

**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**ETEC DA ZONA LESTE**

**NOVOTEC Desenvolvimento de Sistemas**

**Miguel Gustavo de Sousa Campos**

**Pablo de Sousa Santos**

**Ricardo Luquetti Codo**

**Victor Hugo Navarro Taveira**

**CANISHERZ: Sistema embarcado para medição cardíaca em tempo  
real de cachorros domésticos**

**São Paulo**

**2024**

**Miguel Gustavo de Sousa Campos**

**Pablo de Sousa Santos**

**Ricardo Luquetti Codo**

**Victor Hugo Navarro Taveira**

**CANISHERZ: Sistema IoT para medição cardíaca em tempo real de  
cachorros domésticos**

Trabalho de Conclusão apresentado ao  
Curso do Ensino Médio com Habilitação  
Profissional de Técnico em Desenvolvimento  
de Sistemas da ETEC da Zona Leste,  
orientado pelo Prof. Jeferson Roberto de  
Lima, como requisito parcial para obtenção  
do título de técnico em Desenvolvimento de  
Sistemas.

**São Paulo**

**2024**

“A vida dos cães é muito curta. Sua única falha, realmente”.

**Agnes Sligh Turnbull**

## **RESUMO**

A presença dos problemas cardiopatas em cães, apesar de pouco retratado, mostra-se um desafio para o animal e seus cuidadores. Seus sintomas, mesmo incomuns, são de difícil detecção, considerando a barreira comunicativa existente. Para uma melhor compreensão destas adversidades, objetiva-se o desenvolvimento de um dispositivo capaz de agilizar e evitar possíveis eventualidades e ameaças à saúde, onde a ferramenta terá a função de monitorar a frequência cardíaca em tempo real. Além disso, para a visualização e notificação de possíveis anomalias nos batimentos, visa-se também a criação de uma aplicação móvel, responsável por informar ao cuidador sobre a saúde do animal. Para uma maior portabilidade e acessibilidade, almeja-se a implementação deste aplicativo para sua utilização em relógios inteligentes, facilitando o acesso a estes dados.

**Palavras-chave:** cardiopatas; cães; aplicação; dispositivo; monitoramento; relógios inteligentes.

## **ABSTRACT**

The presence of heart problems in dogs, although not often reported, is a challenge for the animal and its carers. Their symptoms, although uncommon, are difficult to detect, given the existing communication barrier. To better understand these adversities, the aim is to develop a device capable of speeding up and avoiding eventualities and threats to health, where the tool will have the function of monitoring heart rate in real time. In addition, to visualize and notify anomalies in the heartbeat, the aim is also to create a mobile application responsible for informing the caregiver about the animal's health. For greater portability and accessibility, the aim is to implement this application for use in smart watches, facilitating access to this data.

**Keywords:** heart problems; dogs; application; device; monitoring; smart watches.

## LISTA DE ILUSTRAÇÕES

|  |    |
|--|----|
| Figura 1 – Exemplo Diagrama de caso de uso.....  | 12 |
| Figura 2 – Exemplo Diagrama de classe.....   | 13 |
| Figura 3 – Exemplo de diagrama de sequência .....                                      | 14 |
| Figura 4 – Exemplo de diagrama de atividade .....                                      | 15 |
| Figura 5 – Construção de um formulário em linguagem HTML.....                          | 16 |
| Figura 6 – Exemplo formulário em HTML.....   | 17 |
| Figura 7 – Estilização do HTML utilizando o CSS .....                                  | 18 |
| Figura 8 – Estilização do CSS no HTML utilizado .....                                  | 19 |
| Figura 9 – A implementação do Javascript no HTML .....                                 | 20 |
| Figura 10 – Resultado do Javascript no HTML .....                                      | 21 |
| Figura 11 – Criação de projeto expo React Native.....                                  | 22 |
| Figura 12 – Exemplo de estrutura de pasta e um formulário básico em React Native ..... | 23 |
| Figura 13 – Continuação formulário básico, Estilização .....                           | 25 |
| Figura 14 – Resultado do aplicativo de formulário .....                                | 27 |
| Figura 15 – Resultado do aplicativo de formulário após pressionamento do botão ..      | 28 |
| Figura 16 – Programa para manipulação de LEDs .....                                    | 31 |
| Figura 17 – Exemplo código C++ LED apagado .....                                       | 32 |
| Figura 18 – Exemplo código C++ LED aceso.....  | 33 |
| Figura 19 – Exemplo de Estrutura do Firecloud .....                                    | 35 |
| Figura 20 – Exemplo de Prototipação da Interface de Baixa Fidelidade .....             | 37 |
| Figura 21 – Exemplo de Prototipação da Interface de Alta Fidelidade.....               | 38 |

## LISTA DE QUADROS

## **LISTA DE ABREVIATURAS E SIGLAS**

Cascading Style Sheet (CSS)

Hypertext Markup Language (HTML)

integrated development environment (IDE)

Internet of Things (IoT)

JavaScript (JS)

Light Emitting Diode (LED)

Npx (Node Package Executor)

Structured Query Language (SQL)

Computer Aided Design (CAD)



|  |                               |
|--|-------------------------------|
| <b>INTRODUÇÃO .....</b>                          | <b>10</b>                     |
| <b>2 REFERENCIAL TEÓRICO.....</b>                | <b>11</b>                     |
| 2.1 Cardiopatias.....                            | 11                            |
| 2.2 Uml .....                                    | 11                            |
| 2.3 Html .....                                   | 15                            |
| 2.3.1 Funcionamento do HTML na prática.....      | 15                            |
| 2.4 Css.....                                     | 17                            |
| 2.5 Javascript.....                              | 19                            |
| 2.6 React.....                                   | 21                            |
| 2.7 React Native .....                           | 21                            |
| 2.7.1 Criação de aplicação em React Native ..... | 22                            |
| 2.7.2 StyleSheet .....                           | Erro! Indicador não definido. |
| 2.8 IoT .....                                    | 29                            |
| 2.9 ESP32 .....                                  | 29                            |
| 2.10 Sensor MAX30102 .....                       | 30                            |
| 2.11 Battery Shield V8.....                      | 30                            |
| 2.12 C.....                                      | 30                            |
| 2.13 C++.....                                    | 30                            |
| 2.14 Wear OS .....                               | 33                            |
| 2.15 Kotlin .....                                | 33                            |
| 2.15.1 Jetpack Compose.....                      | 34                            |
| 2.16 Firebase.....                               | 34                            |
| 2.16.1 Firestore .....                           | 35                            |
| 2.17 Impressão 3D .....                          | 36                            |
| 2.17.1 Modelagem 3D .....                        | 36                            |
| 2.18 Wireframe.....                              | 36                            |
| <b>REFERENCIAS.....</b>                          | <b>39</b>                     |



## 1 INTRODUÇÃO

## **2 REFERENCIAL TEÓRICO**

Essa seção abrange a documentação teórica que fundamenta o projeto, atribuindo noções fundamentais sobre as tecnologias que serão empregadas na criação e elaboração do dispositivo CanisHerz.

### **2.1 Cardiopatias**

Conforme Feldman (2024), A cardiopatia é um termo geral que se refere a qualquer doença do coração. Isso pode incluir uma variedade de condições, como arritmias cardíacas.

Franco (2022), afirma que os efeitos negativos das doenças cardíacas em pacientes veterinários podem afetar o tempo de sobrevivência dos animais, visando analisar a qualidade de vida de cães com cardiopatias.

O sistema muscular responsável pela excitação e condução controla a função cardíaca. Segundo Jericó, Neto e Kogika (2015), O ritmo e a eficácia da bomba cardíaca são diretamente afetados pelo sistema de condução.

### **2.2 Uml**

UML é uma linguagem para modelar softwares, adotada na indústria de engenharia de software, de propósito geral aplicada a todos os domínios. Sendo uma ferramenta padrão globalmente reconhecida como demonstrado por Guedes (2018).

Sua criação foi um processo experimental diz Fowler et al (2005). O resultado é um padrão baseado em diversas ideias e contribuições feitas por diversos colaboradores durante o processo.

