Trabalho Prático 2 - Processamento de Linguagem Natural

Ana Patrícia Costa PG53062, Inês Mendes PG53875, Luís Cunha PG54020

June 11, 2024

1 Introdução

O trabalho prático 2 tem como objetivo principal o desenvolvimento de um sistema capaz de utilizar os dados extraídos de PDFs médicos, conforme realizado no trabalho prático 1, com adição de novas informações obtidas de fontes externas, como websites ou outros dicionários médicos disponíveis. Tendo em consideração o objetivo referido, foi desenvolvida uma ferramenta de visualização dos dados através da framework *Flask* estando estes contidos numa página *web* onde é permitido ao utilizador manipular e visualizar os mesmos.

Relativamente aos dados obtidos no trabalho anterior apresentados na página *web*, estes foram extraídos anteriormente dos PDFs: "glossario_ministerio_saude.pdf", "Minidicionário de Cardiologista.pdf" e "Glossário de Termos Médicos Técnicos e Populares.pdf". Para cada um, foram extraídas as informações necessárias, resultando em cinco ficheiros *JSON*. Para "glossario_ministerio_saude.pdf", foram extraídos "dicionario.json" e "siglas.json", que contêm os termos e descrições adotadas pelo Ministério da Saúde e as siglas, respetivamente. O "Minidicionário de Cardiologista.pdf" consiste num dicionário com os termos mais utilizados pelos cardiologistas em português e inglês, do qual foram extraídos "EN_PT.json" e "PT_EN.json". Por fim, foram extraídos os termos médicos e populares para "texto.json" com base no "Glossário de Termos Médicos Técnicos e Populares.pdf".

2 Processamento de dados

2.1 Primeira junção dos JSON

Numa primeira fase, foram analisados os conceitos e a estrutura dos ficheiros JSON "dicionario.json" e "siglas.json" para compreender melhor os dados neles contidos. Cada um destes ficheiros possui uma estrutura específica que facilita a organização e o acesso às informações:

"dicionario.json": termo: "categoria": categoria, "descricao": descrição

"siglas.json": sigla: significado

Após a análise, foi realizado o processamento de dados dos ficheiros *JSON*, "dicionario.json" e "siglas.json", com a junção das informações de ambos e criação de um novo ficheiro, "dic_novo.json".

De modo a atingir este objetivo, foi criado um script denominado "jsoncreator.py". Este código *Python* começa por abrir os dois ficheiros *JSON*, "dicionario.json" e "siglas.json", em modo de leitura ("r"), e carrega os seus conteúdos em variáveis chamadas dicionario_e siglas_, respectivamente, utilizando a função "*json.load()*".

Depois disso, obteve-se as chaves de ambos os dicionários carregados e armazenou-se nas variáveis "dic_keys" e "sig_keys". De seguida, é executado um ciclo que itera sobre cada termo no dicionário "dicionario_". Para cada termo, é adicionada uma nova chave "siglas" e inicializado um dicionário vazio sob esta. De seguida, faz-se uma iteração sob cada chave do dicionário siglas_, e se uma dessas chaves estiver presente no termo ou na sua descrição, ela é adicionada ao dicionário "siglas" do termo em questão.

Após atualizar todos os termos com as siglas relevantes, é criado um novo arquivo JSON chamado "dic_novo.json" em modo de escrita ("w"). O conteúdo do dicionário modificado (dicionario_) é escrito neste novo ficheiro utilizando a função "*json.dump()*".


```
"GPSM": "Gestão Plena do Sistema Municipal",
"HIV": "Vírus da Imunodeficiência Humana",
"HOSPUB": "Sistema de Gerenciamento de Unidade Hospitalar",
"IAPI": "Incentivo de Apoio e Diagnóstico Ambulatorial e Hospitalar à População Indígena",
"IBAM": "Instituto Brasileiro de Administração Municipal",
```


Figure 1: Excerto do dicionário "siglas.json"


```
"Anti-retroviral": {
    "categoria": "Medicamentos, Vacinas e Insumos",
    "descricao": "Denominação genérica para os medicamentos atualmente utilizados no tratamento da infecção pelo HIV, que é um retrovírus."
},
```


