

Tutorial CRUD em Android / iOS com React Native

João Ozorio

Tutorial CRUD em Android / iOS com React Native

Esse app será feito usando o React Native, para código e o Expo para interface gráfica. Vai ser um CRUD (Create, Read, Update, Delete) completo, já com as funcionalidades de listagem, cadastro, edição e exclusão de dados e também de armazenamento, utilizando a memória do dispositivo.

#1: Começando o projeto

Após o projeto estar pronto, entre na pasta, digitando “cd crud”, aproveite e já inicie o VS Code em seguida, pelo comando “code .”

#1: Começando o projeto

Antes de começar, vamos digitar o comando “`npm install @reactnavigation/native`”, adicionando o suporte à navegação por abas (tabs) no nosso app. Em seguida, se prepare, o comando é gigante.

“`expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view`”.

```
C:\Users\Aluno\Desktop\ozoriocrud10032002\crud>expo install react-native-gesture-handler react-native-reanimated  
react-native-screens react-native-safe-area-context @react-native-community/masked-view  
  
There is a new version of expo-cli available (5.3.0).  
You are currently using expo-cli 5.1.2  
Install expo-cli globally using the package manager of your choice;  
for example: `npm install -g expo-cli` to get the latest version  
  
Installing 4 SDK 44.0.0 compatible native modules and 1 other package using Yarn.  
> yarn add react-native-gesture-handler@~2.1.0 react-native-reanimated@~2.3.1 react-native-screens@~3.10.1 react-native-safe-area-context@3.3.2 @react-native-community/masked-view  
yarn add v1.22.17  
warning package-lock.json found. Your project contains lock files generated by tools other than Yarn. It is advised not to mix package managers in order to avoid resolution inconsistencies caused by unsynchronized lock files.  
To clear this warning, remove package-lock.json.  
[1/4] Resolving packages...  
[2/4] Fetching packages...  
[3/4] Linking dependencies...  
[4/4] Building fresh packages...  
success Saved lockfile.  
success Saved 11 new dependencies.  
info Direct dependencies  
└── @react-native-community/masked-view@0.1.11  
    ├── react-native-gesture-handler@2.1.3  
    ├── react-native-reanimated@2.3.3  
    ├── react-native-safe-area-context@3.3.2  
    └── react-native-screens@3.10.2  
info All dependencies  
└── @egjs/hammerjs@2.0.17  
    ├── @react-native-community/masked-view@0.1.11  
    ├── @types/hammerjs@2.0.41  
    ├── @types/invariant@2.2.35  
    ├── hoist-non-react-statics@3.3.2  
    ├── mockdate@3.0.5  
    ├── react-native-gesture-handler@2.1.3  
    ├── react-native-reanimated@2.3.3  
    ├── react-native-safe-area-context@3.3.2  
    └── react-native-screens@3.10.2  
    string-hash-64@1.0.3  
Done in 24.49s.
```

