

React Native - Navigation

O que vamos ver hoje?

- React Navigation
- Stack Navigation
- Tab Navigation
- Drawer Navigation

O que é React Navigation

React Navigation - O que é

- React Native não apresenta mecanismos de navegação de forma nativa.
- *React Navigation* é uma das soluções existentes mais famosas e recomendadas para roteamento e navegação entre telas no React Native.
 - <https://reactnavigation.org/>

Instalação

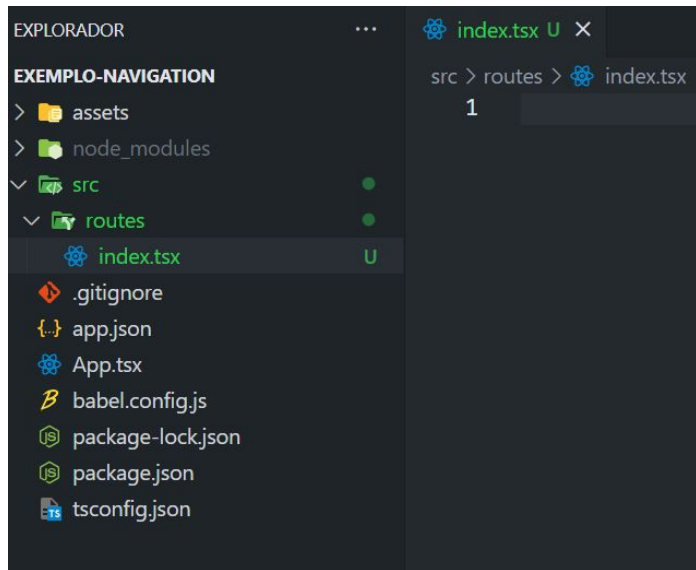
Instalação

- Instalar a base com *npm install @react-navigation/native*
- Se utilizar Expo: *npx expo install react-native-screens react-native-safe-area-context*
- Se utilizar CLI: *npm install react-native-screens react-native-safe-area-context*
- Se for usuário de MacOS: *npx pod-install ios*
- Ao utilizar React Native CLI existem passos adicionais. Conferir a documentação!

Iniciando

Iniciando

- Após iniciar seu projeto, crie uma pasta **src**, dentro dela uma pasta **routes**
- Dentro da pasta routes, crie um arquivo **index.tsx**



Iniciando


- Faça a importação do NavigationContainer e a declaração no retorno:



```
1  import { NavigationContainer } from "@react-navigation/native";
2
3  export const Routes = () => {
4    return (
5      <NavigationContainer>
6
7      </NavigationContainer>
8    );
9  };
```

Iniciando

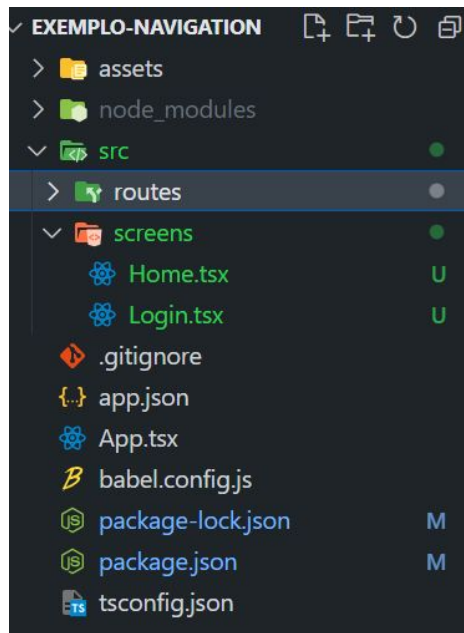
- No arquivo **App.tsx** faça a importação do componente de rotas:



```
1  import { Routes } from "../src/routes";  
2  
3  export default function App() {  
4    return <Routes />;  
5  }
```

Iniciando

- Dentro da pasta **src**, crie uma pasta **screens**
- Dentro da pasta **screens** crie um arquivo **Login.tsx** e um arquivo **Home.tsx**



Iniciando



```
1 import React from "react";
2 import { View, Button } from "react-native";
3
4 export const Login = () => {
5   return (
6     <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
7       <Button title="Ir para Home" />
8     </View>
9   );
10 };
```



```
1 import React from "react";
2 import { View, Button } from "react-native";
3
4 export const Home = () => {
5   return (
6     <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
7       <Button title="Voltar para Login" />
8     </View>
9   );
10 };
```

Stack Navigation

Stack Navigation

- Navegação de pilhas: cada página é empilhada em cima da outra.
- Instalação: `npm install @react-navigation/native-stack`
- Na pasta **src**, crie uma pasta **types**
- Dentro dela crie um arquivo **navigation.ts**

Stack Navigation

- Dentro do arquivo **navigation.ts** insira o código:



```
1 import { NativeStackScreenProps } from "@react-navigation/native-stack";
2
3 export type ScreenParamList = {
4   login: undefined;
5   home: undefined;
6 };
7
8 export type LoginStackProps = NativeStackScreenProps<ScreenParamList, "login">;
9 export type HomeStackProps = NativeStackScreenProps<ScreenParamList, "home">;
```

- Neste arquivo ficará a tipagem necessária para o navigation do nosso projeto.

