Wellington Cesar Fonseca

DESENVOLVIMENTO FULL STACK

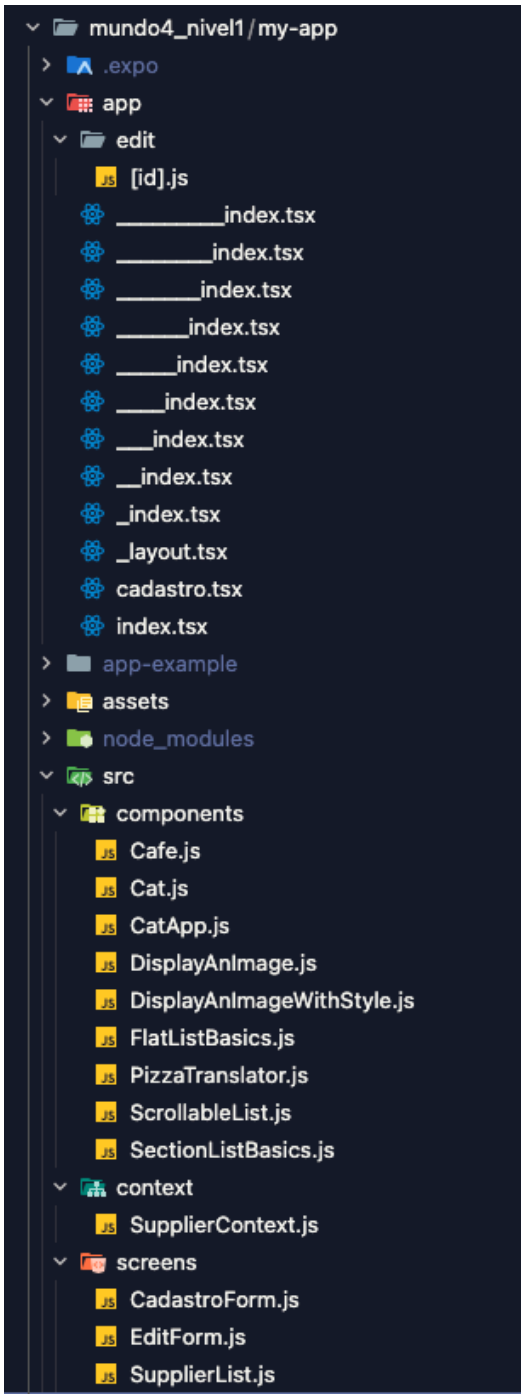
MISSÃO PRÁTICA | NÍVEL 4 | MUNDO 1

RPG0023 - VAMOS CRIAR UM APP!

OBJETIVO

1. Configurar o ambiente de desenvolvimento React Native;
2. Implementar a funcionalidade de entrada de texto em um componente React Native;
3. Implementar um Componente de Lista Dinâmica (ScrollView);
4. Implementar componentes React Native para exibir informações de forma dinâmica em listas;
5. Empregar elementos visuais em um aplicativo React Native.

MISSÃO PRÁTICA | VAMOS CRIAR UM APP!



app/edit/[id].js

```
import EditForm from '../../src/screens/EditForm';

export default function EditFormPage() {
 return <EditForm />;
}
```

app/index.tsx

```
import React from 'react';
import { SupplierProvider } from '../src/context/SupplierContext';
import SupplierList from '../src/screens/SupplierList';

const Index = () => {
 return (
 <SupplierProvider>
 <SupplierList />
 </SupplierProvider>
 );
};

export default Index;
```

app/cadastro.tsx

```
import React from 'react';
import CadastroForm from '../src/screens/CadastroForm';
import { SupplierProvider } from '../src/context/SupplierContext';

const Cadastro = () => {
 return (
 <SupplierProvider>
 <CadastroForm />
 </SupplierProvider>
 );
};

export default Cadastro;
```