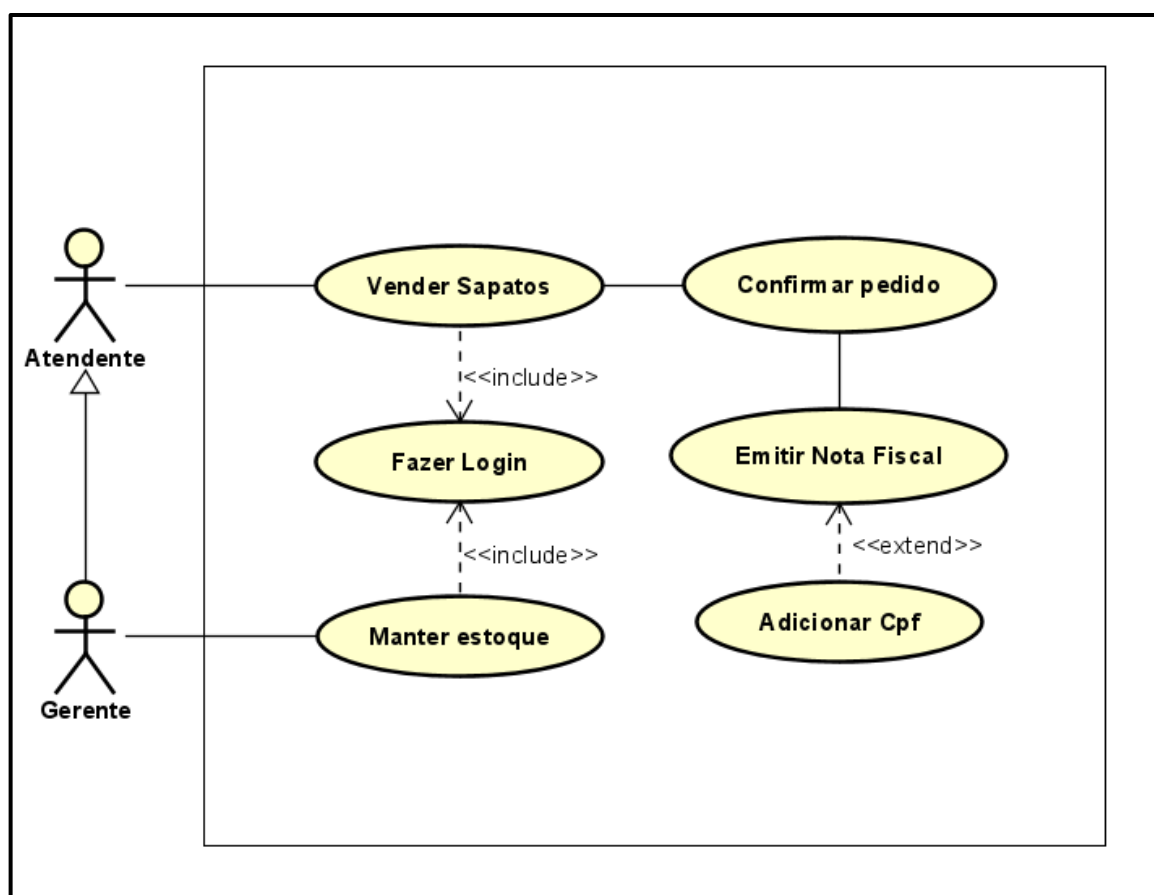
Mencionado por Booch, Rumbaugh e Jacobson (2012), ela possui uma semântica precisa, garantindo interpretação clara por qualquer desenvolvedor ou ferramenta. Isso assegura que os modelos possam ser compreendidos sem ambiguidades.

A UML é composta por vários diagramas, com o objetivo de expor uma variedade de visões do sistema, em vários aspectos, dessa maneira, almejando uma modelagem com um número mínimo de falhas segundo Guedes (2018).

Entre esses vários tipos de diagramas, uma parte é utilizado nesse projeto, como: Diagrama de Caso de Uso, Diagrama de Classe, Diagrama de Atividade e Diagrama de Sequência.

O diagrama de caso de uso fornece a visão das funcionalidades para os usuários. Concentra-se na representação das interações do sistema, sem entrar em detalhes de implementação, como dito por Booch, Rumbaugh e Jacobson (2012).

Figura 1 – Exemplo Diagrama de caso de uso

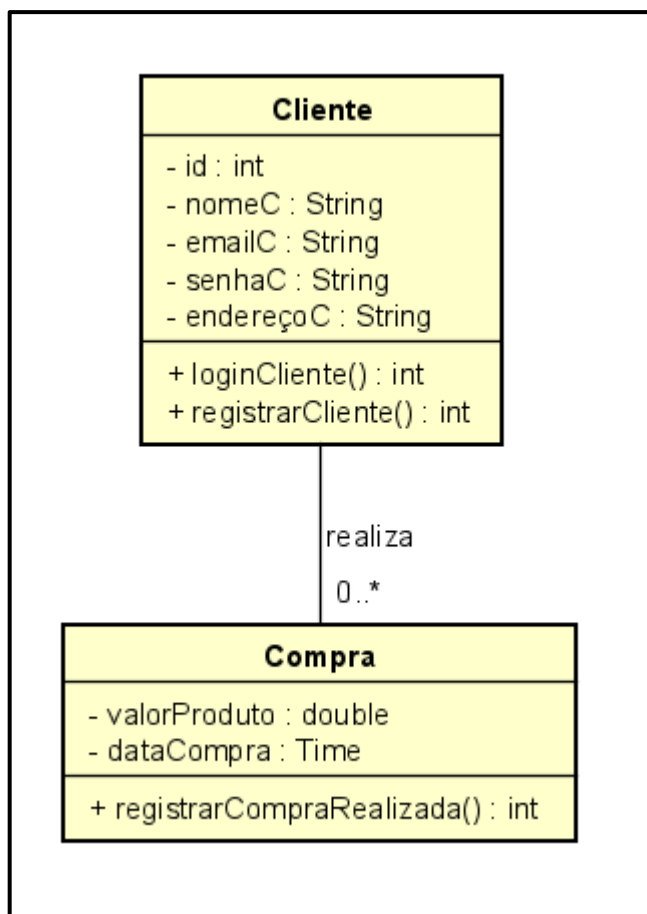


Fonte: Do próprio autor, 2024

A imagem acima mostra as responsabilidades de um Atendente e um Gerente em uma loja de sapatos. O Atendente pode vender sapatos e confirmar pedidos, que pode incluir a nota fiscal e opcionalmente adicionar o cpf. O Gerente tem a tarefa de manter estoque, porém herda todas as funções do atendente, podendo realizar vendas de sapatos. Ambos os atores requerem Fazer Login para suas tarefas.

Conforme Fowler et al (2005), o diagrama de classes descreve os tipos de objetos e seus relacionamentos estáticos no sistema, incluindo propriedades, operações e restrições. Eles oferecem uma visão estruturada das classes e suas características.

Figura 2 – Exemplo Diagrama de classe

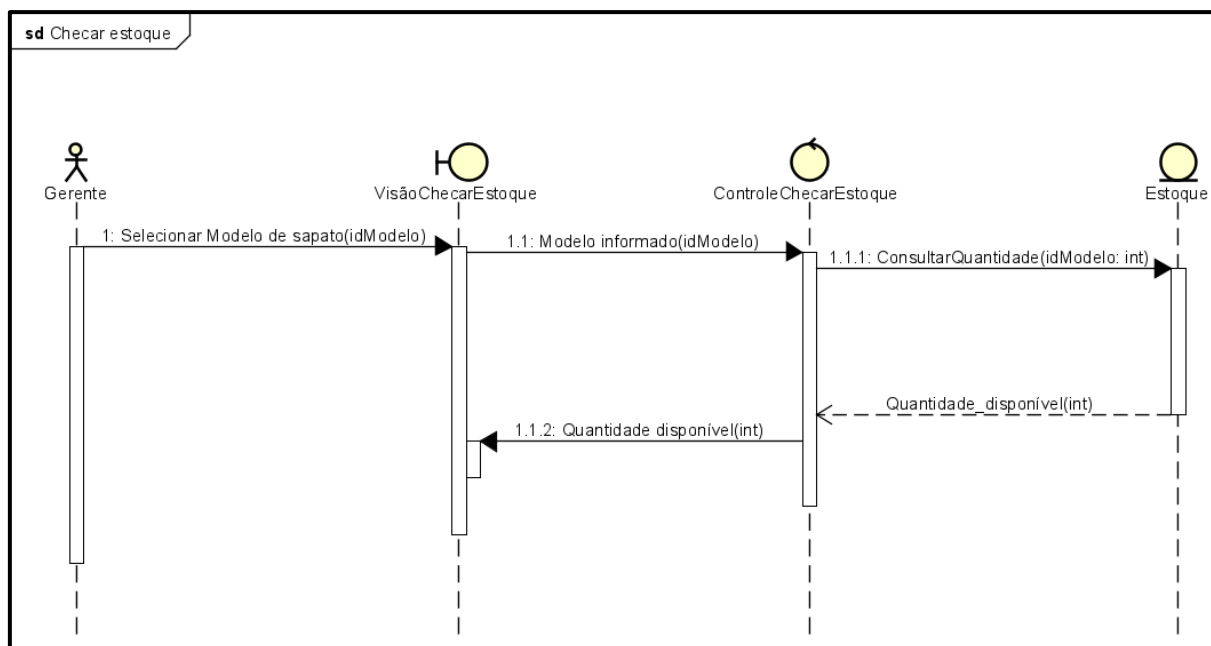


Fonte: Do próprio autor, 2024

Acima é apresentada duas classes, cliente e compra. Cliente possui atributos como id, nome, email, senha e endereço, e métodos para login e registro do comprador. Compra tem atributos para o valor do produto e a data da compra, assim como um método para registrar compras realizadas. E temos uma relação “realiza” entre Cliente e Compra, indicando que um cliente pode realizar compras.

Parafraseando Guedes (2018), o diagrama de sequência tem o objetivo de detalhar a ordem precisa de eventos, mensagens enviadas e métodos chamados, oferecendo uma visão clara de como os objetos interagem durante os processos.

