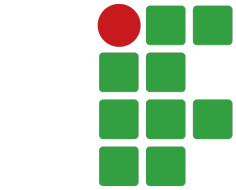

INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Sistema modular de coleta de dados da qualidade do ar

Arthur Cadore Matuella Barcella
Gustavo Paulo
Matheus Pires Salaza
Rhenzo Hideki Silva Kajikawa

Engenharia de Telecomunicações - IFSC-SJ

Sumário

1. Introdução

O presente relatório documenta o desenvolvimento de um sistema integrado que abrange hardware, backend e frontend. O projeto, concebido e desenvolvido por PJI029008, sendo o grande motivador as queimadas que ocorreram no segundo semestre de 2024, junto com outros estudos públicados da mesma época debatendo assuntos climáticos como ilhas de calor.

Este projeto tem como foco a coleta, processamento e visualização de dados provenientes de diversos sensores conectados a um ESP32. O sistema utiliza o protocolo MQTT para a comunicação dos dispositivos e conta com um conjunto de aplicações que englobam tanto a parte física (hardware) quanto as camadas de software (backend e frontend) para gerenciar e exibir as informações.

1.1. Objetivo Geral

Desenvolver um sistema integrado para monitoramento de variáveis ambientais, que permita a coleta de dados através de sensores (temperatura, pressão, gás, umidade, luminosidade, etc.) conectados a um ESP32, e a transmissão desses dados via MQTT para uma plataforma central que os processa e disponibiliza em uma interface web.

1.2. Objetivo Específicos

Projetar e construir um hardware próprio baseado em ESP32, com sensores adequados e circuitos de interface (incluindo reguladores de tensão e protoboard) para a medição das grandezas ambientais. Implementar uma biblioteca modular para cada sensor, de modo a facilitar a manutenção e a extensão do código. Estabelecer uma comunicação padrão via MQTT, definindo a estrutura das mensagens e os símbolos delimitadores para garantir a correta interpretação dos dados. Desenvolver um backend capaz de receber, armazenar e processar os dados enviados pelos dispositivos. Criar um frontend intuitivo que permita a visualização em tempo real dos dados coletados e a realização de análises históricas. Fornecer documentação completa para que outros desenvolvedores possam replicar ou adaptar o sistema com seus próprios dispositivos.

2. Fundamentação Teórica

Como o projeto tem um escopo grande , foi fundamental dividir este projeto em partes menores. Assim o projeto foi dividido em partes menores, sendo essas o frontend, hardware , backend. Será discutido mais a fundo cada frente desse projeto mais á frente.

Com essa divisão de tarefas possibilitou que os diferentes grupos tivessem melhor controle para fazer suas pesquisas e decisões em relação ao que era melhor para o projeto.

2.1. Fundamentação do Hardware

O desenvolvimento do hardware envolveu diversos desafios, incluindo a seleção dos sensores necessários, a definição dos protocolos de comunicação, a escolha do tipo de transmissão e, principalmente, a determinação do microcontrolador mais adequado para a aplicação.

2.1.1. Escolha do Microcontrolador

O microcontrolador é o núcleo do dispositivo IoT, responsável pelo processamento de dados e controle dos periféricos. Sua escolha depende de fatores como consumo de energia, conectividade, compatibilidade com os sensores, facilidade de desenvolvimento e custo. As principais opções avaliadas foram:

- ESP32: Oferece conectividade Wi-Fi e Bluetooth integrada, possui consumo energético moderado com modos de economia de energia e é compatível com interfaces I2C, SPI e UART. Além disso, é amplamente utilizado e de fácil programação.
- STM32: Destaca-se pelo consumo extremamente baixo e oferece diversas opções de conectividade, incluindo LoRa com módulos adicionais. Conta com periféricos avançados e é programado principalmente através do STM32CubeMX.
- Arduino MKR WAN 1310: Projetado para facilidade de uso, já conta com conectividade LoRa integrada. Compatível com I2C, SPI e UART, inclui recursos como carregamento de bateria integrado e é programado via Arduino IDE.
- Adafruit Feather M0 LoRa: Compacto e leve, com conectividade LoRa integrada. Também compatível com I2C, SPI e UART, sendo programado via Arduino IDE, ideal para aplicações que requerem um dispositivo pequeno e eficiente.

