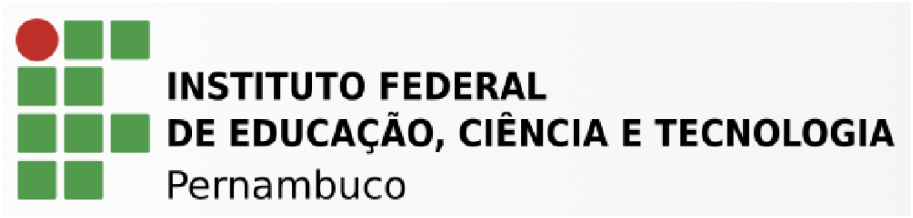


Campus Recife

Tecnólogo em Análise e Desenvolvimento de Sistemas

AIRPOWER APP: SISTEMA PARA CONFIGURAÇÃO DE AMBIENTE IOT

Trabalho de Conclusão de Curso de Graduação

por

JOÃO VICTOR TORPE SILVA

Orientador(a): Prof Meuse Nogueira de Oliveira Junior, DSc

Recife, 2026

JOÃO VICTOR TORPE SILVA

AIRPOWER APP: SISTEMA PARA CONFIGURAÇÃO DE AMBIENTE IOT

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, no Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE), Campus Recife.

Orientador(a): Prof Meuse Nogueira de Oliveira Junior, DSc

Recife

2025

S586a
2026 Silva, João Victor Torpe.
Airpower App : sistema para configuração de ambiente IoT. / João Victor Torpe Silva. --- Recife: O autor, 2026.
32f. il. Color.

TCC (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Pernambuco, 2025.

Inclui Referências.

Orientador: Professor Dr. Meuse Nogueira de Oliveira Júnior.

1. Desenvolvimento de sistemas. 2. Manutenção preditiva. 3. Internet das Coisas. 4. Integração de Sistemas. I. Título. II. OLIVEIRA JÚNIOR, Meuse Nogueira de. (orientador). III. Instituto Federal de Pernambuco.

CDD 003 (23.ed.)

Catalogação na fonte: Emmely Silva CRB4/1876

JOÃO VICTOR TORPE SILVA

AIRPOWER APP: SISTEMA PARA CONFIGURAÇÃO DE AMBIENTE IOT

Trabalho de Conclusão de Curso apresentado, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco – Campus Recife.

Aprovado em: 13 de 11 de 2025.

Banca Examinadora:

Prof. Dr. Meuse Nogueira Oliveira Junior (Orientador)
Instituto Federal de Pernambuco

Prof. Me. Tiago Lins Falcão
Instituto Federal de Pernambuco

Prof. Me. Paulo Abadie Guedes
Instituto Federal de Pernambuco

Recife

2025

RESUMO

Cerca de 40% do consumo de energia global são decorrentes de prédios, tornando essencial o monitoramento constante e a adoção de estratégias que reduzam custos operacionais e minimizem impactos ambientais (Rocha et al.). Soluções voltadas à gestão do consumo e à manutenção preditiva de ar-condicionados são de fundamental importância nesse contexto. Este projeto propõe o desenvolvimento de um sistema que viabiliza soluções de monitoramento de ar-condicionado, oferecendo uma infraestrutura responsável pelo levantamento dos dados necessários para o monitoramento, bem como pela configuração de dispositivos IoT envolvidos na coleta do consumo energético. O sistema integra-se a uma plataforma *Open Source* de gestão IoT e banco de dados por meio de uma API REST, além de permitir a configuração de dispositivos IoT via *sockets* TCP. A API REST centraliza as operações e a comunicação entre os componentes do sistema, simplificando a implantação e fornecendo a base necessária para que soluções de monitoramento de ar-condicionado possam ser implementadas.

Palavras-chave: Manutenção Preditiva; Internet das Coisas (IoT); Integração de Sistemas; API REST.

ABSTRACT

Approximately 40% of global energy consumption originates from buildings, making constant monitoring and the adoption of strategies to reduce operational costs and minimize environmental impacts essential (Rocha et al.). Solutions focused on energy consumption management and predictive maintenance of air conditioners are of fundamental importance in this context. This project proposes the development of a system that enables air conditioner monitoring solutions, providing an infrastructure responsible for collecting the data required for monitoring, as well as configuring IoT devices involved in energy consumption measurement. The system integrates with an *Open Source* IoT management platform and database via a REST API, and also allows IoT device configuration through *TCP sockets*. The REST API centralizes operations and communication between system components, simplifying deployment and providing the foundation necessary for implementing air conditioner monitoring solutions.

Keywords: Predictive Maintenance; Internet of Things (IoT); Systems Integration; REST API.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com o avanço de tecnologias como a Internet das Coisas (IoT) e a inteligência artificial, possibilita-se desenvolver sistemas inteligentes capazes de monitorar o desempenho dos condicionadores de ar em tempo real, estimar o desgaste dos componentes e otimizar sua eficiência. Este trabalho está inserido em um projeto maior, cujo objetivo é criar um sistema estimador de desgaste e eficiência, oferecendo uma solução completa para monitoramento e manutenção de condicionadores de ar em larga escala (OLIVEIRA JUNIOR, 2023).

