

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

HENRIQUE MARTINS MIRANDA

**SISTEMA DE MONITORAMENTO DE QUALIDADE DE  
IMUNOBIOLÓGICOS NA CADEIA DE DISTRIBUIÇÃO E  
ARMAZENAMENTO**

Campina Grande

2021

Henrique Martins Miranda

**Sistema de Monitoramento de Qualidade de  
Imunobiológicos na Cadeia de Distribuição e  
Armazenamento**

Monografia apresentada à Coordenação do  
Curso de Telemática do IFPB - CampusCampina  
Grande, como requisito parcial para con-  
clusão do curso de Tecnologia em Telemática.

Orientador: Paulo Ribeiro Lins Junior

Campina Grande  
2021

# Resumo

O armazenamento é um dos componentes mais importantes da cadeia de distribuição de vacinas, principalmente pela sensibilidade delas a variações de temperatura, que podem ocasionar diminuição da sua eficácia. Considerando isso, existe uma necessidade de manter constante monitoramento dessa variável, a fim de garantir que o produto final venha a manter suas características originais e a eficiência esperada. Esse trabalho apresenta uma solução baseada em Internet das Coisas, usando comunicação sem fio de baixa potência, para monitorar a qualidade de vacinas, por meio das medidas de temperatura e umidade nos locais de armazenamento. Os dados coletados são organizados em um banco de dados, podendo ser acessados por sistemas decisórios com a finalidade de avaliar a qualidade da vacina antes de sua aplicação, evitando transtornos em decorrência de problemas de armazenamento.

**Palavras-chaves:** Monitoramento. Vacinas. IoT. LoRa.

# Abstract

Storage is one of the most important components of the vaccine distribution chain, mainly due to its sensitivity to temperature variations, which can cause a decrease in its effectiveness. Considering this, there is a need to keep constant monitoring of this variable, to guarantee that the final product will maintain its original characteristics and the expected efficiency. This work presents a solution based on the Internet of Things, using low power wireless communication, to monitor the quality of vaccines, by measuring temperature and humidity in storage locations. The collected data are organized in a database, which can be accessed by decision systems to assess the quality of the vaccine before its application, avoiding problems due to storage problems.

**Key-words:** Monitoring. Vacaciones. IoT. LoRa.

# **Lista de ilustrações**

Figura 1 – Representação da relação entre os dispositivos LoRa (Adaptada de [1]).	12
Figura 2 – Exemplo de uma requisição POST (Retirado de [2]).	13
Figura 3 – Dimensões em milímetros do sensor DHT-22 (Retirado de [3]).	14
Figura 4 – Representação do modelo cliente/servidor (Autoral).	15
Figura 5 – Ciclo de eventos no Node (Autoral).	16
Figura 6 – Composição de um banco de dados do InfluxDB (Autoral).	17
Figura 7 – Comparação entre o modelo de virtualização e modelo de containers (Adaptada de [4]).	18
Figura 8 – Representação da conexão entre os componentes principais do end node (autoral).	22
Figura 9 – Foto do primeiro protótipo (autoral).	23
Figura 10 – Foto do segundo protótipo (autoral).	24
Figura 11 – Foto do ESP32 LoRa da Heltec Automation (autoral).	25
Figura 12 – Página de configuração do gateway (autoral).	26
Figura 13 – Medidas dentro do banco de dados InfluxDB (autoral).	28
Figura 14 – Modelo entidade relacionamento do PostgreSQL (autoral).	29
Figura 15 – Desenho das telas de entrar e cadastrar na aplicação (autoral).	30

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>7</b>
<b>1.1</b>	<b>O Programa Nacional de Imunizações</b>	<b>7</b>
<b>1.2</b>	<b>Rede de Frio</b>	<b>8</b>
<b>1.3</b>	<b>Justificativa e Relevância do Trabalho</b>	<b>8</b>
<b>1.4</b>	<b>Objetivos</b>	<b>9</b>
1.4.1	Objetivo Geral	9
1.4.2	Objetivos Específicos	9
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>11</b>
<b>2.1</b>	<b>Internet das Coisas</b>	<b>11</b>
<b>2.2</b>	<b>Protocolos de Comunicação</b>	<b>11</b>
2.2.1	LoRa	12
2.2.2	HTTP/HTTPS	12
<b>2.3</b>	<b>Plataformas de Prototipagem</b>	<b>13</b>
<b>2.4</b>	<b>Sensor DHT-22</b>	<b>14</b>
<b>2.5</b>	<b>Servidor</b>	<b>14</b>
2.5.1	Node.js	15
2.5.2	InfluxDB	16
2.5.3	Docker	17
2.5.4	Computação em Nuvem	18
<b>2.6</b>	<b>Aplicativo Móvel</b>	<b>18</b>
2.6.1	React Native	19
<b>2.7</b>	<b>Trabalhos Relacionados</b>	<b>19</b>
<b>3</b>	<b>METODOLOGIA</b>	<b>21</b>
<b>3.1</b>	<b>Estrutura dos Pacotes</b>	<b>21</b>
<b>3.2</b>	<b>End Node</b>	<b>21</b>
3.2.1	Primeiro Protótipo	23
3.2.2	Segundo Protótipo	24
<b>3.3</b>	<b>Gateway</b>	<b>24</b>
<b>3.4</b>	<b>Servidor</b>	<b>26</b>
3.4.1	Padrões de Projetos	26
3.4.2	Banco de Dados	27
3.4.3	Containers	29
3.4.4	Deploy	29
<b>3.5</b>	<b>Aplicativo móvel</b>	<b>30</b>



# 1 Introdução

A saúde é um fator de suma importância para todos os seres vivos, ele é um problema científico, tecnológico, político, prático e filosófico que refere-se a um estado completo de bem estar físico, emocional, social, intelectual e espiritual [5].

Segundo o artigo 196 [6] da Constituição Federal Brasileira a saúde é um direito de todos e dever do Estado garantir medidas políticas sociais e econômicas que visam à diminuição do risco de doenças e de outros agravamentos e ao acesso universal e imparcial às ações e serviços para a sua promoção, proteção e recuperação.

Para garantirmos nossa saúde, precisamos cuidar do nosso corpo e mente, para isto, uma ferramenta que podemos contar são os imunobiológicos, como as vacinas e os soros, diferente de remédios que ajudam no tratamento de pessoas doentes, as imunobiológicos são uma preparação biológica que fornece imunidade total ou parcial de uma determinada doença autoimune para um indivíduo saudável. As vacinas e os soros se diferem pela sua forma de imunização, as vacinas fornecem uma imunização ativa, estimulando o nosso organismo na produção de anticorpos, os soros fornecem uma imunização passiva, provendo os anticorpos para o nosso organismo que foram produzidos em outros organismo [7].

Contudo, os imunobiológicos requerem um cuidado elevado para manter a qualidade e sua eficiência, um dos fatores é que são produtos termolábeis, ou seja, se deterioram após determinado tempo expostos a variações de temperaturas e umidade, portanto, é imprescindível assegurar que seu ambiente de armazenagem mantenha uma temperatura e umidade constante [8] para garantir uma longevidade maior para o produto. Para este propósito, existe a Rede de Frio, um processo desenvolvido pelo Programa Nacional de Imunizações, PNI, de conversação, armazenamento e transporte dos medicamentos, objetivando as condições adequadas dos mesmos, mantendo suas características iniciais [8].

No ano de 2014, foi relatado no estudo [9] que a qualidade de conservação das vacinas não eram adequadas em boa parte dos municípios da macrorregião Oeste de Minas Gerais, alguns dos motivos citados foram a má gestão dos refrigeradores, falhas no monitoramento da temperatura e insuficiência de recursos humanos.

## 1.1 O Programa Nacional de Imunizações

Com o sucesso da Campanha de Erradicação da Varíola, CEV, iniciada em 1965, tendo seu fim em 1973 [10], amplificou dentro do Ministério da Saúde maiores investimentos no controle de doenças autoimunes, dando um impulso na criação do PNI [11]. O PNI foi fundado com objetivo de controlar e erradicar as doenças imunopreveníveis, através de

ações metalizadas de vacinação da população. Em 1980 foi realizada a primeira campanha de vacinação da poliomielite e desde então foram realizadas diversas campanhas, tais como a da rubéola, sarampo, tuberculose, febre amarela [11, 8] e atualmente contra a COVID-19.

De acordo com a Lei n.º 6.259 de 30 de outubro de 1975, regularizada pelo Decreto nº 78.231 em 1976, certificar o PNI, sobre a responsabilidade do Ministério da Saúde e define as seguintes competências [8]:

- implantar e implementar as ações do Programa, relacionadas com as vacinações de caráter obrigatório;
- estabelecer critérios e prestar apoio técnico e financeiro à elaboração, implantação e implementação dos programas de vacinação a cargo das secretarias de saúde das unidades federadas;
- estabelecer normas básicas para a execução das vacinações;
- supervisionar, controlar e avaliar a execução das vacinações no território nacional, principalmente o desempenho dos órgãos das Secretarias de Saúde, encarregados dos programas de vacinação.

