

**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

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

AGRADECIMENTOS

Agradeço

a

XX

XX

XX

XX

XX

"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX."

Autor

RESUMO

XXXXXXXXXXXXXXXXXX; XXXXXXXXXXXXXXXX; XXXXXXXXXXXXXXX; XXXXXXXXXXXXXXX; XXXXXXXXXXXXXXX.

Palavras-Chave: XXXXXXXXXXXXXXXX; XXXXXXXXXXXXXXXX; XXXXXXXXXXXXXXXX;
XXXXXXXXXXXXXX; XXXXXXXXXXXXXXXX.

ABSTRACT

Keywords: xxxxxxxxxxxxxxxxx; xxxxxxxxxxxxxxxxx; xxxxxxxxxxxxxxxxx; xxxxxxxxxxxxxxxxx; xxxxxxxxxxxxxxxxx.

LISTA DE ILUSTRAÇÕES

Figura 1 - Estrutura simples feita em HTML	15
Figura 2 - Exemplo de Formulário usando HTML.....	17
Figura 3 - Exemplo do Formulário no navegador	18
Figura 4 - Exemplo de código CSS	19
Figura 5 - Importação do CSS no HTML	20
Figura 6 - Resultado da Codificação CSS	21
Figura 7 - Exemplo de código Javascript	22
Figura 8 - Resultado da codificação JavaScript	22
Figura 9 - Importação do Javascript no HTML	23
Figura 10 - Exemplo de C++ utilizado no ESP32	24
Figura 11 - Circuito ESP32 com LED	26
Figura 12 - Imagem Geofencing	27
Figura 13 - Exemplificação de um diagrama de caso de uso	29
Figura 14 - Exemplificação de um diagrama de caso de uso	29
Figura 15 - Exemplificação de um diagrama de atividade	30
Figura 16 - Exemplo de diagrama de Sequência	31
Figura 17- Exemplo de diagrama de Máquina de Estados.....	31
Figura 18 - Imagem do ESP32	32
Figura 19 - Módulo Transmissor LoRa SX1278	33
Figura 20 - Módulo controlador de carga TP4057	33
Figura 21 - Mini painel solar	35
Figura 22- Módulo GPS Neo-M8N	35
Figura 23 - Bateria de lítio-polímero	36

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	14
2	Referencial Teórico.....	15
2.1	A Segurança nos campos rurais	15
2.2	Tecnologias Utilizadas	15
2.2.1	HTML	15
2.2.2	CSS.....	18
2.2.3	Javascript.....	21
2.2.4	React Native	23
2.2.4.1	EXPO	23
2.2.5	C++	24
2.2.6	Node.JS e NPM	26
2.2.7	Google Maps API	26
2.2.8	Geofencing.....	27
2.2.9	LoRa	27
2.2.9.1	UML (Unified Modeling Language).....	28
2.2.9.2	UML- Diagrama de Caso de Uso.....	28
2.2.9.3	UML- Documentação de Caso de Uso	29
2.2.9.4	UML- Diagrama de Atividade.....	30
2.2.9.5	UML- Diagrama de Sequência	30
2.2.9.6	UML- Diagrama de Máquina de Estado	31
2.2.10	IoT.....	32
2.2.10.1	ESP32	32
2.2.10.2	Módulo LoRa	33

2.2.10.3	Módulo controlador de carga	33
2.2.10.4	Mini painel solar.....	34
2.2.10.5	Módulo GPS Neo-M8N.....	35
2.2.10.6	Bateria Li-Po.....	35
REFERÊNCIAS	37

1 Introdução

XX
XX
XX

2 Referencial Teórico

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

2.1 A Segurança nos campos rurais

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

```

<!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: “
” 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

Formulário de Contato

Nome:

E-mail:

Sexo:

- Masculino
- Feminino

Mensagem:

Enviar

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 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

