UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO MANOEL HERMINIO ALVES

SOLUÇÃO DE IOT ANALYTICS PARA SENSORES AMBIENTAIS

CAMPINA GRANDE - PB

2024

PEDRO MANOEL HERMINIO ALVES

SOLUÇÃO DE IOT ANALYTICS PARA SENSORES AMBIENTAIS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Cláudio de Souza Baptista

CAMPINA GRANDE - PB

2024

PEDRO MANOEL HERMINIO ALVES

SOLUÇÃO DE IOT ANALYTICS PARA SENSORES AMBIENTAIS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Cláudio de Souza Baptista

Orientador – UASC/CEEI/UFCG

Reinaldo Cézar de Morais Gomes

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 15 de Maio de 2024.

CAMPINA GRANDE - PB

RESUMO

Em um mundo onde a preocupação com o meio ambiente e a saúde pública está em ascensão, a necessidade de sistemas de monitoramento ambiental é cada vez mais premente. Os sensores ambientais são fundamentais neste cenário, coletando uma variedade de métricas que espelham as condições do ambiente, incluindo qualidade do ar, temperatura, umidade e pressão atmosférica. A solução proposta neste estudo tem como objetivo coletar, analisar e visualizar dados em tempo real e históricos, proporcionando uma compreensão mais completa das condições ambientais e permitindo a implementação de medidas preventivas. Utilizando ferramentas de código aberto e uma abordagem baseada em containers, juntamente com os princípios de uma arquitetura em microsserviços, a solução busca aprimorar a eficiência e a precisão do monitoramento ambiental. Esta solução pode servir como um modelo replicável para desafios semelhantes em outras áreas ou contextos que requerem processamento e análises de dados em tempo real. Este estudo apresenta uma solução de IoT Analytics para sensores ambientais, focada na monitorização de métricas ambientais. A eficácia desta solução foi comprovada no estado do Acre, através de um estudo comparativo que a colocou à prova contra uma solução já existente.

IOT ANALYTICS SOLUTION FOR ENVIRONMENTAL SENSORS

ABSTRACT

In a world where concern for the environment and public health is on the rise, the need for environmental monitoring systems is increasingly pressing. Environmental sensors play a crucial role in this scenario, collecting a range of metrics that mirror the conditions of the surrounding environment, including air quality, temperature, humidity, and atmospheric pressure. The proposed solution in this study aims to collect, analyze, and visualize real-time and historical data, providing a more comprehensive understanding of environmental conditions and enabling the implementation of preventive measures. Using open-source tools and a container-based approach, along with the principles of a microservices architecture, the solution seeks to enhance the efficiency and accuracy of environmental monitoring. This solution can serve as a replicable model for similar challenges in other areas or contexts that require real-time data processing and analysis. This study introduces an IoT Analytics solution for environmental sensors, focusing on the monitoring of environmental metrics. The efficacy of this solution was proven in the state of Acre, through a comparative study that tested it against an existing solution

Solução de *IoT Analytics* para sensores ambientais

Pedro Manoel Herminio Alves
Unidade Acadêmica de Sistemas e Computação - UASC
Universidade Federal de Campina Grande - UFCG
Campina Grande, Paraíba, Brasil
pedro.alves@ccc.ufcg.edu.br

Cláudio de Souza Baptista
Unidade Acadêmica de Sistemas e Computação - UASC
Universidade Federal de Campina Grande - UFCG
Campina Grande, Paraíba, Brasil
baptista@computacao.ufcg.edu.br

RESUMO

Em um mundo onde a preocupação com o meio ambiente e a saúde pública está em ascensão, a necessidade de sistemas de monitoramento ambiental é cada vez mais premente. Os sensores ambientais são fundamentais neste cenário, coletando uma variedade de métricas que espelham as condições do ambiente, incluindo qualidade do ar, temperatura, umidade e pressão atmosférica. A solução proposta neste estudo tem como objetivo coletar, analisar e visualizar dados em tempo real e históricos, proporcionando uma compreensão mais completa das condições ambientais e permitindo a implementação de medidas preventivas. Utilizando ferramentas de código aberto e uma abordagem baseada em containers, juntamente com os princípios de uma arquitetura em microsserviços, a solução busca aprimorar a eficiência e a precisão do monitoramento ambiental. Esta solução pode servir como um modelo replicável para desafios semelhantes em outras áreas ou contextos que requerem processamento e análises de dados em tempo real. Este estudo apresenta uma solução de IoT Analytics para sensores ambientais, focada na monitorização de métricas ambientais. A eficácia desta solução foi comprovada no estado do Acre, através de um estudo comparativo que a colocou à prova contra uma solução já existente.

Palavras-chave: *IoT Analytics*, Sensores ambientais, Open-source, Monitoramento em tempo real, BI

Repositório:
https://github.com/pedro-manoel/iot-analytics-solution-tcc

1. INTRODUÇÃO

No mundo contemporâneo, a crescente preocupação com o ambiente e a saúde pública tem impulsionado o desenvolvimento e a implementação de sistemas de monitoramento ambiental cada vez mais sofisticados. Sensores ambientais desempenham um papel crucial nesse contexto, coletando uma ampla gama de métricas que refletem a qualidade e as condições do ambiente circundante. Essas métricas incluem, entre outras, a qualidade do ar, temperatura, umidade e pressão atmosférica. Dessa forma, é de suma importância compreender tais métricas visando proteger nossa saúde, o meio ambiente e tomar medidas preventivas.

A qualidade do ar, por exemplo, é um fator crítico para a saúde humana. Estudos mostram que a poluição atmosférica está associada a uma série de doenças, incluindo doenças respiratórias, cardiovasculares e até mesmo câncer [1]. A exposição prolongada a poluentes como partículas finas (PM2.5) e ozônio pode ter efeitos adversos na saúde, aumentando o risco de morte prematura e agravando condições crônicas.

Nesse contexto, a Internet das Coisas (IoT) emerge como uma ferramenta indispensável para monitorar métricas ambientais em tempo real. Isso é possível através da utilização de sensores ambientais, que foram projetados exclusivamente para esse propósito. A IoT refere-se à interconexão de dispositivos físicos através da internet. Esses dispositivos, que podem variar desde eletrodomésticos até sensores e veículos autônomos, os quais coletam e trocam dados entre si e com sistemas de computação em nuvem e outras aplicações [2].

Contudo, é evidente que, em muitos casos, os dados coletados por esses dispositivos IoT não são plenamente aproveitados devido à carência de uma solução adequada para coleta e visualização dessas informações, o que representa uma significativa lacuna no processo de tomada de decisão, sobretudo em tempo real.

No estado do Acre, o projeto *"Qualidade do Ar: Efeito de Desmatamento, Incêndios e Queimadas"* [8] desenvolvido por docentes e pesquisadores da Universidade Federal do Acre (UFAC), tem estabelecido um sistema de monitoramento da qualidade do ar, através da utilização de uma rede de sensores de baixo custo, e empreendido esforços para garantir um ambiente livre da poluição atmosférica, antecipando soluções para os problemas identificados.

Apesar desta iniciativa, o sistema disponível para coletar e analisar os dados provenientes da rede de sensores, que pode ser visualizado na Figura 1, não se alinha com as melhores práticas adotadas atualmente. O sistema não opera em tempo real em todas as suas funcionalidades, o que é essencial no contexto da IoT. Além disso, não se utiliza um banco de dados analítico no armazenamento dos dados, e a visualização não se beneficia de uma ferramenta que permita análises detalhadas e a visualização de gráficos e tabelas dinâmicas, como as ferramentas de *Business Intelligence* (BI).

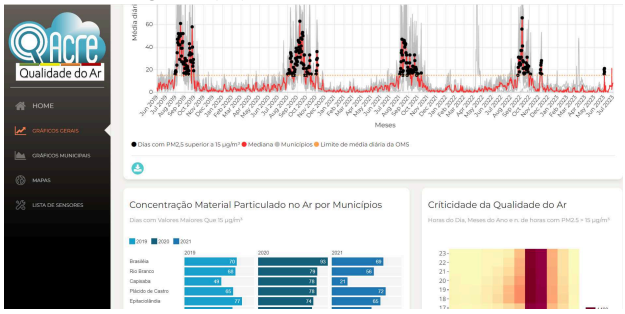

Figura 1: Site de qualidade do ar do Acre.

Em resposta ao problema proposto, o principal objetivo deste estudo é conceber uma solução de *IoT Analytics*, especificamente adaptada à rede de sensores de qualidade do ar do Acre. Contudo, sua aplicação não se restringe apenas a este contexto. A solução proposta tem o potencial de servir como um modelo replicável para quaisquer desafios semelhantes que envolvam sensores ambientais. Esta solução será desenvolvida com o uso de ferramentas de código aberto e projetada para funcionar em tempo real.

2. REFERENCIAL TEÓRICO

Nesta seção, serão abordadas as informações e conceitos-chave para uma melhor compreensão deste trabalho. Tal fundamentação visa estabelecer uma base sólida para o estudo em tela, o contexto desta pesquisa.

2.1. Métricas de condições ambientais

As métricas de condições ambientais são medidas quantitativas usadas para avaliar e monitorar a qualidade do ambiente em que vivemos. Essas métricas são essenciais para entender a saúde do nosso planeta e podem nos ajudar a tomar decisões informadas sobre como proteger e preservar nosso ambiente.

