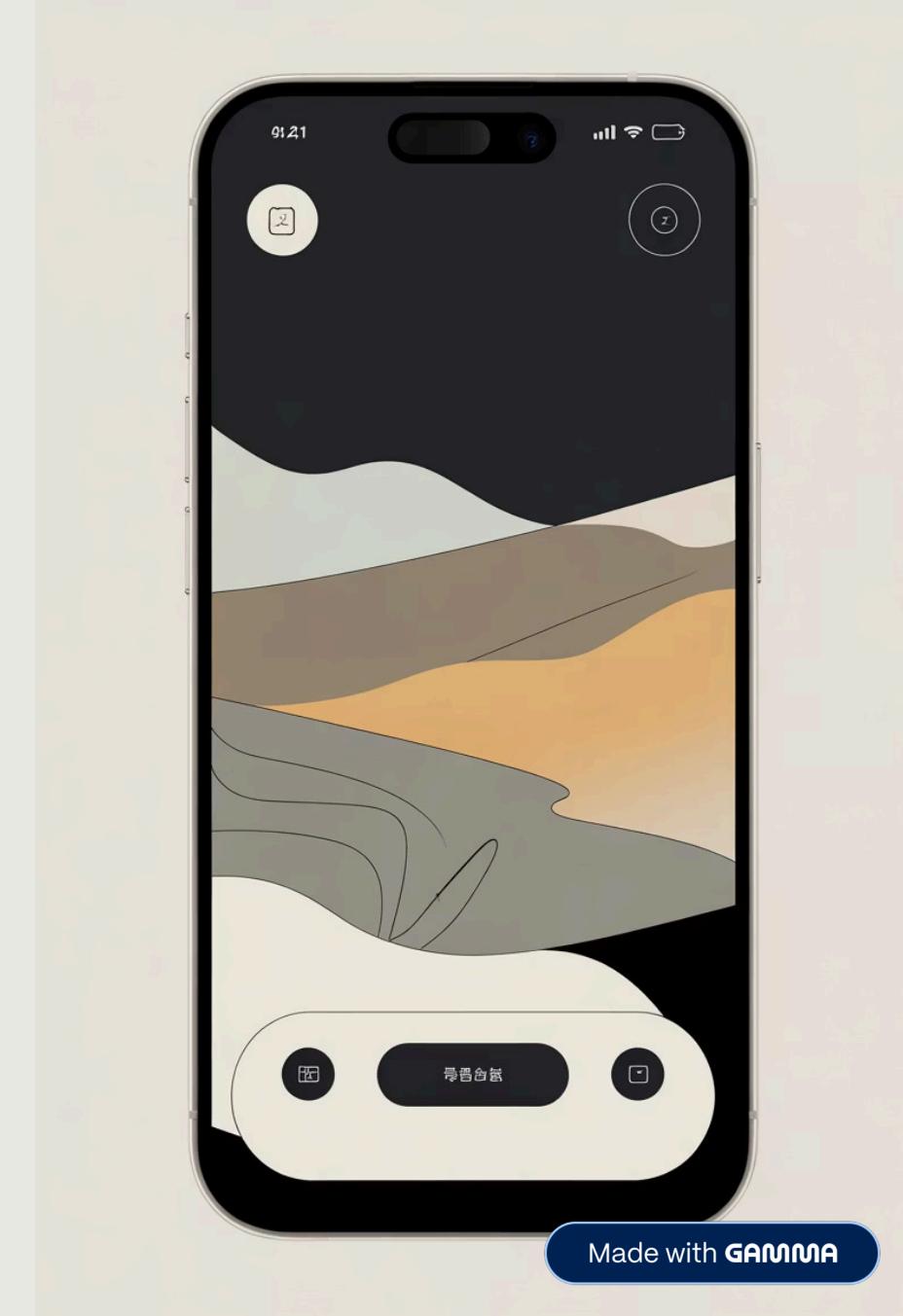


COMPONENTES ESSENCIAIS DO REACT NATIVE

Domine os blocos fundamentais para construir interfaces móveis profissionais e funcionais.