Stack Navigation

- Na pasta **routes**, crie um arquivo **stack.routes.tsx** e insira o código:

```
1 import React from "react";
2 import { createNativeStackNavigator } from "@react-navigation/native-stack";
3 import { Login } from "../screens/Login";
4 import { Home } from "../screens/Home";
5 import { ScreenParamList } from "../types/navigation";
6
7 const { Screen, Navigator } = createNativeStackNavigator<ScreenParamList>();
8
9 export function StackRoutes() {
10   return (
11     <Navigator>
12       <Screen name="login" component={Login} />
13       <Screen name="home" component={Home} />
14     </Navigator>
15   );
16 }
17
```


Stack Navigation

- Precisamos agora fazer as importações para utilizar o navigation:

```
1 import React from "react";
2 import { View, Button } from "react-native";
3 import { LoginStackProps } from "../types/navigation";
4
5 export const Login = ({ navigation }: LoginStackProps) => {
6   return (
7     <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
8       <Button title="Ir para Home" />
9     </View>
10  );
11 };
```

```
1 import React from "react";
2 import { View, Button } from "react-native";
3 import { HomeStackProps } from "../types/navigation";
4
5 export const Home = ({ navigation }: HomeStackProps) => {
6   return (
7     <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
8       <Button title="Voltar para Login" />
9     </View>
10  );
11 };
12
```

Stack Navigation

- Para saber mais das opções do navigation acesse a documentação oficial:
 - <https://reactnavigation.org/docs/navigation-prop/>

Stack Navigation

- Nas telas **Login.tsx** e **Home.tsx** criar uma função `openScreen` e chamá-la no `onPress` do button:

```
import React from "react";
import { View, Button } from "react-native";
import { LoginStackProps } from "../types/navigation";

export const Login = ({ navigation }: LoginStackProps) => {
  const openScreen = () => {
    navigation.navigate("home");
  };

  return (
    <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
      <Button title="Ir para Home" onPress={openScreen} />
    </View>
  );
};
```

```
import React from "react";
import { View, Button } from "react-native";
import { HomeStackProps } from "../types/navigation";

export const Home = ({ navigation }: HomeStackProps) => {
  const openScreen = () => {
    navigation.navigate("login");
  };

  return (
    <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
      <Button title="Voltar para Login" onPress={openScreen} />
    </View>
  );
};
```

Stack Navigation

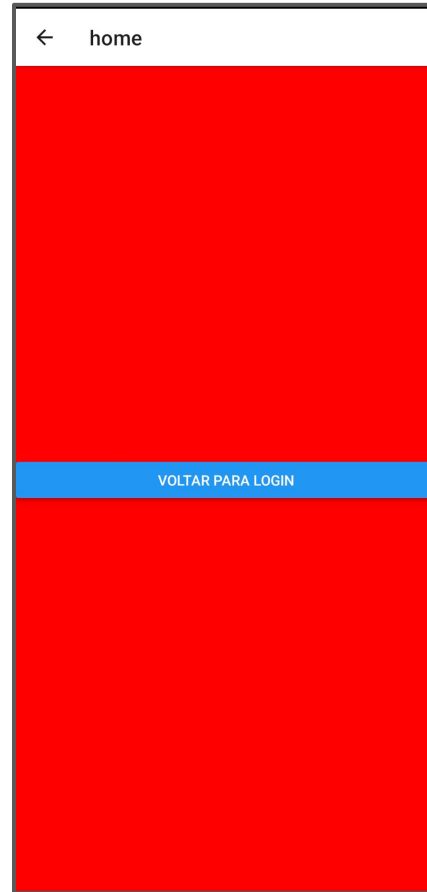
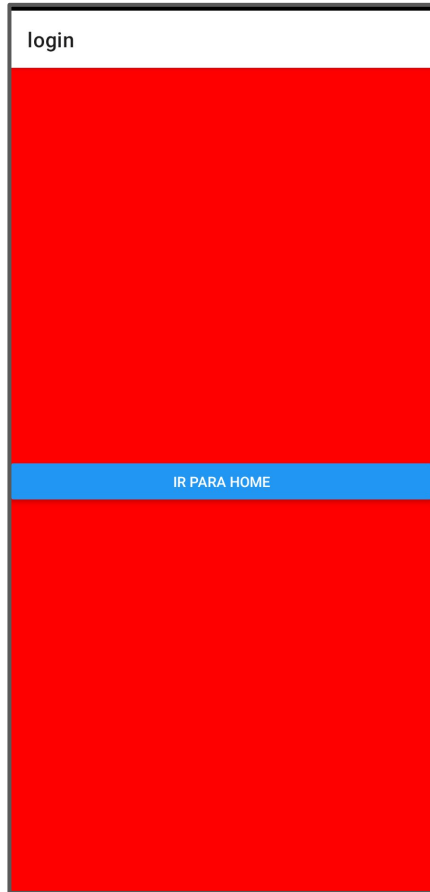
- No arquivo **index.tsx** da pasta **routes**, importar o *StackRoutes*:



```
1  import { NavigationContainer } from "@react-navigation/native";
2  import { StackRoutes } from "../stack.routes";
3
4  export const Routes = () => {
5    return (
6      <NavigationContainer>
7        <StackRoutes />
8      </NavigationContainer>
9    );
10  };

