



Universidad Politécnica de Querétaro
Tecnologías de la Información e Innovación Digital
Alumno: Diego Irineo Atanacio
Fecha de entrega: 09/10/25 Grupo: TIID211



Materia: Desarrollo Apps Móviles

Profesor: Ivan Isay Guerra Lopez

Alumno: Diego Irineo

Fecha de entrega: 09/10/25



StyleSheets

- Permite definir estilos como objetos JavaScript
- Muchas propiedades son parecidas a CSS, pero en camelCase

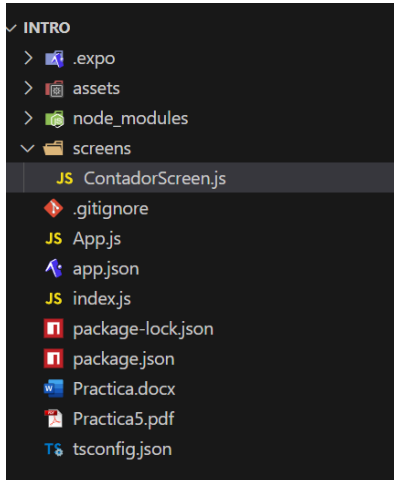
backgroundColor

background-color

- Usa Flexbox para distribuir elementos
- Puedes tener un archivo JS para estilos compartidos entre componentes.



Primero adentro de nuestro proyecto INTRO agregamos una carpeta llamada screens y agregamos un archivo llamado ContadorScreen.js



Después en ese mismo copiamos todo el código que ya teníamos en el archivo App.js pero cambiando la variable al nombre de el nuevo archivo

```
JS ContadorScreen.js X
screens > JS ContadorScreen.js > ContadorScreen
1 //1 Imports: Zona de Imports
2 import { StatusBar } from 'expo-status-bar';
3 import { StyleSheet, Text, View, Button } from 'react-native';
4 import React, { useState } from 'react';
5
6 //2 Main: Zona del Componente
7 export default function ContadorScreen() {
8   const [contador, setContador] = useState(0);
9   return (
10     <View style={styles.container}>
11       <Text>Contador: {contador}</Text>
12       <Button title='Agregar' onPress={() => setContador(contador + 1)}>
13       <Button title='Quitar' onPress={() => setContador(contador - 1)}>
14       <Button title='Reiniciar' onPress={() => setContador(0)}>
15       <StatusBar style="auto" />
16     </View>
17   );
18 }
19
20 //3 Styles: Zona de Estilos para el Componente
21 const styles = StyleSheet.create({
22   container: {
23     flex: 1,
24     backgroundColor: '#fff',
25     alignItems: 'center',
26     justify-content: 'center',
27   },
28 });
```



Al igual en el archivo App borramos los estilos los import de statusbar y React

```
JS ContadorScreen.js JS App.js 1 X
JS App.js > ...
1 //1 Imports: Zona de Imports
2 import { StyleSheet, Text, View, Button } from 'react-native';
3
4 //2 Main: Zona del Componente
5 export default function App() {
6   return (
7
8   );
9 }
10
11
```

Agregamos a esta misma un import para llamar todo el código de el archivo nuevo

```
JS ContadorScreen.js JS App.js 1 X
JS App.js > ...
1 //1 Imports: Zona de Imports
2 import { StyleSheet, Text, View, Button } from 'react-native';
3 import ContadorScreen from './screens/ContadorScreen';
4
5 //2 Main: Zona del Componente
6 export default function App() {
7   return (
8
9   );
10 }
11
12
```

Despues agregamos el contador y corroboramos que funcione correctamente

JS ContadorScreen.js

JS App.js

X

JS App.js > App

```
1 //1 Imports: Zona de Imports
2 import { StyleSheet, Text, View, Button } from 'react-native';
3 import ContadorScreen from './screens/ContadorScreen';
4
5 //2 Main: Zona del Componente
6 export default function App() {
7   return (
8     <ContadorScreen></ContadorScreen>
9   );
10 };
11
12
13
```

