Navegando no App: Rotas, Parâmetros e Estado Simples

Objetivos

Entender a importância da navegação em aplicativos mobile.

Aprender a configurar e utilizar o React Navigation (Stack Navigator).

Passar parâmetros entre telas.

Gerenciar estado básico entre componentes.

BÔNUS (Eng. Software): Introduzir os conceitos de SPA (Single Page Application) e roteamento.

Revisão Rápida

Qual a diferença entre props e state em React Native?

Como criamos e estilizamos componentes básicos?

Conteúdo Teórico

Por que precisamos de navegação?

Aplicativos multi-tela vs. Single Page Applications (SPA).

Experiência do usuário: como ele se move entre as funcionalidades.

Introdução ao React Navigation:

A biblioteca padrão para navegação em React Native.

Tipos de navegadores: Stack, Tab, Drawer (foco no Stack Navigator).

Instalação e configuração inicial.

Stack Navigator:

Como funciona: uma pilha de telas.

navigation.navigate(): para ir para uma nova tela.

navigation.goBack(): para retornar à tela anterior.

navigation.push(): para adicionar uma nova instância da mesma tela na pilha.

navigation.replace(): para substituir a tela atual na pilha.

Passagem de Parâmetros:

Como enviar dados de uma tela para outra (route.params).

Gerenciamento de Estado Básico:

Revisão de useState e useEffect para estado local.

Passando funções como props para atualizar o estado do componente pai.

Considerações de Engenharia de Software:

Organização do código de navegação.

Cuidado com o "prop drilling" (passar props por muitos níveis).

Prática Laboratorial (MÃO NA MASSA - com Snippets)

Slide: Passo 1: Instalar e Configurar React Navigation

Ação: Primeiro, instalamos as bibliotecas necessárias para a navegação.

Comando: Execute no terminal do seu projeto:

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

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

Explicação:

@react-navigation/native: O core da biblioteca de navegação.

@react-navigation/stack: O navegador de pilha que usaremos para ir e voltar entre telas.

react-native-screens e react-native-safe-area-context: Dependências que melhoram a performance e a área segura da tela.

Slide: Passo 2: Estrutura do Navegador Principal (App.js)

Ação: Vamos configurar o NavigationContainer e o StackNavigator no seu App.js.

Mudança Chave: Envolvemos todo o aplicativo com NavigationContainer e definimos as telas na Stack.Navigator.

// App.js

import 'react-native-gesture-handler'; // Importante para React Navigation

import React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

// Importe suas telas

import ListaPontosTuristicos from './screens/ListaPontosTuristicos';

import DetalhesPontoTuristico from './screens/DetalhesPontoTuristico';

const Stack = createStackNavigator();

export default function App() {

  return (

    <NavigationContainer>

      <Stack.Navigator initialRouteName="ListaPontos">

        <Stack.Screen

          name="ListaPontos"

          component={ListaPontosTuristicos}

          options={{ title: 'Pontos Turísticos' }}

        />

        <Stack.Screen

          name="DetalhesPonto"

          component={DetalhesPontoTuristico}

          options={{ title: 'Detalhes do Ponto' }}

        />

      </Stack.Navigator>

    </NavigationContainer>

  );

}

Entendimento:

NavigationContainer: Gerencia o estado da navegação.

createStackNavigator(): Cria um objeto que contém os componentes para o navegador de pilha.

Stack.Navigator: O componente que define a pilha de telas.

Stack.Screen: Define cada tela na pilha, com um name (para navegação) e o component a ser renderizado.

options: Permite configurar o cabeçalho da tela (ex: title).

Slide: Passo 3: Criar as Telas (ListaPontosTuristicos.js e DetalhesPontoTuristico.js)

Ação: Crie dois novos arquivos na pasta screens/.

screens/ListaPontosTuristicos.js (Inicial):

// screens/ListaPontosTuristicos.js

import React from 'react';

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

import { useNavigation } from '@react-navigation/native'; // <--- Importe useNavigation

const ListaPontosTuristicos = () => {

  const navigation = useNavigation(); // <--- Hook para acessar a navegação

  const handleGoToDetails = () => {

    // <--- Navega para a tela 'DetalhesPonto' e passa parâmetros

    navigation.navigate('DetalhesPonto', {

      pontoId: '123',

      nomePonto: 'Parque Barigui',

      descricaoPonto: 'Um lindo parque em Curitiba com capivaras.',

    });

  };

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Lista de Pontos Turísticos</Text>

      <Button title="Ver Detalhes do Parque Barigui" onPress={handleGoToDetails} />

      {/\* Aqui virá a lista real de pontos turísticos \*/}

    </View>

  );

};

const styles = StyleSheet.create({

  container: { flex: 1, justifyContent: 'center', alignItems: 'center' },

  title: { fontSize: 24, marginBottom: 20 },

});

export default ListaPontosTuristicos;

Explicação:

useNavigation(): Hook que nos dá acesso ao objeto navigation, que contém métodos para navegar.

navigation.navigate('DetalhesPonto', { ... }): O primeiro argumento é o name da tela (definido em App.js), o segundo é um objeto com os parâmetros a serem passados.

Slide: Passo 3 (Continuação): DetalhesPontoTuristico.js

Ação: Crie o componente para a tela de detalhes.

Mudança Chave: Usamos o hook useRoute para acessar os parâmetros passados.

// screens/DetalhesPontoTuristico.js

import React from 'react';

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

import { useRoute, useNavigation } from '@react-navigation/native'; // <--- Importe useRoute

const DetalhesPontoTuristico = () => {

  const route = useRoute(); // <--- Hook para acessar a rota atual

  const navigation = useNavigation(); // Para o botão de voltar

  // <--- Acessando os parâmetros passados

  const { pontoId, nomePonto, descricaoPonto } = route.params;

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Detalhes do Ponto</Text>

      <Text style={styles.detailText}>ID: {pontoId}</Text>

      <Text style={styles.detailText}>Nome: {nomePonto}</Text>

      <Text style={styles.detailText}>Descrição: {descricaoPonto}</Text>

      <Button title="Voltar" onPress={() => navigation.goBack()} />

    </View>

  );

};

const styles = StyleSheet.create({

  container: { flex: 1, justifyContent: 'center', alignItems: 'center', padding: 20 },

  title: { fontSize: 24, marginBottom: 20 },

  detailText: { fontSize: 18, marginBottom: 10, textAlign: 'center' },

});

export default DetalhesPontoTuristico;

Entendimento:

useRoute(): Fornece acesso ao objeto route, que contém informações sobre a rota atual, incluindo params.

route.params: O objeto que contém todos os parâmetros passados para esta tela.

Desafio da Aula

No seu aplicativo "Guia Turístico":

Configure a navegação usando React Navigation (Stack Navigator).

Crie uma tela ListaPontosTuristicos e uma tela DetalhesPontoTuristico.

Na ListaPontosTuristicos, adicione um botão ou item clicável que, ao ser pressionado, navegue para a DetalhesPontoTuristico.

Passe um parâmetro (ex: o ID do ponto turístico) da ListaPontosTuristicos para a DetalhesPontoTuristico.

Na DetalhesPontoTuristico, exiba o parâmetro recebido.

Faça commits no Git com as alterações.

Próximos Passos

Na próxima aula, vamos aprofundar no consumo de APIs externas para carregar dados dinamicamente.

---