2.1.1. Qualidade do Ar

A qualidade do ar é uma métrica que avalia a presença de poluentes no ar que respiramos. Isso inclui partículas suspensas, gases como dióxido de enxofre, monóxido de carbono, óxidos de nitrogênio e compostos orgânicos voláteis. A qualidade do ar é crucial para a saúde humana e do ecossistema, e a má qualidade do ar pode levar a uma série de problemas de saúde e danos ambientais.

2.1.2. Umidade Relativa

A umidade relativa é uma medida que mede a quantidade de vapor de água presente no ar em relação à quantidade máxima que o ar poderia conter na mesma temperatura. A umidade relativa afeta o conforto humano, a saúde e a eficiência de muitos processos industriais e agrícolas. Além disso, pode influenciar o crescimento e a distribuição de plantas e animais.

2.1.3. Temperatura

A temperatura é uma métrica fundamental que descreve o estado térmico de um ambiente. Ela influencia uma ampla gama de processos naturais e humanos, desde o comportamento dos ecossistemas até a eficiência energética dos edifícios. As mudanças na temperatura média global são uma preocupação particular devido ao seu papel nas mudanças climáticas.

2.1.4. Pressão Atmosférica

A pressão atmosférica é a força exercida pelo peso do ar acima de um ponto na superfície da Terra. Ela é uma métrica importante na meteorologia, pois influencia o clima e o tempo. A pressão atmosférica também afeta a respiração humana e o desempenho de certas tecnologias, como a aviação e a geração de energia eólica.

2.2. Sensores ambientais

Os sensores ambientais são dispositivos que medem variáveis como temperatura, umidade e qualidade do ar, desempenhando um papel crucial no monitoramento do meio ambiente. Eles coletam dados em tempo real, permitindo uma análise atualizada das condições ambientais. São usados em diversas aplicações, desde o monitoramento da qualidade do ar em áreas urbanas até a detecção de mudanças climáticas em escala global.

2.3. Rede de Sensores do Acre

Atualmente, o Acre conta com uma rede de 30 sensores de qualidade do ar, distribuídos entre seus 22 municípios [16]. Esses sensores, de baixa qualidade, são produtos da empresa PurpleAir[1]. Essa empresa é conhecida por fabricar sensores que coletam e compartilham dados hiperlocais sobre a qualidade do ar com o público [17]. Os seus sensores medem a poluição por partículas (PM2.5), são fáceis de instalar e conectam-se facilmente às redes WI-FI. Segundo a empresa, o custo desses sensores é uma fração do valor de um sensor comercial.

Além de coletar métricas de qualidade do ar, os sensores também coletam uma variedade de outros dados, como temperatura, umidade, pressão atmosférica e até mesmo informações sobre o estado do sensor, como a intensidade do sinal WI-FI, o índice de confiabilidade da medida do PM2.5 e a data da sua última leitura registrada. Além disso, a PurpleAir disponibiliza uma funcionalidade para os proprietários de seus sensores. Após o registro do dispositivo no sistema da empresa, é possível acessar os dados gerados por esses sensores através de uma API RESTful.[17].

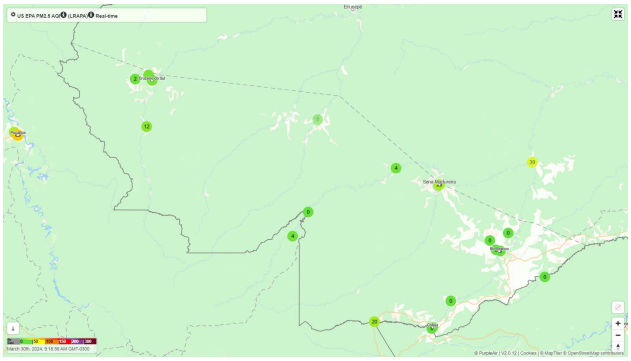

Figura 2: Rede de sensores de qualidade do ar do Acre.

2.4. Internet das Coisas (IoT)

A Internet das Coisas (IoT) é uma rede de objetos físicos conectados à internet que trocam dados entre si e com outros dispositivos ou sistemas computacionais. Esses dispositivos variam desde objetos domésticos comuns até ferramentas industriais sofisticadas. A IoT tem se tornado uma das tecnologias mais importantes do século XXI, permitindo uma comunicação perfeita entre pessoas, processos e coisas. Por meio da computação de baixo custo, nuvem, big data, análise avançada e tecnologias móveis, coisas físicas podem compartilhar e coletar dados com o mínimo de intervenção humana.

2.5. *IoT Analytics*

IoT Analytics pode ser entendido como uma plataforma projetada para a ingestão contínua e em tempo real de dados gerados por uma variedade de dispositivos IoT [18]. Esta

[1] https://www2.purpleair.com

plataforma multifuncional é responsável pela ingestão de dados, processamento, armazenamento e, finalmente, pela visualização desses dados. A sua principal função é transformar um fluxo constante de dados brutos em informações significativas e acionáveis, permitindo assim uma tomada de decisão mais eficaz.

Empresas como a Google e Amazon oferecem soluções tecnológicas para lidar com dados de IoT em tempo real. O *Google Cloud's Streaming Analytics* e o *AWS IoT Analytics* facilitam a ingestão, processamento e análise de fluxos de eventos em tempo real, permitindo a análise de centenas de milhões de eventos por segundo de aplicativos ou dispositivos praticamente em qualquer lugar do globo [24, 25].

3. TRABALHOS RELACIONADOS

No trabalho de Myrian et al. [9] é abordado o uso de ferramentas de IoT com o objetivo de monitorar e reduzir o consumo de energia em uma fábrica de escovas de dente e creme dental, numa perspectiva de sustentabilidade. A solução proposta pelos autores envolveu o desenvolvimento de um sistema de monitoramento do consumo de ar comprimido dos equipamentos, utilizando um sensor de vazão calorimétrico inteligente em conjunto com softwares de gerenciamento.

No estudo de Pedro Ivo et al. [10], é proposta uma análise do consumo de combustível após a implantação de automação para controle de abastecimento, utilizando ferramentas de *Data Analytics* e IoT. Este trabalho analisa o impacto da automação e das tecnologias da Indústria 4.0 na gestão do abastecimento de combustível em uma obra de construção de parque eólico. Através da implantação de sistemas de controle e monitoramento baseados em IoT e Data Analytics, o autor busca identificar oportunidades de redução de custos e aumento da eficiência nesse processo.

No trabalho de Sin et al. [13], é proposta uma nova arquitetura IoT que utiliza a abordagem de design baseada em componentes. Os autores definiram componentes de serviço independentes, porém interoperáveis, para sistemas IoT. Além disso, uma função de feedback baseada em dados é incluída como um recurso chave da arquitetura proposta, permitindo um maior grau de automação do sistema e reduzindo a dependência humana para análise de dados e tomada de decisão.

No estudo de Saverio et al. [14], é apresentada uma arquitetura de IoT para monitoramento pervasivo da qualidade do ar e *exposome*. A arquitetura proposta utiliza a abordagem de *crowdsensing* e é projetada para avaliar a qualidade do ar em um cenário urbano ou móvel. Isso é fundamental para ações de redução de exposição pessoal ou em toda a cidade.

No estudo de Alain et al. [15], é abordado o problema da má qualidade do ar, que afeta a saúde humana e os ecossistemas em todo o mundo. Os autores propuseram um sistema de monitoramento de qualidade do ar em que o custo e a confiabilidade foram otimizados. Dispositivos foram acoplados a uma plataforma IoT, permitindo que usuários inexperientes visualizem e compartilhem as informações coletadas.

4. ARQUITETURA

A arquitetura desenvolvida para a solução de *IoT Analytics* foi projetada seguindo os princípios de uma arquitetura de microsserviços. Trata-se de uma arquitetura composta por pequenos serviços independentes que se comunicam entre si. A arquitetura baseada em microsserviços torna mais fácil a criação de sistemas altamente adaptáveis e com grande capacidade de expansão, superando os métodos complexos de processamento centralizado em termos de escalabilidade, facilidade de manutenção e extensibilidade [12].

Além de ser fundamentada na arquitetura de microsserviços, a arquitetura proposta foi construída para suportar características-chave no contexto de dados da IoT, como interoperabilidade, reutilização e flexibilidade [13]. A interoperabilidade refere-se a capacidade da arquitetura de suportar conexões com diferentes dispositivos, sistemas e protocolos. A reutilização é a capacidade da arquitetura de operar em diversos segmentos além do que foi originalmente projetada. E a flexibilidade é a sua capacidade de se adaptar tanto ao ambiente em que vai operar quanto ao problema que vai resolver de forma fácil e intuitiva. O resultado final da arquitetura pode ser visualizado na Figura 3.

4.1. Componentes da Arquitetura

A arquitetura proposta é formada de um conjunto de componentes. Cada componente utilizado na arquitetura foi selecionado para desempenhar um serviço primário essencial. Todos pertencem à Fundação Apache[2] (*Apache Software Foundation - ASF*), uma organização sem fins lucrativos estabelecida para apoiar os projetos de código aberto, principalmente os da Apache. Dessa forma, foram utilizados exclusivamente componentes de código aberto, proporcionando assim uma arquitetura com personalização e flexibilidade.

