

Aula 5 – React Native

PROFESSOR: HARLEY MACÊDO DE MELLO

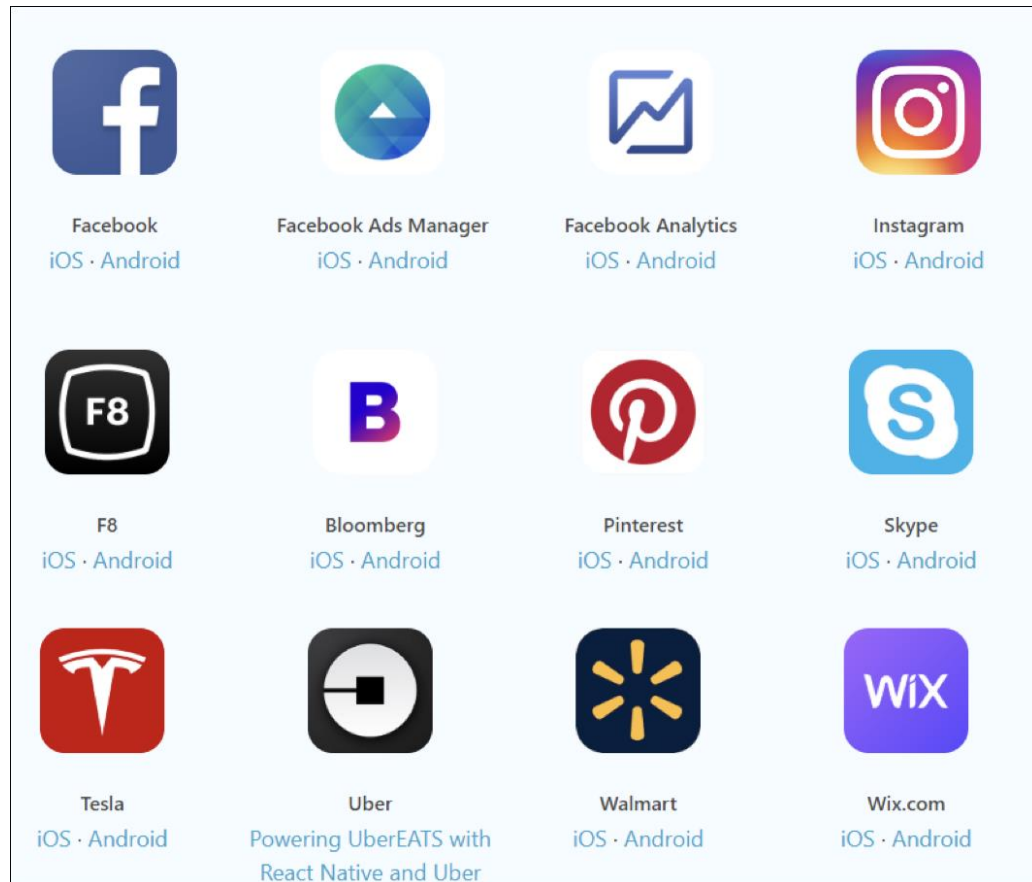
Roteiro

- Conceitos
- Ambiente Expo
- Componentes
- Propriedade e estado
- Eventos
- Estilização
- Navegação
- React Navigation
- Projeto completo

