

Vitor Amadeu Souza

Introdução ao

Mobile-First

com

React Native

No ambiente Expo Snack

© 2024 by Cerne Tecnologia e Treinamento Ltda.

© 2024 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Novembro de 2024

Direitos reservados por:

Cerne Tecnologia e Treinamento Ltda

Produção: Cerne Tecnologia e Treinamento

E-mail da Empresa: cerne@cerne-tec.com.br

Home Page: www.cerne-tec.com.br.com.br

Atendimento ao Consumidor: sac@cerne-tec.com.br

Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

Dedicatória

Aos alunos e alunas dos cursos de graduação em EDC e ADS, participantes da disciplina Mobile-First UI com React Native no período de 2024.4, cuja contribuição foi fundamental para a elaboração deste material.

*“A felicidade de sua vida depende da qualidade de seus
pensamentos.”*

Marco Aurélio

Cerne Tecnologia

A Cerne Tecnologia tem uma equipe preparada para desenvolvimento de projetos eletrônicos em diversas áreas: Médica, Entretenimento, Industrial, Robótica, Científica, Automobilística, Aeronáutica, etc. Trabalhamos com tecnologia microcontrolada usando o PIC, ARM, AVR, 8051, dsPIC, PIC24, PIC32 além do Arduino, Raspberry, Beaglebone etc. Desenvolvemos o projeto desde sua concepção até a entrega do produto final, passando pelas etapas de esquema elétrico, protótipo e desenvolvimento de circuito impresso.

Desenvolvemos aplicativos para smartphones/tablets Android, iOS, Blackberry, Windows Phone e no desenvolvimento de softwares a nível PC para plataforma Windows, usando ferramentas como o Visual Basic, C# e C++.

Atuamos na parte de montagem de placas, onde podemos fornecer ambos os serviços de desenvolvimento de projetos e produção ou apenas um destes.

Desenvolvemos esquemas elétricos e layout de PCI, tanto em tecnologia convencional como SMD.

Temos a flexibilidade de customizar um de nossos produtos, de modo a atender a uma necessidade específica do cliente, tornando o custo de desenvolvimento menor se comparado a construção de um projeto desde a sua fase inicial.

Desenvolvemos e fornecemos kits didáticos para diversos microcontroladores além de apostilas, livros e e-books.

Na hora de desenvolver um projeto ou equipar seu laboratório não hesite em nos contatar. Entre em contato conosco através do endereço cerne-tec.com.br para obter mais informações.



Sumário

Capítulo I – Metodologia de desenvolvimento.....	8
1. Introdução.....	8
Capítulo II – Programação em React Native.....	10
1. React Native.....	10
2. Plataformas de desenvolvimento.....	11
3. Componente View.....	17
4. Componente Text.....	20
5. Componente Image.....	23
6. Componente TextInput.....	25
7. Componente Alert.....	28
8. Componente TouchableHighlight.....	31
9. Componente ScrollView.....	34
10. Componente ScrollView II.....	38
11. Componente SectionList.....	43
12. Componente SafeAreaView.....	47
13. Componente FlatList.....	51
14. Componente Switch.....	56
15. Componente ActivityIndicator.....	61
16. Componente Slider.....	64
17. Componente Modal.....	68
18. Componente Picker.....	73
19. Componente Progress.Bar.....	77
20. Componente Animated.View.....	80
21. Componente ImageBackground.....	85
22. Componente Platform.....	88
23. Componente NavigationContainer.....	93
24. Componente useWindowDimensions.....	99
25. Exemplo Dark Mode.....	103
26. Links externos.....	109

27. Comunicação com API.....	112
28. Exemplo Calculadora.....	118
29. Envio de e-mail.....	123
30. Comunicação com API PokeAPI.....	129
31. Comunicação com API PokeAPI com Imagem.....	134
32. API de Geolocalização.....	139
33. API de Geolocalização com Google Maps.....	144
34. Animated Button.....	149
35. Async Storage.....	154
36. Formulário.....	161
37. Login.....	173
38. Header, Rodapé e Scroll.....	176
39. Componente Botão.....	180
40. React-navigation.....	186
41. Vector icons.....	191
42. Jogo de adivinhação.....	195
43. Lista de Tarefas.....	202
44. Citações aleatórias.....	205
45. Cronômetro.....	212
46. Contador de toques.....	216
47. MapView.....	219
48. Conexão com API.....	221
49. Quiz.....	225
50. Cores.....	232
51. Imagens aleatórias.....	234
52. Jogo da Velha.....	236
53. Jogo da força.....	240
54. Leitura de JSON.....	244
55. Módulo de estilo.....	248
56. Tela de Checkout.....	251
57. Carrossel Horizontal.....	260
58. Ampliando fotos.....	265

59. Drawer Screen.....	270
60. Drawer Screen II.....	274
61. Cadastro com 3 telas.....	279
62. Navegador com tela inicial.....	286
63. Navegador com tela inicial de login.....	291
64. Tela de Perfil.....	298
65. Instrumentos de Orquestra Sinfônica.....	303
66. Histograma.....	308
67. Acesso por biometria.....	313
68. Tirando foto e enviando por e-mail.....	317
69. Filtro e busca.....	322
70. Seleção de filmes.....	327
71. Detecção e gestos.....	335
72. Chamada a um telefone.....	339
73. Envio de SMS.....	343
74. Trocando a cor da tela.....	347
75. Sintetizador de voz.....	350
76. Sintetizador de voz em outras línguas.....	353
77. Youtube.....	357
78. Lista de tarefas.....	360
79. Carteira digital.....	365
80. App de exercícios4. Componente Text.....	374
81. Lista de Compras Colaborativa.....	379
82. Agenda de contatos.....	385
83. Bloco de notas.....	392
84. Conversor de unidades.....	397
85. Catálogo de Livros.....	403
86. Queima de calorias.....	408
87. Comunicação com API do Mercado Livre.....	413
88. Jogo gráfico.....	418

