Estácio	**Universidade Estácio de Sá** Desenvolvimento Full Stack Nível 1 – Vamos criar um App Turma 2022.3 – 4º Semestre
Nome:	Luan Augusto Vieira Bandeira
Repositório:	https://github.com/luanguto/nivel-1-mundo-4

Objetivos da prática

Requisitos Funcionais:

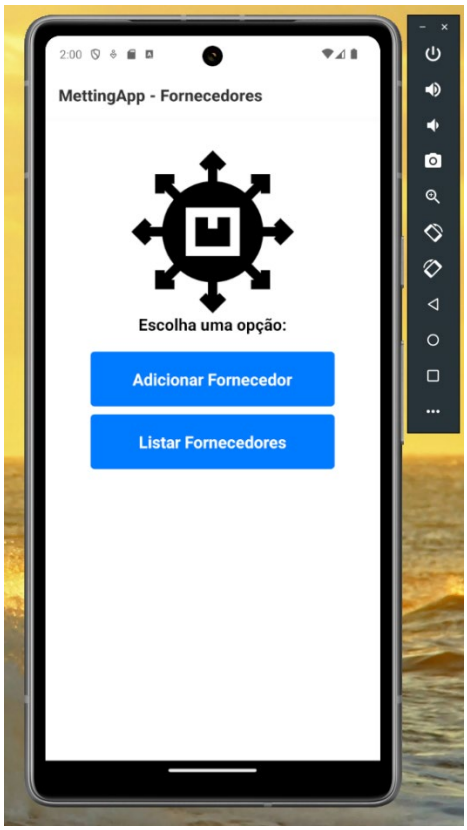
Cadastro de Fornecedores: O aplicativo deve permitir o cadastro de fornecedores, incluindo informações detalhadas, como nome, endereço, contato e categorias de produtos fornecidos. Essas informações serão exibidas utilizando componentes como <Text>, <TextInput>, e <Image>.

Listagem de Fornecedores: Deve ser possível visualizar uma lista de fornecedores cadastrados, com opções de pesquisa e filtragem com base em critérios como categoria ou localização. A lista de fornecedores será exibida utilizando componentes como <Text> e <Image>.

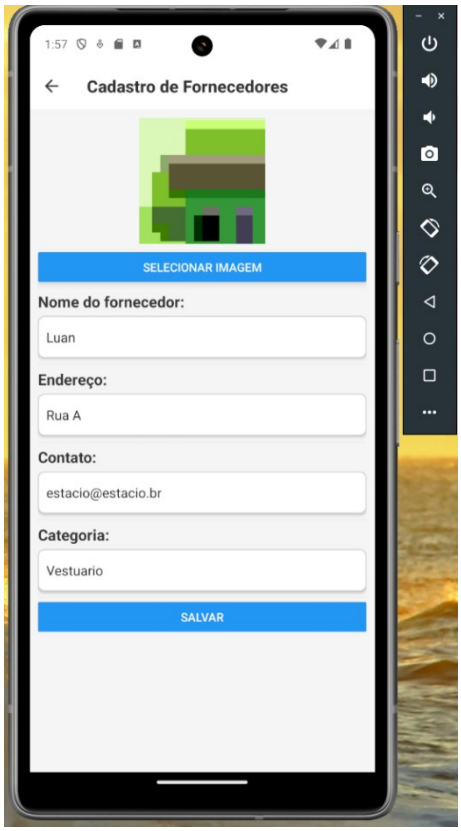
Associação de Imagens: O aplicativo deve permitir a associação de imagens aos perfis dos fornecedores. Os usuários devem poder fazer o upload de logotipos ou fotos relacionadas ao fornecedor, utilizando o componente <Image>.

