CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ETEC DA ZONA LESTE

MTEC Desenvolvimento De Sistemas

Bernardo Vieira Costa dos Santos

David Romero Garcia

Giovanna Andrade Dantas

MIVICK: Sistema de Monitoramento e Auxílio para Ciclistas e Motociclistas

São Paulo

2025

Bernardo Vieira Costa dos Santos

David Romero Garcia

Giovanna Andrade Dantas

MIVICK: Sistema de Monitoramento e Auxílio para Ciclistas e Motociclistas

Trabalho de Conclusão de Curso apresentado ao mtec Desenvolvimento de Sistemas da Etec da 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

MIVICK

Sistema de Monitoramento e Auxílio para Ciclistas e Motociclistas

Bernardo Vieira Costa dos Santos

David Romero Garcia

Giovanna Andrade Dantas

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

AGRADECIMENTOS

Gostaríamos de agradecer primeiramente a Deus, que nos capacita e nos dá forças para continuar a cada dia e tornou possível a conclusão do nosso projeto e está conosco em cada parte de nossa caminhada.

Somos extremamente gratos também aos nossos familiares, que nos deram apoio em cada etapa do projeto, que estavam com a gente nos momentos difíceis e ajudaram a nos mantermos firmes e não desistir, mesmo quando tudo parecia sem solução. Em especial, prestamos nossa homenagem ao Jean Costa dos Santos que infelizmente nos deixou no meio do ano e não pode ver o projeto concluído, mas acreditava que seu filho iria dar o seu melhor.

Agradecemos profundamente a todos os professores e orientadores que nos apoiaram ao longo do projeto, especialmente ao nosso orientador Salomão Santana, que durante o ano nos ajudou dando dicas e ideias, além de nos encorajar e mostrar que o projeto era possível.

Aos nossos amigos, que estiveram conosco nos ajudando durante o processo e sempre estendera mão quando mais era preciso. Somos gratos a toda ajuda que nosso colega Nicolas Sinelli nos deu durante todo o processo. Seu apoio foi extremamente importante para nós

Para todos aqueles que acreditaram no projeto e seu potencial, nosso sincero obrigado, nada disso seria possível sem a ajuda e o apoio de todos vocês.

RESUMO

Este trabalho propõe o desenvolvimento de um sistema de monitoramento voltado à segurança de ciclistas e motociclistas em ciclofaixas e faixas azuis. Utilizando tecnologia IoT e sensores embarcados, como acelerômetro, giroscópio, sensor de impacto e sensor de distância, o dispositivo é capaz de detectar quedas, colisões leves e situações de risco. A motivação decorre do elevado risco de acidentes envolvendo motoristas desatentos, desgovernados ou sob efeito de álcool, que tornam esses usuários mais vulneráveis. O objetivo central é oferecer uma solução tecnológica que aumente a proteção, reduza incidentes e promova maior confiança no uso desses espaços. A pesquisa combina abordagens qualitativas e quantitativas para identificar os principais cenários de perigo e compreender as necessidades dos usuários. O sistema transmite alertas imediatos via aplicativo móvel. Espera-se que a solução aumente a segurança viária, reduza o número de acidentes e promova maior autonomia e confiança para ciclistas e motociclistas no ambiente urbano.

Palavras-Chave: Segurança; ciclistas; motociclistas; Iot; prevenção de acidentes.

ABSTRACT

This work proposes the development of a monitoring system aimed at the safety of cyclists and motorcyclists in cycle lanes and blue lanes. Using IoT technology and embedded sensors, such as an accelerometer, gyroscope, impact sensor and distance sensor, the device is capable of detecting falls, light collisions and risky situations. The motivation arises from the high risk of accidents involving drivers who are inattentive, out of control or under the influence of alcohol, which makes these users more vulnerable. The central objective is to offer a technological solution that increases protection, reduces incidents and promotes greater confidence in the use of these spaces. The research combines qualitative and quantitative approaches to identify key hazard scenarios and understand user needs. The system transmits immediate alerts via mobile app. The solution is expected to increase road safety, reduce the number of accidents and promote greater autonomy and confidence for cyclists and motorcyclists in the urban environment

Keywords: Security; cyclists; motorcyclists; IoT; accident prevention.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Associação Brasileira de Medicina de Tráfego (ABRAMET)

Cable News Network (CNN)

Cascading Style Sheet (CSS)

HyperText Markup Language (HTML)

Internet das Coisas (IoT)

JavaScript (JS)

Micro-Electro-Mechanical Systems (MEMS)

Open-Source Computer Vision Library (OpenCV)

Unified Modeling Language (UML)

eXtensible Markup Language (XML)

Sumário

1 INTRODUÇÃO

A falta de segurança no trânsito é evidente, especialmente para ciclistas e motociclistas, que são seus usuários mais vulneráveis. Mesmo com a existência de ciclofaixas e faixas azuis, muitos motoristas desrespeitam essas delimitações, expondo os usuários a diversos riscos. Casos de invasão de veículos, muitas vezes por motoristas embriagados ou desatentos, resultam em acidentes graves. Em 2024, a cidade de São Paulo registrou 1.031 mortes no trânsito, sendo 37% delas envolvendo motociclistas, segundo dados do Detran (2024).

Mediante ao problema apresentado, o objetivo do projeto se trata da criação do protótipo de um sistema IoT integrado com dispositivos e um aplicativo móvel capaz de prevenir acidentes e alertar em caso de emergência no trânsito, onde com a utilização de sensores acionará alertas em tempo real, contribuindo para a prevenção de acidentes e a proteção de vidas no trânsito.

Para alcançar tal objetivo, levantamos a seguinte questão: como a tecnologia pode melhorar a segurança nas ruas e diminuir a grande quantidade de acidentes envolvendo ciclistas e motociclistas?

Seguimos as hipóteses de como a falta de segurança e a negligência de motoristas têm aumentado o número de acidentes a cada ano. As principais causas encontradas foram as vias e ciclofaixas que se encontram em estado de abandono, sendo deixadas em condições precárias ou nem mesmo sendo terminadas. Isso faz com que os ciclistas e motociclistas acabem usando vias com um grande movimento de carros, caminhões e pouco monitoramento para esse tipo de caso. Ainda que os ciclistas usem as ciclofaixas e os motociclistas usem a faixa azul nas pistas, a falta de atenção também é um dos maiores fatores encontrados de acidentes, sendo causada, por exemplo, pelo uso de celular no trânsito, uso de fones e a falta de sinalização em estradas e rodovias.

Tendo em mente tais hipóteses, pode-se observar a necessidade da utilização da tecnologia para diminuir a quantidade de mortes e acidentes no trânsito, que levou a criação desse projeto, que visa proporcionar uma melhor segurança para o ciclista e motociclistas, que até o presente momento tem apenas uma equipamentos de segurança básicos que colocam a vida de ambos em risco, mesmo tendo os cuidados recomendados.

A área de foco do projeto se trata da zona metropolitana da capital de São Paulo, onde se encontra muitas vias, estradas e avenidas usadas tanto por motoristas de veículos grandes quanto ciclistas e motociclista, sendo eles o público que será focado ao decorrer da criação do sistema.

Para a realização do referencial teórico, foram utilizadas as seguintes referências: na parte da criação do dispositivo feito em React Native, o livro utilizado foi React Native: desenvolvimento de aplicativos mobile com React, de Bruna Escudelario e Diego Pinho. A escrita sobre UML foi baseada principalmente no livro

UML 2: uma abordagem prática, do autor Gilleanes Guedes. Na parte de IoT (Internet das coisas), foi-se utilizado para um maior entendimento o livro A Internet das Coisas: evolução, impactos e benefícios, de Maiko Gustavo De Godoi e Liriane Soares de Araújo.

Nos capítulos subsequentes, será mostrado primeiramente o referencial, com as tecnologias que utilizamos para a criação do projeto, o desenvolvimento de cada parte do sistema e, por fim, as considerações finais, que nos traz uma visão geral de todo o projeto.

2 REFERENCIAL TEÓRICO

Neste capítulo será introduzida a fundamentação teórica que fora de suma importância para a criação e desenvolvimento do projeto Mivick.

2.1 Acidentes de trânsito envolvendo ciclistas e motociclistas

Acidentes no trânsito que envolvem ciclistas e motociclistas ainda são uma das grandes razões de mortes no trânsito, fazendo com que o Brasil tenha números altos de acidentes fatais (MINISTÉRIO DA SAÚDE, 2023). A legislação brasileira reconhece a debilidade desses grupos, porém a falta de cuidado, prudência e ausência de infraestrutura útil aumenta os riscos enfrentados pelos ciclistas e motociclistas causando impactos físicos e mentais graves para eles (ABRAMET 2023).

De acordo com o Código de Trânsito Brasileiro, “o condutor de veículo motorizado, ao ultrapassar uma bicicleta, deverá reduzir a velocidade e manter a distância lateral de um metro e cinquenta centímetros” (BRASIL, 1997, art. 201). Além disso, “os veículos de maior porte serão sempre responsáveis pela segurança dos menores, os motorizados pelos não motorizados e, juntos, pela incolumidade dos pedestres” (BRASIL, 1997, art. 29). Embora essas regras existam, muitas vezes elas não são respeitadas. A violência no trânsito resulta em 76% dos casos de morte de ciclistas, pedestres e motociclistas (CNN, 2024) além de que também pode trazer lesões físicas, traumas psicológicos e limitações sociais que podem durar para sempre. Dessa forma, como cita o estudo feito por Bastos et al. (2018), é necessário investir na infraestrutura e na falta de segurança no trânsito, além de aumentar a visibilidade do ciclista em vias públicas, utilizando a tecnologia para isto.

Os dados até agora apresentados trouxeram a idealização de uma proposta que trará segurança nas vias públicas, utilizando sistemas de observação inteligente e sinalizações claras para ciclovias e espaços dos motoqueiros para que tais problemas possam ser diminuídos e mais pessoas possam se locomover dessa maneira. Com isso, o intuito é desenvolver um projeto baseado em IoT (Internet das Coisas) que pretende trazer dois aparelhos, ambos conectados em um aplicativo de celular, que conseguem detectar o perigo e alertar os ciclistas, motoqueiros e seus contatos.

2.2 Tecnologias utilizadas

Neste presente capítulo serão apresentadas as tecnologias utilizadas para o desenvolvimento do projeto.

2.2.1 React Native

React Native é considerado um framework utilizado com o objetivo de criar aplicações mobile híbridas, que funcionam tanto em dispositivos Android quanto iOS, e é completamente baseado na linguagem Javascript (ESCUDELARIO; Pinho, 2021, p. 2).

2.2.1.1 EXPO

Para Braga (2019), o EXPO possibilita que os desenvolvedores configurem rapidamente uma ferramenta de desenvolvimento sem precisar lidar diretamente com configurações nativas complexas.

Segundo Escudelario e Pinho (2021, p. 15-25), o Expo é considerado como um framework construído com base no React Native que viabiliza a construção de aplicações móveis utilizando apenas JavaScript, dispensando ferramentas como Android Studio ou Xcode.

Contudo, a adoção de módulos nativos não fornecidos pelo EXPO exige a ejeção da estruturação para um ambiente React Native puro (GALVÃO, 2018, p. 14).

Figura 1 - Exemplo feito com React Native

```
import { View, Text, TextInput} from "react-native";
import React, { useState } from 'react';

export default function Index() {

    const [nome, setNome] = useState("");
    const [email, setEmail] = useState("");

    return (
        <View>

            <Text>Bem vindo ao Aplicativo</Text>

            <TextInput

                onChangeText={setNome}
                value={nome}
                placeholder="Digite seu Nome no campo:"
            />

            <TextInput

                onChangeText={setEmail}
                value={email}
                placeholder="Digite seu Email no campo:"
            />

        </View>
    );
}
```

Fonte: Autoria Própria, 2025

Acima está um exemplo simples que pode ser feito usando react native no programa Visual Studio Code. O exemplo se trata de criar dois campos que podem ser preenchidos, um de nome e outro de Email. Esclarecendo o código de forma mais ampla:

Linha 1: Importa os componentes view, text e textInput da biblioteca react native.

Linha 2: Importa o React e o hook useState da biblioteca react.

Linha 4: Início da função index, que será exportada como padrão deste arquivo.

Linha 6: Cria o estado nome com o valor inicial vazio e a função setNome para atualizá-lo.

Linha 7: Cria o estado email com valor inicial vazio e a função setEmail para atualizá-lo.

Linha 9: Início do retorno do componente return, que define o que será renderizado na tela.

Linha 10: Abre a tag view, que serve como contêiner principal da tela.

Linha 12: Exibe um texto "Bem-vindo ao Aplicativo" dentro da tag text.

Linha 14: Abre um campo de entrada de texto com o componente textInput.

Linha 16: Define a função setNome para atualizar o valor do estado ao digitar no campo.

Linha 17: Liga o campo de entrada ao estado nome, para refletir seu valor atual.

Linha 18: Define o texto que aparece como dica dentro do campo: "Digite seu Nome no campo:".

Linha 19: Fecha a tag textinput do nome.

Linha 21: Abre outro campo de entrada de texto com textinput para o e-mail.

Linha 23: Define a função SetEmail para atualizar o estado ao digitar.

Linha 24: Liga o campo de entrada ao estado email.

Linha 25: Define o texto de dica dentro do campo: "Digite seu Email no campo:".

Linha 26: Fecha a tag textinput do e-mail.

Linha 28: Fecha a tag view principal.

Linha 29: Fecha o retorno da função.

Linha 30: Fecha a função index.

Figura 2 - QR Code gerado através do EXPO.

Fonte: Autoria Própria, 2025

Após escrever o código, para executá-lo basta ir ao terminal do programa Visual Studio Code, e digitar o comando “npx expo start”, neste processo um QR Code é gerado, podendo ser escaneado em um aplicativo mobile, especificamente no Expo Go, disponibilizado pela própria empresa Expo, dessa forma é possível visualizar o resultado da codificação diretamente no dispositivo móvel.

Figura 3 - Resultado exemplo código de React Native.

Fonte: Autoria Própria, 2025

Ao ser escaneado, o QR code leva até a tela acima, que mostra o resultado do código mostrado anteriormente na figura 1.

2.2.2 Estilização

Jobstribizer (2009, p. 37), define o css como uma linguagem voltada para estilização que opera juntamente com linguagens de marcação como HTML e XML

O principal objetivo é separar a construção do conteúdo da apresentação visual, possibilitando que os desenvolvedores definam estilos aplicados para elementos HTML de forma centralizada e reutilizável (BARROS; santos, 2008).

De acordo com Miletto (2014, p. 73-85) CSS facilita a manutenção e garante uniformidade na apresentação das páginas, sendo essencial no desenvolvimento moderno de interfaces visuais.

Figura 4 - Exemplo estilização com CSS.

```
1 const style =
2 StyleSheet.create({
3     container: {
4       flex: 1,
5       justifyContent: 'center',
6       alignItems: 'center',
7       padding: 25,
8     },
9
10    Text: {
11     fontSize: 12,
12     fontWeight: 'bold',
13     textAlign: 'center',
14     marginBottom: 17,
15     color: 'red'
16
17    },
18
19    input: {
20     height: 40,
21     borderRadius: 5,
22     backgroundColor: 'grey',
23     marginBottom: 13,
24    }
25 })
26
```

Fonte: Autoria Própria, 2025

Neste exemplo, o seguinte código está sendo usado para estilizar com CSS o que foi feito de react native até o momento. Explicando as linhas de código com mais detalhe:

Linha 2: Cria uma constância chamada Style.

Linha 3: Cria uma classe chamada container.

Linha 4: Define a propriedade Flex do container como 1.

Linha 5: Define a propriedade JustifyContent do container como center.

Linha 6: Define a propriedade AlignItems do container como center.

Linha 8: Define a propriedade Padding do container como 25.

Linha 9: Fecha a classe container.

Linha 11: Cria uma classe chamada Text.

Linha 12: Define a propriedade FontSize do Text como 12.

Linha 13: Define a propriedade FontWeight do Text como Bold.

Linha 14: Define a propriedade textAlign do Text como Center.

Linha 15: Define a propriedade MarginBottom do Text como 17.

Linha 16: Define a propriedade Color do Text como Red.

Linha 19: Fecha a classe Text.

Linha 22: Cria uma classe chamada input.

Linha 23: Define a propriedade height do input como 40.

Linha 24: Define a propriedade BorderRadius do input como 5.

Linha 25: Define a propriedade backGroundColor do input como grey.

Linha 26: Define a propriedade MarginBottom do input como 13.

Linha 27: Fecha a classe input.

Linha 28: Fecha a constância Style.

Figura 5 - Resultado estilização CSS.

Fonte: Autoria Própria, 2025

Dessa forma, é possível visualizar a tela feita com react native estilizada com CSS, trazendo assim uma interface mais agradável de se utilizar.

2.2.3 JavaScript

Segundo Moraes (2014), o JavaScript é uma linguagem de programação de alto nível caracterizada por sua leveza, dinamismo e flexibilidade. Sua sintaxe é influenciada pela linguagem C, o que traz uma estrutura familiar aos desenvolvedores.

Figura 6 - Exemplo feito com JavaScript.

```
1 const arr = ['Morango' , 'Abacaxi', 'Maça' , 'Uva' , 'Pêssego' , 'Torta','Chocolate',
2 console.log(arr)
3
4 arr.push("Batatinha")
5 console.log(arr)
6
7 arr.unshift("Batatao")
8 console.log(arr)
9
10 const indice = arr.indexOf("Óleo")
11 console.log(indice)
12
13 const comida = arr.slice(0, 4)
14
15 const outros = arr.slice(-4)
16 console.log(arr)
17 console.log(comida)
18
19 const alimentos = comida.concat(outros, "Batata doce")
20 console.log(alimentos)
21
22
23 const elementosRemovidos = alimentos.splice(indice, 1, 'Uva Verde')
24 console.log(alimentos)
25 console.log(elementosRemovidos)
26
27 for (let indice = 0; indice < alimentos.length; indice++){
28     const elemento = alimentos[indice]
29     console.log(elemento + "se encontra na posição " + indice)
30 }
31
```

Fonte: Autoria Própria, 2025

Nesta figura temos um exemplo feito utilizando JavaScript. Este código deverá mostrar um elemento e a posição onde se encontra. De forma mais clara:

Linha 1: Cria uma constância chamada “arr” com os valores Morango, Abacaxi, Maça, Uva, Torta, Chocolate.

