



# Zero um cursos

## Seja o primeiro

[zeroumcursos.com](http://zeroumcursos.com)

### REACT NATIVE APPS

#### ANIMAÇÃO

## REACT NATIVE APPS

### ANIMAÇÃO

Sobre o Autor:

Éder Pires Batista

[ederpbj@gmail.com](mailto:ederpbj@gmail.com)

<https://www.linkedin.com/in/eder-pires-batista-a46b867b/>

Graduado em Sistemas Para Internet pelo Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - (2017) e graduação em Ciências Biológicas pela Universidade de Pernambuco (2012), Pós-graduado em Banco de Dados e Bussines Intelligence, Pós-graduado em Projetos e Desenvolvimento de Jogos Digitais, com experiência em docência na disciplina de Programação de Jogos Digitais. Atualmente é analista de sistemas no Centro Integrado de Comando e Controle da PB. Com experiência na área de Ciência da Computação, com ênfase em Análise de dados (bigdata) e desenvolvimento de jogos e aplicativos, atuando principalmente no seguinte tema: segurança pública, educação, desenvolvimento de aplicativos e jogos digitais.

## **Prefácio**

Bem-vindo ao fascinante mundo do React Native! Este manual foi concebido para guiá-lo através das complexidades e possibilidades desta poderosa tecnologia de desenvolvimento de aplicativos móveis. À medida que a demanda por soluções eficientes e rápidas para o desenvolvimento de aplicativos móveis continua a crescer, React Native emerge como uma ferramenta fundamental, permitindo aos desenvolvedores criar experiências de usuário excepcionais em plataformas iOS e Android, utilizando uma única base de código.

Neste manual, exploraremos desde os conceitos fundamentais até técnicas avançadas, proporcionando uma compreensão abrangente do React Native. Se você é um desenvolvedor iniciante, encontrará um guia claro e progressivo para começar sua jornada no desenvolvimento de aplicativos móveis. Para aqueles mais experientes, há profundidade nas discussões sobre arquitetura, boas práticas e estratégias avançadas para otimizar o desempenho e a manutenibilidade de seus aplicativos.

## Sumário

1. Primeiros passos .....	4
1.1 Repositório .....	4
1.2 Documentação .....	4
1.3 Iniciando Código da animação .....	4
1.4 Animação de esticar .....	5
1.5 Rodar o projeto.....	5
1.6 Diagnostico de erro .....	5
1.7 Primeira animação, altura.....	6
2. Animações em paralelo e sequência .....	7
2.1 Sequência de animações .....	7
2.2 Animação em paralelo.....	9
2.3 Usando os dois tipos de animação.....	10
3. Animações em loop .....	11
4. Aprendendo Interpolações .....	13
5. Usando Animatable .....	18
5.1 Projeto base App.js.....	18
5.2 Git-animatable .....	18
5.3 Instalando animatable .....	18
5.4 Testando no projeto .....	19
5.5 Inserindo animação do git animatable .....	22
5.6 Definindo função para chamar animação .....	23
6. Referências.....	25

