

React Native - Componentes e estilização

O que vamos ver hoje?

- Principais componentes
- Estilização

Principais componentes do React Native

Componentes Básicos

- Diferentemente do React em que é possível utilizar componentes HTML, no React Native é necessário utilizar componentes próprios ou alguma biblioteca de UI.
- Vamos conhecer os principais componentes.
- <https://reactnative.dev/docs/components-and-apis>

ActivityIndicator

ActivityIndicator

- Usado para exibir uma indicação de carregamento da tela: <https://reactnative.dev/docs/activityindicator>

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

const App = () => (
  <View style={[styles.container, styles.horizontal]}>
    <ActivityIndicator />
    <ActivityIndicator size="large" />
    <ActivityIndicator size="small" color="#0000ff" />
    <ActivityIndicator size="large" color="#00ff00" />
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
  },
  horizontal: {
    flexDirection: 'row',
    justifyContent: 'space-around',
    padding: 10,
  },
});

export default App;
```



Text

Text

- Usado para exibir informações:
<https://reactnative.dev/docs/text>

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

const TextInANest = () => {
  const [titleText, setTitleText] = useState("Bird's Nest");
  const bodyText = 'This is not really a bird nest.';

  const onPressTitle = () => {
    setTitleText("Bird's Nest [pressed]");
  };

  return (
    <Text style={styles.baseText}>
      <Text style={styles.titleText} onPress={onPressTitle}>
        {titleText}
        {'\n'}
        {'\n'}
      </Text>
      <Text numberOfLines={5}>{bodyText}</Text>
    </Text>
  );
};

const styles = StyleSheet.create({
  baseText: {
    fontFamily: 'Cochin',
  },
  titleText: {
    fontSize: 20,
    fontWeight: 'bold',
  },
});
```

Bird's Nest

This is not really a bird nest.

TextInput

TextInput

- Usado para entrada de dados do usuário:
<https://reactnative.dev/docs/textinput>

```
import React from 'react';
import {SafeAreaView, StyleSheet, TextInput} from 'react-native';

const TextInputExample = () => {
  const [text, onChangeText] = React.useState('Useless Text');
  const [number, onChangeNumber] = React.useState('');

  return (
    <SafeAreaView>
      <TextInput
        style={styles.input}
        onChangeText={onChangeText}
        value={text}
      />
      <TextInput
        style={styles.input}
        onChangeText={onChangeNumber}
        value={number}
        placeholder="useless placeholder"
        keyboardType="numeric"
      />
    </SafeAreaView>
  );
};
```

Useless Text

useless placeholder

Button

Button

- Usado para exibir botões:

<https://reactnative.dev/docs/button>

```
import React from 'react';
import {
  StyleSheet,
  Button,
  View,
  SafeAreaView,
  Text,
  Alert,
} from 'react-native';

const Separator = () => <View style={styles.separator} />;

const App = () => (
  <SafeAreaView style={styles.container}>
    <View>
      <Button
        title="Press me"
        onPress={() => Alert.alert('Simple Button pressed')}
      />
    </View>
    <Separator />
    <View>
      <Button
        title="Press me"
        color="#f194ff"
        onPress={() => Alert.alert('Button with adjusted color pressed')}
      />
    </View>
    <Separator />
    <View>
      <Button
        title="Press me"

```

PRESS ME

PRESS ME

PRESS ME

LEFT BUTTON

RIGHT BUTTON

FlatList

FlatList

- Usado para renderizar listas:
<https://reactnative.dev/docs/flatlist>

```
{
  id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',
  title: 'First Item',
},
{
  id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',
  title: 'Second Item',
},
];

type ItemProps = {title: string};

const Item = ({title}: ItemProps) => (
  <View style={styles.item}>
    <Text style={styles.title}>{title}</Text>
  </View>
);

const App = () => {
  return (
    <SafeAreaView style={styles.container}>
      <FlatList
        data={DATA}
        renderItem={({item}) => <Item title={item.title} />}
        keyExtractor={item => item.id}
      />
    </SafeAreaView>
  );
};
```

