Venícius Garcia Rego
Matrícula: 2212638

Projeto Final de Programação: Spotter - Uma ferramenta de coleta de comentários em redes sociais para construção de bases de dados para tarefas de NLP

Rio de Janeiro, Brasil

2023

Venícius Garcia Rego
Matrícula: 2212638

Projeto Final de Programação: Spotter - Uma ferramenta de coleta de comentários em redes sociais para construção de bases de dados para tarefas de NLP

Trabalho apresentado ao coordenador do programa de pós-graduação em informática da PUC-Rio como requisito para obtenção de nota na disciplina INF2102-Projeto Final de Programação

Pontifícia Universidade Católica do Rio de Janeiro

Departamento de Informática

Programa de Pós-Graduação em Informática

Orientador: Hélio Côrtes Vieira Lopes

Rio de Janeiro, Brasil

2023

Sumário

Sumário 2

1 ESPECIFICAÇÃO DO PROGRAMA 3
1.1 Objetivo 3
1.2 Escopo 3
1.3 Requisitos 3
1.3.1 Requisitos Funcionais 3
1.3.2 Requisitos Não-Funcionais 4

2 ARQUITETURA 5
2.1 Controller 6
2.2 Database 6
2.3 Interface APIs 8
2.4 Interface Qt 8
2.5 Diagramas 9

3 TESTES 11

4 DOCUMENTAÇÃO PARA O USUÁRIO 13
4.1 Instalação 13
4.2 Telas de navegação 13
4.3 Inserindo chaves de acesso 14
4.4 Coletando novos dados 14
4.5 Visualizando os dados de coleta 15
4.6 Mensagens de Erro 18
4.7 Rodando os Testes de unidade 18

1 Especificação do Programa

1.1 Objetivo

O processamento de linguagem natural(PLN) é uma subárea do aprendizado de máquina que é voltada a manipulação, compreensão, e interpretação de dados textuais. Alguns dos modelos atuais desenvolvidos possuem 175 bilhões de parâmetros e são treinados em mais de uma língua, exemplo a terceira geração do modelo Generative Pre-trained Transformer (GPT) desenvolvido pela OpenAI. O GPT-3 foi treinado utilizando diversas fontes como Common Crawl, WebText2, Books2, etc, mas nem sempre há disponibilidade de base de dados abertas para o desenvolvimento desses modelos, além disso a língua utilizada para a construção dessas bases na maioria das vezes é a lingua inglesa, inutilizando-a ou acrescentando uma nova etapa de tradução se a língua utilizada for diferente.

Pensando nisso, essa ferramenta tem como objetivo permitir ao usuário gerar uma base a partir de comentários disponíveis nas redes sociais através de suas APIs e armazená-las em um banco de dados, sendo possível filtrar os comentários por palavras-chaves ou tópicos.

1.2 Escopo

O escopo do projeto é definido como a produção de uma interface gráfica para a confecção de novas base de dados baseadas em comentários disponíveis na internet, facilitado o processo de treinamento de modelos de PLN.

1.3 Requisitos

Antes do processo de desenvolvimento foram elicitados alguns requisitos da ferramenta divididos em funcionais e não-funcionais.

1.3.1 Requisitos Funcionais

Os Requisitos Funcionais (RF) elicitados foram:

- **RF-01**: O sistema deve coletar dados textuais das redes sociais, exemplo, reddit, twitter, etc.
- **RF-02**: O sistema deve permitir escolher de qual rede social os dados devem ser coletados.

- **RF-03**: O sistema deve ser capaz de armazenar os dados coletados em um banco de dados.
- **RF-04**: O sistema deve permitir ao usuário inserir as chaves e demais dados necessários para autenticação nas APIs das redes sociais.
- **RF-05**: O sistema deve permitir aplicar filtros por palavras-chaves aos dados coletados.
- **RF-06**: O sistema deve permitir aplicar filtros por tópico aos dados coletados.
- **RF-07**: O sistema deve apresentar dados das últimas coletas como os filtros aplicados, fonte dos dados, data, etc.
- **RF-08**: O sistema deve deletar do banco de dados os dados separados por coleta quando solicitado pelo usuário.

1.3.2 Requisitos Não-Funcionais

Os Requisitos Não-Funcionais (RNF) elicitados foram:

- **RNF-01**: Deve-se utilizar bancos de dados não relacionais como MongoDB, etc, para armazenar os dados coletados.
- **RNF-02**: O processo para inserir novas chaves deve ser simples, facilitando a iteração do usuário com a ferramenta.
- **RNF-02**: O processo para selecionar as APIs de onde os dados serão retirados não deve demorar mais de 30 segundos

2 Arquitetura

A interface da ferramenta foi desenvolvida utilizando a biblioteca *Qt* para Python, *PyQt*. O *backend* foi desenvolvido utilizando Python, além de fazer usos de bibliotecas para comunicação com o banco de dados não estruturado MongoDB, *PyMongo*.

A estrutura do programa está dividida em três diretórios o *src*, *view*, *tests*. O diretório *src* possui três subpastas *controller*, onde estão localizados os controladores; *service*, serviço de comunicação com o banco; *interface*, interface para implementação das APIs.

Assim como a *src* o diretório *view* também está dividido em outras subpastas, são elas a *gui*, onde são armazenados alguns dos *widgets* e interface da aplicação; *resources*, onde estão os recursos utilizados para construir a interface, como os ícones por exemplo, etc.

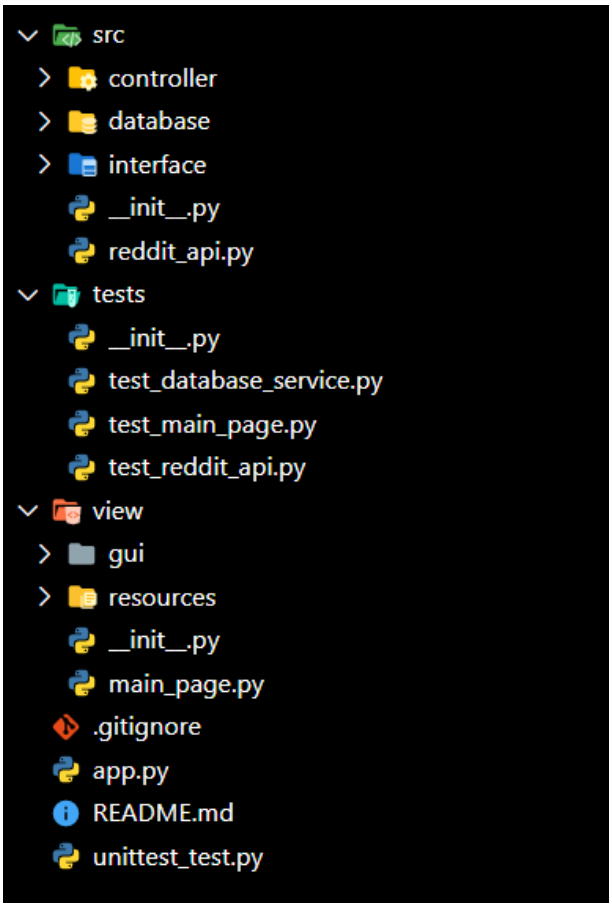

Figura 1 – Estrutura de diretórios do programa.

