UFRR

UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Pedro Aleph Gomes de Souza Vasconcelos

Sistema Integrado de Monitoramento Climático na Amazônia com Tecnologias Abertas

Boa Vista - RR
12 de novembro de 2025

Sistema Integrado de Monitoramento Climático na Amazônia com Tecnologias Abertas

Pedro Aleph Gomes de Souza Vasconcelos

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação do Departamento de Ciência da Computação da Universidade Federal de Roraima, em cumprimento às exigências legais como requisito à obtenção do título Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Herbert Oliveira Rocha.

Boa Vista - RR
12 de novembro de 2025

Dados Internacionais de Catalogação na publicação (CIP)
Biblioteca Central da Universidade Federal de Roraima

V331s Vasconcelos, Pedro Aleph Gomes de Souza.
Sistema integrado de monitoramento climático na Amazônia com tecnologias abertas / Pedro Aleph Gomes de Souza Vasconcelos. – Boa Vista, 2025.
33 f. : il. Inclui Anexo.

Orientador: Prof. Dr. Herbert Oliveira Rocha.

Monografia (graduação) – Universidade Federal de Roraima, Curso de Ciência da Computação.

1. Estações meteorológicas. 2. API RESTful. 3. Interface de usuário. I. Título. II. Rocha, Herbert Oliveira (orientador).

CDU – 551.502.3:004.512(811)

Ficha Catalográfica elaborada pela Bibliotecária/Documentalista:
Mariede Pimentel e Couto Diogo - CRB-11-354 - AM

AGRADECIMENTOS

Aos meus amigos, que compartilharam momentos de alegria, companheirismo e motivação nos períodos mais desafiadores desta jornada acadêmica.

Registro minha gratidão ao meu orientador Prof. Herbert Rocha, pela paciência, orientação, disponibilidade e contribuições valiosas para a realização deste trabalho.

Estendo também meus agradecimentos aos professores e colegas do curso, que contribuíram direta ou indiretamente para minha formação acadêmica e pessoal.

Por fim, agradeço a todos que, de alguma forma, participaram desta etapa da minha vida, seja com palavras de incentivo, apoio técnico ou simplesmente acreditando no meu potencial. Este trabalho é fruto não apenas de esforço individual, mas também da colaboração e carinho de todos vocês.

Boa Vista - RR
12 de novembro de 2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEP. DE CIENCIA DA COMPUTACAO

ATA Nº 21 / 2025 - DCC (11.05.09)

Nº do Protocolo: 23129.020054/2025-15

Boa Vista-RR, 17 de setembro de 2025.

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

No dia 15 de agosto de 2025, foi realizada a sessão pública de defesa do Trabalho de Conclusão de Curso II, intitulado "Sistema Integrado de Monitoramento Climático na Amazônia com Tecnologias Abertas", pelo aluno **PEDRO ALEPH GOMES DE SOUZA VASCONCELOS**, matrícula nº 2016007150, vinculado ao Departamento de Ciência da Computação (DCC/CCT/UFRR).

A sessão teve início às 10h30min, conduzida pelo Professor Herbert Oliveira Rocha, presidente da banca examinadora, composta pelos seguintes membros:

- Profa. Felipe Lobo
- Prof. Marcelle Urquiza
- Prof. Leandro Balico

Após a apresentação do trabalho, a banca examinadora procedeu com a arguição do candidato. Concluído esse processo às 11h50min, os membros da banca reuniram-se para deliberação e emitiram o seguinte parecer final sobre a apresentação e defesa oral:
✅ Aprovado
⬜ Reprovado
⬜ Aprovado com restrições

Nota final: 9,0 (nove)

Proclamado o resultado pelo presidente da banca examinadora, foram encerrados os trabalhos. Para constar, eu, Professor Herbert Oliveira Rocha, lavrei a presente ata, que assino juntamente com os demais membros da banca examinadora.

(Assinado digitalmente em 19/09/2025 12:10)
FELIPE LEITE LOBO
PROFESSOR DO MAGISTÉRIO SUPERIOR
DCC (11.05.09)
Matrícula: 1988091

(Assinado digitalmente em 17/09/2025 11:31)
HERBERT OLIVEIRA ROCHA
PROFESSOR DO MAGISTÉRIO SUPERIOR
DCC (11.05.09)
Matrícula: 2227352

(Assinado digitalmente em 18/09/2025 16:30)
LEANDRO NELINHO BALICO
PROFESSOR DO MAGISTÉRIO SUPERIOR
DCC (11.05.09)
Matrícula: 2689697

(Assinado digitalmente em 17/09/2025 21:08)
MARCELLE ALENCAR URQUIZA
PROFESSOR DO MAGISTÉRIO SUPERIOR
DCC (11.05.09)
Matrícula: 1491557

Visualize o documento original em **https://sipac.ufrr.br/public/documentos/index.jsp** informando seu número: **21**, ano: **2025**, tipo: **ATA**, data de emissão: **17/09/2025** e o código de verificação: **8e45b9e98e**

SUMÁRIO

Sistema Integrado de Monitoramento Climático na Amazônia com Tecnologias Abertas

Pedro Aleph Gomes de Souza Vasconcelos

Curso de Bacharelado em Ciência da Computação
Departamento de Ciência da Computação
Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brasil

pedro.aleph.vasconcelos@gmail.com

Resumo. *O objetivo deste estudo é desenvolver uma solução web para visualização de dados de estações meteorológicas, voltada a profissionais da produção agropecuária. A aplicação integra uma API RESTful para comunicação com o servidor e uma interface interativa para exibição das informações, oferecendo funcionalidades como painéis de visualização, exportação de dados em formatos CSV e SVG, e filtragem de períodos dentro de um intervalo anual. A avaliação experimental indicou 76% de qualidade em usabilidade, tempos de resposta em torno de dois segundos nas requisições, desempenho da interface satisfatório e baixo risco nos problemas identificados no código-fonte. Conclui-se que a solução apresenta resultados promissores, embora ainda restritos ao ambiente de desenvolvimento, sendo passível de melhorias e da inclusão de novas funcionalidades conforme as necessidades dos usuários.*

Abstract. *The objective of this study is to develop a web-based solution for visualizing weather station data, specifically designed for agricultural production professionals. The application integrates a RESTful API for server communication and an interactive interface for displaying information, offering features such as visualization panels, data export in CSV and SVG formats, and filtering by period within a year. The experimental evaluation yielded a usability quality of 76%, response times of around two seconds for requests, satisfactory interface performance, and a low risk of problems identified in the source code. The conclusion is that the solution presents promising results, although it is still limited to the development environment, and is subject to further improvements and the inclusion of new features according to user needs.*

Palavras-chave: Estações meteorológicas, API RESTful, Interface de Usuário.

1. Introdução

A produção agropecuária é uma atividade crucial para a sustentabilidade global. Sendo a principal fonte de renda para milhões de pessoas e é estratégica para o combate à fome e o desenvolvimento socioeconômico (RITCHIE; ROSADO; ROSER, 2023). Apesar dos avanços tecnológicos observados nas últimas décadas, a produção agropecuária permanece altamente vulnerável às variações do tempo e do clima. Nesse contexto, o acesso a informações meteorológicas precisas e atualizadas torna-se indispensável para os produtores, pois favorece o entendimento dos padrões climáticos e embasa a tomada de decisões em tempo real, promovendo uma agricultura mais eficiente, resiliente e sustentável (GOMMES et al., 2010).

Organizações como World Meteorological Organization (WMO) e Agriculture Organization of the United Nations (FAO) promovem iniciativas para fortalecer serviços agrometeorológicos e ampliar o acesso a dados climáticos confiáveis, essenciais para apoiar o setor agropecuário diante das variações climáticas (World Meteorological Organization, 2024; Food and Agriculture Organization of the United Nations, 2019). No Brasil, o Ministério da Ciência, Tecnologia e Inovação (MCTI) desenvolve estudos e modelagens climáticas que subsidiam estratégias de adaptação e planejamento agrícola (Brasil. Ministério da Ciência, Tecnologia e Inovação, 2016). Esses esforços evidenciam a importância dos dados meteorológicos como base para decisões mais precisas e sustentáveis no campo.

O interesse por soluções voltadas à visualização de dados meteorológicos tem sido destacado em diversos estudos, tais como: Em Brendel, Dymond e Aguilar (2019), por exemplo, é apresentado o *SHARKS*, um aplicativo que integra ferramentas de análise para investigar processos hidrológicos em bacias hidrográficas urbanas na cidade de Roanoke, Virgínia (EUA). Já em Koch (2018), por meio da biblioteca *Kepler.gl*, desenvolvida pela UBER, é implementada uma aplicação de visualização de dados meteorológicos em 3D, permitindo uma análise heterogênea ao correlacionar múltiplas variáveis. Por sua vez, Soler-Méndez et al. (2022) propõe uma aplicação web voltada ao cálculo da evapotranspiração, com o objetivo de otimizar o uso dos recursos hídricos a partir da aquisição e análise de dados climáticos. Esses trabalhos evidenciam que o uso de interfaces web representa uma solução eficaz para a visualização e interpretação de dados meteorológicos complexos.

Neste sentido, o objetivo deste trabalho é propor e implementar uma aplicação web capaz de integrar e exibir de forma interativa dados meteorológicos, buscando superar barreiras técnicas e otimizar a comunicação dessas informações para produtores e profissionais da agropecuária. A proposição e implementação de uma aplicação web para integração e visualização interativa de dados meteorológicos envolve superar uma série de desafios técnicos e operacionais. Um dos principais refere-se à coleta e padronização de dados, já que as informações podem ser provenientes de diferentes fontes, com formatos, resoluções temporais e padrões de qualidade distintos. Além disso, o processamento em tempo real de grandes volumes de dados exige arquiteturas eficientes e escaláveis, capazes de lidar com a variabilidade e a alta taxa de atualização típica de informações climáticas.

Diante do contexto apresentado, o problema considerado neste trabalho pode ser expresso na seguinte questão: **Como aplicações web de visualização interativa podem enfrentar desafios computacionais relacionados ao processamento, integração e exibição em tempo real de dados meteorológicos?** Assim, ampliando o acesso à informação por técnicos, produtores e pesquisadores do setor agropecuário.

Organização do Trabalho. As demais seções restantes são organizados da seguinte forma: Nos **Fundamentos Teóricos**, são apresentados os conceitos abordados neste trabalho, especificamente: Estações meteorológicas, Arquitetura de Software, Interface e Visualização de Dados. Nos **Trabalhos Correlatos**, são analisados os trabalhos correlatos a solução proposta. Na **Método da Solução Proposta**, é descrito as etapas de execução do método da solução proposta para uma visualização de dados de estações meteorológicas. Na **Avaliação Experimental**, são apresentados o planejamento, o projeto, a execução e a discussão dos resultados obtidos na avaliação da solução proposta. E por fim nas **Considerações Finais**, apresenta-se as considerações finais e análise das atividades desenvolvidas.

2. Fundamentos Teóricos

Nesta seção serão apresentados definições de conceitos utilizados no desenvolvimento deste trabalho, que são práticas que vão desde desenvolvimento da aplicação até a entrega ao usuário final.