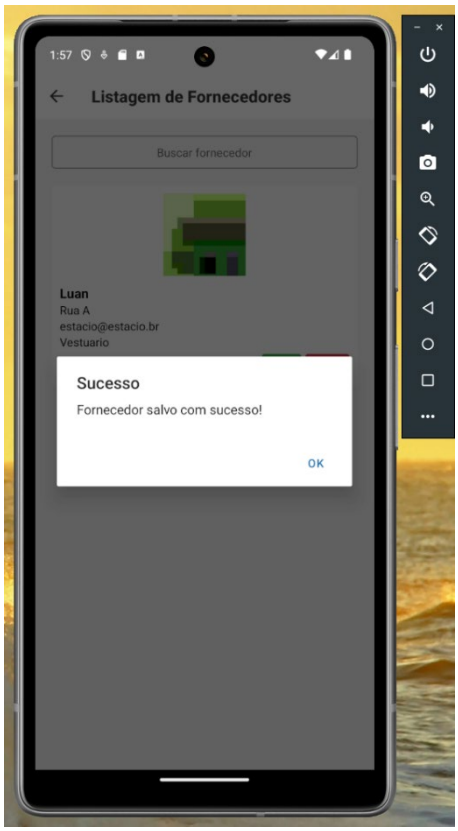
Experiência de Usuário Intuitiva: A interface do aplicativo deve ser intuitiva e fácil de usar, garantindo que os usuários possam navegar, adicionar e editar informações de forma eficiente. Isso será alcançado usando componentes como <Text>, <TextInput>, e <Image>.

