

Nessa atividade revisaremos tudo o que utilizamos nas micro atividades anteriores.

Além disso, veremos também como o React Native responde a eventos e qual lógica de manipulação será utilizada neste exercício.

.

Eros Santos de Vasconcelos/ 202307120545

Polo Iputinga

RPG0023 - Vamos criar um App! – 4º semestre

Objetivo da Prática

- Implementar um sistema de cadastro de fornecedores.
- Exibir lista de fornecedores com opção de pesquisa e filtro.
- Permitir upload de imagens associadas aos fornecedores.
- Garantir uma experiência de usuário intuitiva.

Requisitos Funcionais

- **Cadastro de Fornecedores:**
 - Campos obrigatórios: Nome, Endereço, Contato, Categorias.
 - Uso dos componentes <Text>, <TextInput>, <Image>.

- **Listagem de Fornecedores:**
 - Exibição dos dados cadastrados.
 - Possibilidade de busca e filtro por categoria/localização.
- **Associação de Imagens:**
 - Upload de logotipos e imagens representativas.
- **Experiência de Usuário:**
 - Interface intuitiva e responsiva.

Códigos Solicitados:

1. index.tsx:

```
• import { Tabs } from 'expo-router';
• import React from 'react';
• import { Platform } from 'react-native';
•
• import { HapticTab } from '@/components/HapticTab';
• import { IconSymbol } from '@/components/ui/IconSymbol';
• import TabBarBackground from '@/components/ui/TabBarBackground';
• import { Colors } from '@/constants/Colors';
• import { useColorScheme } from '@/hooks/useColorScheme';
•
• export default function TabLayout() {
•   const colorScheme = useColorScheme();
•
•   return (
•     <Tabs
•       screenOptions={{
•         tabBarActiveTintColor: Colors[colorScheme ?? 'light'].tint,
•         headerShown: false,
•         tabBarButton: HapticTab,
•         tabBarBackground: TabBarBackground,
•         tabBarStyle: Platform.select({
```

```

•     ios: {
•       position: 'absolute',
•     },
•     default: {},
•   },
• }
• >
• <Tabs.Screen
•   name="index"
•   options={{
•     title: 'Home',
•     tabBarIcon: ({ color }) => <IconSymbol size={28}
name="house.fill" color={color} />,
•   }}
• />
• <Tabs.Screen
•   name="explore"
•   options={{
•     title: 'Explore',
•     tabBarIcon: ({ color }) => <IconSymbol size={28}
name="paperplane.fill" color={color} />,
•   }}
• />
• </Tabs>
• );
• }

```

2. _layout.tsx:

```

• import { DarkTheme, DefaultTheme, ThemeProvider } from '@react-
navigation/native';
• import { useFonts } from 'expo-font';
• import { Stack } from 'expo-router';
• import * as SplashScreen from 'expo-splash-screen';
• import { StatusBar } from 'expo-status-bar';
• import { useEffect } from 'react';
• import 'react-native-reanimated';
•
• import { useColorScheme } from '@/hooks/useColorScheme';
• import FornecedoresProvider from './FornecedoresContext';
• SplashScreen.preventAutoHideAsync();
•
• export default function RootLayout() {
•   const colorScheme = useColorScheme();
•   const [loaded] = useFonts({
•     SpaceMono: require('../assets/fonts/SpaceMono-Regular.ttf'),
•   });
•

```

```

•     useEffect(() => {
•       if (loaded) {
•         SplashScreen.hideAsync();
•       }
•     }, [loaded]);
•
•     if (!loaded) {
•       return null;
•     }
•
•   return (
•     <FornecedoresProvider>
•       <ThemeProvider value={colorScheme === 'dark' ? DarkTheme :
DefaultTheme}>
•         <Stack>
•           <Stack.Screen name="(tabs)" options={{ headerShown: false }} />
•           <Stack.Screen name="+not-found" />
•         </Stack>
•         <StatusBar style="auto" />
•       </ThemeProvider>
•     </FornecedoresProvider>
•   );
}

```

3. cadastro.tsx:

```

• import { useState } from "react";
• import { View, Text, TextInput, Button, Alert, Image, StyleSheet } from
  "react-native";
• import * as ImagePicker from "expo-image-picker";
• import { useRouter } from "expo-router";
• import { useFornecedores } from "./FornecedoresContext";
•
• export default function CadastroScreen() {
•   const router = useRouter();
•   const { adicionarFornecedor } = useFornecedores();
•
•   const [nome, setNome] = useState("");
•   const [endereco, setEndereco] = useState("");
•   const [contato, setContato] = useState("");
•   const [categoria, setCategoria] = useState("");
•   const [imagem, setImagem] = useState<string | undefined>();
•
•   const escolherImagem = async () => {
•     let result = await ImagePicker.launchImageLibraryAsync({
•       mediaTypes: ImagePicker.MediaTypeOptions.Images,
•       allowsEditing: true,
•       aspect: [4, 4],
•       quality: 1,
•     });
•

