```
import React from "react";
//Aqui temos a importação do método "createNativeStackNavigation"...
import { createNativeStackNavigator } from "@react-navigation/native-stack";

//Aqui temos a importação das Screens...
import TelaA from "../views/TelaA";
import TelaB from "../views/TelaB";
import TelaC from "../views/TelaC";

//Aqui temos a importação de um Componente que serve somente para passar parâmetros a cada Screen, depedendo do
parâmetro passado, teremos um comportamento diferente para cada Screen...
import PassoStack from "../componentes/PassoStack";

/*  STACK:

    Uma das navegações que o React Navigation usa é a navegação em Stack, esse tipo de navegação consiste em fazer
com que o aplicativo navegue entre telas como se elas estivessem empilhadas. Mas, o mais legal é que elas não
precisam necessariamente estarem empilhadas de uma determinada forma para que possamos ir para uma determinada tela
ou outra. Podemos escolher para que tela queremos navegar.

                        ________
                       /       /  -----> Tela 1
                      /       /___
                     /_______/   / -----> Tela 2
                        /       /___
                       /_______/   /
                          /       / ------> Tela 3
                         /_______/

    Para usar o Stack precisamos seguir um passo a passo importante:

        1º Criação de um Componente que terá um wrapper onde todas as telas serão comportadas;
```

```
        2º Dentro deste componente, teremos que importar o método "createNativeStackNavigator" da biblioteca
           "native-stack", conforme podemos ver nas linhas de cima. Para facilitar temos o costume de atribuir
           esse método a uma variável entitulada Stack;

        3º Criar um wrapper de navegação usando o componente "Navigator" a partir da variável Stack, é esse
           wrapper que irá comportar as telas;

        4º Todos os Componentes deverão ser passados por através de um componente "Screen", esse componente
           também é nativo da variável "Stack", os componentes podem ser passados tanto por através da propriedade
           "component", embutida no próprio componente Screen, quanto por através de wrapper, onde o componente
           Screen será o wrapper do componente passado (Nesse caso não poderemos ter a propriedade "component"
           declarada sobre o componente Screen);

        5º Todo esse componente, com suas devidas telas, deverá ser exportado para um componente externo que irá
            renderizar a nossa tela de navegação Stack. O componente externo deverá importar o componente
            "NavigationContainer" a partir da biblioteca "@react-navigation/native". Esse componente que irá
            comportar o nosso componente Stack.

        Veja um exemplo detalhado abaixo...

*/

//Atribuímoso método "createNativeStackNavigator" sobre uma constante intitulada "Stack", como é costume...
const Stack = createNativeStackNavigator()

export default props => {
    return (
        //Note que usamos a propriedade Navigator para gerar o wrapper de navegação dos screens...
        //Veja que usamos a propriedade "initialRouteName" para dar ao Navigator uma tela inicial, perceba que essa
          tela é referenciada pelo mesmo nome atribuído na propriedade "name" no componente Screen...
```

```
      //Veja que também temos a propriedade "ScreenOptions" ela traz uma série de subpropriedades que podemos
        estilizar, por exemplo, temos a "headerShown" que faz com que o cabeçalho do menu stack apareça ou não,
        por padrão é sempre "true"...
      <Stack.Navigator initialRouteName="TelaA" screenOptions={{headerShown: true}}>
          {//Abaixo temos o exemplo de uso de uma Screen, veja que estamos usando ela como wrapper...
           //Note que ela recebe a propriedade "name" que é responsável por dar um nome a cada Tela...
           //E também a propriedade options, que recebe uma série de subpropriedades que podem trazer informações
             sobre a Tela e até estilizar os botões do cabeçalho de cada tela...
          }
          {/* Eu poderia colocar dentro do componente "Stack.Screen" a propriedade "component" com o nome do
              componente que eu quero gerar, mas como aqui nós queremos envolver esse componente dentro de outro
              componente e aproveitar as propriedades internas de cada um, geramos um wrapper em volta do
              "Stack.Screen" */}
          <Stack.Screen
              name='TelaA'
              // component={TelaA} #OBS: não é possível fazer um wrapper e um component juntos!!!
              options={{title: 'Info'}}
              >
              {/* Veja que detalhe importante logo abaixo:
                  Note que invés de envolver o corpo de arrow function com chaves, nós a envolvemos com
                  parênteses, isso faz com que a arrow function retorne uma expressão literal (uma expressão vai
                  ser lida instântaneamente pelo código) em vez de uma função de corpo comum que precisará de ser
                  invocada.
                  E por que usamos uma função aqui invés de chamar o PassoStack diretamente?
                  Por que precisamos das propriedades de navegação que o props vai receber a partir do elemento
                  pai, que no caso é o "Stack.Navigation" que está neste mesmo arquivo. Pois caso você olhe no
                  arquivo do "PassoStack" vai perceber que ele precisa usar o método "navigate" para indicar para
                  onde a Tela A vai, por através do botão "Avançar". Esse método só pode ser alcançado a partir
                  desse arquivo em que estamos no momento.
              */}
              {props => (
```

```
                //A função do componente "PassoStack" é renderizar botões na tela quando necessário e indicar
                 para que tela os botões devem ir quando clicados...
                //Note que estamos passando para o componente todas as propriedades do Componente mãe, que irá
                 receber todos os atributos e métodos da variável Stack, isso é importante para que possamos
                 usar alguns métodos do Stack dentro de um componente externo...
                <PassoStack {...props} avancar="TelaB">
                    <TelaA />
                </PassoStack>
            )}
        </Stack.Screen>
        <Stack.Screen name='TelaB'>
            {props => (
                <PassoStack {...props} avancar="TelaC" voltar>
                    <TelaB />
                </PassoStack>
            )}
        </Stack.Screen>
        {//Note que na Tela C nós incluímos o componente por através da propriedade "component" só para
           exemplificar...
        }
        <Stack.Screen name='TelaC' component={TelaC} />
    </Stack.Navigator>
  )
}
```

