CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ETEC DA ZONA LESTE

M-Tec Desenvolvimento De Sistemas

Vinicius Augusto Rodrigues da Silva

Vinicius Fernandes de Lima

Jônatas Frinhani de Souza Palmieri

T-Fence: Sistema de Rastreamento de Animais em Áreas Rurais

São Paulo

2025

Vinicius Augusto Rodrigues da Silva
Vinicius Fernandes de Lima
Jônatas Frinhani de Souza Palmieri

T-Fence: Sistema de Rastreamento de Animais em Áreas Rurais

Trabalho de Conclusão de Curso apresentado ao M-Tec Desenvolvimento de Sistemas da Etec Zona Leste, orientado pelo Prof. Esp. Jeferson Roberto de Lima, como requisito parcial para a obtenção do título de técnico em Desenvolvimento de Sistemas

São Paulo
2025

T-Fence

Sistema de Rastreamento de Animais em Áreas Rurais.

Vinicius Augusto Rodrigues da Silva
Vinicius Fernandes de Lima
Jônatas Frinhani de Souza Palmieri

Aprovada em __/__/____.

BANCA EXAMINADORA

Prof. Esp. Jeferson Roberto de Lima

Universidade do Jeferson

Prof. (Professor avaliador)

Universidade do Avaliador

Prof. (Professor avaliador)

Universidade do Avaliador

Dedico
XXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXX

AGRADECIMENTOS

Agradeço a xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX."

Autor

RESUMO

xxxxxxxxxxxxxxxx; xxxxxxxxxxxxxxxx; xxxxxxxxxxxxx; xxxxxxxxxxxx; xxxxxxxxxxxxxx.

Palavras-Chave: xxxxxxxxxxxxxxxx; xxxxxxxxxxxxxxxx; xxxxxxxxxxxxx; xxxxxxxxxxxx; xxxxxxxxxxxxxx.

ABSTRACT

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Keywords: xxxxxxxxxxxxxxxxx; xxxxxxxxxxxxxxxxx; xxxxxxxxxxxxxx; xxxxxxxxxxxxx; xxxxxxxxxxxxxx.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Cascading Style Sheet (CSS)

HyperText Markup Language (HTML)

Internet of Things (IoT)

Unified Modeling Language (UML)

LISTA DE SÍMBOLOS

SUMÁRIO

1 Introdução

A T-Fence é um sistema de monitoramento de animais voltado para áreas rurais, que busca auxiliar o pecuarista a gerir seu rebanho com mais eficiência e segurança. O sistema é composto por uma coleira IoT que envia a localização do animal para uma aplicação mobile que possibilita o pecuarista acompanhar e controlar seu rebanho de forma prática e precisa.

O objetivo geral desse projeto é desenvolver um sistema de rastreamento para monitoramento de gado em áreas rurais auxiliando pecuaristas na gestão de seus animais, proporcionando mais eficiência, controle e segurança. Isso contribui para a modernização e inovação tecnológica no setor agropecuário.

A agricultura tem uma forte relevância quando se trata sobre o crescimento econômico do país. O Centro de Estudos Avançados em Economia Aplicada (Cepea, 2025), abordou que em 2025, o agronegócio pode representar cerca de 29,4% do PIB nacional. Ainda assim, o setor agropecuário sofre com a alta taxa de criminalidade na região, o que afeta negativamente a produção agrícola e pecuária. Isso evidencia a necessidade de investimento tecnológico no setor, tendo em vista a sua importância no panorama nacional. Nos últimos anos, furtos e perdas de animais têm gerado grandes prejuízos no meio rural, afetando a produtividade e rentabilidade do agronegócio no Brasil. Segundo os dados divulgados pela Polícia Civil de Santa Catarina por meio do Centro de Apoio Operacional de Combate aos Crimes Contra o Agronegócio (CAOAGRO, 2024) foram registrados 327 Boletins de ocorrência sobre crimes contra o agronegócio, sendo que o crime de abigeato(furto de bovinos) representou 28,8% de todos os boletins de ocorrências, totalizando 273 bovinos furtados, um aumento de 13,7% comparado ao primeiro quadrimestre de 2023. Esses dados, apresentam a dificuldade que os pecuaristas enfrentam diariamente, em relação a gestão de suas propriedades e de seus animais. Esses crimes podem acabar desestimulando a produção rural, já que geram grandes prejuízos ao produtor rural.

De início, acreditava-se que o problema principal era a ausência de ação do poder público, em tomar medidas contra esse tipo de criminalidade. Outra hipótese

levantada era a negligência dos produtores rurais, aliado a limitação da infraestrutura nesses espaços, justificando as tais ocorrências.

Os principais objetivos foram desenvolver um dispositivo capaz de capturar a sua localização GPS e transmitir sua posição utilizando o LoRa para uma central, projetar um sistema com baixo consumo de energia e de longa duração capaz de se auto alimentar utilizando tecnologia fotovoltaica, utilizar a tecnologia de Geofencing para que seja possível a criação de um perímetro virtual, desenvolver um sistema capaz de identificar se o animal saiu do perímetro delimitado e gerar um alerta, integrar o dispositivo com uma aplicação para exibição das informações por meio de um mapa.

Em razão da grande extensão territorial das propriedades rurais, além da pobre infraestrutura tecnológica, o monitoramento dos animais se torna difícil, prejudicando a resolução desta situação. Com isso, foi pensada uma solução energeticamente independente e adaptável em cenários extremos, como áreas extensas e remotas, com pouca disponibilidade de energia elétrica e internet escassa. Com isso, a proposta consiste no desenvolvimento de um sistema de monitoramento via GPS, capaz de aumentar a segurança e otimizar a gestão do gado.

Nesta pesquisa serão utilizados os principais autores e fontes relevantes para o trabalho. Para trazer dados sobre a relevância da agropecuária no país e sua importância na economia nacional, foram consultados MAPA (2023) e CEPEA (2025). Já para o desenvolvimento do nosso IoT, utilizamos os seguintes autores: MAGRANI, Eduardo. Para o desenvolvimento do nosso aplicativo, utilizamos ESCUDELARIO, Bruna; PINHO, Diego (2021) e FALCÃO, Filipe Dourado (2022), bem como a documentação oficial do React Native (REACT NATIVE, 2025).

Este documento se divide em 4 capítulos que auxiliam na compreensão do desenvolvimento do projeto: no capítulo 1 é abordado uma introdução sobre o tema, no capítulo 2 é apresentado o referencial teórico com as ferramentas e tecnologias utilizadas, no capítulo 3 é descrito o desenvolvimento do trabalho e por fim as considerações finais sobre o projeto está contido no capítulo 4.

2 Referencial Teórico

Neste capítulo, serão apresentados os principais fundamentos teóricos e as tecnologias utilizadas no desenvolvimento do nosso projeto.

2.1 Segurança e informatização nos campos rurais

O setor agrário tem uma grande importância na economia brasileira, possuindo um papel fundamental no desenvolvimento do país. O Ministério da Agricultura e Pecuária (MAPA), afirma que em 2023, a agropecuária cresceu 15,1%, refletindo diretamente no desenvolvimento do Produto Interno Bruto (PIB) do país, aumentando 2,9% em relação a 2022. Entretanto, mesmo sendo um dos setores fundamentais para a economia do Brasil, a agropecuária enfrenta grandes problemas em relação à segurança de suas propriedades, o que pode facilitar a ocorrência de crimes contra o setor.

Segundo Oliveira, Medina e Teixeira (2021), cerca de 90% dos crimes cometidos em áreas rurais de Goiás nos anos de 2017 e 2018 foram classificados como furtos, reconhecidos juridicamente como abigeatos, tendo como um dos principais alvos os animais bovinos. De acordo com a Confederação da Agricultura e Pecuária do Brasil (2017), os crimes que ocorrem nas propriedades rurais, muitas vezes são enquadrados como conflitos agrários, apresentando os produtores rurais como causadores dos problemas e não como vítimas dessa situação. Levando em consideração, a importância do setor no cenário nacional e as ocorrências apresentadas anteriormente, esses fatos podem acabar desestimulando a população agrícola em suas atividades, impactando diretamente na economia nacional.

Outro fator que deve ser levado em consideração é a dificuldade na aplicação de soluções tecnológicas que trazem a informatização para o campo, tendo em vista a infraestrutura pobre da região. Hoje em dia o cenário rural demonstra uma demanda por soluções inovadoras, tendo em vista essa necessidade, a Embrapa (2021), apontou que um proprietário rural de Minas Gerais, gasta R$78 mil por ano para realizar a contagem dos animais em seu rebanho de forma manual. Isso nos mostra a necessidade de soluções mais eficientes e rentáveis.

Em resumo, mesmo sendo fundamental para a economia do país, o setor agropecuário continua enfrentando problemas relacionados à falta de segurança e à

escassez de tecnologias, o que pode dificultar o processo produtivo. A implementação de novas tecnologias no setor primário, adequadas à realidade do meio rural, pode melhorar a eficiência e a qualidade da produção, além de impulsionar a economia agropecuária e a economia nacional como um todo.

2.2 Tecnologias Utilizadas

A seguir veremos as tecnologias envolvidas no desenvolvimento e prototipagem do nosso trabalho.

2.2.1 HTML

Segundo o que diz Maurício Samy Silva (2011), o HTML, sigla que significa HyperText Markup Language, é uma linguagem de marcação onde o programador cria marcações para se comunicar com o navegador, que as interpreta e apresenta ao usuário em forma de página.