2.1. Estações Meteorológicas com IoT

As estações meteorológicas são compostas por um conjunto de instrumentos e sensores instalados de forma integrada, com o objetivo de medir variáveis climáticas como precipitação, temperatura do ar, velocidade do vento e umidade do solo, entre outras (PERAZZI et al., 2021), e podem ser classificadas em dois tipos: convencionais e automáticas. As estações convencionais requerem intervenção humana direta para a coleta dos dados, geralmente realizada pelo menos uma vez ao dia. Por outro lado, as estações automáticas são conectadas a uma rede de transmissão de dados, permitindo o envio automático das medições em intervalos regulares, geralmente a cada hora (JARRAUD, 2008).

O avanço acelerado das tecnologias da informação e comunicação nos últimos anos viabilizou a integração de diversos serviços capazes de se comunicar tanto em redes locais quanto entre diferentes redes, consolidando o ecossistema conhecido como Internet das Coisas (*Internet of Things – IoT*) (PATEL; SHANGKUAN; THOMAS, 2017). Nesse contexto, destaca-se a aplicação da IoT em estações meteorológicas automatizadas de baixo custo, uma abordagem que emprega sensores sem fio para a medição de variáveis meteorológicas e cuja operação está interligada por meio de uma rede comum (IOANNOU et al., 2021). As informações coletadas por esses dispositivos são armazenadas em bancos de dados, permitindo análises em tempo real ou em momentos posteriores, ampliando a capacidade de monitoramento e tomada de decisão baseada em dados ambientais.

Os dados gerados por essas estações, especialmente as automáticas integradas por IoT, constituem uma fonte confiável e contínua de informações meteorológicas que podem ser consumidas por sistemas de análise e visualização, como o desenvolvido neste trabalho.

2.2. Arquitetura de Software

O princípio fundamental de toda arquitetura de software é que um sistema deve ser projetado com o propósito de atender às metas e objetivos de uma organização. Nesse contexto, a arquitetura atua como uma ponte entre essas metas e o sistema de software que as materializa (BASS; CLEMENTS; KAZMAN, 2021). Neste trabalho, propõe-se o desenvolvimento de um *dashboard* interativo que permite aos usuários visualizar, em tempo real, os dados provenientes de uma estação meteorológica selecionada. Para viabilizar essa funcionalidade, adotou-se uma arquitetura baseada em microserviços, um estilo arquitetural que evolui a partir da abordagem orientada a serviços - *SOA* (DAVE; PATEL; KESHRI, 2021). A comunicação entre os componentes do sistema é realizada por meio de uma *API RESTful*. Embora RESTful não constitua, por si só, uma arquitetura completa, ela segue os princípios do estilo arquitetural REST, conforme proposto por Fielding[1].

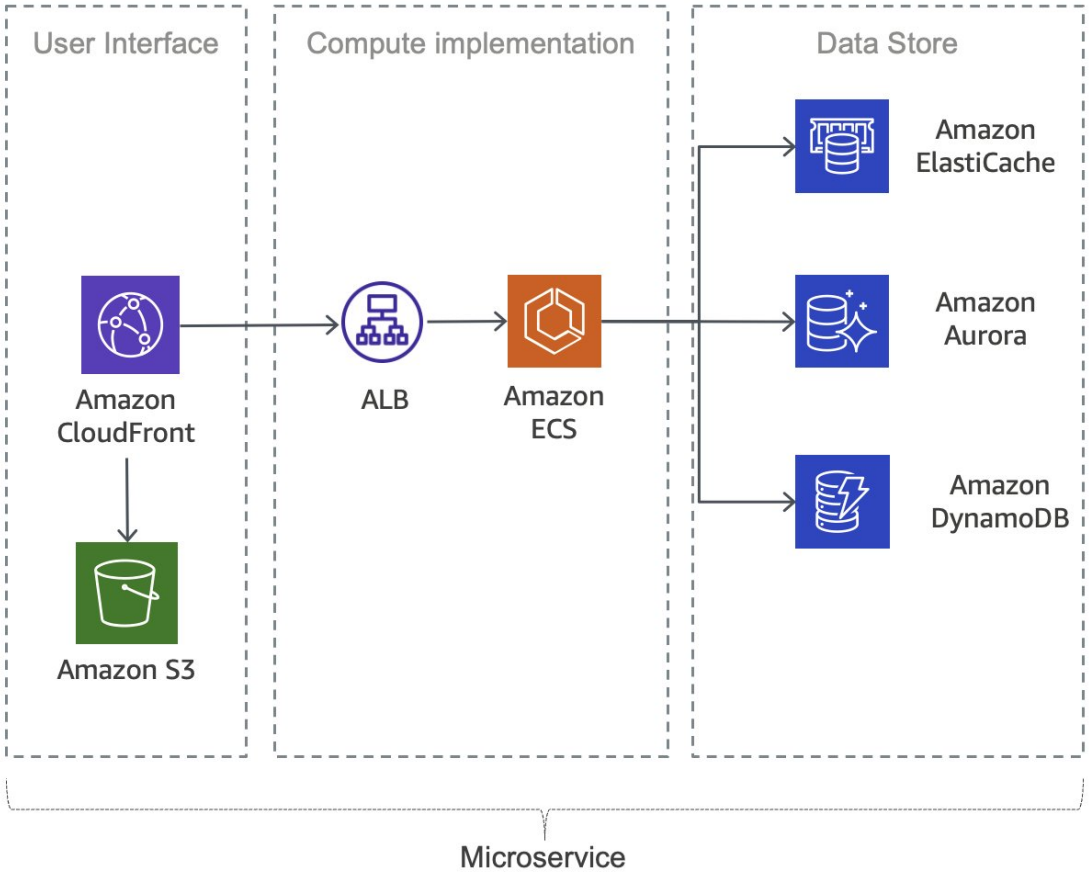
2.2.1. Microsserviços

Microsserviços são componentes independentes e coesos de software que possuem uma única responsabilidade e podem ser implantados, testados e escalados de forma autônoma. Uma

[1] ⟨https://ics.uci.edu/~fielding/pubs/dissertation/top.htm⟩

aplicação baseada nessa arquitetura é composta por diversos microsserviços que se comunicam entre si, colaborando para o funcionamento do sistema como um todo. Cada serviço pode ser mantido, dimensionado ou mesmo substituído de maneira isolada, sem impactar os demais (THÖNES, 2015). Essas características conferem maior flexibilidade e agilidade ao desenvolvimento e à manutenção do software (JAMSHIDI et al., 2018).

Figura 1. Aplicação típica de microsserviços na AWS.



Fonte: Services (2025).

A Figura 1 apresenta uma arquitetura de microsserviços típica em ambientes AWS, conforme descrita no whitepaper da AWS sobre o tema. Nela, funcionalidades são segmentadas em verticais alinhadas ao domínio do negócio, abandonando a lógica por camadas tradicionais (SERVICES, 2025). Essa arquitetura promove modularidade, escalabilidade independente e implantação autônoma de serviços, além de facilitar atualizações sem impacto global no sistema.

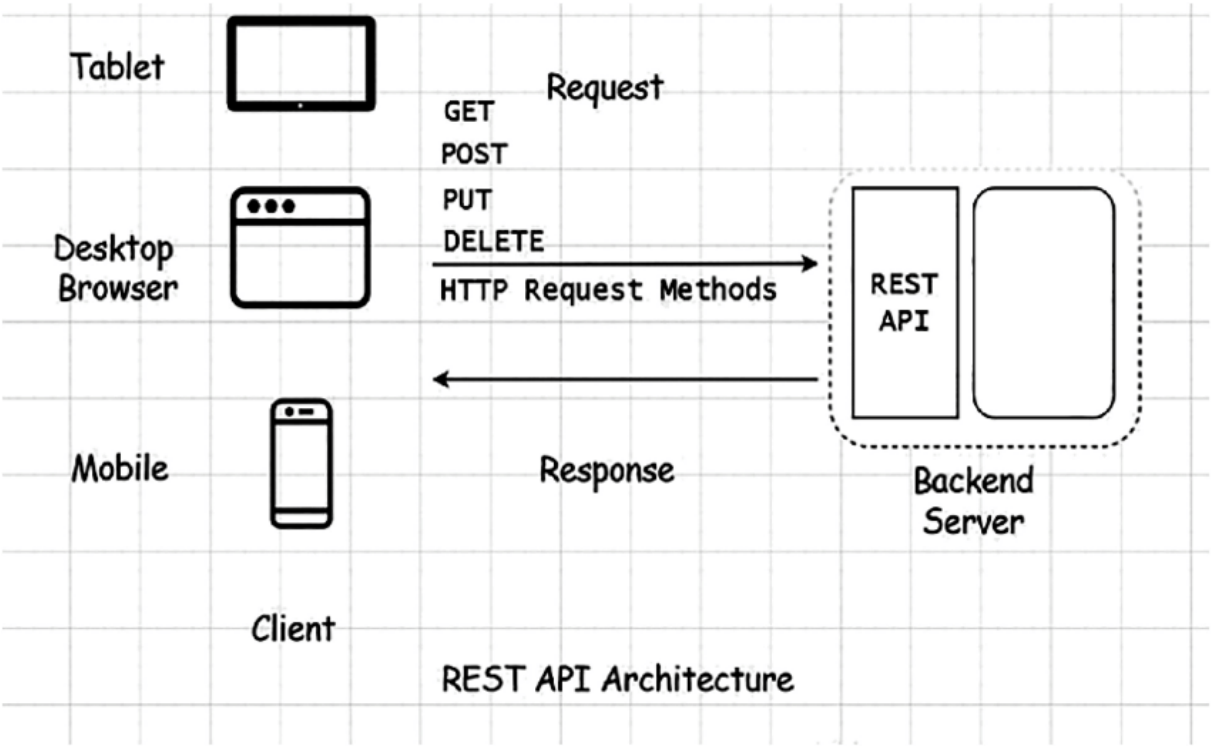
2.2.2. API RESTful

APIs RESTful são compostas por *endpoints*, que representam implementações concretas de funcionalidades específicas de processos de negócio. Essas APIs são, em sua maioria, acessadas por meio do protocolo HTTP, utilizando os verbos padrão definidos pela arquitetura REST, tais como `GET`, `POST`, `PUT` e `DELETE` (EHSAN et al., 2022). As respostas seguem padrões que combinam um **código de status HTTP** (por exemplo, `200 OK`, `201 Created`, *404 Not*

Found) e um corpo estruturado, geralmente em formato *JSON*, contendo os dados solicitados ou mensagens de erro claras, assegurando consistência e previsibilidade na comunicação cliente-servidor (AKINSOLA, 2018).

A Figura 2 ilustra uma API que recebe requisições originadas de diferentes dispositivos e, dependendo do tipo de requisição e do método HTTP utilizado, processa e retorna uma resposta adequada ao cliente.

Figura 2. Exemplo de arquitetura de API RESTful.



Fonte: Mitropoulos et al. (2021).

2.3. Interface e Visualização de Dados

A interface de usuário é a parte de um sistema interativo de computador que se comunica com o usuário. Seu design inclui tudo o que é visível ao usuário e se estende profundamente ao design do sistema interativo como um todo. Devido a isso, uma boa interface de usuário deve ser parte do processo de design do sistema desde o início (JACOB, 2003). Junto a isso, há a necessidade de exibir grandes quantidades de dados de forma acessível e compreensível. Nesse cenário, tem-se a visualização de dados, que se preocupa com o design, desenvolvimento e aplicação de representações gráficas geradas por computador. Ela fornece uma representação eficaz de dados originados de diferentes fontes, permitindo que os tomadores de decisão compreendam as análises com mais clareza e tomem decisões com maior eficiência (SADIKU et al., 2016). Para o desenvolvimento deste projeto, tanto a interface de usuário quanto a visualização de dados são conceitos cruciais, pois influenciam diretamente na usabilidade, na eficiência do sistema e na compreensão das informações apresentadas.