INTRO

X

+

←

→

↺

localhost:8081

🔍

☆

📄

🌐

⋮

Contador: 3

AGREGAR

QUITAR

REINICIAR



Para agregar una nueva clase de estilo lo debemos hacer después de la coma del original, después lo agregamos adentro del texto contador pero cambiando el style con la nueva clase, todo esto en el archivo ContadorScreen

```
JS ContadorScreen.js X JS App.js
screens > JS ContadorScreen.js > ContadorScreen
7 export default function ContadorScreen() {
11   <View style={styles.container}>
12     <Text style={styles.texto}>Contador: {contador}</Text>
13     <Button title='Agregar' onPress={() => setContador(contador + 1)}>
14     <Button title='Quitar' onPress={() => setContador(contador - 1)}>
15     <Button title='Reiniciar' onPress={() => setContador(0)}>
16   </View>
17   <StatusBar style="auto" />
18 }
19
20 //3 Styles: Zona de Estilos para el Componente
21 const styles = StyleSheet.create({
22   container: {
23     flex: 1,
24     backgroundColor: '#ea34e4da',
25     alignItems: 'center',
26     justifyContent: 'center',
27   },
28   texto: {
29     color: '#2c00deff',
30     fontSize: 30,
31   }
32 });
```

Agregamos una mas para el contador numerico

```
JS ContadorScreen.js X JS App.js
screens > JS ContadorScreen.js > ContadorScreen
26 const styles = StyleSheet.create({
27   container: {
28     alignItems: 'center', //Alineación Horizontal
29     justifyContent: 'center', //Alineación Vertical
30   },
31   texto: {
32     color: '#2c00deff',
33     fontSize: 30, //Tamaño
34     fontFamily: 'Times New Roman', //Tipo de Letra
35     fontWeight: 'bold', //Grosor de la letra
36     textDecorationLine: 'line-through' //Decoración de la letra (underline, line-through, none)
37   },
38   texto2: {
39     color: '#fc0303e6',
40     fontSize: 40, //Tamaño
41     fontFamily: 'courier', //Tipo de Letra
42     fontWeight: '300', //Grosor de la letra
43     fontStyle: 'italic', //Estilo de la letra (normal, italic)
44     textDecorationLine: 'underline' //Decoración de la letra (underline, line-through, none)
45   },
46 });
```



Después agregamos una clase más y agregamos un view para los estilos de los botones

```
<View style={styles.botonContainer}>
  <Button title='Agregar' onPress={() => setContador(contador + 1)}>
    <Button title='Quitar' onPress={() => setContador(contador - 1)}>
      <Button title='Reiniciar' onPress={() => setContador(0)}>
    </Button>
  </Button>
</View>
```

```
JS ContadorScreen.js X JS App.js
screens > JS ContadorScreen.js > [e] styles > [e] botonesContainer
29 const styles = StyleSheet.create({
44   texto2: {
45     //Tamaño de la letra
47     fontFamily: 'courier', //Tipo de Letra
48     fontWeight: '300', //Grosor de la letra
49     fontStyle: 'italic', //Estilo de la letra (normal, italic)
50     textDecoracionLine: 'underline' //Decoración de la letra (underline, line-through, none)
51   },
52   botonesContainer: {
53     marginTop: 20,
54     flexDirection: 'row', //Fila o Columna (row, column)
55     gap: 20 //Espacio entre los elementos (solo en React Native 0.71 o superior)
56   },
57
58
59
60 });
61
```

Para cambiar el color de los botones se pone una variable “color” adentro de el boton y seleccionas el color que desees

```
JS ContadorScreen.js X JS App.js
screens > JS ContadorScreen.js > [e] ContadorScreen
7 export default function ContadorScreen() {
15
16   <View style={styles.botonContainer}>
17     <Button color='red' title='Agregar' onPress={() => setContador(contador + 1)}>
18     <Button color='gold' title='Quitar' onPress={() => setContador(contador - 1)}>
19     <Button color='orange' title='Reiniciar' onPress={() => setContador(0)}>
20   </Button>
21 </View>
22
23   <StatusBar style="auto" />
24
25 </View>
26
27 }
```



Universidad Politécnica de Querétaro
Tecnologías de la Información e Innovación Digital
Alumno: Diego Irineo Atanacio
Fecha de entrega: 09/10/25 Grupo: TIID211

Conclusion

Durante la actividad logramos implementar y personalizar botones dentro de un archivo JavaScript de React Native, aplicando diferentes estilos, colores y alineaciones. Este proceso permitió comprender mejor cómo se estructuran los componentes en React Native y cómo el uso de estilos (mediante StyleSheet) influye en la apariencia y disposición de los elementos en pantalla. Además, aprendimos a combinar propiedades como `backgroundColor`, `alignItems`, `justifyContent` y `margin` para obtener un diseño más atractivo y funcional. En general, esta práctica reforzó los conceptos de diseño de interfaces y la importancia de la coherencia visual dentro de una aplicación móvil.