## 1. Primeiros passos

### 1.1 Repositório

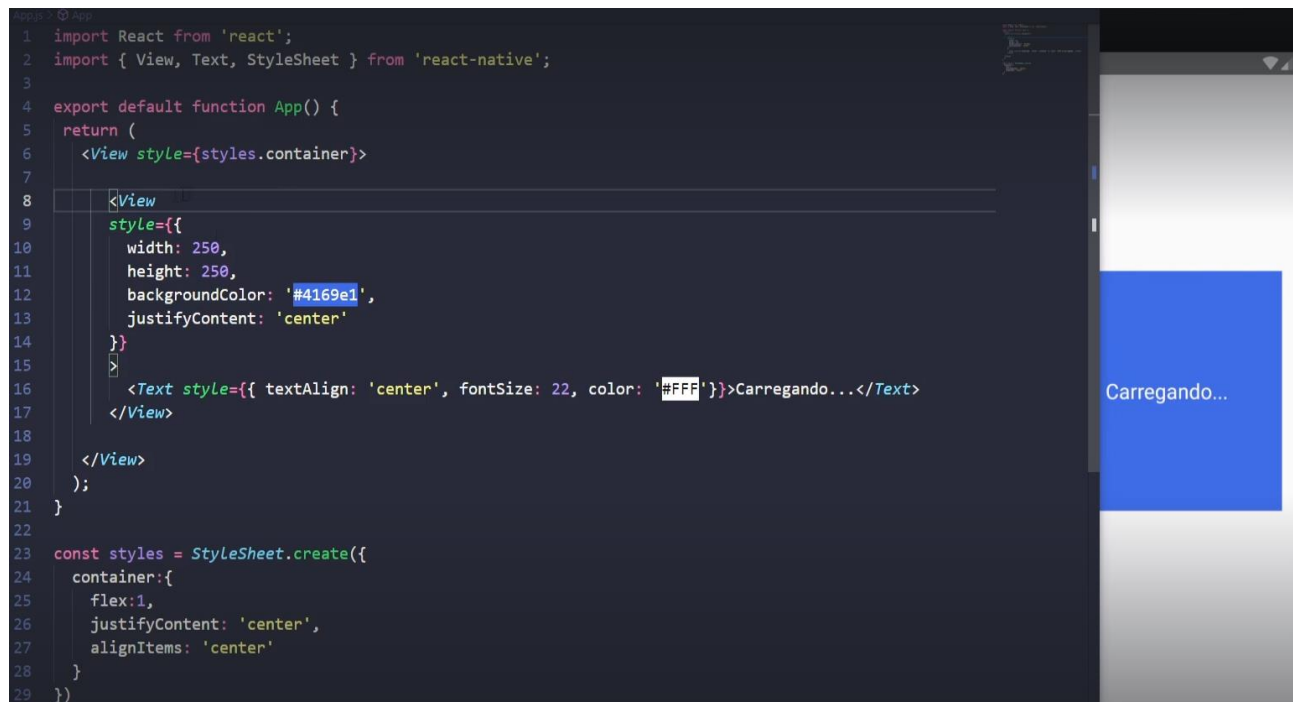
<https://github.com/ederPires/animacao>

### 1.2 Documentação

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

### 1.3 Iniciando Código da animação

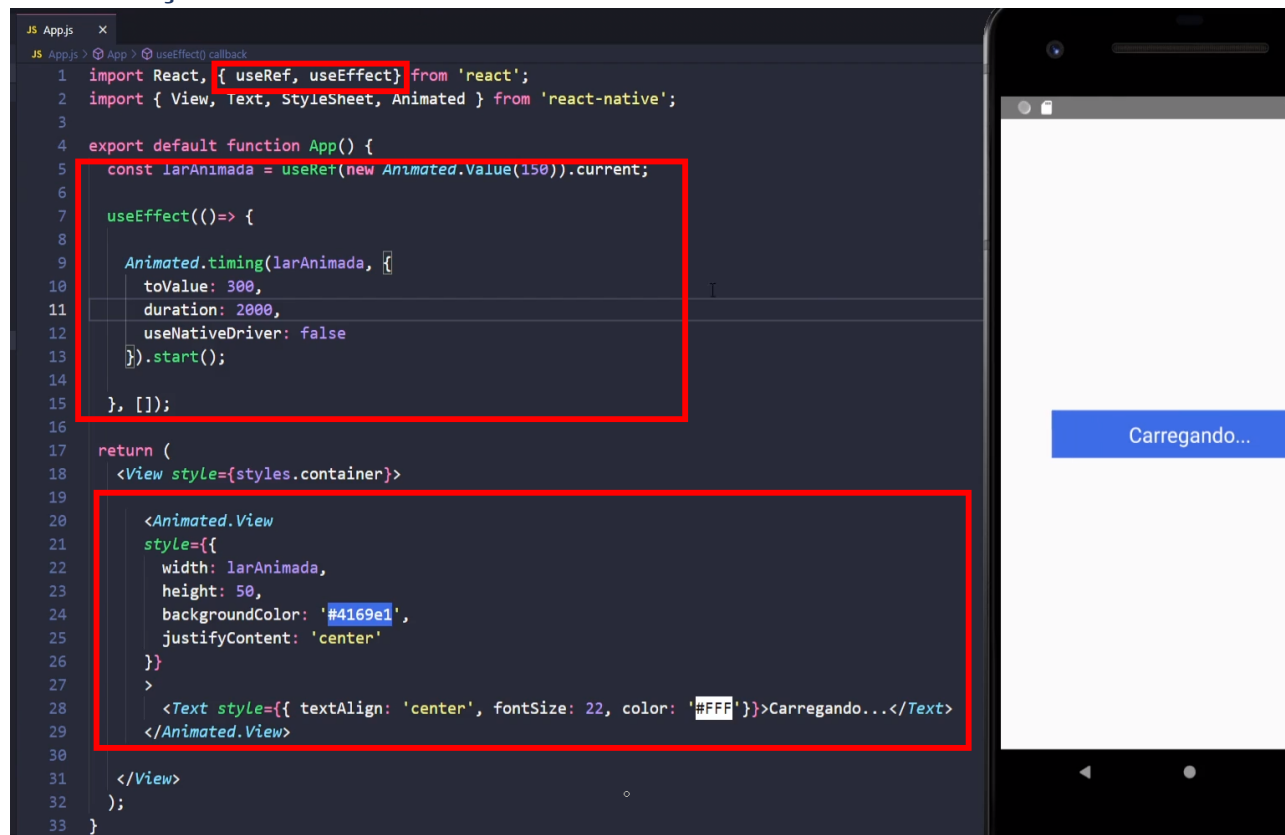
App.js



Passa através de propriedades o que queremos animar.