2.3.1. Experiência de usuário (UX)

Experiência de usuário ou UX, descreve os sentimentos de um usuário ao usar um produto, seja antes, durante ou após o uso. Esses sentimentos são governados por diversos fatores, como por exemplo, a expectativas do usuário, o ambiente ou o quão satisfatório os problemas do cliente foram resolvidos pelo produto (ROTO; KAASINEN, 2008).

O design responsivo para a Web é uma técnica utilizada para manter a coesão visual em diferentes tamanhos de navegadores, assim influenciado na UX. Um design é considerado responsivo se utilizar *grid* responsivas (*grid* ou *flexbox* em *CSS*), imagens flexíveis e media queries, indicadores de quais estilos devem ser aplicados para cada resolução (LESTARI; HARDIANTO; HIDAYANTO, 2014). Design responsivo contribui para que não haja perda de dados entre resoluções, o que influencia na boa experiência do usuário com o produto em diversos aparelhos, de *desktop* ao *mobile*.

2.4. Tecnologias Web para Visualização Geoespacial

A visualização geoespacial na web, ou ***web mapping***, é o processo de criar, analisar e compartilhar representações visuais de dados geoespaciais por meio de mapas acessíveis via navegador[2]. Esses sistemas permitem que usuários interajam com camadas geográficas, façam zoom, visualizem informações contextuais e realizem navegação dinâmica, usualmente sem necessidade de software especializado. Tecnologias como *WebGL*[3], *Leaflet*[4] e *OpenLayers*[5] exemplificam abordagens amplamente utilizadas, oferecendo recursos como renderização acelerada por GPU, manipulação de múltiplas camadas e suporte a diferentes projeções cartográficas.

2.5. Escalabilidade e Performance de Sistemas

O desempenho (***performance***) de um sistema refere-se à sua capacidade de processar requisições de forma eficiente, equilibrando dois aspectos fundamentais: **latência**, ou o atraso na resposta, tempo entre o envio da solicitação e o recebimento da resposta, e ***throughput***, isto é, o volume de dados ou número de solicitações que podem ser processados por unidade de tempo[6].

A escalabilidade complementa o desempenho, definindo a capacidade de um sistema lidar com uma demanda crescente, seja por aumento de carga ou volume de dados, sem degradar sua performance (BONDI, 2000). Bondi (2000) distingue dois tipos principais:

- ***Load scalability***: manter o desempenho sob demanda crescente;
- ***Structural scalability***: permitir crescimento (como número de usuários) sem reformular a arquitetura.

A sinergia entre performance e escalabilidade é crucial: um sistema que não escala corretamente pode perder eficiência sob cargas elevadas, exigindo ajustes ou redistribuições na arquitetura.

3. Trabalhos Relacionados

Nesta seção são apresentados trabalhos que se relacionam com este estudo, tais como, construção de uma interface web para recuperação e visualização de dados climáticos e técnicas de otimização para o framework utilizado.

[2] ⟨https://www.precisely.com/glossary/web-mapping⟩
[3] ⟨https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API⟩
[4] ⟨https://leafletjs.com⟩
[5] ⟨https://openlayers.org⟩
[6] ⟨https://aws.amazon.com/pt/compare/the-difference-between-throughput-and-latency⟩

3.1. An interactive web app for retrieval, visualization, and analysis of hydrologic and meteorological time series data

O trabalho de Brendel, Dymond e Aguilar (2019) apresenta o SHARKS (*Stream Hydrology And Rainfall Knowledge System*), uma aplicação web desenvolvida em R com o framework Shiny[7] para visualização e análise de séries temporais meteorológicas e hidrológicas. Voltado a usuários com conhecimentos básicos em hidrologia, o sistema surgiu da necessidade, identificada pela equipe da cidade de Roanoke (Virgínia, EUA) e seus desenvolvedores, de monitorar e interpretar variáveis como precipitação (profundidade, intensidade, recorrência) e fluxo (profundidade, descarga), além de indicadores hidrológicos mais complexos, como volumes de escoamento e coeficientes associados.

A aplicação adota a arquitetura reativa do Shiny, composta por um módulo de interface de usuário e uma função de servidor. As entradas do usuário (intervalos de data, seleções, interruptores) disparam cálculos no servidor, que retornam mapas, gráficos e tabelas atualizados automaticamente. Alguns recursos são executados sem intervenção, como separação de hidrogramas e cálculo de volumes e intensidades máximas de precipitação, enquanto outros dependem de ações do usuário, como estimar intervalos de recorrência e gerar visualizações interativas.

Implementado para a cidade de Roanoke, o SHARKS demonstrou eficácia como ferramenta para investigação de processos hidrológicos urbanos e apresenta potencial de adaptação para diferentes contextos e instituições. O sistema está disponível em ⟨https://bigbadcrad.shinyapps.io/SHARKS⟩, com código-fonte aberto sob licença MIT em ⟨https://github.com/bigbadcrad/SHARKS⟩.

3.2. Using Kepler.gl to visualize weather data

Koch (2018) apresenta uma solução para visualização de dados meteorológicos com grande volume de dados e muitas variáveis, para correlacionar essas diferentes variáveis, argumentando que a visualização em 3D é a mais apropriada. Por isso a aplicação foi desenvolvida utiliza a biblioteca *Kepler.gl*[8](desenvolvida pela Uber) é uma ferramenta de código aberto de análise geoespacial. Dado o conjunto de dados com informações de temperatura, precipitação, radiação solar e umidade. A aplicação permite correlacionar algumas dessas variáveis, por exemplo, umidade com temperatura; temperatura com precipitação; e temperatura com radiação solar. A interface do aplicativo de Koch (2018) exibe o conjunto de dados em 3D usando visualizações em espaço de tempo, com o objetivo de explorar mais detalhes da informação. A Figura 3 mostra um uso da aplicação, dado a distribuição de algumas variáveis, em um dado tempo. Do lado esquerdo da Figura 3, a temperatura e radiação solar; do lado direito, umidade e precipitação. Na parte de baixo da Figura 3 está o controle de espaço de tempo, possibilitando a visualização da mudança dos dados durante o período de análise.

O caso de estudo do trabalho de Koch (2018), utilizou dados providos do Instituto Nacional de Pesquisas Espaciais do Brasil (INPE). Estes dados possuem informações meteorológicas considerando o período entre 14 de novembro a 22 de novembro de 2018. Neste estudo procurava se temperaturas e umidade que seriam ideais para pulverizar um pomar de maçãs. A temperatura deve estar abaixo de 30 graus celsius, a umidade acima de 55 por cento e não pode estar chovendo. Os dados então foram filtrados para uma cidade específica considerando os períodos de 15 de novembro a 18 de novembro. Usando a aplicação, rodando pelos espaços

[7] ⟨https://shiny.rstudio.com⟩
[8] ⟨https://kepler.gl⟩

Figura 3. Visualizações em 3D.

Fonte: Koch (2018).

de tempo, foram encontrados os períodos ideais para aplicar os pesticidas. O melhor período é cedo pelas manhãs por que na tarde a umidade começa a diminuir e a radiação solar aumenta.

3.3. Agroclimatic Evolution web application as a powerful solution for managing climate data

No trabalho de Soler-Méndez et al. (2022) é proposto uma aplicação web nomeada de *Agro-Climatic Evolution*[9] que ao receber dados provindos de diferentes estações agrometeorológicas, permite a consulta em tempo real a cada hora e diariamente da evapotranspiração (ET) de distintas estações. Esta aplicação é responsiva para se adaptar a diferentes telas de dispositivos (*mobile*, *tablet* e *desktop*). O objetivo desta aplicação é promover gerenciamento de irrigação por fazer melhor uso de água e atendendo as necessidades das plantações por coleta e análise desses dados. O ETo (evapotranspiração de referência) calculado neste projeto, é o valor que expressa a capacidade de evapotranspiração na atmosfera em dada localização e tempo do ano sem levar em conta as características da cultura ou tipo de solo, sua grande vantagem é estimar os requerimentos de água apenas dependendo de parâmetros climáticos para obter o seu valor. Para calcular o valor do ETo, é utilizado o FAO Penman–Monteith (ZOTARELLI et al., 2010) que é um método padronizado que determina o valor de ETo usando parâmetros climáticos.

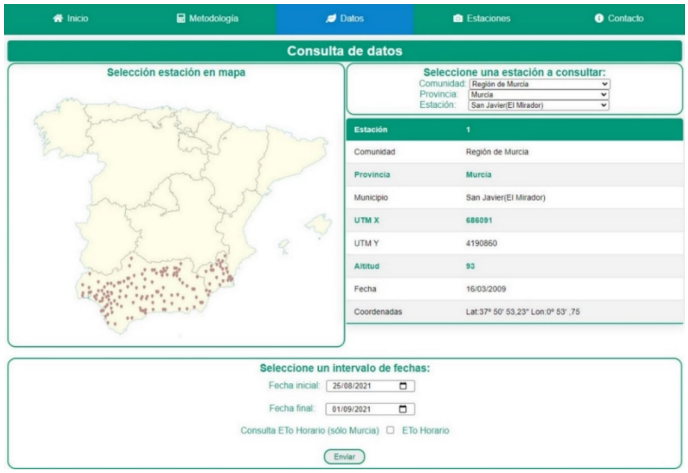
A interface de usuário da aplicação de Soler-Méndez et al. (2022) é dividida em abas. A aba inicial, mostra as informações sobre a aplicação; a segunda aba descreve o método de cálculo do valor de ETo; a terceira aba é onde os dados das estações podem ser consultados; a quarta aba é uma galeria de fotos das estações; a última aba fornece informações de contato em detalhes. Na aba de consulta de dados, a estação pode ser selecionada pelo mapa interativo na esquerda ou pelo menu de seleções a direita conforme a figura 4. Adicionalmente, podemos observar as opções de consulta do valor de ETo, com entradas de data inicial e final, e horários (se disponível na estação) quando estiver marcado na caixa de seleção para ser a cada hora.

3.4. Performance Optimization Techniques for ReactJS

No trabalho de Javeed (2019), propõem algumas práticas para lidar com problemas de performance de aplicações web em *React*[10]. Apesar de React ser capaz de resolver questões de

[9] ⟨http://josemiguel.myqnapcloud.com:49169/AgroClimatic-Evolution⟩
[10] ⟨http://react.dev⟩

Figura 4. Aba de acesso a área onde os dados podem ser consultados.



Fonte: Soler-Méndez et al. (2022).

performance comuns em aplicações web, ele sozinho ainda pode ter a performance degradada por designs complexos de aplicações no qual tem que lidar com muitos processamento de dados. Os problemas comuns destacados, tais como, renderização de componente, atraso na aplicação devido a execução de cálculos em segundo plano e atraso devido ao processamento de um grande grande conjunto de dados em um único trecho. React utiliza o virtual *DOM*, que decide se um componente deve ser recarregado ou não baseado no estado atual do componente e as mudanças que ocorrem, isto previne a aplicação de re-renderizar desnecessariamente. Um componente é uma peça da interface de usuário com sua própria lógica e aparência, pode ser pequeno um elemento como um botão, ou largo como uma página inteira, bem como composto por vários componentes. Na lógica do componente, os estados e as propriedades são os dois parâmetros que determinam quando o componente deve re-renderizar na aplicação. Quando há uma mudança no componente, este por exemplo recebendo uma propriedade passada por outro componente, o React DOM compara os novos valores com os anteriores e re-renderiza somente quando há diferença entre os estados.