Conceitos

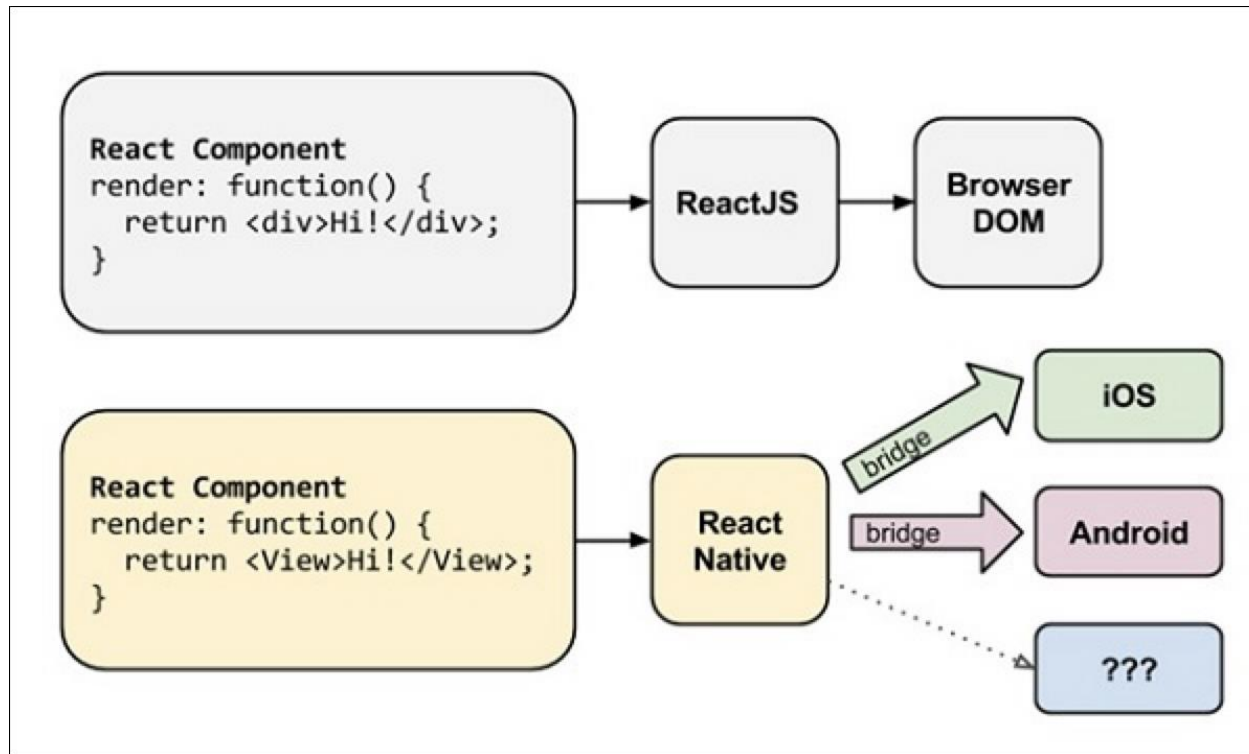
- Necessidade de ter Apps em mais de um S.O.
- Agilizar o desenvolvimento
- Padronizar os Apps
- Trabalhar com base em componentes reutilizáveis
- Gera uma aplicação nativa, não apenas uma simulação

Conceitos



Algumas empresas que usam o React Native para desenvolver seus Apps.

Conceitos



Assim como no ReactJS, o React Native trabalha com componentização e geração de conteúdo nativo.

Conceitos



Componentes reutilizáveis formados à partir de core componentes.