Tela inicial

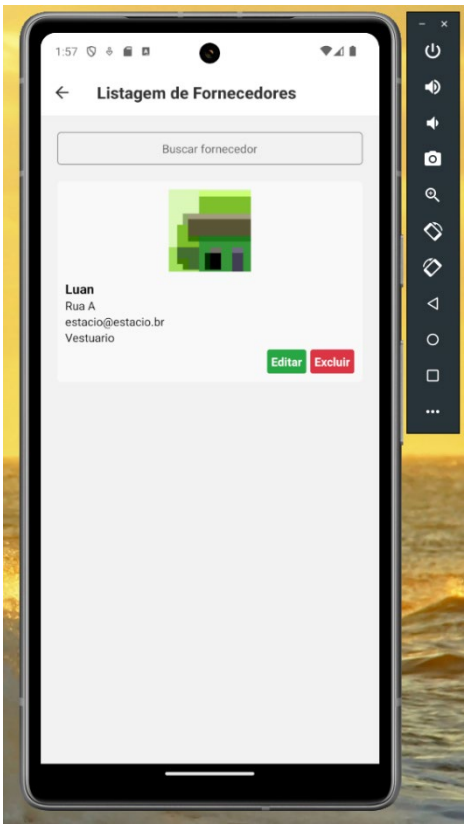


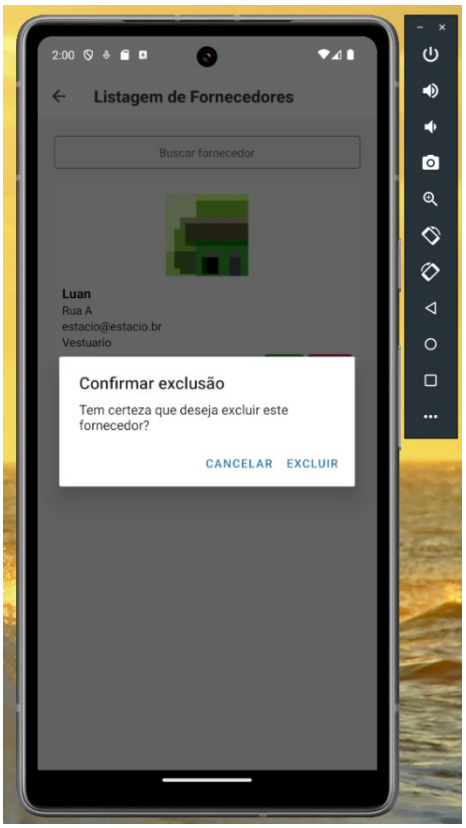
Cadastro de fornecedores

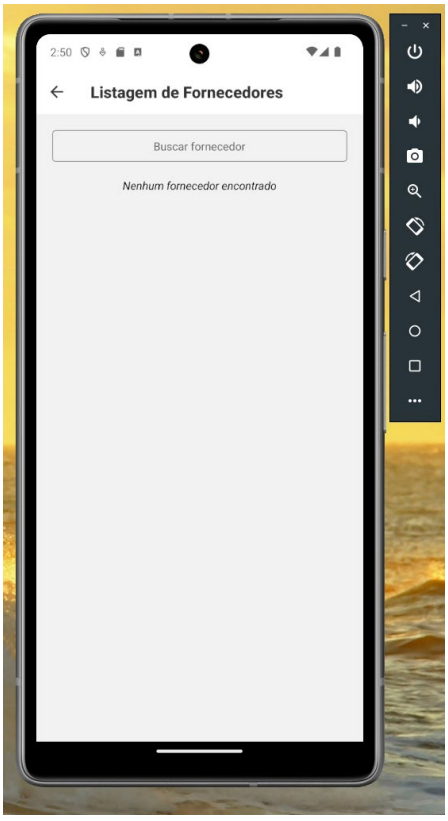




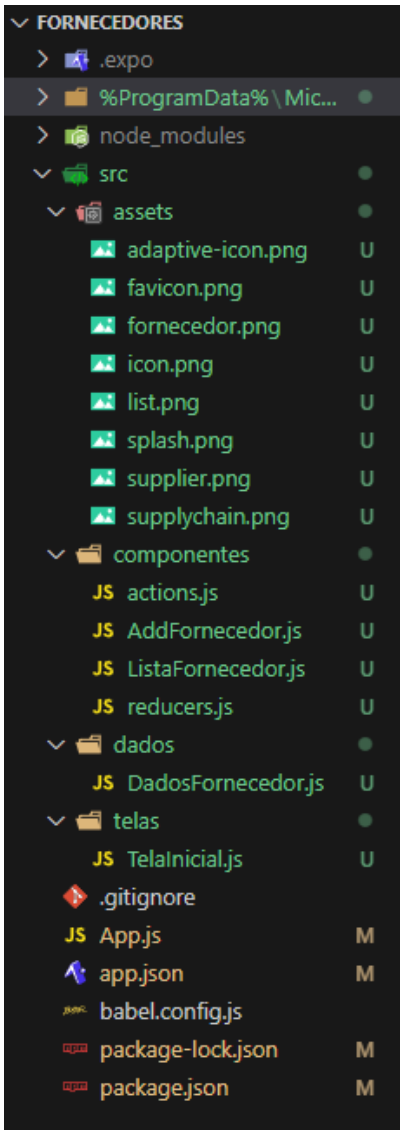
Listagem dos Fornecedores







Estrutura e códigos:



Foi criado a seguinte estrutura **SRC** na qual armazenou os **assets, componentes, dados, telas**

Actions.js

```
export const adicionarFornecedor = (fornecedor) => ({
    type: 'ADICIONAR_FORNECEDOR',
    payload: fornecedor,
  });

  export const editarFornecedor = (id, novoFornecedor) => ({
    type: 'EDITAR_FORNECEDOR',
    payload: { id, novoFornecedor },
  });

  export const excluirFornecedor = (id) => ({
    type: 'EXCLUIR_FORNECEDOR',
    payload: id,
  });
```

