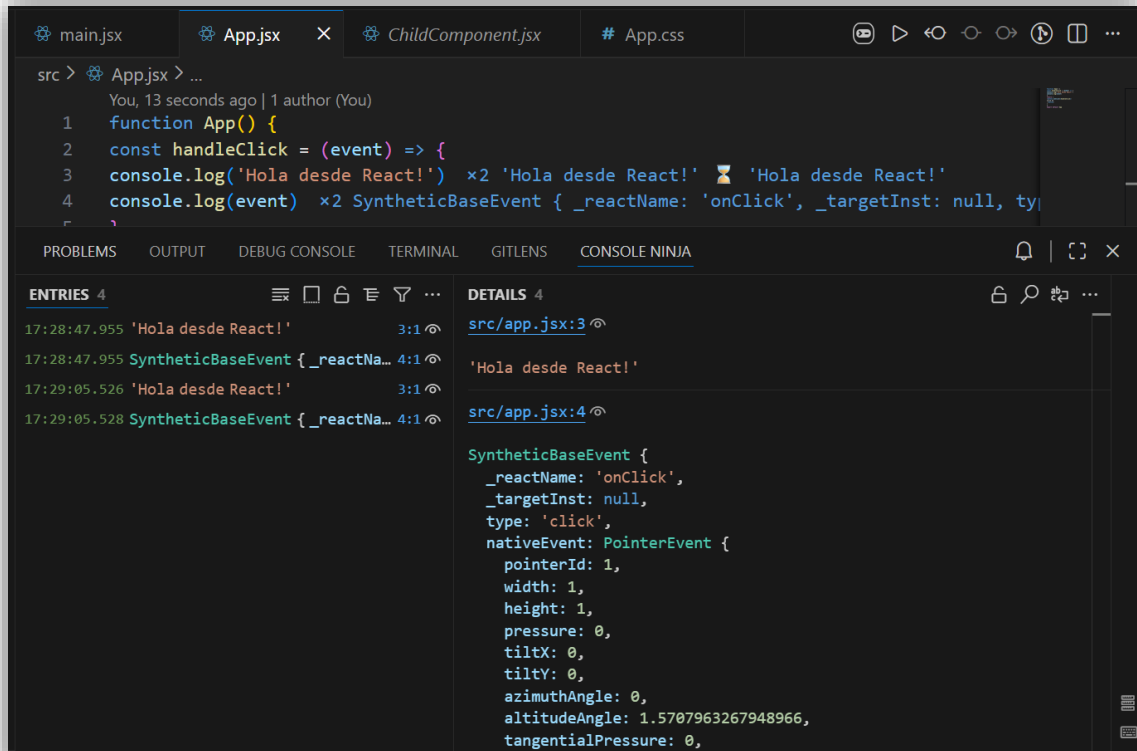

TAREAS REACT - NIVEL 6

Tabla de contenido

Parte A: onClick y el objeto evento.....	2
Parte B: Cambiar la UI con estado + evento	2
Parte C: onChange (entrada controlada)	3
.....	3
.....	3
Parte D: onSubmit (formulario)	3
Miniretillo:	4
Preguntas cortas:	4

Parte A: onClick y el objeto evento

En la siguiente imagen se puede ver como con cada pulsación se genera el mensaje con el código de la ficha:



The screenshot shows a VS Code editor with a file explorer on the left containing 'main.jsx', 'App.jsx', 'ChildComponent.jsx', and 'App.css'. The 'App.jsx' file is open, showing a function `App()` with a `handleClick` event handler. The handler logs a message and a `SyntheticBaseEvent` object. The bottom panel shows the 'CONSOLE NINJA' with four log entries. The details of the last log entry show the structure of the `SyntheticBaseEvent` object, including `_reactName: 'onClick'`, `_targetInst: null`, `type: 'click'`, and a `nativeEvent: PointerEvent` with various coordinates and pressure values.

```
src > App.jsx > ...
You, 13 seconds ago | 1 author (You)
1 function App() {
2   const handleClick = (event) => {
3     console.log('Hola desde React!') x2 'Hola desde React!' ⌚ 'Hola desde React!'
4     console.log(event) x2 SyntheticBaseEvent { _reactName: 'onClick', _targetInst: null, ty
...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS CONSOLE NINJA

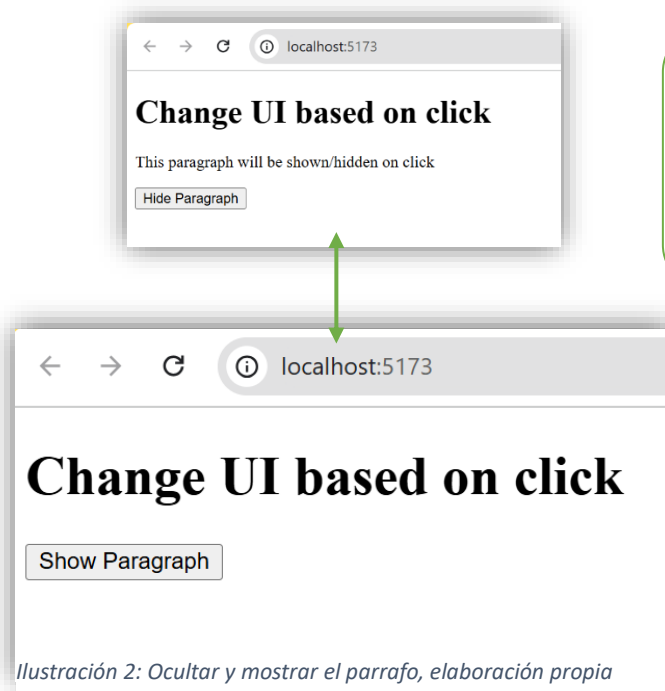
ENTRIES 4
17:28:47.955 'Hola desde React!' 3:1
17:28:47.955 SyntheticBaseEvent { _reactNa... 4:1
17:29:05.526 'Hola desde React!' 3:1
17:29:05.528 SyntheticBaseEvent { _reactNa... 4:1

DETAILS 4
src/app.jsx:3
'Hola desde React!'

src/app.jsx:4
SyntheticBaseEvent {
  _reactName: 'onClick',
  _targetInst: null,
  type: 'click',
  nativeEvent: PointerEvent {
    pointerId: 1,
    width: 1,
    height: 1,
    pressure: 0,
    tiltX: 0,
    tiltY: 0,
    azimuthAngle: 0,
    altitudeAngle: 1.5707963267948966,
    tangentialPressure: 0,
```

Ilustración 1: Resultado en la consola, elaboración propia

Parte B: Cambiar la UI con estado + evento



Se gestiona la visibilidad del párrafo mediante “isParagraphVisible”, que se invierte al pulsar el botón y utiliza el operador `&&` para mostrar el texto.

Ilustración 2: Ocultar y mostrar el párrafo, elaboración propia

Parte C: onChange (entrada controlada)



← → ↻ ⓘ localhost:5173

Alias del jugador

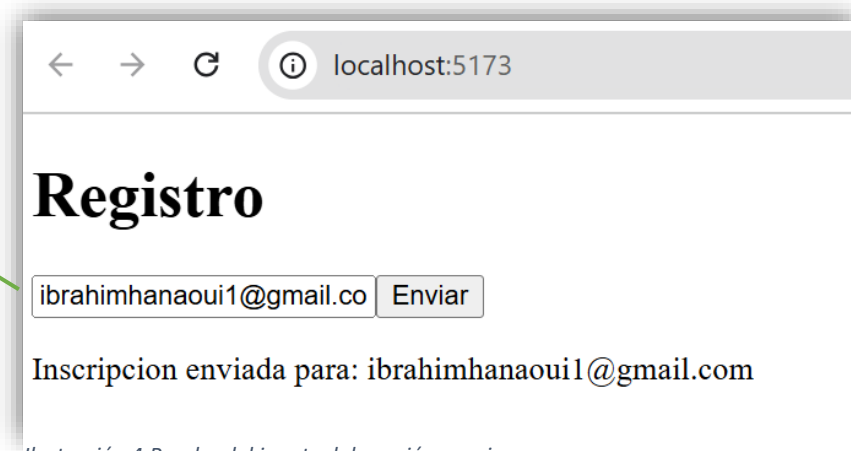
Tu alias es: **Ibra**

En este caso usamos el evento “onChange” para cuando haya un cambio se actualice el output.