Linha 2: Apresenta os valores de “arr” no console.

Linha 4: Adiciona Batatinha ao final de “arr”.

Linha 5: Apresenta os valores de “arr” no console.

Linha 7: Adiciona “Batatão” ao início de arr.

Linha 8: Apresenta os valores de “arr” no console.

Linha 10: “indexOf()” procura o índice do elemento Óleo no array.

Linha 11: Salva o índice encontrado na variável índice e apresenta o valor do índice no console.

Linha 13: Pega os elementos de índice 0 até 3 (o 4 é exclusivo), salva essa fatia na variável comida.

Linha 15: Pega os últimos 4 elementos do array, salva na variável outros.

Linha 16: Apresenta os valores de arr no console.

Linha 17: Apresenta os valores de Comida no console.

Linha 19: Junta os valores de comida, outros e adiciona 'Batata doce' no final.

Linha 20: Apresenta os valores de alimentos no console.

Linha 23: Remove 1 elemento a partir do índice (obtido lá na linha 10) e insere 'Uva Verde' no lugar.

Linha 24: Apresenta os valores de alimentos no console.

Linha 25: Apresenta os valores de “elementosRemovidos” no console.

Linha 27: Um loop for que percorre o array alimentos.

Linha 28: Pega o elemento atual e seu índice.

Linha 29: Apresenta o elemento e seu índice no console.

Linha 30: Fecha o for.

2.2.4 Modelagem de dados

De acordo com Já Castro, Baião e Guizzardi (2009), o MER serve como uma etapa intermediária no projeto lógico de banco de dados, assim permitindo que ele seja uma representação conceitual independente de preocupações técnicas.

Segundo Heuser (2009), o Diagrama Entidade-Relacionamento (DER) é essencial para facilitar a comunicação entre usuários e projetistas, diminuir ambiguidades e aumentar a qualidade do projeto de banco de dados.

2.2.5 Unified Modeling Language (UML)

Segundo BOOCH (2006) a UML é uma linguagem-padrão para elaboração da estrutura de projetos de software. A UML é apenas uma linguagem e, portanto, é somente uma parte de um método para desenvolvimento de software.

UML é uma linguagem de modelagem amplamente utilizada na engenharia de software por ser versátil e aplicável a diferentes áreas. Tornou-se o padrão internacional da indústria para representar sistemas de forma clara e estruturada (GUEDES, 2011).

Para BOOCH (2006) um diagrama é uma apresentação gráfica de um conjunto de elementos, geralmente representados como um gráfico conectado de vértices e arcos. Sendo utilizados para visualizar um sistema sob diferentes perspectivas.

2.2.4.1 Diagrama de Caso de Uso

Por meio de uma linguagem simples esses diagramas possibilitam o entendimento do comportamento de um sistema sendo considerado o mais flexível e informal diagrama presente no UML (GUEDES, 2011).

Segundo BOOCH (2006), eles têm um papel central na modelagem de comportamento de um sistema demonstrando uma junção de casos de uso, atores e seus relacionamentos.

2.2.4.2 Diagrama Máquina de Estado

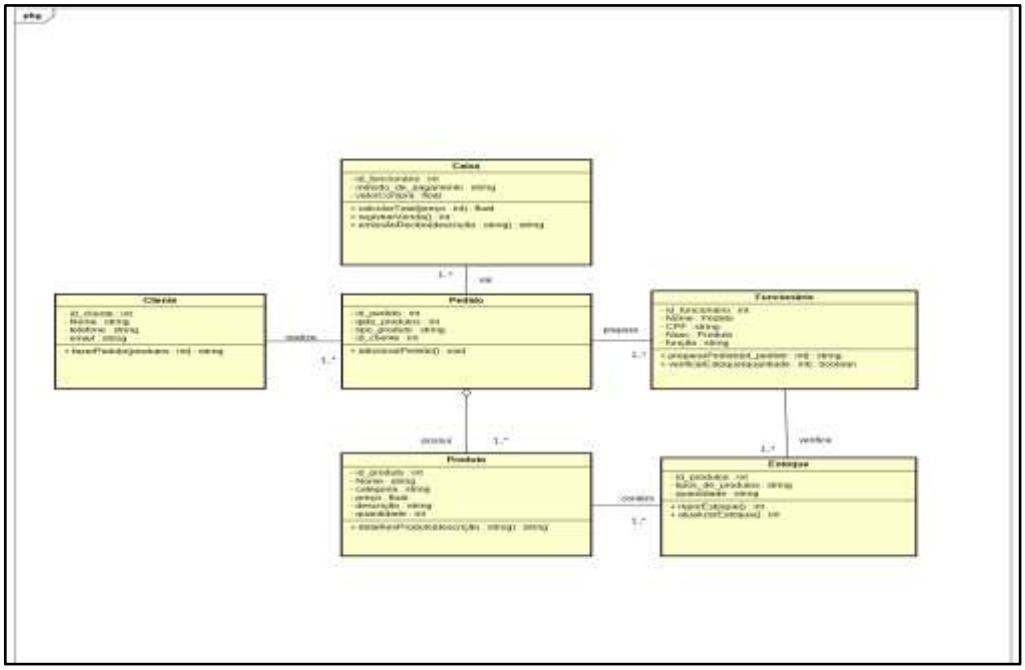
Para GUEDES (2011) esses diagramas são utilizados para definir o comportamento de um objeto em relação a acontecimentos internos e externos, representando os Estados possíveis e a transição entre esses Estados.

Os diagramas de máquina de estado demonstram o comportamento dinâmico de um único objeto por meio de Estados, transições com eventos, condições, ações associadas, Estado inicial e Estado final (BOOCH, 2006).

2.2.4.3 Diagrama de Classe

O diagrama de classes da UML foca em representar visualmente as classes de um sistema, seus atributos, métodos e as relações entre elas. Ele é fundamental por servir como base para a criação dos demais diagramas (GUEDES, 2011).

Figura 7 – Exemplo diagrama de classe



Fonte: Autoria Própria, 2025

Foram criadas 6 classes para realizar um exemplo de um sistema de padaria, que são as classes “Cliente”, “Pedido”, “Produto”, “Produto”, “Estoque”, “Funcionário” e “Caixa”.

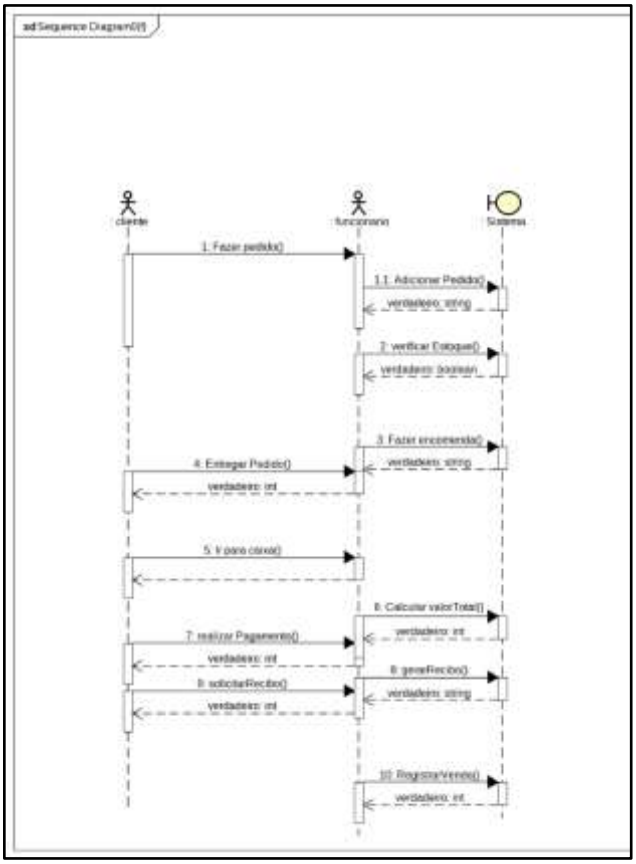
A classe “Cliente” é a primeira classe onde irá possuir informações básicas do cliente para realizar um pedido, e em caso de encomenda, algumas informações pessoais para contato, assim o cliente poderá realizar o pedido e seguir adiante. Na classe “Pedido” teremos as características do pedido feito, como por exemplo, todo pedido precisa de um ID, para saber que aquele pedido é de tal cliente, o resto são as outras informações sobre os produtos que verá adiante, já a classe “Produto” guarda as informações referentes aos produtos, como id do produto, categoria etc.

A classe “Estoque” é onde os produtos estão guardados, estoque de produtos, com as identificações de cada produto (ID), os tipos, e a quantidade, na classe “Funcionários” temos a classe de funcionários da padaria, com seus códigos, respectivas informações necessárias e suas funções, e a classe “Caixa” é onde o pagamento e registro da venda será feito, com um funcionário fixo para isso que seria o da caixa.

2.2.4.4 Diagrama de sequência

O diagrama de sequência é um diagrama comportamental que descreve a ordem dos eventos em um processo específico, sendo comum a existência de um diagrama de sequência para cada caso de uso no sistema (GUEDES, 2011).

Figura 8 – Exemplo diagrama de sequência



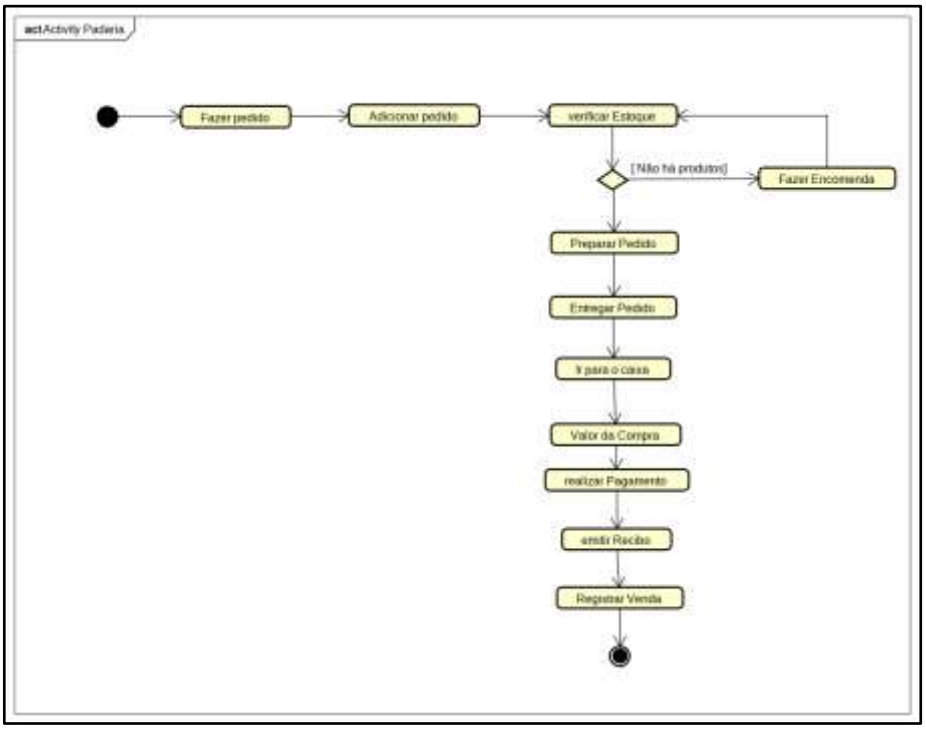
Fonte: Autoria Própria, 2025

Aqui temos o diagrama de sequência mostrando a sequência de passos feitos, onde utilizamos os atores, que irão interagir com o sistema. Como principais atores temos o cliente e o funcionário, porém é necessário entender as lifelines (linhas de vida), que são as linhas tracejadas. Essas caixinhas de texto são chamadas mensagens ou estímulos e são usadas para demonstrar a ocorrência de eventos, começando pelo enviar pedido até o registro da venda e a emissão do recibo.

2.2.4.5 Diagrama de atividade

Eles pertencem ao grupo de diagramas utilizados para representar o comportamento de um sistema e funcionam como fluxogramas, ilustrando a transição do controle entre diferentes atividades dentro de um processo (BOOCH, 2006).

Figura 9 – Exemplo diagrama de atividades



Fonte: Autoria Própria, 2025

Aqui temos o diagrama de atividades, sequenciando os passos, essa bolinha preta no começo representa o início do processo, seguindo pelos passos junto com as setas, o cliente faz o pedido, o pedido é adicionado, verificam o estoque, se não tiver, a encomenda será feita, caso tenha produtos, o pedido será realizado, entregado, e o cliente será direcionado para o caixa, onde irá ver o valor da compra, realizar o pagamento, a venda será registrada e emitida com o recibo, assim finalizando esse processo.

2.2.6 Node JS

Node.js é um ambiente de execução de código Javascript no lado do servidor, open-source e multiplataforma, permitindo usar Javascript para criar conteúdo web dinâmico antes do carregamento da página no navegador (DUARTE JÚNIOR, 2020).

De acordo com FERREIRA (2015) Node.js também possui o seu próprio gerenciador de pacotes ele se chama NPM, que se tornou popular a partir da versão 0.6.0 do Node.js que se integrou ao instalador, tornando-se gerenciador default.

Para DUARTE JÚNIOR (2020) Node.js por ser assíncrona entenda que cada requisição ao Node.js não bloqueia o processo, atendendo a um volume absurdamente grande de requisições ao mesmo tempo mesmo sendo single thread.

2.2.7 SQLite

Segundo Comachio (2011), o SQLite é o sistema de gerenciamento de banco de dados da plataforma Android, armazenado os dados localmente, ou seja, no próprio dispositivo sendo ideal para aplicações que exigem rapidez e operações offline.

Para Gomes (2022) O SQLite é essencial para o funcionamento offline dos aplicativos móveis, mantendo o app funcional mesmo sem conexão, armazenando dados estruturados de forma segura sendo multiplataforma e amplamente suportado.

Figura 10 - Exemplo SQLite

```python
import sqlite3

conexao = sqlite3.connect("meu_banco.db")

cursor = conexao.cursor()

cursor.execute("""
CREATE TABLE IF NOT EXISTS usuarios (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL
)
""")

cursor.execute("INSERT INTO usuarios (nome, email) VALUES (?, ?)",
               ("Maria Silva", "maria@email.com"))
cursor.execute("INSERT INTO usuarios (nome, email) VALUES (?, ?)",
               ("João Souza", "joao@email.com"))

conexao.commit()

cursor.execute("SELECT * FROM usuarios")
usuarios = cursor.fetchall()

print("Usuários cadastrados:")
for usuario in usuarios:
    print(usuario)

conexao.close()
```

Fonte: Autoria Própria, 2025

Linhas 1 a 5: Cria um arquivo de banco de dados local chamado meu_banco.db

Linhas 7 a 13: Cria a tabela usuarios se ela ainda não existir

Linhas 15 a 20: Insere novos registros e salva as alterações

Linhas 22 e 23: Busca todos os dados da tabela

Linhas 24 a 27: Exibe todos os dados da tabela

Linha 29: Fecha a conexão corretamente

2.2.8 Internet das Coisas (IoT)

Para Monteiro e Santos (2020) IoT é definido como um conjunto de tecnologias que conecta o mundo físico ao mundo digital, utilizando sensores e sistemas embarcados permitindo monitorar, controlar e automatizar tarefas de forma remota e inteligente.

A IoT é caracterizada pela interconexão de dispositivos físicos com a internet, permitindo que esses objetos enviem e recebam dados sem intervenção humana direta (GODOI; ARAÚJO, 2020).

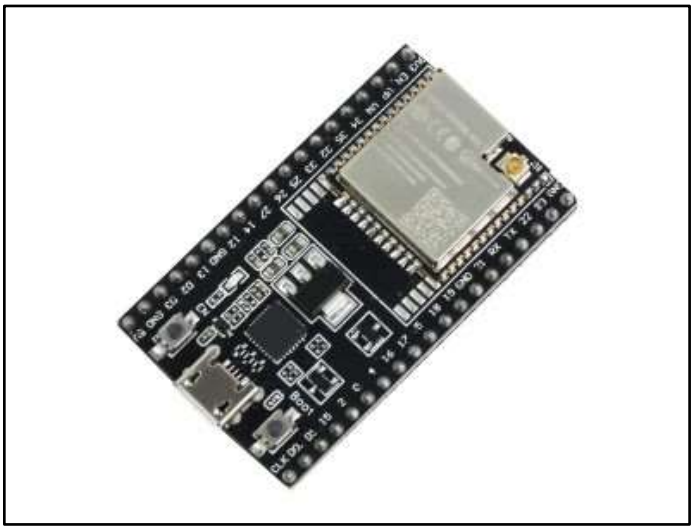
Parafraseando SOARES (2017) IoT não é apenas uma tecnologia, mas um fenômeno social e econômico, que exige preparo multidisciplinar para ser utilizado de forma sustentável e segura.

2.2.8.1 ESP32

Segundo Morais (2023, p. 4-15) o ESP32 é um microcontrolador de alta performance se destaca por sua versatilidade e capacidade de comunicação sem fio integrada, incluindo Wi-Fi e Bluetooth.

O ESP32 é ideal para aplicações de Internet das Coisas (IoT). Equipado com um processador dual-core, oferecendo alto desempenho para tarefas complexas em projetos de monitoramento remoto (SOUZA; 2023, p. 6).

Figura 11 - ESP32.



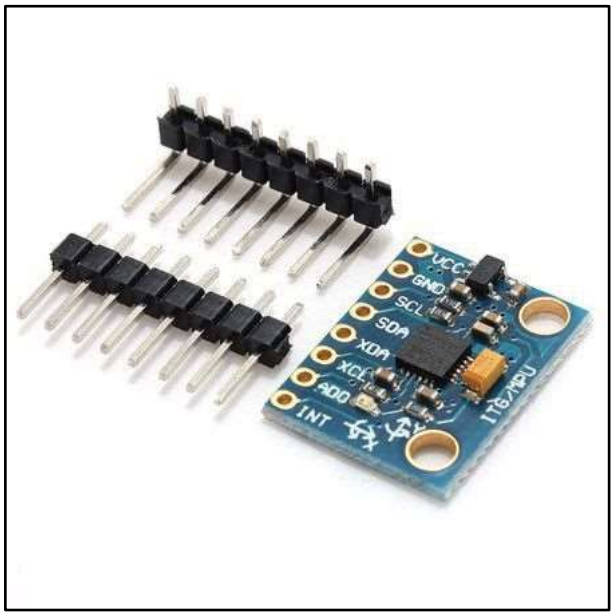
Fonte: USINAINFO, 2024

2.2.8.2 Acelerômetro e Giroscópio

Silva (2017, p. 125-135) apresenta os sensores MEMS, destacando os acelerômetros como dispositivos que convertem movimento físico em sinais elétricos para sistemas embarcados.