Após a análise dessas opções, o ESP32 foi escolhido por sua flexibilidade, facilidade de uso e custo reduzido, uma vez que a equipe já possuía acesso a esse microcontrolador.

2.1.2. Seleção do Protocolo de Comunicação

A definição do protocolo de comunicação foi um aspecto crítico para garantir eficiência e confiabilidade na transmissão dos dados. As opções consideradas foram:

- CoAP: Utiliza o modelo REST sobre UDP e é adequado para dispositivos com recursos restritos, garantindo comunicação eficiente com baixa sobrecarga.
- HTTP: Amplamente utilizado, mas pode ser pesado para dispositivos com recursos limitados devido à sua sobrecarga. Mais indicado para aplicações com maior capacidade de processamento.

- LoRaWAN: Focado em comunicação de longa distância com baixo consumo de energia, sendo ideal para transmissão de dados a longas distâncias com baixa taxa de dados.
- MQTT: Protocolo leve, baseado no modelo de publicação/assinatura, ideal para dispositivos com recursos limitados e redes de baixa largura de banda.

O protocolo MQTT foi escolhido devido à sua leveza, facilidade de implementação e familiaridade da equipe com sua utilização.

2.1.3. Escolha dos Sensores

Os sensores são componentes essenciais do projeto, permitindo a coleta de dados ambientais. A seleção foi baseada nas grandezas físicas a serem monitoradas, resultando na escolha dos seguintes dispositivos:

- DS18B20: Sensor de temperatura digital de alta precisão, utilizando comunicação 1-Wire.
- BMP280: Sensor barométrico que mede pressão atmosférica e temperatura, compatível com interfaces I2C e SPI.
- MQ-2 e MQ-7: Sensores para detecção de gases como GLP, metano e monóxido de carbono.
- DHT-22: Sensor para medição de temperatura e umidade relativa do ar, com comunicação digital simples.
- LDR: Resistor dependente de luz utilizado para medir a intensidade luminosa do ambiente.

Com essa configuração de hardware, o projeto garante um equilíbrio entre eficiência, confiabilidade e viabilidade econômica.

2.2. Fundamentação Teórica do Backend

O presente sistema de monitoramento da qualidade do ar apresenta uma arquitetura modular, cuja parte de backend integra os elementos necessários para o processamento, armazenamento e disponibilização dos dados provenientes dos dispositivos IoT. A seguir, descreve-se os principais componentes teóricos que embasam o desenvolvimento do backend.

2.2.1. MQTT Broker

O MQTT (Message Queuing Telemetry Transport) é um protocolo leve de publicação/assinatura, ideal para a comunicação entre dispositivos com recursos limitados. No contexto deste projeto, o MQTT Broker é responsável por:

- Estabelecer a comunicação em tempo real entre os dispositivos de campo e o backend;
- Garantir baixa latência e overhead reduzido, mesmo em redes de baixa largura de banda;
- Facilitar a escalabilidade do sistema por meio do roteamento eficiente das mensagens.

2.2.2. Backend Connector

O Backend Connector atua como a ponte entre o MQTT Broker e os demais módulos do sistema. Sua função é processar, transformar e direcionar as mensagens recebidas para o armazenamento ou para a exposição via API. Dentre suas atribuições, destaca-se:

- A integração dos dados oriundos dos sensores com o banco de dados;

- A implementação de rotinas de pré-processamento e validação das mensagens;
- O encaminhamento das informações para a API que será consumida pelo frontend.

2.2.3. BANCO DE DADOS

O banco de dados constitui o repositório central de armazenamento das informações coletadas. Sua escolha baseou-se na necessidade de:

- Armazenamento confiável e estruturado dos dados históricos e em tempo real;
- Suporte a consultas rápidas e escaláveis, mediante a utilização de soluções relacionais ou NoSQL;
- Manutenção da integridade e segurança das informações.

2.2.4. API FRONTEND CONNECTOR

A API Frontend Connector é a interface que disponibiliza os dados armazenados para consumo pelas aplicações do frontend. Esta camada é implementada, geralmente, com princípios RESTful, visando:

- A padronização e segurança no acesso aos dados;
- A interoperabilidade com aplicações externas e dispositivos móveis;
- A exposição de endpoints para consulta, atualização e gerenciamento dos dados.