Capítulo I

Metodologia de desenvolvimento

1. Introdução

Este livro é uma porta de entrada para um vasto e abrangente guia sobre desenvolvimento com React Native, proporcionando aos leitores uma base sólida e um entendimento prático das ferramentas e recursos que esta tecnologia oferece. O objetivo é facilitar o aprendizado e a aplicação de conceitos essenciais para criar aplicações móveis modernas e responsivas.

O capítulo II é dedicado ao estudo dos componentes fundamentais como View, Text, Image, TextInput e muitos outros, explorando suas funcionalidades e aplicações específicas. O livro avança para exemplos práticos que incluem desde a comunicação com APIs até o desenvolvimento de interfaces interativas e elementos visuais sofisticados, como o Animated Button e carrosséis horizontais. Em sua essência, este trabalho pretende não só ensinar a codificar, mas capacitar os leitores a desenvolverem soluções completas, como aplicativos de geolocalização integrados com o Google Maps, ferramentas de acesso por biometria, sintetizadores de voz e até jogos clássicos como Jogo da Velha e Jogo da Forca.

Com a leitura, espera-se que os desenvolvedores adquiram uma proficiência que permita tanto a criação de aplicativos simples quanto de aplicações mais complexas, que vão desde listas de tarefas e agendas de contatos até carteiras digitais e aplicativos de exercícios.

Este livro é, portanto, uma jornada de aprendizado prático e detalhado, destinada a todos que desejam dominar o desenvolvimento de aplicativos móveis com React Native, utilizando o ambiente Expo Snack como uma plataforma de testes e desenvolvimento contínuo.

Capítulo II

Programação em React Native

1. React Native

React Native é um framework de desenvolvimento de aplicativos móveis que utiliza JavaScript e os princípios do React para criar aplicações nativas para iOS e Android. Desenvolvido pelo Facebook, ele permite que os desenvolvedores escrevam um único código que pode ser executado em múltiplas plataformas, resultando em uma experiência de usuário mais fluida e próxima das aplicações nativas. Ao contrário de soluções híbridas que utilizam webviews, o React Native renderiza componentes nativos, proporcionando melhor desempenho e acesso direto aos recursos do dispositivo.

Por outro lado, React (ou ReactJS) é uma biblioteca JavaScript voltada para a construção de interfaces de usuário em aplicações web. Ele se concentra na criação de componentes reutilizáveis que manipulam o DOM virtual, permitindo atualizações eficientes da interface sem a necessidade de recarregar a página. O React é ideal

para desenvolver aplicações web dinâmicas e complexas, onde a interação do usuário e a manipulação do estado são essenciais.

As principais diferenças entre React e React Native incluem o ambiente de execução, onde React é utilizado para desenvolvimento web e React Native é voltado para aplicativos móveis. Além disso, os componentes utilizados são diferentes: enquanto o React utiliza elementos HTML e CSS, o React Native utiliza componentes nativos específicos de cada plataforma. A renderização também varia, já que o React manipula o DOM virtual para renderizar a interface, enquanto o React Native interage diretamente com APIs nativas. Essas distinções tornam cada ferramenta adequada para diferentes tipos de projetos, dependendo das necessidades específicas de desenvolvimento web ou móvel.

2. Plataformas de desenvolvimento

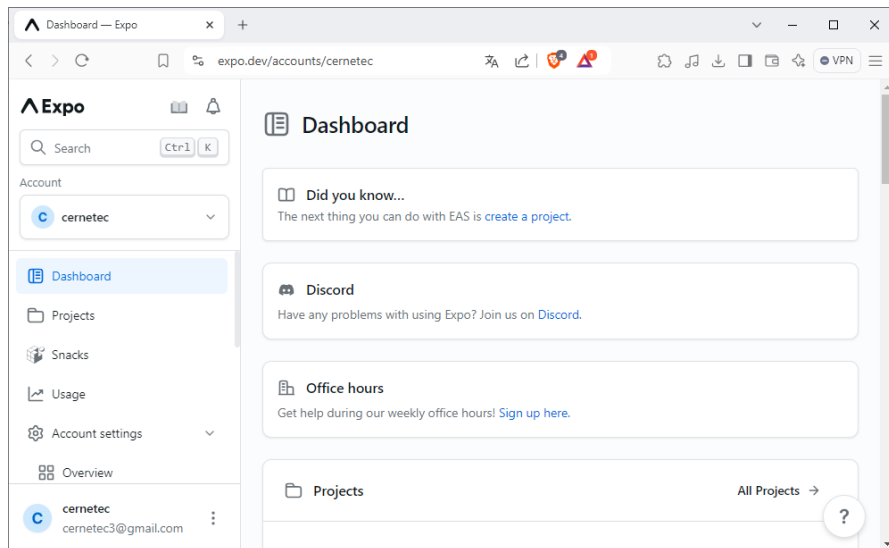
React Native é um framework de desenvolvimento de aplicativos móveis que permite criar aplicações nativas para iOS e Android utilizando JavaScript e os princípios do React. Para desenvolver com React Native, existem várias plataformas e ferramentas disponíveis, incluindo Android Studio, Xcode e Expo Snack.

Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android. Ele oferece uma série de ferramentas avançadas, como um emulador para testar aplicativos em diferentes dispositivos e configurações, além de recursos de depuração robustos. Para usar React Native no Android Studio, é necessário configurar o ambiente, instalar o Android SDK e definir variáveis de ambiente como `ANDROID_HOME` e `JAVA_HOME`. Isso permite que os desenvolvedores integrem facilmente funcionalidades específicas do Android em suas aplicações.

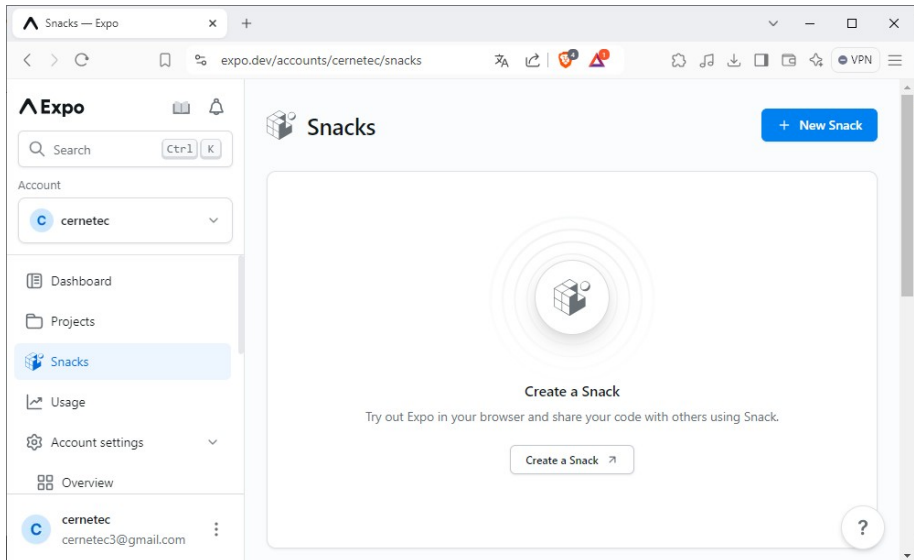
Xcode, por outro lado, é a IDE oficial da Apple para desenvolvimento de aplicativos iOS. Assim como o Android Studio, o Xcode fornece um conjunto completo de ferramentas para criar, testar e depurar aplicativos iOS. O uso do Xcode é essencial para quem deseja desenvolver aplicativos nativos para iPhones e iPads utilizando React Native, pois ele permite acesso a APIs nativas da Apple e facilita a publicação na App Store.

Expo Snack é uma plataforma online que simplifica o desenvolvimento com React Native, permitindo que os desenvolvedores criem, testem e compartilhem aplicativos

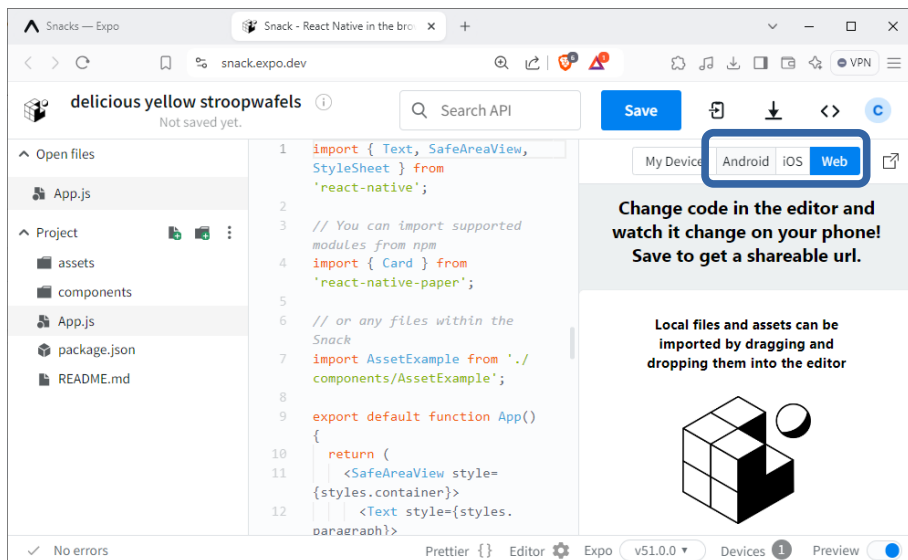
diretamente no navegador. O Expo Snack elimina a necessidade de configuração complexa do ambiente local, oferecendo uma interface intuitiva onde as alterações no código são refletidas em tempo real em um simulador ou dispositivo móvel. Essa ferramenta é especialmente útil para iniciantes ou para prototipagem rápida, pois permite experimentar com código sem a necessidade de instalação de software adicional. Essa ferramenta será usada ao longo desta série. Para acessá-la, visite <https://expo.dev/> e faça o seu login na plataforma. Após este passo, clique em *Snacks* conforme ilustra a próxima figura.



