



Curso de Desenvolvimento de Aplicativos

Módulo IV



PROGRAMA ENEL COMPARTILHA INCLUSÃO DIGITAL

REDE MATRIZ CRIATIVA

DESENVOLVIMENTO DE APLICATIVOS

Módulo 04

AUTOR: 2A TECNOLOGIA

FORTALEZA – NOVEMBRO/ 2022

React Native	1
Aula 1	1
1. História	1
2. IDEs mais utilizadas no desenvolvimento	2
2.1 Atom	2
2.2 Visual Studio Code	3
2.3 Editor Vim	4
2.4 Sublime Text	5
2.5 Editor GNU Emacs	6
2.6 Editor Spacemacs	7
2.7 Webstorm	8
3. Expo Snack	9
3.1 Criar a nossa conta:	9
4. Estrutura Web	11
4.1 HTML	11
4.2 CSS	11
4.3 Javascript	12
5. Código	12
Aula 2	17
1. Objetivo	17
2. Criando o novo projeto	17
3. Mudando os textos	17
4. Propriedades CSS para Text	18
5. Componente TextInput	20
6. Variáveis e useState	22
7. Function	24

Aula 3	26
1. Objetivo	26
2. Criando um novo projeto	26
3. Prototipagem	26
3.1 Objetivo?	26
3.2 Como?	27
3.3 Abas	27
3.4 Desenhos	28
4 Atividade	29
Aula 4	30
1. Objetivo	30
2. Abrindo o projeto	30
2.1 Limpando o código inicial	30
3. React Navigation	31
4. CreateBottomTabNavigator	34
5. Montando a estrutura	35
6. Criando e importando arquivos locais	36
7. Screen Home	37
8. Atividade	41
Aula 5	42
1. Objetivo	42
2. Abrindo o projeto	42
3. Listando as nossas Notícias.	42
3.1 Scrollview	42
3.1 Json	43

React Native

Aula 1

1. História

Em 2012, Mark Zuckerberg, CEO do Facebook, empresa dona das maiores redes sociais do mundo, como Whatsapp, Instagram e o próprio Facebook, comentou, “O maior erro que cometemos como empresa foi apostar demais em HTML5 em oposição ao nativo”, prometendo que a companhia entregaria uma melhor experiência em dispositivos móveis em breve.

Cumprindo a promessa de Zuckerberg, em 2013, o Facebook lançou um projeto interno chamado React.js, uma espécie de Hackathon, competição em que são desenvolvidos softwares, que viria a ser responsável pela criação do framework React Native. Durante a React.js Con. em janeiro de 2015, ocorreu a primeira prévia pública do framework e em março de 2015, durante a conferência anual de desenvolvimento do Facebook, o F8, foi anunciado que o framework estava disponível de forma aberta no GitHub, plataforma de hospedagem de código-fonte.

Ainda durante o Facebook F8 de 2015, foi anunciado que duas gigantes da tecnologia, Microsoft e Samsung, iriam trazer o React Native para o Windows e Tizen, sistema operacional de Smart TVs. Com tamanho apoio, a busca pela plataforma ultrapassou, em setembro de 2016, a busca por desenvolvimento IOS e Android, de acordo com o Google Trends, mantendo sua posição até os dias atuais.

Atualmente, aplicativos extremamente utilizados como Facebook, Instagram, UberEATS, Airbnb e Nubank são desenvolvidos em React Native, isso ocorre pois a programação é facilitada devido a versatilidade da plataforma, que propicia a geração de versões nativas para IOS e Android com

o mesmo código-fonte, o que poupa muitas horas e custos das desenvolvedoras. Além disso, o framework utiliza uma das linguagens de programação dinâmicas de alto nível mais populares da atualidade, o JavaScript, combinando assim os benefícios da linguagem com a biblioteca React.js.

2. IDEs mais utilizadas no desenvolvimento

2.1 Atom

- **URL:** atom.io
- **Github:** atom
- **Documentação:**
 - Configuração;
 - Atom com React Native.
- **Plataformas:** Windows, Mac, Linux
- **Licença:** Open Source
- **Recursos:**
 - Edição multiplataforma;
 - Gerenciador de pacotes integrado;
 - Preenchimento automático inteligente;
 - Navegador do sistema de arquivos;
 - Vários painéis;
 - Encontre e substitua.
- **Pacotes:**
 - **atom-react-native-css:** É um pacote para estilizar componentes React-Native com suporte integrado para SASS / SCSS. React-native-css transforma CSS / SASS válidos no subconjunto de CSS do Facebook;

- o **react-native-snippets:** É um pacote para os snippets React Native para Atom e Nuclide;
- o **zenchat-snippets:** É uma coleção de snippets para react-native, redux e ES6;
- o **atom-xcode:** This package bridges the gap between Mac Xcode and atom. Once installed, the iOS simulator can be controlled from within the atom itself;
- o **language-babel:** This package includes Language grammar for all versions of JavaScript including ES2016 and ESNext, JSX syntax as used by Facebook React, Atom's etch and others.

2.2 Visual Studio Code

- **URL:** code.visualstudio.com
- **Github:** Microsoft / vscode
- **Licença:** Free-source Code
- **Documentação:**
 - o Configuração;
 - o Desenvolva aplicativos React Native no Visual Studio Code.
- **Plataformas:** Windows, Mac, Linux
- **Recursos:**
 - o Comandos Git integrados;
 - o Extensível e personalizável.
- **Extensões:**
 - o **Ferramentas ReactNative:** Essa extensão fornece um ambiente de desenvolvimento para projetos React Native. Você pode depurar seu código, executar react-native comandos rapidamente a partir da paleta de comandos e usar o

IntelliSense para navegar por objetos, funções e parâmetros para APIs do React Native.

2.3 Editor Vim

- **URL:** vim.org
- **Github:** vim / vim
- **Documentação:** o Vim Docs
 - Configure o Vim para React-JSX
- **Licença:** Open Source
- **Plataformas:** Mac, Linux
- **Recursos:**
 - Árvore de desfazer persistente e de vários níveis; o Sistema extenso de plugins;
 - Suporte para centenas de linguagens de programação e formatos de arquivo;
 - Pesquisa e substituição poderosa;
 - Integra-se com muitas ferramentas.
- **Plug-ins:**
 - **vim-jsx:** Syntax highlighting and indenting for JSX;
 - **vim-react-snippets:** A set of snippets for Vim to work with Facebook's React library;
 - **vim-babel:** A set of snippets for Vim to work with Facebook's React library

2.4 Sublime Text

- **URL:** sublimetext.com
- **Github:** SublimeText
- **Documentação:**
 - Configuração
- **Plataformas:** Windows, Mac, Linux
- **Licença:** Sublime Text pode ser baixado e avaliado gratuitamente; no entanto, uma licença deve ser adquirida para uso por um longo período.
- **Recursos:**
 - Vá para qualquer coisa;
 - Múltiplas seleções;
 - Paleta de Comandos;
 - Modo livre de distração;
 - Edição Dividida;
 - Mudança de projeto instantânea;
 - API Plugin;
 - Personalize qualquer coisa;
 - Plataforma Cruzada.
- **Pacotes:**
 - **react-native-snippets:** É uma coleção de Sublime Text Snippets para react-native;
 - **babel-sublime:** Definições de sintaxe para ES6 JavaScript com extensões React JSX

2.5 Editor GNU Emacs

- **URL:** gnu.org/software/emacs/
- **Documentação:**
 - Documentação oficial;
 - Configuração inicial para React Native.
- **Licença:** Gratuito sob licença GPL
- **Plataformas:** Windows, Mac, Linux
- **Recursos:**
 - Modos de edição com reconhecimento de conteúdo, incluindo coloração de sintaxe, para muitos tipos de arquivo;
 - Documentação integrada completa, incluindo um tutorial para novos usuários;
 - Suporte total ao Unicode para quase todos os scripts humanos.
 - Altamente personalizável, usando código Emacs Lisp ou uma interface gráfica;
 - Um sistema de empacotamento para baixar e instalar extensões.
- **Extensões:**
 - **web-mode.el:** É um modo principal autônomo do emacs para edição de modelos da web. É compatível com várias linguagens, incluindo JSX (React).

2.6 Editor Spacemacs

- **URL:** spacemacs.org
- **Github:** [syl20bnr / spacemacs](https://github.com/syl20bnr/spacemacs)
- **Documentação:**
 - Documentação oficial
- **Licença:** Open Source
- **Plataformas:** Windows, Mac, Linux
- **Recursos:**
 - As combinações de teclas são organizadas usando prefixos mnemônicos;
 - Detectável - exibição inovadora em tempo real das combinações de teclas disponíveis.;
 - Funcionalidades semelhantes têm a mesma ligação de chave em todos os lugares;
 - Sistema de consulta simples para encontrar rapidamente as camadas disponíveis, pacotes e muito mais;
 - A configuração voltada para a comunidade fornece pacotes com curadoria ajustados por usuários avançados e os bugs são corrigidos rapidamente.
- **Extensões:**
 - **Camada React:** camada de configuração pronta para ES6 e JSX para React. Ele reconhecerá automaticamente os arquivos .jsx e .react.js. Uma camada de pacote para integração do React.

2.7 Webstorm

- **URL:** [jetbrains.com/webstorm/](https://www.jetbrains.com/webstorm/)
- **Documentação:**
 - Documentação oficial
 - Usando ferramentas externas
- **Licença:** Paga (US\$ 129,00 para usuário único / primeiro ano)
- **Plataformas:** Windows, Mac, Linux
- **Recursos:**
 - Assistência de codificação inteligente;
 - Suporte para as tecnologias mais recentes; o Sistema de controle de versão;
 - Integração Perfeita de Ferramentas; o Depuração, rastreamento e teste; o Terminal integrado.
- **Plug-ins:**
javascript-jsx.tmbundle: Pacote Textmate para JSX (React). Atualmente suporta realce de sintaxe.

3. Expo Snack

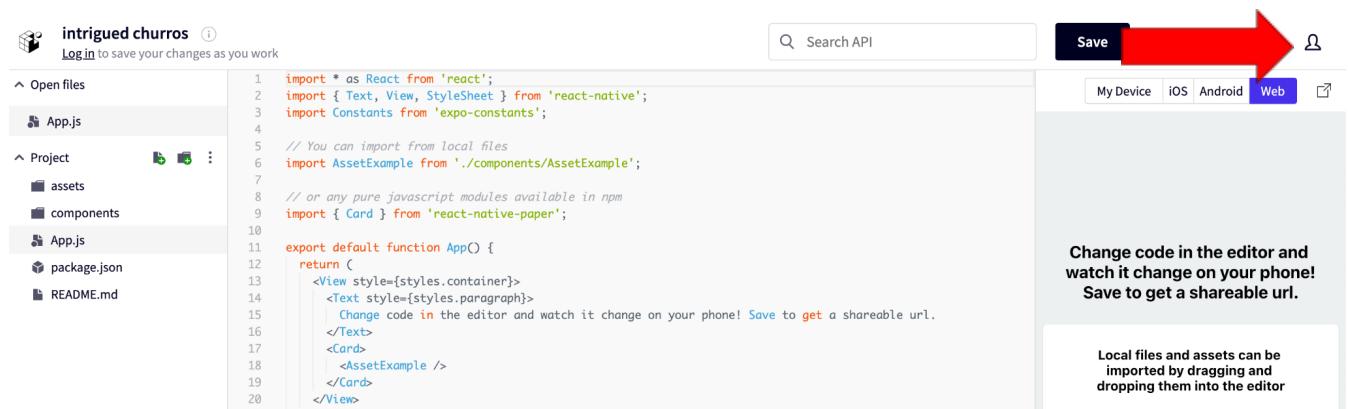
No nosso curso vamos utilizar uma tecnologia nova e inovadora, retirando todas as dificuldades de instalação e download que temos em outros curso, vamos utilizar o react-native para facilitar a nossa vida, e vamos montar e ver nossos aplicativos no Expo Snack.

Expo é uma ferramenta utilizada no desenvolvimento mobile react-native. Permite fácil acesso às API's desenvolvidas pelo expo.com. Não é preciso instalar qualquer dependência, transmutar ou configurar módulos de código nativo (java ou swift). Tudo acontece utilizando somente o javascript.

Utilizar Expo é um atalho para que o desenvolvedor front end comece a criar aplicativos mobile. Muito coisa pode ser feita com o seu conhecimento prévio em desenvolvimento de interfaces.

