Objetivos:

i. React Navigation;
ii. Navegação com StackNavigator;
iii. Navegação com TabNavigator;
iv. Navegação com DrawerNavigator.

i. React Navigation

A navegação entre telas é um aspecto fundamental na construção de aplicativos mobile com RN. Ela permite que o usuário se mova fluidamente entre diferentes partes do aplicativo, proporcionando uma experiência intuitiva.
A biblioteca mais utilizada para gerenciar a navegação é o React Navigation (https://reactnavigation.org). Ela oferece uma interface flexível e poderosa para criar diferentes tipos de fluxos de navegação, como pilhas (stacks), abas (tabs) e menus (drawers).
Fundamentos da navegação com React Navigation:

- Navegadores: são os componentes que gerenciam a transição entre as telas. Os principais tipos são:
  - StackNavigator: cria uma pilha de telas, onde cada tela é empilhada sobre a anterior. É ideal para navegação hierárquica;
  - TabNavigator: cria uma barra de abas, permitindo navegar entre diferentes telas;
  - DrawerNavigator: cria um menu lateral com as opções de navegação.
- Rotas: definem as telas do aplicativo e seus nomes. Cada rota é associada a um componente de tela;
- Navegação: O processo de mover o usuário entre as rotas. É realizado através de métodos como `navigation.navigate` e `navigation.goBack`.

Para configurar a navegação no RN utilizando a biblioteca `@react-navigation/native` é necessário instalar algumas dependências. Cada uma delas desempenha um papel específico na configuração e no funcionamento da navegação no aplicativo:

```
npm i @react-navigation/native @react-navigation/native-stack
npm i react-native-screens react-native-safe-area-context
```

Explicação sobre a necessidade de cada dependência:

1. @react-navigation/native (https://www.npmjs.com/package/@react-navigation/native): biblioteca central para navegação.
   - Função: esta é a biblioteca principal que fornece as funcionalidades básicas de navegação no RN. Ela gerencia o estado da navegação e o deep linking entre telas;
   - Necessidade: sem esta dependência, não podemos implementar a navegação em um aplicativo RN de forma estruturada e consistente.

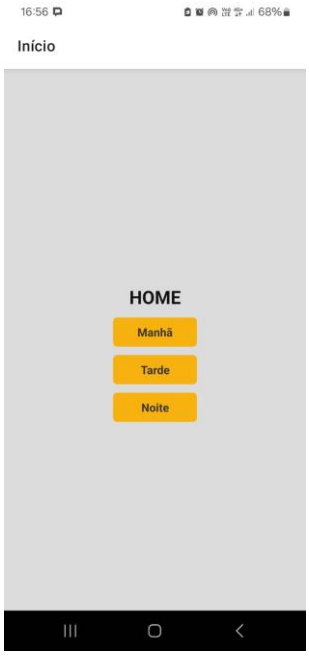
2. @react-navigation/native-stack (https://www.npmjs.com/package/@react-navigation/native-stack): implementa a navegação em pilha.
    - Função: esta dependência fornece a implementação da navegação em pilha (stack navigation), que é uma das formas mais comuns de navegação. Uma pilha permite navegar de uma tela para outra e retornar à tela anterior;
    - Necessidade: a navegação em pilha é essencial para a maioria dos aplicativos, onde o usuário pode navegar por várias telas e, em seguida, voltar para a tela anterior.
3. react-native-screens (https://www.npmjs.com/package/react-native-screens): otimiza o desempenho de navegação.
    - Função: esta biblioteca melhora o desempenho da navegação no RN ao usar componentes nativos de gerenciamento de telas. Ela permite que as telas sejam renderizadas de forma mais eficiente, especialmente em dispositivos móveis;
    - Necessidade: embora opcional, react-native-screens é altamente recomendado para melhorar o desempenho e a experiência do usuário em aplicativos que utilizam navegação.
4. react-native-safe-area-context (https://www.npmjs.com/package/react-native-safe-area-context): gerencia áreas seguras para evitar sobreposição de conteúdo.
    - Função: esta biblioteca lida com as áreas seguras (safe areas) do dispositivo, como as margens ao redor do *notch* em iPhones, bordas arredondadas e barras de navegação em dispositivos Android;
    - Necessidade: garantir que o conteúdo não seja sobreposto por áreas não utilizáveis da tela é crucial para a experiência do usuário. Esta biblioteca nos permite gerenciar essas áreas com facilidade.