First Item

Second Item

Image

Image

- Usado para exibir imagens:
<https://reactnative.dev/docs/image>

```

    neight: >80,
  },
});

const DisplayAnImage = () => {
  return (
    <View style={styles.container}>
      <Image
        style={styles.tinyLogo}
        source={require('@expo/snack-static/react-native-logo.png')}
      />
      <Image
        style={styles.tinyLogo}
        source={{
          uri: 'https://reactnative.dev/img/tiny_logo.png',
        }}
      />
      <Image
        style={styles.logo}
        source={{
          uri:
            'data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAADMAAAAZCAYAAAA6oTAqAAAAEXR  
FWHR7b2Z0dFyZQ8wbmdjcnVzaEB1SfMAAABQSURBVGje7dSxCOBACARB+2/ab8BEeQNhfI6WS  
YzYLudDQYGBgYGBgYGBgYGBgYGBgYGBgYGBgYGBgYGBgYGBgmQw+P/eM  
rCSUTVAIAAAABJRUSErKJggg==',
        }}
      />
    </View>
  );
};
```



Pressable

Pressable

- Usado para disparar ações de toque, semelhante a um botão: <https://reactnative.dev/docs/pressable>

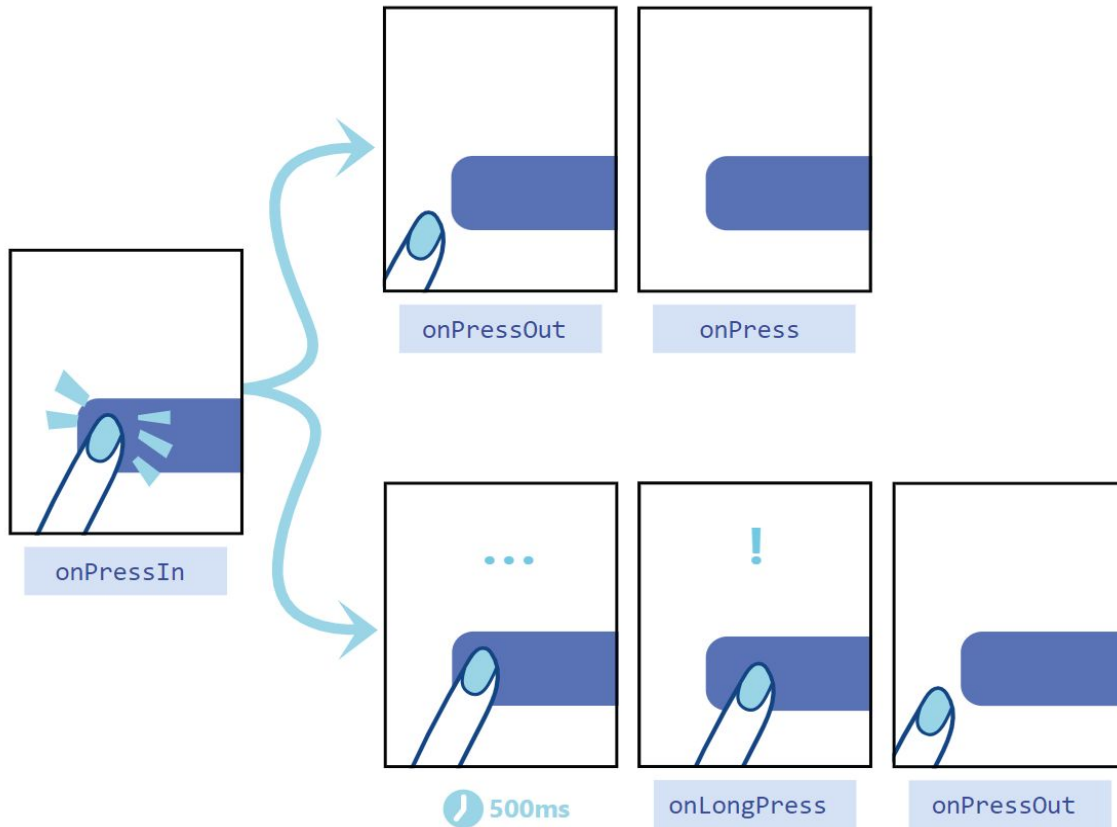
```
let textLog = '';
if (timesPressed > 1) {
  textLog = timesPressed + 'x onPress';
} else if (timesPressed > 0) {
  textLog = 'onPress';
}

return (
  <View style={styles.container}>
    <Pressable
      onPress={() => {
        setTimesPressed(current => current + 1);
      }}
      style={({pressed}) => [
        {
          backgroundColor: pressed ? 'rgb(210, 230, 255)' : 'white',
        },
        styles.wrapperCustom,
      ]>
      <Text style={styles.text}>{pressed ? 'Pressed!' : 'Press Me'}
    </Text>
  </Pressable>
  <View style={styles.logBox}>
    <Text testID="pressable_press_console">{textLog}</Text>
  </View>
</View>
);
```