Contudo em aplicações mais complexas, com componentes compostos por vários elementos e muitos parâmetros, muitas re-renderização causam sérios problemas de performance. Dessa forma, é necessário tomar medidas adicionais para renderizar os componentes somente quando requerido. Algumas dessas medidas são: **Reduzir o número de variáveis de estados e propriedades**, assim as chances de renderizações desnecessárias são reduzidas. Evitar frequentes e desnecessárias mudanças dos estados e as propriedades também ajuda. o Estado atualiza somente se tiver um impacto visual no usuário, ou o estado atualiza somente no fim. E **Dividir o componente principal em componentes independentes**, dado um componente que renderiza toda vez que ocorre uma mudança de estado que altera uma pequena parte do componente. Por exemplo, uma tabela que altera o conteúdo na linha quando um clique é feito nesta linha específica, isto é negligenciável se o tamanho dos dados for pequeno (ao menos 100), mas em tempo real, os dados podem ser enormes (em milhões), e assim, renderizar a toda a tabela a cada clique é ineficiente. Uma melhor forma é isolar cada linha da tabela em um componente independente e habilitar estes para escutarem e agirem em eventos independentes. Dessa forma, quando um evento de clique ocorre, somente uma linha particular é renderizada ao invés de toda tabela. O componente principal fica responsável somente por criar e gerenciar os filhos e não deve escutar todos os eventos destes a menos que o filho passe uma propriedade

para o parente, e o filho deve ser capaz de lidar com todos os eventos que ocorrem sem notificar outros componentes.

3.5. Correlações entre os trabalhos e a pesquisa

A Figura 5, apresenta uma tabela de comparação entre os trabalhos correlatos e a pesquisa deste papel. Os pontos abordados são os que a pesquisa tem como proposta de uso na solução e que foram identificados nos trabalhos. O trabalho de Javeed (2019) nesta pequisa é utilizado como base para aplicação das técnicas de otimização, os demais podem ser comparados com: soluções semelhantes, diferentes e recorrentes tecnologias utilizadas, designs responsivos, se as informações pode ser acompanhadas em tempo real enquanto são coletadas, e se é possível importar dados pelas interfaces.

Figura 5. Tabela de comparação dos trabalhos correlatos e a pesquisa.

Trabalho	Solução	Tecnologias identificadas	Responsivo	Dados em tempo real	Importação de dados
An interactive web app for retrieval, visualization, and analysis of hydrologic and meteorological time series data	Visualização e análise de dados meteorológicos	Shiny, R, HTML, CSS e JavaScript	Sim	Não	Sim
Using Kepler.gl to visualize weather data	Visualização e análise de dados meteorológicos	Kepler.gl, Mapbox, R, JSON, GeoJSON, React e WebGL	Não Identificado	Não	Sim
Agroclimatic Evolution web application as a powerful solution for managing climate data	Visualização e análise de dados meteorológicos	Node-RED, HTML, CSS, phpMyAdmin e JavaScript	Sim	Sim	Sim
Performance Optimization Techniques for ReactJS	Técnicas de otimização de software	React	Não se aplica	Não se aplica	Não se aplica
Sistema Integrado de Monitoramento Climático na Amazônia com Tecnologias Abertas	Visualização e análise de dados meteorológicos	Nodejs, React, Bootstrap, SASS, JSON, MongoDB, Mapbox, Recharts	Sim	Sim	Não

Fonte: Própria do autor.

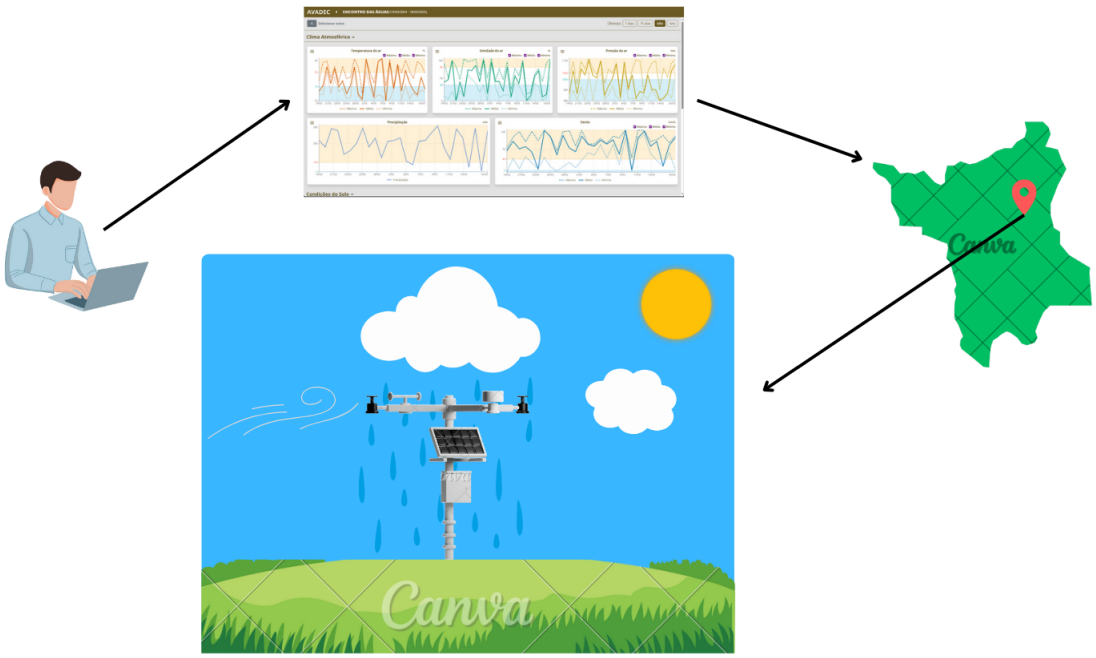
4. Solução Proposta

Esta seção apresenta a aplicação web para visualização e análise de dados meteorológicos, descrevendo a solução em termos de sua arquitetura IoT, o fluxo da solução, o projeto da aplicação e os componentes que a compõem.

4.1. Arquitetura

A Figura 6 apresenta uma visão geral do sistema de aquisição e recuperação de dados meteorológicos que fazem parte do desenvolvimento da solução proposta visa o desenvolvimento e análise de um aplicativo de visualização desses dados.

Figura 6. Sistema de aquisição e visualização de informações meteorológicas.



Fonte: Própria do autor.

No trabalho de Sales (2022), desenvolvido no âmbito da Embrapa Roraima[11], foi implementado um sistema baseado na arquitetura de IoT, abrangendo desde a coleta de dados meteorológicos até seu armazenamento e recuperação em nuvem. O foco principal da pesquisa esteve na transmissão dos dados e na rede de sensores sem fio utilizada para a coleta. Também foi desenvolvida uma aplicação simples para disponibilizar esses dados aos usuários.

Com base nesse desenvolvimento anterior, propõe-se, neste trabalho, uma solução voltada à visualização interativa e atualizada desses dados por meio de um dashboard. A solução consiste em uma aplicação web, na qual a interface de usuário se comunica com uma API RESTful responsável por acessar o banco de dados que armazena os dados coletados pelo servidor desenvolvido no trabalho anterior. A API RESTful então processa essas informações e as disponibiliza em um formato apropriado para a interface de usuário.

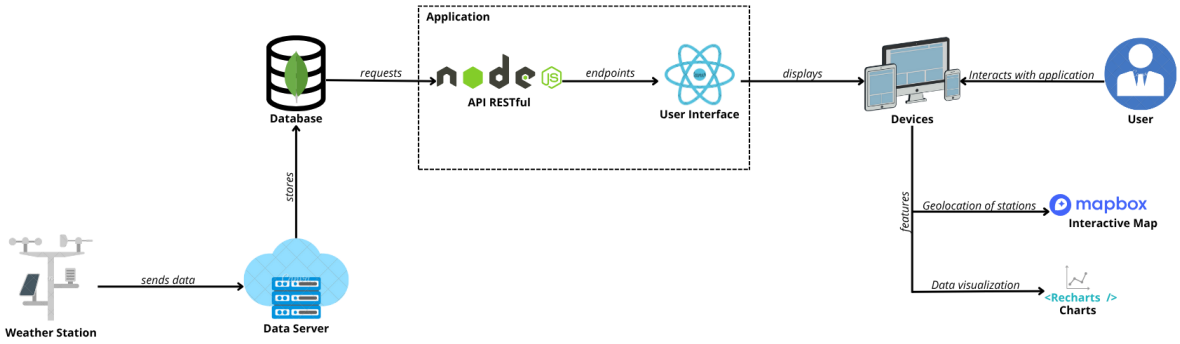
A Figura 7 apresenta o diagrama desse fluxo, juntamente com as ferramentas utilizadas. O fluxo inicia-se na estação meteorológica (*Weather Station*), responsável por coletar dados ambientais como temperatura, umidade, velocidade do vento e precipitação. Essas informações são transmitidas ao servidor de dados (*Data Server*), que realiza o pré-processamento e encaminha os registros para o banco de dados (*Database*), implementado com *MongoDB*, garantindo armazenamento estruturado e persistente. A API RESTful, desenvolvida em *Node.js*, atua como camada intermediária entre o banco de dados e a interface do usuário, oferecendo endpoints

[11] ⟨https://www.embrapa.br/roraima⟩

para consulta, inserção e atualização de informações meteorológicas. A interface do usuário (*User Interface*), construída com *React*, consome esses endpoints para exibir os dados em dispositivos diversos, como computadores, *tablets* e *smartphones*, assegurando responsividade e acessibilidade.

Recursos adicionais incluem a integração com o *Mapbox* para geolocalização das estações meteorológicas em um mapa interativo e o uso da biblioteca *Recharts* para visualização gráfica de séries temporais e análises estatísticas. Essa abordagem permite que produtores e profissionais da agropecuária acessem, interpretem e interajam com os dados em tempo real, favorecendo decisões mais precisas e eficientes no manejo agrícola. A arquitetura proposta adota tecnologias abertas e amplamente utilizadas que oferecem alta eficiência, flexibilidade e escalabilidade para aplicações web. O uso de soluções *open source* reduz custos de desenvolvimento e facilita a personalização, enquanto a integração com bibliotecas e serviços consolidados garante desempenho na coleta, processamento e visualização dos dados meteorológicos.

Figura 7. Diagrama de fluxo de obtenção e visualização dos dados.



Fonte: Própria do autor.

4.2. Aplicação

O objetivo da aplicação é fornecer uma visualização dos dados coletados de cada estação meteorológica disponível de acesso aos usuários. Para atingir esta finalidade, os requisitos funcionais básicos do sistema são:

RF01 Exibição de Estações Meteorológicas
Descrição: O sistema deve exibir a lista de estações meteorológicas disponíveis, permitindo que o usuário selecione uma estação específica para consulta de dados.
Prioridade: Alta
Critérios de Aceitação:
- A lista deve carregar todas as estações meteorológicas disponíveis no sistema.
- O usuário pode selecionar qualquer estação da lista.
- A seleção de uma estação atualiza o contexto para exibição dos dados relacionados.

