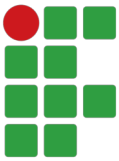

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DA PARAÍBA - CAMPUS CAMPINA GRANDE
CURSO SUPERIOR DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

ANTONIO GABRIEL ARAÚJO SILVA
JOÃO EDINALDO GOMES DOS SANTOS JUNIOR

VACMONITOR: UMA APLICAÇÃO PARA O MONITORAMENTO DE VACINAS UTILIZANDO FLUTTER E FIREBASE

Campina Grande

2023

Antonio Gabriel Araújo Silva
João Edinaldo Gomes dos Santos Junior

VACMONITOR: UMA APLICAÇÃO PARA O MONITORAMENTO DE VACINAS UTILIZANDO FLUTTER E FIREBASE

Monografia apresentada à Coordenação do Curso Superior de Bacharelado em Engenharia de Computação do IFPB - Campus Campina Grande, como requisito parcial para conclusão do curso de Bacharelado em Engenharia de Computação.

Orientador: Bruno Jácome Cavalcanti

Campina Grande
2023

S586v Silva, Antonio Gabriel Araújo.
Vacmonitor: uma aplicação para o monitoramento de vacinas utilizando Flutter e Firebase / Antonio Gabriel Araújo Silva, João Edinaldo Gomes dos Santos Junior. Campina Grande, 2023.
65 f. : il.

Trabalho de Conclusão de Curso (Curso Superior de Bacharelado em Engenharia de Computação) - Instituto Federal da Paraíba, 2023.
Orientador: Prof. Dr. Bruno Jácome Cavalcanti.

1. IoT 2. Desenvolvimento mobile 3. Desenvolvimento web I. Santos Júnior, João Edinaldo Gomes dos. II, Cavalcanti, Bruno Jácome III.Título.

CDU 004

Antonio Gabriel Araújo Silva
João Edinaldo Gomes dos Santos Junior

VACMONITOR: UMA APLICAÇÃO PARA O MONITORAMENTO DE VACINAS UTILIZANDO FLUTTER E FIREBASE

Monografia apresentada à Coordenação do Curso Superior de Bacharelado em Engenharia de Computação do IFPB - Campus Campina Grande, como requisito parcial para conclusão do curso de Bacharelado em Engenharia de Computação.

Bruno Jácome Cavalcanti
Orientador

Petrônio Carlos Bezerra
Coorientador

Anderson Fabiano Batista Ferreira da Costa
Membro da Banca

Igor Barbosa da Costa
Membro da Banca

"A verdadeira inovação vem de resolver problemas que ninguém percebeu que existiam."
(Naval Ravikant)

Agradecimentos

- A nossos pais e familiares, nosso profundo agradecimento pelo apoio constante ao longo dessa jornada acadêmica. Seu apoio foi vital, tornando cada conquista não apenas nossa, mas também de vocês;
- A nossas cônjuges, agradecemos pela paciência e compreensão durante este período desafiador;
- Aos amigos, nosso reconhecimento pela amizade duradoura e apoio moral;
- A todos os docentes do Campus IFPB Campina Grande, que contribuíram fortemente em nosso aprendizado;
- A nosso orientador, expressamos nossa sincera gratidão pela confiança em nosso trabalho e pelo valioso conhecimento compartilhado durante o desenvolvimento do Trabalho de Conclusão de Curso;
- Ao coorientador, agradecemos pelas contribuições valiosas que enriqueceram nosso projeto.

Resumo

A atual dependência de processos manuais que exigem recursos humanos representa um desafio substancial para a eficiência da Rede de Frio, que faz parte do Programa Nacional de Imunizações (PNI). A ausência de automatização torna a detecção de variações de temperatura e possíveis problemas de conservação de imunobiológicos demorada e menos eficaz. Nesse contexto, o presente trabalho descreve o processo de desenvolvimento de uma aplicação multiplataforma, chamada de VacMonitor, que é parte integrante de uma proposta tecnológica para a rede de distribuição e armazenamento de imunobiológicos, implementada por meio de uma solução baseada em Internet das Coisas (IoT). A aplicação tem como propósito a automatização de determinadas rotinas já empregadas na Rede de Frio, que ainda são executadas de forma manual, viabilizando a eficácia do processo de armazenamento. Como resultado, a aplicação apresenta funcionalidades que vão garantir um melhor controle na qualidade dos insumos, como monitoramento em tempo real da temperatura, persistência desses registros e emissão de alertas em casos de alteração de temperatura. A aplicação foi desenvolvida utilizando o framework Flutter e a linguagem Dart e permite a criação de usuários, e gerenciamento de suas responsabilidades, a partir de relacionamento com entidades de saúde da qual tenha atuação, e seus respectivos refrigeradores.

Palavras-chave: IoT. Controle e Automação. Desenvolvimento Mobile. Desenvolvimento Web. Rest API.

Abstract

The current dependence on manual processes that require human resources poses a substantial challenge to the efficiency of the Cold Chain, which is part of the National Immunization Program (NIP). The lack of automation makes the detection of temperature variations and potential issues in the conservation of immunobiologicals time-consuming and less effective. In this context, the present work describes the development process of a cross-platform application called VacMonitor, which is an integral part of a technological proposal for the distribution and storage network of immunobiologicals. This proposal is implemented through an Internet of Things (IoT)-based solution. The purpose of the application is to automate certain routines already employed in the Cold Chain, which are still manually executed, enabling the effectiveness of the storage process. As a result, the application features functionalities that ensure better control over the quality of supplies, such as real-time temperature monitoring, persistence of these records, and issuance of alerts in case of temperature changes. The application was developed using the Flutter framework and the Dart language, allowing the creation of users and the management of their responsibilities. Users can establish relationships with healthcare entities in which they operate and their respective refrigerators.

Keywords: VacMonitor. Internet of Things. Control and Automation. Mobile Development. Web Development. Rest API.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

IFPB	Instituto Federal da Paraíba
API	Application Programming Interface
REST	Representational State Transfer
PNI	Programa Nacional de Imunizações
IoT	Internet das Coisas
TSMC	Taiwan Semiconductor Manufacturing Company
BaaS	Backend as a Service
MVT	Model-View-Template
MVP	Minimum Viable Product
UML	Unified Modeling Language
URL	Uniform Resource Locator
AWS	Amazon Web Services

Sumário

1 Introdução

Os imunobiológicos, como vacinas e soros, são essenciais para prevenir e tratar doenças específicas. Eles atuam fortalecendo o sistema imunológico para combater, ou mesmo prevenir, agentes causadores de doenças. Esse sistema se trata do nosso próprio mecanismo de defesa, caracterizado pela capacidade de identificar estruturas moleculares específicas, conhecidas como antígenos, e a partir deste reconhecimento, elaborar uma resposta ativa, provocando destruição ou inativação dos mesmos (MARTíNEZ; ALVAREZ-MON, 1999).

Os imunobiológicos são produtos termolábeis e fotossensíveis, ou seja, são afetados pelas condições de temperatura e luz em que se encontram. Assim, essas variáveis se tornam fatores imprescindíveis no armazenamento dos mesmos, afetando diretamente na segurança e eficácia do material armazenado.

No Brasil, o Programa Nacional de Imunizações (PNI) é o órgão regulador encarregado de estabelecer as diretrizes e normas para o armazenamento de imunobiológicos em todo o país. Em seu manual (SAÚDE, 2017), o PNI indica, a partir de estudos da OMS, que o armazenamento de imunobiológicos deve ocorrer no intervalo de $+2^{o}$C a $+8^{o}$C, a todo momento, desde a produção até que ocorra sua aplicação.

Nesse sentido, as Redes de Frio são responsáveis – em nível nacional, estadual e municipal, pelo processo de armazenamento, conservação, manipulação, distribuição e transporte dos imunobiológicos do PNI (TINOCO, 2010).

Uma análise feita a partir de notificações de alterações de temperatura, na Rede de Frio na região noroeste de São Paulo, mostrou que das 341, 70.1 % das ocorrências foram causadas por motivos estruturais, e também, das doses que sofreram alteração de temperatura, 41.4 % foram perdidas (PATINE et al., 2021). Essa análise desperta o alerta que parte dos processos empregados no armazenamento dos imunobiológicos, não acompanham a evolução tecnológica e se encontram defasados, causando ineficiência no processo.

O acompanhamento da temperatura e documentação dos valores ainda ocorrem de maneira manual, refletindo na necessidade do uso de um recurso humano apenas para execução dessas tarefas. Além de ser um potencial gargalo nas rotinas de funcionamento de uma instituição de saúde, a dependência de intervenção manual pode ocasionar possíveis erros humanos.

Nesse sentido, foi submetido e aprovado o projeto “Sistema de Monitoramento de Qualidade de Vacinas na Cadeia de Distribuição e Armazenamento”, para o edital Nº 010/2021 - FAPESQ/PB - MCTIC/CNPq. Os autores foram os professores do Instituto Federal da Paraíba (IFPB), Petrônio Carlos Bezerra (coordenador do projeto) e Bruno Jácome Cavalcanti (professor colaborador).

O projeto proposto teve como objetivo a construção de um sistema, baseado no conceito de IoT (*Internet of Things* – Internet das Coisas), para o monitoramento de qualidade de vacinas na cadeia de distribuição e nas unidades de armazenamento, com capacidade de notificação automática para níveis de temperatura considerados críticos.

Os objetivos específicos podem ser destacados:

1. Desenvolver um protótipo, utilizando microcontrolador, para aquisição de informações, dos ambientes de armazenamento e de comunicação para transmissão dos dados coletados pelos sensores;
2. Utilizar uma plataforma na nuvem, desenvolvendo um banco de dados, para armazenamento e disponibilização dos dados obtidos;
3. Desenvolver um *front-end* para plataforma Android, com interface baseada em *dashboard*, para compartilhamento dos dados coletados, além da implementação de sistema de alerta.

Nesse sentido, esse trabalho de conclusão de curso busca atender uma demanda levantada pelos professores citados, envolvendo os pontos 2 e 3 dos objetivos específicos apresentados. A aplicação tem como propósito a integração de sensores e microcontroladores nos locais de armazenamento da Rede de Frio. Essa integração possibilitará a coleta e transmissão de dados de temperatura através de protocolos de comunicação para a camada da aplicação, onde os dados serão processados e então repassados para a próxima terminação, onde serão disponibilizados para os usuários finais.

1.1 Objetivos

1.1.1 Objetivo geral

Este trabalho tem como objetivo o desenvolvimento de uma aplicação multiplataforma, responsável por desempenhar funcionalidades utilizadas pelos usuários finais, partes interessadas pela administração dos pontos de armazenamento da Rede de Frio. Trazendo ênfase especial na modernização do monitoramento e documentação de registros de temperatura, captados por um protótipo formado por um microcontrolador e sensor de humidade, presentes nas instalações de instituições de saúde, bem como à introdução de um sistema de emissão de notificações automatizadas, em casos de alteração crítica de temperatura.

1.1.2 Objetivos específicos

(i) Integração da aplicação multiplataforma com sistema de clouding usado pela aplicação IoT, permitindo o resgate de registros enviados pelos dispositivos físicos nos pontos de armazenamento;

(ii) Concepção de protótipo de baixa fidelidade da interface, e modelagem do sistema;

(iii) Implementação de fluxo de autenticação de usuário;

(iv) Desenvolvimento do fluxo de cadastro de entidade 'Refrigerador', representante do ponto físico de armazenamento;

(v) Desenvolvimento de fluxo de cadastro de usuário;

(vi) Desenvolvimento da área de resgate de detalhes de um refrigerador, obtendo em tempo real registros de leitura de temperatura, além de informações estatísticas a respeito;

(vii) Implementação do módulo responsável pela emissão de notificações de forma automática.

2 FUNDAMENTAÇÃO TEÓRICA

Neste tópico, são apresentados conceitos de suma importância para o melhor entendimento do desenvolvimento do projeto. Serão apresentadas as definições de: aplicação WEB, computação móvel, Rest API, Internet das Coisas, Microcontroladores, Sensores e Atuadores, Flutter, Dart, Firebase, Python e Django REST.

2.1 Computação Móvel

Computação móvel pode ser representada como um novo paradigma computacional que permite que usuários desse ambiente tenham acesso a serviços independentemente de sua localização podendo, inclusive, estarem em movimento (FIGUEIREDO; NAKAMURA, 2003). Dessa forma, a computação móvel amplia o conceito tradicional de computação distribuída. Isso é possível graças à comunicação sem fio que elimina a necessidade do usuário manter-se conectado à uma infraestrutura fixa e, em geral, estática (MATEUS; LOUREIRO, 1998).

Sendo assim, é evidente que a computação móvel é a ideia de que os usuários podem processar dados ou executar tarefas digitais em dispositivos móveis, uma vez que um dispositivo móvel é formado por hardware e software, incluindo um sistema operacional e aplicações (MATEUS; LOUREIRO, 1998). Atualmente, há uma gama de sistemas operacionais para dispositivos móveis, bem como aplicações que podem funcionar somente em determinado sistema operacional ou até mesmo em mais de um. Os sistemas operacionais mais utilizados são Android (71.47 %), iOS (27.88 %), além de outro 0.65 % (ALMENARA; CIRIACO, 2022).

2.2 Microcontroladores

Microcontroladores são circuitos integrados que possuem em seu interior todos os componentes necessários ao seu funcionamento dependendo unicamente da fonte de alimentação externa (KERSCHBAUMER, 2013).

De acordo com (CHASE; ALMEIDA, 2007), a vantagem dos microcontroladores é que além de possuir os periféricos integrados a um único chip, são responsáveis por executar e armazenar os programas escritos para eles (*firmware*). A partir disso, outro conceito que pode ser inserido nesse contexto são os sistemas embarcados, onde: o sistema embarcado geralmente é uma solução formada de microcontrolador e software (*firmware*) dedicados e específicos para desempenhar as funções operacionais de um equipamento/produto para o qual foi projetado e desenvolvido (CHASE; ALMEIDA, 2007).

Sobre as aplicações dos microcontroladores:

> Os microcontroladores são utilizados em praticamente todos os dispositivos eletrônicos digitais que nos cercam, como por exemplo, centrais de alarme, teclados do computador, monitores, discos rígidos de computador, relógio de pulso, máquinas de lavar, forno de micro-ondas, telefones, rádios, televisores, automóveis, aviões, impressoras, marca passos, calculadoras, etc (KERSCHBAUMER, 2013, p.11)

Sobre o ESP32 (Figura 1) (que é microcontrolador utilizado no desenvolvimento relativo ao *hardware* do projeto), de acordo com (PEREIRA, 2021), é uma série de microcontroladores de baixo custo e baixo consumo de energia, que foi criado e desenvolvido por Espressif Sistemas, uma empresa Chinesa com sede em Xangai e é fabricado pela Taiwan Semiconductor Manufacturing Company (TSMC).

Este dispotivo pode ser usado na automação residencial, controlando lâmpadas, portões, aparelhos de TV, aparelhos de Som, motores como bombas da água e de piscina, câmeras de segurança e alarmes (PEREIRA, 2021).

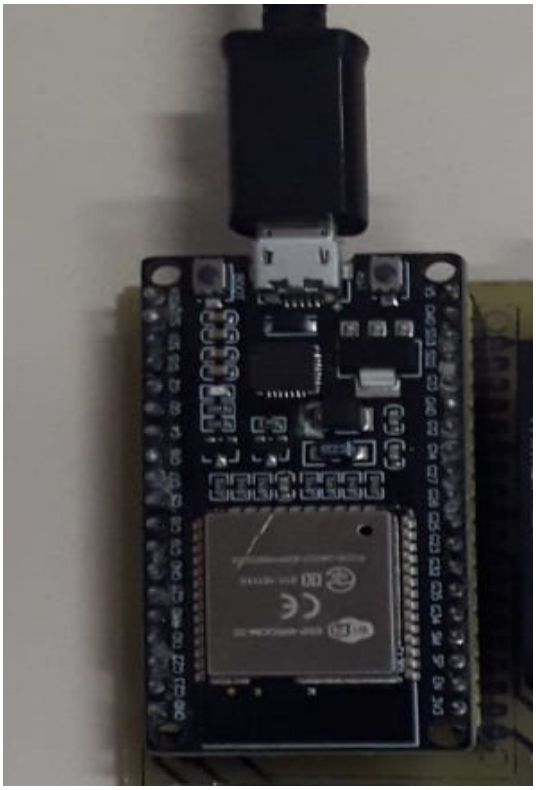

Figura 1 – Microcontrolador Esp32

Fonte: Elaborado pelo autor

2.3 Sensores e Atuadores

Os atuadores podem ser definidos como (WENDLING, 2010):

1. Dispositivos que modificam uma variável controlada;
2. Recebem um sinal proveniente do controlador e agem sobre o sistema controlado;
3. Geralmente trabalham com potência elevada.