## 1.2 Rede de Frio

A Rede de Frio, também chamada de Cadeia de Frio é um processo definido pelo PNI designado a auxiliar os profissionais da área da saúde, responsáveis pela imunização no Brasil, para que possa assim, garantir a efetividade e durabilidade dos imunobiológicos e medicamentos termolábeis.

No Manual de Rede de Frio [8] é definido os requisitos dos ambientes de armazenagem para garantir a efetividade dos produtos, desde os laboratórios produtores às instâncias locais, passando pela instância nacional, estadual, e no transporte entre eles, para as Câmaras frigoríficas, a temperatura de operação é entre -20°C a +2°C, variando conforme o material armazenado, para a maioria dos imunobiológicos o recomendado é de +2°C e +8°C para ter um melhor controle da sua validade, havendo algumas exceções, como por exemplo, as vacinas Pfizer-BioNTech e Moderna, produzidas para combater o COVID-19, que precisam ser armazenadas entre -80°C a -60°C e -25°C e -15°C, respectivamente [12].

## 1.3 Justificativa e Relevância do Trabalho

Atualmente, com o início da distribuição das vacinas contra o COVID-19 em todo o mundo, uma das dificuldades enfrentadas é o controle de qualidade no armazenamento

tanto em transporte [13], quanto no local da aplicação justamente por serem produtos sensíveis a temperatura e necessitarem de muita cautela, em contrapartida, por ser um produto com uma demanda elevada, a sua oferta deve ser rápida para que haja imunização em massa da população, encaminhando-se para o fim da pandemia.

Esses desafios enfrentados na distribuição das vacinas do COVID-19 não são exclusivamente deles, enfermeiras de Minas Gerais, com o objetivo de inteirar-se acerca do sistema de manutenção dos produtos, realizaram um estudo sobre a conservação de vacinas em unidades básicas de saúde, UBSs. Nessa pesquisa foi relatado diversas irregularidades no armazenamento dos materiais termolábeis sem o comprimento das normais da PNI, como por exemplo, a presença de vacinas que deveriam ter sido descartadas por terem atingido seu tempo máximo de diluição, ainda presentes nos refrigeradores, cerca de 52% dos imunobiológicos armazenados nos refrigeradores eram estabelecidos erroneamente e 36% dos refrigeradores observados contavam com objetos portas como fracos vazios.

Analizando as temperaturas dos ambientes de armazenagem, foi observado que 4% das unidades não realizavam o registro da temperatura dos refrigeradores, 88% dos refrigeradores usavam termômetros analógicos de baixa confiabilidade e cerca de 12% foram analisados na hora da visita temperaturas abaixo da faixa recomendada (+2°C a +8°C), chegando a 0°C. Outros estudos realizado [9, 14, 15] também testemunharam essa irregularidade na temperatura em seus respectivos locais ao longo do mundo, valendo salientar o estudo realizado na Bolívia [14], que teve resultados ainda piores, sendo registado um temperatura mínima de -7.2°C e uma máxima de 22.7°C.

Pensando nesse cenário, esse trabalho apresenta uma alternativa para melhorar a forma de monitoramento de temperatura e umidade realizada em produtos imunobiológicos por laboratórios, unidades públicas de saúde e afins, no intuito de auxiliá-los a manter os materiais em suas melhores condições.

## 1.4 Objetivos

### 1.4.1 Objetivo Geral

Construir uma arquitetura baseada em conceitos de IoT visando o monitoramento de temperatura e umidade de imunobiológicos para auxiliar funcionários da saúde, garantindo melhores condições para a vacinação da população frente a incidência de doenças.

### 1.4.2 Objetivos Específicos

- Construir um protótipo inicial para coleta da temperatura e umidade nos ambientes de armazenagens dos imunobiológicos.

- Implementar um servidor para a armazenagem dos dados coletados e posteriormente fornecer históricos das temperaturas e umidade ao aplicativo móvel.
- Desenvolver um aplicativo móvel para fornecer uma interface amigável para os usuários, auxiliando no controle de qualidade dos produtos.
- Realizar testes e análises dos dados de transmissões a fim de garantir a confiabilidade das temperaturas e umidade coletadas.

## 2 Fundamentação Teórica

Neste capítulo serão abordados os principais conceitos e tecnologias utilizadas no desenvolvimento deste projeto, do hardware à aplicação móvel, iniciando pelo conceito principal, a Internet das Coisas.

### 2.1 Internet das Coisas

Com o intuito de ampliar a internet atual, interligando os objetos do nosso cotidiano, animais e humanos em uma única rede, foi criado a Internet das Coisas, IoT, também conhecida como internet de todas as coisas. Para tal, os objetos viram objetos inteligentes, possuindo capacidade de comunicação associados com sensores que fornecem dados para outros dispositivos. Estes aparelhos, conectados a IoT, se comunicam via internet, trocando dados em tempo real, transmitindo informações acerca do ambiente que estão inseridos e/ou até mesmo dos seus respectivos estados.

Deste modo, é adicionando uma nova gama de possibilidades, trazendo grandes benefícios para ambientes domésticos, mas principalmente para a zona industrial. No primeiro caso, aplicações, tais como: aprendizagem reforçada, monitoramento e vigilância inteligentes e vida assistida, têm despontado entre aquelas que mais chamam atenção, tanto dos usuários, como das empresas de desenvolvimento de soluções tecnológicas. No segundo caso, IoT se apresenta como um diferencial competitivo importante em campos, com a automação e manufatura industrial, logística, gestão de processos de negócio, etc.

Do ponto de vista da produtividade, IoT apresenta-se como um importante meio pelo qual pode-se desenvolver aplicações sofisticadas que podem integrar o mundo real e o mundo virtual. Em empresas de manufatura, produtos conectados permitem a existência de um ambiente de serviços e produtos no qual a manutenção pode ser realizada com base na necessidade real, em vez de uma suposição estatística. Adicionalmente, produtos e máquinas conectados podem receber atualizações de software quando disponíveis, para garantir que estejam sempre funcionando em eficiência ótima.

### 2.2 Protocolos de Comunicação

Protocolos de comunicação são conjuntos de normas que definem uma forma de troca de mensagens entre dois ou mais dispositivos em um determinado meio físico. Os meios mais comuns são por cabos de materiais condutores e pelo ar, transmitindo ondas eletromagnéticas, na IoT o que se predomina é por meio sem fio (*wireless*). Neste projeto,

foram utilizados dois protocolos de comunicações, o LoRa e o HTTP.

### 2.2.1 LoRa

A tecnologia Long Range, LoRa, é uma forma de comunicação sem fio, semelhante ao Wi-Fi e ao Bluetooth, que permite um longo alcance de comunicação com baixo custo. O raio de comunicação sem fio utilizando o LoRa, dependendo do dispositivo selecionado, pode alcançar quilômetros de distância.

Embora o LoRa tenha sido fundamentalmente desenvolvido pela Semtech Corporation, seu padrão imposto pelo LoRa permitiu que muitas empresas a utilizassem para diversos projetos com um custo benefício satisfatório, aumentando o ecossistema e ganhando um envolvimento significativamente maior, uma variedade superior de produtos e um acréscimo geral no uso e aceitação.

O LoRa em si diz respeito à camada física, sendo a camada lógica é chamada de LoRaWAN, um protocolo usado pelo LoRa para comunicação entre pontos de conexões de um nós final (*End Nodes*) para envios de informações diretamente a um concentrador (*Gateway*), que centraliza as informações e envia a um determinado sistema [16]. Os End Nodes são dispositivos responsáveis por coletar os dados necessários e transmiti-los a um determinado Gateway, o Gateway por sua vez, é responsável por receber as informações de múltiplos End Nodes e repassá-lo a um determinado sistema, por exemplo um servidor onde será armazenado esses dados, como podemos ver na figura 1.

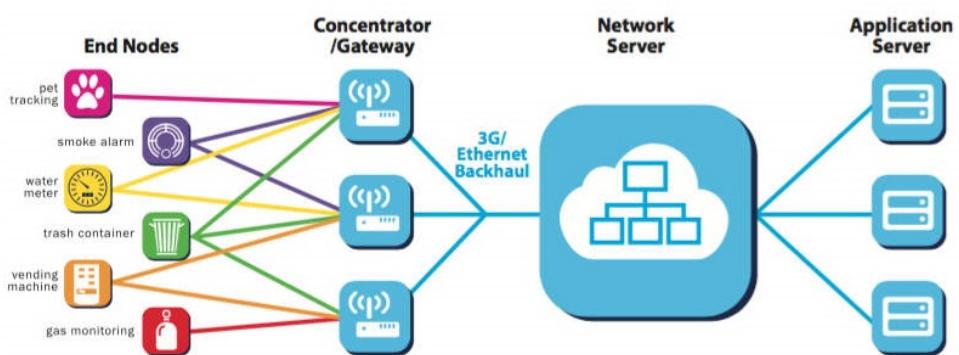


Figura 1 – Representação da relação entre os dispositivos LoRa (Adaptada de [1]).