- Não é recomendado usar `useState` em animação. E sim `useRef`

## 1.4 Animação de esticar



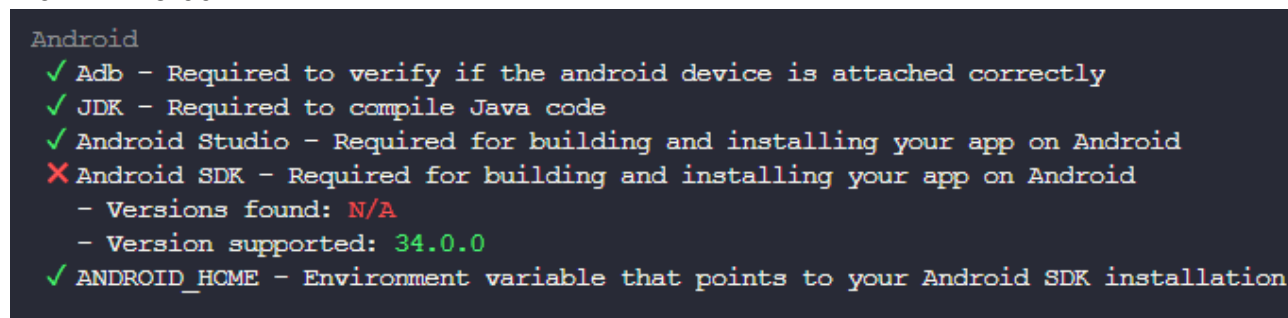
## 1.5 Rodar o projeto

- `npx react-native start`

## 1.6 Diagnostico de erro

- `npx react-native doctor`
- Para corrigir o erro basta clicar na letra E

### 1.6.1 Erro 001



## 1.7 Primeira animação, altura

Alterando para animar a altura do view.