Em contrapartida, o sensor é o termo empregado para designar dispositivos sensíveis à alguma forma de energia do ambiente (WENDLING, 2010). Esse tipo de energia pode ser: luminosa, térmica, cinética, etc.

Relativo aos atuadores, tem-se como exemplo: Relés, Motores, Solenóides e etc. Já sobre os sensores, existem uma gama deles dentro de cada um de seus tipos, que são: Sensores Mecânicos, Sensores Fotoelétricos, Sensores Térmicos, Capacitivo, Indutivo, Ultrassônico e etc.

O sensor utilizado no *hardware* do dispositivo que faz as medições é o DS18B20 (Figura 2), onde fornece 9 bits para medições de temperatura Celsius de 12 bits e tem um função de alarme com pontos de disparo superiores e inferiores não voláteis programáveis pelo usuário (INTEGRATED, 2019). De acordo com Maxim Integrated (INTEGRATED, 2019), pode-se citar algumas aplicações de sensor como:

1. Controles termostáticos;
2. Sistemas Industriais;
3. Produtos de consumo;
4. Termômetros;
5. Sistemas termicamente sensíveis.

Figura 2 – Sensor de Temperatura DS18B20

Fonte: ELETROGATE

2.4 Flutter e Dart

Flutter é uma estrutura de código aberto do Google para criar aplicativos multiplataforma e compilados nativamente a partir de uma única base de código (FLUTTER, 2023). Assim, o Flutter permite que os desenvolvedores criem aplicativos da Web, desktop e multiplataforma executados em dispositivos Android e iOS. O Flutter usa uma linguagem de programação reativa chamada Dart, tornando o desenvolvimento mais rápido e fácil do que os métodos tradicionais (ADSERVIO, 2022).

O Dart é uma linguagem otimizada para o cliente para aplicativos rápidos em qualquer plataforma. Comumente a linguagem Dart é associada ao Flutter, entretanto, também existe Dart para backend apesar de nos dias atuais ainda não ser uma abordagem tão comum, As linguagens de programação mais frequentemente empregadas no desenvolvimento de aplicações para o backend são listadas em ordem decrescente de utilização: Python, Rails, Java, C#, Swift, PHP, JavaScript, Ruby, C, e C++ (Redação XP Educação, 2022)

Sobre o funcionamento do framework[1] em questão, superficialmente, o Flutter é uma estrutura de interface de usuário reativa e declarativa, na qual o desenvolvedor fornece

[1] Conjunto de bibliotecas predefinidas que fornece funcionalidades e ferramentas comuns para o desenvolvimento de aplicações.

um mapeamento do estado do aplicativo para o estado da interface, e a estrutura assume a tarefa de atualizar a interface em tempo de execução quando o estado do aplicativo muda (FLUTTER, 2023).

É possível listar as vantagens em se utilizar o Flutter (PEDRO, 2022):

1. Multiplataforma
2. Acesso a recursos nativos
3. Maior desempenho
4. Fácil de aprender
5. Custo-benefício

Com relação à característica de ser multiplataforma, além da possibilidade da criação de aplicações para dispositivos Android e iOS, também é possível gerar uma versão web, o que gera uma maior portabilidade e opção de uso para o desenvolvedor, uma vez que, com o mesmo código é possível ter uma aplicação para vários ambientes.

Ainda sobre vantagens, Dart é uma linguagem com estrutura muito semelhante a linguagens focadas em orientação à objetos, podendo ser citado o Java como exemplo. Ou seja, nativamente consegue-se (com o Dart) utilizar de conceitos como: Classes e Objetos, Abstração, Polimorfismo, Encapsulamento, dentre outras técnicas características da programação orientada a objetos (ALBERTO, 2023). Portanto, desde que o desenvolvedor conheça estes conceitos, a etapa de desenvolvimento de software se torna mais eficiente, fácil de entender, e de manter, isso devido a aplicação dessas técnicas permitir a redução e ou até total eliminação da duplicação de código, além de, reduzir a complexidade, simplificando o desenvolvimento, teste e manutenção do software.

2.5 Firebase

O Firebase é uma plataforma *Backend as a Service* (Baas) criada sobre a infraestrutura do Google com o objetivo de auxiliar os desenvolvedores a acelerar o desenvolvimento de aplicações, sem a necessidade de gerenciar a infraestrutura (ANDRADE, 2018). O uso dessa plataforma facilita e agiliza o desenvolvimento de um sistema, considerando que tal sistema possa necessitar de banco de dados, autenticação, *real-time*, armazenamento de arquivos, dentre outras funcionalidades cruciais de uma aplicação web.

O próprio site de produtos do Firebase[2] apresenta 3 categorias, onde cada uma delas abrange diferentes tipos de serviço (GOOGLE FIREBASE, 2023):

[2] https://firebase.google.com/products-build?authuser=2

- **Criação:** Fazer o lançamento do mercado e oferecer valor para os usuários mais rápido.
- **Liberar e Monitorar:** Melhorar a qualidade do app com menos tempo e esforço.
- **Engajamento:** Otimizar a experiência no app.

Dentre esses serviços, pode-se citar alguns mais populares como:

- **Firebase Realtime Database:** É um banco de dados NoSQL hospedado na nuvem que possibilita a sincronização de dados JSON em tempo real.
- **Cloud Messaging:** Oferece uma conexão confiável e com baixo consumo de bateria entre servidor e dispositivos para enviar e receber mensagens e notificações no Android, no iOS e na Web sem custo.
- **Cloud Firestore:** O Cloud Firestore é um banco de dados de documentos NoSQL[3] que permite armazenar, sincronizar e consultar dados facilmente para seus apps para dispositivos móveis e da Web, em escala global.
- **Firebase Authentication:** Oferece uma solução de identidade completa, compatível com contas de e-mail/senha, autenticação por telefone, *login* do Google, Twitter, Facebook, GitHub e outros.

2.6 Rest API

Uma *Application Programming Interface* (API) é um conjunto de protocolos e regras que possibilitam a comunicação entre diferentes softwares. Esta é a responsável por definir formatos e métodos que um sistema pode utilizar para requisitar informações de outro sistema ou serviço. O *Representational State Transfer* (REST) é uma arquitetura de software que, baseada em um conjunto de regras, impõe condições de como uma API deve funcionar. As restrições determinadas pela Rest são:

- **Separação entre servidor e cliente:** Aplicações existentes no servidor e no cliente devem ser separadas (Amazon Web Services, 2023).
- **Estado ausente:** O servidor completa cada requisição do cliente independentemente, deixando o sistema ausente de estados.
- **Armazenamento em cache:** A API deve utilizar armazenamento em cache, seja no cliente ou intermediário, evitando chamadas para recursos usados previamente.

[3] O termo 'NoSQL' se refere a tipos não relacionais de bancos de dados, e esses bancos de dados armazenam dados em um formato diferente das tabelas relacionais.

- **Interface uniforme:** Indica que o servidor transfere informações de forma padrão, agrupando conceitos como identificação única, manipulação de recursos por meio de representação, comunicação clara, e utilização de links para navegação na aplicação.

2.7 Django e Python

Django é um framework de desenvolvimento de alto nível, publicado em código aberto em 2005, que disponibiliza ferramentas que possibilitam o desenvolvimento de aplicações de forma eficiente e ágil. O Django segue o padrão de projeto MVT (Model-View-Template), onde o *Model* representa a estrutura responsável por cuidar das regras de negócio; a *View* lida com a descrição da informação que é apresentada pela aplicação; e o Template a representação visual da informação. Esse padrão facilita a separação de responsabilidades da aplicação, fazendo com que a escalabilidade seja feita de forma mais eficiente (DJANGO, 2023).

O Django é desenvolvido e sustentado até os dias atuais pela linguagem Python. Python é uma linguagem de programação interpretada, orientada a objetos e de alto nível. As estruturas de dados de alto nível implementadas em Python, combinadas com a tipagem dinâmica, tornam a linguagem, e assim também o Django, bastante atrativos para o desenvolvimento rápido de aplicações (DJANGO, 2023).

2.8 Internet das Coisas

Internet das coisas foi um termo concebido/criado em 1999 por Kevin Ashton (ASHTON, 2009), cientista da computação que pesquisava uma solução na distribuição de produtos da empresa P&G usando etiquetas de identificação, que contribuíram no monitoramento de informações como identificação de lotes e localização.

O conceito geral da Internet das Coisas pode ser considerado aquele que transforma objetos físicos, tradicionais do dia a dia, em inteligentes, fazendo isso por meio da exploração de bases tecnológicas como sistemas embarcados, tecnologias de comunicação, fazendo com que os objetos físicos vejam, ouçam, tomem decisões, executem tarefas, "conversem", compartilhem informações e coordenam decisões (AL-FUQAHA, 2015).

Entretanto, (MOUHA, 2021) declara que:

> A definição de Internet das Coisas (IoT) não está definitivamente limitada e não está atualmente definida, o que significa que não existe uma definição geral aprovada pela maioria ou pela comunidade global de utilizadores e, portanto, a Internet das Coisas está a amadurecer e continua a ser a conceito mais novo e popular no mundo da tecnologia da informação. (MOUHA, 2021, p.3)

Entre as aplicações mais prevalentes da Internet das Coisas (IoT), destacam-se as casas inteligentes, cujo objetivo é oferecer conveniência e eficiência na gestão de recursos, como água e eletricidade. Além disso, na agricultura, as tecnologias inteligentes desempenham um papel significativo ao possibilitar a avaliação da saúde do solo com base em parâmetros como umidade e nutrientes. Isso inclui o monitoramento do gado, identificando animais doentes ou em situações de risco, o que viabiliza intervenções precoces e eficazes.

A integração de sistemas baseados em IoT em todos os aspectos da vida humana, juntamente com as diversas tecnologias envolvidas na transferência de dados entre dispositivos incorporados, tornou-a complexa, resultando em vários problemas e desafios tais como: segurança e privacidade. Embora esses termos não tenham surgido e nem sejam exclusivos dos produtos e serviços IoT, os provedores dessas tecnologias precisam garantir que os dispositivos inteligentes e os serviços de dados relacionados estão protegidos contra quaisquer tipos de vulnerabilidades (MOUHA, 2021).

3 METODOLOGIA

Nessa etapa, serão detalhados os passos seguidos durante o processo de desenvolvimento.

Com o objetivo principal de alcançar uma aplicação multiplataforma, capaz de automatizar rotinas utilizadas no processo desempenhado pela Rede de Frio, o início do processo envolveu a execução da engenharia de requisitos. Com isso, o marco inicial partiu de conversas entre as partes interessadas (cliente e desenvolvedores), com o propósito de entender o escopo e necessidades do projeto. Na Tabela 1 estão destacados os requisitos funcionais e não funcionais do sistema desenvolvido.

Tabela 1 – Requisitos da aplicação VacMonitor

Requisitos Funcionais	**Requisitos Não Funcionais**
Login de usuário	Sistema de fácil manutenção
Redefinição de senha	Responsividade da aplicação
Cadastro de usuários	Fácil usabilidade e compreensão da aplicação
Cadastro de refrigeradores	Performance
Cadastro de instituições	Portabilidade da plicação entre as plataformas móvel e web
Cadastro de distritos	-
Exibição da temperatura em tempo real na aplicação	-
Gerar o alerta automático	-
Integração de funcionalidades da plataforma IoT escolhida	-
Exibição da temperatura em tempo real na aplicação	-
Página com exibição de média de temperaturas de um dia do ano	-
Página com exibição de temperaturas mínimas e máximas de um dia do ano	-

Fonte: Elaborado pelos autores

Após alguns diálogos, acordou-se que a iteração de desenvolvimento se basearia no desenvolvimento de uma aplicação com a principal responsabilidade de usar os dados de temperatura, coletados pelo produto em hardware gerado na iteração anterior do projeto, e possibilitar aos usuários o monitoramento em tempo real desses dados. A partir da discussão da possibilidade de implementação dessa funcionalidade, foi considerado também a necessidade de adição de outras funções que foram entendidas como coerentes com o escopo da aplicação, quais sejam:

- Um sistema de gerenciamento das entidades do sistema baseado em formas de hierarquia. Considerando que a expectativa de distribuição dessa aplicação residia principalmente em postos de saúde da rede pública de saúde, o acesso de informações do monitoramento seria compartilhado por diversos usuários, com níveis de interesse distintos, e assim, acessos diferentes.

- A possibilidade de geração de notificações de alerta, em casos onde a temperatura monitorada se aproxima de limites – considerados "críticos" para a integridade do produto armazenado – pré definidos na hora do cadastro das entidades que sustentam tal regra.

Por fim, foi entendido que esse seria um conjunto de funcionalidades suficiente para a concepção de um MVP (*Minimum Viable Product*) nessa iteração. Então definiu-se as partes interessadas envolvidas nessa etapa do processo como, professores orientadores exercendo papel das personas de Cliente – a figura que fornece diretrizes, requisitos e expectativas fundamentais para o projeto – e *Product Owner* (Dono do Produto) – figura que atua garantindo que o produto evolua de acordo com as demandas do mercado e as mudanças nas necessidades dos usuários.

Os alunos orientados desempenharam os papéis de Gerente de Processos – garantir que os procedimentos sejam eficientes, consistentes e estejam alinhados com os objetivos do produto – e Pessoas Desenvolvedoras – responsáveis diretos pela execução prática do processo de desenvolvimento.

Com o escopo definido, a equipe começou o processo de planejamento. Para isso, utilizou uma abordagem híbrida, combinando elementos de metodologias ágeis e tradicionais. No início, o escopo foi dividido em módulos, seguindo o princípio de modularização das metodologias tradicionais. Essa divisão permitiu que o desenvolvimento fosse dividido em etapas menores, o que facilitou a gestão e o controle do projeto.

Em seguida, cada módulo foi analisado para identificar as atividades necessárias para sua implementação. Essas atividades foram organizadas em um backlog, uma lista de tarefas que deve ser priorizada e executada de acordo com as necessidades do projeto.O backlog foi utilizado para definir o planejamento das atividades. As atividades foram distribuídas em sprints, períodos de tempo fixos, seguindo o conceito do Scrum, uma metodologia ágil. Ao final do processo de planejamento, a equipe tinha um plano completo da execução do escopo da aplicação. Esse plano era híbrido, pois combinava elementos de metodologias ágeis e tradicionais, como a modularização e o backlog.

A tabela a seguir apresenta as prints seguidas no desenvolvimento, organizadas em ordem cronológica. Cada sprint é descrita de forma breve, indicando suas principais atividades e resultados esperados:

Tabela 2 – Sprints do projeto

Sprint	**Descrição**
1	Modelagem do sistema e prototipação
2	Configurações dos projetos da aplicação
3	Autenticação do usuário e tela de listagem dos refrigeradores
4	Área de cadastro de refrigeradores
5	Área de cadastro de usuário
6	Módulo de monitoramento de um refrigerador
7	Módulo de processamento dos dados persistidos

3.1 Modelagem do sistema e prototipação

Com o escopo do projeto, e o planejamento do processo definidos, os primeiros passos para a consolidação da aplicação consistiram nas concepções da modelagem do sistema e protótipos da interface.

A modelagem do sistema foi um passo executado de acordo com o esperado, a definição do escopo da iteração, e também funcionalidades futuras, possibilitaram a equipe idealizar o sistema e suas entidades, de forma que suas responsabilidades ficassem bem divididas e distintas entre si.