Press Me

onPress

Pressable

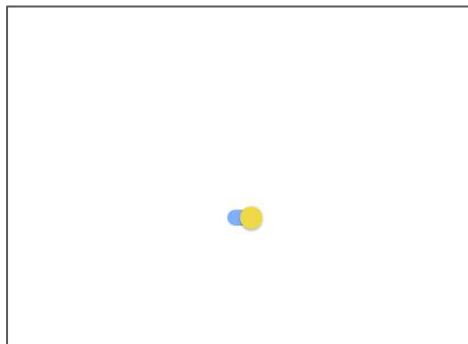
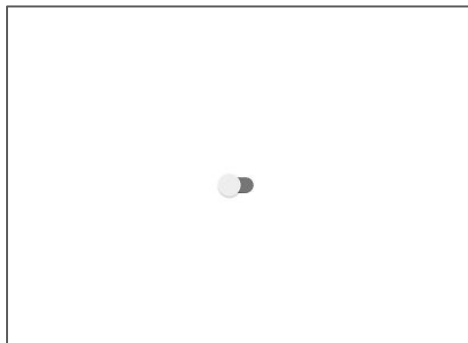


Switch

Switch

- Botão que retorna true ou false
 - <https://reactnative.dev/docs/switch>

```
const App = () => {  
  const [isEnabled, setIsEnabled] = useState(false);  
  const toggleSwitch = () => setIsEnabled(previousState => !previousState);  
  
  return (  
    <View style={styles.container}>  
      <Switch  
        trackColor={{false: '#767577', true: '#81b0ff'}}  
        thumbColor={isEnabled ? '#f5dd4b' : '#f4f3f4'}  
        ios_backgroundColor="#3e3e3e"  
        onChange={toggleSwitch}  
        value={isEnabled}  
      />  
    </View>  
  );  
};
```



Alert

Alert

- Exibe um alerta na tela <https://reactnative.dev/docs/alert>

```
Alert.alert('Alert Title', 'My Alert Msg', [  
  {  
    text: 'Cancel',  
    onPress: () => console.log('Cancel Pressed'),  
    style: 'cancel',  
  },  
  {text: 'OK', onPress: () => console.log('OK Pressed')},  
]);
```

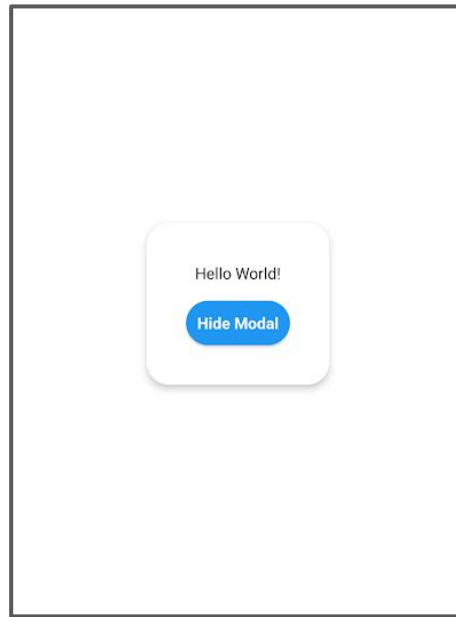
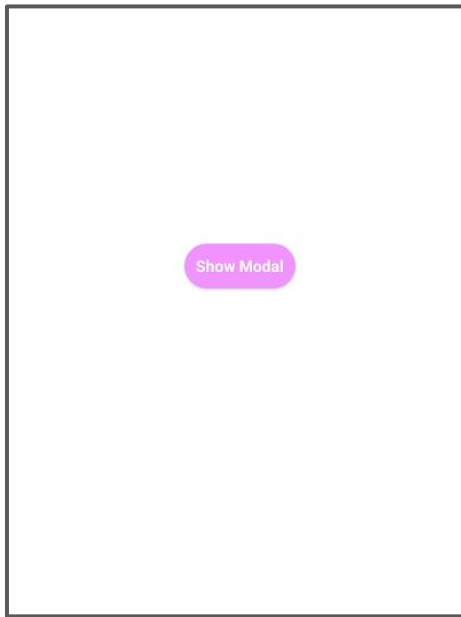
Modal