```

style.css
1 * {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5 }
6
7 body {
8   font-family: Arial, sans-serif;
9   background-color: #F0F0F0;
10  padding: 20px;
11 }
12
13 h1 {
14   color: #333;
15   text-align: center;
16 }
17
18 form {
19   background-color: #FFF;
20   padding: 20px;
21   border-radius: 8px;
22   max-width: 400px;
23   margin: 0 auto;
24   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
25 }
26
27 label {
28   font-weight: bold;
29 }
30
31 input[type="text"],
32 input[type="email"],
33 textarea {
34   width: 100%;
35   padding: 8px;
36   margin-top: 4px;
37   margin-bottom: 12px;
38   border: 1px solid #ccc;
39   border-radius: 4px;
40   box-sizing: border-box;
41 }
42
43 input[type="radio"] {
44   margin-right: 8px;
45 }
46
47 input[type="submit"] {
48   background-color: #E4CAF8;
49   color: #fff;
50   padding: 10px;
51   border: none;
52   border-radius: 4px;
53   cursor: pointer;
54   width: 100%;
55 }
56
57 input[type="submit"]:hover {
58   background-color: #D5E4FA;
59 }
60

```

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 altura 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

```
<head>
    <meta charset="UTF-8">
    <title>Formulário Simples</title>
    <link rel="stylesheet" href="style.css">
</head>
```

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

The screenshot shows a contact form titled "Formulário de Contato". The form fields include "Nome:" (Name), "E-mail:" (Email), "Sexo:" (Sex) with options "Masculino" and "Feminino", and "Mensagem:" (Message). A green "Enviar" (Send) button is at the bottom.

Formulário de Contato

Nome:

E-mail:

Sexo:

Masculino
 Feminino

Mensagem:

Enviar

Fonte: De autoria própria, 2025.

2.2.3 Javascript

Criado pela Netscape e licenciado pela atual Oracle, o Javascript 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

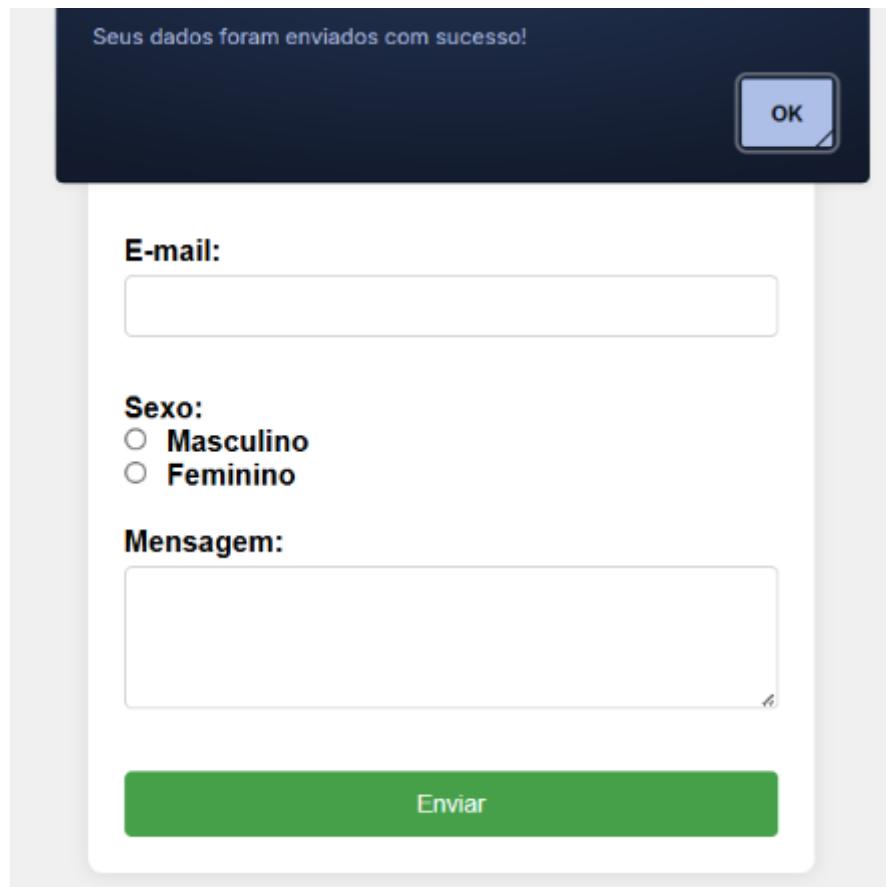
```
1  function enviarForm() {  
2      alert("Seus dados foram enviados com sucesso!");  
3      document.getElementById("meuFormulario").submit();  
4 }
```

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.

2.2.4 React Native

Segundo ESCUDELARIO e PINHO (2021), o React Native é um framework, baseada no React e criada por engenheiros do Facebook. 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.

2.2.4.1 EXPO

Expo é um framework de código aberto utilizado para facilitar o desenvolvimento de aplicações multiplataforma com React Native. Oferece 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.

2.2.5 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 10 - Exemplo de C++ utilizado no ESP32



