

ÂNGELO GABRIEL HOLANDINI FREITAS

**SISTEMA WEB DE GERENCIAMENTO DE DISPOSITIVOS IOT**

CAMPINAS  
2025

ÂNGELO GABRIEL HOLANDINI FREITAS

**SISTEMA WEB DE GERENCIAMENTO DE DISPOSITIVOS IOT**

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do diploma do Curso Tecnólogo de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia Campus Campinas.

Orientador: Prof. Dr. Glauber da Rocha Balthazar.

CAMPINAS  
2025

Ficha Catalográfica  
Instituto Federal de São Paulo – Campus Campinas  
Biblioteca – Pedro Augusto Pinheiro Fantinatti  
Rosangela Gomes – CRB 8/8461

Freitas, Ângelo Gabriel Holandini.

Sistema web de gerenciamento de dispositivos IoT./ Ângelo Gabriel Holandini  
Freitas. – Campinas, SP: [s.n.], 2025.  
42 f. : il.

Orientador: Glauber da Rocha Balthazar. Trabalho de Conclusão de Curso  
(graduação) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo -  
Campus Campinas. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas,  
2025.

1. Internet das coisas. 2. Aplicação web. 3. Java. 4. React.JS. 5. Scrum. I. Instituto  
Federal de Educação, Ciência e Tecnologia de São Paulo, Campus Campinas. Curso de  
Tecnologia em Análise e Desenvolvimento de Sistemas. II. Título

ATA N.º 3/2025 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - Graduação

Na presente data, realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulado **SISTEMA WEB DE GERENCIAMENTO DE DISPOSITIVOS IOT**, apresentado(a) pelo(a) estudante **ÂNGELO GABRIEL HOLANDINI FREITAS CP3013901** do Curso **SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS** (Campus Campinas). Os trabalhos foram iniciados às 18:30 pelo(a) Professor(a) presidente da banca examinadora, constituída pelos seguintes membros:

Membros	Instituição	Presença (Sim/Não)
Glauber da Rocha Balthazar	IFSP	Sim
Zady Castaneda Salazar	IFSP	Sim
Ricardo Sovat Barz	IFSP	Sim

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição do(a) candidato(a). Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo(a) estudante, tendo sido atribuído o seguinte resultado:

☒ [ X ] Aprovado(a) ☐ [ ] Reprovado(a)

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino em nome dos demais membros da banca examinadora.

Campus Campinas, 13 de maio de 2025

Documento assinado eletronicamente por:

- **Glauber da Rocha Balthazar**, PROFESSOR ENS BASICO TECN TECNOLOGICO , em 13/05/2025 19:55:22.
- **Ricardo Barz Sovat**, PROFESSOR ENS BASICO TECN TECNOLOGICO , em 13/05/2025 19:55:50.
- **Zady Castaneda Salazar**, PROFESSOR ENS BASICO TECN TECNOLOGICO , em 13/05/2025 20:21:52.

Este documento foi emitido pelo SUAP em 13/05/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifsp.edu.br/autenticar-documento/> e forneça os dados abaixo:

Código Verificador: 949059  
Código de Autenticação: 1a05a51732



ATA N.º 3/2025 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

## **AGRADECIMENTOS**

Gostaria de expressar minha gratidão ao IFSP pelo apoio técnico e estrutura oferecida durante a realização deste trabalho. Agradeço especialmente ao Prof. Glauber da Rocha Balthazar, meu orientador, pela paciência, orientações valiosas e incentivo constante.

À minha família, principalmente à minha mãe Tatiana e ao meu irmão Filipe, pelo apoio incondicional e compreensão durante os momentos de dedicação exclusiva a esta pesquisa. E também direciono meus agradecimentos aos colegas de curso, que contribuíram com discussões enriquecedoras ao longo do processo.

## RESUMO

Este trabalho apresenta o desenvolvimento de um sistema web para gerenciamento de dispositivos IoT, com foco na coleta, transmissão, armazenamento e visualização de dados. Utilizou-se o microcontrolador ESP32 conectado ao sensor DHT22 para a leitura de temperatura e umidade, enviando os dados via protocolo HTTP para um servidor backend desenvolvido em Java. Os dados são armazenados em um banco de dados MySQL, administrados por um webservice construído com Java e disponibilizados através de uma aplicação web construída com React.js. A metodologia Scrum foi adotada para o gerenciamento do projeto, garantindo entregas incrementais e feedback contínuo. Foram realizados testes práticos comprovando a eficácia do sistema em ambientes controlados, validando sua capacidade de monitoramento em tempo real e geração de relatórios históricos. O sistema proposto surge como uma alternativa acessível e customizável frente a soluções comerciais existentes.

**Palavras-chave:** Internet das Coisas; Aplicação Web; Java; React.js; Scrum;

## ABSTRACT

This work presents the development of a web-based system for managing IoT devices, focusing on data collection, transmission, storage, and visualization. The ESP32 microcontroller connected to the DHT22 sensor was used to measure temperature and humidity, with data being sent via the HTTP protocol to a backend server developed in Java. The data is stored in a MySQL database, managed by a web service built with Java, and made available through a web application developed using React.js. The Scrum methodology was adopted for project management, ensuring incremental deliveries and continuous feedback. Practical tests were conducted to validate the system's effectiveness in controlled environments, proving its capability for real-time monitoring and generation of historical reports. The proposed system emerges as an affordable and customizable alternative compared to existing commercial solutions.

**Keywords:** Internet of Things; Web Application; Java; React.js; Scrum;

## SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>8</b>
1.1 Justificativa.....	9
1.2 Objetivos.....	10
1.3.1 <i>Objetivos específicos</i> .....	10
<b>2 FUNDAMENTAÇÃO TEÓRICA.....</b>	<b>11</b>
2.1 ESP8266.....	11
2.2 DHT22.....	11
2.3 JavaScript.....	12
2.4 React.js.....	12
2.5 HTML.....	13
2.6 CSS.....	13
2.7 Java.....	14
2.8 Maven.....	14
2.9 Apache Tomcat.....	15
2.10 MySQL.....	15
<b>3 METODOLOGIA.....</b>	<b>16</b>
<b>4 RESULTADOS.....</b>	<b>17</b>
4.1 Framework Scrum.....	17
4.1.1 <i>Product backlog</i> .....	17
4.1.1.1 <i>Requisitos funcionais</i> .....	17
4.1.1.2 <i>Requisitos não funcionais</i> .....	19
4.1.2 <i>Sprint planning</i> .....	21
4.1.3 <i>Sprint goal</i> .....	23
4.1.4 <i>Daily Scrum</i> .....	24
4.2 Diagramações.....	25
4.2.1 <i>Casos de uso</i> .....	25
4.2.2 <i>Diagrama de componentes</i> .....	27
4.2.3 <i>Diagrama de tabelas e relacionamentos (DTR)</i> .....	29
4.3 Programação e testes.....	29
4.3.1 <i>Programação</i> .....	29
4.3.2 <i>Testes</i> .....	35
<b>5 CONSIDERAÇÕES FINAIS.....</b>	<b>39</b>
<b>REFERÊNCIAS.....</b>	<b>40</b>



## 1 INTRODUÇÃO

Os dispositivos de Internet das Coisas (IoT) têm se tornado fundamentais em diversos setores pois possibilitam a coleta e análise de dados em tempo real resultando em melhorias significativas na eficiência operacional e na tomada de decisões (Santos; Silva, 2023). O crescimento recente do mercado de IoT apresenta previsões indicando que o mercado de gerenciamento de dados IoT deve crescer a uma taxa composta de crescimento anual (Taxa de Crescimento Anual Composta - CAGR) de 16,58% entre 2021 e 2026 impulsionado pela modernização das arquiteturas de dados (Fernandez, 2024) e pela crescente demanda por soluções de segurança e análise de dados (Lara; Reis; Tissot-Lara; Silva, 2021). Além disso, a Cisco estima que a IoT gerará cerca de 507,5 zettabytes de dados representando um desafio considerável para as organizações em termos de armazenamento, triagem e análise (Rosa; De Souza; Da Silva, 2020).

O surgimento de novos dispositivos associado a formas facilitadoras de programação e manipulação de dados facilitaram a coleta, transmissão e digitalização de processos físicos que são utilizados em diversas etapas de gestão de serviços, como por exemplo o monitoramento meteorológico (Santos; Silva, 2023). Além disso, a computação em nuvem facilita o acesso ágil e rápido aos recursos obtidos por dispositivos de IoT (parâmetros de sensores e atuadores) possibilitando a criação de modelos computacionais que permitem o acesso sob demanda a um conjunto de recursos do sensoriamento remoto compartilhados em diferentes ambientes físicos (Nacional Institute of Standards and Technology, 2011). Portanto, a implementação de estratégias robustas de gerenciamento de dados é essencial para que as empresas possam extrair valor dos insights gerados por esses dispositivos, garantindo a segurança e a integridade das informações.