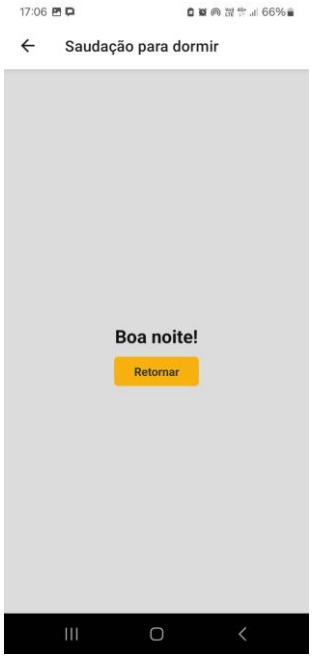
ii. Navegação com StackNavigator

Para implementar a navegação entre telas usando `@react-navigation/native`, três componentes desempenham papéis fundamentais:

- `NavigationContainer`: é o componente principal que gerencia o estado de navegação de toda a aplicação. Ele atua como um contêiner para todos os navegadores (navigators) e deve envolver o `Stack.Navigator`. Sem o `NavigationContainer`, a navegação não funcionará, pois ele é responsável por fornecer o contexto de navegação e garantir que o histórico de navegação seja mantido;
- `Stack.Navigator`: é o componente que define o escopo do stack navigator, ou seja, ele organiza as telas em uma estrutura de pilha. Ele é responsável por criar o contexto de navegação entre as telas definidas como `Stack.Screen`. Ele também permite configurar as opções de navegação, como o comportamento do botão de retorno, a animação entre as telas, e o título do cabeçalho;
- `Stack.Screen`: é utilizado para definir cada tela que faz parte do stack navigator. Cada `Stack.Screen` corresponde a uma rota no navegador, especificando um nome (`name`) e um componente (`component`) que será renderizado quando essa rota estiver ativa.

O objetivo é criar uma navegação por Stack entre os componentes `Home`, `Manha`, `Tarde` e `Noite`. As telas são mostradas a seguir. Por ser uma navegação por empilhamento, ao clicar no botão retornar, será retornada para a tela anterior.





Siga os passos:

1. Instale as seguintes dependências no projeto:

```
npm i @react-navigation/native @react-navigation/native-stack
npm i react-native-screens react-native-safe-area-context
```

2. No componente `App` são definidas as rotas de navegação para as telas:

Código do arquivo App.tsx:

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import {Home, Manha, Noite, Tarde} from './screens';
import { RootStackParamList } from './types';

const Stack = createNativeStackNavigator<RootStackParamList>();

const App: React.FC = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen
                name="Home" component={Home}
                options={{ title: 'Início' }} />
        <Stack.Screen
                name="Morning" component={Manha}
                options={{ title: 'Saudação de manhã' }} />
        <Stack.Screen
```

```
                name="Afternoon" component={Tarde}
                options={{ title: 'Saudação da tarde' }} />
        <Stack.Screen
                name="Night" component={Noite}
                options={{ title: 'Saudação para dormir' }} />
      </Stack.Navigator>
    </NavigationContainer>
  );
};

export default App;
```

Função `createNativeStackNavigator` é usada para criar um navegador (navigator) baseado em stack (pilha).

No arquivo types/index.ts, a interface `RootStackParamList` define as rotas disponíveis no stack navigator e os parâmetros que cada uma espera receber. O tipo `undefined` significa que a tela (componente) não recebem parâmetros.

Código do arquivo types/index.ts:

```
import { ParamListBase } from '@react-navigation/native';

export interface RootStackParamList extends ParamListBase {
  Home: undefined; // A tela Home não espera parâmetros
  Morning: undefined;
  Afternoon: undefined;
  Night: undefined;
}
```

Os nomes `Home`, `Morning`, `Afternoon` e `Night` presentes na interface `RootStackParamList` são nomes de rotas, ou seja, não existem componentes com esses nomes. Cada rota definida em `Stack.Screen` vincula um nome de rota (`Morning`) a um componente (`Manha`). No exemplo a seguir a propriedade `name` especifica o nome da rota para o componente `Manha`. Esse nome é utilizado posteriormente para navegar para a tela `Manha` usando funções como `navigation.navigate('Home')`.

```
<Stack.Screen name="Morning" component={Manha}
                options={{ title: 'Saudação de manhã' }} />
```

3. Definição da tela Home: o componente `Home` foi definido, no componente `App`, como tela inicial. Porém, poderia ter sido qualquer outro componente que recebesse o parâmetro `navigation`.

   O parâmetro `navigation` é disponibilizado pelo React Navigation e é injetado nos componentes de tela pelo `Stack.Navigator`. Ele fornece várias funções para navegar entre as diferentes rotas do aplicativo.

   O método `navigation.navigate('RouteName')` é usado para navegar para uma tela específica usando o nome da rota.

A instrução a seguir tem o objetivo de definir e tipar as propriedades (`Props`) do componente `Home`, fornecendo informações específicas sobre os parâmetros de navegação e as funções disponíveis para essa tela.

```
interface Props extends NativeStackScreenProps<RootStackParamList, "Home"> {}
```

Código do arquivo screens/Home.ts:

```
import React from "react";
import { View, Text, SafeAreaView, TouchableOpacity } from "react-native";
import styles from "./styles";
import { NativeStackScreenProps } from "@react-navigation/native-stack";
import { RootStackParamList } from "../../types";

interface Props extends NativeStackScreenProps<RootStackParamList, "Home"> {}

const Home: React.FC<Props> = ({ navigation }) => {
  return (
    <SafeAreaView style={styles.container}>
      <Text style={styles.title}>HOME</Text>
      <View style={styles.rowButton}>
        <TouchableOpacity
          style={styles.button}
          onPress={() => navigation.navigate("Morning")}
        >
          <Text style={styles.buttonLabel}>Manhã</Text>
        </TouchableOpacity>
        <TouchableOpacity
          style={styles.button}
          onPress={() => navigation.navigate("Afternoon")}
        >
          <Text style={styles.buttonLabel}>Tarde</Text>
        </TouchableOpacity>
        <TouchableOpacity
          style={styles.button}
          onPress={() => navigation.navigate("Night")}
        >
          <Text style={styles.buttonLabel}>Noite</Text>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
};

export default Home;
```

4. Definição da tela Manha: o componente `Manha` foi mapeado para a rota de nome `Morning`.

O método `navigation.goBack()` é usado para navegar de volta à tela anterior na pilha de navegação. Em outras palavras, ela faz com que o aplicativo retorne à última tela visitada antes da atual, removendo a tela atual da pilha de navegação.

As telas `Tarde` e `Noite` são semelhantes à tela `Manha`.