```

Stack Navigation



Tab Navigation

Tab Navigation

- Navegação por abas
- Instalação: `npm install @react-navigation/bottom-tabs`
- Na pasta screens crie dois arquivos: `Feed.tsx` e `Configurations.tsx`

Tab Navigation

- Nos arquivos insira os conteúdos:

```
import React from "react";
import { View, Text } from "react-native";

const Feed = () => {
  return (
    <View
      style={{ flex: 1, backgroundColor: "blue", justifyContent: "center" }}
    >
      <Text style={{ alignSelf: "center", color: "#fff" }}>Feed</Text>
    </View>
  );
};

export default Feed;
```

```
import React from "react";
import { View, Text } from "react-native";

const Configurations = () => {
  return (
    <View
      style={{ flex: 1, backgroundColor: "green", justifyContent: "center" }}
    >
      <Text style={{ alignSelf: "center", color: "#fff" }}>Configurations</Text>
    </View>
  );
};

export default Configurations;
```


Tab Navigation

- Em **types/navigation.ts** adicione a tipagem:

```
1 import { NativeStackScreenProps } from "@react-navigation/native-stack";
2 import { BottomTabScreenProps } from "@react-navigation/bottom-tabs";
3
4 export type ScreenParamList = {
5   login: undefined;
6   home: undefined;
7   feed: undefined;
8   configurations: undefined;
9 };
10
11 export type LoginStackProps = NativeStackScreenProps<ScreenParamList, "login">;
12 export type HomeStackProps = NativeStackScreenProps<ScreenParamList, "home">;
13
14 export type FeedTabProps = BottomTabScreenProps<ScreenParamList, "feed">;
15 export type ConfigurationsTabProps = BottomTabScreenProps<
16   ScreenParamList,
17   "configurations"
18 >;
```

Tab Navigation

- Na pasta **routes**, crie um arquivo **tab.routes.tsx** e insira o código:

```
1 import React from "react";
2 import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
3 import { MaterialIcons } from "@expo/vector-icons";
4 import { ScreenParamList } from "../types/navigation";
5 import Feed from "../screens/Feed";
6 import Configurations from "../screens/Configurations";
7
8 const { Screen, Navigator } = createBottomTabNavigator<ScreenParamList>();
9
10 export function TabRoutes() {
11   return (
12     <Navigator>
13       <Screen
14         name="feed"
15         component={Feed}
16         options={{
17           tabBarIcon: ({ color, size }) => (
18             <MaterialIcons name="feed" color={color} size={size} />
19           ),
20         }}
21       />
22       <Screen
23         name="configurations"
24         component={Configurations}
25         options={{
26           tabBarIcon: ({ color, size }) => (
27             <MaterialIcons name="settings" color={color} size={size} />
28           ),
29         }}
30     />
31   </Navigator>
32 );
33 }
34
```