Consoante a Moraes (2015), o giroscópio é essencial para a medição da velocidade angular em sistemas embarcados, convertendo forças em sinais elétricos, e sua integração com acelerômetros permite maior precisão para detectar movimento.

Figura 12 - Acelerômetro e Giroscópio.



Fonte: (Arducore, 2025)

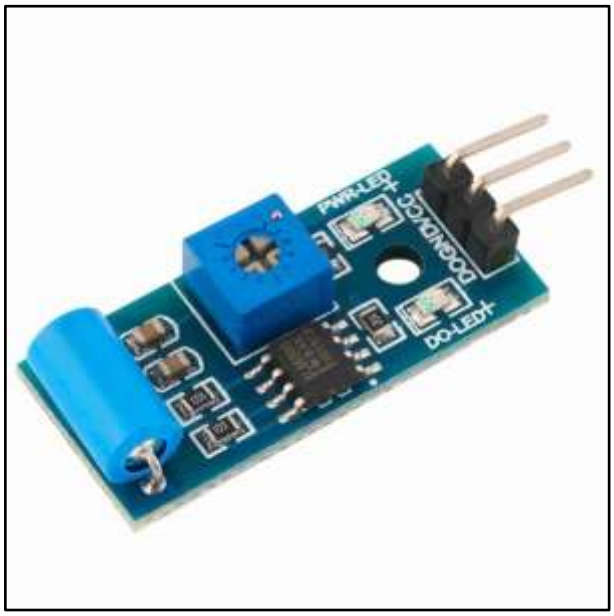
2.2.8.3 Sensor Vibratório SW-420

De acordo com Cunha (2016, p. 98-102), o sensor vibratório SW-420 é utilizado para a detecção de vibração e impacto. Trata-se de um sensor piezoelétrico combinado com um circuito comparador que envia um sinal digital.

Para Costa (2018) a facilidade de integração do sensor com microcontroladores o torna uma escolha popular para sistemas de alarme, monitoramento de máquinas e outras aplicações que requerem resposta rápida a choques e vibrações.

Quando o limiar de movimento é ultrapassado, sendo amplamente empregado em sistemas de monitoramento e segurança (CUNHA; 2016, p. 98-102).

Figura 13 - Sensor Vibratório SW-420.



Fonte: (Casa da Robótica, 2025)

2.2.8.4 Sensor de Distância HC-SR04

Parafraseando NAKATANI, GUIMARÃES e MACHADO NETO (2014) o HC-SR04, é amplamente utilizado em projetos de automação e robótica para medições de distância.

Já segundo Teixeira Júnior (2022), trata-se de uma alternativa de baixo custo para medições de distância, com potencial de aplicação em ambientes variados, embora fatores como a temperatura possam interferir em sua precisão.

O sensor opera emitindo pulsos ultrassônicos e medindo o tempo de retorno após a reflexão em um obstáculo, permitindo calcular a distância com base na velocidade do som (NAKATANI, GUIMARÃES e MACHADO NETO, 2014).

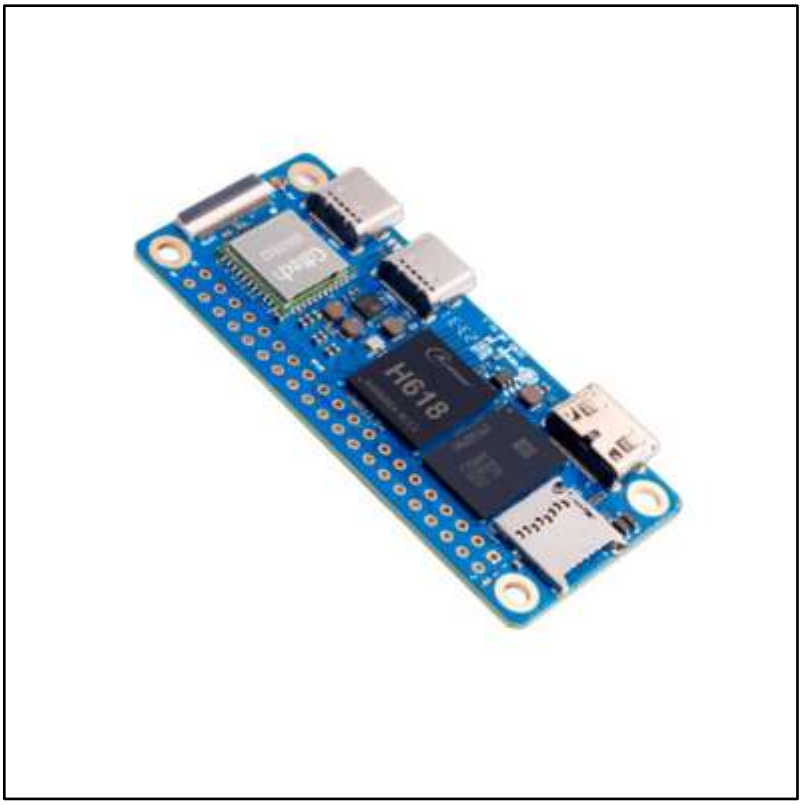
Figura 14 - Sensor de Distância HC-SR04.

Fonte: (MakerHero, 2025)

2.2.8.5 Orange PI 2w

A Orange Pi Zero 2W é uma placa de desenvolvimento compacta, inspirada na Raspberry Pi Zero 2W sendo uma ótima opção acessível e versátil para automação, servidores compactos e sistemas embarcados (SILVA, 2025).

Figura 15 - Orange PI Zero 2W



Fonte: (Curto Circuito, 2025)

2.2.8.6 Open-Source Computer Vision Library

De acordo com MARENGONI e STRINGHINI (2010) o OpenCV é uma biblioteca de código aberto desenvolvida com o objetivo de facilitar o desenvolvimento de aplicações de visão computacional e processamento de imagens.

Para MILANO & HONORATO (2023) o sistema envia os dados coletados para um servidor web, permitindo o acesso remoto às informações em tempo real. Essa integração facilita a gestão e o controle do tráfego em ambientes urbanos.

A visão computacional é a área da computação que busca fazer com que máquinas interpretem imagens e vídeos, utilizando câmeras e algoritmos para simular a percepção humana (BARELLI, 2018).

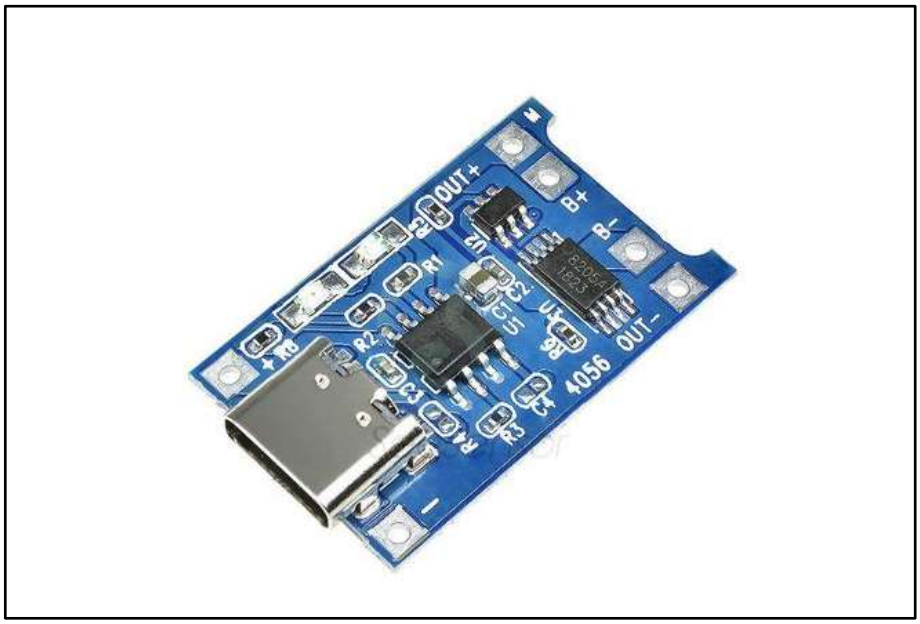
2.2.8.7 Impressão 3D

De acordo com LIRA (2021) O objeto 3d é inicialmente projetado em um ambiente virtual utilizando dimensões reais. Depois a modelagem é convertida em uma malha 3D e exportada em formato STL, destacando que esse processo é financeiramente vantajoso.

2.2.8.8 TP4056

Consoante XAVIER (2022) o módulo TP4056 é responsável pelo carregamento seguro de uma bateria Li-Ion, possuindo um circuito de proteção contra sobrecarga e descarga profunda, assim diminuindo as chances de a bateria sofrer danos.

Figura 16 - TP4056



Fonte: Lemona Eletronics, 2025

2.2.8.9 Baterias 18650

Para ALVES (2022) as baterias de lítio entregam melhor eficiência energética, menor peso e maior número de ciclos de carga/descarga comparadas às baterias tradicionais, assim sendo economicamente mais vantajosa para várias aplicações.

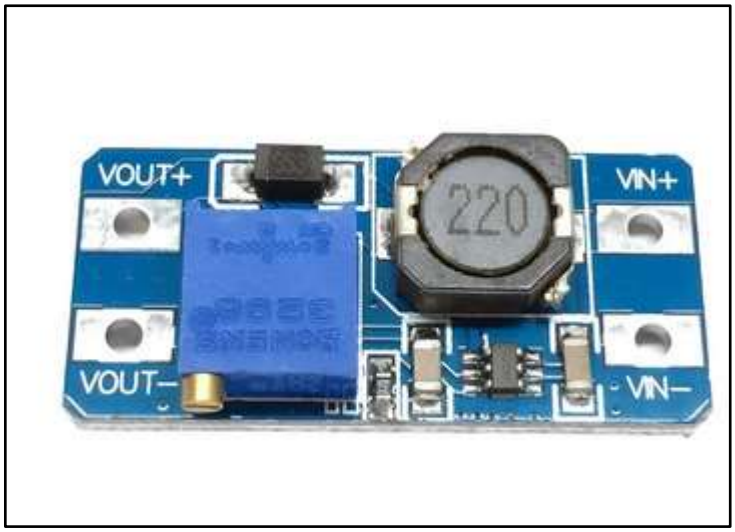
Figura 17 - Baterias 18650

Fonte: Eletroinfo Cia, 2025

2.2.8.10 MT3608

Segundo XAVIER (2022) o módulo conversor step up MT3608 é responsável por elevar a tensão da bateria 3,0 V para 5 V, podendo permitir uma saída estável até 28 V, além de ser eficiente em sistemas portáteis e funcionar bem com baterias Li-Ion.

Figura 18 – MT3608



Fonte: Mercado Livre, 2025

2.2.9 Python

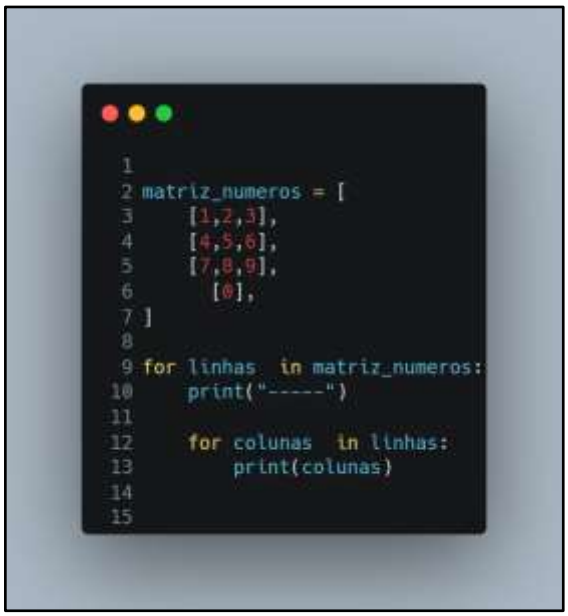
Python é uma linguagem de programação, de alto nível e com sintaxe simples e clara, que facilita o aprendizado para iniciantes e ao mesmo tempo oferece recursos avançados para programadores experientes (SWARUP; PILLAI, 2018).

Essa linguagem destaca-se por sua capacidade de permitir automatizar tarefas do dia a dia de forma simples, mesmo para quem não é programador, ajudando a economizar tempo e esforço (SWEIGART, 2016).

De acordo com Ribeiro e Santos (2021), Python é a linguagem ideal para iniciantes, pois combina simplicidade com potência, facilitando o aprendizado da programação para pessoas de diferentes áreas.

Para SWARUP e PILLAI (2018), um grande diferencial da linguagem é sua vasta biblioteca padrão e enorme acervo de bibliotecas de terceiros disponíveis na comunidade, que atendem em diversas áreas de desenvolvimento.

Figura 19 - Exemplo feito em Python.



Fonte: Autoria Própria, 2025

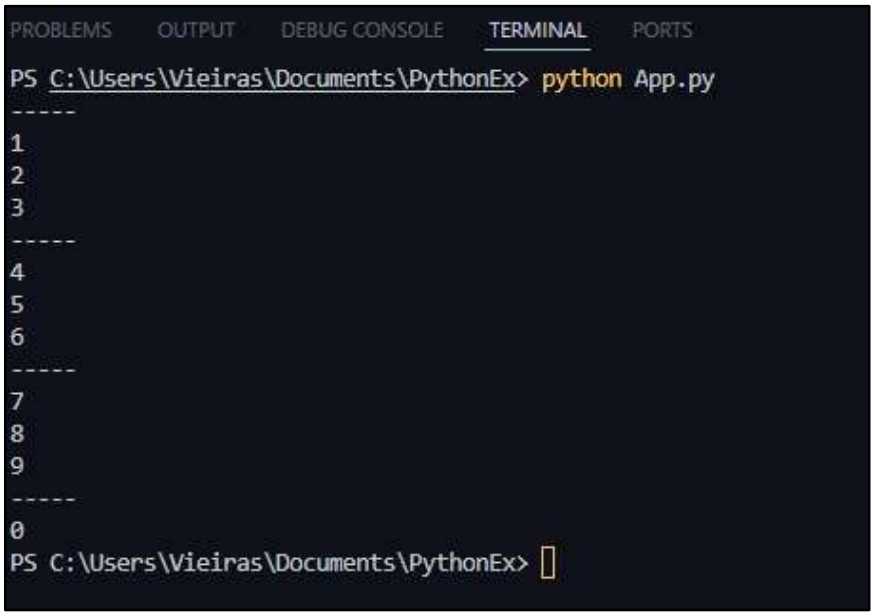
O código acima mostra uma matriz sendo criada na linguagem python, exemplificando melhor:

Linhas de 2 a 7: foram criadas matrizes ou arrays bidimensionais, o nome da variável é matriz_numeros e dentro dela temos 4 listas, ou seja, 4 linhas, porém não possui o mesmo número de colunas em cada uma delas;

Linhas 9 e 10: O for percorre cada elemento da lista, e vai pegar à variável linha para assumir a cada iteração uma das sublistas, dentro do for apenas existe um print;

Linhas 12 e 13: Veja que este for está dentro do anterior e este 2 for percorre cada elemento da sublista linhas. Assim a cada iteração colunas é um número da sublista e print irá imprimir cada número.

Figura 20 - Resultado exemplo feito em python.



Fonte: Autoria Própria, 2025

Assim, o resultado acima será apresentado no terminal do programa usado para criar o código.

2.2.10 Haar Cascade

O Haar Cascade é implementado no OpenCV e utiliza classificadores em cascata. O processo envolve a extração de características simples a aplicação dessas características em uma estrutura de decisão eficiente (Marengoni e Stringhini, 2010).

Segundo Castro (2021), essas características são padrões de contraste entre áreas claras e escuras de uma imagem. O uso dessas características torna o algoritmo Haar Cascade eficiente para detecção rápida e precisa de objetos em tempo real.

Figura 21 - Exemplo haar cascade

Fonte: Autoria Própria, 2025

Pré-Requisitos:

- Instalação do Python (install python)
- Verificar a versão do python (python – version)
- instalar pip gerenciador de pacotes do python (install pip)
- instalar a biblioteca OpenCV (pip install open-CV python)
- Criação do arquivo main.py
- importando a biblioteca openCV2

Após isso é necessário criar uma variável que contenha o classificador com a imagem .xml. a variável frontalface_xml carrega a imagem.

Depois, transformamos a imagem em tons de cinza pois o haar funciona melhor assim.

A variável faces detecta o rosto na imagem utilizando:

- Gray
- scaleFactor para reduzir o tamanho da imagem em 10% em cada escala
- minNeighbors para definir quantos vizinhos cada retângulo candidato deve ter para ser mantido
- minSize tamanho minimo do objeto a ser detectado pela aplicação.

- Enquanto isso um loop em retângulo para ser identificado em volta do rosto. A saída esperada deve ser a imagem e o rosto na imagem marcado.