AddFornecedor.js

```
import React, { useState, useEffect } from "react";
import { StyleSheet, TextInput, Text, Button, View, Image, Alert,
ScrollView } from "react-native";
import { useFornecedores } from '../dados/DadosFornecedor';
import { useNavigation, useRoute } from "@react-navigation/native";
import * as ImagePicker from 'expo-image-picker';
import * as FileSystem from 'expo-file-system';
import AsyncStorage from '@react-native-async-storage/async-storage';

const Separator = () => <View style={styles.separator} />;

const FormularioFornecedor = () => {
    const route = useRoute();
    const [idEditando, setIdEditando] = useState(route.params?.idEditando
|| "");
    const [nome, setNome] = useState("");
    const [endereco, setEndereco] = useState("");
    const [contato, setContato] = useState("");
    const [categoria, setCategoria] = useState("");
    const [imagem, setImagem] = useState("");

    const { fornecedores, setFornecedores } = useFornecedores();
    const navigation = useNavigation();

    useEffect(() => {
        if (idEditando && fornecedores) {
            const fornecedorEditando = fornecedores.find(f => f.id ===
idEditando);
            if (fornecedorEditando) {
                setNome(fornecedorEditando.nome);
                setEndereco(fornecedorEditando.endereco);
                setContato(fornecedorEditando.contato);
                setCategoria(fornecedorEditando.categoria);
                setImagem(fornecedorEditando.imagem);
            }
        }
    }, [idEditando, fornecedores]);

    const handleSave = async () => {
        if (!nome || !endereco || !contato || !categoria || !imagem) {
            Alert.alert('Erro', 'Todos os campos são obrigatórios.');
            return;
        }

        let fornecedoresAtualizados;
        if (idEditando) {
            fornecedoresAtualizados = fornecedores.map(fornecedor =>
                fornecedor.id === idEditando
```

```
                    ? { ...fornecedor, nome, endereco, contato,
categoria, imagem }
                    : fornecedor
            );
        } else {
            const novoFornecedor = { id: new Date().getTime().toString(),
nome, endereco, contato, categoria, imagem };
            fornecedoresAtualizados = [...fornecedores, novoFornecedor];
        }

        try {
            await AsyncStorage.setItem('fornecedores',
JSON.stringify(fornecedoresAtualizados));
            setFornecedores(fornecedoresAtualizados);
            Alert.alert('Sucesso', 'Fornecedor salvo com sucesso!');
            setNome("");
            setEndereco("");
            setContato("");
            setCategoria("");
            setImagem("");
            setIdEditando("");
            navigation.navigate('Listagem de Fornecedores');
        } catch (error) {
            Alert.alert('Erro', 'Não foi possível salvar o fornecedor.
Por favor, tente novamente.');
        }

    };

    const selecionarImagem = async () => {
        let permissionResult = await
ImagePicker.requestMediaLibraryPermissionsAsync();
        if (!permissionResult.granted) {
            Alert.alert('Permissão necessária', 'É necessário conceder
permissão para acessar a galeria.');
            return;
        }

        let resultado = await ImagePicker.launchImageLibraryAsync({
            mediaTypes: ImagePicker.MediaTypeOptions.Images,
            allowsEditing: true,
            aspect: [4, 3],
            quality: 1,
        });

        if (resultado.cancelled) {
            Alert.alert('Seleção Cancelada', 'A seleção de imagem foi
cancelada.');
            return;
```

```
        }

        const firstAsset = resultado.assets ? resultado.assets[0] : null;
        if (!firstAsset || !firstAsset.uri) {
            Alert.alert('Erro', 'Não foi possível obter a URI da imagem
selecionada.');
            return;
        }

        try {
            let base64Image = firstAsset.uri.startsWith('file://')
                ? `${await
FileSystem.readAsStringAsync(firstAsset.uri, { encoding:
FileSystem.EncodingType.Base64 })}`
                : firstAsset.uri;
            setImagem(base64Image);
        } catch (error) {
            console.error('Erro ao converter imagem:', error);
            Alert.alert('Erro', 'Não foi possível converter a imagem para
formato adequado.');
        }
    };

    return (
        <ScrollView style={styles.container}>
            {imagem ? <Image source={{ uri: imagem }}
style={styles.imagem} /> : null}
            <Button title="Selecionar Imagem" onPress={selecionarImagem}
/>
            <Separator />
            <Text style={styles.texto}>Nome do fornecedor:</Text>
            <TextInput style={styles.input} value={nome}
onChangeText={setNome} placeholder="Digite o nome" />
            <Separator />
            <Text style={styles.texto}>Endereço:</Text>
            <TextInput style={styles.input} value={endereco}
onChangeText={setEndereco} placeholder="Digite o endereço" />
            <Separator />
            <Text style={styles.texto}>Contato:</Text>
            <TextInput style={styles.input} value={contato}
onChangeText={setContato} placeholder="Digite o contato" />
            <Separator />
            <Text style={styles.texto}>Categoria:</Text>
            <TextInput style={styles.input} value={categoria}
onChangeText={setCategoria} placeholder="Digite a categoria" />
            <Separator />
            <Button title="Salvar" onPress={handleSave} />
        </ScrollView>
    );
```

```
};

const styles = StyleSheet.create({
    imagem: {
        width: 150,
        height: 150,
        marginBottom: 10,
        alignSelf: 'center',
    },
    texto: {
        paddingBottom: 5,
        fontSize: 18,
        fontWeight: 'bold',
        color: '#333',
    },
    container: {
        flex: 1,
        padding: 10,
        backgroundColor: '#f5f5f5',
    },
    input: {
        borderWidth: 1,
        borderColor: '#ccc',
        borderRadius: 8,
        padding: 10,
        marginBottom: 2,
        backgroundColor: '#fff',
        fontSize: 16,
        shadowColor: "#000",
        shadowOffset: {
            width: 0,
            height: 2,
        },
        shadowOpacity: 0.1,
        shadowRadius: 1.41,
        elevation: 2,
    },
    separator: {
        marginVertical: 5,
        borderBottomColor: '#eee',
        borderBottomWidth: 1,
    },
});

export default FormularioFornecedor;
```

