RPG0023 - Vamos criar um App

Alessandro Sendi Shigematsu Matrícula 202208809812

Polo Parque Andorinhas
2024.1 Desenvolvimento Full Stack – 1o Semestre Letivo

Objetivo da Prática

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

1º Procedimento | Micro Atividades

As atividades estão no diretório GIT dentro da pasta micro_atividades.

2º Procedimento | Componentes Cats

As atividades dos componentes Cats estão no diretório GIT dentro da pasta components.

3º Procedimento | Missão Prática Vamos Criar um App!

As atividades estão no diretório GIT dentro da pasta paginass

1. Código fontes

1.1.App.js

```
import React, { useState } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { Provider as PaperProvider } from 'react-native-paper';
import CadastroFornecedorScreen from './paginas/Cadastro.js';
import ListaFornecedoresScreen from './paginas/Lista.js';
```

```
import EdicaoFornecedorScreen from './paginas/edicao.js';
import { View,Text } from 'react-native';
import HomeScreen from './paginas/home.js';

const Stack = createStackNavigator();

const App = () => {
  const [fornecedores, setFornecedores] = useState([]);

  return (

    <PaperProvider>
      <NavigationContainer>
        <Stack.Navigator initialRouteName="Home">
        <Stack.Screen name="Home" component={HomeScreen} />
          <Stack.Screen name="CadastroFornecedor">
            {props      =>      <CadastroFornecedorScreen      {...props}
setFornecedores={setFornecedores} />}
          </Stack.Screen>
          <Stack.Screen name="EdicaoFornecedor">
            {props => <EdicaoFornecedorScreen {...props}  />}
          </Stack.Screen>

          <Stack.Screen name="ListaFornecedores">
            {props      =>      <ListaFornecedoresScreen      {...props}
fornecedores={fornecedores} setFornecedores={setFornecedores} />}
          </Stack.Screen>

        </Stack.Navigator>
      </NavigationContainer>
    </PaperProvider>
  );
};

export default App;
```

1.2.Home.js

```
import React from 'react';
```

```
import { View, StyleSheet } from 'react-native';
import { Button } from 'react-native-paper';

const HomeScreen = ({ navigation }) => {
  return (
    <View style={styles.container}>
      <Button            mode="contained"            style={styles.button}
labelStyle={styles.buttonLabel}              onPress={()               =>
navigation.navigate('CadastroFornecedor')} >
        Cadastrar Fornecedor
      </Button>

      <Button               mode="contained"        style={styles.button}
labelStyle={styles.buttonLabel}              onPress={()               =>
navigation.navigate('ListaFornecedores')}>
        Listar Fornecedores
      </Button>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  button: {
    marginBottom: 20,
    width: '70%',
    paddingVertical: 10,
    borderRadius: 10,
  },
  buttonLabel: {
    fontSize: 16,
  },
});

export default HomeScreen;
```