Figura 3 – Exemplo de diagrama de sequência

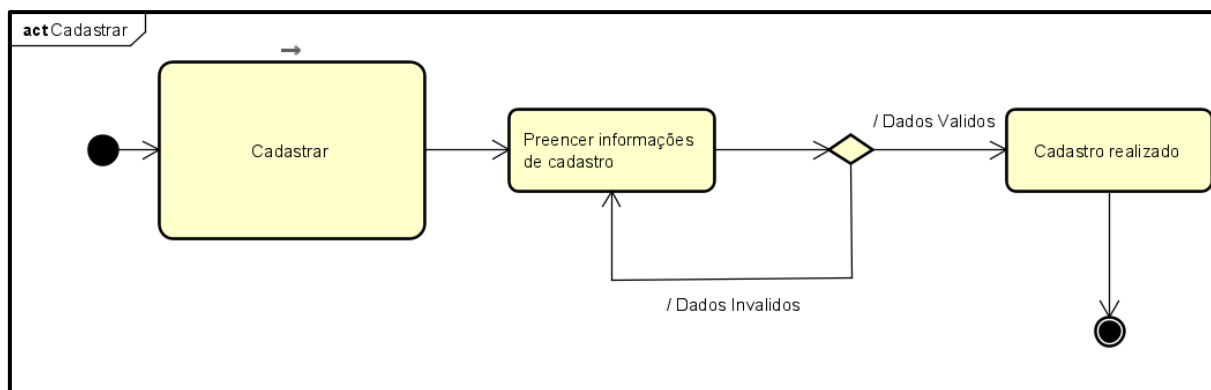


Fonte: Do próprio autor, 2024

O diagrama de sequência "Checar Estoque" apresentado anteriormente demonstra o procedimento de verificação de disponibilidade de um modelo de sapato no estoque de uma loja. O Gerente começa o processo na interface gráfica "Visão Checar Estoque", selecionando o modelo que deseja ver a quantidade disponível no estoque. A escolha é enviada ao "Controle Checar Estoque", que consulta o estoque. O estoque responde com a informação, retornando-a à interface gráfica para visualização do gerente. As lifelines mostram a existência dos objetos durante a interação, enquanto as setas preenchidas indicam mensagens e as pontilhadas representam retornos de informação.

O diagrama de atividades como lembrado por Booch, Rumbaugh e Jacobson (2012), apresenta como uma atividade leva a outra em um processo. Essas atividades resultam em ações concretas, como mudanças de estado no sistema.

Figura 4 – Exemplo de diagrama de atividade



Fonte: Do próprio autor, 2024

O diagrama apresenta o processo de cadastro, começando com a ação do usuário para iniciar o cadastro. Ele preenche as informações necessárias. O sistema confirma se os dados apresentados são corretos. Caso corretos, o cadastro é concluído com sucesso, se não, o usuário é solicitado a reinserir os dados.

O diagrama de máquina de estados apresenta o comportamento de um elemento por meio de um grupo finito de transições de estado. Além de ser utilizado para expressar o comportamento de uma parte do sistema como lembrado por Fowler et al (2005).

### 2.3 Html

A linguagem HTML, traduzida para o português como linguagem de marcação de hipertexto citado pelo autor Silva (2008), tem o objetivo de criar sites de computadores com uma estrutura de visualização mais simples.

Segundo o Flatschart (2011) destaca-se que a linguagem de marcação de hipertexto, pode ser criada de forma semântica, ou seja, permitindo a maneira que será projetada na janela de visualização da página.

Portanto, o acesso a esses códigos é feito a via Browser, utilizando programas como o *Google*, *Internet Explorer*, entre outros tipos de *softwares*, conforme referido por Marly (2012).

#### 2.3.1 Funcionamento do HTML na prática

Usaremos um bloco de notas ou uma ferramenta específica para programação, para criar um tipo de arquivo HTML e estabelecer o escopo do código desta linguagem.



Figura 5 – Construção de um formulário em linguagem HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Somar</title>
</head>
<body>
  <section>
    <h1>Somatória de Dois Números</h1>
    <input type="number" id="n1" onfocus="calcular()" placeholder="Digite o Primeiro Valor">
    <input type="number" id="n2" onblur="calcular()" placeholder="Digite o Segundo Valor">
    <input type="text" id="result" placeholder="Resultado da Soma">
  </section>
</body>
</html>
```

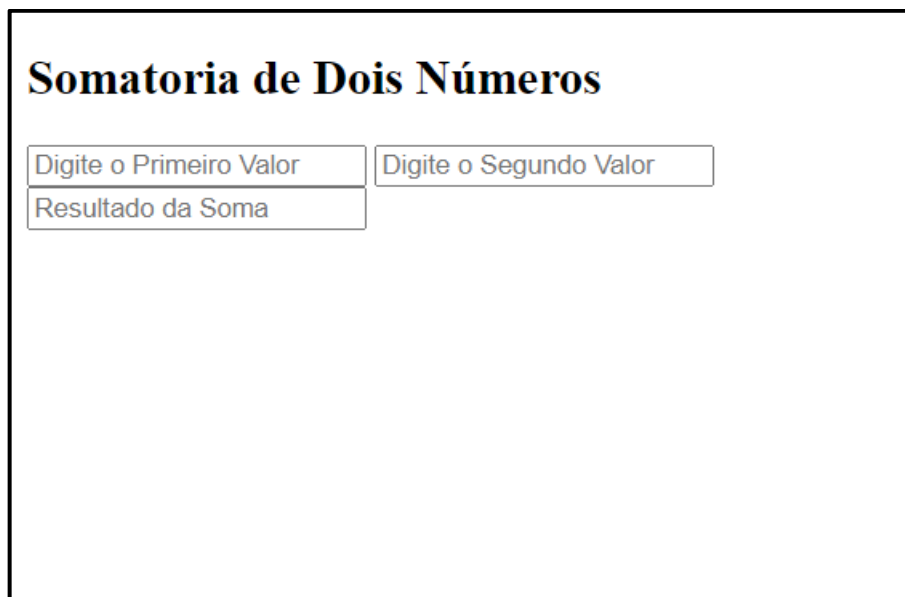
Fonte: Do próprio autor, 2024

Esse exemplo, mostra um formulário simples na estrutura da linguagem abordada anteriormente, utilizando os seguintes blocos de comandos:

- `<!DOCTYPE html>`: Declara o tipo de versão do HTML.
- `<html lang="en">`: Determina o idioma da estrutura HTML que será abordada.
- `<head>`: Inicia as informações dos registros que não são expostos.
- `<meta charset="UTF-8">`: Especifica a codificação de caracteres para que o navegador execute corretamente os caracteres internacionais e especiais.
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: São configurações do *viewport* (janela de exibição) para dimensões de dispositivos móveis.
- `<title>`: Declara o nome da página, que no exemplo utilizado é "Somar".
- `<body>`: É responsável pelo conteúdo visível da documentação.
- `<section>`: Define uma seção no documento, que neste caso contém o formulário para realizar a soma de dois números.
- `<h1>`: Denomina o título de nível 1 de espessura.
- `<input type="number" id="n1" onfocus="calcular()" placeholder="Digite o Primeiro Valor">`: Esta *tag* é um campo de digitação, utilizando o atributo chamado *'type'* para especificar o tipo de caracteres que será utilizado. O

atributo *'placeholder'* também pode ser utilizado para fornecer um texto de exemplo dentro do campo de entrada. Neste caso, são apenas números.

Figura 6 – Exemplo formulário em HTML



O formulário, intitulado "Somatoria de Dois Números", contém três campos de entrada. O primeiro campo, rotulado "Digite o Primeiro Valor", e o segundo campo, rotulado "Digite o Segundo Valor", estão alinhados horizontalmente. Abaixo do primeiro campo, há um terceiro campo rotulado "Resultado da Soma".

Fonte: Do próprio autor, 2024

## 2.4 Css

Envidenciado pelo Scheidt (2014), CSS (Cascading Style Sheets) ou Folhas de Estilo em Cascata é uma linguagem de estilo para a linguagem de marcação de hipertexto.

Segundo Miletto e Bertagnolli (2015) o CSS propõe a possibilidade de criar estilos personalizados para títulos, listas, botões, entre outros elementos visuais da documentação do HTML.

Esta linguagem pode ser programada em qualquer editor de HTML ou Bloco de notas. Sendo acessível, o CSS é bastante utilizado em projetos que envolve a linguagem de marcação de elementos, citado por Jobstraibizer (2009).

Figura 7 – Estilização do HTML utilizando o CSS

