CENTRO UNIVERSITÁRIO UNIFECAF
GRADUAÇÃO EAD

Weslei Luciano de Sousa
Oliveira
RA: 85431

Pipeline de Dados com IoT e Docker: Análise de Leituras de Temperatura

BRASÍLIA, 2025

Weslei Luciano de Sousa
Oliveira
RA: 85431

PIPELINE DE DADOS COM IOT E DOCKER: ANÁLISE DE LEITURAS DE TEMPERATURA

Trabalho apresentado como requisito parcial de avaliação da disciplina de Disruptive Architectures: IOT, Big Data e IA do Curso de Graduação em Tecnologia em **Análise e Desenvolvimento de Sistemas** do Centro Universitário UniFECAF.

Tutor: Felipe Bonatto

BRASÍLIA, 2025

SUMÁRIO

1. CONTEXTUALIZAÇÃO DO PROJETO

A era da transformação digital é marcada pela proliferação de dispositivos de Internet das Coisas (IoT), que conectam o mundo físico ao digital por meio de sensores que coletam dados continuamente. Em setores que vão da indústria 4.0 e cidades inteligentes à agricultura de precisão e monitoramento de saúde, esses dispositivos geram um volume massivo e constante de informações. O grande desafio, no entanto, não reside apenas na coleta, mas no processamento eficiente e na extração de valor desses dados brutos. Sem uma arquitetura de dados bem estruturada, essa avalanche de informações pode se tornar mais um obstáculo do que uma vantagem competitiva.

Este projeto aborda diretamente esse desafio, focando em um dos casos de uso mais comuns e críticos da IoT: o monitoramento de leituras de temperatura. Utilizando um conjunto de dados público com quase 100.000 registros de sensores, o trabalho simula um cenário real onde é fundamental acompanhar as variações de temperatura para garantir a qualidade de processos, a segurança de equipamentos ou o conforto de ambientes.

Para solucionar este problema, a proposta é a construção de um *pipeline de dados* completo, moderno e escalável. Um pipeline de dados é uma série de etapas que movem os dados de uma fonte para um destino, aplicando transformações ao longo do caminho. A arquitetura escolhida para este projeto integra tecnologias de ponta, cada uma com um papel estratégico:

1. **Python**, com as bibliotecas **Pandas** e **SQLAlchemy**, foi selecionado como a linguagem central para a ingestão e tratamento dos dados. Sua versatilidade e o poder de seu ecossistema o tornam ideal para ler, limpar e estruturar grandes volumes de informação.
2. **Docker** é utilizado para criar um ambiente de banco de dados isolado e replicável. Através da conteinerização do **PostgreSQL**, garantimos que o sistema de armazenamento seja consistente em qualquer máquina, eliminando problemas de configuração e facilitando a implantação.
3. **PostgreSQL** atua como o sistema de gerenciamento de banco de dados relacional, oferecendo uma solução robusta e confiável para armazenar os dados estruturados de temperatura de forma persistente e segura.
4. **Streamlit**, um framework Python, é a ferramenta escolhida para a camada de visualização. Ele permite a criação rápida de dashboards web interativos, transformando as consultas complexas ao banco de dados em gráficos e visualizações intuitivas.

O objetivo final, portanto, é demonstrar na prática como transformar um grande volume de dados brutos de sensores em uma ferramenta de análise visual e acionável. Ao final deste projeto, teremos um dashboard funcional que não apenas exibe os dados, mas também permite a identificação de padrões, anomalias e tendências, provando o valor de

uma arquitetura de dados bem planejada no universo da Internet das Coisas.

2. DESCRIÇÃO DOS PASSOS REALIZADOS

A materialização deste pipeline de dados seguiu uma metodologia estruturada, dividida em cinco etapas lógicas e sequenciais, garantindo a correta implementação desde a base do ambiente até a camada final de visualização. Cada passo foi fundamental para construir uma solução coesa e funcional.

a) Configuração do Ambiente de Desenvolvimento O alicerce de qualquer projeto de software robusto é um ambiente de desenvolvimento bem configurado e replicável. O primeiro passo consistiu na instalação das ferramentas essenciais: **Python**, como linguagem de programação principal, e **Docker Desktop**, para a gestão de contêineres. Para evitar conflitos de dependências e garantir que o projeto pudesse ser facilmente executado em outras máquinas, foi criado um ambiente virtual Python (venv). Dentro deste ambiente isolado, todas as bibliotecas necessárias, como pandas, sqlalchemy, psycopg2-binary, streamlit e plotly, foram instaladas de forma automatizada através do comando pip install -r requirements.txt, assegurando a consistência das versões e a reprodutibilidade do projeto.

