

IDENTIFICAÇÃO

NOME DO ACADÊMICO: ADEMILTON AMARO MARIANO

CÓDIGO DE MATRÍCULA 2021010782

TÍTULO DO TRABALHO: Proposta de uma Plataforma para Gerenciamento de Dispositivos na Internet das Coisas

ÁREA DE PESQUISA: Tecnologia

E-MAIL DO ACADÊMICO: ademilton.mariano@gmail.com

TELEFONE COMERCIAL:

TELEFONE RESIDENCIAL:

TELEFONE CELULAR: 47997617608

ENDEREÇO RESIDENCIAL: Rua Artelina Vassoler, 71, Perequê, Porto Belo - SC

*Considerando verídicas as Informações fornecidas neste formulário,
encaminho a Proposta de PI-I (PPC-2019) para avaliação*

ASSINATURA DO ACADÊMICO:

DATA: 27/11/2023

Proposta de uma Plataforma para Gerenciamento de Dispositivos na Internet das Coisas

Ademilton Amaro Mariano

RESUMO

Este trabalho aborda o tema do gerenciamento e customização de APIs para dispositivos Internet das Coisas (IoT), que são dispositivos que disponibilizam dados a partir de diferentes tipos de sensores e medições. A Internet das Coisas é descrita como a conexão inteligente de objetos à internet, permitindo a coleta e compartilhamento de dados. A variedade e o volume de dados gerados pelos dispositivos IoT requerem uma estrutura eficiente para a coleta e processamento dessas informações. As APIs REST são destacadas como conjuntos de definições e protocolos que facilitam a comunicação entre provedores e usuários de informações. Nesse contexto, este trabalho visa analisar as características necessárias para uma plataforma de coleta de dados de IoT e propor a implantação de uma solução que atenda a essas características. As plataformas IoT são mencionadas como soluções que simplificam a integração e o gerenciamento de dispositivos, permitindo o controle centralizado e o processamento eficiente de dados. A implementação de uma solução adequada de coleta e processamento de dados de IoT permite a obtenção de informações valiosas e serviços otimizados, facilitando a tomada de decisões embasadas e a otimização de processos.

Palavras-chave: API; Internet das Coisas; Software; Dispositivos; Coleta de Dados; Processamento de Dados.

SUMÁRIO

1	INTRODUÇÃO	6
2	OBJETIVOS	7
2.1	Objetivo Geral	7
2.2	Objetivos Específicos	7
3	ESCOPO DO PROJETO	8
4	METODOLOGIA	9
5	RECURSOS	11
6	DESENVOLVIMENTO	12
6.1	Internet das Coisas	12
6.2	APIs REST	14
6.3	Propósito do Projeto	14
6.4	Plataformas	17
6.4.1	Plataforma ThingsBoard	18
6.4.2	Plataforma Blink	18
6.4.3	Plataforma ThingSpeak	19
6.4.4	Comparaçāo das Plataformas	20
6.4.5	Escolha da Plataforma para Implementaçāo	21
6.5	Protótipo de Implementaçāo	21
6.5.1	Instalação	21
6.5.2	Funcionalidades	26
7	APLICAÇĀO	28
7.1	Teste com o Protocolo HTTP	31
7.2	Teste com o Protocolo MQTT	33
7.3	Testes Externos de Validaçāo	36
7.4	Consumindo Dados da plataforma	40
8	RESULTADOS ALCANÇADOS	41

9	CONCLUSÕES E RECOMENDAÇÕES	42
10	REFERÊNCIAS	43
	APÊNDICE A – Cronograma	45
	ANEXO A – Passo a Passo dos Testes Realizados	46
	ANEXO B - Código para comunicação por HTTP	53
	ANEXO C - Código para comunicação por MQTT	56
	ANEXO D - Demonstração de consumo de dados na API	58

1. INTRODUÇÃO

O avanço da tecnologia tem proporcionado a criação de dispositivos cada vez mais inteligentes, capazes de coletar e transmitir dados em tempo real. Esses dispositivos estão presentes em diversos segmentos, e a quantidade de dados gerados por eles é enorme. Um comparativo no estudo (Arun Solanki, Adarsh Kumar, 2021) indica que a quantidade de dispositivos inteligentes conectados cresce exponencialmente, mais veloz do que a quantidade de pessoas no mundo. No entanto, para aproveitar plenamente o potencial desses dispositivos conectados, é imprescindível contar com APIs que sejam capazes de gerenciar e customizar a comunicação entre eles e os sistemas que os utilizam.

As APIs (Interfaces de Programação de Aplicativos) desempenham um papel fundamental nesse cenário, atuando como uma ponte de comunicação eficiente entre dispositivos IoT e os sistemas que consomem seus dados. Uma API bem projetada e customizada permite o compartilhamento de informações de forma segura e a interação inteligente entre dispositivos.

Diante dessa necessidade, este trabalho tem como objetivo explorar as melhores práticas de gerenciamento e customização de APIs para dispositivos IoT. A compreensão dessas práticas é crucial para maximizar o potencial dessa tecnologia inovadora, permitindo uma integração eficiente e segura dos dispositivos IoT com os sistemas que os utilizam. Através do gerenciamento e customização adequados das APIs, é possível promover a compatibilidade entre dispositivos, facilitar a troca de informações de forma confiável e promover interações inteligentes entre os dispositivos conectados. Essas práticas contribuem para a criação de soluções mais inteligentes, conectadas e adaptáveis, impulsionando a eficiência operacional, a segurança e a usabilidade dos dispositivos IoT.

2. OBJETIVOS

2.1. Objetivo Geral

O objetivo principal deste trabalho é propor uma plataforma que consiga gerenciar com eficiência dispositivos de Internet das Coisas.

2.2. Objetivos Específicos

- Caracterizar o tema Internet das Coisas, suas tecnologias e ferramentas;
- Identificar os requisitos para uma plataforma de coleta, processamento e visualização de dados de dispositivos da Internet das Coisas;
- Analisar ferramentas correlatas;
- Implementar e implantar uma plataforma de coleta, processamento e visualização de dados para soluções de Internet das Coisas.

3. ESCOPO DO PROJETO

O escopo deste projeto abrange o desenvolvimento de uma plataforma para coleta, processamento e visualização de dados na Internet das Coisas (IoT). A plataforma será projetada para receber e integrar dados provenientes de dispositivos IoT, por meio de protocolos de comunicação compatíveis. A plataforma também fornecerá recursos de visualização e monitoramento dos dados, por meio de interfaces amigáveis e personalizáveis. No entanto, é importante ressaltar que a implementação prática da plataforma não está contemplada neste trabalho, sendo o foco principal a análise das características necessárias e a proposta de uma solução conceitual para a coleta, processamento e visualização de dados na IoT.

4. METODOLOGIA

Para abordar o problema específico relacionado à coleta de dados de dispositivos IoT, este trabalho adotará uma natureza aplicada e uma abordagem qualitativa. Será realizada uma pesquisa exploratória, por meio de pesquisa bibliográfica, levantamento de requisitos e análise de casos semelhantes. A coleta de dados ocorrerá de forma não estruturada, utilizando o ambiente natural como fonte direta. Os procedimentos incluirão uma pesquisa bibliográfica abrangente para identificar conceitos, tecnologias e melhores práticas relacionadas à coleta de dados da IoT. Também serão conduzidas entrevistas com especialistas e análise de casos de sucesso para obter insights relevantes sobre o problema em questão.

A Metodologia deste trabalho será dividida em 6 etapas que são:

Etapa 1: Pesquisa e Análise

Atividade 1.1: Realizar levantamento bibliográfico sobre o gerenciamento e customização de APIs para dispositivos IoT.

Duração: 7 dias.

Etapa 2: Levantamento de Requisitos

Atividade 2.1: Identificar os requisitos funcionais e não funcionais da plataforma de coleta, processamento e visualização de dados na IoT.

Duração: 5 dias.

Etapa 3: Análise de Protocolos de Comunicação

Atividade 3.1: Estudar os principais protocolos de comunicação utilizados.

Duração: 10 dias.

Etapa 4: Desenvolvimento do Protótipo

Atividade 4.1: Implementar um protótipo funcional da plataforma, focando nas

principais funcionalidades de coleta, processamento e visualização de dados.

Duração: 20 dias.

Etapa 5: Instalação e Implementação da Plataforma

Atividade 5.1: Instalar e configurar a plataforma no servidor local.

Atividade 5.2: Configurar e conectar um dispositivo IOT a plataforma.

Atividade 5.3: Realizar testes de comunicação e consumo de dados com os protocolos analisados.

Duração: 90 dias.

Etapa 6: Documentação e Elaboração do Relatório

Atividade 6.1: Documentar todas as etapas do projeto e elaborar o relatório final, descrevendo a proposta da plataforma, as metodologias utilizadas e os resultados obtidos.

Duração: 30 dias.

5. RECURSOS

Recursos de Hardware:

Recurso	Descrição	Qtd	Disp	Local
Computador pessoal	Processador: Intel(R) Core(TM) i5-8265U CPU 1.60GHz 1.80 GHz. Memória RAM: 8GB	01	Sim	Casa
Dispositivo IoT	Placa ESP32 DevKit	01	Sim	Casa
Sensor de temperatura	Sensor DS18B20	01	Sim	Casa

Recursos de Software:

Recurso	Descrição	Qtd	Disp	Local
Sistema operacional	Windows 11	01	Sim	Máquina Local
Ferramenta de documentação	Microsoft Word ou Google Docs	01	Sim	https://docs.google.com/?hl=pt-BR
Plataforma	ThingsBoard Community Edition	01	Sim	https://thingsboard.io/
Aplicativo de Cliente de API	Insomnia: Utilizado para fazer requisições HTTP	01	Sim	https://insomnia.rest/download
IDE para programar o dispositivo IoT	Arduino IDE	01	Sim	Support the Arduino IDE Arduino

Outros Recursos:

Recurso	Descrição	Qtd	Disp	Local
Acesso à Internet	Velocidade de conexão: 500 Mbps. Conexão estável.	01	sim	Casa

6. DESENVOLVIMENTO

6.1 Internet das coisas

A IoT (Internet das Coisas) é conhecida como a conexão de qualquer objeto à internet, esses objetos, é um conceito em que todos esses objetos estejam conectados de uma forma inteligente diminuindo o custo da conectividade (MARTINS et al, 2020).

A proposta da IoT consiste na ideia de que “coisas” (objetos) façam o gerenciamento de uma vasta quantidade de dados, coletando, armazenando e compartilhando os mesmos. Que uma vez processados e analisados podem gerar uma enorme quantidade de informações e serviços, operando de maneira independente, por meio da internet (FERRASI, 2023).

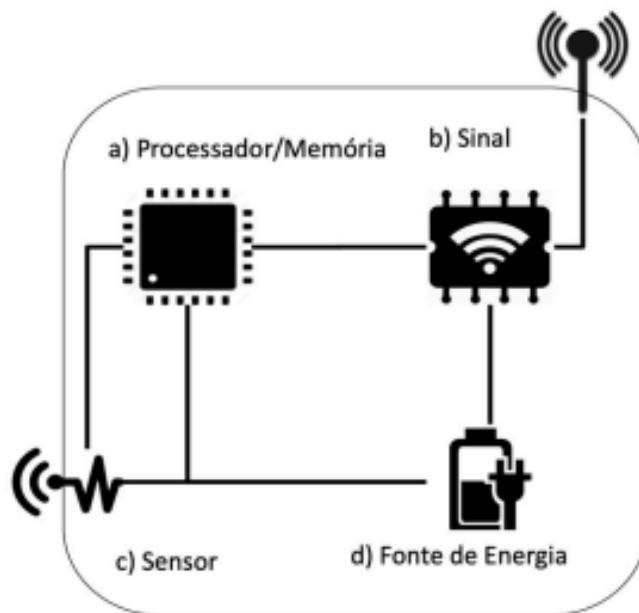
Os sensores são amplamente utilizados na IoT para monitorar pessoas e ambientes. Exemplos desses sensores incluem aqueles que medem umidade, temperatura e presença. Esses dispositivos coletam dados, geralmente gerando grandes volumes de informações (WESSER, 2019).

A interconexão de objetos IoT e a coleta de uma grande quantidade de dados trazem consigo o potencial de gerar informações valiosas e serviços inovadores. Com a análise desses dados, é possível obter percepção sobre padrões de comportamento, otimizar processos, melhorar a eficiência energética, prever falhas em equipamentos e tomar decisões mais embasadas. Além disso, a capacidade dos dispositivos IoT de operar de maneira autônoma, por meio da internet, permite a tomada de ações em tempo real, tornando possível a automação de tarefas e a criação de ambientes mais inteligentes e conectados(AL-FUQAHÀ et. al, 2015).

Segundo SANTOS et. al (2017) a arquitetura básica de um hardware na IoT deve apresentar as seguintes características:

- Unidade de processamento e memória: Consiste em uma memória interna para armazenamento de dados e programas, um microcontrolador e um conversor analógico-digital para receber informações dos sensores.

- Unidade de comunicação: Refere-se a algum meio de comunicação, seja com fio ou sem fio, utilizado para transmitir dados.
- Fonte de energia: Fornecer energia aos componentes do sistema, podendo ser uma bateria (recarregável ou não), um conversor AC-DC ou outras fontes de alimentação, como energia elétrica ou solar.
- Unidade sensora ou atuadora: Compreende os sensores, responsáveis por capturar valores de grandezas físicas, como temperatura, umidade, pressão e presença, e os atuadores, dispositivos que executam ações em resposta a comandos elétricos ou mecânicos.



Adaptado de SANTOS et al. (2016)

Figura 1 - Arquitetura básica de um hardware para IoT.

6.2 APIs REST

Uma API REST é um conjunto de definições e protocolos para criar e integrar software de aplicativo. Às vezes, é denominada como um contrato, interface ou ponto de acesso entre um provedor de informações e um usuário de informações, estabelecendo os requisitos de entrada do cliente e os requisitos de saída do servidor (RED HAT, 2020).

A REST (Representational State Transfer) é um conjunto de limitações arquitetônicas que visa reduzir a latência e a comunicação em rede, ao mesmo tempo em que maximiza a independência e a escalabilidade das implementações dos componentes web. Ao contrário de outros estilos, o REST permite o acesso à informação e a execução de comandos nos componentes por meio de uma semântica direta de acesso ao serviço (FERREIRA, 2014).

6.3 Propósito do Projeto

No Campus de Camboriú do Instituto Federal de Ciência e Tecnologia (IFC), onde estou atualmente estudando, está em andamento um projeto de implantação de estações meteorológicas nas escolas da cidade de Camboriú. Com o objetivo de coletar os dados provenientes dessas estações, que funcionam como dispositivos IoT, surge a necessidade de desenvolver uma aplicação que seja capaz de gerenciar e processar os dados gerados por esses dispositivos. Essa demanda surge devido à quantidade de dispositivos envolvidos no projeto, requerendo uma solução eficiente para a coleta e organização dos dados.

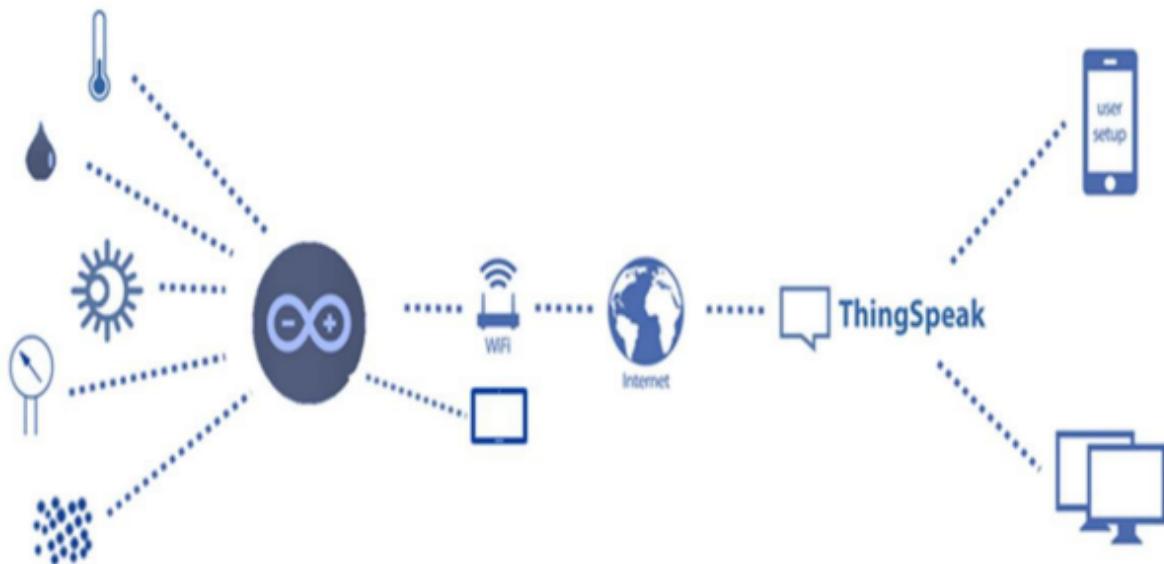
Ao abordar o problema da coleta de dados de dispositivos IoT, identifiquei os requisitos essenciais que uma plataforma de gerenciamento deve atender para solucionar esse desafio. Esses requisitos são os seguintes:

- Conectividade: A aplicação deve ser capaz de se comunicar e se conectar facilmente com os dispositivos.

- Coleta de dados em tempo real: A aplicação deve ser capaz de receber e armazenar os dados em tempo real, para que assim as informações mais recentes estejam disponíveis para análise e tomada de decisões.
- Armazenamento e processamento de dados: A aplicação deve ter a capacidade de armazenar os dados coletados de forma eficiente e escalável, permitindo o acesso e processamento posterior dos dados para análises e visualizações.
- Gerenciamento de dispositivos: A aplicação deve oferecer recursos de gerenciamento dos dispositivos IoT, como registro, autenticação e configuração, facilitando o controle e o monitoramento dos dispositivos conectados.
- Visualização e análise de dados: A aplicação deve fornecer recursos para visualização e análise dos dados coletados.
- Segurança: A aplicação deve garantir a segurança dos dados coletados e transmitidos, adotando práticas adequadas de criptografia, autenticação e controle de acesso.

