Projeto Final RPA - Raça de Cães

Nome

Guilherme Scarpelli Bellemo - RA: 2400117

API

A API escolhida foi a The Dog API, uma API pública e gratuita que fornece informações detalhadas sobre raças de cães. Ela disponibiliza dados como nome da raça, país de origem, expectativa de vida, temperamento, tamanho, peso, entre outros. A escolha se deu pelo interesse pessoal em adotar um cachorro, o que tornou o tema especialmente relevante e atrativo para o desenvolvimento deste projeto.

Etapas Executadas

prova_app.py > ...

```python
import requests
import sqlite3
import re
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

```

Import das bibliotecas.

```python

def main():
    dados = coletar_dados()
    if dados:
        armazenar_dados(dados)
        processar_dados()
        enviar_email()

if __name__ == "__main__":
    main()

```

Função principal que executa o fluxo completo do projeto:

- Coleta os dados via API.
- Armazena os dados no banco.
- Processa os dados da expectativa de vida.
- Envia o e-mail com o relatório.

O bloco final:

```
if __name__ == "__main__":
    main()
```

Garante que o script só será executado se for chamado diretamente, não ao ser importado como módulo.

```
8
9   def coletar_dados():
10      url = "https://api.thedogapi.com/v1/breeds"
11      response = requests.get(url)
12      if response.status_code == 200:
13          print("Dados coletados com sucesso.")
14          return response.json()
15      else:
16          print("Erro na requisição:", response.status_code)
17          return []
```

Função acessa a API pública TheDogAPI e coleta informações sobre raças de cães.

- Define a URL da API.
- Realiza uma requisição GET.
- Se a resposta for 200 (OK), retorna os dados no formato JSON.
- Caso contrário, retorna uma lista vazia.

```python
def armazenar_dados(dados):
    conn = sqlite3.connect('projeto_rpa.db')
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS racas (
            id INTEGER PRIMARY KEY,
            name TEXT,
            origin TEXT,
            life_span TEXT
        )
    ''')

    for dog in dados:
        cursor.execute('''
            INSERT OR IGNORE INTO racas (id, name, origin, life_span)
            VALUES (?, ?, ?, ?)
        ''', (
            dog['id'],
            dog['name'],
            dog.get('origin', ''),
            dog['life_span']
        ))

    conn.commit()
    conn.close()
    print("Dados armazenados no banco de dados.")
```

Função armazena os dados coletados em um banco de dados SQLite chamado 'projeto_rpa.db'.

- Conecta ao banco e cria a tabela 'raças' caso não exista.
- Insere os dados coletados utilizando INSERT OR IGNORE para evitar duplicatas.
- Salva as alterações e fecha a conexão.

```python
def processar_dados():
    conn = sqlite3.connect('projeto_rpa.db')
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS dados_processados (
            id INTEGER PRIMARY KEY,
            nome TEXT,
            expectativa_min INTEGER,
            expectativa_max INTEGER
        )
    ''')

    cursor.execute("SELECT id, name, life_span FROM racas")
    dados = cursor.fetchall()

    for id, nome, life_span in dados:
        numeros = re.findall(r'\d+', life_span)
        if len(numeros) >= 2:
            expectativa_min, expectativa_max = int(numeros[0]), int(numeros[1])
        elif len(numeros) == 1:
            expectativa_min = expectativa_max = int(numeros[0])
        else:
            expectativa_min = expectativa_max = None

        cursor.execute('''
            INSERT OR REPLACE INTO dados_processados (id, nome, expectativa_min, expectativa_max)
            VALUES (?, ?, ?, ?)
        ''', (id, nome, expectativa_min, expectativa_max))

    conn.commit()
    conn.close()
    print("Dados processados e armazenados.")
```

Função processa a expectativa de vida presente nos dados, separando os valores mínimo e máximo.