Com o avanço das tecnologias de IoT tornou-se possível desenvolver sistemas de baixo custo e alta eficiência para monitorar parâmetros em tempo real (Soares; Faria, 2021). Diversas plataformas online atualmente fazem o armazenamento de dados de IoT e monitoramento online como por exemplo ThingSpeak<sup>1</sup>, Ubidots<sup>2</sup> e ThingsBoard<sup>3</sup>, porém todas pagas ou com planos muito limitados para acesso livre. Essas plataformas permitem a coleta contínua de dados de sensores integrados em diversos dispositivos, facilitando a análise e a tomada de decisões informadas. Além disso, a conectividade proporcionada pela IoT

---

<sup>1</sup> <https://thingspeak.mathworks.com/>

<sup>2</sup> <https://ubidots.com/>

<sup>3</sup> <https://thingsboard.io/device-management/>

possibilita a integração de diferentes dispositivos e plataformas, resultando em soluções inteligentes que podem otimizar processos em setores como agricultura, saúde, manufatura e transporte. A capacidade de identificar problemas rapidamente e implementar ações corretivas em tempo real não só melhora a eficiência operacional mas também contribuem para a redução de custos e o aumento da sustentabilidade. Este trabalho apresenta a hipótese de ser possível a criação de uma solução informatizada e integrada a diferentes tipos de dispositivos de IoT para coleta, transmissão, armazenamento e disponibilização de dados em uma aplicação web possibilitando o gerenciamento dos dados de forma remota e persistente.

## **1.1 Justificativa**

O monitoramento por meio de Internet das Coisas (IoT) tem revolucionado a forma como os dados são coletados e geridos em diversas aplicações. A transmissão de dados em tempo real, possibilitada por dispositivos conectados, permite a coleta contínua de informações cruciais para a tomada de decisões (Gubbi; Buyya; Marusic; Palaniswami, 2013). Após a coleta, o armazenamento desses dados deve ser realizado com segurança objetivando a garantia de que as informações possam ser analisadas posteriormente e, nestes casos, as soluções em nuvem têm se mostrado eficazes devido à sua escalabilidade e acessibilidade (Chang; Srirama; Buyya, 2019). A visualização adequada dos dados é igualmente importante, pois transforma informações complexas em representações compreensíveis, facilitando a interpretação e análise (Chang; Srirama; Buyya, 2019). Por fim, o gerenciamento desses dados, que envolve a persistência, organização, análise e segurança, é fundamental para extrair insights valiosos e garantir a integridade das informações (Kirk, 2012).

A utilização de tecnologias modernas, como o ESP32, e protocolos de comunicação, como Hypertext Transfer Protocol (HTTP), oferecem uma alternativa eficiente e de baixo custo para o monitoramento em tempo real. Além disso, a integração destas tecnologias com uma aplicação web permite o acesso remoto aos dados facilitando a tomada de decisão de forma rápida e precisa. Este trabalho busca preencher essa lacuna (entre coleta, armazenamento e disponibilização de dados) propondo uma solução acessível e de fácil utilização para o monitoramento de diferentes módulos sensores em ambientes remotos.

Por fim, como técnicas de implementação tecnológica dessa solução são propostos a utilização de tecnologias modernas de gerenciamento e desenvolvimento de sistemas web. Para o gerenciamento das etapas de desenvolvimento do sistema web foi adotado o Scrum

como metodologia ágil de gerenciamento de projetos e como tecnologias de programação web React.js, HTML e CSS. O Scrum justifica-se por ser um framework que consegue entregar valor ao cliente de forma rápida e contínua por meio da criação de produtos de software homologáveis a cada iteração (sprint) de desenvolvimento possibilitando rápido feedback e melhoria contínua no produto final (Leffingwell, 2011). React.js justifica-se pela eficiência e popularidade dessas tecnologias no desenvolvimento de sistemas IoT e aplicações web modernas.

## **1.2 Objetivos**

Desenvolver uma solução web integrada para o armazenamento e gerenciamento dos dados de dispositivos IoT responsáveis pela coleta e transmissão de dados a partir de sensores e atuadores de monitoramento local.

### ***1.3.1 Objetivos específicos***

- Implementar a coleta de dados de temperatura e umidade utilizando o ESP32 e o sensor DHT22;
- Desenvolver um web service para armazenamento e disponibilização dos dados coletados;
- Criar uma aplicação web em React.js para visualização dos dados em tempo real e geração de relatórios históricos; e
- Testar o sistema em um ambiente controlado para validar sua funcionalidade e eficiência.

## **2 FUNDAMENTAÇÃO TEÓRICA**

### **2.1 ESP8266**

O ESP8266 é um microcontrolador amplamente utilizado em projetos de Internet das Coisas (IoT) devido ao seu baixo custo, alta eficiência energética e capacidade de conexão Wi-Fi e Bluetooth. Desenvolvido pela Espressif Systems, ele possui um processador dual-core, memória RAM integrada e uma variedade de periféricos, como GPIOs, interfaces SPI, I2C e ADC, que o tornam versátil para diversas aplicações (Sathish, 2021). No contexto deste trabalho, o ESP32 é responsável por coletar dados do sensor DHT22 e transmiti-los via Wi-Fi sendo a peça central do sistema de monitoramento proposto.

Além disso, o ESP32 é programado utilizando a linguagem C/C++ no ambiente Arduino IDE ou no framework ESP-IDF, o que facilita o desenvolvimento de firmware para aplicações específicas. Sua capacidade de operar em modo de baixo consumo de energia e sua robustez em ambientes industriais o tornam ideal para projetos como o monitoramento de frigoríficos, onde a confiabilidade e a eficiência são essenciais.

### **2.2 DHT22**

O DHT22 é um sensor digital de temperatura e umidade amplamente utilizado em projetos de monitoramento ambiental devido à sua precisão e facilidade de integração com microcontroladores como o ESP8266 e o ESP32. Ele oferece medições precisas, com uma faixa de temperatura de -40°C a 80°C e umidade relativa de 0% a 100%, sendo ideal para aplicações que exigem confiabilidade e baixo custo (Silva; Santos, 2023). O sensor utiliza um protocolo serial de comunicação, enviando dados em formato digital, o que reduz a necessidade de circuitos adicionais para conversão analógico-digital e simplifica sua conexão com sistemas embarcados.

No sistema proposto, o DHT22 desempenha um papel essencial ao coletar dados de temperatura e umidade no interior dos frigoríficos, fornecendo informações cruciais para o monitoramento em tempo real (Oliveira; Costa, 2024). Sua capacidade de operar de forma eficiente em ambientes controlados o torna adequado para garantir que as condições de armazenamento estejam dentro dos limites aceitáveis, evitando perdas de produtos e assegurando a qualidade dos itens armazenados.

## 2.3 JavaScript

JavaScript é uma linguagem de programação amplamente utilizada para adicionar interatividade e dinamismo a páginas da web, sendo interpretada diretamente no navegador como parte essencial do desenvolvimento front-end, ao lado de HTML e CSS. Essa linguagem de alto nível, conhecida por sua legibilidade e facilidade de uso, foi originalmente desenvolvida por Brendan Eich na Netscape Corporation em 1995 e evoluiu para se tornar uma ferramenta indispensável no desenvolvimento de sistemas modernos (Flanagan, 2020). Além de possibilitar a criação de elementos interativos, como mapas dinâmicos, gráficos animados e interfaces responsivas, o JavaScript também permite a implementação de funcionalidades avançadas, como aplicativos baseados em banco de dados e atualizações automáticas de conteúdo sem a necessidade de recarregar a página (Crockford, 2018).

Entre suas principais características, destaca-se o fato de ser uma linguagem de script, ou seja, voltada para a execução de sequências de comandos, além de ser baseada em objetos, permitindo a manipulação de estruturas modulares e reutilizáveis (Goodrich; Tamassia; Mount, 2021). Outra peculiaridade notável é sua tipagem fraca, que dispensa a definição explícita do tipo de variável durante a codificação, conferindo flexibilidade ao desenvolvedor (Zakas, 2019). Essa versatilidade, aliada à sua simplicidade, faz com que o JavaScript seja uma linguagem acessível tanto para iniciantes quanto para profissionais experientes, sendo amplamente empregada no desenvolvimento de sistemas web, incluindo aqueles voltados para gerenciamento de dispositivos IoT (González; Martínez; García, 2022).

## 2.4 React.js

React.js é uma biblioteca JavaScript de código aberto, inicialmente desenvolvida pelo Facebook (atual Meta), projetada para a construção de interfaces de usuário (UI) dinâmicas e reativas (Walke, 2015). Sua arquitetura baseada em componentes permite a modularização do código, facilitando a reutilização de elementos e a manutenção de sistemas complexos, como os que envolvem atualizações em tempo real de dados de sensores IoT (Stoyanov, 2016). No sistema proposto, React.js é adotado para desenvolver a aplicação web que exibe métricas de temperatura e umidade em tempo real, além de gerar relatórios históricos, graças à sua

eficiência em manipular estados dinâmicos e renderização otimizada (Adams; Bisch; Newcomer, 2021).

