	Especificação de Projeto	
C.E.S.A.R school	**DISCIPLINA:** Análise e Visualização de Dados	**PERÍODO:** 2025.2
		UNIDADE: 2a

1. Introdução

Este projeto visa o desenvolvimento de um pipeline completo de Business Intelligence (BI) voltado à análise e visualização de dados meteorológicos obtidos do **INMET** (Instituto Nacional de Meteorologia). A proposta envolve o uso de tecnologias para coleta, tratamento, integração e visualização de dados, dentro de uma arquitetura em contêineres baseada em Docker, conforme ilustrado na Figura 1.

Além disso, o projeto visa integrar conceitos de análise exploratória, modelagem preditiva e visualização de dados, desenvolvendo habilidades práticas de engenharia de dados e BI.

O pipeline (Figura 1) deverá abranger desde a coleta de dados e ingestão via FastAPI, armazenamento em MinIO/S3 e Snowflake, até o treinamento de modelos de machine learning em Jupyter Notebooks e publicação de dashboards interativos no ThingsBoard ou Trendz Analytics.

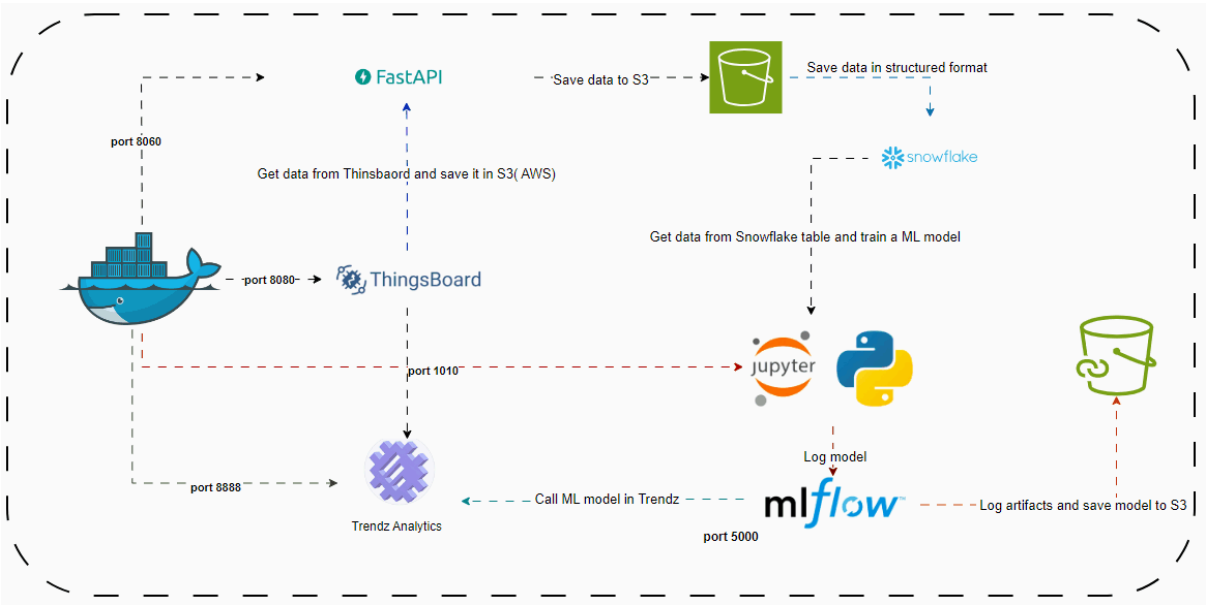

Figura 1 – Arquitetura proposta do pipeline de BI meteorológico, integrando coleta, processamento e visualização em contêineres.

O projeto deverá ser desenvolvido em grupos de até 6 integrantes e corresponderá a 100% da nota da segunda unidade. A avaliação será composta pela entrega técnica, escrita de um relatório e qualidade da documentação.

2. Objetivos

Desenvolver um pipeline de análise e visualização que integre:

- **Coleta** de dados meteorológicos do INMET (API ou CSV);
- **Armazenamento estruturado** em banco local e/ou S3;
- **Tratamento e limpeza** de dados (ex: remoção de nulos, interpolação de horários);
- **Modelagem preditiva simples** (regressão, classificação, séries temporais);
- **Visualização interativa** de dados brutos e resultados de modelos via ThingsBoard e Trendz Analytics.