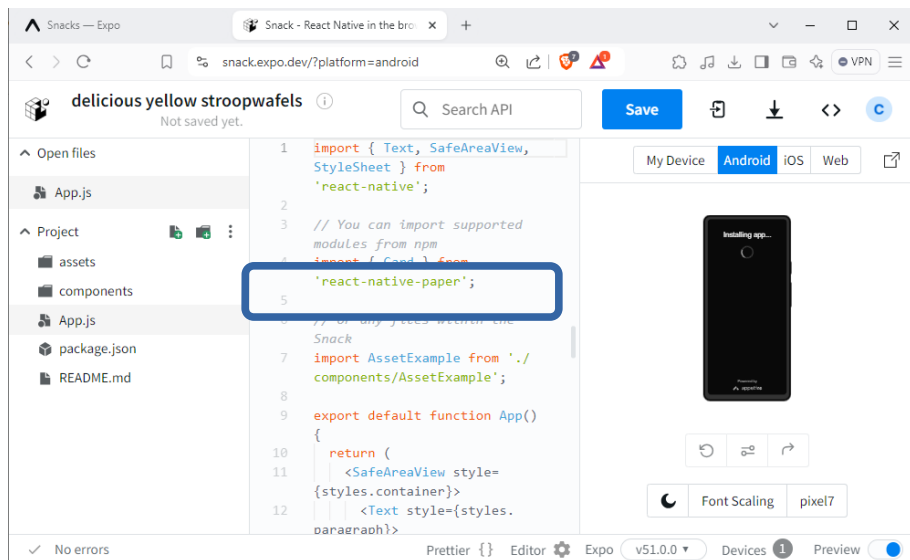
A próxima tela será carregada. Clique no botão *+ New Snack*.



A próxima tela será carregada. Note que um código default é apresentada. Também é possível escolher a tela no qual a aplicação será renderizada, podendo ser Android ou iOS.



A próxima tela mostra a aplicação *default* em funcionamento, onde basta para isso clicar no botão *Launch App* para uma das ferramentas.



Para mudar o nome do projeto basta modificar a área hachurada acima e em seguida o botão *Save*. Outra opção para simular é instalar o app Expo Go no seu smartphone em seguida abrir o mesmo e ler o QR Code gerado através da opção *My Device*, ao lado da opção Android do simulador.

3. Componente View

Este primeiro experimento apresenta o componente View. Para isso, crie um novo *Snack* e edite o arquivo App.js como apresentado a seguir.

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

const ExemploView = () => (
  <View style={{padding:20,marginTop:100,backgroundColor:'#f0f0f0'}}>
    <Text>Este é um exemplo do componente View.</Text>
  </View>
);

export default ExemploView;
```

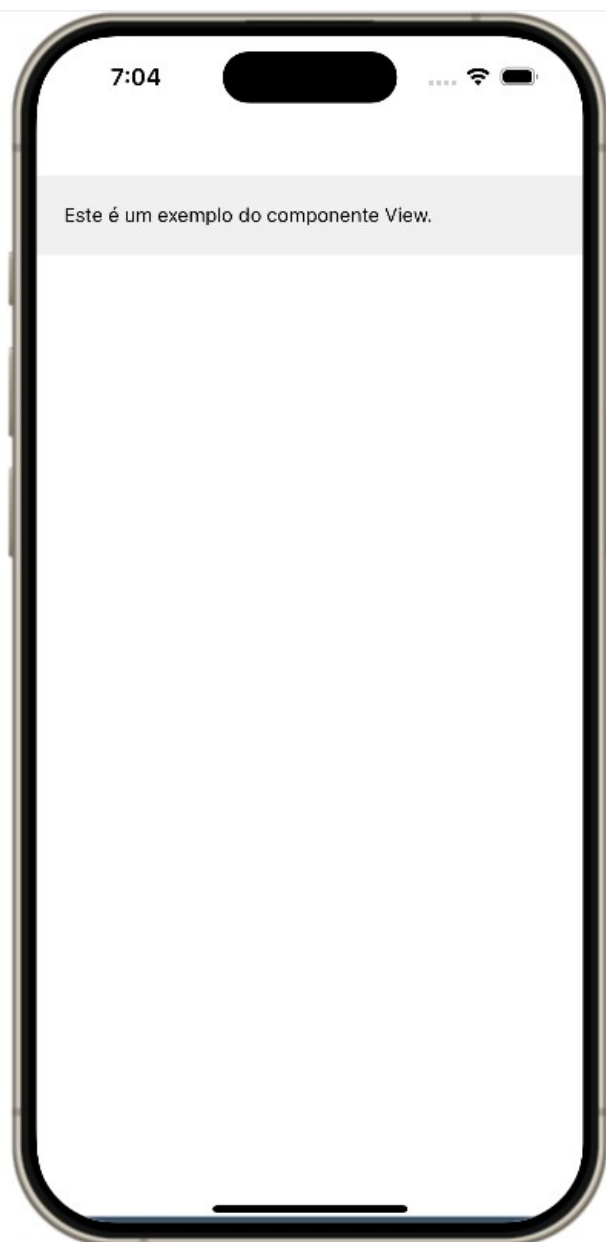
O código apresentado é um exemplo de um componente funcional em React Native chamado ExemploView. Ele importa os módulos necessários do React e do React Native para construir a interface, utilizando View e Text, que são elementos básicos desse framework para estruturar layouts e exibir texto, respectivamente.

No componente ExemploView, temos uma função que retorna um elemento View. A View é um contêiner utilizado para organizar e agrupar outros componentes, semelhante a uma <div> no React para web. No exemplo, a View possui algumas propriedades de estilo aplicadas através do objeto style, que define características visuais como padding (espaçamento interno), marginTop (margem superior) e backgroundColor (cor de fundo).

Dentro desse contêiner View, há um componente Text que exibe a mensagem "Este é um exemplo do componente View.". O Text é o componente utilizado para mostrar textos no React Native, análogo à tag <p> no HTML. Assim, toda a estrutura é encapsulada na View com estilos aplicados, e o texto é exibido dentro dela.

Por fim, o componente ExemploView é exportado usando export default ExemploView, o que permite que ele seja importado e utilizado em outras partes do aplicativo. Em termos práticos, esse exemplo mostra como criar e estilizar componentes básicos no React Native e é útil para ilustrar a estrutura de um layout com textos dentro de um contêiner visual.

Obs: Não deixe de salvar o seu projeto clicando no botão *Save*.



4. Componente Text

Este exemplo apresenta o componente Text que é similar ao `<p>` do HTML. Para isso, crie um novo *Snack* e edite o arquivo App.js como apresentado a seguir.