Uma das principais características do React.js é a utilização do Virtual DOM, um mecanismo que minimiza o custo computacional de atualizações na interface, comparando apenas as alterações necessárias antes de aplicá-las ao DOM real. Esse recurso é crítico em sistemas de monitoramento IoT, onde a frequência de atualizações pode ser alta, garantindo desempenho mesmo em ambientes com limitações de recursos. Além disso, a ampla adoção do React.js pela indústria impulsiona um ecossistema robusto, com bibliotecas complementares (como Redux para gerenciamento de estado) e ferramentas que aceleram o desenvolvimento (Alexander, 2022).

## **2.5 HTML**

HTML (HyperText Markup Language) é a linguagem padrão para estruturação de conteúdo em páginas web, definindo elementos como textos, imagens e formulários que são interpretados pelos navegadores. No sistema proposto, o HTML é integrado ao React.js para estabelecer a arquitetura da aplicação, organizando componentes que exibem dados de temperatura e umidade em tempo real. Sua simplicidade estrutural contrasta com sua importância fundamental: o HTML serve como base para interfaces de usuário, permitindo a combinação com CSS e JavaScript para a criação de páginas dinâmicas e acessíveis (Silva, 2022).

A semântica do HTML é essencial para garantir a interoperabilidade entre dispositivos e navegadores, alinhando-se a padrões de acessibilidade que facilitam o acesso a usuários com necessidades especiais. Essa característica é crítica no contexto deste trabalho, onde a aplicação web deve ser funcional em múltiplos ambientes, seguindo diretrizes de desenvolvimento moderno.

## **2.6 CSS**

CSS (Cascading Style Sheets) é uma linguagem essencial para definir a apresentação visual de páginas web, controlando elementos como cores, fontes, layouts e animações. Ele opera em conjunto com HTML, promovendo a separação entre conteúdo e estilo, o que facilita a manutenção e escalabilidade de aplicações web. No sistema proposto, o CSS é

utilizado para estilizar a aplicação React.js, garantindo uma interface moderna, responsiva e alinhada com padrões de design contemporâneos.

Uma de suas principais vantagens é a capacidade de adaptar layouts a diferentes dispositivos por meio de técnicas como media queries e flexbox, que permitem o ajuste dinâmico de elementos conforme a tela do usuário. Essa adaptabilidade é crítica para aplicações que demandam acesso multiplataforma, como sistemas de monitoramento IoT, onde a experiência do usuário deve ser consistente em desktops, tablets e smartphones. No contexto deste trabalho, o CSS foi fundamental para garantir não apenas a estética, mas também a usabilidade da aplicação, seguindo boas práticas de design responsivo e acessibilidade.

## **2.7 Java**

Java é uma linguagem de programação orientada a objetos, multiplataforma e robusta, amplamente utilizada no desenvolvimento de sistemas empresariais e aplicações backend. Sua tipagem estática e capacidade de processamento avançado simplificam a criação de APIs RESTful e serviços web escaláveis. No sistema proposto, Java é empregado para desenvolver o web service responsável por receber dados, armazená-los em um banco de dados MySQL e disponibilizá-los ao frontend React.js por meio de endpoints estruturados (Silva, 2023). A escalabilidade e segurança do Java são características críticas para aplicações industriais, onde confiabilidade e desempenho são prioritários (Oliveira, 2023).

## **2.8 Maven**

Maven é uma ferramenta de automação de construção e gerenciamento de projetos amplamente utilizada em ecossistemas Java, destacando-se por sua capacidade de padronizar processos de desenvolvimento. Ele simplifica processos de build, gerenciamento de dependências e configuração de projetos por meio de arquivos XML, permitindo a padronização de projetos e integração com frameworks como Spring. Sua comunicação com Java ocorre por meio da definição de dependências em arquivos POM (*Project Object Model*), que automatizam o download de bibliotecas necessárias e garantem a compatibilidade entre componentes. Além disso, o Maven facilita a construção de aplicações escaláveis ao

integrar-se com repositórios remotos, como o Maven Central, otimizando o fluxo de trabalho em sistemas corporativos.

## 2.9 Apache Tomcat

Apache Tomcat é um servidor de aplicações Java amplamente utilizado para implementar e executar aplicações web baseadas em tecnologias como Servlets, JavaServer Pages (JSP) e Jakarta EE. Desenvolvido pela Apache Software Foundation, ele atua como um contêiner de *servlets*, gerenciando requisições HTTP e processando lógicas de negócios em Java, conforme especificações da plataforma Jakarta. Sua arquitetura leve e flexível permite a hospedagem de aplicações empresariais críticas, sendo adotado em setores que demandam escalabilidade e confiabilidade.

Diferente de servidores HTTP tradicionais, Tomcat é otimizado para aplicações Java, suportando tanto a execução de código dinâmico quanto a interoperabilidade com bancos de dados e serviços externos. Sua licença open-source e documentação detalhada reforçam seu papel central em projetos que exigem customização e desempenho.

## 2.10 MySQL

MySQL é um sistema de gerenciamento de banco de dados relacional (SGBD) de código aberto, amplamente utilizado em aplicações web. Ele permite o armazenamento, organização e recuperação de dados de forma eficiente, utilizando a linguagem SQL (Structured Query Language). No sistema proposto, MySQL é utilizado para armazenar os dados de temperatura e umidade coletados pelo ESP8266, garantindo que as informações estejam disponíveis para consulta e geração de relatórios.

Além disso, MySQL é conhecido por sua confiabilidade, desempenho e facilidade de integração com outras tecnologias. No contexto deste trabalho, MySQL é essencial para garantir que os dados coletados sejam armazenados de forma segura organizada, permitindo a geração de relatórios históricos e a análise das condições de armazenamento ao longo do tempo.



### 3 METODOLOGIA

A proposta de desenvolvimento de software foi amparada pelo suporte do framework ágil para gerenciamento de projetos Scrum. Esse framework se baseia em ciclos curtos de trabalho chamados Sprints, onde equipes auto-organizadas desenvolvem funcionalidades priorizadas, garantindo flexibilidade e adaptação a mudanças. Para tanto o trabalho foi proposto na adaptação de um ciclo iterativo e incremental de desenvolvimento de software baseado nas seguintes atividades:

- desenvolvimento do product backlog: baseado em um conjunto de requisitos funcionais e não funcionais,
- Sprint Planning: detalhamento das atividades dos sprints baseado em um modelo de vida útil estipulado no período de uma a quatro semanas,
- Sprint Goal: determinação das metas a serem produzidas em formato de software para contemplar o modelo iterativo e incremental e
- Daily Scrum: com as reuniões de orientação do desenvolvimento do software e também de apresentação, validação e homologação dos sprints goals.

Além disso, para atender aos requisitos mínimos de documentação de software foram propostos o desenvolvimento dos seguintes diagramas:

- Casos de Uso: contemplando as funcionalidades do software,
- Componentes: contemplando as interações entre os sistemas e
- Tabelas e Relacionamentos: contemplando a persistência dos dados.

Por fim, o software produzido foi testado utilizando uma plataforma de um microcontrolador com captura de dados de um microambiente (dados ambientais de temperatura e umidade) para posterior envio, armazenamento e visualização de dados.

4 RESULTADOS

4.1 Framework Scrum

4.1.1 Product backlog

O Product Backlog é uma lista dinâmica de funcionalidades e características desejadas para o sistema de gerenciamento de dispositivos IoT. Este backlog inclui requisitos funcionais e não funcionais que guiam o desenvolvimento do projeto.

4.1.1.1 Requisitos funcionais

Os requisitos funcionais foram classificados como sendo todas as necessidades, funcionalidades ou características que são esperadas no software desenvolvido. Portanto, os quadros de 1 a 5 listam os requisitos funcionais do sistema, contendo seu identificador, nome, data de criação, data da última criação, prioridade, versão, e descrição.

Quadro 1 - Coleta de Dados de Sensores

Identificador	RF001
Nome	Coleta de Dados de Sensores
Data de criação	01/03/2025
Data da última alteração	N/A
Prioridade	Alta
Versão	1.0
Descrição	O sistema deve coletar dados de temperatura e umidade do sensor DHT22.

Fonte: Elaborado pelo autor (2025)

Quadro 2 - Transmissão de Dados

<b>Identificador</b>	RF002
<b>Nome</b>	Transmissão de Dados
<b>Data de criação</b>	01/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Alta
<b>Versão</b>	1.0
<b>Descrição</b>	Transmitir os dados coletados para o servidor usando protocolo HTTP.

Fonte: Elaborado pelo autor (2025)

Quadro 3 - Armazenamento de Dados

<b>Identificador</b>	RF003
<b>Nome</b>	Armazenamento de Dados
<b>Data de criação</b>	01/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Alta
<b>Versão</b>	1.0
<b>Descrição</b>	Implementar banco de dados MySQL para armazenar dados de sensores.

Fonte: Elaborado pelo autor (2025)

Quadro 4 - Visualização em Tempo Real

<b>Identificador</b>	RF004
<b>Nome</b>	Visualização em Tempo Real
<b>Data de criação</b>	01/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Média
<b>Versão</b>	1.0
<b>Descrição</b>	Exibir dados de temperatura e umidade em tempo real via aplicação web.

