

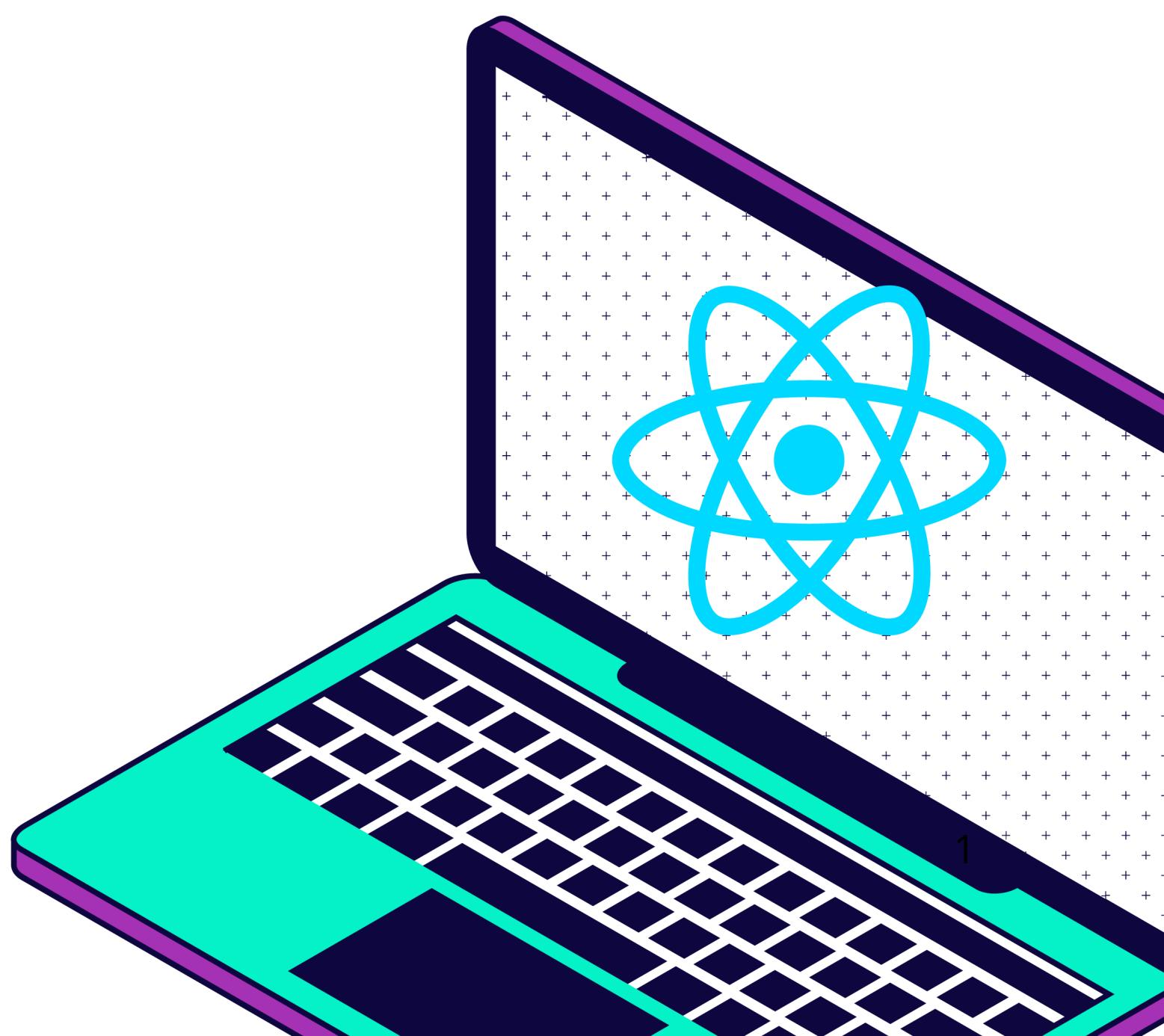
E-Book

# React Native

**Start Mobile: React Native  
Edition**

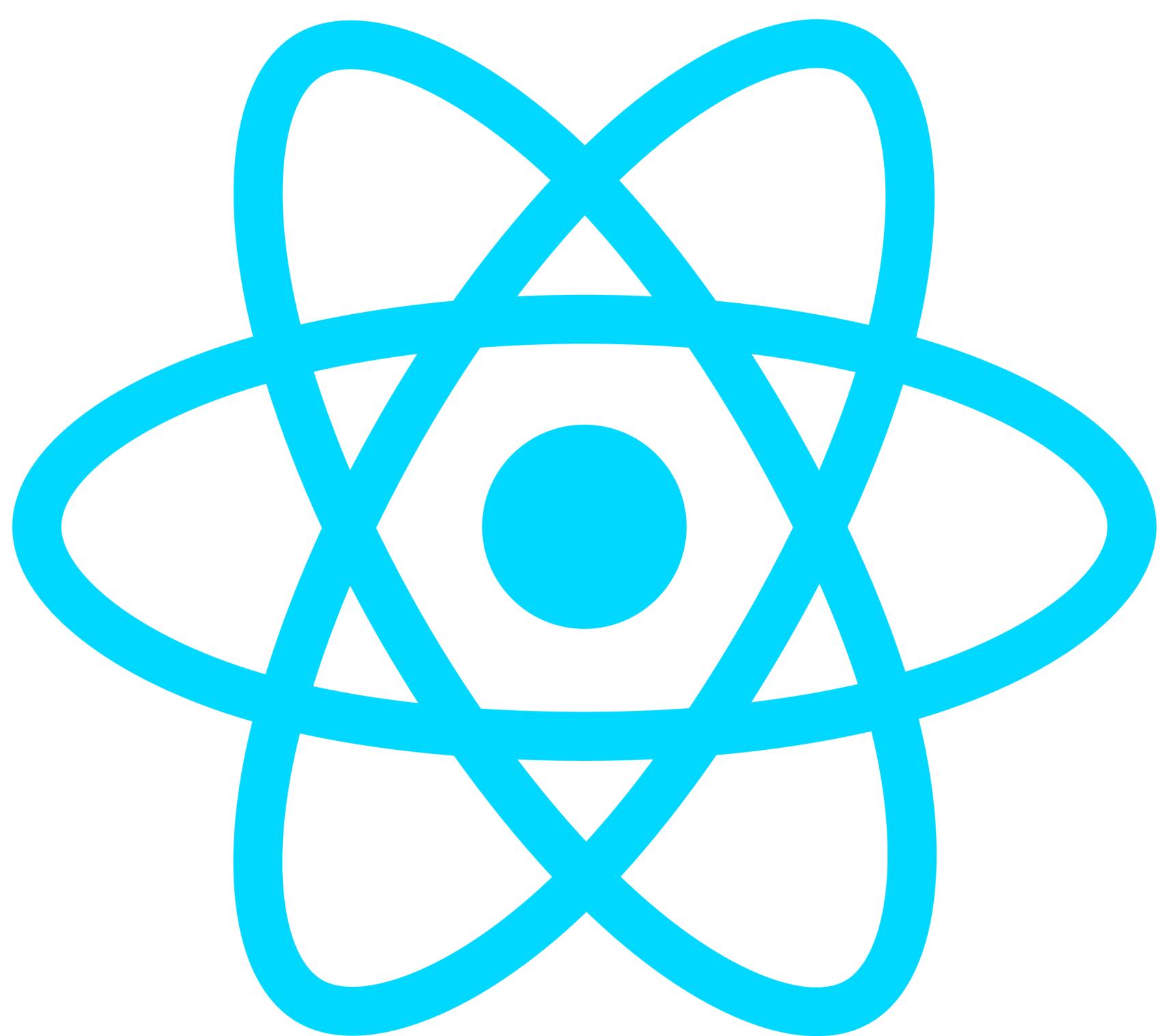
{ DEV }

Jane Pimentel



# O que é React Native?

React Native é um framework criado pelo Facebook para desenvolver aplicativos mobile nativos usando JavaScript e React. Com ele, você escreve uma base de código que funciona tanto para Android quanto iOS



# 01

## ✳️ Por que usar?

---

- Código compartilhado entre plataformas (Android e iOS)
- Desenvolvimento mais rápido
- Grande comunidade e ecossistema
- Integra-se com funcionalidades nativas do aparelho (câmera, GPS, Bluetooth, etc.)

# ⭐ Exemplo Real

🧪 Exemplo real: App de boas-vindas

Abaixo temos um exemplo básico de como criar um aplicativo simples com React Native que mostra uma saudação:

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

export default function App() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Olá, Jane! Bem-vinda ao seu app em React Native 🎉</Text>
    </View>
  );
}
```

O que está acontecendo aqui?

- View: funciona como uma div, serve de contêiner para outros elementos.
- Text: exibe textos na tela.
- style: usamos JavaScript puro para aplicar estilos.

Esse é o primeiro passo. Com pouco código, já é possível ter um app funcional!



## ✨ Componentes: a base de tudo

---

No React Native, componentes são como blocos de montar. Você os combina para construir a interface do seu app. Cada componente pode receber propriedades, conhecidas como props, que permitem deixá-los mais dinâmicos e reutilizáveis.

# ⭐ Componentização

Componentes são como peças LEGO da sua interface. Desde um botão até uma tela completa.