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

export default function App() {
  const larAnimada = useRef(new Animated.Value(150)).current;
  const altAnimada = useRef(new Animated.Value(50)).current;

  useEffect(() => {
    Animated.timing(altAnimada, {
      toValue: 300,
      duration: 2000,
      useNativeDriver: false
    }).start();
  }, []);

  return (
    <View style={styles.container}>
      <Animated.View
        style={{
          width: larAnimada,
          height: altAnimada,
          backgroundColor: '#4169e1',
          justifyContent: 'center'
        }}
      >
        <Text style={{ textAlign: 'center', fontSize: 22, color: '#FFF' }}>Carregando...</Text>
      </Animated.View>
    </View>
  );
}
```

## 2. Animações em paralelo e sequência

### 2.1 Sequência de animações

Para criar uma sequência de animações basta incluir este código. São executadas na sequência.

```
1  import React, { useRef, useEffect } from 'react';
2  import { View, Text, StyleSheet, Animated } from 'react-native';
3
4  export default function App() {
5    const larAnimada = useRef(new Animated.Value(150)).current;
6    const altAnimada = useRef(new Animated.Value(50)).current;
7
8    useEffect(() => {
9
10     Animated.sequence([
11       Animated.timing(larAnimada, {
12         toValue: 300,
13         duration: 2000,
14         useNativeDriver: false
15       }),
16       Animated.timing(altAnimada, {
17         toValue: 200,
18         duration: 2000,
19         useNativeDriver: false
20       })
21     ]).start();
22
23   }, []);
24
25   return (
26     <View style={styles.container}>
```



Incluindo mais uma animação na sequência.

```
const altAnimada = useRef(new Animated.Value(50)).current;  
const opacidadeAnimada = useRef(new Animated.Value(1)).current;
```

```
    Animated.timing(opacidadeAnimada, {  
      toValue: 0,  
      duration: 1000,  
      useNativeDriver: false  
    })  
  ].start();  
  
}, []);  
  
return (  
  <View style={styles.container}>  
  
    <Animated.View  
      style={{  
        width: larAnimada,  
        height: altAnimada,  
        backgroundColor: '#4169e1',  
        justifyContent: 'center',  
        opacity: opacidadeAnimada  
      }}  
    >
```

## 2.2 Animação em paralelo

São executadas, as animações, ao mesmo tempo.

```
9      useEffect(() => {[
10
11        Animated.parallel([
12          Animated.timing(larAnimada, {
13            toValue: 300,
14            duration: 2000,
15            useNativeDriver: false
16          }),
17          Animated.timing(altAnimada, {
18            toValue: 200,
19            duration: 2000,
20            useNativeDriver: false
21          })
22        ]).start();
23      }], []);
24
25
26    return (
27      <View style={styles.container}>
28
29        <Animated.View
30          style={{
31            width: larAnimada,
32            height: altAnimada,
33            backgroundColor: '#4169e1',
34            justifyContent: 'center',
35            opacity: opacidadeAnimada
36          }}
37        >
38          <Text style={{ textAlign: 'center', fontSize: 22, color: '#FFF' }}>Carregand
39        </Animated.View>
```

## 2.3 Usando os dois tipos de animação

```

1 import React, { useRef, useEffect } from 'react';
2 import { View, Text, StyleSheet, Animated } from 'react-native';
3
4 export default function App() {
5   const larAnimada = useRef(new Animated.Value(150)).current;
6   const altAnimada = useRef(new Animated.Value(50)).current;
7   const opacidadeAnimada = useRef(new Animated.Value(0)).current;
8 }

```

```

useEffect(() => {
  Animated.sequence([
    Animated.timing(opacidadeAnimada, {
      toValue: 1,
      duration: 2000,
      useNativeDriver: false
    }),
    Animated.parallel([
      Animated.timing(larAnimada, {
        toValue: 300,
        duration: 2000,
        useNativeDriver: false
      }),
      Animated.timing(altAnimada, {
        toValue: 300,
        duration: 1000,
        useNativeDriver: false
      })
    ])
  ]).start();
}, []);

```

Para ficar com o mesmo tempo, basta alterar o valor duration

```

Animated.parallel([
  Animated.timing(larAnimada, {
    toValue: 300,
    duration: 2000,
    useNativeDriver: false
  }),
  Animated.timing(altAnimada, {
    toValue: 300,
    duration: 2000,
    useNativeDriver: false
  })
])

```

Podemos incluir mais uma animação para sumir.