Fonte: Elaborado pelo autor (2025)

Quadro 5 - Geração de Relatórios

<b>Identificador</b>	RF005
<b>Nome</b>	Geração de Relatórios
<b>Data de criação</b>	01/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Média
<b>Versão</b>	1.0
<b>Descrição</b>	Permitir geração de relatórios históricos dos dados coletados.

Fonte: Elaborado pelo autor (2025)

#### **4.1.1.2 Requisitos não funcionais**

Os requisitos não funcionais descrevem uma funcionalidade, necessidade ou característica que o sistema deve conter, ou seja, uma descrição de como o sistema realizará determinada função. Desta forma, nos quadros de 6 até 9 estão listados os requisitos não funcionais do sistema, contendo seu identificador, nome, data de criação, data da última criação, prioridade, versão, e descrição.

Quadro 6 - Segurança de Dados

<b>Identificador</b>	RNF001
<b>Nome</b>	Segurança de Dados
<b>Data de criação</b>	02/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Alta
<b>Versão</b>	1.0
<b>Descrição</b>	Garantir transmissão e armazenamento seguro dos dados.

Fonte: Elaborado pelo autor (2025)

Quadro 7 - Desempenho

<b>Identificador</b>	RNF002
<b>Nome</b>	Desempenho
<b>Data de criação</b>	02/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Alta
<b>Versão</b>	1.0
<b>Descrição</b>	Otimizar a aplicação para garantir tempos de resposta rápidos.

Fonte: Elaborado pelo autor (2025)

Quadro 8 - Usabilidade

<b>Identificador</b>	RNF003
<b>Nome</b>	Usabilidade
<b>Data de criação</b>	02/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Média
<b>Versão</b>	1.0
<b>Descrição</b>	Interface intuitiva e fácil de usar para uma experiência agradável.

Fonte: Elaborado pelo autor (2025)

Quadro 9 - Compatibilidade

<b>Identificador</b>	RNF004
<b>Nome</b>	Compatibilidade
<b>Data de criação</b>	02/03/2025
<b>Data da última alteração</b>	N/A
<b>Prioridade</b>	Média
<b>Versão</b>	1.0
<b>Descrição</b>	Compatibilidade do sistema com diferentes navegadores e dispositivos.

Fonte: Elaborado pelo autor (2025)

#### 4.1.2 *Sprint planning*

O desenvolvimento de projeto é pensado para ser dividido em *sprints* - períodos de tempo onde um ou mais requisitos (funcionais ou não funcionais) são escolhidos para serem construídos e entregues. O período ideal de duração para cada *sprint* é de 1 a 4 semanas.

Antes do início de cada *sprint*, é feito um planejamento de *sprint* (*Sprint Planning*, em inglês), onde é definido quantos e quais requisitos podem ser construídos e entregues. Após o

término de uma *sprint*, é esperada a entrega de uma nova adição ao software focada nos requisitos desenvolvidos.

A seguir são apresentadas a divisão dos requisitos em *sprints* e o tempo estimado para cada um. Também é apresentada no quadro 10 a visualização geral do *Sprint Planning* em um gráfico de Gantt.

- Sprint 1: Coleta e Transmissão de Dados
  - Requisitos: [RF001] Coleta de Dados de Sensores; [RF002] Transmissão de Dados
  - Duração: 2 semanas
- Sprint 2: Armazenamento
  - Requisitos: [RF003] Armazenamento de Dados
  - Duração: 2 semanas
- Sprint 3: Visualização e Relatórios
  - Requisitos: [RF004] Visualização em Tempo Real; [RF005] Geração de Relatórios
  - Duração: 4 semanas
- Sprint 4: Requisitos Não Funcionais
  - Requisitos: [RNF001] Segurança de Dados; [RNF002] Desempenho; [RNF003] Usabilidade; [RNF004] Compatibilidade
  - Duração: 4 semanas

Quadro 10 - Sprint Planning

Semana	Sprint 1	Sprint 2	Sprint 3	Sprint 4
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Fonte: Elaborado pelo autor (2025)

#### 4.1.3 Sprint goal

O *Sprint Goal* é uma declaração de propósito para cada *sprint*, orientando o que precisa ser alcançado durante o ciclo de desenvolvimento. Essa meta deve alinhar os esforços do desenvolvedor e servir como um critério para avaliar o sucesso do *sprint*. Os objetivos devem ser revisados e ajustados conforme necessário, com base em *feedback* e resultados dos *sprints* anteriores.

A seguir, são listados os *sprints* e seus respectivos objetivos.

- Sprint 1 Goal: Estabelecer Conectividade e Coleta de Dados
  - Objetivo: Garantir que o ESP32 esteja coletando dados de temperatura e umidade com sucesso e transmitindo-os ao servidor via protocolo HTTP. A meta é ter um fluxo de dados contínuo e confiável entre o dispositivo e o servidor.



- Sprint 2 Goal: Implementar Segurança e Persistência de Dados
  - Objetivo: Configurar o armazenamento seguro dos dados no banco de dados MySQL, garantindo que apenas usuários autenticados tenham acesso às informações. A meta é garantir a integridade e segurança dos dados armazenados.
- Sprint 3 Goal: Prover Visualização Eficaz e Relatórios Precisos
  - Objetivo: Desenvolver uma interface web que permita aos usuários visualizar dados em tempo real e acessar relatórios históricos. A meta é garantir que os dados sejam exibidos claramente e que os relatórios sejam gerados com precisão.
- Sprint 4 Goal: Assegurar Qualidade de Serviço e Compatibilidade
  - Objetivo: Otimizar o sistema para garantir segurança, escalabilidade, desempenho e compatibilidade com diferentes dispositivos e navegadores. A meta é assegurar que o sistema atenda aos requisitos não funcionais, proporcionando uma experiência de usuário consistente e eficiente.

#### **4.1.4 Daily Scrum**

As reuniões de *Daily Scrum* são fundamentais para garantir a comunicação eficaz entre o desenvolvedor e o orientador, permitindo ajustes rápidos e alinhamento sobre o progresso do projeto. No contexto deste trabalho, as reuniões com o orientador desempenham um papel similar, fornecendo *feedback* e orientação contínua.

A seguir é apresentado a Tabela 11, que incorpora a Tabela 10 (Sprint Planning) às datas das reuniões com o orientador para validação e homologação dos Goals.

Quadro 11 - Sprint Planning e Daily Scrums

Semana	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Daily Scrums
1					
2					07/03
3					14/03
4					
5					25/03
6					
7					
8					14/04
9					
10					30/04
11					
12					

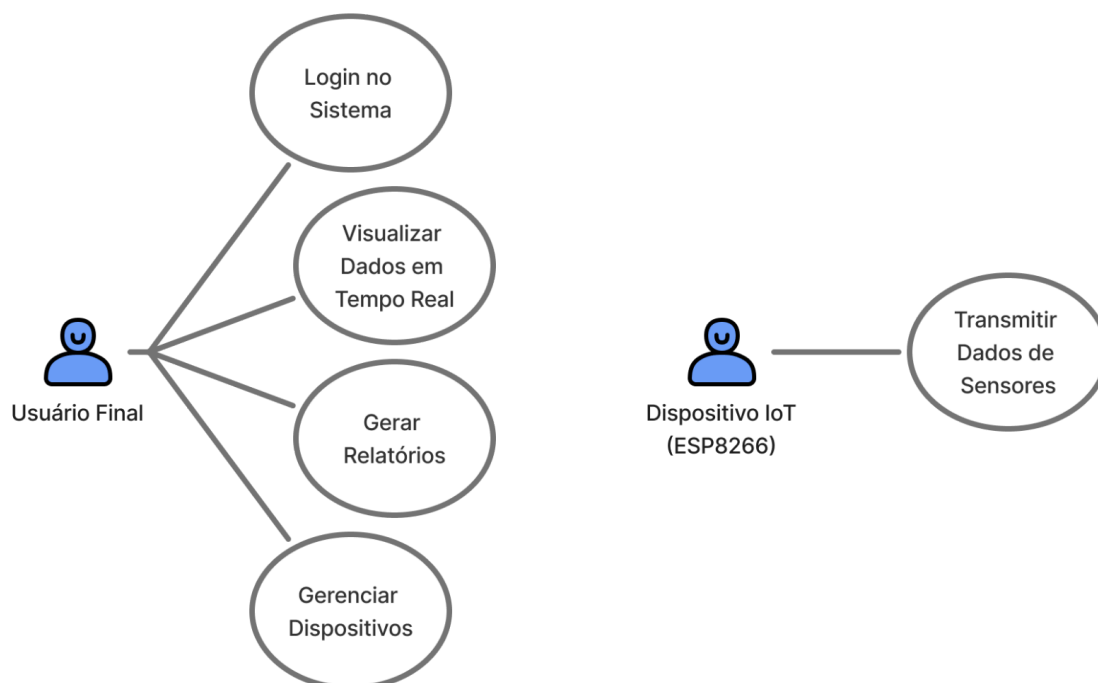
Fonte: Elaborado pelo autor (2025)

## 4.2 Diagramações

### 4.2.1 Casos de uso

Os casos de uso descrevem as interações entre atores (usuários ou sistemas externos) e o sistema com o objetivo de atingir um objetivo específico auxiliando a compreensão de como o sistema deve se comportar em diferentes situações, identificando funcionalidades, regras de negócio e possíveis fluxos de execução. Para este projeto foram determinados dois atores (usuário final e dispositivo IoT) e cinco casos de uso, todos descritos na Figura 1.