A proposta deste TCC consiste no desenvolvimento de uma solução integrada, composta por um aplicativo Android e uma API REST, que permite configurar dispositivos IoT via hotspot e sockets TCP, além disso, centraliza a comunicação com a plataforma ThingsBoard e cadastra os equipamentos de condicionadores de ar e as melhores redes Wi-Fi em um banco de dados relacional, incluindo suas geolocalizações. Essa abordagem permite a configuração e o registro de forma paralela, garantindo mais agilidade e eficiência na configuração dos dispositivos IoT e no levantamento dos dados necessários para as regras de negócio da solução maior.

O desenvolvimento deste trabalho seguiu a metodologia ágil Scrum, com sprints de 4 semanas para entrega de funcionalidades incrementais (Schwaber and Sutherland, 2020). O Trello foi utilizado como ferramenta de gestão do backlog, permitindo o acompanhamento do progresso do projeto e facilitando a comunicação e o planejamento das atividades (Trello, 2025).

1.1 Objetivos

Nesta seção, será apresentado o objetivo geral do trabalho, bem como os objetivos específicos.

1.1.1 Objetivo Geral

Desenvolver um sistema integrado, composto por um aplicativo Android e uma API REST, para simplificar a configuração de dispositivos IoT e o registro de aparelhos de ar-condicionado e redes Wi-FI.

1.1.2 Objetivos Específicos

- Implementar funcionalidades para configurar dispositivos IoT utilizando hotspot do Android e sockets TCP;
- Desenvolver um módulo para cadastrar aparelhos de ar-condicionado com suas geolocalizações em um banco de dados;
- Desenvolver um módulo para cadastrar, em um banco de dados, as melhores redes WI-FI para cada geolocalizações;
- Implementar banco de dados local no front-end visando aumentar a tolerância a falhas de conexão entre o front-end e back-end.
- Criar uma integração com a plataforma IoT Thingsboard para centralizar o monitoramento e a análise dos dispositivos conectados;
- Projetar a API REST para centralizar e gerenciar a comunicação entre os componentes do sistema, garantindo eficiência e escalabilidade.

1.2 Prospecção do Projeto

O projeto foi idealizado visando facilitar o registro de dados ea configuração de dispositivos IoT aplicados ao monitoramento de aparelhos de ar-condicionado. A crescente demanda por soluções para gestão de aparelhos de ar-condicionado em instituições públicas e privadas impulsionou a necessidade de um sistema que permitisse:

- Cadastro e monitoramento remoto de dispositivos IoT;
- Registro de aparelhos de ar-condicionado com suas respectivas localizações geográficas;
- Identificação das melhores redes Wi-Fi para comunicação dos dispositivos em diferentes pontos da instituição;
- Sincronização eficiente entre banco de dados local e remoto;
- Automação da implantação da API responsável pela interação entre o front-end, banco de dados e Thingsboard.

1.3 Etapas do Projeto

Para garantir o desenvolvimento estruturado e eficiente do sistema, o projeto foi dividido nas seguintes etapas principais:

1. **Análise e Planejamento:** Nesta etapa, foram definidas as necessidades do sistema, os requisitos e a viabilidade da solução.

2. **Definição da Arquitetura:** A estrutura do sistema foi projetada considerando escalabilidade e manutenção. A escolha da API REST com Spring Boot, PostgreSQL como banco de dados e a plataforma ThingsBoard para IoT foi consolidada nesta fase.

3. **Desenvolvimento do Aplicativo Android:** Implementação das funcionalidades essenciais, como login, criação de dispositivos IoT, configuração via hotspot e registro de informações no banco de dados.

4. **Desenvolvimento da API REST:** Criação dos endpoints para comunicação entre o aplicativo Android, o banco de dados e o ThingsBoard, garantindo integração e segurança na troca de informações.

5. **Implantação e Manutenção:** Configuração do ambiente de execução no servidor, containerização com Docker e automação do deploy com GitHub Actions.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Linguagens de Programação

A escolha das linguagens de programação foi pautada na necessidade de obter um desenvolvimento ágil e seguro:

- **Kotlin:** Adotado para o desenvolvimento do aplicativo Android, devido à sua sintaxe concisa, recursos de segurança como o tratamento de nulidade e interoperabilidade com Java. Além disso, Kotlin é a linguagem oficial para desenvolvimento Android. Esses atributos facilitam a manutenção do código e a integração com bibliotecas (Kotlin, 2024).
- **Java (para API REST):** Na criação da API REST, optou-se por utilizar uma linguagem compatível com o ecossistema do *Spring Boot*, garantindo robustez e escalabilidade à aplicação (Corporation, 2024).

2.2 Frameworks

A utilização de frameworks específicos acelerou o desenvolvimento e assegurou a qualidade do sistema:

- **Jetpack Compose:** O Jetpack Compose é o kit de ferramentas moderno recomendado pelo Android para a construção de interfaces de usuário (UI) nativas. Ele simplifica e acelera o desenvolvimento de UI no Android, permitindo que os desenvolvedores criem aplicativos com menos código, utilizando ferramentas poderosas e APIs em Kotlin (Compose, 2024).