```

    }),

    Animated.timing(opacidadeAnimada, {
      toValue: 0,
      duration: 2000,
      useNativeDriver: false
    })
  })

```

### 3. Animações em loop

Para fazer animação em loop é simples basta utilizar a função `Animated.loop()`. Vai esticar a caixa azul e voltar a forma original.

Lembrar de remover a opacity 0, que deixa invisível.

```

import React, { useRef, useEffect } from 'react';
import { View, Text, StyleSheet, Animated } from 'react-native';

export default function App() {
  const larAnimada = useRef(new Animated.Value(150)).current;
  const altAnimada = useRef(new Animated.Value(50)).current;

  useEffect(() => {
    Animated.loop([
      Animated.timing(larAnimada, {
        toValue: 300,
        duration: 2000,
        useNativeDriver: false
      })
    ]).start();
  }, []);

  return (
    <View style={styles.container}>

      <Animated.View
        style={{
          width: larAnimada,
          height: altAnimada,
          backgroundColor: '#4169e1',
          justifyContent: 'center',
        }}
      />

      <Text style={{ textAlign: 'center', fontSize: 22, color: 'white' }}>Carregando...</Text>
    </Animated.View>
  </View>

```

Carregando...

Agora vamos fazer o loop junto com a sequência. Onde aparece a animação de esticar e encolher a caixa.

```
useEffect(() => {
  Animated.loop(
    Animated.sequence([
      Animated.timing(larAnimada, {
        toValue: 300,
        duration: 2000,
        useNativeDriver: false
      }),
      Animated.timing(larAnimada, {
        toValue: 150,
        duration: 2000,
        useNativeDriver: false
      })
    ])
  ).start();
}, []);
```

Podemos arredondar as bordas com borderRadius

```
return (
  <View style={styles.container}>
    .
    <Animated.View
      style={{
        width: larAnimada,
        height: altAnimada,
        backgroundColor: '#4169e1',
        justifyContent: 'center',
        borderRadius: 50
      }}
    >
      <Text style={{ textAlign: 'center', fontSize: 22, color: '#FFF' }}>Carregando...</Text>
    </Animated.View>
  </View>
);
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center'
  }
});
```

Borda arredondada com efeito sanfona.



## 4. Aprendendo Interpolações

Ajuste o projeto dessa fora:

```
export default function App() {
  const larAnimada = useRef(new Animated.Value(150)).current;
  const altAnimada = useRef(new Animated.Value(50)).current;

  useEffect(() => {

    Animated.timing(larAnimada, {
      toValue: 300,
      duration: 2000,
      useNativeDriver: false
    }).start();

  }, []);
}
```

```
return (
  <View style={styles.container}>

    <Animated.View
      style={{
        width: larAnimada,
        height: altAnimada,
        backgroundColor: '#4169e1',
        justifyContent: 'center',
      }}
    >

    </Animated.View>

  </View>
);
}
```



Vamos ajustar para pegar a tela inteira na animação.

```
export default function App() {
  const larAnimada = useRef(new Animated.Value(0)).current;
  const altAnimada = useRef(new Animated.Value(50)).current;
```

Vamos trabalhar com interpolação para que seja utilizada 100% da tela.

```
export default function App() {
  const larAnimada = useRef(new Animated.Value(0)).current;
  const altAnimada = useRef(new Animated.Value(50)).current;

  useEffect(() => {

    Animated.timing(larAnimada, {
      toValue: 100,
      duration: 4000,
      useNativeDriver: false
    }).start();

  }, []);

  let porcentagemLargura = larAnimada.interpolate({
    inputRange: [0, 100], //Entrada
    outputRange: ['0%', '100%'] //Vai sair 0% até 100%
  })

  return (
    <View style={styles.container}>

      <Animated.View
        style={{
          width: porcentagemLargura,
          height: altAnimada,
          backgroundColor: '#4169e1',
          justifyContent: 'center',
        }}
      >
```

Agora vamos fazer a altura animada até 100%, executando em sequência.