2.2.5. FRONTEND HTTP SERVER (REVERSE PROXY)

O Frontend HTTP Server, atuando como um reverse proxy, gerencia as requisições oriundas do frontend, assegurando:

- O balanceamento de carga e a distribuição eficiente das requisições;
- A implementação de mecanismos de cache e segurança;
- A facilitação do acesso a conteúdos estáticos e dinâmicos, integrando o sistema de forma transparente.

2.3. Fundamentação do Frontend

O desenvolvimento do frontend do sistema modular de coleta de dados da qualidade do ar teve como objetivo principal criar uma interface intuitiva, responsiva e eficiente para a visualização dos dados coletados pelos sensores. Para alcançar esses objetivos, foram adotadas tecnologias modernas que garantem boa performance, escalabilidade e facilidade de manutenção.

2.3.1. Tecnologias Utilizadas

A escolha das tecnologias para o frontend considerou aspectos como facilidade de desenvolvimento, performance e compatibilidade com a arquitetura do projeto. As principais tecnologias utilizadas foram:

- React.js: Biblioteca JavaScript utilizada para a construção da interface do usuário de forma modular e reutilizável. Sua abordagem baseada em componentes facilita a manutenção e evolução do sistema.
- Tailwind CSS: Framework CSS utilizado para estilização da interface, proporcionando uma abordagem eficiente e flexível na personalização do design.
- Vite: Ferramenta utilizada para o build e desenvolvimento do frontend, garantindo maior velocidade e otimização na entrega do código ao usuário.
- Axios: Biblioteca para realizar requisições HTTP e facilitar a comunicação entre o frontend e a API do backend.
- Recharts: Biblioteca para a visualização gráfica dos dados coletados pelos sensores, permitindo a criação de gráficos dinâmicos e interativos.

2.3.2. Estrutura do Frontend

O frontend foi projetado para fornecer uma experiência fluida e responsiva aos usuários, garantindo a apresentação eficiente dos dados coletados. A estrutura principal foi dividida em três seções:

- Página Principal: Exibe um mapa que mostra a localização dos dispositivos do usuário em questão.
- Cooperativas: Tela que o administrador do sistema pode visualizar todas as cooperativas cadastradas e suas respectivas informações.
- Usuários: Tela em que o dono da cooperativa pode acessar e gerenciar as informações dos usuários de sua cooperativa.
- Dispositivos: Mostra cada dispositivo que o usuário tiver, também pode direcionar para visualizar informações detalhadas das medições dos sensores, exibindo os dados coletados pelos sensores em tempo real, utilizando gráficos interativos e indicadores numéricos. Permitindo a visualização de dados passados, possibilitando a análise de tendências e padrões ambientais.

2.3.3. Comunicação com o Backend

A interação entre o frontend e o backend foi implementada através de uma API REST, permitindo o consumo dos dados processados pelo servidor. O fluxo de comunicação segue os seguintes passos:

1. O frontend realiza uma requisição HTTP à API para obter os dados coletados.
2. A API retorna as informações em formato JSON.
3. O frontend processa os dados e os exibe de maneira visualmente compreensível para o usuário.

A escolha das tecnologias e a estruturação do frontend proporcionam diversos benefícios ao projeto, tais como:

- Responsividade: A interface se adapta a diferentes tamanhos de tela, garantindo uma boa experiência de uso em dispositivos móveis e desktops.
- Performance otimizada: O uso de React.js e Vite melhora o tempo de carregamento da aplicação, proporcionando uma navegação fluida.
- Modularidade: A abordagem baseada em componentes facilita a manutenção e expansão da aplicação.
- Atualização em tempo real: Permitindo que os dados sejam exibidos de forma dinâmica, sem a necessidade de recarregar a página

3. Metodologia

A metodologia adotada para o desenvolvimento do sistema modular de coleta de dados da qualidade do ar envolveu um planejamento detalhado e uma execução integrada entre as diversas frentes do projeto. Foram consideradas e sistematizadas as etapas de seleção, integração e validação dos componentes, garantindo que as escolhas fossem fundamentadas em critérios objetivos, embasamento teórico e na experiência prévia da equipe.

3.1. Levantamento de Requisitos e Planejamento

- Definição dos Objetivos:

Inicialmente, foram identificadas as necessidades do projeto, considerando a importância do monitoramento ambiental e a relevância dos dados coletados para a avaliação da qualidade do ar.