```
import React from "react";
import {
  View,
  Text,
  SafeAreaView,
  TouchableOpacity,
} from "react-native";
import styles from "./styles";
import { NativeStackScreenProps } from "@react-navigation/native-stack";
import { RootStackParamList } from "../../types";

interface Props extends NativeStackScreenProps<RootStackParamList, "Morning"> {}

const Manha: React.FC<Props> = ({ navigation }) => {
  return (
    <SafeAreaView style={styles.container}>
      <Text style={styles.title}>Bom dia!</Text>
      <View style={styles.rowButton}>
        <TouchableOpacity
          style={styles.button}
          onPress={() => navigation.goBack()}
        >
          <Text style={styles.buttonLabel}>Retornar</Text>
        </TouchableOpacity>
        <TouchableOpacity
          style={styles.button}
          onPress={() => navigation.navigate("Afternoon")}
        >
          <Text style={styles.buttonLabel}>Tarde</Text>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
};

export default Manha;
```

iii. Navegação com TabNavigator

O TabNavigator são as abas na parte superior ou inferior da tela do dispositivo. Como exemplo utilizaremos o projeto anterior.

O primeiro passo é instalar o pacote @react-navigation/bottom-tabs (https://www.npmjs.com/package/@react-navigation/bottom-tabs):

```
npm i @react-navigation/bottom-tabs
```

No componente `App` temos de definir a navegação por abas usando:

```
const Tab = createBottomTabNavigator<RootStackParamList>();
```

A seguir tem-se o código do arquivo `App.tsx`.

```tsx
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import {Home, Manha, Noite, Tarde} from './screens';
import { RootStackParamList } from './types';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

const Tab = createBottomTabNavigator<RootStackParamList>();

const App: React.FC = () => {
  return (
    <NavigationContainer>
      <Tab.Navigator initialRouteName="Home">
        <Tab.Screen name="Home" component={Home}
                    options={{ title: 'Início' }} />
        <Tab.Screen name="Morning" component={Manha}
                    options={{ title: 'Saudação de manhã' }} />
        <Tab.Screen name="Afternoon" component={Tarde}
                    options={{ title: 'Saudação da tarde' }} />
        <Tab.Screen name="Night" component={Noite}
                    options={{ title: 'Saudação para dormir' }} />
      </Tab.Navigator>
    </NavigationContainer>
  );
};

export default App;
```

As demais telas continuam iguais, exceto pela definição do tipo recebido pela tela:

Nos componentes `Home`, `Manha`, `Tarde` e `Noite`, substitua

```
import { NativeStackScreenProps } from "@react-navigation/native-stack";
```

por

```
import { BottomTabScreenProps } from "@react-navigation/bottom-tabs";
```

substitua

```
interface Props extends NativeStackScreenProps<RootStackParamList, "Home"> {}
```

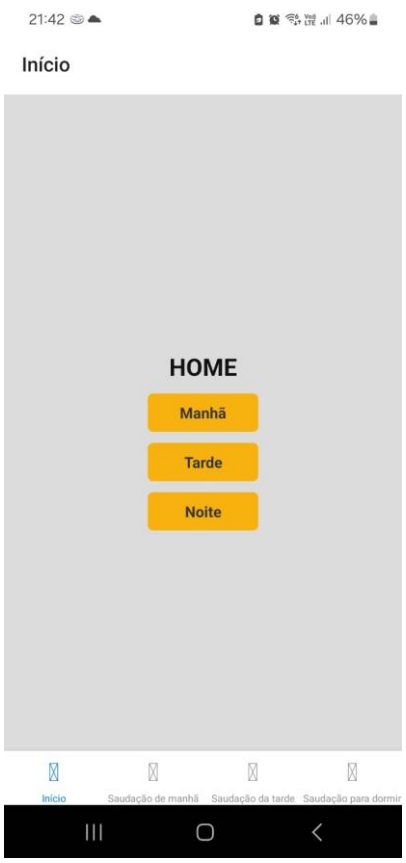
por

```
interface Props extends BottomTabScreenProps<RootStackParamList, "Home"> {}
```

Lembre-se que o nome da rota é diferente em cada tela.

É possível colocar as abas na parte superior da tela utilizando o `MaterialTopTabNavigator` do pacote @react-navigation/material-top-tabs.

A figura ao lado mostra o resultado. Veja que as abas não possuem ícones.



Para exibir ícones nas abas criadas com o `createBottomTabNavigator`, usaremos a propriedade `tabBarIcon` nas configurações de cada tela. Primeiramente instale o pacote (https://www.npmjs.com/package/react-native-vector-icons) e sua definição de tipos:

```
npm i react-native-vector-icons
npm i -D @types/react-native-vector-icons
```

Importamos a biblioteca `Ionicons`, que oferece uma coleção de ícones vetoriais que podem ser utilizados em aplicações RN.