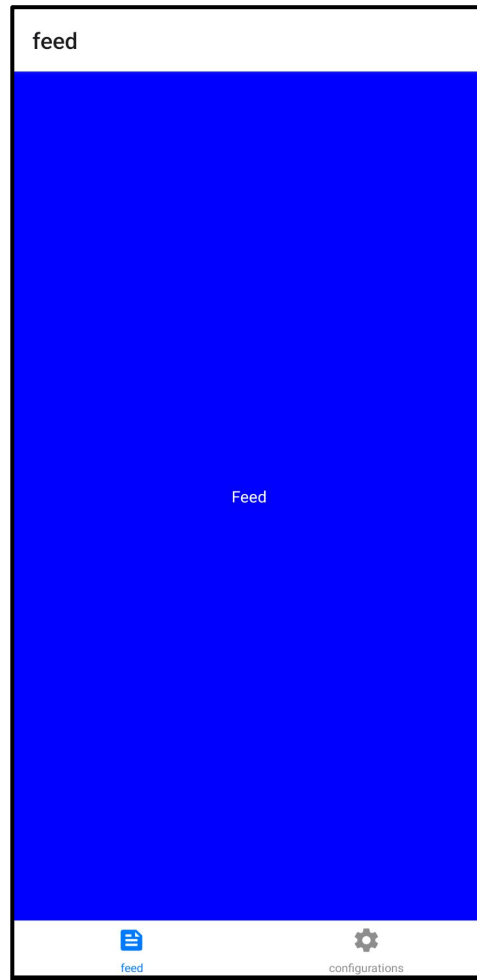
Tab Navigation

- No arquivo **routes/index.tsx** substitua o `<StackRoutes />` por `<TabRoutes />`



```
1  import { NavigationContainer } from "@react-navigation/native";
2  import { TabRoutes } from "../tab.routes";
3
4  export const Routes = () => {
5    return (
6      <NavigationContainer>
7        <TabRoutes />
8      </NavigationContainer>
9    );
10  };
```

Tab Navigation



Drawer Navigation

Drawer Navigation

- Navegação de gaveta
- Instalação:
 - `npm install @react-navigation/drawer`
 - `npx expo install react-native-gesture-handler react-native-reanimated` ou `npm install react-native-gesture-handler react-native-reanimated`
 - `npx pod-install ios` (se estiver utilizando MacOS)
- No topo do arquivo `App.tsx` insira:
 - `import "react-native-gesture-handler";`

Drawer Navigation

- No arquivo **babel.config.js** insira o trecho: *plugins:*
["react-native-reanimated/plugin"],



```
1  module.exports = function (api) {  
2    api.cache(true);  
3    return {  
4      presets: ["babel-preset-expo"],  
5      plugins: ["react-native-reanimated/plugin"],  
6    };  
7  };  
8
```



Tab Navigation

- Em **types/navigation.ts** adicione a tipagem:

```
import { NativeStackScreenProps } from "@react-navigation/native-stack";
import { BottomTabScreenProps } from "@react-navigation/bottom-tabs";
import { DrawerScreenProps } from "@react-navigation/drawer";

export type ScreenParamList = {
  login: undefined;
  home: undefined;
  feed: undefined;
  configurations: undefined;
};

export type LoginStackProps = NativeStackScreenProps<ScreenParamList, "login">;
export type HomeStackProps = NativeStackScreenProps<ScreenParamList, "home">;

export type FeedTabProps = BottomTabScreenProps<ScreenParamList, "feed">;
export type ConfigurationsTabProps = BottomTabScreenProps<
  ScreenParamList,
  "configurations"
>;

export type FeedDrawerProps = DrawerScreenProps<ScreenParamList, "feed">;
export type ConfigurationsDrawerProps = DrawerScreenProps<
  ScreenParamList,
  "configurations"
>;
```


Drawer Navigation

- Vamos aproveitar as telas que temos.
- Em **routes** crie um arquivo **drawer.routes.tsx**