Faz-se necessário a ressalva na problemática enfrentada no entendimento da necessidade de diferentes níveis de acesso ao sistema a qual é contornada com a implementação de níveis de hierarquias entre diferentes tipos de *persona*[1] de usuário, baseadas na concepção de uma organização genérica na hierarquia de entidades responsáveis pela organização da vertente pública de saúde municipal. A divisão de personas definiu-se então como:

- **Coordenador**: Entidade responsável pela organização a nível de município. Analogamente, tem acesso total a todas as partes da aplicação, e dados correspondentes a de seu município;
- **Gerente**: Ou Gerente Distrital, entidade responsável pela organização apenas de determinada porção do município, denominada "Distrito". Assim tem acesso apenas a informações e funcionalidades correspondentes às instituições de saúde pertencentes ao seu distrito;
- **Agente**: Este tem controle e acesso de informações referentes apenas ao posto de saúde que for relacionado. É a entidade com nível hierárquico mais baixo.

Após a modelagem, a sprint seguiu com a concepção de protótipos da interface da aplicação. Esta etapa é entendida por ser a mais difícil de ser executada em todo o processo de desenvolvimento. Nenhum dos integrantes da equipe tinha nenhuma experiência com

[1] Representação fictícia do seu cliente ideal.

design de interfaces, então os métodos e escolhas seguidas foram baseadas totalmente em experiências e percepções pessoais anteriores da equipe.

A ferramenta utilizada para execução desta etapa foi o Figma[2]. O resultado consistiu de um protótipo de baixa fidelidade, isto é, que incluía apenas estrutura e estilo básico de componentes e telas, que possibilitou o desenvolvimento do produto adotar um mínimo padrão e identidade.

3.2 Configurações dos projetos da aplicação

Essa etapa do desenvolvimento é marcada pela criação e configuração dos projetos que sustentarão a aplicação. O primeiro passo foi a escolha das stacks de desenvolvimento. A equipe optou por utilizar o Flutter para a aplicação móvel e o Django para a API REST.

Essas escolhas foram baseadas em dois fatores principais:

- **Experiência prévia da equipe:**: A equipe já tinha experiência com ambas as tecnologias, o que facilitou o processo de desenvolvimento e reduziu o risco de erros;
- **Aplicabilidade das tecnologias:**: O ambiente de desenvolvimento proporcionado por ambas tecnologias permite que o começo do processo seja feito de forma fácil e ágil;

Após a definição das stacks de desenvolvimento, o restante da sprint pôde ser focada na escolha da ferramenta de clouding. Como essa definição era determinante na implementação de alguns funcionamentos, essa etapa foi marcada principalmente pela exploração e testes de opções das ferramentas de clouding.

A escolha foi baseada principalmente nos quesitos integração, comunicação síncrona em tempo real e custo de implementação. As três ferramentas que apresentaram maior potencial de uso foram ThingSpeak, *AWS* (Amazon Web Services) e Firebase.

Durante o período de descoberta do serviço em nuvem que permitiria integrar-se a aplicação VacMonitor e exibir informações das temperaturas medidas pelo microncontrolador, a usabilidade e facilidade de integração, fez o Firebase se destacar quanto às outras opções. A escolha se concretizou com a análise de um cenário de distribuição real. Alguns passos tornavam o cadastro de dispositivos nas ferramentas da ThingSpeak e AWS um processo burocrático, aumentando o nível de complexidade técnica demandada para o suporte da aplicação.

[2] Figma é um editor gráfico de vetor e prototipagem de projetos de design baseado principalmente no navegador web, com ferramentas offline adicionais para aplicações desktop para GNU/Linux, macOS e Windows.

Com o *clouding* definido, a configuração dos projetos foram executados sem dificuldades devido a experiência prévia com as tecnologias escolhidas. Esse passo pode ser distinguido em duas execuções principais, configuração do *frontend* – que consistiu na implementação de variáveis globais de cores e fontes que seriam reutilizadas durante todo o processo de desenvolvimento, como também a definição da arquitetura de diretórios que o projeto seguiria – e configuração do *backend* – onde foram descritas em código todas as entidades do banco de dados, e suas relações, e definição e instalação de dependências pertinentes para o projeto.

3.3 Autenticação do usuário e tela de listagem dos refrigeradores.

A *sprint* seguinte teve como principal objetivo a implementação do sistema de autenticação para o usuário na aplicação. As tarefas que representavam os passos necessários para a execução dessa parte consistiam no, desenvolvimento da interface responsável por captar os dados do usuário, e integração da dependência de autenticação presente no backend com a entidade representante de Usuário, além da disponibilização do método de processamento das credenciais inseridas pelo usuário.

As implementações do escopo dessa sprint por parte do *backend* foram alcançadas a partir do uso da biblioteca de autenticação *Djoser*, distribuída para Django e amplamente utilizada em diversas aplicações. A biblioteca já disponibiliza ações básicas de autenticação como *login* e *logout*. A autenticação a partir dessas ações consiste na verificação da validade dos dados de credenciamento do usuário que foram fornecidos, onde, se válidos, uma cadeia de caracteres – denominada *token* – é criada para a identificação do usuário.

Como o sistema de autenticação da aplicação não envolvia nenhum passo de complexidade diferente dos já implementados nas ações disponibilizadas pela biblioteca, sendo estas então suficientes para o atendimento da regra do sistema sendo desenvolvida, se mostrando necessário apenas da indicação do model (modelo) responsável por representar o usuário, e o cadastro das URLs (*Uniform Resource Locator*) responsáveis pelo acesso dos métodos na interface da aplicação.

O escopo que precisava ser atendido pelo frontend seria alcançado com o desenvolvimento das telas de login e splash. A tela de login é a responsável por captar os dados do usuário a partir da inserção destes em campos de entrada, e então o envio desses dados para o processamento do credenciamento através do método já implementado na API Rest da aplicação. Em caso de sucesso no credenciamento, também é responsabilidade dessa tela armazenar o token do usuário no armazenamento interno do dispositivo que a aplicação esteja sendo usada, para que este sirva para a identificação do usuário durante toda a seção de uso na aplicação.

A tela de *splash* tem como responsabilidade, durante uma tentativa de acesso à aplicação por parte do usuário, acessar o armazenamento interno do dispositivo e verificar se há algum token de identificação já armazenado, configurando assim uma sessão de usuário já existente, portanto sem a necessidade do credenciamento. Para o caso da não identificação do usuário, a tela de splash deve direcioná-lo até a tela de login para que o credenciamento aconteça e uma sessão seja iniciada.

A execução das tarefas dessa etapa do desenvolvimento foram realizadas de maneira eficiente e sem maiores problemas, de tal forma que, no fim da execução das implementações, ainda restava uma parcela de tempo para a conclusão da sprint. A equipe seguiu então com o adiantamento de algumas tarefas referentes ao escopo planejado para a próxima sprint, que consistiam na área de listagem de refrigeradores.

Essas implementações foram alcançadas com o desenvolvimento da tela de home, a tela tela da qual a splash deve redirecionar o usuário em caso de sucesso no credenciamento. A tela de home tem como responsabilidade disponibilizar ao usuário o acesso do início de maioria, ou todos, os fluxos de funcionalidades presentes em toda a aplicação, junto com a funcionalidade de também fazer a listagem de todo o conjunto de refrigeradores que o usuário tem acesso, disponibilizando as informações referentes a cada refrigerador, através de cartões de exibição.

3.4 Área de cadastro de refrigeradores

Para essa sprint o conjunto de funcionalidades a serem implementadas deveriam resultar no fluxo de cadastro de um novo refrigerador. Este fluxo deveria ser acessado a partir da tela de home e redirecionar o usuário para uma tela de formulário onde devem ser inseridos os dados referentes ao refrigerador. Durante as fases iniciais das implementações notou-se um potencial gargalo na execução do fluxo por parte do usuário.

A problemática estava contida na necessidade do alinhamento de um valor chave na integração do Firebase com ambos aplicação e o microcontrolador ligado ao refrigerador cadastrado, que atua como um nó da rede dentro do contexto de uma aplicação de Internet das Coisas. O valor em questão diz respeito ao identificador do Documento do Firebase responsável por fazer a transmissão de dados em tempo real, da qual a aplicação irá consumir os dados, e o dispositivo node irá transmitir.

Durante o cadastro de um novo refrigerador, o valor deste identificador na entidade representante de um refrigerador, deve necessariamente ser o mesmo valor que foi compilado no microcontrolador. Qualquer inconsistência nesses valores, causaria uma consulta em um Documento inexistente, impossibilitando o funcionamento esperado.

Como solução deste impasse, foi necessária a criação de uma nova persona usuária

do sistema, um usuário do tipo Técnico. Esse tipo de usuário seria o único com permissão para cadastrar um novo refrigerador, delegando a responsabilidade de garantir a assertividade do valor de definição do identificador do Documento referente ao dispositivo físico sendo monitorado e sua entidade representante no sistema. Por conseguinte, tornando a solução da problemática, a restrição da permissão para criar um novo refrigerador a apenas um pequeno conjunto de usuários, a fim de diminuir a chance de inconsistência entre o cadastro e o consumo dos dados do Documento.

A partir da implementação da nova persona à API Rest, foi possível avançar com o desenvolvimento do restante das funcionalidades a serem executadas pelo backend, para o cadastro de um novo refrigerador. A entidade representante de um refrigerador é, necessariamente, relacionada à uma entidade representante de posto de saúde, a fim de sustentar a regra de que um refrigerador deve pertencer a um posto. Posto este, que consequentemente deve pertencer a um distrito de um município. Todas estas unidades citadas, estão representadas por entidades no banco de dados, e se relacionam entre si, seguindo a seguinte estrutura: um refrigerador deve pertencer a um posto, este que deve pertencer a um distrito, este que por sua vez, deve pertencer a um município.

Considerando as entidades e seus relacionamentos entre si, a tela de criação do novo refrigerador possui uma estrutura de preenchimento condicional, onde o técnico executando o cadastro pode escolher entre selecionar uma entidade já existente, para a relacioná-la ao refrigerador, ou criar uma nova. Isto é, durante a criação, pode-se selecionar um distrito já existente, ou adicionar um novo, e por sua vez, escolher um posto existente, que pertença ao distrito escolhido, ou adicionar um novo.

Caso seja adicionado um distrito novo, necessariamente deve-se adicionar um posto, já que a instância nova não tem nenhuma outra relacionada. O método de criação API Rest foi projetado para receber as informações de identificação geral do refrigerador como nome e identificador referente ao documento do Firebase.

A entrada também consiste de dois valores que representam limites superior e inferior, entendidos como limites de forma que a integridade do material deve ser armazenado. Também recebe a entrada condicional de dados, criando entidades novas, ou referenciando as mesmas já existentes que foram escolhidas pelo usuário, então às relacionando entre si, e por fim às relacionando com a nova instância de refrigerador.

3.5 Área de cadastro de usuário

A *sprint* que se seguiu, teve como principal objetivo o desenvolvimento da área de cadastro de um usuário. Essa funcionalidade tinha particularidades em seus requisitos, das quais consistiam: o usuário não seria o responsável por realizar seu próprio cadastro, para algum outro usuário já existente seria delegado essa responsabilidade – essa opção

se mostrou coerente junto com necessidade de um rigoroso controle sobre o acesso à informação das instituições de saúde, e também da ideia de distribuição, em um contexto de suporte apenas na área de um município – e o usuário cadastrado teria de, necessariamente, pertencer a um mesmo grau, ou inferior, de hierarquia, referente ao usuário efetuando o cadastro.

Como medida de segurança, foi adotado um campo de "criado por" no modelo representante de usuário, fazendo relação com outro usuário, guardando a informação do responsável por seu cadastro.

As regras de cadastro de um usuário, se assemelham em parte com o cadastro de um novo refrigerador. O fluxo de atribuição a entidades de posto ou distrito, mantém a mesma ideia de definição de relação a entidades existentes, ou, caso necessário, criação de novas instâncias, mas com a ressalva que a entidade a ser atribuída dependerá do tipo de usuário sendo criado.

- Um usuário do tipo coordenador, tem responsabilidade a nível de município, por isso, automaticamente o município a ser atribuído, será o mesmo do usuário efetuando o cadastro – essa regra tem respaldo na ideia de que a área de atuação de qualquer usuário, independente do seu nível de hierarquia, ter limite apenas o município que lhe foi relacionado. Assim, em via de regra, o usuário de tipo coordenador terá por acesso a informações de todos os distritos pertencentes àquele município, e consequentemente, todos os postos deste, e por sua vez, todos os refrigeradores;

- Um usuário do tipo gerente distrital, tem nível de acesso referente a um distrito, então em sua criação, deve lhe ser atribuída uma entidade representativa do distrito que será responsável, lhe concedendo assim acesso a informações de todos os postos relacionados à mesma;

- O usuário do tipo agente deve ser relacionado a uma entidade representativa de posto, da qual ele terá acesso apenas às informações de refrigeradores que o são relacionados.

O funcionamento da interface para essa área segue a mesma abordagem de exibição condicional dos componentes, a depender do tipo de usuário sendo cadastrado. Campos de informações gerais, como nome, email e telefone são sempre preenchidos, porém a seleção da entidade e seu tipo – dentre distrito ou posto – será inerente ao tipo escolhido no momento do cadastro.

3.6 Módulo de monitoramento de um refrigerador

Para a sexta sprint o conjunto de tarefas a serem executadas são responsáveis por exibir o detalhe das informações a respeito de determinado refrigerador, além de estabelecer a conexão com o Firebase, que possibilita a leitura dos dados de temperatura em tempo real, junto com o envio de alertas de notificação, caso o valor da leitura da temperatura ultrapasse os valores pré definidos como limite superior e inferior de temperatura, no momento do cadastro do refrigerador.

As implementações com a Rest API para essa sprint são divididas em duas funcionalidades distintas: retorno das informações referentes ao refrigerador, e ação de atribuição do dispositivo de acesso à aplicação do usuário, com o serviço Firebase Cloud Messaging, responsável pelo envio de notificações do tipo push para aplicações. O método de atribuição de dispositivos é executado logo após o login, recolhendo um valor chave, identificador do dispositivo físico executando a aplicação, disponibilizando à API pelo sistema operacional do dispositivo.

Durante a execução do método, o identificador do dispositivo é armazenado em uma instância da classe representante do Dispositivo, junto com a entidade referente ao usuário da sessão. A partir dessa instância, a API é capaz de enviar um alerta de notificação a todos os usuários que estão relacionados com o refrigerador que teve os limites de temperatura pré-definidos infringidos.

Os passos da implementação do cadastro do dispositivo e envio de notificações, foram entendidas como os principais desafios desta sprint, dada a falta de experiência da equipe com o uso da dependência usada pelo envio das notificações. Com isso, o prazo planejado da sprint acabou não sendo o suficiente para a completa execução das tarefas, tendo que ser necessário o uso de tempo atribuído à sprint posterior.

A partir da implementação na Rest API, a interface da aplicação é capaz de resgatar todas as informações do refrigerador, incluindo o valor identificador, responsável por estabelecer a conexão em tempo real com o Firebase, e receber a informação referente à leitura da temperatura do refrigerador. Essas informações são exibidas na única tela que pertence ao fluxo, que é acessada a partir do cartão de exibição da tela anterior responsável pela listagem.

3.7 Módulo de processamento dos dados persistidos

A última sprint desta iteração de desenvolvimento tinha como seu objetivo principal alcançar a implementação de funcionalidades relacionadas ao processamento de dados que foram registrados a partir das leituras de temperatura de cada refrigerador. Com o uso do Firebase como ferramenta responsável pela transmissão dos dados de tempera-

tura, a equipe decidiu pela abordagem do uso da vertente de persistência desses dados, diretamente no Firebase, denominada FireStorage.