Figure 2: Excerto do dicionário "dicionario.json"


```
"Anti-retroviral": {
    "categoria": "Medicamentos, Vacinas e Insumos",
    "descricao": "Denominação genérica para os medicamentos atualmente utilizados no tratamento da infecção pelo HIV, que é um retrovírus.",
    "siglas": {
        "HIV": "Vírus da Imunodeficiência Humana"
    }
},
```


Figure 3: Excerto do dicionário "dicti_novo.json"

2.2 Web Scrapping

Para enriquecer os dados, foram extraídos e processados informação de um glossário médico utilizando *Web Scrapping*.[1]

O glossário é um dicionário médico online que apresenta conceitos ordenados alfabeticamente com descrições e, em alguns casos, são também apresentados sinónimos e artigos relacionados. Para extrair as informações, foi criado um *script Python* denominado "web_scrapping.py" e foram importadas as bibliotecas necessárias: "*requests*" para fazer requisições *HTTP* e obter o conteúdo *HTML* da página, "*BeautifulSoup*" para analisar e manipular o *HTML* obtido, "*json*" para guardar os dados extraídos em formato *JSON*, e "*re*" para operações com expressões regulares, necessárias para limpeza e manipulação de texto.

A seguir foi definida a função "*get_content(url)*" para realizar a extração dos dados. Inicialmente, é feito um pedido *HTTP* para a *URL*, obtendo-se o conteúdo *HTML* da página. Em seguida, utiliza-se "*BeautifulSoup*" para analisar o *HTML* e encontrar todas as divisões (*div*) que contêm os termos do glossário, especificamente as divisões com a classe "*kt-tab-inner-content-inner*". O conteúdo é então dividido em partes menores relativas a cada letra do alfabeto utilizando o separador "*hr class="wp-block-separator has-alpha-channel-opacity"*", resultando numa lista de listas, sendo cada elemento destas relativo a um termo. Cada termo é então processado fazendo a divisão pela quebra de linha ("/n") de modo a obter elementos como o termo, a descrição, sinónimos e artigos. Também são removidos elementos *HTML* desnecessários como *div*, *ul* e *ol* garantindo assim que apenas o texto relevante seja mantido.

Para cada termo, o código extrai a descrição, sinónimos e artigos relacionados. Esta extração é feita utilizando *BeautifulSoup* para identificar elementos específicos: o termo é identificado pela *tag strong*, a descrição pela *tag p*, os sinónimos pela *tag li*, e os artigos relacionados pela *tag a* com um atributo *href*. Os dados são organizados num dicionário onde cada chave é um termo do glossário e o valor é outro dicionário contendo a descrição (desc), sinónimos (sinonimos) e artigos relacionados (artigos relacionados). O dicionário completo é então guardado num ficheiro *JSON* denominado "glossario.json".

2.3 Segunda junção dos JSON

Numa segunda fase, foi aglomerada a informação presente no ficheiro *JSON* extraído do "Glossário de Termos Médicos Técnicos e Populares.pdf" ("texto.json"), numa versão melhorada, através do script "PTENeditor.py", do ficheiro *JSON* extraído do "Minidicionário de Cardiologista.pdf" ("novo_PT_EN.json") e no ficheiro "glossario.json".

Esta foi aglomerada num novo dicionário - "global_dicti" - cujas chaves correspondem ao conjunto da soma das chaves dos dicionários obtidos através do "texto.json", do "novo_PT_EN.json" e do "glossario.json", e os valores um dicionário cujas chaves são "traduc", "desc", "sinon" e "pop". Os valores de cada chave neste dicionário são definidos da seguinte forma: se a chave estiver presente no conjunto de traduções ("novo_PT_EN.json"), o valor correspondente é atribuído à chave "traduc" no dicionário global; se a chave estiver presente no glossário ("glossario.json"), as descrições e sinónimos associados são atribuídos às chaves "desc" e "sinon", respetivamente; se a chave estiver presente nas expressões populares ("texto.json"), o valor correspondente é atribuído à chave "pop"; se uma chave não estiver presente em algum dos conjuntos de dados, os valores correspondentes no dicionário global são definidos como "*None*".

Para tornar mais fácil a análise do resultado desta aglomeração o dicionário "global_dicti" foi armazenado no ficheiro JSON "teste.json".