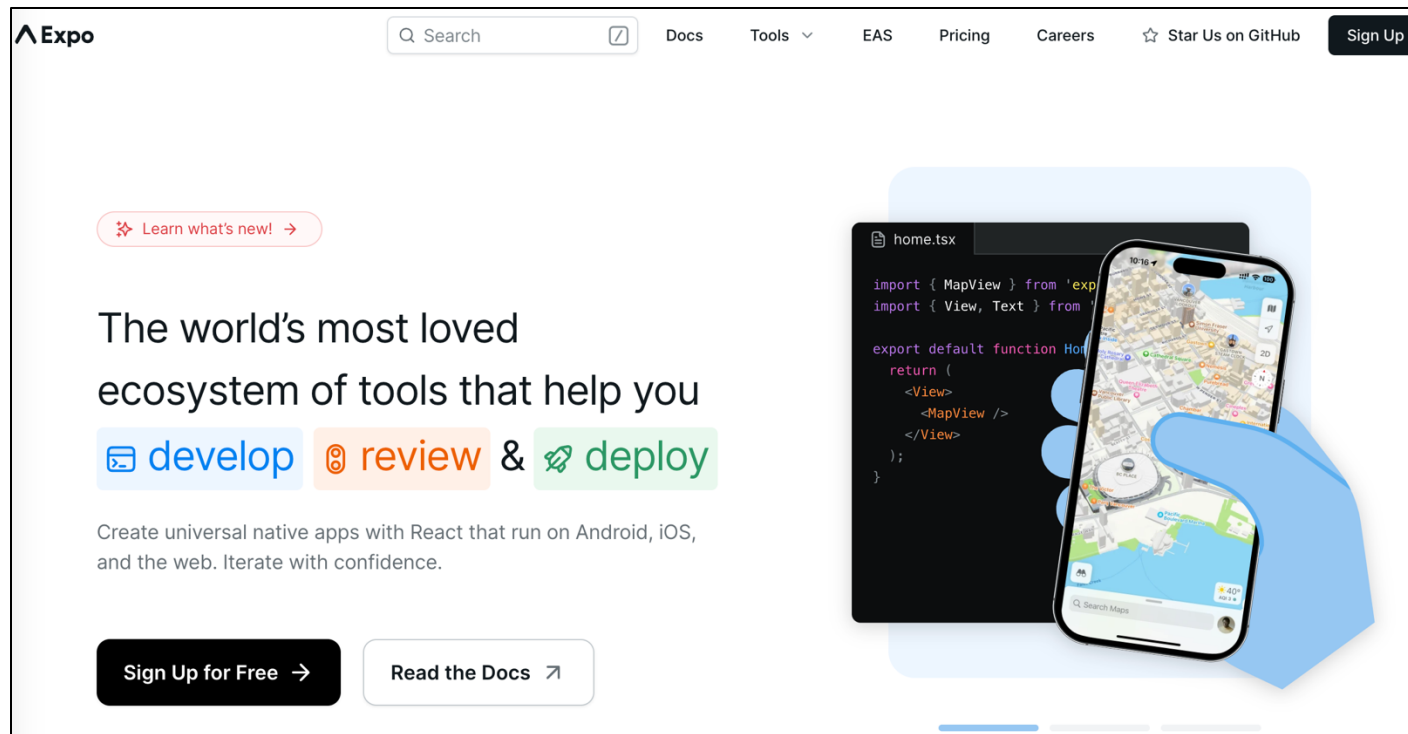
Ambiente Expo

- O Expo é uma plataforma para desenvolvimento de React Native
- Possui diversas ferramentas e serviços para auxiliar o desenvolvedor
- Ajuda nos processos de desenvolvimento, build e deploy
- Abstrai a configuração inicial pesada
- Possui bibliotecas próprias, como a de câmera e de mapa

Ambiente Expo

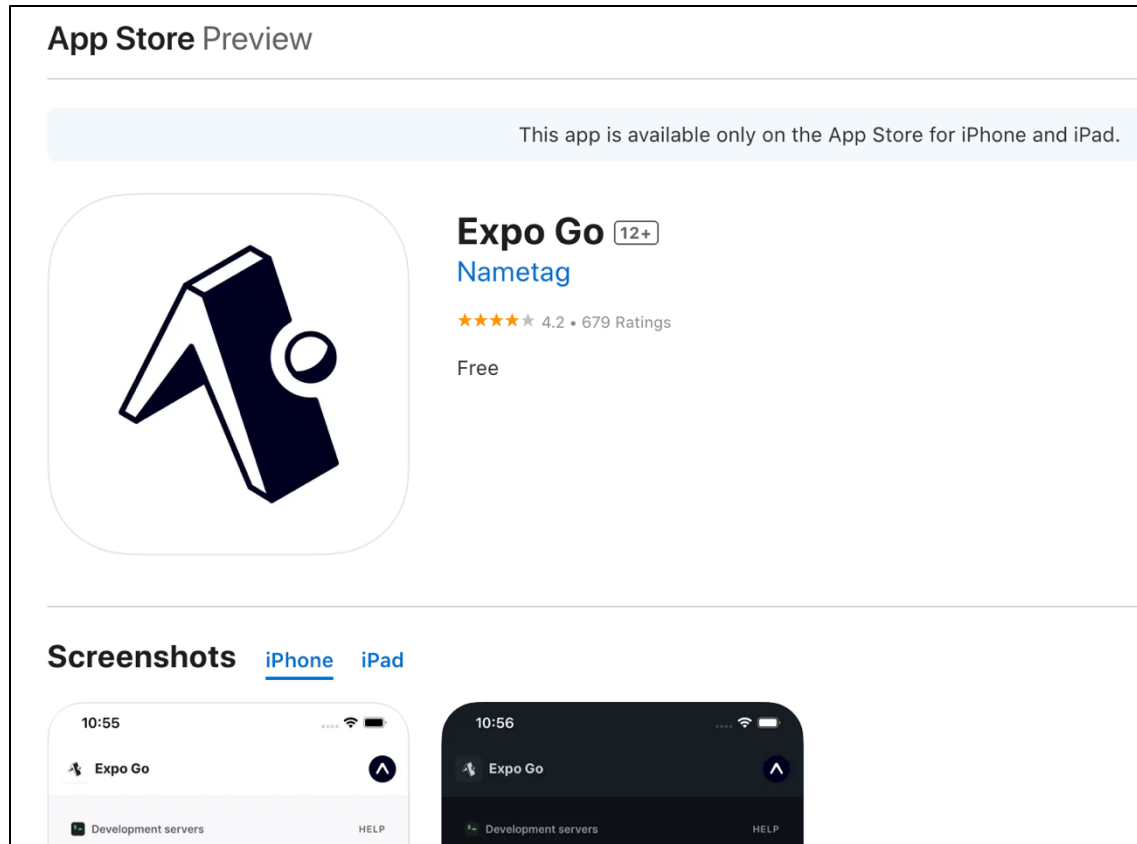
- Para usar o Expo basta instalar seu pacote `'npx create-expo-app app1 --template blank'`
- Entre na pasta criada com `'cd app1'`
- Instale as dependências com `'npx expo install'`
- Rode a aplicação com `'npx expo start'`
- Faça a leitura do QRCode usando o celular

