**Explicação do Funcionamento do Código**

**1. Problemas com o Acesso à API do Twitter**

No início, tentamos utilizar a API do Twitter para obter informações em tempo real. No entanto, devido às limitações do plano gratuito da API do Twitter (ou X), não conseguimos acessar os endpoints necessários para buscar tendências e tweets. A API gratuita fornece acesso limitado e muitas funcionalidades estão disponíveis apenas para planos pagos.

**2. Armazenamento de Informações**

**Coleta de Dados do Twitter (collect\_data.py)**

O script collect\_data.py é utilizado para coletar dados do Twitter. Aqui está um resumo do seu funcionamento: em Python

**collect\_data.py**

from src.services import save\_tweets

save\_tweets()

**services.py**

import tweepy

from src.constants import (

CONSUMER\_KEY, CONSUMER\_SECRET, ACCESS\_TOKEN, ACCESS\_TOKEN\_SECRET,

MONGO\_CONNECTION\_STRING, DB\_NAME, COLLECTION\_NAME

)

from pymongo import MongoClient

# Autenticação com a API do Twitter

auth = tweepy.OAuth1UserHandler(CONSUMER\_KEY, CONSUMER\_SECRET, ACCESS\_TOKEN, ACCESS\_TOKEN\_SECRET)

api = tweepy.API(auth)

# Conexão com o MongoDB

client = MongoClient(MONGO\_CONNECTION\_STRING)

db = client[DB\_NAME]

collection = db[COLLECTION\_NAME]

# Função para coletar tweets do timeline de um usuário

def \_get\_user\_timeline\_tweets(username, api):

tweets = api.user\_timeline(screen\_name=username, count=10)

return [{"text": tweet.text, "created\_at": tweet.created\_at} for tweet in tweets]

# Função para salvar os tweets no MongoDB

def save\_tweets():

tweets = \_get\_user\_timeline\_tweets(username="TwitterDev", api=api)

collection.insert\_many(tweets)

**Explicação**

Neste exemplo, estamos utilizando o MongoDB para armazenar os dados coletados da API do Twitter.

1. **Autenticação com a API do Twitter**:
   * Utilizamos a biblioteca tweepy para autenticação e coleta de dados.
   * As credenciais (CONSUMER\_KEY, CONSUMER\_SECRET, ACCESS\_TOKEN, ACCESS\_TOKEN\_SECRET) são usadas para autenticar com a API do Twitter.
2. **Conexão com o MongoDB**:
   * Utilizamos a biblioteca pymongo para conectar ao MongoDB.
   * As informações de conexão (MONGO\_CONNECTION\_STRING, DB\_NAME, COLLECTION\_NAME) são usadas para configurar a conexão e selecionar o banco de dados e a coleção onde os dados serão armazenados.
3. **Função \_get\_user\_timeline\_tweets**:
   * Esta função coleta os tweets mais recentes do timeline de um usuário específico (TwitterDev neste caso) e retorna uma lista de dicionários contendo o texto e a data de criação dos tweets.
4. **Função save\_tweets**:
   * Esta função chama \_get\_user\_timeline\_tweets para obter os tweets e, em seguida, insere esses tweets na coleção do MongoDB.

**Alternativa sem API do Twitter**

**Dashboard (dashboard.py)**

**import plotly.graph\_objs as go**

**from plotly.subplots import make\_subplots**

**import random**

**# Dados fictícios**

**trend\_names = ["#Python", "#DataScience", "#AI", "#MachineLearning", "#DeepLearning", "#BigData", "#Analytics", "#Programming", "#Coding", "#Tech"]**

**likes = [random.randint(100, 1000) for \_ in trend\_names]**

**retweets = [random.randint(50, 500) for \_ in trend\_names]**

**engagement = [likes[i] + retweets[i] for i in range(len(trend\_names))]**

**# Criação dos subplots**

**fig = make\_subplots(**

**rows=2, cols=2,**

**specs=[[{"type": "bar"}, {"type": "pie"}],**

**[{"type": "scatter"}, {"type": "table"}]],**

**subplot\_titles=("Gráfico de Barras", "Gráfico de Pizza", "Gráfico de Dispersão", "Mini Relatório"),**

**horizontal\_spacing=0.1,**

**vertical\_spacing=0.15**

**)**

**# Gráfico de Barras**

**fig.add\_trace(**

**go.Bar(x=trend\_names, y=likes, name="Likes", marker=dict(color='lightblue')),**

**row=1, col=1**

**)**

**# Gráfico de Pizza**

**fig.add\_trace(**

**go.Pie(labels=trend\_names, values=engagement, name="Engajamento"),**

**row=1, col=2**

**)**

**# Gráfico de Dispersão (Scatter Plot)**

**fig.add\_trace(**

**go.Scatter(x=trend\_names, y=retweets, mode='markers', name="Retweets", marker=dict(size=10, color='orange')),**

**row=2, col=1**

**)**

**# Mini Relatório**

**fig.add\_trace(**

**go.Table(**

**header=dict(values=["Tendência", "Likes", "Retweets", "Engajamento"], fill\_color='paleturquoise', align='left'),**

**cells=dict(values=[trend\_names, likes, retweets, engagement], fill\_color='lavender', align='left')**

**),**

**row=2, col=2**

**)**

**# Layout**

**fig.update\_layout(**

**height=800,**

**title\_text="Dashboard Personalizado",**

**title\_x=0.5,**

**plot\_bgcolor='lightyellow',**

**paper\_bgcolor='lightcyan',**

**margin=dict(t=50, b=50, l=25, r=25),**

**showlegend=False,**

**shapes=[**

**go.layout.Shape(type="line", x0=0.5, y0=0, x1=0.5, y1=1, line=dict(color="Black", width=2)),**

**go.layout.Shape(type="line", x0=0, y0=0.5, x1=1, y1=0.5, line=dict(color="Black", width=2))**

**]**

**)**

**# Anotações**

**for annotation in fig['layout']['annotations']:**

**annotation['font'] = dict(size=14, color='black')**

### **fig.show()**

### **Explicação do Dashboard**

* **Dados Fictícios**:
  + Usamos dados gerados aleatoriamente para simular likes, retweets e engajamento de várias tendências.
* **Criação de Gráficos**:
  + Criamos diferentes tipos de gráficos (barras, pizza, dispersão, tabela) e adicionamos aos subplots.
* **Layout e Estilo**:
  + Configuramos o layout para incluir título, bordas visíveis entre os subplots, e cores de fundo para melhorar a visualização.

**Conclusão**

Embora não tenhamos conseguido acessar a API do Twitter devido a limitações do plano, conseguimos criar um dashboard funcional usando dados fictícios para demonstrar a funcionalidade. O MongoDB seria utilizado para armazenar dados reais, caso tivéssemos acesso completo à API. Este exemplo serve como uma base sólida que pode ser expandida quando o acesso à API for obtido ou ao integrar outras fontes de dados