b) Criação do Contêiner Docker com PostgreSQL Para o armazenamento dos dados de IoT, a escolha foi o **PostgreSQL**, um sistema de gerenciamento de banco de dados relacional conhecido por sua robustez e confiabilidade. Em vez de uma instalação local tradicional, optou-se por uma abordagem moderna utilizando **Docker**. Através do arquivo de configuração docker-compose.yml, foi definido um serviço de banco de dados conteinerizado. Este arquivo automatizou todo o processo de:

- Baixar a imagem oficial do PostgreSQL (versão 13).
- Configurar as variáveis de ambiente essenciais, como usuário, senha e nome do banco.
- Mapear a porta 5432 do contêiner para a máquina local, permitindo que os scripts Python se conectassem ao banco.
- Criar um volume nomeado (postgres_data) para garantir a persistência dos dados, de modo que as informações não fossem perdidas ao reiniciar o contêiner. Esta abordagem garantiu um banco de dados isolado, portátil e de fácil configuração.

c) Desenvolvimento do Pipeline de Ingestão (pipeline.py) O script pipeline.py representa o núcleo do processo de Extração, Transformação e Carga (ETL). Sua execução foi responsável por popular o banco de dados e prepará-lo para a análise. As principais funcionalidades implementadas foram:

1. **Conexão Segura:** O script utiliza a biblioteca SQLAlchemy para criar uma engine de conexão com o banco de dados PostgreSQL em execução no contêiner Docker.
2. **Criação da Tabela:** Uma função verifica se a tabela temperature_readings já existe e, caso contrário, a cria com a estrutura correta para receber os dados dos sensores.
3. **Leitura e Transformação:** Utilizando a biblioteca Pandas, o script lê o conjunto de dados do arquivo temperature_readings.csv. Em seguida, realiza transformações cruciais: renomeia as colunas para um padrão mais claro e, mais importante, converte a coluna de data/hora para o formato timestamp do Python, especificando o parâmetro dayfirst=True para interpretar corretamente o formato de data Dia-Mês-Ano presente no arquivo.
4. **Carga dos Dados:** Por fim, o DataFrame tratado, contendo mais de 97.000 registros, é inserido de forma eficiente na tabela temperature_readings do banco de dados.

d) Criação das Views SQL (create_views.sql) Com os dados brutos já armazenados, o próximo passo foi otimizar as futuras consultas. Em vez de realizar cálculos complexos e repetitivos diretamente no dashboard, foram criadas três **Views SQL**. As views são, essencialmente, consultas salvas que se comportam como tabelas virtuais. O arquivo create_views.sql contém os comandos para criar visões que agregam os dados de formas específicas: uma para calcular a temperatura média por dispositivo, outra para contar o número de leituras por hora e uma terceira para encontrar as temperaturas máxima e mínima de cada dia. Esta abordagem desacopla a lógica de agregação do código da aplicação, melhora a performance e simplifica as consultas no dashboard.

e) Construção do Dashboard Interativo (dashboard.py) A etapa final foi a criação da interface de visualização. O script dashboard.py utiliza o framework **Streamlit** para construir uma aplicação web interativa de forma rápida e eficiente. O script executa as seguintes ações:

1. Estabelece uma conexão com o banco de dados PostgreSQL.
2. Executa consultas simples (SELECT * FROM ...) nas três views SQL previamente criadas, carregando os dados agregados em DataFrames do Pandas.
3. Utiliza a biblioteca **Plotly Express** para renderizar três tipos de gráficos interativos: um gráfico de barras, um gráfico de linha e um gráfico de múltiplas linhas. Ao ser executado com o comando streamlit run src/dashboard.py, o script inicia um servidor local e exibe o dashboard diretamente no navegador, apresentando os resultados da análise de forma clara e profissional.