```
import React from 'react';
import { Text } from 'react-native';

const ExemploText = () => (
  <Text style={{ marginTop: 50, fontSize: 20, color: 'blue' }}>
    Este é um exemplo do componente Text.
  </Text>
);

export default ExemploText;
```

O código apresentado é um exemplo de um componente funcional em React Native chamado ExemploText. Ele importa o módulo React e o componente Text do React Native, que é usado para exibir textos na interface do aplicativo.

Dentro do componente ExemploText, uma função retorna um elemento Text, que é responsável por mostrar uma mensagem na tela. O texto que será exibido é "Este é um exemplo do componente Text.". Para estilizar esse texto, o componente Text utiliza a

propriedade `style`, que aplica um objeto de estilo diretamente ao elemento.

As propriedades de estilo especificadas incluem `marginTop`, que adiciona um espaçamento superior de 50 pixels, `fontSize`, que define o tamanho da fonte como 20 pixels, e `color`, que altera a cor do texto para azul. Essas propriedades permitem personalizar a aparência do texto exibido, tornando-o mais destacado e visualmente agradável.

Por fim, o componente `ExemploText` é exportado usando `export default ExemploText`, o que permite que ele seja importado e utilizado em outras partes do aplicativo. Em resumo, esse exemplo demonstra como criar e estilizar um componente de texto simples no React Native, ilustrando a forma de apresentação de informações ao usuário de maneira clara e personalizada.



5. Componente Image

Esta aplicação usa o componente Image para carregar na tela uma imagem disponível na internet. Para isso, crie um novo *Snack* e edite o arquivo App.js com o código-fonte apresentado a seguir.

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

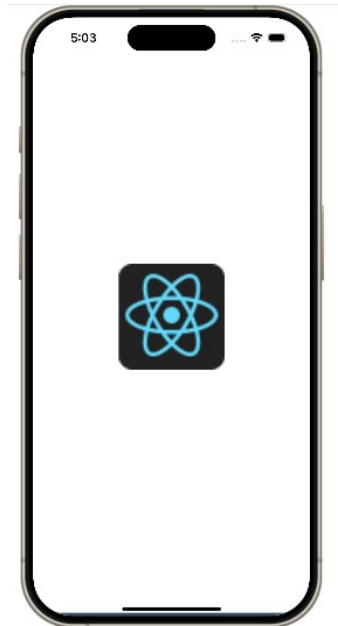
const ExemploImage = () => (
  <View style={styles.container}>
    <Image
      source={{ uri: 'https://reactnative.dev/img/tiny_logo.png' }}
      style={styles.image}
    />
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1, // Faz o View ocupar toda a tela
    justifyContent: 'center', // Centraliza verticalmente
    alignItems: 'center', // Centraliza horizontalmente
  },
  image: {
    width: 150,
    height: 150,
  },
});
```

O código apresentado é um componente funcional em React Native chamado ExemploImage, que tem como objetivo exibir uma imagem na tela. Ele começa com a importação das bibliotecas necessárias, como React e componentes do react-native, incluindo

View, Image e StyleSheet. Dentro do componente, uma View é criada para servir como contêiner da imagem. A imagem é carregada a partir de uma URL específica, que, neste caso, é https://reactnative.dev/img/tiny_logo.png, utilizando a propriedade source do componente Image.

Os estilos são definidos através do StyleSheet.create, onde duas classes são especificadas. A primeira, chamada container, faz com que a View ocupe toda a tela e centralize seu conteúdo tanto vertical quanto horizontalmente. A segunda classe, chamada image, define as dimensões da imagem, que são de 150 pixels de largura e 150 pixels de altura. Quando o componente é renderizado, ele exibe a imagem centralizada na tela com as dimensões especificadas, resultando em uma interface simples e visualmente agradável.



6. Componente TextInput

Este exemplo usa um componente TextInput como elemento de entrada e reflete em outro componente Text tudo que nele é digitado.

