IESB

Big Data e Inteligência Analítica

Projeto Integrador em Big Data e Inteligência Analítica

Lucas Siqueira Rodrigues

Brasília - DF
Junho de 2025

1 Obtenção da base de dados

1.1 Introdução

O trabalho tem como objetivo explorar os dados públicos disponibilizados pela Câmara dos Deputados para analisar e compreender os gastos realizados pelos parlamentares. Para isso, os dados serão obtidos por meio do portal dos dados abertos da câmara[1], armazenados em um banco de dados relacional na nuvem e, posteriormente, analisados.

1.2 Motivação

A transparência pública é essencial para garantir a confiança da população nas instituições governamentais. Esse projeto busca realizar um ciclo completo de extração, transformação, armazenamento e análise dos dados de gastos públicos. Além disso, ao construir dashboards dinâmicos espera-se fornecer ferramentas que possam auxiliar na identificação de possíveis irregularidades e na fiscalização das despesas parlamentares.

1.3 Script e Banco de Dados

Para o ano de 2022, a obtenção dos dados por meio da API[1] retornou apenas 32 registros.

```python
import io
import json
import zipfile

import httpx
from tqdm import tqdm

class CamaraAPI:
  def __init__(self) -> None:
    self.base_url = "https://dadosabertos.camara.leg.br/api/v2"

  def request(self, endpoint: str) -> dict:
    response = httpx.get(f"{self.base_url}/{endpoint}")
    return response.json()

  def get_deputados(self) -> dict:
    return self.request("deputados").get("dados", {})

  def get_despesas(self, id_: int, year: int = 2022) -> dict:
    return self.request(f"deputados/{id_}/despesas?ano={year}")

despesas = []

api = CamaraAPI()

deputados = api.get_deputados()
for deputado in tqdm(deputados):
  id_ = deputado["id"]
  despesas_deputado = api.get_despesas(id_=id_, year=2022)
  despesas.extend(despesas_deputado["dados"])

print(len(despesas))  # output: 32
```

Por isso, apenas para esse ano a coleta de dados foi por meio de um arquivo no formato JSON, que também é fornecido no portal de dados abertos da câmara por meio da aba "Arquivos".

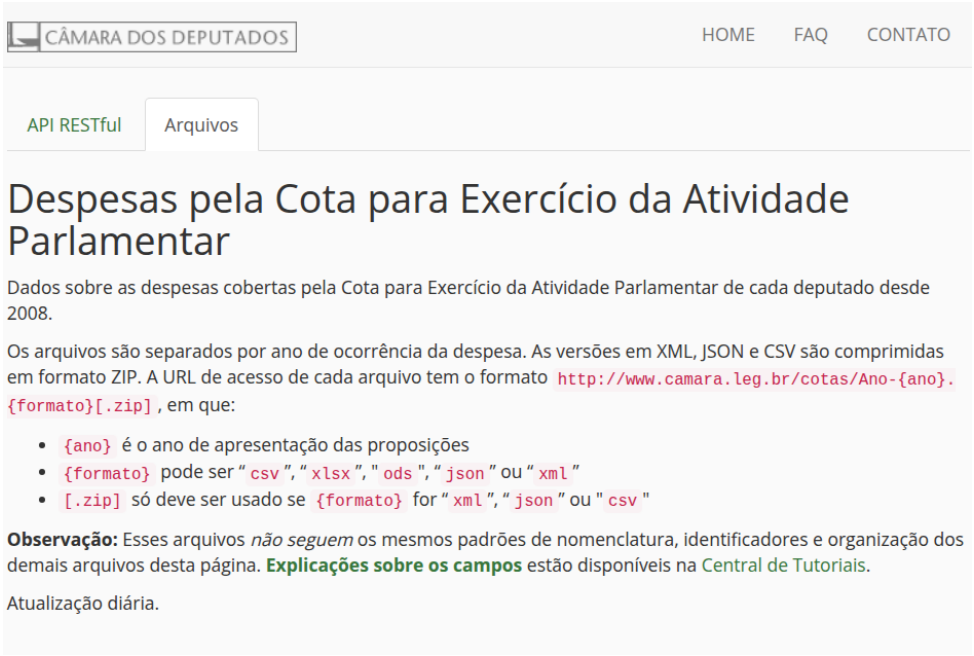

Figura 1: Coleta de dados por meio da aba de arquivos.

