Projeto de Algoritmos e Técnicas de Programação

Relatório Final

Grupo 27

Liliana Miranda Quelha — A93767

Mafalda Santos Meixedo — A103834

Paulo Samuel Lopes de Vasconcelos — AE8186

Curso: Mestrado Integrado e Licenciatura de Engenharia Biomédica

Data de Entrega: 5 de Janeiro de 2024

Docentes: José Carlos Ramalho, Dep. Informática - UMinho, `jcr@di.uminho.pt`

Luís Filipe Cunha, Dep. Informática - UMinho, `lfc@di.uminho.pt`

1 Introdução

Este relatório apresenta o desenvolvimento de um sistema para consulta e análise de publicações científicas, criado no âmbito da unidade curricular de Algoritmos e Técnicas de Programação. O sistema visa a manipulação eficiente de uma base de dados, oferecendo funcionalidades para inserção, atualização, pesquisa, análise e exportação de informações sobre publicações.

A motivação para este projeto surge da necessidade de organizar e explorar grandes volumes de informação científica no formato de publicações, tornando-as mais acessíveis e mais facilmente analisáveis. A aplicação é implementada em Python, tirando partido das suas bibliotecas e ferramentas para manipulação de dados, visualização gráfica e criação de interfaces interativas. O sistema desenvolvido é estruturado para suportar operações fundamentais, como carregamento e gravação de dados, e funcionalidades avançadas, como análise estatística por autor e palavras-chave entre outras funcionalidades que serão descritas neste relatório.

A aplicação desenvolvida contém diversos botões, como: **Ordenar**, **Filtrar**, **Gráficos**, **Gravar** e **Sair**. Para além disso, de forma a tornar a aplicação mais completa, criamos uma janela para cada publicação com a informação do artigo.

2 Funcionalidade do Sistema

Na execução deste projeto, teve que se ter atenção aos seguintes requisitos:

2.1 [1] Carregamento da Base de Dados

No início do programa, o dataset que contém as publicações científicas é carregado para a memória a partir de um ficheiro JSON. Esse processo é essencial para garantir que todas as publicações previamente armazenadas sejam recuperadas e disponibilizadas para manipulação no sistema.

Este código é executado automaticamente ao iniciar o sistema para carregar as publicações existentes do ficheiro `ata_medica_papers.json`. Caso o ficheiro não exista ou esteja vazio, o sistema exibe um erro e requer que o utilizador verifique o ficheiro de suporte.

2.1.1 Código:

```
import json

def abre_ficheiro(fnome):
    f = open(fnome, encoding='utf-8')
    novo = json.load(f)
    f.close()
    return novo
```

2.1.2 Explicação:

- **import json:** Importa o módulo necessário para manipulação de ficheiros JSON.
- **def abre_ficheiro(fnome):** Define a função responsável por carregar o ficheiro especificado.
- **f = open(fnome, encoding='utf-8'):** Abre o ficheiro especificado pelo parâmetro `fnome` em modo de leitura e codificação UTF-8.
- **novo = json.load(f):** Carrega o conteúdo do ficheiro JSON em formato de dicionário Python.
- **f.close():** Fecha o ficheiro após o carregamento para liberar recursos.
- **return novo:** Retorna o conteúdo carregado para ser utilizado pelo sistema.

2.2 Criação de Publicações

O utilizador pode adicionar novas publicações especificando detalhes como título, resumo, palavras-chave, autores, afiliações, DOI, URLs e datas de publicação. Antes da inserção, o sistema verifica a unicidade do identificador (DOI) para evitar duplicações. Após a validação, os dados são adicionados à estrutura em memória e podem ser imediatamente gravados no ficheiro de suporte.

O sistema permite ao utilizador criar novas publicações especificando os seguintes campos obrigatórios:

- **Título**: Nome do artigo.
- **Resumo**: Breve descrição do conteúdo.
- **Palavras-chave**: Termos relacionados ao tema do artigo.
- **DOI**: Identificador digital único do artigo.
- **Autores**: Lista de autores com nomes e afiliações.
- **URL do artigo**: Link para o artigo online.
- **URL do PDF**: Link para o ficheiro PDF.
- **Data de publicação**: Data formatada como YYYY-MM-DD.