Ao seguir esses requisitos e utilizar uma arquitetura adequada, será possível implementar uma plataforma eficiente para coleta e gerenciamento dos dados provenientes de dispositivos IoT, como as estações meteorológicas.

A seguir, apresento uma representação da arquitetura de funcionamento de uma estação meteorológica, conforme demonstrado na **Figura 2** abaixo:



Fonte (ROCKENBACH, 2019)

Figura 2 - Arquitetura do funcionamento de uma estação meteorológica.

Uma estação meteorológica é composta por vários componentes que trabalham juntos para coletar dados meteorológicos e exibi-los em uma tela ou dispositivo de exibição. Aqui está uma visão geral da arquitetura de funcionamento de uma estação meteorológica:

- Sensores: Uma estação meteorológica possui vários sensores para medir diferentes parâmetros meteorológicos, como temperatura, umidade, pressão atmosférica, velocidade e direção do vento, precipitação, entre outros. Esses sensores podem ser analógicos ou digitais e são responsáveis por captar os dados no ambiente.
- Unidade de controle: É responsável por gerenciar os sensores, coletar os dados capturados e processá-los. A unidade de controle também pode conter um microcontrolador ou um computador embarcado para realizar os cálculos e manipulações necessárias. Um exemplo de implementação de uma unidade controladora seria usando o Arduino.
- Transmissão de dados: Os dados coletados pelos sensores são transmitidos para a unidade de controle por meio de cabos ou comunicação sem fio, dependendo do

design da estação meteorológica.

- Armazenamento de dados: Os dados podem ser armazenados em formato bruto ou podem passar por um pré-processamento antes de serem armazenados. Esses dados são armazenados em algum tipo de memória, como um banco de dados que já tenha integração com uma plataforma de gerenciamento.
- Processamento de dados: Os dados coletados pelos sensores podem exigir algum processamento antes de serem exibidos. O processamento de dados é realizado pela plataforma de gerenciamento de dispositivos.
- Interface de usuário e exibição: Os dados processados são enviados para uma interface de usuário, que pode ser uma tela LCD, um computador, um aplicativo móvel ou uma plataforma online. Essa interface exibe as informações meteorológicas de maneira comprehensível para os usuários, com gráficos, tabelas e afins.

6.4 Plataformas

Hoje em dia temos plataformas que oferecem recursos abrangentes para conectar, controlar e visualizar dispositivos IoT. As plataformas simplificam a integração de dispositivos e facilitam o gerenciamento centralizado. Uma plataforma de IoT é uma solução tecnológica com múltiplas camadas que viabiliza o controle de dispositivos conectados à Internet das Coisas. Essa plataforma estabelece a conexão entre os dispositivos, independentemente da sua diversidade, e a infraestrutura em nuvem, por meio de alternativas flexíveis de conectividade. Ademais, essas plataformas possuem uma elevada capacidade de processamento de dados. Para os profissionais envolvidos no desenvolvimento de soluções IoT, uma plataforma oferece um conjunto de recursos pré-configurados que agilizam o processo de criação dessas soluções(WHAT..., 2019).

6.4.1 Plataforma ThingsBoard

O ThingsBoard é uma plataforma de IoT de código aberto que possibilita o desenvolvimento, gerenciamento e escalabilidade ágil de projetos de IoT. Seu principal propósito é disponibilizar uma infraestrutura pronta para uso na nuvem ou em soluções locais, oferecendo suporte de servidor para aplicativos de IoT. Com o ThingsBoard, é possível agilizar o processo de implementação de projetos de IoT e garantir uma base sólida para o crescimento e o gerenciamento eficiente das soluções (THINGSBOARD, 2023).

A plataforma do ThingsBoard é composta por duas partes principais: a API do dispositivo e a API do servidor. A API do dispositivo é organizada de acordo com os protocolos de comunicação suportados, como a API MQTT, a API MQTT Sparkplug, a AwhatPI CoAP, a API HTTP, a API LWM2M e a API SNMP (THINGSBOARD, 2023).

Esta plataforma oferece um token de acesso exclusivo que simplifica a gestão dos dispositivos, permitindo estabelecer a conexão entre eles e identificar a propriedade. Além disso, o token possibilita controlar quem terá acesso aos dados e visualização das informações (INACIO, 2022). Além do Token a plataforma oferece outras opções de autenticação como: credenciais MQTT básicas, Certificados X.509.

6.4.2 Plataforma Blink

Blynk é um conjunto abrangente de software que possibilita a prototipagem, implantação e gerenciamento remoto de dispositivos eletrônicos conectados em qualquer escala. Seja para projetos pessoais de IoT ou produtos conectados comerciais em larga escala, o Blynk capacita os usuários a conectar seus dispositivos de hardware à nuvem e criar aplicativos para iOS, Android e web, analisar dados em tempo real e históricos dos dispositivos, controlá-los remotamente de qualquer lugar, receber notificações importantes e muito mais (BLINK, 2023).

Existem diversas maneiras de conectar o dispositivo IoT nesta plataforma, que são: De acordo com a documentação da Blynk (2023) disponível em seu site, existem diversas maneiras de conectar dispositivos IoT à plataforma. Algumas delas incluem:

- Blynk Library (Biblioteca Blynk): É uma biblioteca C++ fácil de usar e portátil, pré-configurada para funcionar com várias placas de desenvolvimento. Ela implementa um protocolo de conexão contínua que permite uma comunicação bidirecional com baixa latência.
- Blynk.Edgent: É uma solução empacotada que simplifica a conexão de dispositivos à plataforma Blynk, sem a necessidade de extensa programação. Ela inclui a API da biblioteca Blynk, Blynk.Inject (provisionamento dinâmico de credenciais do dispositivo).
- Blynk.NCP: É uma pilha de software oferecida pela Blynk para diversos Co-Processadores de Rede (NCP). O NCP é um chip ou módulo dedicado que gerencia a conectividade com a nuvem Blynk, descarregando essa função do MCU principal do dispositivo.
- API HTTP(s): A Blynk também oferece uma API padrão baseada em comunicação HTTP(s) que pode ser utilizada por qualquer dispositivo conectado à Internet. Essa API permite que o dispositivo envie dados para a nuvem Blynk periodicamente ou em lotes, sendo especialmente útil para dispositivos celulares.

6.4.3 Plataforma ThingSpeak

O ThingSpeak é um serviço de plataforma de análise de IoT que possibilita reunir, observar e examinar fluxos de informações em tempo real na nuvem. É possível transmitir dados para o ThingSpeak a partir dos seus dispositivos, elaborar representações visuais imediatas dos dados em tempo real e enviar notificações utilizando serviços online. A plataforma possibilita que engenheiros e cientistas desenvolvam protótipos e construam sistemas IoT sem necessidade de configurar servidores ou desenvolver software web

(THINGSPEAK, 2023).

Além disso, o ThingSpeak oferece recursos de integração e compartilhamento de dados que ampliam suas funcionalidades. Os usuários podem facilmente integrar o ThingSpeak a outras plataformas e serviços IoT, permitindo a troca de informações e a criação de soluções mais completas. Através de APIs (Interfaces de Programação de Aplicativos), é possível conectar o ThingSpeak a sistemas de terceiros, como aplicativos móveis, bancos de dados e plataformas de análise de dados (THINGSPEAK, 2023).

6.4.4 Comparação das Plataformas

Requisitos	ThingsBoard	Blink	ThingSpeak
Conectividade	Sim	Sim	Sim
Coleta de dados em tempo real	Sim	Sim	Sim
Armazenamento e processamento de dados	Sim	Sim	Sim
Gerenciamento de dispositivos	Sim	Sim	Básico
Visualização e análise de dados	Sim	Básico	Sim
Segurança	Sim	Sim	Sim
Software Livre	Sim	Não	Não

6.4.5 Escolha da Plataforma para Implementação

A plataforma escolhida para a implementação e implantação do projeto foi o ThingBoard. Essa escolha foi baseada na capacidade da plataforma de atender a todos os requisitos necessários, além de ser uma solução Open Source, o que significa que seu código-fonte está disponível para visualização, modificação e distribuição pela comunidade de desenvolvedores. Essa característica proporciona flexibilidade e a possibilidade de personalização de acordo com as necessidades específicas do projeto.

6.5 Protótipo de Implementação

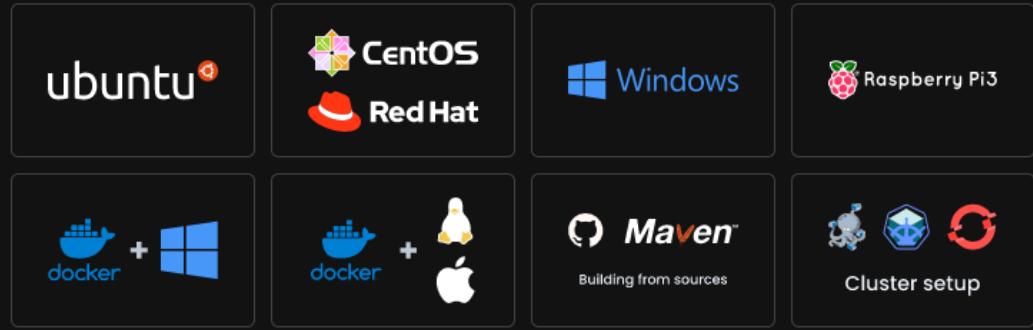
6.5.1 Instalação

A primeira etapa da implementação da plataforma consiste na instalação do ThingsBoard na máquina. Seguindo a documentação oficial do ThingsBoard (2023), existem três opções de instalação disponíveis: Demonstração ao vivo, Na premissa (máquina local) e Nuvem. Optei pela instalação 'Na premissa' e escolhi a opção Docker + Windows para o sistema operacional Windows da Microsoft em minha máquina pessoal, conforme ilustrado na **Figura 3**. É necessário ter o Docker Toolbox para Windows instalado na máquina.

Opções de instalação do ThingsBoard

O ThingsBoard foi projetado para ser executado e utilizado na maioria dos hardwares, desde Raspberry Pi local até servidores poderosos na nuvem

 Demonstração ao vivo  Na premissa  Nuvem



Fonte: (THINGSBOARD, 2023)

Figura 3 - Opções de Instalação do ThingsBoard

Existem três tipos de imagens docker de instância única do ThingsBoard, dependendo do banco de dados usado (ThingsBoard, 2023):

1. ThingsBoard/tb-postgres - Instância única do ThingsBoard com banco de dados PostgreSQL. Recomendado para servidores pequenos com pelo menos 1GB de RAM e baixa carga de mensagens por segundo. Recomenda-se ter 2-4GB de RAM. Opção utilizada nessa fase do projeto.
2. ThingsBoard/tb-cassandra - Instância única do ThingsBoard com banco de dados Cassandra. Opção de melhor desempenho, porém requer pelo menos 6GB de RAM. Recomenda-se ter 8GB de RAM.
3. ThingsBoard/tb - Instância única do ThingsBoard com banco de dados HSQLDB incorporado. Não recomendado para avaliações ou uso em

produção. Destina-se apenas para fins de desenvolvimento e testes automáticos.

A próxima etapa é escolher o serviço de fila da plataforma, responsável por organizar e gerenciar as mensagens enviadas pelos dispositivos IoT. O ThingsBoard possui suporte para integração com diversos sistemas de mensageria, como MQTT, Kafka, RabbitMQ, entre outros. Para este projeto, decidi dar preferência às seguintes implementações:

- A fila In Memory é padrão para ambientes de desenvolvimento, mas não é adequada para implantações de produção ou clusters.
- Kafka é recomendado para implantações de produção, amplamente utilizado no ThingsBoard. É útil para implantações locais e em nuvem privada, garantindo independência do provedor de nuvem. Além disso, alguns provedores, como AWS MSK, também oferecem serviços gerenciados para Kafka.

No entanto, para esta apresentação, optei por utilizar a fila In Memory, que é recomendada para testes. O ThingsBoard inclui o serviço In Memory Queue e o usa por padrão sem configurações extras.

Criei um arquivo de composição do docker **Figura 4** para o serviço de fila do ThingsBoard, conforme a documentação orienta:

```
1  version: '3.0'
2
3  services:
4    mytb:
5      restart: always
6      image: "thingsboard/tb-postgres"
7      ports:
8        - "8080:9090"
9        - "1883:1883"
10       - "7070:7070"
11       - "5683-5688:5683-5688/udp"
12     environment:
13       TB_QUEUE_TYPE: in-memory
14     volumes:
15       - mytb-data:/data
16       - mytb-logs:/var/log/thingsboard
17   volumes:
18     mytb-data:
19       external: true
20     mytb-logs:
21       external: true
```

Fonte: Autor (2023)

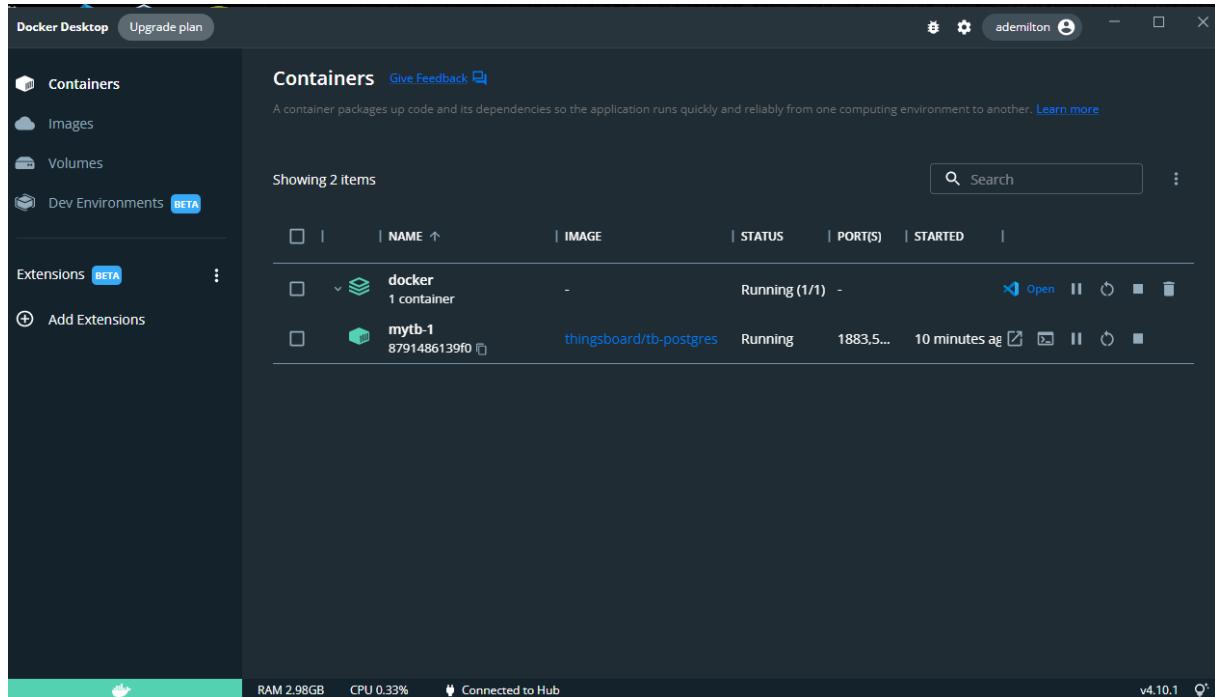
Figura 4 - Arquivo de composição do docker para o serviço de fila do ThingsBoard

Após seguir as instruções da documentação do ThingsBoard (2023), a instalação foi concluída com sucesso, como mostrado na **Figura 5**. A imagem da instância thingsboard/tb-postgres está rodando no Docker, conforme mostra a **Figura 6**.

Relatório – Projeto Integrador I (PPC-2019)

Fonte: Autor (2023)

Figura 5 - Exibição de conclusão de instalação do Thingsboard

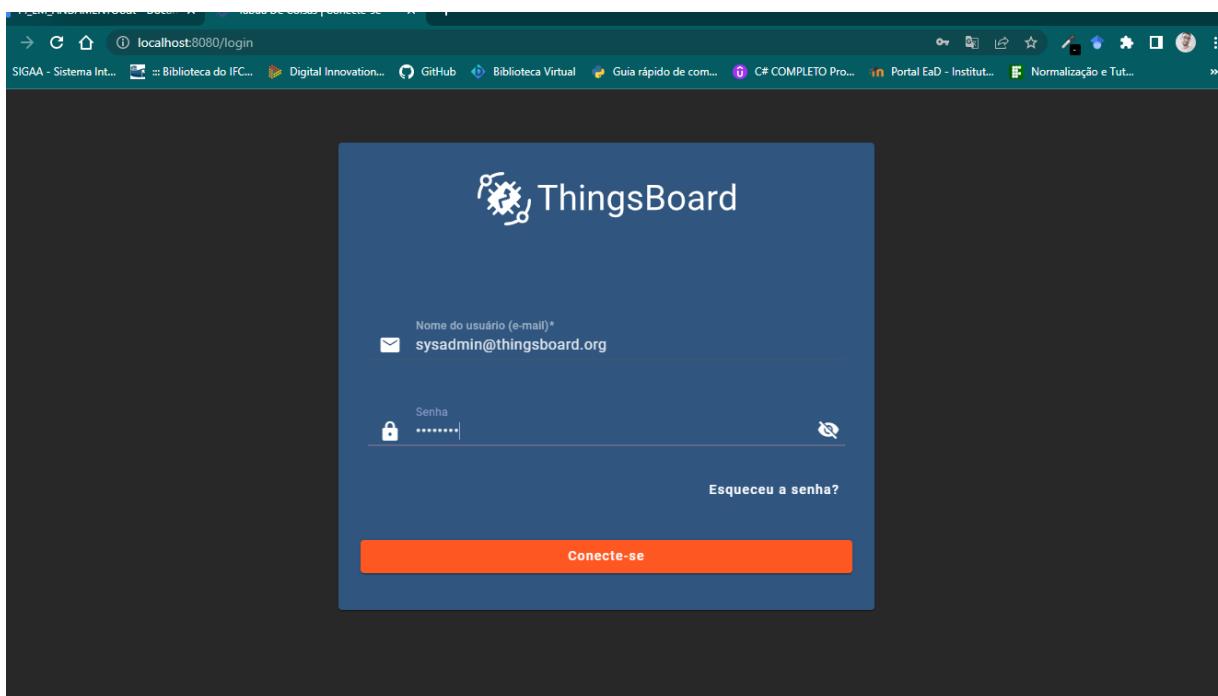


Fonte: Autor (2023)

Figura 6 - Imagem da instância thingsboard/tb-postgres rodando

6.5.2 Funcionalidades

Após instalar o ThingsBoard e acessar a IU (Interface do Usuário) da plataforma disponível usando o URL: <http://localhost:8080> temos acesso a tela de login apresentada na **Figura 7.**



Fonte: Autor (2023)

Figura 7 - Tela de Login do ThingsBoard

No contexto do ThingsBoard, destacam-se três papéis de usuários principais: administrador, inquilino e cliente. O administrador possui os mais altos privilégios, sendo responsável por configurar, gerenciar e monitorar a plataforma como um todo. Já o inquilino é uma entidade ou organização que utiliza o ThingsBoard para gerenciar e monitorar seus dispositivos IoT e dados associados, tendo um ambiente isolado para essa finalidade. Por fim, o cliente é o usuário final que interage com a plataforma para monitorar e controlar seus

dispositivos IoT.