A estrutura do HTML é como uma árvore, tendo dessa forma, elementos filhos de outros elementos (FERREIRA, 2012). Logo abaixo está um exemplo da estrutura simples HTML:

Figura 1 - Estrutura simples feita em HTML

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>

</body>
</html>
```

Fonte De Autoria Própria, 2025.

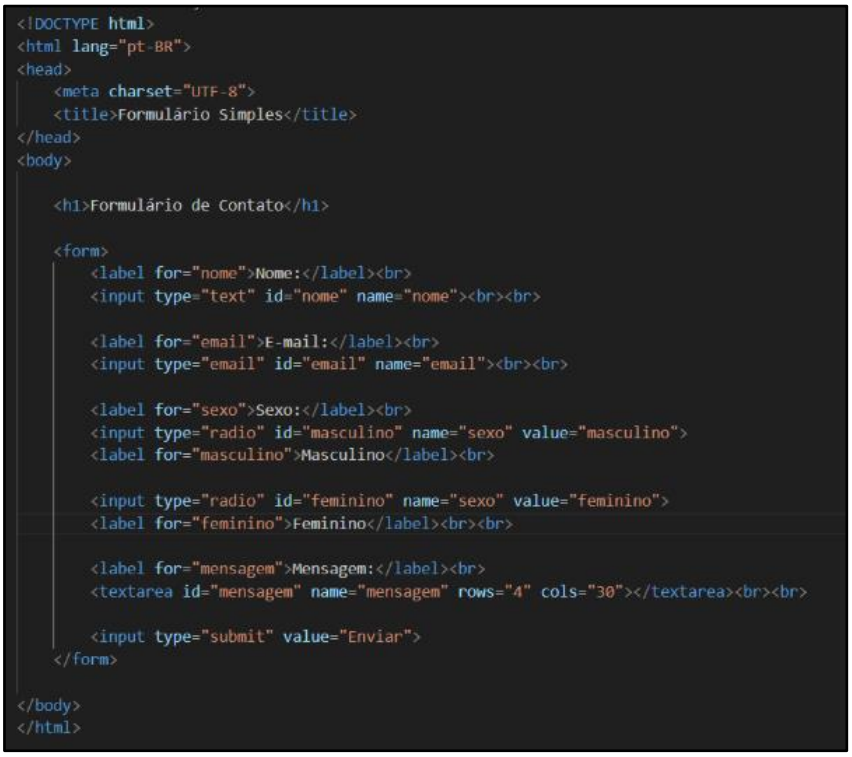
Para melhor entendimento das tags presentes nessa estrutura, nós as explicaremos:

- DOCTYPE: “<!DOCTYPE html>” mostra para o navegador o tipo de documento e a versão em HTML.

- Html: “<html lang="pt-BR"></html>” é a tag responsável por definir onde começa e termina o documento HTML, enquanto o atributo “Lang” indica o idioma da página.
- Head: “<head></head>” essa tag delimita o cabeçalho, nela são colocados conteúdos de configuração, links de estilos ou scripts e o título da página.
- Metatag Charset: “<meta charset="UTF-8">” Responsável pela codificação dos caracteres, garantindo que acentos e símbolos especiais apareçam da forma correta na página.
- Title: “<title></title>” define o título da página; não é exibido na página, mas aparece na aba do navegador.
- Body: “<body></body>” é o corpo da página, nela é colocado todo o conteúdo no qual o usuário pode interagir.

Serão adicionadas algumas tags para a criação de um formulário simples. As tags adicionadas serão explicadas abaixo do exemplo:

Figura 2 - Exemplo de Formulário usando HTML

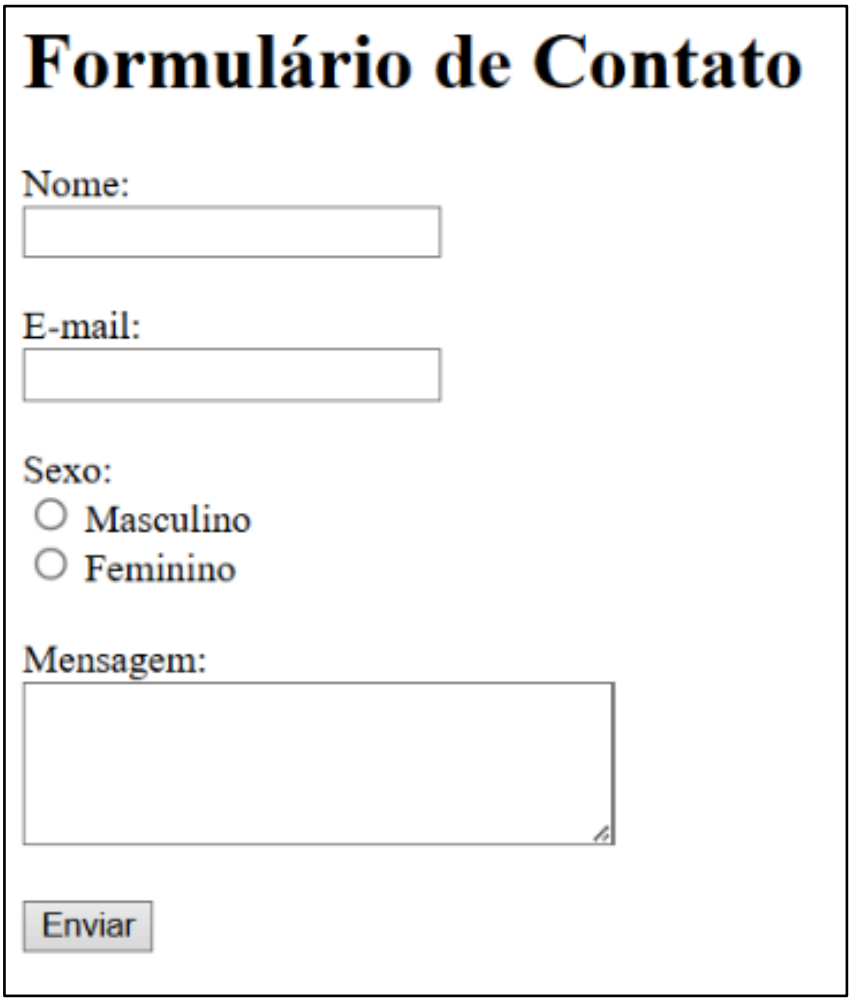


```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Formulário Simples</title>
</head>
<body>

    <h1>Formulário de Contato</h1>

    <form>
        <label for="nome">Nome:</label><br>
        <input type="text" id="nome" name="nome"><br><br>

        <label for="email">E-mail:</label><br>
        <input type="email" id="email" name="email"><br><br>

        <label for="sexo">Sexo:</label><br>
        <input type="radio" id="masculino" name="sexo" value="masculino">
        <label for="masculino">Masculino</label><br>

        <input type="radio" id="feminino" name="sexo" value="feminino">
        <label for="feminino">Feminino</label><br><br>

        <label for="mensagem">Mensagem:</label><br>
        <textarea id="mensagem" name="mensagem" rows="4" cols="30"></textarea><br><br>

        <input type="submit" value="Enviar">
    </form>

</body>
</html>
```

Fonte: De Autoria Própria, 2025.

- H1: “<h1></h1>” é uma forma de definir o tamanho do texto, geralmente é usado para como título da página por ser maior visualmente.
- Form: “<form></form>” responsável pelo formulário da página, essa tag organiza visualmente os campos.
- Label: “<label></label>” é usada para mostrar os campos de texto no formulário de uma página.
- Br: “<br>” é responsável, basicamente, pela quebra de linha de um texto. Essa tag não precisa de fechamento.
- Input: “<input>” cria os campos dos formulários, não precisa de fechamento e pode mudar seu comportamento de acordo com seu atributo.

Figura 3 - Exemplo do Formulário no navegador



Fonte: De Autoria Própria, 2025.

2.2.2 CSS

Conforme Maurício Samy Silva (2015), CSS é a sigla para Cascading Style Sheets, que em português significa folha de estilo em cascata. Sua finalidade é formatar para o navegador do usuário as informações dos elementos que o HTML fornece, sendo essa sua principal função (FERREIRA, 2012).

O CSS tem a sua estrutura mais simples dividida em duas partes: o seletor e a declaração, sendo o seletor usado para apresentar o elemento alvo e declaração estabelecer os parâmetros de estilização (SILVA, 2011).

Figura 4 - Exemplo de código CSS

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    padding: 20px;
}

h1 {
    color: #333;
    text-align: center;
}

form {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    max-width: 400px;
    margin: 0 auto;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

label {
    font-weight: bold;
}

input[type="text"],
input[type="email"],
textarea {
    width: 100%;
    padding: 8px;
    margin-top: 4px;
    margin-bottom: 12px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type="radio"] {
    margin-right: 6px;
}

input[type="submit"] {
    background-color: #4CAF50;
    color: white;
    padding: 10px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    width: 100%;
}

input[type="submit"]:hover {
    background-color: #45a049;
}
```

Fonte: De Autoria Própria, 2025.

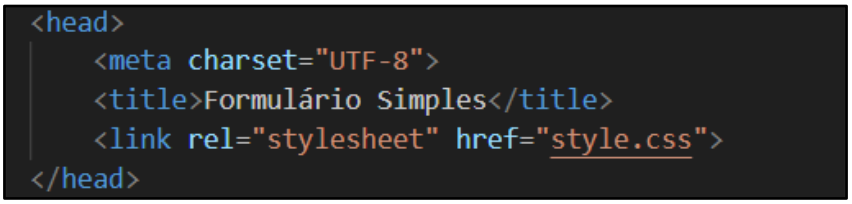
Para auxiliar na compreensão do código, segue a explicação das propriedades presentes na figura:

- *: É o seletor universal, seleciona todos os elementos da página.
- margin: Adiciona um espaçamento fora da borda entre o elemento alvo dos outros.
- padding: Adiciona um espaçamento dentro da borda, separando-a do conteúdo.
- body: Usado para selecionar o corpo da página.
- font-family: Define a fonte para o texto.

- background-color: Determina a cor de fundo do elemento.
- color: Usada para determinar a cor do conteúdo dentro do elemento.
- text-align: Posiciona o texto de forma horizontal dentro o elemento.
- border-radius: Arredonda as bordas do elemento na página.
- max-width: Limita a largura máxima que o elemento pode ter.
- box-shadow: Adiciona uma sombra em volta do elemento na página.
- font-weight: Determina a espessura do texto no elemento.
- width: Define o tamanho do elemento.
- input[type="submit"]:hover: Muda o estado do elemento quando o cursor passa por cima.
- margin-top: Define o espaçamento na parte de cima do elemento na página.
- margin-bottom: Define o espaçamento na parte inferior do elemento na página.
- border: É a borda do elemento.
- Margin-right: Estabelece o espaçamento direito do elemento.
- cursor: Muda o formato do cursor do mouse ao passar por cima do elemento na página.

É necessário importar o código CSS para o arquivo HTML através da tag <LINK>, e somente assim a codificação funcionará corretamente. (MCFARLAND, 2010).