```
1 #define LED_PIN 2
2
3 void setup() {
4     pinMode(LED_PIN, OUTPUT);
5 }
6
7 void loop() {
8
9     digitalWrite(LED_PIN, HIGH);
10    delay(1000);
11
12    digitalWrite(LED_PIN, LOW);
13    delay(1000);
14
15 }
```

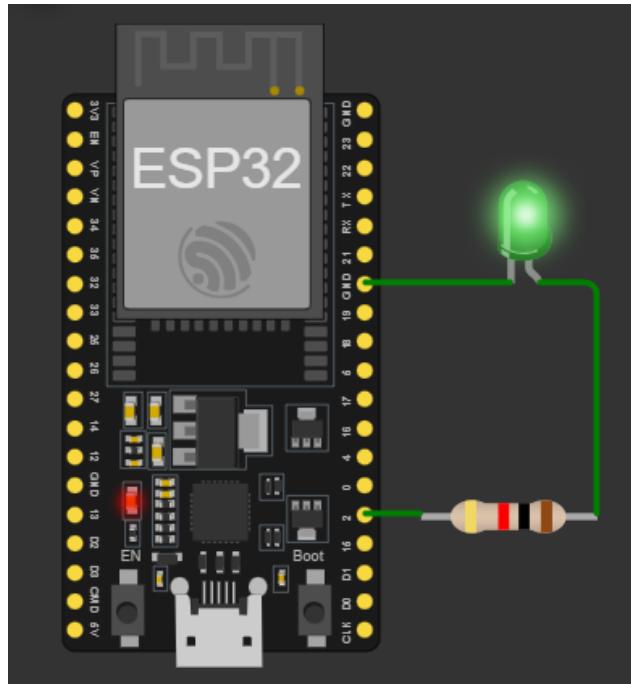
Fonte: De autoria própria, 2025.

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 11 - Circuito ESP32 com LED



Fonte: De autoria própria, 2025.

2.2.6 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 o gerenciamento de tabelas e já vem instalado junto do node. (BROWN, 2020).

2.2.7 Google Maps API

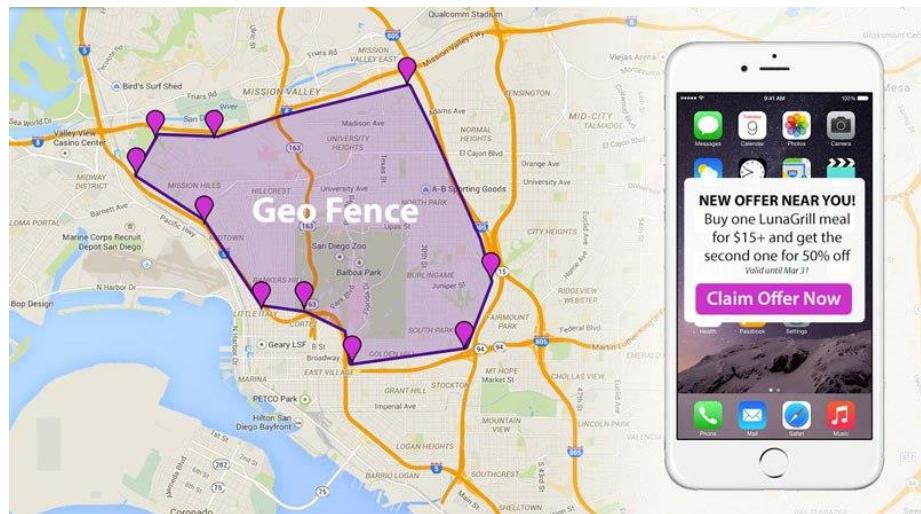
A API Maps Javascript é um serviço do google que possibilita a utilização e personalização de mapas interativos em aplicações web. Com ela é possível exibir localizações, criar marcadores personalizados e até traçar linhas que representam o

trajeto por onde um objeto se deslocou. A utilização desta API pode ser enriquecedora para projetos que tenhas funções como geolocalização e monitoramento em tempo real (GOOGLE DEVELOPERS, 2025).

2.2.8 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 12 - Imagem Geofencing