Vamos acessar o site: <https://snack.expo.dev/>

3.1 Criar a nossa conta:



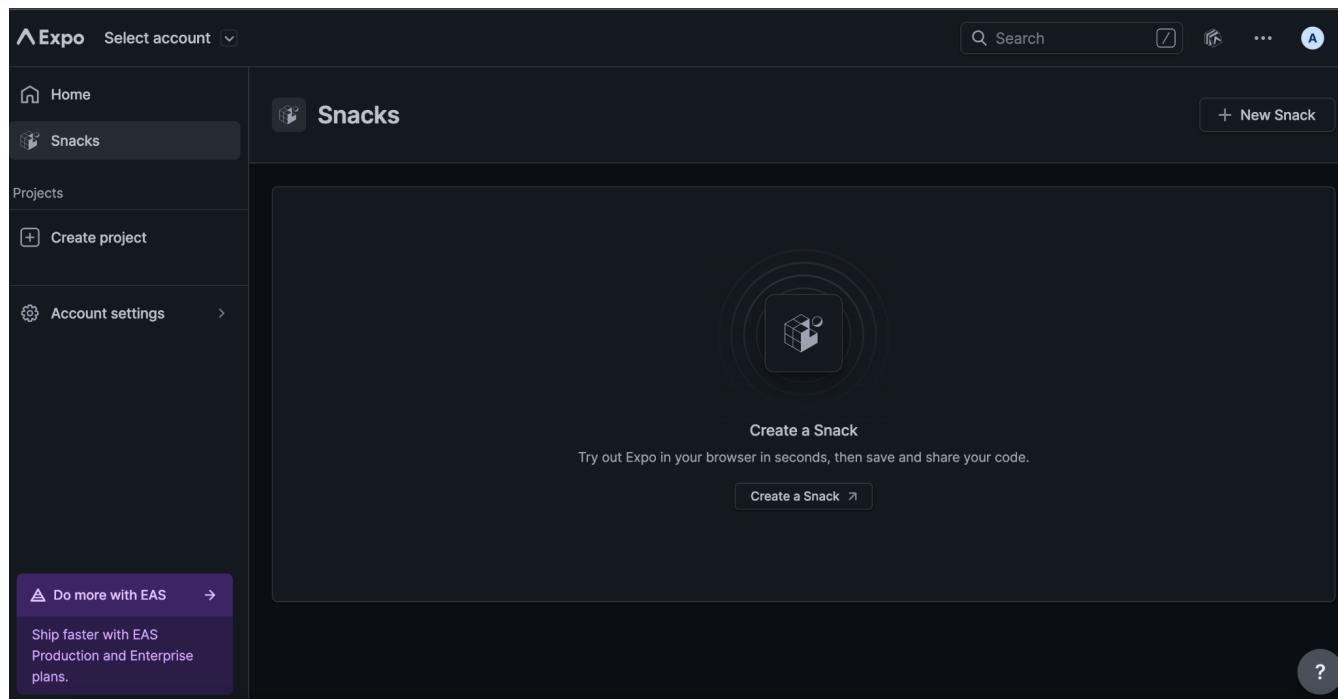
The screenshot shows the Expo Snack editor interface. On the left, there is a file tree for a project named "intrigued churros". The "App.js" file is selected and contains the following code:

```
1 import * as React from 'react';
2 import { View, Text, StyleSheet } from 'react-native';
3 import Constants from 'expo-constants';
4
5 // You can import from local files
6 import AssetExample from './components/AssetExample';
7
8 // or any pure javascript modules available in npm
9 import { Card } from 'react-native-paper';
10
11 export default function App() {
12   return (
13     <View style={styles.container}>
14       <Text style={styles.paragraph}>
15         Change code in the editor and watch it change on your phone! Save to get a shareable url.
16       </Text>
17       <Card>
18         <AssetExample />
19       </Card>
20     </View>
21 }
```

On the right, there is a toolbar with buttons for "Save", "My Device", "iOS", "Android", and "Web". A large red arrow points to the "Web" button. Below the toolbar, there is a message: "Change code in the editor and watch it change on your phone! Save to get a shareable url." and "Local files and assets can be imported by dragging and dropping them into the editor".

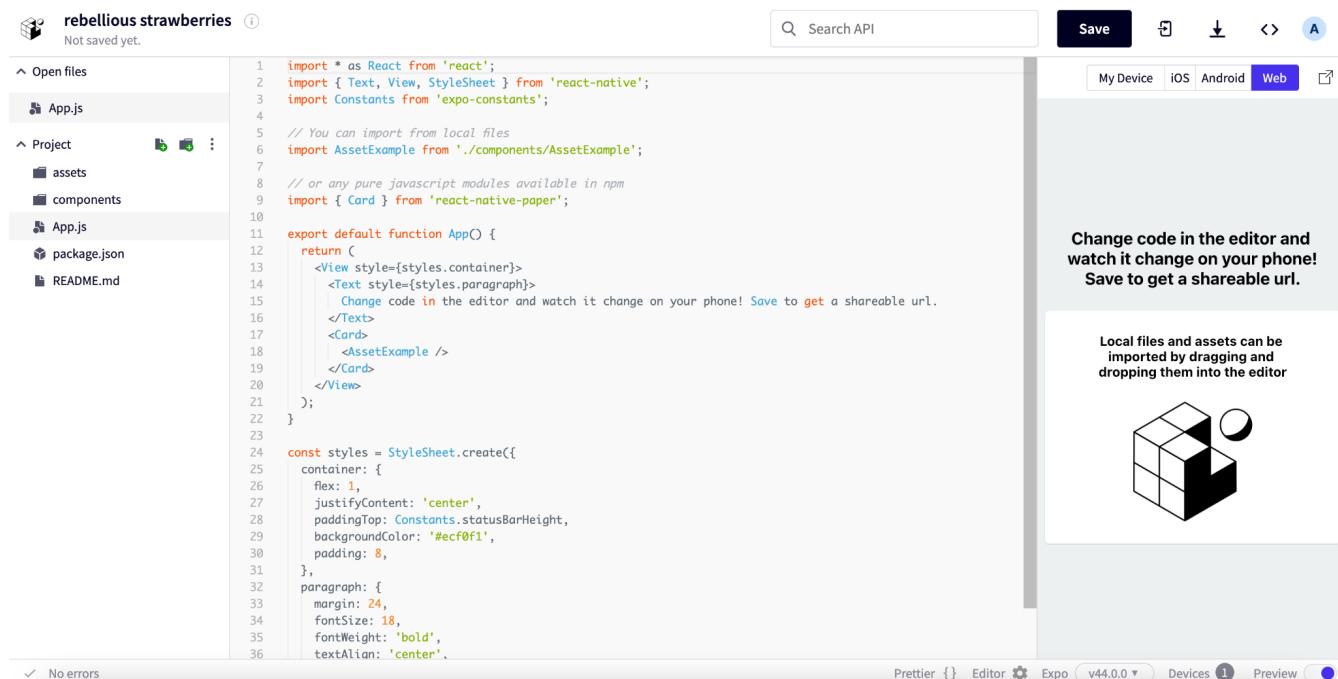
Quando fazemos o login vamos no mesmo local que clicamos para criar a nossa conta, e apertando no ícone da sua foto ou com a sua inicial.

Clique em My Snacks. E vai ser aberto essa tela:



The screenshot shows the Expo Snacks interface. On the left, there's a sidebar with options like Home, Snacks, Projects, Create project, Account settings, and a section for EAS integration. The main area is titled 'Snacks' and features a large button labeled 'Create a Snack'. Below it, there's a sub-section with the text 'Try out Expo in your browser in seconds, then save and share your code.' and a 'Create a Snack' button.

Vamos criar o nosso primeiro projeto, clicando em Nem Snack e vai abrir um projeto padrão..



The screenshot shows the Expo Dev Client editor. The left sidebar lists files: App.js (selected), package.json, and README.md. The main area displays the content of App.js:

```

1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import Constants from 'expo-constants';
4
5 // You can import from local files
6 import AssetExample from './components/AssetExample';
7
8 // or any pure javascript modules available in npm
9 import { Card } from 'react-native-paper';
10
11 export default function App() {
12   return (
13     <View style={styles.container}>
14       <Text style={styles.paragraph}>
15         Change code in the editor and watch it change on your phone! Save to get a shareable url.
16       </Text>
17       <Card>
18         <AssetExample />
19       </Card>
20     </View>
21   );
22 }
23
24 const styles = StyleSheet.create({
25   container: {
26     flex: 1,
27     justifyContent: 'center',
28     paddingTop: Constants.statusBarHeight,
29     backgroundColor: '#ecf0f1',
30     padding: 8,
31   },
32   paragraph: {
33     margin: 24,
34     fontSize: 18,
35     fontWeight: 'bold',
36     text-align: 'center'.

```

On the right, there are instructions: 'Change code in the editor and watch it change on your phone!', 'Save to get a shareable url.', and 'Local files and assets can be imported by dragging and dropping them into the editor'. A small icon of a Rubik's cube is shown.

Primeiro vamos mudar o nome do nosso projeto, na parte superior esquerda.
Nome: Meu primeiro projeto.

The screenshot shows the Expo Go app interface. Part 1 (left) displays the project structure with files like App.js, package.json, and README.md. Part 2 (center) shows the code editor with the contents of App.js. Part 3 (right) shows the running application on a mobile device.

```

1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import Constants from 'expo-constants';
4
5 // You can import from local files
6 import AssetExample from './components/AssetExample';
7
8 // or any pure javascript modules available in npm
9 import { Card } from 'react-native-paper';
10
11 export default function App() {
12   return (
13     <View style={styles.container}>
14       <Text style={styles.paragraph}>
15         Change code in the editor and watch it change on your phone! Save to get a shareable url.
16       </Text>
17       <Card>
18         <AssetExample />
19       </Card>
20     </View>
21   );
22 }
23
24 const styles = StyleSheet.create({
25   container: {
26     flex: 1,
27     justifyContent: 'center',
28     paddingTop: Constants.statusBarHeight,
29     backgroundColor: '#ecf0f1',
30     padding: 8,
31   },
32   paragraph: {
33     margin: 24,
34     fontSize: 18,
35     fontWeight: 'bold',
36     textAlign: 'center',
37   },
38 });

```

1

2

3

Parte 1: Mostra todo os arquivos do nosso projeto

Parte 2: Mostra o código do arquivo que está selecionado.

Parte 3: Mostra o exemplo de como está ficando o projeto.

Antes de fazer o nosso primeiro projeto, temos que entender um pouco sobre a estrutura de todo projeto mobile e web.

4. Estrutura Web

4.1 HTML

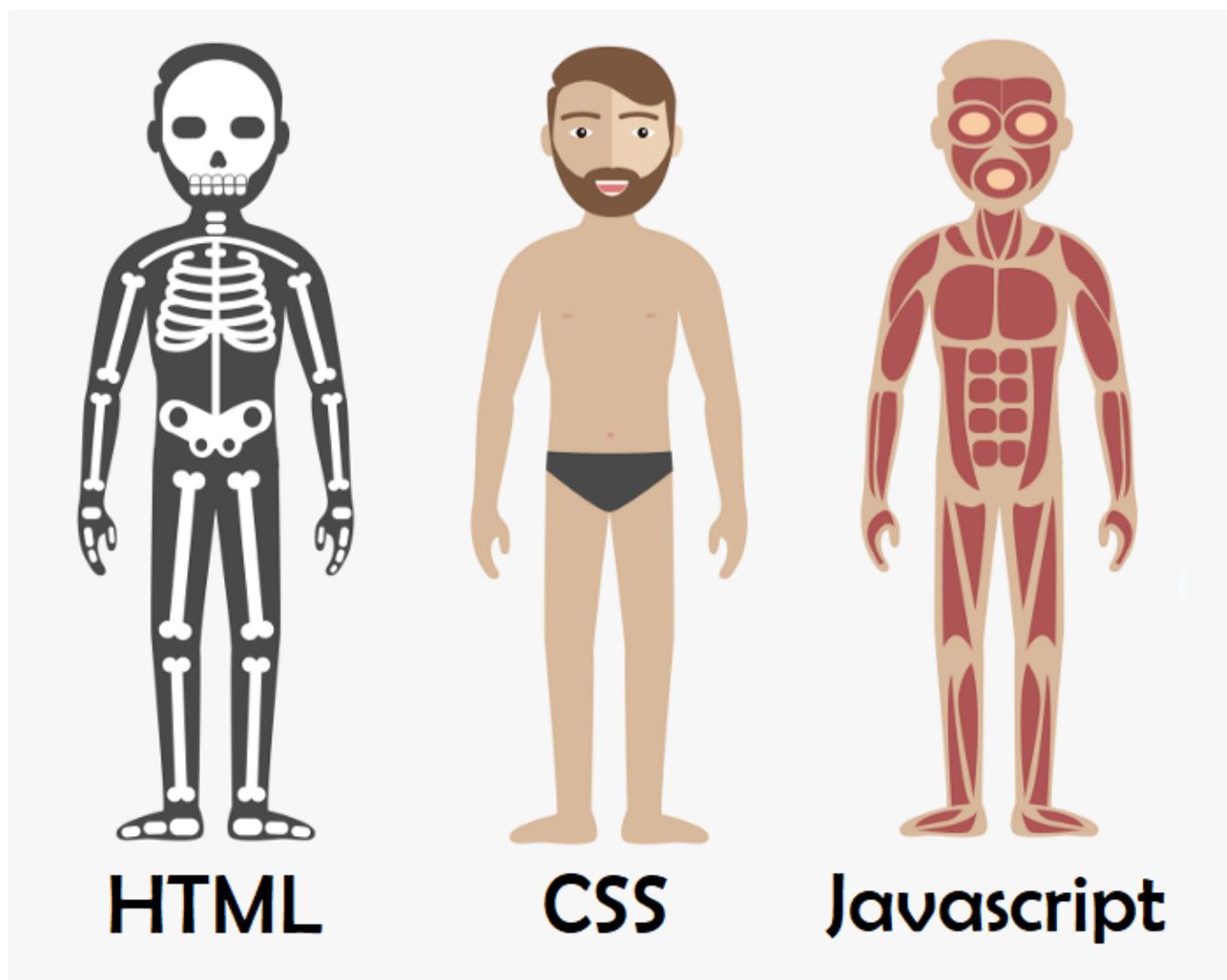
Linguagem de marcação utilizada para estruturar os elementos da página, como parágrafos, links, títulos, tabelas, imagens e até vídeos.

