



# Desenvolvimento para Dispositivos Móveis

## React Native

parte 2



# App inicial

Componentes  
Funcionais  
Classe

Estados

# App inicial

The image shows a development environment with Visual Studio Code on the left and a browser window on the right.

**Visual Studio Code:**

- EXPLORER:** Shows the project structure for `DS151_NOTAS`. The `src/components/Form` directory is expanded, showing `index.js` and `validationCalc`.
- Code Editor:** Displays the `index.js` file in the `Form` component. The code includes state management for two notes and a calculator function.

```
export default function Form() {  
  const [nota1, setNota1] = useState(null)  
  const [nota2, setNota2] = useState(null)  
  const [messageCalc, setMessageCalc] = useState("preencha as notas")  
  const [calc, setCalc] = useState(null)  
  const [textButton, setTextButton] = useState("Calcular")  
  
  function calcCalculator() {  
    return setCalc(((nota1+nota2)/2).toFixed(2))  
  }  
  
  function validationCalc() {  
    if(nota1 !== null && nota2 !== null) {  
      calcCalculator()  
      setMessageCalc("sua media eh igual: ")  
      return  
    }  
    setCalc(null)  
    setTextButton("Calcular")  
  }  
}
```

**Browser Window:**

- Address Bar:** `localhost:19006`
- Page Content:** Displays the initial state of the application with the text "Media do Aluno" and two input fields for "Nota 1" and "Nota 2", both containing the value "90". A blue button labeled "CALCULAR MEDIA" is visible.
- Footer:** The text "preencha as notas!" is displayed at the bottom of the page.

**Terminal:**

```
web compiled with 1 error  
Web Bundling complete 307ms  
web compiled successfully
```



## Para hoje

Relembrar funcionamento dos estados

Hook e State Hook

Estados com listas (FlatList)

Recursos de entrada (TextInput, placeholder, onChangeText)



## Hooks

Hooks é o termo para tratarmos estados utilizando componentes funcionais.

Para classes os estados eram variáveis de objeto mantidas durante renderizações

Se preciso que alguma informação (estado visual ou funcional) preciso armazenar como estado



# Hooks

O estado tem um link com a renderização de um componente.

State Hook: Atualizou redesenha o componente

3 perguntas ao se criar um Estado

- Qual dado do componente será alterado?
- Qual é o tipo deste dado
- Qual o valor inicial deste dados

# Hooks

Já utilizamos!

```
src > components > Form > JS index.js > Form > validationCalc
1  import React, {useState} from "react"
2  import {View} from "react-native"
3  import {Text, TextInput, Button } from "react-native-web";
4  import ResultCalc from "../ResultCalc";
5
6  export default function Form(){
7
8      const [nota1, setNota1] = useState(null)
9      const [nota2, setNota2] = useState(null)
10     const [messageCalc, setMessageCalc] = useState("preencha as notas!")
11     const [calc, setCalc] = useState(null)
12     const [textButton, setTextButton] = useState("Calcular")
13
```



