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".

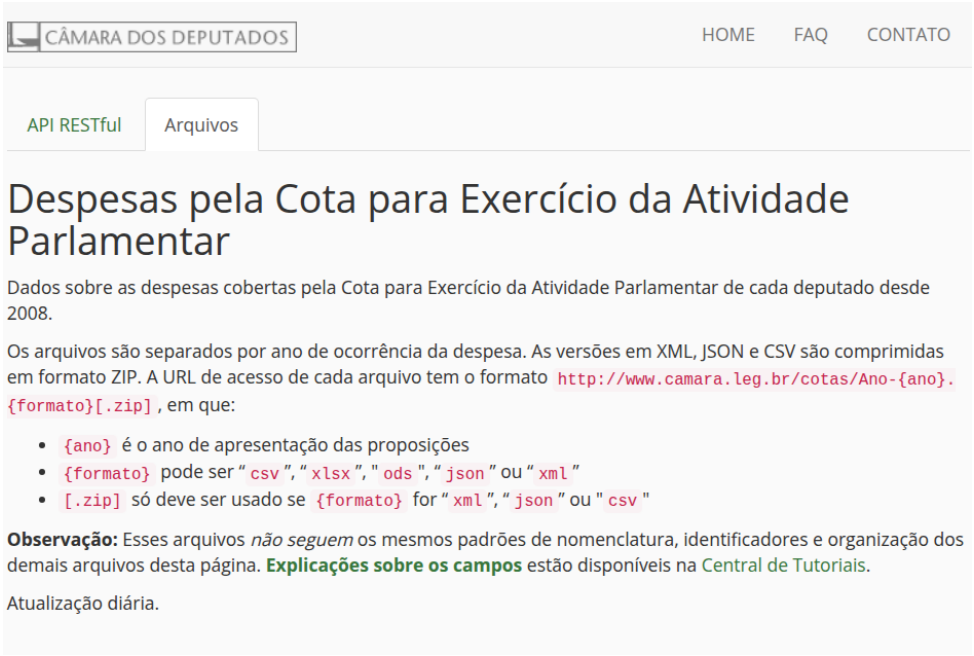

Figura 1: Coleta de dados por meio da aba de arquivos.

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].

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"

```

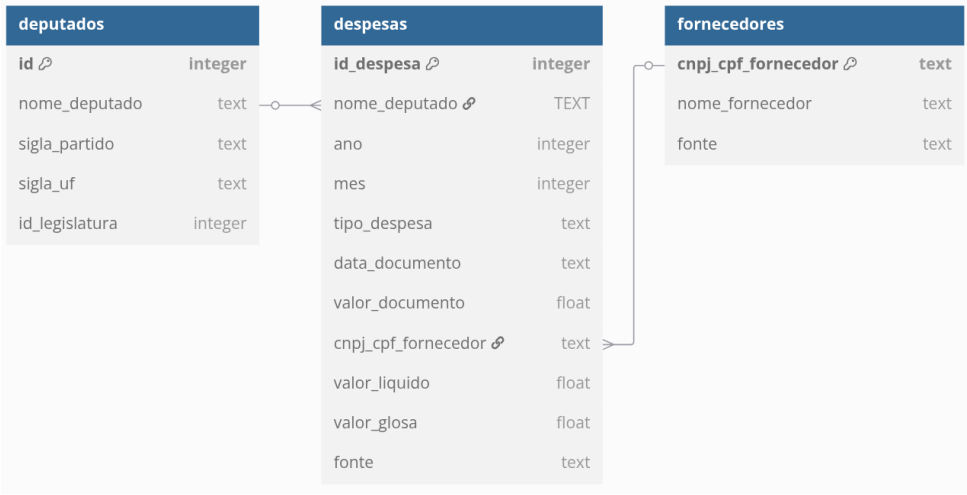

Figura 2: Diagrama relacional.