4.2 CSS

Linguagem de estilos utilizada para definir cores, fontes, tamanhos, posicionamento e qualquer outro valor estético para os elementos da página.

4.3 Javascript

Linguagem de programação utilizada para deixar a página com mais movimento, podendo atualizar elementos dinamicamente e lidar melhor com dados enviados e recebidos na página.



HTML: Funciona como a parte óssea do nosso projeto, toda a estrutura e divisões;

CSS: Funciona como a parte aparente do nosso projeto, tudo relacionado a alinhamento, cores e formas.

Javascript: Essa é a funcionalidade do nosso projeto, todas as funções e cálculos.

5. Código

Agora vamos entender o nosso código javascript.

```
1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import Constants from 'expo-constants';
4
5 // You can import from local files
6 import AssetExample from './components/AssetExample';
7
8 // or any pure javascript modules available in npm
9 import { Card } from 'react-native-paper';
10
11 export default function App() {
12   return (
13     <View style={styles.container}>
14       <Text style={styles.paragraph}>
15         Change code in the editor and watch it change on your phone! Save to get a shareable url.
16       </Text>
17       <Card>
18         <AssetExample />
19       </Card>
20     </View>
21   );
22 }
23
24 const styles = StyleSheet.create({
25   container: {
26     flex: 1,
27     justifyContent: 'center',
28     paddingTop: Constants.statusBarHeight,
29     backgroundColor: '#ecf0f1',
30     padding: 8,
31   },
32   paragraph: {
33     margin: 24,
34     fontSize: 18,
35     fontWeight: 'bold',
36     textAlign: 'center',
37   },
38 });
39
```

As linhas 1, 2, 3, 6 e 9 são as nossas importações, pegamos códigos pronto para utilizarmos no nosso projeto, facilitando ainda mais a nossa vida.

As linhas 5 e 8 são os comentários, eles servem para ajudar o desenvolvedor, com alguma explicação sobre aquela linha de código.

As linhas 11 a 22, é como se fosse o HTML do nosso celular, mostrando toda a estruturação.

Temos a função `App` na linha, que é todo o nosso aplicativo, como podemos perceber a função `return` está um pouco avançada (com um espaçamento a direita), mostrando que a função `return` está dentro da função `App`. Nessa mesma

lógica, o nosso aplicativo uma View, que é um Componente fundamental na construção de aplicativos, pois ele é simplesmente a Visualização da tela, como podemos ver também dentro dessa View, temos um Text, que é o nosso componente para textos, e o componente Card, que foi importado e depois explicaremos a sua função.

Todos esses componentes têm uma marcação de tag (`<>`) no início, e (`</>`) no final.

As linhas 24 até 38, nós temos o nosso CSS, que é a parte estética da nossa aplicação.

Primeiramente vamos retirar tudo aquilo que não iremos utilizar do nosso código.

Começando pelas importações, vamos deixar apenas essa linha:

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

Agora vamos limpar a nossa função App, dessa forma:

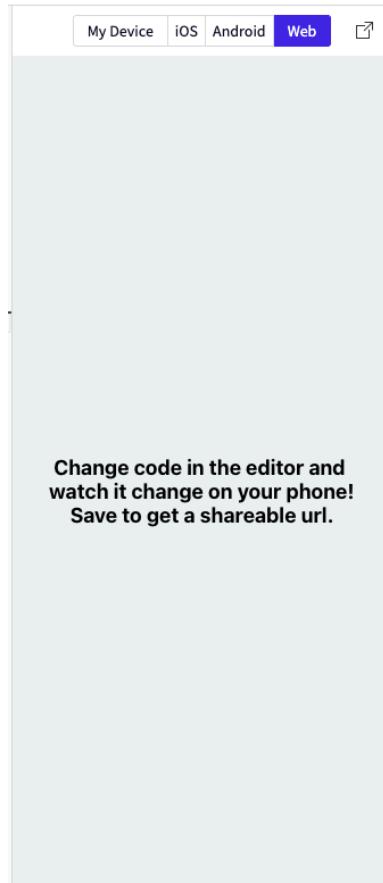
No nosso CSS vamos excluir apenas essa linha:

```
paddingTop: Constants.statusBarHeight,
```

O nosso código vai ficar da seguinte forma:

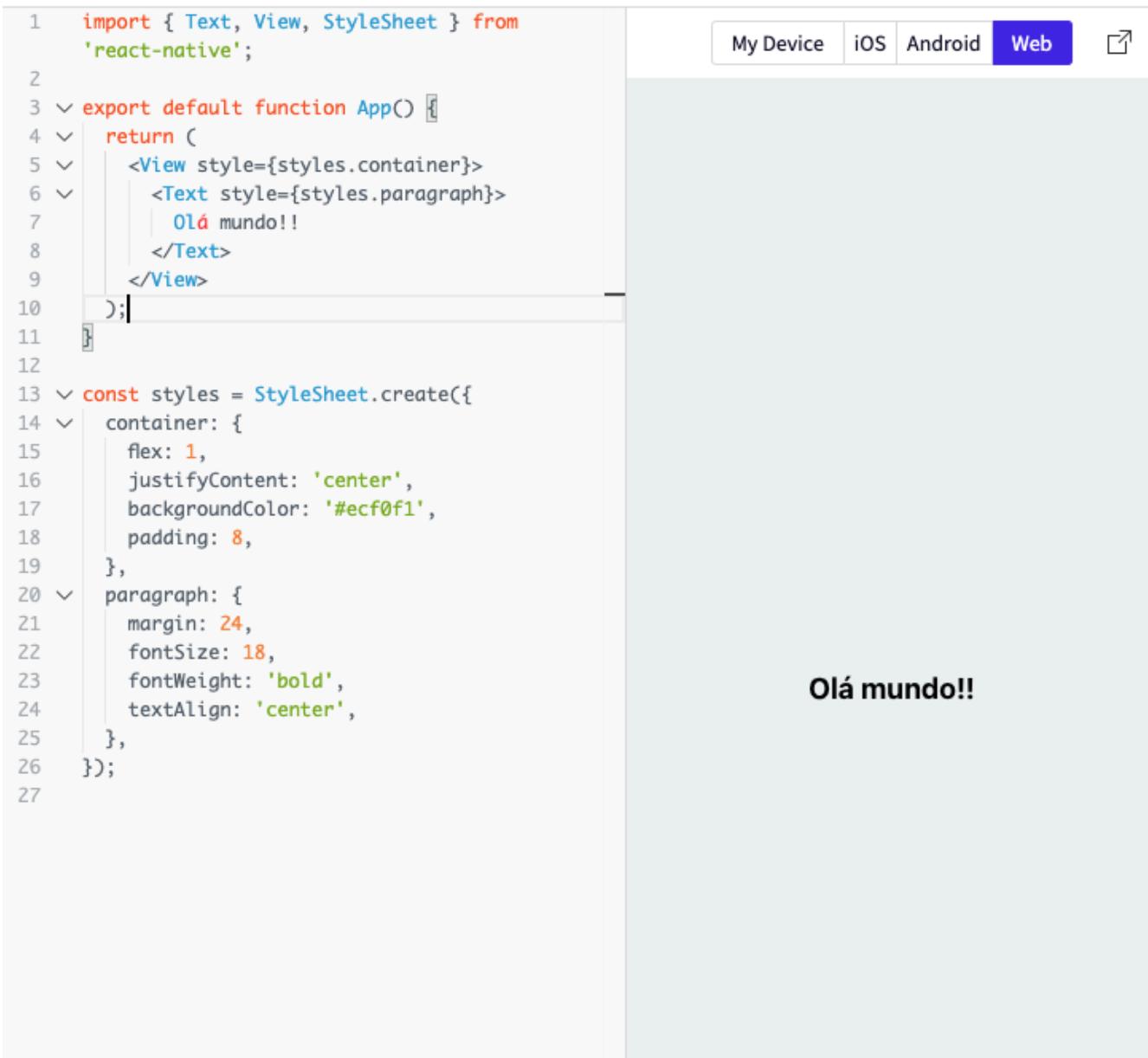
```
1 import { Text, View, StyleSheet } from 'react-native';
2
3 export default function App() {
4   return (
5     <View style={styles.container}>
6       <Text style={styles.paragraph}>
7         Change code in the editor and watch it change on your phone! Save to get a shareable url.
8       </Text>
9     </View>
10   );
11 }
12
13 const styles = StyleSheet.create({
14   container: {
15     flex: 1,
16     justifyContent: 'center',
17     backgroundColor: '#ecf0f1',
18     padding: 8,
19   },
20   paragraph: {
21     margin: 24,
22     fontSize: 18,
23     fontWeight: 'bold',
24     textAlign: 'center',
25   },
26 });
27
```

E nosso aplicativo dessa forma:



Agora vamos trocar o texto de dentro do nosso aplicativo, e vamos escrever o nosso "Olá mundo!!"

Sendo esse o resultado final:



The screenshot shows a code editor interface with a React Native file named 'App.js'. The code defines a component 'App' that returns a single 'Text' element with the bolded text 'Olá mundo!!'. The styling is managed by a local StyleSheet. The code editor has a toolbar at the top with tabs for 'My Device', 'iOS', 'Android', and 'Web', with 'Web' being the active tab. To the right of the editor, the rendered output of the code is displayed, showing the bolded text 'Olá mundo!!'.

```
1 import { Text, View, StyleSheet } from
2   'react-native';
3
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Text style={styles.paragraph}>
8         Olá mundo!!
9       </Text>
10    </View>
11  );
12}
13 const styles = StyleSheet.create({
14   container: {
15     flex: 1,
16     justifyContent: 'center',
17     backgroundColor: '#ecf0f1',
18     padding: 8,
19   },
20   paragraph: {
21     margin: 24,
22     fontSize: 18,
23     fontWeight: 'bold',
24     textAlign: 'center',
25   },
26 });
27
```

Olá mundo!!

Aula 2

1. Objetivo

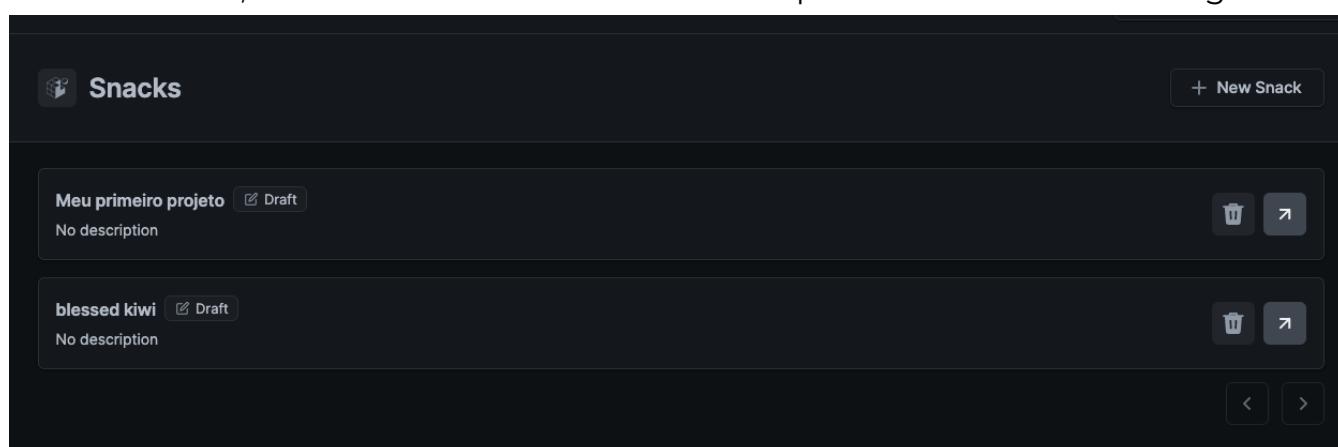
Nessa aula vamos fazer um aplicativo funcional, não será o mais bonito, porém vai funcionar direitinho.

Vamos criar juntos um aplicativo para fazer um pequeno cálculo de área de qualquer triângulo.

Vamos inserir o valor numérico para a base e o valor numérico para a altura, então o nosso aplicativo irá fazer o cálculo $(\text{base} \times \text{altura})/2$, depois mostrará o resultado desse cálculo para o usuário.

2. Criando o novo projeto

Vamos ter que criar um novo projeto para essa aula, vamos seguir um padrão com os nomes, vamos colocar o nome do nosso aplicativo de "Area de triangulo".



3. Mudando os textos

Com o nosso projeto padrão criado, vamos alterar a linha 15, mudando o Text.

```
14 <Text style={styles.paragraph}>
15   Calculadora área de Triangulo
16 </Text>
```

Agora vamos apagar o Card, e colocar um Text, falando para inserir o valor da base do triângulo, só que esse Text vai ser um pouco diferente, pois ele terá um estilo diferente.