Ambiente Expo



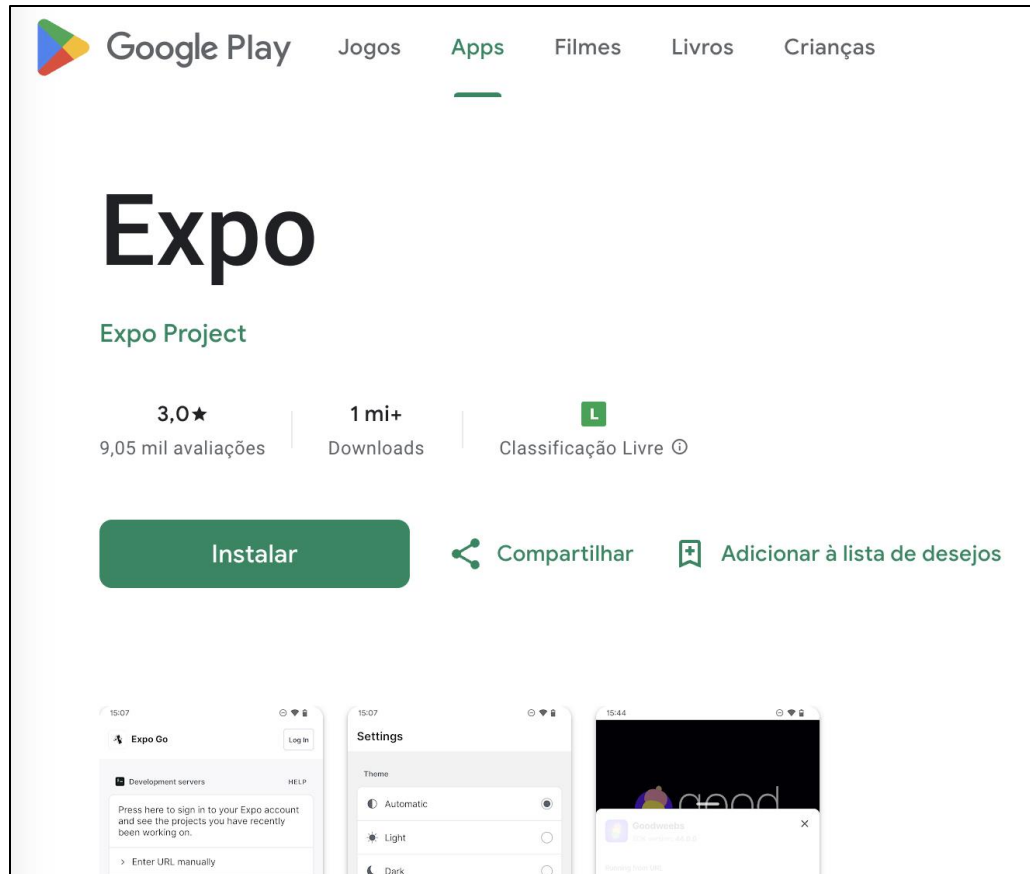
Página inicial do Expo, com links para documentação, cadastro e login.

Ambiente Expo



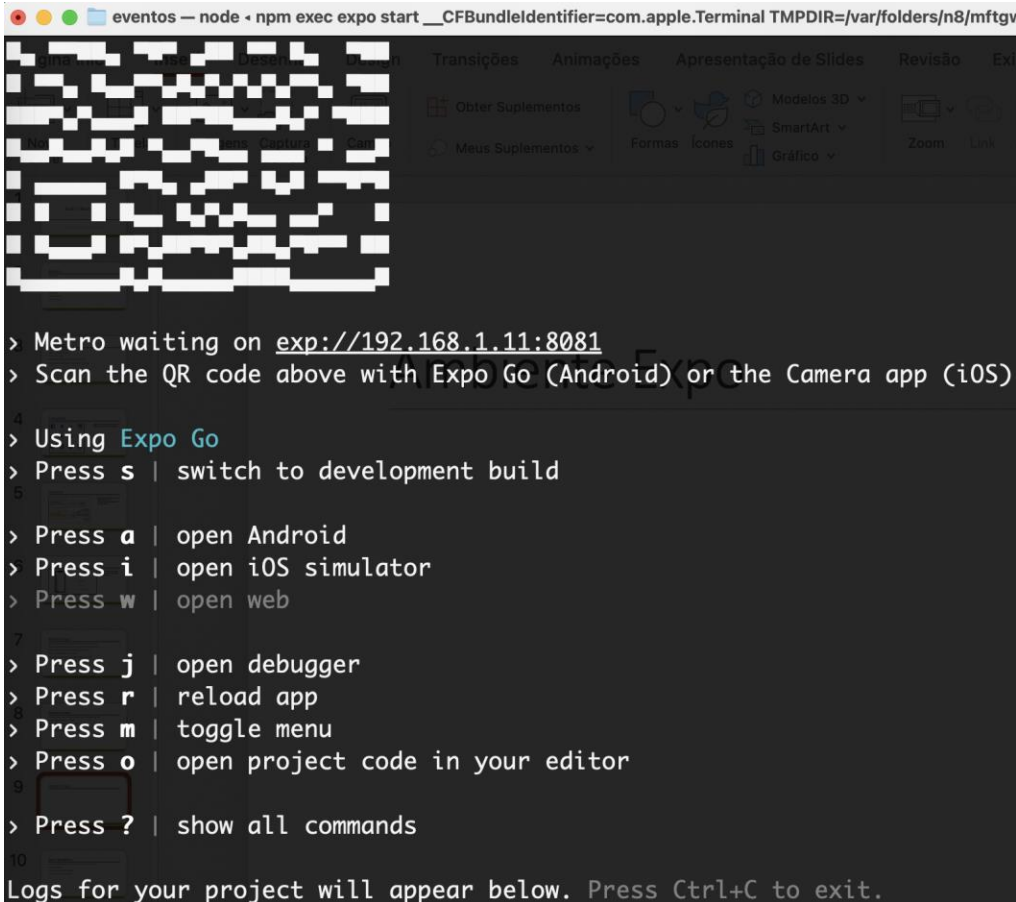
Loja da Apple com o App Expo Go, para simular seu aplicativo em desenvolvimento.

