



# Exemplo prático IMC

Ewerton J. Silva

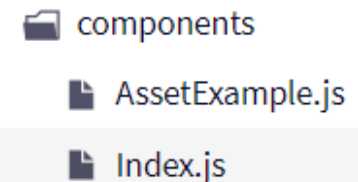
[ewerton.silva41@etec.sp.gov.br](mailto:ewerton.silva41@etec.sp.gov.br)

# Criando componente principal

```
1 import React, { useState } from 'react';
2 import { Text, View, StyleSheet, TextInput, TouchableOpacity } from 'react-native';
3
4 export default function Index() {
5   return (
6
7     <View style={styles.container}>
8       <Text style={styles.paragraph}> Exemplo 6 </Text>
9     </View>
10
11   );
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     justifyContent: 'center',
18     backgroundColor: '#EEE',
19     padding: 8,
20   },
21   paragraph: {
22     margin: 6,
23     fontSize: 18,
24     fontWeight: 'bold',
25     textAlign: 'center',
26     color: '#AAA',
27   },
28 });
```

Em um novo projeto dentro da pasta “components” crie um novo arquivo com o nome “Index.js” e insira o código apresentado ao lado.

Este componente vai retornar uma View com um Text dentro.



```
components
├── AssetExample.js
└── Index.js
```

Em “App.js”, vamos importar o componente “Index.js” e utilizá-lo dentro do “App.js”.

Deixe o código de “App.js” conforme o exemplo ao lado, aqui iremos retornar uma “View” com o componente criado anteriormente.

```
1  import * as React from 'react';
2  import { View, StyleSheet } from 'react-native';
3  import Constants from 'expo-constants';
4
5  import Index from './components/Index';
6
7  export default function App() {
8    return (
9      <View style={styles.container}>
10        <Index />
11      </View>
12    );
13  }
14
15  const styles = StyleSheet.create({
16    container: {
17      flex: 1,
18      justifyContent: 'center',
19      paddingTop: Constants.statusBarHeight,
20      backgroundColor: '#AAA',
21      padding: 8,
22    },
23  });
```

Exemplo 6

# Introdução

Vamos criar um programa que receba o peso e a altura de uma pessoa e apresente o valor do IMC.

## Índice de Masa Corporal

$$\text{IMC} = \frac{\text{Peso (Kg)}}{\text{Altura (m)}^2}$$

<https://www.calculoexato.net/calculadora-imc/>

# Componentes de entrada de dados

Adicione 2 “TextInput” dentro de uma “View”, estes irão receber os valores da massa e altura, utilizaremos nele as propriedades “placeholder” que apresenta um texto dentro do “TextInput”, “keyboardType” para definir que o tipo de teclado será apresentado quando a aplicação for utilizada em dispositivos com tela “Touch” e “style” para passar um objeto de estilização JSX.

```
<View style={styles.container}>
  <Text style={styles.paragraph}> Exemplo 6 </Text>
  <View style={styles.entradaImc}>
    <TextInput placeholder='Massa' placeholderTextColor='lightgray' keyboardType='numeric' style={styles.input} />
    <TextInput placeholder='Altura' placeholderTextColor='lightgray' keyboardType='numeric' style={styles.input} />
  </View>
</View>
```

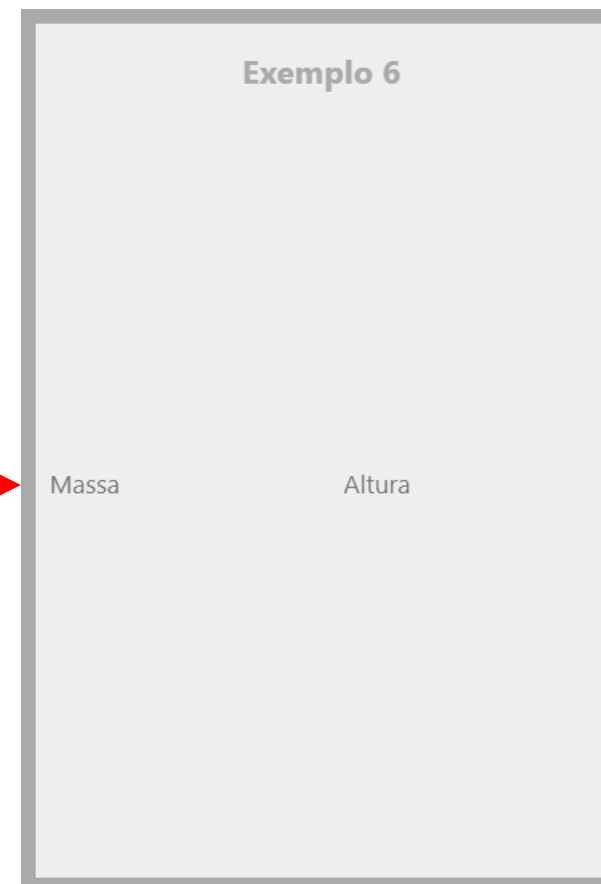
Veja sobre as opções disponíveis na propriedade keyboardType em:

<https://lefkowitz.me/visual-guide-to-react-native-textinput-keyboardtype-options/>

# Mudando posicionamento dos objetos no container.

No “StyleSheet” adicione um novo objeto com o nome “entradaImc”, nele vamos mudar a forma que os objetos estão dispostos dentro do container com a propriedade `flexDirection` que por padrão tem o valor “column”.

```
},  
entradaImc: {  
  flex: 1,  
  flexDirection: 'row',  
},
```



Saiba mais sobre flexbox no react native em:

<https://reactnative.dev/docs/flexbox>

# Estilizando TextInput

Ainda no “StyleSheet” insira outro objeto, onde iremos configurar o TextInput, este terá tamanho 80 para sua altura, seu texto centralizado, largura de 50% da tela, fonte com tamanho 50 e margem acima dele de 24.

```
},  
input: {  
  height: 80,  
  textAlign: 'center',  
  width: '50%',  
  fontSize: 50,  
  marginTop: 24,  
  color: 'lightgray',  
},
```



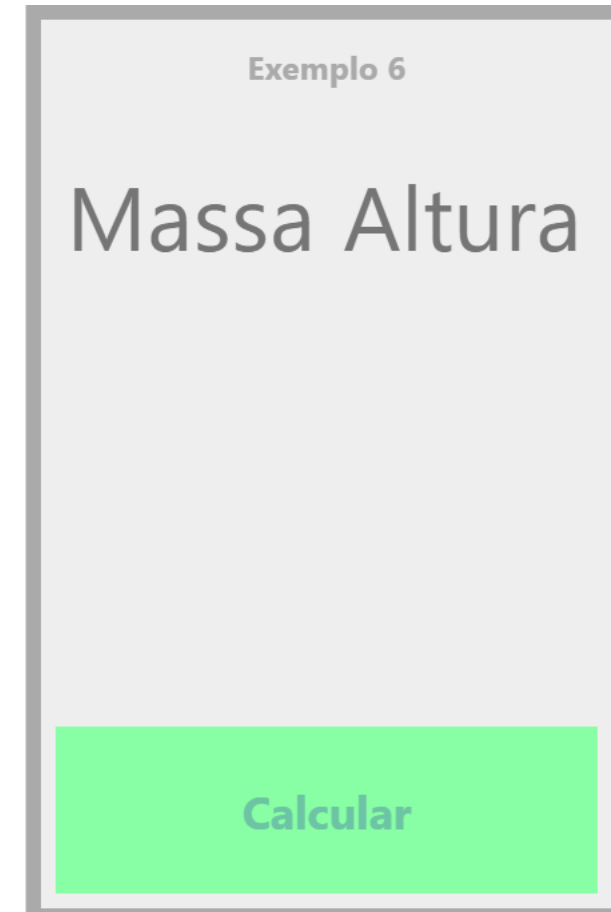
Após a “View” que foi inserida para permitir o alinhamento horizontal dos “TextInput” iremos definir um botão que será utilizado para calcular o valor do IMC.

```
<View style={styles.container}>
  <Text style={styles.paragraph}> Exemplo 6 </Text>
  <View style={styles.entradaImc}>
    <TextInput placeholder='Massa' keyboardType='numeric' style={styles.input} />
    <TextInput placeholder='Altura' keyboardType='numeric' style={styles.input} />
  </View>
  {
    <TouchableOpacity style={styles.button} onPress={() => {}} >
      <Text style={styles.buttonText}> Calcular </Text>
    </TouchableOpacity>
  }
</View>
```