Código:

```
from datetime import datetime

def cria_artigo():
    titulo = str(input("Introduza o título do artigo"))
    resumo = str(input("Pode digitar o resumo do artigo"))
    palavrachave = str(input("Introduza as palavras chaves relacionadas "))
    doi = str(input("Introduza o número identificador do artigo"))
    data = cria_data()
    num = int(input("Introduza o número de autores do artigo"))
    autores = []
    for i in range(0,num):
        nome = str(input("Introduza o nome do autor"))
        afil = str(input("Introduza a afiliação do autor"))
        autores.append({
            "name":nome,
            "affiliation":afil
        })
    url = str(input("Introduza o url do artigo"))
    pdf = str(input("Introduza o url do pdf"))
    artigo = {
        "title":titulo,
        "abstract":resumo,
        "keywords":palavrachave,
        "doi":doi,
        "url":url,
        "pdf":pdf,
        "publish_date":data,
        "authors":autores
    }
    print("Artigo criado com sucesso.")
    return artigo

def cria_data():
    return datetime.today().strftime('%Y-%m-%d')
```

Explicação:

- `from datetime import datetime`: Importa a biblioteca para manipulação de datas.
- `def cria`$_a rtigo()$: $Define a função principal para criar uma nova publicação.$
- `input()`: Captura os campos obrigatórios preenchidos pelo utilizador.
- `cria`$_a ata()$: $Gera automaticamente a data atual no formato YYYY - MM - DD.$
- A função cria um dicionário estruturado e exibe uma mensagem de sucesso após a criação.

Utilização no Sistema:

- **CLI**: Selecionar a opção `1` no menu e preencher os campos solicitados.
- **GUI**: Preencher os campos no formulário da interface gráfica e clicar em `Criar`.

2.3 Criação de Publicações

O utilizador pode adicionar novas publicações especificando detalhes como título, resumo, palavras-chave, autores, afiliações, DOI, URLs e datas de publicação. Antes da inserção, o sistema verifica a unicidade do identificador (DOI) para evitar duplicações. Após a validação, os dados são adicionados à estrutura em memória e podem ser imediatamente gravados no ficheiro de suporte.

2.4 Atualização de Publicações

Esta funcionalidade permite a edição de publicações já existentes. O utilizador pode modificar informações como título, resumo, palavras-chave, autores e datas de publicação. Para realizar a atualização, o sistema solicita um identificador (DOI) ou outro critério de busca. Após localizar o registo, as alterações são aplicadas e podem ser salvas no ficheiro JSON, garantindo a atualização imediata dos dados.

2.4.1 Atualizar a Data de Publicação

```
def atualiza_data(artigos, doi):
    datan = cria_data()
    for elem in artigos:
        if elem["doi"] == doi:
            elem["publish_date"] = datan
    return artigos
```

Explicação:

- Atualiza a data da publicação para a data atual.
- Utilização: Inserir o DOI do artigo e confirmar a nova data.

2.4.2 Atualizar o Resumo

```
def atualiza_resumo(artigos, doi):
    resumon = str(input("Introduza um novo resumo"))
    for elem in artigos:
        if elem["doi"] == doi:
            elem["abstract"] = resumon
    return artigos
```

Explicação:

- Atualiza o resumo do artigo.
- Utilização: Inserir o DOI e fornecer o novo resumo.

2.4.3 Atualizar Palavras-chave

```
def atualiza_key(artigos, doi):
    chaven = str(input("Introduza novas palavras chave!"))
    for elem in artigos:
        if elem["doi"] == doi:
            elem["keywords"] = chaven
    return artigos
```

Explicação:

- Atualiza as palavras-chave associadas ao artigo.
- Utilização: Inserir o DOI e digitar as novas palavras-chave.

2.4.4 Atualizar Autores

```
def atualiza_autores(artigos, doi):
    listaut = []
    a = int(input("Qual o número de autores que deseja introduzir?"))
    while a > 0:
        b = str(input("Nome do autor"))
        listaut.append(b)
        a = a - 1
    for elem in artigos:
        if elem["doi"] == doi:
            elem["authors"] = [{"name": nome, "affiliation": ""} for nome in listaut]
    return artigos
```

Explicação:

- Atualiza a lista de autores do artigo.
- Utilização: Fornecer o DOI e os nomes dos autores.

2.4.5 Atualizar Afiliações

```
def atualiza_afiliacoes(artigos, doi):
    afi = []
    t = int(input("Introduz o número de afiliações"))
    while t > 0:
        c = str(input("Introduza a afiliação."))
        afi.append(c)
        t = t - 1
    for elem in artigos:
        if elem["doi"] == doi:
            for aut in elem["authors"]:
                aut["affiliation"] = afi[0]
    return artigos
```

Explicação:

- Atualiza as afiliações dos autores.
- Utilização: Inserir o DOI e adicionar afiliações.

3 Consulta de Publicações

A consulta permite ao utilizador pesquisar publicações na base de dados utilizando critérios específicos, como título, autor, afiliação, data de publicação ou palavras-chave. A pesquisa retorna uma lista formatada de resultados, permitindo ao utilizador verificar os detalhes de cada publicação. Os resultados podem ser ordenados alfabeticamente ou por data, facilitando a navegação em conjuntos extensos de registos.