3 Arquitetura e Funcionalidades

Este projeto envolveu a criação de uma plataforma *web* interativa projetada para facilitar a visualização e manipulação dos termos. Para alcançar este objetivo, foi desenvolvida uma página *web* utilizando o *framework Flask*. Para garantir uma aplicação coesa e dinâmica, foram implementados múltiplos *templates* com a ajuda do mecanismo de modelo *Jinja*.

Na aplicação *web*, definiu-se o *template* inicial como "*Home*", que serve de página principal do sistema. Este sistema possui uma barra lateral à esquerda que permite a navegação entre diferentes páginas. A barra lateral inclui várias opções que direcionam o utilizador para as páginas desejadas, nomeadamente "*Home*", "Conceitos" e "Categorias". A opção "Conceitos" apresenta todos os conceitos do dicionário, enquanto "Categorias" abre um subtemplate com todas as categorias disponíveis, onde cada categoria contém uma lista de termos associados.

Ao escolher a opção "Conceitos" no menu lateral, o utilizador é direcionado para uma página de listagem de conceitos onde dispõe de diversas funcionalidades. É possível adicionar um novo conceito, pesquisar por um conceito e explorar a lista com todos os termos disponíveis. Na lista de termos, é possível selecionar o termo desejado, que abrirá a página do conceito escolhido. A estrutura desta página inclui o conceito como título, seguido da sua descrição, e três botões: "Pesquisar no *Google*", "Editar conceito" e "Excluir conceito". Adicionalmente, apresentam-se *links* de artigos relacionados com os termos da pesquisa presentes no "Glossário MD Saúde" e links de artigos do *PubMed* relacionados com o termo obtidos através de *web scrapping*.

Além destas funcionalidades, algumas palavras na descrição estão sublinhadas e quando é feito o *hover* em cima da mesma é apresentado uma caixa com pelo menos uma das seguintes: tradução em inglês, expressão popular, sinónimos e descrição adicionais.

O botão "Pesquisar no *Google*" redireciona o utilizador para a página do *Google* com os resultados da pesquisa do termo em questão, mostrando o termo na barra de pesquisa e os resultados fornecidos pelo motor de busca. Para além desta funcionalidade, existem mais duas: editar conceito e eliminar conceito. Quando se clica no botão "Editar conceito", o utilizador é redirecionado para uma página que permite visualizar o termo e a descrição atual, sendo possível alterá-la conforme necessário. Ao pressionar o botão "Excluir conceito", o sistema exibe uma caixa de confirmação perguntando ao utilizador se deseja realmente eliminar o conceito. Após a confirmação, o conceito é removido da lista de termos.

Também se verificou que alguns conceitos na sua descrição indicavam a necessidade de consultar outro conceito para obter informações completas. Assim, o utilizador pode clicar no conceito mencionado na descrição e será redirecionado para a página do conceito correspondente.

Ao aceder a "Categorias", o sistema abrirá uma lista com todas as categorias na barra lateral. Após a escolha de uma destas, será aberta uma página referente à categoria selecionada e uma lista de conceitos pertencentes a essa categoria.

4 Plataforma Web

Como anteriormente mencionado, de modo a implementar a plataforma web pretendida com todas as suas funcionalidades foi usada a *framework Flask*. Esta centra-se num *script* principal "*app.py*" e diversos *templates*.

O coração da plataforma é o ficheiro "*app.py*", que funciona como o cérebro por trás de todas as operações realizadas. Desde o carregamento dos dados até à manipulação das requisições *HTTP*, este é responsável por orquestrar todas as interações entre o utilizador e a aplicação.

A arquitetura da plataforma é composta por diversos elementos-chave, incluindo *templates HTML*, rotas definidas e funcionalidades específicas. Os *templates* fornecem a estrutura visual da plataforma, permitindo que os utilizadores interajam de forma amigável e intuitiva. As rotas, por sua vez, direcionam as solicitações do utilizador para as funções apropriadas, garantindo uma experiência fluida e eficiente.

4.1 app.py

O arquivo `app.py` contém o código do servidor *Flask* que implementa as funcionalidades da plataforma *web*. Aqui são geridos o carregamento dos dados, a definição das rotas para interação com o utilizador e a manipulação das requisições *HTTP*.