```python
  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}")
      for deputado in tqdm(deputados):
        response = httpx.get(f"{self.api_base_url}/deputados/{deputado['
                                      id']}/despesas?ano={ano}")
        despesas_deputado = response.json().get("dados")

        for item in despesas_deputado:
          despesas.append({
            "nome_deputado": deputado.get("nome"),
            "ano": item.get("ano"),
            "mes": item.get("mes"),
            "tipo_despesa": item.get("tipoDespesa"),
            "data_documento": item.get("dataDocumento"),
            "valor_documento": item.get("valorDocumento"),
            "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("
                                      cnpjCpfFornecedor")),
            "valor_liquido": item.get("valorLiquido"),
            "valor_glosa": item.get("valorGlosa"),
```

```
            "fonte": "api",
          })

          fornecedores.append({
            "nome_fornecedor": item.get("nomeFornecedor"),
            "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("
                                    cnpjCpfFornecedor")),
            "fonte": "api",
          })

    logger.info(f"Total number of despesas: {len(despesas)}")

    return despesas, fornecedores

  @staticmethod
  def get_data_from_url(url: str) -> tuple[list[dict], list[dict]]:
    logger.info(f"Getting data from url: {url}")
    response = httpx.get(url=url)
    zip_content = response.content

    with zipfile.ZipFile(io.BytesIO(zip_content)) as zip_file:
      with zip_file.open(zip_file.namelist()[0]) as json_file:
        json_content = json_file.read().decode("utf-8")
        json_data = json.loads(json_content).get("dados")

    despesas = []
    fornecedores = []

    for item in json_data:
      despesas.append({
        "nome_deputado": item.get("nomeParlamentar"),
        "ano": item.get("ano"),
        "mes": item.get("mes"),
        "tipo_despesa": item.get("descricao"),
        "data_documento": item.get("dataEmissao"),
        "valor_documento": item.get("valorDocumento"),
        "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("cnpjCPF")
                                ),
        "valor_liquido": item.get("valorLiquido"),
        "valor_glosa": item.get("valorGlosa"),
        "fonte": "url",
      })

      fornecedores.append({
        "nome_fornecedor": item.get("fornecedor"),
        "cnpj_cpf_fornecedor": re.sub(r"[^0-9]", "", item.get("cnpjCPF")
                                ),
        "fonte": "url",
      })

    logger.info(f"Total number of despesas: {len(despesas)}")

    return despesas, fornecedores
```

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

```
import os

```

```python
from dotenv import load_dotenv

from data_ingestion.services.data_service import DataService
from data_ingestion.services.db_service import DBService

_ = load_dotenv()


def main() -> None:
  url = "https://www.camara.leg.br/cotas/Ano-2022.json.zip"
  anos = [2023, 2024]

  data_service = DataService()
  deputados = data_service.get_deputados()

  api_despesas, api_fornecedores = data_service.get_data_from_api(
                                    deputados=deputados, anos=anos)
  url_despesas, url_fornecedores = data_service.get_data_from_url(url=
                                    url)

  despesas = [*api_despesas, *url_despesas]
  fornecedores = [*api_fornecedores, *url_fornecedores]

  db_service = DBService(
      user=os.environ["DB_USER"],
      password=os.environ["DB_PASSWORD"],
      host=os.environ["DB_HOST"],
      port=int(os.environ["DB_PORT"]),
      dbname=os.environ["DB_NAME"],
  )
  db_service.insert_data(deputados=deputados, despesas=despesas,
                                    fornecedores=fornecedores)