Dado que o código-fonte das aplicações é totalmente acessível, tem-se a capacidade de ajustar o software para atender a necessidades específicas. Isso pode englobar a integração com outros sistemas, a adição de funcionalidades personalizadas, ou a correção de bugs ou problemas de segurança. Além disso, o uso de software de código aberto pode resultar em custos mais baixos. Embora possam existir custos associados ao suporte, treinamento ou implementação, não há custos de licenciamento de software.

4.1.1. Apache NiFi

O Apache NiFi é o componente escolhido para gerir a ingestão de dados na arquitetura, responsável por solicitar dados de fontes externas, processá-los e enviá-los ao destino adequado. O NiFi é um sistema que administra fluxos de dados, oferecendo uma interface visual na web para projetar, visualizar e controlar esses fluxos [19]. Ele também permite a automação de transferências de dados entre sistemas e lida eficientemente com diferentes formatos de dados e protocolos, garantindo a interoperabilidade da arquitetura.

[2] https://www.apache.org

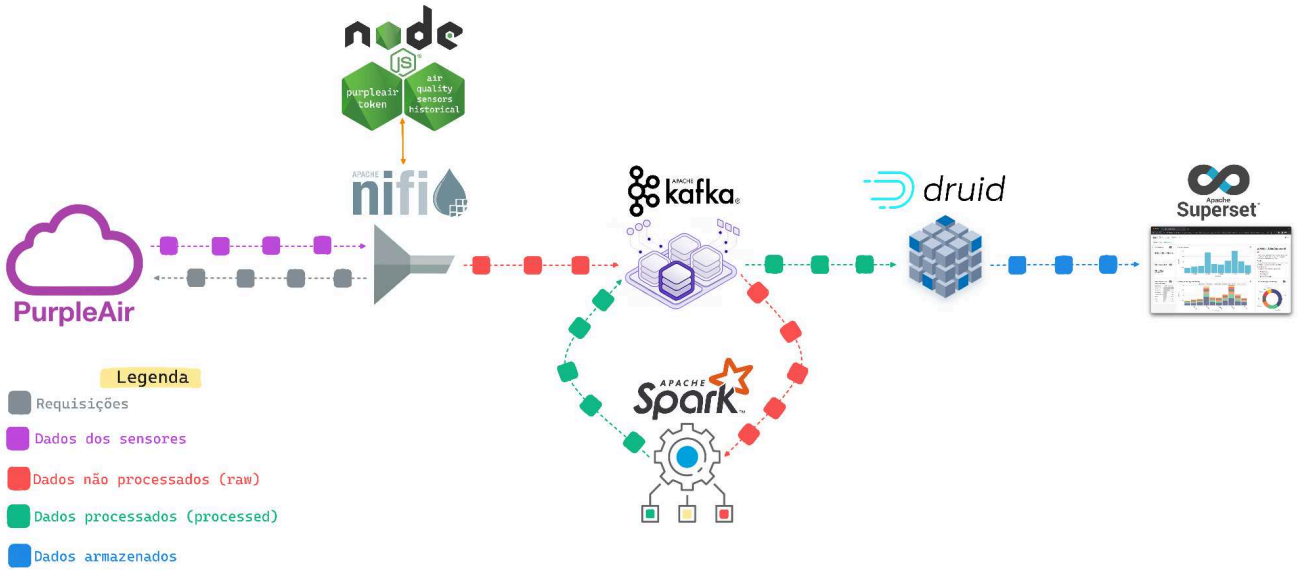

Figura 3: Arquitetura da Solução de *IoT Analytics*.

O NiFi opera através de componentes como o *processor*, o *flowfile* e a *connection*. O *processor* é um elemento que realiza uma tarefa simples, como uma operação de *split* em um JSON, e o NiFi possui vários deles, que podem ser configurados conforme necessário. O *flowfile* é uma abstração do dado no ambiente do NiFi, podendo ser gerado a partir de uma fonte de dados ou transformado em outro formato. A *connection* é a ligação entre dois *processors*, gerando uma fila (*Queue*) onde é possível ver a quantidade de *flowfiles* enfileirados e o tamanho deles em bytes.

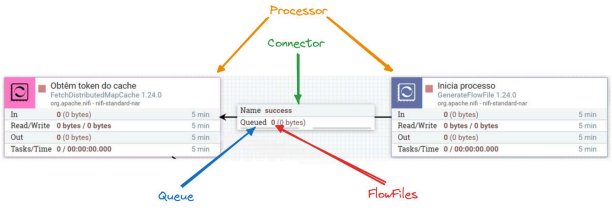

Figura 4: Componentes de um fluxo de dados no Apache NiFi.

4.1.2. *Apache Kafka*

O Apache Kafka é a ferramenta escolhida para gerenciar os fluxos de dados na arquitetura. Ele é uma plataforma de streaming de eventos distribuída, otimizada para ingestão e processamento de dados de em tempo real [20].

Para entender o Kafka, faz-se necessário entender os conceitos de tópicos e partições. Um tópico é um *feed* nomeado para o qual os registros são publicados e pode ter vários consumidores. Para permitir o consumo paralelo de dados, um tópico é dividido em várias partições, cada uma sendo uma sequência ordenada e imutável de registros anexados a um *commit log*.

O funcionamento do Kafka é baseado na interação com produtores e consumidores. Os produtores enviam dados para tópicos específicos no Kafka, enquanto os consumidores inscrevem-se em um ou mais tópicos para consumir esses dados enviados.

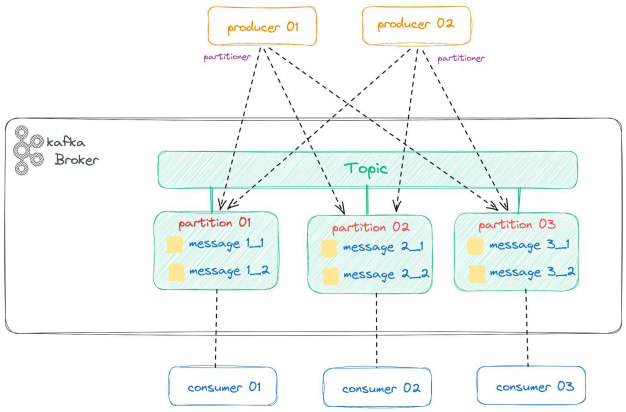

Figura 5: Funcionamento do Apache Kafka.

4.1.3. *Apache Spark*

O Apache Spark foi a ferramenta escolhida para realizar o processamento de dados na arquitetura proposta, utilizando especificamente o *Spark Structured Streaming*. Este serviço recebe dados brutos (*raw*), realiza transformações, resultando em dados processados (*processed*) que são encaminhados para o local apropriado.

O Spark é um mecanismo usado para tarefas de engenharia de dados, ciência de dados e aprendizado de máquina, que pode ser executado em máquinas individuais ou clusters de nós [21]. O *Spark Structured Streaming*, uma extensão do Spark, permite o processamento de dados em tempo real.

No contexto do *Spark Structured Streaming*, os dados são manipulados como *DataFrames* ou *Datasets*. Estas são abstrações de alto nível que permitem que você pense nos seus dados como se estivessem em uma tabela, mesmo quando estão sendo processados em tempo real. Internamente, as consultas do *Spark Structured Streaming* são executadas, por padrão, através de um mecanismo de processamento de *micro-batch*. Este mecanismo trata os fluxos de dados como uma sequência de pequenos lotes de trabalho, otimizando assim o processamento.

O funcionamento do *Spark Structured Streaming* é baseado na execução de operações de transformação e ação nos *DataFrames* e *Datasets*. As transformações criam um novo conjunto de dados a partir de um existente, enquanto as operações de ação realizam cálculos e produzem um resultado retornado para o programa principal.

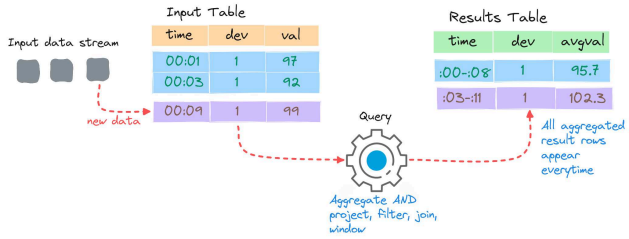

Figura 6: Funcionamento do *Spark Structured Streaming*.

4.1.4. Apache Druid

O Apache Druid foi a ferramenta selecionada para armazenar e analisar os dados processados em tempo real na arquitetura. Este serviço é responsável por armazenar os dados processados e disponibilizá-los para serem utilizados por outras ferramentas e serviços, permitindo a realização de análises em tempo real.

O Apache Druid é um banco de dados colunar de análise em tempo real de alto desempenho, projetado especificamente para consultas rápidas de análise (conhecidas como "consultas OLAP") em grandes conjuntos de dados. Ele oferece consultas eficientes em dados de streaming e em lote, mantendo a performance mesmo sob alta carga. O Druid armazena seus dados em arquivos chamados segmentos. Cada segmento é um único arquivo, normalmente composto por até alguns milhões de linhas de dados [22].