```
section{
  font-family: sans-serif;

  width: 20rem;
  padding: 1rem;
  background-color: cornflowerblue;
  color: aliceblue;
  border-radius: 1rem;
}

input[type=number]{
  width: 10rem;
  margin: .5rem 0px;
  border: none;
  border-radius: .5rem;
  font-size: 1rem;
  padding: .5rem 0rem;
  width: 100%;
}

input[type=text]{
  background-color: transparent;
  border: none;
  color: aliceblue;
  font-size: 1.5rem;
  margin-top: 1rem;
  width: 100%;
}

input[type=text]::-webkit-input-placeholder{
  color: aliceblue;
}
```

Fonte: Autoria Própria

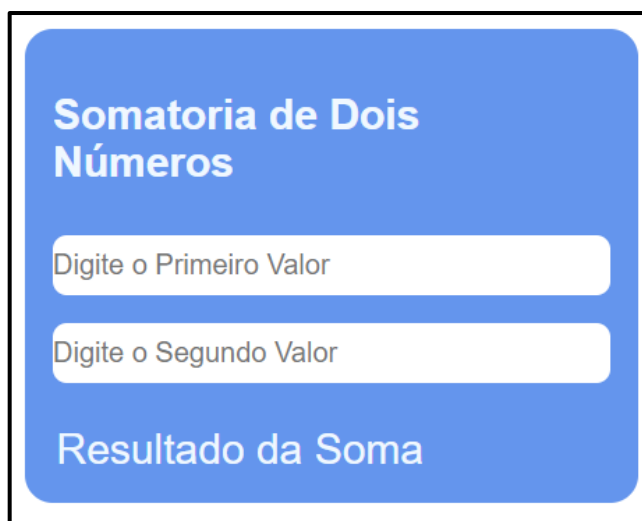
Estes são os comandos de estilização do HTML feito em blocos de comando, visando as marcações do HTML:

- `section {...}`: Este bloco de código aplica estilos a elementos HTML dentro de uma seção. Ele define a fonte como "sans-serif", define a largura como 20rem, adiciona um preenchimento interno de 1rem, define a cor de fundo como "cornflowerblue", a cor do texto como "aliceblue" e aplica um raio de borda de 1rem.
- `input[type=number] {...}`: Este bloco de código aplica estilos a elementos de entrada do tipo "number". Ele define a largura como 10rem, adiciona margens

de 0.5rem acima e abaixo, remove a borda, define um raio de borda de 0,5rem, define o tamanho da fonte como 1rem, adiciona um preenchimento de 0,5rem vertical e define a largura como 100%.

- `input[type=text] {...}`: Este bloco de código aplica estilos a elementos de entrada do tipo "text". Ele define a cor de fundo como transparente, remove a borda, define a cor do texto como "aliceblue", define o tamanho da fonte como 1.5rem, adiciona uma margem superior de 1rem e define a largura como 100%.
- `input[type=text]::-webkit-input-placeholder {...}`: Este bloco de código aplica estilos ao espaço reservado (placeholder). Ele define a cor do espaço reservado como "aliceblue", garantindo que o texto de *placeholder* seja de outra cor.

Figura 8 – Estilização do CSS no HTML utilizado



A imagem mostra uma interface web com um fundo azul sólido. No topo, o título "Somatoria de Dois Números" está em branco. Abaixo dele, há dois campos de entrada de texto brancos com o texto cinza "Digite o Primeiro Valor" e "Digite o Segundo Valor". No rodapé da interface, o texto "Resultado da Soma" também está em branco.

Fonte: Autoria Própria

## 2.5 Javascript

Javascript é uma linguagem dinâmica e interpretada, que segundo Flanagan (2013) não requer tipagem, tornando-a adequada para abordagens orientadas a objetos e funcionais.

Parafraseando Morrison (2008) ela é a chave para transformar um projeto web em uma página interativa, capacitando os elementos que podem responder às suas ações, processar entradas do usuário e atender às suas necessidades.

A linguagem não possui conceito de classe como apresentado por Zakas (2017), em vez disso, utiliza dois tipos: os primitivos que são armazenados como tipos simples e os tipos de referência que são armazenados como objetos.

Figura 9 – A implementação do Javascript no HTML

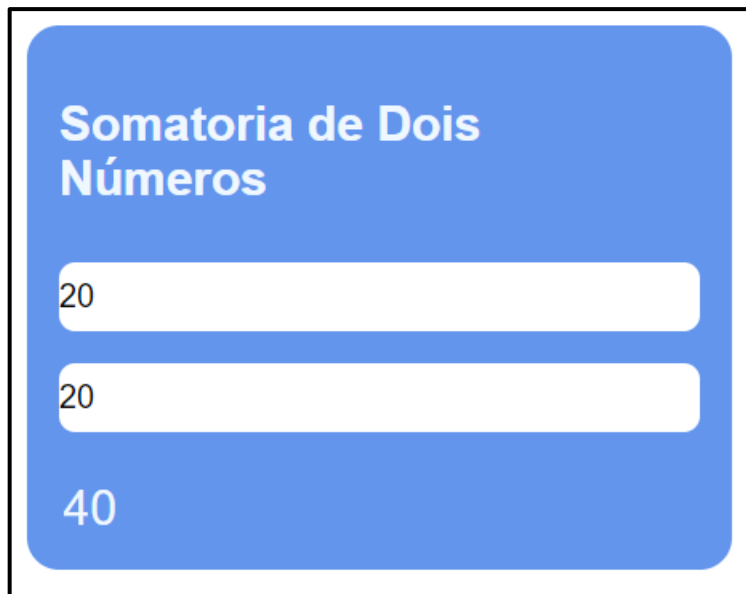
```
function calcular(){  
    var valor1 = parseInt(document.getElementById('n1').value, 10);  
    var valor2 = parseInt(document.getElementById('n2').value, 10);  
    document.getElementById('result').value = valor1 + valor2;  
}
```

Fonte: Do próprio autor, 2024

Nesse é exemplo, são os comandos para o funcionamento “somar” do formulário criado por HTML e estilizado por CSS:

- `Function calcular() {...}`: Isso define uma função chamada "calcular". Neste caso, a função "calcular" será chamada quando desejarmos realizar cálculo da soma.
- `Var valor1 = parseInt(document.getElementById('n1').value, 10)`: Esta linha declara uma variável chamada "valor1" e utiliza a função *parseInt* para converter o valor obtido do elemento HTML com o ID 'n1' para um número inteiro. O método *getElementById('n1')* obtém o elemento HTML com o ID 'n1', e o *".value"* recupera o valor inserido nesse elemento. O *parseInt* converte esse valor para um número inteiro.
- `Var valor2 = parseInt(document.getElementById('n2').value, 10)`: Similarmente à linha anterior, esta declara uma variável chamada "valor2" e converte o valor do elemento HTML com o ID 'n2' para um número inteiro.
- `Document.getElementById('result').value = valor1 + valor2`: Esta linha define o valor do elemento HTML com o ID 'result' como a soma dos valores de "valor1" e "valor2". O método *getElementById('result')* obtém o elemento HTML com o ID 'result', e *".value"* define o valor do elemento. O valor atribuído é a soma dos valores de "valor1" e "valor2", realizada pela operação de adição (+).

Figura 10 – Resultado do Javascript no HTML



**Somatoria de Dois Números**

20

20

40

Fonte: Do próprio autor, 2024

## 2.6 React

O React é uma biblioteca JavaScript capaz de facilitar o desenvolvimento de interfaces gráficas voltadas para o âmbito *web*, agilizando o processo de criação dessas telas e disponibilizando uma maior interação do usuário ao sistema (React, 2024).

Segundo Silva (2021), essa ferramenta, criada pelo Facebook, mostra-se uma das mais populares no ramo, sendo utilizada por grandes empresas como a Netflix, PayPal e Twitter, evidenciando sua eficácia no que propõe.

## 2.7 React Native

Como explicado por Escudelar e Pinho (2020), O React Native é considerado uma biblioteca do React, que disponibiliza várias ferramentas que auxiliam na criação de aplicações moveis, para iOS e Android, utilizando linguagens de suporte como HTML e CSS.

Foi introduzido em 2015 pelo Facebook. Permitindo aos desenvolvedores reutilizar conhecimentos das tecnologias web para o desenvolvimento de aplicativos móveis, facilitando a criação dos mesmos mencionado por Bezerra e Viana (2021).

Ele adota uma sintaxe do ReactJS, pelo uso do JSX. Porém, sua abordagem difere de sua raiz. Devido os seus elementos serem emulados de forma nativa, utilizando o JS Core como uma ponte entre os dois como citado por Silva e Sousa (2019).

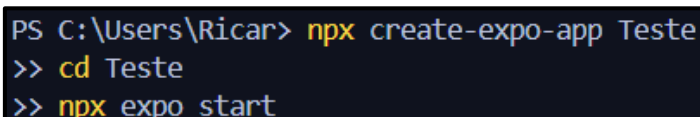
### 2.7.1 Expo

Segundo Alura (2022), a ferramenta expo, é um conjunto de ferramentas e serviços em torno de plataformas nativas que ajudam no desenvolvimento, construção, implementação e permitem a criação rapidamente de aplicativos iOS, Android e web.

### 2.7.2 Criação de aplicação em React Native

Utilizando a ferramenta anteriormente mencionada, e disposto os seguintes comandos no *Prompt* de Comando do Windows:

Figura 11 – Criação de projeto expo React Native



```
PS C:\Users\Ricar> npx create-expo-app Teste
>> cd Teste
>> npx expo start
```

Fonte: Do próprio autor, 2024

Os comandos acima utilizando o prefixo npx que permite executar comandos de pacotes diretamente, sem precisar instalá-los globalmente ou localmente, respectivamente farão:

- A criação do ambiente do projeto, com o nome informado;
- Direcionamento de diretório do Prompt para a pasta informada;
- Inicialização da aplicação no Expo, permitindo uma visualização mediante ao aplicativo Expo Go, instalado em um dispositivo móvel.

Após a execução, no diretório será criado uma variedade de pastas para configuração do ambiente da aplicação, assim o projeto poderá ser desenvolvido em uma IDE de preferência, modificando inicialmente o arquivo app.js para alteração das *views*.

Figura 12 – Exemplo de estrutura de pasta e um formulário básico em React Native

