**Objetivos**

* Combinar a **navegação entre telas** com a exibição de **dados dinâmicos** de uma API.
* Aprender a passar objetos completos como **parâmetros de rota**.
* Exibir os detalhes de um item clicado na tela de destino.
* Discutir o problema do **"Prop Drilling"** e introduzir soluções iniciais (contexto).
* **BÔNUS:** Princípios de responsabilidade única e coesão de componentes.

**Revisão Rápida**

* Como fazemos requisições GET para uma API usando Axios e useEffect?
* Qual a função do FlatList e como ele melhora a performance em listas?

**Conteúdo Teórico**

* **Conectando Navegação e Dados:**
  + Aprimorar a FlatList para que cada item seja clicável.
  + Usar o navigation.navigate() para ir para a tela de detalhes.
  + Passar o **objeto completo** do ponto turístico como parâmetro para a tela de detalhes.
* **Recebendo e Exibindo Parâmetros Completos:**
  + Na tela de detalhes, acessar o objeto completo via route.params.
  + Exibir as informações detalhadas do ponto turístico.
* **Refatorando PontoTuristicoCard para Interação:**
  + Transformar o PontoTuristicoCard em um componente clicável usando TouchableOpacity.
  + Enviar uma função onPress como prop para o card.
* **O Problema do "Prop Drilling":**
  + O que é: passar props através de múltiplos níveis de componentes, mesmo que um componente intermediário não precise delas.
  + Por que é um problema: torna o código mais difícil de ler, manter e refatorar.
  + Introdução à necessidade de um **gerenciamento de estado global** (será abordado em aulas futuras).
* **Princípios de Design de Componentes:**
  + **Responsabilidade Única:** Cada componente deve ter apenas um motivo para mudar.
  + **Coesão:** Componentes devem manter funções relacionadas juntas.

**Prática Laboratorial**

**Passo 1: Adaptar o Componente PontoTuristicoCard para ser Clicável**

* **Ação:** Vamos transformar nosso PontoTuristicoCard em um componente que pode ser clicado, passando uma função onPress para ele.
* **Mudança em components/PontoTuristicoCard.js:**
* // components/PontoTuristicoCard.js
* import React from 'react';
* import { View, Text, StyleSheet, TouchableOpacity } from 'react-native'; // <--- Importe TouchableOpacity
* // O componente agora também recebe a prop 'onPress'
* const PontoTuristicoCard = ({ nome, descricao, onPress }) => { // <--- Desestruturação das props e adição de onPress
* return (
* <TouchableOpacity onPress={onPress} style={styles.touchable}> {/\* <--- Torna o card clicável \*/}
* <View style={styles.card}>
* <Text style={styles.titulo}>{nome}</Text>
* <Text style={styles.descricao}>{descricao}</Text>
* </View>
* </TouchableOpacity>
* );
* };
* const styles = StyleSheet.create({
* touchable: { // Estilo para garantir que o TouchableOpacity preencha o card
* width: '100%', // Para que o TouchableOpacity ocupe toda a largura do item
* },
* card: {
* backgroundColor: '#fff',
* padding: 15,
* marginVertical: 10,
* marginHorizontal: 20,
* borderRadius: 8,
* shadowColor: '#000',
* shadowOffset: { width: 0, height: 2 },
* shadowOpacity: 0.1,
* shadowRadius: 3.84,
* elevation: 5,
* },
* titulo: {
* fontSize: 20,
* fontWeight: 'bold',
* marginBottom: 5,
* color: '#333',
* },
* descricao: {
* fontSize: 14,
* color: '#666',
* },
* });
* export default PontoTuristicoCard;
* **Entendimento:**
  + TouchableOpacity: É um componente que envolve outros componentes e responde a toques, diminuindo a opacidade para dar feedback visual.
  + A prop onPress do TouchableOpacity é atribuída à função onPress que será recebida pelas props do PontoTuristicoCard.