O Apache Druid processa dados em lote e em fluxo contínuo através dos nós de dados, compostos pelos componentes "Indexer" (processamento e armazenamento de dados) e "Historical" (recuperação rápida de dados). As consultas são direcionadas ao nó de consulta, que inclui um "Broker" e um console web. Os dados são armazenados de forma segura no "Deep storage". O nó mestre, com o "Coordinator" e o Apache Zookeeper[3], gerenciam o cluster. Essa arquitetura torna o Druid flexível, escalável e capaz de lidar com grandes volumes de dados em tempo real.

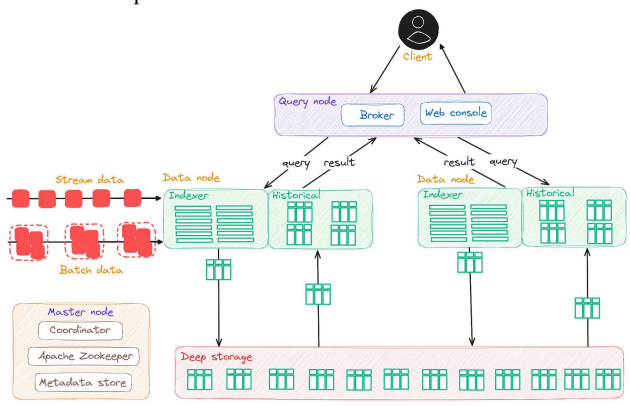

Figura 7: Funcionamento do Apache Druid.

4.1.5. Apache Superset

O Apache Superset foi a ferramenta selecionada para visualizar e analisar os dados na arquitetura. Este componente é responsável por apresentar de forma analítica os dados para os usuários através da utilização de painéis e gráficos.

O Apache Superset é uma plataforma moderna de exploração e visualização de dados bem como uma ótima ferramenta de *business intelligence* (BI) [23]. Foi desenvolvido para ser leve, rápido e intuitivo e está repleto de opções que tornam mais fácil para os usuários de todos os níveis de habilidade explorar e visualizar seus dados, desde gráficos de linhas simples até gráficos geoespaciais altamente detalhados [23].

O Apache Superset trabalha de forma hierárquica, onde primeiro adicionamos o *Database*, depois as tabelas ou *datasets*. Ele permite explorar de forma complexa cada visualização ao permitir detalhar cada gráfico/métrica e facilmente filtrar e agrupar os dados, permitindo a construção de dashboards com elevado valor para o utilizador.

4.2. Serviços desenvolvidos

Para auxiliar o NiFi, a ferramenta de ingestão de dados da arquitetura, foi necessário desenvolver alguns serviços. Estes serviços foram todos desenvolvidos utilizando o Node.js[4], um ambiente de execução JavaScript. O Node.js permite que aplicações desenvolvidas nesta linguagem sejam executadas de forma autônoma, sem a necessidade de um navegador. Com o Node.js, é possível criar uma ampla variedade de aplicações web, desde servidores para sites estáticos e dinâmicos, até APIs e sistemas baseados em microserviços.

4.2.1. O Serviço air-quality-sensors-historical

Esse serviço de dados históricos foi desenvolvido com o objetivo de facilitar a ingestão controlada de dados a partir de dados salvos em um arquivo CSV. Ele disponibiliza uma API que é capaz de retornar um número pré-definido de linhas de dados por requisição. Por exemplo, se a configuração estipula o retorno de 700 linhas por requisição, a API irá, na primeira requisição, retornar as 700 primeiras linhas do arquivo. Na segunda requisição, ela retornará as 700 linhas subsequentes, iniciando a partir da última linha lida na requisição anterior. Este processo se repete até que todas as linhas do arquivo tenham sido lidas.

As configurações desse serviço são determinadas pelas opções existentes em suas variáveis de ambiente, localizadas no arquivo ".env" que pode ser visualizado na figura abaixo.

```
1 # App
2 APP_PORT=3001
3 APP_NUMBER_READING_LINES_CSV=10000
4 APP_NUMBER_SPACES_BEFORE_LINE_NUMBERS=7
5
6 # Data
7 HISTORICAL_CSV_FILE_NAME='historical_data.csv'
```

Figura 8: Arquivo .env do serviço de dados históricos.

Dentre as opções disponíveis, as de importância considerável são a *APP_NUMBER_READING_LINES_CSV* (define quantas

[3] https://zookeeper.apache.org

[4] https://nodejs.org/en

linhas devem ser retornadas a cada requisição), e *HISTORICAL_CSV_FILE_NAME* (especifica o nome do arquivo CSV que deve ser utilizado).

4.2.2. O serviço purpleair-token

Este serviço desempenha um papel crucial no fluxo principal da arquitetura. Sua principal responsabilidade é extrair e fornecer um token válido que pode ser empregado nas solicitações para a API da PurpleAir. Para acessar os dados da rede de sensores, é imprescindível enviar uma solicitação à API da PurpleAir. Contudo, essa API requer uma chave de acesso ou um token para liberar seus dados.

A utilização da chave de acesso representa um desafio: ela é adquirida através de um cadastro e cada usuário possui um limite de uso baseado em créditos. Cada solicitação consome créditos, cujo valor é estabelecido pela PurpleAir, levando em consideração fatores como o tamanho do conjunto de dados e o período solicitado [17].

Ao analisar a página do mapa da PurpleAir (Figura 2), notou-se que é empregado um token para acessar os dados apresentados. Assim, a finalidade deste serviço é utilizar o mesmo mecanismo empregado por essa página para obter esse token. Dessa forma, a arquitetura proposta poderá acessar os dados de maneira idêntica à página.

Para atingir esse objetivo, o serviço utiliza a técnica de web *scraping*[5], que extrai dados de sites de forma automatizada. Ele opera de maneira similar a um usuário humano que acessa e extrai informações de um site, porém de forma automatizada. Para aprimorar as funcionalidades do serviço e estabelecer sua operação de maneira eficiente, foram utilizadas variáveis de ambiente. Essas variáveis, presentes no arquivo ".env", foram utilizadas para configurar este serviço, assim como os demais serviços desenvolvidos na arquitetura.

O arquivo ".env" contém opções como *APP_MOCK* (determina se a API do serviço deve retorna um token falso), *SCRAPER_URL* (URL para o web *scraping*) e *SCRAPER_TIMEOUT* (tempo limite do processo de web *scraping*). Existe também a variável *TOKEN_MUST_REUSE*. Quando esta variável está definida como TRUE, o serviço opta por reutilizar tokens válidos, em vez de gerar novos a cada solicitação. No entanto, se um token for considerado inválido, um novo token é gerado automaticamente.

4.3. Fluxos

O funcionamento da arquitetura é dividido em quatro fluxos: ingestão, processamento, armazenamento e visualização. Cada um com uma funcionalidade bem definida. Aqui definimos fluxo como a interação entre duas ou mais aplicações que funcionam em conjunto com o intuito de atingir um certo objetivo. Esses fluxos podem ser observados na Figura 9.

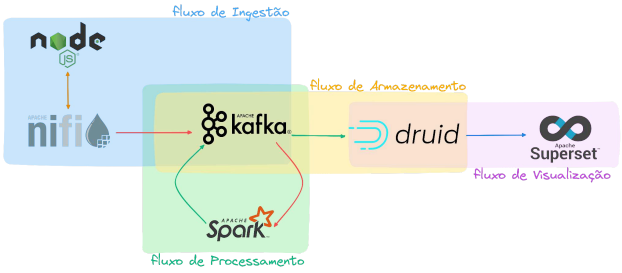

Figura 9: Fluxos da arquitetura.

4.3.1. Ingestão

Esse é o primeiro fluxo da arquitetura que inicia todo o processo. Este fluxo tem como objetivo realizar a ingestão periódica e automatizada dos dados de sensores. Este fluxo é composto por dois componentes: o Apache NiFi e o Apache Kafka.

No Apache NiFi, foram criados dois *process group*. O primeiro é responsável pela ingestão dos dados dos sensores em tempo real e o segundo pela ingestão dos dados históricos, utilizando o serviço de dados históricos (*air-quality-sensors-historical*). Esses *process groups* podem ser visualizados na Figura 10.

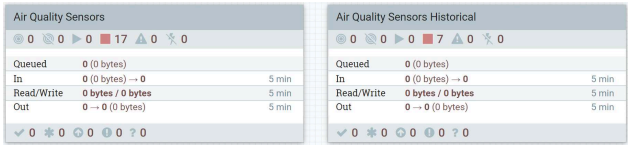

Figura 10: *Process groups* da solução no Apache NiFi.

No *process group* de ingestão em tempo real, "Air Quality Sensors", que pode ser visualizado na Figura 11, dois processos são realizados em paralelo dentro do módulo "Obtém dados dos sensores". O primeiro faz as requisições que extraem o token da PurpleAir, através do serviço (*purpleair-token*), e salva no cache do NiFi, isso é feito a cada seis horas, período conhecido em que o token ainda não expirou. O segundo faz as requisições para obter os dados dos sensores utilizando esse token salvo no cache.