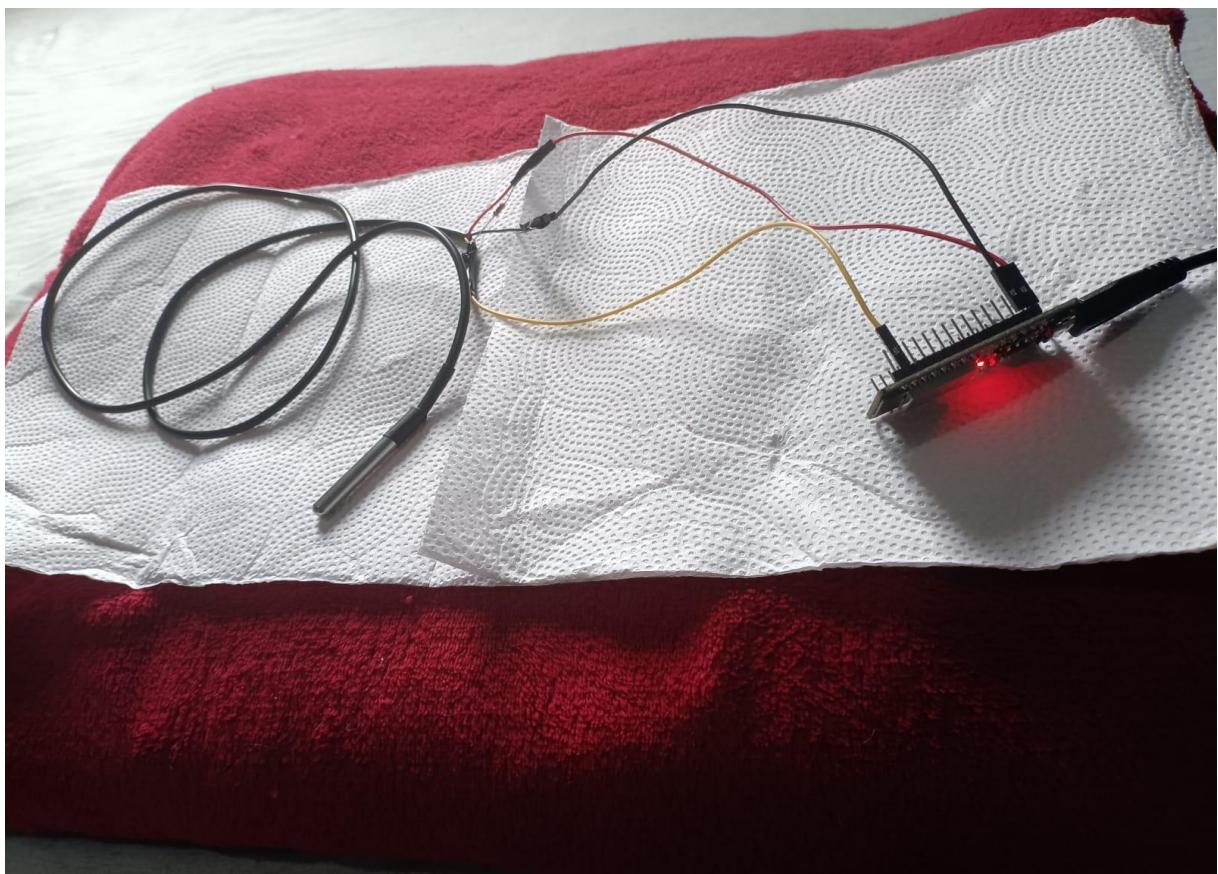
Durante o protótipo de implementação, foram utilizadas credenciais de administrador para acessar o sistema. A partir disso, um usuário de inquilino foi criado para realização dos testes. Foram realizadas simulações de dispositivos IoT, utilizando requisições HTTP para enviar dados de temperatura e umidade para a plataforma. Os testes envolveram a validação da conectividade com um suposto dispositivo, autenticação por meio de token e a visualização dos dados por meio de um painel (dashboard).

Para visualizar o passo a passo detalhado dos testes realizados, consulte o Anexo A, que contém os prints das telas e as descrições correspondentes.

7. APLICAÇÃO

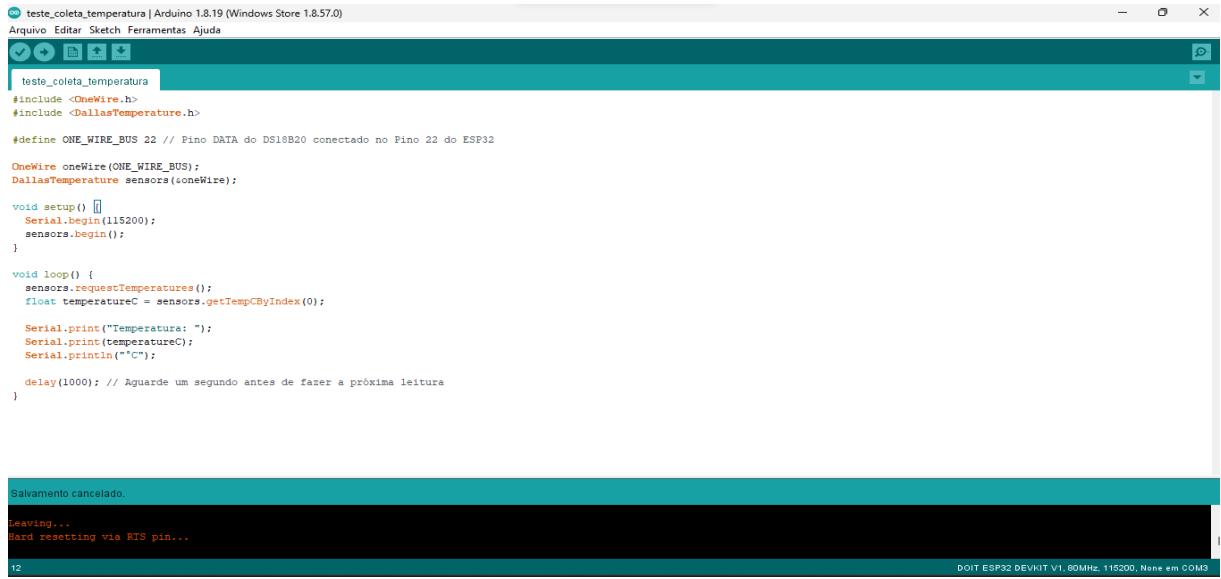
Nesta fase, conduzimos testes empregando os protocolos HTTP e MQTT. Para dar início aos experimentos, foi necessária a aquisição e configuração de uma placa ESP32 DevKit, juntamente com um sensor de temperatura DS18B20, conforme ilustrado na **Figura 8**, equipado com três pinos essenciais: VCC (alimentação), GND (terra) e DATA (dados).

A etapa inicial teve por objetivo verificar o desempenho do dispositivo no que diz respeito à coleta precisa de dados de temperatura. Para este fim, utilizamos o ambiente de desenvolvimento ARDUINO IDE para programar a placa ESP32. O Código exibido na **Figura 9** foi empregado para configurar a placa e coletar os dados de temperatura, os quais eram exibidos no console.



Fonte: Autor (2023)

Figura 8 - ESP32 DevKit com o sensor de temperatura DS18B20.



```

teste_coleta_temperatura | Arduino 1.8.19 (Windows Store 1.8.57.0)
Arquivo Editar Sketch Ferramentas Ajuda
teste_coleta_temperatura
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 22 // Pino DATA do DS18B20 conectado no Pino 22 do ESP32

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(115200);
  sensors.begin();
}

void loop() {
  sensors.requestTemperatures();
  float temperatureC = sensors.getTempCByIndex(0);

  Serial.print("Temperatura: ");
  Serial.print(temperatureC);
  Serial.println("°C");

  delay(1000); // Aguarde um segundo antes de fazer a próxima leitura
}

```

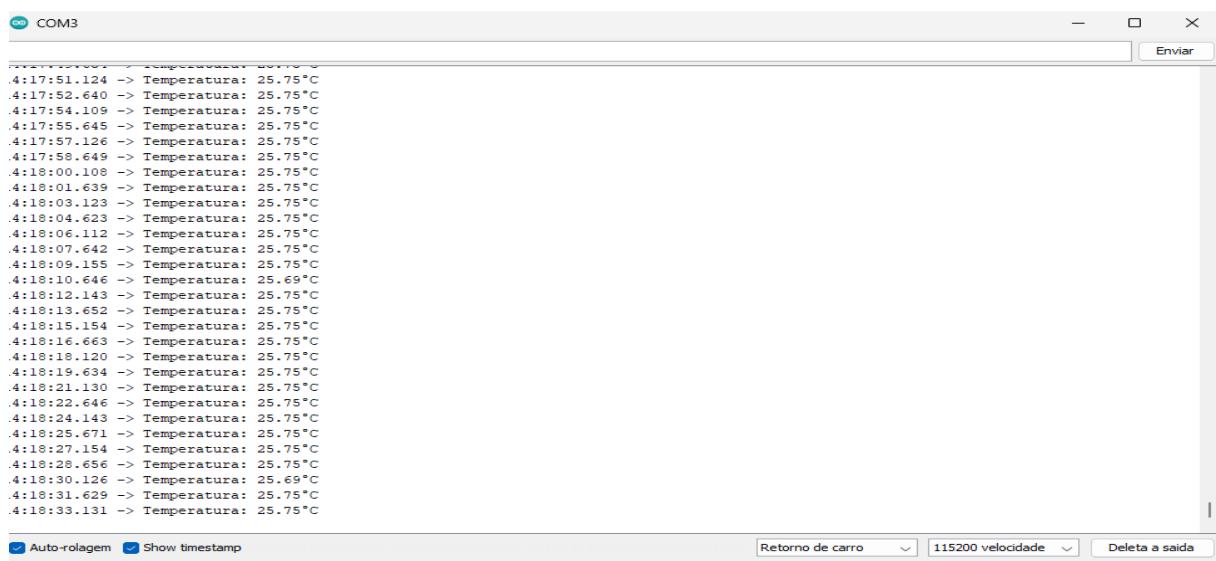
Salvamento cancelado.
Leaving...
Hard resetting via RTS pin...
12

DOIT ESP32 DEVKIT V1, 80MHz, 115200, None em COM3

Fonte: Autor (2023)

Figura 9 - Código para coletar temperatura utilizando o ARDUINO IDE

Durante a fase de carregamento do código na placa, identificamos que os dados de temperatura estavam sendo registrados de maneira incorreta, exibindo valores inválidos. Neste contexto, implementamos a solução de inserir um resistor de pull-up de 4.7k ohms entre o pino DATA e a alimentação (VCC). Com a devida instalação deste resistor, a placa passou a coletar e exibir as informações de temperatura de forma precisa, como é evidenciado no console, conforme demonstrado na **Figura 10**.



The screenshot shows the Arduino IDE's Serial Monitor window titled "COM3". It displays a continuous stream of temperature readings in degrees Celsius, with each entry timestamped. The data starts at 4:17:51 and continues until 4:18:33. The temperatures are consistently shown as 25.75°C.

```

4:17:51.124 -> Temperatura: 25.75°C
4:17:52.644 -> Temperatura: 25.75°C
4:17:54.109 -> Temperatura: 25.75°C
4:17:55.645 -> Temperatura: 25.75°C
4:17:57.126 -> Temperatura: 25.75°C
4:17:58.649 -> Temperatura: 25.75°C
4:18:00.103 -> Temperatura: 25.75°C
4:18:01.639 -> Temperatura: 25.75°C
4:18:03.123 -> Temperatura: 25.75°C
4:18:04.623 -> Temperatura: 25.75°C
4:18:06.112 -> Temperatura: 25.75°C
4:18:07.642 -> Temperatura: 25.75°C
4:18:09.155 -> Temperatura: 25.75°C
4:18:10.646 -> Temperatura: 25.69°C
4:18:12.143 -> Temperatura: 25.75°C
4:18:13.652 -> Temperatura: 25.75°C
4:18:15.154 -> Temperatura: 25.75°C
4:18:16.663 -> Temperatura: 25.75°C
4:18:18.120 -> Temperatura: 25.75°C
4:18:19.634 -> Temperatura: 25.75°C
4:18:21.130 -> Temperatura: 25.75°C
4:18:22.646 -> Temperatura: 25.75°C
4:18:24.143 -> Temperatura: 25.75°C
4:18:25.671 -> Temperatura: 25.75°C
4:18:27.154 -> Temperatura: 25.75°C
4:18:28.656 -> Temperatura: 25.75°C
4:18:30.126 -> Temperatura: 25.69°C
4:18:31.629 -> Temperatura: 25.75°C
4:18:33.131 -> Temperatura: 25.75°C

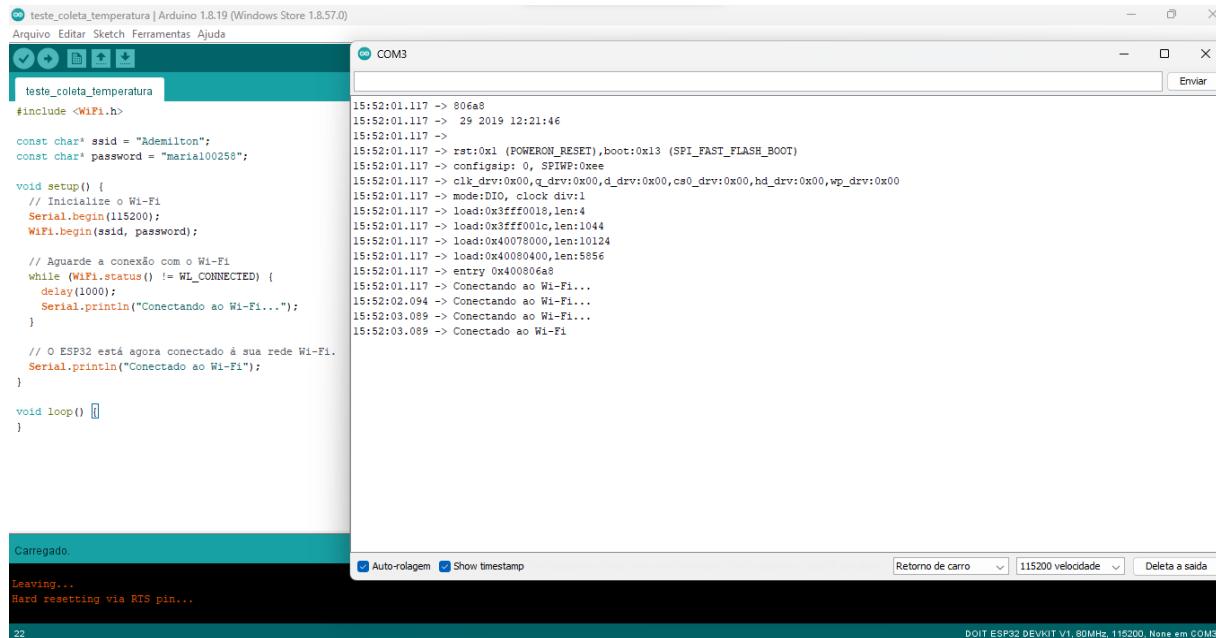
```

At the bottom of the window, there are checkboxes for "Auto-rolagem" and "Show timestamp", and buttons for "Enviar", "Retorno de carro", "115200 velocidade", and "Delete a saída".

Fonte: Autor (2023)

Figura 10- Console o ARDUINO IDE exibindo os dados de temperatura.

A segunda fase do experimento envolveu a programação da placa para estabelecer uma conexão de rede via Wi-Fi, possibilitando assim a comunicação com a plataforma ThingsBoard. O código para a realização desta conexão pode ser observado na **Figura 11**.



The screenshot shows the Arduino IDE with a sketch named "teste_coleta_temperatura" running on version 1.8.57.0. The code uses the WiFi library to connect to a Wi-Fi network. It includes a setup function to initialize the WiFi and a loop function to print connection status and a message when connected. The serial monitor shows the connection process, including timestamps and log messages like "Conectando ao Wi-Fi..." and "Conectado ao Wi-Fi".

```

teste_coleta_temperatura | Arduino 1.8.19 (Windows Store 1.8.57.0)
Arquivo Editar Sketch Ferramentas Ajuda
teste_coleta_temperatura
#include <WiFi.h>
const char* ssid = "Ademilton";
const char* password = "maria100258";
void setup() {
  // Inicialize o Wi-Fi
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  // Aguarde a conexão com o Wi-Fi
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando ao Wi-Fi...");
  }
  // O ESP32 está agora conectado à sua rede Wi-Fi.
  Serial.println("Conectado ao Wi-Fi");
}
void loop() {}

Carregado.
Leaving...
Hard resetting via RTS pin...

```

At the bottom of the window, there are checkboxes for "Auto-rolagem" and "Show timestamp", and buttons for "Enviar", "Retorno de carro", "115200 velocidade", and "Delete a saída". The status bar indicates "DUE ESP32 DEVKIT V1, 80MHz, 115200, None em COM3".

Fonte: Autor (2023)

Figura 11 - Código de conexão à rede via Wi-Fi.