Inicialmente, são importadas as diversas bibliotecas necessárias:

- Flask: *Framework* web utilizado para criar a aplicação.
- json: Biblioteca para manipulação de dados em formato *JSON*.
- os: Biblioteca para interações com o sistema operacional.
- re: Biblioteca para operações com expressões regulares.
- urllib.parse.quote_plus: Função para codificar *URLs*.
- BeautifulSoup: Biblioteca para scraping de dados *HTML*.
- requests: Biblioteca para fazer requisições *HTTP*.

De seguida, são carregados os dados de vários ficheiros *JSON* que contêm os termos médicos (dic_novo.json), traduções (novo_PT_EN.json), glossário (glossario.json), termos populares (text.json) e siglas (siglas.json) para as variáveis "termos", "traduc", "gloss", "popular" e "siglas", respetivamente. Depois, são extraídas as categorias e armazenadas numa lista única e ordenada chamada "categorias".

Neste código existem duas funções auxiliares, uma denominada "guardar_termos" para guardar os termos no ficheiro "dic_novo.json" e outra denominada "google_url" para construir um *URL* de pesquisa no *Google*.

Quanto às rotas, estas são fundamentais para definir como a aplicação responde a diferentes solicitações *HTTP* feitas pelo usuário e são usadas para mapear *URLs* específicas para funções *Python* que processam as solicitações e retornam respostas apropriadas. Cada rota está associada a um endpoint específico da aplicação. Na aplicação desenvolvida foram criadas as seguintes rotas:

- Rota principal ("/"): Esta rota é a página inicial da aplicação, ou seja, quando o utilizador acede ao site, a função "*home()*" é executada. Esta função renderiza a página "*home.html*" e passa a lista de categorias como contexto para o template.
- Termos por categoria ("/<string:categoria>"): esta rota diz respeito à exibição de termos por categoria, ou seja, quando o utilizador seleciona uma categoria no menu lateral é executada a função desta rota, "termos_categoria(categoria)". Primeiramente é feita a substituição da expressão "%20" por um espaço real na string categoria. Isto é necessário porque os espaços nas *URLs* são representados como "%20", e essa substituição garante que a categoria seja interpretada corretamente. Em seguida, são verificados todos os termos na variável "termos" e aqueles que pertencerem à categoria especificada são adicionados a uma lista denominada "termos_cat". Após filtrar os termos pela categoria, a função renderiza a página *HTML* "categoria.html", passando as listas "termos_cat" e "categorias" como contexto.

- Informações por termo ("/conceito/<string:termo>"): esta rota é referente à exibição das informações de um termo, ou seja, quando o utilizador seleciona um termo para visualizar é executada a função desta rota, "termo_info(termo)". Primeiramente, a função substitui a expressão "%20" por um espaço real na *string* do termo, garantindo que o termo seja interpretado corretamente. Em seguida, é verificado se o termo existe no dicionário "termos". Caso exista, segue-se a parte do código para a obtenção dos artigos do *PubMed* referentes ao termo e a renderização da página *HTML* "conceito.html", passando como contexto o termo, as listas "categorias" e "gloss", o valor obtido da função "google_url(termo)", o valor de "termos[termo]" a lista de URLs dos artigos do *PubMed* e o dicionário "global_dicti" que agrega toda a informação dos dicionários "traduc", "gloss" e "popular". Caso não exista, é criada uma mensagem de erro e é renderizado o template "erro.html", passando como contexto a mensagem e a lista "categorias". Para a obtenção dos *URLs* dos artigos do *PubMed* é seguida a seguinte lógica: se o termo consiste numa única palavra, a função cria uma *URL* simples para pesquisar o termo no *PubMed*; se o termo consistir em múltiplas palavras, as palavras são concatenadas com sinais de mais (+) entre elas, criando uma *URL* apropriada para a pesquisa, a função então faz uma solicitação *HTTP* para a *URL* do *PubMed*, obtém o *HTML* da página e usa *BeautifulSoup* para o analisar; é feita uma procura por todos os elementos "*a*" com a classe "*docsum-title*", que representam os títulos dos artigos; para cada artigo encontrado, a função obtém o *link* e o título do artigo, construindo um dicionário denominado "*art_pubmed*" onde as chaves são os nomes dos artigos e os valores são os *URLs* completos.