Para implementação do pipeline, as seguintes tarefas deverão ser executadas:

- Criar um fluxo de ingestão via **FastAPI**, salvando dados em **S3/MinIO**;
- Estruturar os dados no **Snowflake** (ou base local em **SQLite/PostgreSQL**);
- Processar e modelar dados com **Python e Jupyter Notebook**;
- Registrar experimentos no **MLFlow** (port 5000);
- Exibir resultados e predições em dashboards **ThingsBoard (8080)** e **Trendz (8888)**.

3. Avaliação

A nota final do projeto será composta por entrega técnica, dashboard funcional, e relatório técnico descritivo, considerando também aspectos de organização, clareza e integração entre as camadas do pipeline. **A data final para entrega será dia 04/12 via Github e Google Classroom.**

Critério	Descrição	Peso
Integração entre camadas	Coerência e funcionamento do pipeline completo (coleta, tratamento, modelagem, visualização).	2.0
Tratamento e limpeza de dados	Identificação e correção de inconsistências, uso de técnicas adequadas de pré-processamento e enriquecimento de dados.	1.5
Modelagem e inteligência aplicada	Aplicação correta e interpretação dos modelos de aprendizado de máquina (regressão, classificação ou séries temporais).	1.5
Visualizações e dashboards	Qualidade, estética e clareza das visualizações no ThingsBoard/Trendz, demonstrando capacidade de análise exploratória e preditiva.	2.0

Relatório técnico (modelo)	Documento técnico explicando todo o pipeline: objetivos, arquitetura, tratamento de dados, modelo aplicado, resultados obtidos e conclusões. Deve conter capturas de tela do dashboard e exemplos de visualizações.	2.0
Organização e documentação (README, estrutura do repositório)	Clareza das instruções, padronização das pastas, uso de Docker Compose e versionamento no GitHub.	1.0

Importante! O relatório deverá ser entregue em formato PDF junto ao repositório GitHub e conter:

1. Introdução e objetivos do trabalho.
2. Descrição da arquitetura e das ferramentas utilizadas.
3. Metodologia de tratamento e modelagem de dados.
4. Análises e resultados (incluindo gráficos e tabelas).
5. Dashboard e insights obtidos.
6. Conclusões e possíveis melhorias futuras.

4. Requisitos Técnicos

Os dados devem ser obtidos das estações automáticas do INMET utilizando dados horários contendo, no mínimo, as variáveis: temperatura do ar, umidade relativa, pressão atmosférica, direção e velocidade do vento, radiação solar e precipitação. O grupo deverá usar os dados das estações meteorológicas do estado de Pernambuco, resolvendo um dos problemas descritos na Seção 7.

A arquitetura mínima do projeto deverá conter os seguintes serviços (em Docker Compose):

Serviço	Função Principal
FastAPI	Interface de ingestão dos dados do INMET e integração com S3
MinIO ou AWS S3	Armazenamento de dados brutos e modelos
Snowflake	Estruturação de dados tratados
Jupyter Notebook	Ambiente de análise, limpeza e modelagem preditiva
MLFlow	Registro e versionamento dos modelos de ML

ThingsBoard/Trendz Analytics	Visualização dos dados e dashboards interativos

Fluxo geral:

1. O FastAPI recebe os dados do INMET (API/CSV) e armazena em S3.
2. Os dados são estruturados em Snowflake.
3. Jupyter Notebook lê da base estruturada, trata e treina um modelo.
4. O modelo é versionado no MLFlow e exportado novamente para o S3.
5. O dashboard (ThingsBoard/Trendz) consome os dados e mostra visualizações e insights.

5. Estrutura do Repositório

A estrutura de pastas sugerida para o repositório no Github é a seguinte:

```
/repo
├── docker-compose.yml    # Orquestração dos contêineres
├── jupyterlab/           # Ambiente de análise e exploração (Dockerfile e configs)
├── mlflow/               # Configuração e armazenamento de experimentos
├── fastapi/              # Camada de ingestão (API)
├── notebooks/            # Notebooks de tratamento, visualização e modelagem
├── sql_scripts/          # Scripts SQL de estruturação e consultas
├── trendz/               # Dashboards exportados
├── reports/              # Relatórios e resultados
├── README.md             # Descrição do projeto
└── LICENSE               # Licença
```

O arquivo de licença é opcional, mas caso seja definido, licenças como MIT e BSD são recomendadas. O arquivo `README.md` deverá conter as seguintes informações:

1. Nome e sobrenome dos membros do projeto e seus respectivos **usuários no GitHub (@fulano, @beltrano, @sicrano)**.
2. Nome da disciplina: **Análise e Visualização de Dados - 2025.2**.
3. Nome da instituição de ensino: **CESAR School**.
4. Instruções detalhadas para levantar a infraestrutura, executar e visualizar o dashboard.