  Com o Jetpack Compose, é possível definir a interface de maneira programática por meio de funções de composição que descrevem a forma e as dependências dos componentes de UI. Isso promove uma abordagem declarativa, onde os desenvolvedores descrevem o que a UI deve exibir, facilitando a criação de interfaces responsivas e dinâmicas (Compose, 2024).

  Além disso, o Compose foi projetado para se integrar perfeitamente a aplicativos Android existentes e às bibliotecas do Jetpack, permitindo uma adoção gradual e facilitando a combinação de visualizações tradicionais do Android com componentes do Compose (Compose, 2024).

- **Spring Boot:** O Spring Boot é um framework open-source baseado no ecossistema Spring, projetado para simplificar a criação de aplicações Java autônomas e prontas para produção. Ele adota uma abordagem de configurações automáticas e abstrações que reduzem a necessidade de configurações manuais, permitindo que desenvolvedores foquem na lógica de negócios sem se preocupar com detalhes complexos de infraestrutura (Spring, 2024).

  Uma das principais vantagens do Spring Boot é a capacidade de criar aplicações sem necessidade de configuração extensa e arquivos XML complexos. A inicialização de projetos pode ser feita por meio do Spring Initializr, uma ferramenta que facilita a escolha das dependências necessárias e gera um projeto configurado de forma otimizada (Initializr, 2024).

  Entre os principais recursos do Spring Boot, destacam-se:

  Servidores embutidos: suporte a Tomcat, Jetty e Undertow, permitindo que as aplicações sejam executadas sem necessidade de um servidor externo. Configuração automática: o Spring Boot identifica e configura automaticamente os componentes do sistema, reduzindo a necessidade de configurações manuais. Facilidade de integração: compatível com bancos de dados SQL e NoSQL, APIs RESTful e microsserviços.

  Gerenciamento simplificado: inclui ferramentas para monitoramento, métricas e verificações de saúde da aplicação (Spring, 2024).

2.3 IDEs

Para garantir um ambiente de desenvolvimento integrado e produtivo, foram escolhidas as seguintes IDEs:

- **Android Studio:** O Android Studio é um ambiente de desenvolvimento integrado (IDE – Integrated Development Environment) projetado especificamente para a criação de aplicativos Android. A estrutura dos projetos no Android Studio segue uma organização modular, dividida em três principais seções: (a) Manifests, onde está localizado o arquivo AndroidManifest.xml, responsável por armazenar informações essenciais para a compilação do aplicativo; (b) Java, que contém os arquivos do código-fonte; e (c) Res, que abriga os layouts escritos em Extensible

Markup Language (XML), além de recursos visuais como imagens em bitmap e strings utilizadas na interface do usuário (UI) (Google, 2024) .

O Android Studio utiliza o Gradle como sistema de compilação, permitindo automação e gerenciamento eficiente das dependências do projeto. Esse sistema pode ser executado de forma independente via linha de comando ou integrado à interface da IDE, garantindo flexibilidade na configuração. Cada projeto conta com um arquivo de compilação global que abrange toda a aplicação, enquanto cada módulo possui seus próprios arquivos de build individuais, possibilitando um gerenciamento mais estruturado e modular do desenvolvimento (Google, 2024).

- **IntelliJ IDEA:** O IntelliJ IDEA é um ambiente de desenvolvimento integrado amplamente utilizado no desenvolvimento de aplicações Java e em outras linguagens da JVM, como Kotlin, Scala e Groovy. Desenvolvido pela JetBrains, o IntelliJ IDEA se destaca por oferecer um conjunto abrangente de ferramentas que aumentam a produtividade dos desenvolvedores, proporcionando um fluxo de trabalho eficiente.

  Uma das principais características do IntelliJ IDEA é a sua inteligência no suporte ao código, que inclui análise semântica, sugestões contextuais e refatoração avançada. O IDE também conta com ferramentas de depuração e testes, permitindo a identificação e correção de erros de forma eficiente, além de integrar suporte a testes automatizados.

  Outro ponto forte do IntelliJ IDEA é a sua integração com sistemas de controle de versão como Git e GitHub, facilitando o gerenciamento colaborativo de código. Além disso, a IDE possui suporte nativo para frameworks populares como Spring Boot e Hibernate (JetBrains, 2024).

2.4 Banco de Dados Remoto PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional de código aberto, amplamente reconhecido por sua robustez e extensibilidade.O PostgreSQL é projetado para ser altamente escalável, oferecendo suporte a grandes volumes de dados.

Uma das características principais do PostgreSQL é sua conformidade com os padrões SQL, proporcionando uma interface de consulta poderosa. O sistema suporta transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), que garantem

a integridade dos dados e permitem operações confiáveis em ambientes críticos (PostgreSQL Global Development Group, 2024).

2.5 Banco de Dados local Room

O Room é uma biblioteca de persistência do Android que fornece uma camada de abstração sobre o SQLite, facilitando a implementação e garantindo a integridade dos dados. Seu principal objetivo é oferecer uma abordagem mais segura e eficiente para o armazenamento local, permitindo que os desenvolvedores manipulem os dados usando objetos Java ou Kotlin sem precisar escrever consultas SQL complexas diretamente (Google Developers, 2025).