Com a conclusão e o pleno funcionamento dessas etapas, estamos prontos para prosseguir com os testes envolvendo os protocolos apropriados.

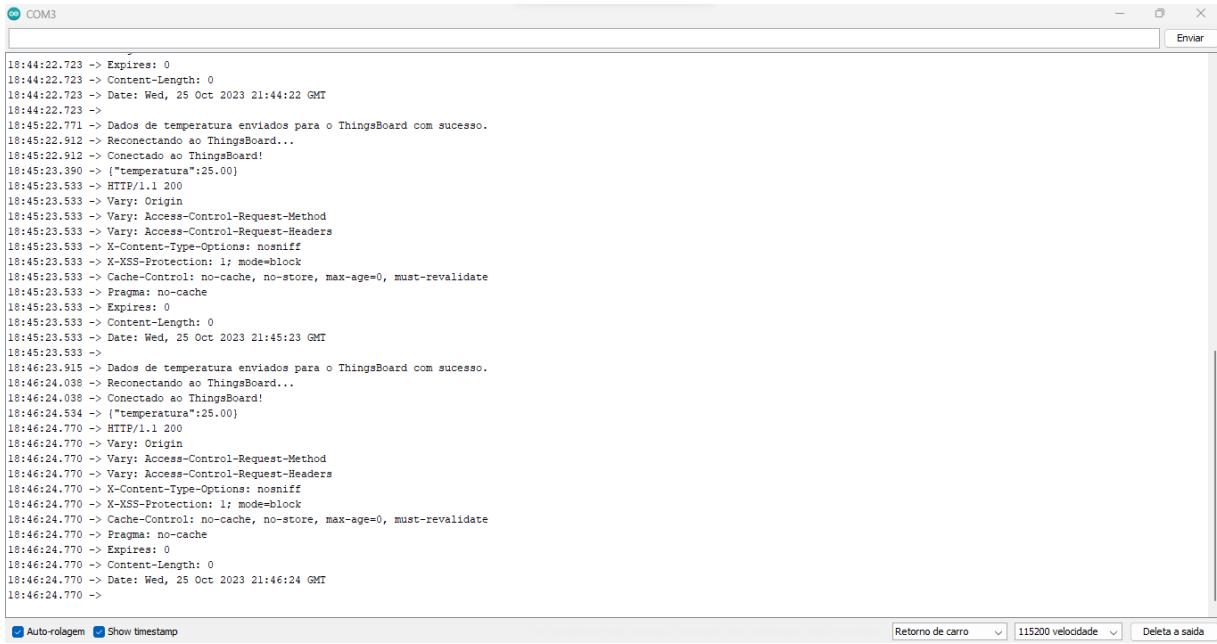
7.1 Teste com o Protocolo HTTP

O Protocolo de Transferência de Hipertexto (HTTP) tem sido o protocolo de comunicação dominante na World Wide Web (WWW) desde 1990. Ele desempenha um papel fundamental na definição das mensagens que os navegadores podem transmitir aos servidores e na determinação das respostas que receberão (JÚNIOR, 2021). Em essência, o HTTP é um protocolo orientado por solicitações e respostas.

Foram realizados testes empregando o protocolo HTTP para estabelecer a comunicação entre o dispositivo IoT (ESP32) e a plataforma ThingsBoard. No ANEXO B, você encontrará o código, incluindo comentários que guiarão a configuração do ESP32 para a execução eficaz dessa comunicação.

Após a configuração do dispositivo IoT e a execução do servidor da plataforma, é possível estabelecer a comunicação por meio de requisições HTTP. Como demonstrado na **Figura 12**, a solicitação é realizada com sucesso, exibindo detalhes do cabeçalho enviado. Na **Figura 13**, observamos os dados recebidos no painel de dashboards da plataforma. Cada dispositivo possui um token exclusivo, garantindo uma conexão segura e privada.

Relatório – Projeto Integrador I (PPC-2019)



```

COM3
18:44:22.723 -> Expires: 0
18:44:22.723 -> Content-Length: 0
18:44:22.723 ->
18:45:22.771 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
18:45:22.912 -> Reconectando ao ThingsBoard...
18:45:22.912 -> Conectado ao ThingsBoard!
18:45:23.390 -> {"temperatura":25.00}
18:45:23.533 -> HTTP/1.1 200
18:45:23.533 -> Vary: Origin
18:45:23.533 -> Vary: Access-Control-Request-Method
18:45:23.533 -> Vary: Access-Control-Request-Headers
18:45:23.533 -> X-Content-Type-Options: nosniff
18:45:23.533 -> X-XSS-Protection: 1; mode=block
18:45:23.533 -> Cache-Control: no-cache, no-store, max-age=0, must-revalidate
18:45:23.533 -> Pragma: no-cache
18:45:23.533 -> Expires: 0
18:45:23.533 -> Content-Length: 0
18:45:23.533 -> Date: Wed, 25 Oct 2023 21:45:23 GMT
18:45:23.533 ->
18:46:23.915 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
18:46:24.038 -> Reconectando ao ThingsBoard...
18:46:24.038 -> Conectado ao ThingsBoard!
18:46:24.534 -> {"temperatura":25.00}
18:46:24.770 -> HTTP/1.1 200
18:46:24.770 -> Vary: Origin
18:46:24.770 -> Vary: Access-Control-Request-Method
18:46:24.770 -> Vary: Access-Control-Request-Headers
18:46:24.770 -> X-Content-Type-Options: nosniff
18:46:24.770 -> X-XSS-Protection: 1; mode=block
18:46:24.770 -> Cache-Control: no-cache, no-store, max-age=0, must-revalidate
18:46:24.770 -> Pragma: no-cache
18:46:24.770 -> Expires: 0
18:46:24.770 -> Content-Length: 0
18:46:24.770 -> Date: Wed, 25 Oct 2023 21:46:24 GMT
18:46:24.770 ->

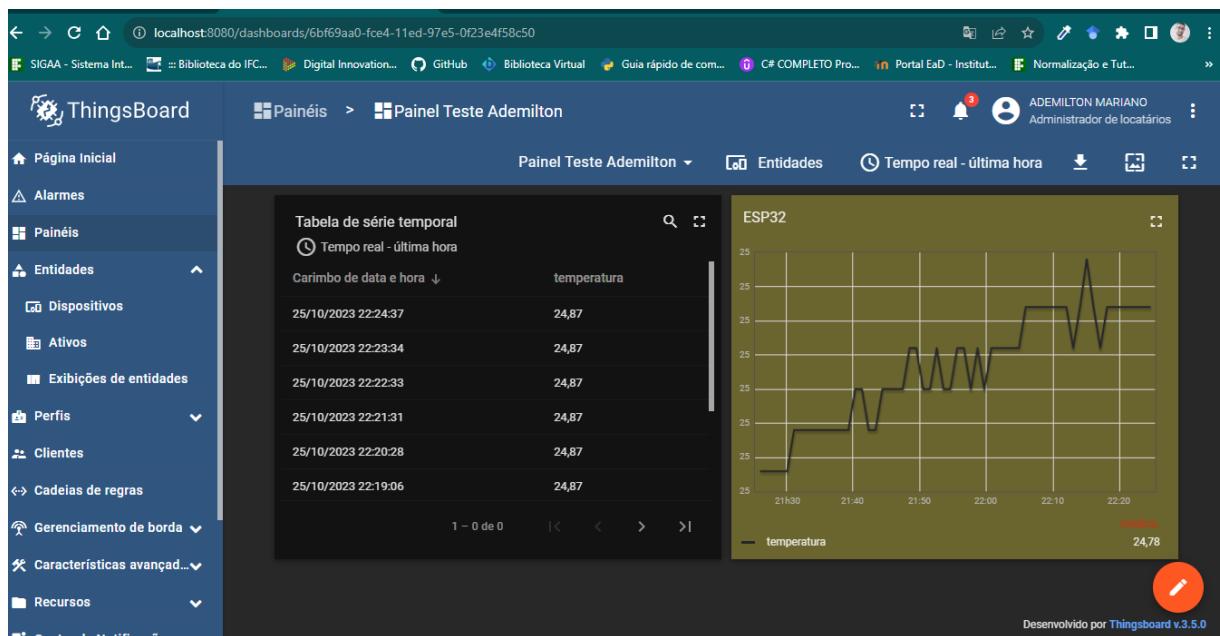
```

Auto-rolagem Show timestamp
 Retorno de carro ▾ 115200 velocidade ▾ Deleta a saída

Fonte: Autor (2023)

Figura 12 - Console do Arduino IDE..

No entanto, é importante notar que os intervalos entre as requisições HTTP costumam ser, em média, de um minuto, o que pode ser considerado um tempo prolongado para certas aplicações. Isso ocorre devido ao fato de o protocolo HTTP estabelecer uma nova conexão para cada solicitação. Além disso, os cabeçalhos do HTTP podem ser extensos, especialmente em solicitações curtas e frequentes, resultando em um maior consumo de largura de banda e aumentando o tempo de transmissão."



Fonte: Autor (2023)

Figura 13 - Gráficos exibindo os dados recebidos na plataforma ThingsBoard.

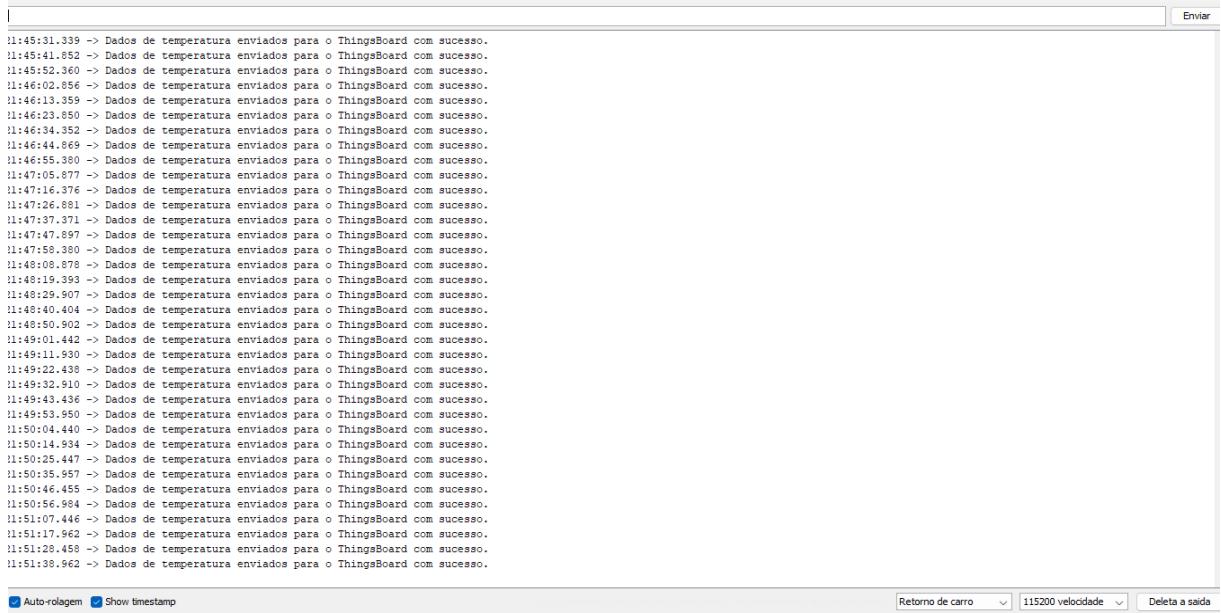
7.2 Teste com o Protocolo MQTT

O MQTT, desenvolvido em 1999 por Andy Stanford-Clark e Arlen Nipper, é um protocolo de mensagens que adota a estrutura publish/subscribe. Este protocolo é direcionado a dispositivos com recursos limitados, operando em redes não confiáveis, caracterizadas por uma escassa capacidade de transmissão e atrasos substanciais. Os princípios de design subjacentes ao MQTT visam reduzir os requisitos de hardware do dispositivo e a largura de banda necessária, assegurando, ao mesmo tempo, a confiabilidade e a garantia de comunicação (JÚNIOR, 2021).

Para conduzir os testes com o protocolo MQTT, foi empregado o código fornecido no ANEXO C, o qual configura a comunicação entre o dispositivo IoT e o ThingsBoard por meio do MQTT. Nas **Figuras 14** e **15**, é possível observar o console da IDE, indicando que os dados estão sendo transmitidos com êxito. Além disso, percebe-se que a taxa de envio e

publicação dos dados no ThingsBoard é significativamente mais rápida em comparação com o protocolo HTTP, com intervalos médios de cerca de 10 segundos por mensagem.

No contexto do protocolo MQTT, utilizamos um método de autenticação distinto, conforme ilustrado na **Figura 16**. Em vez de empregar o token, recorre-se ao MQTT Básico, no qual se define um ClientId do MQTT, um nome de usuário e uma senha. Isso implica que cada dispositivo possui credenciais exclusivas, o que desempenha um papel crucial na garantia da integridade e segurança da comunicação.



```

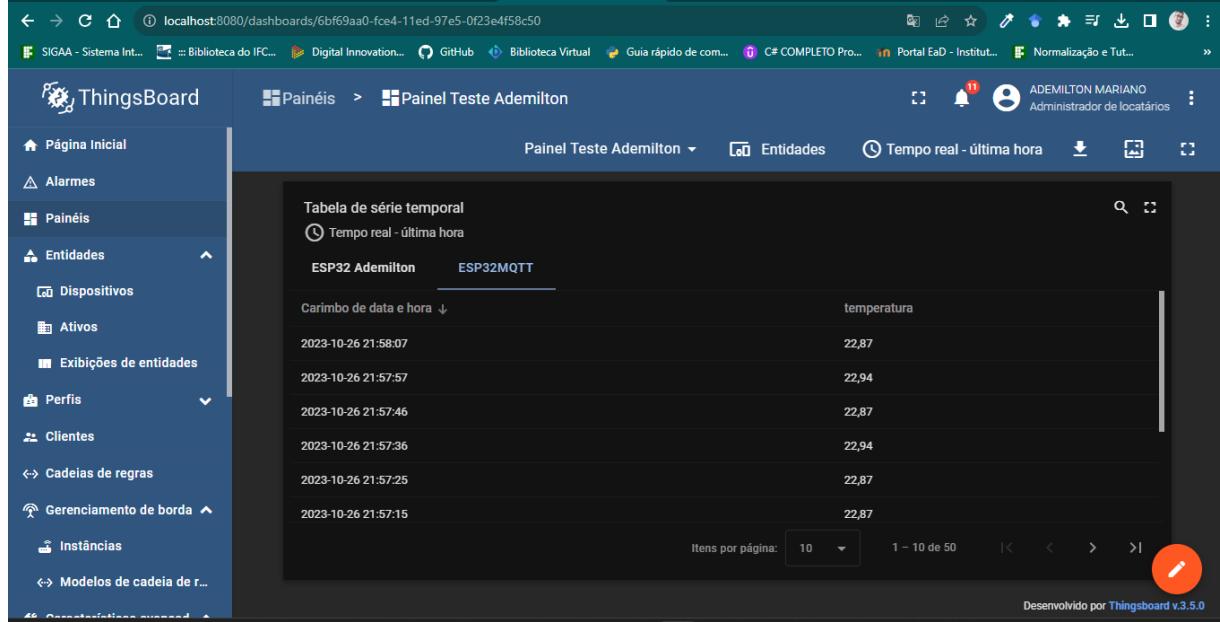
11:45:31.339 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:45:41.852 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:45:52.360 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:46:02.886 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:46:13.359 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:46:23.850 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:46:34.352 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:46:44.869 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:46:55.389 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:47:05.877 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:47:16.376 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:47:26.881 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:47:37.371 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:47:47.897 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:47:58.389 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:48:08.871 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:48:19.391 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:48:29.907 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:48:40.404 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:48:50.902 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:49:01.442 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:49:11.930 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:49:22.498 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:49:32.910 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:49:43.430 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:49:53.950 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:50:04.440 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:50:14.930 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:50:25.447 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:50:35.957 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:50:46.455 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:50:56.984 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:51:07.446 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:51:17.962 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:51:28.458 -> Dados de temperatura enviados para o ThingsBoard com sucesso.
11:51:38.962 -> Dados de temperatura enviados para o ThingsBoard com sucesso.

```

Auto-rolagem Show timestamp Retorno de carro Deleta a saída

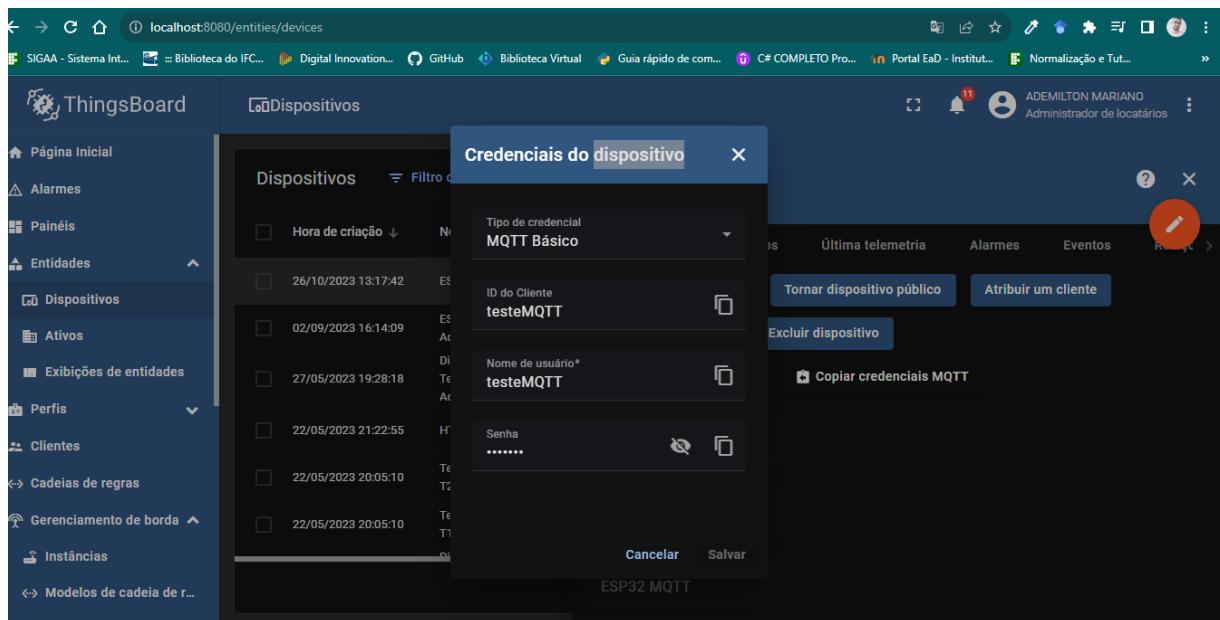
Fonte: Autor (2023)

Figura 14 - Console do Arduino IDE.