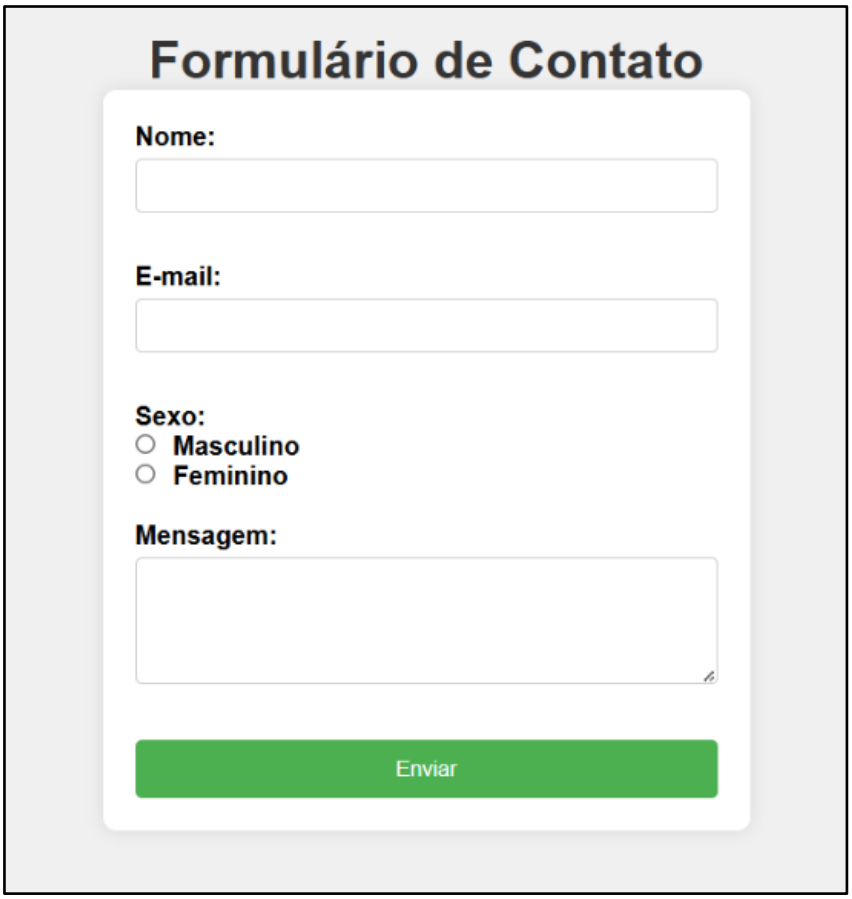
Figura 5 - Importação do CSS no HTML



Fonte: De autoria própria, 2025.

Assim, com o código CSS junto do HTML formam um formulário simples e estilizado. Segue abaixo a figura:

Figura 6 - Resultado da Codificação CSS



Fonte: De Autoria Própria, 2025.

2.2.2.1 JavaScript

Criado pela Netscape e licenciado pela atual Oracle, o JavaScript, além de uma marca, é uma linguagem web que tem como principal função especificar o comportamento das páginas (FLAGANAN, 2013). É considerada uma das linguagens mais usadas no mundo inteiro, pois pode ser usada no front-end e no back-end (GRONER, 2018). De acordo com Maurício Samy Silva (2010), o Javascript é capaz de acessar campos em um formulário feito em HTML e validá-los. Veja abaixo um exemplo simples de validação com a explicação do código:

Figura 7 - Exemplo de código Javascript

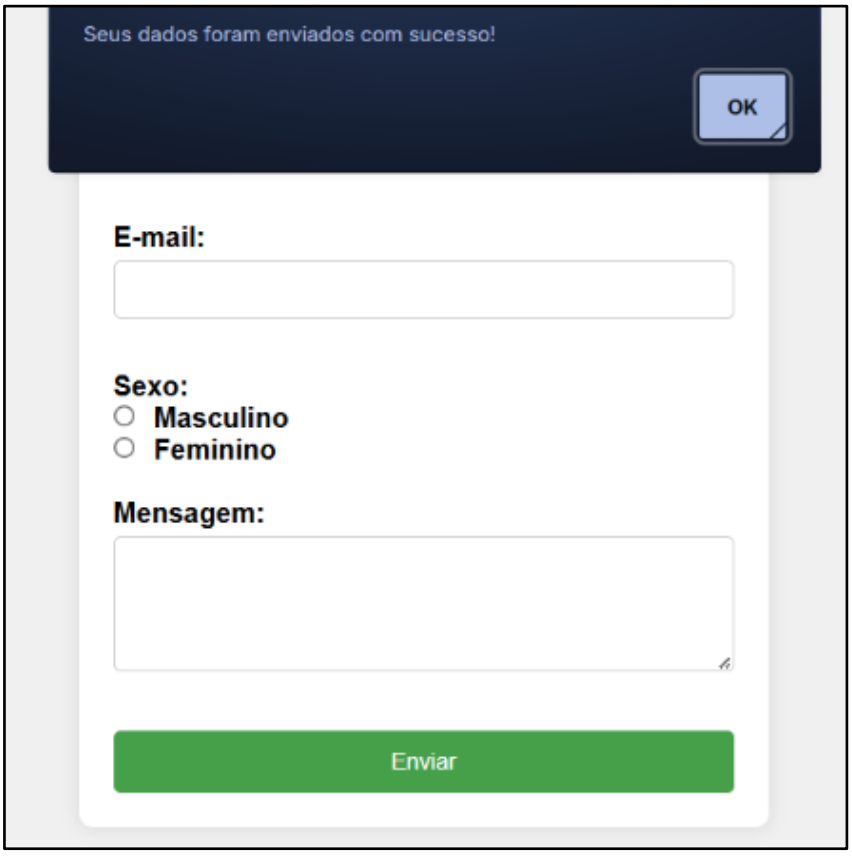
```javascript
function enviarForm() {
    alert("Seus dados foram enviados com sucesso!");
    document.getElementById("meuFormulario").submit();
}
```

Fonte: De Autoria Própria, 2025.

- Function: “function enviarForm()” é usado para chamar a função enviarForm quando for preciso.
- Alert: “alert()” Exibe uma caixa de mensagem anunciando que os dados foram salvos com sucesso.
- Document: “Document.getElementById()” Procura o formulário com o ID meuForm no código HTML e envia quando o usuário clica no botão.

O Resultado ficará assim:

Figura 8 - Resultado da codificação JavaScript



Fonte: De Autoria Própria, 2025.

Para que tudo funcione da forma certa, é necessário importar o Javascript no arquivo HTML. (ZAKAS, 2010).

Figura 9 - Importação do Javascript no HTML

```
<script src="script.js"></script>
```

Fonte: De Autoria Própria, 2025.

2.2.3 React Native

Segundo ESCUDELARIO e PINHO (2021), o React Native é um framework, baseada no React e criada por engenheiros do Facebook, no qual busca trazer modernidade aos aplicativos mobiles, utilizando as melhores ferramentas do mercado para o desenvolvimento front-end (HTML, CSS e JS).

O React Native permite a criação de aplicativos nativos, sem a quebra de experiência dos seus usuários, trazendo a programação em React, para plataformas como Android e iOS (React Native).

Como observa ESCUDELARIO e PINHO (2021), a capacidade de transformar código JavaScript em código nativo das plataformas, é o principal atrativo do React Native. Isso facilita no desenvolvimento, já que uma única aplicação pode ser utilizada em ambas as plataformas, reduzindo custos e oferecendo uma melhor experiência.

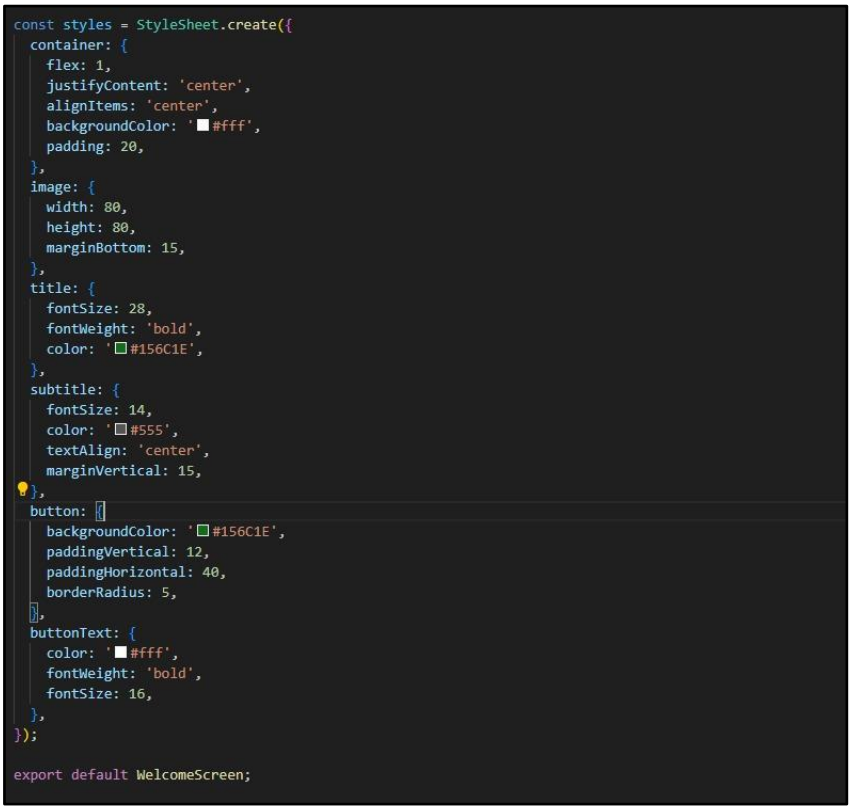
Para maior clareza, segue abaixo um exemplo do código com as funcionalidades explicadas:

Figura 10 – Exemplo de código React Native

```
import React from 'react';
import { View, Text, TouchableOpacity, StyleSheet, Image } from 'react-native';

const WelcomeScreen = () => {
  return (
    <View style={styles.container}>
      <Image
        source={require('../../assets/cadeado.png')}
        style={styles.image}
      />
      <Text style={styles.title}>Bem-vindo!</Text>
      <Text style={styles.subtitle}>
        Comece sua jornada na fazenda digital com segurança e praticidade.
      </Text>
      <TouchableOpacity style={styles.button}>
        <Text style={styles.buttonText}>Começar</Text>
      </TouchableOpacity>
    </View>
  );
};
```

Fonte: De Autoria Própria, 2025.

Figura 11 – Exemplo de código de estilização com React Native



```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#fff',
    padding: 20,
  },
  image: {
    width: 80,
    height: 80,
    marginBottom: 15,
  },
  title: {
    fontSize: 28,
    fontWeight: 'bold',
    color: '#156C1E',
  },
  subtitle: {
    fontSize: 14,
    color: '#555',
    textAlign: 'center',
    marginVertical: 15,
  },
  button: {
    backgroundColor: '#156C1E',
    paddingVertical: 12,
    paddingHorizontal: 40,
    borderRadius: 5,
  },
  buttonText: {
    color: '#fff',
    fontWeight: 'bold',
    fontSize: 16,
  },
});

export default WelcomeScreen;
```

Fonte: De Autoria Própria, 2025.