A partir da implementação do FireStorage, tornou-se possível o resgate das informações das temperaturas que foram persistidas, possibilitando implementações de processamento desses dados. Como nesse ponto, o tempo restante para o desenvolvimento das funcionalidades estava chegando ao seu limite, o resultado alcançado nesse módulo acabou sendo reduzido quando comparado com as funcionalidades planejadas. Contudo, mesmo o escopo sendo reduzido, o resultado alcançado ao final do prazo, se mostraram consistentes e práticos.

Foi implementado, juntamente do módulo de monitoramento, componentes que servem como exibição da saída do processamento dos dados de temperatura. O primeiro deles consiste em um gráfico, que mostra a progressão da variação de temperatura durante um dia, permitindo ainda que o usuário faça a escolha de um dia que queira acompanhar a progressão. Junto com o gráfico, também foram implementados, componentes de exibição que disponibilizam informações sobre o pico superior e inferior de temperatura para determinado dia, e também a medida da média aritmética de todos os registros de temperatura para aquele mesmo dia.

Sobre como estes dados estão estruturados no Firebase (Figura 3), baseado no **modelo de dados** do **Cloud Firestore**[3], existe uma coleção chamada "containers", onde nela estarão presentes documentos, em que esses documentos são relativos ao refrigerador e a data que o ocorreu a medição das temperaturas. Com relação a como as temperaturas medidas estão dispostas, dentro de cada documento há um campo chamado "temperaturas", onde esta, é uma lista de informações do qual podem ser extraídas:

1. **date**: Data que ocorreu a medição;
2. **temperature**: Valor de temperatura lido pelo sensor;
3. **timestamp**: Representa um instante único, onde é possível adquirir data, hora, minutos e segundos.

[3] https://firebase.google.com/docs/firestore/data-model?hl=pt-br

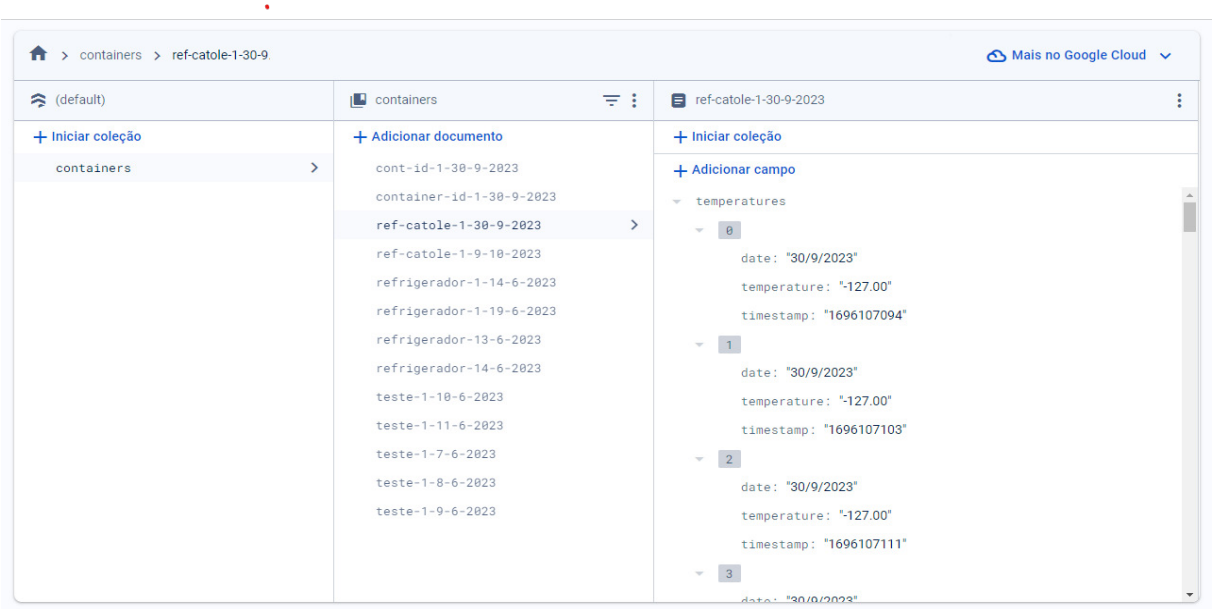

Figura 3 – Firestore Cloud: Dados de temperatura persistidos

Fonte: Elaborado pelos autores

3.8 Disponibilização do serviço na nuvem

O serviço escolhido para a disponibilização da API do sistema na Internet foi a Divio. Com isso, como pode ser visto na Figura 1, dois *environments* (ambientes) na plataforma foram criados. Essa divisão foi escolhida pelos desenvolvedores para separar os ambientes que seriam de produção e de testes.

A forma como se cria e atualiza cada um desses ambientes é através da integração do GitHub com a plataforma Divio. Os *branches* (ramos) **dev** e **staging** existem no repositório GitHub da *API* da aplicação, elas possuem o código que permite a integração com o frontend. A plataforma Divio por sua vez, da forma como foi configurada pelos desenvolvedores, faz com que cada ambiente criado esteja relacionado a uma desses branches e, por conseguinte, o deploy é feito manualmente através de um clique. Dessa forma, consegue-se disponibilizar o serviço na internet.

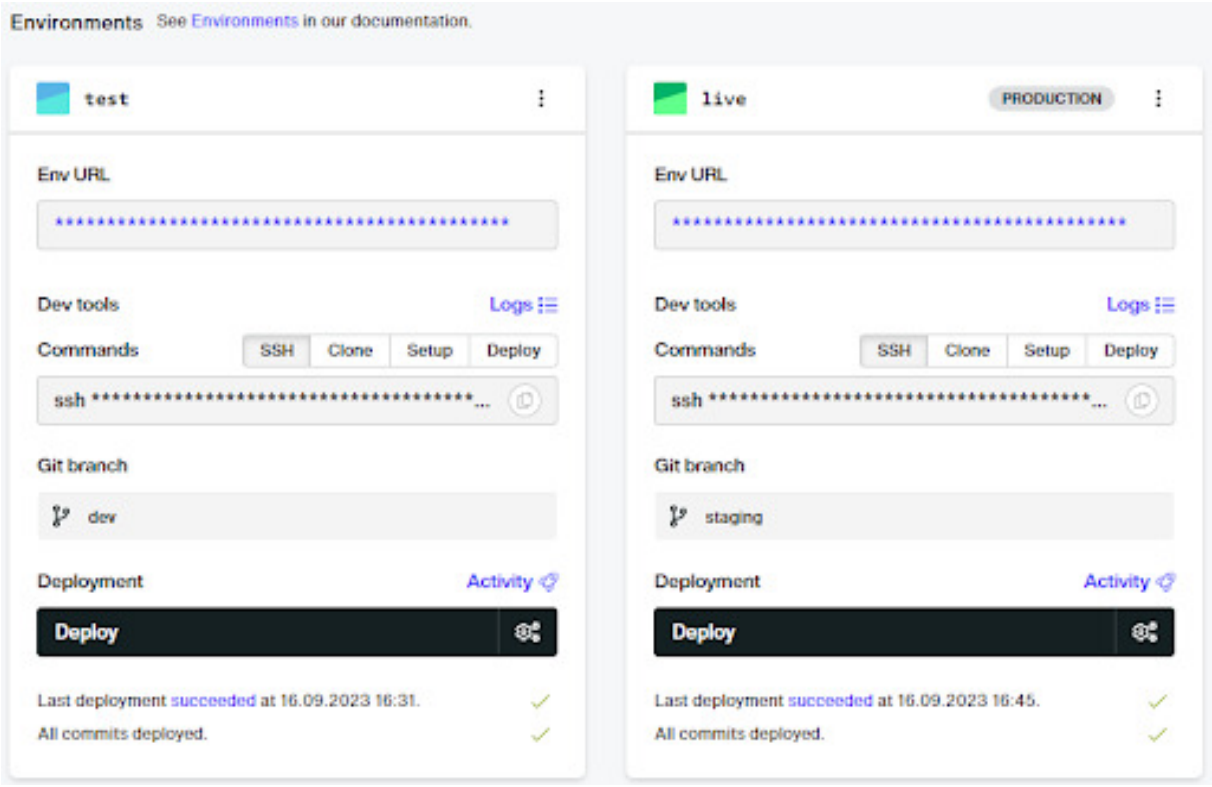

Figura 4 – Ambientes criados na Divio

Fonte: Elaborado pelo autor

3.9 Diagrama arquitetural

Pode-se dividir o sistema, em sua totalidade, em quatro módulos, estando eles numerados na Figura 2.

Aquele enumerado com 1, é o dispositivo que gerenciará o sensor que realizará as medições de temperatura e se comunicará com o Firebase, para o envio dos valores dessas temperaturas, e também está integrando com a API Rest desenvolvida, sendo a única comunicação entre eles o envio de alerta quando os limites ideais de temperatura são extrapolados.

O módulo 2 é o serviço em nuvem Firebase responsável por salvar e enviar as temperaturas em tempo real para a interface do usuário. Já o serviço em nuvem 3 é onde está presente todo o banco de dados que armazena todas as informações que não são sobre as temperaturas medidas e também é a responsável por realizar toda a regra de negócio.

Por fim, no módulo 4, está presente a interface do usuário, seja web ou mobile, onde está integrada com os 2 (dois) serviços em nuvem, que por sua vez, resgata as informações em tempo real do Firebase (relativos às temperaturas) e, recupera informações dos refrigeradores, usuários, instituições, distritos e faz envio de formulários para a API Rest.

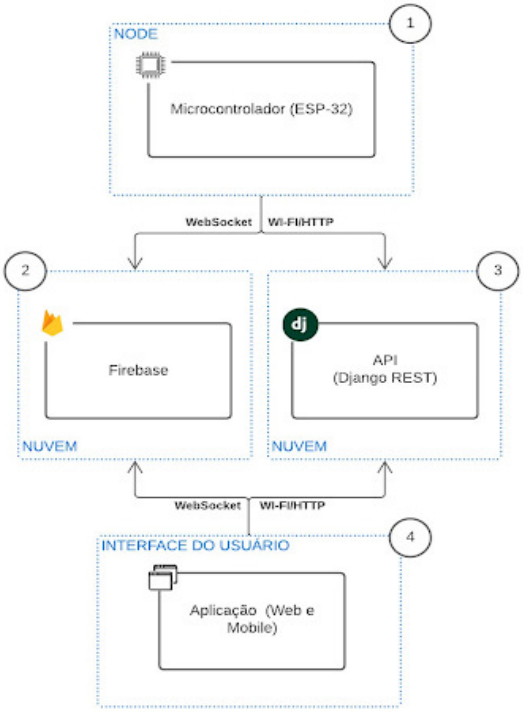

Figura 5 – Diagrama arquitetural

Fonte: Elaborado pelo autor

3.10 Distribuição e disponibilização da aplicação

O serviço de hospedagem da aplicação na web escolhido foi o *Firebase Hosting*, a facilidade de *deploy*, além do plano gratuito para testes, tornou uma das opções mais viáveis para disponibilizar a Web Application. Já a aplicação mobile foi disponibilizada através do *App Distribution*. Os serviços são *plugins* do Firebase e ambos permitem visualizar todas as versões lançadas, além de comentários (por parte do desenvolvedor) e *feedbacks* (por parte dos testadores). O pré-requisito para o uso dessas ferramentas é ter o projeto criado no Firebase e a configuração dele com o código da aplicação.

4 Resultados

Nesta seção serão apresentados os resultados obtidos através das iterações de desenvolvimento, dispostas em forma de sprints, discorridas no último capítulo.

4.1 Modelagem do sistema

As duas primeiras iterações do processo, se baseiam principalmente em concepções de ideias e conceitos, e configuração das tecnologias usadas no projeto, portanto os resultados obtidos a partir destas, não possuem uma representação concreta. A modelagem final do sistema foi compreendida como o seguinte diagrama UML[1].

O diagrama (Figura 6) se mostra imprescindível para o entendimento da estrutura e relacionamento das entidades presentes no sistema. A partir dele, a equipe foi capaz de determinar mais claramente, a estrutura de hierarquias, tratando os requisitos que as procediam.

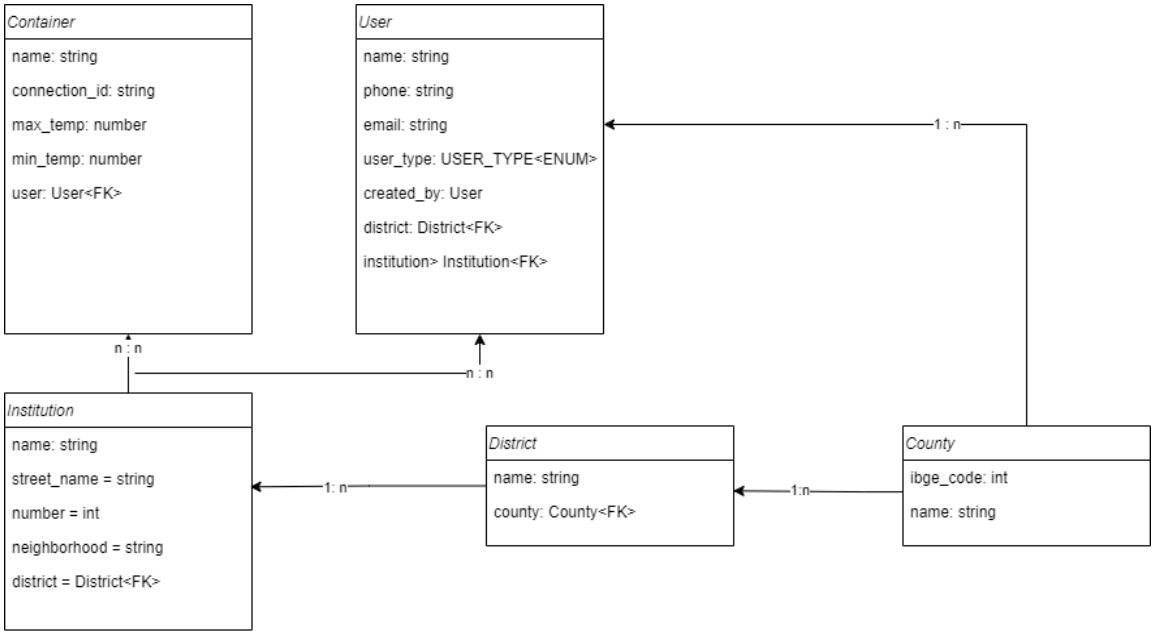

Figura 6 – Modelagem do banco de dado

Fonte: Elaborado pelo autor

4.2 Autenticação

O desenvolvimento da autenticação do sistema seguiu em um ritmo rápido, com a execução das tarefas seguindo exatamente conforme o planejado. A utilização de uma

[1] Um diagrama UML é uma forma de visualizar sistemas e softwares usando a Linguagem de Modelagem Unificada (do inglês Unified Modeling Language - UML).

biblioteca voltada exclusivamente para o passo da geração de sessão do usuário no *backend* agilizou o processo de desenvolvimento, além de trazer um nível a mais de consistência e confiabilidade no funcionamento, do que um sistema de autenticação implementado manualmente geralmente proporciona.

Em questão da aplicação, a forma de obtenção das credenciais do usuário e a forma de envio para a API, tudo isso é feito na tela de login (Figura 7 e Figura 8). Inicialmente, na aplicação, é checado se anteriormente o usuário já fez *login*, caso contrário, o usuário é redirecionado a tela de autenticação. A partir das informações dadas pelo mesmo e resgatadas pelos *inputs* da aplicação, caso o usuário não esteja cadastrado no sistema, ou tenha colocado errado alguma de suas credenciais, um aviso é mostrado em tela, dando contexto ao cliente que as informações digitadas por ele possuem alguma irregularidade.