## Navegação entre telas

```
npm install @react-navigation/native
```

```
npm install @react-navigation/stack
```

É necessário importar pacotes de navegação

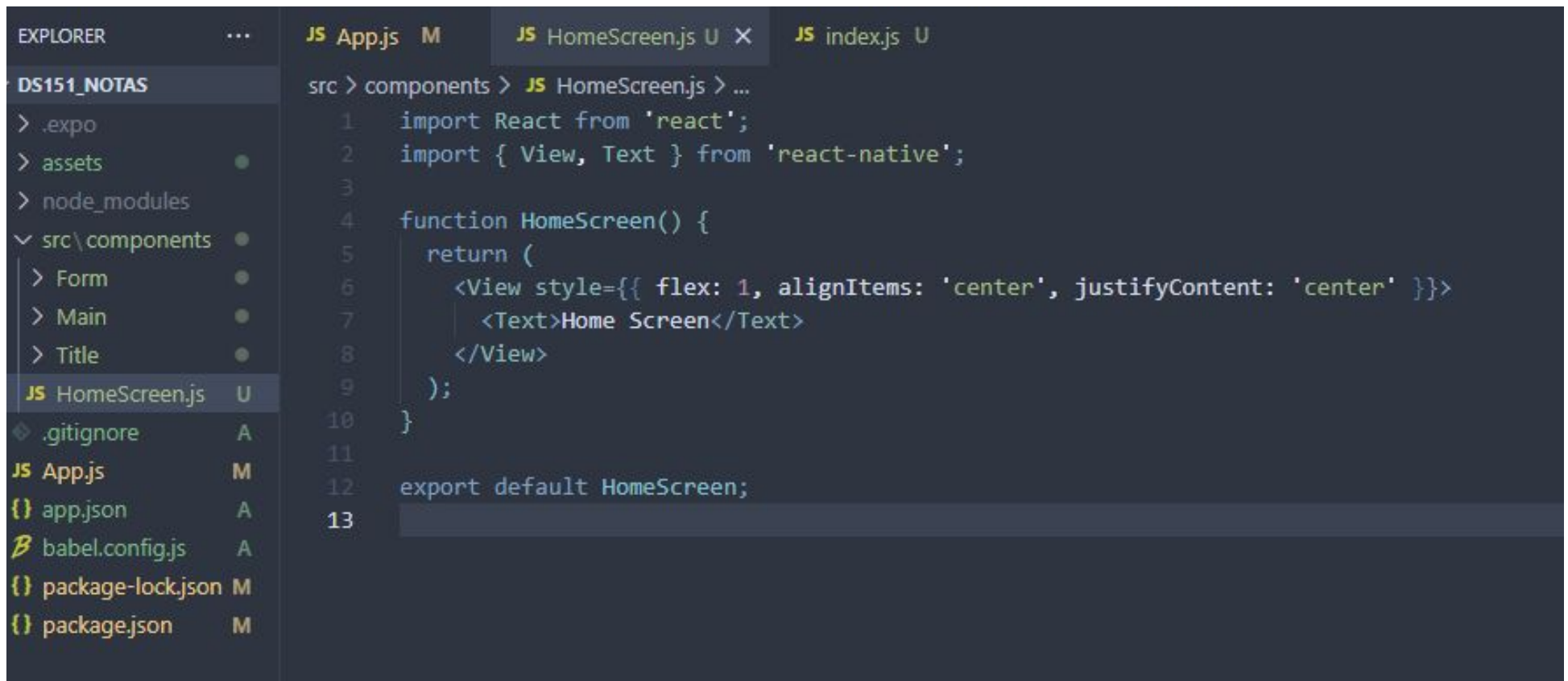
Vamos utilizar o mesmo projeto da aula passada



# Navegação entre telas

```
JS App.js > ...
1 import React from 'react';
2 import { NavigationContainer } from '@react-navigation/native';
3 import { createStackNavigator } from '@react-navigation/stack';
4 import HomeScreen from '../src/components/HomeScreen';
5
6 const Stack = createStackNavigator();
7
8 function App() {
9   return (
10     <NavigationContainer>
11       <Stack.Navigator>
12         <Stack.Screen name="Home" component={HomeScreen} />
13       </Stack.Navigator>
14     </NavigationContainer>
15   );
16 }
17
18 export default App;
```

# Navegação entre telas



The screenshot shows a code editor with three tabs: `JS App.js`, `JS HomeScreen.js`, and `JS index.js`. The `HomeScreen.js` tab is active, displaying the following code:

```
src > components > JS HomeScreen.js > ...
1  import React from 'react';
2  import { View, Text } from 'react-native';
3
4  function HomeScreen() {
5    return (
6      <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
7        <Text>Home Screen</Text>
8      </View>
9    );
10 }
11
12 export default HomeScreen;
13
```

The Explorer panel on the left shows the project structure for `DS151_NOTAS`:

- `> .expo`
- `> assets`
- `> node_modules`
- `▼ src\components`
  - `> Form`
  - `> Main`
  - `> Title`
  - `JS HomeScreen.js` (selected)
- `◆ .gitignore`
- `JS App.js`
- `{ } app.json`
- `B babel.config.js`
- `{ } package-lock.json`
- `{ } package.json`