3.1 Código

```
def consulta_publicacoes(artigos):
    print("Filtros disponíveis:")
    print("1. Título")
    print("2. Autor")
    print("3. Afiliação")
    print("4. Data de Publicação")
    print("5. Palavras-chave")
    filtro = int(input("Escolha um filtro (1-5): "))
    resultados = []

# Explicação: Exibe os filtros disponíveis e solicita a escolha do utilizador.

    if filtro == 1:
        busca = input("Escreva o título ou parte dele: ").lower()
        resultados = [a for a in artigos if busca in a['title'].lower()]

# Explicação: Filtra os artigos cujo título contenha o texto pesquisado,
#ignorando maiúsculas/minúsculas.

    elif filtro == 2:
        busca = input("Escreva o nome do autor: ").lower()
        resultados = [a for a in artigos if any(busca in aut['name'].lower() for aut in a['authors'])]

# Explicação: Filtra os artigos por autores,
# verificando se algum autor contém o texto pesquisado.

    elif filtro == 3:
        busca = input("Escreva a afiliação: ").lower()
        resultados = [a for a in artigos if any(busca in aut['affiliation'].lower() for aut in a['autho

# Explicação: Filtra os artigos por afiliação dos autores,
# verificando correspondências parciais.
```

```
    elif filtro == 4:
        busca = input("Escreva a data (YYYY-MM-DD): ")
        resultados = [a for a in artigos if a['publish_date'] == busca]

# Explicação: Filtra os artigos com a data de publicação exata informada.

    elif filtro == 5:
        busca = input("Escreva uma palavra-chave: ").lower()
        resultados = [a for a in artigos if busca in [kw.strip().lower() for kw in a['keywords'].split(

# Explicação: Filtra os artigos verificando a presença da palavra-chave na lista de palavras-chave.

    else:
        print("Opção inválida!")
        return

    print("Ordenar resultados por:")
    print("1. Título")
    print("2. Data de publicação")
    ordem = int(input("Escolha a ordenação (1-2): "))

# Explicação: Exibe opções de ordenação e solicita escolha entre título ou data.

    if ordem == 1:
        resultados = sorted(resultados, key=lambda x: x['title'].lower())
    elif ordem == 2:
        resultados = sorted(resultados, key=lambda x: x['publish_date'])

# Explicação: Ordena os resultados com base no critério selecionado.

    for r in resultados:
        print(f"\nTítulo: {r['title']}")
        print(f"Autores: {', '.join([aut['name'] for aut in r['authors']])}")
        print(f"Data: {r['publish_date']}")
        print(f"Palavras-chave: {r['keywords']}")
        print(f"DOI: {r['doi']}")
        print(f"URL: {r['url']}")
        print("-" * 50)

    # Explicação: Exibe os detalhes de cada artigo encontrado, incluindo título, autores e links.
```

4 Armazenamento, Importação e Exportação dos Dados

O sistema suporta a importação de novos registos a partir de ficheiros JSON com a mesma estrutura do ficheiro de suporte. Esta funcionalidade é útil para combinar múltiplas bases de dados ou atualizar a base existente. Adicionalmente, é possível exportar subconjuntos de dados resultantes de pesquisas para ficheiros JSON separados. Esse recurso facilita o compartilhamento e análise de dados filtrados.

4.1 Importação de Publicações

4.1.1 Código de Importação

```
def importar_publicacoes(ficheiro, artigos):
    with open(ficheiro, 'r', encoding='utf-8') as f:
        novos_artigos = json.load(f)
        artigos.extend(novos_artigos)
    print("Publicações importadas com sucesso!")
```

4.1.2 Explicação

Esta função permite importar novas publicações de um ficheiro externo com estrutura JSON e adicioná-las à lista existente de artigos carregados na memória.

- **ficheiro:** Caminho do ficheiro a ser importado (deve estar no formato JSON).
- **artigos:** Lista atual de publicações carregadas na memória.

Detalhes Técnicos:

- `open(ficheiro, 'r', encoding='utf-8')`: Abre o ficheiro especificado no modo leitura ('r') e utiliza a codificação 'utf-8' para suportar caracteres especiais (acentos, símbolos).
- A utilização de `with...` garante que o ficheiro fecha automaticamente após a execução, evitando vazamentos de memória ou ficheiros corrompidos.
- `extend()`: Adiciona os novos artigos importados à lista já existente na memória e preserva os artigos carregados anteriormente, apenas acrescentando os novos.

4.2 Exportação de Publicações

4.2.1 Código de Exportação

```
def exportar_publicacoes(ficheiro, artigos):
    with open(ficheiro, 'w', encoding='utf-8') as f:
        json.dump(artigos, f, indent=4)
    print("Resultados exportados com sucesso!")
```

4.2.2 Explicação

Esta função exporta as publicações presentes na memória para um ficheiro JSON, criando um backup ou permitindo a exportação de resultados filtrados.