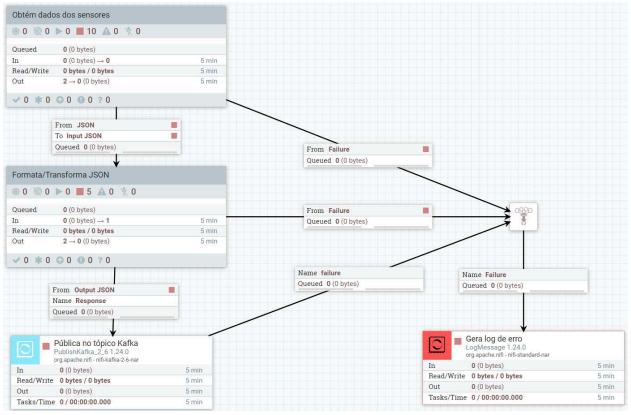

Figura 11: *Process group* de ingestão em tempo real no Apache NiFi.

As requisições para obter os dados dos sensores são feitas a cada dois minutos, que é também o intervalo de tempo em que os sensores geram novos dados. O resultado dessa requisição é mostrado na Figura 12. Este resultado é passado para um processo que formata e transforma o dado JSON para que possa

[5] https://canaltech.com.br/seguranca/o-que-e-web-scraping

ser salvo no Apache Kafka, o “Formata/Transforma JSON”. O resultado desse processo pode ser observado na Figura 13.


```
{
  "api_version": "V1.0.12-0.0.54",
  "time_stamp": 1712490631,
  "data_time_stamp": 1712490608,
  "max_age": 604800,
  "firmware_default_version": "7.02",
  "fields": [
    "sensor_index", "last_seen", "rssi", "channel_flags", "confidence", "humidity",
    "temperature", "pressure", "pm2.5"
  ],
  "channel_flags": ["Normal", "A-Downgraded","B-Downgraded", "A+B-Downgraded"],
  "data": [
    [13609, 1712490589, -71, 0, 100, 64, 85, null, 6.0],
    [25499, 1705676782, -71, 0, 53, 56, 92, 992.61, 2.8],
    [25501, 1712490554, -50, 0, 34, 44, 95, 987.44, 0.5]
  ]
}
```


Figura 12: Resultado da requisição dos dados dos sensores (limitado a três sensores para simplificação).


```
{
  "data" : [ {
    "id" : 13609, "last_seen" : 1712490589, "rssi" : -71, "channel_status" : 0,
    "confidence" : 100, "humidity" : 64, "temperature" : 85, "pressure" : null,
    "pm2_5" : 6, "date" : 1712490631
  }, {
    "id" : 25499, "last_seen" : 1705676782, "rssi" : -71, "channel_status" : 0,
    "confidence" : 53, "humidity" : 56, "temperature" : 92, "pressure" : 992.61,
    "pm2_5" : 2.8, "date" : 1712490631
  }, {
    "id" : 25501, "last_seen" : 1712490554, "rssi" : -50, "channel_status" : 0,
    "confidence" : 34, "humidity" : 44, "temperature" : 95, "pressure" : 987.44,
    "pm2_5" : 0.5, "date" : 1712490631
  } ]
}
```


Figura 13: Resultado da transformação do dado de JSON da figura 12.

Após esses processos, o dado JSON é dividido em novos dados JSONs, cada um representando um sensor, ainda no processo “Formata/Transforma JSON”. Após essa divisão, cada conjunto de dados de um sensor é salvo em um tópico no Kafka usando o *processor* “Pública no tópico Kafka”.

No Apache Kafka, os dados enviados pelo NiFi estão sendo salvos no tópico *raw_air_sensors_data*, que contém os dados brutos dos sensores. Este tópico mantém esses dados salvos por um período de dois dias antes de serem excluídos. A intenção é que o fluxo de processamento utilize esses dados em até dois dias.

O *process group* para a ingestão de dados históricos, denominado "Air Quality Sensors Historical", segue um padrão parecido com o de ingestão em tempo real. No entanto, difere por não necessitar de um token para as requisições. Isso ocorre porque a requisição é direcionada ao serviço *air-quality-sensors-historical*, que foi especificamente desenvolvido para fornecer os dados históricos armazenados em um arquivo CSV, como já dito anteriormente.

No Apache Kafka, os dados enviados pelo NiFi, desse *process group*, estão sendo salvos no tópico *raw_air_sensors_historical_data*, que contém os dados históricos brutos dos sensores. Este tópico mantém esses dados salvos por um período de dois dias antes de serem excluídos, da mesma forma que o tópico *raw_air_sensors_data*.

4.3.2. Processamento

Este é o segundo fluxo da arquitetura. O objetivo deste fluxo é processar os dados brutos armazenados no Apache Kafka e, após o processamento, salvar esses dados em novos tópicos destinados a receber esses dados processados. Este fluxo é composto por dois componentes: o Apache Kafka e o Apache Spark.

No Apache Spark, existem três *scripts* desenvolvidos para o processamento desses dados brutos. Eles foram desenvolvidos usando o PySpark[6], que fornece uma interface Python para programação em Spark.

No *script process_air_sensors_metrics_data.py*, estamos processando os dados métricos dos sensores. Primeiramente, ele lê os dados brutos do tópico Kafka *raw_air_sensors_data* e os dados do arquivo *immutable_air_sensors_metrics_data.jsonl*. Este arquivo JSONL foi criado para conter os dados métricos dos sensores que raramente sofrem alterações, como o ID e o nome do município onde o sensor está localizado. Essa estratégia foi adotada para diminuir o tamanho dos dados trafegados no fluxo de ingestão.

Em seguida, o script aplica várias transformações nos dados, incluindo a aplicação da equação de correção LRAPA [16] no campo pm2_5, ajustando a umidade e a temperatura para refletir as diferenças entre as leituras do sensor e as condições ambientais reais, arredondando a pressão e convertendo a data para o fuso horário do Brasil. Após todas essas transformações, o script escreve os dados processados em um novo tópico Kafka, o *processed_air_sensors_metrics_data*.

O script *process_air_sensors_status_data.py* está processando os dados de status dos sensores. Ele lê os dados brutos do tópico Kafka *raw_air_sensors_data* e os dados do arquivo *immutable_air_sensors_status_data,jsonl*, que segue o mesmo esquema do outro arquivo JSONL, com a diferença de que ele possui alguns campos a mais, como o nome e a latitude e longitude do sensor. Em seguida, une os dois conjuntos de dados.

Depois, o script aplica várias transformações nos dados, incluindo a transformação do valor de *rssi* em um valor positivo, alocando-o para um novo campo, o wifi, e em seguida removendo-o. Também renomeia o campo *last_seen* para *last_reading_date*, cria um novo campo (status) baseado no campo *channel_status* e transforma as datas para o fuso horário do Brasil.

Além disso, cria um campo *time_since_last_reading*, que é a diferença entre a data da coleta de dados do sensor e a data da última leitura. Após todas essas transformações, o *script* escreve os dados processados em um novo tópico Kafka, o *processed_air_sensors_status_data*.

O script *process_air_sensors_historical_data.py* processa os dados históricos dos sensores. Ele lê esses dados brutos do tópico Kafka *raw_air_sensors_historical_data*, que contém apenas o município, a medição de pm2.5 e a data. Após a leitura dos dados, o script escreve os dados processados em um novo tópico Kafka, o *processed_air_sensors_historical_data*, sem realizar nenhuma transformação ou refinamento.

4.3.3. Armazenamento

Este é o terceiro fluxo da arquitetura. O objetivo deste fluxo é armazenar os dados processados dos sensores que foram produzidos no fluxo anterior, o de processamento. Além disso, este

[6] https://spark.apache.org/docs/latest/api/python/index.html

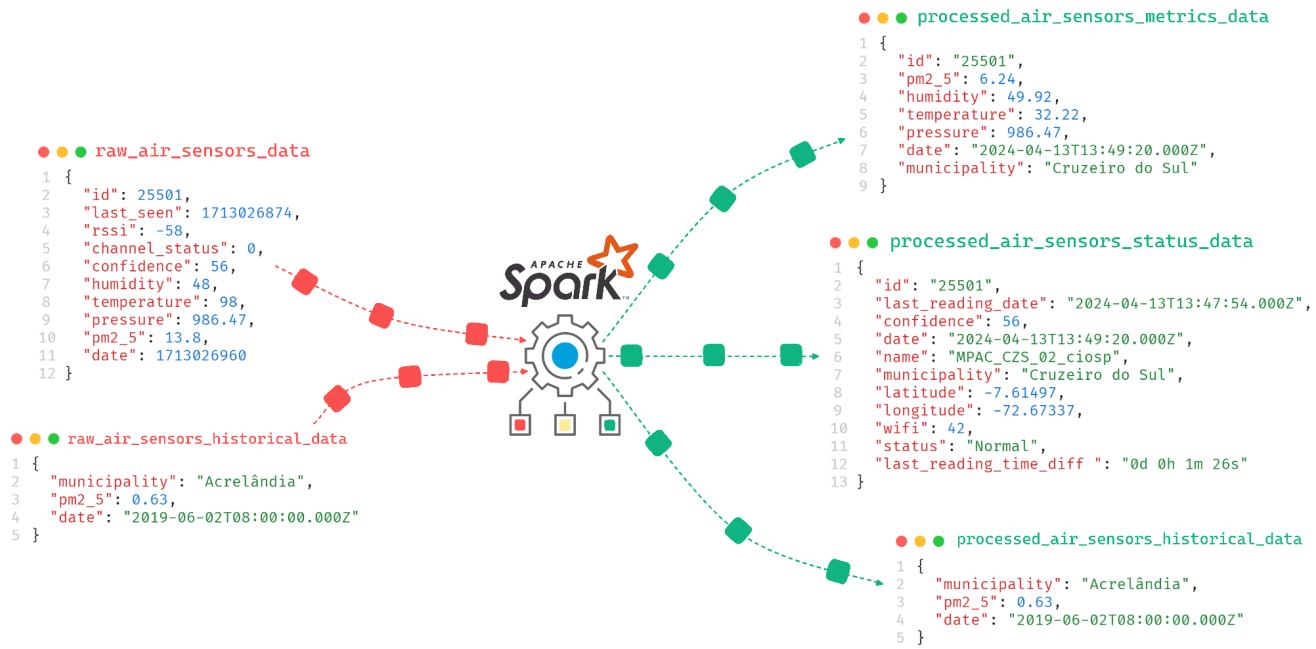

Figura 14: Comparação dos dados dos sensores antes e depois do processamento no Apache Spark.

fluxo também disponibiliza esses dados para serem utilizados, analisados e visualizados para o próximo, o de visualização. Este fluxo é composto por dois componentes: o Apache Kafka e o Apache Druid.