### 2.2.2 HTTP/HTTPS

Inicialmente utilizado pelo *World-Wide Web*, WWW, em 1990, como o protocolo base da internet por ser uma protocolo leve e rápido para transições de informações em sistemas distribuídos, o *Hypertext Transfer Protocol*, HTTP, é um protocolo que tramite documentos de diversos formatos através da sistemas distribuídos, mais conhecido pela sua disseminação pela internet [17]. Quanto anos depois, em 1994, a Netscape Communications

criou o HTTPS, uma implementação do HTTP adicionando uma nova camada, a de segurança, utilizando-se do protocolo SSL/TLS, um protocolo que fornece segurança entre comunicações sobre rede de computadores.

O HTTP funcionando baseado no paradigma de cliente e servidor, onde o cliente estabelece uma conexão fazendo uma requisição solicitando algo ao servidor, e o servidor, por sua vez, o responde enviando uma mensagem. Tais requisições possuem um método atrelado a ele, esses métodos são usados semanticamente, com o objetivo de organizar e dividir suas responsabilidades, os métodos mais utilizados são: GET, POST, PUT e DELETE. o GET é usado quando o cliente solicita dados ao servidor, o POST quando o cliente quer enviar alguma nova informação, o PUT é semelhante ao POST, entretanto, é referente a atualização de um dado existente e o DELETE é uma requisição pedindo a remoção de uma determinada informação [2].

Essas mensagens que transitam entre clientes e servidores, são compostas por linhas de textos divididas em três blocos como pode ser visto na figura 2. O primeiro bloco consiste em uma linha com o método da requisição, a url, ou seja, o caminho do documento seguido pela a versão do protocolo; o segundo bloco é referente ao cabeçalho HTTP, responsável por fornecer informações e metadados ao servidor, como o tipo de dado transitado ou dados que alteram seu comportamento; o terceiro bloco é respectivo ao bloco de dados, este bloco é opcional, dependendo do tipo de requisição realizada, mais utilizado pelos métodos POST e PUT.

```
POST /contact_form.php HTTP/1.1
Host: developer.mozilla.org
Content-Length: 64
Content-Type: application/x-www-form-urlencoded

name=Joe%20User&request=Send%20me%20one%20of%20your%20catalogue
```

Figura 2 – Exemplo de uma requisição POST (Retirado de [2]).

## 2.3 Plataformas de Prototipagem

Plataformas de prototipagem eletrônica são formas facilitadas de desenvolver um protótipo inicial de determinado projeto. Uma das plataformas de prototipagem existentes mais famosa é o Arduino, uma plataforma de código aberto criada em 2005 na Itália pelo professor Massimo Banzi, que tinha como objetivo, ensinar eletrônica e programação para seus alunos, provendo uma forma acessível e de fácil uso. O Arduino possui seu próprio ambiente de desenvolvimento de mesmo nome, utilizando de sua linguagem de

programação baseada em C.

Outra plataforma bem conhecida é a família de microcontroladores ESP, e assim como suas concorrentes, oferece módulos de baixo custo e versáteis para várias soluções tecnológicas, dos quais destacam-se a série ESP32. Tal série, acabou sendo mais conhecida pelas suas variações de modelos feitas por empresas distintas, cada modelo possuindo seus diferenciais, como por exemplo o modelo da Heltec Automation, que inclui conectividades com WiFi, bluetooth e LoRa.

## 2.4 Sensor DHT-22

O sensor DHT-22 é um sensor de temperatura e umidade da família de sensores DHT comumente utilizado em aplicações de IoT por ser um sensor pequeno (como podemos ver na figura 3) e possuir um baixo custo. Ele opera na faixa de 3.3 a 6 Volts e consegue capturar dados de temperatura entre -40°C a 80°C e umidade entre 0% a 100% RH, com uma acurácia de 0.5°C e 2% RH, respectivamente [3].

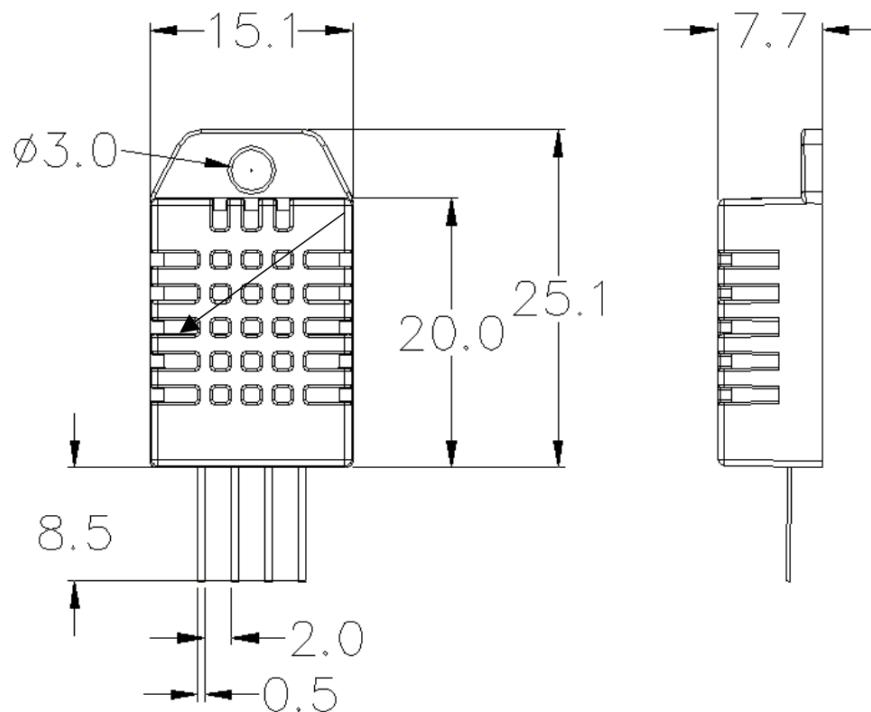


Figura 3 – Dimensões em milímetros do sensor DHT-22 (Retirado de [3]).

## 2.5 Servidor

Um servidor é um sistema de computação centralizada, responsável por fornecer serviços em uma rede de computadores. Esses serviços variam conforme a necessidade

do sistema, podendo ser um controlador de domínio, provedor de arquivos, de impressão, de e-mails entre outros. No modelo cliente/servidor (figura 4), por sua vez, o servidor é responsável por receber as requisições de clientes, provendo informações e dados de acordo com a demanda.

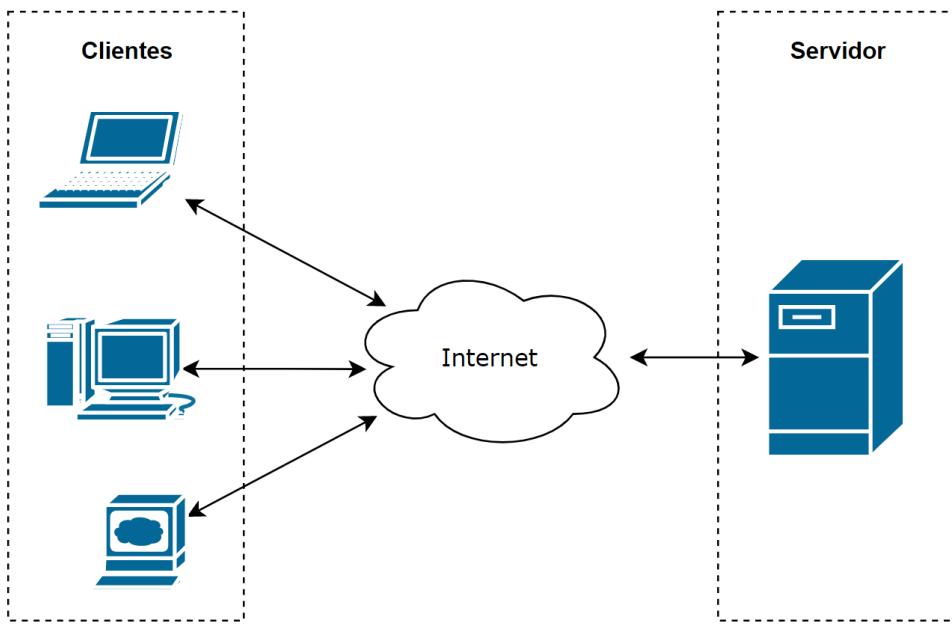


Figura 4 – Representação do modelo cliente/servidor (Autoral).

### 2.5.1 Node.js

Em 2009, Ryan Dahl criou o Node.js ou simplesmente Node, um ambiente de código aberto de execução Javascript para ser utilizado por servidores (*server-side*) baseado no interpretador V8 da Google. O objetivo do Node é fornecer uma forma de criar serviços com alta capacidade de escala, consumindo pouca memória e de fácil aprendizado. Apesar da linguagem de programação Javascript fornecer uma performance inferior às linguagens já existentes para esse objetivo, seu uso oferece dois grande benefícios, ser uma linguagem de fácil aprendizado e de grande uso por ser a linguagem padrão dos navegadores, e consequentemente do desenvolvimento web [18].