- **ficheiro:** Caminho do ficheiro onde os artigos serão salvos.
- **artigos:** Lista de publicações que será exportada.

Detalhes Técnicos:

- `with open(ficheiro, 'w', encoding='utf-8') as f:` Abre o ficheiro no modo escrita ('w'). Substitui qualquer conteúdo existente no ficheiro e utiliza codificação 'utf-8' para garantir compatibilidade com caracteres especiais. A instrução `with...` garante que o ficheiro é fechado automaticamente após salvar os dados.
- `json.dump()`: Converte os artigos armazenados na memória para formato JSON e salva-os no ficheiro especificado. O parâmetro `indent=4` formata os dados com indentação de 4 espaços, tornando-os mais legíveis.

5 Interfaces

O sistema inclui duas interfaces distintas para interação com o utilizador: uma interface de linha de comando (CLI) e uma interface gráfica (GUI).

5.1 Interface de Linha de Comando (CLI)

A interface CLI permite que os utilizadores executem comandos específicos para criar, consultar, atualizar e remover publicações. Essa abordagem é eficiente para utilizadores avançados que preferem interação textual e scripts automatizados. Os comandos são acompanhados de mensagens de ajuda detalhadas, facilitando a utilização do sistema mesmo sem conhecimento prévio.

5.1.1 Comando de Ajuda

Após executar o programa Python, o utilizador será levado ao menu principal. Para exibir as opções disponíveis, digite:

```
help
```

Explicação: Este comando imprime uma mensagem de ajuda com os comandos disponíveis.

5.2 Funcionalidades do Sistema

5.2.1 (1) Criar uma Nova Publicação

Selecione a opção 1 para criar uma publicação, preenchendo os seguintes campos:

- Introduza o título do artigo:
- Digite o resumo:
- Adicione palavras-chave:
- Forneça o DOI do artigo:
- Introduza o número identificador do artigo:
- Insira a data de publicação (YYYY-MM-DD):
- Adicione autores e afiliações:
  - Introduza o número de autores do artigo:
  - Introduza o nome do autor:
  - Introduza a afiliação do autor:
- Insira URLs do artigo e PDF:
  - Introduza o URL do artigo:
  - Introduza o URL do PDF:

```
def cria_artigo():
    titulo = str(input("Introduza o título do artigo"))
    resumo = str(input("Pode digitar o resumo do artigo"))
    palavrachave = str(input("Introduza as palavras chaves relacionadas "))
    doi = str(input("Introduza o número identificador do artigo"))
    data = cria_data()
    num = int(input("Introduza o número de autores do artigo"))
    autores = []
    for i in range(0,num):
        nome = str(input("Introduza o nome do autor"))
        afil = str(input("Introduza a afiliação do autor"))
        autores.append({
            "name":nome,
            "affiliation":afil
        })
    url = str(input("Introduza o url do artigo"))
    pdf = str(input("Introduza o url do pdf"))
    artigo = {
        "title":titulo,
        "abstract":resumo,
        "keywords":palavrachave,
        "doi":doi,
        "url":url,
        "pdf":pdf,
        "publish_date":data,
        "authors":autores
    }
    return artigo
```

Figure 1: Função cria_artigo

5.2.2 (2) Consultar uma Publicação Específica

Selecione a opção 2 para consultar um artigo por DOI:

- Digite o DOI do artigo:

Exemplo de saída:

```
Título: Introdução à IA
Data: 2023-05-15
Resumo: Este artigo explora a inteligência artificial.
Autores: João Silva, Maria Oliveira
Palavras-chave: IA, Machine Learning, Deep Learning
DOI: 10.1234/ia2023
URL: www.exemplo.com/ia2023
```

```python
def achar_artigo(id, artigos):
    for artigo in artigos:
        if id == artigo["doi"]:
            a = artigo
    return a
```

Figure 2: Função achar_artigo

5.2.3 (3) Consultar e Filtrar Publicações

Selecione a opção 3 para consultar publicações aplicando filtros:

- Escolha um filtro:
  - 1. Título
  - 2. Autor
  - 3. Afiliação
  - 4. Data de Publicação
  - 5. Palavras-chave
- Digite o valor de busca conforme o filtro selecionado:
- Escolha a ordenação:
  - 1. Título
  - 2. Data de publicação

```python
def filtrar_artigos(artigos):
    filtro = sg.popup_get_text("Escolha o filtro: Título, Autor, Afiliação, Data ou Palavra-chave:").lower()
    busca = sg.popup_get_text("Digite o valor para filtro:").lower()
    resultados = []

    if filtro == 'título':
        resultados = [a for a in artigos if busca in a['title'].lower()]
    elif filtro == 'autor':
        resultados = [a for a in artigos if any(busca in aut['name'].lower() for aut in a['authors'])]
    elif filtro == 'afiliação':
        resultados = [a for a in artigos if any(busca in aut['affiliation'].lower() for aut in a['authors'])]
    elif filtro == 'data':
        resultados = [a for a in artigos if a['publish_date'] == busca]
    elif filtro == 'palavra-chave':
        resultados = [a for a in artigos if busca in [kw.strip().lower() for kw in a['keywords'].split(',')]]
    else:
        sg.popup("Filtro inválido!")
        return artigos

    return resultados
```

Figure 3: Função filtrar_artigos

5.2.4 (4) Eliminar uma Publicação

Selecione a opção 4 para eliminar uma publicação:

- Introduza o DOI da publicação a eliminar:

```python
if nevent == 'Eliminar':
    artigos.remove(artigo_escolhido)
    janela_info.close()
    dados = cria_dados(artigos)
    window['-TABELA-'].update(values = dados)
    break
```

Figure 4: Função que Elimina um Artigo

5.2.5 (5) Gerar Relatório de Estatísticas

Selecione a opção 5 para gerar estatísticas, como:

- Distribuição por Ano
- Palavras-chave mais frequentes
- Publicações por Autor

```python
def estat(artigos):
    palavras = []
    for artigo in artigos:
        palavras.extend(artigo['keywords'].lower().split(','))
    contagempalchave = Counter([p.strip() for p in palavras])

    anos = [artigo['publish_date'][:4] for artigo in artigos]
    contagemanos = Counter(anos)

    autores = []
    for artigo in artigos:
        for autor in artigo['authors']:
            autores.append(autor['name'])
    contagemautores = Counter(autores)

    print(contagemanos, contagempalchave, contagemautores)
    return
```

Figure 5: Função que Gera Relatório de Estatísticas

5.2.6 (6) Listar Autores e Seus Artigos

Selecione a opção 6 para listar os autores e seus artigos.

```
def listar_autores(artigos):
    a = []
    autores = []
    for elem in artigos:
        a.append(elem['authors'])
        for x in a:
            if x['name'] not in autores:
                autores.append(x['name'])
    return autores
```

Figure 6: Função listar_autores

5.2.7 (7) Importar Novas Publicações

Selecione a opção 7 para importar publicações:

– Digite o caminho do ficheiro a importar: `novo_dataset.json`

Confirmação:

```
Publicações importadas com sucesso!
```

```
def importar_publicacoes(ficheiro, artigos):
    with open(ficheiro, 'r', encoding='utf-8') as f:
        novos_artigos = json.load(f)
        artigos.extend(novos_artigos)
    print("Publicações importadas com sucesso!")
```

Figure 7: Função importar_publicacoes

5.2.8 (8) Exportar Publicações

Selecione a opção 8 para exportar resultados:

– Digite o caminho do ficheiro para exportar: `exportados.json`

Confirmação:

```
Resultados exportados com sucesso!
```

```
def exportar_publicacoes(ficheiro, artigos):
    with open(ficheiro, 'w', encoding='utf-8') as f:
        json.dump(artigos, f, indent=4)
    print("Resultados exportados com sucesso!")
```

Figure 8: Função exportar_publicacoes

5.2.9 (9) Guardar Alterações

Selecione a opção 9 para guardar todas as alterações realizadas.
Confirmação:

```
Dados salvos com sucesso!
```

```python
def gravar(artigos,fnome):
    f = open(fnome,"w",encoding="utf-8")
    json.dump(artigos,f)
    return
```

Figure 9: Função gravar

5.2.10 (10) Sair do Sistema

Selecione a opção 10 para sair do sistema.

Confirmação:

```
Programa encerrado.
```

5.3 Interface Gráfica (GUI)

A interface gráfica (GUI) apresenta uma janela principal com botões organizados para acesso rápido às funcionalidades.