```
import Ionicons from 'react-native-vector-icons/Ionicons';
```

A propriedade `tabBarIcon` define o ícone da aba. Ele recebe o nome do ícone (`iconName`) correspondente a cada rota e o renderiza com `Ionicons`.

A seguir tem-se o código do arquivo `App.tsx` atualizado.

Arquivo `App.tsx`:

```tsx
import React from "react";
import { NavigationContainer } from "@react-navigation/native";
import { Home, Manha, Noite, Tarde } from "./screens";
import { RootStackParamList } from "./types";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import Ionicons from "react-native-vector-icons/Ionicons";

const Tab = createBottomTabNavigator<RootStackParamList>();

const App: React.FC = () => {
  return (
    <NavigationContainer>
      <Tab.Navigator
        initialRouteName="Home"
        screenOptions={({ route }) => ({
          tabBarIcon: ({ color, size }) => {
            let iconName: string;

            switch (route.name) {
              case "Home":
                iconName = "home-outline";
                break;
```

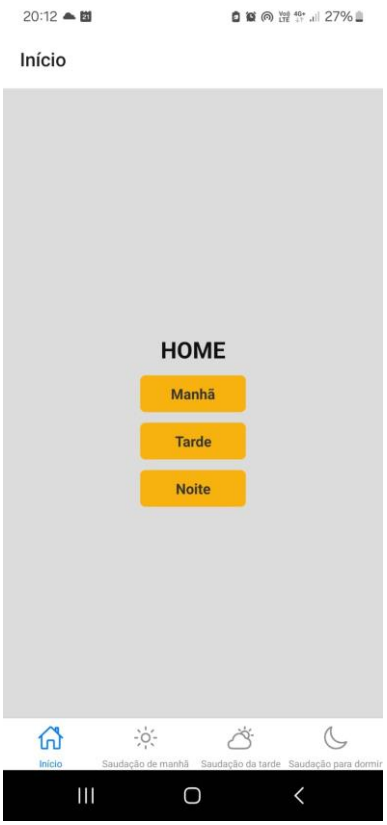
```
            case "Morning":
              iconName = "sunny-outline";
              break;
            case "Afternoon":
              iconName = "partly-sunny-outline";
              break;
            case "Night":
              iconName = "moon-outline";
              break;
            default:
              iconName = "alert-circle-outline";
          }

          // Retorna o componente de ícone
          return <Ionicons name={iconName} size={size} color={color} />;
        },
      })}
    >
      <Tab.Screen
        name="Home"
        component={Home}
        options={{ title: "Início" }}
      />
      <Tab.Screen
        name="Morning"
        component={Manha}
        options={{ title: "Saudação de manhã" }}
      />
      <Tab.Screen
        name="Afternoon"
        component={Tarde}
        options={{ title: "Saudação da tarde" }}
      />
      <Tab.Screen
        name="Night"
        component={Noite}
        options={{ title: "Saudação para dormir" }}
      />
    </Tab.Navigator>
  </NavigationContainer>
 );
};