A forma de execução no Node é por eventos assíncronos, diferente da maioria dos outros ambientes modernos que utilizam threads do sistema operacional, SO. O Node usa apenas uma thread assíncrona principal que executa operações de entrada e saída de dados, E/S, chamada de *Event Loop* [19]. O *Event Loop* funciona em um ciclo, escutando uma lista de chamas, onde são armazenadas as requisições recebidas, e as direcionando cada uma para uma Thread a parte, que executará a função recebida, como podemos ver na figura 5. A priori, o Node possui quatro Threads trabalhadoras, responsáveis por executar as funções, entretanto, é possível configurar, dependendo da máquina, onde está sendo efetuado a instância do Node.

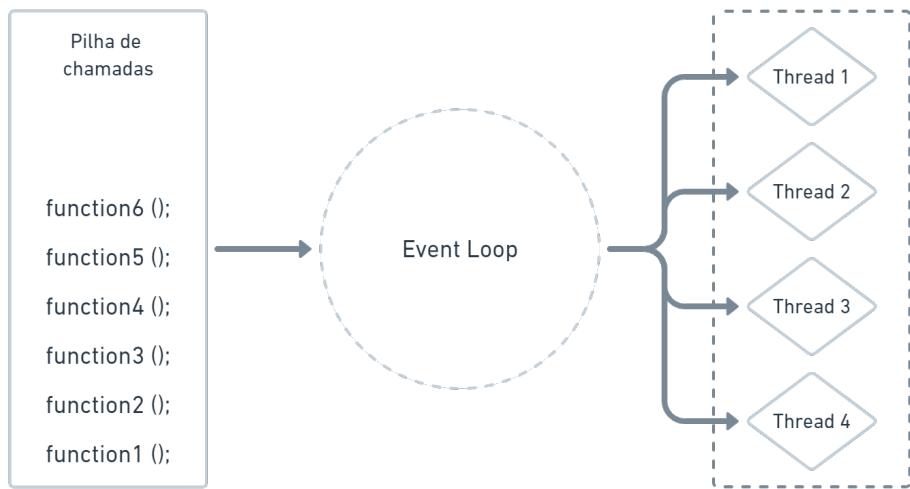


Figura 5 – Ciclo de eventos no Node (Aitoral).

### 2.5.2 InfluxDB

O InfluxDB é um banco de dados que armazena séries temporais (*data series database*), ou seja, sua chave é o tempo e sua forma de armazenagem de dados é em ordem cronológica. Ele foi projetado para lidar com grandes cargas de escrita e consulta, perfeito para armazenar dados em tempo real, como monitoramento de DevOps, métricas de aplicativos, big data e dados de sensores da IoT [20]. De forma geral, séries temporais acabam se tornando gráficos em função do tempo em um determinado período, por exemplo a temperatura de um freezer ao decorrer do dia, podendo assim ver facilmente a máxima, a mínima e suas variações. Esses dados podem ser também coletados e feito uma análise mais complexa usando qualquer ferramenta estatística, dependendo da sua necessidade.

O InfluxDB é composto por *databases* (banacos de dados), *measurements* (medidas), *fields* (campos) e *tags*. Podemos representar essa estrutura como conjuntos como podemos ver na figura 6, no InfluxDB é possível ter inúmeros *databases*, onde cada *database* contém suas *measurements* que são tabelas de dados correspondentes a algum dado em específico. Por exemplo, se tivermos 2 sensores que coletam dados diferentes, cada sensor viraria um *measurement*, e cada *measurement* é composto de dois tipos de atributos, os *fields*, onde ficam os dados da sua medida, e as *tags*, que são campos de dados que diferem *fields* por serem campos indexáveis, feitos exclusivamente para realizar buscas, tal como, é comum adicionar uma *tag* que seja um identificador do dispositivo que coletou esse medida.

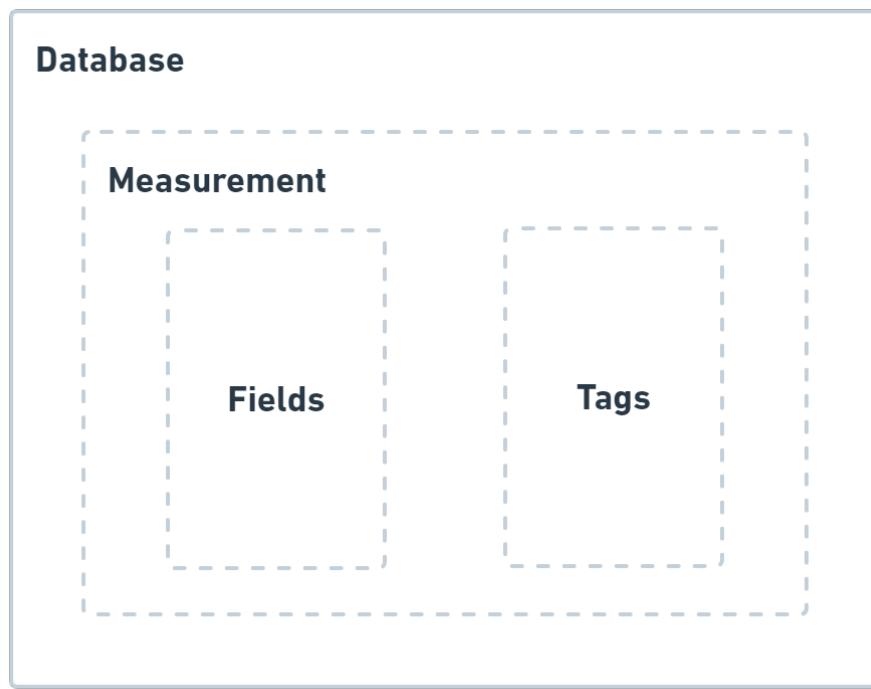


Figura 6 – Composição de um banco de dados do InfluxDB (Autoral).

### 2.5.3 Docker

Docker é uma plataforma *open source*, desenvolvida utilizando a linguagem de programação GO pela Google, com o objetivo de criar facilmente ambientes isolados (containers) com um alto desempenho e portáteis, sendo uma opção em relação às virtualizações. Desta maneira, é possível, por exemplo, criar inúmeras aplicações usando as mesmas tecnologias, cada uma em um *container* diferente e nenhuma vai interferir na outra, e todas na mesma máquina. Se for preciso replicar em outra máquina, é possível criar uma imagem do container e instalar o mesmo ambiente nesta outra máquina.

Em comparação com as virtualizações, os containers não precisam de um sistema operacional, apenas o essencial para executar determinada função. Dessa forma, os containers conseguem ter um controle maior, consomem menos recursos, ganham uma maior flexibilidade e uma manutenibilidade. Podemos ver a comparação entre virtualização e containers na figura abaixo.

Apesar dessa facilidade que containers trazem consigo, gerenciar vários ao mesmo tempo pode ser bem trabalhoso, para isto, foram criados os orquestradores de containers, se baseando em orquestradores de orquestras sinfônicas, que rege o comportamento da sua banda durante uma determinada apresentação. Tais orquestradores auxiliam na organização dos containers, informando que devem iniciar primeiro, quais são as dependências de cada um, interligam suas redes se preciso, entre outras funcionalidades. Para o Docker, um dos

orquestradores mais utilizados é o Docker Compose.

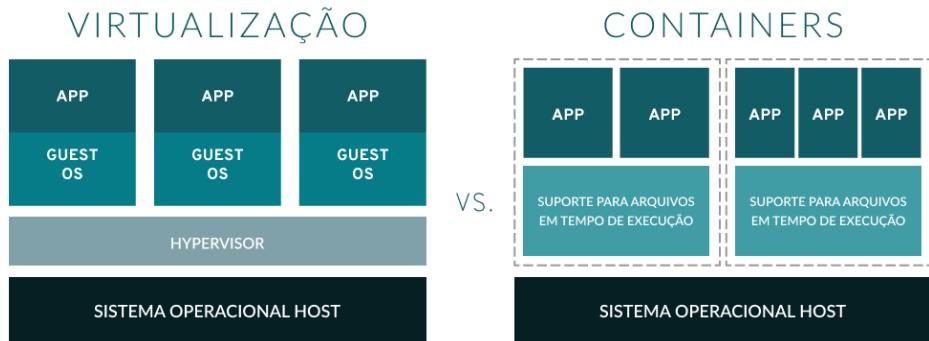


Figura 7 – Comparação entre o modelo de virtualização e modelo de containers (Adaptada de [4]).

#### 2.5.4 Computação em Nuvem

Há uma crescente demanda por serviços de Tecnologia em Informação, TI, oferecidos através da Internet, incluindo armazenamento de dados, hospedagem de sites, softwares, banco de dados, rede entre outros. Este fornecimento de serviços de computação, via Internet, é chamado de computação em nuvem. Tal crescimento vem das diversas vantagens oferecidas pela computação em nuvem, como uma disponibilidade maior, recursos flexíveis, normalmente se pagando apenas o usado, trazendo assim, uma redução no custo, possibilidade de escalonamento de acordo com as necessidades da empresa, não precisar gastar nem se preocupar com a infraestrutura entre outras [21].