2.6 Implantação da API no servidor

A implantação da API REST foi realizada utilizando **Docker** e **GitHub Actions**, o que oferece diversas vantagens:

- O Docker proporciona uma maneira eficiente de empacotar aplicações e suas dependências em contêineres isolados. Essa abordagem garante que as aplicações sejam executadas de forma consistente em diferentes ambientes, desde o desenvolvimento até a produção (Docker, 2024).

  Os contêineres do Docker são leves e portáteis, permitindo que os desenvolvedores construam uma aplicação em um ambiente local e a implantem facilmente em servidores, na nuvem ou em qualquer infraestrutura que suporte Docker. Cada contêiner é executado em uma camada do sistema operacional, compartilhando o kernel do host, o que resulta em uma utilização eficiente de recursos em comparação com máquinas virtuais tradicionais (Docker, 2024).

- GitHub Actions é uma plataforma de automação que permite criar fluxos de trabalho personalizados para automatizar tarefas de desenvolvimento diretamente em seus repositórios do GitHub.O GitHub Actions possibilita a execução de scripts em resposta a eventos, como push, pull requests, ou agendamentos, facilitando a integração contínua (CI) e a entrega contínua (CD) de aplicações (GitHub, 2023).

Os fluxos de trabalho do GitHub Actions são definidos em arquivos YAML, que descrevem as ações a serem executadas, a ordem em que devem ocorrer e as condições para sua execução. Esses arquivos são armazenados no diretório **.github/workflows** de um repositório, permitindo fácil acesso e edição. O GitHub fornece uma ampla variedade de ações pré-configuradas que podem ser reutilizadas, além da possibilidade de criar ações personalizadas (GitHub, 2023).

Uma das principais vantagens do GitHub Actions é sua integração nativa com o ecossistema do GitHub, permitindo acionar automaticamente fluxos de trabalho (GitHub, 2023).

2.7 Ferramenta de Versionamento

Para garantir o controle e a integridade do código-fonte durante o ciclo de desenvolvimento, adotou-se o **Git** com repositório hospedado no **GitHub**:

- Git é um sistema de controle de versão distribuído, projetado para gerenciar projetos de desenvolvimento de software de maneira eficiente e confiável.Ele foi desenvolvido para facilitar o trabalho colaborativo em projetos de código aberto e tornou-se um padrão na indústria devido à sua robustez e flexibilidade.

  Uma das principais características do Git é a sua capacidade de rastrear alterações em arquivos, permitindo que múltiplos desenvolvedores trabalhem em paralelo sem interferir no trabalho uns dos outros. O Git armazena o histórico de alterações em um repositório local, o que possibilita aos usuários reverter para versões anteriores do código, comparar alterações e criar ramificações para implementar novas funcionalidades de forma isolada (Chacon and Straub, 2014).

- GitHub é uma plataforma de hospedagem de código que utiliza o sistema de controle de versão Git.A plataforma facilita a colaboração entre desenvolvedores, proporcionando ferramentas para controle de versões, rastreamento de problemas, revisão de código e integração contínua (GitHub, 2024).

  O GitHub suporta o uso de branches, o que permite que os desenvolvedores trabalhem em novas funcionalidades ou correções de bugs de forma isolada, antes de integrar as alterações ao código principal.O GitHub também oferece ferramentas de colaboração, como pull requests e comentários em commits, que facilitam a revisão e

discussão sobre alterações no código. Isso promove uma cultura de desenvolvimento aberto e colaborativo, onde os membros da equipe podem contribuir e revisar o trabalho uns dos outros (GitHub, 2024).

Outra funcionalidade importante do GitHub é a integração com GitHub Actions, que permite a automação de fluxos de trabalho de desenvolvimento, como testes e implantações (GitHub, 2024).

2.8 Ferramenta de Gestão de Organização

A gestão do projeto é crucial para o acompanhamento do progresso e a priorização das tarefas, sendo utilizada a ferramenta **Trello**:

- O Trello é uma ferramenta de gerenciamento de projetos baseada, que utiliza o método Kanban para facilitar a organização e o acompanhamento de tarefas. O Trello permite que equipes e indivíduos visualizem e gerenciem seus fluxos de trabalho de forma colaborativa e intuitiva. A interface do Trello é composta por quadros, listas e cartões, o que proporciona uma visualização clara e simples das tarefas em andamento(Trello, 2025).

  Os quadros no Trello representam projetos ou áreas de trabalho, enquanto as listas dentro de cada quadro podem ser usadas para as etapas do projeto, como "A Fazer", "Em Andamento"e "Concluído". Os cartões são usados para representar tarefas ou itens individuais, nos quais os usuários podem adicionar descrições, checklists, datas de vencimento, etiquetas, comentários e anexos (Trello, 2025).

3 METODOLOGIA

3.1 Organização do Trabalho

Este trabalho utilizou a metodologia ágil *Scrum* (Schwaber and Sutherland, 2020). As *sprints* foram definidas com uma duração de 4 semanas, ao final das quais foram realizadas revisões e retrospectivas para garantir a evolução adequada do sistema.