Por fim o diretório *tests* é onde os testes unitários da ferramenta estão organizados, visite a seção 3 para entender mais sobre os testes desenvolvidos.

2.1 Controller

Os *controllers* ou controladores são classes especializadas na manipulação de dados, neste caso os controladores desenvolvidos são responsáveis por manipular os dados das páginas de coleta e navegação, realizando chamadas ao serviço de banco de dados para inserção, seleção e remoção dos dados.

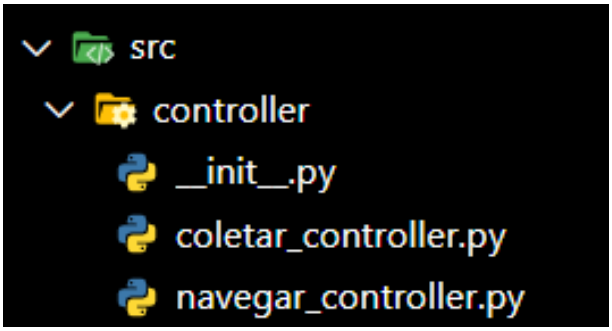

Figura 2 – Controladores desenvolvidos

- *coletar_controller*: contém o script responsável por gerenciar as coletas de novos dados, executando os métodos de coleta das APIs, por palavras-chaves, tópicos, ou ambos, armazenando os resultados no banco de dados.
- *navegar_controller*: contém o script responsável por consumir os dados de coleta do banco de dados e atualizar a página de navegação inserindo ou removendo *widgets* da página.

2.2 Database

A comunicação com o banco de dados é feita através da classe *DatabaseService* utilizando a biblioteca *PyMongo*. A classe cria um banco em um servidor do MongoDB utilizando dados de endereço do *host*, porta, e nome do banco de dados. A *DatabaseService* retorna um serviço sempre que chamada, permitindo que outras classes manipulem os dados registrados.

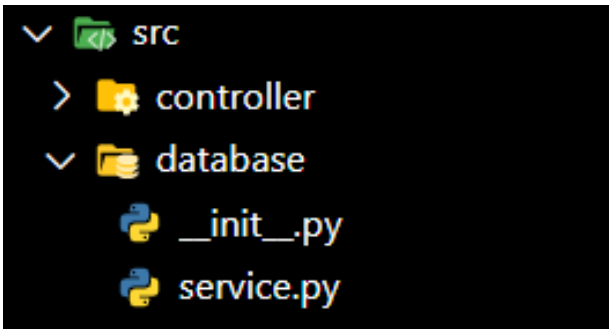

Figura 3 – Serviço de banco de dados desenvolvido

O servidor de banco de dados será criado utilizando as configurações padrão do MongoDB, o endereço do *host* será o *localhost*, e a porta a 27017, o nome do banco de dados será **spotter_base**. Os metadados das coletas serão armazenados na coleção de nome *coleta*, Figura 4, e os comentários na coleção *sentencas*, Figura 5.