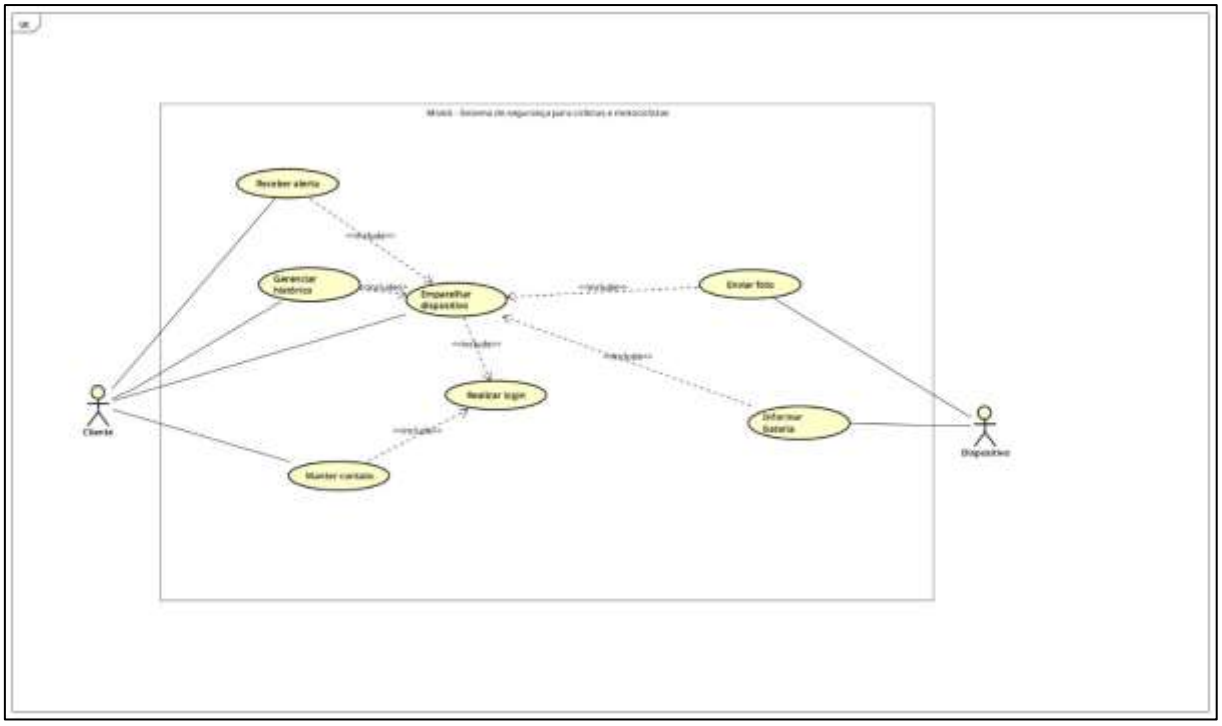
3 DESENVOLVIMENTO

O seguinte capítulo se trata do processo de desenvolvimento do sistema Mivick, descrevendo o processo de documentação utilizando a linguagem UML, junto com a prototipação de ambos os dispositivos e a demonstração das telas do aplicativo.

3.1 Diagrama de Caso de Uso

Logo abaixo, pode-se visualizar e analisar o caso de uso do sistema Mivick, onde temos os seguintes atores: Cliente e IoT. O caso de uso contém sete elipses e cada um deles demonstra a ação que os atores podem fazer no sistema.

Figura 22 - Caso de uso Mivick



Fonte: Autoria Própria, 2025

3.1.1 Documentação de Caso de Uso

Para a concepção da documentação de caso de uso, foi criado, tendo em base o modelo UML, os requisitos funcionais, que representam uma funcionalidade do próprio sistema; requisitos não funcionais, representando características que são requisitadas pelo usuário, geralmente se tratando de características visuais e de desempenho; e as regras de negócio, se tratando de (alguma coisa)

Requisitos funcionais:

- RF01: Detectar impactos bruscos com sensor SW-420;
- RF02: Detectar quedas e mudanças bruscas de inclinação com acelerômetro giroscópio MPU6050;
- RF03: Medir distância de objetos próximo com sensor ultrassônico;
- RF04: Enviar foto quando um acidente for detectado;
- RF05: Notificar contatos de emergência via aplicativo, e-mail ou WhatsApp;
- RF06: Registrar eventos críticos em banco de dados;
- RF07: Atuar automaticamente se múltiplos sensores confirmarem um risco;
- RF08: Permitir a configuração de contatos de emergência via app;
- RF09: Carregar bateria 18650 automaticamente via TP4056 quando conectado

  a fonte externa;
- RF10: Fornecer 5V ao ESP32-CAM usando MT3608 a partir da bateria 3.5V;
- RF11: Apresentar histórico e informações coletados do dispositivo no aplicativo;
- RF12: Editar usuários e contatos.

Requisitos não funcionais:

- RFN01: Os sensores devem funcionar corretamente mesmo com vibrações;
- RFN02: O tempo entre a detecção do evento e o envio de alerta deve ser inferior a 5 segundos;
- RFN03: O dispositivo deve funcionar por várias horas em modo de espera;
- RFN04: O sistema deve continuar operando mesmo com a falha em um dos sensores;
- RFN05: O dispositivo deve ser leve e de tamanho reduzido;
- RFN06: Deverá ser fácil substituir a bateria dos dispositivos ou recarregá-la;
- RFN07: A configuração inicial via app deve ser simples;

- RFN08: O dispositivo deve ser resistente a água/respingos;
- RFN09: O app deve ser rápido no envio de alerta;
- RFN10: A experiência no app deve focar em simplicidade e segurança.

Regras de negócio:

- É necessário realizar um cadastro ou login para acessar o sistema;
- Ao menos 1 contato deve ser cadastrado no app;
- O sistema não envia alerta com menos de 5% de carga na bateria;
- O sistema só ativa a câmera se estiver conectado;
- Cada dispositivo só pode ter até 3 contatos de emergência configurados;
- O sistema entra em modo de economia de energia após 60 minutos de Inatividade;
- Não é permitido recarregar as baterias enquanto o sistema estiver operando com carga alta para evitar superaquecimento;
- A distância mínima para colisão é de 50 cm; se um objeto for detectado a menos que isso, o sistema registra como quase colisão.

Tabela 1 - Realizar login

Nome do Caso de Uso	**Realizar login**
Ator Principal	Cliente
Atores Secundários	
Resumo	Acesso do cliente por meio de suas credenciais
Pré-condições	O cliente deve ter um cadastro prévio no sistema
Pós-condições	O cliente está autenticado e logado no sistema
	Cenário Principal
Ações do Ator	**Ações do Sistema**
1. O cliente seleciona a opção de “Fazer login” na interface do app	
	2. O app apresenta um formulário de login
3. O cliente informa seu nome de usuário e senha	
4. O cliente clica no botão “Entrar”	
	5. O app verifica se as credenciais estão corretas
	Cenário Alternativo
Ações do Ator	**Ações do Sistema**
	1. Credenciais inválidas, se o email ou senha estiverem incorretas, o sistema exibirá uma mensagem de erro

Fonte: Autoria Própria, 2025.

Tabela 2 - Emparelhar dispositivo

Nome do Caso de Uso	**Emparelhar dispositivo**
Ator Principal	Cliente
Atores Secundários	IoT
Resumo	O cliente inicia a conexão entre o aplicativo e o IoT
Pré-condições	O cliente deve realizar um login no sistema
Pós-condições	Dispositivo conectado, agora envia dados ao aplicativo
Cenário Principal	
Ações do Ator	**Ações do Sistema**
1. O cliente inicia o processo de pareamento no aplicativo 2. O cliente busca e se conecta com o dispositivo IoT	3. O dispositivo IoT se comunica com o aplicativo
Cenário Alternativo	
Ações do Ator	**Ações do Sistema**

Fonte: Autoria Própria, 2025.

Tabela 3 - Receber alerta

Nome do Caso de Uso	**Receber alerta**
Ator Principal	Cliente
Atores Secundários	App
Resumo	O app enviará um alerta para os contatos cadastrados em caso de acidente
Pré-condições	O cliente deve ter um cadastro prévio no sistema
Pós-condições	Emissão do alerta para todos os contatos
Cenário Principal	
Ações do Ator	**Ações do Sistema**
	1. Verifica se há conexão com os dispositivos 2. Sistema identifica um acidente 3. IoT envia informações ao app 4. App faz a emissão do alerta
5. Cliente recebe o alerta em seu dispositivo	
Cenário Alternativo	
Ações do Ator	**Ações do Sistema**

Fonte: Autoria Própria, 2025.

Tabela 4 - Enviar foto

Nome do Caso de Uso	**Enviar foto**
Ator Principal	IoT
Atores Secundários	
Resumo	O dispositivo captura uma imagem e envia para o aplicativo
Pré-condições	Dispositivo conectado, duas funcionalidades do IoT ativadas
Pós-condições	É verificado se a imagem é um veículo ou não
	Cenário Principal
Ações do Ator	**Ações do Sistema**
	1. Em caso de impacto e distância, sistema captura imagem 2. Sistema envia foto para o aplicativo
3. Cliente pode visualizar foto capturada	
Restrições/Validações	1. A foto só será tirada caso 2 ou mais fases acontecerem

Fonte: Autoria Própria, 2025.

Tabela 5 - Manter contato

Nome do Caso de Uso	**Manter contato**
Ator Principal	Cliente
Atores Secundários	
Resumo	Permite que o cliente adicione contatos em caso de emergência
Pré-condições	O cliente deve estar logado
Pós-condições	O contato é salvo e armazenado no banco de dados
	Cenário Principal
Ações do Ator	**Ações do Sistema**
1. Cliente cadastra algum contato próximo	
	2. Sistema armazena contato e apresenta os contatos atuais
Restrições/Validações	1. Ao menos um contato deve ser cadastrado no sistema

Fonte: Autoria Própria, 2025.

Tabela 6 - Informar bateria

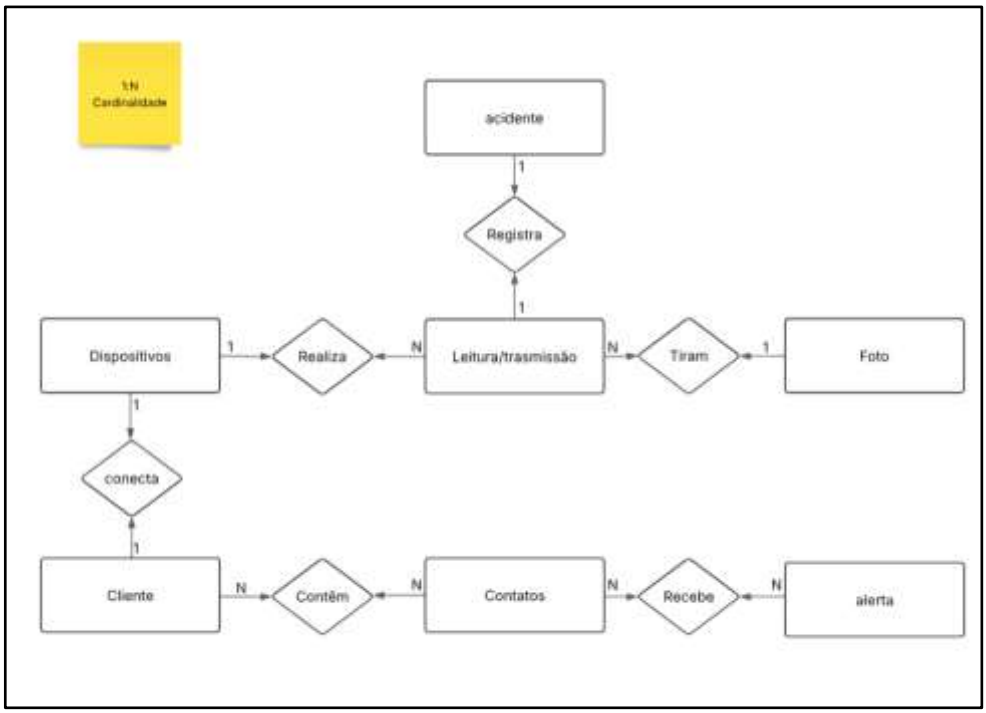
Nome do Caso de Uso	**Informar bateria**
Ator Principal	IoT
Atores Secundários	
Resumo	O IoT irá enviar constantemente o nível de bateria dos dispositivos
Pré-condições	Conexão estabelecida
Pós-condições	Envio do nível de bateria
	Cenário Principal
Ações do Ator	**Ações do Sistema**
	1. Verificar conexão entre dispositivos
	2. Enviar nível de bateria

Fonte: Autoria Própria, 2025.

3.2 Diagrama Entidade Relacionamento

Na figura abaixo, constata-se o DER da Mivick, que mostra a estrutura do banco de dados do sistema de maneira simplificada e resumida, podendo, com base nele, elaborar os diagramas que virão a seguir.

Figura 23 - DER do sistema Mivick

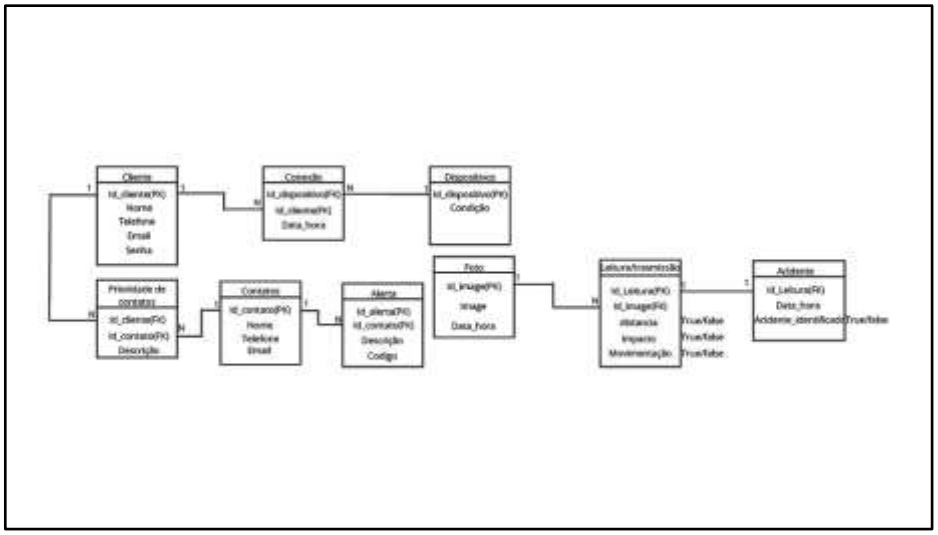


Fonte: Autoria Própria, 2025

3.3 Modelo Entidade Relacionamento

Subsequentemente, observa-se o modelo entidade relacionamento do projeto, um modelo baseado na estrutura das tabelas do banco de dados, tal qual suas entidades e relacionamento.

Figura 24 – MER do sistema Mivick

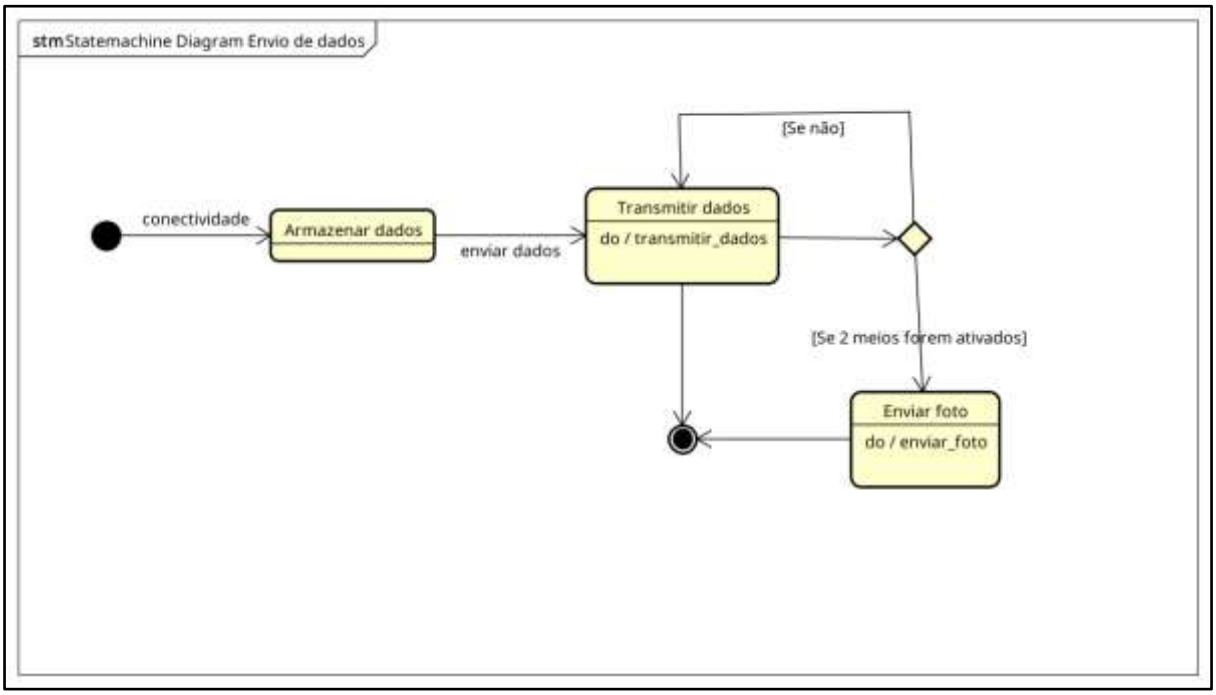


Fonte: Autoria Própria, 2025

3.4 Diagrama Máquina de Estado

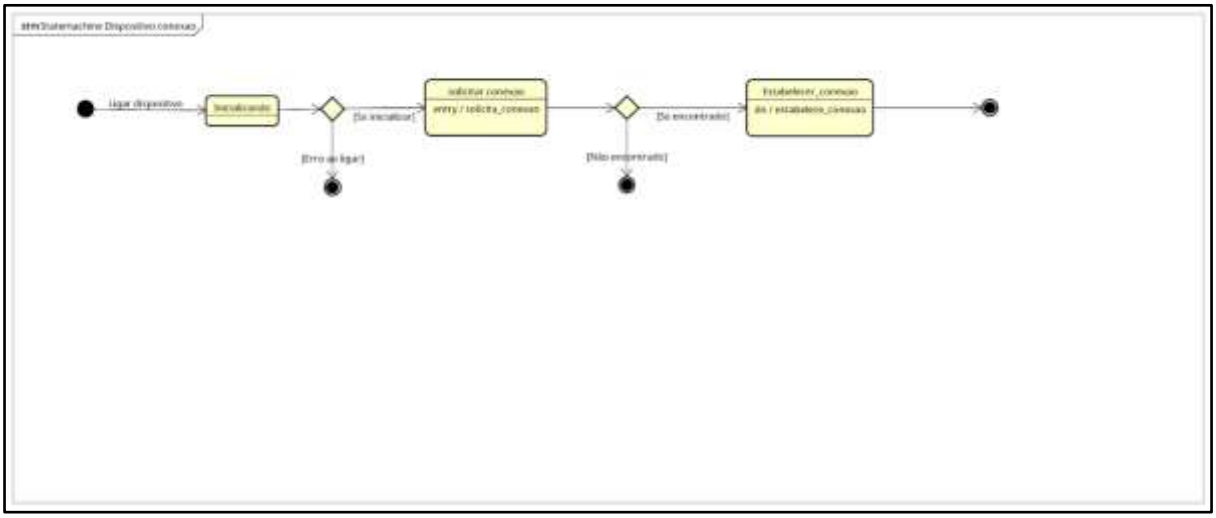
Os diagramas máquina de estado descrevem quando ocorre uma mudança de estado para cada fase em nosso sistema. Ao todo, foram feitos 2 diagramas, um para o estado de envio de dados e outro para a conexão com o dispositivo.