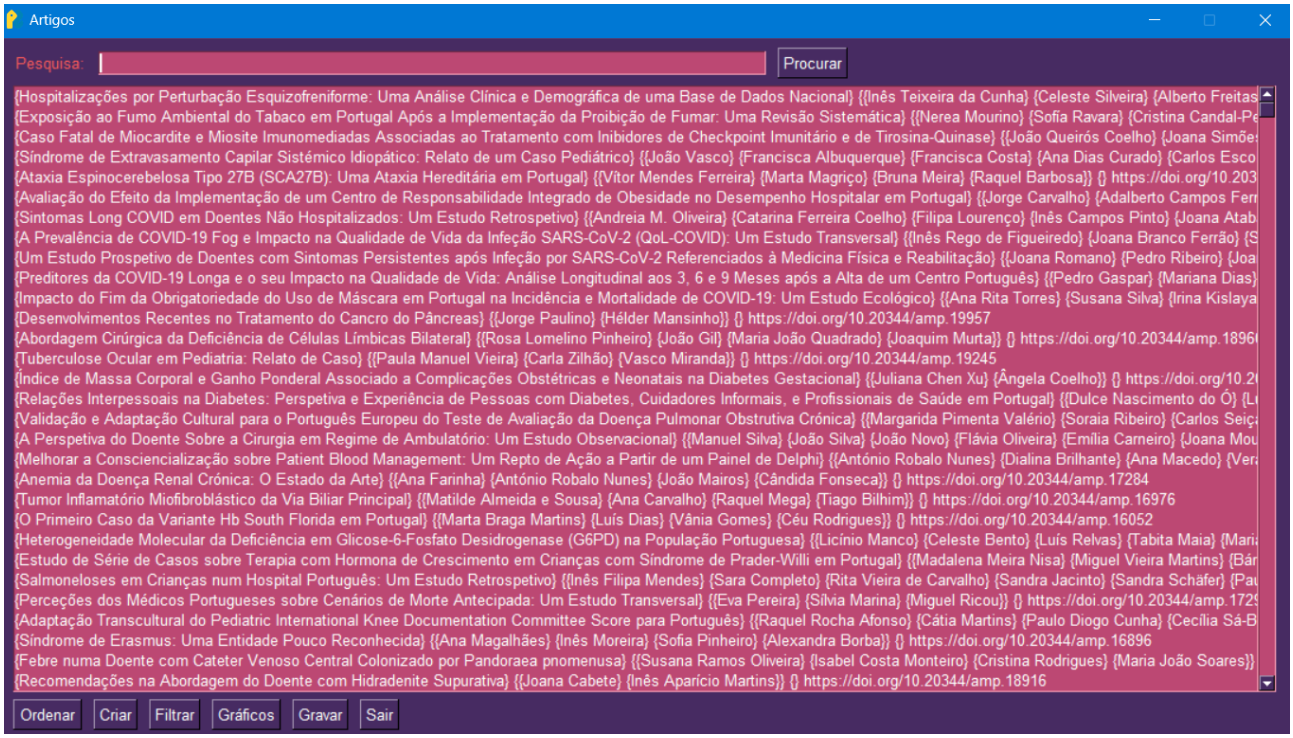

Figure 10: Interface Gráfica

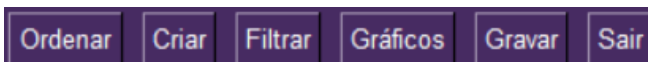

Figure 11: Botões

5.3.1 3.1. Botão Criar

Foi desenvolvido o botão **'Criar'**, permitindo que o utilizador crie um artigo especificando os seguintes campos:

- **Título:** Título do artigo.
- **Resumo:** Breve descrição do conteúdo do artigo.
- **Palavras-chave:** Lista de palavras-chave relacionadas ao artigo.

- **DOI:** Identificador Digital do Objeto.
- **Autores e Afiliações:**
  * Nome do autor.
  * Afiliação do autor.
- **URLs:**
  * URL para o ficheiro PDF do artigo.
  * URL do artigo publicado.
- **Data de Publicação:** Data no formato YYYY-MM-DD.

Funcionamento: Quando o utilizador preenche todos os campos e clica no botão **'Criar'**, os dados são validados e adicionados à base de dados do sistema. Uma mensagem de confirmação é exibida ao concluir a criação do artigo.

Exemplo de Mensagem de Confirmação:

```
Artigo criado com sucesso!
```

Figure 12: Interface do Botão "Criar"

5.3.2 3.2. Botão Ordenar

O botão **'Ordenar'** permite ao utilizador ordenar o conjunto de publicações por autor, título, afiliação e data.

Funcionamento:

- Caso o utilizador clique no botão **Ordenar**, será ativada a ordenação através da chave selecionada.
- Existem 4 opções disponíveis para ordenação:
  1. Título
  2. Data
  3. Autor
  4. Afiliação
- Se a ordenação for feita por título, os artigos serão organizados em ordem alfabética utilizando o método `sort()`.
- Se a ordenação for feita por data, os artigos serão ordenados em ordem crescente (passado para presente) também utilizando o método `sort()`.
- As opções autor e afiliação seguem o mesmo comportamento, classificando os dados alfabeticamente.

5.3.3 3.3. Botão Filtrar

O botão **'Filtrar'** permite pesquisar publicações no sistema utilizando diferentes filtros e exibe os resultados ordenados conforme a preferência do utilizador.