```
useEffect(() => {

  Animated.sequence([
    Animated.timing(larAnimada, {
      toValue: 100,
      duration: 4000,
      useNativeDriver: false
    }),

    Animated.timing(altAnimada, { ←
      toValue: 100,
      duration: 4000,
      useNativeDriver: false
    })
  ]).start();

}, []);
```

```
let porcentagemLargura = larAnimada.interpolate({
  inputRange: [0, 100], //Entrada
  outputRange: ['0%', '100%'] //Vai sair 0% até 100%
})

let porcentagemAltura = altAnimada.interpolate({ ←
  inputRange: [50, 100],
  outputRange: ['5%', '100%']
})

return (
  <View style={styles.container}>

    <Animated.View
      style={{
        width: porcentagemLargura,
        height: porcentagemAltura, ←
        backgroundColor: '#4169e1',
        justifyContent: 'center',
      }}
    >

    </Animated.View>

  </View>
);
```

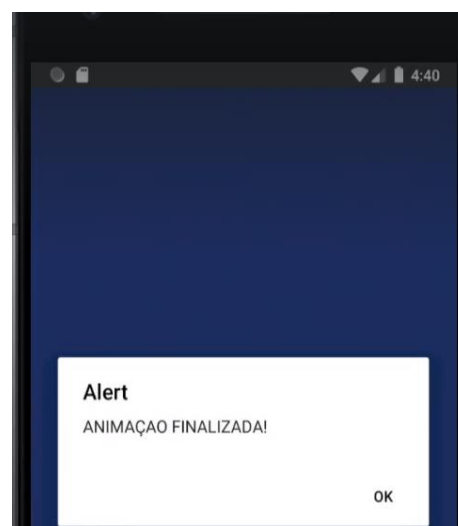


Dentro do retorno posso colocar meus componentes.

```
return (
  <View style={styles.container}>
    <Animated.View
      style={{
        width: porcentagemLargura,
        height: porcentagemAltura,
        backgroundColor: '#4169e1',
        justifyContent: 'center',
      }}
    >
      <Text>TESTE</Text>
    </Animated.View>
  </View>
);
}
```

Como saber quando minha animação vai ser finalizada. Dentro do start() pode ser colocada uma função anônima que só é chamada após finalizar a animação.

```
Animated.timing(altAnimada, {
  toValue: 100,
  duration: 4000,
  useNativeDriver: false
}).start( () => {
  alert('ANIMAÇÃO FINALIZADA!');
});
```



Também pode ser usado o `finished`, se finalizou ele recebe `true`.

```

    Animated.timing(altAnimada, {
      toValue: 100,
      duration: 4000,
      useNativeDriver: false
    })
  ]).start(({ finished }) => {
    alert('ANIMAÇÃO FINALIZADA!');
    console.log(finished);
  });

}, []);

```

```

useEffect(() => {

  Animated.timing(larAnimada, {
    toValue: 100,
    duration: 4000,
    useNativeDriver: false
  }).start(() => {
    //só é chamada quando a animação finalizar!
  });
});

```

## 5. Usando Animatable

### 5.1 Projeto base App.js

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

export default function App() {

  return (
    <View style={styles.container}>
      <Text>Sujeito Programador</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center'
  }
})
```

Para isso vamos usar a biblioteca, react-native-animatable.

### 5.2 Git-animatable

<https://github.com/oblador/react-native-animatable>

### 5.3 Instalando animatable

```
C:\Windows\system32\cmd.exe

D:\react-native\curso\animacao>npm install react-native-animatable_
```

## 5.4 Testando no projeto

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

import * as Animatable from 'react-native-animatable';

export default function App() {

  return (
    <View style={styles.container}>
      <Animatable.Text
        style={styles.title}
        animation="pulse"
      >
        Sujeito Programador
      </Animatable.Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container:{
    flex:1,
    justifyContent: 'center',
    alignItems: 'center'
  },
  title:{
    fontSize: 25,
  }
})
```

Posso definir quantas vezes a animação executa com `iterationCount`.

