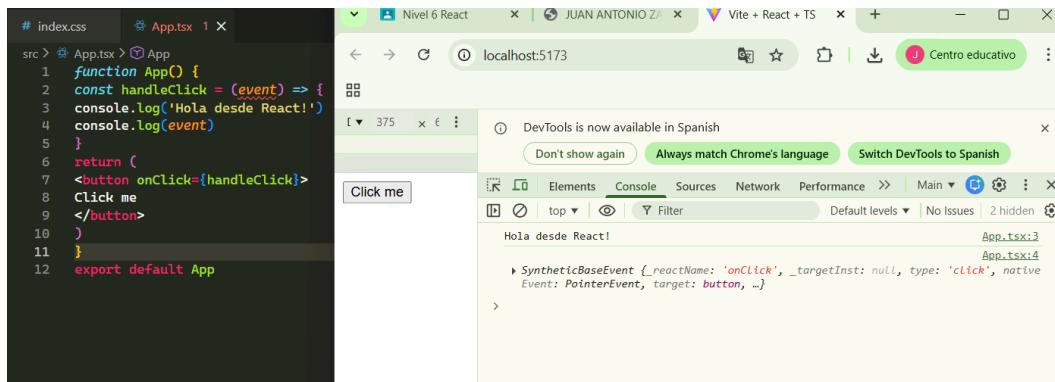


NIVEL 6 REACT



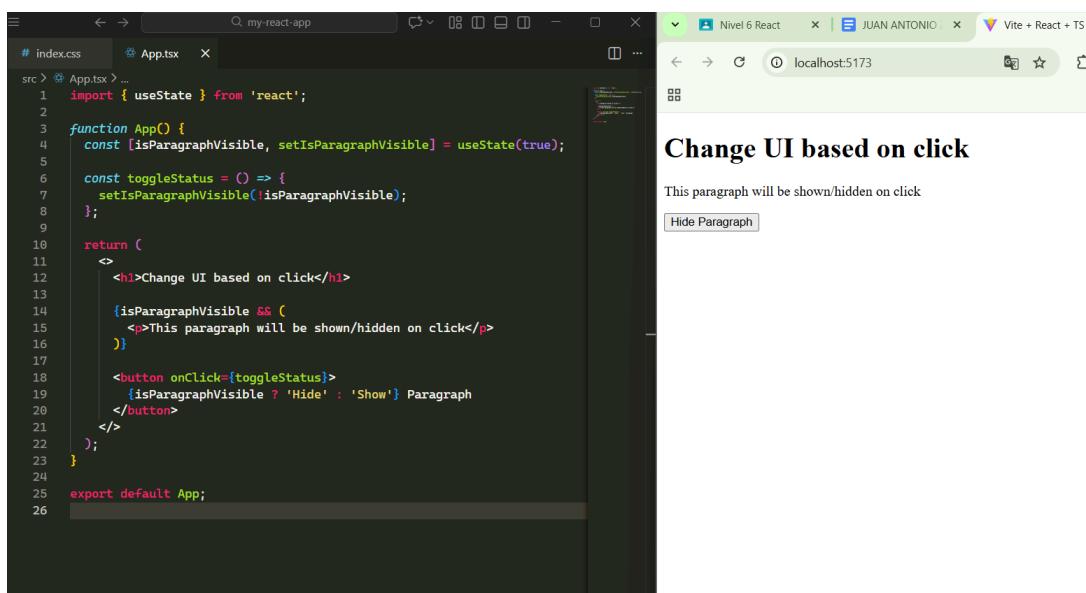
PARTE A: onClick y el objeto event:	2
PARTE B: Cambiar la UI con estado + evento:	2
PARTE C: onChange (entrada controlada):	3
PARTE D: onSubmit (formulario):	3
MINIRETILLO:	4
PREGUNTAS	5

PARTA A: onClick y el objeto event:



(Ilustración 1. Captura de pantalla del código y de la página web mostrando por la consola de la App como aparece el objeto “SyntheticBaseEvent” con nuestro mensaje :”Hola desde React!”. Elaboración propia.)