Volte mais uma vez ao “StyleSheete” e defina os objetos de estilização para o botão e o texto do botão.

```
→ },  
→ button: {  
    backgroundColor: '#89FFA5',  
  },  
  buttonText: {  
    alignSelf: 'center',  
    padding: 30,  
    fontSize: 25,  
    color: '#6DC4A4',  
    fontWeight: 'bold',  
  },  
},
```



# Inserindo objeto para apresentação dos resultados

Insira um “Text”, onde apresentaremos o valor do resultado do cálculo.

```
<View style={styles.container}>
  <Text style={styles.paragraph}> Exemplo 6 </Text>
  <View style={styles.entradaImc}>
    <TextInput placeholder='Massa' keyboardType='numeric' style={styles.input} />
    <TextInput placeholder='Altura' keyboardType='numeric' style={styles.input} />
  </View>
  <TouchableOpacity style={styles.button} onPress={() => {}} >
    <Text style={styles.buttonText}> Calcular </Text>
  </TouchableOpacity>
  <Text style={styles.resultados}>Teste</Text>
</View>
```



# Estilizando a apresentação do resultado

Defina um objeto pra estilização do resultado.

```
},  
resultados: {  
  alignSelf: 'center',  
  color: 'lightgray',  
  fontSize: 65,  
  padding: 15,  
},
```



# Definindo state necessário para a aplicação

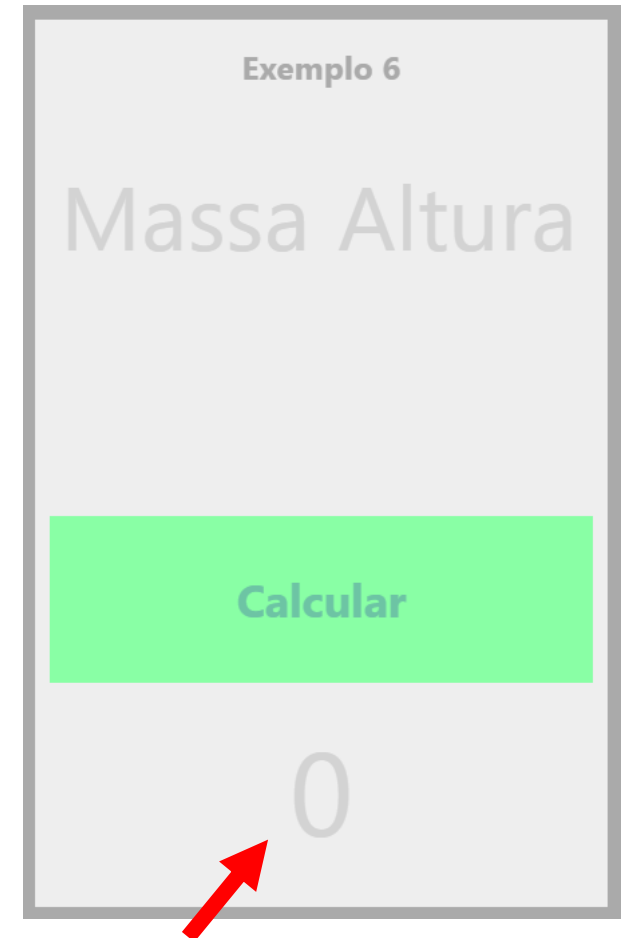
Agora precisamos definir os “state” para armazenar os valores digitados como massa e altura e o cálculo do resultado.

```
4   export default function Index() {  
5  
6       {  
7           const [massa, setMassa] = useState(0);  
8           const [altura, setAltura] = useState(0);  
9           const [resultado, setResultado] = useState(0);  
10      }  
11      return (  
12
```

# Exibindo valores do resultado

No lugar de um texto estático iremos apresentar o valor do “state” no resultado.

```
<Text style={styles.resultados}>{resultado}</Text>
```



# Passando valores da massa e altura para o state.

No evento onChangeText iremos inserir uma arrow function que passa o valor do TextInput como parâmetro e atualiza o valor.