Primeiro vamos colocar o Text dessa forma:

```

12   return (
13     <View style={styles.container}>
14       <Text style={styles.paragraph}>
15         Calculadora área de Triangulo
16       </Text>
17       <Text style={styles.insert}>
18         Insira o valor da base do triangulo
19       </Text>
20     </View>
21   );

```

Agora vamos adicionar o estilo que colocamos pra ele, demos o nome de "insert", e para colocar um novo estilo nesse texto, temos que colocar dentro da nossa variável styles, dessa maneira:

```

24   const styles = StyleSheet.create({
25     container: {
26       flex: 1,
27       justifyContent: 'center',
28       paddingTop: Constants.statusBarHeight,
29       backgroundColor: '#ecf0f1',
30       padding: 8,
31     },
32     paragraph: {
33       margin: 24,
34       fontSize: 18,
35       fontWeight: 'bold',
36       textAlign: 'center',
37     },
38     insert: [
39       {
40         fontSize: 12,
41         textAlign: 'center',
42       },
43     ],
44   });

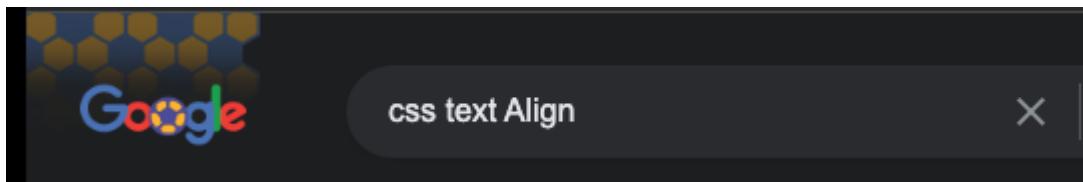
```

4. Propriedades CSS para Text

Na linha 38, adicionamos o estilo "insert" na nossa variável styles, para que ela possa ser utilizada pelo Text na linha 17.

Na linha 39, contém a propriedade CSS `fontSize`, que nada mais é do que o tamanho da letra que vai estar no texto, vamos colocar 12, mas podem alterar valores para observarem a mudança.

Na linha 40 temos a propriedade `textAlign`, que é a orientação do nosso texto, onde ele vai se posicionar na tela, no nosso aplicativo, colocamos ele no "center", mas existe outras opções, e para procurar vamos procurar no Google:



Eu selecionei essa opção da pesquisa, mas vocês podem escolher qualquer uma.

The screenshot shows the "CSS text-align property - W3Schools" page. The title is "CSS text-align property - W3Schools". Below the title, a sub-section header reads "The `text-align` property specifies the horizontal alignment of text".

CSS text-align Property

◀ Previous

Complete CSS Reference

Next ▶

Example

Set the text alignment for different <div> elements:

```
div.a {  
  text-align: center;  
}  
  
div.b {  
  text-align: left;  
}  
  
div.c {  
  text-align: right;  
}  
  
div.c {  
  text-align: justify;  
}
```

Try it Yourself »

E aqui ele me mostra as opções, vamos testar essas opções e observar.

5. Componente TextInput

Agora vamos trabalhar com um componente novo, o TextInput, e como o nome já sugere, ele serve para escrevermos texto dentro dele.

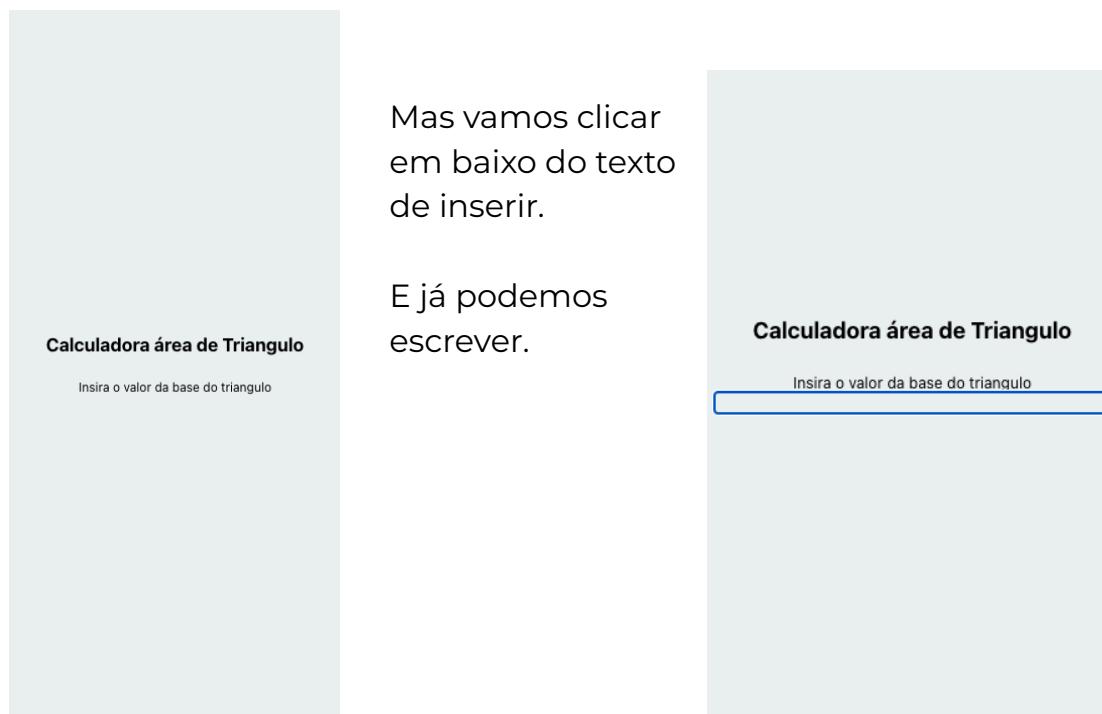
A primeira coisa que devemos fazer para utilizar ele é importado ao nosso projeto. Na linha 2 vamos colocar ele para ser importado da biblioteca 'react-native'.

```
1  import * as React from 'react';
2  import { Text, View, StyleSheet, TextInput } from
3    'react-native';
4  import Constants from 'expo-constants';
```

Ele está clarinho pois não está sendo utilizado, vamos colocar ele embaixo do Text que temos que fala para inserir o valor da base. (Lembrando que a tag abre <> e ela tem que fechar </>, ficando dessa forma:

```
11  export default function App() {
12    return (
13      <View style={styles.container}>
14        <Text style={styles.paragraph}>
15          Calculadora área de Triangulo
16        </Text>
17        <Text style={styles.insert}>
18          Insira o valor da base do triangulo
19        </Text>
20        <TextInput/>
21      </View>
22    );
23  }
```

Se olhar o aplicativo é como se não tivesse mudado nada.



Vamos melhorar a visualização do nosso componente, criando um estilo para ele.

```
39   insert: {  
40     fontSize: 12,  
41     textAlign: 'center',  
42   },  
43   input: [REDACTED]  
44     height: 40,  
45     margin: 12,  
46     borderWidth: 1,  
47     padding: 10,  
48   ],  
49 );  
50
```

A propriedade, height, é a altura do nosso TextInput.

A propriedade, margin, é o espaço ao redor do TextInput.

A propriedade, borderWidth, é a grossura da linha da borda.

A propriedade, padding, é espaço entre a borda e o texto do nosso TextInput.

Agora mudem os valores para observarem como fica o nosso TextInput.

Está sublinhado de laranja, pois ainda não está sendo utilizado, então colocar esse estilo no nosso TextInput.

```
11  export default function App() {  
12    return [  
13      <View style={styles.container}>  
14        <Text style={styles.paragraph}>  
15          Calculadora área de Triangulo  
16        </Text>  
17        <Text style={styles.insert}>  
18          Insira o valor da base do triangulo  
19        </Text>  
20        <TextInput  
21          style={styles.input}  
22        />  
23      </View>  
24    ];  
25  }
```

TextInput começa na linha 20 e fecha na linha 22.

Agora vamos colocar o placeholder, que nada mais é que uma pequena dica para o usuário:

```

11  export default function App() {
12    return [
13      <View style={styles.container}>
14        <Text style={styles.paragraph}>
15          Calculadora área de Triangulo
16        </Text>
17        <Text style={styles.insert}>
18          Insira o valor da base do triangulo
19        </Text>
20        <TextInput
21          style={styles.input}
22          placeholder="Insira a base"
23        />
24      </View>
25    ];
26  }
27
28 const styles = StyleSheet.create({
29   container: {
30     flex: 1,
31     justifyContent: 'center',
32     paddingTop: Constants.statusBarHeight,
33     backgroundColor: '#ecf0f1',
34     padding: 8,
35   },
36   paragraph: {
37     margin: 24,
38     fontSize: 18,
39     fontWeight: 'bold',
40     textAlign: 'center',
41   },
42 });

```

Calculadora área de Triangulo

Insira o valor da base do triangulo

Insira a base

Como podemos observar, já apareceu no nosso aplicativo, lembrando que se formos escrever alguma coisa, o placeholder que está dentro dele vai apagar.

6. Variáveis e useState

Agora temos que salvar o texto que vamos escrever dentro de uma variável, e da mesma forma, temos que falar a função que irá mudar esse valor, dessa maneira:

Primeiro vamos criar a variável e função para alterá-la:

```

11  export default function App() {
12    const [base, onChangeBase] = React.useState("");
13    return (
14      <View style={styles.container}>
15        <Text style={styles.paragraph}>

```

Na linha 12 estamos falando que estamos criando uma variável constante (const), que tem o nome de "base", e a sua função de mudança de estado é a "onChangeBase", estamos colocando dentro dessa variável `React.useState("")`, pois isso fala para o aplicativo que essa variável tem estados e o estado inicial dela é vazio.

Segundo, vamos colocar as propriedades no nosso TextInput:

```

21   </View>
22   <TextInput
23     style={styles.input}
24     placeholder="Insira a base"
25     value={base}
26     onChangeText={onChangeBase}
27   />
28 </View>

```

Agora o nosso TextInput está completo.

Se para calcularmos a área do triângulo temos que ter o valor da base e da altura, então temos que fazer o usuário inserir o valor da altura.

Fica o desafio para vocês fazerem, com as informações que temos.

Vamos considerar que vocês já fizeram e estão aqui apenas para conferir, vai ficar dessa forma:

```

export default function App() {
  const [base, onChangeBase] = React.useState("");
  const [altura, onChangeAltura] = React.useState("");
  return (
    <View style={styles.container}>
      <Text style={styles.paragraph}>
        Calculadora área de Triangulo
      </Text>
      <Text style={styles.insert}>
        Insira o valor da base do triangulo
      </Text>
      <TextInput
        style={styles.input}
        placeholder="Insira a base"
        value={base}
        onChangeText={onChangeBase}
      />
      <Text style={styles.insert}>
        Insira o valor da altura do triangulo
      </Text>
      <TextInput
        style={styles.input}
        placeholder="Insira a altura"
        value={altura}
        onChangeText={onChangeAltura}
      />
    </View>
  );
}

```

7. Function

Já estamos inserindo o valor da base e da altura, agora falta apenas mostrar para o usuário o valor da área.

Primeiro vamos ter que criar uma função para calcular e a variável para armazenar o cálculo.

```

11  export default function App() {
12    const [base, onChangeBase] = React.useState("");
13    const [altura, onChangeAltura] = React.useState("");
14    const [area, onChangeArea] = React.useState(0);
15
16    function calculoArea(){
17      let area = (base*altura)/2
18      onChangeArea(area);
19    }
20
21    return (
22      <View style={styles.container}>
23        <Text style={stvles.paraaraph}>
```

Nossa função vamos colocar ela na linha 16, o nome da função é "calculoArea", na linha 17 temos a nossa variável área que irá fazer o cálculo da área, e depois na linha 18 vamos mudar o estado da variável area, através da função que muda o valor.

Essa função tem que ser chamada pelo usuário quando ele quiser ver o valor da área, então vamos precisar de um Button, que é o componente de botão, e o primeiro passo para utilizá-lo é importar para o projeto.

```

1  import { View, Text, StyleSheet, TextInput } from 'react-native';
2  import Constantes from 'envn-constantes'.
```

Agora vamos colocar ele embaixo do nosso último TextInput.

```

44    <Button
45      onPress={calculoArea}
46      title="Calcular a área"
47      color="#841584"
48    />
```

O "onPress" é a função que será chamada quando clicarmos no botão.

O "title", é o texto que estará dentro do botão.

A "color", é a cor do botão, ela está codificada em hexadecimal e para trocarmos essa cor temos que inserir os valores corretos, acessando o site:

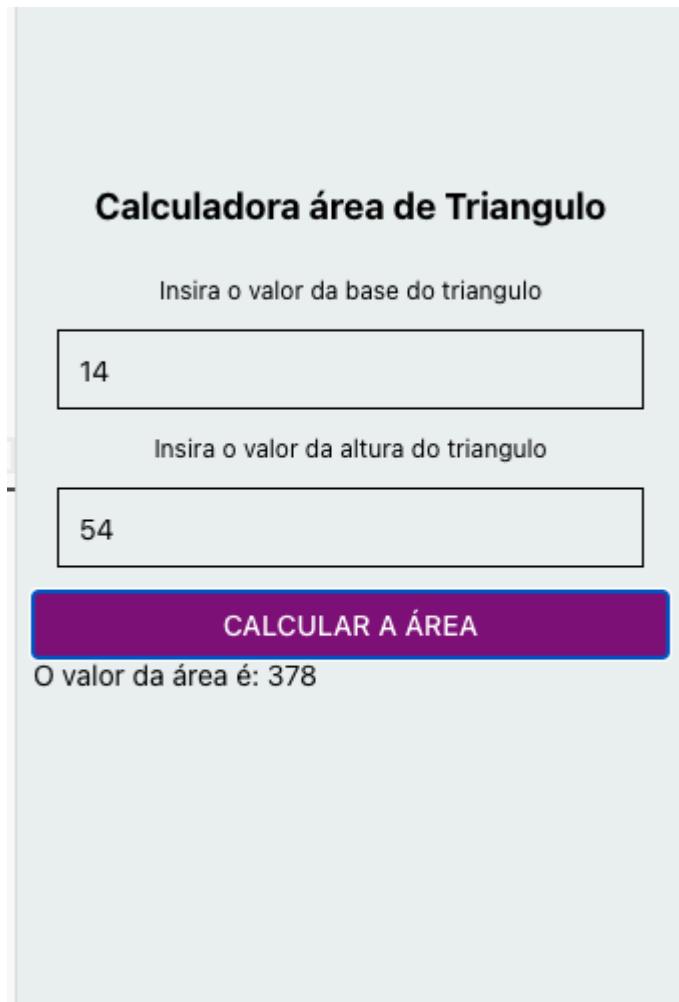
<https://celke.com.br/artigo/tabela-de-cores-html-nome-hexadecimal-rgb>

vamos poder ver várias cores para utilizarmos.