export default App;
```

A figura ao lado mostra as abas com os ícones.

Os botões de retornar em cada aba retorna sempre para a aba Home.

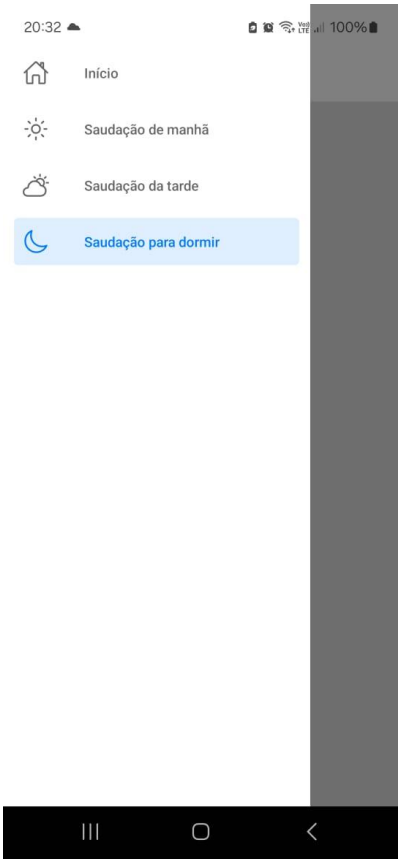


iv. Navegação com DrawerNavigator

O DrawerNavigator é o menu na lateral esquerda e as vezes colocados na direita. Como exemplo utilizaremos o projeto anterior.

Será necessário instalar a seguinte dependência:

```
npm i @react-navigation/drawer
```



No componente `App` temos de definir a navegação usando Drawer:

```
const Drawer = createDrawerNavigator<RootStackParamList>();
```

A seguir tem-se o código do arquivo `App.tsx`.

```
import React from "react";
```

```
import { NavigationContainer } from "@react-navigation/native";
import { createDrawerNavigator } from "@react-navigation/drawer";
import Ionicons from "react-native-vector-icons/Ionicons";
import { Home, Manha, Noite, Tarde } from "./screens";
import { RootStackParamList } from "./types";

const Drawer = createDrawerNavigator<RootStackParamList>();

const App: React.FC = () => {
  return (
    <NavigationContainer>
      <Drawer.Navigator
        initialRouteName="Home"
        screenOptions={({ route }) => ({
          drawerIcon: ({ color, size }) => {
            let iconName: string;

            switch (route.name) {
              case "Home":
                iconName = "home-outline";
                break;
              case "Morning":
                iconName = "sunny-outline";
                break;
              case "Afternoon":
                iconName = "partly-sunny-outline";
                break;
              case "Night":
                iconName = "moon-outline";
                break;
              default:
                iconName = "alert-circle-outline";
            }

            return <Ionicons name={iconName} size={size} color={color} />;
          },
        })}
      >
        <Drawer.Screen
          name="Home"
          component={Home}
          options={{ title: "Início" }}
        />
        <Drawer.Screen
          name="Morning"
          component={Manha}
          options={{ title: "Saudação de manhã" }}
        />
        <Drawer.Screen
```

```
          name="Afternoon"
          component={Tarde}
          options={{ title: "Saudação da tarde" }}
        />
        <Drawer.Screen
          name="Night"
          component={Noite}
          options={{ title: "Saudação para dormir" }}
        />
      </Drawer.Navigator>
    </NavigationContainer>
  );
};

export default App;
```

A seguir tem-se o código do arquivo `Home.tsx`.

```
import React from "react";
import {
  View,
  Text,
  SafeAreaView,
  TouchableOpacity,
} from "react-native";
import styles from "./styles";
import { DrawerScreenProps } from "@react-navigation/drawer";
import { RootStackParamList } from "../../types";

interface Props extends DrawerScreenProps<RootStackParamList, "Home"> {}

const Home: React.FC<Props> = ({ navigation }) => {
  return (
    <SafeAreaView style={styles.container}>
      <Text style={styles.title}>HOME</Text>
      <View style={styles.rowButton}>
        <TouchableOpacity
          style={styles.button}
          onPress={() => navigation.navigate("Morning")}
        >
          <Text style={styles.buttonLabel}>Manhã</Text>
        </TouchableOpacity>
        <TouchableOpacity
          style={styles.button}
          onPress={() => navigation.navigate("Afternoon")}
        >
          <Text style={styles.buttonLabel}>Tarde</Text>
        </TouchableOpacity>
        <TouchableOpacity
```

