urls = []

# Iterar sobre os anos de 2009 a 2023
for year in range(2009, 2023):
    pastURLs = dataExtraction.getPastURLs(year=str(year), newspaper_url=
        'https://noticiasdacovilha.pt/', startMonth='01', endMonth='12')
    # Adiciona os URLs na lista
    all_urls.extend(pastURLs)

csv_filename = 'links_NC_finais.csv'

# Escreve os URLs num .csv
with open(csv_filename, 'w', newline='', encoding='utf-8') as csvfile:
    fieldnames = ['URL']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    # Escreve URL na primeira linha
    writer.writeheader()
    # Escreve os URLs
    for url in all_urls:
        writer.writerow({'URL': url})

print(f'Os links foram salvos em {csv_filename}')
```

Excerto de Código 4.1: Código para extração e armazenamento de URLs

O exemplo acima 4.1 demonstra o procedimento para o site do Notícias da Covilhã (o ano de início da iteração é 2009 pois foi o ano do primeiro URL disponível para este site coletado pelo Arquivo.pt). Este procedimento foi feito para todos os quatro sites de notícias selecionados para a nossa coleta de dados: NC, RCB, RCC, RC.

4.1.2 Extração de Informação a partir das Notícias

O processo de extração de informação a partir das notícias exigiu implementações diferentes para os quatro sites selecionados. Nesta secção serão descritas as características das implementações em comum para os quatro sites.

Inicialmente, os URLs das páginas iniciais desses sites foram armazenados num ficheiro CSV, por cada site. Para obter os links das notícias presentes nas páginas iniciais, foi necessário inspecionar o código HTML da página, parte essencial do processo de web scraping 2.2, identificando por meio deste processo a tag HTML específica do URL que continha os links das notícias. A informação que fornecemos ao script sobre o código HTML é vital para retirar corretamente a informação pretendida.

Este processo foi repetido para cada site, adaptando os scripts sempre que a estrutura base do site mudava. Por exemplo, o site "Rádio Cova da Beira"teve duas estruturas distintas: uma de 2007 a 2012 e outra de 2012 a 2023. Para cada intervalo de anos com estruturas diferentes, foram desenvolvidos scripts específicos para extrair os links das notícias e salvá-los num novo ficheiro CSV. Este processo foi repetido para os quatro sites, onde cada script (um por cada vez que o site mudou a sua estrutura ao longo do tempo) tem por base a seguinte estrutura:

Inicialmente, são importadas as bibliotecas necessárias: `csv`, `requests` para enviar pedidos HTTP e `BeautifulSoup` para análise de documentos HTML.

A função `extract_news_content(url)` envia um pedido `GET` para a URL fornecida e utiliza o `BeautifulSoup` para analisar o conteúdo HTML. Procura todas as tags cujo link da notícia esteja contido nelas - cada estrutura HTML diferente terá os links das notícias contidos em diferentes tags - através do método `find_all` e extrai os links das notícias contidos nesses elementos do código HTML (tags), iterando sobre cada tag através de um ciclo `for`. Se encontrar links, adiciona-os à lista `links`.

O script processa as linhas do ficheiro CSV de entrada - o ficheiro que contém os URLs - entre os índices `linha_inicio` e `linha_fim`. Deste modo são iterados apenas os URLs correspondentes à estrutura HTML que estamos a trabalhar. Como os URLs estão ordenados pela data no CSV, conseguimos perceber qual o último URL correspondente à estrutura HTML anterior, e o primeiro URL correspondente à estrutura HTML seguinte.

O ficheiro de entrada é lido linha por linha, e para cada URL no intervalo especificado, a função `extract_news_content(url)` é chamada. Os links extraídos são armazenados na lista `links` e escritos num novo ficheiro CSV. Abaixo (4.2) segue, como exemplo, o script desenvolvido para retirar os links das notícias do site da Rádio Cova da Beira entre 2007 e 2012:

```
import csv
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
```

```
def extract_news_content(url):
    response = requests.get(url)
    if response.status_code == 200:
        content = response.content.decode('utf-8', errors='ignore')
        site = BeautifulSoup(content, 'html.parser')
        noticia = site.find_all('tr', class_="tabela_titulos")
        if noticia:
            for noticia_item in noticia:
                link = noticia_item.find('a', href=True)
                if link:
                    news_url = urljoin(url, link['href'])
                    links.append(news_url)
        else:
            print(f'Nenhuma noticia encontrada em {url}')
            return None
    else:
        print(f'Falha ao acessar {url}')
        return None

# Nome dos arquivo CSV
csv_filename = 'links_RCB.csv' #ficheiro com todos os URLs do site
csv_filename_2 = 'links_noticias_RCB_07_12.csv'

linha_inicio = 63
linha_fim = 101

# Lista para armazenar os links das noticias
links = []

# Abrir o arquivo CSV e iterar sobre as linhas
with open(csv_filename, 'r', newline='', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for idx, row in enumerate(reader):
        if linha_inicio <= idx + 1 <= linha_fim:
            url = row['URL']
            content = extract_news_content(url)
            if content:
                links.append(content)

# Salvar os dados no arquivo CSV
with open(csv_filename_2, 'w', newline='', encoding='utf-8') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Links'])
    for link in links:
        writer.writerow([link])

print("Links das noticias salvos em", csv_filename_2)
```

Excerto de Código 4.2: Código para extração e armazenamento de links de noticias

O primeiro passo, após ter os links das notícias recolhidos e salvos nos ficheiros CSV, foi perceber se havia notícias repetidas nos ficheiros. Como cada ficheiro é resultado da extração de todas as notícias presentes nas páginas iniciais, as notícias podem ser repetidas em URLs diferentes, isto é, estarem presentes em mais que uma página inicial. Como tal, para cada site foi desenvolvido um script que filtrava os links correspondentes a notícias repetidas. Os detalhes desta implementação para cada site serão explicados abaixo para cada um dos quatro sites.

Após a filtragem dos links, foram criados scripts adicionais para cada ficheiro CSV. Estes scripts extraíram os seguintes dados de cada notícia: título, subtítulo, secção, texto, data, autor, foto e o link original da notícia. Finalmente, os dados extraídos foram salvos em ficheiros JSON. Na base de cada script com esta finalidade, estão presentes os seguintes métodos: Primeiramente, são importadas as bibliotecas necessárias: `csv`, `json`, `requests` para enviar pedidos HTTP e `BeautifulSoup` para análise de documentos HTML.

A função `extract_news_content(url)` envia um pedido `GET` para a URL fornecida e utiliza o `BeautifulSoup` para analisar o conteúdo HTML A função procura e extrai diversos elementos da página: título, secção, subtítulo, texto, data, autor e foto. Para cada elemento encontrado, o texto é extraído e processado para remover espaços em branco desnecessários.

Os dados extraídos são armazenados num dicionário que contém os campos: `titulo`, `section`, `subtitulo`, `texto`, `data`, `autor`, `foto` e o `link`. Este dicionário é então adicionado a uma lista chamada `noticias`. Os dados desta lista serão salvos num ficheiro JSON. Caso um dos elementos referidos não exista na notícia iterada, o campo correspondente ao elemento fica vazio.
Este processo garantiu que todas as notícias fossem capturadas e armazenadas de forma organizada e consistente, independentemente das variações na estrutura dos sites ao longo do tempo.

Por fim, dados os ficheiros JSON gerados, foi desenvolvido um script onde são combinadas todas as notícias presentes nos ficheiros num só ficheiro JSON. Este ficheiro combina assim todas as notícias retiradas do site em questão.

Seguirá uma explicação detalhada do processo para os quatro sites utilizados no processo de extração de dados.

4.1.2.1 Rádio Cova da Beira

Dados os links das notícias recolhidos nos ficheiros CSV, o primeiro passo foi perceber de que forma poderíamos filtrar os links correspondentes a notícias repetidas. Analisando os ficheiros CSV, percebi que cada link terminava com um conjunto de algarismos que identificavam a notícia, e que caso encontrasse links com esse conjunto de algarismos igual ao de outro link já lido, a

notícia era repetida. Posto isto foi desenvolvido o script que tratava os últimos 5 caracteres do link como o seu identificador, e caso fossem encontrados novos links com o mesmo identificador, esses links não eram adicionados ao ficheiro CSV filtrado. Deste modo conseguimos garantir a não repetição de notícias extraídas, o que torna o processo de extração de dados das notícias mais rápido e eficiente, pois o ficheiro filtrado terá uma quantidade menor de links para percorrer. Abaixo é representado um pequeno trecho de um dos ficheiros CSV com links de notícias (4.3) e o script desenvolvido para filtrar as notícias repetidas (4.4).

```
https://arquivo.pt/noFrame/replay/20140605072357/http://www.rcb-
    radiocovadabeira.pt/pag/22469
https://arquivo.pt/noFrame/replay/20140605072357/http://www.rcb-
    radiocovadabeira.pt/pag/22468
https://arquivo.pt/noFrame/replay/20140605072357/http://www.rcb-
    radiocovadabeira.pt/pag/22467
```

Excerto de Código 4.3: Trecho de links de notícias da RCB

```
import csv

csv_filename = 'links_noticias_RCB_12_23.csv'
# Nome do arquivo CSV filtrado
filtered_csv_filename = 'filtered_RCB_12_23.csv'

# Armazenar os identificadores unicos dos links
unique_identifiers = set()
filtered_links = []

# Abrir o arquivo CSV original e iterar sobre as linhas
with open(csv_filename, 'r', newline='', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        url = row['Links']
        # Extrair os ultimos 5 caracteres do link
        identifier = url[-5:]
        # Adicionar o link ao filtro se o identificador for unico
        if identifier not in unique_identifiers:
            unique_identifiers.add(identifier)
            filtered_links.append(url)

# Escrever os links filtrados e unicos no novo arquivo CSV
with open(filtered_csv_filename, 'w', newline='', encoding='utf-8') as
    csvfile:
    writer = csv.writer(csvfile)
```

```
    writer.writerow(['Links'])  # Escreve o cabecalho
    for link in filtered_links:
        writer.writerow([link])
```

Excerto de Código 4.4: Código para filtrar links de notícias repetidas

Filtrados os links, foram desenvolvidos dois novos scripts em Python, um que itera sobre os links filtrados de 2007 a 2012, outro que itera sobre os links filtrados de 2013 a 2023, pois foram estes dois os intervalos de tempo em que a estrutura HTML do site se manteve igual (e de 2012 para 2013 mudou). O objetivo agora era retirar o título, subtítulo, secção, texto, data, autor, foto e link original de cada notícia, estruturando e armazenando os dados num ficheiro JSON, como já explicado anteriormente.

Em todas as notícias com uma dada estrutura encontramos peculiaridades que tornam o processo de fornercer informações do código HTML da página analisada não muito intuitivo.

Falando especificamente do site da RCB, um dos exemplos destas peculiaridades é que várias notícias de 2013 a 2023 continham o autor e a data escritos por extenso numa só tag HTML:

Figura 4.1: Exemplo de notícia com a data e o autor escritos no mesmo elemento

Aplicar corretamente os princípios que compõe o web scraping é também saber contornar peculiaridades como esta tendo em vista o objetivo, que no caso é separar devidamente os campos `data` e `autor`. Esta peculiaridade, associada ao facto de em algumas notícias o campo `data` ter escrito *"há X horas"* ou *"Ontem"* em vez da data propriamente dita, levaram à aplicação de várias condições no código para que os dados fossem corretamente extraídos e separados. A função `extract_news_content(url)` para o script correspondente

às notícias de 2013 a 2023 ficou adaptada às peculiaridades referidas da seguinte forma:

```
def extract_news_content(url):
    response = requests.get(url)
    if response.status_code == 200:
        content = response.content.decode('utf-8', errors='ignore')
        site = BeautifulSoup(content, 'html.parser')

        titulo = site.find('div', class_='titulo')
        section = site.find('div', class_='categoriaData')
        subtitulo = site.find('div', class_='superlead')
        texto = site.find('div', class_='corpo')
        data = site.find('div', class_='data')
        autor = site.find('div', class_='jornalistas')
        foto = site.find('div', class_='foto')
        foto_fin = foto.find('img')['src'] if foto else ""

        titulo_texto = titulo.text.strip() if titulo else ""
        section_texto = section.text.strip() if section else ""
        subtitulo_texto = subtitulo.text.strip() if subtitulo else ""
        texto_texto = texto.text.strip() if texto else ""
        data_texto = ""
        if data:
            if "há" not in data.text and "ontem" not in data.text:
                data_texto = data.text.strip()

        autor_texto = autor.text.strip() if autor else ""
        foto_texto = foto_fin.strip() if foto else ""

        if data_texto != "":
            data_fin = data_texto
            autor_fin = autor_texto
        elif data_texto == "" and "de 20" in autor_texto:
            data_fin = autor_texto[-17:]
            autor_fin = autor_texto[:-21]
        else:
            data_fin = ""
            autor_fin = ""

            # Armazenar os dados no dicionário
        noticia_dict = {
            "titulo": titulo_texto,
            "section": section_texto,
            "subtitulo": subtitulo_texto,
            "texto": texto_texto,
            "data": data_fin,
            "autor": autor_fin[4:],
            "foto": foto_texto,
```

```
            "link": url
        }
        noticias.append(noticia_dict)
    else:
        return None
```

Excerto de Código 4.5: Função para recolher os dados do conteúdo HTML do site da RCB, entre 2013 e 2023

Deste modo, os dados da notícia correspondente à demonstrada acima 4.1 ficaram assim armazenados no ficheiro JSON:

```
    {
        "titulo": "IDANHA-A-NOVA: SUSPEITOS DE 17 FURTOS EM
            ESTABELECIMENTOS",
        "section": "SOCIEDADE",
        "subtitulo": "Os destacamentos da Guarda Nacional Republicana do
             Fundão e de Idanha-a-Nova identificaram na quarta-feira, 20
             de Fevereiro, em nIdanha-a-Nova, um homem e quatro mulheres
            , com idades compreendidas entre os 35 e os 50 anos, por
            suspeita de furtos em estabelecimentos comerciais.",
        "texto": "No decorrer de uma denúncia por tentativa de furto num
             supermercado no concelho de Penamacor, a GNR efectuou uma
            operação policial no sentido de localizar a viatura em fuga,
             acabando uma patrulha por detectar e interceptar o veículo
            que transportava os cinco indivíduos suspeitos. Os suspeitos
             foram constituídos arguidos e sujeitos à medida de coação
            de termo de identidade e residência, sendo os factos
            remetidos ao Tribunal Judicial do Fundão.",
        "data": "22 de Feb de 2019",
        "autor": "Paulo Pinheiro",
        "foto": "https://arquivo.pt/noFrame/replay/20190224183145im_/
            http://www.rcb-radiocovadabeira.pt/media/201902221341-6.jpg"
            ,
        "link": "https://arquivo.pt/noFrame/replay/20190224180510/http
            ://www.rcb-radiocovadabeira.pt/pag/51289"
    },
```

Excerto de Código 4.6: Notícia da RCB armazenada no ficheiro JSON

4.1.2.2 Rádio Caria

O site da RC mudou quatro vezes de estrutura base ao longo do tempo, logo a extração dos links das notícias (4.1.2) originou quatro ficheiros CSV com links de notícias, um por cada estrutura HTML diferente. Ao analisar os ficheiros CSV com os links das notícias recolhidos para a Rádio Caria (4.7) foi possível perceber que dividindo por partes cada link, sendo cada parte separada por cada barra presente no link ("/"), a última parte do link identificava a notí-

cia, e se esta fosse repetida em mais algum link, a notícia referente ao link era repetida. Para evitar a repetição de notícias foi desenvolvido o script 4.8 que dado o ficheiro CSV original cumpre estas condições e devolve o ficheiro filtrado sem notícias repetidas. Este script foi aplicado para os quatro ficheiros CSV referidos acima (mudando apenas os nomes dos ficheiros de entrada e de saída).

```
https://arquivo.pt/noFrame/replay/20190119180512/http://www.radiocaria.
    com/index.php/10-cultura/7339-camara-de-belmonte-entrega-premios-3
https://arquivo.pt/noFrame/replay/20190107180011/http://www.radiocaria.
    com/index.php/9-desporto/7310-ud-belmonte-empata-num-jogo-
    emocionante
https://arquivo.pt/noFrame/replay/20190107180011/http://www.radiocaria.
    com/index.php/9-desporto/7309-cecurde-continua-invicto-no-campeonato
    -distrital
```

Excerto de Código 4.7: Trecho de links de notícias da RC

```
import csv
csv_filename = 'links_noticias_RC_16_19.csv'
filtered_csv_filename = 'filtered_RC_16_19.csv'

unique_identifiers = set()
filtered_links = []

# Função para extrair a parte do link à direita da última barra
def get_link_identifier(url):
    parts = url.split('/')
    return parts[-1]

with open(csv_filename, 'r', newline='', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        url = row['Links']
        identifier = get_link_identifier(url)
        # Adicionar o link ao filtro se o identificador for único
        if identifier not in unique_identifiers:
            unique_identifiers.add(identifier)
            filtered_links.append(url)

with open(filtered_csv_filename, 'w', newline='', encoding='utf-8') as
    csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Links'])  # Escreve o cabeçalho
    for link in filtered_links:
        writer.writerow([link])
```

Excerto de Código 4.8: Código para filtrar links de notícias repetidas

Após este processo, foi desenvolvido um script em Python para cada ficheiro CSV resultante (logo, 4 scripts), dos quais resultaram 4 ficheiros JSON com notícias referentes ao site desta Rádio. Algumas das peculiaridades das notícias deste site incluem o campo `data` conter *"Publicado em"* e conter a hora da publicação, assim como o campo `autor` conter *"Escrito por"*. Para este caso, a função `extract_news_content(url)` para o script correspondente ficou adaptada às peculiaridades referidas da seguinte forma:

```
def extract_news_content(url):
    response = requests.get(url)
    if response.status_code == 200:
        content = response.content.decode('utf-8', errors='ignore')
        site = BeautifulSoup(content, 'html.parser')


        titulo = site.find('h2', class_='art-postheader')
        section = site.find('div', class_='categoriaData') #nao tem
        subtitulo = site.find('div', class_='superlead') #nao tem
        texto = site.find('div', class_='art-article')
        data = site.find('span', class_='art-postdateicon')
        autor = site.find('span', class_='art-postauthoricon')
        foto = site.find('img')['src']

        titulo_texto = titulo.text.strip() if titulo else ""
        section_texto = section.text.strip() if section else ""
        subtitulo_texto = subtitulo.text.strip() if subtitulo else ""
        texto_texto = texto.text.strip() if texto else ""
        data_texto = data.text.strip() if data else ""
        autor_texto = autor.text.strip() if autor else ""
        foto_texto = foto.strip() if foto else ""

        if ":" in data_texto:
            data_texto_fin = data_texto[:-6]
        else:
            data_texto_fin = data_texto


        noticia_dict = {
            "titulo": titulo_texto,
            "section": section_texto,
            "subtitulo": subtitulo_texto,
            "texto": texto_texto,
            "data": data_texto_fin[13:],
            "autor": autor_texto[12:],
            "foto": foto_texto if 'https://arquivo.pt/noFrame/' in
```

```
                foto_texto else "",
            "link": url
        }
        noticias.append(noticia_dict)
    else:
        return None
```

Excerto de Código 4.9: Função para recolher os dados do conteúdo HTML do site da RC, entre 2016 e 2019

4.1.2.3 Rádio Clube da Covilhã

O método para filtrar notícias repetidas do site da RCC é muito semelhante ao utilizado no site da Rádio Caria (4.1.2.2). A única diferença da estrutura dos links das notícias (4.10) recolhidos para este site em relação ao anterior é que todos os links contêm também uma barra ("/") no fim do link. Logo, para filtrar os links das notícias da RCC foi necessária apenas uma pequena alteração na função `get_link_identifier(url)`, de modo a extrair o identificador único da parte final do link (4.11). Especificamente, ela divide a URL pelas barras, remove a barra final se existir, e devolve os dois últimos segmentos do link unidos por uma barra (que correspondem à secção da notícia e ao título). Se a secção e o título, que correspondem ao identificador, da notícia aparecer repetido e mais links, o link corresponde a uma notícia repetida, sendo filtrado da mesma forma que para os sites anteriores.

```
https://arquivo.pt/noFrame/replay/20181125220315/http://radio-covilha.pt
    /2018/11/politica/pcp-questiona-governo/
https://arquivo.pt/noFrame/replay/20180927204048/http://radio-covilha.pt
    /2018/09/desporto/covilha-acolhe-challenge-3000/
https://arquivo.pt/noFrame/replay/20180927204048/http://radio-covilha.pt
    /2018/09/destaque/consolidar-relacoes-com-a-ubi/
```

Excerto de Código 4.10: Trecho de links de notícias da RCC

```
def get_link_identifier(url):
    parts = url.rstrip('/').split('/')
    if len(parts) > 2:
        return parts[-2] + '/' + parts[-1]
    return url
```

Excerto de Código 4.11: Função que torna as duas últimas partes do link no seu identificador

Passando à coleta dos dados das notícias, o site da RCC teve uma estrutura no ano de 2001 e uma nova de 2002 a 2005. De seguida foram 12 anos de inatividade do site, que voltou em 2017 com uma nova estrutura. Como tal, foram desenvolvidos três scripts que resultaram em três ficheiros JSON com notícias deste site.

Na estrutura correspondente ao intervalo de tempo de 2017 a 2023, verificou-se que em algumas notícias o autor era referenciado no mesmo elemento HTML que o próprio texto da notícia. Também se verificou que em certas notícias a data (que foi escrita pela ordem Mês, dia, Ano) aparecia duplicada. Para tal foi desenvolvida uma condição em que se verifica se "Por:"está entre as últimas três palavras do texto. Se sim, o texto é dividido numa lista de palavras e as duas últimas palavras (correspondentes ao nome do autor) são atribuídas ao campo `autor_texto`. Caso contrário, define `autor_texto` com o texto encontrado na variável `autor`.

Foi implementada também uma condição que verifica se `data_texto` contém uma data no formato "Mês dia, ano". Se encontrar uma data nesse formato, extrai a primeira data encontrada e atualiza `data_texto` com a data encontrada.