Fonte: Autor (2023)

Figura 15 - Gráfico exibindo os dados recebidos na plataforma ThingsBoard.



Fonte: Autor (2023)

Figura 16 - Credencial do tipo MQTT Básico.

Além desses protocolos que foram realizados os testes o ThingsBoard Oferece outros protocolos como:

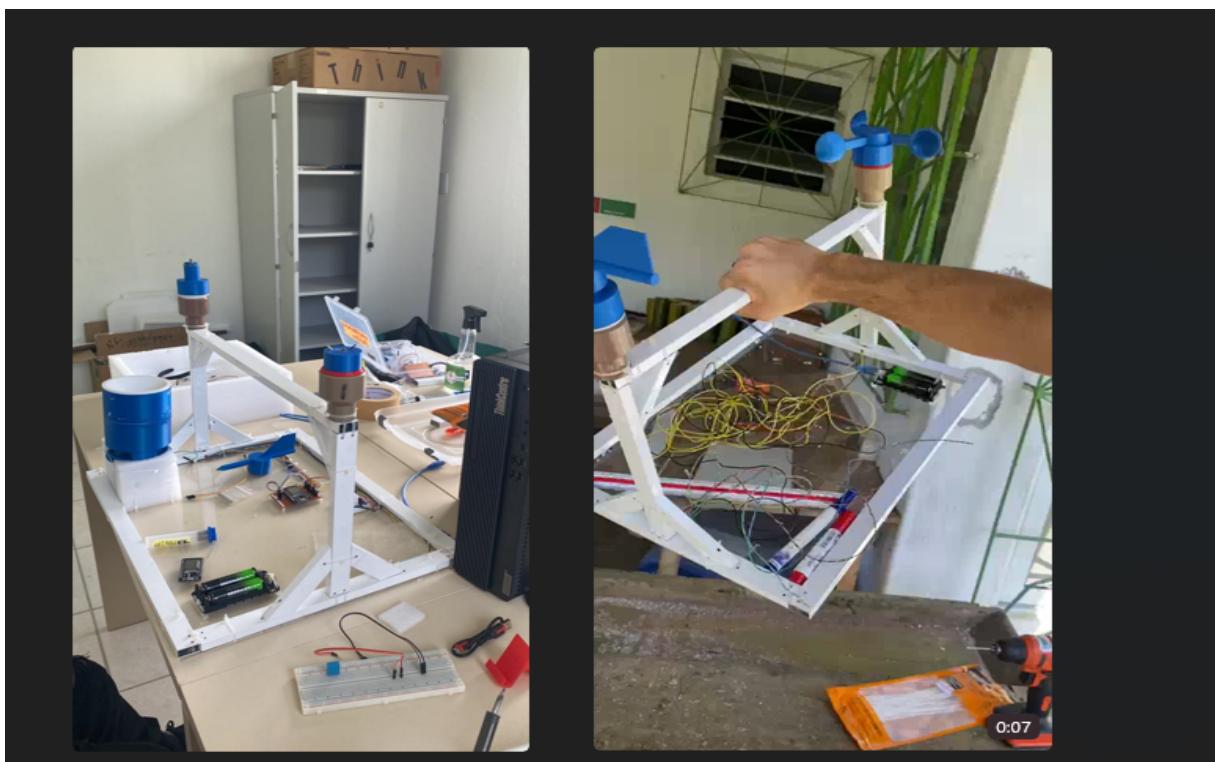
- CoAP (Protocolo de Aplicação Restrita);
- LwM2M (Máquina-a-Máquina Leve);
- SNMP (Protocolo Simples de Gerenciamento de Rede).

Segundo a documentação da plataforma ThingsBoard a maioria dos protocolos acima suporta JSON, Protobuf ou formato de dados próprio. Esses protocolos incluindo HTTP e o MQTT são as melhores opções para novos dispositivos quando você tem controle sobre o firmware. Mas a plataforma suporta também LoRaWAN (Rede de Área Ampla de Longo Alcance), Modbus, SigFox e NB IoT, entre outros. A plataforma suporta muitas integrações que abrangem a maioria dos dispositivos no mercado (ThingsBoard, 2023).

7.3 Testes Externos de Validação

No Instituto Federal Catarinense (IFC) de Camboriú, encontra-se em curso um projeto de desenvolvimento de uma estação meteorológica modular de código aberto e acessível, visando a coleta de dados e a medição de indicadores do microclima. Este projeto está sendo desenvolvido em colaboração com o Laboratório LabMaker do Campus de Camboriú.

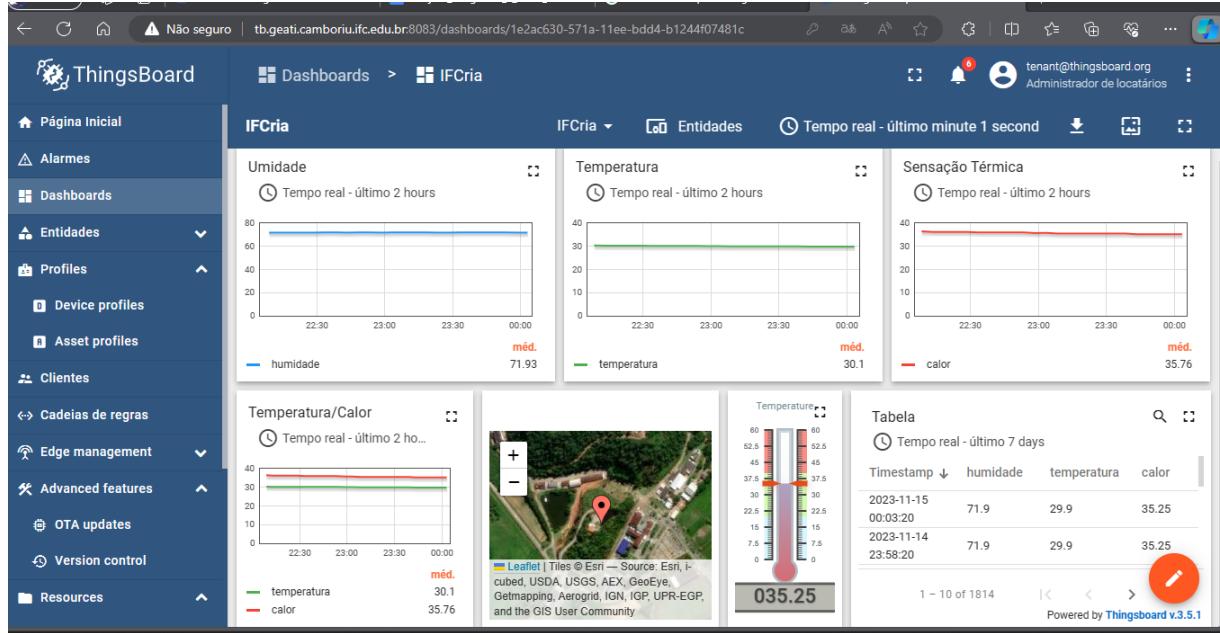
A concepção das estações meteorológicas de custo reduzido baseia-se na adoção de plataformas de hardware e software de código aberto, bem como na utilização de componentes produzidos por meio de impressoras 3D(GRANDI et al., 2023). Na **Figura 17** é possível ver um protótipo em construção da estação meteorológica. A ideia de **Proposta de uma Plataforma para Gerenciamento de Dispositivos na Internet das Coisas** surgiu da necessidade do projeto de desenvolvimento de uma estação meteorológica de baixo custo para mapeamento de microclima.



Fonte: Speroni (2023)

Figura 17 - Protótipo da estação meteorológica em construção.

Após a análise dos requisitos e a seleção da plataforma mais adequada ao projeto, uma instância da plataforma ThingsBoard foi implementada no LabMaker do IFC. Embora a instalação estivesse inicialmente voltada para a estação meteorológica de baixo custo, que ainda não está completamente operacional, a plataforma está sendo utilizada para capturar os dados de temperatura e umidade do Laboratório do IFCria, localizado no mesmo Campus de Camboriú. Os dados apresentados na **Figura 18** são extremamente relevantes ao evidenciar a necessidade de uma cobertura térmica para o Laboratório do IFCria. Estes dados estão sendo coletados por meio de uma placa ESP32 em conjunto com um sensor de umidade e temperatura DHT11, conforme ilustrado na **Figura 19**.



Fonte: Autor (2023)

Figura 18 - Gráficos exibindo os dados coletados no Laboratório do IFCria.



Fonte: Autor (2023)

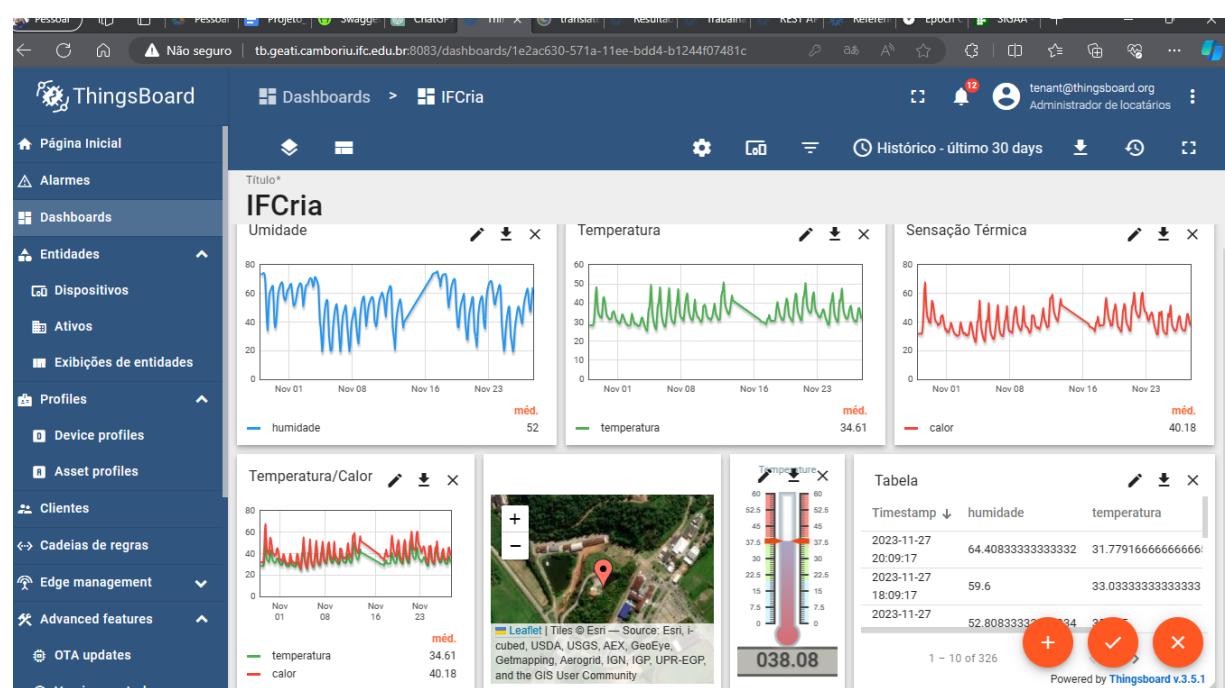
Figura 19 - ESP32 com osensor de umidade e temperatura DHT11.

Embora todos os testes e implementações estejam focados na coleta de dados meteorológicos, é importante destacar que a plataforma tem a capacidade de receber uma variedade de tipos de dados que podem ser enviados e monitorados através dos gráficos desejados. Isso permite uma utilização versátil e otimizada dos dados, buscando explorar ao máximo suas potencialidades.

7.4 Consumindo Dados da plataforma

O ThingsBoard oferece uma API REST que possibilita o tratamento flexível dos dados conforme sua necessidade. Essa API permite acessar e manipular os dados coletados pela plataforma de maneira personalizada. A documentação detalhada está disponível no Swagger UI (thingsboard.io), onde você encontrará uma variedade de interações possíveis com a plataforma. É viável conectar dispositivos, enviar informações e também recuperá-las.

Nos Dashboards (ou Painéis) criados para visualização gráfica dos dados, é possível analisar informações dentro de um período máximo de 30 dias, conforme ilustrado na **Figura 20**.



Fonte: Autor (2023)

Figura 20 -Gráficos exibindo os dados coletados no Laboratório do IFCria no periodo de 30 dias.

Entretanto, é viável consumir os dados da plataforma acessando a API mencionada no **ANEXO D**, onde é apresentada uma demonstração clara desse processo.

8. RESULTADOS ALCANÇADOS

Durante o desenvolvimento deste projeto, foram obtidos resultados significativos e relevantes. A seguir, são apresentados os principais resultados alcançados:

Conhecimento aprofundado sobre o gerenciamento e customização de APIs para dispositivos IoT: Através de extensa pesquisa e análise, foi adquirido um sólido conhecimento sobre o gerenciamento eficiente de APIs na Internet das Coisas.

Identificação dos requisitos necessários para a plataforma de coleta, processamento e visualização de dados na IoT: Por meio do levantamento de requisitos, foram identificadas as funcionalidades e características essenciais que a plataforma deve possuir para atender às demandas específicas da IoT.

Definição da arquitetura da plataforma: Foi estabelecida uma visão clara da estrutura e dos componentes necessários para a plataforma. Consideraram-se aspectos como a integração de dispositivos, o armazenamento e processamento de dados, e a interface de visualização.

Documentação e relatório final: Durante todo o projeto, foram documentadas todas as etapas, desde o levantamento de requisitos até a implementação da plataforma. Além disso, foi elaborado um relatório final abrangente, que descreve a proposta da plataforma, as metodologias utilizadas e os resultados obtidos.

Os resultados alcançados neste projeto abrangem um conhecimento aprofundado sobre o gerenciamento de APIs na IoT, a identificação dos requisitos essenciais da plataforma, a avaliação dos protocolos de comunicação, a definição arquitetural da plataforma e a documentação completa de todas as etapas do projeto. Esses resultados fornecem uma base sólida para futuros desenvolvimentos nesta área, contribuindo para a implementação de soluções eficientes e inovadoras na Internet das Coisas.

9. CONCLUSÕES E RECOMENDAÇÕES

Ao concluir este projeto de desenvolvimento de uma plataforma para coleta, processamento e visualização de dados na Internet das Coisas (IoT), gostaria de compartilhar algumas reflexões importantes que obtive ao longo dessa jornada:

Conhecimento em gerenciamento de APIs na IoT: Durante todo o processo, ficou evidente para mim a relevância de possuir um conhecimento aprofundado sobre o gerenciamento e a customização de APIs na IoT. Essa compreensão foi fundamental para realizar testes satisfatórios na plataforma proposta.

Importância de entender os requisitos específicos da IoT: Uma das lições mais valiosas que aprendi foi a necessidade de identificar os requisitos necessários para a plataforma. Compreender as demandas da IoT revelou-se essencial para projetar uma boa solução.

Relevância da arquitetura da plataforma: A definição da arquitetura da plataforma foi um passo fundamental em meu trabalho. Ao estabelecer uma visão clara da estrutura e dos componentes da plataforma, considerando aspectos como a integração de dispositivos, o armazenamento e processamento de dados, e a interface de visualização.

Documentação e o relatório final: Esses recursos desempenharam um papel crucial, pois facilitaram a compreensão do projeto, permitir melhorias futuras. Através desses registros, espero compartilhar meu aprendizado com outros entusiastas da IoT, estimulando o avanço coletivo nessa área.

Além das conclusões, gostaria de fazer algumas recomendações para futuros projetos na área da IoT:

Priorizar a identificação clara dos requisitos específicos da IoT, buscando compreender profundamente as particularidades desse ambiente para criar soluções adaptadas e eficazes. Dar importância à definição da arquitetura da plataforma, pois isso ajudará a estabelecer bases sólidas para o desenvolvimento futuro. Valorizar a documentação de todas as etapas do projeto. Esses recursos serão inestimáveis para a continuidade do projeto, facilitando melhorias e fornecendo suporte para a comunidade interessada na plataforma.

10. REFERÊNCIAS

ARUN SOLANKI, ADARSH KUMAR, A. N. Digital Cities Roadmap: IoT - Based Architecture and Sustainable Buildings. 1. ed. [s.l.] John Wiley & Sons, 2021, 2021.

DE SOUZA MARTINS, Gabriel et al. Internet das Coisas (IoT): Monitoramento remota de sinais biomédicos. Caleidoscópio, v. 12, n. 1, 2020.

FERRASI, Emerson Carlos Sarti. Internet das coisas aplicada ao gerenciamento de rastreabilidade de amostras biológicas. 2023.

AL-FUQAHÀ, Ala et al. Internet das coisas: uma pesquisa sobre tecnologias, protocolos e aplicativos facilitadores. IEEE Communications Surveys & Tutoriais, v. 17, n. 4, 2015.

ESSER, Wagner. Plataforma para coleta e persistência de dados de condições físicas e ambientais no IFC Campus Araquari. 2019. 54 f. Trabalho de curso (Bacharel em Sistemas de Informação) - Campus Araquari, Instituto Federal Catarinense, Araquari 2019. Disponível em: https://pergamumweb.ifc.edu.br/pergamumweb_ifc/vinculos/00001e/00001e25.pdf. Acesso em: 20 Maio. 2023.

THINGSBOARD. ThingsBoard Authors. ThingsBoard Community Edition, 2023. Disponível em: <https://thingsboard.io/docs/>. Acesso em: 21 Maio. 2023.

SANTOS, Bruno P. et al. Internet das coisas: da teoria à prática. Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, v. 31, 2016.

RED HAT. What is a REST API? Disponível em: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>. Acesso em: 20 Maio. 2023.

INACIO, Vinicius Vilvert et al. Aplicação de IoT industrial no LabFaber: seleção de plataforma e estudo de caso. 2022.