Fonte: Expert Digital, 2025.

2.2.9 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.

2.2.10 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.

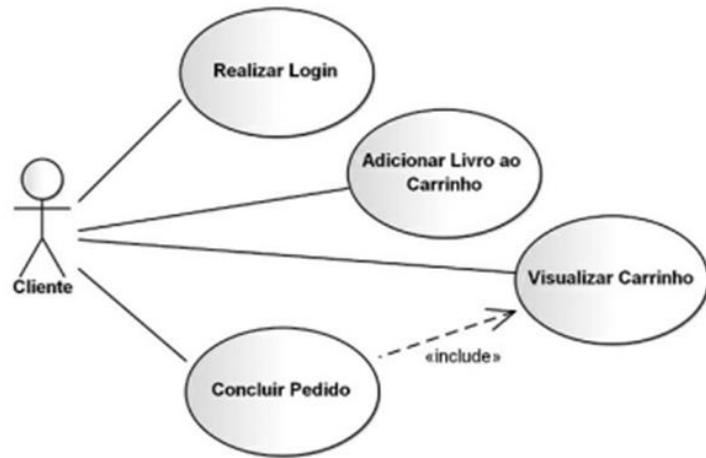
Completando essa ideia, GUEDES (2018) destaca que a UML tem diversos diagramas, com objetivo de fornecer várias visões do sistema, analisando diversos aspectos diferentes. O autor também reforça que a utilização desses diagramas é importante, já que ele auxilia na identificação de falhas, diminuindo a possibilidade de erros futuros.

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

2.2.10.1 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 13 - Exemplificação de um diagrama de caso de uso



2.2.10.2 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 14 - Exemplificação de um diagrama de caso de uso

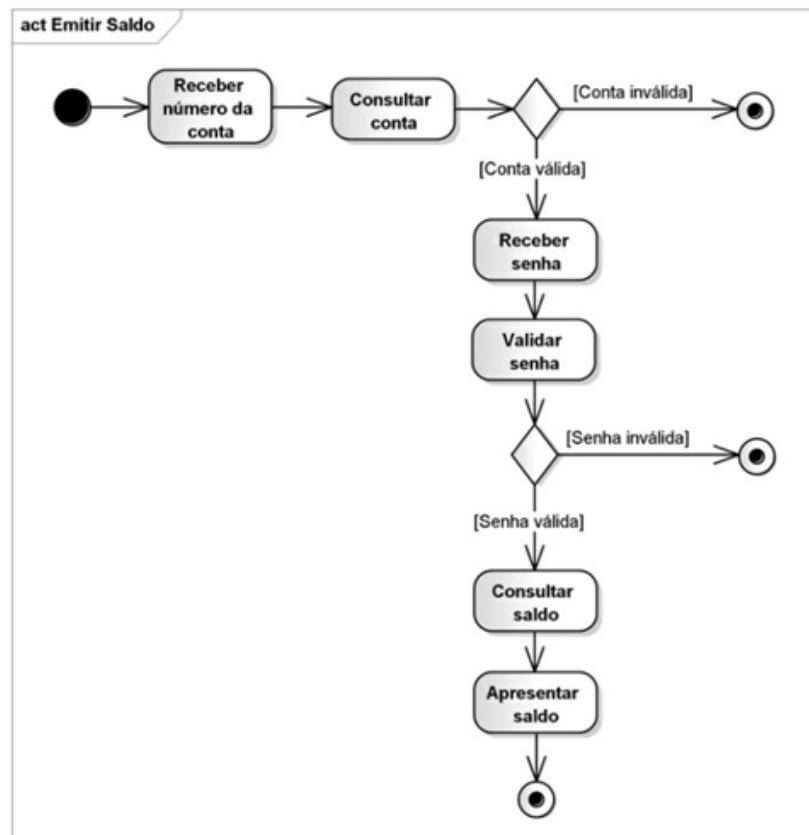
Nome do Caso de Uso		UC06 – Emitir Saldo
Ator Principal	Cliente	
Atores Secundários		
Resumo		Descreve os passos necessários para um cliente obter o saldo referente a uma determinada conta
Pré-condições		
Pós-condições		
Cenário Principal		
Ações do Ator		Ações do Sistema