### 2.6 Aplicativo Móvel

Aplicativo móvel, APP, é um *software* desenvolvido especificamente para dispositivos móveis, como telefones celulares, tablets, relógios inteligentes entre outros. Cada SO utiliza uma determinada linguagem de programação para desenvolvedores poderem criar os apps. Com uma grande diversidade de sistemas operacionais, com uma linguagem de programação distintas, transformam o desenvolvimento de aplicativos multiplataformas mais trabalhoso e repetitivo, pois tem que ser programada uma versão diferente para cada plataforma. Tendo esse problema em mente, várias formas de programação híbridas foram criadas, com o princípio de poder programar utilizando apenas um código para várias plataformas distintas, dentre elas, as duas que mais se destacam atualmente são o Flutter e o React Native.

### 2.6.1 React Native

O React Native é um *framework* para desenvolvimento nativo voltado para dispositivos para os dois principais SO móveis atualmente, Android e iOS. Ele dispõe de um código aberto, mantido pelo Facebook e sua comunidade. É um *framework* bastante popular, utilizado por grandes empresas globais atualmente, como Netflix, Uber, a própria Facebook e empresas brasileiras como o EBANX e a Globo [22].

O React Native utiliza a linguagem JavaScript como principal e a renderiza para código nativo, para isso é adicionado uma camada em JavaScript, chamada de *Bridge* que se comunica com o sistema operacional, mandando comandos para renderizar os componentes nativos [23].

## 2.7 Trabalhos Relacionados

Estudantes do Instituto Federal da Bahia desenvolveram uma pesquisa [24] sobre diferentes métodos de construção de um sensor de temperatura de baixo custo voltado para o monitoramento de vacinas, pois a temperatura é uma das variantes de grande importância no processo de conservação de materiais imunobiológicos.

No estudo [25], foi desenvolvido um produto de monitoramento de temperaturas de caixas térmicas usadas no transporte de materiais termolábeis, com o objetivo de alertar os transportadores de quando a condição de armazenagem estiver fora da faixa ideal, dispondo de sinais auditivos e visíveis. No seu protótipo foi utilizado uma tela LCD para visualizar a temperatura atual, um led vermelho e outros verde para informar visivelmente sobre a condição do ambiente, um sensor de temperatura DS18B20, para coletar os dados necessários e um Arduíno Uno R3, como controlador dos equipamentos.

Em [26], um estudo similar ao citado posteriormente, foi construído um dispositivo de baixo custo para o monitoramento de temperaturas voltado para geladeiras domésticas que armazenam vacinas em municípios do Brasil de pequeno porte, pois possuem baixa quantidade de insumo que consequentemente, seus ambientes de armazenagem de materiais termolábeis são precários, possuindo poucas câmeras específicas para armazenagem deste tipo de material, que acabam utilizando geladeiras domésticas. Seu protótipo é composto por um Arduino conectado a uma célula de Peltier, refrigerado por ventoinhas acoplado em uma caixa de isopor, os dados da temperatura são coletados pelo Arduino e enviado para um computador onde pode ser visualizado por um web site e armazenado em uma planilha excel.

Na dissertação de Mestrado de Lopes Neto [27], foi desenvolvido uma plataforma em Arduino para transmissão de dados de temperatura do interior dos ambientes gerais de conservação de materiais sensíveis à temperatura, transmitido via SMS e GPRS para

uma central onde é armazenado os dados para consultas posteriores.

Tendo em vista as pesquisas mencionadas acima, venho através deste trabalho, contribuir para uma solução mais completa, de baixo custo e energeticamente eficaz, utilizando tecnologias modernas e escaláveis, com o intuito de auxiliar os profissionais da saúde no controle dos imunobiológicos.

# 3 Metodologia

O sistema proposto neste projeto, se refere a uma plataforma que fornece tanto os dispositivos para a captação dos dados dos imunobiológicos, quanto ao sistema em nuvem para armazenagem dos dados e de um aplicativo móvel para a gerência e análise das informações, seguindo a mesma estrutura representada na figura 1 na sessão 2.2.1, de modo a fornecer aos usuários, um produto de fácil uso para monitorização de temperatura e umidade, possibilitando o histórico dos dados coletados em um determinado cliente.

Podemos então, separar este sistema em quatro partes, cada uma delas com uma determinada função, elas são, end-node, gateway, servidor e aplicativo móvel. Mas antes, para uma visão geral, vamos ver como foi estruturado o envio de pacotes entre cada etapa.

## 3.1 Estrutura dos Pacotes

Definir um padrão de transmissão dos pacotes é de suma importância para facilitar a identificação dos dados e ajuda no reconhecimento de possíveis erros que possam ocorrer na transmissão dos dados.

Dentre a comunicação do gateway, servidor e o aplicativo móvel, que utiliza o protocolo HTTP, existem vários padrões de transmitir dados já definidos e consolidados pela comunidade, um dos mais utilizados é o *JavaScript Object Notation*, JSON, um formato leve e de código aberto, se baseando na estrutura de objetos do JavaScript.

Entretanto, o JSON é considerado uma formato leve para o HTTP e seu ecossistema, para o LoRa, que tem um limite bem menor de dados para ser transmitido, é preciso utilizar outro padrão. Para isto, optamos pelo simples, transmitir os dados separando-os pelo caractere ponto e vírgula, na seguinte ordem: temperatura, umidade, identificador do end node e o contador de pacotes transmitidos.

## 3.2 End Node

O end node é o hardware que fica dentro dos ambientes de armazenagem coletando os dados de temperatura e umidade e enviado para o gateway. Para o melhor entendimento podemos separá-lo em três partes, o sensor, o transceptor LoRa e o microcontrolador, como pode ser visto na figura 8.

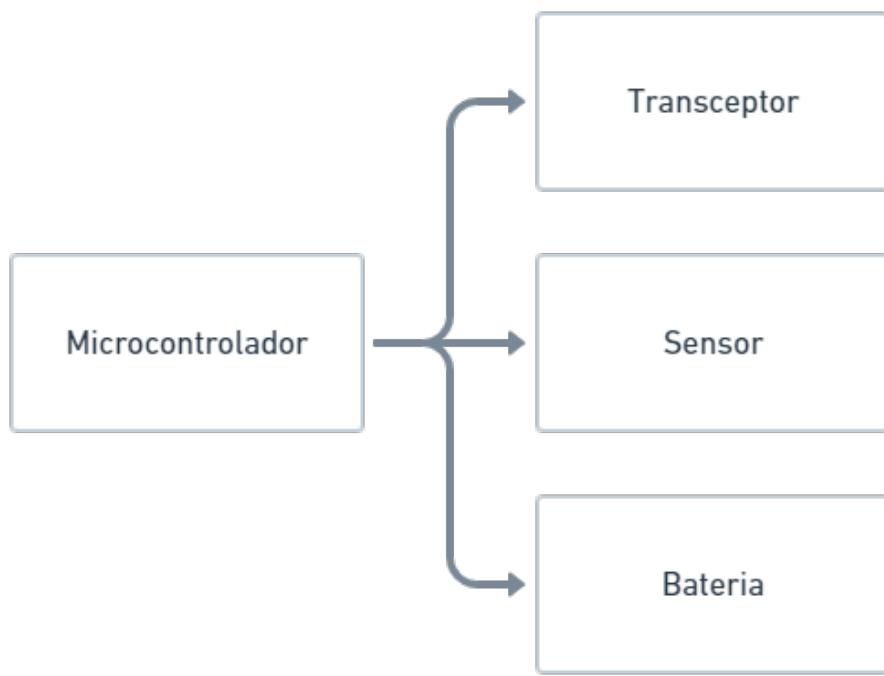


Figura 8 – Representação da conexão entre os componentes principais do end node (autor-al).

Dentre os diversos sensores disponíveis no mercado para monitoramento de temperatura e umidade, optou-se por escolher um sensor da família DHT, pois é bastante difundida na comunidade, tendo uma boa relação custo benefício. Entre esses sensores, aderimos por usar o DHT22, que consegue captar a temperatura e umidade do ambiente dentro das faixas necessárias para o projeto, com uma boa precisão.

Para o microcontrolador, foi escolhido usar um ATmega328, pois é o utilizado na maioria dos Arduinos, o que nos trás dois grandes benefícios, poder utilizar um arduino como protótipo inicial, por ser fácil de encontrar, de programar e para um protótipo secundário, utilizando o microcontrolador sem o Arduino em si, ainda é possível utilizar o mesmo código, bibliotecas e o mesmo Ambiente de desenvolvimento integrado, IDE, utilizado no primeiro protótipo.