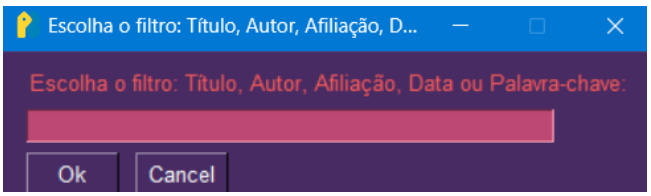

Figure 13: Interface do Botão "Filtrar"

Filtros Disponíveis:

- Título: Busca publicações que contenham o texto especificado no campo do título.
- Autor: Verifica se o nome do autor informado está presente entre os autores listados no artigo.
- Afiliação: Pesquisa se a afiliação dos autores contém o texto fornecido.
- Data de Publicação: Localiza artigos com data exata no formato YYYY-MM-DD.
- Palavras-chave: Busca publicações contendo palavras-chave relacionadas ao termo digitado.

Funcionamento:

- O utilizador seleciona um filtro e fornece um valor de busca.
- Se o filtro for inválido, o sistema exibe uma mensagem de erro e encerra a função.
- Após aplicar o filtro, os resultados podem ser ordenados por título ou data, utilizando o método `sorted()`.
- Os resultados filtrados e ordenados são exibidos detalhadamente, incluindo:
  - Título
  - Autores
  - Data de publicação
  - Palavras-chave
  - DOI
  - URL
- Caso nenhum resultado seja encontrado, o sistema exibe uma mensagem informando que nenhuma publicação foi localizada.

5.3.4 3.4. Botão Gráficos

Os relatórios estatísticos são exibidos diretamente na interface gráfica, com gráficos gerados dinamicamente através do Matplotlib. Esses gráficos são interativos e podem ser salvos como imagens para utilização futura.

Exemplo de Relatório: A função `grafico_publicacoes_por_ano(artigos)` cria um gráfico de barras apresentando a distribuição das publicações por ano.

Funcionamento:

- Extrai os anos das datas de publicação de todos os artigos.
- Conta a frequência de cada ano utilizando a classe `Counter`.
- Utiliza o Matplotlib para gerar um gráfico de barras:
  - Eixo X: Anos
  - Eixo Y: Número de publicações por ano
- Aplica título e rótulos apropriados para facilitar a interpretação do gráfico.

5.3.5 3.4. Botão Gráficos

O botão **'Gráficos'** gera visualizações interativas para análise estatística dos dados. Os gráficos são criados dinamicamente usando Matplotlib e podem ser salvos como imagens para uso posterior.

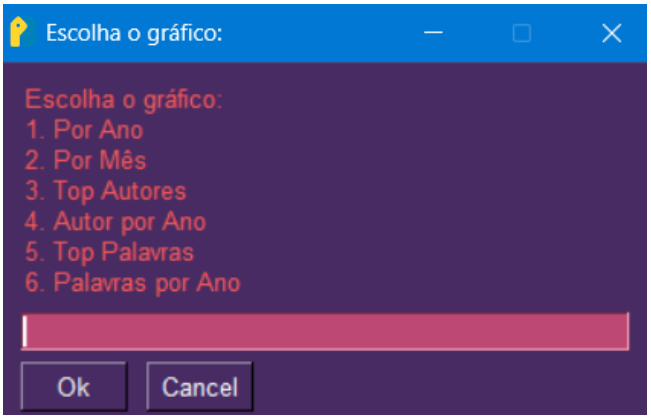

Figure 14: Interface do Botão "Gráficos"

Funções de Gráficos: 1. gráfico_publicacoes_por_ano(artigos)

Cria um gráfico de barras que apresenta a distribuição das publicações por ano. Primeiramente, extrai os anos das datas de publicação de todos os artigos e conta a frequência de cada ano utilizando a classe Counter.

Em seguida, utiliza Matplotlib para gerar um gráfico de barras onde o eixo X representa os anos e o eixo Y mostra o número de publicações por ano. O gráfico é exibido com título e rótulos apropriados para facilitar a interpretação.

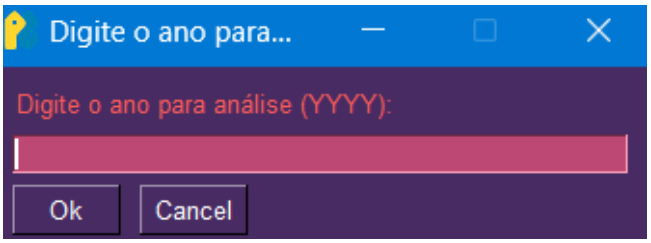

Figure 15: Interface de Seleção do Ano para Análise

2. gráfico_publicacoes_por_mes(artigos)

Gera um gráfico de barras exibindo a distribuição de publicações por mês em um ano específico.

