Missão Prática | Nível 1 | Mundo 4

Eduardo Eugênio de Araujo e Silva Domingues 202302579043

1172 POLO CASA CAIADA – OLINDA – PE
Vamos Criar um App – 2024.2

https://github.com/eduardoduud/Missao-Pratica-Nivel-1-Mundo-4

Index

app > index.tsx > ...

```tsx
import React from "react";
import AppNavigator from "../navigation/appNavigator";

Codeium: Refactor | Explain | Generate JSDoc | X
const App: React.FC = () => {
  return <AppNavigator />;
};

export default App;

```

Renderiza o componente AppNavigator, que por sua vez renderiza o resto das telas

AppNavigator

```
7   const Tab = createBottomTabNavigator();
8
9   type IoniconsName = keyof typeof Ionicons.glyphMap;
10
    Codeium: Refactor | Explain | Generate JSDoc | ×
11  const AppNavigator: React.FC = () => {
12    return (
13      <Tab.Navigator
14        screenOptions={({ route }) => ({
            Codeium: Refactor | Explain | Generate JSDoc | ×
15          tabBarIcon: ({ color, size }) => {
16            let iconName: IoniconsName = "ellipse";
17
18            if (route.name === "Cadastrar Fornecedor") {
19              iconName = "add-circle-outline";
20            } else if (route.name === "Lista de Fornecedores") {
21              iconName = "list-outline";
22            } else if (route.name === "Upload de Imagem") {
23              iconName = "image-outline";
24            }
25
26            return <Ionicons name={iconName} size={size} color={color} />;
27          },
28          tabBarActiveTintColor: "blue",
29          tabBarInactiveTintColor: "gray",
30          headerShown: false,
31        })}
32      >
33        <Tab.Screen
34          name="Lista de Fornecedores"
35          component={SupplierList}
36          options={{ headerTitle: "Lista de Fornecedores" }}
37        />
38        <Tab.Screen
39          name="Cadastrar Fornecedor"
40          component={SupplierForm}
41          options={{ headerTitle: "Cadastro de Fornecedores" }}
42        />
43      </Tab.Navigator>
44    );
45  };
46
47  export default AppNavigator;
48
```

Navegação: O AppNavigator usa createBottomTabNavigator para criar uma navegação em abas entre as telas de "Cadastrar Fornecedor" e "Lista de Fornecedores".

Ícones: Os ícones são gerados usando Ionicons, que mudam de acordo com a aba selecionada.

Estilos da Aba: As cores dos ícones das abas mudam entre azul (ativo) e cinza (inativo), melhorando a experiência do usuário.

