🔄 ¿Qué es un estado (state) en React Native?

El **estado** representa datos internos de un componente que pueden cambiar a lo largo del tiempo.
Cuando el estado cambia, el componente se vuelve a renderizar para reflejar ese cambio en la interfaz.

📌 Declaración de un estado con `useState`

Para manejar estados en React Native, usamos el hook `useState`.
Este hook permite inicializar un valor y actualizarlo cuando sea necesario.

jsx
```jsx
import { useState } from 'react';

const [contador, setContador] = useState(0);
```

🔍 Explicación:

- `contador`: es el valor inicial del estado.
- `setContador`: es una función que se usa para actualizar el valor.
- `0`: es el valor inicial del estado.

Fuente: [React Docs - useState Hook]

🖱️ Actualización del estado con eventos

Para actualizar el valor del estado, usamos la función que devuelve `useState`.
Podemos asociar esta actualización a un evento, por ejemplo, un botón.

jsx
```jsx
<Button title="Incrementar" onPress={() => setContador(contador + 1)} />
```

🔍 Explicación:

- Cada vez que se presiona el botón, el valor de `contador` se incrementa en 1.
- React Native vuelve a renderizar el componente y muestra el nuevo valor.

Fuente: [React Native Docs - Handling Touches]

✨ Componentes interactivos en React Native

🔘 Button

Componente básico para disparar eventos al ser presionado.

jsx
```
<Button title="Presioname" onPress={() => alert('Botón presionado')}
/>
```

- `title`: texto que muestra el botón.
- `onPress`: función que se ejecuta al hacer click.

Fuente: [React Native Docs - Button]

✋ TouchableOpacity

Componente más flexible que un botón, permite estilos personalizados y tiene un efecto de opacidad al ser presionado.

jsx
```
<TouchableOpacity onPress={() => alert('Presionado')}>
  <Text style={{ color: 'blue' }}>Presioname</Text>
</TouchableOpacity>
```

🔍 Diferencias con `Button`:

- Más personalizable (colores, bordes, imágenes, etc.).
- Permite múltiples hijos (no solo texto).

- Es ideal para crear botones más estilizados.

Fuente: [React Native Docs - TouchableOpacity]

🗃️ Lista de Tareas (Todo List) en React Native

Un ejemplo común para practicar estados es una **Lista de Tareas**.
Podemos usar un array para manejar los elementos de la lista y el estado para actualizarla en tiempo real.

jsx
```
const [tareas, setTareas] = useState([]);

const agregarTarea = (nuevaTarea) => {
  setTareas([...tareas, nuevaTarea]);
};
```

🔍 **Explicación:**

- El estado inicial es un array vacío `[]`.
- Cuando se agrega una tarea, se actualiza el array usando el spread operator `...`.

Fuente: [React Native Docs - Lists and ScrollView]

🗑️ Eliminar elementos de una lista

Para eliminar un elemento de la lista, podemos filtrar los elementos que no coincidan con el que queremos borrar.

jsx
```
const eliminarTarea = (id) => {
  setTareas(tareas.filter((tarea) => tarea.id !== id));
};
```

🔍 **Explicación:**

- `filter` devuelve un nuevo array con los elementos que cumplen la condición.

- En este caso, se eliminan los que coincidan con el `id` pasado como parámetro.

✨ Extra: Cambiar el estilo de un elemento al ser presionado

Podemos aprovechar los eventos de `TouchableOpacity` para cambiar el estilo de un elemento.

jsx

```
<TouchableOpacity
  onPress={() => setEstilo(!estilo)}
  style={{ backgroundColor: estilo ? 'green' : 'red' }}>
  <Text>Cambiar color</Text>
</TouchableOpacity>
```

🔍 **Explicación:**

- Usando un estado (`estilo`) cambiamos el color de fondo.
- Cada vez que se presiona, alterna entre verde y rojo.