3. EXPLICAÇÃO DETALHADA DAS VIEWS SQL

Uma estratégia fundamental na arquitetura deste pipeline de dados foi a otimização das consultas através do uso de **Views SQL** no PostgreSQL. Em vez de sobrecarregar a aplicação de dashboard com cálculos e agregações complexas a cada carregamento, a lógica de transformação foi delegada ao banco de dados. As views funcionam como tabelas virtuais, pré-processando e sumarizando os dados brutos da tabela temperature_readings. Esta abordagem não apenas melhora significativamente a performance do dashboard, mas também simplifica o código Python, tornando-o mais limpo e focado exclusivamente na visualização.

Foram criadas três views estratégicas, cada uma projetada para responder a uma pergunta de negócio específica sobre os dados dos sensores.

a) View: avg_temp_por_dispositivo

- **Propósito:** O objetivo principal desta view é consolidar todas as leituras de temperatura para cada dispositivo individual e calcular a sua média operacional. Ela responde à pergunta: "Qual é a temperatura média de funcionamento de cada sensor?".
- **Utilidade Prática:** Em um cenário real, esta visualização é uma ferramenta de diagnóstico poderosa. Ela permite comparar diretamente o comportamento térmico dos dispositivos, facilitando a identificação de anomalias. Um sensor com uma média de temperatura consistentemente mais alta pode indicar um problema de hardware, uma localização em um ambiente mal ventilado ou uma carga de trabalho excessiva. Portanto, essa view serve como um indicador de primeira linha para manutenções preditivas e para garantir a saúde da infraestrutura de IoT.
- **Análise do Código SQL:**
- CREATE OR REPLACE VIEW avg_temp_por_dispositivo AS
- SELECT
- device_id,
- AVG(temperature) AS avg_temp
- FROM
- temperature_readings
- GROUP BY
- device_id
- ORDER BY
- avg_temp DESC;

O comando GROUP BY device_id agrupa todas as linhas pertencentes ao mesmo sensor. A função de agregação AVG(temperature) então calcula a média para

cada um desses grupos. Finalmente, ORDER BY avg_temp DESC organiza o resultado de forma decrescente, destacando imediatamente os dispositivos que operam em temperaturas mais elevadas.

b) View: leituras_por_hora

- **Propósito:** Esta view foi projetada para analisar a distribuição temporal das coletas de dados, agrupando e contando o número total de leituras para cada hora do dia (de 0 a 23). Ela responde à pergunta: "Quais são os horários de maior e menor atividade dos sensores?".
- **Utilidade Prática:** A análise da frequência de leituras ao longo do dia é crucial para entender o padrão operacional da rede de sensores. Ela pode confirmar se os dispositivos estão reportando dados na frequência esperada e revelar picos de atividade que podem estar correlacionados com eventos específicos. Essa informação é valiosa para o planejamento de janelas de manutenção (em horários de baixa atividade) e para a otimização da carga no banco de dados.
- **Análise do Código SQL:**
- CREATE OR REPLACE VIEW leituras_por_hora AS
- SELECT
- EXTRACT(HOUR FROM timestamp) AS hora,
- COUNT(*) AS contagem
- FROM
- temperature_readings
- GROUP BY
- hora
- ORDER BY
- hora ASC;

A função EXTRACT(HOUR FROM timestamp) é o elemento chave aqui, pois isola apenas a porção da hora da data completa. O GROUP BY hora agrupa todos os registros da mesma hora, e COUNT(*) conta quantas leituras ocorreram em cada um desses grupos. O resultado é ordenado por hora para uma visualização cronológica clara.

c) View: temp_max_min_por_dia

- **Propósito:** O objetivo desta view é sumarizar a volatilidade diária da temperatura, identificando os valores máximo (temp_max) e mínimo (temp_min) registrados para cada dia no conjunto de dados. Ela responde à pergunta: "Qual foi a amplitude térmica em cada dia?".
- **Utilidade Prática:** Esta é uma view essencial para o monitoramento de condições

ambientais e o controle de qualidade. Ela permite analisar a faixa de operação dos sensores e identificar rapidamente dias com picos ou quedas extremas de temperatura, que podem indicar eventos anômalos, falhas no sistema de climatização ou condições ambientais atípicas. Em muitos setores, manter a temperatura dentro de uma faixa específica é crítico, e esta view fornece essa visibilidade de forma direta.