BLINK. Blynk Documentation. Disponível em:
https://docs.blynk.io/en/?_ga=2.157002495.556598436.1651813551-753677164.1651678057&_gl=1*1cxt6yx*_ga*NjExNDA5MjE3LjE2ODQ5NzkwNzk*_ga_E376ZQ635Y*MTY4NDk3OTA3OS4xLjEuMTY4NDk3OTQ5Ni4wLjAuMA. Acesso em: 24 Maio. 2023.

THINGSPEAK. 1994-2023 The MathWorks. Disponível em:
<https://www.mathworks.com/help/thingspeak/> Acesso em: 24 Maio. 2023.

ROCKENBACH, Fernando Roberto et al. Projeto e fabricação de uma estrutura modular para o uso em estações meteorológicas de baixo custo. 2019. Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná.

WHAT is an IoT platform? KaalIoT, [s. l.], 26 jan. [2019]. Disponível em:
<https://www.kaaiot.com/blog/what-is-iot-platform>. Acesso em: 24 Maio. 2023.

SILVA JUNIOR, Mauricio Pavan da. Análise entre protocolos HTTP e MQTT em projetos IOT. 2021. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2021.

GRANDI, Arthur; KUSS, Paulo Vinicius; SPERONI, Rafael; ANDERLE, Daniel. Desenvolvimento de uma estação meteorológica de baixo custo para mapeamento de microclima . XIV Feira de Iniciação Científica e Extensão - Instituto Federal Catarinense - Campus Camboriú, Camboriú, 2023.

APÊNDICE A – Cronograma

Item / Descrição	Ago	Set	Out	Nov	Dez
Etapa 1 - Testes de Integração	X	X	X		
Etapa 2 - Testes com protocolo HTTP		X			
Etapa 3 - Testes com Protocolo MQTT			X		
Etapa 4 - Integração de dados e testes de validação				X	
Etapa 5 - Elaboração de relatórios e gráficos				X	
Etapa 6 - Elaboração do relatório do projeto integrador II				X	

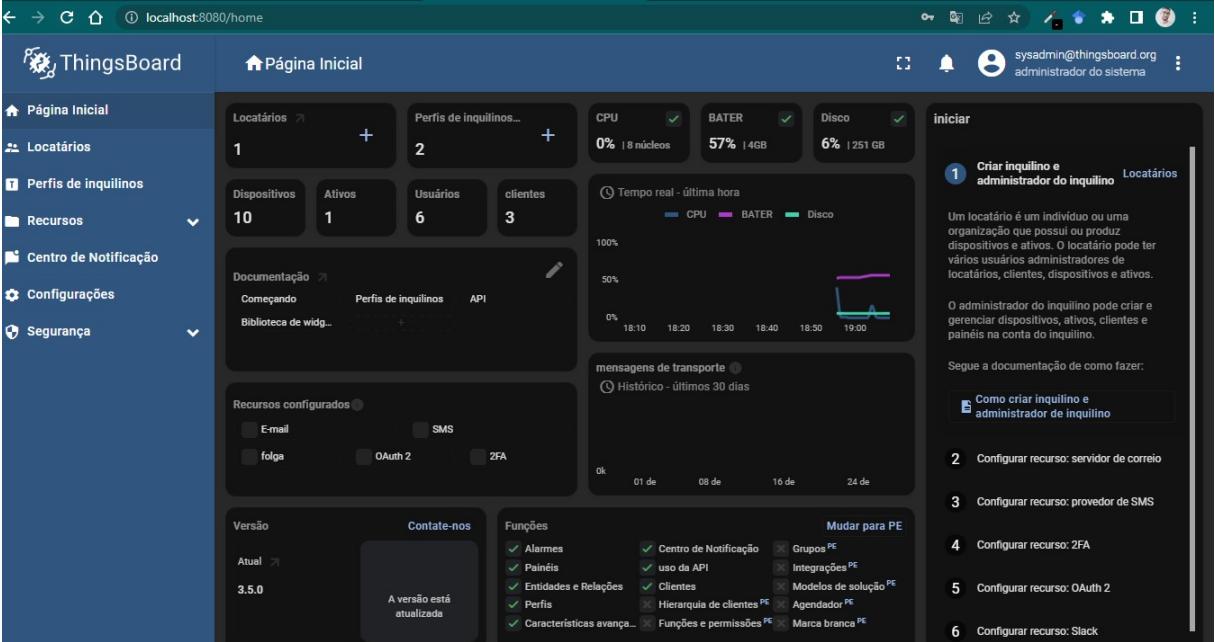
ANEXO A - Passo a Passo dos Testes Realizados.

Após realizar o login como usuário administrador na plataforma ThingsBoard, foi criado um perfil de inquilino para fins de teste. Em seguida, o login foi feito utilizando o perfil recém-criado. Um dispositivo de teste foi criado na plataforma, e um token de acesso foi gerado para permitir a realização de requisições HTTP utilizando o token. Utilizando o prompt do PowerShell, foi feito um POST conforme indicado na documentação do ThingsBoard. Após o envio do POST, foi possível verificar que o dispositivo estava ativo, conectado e exibindo os últimos dados enviados.

Posteriormente, um painel foi criado para visualizar gráficos dos dados gerados. Para isso, mais alguns posts foram feitos enviando valores diferentes, a fim de evidenciar as alterações no gráfico. Durante esse processo, optou-se por utilizar o Insomnia, uma aplicação cliente de API que é altamente eficiente para fazer requisições HTTP. Por meio de outras requisições POST realizadas, foi possível visualizar as atualizações no gráfico. Vale ressaltar que a plataforma oferece diversas funcionalidades adicionais, que serão exploradas em fases futuras do projeto.

Os endpoints da API REST do ThingsBoard podem ser consultados no Swagger disponível em <https://demo.thingsboard.io/swagger-ui/>.

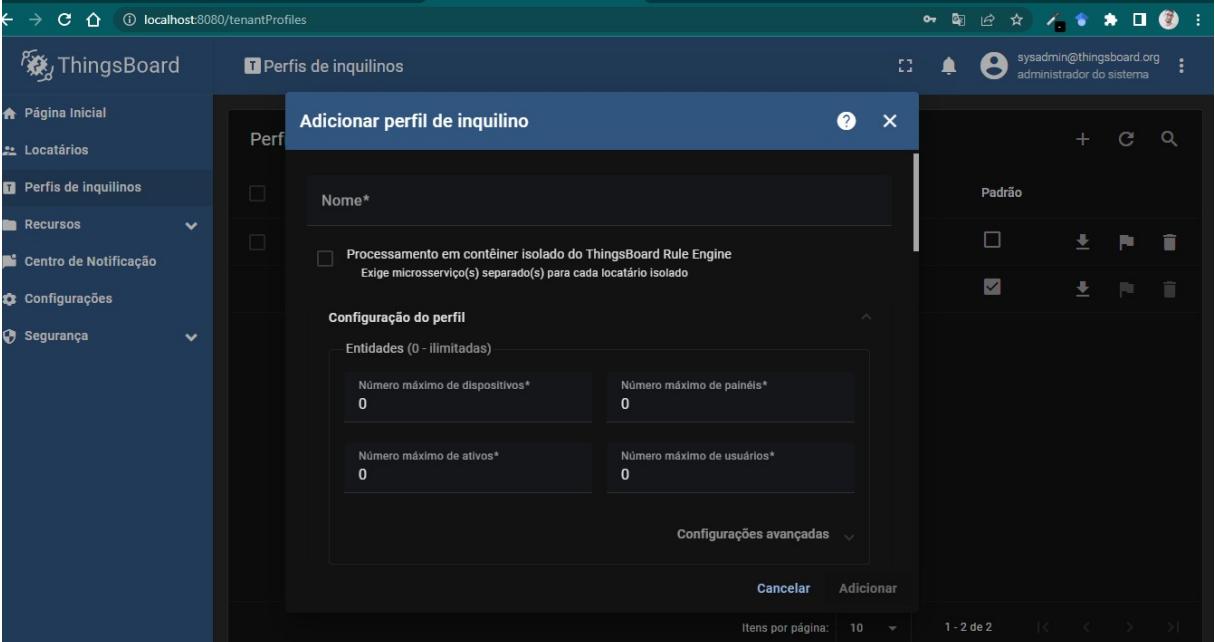
A seguir, serão apresentados os prints capturados durante o processo, acompanhados de breves descrições de cada um.



The screenshot shows the ThingsBoard administrator dashboard. On the left, there's a sidebar with navigation links: Página Inicial, Locatários, Perfis de inquilinos, Recursos, Centro de Notificação, Configurações, and Segurança. The main area is titled "Página Inicial". It displays several cards: "Locatários" (1 locatário), "Perfis de inquilinos" (2 perfis), "CPU" (0% usage), "BATER" (57% battery), and "Disco" (6% disk). Below these are cards for "Dispositivos" (10 devices), "Ativos" (1 active), "Usuários" (6 users), and "clientes" (3 clients). A "Tempo real - última hora" chart shows CPU, BATER, and Disco usage over time. There's also a "mensagens de transporte" section with a "Histórico - últimos 30 dias" chart. At the bottom, there's a "Versão" card (3.5.0) stating "A versão está atualizada" and a "Contate-nos" card. On the right, a sidebar titled "iniciar" provides a step-by-step guide for creating a tenant and administrator, along with documentation links.

Fonte: Autor (Plataforma ThingsBoard)

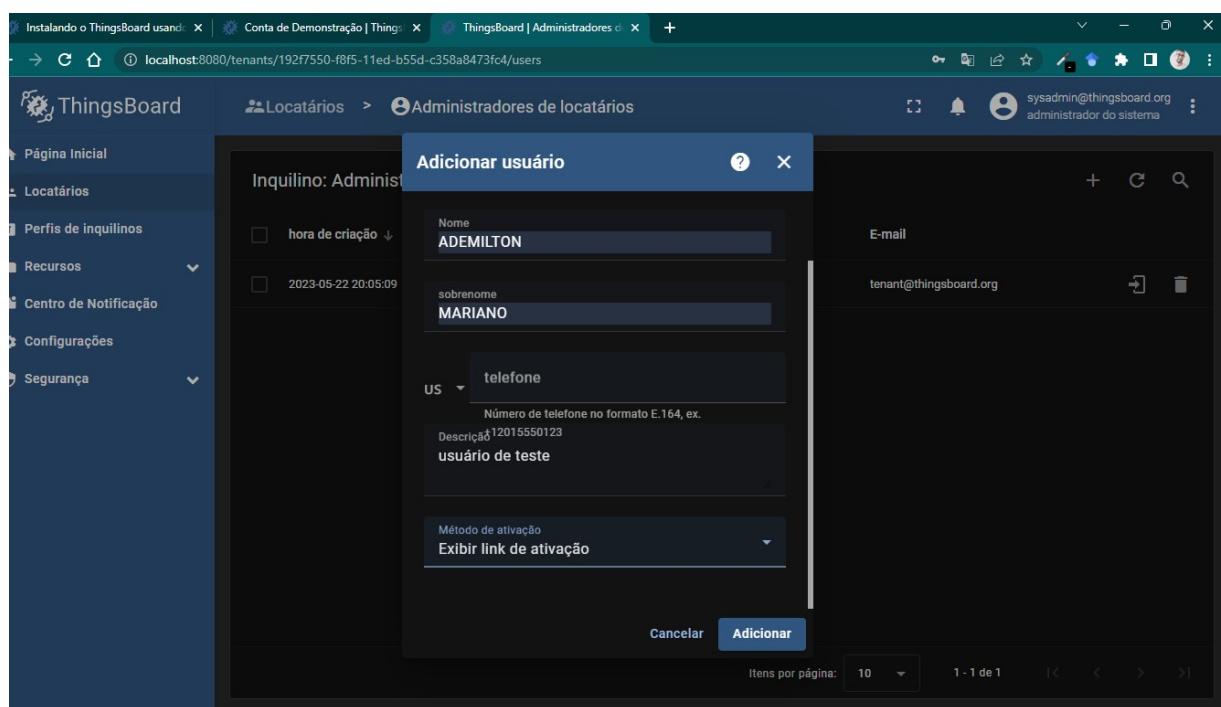
Figura 21: Passo 1 - Página inicial do sistema para o administrador.



The screenshot shows the "Adicionar perfil de inquilino" (Add tenant profile) dialog box. The "Nome*" field is empty. A note below it says: "Processamento em contêiner isolado do ThingsBoard Rule Engine Exige microsserviço(s) separado(s) para cada locatário isolado". The "Configuração do perfil" section has two tables: "Entidades (0 - ilimitadas)" and "Número máximo de dispositivos*" (0) and "Número máximo de painéis*" (0). Below these are "Número máximo de ativos*" (0) and "Número máximo de usuários*" (0). At the bottom, there are "Cancelar" and "Adicionar" buttons, and a pagination bar showing "1 - 2 de 2" items per page.

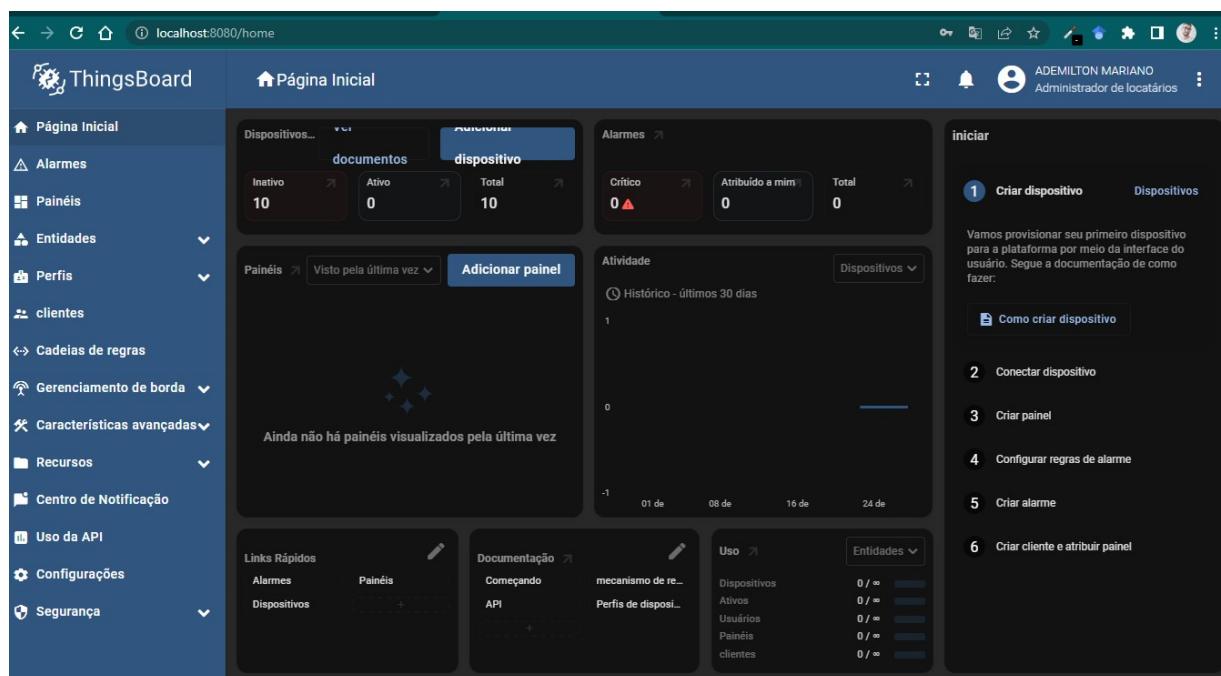
Fonte: Autor (Plataforma ThingsBoard)

Figura 22: Passo 2 - Criação de um um perfil de inquilino para testes.



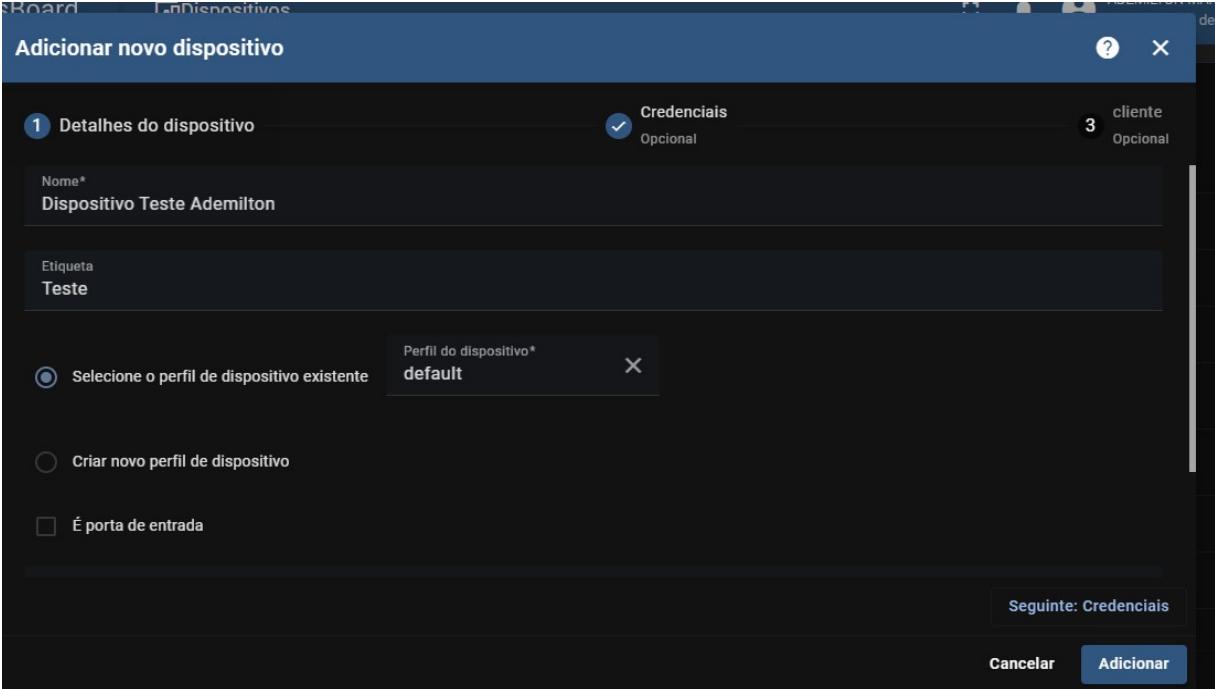
Fonte: Autor (Plataforma ThingsBoard)

Figura 23: Passo 3 - Criação de uma usuário para testes.