```

```

•     if (!result.canceled) {
•         setImagem(result.assets[0].uri);
•     }
• };
•
•     const cadastrarFornecedor = () => {
•         if (!nome || !endereco || !contato || !categoria) {
•             Alert.alert("Erro", "Todos os campos devem ser preenchidos!");
•             return;
•         }
•
•         const novoFornecedor = { id: Date.now().toString(), nome, endereco,
•         contato, categoria, imagem };
•         adicionarFornecedor(novoFornecedor);
•         Alert.alert("Sucesso", "Fornecedor cadastrado!");
•         router.push("/listagem");
•     };
•
•     return (
•         <View style={styles.container}>
•             <Text style={styles.titulo}>Cadastro de Fornecedor</Text>
•             <TextInput style={styles.input} placeholder="Nome" value={nome}
•             onChangeText={setNome} />
•             <TextInput style={styles.input} placeholder="Endereço"
•             value={endereco} onChangeText={setEndereco} />
•             <TextInput style={styles.input} placeholder="Contato" value={contato}
•             onChangeText={setContato} keyboardType="phone-pad" />
•             <TextInput style={styles.input} placeholder="Categoria"
•             value={categoria} onChangeText={setCategoria} />
•             <Button title="Selecionar Imagem" onPress={escolherImagem} />
•             {imagem && <Image source={{ uri: imagem }} style={styles.imagem} />}
•             <Button title="Cadastrar" onPress={cadastrarFornecedor} />
•         </View>
•     );
• }
•
• const styles = StyleSheet.create({
•     container: { flex: 1, padding: 20, backgroundColor: "white" },
•     titulo: { fontSize: 24, fontWeight: "bold", marginBottom: 20, textAlign: "center" },
•     input: { height: 40, borderColor: "gray", borderWidth: 1, marginBottom: 10, paddingHorizontal: 10 },
•     imagem: { width: 100, height: 100, alignSelf: "center", marginVertical: 10 },
• });
•

```

4. editar.tsx:

```

5. import { useState } from "react";
6. import { View, Text, TextInput, Button, Alert, Image, StyleSheet }
   from "react-native";
7. import * as ImagePicker from "expo-image-picker";

```

```
8. import { useLocalSearchParams, useRouter } from "expo-router";
9.
10. export default function EditarFornecedor() {
11.   const router = useRouter();
12.   const params = useLocalSearchParams();
13.
14.   const getString = (value: string | string[] | undefined) =>
    (Array.isArray(value) ? value[0] : value) || "";
15.
16.   const [novoNome, setNovoNome] = useState(getString(params.nome));
17.   const [novoEndereco, setNovoEndereco] =
    useState(getString(params.endereco));
18.   const [novoContato, setNovoContato] =
    useState(getString(params.contato));
19.   const [novaCategoria, setNovaCategoria] =
    useState(getString(params.categoria));
20.   const [imagemUri, setImagemUri] =
    useState(getString(params.imagem));
21.
22.   const escolherNovaImagem = async () => {
23.     let result = await ImagePicker.launchImageLibraryAsync({
24.       mediaTypes: ImagePicker.MediaTypeOptions.Images,
25.       allowsEditing: true,
26.       aspect: [4, 4],
27.       quality: 1,
28.     });
29.
30.     if (!result.canceled) {
31.       setImagemUri(result.assets[0].uri);
32.     }
33.   };
34.
35.   const salvarEdicao = () => {
36.     if (!novoNome || !novoEndereco || !novoContato || !novaCategoria)
37.     {
38.       Alert.alert("Erro", "Todos os campos devem ser preenchidos!");
39.       return;
40.     }
41.     Alert.alert("Sucesso", "Fornecedor atualizado com sucesso!");
42.     router.push("/listagem");
43.   };
44.
45.   return (
46.     <View style={styles.container}>
47.       <Text style={styles.titulo}>Editar Fornecedor</Text>
48.       <TextInput style={styles.input} value={novoNome}
        onChangeText={setNovoNome} />
49.       <TextInput style={styles.input} value={novoEndereco}
        onChangeText={setNovoEndereco} />
50.       <TextInput style={styles.input} value={novoContato}
        onChangeText={setNovoContato} keyboardType="phone-pad" />
```

```

51.      <TextInput style={styles.input} value={novaCategoria}
52.        onChangeText={setNovaCategoria} />
53.      {imagemUri && <Image source={{ uri: imagemUri }} 
54.        style={styles.imagem} />}
55.      <Button title="Alterar Imagem" onPress={escolherNovaImagem} />
56.      <Button title="Salvar Alterações" onPress={salvarEdicao} />
57.    </View>
58.  );
59.// Estilos
60.const styles = StyleSheet.create({
61.  container: {
62.    flex: 1,
63.    padding: 20,
64.    backgroundColor: "#f9f9f9",
65.  },
66.  titulo: {
67.    fontSize: 24,
68.    fontWeight: "bold",
69.    textAlign: "center",
70.    marginBottom: 15,
71.  },
72.  input: {
73.    height: 40,
74.    borderWidth: 1,
75.    borderColor: "gray",
76.    marginBottom: 10,
77.    paddingHorizontal: 10,
78.    borderRadius: 5,
79.    backgroundColor: "white",
80.  },
81.  imagem: {
82.    width: 100,
83.    height: 100,
84.    alignSelf: "center",
85.    marginVertical: 10,
86.    borderRadius: 10,
87.  },
88.});