- Const: “const WelcomeScreen = () => {}” cria um componente funcional e o coloca em uma constante.
- Return: “return ()” é usado para devolver o estilo do componente exibido na tela.
- Import: “import {} from 'react-native'” usado para inserir os componentes do React Native usados para montar a tela.-
- View: “<View style={}>” tem como função criar um contêiner para auxiliar na organização dos elementos na tela.
- Image: “<Image source={require('')} style={} />” usado para mostrar a imagem definida pelo caminho.
- Text: “<Text style={}></Text>” é usado para inserir um texto na tela.
- TouchableOpacity: “<TouchableOpacity style={}>” tem a função de acionar o botão quando clicado.
- StyleSheet.create(): “const styles = StyleSheet.create({})” é usado para definir os estilos, como o tamanho e cores.

- Export default: “export default WelcomeScreen” faz o componente ser importado na tela.

O código acima ficará da seguinte forma:

Figura 12 - Exemplo de aplicação React Native

Fonte: De Autoria Própria, 2025.

2.2.3.1 EXPO

Expo é um framework de código aberto utilizado para facilitar o desenvolvimento de aplicações multiplataforma com React Native, oferecendo algumas funcionalidades como o roteamento baseado em arquivos e biblioteca padrão de módulos nativos (Expo).

Para ESCUDELARIO e PINHO (2021), o Expo é um conjunto de ferramentas que fornece todo o ambiente necessário para desenvolver as aplicações, possibilitando criar, testar e rodar o código de uma maneira fácil e simples.

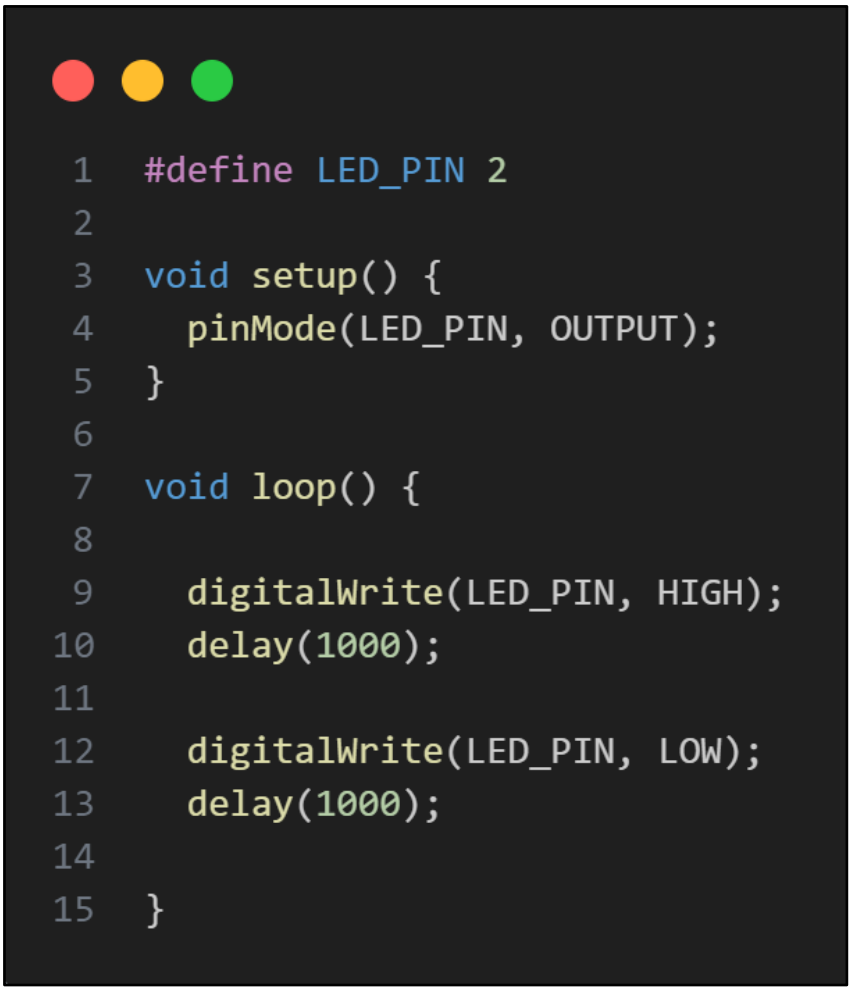
De acordo com Falcão (2022), o Expo busca diminuir a complexidade do código, simplificando o processo de construção dos aplicativos, ocultando etapas mais complexas, permitindo assim que o desenvolvedor foque apenas no código JavaScript do ambiente de desenvolvimento.

2.2.4 C++

A linguagem de programação C++ foi criada na década de 1980 pelo Bjarne Stroustrup, nos laboratórios da Bell Labs, com o objetivo de ampliar recursos da linguagem C, adicionando conceitos de programação orientada a objetos (DEITEL; DEITEL, 2005).

De acordo com Villas-Boas (2001), o C++ é uma linguagem muito versátil que pode ser utilizada para desenvolvimento em diversos ambientes e plataformas, desde sistemas operacionais de baixo nível até sistemas embarcados.

Figura 13 - Exemplo de C++ utilizado no ESP32



```cpp
#define LED_PIN 2

void setup() {
  pinMode(LED_PIN, OUTPUT);
}

void loop() {

  digitalWrite(LED_PIN, HIGH);
  delay(1000);

  digitalWrite(LED_PIN, LOW);
  delay(1000);

}
```

Fonte: De Autoria Própria, 2025.

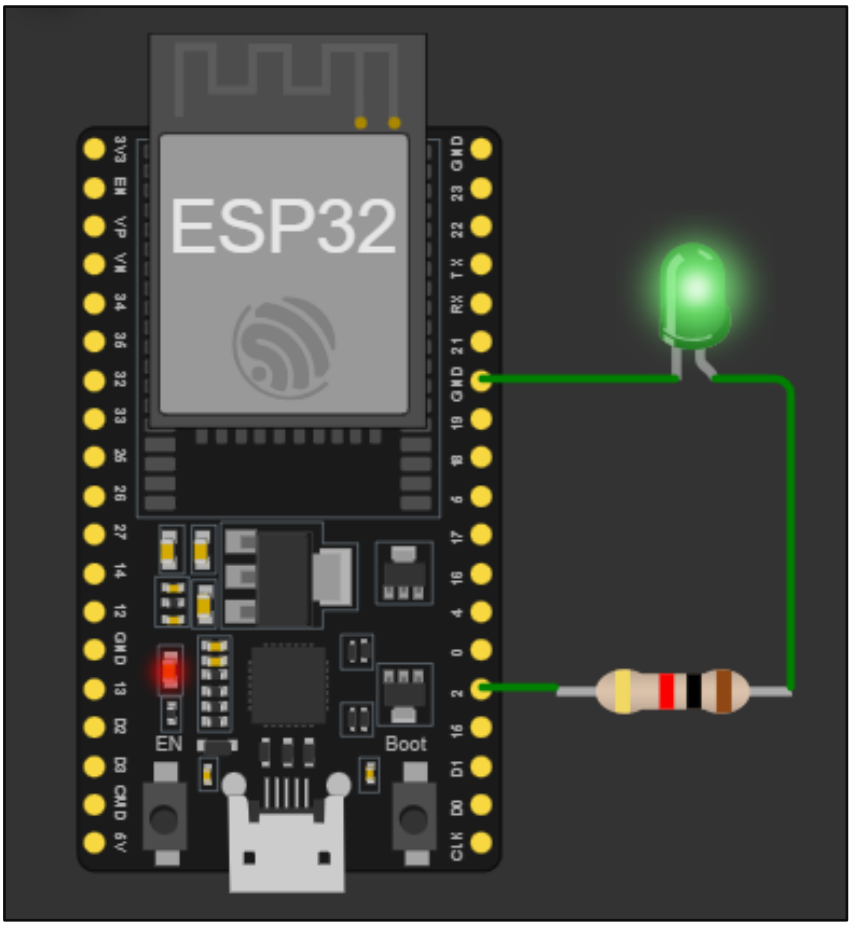
Explicação do código:

- #define LED_PIN 2: É uma diretiva do pré-processador em C/C++ ela cria um nome simbólico para um valor constante. Antes do código ser enviado para o ESP32, o pré-processador troca todas as aparições de LED_PIN por 2.
- void setup(): É uma função que é executada uma única vez quando o ESP32 ligar ou reiniciar, utilizada para definir modo de um pino, iniciar sensores, displays etc.
- void loop(): É uma função que sempre se repete, tudo que estiver dentro desta função roda em ciclo infinito enquanto o ESP32 estiver ligado.
- digitalWrite(): É uma função que envia um valor alto ou baixo para um pino digital.

- digitalWrite(LED_PIN, HIGH): LED_PIN é o pino 2 (Definido com #define), HIGH manda o valor alto (3.3V), ou seja, acende o LED.
- delay(1000): É uma função que pausa o programa por um tempo, neste caso em 1000 milissegundos que é equivalente à 1 segundo, serve para deixar o LED aceso por 1 segundo.
- digitalWrite(LED_PIN, LOW): LOW manda um valor baixo (0V) para o pino 2, desligando o LED.
- delay(1000): Espera mais 1 segundo com o LED apagado.
- O processo se repete enquanto o ESP estiver ligado.

Abaixo uma imagem que demonstra o código em execução

Figura 14 - Circuito ESP32 com LED



Fonte: De Autoria Própria, 2025.

2.2.5 Node.js e NPM

Lançado no final de 2009 e criado pelo Ryan Dahl e sua equipe, o Node.js surgiu devido a ineficiência da arquitetura clássica, que gastava grande parte do tempo em uma fila ociosa enquanto era usado para executar um I/O (PEREIRA, 2014).
Segundo Aguiar (2015), o Node.js é uma plataforma não bloqueante orientada a eventos, usada para desenvolver aplicações de redes escaláveis, resolvendo o problema da antiga arquitetura com o grande volume de dados.
O NPM é um gerenciador de pacotes que tem como função principal a instalação de pacotes, e já vem instalado junto do node (BROWN, 2020).