- Pesquisa e listagem de conceitos ("/Conceitos"): esta rota diz respeito à exibição de conceitos, quer sejam todos ou o resultado de uma pesquisa, ou seja, quando o utilizador apenas seleciona a opção "Conceitos" no menu lateral ou seleciona e faz uma pesquisa, é executada a função desta rota, "conceitos()". Primeiro, a função tenta obter o parâmetro de pesquisa "search" a partir dos parâmetros de consulta na *URL* usando "*request.args.get('search')*". Se um parâmetro de pesquisa for fornecido, é criada uma mensagem indicando que os resultados da pesquisa serão exibidos e é feita a filtragem dos termos que contêm a string de pesquisa. Para fazer isso, a função utiliza um dicionário em compreensão que verifica se a *string* de pesquisa, convertida para minúsculas, está presente no nome do termo ou na descrição. Se a pesquisa encontrar resultados, a função renderiza o template "Conceitos.html", passando os resultados filtrados, a lista de categorias e a mensagem de pesquisa como contexto. Se nenhum parâmetro de pesquisa for fornecido, a função simplesmente renderiza o template "Conceitos.html", passando todos os termos do dicionário, a lista de categorias e uma mensagem vazia como contexto.

- Adição de conceitos ("/Conceitos", methods = ["POST"]): esta rota é referente à adição de um novo conceito, ou seja, quando o utilizador está na página "/Conceitos" e pretende adicionar um novo, é executada a função desta rota, "adicionar_conceitos()". A lógica seguida é a seguinte: a função utiliza "*request.form.get()*" para obter os dados enviados pelo formulário *HTML*, incluindo o novo termo, sua descrição e categoria; é verificado se o termo já existe no dicionário "termos"; se o termo já estiver presente, a função cria uma mensagem informando que o termo não pode ser adicionado novamente e renderiza a página "Conceitos.html", passando os termos existentes, as categorias e a mensagem como contexto; se o termo não existir, a função adiciona o novo termo ao dicionário "termos" com a sua categoria, descrição e siglas correspondentes; de seguida, chama a função "guardar_termos()" para guardar as alterações no dicionário; por fim, cria uma mensagem indicando que o termo foi adicionado com sucesso e renderiza a página "Conceitos.html", passando os termos atualizados, as categorias e a mensagem como contexto.

- Exclusão de conceitos ("/excluir_conceito/<string:termo>", methods = ["POST"]): esta rota diz respeito à eliminação de conceitos, ou seja, quando o utilizador pretende eliminar um conceito, é executada a função desta rota, "excluir_conceito(termo)". A função começa por verificar se o termo a ser excluído está presente no dicionário "termos". Se existir, ele é excluído usando "*del* termos[termo]". Em seguida, é chamada a função "guardar_termos()" para guardar as alterações no dicionário e é cria uma mensagem informando que o conceito foi excluído com sucesso. Se o termo não existir no dicionário, a função cria uma mensagem informando que o conceito não foi encontrado e não pode ser excluído. Em ambos os casos, a função renderiza a página "Conceitos.html", passando os termos atualizados, as categorias e a mensagem correspondente à exclusão do conceito como contexto.

- Página de edição do conceito ("/editar_conceito/<string:termo>"): esta rota é referente à página de edição de um determinado conceito, ou seja, quando o utilizador pretender alterar a descrição de um determinado conceito, é executada a função desta rota, "editar_conceito_pagina(termo)". A função verifica se o termo a ser editado está presente no dicionário "termos". Se o termo existir no dicionário, a função renderiza a página "editar_conceito.html", passando o título da página como o nome do termo, as informações detalhadas do conceito (conceito_info) e a lista de categorias disponíveis como contexto. Se o termo não existir no dicionário, a função cria uma mensagem informando que o conceito não foi encontrado e renderiza a página "erro.html", passando essa mensagem como contexto.

