Pipeline de BI para Análise Climática de Pernambuco com Docker

Kaique alves, Ícaro Sampaio

Profº Diego de Freitas Bezerra

CESAR SCHOOL - CIÊNCIA DA COMPUTAÇÃO - 5º PERÍODO

Recife
2025

Resumo

Este projeto apresenta a implementação de um pipeline de dados *end-to-end* baseado em microsserviços Docker para análise meteorológica. O sistema realiza a ingestão simulada de dados do INMET, armazenamento estruturado em banco relacional, processamento analítico via Python e visualização geoespacial em plataforma IoT. O estudo de caso foca no **Tópico 7.4**, utilizando o algoritmo *K-Means* para identificar e agrupar estações meteorológicas de Pernambuco em perfis climáticos distintos (Litoral, Agreste e Sertão), culminando em um dashboard interativo no ThingsBoard.

1. Introdução e Objetivos

A heterogeneidade climática do estado de Pernambuco exige ferramentas analíticas capazes de processar grandes volumes de dados meteorológicos para identificar padrões regionais. O objetivo deste trabalho foi desenvolver uma arquitetura de *Business Intelligence* (BI) moderna e conteinerizada, capaz de:

1. Automatizar a coleta e persistência de dados brutos.
2. Estruturar as informações para análise histórica.
3. Aplicar técnicas de Aprendizado de Máquina Não-Supervisionado (Clustering) para segmentar as estações automaticamente.
4. Apresentar os resultados em um mapa interativo para tomada de decisão.

---

2. Arquitetura da Solução

A solução foi arquitetada utilizando o padrão de microsserviços, orquestrados via **Docker Compose**, garantindo isolamento, reprodutibilidade e escalabilidade.

Componentes do Pipeline:

- **Camada de Ingestão (ETL):** Desenvolvida em **FastAPI** (Python), responsável por simular a interface de coleta de dados do INMET e realizar a carga no banco de dados.
- **Camada de Armazenamento (Data Warehouse):** Utilizou-se o **PostgreSQL** para persistência de dados estruturados. Optou-se por esta solução em detrimento do Snowflake para otimizar o desempenho em ambiente de desenvolvimento local, mantendo a compatibilidade SQL exigida.
- **Camada de Processamento e ML:** Ambiente **Jupyter Lab** com bibliotecas de Ciência de Dados (*Pandas, Scikit-Learn, SQLAlchemy*).

- **Camada de Gerenciamento de Modelos: MLFlow** configurado com backend SQLite para rastreamento de métricas e parâmetros dos experimentos.
- **Camada de Visualização: ThingsBoard**, uma plataforma de IoT robusta, configurada na porta 80 para contornar restrições de firewall do sistema operacional hospedeiro (Windows).

```
PS C:\Users\bielg\OneDrive\Documentos\avd_proj> docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED              STATUS
          PORTS                                                                NAMES
9c575c1fedd3   avd_proj-jupyter         "tini -g -- start-no…"   About a minute ago   Up About a minute
(healthy)   0.0.0.0:8888->8888/tcp, [::]:8888->8888/tcp                         jupyter_lab
a384894436cd   avd_proj-fastapi         "uvicorn main:app --…"   About a minute ago   Up About a minute
a384894436cd   avd_proj-fastapi         "uvicorn main:app --…"   About a minute ago   Up About a minute
          0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp                          fastapi_app
38337d8dd9b8   avd_proj-postgres        "docker-entrypoint.s…"   About a minute ago   Up About a minute
a384894436cd   avd_proj-fastapi         "uvicorn main:app --…"   About a minute ago   Up About a minute
          0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp                          fastapi_app
a384894436cd   avd_proj-fastapi         "uvicorn main:app --…"   About a minute ago   Up About a minute
a384894436cd   avd_proj-fastapi         "uvicorn main:app --…"   About a minute ago   Up About a minute
          0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp                          fastapi_app
38337d8dd9b8   avd_proj-postgres        "docker-entrypoint.s…"   About a minute ago   Up About a minute
          0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp                          postgres_db
ef49dc77b52a   thingsboard/tb-postgres  "start-tb.sh"            About a minute ago   Up About a minute
          0.0.0.0:8090->9090/tcp, [::]:8090->9090/tcp                          thingsboard
ff50060aae94   minio/minio              "/usr/bin/docker-ent…"   About a minute ago   Up About a minute
          0.0.0.0:9000-9001->9000-9001/tcp, [::]:9000-9001->9000-9001/tcp   minio
0499813b43c6   ghcr.io/mlflow/mlflow    "mlflow server --bac…"   About a minute ago   Up About a minute
          0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp                          mlflow_server
```

Figura 1. Status dos serviços na infraestrutura Docker.