Entre os modelos de transceptores LoRa, o eleito foi o RFM95W, por ser o modelo mais simples e com menos custo e por ser o utilizado dentro do *shield* Arduino, um componente modular feito para adicionar novas funcionalidades a um Arduino de forma fácil e prática, facilitando assim, o desenvolvimento do primeiro protótipo.

Em relação à alimentação do protótipo, é um ponto mais complicado de se analisar e ter uma escolha realmente boa para o caso, mesmo fazendo uma análise aprofundada, apenas teremos uma aproximação da sua eficiência, tal processo é melhor detalhado na seção 3.2.2.

### 3.2.1 Primeiro Protótipo

O primeiro protótipo foi feio com o objetivo de validar os usos das tecnologias, seu alcance, e sua competência em relação a interferências ao longo do trajeto. O consumo energético foi deixado de lado neste protótipo, pois foi utilizado o Arduino e ele possui muitos componentes desnecessários para esta aplicação, que acabam aumentando o gasto energético do dispositivo.

Ele coleta os dados utilizando o sensor DHT-22, formata os dados em uma string e o envia através do LoRa, após isto, ele entra em modo de sono profundo (deep sleep) para economizar bateria por um determinado tempo, ao acordar, ele realiza a medição novamente, e fica nesse ciclo.



Figura 9 – Foto do primeiro protótipo (autoral).

### 3.2.2 Segundo Protótipo

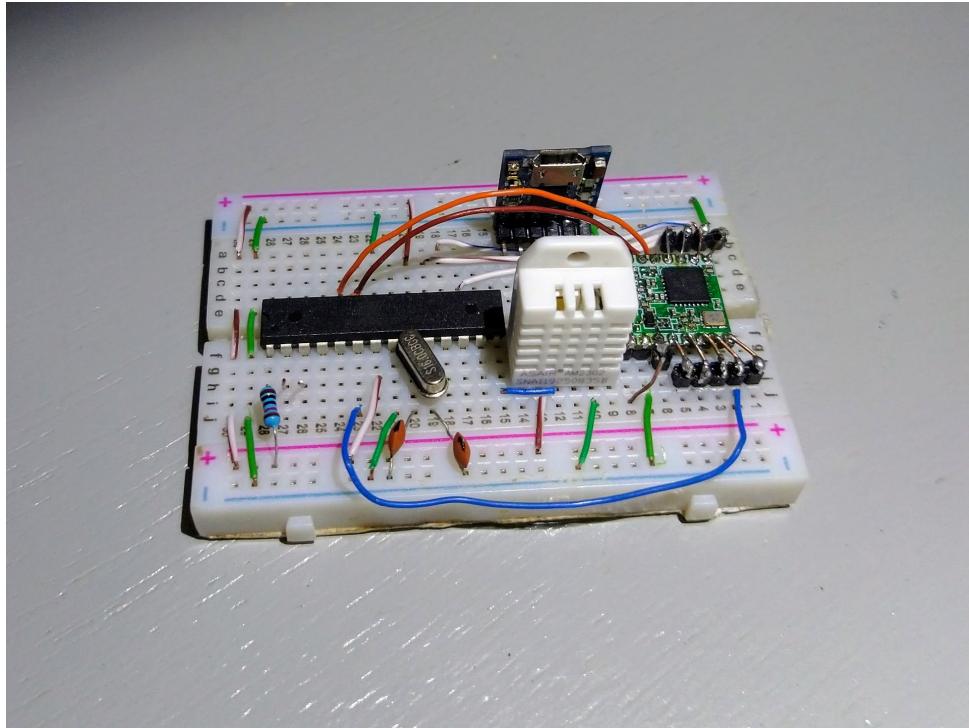


Figura 10 – Foto do segundo protótipo (autoral).

## 3.3 Gateway

O gateway é o dispositivo que fica entre os end nodes e o servidor, e em cada um desses lados, é utilizado uma tecnologia de comunicação diferente, para receber os dados enviados pelos end nodes, se utiliza o LoRa, já para enviar estes dados para o servidor, utiliza-se o HTTP via WiFi. Por isso, é preciso que o gateway possua essas duas tecnologias nele. Felizmente, a Heltec Automation, uma empresa que desenvolve produtos voltada para a IoT utilizando o LoRa, possui um microcontrolador ESP32 que contém tanto o transceptor LoRa, quanto o WiFi, como podemos ver na figura 11. Para programá-lo, foi utilizado a linguagem Arduino, compartilhando assim, a mesma biblioteca entre o end node e o gateway, para o manuseio do transceptor LoRa.

Seu funcionamento segue a seguinte linha, ele fica no aguardo dos pacotes dos end nodes, ao receber um, é realizado uma análise de erro simples, onde é verificado se a formatação do pacote está como esperado, se for detectado algum erro de formatação é enviado para o servidor os valores do RSSI, SNR, o contador do pacote e um atributo booleano marcado como falso, indicando que veio com erro para a seguinte rota, `/packages/:endnodeId`, onde `:endnodeId` é substituído pelo identificador do end node recebido pelo gateway. Se nenhum erro foi detectado, é enviado para a rota do servidor

`/sensors/:endnodeId`, os mesmo dados de RSSI, SNR e contador do pacote com o acréscimo dos dados de temperatura e umidade.



Figura 11 – Foto do ESP32 LoRa da Heltec Automation (autoral).

A configuração do gateway, seu identificador criado pelo servidor e o nome e a senha da rede sem fio que vai ficar conectado, é feita por um ponto de acesso, onde ao liga-lo, ele fornece uma rede sem fio onde é possível acessar seu IP via navegador de Internet, e poder realizar as configurações necessárias.

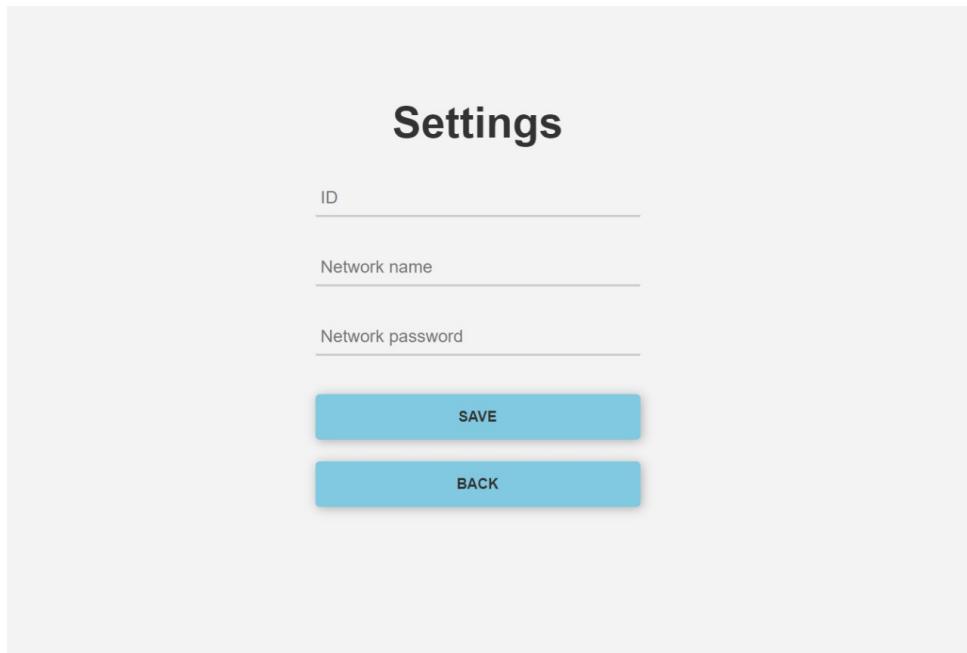


Figura 12 – Página de configuração do gateway (autorral).

## 3.4 Servidor

Para este projeto, o servidor tem as seguintes responsabilidades: armazenar os dados coletados pelos end-nodes, juntamente com os dados do instituto que está usando a aplicação e dos seus respectivos end-nodes e gateways adquiridos; fornecer rotas para troca de informações com o aplicativo e o gateway, para que possam realizar suas devidas funções.

### 3.4.1 Padrões de Projetos

Foi escolhido para desenvolver o servidor, a utilização do Node.js e a linguagem Typescript, se baseando em alguns padrões de softwares, como o Domain-Driven Design (DDD), Test-Driven Development (TDD) e o SOLID visando assim, a construção de um servidor robusto, escalável e de fácil manutenção.

Vale salientar que, esses padrões de softwares, são metodologias de desenvolvimento, e devem ser aplicados de acordo com a demanda da aplicação em construção. Elas são boas práticas, entretanto, dependendo do projeto, alguns pontos podem acabar deixando o processo lento, e dando pouco benefício, e por isso, nesse projeto, não foi seguido fielmente esses padrões, apenas foi baseado.

Antes de tudo, é preciso definir as regras de negócio básicas do servidor, elas servem para garantir que a aplicação atenda as necessidades esperadas. Para este serviço deve ser possível o usuário se cadastrar, para assim realizar seu *login*, e ter acesso às funcionalidades

de cadastrar, editar e remover seus end nodes e seus gateways, além de poder visualizar os dados coletados.