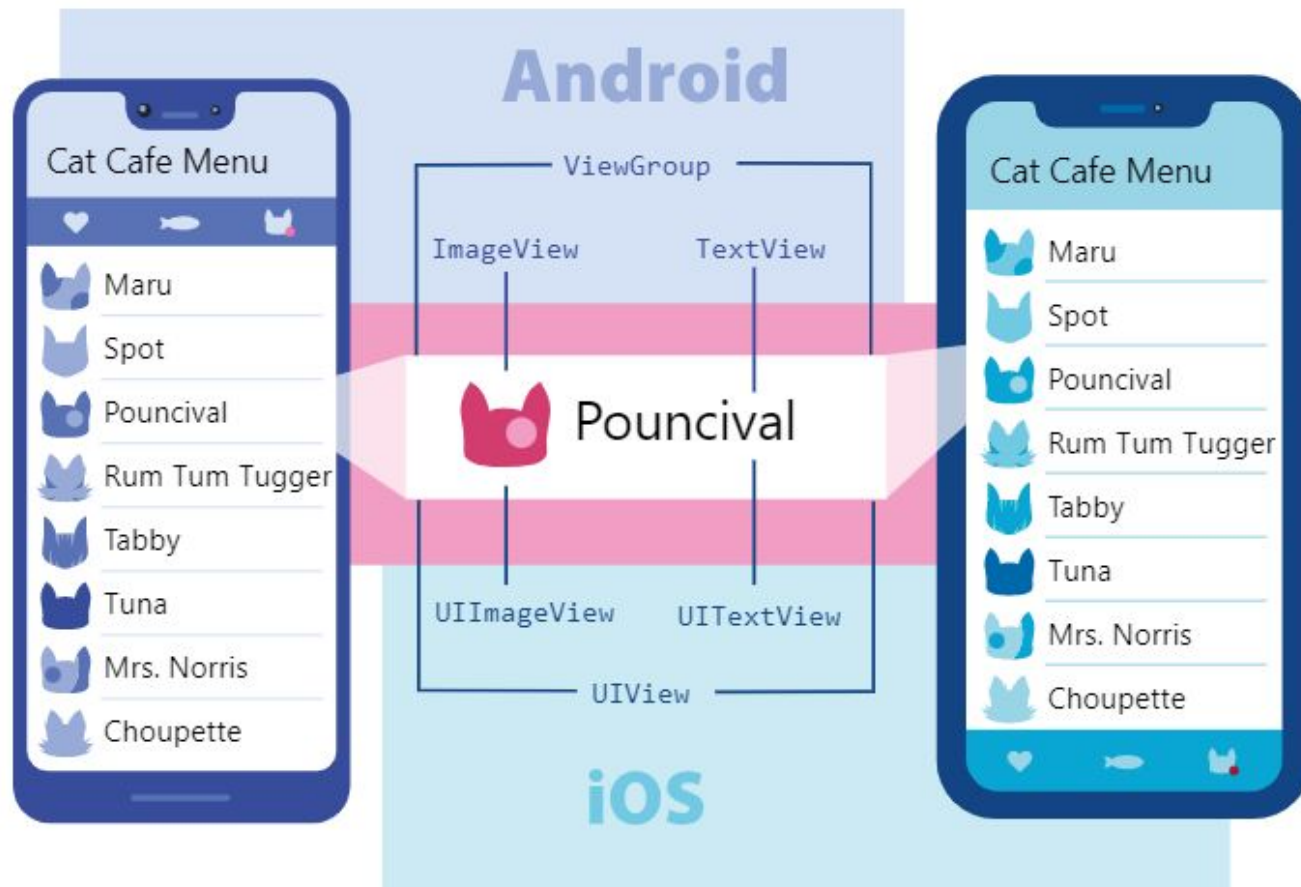


# Navegação entre telas

Assim como na aula passada vamos seguir basicamente este formato para componentes:

- importa o react
- importa os componentes do RN que irá utilizar
- cria o componente funcional
- exporta por padrão seu elemento

# Navegação entre telas



Just a sampling of the many views used in Android and iOS apps.

# Navegação entre telas

A parte visual pode ser definida utilizando estilos próximos de CSS

```
3
4  function HomeScreen() {
5    return (
6      <View style={style.mainView }>
7        <Text>Home Screen</Text>
8      </View>
9    );
10  }
11
12  export default HomeScreen;
13
14  const style = StyleSheet.create({
15    mainView:{
16      flex: 1,
17      alignItems: 'center',
18      justifyContent: 'center',
19    }
20  })
```



# Navegação entre telas

Criando um novo componente

## **Boilerplate**

- Importações
- Componente
- Estilo
- Exportação/Renderização



# Navegação entre telas

Criando novo componente

**Flatlist:** Insere lista de elementos

Sempre importamos os elementos de RN:

```
import React from 'react';
```

```
import {SafeAreaView, View, FlatList, StyleSheet, Text,  
StatusBar,} from 'react-native';
```

## Navegação entre telas

Criamos um array com os dados e um elemento para apresentar os dados

```
3
4  const DATA = [
5    {id: '01', title: 'First Item'},
6    {id: '02', title: 'Second Item'},
7    {id: '03', title: 'Third Item'},
8  ];
9
10 const Item = ({title}) => (
11   <View style={styles.item}>
12     <Text style={styles.title}>{title}</Text>
13   </View>
14 );
15
```



# Navegação entre telas

Os estilos seguem a base para qualquer elemento gráfico

```
27
28 const styles = StyleSheet.create({
29   container: {
30     flex: 1,
31     marginTop: StatusBar.currentHeight || 0,
32   },
33   item: {
34     backgroundColor: '#f9c2ff',
35     padding: 20,
36     marginVertical: 8,
37     marginHorizontal: 16,
38   },
39   title: {
40     fontSize: 32,
41   },
42 });
43
```