- Edição do conceito ("/editar_conceito/<string:termo>", methods = ["POST"]): esta rota trata das solicitações "*POST*" para a edição de um termo, ou seja, é responsável por efetivamente editar a descrição do conceito. A função "editar_conceito(termo)" obtém a nova descrição do conceito a partir do formulário *HTML* usando "*resquest.form.get()*". Se o termo existir no dicionário "termos", a descrição do termo é atualizada com a nova descrição e as alterações são guardadas no dicionário utilizando a função "guardar_termos()". De seguida, é feita a mesma lógica usada na rota ("/conceito/<string:termo>") para obter os artigos do *PubMed*. Por fim, a função renderiza a página "conceito.html" com as informações atualizadas do termo, incluindo a sua descrição editada, as informações detalhadas do termo, a lista de categorias, traduções, glossário, a função para gerar *URLs* de pesquisa no *Google*, expressões populares e o dicionário de artigos do *PubMed*.

4.2 Templates

4.2.1 parent.html

Este ficheirp *HTML* é o modelo base da interface. Nele, foi criada uma barra lateral colapsável à esquerda na página *Web* com três secções: "Home", que conduz o utilizador à página inicial; "Conceitos", que leva o utilizador à página contendo todos os conceitos, assim como operações de pesquisa e de adição de conceitos; e "Categoria", que revela uma nova lista na barra lateral contendo todas as categorias quando clicada, ao clicar em qualquer categoria o utilizador é subsequentemente redirecionado para a página contendo todos os termos dessa categoria.

4.2.2 home.html

Neste ficheiro, é possível encontrar apenas a barra lateral e, comum a todos os restantes *templates*, alguma informação básica, tendo a função de "capa" da nossa página.

4.2.3 Conceitos.html

Esta página permite a listagem de conceitos, total ou parcial, assim como várias funcionalidades essenciais para a página Web.

- A primeira funcionalidade permite que o utilizador adicione um novo conceito. Para isso, há um formulário que envia uma requisição *POST* para o *URL* /Conceitos, contendo campos para o termo, descrição e categoria do novo conceito e um botão de submissão permite enviar o formulário para adicionar o novo conceito.. A categoria é preenchida dinamicamente a partir de uma lista de categorias disponíveis.

- A segunda funcionalidade é a pesquisa de conceitos existentes. É disponibilizado um formulário, que envia uma requisição *GET* para o *URL* /Conceitos, e permite que o utilizador escreva termos de pesquisa num campo de texto. O botão de pesquisa envia o formulário, procurando os conceitos que correspondem ao termo inserido. O botão de limpeza envia uma requisição *GET*, que redefine a pesquisa e mostrando todos os conceitos disponíveis novamente.

Além disso, quando alguma operação de pesquisa ou adição de conceito é realizada com sucesso, é mostrada uma mensagem de acordo com os resultados da pesquisa ou conceito adicionado, de modo a proporcionar feedback.

Por fim, uma lista de conceitos existentes é exibida dinamicamente. Cada conceito é apresentado como um link que aponta para um *URL* específico, detalhando o conceito selecionado. Estas funcionalidades combinadas oferecem uma interface completa, permitindo que os utilizadores adicionem novos conceitos, pesquisem e visualizem conceitos existentes, e recebam *feedback* informativo sobre as suas ações.

4.2.4 conceito.html

Esta página dedica-se à apresentação de informação sobre um determinado conceito, nomeadamente a sua descrição, as siglas presentes no texto, um botão de pesquisa no *Google*, botões de edição e exclusão de conceitos e uma secção dedicada a artigos potencialmente úteis relacionados com o termo em questão.

O bloco *body* começa com um *script JavaScript* embutido. Primeiro, quando a página é completamente carregada, o *script* começa a procurar por todos os parágrafos (elementos *p*) na página. Para cada parágrafo encontrado, verifica se há palavras-chave específicas. No caso da existência de palavras-chave (palavras presentes no dicionário "global_dicti"), transforma essas mesmas palavras em âncoras *hoverable*, adicionando informações, como traduções, expressões populares, sinónimos e descrições, quando estas assim existem, aquando da colocação do rato sobre elas.

A página começa por exibir o conceito e a respetiva descrição, armazenada na variável "conceito_info" (passada como contexto através do valor de "termos[termo]"). Se a descrição começar com "Ver", isso indica que a descrição é na verdade uma referência a outro conceito. Nesse caso, o conceito em questão é tornado clicável e direciona para a página de informações respetiva. Se a descrição não começar com "Ver", a descrição completa é exibida na página.