- **Análise do Código SQL:**
- CREATE OR REPLACE VIEW temp_max_min_por_dia AS
- SELECT
- DATE(timestamp) AS data,
- MAX(temperature) AS temp_max,
- MIN(temperature) AS temp_min
- FROM
- temperature_readings
- GROUP BY
- data
- ORDER BY
- data ASC;

A função DATE(timestamp) trunca a informação de hora, permitindo que o GROUP BY data agrupe todos os registros do mesmo dia. As funções de agregação MAX(temperature) e MIN(temperature) encontram os valores extremos dentro de cada grupo diário, fornecendo um resumo conciso e poderoso da variação térmica ao longo do tempo.

4. VISUALIZAÇÕES GERADAS NO DASHBOARD

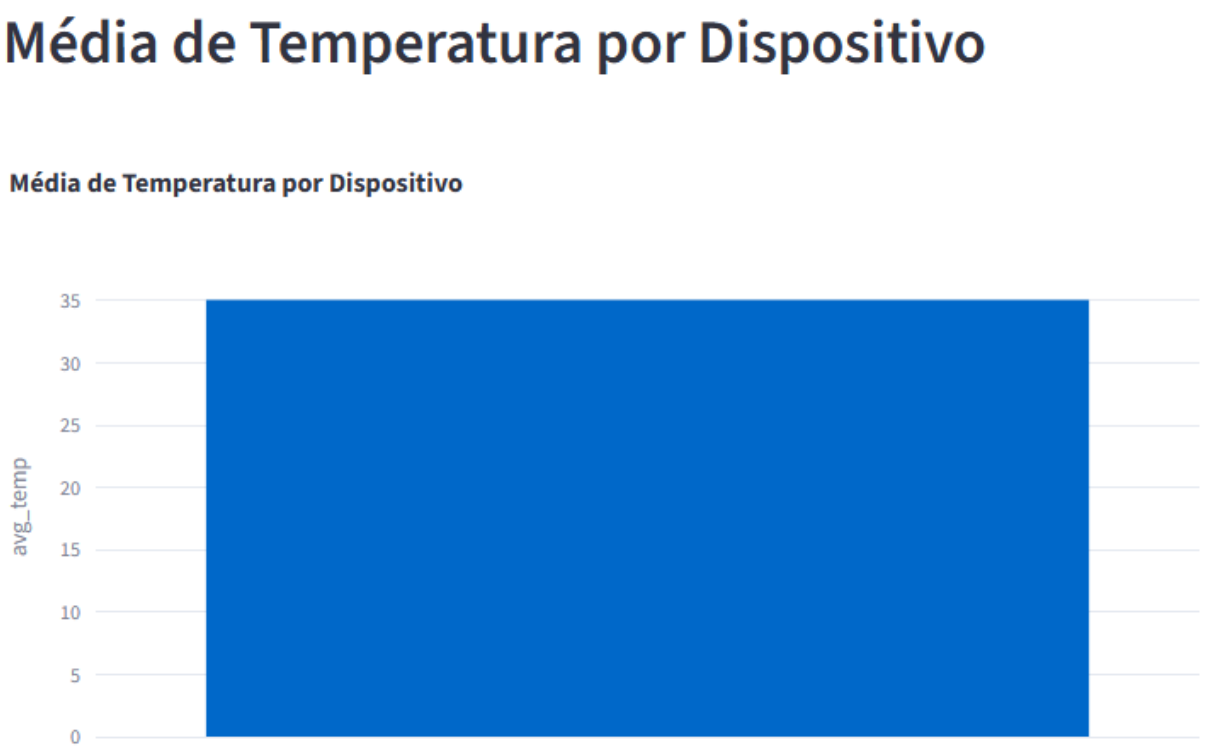
A etapa final do pipeline de dados é a transformação dos dados armazenados em insights visuais e acionáveis. Para isso, foi desenvolvido um dashboard interativo utilizando a biblioteca **Streamlit**, que serve como a interface principal para a análise. O dashboard foi projetado para ser intuitivo, permitindo que um usuário, mesmo sem conhecimento técnico do banco de dados, possa extrair informações valiosas. Cada visualização foi construída com a biblioteca **Plotly Express**, garantindo interatividade e clareza.

a) Gráfico de Barras: Média de Temperatura por Dispositivo

O primeiro componente do dashboard é um gráfico de barras que mapeia a temperatura média de operação de cada dispositivo, consultando diretamente a view avg_temp_por_dispositivo. As barras representam cada device_id único, e sua altura

corresponde à temperatura média calculada ao longo de todo o período de dados.