- Conecta ao banco de dados e cria a tabela 'dados_processados'.
- Extrai os números da expectativa de vida com regex.
- Insere os dados tratados nessa nova tabela.

```python
def enviar_email():
    conn = sqlite3.connect('projeto_rpa.db')
    cursor = conn.cursor()

    cursor.execute("SELECT nome, expectativa_min, expectativa_max FROM dados_processados LIMIT 20")
    linhas = cursor.fetchall()

    relatorio = "Relatório de Raças de Cães\n\n"
    for nome, min_, max_ in linhas:
        if min_ is not None and max_ is not None:
            relatorio += f"Raça: {nome} - Expectativa: {min_} a {max_} anos\n"
        else:
            relatorio += f"Raça: {nome} - Expectativa: desconhecida\n"

    conn.close()

    remetente = "gbellemo@gmail.com"
    senha = "kqoc qmxy uvts qylp"
    destinatario = "guilherme.bellemo@aluno.faculdadeimpacta.com.br"

    msg = MIMEMultipart()
    msg['From'] = remetente
    msg['To'] = destinatario
    msg['Subject'] = "Relatório RPA - Raças de Cães"
    msg.attach(MIMEText(relatorio, 'plain'))

    try:
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as servidor:
            servidor.login(remetente, senha)
            servidor.send_message(msg)
            print("E-mail enviado com sucesso.")
    except Exception as e:
        print(" Erro ao enviar o e-mail:", e)
```

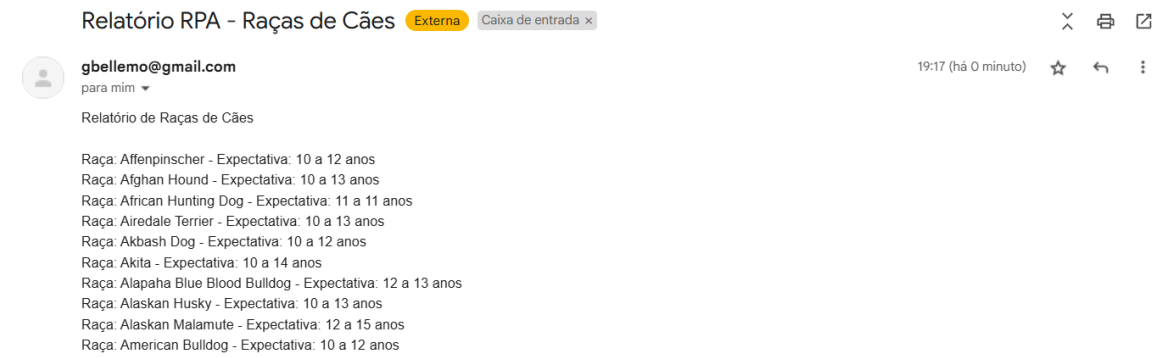
Função que gera um relatório com os dados de até 20 raças e envia por e-mail.