A página inclui também um botão para pesquisar o conceito no *Google*, o que permite ao usuário obter mais informações sobre o conceito na *web*. Além disto, há um botão que direciona o utilizador para a página de edição do conceito, permitindo a atualização das informações do mesmo, se necessário, e também um botão para excluir o conceito, que permite remover o conceito diretamente da interface, que necessita de uma confirmação por parte do utilizador para que seja efetivamente eliminado. Isto permite diminuir a chance de erros ou acidentes na eliminação de conceitos.

De seguida, se houver siglas associadas ao conceito, elas são listadas na página, fornecendo mais informações sobre o conceito. Também são exibidos artigos relacionados encontrados no "Glossário - Dicionário Médico" do MD.Saúde. Para isso, é verificado se os termos presentes na descrição do conceito correspondem a termos no glossário. Se houver correspondências, os artigos relacionados são listados com *links* diretos para esses artigos. O mesmo acontece para os artigos do *PubMed*.

4.2.5 editar_conceito.html

Esta é a página *HTML* para a edição de conceitos. Primeiro é exibido o conceito a editar e de seguida existe um formulário com um campo para inserir uma nova descrição do conceito. O valor desse campo é preenchido automaticamente com a descrição atual do conceito, permitindo que o utilizador a edite facilmente. Por fim, é apresentado o botão "Guardar" que, quando clicado, envia os dados do formulário para a rota de edição correspondente.

4.2.6 categoria.html

Este template diz respeito à listagem de termos de uma determinada categoria, ou seja, quando o utilizador escolhe uma categoria do menu lateral, é direcionado para esta página. Aqui, se o título da categoria for diferente de "Outro", é exibido o título da categoria no topo da página. Caso contrário, se for "Outro", é exibido "Outros Termos". Isto permite uma diferenciação visual entre diferentes tipos de categorias. Abaixo do título da categoria, há a lista de termos respetivos. Cada termo é exibido como um *link* clicável que leva à página de detalhes do mesmo.

5 Demonstração/Caso de Estudo

Para demonstrar o funcionamento da aplicação, foram observados dois exemplos de conceitos que foram pesquisados na aplicação: "ruído" e "doenças falciformes". Aquando da abertura da aplicação

na página inicial, foi utilizada a barra lateral e foi selecionada a opção "Conceitos". Na lista de conceitos, foi utilizado o motor de busca para encontrar os conceitos desejados.

Primeiramente, na barra de pesquisa, foi inserida a palavra "ruído" o que resultou no retorno de dois resultados. Dos dois resultados obtidos, foi pressionado o termo "Ruído" para redirecionar à página correspondente. Na figura 4, pode-se observar o resultado do redirecionamento anterior. Nesta é possível observar o conceito "Ruído" como título, a descrição correspondente, com algumas palavras sublinhadas e de cor azul, botão para pesquisar no *Google*, lista dos *links* dos artigos relacionados e botões para editar ou eliminar o conceito.