Para os anos de 2023 e 2024 a obtenção dos dados foi realizada por meio da API[1] da Câmara dos Deputados, explorando dois principais endpoints:

- `/deputados`: Retorna informações gerais sobre os parlamentares, como seus nomes, partidos, estados e e-mails.
- `/deputados/{id}/despesas`: Fornece detalhes sobre as despesas realizadas pelos parlamentares, incluindo valores, fornecedores, tipos de despesa e datas.

Para organizar os dados de forma eficiente e integrar os dados obtidos por meio da API e por meio do JSON, foi criado um modelo de banco de dados relacional com tabelas normalizadas para representar as informações de deputados, despesas e fornecedores[2].

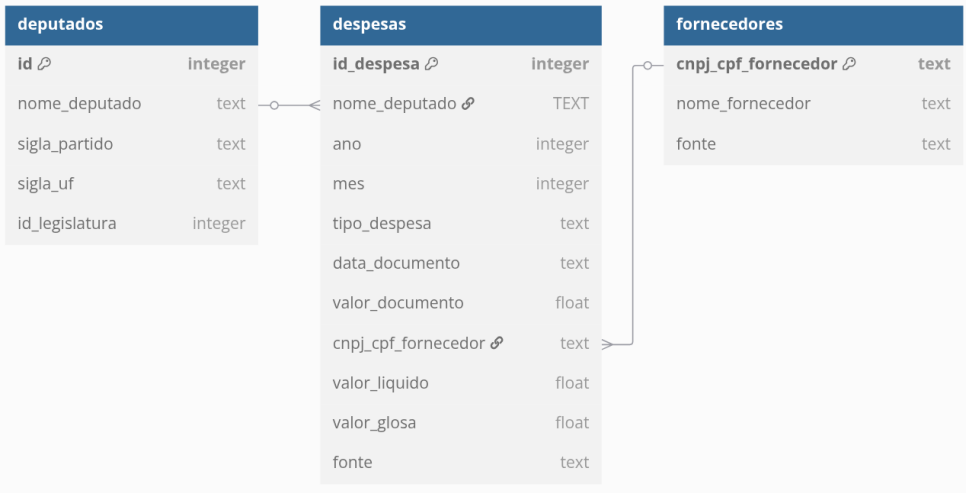

Figura 2: Diagrama relacional.

Os dados são obtidos e unificados por meio da classe DataService.

```python
import io
import json
import re
import zipfile

import httpx
from tqdm import tqdm

from data_ingestion.services.log_service import logger


class DataService:
  def __init__(self) -> None:
    self.api_base_url = "https://dadosabertos.camara.leg.br/api/v2"

  def get_deputados(self) -> list:
    response = httpx.get(f"{self.api_base_url}/deputados")
    data = response.json().get("dados")
    selected_data = [
      {
        "id": i["id"],
        "nome": i["nome"],
        "sigla_partido": i["siglaPartido"],
        "id_legislatura": i["idLegislatura"],
        "sigla_uf": i["siglaUf"],
      }
      for i in data
    ]

    return selected_data

  def get_data_from_api(self, deputados: list[dict], anos: list[int]) ->
                                        tuple[list[dict], list[dict]]:
    despesas = []
    fornecedores = []
    for ano in anos:
    logger.info(f"Getting data from API for year {ano}")
```

```
37      for deputado in tqdm(deputados):
38      response = httpx.get(f"{self.api_base_url}/deputados/{deputado['id']
                                        }/despesas?ano={ano}")
39      despesas_deputado = response.json().get("dados")
40
41      for item in despesas_deputado:
42      despesas.append(
43      {
44        "nome_deputado": item.get("nome"),
45        "ano": item.get("ano"),
46        "mes": item.get("mes"),
47        "tipo_despesa": item.get("tipoDespesa"),
48        "data_documento": item.get("dataDocumento"),
49        "valor_documento": item.get("valorDocumento"),
50        "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("
                                        cnpjCpfFornecedor")),
51        "valor_liquido": item.get("valorLiquido"),
52        "valor_glosa": item.get("valorGlosa"),
53        "fonte": "api",
54      },
55      )
56
57      fornecedores.append({
58        "nome_fornecedor": item.get("nomeFornecedor"),
59        "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("
                                        cnpjCpfFornecedor")),
60        "fonte": "api",
61      })
62
63      logger.info(f"Total number of despesas: {len(despesas)}")
64
65      return despesas, fornecedores
66
67    @staticmethod
68    def get_data_from_url(url: str) -> tuple[list[dict], list[dict]]:
69      logger.info(f"Getting data from url: {url}")
70      response = httpx.get(url=url)
71      zip_content = response.content
72
73      with zipfile.ZipFile(io.BytesIO(zip_content)) as zip_file:
74      with zip_file.open(zip_file.namelist()[0]) as json_file:
75      json_content = json_file.read().decode("utf-8")
76      json_data = json.loads(json_content).get("dados")
77
78      despesas = []
79      fornecedores = []
80
81      for item in json_data:
82      despesas.append(
83      {
84        "nome_deputado": item.get("nomeParlamentar"),
85        "ano": item.get("ano"),
86        "mes": item.get("mes"),
87        "tipo_despesa": item.get("descricao"),
88        "data_documento": item.get("dataEmissao"),
89        "valor_documento": item.get("valorDocumento"),
90        "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("cnpjCPF")),
91        "valor_liquido": item.get("valorLiquido"),
```

```
92          "valor_glosa": item.get("valorGlosa"),
93          "fonte": "url",
94      },
95      )
96
97      fornecedores.append({
98          "nome_fornecedor": item.get("fornecedor"),
99          "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("cnpjCPF")),
100         "fonte": "url",
101     })
102
103     logger.info(f"Total number of despesas: {len(despesas)}")
104
105     return despesas, fornecedores
```