- Análise dos Conceitos:

Realizou-se uma revisão dos conceitos teóricos relacionados à Internet das Coisas (IoT), microcontroladores, protocolos de comunicação e sensores, a fim de embasar a escolha dos componentes e orientar a implementação de cada módulo.

- Identificação dos Componentes Essenciais:

Foram definidos os sensores a serem utilizados – DS18B20, BMP280, MQ-2, MQ-7, DHT-22 e LDR – bem como o microcontrolador ESP32, levando em consideração fatores como custo, flexibilidade, compatibilidade e disponibilidade.

- Estudo dos Protocolos de Comunicação:

A análise comparativa dos protocolos (CoAP, HTTP, LoRaWAN e MQTT) evidenciou que o MQTT era o mais adequado para o cenário de redes com largura de banda restrita, devido à sua leveza e eficiência na transmissão dos dados.

3.2. Seleção e Desenvolvimento do Hardware

Esta etapa envolveu a escolha criteriosa dos componentes e a integração física dos mesmos:

3.2.1. Escolha dos Componentes

- Microcontrolador:

Foram avaliadas opções como ESP32, STM32, Arduino MKR WAN 1310 e Adafruit Feather M0 LoRa. Critérios de avaliação: consumo de energia, conectividade (Wi-Fi, Bluetooth, LoRa), facilidade de programação e custo. Decisão: O ESP32 foi escolhido por sua flexibilidade, facilidade de uso e custo reduzido, além de já estar disponível para a equipe.

- Protocolo de Comunicação:

Foram analisados diferentes protocolos quanto à eficiência e à sobrecarga para dispositivos com recursos limitados. Decisão: O protocolo MQTT foi selecionado pela sua leveza, facilidade de implementação e pela compatibilidade com a infraestrutura do projeto.

- Sensores:

A seleção dos sensores levou em conta as grandezas físicas a serem monitoradas e a precisão necessária para cada medição. Componentes escolhidos:

- DS18B20: Sensor de temperatura digital com alta precisão (comunicação 1-Wire).
- BMP280: Sensor barométrico que mede pressão atmosférica e temperatura (interfaces I2C/SPI).
- MQ-2 e MQ-7: Sensores para detecção de gases como GLP, metano e monóxido de carbono.
- DHT-22: Sensor para medição de temperatura e umidade com interface digital.
- LDR: Utilizado para medir a intensidade luminosa do ambiente.

3.2.2. Projeto e Montagem do Circuito

- Desenho do Circuito:

Com os componentes selecionados, foi realizado o projeto do circuito, garantindo a correta interligação entre sensores, microcontrolador e demais periféricos, com atenção especial à estabilidade dos sinais.

- Montagem e Testes Iniciais:

Após a montagem, foram executados testes individuais para verificar o funcionamento de cada sensor e a integridade das conexões, permitindo ajustes e refinamentos que garantiram a precisão na coleta dos dados.

3.3. Implementação do Firmware

- Desenvolvimento do Firmware:

Foi criado um firmware específico para o ESP32, que integra a leitura dos sensores e a transmissão dos dados via MQTT.

- Modularização:

Bibliotecas modulares foram desenvolvidas para cada sensor, facilitando futuras manutenções e expansões do sistema.

- Testes Unitários:

Cada módulo foi testado individualmente para assegurar seu funcionamento correto e a eficácia da comunicação entre os sensores e o microcontrolador.

3.4. Planejamento e Levantamento de Requisitos

Inicialmente, foram definidos os requisitos funcionais e não funcionais do sistema, considerando a necessidade de:

- Recepção e processamento de mensagens em tempo real dos dispositivos IoT;
- Armazenamento seguro e escalável dos dados coletados;
- Disponibilização dos dados para aplicações externas e para o frontend;
- Garantir a robustez e a interoperabilidade entre os módulos do backend.

3.5. Desenvolvimento Modular

O desenvolvimento foi dividido em módulos, conforme descrito abaixo:

3.5.1. Implementação do MQTT Broker

- Configuração do ambiente para recepção de mensagens dos dispositivos;
- Realização de testes de performance para validar a comunicação em tempo real;
- Ajustes na configuração para assegurar baixa latência e alta disponibilidade.