No Apache Druid, foram configuradas três tarefas de ingestão de dados de streaming, conectadas aos três tópicos dos dados processados no Kafka: o *processed_air_sensors_status_data*, *processed_air_sensors_metrics_data* e *processed_air_sensors_historical_data*.

Para consultas em tempo real, os usuários podem consultar diretamente os segmentos de dados que estão sendo atualmente ingeridos pelo Druid. Para consultas históricas, o Druid move os segmentos de dados completamente ingeridos para o "Deep storage" como segmentos históricos. Esses segmentos históricos são então carregados por servidores históricos conforme necessário para responder às consultas dos usuários.

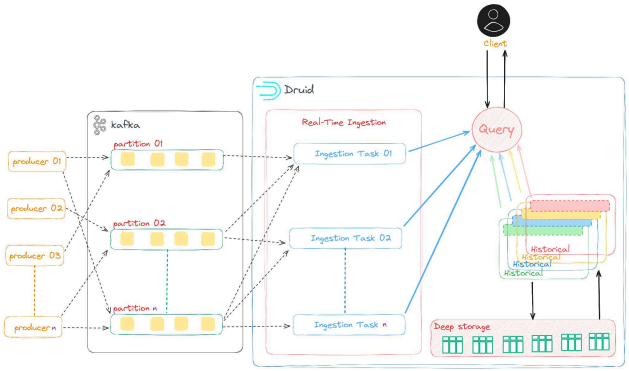

Figura 15: Funcionamento da ingestão e armazenamento de dados entre o Apache Kafka e o Apache Druid.

4.3.4. Visualização

Esta representa o quarto e derradeiro fluxo da arquitetura. A principal finalidade deste fluxo é proporcionar uma apresentação analítica, utilizando-se de técnicas de *Business Intelligence*, dos dados processados e armazenados provenientes dos sensores. Este fluxo é estruturado por dois componentes: o Apache Druid e o Apache Superset. No Apache Superset, é estabelecida uma conexão com o Apache Druid. A partir desta conexão, foram utilizados os conjuntos de dados para a elaboração dos dashboards.

Foram criados três dashboards distintos, cada um atendendo a um propósito específico e segmento da solução ou dos dados coletados. Cada dashboard foi projetado para fornecer *insights* relevantes e úteis para os usuários, permitindo-lhes tomar decisões com base nos dados coletados.

A construção dos dashboards baseou-se na solução já existente [16], levando em consideração as análises viáveis e pertinentes para o conjunto de dados disponíveis. Além disso, eles foram projetados com foco em aprimoramentos e funcionalidades que a solução atual não oferece de maneira completa ou intuitiva, como a filtragem dinâmica.

4.3.4.1 Dashboard de métricas

Esse dashboard, foi criado visando atender a demanda principal da solução, que era possibilitar a análise de forma fácil e intuitiva dos dados de qualidade do ar coletados pelos sensores. O dashboard possui uma divisão principal que distingue as visualizações voltadas para os dados em tempo real e em série histórica. Cada uma dessas visualizações contém quatro guias internas, nomeadas de acordo com os dados da métrica

Figura 16: Painel do PM2.5 em tempo real.

apresentada: PM2.5, Umidade, Temperatura e Pressão Atmosférica. Cada uma dessas subdivisões apresenta componentes gráficos projetados para visualizar informações relevantes à métrica e ao contexto vigente.

Na interface de visualização em tempo real, mostrado na Figura 16, cada guia exibe a informação do valor corrente da métrica em um formato textual. Além disso, o valor da métrica para cada município do Acre é representado por meio de um gráfico de barras. Adicionalmente, um gráfico de medidor (gauge) é fornecido, que também mostra o valor atual, porém contextualizado dentro dos limites apropriados para cada métrica específica.

Na interface de visualização de série histórica, cada guia exibe um gráfico de linhas com o eixo X representando tempo e o eixo Y, o valor da métrica. Além disso, tem-se a informação do valor geral da métrica em um formato textual. Também tem-se o valor da métrica para cada município do Acre representado por meio de um gráfico de barras. Adicionalmente, um mapa colorido é fornecido, onde cada município possui uma cor correspondente com o valor da sua métrica e a escala exibida na legenda. Todas essas guias podem ser visualizadas na Figura 17.

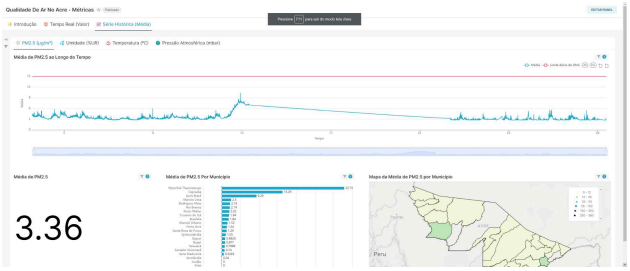

Figura 17: Painel de PM2.5 em série histórica.

4.3.4.2 Dashboard dos sensores

Esse dashboard foi criado visando atender a uma necessidade secundária da solução, que é permitir a análise dos dados do estado dos sensores de maneira fácil e intuitiva. Isso é de extrema importância, pois devemos sempre monitorar e avaliar, na medida do possível, as métricas de confiabilidade, disponibilidade e desempenho desses dispositivos. Isso se deve ao fato de que esses dispositivos geralmente estão sujeitos a condições ambientais adversas, como calor intenso e desgaste mecânico [11].

Dessa forma, esse dashboard apresenta as métricas destinadas a compreender o estado de cada sensor ou de todos eles. Temos, portanto, a métrica de intensidade do sinal Wi-Fi, bem como a métrica de confiabilidade. Além disso, apresentamos o status dos sensores e o registro da data de sua última leitura.

O dashboard segue o mesmo esquema de divisão de visualizações do dashboard de métricas, se diferenciando apenas por possuir uma visualização extra denominada “Visão Geral”. Na interface de “Visão Geral”, dispomos de um mapa que exibe a localização de todos os sensores no estado do Acre. Adicionalmente, apresentamos um gráfico de barras que mostra a quantidade de sensores em cada município. Há também um gráfico de pizza que exibe a quantidade e a porcentagem de sensores, divididos por seu status. Finalizando temos uma tabela que fornece diversas informações relevantes sobre os sensores, incluindo seu nome, o valor de sua confiabilidade e o tempo decorrido desde a sua última leitura. Como pode ser visto na figura abaixo.

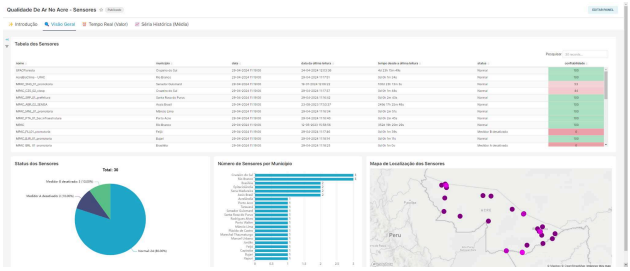

Figura 18: Visualização de visão geral dos sensores.

A interface de visualização em tempo real e série histórica segue o mesmo esquema do dashboard de métricas, diferenciando-se apenas pelas métricas exibidas, que agora são as métricas de estado dos sensores.

4.3.4.3 Dashboard Histórico

Este dashboard foi desenvolvido para atender a uma demanda que surgiu durante o desenvolvimento da solução. A solução original foi projetada para trabalhar com dados em tempo real. No entanto, surgiu a questão se seria possível adaptá-la para funcionar com dados históricos armazenados em um arquivo CSV, por exemplo.

Com a disponibilidade de um arquivo CSV contendo dados dos sensores da medição do PM2.5 entre 01/06/2019 a 02/07/2023, foi possível processar e armazenar esses dados na solução, utilizando-se do serviço *air-quality-sensors-historical*. Sequencialmente, foi criado este dashboard para permitir a visualização e análise desses dados. A figura abaixo mostra como o dashboard pode ser visualizado.