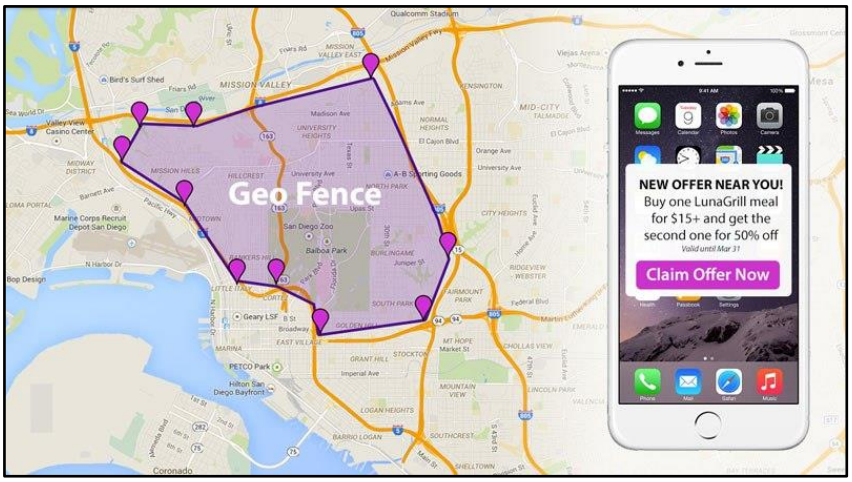
2.2.6 Google Maps API

A API Maps JavaScript é um serviço do Google que possibilita a utilização e personalização de mapas interativos em aplicações web. Com ela é possível exibir localizações, criar marcadores personalizados e até traçar linhas que representam o trajeto por onde um objeto se deslocou. A utilização desta API pode ser enriquecedora para projetos que tenham funções como geolocalização e monitoramento em tempo real (GOOGLE DEVELOPERS, 2025).

2.2.7 Geofencing

O *Geofencing* é uma tecnologia que permite criar uma cerca virtual que delimita um espaço geográfico. Essa cerca pode ter qualquer formato e serve para identificar se um objeto entrou ou saiu da área delimitada, com base em sua localização GPS (WAWRZYNIAK; HYLA, 2016, **apud** LOBÔ, 2022).

Figura 15 - Imagem Geofencing



Fonte: Expert Digital, 2025.

2.2.8 MongoDB

Conforme a expansão da internet, aumentaram-se as exigências cobradas de um banco de dados. Diante dessa necessidade de mudança, Paniz (2023) afirma que surgiu o conceito de NoSQL (Not Only SQL), buscando apresentar uma nova maneira de armazenar dados.

MongoDB é um sistema de gerenciamento de banco de dados (SGBD) não relacional que utiliza documentos em formato JSON (JavaScript Object Notation) para armazenar informações, simplificando a gestão dos dados pelos desenvolvedores (IBM).

Boaglio (2020) afirma que, diferente das maneiras convencionais, com MongoDB, os dados são organizados conforme a necessidade do sistema, obedecendo às regras da aplicação, sem possuir as limitações de uma tabela relacional.

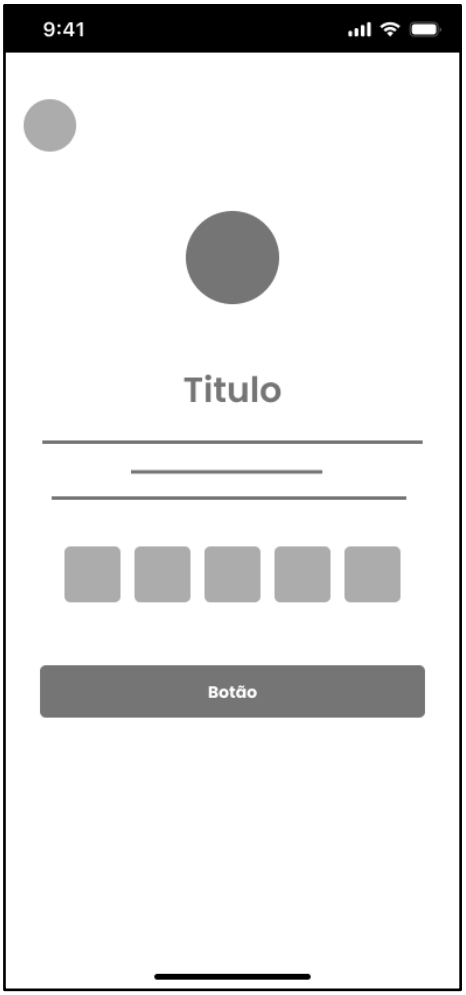
2.2.9 Wireframes

De acordo com Teixeira (2014) o Wireframe é o desenho de uma interface que indica onde cada elemento ficará apresentando de forma intuitiva como será o resultado, além de ser usado como guia para ter uma ideia sobre o layout final, agilizando o processo.

2.2.9.1 Wireframes baixa fidelidade

Para Rodrigues (2017) os Wireframes de baixa fidelidade são esboços simples usados como uma forma rápida de mostrar uma ideia do produto, mesmo não sendo fiel ao layout final. No entanto, os Wireframes de baixa fidelidade possuem diversas vantagens, como baixo custo e a rápida produção, sendo modificados mais facilmente quando necessário.

Figura 16 - Exemplo de wireframes de baixa fidelidade



Fonte: De Autoria Própria, 2025.

2.2.9.2 Wireframes alta fidelidade

Os Wireframes de alta fidelidade usam elementos que serão colocados no estado final do produto, constituindo, consequentemente, em uma versão mais fiel do que os

wireframes de baixa fidelidade, no entanto, acabam por ser mais caro e demandar mais tempo para ser produzido (Rodrigues, 2017).

Figura 17 - Exemplo de wireframes de alta fidelidade

Fonte: De Autoria Própria, 2025.

2.2.10 UX (User Experience)

UX é um termo utilizado para se referir a experiência que um usuário tem ao utilizar certa aplicação. Grant (2019) aponta que para trabalhar com essa questão de experiência, deve se ter empatia e objetividade para poder entender as necessidades e frustrações dos usuários.

Como afirma Teixeira (2014), UX é uma área bem ampla, possuindo várias faces que são utilizadas para aprimorar a interação do usuário, como: organizar um conteúdo dentro da tela, garantir que as interfaces sejam fáceis de usar, definir o fluxo de comportamentos de uma aplicação, entender o público-alvo do produto, entre outros fatores.

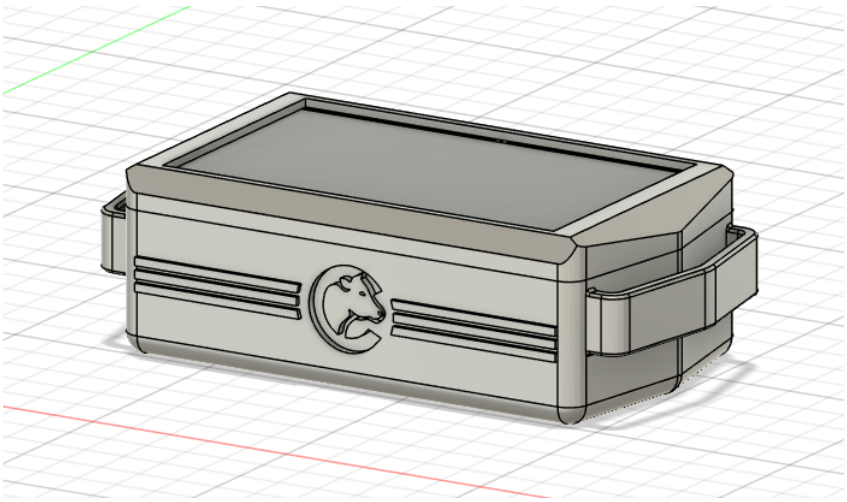
2.2.11 LoRa

Segundo Teixeira e Almeida (2017) O LoRa (Long Range - Longo alcance) desenvolvido pela Semtech Corporation, é uma tecnologia de comunicação sem fio por radiofrequência e baixo consumo de energia. Ela permite a comunicação de dispositivos em uma mesma rede a grandes distâncias sem a necessidade de um cabeamento. Devido a essas características se torna o ideal para aplicações de internet das coisas (IoT) em locais remotos, onde não existe uma conexão Wi-Fi ou rede de celular.

2.2.12 Modelagem 3D

Muito utilizado pela indústria, a modelagem 3D é o processo de desenvolvimento de objetos, personagens, cenários e formas complexas em três dimensões e pode ser utilizada em diversas áreas, desde o design de móveis e objetos até a fabricação de peças automotivas (LOPES, 2023). Abaixo um exemplo de modelagem 3D:

Figura 18 – Exemplo de modelagem 3D T-Fence



Fonte: De Autoria Própria, 2025.

2.2.13 UML (Unified Modeling Language)

De acordo com GUEDES (2018), UML é uma linguagem de modelagem utilizada internacionalmente, buscando padronizar o desenvolvimento dos softwares se baseando na programação orientada a objetos. Surgiu da unificação de várias linguagens de modelagem que existiam nos anos 80 e 90, trazendo um padrão para o segmento na época (FOWLER, 2005).

Consoante BOOCH (2012), a modelagem nos permite compreender o funcionamento de um sistema por completo. No entanto, para se obter esse entendimento é necessários vários modelos conectados entre si. Assim, é possível utilizar a UML para a visualização, especificação, construção e documentação de um projeto.

Completando essa ideia, GUEDES (2018) destaca que a UML tem diversos diagramas, com objetivo de fornecer várias visões do sistema, analisando diversos aspectos diferentes. O autor também reforça que a utilização desses diagramas é

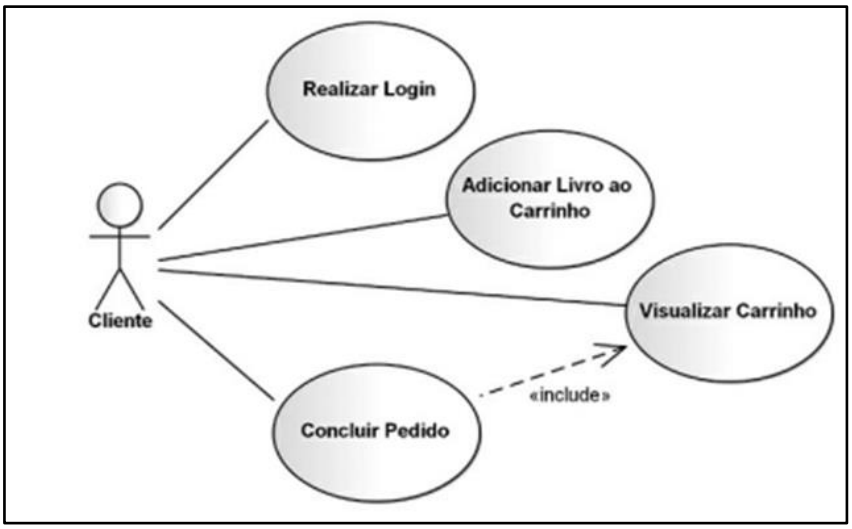
importante, já que ele auxilia na identificação de falhas, diminuindo a possibilidade de erros futuros.