Dependências: Nenhuma.

RF02 Apresentação de Dados por Período
Descrição: O sistema deve apresentar os dados da estação meteorológica selecionada, possibilitando a filtragem e visualização em diferentes períodos de tempo (por exemplo: diário, semanal, mensal).
Prioridade: Alta
Critérios de Aceitação:
- Os dados devem ser exibidos em gráficos ou tabelas claros e legíveis.
- O usuário pode selecionar o período desejado para visualização dos dados.
- A apresentação dos dados é atualizada corretamente conforme o período selecionado.

Dependências: Depende do RF01 para a seleção da estação meteorológica.

RF03 Salvamento de Informações de Gráficos
Descrição: O sistema deve permitir que o usuário salve, em formato de arquivo, as informações exibidas nos gráficos gerados a partir dos dados da estação meteorológica selecionada.
Prioridade: Média
Critérios de Aceitação:
- O usuário pode salvar as informações atuais do gráfico em formatos comuns (ex: PDF, PNG e SVG).
- O arquivo salvo deve conter os dados correspondentes ao período e estação selecionados.
- O processo de salvamento deve informar sucesso ou falha ao usuário.

Dependências: Depende do RF02 para apresentação dos dados a serem salvos.

Além dos requisitos de uso, a aplicação deve atender a requisitos não funcionais para proporcionar uma boa experiência para o usuário, a aplicação deve ser:

RNF01 Responsividade
Descrição: A aplicação deve ser responsiva, ou seja, adaptada para se ajustar adequadamente a diferentes tamanhos de tela, incluindo dispositivos desktop, tablet e mobile.
Critérios de Aceitação:
- A interface deve manter usabilidade e legibilidade em telas de diferentes resoluções e dimensões.

- Elementos da interface devem se reorganizar ou redimensionar automaticamente conforme o dispositivo utilizado.
- Testes em múltiplos dispositivos e navegadores devem comprovar a adequação visual e funcional.

RNF02 Otimização de Desempenho
Descrição: Os componentes da aplicação devem ser estruturados para evitar degradação de desempenho perceptível durante o uso.
Critérios de Aceitação:
- Tempo de carregamento inicial da aplicação deve ser inferior a 7 segundos em conexões padrão.
- A interface deve responder às ações do usuário em menos de 300 ms.
- O sistema não deve apresentar travamentos ou lentidão perceptível durante o uso normal.

RNF03 Qualidade da Interação
Descrição: A interface do sistema deve proporcionar fluidez, clareza e eficiência no uso pelo usuário.
Critérios de Aceitação:
- Os fluxos de interação devem ser intuitivos e facilmente compreendidos por usuários sem treinamento prévio.
- A interface deve evitar elementos confusos, garantir consistência visual e oferecer feedback claro para as ações do usuário.
- Deve ser realizada testes de usabilidade, demonstrando a eficiência e satisfação do usuário.

A aplicação é composta por dois microsserviços correspondentes à API RESTful e à interface de usuário, que atuam de forma integrada para atender aos requisitos descritos.

4.2.1. API RESTful

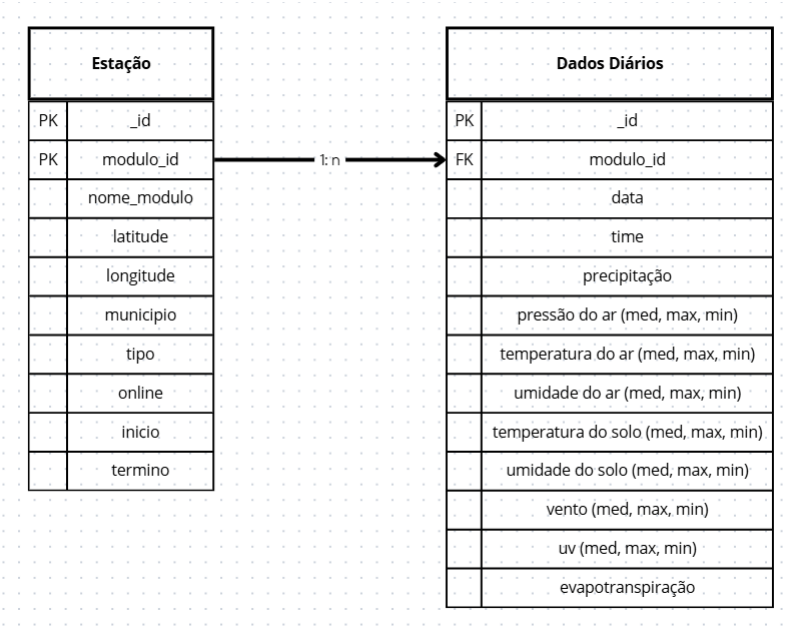
A API RESTful desta aplicação é responsável por recuperar os dados enviados pelas estações meteorológicas, armazenados em um banco de dados *MongoDB* hospedado na nuvem. Esses dados são processados pela API e disponibilizados à interface de usuário por meio de requisições HTTP. A API foi desenvolvida utilizando *Node.js*[12] e o framework *Express*[13].

A Figura 8 apresenta o diagrama UML das entidades que representam os dados já adaptados conforme os requisitos da interface. A entidade de Estações inclui campos essenciais como o nome da estação (**modulo_nome**) e suas coordenadas geográficas (**latitude** e **longitude**). Além disso, foram adicionados os campos de **início** e **término**, que correspondem ao período dos dados disponíveis para cada estação, essas informações são extraídas a partir dos dados diários.

Já a entidade de dados diários, além de conter as variáveis meteorológicas e a referência à estação correspondente(**modulo_id**), inclui um campo **time**, que representa a data no formato *timestamp*. Esse campo foi adicionado para facilitar a ordenação temporal dos dados nos gráficos. Também foi incluído o campo de **evapotranspiração**, cujo valor é calculado diretamente na API utilizando as variáveis meteorológicas por meio do método de Penman-Monteith[14], inspirado no trabalho de Soler-Méndez et al. (2022).

[12] ⟨https://nodejs.org⟩
[13] ⟨https://expressjs.com/pt-br⟩
[14] ⟨https://www.fao.org/4/X0490E/x0490e06.htm⟩

Figura 8. UML das entidades do sistema.



Fonte: Própria do autor.

A entidade de estação é acessível pelo endpoint GET **/stations**, que retorna uma lista de estações, ou, alternativamente, uma estação específica por seu **modulo_id**, através de GET **/stations/:id**. Já os dados diários são acessados com base no **modulo_id** pelo endpoint **/daily-data/:id**, que também aceita parâmetros opcionais: **period**, que pode receber o valor *monthly* para agrupar os resultados por mês, e **start** e **end**, que definem um intervalo de datas para filtrar os dados retornados.

Para fins de teste, o desenvolvimento não utilizou dados do banco de dados real. Em vez disso, foram gerados dados fictícios com 45 estações simuladas no estado de Roraima, cada uma contendo 365 registros diários, correspondentes a um ano. Os valores das variáveis meteorológicas foram criados por meio de um algoritmo simples de geração de valores pseudo-aleatórios.

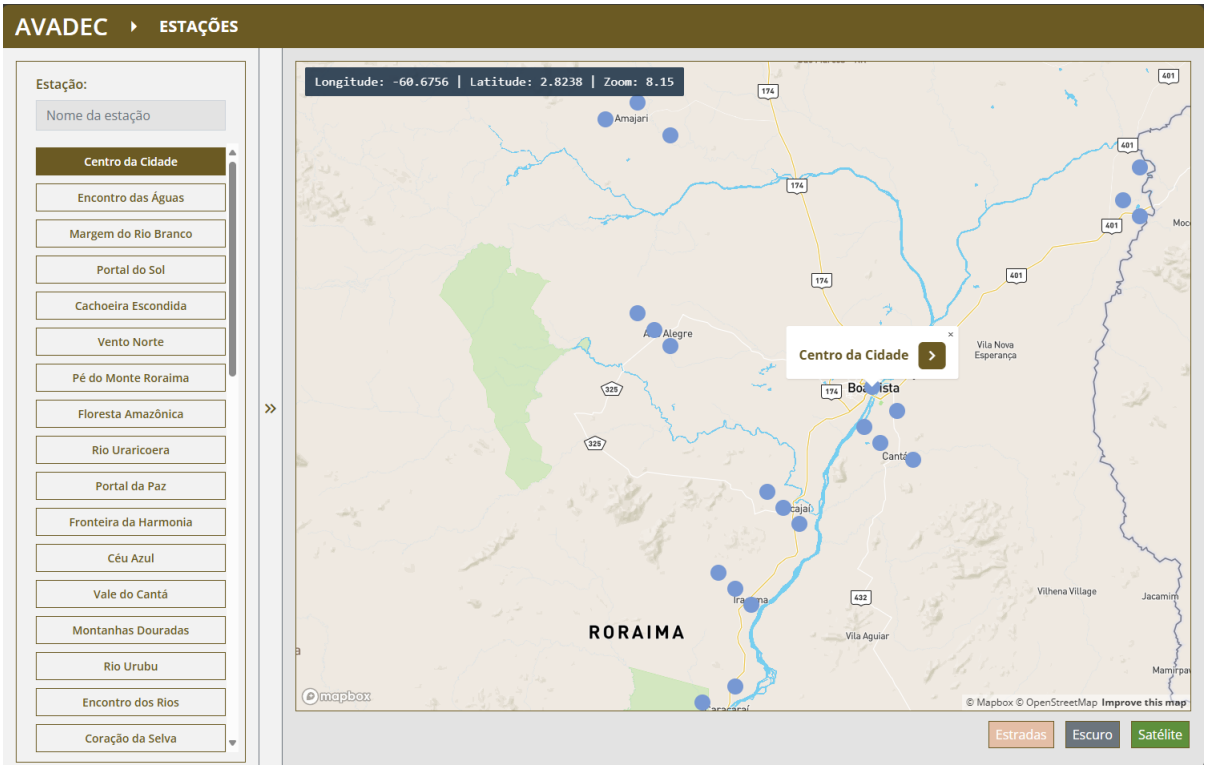
4.2.2. Interface de Usuário

A interface de usuário (UI) desta aplicação foi desenvolvida com a biblioteca *React*[15], utilizando *TypeScript*[16] para maior segurança e organização do código. Além disso, foram aplicadas técnicas de otimização de componentes descritas por Javeed (2019), com o objetivo de melhorar o desempenho e a experiência do usuário. O layout responsivo foi implementado com o pré-processador *SASS*[17] e a biblioteca *Bootstrap*[18], garantindo compatibilidade com diferentes tamanhos de tela.

A Figura 9 apresenta a página de seleção de estação, que exibe um mapa interativo cons-

[15] ⟨https://react.dev⟩
[16] ⟨https://www.typescriptlang.org⟩
[17] ⟨https://sass-lang.com⟩
[18] ⟨https://getbootstrap.com⟩

Figura 9. Página de seleção de estação.



Fonte: Própria do autor.