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

const ExemploTextInput = () => {
  const [value, setValue] = useState('');

  return (
```

```

<View style={{ padding: 20, justifyContent: 'center', flex: 1 }}>
  <TextInput
    style={{
      height: 40,
      borderColor: 'gray',
      borderWidth: 1,
      paddingHorizontal: 8,
      marginBottom: 20,
      borderRadius: 5,
    }}
    placeholder="Digite algo aqui"
    onChangeText={text => setValue(text)}
    value={value}
  />
  <Text>Valor digitado: {value}</Text>
</View>
);
};

export default ExemploTextInput;

```

Esse código é um exemplo de um componente React Native chamado `ExemploTextInput`, que utiliza o hook `useState` para gerenciar o estado do texto inserido pelo usuário. O componente principal é uma `View` que contém um campo de entrada de texto (`TextInput`) e um elemento `Text` que exibe o valor digitado.

No `TextInput`, o usuário pode digitar qualquer texto, e o valor digitado é armazenado na variável de estado `value`, inicializada como uma string vazia. O método `onChangeText` captura qualquer alteração no campo de texto e atualiza o estado com o novo valor, utilizando a função `setValue`.

O estilo aplicado à View posiciona os elementos com espaçamento interno (padding: 20), centraliza verticalmente o conteúdo (justifyContent: 'center') e faz com que a View ocupe toda a tela (flex: 1). O TextInput é estilizado para ter altura de 40 pixels, borda cinza com largura de 1 pixel, espaçamento interno horizontal, margem inferior, e cantos arredondados. O texto digitado no campo é exibido dinamicamente logo abaixo por meio do componente Text, que exibe a mensagem "Valor digitado:" seguida pelo conteúdo da variável value.

Ao interagir com o campo de entrada, o usuário vê o valor digitado atualizado em tempo real no componente Text.



7. Componente Alert

Este exemplo usa o componente Alert para apresentar uma mensagem ao usuário assim que o botão for pressionado.

```
import React from 'react';
import { Button, Alert, View, Platform } from 'react-native';

const ExemploButton = () => (
  <View style={{flex:1,justifyContent:'center',alignItems:'center'}}>
    <Button
      title="Clique Aqui"
    />
  </View>
);
```

```

    onPress={() =>
      Platform.OS === 'web'
        ? alert('Botão Clicado!')
        : Alert.alert('Botão Clicado!')
    }
  />
</View>
);

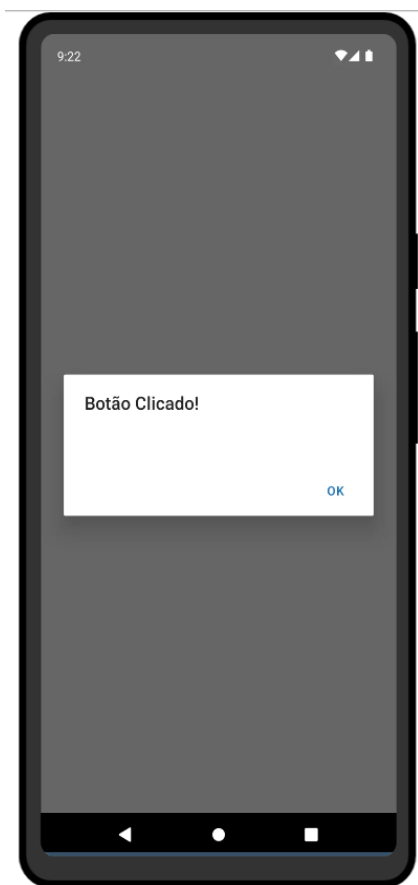
export default ExemploButton;

```

O código apresentado define um componente funcional em React Native chamado `ExemploButton`, que exibe um botão na tela e reage ao seu clique. O componente utiliza a biblioteca `react-native` para criar uma interface interativa. Ao ser renderizado, o componente cria uma `View` que ocupa toda a tela, centralizando seu conteúdo tanto vertical quanto horizontalmente. Dentro dessa `View`, um botão é exibido com o título "Clique Aqui".

Quando o usuário pressiona o botão, a função definida na propriedade `onPress` é acionada. Esta função verifica a plataforma em que o aplicativo está sendo executado utilizando `Platform.OS`. Se o aplicativo estiver rodando na web, um alerta simples é exibido através da função `alert`, mostrando a mensagem "Botão Clicado!". Caso contrário, se o aplicativo estiver em um dispositivo móvel (iOS ou Android), um alerta mais elaborado é apresentado utilizando `Alert.alert`, também com a mensagem "Botão Clicado!".

Essa abordagem permite que o comportamento do botão seja adaptável, oferecendo uma experiência de usuário consistente e apropriada para diferentes plataformas. O resultado é uma interação clara e direta, onde o usuário recebe feedback imediato ao clicar no botão.



8. Componente TouchableHighlight

Através do componente TouchableHighlight podemos deixar o botão com outra cor após clicar nele, indicando que ele foi pressionado.

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

const ExemploTouchableHighlight = () => (
  <View style={styles.container}>
    <TouchableHighlight
      style={styles.button}
      onPress={() => alert('Área destacada!')}
      underlayColor="gray"
    >
      <Text style={styles.buttonText}>Pressione Aqui</Text>
    </TouchableHighlight>
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  button: {
    paddingVertical: 15,
    paddingHorizontal: 30,
    backgroundColor: 'lightblue',
    borderRadius: 5,
    marginVertical: 10,
    minWidth: 200,
    alignItems: 'center',
  },
  buttonText: {
    fontSize: 18,
    color: 'black',
  },
});
```

```
    },
  });

export default ExemploTouchableHighlight;
```

O código apresentado define um componente funcional em React Native chamado `ExemploTouchableHighlight`, que utiliza o componente `TouchableHighlight` para criar uma área clicável com feedback visual. Este componente é ideal para interações que requerem um toque, proporcionando uma experiência de usuário mais intuitiva.

