**Implementando Mapas e Marcadores**

Condições de conclusão

**Implementando Mapas e Marcadores**

Localizando o Guia: Adicionando Mapas Interativos e Marcadores

**Objetivos**

* Entender a importância de **integrar mapas** em aplicativos turísticos.
* Instalar e configurar a biblioteca **react-native-maps**.
* Exibir um **MapView** e posicioná-lo em uma localização inicial.
* Adicionar **marcadores (Marker)** no mapa para representar os pontos turísticos.
* Personalizar os marcadores com **balões de informação (Callout)**.
* **BÔNUS:** Considerações sobre performance de mapas e gerenciamento de dados de localização.

**Revisão Rápida**

* Como configuramos a navegação por abas com BottomTabNavigator?
* Qual a vantagem de aninhar Stack Navigators dentro de Tab Navigators?

**Conteúdo Teórico**

* **Por que Mapas em Apps de Turismo?**
  + Ajuda na localização e planejamento de rotas.
  + Visualização intuitiva dos pontos de interesse.
  + Aprimora a experiência de exploração.
* **Introdução ao react-native-maps:**
  + Uma biblioteca popular que fornece componentes de mapa nativos para iOS (Apple Maps) e Android (Google Maps).
  + Permite interações como zoom, pan e exibição de marcadores.
* **Componente MapView:**
  + O componente principal para exibir o mapa.
  + Propriedades importantes: initialRegion (posição inicial do mapa e zoom), showsUserLocation (mostra a localização do usuário).
* **Componente Marker:**
  + Usado para adicionar pinos/marcadores em locais específicos no mapa.
  + Propriedades: coordinate (latitude e longitude do marcador), title, description.
  + **Callout**: Um componente filho do Marker que exibe informações adicionais quando o marcador é clicado (o "balão" de informação).
* **Obtenção de Coordenadas:**
  + Para exibir pontos no mapa, você precisará de dados com **latitude e longitude**.
  + Por enquanto, podemos usar coordenadas fictícias ou dados de uma API que as forneça.

**Prática Laboratorial**

**Passo 1: Instalar react-native-maps**

* **Ação:** Primeiro, instale a biblioteca de mapas.
* **Comando:** No terminal, na raiz do seu projeto:
* expo install react-native-maps
* **Explicação:**
  + Este comando instala a versão compatível do react-native-maps com o seu ambiente Expo.