Falta apenas mostrar para o usuário o resultado do cálculo, e para isso vamos utilizar novamente o nosso componente Text:

```
48      />
49      <Text>
50          O valor da área é: {area}
51      </Text>
52      </View>
```

Dentro do Text temos o nosso texto escrito de forma normal, e também temos a nossa variável area, dentro dos {}, simbolizando que ela não é um texto, mas sim uma variável.



Calculadora área de Triangulo

Insira o valor da base do triangulo

Insira o valor da altura do triangulo

CALCULAR A ÁREA

O valor da área é: 378

6. Atividade

Vamos colocar um estilo novo para o no último Text.

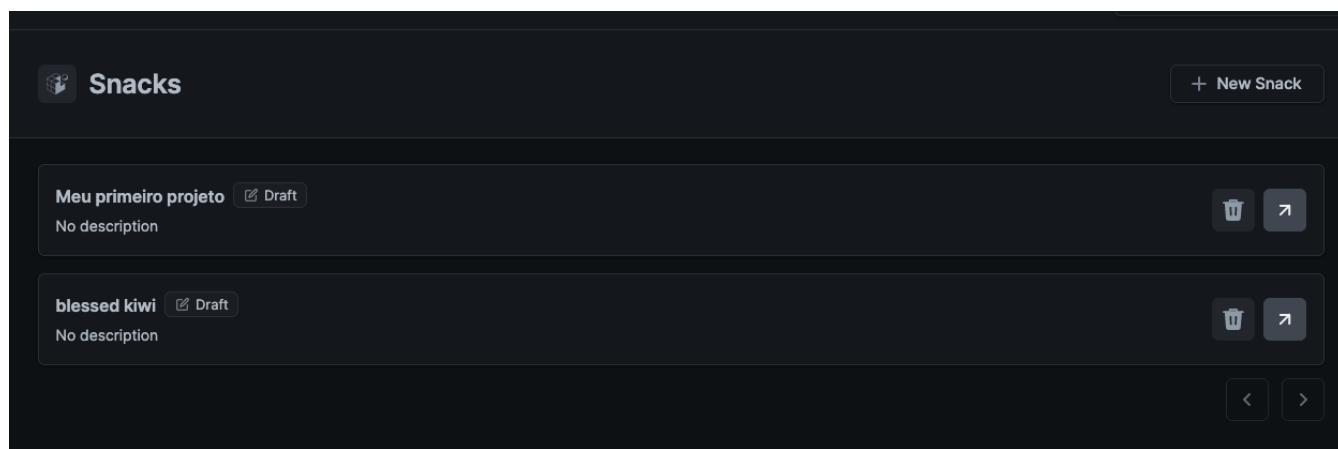
Aula 3

1. Objetivo

Nessa aula vamos começar o nosso aplicativo final, vamos aprender conceitos de navegação de telas, imagens, arquivos e várias outras coisas.

2. Criando um novo projeto

Vamos ter que criar um novo projeto para essa aula, vamos seguir um padrão com os nomes, vamos colocar o nome do nosso aplicativo de "Projetinho Final".



3. Prototipagem

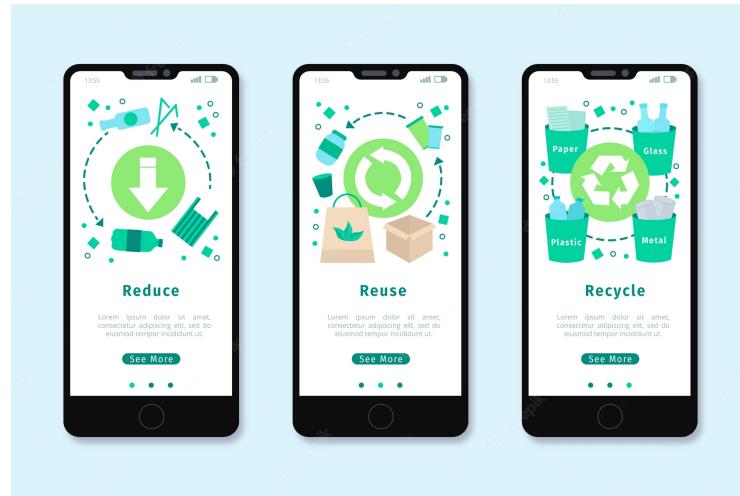
Trata-se de um processo realizado para transferir ideias do conceito para a realidade durante o desenvolvimento de um objeto virtual ou físico, por meio de protótipos.

Vamos pensar no nosso projeto antes de fazer o nosso projeto.

Temos que fazer um aplicativo relacionado a reciclagem e meio ambiente. Então vamos pensar como será o nosso aplicativo:

3.1 Objetivo?

Nosso aplicativo deve informar ao usuário, como reciclar corretamente o lixo, notícias sobre o meio ambiente e etc,



3.2 Como?

- Ele precisa ter varias telas
- As telas navegáveis entre si
- Precisamos utilizar imagens e textos.
- Pesquisar na internet sobre notícias para postarmos.

3.3 Abas



É a parte principal de navegação no nosso aplicativo, vamos fazer como o Instagram que as suas abas ficam na parte inferior do aplicativo.

Nosso aplicativo deve ter 4 abas principais para navegação do usuário.

Sendo elas:

Home: Onde será a mesclagem das notícias e dicas.

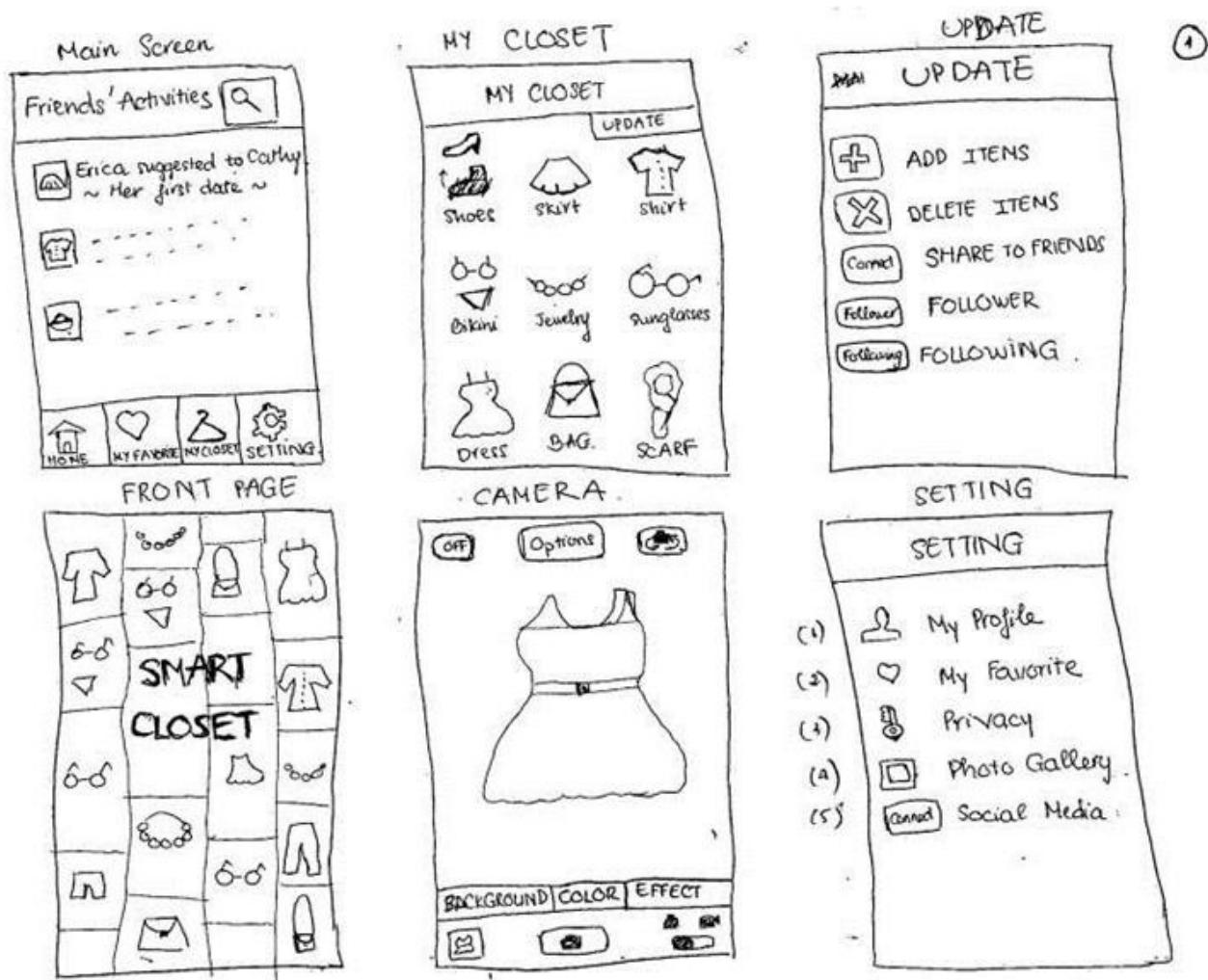
Notícias: Onde mostraremos aos usuários as últimas notícias sobre o tema de meio ambiente e reciclagem.

Dicas: Onde vamos dar dicas, de como reciclar, a maneira mais correta de descarte de algum item.

Desenvolvedor: Mostraremos a nós mesmos, uma página para falar e mostrar quem desenvolveu o aplicativo.

3.4 Desenhos

Agora vamos utilizar alguma ferramenta online ou apenas o velho papel, para desenhar como queremos que fique o nosso aplicativo, vou dar alguns exemplos:



Importante é se atentar aos detalhes, vamos literalmente desenhar como queremos que fique, primeiro vamos começar pelas nossas abas na parte inferior, depois vamos desenhando cada dela, e vamos colocar nomes a cada uma delas.

4 Atividade

Vamos pesquisar em alguns sites 8 notícias que vamos apresentar.

O que precisamos, um texto da notícia, imagens para colocarmos na nossa pagina, titulo e os autores dessa notícia.

Podem salvar tudo isso em algum documento de texto, para usarmos nas aulas posteriores.

Vamos fazer o mesmo para dicas, e aqui queria que usassem a criatividade de vocês, onde vamos premiar no final do curso os mais criativos, com alguma coisa saborosa.

Salvem no mesmo documento ou em outro diferente.

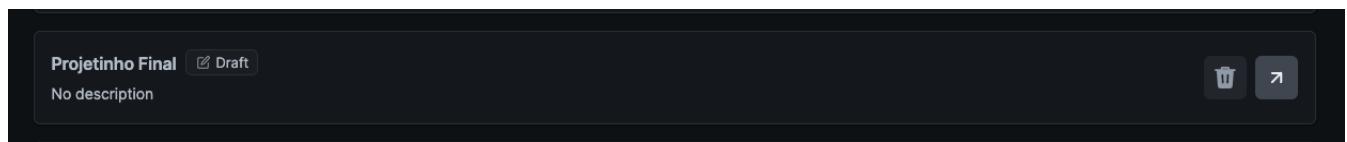
Aula 4

1. Objetivo

Nessa aula vamos começar a fazer toda a parte de navegação que vai existir dentro do nosso projeto, passar de uma tela para a outra, fazer o menu, e etc.

2. Abrindo o projeto

Para essa aula vamos utilizar o projeto que criamos na aula passada, que eu coloquei o nome de "Projetinho Final".



Apenas abra o projeto, para fazermos as nossas primeiras alterações.

2.1 Limpando o código inicial

Vamos limpar o nosso código, retirar tudo que não vamos precisar inicialmente.

No nosso arquivo App.js, vamos deixar dessa maneira.

```
1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3
4 export default function App() {
5   return (
6     <View>
7       <Text>
8         Projetinho Final
9       </Text>
10      </View>
11    );
12  }
13
14
```

No nosso aplicativo ficará dessa forma:

[My Device](#)[iOS](#)[Android](#)[Web](#)

Projetinho Final

3. React Navigation

Possivelmente, o estilo mais comum de navegação em aplicativos móveis é a navegação baseada em guias. Podem ser guias na parte inferior da tela ou na parte superior abaixo do cabeçalho.

Vamos ter que importar um conjunto de códigos para facilitar o nosso trabalho, esse conjunto chamamos de biblioteca, e essa em específica, vai fazer todo o trabalho de mudar de telas e mostrar para os usuários aquilo que for preciso.

Para importá-lo vamos colocar na nossa linha 3:

```
import { NavigationContainer } from '@react-navigation/native';
```

Esse erro foi mostrado:

Projetinho Final

Did you know: You can turn off automatic updates under Devices in the footer?