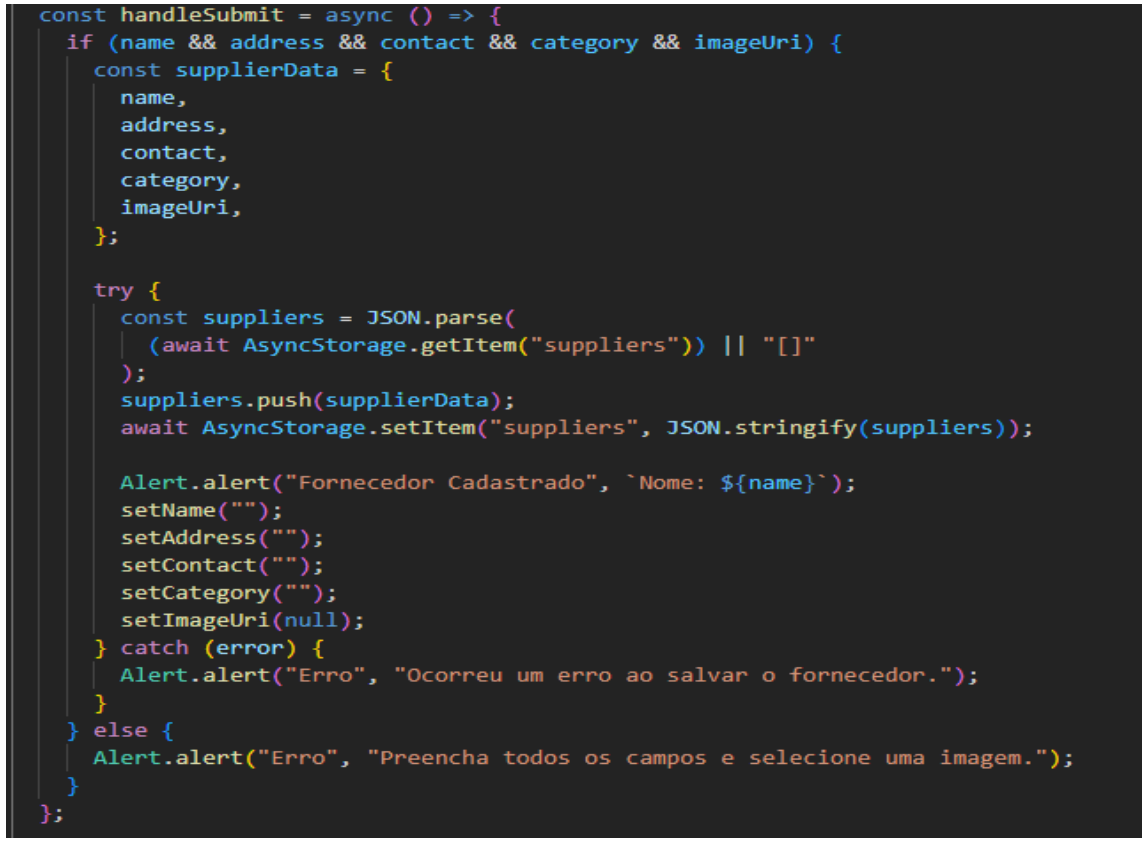
SupplierForm

```
  const handleImagePicker = async () => {
    const permissionResult =
      await ImagePicker.requestMediaLibraryPermissionsAsync();

    if (permissionResult.granted === false) {
      Alert.alert(
        "Permissão necessária",
        "Você precisa permitir o acesso à galeria."
      );
      return;
    }

    const result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.Images,
      allowsEditing: true,
      quality: 1,
    });

    if (!result.canceled && result.assets && result.assets.length > 0) {
      setImageUri(result.assets[0].uri);
    }
  };
```



```
const handleSubmit = async () => {
  if (name && address && contact && category && imageUri) {
    const supplierData = {
      name,
      address,
      contact,
      category,
      imageUri,
    };

    try {
      const suppliers = JSON.parse(
        (await AsyncStorage.getItem("suppliers")) || "[]"
      );
      suppliers.push(supplierData);
      await AsyncStorage.setItem("suppliers", JSON.stringify(suppliers));

      Alert.alert("Fornecedor Cadastrado", `Nome: ${name}`);
      setName("");
      setAddress("");
      setContact("");
      setCategory("");
      setImageUri(null);
    } catch (error) {
      Alert.alert("Erro", "Ocorreu um erro ao salvar o fornecedor.");
    }
  } else {
    Alert.alert("Erro", "Preencha todos os campos e selecione uma imagem.");
  }
};
```

Estados: Os estados name, address, contact, category e imageUri armazenam as informações do fornecedor.

Image Picker: O método handleImagePicker solicita permissão para acessar a galeria e permite que o usuário escolha uma imagem.

Submissão: O método handleSubmit valida se todos os campos estão preenchidos e salva as informações do fornecedor no AsyncStorage.

SupplierList

```
const SupplierList: React.FC = () => {
  const [suppliers, setSuppliers] = useState<Supplier[]>([]);

  Codeium: Refactor | Explain | Generate JSDoc | ×
  const loadSuppliers = async () => {
    try {
      const suppliersData = await AsyncStorage.getItem("suppliers");
      if (suppliersData) {
        setSuppliers(JSON.parse(suppliersData));
      }
    } catch (error) {
      Alert.alert("Erro", "Não foi possível carregar os fornecedores.");
    }
  };

  useEffect(() => {
    loadSuppliers();
  }, []);

  Codeium: Refactor | Explain | Generate JSDoc | ×
  const renderSupplierItem = ({ item }: { item: Supplier }) => (
    <View style={styles.supplierCard}>
      <Image source={{ uri: item.imageUri }} style={styles.supplierImage} />
      <View style={styles.supplierInfo}>
        <Text style={styles.supplierName}>{item.name}</Text>
        <Text style={styles.supplierDetail}>Endereço: {item.address}</Text>
        <Text style={styles.supplierDetail}>Contato: {item.contact}</Text>
        <Text style={styles.supplierDetail}>Categoria: {item.category}</Text>
      </View>
    </View>
  );
```

Carregar Fornecedores: O método loadSuppliers é chamado quando a tela é carregada. Ele busca os dados salvos no AsyncStorage e atualiza o estado.

Renderização: A lista de fornecedores é renderizada usando FlatList, que exibe cada fornecedor em um cartão com as informações correspondentes.

Atualização: Um botão de atualização permite que o usuário recarregue a lista manualmente.