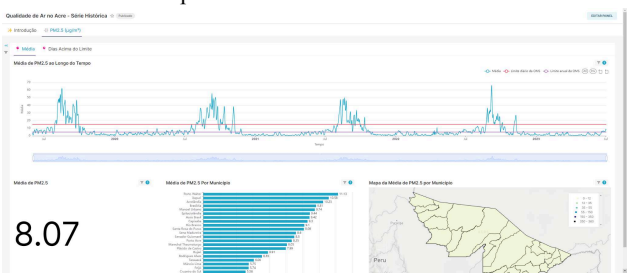

Figura 19: Dashboard de dados históricos.

O dashboard utiliza-se dos mesmos componentes dos outros para apresentar os dados, como o gráfico de barras por município e o gráfico de linhas.

4.4. Conteinerização

A conteinerização é uma abordagem de virtualização que permite executar aplicações e seus ambientes de execução como "containers" isolados. No contexto deste trabalho, foi utilizado o Docker[7], uma plataforma de conteinerização que permite empacotar uma aplicação e suas dependências em um recipiente virtual que pode ser executado em qualquer servidor Linux.

Na arquitetura desenvolvida, cada componente: Nifi, Kafka, Spark, Druid, Superset e Serviços teve um container alocado. Para isso, foi utilizado um arquivo Docker Compose para gerenciar a configuração e a orquestração desses containers. Além disso, foi criado um Dockerfile específico para os serviços. Esse arquivo de texto contém uma série de instruções para construir uma imagem Docker, a partir da qual é possível instanciar um container.

Cada container continha apenas os serviços e ferramentas necessários para o funcionamento de cada componente. Todos esses containers foram executados na mesma rede, chamada "iot_analytics", para permitir a comunicação entre os mesmos. Ademais, foi configurada a nomeação dos volumes de dados para cada aplicação, garantindo que, ao deletar e reiniciar a aplicação, ela continue com as mesmas bases de dados.

5. AVALIAÇÃO

É de suma importância verificar se a interface de visualização de dados da solução é intuitiva e fácil de usar, ou seja, se ela permite aos usuários visualizar e entender os dados facilmente. Para isso, foi utilizado um questionário que foi elaborado através de uma adaptação do *Post-Study System Usability Questionnaire*[8] (PSSUQ), em sua versão 3. Essa adaptação foi realizada para ter apenas questionamentos inerentes ao contexto da solução desenvolvida .

A finalidade primordial deste questionário era coletar dados sobre a satisfação de vários usuários com a solução proposta. O questionário é composto por oito declarações que os usuários devem avaliar com base em suas experiências individuais ao usar a solução. As declarações a seguir foram incluídas no questionário:

- A solução foi simples de utilizar.
- Foi fácil de aprender a usar a solução.
- Eu fui capaz de completar as tarefas e cenários rapidamente utilizando a solução.
- Foi fácil encontrar as informações que eu precisava.
- A organização das informações nas telas da solução eram claras.
- Gostei de usar a interface desta solução.
- Recomendaria o uso dessa solução a outras pessoas.

Cada afirmação foi avaliada em uma escala de 1 a 7, onde 1 indica 'discordo totalmente' e 7 representa 'concordo totalmente'. Cada usuário respondeu ao questionário em duas ocasiões: uma para a solução desenvolvida no âmbito deste trabalho e outra para a solução atualmente em uso, desenvolvida pela UFAC. O objetivo dessa abordagem era comparar as respostas e determinar se houve melhorias da solução existente para a solução desenvolvida.

Antes de iniciar o questionário, cada usuário recebeu uma explicação sobre o intuito de ambas as soluções. As seguintes tarefas foram atribuídas para serem realizadas:

1. Identificar o valor atual de PM2.5 no município de Rio Branco.
2. Identificar o valor médio da temperatura em todo o estado do Acre.
3. Identificar quantos sensores estão com o medidor A desativado no momento.
4. Identificar o valor atual do sinal de WI-FI do sensor MPAC_SNM_01_ifac.
5. Identificar a data específica na qual foi registrada a maior média de PM2.5 no estado do Acre utilizando-se dos dados históricos dos sensores de 2019 até 2023.

É importante enfatizar que todas as tarefas propostas podem ser resolvidas tanto pela solução existente quanto pela solução desenvolvida. A eficácia na resolução dessas tarefas depende, em grande medida, da interação do usuário com a interface de cada solução e da facilidade de uso proporcionada por cada uma delas.

[7] https://www.docker.com

[8] https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire

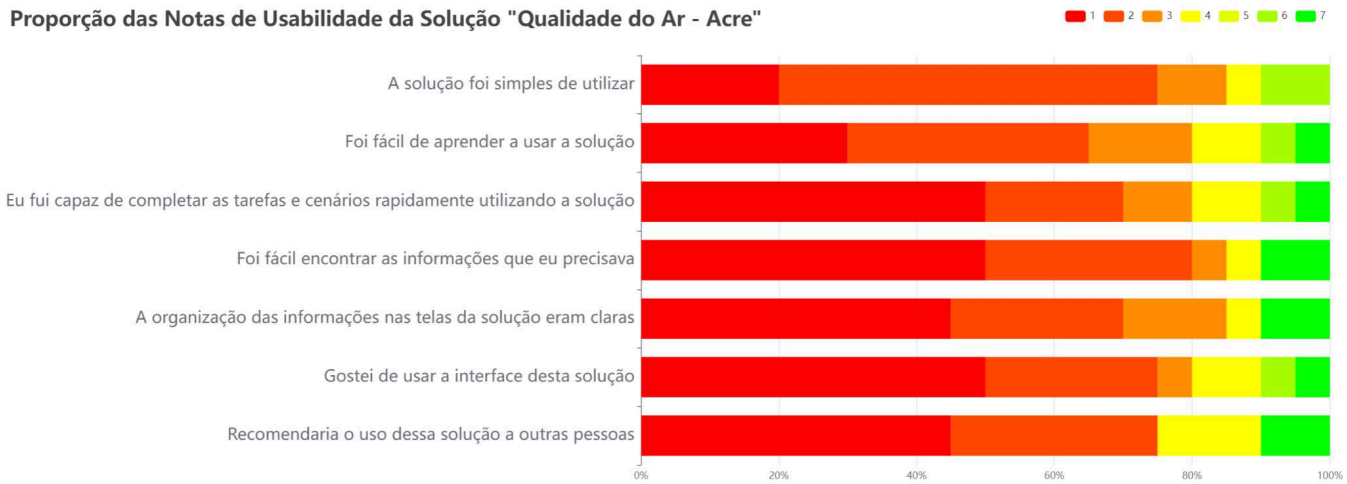

Figura 20: Gráfico do resultado do teste de usabilidade da solução de qualidade do ar do Acre.

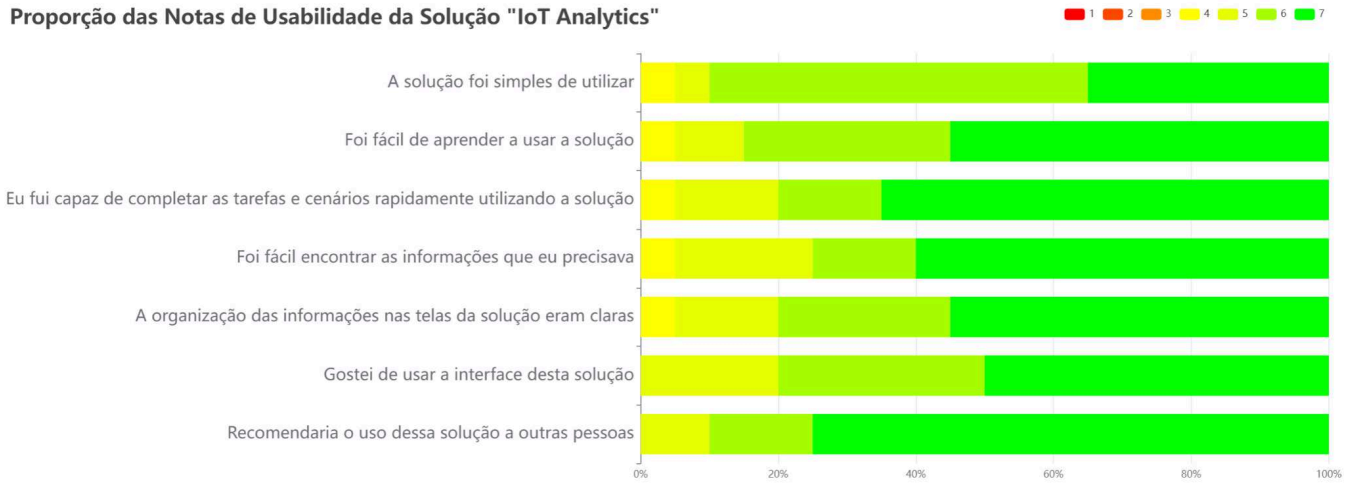

Figura 21: Gráfico do resultado do teste de usabilidade da solução de *IoT Analytics* desenvolvida.

O questionário foi aplicado a um grupo de 20 indivíduos e resultou em *insights* importantes. Quando se trata da conclusão das tarefas, a solução desenvolvida superou significativamente a solução existente, conforme ilustrado na Figura 22. Observou-se que 58% das tarefas foram concluídas exclusivamente com o uso da solução de *IoT analytics*, em contraste com apenas 2% que foram concluídas usando somente a solução existente.