Nos próximos capítulos iremos apresentar os diagramas que utilizamos para a modelagem do nosso sistema.

2.2.13.1 UML- Diagrama de Caso de Uso

Tem como objetivo apresentar uma visão geral do sistema, exibindo as funcionalidades que devem ser oferecidas. No diagrama, a representação se dá por um conjunto de casos de uso, retratando as funcionalidades, e os atores, sendo tudo aquilo que interage com o sistema (GUEDES, 2018).

Figura 19 - Exemplificação de um diagrama de caso de uso



Fonte: Guedes, 2018.

2.2.13.2 UML- Documentação de Caso de Uso

Segundo GUEDES (2018), a documentação do caso de uso é realizada de maneira simples, trazendo informações básicas, como as funcionalidades do caso de uso, atores e as etapas que devem ser executadas. Ainda sobre esse assunto, o autor diz que não possui um formato padronizado para se documentar, sendo maleável e de sua escolha.

Figura 20 - Exemplificação de um diagrama de caso de uso

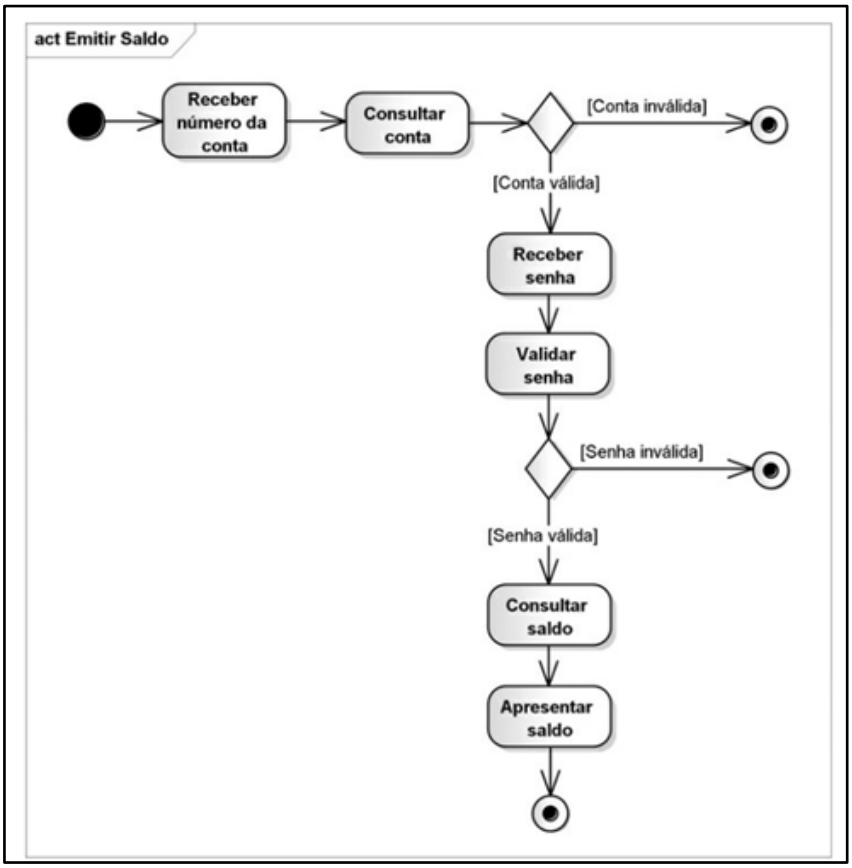
Nome do Caso de Uso	**UC06 – Emitir Saldo**
Ator Principal	Cliente
Atores Secundários	
Resumo	Descreve os passos necessários para um cliente obter o saldo referente a uma determinada conta
Pré-condições	
Pós-condições	
	Cenário Principal
Ações do Ator	**Ações do Sistema**
1. Informar o número da conta	
	2. Verificar a existência da conta
	3. Solicitar a senha da conta
4. Informar a senha	
	5. Verificar se a senha está correta
	6. Emitir o saldo
Restrições/Validações	1. A conta precisa existir e estar ativa 2. A senha precisa estar correta
	Cenário de Exceção I – Conta não encontrada
Ações do Ator	**Ações do Sistema**
	1. Comunicar ao cliente que o número da conta informada não foi encontrado
	Cenário de Exceção II – Senha inválida
Ações do Ator	**Ações do Sistema**
	1. Comunicar ao cliente que a senha fornecida não combina com a senha da conta

Fonte: Guedes, 2018.

2.2.13.3 UML- Diagrama de Atividade

O diagrama de atividade busca modelar a ordem de atividades em um processo, exibindo a sua ocorrência, contendo ações, nós de atividade, fluxos e valores de objeto (BOOCH, 2012).

Figura 21 - Exemplificação de um diagrama de atividade

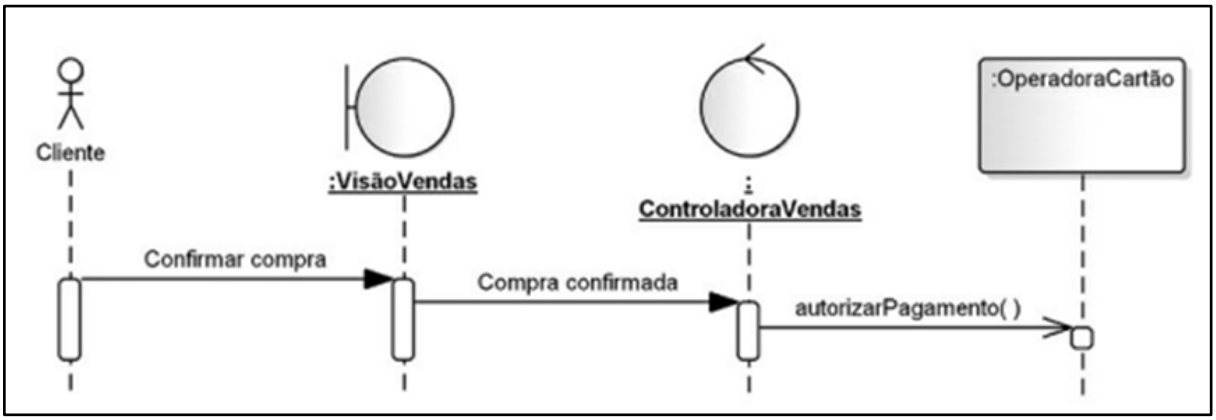


Fonte: Guedes, 2018.

2.2.13.4 UML- Diagrama de Sequência

O diagrama de sequência tem como objetivo determinar a ordem dos eventos em um determinado processo, baseando-se no caso de uso. É normal que cada caso de uso tenha um diagrama de sequência, sendo um processo disparado por um ator (GUEDES, 2018).

Figura 22 - Exemplo de diagrama de Sequência

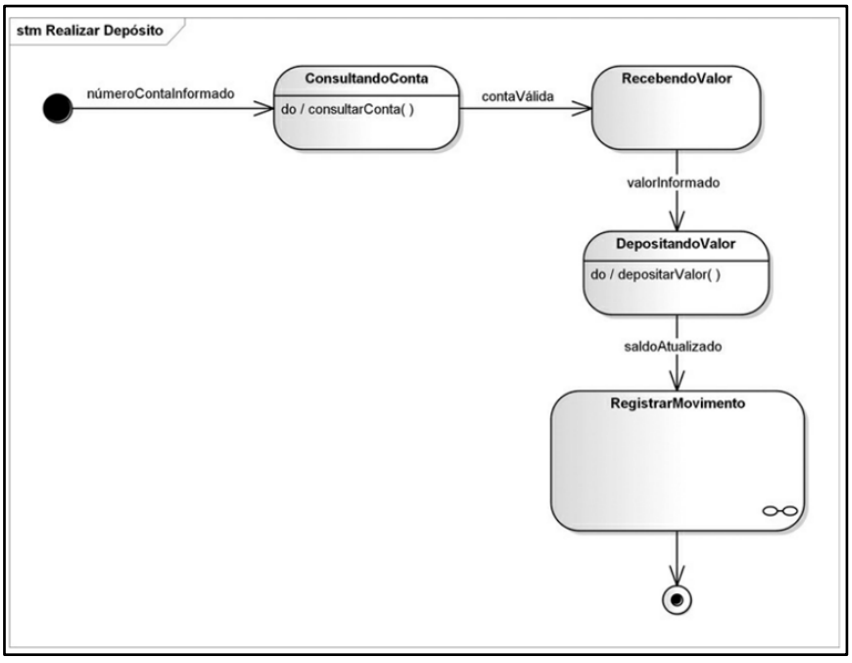


Fonte: Guedes, 2018.

2.2.13.5 UML- Diagrama de Máquina de Estado

Para BOOCH (2012), esse diagrama especifica a sequência de estados que um objeto passa durante seu tempo de vida, levando em consideração seus eventos. O autor também afirma que o estado é a situação de vida de um objeto, enquanto evento é a especificação de uma ocorrência.

Figura 23- Exemplo de diagrama de Máquina de Estados



Fonte: Guedes, 2018.

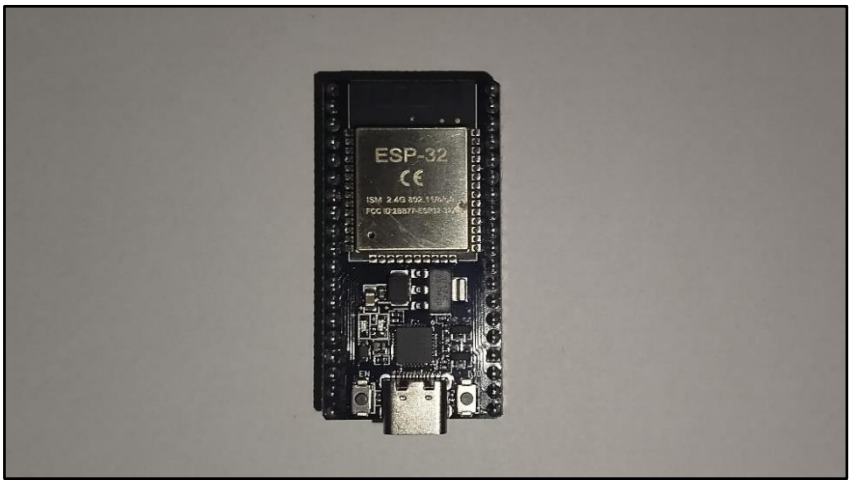
2.2.14 IoT

Conforme afirma Magrani (2018), IoT (Internet of Things - Internet das Coisas) se baseia em objetos físicos que estejam conectados com a internet e que compartilhem informações por meio de sensores, com a finalidade de auxiliar ou automatizar tarefas. Segundo Oliveira (2021), o IoT já é um conceito que existe há muito tempo, começou a ter destaque com o avanço da internet, surgiu com a ideia de interligar dispositivos por meio da internet, assim possibilitando não apenas a comunicação entre eles, mas também torná-los inteligentes melhorando a forma com que interagem com o ambiente.