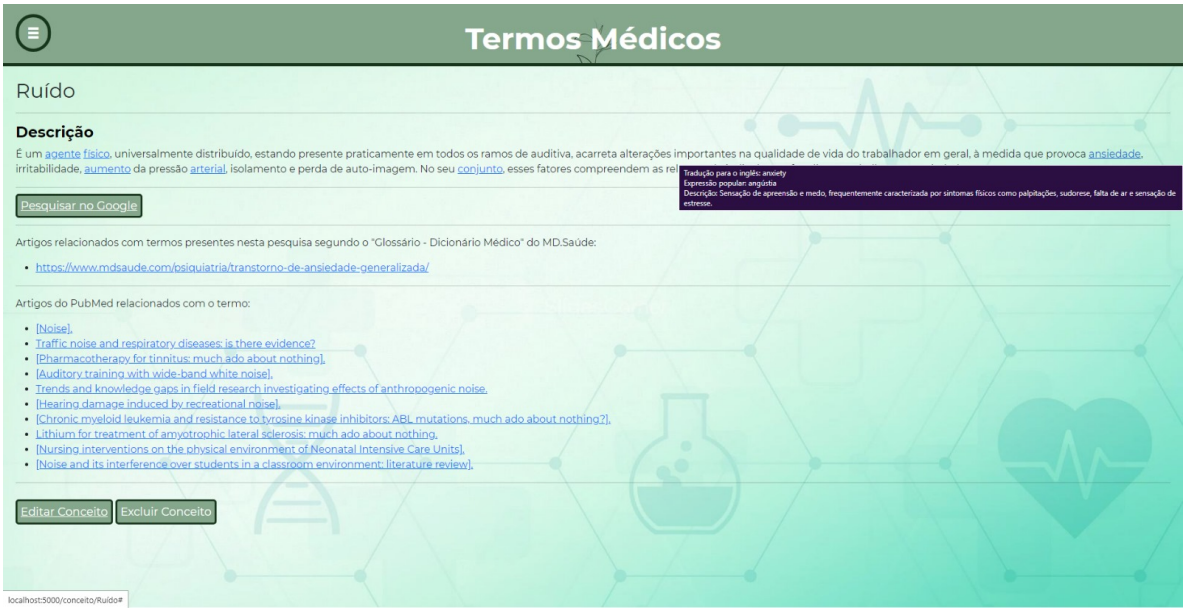

Figure 4: Demonstração da pesquisa do conceito ruído

Como anteriormente referido, a descrição do conceito "Ruído" possui uma série de palavras sublinhadas. Nestas, quando se efetua a ação de *hovering* com o cursor, é revelada uma ou mais informações presentes numa etiqueta. Dentro destas informações podem estar: a tradução para inglês, expressões populares, sinónimos e/ou uma descrição da expressão. Na descrição do conceito pesquisado, há uma palavra que é particularmente interessante de analisar - "ansiedade". Nesta palavra, quando realizado o *hover* com o cursor, são apresentadas três informações: a tradução para o inglês, a expressão popular e a descrição da palavra, sendo um bom exemplo para demonstrar a pertinência da realização da segunda junção dos *JSON* (visto que este resultado significa a presença desta palavra como chave em todos os dicionários pré-aglomeração).

O processo anterior, foi seguidamente repetido de modo a encontrar o conceito de "Doenças falciformes" (Figura 5). A estrutura da página deste é igual à do conceito pesquisado em primeiro lugar, esta é, no entanto, utilizada como exemplo para exibir algo não presente na página resultante da pesquisa anterior. Na descrição de "Doenças falciformes", existe também uma expressão hoverable interessante de menção - "acidente vascular cerebral". Na etiqueta relativa a esta expressão é possível verificar a presença da única informação ausente na etiqueta da expressão "ansiedade", o sinónimo.

Figure 5: Demonstração da pesquisa do conceito Doenças falciformes

Ambos os exemplos explorados são apropriados para demonstrar a funcionalidade do nosso sistema, assim como apresentar uma justificação para a parte final do processamento de dados efetuado.

6 Conclusão

A execução deste projeto destaca a implementação bem-sucedida de uma plataforma *web* interativa para visualização e manipulação de conceitos médicos. Desde a extração de dados de diversas fontes, incluindo *PDFs* médicos e *web scraping* do glossário *online*, até ao desenvolvimento de funcionalidades como adição, edição, exclusão de conceitos e obtenção de artigos do *PubMed*, o projeto demonstrou uma integração abrangente de várias técnicas e tecnologias.

A arquitetura da aplicação, centrada na *framework Flask*, facilitou a organização e a gestão das diferentes partes do sistema, incluindo rotas, manipulação de dados e interação com o utilizador. Os *templates HTML* concebidos com recurso à ferramenta *Jinja* forneceram uma estrutura visual intuitiva, enquanto as rotas definidas mapearam *URLs* específicas para funções *Python* correspondentes, garantindo uma experiência fluida e eficiente.

A demonstração do projeto destacou a sua capacidade de fornecer uma interface intuitiva e informativa. Através da plataforma, os utilizadores podem aceder uma ampla gama de informações médicas de forma rápida e fácil, além de contribuir para a expansão e atualização contínua do dicionário médico.

Em suma, o projeto demonstrou a aplicação eficaz de técnicas de processamento de dados e desenvolvimento *web* de modo a criar uma plataforma útil e acessível para a comunidade médica. A integração de múltiplas fontes de dados e a implementação de funcionalidades avançadas foram realizadas de modo a proporcionar uma experiência abrangente e enriquecedora para os utilizadores.

7 Referências

[1] MD.Saúde Glossário, Disponível em: https://www.mdsaude.com/glossario/.