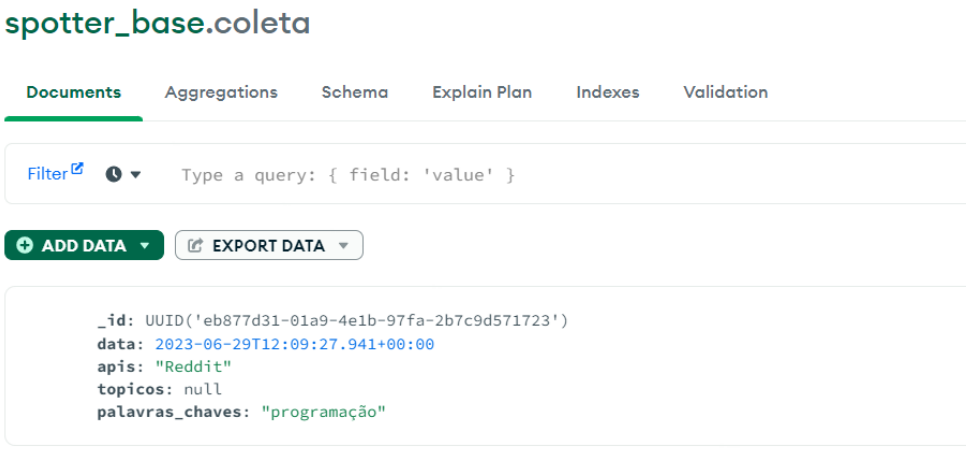

Figura 4 – Exemplo de metadao de coleta registrado no banco

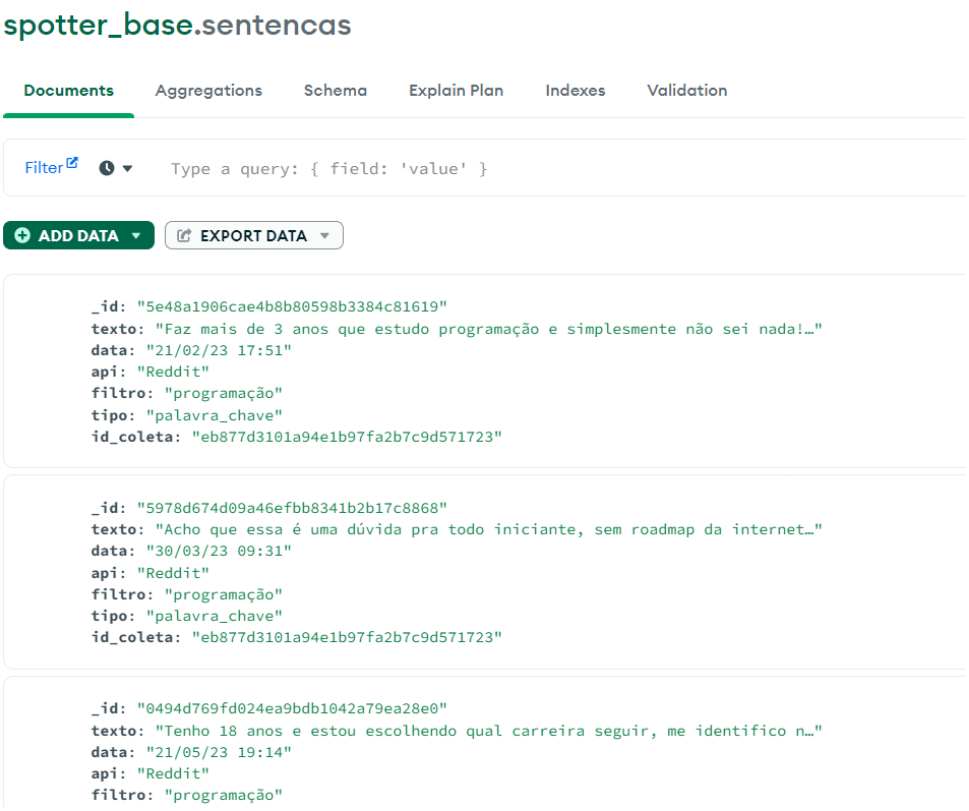

Figura 5 – Exemplo de comentários registrados no banco

2.3 Interface APIs

Cada API disponibilizada pelas plataformas possuem uma arquitetura e um funcionamento diferente, de forma que seria inviável produzir diferentes códigos para cada API selecionada na ferramenta, portanto, foi desenvolvida uma interface que seria implementada por cada uma das APIs, padronizando os métodos, parâmetros enviados e respostas.

Um outro efeito produzido pela criação da interface é a possibilidade de adicionar novas APIs sem a necessidade de reimplementar os controladores ou os serviços do banco, basta acoplar o código da API na aplicação.

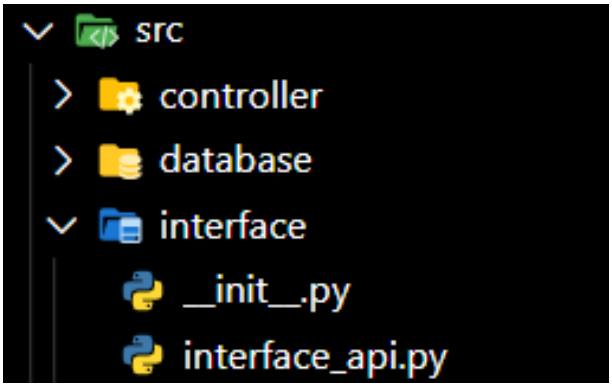

Figura 6 – Interface das APIs desenvolvida

2.4 Interface Qt

A interface *Qt* foi desenvolvida utilizando a ferramenta *designer* da biblioteca do *PyQt-tools*, onde é possível arrastar e posicionar *widgets*, *frames*, *labels*, *buttons*, etc, para dentro da janela, construindo a interface da aplicação dinamicamente. Ao final do processo a interface foi exportada em um arquivo *.ui* para ser carregada no script *main_page*.

Outros componentes próprios foram desenvolvidos para compor a interface da aplicação, como por exemplo o *WidgetNavegarPage*, desenhado para apresentar os dados das coletas registrados no banco de dados.

O script *main_page* faz uso de todas as classes vistas acima, é ele que contém a lógica da aplicação. Nesse script o estilo da interface armazenado no arquivo *.ui* é carregado junto com outros recursos da interface como ícones, além disso as ações que cada botão deve fazer ao ser clicado, a inicialização da thread de coleta de novos dados, atualização da página de navegação, etc.

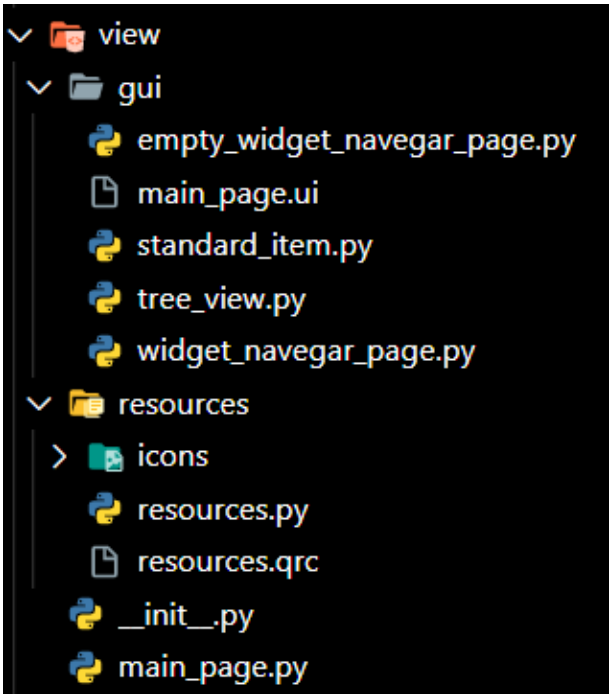

Figura 7 – Componentes da Interface Qt desenvolvidos

2.5 Diagramas

Para este projeto foi desenvolvido um diagrama de atividade, sobre o funcionamento da aplicação, exemplificado na Figura 8.

Durante o uso do sistema o usuário poderá informar as chaves de acesso das APIs que deseja utilizar, em seguida será feita uma conexão utilizando os dados fornecidos, e se desejar o usuário poderá continuar informando novas chaves de acesso para as APIs disponíveis no sistema.

Para construir a base de dados os usuários deverão informar uma ou mais palavras-chaves ou tópicos, logo após serão pesquisadas utilizando as APIs instânciadas. No escopo das palavras-chaves os comentários selecionados são aqueles que possuem a palavra escrita diretamente em seu texto, enquanto na filtragem por tópico os comentários serão coletados a partir de uma identificação prévia do tópico de discussão feito pelas plataformas, portanto, todos os comentários pertencentes ao tópico informado serão retornados.

Ao fim do processo de coleta os comentários serão salvos no banco de dados, além disso os metadados da coleta como data, filtros utilizados, APIs utilizadas, também serão registrados.

