Etec
Bento Quirino
Campinas

Atividade Prática para Criar um Aplicativo de Cálculo de IMC em React Native com Expo

Este guia mostra como criar um aplicativo para calcular o Índice de Massa Corporal (IMC) utilizando React Native e Expo. Vamos dividir o aplicativo em componentes para tornar o código mais modular e organizado.

Passo 1: Configurando o Ambiente de Desenvolvimento

1. **Instale o Node.js**: Certifique-se de ter o Node.js instalado. Download aqui.
2. **Instale o Expo CLI**: No terminal, execute:

```
npm install -g expo-cli
```

3. **Crie um Novo Projeto Expo**:

```
npx create-expo-app CalculadoraIMC --template blank
```

   Escolha a opção `blank` para iniciar um projeto básico.

4. **Navegue para o Diretório do Projeto**:

```
cd CalculadoraIMC
```

5. **Inicie o Projeto**:

```
npx expo start --tunnel
```

Passo 2: Estrutura do Projeto

Vamos organizar o projeto criando uma pasta `src` para armazenar nossos componentes.

1. **Crie a pasta `src` e `components`**:
   - No terminal:

```
mkdir -p src/components
```

Passo 3: Criando os Componentes

Vamos criar três componentes: `Title`, `FormIMC` e `Result`, todos dentro da pasta `components`.

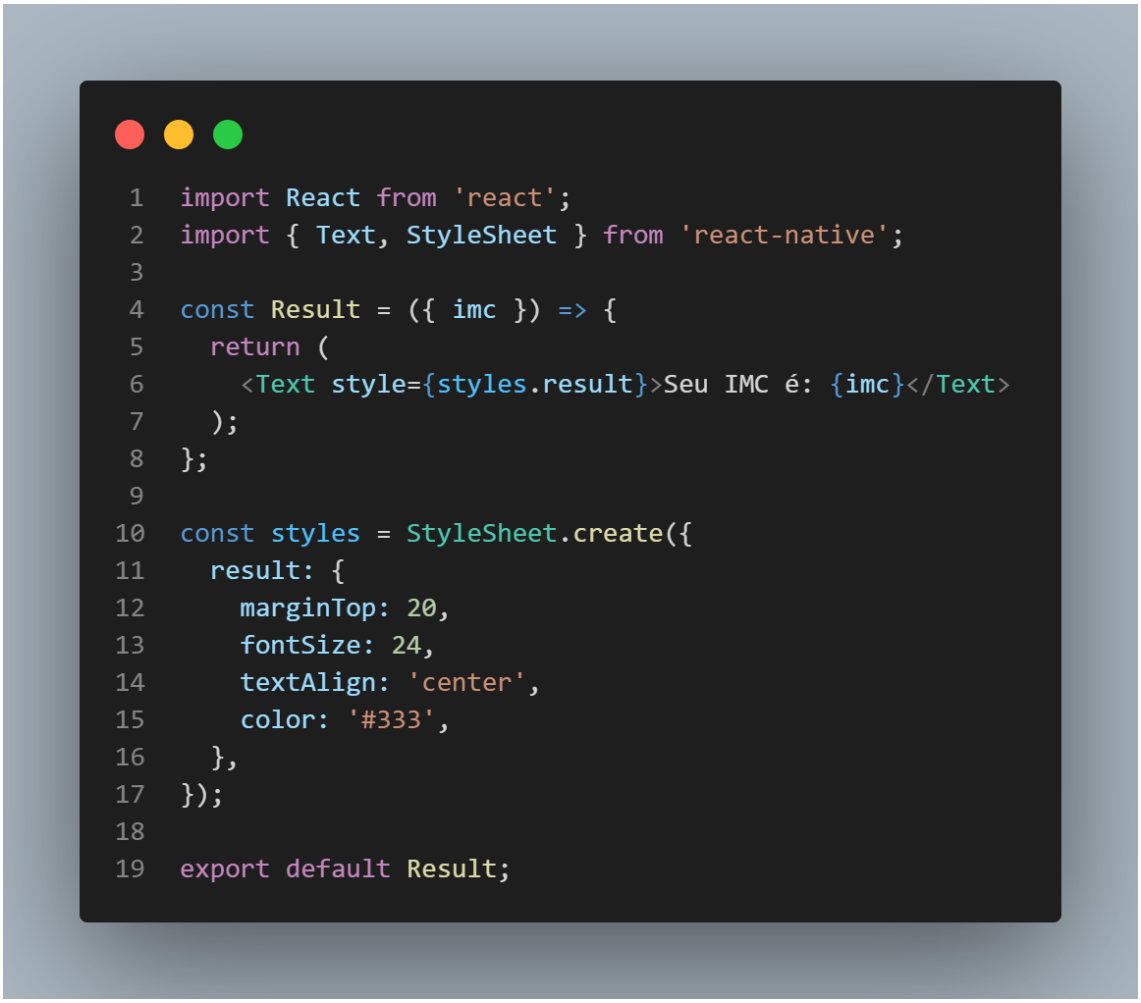
1. **Componente `Title`**:

- Crie um arquivo chamado `Title.js` dentro da pasta `src/components`:

```
1  import { Text, StyleSheet } from 'react-native';
2
3  const Title = () => {
4    return (
5      <Text style={styles.title}>Calculadora de IMC</Text>
6    );
7  };
8
9  const styles = StyleSheet.create({
10   title: {
11     fontSize: 32,
12     fontWeight: 'bold',
13     textAlign: 'center',
14     marginBottom: 24,
15   },
16 });
17
18 export default Title;
```

2. **Componente `Result`:**

- Crie um arquivo chamado `Result.js` dentro da pasta `src/components`:



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

const Result = ({ imc }) => {
  return (
    <Text style={styles.result}>Seu IMC é: {imc}</Text>
  );
};

const styles = StyleSheet.create({
  result: {
    marginTop: 20,
    fontSize: 24,
    textAlign: 'center',
    color: '#333',
  },
});

export default Result;
```

3. **Componente `FormIMC`:**

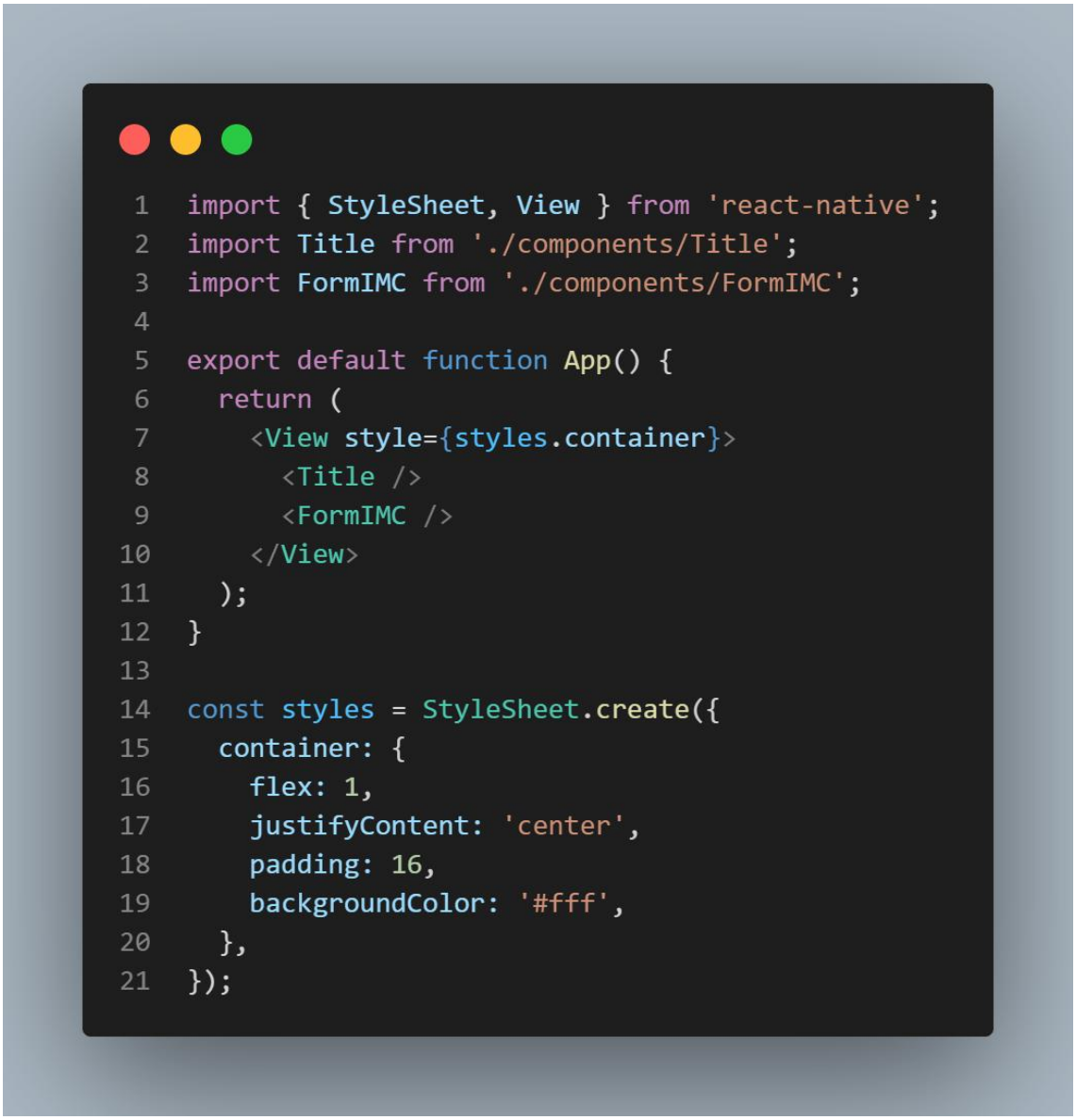
   - Crie um arquivo chamado `FormIMC.js` dentro da pasta `src/components`:

```jsx
import { View, TextInput, Button, StyleSheet } from 'react-native';
import Result from './Result';

const FormIMC = () => {
  const [peso, setPeso] = useState('');
  const [altura, setAltura] = useState('');
  const [imc, setImc] = useState(null);

  const calcularIMC = () => {
    if (peso && altura) {
      const alturaMetros = parseFloat(altura) / 100;
      const imcCalculado = (parseFloat(peso) / (alturaMetros * alturaMetros)).toFixed(2);
      setImc(imcCalculado);
    }
  };

  return (
    <View style={styles.formContainer}>
      <TextInput
        style={styles.input}
        placeholder="Peso (kg)"
        keyboardType="numeric"
        value={peso}
        onChangeText={setPeso}
      />
      <TextInput
        style={styles.input}
        placeholder="Altura (cm)"
        keyboardType="numeric"
        value={altura}
        onChangeText={setAltura}
      />
      <Button title="Calcular IMC" onPress={calcularIMC} />
      {imc && <Result imc={imc} />}
    </View>
  );
};

const styles = StyleSheet.create({
  formContainer: {
    backgroundColor: '#f0f0f0',
    padding: 16,
    borderRadius: 10,
  },
  input: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,
    marginBottom: 12,
    paddingHorizontal: 8,
    borderRadius: 5,
  },
});

export default FormIMC;
```

Passo 4: Atualizando o `App.js`

Agora vamos atualizar o arquivo `App.js` para utilizar os componentes criados.

1. **Substitua o conteúdo do arquivo `App.js`**:



```jsx
import { StyleSheet, View } from 'react-native';
import Title from './components/Title';
import FormIMC from './components/FormIMC';