**Passo 2: Passar Dados Completos na Navegação (ListaPontosTuristicos.js)**

* **Ação:** Modificaremos a ListaPontosTuristicos para que, ao clicar em um item, o aplicativo navegue para DetalhesPonto e passe o **objeto completo** do ponto turístico.
* **Mudança em screens/ListaPontosTuristicos.js:**
* // screens/ListaPontosTuristicos.js
* import React, { useState, useEffect } from 'react';
* import { View, Text, StyleSheet, ActivityIndicator, FlatList } from 'react-native';
* import PontoTuristicoCard from '../components/PontoTuristicoCard';
* import api from '../services/api';
* import { useNavigation } from '@react-navigation/native'; // <--- Importe useNavigation
* const ListaPontosTuristicos = () => {
* const navigation = useNavigation(); // <--- Hook para acessar a navegação
* const [pontosTuristicos, setPontosTuristicos] = useState([]);
* const [isLoading, setIsLoading] = useState(true);
* const [error, setError] = useState(null);
* useEffect(() => {
* const fetchPontosTuristicos = async () => {
* try {
* const response = await api.get('/posts');
* const dadosAdaptados = response.data.map(item => ({
* id: String(item.id),
* nome: item.title,
* descricao: item.body,
* imagem: `https://picsum.photos/id/${item.id % 100}/150/150`,
* // Supondo que você queira passar mais detalhes da API,
* // como o corpo completo do post, etc.
* detalhesCompletos: item.body + ' ' + item.title, // Exemplo de detalhe adicional
* // lat: -25.0 + (item.id \* 0.01), // Adicionar coordenadas se a API as fornecer
* // lon: -49.0 + (item.id \* 0.01),
* }));
* setPontosTuristicos(dadosAdaptados);
* } catch (err) {
* console.error("Erro ao buscar dados:", err);
* setError("Não foi possível carregar os pontos turísticos.");
* } finally {
* setIsLoading(false);
* }
* };
* fetchPontosTuristicos();
* }, []);
* // <--- Nova função para lidar com o clique no item
* const handlePontoPress = (ponto) => {
* navigation.navigate('DetalhesPonto', { pontoDetalhes: ponto }); // <--- Passa o objeto 'ponto' completo
* };
* if (isLoading) {
* return (
* <View style={styles.loadingContainer}>
* <ActivityIndicator size="large" color="#0000ff" />
* <Text style={styles.loadingText}>Carregando pontos turísticos...</Text>
* </View>
* );
* }
* if (error) {
* return (
* <View style={styles.errorContainer}>
* <Text style={styles.errorText}>{error}</Text>
* </View>
* );
* }
* return (
* <View style={styles.container}>
* <Text style={styles.mainTitle}>Pontos Turísticos</Text>
* <FlatList
* data={pontosTuristicos}
* keyExtractor={(item) => item.id}
* renderItem={({ item }) => (
* <PontoTuristicoCard
* nome={item.nome}
* descricao={item.descricao}
* onPress={() => handlePontoPress(item)} // <--- Passa a função com o item clicado
* />
* )}
* />
* </View>
* );
* };
* const styles = StyleSheet.create({
* container: { flex: 1, backgroundColor: '#f5f5f5', paddingTop: 50 },
* mainTitle: { fontSize: 28, fontWeight: 'bold', textAlign: 'center', marginBottom: 20, color: '#333' },
* loadingContainer: { flex: 1, justifyContent: 'center', alignItems: 'center', backgroundColor: '#f5f5f5' },
* loadingText: { marginTop: 10, fontSize: 16, color: '#666' },
* errorContainer: { flex: 1, justifyContent: 'center', alignItems: 'center', backgroundColor: '#ffe0e0' },
* errorText: { fontSize: 16, color: 'red', textAlign: 'center', marginHorizontal: 20 }
* });
* export default ListaPontosTuristicos;
* **Entendimento:**
  + A função handlePontoPress agora recebe o item completo da lista.
  + navigation.navigate('DetalhesPonto', { pontoDetalhes: ponto }): Passamos o **objeto completo** (ponto) sob a chave pontoDetalhes. Isso evita que a tela de detalhes precise fazer outra requisição para obter os dados já disponíveis.