app/src/context/SupplierContext.js

```
import React, { createContext, useState, useEffect } from 'react';
import AsyncStorage from '@react-native-async-storage/async-storage';

export const SupplierContext = createContext();

export const SupplierProvider = ({ children }) => {
 const [suppliers, setSuppliers] = useState([]);
 const [isLoading, setIsLoading] = useState(true);

 // Carregar fornecedores do armazenamento ao iniciar o app
 useEffect(() => {
 const loadSuppliersFromStorage = async () => {
 try {
 const storedSuppliers = await AsyncStorage.getItem('suppliers');
 if (storedSuppliers) {
 console.log('Fornecedores carregados:', JSON.parse(storedSuppliers)); // Verifique se a imagem está incluída
 setSuppliers(JSON.parse(storedSuppliers));
 }
 } catch (error) {
 console.error('Erro ao carregar fornecedores:', error);
 } finally {
 setIsLoading(false);
 }
 };

 loadSuppliersFromStorage();
 }, []);

 // Salvar a lista de fornecedores no armazenamento
 const saveSuppliersToStorage = async (updatedSuppliers) => {
 try {
 await AsyncStorage.setItem('suppliers', JSON.stringify(updatedSuppliers));
 } catch (error) {
 console.error('Erro ao salvar fornecedores:', error);
 }
 };

 // Adicionar um fornecedor e salvar no armazenamento
 const addSupplier = (supplier) => {
 const updatedSuppliers = [
 ...suppliers,
 { ...supplier, id: String(suppliers.length + 1) },
 ];
 console.log('Fornecedores atualizados:', updatedSuppliers); // Verifique se a imagem está incluída
 setSuppliers(updatedSuppliers);
 saveSuppliersToStorage(updatedSuppliers);
 };

 const updateSupplier = (updatedSupplier) => {
 const updatedSuppliers = suppliers.map((supplier) =>
 supplier.id === updatedSupplier.id ? updatedSupplier : supplier
 );
 setSuppliers(updatedSuppliers);
 saveSuppliersToStorage(updatedSuppliers);
 };

 return (
 <SupplierContext.Provider value={{ suppliers, addSupplier, updateSupplier, isLoading }}>
 {children}
 </SupplierContext.Provider>
 );
};


// const clearStorage = async () => {
// try {
// await AsyncStorage.removeItem('suppliers');
// console.log('Lista de fornecedores deletada com sucesso!');
// } catch (error) {
// console.error('Erro ao deletar fornecedores:', error);
// }
// };

// // Chame a função ao iniciar o aplicativo para testar
// clearStorage();
```

app/src/screens/CadastroForm.js

```
import React, { useState, useContext } from 'react';
import { View, Text, TextInput, Button, StyleSheet } from 'react-native';
import { useRouter } from 'expo-router';
import * as ImagePicker from 'expo-image-picker';
import { SupplierContext } from '../context/SupplierContext';


const CadastroForm = () => {
 const { addSupplier } = useContext(SupplierContext);

 const router = useRouter();
 const [name, setName] = useState('');
 const [address, setAddress] = useState('');
 const [contact, setContact] = useState('');
 const [category, setCategory] = useState('');
 const [image, setImage] = useState(null);
 const [message, setMessage] = useState('');
 const [messageType, setMessageType] = useState(''); // 'error' ou 'success'

 const pickImage = async () => {
 let result = await ImagePicker.launchImageLibraryAsync({
 mediaTypes: ImagePicker.MediaTypeOptions.Images,
 allowsEditing: true,
 quality: 0.3, // Reduz a qualidade para economizar espaço
 });

 console.log('Resultado do ImagePicker:', result); // Verifique a estrutura do retorno

 if (!result.canceled && result.assets && result.assets.length > 0) {
 const selectedImage = result.assets[0].uri;
 console.log('Imagem selecionada:', selectedImage); // Certifique-se de que o URI é válido
 setImage(selectedImage);
 } else {
 console.log('Nenhuma imagem selecionada ou operação cancelada.');
 }

 };

 const handleSave = () => {
 if (!name || !address || !contact || !category) {
 setMessage('Todos os campos são obrigatórios!');
 setMessageType('error');
 return;
 }

 const supplier = { name, address, contact, category, image };
 console.log('Fornecedor a ser salvo:', supplier); // Verifique se a imagem está incluída
 addSupplier(supplier); // Adiciona o fornecedor à lista global
 setMessage('Fornecedor cadastrado com sucesso!');
 setMessageType('success');

 setTimeout(() => {
 router.push('/'); // Redireciona para a tela inicial
 }, 2000);
 };

 return (
 <View style={styles.container}>
 {message ? (
 <Text style={[styles.message, messageType === 'error' ? styles.error : styles.success]}>
 {message}
 </Text>
 ) : null}
 <Text style={styles.label}>Nome</Text>
 <TextInput style={styles.input} value={name} onChangeText={setName} placeholder="Nome do fornecedor" />
 <Text style={styles.label}>Endereço</Text>
 <TextInput style={styles.input} value={address} onChangeText={setAddress} placeholder="Endereço" />
 <Text style={styles.label}>Contato</Text>
 <TextInput style={styles.input} value={contact} onChangeText={setContact} placeholder="Contato" />
 <Text style={styles.label}>Categoria</Text>
 <TextInput style={styles.input} value={category} onChangeText={setCategory} placeholder="Categoria" />
 <Button title="Escolher Imagem" onPress={pickImage} />
 {image && <Text style={styles.imageText}>Imagem selecionada!</Text>}
 <Button title="Salvar" onPress={handleSave} />
 <Button title="Voltar" onPress={() => router.push('/')} />
 </View>
 );
};

const styles = StyleSheet.create({
 container: { padding: 20 },
 label: { fontSize: 16, marginTop: 10 },
 input: { borderWidth: 1, borderColor: 'gray', borderRadius: 5, padding: 10, marginTop: 5 },
 imageText: { marginTop: 10, color: 'green' },
 error: { color: 'red' },
 success: { color: 'green' },
});

export default CadastroForm;
```