Figura 7 – *Mobile App*: Tela de *login* do VacMonitor

Fonte: Elaborado pelo autor

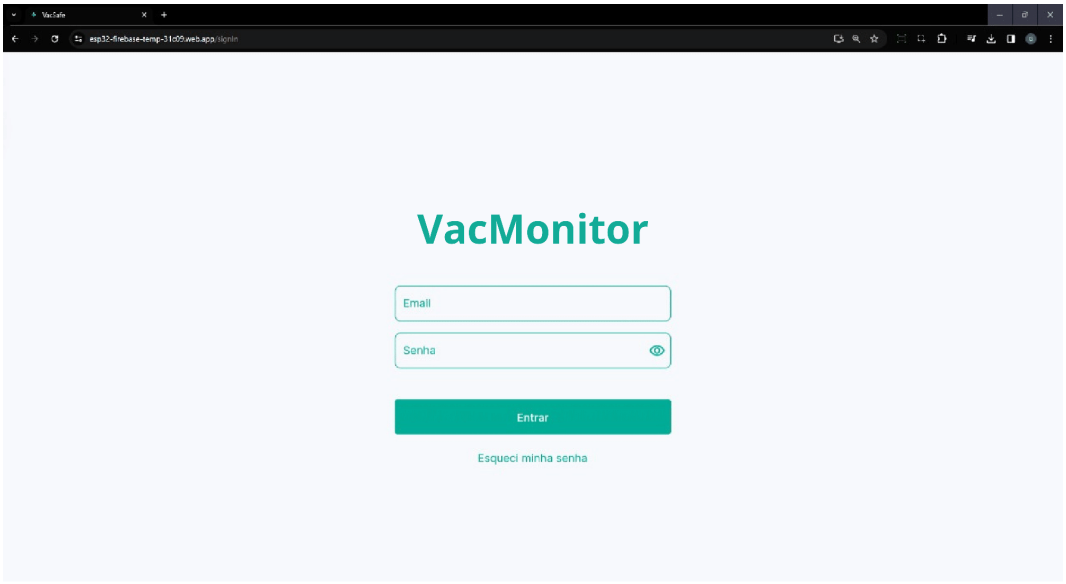

Figura 8 – Web *App*: Tela de *login* do VacMonitor

Fonte: Elaborado pelo autor

Como resultado, é garantido a autenticação do usuário, desde que suas credenciais estejam corretas, além de guardar a seção do usuário na aplicação, de forma que não seja necessário fazer *login* constantemente.

4.3 Listagem de refrigeradores cadastrados

A seguinte funcionalidade foi planejada para *sprints* futuras, uma vez que, no fim da execução das tarefas citadas no tópico anterior, apenas parte do tempo da *sprint* foi utilizado, resultando na possibilidade de implementação da funcionalidade da listagem dos refrigeradores atribuídos a determinada instituição a qual o usuário faz parte. Funcionalidade esta que compila tarefas de execução simples.

Como resultado, a tela de listagem consiste em uma rolagem, – vertical para a aplicação *mobile* (Figura 10), horizontal para web (Figura 9) – contendo componentes que exibem certas informações a respeito do refrigerador, tais como: nome, endereço e gerente(s) de distrito. Além disso, partindo do que foi apresentado na modelagem de dados, a listagem dos refrigeradores foi implementada de forma que os itens dispostos na tela dependem do tipo do usuário autenticado. Ou seja, na Figura 10 e Figura 9, o usuário atual é de nível Técnico, e assim como o usuário de nível Coordenador, ele pode visualizar todos os refrigeradores cadastrados no município de Campina Grande.

Para facilitar o entendimento, pode-se observar a Tabela 3 abaixo:

Tabela 3 – Listagem de Refrigeradores

Tipo de usuário logado	**Como ocorre a visualização (listagem) dos refrigeradores?**
Técnico	Conseguirá ver todos os refrigeradores cadastrados no município que este usuário pertence
Coordenador	Conseguirá ver todos os refrigeradores cadastrados no município que este usuário pertence
Gerente de Distrito	Conseguirá ver todos os refrigeradores cadastrados no distrito que este usuário gerencia
Agente	Conseguirá ver todos os refrigeradores cadastrados no posto de saúde que este usuário está alocado

Fonte: Elaborado pelos autores

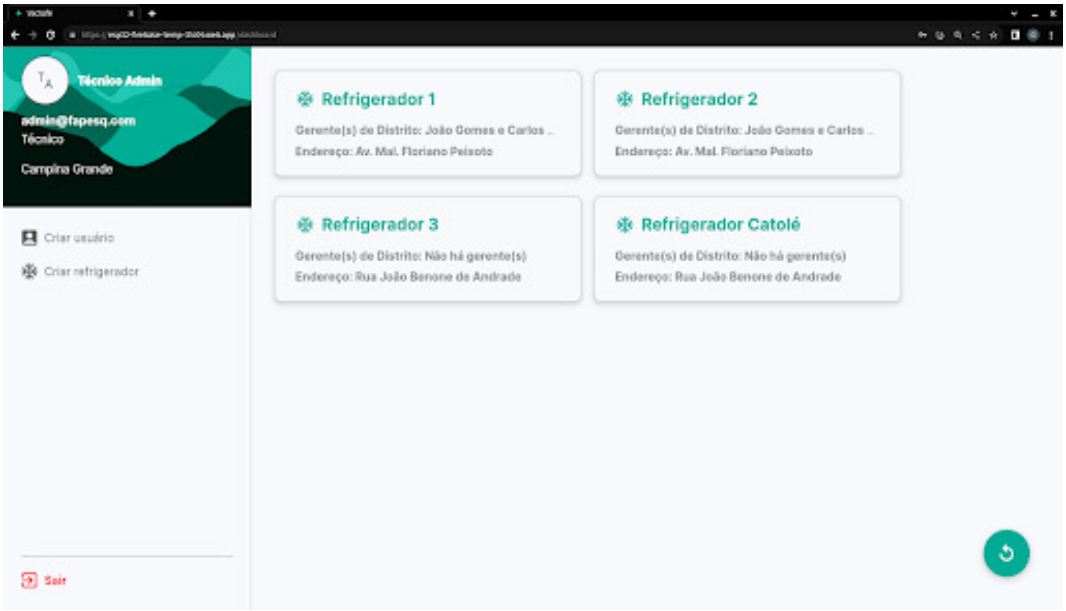

Figura 9 – Web *App*: Tela de listagem de refrigeradores (Técnico)

Fonte: Elaborado pelo autor

Figura 10 – *Mobile App*: Tela de listagem de refrigeradores (Técnico)

Fonte: Elaborado pelo autor

Na Figura 11 é apresentada uma simulação da listagem dos refrigeradores de um usuário do tipo Gerente de Distrito, o "Refrigerador 1" e "Refrigerador 2" listados fazem parte do distrito em que o atual usuário é gerente. Por fim, na Figura 12 é apresentada a simulação da listagem dos refrigeradores de um usuário do tipo Agente, como o "Refrigerador 3" e "Refrigerador Catolé" pertencem à mesma instituição do usuário atualmente autenticado, ele consegue visualizar estes itens.

Com isso, é garantido a responsabilidade de cada tipo de usuário do sistema para com os refrigeradores cadastrados no sistema. Portanto, consegue-se dar ou retirar acessos ao usuário a depender de sua hierarquia dentro do sistema.

Figura 11 – *Mobile App*: Tela de listagem de refrigeradores (Gerente de distrito)

Fonte: Elaborado pelo autor

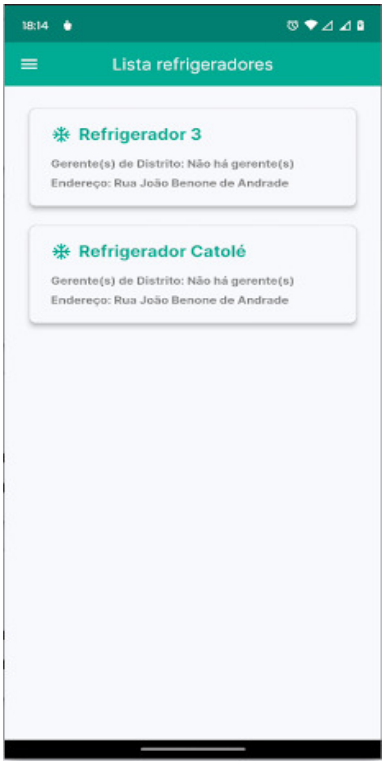

Figura 12 – Web *App*: Tela de listagem de refrigeradores (Gerente de distrito)

Fonte: Elaborado pelo autor

4.4 Área de cadastro de novos usuários

Esta funcionalidade é uma das etapas cruciais do fluxo da aplicação, uma vez que o usuário cadastrado terá relacionamentos com outras entidades do sistema e assim resultar no funcionamento geral da aplicação. Como se pode observar nas Figuras 13 e 14, quando o cliente ingressar no fluxo de cadastro de usuário, haverão *inputs* de texto relativos ao nome, email e telefone daquele que será registrado, além disso, o cliente deverá escolher qual será o tipo do usuário dentro do sistema.

As opções disponíveis de tipo de usuário estão diretamente relacionadas a hierarquia de usuário a qual o cliente logado está relacionado. Como já citado anteriormente, o sistema contempla três categorias, sendo elas: coordenador, gerente de distrito e agente. A Tabela 4 mostra a permissão que cada tipo de usuário possui com relação ao cadastro de outras entidades do mesmo tipo.

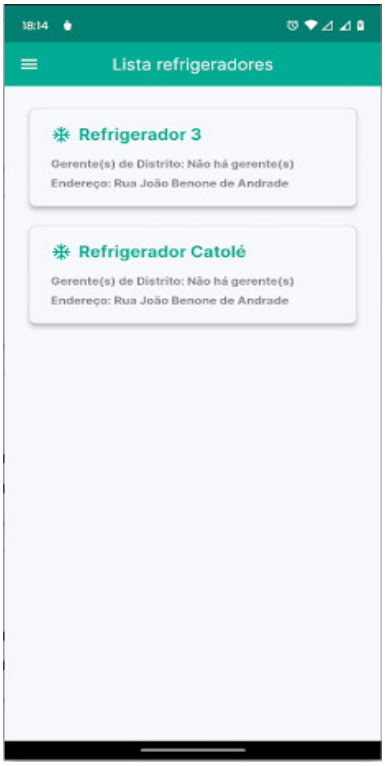

Figura 13 – *Mobile App*: Tela de cadastro de usuário

Fonte: Elaborado pelo autor

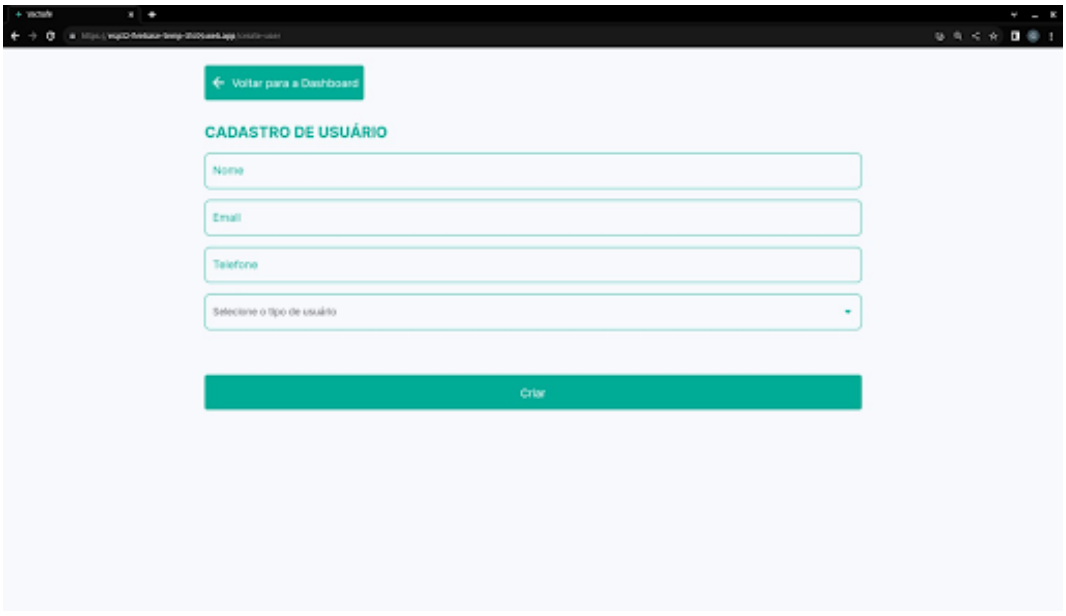

Figura 14 – Web *App*: Tela de cadastro de usuário

Fonte: Elaborado pelo autor

Por conseguinte, quando a escolha de qual o tipo de usuário que será cadastrado for realizado, uma nova entrada de dados para o cliente preencher poderá ou não aparecer. Seguindo o cenário que o usuário logado é um coordenador (Figura 15) e irá seguir o fluxo de registrar uma nova entidade, ao preencher as informações e selecionar o tipo de usuário

Tabela 4 – Cadastro de usuário

Tipo de usuário logado	**Tipo de usuário que ele poderá registrar**
Técnico	Coordenador, Gerente de Distrito, Agente
Coordenador	Coordenador, Gerente de Distrito, Agente
Gerente de Distrito	Gerente de Distrito, Agente
Agente	Não faz cadastro de usuário

Fonte: Elaborado pelos autores

sendo "Coordenador", nenhuma nova entrada aparecerá e poderá concluir a criação desse novo usuário.

Agora, se o tipo de entidade for escolhida sendo "Gerente de distrito", um novo *input* do tipo caixa de seleção aparecerá (Figura 16) e trará todos os distritos cadastrados no município do qual o usuário logado pertence, após a seleção desse distrito, o usuário poderá ser registrado.

Por fim, se o tipo escolhido for "Agente", uma nova entrada surge (Figura 17) a qual se refere a instituições, isso acontece porque o usuário "Agente" está diretamente relacionado a uma instituição (também citado alguns momentos como posto de saúde), entretanto, há um porém, caso o usuário a ser registrado seja do tipo "Agente" e o cliente logado seja do tipo "Coordenador", as instituições listadas nessa caixa de seleção devem estar sempre relacionadas ao município atribuído a este coordenador, entretanto, se o cliente logado for do tipo gerente, as instituições devem estar relacionadas com o distrito o qual o gerente administra.

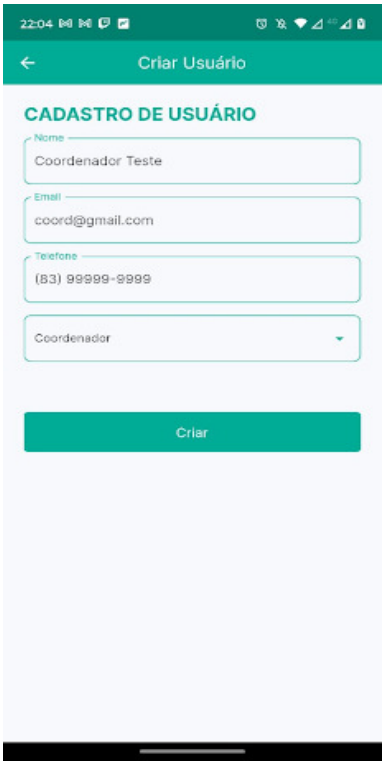

Figura 15 – *Mobile App*: Cadastro de um Coordenador

Fonte: Elaborado pelo autor

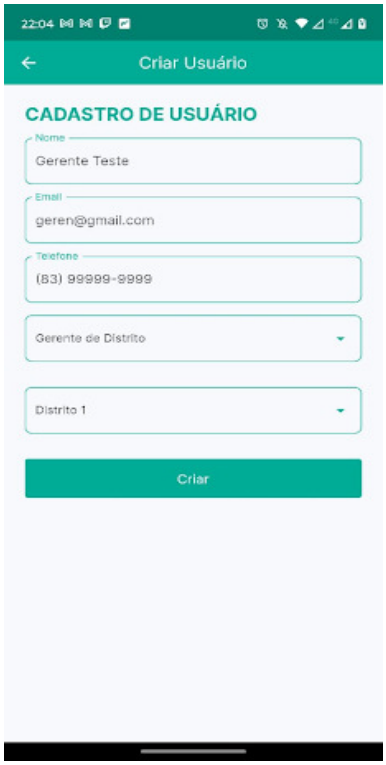

Figura 16 – *Mobile App*: Cadastro de um Gerente de Distrito

Fonte: Elaborado pelo autor

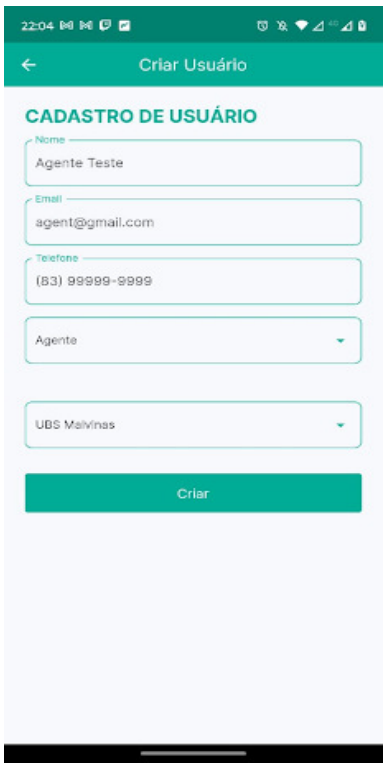

Figura 17 – *Mobile App*: Cadastro de um Agente

Fonte: Elaborado pelo autor

Para sinalizar o sucesso do registro do novo usuário, o cliente é redirecionado para a tela de listagem dos refrigeradores e um aviso na parte inferior na aplicação é exibido, como mostrado na Figura 18. Com a conclusão da criação, o novo usuário deverá seguir o fluxo de *reset* de senha para assim poder fazer *login* na aplicação.