1.3.Cadastro dos Fornecedores

```
import React, { useState } from 'react';
```

```
import { View,StyleSheet,Alert, Image, TouchableOpacity,Text,ScrollView  }
from 'react-native';
import { Button, TextInput,Avatar } from 'react-native-paper';
import * as ImagePicker from 'expo-image-picker';

const CadastroFornecedorScreen = ({ navigation, setFornecedores }) => {
  const [nome, setNome] = useState('');
  const [endereco, setEndereco] = useState('');
  const [contato, setContato] = useState('');
  const [categorias, setCategorias] = useState('');
  const [imagem, setImagem] = useState(null);

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

    if (permissionResult.granted === false) {
      Alert.alert('Permissão  negada',  'A  permissão  para  acessar  a
biblioteca de mídia é necessária.');
      return;
    }

    const pickerResult = await ImagePicker.launchImageLibraryAsync();

    if (pickerResult.cancelled === true) {
      return;
    }

    Alert.alert('Sucesso', 'Fornecedor imagem com sucesso!');
    setImagem( pickerResult.assets[0].uri );
  };

  const cadastrarFornecedor = () => {
    if (nome && endereco && contato && categorias && imagem) {
      setFornecedores(prevFornecedores => [
        ...prevFornecedores,
        { nome, endereco, contato, categorias, imagem }
      ]);
      setNome('');
      setEndereco('');
      setContato('');
      setCategorias('');
      setImagem('');

      navigation.goBack();
      Alert.alert('Sucesso', 'Fornecedor cadastrado com sucesso!');
    } else {
      Alert.alert( "Erro","Favor fornecer todos os dados!");
```

```
    }
  };

  return (
    <ScrollView >
      <View style={styles.container}>
        {imagem && <Avatar.Image source={{ uri: imagem }} size={150}
style={styles.imagem} />}
        <TouchableOpacity onPress={cadastrarImagem}>
          <View style={styles.botaoImagem}>
            <Text style={styles.textoBotaoImagem}>Escolher Imagem</Text>
          </View>
        </TouchableOpacity>

        <TextInput
          label="Nome do Fornecedor"
          value={nome}
          onChangeText={text => setNome(text)}
          style={styles.input}
        />
        <TextInput
          label="Endereço"
          value={endereco}
          onChangeText={text => setEndereco(text)}
          style={styles.input}
        />
        <TextInput
          label="Contato"
          value={contato}
          onChangeText={text => setContato(text)}
          style={styles.input}
        />
        <TextInput
          label="Categorias de Produtos Fornecidos"
          value={categorias}
          onChangeText={text => setCategorias(text)}
          style={styles.input}
        />

        <View style={styles.alinharBotoes}>
          <Button          mode="contained"          style={styles.button}
labelStyle={styles.buttonLabel} onPress={cadastrarFornecedor}>
            Cadastrar
          </Button>

          {/*      <Button      mode="contained"      style={styles.button}
labelStyle={styles.buttonLabel} onPress={navigation.goBack}>
            Voltar
```

```
          </Button> */}
        </View>
      </View>
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    justifyContent: 'center',
    alignItems: 'center',

  },
  input: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,
    marginBottom: 10,
    paddingHorizontal: 10,
    width: '100%',
  },
  button: {
    marginBottom: 20,
    width: '70%',
    paddingVertical: 10,
    borderRadius: 10,

  },
  buttonLabel: {
    fontSize: 16,
   },

  imagem: {
    alignself:"center",
    marginBottom: 20,
  },
  botaoImagem: {
    backgroundColor: '#DDDDDD',
    padding: 10,
    marginBottom: 10,
  },
  textoBotaoImagem: {
    textAlign: 'center',
  },
  alinharBotoes:{
    paddingTop: 10,
```

```
    alignItems: 'center',
  }
});


export default CadastroFornecedorScreen;
```