#1: Começando o projeto

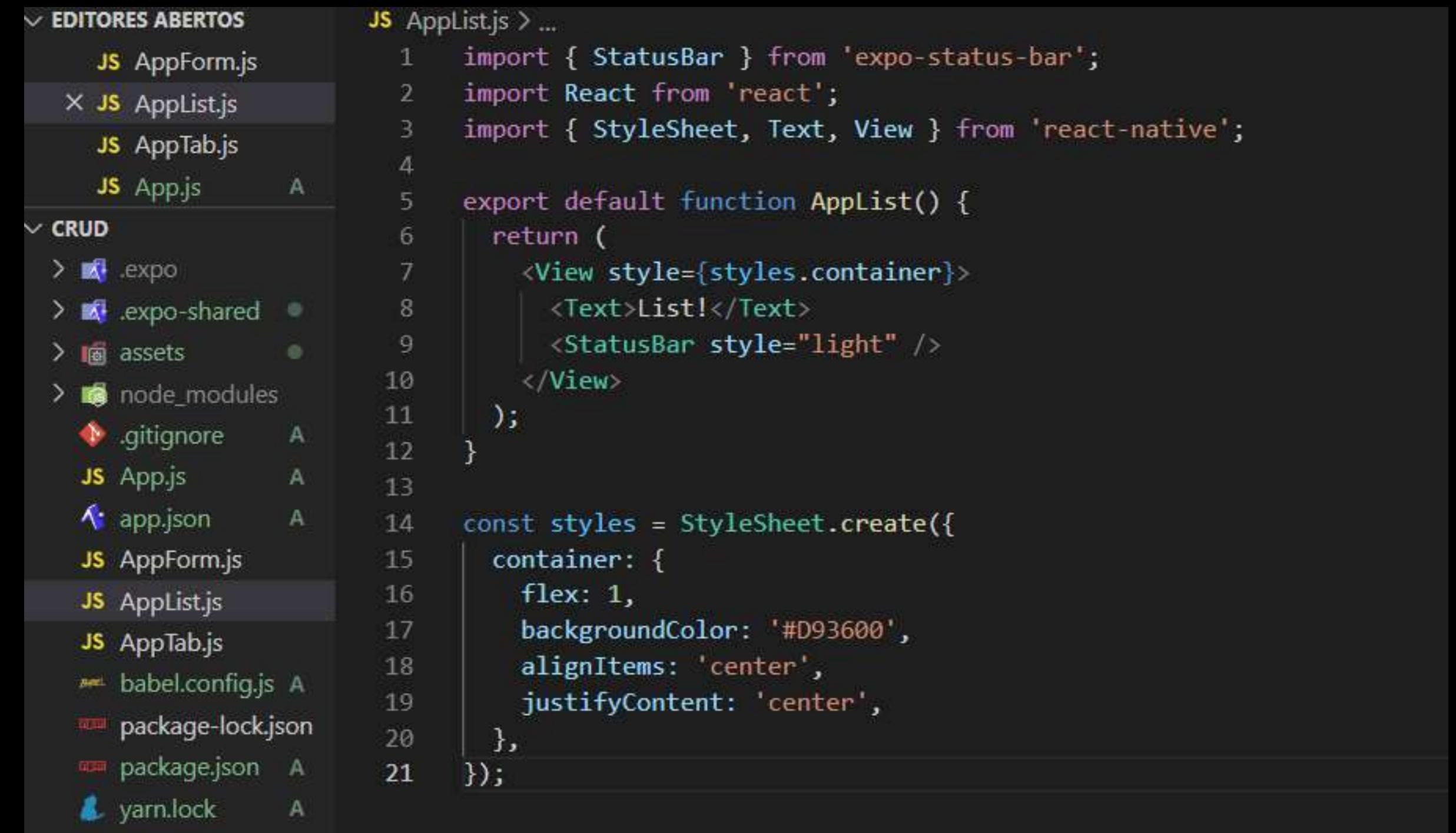
E por fim, vamos adicionar um último módulo, específico para navegação por abas.
“npm install @react-navigation/bottom-tabs”.

#1: Começando o projeto

Agora com o VS Code aberto, vamos criar um novo arquivo na pasta “crud”, chamado de “AppForm.js”.
Preste atenção ao código aberto nessa aba:

#1: Começando o projeto

Vamos criar também o arquivo “AppList.js”.
Preste atenção ao código.