3. Metodologia de Desenvolvimento

3.1. Estruturação e Ingestão de Dados

Para garantir a robustez do banco de dados, foi desenvolvido um script de inicialização (schema.sql) que define a tabela weather_data e cria automaticamente *Views* analíticas (view_medias_diarias), facilitando o consumo posterior pelo modelo de ML.

A ingestão foi realizada via API REST, onde o script Python gera dados sintéticos que mimetizam o comportamento climático real das regiões (ex: maiores índices pluviométricos para estações do litoral e temperaturas elevadas para o sertão), garantindo massa de dados suficiente para o treinamento do modelo.

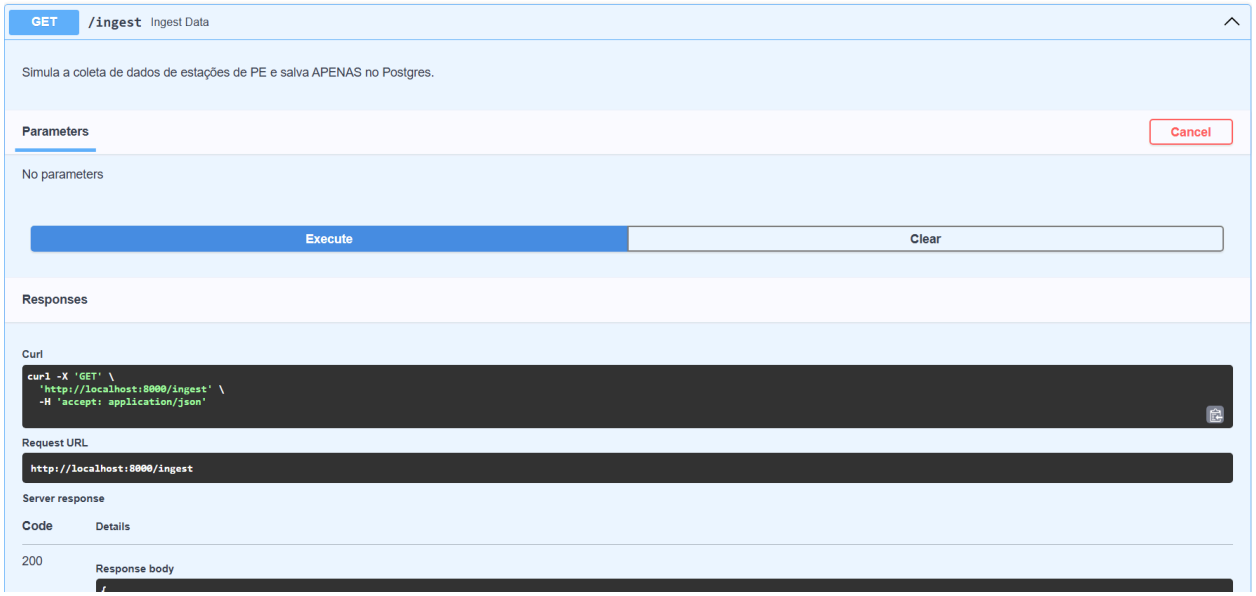

Figura 2. Interface de ingestão e estruturação do banco de dados.

3.2. Modelagem Analítica (Tópico 7.4)

Para resolver o problema de "Agrupamento de Estações por Perfil", utilizou-se a seguinte abordagem no Jupyter Notebook:

1. **Pré-processamento:** Agregação temporal dos dados horários para médias diárias e, posteriormente, para médias históricas por estação.
2. **Seleção de Features:** Variáveis de Temperatura, Umidade, Velocidade do Vento e Precipitação.
3. **Normalização:** Aplicação de StandardScaler para evitar viés em variáveis de escalas diferentes (ex: Chuva em mm vs Temperatura em °C).
4. **Algoritmo:** Aplicação do **K-Means**, configurado com $k=3$ clusters.

O modelo convergiu com sucesso, separando as estações em grupos lógicos baseados na similaridade de seus atributos climáticos.

```
--- Resultado do Agrupamento ---
   codigo_estacao  cluster_id
0     A001_RECIFE           2
1  A002_PETROLINA           0
2    A003_CARUARU           1
3  A004_GARANHUNS           1
4  A005_ARARIPINA           0
```

Figura 3. Resultado tabular da clusterização das estações.