```
def extract_news_content(url):
    response = requests.get(url)
    if response.status_code == 200:
        content = response.content.decode('utf-8', errors='ignore')
        site = BeautifulSoup(content, 'html.parser')
        noticia = site.find('div', id='primary', class_='content-area')
        if noticia:
                titulo = noticia.find('h1', class_='entry-title')
                section = noticia.find('span', class_='cat-links')
                subtitulo = noticia.find('div', class_='texto') #nao há
                texto = noticia.find('div', class_='entry-content')
                paragrafos = texto.find_all('p')
                texto_texto = ""
                if paragrafos:
                    for p in paragrafos:
                        texto_texto += p.get_text().strip() + " "
                else:
                    paragrafos = ""

                data = noticia.find('span', class_='posted-on')
                autor = noticia.find('a', class_="url fn n")
                foto = noticia.find('img')

                titulo_texto = titulo.text.strip() if titulo else ""
                section_texto = section.text.strip() if section else ""
                subtitulo_texto = subtitulo.text.strip() if subtitulo
                    else ""
```

```
                texto_texto = texto_texto.strip() if texto else ""
                data_texto = data.text.strip() if data else ""
                autor_texto = autor.text.strip() if autor else ""
                foto_texto = foto['src'] if foto else ""
                if texto_texto:
                    if "Por:" in texto_texto.split()[-3:]:
                        palavras = texto_texto.split()
                        autor_texto = ' '.join(palavras[-2:]) if
                            autor_texto else ""
                    else:
                        autor_texto = autor.text.strip() if autor_texto
                            else ""
                else:
                    texto_texto = ""

                match = re.match(r'([A-Za-z]+ \d{1,2}, \d{4})',
                    data_texto)
                if match:
                    data_texto = match.group(1)

                noticia_dict = {
                    "titulo": titulo_texto,
                    "section": section_texto,
                    "subtitulo": subtitulo_texto,
                    "texto": texto_texto,
                    "data": data_texto,
                    "autor": autor_texto,
                    "foto": foto_texto,
                    "link": url
                }
                noticias.append(noticia_dict)
        else:
            return None
    else:
        return None
```

Excerto de Código 4.12: Função para recolher os dados do conteúdo HTML do site da RCC, entre 2017 e 2023

4.1.2.4 Notícias da Covilhã

Com o objetivo de filtrar os links de notícias repetidas dos ficheiros CSV originais, os mesmos foram analisados:

```
https://arquivo.pt/noFrame/replay/20101128160213/http://www.
    noticiasdacovilha.pt/pt/artigos/show/scripts/core.htm?p=artigos&f=
    show&lang=pt&pag=&area=1&idseccao=1&idartigo=981
```

```
https://arquivo.pt/noFrame/replay/20101020150215/http://www.
    noticiasdacovilha.pt/pt/artigos/show/scripts/core.htm?p=artigos&f=
    show&lang=pt&pag=&area=1&idseccao=1&idartigo=919
https://arquivo.pt/noFrame/replay/20101020150215/http://www.
    noticiasdacovilha.pt/pt/artigos/show/scripts/core.htm?p=artigos&f=
    show&lang=pt&pag=&area=1&idseccao=1&idartigo=907
```

Excerto de Código 4.13: Trecho de links de notícias da NC

Olhando para um trecho do ficheiro percebemos que todos os links acabam com um número que é a identificação do artigo, ou seja, da notícia. Se este número fosse repetido em algum link, o link corresponderia a uma notícia repetida. Separando cada link em partes (com "&"a separar), a função `get_link_identifier(url)` define como identificador a última parte do link, que contém o número correspondente à notícia.

```
def get_link_identifier(url):
    if 'idartigo=' in url:
        parts = url.split('&')
        for part in parts:
            if part.startswith('idartigo='):
                return part
    return url
```

Excerto de Código 4.14: Função que torna a última parte do link no seu identificador

Tendo, por fim, os links filtrados, seguiu-se a coleta dos dados das notícias. O site do NC teve uma estrutura base de 2009 a 2019, seguindo-se uma nova entre 2019 e 2023. Como tal foram desenvolvidos dois scripts de forma semelhante aos desenvolvidos para os sites tratados anteriormente que originaram dois ficheiros JSON com os dados de todas as notícias coletadas do site.

```
def extract_news_content(url):
    response = requests.get(url)
    if response.status_code == 200:
        content = response.content.decode('utf-8', errors='ignore')
        site = BeautifulSoup(content, 'html.parser')
        noticia = site.find('div', class_='main col-md-8')
        if noticia:
                titulo = noticia.find('h1', itemprop='headline')
                section = noticia.find('a', class_='term-7') #pode ou
                    nao haver
                subtitulo = noticia.find('div', class_='bk-post-subtitle
                    ') #pode haver ou nao subtitulo
                texto = noticia.find('div', class_='article-content
                    clearfix')
                data = noticia.find('div', class_='post-date')
                autor = noticia.find('div', class_='post-author')
```

```
                f = noticia.find('div', class_='s-feat-img')

                foto_texto = f.find('img')['src'] if f else ""
                titulo_texto = titulo.text.strip() if titulo else ""
                section_texto = section.text.strip() if section else ""
                subtitulo_texto = subtitulo.text.strip() if subtitulo
                    else ""
                texto_texto = texto.text.strip() if texto else ""
                data_texto = data.text.strip() if data else ""
                autor_texto = autor.text.strip() if autor else ""
                if not foto_texto.startswith('http'):
                    foto_texto = ""

                    noticia_dict = {
                        "titulo": titulo_texto,
                        "section": section_texto,
                        "subtitulo": subtitulo_texto,
                        "texto": texto_texto,
                        "data": data_texto,
                        "autor": autor_texto,
                        "foto": foto_texto,
                        "link": url
                    }
                    noticias.append(noticia_dict)
                else:
                    return None
            else:
                return None
        else:
            return None
```

Excerto de Código 4.15: Função para recolher os dados do conteúdo HTML do site do NC, entre 2019 e 2023

4.1.3 Seleção da informação

Tendo todas as notícias devidamente coletadas dos quatro sites que foram utilizados neste processo, foi necessário filtrar os ficheiros JSON com as notícias documentadas pelo tema que serve de base a todo o projeto: o Sporting Clube da Covilhã.

Após combinar os quatro ficheiros JSON (um por cada site tratado no processo) num só, foi desenvolvido um script onde para cada notícia é verificado se pelo menos um dos termos definidos numa lista está presente num dos campos: `titulo`, `subtitulo` ou `texto`. Essa lista de termos identifica o SCC de várias formas possíveis, sejam pelo próprio nome ou por alcunhas, considerando as várias formas possíveis de escrita:

```
termos = [
    'Sp. Covilhã', 'Sporting da Covilhã', 'SC Covilhã', 'SC da Covilhã',
        'S. da Covilhã', 'S. Covilhã', 'leão da serra', 'leões da serra
        ', 'SCC'
]
```

Excerto de Código 4.16: Lista de termos que identificam o SCC numa notícia

Além da verificação dos termos, também é verificado se o campo `texto` da notícia não é repetido em mais nenhuma notícia de modo a eliminar possíveis repetições. Notícias cujo campo `section`, não estando vazio, não seja "Desporto", também são filtradas pois é sabido à partida que não se trata de uma notícia cujo tema inclua os "leões da serra". O código utiliza também o método `lower()` afinal de indiferenciar os termos em letras maiúsculas ou minúsculas.

Deste modo o código recebe como entrada o ficheiro JSON com todas as notícias coletadas e devolve um novo ficheiro JSON apenas com notícias correspondentes ao SCC.

```
import json

json_filename = 'JSON_FINAL.json'
filtered_json_filename = 'filtered_JSON.json'

# Lista de termos para verificação
termos = [
    'Sp. Covilhã', 'Sporting da Covilhã', 'SC Covilhã', 'SC da Covilhã',
        'S. da Covilhã', 'S. Covilhã', 'leão da serra', 'leões da serra
        ', 'SCC'
]
# Função que verifica se pelo menos um termo está presente num dos
    campos
def contains_any_term(text, termos):
    text_lower = text.lower()
    return any(term.lower() in text_lower for term in termos)

# Conjunto para armazenar textos únicos (para verificar duplicidade)
unique_texts = set()
filtered_noticias = []

# Ler o arquivo JSON original
with open(json_filename, 'r', encoding='utf-8') as jsonfile:
    noticias = json.load(jsonfile)
    for noticia in noticias:
        texto = noticia['texto'].strip()
        section = noticia['section'].strip().lower()
        titulo = noticia['titulo'].strip()
```