# Navegação entre telas

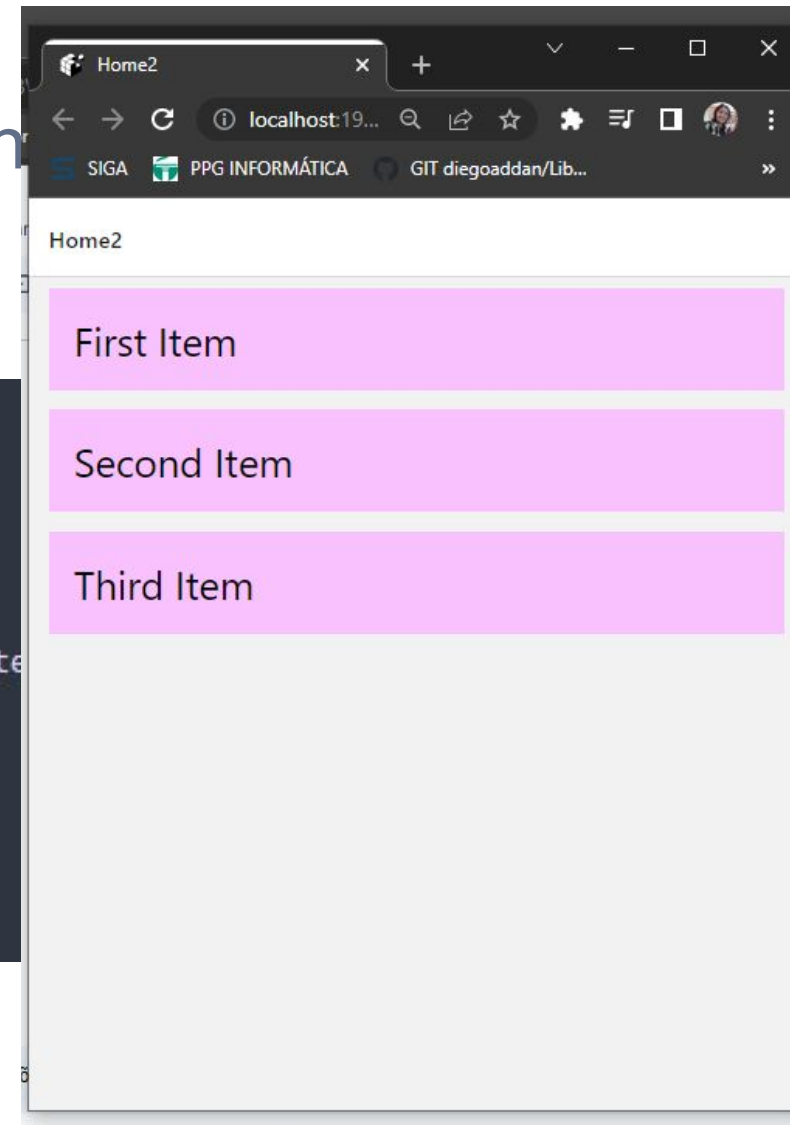
Juntamos o item para apresentar os dados através do componente funcional

```
15
16 const Home2 = () => {
17   return (
18     <SafeAreaView style={styles.container}>
19       <FlatList
20         data={DATA}
21         renderItem={({item}) => <Item title={item.title} />}
22         keyExtractor={item => item.id}
23       />
24     </SafeAreaView>
25   );
26 };
27
```

# Navegação entre telas

Juntamos o item para apresent  
do componente funcional

```
15  
16 const Home2 = () => {  
17   return (  
18     <SafeAreaView style={styles.container}>  
19       <FlatList  
20         data={DATA}  
21         renderItem={({item}) => <Item title={ite  
22         keyExtractor={item => item.id}  
23       />  
24     </SafeAreaView>  
25   );  
26 };  
27
```





## Navegação entre telas

Conseguimos então criar uma lista de conteúdo em uma página e criar componentes que agregam estes elementos.