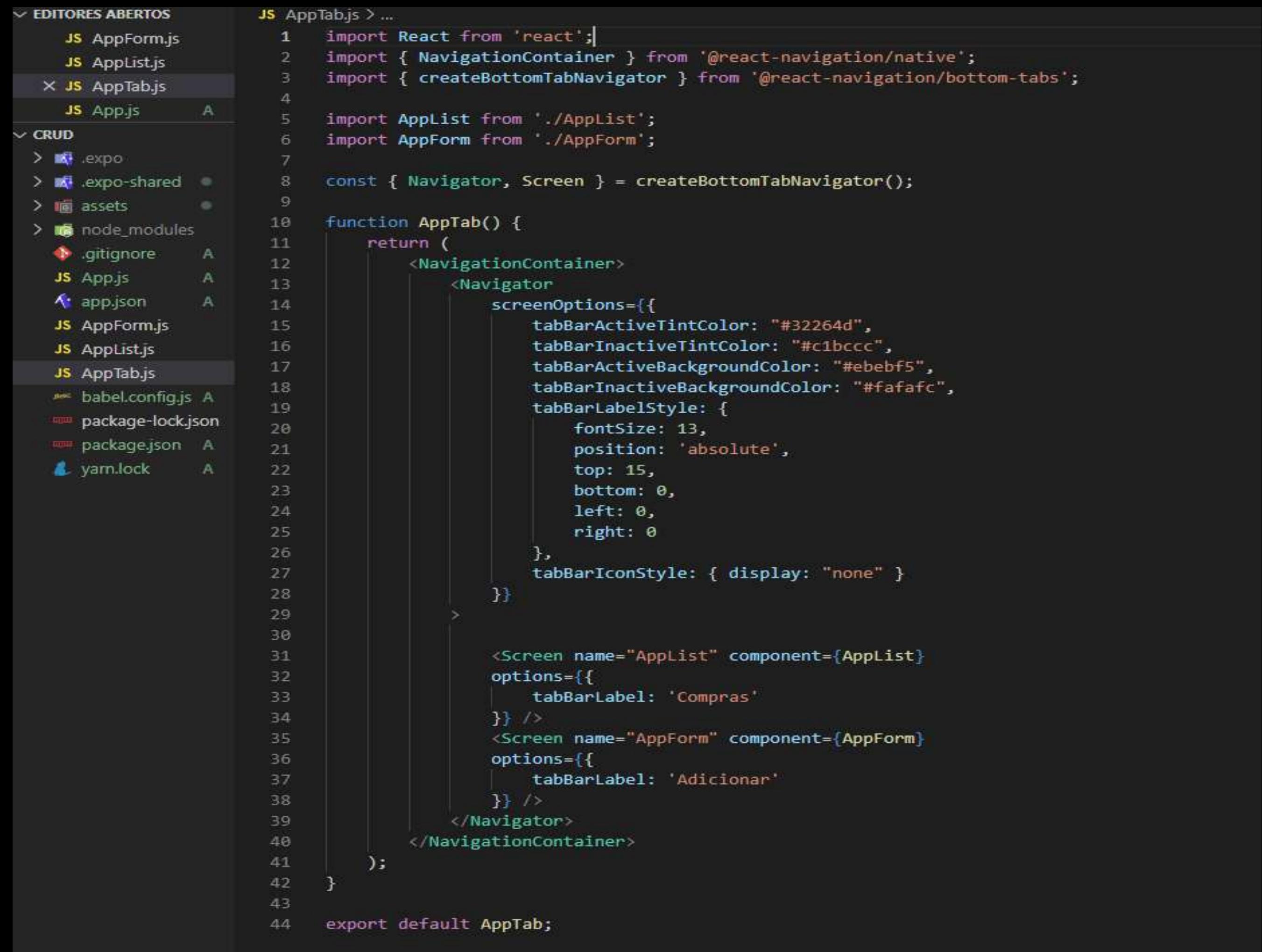


The image shows a code editor interface with a sidebar on the left displaying a file tree and a main panel on the right showing the code for `AppList.js`. The sidebar lists files under two categories: 'EDITORES ABERTOS' and 'CRUD'. In the 'EDITORES ABERTOS' category, `AppForm.js`, `AppList.js`, `AppTab.js`, and `App.js` are listed. In the 'CRUD' category, there are several files: `.expo`, `.expo-shared`, `assets`, `node_modules`, `.gitignore`, `App.js`, `app.json`, `AppForm.js`, `AppList.js` (which is currently selected), `AppTab.js`, `babel.config.js`, `package-lock.json`, `package.json`, and `yarn.lock`. The main panel contains the following code:

```
JS AppList.js > ...
1 import { StatusBar } from 'expo-status-bar';
2 import React from 'react';
3 import { StyleSheet, Text, View } from 'react-native';
4
5 export default function AppList() {
6   return (
7     <View style={styles.container}>
8       <Text>List!</Text>
9       <StatusBar style="light" />
10    </View>
11  );
12}
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#D93600',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21});
```

#1: Começando o projeto

Vamos criar também o arquivo “AppTab.js”.
Preste atenção ao código.