VIEW: O CONTAINER FUNDAMENTAL

PARA QUE SERVE?

A View é o componente básico de layout no React Native, equivalente à `<div>` no HTML. É o container que você usa para organizar e estruturar todos os elementos visuais na tela.

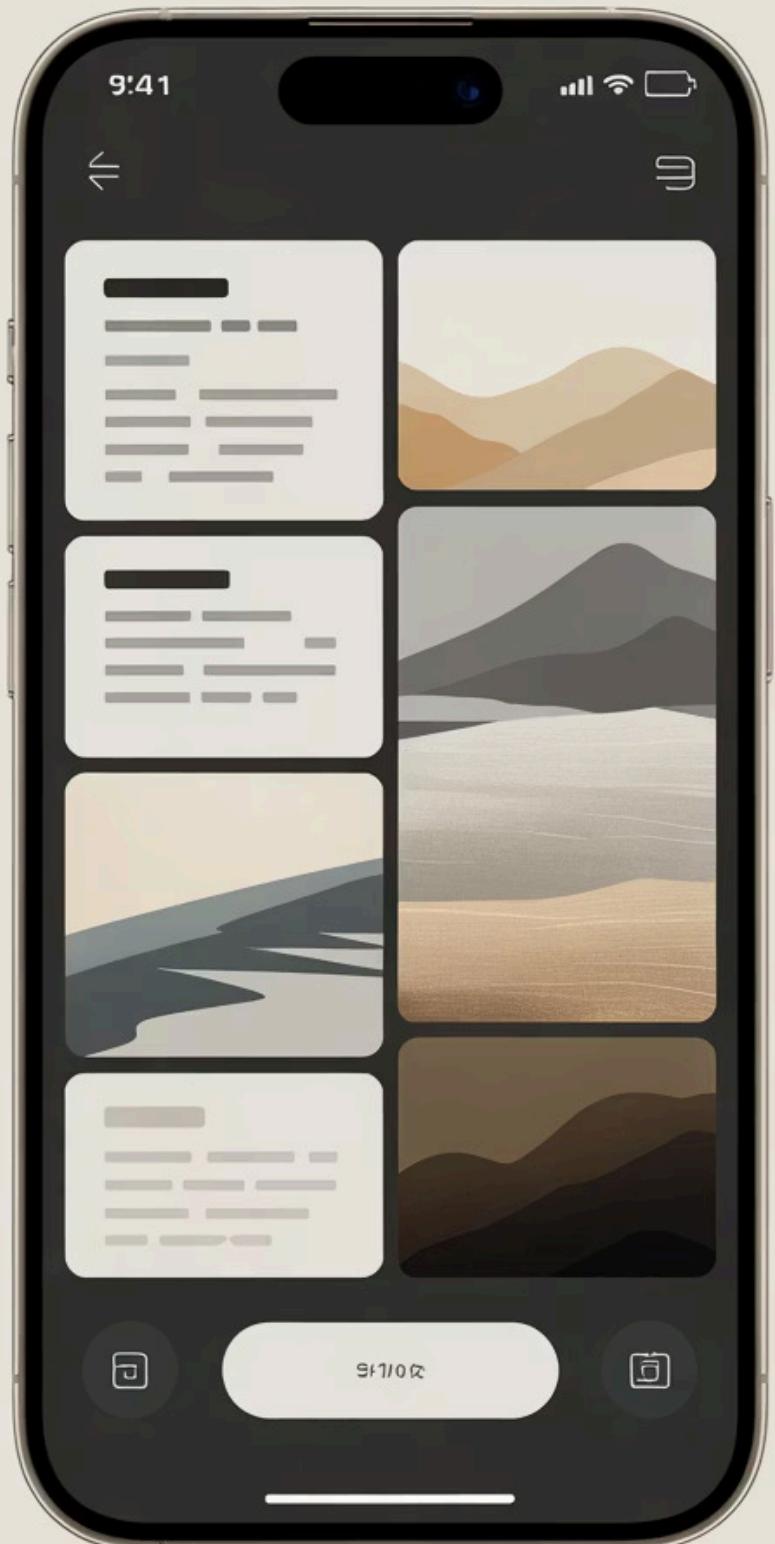
Pense nela como uma caixa invisível que agrupa conteúdo e define como ele será posicionado.