**Passo 2: Implementar o Mapa na MapaScreen.js**

* **Ação:** Vamos modificar a MapaScreen.js para exibir um MapView e alguns marcadores de exemplo.
* **Mudança em screens/MapaScreen.js:**
* // screens/MapaScreen.js
* import React from 'react';
* import { View, Text, StyleSheet } from 'react-native';
* import MapView, { Marker, Callout } from 'react-native-maps'; // <--- Importe MapView, Marker, Callout
* import { useNavigation } from '@react-navigation/native'; // Para navegação a partir do Callout
* // Dados fictícios de pontos turísticos com coordenadas
* // Em breve, usaremos os dados da API com coordenadas reais ou adaptadas
* const pontosDeExemplo = [
* { id: '1', nome: 'Jardim Botânico', descricao: 'Estufa icônica de vidro.', latitude: -25.4411, longitude: -49.2329 },
* { id: '2', nome: 'Ópera de Arame', descricao: 'Teatro com estrutura de tubo de aço.', latitude: -25.3934, longitude: -49.2608 },
* { id: '3', nome: 'Parque Tanguá', descricao: 'Mirante e cascata em antiga pedreira.', latitude: -25.3670, longitude: -49.2740 },
* ];
* const MapaScreen = () => {
* const navigation = useNavigation();
* // <--- Região inicial do mapa (Curitiba, Brasil)
* const initialRegion = {
* latitude: -25.4284,
* longitude: -49.2733,
* latitudeDelta: 0.15, // Nível de zoom (menor valor = mais zoom)
* longitudeDelta: 0.15,
* };
* // <--- Função para ir para a tela de detalhes ao clicar no Callout
* const handleMarkerPress = (ponto) => {
* // Navega para a tela de detalhes, passando o objeto ponto
* navigation.navigate('ExplorarStack', { // Nome do Stack Navigator onde DetalhesPonto está
* screen: 'DetalhesPonto', // Nome da tela de detalhes
* params: { pontoDetalhes: ponto },
* });
* };
* return (
* <View style={styles.container}>
* <MapView
* style={styles.map}
* initialRegion={initialRegion} // <--- Define a região inicial
* showsUserLocation={true} // <--- Mostra a localização do usuário (se permitida)
* >
* {pontosDeExemplo.map(ponto => ( // <--- Mapeia os pontos para Marcadores
* <Marker
* key={ponto.id}
* coordinate={{ latitude: ponto.latitude, longitude: ponto.longitude }}
* title={ponto.nome}
* description={ponto.descricao}
* >
* <Callout onPress={() => handleMarkerPress(ponto)}> {/\* <--- Callout clicável \*/}
* <View style={styles.calloutContainer}>
* <Text style={styles.calloutTitle}>{ponto.nome}</Text>
* <Text style={styles.calloutDescription}>{ponto.descricao}</Text>
* <Text style={styles.calloutLink}>Ver Detalhes »</Text>
* </View>
* </Callout>
* </Marker>
* ))}
* </MapView>
* </View>
* );
* };
* const styles = StyleSheet.create({
* container: { flex: 1 },
* map: {
* width: '100%',
* height: '100%',
* },
* calloutContainer: { // Estilo do balão de informação
* width: 150,
* padding: 5,
* },
* calloutTitle: {
* fontWeight: 'bold',
* fontSize: 16,
* marginBottom: 5,
* },
* calloutDescription: {
* fontSize: 12,
* color: '#666',
* },
* calloutLink: {
* fontSize: 12,
* color: 'blue',
* marginTop: 5,
* textAlign: 'right',
* }
* });
* export default MapaScreen;
* **Entendimento:**
  + MapView: É o componente que renderiza o mapa. initialRegion define a latitude, longitude e o zoom.
  + Marker: Coloca um pino no mapa. A propriedade coordinate é essencial.
  + Callout: É o balão de informação que aparece ao clicar no marcador. Dentro dele, você pode colocar qualquer componente React Native.
  + handleMarkerPress: Uma função para navegar para a tela de detalhes quando o Callout de um marcador é clicado. **Atenção ao navigation.navigate('ExplorarStack', { screen: 'DetalhesPonto', params: { pontoDetalhes: ponto } })**: Quando você navega entre stacks aninhadas (de uma aba para uma tela dentro de outra aba), precisa especificar o nome da Stack e depois o nome da tela e seus parâmetros.