3.5.2. Desenvolvimento do Backend Connector

- Criação de rotinas para processamento e roteamento das mensagens recebidas;
- Integração com o banco de dados para armazenamento dos dados;
- Implementação de mecanismos de validação e transformação dos dados.

3.5.3. Configuração do Banco de Dados

- Escolha do modelo de banco de dados (relacional ou NoSQL) de acordo com a necessidade de escalabilidade;
- Estruturação dos esquemas de dados e implementação de índices para otimização das consultas;
- Testes de integridade e desempenho do armazenamento.

3.5.4. Desenvolvimento da API Frontend Connector

- Implementação dos endpoints seguindo boas práticas de desenvolvimento (RESTful ou GraphQL);
- Garantia da segurança dos dados por meio de mecanismos de autenticação e autorização;
- Integração com o frontend para exposição dos dados de forma padronizada.

3.5.5. Configuração do Frontend HTTP Server (Reverse Proxy)

- Implantação do servidor HTTP para gerenciamento das requisições do frontend;
- Implementação de funcionalidades de cache e balanceamento de carga;
- Configuração de segurança adicional para proteger as comunicações.

3.5.6. Testes e Validação

Foram conduzidos testes unitários e de integração em cada módulo, com o objetivo de:

- Validar o fluxo de dados, desde a entrada via MQTT até a disponibilização via API;
- Identificar e corrigir gargalos de desempenho;
- Assegurar a robustez e a escalabilidade do sistema sob diferentes cargas de trabalho.

3.5.7. Deploy e Gestão

Utilizou-se controle de versões (Git) e gerenciamento de releases (exemplo: release beta_1.0.0) para acompanhar o desenvolvimento. Adicionalmente, a utilização de containers (Docker) e orquestração (Kubernetes) foi considerada para facilitar o deploy e a escalabilidade da solução.

3.6. Desenvolvimento do Backend e Frontend

- Backend:

Foi implementado um banco de dados para o armazenamento das informações coletadas, permitindo o gerenciamento, consulta e análise histórica dos dados. Uma API foi desenvolvida em Go, escolhida por seu desempenho e escalabilidade, garantindo a correta interpretação das mensagens MQTT e seu armazenamento.

- Frontend:

Desenvolvida uma interface web intuitiva e responsiva utilizando React JS e Tailwind CSS, que possibilita a visualização em tempo real e a análise histórica dos dados. Testes de usabilidade foram conduzidos para garantir uma experiência interativa e eficiente para o usuário.

3.7. Integração e Validação do Sistema

- Integração:

Todas as camadas do sistema – hardware, firmware, backend e frontend – foram integradas e validadas de forma colaborativa. Testes de Integração: Foram realizados testes em ambiente controlado para verificar a comunicação via MQTT e a robustez do fluxo completo de dados, desde a coleta até a exibição.

- Ajustes e Otimizações:

Com base nos resultados dos testes, foram implementadas melhorias que asseguraram maior eficiência e robustez do sistema.

3.8. Documentação e Gestão do Projeto

- Controle de Versões:

O desenvolvimento foi acompanhado através de versionamento no GitHub, permitindo rastreabilidade e colaboração contínua.

- Reuniões e Feedback:

Reuniões periódicas e revisões de progresso foram realizadas para alinhar expectativas, identificar problemas precocemente e implementar soluções eficazes.

- Registro e Análise:

Toda a metodologia e as decisões técnicas foram documentadas, permitindo uma análise crítica dos resultados e a identificação de pontos de melhoria para projetos futuros.

4. Resultados e Discussão

4.1. Resultados do Hardware

Esta seção apresenta os resultados obtidos com o desenvolvimento do hardware, destacando as conquistas e as limitações encontradas durante o processo, bem como perspectivas para melhorias futuras.

4.1.1. Componentes

Resultados:

Foram selecionados e integrados o microcontrolador ESP32 e os sensores DS18B20, BMP280, MQ-2, MQ-7, DHT-22 e LDR, com base em critérios como custo, flexibilidade e compatibilidade.

O protótipo demonstrou a viabilidade técnica de coletar dados ambientais por meio desses componentes, permitindo a leitura de temperatura, pressão, gases e intensidade luminosa.

Discussão e Perspectivas Futuras:

Embora a seleção dos componentes tenha permitido a obtenção de dados, constatou-se que nem todos os sensores atingiram o nível de precisão necessário para análises aprofundadas da qualidade do ar.