Durante o desenvolvimento de um software, é essencial que a entrega do software funcione corretamente, com qualidade e de acordo com as regras de negócio. Para garantir tais exigências, é importante que se realize testes, tendo em vista identificar possíveis erros antes de chegar aos usuários. Entretanto, realizar testes corretamente é uma tarefa complicada, por isto, existem diversas metodologias, objetivamente facilitar e simplificar os testes dos diferentes componentes de um produto. O TDD é uma dessas metodologias, ela defende o desenvolvimento do teste antes das funcionalidades, garantindo a cobertura completa do código pelos testes.

Junto com o TDD, foi utilizado também o DDD, uma filosofia para auxiliar os desenvolvedores na construção de aplicações complexas de software, ela é referência na organização do código, separando por domínios, de forma isolada. Para poder implementar bem o DDD é preciso definir quais são os domínios da aplicação, analisando as regras de negócio, foi separado em três domínios: end nodes, gateways e usuários.

É preciso também, separar a camada de domínio da aplicação da camada de infraestrutura, a camada de infraestrutura é responsável pelas tecnologias utilizadas para realizar determinada função, por exemplo, a camada de domínio sabe que precisa armazenar os dados dos sensores, mas não é precisar saber onde e nem como, isso é de responsabilidade da camada de infraestrutura.

Para ter uma melhor separação das responsabilidades entre as camadas, é comum utilizar o DDD junto com princípios do SOLID. O SOLID é um acrônimo de 5 princípios da programação orientada a objetos que ajudam o programador a escrever códigos mais limpos, com alta manutenibilidade, separando as responsabilidades e diminuindo acoplamentos.

### 3.4.2 Banco de Dados

É preciso armazenar os dados coletados pelos sensores, para futuras consultas, os dados de transmissão dos pacotes, para utilizar se houver perdas entre as transmissões, e é preciso também, salvar os dados dos usuários e dos seus respectivos produtos cadastrados na plataforma. Tais dados têm suas diferenças, e por isto foi escolhido utilizar banco de dados diferentes para cada um deles.

Começando pelos dados coletados pelos sensores e transmitido pelos end-nodes, esses dados são de medidas, coletados com um período curto, e consequentemente, possuem um grande volume. Por isto, foi escolhido utilizar o InfluxDB para armazenar tais dados.

Foi criado duas *measurements* dentro do banco, uma referente aos dados dos sensores e outra aos dados do pacote transmitido, os dois possuindo a mesma *tag*, o identificador do end node, chamado de *endnodeId*. Dentro da medida Sensor, foi adicionado dois campos

referente a temperatura e umidade do ambiente. Para a medida Package, foi adicionado os campos success, rssi e snr, atributos necessários para saber se o pacote foi entregue com sucesso, a dificuldade de transmitir esse pacote e a relação sinal-ruído. Podemos então ver uma representação do banco na figura 13.

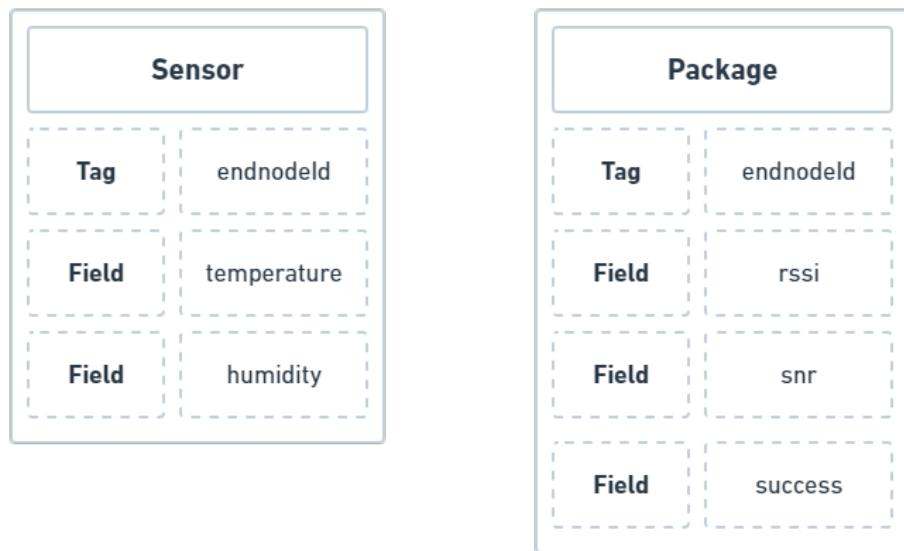


Figura 13 – Medidas dento do banco de dados InfluxDB (autorral).

Diferentes dos dados que vem dos hardwares, as informações dos usuários e seus dispositivos, não possuem essa demanda alta de escrita e leitura, e mais importante, contém relações entre eles, o que torna banco de dados relacionais uma boa opção para armazenar tais dados. Entre tantos bancos de dados relacionais, foi optado por usar o PostgreSQL por motivos de afinidade.

Cada usuário pode possuir inúmeras gateways e end nodes cadastrados, e cada end node precisa ser vinculado com apenas um gateway. Essas relações entre entre as entidades no postgres representada na figura 14, foi pensada olhando tanto para os cenários de pequenos postos de saúde, que tem apenas um local para armazenar seus imunobiológicos, quando para grande hospitais e laboratórios, que pode tem varias salas, com dada sala contendo uma ou mais equipamentos frigoríficos.

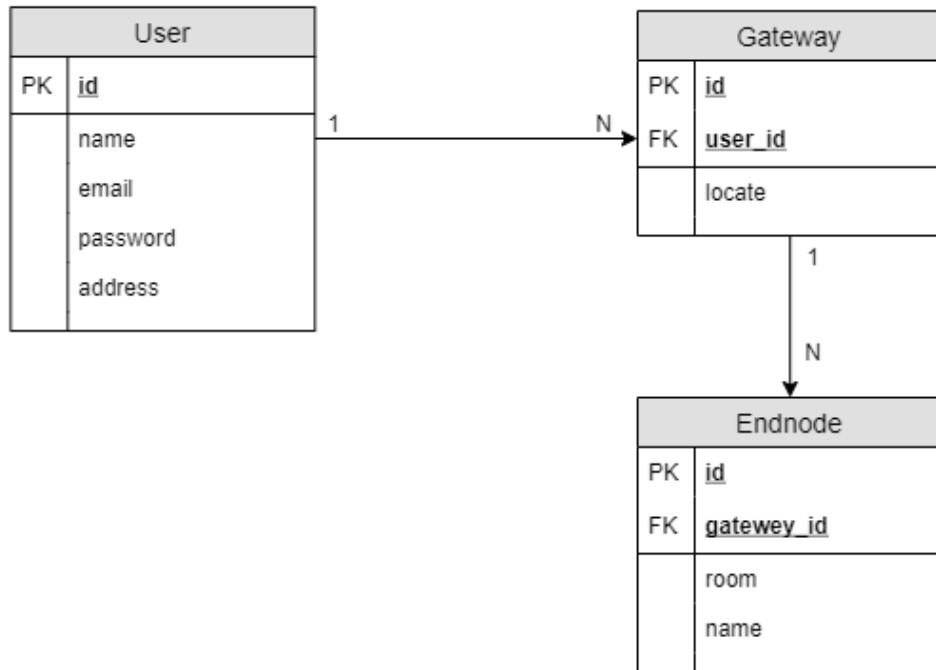


Figura 14 – Modelo entidade relacionamento do PostgreSQL (autoral).

### 3.4.3 Containers

Para facilitar o desenvolvimento do servidor, seu deploy e diminuir possíveis problemas por ter várias aplicações rodando na mesma máquina, foi escolhido por conter tantos os bancos de dados quanto a aplicação em si, utilizando o Docker e para facilitar o gerenciamento dos três containers (PostgreSQL, InfluxDB e a aplicação em Node.js), interligando as redes entre eles, foi utilizado o Docker Compose.

Para os bancos de dados, não foi preciso criar as imagens dos containers, dado que são aplicações muito utilizadas e a própria comunidade já oferece algumas opções dependendo da sua finalidade. Entretanto, para a aplicação em si, foi necessário a criação de uma nova imagem, devido algumas configurações extras exigidas pelas tecnologias utilizadas pela aplicação. Tal imagem foi feita baseando-se na versão alpine do node, uma versão mais leve, com apenas as funcionalidades principais.

### 3.4.4 Deploy

Entre as opções de servidores na nuvem, foi escolhido por usar a Microsoft Azure para realizar o deploy do servidor. Sua escolha se deu pois a Microsoft oferece uma assinatura para estudantes, o que não teve nenhum custo adicional no desenvolvimento deste protótipo e na realização dos testes.