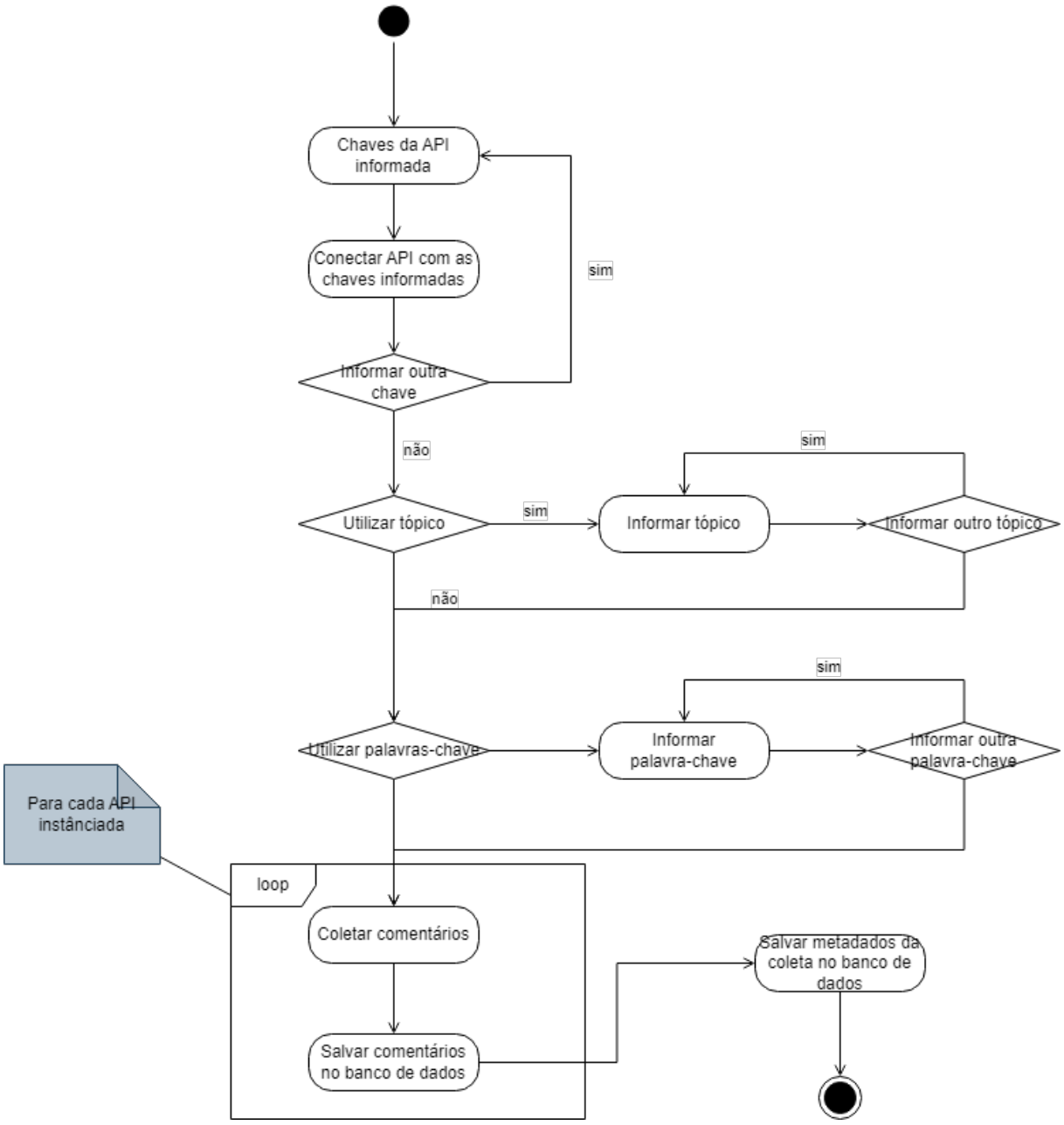

Figura 8 – Diagrama de atividade processamento de item.

3 Testes

A biblioteca utilizada para a realização dos testes foi a *unittest*.[1]. Foram realizados dez testes de unidades para verificar o pleno funcionamento da interface desenvolvida. Estes testes de unidades consistem em verificar o conteúdo das páginas de coleta, navegação e das APIs e se possuem o valor correto, verificar o menu lateral, o funcionamento dos botões de paginação do menu lateral, conexão com o banco de dados e por fim a coleta de dados utilizando a API do Reddit, com filtro de palavras-chaves e tópicos.

Uma descrição dos testes desenvolvidos pode ser encontrada abaixo:

- **TestMainPage**
  - *test_labels_coletar_page*: Teste unitário para verificar o conteúdo da página de coleta, suas *labels* e os valores contido nelas.
  - *test_labels_api_page*: Teste unitário para verificar o conteúdo da página de APIs, suas *labels* e os valores contido nelas.
  - *test_slide_menu*: Teste unitário para verificar a expansão do menu lateral.
  - *test_slide_menu_click_coleta*: Teste unitário para verificar a paginação do botão Coleta do menu lateral.
  - *test_slide_menu_click_navegar*: Teste unitário para verificar a paginação do botão Navegar do menu lateral.
  - *test_slide_menu_click_api*: Teste unitário para verificar a paginação do botão API do menu lateral.
  - *test_slide_menu_click_info*: Teste unitário para verificar a paginação do botão Info do menu lateral.
- **TestDatabaseService**
  - *test_collection*: Teste unitário para criação da collection do banco de dados.
- **TestRedditAPI**
  - *test_login*: Teste unitário para verificar login na API do Reddit.
  - *test_get_topico*: Teste unitário para verificar coleta por tópico na API do Reddit.
  - *test_get_palavra_chave*: Teste unitário para verificar coleta por palavra-chave na API do Reddit.

[1] https://docs.python.org/3/library/unittest.html

O código dos testes de unidade está disponível no repositório do programa[2].

[2] https://github.com/venigarcia/spotter_pfp

4 Documentação para o Usuário

4.1 Instalação

Para utilizar a aplicação, o usuário deverá instalar algumas dependências. É necessário ter instalado o Python $\geq$ 3.10, disponibilizado para download na página oficial do Python[1], o Git[2] que será utilizado para baixar o código fonte da aplicação e por fim o MongoDB, seguindo o manual de instalação disponível no site do sistema[3].