```

1  import React, { useState } from 'react';
2  import { View, Text, TextInput, Button, Alert, StyleSheet } from 'react-native';
3
4  const SimpleForm = () => {
5    const [name, setName] = useState('');
6    const [email, setEmail] = useState('');
7    const [message, setMessage] = useState('');
8
9    const handleSubmit = () => {
10     Alert.alert('Dados do Formulário', `Nome: ${name}\nEmail: ${email}\nMensagem: ${message}`);
11   };
12
13   return (
14     <View style={styles.container}>
15       <Text style={styles.title}>Formulário Simples</Text>
16       <TextInput
17         placeholder="Nome"
18         value={name}
19         onChangeText={text => setName(text)}
20         style={styles.input}
21       />
22       <TextInput
23         placeholder="Email"
24         value={email}
25         onChangeText={text => setEmail(text)}
26         style={styles.input}
27         keyboardType="email-address"
28       />
29       <TextInput
30         placeholder="Mensagem"
31         value={message}
32         onChangeText={text => setMessage(text)}
33         multiline
34         style={[styles.input, styles.messageInput]}
35       />
36       <Button title="Enviar" onPress={handleSubmit} />
37     </View>
38   );
39 }

```

Fonte: Do próprio autor, 2024

Nesse exemplo é apresentado um código de formulário simples como exemplo de aplicação, utilizando os seguintes elementos:

- `import { View, Text, TextInput, Button, Alert, StyleSheet } from 'react-native':` É utilizado para importar seis componentes do pacote padrão do react, fundamentais para construir interfaces e utilizar os comandos seguintes.
- `import React, { useState } from 'react':` Essa linha importa o React e a função `useState`. Ela é uma função especial que permite adicionar estado aos componentes.
- `const [name, setName] = useState('');` `const [email, setEmail] = useState('');` `const [message, setMessage] = useState('');` Esses comandos estão criando três estados: *name*, *email* e *message*, que serão usados para armazenar os valores do formulário. Inicializando cada estado com uma *string* vazia.



- `const handleSubmit = () =>` É a criação de uma função *handleSubmit*, que será chamada quando o botão "Enviar" for apertado.
- `Alert.alert('Dados do Formulário', `Nome: ${name}\nEmail: ${email}\nMensagem: ${message}`);` É utilizado para exibir um alerta com os dados inseridos no formulário posteriormente.
- `<View>`: Essa *tag* abre uma View. Essa View serve como um recipiente para os elementos do formulário.
- `<Text>`: Usada para representar texto em linha exibindo o mesmo.
- `<TextInput>`: Permite que os usuários insiram textos. Pode ser usado para receber entradas e interações onde elas são fornecidas pelo usuário.
- `<Button>`: Usado para criar um botão que os usuários podem tocar para realizar ações específicas, como especificado no exemplo: enviar um formulário.

A imagem abaixo mostra a estilização dos componentes anteriores, utilizando o *StyleSheet*, uma ferramenta que permite estilizar componentes, definindo estilos de forma eficiente e reutilizável, semelhante ao CSS em páginas da web.:

Figura 13 – Continuação formulário básico, Estilização

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 20,
    backgroundColor: '#f0f0f0',
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
    color: '#333',
  },
  input: {
    width: '100%',
    marginBottom: 10,
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    padding: 12,
    fontSize: 16,
    backgroundColor: '#fff',
  },
  messageInput: {
    height: 100,
    textAlignVertical: 'top',
  },
});

export default SimpleForm;
```

Fonte: Do próprio autor, 2024

- `const styles = StyleSheet.create({ container: { flex: 1, justifyContent: 'center', alignItems: 'center', padding: 20, backgroundColor: '#f0f0f0', }:` Esse bloco, apresenta um estilo chamado `container` sendo definido. Ele configura o componente para ocupar todo o espaço disponível utilizando o (`flex`), centralizando verticalmente (`justifyContent: 'center'`) quanto horizontalmente (`alignItems: 'center'`). Também, adiciona um preenchimento e define a cor de fundo (`padding`), (`backgroundColor`).
- `title: { fontSize: 24, fontWeight: 'bold', marginBottom: 20, color: '#333', }:` O Estilo é definido com o tamanho da fonte vinte e quatro (`fontSize`), em negrito (`fontWeight: 'bold'`). Ele adiciona uma margem inferior e define a cor do texto (`marginBottom`), (`color`).
- `input: { width: '100%', marginBottom: 10, borderWidth: 1, borderColor: '#ccc', borderRadius: 5, padding: 12, fontSize: 16, backgroundColor: 'fff', }:` A largura é definida (`width`), uma margem inferior (`marginBottom`) e uma borda (`borderWidth`), com cor (`borderColor`) e cantos arredondados (`borderRadius`). O preenchimento interno e a fonte são estilizados (`padding`), (`fontSize`). O fundo do campo é definido como branco (`backgroundColor`).
- `messageInput: { height: 100, textAlignVertical: 'top', }, }:` Por fim, é aplicado a um campo de entrada de mensagem um estilo. Ele define a altura (`height`) e alinha o texto verticalmente ao topo do campo (`textAlignVertical: 'top'`).

Dessa maneira, ao ser executado em um dispositivo *mobile* utilizando o Expo, é apresentado o seguinte resultado:

Figura 14 – Resultado do aplicativo de formulário



The image shows a mobile application interface for a simple form. The title "Formulário Simples" is centered at the top. Below the title are three input fields. The first field contains the text "Teste". The second field contains the email address "Teste@gmail.com". The third field contains the text "Teste" followed by a green vertical cursor. Below the input fields is a blue button with the text "ENVIAR" in white capital letters.

**Formulário Simples**

Teste

Teste@gmail.com

Teste

ENVIAR

Fonte: Do próprio autor, 2024

Figura 15 – Resultado do aplicativo de formulário após pressionamento do botão



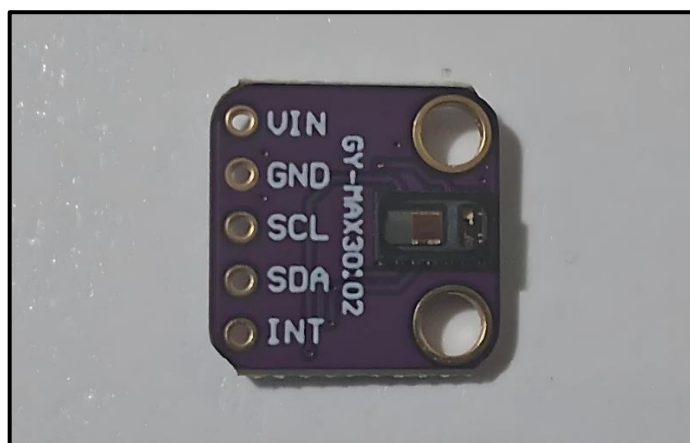
Fonte: Do próprio autor, 2024



### 2.10 Sensor MAX30102

Responsável pela medição da frequência cardíaca, o MAX30102, como explica Pascoal (2020), é um oxímetro e sensor de batimentos que prioriza a utilização em projetos portáteis, minimizando seu tamanho e mantendo a eficiência.

Figura 17 – Sensor MAX30102



Fonte: Do próprio autor, 2024

### 2.11 Battery Shield V8

Mencionado por Santos (2021), a Battery Shield V8 é uma *shield* eletrônico que integra duas baterias recarregáveis. Essas baterias oferecem um fornecimento de 6800 mAh e 3,7 V, proporcionando uma fonte de energia confiável e duradoura para o esp32.

### 2.12 C

O C é uma linguagem de programação de alto nível criada por Dennis Ritchie, de uso geral, portátil e multiplataforma, permitindo a criação de programas complexos e acessíveis a todos os tipos de máquinas, como explicado por Damas (2016).

De grande popularidade, mesmo considerada de difícil aprendizado, possui influência em diversas linguagens diferentes, como o Java, PHP e C++, aponta Backes (2023).

### 2.13 C++

Segundo Filho (2010), o C++ é uma linguagem que tem como base o C, considerada um subconjunto dele, possuindo diversas melhorias, como a orientação a objeto, e facilitando a criação de aplicações.

Por ser uma linguagem de alto nível, a execução de seus programas exige um tradutor, responsável por torná-lo de baixo nível, linguagem presente nas máquinas, assim explica Aguilar (2008).