if __name__ == "__main__":
  main()

```

O programa, ao ser executado, faz a extração, transformação e carga de 223.982 registros de despesas dos anos de 2022 a 2024.

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

Após a base de dados ser criada, podemos nos conectar a ela usando o DBeaver[3].

Todo o código relacionado ao projeto está no Github[4].

1.4 Considerações finais

A combinação de tecnologias como Python, AWS e MySQL 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 da aba Arquivos permitiu superar essa restrição e criar uma base robusta com uma quantidade considerável de dados.

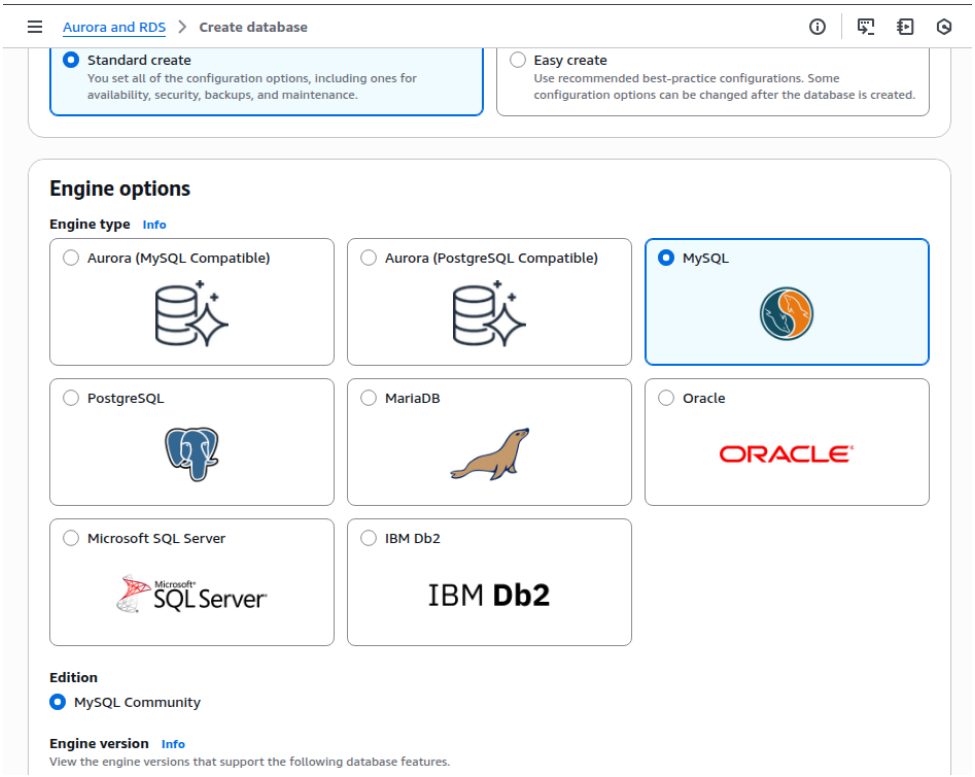