Após isso, o usuário deve-rá realizar o download do código-fonte do projeto, contido em um repositório GitHub[4], utilizando a instrução abaixo em um terminal de comando.

```
$ git clone https://github.com/venigarcia/spotter_pfp.git
```

Após a execução do comando acima surgirá uma pasta de nome *spotter_pfp* na raíz do diretório de trabalho usuado no terminal. O usuário deverá caminhar através do terminal até a pasta criada para a instalação das dependências de bibliotecas do Python. Execute a instrução abaixo para instalar as bibliotecas:

```
$ pip install -r requirements.txt
```

Finalmente, para executar a aplicação deverá executar o comando abaixo na raíz da pasta *spotter_pfp*

```
$ python app.py
```

4.2 Telas de navegação

A interface desenvolvida possui quatro telas, a de coleta de novos dados, onde é possível inserir as palavras-chaves e tópicos que serão utilizados para filtrar os comentários; a tela de API onde as chaves de acesso às APIs são cadastradas e também onde é possível selecionar as APIs que serão utilizadas na coleta; a tela de navegação, apresenta um histórico das coletas realizadas, informando o código da coleta, a data de realização, APIs e filtros utilizados; e a tela de informações adicionais.

[1] https://www.python.org/downloads/
[2] https://git-scm.com/downloads
[3] https://www.mongodb.com/docs/manual/installation/
[4] https://github.com/

4.3 Inserindo chaves de acesso

Primeiramente, no menu lateral clique no botão *APIs* e será redirecionado para a página de APIs, como exemplificado pela Figura 9. No centro da página existem alguns campos marcados com as *labels* usuário, senha, API Key e secret key, preencha os campos com seus dados e selecione o botão à direta dos campos para adicionar a API na coleta, como na Figura 10.

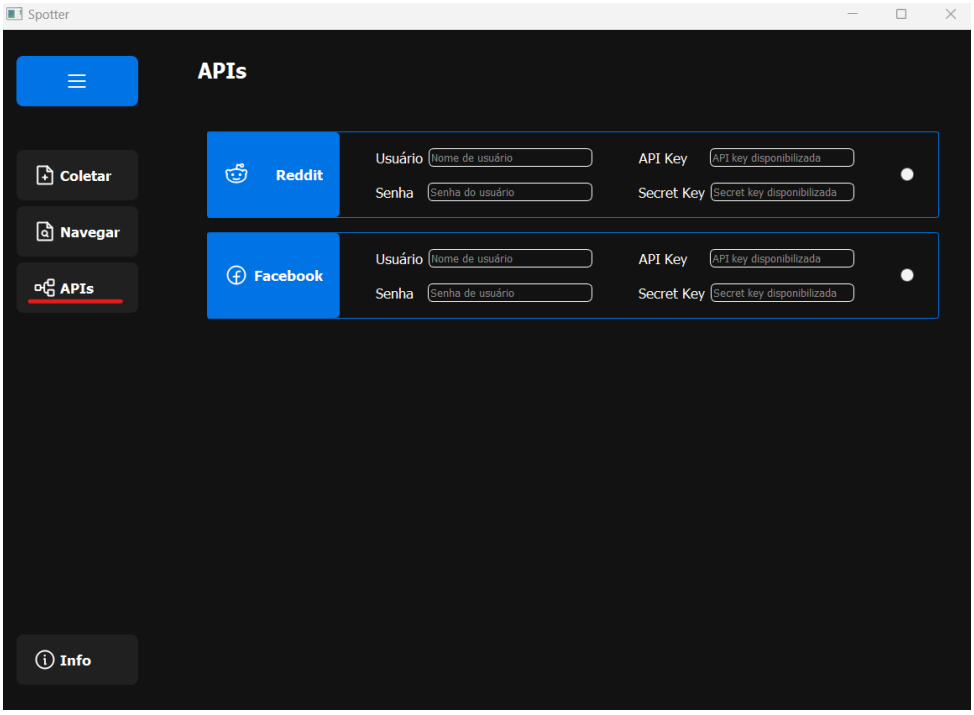

Figura 9 – Tela APIs

4.4 Coletando novos dados

Para coletar novos dados navegue até a página de coleta utilizando o botão *Coletar* no menu lateral, Figura 11.

Na página de coleta existem dois campos de texto, um para inserção dos tópicos e outro para as palavras-chaves. Preencha-os separando cada item com ponto e vírgula, exemplo, *python;javascript*. Não é necessário utilizar o ponto e vírgula quando inserir somente um tópico ou palavra-chave, nem ao final do texto, exemplo *python;* ou *python;javascript;*.

Após preencher os campos poderá selecionar qual filtro usuará, pondendo escolher um ou ambos, veja a Figura 12. Por fim, clique no botão *Extrair Novos Dados* no canto inferior direito da página de coleta para iniciar o processo de coleta.

É importante ressaltar que para realizar uma coleta de dados uma API deve ser selecionada na página de APIs, verifique a seção 4.3.