- Conecta ao banco e extrai os dados processados.
- Gera o corpo do e-mail com o relatório das raças.
- Usa SMTP com SSL para login e envio do e-mail.
- Captura exceções caso ocorra erro no envio.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Guilherme Notebook\Documents\RPA\Projeto-RPA> py .\prova_app.py
Dados coletados com sucesso.
Dados armazenados no banco de dados.
Dados processados e armazenados.
E-mail enviado com sucesso.
PS C:\Users\Guilherme Notebook\Documents\RPA\Projeto-RPA>
```

No terminal, é informado a coleta de dados, o armazenamento, processamento e envio destes dados.



Relatório RPA - Raças de Cães Externa Caixa de entrada ×

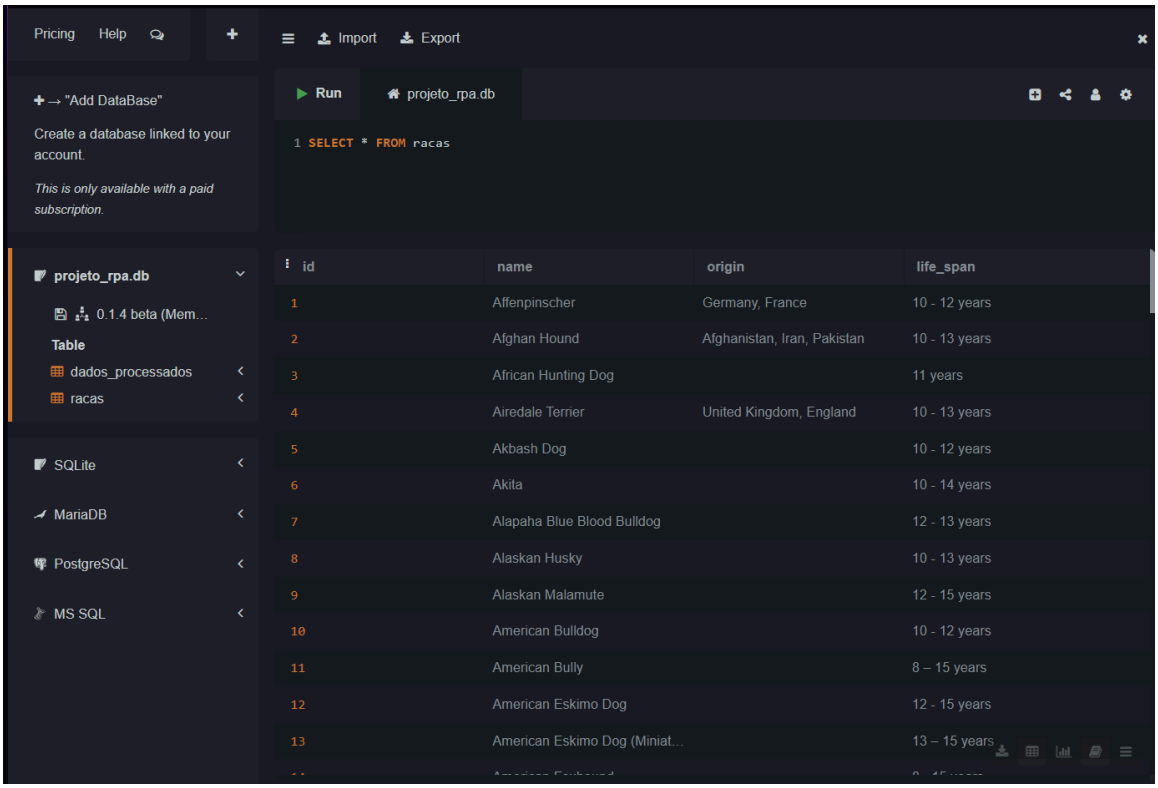
gbellemo@gmail.com
para mim

19:17 (há 0 minuto)

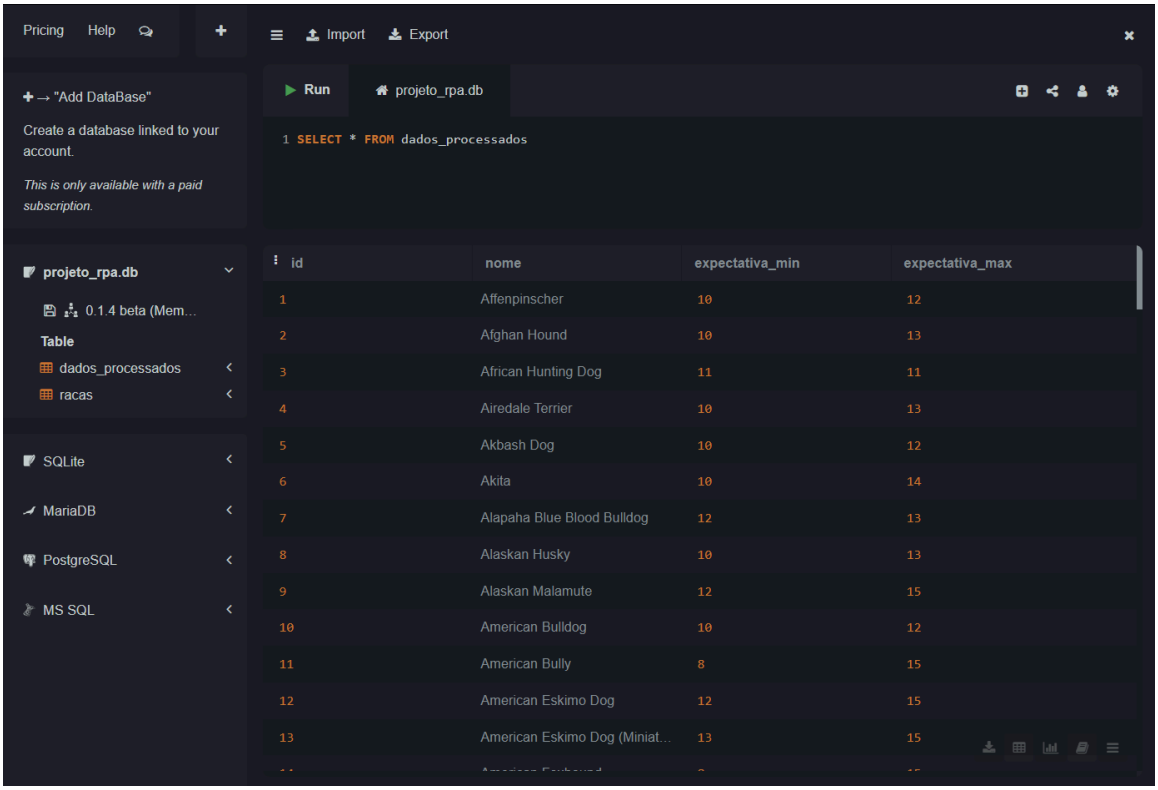
Relatório de Raças de Cães

Raça: Affenpinscher - Expectativa: 10 a 12 anos
Raça: Afghan Hound - Expectativa: 10 a 13 anos
Raça: African Hunting Dog - Expectativa: 11 a 11 anos
Raça: Airedale Terrier - Expectativa: 10 a 13 anos
Raça: Akbash Dog - Expectativa: 10 a 12 anos
Raça: Akita - Expectativa: 10 a 14 anos
Raça: Alapaha Blue Blood Bulldog - Expectativa: 12 a 13 anos
Raça: Alaskan Husky - Expectativa: 10 a 13 anos
Raça: Alaskan Malamute - Expectativa: 12 a 15 anos
Raça: American Bulldog - Expectativa: 10 a 12 anos

Resultado final do envio do relatório.



Consulta da tabela “racas”.



Consulta da tabela “dados_processados”.

Conclusão

O projeto foi concluído com êxito, atendendo aos objetivos de automatizar a coleta, armazenamento, processamento e envio de dados sobre raças de cães a partir de uma API pública. O sistema desenvolvido integra diferentes tecnologias e bibliotecas em Python para realizar todo o fluxo, desde a obtenção dos dados até a geração e envio de um relatório por e-mail.

Durante a implementação, a etapa de envio de e-mails apresentou maior complexidade. A configuração correta do servidor SMTP, uso de autenticação segura e montagem da mensagem exigiram maior atenção. Além disso, o processamento das expectativas de vida das raças, utilizando expressões regulares para extrair e organizar os dados de forma estruturada no banco, também demandou esforço de compreensão e teste.

Apesar dos desafios, a execução integrada das funções — coleta com requests, armazenamento e processamento com sqlite3, e envio com smtplib — demonstrou-se eficiente. O projeto serviu como uma aplicação prática de conceitos de automação com Python e manipulação de dados em múltiplas etapas.