Fonte: Autor (Plataforma ThingsBoard)

Figura 24: Passo 4 - Página inicial do usuário de testes.



Adicionar novo dispositivo

1 Detalhes do dispositivo

Nome*
Dispositivo Teste Ademilton

Etiqueta
Teste

Perfil do dispositivo*
default

Selecionar o perfil de dispositivo existente

Criar novo perfil de dispositivo

É porta de entrada

2 Credenciais
Opcional

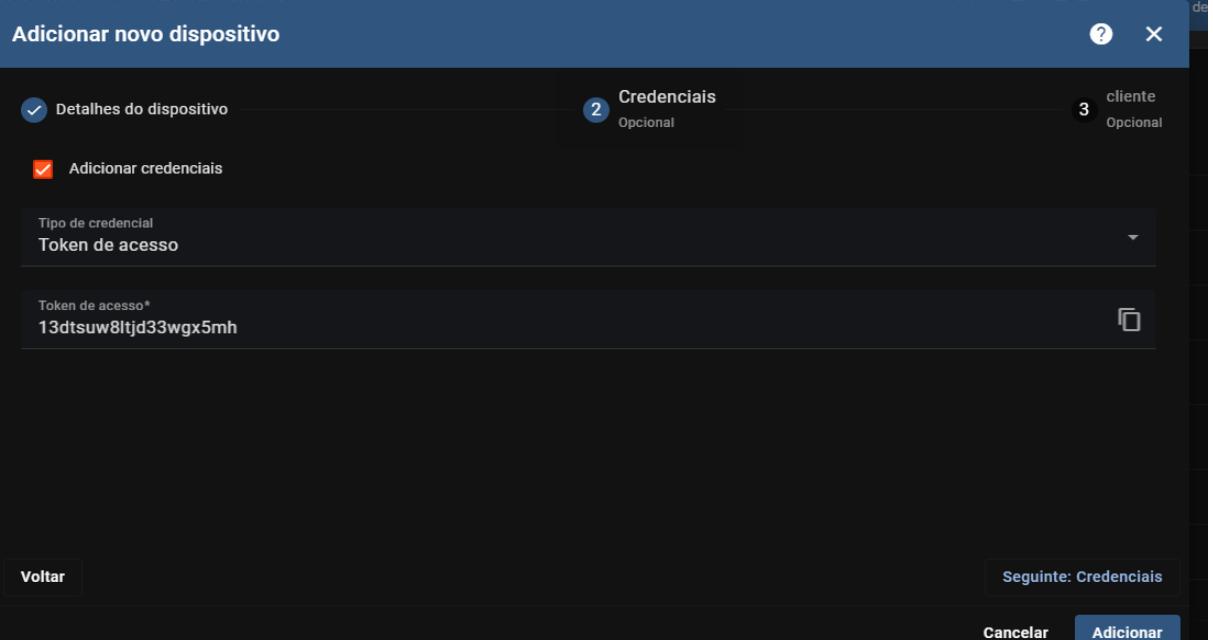
3 cliente
Opcional

Seguinte: Credenciais

Cancelar **Adicionar**

Fonte: Autor (Plataforma ThingsBoard)

Figura 25: Passo 5 -Adicionando um novo dispositivo.



Adicionar novo dispositivo

1 Detalhes do dispositivo

2 Adicionar credenciais

3 cliente
Opcional

Tipo de credencial
Token de acesso

Token de acesso*
13dtsuw8ltjd33wgx5mh

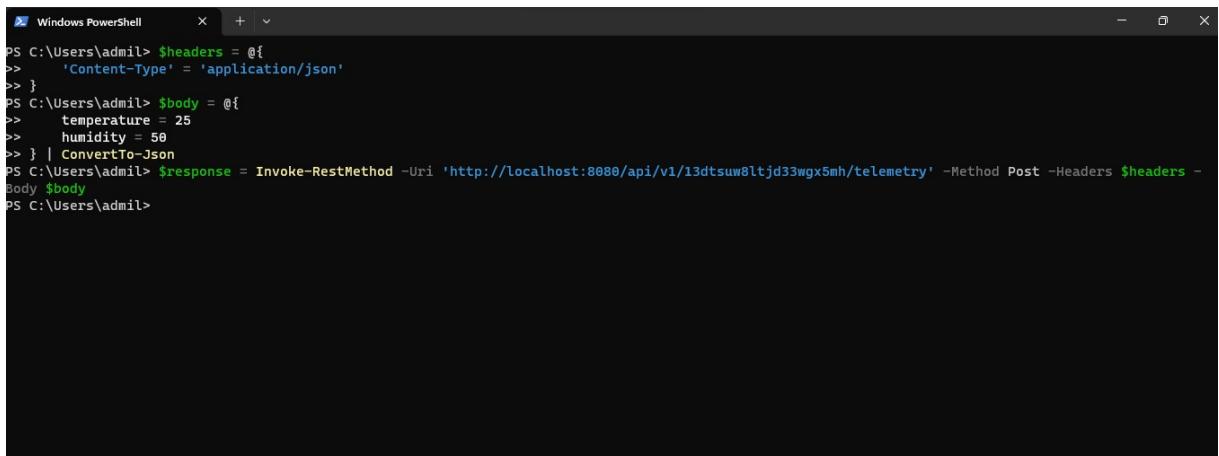
Seguinte: Credenciais

Voltar

Cancelar **Adicionar**

Fonte: Autor (Plataforma ThingsBoard)

Figura 26: Passo 6 - Gerando o token de autenticação para o dispositivo.



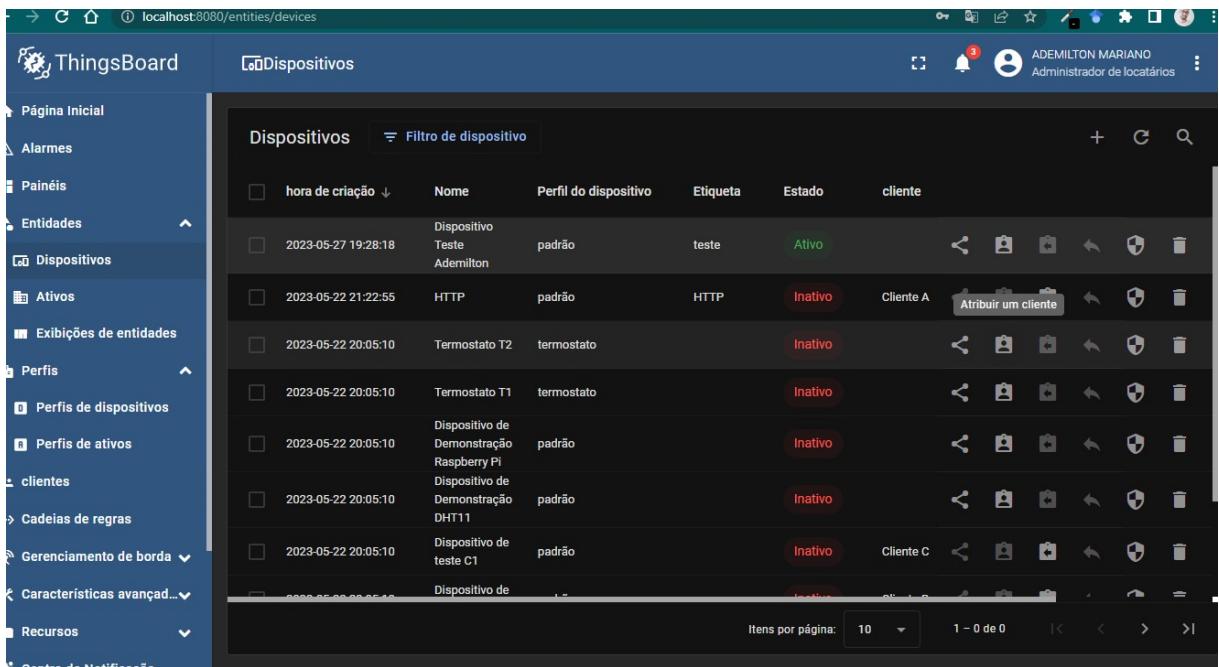
```

Windows PowerShell x + v
PS C:\Users\adminl> $headers = @{
>>     'Content-Type' = 'application/json'
>> }
PS C:\Users\adminl> $body = @{
>>     temperature = 25
>>     humidity = 50
>> } | ConvertTo-Json
PS C:\Users\adminl> $response = Invoke-RestMethod -Uri 'http://localhost:8080/api/v1/13dtsumw8ltjd33wgx5mh/telemetry' -Method Post -Headers $headers -
Body $body
PS C:\Users\adminl>

```

Fonte: Autor (PowerShell)

Figura 27: Passo 7 - Fazendo uma requisição POST pelo PowerShell.



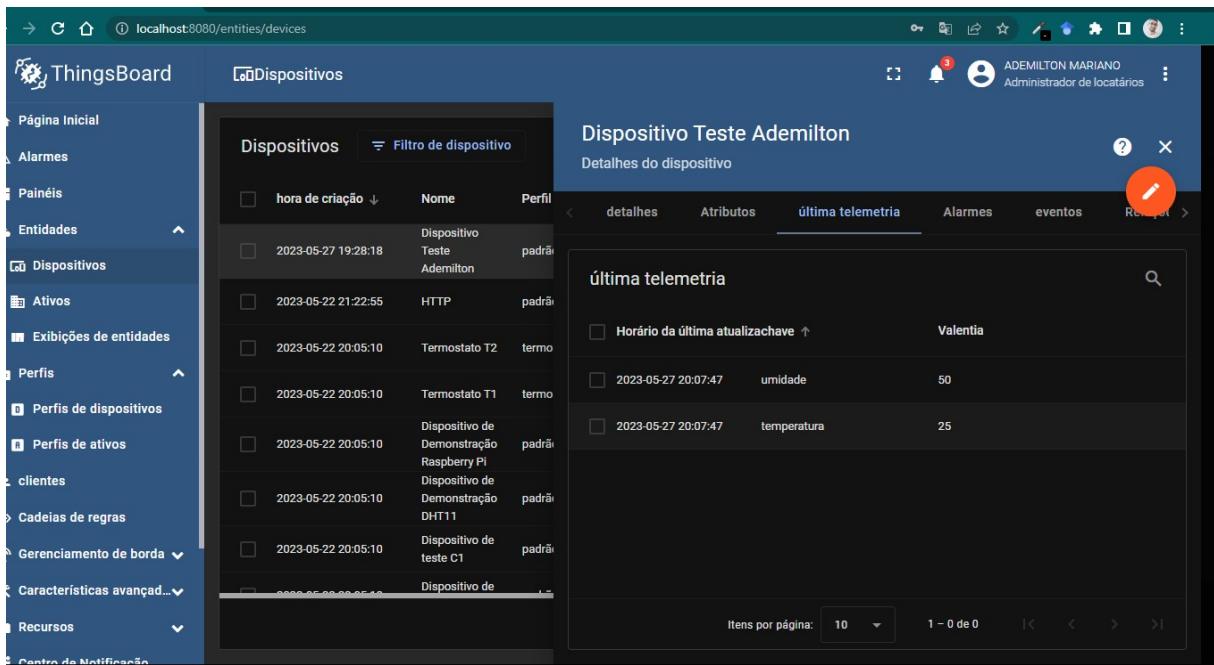
The screenshot shows the ThingsBoard web interface. On the left is a sidebar with navigation links: Página Inicial, Alarmes, Painéis, Entidades, Dispositivos (selected), Ativos, Exibições de entidades, Perfis, Perfil de dispositivos, Perfil de ativos, clientes, Cadeias de regras, Gerenciamento de borda, Características avançadas, Recursos, and Centro de Notificações. The main area is titled "Dispositivos" and contains a table with the following data:

	hora de criação	Nome	Perfil do dispositivo	Etiqueta	Estado	cliente	Actions
<input type="checkbox"/>	2023-05-27 19:28:18	Dispositivo Teste Ademilton	padrão	teste	Ativo		
<input type="checkbox"/>	2023-05-22 21:22:55	HTTP	padrão	HTTP	Inativo	Cliente A	
<input type="checkbox"/>	2023-05-22 20:05:10	Termostato T2	termostato		Inativo		
<input type="checkbox"/>	2023-05-22 20:05:10	Termostato T1	termostato		Inativo		
<input type="checkbox"/>	2023-05-22 20:05:10	Dispositivo de Demonstração Raspberry Pi	padrão		Inativo		
<input type="checkbox"/>	2023-05-22 20:05:10	Dispositivo de Demonstração DHT11	padrão		Inativo		
<input type="checkbox"/>	2023-05-22 20:05:10	Dispositivo de teste C1	padrão		Inativo	Cliente C	
<input type="checkbox"/>	2023-05-22 20:05:10	Dispositivo de	-		Inativo	Cliente B	

Items per página: 10 | 1 - 0 de 0 | < > >>

Fonte: Autor (Plataforma ThingsBoard)

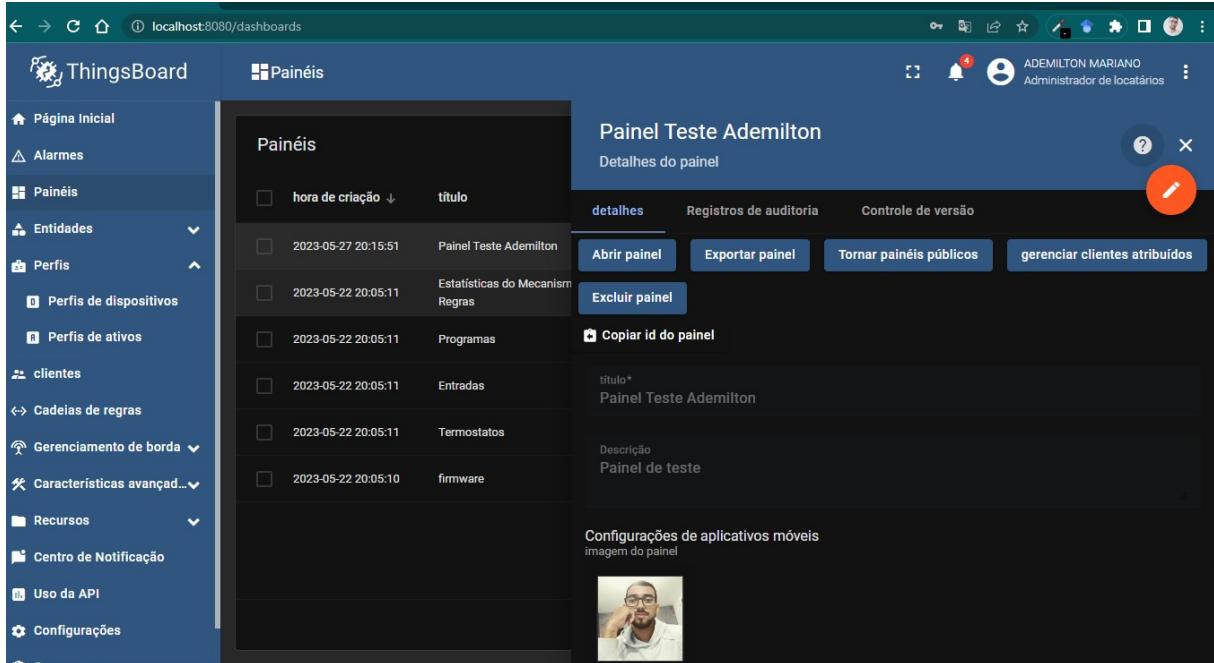
Figura 28: Passo 8 - Validando a ativação do dispositivo.



The screenshot shows the ThingsBoard interface at localhost:8080/entities/devices. The left sidebar has sections like Página Inicial, Alarmes, Painéis, Entidades, Dispositivos, Ativos, Exibições de entidades, Perfis, Perfil de dispositivos, Perfil de ativos, clients, Cadeias de regras, Gerenciamento de borda, Características avançadas, Recursos, and Centro de Notificação. The main area is titled 'Dispositivos' and shows a list of devices with columns for hora de criação (creation time), Nome (Name), and Perfil (Profile). One device, 'Dispositivo Teste Ademilton', is selected. A detailed view on the right shows 'última telemetria' (last telemetry) with data for Horário da última atualização (last update time), umidade (humidity), and temperatura (temperature). The top right shows the user 'ADEMILTON MARIANO' (Administrator of locatários) with a notification count of 3.

Fonte: Autor (Plataforma ThingsBoard)

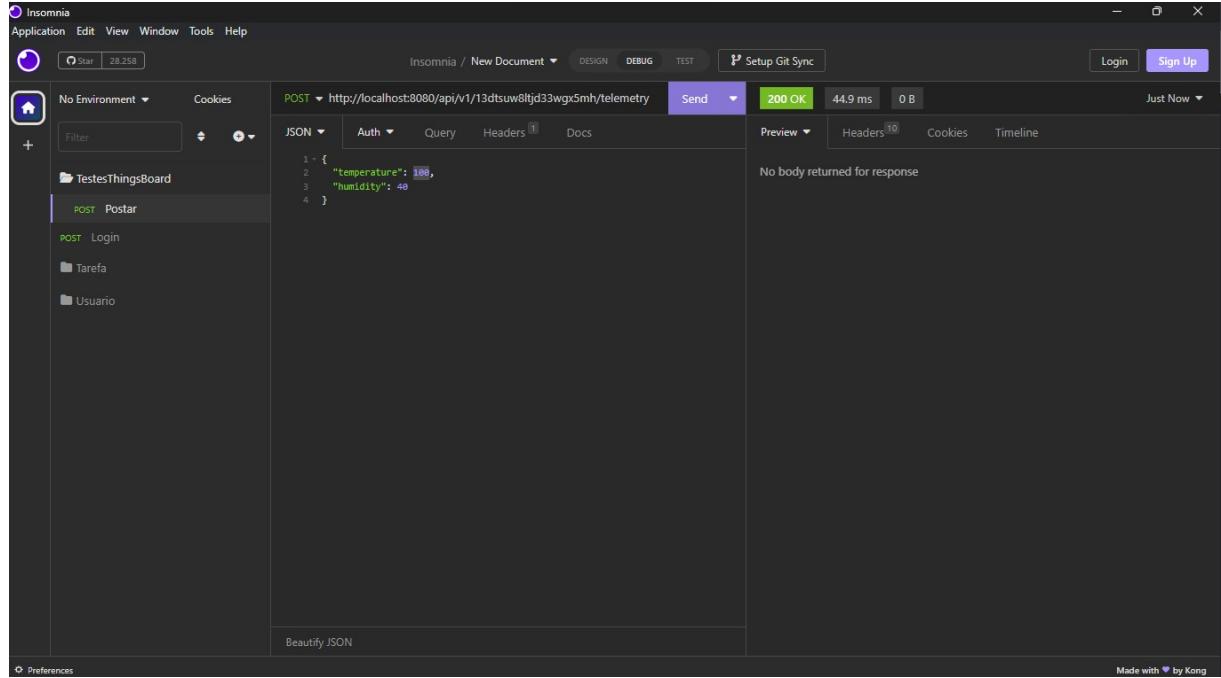
Figura 29: Passo 9 - Verificando os últimos dados enviados.