A seguinte imagem exibe um exemplo de um código desenvolvido com o C++ em um Arduino Uno:

Figura 18 – Programa para manipulação de LEDs

```
int valor;  
void setup()  
{  
  pinMode(6, OUTPUT);  
}  
void loop()  
{  
  valor = analogRead(A0);  
  if(valor > 500){  
    digitalWrite(6,HIGH);  
  }else{  
    digitalWrite(6,LOW);  
  }  
  delay(250);  
}
```

Fonte: do próprio autor, 2024

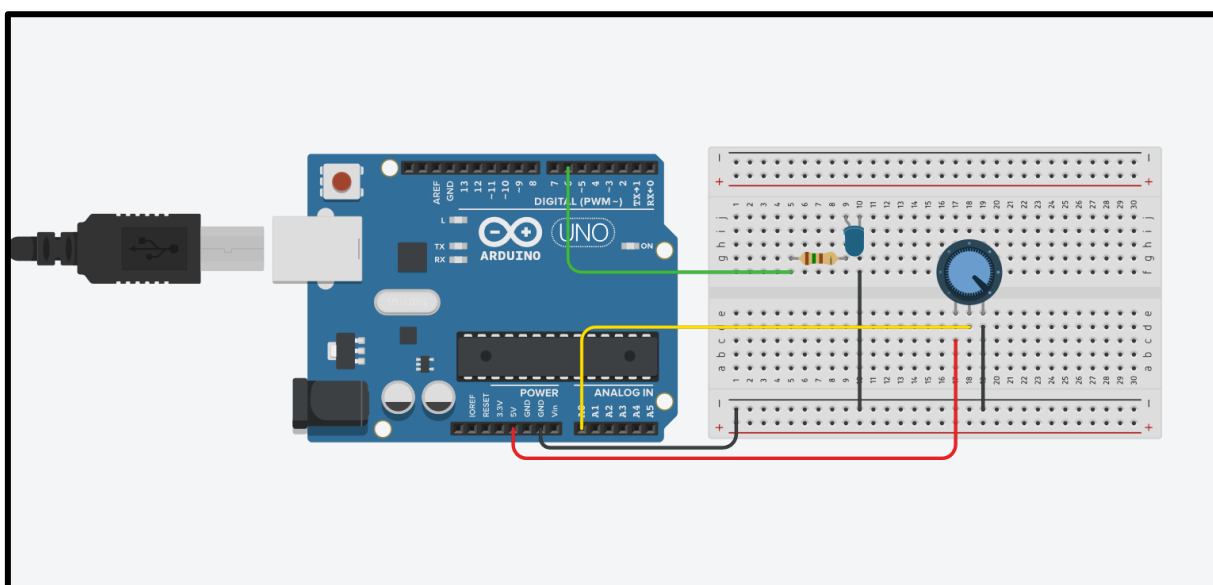
Esse exemplo é responsável pela manipulação de LEDs utilizando um potenciômetro, onde:



- `int valor`: declara uma variável, responsável por receber alguma informação.
- `void setup()`: função que declara configurações iniciais, como estado do LED, sendo ele ligado ou desligado.
- `pinMode`: declara qual porta será utilizada, além de definir como a informação será manipulada, seja ela uma entrada de dados ou saída.
- `void loop()`: função de ciclo responsável por executar todas as funcionalidades e comandos presentes nele, repetidamente.
- `Valor = analogRead(A0)`: variável recebe a porta pertencente ao potenciômetro.
- `if`: bloco de decisão para limitar o valor do potenciômetro, onde é definido o estado do LED, se o valor for maior que 500, a lâmpada acende. Caso seja menor, ela irá desligar.
- `digitalWrite(6, HIGH)`: define o LED como aceso.
- `digitalWrite(6, LOW)`: torna o LED apagado.
- `delay`: tempo de espera para execução da ação.

A figura abaixo exibe o resultado do código, onde o potenciômetro está desligado, deixando o LED apagado:

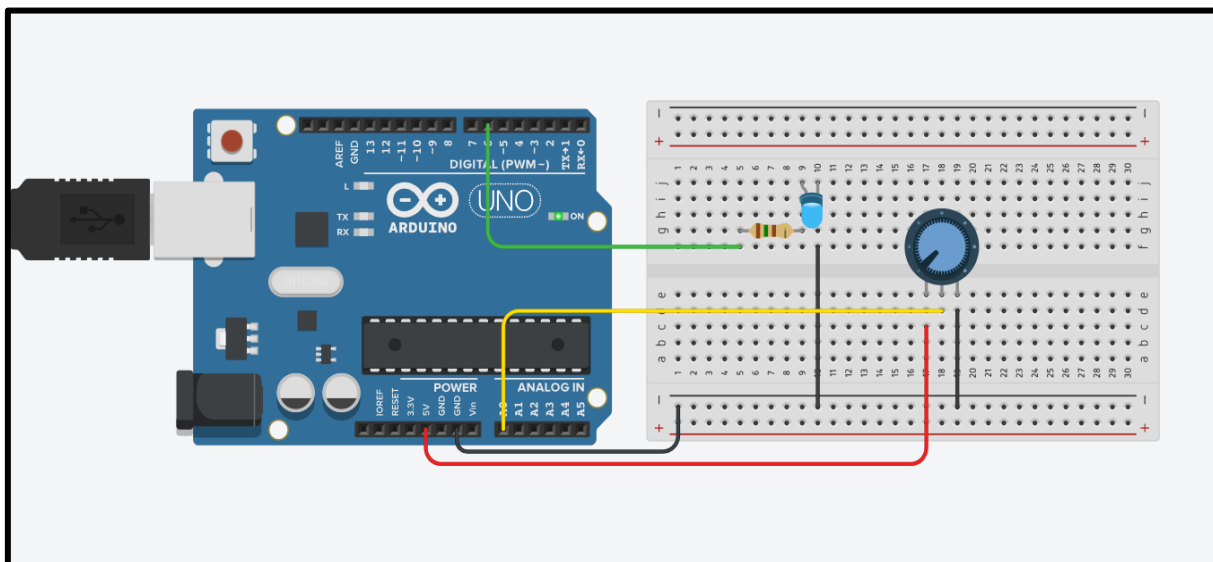
Figura 19 – Exemplo código C++ LED apagado



Fonte: do próprio autor, 2024

A seguir, o potenciômetro encontra-se ligado, tornando o LED aceso:

Figura 20 – Exemplo código C++ LED aceso



Fonte: do próprio autor, 2024

## 2.14 Wear OS

De acordo com Syozi (2024), O Wear OS, desenvolvido pela Google, é uma plataforma sofisticada para relógios inteligentes que abrange produtos desenvolvidos em conjunto com parceiras selecionadas.

O wear OS conecta o smartwatch ao smartphone. Assim, a pessoa pode usar o relógio para várias coisas, como ver mensagens importantes e rastrear sua saúde e condicionamento físico, diz Chaves (2023).

Conforme Developer (2024), os dispositivos wear são menores do que os dispositivos móveis, o que gera novos desafios. A utilização de uma tela de tamanho reduzido pode representar dificuldades para quem tem coordenação motora limitada.

## 2.15 Kotlin

O Kotlin é uma linguagem de programação dirigida a diferentes utilizações, possuindo uma sintaxe simples e de agradável utilização, além de entregar segurança ao usuário como aponta Jemerov e Isakova (2017).

Como explica Lecheta (2017), essa linguagem possui interoperabilidade com Java e outras, podendo executar todas as suas utilidades, além de permitir a utilização de suas bibliotecas de desenvolvimento.

### **2.15.1 Jetpack Compose**

De acordo com Developer (2024), O jetpack compose é um framework do android studio que se torna a criação de interfaces nativas mais fácil e mais rápida. Utilizando menos linhas de código.

O Jetpack Compose inclui a *Composable Function*. Elas são essenciais para a criação de uma interface de usuário. Essas funções devem conter tudo o que você quiser exibir na tela do seu *app*, Diz Murua (2023).

### **2.17 Banco de dados**

Um banco de dados, segundo Silberschatz, Korth e Sudarshan (2016), é considerado um grupo de informações organizadas com o objetivo de facilitar o acesso, controle e revisão desses dados.

Date (2004) insinua que as bases de dados são conhecidos por integrar os dados, diminuir as inconsistência e fornecer segurança aos dados. Os novos sistemas precisam dessas características para funcionar.

Elmasri e Navathe (2011) diz que os bancos de dados tanto relacionais, orientados a objetos e NoSQL foram desenvolvido para atender a diferentes requisitos e situações, oferecendo um leque de soluções caso necessário.

Nesse vizez utilizamos a solução de um banco NoSQL.

Assim como lembrado por Sadalage e Fowler (2013), os bancos de dados NoSQL pertencem a uma subcategoria de sistemas de gerenciamento de banco de dados que não usam o modelo relacional convencional.

### **2.18 Firebase**

Conforme mencionado pelo (Firebase, 2024), o serviço é uma plataforma de desenvolvimento de aplicativos fornecida pelo Google. Ela oferece uma variedade de serviços de nuvem e banco de dados, tanto para web quanto para dispositivos móveis.

O Firebase tem diversas bibliotecas específicas para cada ferramenta, simplificando o processo para os desenvolvedores, que só precisam identificar e instalar as bibliotecas necessárias como apresentado por Mezzari, Leal e Viegas (2019).