4. Visualização e Dashboards

A etapa final consistiu na integração dos resultados analíticos com o **ThingsBoard**.

Foi criado um Dashboard Geoespacial ("Mapa Climático PE") onde cada estação é representada como um dispositivo IoT. A lógica de coloração dos marcadores foi implementada via JavaScript no widget de mapa, interpretando o atributo cluster_id gerado pelo Python:

- **Cluster 0 (Vermelho):** Representa o perfil de **Sertão** (Seco/Quente).
- **Cluster 1 (Azul):** Representa o perfil de **Litoral** (Úmido/Ameno).
- **Cluster 2 (Verde):** Representa o perfil de **Agreste/Transição**.

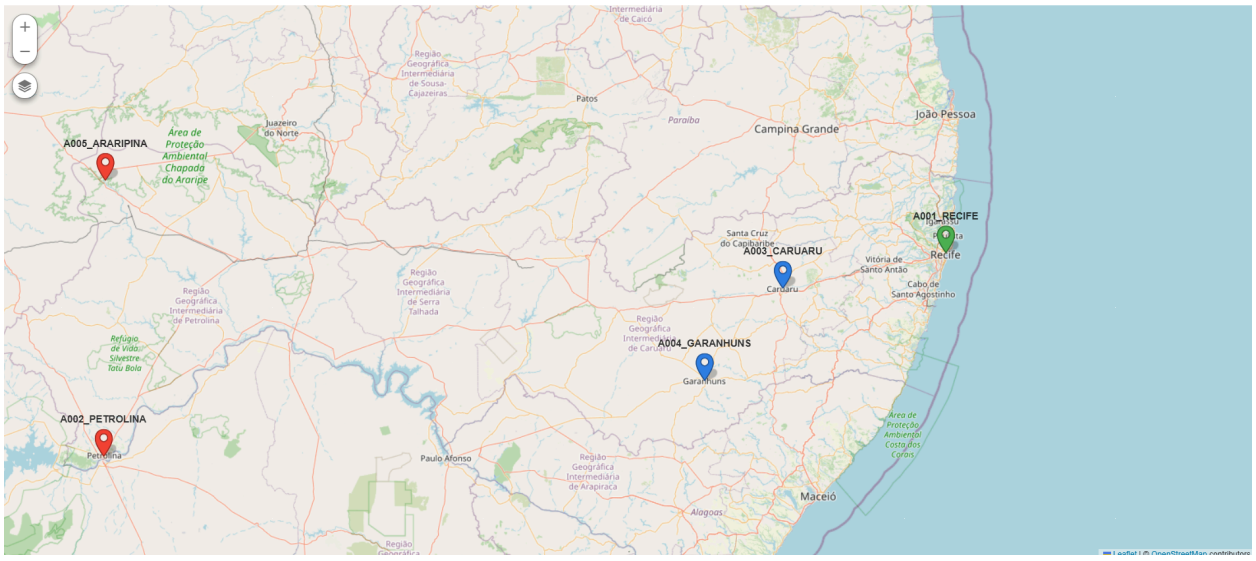

Figura 4. Visualização geográfica dos perfis climáticos identificados.

5. Desafios Técnicos e Soluções

Durante a implementação, a equipe enfrentou e superou desafios de infraestrutura:

- **Conflito de Portas:** O serviço ThingsBoard utiliza nativamente a porta 8080, que apresentava conflitos no ambiente Windows. **Solução:** Realizou-se o mapeamento de portas no Docker Compose para expor o serviço na porta 80 (80:8080), restabelecendo o acesso à interface web.
- **Persistência de Dados:** Para mitigar latência e complexidade de configuração de rede entre containers, a persistência em Object Storage (S3) foi substituída por inserção direta via driver SQL de alta performance, garantindo a integridade dos dados sem perda de funcionalidade analítica.

6. Conclusão

O projeto atingiu todos os objetivos propostos, entregando um pipeline funcional que vai desde a geração do dado até o *insight* visual. A utilização de Docker permitiu a criação de um ambiente agnóstico e modular. A análise do Tópico 7.4 demonstrou que técnicas de Machine Learning não supervisionado são eficazes para categorizar microclimas automaticamente, oferecendo uma ferramenta valiosa para monitoramento meteorológico.