Ambiente Expo



Loja do Google com o App Expo, para simular seu aplicativo em desenvolvimento.

Ambiente Expo



```
eventos - node - npm exec expo start __CFBundleIdentifier=com.apple.Terminal TMPDIR=/var/folders/n8/mftgv
> Metro waiting on exp://192.168.1.11:8081
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
> Using Expo Go
> Press s | switch to development build
> Press a | open Android
> Press i | open iOS simulator
> Press w | open web
> Press j | open debugger
> Press r | reload app
> Press m | toggle menu
> Press o | open project code in your editor
> Press ? | show all commands
Logs for your project will appear below. Press Ctrl+C to exit.
```

QrCode gerado após o comando
'npx create-expo-app app1',
comandos como 'i' e 'a' são opções
de executar em simuladores iOS e
Android respectivamente.

Componentes

- Função que retorna um JSX
- Deve retornar um elemento principal
- Pode conter outros componentes
- Pode manter estado e propriedades
- Pode ter funções e eventos

Componentes

```
1  import { StatusBar } from 'expo-status-bar'
2  import { StyleSheet, Text, View } from 'react-native'
3  import BuscaFrete from './components/BuscaFrete'
4
5  export default function App() {
6    return (
7      <View>
8        <BuscaFrete />
9        <StatusBar style="auto" />
10      </View>
11    )
12  }
```

Componente React Native, que utiliza outro componente, chamado 'BuscaFrete'.

Propriedade e estado

- As propriedades são parâmetros passados para um componente filho
- Não podem ser modificados, apenas lidos
- Tem o objetivo de passar dados para inicializar o componente filho
- Pode ser passado objeto complexo nas props
- Para um componente receber props use `'const NomeComp = (props) => { }'`
- No componente pai usar o componente passando as props
 - `'<NomeComp cor={cor} tipo={tipo} />'`

Propriedade e estado

```
3  import BuscaFrete from './components/BuscaFrete'
4
5  export default function App() {
6    return (
7      <View>
8        <BuscaFrete cor='green' tipo="detalhado" />
9        <StatusBar style="auto" />
10      </View>
11    )
12  }
```

Propriedades 'cor' e 'tipo' sendo passadas para o componente 'BuscaFrete.'

Propriedade e estado

- O estado pode ser criado com a função `'useState()'`
- Esta função retorna um Array com 2 elementos
 - O primeiro com o atributo em si
 - O segundo com uma função para alterar esse atributo
- Para cada atributo a ser armazenado, deve-se chamar a função `'useState()'`
- Importar `useState` com `'import { useState } from "react"'`
- Para criar um estado, usar o comando `'const [cep, setCep] = useState("")'`
- Um estado inicial é definido quando se chama a função `useState()`

Propriedade e estado

```
8 | import { useState } from 'react'
9 |
10 | const BuscaFrete = (props) => {
11 |
12 |     const [cep, setCep] = useState('')
13 |     const [endereco, setEndereco] = useState('')
14 |     const [frete, setFrete] = useState(0)
15 |
16 |     return (
17 |         <View>
18 |             <Text>Busca de frete</Text>
19 |             <TextInput placeholder='Digite o CEP' />
20 |             <Button title='Calcular' />
21 |             <Text> Cep: {cep} </Text>
22 |             <Text> Endereço: {endereco} </Text>
23 |             <Text> Frete: {frete} </Text>
24 |         </View>
25 |     )
26 | }
27 |
28 | export default BuscaFrete
```

Atributos sendo
definidos no estado
com o 'useState()'.