Figura 25 - Diagrama máquina de estado envio de dados



Fonte: Autoria Própria, 2025

Figura 26 - Máquina de estado conexão do dispositivo

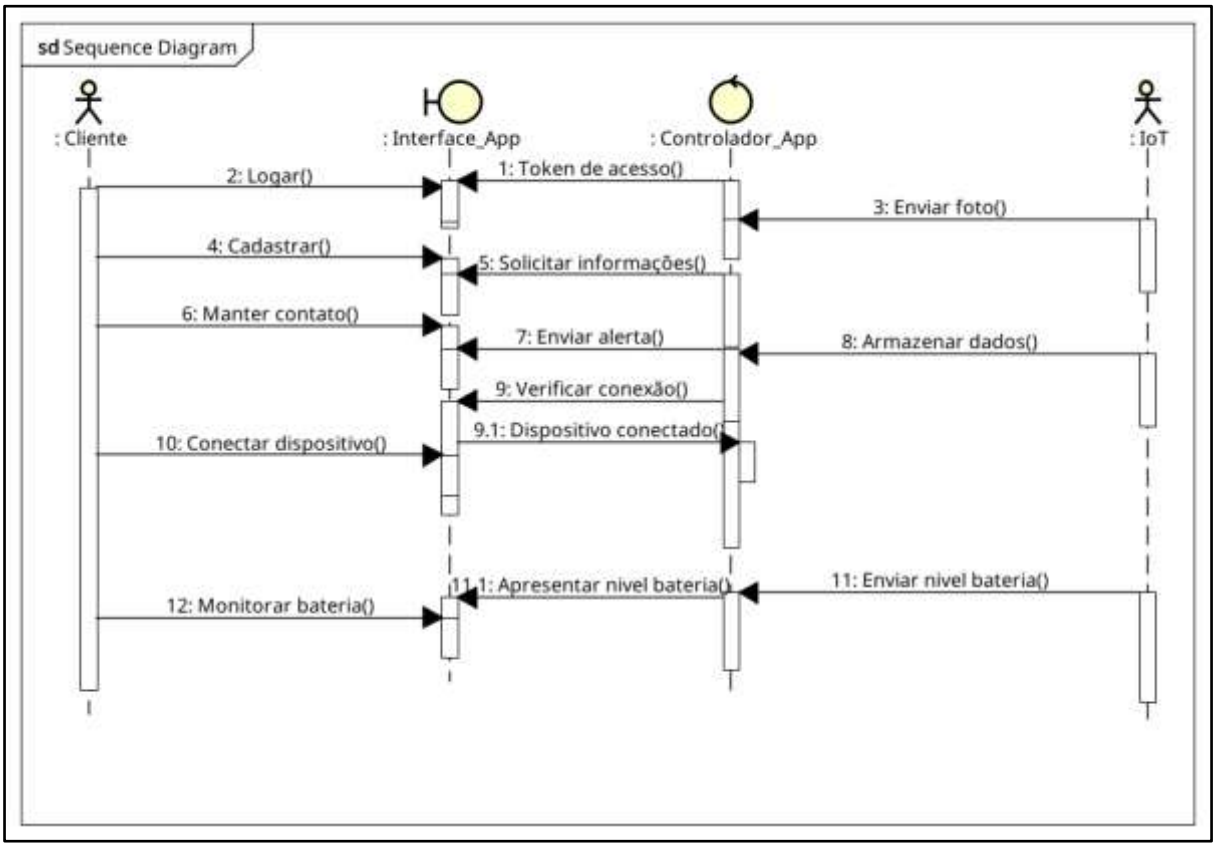


Fonte: Autoria Própria, 2025

3.5 Diagrama de Sequência

Conseguinte, temos os diagramas de sequência do sistema, que tem o objetivo principal de enfatizar a ordem temporal da troca de mensagens entre objetos. Esse sistema possui apenas um diagrama de sequência

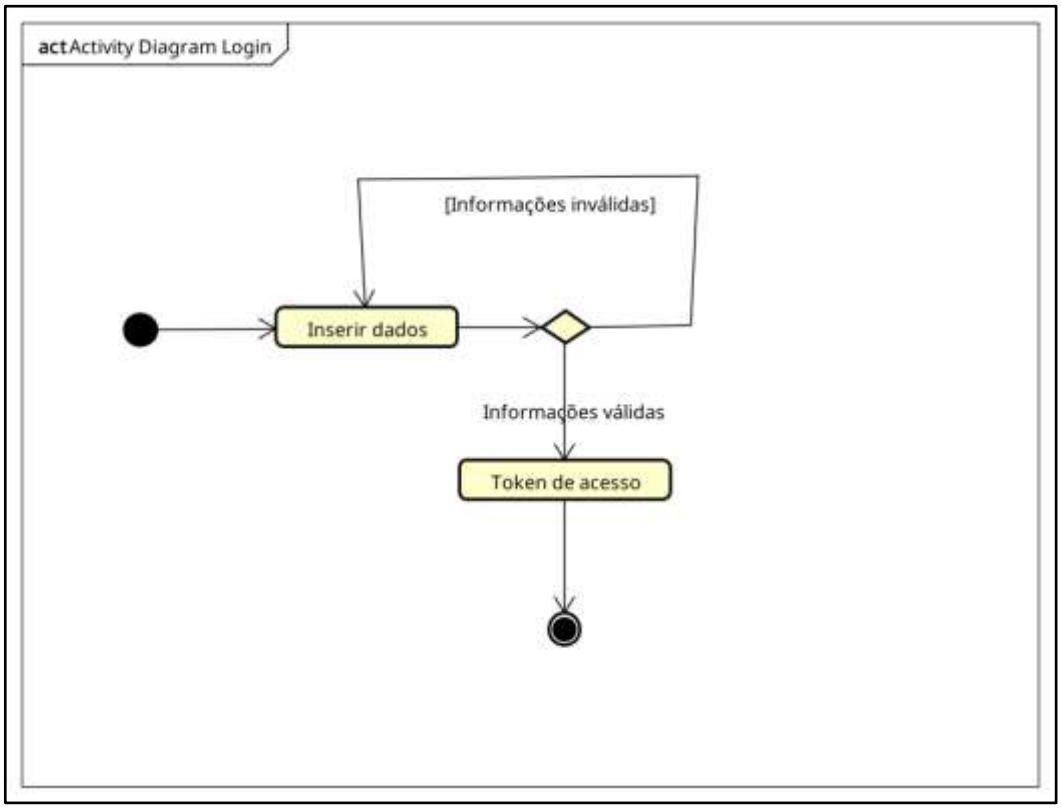
Figura 27 - Diagrama de sequência do sistema



3.6 Diagrama de Atividade

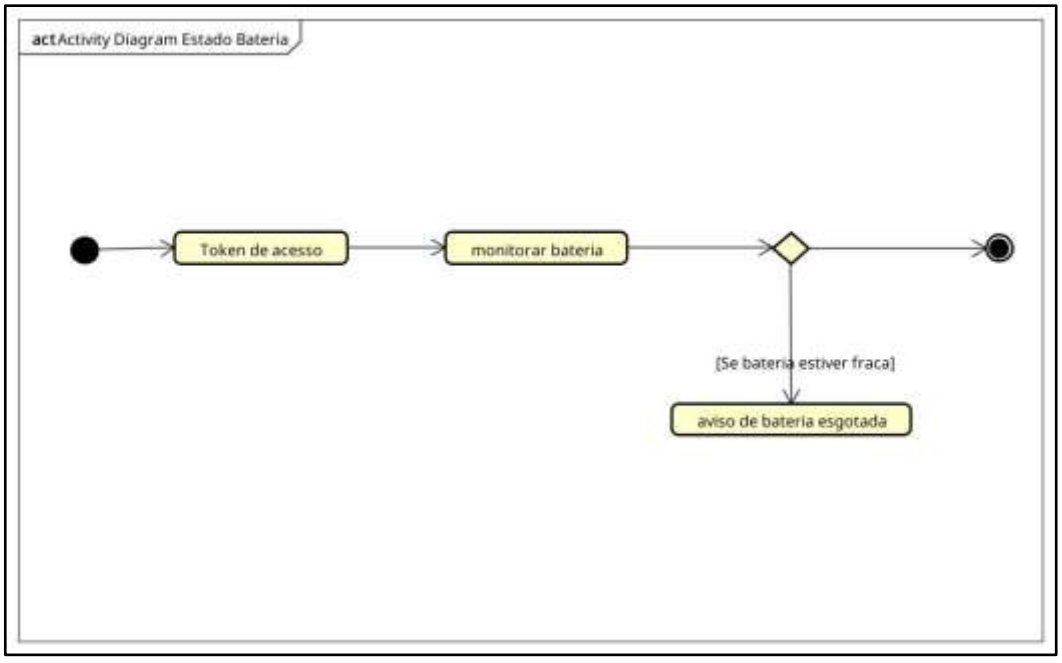
Os diagramas de atividade descrevem o fluxo de controle de uma atividade para a outra, detalhando as ações passo a passo. As imagens abaixo se trata dos 9 diagramas de atividade que demonstram cada ação que o sistema possui.