app/src/screens/EditForm.js

```
import React, { useState, useContext, useEffect } from 'react';
import { View, Text, TextInput, Button, StyleSheet } from 'react-native';
import { useRouter, useLocalSearchParams } from 'expo-router';
import * as ImagePicker from 'expo-image-picker';
import { SupplierContext } from '../context/SupplierContext';

const EditForm = () => {
 const { suppliers, updateSupplier } = useContext(SupplierContext);
 const router = useRouter();
 const { id } = useLocalSearchParams();

 const [name, setName] = useState('');
 const [address, setAddress] = useState('');
 const [contact, setContact] = useState('');
 const [category, setCategory] = useState('');
 const [image, setImage] = useState(null);
 const [message, setMessage] = useState('');
 const [messageType, setMessageType] = useState(''); // 'error' ou 'success'

 useEffect(() => {
 const supplier = suppliers.find((supplier) => supplier.id === id);

 if (supplier) {
 setName(supplier.name);
 setAddress(supplier.address);
 setContact(supplier.contact);
 setCategory(supplier.category);
 setImage(supplier.image);
 }
 }, [id, suppliers]);

 const pickImage = async () => {
 let result = await ImagePicker.launchImageLibraryAsync({
 mediaTypes: ImagePicker.MediaTypeOptions.Images,
 allowsEditing: true,
 quality: 0.3, // Reduz a qualidade para economizar espaço
 });

 console.log('Resultado do ImagePicker:', result); // Verifique a estrutura do retorno

 if (!result.canceled && result.assets && result.assets.length > 0) {
 const selectedImage = result.assets[0].uri;
 console.log('Imagem selecionada:', selectedImage); // Certifique-se de que o URI é válido
 setImage(selectedImage);
 } else {
 console.log('Nenhuma imagem selecionada ou operação cancelada.');
 }

 };

 const handleSave = () => {
 if (!name || !address || !contact || !category) {
 setMessage('Todos os campos são obrigatórios!');
 setMessageType('error');
 return;
 }

 const updatedSupplier = { id, name, address, contact, category, image };
 console.log('Fornecedor a ser salvo:', updatedSupplier); // Verifique se a imagem está incluída
 updateSupplier(updatedSupplier); // Atualiza o fornecedor na lista global
 setMessage('Fornecedor cadastrado com sucesso!');
 setMessageType('success');

 setTimeout(() => {
 router.push('/'); // Redireciona para a tela inicial
 }, 2000);
 };

 return (
 <View style={styles.container}>
 {message ? (
 <Text style={[styles.message, messageType === 'error' ? styles.error : styles.success]}>
 {message}
 </Text>
 ) : null}
 <Text style={styles.label}>id: {id}</Text>
 <Text style={styles.label}>Nome</Text>
 <TextInput style={styles.input} value={name} onChangeText={setName} placeholder="Nome do fornecedor" />
 <Text style={styles.label}>Endereço</Text>
 <TextInput style={styles.input} value={address} onChangeText={setAddress} placeholder="Endereço" />
 <Text style={styles.label}>Contato</Text>
 <TextInput style={styles.input} value={contact} onChangeText={setContact} placeholder="Contato" />
 <Text style={styles.label}>Categoria</Text>
 <TextInput style={styles.input} value={category} onChangeText={setCategory} placeholder="Categoria" />
 <Button title="Escolher Imagem" onPress={pickImage} />
 {image && <Text style={styles.imageText}>Imagem selecionada!</Text>}
 <Button title="Salvar" onPress={handleSave} />
 <Button title="Voltar" onPress={() => router.push('/')} />
 </View>
 );
};

const styles = StyleSheet.create({
 container: { padding: 20 },
 label: { fontSize: 16, marginTop: 10 },
 input: { borderWidth: 1, borderColor: 'gray', borderRadius: 5, padding: 10, marginTop: 5 },
 imageText: { marginTop: 10, color: 'green' },
 error: { color: 'red' },
 success: { color: 'green' },
});

export default EditForm;
```