```

5. FornecedoresContext.tsx:

```

1. import { createContext, useContext, useState, ReactNode } from "react";
2.
3. export type Fornecedor = {
4.   id: string;
5.   nome: string;
6.   endereco: string;
7.   contato: string;
8.   categoria: string;
9.   imagem?: string;

```

```

10.};
11.
12.type FornecedoresContextType = {
13.  fornecedores: Fornecedor[];
14.  adicionarFornecedor: (fornecedor: Fornecedor) => void;
15.  editarFornecedor: (fornecedor: Fornecedor) => void;
16.};
17.
18.const FornecedoresContext = createContext<FornecedoresContextType | undefined>(undefined);
19.
20.export function FornecedoresProvider({ children }: { children: ReactNode }) {
21.  const [fornecedores, setFornecedores] = useState<Fornecedor[]>([]);
22.
23.  const adicionarFornecedor = (fornecedor: Fornecedor) => {
24.    setFornecedores((prev) => [...prev, fornecedor]);
25.  };
26.
27.  const editarFornecedor = (fornecedorAtualizado: Fornecedor) => {
28.    setFornecedores((prev) =>
29.      prev.map((f) => (f.id === fornecedorAtualizado.id ?
30.        fornecedorAtualizado : f))
31.    );
32.  };
33.  return (
34.    <FornecedoresContext.Provider value={{ fornecedores,
35.      adicionarFornecedor, editarFornecedor }}>
36.      {children}
37.    </FornecedoresContext.Provider>
38.  );
39.
40.export function useFornecedores() {
41.  const context = useContext(FornecedoresContext);
42.  if (!context) {
43.    throw new Error("useFornecedores deve ser usado dentro de FornecedoresProvider");
44.  }
45.  return context;
46.}
47.
48.export default FornecedoresProvider;
49.

```

6. listagem.tsx:

```

50.import { View, Text, FlatList, Image, Button, StyleSheet, TextInput } from "react-native";
51.import { Picker } from "@react-native-picker/picker";
52.import { useRouter } from "expo-router";
53.import { useState } from "react";