The screenshot shows the ThingsBoard interface at localhost:8080/dashboards. The left sidebar has sections like Página Inicial, Alarmes, Painéis, Entidades, Perfis, Perfil de dispositivos, Perfil de ativos, clients, Cadeias de regras, Gerenciamento de borda, Características avançadas, Recursos, Uso da API, and Configurações. The main area is titled 'Painéis' and shows a list of dashboards with columns for hora de criação (creation time) and título (title). One dashboard, 'Painel Teste Ademilton', is selected. A detailed view on the right shows 'detalhes', 'Registros de auditoria', 'Controle de versão', and buttons for 'Abrir painel', 'Exportar painel', 'Tornar painéis públicos', and 'gerenciar clientes atribuídos'. The top right shows the user 'ADEMILTON MARIANO' (Administrator of locatários) with a notification count of 4.

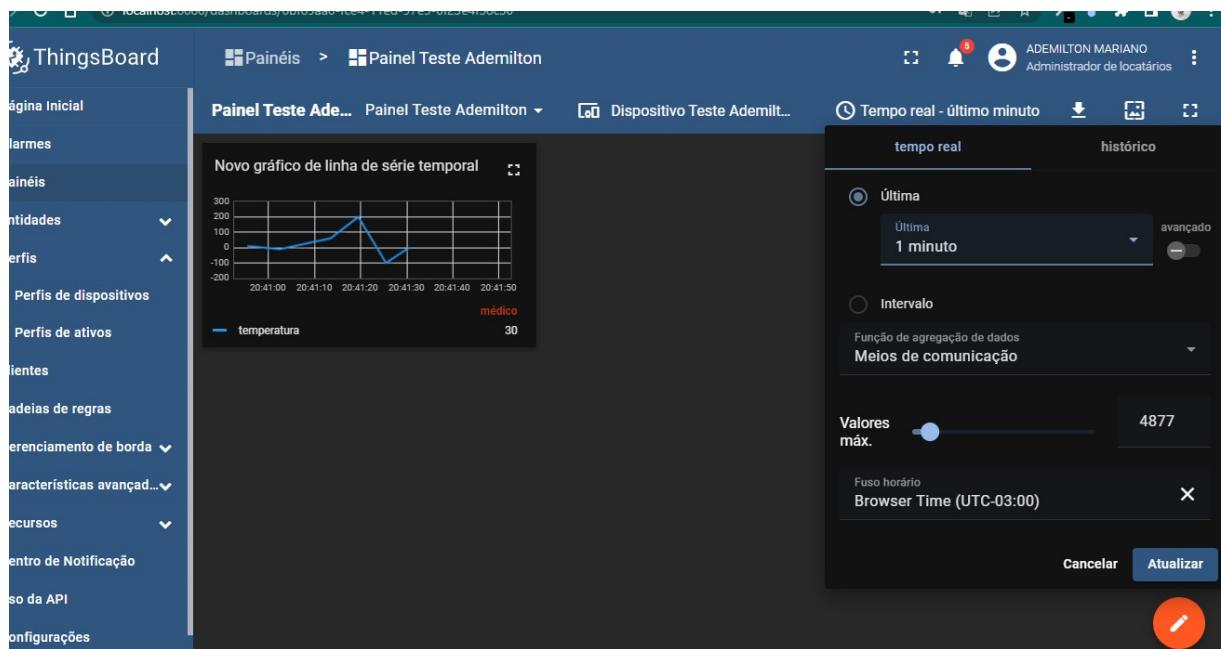
Fonte: Autor (Plataforma ThingsBoard)

Figura 30: Passo 10 - Criação do Painel.



Fonte: Autor (Insomnia)

Figura 31: Passo 11 - POST de dados feito pelo Insomnia para movimentar o gráfico.



Fonte: Autor (Plataforma ThingsBoard)

Figura 32: Passo 12 - Visualização do gráfico gerado pelos dados recebidos.

ANEXO B - Código para comunicação por HTTP

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFi.h>

#define ONE_WIRE_BUS 22 // Pino D22 do ESP32

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

const char* ssid = "Ademilton"; // Nome da rede Wi-Fi
const char* password = "*****"; // Senha da rede Wi-Fi
const char* thingsboardServer = "*****"; // Endereço IP do servidor ThingsBoard
const char* telemetryEndpoint = "/api/v1/xhmh7ek1za4qib01qwbe/telemetry"; // Endpoint de telemetria no
servidor
const char* contentType = "application/json"; // Tipo de conteúdo para a requisição HTTP

WiFiClient client;

bool sendTelemetry(String payload);
void reconnect();

void setup() {
    // Inicialize o Wi-Fi
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    // Aguarde a conexão com o Wi-Fi
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Conectando ao Wi-Fi...");
    }
}
```

```
sensors.begin();  
}  
  
void loop() {  
    while (!client.connected()) {  
        reconnect();  
    }  
  
    // Realize a leitura da temperatura  
    sensors.requestTemperatures();  
    float temperatura = sensors.getTempCByIndex(0);  
  
    // Crie um payload JSON com os dados  
    String payload = "{\"temperatura\":" + String(temperatura) + "}";  
    Serial.println(payload);  
  
    // Envie os dados via HTTP POST  
    if (sendTelemetry(payload)) {  
        Serial.println("Dados de temperatura enviados para o ThingsBoard com sucesso.");  
    } else {  
        Serial.println("Falha ao enviar dados de temperatura para o ThingsBoard.");  
    }  
  
    delay(100);  
}  
  
bool sendTelemetry(String payload) {  
    if (client.connect(thingsboardServer, 8080)) {  
        client.print("POST ");  
        client.print(telemetryEndpoint);  
        client.println(" HTTP/1.1");  
        client.print("Host: ");  
        client.println(thingsboardServer);  
        client.println("Content-Type: " + String(contentType));  
        client.print("Content-Length: ");  
    }  
}
```

```
client.println(payload.length());
client.println();
client.print(payload);

// Aguarde a resposta do servidor (opcional)
while (client.connected()) {
    if (client.available()) {
        char c = client.read();
        Serial.print(c);
    }
}

client.stop();
return true;
} else {
    Serial.println("Falha na conexão HTTP com o ThingsBoard.");
    client.stop();
    return false;
}
}

void reconnect() {
    while (!client.connected()) {
        Serial.println("Reconectando ao ThingsBoard...");
        if (client.connect(thingsboardServer, 8080)) {
            Serial.println("Conectado ao ThingsBoard!");
        } else {
            Serial.println("Falha na reconexão com o ThingsBoard. Tentando novamente em 5 segundos...");
            Serial.println(WiFi.localIP());
            delay(5000);
        }
    }
}
```

ANEXO C - Código para comunicação por MQTT

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define ONE_WIRE_BUS 22 // Pino D22 do ESP32

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

const char* ssid = "Ademilton"; // Nome da rede Wi-Fi
const char* password = "*****"; // Senha da rede Wi-Fi
const char* mqttServer = "*****"; // Endereço IP do servidor MQTT (ThingsBoard)
const int mqttPort = 1883; // Porta MQTT padrão
const char* mqttUsername = "testeMQTT"; // Nome de usuário MQTT
const char* mqttPassword = "mqteste"; // Senha MQTT
const char* mqttClient = "testeMQTT"; // ClientIDMQTT

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Conectando ao Wi-Fi...");
    }

    client.setServer(mqttServer, mqttPort);
    client.setCallback(callback);
```

```
sensors.begin();  
}  
  
void loop() {  
    if (!client.connected()) {  
        reconnect();  
    }  
  
    sensors.requestTemperatures();  
    float temperatura = sensors.getTempCByIndex(0);  
  
    String topic = "v1/devices/me/telemetry";  
    String payload = "{\"temperature\"::" + String(temperatura) + "}";  
  
    if (client.publish(topic.c_str(), payload.c_str())) {  
        Serial.println("Dados de temperatura enviados para o ThingsBoard com sucesso.");  
    } else {  
        Serial.println("Falha ao enviar dados de temperatura para o ThingsBoard.");  
    }  
  
    client.loop();  
    delay(10000); // Envie dados a cada 10 segundos  
}  
  
void reconnect() {  
    while (!client.connected()) {  
        Serial.println("Conectando ao servidor MQTT...");  
        if (client.connect(mqttClient, mqttUsername, mqttPassword)) {  
            Serial.println("Conectado ao servidor MQTT!");  
        } else {  
            Serial.println("Falha na conexão MQTT. Tentando novamente em 5 segundos...");  
            delay(5000);  
        }  
    }  
}
```

ANEXO D - Demonstração de consumo de dados na API

Para acessar os dados dessa demonstração, seguimos os passos indicados na documentação do ThingsBoard. Inicialmente, realizei o login utilizando o Insomnia para obter o token JWT. Em seguida, com o token em mãos, fiz uma requisição GET para extrair os dados desejados de um dispositivo específico, utilizando o seguinte endpoint:

<http://tb.geati.camboriu.ifc.edu.br:8083/api/plugins/telemetry/DEVICE/2b03c3d0-5719-11ee-bdd4-b1244f07481c/values/timeseries?keys=calor,humidade,temperatura&startTs=1698460561107&endTs=1701138741000&interval=3600000&limit=100000>

Os parâmetros utilizados na requisição estão explicados abaixo:

keys: lista separada por vírgulas das chaves de telemetria a serem recuperadas.

startTs: timestamp Unix que marca o início do intervalo, medido em milissegundos.

endTs: timestamp Unix que marca o fim do intervalo, medido em milissegundos.

interval: o intervalo de agregação, medido em milissegundos.

limit: a quantidade máxima de pontos de dados a serem retornados ou intervalos a serem processados.

O ThingsBoard utiliza os parâmetros startTs, endTs e interval para identificar partições ou subconsultas de agregação e executar consultas assíncronas no banco de dados, utilizando funções de agregação internas. Os dados retornados geralmente contêm o timestamp ts (timestamp Unix em milissegundos) e o respectivo valor. Abaixo, descreverei passo a passo o processo utilizado para obter esses dados:

Application File Edit View Window Tools Help

Star 31.223

Insomnia / Padrao

Base Environment Add Cookies

Filter

POST New Request

GET New Request

POST http://tb.geati.camboriu.ifc.edu.br:8083/api/auth/login Send

200 OK 450 ms 1093 B 2 Hours Ago

Preview Headers (11) Cookies Timeline

```
1: { "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZWshbnRAdghpmzMyhcmQub3jNiiwiwidXNlcijoiJiMwK1SByTATNMHML10xMwV1lWE10gHNTUSYJUUmj1SkgH2iwiC2nvCvGiJpb1lkF0V9BkE1J13dLcZxxNzaw9usQ01i01Xzj21MrQs0271g0L7Qm2yL00n10503y22YRmQ3Z0p11UcjkAm01i066iUzj3i0f2YzL5p0yis1in0lGt1CmWtCvIey0A2hSwzLzxNwJ0AxvTH3DyKLJ1bm1bgV1jpoChv1lCp1c181mxpry16Mcscus0in1hlfide1k7j5imZjg0ZADNTMh10xMwV1lWE10gHNTUSYJUUmj1SkgH2iwiYi3v2d973Xzj21C161jZE00AwLTfkD1MTMf1iM104Bg1LgwD4Hngw0044mC19.chinckyHgPQewaTygbGg8xau2Yf35SEBjKNU-ue_FTEK4CnqgvRff74tVv0xP150CxFu0VT11KcDA", "refreshToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZWshbnRAdghpmzMyhcmQub3jNiiwiwidXNlcijoiJiMwK1SByTATNMHML10xMwV1lWE10gHNTUSYJUUmj1SkgH2iwiC2nvCvGiJpb1lkJF0UhfV91R41Xs1cV2z1VBk1ki1j01MwY2jFY2ktMME4c00tMltGz2tkt2Nz1ZDM0N2c41iwlaxNz1j01dmhp0mzMyhcmQub3jNiiwiC1pJx010E3MDExBjWj51m4c1CicwMTCz1jg2MsW1axNq0saw10mzbhnlCqdqGk101iw0q2y0yyByTzLjtQyOGt0ZS01N210cYTYE3j1cf0.1ncvheA4N47gBpnmb1zwhCbt1393veL21iJh6Q2qknrcSVLW-yi2mvyft9blPzzghMSrnfdsFm", "scope": null }
```

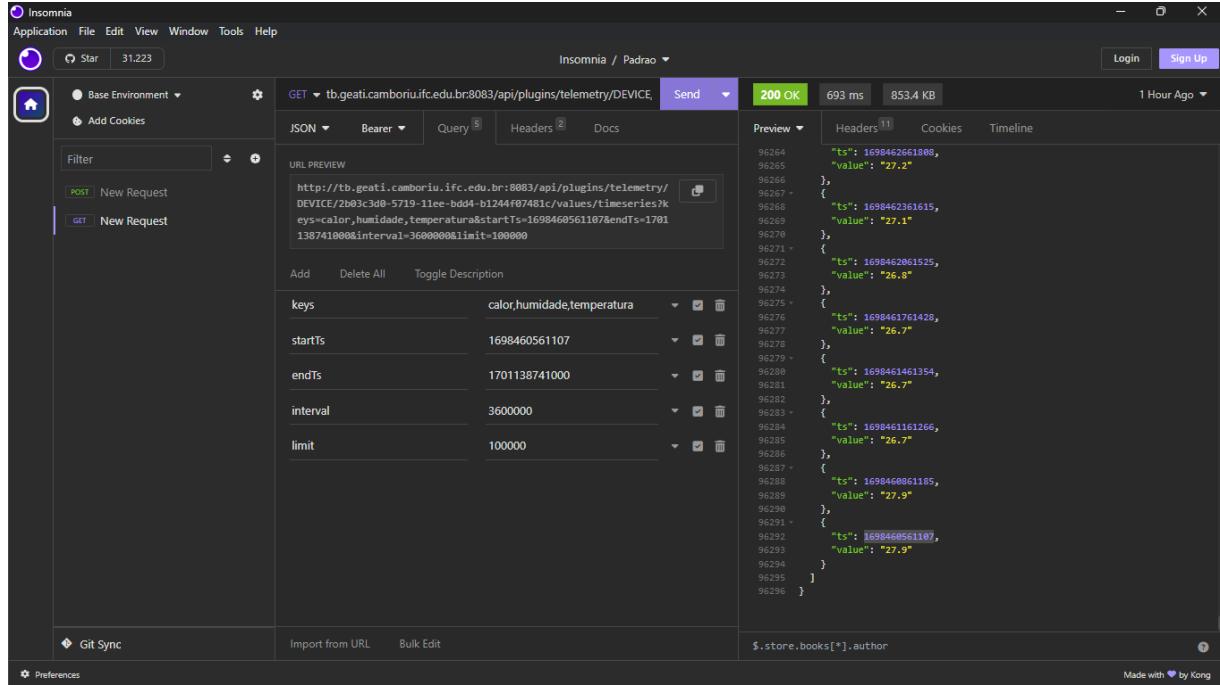
Fonte: Autor (Insomnia)

Figura 33: Passo 01 - POST de login para obter o token JWT de acesso.

The screenshot shows the Insomnia REST client interface. The top navigation bar includes 'File', 'Edit', 'View', 'Window', 'Tools', and 'Help' menus. On the right, there are 'Login' and 'Sign Up' buttons. The main workspace displays a request configuration for a 'GET' method to the URL `tb.geati.camboriu.ifc.edu.br:8083/api/plugins/telemetry/DEVICE/2b03c3d0-5719-11ee-bdd4-b1244f07481c/values/timeseries?`. The request is set to 'JSON' format and includes a 'Bearer' token. A large text input field contains a long JWT token: `eyJhbGciOiJIUzIwMiJ9.eyJzdWIiOiJ0ZW5hbnpRadGhpbmdzYm9hcmQub3JniwidXNlcklkjoiMjEwM2I5YTA1NTM1Mi0xMWVlLWE1OGMTNTUsYjU0MjI5MGMzIwiic2NvcG`. The left sidebar features a 'Base Environment' dropdown, a 'Filter' search bar, and two 'New Request' buttons for 'POST' and 'GET' methods.

Fonte: Autor (Insomnia)

Figura 34: Passo 02 - Usando o token para fazer a autenticação.



Fonte: Autor (Insomnia)

Figura 35: Passo 03 - GET passando os parâmetros para obter os dados desejados.

Uma vez obtidos os dados, é possível manipulá-los conforme necessário, seja salvando-os em um banco de dados alternativo, gerando relatórios ou qualquer outra aplicação que melhor se adeque ao seu propósito. Neste exemplo específico, apliquei um filtro de um período de 3 meses, com intervalos de uma hora para cada conjunto de dados coletados. Estes dados consistem em medições de calor, temperatura e umidade provenientes do dispositivo do IFCria, que está atualmente conectado e operante no laboratório LabMaker.

Após a obtenção desses dados, podem ser executadas diversas ações, desde análises mais aprofundadas até a integração com outros sistemas ou a geração de insights valiosos para os objetivos do projeto ou pesquisa em questão. Este relatório e códigos de implementação, se encontram disponíveis no seguinte endereço: [ademilton-mariano/Projeto_Integrador_II \(github.com\)](https://github.com/ademilton-mariano/Projeto_Integrador_II)