```
return (
  <View style={styles.container}>
    <Animatable.Text
      style={styles.title}
      animation="bounce"
      iterationCount={3}
    >
      Sujeito Programador
    </Animatable.Text>
  </View>
);
```

Também posso passar infinity, e será executado infinitamente.

```
animation="bounce"  
iterationCount={Infinity}  
>
```

Outra configuração é a duração de execução, usando o duration.

```
<Animatable.Text  
  style={styles.title}  
  animation="bounce"  
  duration={5000}  
  >  
    Sujeito Programador  
</Animatable.Text>
```

Usando animação em um botão **TouchableOpacity**.

```
JS App.js x
JS App.js > App
10 <Animatable.Text
11   style={styles.title}
12   animation="shake"
13 >
14   Sujeito Programador
15 </Animatable.Text>
16
17 <TouchableOpacity style={styles.button}>
18   <Text style={{ color: '#FFF' }}>Animar</Text>
19 </TouchableOpacity>
20 </View>
21 );
22 }
23
24 const styles = StyleSheet.create({
25   container:{
26     flex:1,
27     justifyContent: 'center',
28     alignItems: 'center'
29   },
30   title:{
31     fontSize: 25,
32   },
33   button:{
34     width: '70%',
35     height: 30,
36     backgroundColor: '#121212',
37     justifyContent: 'center',
38     alignItems: 'center',
39     marginTop: 25
40   }
41 })
```

Para isso precisamos criar um componente de animação.

```

4 import * as Animatable from 'react-native-animatable';
5
6 const ButtonAnimated = Animatable.createAnimatableComponent(TouchableOpacity);
7
8 export default function App() {
9
10   return (
11     <View style={styles.container}>
12       <Animatable.Text
13         style={styles.title}
14         animation="shake"
15       >
16         Sujeito Programador
17     </Animatable.Text>
18
19     <ButtonAnimated style={styles.button}>
20       <Text style={{ color: '#FFF' }}>Animar</Text>
21     </ButtonAnimated>
22   </View>
23 );
24 }
25
26 const styles = StyleSheet.create({
27   container:{
28     flex:1,
29     justifyContent: 'center',
30     alignItems: 'center'
31   },
32   title:{
33     fontSize: 25,

```

Sujeito Programador

Animar

## 5.5 Inserindo animação do git animatable

Sliding Entrances

slideInDown

slideInUp

slideInLeft

slideInRight



```

<ButtonAnimated style={styles.button} animation="slideInDown">
  <Text style={{ color: '#FFF' }}>Animar</Text>
</ButtonAnimated>
</View>
);
}

```



## 5.6 Definindo função para chamar animação

Podemos definir uma função para chamar a animação.

```

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

import * as Animatable from 'react-native-animatable';

const ButtonAnimated = Animatable.createAnimatableComponent(TouchableOpacity);





export default function App() {
  const buttonRef = useRef(null);

  function handleClick(){
    buttonRef.current.pulse();
  }

  return (
    <View style={styles.container}>
      <Animatable.Text
        style={styles.title}
        animation="shake"
      >
        Sujeito Programador
      </Animatable.Text>

      <ButtonAnimated
        style={styles.button}
        animation="pulse"
        ref={buttonRef}
        onPress={handleClick}
      >
        <Text style={{ color: '#FFF' }}>Animar</Text>
      </ButtonAnimated>
    </View>
  );
}

```



Aqui escolhemos qual animação executar.

```
function handleClick(){  
  buttonRef.current.bounce();  
}
```

```
function handleClick(){  
  buttonRef.current.shake();  
}
```

## 6. Referências

Fábrica de Aplicativos React Native, 2023. Disponível em: < <https://fabricadeapps.club>>. Acesso em: 03/01/2024.

Github react-native-animated, Disponível em: <<https://github.com/oblador/react-native-animated>> Acesso em: 04/01/2024.

React Native Animations, Disponível em: < <https://reactnative.dev/docs/animations>> Acesso em: 04/01/2024.