app/src/screens/SupplierList.js

```
import React, { useContext, useEffect } from 'react';
import { View, Text, Button, FlatList, Image, StyleSheet, ActivityIndicator } from 'react-native';
import { useRouter } from 'expo-router';
import { SupplierContext } from '../context/SupplierContext';

const SupplierList = () => {
 const router = useRouter();
 const { suppliers, isLoading } = useContext(SupplierContext);

 // Força a atualização da lista após voltar do cadastro
 useEffect(() => {
 console.log('Fornecedores na lista:', suppliers); // Verifique se a imagem está incluída
 console.log('Tela SupplierList carregada ou atualizada');
 }, [suppliers]);

 if (isLoading) {
 return (
 <View style={styles.loadingContainer}>
 <ActivityIndicator size="large" color="blue" />
 <Text>Carregando fornecedores...</Text>
 </View>
 );
 }

 return (
 <View style={styles.container}>
 {suppliers.length === 0 ? (
 <Text style={styles.emptyText}>Nenhum fornecedor cadastrado.</Text>
 ) : (
 <FlatList
 data={suppliers}
 keyExtractor={(item) => item.id}
 renderItem={({ item }) => (
 <View style={styles.card}>
 {item.image ? (
 <Image source={{ uri: item.image }} style={styles.image} />
 ) : (
 <View style={styles.placeholder} />
 )}
 <Text style={styles.text}>Id: {item.id}</Text>
 <Text style={styles.text}>Nome: {item.name}</Text>
 <Text style={styles.text}>Endereço: {item.address}</Text>
 <Text style={styles.text}>Contato: {item.contact}</Text>
 <Text style={styles.text}>Categoria: {item.category}</Text>
 <Button style={styles.buttonEdit} title="Editar" onPress={() => router.push(`/edit/${item.id}`)} />
 </View>
 )}
 />
 )}
 <Button title="Cadastrar Fornecedor" onPress={() => router.push('/cadastro')} />
 </View>
 );
};

const styles = StyleSheet.create({
 container: { flex: 1, padding: 20 },
 loadingContainer: { flex: 1, justifyContent: 'center', alignItems: 'center' },
 card: { padding: 10, marginBottom: 10, borderWidth: 1, borderColor: 'gray', borderRadius: 5 },
 image: { width: 50, height: 50 },
 placeholder: { width: 50, height: 50, backgroundColor: 'gray' },
 text: { marginTop: 5, fontSize: 16 },
 emptyText: { fontSize: 18, textAlign: 'center', marginTop: 20, color: 'gray' },
 buttonEdit: {backgroundColor: 'red'},
});

export default SupplierList;
```