1.4.Listagem dos Forncedores

```
import React, { useState } from 'react';
import { View, StyleSheet, FlatList , Modal,TouchableOpacity,Text } from
'react-native';
import { List, Avatar, Divider, Title, Caption,TextInput } from 'react-
native-paper';

const ListaFornecedoresScreen = ({navigation,fornecedores,setFornecedores
}) => {
  const [searchQuery, setSearchQuery] = useState('');
  const       [filteredFornecedores,       setFilteredFornecedores]       =
useState(fornecedores);
  const [filterCriteria, setFilterCriteria] = useState('nome');
  const [isModalVisible, setIsModalVisible] = useState(false);

  const handleSearch = () => {
    let filtered = fornecedores;
    if (filterCriteria === 'nome') {
      filtered = fornecedores.filter(fornecedor =>
        fornecedor.nome.toLowerCase().includes(searchQuery.toLowerCase())
      );
    } else if (filterCriteria === 'categorias') {
      filtered = fornecedores.filter(fornecedor =>

fornecedor.categorias.toLowerCase().includes(searchQuery.toLowerCase())
      );
    } else if (filterCriteria === 'contato') {
      filtered = fornecedores.filter(fornecedor =>

fornecedor.contato.toLowerCase().includes(searchQuery.toLowerCase())
      );
    } else if (filterCriteria === 'endereco') {
      filtered = fornecedores.filter(fornecedor =>

fornecedor.endereco.toLowerCase().includes(searchQuery.toLowerCase())
      );
    }
    setFilteredFornecedores(filtered);
  };
```

```
  const handleEditFornecedor = (fornecedor) => {
    navigation.navigate('EdicaoFornecedor',                                 {
fornecedor,fornecedores,setFornecedores });
  };

  const handleSelectCriteria = (criteria) => {
    setFilterCriteria(criteria);
    setIsModalVisible(false);
  };

  const renderItem = ({ item }) => (
    <>
      <List.Item
        title={item.nome}
        description={
          `Endereço:                          ${item.endereco}\nContato:
${item.contato}\nCategoria: ${item.categorias}`
        }
        descriptionNumberOfLines={3}
        left={() => <Avatar.Image source={{ uri: item.imagem }} size={60}
/>}
        right={()=>        <TouchableOpacity        onPress={()        =>
handleEditFornecedor(item)}>
                <Text style={styles.editButton}>Editar</Text>
              </TouchableOpacity>}
      />
      <Divider />
    </>
  );

  return (
    <View style={styles.container}>
      <Title style={styles.title}>Lista de Fornecedores</Title>

      <TextInput
        style={styles.searchInput}
        placeholder={`Pesquisar por ${filterCriteria}...`}
        value={searchQuery}
        onChangeText={setSearchQuery}
        onSubmitEditing={handleSearch}
      />

      <TouchableOpacity     onPress={()     =>     setIsModalVisible(true)}
style={styles.criteriaButton}>
        <Text style={styles.buttonText}>Critério: {filterCriteria}</Text>
      </TouchableOpacity>
```

```
      <Modal
        visible={isModalVisible}
        animationType="slide"
        transparent={true}
        onRequestClose={() => setIsModalVisible(false)}
      >
        <View style={styles.modalContainer}>
          <View style={styles.modalContent}>
            <TouchableOpacity onPress={() => handleSelectCriteria('nome')}
style={styles.modalItem}>
              <Text style={styles.modalText}>Nome</Text>
            </TouchableOpacity>
            <TouchableOpacity                onPress={()                =>
handleSelectCriteria('categorias')} style={styles.modalItem}>
              <Text style={styles.modalText}>Categoria</Text>
            </TouchableOpacity>
            <TouchableOpacity                onPress={()                =>
handleSelectCriteria('contato')} style={styles.modalItem}>
              <Text style={styles.modalText}>Contato</Text>
            </TouchableOpacity>
            <TouchableOpacity                onPress={()                =>
handleSelectCriteria('endereco')} style={styles.modalItem}>
              <Text style={styles.modalText}>Endereco</Text>
            </TouchableOpacity>
          </View>
        </View>
      </Modal>

      {filteredFornecedores && filteredFornecedores.length > 0 ? (
        <FlatList
          data={fornecedores}
          renderItem={renderItem}
          keyExtractor={(item, index) => index.toString()}
        />
      ) : (
        <Caption          style={styles.caption}>Nenhum          fornecedor
cadastrado</Caption>
      )}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
   },
  title: {
```

```
    marginBottom: 20,
    fontWeight: 'bold',
    fontSize: 20,
  },
  caption: {
    marginTop: 20,
    fontSize: 16,
  },
  criteriaButton: {
    marginBottom: 10,
    padding: 10,
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    alignItems: 'center',
  },
  buttonText: {
    fontSize: 16,
  },
  searchInput: {
    marginBottom: 10,
    padding: 10,
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
  },
  modalContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: 'rgba(0, 0, 0, 0.5)',
  },
  modalContent: {
    backgroundColor: 'white',
    padding: 20,
    borderRadius: 10,
    width: '80%',
  },
  modalItem: {
    padding: 10,
    borderBottomWidth: 1,
    borderBottomColor: '#ccc',
  },
  modalText: {
    fontSize: 16,
  },
});
```

```
export default ListaFornecedoresScreen;
```