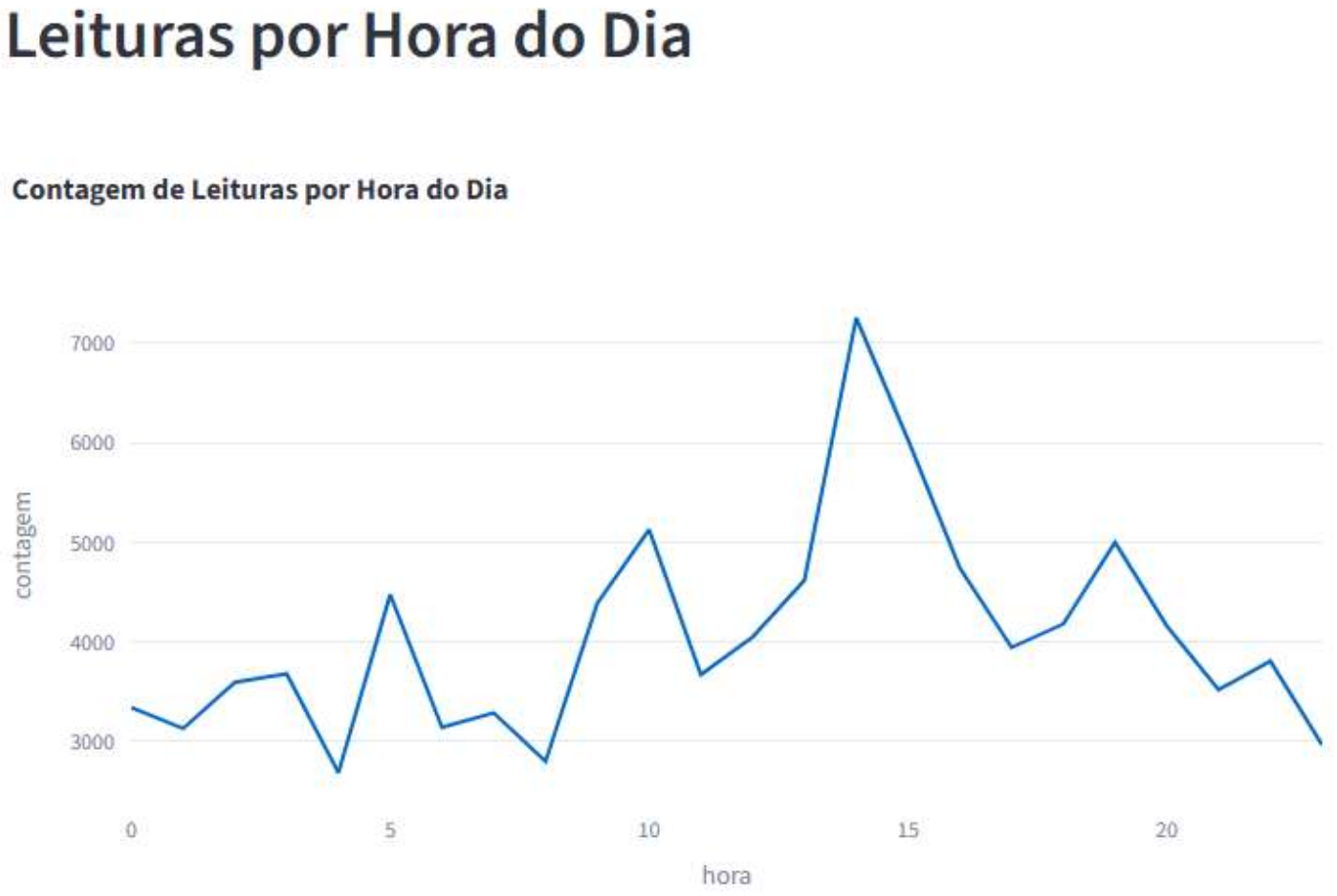
Esta visualização é fundamental para uma análise comparativa direta. Com um olhar rápido, é possível identificar outliers, como o dispositivo "Room Admin", que, neste conjunto de dados, opera com uma temperatura média visivelmente superior aos demais. Em um ambiente de produção, isso seria um alerta imediato para uma investigação, podendo indicar desde uma falha no sensor até um problema no ambiente em que ele está localizado.



b) Gráfico de Linha: Contagem de Leituras por Hora do Dia

A segunda visualização utiliza um gráfico de linha para ilustrar a distribuição da atividade de coleta de dados ao longo das 24 horas do dia. O eixo X representa as horas (de 0 a 23), e o eixo Y mostra o número total de leituras de temperatura registradas naquele horário, com base na view leituras_por_hora.