Figura 28 - Diagrama de atividades Login



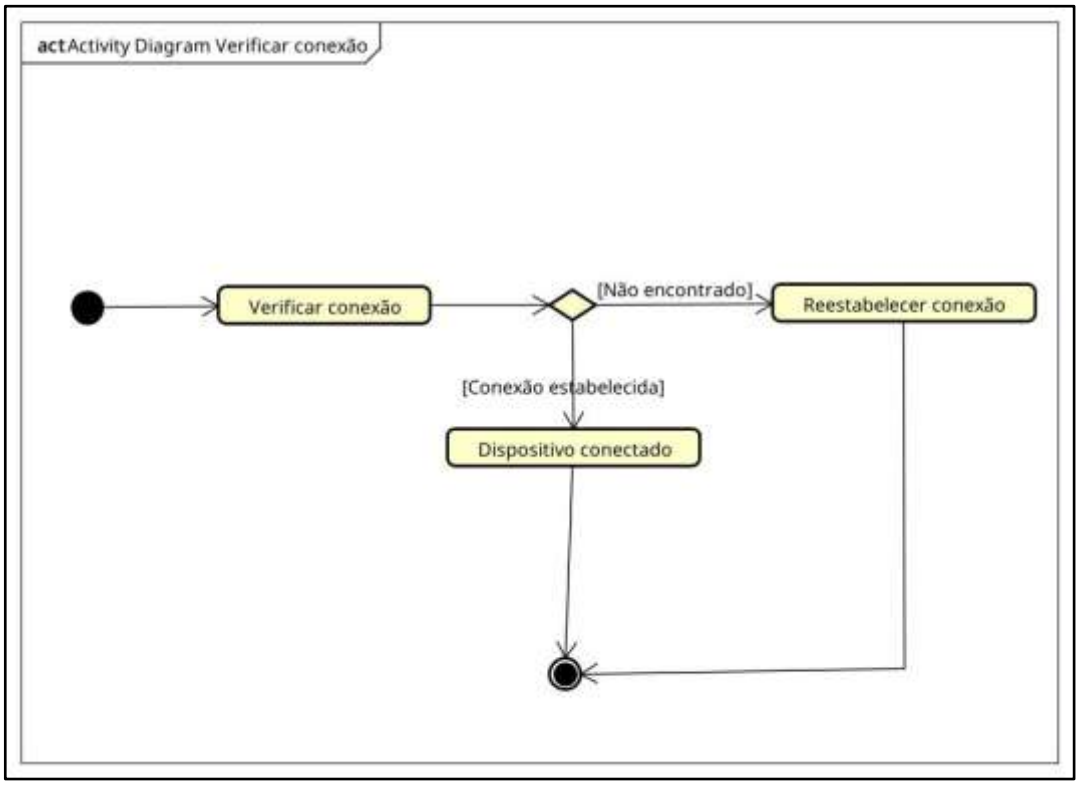
Fonte: Autoria Própria, 2025

Figura 29 - Diagrama de atividades estado da bateria



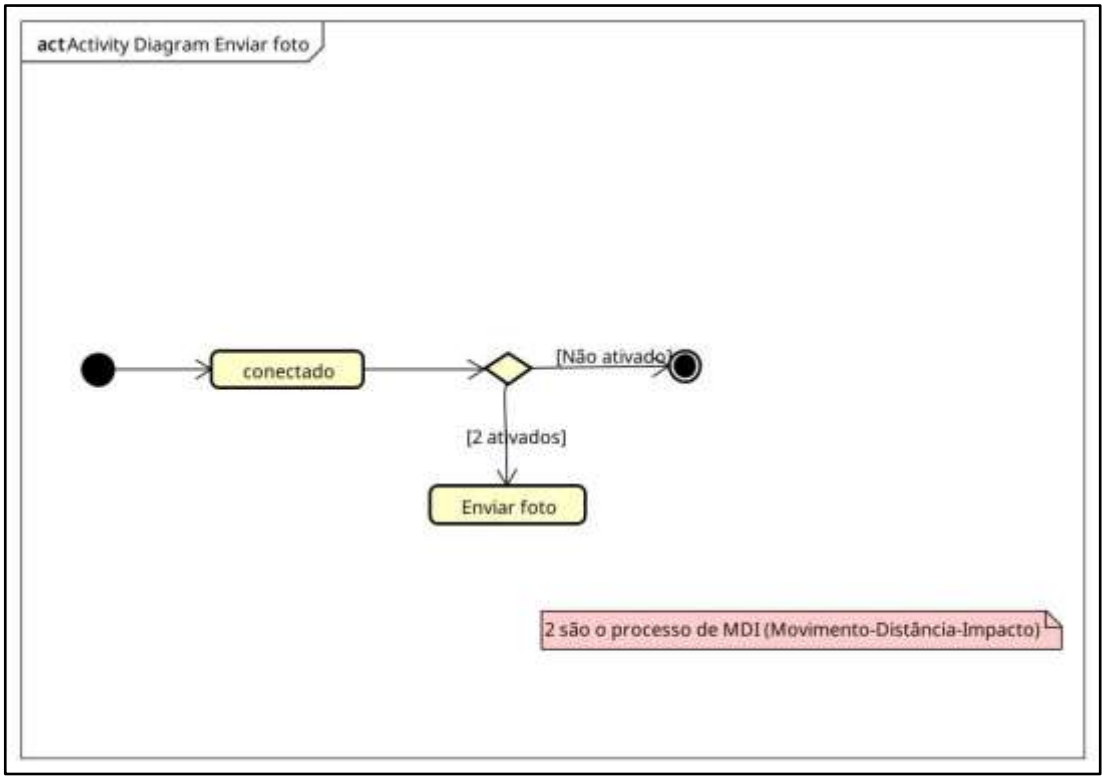
Fonte: Autoria Própria, 2025

Figura 30 - Diagrama de atividades verificar conexão



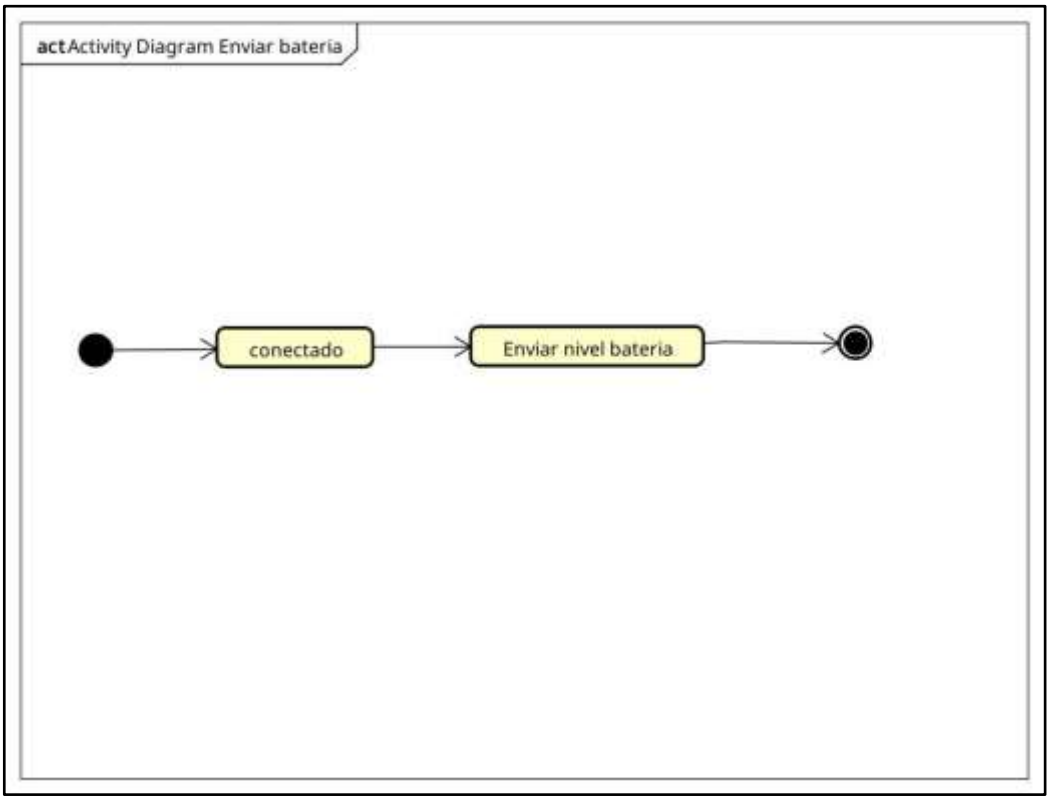
Fonte: Autoria Própria, 2025

Figura 31 - Diagrama de atividade enviar foto



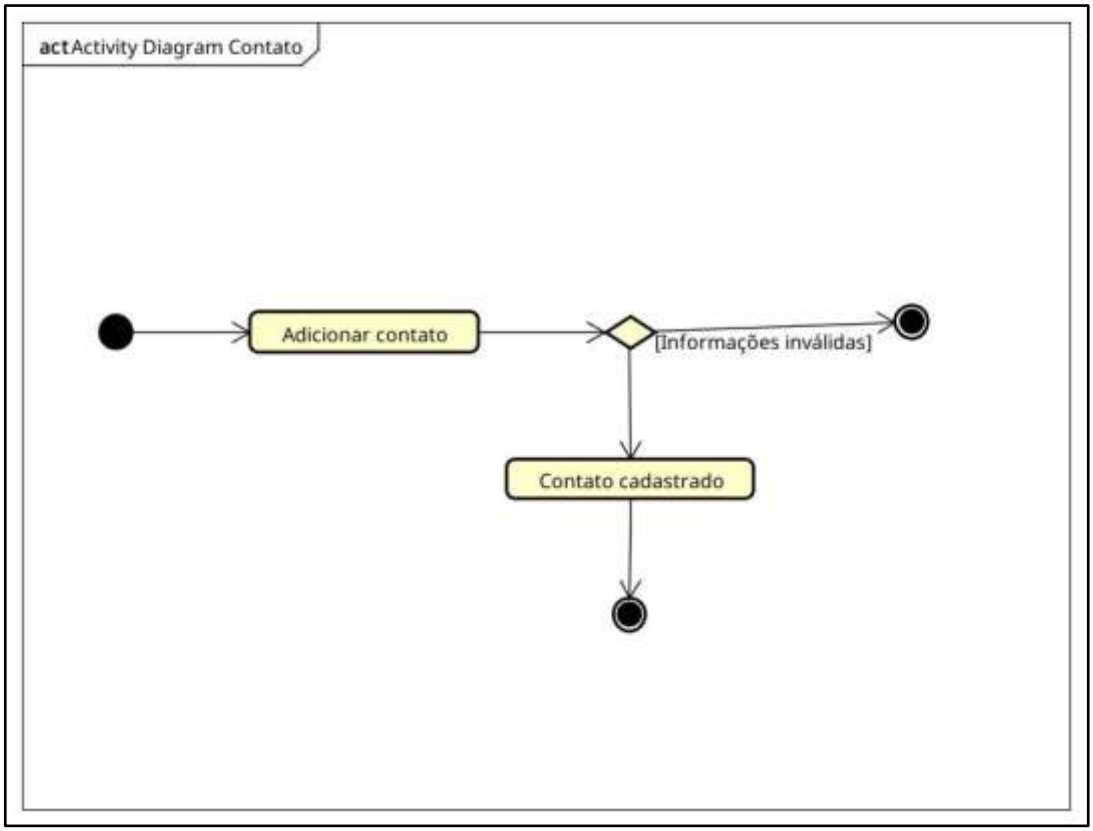
Fonte: Autoria Própria, 2025

Figura 32 - Diagrama de atividades enviar bateria



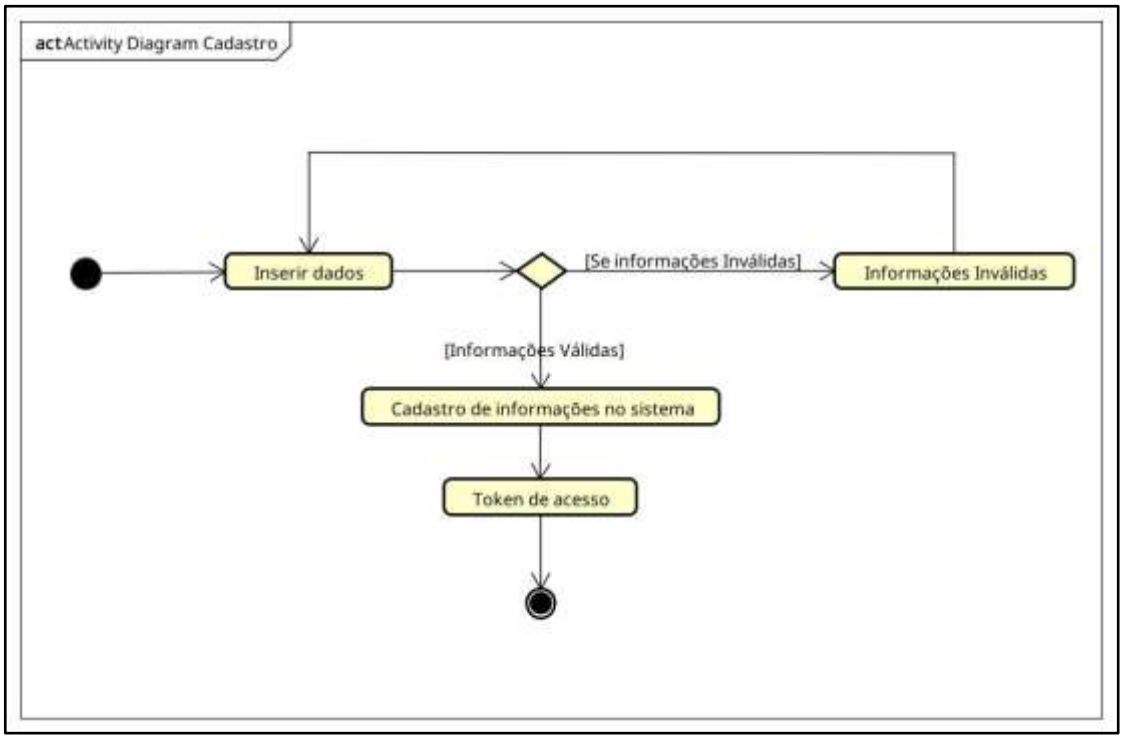
Fonte: Autoria Própria, 2025

Figura 33 - Diagrama de atividades contato



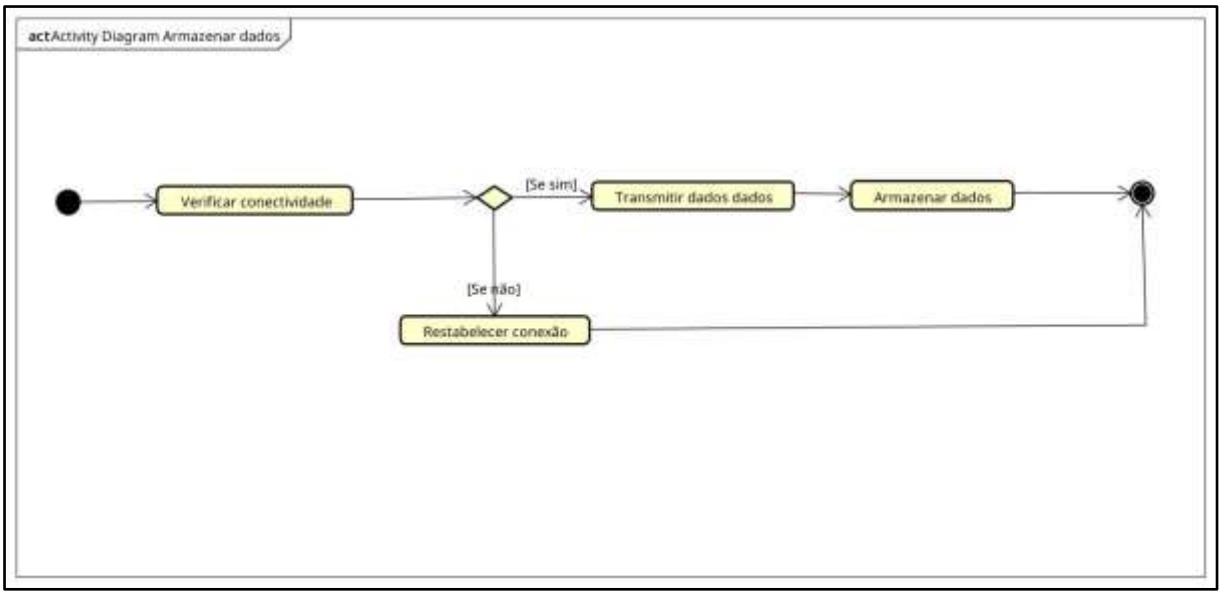
Fonte: Autoria Própria, 2025

Figura 34 - Diagrama de atividades cadastro



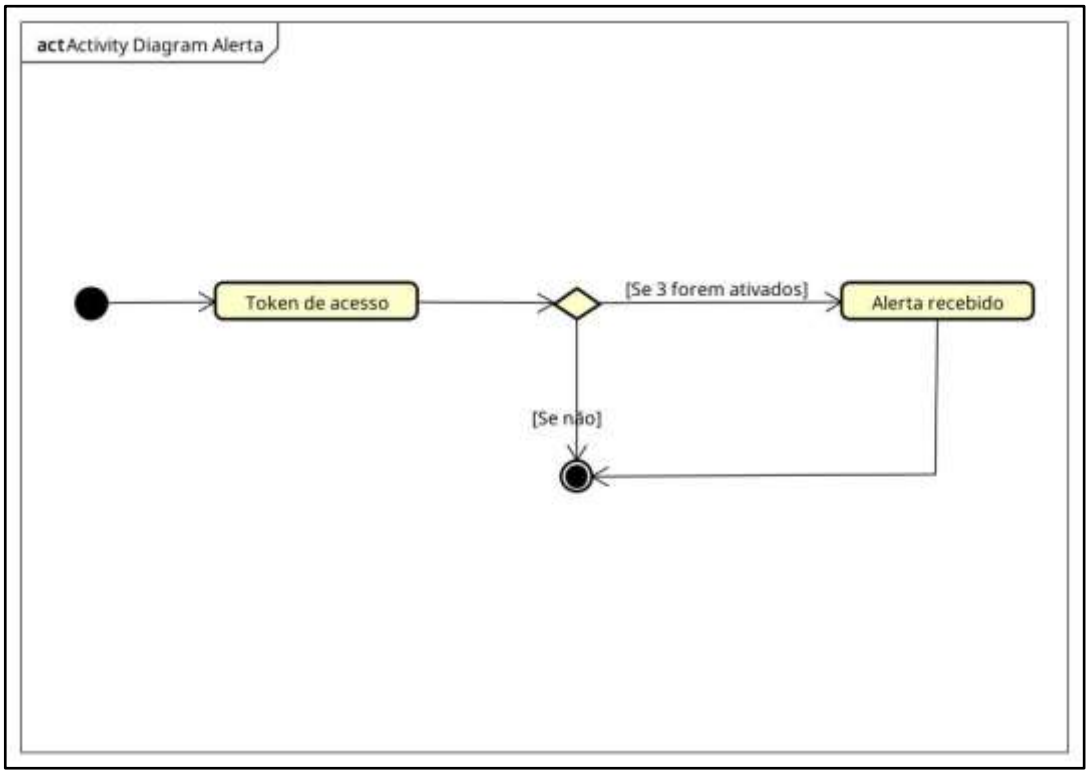
Fonte: Autoria Própria, 2025

Figura 35 - Diagrama de atividades armazenar dados



Fonte: Autoria Própria, 2025

Figura 36 - Diagrama de atividades alerta

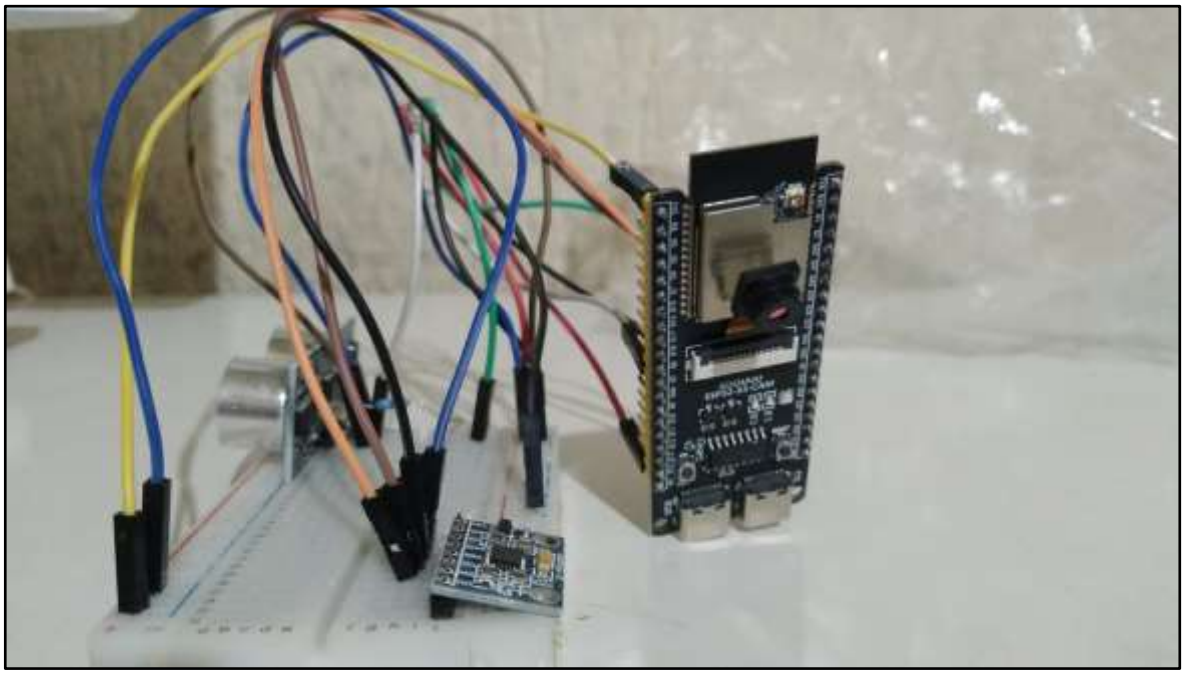


Fonte: Autoria Própria, 2025

3.7 Montagem do dispositivo

Neste tópico, será apresentada a montagem dos dois dispositivos que compõem o sistema, junto com o protótipo da case e seu modelo 3D. A princípio, foi feito um protótipo do circuito apenas para testar as funcionalidades de cada sensor, o mesmo protótipo serve para os dois dispositivos.

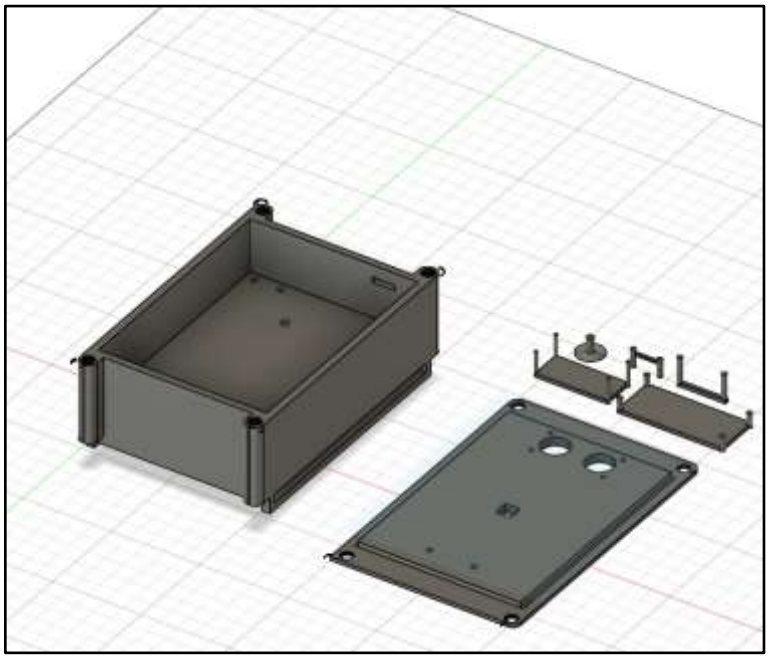
Figura 37 - Protótipo do dispositivo



Fonte: Autoria Própria, 2025

Esse protótipo foi criado apenas como um teste para criar os dois dispositivos do sistema. Com ele, podemos ter uma base de como os sensores e os dispositivos em si funcionará.

Figura 38 - Modelo 3D da case



Fonte: Autoria Própria, 2025

Com base no protótipo, pode-se ser criado o modelo 3d do projeto, que conta com a case, uma tampa e cinco suportes, todos adaptados para encaixar os

dispositivos e mantê-los presos e firmes dentro da caixa para conseguir uma maior resistência e evitar que alguma peça importante quebre.

Abaixo, a imagem mostra a case finalizada e imprimida com cada parte do dispositivo encaixada em seu devido lugar, assim sendo o protótipo final do projeto.

Figura 39 - Protótipo final do sistema

Fonte: Autoria Própria, 2025

3.8 Prototipação das páginas do aplicativo

Para o aplicativo do sistema Mivick, foi desenvolvido o protótipo da interface gráfica no modelo de wireframe de alta fidelidade, que representa o visual de cada página, podendo assim, ter noção de cada elemento gráfico que compõe cada uma das 10 páginas da aplicação mobile.

3.8.1 Elementos visuais

Antes de desenvolver o visual do aplicativo, fora primeiramente pensado nos elementos visuais, como a fonte de cada texto, paleta de cores, logo e estilo de cada componente.

Para iniciar o protótipo do layout, a seguinte paleta de cores foi escolhida:

Figura 40 - Paleta de cores do aplicativo

Fonte: Autoria Própria, 2025

Essas cores foram escolhidas para trazer mais conforto a navegação do usuário, sendo cores que remetem a alerta e segurança, assunto principal de nosso projeto.