ARQUIVO PASSOSTACK...

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


export default props => {

    //Note que mesmo aqui estamos usando os métodos "navigate" e "goBack" esses métodos são do componente Stack que retirou do método "createNativeStackNavigator", só conseguimos utilizar esses métodos aqui por que eles foram passados via "props" lá no componente Stack. Precisa ser assim para que o React Native entenda que esse componente e as suas Telas estão referenciando ao mesmo menu Stack.

    //O método navigate indica para que tela devemos ir...
    const avancar = tela => props.navigation.navigate(tela)

    //O método goBack indica que devemos voltar para a tela de onde viemos...
    const voltar = () => props.navigation.goBack()

    return (
        <View style={styles.flexInteiro}>
            <View>
                {/* Note que a função principal desse componente é incluir ou não botões de "avançar" e "voltar"
                    dependendo do que for passado como parâmetro. */}
                {props.voltar ? <Button title="Voltar" onPress={voltar} /> : false}
                {props.avancar ? <Button title="Avançar" onPress={() => avancar(props.avancar)} /> : false}
            </View>
            <View style={styles.flexInteiro}>
                {/* //Não se esqueça que "props.children" pode renderizar tudo oque estiver dentro de um componente
                      pai, no caso, onde quer que coloquemos o componente "PassoStack" e lá nós coloquemos qualquer
                      coisa dentro dele por através de um wrapper, essas coisas serão renderizadas dentro dessa
                      View. */}
```

```
                {props.children}
            </View>
        </View>
    )
}

const styles = StyleSheet.create({
    flexInteiro: {
        flex: 1,
    },
})
```

ARQUIVO TELAA...

```
import React from "react";
import PassoStack from "../componentes/PassoStack";
import TextoCentral from "../componentes/TextoCentral";

export default props => {
    return (
        <TextoCentral children={'TELA A'} corFundo="red" />
    )
}
```

ARQUIVO EXTERNO...

```
import React from "react";
import { SafeAreaView, StyleSheet } from "react-native";
//Aqui temos a importação do componente NavigationContainer...
import { NavigationContainer } from "@react-navigation/native";
import Stack from "./Stack";
```

```
export default props => {
    return (
        <SafeAreaView style={styles.safeArea}>
            {/* Note que o NavigationContainer comporta a nossa Stack... */}
            <NavigationContainer>
                <Stack />
            </NavigationContainer>
        </SafeAreaView>
    )
}

const styles = StyleSheet.create({
    safeArea: {
        flex: 1,
    },
})
```

RESULTADOS NO ANDROID...