**Passo 3: Exibir Detalhes Completos na Tela de Destino (DetalhesPontoTuristico.js)**

* **Ação:** Atualizaremos a tela DetalhesPontoTuristico para receber e exibir o objeto completo do ponto turístico.
* **Mudança em screens/DetalhesPontoTuristico.js:**
* // screens/DetalhesPontoTuristico.js
* import React from 'react';
* import { View, Text, StyleSheet, Button, ScrollView } from 'react-native'; // <--- Importe ScrollView
* import { useRoute, useNavigation } from '@react-navigation/native';
* const DetalhesPontoTuristico = () => {
* const route = useRoute();
* const navigation = useNavigation();
* // <--- Acessando o objeto completo 'pontoDetalhes'
* const { pontoDetalhes } = route.params;
* if (!pontoDetalhes) { // <--- Tratamento caso os detalhes não sejam passados
* return (
* <View style={styles.container}>
* <Text style={styles.errorText}>Detalhes do ponto turístico não encontrados.</Text>
* <Button title="Voltar" onPress={() => navigation.goBack()} />
* </View>
* );
* }
* return (
* <ScrollView style={styles.scrollViewContainer}> {/\* <--- Para conteúdo que pode ser longo \*/}
* <View style={styles.contentContainer}>
* <Text style={styles.title}>{pontoDetalhes.nome}</Text>
* <Text style={styles.detailText}>ID: {pontoDetalhes.id}</Text>
* <Text style={styles.descriptionText}>{pontoDetalhes.descricao}</Text>
* <Text style={styles.extraDetails}>Detalhes Completos: {pontoDetalhes.detalhesCompletos}</Text> {/\* <--- Exibindo o novo detalhe \*/}
* {/\* Aqui você pode adicionar uma imagem, localização no mapa, etc. \*/}
* <Button title="Voltar para a Lista" onPress={() => navigation.goBack()} />
* </View>
* </ScrollView>
* );
* };
* const styles = StyleSheet.create({
* scrollViewContainer: {
* flex: 1,
* backgroundColor: '#f5f5f5',
* },
* contentContainer: {
* padding: 20,
* alignItems: 'center',
* },
* title: {
* fontSize: 28,
* fontWeight: 'bold',
* marginBottom: 15,
* textAlign: 'center',
* color: '#333',
* },
* detailText: {
* fontSize: 16,
* marginBottom: 10,
* color: '#666',
* },
* descriptionText: {
* fontSize: 16,
* lineHeight: 24,
* textAlign: 'justify',
* marginBottom: 20,
* color: '#444',
* },
* extraDetails: {
* fontSize: 14,
* fontStyle: 'italic',
* color: '#888',
* marginBottom: 30,
* },
* errorText: {
* fontSize: 18,
* color: 'red',
* textAlign: 'center',
* marginVertical: 50,
* }
* });
* export default DetalhesPontoTuristico;
* **Entendimento:**
  + route.params.pontoDetalhes: Acessamos o objeto completo do ponto turístico que foi passado.
  + Podemos agora exibir qualquer propriedade desse objeto (nome, descricao, id, detalhesCompletos, etc.).
  + Um simples tratamento para o caso de pontoDetalhes ser nulo foi adicionado.

**O Problema do "Prop Drilling" (Discussão)**

* **Cenário:** Imagine que o PontoTuristicoCard precisa de uma nova informação que está no App.js. Se App.js passa para ListaPontosTuristicos, que por sua vez passa para PontoTuristicoCard, isso é prop drilling.
* **Visualização:**
* App.js
* | (passa 'userInfo')
* V
* MainNavigator.js
* | (passa 'userInfo')
* V
* ListaPontosTuristicos.js
* | (passa 'userInfo')
* V
* PontoTuristicoCard.js <-- Só PontoTuristicoCard precisa de 'userInfo'
* **Consequências:**
  + **Dificuldade de Manutenção:** Mudar o nome de uma prop ou adicionar uma nova exige alterar todos os componentes intermediários.
  + **Baixa Legibilidade:** É difícil rastrear de onde uma prop realmente vem.
  + **Reusabilidade Reduzida:** Componentes intermediários se tornam excessivamente acoplados a props que não lhes pertencem.
* **Solução (Visão Geral):** Para resolver isso, usamos gerenciamento de estado global (como **Context API** ou Redux), que veremos em aulas futuras. Isso permite que componentes distantes acessem dados sem passá-los por cada nível.

**Desafio da Aula**

* No seu aplicativo "Guia Turístico":
  + Adapte o componente PontoTuristicoCard.js para receber e usar uma prop onPress, tornando-o clicável via TouchableOpacity.
  + Na ListaPontosTuristicos.js, modifique o renderItem da FlatList para que, ao clicar em um PontoTuristicoCard, ele chame navigation.navigate('DetalhesPonto', { pontoDetalhes: item }), passando o **objeto item completo**.
  + Na DetalhesPontoTuristico.js, acesse o pontoDetalhes via route.params e exiba todas as informações relevantes do ponto turístico (ID, nome, descrição, e a imagem se você já tiver adicionado ela no card).
  + **Opcional:** Adicione um tratamento simples (if (!pontoDetalhes)) na tela de detalhes caso os parâmetros não sejam encontrados.
* Faça **commits no Git** com as alterações.

**Próximos Passos**

* Na próxima aula, vamos explorar como persistir dados localmente no dispositivo usando AsyncStorage, para que as informações não se percam ao fechar o app.