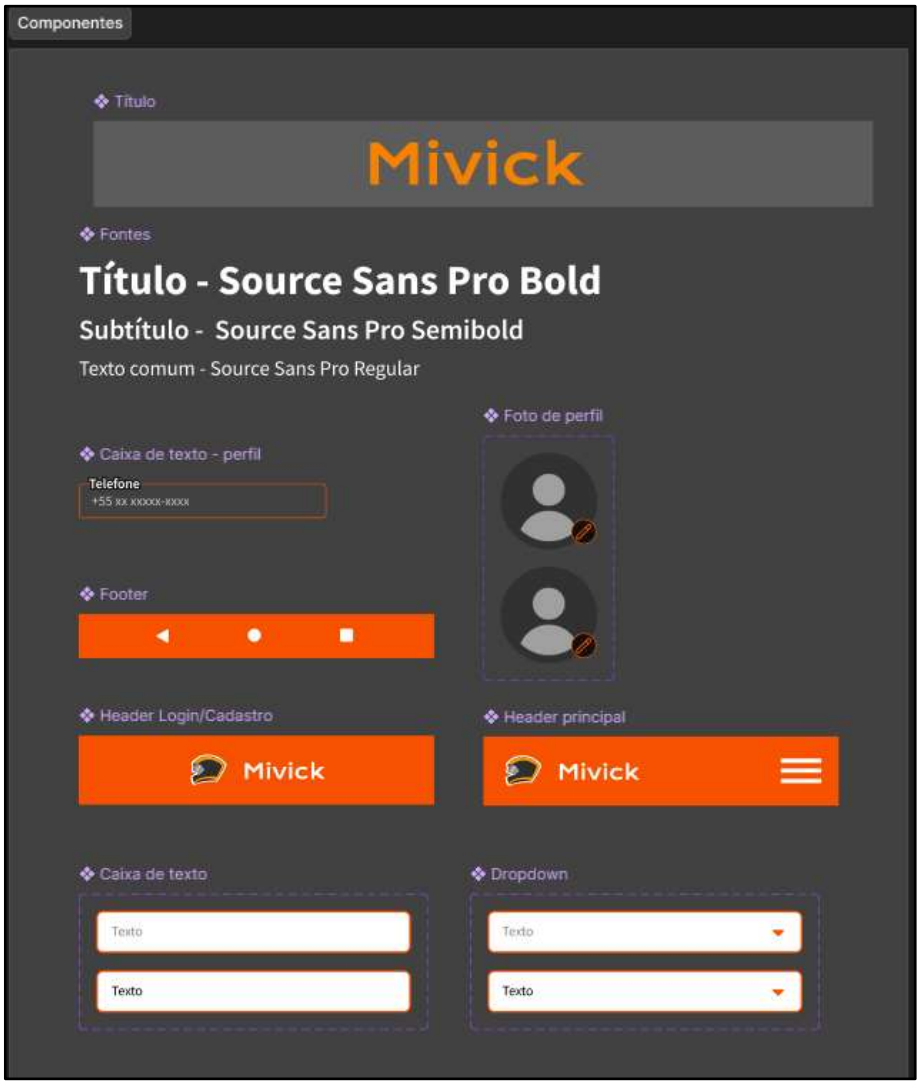
Para dar uma marca ao sistema, também fora criado o logotipo, representando um capacete, que pode ser símbolo de segurança no trânsito, e um sinal que representa os sensores de alerta do sistema.

Figura 41 - Logo da Mivick

Fonte: Autoria Própria, 2025

A imagem a seguir mostra alguns dos componentes utilizados para fazer o wireframe, componentes esses que se trata do título do aplicativo, fontes utilizadas, componentes de texto, headers, footer e as variantes de foto de perfil.

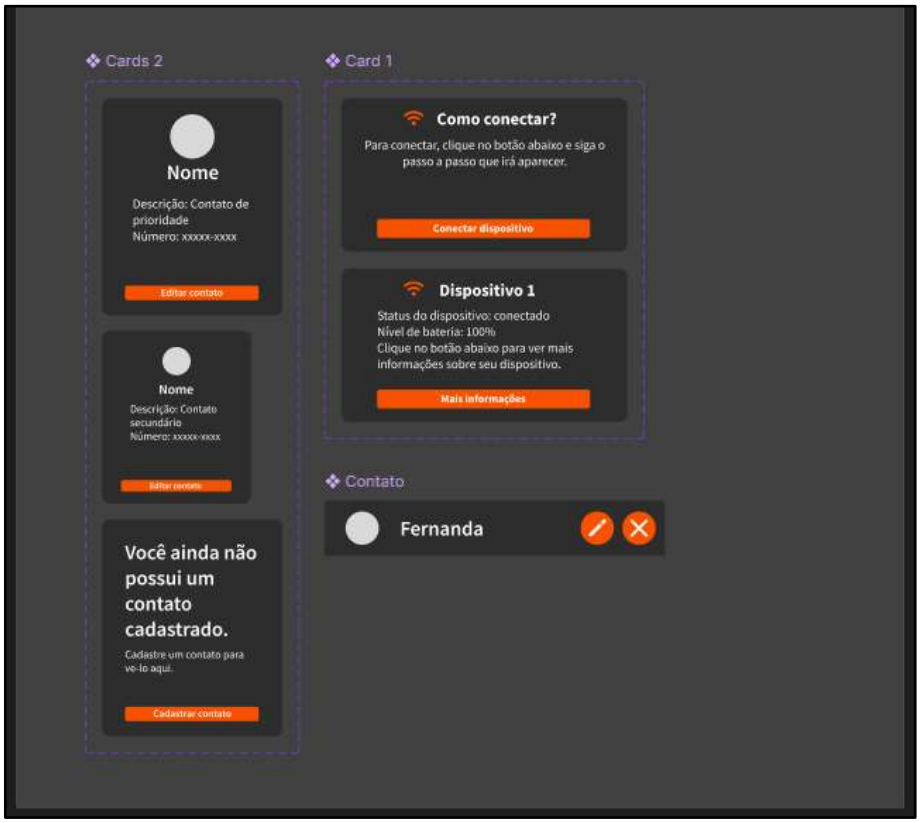
Figura 42 - Componentes iniciais do wireframe



Fonte: Autoria Própria, 2025

Logo abaixo, outros componentes que também foram usados na prototipação do aplicativo.

Figura 43 - Cards do aplicativo



Fonte: Autoria Própria, 2025

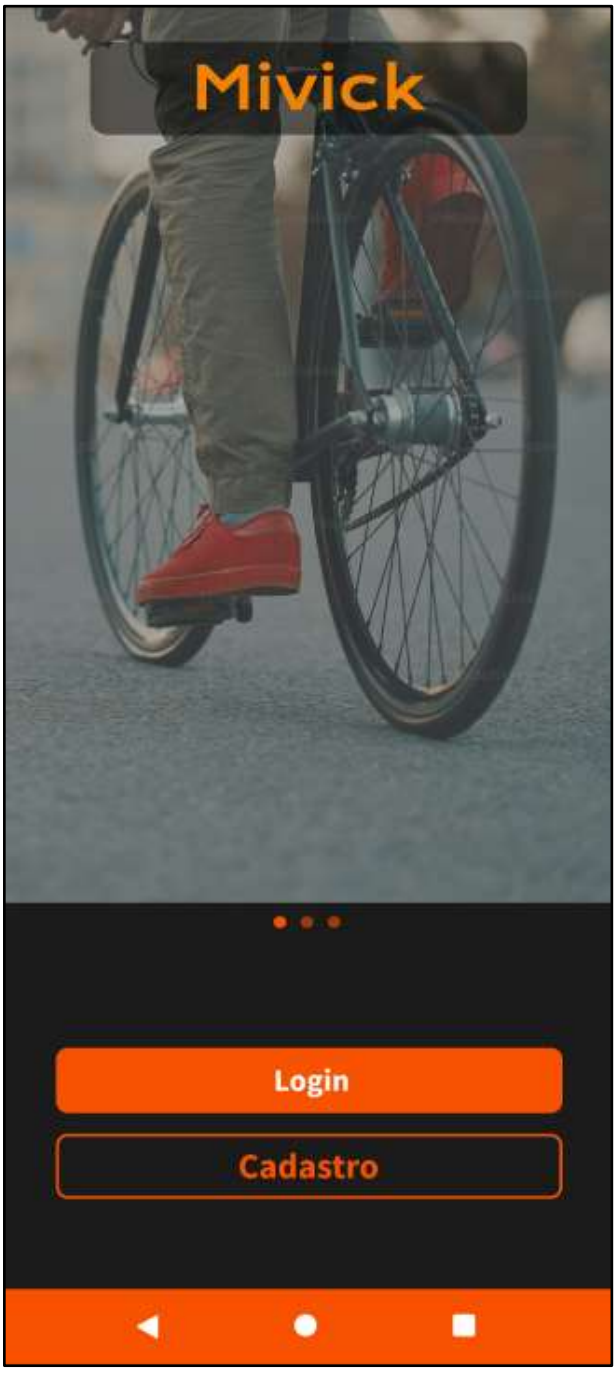
Os componentes acima se trata dos cards utilizados em certas páginas da aplicação mobile, sendo as variações dos cards de contatos cadastrados, do dispositivo conectado e de editar contatos.

3.8.2 Wireframes

Os wireframes são a base do desenvolvimento visual de uma aplicação, com base nisso, utilizando as cores e os elementos descritos acima, pode-se ser projetado as páginas do aplicativo, que faz conexão com os dispositivos e mostra informações de ambos os dispositivos, dos contatos cadastrados no sistema e do histórico de acontecimentos do usuário.

A primeira página é a que aparece assim que o usuário entra no aplicativo, sendo ela composta por um carrossel de imagens que mostra imagens referentes ao tema do aplicativo, o nome do sistema e botões para fazer login e cadastro.

Figura 44 - Página inicial



Fonte: Autoria Própria, 2025

O segundo wireframe mostra a interface de login, onde o usuário pode colocar suas informações, como nome e senha, para entrar no aplicativo. O usuário também tem a escolha de realizar o login com o Google, assim entrando com a conta ligada diretamente ao seu email. O aviso abaixo leva aos termos de uso do aplicativo, sendo necessário aceitá-los para poder realizar o login.

Figura 45 - Tela de login

Fonte: Autoria Própria, 2025

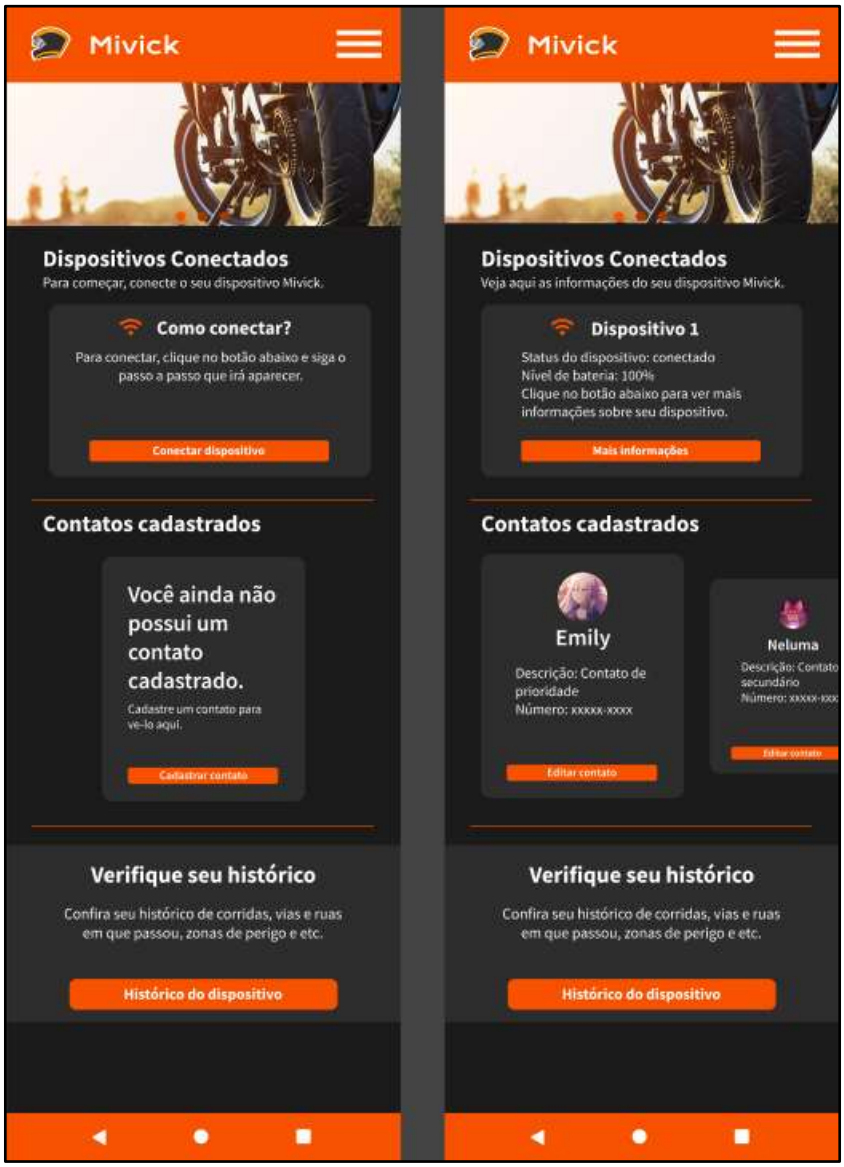
Caso o usuário não possua uma conta, clicando no botão de cadastro na tela inicial ele será direcionado para a interface de cadastro, onde é necessário colocar informações como o nome, telefone, email, senha e confirmar a senha anteriormente escolhida. Nesta página, o usuário também tem a escolha de fazer cadastro com o google, assim, facilitando a entrada no aplicativo. Assim como no login, é necessário confirmar se concorda com os termos de uso da aplicação.

Figura 46 - Tela de cadastro

Fonte: Autoria Própria, 2025

Após realizar o login, o usuário é direcionado para a interface principal do aplicativo, a chamada home. Nela, encontramos o header principal, que possui um menu sanduíche – usado para acessar outras partes do aplicativo – e um carrossel de imagens, mostrando conteúdos com o foco do projeto. Logo abaixo, como conectar o dispositivo e os contatos cadastrados no sistema do app, ambos possuindo variações caso o usuário não tenha conectado um dispositivo ou cadastrado um contato.

Figura 47 - Variações da home



Fonte: Autoria Própria, 2025

A tela abaixo se trata do cadastro de contato, que pode ser acessada através do card na tela home de uma opção no menu sanduíche, assim, mostrando a seguinte tela:

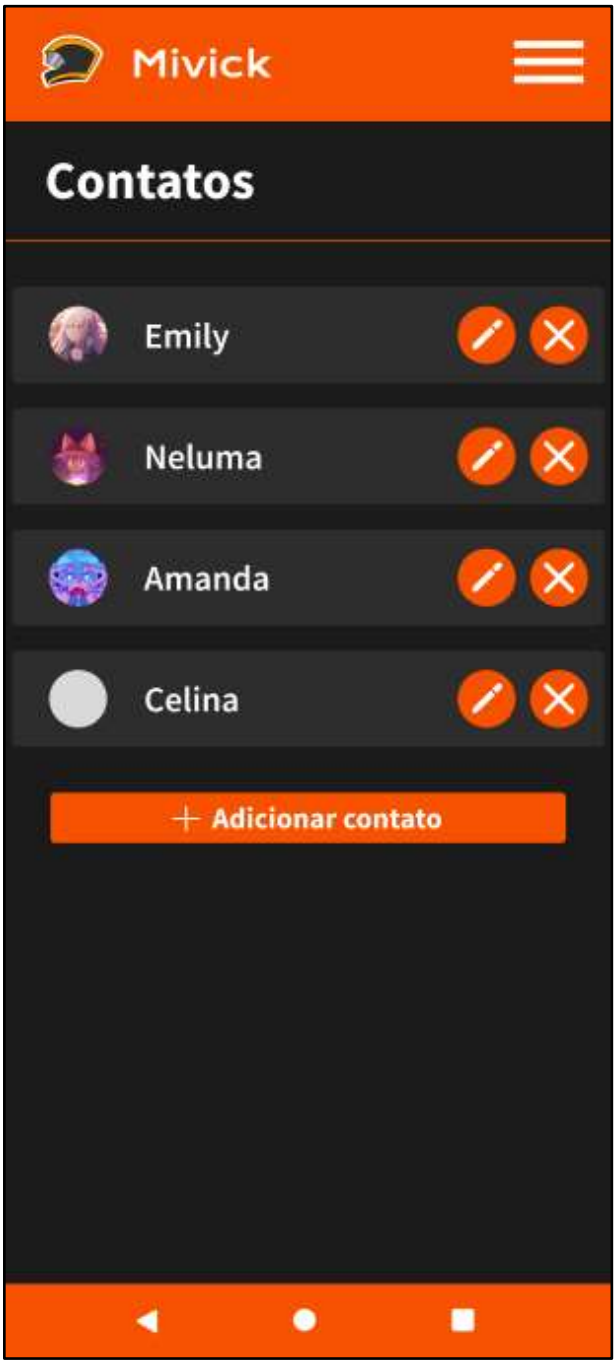
Figura 48 - Cadastro de contatos

Fonte: Autoria Própria, 2025

Nesta interface, para cadastrar o contato é necessário colocar seu nome, sobrenome, número de telefone, email e informar a prioridade dele, sendo o contato de maior prioridade o que irá ser avisado primeiro caso algum alerta de acidente seja acionado no dispositivo. Também é possível colocar uma foto no contato, assim, facilitando a identificação dele.

Assim que o contato é cadastrado, suas informações apareceram em uma tela junto com outros cadastros anteriormente cadastrados, assim como mostra a imagem a seguir:

Figura 49 - Contatos cadastrados

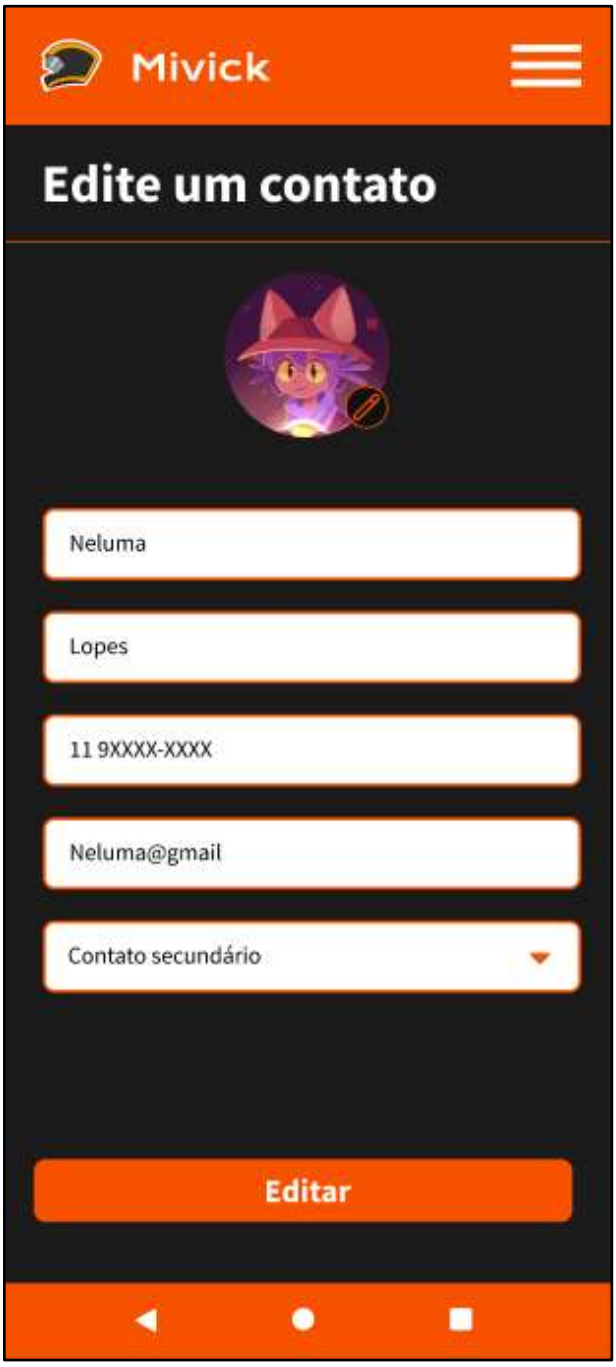


Fonte: Autoria Própria, 2025

Pode-se ver que a tela mostra o primeiro nome de cada cadastro cadastrado no aplicativo, e do lado, botões para editar e excluir o contato. Abaixo a interface possui um botão que leva novamente para a tela de cadastro de contato.