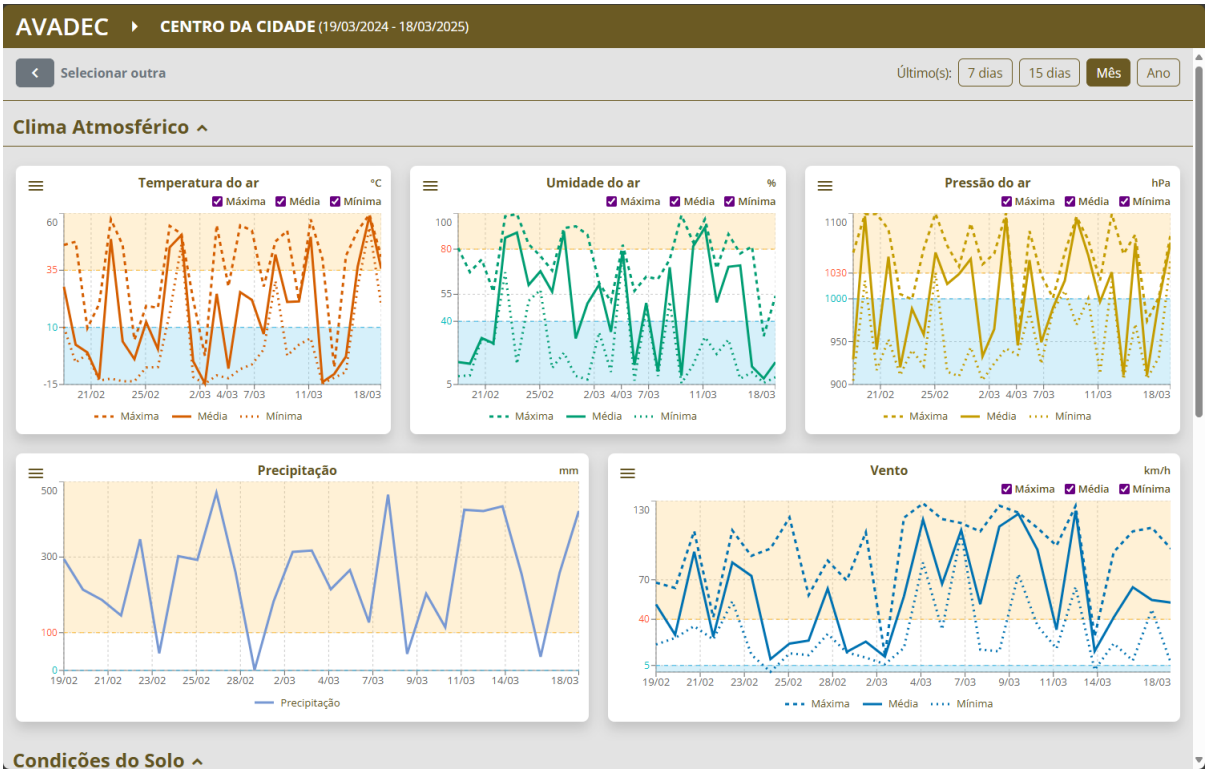
truído com *MapboxGL*[19], onde cada estação é representada por um marcador em sua respectiva coordenada geográfica. Ao clicar em um marcador, um *pop-up* exibe o nome da estação e um botão que redireciona para a página de visualização de seus dados. Alternativamente, o usuário pode utilizar uma lista lateral expansível, localizada à esquerda da tela, onde cada botão corresponde a uma estação. Acima da lista há um campo de busca que permite filtrar as estações pelo nome. Na parte inferior direita do mapa, três botões permitem alternar entre diferentes estilos de visualização: estrada, escuro e satélite.

A Figura 10 apresenta a página de uma estação selecionada, estruturada como um painel com gráficos que exibem a variação temporal das variáveis meteorológicas. Os gráficos foram implementados com a biblioteca *Recharts*[20]. No cabeçalho da página são exibidos o nome da estação e o período total de dados disponíveis. Um botão permite o retorno à página anterior, e ao lado há filtros temporais que permitem visualizar os dados mais recentes, com botões de “7 dias”, “15 dias”, “Mês” e “Ano”, que ajustam os dados exibidos conforme o intervalo correspondente ao período da estação.

As variáveis são organizadas em seções expansíveis conforme sua natureza: Clima Atmosférico (temperatura e umidade do ar, precipitação e vento), Condições do Solo (temperatura e umidade do solo) e Radiação (nível UV e evapotranspiração). Cada gráfico exibe linhas de valor máximo e mínimo esperado para a variável, com áreas sombreadas fora desses limites, inspirados nos mapas de calor de estações do Inmet[21]. Além disso, o usuário pode interagir com os gráficos de diversas formas: visualizar os valores em *pop-ups* ao passar o cursor, ativar

[19] ⟨https://docs.mapbox.com/mapbox-gl-js/guides⟩
[20] ⟨https://recharts.org⟩
[21] ⟨https://mapas.inmet.gov.br⟩

Figura 10. Página de dados da estação.



Fonte: Própria do autor.

ou desativar a exibição de valores máximos, médios e mínimos, bem como baixar os gráficos em formato SVG ou os dados em formato CSV.

A navegação entre as páginas é fluida, sem recarregamento completo, aproveitando as capacidades de uma aplicação de página única (SPA) desenvolvida com React. A interface também permite acesso direto via URL, como por exemplo, acessando **/stations/:id** para visualizar uma estação específica. Além disso, a última estação acessada é armazenada localmente no navegador. Assim, caso o usuário acesse novamente a aplicação pela URL base, ele será automaticamente redirecionado para a estação previamente selecionada.

5. Avaliação Experimental

Esta seção descreve o planejamento, execução, e resultados da avaliação experimental da solução proposta, realizada com base em diferentes conceitos atrelados aos seus objetivos, utilizando ferramentas consagradas e de amplo acesso.

5.1. Projeto da Avaliação Experimental

Esta seção descreve o planejamento da execução da avaliação experimental da solução proposta para a visualização de dados meteorológicos, voltada a profissionais da produção agropecuária. A avaliação é conduzida com base na norma ISO/IEC 25010[22], que define a qualidade de software segundo nove características, conforme ilustrado na Figura 11. Os conceitos avaliados neste trabalho são:

- **Usabilidade**[23]: Refere-se à facilidade e eficiência com que os usuários interagem com a interface. Na ISO/IEC 25010, está representado pela característica **Capacidade de**

[22] ⟨https://iso25000.com/index.php/en/iso-25000-standards/iso-25010⟩
[23] ⟨https://www.nngroup.com/articles/usability-101-introduction-to-usability⟩

Figura 11. Características da Qualidade de Software.

SOFTWARE PRODUCT QUALITY								
FUNCTIONAL SUITABILITY	PERFORMANCE EFFICIENCY	COMPATIBILITY	INTERACTION CAPABILITY	RELIABILITY	SECURITY	MAINTAINABILITY	FLEXIBILITY	SAFETY
FUNCTIONAL COMPLETENESS FUNCTIONAL CORRECTNESS FUNCTIONAL APPROPRIATENESS iso25000.com	TIME BEHAVIOUR RESOURCE UTILIZATION CAPACITY	CO-EXISTENCE INTEROPERABILITY	APPROPRIATENESS RECOGNIZABILITY LEARNABILITY OPERABILITY USER ERROR PROTECTION USER ENGAGEMENT INCLUSIVITY USER ASSISTANCE SELF-DESCRIPTIVENESS	FAULTLESSNESS AVAILABILITY FAULT TOLERANCE RECOVERABILITY	CONFIDENTIALITY INTEGRITY NON-REPUDIATION ACCOUNTABILITY AUTHENTICITY RESISTANCE	MODULARITY REUSABILITY ANALYSABILITY MODIFIABILITY TESTABILITY	ADAPTABILITY SCALABILITY INSTALLABILITY REPLACEABILITY	OPERATIONAL CONSTRAINT RISK IDENTIFICATION FAIL SAFE HAZARD WARNING SAFE INTEGRATION

Fonte: ISO/IEC 25010[22].

Interação, que envolve a comunicação entre usuário e sistema para a execução de tarefas.

- **Desempenho da Web**[24]: Refere-se ao tempo de carregamento e execução das páginas, medido tanto por métricas objetivas quanto pela percepção do usuário. Na ISO/IEC 25010, está relacionado à **Eficiência de Desempenho**, que avalia o cumprimento das funções dentro dos limites de tempo e uso de recursos especificados.
- **Qualidade do Código**[25]: Refere-se à clareza, confiabilidade e facilidade de manutenção do código. Na ISO/IEC 25010, corresponde à característica **Manutenibilidade**, que trata da facilidade de modificar o sistema para correções, melhorias ou adaptações.

Esses conceitos foram escolhidos por estarem alinhados aos objetivos da solução proposta: oferecer uma interface amigável ao usuário, eficiente em termos de desempenho e sustentável a longo prazo. Esses objetivos podem, por sua vez, ser expressos pelas seguintes questões de pesquisa (QP), que esta avaliação experimental busca investigar:

QP1 : **Usabilidade** - A interface permite uma interação intuitiva e eficaz para usuários com diferentes níveis de experiência?
QP2 : **Desempenho da Web** - A aplicação oferece tempos de resposta rápidos e escalabilidade adequada?
QP3 : **Qualidade do Código** - O código-fonte garante facilidade de manutenção e confiabilidade a longo prazo?

Visando responder as questões de pesquisa, divididas nos seus respectivos conceitos, foi utilizado as seguintes ferramentas:

- ***Heurix***[26]: Uma ferramenta de análise que permite avaliar *web sites*, reunindo as heurísticas essenciais, com base em *benchmarks* de usabilidade renomados, incluindo a heurística do grupo Nielsen Norman.
- ***Apache JMeter***[27]: Um software de código aberto desenvolvido em Java para testar o comportamento funcional e medir o desempenho de vários tipo de aplicações. Suas funcionalidades incluem testes em aplicações web.

[24] ⟨https://developer.mozilla.org/en-US/docs/Web/Performance⟩
[25] ⟨https://www.sonarsource.com/learn/code-quality⟩
[26] ⟨https://www.heurix.io⟩
[27] ⟨https://jmeter.apache.org/index.html⟩

- ***Google Lighthouse***[28]: Uma ferramenta de código aberto para melhorar a qualidade das páginas da Web, tendo auditorias de desempenho, acessibilidade, SEO e muito mais.
- ***SonarQube***[29]: Uma ferramenta de análise estática e revisão de código automatizada padrão do setor, projetada para detectar problemas de codificação.

A interface de usuário terá sua usabilidade avaliada com o Heurix e a performance de suas páginas analisada com o Google Lighthouse. Já a API RESTful será submetida a testes de desempenho das requisições em seus endpoints por meio do Apache JMeter. A qualidade do código-fonte, tanto da interface quanto da API, será verificada com o SonarQube. Os experimentos foram conduzidos nas seguintes especificações:

- Em um notebook Intel Core i7-11800H 2.30GHz, 16 GB RAM, NVIDIA GeForce RTX 3060, 512 GB NVMe SSD (ADATA IM2P33F3A), Windows 11 Pro 64-bit (Version 24H2, Build 26100.4061).
- Interface de usuário hospedada em: https://avadec.vercel.app na versão gratuita da plataforma Vercel[30].
- API RESTful hospedada em: https://avadec-backend.onrender.com. Antes dos testes serem conduzidos, a aplicação foi acessada manualmente para "acordar" o servidor gratuito da plataforma Render[31].
- O servidor do SonarQube foi executado localmente no WSL 2, com Ubuntu 20.04.6 LTS, em um contêiner *Docker*[32] utilizando sua imagem oficial na versão gratuita(*community*).