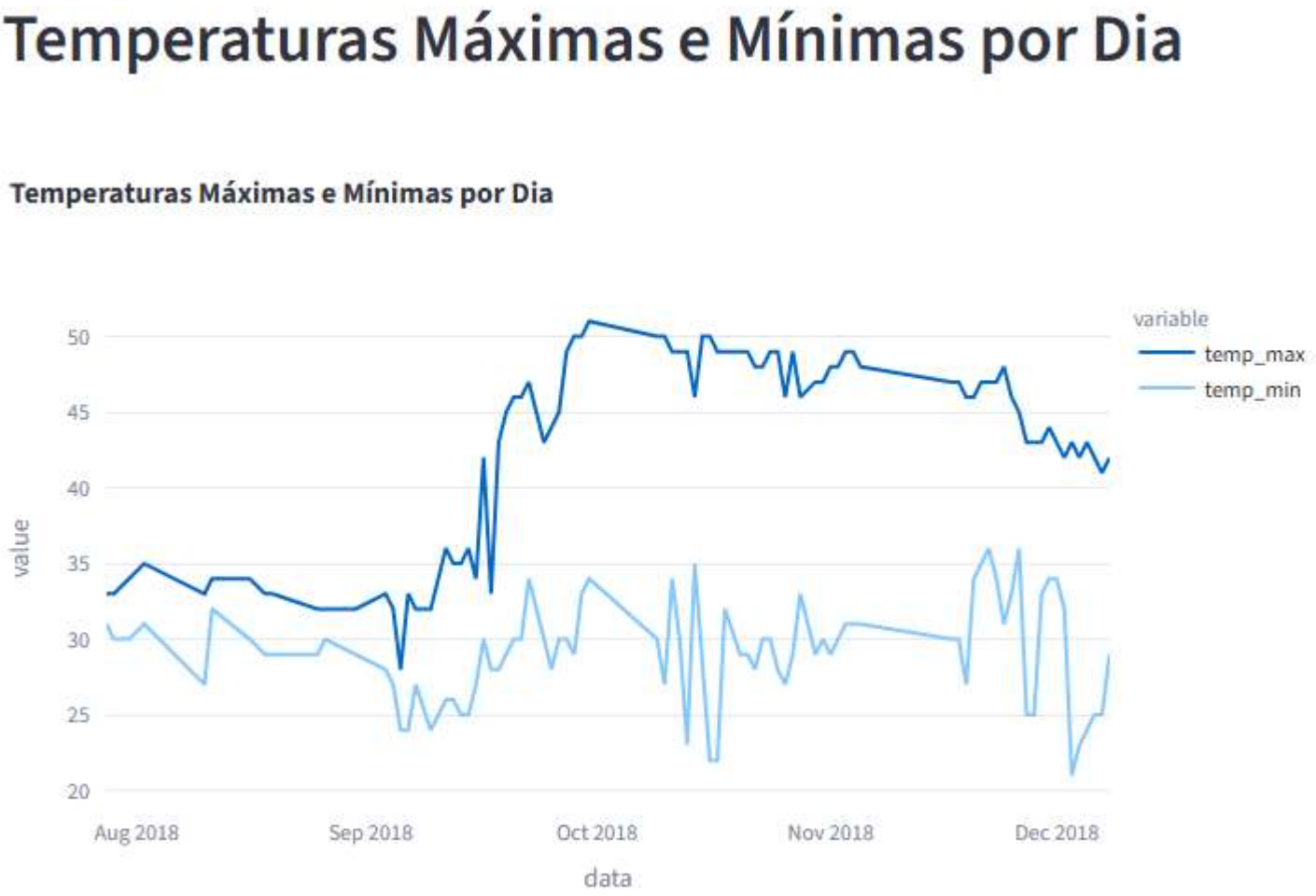
O gráfico revela o "pulso" do sistema de IoT. Picos em determinados horários podem indicar períodos de maior atividade ou testes no sistema, enquanto vales podem sinalizar momentos de baixa operação ou até mesmo falhas intermitentes na comunicação dos sensores. A análise deste padrão temporal é essencial para entender o comportamento da rede de dispositivos e para planejar manutenções em períodos de menor impacto.