Modal

- Exibe um alerta customizado na tela
<https://reactnative.dev/docs/modal>

```
import React, {useState} from 'react';
import {Alert, Modal, StyleSheet, Text, Pressable, View} from 'react-native';

const App = () => {
  const [modalVisible, setModalVisible] = useState(false);
  return (
    <View style={styles.centeredView}>
      <Modal
        animationType="slide"
        transparent={true}
        visible={modalVisible}
        onRequestClose={() => {
          Alert.alert('Modal has been closed.');
```



View

View

- É um container, semelhante a uma div do HTML:
<https://reactnative.dev/docs/view>

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

const ViewBoxesWithColorAndText = () => {
  return (
    <View
      style={{
        flexDirection: 'row',
        height: 100,
        padding: 20,
      }}>
      <View style={{backgroundColor: 'blue', flex: 0.3}} />
      <View style={{backgroundColor: 'red', flex: 0.5}} />
      <Text>Hello World!</Text>
    </View>
  );
};

export default ViewBoxesWithColorAndText;
```



ScrollView

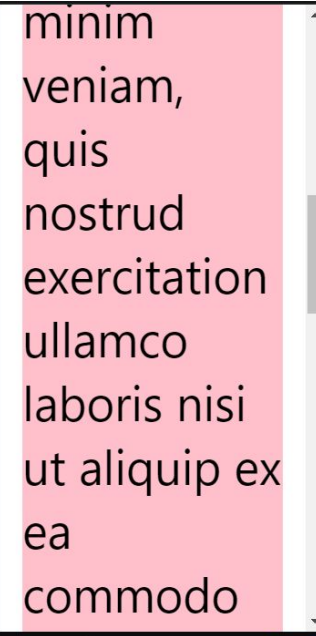
ScrollView

- Serve para rolar componentes na tela. De preferência use FlatList pois é mais performática!
 - <https://reactnative.dev/docs/scrollview>

```
import React from 'react';
import {
  StyleSheet,
  Text,
  SafeAreaView,
  ScrollView,
  StatusBar,
} from 'react-native';

const App = () => {
  return (
    <SafeAreaView style={styles.container}>
      <ScrollView style={styles.scrollView}>
        <Text style={styles.text}>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
          enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
          nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
          in
          culpa qui officia deserunt mollit anim id est laborum.
        </Text>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
```



SafeAreaView

SafeAreaView

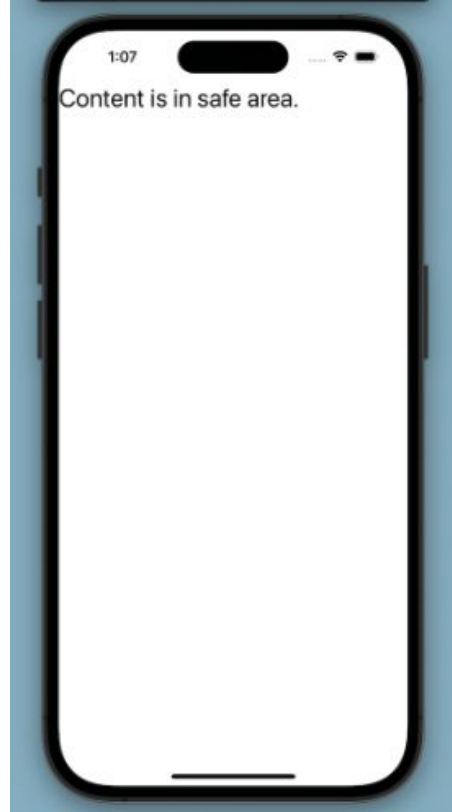
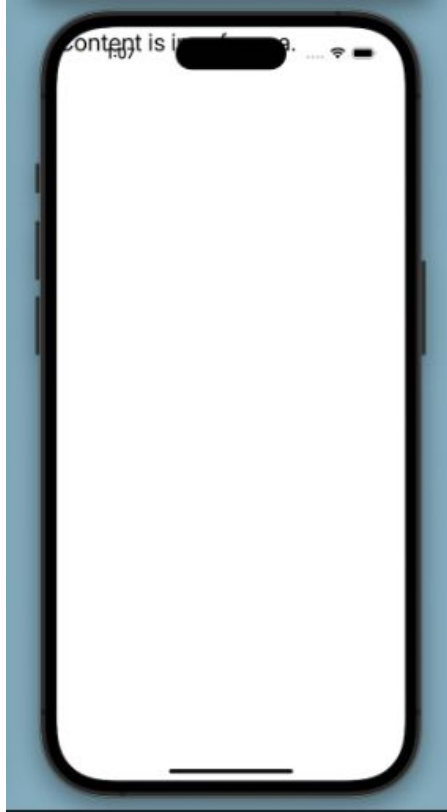
- É um componente que ajusta o layout do aplicativo para evitar que elementos fiquem ocultos pela barra de status
 - O React Native fornece um recurso que funciona somente com o iOS:
<https://reactnative.dev/docs/safeareaview>
 - Recomendamos utilizar a biblioteca react-native-safe-area-context que funciona para ambos:
<https://github.com/th3rdwave/react-native-safe-area-context>
 - *`npx expo install react-native-safe-area-context`*

SafeAreaView