1. Informar o número da conta		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.

2.2.10.3 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 15 - Exemplificação de um diagrama de atividade

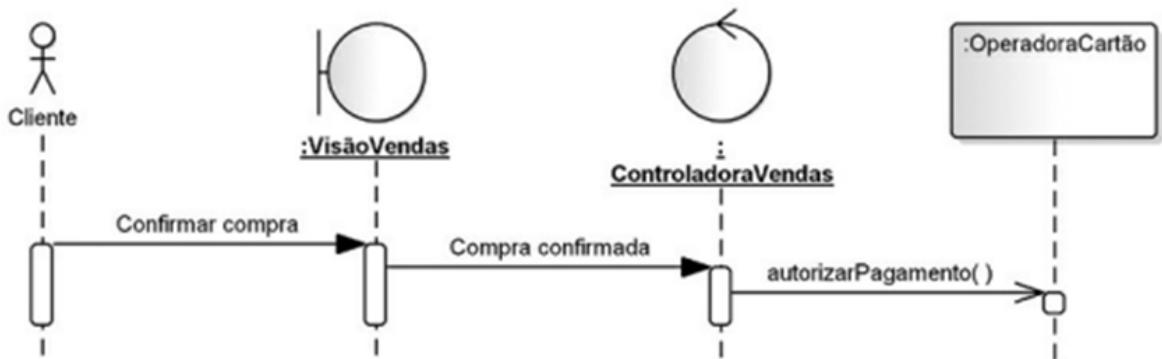


Fonte: Guedes, 2018.

2.2.10.4 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 16 - Exemplo de diagrama de Sequência

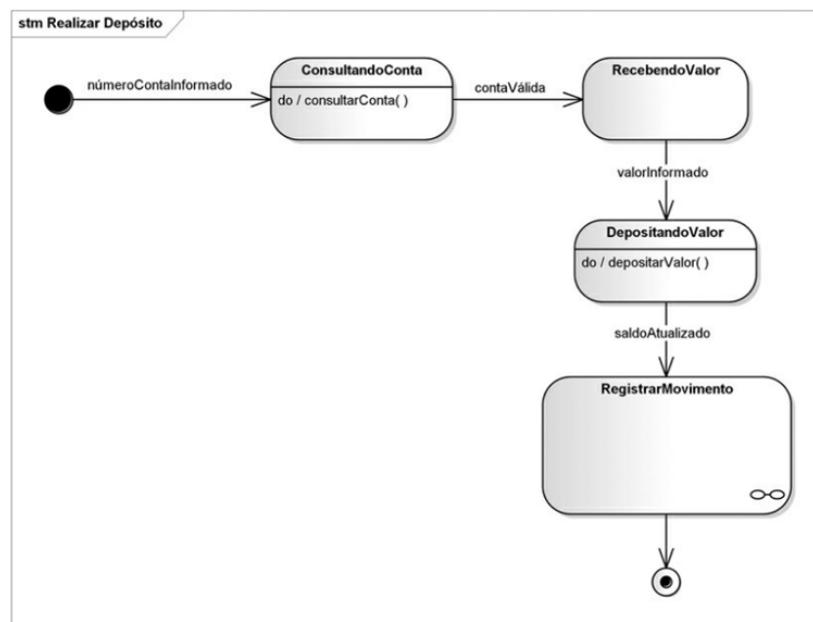


Fonte: Guedes, 2018.

2.2.10.5 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 17- Exemplo de diagrama de Máquina de Estados



Fonte: Guedes, 2018.

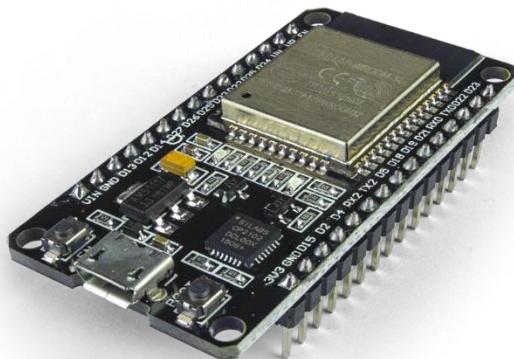
2.2.11 IoT

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.

2.2.11.1 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 18 - Imagem do ESP32

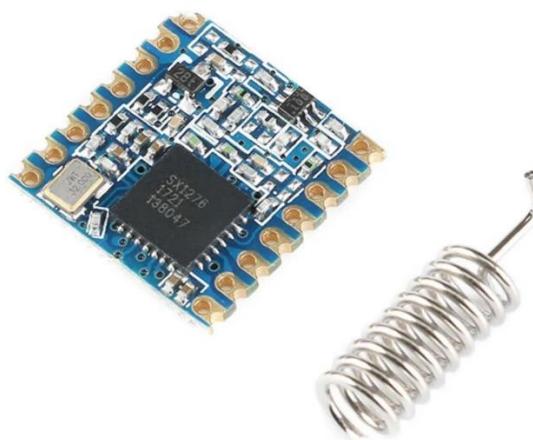


Fonte: RoboCore, 2025.

2.2.11.2 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 19 - Módulo Transmissor LoRa SX1278

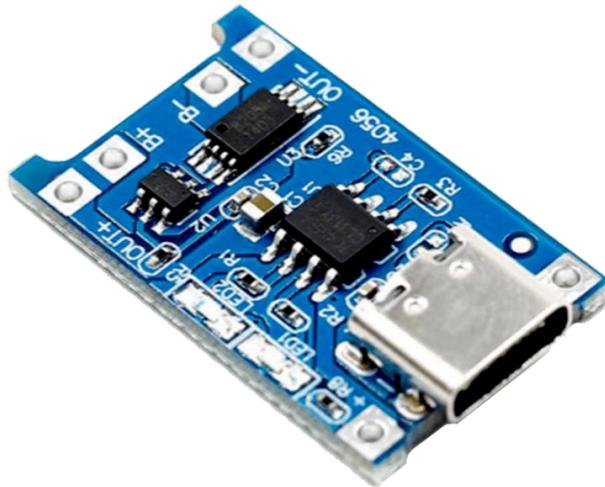


Fonte: Autocorerobotica, 2025.

2.2.11.3 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 20 - Módulo controlador de carga TP4057



Fonte: RoboCore, 2025.

2.2.11.4 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 21 - Mini painel solar



Fonte: RoboCore, 2025.

2.2.11.5 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 22- Módulo GPS Neo-M8N



Fonte: curtocircuito, 2025.

2.2.11.6 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 23 - Bateria de lítio-polímero



Fonte: makerhero, 2025.

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.

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

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. I.]. GEN LTC, 2006.

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 Algoritmos em JavaScript.** 2. ed. Rio de Janeiro: Novatec Editora, 2018.

LÔBO, Rafael Miranda. **Sistema de rastreamento de animais em áreas rurais utilizando tecnologia GPS e LoRa**. 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.

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

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.

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.I.], 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.

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.