```
        subtitulo = noticia['subtitulo'].strip()

        if texto in unique_texts:
            continue
        unique_texts.add(texto)
        if "desporto" not in section and section != "":
            continue
        if not (contains_any_term(titulo, termos) or contains_any_term(
            subtitulo, termos) or contains_any_term(texto, termos)):
            continue

        filtered_noticias.append(noticia)

with open(filtered_json_filename, 'w', encoding='utf-8') as jsonfile:
    json.dump(filtered_noticias, jsonfile, ensure_ascii=False, indent=4)
```

Excerto de Código 4.17: Código que filtra as notícias pelo tema SCC

4.1.4 Sumário

Findo o processo de extração de informação, foi obtido o ficheiro JSON final com 822 notícias correspondentes ao Sporting Clube da Covilhã. Nesta secção ficam representados, em forma de tabela (4.1), dados estatísticos referentes a todo o processo de recolha de dados, que espelham em números a coleta de dados para os quatro sites de notícias utilizados como base do processo.

	Nº total Notícias	**Nº total Notícias sobre o SCC**
RCB	6097	326
RC	1887	120
RCC	1466	65
NC	2661	311
TOTAL:	12111	822

Tabela 4.1: Estatísticas do processo da extração de dados

4.2 Desenvolvimento do Website e Recuperação de Informação

Com o arquivo JSON final contendo todas as notícias coletadas, devidamente formatadas e selecionadas de acordo com o tema do nosso projeto, o SCC, chegou o momento de iniciar o desenvolvimento do website. Todo o processo envolveu várias etapas, das quais destacamos as principais: Implemen-

tação do ElasticSearch (4.2.1), desenvolvimento do backend (4.2.2), , desenvolvimento do frontend (4.2.3).

Cada um dos três processos descritos utiliza um container Docker distinto. A utilização do Docker neste projeto foi fundamental devido à sua capacidade de fornecer ambientes de desenvolvimento, teste e produção consistentes. O Docker permite a criação de containers que encapsulam todas as dependências e configurações necessárias para a execução de uma aplicação, garantindo que ela funcione da mesma maneira em qualquer ambiente, seja no desenvolvimento local ou em produção.

No contexto do nosso projeto, utilizamos três containers Docker distintos para gerir as diferentes partes da aplicação: frontend, backend e Elasticsearch. Cada container é responsável por uma parte específica do sistema, garantindo a modularidade e facilitando a manutenção e escalabilidade do projeto.

Para conectar esses containers e permitir a comunicação entre eles, utilizamos um arquivo `docker-compose.yml`. Este arquivo define os três serviços (Elasticsearch, backend e frontend) e conecta-os através de uma rede Docker, cujo nome dado foi `meu_projeto_network`. Isso garante que cada serviço possa comunicar entre si de maneira eficiente e segura. Este arquivo será devidamente explicado e descrito durante o capítulo.

As etapas descritas nem sempre foram seguidas estritamente por esta ordem, pois as mesmas estão diretamente interligadas, pelo que os desenvolvimentos numa das etapas em específico, por vezes implicaram mudanças em outras, de modo a todos os processos implementados resultarem numa aplicação consistente.

4.2.1 Implementação do Elastic Search

A implementação de um mecanismo eficiente de armazenamento e indexação de dados é crucial para o desempenho de sistemas que lidam com grandes volumes de informações, como no caso de notícias arquivadas. O Elasticsearch foi escolhido para esta tarefa devido à sua capacidade de fornecer uma pesquisa rápida e escalável. O Elasticsearch é uma ferramenta poderosa de busca e análise que permite indexar documentos num formato que facilita buscas rápidas e complexas.

Para facilitar a implantação e garantir a portabilidade e replicabilidade do ambiente, utilizamos o Docker para configurar e executar o Elasticsearch. O Docker permite-nos criar containers isolados que contêm todas as dependências necessárias, garantindo que o Elasticsearch funcione de maneira consistente em qualquer ambiente.

Para configurar o serviço Elasticsearch dentro do nosso ambiente Docker, foi necessário definir várias condições dentro do ficheiro `docker-compose.yml`.

```
services:
  elasticsearch:
    image: elasticsearch:8.14.1
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
    ports:
      - "9200:9200"
      - "9300:9300"
    networks:
      - meu_projeto_network
```

Excerto de Código 4.18: Configurações referentes ao Elastic Search, dentro do ficheiro que configura os containers em Docker

Este trecho de código define os serviços que serão executados com vista à correta configuração do Elastic Search no Docker. Na chave *image* é especificada a imagem do Elasticsearch que será utilizada. Já a chave *environment* define variáveis de ambiente, configura o Elasticsearch para funcionar como um nó único, e desativa a segurança para simplificar a configuração, pois só desativando os pacotes associados à segurança foi possível estabelecer a conexão com as portas corretamente. A chave *ports* mapeia as portas do container para as portas do host, permitindo o acesso ao Elasticsearch. Neste exemplo, as portas 9200 e 9300 dentro do conteiner são mapeadas para a porta 9200 e 9300 do host, respetivamente. A chave *networks* define a rede *meu_projeto_network* para facilitar a comunicação entre containers. Os três containers que compõe a aplicação terão esta rede definida.

4.2.1.1 Indexação dos dados

Tendo o Elastic Search configurado e devidamente implementado, o passo seguinte foi indexar e armazenar as notícias, presentes no ficheiro JSON, no Elasticsearch. A natureza NoSQL do Elasticsearch permite a indexação eficiente de grandes volumes de dados textuais. Para tal, foi desenvolvido o script *index_noticias.py* que lê os dados de um arquivo JSON, valida os campos necessários e indexa-os no Elasticsearch.

```
import json
from elasticsearch import Elasticsearch
from datetime import datetime

# Inicializa a conexão com o Elasticsearch
```

```
es = Elasticsearch([{'host': 'localhost', 'port': 9200, 'scheme': 'http'
    }])

with open('data/noticias_normalized.json', 'r', encoding='utf-8') as
    file:
    noticias = json.load(file)

# Função para validar a data
def is_valid_date(date_str):
    try:
        # Verifica se a data pode ser convertida para o formato correto
        if date_str:
            date_format = "%Y-%m-%d" # ajuste conforme necessário
            datetime.strptime(date_str, date_format)
            return True
        else:
            return False
    except ValueError:
        return False

# Define uma data padrão para notícias sem data
default_date = "1970-01-01"

# Indexe os dados no Elasticsearch
for i, noticia in enumerate(noticias):
    if not is_valid_date(noticia.get('data', '')):
        noticia['data'] = default_date
    es.index(index='noticias', id=i, body=noticia)
```

Excerto de Código 4.19: Código referente à indexação das notícias na base de dados NoSQL, Elastic Search

O código inicializa a conexão com o Elasticsearch, em execução no localhost, na porta 9200. É definida uma função para validar se a data de cada notícia está no formato correto (YYYY-MM-DD), pois algumas das notícias recolhidas continham o campo `data` vazio, obrigando a definir esta função e a atribuição de uma data padrão ("1970-01-01") para estes casos. Os dados das notícias do arquivo são lidos. Para cada notícia, é validada a data. Se a data for inválida ou estiver ausente, atribui a data padrão. E por fim, indexa a notícia no Elasticsearch.

4.2.2 Desenvolvimento do Backend

Com o objetivo de implementar uma das principais funcionalidades do site - a secção de pesquisa - foi necessário desenvolver o backend necessário para a secção ficar devidamente configurada.

O backend do projeto foi desenvolvido em Python utilizando o Flask, um microframework leve e flexível que facilita a criação de aplicações web (já explicado em 3.3). A principal função do backend no projeto é fornecer uma interface de Application Programming Interface (API) para realizar pesquisas de notícias no Elasticsearch, permitindo que os utilizadores possam pesquisar e recuperar notícias de forma eficiente.

O primeiro passo para implementar corretamente esta funcionalidade foi desenvolver o seguinte script, que configura um backend simples mas eficaz, utilizando Flask para criar uma API RESTful que interage com o Elasticsearch. Essa API permite realizar buscas nas notícias armazenadas e indexadas no Elasticsearch, oferecendo resultados relevantes baseados na query fornecida pelo utilizador na pesquisa:

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from elasticsearch import Elasticsearch

app = Flask(__name__)
CORS(app)

es = Elasticsearch([{'host': 'elasticsearch', 'port': 9200, 'scheme': '
    http'}])

@app.route('/search', methods=['GET'])
def search():
    #explicação completa mais à frente

    response = es.search(index='noticias', body=search_query)
    hits = response['hits']['hits']
    return jsonify(hits)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Excerto de Código 4.20: Código base que configura a ligação do backend da aplicação com o Elasticsearch

Nó código acima (4.20), a biblioteca `CORS` é importada e utilizada para iniciar a aplicação Flask, o que permite requisições de diferentes origens. É essencial para permitir que o frontend faça requisições ao backend.

De seguida, é inicializada uma conexão com o Elasticsearch, que está configurado para se conectar ao container do Elasticsearch, que está a correr na mesma rede Docker, na porta 9200.

A linha `@app.route('/search', methods=['GET'])` define uma rota para a URL /search, que aceita requisições do tipo GET.

De seguida é definida a função `search()` que aplicará os parâmetros introduzidos pelo utilizador, concatenando-os à variável `search_query`. (Explicação completa da função aqui: 4.21).

Seguidamente é executada a pesquisa no índice `noticias`, usando a *query* construída anteriormente. São extraídos os resultados relevantes (`hits`) da resposta do Elasticsearch e são devolvidos os resultados da pesquisa em formato JSON.

A última linha inicia o servidor Flask, configurando-o para estar disponível em todas as interfaces de rede (0.0.0.0) na porta 5000.

Falando agora especificamente da função `search()`, esta foi implementada para processar as requisições de pesquisa de notícias que são enviadas pela interface de pesquisa do utilizador. Esta interface permite que o utilizador faça a pesquisa de notícias por termos textuais, além de poder filtrar a pesquisa por ano específico e por fonte de notícias (meio de Comunicação Social que as publicou), além de permitir a paginação dos resultados, isto é, caso a pesquisa origine muitos resultados, de modo a não sobrecarregar demasiado a página, a página mostrará apenas 9 resultados de cada vez, com o utilizador a poder carregar mais ou menos resultados, consoante a sua escolha.

```
def search():
    query = request.args.get('query', '')
    year = request.args.get('year', '')
    source = request.args.get('source', '')
    page = int(request.args.get('page', 1))
    size = int(request.args.get('size', 9))

    search_query = {
        "query": {
            "bool": {
                "must": [],
                "filter": []
            }
        },
        "from": (page - 1) * size,
        "size": size
    }
    if query:
        search_query["query"]["bool"]["must"].append({
            "multi_match": {
                "query": query,
                "fields": ["titulo", "subtitulo", "texto", "autor"]
            }
        })
    if year:
        search_query["query"]["bool"]["filter"].append({
```