5.2. Avaliação e Discussão dos Resultados

Esta seção apresenta os resultados obtidos na avaliação experimental dos três conceitos abordados: Usabilidade, Desempenho da Web e Qualidade do Código. Para cada um, é explicado o funcionamento da ferramenta utilizada, seguido da exposição dos dados coletados e da análise dos resultados em relação aos objetivos da solução proposta.

5.2.1. Avaliação de Usabilidade

A ferramenta Heurix permite avaliações heurísticas por meio de um conjunto de perguntas guiadas passo a passo, gerando relatórios detalhados de forma rápida e acessível. Cada pergunta é fundamentada em uma heurística relevante de usabilidade e abrange aspecto essencial da experiência do usuário, como: Primeiras impressões; Navegação no site; Informações; Confiança e persuasão; Interação; Formulários; e Busca.

A interface de usuário da solução foi avaliada, com esta ferramenta, pelo próprio autor do trabalho. Além da finalidade de medir o quanto a interface satisfaz a qualidade em termos de usabilidade, buscou se obter insights para oportunidade melhoria. A Tabela 1 exibe os resultados obtidos em cada aspecto ao completada a avaliação:

No relatório gerado, a pontuação geral de 76% foi considerada ótima, indicando que todos os elementos de experiência de usuário estavam bem implementados e funcionando bem, mas que há espaço de melhoria. Os pontos de melhoria informados foram:

[28] ⟨https://developer.chrome.com/docs/lighthouse/overview⟩
[29] ⟨https://docs.sonarsource.com/sonarqube-community-build⟩
[30] ⟨https://vercel.com⟩
[31] ⟨https://render.com⟩
[32] ⟨https://www.docker.com⟩

Tabela 1. Resultado da avaliação de Usabilidade.

Aspecto	**Pontuação** %
Primeiras impressões	100%
Navegação no site	100%
Informações	95%
Confiança e persuasão	50%
Interação	100%
Formulários	66%
Busca	0%
Total	76%

1. Localização de conteúdo.
2. Informações adicionais sobre a empresa e referências a pessoas reais para aumentar a confiança.
3. Ajuda e assistência fornecidas.
4. Prova social persuasiva.
5. Atalhos para inserir informações.
6. Visibilidade da caixa de pesquisa.
7. Sugestões nos resultados da pesquisa.
8. Recursos de pesquisa avançada.
9. Capacidade de desfazer ou remover recursos de pesquisa avançada.

Analisando os pontos de melhoria mencionados, no contexto da solução, foram feitas algumas considerações:

- O item 1 se refere ao aspecto de informação. Todas informações da interface estão em português, mas os números estão em formato inglês, que é o formato padrão para o tipo de variável da linguagem do código-fonte. A melhoria que pode ser feita é adicionar uma funcionalidade de escolha de idioma, português e inglês, disponibilizando as informações e formato dos números em seus respectivos idiomas.
- Os itens 2, 3 e 4 se referem ao aspecto de confiança e persuasão. Não há qualquer informação de contato na interface da solução. A melhoria então seria uma funcionalidade para prover essas informações.
- O item 5 se refere ao aspecto de formulários. Mas na interface da solução não há páginas de preenchimento de informações com persistência, posso não há espaço para funcionalidades para esse aspecto. É aplicado apenas por que há uma entrada de filtro por nome de estação, mas por exemplo, uma funcionalidade de preenchimento automático não seria necessário, pois a lista de estações que pode ser filtrada é logo abaixo do filtro.
- Os itens 6, 7, 8 e 9 se referente ao aspecto de busca. O filtro por nome de estação, que por ter um objetivo especifico que é filtrar pelo nome da estação, não há oprtunidades para funcionalidades avançadas. Como melhoria, apenas pode ser adicionado um botão para limpar a entrada do filtro.

5.2.2. Avaliação de Desempenho da API RESTful

Nesta avaliação foi considerado o tempo de resposta da requisição nos seguintes endpoints e suas respectivas saídas(com os dados do banco de teste):

- **/stations**: retorna uma lista de estações, 45 objetos.

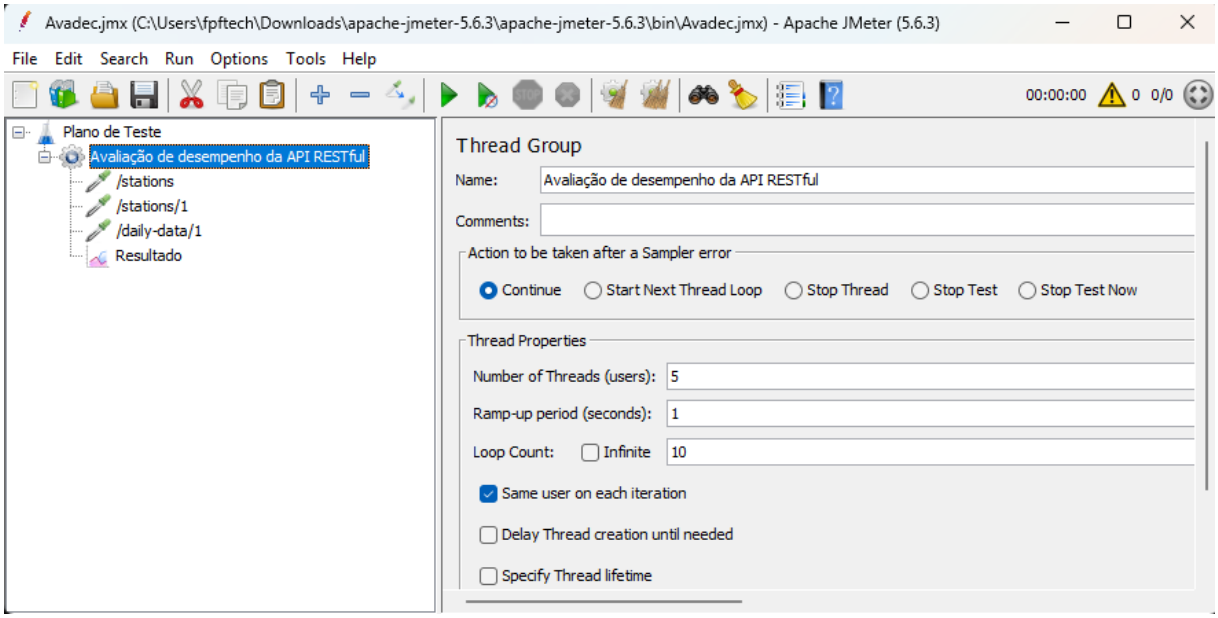
- **/stations/1**: retorna a estação com id igual a 1, um objeto.
- **/daily-data/1**: retorna uma lista de dados diários da estação com id igual a 1, 365 objetos.

Para medir o tempo de resposta, a avaliação se baseia nos três limites importantes segundo Nielsen[33], os quais seriam:

- **100 milissegundos**: Mínimo e ideal.
- **1 segundo**: Perceptível e aceitável.
- **10 segundos**: Tempo limite de atenção do usuário, evitável.

No Apache JMeter, um plano de teste Web é configurado definindo-se o **número de *threads*** (usuários virtuais), o ***ramp-up*** em segundos (tempo para iniciar todos os usuários) e o ***loop count*** (quantidade de vezes que cada usuário executa o teste). A Figura 12 mostra a configuração adotada.

Figura 12. Configuração do Plano de Teste Web.



Fonte: Própria do autor.

Considerando as limitações do servidor gratuito da Render, para evitar sobrecarga foi configurado um cenário com **5 threads**, **loop count** igual a **10** e **ramp-up de 1 segundo**. Assim, cada usuário virtual executou o teste 10 vezes, totalizando **50 requisições** por endpoint. O tempo de resposta esperado foi de até 2 segundos por requisição. A Tabela 2 apresenta os resultados, obtidos por meio do *listener* ***Aggregate Report***, para cada endpoint testado.

Tabela 2. Resultado da avaliação de desempenho da API RESTful.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
/stations	50	1507	1463	1872	2014	2135	1156	2135	0.000%	1.46955	16.23	0.20
/stations/1	50	498	484	503	510	888	456	888	0.000%	1.53563	1.02	0.21
/daily-data/1	50	1384	1298	1886	2016	2270	715	2270	0.000%	1.52304	505.80	0.21
TOTAL	150	1130	1252	1840	1949	2238	456	2270	0.000%	4.20805	482.25	0.57

Pela tabela, podemos dizer que em todos os endpoints, as requisições cumpre o tempo de espera estimado em ao menos 90% dos casos, e no resto, o tempo excedido ainda é mínimo. O que permite afirmar que, no cenário de teste apresentado, a API RESTful possui um desempenho aceitável.

[33] ⟨https://www.nngroup.com/articles/response-times-3-important-limits⟩

5.2.3. Avaliação de Desempenho da Interface de Usuário

Lighthouse foi executado utilizando sua extensão no navegador chrome[34], sua avaliação foi composta pelas auditiorias: desempenho, acessibilidade, práticas recomendadas e SEO. Foram testadas tanto a página de seleção de estação(/stations) quanto a página de dados da estação selecionada(/stations/1). Junto com os dois modos de tela: computador e celular.

A qualidade é considerada pelos intervalos das pontuações de métrica e desempenho nas seguintes formas:

- **Ruim**: De 0 a 49.
- **Precisa de melhoria**: De 50 a 89.
- **Bom**: De 90 a 100.

A Tabela 3 exibe os resultado obtidos da avaliação:

Tabela 3. Resultado da avaliação de desempenho da Interface de Usuário.

Rota	Tela	Desempenho	Acessibilidade	Práticas recomendadas	SEO
/stations	computador	53	90	100	91
/stations	celular	36	96	100	91
/stations/1	computador	71	80	100	91
/stations/1	celular	48	80	100	91

No resultado, pode se observar que a interface se saiu bem na maioria das auditorias, exceto pelo desempenho. Sendo considerada que precisa melhorar no modo computador, e ruim no modo celular. A maioria dos problemas apresentados são relacionados as bibliotecas MapboxGL e Recharts, descrevendo que há um gasto desnecessário com esses recursos. Então optimizar o uso dessas bibliotecas deve melhorar o desempenho da interface significativamente. A acessibilidade da página de estação selecionada também tem qualidade que pode melhorar, mas os problemas apontados, como contraste entre os elementos insuficiente, podem ser resolvidos facilmente. Por esse resultado obtido, que tem espaço para melhora mas não tão baixo a ponto ser considerado péssimo, pode se afirmar que a interface usuário tem qualidade razoável em termos de desempenho.

5.2.4. Avaliação de Qualidade do Código

SonarQube[35] analisa o código-fonte do software com base em três aspectos de qualidade, os quais também estão presentes na norma ISO/IEC 25010:

- **Segurança**: Refere-se à proteção do software contra ataques maliciosos e ao controle de acesso a informações por entidades não autorizadas.
- **Confiabilidade**: Refere-se à capacidade do software de manter seu desempenho esperado sob determinadas condições e durante um período específico.
- **Manutenibilidade**: Refere-se à facilidade e eficiência com que o software pode ser modificado para correções, melhorias ou adaptações.

Os problemas encontrados são classificados em níveis de severidade, que variam desde riscos apenas **informativos**, nos quais não se espera impacto algum; passando por riscos de impacto **baixo**, **médio** e **alto**; até os riscos **bloqueadores**, que podem gerar consequências graves

[34] ⟨https://chromewebstore.google.com/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk⟩
[35] ⟨https://docs.sonarsource.com/sonarqube-server/latest/user-guide/rules/software-qualities⟩

e, por isso, devem ser solucionados imediatamente, como por exemplo, falhas de segurança que permitem ataques ou o roubo de informações.