```
          style={styles.button}
          onPress={() => navigation.navigate("Night")}
        >
          <Text style={styles.buttonLabel}>Noite</Text>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
};

export default Home;
```

As demais telas continuam iguais, exceto pela definição do tipo recebido pela tela:

Nos componentes `Home`, `Manha`, `Tarde` e `Noite`, substitua

```
import { BottomTabScreenProps } from "@react-navigation/bottom-tabs";
```

por

```
import { DrawerScreenProps } from "@react-navigation/drawer";
```

substitua

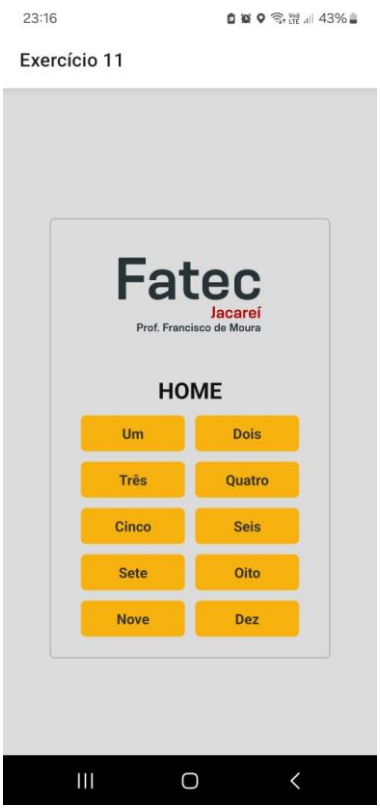
```
interface Props extends BottomTabScreenProps<RootStackParamList, "Home"> {}
```

por

```
interface Props extends DrawerScreenProps<RootStackParamList, "Home"> {}
```

Exercícios

Exercício 1 – Alterar o aplicativo do Exercício 11 da Aula 1 para que os botões da tela `Onze` permitam a navegação ao clicar nos botões.



Requisitos:

- Utilize `StackNavigator`.

Dicas:

- No componente `App` adicione a seguinte estrutura:
  - Crie um objeto Stack usando a função `createNativeStackNavigator`;
  - Defina as rotas usando os componentes `NavigationContainer`, `Stack.Navigator` e `Stack.Screen`.
- No componente `Onze`:
  - A função componente precisa receber o objeto `navigation`;
  - Use o método `navigate` do objeto `navigation` para abrir a tela desejada.
- As demais telas não precisam de alteração;
- Crie o arquivo de tipos `index.ts` na pasta `types` para definir os parâmetros recebidos por cada tela roteada:

```
import { ParamListBase } from '@react-navigation/native';