Além disso, vamos aproveitar para alterar a propriedade “placeholderTextColor”, permitindo assim modificar a cor do texto apresentado no “placeholder”

```
<View style={styles.entradaImc}>
  <TextInput
    placeholder='Massa'
    placeholderTextColor='lightgray'
    keyboardType='numeric'
    style={styles.input}
    onChangeText={(entrada) => setMassa(entrada)}
  />
  <TextInput
    placeholder='Altura'
    placeholderTextColor='lightgray'
    keyboardType='numeric'
    style={styles.input}
    onChangeText={(entrada) => setAltura(entrada)}
  />
</View>
```

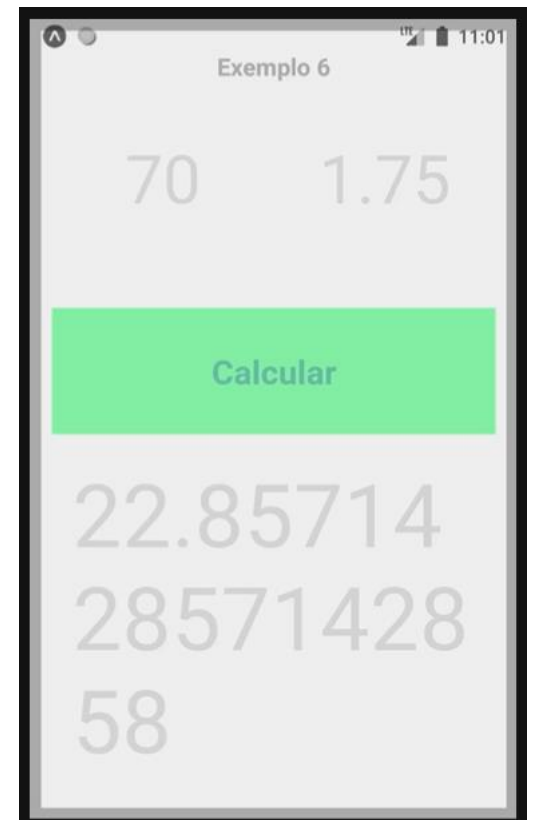
# Função para cálculo do IMC

Crie uma função sem parâmetro nem retorno para calcular o IMC, sua chamada será no evento “onPress” do “TouchableOpacity” do botão com o texto “Calcular”.

```
export default function Index() {  
  
  const [massa, setMassa] = useState(0);  
  const [altura, setAltura] = useState(0);  
  const [resultado, setResultado] = useState(0);  
  
  function Calcular() {  
    const valor = massa / (altura * altura);  
    setResultado(valor);  
  }  
  
  return (  
    <TouchableOpacity style={styles.button} onPress={() => Calcular()} >  
      <Text style={styles.buttonText}> Calcular </Text>  
    </TouchableOpacity>  
  )  
}
```



```
<TouchableOpacity style={styles.button} onPress={() => Calcular()} >  
  <Text style={styles.buttonText}> Calcular </Text>  
</TouchableOpacity>
```




# Definindo casas decimais

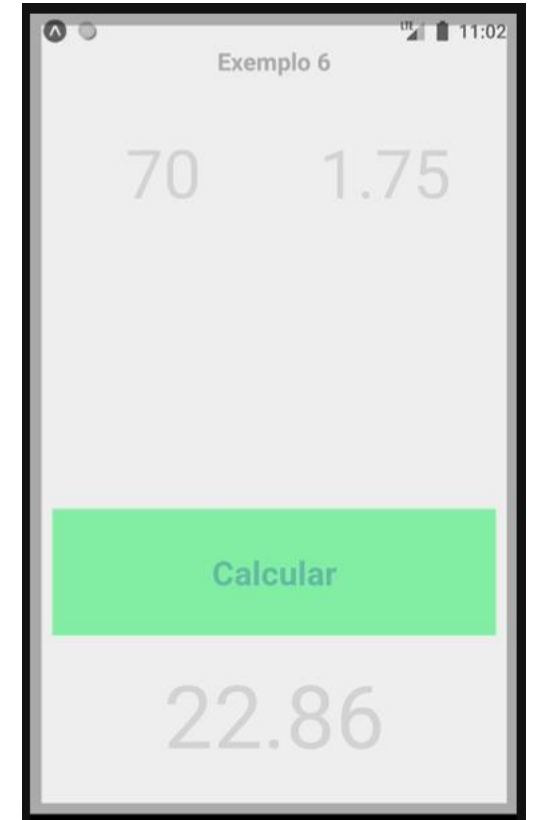
Para deixar o resultado com 2 casa decimais iremos utilizar o comando “toFixed” na apresentação do resultado.