6. Itens obrigatórios de entrega

Cada grupo deverá entregar:

1. **Pipeline executável via Docker Compose**, com os serviços descritos na Seção 4.
2. **Repositório no GitHub** contendo:
    - Código fonte (FastAPI, Notebooks, etc.);
    - Docker Compose funcional;
    - README com instruções de execução;

    - Relatório técnico em PDF (em `/reports`).
3. **Dashboard online** (no ThingsBoard/Trendz).
4. **Apresentação oral** com demonstração do pipeline e discussão dos resultados.

7. Sugestões de Métricas e Modelos

Cada grupo deverá escolher **um problema meteorológico** que envolva tratamento de dados, cálculo de métricas e aplicação de uma técnica de aprendizado de máquina. A solução deve ser integrada ao pipeline proposto e apresentar resultados no dashboard.

Abaixo estão **10 propostas possíveis**, cobrindo técnicas de **agrupamento, classificação e regressão**:

7.1. Agrupar Padrões Climáticos por Horário

- **Objetivo:** Agrupar horas do dia com comportamento climático semelhante.
- **Dados:** Temperatura, umidade e radiação solar horária.
- **Visualização:** Gráfico de dispersão colorido por cluster + painel com médias de cada grupo.

7.2. Classificar Dias Chuvosos vs. Ensolarados

- **Objetivo:** Prever se um dia será “chuvoso” ou “ensolarado”.
- **Dados:** Temperatura máxima, umidade média, pressão e radiação solar.
- **Visualização:** Barras de frequência por classe + probabilidade de chuva.

7.3. Prever Temperatura Horária

- **Objetivo:** Prever a temperatura de uma hora futura com base em variáveis meteorológicas.
- **Dados:** Temperatura, umidade, vento, radiação solar e precipitação.
- **Visualização:** Série temporal com curva real vs. prevista + erro médio (RMSE).

7.4. Agrupar Estações Meteorológicas por Perfil

- **Objetivo:** Identificar grupos de estações com padrões climáticos semelhantes.
- **Dados:** Médias diárias de temperatura, chuva, vento e umidade.
- **Visualização:** Mapa ou gráfico de clusters + tabela com características médias.

7.5. Classificar Níveis de Conforto Térmico

- **Objetivo:** Classificar períodos em níveis de conforto (“frio”, “confortável”, “quente”).
- **Dados:** Temperatura, umidade e velocidade do vento.
- **Visualização:** Gauge com classes + gráfico de linha mostrando transições entre zonas de conforto.

7.6. Prever Umidade Relativa

- **Objetivo:** Estimar a umidade relativa do ar com base na temperatura e pressão.
- **Dados:** Temperatura, pressão atmosférica e radiação solar.
- **Visualização:** Gráfico de linha (real vs. previsto) + indicador de erro médio (MAE).

7.7. Agrupar Padrões de Vento

- **Objetivo:** Agrupar horários ou dias com comportamentos semelhantes de vento.
- **Dados:** Direção e velocidade do vento.
- **Visualização:** Rosa dos ventos colorida por cluster + painel com médias por grupo.

7.8. Classificar Intensidade da Chuva

- **Objetivo:** Classificar a intensidade da precipitação (“sem chuva”, “leve”, “moderada”, “forte”).
- **Dados:** Precipitação, pressão atmosférica, umidade e vento.
- **Visualização:** Gráfico de barras por classe + linha temporal colorida por categoria.

7.9. Agrupar Condições Extremas

- **Objetivo:** Detectar padrões que indiquem condições extremas (altas temperaturas, ventos fortes).
- **Dados:** Temperatura, umidade, vento e radiação solar.
- **Visualização:** Gráfico de dispersão 3D (variáveis principais) + contadores de registros extremos.

7.10. Prever Sensação Térmica

- **Objetivo:** Estimar a sensação térmica percebida a partir de variáveis climáticas.
- **Dados:** Temperatura, umidade e vento.
- **Visualização:** Curva real vs. prevista + importância das variáveis na árvore.

Bom trabalho!