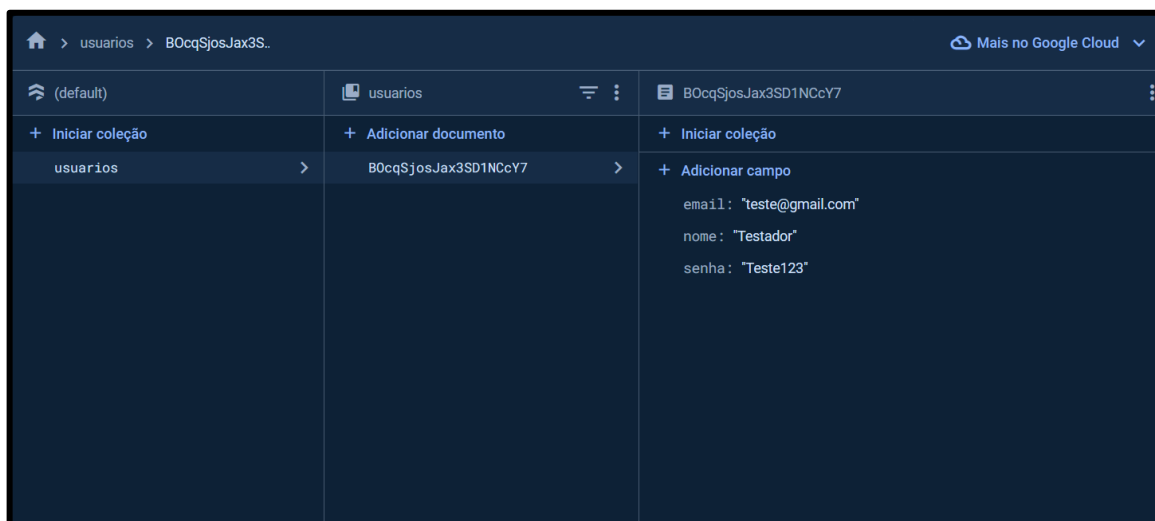
### 2.18.1 Firestore

Firestore é uma solução de banco de dados altamente versátil e dimensionável, parte integrante do Firebase. Utilizando uma abordagem não relacional ideal para o desenvolvimento de aplicativos móveis, como abordado por Araújo e Azevedo (2019).

Parafraseando Martins (2023), Ele desempenha um papel vital na gestão e armazenamento eficiente de informações. Graças a suas capacidades, é viável criar apps com escalabilidade e performance excepcionais.

Abaixo podemos observar uma ilustração da interface do Cloud Firestore:

Figura 21 – Exemplo de Estrutura do Firecloud



Fonte: do próprio autor, 2024

Na imagem é possível observar três sessões, a primeira é o painel central, que mostra o banco atual selecionado e a opção de iniciar uma nova coleção, que podemos compreender como um grupo de documentos dentro de um banco de dados, sendo análoga a uma tabela em um banco de dados relacional. O segundo apresenta documentos dentro da coleção “usuários” selecionada anteriormente, documentos

são uma unidade de armazenamento de dados com um código próprio, semelhante aos registros utilizados em unidades relacionais. Na terceira divisão é possível observar os campos exemplificado por “email”, “nome” e “senha”, seguido de seus respectivos valores, novamente semelhantes às colunas em uma tabela relacional. Observando essas funções é possível entender sua versatilidade e facilidade de uso para a criação de uma base de dados, facilitando os processos, anteriormente mais complexos em bancos de dados relacionas SQL.

Como lembrado por (Firebase, 2024), a ferramenta oferece suporte para regionalização do banco, disponibilizando um servidor em São Paulo, dessa maneira reduzindo a latência e aumentando a disponibilidade para projetos regionais.

Diante dessas informações optamos pelo Firebase, assim como o Cloud Firestore, devido a uma série de fatores. A plataforma é adequada para satisfazer as diversas e em constante mudança exigências do projeto, como apresentado por Oliveira (2023).

## **2.19 Impressão 3D**

A impressão 3D, ou manufatura aditiva, é o método de criação de um objeto tridimensional utilizando materiais para sua produção, através de uma máquina responsável por processar essa matéria e moldar o produto, explica Volpato (2017).

## **2.20 Modelagem 3D**

As impressões criadas são desenhadas previamente em um *software* específico para o desenvolvimento de peças tridimensionais. Esse processo de planejamento é chamado de modelagem 3D, assim conclui a Secretaria da Educação (2018).

Essas ferramentas, conhecidas como sistemas CAD (Computer Aided-Design), fornecem diversas utilidades para a facilitação da criação e visualização de todo o desenvolvimento dos componentes, como relata AutoDesk (2024).

## **2.21 Prototipação**

O autor Texeira (2014), informa que os *wireframes* são guias visuais utilizados para representar as estruturas das visualizações, assim como os elementos que as integram.

Descrito por Busarello, Bieging e Ulbricht (2013), tornam-se junções de ideias para a construção de protótipos das funcionalidades de tais componentes para a usabilidade do usuário.

Conforme destacado por Corrêa (2011), a primeira etapa da prototipação da interface envolve um visual simples (baixa fidelidade), seguido pelo refinamento do resultado com mais detalhes, podendo ser sua versão final (alta fidelidade).

Logo abaixo temos um exemplo de um Wireframe de Baixa Fidelidade:

Figura 22 – Exemplo de Prototipação da Interface de Baixa Fidelidade

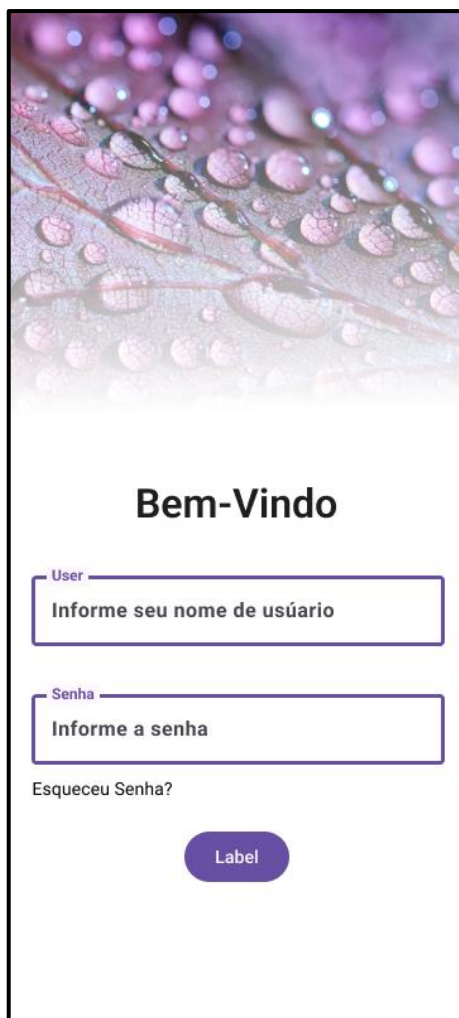
Este wireframe de baixa fidelidade representa uma interface de login. A estrutura é a seguinte:

- Um cabeçalho cinza sólido na parte superior.
- O título "Bem-Vindo" centralizado em uma seção branca.
- Dois campos de entrada cinza para "Informe seu nome de usuário" e "Informe a senha", dispostos verticalmente.
- O link "Esqueceu Senha?" localizado abaixo do campo de senha.
- Um botão cinza "Entrar" centralizado na base da seção de login.

Fonte: do próprio autor, 2024

E em seguida, o Prototipação de Alta Fidelidade, ilustrando a versão final da aplicação:

Figura 23 – Exemplo de Prototipação da Interface de Alta Fidelidade



The image shows a high-fidelity prototype of a mobile application's login screen. The top half of the screen features a background image of water droplets on a leaf, with a purple-to-white gradient overlay. Below the image, the text "Bem-Vindo" is centered in a bold, black font. Underneath, there are two input fields: the first is labeled "User" and contains the text "Informe seu nome de usuário"; the second is labeled "Senha" and contains the text "Informe a senha". Below these fields, the text "Esqueceu Senha?" is displayed. At the bottom, there is a purple button with the text "Label" in white.

Fonte: do próprio autor, 2024

## REFERENCIAS

AGUILAR, Luis Joyanes. **Programação em C ++**: Algoritmos, estruturas de dados e objetos. 2. ed. Porto Alegre: AMGH, 2011.

BACKES, André. **Linguagem C**: completa e descomplicada. 2. ed. Rio de Janeiro: GEN, LTC, 2023.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**: Guia do Usuário. 2. ed. Rio de Janeiro: Elsevier, 2012.

BRASIL. Secretaria de Estado da Educação do Paraná. Superintendência da Educação, Departamento de Políticas e Tecnologias Educacionais. **Impressão 3D: imaginar, planejar e materializar**. Paraná, 2018.

BUSARELLO, Raul Inácio; BIEGING, Patrícia; ULBRICHT, Vania Ribas. **Mídia e educação**: novos olhares para a aprendizagem sem fronteiras. São Paulo: Pimenta Cultural, 2013.

CARRIL, Marly. **HTML**: Passo a Passo. Joinville: Clube de Autores, 2012.

DAMAS, Luís. **Linguagem C**. 10. ed. Rio de Janeiro: GEN, LTC, 2016.

DATE, Christopher John. **Introdução a Sistemas de Bancos de Dados**. 8. ed. Rio de Janeiro: Elsevier, 2004.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Bancos de Dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

ESCUDELARIO, Bruna de Freitas; PINHO, Diego Martins. **React Native**: Desenvolvimento de Aplicativos mobile com React. 2020. E-book.

FILHO, Antonio Mendes da Silva. **Introdução à programação orientada a objetos com C++**. Rio de Janeiro: Elsevier, 2010.

FLANAGAN, David. **JavaScript: o guia definitivo**. 6. ed. Porto Alegre: Bookman, 2013.



FLATSCHART, Fábio. **HTML 5: Embarque Imediato**. Rio de Janeiro: Brasport, 2011.

FOWLER, Martin et al. **UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos**. 3. ed. Porto Alegre: Bookman, 2005.

GUEDES, Guilleanes Thorwald Araujo. **UML 2: Uma Abordagem Prática**. 3. ed. São Paulo: Novatec Editora, 2018.