```
            "range": {
                "data": {
                    "gte": f"{year}-01-01",
                    "lte": f"{year}-12-31"
                }
            }
        })
    if source:
        source_conditions = {
            "Rádio Cova da Beira": "rcb-radiocovadabeira.pt",
            "Rádio Caria": ["radiocaria.com", "radiocaria.pt"],
            "Rádio Clube da Covilhã": "radio-covilha.pt",
            "Notícias da Covilhã": "noticiasdacovilha.pt"
        }
        if source in source_conditions:
            condition = source_conditions[source]
            if isinstance(condition, list):
                search_query["query"]["bool"]["filter"].append({
                    "bool": {
                        "should": [{"match": {"link": c}} for c in
                            condition],
                        "minimum_should_match": 1
                    }
                })
            else:
                search_query["query"]["bool"]["filter"].append({"match":
                     {"link": condition}})
```

Excerto de Código 4.21: Função que define os parâmetros pelos quais o utilizador efetua a pesquisa

Explicando detalhadamente a função, esta começa por receber os parâmetros de pesquisa da URL. Estes parâmetros são fornecidos pelo utilizador através da interface de pesquisa, como foi já explicado.

A variável `query` é definida como a string de busca que o utilizador deseja procurar nas notícias, `year` é o ano específico para filtrar as notícias, e `source` a fonte de notícias específica para filtrar. `page` permite a paginação e `size` define número de resultados por página (9).

De seguida a query é construída utilizando a estrutura `bool` do Elasticsearch, que permite combinar várias condições de pesquisa, que no caso são separadas como `must` (condições que devem ser satisfeitas para um documento ser incluído nos resultados) e `filter` (condições de filtro, que limitam os resultados). A cláusula `from` calcula o deslocamento para a paginação, baseado no número da página e no tamanho da página.

Se a string de pesquisa (`query`) for fornecida, ela é adicionada à cláusula `must` da query. A condição `multi_match` busca a string fornecida nos campos

`titulo, subtitulo, texto` e `autor` das notícias (deste modo o utilizador pode também pesquisar pelo nome do autor da notícia, se assim o entender).

Se o ano (`year`) for fornecido, ele é adicionado à cláusula `filter` da query. A condição `range` filtra as notícias que estão dentro do intervalo do ano fornecido, desde o início (`gte`) até o final (`lte`) do ano.

Da mesma forma, se a fonte de notícias (`source`) for fornecida, ela é adicionada à cláusula `filter` da query. A função verifica se a fonte de notícias fornecida está nas condições pré-definidas (`source_conditions`). Estas condições foram definidas para encontrar uma correspondência entre a fonte de notícias e o link que lhe corresponde. É pelo link da notícia que é possível estabelecer uma correspondência com a `source` inserida pelo utilizador. Como são quatro fontes de notícias possíveis, são definidas quatro condições que associam cada opção de escolha de fonte de notícias com uma parte do link da notícia. Deste modo é possível que o utilizador filtre os resultados pelo meio de Comunicação Social que publicou a notícia.

Virtualização do Backend

Tal como para o Elasticsearch (4.2.1), para facilitar a implantação e garantir a portabilidade e replicabilidade do ambiente, utilizamos o Docker para configurar e executar o Backend. Foi criado o ficheiro `Dockerfile.backend` para o backend que realiza as seguintes etapas:

- Utilização de uma Imagem Base do Python, garantindo que todas as dependências do Python estejam disponíveis.
- Definição da diretoria dentro do container, onde todos os arquivos da aplicação serão armazenados.
- O arquivo `requirements.txt`, que contém todas as dependências do projeto (Flask, Elasticsearch), é copiado para o container e as dependências são instaladas.
- Cópia do Restante do Código da Aplicação para o container.
- Exposição da Porta 5000 para permitir o acesso à aplicação Flask.
- O comando `python app.py` é definido para iniciar a aplicação quando o container for executado.

Por fim são definidas as condições dentro do ficheiro `docker-compose.yml` para configurar o backend devidamente:

```
services:
  backend:
    build:
      context: .
      dockerfile: Dockerfile.backend
    ports:
      - "5002:5000"
    networks:
      - meu_projeto_network
    depends_on:
      - elasticsearch
```

Excerto de Código 4.22: Configurações referentes ao Backend, dentro do ficheiro que configura os containers em Docker

Este trecho de código define os serviços que serão executados com vista à correta implementação das funcionalidades do backend no Docker. Na chave *build* são dadas as instruções (diretoria, e nome do `Dockerfile`) para construir a imagem Docker do backend. Já a chave *ports* mapeia a porta 5000 do container para a porta 5002 do host, permitindo acesso ao serviço backend. A chave *networks* define a rede *meu_projeto_network* para facilitar a comunicação entre conteiners. Os três containers que compõe a aplicação terão esta rede definida. Por fim, a chave *depends_on* define a dependência do serviço backend em relação ao serviço Elasticsearch.

4.2.3 Desenvolvimento do Frontend

O desenvolvimento do frontend do website envolve a combinação de HTML, CSS e JavaScript, cada um desempenhando um papel específico e essencial.

O HTML define a estrutura do conteúdo da página, organizando texto, imagens, links e outros elementos através de tags. Ele cria uma hierarquia de informações que facilita a navegação e a leitura.

O CSS controla a apresentação visual do site, incluindo layout, cores, fontes e espaçamentos. Ele transforma documentos HTML simples em interfaces atraentes e profissionais.

O JavaScript adiciona interatividade e dinamismo ao site. Ele responde a eventos do utilizador, valida formulários e realiza chamadas assíncronas ao servidor, criando interfaces mais interativas e melhorando a experiência do utilizador.

Neste projeto, cada página do site contém um arquivo HTML para estrutura, um arquivo CSS para estilização e um arquivo JavaScript para funcionalidades dinâmicas, proporcionando uma experiência ao utilizador mais rica e envolvente.

O website é composto essencialmente por quatro páginas: Página Inicial (4.2.3.1), Página de Notícias (4.2.3.2), onde é estruturada cada notícia; Página dos Resultados da Pesquisa (4.2.3.3), com as notícias resultantes da pesquisa do utilizador; e a Página das Motivações (4.2.3.4), onde estão descritas as motivações inerentes ao desenvolvimento do site.

Cada uma destas páginas tem por base um ficheiro HTML, um ficheiro Javascript e um ficheiro CSS. Como tal, a descrição do procedimento terá como estrutura quatro tópicos, um por cada página, e por cada um deles uma explicação do ficheiro HTML, CSS e JavaScript correspondente.

4.2.3.1 Página inicial

O código HTML define a estrutura e o conteúdo da página inicial do site. Ele inclui links para um ficheiro com estilos CSS, fontes de texto do Google, ícone do site, scripts como jQuery (biblioteca JavaScript rápida e concisa que simplifica a manipulação de documentos HTML) e um ficheiro JavaScript personalizado. O corpo contém um cabeçalho com o logotipo que redireciona para a página inicial e links de navegação para as secções "Motivações"e "Pesquisa". A sceção de introdução exibe uma imagem e um texto sobre o site, com uma subsecção adicional oculta inicialmente. A secção "On This Day"apresenta notícias relevantes para o dia, com navegação entre diferentes conjuntos de notícias. A secção de pesquisa permite que os utilizadores filtrem notícias por texto, ano e fonte, exibindo os resultados dinamicamente. O rodapé fornece informações sobre o projeto, links para os sites dos órgãos de comunicação social e imagens das instituições colaboradoras (UBI e Arquivo.pt), com links para seus respectivos sites.

O ficheiro JavaScript correspondente realiza diversas funcionalidades essenciais. Após o carregamento do documento, ele define duas funções: uma para verificar se uma notícia ocorreu "neste dia"ou em datas próximas, e outra para obter uma descrição relativa do tempo desde a data da notícia. Estas funções são essenciais para o bom funcionamento da secção "Neste dia..."(5.2.1). Em seguida, há funções para navegação entre secções da página e para redirecionar cliques no logotipo para a página inicial, e em links específicos.

O código faz *fetch* das notícias a partir do ficheiro JSON (permitindo que o navegador solicite e receba dados sem recarregar a página), filtrando-as para mostrar apenas aquelas que ocorreram em datas próximas à data atual. Essas notícias são exibidas dinamicamente na secção "Neste dia...". Há também navegação com setas para ver mais notícias.

Além disso, o código adiciona um evento de envio do formulário de pesquisa, que redireciona o utilizador para a página de resultados da pesquisa com os parâmetros de pesquisa apropriados. Este evento é comum a todos os

ficheiros JavaScript implementados para cada página, pois a secção de pesquisa é comum a todas as páginas tratadas.

```
function isOnThisDay(newsDate, currentDate) {
    const today = new Date(currentDate.getFullYear(), currentDate.
        getMonth(), currentDate.getDate());
    const news = new Date(today.getFullYear(), newsDate.getMonth(),
        newsDate.getDate());
    const diffTime = today.getTime() - news.getTime();
    const diffDays = diffTime / (1000 * 3600 * 24);

    return diffDays === 0 || diffDays === 1 || diffDays === 2 ||
        diffDays === -1 || diffDays === -2;
}

function getRelativeTime(newsDate, currentDate) {
    const yearsDiff = currentDate.getFullYear() - newsDate.
        getFullYear();
    const today = new Date(currentDate.getFullYear(), currentDate.
        getMonth(), currentDate.getDate());
    const news = new Date(today.getFullYear(), newsDate.getMonth(),
        newsDate.getDate());
    const diffTime = today.getTime() - news.getTime();
    const diffDays = diffTime / (1000 * 3600 * 24);

    if (diffDays === 0) {
        return `HOJE, HÁ ${yearsDiff} ANOS...`;
    } else if (diffDays === 1) {
        return `ONTEM, HÁ ${yearsDiff} ANOS...`;
    } else if (diffDays === 2) {
        return `ANTEONTEM, HÁ ${yearsDiff} ANOS...`;
    } else if (diffDays === -1) {
        return `AMANHÃ, FAZ ${yearsDiff} ANOS...`;
    } else if (diffDays === -2) {
        return `DEPOIS DE AMANHÃ, FAZ ${yearsDiff} ANOS...`;
    }
    return '';
}
```

Excerto de Código 4.23: Trecho de código referente à implementação da secção dinâmica "Neste dia..."

4.2.3.2 Página de Notícias

O código HTML tem uma estrutura semelhante à do código HTML para a página inicial (4.2.3.1, pois todas as páginas são coerentes em termos de estrutura base, de modo à aplicação final ser concisa e uniforme. Como tal, código HTML referente ao cabeçalho, à secção de pesquisa e ao rodapé das páginas é

semelhante a todas as páginas tratadas, pois em todas, os elementos referidos serão exibidos ao utilizador.

Neste código é definido o elemento principal: `<main id="news-content»`, onde o conteúdo da notícia será carregado dinamicamente, através das funções definidas no ficheiro Javascript correspondente. O código deste ficheiro recupera o título da notícia a partir do link, busca as notícias no arquivo JSON, e encontra a notícia correspondente ao título. Se a notícia é encontrada, insere todo o seu conteúdo no elemento com ID "news-content", definido no código HTML, que utiliza o ficheiro CSS adequado de modo a exibir corretamente a notícia ao utilizador. Por fim, é definida a função que adiciona um evento de envio do formulário de pesquisa, que redireciona o utilizador para a página de resultados da pesquisa com os parâmetros de pesquisa apropriados.