export default function App() {
  return (
    <View style={styles.container}>
      <Title />
      <FormIMC />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    padding: 16,
    backgroundColor: '#fff',
  },
});
```

Passo 5: Explicação do Código

1. **Componentes**:
   - `Title`: Exibe o título da aplicação.
   - `Result`: Exibe o resultado do IMC calculado.
   - `FormIMC`: Contém o formulário para entrada de dados (peso e altura) e a lógica para calcular o IMC.
2. **Modularização**:
   - Dividimos o código em componentes para facilitar a manutenção e escalabilidade.
3. **Estilos**:
   - Usamos `StyleSheet` para estilizar cada componente de forma consistente.

Passo 6: Executando o Aplicativo

Com todas as alterações feitas, inicie o aplicativo com o Expo Go em um dispositivo físico ou emulador.

Conclusão

Agora você tem um aplicativo de cálculo de IMC em React Native bem estruturado e modular, fácil de manter e expandir.

Desafio: Complementando a Calculadora de IMC

Objetivo:

O objetivo deste desafio é aplicar os conhecimentos adquiridos sobre React Native e a utilização de Hooks para estender a funcionalidade da calculadora de IMC. O aluno deverá implementar novas funcionalidades que irão exibir a classificação do IMC com base na tabela de IMC, além de calcular e exibir o peso mínimo ideal e o peso máximo ideal para uma altura específica.

Instruções:

1. **Exibir a Classificação do IMC:**
   - Após calcular o IMC, o aluno deverá implementar uma lógica para exibir a classificação do IMC de acordo com a tabela abaixo:
     - **Menor 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
     - **Maior ou igual a 40:** Obesidade grau 3 (obesidade mórbida)
   - A classificação deverá ser exibida em um componente próprio abaixo do resultado do IMC.
2. **Calcular e Exibir o Peso Ideal:**
   - O aluno deverá implementar um cálculo que determine o peso mínimo e máximo ideal para a altura informada, com base no IMC ideal (entre 18.5 e 24.9).
   - Fórmula para o cálculo:
     - **Peso mínimo ideal** = $18.5 \times (\text{altura em metros})^2$
     - **Peso máximo ideal** = $24.9 \times (\text{altura em metros})^2$
   - Os valores de peso mínimo e máximo ideais deverão ser exibidos em um componente próprio abaixo da classificação do IMC.
3. **Organização dos Componentes:**
   - Todos os novos componentes que forem criados para este desafio devem ser organizados na pasta `components`, dentro da pasta `src`, conforme a estrutura já utilizada no projeto original.
   - Comente todo o código explicando a lógica utilizada.

Sugestão de Implementação:

1. **Adicionar a Classificação do IMC:**
   - Crie um componente `Classification` para exibir a classificação.

- No arquivo onde a calculadora de IMC foi implementada, adicione uma função que receba o valor do IMC e retorne a classificação correspondente.
- Exiba a classificação abaixo do valor do IMC usando o componente `Classification`.

2. **Adicionar o Cálculo do Peso Ideal:**
   - Crie um componente `IdealWeight` para calcular e exibir o peso mínimo e máximo ideais.
   - Adapte a função de cálculo para considerar a altura e calcular o peso ideal.
   - Exiba o peso ideal abaixo da classificação do IMC.
3. **Testes e Validações:**
   - Teste o aplicativo com diferentes entradas para garantir que as novas funcionalidades estejam funcionando corretamente.
   - Valide as entradas do usuário para garantir que apenas valores numéricos válidos sejam aceitos.

Dicas:

- Utilize `useState` para gerenciar o estado das novas informações (classificação do IMC e peso ideal).
- Utilize `useEffect` se precisar reagir a mudanças específicas, como a atualização da altura ou peso.
- Consulte a documentação do React Native para ajudar na implementação e estilização dos componentes.

Critérios de Avaliação

Para avaliar a implementação do desafio de complementação da calculadora de IMC, serão utilizados os seguintes critérios:

1. Funcionalidade (40 pontos)

- **Classificação do IMC (20 pontos):**
  - A classificação do IMC é calculada corretamente e exibida de forma precisa.
  - O componente `Classification` é implementado corretamente e integrado à interface.
- **Cálculo do Peso Ideal (20 pontos):**
  - O peso mínimo e máximo ideais são calculados corretamente com base na altura informada.
  - O componente `IdealWeight` é implementado corretamente e exibe os valores de forma clara.

2. Qualidade do Código (30 pontos)

- **Organização e Estrutura do Código (15 pontos):**
  - O código está organizado e segue a estrutura de pastas recomendada (`components` dentro de `src`).
  - Componentes estão devidamente modularizados e reutilizados quando necessário.

- **Comentários e Nomenclatura (15 pontos):**
  - O código está bem comentado, explicando a lógica de cada função e componente.
  - Variáveis, funções e componentes têm nomes claros e descritivos, facilitando a leitura e manutenção do código.

3. Interface e Estética (20 pontos)

- **Clareza e Legibilidade da Interface (10 pontos):**
  - A interface está organizada de forma que as informações sejam facilmente compreendidas pelo usuário.
  - Elementos de texto e botões são apresentados de forma clara e estão bem alinhados.
- **Estilização e Design (10 pontos):**
  - O design segue uma paleta de cores harmoniosa e respeita as diretrizes de design.
  - Os cantos arredondados e outras estilizações solicitadas são aplicados corretamente.

4. Testes e Validação (10 pontos)

- **Validação de Entradas (5 pontos):**
  - O aplicativo valida corretamente as entradas do usuário, garantindo que apenas valores numéricos válidos sejam aceitos.
- **Funcionamento Geral (5 pontos):**
  - O aplicativo foi testado com diferentes entradas e se comporta conforme o esperado, sem bugs ou falhas.

Total: 100 pontos

Considerações Finais:

- **Bônus (até 5 pontos extras):**
  - Implementação de funcionalidades adicionais ou melhorias que vão além do que foi solicitado, como uma mensagem de erro personalizada para entradas inválidas ou um layout adaptativo para diferentes tamanhos de tela.

Este desafio visa consolidar os conceitos de manipulação de estado e criação de componentes dinâmicos em React Native, além de incentivar a aplicação prática de fórmulas matemáticas na programação de aplicativos móveis.