```

```
54. import { useFornecedores } from "./FornecedoresContext";
55.
56. export default function ListagemScreen() {
57.   const { fornecedores } = useFornecedores();
58.   const router = useRouter();
59.
60.   // Estado para pesquisa e filtro
61.   const [pesquisa, setPesquisa] = useState("");
62.   const [categoriaSelecionada, setCategoriaSelecionada] =
63.     useState<string>("");
64.   // Filtrando fornecedores dinamicamente
65.   const fornecedoresFiltrados = fornecedores.filter((fornecedor) => {
66.     const nomeCorrespondente =
67.       fornecedor.nome.toLowerCase().includes(pesquisa.toLowerCase());
68.     const categoriaCorrespondente = categoriaSelecionada ?
69.       fornecedor.categoria === categoriaSelecionada : true;
70.     return nomeCorrespondente && categoriaCorrespondente;
71.   });
72.   return (
73.     <View style={styles.container}>
74.       <Text style={styles.titulo}>Lista de Fornecedores</Text>
75.       {/* Barra de pesquisa */}
76.       <TextInput
77.         style={styles.input}
78.         placeholder="Pesquisar fornecedores..."
79.         value={pesquisa}
80.         onChangeText={setPesquisa}
81.       />
82.
83.       {/* Filtro de categoria */}
84.       <Picker
85.         selectedValue={categoriaSelecionada}
86.         onValueChange={(itemValue: string) =>
87.           setCategoriaSelecionada(itemValue)}
88.         style={styles.picker}
89.       >
90.         <Picker.Item label="Todas as categorias" value="" />
91.         <Picker.Item label="Alimentos" value="Alimentos" />
92.         <Picker.Item label="Roupas" value="Roupas" />
93.         <Picker.Item label="Eletrônicos" value="Eletrônicos" />
94.       </Picker>
95.       <FlatList
96.         data={fornecedoresFiltrados}
97.         keyExtractor={({ item }) => item.id}
98.         renderItem={({ item }) => (
99.           <View style={styles.card}>
100.             {item.imagem && <Image source={{ uri: item.imagem }}>
101.               style={styles.imagem} />}
102.             <View style={styles.infoContainer}>
```

```
102.          <Text style={styles.nome}>{item.nome}</Text>
103.          <Text style={styles.texto}>📍 {item.endereco}</Text>
104.          <Text style={styles.texto}>📞 {item.contato}</Text>
105.          <Text style={styles.texto}>📌 {item.categoria}</Text>
106.          <Button
107.            title="Editar"
108.            onPress={() => router.push({ pathname: "/editar",
109.              params: { id: item.id } })}>
110.            />
111.          </View>
112.        )}>
113.      />
114.    </View>
115.  );
116.
117.
118.  const styles = StyleSheet.create({
119.    container: { flex: 1, padding: 20, backgroundColor: "#f9f9f9" },
120.    titulo: { fontSize: 24, fontWeight: "bold", marginBottom: 10 },
121.    input: { height: 40, borderColor: "#ccc", borderWidth: 1,
122.      marginBottom: 10, paddingLeft: 8, borderRadius: 5 },
123.    picker: { height: 50, marginBottom: 10 },
124.    card: { flexDirection: "row", padding: 15, backgroundColor:
125.      "white", marginVertical: 5, borderRadius: 5, alignItems: "center" },
126.    imagem: { width: 60, height: 60, borderRadius: 30, marginRight: 10
127.      },
128.    infoContainer: { flex: 1 },
129.    nome: { fontSize: 18, fontWeight: "bold" },
130.    texto: { fontSize: 14, color: "#555" },
131.  });
132.
```

Resultados da execução:

9:30



Bem-vindo ao Meeting!

CADASTRAR FORNECEDOR

LISTAR FORNECEDORES



Home



Explore



9:30



← cadastro

Cadastro de Fornecedor

Nome

Endereço

Contato

Categoria

SELECIONAR IMAGEM

CADASTRAR



9:25



← listagem

Lista de Fornecedores

Pesquisar fornecedores...

Todas as categorias



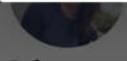
Todas as categorias

undefined

Alimentos

Roupas

Eletrônicos



Lis

EDITAR

9:42



← listagem

Lista de Fornecedores

Pesquisar fornecedores...

Todas as categorias ▾

Aatrox



📍 Los Angeles

📞 9487575497

❤️ Alimentos

[EDITAR](#)



Análise e Conclusão:

Desenvolvimento do Aplicativo

- 1. Criando um Componente Simples**
 - 2. Adicionando Elementos ao Componente**
 - 3. Utilizando Props**
 - 4. Utilizando Imagens**
-

Implementação Final do Aplicativo

Com base nas práticas anteriores, o aplicativo final incluirá:

- Cadastro: Formulário interativo para adicionar fornecedores.**
 - Listagem: Exibição dinâmica de fornecedores cadastrados.**
 - Filtragem e Pesquisa: Mecanismos para facilitar a busca.**
 - Imagens: Upload e exibição de imagens de fornecedores.**
-

Resultados Esperados

- Compreensão de React Native: Criação de componentes reutilizáveis.**
 - Gerenciamento de Estados e Props: Manipulação dinâmica dos dados.**
 - Exibição de Listas e Imagens: Renderização eficiente.**
 - Melhoria na Experiência do Usuário: Layout intuitivo e funcional.**
-

Conclusão

A missão prática proporcionou o aprendizado de conceitos fundamentais do React Native, resultando na construção de um aplicativo funcional para a empresa "Meeting". O conhecimento adquirido pode ser expandido para futuras melhorias e integrações com bancos de dados e APIs externas.