Figura 1 - Diagrama de Caso de Uso



Fonte: Elaborado pelo autor (2025)

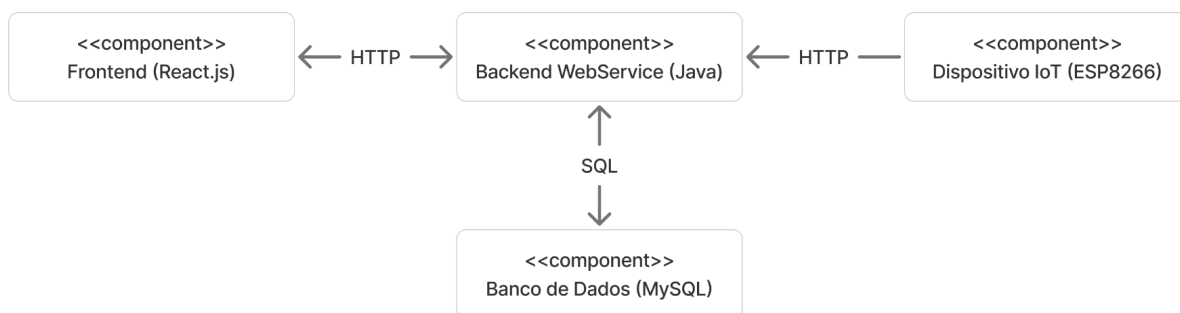
A seguir, cada um dos itens do diagrama de caso de uso é detalhado:

- Ator “Usuário Final”: Interage com a aplicação web para visualização de dados, geração de relatórios, e gerenciamento de dispositivos.
- Ator “Dispositivo IoT (ESP8266)”: Coleta e transmite dados de sensores.
- Caso de Uso “Login no Sistema”: Permite que o usuário se autentique no sistema.
- Caso de Uso “Visualizar Dados em Tempo Real”: Permite visualizar dados de temperatura e umidade em tempo real.
- Caso de Uso “Gerar Relatórios”: Permite ao usuário gerar relatórios sobre dados coletados ao longo do tempo.
- Caso de Uso “Configurar Dispositivos”: Permite ao usuário adicionar ou modificar configurações de dispositivos IoT.
- Caso de Uso “Gerenciar Usuários”: Permite ao usuário adicionar, remover ou modificar usuários do sistema.
- Caso de Uso “Transmitir Dados de Sensores”: Envia dados coletados para o servidor.

### 4.2.2 Diagrama de componentes

Esse diagrama apresenta a estrutura física e lógica de um sistema em termos de componentes, suas interfaces e dependências possibilitando a visualização e organização dos relacionamentos entre as partes modulares de um sistema de software. O diagrama é apresentado na Figura 2.

Figura 2 - Diagrama de Componentes



Fonte: Elaborado pelo autor (2025)

O detalhamento dos componentes apresentados na Figura 2 é descrito a seguir:

- Frontend (React.js)
  - Descrição: Interface de usuário para interação e visualização de dados.
  - Responsabilidades: Exibir dados em tempo real, permitir login de usuários, gerar relatórios.
- Backend WebService (Java)
  - Descrição: Processamento de lógica de negócios e comunicação com o banco de dados.
  - Responsabilidades: Autenticar usuários, processar dados recebidos dos dispositivos IoT, gerar endpoints para o frontend.
- Dispositivo IoT (ESP8266)
  - Descrição: Hardware que coleta e transmite dados de sensores.
  - Responsabilidades: Coletar dados de temperatura e umidade e transmitir para o servidor.
- Banco de Dados (MySQL)
  - Descrição: Armazenamento de dados de sensores e informações de usuários.

- Responsabilidades: Persistir dados de temperatura e umidade, armazenar dados de autenticação.

Para que alguns dos componentes sejam executados e se comuniquem entre si, foi utilizada a ferramenta Jelastic, um serviço em nuvem que fornece plataforma multinuvem baseada em tecnologia de contêiner. Nela, é possível criar contêineres de diversas tecnologias e disponibilizá-las para utilização em nuvem. Com o Jelastic, foram criados os seguintes ambientes e contêineres:

- Ambiente **webapp-iot**
  - Contêiner **Node.js** - utilizado para hospedar o componente Frontend (React.js).
- Ambiente **webservice-db-iot**
  - Contêiner **Apache** - utilizado para hospedar o componente Backend WebService (Java);
  - Contêiner **MySQL** - utilizado para hospedar o componente Banco de Dados (MySQL);
  - Contêiner **Maven** - utilizado para executar a *build* do componente Backend WebService (Java).

A Figura 3 apresenta a tabela de ambientes e contêineres criados no Jelastic.

Figura 3 - Ambientes e contêineres criados no Jelastic

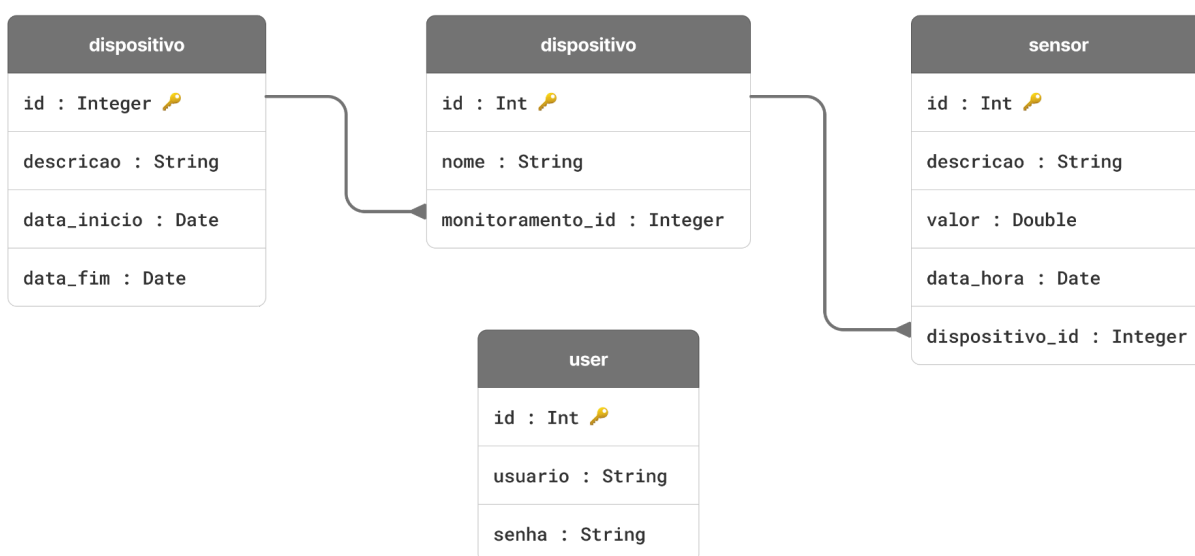
Name	Status
<b>webapp-iot</b> webapp-iot.sp1.br.savencloud.net.br	Running
<b>Application Servers</b> Node.js 23.11.0	
> Node ID: 228474	
> Deployments	
<b>webservice-db-iot</b> webservice-db-iot.sp1.br.savencloud.net.br	Running
<b>Application Servers</b> Tomcat 9.0.104	
> Node ID: 226416	
> Deployments	
<b>SQL Databases</b> MySQL CE 9.2.0	
> Node ID: 226415	
<b>Build Node</b> Maven 3.9.9	
> Node ID: 226551	
> Projects	

Fonte: Elaborado pelo autor (2025)

### 4.2.3 Diagrama de tabelas e relacionamentos (DTR)

Trata-se de uma representação visual que descreve a estrutura do banco de dados relacional utilizado neste projeto (incluindo suas tabelas, atributos e relacionamentos) objetivando apresentar o planejamento da documentação do banco de dados para posterior implementação. A Figura 4 apresenta o DTR.

Figura 4 - Diagrama de Tabelas e Relacionamentos



Fonte: Elaborado pelo autor (2025)

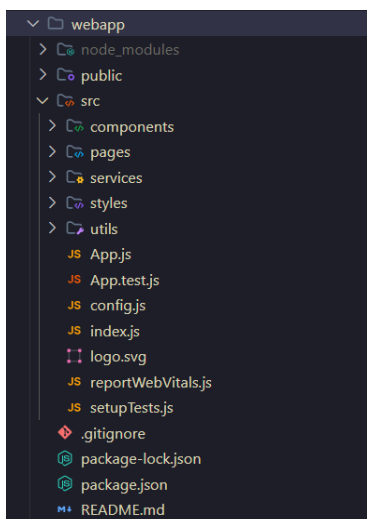
## 4.3 Programação e testes

### 4.3.1 Programação

A seguir são apresentados os principais trechos de código fonte (mais representativos) desenvolvidos durante o projeto, ilustrando a estruturação lógica e organização adotada na implementação do sistema., onde é possível visualizar como o código foi organizado próximo ao padrão arquitetural MVC (Model-View-Controller), evidenciando a separação entre as camadas de interface, lógica de negócios e persistência dos dados. Além disso, são destacadas partes do código responsáveis pela comunicação entre o dispositivo IoT e o sistema central, bem como a rotina responsável por receber, tratar e armazenar os dados coletados pelos sensores.