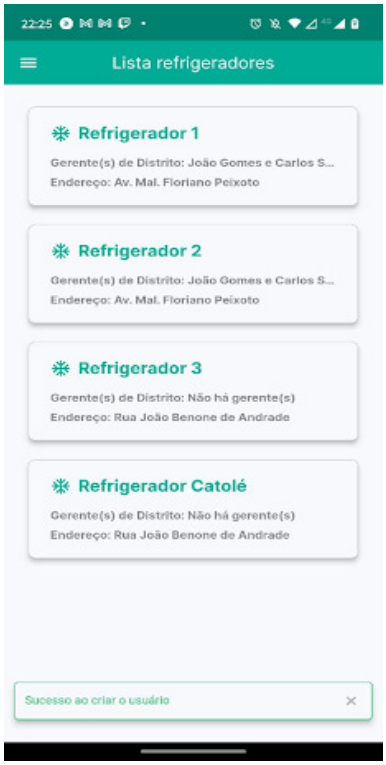

Figura 18 – *Mobile App*: Notificação de sucesso

Fonte: Elaborado pelo autor

Como resultado, pela aplicação, seja *mobile* ou web, consegue-se adicionar novos usuários para o sistema, consequentemente, essas novas entidades terão responsabilidades dentro do *app*, seja monitorar um ou mais refrigeradores e/ou cadastrar ainda mais usuários.

4.5 Área de cadastro de refrigeradores, instituições e distritos

Este tópico, bem como o anterior, é uma funcionalidade central da aplicação apresentada. A criação da entidade **refrigerador** é o ponto inicial para conectar o serviço com o microcontrolador responsável por medir as temperaturas e, além disso, atribuir a responsabilidade para a instituição que armazena os insumos biológicos (as vacinas) a serem monitorados.

Como apresentado na Figura 19, a tela de criação do refrigerador possui inicialmente inputs com relação às informações do refrigerador:

1. Apelido - Campo para facilitar a identificação do refrigerador no dia a dia;
2. ID da conexão - Identificador único que permitirá a conexão do refrigerador a ser criado com seu respectivo microcontrolador que fará a medição das temperaturas;
3. Temperatura Mínima - Valor em ponto flutuante da temperatura mínima que pode estar dentro do refrigerador;
4. Temperatura Máxima - Valor em ponto flutuante da temperatura máxima que pode estar dentro do refrigerador.

Figura 19 – *Mobile App*: Criação de um refrigerador

Fonte: Elaborado pelo autor

Além dos *inputs* citados, há mais campos relativos às informações do local onde o refrigerador vai estar alocado, esta é outra importante etapa dessa funcionalidade, isso porque, uma vez que a instituição já exista no sistema, não será necessário refazer o preenchimento dos campos, o usuário pode clicar no botão de alternância com o rótulo "Selecionar posto existente" e o formulário irá modificar seu comportamento conforme a Figura 20, diante disso, o cliente agora deve escolher esta instituição já inserida na aplicação e seus dados serão automaticamente preenchidos.

Figura 20 – *Mobile App*: Criação de um refrigerador com instituição existente

Fonte: Elaborado pelo autor

Além disso, pode existir um cenário em que, a instituição ainda não exista no banco de dados, mas, o distrito a qual ela está inserida já existe no sistema. Portanto, como se pode observar na Figura 21, o usuário deve digitar os dados da instituição a ser registrada e clicar no botão de alternância com o rótulo "Selecionar distrito existente", em consequência disso, o formulário irá modificar, o cliente pode então selecionar este distrito existente no sistema e concluir o cadastro.

Vale ressaltar que a API da aplicação trata casos de valores de *inputs* duplicados para os seguintes campos: ID da conexão, Apelido e Unidade. A duplicidade desses valores pode gerar confusão em alguns fluxos dentro da aplicação, a partir dessa possibilidade, de forma preventiva, foi implementado no *backend* a capacidade de detectar estes cenários e, com isso, informar para o *app* e consequentemente para o usuário.

Figura 21 – *Mobile App*: Criação de um refrigerador com distrito existente

Fonte: Elaborado pelo autor

A permissão de usuário necessária para realizar essa funcionalidade está apresentada na Tabela 5:

Tabela 5 – Cadastro de refrigerador

Tipo de usuário logado	**Tem permissão para criar um refrigerador?**
Técnico	Sim
Coordenador	Não
Gerente de Distrito	Não
Agente	Não

Fonte: Elaborado pelos autores

Esta função é reservada para um técnico porque depende de informações que normalmente nenhum coordenador, gerente de distrito ou agente terão conhecimento. Uma vez que no cadastro e configuração do microcontrolador é necessário conhecimentos técnicos que serão executados por um usuário de nível técnico dentro da aplicação.

Por fim, se a instituição a qual o refrigerador está alocado e o distrito que a mesma pertence ainda não estiverem inseridas no sistema, o usuário deverá preencher obrigatoriamente todos os campos apresentados na Figura 19. Independente do cenário seguido, após o fim do cadastro e o retorno de sucesso da API, o usuário é redirecionado para a

listagem dos refrigeradores e conseguirá ver na listagem o recente adicionado refrigerador e uma mensagem na parte inferior da tela sinalizando sucesso (Figura 22).

Figura 22 – *Mobile App*: Sucesso na criação do refrigerador

Fonte: Elaborado pelo autor

Como resultado, a possibilidade da criação das entidades "Refrigerador", "Instituição" e "Distrito" permitirá a atribuição de responsabilidade de monitoramento para um posto de saúde e, consequentemente, aqueles usuários pertencentes à instituição em questão irão conseguir monitorar o refrigerador e receber notificações a depender se os valores mínimos e máximos se aproximam de seus limites. Isso também inclui o gerente que administra o distrito onde a instituição está alocada e, também, o coordenador do município onde o distrito está inserido.

4.6 Área de monitoramento e detalhes do refrigerador

Para que o usuário possa visualizar a temperatura em tempo real, foi desenvolvido uma página que, via integração com o Firebase, resgatará o valor de temperatura mais atualizado e, por conseguinte, exibirá para o usuário. Essa tela em questão, - na aplicação mobile - está dividida em três abas, como se pode observar na Figura 23, a primeira aba é responsável por mostrar ao usuário um valor de temperatura e, o dia e horário exatos de quando ocorreu tal medição.

Diferentemente da versão para dispositivos móveis, na web, geralmente a tela é de uma proporção maior, devido a isso, não foi necessário dividir a página em em abas, portanto, analisando a Figura 24, todas as informações relativas ao refrigerador são dispostas de uma vez em uma única página.

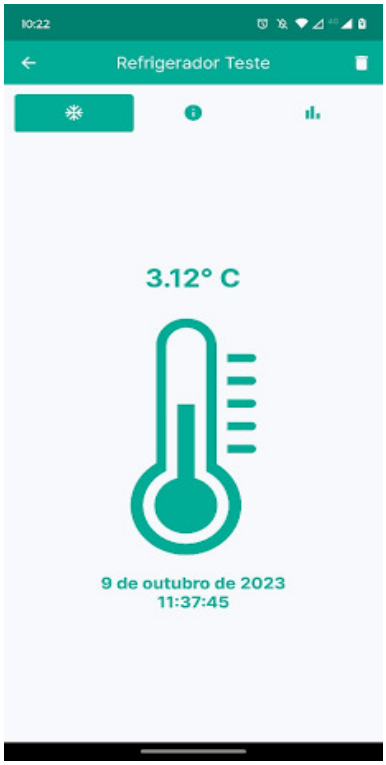

Figura 23 – *Mobile App*: Monitoramento do refrigerador

Fonte: Elaborado pelo autor

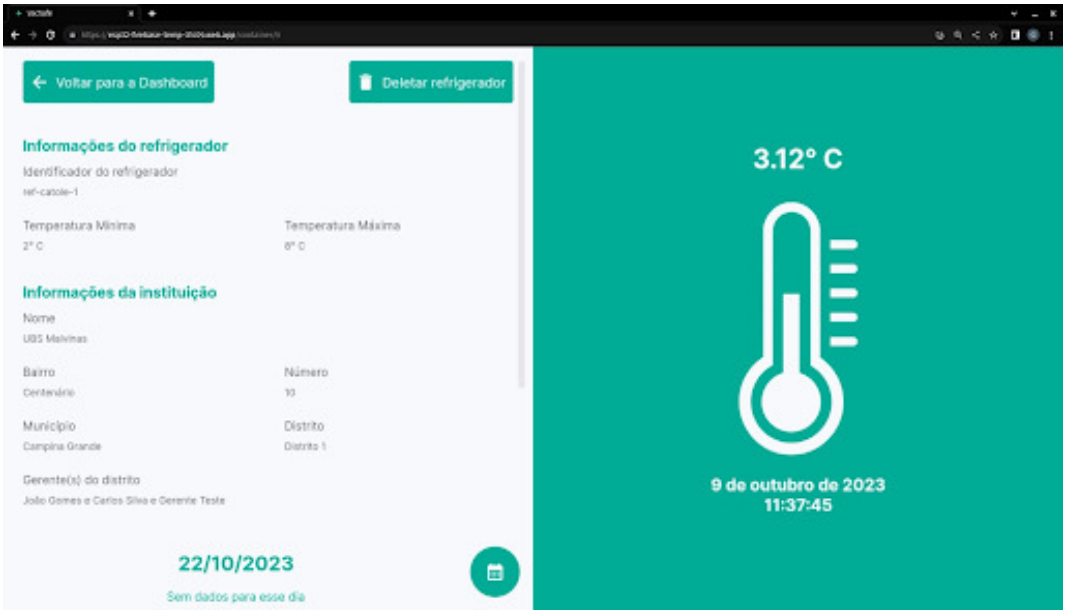

Figura 24 – Web *App*: Monitoramento do refrigerador

Fonte: Elaborado pelo autor

Ademais, como discutido em tópicos anteriores, existe um valor mínimo e máximo em que a temperatura deve estar, dessa forma, uma estratégia idealizada e desenvolvida, de forma a ajudar ainda mais o cliente foi: exibir de forma visual quando estes valores são infligidos, seja o limite mínimo ou máximo. Ou seja, como mostrado na Figura 25 e Figura 26, a página de monitoramento apresenta uma tonalidade azul para alertar, de forma visual, que o valor de temperatura está abaixo do mínimo esperado. Outrora, quando este valor é maior que o máximo definido, uma cor de tonalidade vermelha é exibida, como mostrado nas Figuras 27 e 28.

Vale ressaltar que essa não é a principal estratégia para alertar o cliente, esta é apenas uma forma de, ainda que de maneira visual, deixá-lo mais atento a essas variações indesejadas de temperatura e, em consequência disso, proporcionar que ocorra a tomada de decisão para o ajuste adequado do refrigerador mais rapidamente.

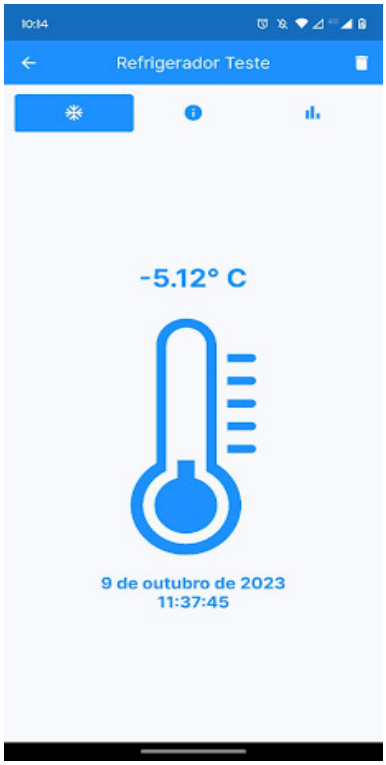

Figura 25 – *Mobile App*: Monitoramento refrigerador (baixa temperatura)

Fonte: Elaborado pelo autor

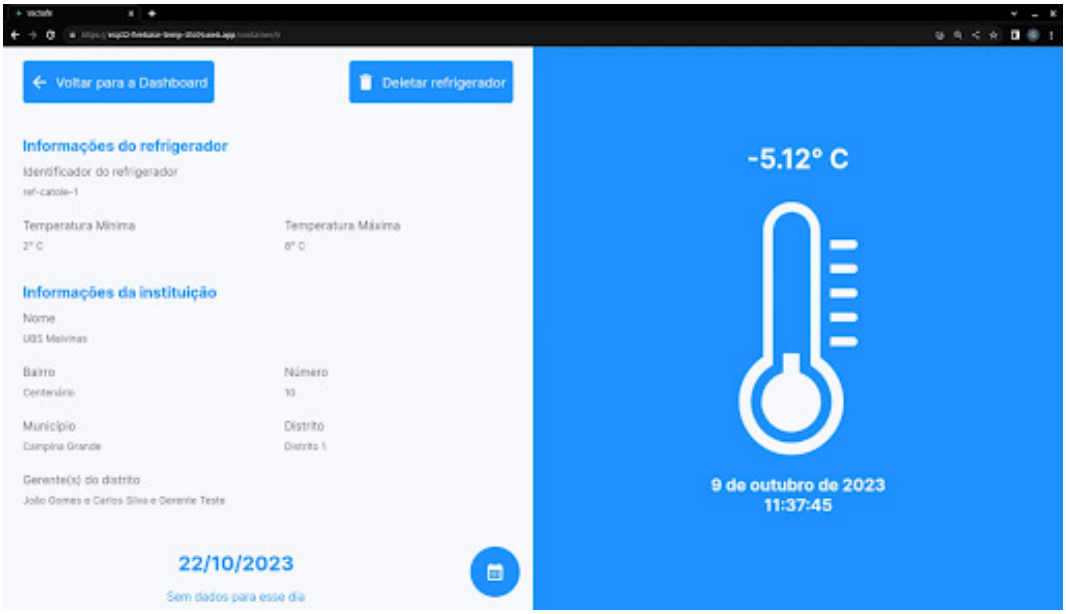

Figura 26 – Web *App*: Monitoramento refrigerador (baixa temperatura)

Fonte: Elaborado pelo autor

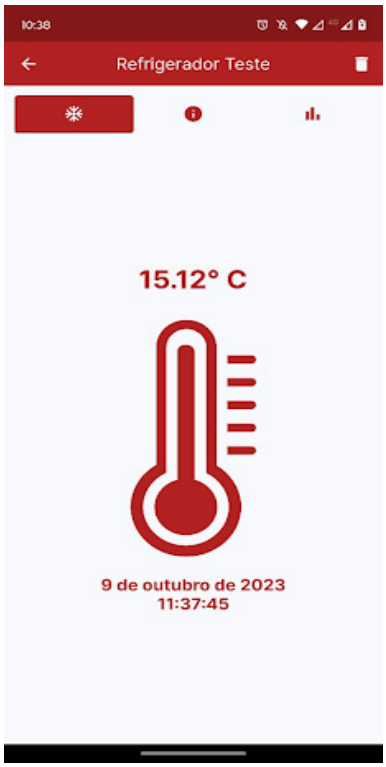

Figura 27 – *Mobile App*: Monitoramento refrigerador (alta temperatura)