Drawer Navigation

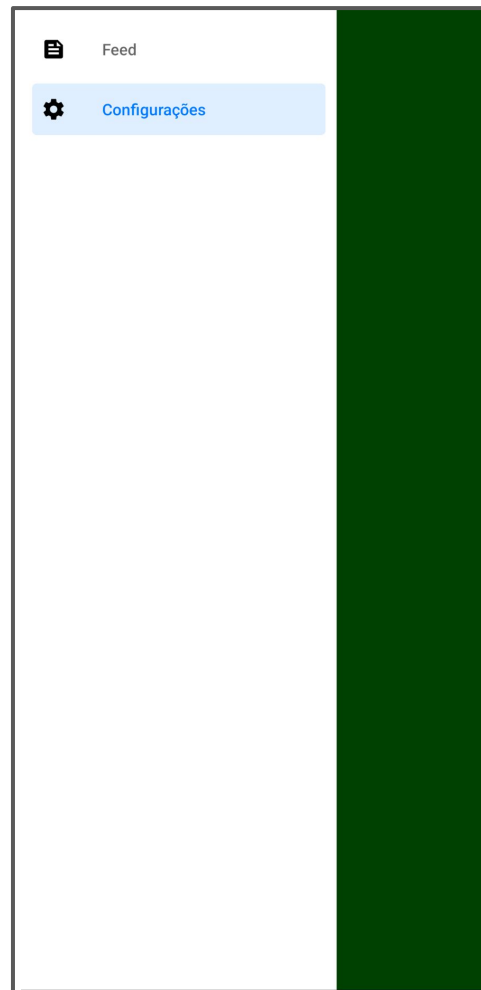
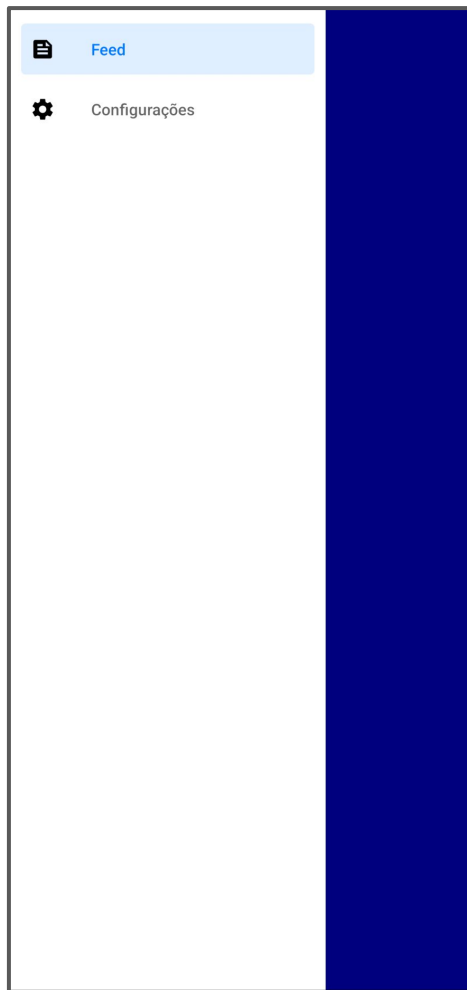
```
1 import React from "react";
2 import { createDrawerNavigator } from "@react-navigation/drawer";
3 import { MaterialIcons } from "@expo/vector-icons";
4 import { ScreenParamList } from "../types/navigation";
5 import Feed from "../screens/Feed";
6 import Configurations from "../screens/Configurations";
7
8 const { Screen, Navigator } = createDrawerNavigator<ScreenParamList>();
9
10 export function DrawerRoutes() {
11   return (
12     <Navigator>
13       <Screen
14         name="feed"
15         component={Feed}
16         options={{
17           drawerLabel: "Feed",
18           headerShown: false,
19           drawerIcon: () => <MaterialIcons name="feed" size={22} />,
20         }}
21       />
22       <Screen
23         name="configurations"
24         component={Configurations}
25         options={{
26           drawerLabel: "Configurações",
27           headerShown: false,
28           drawerIcon: () => <MaterialIcons name="settings" size={22} />,
29         }}
30     />
31   </Navigator>
32 );
33 }
34
```

Drawer Navigation

- No arquivo **routes/index.tsx** substitua o `<TabRoutes />` por `<DrawerRoutes />`

```
1  import { NavigationContainer } from "@react-navigation/native";
2  import { DrawerRoutes } from "../drawer.routes";
3
4  export const Routes = () => {
5    return (
6      <NavigationContainer>
7        <DrawerRoutes />
8      </NavigationContainer>
9    );
10  };
```

Drawer Navigation



useNavigation()

useNavigation()

- Uma forma mais resumida de obter o objeto de navegação é com o hook useNavigation()
- Ao invés de usar o navigation como props e precisando tipar esse objeto dessa forma:

```
export const Login = ({ navigation }: LoginStackProps) => {  
  const openScreen = () => {  
    navigation.navigate("home");  
  };  
  
  return (  
    <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>  
      <Button title="Ir para Home" onPress={openScreen} />  
    </View>  
  );  
};
```

useNavigation()


- Podemos fazer assim:

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

export const Login = () => {
  const navigation = useNavigation();

  const openScreen = () => {
    navigation.navigate("home");
  };

  return (
    <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
      <Button title="Ir para Home" onPress={openScreen} />
    </View>
  );
};
```



Passagem de parâmetros

Parâmetros


- Muitas vezes precisamos passar dados de uma tela para outra
- Para fazer isso é bem simples. Aqui precisaremos utilizar os types que foram criados anteriormente.
- Vamos voltar nosso arquivo **routes/index.tsx** para utilizar o StackRoutes onde já temos botões configurados na tela:

```
import { NavigationContainer } from "@react-navigation/native";
import { StackRoutes } from "../stack.routes";

export const Routes = () => {
  return (
    <NavigationContainer>
      <StackRoutes />
    </NavigationContainer>
  );
};
```

Parâmetros

- Na pasta **types** crie um arquivo **user.ts**
- vamos criar uma interface para definir os tipos de dados que vamos passar de parâmetro:



```
1  export interface User {  
2      name: string;  
3      job: string;  
4  }
```

Parâmetros

- Em **types/navigation.ts** precisamos dizer qual tipo de dados que nossa tela irá receber. No caso vamos passar dados para a tela Home:



```
1  export type ScreenParamList = {
2    login: undefined;
3    home: User;
4    feed: undefined;
5    configurations: undefined;
6  };
```

Parâmetros

- Na nossa tela de **login**, criamos um objeto com os dados que queremos passar e inserimos como segundo parâmetro do `navigate`:



```
1  const user: User = {  
2    name: "Rafael",  
3    job: "Desenvolvedor Pleno",  
4  };  
5  
6  const openScreen = () => {  
7    navigation.navigate("home", user);  
8  };
```

Parâmetros

- Na tela **Home**, precisamos receber esses parâmetros
- Importamos a função `route` do `navigation` e recuperamos os parâmetros com a desestruturação de objeto



```
1 export const Home = ({ navigation, route }: HomeStackProps) => {  
2   const { name, job } = route.params as User;
```

Parâmetros

- Criamos dois Text para exibir os dados



```
1 <Text style={{ alignSelf: "center", color: "#fff" }}>{name}</Text>
2 <Text style={{ alignSelf: "center", color: "#fff" }}>{job}</Text>
```

Parâmetros



```
1 import React from "react";
2 import { View, Button, Text } from "react-native";
3 import { HomeStackProps } from "../types/navigation";
4 import { User } from "../types/user";
5
6 export const Home = ({ navigation, route }: HomeStackProps) => {
7   const { name, job } = route.params as User;
8
9   const openScreen = () => {
10     navigation.navigate("login");
11   };
12
13   return (
14     <View style={{ flex: 1, backgroundColor: "red", justifyContent: "center" }}>
15       <Text style={{ alignSelf: "center", color: "#fff" }}>{name}</Text>
16       <Text style={{ alignSelf: "center", color: "#fff" }}>{job}</Text>
17       <Button title="Voltar para Login" onPress={openScreen} />
18     </View>
19   );
20 };
```

Parâmetros

← home

Rafael

Desenvolvedor Pleno

VOLTAR PARA LOGIN

useRoute()

useRoute()

- Podemos simplificar a maneira de recuperar parâmetros da rota com o hook useRoute()
- Ao invés de recuperar os parâmetros desta forma:

```
export const Home = ({ navigation, route }: HomeStackProps) => {  
  const { name, job } = route.params as User;
```

- Podemos fazer assim:

```
export const Home = ({ navigation }: HomeStackProps) => {  
  const route = useRoute();  
  const { name, job } = route.params as User;
```

Navegação complexa

Navegação Complexa

- Em muitos aplicativos, apenas uma estratégia de navegação não é suficiente para dar conta da necessidade.
- Para isso precisamos combinar as estratégias.

Navegação Complexa

- Vamos supor que nosso aplicativo tenha que ter telas acessíveis publicamente (login e cadastro) e outras telas acessíveis somente após o login.
- Desta forma podemos ter duas rotas principais: as de autenticação e as da aplicação.

Navegação Complexa

- Com essa lógica verificamos se o usuário está logado e direciona para as rotas indicadas, caso não esteja, direciona para a tela de login

```
return (  
  <NavigationContainer>  
    {user ? <AppRoutes /> : <AuthRoutes />}  
  </NavigationContainer>  
);
```

Navegação Complexa

- Vamos desenvolver nossa <AuthRoutes/>
- Ela será um StackNavigator com duas telas: SignIn e SignUp

```
export const AuthRoutes = () => {  
  return (  
    <Navigator screenOptions={{ headerShown: false }}>  
      <Screen name="signIn" component={SignIn} />  
      <Screen name="signUp" component={SignUp} />  
    </Navigator>  
  );  
};
```

Navegação Complexa

- Vamos desenvolver nossa <AppRoutes/>
- Aqui que começa a magia!
- Nessa rota vamos combinar as estratégias de Stack, Tab e Drawer.
- Podemos escrever no mesmo arquivo ou em separados.

Navegação Complexa

- O primeiro passo é criar as rotas Stack, que terão as páginas Home e Details:

```
const StackRoutes = () => {  
  return (  
    <Stack.Navigator screenOptions={{ headerShown: false }}>  
      <Stack.Screen  
        name="home"  
        component={Home}  
        options={({ navigation }) => ({  
          headerLeft: () => (  
            <Pressable onPress={() => navigation.toggleDrawer()}>  
              <MaterialIcons name="menu" size={25} />  
            </Pressable>  
          ),  
        })}  
      </Stack.Screen>  
      <Stack.Screen name="details" component={Details} />  
    </Stack.Navigator>  
  );  
};
```

Navegação Complexa

- Precisamos criar agora nosso TabNavigation. Ele terá como rota o StackNavigation criado anteriormente e uma tela Settings:

Navegação Complexa