```
import { SafeAreaView } from 'react-native-safe-area-context';

function SomeComponent() {
  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: 'red' }}>
      <View style={{ flex: 1, backgroundColor: 'blue' }} />
    </SafeAreaView>
  );
}
```


SafeAreaView



Icons

Icons

- Recomendo usar a biblioteca já disponível no Expo:
expo/vector-icons
 - <https://icons.expo.fyi/Index>

```
import * as React from 'react';
import { View, StyleSheet } from 'react-native';
import Ionicons from '@expo/vector-icons/Ionicons';

export default function App() {
  return (
    <View style={styles.container}>
      <Ionicons name="md-checkmark-circle" size={32} color="green" />
    </View>
  );
}
```



battery

Feather

1. Import the icon family

COPY

```
import { Feather } from '@expo/vector-icons';
```

2. Render the component

COPY

```
<Feather name="battery" size={24} color="black" />
```

Estilização

Estilização

- Para estilizar os componentes React Native utilizamos o CSS.
- Podemos utilizar CSS in line, ou seja, diretamente no próprio componente:
 - `<Text style={{backgroundColor: 'red'}}>just red</Text>`

Estilização

- Ou utilizar uma ferramenta chamada StyleSheet.create do React Native
- <https://reactnative.dev/docs/stylesheet>

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

const LotsOfStyles = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.red}>just red</Text>
      <Text style={styles.bigBlue}>just bigBlue</Text>
      <Text style={[styles.bigBlue, styles.red]}>bigBlue, then red</Text>
      <Text style={[styles.red, styles.bigBlue]}>red, then bigBlue</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    marginTop: 50,
  },
  bigBlue: {
    color: 'blue',
    fontWeight: 'bold',
    fontSize: 30,
  },
  red: {
    color: 'red',
  },
});

export default LotsOfStyles;
```

just red
just bigBlue
bigBlue, then red
red, then bigBlue

Estilização

- O padrão de posicionamento no React Native é o Flexbox!
- O Flexbox funciona no React Native da mesma maneira que no CSS na web, com algumas exceções.
- Os padrões são diferentes, com `flexDirection` em `column` ao invés de `row`, `alignContent` em `flex-start` ao invés de `stretch` e `flexShrink` em `0` invés de `1`.

Styled-Components

Styled-Components

- É uma biblioteca que utiliza o conceito de CSS-in-JS, ou seja, nos permite escrever códigos CSS dentro do Javascript.

Iniciando

Iniciando

- Primeiramente precisamos instalar a biblioteca:
 - `npm install styled-components`

Aplicando estilos

Aplicando estilos

- Com o projeto recém criado, no arquivo App.tsx vamos criar um estilo de modelo
- Vamos apagar o conteúdo que está no arquivo e escrever nosso código

Aplicando estilos



```
1  import styled from "styled-components/native";
2
3  const Container = styled.View`
4    flex: 1;
5    background-color: #2b58de;
6    align-items: center;
7    justify-content: center;
8  `;
9
10 const StyledText = styled.Text`
11   color: #fff;
12   font-size: 20px;
13 `;
14
15 export default function App() {
16   return (
17     <Container>
18       <StyledText>View criada com Styled Components</StyledText>
19     </Container>
20   );
21 }
```

Arquivos separados

Arquivos separados

- Podemos também criar nossos styled-components em arquivos separados.
- Por exemplo, podemos criar uma pasta chamada **components** e dentro dela um arquivo **View.ts** e um **Text.ts**
- E antes da const de declaração, precisamos inserir o “export”.