Ilustración 3: Prueba del input, elaboración propia

Parte D: onSubmit (formulario)

A diferencia del apartado anterior, en este caso usamos un botón para mandar y mostrar el input.



← → ↻ ⓘ localhost:5173

Registro

Inscripcion enviada para: ibrahimhanaoui1@gmail.com

Ilustración 4: Prueba del input, elaboración propia

Miniretillo:

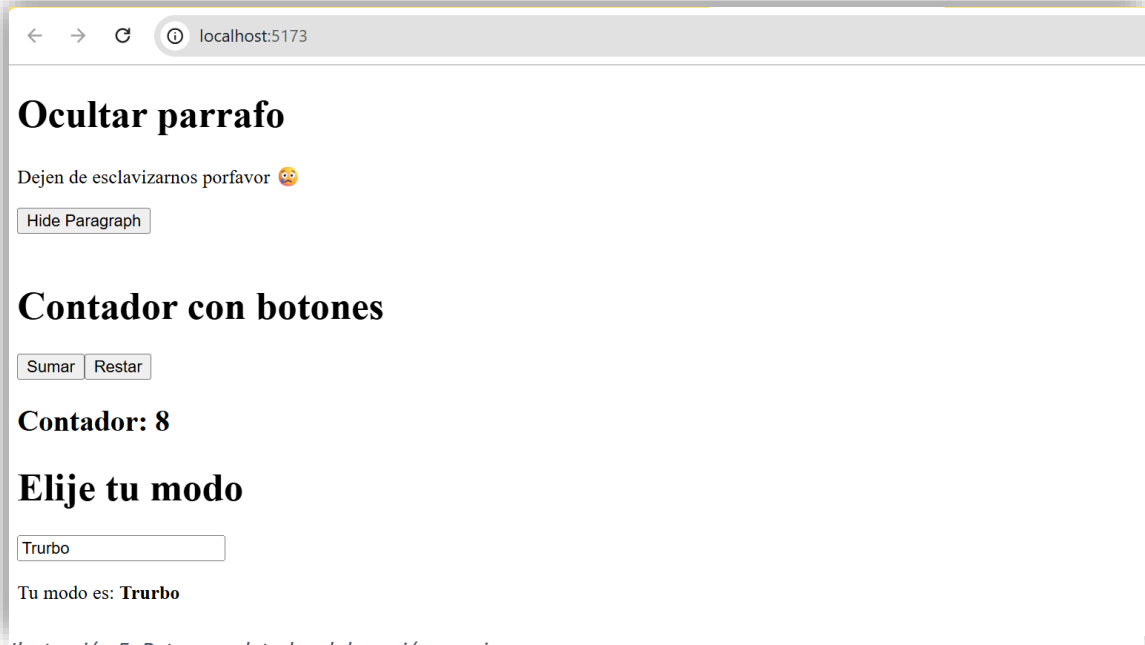


Ilustración 5: Reto completado, elaboración propia

Para este reto, he reciclado código visto en otros ejercicios y adaptándolo a lo que me exigía el reto, e disfrutado y entendido de manera más detallada que hace cada elemento para poder adaptarlo, ya que realmente no he tenido que crear código nuevo.

Preguntas cortas:

1. **¿Qué diferencia hay entre onClick y onSubmit?** El **onClick** se ejecuta cuando haces click pero en cambio el **onSubmit** se va ejecutando conforme hayan cambios.
2. **¿Por qué usamos e.preventDefault() en un formulario?** Para evitar que la página se recargue al enviar el formulario y permitir que React gestione los datos.
3. **¿Qué es una "entrada controlada"?** Es un input cuyo valor depende del estado **value** y se actualiza mediante un evento **onChange**, dando a React el control total de lo que se escribe.
4. **¿Qué estados y eventos manejas?**

En el mini-reto manejo tres estados: `isParagraphVisible` , `newContador` y `modo`. Estos se actualizan respectivamente mediante eventos **onClick** en los botones de visibilidad y del contador, y a través del evento **onChange** en el campo de entrada de texto.