



Universidad Politecnica de Queretaro  
Tecnologias de la Informacion e Innovacion Digital  
Fecha de entrega:16/10/25      Grupo:TIID211



Materia:Desarrollo Apps Moviles

Profesor: Ivan Isay Guerra Lopez

Alumno:

Irineo Atanacio Diego

Fecha de entrega:16/10/25



En esta práctica usamos el código llamado BotonesScreen que permite al usuario encender o apagar un estado representado por la variable Prendido

utiliza los siguientes elementos de React Native:

useState: Para manejar el estado de la variable Prendido

TouchableOpacity: Para crear botones de encendido y apagado

Switch: Para alternar el estado con un interruptor

StyleSheet: Para aplicar estilos personalizados a los componentes

Aquí se importan los módulos que necesitamos:

React: para poder usar JSX y los hooks

useState: hook que permite guardar y actualizar los valores “prendido” o “apagado”

View: contenedor principal (como un <div> en HTML)

Text: para mostrar texto en pantalla

StyleSheet: para crear estilos

TouchableOpacity: crea botones táctiles con efecto visual al presionar

Switch: componente tipo interruptor (encendido/apagado)

```
JS BotonesScreen.js X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > BotonesScreen
1   import React, {use, useState} from 'react';
2   import {View, Text, StyleSheet, TouchableOpacity, Switch} from 'react-native';
3
```



Se define un componente funcional llamado BotonesScreen

Se crea una variable de estado llamada Prendido

```
JS BotonesScreen.js X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > BotonesScreen
4 export default function BotonesScreen(){
5   const [Prendido, setPrendido] = useState(false);
6
```

Estas dos líneas se usan para decidir el color del fondo y del texto según el estado

```
JS BotonesScreen.js X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > BotonesScreen
4 export default function BotonesScreen(){
7   const backgroundColor = Prendido ? 'white' : 'black';
8   const textColor = Prendido ? 'black' : 'white';
9
```

Muestra el texto “Estado: Prendido” o “Estado: Apagado” según el valor del estado  
También cambia el color del texto dinámicamente

```
JS BotonesScreen.js X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > BotonesScreen
4 export default function BotonesScreen(){
11   <View style={[styles.container, {backgroundColor}]}>
12     <Text style={[styles.texto, {color: textColor}]}>Estado: {Prendido ? 'Prendido' : 'Apagado'}</Text>
13
```



Se crea un botón táctil verde que al presionarlo cambia el estado a encendido

```
JS BotonesScreen.js X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > BotonesScreen
4 export default function BotonesScreen(){
13
14   <TouchableOpacity
15     style={styles.botonEncender}
16     onPress={() => setPrendido(true)}>
17     <Text style={styles.textoBoton}>Encender</Text>
18   </TouchableOpacity>
```

Igual al botón anterior, pero con color rojo y que apaga el estado

```
JS BotonesScreen.js M X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > BotonesScreen
4 export default function BotonesScreen(){
19
20   <TouchableOpacity
21     style={styles.botonApagar}
22     onPress={() => setPrendido(false)}>
23     <Text style={styles.textoBoton}>Apagar</Text>
24   </TouchableOpacity>
25
```

Aquí se crea un interruptor que también controla el mismo estado

```
JS BotonesScreen.js M X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > styles > container
4 export default function BotonesScreen(){
25
26   <View style={styles.switchContainer}>
27     <Text style={[styles.switchLabel, {color: textColor}]}>Control Switch:</Text>
28     <Switch value={Prendido} onChange={setPrendido}> </Switch>
29   </View>
30 </View>
```



Y por último esta sería la parte de los estilos

```
JS BotonesScreen.js M X JS MenuScreen.js
INTRO > screens > JS BotonesScreen.js > [🔗] styles > [🔗] container
33  const styles = StyleSheet.create({
34    container: {
35      flex: 1,
36      justifyContent: 'center',
37      alignItems: 'center',
38      backgroundColor: 'hsl(209, 73%, 88%, 1.00)',
39    },
40    texto : {
41      fontSize: 20,
42      fontWeight: 'bold',
43      marginBottom: 30,
44    },
45    botonEncender: {
46      backgroundColor: 'green',
47      paddingVertical: 10,
48      paddingHorizontal: 25,
49      borderRadius: 8,
50      marginBottom: 20,
51    },
52    botonApagar: {
53      backgroundColor: 'red',
54      paddingVertical: 10,
55      paddingHorizontal: 25,
56      borderRadius: 8,
57      marginBottom: 20,
58    },
59    textoBoton: {
60      color: 'white',
61      fontSize: 16,
62      fontWeight: 'bold',
63    },
64    switchContainer: {
65      flexDirection: 'row',
66      alignItems: 'center',
67    },
68    switchLabel: {
69      fontSize: 16,
70      marginRight: 10,
71    },
72  })
```



Universidad Politecnica de Queretaro  
Tecnologias de la Informacion e Innovacion Digital  
Fecha de entrega:16/10/25      Grupo:TIID211

## **Conclusión:**

En esta práctica comprendimos mejor el uso del useState para controlar el estado de los componentes y cómo los botones y switches pueden modificar dinámicamente la interfaz de usuario en React Native.