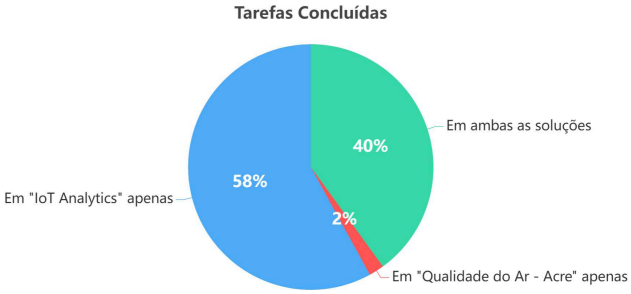

Figura 22: Gráfico de conclusão de tarefas.

Se considerarmos a possibilidade de concluir a tarefa em ambas as soluções como um indicativo de que a tarefa poderia ser concluída em qualquer uma das soluções, então esse percentual aumenta para 78% para a solução de *IoT analytics*, em comparação com 22% para a solução existente.

Com base nesses resultados, podemos inferir que a solução desenvolvida é mais eficaz na obtenção de informações em tempo real e de dados históricos do que a solução existente. Portanto, a solução desenvolvida demonstra um desempenho superior e oferece uma melhoria significativa em relação à solução existente.

No quesito de usabilidade, conforme ilustrado na Figura 20, a predominância das respostas para a solução existente se concentra nas tonalidades de vermelho e cores adjacentes. Isso indica uma tendência dos usuários em discordar das proposições apresentadas. Tal observação sugere que os usuários podem ter encontrado obstáculos ao utilizar a solução, possivelmente relacionados a problemas de usabilidade ou dificuldades na compreensão da interface.

No entanto, também há uma predominância sutil de cores verdes e adjacentes, indicando que alguns usuários concordam com as afirmações. Isso pode sugerir que, embora a maioria dos usuários tenha encontrado dificuldades, alguns usuários foram capazes de usar a solução com sucesso.

Observando agora o resultado do teste para a solução desenvolvida, o *IoT analytics*, na Figura 21, é notável uma predominância de cores verdes e adjacentes, indicando que a maioria dos usuários concorda com as afirmações apresentadas. Isso reflete uma avaliação positiva da usabilidade e eficiência desta solução em comparação com a existente.

Os usuários expressaram satisfação em aspectos cruciais como simplicidade de uso, facilidade de aprendizado, eficiência na realização de tarefas e cenários, clareza na organização das informações nas telas e prazer em usar a interface. A recomendação do uso dessa solução também foi alta, evidenciando sua aceitação e aprovação ampla.

Porém, é notável uma presença discreta de tonalidades mais próximas ao vermelho, sinalizando que uma parcela dos usuários discorda de algumas das afirmações. Isso pode indicar que, apesar de a maioria dos usuários considerar a solução fácil de usar e eficaz, alguns podem ter encontrado desafios ou dificuldades ao utilizá-la, possivelmente por serem novatos no uso de uma ferramenta de BI.

6. CONCLUSÃO

A implementação da solução de *IoT Analytics* para sensores ambientais sinaliza um avanço considerável no monitoramento ambiental no Acre. A integração de tecnologias de Internet das Coisas (IoT), análise de dados e visualização em tempo real e série histórica proporcionou *insights* profundos sobre as condições ambientais e o estado da rede de sensores, facilitando a tomada de decisões embasadas. A comparação com o sistema existente, no estado do Acre, destacou a eficácia e a superioridade da solução proposta, enfatizando sua capacidade de fornecer informações precisas e atualizadas de maneira mais eficiente. A expectativa é que essa solução não apenas beneficie o Acre, mas também possa ser adaptada e replicada em outros locais, contextos ou situações, melhorando o monitoramento ambiental e protegendo a saúde pública em escala global. Esta solução representa um marco importante para a utilização eficaz de tecnologias emergentes na proteção do meio ambiente e da saúde pública.

REFERÊNCIAS

[1] Dapper, S. N.; Spohr, C.; Zanini, R. R. Poluição do ar como fator de risco para a saúde: uma revisão sistemática no estado de São Paulo. *Estudos Avançados* **2016**, 30 (86), 83–97. https://doi.org/10.1590/s0103-40142016.00100006.

[2] Raja, P.; Kumar, S.; Yadav, D. S.; Singh, T. The Internet of Things (IOT): A Review of Concepts, Technologies, and Applications. *International Journal of Information Technology and Computer Engineering* **2023**, No. 32, 21–32. https://doi.org/10.55529/ijitc.32.21.32.

[3] Sinha, S. *State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally*. IoT Analytics. https://iot-analytics.com/number-connected-iot-devices/.

[4] Kim, J. H.; Jo, B.-W.; Jo, J. H.; Kim, D. K. Development of an IoT-Based Construction Worker Physiological Data Monitoring Platform at High Temperatures. *Sensors* **2020**, 20 (19), 5682. https://doi.org/10.3390/s20195682.

[5] Wiryasaputra, R.; Huang, C.; Lin, Y.-J.; Yang, C.-T. An IoT Real-Time Potable Water Quality Monitoring and Prediction Model Based on Cloud Computing Architecture. *Sensors* **2024**, 24 (4), 1180. https://doi.org/10.3390/s24041180.

[6] Murthy, M. K.; Kumar, R.; Saikiran, B.; Nagaraj, I.; Annavarapu, T. Real Time Weather Monitoring System using IoT. *E3S Web of Conferences* **2023**, 391, 01142. https://doi.org/10.1051/e3sconf/202339101142.

[7] Pragadeswaran, S.; Vishnu, S.; Surya, V.; Kurup, V.; Tamilselvan, S. An Investigation on Real Time Monitoring System for Livestock and Agriculture using IoT. *International Journal of Advanced Research in Science, Communication and Technology* **2023**, 102–109. https://doi.org/10.48175/ijarsct-8566.

[8] *Projeto desenvolvido na Ufac monitora qualidade do ar no Acre — Universidade Federal do Acre*. https://www.ufac.br/site/noticias/2023/projeto-desenvolvido-na-ufac-monitora-qualidade-do-ar-no-acre.

[9] Zilber, S. N. Redução do consumo de energia em uma fábrica de escovas de dente e creme dental através do uso de ferramentas de IoT para monitoramento do consumo de ar comprimido. *XLII Encontro Nacional de Engenharia de Produção, Foz do Iguaçu, Paraná, Brasil*, **04-07 out. 2022**. https://abepro.org.br/biblioteca/TCE_382_1886_43249.pdf

[10] Souza Júnior, P. I. Análise do consumo de combustível após implantação de automação para controle de abastecimento através de ferramenta Data Analytics e IoT. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - *Departamento de Engenharia de Computação e Automação, Universidade Federal do Rio Grande do Norte* **2023**, 36 f. https://repositorio.ufrn.br/handle/123456789/53807.

[11] Aikhuele, D. O.; Nwosu, H. U.; Ighravwe, D. E. Data-driven model for the evaluation of the reliability of sensors and actuators used in IoT system architecture. *Journal of Reliable Intelligent Environments* **2022**, 9 (2), 135–145. https://doi.org/10.1007/s40860-022-00179-0.

[12] Ouyang, R.; Wang, J.; Xu, H.; Chen, S.; Xiong, X.; Tolba, A.; Zhang, X. A microservice and serverless architecture for secure IoT system. *Sensors* **2023**, 23 (10), 4868. https://doi.org/10.3390/s23104868.

[13] Lo, S. K.; Liew, C. S.; Soon, T. K.; Mekhilef, S. An Interoperable Component-Based Architecture for Data-Driven IoT System. *Sensors* **2019**, 19 (20), 4354. https://doi.org/10.3390/s19204354.

[14] De Vito, S.; Esposito, E.; Massera, E.; Formisano, F.; Fattoruso, G.; Ferlito, S.; Del Giudice, A.; D'Elia, G.; Salvato, M.; Polichetti, T.; D'Auria, P.; Ionescu, A. M.; Di Francia, G.Crowdsensing IoT Architecture for Pervasive Air Quality and Exposome Monitoring: Design, Development, Calibration, and Long-Term Validation. *Sensors* **2021**, 21 (15), 5219. https://doi.org/10.3390/s21155219.

[15] Martínez, A.; Hernández-Rodríguez, E.; Hernández, L.; Schalm, O.; González-Rivero, R. A.; Alejo-Sanchez, D. Design of a Low-Cost System for the Measurement of Variables Associated With Air Quality. *IEEE Embedded Systems Letters* **2023**, 15 (2), 105–108. https://doi.org/10.1109/les.2022.3196543.

[16] *Qualidade do Ar no Acre*. http://www.acrequalidadedoar.info/.

[17] PurpleAir, Inc. *Real-time air quality monitoring by PurpleAir*. PurpleAir, Inc. https://www2.purpleair.com/.

[18] Gill, N. S. *IoT Analytics Platform for Real-Time Data Ingestion*. XenonStack. https://www.xenonstack.com/blog/iot-analytics-platform.

[19] *Apache NiFi*. Apache NiFi. https://nifi.apache.org/.

[20] *Apache Kafka*. Apache Kafka. https://kafka.apache.org/.

[21] *Apache Spark*. Apache Spark.. https://spark.apache.org/.

[22] *Apache Druid.* Apache Druid. https://druid.apache.org/.

[23] *Superset*. Superset. https://superset.apache.org/.