Fonte: Elaborado pelo autor

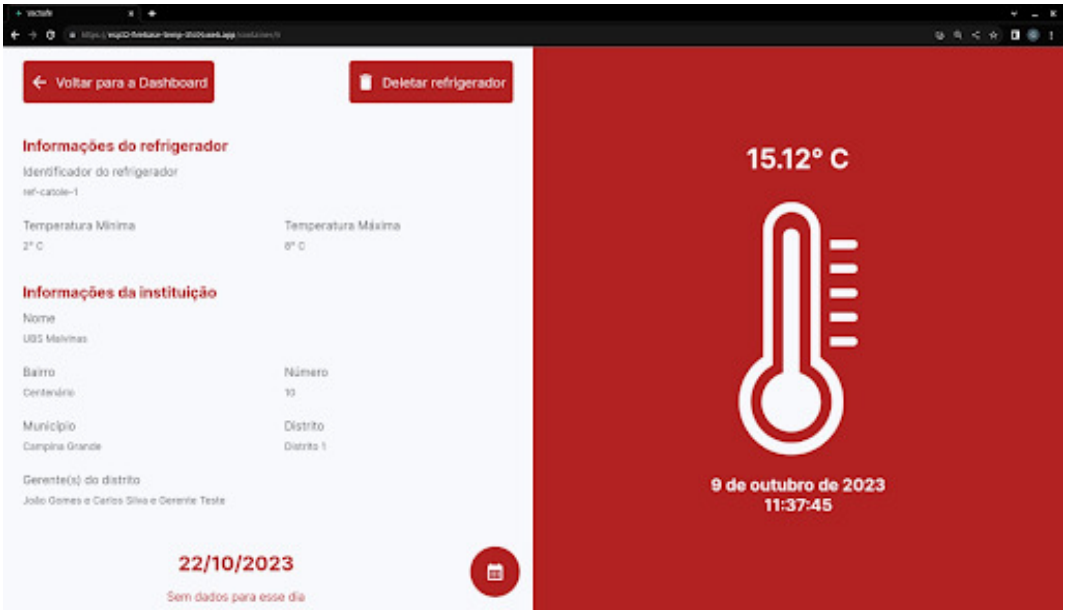

Figura 28 – Web *App*: Monitoramento refrigerador (alta temperatura)

Fonte: Elaborado pelo autor

Continuando, na versão *mobile* da aplicação, a segunda aba da página (Figura 29) mostra informações referentes ao refrigerador e dados sobre a instituição onde o mesmo está alocado. Já na visão web, como na Figura 28, estas mesmas informações estão dispostas na parte esquerda da página.

Por fim, na terceira aba, como se pode observar na Figura 30, existe um gráfico de linha onde o eixo X possui valores crescentes (da esquerda para direita) relativos ao horário do dia, e o eixo Y também valores crescentes (de baixo para cima) relativos às temperaturas mensuradas no dia em questão. Os dados observados no gráfico são resultados de todas as médias das medições de temperaturas realizadas a cada minuto. A forma de plotagem dos dados no gráfico é definida e realizada pela biblioteca flutter_charts[2] do próprio Flutter.

Para facilitar a visualização e entendimento desses dados, e também por questão de performance, o usuário escolhe um dia de qualquer mês e ano (Figura 31), e todas as informações de temperatura apresentadas em tela serão relativas a esse dia em específico. Ou seja, o usuário conseguirá visualizar, além das médias de temperatura em minutos do dia, a temperatura mínima e máxima da data escolhida por ele.

[2] https://pub.dev/packages/syncfusion$_f lutter_c harts$

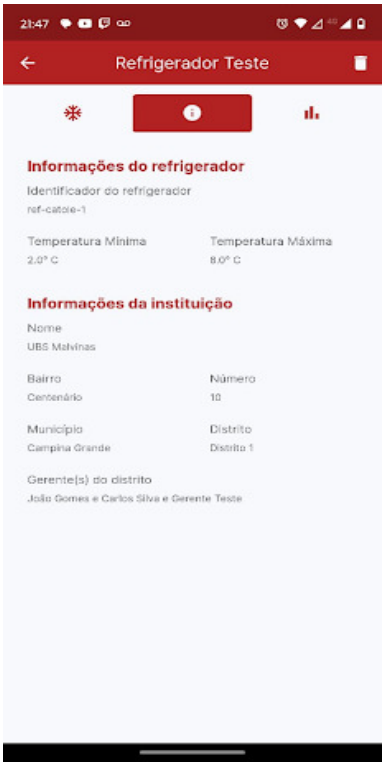

Figura 29 – *Mobile App*: Monitoramento refrigerador (aba de informações)

Fonte: Elaborado pelo autor

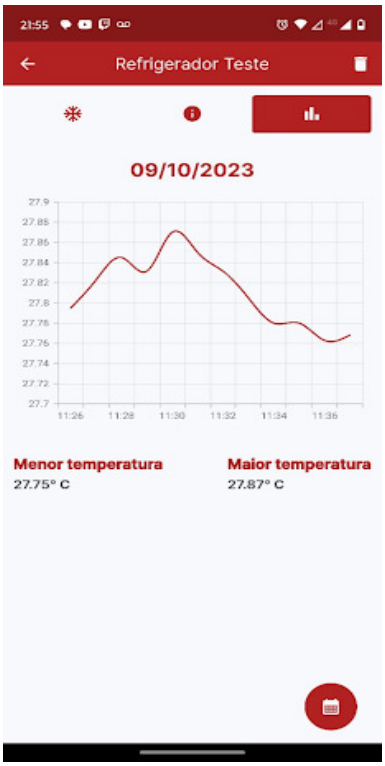

Figura 30 – *Mobile App*: Aba do gráfico

Fonte: Elaborado pelo autor

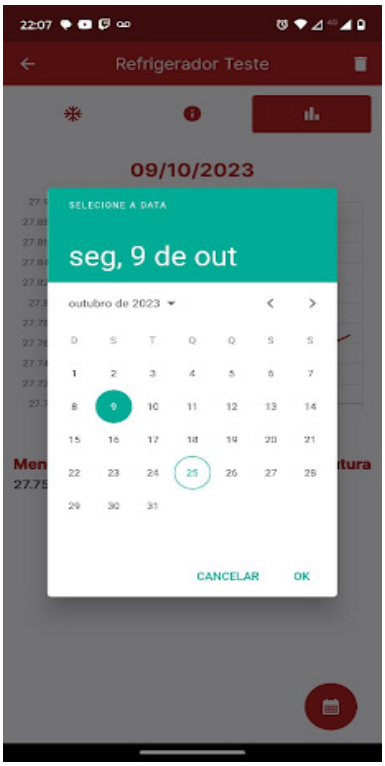

Figura 31 – *Mobile App*: Seleção de data

Fonte: Elaborado pelo autor

Como resultado, o usuário poderá visualizar em tempo real a temperatura de um refrigerador selecionado, além disso, poderá verificar informações do local onde o mesmo está presente, gerentes do distrito onde o posto de saúde está alocado, além de conseguir resgatar valores mensurados em dias anteriores para qualquer tipo de consulta necessária.

Acerca das permissões da tela em questão, pode-se analisar na Tabela 6 quem pode acessar as funcionalidades descritas neste tópico.

Tabela 6 – Detalhe do refrigerador

Tipo de usuário logado	**Tem permissão para monitorar e consultar informações de um refrigerador?**
Técnico	Sim
Coordenador	Sim
Gerente de Distrito	Sim
Agente	Sim

Fonte: Elaborado pelos autores

As funções descritas nesse tópico são funcionalidades centrais da aplicação e serão acessíveis por todos os tipos de usuários que estão cadastrados no sistema.

4.7 Esqueci minha senha

Uma funcionalidade imprescindível em uma aplicação onde é necessário o uso de credenciais para acesso, é o *reset* de senha. Como pode-se observar na Figura 32, na tela de login há um botão com o rótulo "Esqueci minha senha" que quando clicado pelo usuário, ele será redirecionado para outra página onde, para prosseguir com a troca de senha, ele precisará digitar seu e-mail registrado dentro do sistema (Figura 30).

Ou seja, se o e-mail digitado pelo usuário não tiver sido anteriormente cadastrado na aplicação, obviamente o restante do fluxo não ocorrerá, mas, uma vez que o e-mail exista, a API enviará um mensagem para a conta do usuário informando o código que ele precisará inserir na próxima tela para concluir o fluxo (Figura 33).

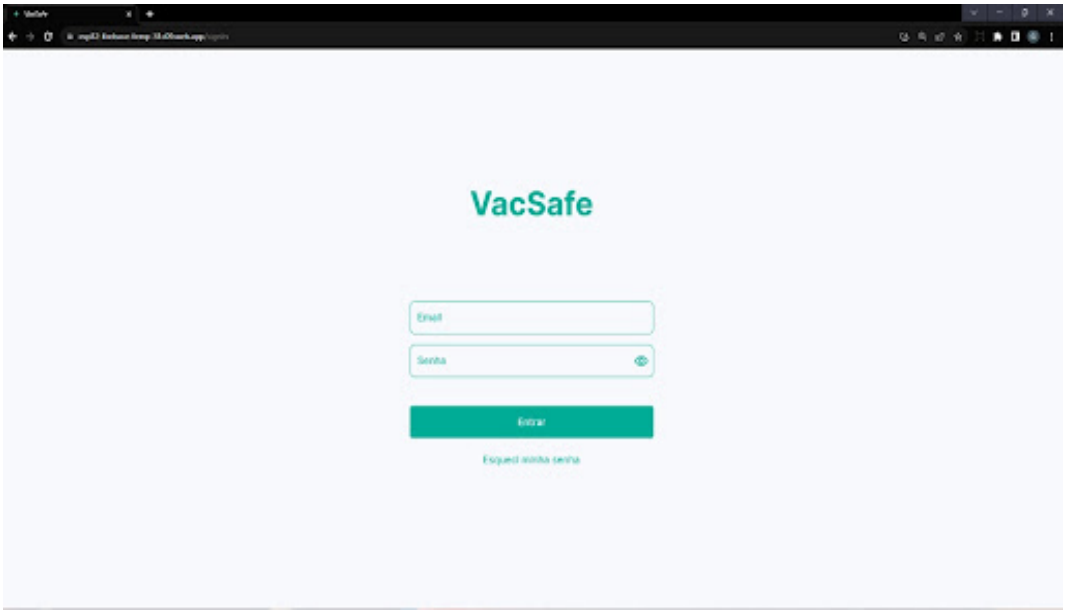

Figura 32 – Web *App*: Tela de login (esqueci minha senha)

Fonte: Elaborado pelo autor

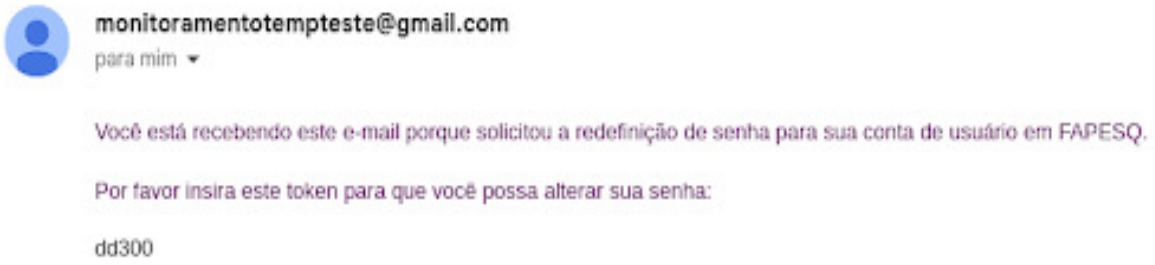

Figura 33 – *Reset* de senha (Código)

Fonte: Elaborado pelos autores

A partir disso, como apresentado na Figura 34, o usuário deverá digitar esse código no input que aparecerá na tela seguinte ao digitar o e-mail. Após isso, ao clicar em "Verificar código" o usuário é então encaminhado a tela que o permitirá digitar uma nova senha, isso caso o código digitado seja válido, caso inválido, uma mensagem de erro aparecerá e o informará (Figura 35).

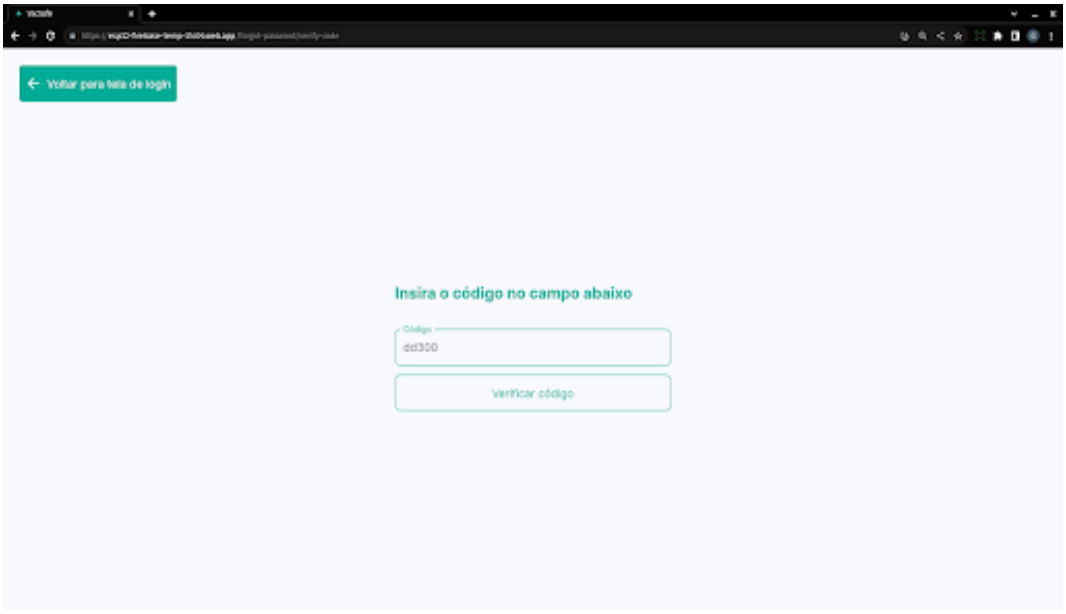

Figura 34 – Web *App*: *Reset* de senha (digitar código)

Fonte: Elaborado pelo autor

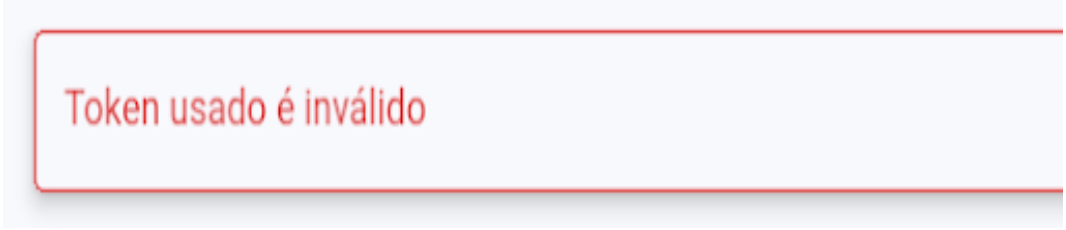

Figura 35 – Mensagem de erro: Código de *reset* de senha inválido

Fonte: Elaborado pelos autores

Por fim, seguindo a ideia que o código digitado está válido, a página na Figura 36 surge, onde nela o usuário deve digitar sua nova senha, duas vezes para garantir assertividade em que, por conseguinte, ao clicar em "Trocar senha", uma mensagem de sucesso é exibida e então ele é redirecionado para a tela de *login* e poderá entrar no sistema com a nova senha.