1.5.Edição dos Fornecedores

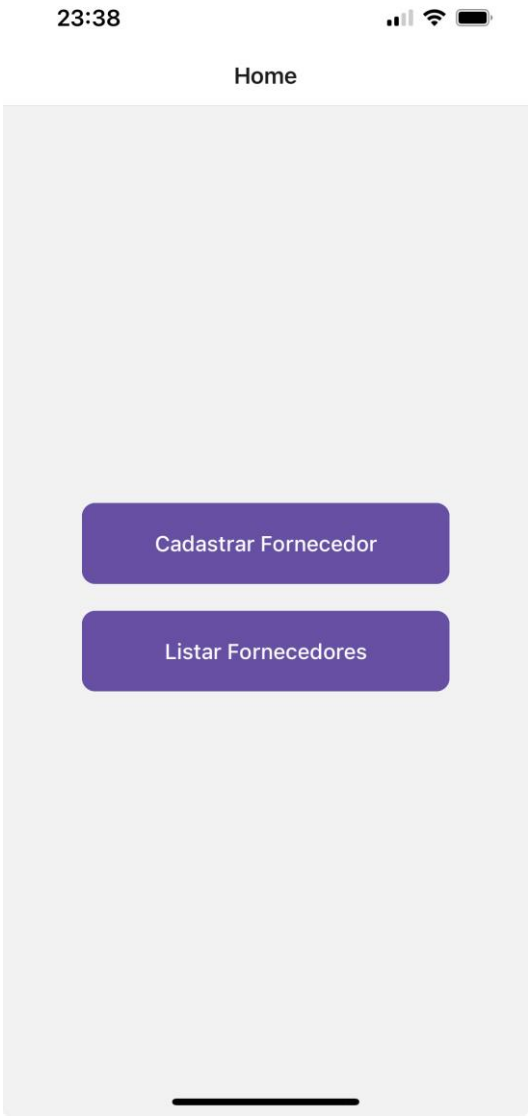
```
import React, { useState } from 'react';
import { View, Alert, TouchableOpacity,StyleSheet,Text } from 'react-
native';
import { Avatar ,TextInput, Button} from 'react-native-paper';
import * as ImagePicker from 'expo-image-picker';

const EdicaoFornecedorScreen = ({ route, navigation }) => {
  const { fornecedor,setFornecedores,fornecedores } = route.params;
  const [nome, setNome] = useState(fornecedor.nome);
  const [endereco, setEndereco] = useState(fornecedor.endereco);
  const [contato, setContato] = useState(fornecedor.contato);
  const [categorias, setCategorias] = useState(fornecedor.categorias);
  const [imagem, setImagem] = useState(fornecedor.imagem);

  const handleSubmit = () => {
    const fornecedorAtualizado = {
        ...fornecedor,
        nome,
        endereco,
        contato,
        categorias,
        imagem,
      };

      const fornecedoresAtualizados = fornecedores.map((item) =>
         item.id === fornecedorAtualizado.id ? fornecedorAtualizado : item
      );

      setFornecedores(fornecedoresAtualizados);
      Alert.alert('Sucesso', 'Fornecedor atualizado com sucesso!');
      navigation.goBack();
  };

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

    if (permissionResult.granted === false) {
      Alert.alert('Permissão negada', 'A permissão para acessar a
biblioteca de mídia é necessária.');
      return;
    }
```

```
    const pickerResult = await ImagePicker.launchImageLibraryAsync();

    if (pickerResult.cancelled === true) {
      return;
    }

    Alert.alert('Sucesso', 'Fornecedor imagem com sucesso!');
    setImagem( pickerResult.assets[0].uri );
  };


  return (
    <View style={styles.container}>
      {imagem  &&  <Avatar.Image  source={{  uri:  imagem  }}  size={150}
style={styles.imagem} />}
      <TouchableOpacity onPress={editarImagem}>
          <View style={styles.botaoImagem}>
            <Text style={styles.textoBotaoImagem}>Escolher Imagem</Text>
          </View>
        </TouchableOpacity>

      <TextInput
        style={styles.input}
        label="Nome do fornecedor"
        value={nome}
        onChangeText={setNome}
      />
      <TextInput
        style={styles.input}
        label="Endereço"
        value={endereco}
        onChangeText={setEndereco}
      />
      <TextInput
        style={styles.input}
        label="Contato"
        value={contato}
        onChangeText={setContato}
      />
      <TextInput
        style={styles.input}
        label="Categoria de produtos"
        value={categorias}
        onChangeText={setCategorias}
      />
      <Button                          mode="contained"style={styles.button}
labelStyle={styles.buttonLabel}          title="Atualizar     Fornecedor"
onPress={handleSubmit} >
```