```
● ● ●  
import { View, Text } from 'react-native';  
  
function Saudacao(props) {  
  return (  
    <View>  
      <Text>Olá, {props.nome}!</Text>  
    </View>  
  );  
}  
  
export default function App() {  
  return (  
    <>  
      <Saudacao nome="Jane" />  
      <Saudacao nome="Carlos" />  
    </>  
  );  
}
```

💡 Usando o componente com diferentes props

```
● ● ●  
export default function App() {  
  return (  
    <>  
      <Saudacao nome="Jane" />  
      <Saudacao nome="Carlos" />  
    </>  
  );  
}
```

# O3

## ★ Estilizando com o StyleSheet

---

Você não usa CSS diretamente, mas uma estrutura parecida com objetos JavaScript.



# Estilização

✍️ No React Native, o estilo dos elementos é feito em JavaScript, usando o StyleSheet. Esqueça className ou arquivos .css: aqui, você cria objetos de estilo direto no seu código

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

const estilos = StyleSheet.create({
  texto: {
    color: 'purple',
    fontSize: 20
  }
});

function Saudacao() {
  return (
    <View>
      <Text style={estilos.texto}>Olá, mundo!</Text>
    </View>
  );
}
```

- Ajuda na organização do código
- Permite melhor performance, pois os estilos são pré-processados
- Traz autocompletar e sugestões nos editores de código

# 04

## ⭐ Hooks: Estados e Efeitos

---

Os hooks são funções do React que permitem adicionar lógica e comportamento aos componentes. O mais usado é o useState, que serve para controlar valores que mudam com o tempo — como cliques, textos digitados, etc.

# ✨ Hooks: Controle e Interatividade

useState: para controlar valores dinâmicos

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

export default function Contador() {
  const [contador, setContador] = useState(0);

  return (
    <View>
      <Text>Valor: {contador}</Text>
      <Button title="Somar +1" onPress={() => setContador(contador + 1)} />
    </View>
  );
}
```

💡 Entendendo o que acontece:

- useState(0): define o valor inicial como 0
- contador: guarda o valor atual
- setContador: altera esse valor
- Sempre que setContador for chamado, o app re-renderiza e mostra o novo valor



## ✨ Navegação entre Telas com React Navigation

---

Se seu app tem mais de uma tela (e quase todos têm), você vai precisar navegar entre elas. O pacote mais usado pra isso é o react-navigation.

# ✨ Instalação React Navigation

## ⚙️ Instalação (passo único)

No terminal:



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

E se estiver usando navegação tipo "pilha"  
(Stack), instale também:



```
npm install @react-navigation/native-stack
```



# Definindo as telas e seus componentes

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

function TelaInicial({ navigation }) {
  return (
    <View>
      <Text>Bem-vinda, Jane!</Text>
      <Button title="Ir para detalhes" onPress={() => navigation.navigate('Detalhes')} />
    </View>
  );
}

function TelaDetalhes() {
  return (
    <View>
      <Text>Você chegou na tela de detalhes!</Text>
    </View>
  );
}
```



## Explicando:

- Telainicial: é a primeira tela que o usuário vê. Ao clicar no botão, ela navega para a tela "Detalhes".
- navigation.navigate('Detalhes'): é a função que muda para outra tela registrada com o nome "Detalhes".
- TelaDetalhes: é o conteúdo que aparece quando o usuário chega na segunda tela.

# ✨ Criando a navegação com Stack Navigator

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

const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Início" component={TelaInicial} />
        <Stack.Screen name="Detalhes" component={TelaDetalhes} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

## 💡 Explicando:

- **NavigationContainer**: envolve todo o sistema de navegação. Ele é o provedor de contexto.
- **createNativeStackNavigator()**: cria um navegador no estilo “pilha” (como páginas empilhadas).
- **Stack.Navigator**: é onde você registra suas telas.
- **Stack.Screen**: define cada tela, com um **name** (rótulo) e o componente que será exibido.

# 06

## ✳ Trabalhando com Listas no React Native

---

O React Native oferece o componente FlatList para renderizar listas grandes de forma performática e simples.

# ⭐ Trabalhando com Listas: FlatList

```
● ● ●  
  
import { FlatList, Text, View } from 'react-native';  
  
const dados = [  
  { id: '1', nome: 'Jane' },  
  { id: '2', nome: 'Carlos' },  
  { id: '3', nome: 'Luna' },  
];  
  
export default function ListaDeNomes() {  
  return (  
    <FlatList  
      data={dados}  
      keyExtractor={item => item.id}  
      renderItem={({ item }) => (  
        <View>  
          <Text>{item.nome}</Text>  
        </View>  
      )}  
    />  
  );  
}
```

📌 Entenda o que faz cada parte:

- data: o array de itens a serem renderizados
- keyExtractor: função que retorna uma chave única pra cada item (melhora desempenho)
- renderItem: função que define como cada item será exibido

# Agradecimentos

---



# Sobre a Criação deste eBook

E-Book foi desenvolvido com o apoio de Inteligência Artificial, e cuidadosamente diagramado por uma pessoa real, com foco na clareza, estética e aprendizado acessível. Tanto o texto quanto os exemplos de código foram organizados com fins educativos, e podem conter imprecisões ou limitações oriundas do uso de IA.

Sinta-se à vontade para usar, adaptar e melhorar este material conforme suas necessidades.

