


Programação para Dispositivos Móveis

Aula 3



Prof: Uemerson Pinheiro Junior

Trabalhando com imagens

- Iremos trabalhar com Image
- Iremos trabalhar com valores aleatórios

Passo 1

- Entre no site: <https://snack.expo.dev/>
- Mude a versão do expo para a 50.0.0
- Mude o nome do projeto para heads-or-tails (Cara ou Coroa)
- Salve o projeto na sua conta



Passo 2

- Delete o arquivo snack-icon.png dentro da pasta assets:



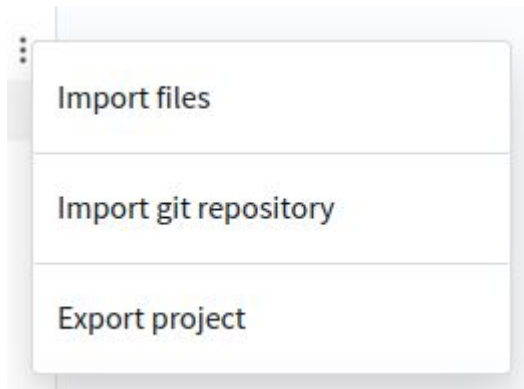
assets



snack-icon.png

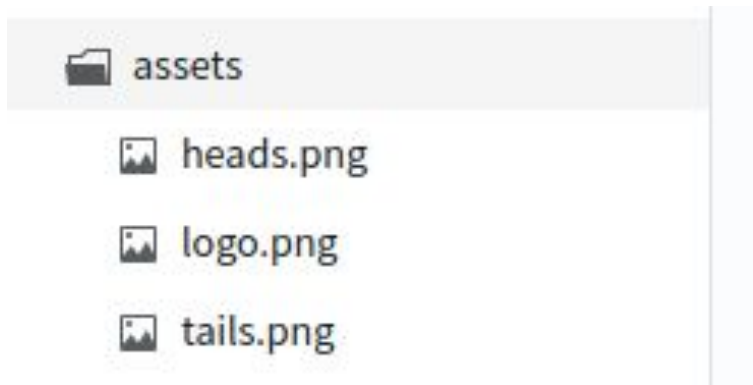
Passo 3

- Baixe os arquivos da pasta assets no repositório:
<https://github.com/Uemerson/minhas-aulas/tree/main/curso-tecnico/programacao-para-dispositivos-moveis/aula-3/assets>
- Importe os arquivos que baixou para o projeto, clique nos três pontos e depois em import files:



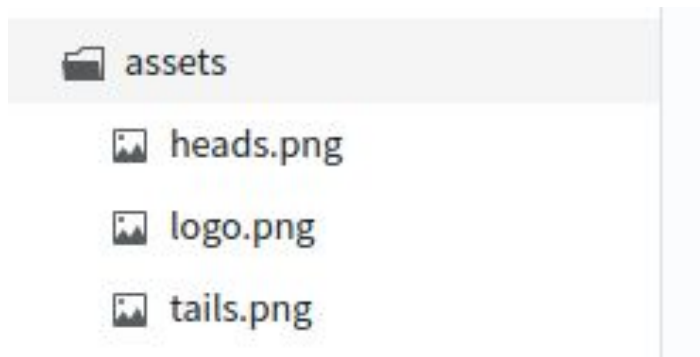
Passo 4

- Mova os arquivos para a pasta assets:



Passo 5

- Mova os arquivos para a pasta assets:



- Delete a pasta components:



Passo 6

- Substitua todo o conteúdo do arquivo App.js por:

```
import { SafeAreaView, StyleSheet } from 'react-native';

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

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 8,
    backgroundColor: '#61bd8c'
  },
  coin: {
    marginTop: 10,
    width: 150,
    height: 150,
  },
});
```


Passo 7

- Na linha 1, importe os componentes que iremos utilizar:

```
import {  
  SafeAreaView,  
  StyleSheet,  
  Image,  
  TouchableOpacity,  
  Text,  
  View,  
} from 'react-native';
```

Passo 8

- Entre o componente SafeAreaView adicione a logo do app:

```
<SafeAreaView style={styles.container}>  
  <Image source={require('./assets/logo.png')} />  
</SafeAreaView>
```

Passo 9

- Depois da importação dos componentes, importe também o **useState**, que é um hook fundamental do React JS que permite a criação e manipulação de estado:

```
import { useState } from 'react';
```

- Vamos criar três estados, que iram guardar as informações de quantas vezes deu cara ou coroa e qual foi o lado da moeda sorteado. Adicione essas três linhas antes do return

```
const [coinSide, setCoinSide] = useState('Heads');  
const [headsCount, setHeadsCount] = useState(0);  
const [tailsCount, setTailsCount] = useState(0);
```

Passo 10

- Vamos adicionar nossas imagens, para aparecerem de acordo com o lado da moeda sorteado, adicione o código abaixo depois da Image logo:

```
<Image
  style={styles.coin}
  source={
    coinSide === 'Heads'
      ? require('./assets/heads.png')
      : require('./assets/tails.png')
  }
/>
```

Passo 11

- Depois das imagens das moedas, adicione o código a seguir:

```
<View style={styles.countContainer}>  
  <View style={styles.count}>  
    <Text style={styles.countText}>Cara: {headsCount}</Text>  
  </View>  
  <View style={styles.count}>  
    <Text style={styles.countText}>Coroa: {tailsCount}</Text>  
  </View>  
</View>
```

Passo 12

- Dentro dos styles, adicione o código a seguir:

```
countContainer: {  
  flexDirection: 'row',  
  marginBottom: 10,  
},  
count: {  
  marginRight: 20,  
},  
countText: {  
  fontSize: 18,  
  fontWeight: 'bold',  
  color: '#007BFF',  
},
```

Passo 13

- Adicione o botões a seguir, depois dos textos:

```
<TouchableOpacity
  style={styles.button}
  onPress={() => {
    const randomCoin = Math.floor(Math.random() * 2);
    setCoinSide(randomCoin == 0 ? 'Heads' : 'Tails');

    if (coinSide === 'Heads') setHeadsCount(headsCount + 1);
    else setTailsCount(tailsCount + 1);
  }}>
  <Text style={styles.buttonText}>Jogar moeda</Text>
</TouchableOpacity>
```

Passo 14

- Adicione nos estilos, os estilos dos botões:

```
button: {  
  backgroundColor: '#007BFF',  
  padding: 10,  
  margin: 10,  
  borderRadius: 5,  
},  
buttonText: {  
  color: 'white',  
  fontWeight: 'bold',  
},
```


Agora vamos entender o que fizemos!

Exercício

- 1. Adicione um botão para resetar a quantidade de vezes que foi sorteado cara ou coroa**