c) Gráfico de Linha: Temperaturas Máximas e Mínimas por Dia

O terceiro e último gráfico oferece uma visão macro da variação térmica ao longo do tempo, utilizando os dados da view temp_max_min_por_dia. Ele plota duas linhas distintas: uma para a temperatura máxima (temp_max) e outra para a mínima (temp_min) de cada dia. O eixo X representa a linha do tempo, abrangendo todo o período dos dados coletados.

Esta visualização é crucial para analisar a amplitude térmica e a volatilidade. A distância entre as duas linhas indica a variação de temperatura em um único dia. Períodos onde a linha de temperatura máxima sobe abruptamente ou a mínima cai de forma acentuada são facilmente identificáveis, permitindo a correlação com eventos externos ou a detecção de anomalias operacionais. É a ferramenta principal para o monitoramento de tendências e para garantir que as condições ambientais permaneçam dentro dos limites operacionais seguros.



5. PRINCIPAIS INSIGHTS E CONCLUSÕES

O objetivo final de qualquer pipeline de dados não é apenas processar e armazenar informações, mas sim extrair conhecimento que possa guiar decisões estratégicas e otimizar operações. A análise dos gráficos gerados no dashboard permitiu a identificação de padrões claros e anomalias, que se traduzem em insights valiosos e abrem portas para aplicações práticas em um ambiente de produção.

a) Insights Obtidos a Partir dos Dados

A visualização interativa dos dados agregados revelou três conclusões principais:

- **Desempenho Anômalo de Dispositivos:** A análise do gráfico de temperatura média por dispositivo destacou imediatamente um outlier. Foi possível observar que o dispositivo **Room Admin** opera consistentemente em uma temperatura média significativamente mais alta que os demais. Este é um insight crítico, pois em um ambiente real, seria um forte indicativo de que este sensor específico merece uma investigação, seja pela sua localização (próximo a uma fonte de calor), por uma carga de trabalho diferente ou por uma potencial falha de hardware iminente.
- **Padrões de Atividade Noturnos:** O gráfico de contagem de leituras por hora revelou

um padrão de atividade inesperado. A grande maioria das coletas de dados está concentrada em um período noturno, com um pico claro de atividade entre as **00h e as 07h da manhã**. Após esse horário, o número de leituras cai drasticamente. Este padrão sugere que a coleta de dados pode não ser contínua, mas sim um processo em lote (batch) agendado para rodar durante a madrugada, possivelmente para minimizar o impacto na rede ou nos sistemas durante o horário comercial.

- **Anomalias Temporais e Mudança de Regime:** No gráfico de temperaturas diárias, foi identificada uma mudança de comportamento notável a partir do **início de Outubro de 2018**. Neste período, a temperatura máxima registrada salta para um novo patamar, consistentemente acima dos 40 graus, enquanto a temperatura mínima também sobe, embora de forma menos acentuada. Esta variação atípica, que se sustenta por meses, sugere um evento significativo, como uma mudança nas condições ambientais do local, uma alteração na configuração dos equipamentos monitorados ou o início de um período de operação mais intensivo.

b) Sugestões de Uso Prático em um Ambiente Real

Os insights obtidos demonstram o potencial do pipeline. Em um ambiente de produção, esta solução poderia ser expandida para agregar ainda mais valor ao negócio:

- **Manutenção Preditiva:** O monitoramento contínuo da temperatura média dos dispositivos, como visto com o Room Admin, é a base para um sistema de manutenção preditiva. Ao invés de apenas reagir a falhas, o sistema poderia ser aprimorado para analisar tendências ao longo do tempo. Um aumento gradual e contínuo na temperatura de um dispositivo específico, por exemplo, poderia disparar um alerta automático para a equipe de manutenção, sugerindo uma inspeção antes que uma falha catastrófica ocorra, economizando custos e tempo de inatividade.
- **Eficiência Energética e Otimização de Processos:** Em ambientes que necessitam de controle de temperatura (como data centers ou fábricas), os dados deste pipeline são uma mina de ouro para a otimização de custos. Ao cruzar as informações de variação térmica diária com os dados de consumo de energia dos sistemas de climatização (ar condicionado, ventilação), a empresa pode identificar ineficiências, ajustar termostatos de forma inteligente e otimizar o fluxo de ar, resultando em uma redução significativa nos custos com energia elétrica.
- **Alertas em Tempo Real e Segurança Operacional:** O pipeline atual opera em lote, mas poderia ser evoluído para um modelo de processamento em tempo real (streaming). Neste cenário, cada nova leitura de temperatura seria analisada instantaneamente. Regras de negócio poderiam ser criadas para definir limites operacionais seguros (por exemplo, temperatura máxima de 50°C). Se um dispositivo

registrar uma leitura que viole essa regra, o sistema poderia enviar alertas automáticos e imediatos (via e-mail, SMS ou notificações em aplicativos como o Slack) para a equipe de operações, permitindo uma resposta rápida para evitar danos a equipamentos sensíveis e garantir a segurança do ambiente.

6. CONCLUSÃO

O desenvolvimento deste projeto demonstrou, de forma prática e eficaz, a construção de um pipeline de dados de ponta a ponta, desde a ingestão de dados brutos de sensores de IoT até a sua transformação em insights visuais e acionáveis. O objetivo principal, que era criar uma arquitetura de dados robusta, escalável e replicável para o monitoramento de temperaturas, foi plenamente alcançado.

A utilização estratégica de tecnologias modernas foi um pilar fundamental para o sucesso. O **Docker** garantiu um ambiente de banco de dados isolado e consistente, eliminando complexidades de configuração e assegurando que a solução possa ser facilmente implantada em qualquer máquina. O **PostgreSQL**, com o uso inteligente de **Views SQL**, provou ser uma ferramenta poderosa não apenas para o armazenamento, mas também para o pré-processamento de dados, otimizando a performance e simplificando a lógica da aplicação final.

Por fim, as bibliotecas **Python**, como Pandas, SQLAlchemy, Streamlit e Plotly, integraram-se de maneira fluida para criar um fluxo de trabalho coeso, culminando em um dashboard interativo que traduz milhares de registros de dados em gráficos claros e informativos.

Este trabalho não se limitou a um exercício técnico; ele simulou um desafio real enfrentado por inúmeras indústrias na era da Internet das Coisas. A solução desenvolvida serve como um protótipo funcional e uma prova de conceito sólida, demonstrando que é possível, com as ferramentas certas, transformar o imenso volume de dados gerado por dispositivos conectados em conhecimento estratégico para a tomada de decisões, seja para manutenção preditiva, otimização de processos ou eficiência energética.

7. REFERÊNCIAS BIBLIOGRÁFICAS

Conjunto de Dados

JHA, Atul Anand. **Temperature Readings: IoT Devices**. Kaggle, 2021. Disponível em: https://www.kaggle.com/datasets/atulanandjha/temperature-readings-iot-devices. Acesso em: 01 set. 2025.

Documentação das Tecnologias

DOCKER INC. **Docker Documentation**. Disponível em: https://docs.docker.com/. Acesso em: 01 set. 2025.

PLOTLY TECHNOLOGIES INC. **Plotly Python Graphing Library**. Disponível em: https://plotly.com/python/. Acesso em: 01 set. 2025.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL Documentation**. Disponível em: https://www.postgresql.org/docs/. Acesso em: 01 set. 2025.

PYTHON SOFTWARE FOUNDATION. **Python 3 Documentation**. Disponível em: https://docs.python.org/3/. Acesso em: 01 set. 2025.

SQLALCHEMY AUTHORS AND CONTRIBUTORS. **SQLAlchemy Documentation**. Disponível em: https://www.sqlalchemy.org/docs/. Acesso em: 01 set. 2025.

STREAMLIT INC. **Streamlit Documentation**. Disponível em: https://docs.streamlit.io/. Acesso em: 01 set. 2025.

THE PANDAS DEVELOPMENT TEAM. **pandas Documentation**. Disponível em: https://pandas.pydata.org/docs/. Acesso em: 01 set. 2025.