The image shows a code editor interface with two panes. The left pane displays a file tree with the following structure:

- EDITORES ABERTOS:
 - JS AppForm.js
 - JS AppList.js
 - X JS AppTab.js
 - JS App.js A
- CRUD:
 - > .expo
 - > .expo-shared
 - > assets
 - > node_modules
 - > .gitignore A
 - JS App.js A
 - app.json A
 - JS AppForm.js
 - JS AppList.js
 - JS AppTab.js
 - babel.config.js A
 - package-lock.json
 - package.json A
 - yarn.lock A

The right pane shows the content of the AppTab.js file:

```
JS AppTab.js > ...
1 import React from 'react';
2 import { NavigationContainer } from '@react-navigation/native';
3 import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
4
5 import AppList from './AppList';
6 import AppForm from './AppForm';
7
8 const { Navigator, Screen } = createBottomTabNavigator();
9
10 function AppTab() {
11   return (
12     <NavigationContainer>
13       <Navigator
14         screenOptions={{
15           tabBarActiveTintColor: "#32264d",
16           tabBarInactiveTintColor: "#c1bccc",
17           tabBarActiveBackgroundColor: "#ebef5",
18           tabBarInactiveBackgroundColor: "#fafafc",
19           tabBarLabelStyle: {
20             fontSize: 13,
21             position: 'absolute',
22             top: 15,
23             bottom: 0,
24             left: 0,
25             right: 0
26           },
27           tabBarIconStyle: { display: "none" }
28         }}
29       >
30         <Screen name="AppList" component={AppList}
31           options={{
32             tabBarLabel: 'Compras'
33           }} />
34         <Screen name="AppForm" component={AppForm}
35           options={{
36             tabBarLabel: 'Adicionar'
37           }} />
38       </Navigator>
39     </NavigationContainer>
40   );
41 }
42
43 export default AppTab;
```

#1: Começando o projeto

E o que cada propriedade significa?

`tabBarActiveTintColor`: cor da fonte quando aba selecionada;

`tabBarInactiveTintColor`: cor da fonte quando aba não-selecionada;

`tabBarActiveBackgroundColor`: cor de fundo da aba quando selecionada;

`tabBarInactiveBackgroundColor`: cor de fundo da aba quando não-selecionada;

`tabBarLabelStyle`: estilos da label da aba;

`tabBarIconStyle`: estilos do ícone da aba;

#1: Começando o projeto

Agora dentro do arquivo “AppForm.js”,
vamos estilizar com esse código:

#1: Começando o projeto

O nosso app deverá ser como a printscreen ao lado:
Enfim, agora que temos o “chassi”, vamos alterar as
legendas. Esse será um CRUD de compras, abordando
todas as características e funcionalidades necessárias
para o nosso app e o aprendizado.



#1: Começando o projeto

Tá, já que adicionamos a estilização CSS para nosso código, o que cada propriedade significar? Jovem pupilo, aqui estão as definições:

container: é o estilo da tela como um todo. Aqui definimos cor de fundo e alinhamento geral;

title: é o estilo do título da tela. Definimos cor da fonte, tamanho, peso e margem;

inputContainer: é o estilo do formulário como um todo. Definimos margem, largura, margem interna, arredondamento das bordas, cor de fundo e com ‘stretch’ dizemos que os inputs vão ocupar todo o espaço do formulário;

input: é o estilo dos campos de texto. Definimos margem, altura, cor de fundo, arredondamento da borda, margem interna e tamanho da fonte;

button: é o estilo do botão. Definimos margem, cor de fundo, altura, arredondamento dos cantos, margem interna, tamanho da fonte, alinhamento e elevação/sombra (para dar profundidade);

buttonText: é o estilo do texto do botão. Apenas cor da fonte e peso aqui;

Fim da primeira parte da jornada.

Com o fim de uma parte das cinco nessa jornada, concluímos que essas propriedades e algumas funções do código lhe ajudarão a ganhar familiaridade no desenvolvimento de qualquer projeto.

Não desista, você precisa descansar sua mente, treinar sua alma para se alinhar.
Parabéns pela conclusão da parte 1 de 5.

Que a força esteja com você.