```
document.getElementById('search-form').addEventListener('submit',
    function(event) {
    event.preventDefault();
    const query = document.getElementById('search-query').value;
    const year = document.getElementById('search-year').value;
    const source = document.getElementById('search-source').value;
    const searchParams = new URLSearchParams({ query, year, source
        }).toString();
    window.location.href = `search.html?${searchParams}`;
});
```

Excerto de Código 4.24: Função que redireciona o utilizador para a página que contém os resultados da pesquisa que efetuou

4.2.3.3 Página com Resultados da Pesquisa

O código HTML, além dos elementos em comum já referidos (4.2.3.2), tem definido um elemento `section`, que exibirá os resultados da pesquisa do utilizador na página. Além do título da secção "Resultados da pesquisa", um parágrafo mostra uma mensagem enquanto a pesquisa está em processamento, antes de exibir dinamicamente os resultados da pesquisa. O código tem ainda definido o botão que permite carregar mais resultados, quando clicado, caso existam mais resultados inerentes à pesquisa por exibir.

De seguida foi implementado o código Javascript correspondente, que:

- Obtem os parâmetros da URL para extrair os valores de `query`, `year` e `source`, armazenando-os em variáveis. Define a página atual da pesquisa e o número de resultados por página. Exibe a mensagem de processamento.

- A função `fetchResults(page)` realiza uma requisição GET para o backend (pela porta 5002, que é a porta do host do backend), passando `query`, `year`, `source`, `page` e `size` como parâmetros. Quando os dados são recebidos, a mensagem de processamento é oculta. Se não houver resultados na primeira página, uma mensagem "Nenhum resultado encontrado."é exibida.

- Para cada resultado, cria-se um elemento `div` com cada notícia que resultou da pesquisa, com a sua data, título, um trecho do texto e um botão "Continuar a Ler"que redireciona para a página da notícia completa.

- Se o número de resultados recebidos for igual ou maior ao `resultsPerPage`, que foi definido como 9, o botão "Carregar mais"é exibido; caso contrário, é ocultado.

- A função `fetchResults(currentPage)` é chamada inicialmente para carregar a primeira página de resultados. Em caso de clique no botão "Carregar mais", é incrementada a página atual e chamada a função `fetchResults(currentPage)` novamente para carregar mais resultados.

```
$(document).ready(function() {
    const urlParams = new URLSearchParams(window.location.search);
    const query = urlParams.get('query');
    const year = urlParams.get('year');
    const source = urlParams.get('source');
    let currentPage = 1;
    const resultsPerPage = 9;
    // Exibir mensagem de carregamento
    $('#loading-message').show();
    function fetchResults(page) {
        $.get('http://localhost:5002/search', { query: query, year: year
            , source: source, page: page, size: resultsPerPage },
            function(data) {
            $('#loading-message').hide(); // Ocultar mensagem de
                carregamento
            if (page === 1) {
                $('#results-container').empty();
            }
            if (data.length === 0 && page === 1) {
                $('#results-container').append('<p>Nenhum resultado
                    encontrado.</p>');
            } else {
                data.forEach(function(hit) {
                    const newsDate = new Date(hit._source.data);
```

```
                    const formattedDate = newsDate.toLocaleDateString('
                        pt-PT', {
                        day: '2-digit',
                        month: '2-digit',
                        year: 'numeric'
                    });
                    const resultHtml = `
                        <div class="result-item">
                            <p class="news-time" style="color: green;">$
                                {formattedDate}</p>
                            <h2><a href="noticia.html?title=${
                                encodeURIComponent(hit._source.titulo)}"
                                >${hit._source.titulo}</a></h2>
                            <p>${hit._source.texto.substring(0, 100)
                                }...</p>
                            <button onclick="window.location.href='
                                noticia.html?title=${encodeURIComponent(
                                hit._source.titulo)}'">Continuar a Ler</
                                button>
                        </div>
                    `;
                    $('#results-container').append(resultHtml);
                });
                if (data.length === resultsPerPage) {
                    $('#load-more').show();
                } else {
                    $('#load-more').hide();
                }
            }
        });
    }
    fetchResults(currentPage);
    $('#load-more').on('click', function() {
        currentPage++;
        fetchResults(currentPage);
    });
});
```

Excerto de Código 4.25: Código do ficheiro Javascript referente à página com os resultados da pesquisa

4.2.3.4 Página das Motivações

O código HTML referente a esta página define, além dos elementos e secções em comum a todas as páginas, a secção principal da página "Motivações"do site. Dentro desta secção, há um título "Motivações"seguido de um parágrafo introdutório sobre o desenvolvimento do projeto. Logo abaixo, um `iframe` é utilizado para incorporar o documento PDF referente ao enunciado que ex-

plica a proposta do projeto. Adicionalmente, o texto descreve a finalidade do "Arquivo de Notícias SCC", explicando sucintamente como foram extraídas e filtradas as notícias de quatro órgãos de comunicação social da região da Beira Interior, focando no Sporting Clube da Covilhã (SCC). O código finaliza com referência aos autores do projeto e destacando a importância de preservar e honrar a história da instituição SCC e seus associados.

4.2.4 Virtualização do Frontend

A virtualização do Frontend foi, à semelhança das duas estruturas já definidas, implementada através do Docker. No caso do frontend do projeto, utilizamos Docker para empacotar a aplicação num container que utiliza Nginx [24], um servidor web leve e de alto desempenho (3.3.1). Para tal, foi definido o ficheiro Dockerfile:

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
EXPOSE 80
```

Excerto de Código 4.26: Dockerfile referente ao frontend

O ficheiro especifica a imagem base a ser utilizada (`FROM nginx:alpine.`). Esta imagem é uma versão compacta e otimizada do Nginx. Em seguida, os arquivos da aplicação frontend são copiados para a diretoria padrão do Nginx dentro do container. A linha `EXPOSE 80` indica que o container irá expôr a porta 80, que é a porta padrão para o tráfego HTTP. Quando o container é iniciado, o Nginx será executado automaticamente, servindo os arquivos da aplicação.

Por fim, é configurado o serviço do frontend no ficheiro `docker-compose.yml`:

```
services:
  frontend:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    networks:
      - meu_projeto_network
```

Excerto de Código 4.27: Trecho do ficheiro docker-compose, referente ao frontend

Aqui é definido que a construção do container será feita a partir do contexto atual e que será utilizado o Dockerfile mencionado (4.26). A chave `ports` mapeia a porta 80 do container para a porta 8080 do host, permitindo o acesso ao frontend do site através desta porta. Finalmente, o serviço frontend é conectado à rede `meu_projeto_network`, garantindo que ele possa comunicar

eficientemente com os outros serviços, o Backend e o Elasticsearch, dentro do mesmo ambiente de rede.

Virtualização conjunta dos três ambientes

Por fim, com os três serviços definidos no ficheiro `docker-compose.yml`, é utilizado o comando DOCKER-COMPOSE UP , que faz com que o Docker Compose leia as definições de serviço no ficheiro `docker-compose.yml` e inicie todos os containers conforme especificado. O frontend do site é servido pelo Nginx, mapeando a porta 80 do container para a porta 8080 do host, como explicado anteriormente. Portanto, após ter os três containers iniciados e em execução, podemos aceder, através de um navegador, a *localhost:8080*, onde o website estará disponível com todas as funcionalidades devidamente implementadas.

4.3 Sumário

Neste capítulo, é descrita a implementação do sistema abrangente de recuperação de artigos noticiosos, através de um website acessível e de fácil utilização. O processo de extração de informações começou com a coleta de artigos de notícias por meio do web scraping, desenvolvendo scripts Python, utilizando pacotes Python como `BeautifulSoup`, `requests` e `PublicNewsArchive`, essenciais ao processo.

Para o processo de recuperação de informações, o Elasticsearch, uma Base de dados NoSQL, foi responsável pela indexação e recuperação eficiente e escalonável dos dados das notícias. O backend, desenvolvido em Python com Flask, fornece uma API para realizar pesquisas no Elasticsearch. O frontend, criado com HTML, CSS e JavaScript, estrutura e estiliza o website. Docker foi usado para containerizar cada componente, fornecendo um ambiente consistente e fácil de implementar, e garantindo que o website pode ser acedido em qualquer sitema operativo.

Capítulo

5

Demonstração do website

O design é crucial no desenvolvimento de uma página web, influenciando a funcionalidade. Navegação intuitiva, layouts claros e design responsivo otimizam a interação dos utilizadores com o website, garantindo que o design seja atraente, funcional e adequado ao tema do website. Este capítulo apresenta uma visão geral da interface do site, os principais componentes e as respectivas funcionalidades. A Secção 5.1 fornece uma visão geral introdutória da estrutura e do design do site. A Secção 5.2 exibe os componentes da página inicial e todos os recursos dentro dela. A secção 5.3 explica e expõe a página que contém os resultados da pesquisa do utilizador. A secção 5.4 demonstra como é estrurada a página referente a cada artigo noticioso. A secção 5.5 dá a conhecer a página onde estão descritas as motivações que levaram ao desenvolvimento do website. Por fim, a Secção 5.6 apresenta as conclusões retiradas da análise da estrutura e design do website.

5.1 Estrutura e Design

A estrutura base do site contém elementos estruturais e design que são comuns a todas as páginas que o constituem. Um desses elementos é a imagem de fundo da página, que não é nada mais que a página noticiosa retirada de página da revista Stadium, número 300, de 1 de setembro de 1948, que serviu de base para o mural comemorativo do centenário do SCC, presente na cidade da Covilhã, junto ao elevador/escadas de S. André, na rua Marquês D'Ávila e Bolama. Sendo este projeto um site de notícias arquivadas do SCC que serve como homenagem ao centenário da instituição, nada como retirar inspiração de uma das grandes homenagens ao mesmo acontecimento na cidade.

OS "LEÕES" DA SERRA
NA PRIMEIRA DIVISÃO

I — Taças recebidas pelo Sporting da Covilhã por virtude do Campeonato da Segunda Divisão — Monumental «O Século», Federação, (Campeonato Distrital), Sportinguistas do Refugio, Grupo Desportivo Covilhanense, Sportinguistas do Serrado, Circo Alegria, e Um admirador. — 2 Após o desafio que difinitivamente conferia o ambicionado título de Campeão da Segunda Divisão, a multidão, entusiasmada, desfila em cortejo, na Praça do Município, na Covilhã, dando largas ao seu entusiasmo e satisfação

O Sporting Clube da Covilhã, filial do Sporting Clube de Portugal, instituição que honra uma cidade, bela, hospitaleira e industrial, mesmo toda uma região privilegiada da Natureza, ascendeu por direito próprio, à custa de trabalho e sacrifícios, demonstrando uma fé inquebrantável e tenacidade invulgar, à Primeira Divisão do Campeonato Nacional de Futebol, após ter ganho com brilhantismo, uma prova dura e difícil, o Campeonato da Segunda Divisão.

Não são apenas os associados do Clube que estão contentes, mas todos os covilhanenses que amam verdadeiramente a sua terra — e que vêem no seu representante o melhor cartaz de propaganda da Covilhã.

«Stadium» saúda calorosamente os briosos Leões da Serra.

No encontro Leixões-Covilhã, que, por assim dizer, iniciou uma carreira brilhante, o avançado-centro do Sporting da Covilhã, em magnífico estilo, joga a bola de cabeça

Depois do triunfo — a cidade da Covilhã vive as suas melhores horas!