Eventos

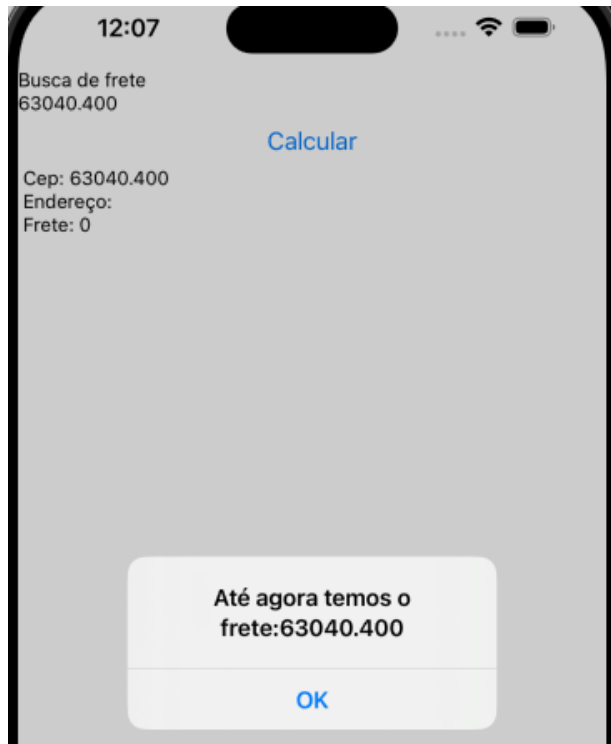
- São ações do usuário ou do sistema que podem chamar funções
- Podem conter referência para o objeto origem do evento
- São levemente diferentes dos eventos de ReactJS
- Alguns eventos requerem chamar uma função genérica, que chama outra função
- Os eventos podem utilizar como parâmetros os atributos do objeto origem do evento

Eventos

```
13  const [cep, setCep] = useState('')
14  const [endereco, setEndereco] = useState('')
15  const [frete, setFrete] = useState(0)
16
17  const atualizarCep = (value) => {
18    setCep(value)
19  }
20
21  const exibirMensagem = () => {
22    Alert.alert('Até agora temos o frete:' + cep)
23  }
24
25  return (
26    <View>
27      <Text>Busca de frete</Text>
28      <TextInput placeholder='Digite o CEP' onChangeText={ atualizarCep } />
29      <Button title='Calcular' onPress={ () => { exibirMensagem() } } />
30      <Text> Cep: {cep} </Text>
```

Eventos 'onChangeText' e 'onPress' em uso.

Eventos



Resultado do uso dos eventos 'onChangeText' e 'onPress'.

Estilização

- A estilização é baseada em declarações CSS
- O objeto StyleSheet é responsável pela criação dos estilos
- Usar o método create de StyleSheet
- Para alterar estilos em tempo de execução usar estilo inline
- Outra opção para alterar o estilo é modificar o atributo style do componente

Estilização

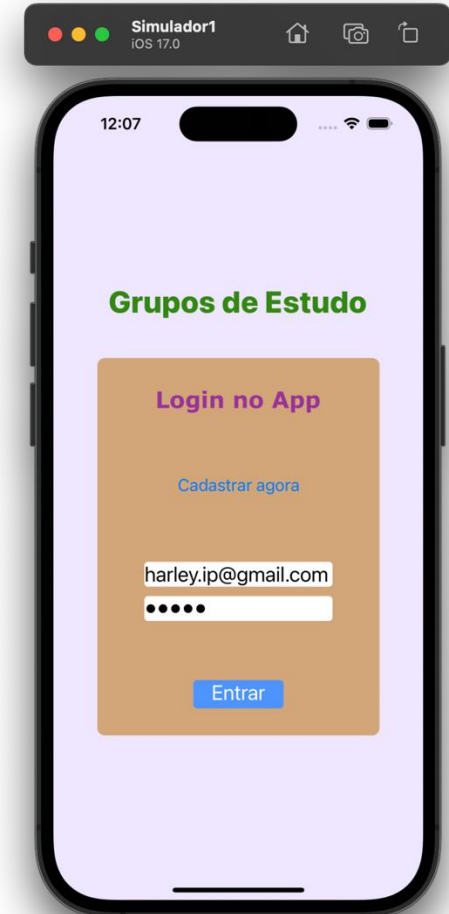
- Atributos de layout
 - Flex: define se o componente é flexível
 - BackgroundColor: cor de fundo do componente
 - JustifyContent: alinhamento dos itens no eixo principal
 - AlignItems: alinhamento dos itens no eixo secundário
 - Height: altura do componente
 - Width: largura do componente

Estilização

- Atributos de texto
 - `FontFamily`: família da fonte
 - `FontSize`: tamanho da fonte
 - `FontWeight`: peso da fonte
 - `Color`: cor da fonte