```
1  const TabRoutes = () => {
2    return (
3      <Tab.Navigator screenOptions={{ headerShown: false }}>
4        <Tab.Screen
5          name="stack"
6          component={StackRoutes}
7          options={{
8            tabBarLabel: "home", // renomenado a Tab label
9            tabBarIcon: ({ color, size }) => (
10              <Feather name="home" color={color} size={size} />
11            ),
12          }}
13        />
14        <Tab.Screen
15          name="settings"
16          component={Settings}
17          options={{
18            tabBarIcon: ({ color, size }) => (
19              <Feather name="settings" color={color} size={size} />
20            ),
21          }}
22        />
23      </Tab.Navigator>
24    );
25  };
```

Navegação Complexa

- Por último criamos o DrawerRoutes, que receberá o TabRoutes e uma screen chamada Profile:

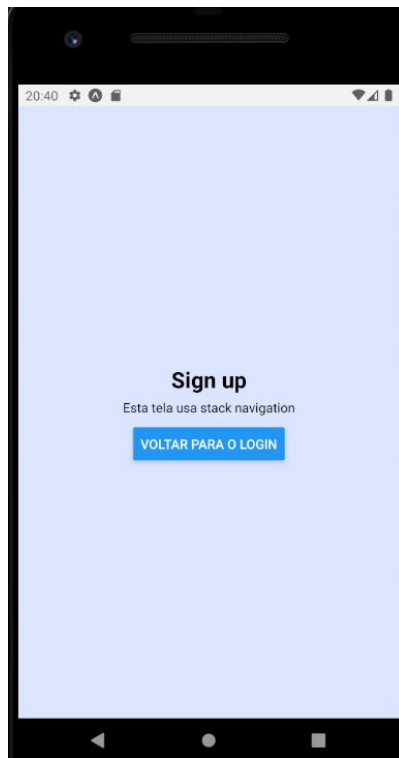
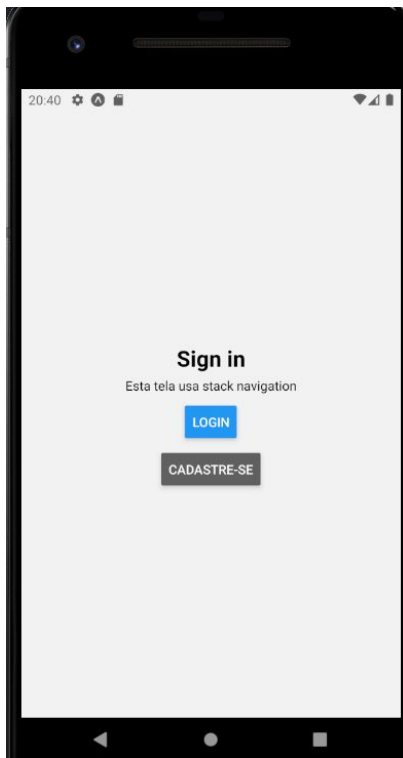
```
const DrawerRoutes = () => {  
  return (  
    <Drawer.Navigator  
      initialRouteName="Home"  
      screenOptions={{ headerTitle: "" }}  
    >  
      <Drawer.Screen name="Home" component={TabRoutes} />  
      <Drawer.Screen name="Profile" component={Profile} />  
    </Drawer.Navigator>  
  );  
};
```

Navegação Complexa

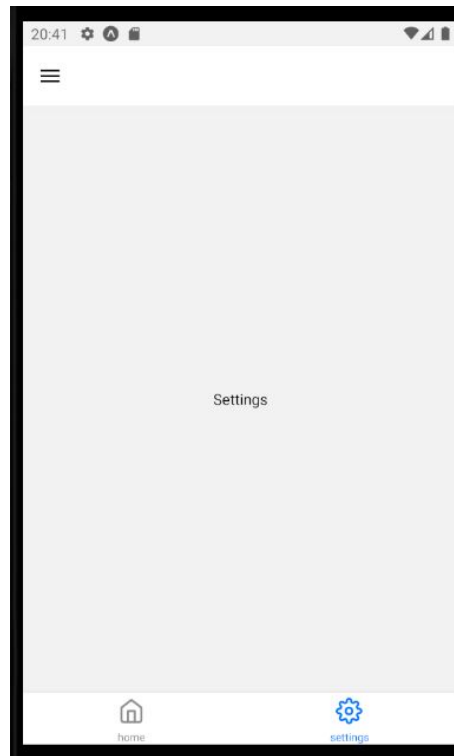
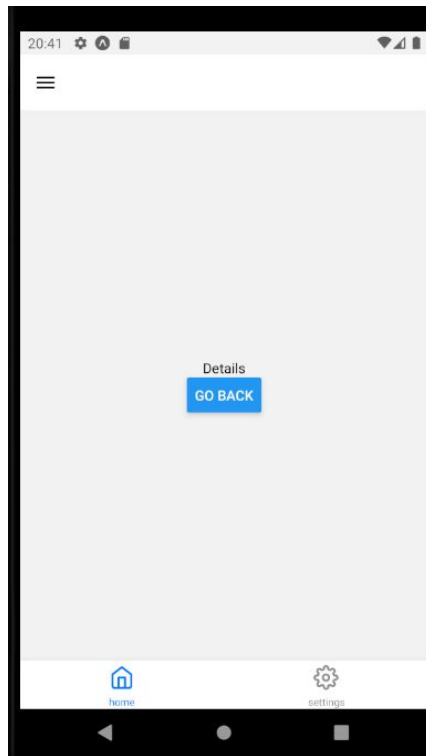
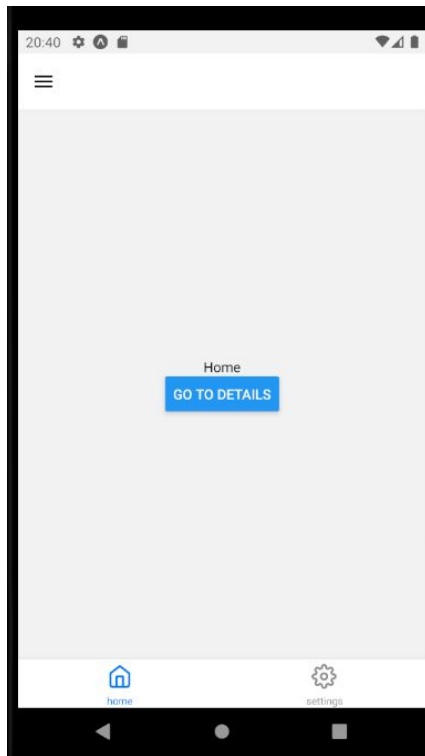
- Mas qual estratégia deverá ser colocada como a principal?
Sempre a de mais alto nível, em nosso exemplo a DrawerRoutes!