Por fim, temos a função main(), que realiza todo o ciclo de extração, transformação e carregamento dos dados.

```
1  from data_ingestion.services.data_service import DataService
2  from data_ingestion.services.db_service import DBService
3
4
5  def main() -> None:
6      url = "https://www.camara.leg.br/cotas/Ano-2022.json.zip"
7      anos = [2023, 2024]
8
9      data_service = DataService()
10     deputados = data_service.get_deputados()
11
12     api_despesas, api_fornecedores = data_service.get_data_from_api(
                                            deputados=deputados, anos=anos)
13     url_despesas, url_fornecedores = data_service.get_data_from_url(url=
                                            url)
14
15     despesas = [*api_despesas, *url_despesas]
16     fornecedores = [*api_fornecedores, *url_fornecedores]
17
18     db_service = DBService()
19     db_service.connect()
20
21     db_service.create_tables()
22     db_service.insert_deputados(deputados)
23     db_service.insert_despesas(despesas)
24     db_service.insert_fornecedores(fornecedores)
25     db_service.close()
26
27
28 if __name__ == "__main__":
29     main()
30
```

O programa, ao ser executado, faz a extração, transformação e carga de 224.042 registros de despesas dos anos de 2022 a 2024.

Logo após serem obtidos, os dados foram inseridos em um banco de dados PostgreSQL, que foi criado usando o serviço Amazon RDS (Relational Database Service).

Todo o código relacionado ao projeto está no Github[3].

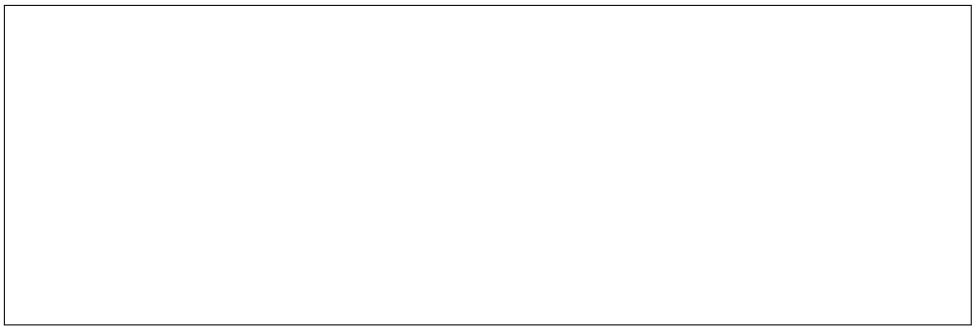

Figura 3: Criação do PostgreSQL na AWS.