Importante: Lista de elementos precisam de **key**

Com a FlatList podemos alimentar uma página de um app com conteúdos externos

# Navegação entre telas

Podemos criar uma segunda página agora!

```
src > screens > JS TopicsScreen.js > ...  
1  ∨ import React from 'react';  
2    import {View, Text, StyleSheet} from 'react-native';  
3  
4  ∨ const TopicsScreen = () => {  
5      return(  
6          <Text>Deu certo!</Text>  
7      )  
8  }  
9  
10 ∨ const styles = StyleSheet.create({  
11  
12  });
```

# Navegação entre telas

Referencio minhas páginas na Stack do App

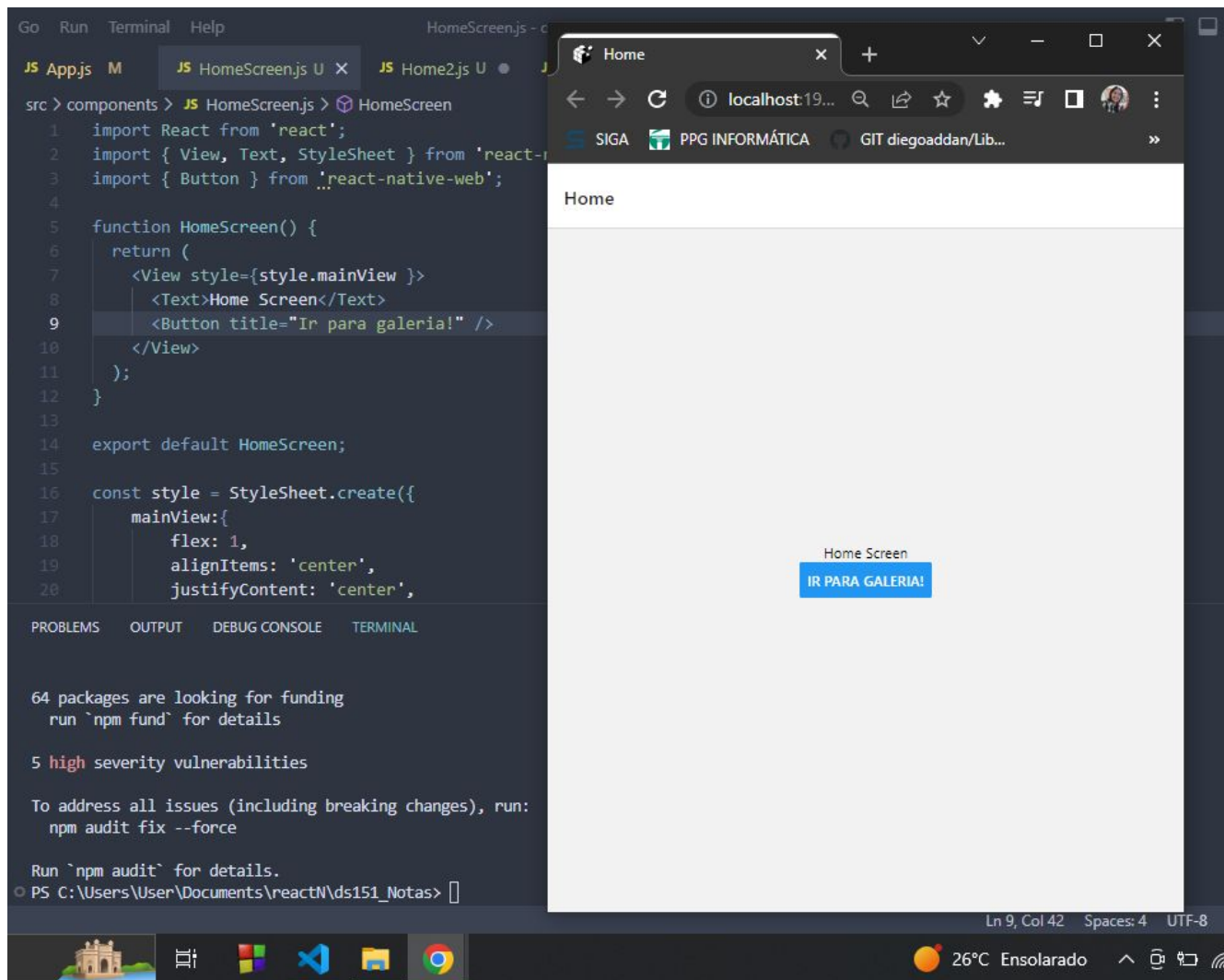
```
JS App.js > App
import React from 'react';
import {NavigationContainer} from '@react-navigation/native';
import {createStackNavigator} from '@react-navigation/stack';
import HomeScreen from './src/components/HomeScreen';
import Home2 from './src/components/Home2';

const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Home2" component={Home2} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

# Navegação entre telas

Crio um botão (servirá para navegar)



The image shows a development environment with a code editor on the left and a web browser on the right. The code editor displays the `HomeScreen.js` file with the following code:

```
1 import React from 'react';
2 import { View, Text, StyleSheet } from 'react-native';
3 import { Button } from 'react-native-web';
4
5 function HomeScreen() {
6   return (
7     <View style={style.mainView}>
8       <Text>Home Screen</Text>
9       <Button title="Ir para galeria!" />
10    </View>
11  );
12 }
13
14 export default HomeScreen;
15
16 const style = StyleSheet.create({
17   mainView: {
18     flex: 1,
19     alignItems: 'center',
20     justifyContent: 'center',
```

The web browser shows the rendered application at `localhost:19...`. It displays the text "Home Screen" and a blue button labeled "IR PARA GALERIA!".

At the bottom of the code editor, there is a terminal window showing the following output:

```
64 packages are looking for funding
  run `npm fund` for details

5 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\User\Documents\reactN\ds151_Notas>
```





## Navegação entre telas

Usaremos um evento (onPress), como na última aplicação.

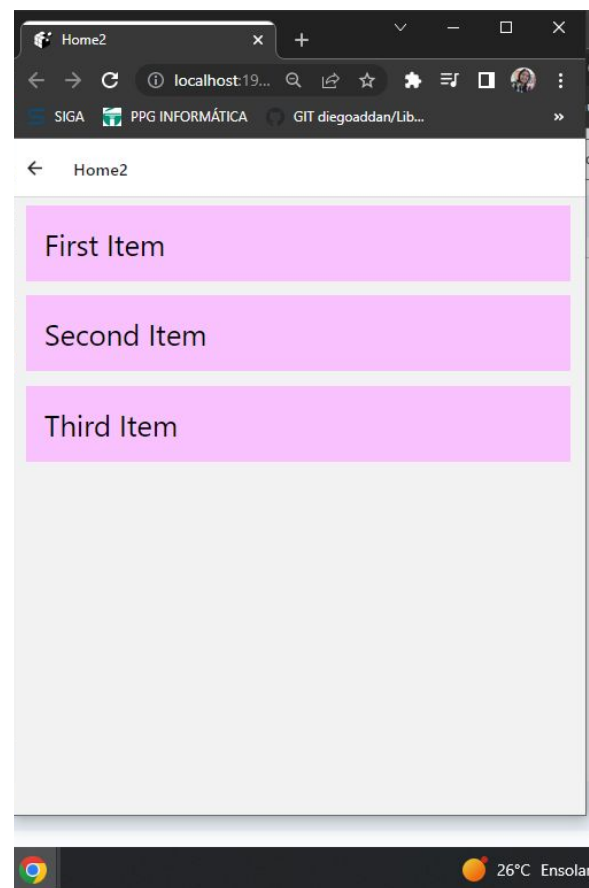
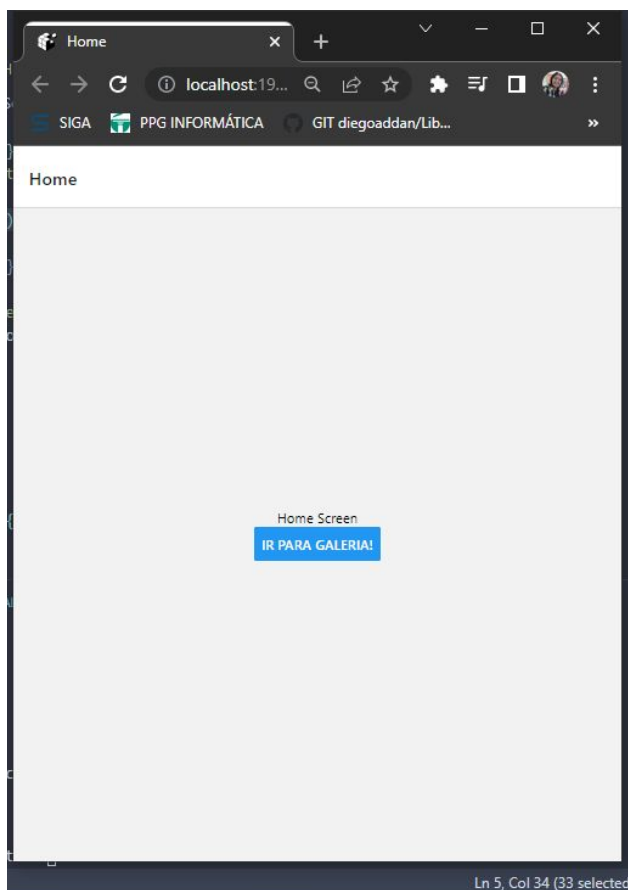
```
<Button title="Ir para galeria!"  
        onPress={ () => navigation.navigate('Home2')}  
/>
```

Junto do parametro navigation na função