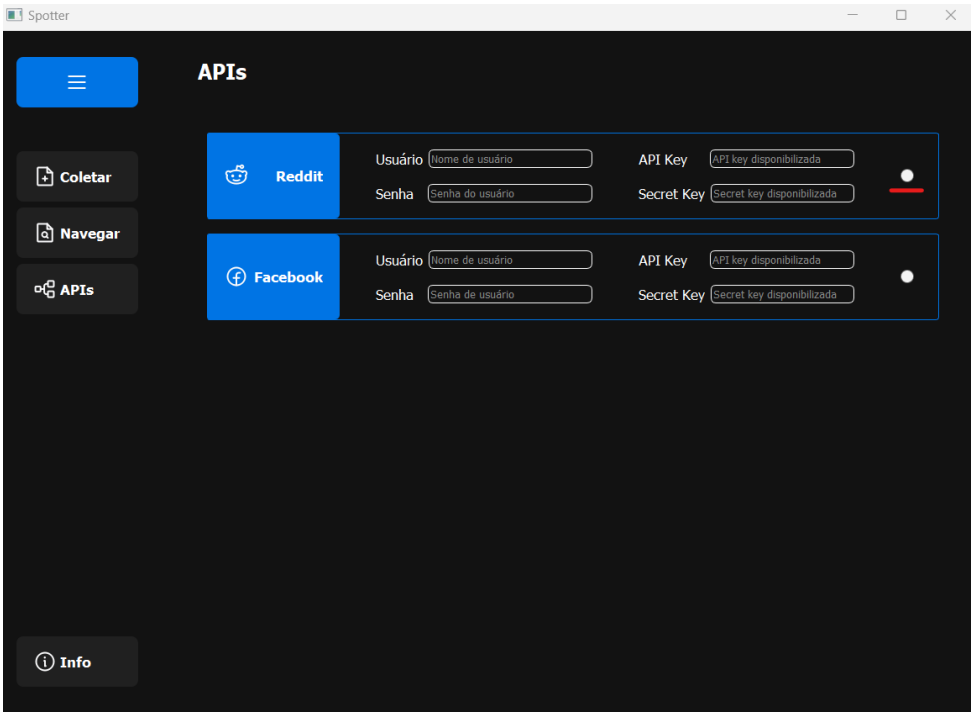

Figura 10 – Botão de adicionar API à coleta

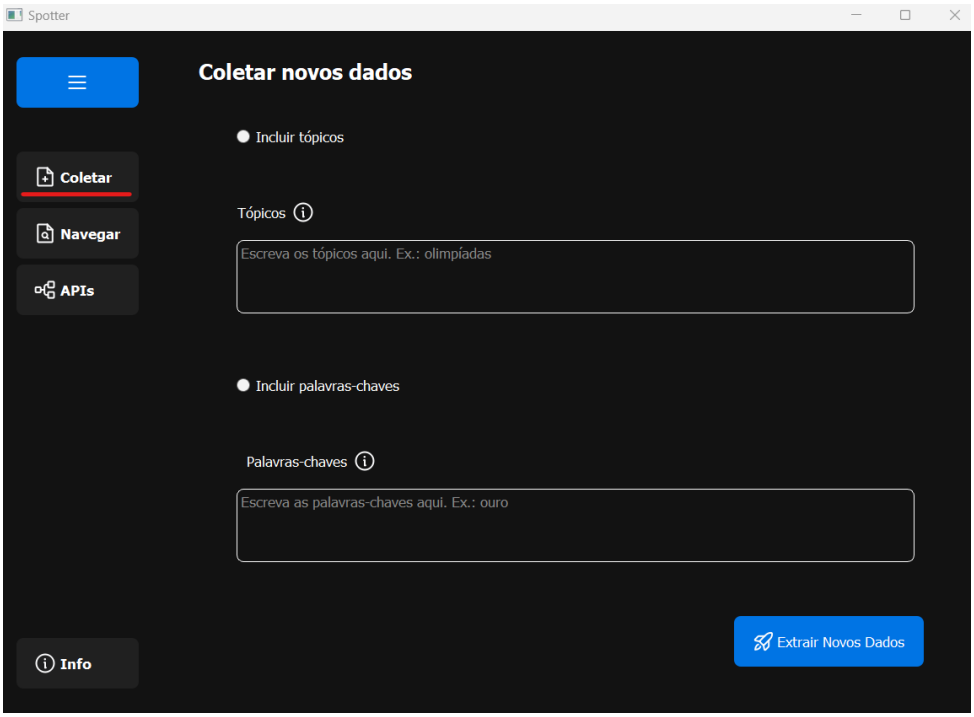

Figura 11 – Tela de Coleta

4.5 Visualizando os dados de coleta

A ferramenta possui uma tela para apresentar os metadados das coletas como data, APIs utilizadas, id da coleta e filtros que estão registrados no banco de dados.

Para acessar a página de visualização dos dados de coleta clique no botão *Navegar*

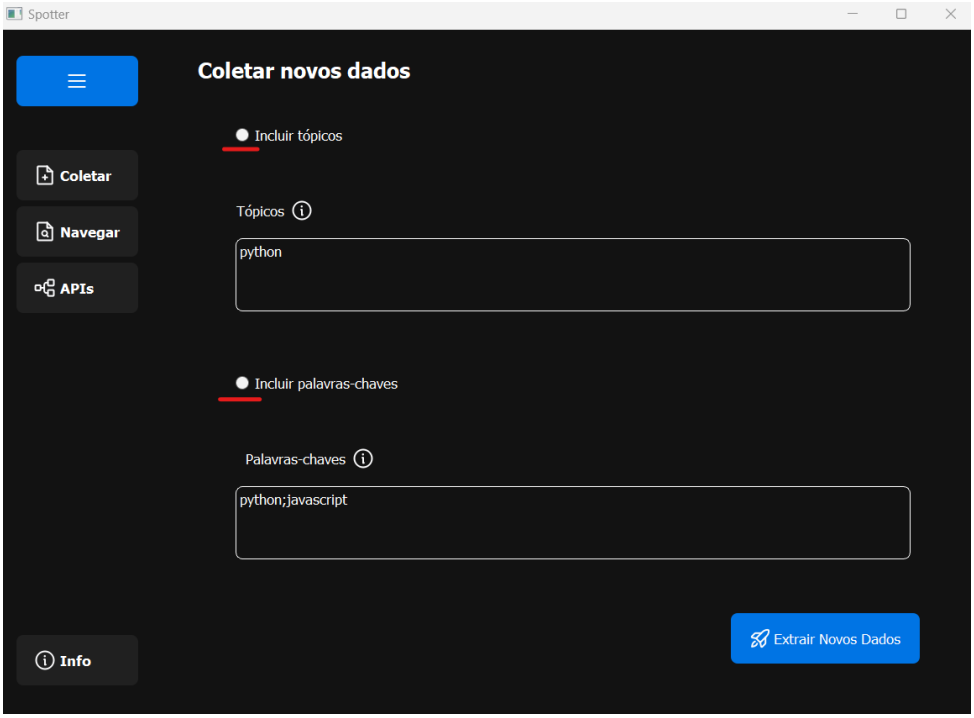

Figura 12 – Botão para adicionar filtro

no menu lateral, Figura 13. Quando não existirem dados registrados no banco de dados a mensagem "Não possui dados coletados"será apresentada no lugar.