No encontro Sporting-União de Coimbra, Covilhã marca mais um golo e o entusiasmo é enorme. O publico vibra intensamente, e anima e aplaude os briosos jogadores, seus representantes

Uma fase do jôgo contra o Leixões, no instante em que o jogador Lirtamento, extremo-direito do Sporting da Covilhã, marca o golo da vitória.

Figura 5.1: Tipografia retirada de página da revista Stadium, número 300, 01/09/1948

A revista em questão está devidamente preservada e arquivada pela Hemeroteca Ditigal da Câmara Municipal de Lisboa, no formato HTML [25] ou no no formato PDF [26].

Voltando ao website, o mesmo baseia o seu espectro de cores nas cores

que identificam o clube: verde, preto e branco.

O site contém um cabeçalho que é composto pelo título do site "Arquivo de Notícias SCC"; pelo seu logótipo - onde o utilizador pode clicar sempre que quiser regressar à página inicial -; por um acesso rápido à página que descreve as motivações da elaboração do site; e por um acesso rápido à secção de pesquisa de notícias, secção essa presente em todas as páginas do website.

Nesta secção, o utilizador pode introduzir termos textuais na caixa "Texto"e pesquisar notícias relacionadas aos mesmos. Pode também se o entender, filtrar a pesquisa por Ano de publicação da notícia, e pelo Orgão de Comunicação Social que a publicou.

Além destes elementos é também comum a todas as páginas um pequeno rodapé que explica sucintamente a base de elaboração do site, assim como as devidas referências, não só aos meios de comunicação social utilizados, mas também ao Arquivo.pt e à Universidade da Beira Interior (UBI).

Figura 5.2: Secção de pesquisa e rodapé do site, presentes em todas as páginas

5.2 Página Inicial

A página inicial, além dos elementos descritos na secção anterior (5.1), conta com uma secção onde contém uma pequena frase de introdução ao website, combinada com uma imagem constituída pelo nome e emblema do SCC. Clicando na seta presente na secção o utilizador pode também encontrar atalhos para a página de motivações e para a secção de pesquisa de notícias.

Figura 5.3: Secção de introdução da Página Inicial do site

5.2.1 Neste dia...

Logo abaixo da secção anterior, o utilizador encontra uma nova secção, chamada *"Neste dia..."*. Nesta secção, irão aparecer todas as notícias indexadas que aconteceram no dia, ou próximas do dia da visita do utilizador à página. Esta secção foi implementada para que diferentes artigos de notícias sejam apresentados ao utilizador a cada novo dia, sem necessidade de pesquisa. Através deste recurso , é garantido não só o dinamismo do site, mas também uma ligação mais eficaz entre o passado e o presente.

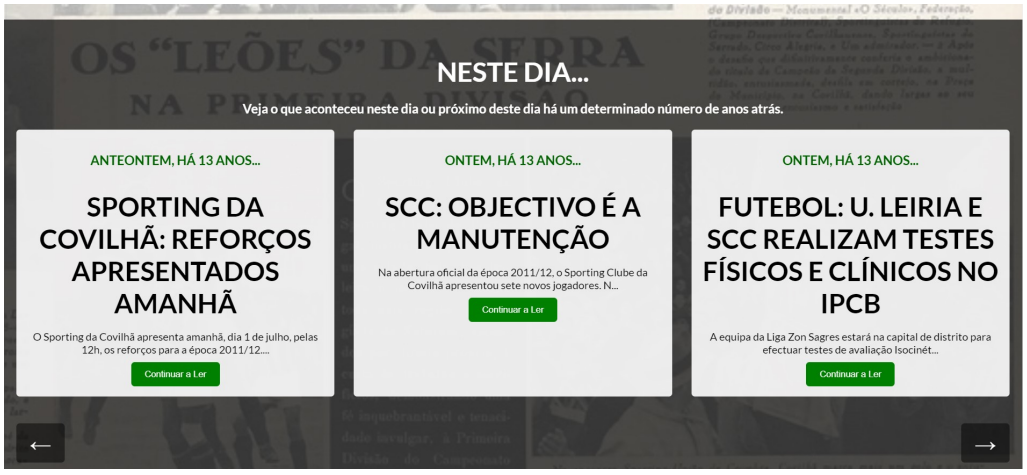

Figura 5.4: Secção "Neste dia..."presente na pág. inicial do site

5.3 Página com Resultados da Pesquisa

Após o utilizador proceder à sua pesquisa de notícias (explicada em 5.2), clicando no botão "Pesquisar", o utilizador é guiado a uma nova página com os resultados da sua pesquisa. A página terá as notícias correspondentes à pesquisa, separadas por três colunas. O exemplo abaixo (5.5) demonstra a página de resultados face à pesquisa pelo termo "benfica"por parte do utilizador.

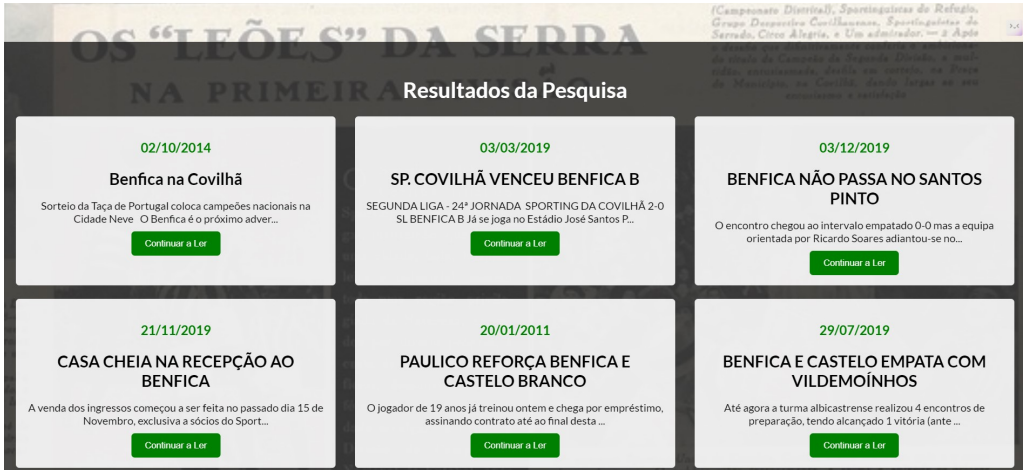

Figura 5.5: Página com os resultados da pesquisa de notícias do utilizador

No caso da pesquisa efetuada encontrar muitas notícias com correspondência, e de modo a não sobrecarregar a página, deixando-a mais limpa e organizada, os resultados da pesquisa terão apenas no máximo 9 notícias de cada vez, com o utilizador a ter a opção de clicar no botão "Carregar Mais", de modo a gerar mais 9 notícias (caso hajam mais 9, ou mais, notícias correspondentes à pesquisa), e assim sucessivamente, enquanto forem encontrados resultados com correspondência à pesquisa do utilizador.

Figura 5.6: Opção de carregar mais notícias afim de deixar a página mais organizada

5.4 Estruturação da Notícia

Quando o utilizador clica no título, ou no botão "Continuar a Ler"de uma notícia - seja pela secção 5.2.1 ou pela secção de Pesquisa -, a notícia será apresentada ao utilizador, numa nova página, com todos os campos (Título, Subtítulo, Secção, Texto, Data, Autor, Fotografia, Link original) devidamente estruturados (se existirem, claro).

Figura 5.7: Exemplo de Notícia, aquando da visita do Benfica à Covilhã, em 2014

O utilizador poderá ler a notícia com a estrutura e estilização do site, mas também tem a opção de clicar em "Ver Notícia Original", onde abrirá um novo

separador com a notícia arquivada (pelo Arquivo.pt) com a versão da altura da publicação.

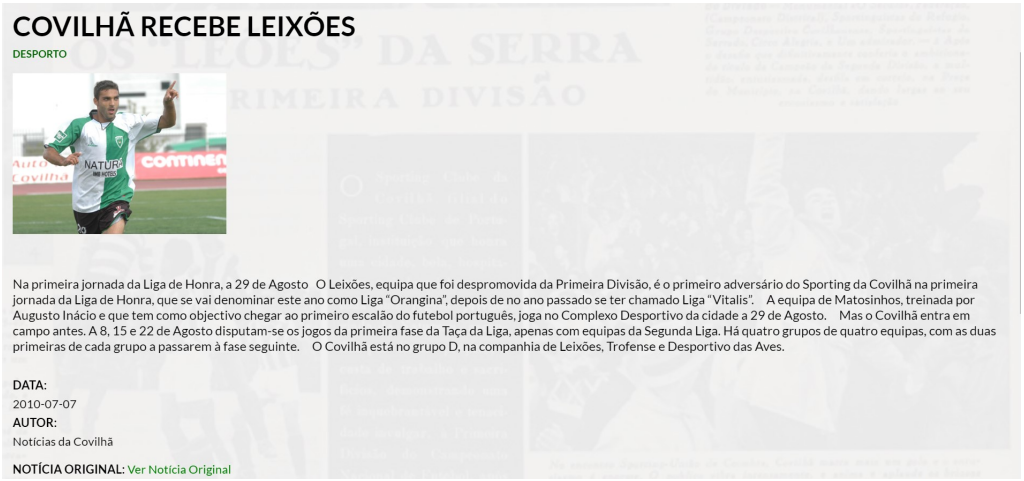

Figura 5.8: Exemplo de notícia com todos os campos disponíveis

5.5 Página das Motivações

Por último, a página das motivações descreve um resumo das motivações relacionadas à elaboração do site, onde é possível ler o enunciado da proposta referente ao Projeto, um pequeno resumo de implementação do site, uma referência aos autores do projeto, e uma mensagem final que espelha o objetivo do desenvolvimento do website.

Figura 5.9: Parte inicial da página referente às motivações do projeto

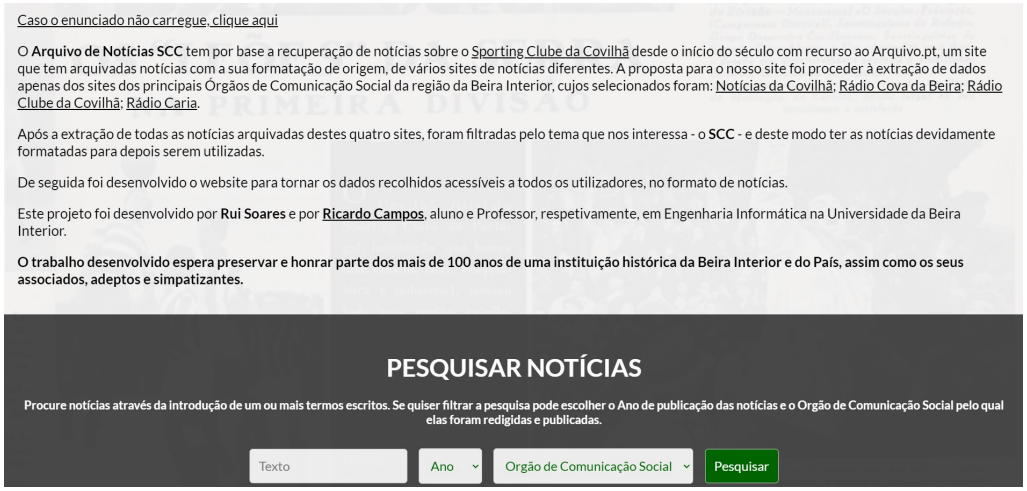

Figura 5.10: Pequeno texto descritivo, inerente às motivações do projeto

5.6 Sumário

Concluindo, a interface do site oferece uma experiência ao utilizador que permite uma fácil navegação pelas suas diversas secções e funcionalidades. Com um *layout* atualizado para uma usabilidade moderna e adequada ao tema do site, os utilizadores podem explorar artigos de notícias sobre o SCC com facilidade por meio da secção de pesquisa. A implementação de secções dinâmicas como "Neste dia..."(5.2.1) adiciona um elemento envolvente, proporcionando aos utilizadores uma visão de eventos históricos e notícias do passado.