A Figura 5 apresenta a estrutura de código do Frontend (React.js), que segue um padrão semelhante ao padrão arquitetural MVC, onde as *pages* fazem o papel de exibição dos dados, e os *services* fazem o papel de controladores de processamento entre os demais componentes.

Figura 5 - Estrutura de código do Frontend (React.js)



Fonte: Elaborado pelo autor (2025)

A Figura 6 apresenta o código do componente principal (App) do Frontend, onde são definidos a barra de navegação e as rotas de acesso da aplicação.

Figura 6 - Código inicial do frontend

```

1 import React from 'react';
2 import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
3 import Navbar from './components/Navbar';
4 import PrivateRoute from './components/PrivateRoute';
5 import Dashboard from './pages/Dashboard';
6 import Sensores from './pages/Sensores';
7 import './styles/App.css';
8 import Login from './components/Login';
9
10 function App() {
11   return (
12     <Router>
13       <div className="App">
14         <Navbar />
15         <div className="App-body">
16           <Routes>
17             <Route path="/" element={
18               <PrivateRoute element={<Dashboard />} />
19             } />
20             <Route path="/sensores" element={
21               <PrivateRoute element={<Sensores />} />
22             } />
23             <Route path="/login" element={<Login />} />
24           </Routes>
25         </div>
26       </div>
27     </Router>
28   )
29 }
30
31 export default App;

```

Fonte: Elaborado pelo autor (2025)

A Figura 7 apresenta o código da página Dashboard, responsável pela exibição dos gráficos de valores de temperatura e umidade capturados pelos sensores DHT22.

Figura 7 - Código da página Dashboard

```

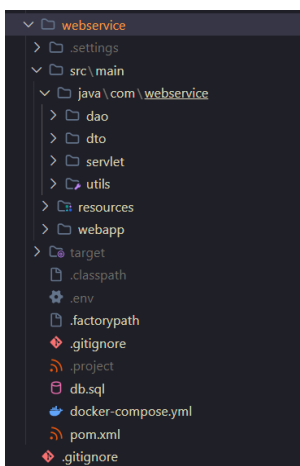
5  const Dashboard = () => {
6
7      const [sensorData, setSensorData] = useState([]);
8      const [loading, setLoading] = useState(true);
9      const [error, setError] = useState(null);
10
11      useEffect(() => {
12          const fetchSensors = async () => {
13              try {
14                  const data = await getAllSensors();
15                  setSensorData(data);
16              } catch (error) {
17                  setError('Failed to fetch sensors: ' + error);
18              } finally {
19                  setLoading(false);
20              }
21          }
22
23          fetchSensors();
24      }, []);
25
26      if (loading) return <div>Carregando...</div>;
27      if (error) return <div>[error]</div>;
28
29      return (
30          <div>
31              <h1>Dashboard</h1>
32              <SensorChart sensorData={sensorData} />
33          </div>
34      );
35  }
36
37  export default Dashboard;

```

Fonte: Elaborado pelo autor (2025)

A Figura 8 apresenta a estrutura de código do Backend (Java), que segue um padrão semelhante ao padrão arquitetural MVC, onde os DAO (Data Access Object) fazem o papel dos modelos de dados, os DTO (Data Transfer Object) fazem o papel de controladores de processamento entre os demais componentes e os *servlets* fazem o papel de comunicação com o Banco de Dados (MySQL).

Figura 8 - Estrutura de código do Backend (Java)



Fonte: Elaborado pelo autor (2025)



A Figura 9 apresenta o código do script de criação e população do Banco de Dados (MySQL), onde são criadas as tabelas Monitoramento, Dispositivo, Sensor e User.

Figura 9 - Código do script de criação do Banco de Dados (MySQL)

```

1  DROP DATABASE IF EXISTS webservice_db;
2  CREATE DATABASE webservice_db;
3  USE webservice_db;
4
5  CREATE TABLE monitoramento (
6      id INT PRIMARY KEY AUTO_INCREMENT,
7      descricao VARCHAR(255) NOT NULL,
8      data_inicio DATETIME NOT NULL,
9      data_fim DATETIME NOT NULL
10 );
11
12 CREATE TABLE dispositivo (
13     id INT PRIMARY KEY AUTO_INCREMENT,
14     nome VARCHAR(100) NOT NULL,
15     monitoramento_id INT,
16
17     FOREIGN KEY (monitoramento_id) REFERENCES monitoramento(id) ON DELETE SET NULL
18 );
19
20 CREATE TABLE sensor (
21     id INT PRIMARY KEY AUTO_INCREMENT,
22     descricao VARCHAR(255) NOT NULL,
23     valor DOUBLE(8,2) NOT NULL,
24     data_hora DATETIME NOT NULL,
25     dispositivo_id INT,
26
27     FOREIGN KEY (dispositivo_id) REFERENCES dispositivo(id) ON DELETE SET NULL
28 );
29
30 CREATE TABLE user (
31     id INT PRIMARY KEY AUTO_INCREMENT,
32     usuario VARCHAR(100) NOT NULL,
33     senha VARCHAR(255) NOT NULL
34 );

```

Fonte: Elaborado pelo autor (2025)

A Figura 10 apresenta o código da classe responsável pela conexão do Backend (Java) com o Banco de Dados (MySQL).

Figura 10 - Código de conexão do Backend com o Banco de Dados

```

1  package com.webservice.utils;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class DatabaseConnection {
8
9      private static final String URL = "jdbc:mysql://191.243.197.149:3306/webservice_db_test";
10     private static final String USER = "root";
11     private static final String PASSWORD = "LSRnh100113";
12
13     public static Connection getConnection() throws SQLException, ClassNotFoundException {
14         try {
15             Class.forName(className:"com.mysql.cj.jdbc.Driver");
16             return DriverManager.getConnection(URL, USER, PASSWORD);
17         } catch (SQLException e) {
18             throw new SQLException(reason:"Error on database connection: ", e);
19         } catch (ClassNotFoundException e) {
20             throw new ClassNotFoundException(s:"Error on Class.forName: ", e);
21         }
22     }
23 }

```

Fonte: Elaborado pelo autor (2025)

A Figura 11 apresenta o código do modelo DTO Sensor (mesma função de Model), responsável por guardar os atributos necessários para registrar os dados de sensores. O mesmo ocorre para os DTO Monitoramento, Dispositivo e User.

**Figura 11 - Código do DTO Sensor**

```

1 package com.webservice.dto;
2
3 import lombok.Data;
4
5 @Data
6 public class Sensor {
7     private Integer id;
8     private String descricao;
9     private Double valor;
10    private String dataHora;
11    private Dispositivo dispositivo;
12
13    public Sensor(Integer id, String descricao, Double valor,
14                  String dataHora, Dispositivo dispositivo) {
15        this.id = id;
16        this.descricao = descricao;
17        this.valor = valor;
18        this.dataHora = dataHora;
19        this.dispositivo = dispositivo;
20    }
21
22    public Sensor(String descricao, Double valor, String dataHora,
23                  Dispositivo dispositivo) {
24        this.descricao = descricao;
25        this.valor = valor;
26        this.dataHora = dataHora;
27        this.dispositivo = dispositivo;
28    }
29 }

```

Fonte: Elaborado pelo autor (2025)

A Figura 12 apresenta o código do servlet Serviços, responsável por administrar a comunicação do Dispositivo ESP8266 com os DTO (modelos) do Backend Webservice (Java).

**Figura 12 - Código do servlet Serviços**

```

28 public class ServicosServlet extends HttpServlet {
29
30     private MonitoramentoDAO monitoramentoDAO;
31     private DispositivoDAO dispositivoDAO;
32     private SensorDAO sensorDAO;
33     private Gson gson;
34
35     private final String[] valoresNotSensor = {
36         "monitoramento",
37         "dispositivo",
38         "datahora"
39     };
40
41     @Override
42     public void init() throws ServletException { ... }
43
44     protected void doGet(HttpServletRequest request, HttpServletResponse response) { ... }
45
46     protected void doPost(HttpServletRequest request, HttpServletResponse response) { ... }
47
48     protected void execute(HttpServletRequest request, HttpServletResponse response)
49         throws ServletException, IOException {
50
51         try {
52             response.setContentType("application/json");
53             String valores = request.getParameter(name="valores");
54
55             if (valores == null) {
56                 response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
57                 throw new Exception(message:"Parameter 'valores' not found.");
58             }
59
60             Map<String, String> valoresMap = new HashMap<String, String>();

```

Fonte: Elaborado pelo autor (2025)

A Figura 13 apresenta o código do servlet Sensor, responsável por administrar a comunicação entre o Frontend (React.js), o DAO e DTO (modelo) Sensor.