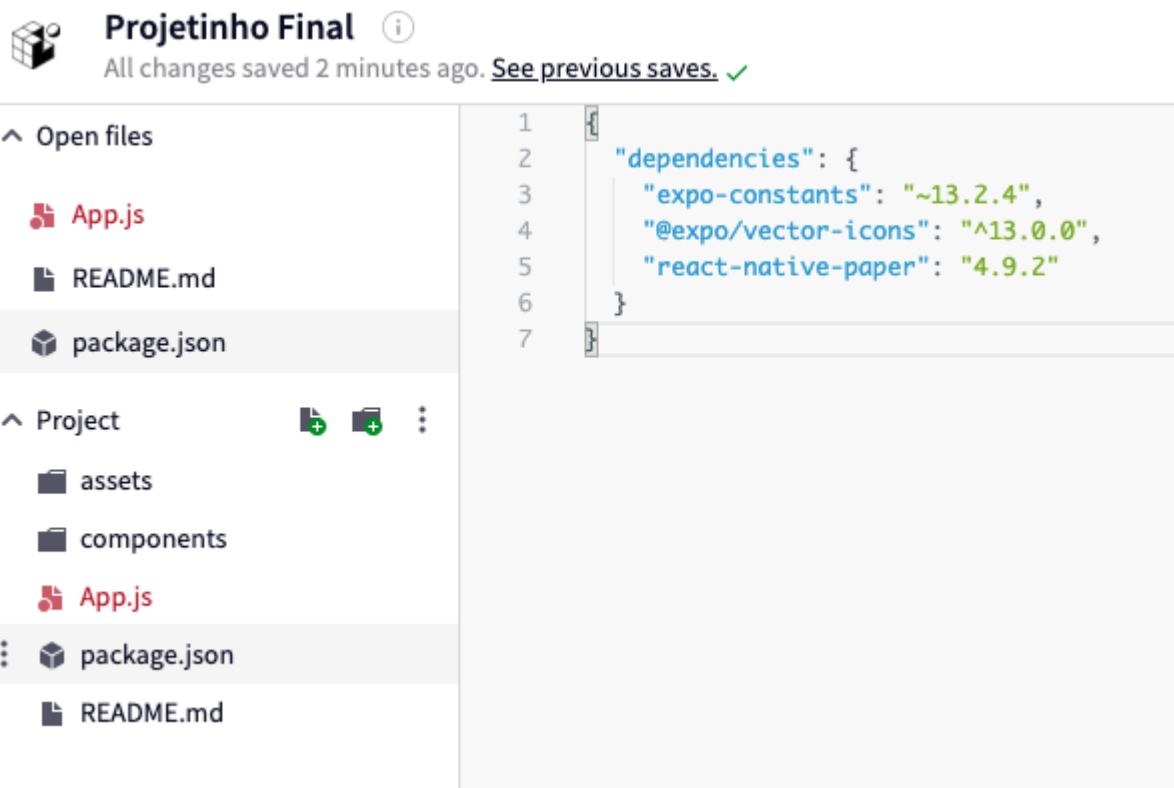
Unable to resolve module '@react-navigation/native.js'
Evaluating @react-navigation/native.js
Evaluating App.js
Loading App.js

Error: Unable to resolve module '@react-navigation/native.js'
at Object.eval (@react-navigation/native.js:1:1)

at eval (@react-navigation/native.js)
at eval (@react-navigation/native.js)
at eval (<anonymous>)
at [snack internals]
at Module.o ([snack internals])
at e.evaluate ([snack internals])
at Ze ([snack internals])
at [snack internals]
at https://snack-web-player.s3.us-west-1.amazonaws.com/v2/46/static/js/app.e3f1aedc.chunk.js:1:30706

O que ele significa? Ele nos mostra que não temos essa biblioteca instalada, então não podemos utilizá-la. Para que ela funcione como queremos, vamos instalar no nosso projeto.

Como fazer? Vamos entrar no arquivo package.json:



```

1  "dependencies": {
2    "expo-constants": "~13.2.4",
3    "@expo/vector-icons": "^13.0.0",
4    "react-native-paper": "4.9.2"
5  }
6
7

```

E dentro desse arquivo, vamos substituir tudo que está dentro por isso:

```
{
  "dependencies": {
    "@expo/vector-icons": "^13.0.0",
    "@react-native-community/masked-view": "*",
    "react-native-gesture-handler": "~2.5.0",
    "react-native-pager-view": "5.4.24",
    "react-native-paper": "^4.7.2",
    "react-native-reanimated": "~2.9.1",
    "react-native-safe-area-context": "4.3.1",
    "react-native-screens": "~3.15.0",
    "react-native-tab-view": "^3.0.0",
    "@react-navigation/bottom-tabs": "6.3.1",
    "@react-navigation/drawer": "6.4.1",
    "@react-navigation/elements": "1.3.3",
    "@react-navigation/material-bottom-tabs": "6.2.1",
    "@react-navigation/material-top-tabs": "6.2.1",
    "@react-navigation/native-stack": "6.6.1",
    "@react-navigation/native": "6.0.10",
    "@react-navigation/stack": "6.2.1"
  }
}
```

Da forma que está vocês podem copiar e colar no projeto de vocês, tem que ficar dessa maneira:

[See previous saves](#) ✓

```
1  {
2    "dependencies": [
3      "@expo/vector-icons": "^13.0.0",
4      "@react-native-community/masked-view": "*",
5      "react-native-gesture-handler": "~2.5.0",
6      "react-native-pager-view": "5.4.24",
7      "react-native-paper": "^4.7.2",
8      "react-native-reanimated": "~2.9.1",
9      "react-native-safe-area-context": "4.3.1",
10     "react-native-screens": "~3.15.0",
11     "react-native-tab-view": "^3.0.0",
12     "@react-navigation/bottom-tabs": "6.3.1",
13     "@react-navigation/drawer": "6.4.1",
14     "@react-navigation/elements": "1.3.3",
15     "@react-navigation/material-bottom-tabs": "6.2.1",
16     "@react-navigation/material-top-tabs": "6.2.1",
17     "@react-navigation/native-stack": "6.6.1",
18     "@react-navigation/native": "6.0.10",
19     "@react-navigation/stack": "6.2.1"
20   }
21 }
```

Agora o erro sumiu e podemos aproveitar todo código que importamos.

(Aproveitei, para instalar várias outras bibliotecas que vamos utilizar no decorrer do curso, para que depois apenas importamos dentro do projeto e já dê certo)

4. CreateBottomTabNavigator

Uma barra de guias simples na parte inferior da tela que permite alternar entre diferentes rotas. As rotas são inicializadas lentamente -- seus componentes de tela não são montados até que sejam focalizados pela primeira vez.

Vamos importar no nosso código, e criar uma variável para colocar os botões dentro dela, dessa forma:

```

1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import { NavigationContainer } from '@react-navigation/native';
4 import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
5
6 const Tab = createBottomTabNavigator();
7
8 export default function App() {
9   return (
10     <View>
11       <Text>
12         Projetinho Final
13       </Text>
14     </View>
15   );
16 }
17
18

```

Na linha 4 importamos os botões para o nosso projeto.

Na linha 6, criamos uma variável chamada `Tab` para deixar armazenado esses botões.

5. Montando a estrutura

```

1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import { NavigationContainer } from '@react-navigation/native';
4 import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
5
6 const Tab = createBottomTabNavigator();
7   function App(): JSX.Element
8 export default function App() {
9   return (
10     <Tab.Navigator>
11       <Tab.Screen name="Tela1" component={} />
12       <Tab.Screen name="Tela2" component={} />
13     </Tab.Navigator>
14   );
15 }
16

```

Na linha 10 acessamos o `Navigator` dentro de `Tab`, que está dizendo para o nosso projeto, que esse `Tab`(botões), são navegáveis(podem mudar de uma tela para outra). Na linha 13 fechamos essa navegação.

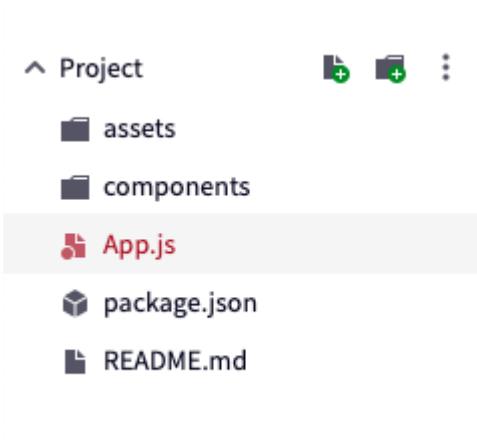
Na linha 11 e 12, criamos Screen (telas), que vamos colocar o nome por enquanto de Tela1 e Tela2, componente está sinalizando erro, pois ele pede que seja passado uma página para dentro dele.

6. Criando e importando arquivos locais

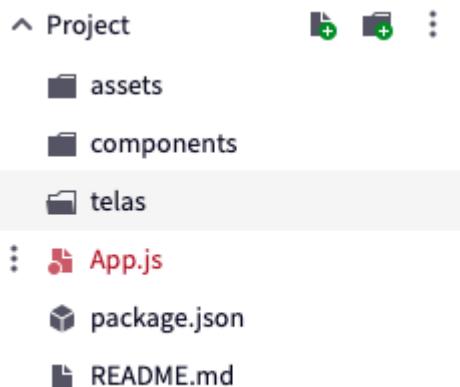
A nossa Tela1 vamos mudar o nome para Home e a nossa Tela2 vamos mudar o nome para Notícias.

```
8 ˜ export default function App() {
9   ˜   return (
10  ˜     <Tab.Navigator>
11    ˜       <Tab.Screen name="Home" component={{} />
12       <Tab.Screen name="Notícias" component={{} />
13    ˜   </Tab.Navigator>
14  ˜ );
15 }
16
17
```

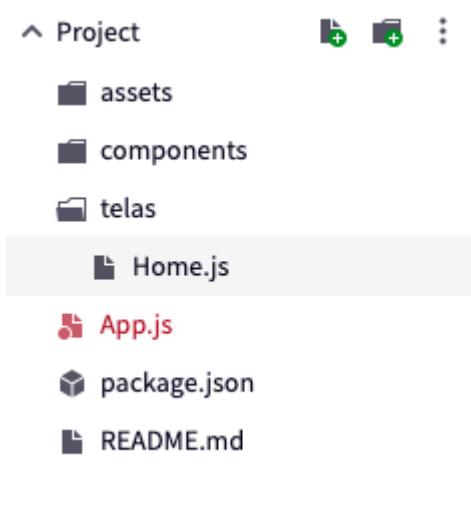
Agora vamos criar um arquivo que será a nossa tela Home. Na parte do nosso projeto, tem 3 botões na parte superior direita, o primeiro deles cria arquivo e o segundo cria pastas.



Então para deixar organizado, vamos criar uma pasta com o nome de telas:



Agora vamos criar um arquivo dentro dele, e vamos colocar o nome de Home.js



Podemos observar que o arquivo Home.js, está um pouco afastado para a direita, mostrando que ele está dentro da pasta telas.

7. Screen Home

Agora vamos montar a nossa tela Home.

```
1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3
4 export function HomeScreen() {
5   return (
6     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
7       <Text>Home!</Text>
8     </View>
9   );
10 }
11
12
```

Agora vamos no nosso arquivo App.js e vamos importar a nossa HomeScreen.

```
1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import { NavigationContainer } from '@react-navigation/native';
4 import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
5 import {HomeScreen} from './telas/Home'
6
7 const Tab = createBottomTabNavigator();
8
9 export default function App() {
10   return (
11     <Tab.Navigator>
12       <Tab.Screen name="Home" component={HomeScreen} />
13       <Tab.Screen name="Notícias" component={HomeScreen} />
14     </Tab.Navigator>
15   );
16 }
17
```

Na linha 5 estamos trazendo a HomeScreen que criamos para App.js, e estamos utilizando na linha 12 e 13.

Mas deu algum erro.

Did you know: You can turn off automatic updates under Devices in the footer?

Couldn't register the navigator. Have you wrapped your app with 'NavigationContainer'?
This can also happen if there are multiple copies of '@react-navigation' packages installed.

Error: Couldn't register the navigator. Have you wrapped your app with 'NavigationContainer'?

This can also happen if there are multiple copies of '@react-navigation' packages installed.
at t.default (@react-navigation/native is

Esse erro nos diz, que precisamos de NavigationContainer, então vamos colocá-lo no nosso projeto.

```
9  export default function App() {
10    return (
11      <NavigationContainer>
12        <Tab.Navigator>
13          <Tab.Screen name="Home" component={HomeScreen} />
14          <Tab.Screen name="Notícias" component={HomeScreen} />
15        </Tab.Navigator>
16      </NavigationContainer>
17    );
18  }
19
20
```

Dessa maneira, fazendo com que ele fique por fora de todos.

Agora ja podemos clicar e mudar entre as telas, a única coisa que vai mudar por enquanto é o cabeçalho da Screen.

Vamos criar a nossa NotíciasScreen da mesma forma que criamos a HomeScreen, fazendo o arquivo na pasta de telas e etc...

The screenshot shows a code editor interface. On the left, there's a sidebar with 'Open files' and 'Project' sections. Under 'Open files', 'Noticias.js' is selected. Under 'Project', 'Noticias.js' is also listed. The main area displays the following code:

```
1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3
4 export function NoticiasScreen() {
5   return (
6     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
7       <Text>Notícias!</Text>
8     </View>
9   );
10}
11
12
```

E também vamos importá-lo no arquivo App.js e já vamos utilizar.