Figura 3: Criação do MySQL na AWS.

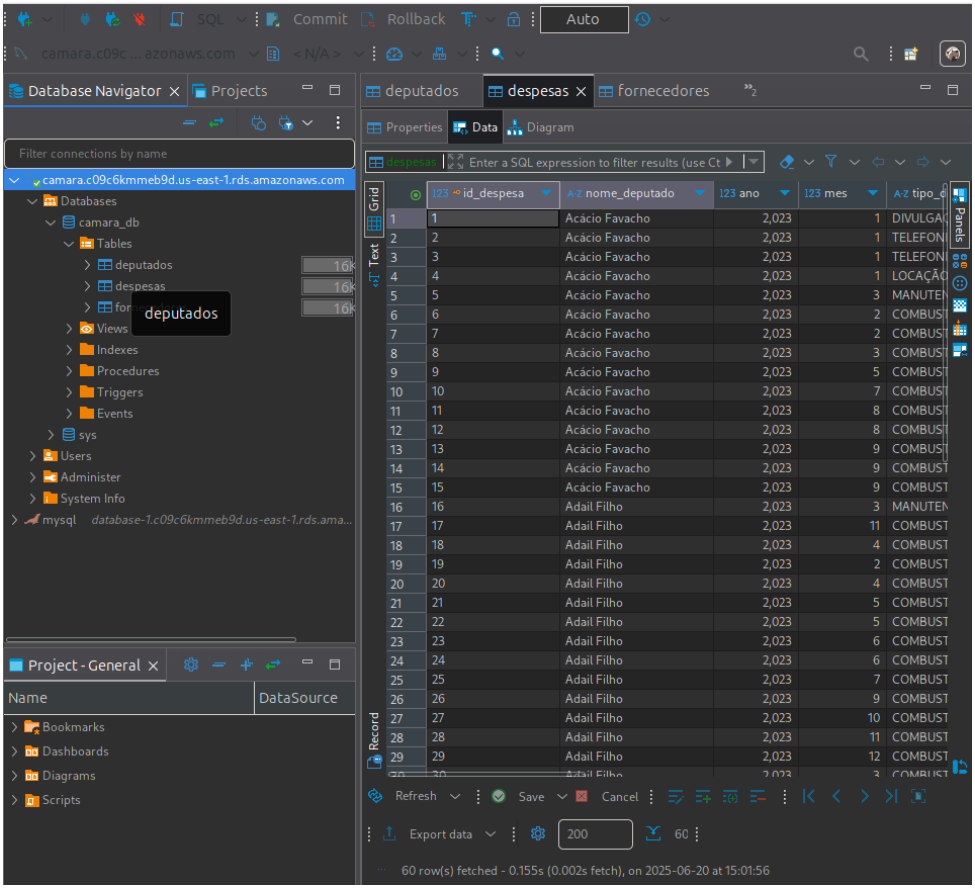

Figura 4: Base de dados conectada no DBeaver.

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 MySQL, 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
 despesas.nome_deputado,
 despesas.ano,
 despesas.mes,
 despesas.tipo_despesa,
 despesas.valor_documento,
 despesas.cnpj_cpf_fornecedor,
 fornecedores.nome_fornecedor,
 deputados.sigla_partido,
 deputados.id_legislatura,
 deputados.sigla_uf
FROM
 despesas
JOIN
 deputados ON despesas.nome_deputado = deputados.nome
JOIN
 fornecedores ON despesas.cnpj_cpf_fornecedor = fornecedores.cnpj_cpf_fornecedor;
```