ListaFornecedor.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, FlatList, StyleSheet, Image,
TouchableOpacity, Alert } from 'react-native';
import { useFornecedores } from '../dados/DadosFornecedor';
import { useNavigation } from '@react-navigation/native';

const ListaFornecedores = () => {
    const { fornecedores, setFornecedores } = useFornecedores();
    const [filtro, setFiltro] = useState('');
    const [carregando, setCarregando] = useState(false);
    const navigation = useNavigation();

    const handleEditarFornecedor = (id) => {
        if (id) {
            //navigation.navigate('Editar Fornecedor', { idEditando:
'123' });
        } else {
            console.log('ID is undefined');
        }

};


    const handleExcluirFornecedor = (id) => {
        Alert.alert(
            'Confirmar exclusão',
            'Tem certeza que deseja excluir este fornecedor?',
            [
                { text: 'Cancelar', style: 'cancel' },
                { text: 'Excluir', onPress: () =>
setFornecedores(fornecedores.filter(f => f.id !== id)), style:
'destructive' },
            ],
            { cancelable: true }
        );
    };

    const dadosFiltrados = fornecedores.filter(fornecedor =>
        fornecedor.nome.toLowerCase().includes(filtro.toLowerCase())
    ).sort((a, b) => a.nome.localeCompare(b.nome));

    return (
        <View style={styles.container}>
            <TextInput
                style={styles.input}
                value={filtro}
                onChangeText={setFiltro}
```

```
                placeholder="Buscar fornecedor"
            />
            <FlatList
                data={dadosFiltrados}
                keyExtractor={item => item.id.toString()}
                renderItem={({ item }) => (
                    <View style={styles.itemContainer}>
                        <Image
                            style={styles.imagem}
                            source={{ uri: item.imagem }}
                            onError={(e) => console.log('Erro ao carregar
imagem', e.nativeEvent.error)}
                        />
                        <Text style={styles.texto}>{item.nome}</Text>
                        <Text>{item.endereco}</Text>
                        <Text>{item.contato}</Text>
                        <Text>{item.categoria}</Text>
                        <View style={styles.acoes}>
                            <TouchableOpacity
                                style={[styles.botaoAcao, {
backgroundColor: '#28a745' }]}
                                onPress={() =>
handleEditarFornecedor(item.id)}
                            >
                                <Text
style={styles.textoBotaoAcao}>Editar</Text>
                            </TouchableOpacity>
                            <TouchableOpacity
                                style={[styles.botaoAcao, {
backgroundColor: '#dc3545' }]}
                                onPress={() =>
handleExcluirFornecedor(item.id)}
                            >
                                <Text
style={styles.textoBotaoAcao}>Excluir</Text>
                            </TouchableOpacity>
                        </View>
                    </View>
                )}
                ListEmptyComponent={() => (
                    <Text style={styles.semResultados}>Nenhum fornecedor
encontrado</Text>
                )}
                onRefresh={() => {
                    setCarregando(true);
                    setTimeout(() => setCarregando(false), 1000);
                }}
                refreshing={carregando}
            />
```

```
        </View>
    );
};

const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 20,
    },
    imagem: {
        width: 100,
        height: 100,
        marginBottom: 10,
        alignSelf: 'center',
    },
    input: {
        textAlign: 'center',
        height: 40,
        marginBottom: 20,
        borderWidth: 1,
        padding: 10,
        borderColor: 'gray',
        borderRadius: 5,
    },
    itemContainer: {
        marginBottom: 10,
        padding: 10,
        backgroundColor: '#f9f9f9',
        borderRadius: 5,
    },
    texto: {
        fontSize: 16,
        fontWeight: 'bold',
    },
    acoes: {
        flexDirection: 'row',
        justifyContent: 'flex-end',
        marginTop: 5,
    },
    botaoAcao: {
        padding: 5,
        borderRadius: 3,
        marginLeft: 5,
    },
    textoBotaoAcao: {
        color: '#fff',
        fontWeight: 'bold',
    },
    semResultados: {
```