```
function HomeScreen({navigation})
```



# Navegação entre telas





## Navegação entre telas

O Stack cuida da renderização em pilha

Parte do processo de composição dos elementos é feito de forma dinâmica. Explorem os logs do expo!, terminal da IDE, etc.

Assim como nossa FlatList utilizou um array é possível inserir outros tipos de dados de forma simples (imagem, dataset)



## Controle de Navegação

Podemos adicionar várias páginas e botões para navegação neste momento!

Adicione o formulário da última aula (main~form)

# Controle de Navegação

Vamos criar um novo componente. Um contador

Crie um arquivo **counterScreen.js**

```
src > components > JS counterScreen.js > [e] CounterScreen
1  import React from "react";
2  import { View, Button, Text, StyleSheet } from "react-native";
3
4  const CounterScreen = () => {
5    return(
6      <View>
7        <Button
8          title="Incrementa"
9        />
10       <Text> Teste contador </Text>
11     </View>
12   )
13 }
14
```



## Controle de Navegação

Qual dado do componente será alterado?

Qual o tipo deste dado?

Qual o valor inicial deste dado?



## Controle de Navegação

Qual dado do componente será alterado?

Um número inteiro com valor incremental

Qual o tipo deste dado?

valor numérico

Qual o valor inicial deste dado?

sem interação - Zero

# Controle de Navegação

Precisamos então criar um Hook para nosso botão

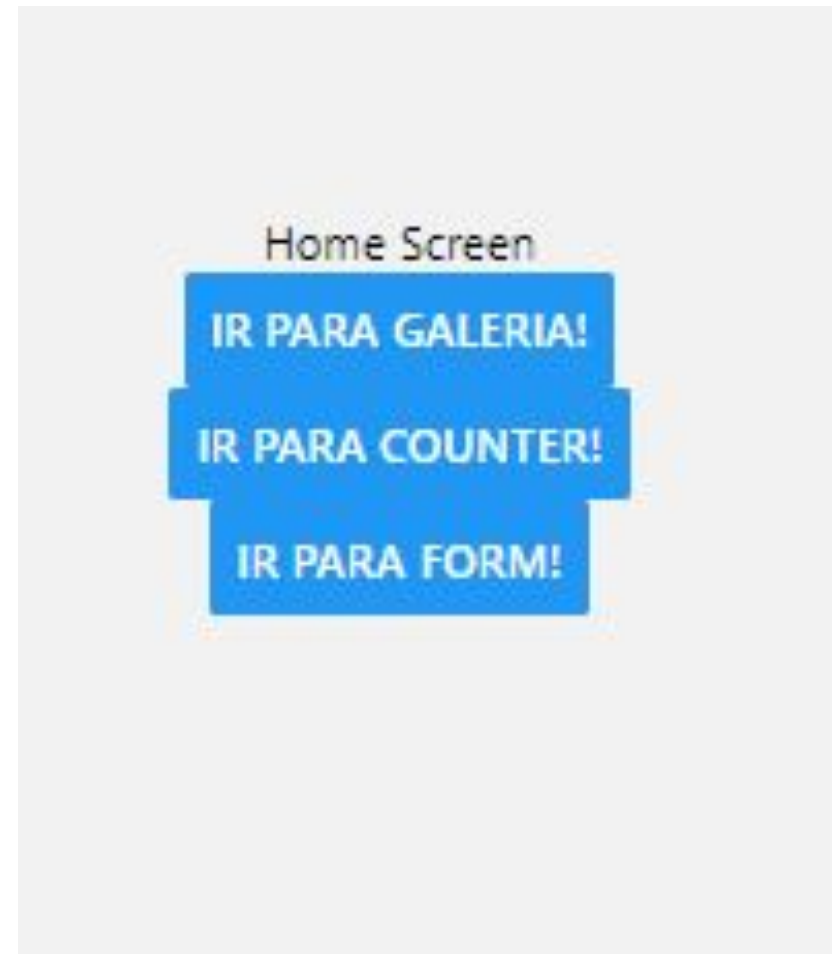
```
4
5  const CounterScreen = () => {
6    const [counter, setCounter] = useState(0);
7    return(
8      <View>
9        <Button
10          title="Incrementa"
11          onPress={() => setCounter(counter+1)}
12        />
13        <Text> {counter} </Text>
14      </View>
15    )
16  }
```

# Controle de Navegação

Neste momento então temos quatro telas: **Home**, **FlatList**, **Form** e **Counter**

Desta forma vimos:

- Cálculo em eventos
- Elementos visuais
- Elementos iterativos





# Controle de Navegação

Vamos criar mais uma página **ColorList**