COMO UTILIZAR

```
import { View } from "react-native";

export default function App() {
  return (
    <View style={{
      backgroundColor: "lightgray",
      padding: 20
    }}>
      /* Conteúdo aqui */
    </View>
  );
}
```

TEXT E IMAGE: CONTEÚDO VISUAL



TEXT

Componente para exibir qualquer texto na tela. Todo conteúdo textual deve estar dentro de um componente Text.

```
<Text style={{  
  fontSize: 20,  
  color: "blue"  
}}>  
  Olá, React Native!  
</Text>
```

IMAGE

Exibe imagens locais ou da internet. Suporta formatos PNG, JPG, GIF e outros.

```
<Image  
  source={{  
    uri: "https://picsum.photos/200"  
  }}  
  style={{  
    width: 100,  
    height: 100  
  }}  
>
```



TEXTINPUT: CAPTURANDO DADOS DO USUÁRIO

O TextInput é o campo de entrada de texto, essencial para formulários, campos de busca, login e qualquer interação que exija digitação do usuário.

1

IMPORTAR COMPONENTE

```
import { TextInput } from "react-native";  
import { useState } from "react";
```

2

CRIAR ESTADO

```
const [nome, setNome] = useState("");
```

3

RENDERIZAR CAMPO

```
<TextInput  
style={{ borderWidth: 1, padding: 8 }}  
placeholder="Digite seu nome"  
value={nome}  
onChangeText={setNome}  
/>
```

TOUCHABLEOPACITY: BOTÕES PERSONALIZADOS



CRIE BOTÕES COM FEEDBACK VISUAL

O TouchableOpacity é perfeito para criar botões personalizados com efeito de opacidade quando pressionados, proporcionando uma experiência tátil natural.

```
<TouchableOpacity  
style={{  
backgroundColor: "blue",  
padding: 10  
}}  
onPress={() => alert("Clicou!")}  
>  
<Text style={{ color: "white" }}>  
Clique Aqui  
</Text>  
</TouchableOpacity>
```

BUTTON E SWITCH: CONTROLES NATIVOS

BUTTON

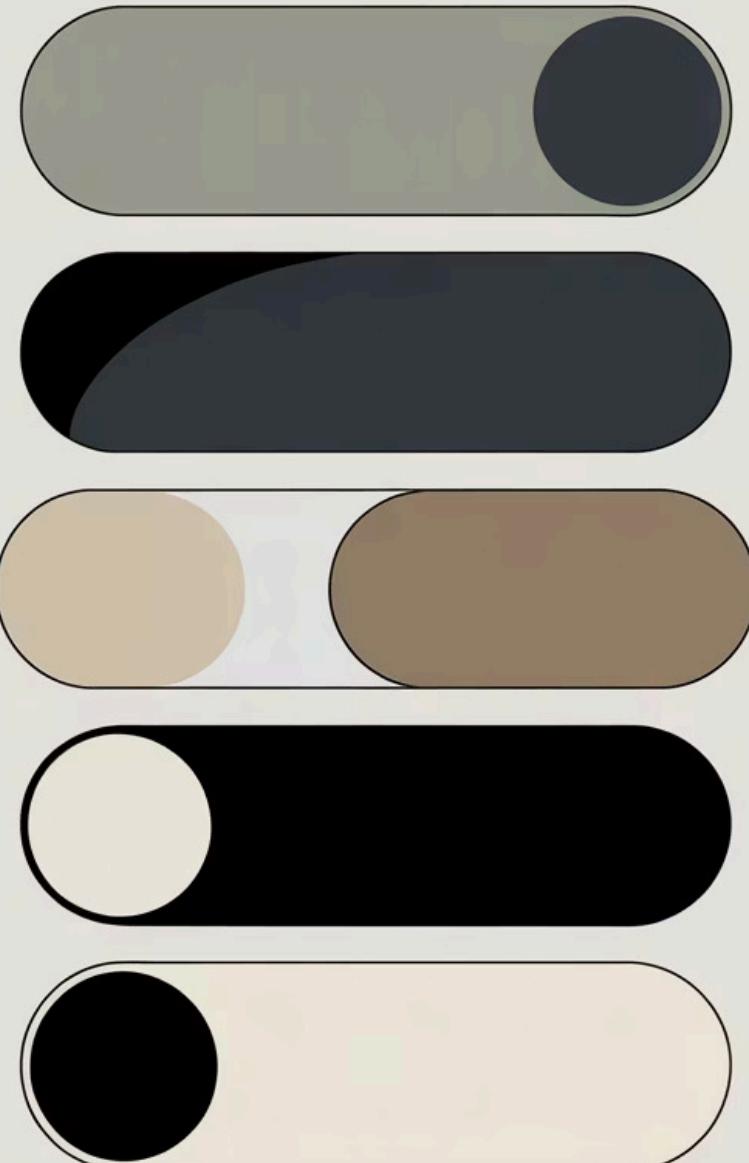
Botão nativo pronto para uso, sem necessidade de estilização complexa. Ideal para ações rápidas.

```
<Button  
    title="Salvar"  
    onPress={() => alert("Salvo!")}  
/>
```