O website está disponível através do link: arquivoscc.ipt.pt.

Capítulo

6

Conclusões e Trabalho Futuro

Este projeto integrou e aplicou diversos aspetos relacionados à extração de dados da web, recuperação de informação e desenvolvimento web, proporcionando uma compreensão abrangente dessas tecnologias. Ao longo do desenvolvimento deste projeto, implementámos e contribuímos com os seguintes componentes: **(1)**A extração de dados referentes a artigos de notícias de quatro órgãos de comunicação social da região da Beira Interior, cuidadosamente organizados. **(2)**Seleção dos artigos noticiosos com o objetivo de formar um conjunto de dados de notícias apenas referentes ao Sporting Clube da Covilhã. **(3)**Desenvolvimento de funcionalidades de recuperação de informações, proporcionando aos utilizadores a possibilidade de utilizar uma ferramenta poderosa para pesquisar por informações específicas, de forma rápida e eficiente, no conjunto de dados formado. **(4)**Desenvolvimento de um website dinâmico e acessível, entregando aos utilizadores dinamicamente artigos noticiosos do passado e a possibilidade de pesquisa por informações específicas.

A implementação conjunta destes componentes contribuiu para preservar o património e a memória coletiva de uma instituição desportiva histórica, não só da região da Beira Interior, como do país, o Sporting Clube da Covilhã. O projeto contribui também para a preservação da história e do património dos meios de comunicação social da região descritos, tão importantes na divulgação e crescimento da região.

Chegados a este ponto, os próximos esforços deverão ser centrados em aspetos como: a extração de dados a partir de um maior número de meios de comunicação social da região, com o objetivo de expandir ainda mais o conjunto de dados recolhidos e garantir uma cobertura mais abrangente das notícias; melhorias no design e na estrutura do website, tornando-o mais intuitivo, atrativo e acessível para os utilizadores; e a integração e exploração

de novas funcionalidades no website, como a implementação de um sistema de recomendação de notícias personalizado, que possa sugerir conteúdos relevantes aos utilizadores com base nas suas preferências e histórico de navegação, visando assim oferecer uma experiência mais ampla ao utilizador, e, consequentemente, expandir e divulgar mais o website.

Apêndice

A

Proposta de Projeto

Este apêndice apresenta o documento referente ao enunciado da proposta do Projeto desenvolvido.

Arquivo de Notícias do Sporting Clube da Covilhã

Proposta de Projeto. Lic. Eng. Informática

Orientador: Ricardo Campos (ricardo.campos@ubi.pt)

Objetivos

No ano do centenário do Sporting Clube da Covilhã, pretende-se utilizar os recursos do Arquivo.pt (plataforma de preservação de conteúdos da web portuguesa) para tornar acessível ao público o conjunto de notícias publicadas ao longo da última década nas plataformas digitais dos principais meios de comunicação social da região com foco no principal clube da cidade da Covilhã. Em particular, este projeto visa a disponibilização desses conteúdos através do desenvolvimento de um website dedicado e de um sistema de pesquisa que permita aos utilizadores da plataforma web, recuperar informações acerca do clube a partir dos dados coletados do Arquivo.pt. Pretende-se com este projeto contribuir para a preservação do património desportivo local e a memória coletiva do Sporting Clube da Covilhã.

Plano de Trabalho

T1: Definição e planeamento (2 semanas)

T2: Recolha e armazenamento dos dados do Arquivo.pt com recurso ao desenvolvimento de scripts em Python. Aplicação de medidas de similaridade textual com vista à eliminação de notícias duplicadas (4 semanas)

T3: Criação de uma imagem Docker com vista à integração dos vários componentes (1 semana)

T4: Desenvolvimento do website (4 semanas)

T5: Implementação do sistema de pesquisa com recurso ao Elastic Search (base de dados noslq) e ao algoritmo de recuperação de informação BM25 (3 semanas)

T6: Testes Finais (2 semanas)

T7: Documentação e Apresentação (2 semanas)

A Tabela 1 apresenta a distribuição das tarefas por cada uma das 15 semanas.

Tabela 1: Cronologia das tarefas (T) por semana (S).

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
T1															
T2															
T3															
T4															
T5															
T6															
T7															

Requisitos Técnicos / Académicos

- conhecimentos sólidos em Python (para a recolha de dados) ou disposição para aprender.
- conhecimento em linguagens de programação web (HTML, CSS, JavaScript) ou Flask para o desenvolvimento do website.
- experiência com Elasticsearch (base de dados nosql) ou disposição para aprender.
- familiaridade com a criação de imagens Docker.
- bons conhecimentos de programação, engenharia de software e composição web

Resultados Esperados

- script python de obtenção dos dados e respetivo dataset coletado
- website responsivo (código a disponibilizar no github do aluno) e disponível online
- relatório
- candidatura ao Prémio Arquivo.pt 2024 (lista de vencedores dos prémios anteriores: https://sobre.arquivo.pt/pt/colabore/premios-arquivo-pt/)

Bibliografia

[1] Jornalismo no século xix. https://www.newsmuseum.pt/pt/imortais/jornalismo-sec-xix.

[2] Célia Gouveia. Media e futebol: uma relação simbiótica, 2018. https://www.researchgate.net/publication/325951338_Media_e_futebol_uma_relacao_simbiotica.

[3] NewsMuseum. Retratar a emoção do Desporto. https://www.newsmuseum.pt/pt/desporto/retratar-emocao-do-desporto.

[4] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the internet. *SIGCOMM Comput. Commun. Rev.*, 39(5):22–31, oct 2009.

[5] Daniel Gomes. Web archives as research infrastructure for digital societies: the case study of Arquivo.pt. *Archeion*, Nov 14 2022.

[6] Introduction to the Special Issue on Web Archiving. *New Review of Hypermedia and Multimedia*, 13(1):3–5, 2007.

[7] Ingrid Parent and Neil Cruickshank. The Growth of the Internet and Knowledge Networks, and their Impact in the Developing World. *Information Development*, 25:91–98, 05 2009.

[8] Daniel Gomes, David Cruz, João Miranda, Miguel Costa, and Simão Fontes. Search the Past with the Portuguese Web Archive. In *22nd International World Wide Web Conference*, Rio de Janeiro, Brasil, May 2013.

[9] Brewster Kahle. Preserving the internet. *Scientific American*, 276:82–83, 1997.

[10] Informações Gerais - Arquivo.pt. https://sobre.arquivo.pt/pt/ajuda/o-que-e-o-arquivo-pt/.

[11] Daniel Gomes, Elena Demidova, Jane Wintes, and Thomas Risse. The Past Web – Exploring Web Archives, 2021.

[12] Stephanie Shreffler Kayla Harris, Christina Beis. The internet archive has been fighting for 25 years to keep what's on the web from disappearing – and you can help. *The conversation*, 2021. https://theconversation.com/the-internet-archive-has-been-fighting-for-25-years-to-keep-whats-on-the-web-f

[13] *The Web as History: Using Web Archives to Understand the Past and the Present*. UCL Press, 2017.

[14] How can you use web scraping to identify new market trends? https://pt.linkedin.com/advice/1/how-can-you-use-web-scraping-identify-new-market-bogse?lang=en.

[15] How can you use sentiment analysis techniques to analyze scraped text data from websites? https://pt.linkedin.com/advice/0/how-can-you-use-sentiment-analysis-techniques-analyze-n2kjc?lang=en.

[16] Adam Jatowt Vítor Mangaravite Alípio Mário Jorge Ricardo Campos, Arian Pasquali. Automatic Generation of Timelines for Past-Web Events, 2021. https://link.springer.com/chapter/10.1007/978-3-030-63291-5_18.

[17] Sql vs. nosql: The most important differences. https://blog.udemy.com/nosql-vs-sql/?utm_source=adwords&utm_medium=udemyads&utm_campaign=DSA_Catchall_la.EN_cc.ROW&campaigntype=Search&portfolio=ROW-English&language=EN&product=Course&test=&audience=DSA&topic=&priority=&utm_content=deal4584&utm_term=_._ag_88010211481_._ad_535397282064_._kw__._de_c_._dm__._pl__._ti_dsa-393987629421_._li_1011711_._pd__._&matchtype=&gad_source=1&gclid=CjwKCAjwnK60BhA9EiwAmpHZw8RgaLq-86tBzs1Rk8I1sk6W0NPcls7PzyeHVuf7q93bXe4bmk7dfhoBwE.

[18] Term frequency - inverse document frequency. [web] https://en.wikipedia.org/wiki/Tf%E2%80%93idf.

[19] Best maching 25. [web] https://www.luigisbox.com/search-glossary/bm25/.

[20] Nikita Kathare, O. Vinati Reddy, and Vishalakshi Prabhu. A Comprehensive Study of Elasticsearch. *International Journal of Science and Research (IJSR)*, 2021.

[21] Elasticsearch. [web] https://codingexplained.com/coding/elasticsearch/understanding-analysis-in-elasticsearch-analyzers.

[22] Ricardo Campos, Diogo Correia, and Adam Jatowt. Public News Archive: A Searchable Sub-archive to Portuguese Past News Articles, 2023. https://link.springer.com/chapter/10.1007/978-3-031-28241-6_16.

[23] What is Flask Python. [web] https://pythonbasics.org/what-is-flask-python//.

[24] What is Nginx. [web] https://nginx.org/en/.

[25] Stadium. Revista Stadium, nº 300, 1948. [HTML] https://hemerotecadigital.cm-lisboa.pt/Periodicos/Stadium/1948/N300/N300_item1/index.html.

[26] Stadium. Revista Stadium, nº 300, 1948. [PDF] https://hemerotecadigital.cm-lisboa.pt/Periodicos/Stadium/1948/N300/N300_master/StadiumN300_1948.PDF.