É criado um dataframe com os dados, a partir desse dataframe podemos fazer agrupamentos de dados para gerar diversos tipos de gráficos.

```
1 grouped_df = df[["tipo_despesa", "valor_documento"]].groupby("
                                        tipo_despesa").sum()
2 st.bar_chart(grouped_df, horizontal=True, x_label="Valor total em R$")
```

2.3 Relatórios e tabelas

Aqui podemos analisar o gastos dos anos de 2022, 2023 e 2024 de forma combinada ou isolada. Temos que o total de gastos são:

Para os anos combinados, temos que MANUTENÇÃO DE ESCRITÓRIO DE APOIO À ATIVIDADE PARLAMENTAR foi a categoria de gastos com o maior valor, seguido de DIVULGAÇÃO DA ATIVIDADE PARLAMENTAR e PASSAGEM AÉREA - SIGEPA.

Figura 5: Gráfico mostrando a quantidade de gastos por tipo de despesa.

Ano	Total de Gastos	Quantidade de Deputados
Combinados	R$ 145 mi	503
2022	R$ 117 mi	280
2023	R$ 14 mi	494
2024	R$ 14 mi	496

Tabela 1: Resumo de gastos.

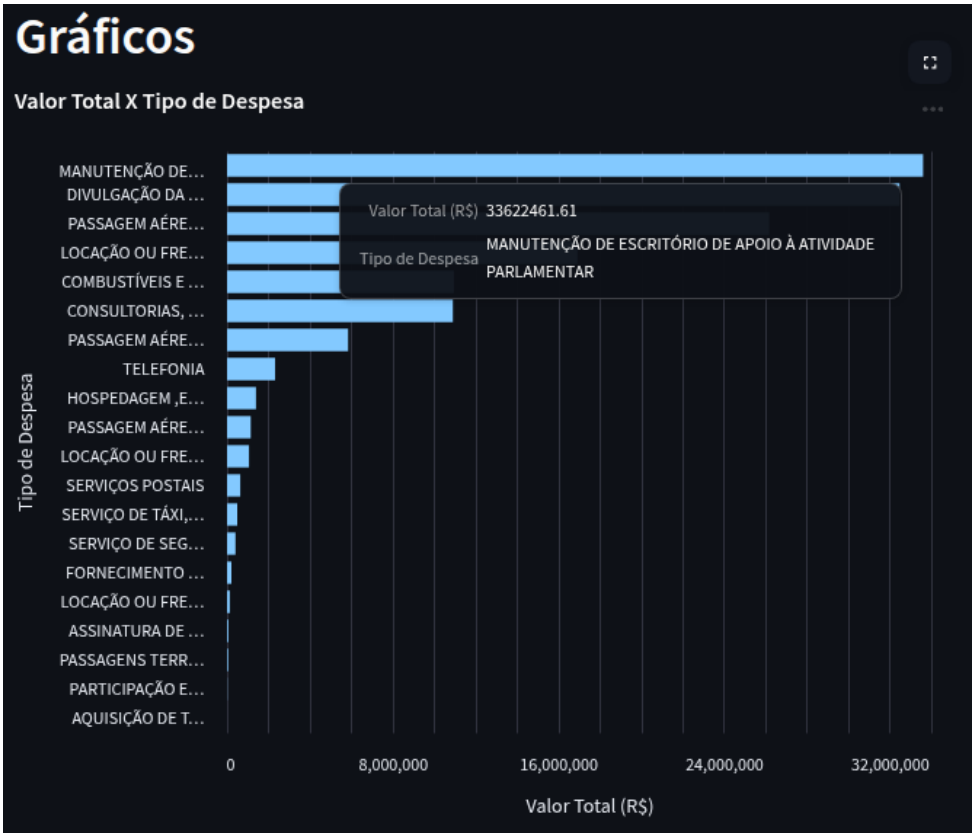

Figura 6: Valor dos gastos por tipo de despesa.

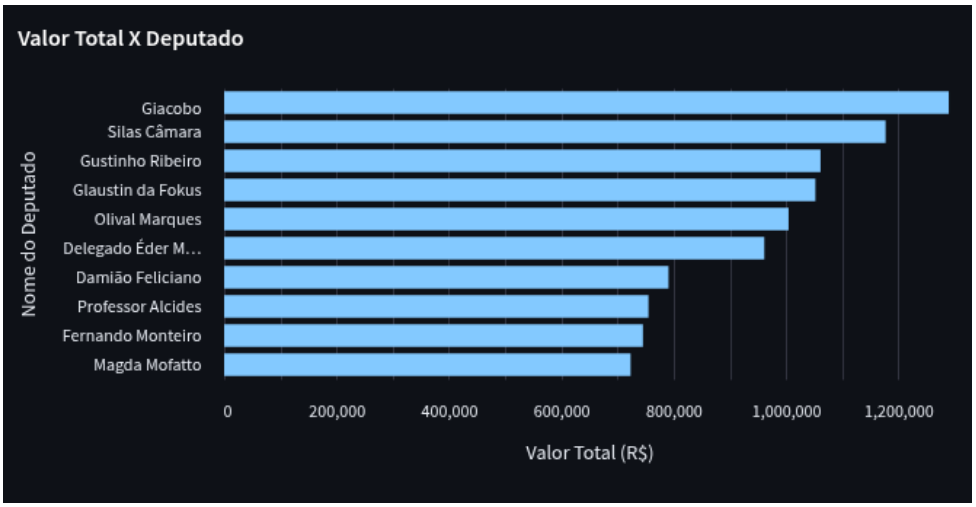

Figura 7: Valor dos gastos por deputado.

Giacobo foi o parlamentar com a maior quantidade de gastos, totalizando R$ 1.289.809,76 em gastos, seguido por Silas Câmara com R$ 1.177.321,09 e Gustinho Ribeiro com R$ 1.061.509,11.

Com relação aos fornecedores, as 4 maiores são empresas de companhias aéreas. A TAM foi disparadamente a maior, com gastos totalizando R$ 26.192.900,91, seguida por Latam Linhas Aéreas com R$ 2.829.783,31, Gol Linhas Aéreas com R$ 2.266.108,56 e Azul Linhas Aéreas com R$ 1.556.265,95.