3.1.1 Ferramentas Utilizadas

Durante o desenvolvimento do projeto, foram utilizadas diversas ferramentas para apoiar as etapas de planejamento, desenvolvimento e controle de qualidade:

- Trello: Utilizado para o gerenciamento do *backlog* e acompanhamento do progresso das tarefas, organizado em colunas.
- Android Studio: IDE utilizada para o desenvolvimento do aplicativo Android (Kotlin, 2024) utilizando jetpack Compose (Compose, 2024).
- IntelliJ IDEA: IDE utilizada para a criação da API REST com Spring Boot Spring (2024).
- Plataforma ThingsBoard: Utilizada como plataforma central para o monitoramento e gerenciamento dos dispositivos IoT conectados ThingsBoard (2024).
- Git e GitHub: Utilizados para controle de versão durante o desenvolvimento(GitHub, 2024).
- GitHub Actions: Utilizado na implementação da esteira de integração continua (CI) da API no servidor(GitHub, 2024).

3.1.2 Arquitetura da Solução

A arquitetura do sistema foi projetada para integrar os componentes, garantindo escalabilidade e eficiência. A solução é composta por:

Figura 1: Arquitetura

Fonte: Própria, 2025

- Aplicativo Android: Responsável pela interface com o usuário que interage com a API e configuração dos dispositivos IoT via *hotspot* e *sockets* TCP.
- API REST: Fornece os serviços necessários para o cadastramento de dispositivos e integração com a plataforma ThingsBoard e o armazenamento dos dados dos ar-condicionados e redes Wi-fi no PostgreSQL.
- Plataforma ThingsBoard: Centraliza o monitoramento e armazenamento dos dados provenientes dos dispositivos IoT.
- Banco de Dados Relacional PostgreSQL: Armazena informações sobre os aparelhos de ar-condicionados como, setor onde o aparelho se encontra, marca, capacidade em BTU, número único de patrimonio, insuflamento e as melhores redes Wi-Fi para cada geolocalização.

3.1.3 Funcionalidades desenvolvidas

- Login no Thingsboard: Essa funcionalidade permite fazer login no Thingsboard e assim acessar as outras funcionalidades que dependem de autenticação. Por meio de um formulário o usuário insere os dados necessários para a autenticação, são eles: email, senha e IP de onde o Thingsboard está hospedado. Ao enviar o formulário, os dados chegam ao endpoint de autenticação da API que redireciona as credências para a camada de serviço que faz login no Rest Client do Thingsboard.

- Criar device no Thingsboard: Essa funcionalidade cria um novo dispositivo dentro da plataforma do Thingsboard. O pré-requisito para utilizar essa funcionalidade é ter realizado o login no Thingsboard. Para criar um dispositivo dentro do Thingsboard o usuário precisa preencher o formulário informando os dados básicos para a criação, são eles: nome, tipo e descrição, após isso os dados são enviados para o endpoint da API que redireciona para a camada de serviço responsável pela criação de dispositivos no Thingsboard por meio do Rest Client.

- Configurar device via socket TCP: Essa funcionalidade permite com que o Android se conecte com a ESP32 e envie os dados necessários para que ela se autoconfigure, nos testes o comportamento da ESP32 foi emulada por um notebook. (a) o usuário deve adicionar o ID da placa a qual ele deseja se conectar, (b) após fazer input do ID e clicar no botão de 'ativar' o usuário recebe as instruções de como configurar o seu hotspot para que a ESP32 possa encontrá-lo, (c) com isso configurado o Socket TCP é aberto na porta 9090 e a placa se conecta nele por meio do IP do gateway do hotspot, por fim (d) ao clicar em 'registar' os dados de autoconfiguração são enviados.

- Registrar ar-condicionado: Essa funcionalidade permite com que o usuário registre aparelhos de ar-condicionados no banco de dados juntamente com suas respectivas geolocalizações.O usuário deve preencher um formulário com as informações necessárias para armazenar o aparelho no banco, são essas informações: setor, marca, capacidade, número de patrimônio, insuflamento. Quando o usuário aperta em 'criar' esses dados juntamente com a geolocalização, capturada automaticamente, são enviados para o endpoint da API que redireciona para a camada de serviço responsável por comunicar com o repositório e persistir os dados no banco.

- Listar dispositivos do Thingsboard: Essa funcionalidade tem como pré-requisito o login no Thingsboard, pois busca todos os dispositivos associados a uma conta e lista eles juntamente com seus respectivos tokens de acesso.

- Ver mapa: Essa funcionalidade está vinculada com a API do Google Maps e mostra a localização atual do Android visando ajudar o usuário a verificar se a sua localização está sendo capturada corretamente.

- Atualizar localização atual: Essa funcionalidade complementa a 'Ver mapa', pois caso a localização no mapa esteja desatualizada o usuário pode selecionar 'Atualizar' e assim atualizar para a localização atual. Essa funcionalidade é essencial, uma vez que a localização atual é a que será utilizada nas operações de registro no banco.

- Registar melhor rede Wi-Fi para a localização atual: Essa funcionalidade tem o objetivo de fazer o levantamento das melhores redes Wi-fi de cada geolocalização, esse levantamento é fundamental para a regra de negócio do projeto maior. Ao selecionar a opção de escaneamento de redes, o usuário pode escolher a melhor rede naquela localização para registrar no banco, após escolher a rede os dados de SSID, força do sinal e geolocalização serão salvos no banco.

