Desenvolvimento para Dispositivos Móveis

Aula 11 - Touchables e Navegação

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula, veremos como criar botões sem necessariamente utilizar o componente *button*. Além disso, abordaremos a navegação entre telas.

Objetivos

- Conhecer como criar objetos clicáveis
- Conhecer os diferentes tipos de invólucros para elementos clicáveis
- Como realizar navegação entre telas

TouchableHighlight e TouchableNativeFeedback

Link do video da aula: https://youtu.be/SBX00THAsN4

Veremos agora como criar links em componentes que não necessariamente são botões. Inicialmente, vamos copiar o código abaixo para utilizá-lo nesta aula.

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

export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.button}>
        <Text style={styles.text}>Clique aqui!</Text>
      </View>
      <Button title="Clique aqui!" />
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({

  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
```

```
  button: {
    backgroundColor: 'powderblue',
    color: 'white',
    margin: 20
  },
  text: {
    color: '#000',
    padding: 15
  }
})
```

No código acima, temos um componente clicável, o *button*, porém, também temos uma *View* que traz a aparência de um botão, mas não clicável.

TouchableHighlight component

O componente *TouchableHighlight* envolve algo que o desenvolvedor deseja que seja clicável e poderá executar ações quando clicado, através da propriedade *onPress*.

Ademais, o *TouchableHighlight* tem a propriedade *underlayColor*, que ajusta a cor subjacente ao botão, que por padrão é preto.

Veja abaixo um exemplo de uso do componente:

```
<TouchableHighlight underlayColor='white' onPress={() => {
alert('Clicou aqui!') }}>
        <View style={styles.button}>
          <Text style={styles.text}>Clique aqui!</Text>
        </View>
</TouchableHighlight>
```

TouchableNativeFeedback component

Além do TouchableHighlight, temos o *TouchableNativeFeedback*. Esse, porém, deve ser utilizado apenas para dispositivos android que têm o comportamento nativo que o componente busca trazer.

Esse comportamento nativo é uma animação que altera de forma suave a cor do botão ao receber o clique. A cor da animação é definida através da propriedade background.

Acompanhando a aula, implemente essas funcionalidades conforme o código abaixo:

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

export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <TouchableNativeFeedback
background={TouchableNativeFeedback.Ripple('red')}
underlayColor='white' onPress={() => { alert('Clicou aqui!') }}>
        <View style={styles.button}>
          <Text style={styles.text}>Clique aqui!</Text>
        </View>
      </TouchableNativeFeedback>
      <Button title="Clique aqui!" />
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({

  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  button: {
    backgroundColor: 'powderblue',
    color: 'white',
    margin: 20
  },
  text: {
    color: '#000',
    padding: 15
  }
})
```

TouchableOpacity e TouchableWithoutFeedback

Link do video da aula: https://youtu.be/hNV90Eq_d1Y

Nesta aula vamos abordar outras formas de criar áreas clicáveis sem utilizar necessariamente o componente *button*.

TouchableOpacity

Esse componente envolve o elemento que pode ser clicado e ao ser pressionado diminui a opacidade, dando um feedback ao usuário.

Veja um exemplo de uso abaixo:

```
<TouchableOpacity onPress={() => { alert('Clicou aqui!') }}>
        <View style={styles.button}>
          <Text style={styles.text}>Clique aqui!</Text>
        </View>
</TouchableOpacity>
```

TouchableWithoutFeedback

Quando estamos desenvolvendo um aplicativo que possui botões, na maioria das vezes, queremos que o usuário tenha um feedback ao clicar naquele objeto. Porém, eventualmente, por alguns motivos pontuais, o desenvolvedor pode não querer que o botão tenha esse feedback, daí surge a necessidade de uso do *TouchableWithoutFeedback*.

Pratique o uso desses componentes desenvolvendo o código abaixo:

```
import React from 'react';
import { SafeAreaView, Text, StyleSheet, View, Button,
TouchableHighlight, TouchableNativeFeedback, TouchableOpacity,
TouchableWithoutFeedback } from 'react-native';

export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <TouchableWithoutFeedback onPress={() => { alert('Clicou
aqui!') }}>
        <View style={styles.button}>
          <Text style={styles.text}>Clique aqui!</Text>
        </View>
      </TouchableWithoutFeedback>
      <Button title="Clique aqui!" />
    </SafeAreaView>
```

```
  );
}

const styles = StyleSheet.create({

  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  button: {
    backgroundColor: 'powderblue',
    color: 'white',
    margin: 20
  },
  text: {
    color: '#000',
    padding: 15
  }
})
```

Navegação com NavigationContainer (Parte 1)

Link do video da aula: https://youtu.be/FmvanvmHoLw

Vamos agora ver como criar navegação, utilizando *react native*, entre as telas no aplicativo. Para iniciar, copie o código abaixo, será ele o utilizado durante a aula.