Estilização

```
26   return (  
27     <View style={styles.containerLogin}>  
28       <Text style={styles.texto1} >Login no App</Text>  
29       <Button  
30         title='Cadastrar agora'  
31       />  
32       <View>  
33         <TextInput  
34           placeholder='Email'  
35           style={styles.caixaEmailSenha}  
36           onChangeText={atualizarEmail}  
37         />  
38         <TextInput  
39           placeholder='Senha'  
40           style={styles.caixaEmailSenha}  
41           secureTextEntry={true}  
42           onChangeText={atualizarSenha}  
43         />  
44       </View>  
    )
```



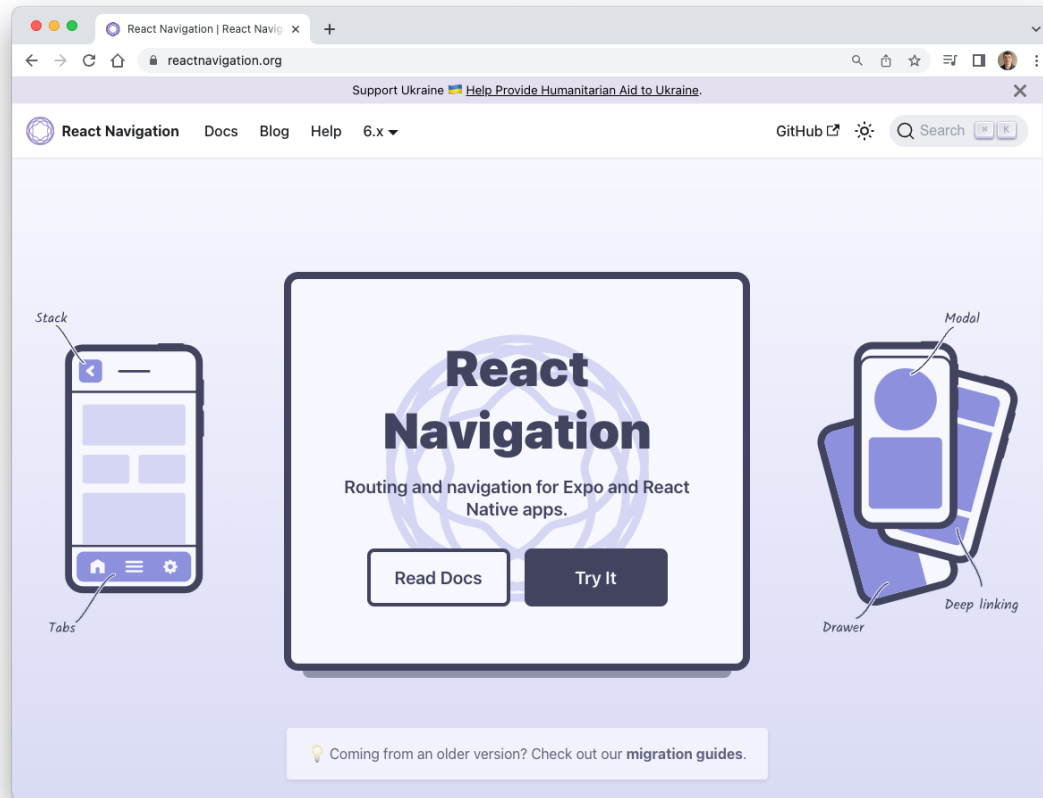
Estilização

```
57  const styles = StyleSheet.create({
58    containerLogin: {
59      flexDirection: 'column',
60      justifyContent: 'space-around',
61      alignItems: 'center',
62      height: 400,
63      width: 300,
64      backgroundColor: '#d2a679',
65      borderRadius: 8,
66    },
67    texto1: {
68      fontSize: 24,
69      fontFamily: 'Verdana',
70      fontWeight: '800',
71      color: '#939',
72    },
```

React Navigation

- Biblioteca para navegação em aplicações React e React Native
- Robusta e flexível
- Permite agregação de navegação
- Possibilita reuso de navegação