Figura 13 - Código do servlet Sensor

```

24 @WebServlet("/api/sensores/")
25 public class SensorServlet extends HttpServlet {
26
27     private SensorDAO sensorDAO;
28     private DispositivoDAO dispositivoDAO;
29     private Gson gson;
30
31     @Override
32     public void init() throws ServletException {
33         try {
34             Connection connection = DatabaseConnection.getConnection();
35             sensorDAO = new SensorDAO(connection);
36             dispositivoDAO = new DispositivoDAO(connection);
37             gson = new Gson();
38             System.out.println("SensorServlet.init(): Database connection " +
39                 "established successfully.");
40         } catch (SQLException | ClassNotFoundException e) {
41             throw new ServletException("Database connection error on " +
42                 "SensorServlet.init: ", e);
43         }
44     }
45
46     @Override
47     public void doGet(HttpServletRequest request, HttpServletResponse response)
48         throws ServletException, IOException {
49
50         String pathInfo = request.getPathInfo();
51         response.setContentType("application/json");
52
53         if (pathInfo != null && pathInfo.matches("^/\\d+$")) {
54             Integer id = Integer.parseInt(pathInfo.substring(1));
55             try {
56                 Sensor sensor = sensorDAO.getSensorById(id);

```

Fonte: Elaborado pelo autor (2025)

A Figura 14 apresenta o código do Dispositivo ESP8266, responsável por estabelecer a conexão com a internet, iniciar a leitura do sensor DHT22, e enviar os valores capturados pelo sensor para o Backend Webservice (Java).

Figura 14 - Código do Dispositivo ESP8266

```

11 // Configurações DHT22
12 #define DHTPIN D5
13 #define DHTTYPE DHT22
14 DHT dht(DHTPIN, DHTTYPE);
15 float Temperature = 0;
16 float humidity = 0;
17
18 void setup() {
19     Serial.begin(115200);
20     //start DHT22
21     dht.begin();
22 }
23
24 void loop() {
25
26     if(lerDados()){
27         conectarWifi();
28         enviarDadosServidor();
29     }else{
30         Serial.println("Falha na leitura do sensor DHT22!");
31     }
32
33     // Aguarda 2 minutos antes de executar novamente
34     delay(600000);
35 }
36
37 void conectarWifi(){
38     WiFi.begin(ssid, password);
39     while (WiFi.status() != WL_CONNECTED) {
40         delay(500);
41         Serial.println("Conectando à rede Wifi...");
42     }
43     Serial.println("Conectado à rede Wifi!");
44 }
45
46 boolean lerDados(){
47     // Lê a temperatura e umidade
48     Temperature = dht.readTemperature();
49     Serial.print("Temperatura: ");Serial.println(Temperature);
50     humidity = dht.readHumidity();
51     Serial.print("Umidade: ");Serial.println(humidity);
52     boolean retorno = true;

```

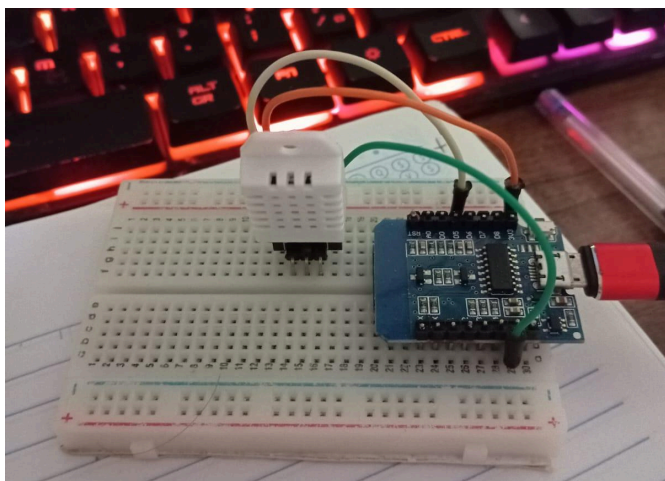
Fonte: Elaborado pelo autor (2025)

#### 4.3.2 Testes

Com o objetivo de validar o funcionamento do sistema desenvolvido, foi realizada uma série de testes práticos envolvendo todo o fluxo desde a coleta dos dados no dispositivo IoT até a sua correta exibição no sistema web. Inicialmente, foi montado o circuito com o sensor DHT conectado ao ESP, responsável pela leitura dos dados de temperatura e umidade (registrado com uma foto feita com o celular). Em seguida, foi realizado o envio dos dados coletados ao servidor, comprovado por meio de prints do Serial Monitor do Arduino IDE. No ambiente web, foram capturadas imagens demonstrando a tabela sendo populada com os dados recebidos, além das telas do sistema web — como a tela de login e navegação — evidenciando que os dados são armazenados e visualizados corretamente pelo usuário.

A Figura 15 apresenta o circuito com o sensor DHT22 conectado ao Dispositivo ESP8266.

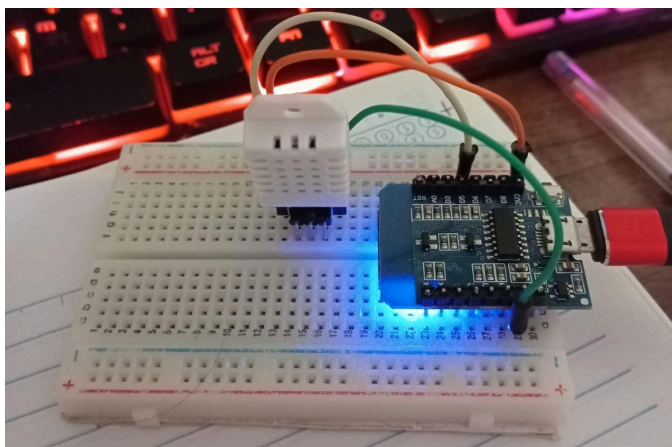
Figura 15 - Circuito montado com ESP8266 e DHT22)



Fonte: Elaborado pelo autor (2025)

A Figura 16 apresenta o mesmo circuito após o código do Dispositivo ESP8266 (Figura 14) ser executado.

Figura 16 - ESP8266 após ser executado



Fonte: Elaborado pelo autor (2025)

A Figura 17 apresenta a saída do ESP8266 após executar o código programado, mostrando que o sensor capturou os valores de temperatura e umidade do ambiente, e as enviou para o Backend Webservice (Java).

Figura 17 - Saída do ESP8266 no momento da conexão com o Backend (Java)

```
20:42:42.814 -> http://192.168.1.27:8181/WebService/servicos?valores=temperatura=26.20;umidade=64.00
20:42:49.852 -> Temperatura: 26.20
20:42:49.852 -> Umidade: 63.90
```

Fonte: Elaborado pelo autor (2025)

A Figura 18 apresenta a tela de Login, onde o usuário deve informar o seu usuário e sua senha para ter acesso às demais telas da aplicação.

Figura 18 - Tela de Login



The login screen features a dark header with two tabs: 'Dashboard' and 'Histórico dos Sensores'. Below the header, the title 'Login' is displayed. There are two input fields: 'Usuário:' and 'Senha:'. A green 'Login' button is positioned below the password field.

Fonte: Elaborado pelo autor (2025)

A Figura 19 apresenta a tela de Dashboard, onde é possível visualizar os gráficos de valores de temperatura e umidade capturados pelos sensores DHT22.

Figura 19 - Tela do Dashboard



Fonte: Elaborado pelo autor (2025)

A Figura 20 apresenta a tela de Histórico dos Sensores, onde é possível visualizar o histórico de valores de temperatura e umidade capturados pelos sensores, além da data e hora da captura, e do Monitoramento e do Dispositivo vinculados ao sensor.

Figura 20 - Tela do Histórico dos Sensores

Dashboard

Histórico dos Sensores

Lista de Sensores

Monitoramento	Dispositivo	Descrição	Valor	Data e Hora
Monitoramento 01	Dispositivo 01	temperatura	25.61	2025-04-01 00:00:00
Monitoramento 01	Dispositivo 01	temperatura	27.96	2025-04-01 00:10:00
Monitoramento 01	Dispositivo 01	temperatura	22.97	2025-04-01 00:20:00
Monitoramento 01	Dispositivo 01	temperatura	20.98	2025-04-01 00:30:00
Monitoramento 01	Dispositivo 01	temperatura	26.00	2025-04-01 00:40:00
Monitoramento 01	Dispositivo 01	temperatura	27.03	2025-04-01 00:50:00
Monitoramento 01	Dispositivo 01	temperatura	27.15	2025-04-01 01:00:00
Monitoramento 01	Dispositivo 01	temperatura	24.66	2025-04-01 01:10:00
Monitoramento 01	Dispositivo 01	temperatura	21.87	2025-04-01 01:20:00
Monitoramento 01	Dispositivo 01	temperatura	25.36	2025-04-01 01:30:00
Monitoramento 01	Dispositivo 01	temperatura	21.19	2025-04-01 01:40:00
Monitoramento 01	Dispositivo 01	temperatura	29.85	2025-04-01 01:50:00
Monitoramento 01	Dispositivo 01	temperatura	25.69	2025-04-01 02:00:00
Monitoramento 01	Dispositivo 01	temperatura	28.92	2025-04-01 02:10:00

Fonte: Elaborado pelo autor (2025)

## 5 CONSIDERAÇÕES FINAIS

O desenvolvimento do sistema web para gerenciamento de dispositivos IoT mostrou-se tecnicamente viável e economicamente acessível, graças ao uso de tecnologias amplamente difundidas e de baixo custo. A integração entre o hardware (ESP32 e DHT22) e o software validou o fluxo completo do sistema, desde a coleta de dados até sua exibição em tempo real e armazenamento seguro. A utilização da metodologia Scrum contribuiu significativamente para a organização do projeto, permitindo flexibilidade e correções rápidas durante o desenvolvimento.