- Sincronizar cache local com banco remoto: Essa funcionalidade tem o objetivo de conferir robustez à aplicação, sempre que alguma informação não conseguir ser salva no banco remoto, devido a algum erro de conexão, os dados serão salvos no banco local. E assim que possível o usuário poderá selecionar a opção de sincronização para tentar enviar os dados locais para o banco de dados remoto, se a sincronização tiver sucesso os dados sincronizados serão removidos da memória local, caso contrario os dados continuam armazenados localmente até a próxima tentativa de sincronização.

3.1.4 Desenvolvimento da API REST

Nessa etapa foi desenvolvida a API Rest que faz as operações entre o front-end Android, o banco de dados PostgreSQL e a plataforma do Thingsboard. A conexão entre a API e o Thingsboard foi feita através do Rest Client da própria plataforma, esse Client permite fazer várias operações entre elas, login e criação de dispositivos. A API está estruturada em camadas:

- **Controller**: Responsável por gerenciar as requisições HTTP, recebendo os dados do front-end e encaminhando as chamadas para os serviços apropriados. Define os endpoints da API e retorna as respostas ao cliente.

- **Entity**: Representa os modelos de dados armazenados no banco de dados. Cada entidade corresponde a uma tabela no PostgreSQL, mapeada com anotações JPA.

- **Service**: Contém a lógica de negócios da aplicação. Essa camada intermedia a comunicação entre os Controllers e os Repositories, garantindo que as regras de negócio sejam aplicadas corretamente.

- **Repository**: Responsável por interagir diretamente com o banco de dados, utilizando JPA/Hibernate. Ele fornece métodos para realizar operações como salvar, buscar, atualizar e excluir dados.

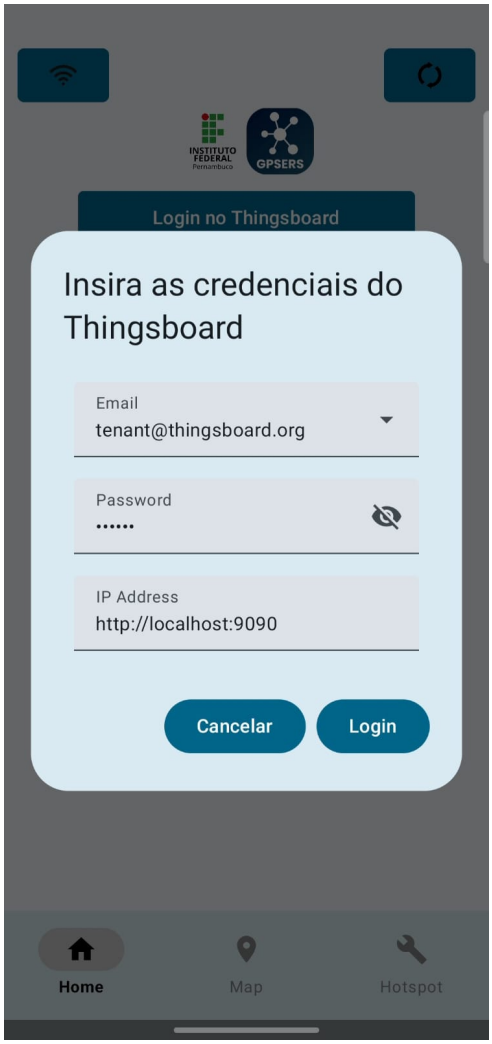
3.1.5 Implantação

A API e o banco de dados PostgreSQL foram implantados no servidor do laboratório D.E.X.T.E.R, pertencente ao bloco A do Instituto Federal de Pernambuco(IFPE) Campus Recife e ambos estão rodando em contêineres Docker. Para manter o deploy da API sempre atualizado foi implementado uma esteira de CI utilizando GitHub Actions (GitHub, 2023), logo, sempre que uma modificação é enviada para a branch main do repositório remoto a imagem atualizada da API é criada e enviada para o Docker Hub.

4 RESULTADOS

- **Login no Thingsboard**: Como mostrado na Figura 2 a autenticação foi implementada com sucesso, permitindo que os usuários acessem as funcionalidades protegidas.