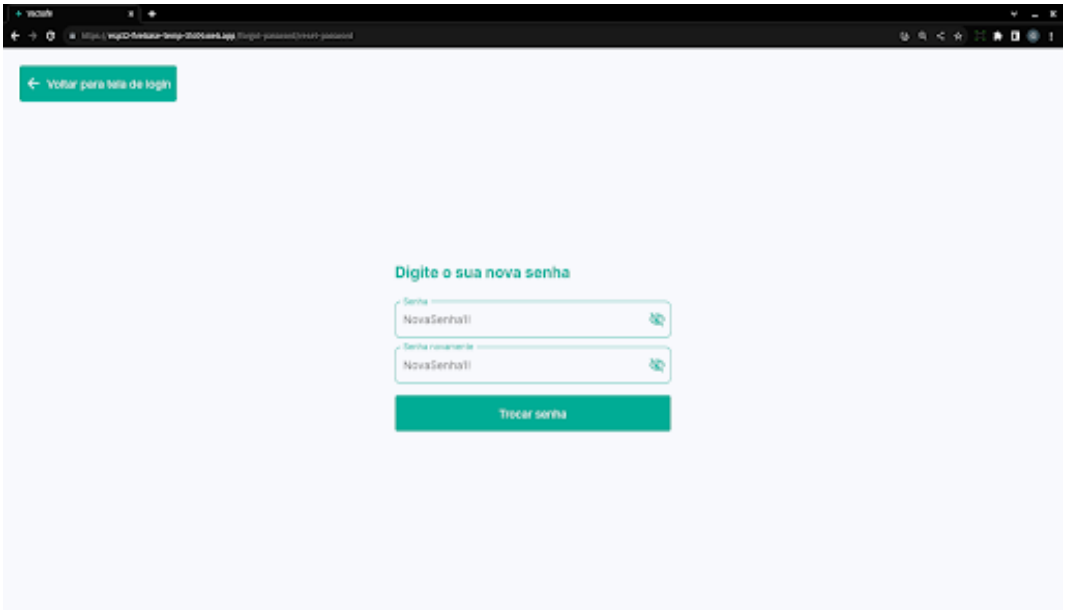

Figura 36 – Web *App*: *Reset* de senha (digitar nova senha)

Fonte: Elaborado pelo autor

Como resultado, o usuário poderá ter acesso novamente a aplicação mesmo que esqueça sua senha, os únicos pré-requisitos seriam:

1. O e-mail deve estar previamente cadastrado no sistema;
2. O usuário deve ter acesso a esse e-mail para poder recuperar a mensagem que contém o código enviado pelo API.

Tabela 7 – Mudança de senha

Tipo de usuário logado	**Tem permissão para trocar a senha?**
Técnico	Sim
Coordenador	Sim
Gerente de Distrito	Sim
Agente	Sim

Fonte: Elaborado pelos autores

4.8 Notificação

A principal forma de alertar o usuário sobre uma variação indesejada de temperatura de um refrigerador é a notificação (Figura 37). Esta se comporta de forma similar como em outros tipos de aplicações, por exemplo: *WhatsApp*, *Facebook*, *Youtube*, *Gmail* e etc.

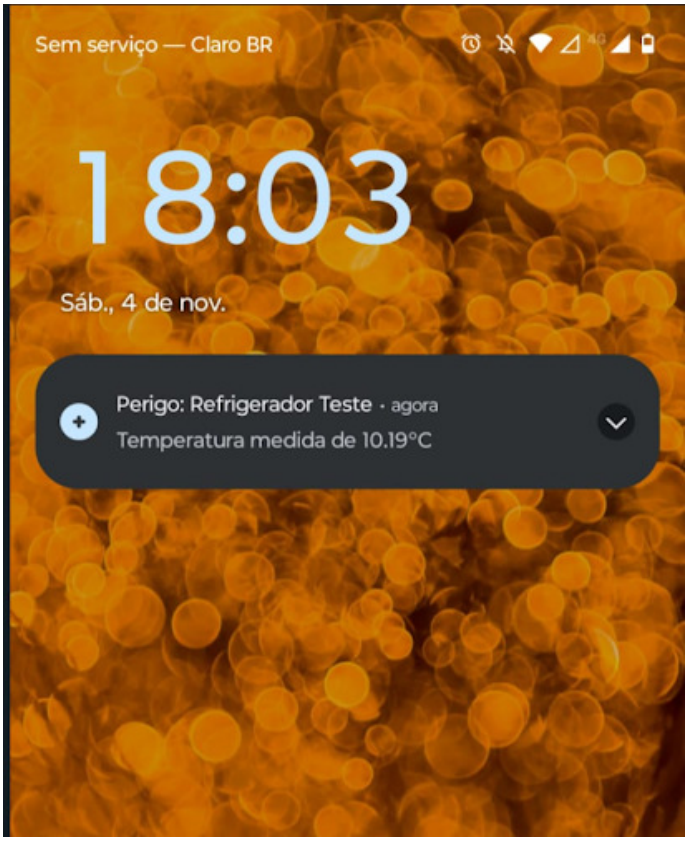

Figura 37 – *Mobile App*: Notificação

Fonte: Elaborado pelos autores

Portanto, conforme é habitual outras aplicações, a notificação independe do *app* estar aberto, fechado ou em segundo plano, independe também do dispositivo estar "bloqueado". A única dependência que esse *feature* (recurso) possui é a conexão com a Internet. Mas, relativo a quem vai receber a notificação, isso depende também de qual refrigerador o alerta foi gerado, como visto na imagem anterior, o conteúdo da notificação contém o apelido do refrigerador e a temperatura medida pelo sensor, ou seja, imaginando o seguinte cenário:

1. Há um refrigerador cadastrado com o apelido Refrigerador 1;
2. Esse refrigerador está alocado no Posto 1;
3. O posto 1 possui os agentes: Agente 1, Agente 2 e Agente 3;
4. Esse Posto 1 está alocado no Distrito 1;
5. O distrito 1 possui gerentes: Gerente 1 e Gerente 2;
6. Esse Distrito 1 está localizado em Campina Grande;
7. Campina Grande possui apenas um coordenador: Coordenador 1.

Baseado nos pontos apresentados de forma enumerada acima, caso ocorra uma medição de um valor atípico no "Refrigerador 1", uma notificação será enviada para todos aqueles agentes relacionados ao posto 1 (Agente 1, Agente 2 e Agente 3), também serão enviados notificações para os gerentes de distrito onde o "Posto 1" está alocado (Gerente 1 e Gerente 2), e serão enviados notificações para os coordenadores de onde o "Distrito 1" está localizado (Coordenador 1). Além destas personas, o técnico responsável por cadastrar o dispositivo no sistema também receberá uma notificação.

Dessa forma, como resultado, temos uma maneira prática e eficiente de alertar o usuário sobre alguma irregularidade no refrigerador, dessa forma, poderá ser realizado a correção e/ou manutenção em tempo hábil.

Tabela 8 – Notificações

Tipo de usuário logado	**Tem permissão para receber notificações?**
Técnico	Sim
Coordenador	Sim
Gerente de Distrito	Sim
Agente	Sim

Fonte: Elaborado pelos autores

5 Conclusão

A versão final do processo de desenvolvimento da aplicação VacMonitor apresentada neste documento, contempla todas as regras de negócio consideradas como essenciais para implementações reais da aplicação. Essa possibilidade se comprova com acompanhamento realizado esporadicamente pela equipe, através de testagem dos fluxos desenvolvidos.

O módulo de monitoramento de um refrigerador, traz avanço aos processos já utilizados por profissionais de saúde, facilitando a execução de tarefas que demandam atenção e intervenção humana, tais quais aferição e persistência do dado da temperatura referente ao armazenamento. Além do monitoramento poder ocorrer em qualquer lugar, e não necessariamente nas dependências da instituição de saúde.

Os módulos de criação de entidades representantes de Refrigerador e Usuário, possibilitam um nível de abstração nos processos de gerência da instituição, agilizando rotinas administrativas. Isto se dá, especialmente pelos requisitos, regras de negócio, terem sido planejados estrategicamente para atender o uso dessas rotinas, em instituições de saúde da rede pública. Resultando assim em um suporte mais específico para a realização destas rotinas.

O módulo de envio de alertas por meio de notificações na aplicação, também pode ser ressaltada como progresso no quesito "eficiência", nos trâmites que envolvem o armazenamento, possibilitando o conhecimento de um possível incidente, a todas as partes interessadas no gerenciamento do determinado refrigerador. Apesar do alerta gerado por a aplicação nessa iteração, ainda não ser caracterizada como um artefato oficial de documentação de incidentes – usado por entidades de secretarias de saúde –, tem o potencial de agilizar ações remediadoras aos mesmos.

Por fim, o módulo de processamento dos dados persistidos, traz um leque de análises primárias para os dados disponíveis para determinados refrigeradores. Ajudando o usuário na tomada de decisão quanto a fatores determinantes no armazenamento. Este módulo é resultado de uma implementação breve, ainda oferecendo funcionalidades básicas, mas é o módulo da aplicação com o maior potencial de desenvolvimento técnico, podendo empregar funcionalidades de complexidade altamente elevadas, dependendo das necessidades do usuário.

5.1 Trabalhos Futuros

Como sugestão para trabalhos posteriores:

- Expansão de funcionalidades de gerenciamento dos usuários, aprimorando o sistema de hierarquias e delegação de responsabilidades;
- Implementação da criação dos artefatos de registro de leitura de temperatura e notificações de incidente, de forma que os resultados gerados, possam ser usados como documentos oficiais aceitos pelas entidades regulamentadoras;
- Aprimoramento dos métodos empregados no módulo de processamento de dados, de modo a oferecer funcionalidades com análises mais aprofundadas para os usuários responsáveis pelo monitoramento;
- Aprimoramento da API Rest desenvolvida de forma que diminua a dependência total ou parcial do uso do Firebase dentro do sistema que, por conseguinte, facilitaria a implementação das funcionalidades descritas nos tópicos anteriores, além da possível redução de custo por não mais utilizar serviços terceiros;
- Suporte via chat ou envio de e-mails: possibilitar que o usuário a qual está monitorando um refrigerador entre em contato com um técnico responsável, seja para relatar algum problema nas medições ou qualquer outro tipo de inconsistência;
- Envio de e-mail de registro de novo usuário: ao cadastrar um novo usuário, um e-mail seria enviado para essa nova entidade com um passo a passo de como ativar sua conta e fazer um reset de senha;
- Implementação de login social;
- Ajuste no layout da aplicação (tanto web quanto mobile) visando uma melhor usabilidade;
- Autenticação de dois fatores.

Referências

ADSERVIO. **What is Flutter and What Are Its Advantages**. 2022. Acesso em: 4 de setembro de 2023. Disponível em: <https://www.adservio.fr/post/what-is-flutter-and-what-are-its-advantages>.

AL-FUQAHA, A. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE Communication Surveys Tutorials**, v. 17, n. 4, p. 1–2, 2015.

ALBERTO, M. **Flutter: o que é, vantagens e desvantagens**. 2023. Acesso em: 4 de setembro de 2023. Disponível em: <https://www.alura.com.br/artigos/flutter#:~:text=O%20Flutter%20utiliza%20como%20base,por%20exemplo%2C%20o%20React%20Native>.

ALMENARA, I.; CIRIACO, D. **Qual o sistema operacional de celular mais usado do mundo?** 2022. Acesso em: 4 de setembro de 2023. Disponível em: <https://canaltech.com.br/software/qual-o-sistema-operacional-de-celular-mais-usado-do-mundo-223862/>.

Amazon Web Services. **O que é uma API RESTful?** 2023. <https://aws.amazon.com/pt/what-is/restful-api/>. Acesso em: 4 de setembro de 2023.

ANDRADE, G. L. C. d. **Desenvolvimento em nuvem: Um estudo de caso utilizando o Firebase como servidor backend**. 2018. Acesso em: 4 de setembro de 2023. Disponível em: <https://di.uern.br/tccs2019/html/ltr/PDF/014005697.pdf>.

ASHTON, K. **That 'Internet of Things' Thing**. 2009. Acesso em: 4 de setembro de 2023. Disponível em: <http://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf>.

CHASE, O.; ALMEIDA. **Sistemas embarcados**. 2007. 13 p. <www.sbajovem.org/chase>.

DJANGO. **Django Web Framework. Página inicial**. 2023. Acesso em: 4 de setembro de 2023. Disponível em: <https://www.djangoproject.com/>.

ELETROGATE. **Sensor de Temperatura DS18B20 a Prova D'água**. Acesso em: 20 de dez. de 2023. Disponível em: <https://www.eletrogate.com/sensor-de-temperatura-ds18b20-a-prova-dagua>.

FIGUEIREDO, C. M. S.; NAKAMURA, E. Computação móvel: Novas oportunidades e novos desafios. **T&C Amazônia**, v. 1, n. 2, p. 21, 2003.

FLUTTER. **Flutter. Página Inicial**. 2023. Acesso em: 4 de setembro de 2023. Disponível em: <https://flutter.dev/#:~:text=Flutter%20is%20an%20open%20source,applications%20from%20a%20single%20codebase>.

GOOGLE FIREBASE. **Produtos Firebase**. 2023. Acesso em: 4 de setembro de 2023. Disponível em: <https://firebase.google.com/products-build?authuser=2&hl=pt-br>.

INTEGRATED, M. **DS18B20 - Programmable Resolution 1-Wire Digital Thermometer**. 2019. Acesso em: 13 de novembro de 2023. Disponível em: <https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf>.

KERSCHBAUMER, R. **Microcontroladores**. Santa Catarina, Brasil: [s.n.], 2013. Disponível em: <https://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/wp-content/uploads/sites/43/2018/02/Apostila-Microcontroladores.pdf>.

MARTíNEZ, A. C.; ALVAREZ-MON, M. O sistema imunológico (i): conceitos gerais, adaptação ao exercício físico e implicações clínicas. **Revista Brasileira de Medicina do Esporte**, Niterói, v. 5, n. 3, 1999.

MATEUS, G. R.; LOUREIRO, A. A. F. **Introdução à Computação Móvel**. [S.l.: s.n.], 1998.

MOUHA, R. A. Internet of things (iot). **Journal of Data Analysis and Information Processing, v. 9, n. 2, p. 77-101**, 2021.

PATINE, F.; LOURENçãO, L.; WYSOCKI, A.; SANTOS, M.; RODRIGUES, I.; VENDRAMINI, S. Analysis of vaccine loss due to temperature change. **Rev Bras Enferm**, v. 74, n. 1, p. e20190762, 2021.

PEDRO, W. **O que é Flutter em programação?** 2022. Acesso em: 4 de setembro de 2023. Disponível em: <https://tecnoblog.net/responde/o-que-e-flutter-em-programacao/>.

PEREIRA, M. R. S. **A aplicação do microcontrolador ESP32 no ensino: medindo posições em função do tempo utilizando o sensor VL53L0X associado ao ESP32**. 2021. Acesso em: 13 de novembro de 2023. Disponível em: <http://repositorio.unifap.br/handle/123456789/898>.

Redação XP Educação. **Linguagens de Back-End**. 2022. Atualizado em: 25 de agosto de 2022. Acesso em: 4 de setembro de 2023. Disponível em: <https://blog.xpeducacao.com.br/linguagens-back-end-2/>.

SAÚDE, B. M. da Saúde. Secretaria de Vigilância em. **Manual de Rede de Frio do Programa Nacional de Imunizações**. 5. ed. Brasília, DF, 2017.

TINOCO, A. C. R. L. C. e. a. Diagnóstico situacional da rede de frio nas unidades básicas de saúde do município de niterói-rj. Universidade Federal Fluminense, 2010.

WENDLING, M. **Sensores**. São Paulo, 2010. 20 p.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

Versão Final do TCC

Assunto:	Versão Final do TCC
Assinado por:	Gabriel Araujo
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Antonio Gabriel Araújo Silva**, **ALUNO (201821250015) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 09/10/2024 11:27:02.

Este documento foi armazenado no SUAP em 09/10/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1272278
Código de Autenticação: 6113885233