SWITCH

Alternador on/off usado em configurações e preferências. Perfeito para ativar ou desativar funcionalidades.

```
const [ativo, setAtivo] = useState(false);  
  
<Switch  
    value={ativo}  
    onValueChange={setAtivo}  
/>
```



STYLESHEET: ORGANIZANDO ESTILOS

O StyleSheet permite criar e reutilizar estilos de forma organizada e eficiente, melhorando a performance e manutenção do código.

θ1

IMPORTAR STYLESHEET

Inclua o StyleSheet junto com outros componentes

θ2

CRIAR ESTILOS

Use StyleSheet.create() para definir seus estilos

θ3

APLICAR NOS COMPONENTES

Referencie os estilos usando styles.nomeDoEstilo

```
const styles = StyleSheet.create({
  container: {
    backgroundColor: "#eee",
    padding: 20,
  },
  title: {
    fontSize: 22,
    fontWeight: "bold",
  },
});

<View style={styles.container}>
  <Text style={styles.title}>Título</Text>
</View>
```



EXERCÍCIO 1: PRIMEIRA TELA



DESAFIO BÁSICO

Crie uma tela completa combinando os componentes aprendidos:

1

ADICIONE UM TEXT

Exiba a mensagem "Olá, React Native" com fonte de tamanho 24 e cor de sua escolha

2

INSIRA UMA IMAGE

Adicione uma imagem da internet usando o componente Image com dimensões 200x200

3

ORGANIZE COM VIEW

Use uma View para centralizar todos os elementos verticalmente e horizontalmente na tela

EXERCÍCIOS 2 E 3: INTERATIVIDADE

EXERCÍCIO 2: SAUDAÇÃO PERSONALIZADA

Crie um TextInput onde o usuário digite seu nome. Abaixo do campo, exiba dinamicamente:

"Olá, [nome digitado]"

Dica: Use useState para armazenar o nome e atualize o texto conforme o usuário digita.

EXERCÍCIO 3: VALIDAÇÃO DE ENTRADA

Adicione um botão (TouchableOpacity ou Button) que ao ser clicado:

- Mostra alert("Bem-vindo!") se o nome estiver preenchido
- Mostra alert("Digite seu nome primeiro") se o campo estiver vazio

Use condicionais para verificar o estado antes de exibir a mensagem.



EXERCÍCIO 4: MODO ESCURO

DESAFIO FINAL: TEMA DINÂMICO

Implemente um Switch para alternar entre modo claro e escuro. Quando o modo escuro estiver ativado:



FUNDO PRETO

O backgroundColor da View principal deve mudar para preto (#000000)



TEXTO BRANCO

Todos os componentes Text devem ter cor branca (#FFFFFF)



SWITCH CONTROLADO

Use useState para controlar o estado do tema e aplicar estilos condicionalmente

Dica: Crie dois objetos de estilo diferentes (claro e escuro) e alterne entre eles baseado no valor do Switch.