Ao ser renderizado, o componente cria uma `View` que ocupa toda a tela e centraliza seu conteúdo tanto vertical quanto horizontalmente. Dentro dessa `View`, um `TouchableHighlight` é utilizado para envolver um texto que serve como botão. O texto exibido é "Pressione Aqui".

Quando o usuário pressiona a área destacada, a função definida na propriedade `onPress` é acionada, exibindo um alerta com a mensagem "Área destacada!". O `TouchableHighlight` também possui a propriedade `underlayColor`, que altera a cor de fundo para cinza quando o botão é pressionado, proporcionando um feedback visual ao usuário sobre a interação.

Os estilos são definidos utilizando o `StyleSheet.create`. A classe `container` garante que a `View` ocupe toda a tela e centralize seu conteúdo. A classe `button` define o estilo do botão, incluindo preenchimento vertical e horizontal, cor de fundo azul clara, bordas arredondadas e uma largura mínima de 200 pixels. A classe `buttonText` estiliza o texto do botão, definindo um tamanho de fonte de 18 pixels e cor preta.

Sendo assim, o componente oferece uma maneira clara e visualmente agradável para os usuários interagirem com a aplicação, com feedback imediato ao pressionar o botão.



9. Componente ScrollView

O componente `ScrollView` no `React Native` é usado para criar uma área de rolagem que pode conter vários elementos dentro de uma `View`. Ele é útil quando o conteúdo dentro da tela excede o espaço visível e precisa ser rolado, como em listas, textos longos ou layouts dinâmicos.

O `ScrollView` permite que o usuário role o conteúdo verticalmente (ou horizontalmente, se configurado), tornando-o ideal para exibir uma quantidade variável de elementos. Ele pode conter diversos componentes ou elementos de layout, como `Text`, `Image`, `View`, entre outros.

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

const ExemploScrollView = () => (
  <ScrollView contentContainerStyle={styles.container}>
    <View style={styles.imageContainer}>

      <Image style={styles.image} source={{ uri: 'https://encrypted-
        tbn0.gstatic.com/images?q=tbn:ANd9GcTu849OpWAZl3plqFZFaPLv517KpSQpZIFY_g&s' }} />

      <Image style={styles.image} source={{ uri: 'https://encrypted-
        tbn0.gstatic.com/images?q=tbn:ANd9GcQiQaJfqMN3mt-
        jCfagygwIaE6Xu6443_bteQ&s' }} />

      <Image style={styles.image} source={{ uri:
        'https://upload.wikimedia.org/wikipedia/commons/e/e8/Museu_da_Inconfid
        %C3%Aancia_7.jpg' }} />
    </View>
  </ScrollView>
);
```

```

        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSOksUcd610laXWpI3571wVYe1R-
KxmqtFR6w&s' }} />

        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?
q=tbn:ANd9GcTLvQ6TFnEHLfxkSsQ9EWATBT8HohdcvJOGmA&s' }} />

        <Image style={styles.image} source={{ uri:
'https://cdn.folhape.com.br/img/pc/1100/1/dn_arquivo/2023/04/museu-do-recife-
forte-das-cinco-pontas-foto-arthur-de-souza-31.jpg' }} />

        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?
q=tbn:ANd9GcTy49LzNd5vcZU1ZL8R23UJXIZjF8X6U55xOQ&s' }} />

    </View>
  </ScrollView>
);

const styles = StyleSheet.create({
  container: {
    justifyContent: 'center',
    alignItems: 'center',
    paddingVertical: 20,
    marginTop: 50,
  },
  imageContainer: {
    alignItems: 'center',
    justifyContent: 'center',
    flexDirection: 'column',
  },
  image: {
    width: 300,
    height: 150,
    marginBottom: 15,
    borderRadius: 15,
    borderWidth: 2,
    borderColor: 'gray',
  },
});