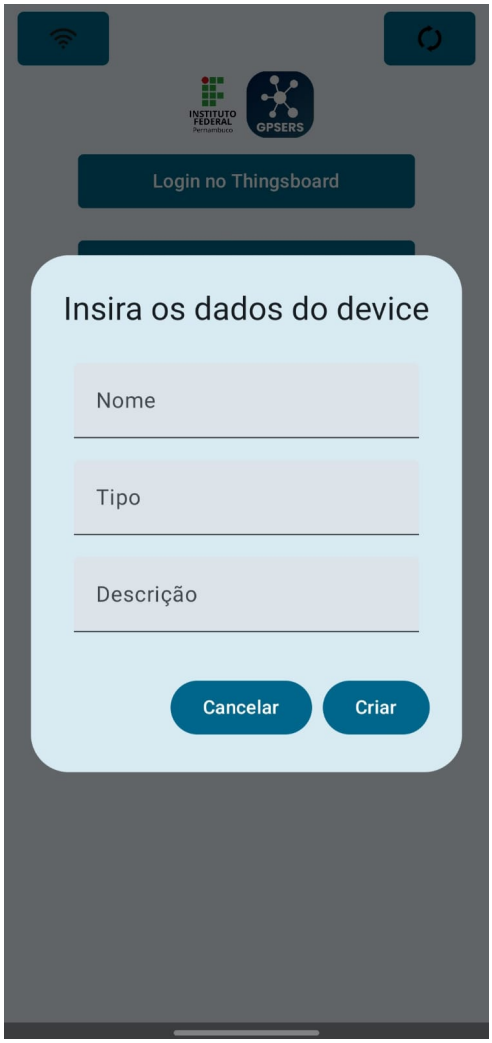
Figura 2: Formulário de login



Fonte: O autor (2025)

- **Criar device no Thingsboard**: A funcionalidade de criação de devices apresentada na Figura 3 foi testada e validada, garantindo que novos dispositivos possam ser registrados na plataforma Thingsboard sem erros.

Figura 3: Formulário para criar device no Thingsboard



Fonte: O autor (2025)

- **Configurar device via hotspot**: A tela de comunicação entre o Android e a ESP32 apresentada na Figura 4, teve a comunicação simulada por meio de um código java que emula o comportamento da ESP32 em um notebook, o aplicativo é capaz de validar o ID recebido e retornar os dados necessários para a autoconfiguração da ESP32.

Figura 4: Configurar ESP32 via hotspot

Fonte: O autor (2025)

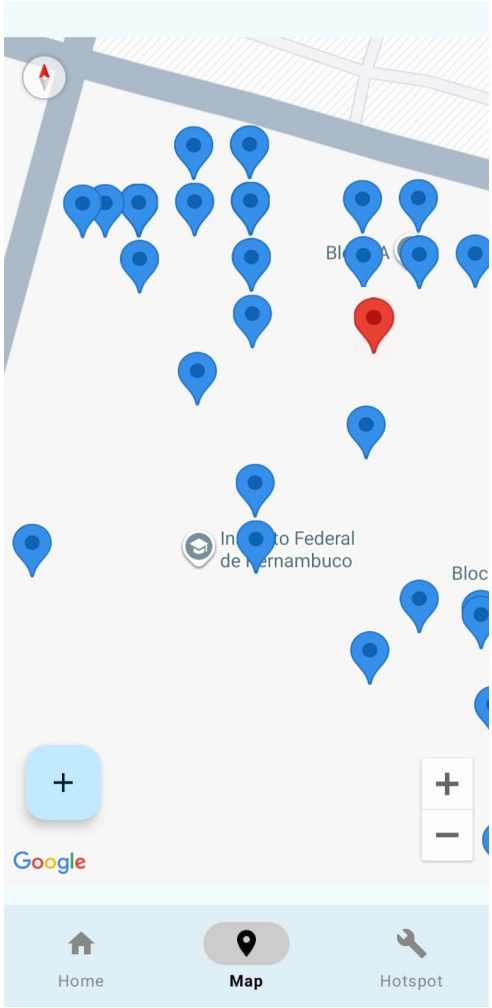
- **Adicionar ar-condicionado**: O registro de aparelhos de ar-condicionado apresentado na Figura 5 foi implementado com sucesso. A funcionalidade armazena corretamente as informações no banco de dados, incluindo geolocalização.

Figura 5: Formulário para registrar de ar-condicionados

Fonte: O autor (2025)

- **Ver mapa**: A integração com a API do Google Maps apresentada na Figura 6 foi concluída com sucesso, permitindo a exibição da localização atual do usuário e a localização dos ar-condicionados registrados.

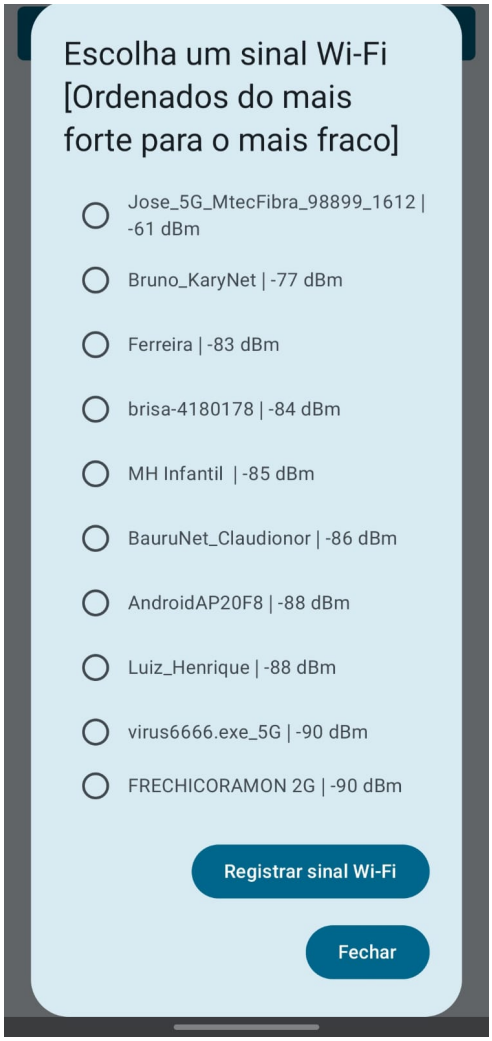
Figura 6: Mapa



Fonte: O autor (2025)

- **Registrar melhor rede Wi-Fi para a localização atual**: O escaneamento de redes Wi-Fi e o registro das melhores opções para cada localização apresentados na Figura 7 foram implementados e testados, garantindo o registro das informações no banco de dados.