A interface abaixo se trata da tela de edição de contato, que segue o mesmo padrão da tela de cadastro, porém, tendo as informações já inseridas na caixa de texto, assim não sendo necessário colocar as informações novamente.

Figura 50 - Edição de contato



Fonte: Autoria Própria, 2025

Em seguida, é demonstrado a tela de perfil, que pode ser acessada através do menu na interface de home. Nela, o usuário pode ver e editar as informações anteriormente cadastradas, como seu nome, email e telefone, além da foto de perfil.

Figura 51 - Tela de perfil

Fonte: Autoria Própria, 2025

Adiante, vem a tela de conexão com o dispositivo, que pode ser acessada na página inicial. Nela, o usuário primeiro aceitar a permissão de acesso ao Bluethooth do aplicativo e depois clicar em “parear” para encontrar o dispositivo via Bluethooth. Logo após, aparece a tela onde o dispositivo pode ser pareado com o aplicativo.

Figura 52 - Telas de conexão com o dispositivo

Fonte: Autoria Própria, 2025

Assim que o dispositivo é conectado, o app direciona para a tela de configurações do dispositivo, aonde o usuário pode ligar e deslgar o dispositivo, os sensores e conectar com wi-fi.

Figura 53 - Página de configurações do dispositivo

Fonte: Autoria Própria, 2025

Nesta página também é possível acessar o histórico de alertas dos sensores, que mostra a hora e o dia do alerta, a foto tirada na hora e se o alerta foi desativado ou não.

Figura 54 - Página de histórico

Fonte: Autoria Própria, 2025

4 CONSIDERAÇÕES FINAIS

A partir dos resultados das pesquisas realizadas, pode-se chegar à conclusão do quão vulneráveis são os ciclistas e motociclistas quando se trata de acidentes de trânsito, assim foi pensado na criação do sistema Mivick, que alerta e traz uma maior segurança para esses usuários. Como resultado, obtivemos a certeza de como podemos usar a tecnologia para ajudar um público a se sentir um pouco mais seguro a realizar uma simples ação como andar de bicicleta e de moto, por exemplo. Nosso sistema é capaz de trazer mais alertas e, consequentemente, mais atenção para os usuários e aqueles que estão em torno deles, diminuindo a taxa de acidentes por falta de atenção.

Essa solução também pode alcançar mais pessoas por ser de baixo custo. Os dispositivos montados, tal qual o aplicativo, foram feitos utilizando uma tecnologia que, apesar de terem funções um tanto complexas, tem um custo acessível, tornando a solução mais barata, porém, não deixando de trazer os resultados esperados.

Em resumo, foi criado uma solução eficaz e de baixo custo que nos mostra a importância de pensar em usuários que são vulneráveis, mas nem sempre são o foco de pesquisas e principalmente de propostas tecnológicas que podem ajudar a diminuir os acidentes de trânsito e a taxa de mortalidade no Brasil, tornando esse tipo de situação um pouco mais negligenciada pela sociedade em comparação a outras causas.

REFERÊNCIAS

ABRAMET – Associação Brasileira de Medicina do Tráfego. Quase 8 em cada 10 vítimas graves do trânsito são pedestres, ciclistas ou motociclistas. 2023. Disponível em: https://abramet.com.br/noticias/quase-8-em-cada-10-vitimas-graves-do-transito-sao-pedestres-ciclistas-ou-motociclistas/. Acesso em: 22 maio 2025.

ALMEIDA, Lucas Henrique de. Firebase e React Native como ferramentas para criação de aplicativo de controle financeiro pessoal. 2022. Trabalho de Conclusão de Curso (Curso Técnico em Informática) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campinas. Disponível em: https://ric.cps.sp.gov.br/handle/123456789/10640. Acesso em: 24 maio 2025.

ALVES, Libni Momberg. As vantagens das baterias de lítio: um estudo de caso. 2022. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Mecânica) — Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, São Paulo, 2022.

ANTONIO, Beatriz de Araujo; TERUYA, Thiago Toshi; MOCHIZUKI, Luis. Uso do acelerômetro e giroscópio no monitoramento de movimento: uma avaliação comparativa por meio de unidade inercial e smartphone. Revista Brasileira de Educação Física e Esporte, São Paulo, v. 34, n. 3, p. 429–436, 2020. Disponível em: https://www.revistas.usp.br/rbefe/article/view/175255. Acesso em 28.abr.2025.

BARELLI, Felipe. Introdução à visão computacional: uma abordagem prática com Python e OpenCV. São Paulo: Casa do Código, [2018]. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=CA5ZDwAAQBAJ&oi=fnd&pg=PT3&dq=Introducao+a+Visao+Computacional+-+Uma+abordagem+pratica+com+Python+e+OpenCV+(Casa+do+Codigo)+(.pdf&ots=3lauA12HR7&sig=EfOOSRKlRlR8VdY2UgRMi2HbrOM#v=onepage&q&f=false. Acesso em 28.abr.2025.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML: guia do usuário. 2. ed. Rio de Janeiro: Elsevier, 2006.

BRASIL. Lei nº 9.503, de 23 de setembro de 1997. Institui o Código de Trânsito Brasileiro. Diário Oficial da União: seção 1, Brasília, DF, p. 1, 24 set. 1997. Disponível em: https://www.planalto.gov.br/ccivil_03/leis/l9503.htm. Acesso em: 22 maio 2025.

CASTRO, Mariana Oleone Marinho de. Estudo de visão computacional e criação de um Haar Cascade utilizando a biblioteca OpenCV em Python para

detecção de objetos. 2021. Trabalho de Conclusão de Curso (Graduação) – Escola de Engenharia de Lorena, Universidade de São Paulo, Lorena, 2021. Disponível em: https://bdta.abcd.usp.br/directbitstream/96046776-c6b4-4cfc-a82d-80ada71fdc83/EF21007%20%281%29%20MARIANA%20OLEONE.pdf. Acesso em: 24 maio 2025.

COSTA, Fábio Luiz da Silva. Microcontroladores e sistemas digitais: projetos, aplicações e técnicas. 2. ed. São Paulo: Érica, 2018. 320 p.

CUNHA, Antônio Carlos de Albuquerque. Projeto de sistemas digitais com microcontroladores. São Paulo: Érica, 2016.

DUARTE JÚNIOR, Luiz Fernando. Node.Js e Microservices. São Paulo: Casa do Código, 2020.

ESCUDELARIO, Bruna; PINHO, Diego. React Native: desenvolvimento de aplicativos mobile com React. São Paulo: Casa do Código, 2021.

FERREIRA, Caio Ribeiro. Node.js: aplicações web real-time com Node.js. São Paulo: Novatec Editora, 2015.

FERREIRA, Rogério. User Experience Design: como criar produtos digitais com foco nas pessoas. Rio de Janeiro: Casa do Código, 2015.

GALVÃO, Pedro Sereno. Comprehensive Repository Analysis of Mobile Projects Built with React Native. 2018. Dissertação (Programa de Graduação em Ciência da Computação) – Universidade Federal de Pernambuco, Recife, 2018.

GODOI, Maiko Gustavo De; ARAÚJO, Liriane Soares de. A Internet das Coisas: evolução, impactos e benefícios. Revista Interface Tecnológica, v. 17, n. 1, p. 38–50, 2020. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/download/538/363. Acesso em: 24 maio 2025.

GOMES, Alberto. Node.js e microservices. 2023. Disponível em: https://pt.scribd.com/document/687719759/Node-Js-e-Microservices. Acesso em 28.abr.2025.

GUEDES, Gilleanes T. A. UML 2: uma abordagem prática. 2. ed. São Paulo: Novatec, 2011. ISBN 978-85-7522-281-2.

JOBSTRAIBIZER, Flávia. Criação de Sites com CSS. 1. ed. São Paulo: Digerati, 2009. 144 p.

LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007.

LEITE, Talita de O.; SETE, Lucas S.; BARROS, Matheus G. M.; LOURENÇO, Giovanna G.; GULO, Carlos Alex Sander J. Design de UX e Prototipagem: Moldando as Escolhas do Usuário. In: ESCOLA REGIONAL DE INFORMÁTICA DE MATO GROSSO (ERI-MT), 13., 2024, Alto Araguaia/MT. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 189-195. ISSN 2447-5386. DOI: https://doi.org/10.5753/eri-mt.2024.245930.

LIMA, João Pedro. React Native: desenvolvimento híbrido com desempenho nativo. Revista de Tecnologia e Inovação, São Paulo, v. 12, n. 2, p. 45–60, 2020. Disponível em https://cbt.ifsp.edu.br/images/Documentos/2021/Trabalho_de_TCC/417_Apostila_React.pdf. Acesso em 28.abr.2025.

LIRA, Valdemir Martins. Processos de fabricação por impressão 3D: tecnologia, equipamentos, estudo de caso e projeto de impressora 3D. São Paulo: Ed. Blucher, 2021. 136 p. ISBN 978-6555062991.

MARENGONI, M.; STRINGHINI, S. Tutorial: introdução à visão computacional usando OpenCV. Revista de Informática Teórica e Aplicada, v. 16, n. 1, p. 125–160, 2010. Disponível em: https://seer.ufrgs.br/index.php/rita/article/view/rita_v16_n1_p125. Acesso em 28.abr.2025.

MACHADO, Kheronn Khennedy. Angular 11 e Firebase: construindo uma aplicação integrada com a plataforma Google. São Paulo: Casa do Código, 2020. 228 p. ISBN 978-85-6625-981-1. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=cRi9DwAAQBAJ&oi=fnd&pg=PT5&dq=FireBase&ots=a6TPe8s21E&sig=JX_GQMe4naSm91u78g_xjHwwVWc#v=onepage&q=FireBase&f=false. Acesso em 28.abr.2025.

MILANO, Danilo de; HONORATO, Luciano Barrozo. Visão Computacional. Limeira: UNICAMP – Universidade Estadual de Campinas, 2023. Disponível em: https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=VIS%C3%83O+COMPUTACIONAL+Danilo+de+Milano+Luciano+Barrozo+Honorato&btnG=. Acesso em 28.abr.2025.

MILETTO, Evandro Manara; BERTAGNOLLI, Silva de Castro. Desenvolvimento de Software II: Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP. 1. ed. Porto Alegre: Bookman, 2014. 276 p.

MINISTÉRIO DA SAÚDE (Brasil). Boletim Epidemiológico – Volume 54 – Nº 06 – 2023. Brasília, DF: Ministério da Saúde, 2023. Disponível em: https://www.gov.br/saude/pt-br/centrais-de-conteudo/publicacoes/boletins/epidemiologicos/edicoes/2023/boletim-epidemiologico-volume-54-no-06/. Acesso em: 22 maio 2025.

MONTEIRO, Jeferson; SANTOS, Lucas F. Internet das Coisas com ESP32: do básico à aplicação completa. São Paulo: Novatec, 2020.

MORAES, Luiz Carlos de Andrade. Sensores para sistemas embarcados: princípios e aplicações. Rio de Janeiro: LTC, 2015.

MORAES, William Bruno. Construindo aplicações com Node.js. São Paulo: Casa do Código, 2014.

MORAIS, José V. S. ESP32 com IDF: 0 Guia Profissional. São Paulo: Editora NCB, 2023. 189 p.

NAKATANI, Alessandro Massayuki; GUIMARÃES, Anderson Valenga; MACHADO NETO, Vicente. Medição com sensor ultrassônico HC-SR04. In: CONGRESSO INTERNACIONAL DE METROLOGIA MECÂNICA, 2014, Gramado. Anais [...]. Gramado: CIMEC, 2014. Disponível em: https://www.researchgate.net/publication/269874147_Medicao_Com_Sensor_Ultrassonico_HC-SR04. Acesso em: 24 maio 2025.

PEREIRA, Caio Ribeiro. Aplicações web real-time com Node.js. 2. ed. São Paulo: Casa do Código, 2014. 202 p. ISBN 978-85-6625-093-0. Disponível em: https://books.google.com.br/books?id=Wm-CCwAAQBAJ&lpg=PT7&ots=_fp0-pxnfJ&dq=Node&lr=lang_pt&hl=pt-BR&pg=PT4#v=onepage&q=Node&f=true. Acesso em 28.abr.2025.

PEREIRA, Rogério. User Experience Design: como criar produtos digitais com foco nas pessoas. São Paulo: Casa do Código, 2018. Disponível em: https://www.kufunda.net/publicdocs/User%20Experience%20Design%20%E2%80%93%20Como%20criar%20produtos%20digitais%20com%20foco%20nas%20pessoas%20(Rog%C3%A9rio%20Pereira).pdf. Acesso em 28.abr.2025.

SILVA, Alessandro Dionisio da. Anemômetro ultrassônico baseado em sensor de distância. 2014. Dissertação (Mestrado em Ciências Climáticas) – Universidade Federal do Rio Grande do Norte, Natal, 2014. Disponível em: https://bdtd.ibict.br/vufind/Record/UFRN_f86c390fa47fefae77e421a04df9446e. Acesso em 28.abr.2025.

SILVA, Carlos Henrique da. Introdução a sistemas embarcados: arquitetura, programação e aplicações. São Paulo: Érica, 2017.

SILVA, João. Firebase: uma abordagem prática. [S.l.]: Independently published, 2020.

SILVA, José Klinsman Jorge. Air CoPilot: um copiloto artificial para auxiliar os pilotos de drones e prevenir acidentes. 2025. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) — Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Campus Cajazeiras, 2025.

SILVA, Rodrigo de; LIMA, Ana Paula de. Uso do Firebase para desenvolvimento de aplicações móveis. Revista de Estudos e Pesquisas em Tecnologia de Informação e Educação – REFAQI, [S. l.], v. 2, n. 3, p. 22–35, 2020.

SILVA, Werliton Carlos Sousa da. Aplicações móveis nativas com React Native e Firebase: um estudo de caso. 2018. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Federal do Maranhão, São Luís, 2018. Disponível em: https://monografias.ufma.br/jspui/handle/123456789/3498. Acesso em 28.abr.2025.

SOUZA, Vitor Amadeu. Monitorando H_2 remotamente através da Internet com ESP32 programado em C. 1. ed. São Paulo: Clube de Autores, 2020. 108 p. ISBN 3410003196117.

SWARUP, Vidyadhar; PILLAI, S. Python para todos: uma introdução prática. São Paulo: Novatec Editora, 2018.

SWEIGART, Al. Automatize tarefas maçantes com Python: programação prática para iniciantes. São Paulo: Novatec Editora, 2016.

TEIXEIRA, Fabricio. Introdução e boas práticas em UX Design. São Paulo: Casa do Código, 2014. Disponível em: https://www.kufunda.net/publicdocs/Introdu%C3%A7%C3%A3o%20e%20boas%20pr%C3%A1ticas%20em%20UX%20Design%20(Fabricio%20Teixeira).pdf.

TEIXEIRA JÚNIOR, Helton José. Tratamento da interferência da temperatura em sensor de distância ultrassônico HC-SR04. 2022. 73 f. Monografia (Graduação em Engenharia de Controle e Automação) – Universidade Federal de Ouro Preto, Ouro Preto, 2022. Disponível em: https://monografias.ufop.br/handle/35400000/4797. Acesso em: 24 maio 2025.

VINÍCIUS, O. L.; RICARDO, Z. Proposta de emprego de giroscópio e acelerômetro na perícia de acidentes automobilísticos. Revista Brasileira de Criminalística, Brasília, v. 5, n. 2, p. 1–10, 2016. Disponível em:

https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=Proposta+de+emprego+de+girosc%C3%B3pio+e+acelerr%C3%B4metro+na+per%C3%ADcia+de+acidentes+automobil%C3%ADsticos+&btnG=. Acesso em 28.abr.2025.

COMACHIO, Vanderson. Funcionamento de banco de dados em Android: um estudo experimental utilizando SQLite. 2011. 68 f. Trabalho de Conclusão de Curso (Graduação) — Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Campus Medianeira, 2011. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/13401/2/MD_COADS_2011_2_07.pdf. Acesso em: 6 out. 2025.

GOMES, Flávio; et al. Ionic 6: Desenvolvimento Multiplataforma para Dispositivos Móveis. São Paulo: Casa do Código, [2022].

CASTRO, Lúcia; BAIÃO, Fernanda; GUIZZARDI, Giancarlo. Metodologia para Modelagem Conceitual de Dados. In: SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO (SBSI), 5., 2009, Brasília. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2009. p. 294-299. DOI: https://doi.org/10.5753/sbsi.2009.6186.

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 6. ed. Porto Alegre: Bookman, 2009.

XAVIER, Felipe. Desenvolvimento de um dispositivo IoT de baixo custo para monitoramento de gases. 2022. Trabalho de Conclusão de Curso (Bacharelado em Tecnologia) — Universidade Tecnológica Federal do Paraná, Toledo, 2022.