```
export { DrawerRoutes as AppRoutes };
```

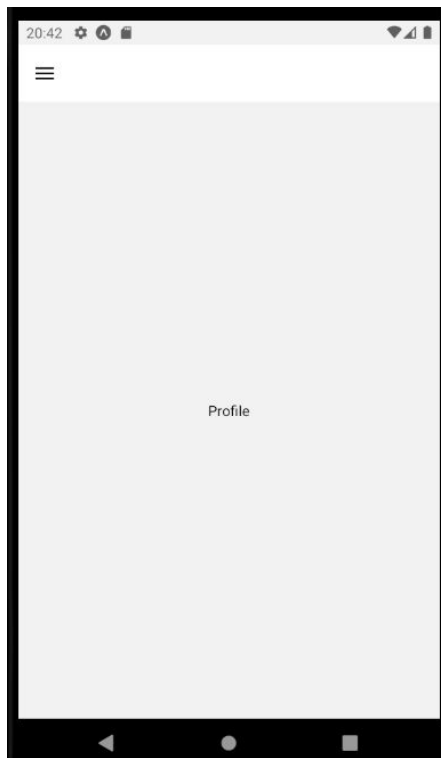
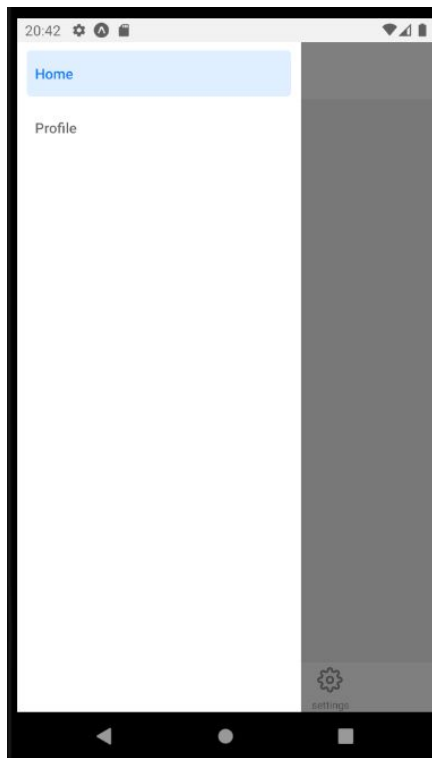
Navegação Complexa



Navegação Complexa



Navegação Complexa



Repositório de exemplo

Repositório de exemplo

- Criei um repositório que pode servir como ponto de partida.
- Nele tem uma configuração de navegação com drawer, tab e stack.
- <https://github.com/rafaelkasper/routes-template>

Dúvidas?



Tarefa

- Crie uma tela de login com inputs de usuário e senha e um botão de confirmação
- Crie uma lógica simples de login (com username e senha fixos)
- Ao logar, direcione para a tela Home
- **Importante: reaproveite os componentes criados nas aulas anteriores!**