Figura 7: Lista das melhores redes



Fonte: O autor (2025)

- **Listar dispositivos, Atualizar localização e Sincronização de cache**: A listagem de dispositivos associados a uma conta no ThingsBoard foi validada, garantindo a exibição correta das informações ao usuário. O sistema também possibilita a atualização manual da localização atual e a sincronização entre o cache local e o banco remoto, assegurando que os dados armazenados localmente sejam enviados assim que a conexão com a internet estiver disponível. A Figura 8 ilustra a interface que disponibiliza os botões correspondentes a essas funcionalidades.

Figura 8: Menu inicial

Fonte: O autor (2025)

5 CONSIDERAÇÕES FINAIS

O desenvolvimento do **Airpower App** demonstrou ser uma solução viável para os desafios de gestão energética em ambientes multiprediais. A integração entre o aplicativo Android, a API REST e a plataforma ThingsBoard mostrou-se eficaz para:

- Simplificar o registro de aparelhos de ar-condicionado com geolocalização
- Simplificar o registro de redes Wi-Fi com geolocalização
- Viabilizar a configuração de dispositivos IoT via conexão TCP

As tecnologias adotadas: Kotlin, Spring Boot e PostgreSQL, provaram ser adequadas aos requisitos do projeto, oferecendo desempenho satisfatório e facilidade de manutenção. A metodologia Scrum, com sprints de 4 semanas, permitiu entregas incrementais consistentes e um bom acompanhamento do progresso.

5.1 Contribuições

O sistema desenvolvido simplificou significativamente o cadastro de aparelhos de ar-condicionado e redes Wi-Fi e a configuração de dispositivos IoT,

5.2 Pontos a serem complementados

- Baixa cobertura de testes unitários no backend:
  A API REST desenvolvida possui um número pequeno de testes unitários implementados. Essa limitação compromete a capacidade de identificar erros de forma antecipada, aumentando o risco de falhas em códigos que não foram testadas. .
- Ausência de Continuous Deployment (CD) na esteira de CI/CD:
  A esteira de integração contínua (implementada com GitHub Actions) não incluiu a etapa de implantação automática em produção, exigindo intervenção manual para atualizações no servidor.

5.3 Trabalhos Futuros

- Aumentar a cobertura de testes na API:
  Implementar testes unitários e de integração para garantir maior robustez e confia-

bilidade do sistema.

- Implementar CD no servidor on-premises:
  Configurar um ambiente de deployment contínuo no servidor local para automatizar o processo de implantação e reduzir a necessidade de intervenção manual.

REFERÊNCIAS

Scott Chacon and Ben Straub. *Pro Git*. Apress, 2 edition, 2014. URL `https://git-scm.com/book/en/v2`. Acesso em: 20 out. 2024.

Jetpack Compose. *Jetpack Compose Reference Documentation*, 2024. Acesso em: 18 out. 2024.

Oracle Corporation. *Java Platform, Standard Edition Documentation*, 2024. Acesso em: 18 out. 2024.

Docker. *Docker Documentation*, 2024. Acesso em: 16 dez. 2024.

GitHub. *GitHub Documentation*, 2024. Acesso em: 2 dez. 2024.

Inc. GitHub. *GitHub Actions*, 2023.

Google. *Android Studio Documentation*, 2024. Acesso em: 3 set. 2024.

Google Developers. *Room Persistence Library - Android Developers*, 2025. URL `https://developer.android.com/training/data-storage/room`. Acessado em: 8 fev. 2025.

Spring Initializr. *Spring Initializr Documentation*, 2024. Acesso em: 3 set. 2024.

JetBrains. *IntelliJ IDEA Documentation*, 2024. Acesso em: 18 out. 2024.

Kotlin. *Kotlin Reference Documentation*, 2024. Acesso em: 18 out. 2024.

M. N. OLIVEIRA JUNIOR. Desenvolvimento de sistema estimador de desgaste e eficiência para o gerenciamento multipredial de ar condicionados. 2023.

PostgreSQL Global Development Group. *PostgreSQL Documentation*, 2024. Acesso em: 2 jan. 2025.

P. Rocha, Afal Siddiqui, and Michel Standler. Improving energy efficiency via smart building energy management systems. *Energy Efficiency Journal*, pages 203–213.

Ken Schwaber and Jeff Sutherland. *O Guia do Scrum: O Guia Definitivo para o Scrum: As Regras do Jogo*. Scrum.org, 2020.

Spring. *Spring Boot Reference Documentation*, 2024. Acesso em: 16 dez. 2024.

ThingsBoard. *ThingsBoard Documentation*, 2024. Acesso em: 2 dez. 2024.

Trello. *Trello Documentation*, 2025. Acesso em: 03 jan. 2025.