React Navigation

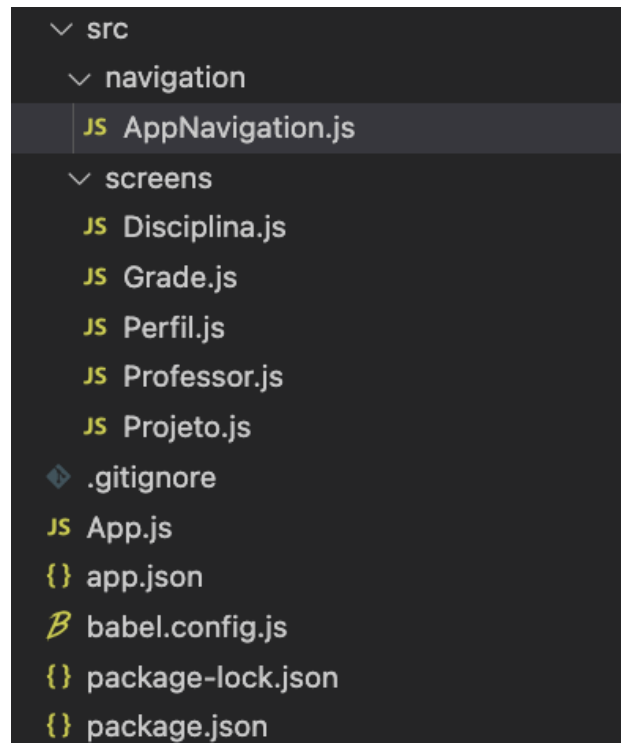


Página do React Navigation, com link para documentação, tutorial e exemplos.

React Navigation

- 'npm install @react-navigation/native'
- 'expo install react-native-screens react-native-safe-area-context'
- 'npm install react-native-screens react-native-safe-area-context'

React Navigation



O arquivo de entrada da aplicação App.js importa o arquivo de navegação AppNavigation.js que contém toda a estrutura de navegação, dessa forma, componentizando a navegação.

React Navigation

```
JS App.js > ...
1  import React from 'react';
2
3  import AppNavigation from './src/navigation/AppNavigation';
4
5  export default function App() {
6    return (
7      <AppNavigation />
8    );
9  }
```

Stack Navigator

- Navegação em pilha
- Nova tela fica sobre a anterior
- Ação de voltar desfaz a pilha, tela por tela
- Utiliza animação padrão do Android e do iOS
- Instalar a biblioteca com `'npm install @react-navigation/stack'`

Stack Navigator

- Para cada rota deve-se configurar um nome e um componente
- O nome servirá de referência para a navegação
- Rotas mantidas em NavigatorContainer

Stack Navigator

```
src > navigation > JS AppNavigation.js > AppNavigation
1  import React from 'react';
2  import { Image } from 'react-native';
3  import { NavigationContainer } from '@react-navigation/native';
4  import { createNativeStackNavigator } from '@react-navigation/native-stack';
5  import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
6
7  import Professor from '../screens/Professor';
8  import Disciplina from '../screens/Disciplina';
9  import Perfil from '../screens/Perfil';
10 import Projeto from '../screens/Projeto';
11 import Grade from '../screens/Grade';
```

Tab Navigator

- Estilo de navegação que utiliza guias
- Vantagem de os itens de navegação ficarem sempre visíveis
- Rotas mantidas em NavigatorContainer
- Instalar biblioteca com 'npm install @react-navigation/bottom-tabs'

Tab Navigator

```
41 export default function AppNavigation() {
42   return (
43     <NavigationContainer>
44       <Tab.Navigator screenOptions={{ headerShown: false }} >
45         <Tab.Screen
46           name="ProfessorTab"
47           component={StackNavigator}
48           options={{
49             tabBarLabel: 'Professor',
50             tabBarIcon: () => (<Image source={require('../assets/professor-16.png')} />) }}
51         />
52         <Tab.Screen
53           name="DisciplinaTab"
54           component={Disciplina}
55           options={{ tabBarLabel: 'Disciplina',
56             tabBarIcon: () => (<Image source={require('../assets/livro-16.png')} />) }}
57         />
58       </Tab.Navigator>
59     </NavigationContainer>
60   )
61 }
```

Tab Navigator

- `tabBarLabel`: Define o rótulo da aba
- `tabBarIcon`: Define o ícone da aba
- `indicatorStyle`: Define estilo da barra
- `onTabPress`: Define função que executa ao pressionar a tab

Parâmetros na navegação

- Possível passar dados simples ou complexos na navegação
- Objeto 'route' agrupa esses parâmetros e enviar automaticamente
- Onde usar o parâmetro, chamar com com 'route.params.[dado]'
- Logo, o componente recebe '{navigation, Route}' como parâmetro

Parâmetros na navegação

```
4  export default function Home ({navigation}) {  
5  
6      function irParaHabilidades () {  
7          navigation.navigate('Habilidades', {'usuario': 'Harley'})  
8      }  
9  
10     function irParaGrupoEstudo () {  
11         navigation.navigate('GrupoEstudo')  
12     }  
13  
14     return (  
15         <SafeAreaView>  
16             <Text>Tela Inicial</Text>  
17             <Button title='Habilidades' onPress={irParaHabilidades}></Button>
```

Parâmetros na navegação

```
3
4  export default function Habilidades ({navigate, route}) {
5      return (
6          <View>
7              <Text>Tela de Habilidades</Text>
8              <Text>Usuário: {route.params.usuario}</Text>
9          </View>
10     )
11 }
```