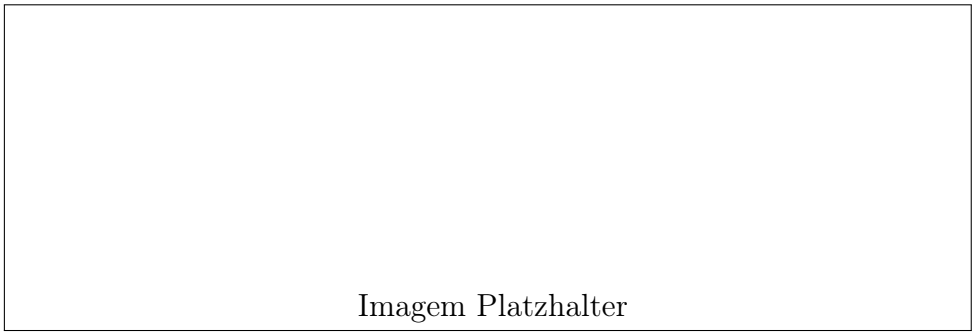

Figura 4: Base de dados em processo de criação.

1.4 Considerações finais

A combinação de tecnologias como Python, AWS e PostgreSQL foi fundamental para realizar as etapas de extração, transformação e carregamento dos dados. A API da Câmara revelou-se limitada em relação à quantidade de dados retornados para o ano de 2022, mas a extração dos dados do JSON permitiu superar essa restrição e criar uma base robusta com uma quantidade considerável de dados.

2 Relatório Analítico

2.1 Introdução

A análise de dados públicos desempenha um papel crucial na promoção da transparência governamental e no combate a irregularidades. Utilizando o Streamlit para a criação de dashboards, é possível transformar grandes volumes de dados em informações compreensíveis e acessíveis para a população e órgãos de fiscalização.

2.2 Demonstração

O Streamlit foi integrado ao banco de dados PostgreSQL, permitindo consultas em tempo real e construção de dashboards dinâmicos.

Na nossa base de dados foram construídas 3 tabelas, e para facilitar o processo de construção de gráficos uma query foi feita realizando o join das tabelas.

```
SELECT * FROM database;
```

E finalmente temos a nossa base de dados conectada com o PowerBI.

Temos diversas colunas interessantes que podemos usar na construção dos nossos gráficos:

- cnpj_cpf_despesa
- mes
- descricao
- descricao_especificacao
- valor_documento
- nome_deputado
- uf
- sigla_partido
- nome_fornecedor

2.3 Relatórios e tabelas

Foram construídas algumas páginas contendo alguns gráficos.

Primeiramente, podemos ver que o valor total de despesas no ano de 2022 foi de 221,4 milhões de reais. Há vários registros na coluna nome_deputado com o nome LIDERANÇA DO CIDADANIA, que possui o maior valor de despesas, totalizando R$ 912.660,00, seguido da Joenia Wapichana com o valor R$ 565.630,00 e do Jesus Sérgio com o valor R$ 549.970,00. A atividade com o maior gasto foi de Divulgação da Atividade Parlamentar, com R$ 52 milhões gastos, seguida por Passagem Aérea com R$ 48 milhões e Locação ou Fretamento do veículos automotores com R$ 29 milhões. Ao separar os gastos pelo fornecedor, temos o seguinte dado: Podemos notar que há uma grande discrepância de gastos com o fornecedor GOL quando comparado com outros fornecedores. Em outra página do dashboard, temos gráficos de gastos separados por Partido, UF e Mês do Ano. O partido que mais gastou foi o PL, totalizando R$ 32 milhões. O estado que mais gastou foi São Paulo, com R$ 26 milhões. E o mês com maior gasto foi o mês de Dezembro, com um gasto total de R$ 24 milhões.

2.4 Considerações finais

O PowerBI é uma ferramenta poderosa para análise de dados, por meio dela podemos construir gráficos dinâmicos que auxiliam em muito a análise e conseguimos tirar diversos insights. Outra grande vantagem é poder criar gráficos que retornam os dados diretamente do banco de dados, assim podemos ter gráficos atualizados e em tempo real.

3 Machine Learning

3.1 Introdução

O machine learning pode ser usado para encontrar padrões nos dados. Nessa análise, não vi nenhum tipo de machine learning que poderia nos trazer algum tipo de informação sobre os dados.

3.2 Dicionário de dados

3.3 Considerações finais

4 Vídeo

O vídeo de 10 minutos mostrando todo o projeto foi gravado e disponibilizado por meio do google drive através do link: link.

Referências

[1] Portal de Dados Abertos da Câmara dos Deputados: https://dadosabertos.camara.leg.br/swagger/api.html

[2] dbdiagram: https://dbdiagram.io/

[3] Repositório do Projeto no Github: https://github.com/lucassiro/pi