### 3.5 Aplicativo móvel

À necessidade de ter um cliente para que os usuário possa acessar o histórico de medidas e o gerenciar seus dispositivos, tal cliente poderia ser uma página web, entretanto, tendo em vista o crescimento do uso dos smartphones, tivemos como preferência a criação de uma aplicação móvel, mas especificamente para o sistema operacional Android, devido ao sistema iOS da Apple, restringir seu desenvolvimento para aparelhos da marca, incluindo a necessidade de um computador do mesmo para poder emular um dispositivo móvel com seu sistema operacional.

Tendo em mente as regras de negócios destrinchadas na sessão 3.4.1, a primeira coisa feita foi o desenho das telas utilizando a ferramenta figma, específica para tal tarefa. Ao finalizar os desenhos das telas, partimos para a programação.

Dentre as diversas formas de construir um aplicativo deste porte, a ferramenta escolhida para designar esta função foi o React Native, por motivos de utilizar a mesma linguagem de programação do servidor, typescript, e ter a possibilidade de criar para sistemas iOS futuramente usufruindo do mesmo código.

Pensando na construção de uma aplicação com possibilidades de crescimento altas, foi decidido documentar os componentes que compõem o programa, utilizando a biblioteca Storybook.

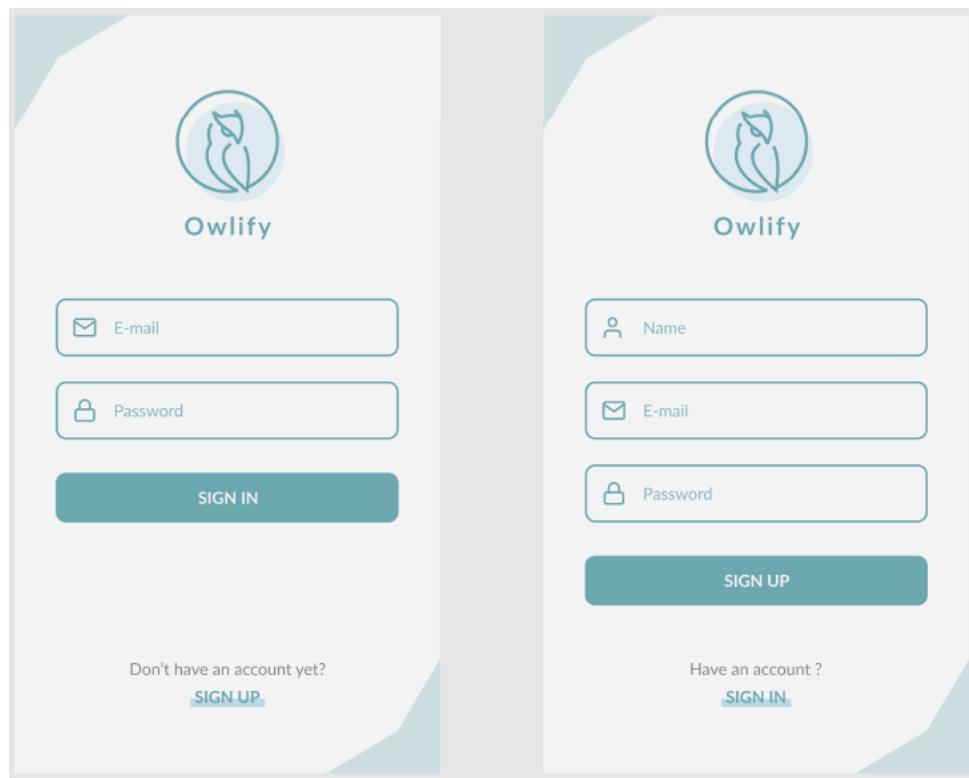


Figura 15 – Desenho das telas de entrar e cadastrar na aplicação (autoral).

## Referências

- 1 3GLTEINFO. *LoRa Architecture*. 2020. Disponível em: <<https://www.3glteinfo.com/lora/lora-architecture>>. Citado 2 vezes nas páginas 4 e 12.
- 2 MOZILLA. *Uma típica sessão HTTP*. 2019. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Session>>. Citado 2 vezes nas páginas 4 e 13.
- 3 ELECTRONICS, A. *DHT22 Datasheet*. 2020. Disponível em: <<https://pdf1.alldatasheet.com/datasheet-pdf/view/1132459/ETC2/DHT22.html>>. Citado 2 vezes nas páginas 4 e 14.
- 4 HAT, R. *Containers x máquinas virtuais*. 2020. Disponível em: <<https://www.redhat.com/pt-br/topics/containers/containers-vs-vms>>. Citado 2 vezes nas páginas 4 e 18.
- 5 FILHO, N. d. A. *O que é saúde?* [S.l.]: Editora Fiocruz, 2011. Citado na página 7.
- 6 MOURA, E. S. D. O direito à saúde na constituição federal de 1988. *Âmbito Jurídico, XVI*, v. 114, 2013. Citado na página 7.
- 7 SOMA, C. *Como funciona o tratamento com imunobiológicos?* 2018. Disponível em: <<https://clinicasoma.com.br/como-funciona-o-tratamento-com-imunobiologicos>>. Citado na página 7.
- 8 SAÚDE, M. da. *Manual de Rede de Frio*. 2001. Disponível em: <[http://bvsms.saude.gov.br/bvs/publicacoes/manual\\_rede\\_frio.pdf](http://bvsms.saude.gov.br/bvs/publicacoes/manual_rede_frio.pdf)>. Citado 2 vezes nas páginas 7 e 8.
- 9 OLIVEIRA, V. C. d. et al. Avaliação da qualidade de conservação de vacinas na atenção primária à saúde. *Ciência & Saúde Coletiva*, SciELO Public Health, v. 19, p. 3889–3898, 2014. Citado 2 vezes nas páginas 7 e 9.
- 10 MUNIZ, É. S. *Memórias da erradicação da varíola*. [S.l.]: SciELO Brasil, 2011. Citado na página 7.
- 11 TEMPORÃO, J. G. O programa nacional de imunizações (pni): origens e desenvolvimento. *História, ciências, saúde-manguinhos*, SciELO Brasil, v. 10, p. 601–617, 2003. Citado 2 vezes nas páginas 7 e 8.
- 12 NIFORATOS, J. D. Common questions about the pfizer-biontech and moderna covid-19 vaccines. Citado na página 8.
- 13 BAE, J. et al. Challenges in equitable covid-19 vaccine distribution: A roadmap for digital technology solutions. Citado na página 9.
- 14 NELSON, C. et al. Monitoring temperatures in the vaccine cold chain in bolivia. *Vaccine*, Elsevier, v. 25, n. 3, p. 433–437, 2007. Citado na página 9.
- 15 FALCÓN, V. C. et al. A vaccine cold chain temperature monitoring study in the united mexican states. *Vaccine*, Elsevier, v. 38, n. 33, p. 5202–5211, 2020. Citado na página 9.

- 16 ALLIANCE, L. *What is LoRaWAN® Specification*. 2020. Disponível em: <<https://lora-alliance.org/about-lorawan>>. Citado na página 12.
- 17 BERNERS-LEE, T.; FIELDING, R.; FRYSTYK, H. *Hypertext transfer protocol–HTTP/1.0*. [S.l.]: May, 1996. Citado na página 12.
- 18 TILKOV, S.; VINOSKI, S. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, IEEE, v. 14, n. 6, p. 80–83, 2010. Citado na página 15.
- 19 FOUNDATION, O. *Documentação oficial do Node.js*. 2019. Disponível em: <<https://nodejs.org/pt-br/about/>>. Citado na página 15.
- 20 GIACOBBE, M. et al. An implementation of influxdb for monitoring and analytics in distributed iot environments. In: SPRINGER. *International conference on the Sciences of Electronics, Technologies of Information and Telecommunications*. [S.l.], 2018. p. 155–162. Citado na página 16.
- 21 SOUSA, F. R.; MOREIRA, L. O.; MACHADO, J. C. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. *II Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)*, p. 150–175, 2009. Citado na página 18.
- 22 BRASIL, R. *Empresas que utilizam React no Brasil*. 2020. Disponível em: <<https://github.com/react-brasil/empresas-que-usam-react-no-brasil>>. Citado na página 19.
- 23 FACEBOOK. *Documentação oficial do React native*. 2020. Disponível em: <<https://reactnative.dev/>>. Citado na página 19.
- 24 CRUZ, R. F.; CORREA, C. S.; SILVA, W. L. Desenvolvimento de um sensor de temperatura de baixo custo aplicado ao controle da qualidade de vacinas. Citado na página 19.
- 25 LIMA, M. S. et al. Controle de temperatura com arduino. *Revista Mythos*, v. 12, n. 2, p. 48–55, 2019. Citado na página 19.
- 26 KERSBAUM, M. et al. Monitoramento de temperatura para câmara de vacina. *Anais da Mostra Nacional de Iniciação Científica e Tecnológica Interdisciplinar (MICTI)-e-ISSN 2316-7165*, v. 1, n. 12, 2019. Citado na página 19.
- 27 NETO, F. D. N. L. Monitoramento remoto de temperatura para a armazenagem de materiais biológicos e vacinas, aplicado em pesquisa clínica. 2019. Citado na página 19.