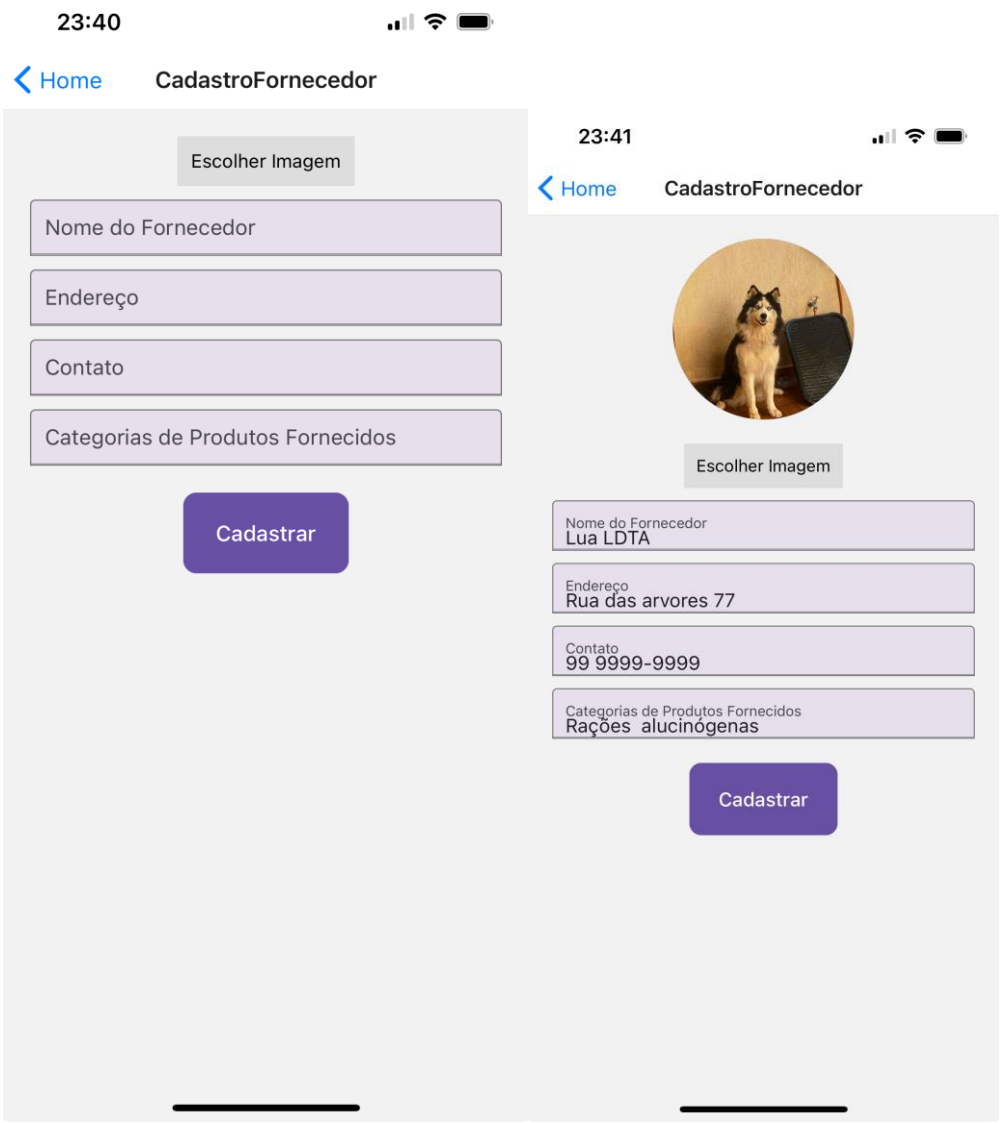
```
        Editar
      </Button>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 20,
  },
  input: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,
    marginBottom: 10,
    paddingHorizontal: 10,
    width: '100%',
  },
  imagem: {
    marginBottom: 20,
  },
  botaoImagem: {
    backgroundColor: '#DDDDDD',
    padding: 10,
    marginBottom: 10,
  },
  textoBotaoImagem: {
    textAlign: 'center',
  },
  button: {
    marginBottom: 20,
    width: '70%',
    paddingVertical: 10,
    borderRadius: 10,

  },
  buttonLabel: {
    fontSize: 16,
   },
});

export default EdicaoFornecedorScreen;
```