export default ExemploScrollView;

```

Esse código cria um componente React Native chamado `ExemploScrollView`, que utiliza o `ScrollView` para exibir várias imagens com a possibilidade de rolagem vertical. O `ScrollView` permite que o conteúdo excedente seja visualizado ao deslizar a tela, já que as imagens são dispostas verticalmente e ocupam mais espaço do que a tela exibe de uma vez. Dentro do `ScrollView`, há uma `View` contendo diversas imagens, todas estilizadas com largura de 300 pixels, altura de 150 pixels, bordas arredondadas de 15 pixels e uma borda cinza de 2 pixels. As imagens são organizadas de forma vertical, com um espaçamento de 15 pixels entre elas, e podem ser roladas para visualizar o conteúdo completo. Esse design é ideal para quando há uma série de imagens ou itens que precisam ser exibidos de forma fluida em uma única coluna rolável.



10. Componente ScrollView II

No React Native, o componente `ScrollView` é utilizado para permitir a rolagem de conteúdo que excede o espaço disponível na tela, tanto vertical quanto horizontalmente. Ele é adequado para exibir blocos de conteúdo como imagens, textos ou formulários que não cabem de uma vez na tela. No entanto, é importante ter em mente que o `ScrollView` renderiza todo o conteúdo de uma só vez, o que pode causar problemas de desempenho se houver muitos elementos. Para listas grandes, o recomendado é usar `FlatList` ou `SectionList`, que só carregam os itens visíveis. O `ScrollView` permite rolagem vertical por padrão, mas, ao definir a propriedade `horizontal` como `true`, a rolagem horizontal também é ativada. Além disso, ele oferece outras propriedades úteis, como `refreshControl`, que adiciona o recurso de "puxar para atualizar", e `onScroll`, que permite capturar eventos de rolagem. Embora seja prático para conteúdos menores, é importante ter cautela ao utilizá-lo em cenários que envolvam grandes quantidades de dados. Para demonstrar o seu uso, crie um novo *Snack* e edite o arquivo `App.js` com o código-fonte apresentado a seguir. Observe que este exemplo é uma continuação do anterior, adicionando caixas de texto para indicar as imagens apresentadas.

```

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

const ExemploScrollView = () => (
  <ScrollView contentContainerStyle={styles.container}>
    <View style={styles.imageContainer}>
      <View style={styles.imageWrapper}>
        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?
q=tbn:ANd9GcTu849OpWAZl3plqFZFaPLv517KpSQpZIFY_g&s' }} />
        <Text style={styles.caption}>Museu Imperial - RJ</Text>
      </View>

      <View style={styles.imageWrapper}>
        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQiQaJfqMN3mt-
jCfagygwIaE6Xu6443_bteQ&s' }} />
        <Text style={styles.caption}>Museu Catavento - SP</Text>
      </View>

      <View style={styles.imageWrapper}>
        <Image style={styles.image} source={{ uri:
'https://upload.wikimedia.org/wikipedia/commons/e/e8/Museu_da_Inconfid
%C3%Aancia_7.jpg' }} />
        <Text style={styles.caption}>Museu da Inconfidência - MG</Text>
      </View>

      <View style={styles.imageWrapper}>
        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSOKsUcd610laXWpI3571wVYe1R-
KxmqtFR6w&s' }} />
        <Text style={styles.caption}>Museu da Gente Sergipana - SE</Text>
      </View>

      <View style={styles.imageWrapper}>
        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?
q=tbn:ANd9GcTLvQ6TFnEHLfxkSsQ9EWATBT8HohdcvJOGmA&s' }} />
        <Text style={styles.caption}>Museu Ricardo Brennand - PE</Text>
      </View>

      <View style={styles.imageWrapper}>
        <Image style={styles.image} source={{ uri:
'https://cdn.folhape.com.br/img/pc/1100/1/dn_arquivo/2023/04/museu-do-recife-
forte-das-cinco-pontas-foto-arthur-de-souza-31.jpg' }} />

```

```

        <Text style={styles.caption}>Forte São Tiago das Cinco Pontas -
PE</Text>
    </View>

    <View style={styles.imageWrapper}>
        <Image style={styles.image} source={{ uri: 'https://encrypted-
tbn0.gstatic.com/images?
q=tbn:ANd9GcTY49LzNd5vcZU1ZL8R23UJXIZjF8X6U55xOQ&s' }} />
        <Text style={styles.caption}>Museu Oscar Niemeyer - PR</Text>
    </View>
</View>
</ScrollView>
);

const styles = StyleSheet.create({
  container: {
    justifyContent: 'center',
    alignItems: 'center',
    paddingVertical: 20,
    marginTop: 50,
  },
  imageContainer: {
    alignItems: 'center',
    justifyContent: 'center',
    flexDirection: 'column',
  },
  imageWrapper: {
    alignItems: 'center',
    marginBottom: 20,
  },
  image: {
    width: 300,
    height: 150,
    borderRadius: 15,
    borderWidth: 2,
    borderColor: 'gray',
  },
  caption: {
    marginTop: 5,
    fontSize: 16,
    fontWeight: 'bold',
    color: '#333',
  },
});

```

```
export default ExemploScrollView;
```

O código apresentado implementa um componente funcional em React Native chamado `ExemploScrollView`, que exibe uma galeria de imagens de museus brasileiros, acompanhadas de legendas. A estrutura principal é composta por um `ScrollView`, que permite que o usuário role verticalmente pelas imagens.

Dentro do `ScrollView`, há um `View` que serve como contêiner para as imagens e suas respectivas legendas. Cada imagem é envolvida por outro `View`, conhecido como `imageWrapper`, que organiza a imagem e o texto em uma coluna, garantindo que a legenda fique logo abaixo da imagem correspondente. As imagens são carregadas a partir de URLs, e cada uma é acompanhada por um texto descritivo que indica o nome do museu e sua localização.

Os estilos são definidos utilizando `StyleSheet`, proporcionando uma apresentação visual atraente. O contêiner principal é centralizado e possui espaçamento vertical, enquanto cada imagem tem dimensões fixas, bordas arredondadas e uma borda cinza para destacá-las. As legendas são estilizadas com um tamanho de fonte maior e em negrito, garantindo boa legibilidade.

Assim, o componente cria uma experiência interativa e visualmente agradável, permitindo que os usuários explorem diferentes museus de forma simples e eficiente.