```
        textAlign: 'center',
        fontStyle: 'italic',
    },
});

export default ListaFornecedores;
```

Reducers.js

```
const fornecedoresReducer = (state = [], action) => {
    switch (action.type) {
      case 'ADICIONAR_FORNECEDOR':
        return [...state, action.payload];
      case 'EDITAR_FORNECEDOR':
        return state.map(fornecedor =>
          fornecedor.id === action.payload.id ?
action.payload.novoFornecedor : fornecedor
        );
      case 'EXCLUIR_FORNECEDOR':
        return state.filter(fornecedor => fornecedor.id !==
action.payload);
      default:
        return state;
    }
  };

  export default fornecedoresReducer;
```

dadosFornecedor.js

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

const DadosFornecedorContext = createContext();

export const Dados = ({ children }) => {
    const [fornecedores, setFornecedores] = useState([]);

    const adicionarFornecedor = fornecedor => {
        setFornecedores([...fornecedores, { ...fornecedor, id:
Date.now().toString() }]);
    };

    return (
        <DadosFornecedorContext.Provider value={{ fornecedores,
setFornecedores, adicionarFornecedor }}>
            {children}
        </DadosFornecedorContext.Provider>
    );
};

export const useFornecedores = () => {
    const context = useContext(DadosFornecedorContext);
    if (!context) {
        throw new Error('useFornecedores deve ser usado com Dados');
    }
    return context;
};
```

TelaInicial.js

```
import React from 'react';
import { StyleSheet, Text, View, Image, TouchableOpacity } from 'react-
native';

const TelaInicial = ({ navigation }) => {
    return (
        <View style={styles.container}>
            <View style={styles.logoContainer}>
                <Image style={styles.logo}
source={require('../assets/supplychain.png')} />
            </View>
            <Text style={styles.titulo}>Escolha uma opção:</Text>
            <TouchableOpacity
                style={styles.botao}
                onPress={() => navigation.navigate('Formulário do
Fornecedor')}
            >
                <Text style={styles.textoBotao}>Adicionar
Fornecedor</Text>
            </TouchableOpacity>
            <TouchableOpacity
                style={styles.botao}
                onPress={() => navigation.navigate('Listagem de
Fornecedores')}
            >
                <Text style={styles.textoBotao}>Listar
Fornecedores</Text>
            </TouchableOpacity>

        </View>
    );
};

const styles = StyleSheet.create({
    container: {
        flex: 1,
        justifyContent: 'flex-start',
        alignItems: 'center',
        backgroundColor: '#fff',
        padding: 20,
    },
    logoContainer: {
        alignItems: 'center',
        marginTop: 20,
    },
    logo: {
        width: 200,
        height: 200,
```

```
    },
    titulo: {
        fontSize: 20,
        fontWeight: 'bold',
        marginBottom: 20,
    },
    botao: {
        backgroundColor: '#007bff',
        padding: 20,
        borderRadius: 5,
        marginBottom: 10,
        width: 300,
        alignItems: 'center',

    },
    textoBotao: {
        color: '#fff',
        fontWeight: 'bold',
        fontSize: 20,

    },
});

export default TelaInicial;
```

App.js

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import 'react-native-gesture-handler';
import { Dados } from './src/dados/DadosFornecedor';
import FormularioFornecedor from './src/componentes/AddFornecedor';
import ListaFornecedores from './src/componentes/ListaFornecedor';
import TelaInicial from './src/telas/TelaInicial';


const Stack = createStackNavigator();

const App = () => {
  return (
    <Dados>
      <NavigationContainer>
        <Stack.Navigator
          screenOptions={{
            headerStyle: {
              backgroundColor: '#fff',
            },
            headerTintColor: '#333',
            headerTitleStyle: {
              fontWeight: 'bold',
              textAlign: 'center',
            },
          }}
        >
          <Stack.Screen
            name="Tela Inicial"
            component={TelaInicial}
            options={{ title: 'MettingApp - Fornecedores' }}
          />
          <Stack.Screen
            name="Formulário do Fornecedor"
            component={FormularioFornecedor}
            options={{ title: 'Cadastro de Fornecedores' }}
          />
          <Stack.Screen
            name="Listagem de Fornecedores"
            component={ListaFornecedores}
            options={{ title: 'Listagem de Fornecedores' }}
          />
        </Stack.Navigator>
      </NavigationContainer>
    </Dados>
  );
};
```

```
export default App;
```