The screenshot shows the 'App.js' file in the code editor. The code includes the import of 'NoticiasScreen' from the 'Noticias' module:

```
1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import { NavigationContainer } from '@react-navigation/native';
4 import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
5 import { HomeScreen } from './telas/Home'
6 import { NoticiasScreen } from './telas/Noticias'
7
8 const Tab = createBottomTabNavigator();
9
10
11
12 export default function App() {
13   return (
14     <NavigationContainer>
15       <Tab.Navigator>
16         <Tab.Screen name="Home" component={HomeScreen} />
17         <Tab.Screen name="Notícias" component={NoticiasScreen} />
18       </Tab.Navigator>
19     </NavigationContainer>
20   );
21 }
```

8. Atividade

Como atividade, vamos criar mais 2 botões na parte de baixo. Lembrando de criar os arquivos, dentro da pasta e importar.

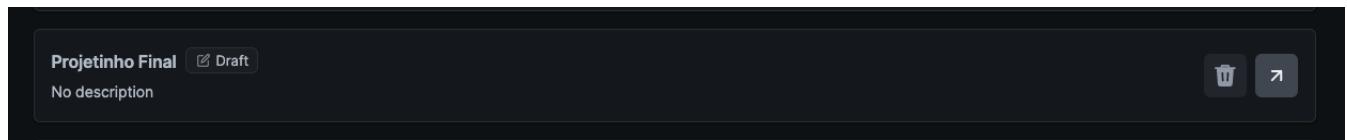
Aula 5

1. Objetivo

Nessa aula vamos começar a fazer toda a parte de navegação que vai existir dentro do nosso projeto, passar de uma tela para a outra, fazer o menu, e etc.

2. Abrindo o projeto

Para essa aula vamos utilizar o projeto que criamos na aula passada, que eu coloquei o nome de "Projetinho Final".



3. Listando as nossas Notícias.

Antes de deixar o nosso aplicativo bonito, vamos deixar ele funcional. Com isso em mente, primeiro vamos criar uma listagem bem grosseira das nossas notícias.

3.1 ScrollView

Vamos abrir o nosso arquivo `Noticias.js`, e vamos mudar a nossa View pela `ScrollView`, porque?

A View, é toda a tela disponível do celular e o `ScrollView`, estou dizendo que se existir coisas que sejam maiores que a minha tela, vamos adicionar um Scroll para poder descer e subir a tela.

```
1 import * as React from 'react';
2 import { Text, ScrollView, StyleSheet } from 'react-native';
3
4 export function NoticiasScreen() {
5   return [
6     <ScrollView style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
7       <Text>Notícias!</Text>
8     </ScrollView>
9   ];
10 }
11
12
```

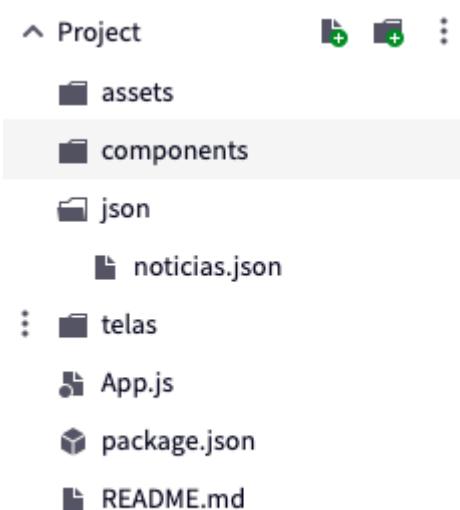
Podemos ver a mudança na importação na linha 2, e a mudança na linha 6 e 8.

3.1 Json

Vamos criar um arquivo no formato json, mas antes, o que é um json?

O JSON (JavaScript Object Notation) é um formato de arquivo para manter e trocar informações legíveis pelas pessoas. Então basicamente é um arquivo de texto estruturado.

Vamos criar uma pasta para depois criar um arquivo, dessa forma:



E dentro do arquivo noticias.json:

```
1 {  
2   "data": [  
3     {  
4       "title": "Titulo noticia 1",  
5       "url": "noticia1.com.br",  
6       "body": "Corpo da noticia",  
7       "imageUrl": "https://media.istockphoto.com/id/1146963915/pt/foto/businessperson-calculating-property-tax.jpg?b=1&s=170667a&w=0&k=20&c=QHU0-PS3Jj_KGlxfpxgA3vCURJhS-TZqmTghUED2CBY="  
8     },  
9     {  
10       "title": "Titulo noticia 2",  
11       "url": "noticia2.com.br",  
12       "body": "Corpo da noticia",  
13       "imageUrl": "https://media.istockphoto.com/id/1146963915/pt/foto/businessperson-calculating-property-tax.jpg?b=1&s=170667a&w=0&k=20&c=QHU0-PS3Jj_KGlxfpxgA3vCURJhS-TZqmTghUED2CBY="  
14     }  
15   ]  
16 }  
17  
18  
19 }
```

Na linha 1 estamos abrindo o nosso documento, da mesma forma que na linha 16, estamos fechando o nosso documento.

Na linha 2, estamos falando que esse nosso documento tem um data, que nada mais é do que dados. Podemos notar que dentro do data existe "[" e na linha 15 temos "]", isso nos diz que o documento json, dentro de data, é um array, um vetor, uma lista, e dentro desta lista vamos ter documentos com 4 propriedades (inicialmente), que é:

title, onde seria o título da nossa notícia;
url, que seria o link para acessar essa notícia;
body, que seria todo o corpo da notícia, o texto dela'
imageUrl, o link da imagem para que possamos renderizar.

Neste momento vamos utilizar dessa forma e fazer mostrar na nossa tela dessa forma.

Vamos agora importar esse documento json para o nosso arquivo noticias.js:

```
1 import * as React from 'react';
2 import { Text, ScrollView, StyleSheet } from 'react-native';
3 import dataNoticias from '../json/noticias.json'

4
5 export function NoticiasScreen() {
6   var data = dataNoticias['data']

7
8   return (
9     <ScrollView style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
10       <Text>Notícias!</Text>
11     </ScrollView>
12   );
13 }
14
15 
```

Na linha 3 temos a nossa importação.

Na linha 6, criamos uma variável para armazenar o nosso arquivo json, e estamos falando que essa variável vai receber o arquivo json, acessando a propriedade "data", que é a nossa listagem de notícias..

```
1 import * as React from 'react';
2 import { Text, ScrollView, StyleSheet, View, Button } from 'react-native';
3 import dataNoticias from '../json/noticias.json'
4
5 const CardNoticia = (props) => {
6   var noticia = props.noticia
7   return (
8     <View>
9       <Text>{noticia['title']}
```

Na nossa linha 2, fizemos algumas importações.

Na linha 5 até a 12, criamos o nosso componente de CardNoticia, ainda sem nenhuma estilização, ainda na linha 5 temos uma coisa chamada props, que é algo que o componente vai precisar para ser utilizado, no nosso caso, ele vai precisar de uma noticia.

Na linha 6 estamos falando que a nossa variável noticia, vai receber todas as props, e dentro dessas props, eu quero a props que se chama noticia. Podemos observar que quando chamamos o nosso componente CardNoticia, estamos falando que noticia = noticia1, noticia2 ou noticia3, estamos enviando a props noticia, para dentro do nosso componente CardNoticia.

Já está aparecendo os titulos das nossas noticias, vamos fazer a seguinte atividade:

Vamos atualizar o nosso documento json de notícias, com as notícias que pesquisamos, depois vamos colocar as 8 notícias na nossa página. Depois disso tudo, vamos fazer a mesma coisa para as dicas.

Aula 6

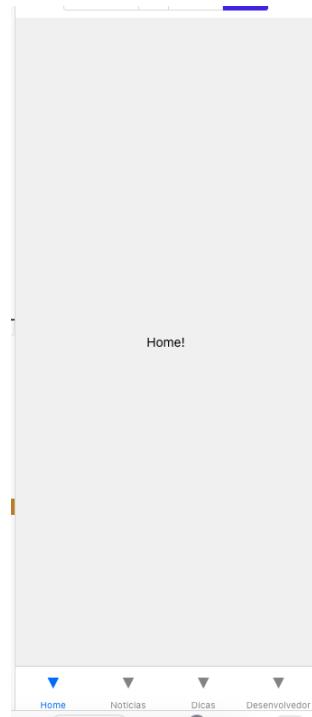
1. ScreenOptions

Vamos inicialmente retirar o nome do nosso header, da seguinte forma:

```
export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator screenOptions={{ headerShown: false }}>
        <Tab.Screen name="Home" component={HomeScreen} />
        <Tab.Screen name="Notícias" component={NoticiasStackScreen} />
        <Tab.Screen name="Dicas" component={DicasScreen} />
        <Tab.Screen name="Desenvolvedor" component={DesenvolvedorScreen} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```

Podemos perceber que na nossa Tab.Navigator, colocamos as nossas opções de tela, onde passamos que não vai ser mostrado o nome lá em cima.

Resultado:



2. Detalhadamente da Notícia

Vamos criar um novo arquivo para Detalhar a nossa Noticia:

 telas

-  Desenvolvedor.js
-  Dicas.js
-  Home.js
-  Notícias.js
-  NotíciasDetalhes.js

Criando um arquivo e colocando o nome dele de NotíciasDetalhes.js

```
1 import * as React from 'react';
2 import { Text, ScrollView, StyleSheet, View, Button } from 'react-native';
3
4 export function NotíciasDetalhes({ route }) {
5   const { notícia } = route.params;
6   return (
7     <ScrollView style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
8       <Text>NotíciasDetalhes!</Text>
9       <Text>{notícia['title']}</Text>
10      </ScrollView>
11    );
12  }
13
```

Essa vai ser a cara dele, na linha 4, estamos falando para o nosso componente que ele vai ter route, pra que serve?

route serve para poder ter controle de rota, nessa tela, vamos mostrar melhor a notícia, e o dado da notícia vai ter que chegar aqui de alguma forma, e a forma que escolhemos foi através de route, então, quando essa tela for mostrada, quem chama-la terá que passar na route, o parâmetro notícia. Já já veremos o funcionamento disto.

3. StackNavigation

Agora, para podermos navegar entre telas, sem utilizar os botões do TabNavigator, vamos utilizar o StackNavigator,

```
1 import * as React from 'react';
2 import { Text, View, StyleSheet } from 'react-native';
3 import { NavigationContainer } from '@react-navigation/native';
4 import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
5 import { createNativeStackNavigator } from '@react-navigation/native-stack';
6 import {HomeScreen} from './telas/Home'
7 import {NoticiasScreen} from './telas/Noticias'
8 import {DicasScreen} from './telas/Dicas'
9 import {DesenvolvedorScreen} from './telas/Desenvolvedor'
10 import {NoticiasDetalhes} from './telas/NoticiasDetalhes'
11
12 const Tab = createBottomTabNavigator();
13 const NoticiasStack = createNativeStackNavigator();
14
15 function NoticiasStackScreen() {
16   return (
17     <NoticiasStack.Navigator>
18       <NoticiasStack.Screen name="Noticias" component={NoticiasScreen} />
19       <NoticiasStack.Screen name="NoticiasDetalhes" component={NoticiasDetalhes} />
20     </NoticiasStack.Navigator>
21   );
22 }
23
24 export default function App() {
25   return (
26     <NavigationContainer>
27       <Tab.Navigator screenOptions={{ headerShown: false }}>
28         <Tab.Screen name="Home" component={HomeScreen} />
29         <Tab.Screen name="Noticias" component={NoticiasStackScreen} />
30         <Tab.Screen name="Dicas" component={DicasScreen} />
31         <Tab.Screen name="Desenvolvedor" component={DesenvolvedorScreen} />
32       </Tab.Navigator>
33     </NavigationContainer>
34   );
35 }
```

Na linha 5, estamos importando ele para o nosso projeto.

Na linha 10, estamos importando a nossa tela de NoticiasDetalhes.

Na linha 13, estamos fazendo a criação desse método de navegação.

Da linha 15 até a 21, estamos criando a nossa função, que vai controlar a navegação dentro das nossas Notícias. Depois teremos que colocar essa função como componente de Notícias, na linha 28.

Na linha 18 e 19, estamos falando que a nossa Notícias, terão duas telas, a NoticiasScreen e NoticiasDetalhes

Está quase tudo funcionando, agora temos que chamar a NoticiaDetalhes de alguma forma.

Vamos para o nosso arquivo Notícias.js:

```
1 import * as React from 'react';
2 import { Text, ScrollView, StyleSheet, View, Button } from 'react-native';
3 import dataNoticias from '../json/noticias.json'
4
5 const CardNoticia = (props) => {
6     var noticia = props.noticia
7     var navigation = props.navigate
8     return (
9         <View>
10            <Text>{noticia['title']}
```

Primeiramente temos que dizer que essa tela irá utilizar a navegação, na linha 21 estamos fazendo isso, passando o navigation para ela.

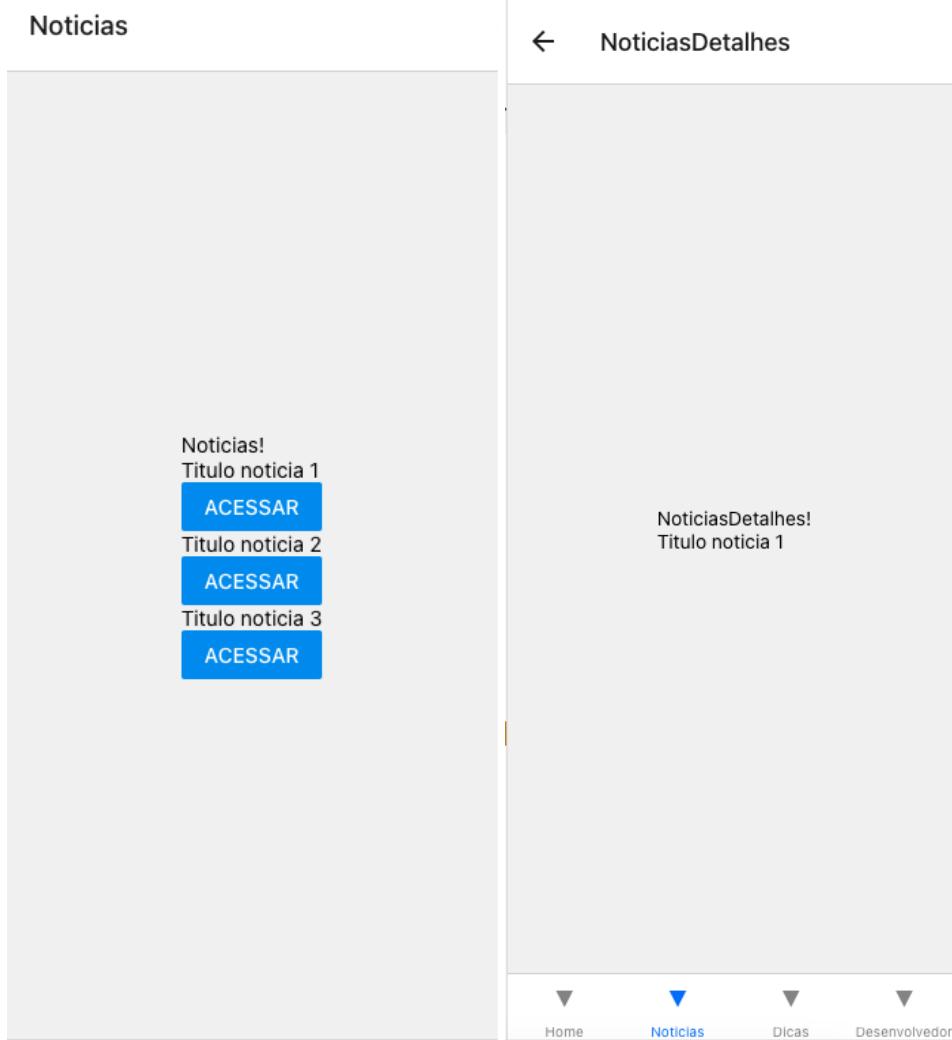
Nas linhas 31, 32 e 33 estamos passando o navigation para o nosso componente CardNoticia.

Na linha 7 estamos criando a variável para controlar a navegação.

Agora vem a grande mudança:

Na linha 11 colocamos um Button, com o título de Acessar, onde ele basicamente vai acessar a nossa Notícia. Na função, OnPress dele, que é a função que vai ser chamada quando ele for clicado, vamos chamar a navigation passando

o nome da Rota que criamos na linha 19 de App.js. E estamos passando como parâmetro a nossa notícia, assim a tela de Detalhes terá acesso a informação.



4. Melhorando o Detalhamento

Vamos abrir o arquivo NotíciasDetalhes.

```
1 import * as React from 'react';
2 import { Text, ScrollView, StyleSheet, View, Button, Image } from 'react-native';
3 import { Linking } from 'react-native';

4
5 export function NoticiasDetalhes({ route }) {
6   const { noticia } = route.params;
7   return (
8     <ScrollView >
9       <Text>Notícias Detalhes!</Text>
10      <Text>{noticia['title']}</Text>
11      <Image
12        resizeMode={'contain'}
13        style={styles.image}
14        source={{
15          uri: noticia['imageUrl'],
16        }}
17      />
18      <Text>{noticia['body']}</Text>
19      <Text style={{color: 'blue'}}>
20        onPress={() => Linking.openURL(noticia['url'])}>
21          Clique aqui para acessar a notícia.
22        </Text>
23      </ScrollView>
24    );
25  }
26
27 const styles = StyleSheet.create({
28   image: {
29     width: 200,
30     height: 200,
31   }
32 });

});
```

Na linha 3 vamos importar o Linking para podermos acessar o site de origem da notícia.

Na linha 11 estamos passando uma imagem, onde o modo que vai ser exibido será o contain, que fala que não vai cortar a imagem e mostrá-la de forma inteira.

Na linha 13 estamos passando o styles, que foi definido na linha 27.

Na linha 14 estamos falando de onde vem a imagem, e a url dela, está na própria notícia, na propriedade imageUrl da própria notícia.

Na linha 18 estamos passando o corpo da notícia, que será um texto.

Na linha 19, estamos passando um texto, com a cor azul, e nesse texto, estamos passando um OnPress, como se fosse um botão, e ao clicar vamos acessar a url da nossa notícia. Com tudo isso funcionando (não de forma bonita), e vamos fazer a mesma coisa para as Dicas.

Aula 7

1. Observação

A maioria das coisas que vamos fazer nessa aula, iremos falar dos arquivos relacionados a notícias, logo como atividade, deve ser feito as alterações para dicas.

2. Modificando o texto no header

Vamos mudar o texto no header, para que ele fique dinâmico, mostrando o título da notícia/dica que selecionamos.

Passo 1: Na nossa stackScreen vamos colocar a propriedade options, fazer com que o title daquela tela seja um parâmetro de route.

```

15  function NoticiasStackScreen() {
16    return (
17      <NoticiasStack.Navigator>
18        <NoticiasStack. any name="Noticias" component={NoticiasScreen} />
19        <NoticiasStack.Screen
20          name="NoticiasDetalhes"
21          component={NoticiasDetalhes}
22          options={({ route }) => ({ title: route.params.title })} />
23      </NoticiasStack.Navigator>
24    );
25  }
26  export default function App() {

```

Passo 2: Vamos no nosso Card e temos que passar esse parâmetro de route, como está na linha 15.

```

5  const CardNoticia = (props) => {
6    var noticia = props.noticia
7    var navigation = props.navigate
8    return (
9      <View>
10        <Text>{noticia['title']}

```

Feito isso vamos limpar a nossa tela de detalhes.

```

5  export function NoticiasDetalhes({ route }) {
6    const { noticia } = route.params;
7    return (
8      <ScrollView >
9        <Text>NotíciasDetalhes!</Text>
10       <Text>{noticia['title']}</Text>
11       <Image
12         resizeMode={'contain'}
13         style={styles.image}
14       >
15     </Image>
16   )
17 
```

Retirando essas duas linhas selecionadas.

3. Estilizando os nossos detalhes das notícias.

```

5  export function NoticiasDetalhes({ route }) {
6    const { noticia } = route.params;
7    return (
8      <ScrollView style={styles.view} >
9        <Image
10          resizeMode={'contain'}
11          style={styles.image}
12          source={{
13            uri: noticia['imageUrl'],
14          }}
15        />
16
17        <View style={styles.viewTexto}>
18          <Text>{noticia['body']}</Text>
19        </View>
20
21        <View style={styles.separador}>
22
23        <View style={styles.viewTexto}>
24          <Text style={{color: 'blue', }}>
25            onPress={() => Linking.openURL(noticia['url'])}>
26              Clique aqui para acessar a notícia.
27            </Text>
28          </View>
29
30        </ScrollView>
31      );
32    }
33 
```

Na linha 8 colocamos, um styles em toda a nossa view;

Na linha 17 e 23, colocamos uma View para abranger os nossos textos, e também colocamos styles neles, reparem que são iguais.

Na linha 21, colocamos uma view que vai servir como um separador, e também tem um styles.

```

33
34 const styles = StyleSheet.create({
35   image: {
36     width: null,
37     height: 250,
38   },
39   view:{}
40   paddingBottom: 20,
41   backgroundColor: "#C2F8E1"
42 },
43   separador:{
44     borderColor: "#1427CC",
45     borderWidth: 1,
46     marginHorizontal: 10,
47     marginVertical:15,
48   },
49   viewTexto:{}
50   marginHorizontal: 20,
51   textAlign: 'justify',
52   backgroundColor: '#ffffff',
53   padding: 10,
54   borderRadius: 10,
55 },
56 });

```

Na linha 36, colocamos o null, pois assim a largura fica sendo toda a largura disponível;

Na linha 39 temos o nosso styles view, quem tem duas propriedades, uma que serve para dar um pequeno espaçamento no final da View, que é a paddingBottom, e a outra que serve para colorir o fundo da View, que é a backgroundColor, essa última propriedade ela recebe um valor hexadecimal de uma cor, e para termos acesso a isso e vocês escolherem as suas cores, acessem o site:

<https://htmlcolorcodes.com/>

Na linha 43, temos o nosso separador:

- borderColor: Cor da borda
- borderWidth: largura da borda
- marginHorizontal: espaçamento horizontal (esquerda e direita)
- marginVertical: espaçamento vertical (cima e baixo)

Na linha 49, temos a nossa viewTexto:

- marginHorizontal: espaçamento horizontal (esquerda e direita)
- textAlign: alinhamento do texto;
- backgroundColor: cor de fundo
- padding: espaçamento nos 4 lados
- borderRadius: grau que a borda vai ser curvada.

Nossa penúltima mudança será:

```

15  function NoticiasStackScreen() {
16    return (
17      <NoticiasStack.Navigator>
18        <NoticiasStack.Screen name="Noticias" component={NoticiasScreen} />
19        <NoticiasStack.Screen
20          name="NoticiasDetalhes"
21          component={NoticiasDetalhes}
22          options={({ route }) => ({
23            title: route.params.title,
24            headerStyle: { backgroundColor: '#23D288' },
25            headerTintColor: '#fff',
26          })} />
27
28      </NoticiasStack.Navigator>
29    );
30  }

```

Na linha 24, colocamos o headerStyle, basicamente para mudarmos a cor do nosso header, escolham as suas cores.

Na linha 25, a cor da letra para que não fique apagada.

Agora vamos para o final

```

31  export default function App() {
32    return (
33      <NavigationContainer>
34        <Tab.Navigator>
35          screenOptions={{ headerShown: false}}
36          tabBarOptions={{
37            activeTintColor: '#2B73DA',
38            inactiveTintColor: '#000000',
39            activeBackgroundColor: '#C2F8E1',
40            inactiveBackgroundColor: '#23D288',
41            style: {
42              backgroundColor: '#23D288',
43              paddingBottom: 2
44            }
45          }}
46        >

```

Adicionamos o tabBarOption que são as opções dos nossos botões que ficam embaixo,

Fica como atividade vocês pesquisarem as cores que querem, e ir mudando para ver o funcionamento.

Aula 8 e 9

1. Tela de Notícias e Dicas

Vamos mudar as cores do nosso header, pra ficar igual a parte de detalhes.

```
15  function NoticiasStackScreen() {
16    return [
17      <NoticiasStack.Navigator>
18        <NoticiasStack.Screen
19          name="Noticias" |
20            component={NoticiasScreen}
21            options={{
22              headerStyle: { backgroundColor: '#23D288' },
23              headerTintColor: '#fff'
24            }}
25          />
26        <NoticiasStack.Screen
```

2. Modificando o CardNoticia e CardDicas

```
5  const CardNoticia = (props) => {
6    var noticia = props.noticia
7    var navigation = props.navigate
8    return (
9      <View style={{
10        flexDirection: 'row',
11        paddingLeft: 15,
12        backgroundColor: "#ffffff",
13        marginBottom: 10,
14        justifyContent: 'space-between',
15        borderRadius: 10,
16        alignItems: 'center'
17      }}>
18        <Text>{noticia['title']}</Text>
19        <View style={{
20          height: '100%',
21          backgroundColor: "#6959CD",
22          justifyContent: "center",
23          alignItems:"center"
24        }}>
25          <Button
26            title="Acessar"
27            onPress={() => navigation.navigate('NoticiasDetalhes', {
28              noticia: noticia,
29              title: noticia["title"],
30            })}
31            color= "#6959CD"
32          />
33        </View>
34      </View>
35    );
36  }
```

Dessa vez não criamos um styles, mas estamos passando dentro de cada componente.

flexDirection: ele fala qual vai ser o sentido que os itens vão ficar, colocamos row, então vai ficar na direção de linha, mas poderíamos colocar column, ai ficaria um em cima do outro.

justifyContent: ele vai organizar a disposição dos itens, colocamos space-between, que vai ficar pegando o início e o final, como se fosse empurrado para as bordas.

3. HomeScreen

Nessa tela vamos misturar as notícias e dicas, dessa forma, lembrando que deve ser feita todas as etapas e componentes, para que funcione corretamente, da mesma forma que fizemos nos outros locais.

```
40  export function HomeScreen() {
41
42    var dataNoticia = dataNoticias['data']
43    var dataDicas = dataNoticias['data']
44
45    var noticia1 = dataNoticia[0]
46    var dica1 = dataDicas[1]
47    var noticia2 = dataNoticia[2]
48    var dica2 = dataDicas[2]
49
```

4. Desenvolvedor

Essa etapa, será onde vocês vão apresentar a criatividade, pois baseado em tudo que vimos, vocês vão criar essa tela sozinhos.

Algumas coisas devem ter na tela, como:

Nome
Data
Contato
Uma breve descrição sobre você

e qualquer outra informação que achar legal colocar na tela.

Boa sorte!