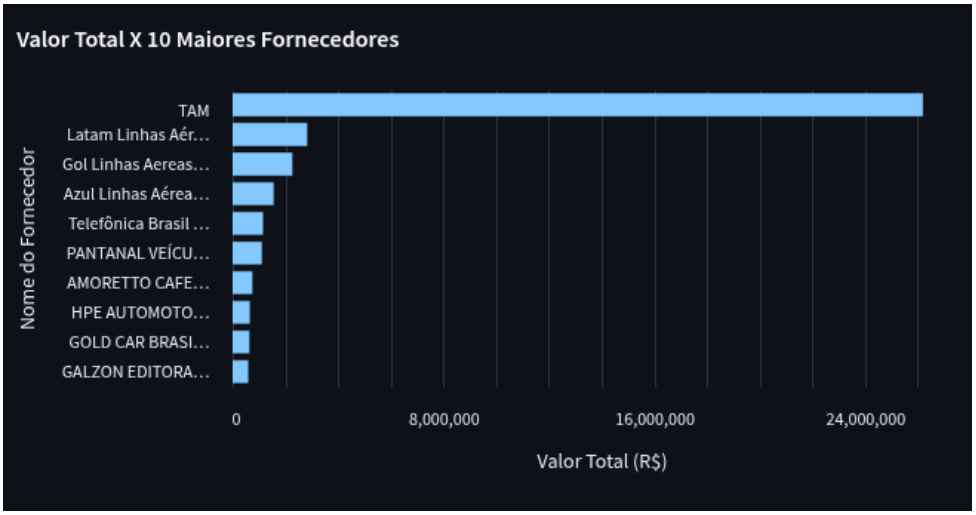

Figura 8: Valor dos gastos por fornecedor.

2.4 Considerações finais

O Streamlit é uma ferramenta poderosa para análise de dados, por meio dela podemos construir gráficos 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

Machine Learning (Aprendizado de Máquina) é um campo da inteligência artificial que permite aos sistemas aprender com dados, identificar padrões, tomar decisões e fazer predições futuras com mínima intervenção humana. Sua aplicação vai desde a detecção de anomalias e o reconhecimento de padrões em grandes volumes de dados até a otimização de processos e a personalização de experiências. É uma ferramenta poderosa para transformar dados brutos em insights acionáveis.

3.2 Dicionário de dados

No contexto do nosso projeto, foi desenvolvido um modelo de Machine Learning para identificar anomalias nos dados, utilizando o algoritmo Isolation Forest.

O Isolation Forest é um algoritmo de Machine Learning não supervisionado, amplamente utilizado para detecção de anomalias. Diferente de outros métodos que tentam "encontrar"a normalidade para depois identificar o que é diferente, o Isolation Forest adota uma abordagem mais direta: ele isola as anomalias.

Como funciona?

1. `Construção de Árvores Aleatórias`: O algoritmo constrói uma floresta de árvores de decisão. Para cada árvore, ele seleciona aleatoriamente um atributo e, em seguida, seleciona aleatoriamente um ponto de divisão dentro do intervalo de valores desse atributo.

2. `Isolamento de Pontos`: Esse processo de divisão continua até que cada ponto de dado esteja isolado ou um limite de profundidade da árvore seja atingido.

3. `Identificação de Anomalias`: O princípio é que as anomalias, por serem pontos mais raros e diferentes, geralmente exigem menos divisões para serem isoladas nas árvores. Pontos normais, por outro lado, são mais densos e requerem mais divisões para serem separados uns dos outros.

Vantagens do Isolation Forest:

- Eficiência: É computacionalmente mais eficiente do que muitos outros algoritmos de detecção de anomalias, especialmente para grandes conjuntos de dados.
- Escalabilidade: Lida bem com dados de alta dimensão.
- Baixo Consumo de Memória: Não exige o armazenamento de modelos de densidade complexos.
- Não Supervisionado: Não requer dados previamente rotulados como "normal"ou "anômalo"para treinar o modelo.

Em resumo, o Isolation Forest é uma escolha robusta para identificar comportamentos ou eventos incomuns que se desviam significativamente do padrão esperado nos 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] DBeaver: https://dbeaver.io/

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

nos graficos contar uma historia fazer mais comparações entre anos