Revisar a qualidade dos sensores e incluir dispositivos adicionais para a detecção de outros gases, ampliando a abrangência das medições e melhorando a acurácia dos resultados.

Considerar alternativas que possibilitem um hardware autônomo com maior vida útil via recarga, atendendo à proposta original de um sistema robusto e sustentável.

4.1.2. Conexão

Resultados:

A conexão entre os sensores e o microcontrolador foi validada com sucesso utilizando a interface Wi-Fi do ESP32.

Testes iniciais demonstraram a estabilidade das conexões e a integridade dos sinais, embora os testes tenham sido realizados com o dispositivo conectado à rede elétrica.

Discussão e Perspectivas Futuras:

A dependência exclusiva do Wi-Fi e da alimentação contínua compromete a aplicação do sistema em áreas isoladas, onde a conectividade e o acesso à energia são limitados.

Explorar tecnologias de comunicação LPWA (como LoRaWAN) para garantir conectividade em locais remotos e reduzir o consumo energético.

Investir na otimização do layout do circuito para aprimorar a estabilidade dos sinais e facilitar a transição para uma operação autônoma.

4.1.3. Transmissão

Resultados:

A transmissão dos dados foi implementada com sucesso utilizando o protocolo MQTT, que se mostrou eficiente para a comunicação em redes com largura de banda restrita.

A estrutura de mensageria dos pacotes permitiu a transmissão correta dos dados coletados, conforme documentado na estrutura de mensagens.

Discussão e Perspectivas Futuras:

Apesar da eficácia do MQTT no ambiente de testes, a escalabilidade e robustez da transmissão podem ser limitadas em cenários com maior volume de dados ou requisitos específicos de latência.

Avaliar a possibilidade de adotar ou desenvolver um protocolo customizado que atenda melhor às demandas futuras do sistema, garantindo maior segurança e eficiência na comunicação.

Implementar rotinas de economia de energia (por exemplo, funções de sleep no firmware) para otimizar o consumo e prolongar a autonomia do dispositivo.

4.2. Resultados Backend

A seguir, apresentam-se os resultados obtidos com o backend, bem como as discussões acerca das limitações e das perspectivas futuras.

4.2.1. Resultados Obtidos

- MQTT Broker:Foi possível estabelecer uma comunicação em tempo real com baixa latência, garantindo a transmissão contínua dos dados dos dispositivos IoT para o backend.
- Backend Connector:O processamento e o roteamento das mensagens ocorreram de forma eficiente, integrando os dados ao banco de dados e assegurando a atualização constante dos registros.
- Banco de Dados:Os dados foram armazenados de forma estruturada e segura, permitindo consultas históricas e em tempo real, atendendo aos requisitos de escalabilidade e integridade.
- API Frontend Connector:Os endpoints desenvolvidos possibilitaram o acesso seguro e padronizado aos dados, demonstrando compatibilidade e integração satisfatória com o frontend.
- Frontend HTTP Server (Reverse Proxy):A configuração do servidor permitiu o balanceamento de carga e a otimização do desempenho, contribuindo para a segurança e a eficiência na entrega dos conteúdos.

4.2.2. Discussões e Perspectivas Futuras

Apesar dos resultados promissores, alguns desafios e limitações foram identificados:

- Desempenho do MQTT Broker:Em cenários de alta demanda, a performance pode ser otimizada para reduzir eventuais gargalos na comunicação.
- Integração do Backend Connector com o Banco de Dados:Sob elevado volume de mensagens, podem ocorrer sobrecargas que exigirão ajustes na estrutura de dados e na capacidade de processamento.
- Segurança e Escalabilidade da API e do Servidor HTTP:É necessário investir em mecanismos adicionais de segurança e em testes de carga para garantir a resiliência do sistema em ambientes de produção.

Perspectivas Futuras:

- Otimização dos módulos para reduzir latências e aumentar a robustez;
- Exploração de bancos de dados híbridos ou distribuídos para suportar o crescimento do volume de dados;
- Implementação de testes de carga e segurança para aprimorar a confiabilidade do sistema;
- Integração de novas funcionalidades, como análise preditiva por meio de Inteligência Artificial, ampliando a capacidade de interpretação dos dados e a interoperabilidade com outros sistemas.

4.3. Frontend

4.4. Resultado Geral

5. Considerações Finais

6. Referências