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

export default function App() {
  return (
    <View style={styles.container}>
      <Button title="Clique aqui!" />
    </View>
  );
}
```

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

Instalando bibliotecas de suporte

Antes de iniciar o desenvolvimento do código para navegação entre as telas, precisaremos instalar algumas bibliotecas.

Para a instalação, execute em seu terminal o comando abaixo:

```
npm install @react-navigation/native @react-navigation/native-stack
```

Feita a instalação, precisamos instalar as dependências necessárias. Para isso, execute no terminal o seguinte comando:

```
expo install react-native-screens react-native-safe-area-context
```

Feito isso, podemos iniciar a prática do conteúdo.

NavigationContainer

O componente *NavigationContainer* será utilizado para envolver as partes da aplicação que desejamos que tenham navegação. Na prática, será criado um componente para cada tela e todos eles são envolvidos pelo componente em questão.

Acompanhando o passo a passo e os conceitos apresentados durante a aula, implemente o código abaixo no arquivo *App.js*

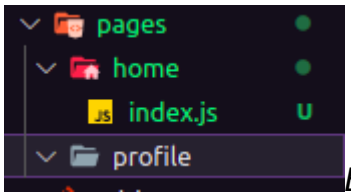
```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-
navigation/native-stack'
import React from 'react';
import { Home } from './pages/home';

const Stack = createNativeStackNavigator()

export default function App() {
```

```
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Bem vindo" component={Home} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Criando um componente

Na raiz do projeto, crie uma pasta chamada *pages*. Dentro de *pages* crie mais duas pastas que receberão os nomes dos componentes que serão criados: *home* e *profile*. Por fim, crie na pasta *home* um arquivo, nomeando-o de *index.js*.

Veja abaixo a estrutura das pastas:



Estrutura de pastas

No arquivo *index.js* implemente o código abaixo:

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

const Home = () => {
  return (
    <View>
      <Text>Bem vindo a tela inicial</Text>
    </View>
  )
}

export { Home }
```

Navegação com NavigationContainer (Parte 2)

Link do video da aula: https://youtu.be/be4M4fyTq-c

Na primeira parte criamos um componente *Home*, porém, ainda não existe navegação no aplicativo. Então, faremos agora esse trabalho.

Aplicando estilo no componente

O componente criado anteriormente não possuía estilo, o que dificulta a percepção de mudança de tela. Então, para diferenciar as telas, vamos aplicar uma leve estilização.

Em seu componente *Home* adicione a estilização, de modo que seu código fique como se vê abaixo:

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

const Home = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Bem vindo a tela inicial</Text>
    </View>
  )
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#ff8',
    alignItems: 'center',
    justifyContent: 'center',
  },
  text: {
    fontSize: 30
  }
})

export { Home }
```

Criando o componente *Profile*

Agora, iremos criar o componente *profile*, que será a segunda tela do aplicativo. Para fazer isso, tomaremos como base o componente *Home*, alterando apenas algumas informações.

Seu componente deve ficar como se vê abaixo:

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

const Profile = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Profile</Text>
    </View>
  )
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f8f',
    alignItems: 'center',
    justifyContent: 'center',
  },
  text: {
    fontSize: 30
  }
})

export { Profile }
```

De posse do novo componente, importe-o no *App.js e* adicione uma nova *Stack Screen* ao *NavigationContainer*.

```
<Stack.Screen name="Profile" component={Profile} />
```

Mudando de tela

Até o momento, foram implementados os componentes, porém, ainda não há navegação entre as telas.

Para navegar entre as telas, criaremos um botão no componente *Home*. Além disso, vamos utilizar uma propriedade que advém do *Stack Screen* denominada de *navigation*.

Acompanhando a aula, altere seu componente *Home* para ficar como abaixo:

```
const Home = ({ navigation }) => {
```

```
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Bem vindo a tela inicial</Text>
      <Button title="Profile" onPress={() => {
navigation.navigate('Profile', { name: 'Profile' }) }} />
    </View>
  )
}
```

Além disso, precisamos alterar também o componente *Profile* para ficar como abaixo:

```
const Profile = ({ navigation, route }) => {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Está é a página
{route.params.name}</Text>
    </View>
  )
}
```

Com essas modificações e implementações, já temos um aplicativo com navegação entre telas.

Resumo

Nesta aula, vimos como transformar componentes que não necessariamente são botões em objetos clicáveis, com ou sem feedback ao usuário. Ademais, vimos como estruturar um código para um aplicativo que possui mais de uma tela, além de configurar a navegação entre as telas.