2. Imagem das páginas

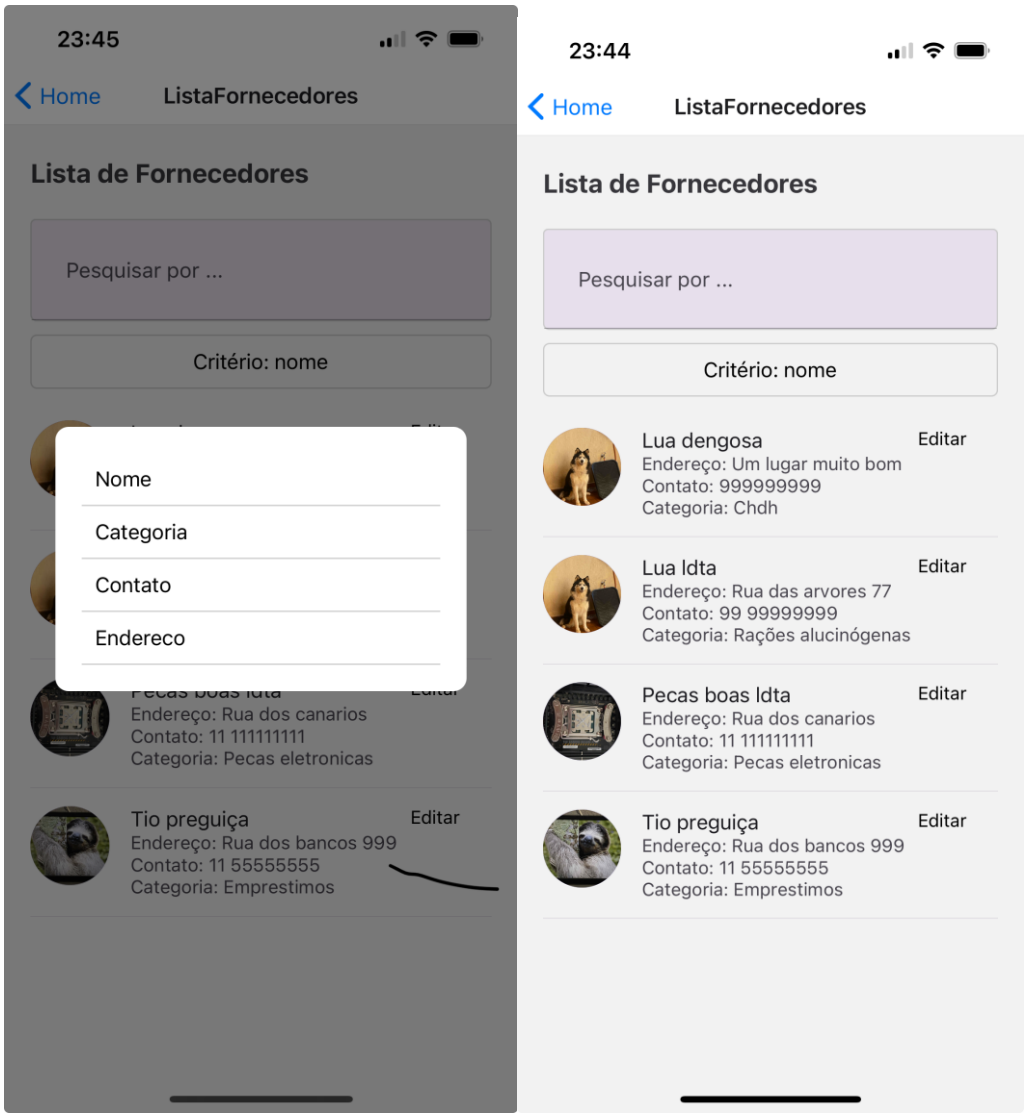
2.1.Home.js



2.2.Cadastro dos Fornecedores



2.3.Listagem dos Fornecedores



2.4.Edição dos Fornecedores

Conclusão

O React Native traz o benefício do ágil e conceituado framework React para criação de aplicativos mobiles. O interessante que possui diversas ferramentas que contribuem na rapidez e no padrão do código como o React.

Possui diversas bibliotecas que podem ser baixadas em yarn e no NPM que facilitam o desenvolvimento por parte do programador como o EXPO. Uma importante colocação é a inexistência da linguagem de marcação HTML (HyperText MarkUp Language) e apenas a sintaxe JSX utilizando o javascript puro que traz vida às páginas.

A facilidade que traz o React Native para desenvolvimento mobile é notável e deve ser considerado em caso de um projeto que o cliente queira dispositivos Androids e IOS.