Arquivos separados



```
1 import styled from "styled-components/native";
2
3 export const Container = styled.View`
4   flex: 1;
5   background-color: #2b58de;
6   align-items: center;
7   justify-content: center;
8 `;
```



```
1 import styled from "styled-components/native";
2
3 export const StyledText = styled.Text`
4   color: #fff;
5   font-size: 20px;
6 `;
```

Arquivos separados



```
1 import { Container } from "../src/components/View";
2 import { StyledText } from "../src/components/Text";
3
4 export default function App() {
5   return (
6     <Container>
7       <StyledText>
8         View criada com Styled Components em arquivos separados
9       </StyledText>
10    </Container>
11  );
12 }
```



Material Complementar

Material Complementar

- Quando estamos construindo nosso app, precisamos nos preocupar com a interface do usuário.
- É preciso entender a melhor disposição dos elementos em tela, tamanhos e cores.

Material Complementar

- <https://medium.com/mackmobile/paleta-de-cores-e-ui-design-bddd71ecca39>
- <https://dribbble.com/resources/choose-colors-mobile-app-design>
- <https://madeinweb.com.br/psicologia-das-cores-influencia-aplicativo/>

Material Complementar

- Paleta de cores:
 - <https://paletadecores.com/>
 - <https://colors.co/>
 - <https://mycolor.space/>
 - <https://colordrop.io/>
 - <https://colorhunt.co/>

Dúvidas?



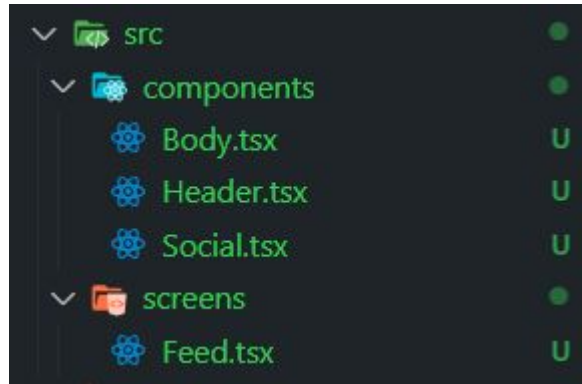
Tarefa

- Reproduzir uma tela de *feed* de um aplicativo de rede social (Facebook, Twiter - X, etc)
- Usar obrigatoriamente os componentes *text*, *image*, *flatList* e *icons*
- Estilizar com stylesheet ou styled-components
- Não esqueça de componentizar seu projeto!
- Utilize a url <https://source.unsplash.com/random> para carregar imagens aleatórias

Tarefa



Tarefa



Tarefa

```
const data = [
  {
    id: "1",
    avatar: "https://avatars.githubusercontent.com/u/42684330?v=4",
    username: "Rafael Kasper",
    comments: 10,
    reposts: 99,
    likes: 1200,
    text: "Conteúdo da postagem 1",
    image: "https://www.senacrs.com.br/assets/images/senac_logo_new.png",
  },
  {
    id: "2",
    avatar:
      "https://robohash.org/e174e9364d38487c8d533c53cf8b6048?set=set4&bgset=&size=400x400",
    username: "Gabriel",
    comments: 8,
    reposts: 54,
    likes: 10,
    text: "Conteúdo da postagem 2",
    image: "https://reactnative.dev/img/tiny_logo.png",
  },
];
```

```
<SafeAreaView style={{ flex: 1 }}>
  <View style={styles.container}>
    <FlatList
      data={data}
      renderItem={({ item }) => (
        <Feed
          avatar={item.avatar}
          username={item.username}
          comments={item.comments}
          reposts={item.reposts}
          likes={item.likes}
          text={item.text}
          image={item.image}
        />
      )}
      keyExtractor={(item) => item.id}
    />
  </View>
</SafeAreaView>
```

Tarefa

```
const Feed: FC<Props> = ({
  avatar,
  username,
  text,
  image,
  comments,
  reposts,
  likes,
}) => {
  return (
    <View style={styles.container}>
      <Header avatar={avatar} username={username} />
      <Body text={text} image={image} />
      <Social comments={comments} reposts={reposts} likes={likes} />
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  container: {
    alignItems: "center",
    justifyContent: "center",
    marginVertical: 10,
    borderRadius: 5,
    margin: 5,
    width: "95%",
    padding: 20,
    shadowColor: "#fff",
    shadowOffset: {
      width: 0,
      height: 2,
    },
    shadowOpacity: 0.25,
    shadowRadius: 3,
    elevation: 5,
  },
});
```