2.2.14.1 ESP32

Lançado em 2016 pela Espressif Systems, o ESP32 é um microcontrolador avançado, considerado o melhor em comparação a outros microcontroladores do mercado, devido ao seu poder de processamento e capacidade de memória, o ESP32 conta com um processador dual core com 2 núcleos físicos, 530Kb de memória RAM, podendo chegar a um clock de até 240Mhz e ainda possui conectividade com Bluetooth e Wi-Fi, sendo assim uma ótima opção para aplicações IoT (SANTOS; LARA JUNIOR, 2019).

Figura 24 - Imagem do ESP32

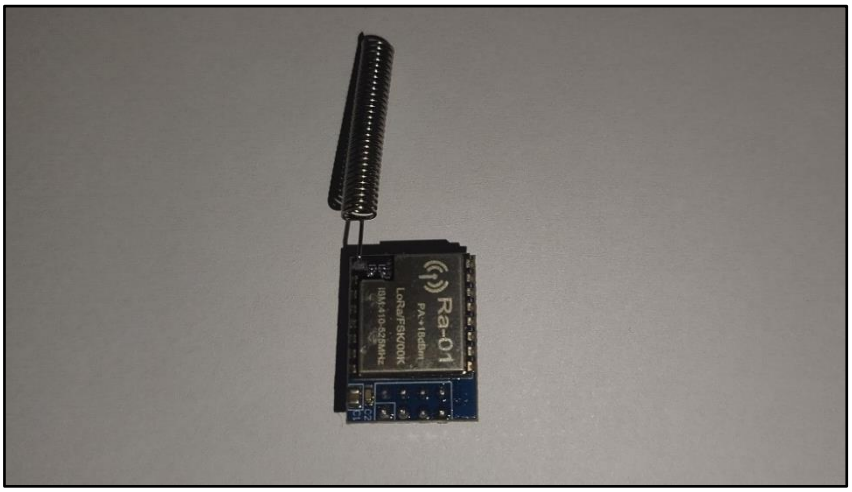


Fonte: De Autoria Própria, 2025.

2.2.14.2 Módulo LoRa

O módulo LoRa é um transmissor sem fio baseado na tecnologia LoRa (Long Range), equipado desenvolvido pela SEMTECH, este módulo permite a comunicação a longas distâncias com um baixo consumo de energia, muito utilizado em aplicações IoT (Internet das coisas) (SARAVATI, 2025).

Figura 25 - Módulo Transmissor LoRa SX1278

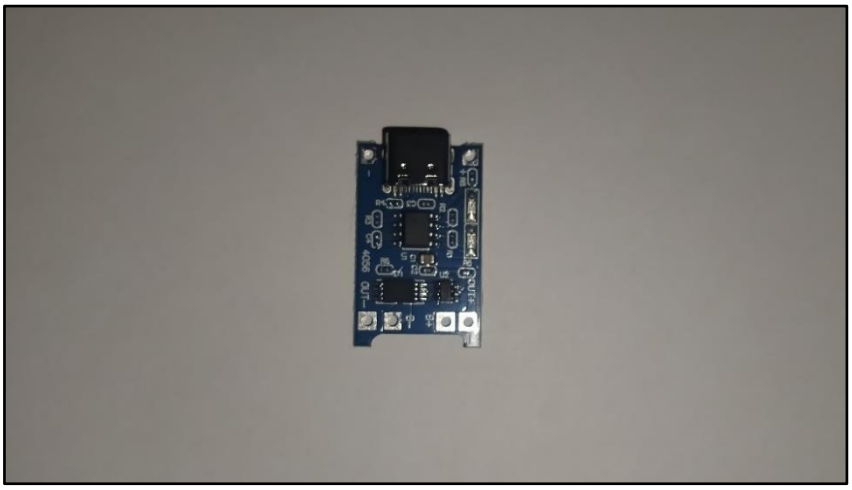


Fonte: De Autoria Própria, 2025.

2.2.14.3 Módulo controlador de carga

Como destaca Oliveira (2021) o controlador de carga é componente essencial de um IoT já que garante o bom funcionamento de sua bateria. Um controlador de carga possui diversas funções, dentre elas estão, evitar descarga total, que pode danificar a bateria diminuindo sua vida útil, limitar a carga máxima, evitando uma sobrecarga o que pode causar um superaquecimento ou até mesmo à explosão da bateria.

Figura 26 - Módulo controlador de carga TP4057



Fonte: De Autoria Própria, 2025.

2.2.14.4 Mini painel solar

De acordo com Braga (2008) os painéis solares são capazes de converter radiação solar em energia elétrica a partir do efeito fotovoltaico. Descoberto pelo físico francês Edmond Becquerel em 1839, que ao observar que certos materiais semicondutores, expostos à luz, geram uma pequena corrente elétrica.

Segundo Oliveira (2021) a energia fotovoltaica tem se mostrado uma ótima solução para alimentar energeticamente dispositivos de Internet das coisas (IoT), principalmente quando se trata de aplicações externas e remotas, onde a infraestrutura elétrica pode ser limitada ou até mesmo inexistente.

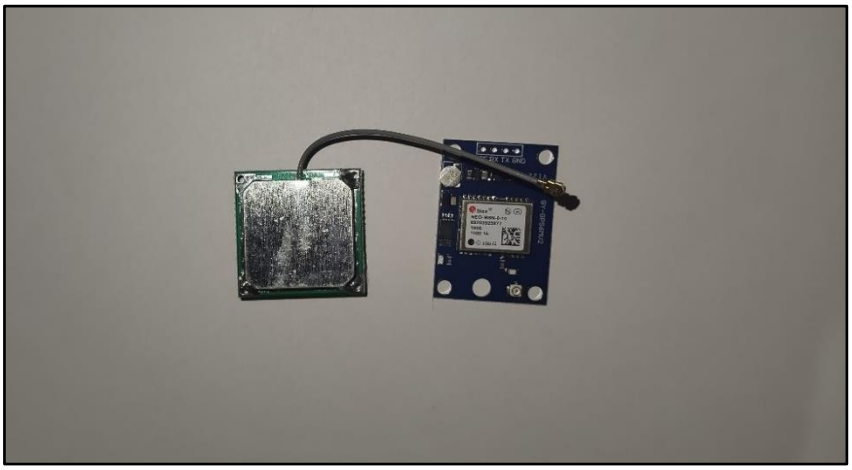
Figura 27 - Mini painel solar

Fonte: RoboCore, 2025.

2.2.14.5 Módulo GPS Neo-M8N

O módulo GPS Neo-M8N é um dispositivo capaz de determinar coordenadas geográficas com alta precisão, se comunica com até três sistemas de navegação por satélite (GNSS), GPS, GLONASS e BeiDou, garantindo menor taxa de erro e alta confiabilidade (CURTOCIRCUITO, 2025).

Figura 28- Módulo GPS Neo-M8N

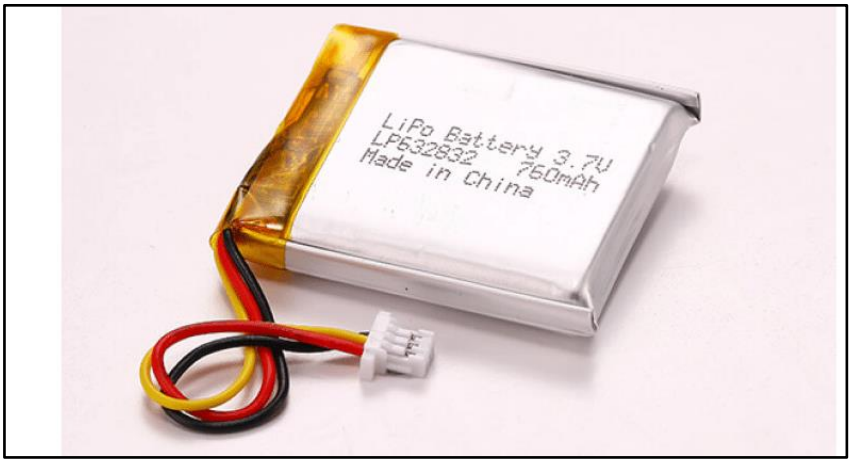


Fonte: De Autoria Própria, 2025.

2.2.14.6 Bateria Li-Po

Existe uma variedade de baterias, dentre elas está a bateria de lítio-polímero (Li-Po), sendo uma variação da bateria de lítio, ela se destaca pela sua capacidade de se adequar em aplicações com espaço limitado, sendo possível encontrar baterias Li-Po de diferentes formas e tamanhos, além de serem mais leves e seguras (MAKERHERO, 2025).

Figura 29 - Bateria de lítio-polímero

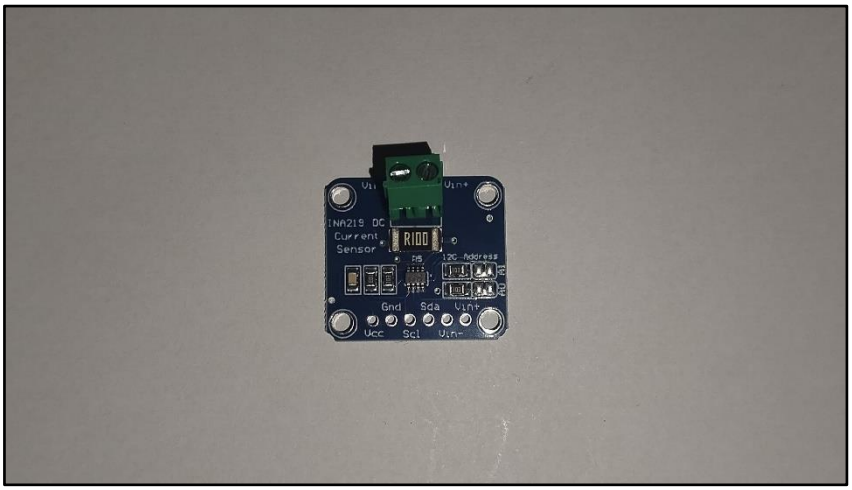


Fonte: Makerhero, 2025.