export interface RootStackParamList extends ParamListBase {
    Um: undefined;
    Dois: undefined;
    Tres: undefined;
    Quatro: undefined;
    Cinco: undefined;
    Seis: undefined;
    Sete: undefined;
    Oito: undefined;
    Nove: undefined;
    Dez: undefined;
    Onze: undefined;
}
```

Exercício 2 – Alterar o aplicativo do Exercício 1 para a navegação ser pelo menu drawer.

Requisito:

- Excluir o componente `Onze`, a navegação será pelo menu drawer.
- O componente `Um` deverá ser a tela de início.

Dicas:

- No componente `App`, substitua a função `createNativeStackNavigator` por `createDrawerNavigator` para criar o `Stack` de navegação;
- Se executar no navegador e estiver dando problema no Expo Go, talvez seja a versão do pacote react-native-reanimated.

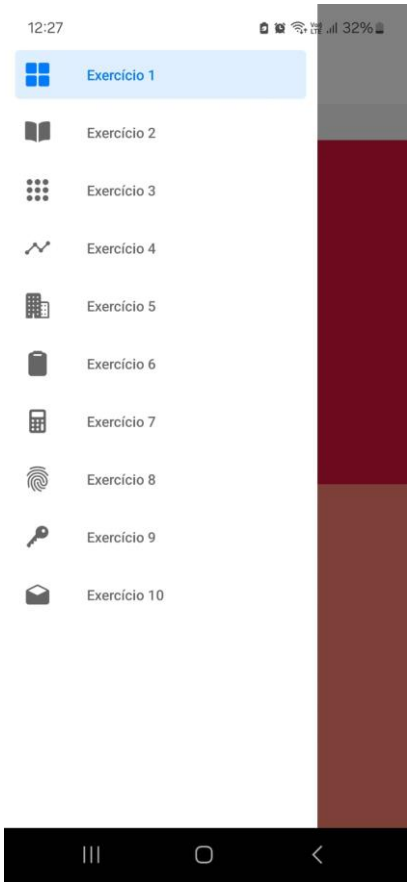
Exercício 3 – Adicione ícones nos itens do menu do Exercício 2.

Requisito:

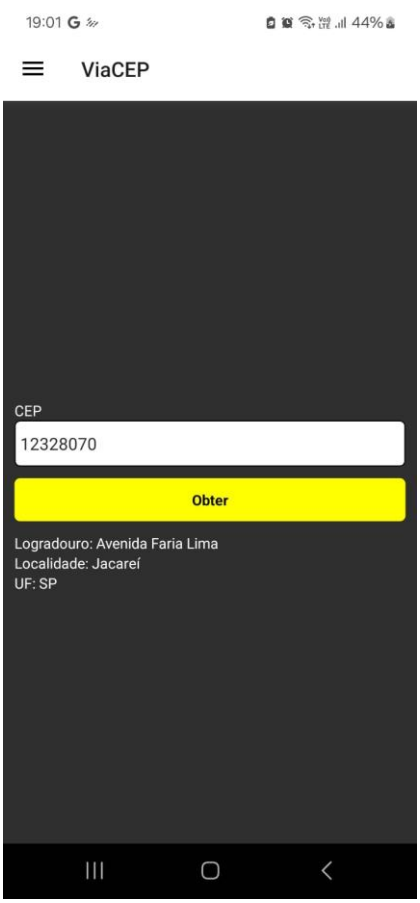
- Cada item do menu precisa ter um ícone escolhido por você em https://ionic.io/ionicons/v4.

Dicas:

- Adicione a biblioteca react-native-vector-icons;
- Adicione a propriedade `screenOptions` na marcação `Drawer.Navigator`.



Exercício 4 – Fazer um aplicativo que faz a conexão com o serviço do ViaCEP para o usuário consultar CEP.



Requisitos:

- Apesar do aplicativo ter apenas uma tela. Essa tela deverá estar disponível através de um drawer menu;
- A aplicação deverá ter o visual apresentado ao lado. O campo de entrada deverá habilitar o teclado numérico e ao clicar no botão "Obter" deverá ser exibido o resultado da consulta ao serviço do ViaCEP;
- A aplicação deverá ser organizada nas pastas services, contexts, hooks, types e screens.

Dicas:

- Use axios para fazer a conexão com o web service;
- Não esqueça de envolver as marcações do componente App pelo Provider do Contexto definido por você na pasta contexts.

Exercício 5 – Alterar o aplicativo do Exercício 4.

Requisitos:

- A aplicação deverá exibir a mensagem de "CEP inválido" quando o web service retonar {"erro":"true"};
- Adicionar uma segunda tela para listar todas as consultas realizadas;
- A tela deverá ter o scroll habilitado.

Dicas:

- Crie uma propriedade no estado do contexto para manter os ceps consultados. Deverá ser mantido nessa lista apenas os CEPs válidos;
- Use o componente ScrollView no componente que exibe a lista de CEPs consultados.