Saiba mais sobre este comando em:

[https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global\\_Objects/Number/toFixed](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Number/toFixed)



```
<Text style={styles.resultados}>{resultado.toFixed(2)}</Text>
```





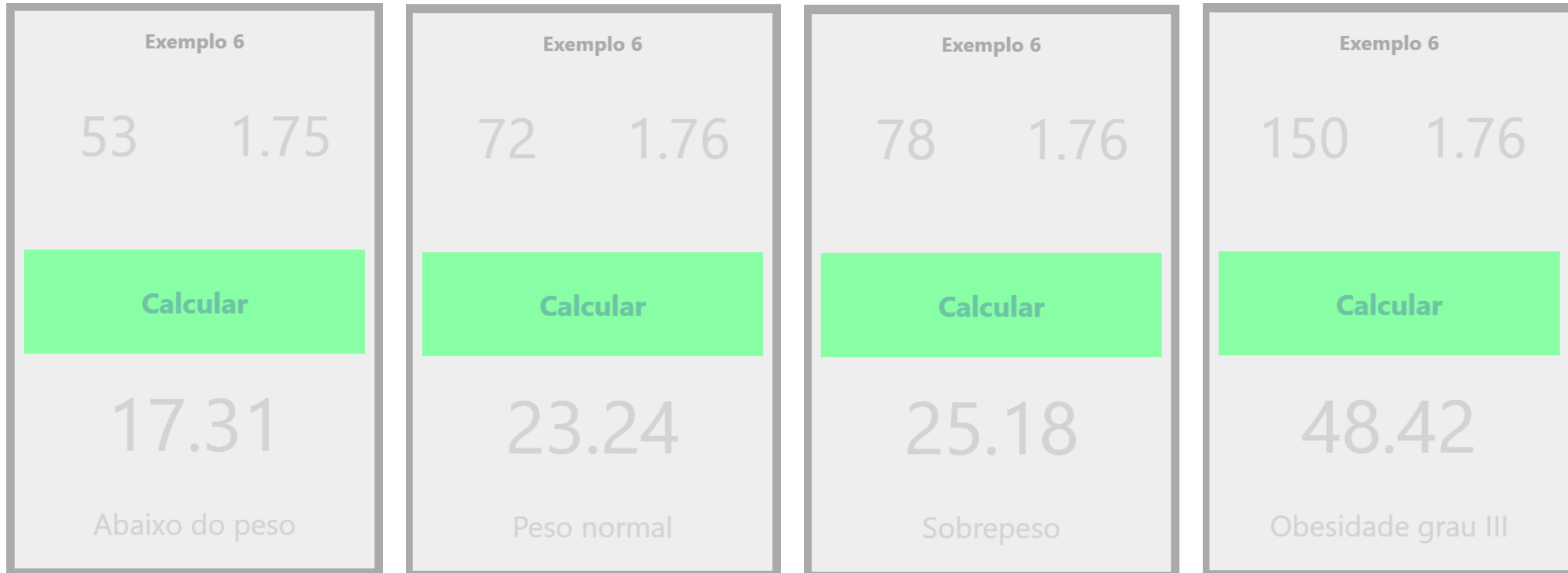
# Atividade

Faça um programa que receba a massa e o peso de uma pessoa e apresente o valor do IMC e o resultado correspondente ao valor.

IMC	Resultado
Menos do que 18,5	Abaixo do peso
Entre 18,5 e 24,9	Peso normal
Entre 25 e 29,9	Sobrepeso
Entre 30 e 34,9	Obesidade grau 1
Entre 35 e 39,9	Obesidade grau 2
Mais do que 40	Obesidade grau 3

<https://www.minhavidacom.br/alimentacao/tudo-sobre/32159-imc>

Ao finalizar teste todas as possibilidades no aplicativo para verificar se está funcionando corretamente.



# Referências

- <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Fun%C3%A7%C3%B5es>
- <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/function>
- [https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow functions](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow_functions)