JEMEROV, Dmitry; ISAKOVA, Svetlana. **Kotlin em Ação**. São Paulo: Novatec Editora, 2017.

JERICÓ, Márcia Marques; NETO, João Pedro de Andrade; KOGIKA, Márcia Mery. **Tratamento de Medicina Interna de Cães e Gatos**. Rio de Janeiro: Roca, 2015.

JOBSTRAIBIZER, Flávia. **Criação de Sites com CSS: Desenvolva páginas Web mais leves e dinâmicas em menos tempo**. São Paulo: Digerati Books, 2009.

JOBSTRAIBIZER, Flávia. **Criação de sites com o CSS: Desenvolva páginas Web mais leves e dinâmicas em menos tempo**. São Paulo: Digerati Books, 2009.

LECHETA, Ricardo Rodrigues. **Android Essencial com Kotlin**. 2. ed. São Paulo: Novatec Editora, 2018.

MAGRANI, Eduardo. **A Internet das Coisas**. Niterói: Cândido, 2021.

MILETTO, Evandro Manara; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP**. Porto Alegre: Bookman, 2014.

MORAIS, José V. M. **ESP32 com IDF: O Guia Profissional**. São Paulo: Instituto Newton C Braga, 2023.

MORRISON, Michael. **Use a cabeça! JavaScript**. Rio de Janeiro: Alta Books, 2008.

SADALAGE, Pramod J.; FOWLER, Martin. **NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota**. São Paulo: Novatec Editora Ltda, 2013.

SANTOS, Sandro. **Introdução à IoT: Desvendando a Internet das Coisas**. Joinville: Clube dos Editores, 2019.

SANTOS, Wallef Ferreira. **Equipamento de baixo custo para monitorar temperatura e umidade de forma contínua e remota: aplicação na compostagem**. 2023. 53 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Ambiental e Sanitária) - Centro de Tecnologia, Universidade Federal de Alagoas, Maceió, 2021.

SCHEIDT, Felipe Alex. **Fundamentos de CSS: Criando Design para Sistemas Web**. Paraná: Outbox Interativa, 2015.

SILBERSCHATZ, Abraham; SUNDARSHAN, S.; KORTH, Henry F. **Sistema de Banco de Dados**. Rio de Janeiro: Elsevier, 2016.

SILVA, Maurício Samy. **Criando Sites com HTML: Sites de Alta Qualidade com HTML e CSS**. São Paulo: Novatec Editora Ltda, 2008.

SILVA, Maurício Samy. **React: Aprenda Praticando** - Desenvolva aplicações web reais com uso da biblioteca React e de seus módulos auxiliares. São Paulo: Novatec Editora Ltda, 2021.

SINCLAIR, Bruce. **IoT: Como Usar a "Internet Das Coisas" Para Alavancar Seus Negócios**. São Paulo: Autêntica Business, 2018.

TEIXEIRA, Fabrício. **Introdução e boas práticas em UX Design**. São Paulo: Casa do Código, 1998.

VOLPATO, Neri. **Manufatura Aditiva: Tecnologias e aplicações da impressão 3D**. São Paulo: Blucher, 2017.

ZAKAS, Nicholas Charles. **Princípios de Orientação a Objetos em JavaScript**. São Paulo: Novatec Editora, 2017.

ZELENOVSKY, Ricardo; MENDONÇA, Alexandre. **Microcontroladores: Programação e Projeto com a família 8051**. Rio de Janeiro: MZ Editora Ltda, 2017.

ARAÚJO, Guilherme Ianhes; AZEVEDO, Rodolfo Jardim. **Implementação de um Sistema de Quiz no Aplicativo WebLectures**. 2019. Relatório Técnico (Graduação em Computação) – Universidade Estadual de Campinas, Campinas, 2019.

BEZERRA, Franklyn Seabra Rogério; VIANA, Wilson. **Desenvolvimento Nativo vs Ionic vs React Native: uma análise comparativa do suporte à acessibilidade em Android**. 2021. TCC (Graduação em Computação) - Universidade Federal do Ceará, Fortaleza, 2021.

CORRÊA, Daniel Felipe Bernardino. **O Papel da Arquitetura de Informação na Experiência do Usuário**. 2011. TCC (Especialização em Arquitetura e Organização da Informação) – Universidade Federal de Minas Gerais, Belo Horizonte, 2011.

FRANCO, Matheus Felipe Souza. **Percepção de tutores sobre a saúde e qualidade de vida de cães cardiopatas**. 2022. TCC (Bacharelado em Medicina Veterinária) - Universidade Federal da Fronteira Sul, Realeza, 2022.

MARTINS, João Antônio Bandeira De Oliveira. **Uma ferramenta para otimizar a relação instrutor-aluno em academias**. 2023. TCC (Bacharelado em Ciência da Computação) – Universidade Federal de Campina Grande, Campina Grande, 2023.

OLIVEIRA, João Olívio Scaramussa Fávero. **Desenvolvimento de uma solução com gamificação para estudos de engenharia de requisitos**. 2023. TCC (Bacharelado em Sistemas da Informação) – Instituto Federal do Espírito Santo, Cachoeiro de Itapemirim, 2023.

PASCOAL, Pedro Gelati. **Desenvolvimento de um sistema para monitoramento da frequência cardíaca em atividades físicas**. 2020. TCC (Graduação em Engenharia Elétrica) – Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Ijuí, 2020.

MEZZARI, Lucas Torres; LEAL, Eduardo Henrique Viva; VIEGAS, Silvio. Internet das Coisas: Arduino, Firebase e Android. **Gestão e Tecnologia**. Gravataí, v. 5, n. 1, set. 2019.

SILVA, Denys Alves; SOUSA, Caio Frias. Construção de App com React Native. **Tecnologias em Projeção**. Brasília, v. 10, n. 1, ago. 2019.

ALURA. **Como instalar e configurar o Expo do React Native**. Disponível em: <https://www.alura.com.br/artigos/como-instalar-configurar-expo-do-react-native>. Acesso em: 07 de mai. 2024

AUTODESK. **Software CAD para arquitetos, engenheiros e projetistas**. Disponível em: <https://www.autodesk.com.br/solutions/cad-software>. Acesso em: 16 de mai. 2024

CHAVES, Matheus. **Quais smartwatches da Samsung tem Wear OS?** Disponível em: <https://olhardigital.com.br/2023/09/07/dicas-e-tutoriais/quais-smartwatches-da-samsung-tem-wear-os/>. Acesso em: 26 de abr. 2024

DEVELOPER. **Acessibilidade no Wear OS.** Disponível em: <https://developer.android.com/training/wearables/accessibility?hl=pt-br>. Acesso em: 07 de mai. 2024

DEVELOPER. **Crie apps melhores com mais rapidez usando o Jetpack Compose.** Disponível em: <https://developer.android.com/develop/ui/compose?hl=pt-br>. Acesso em: 16 de mai. 2024.

FELDMAN, Andre. **Cardiopatias:** o que é, sintomas, causas, tipos e tratamento. 2024. Disponível em: <https://www.tuasaude.com/cardiopatia>. Acesso em: 10 de abr. 2024

FIREBASE. **Armazene e sincronize dados em tempo real.** Disponível em: [https://firebase.google.com/products/realtime-database?utm\\_source=bing&utm\\_medium=cpc&utm\\_campaign=latam-BR-all-pt-dr-SKWS-all-all-trial-e-dr-1707800-LUAC0016441&utm\\_content=text-ad-none-any-DEV\\_c-CRE\\_-ADGP\\_Hybrid%20%7C%20SKWS%20-%20MIX%20%7C%20Txt\\_%20Compute-Firebase-KWID\\_43700067403163247-kwd-78684157082834%3Aloc-20&utm\\_term=KW\\_Firebase-ST\\_Firebase&hl=pt-br](https://firebase.google.com/products/realtime-database?utm_source=bing&utm_medium=cpc&utm_campaign=latam-BR-all-pt-dr-SKWS-all-all-trial-e-dr-1707800-LUAC0016441&utm_content=text-ad-none-any-DEV_c-CRE_-ADGP_Hybrid%20%7C%20SKWS%20-%20MIX%20%7C%20Txt_%20Compute-Firebase-KWID_43700067403163247-kwd-78684157082834%3Aloc-20&utm_term=KW_Firebase-ST_Firebase&hl=pt-br). Acesso em: 27 de abr. 2024

MURUA, Daniely. **Conhecendo o Jetpack Compose.** Disponível em: <https://medium.com/wearejaya/conhecendo-o-jetpack-compose-d0ca4398e5c7>. Acesso em: 16 de mai. 2024

REACT. **React:** A biblioteca para web e interfaces de usuário nativas. Disponível em: <https://pt-br.react.dev>. Acesso em: 21 de abr. 2024

SANTOS, Rullyan Gabriel. **ESP32.** Disponível em: <https://deinfo.uepg.br/~alunoso/2019/SO/ESP32/HARDWARE/>. Acesso em: 15 de mai. 2024

SYOZI, Ricardo. **O que é Wear OS?** Disponível em: <https://canaltech.com.br/apps/o-que-e-wear-os/>. Acesso em: 23 de abr. 2024

CUNHA, Andre. **Como instalar e configurar o Expo do React Native**. Disponível em: <https://www.alura.com.br/artigos/como-instalar-configurar-expo-do-react-native>. Acesso em: 31 maio. 2024.