**Passo 3: Integrar Dados Dinâmicos com Latitude/Longitude (Opcional, Depende da API)**

* **Ação:** Se sua API fornecer latitude e longitude, você pode usar os dados reais. Caso contrário, você pode adicionar valores fictícios aos seus dados adaptados na ListaPontosTuristicos.js para simular.
* **Mudança em screens/ListaPontosTuristicos.js (no fetchPontosTuristicos):**
* // screens/ListaPontosTuristicos.js (trecho do useEffect)
* // ...
* 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`,
* // <--- Adicione latitude e longitude fictícias/reais se disponíveis
* latitude: -25.4284 + (Math.random() - 0.5) \* 0.1, // Exemplo de ajuste aleatório para espalhar marcadores
* longitude: -49.2733 + (Math.random() - 0.5) \* 0.1,
* detalhesCompletos: item.body + ' ' + item.title,
* }));
* setPontosTuristicos(dadosAdaptados);
* // ...
* **E então, use esses pontosTuristicos no MapaScreen.js:**
* // screens/MapaScreen.js (substitua pontosDeExemplo)
* // ...
* const { pontosTuristicos, isLoading, error } = useSomeGlobalStateOrPassViaProps(); // <--- IDEAL: BUSCAR DO CONTEXTO/STORE ou passar via rota
* // Para esta aula, vamos simular que os dados estão disponíveis
* // Na vida real, você buscaria esses dados de um contexto (como o de favoritos)
* // ou de outra requisição API no próprio MapaScreen.
* // Ou, se a lista for pequena, passaria via navigation.navigate.
* // Por simplicidade para a aula, mantenha 'pontosDeExemplo' ou adicione um useEffect aqui.
* // Exemplo de como você faria se os pontos fossem globais ou via API na própria tela do mapa:
* const [mapaPontos, setMapaPontos] = useState([]);
* useEffect(() => {
* // Lógica para carregar pontos aqui, similar a ListaPontosTuristicos
* // Ou, se já estiver no contexto de favoritos:
* // const { favoritePoints, isLoading } = useFavorites();
* // setMapaPontos(favoritePoints);
* // Por simplicidade, usaremos os dados fictícios por agora,
* // mas o objetivo final é usar dados reais do seu app.
* setMapaPontos(pontosDeExemplo); // Ou seus dados dinâmicos
* }, []);
* return (
* <View style={styles.container}>
* <MapView
* style={styles.map}
* initialRegion={initialRegion}
* showsUserLocation={true}
* >
* {mapaPontos.map(ponto => ( // <--- Mapeia os pontos para Marcadores
* <Marker
* key={ponto.id}
* coordinate={{ latitude: ponto.latitude, longitude: ponto.longitude }}
* title={ponto.nome}
* description={ponto.descricao}
* >
* <Callout onPress={() => handleMarkerPress(ponto)}>
* <View style={styles.calloutContainer}>
* <Text style={styles.calloutTitle}>{ponto.nome}</Text>
* <Text style={styles.calloutDescription}>{ponto.descricao}</Text>
* <Text style={styles.calloutLink}>Ver Detalhes »</Text>
* </View>
* </Callout>
* </Marker>
* ))}
* </MapView>
* </View>
* );
* **Nota Importante:** Trazer os dados para o mapa pode ser feito de algumas maneiras:
  1. **Contexto/Gerenciamento de Estado Global:** Se os pontos já são gerenciados globalmente (como seus favoritos), a MapaScreen pode simplesmente consumir esse contexto. **Esta é a abordagem recomendada a longo prazo.**
  2. **Passar via Parâmetro de Rota:** Se a tela do mapa é acessada de um contexto específico que já tem os dados.
  3. **Nova Requisição API:** A MapaScreen faz sua própria requisição para buscar os pontos (útil se a lista for muito grande e você precisar de paginação, por exemplo).
  4. Para esta aula, o foco é na **visualização do mapa e marcadores**, então manter os pontosDeExemplo fixos na MapaScreen é aceitável, mas tenha em mente o objetivo de usar dados dinâmicos.

**Desafio da Aula**

* No seu aplicativo "Guia Turístico":
  + Instale o react-native-maps.
  + No seu MapaScreen.js:
    - Implemente o componente MapView e configure uma initialRegion para Curitiba.
    - Adicione alguns Markers estáticos para pontos turísticos de Curitiba (invente latitude e longitude se necessário, ou use as do Google Maps).
    - Para cada Marker, inclua um Callout exibindo o nome e uma breve descrição do ponto.
    - Faça o Callout clicável para navegar para a tela DetalhesPontoTuristico, passando as informações do ponto (lembre-se de como navegar entre abas/pilhas aninhadas!).
  + **Opcional:** Se você adicionou latitude e longitude aos seus dados da API na Aula 4, tente usar esses dados dinâmicos para renderizar os marcadores no mapa, em vez dos pontos estáticos. Você precisará de uma forma de passar esses dados para a MapaScreen (talvez através de um contexto futuro de "todos os pontos", ou fazendo uma nova requisição na MapaScreen se a lista for grande).
* Faça **commits no Git** com as alterações.

**Próximos Passos**

* Na próxima aula, vamos aprofundar na interação com o usuário, adicionando funcionalidades de pesquisa e filtros na lista de pontos turísticos.