Os testes realizados comprovaram a eficiência do sistema na transmissão de dados com baixa latência e seu correto armazenamento no banco de dados MySQL. A interface desenvolvida com React.js destacou-se pela usabilidade, responsividade e compatibilidade em diferentes dispositivos e navegadores. Além disso, a funcionalidade de geração de relatórios históricos reforçou a utilidade do sistema para análises posteriores, agregando valor às informações coletadas pelos sensores.

Como sugestão para trabalhos futuros, é possível evoluir o sistema com funcionalidades como autenticação avançada, notificações em tempo real, gráficos interativos e suporte a múltiplos dispositivos IoT conectados simultaneamente. Também é promissora a implementação de regras de alerta automáticas com base em condições pré-definidas, aumentando a aplicabilidade do sistema em ambientes industriais e residenciais. Assim, este trabalho apresenta uma solução prática, escalável e de código aberto que contribui para a disseminação do uso de tecnologias IoT em diferentes contextos.



## REFERÊNCIAS

ADAMS, A.; BISCH, N.; NEWCOMER, J. **Fullstack React: The Complete Guide to ReactJS and Friends**. Fullstack.io, 2021.

ALEXANDER, M. **The React Ecosystem: Tools and Libraries for Modern Web Development**. Springer, 2022.

CHANG, C.; SRIRAMA, S. N.; BUYYA, R. **Internet of Things (IoT) and New Computing Paradigms**. Fog And Edge Computing, [S.L.], p. 1-23, 11 jan. 2019. Wiley. Disponível em: <http://dx.doi.org/10.1002/9781119525080.ch1>. Acesso em: 9 mar. 2025.

CROCKFORD, D. **JavaScript: The Good Parts**. O'Reilly Media, 2018. Disponível em: <https://www.oreilly.com/library/view/javascript-the-good/9780596517748/>. Acesso em: 10 mar. 2025.

FERNANDEZ, J. **Tamanho do mercado de gerenciamento de dados IoT e análise de ações – Tendências e previsões de crescimento (2024 – 2029)**. Disponível em: <https://www.mordorintelligence.com/pt/industry-reports/iot-data-management-market>. Acesso em: 01 abr. 2025.

FLANAGAN, D. **JavaScript: The Definitive Guide**. 7. ed. O'Reilly Media, 2020. Disponível em: <https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/>. Acesso em: 16 abr. 2025.

GONZÁLEZ, J. L.; MARTÍNEZ, P.; GARCÍA, F. IoT-Based Web Systems: Challenges and Opportunities. **Journal of Internet of Things and Web Development**, v. 12, n. 2, p. 45-62, 2022. Disponível em: <https://www.jiotwd.com/article/iot-web-systems>. Acesso em: 16 abr. 2025.

GOODRICH, M. T.; TAMASSIA, R.; MOUNT, D. M. **Data Structures and Algorithms in JavaScript**. Wiley, 2021. Disponível em: <https://www.wiley.com/en-us/Data+Structures+and+Algorithms+in+JavaScript-p-9781119701234>. Acesso em: 16 abr. 2025.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of Things (IoT): a vision, architectural elements, and future directions. **Future Generation Computer Systems**, [S.L.], v. 29, n. 7, p. 1645-1660, set. 2013. Elsevier BV. Disponível em: <http://dx.doi.org/10.1016/j.future.2013.01.010>. Acesso em: 10 abr. 2025.

KIRK, A. **Data Visualization: A Structured Design Approach to Equip You with the Knowledge of How to Successfully Accomplish Any Data Visualization Challenge Efficiently and Effectively**. Birmingham, Uk: Packt Press, 2012. 189 p. Disponível em: <https://openlibrary.telkomuniversity.ac.id/pustaka/172826/data-visualization-a-successful-design-process.html>. Acesso em: 25 mar. 2025.

LARA, J. E.; REIS, L. J.; TISSOT-LARA, T. A.; SILVA, A. O. Admirável mundo novo na perspectiva da tríade: internet das coisas, pessoas e mercados. **Perspectivas em Ciência da**

**Informação**, [S.L.], v. 26, n. 2, p. 124-150, jun. 2021. FapUNIFESP (SciELO). Disponível em: <http://dx.doi.org/10.1590/1981-5344/3825>. Acesso em: 23 mar. 2025.

LEFFINGWELL, D. **Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise** (Agile Software Development Series). Boston, Usa: Addison-Wesley Educational Publishers Inc, 2011. p. 560. Disponível em: <https://www.oreilly.com/library/view/agile-software-requirements/9780321685438/>. Acesso em 23 mar. 2025.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Guidelines on Security and Privacy in Public Cloud Computing**. 2011. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf>. Acesso em: 25 mar. 2025.

OLIVEIRA, A. P.; COSTA, L. F. **Integração de Sensores de Baixo Custo em Sistemas IoT para Monitoramento Ambiental**. Anais do Congresso Brasileiro de Automação e Controle, São Paulo, 2024. Disponível em: <https://www.cbac2024.org>. Acesso em: 16 abr. 2025.

OLIVEIRA, R. T. **Java no Desenvolvimento de Sistemas Industriais**. FAM, 2023. Disponível em: <https://fam-cati2023.pdf>. Acessado em: 16 abr. 2025.

ROSA, C. M.; SOUZA, P. A. R.; SILVA, J. M. Inovação em saúde e internet das coisas (IoT): um panorama do desenvolvimento científico e tecnológico. **Perspectivas em Ciência da Informação**, [S.L.], v. 25, n. 3, p. 164-181, jul. 2020. FapUNIFESP (SciELO). Disponível em: <http://dx.doi.org/10.1590/1981-5344/3885>. Acesso em: 16 abr. 2025.

SANTOS, J. C. R.; SILVA, A. L. **Gerenciamento de Dados Coletados de Sensores IOT Utilizando Computação em Nuvem**. Coletânea da Pós-Graduação em Telecomunicações: prédios inteligentes - Volume II, [S.L.], p. 66-84, 2023. Editora Científica Digital. Disponível em: <http://dx.doi.org/10.37885/230412674>. Acesso em: 16 abr. 2025.

SATHISH, J. **ESP32 COOKBOOK: ESP8266, Arduino Coding, Example Code, IoT Project, Sensors, Esp32 Startup**. 1: Ebook Kindle, 2021. 262 p. Disponível em: <https://www.amazon.com/ESP32-COOKBOOK-ESP8266-Arduino-Example/dp/B09BZDSNL7>. Acesso em: 21 mar. 2025.

SILVA, A. P. et al. Web Accessibility Standards: Integrating HTML and ARIA for Inclusive Design. **Journal of Universal Access in the Information Society**, v. 18, n. 2, p. 245-260, 2022. Disponível em: <https://www.researchgate.net/journal/Universal-Access-in-the-Information-Society-1615-5297>. Acesso em: 20 mar. 2025.

SILVA, J. C.; SANTOS, R. M. Sensores Digitais de Temperatura e Umidade: Uma Revisão sobre o Sensor DHT22 . **Revista de Tecnologia em Sistemas Embarcados**, v. 15, n. 3, p. 45-60. 2023.

SILVA, J. S. **Programação Orientada a Objetos em Java**. DataCamp, 2023. Disponível em: <https://www.datacamp.com/java-oop>. Acessado em: 16 abr. 2025.

SOARES, W. O.; FARIA, N. C. S. **Gerenciamento remoto da climatização de data center**

**via IoT.** 2021. 13 f. Monografia (Especialização) - Curso de Engenharia Elétrica, Puc Goiás, Goiânia, 2021. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/handle/123456789/2802>. Acesso em: 25 mar. 2025.

STOYANOV, S. **React: Up & Running.** O'Reilly Media, 2016. Disponível em: <https://www.oreilly.com/library/view/react-up/9781492051459/>. Acesso em 27 mar. 2025.

WALKE, J. React: A declarative approach to building user interfaces. **ACM SIGWEB Newsletter**, v. 25, n. 3, p. 12-20, 2015. Disponível em: <https://doi.org/10.1145/1234567.1234568>. Acessado em: 16 abr. 2025.

ZAKAS, N. C. **Understanding ECMAScript 6:** The Definitive Guide for JavaScript Developers. No Starch Press, 2019. Disponível em: <https://nostarch.com/understandings6>. Acesso em: 16 abr. 2025.