```
src > components > JS ColorList.js > [e] default
1  import React, {useState} from "react";
2  import { View, Button, Text, StyleSheet } from "react-native-web";
3
4
5  function ColorList(){
6    return(
7      <View>
8        <Text> Teste</Text>
9      </View>
10    )
11  }
12
13
14  export default ColorList;
```



## Controle de Navegação

Inclua o import de ColorList no App.js e faça a chamada na Stack.

Inclua também na homepage com um botão de navegação.



# Controle de Navegação

ColorList

Qual dado do componente será alterado?

Lista com Formas

Qual o tipo deste dado?

Array de Obj

Qual o valor inicial deste dado?

sem interação - Vazio

# Controle de Navegação

Adiciona o array de cores

função de gerar RGB aleatório

```
4
5  const ColorList = () => {
6
7    // ['rgb(255,255,255)', 'rgb(122,122,122)', ...]
8    const [colors, setColors] = useState([]);
9
10   function addColor() {
11     const r = Math.floor(Math.random() * 255);
12     const g = Math.floor(Math.random() * 255);
13     const b = Math.floor(Math.random() * 255);
14     setColors(prevColors => [...prevColors, `rgb(${r},${g},${b})`]);
15   }
16 }
```

# Controle de Navegação

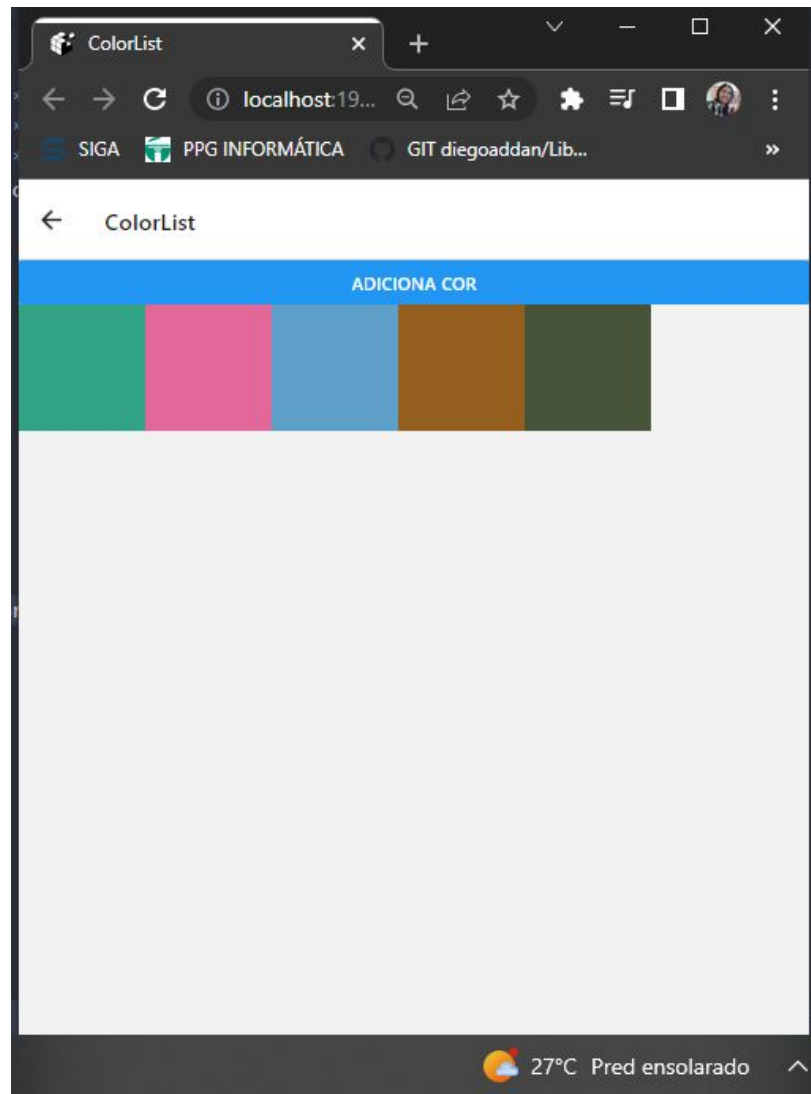
Adiciona o botão com a função de estado

```
<Button title="Adiciona Cor" onPress={() => addColor()} />
```

Adiciona a FlatList

```
<FlatList
  data={colors}
  keyExtractor={item => item}
  renderItem={({ item }) =>
    <View style={{
      backgroundColor: item,
      width: 100,
      height: 100
    }}></View>}</>
```

# Controle de Navegação





# Resumo

Temos até agora:

Navegação

Hooks

Estilos

Comportamento de botões

Componentes Funcionais



## Exercício

Criar repositório de projetos semanais

clonar seu projeto desta aula

- sistema de notas
- navegação
- FlatList, sistema Contador

Vamos prosseguir sempre alterando este repositório