- Solicita ao utilizador que informe o ano desejado para análise.
- Filtra os artigos publicados nesse ano.
- Conta as publicações por mês usando `Counter`.
- Exibe o gráfico com meses no eixo X e quantidade no eixo Y.

3. gráfico_top_autores(artigos) Gera um gráfico horizontal de barras mostrando os 20 autores com mais publicações.

- Coleta os nomes dos autores de todos os artigos.
- Conta as publicações por autor com `Counter`.
- Seleciona os 20 autores mais frequentes.
- Exibe o gráfico com autores no eixo Y e quantidade no eixo X.

4. gráfico_publicacoes_por_autor(artigos) Cria um gráfico de barras exibindo a distribuição das publicações de um autor ao longo dos anos.

- Solicita o nome do autor a ser analisado.

- Filtra os artigos publicados pelo autor.
- Conta as publicações por ano usando `Counter`.
- Exibe o gráfico com anos no eixo X e número de publicações no eixo Y.

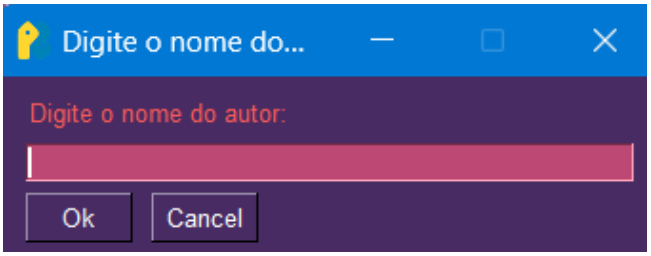

Figure 16: Interface de Seleção do Nome do Autor

5. gráfico_top_palavras_chave(artigos) Gera um gráfico horizontal de barras com as 20 palavras-chave mais frequentes.

- Extrai palavras-chave de todos os artigos.
- Normaliza para letras minúsculas e conta com `Counter`.
- Seleciona as 20 palavras mais usadas.
- Exibe palavras no eixo Y e frequência no eixo X.

6. gráfico_palavras_chave_por_ano(artigos) Cria um gráfico horizontal de barras para as 10 palavras-chave mais frequentes em um ano específico.

- Solicita o ano para análise.
- Filtra artigos desse ano e coleta palavras-chave.
- Conta e classifica os termos com `Counter`.
- Exibe palavras no eixo Y e frequência no eixo X.

5.3.6 3.5. Botão Gravar

O botão **'Gravar'** permite salvar as alterações realizadas nas publicações.

Funcionamento:

- Salva as alterações diretamente no ficheiro JSON de suporte.
- Mensagem de confirmação é exibida após a gravação bem-sucedida.

Exemplo de Mensagem:

```
Alterações gravadas com sucesso!
```

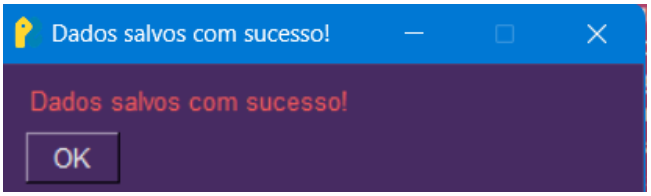

Figure 17: Interface de Confirmação de Gravação dos Dados

5.3.7 3.6. Botão Eliminar

O botão **'Eliminar'** permite remover uma publicação específica do dataset.

Funcionamento:

- Solicita o DOI do artigo a ser eliminado.
- Remove a publicação correspondente do dataset.
- Exibe uma mensagem de confirmação após a remoção.

5.3.8 3.7. Botão Sair

O botão **'Sair'** permite o encerramento seguro da aplicação.

Funcionamento:

- Antes de fechar, o utilizador pode optar por salvar alterações.
- Se nenhuma alteração for feita, o sistema encerra sem modificar o ficheiro.
- Garante que os dados em memória sejam mantidos ou descartados conforme a decisão do utilizador.

6 Conclusão

O sistema desenvolvido procurou atender aos requisitos especificados e demonstrar a aplicação prática de técnicas avançadas de programação utilizando várias funcionalidades da ferramenta Python. A elaboração deste documento culminou numa aplicação com uma funcionalidade simples, mas interativa em que o utilizador pode passar algum tempo a pesquisar e modificar uma base de dados para gestão e análise de publicações científicas, suportando operações básicas e análises avançadas com gráficos interativos.

O projeto também proporcionou experiência na utilização de bibliotecas específicas, como json e matplotlib, e no desenvolvimento de interfaces gráficas, consolidando competências em programação estruturada e orientada a objetos. O desenvolvimento deste projeto revelou-se relevante na nossa formação, permitindo a utilização de todos os conceitos aprendidos ao longo da unidade curricular de uma forma concisa e condensada.