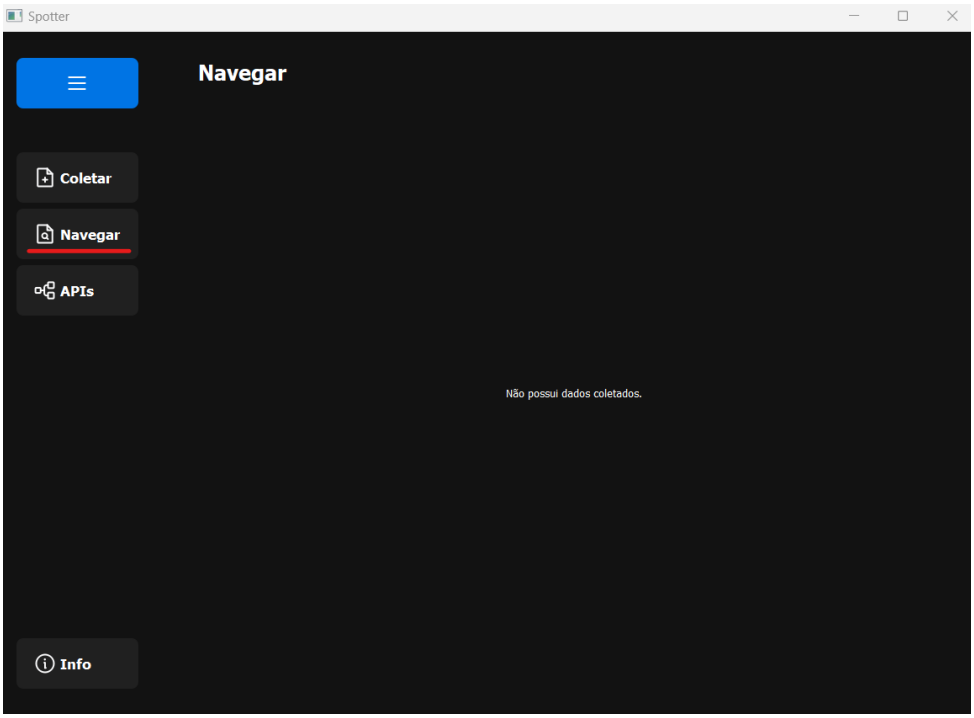

Figura 13 – Tela de Navegação

As coletas em processamento irão aparecer na página de navegação com uma *label* indicando que os dados ainda estão sendo coletados, como na Figura 14. Assim que os

dados estiverem registrados no banco a página atualizará automaticamente e irá apresentar as informações completas da coleta, como na Figura 15.

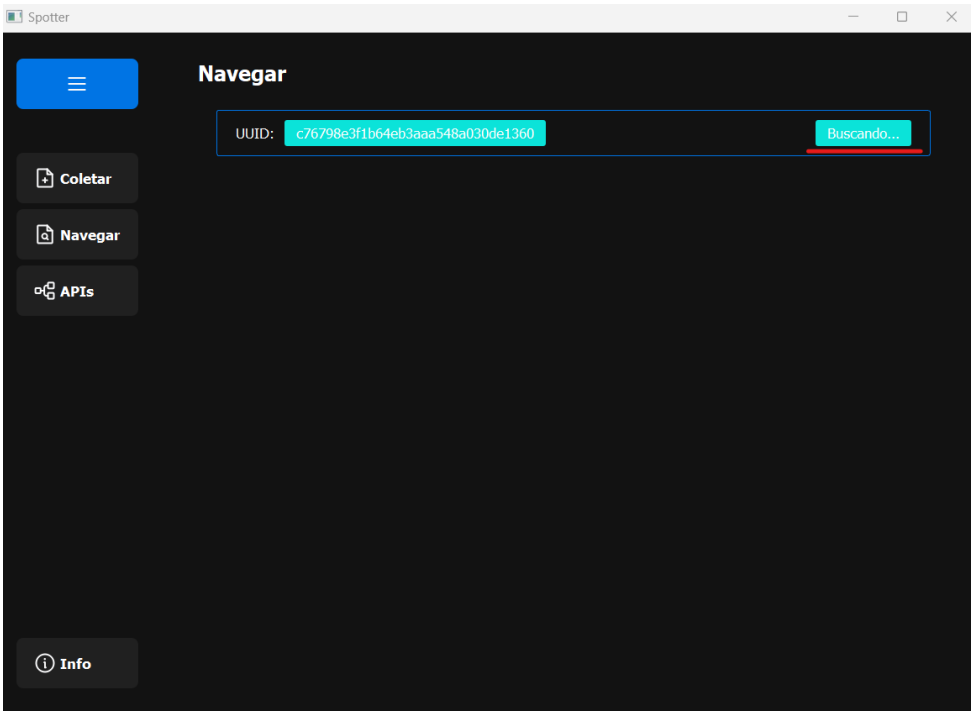

Figura 14 – Coleta em processamento

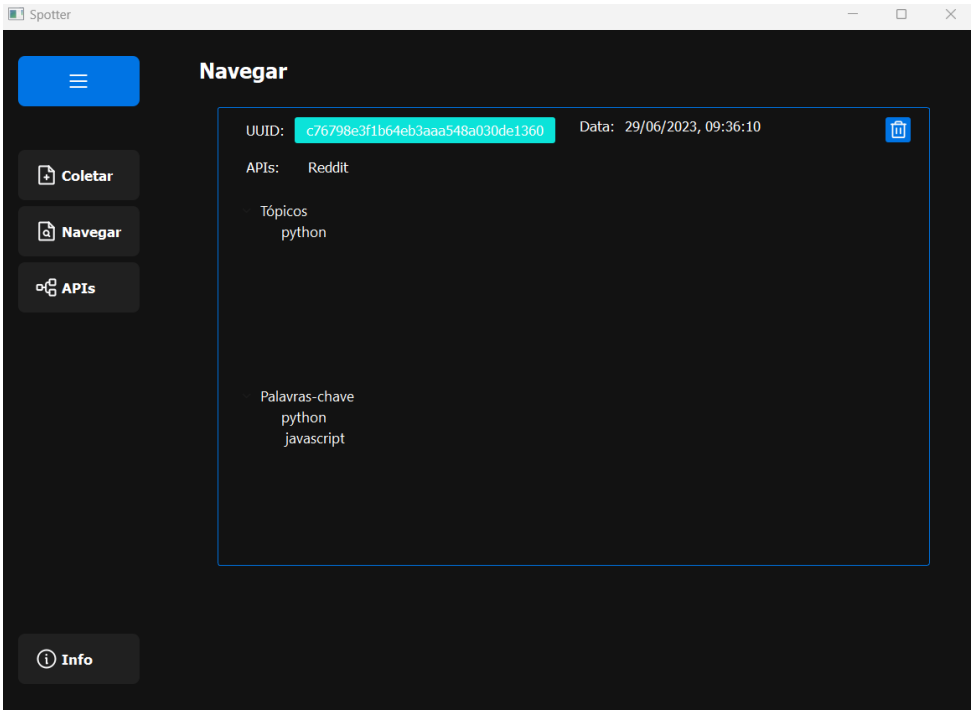

Figura 15 – Coleta concluída

Um ícone de lixeira aparecerá quando a coleta estiver concluída, ao clicar sobre o ícone os dados coletados serão removidos do banco de dados, os metadados da coleta e os comentários selecionados.