2.2.14.7 Módulo MCU-219

O módulo MCU-219 foi projetado para monitoramento de corrente e tensão em sistemas DC, ele opera entre 3V e 5V, mede tensões de um barramento de até 26VDC e correntes de até +3,2 A (CURTOCIRCUITO, 2021). Esse módulo será utilizado para estimar a porcentagem da bateria da coleira.

Figura 30 – Imagem do MCU-219



Fonte: De Autoria Própria, 2025.

3 Desenvolvimento

Neste capítulo será apresentado as etapas utilizadas para o desenvolvimento do nosso projeto, abordando desde a modelagem dos diagramas UML, até a construção de nossos aplicativos e dispositivos IoT.

3.1 Diagrama de Casos de Uso

3.2 Documentação Caso de Uso

3.3 Diagrama de Atividade

3.4 Diagrama de Sequência

3.5 Diagrama de Máquina de Estados

REFERÊNCIAS

AGUIAR, Gustavo Stor. **NODE.JS: Estudo Tecnológico e Desenvolvimento Full-stack JavaScript de Plataforma De competições em Problemas Algorítmicos**. 2015. Dissertação (Graduação em Ciência da Computação) -Universidade Federal de Pernambuco, Recife, 2015.

BRAGA, Renata Pereira**. Energia solar fotovoltaica: fundamentos e aplicações**. 2008. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008. Disponível em: https://pantheon.ufrj.br/handle/11422/7372. Acesso em: 25 maio 2025.

BOAGLIO, Fernando**. MongoDB: Construa novas aplicações com novas tecnologias**. São Paulo, SP: Casa do Código, 2020. ISBN: 978-85-5519-043-8

BROWN, Ethan. **Programação Web com Node e Express Beneficiando-se da stack JavaScript**. Rio de Janeiro: Novatec Editora, 2020.

CONFEDERAÇÃO DA AGRICULTURA E PECUÁRIA DO BRASIL. **Estudo sobre a criminalidade no campo**. Brasília, DF: CNA, 2017. Disponível em: http://cnabrasil.org.br/publicacoes/estudo-sobre-criminalidade-no-campo. Acesso em: 9 ago. 2025.

CURTO CIRCUITO**. INA219: como usar**. Disponível em: https://curtocircuito.com.br/blog/sensores/ina219-como-usar?srsltid=AfmBOooI3oGcmGtmR5k2dmJ79ppY5DVDOe6Rv8eZ9HE8Tjjgo6UYuVW2&utm. Acesso em: 25 maio 2025.

CURTO CIRCUITO. **Módulo GPS Neo-M8N**. Disponível em: https://curtocircuito.com.br/modulo-gps-neo-m8n.html. Acesso em: 26 maio 2025.
DEITEL, Harvey M.; DEITEL, Paul J. **C++: Como Programar**. 5. ed. São Paulo: Pearson Prentice Hall, 2005.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivan. UML: Guia Do Usuário. 2. ed. [S. l.]. GEN LTC, 2006.

EMBRAPA. **Cientistas usam drones com câmeras inclinadas para monitorar gado no pasto**. Brasília, DF: Embrapa, 2021. Disponível em: https://www.embrapa.br/acre/busca-de-noticias/-/noticia/58844049/cientistas-usam-drones-com-cameras-inclinadas-para-monitorar-gado-no-pasto. Acesso em: 9 ago. 2025

ESCUDELARIO, Bruna; PINHO, Diego. React Native: **Desenvolvimento de Aplicativos Mobile com React**. 3.ed. São Paulo: Casa do Código, 2021. 189 p

FALCÃO, Filipe Dourado. **Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native**. 2022.

FERREIRA, Diego Eis Elcio**. HTML5 e CSS3 com Farinha e Pimenta**. São Paulo: Tableless, 2012.

FLANAGAN, David. JavaScript – **O Guia Definitivo**. 6. ed. Porto Alegre: Bookman, 2013.

FOWLER, Martin. **Um Breve Guia para a Linguagem-padrão de Modelagem de Objetos**. 3. ed. Porto Alegre: Bookman, 2005. 160 p.

GUEDES, Gilleanes. **UML 2**: Uma Abordagem Prática. 3. ed. São Paulo: Novatec Editora Ltda., 2018. 496 p.

GOOGLE DEVELOPERS**. Visão geral da API Maps JavaScript**. 2025. Disponível em: https://developers.google.com/maps/documentation/javascript/overview?hl=pt-br. Acesso em: 25 maio 2025.

GRONER, Loaine**. Estrutura de Dados e Algoritimos em JavaScript**. 2. ed. Rio de Janeiro: Novatec Editora, 2018.

IBM. **Por que MongoDB?** Disponível em: https://www.ibm.com/br-pt/think/topics/mongodb Acesso em: 14 de julho de 2025.

LÔBO, Rafael Miranda. **Geofencing: monitoramento de propriedades rurais**. 2022. Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) – Pontifícia Universidade Católica de Goiás, Goiânia, 2022. Disponível em: https://repositorio.pucgoias.edu.br/jspui/handle/123456789/4999. Acesso em: 25 maio 2025.

MAGRANI, Eduardo. **A internet das coisas**. 2. ed. Rio de Janeiro: FGV Editora, 2018.

MAKERHERO. **Pilhas e Baterias: Principais Tipos**. Disponível em: https://www.makerhero.com/blog/pilhas-e-baterias-principais-tipos/. Acesso em: 26 maio 2025.

MCFARLAND, david sawyer. **CSS o Manual Que Faltava**. São Paulo: Universo dos livros 2010.

MINISTÉRIO DA AGRICULTURA E PECUÁRIA. **Crescimento da economia brasileira é impulsionado pela alta de 1,5% da agropecuária em 2023.** Brasília, DF, 2023. Disponível em: https://www.gov.br/agricultura/pt-br/assuntos/noticias/crescimento-da-economia-brasileira-e-impulsionado-pela-alta-de-15-da-agropecuaria-em-2023. Acesso em: 9 ago. 2025.

OLIVEIRA, Sérgio de**. Internet das coisas com ESP8266, Arduino e Raspberry Pi**. 2. ed. São Paulo: Novatec, 2021.

OLIVEIRA, Carlos Antonio Ferreira de; MEDINA, Gabriel da Silva; TEIXEIRA, Lana Mara Silva. **Análise dos registros dos crimes de furto e roubo contra propriedades rurais em Goiás nos anos de 2017 e 2018**. Boletim Goiano de Geografia, Goiânia, v. 41, n. 1, 2021. Disponível em: https://revistas.ufg.br/bgg/article/view/62579. Acesso em: 9 ago. 2025.

PANIZ, David. **NoSQL: Como armazenar os dados de uma aplicação moderna**. São Paulo, SP: Casa do Código, 2023. ISBN:978-85-5519-192-3

PEREIRA, Caio Ribeiro. **Node.JS Aplicações Web Real-Time com Node.JS**. Casa Do Código, 2014.

REACT NATIVE. **React Native · Learn once, write anywhere.** Disponível em: https://reactnative.dev/. Acesso em: 23 de maio de 2025.

RODRIGUES, Paulo Henrique de Araujo. **Aplicação do Conceito Visual Material Design no Desenvolvimento de um Protótipo de Interface Gráfica**. 2017. 51 f. Trabalho de Conclusão de Curso (Graduação) – Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

RODRIGUES, Nadir; ROSSO, Gisele. **Cientistas usam drones com câmeras inclinadas para monitorar gado no pasto.** Editada por Mariana Medeiros. Brasília, DF: Embrapa, 2021. Disponível em: https://www.embrapa.br/acre/busca-de-noticias/-/noticia/58844049/cientistas-usam-drones-com-cameras-inclinadas-para-monitorar-gado-no-pasto. Acesso em: 9 ago. 2025.

EXPO. **Expo.** Disponível em: https://expo.dev/. Acesso em: 24 de maio de 2025.

SANTOS, Jean Willian; LARA JUNIOR, Renato Capelin de**. Sistema de automatização residencial de baixo custo controlado pelo microcontrolador ESP32 e monitorado via smartphone**. 2019. 46 f.

Trabalho de Conclusão de Curso (**Tecnologia em Automação Industrial**) – Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2019. Disponível em: https://repositorio.utfpr.edu.br/jspui/handle/1/16960. Acesso em: 25 maio 2025.

SARAVATI. **LoRa: sem fio de longo alcance e baixo consumo**. [S.l.], 2025. Disponível em: https://blog.saravati.com.br/lora-sem-fio-longo-alcance-baixo-consumo/. Acesso em: 27 maio 2025.

SILVA, Maurício Samy. **CSS3: Desenvolva Aplicações Web Profissionais com Uso dos Poderosos Recursos de Estilização das CSS3**. Rio de Janeiro: Novatec Editora, 2011.

SILVA, Maurício Samy. **Fundamentos de HTML5 e CSS3**. Rio de Janeiro: Novatec Editora, 2015.

SILVA, Maurício Samy. HTML5: **A linguagem de marcação que revolucionou a web**. 2. ed. Rio de Janeiro: Novatec Editora, 2011.

SILVA, Maurício Samy. Javascript, **O Guia Do Programador**. Rio de Janeiro: Novatec Editora, 2010.

TEIXEIRA, Grazielle Bonaldi; ALMEIDA, João Víctor Peroni de. **Rede LoRa® e protocolo LoRaWAN® aplicados na agricultura de precisão no Brasil**. 2017. 76 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica) – Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2017. Disponível em: https://repositorio.utfpr.edu.br/jspui/handle/1/16191. Acesso em: 24 maio 2025.

TEXEIRA, Fabrício. **Introdução e boa práticas em UX Design**. 1. ed. São Paulo: Casa do Código, 2014.

VILLAS-BOAS, Sergio Barbosa. **C/C++ e Orientação a Objetos em Ambiente Multiplataforma.** Rio de Janeiro, RJ, Brasil: [s.n.], 2001.

ZAKAS, Nicholas C. **Javascript de Alto Desempenho**. Rio de Janeiro: Novatec Editora, 2010.