O SonarQube executa sua análise avaliando os atributos do código com base em um conjunto de regras específicas para cada linguagem. Cada regra está associada a um ou mais dos três aspectos da qualidade de software vinculado a um nível de severidade. E juntos determinam o quanto a qualidade do software é afetada quando uma regra é quebrada, identificando o problema. Assim, um problema pode estar em mais de uma das qualidades analisadas e em diferente nível de severidade por cada qualidade. A Figura 13 mostra um exemplo de regra:

Figura 13. Exemplo de regra de severidade média.

Fonte: SonarSource (2025).

Nesse caso, no JavaScript, há operadores de comparação estritos, que comparam valor e tipo, e não estritos, que compara apenas o valor. Em uma comparação de **5 === '5'** por exemplo, o resultado seria falso, mas no caso de **5 == '5'**, o valor seria verdadeiro. Para evitar resultados contraintuitivos, é recomendado utilizar operadores de comparação estritos (SonarSource, 2025).

O SonarQube exibe a cobertura dos testes no software. Na solução proposta, na API e interface, os testes feitos com *Jest*[36] estão inclusos no resultado. O SonarQube permite configurar os arquivos que serão avaliados. Neste caso, foi configurado para serem desconsiderados os arquivos de testes e configurações em cada microsserviço. Assim, direcionando a avaliação para as funcionalidades desenvolvidas na solução. A Tabela 4 exibe os resultados obtidos:

Tabela 4. Resultado da avaliação de qualidade do código.

Projeto	**Problemas encontrados**			**Cobertura**
	Segurança	**Confiabilidade**	**Manutenibilidade**	
API RESTful	0	0	1(baixo)	97.2%
Interface de usuário	0	4(2 baixos, 2 médios)	16(8 baixos, 8 médios)	74.3%

Nos resultados, observa se que foram encontrados muito poucos erros, e com cobertura acima da média em ambos projetos. Os poucos problemas encontrados, por não terem altos níveis de severidade, pouco impactam no funcionamento da solução e são facilmente solucionáveis. O que permite afirmar a solução possui uma qualidade de código considerável.

6. Considerações Finais

O objetivo deste estudo foi prover uma forma de visualização de informações de estações meteorológicas. A partir da aplicação de conceitos de software voltados à experiência do usuário, desenvolveu-se uma solução composta por um microsserviço para comunicação e outro para exibição, oferecendo funcionalidades úteis para análise dos dados. Verificamos que a solução apresentou boa qualidade de usabilidade (76%), tempo de resposta aceitável (em torno de dois segundos), desempenho razoável da interface e boa qualidade de código, com poucos erros de baixo risco.

[36] ⟨https://jestjs.io⟩

Sugerimos que estudos futuros avancem em relação a esta pesquisa, buscando superar as limitações identificadas. Este trabalho implementou apenas funcionalidades básicas de visualização e análise, como painéis de gráficos para cada variável meteorológica em um intervalo anual. Novas funcionalidades podem incluir filtros de período com início e fim definidos pelo usuário e, preferencialmente, a incorporação de demandas levantadas diretamente com os usuários finais.

Os testes de usabilidade poderão ser repetidos após a aplicação das melhorias apontadas, agora com usuários finais, para identificar novas oportunidades de aprimoramento. O teste de desempenho, que atualmente considerou apenas o cenário ideal em ambiente de desenvolvimento, poderá ser refeito em condições reais de uso e sob cenários de estresse.

No desempenho da interface, após a implementação das melhorias e de novas funcionalidades, recomenda-se a realização de um novo teste de performance. O mesmo se aplica à qualidade do código: após corrigir os erros encontrados e incorporar novos recursos, deve-se efetuar nova análise do código-fonte, de modo a identificar mais pontos de melhoria e, assim, elevar a qualidade geral da solução. O repositório do código-fonte está disponível sob a licença GPL-3.0 em https://github.com/pedroaleph/avadec.git.

Referências

AKINSOLA, A. Evaluation of json-api-format rest api for business-to-business integration scenarios. 2018.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software architecture in practice*. [S.l.]: Addison-Wesley Professional, 2021.

BONDI, A. B. Characteristics of scalability and their impact on performance. In: *Proceedings of the 2nd international workshop on Software and performance*. [S.l.: s.n.], 2000. p. 195–203.

Brasil. Ministério da Ciência, Tecnologia e Inovação. *Modelagem climática e vulnerabilidades setoriais à mudança do clima no Brasil*. Brasília: [s.n.], 2016. ⟨https://www.gov.br/mcti/pt-br/acompanhe-o-mcti/cgcl/arquivos/modelagem-climatica-e-vulnerabilidades-setoriais-a-mudanca-do-clima-no-brasil/mcti-livromodelagemclimatica-edicaao-eletroenica-31mai2016_baixa_resolucaao.pdf⟩. Acesso em: 10 ago. 2025.

BRENDEL, C. E.; DYMOND, R. L.; AGUILAR, M. F. An interactive web app for retrieval, visualization, and analysis of hydrologic and meteorological time series data. *Environmental Modelling & Software*, Elsevier, v. 117, p. 14–28, 2019.

DAVE, C. V.; PATEL, A.; KESHRI, U. Microservices software architecture: a review. *Int. J. Res. Appl. Sci. Eng. Technol*, v. 9, n. 11, p. 1494–1496, 2021.

EHSAN, A. et al. Restful api testing methodologies: Rationale, challenges, and solution directions. *Applied Sciences*, MDPI, v. 12, n. 9, p. 4369, 2022.

Food and Agriculture Organization of the United Nations. *FAO's work on climate change*. Rome: [s.n.], 2019. ⟨https://sdgs.un.org/sites/default/files/2021-05/FAO%E2%80%99s%20Work%20on%20Climate%20Change.pdf⟩. Acesso em: 10 ago. 2025.

GOMMES, R. et al. Guide to agricultural meteorological practices. *World Meteorological Organization*, v. 134, 2010.

IOANNOU, K. et al. Low-cost automatic weather stations in the internet of things. *Information*, MDPI, v. 12, n. 4, p. 146, 2021.

JACOB, R. J. User interface. In: *Encyclopedia of Computer Science*. [S.l.: s.n.], 2003. p. 1821–1826.

JAMSHIDI, P. et al. Microservices: The journey so far and challenges ahead. *IEEE Software*, IEEE, v. 35, n. 3, p. 24–35, 2018.

JARRAUD, M. Guide to meteorological instruments and methods of observation (wmo-no. 8). *World Meteorological Organisation: Geneva, Switzerland*, v. 29, 2008.

JAVEED, A. Performance optimization techniques for reactjs. In: IEEE. *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. [S.l.], 2019. p. 1–5.

KOCH, P. R. Using kepler. gl to visualize weather data. Universidade de Passo Fundo, 2018.

LESTARI, D. M.; HARDIANTO, D.; HIDAYANTO, A. N. Analysis of user experience quality on responsive web design from its informative perspective. *International Journal of Software Engineering and its Applications*, v. 8, p. 53–62, 2014.

MITROPOULOS, S. et al. An online emergency medical management information system using mobile computing. *Applied Computing and Informatics*, 05 2021.

PATEL, M.; SHANGKUAN, J.; THOMAS, C. What's new with the internet of things. *McKinsey & Company*, p. 1–8, 2017.

PERAZZI, P. R. et al. O tradicional ou o moderno? uma visao da informaçao da rede de estações meteorológicas brasileiras. *Revista Brasileira de Meteorologia*, SciELO Brasil, v. 36, p. 351–366, 2021.

RITCHIE, H.; ROSADO, P.; ROSER, M. Agricultural production. *Our World in Data*, 2023. Https://ourworldindata.org/agricultural-production.

ROTO, V.; KAASINEN, E. The second international workshop on mobile internet user experience. In: *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*. New York, NY, USA: Association for Computing Machinery, 2008. (MobileHCI '08), p. 571–573. ISBN 9781595939524. Disponível em: ⟨https://doi.org/10.1145/1409240.1409354⟩.

SADIKU, M. et al. Data visualization. *International Journal of Engineering Research And Advanced Technology (IJERAT)*, v. 2, n. 12, p. 11–16, 2016.

SALES, L. H. M. Redes de sensores hierárquicas com aplicação em propriedades rurais baseadas na arquitetura iot. *UFRR*, 2022.

SERVICES, A. W. *Implementing Microservices on AWS*. 2025. Whitepaper. Disponível em: ⟨https://docs.aws.amazon.com/pdfs/whitepapers/latest/microservices-on-aws/microservices-on-aws.pdf⟩.

SOLER-MÉNDEZ, M. et al. Agroclimatic evolution web application as a powerful solution for managing climate data. *Scientific Reports*, Nature Publishing Group UK London, v. 12, n. 1, p. 6716, 2022.

SonarSource. *SonarQube Community Edition, version 25.7.0.110598 (MQR Mode)*. 2025. Acessado em: 11 ago. 2025. Disponível em: ⟨https://www.sonarsource.com/products/sonarqube⟩.

THÖNES, J. Microservices. *IEEE software*, IEEE, v. 32, n. 1, p. 116–116, 2015.

World Meteorological Organization. *Empowering farmers: the role of agrometeorological services in sustainable agriculture*. 2024. ⟨https://wmo.int/media/update/

empowering-farmers-role-of-agrometeorological-services-sustainable-agriculture⟩. Acesso em: 10 ago. 2025.

ZOTARELLI, L. et al. Step by step calculation of the penman-monteith evapotranspiration (fao-56 method). *Institute of Food and Agricultural Sciences. University of Florida*, v. 1, 2010.

ANEXOS

ANEXO 01 - DECLARAÇÃO DE AUTORIA

DECLARAÇÃO DE AUTORIA

Eu, **Pedro Aleph Gomes de Souza Vasconcelos** (código de matricula **2016007150**), autor da monografia/TCC (Trabalho de Conclusão de Curso) sob o título **Sistema Integrado de Monitoramento Climático na Amazônia com Tecnologias Abertas**, declaro que o trabalho em referência é de minha total autoria e de minha inteira responsabilidade o texto apresentado. Declaro, ainda, que as citações e paráfrases dos autores estão indicadas com as respectivas obras e anos de publicação. Declaro, para os devidos fins que estou ciente:

- dos Artigos 297 a 299 do Código Penal, Decreto-Lei n. 2.848 de 7 de dezembro de 1940;
- da Lei n. 9.610, de 19 de fevereiro de 1998, sobre os Direitos Autorais; e
- que plágio consiste na reprodução de obra alheia e submissão da mesma como trabalho próprio ou na inclusão, em trabalho próprio, de ideias, textos, tabelas ou ilustrações (quadros, figuras, gráficos, fotografias, retratos, lâminas, desenhos, organogramas, fluxogramas, plantas, mapas e outros) transcritos de obras de terceiros sem a devida e correta citação da referência.

O corpo docente responsável pela avaliação deste trabalho poderá não aceitar o referido trabalho caso os pontos mencionados acima sejam descumpridos, por conseguinte, considerar-me reprovado.

gov.br
Documento assinado digitalmente
PEDRO ALEPH GOMES DE SOUZA VASCONCELO
Data: 05/11/2025 11:02:55-0300
Verifique em https://validar.iti.gov.br

Assinatura do acadêmico(a)
Boa Vista - RR, 15 de agosto de 2025.