4.6 Mensagens de Erro

A aplicação emitirá uma mensagem de erro quando ocorrer algum problema de falha de comunicação com as APIs, falha no login ou bugs na inserção dos filtros. Quando a mensagem da Figura 16 aparecer ocorreu um erro durante o login na API selecionada, ou houve falha na comunicação, neste caso verifique se as chaves foram inseridas corretamente e tente novamente.

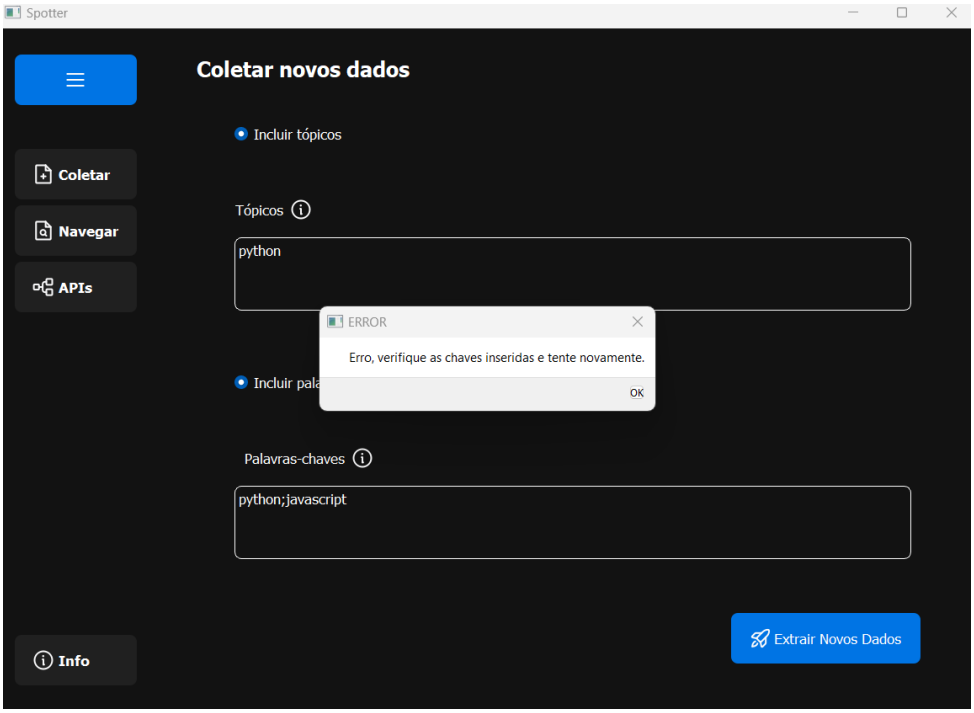

Figura 16 – Erro no Login ou comunicação com as APIs

Se a mensagem de erro da Figura 17 aparecer o processo de filtragem falhou, os campos de filtro podem estar vazios assim como no exemplo, ou mal formatados, verifique as informações inseridas e tente novamente. Para mais informações veja a seção 4.4 de como inserir os filtros de busca.

4.7 Rodando os Testes de unidade

Após o processo de instalação, veja a seção 4.1, abra um terminal de comando na raíz da pasta *spotter_pfp* e execute o comandao abaixo:

```
$ python -W ignore -m unittest tests.<NomeDoModuloDeTeste>.
<NomeDoTeste> -v
```

Substitua os valores entre "<>"pelo nome do módulo de teste e pelo nome do teste. Essas informações podem ser encontradas na seção 3.

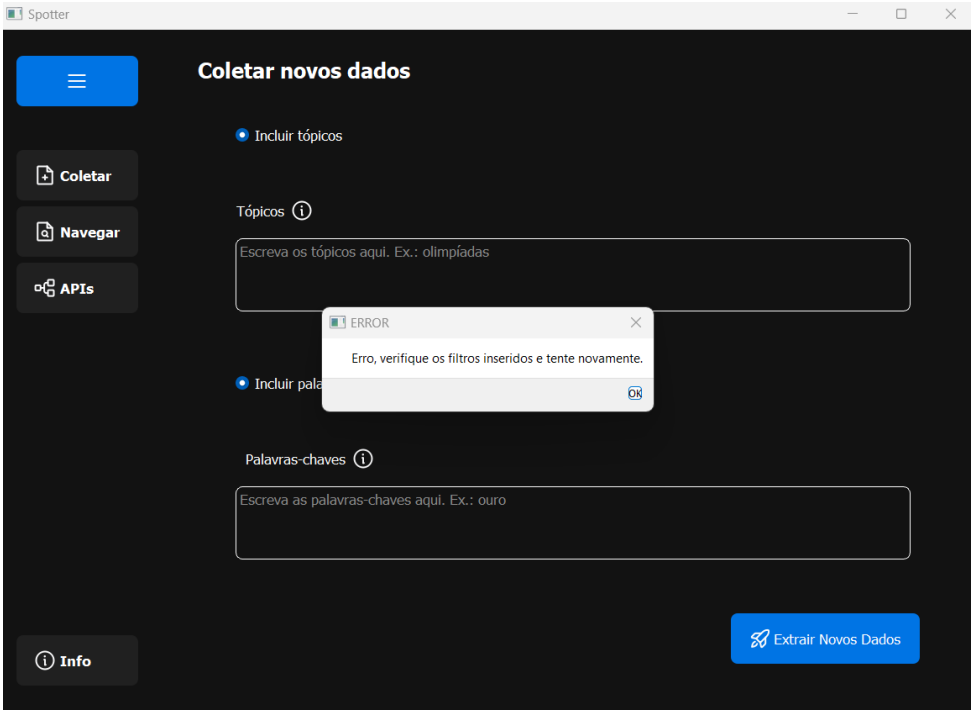

Figura 17 – Erro nos filtros inseridos

Exemplo de um teste válido:

```
$ python -W ignore -m unittest tests.TestMainPage.
test_labels_coletar_page -v
```

Para os testes do módulo de teste *TestRedditAPI* será necessário informar as chaves de acesso a API através de variáveis de ambiente, são elas *API_USER*, nome de usuário; *API_PASSWORD*, senha de usuário; *API_KEY*, key fornecida pela plataforma e *API_SECRET_KEY*, secret key fornecida pela plataforma.

Para definir uma variável de ambiente no Windows use o comando:

```
$ set <NomeDaVariável>=<Valor>
```

Exemplo,

```
$ set API_USER=UsuarioTeste
```