PARTA B: Cambiar la UI con estado + evento:



(Ilustración 2. Captura de pantalla del código y de la App, que implementa un botón que muestra y oculta un párrafo, haciendo uso de un booleano y el renderizado condicional. Elaboración propia)

PARTE C: onChange (entrada controlada):

The screenshot shows a code editor and a browser window. The code editor displays a file named App.tsx with the following content:

```
# index.css    App.tsx
src > App.tsx ...
1 import { useState } from 'react';
2
3 function App() {
4   const [nickname, setNickname] = useState('');
5
6   return (
7     <>
8       <h1>Alias del jugador</h1>
9
10      <input
11        type="text"
12        placeholder="Escribe tu alias"
13        value={nickname}
14        onChange={(e) => setNickname(e.target.value)}
15      />
16
17      <p>
18        Tu alias es: <strong>{nickname || '...'}</strong>
19      </p>
20    </>
21  );
22}
23
24 export default App;
```

The browser window shows the output of the application. It has a title "Alias del jugador". Below it is an input field containing "juanac056". To the right of the input field, the text "Tu alias es: juanac056" is displayed.

(Ilustración 3. Captura de pantalla del código y de la App mostrando como el input guarda al momento el alias del jugador y lo va mostrando a la vez gracias al evento “onChange”.
Elaboración propia.)

PARTE D: onSubmit (formulario):

The screenshot shows a code editor and a browser window. The code editor displays a file named App.tsx with the following content:

```
index.css    App.tsx 1 x
src > App.tsx ...
1 import { useState } from 'react';
2
3 function App() {
4   const [email, setEmail] = useState('');
5   const [msg, setMsg] = useState('');
6
7   const handleSubmit = (e) => {
8     e.preventDefault();
9     setMsg(`Inscripción enviada para: ${email}`);
10  };
11
12  return (
13    <>
14      <h1>Registro</h1>
15
16      <form onSubmit={handleSubmit}>
17        <input
18          type="email"
19          placeholder="tu@email.com"
20          value={email}
21          onChange={(e) => setEmail(e.target.value)}
22          required
23        />
24
25        <button type="submit">Enviar</button>
26      </form>
27
28      {msg && <p>{msg}</p>}
29    </>
30  );
31}
32
33 export default App;
```

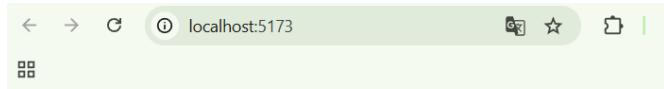
The browser window shows the output of the application. It has a title "Registro". Below it is an input field with the placeholder "tu@email.com" and the value "juanelguapo9041@gmail.com". To the right of the input field, the text "Inscripción enviada para: juanelguapo9041@gmail.com" is displayed.

(Ilustración 4. Captura de pantalla del código y de la App , donde con un input (botón), nos muestra un mensaje si nuestra inscripción ha sido enviada (previa validación con el “type='email””. Elaboración propia).

MINIRETILLO:

```
# index.css  ⌂ App.tsx  ✘
src > ⌂ App.tsx > ⌂ Contador > [x] decrementar
1 import { useState } from 'react';
2
3 function Contador() {
4   const [count, setCount] = useState(0);
5
6   const incrementar = () => {
7     setCount(count + 1);
8   };
9
10  const decrementar = () => {
11    if (count > 0) {
12      setCount(count - 1);
13    }
14  };
15
16  return (
17    <div>
18      <h2>Contador</h2>
19      <p>Valor: {count}</p>
20
21      <button onClick={incrementar}>+1</button>
22      <button onClick={decrementar}>-1</button>
23    </div>
24  );
25}
26
27 function Panel() {
28   const [visible, setVisible] = useState(false);
29   const [modo, setModo] = useState('');
30
31   const togglePanel = () => {
32     setVisible(!visible);
33   };
34
35   return (
36     <div>
37       <h2>Panel</h2>
38
39       <button onClick={togglePanel}>
40         {visible ? 'Ocultar panel' : 'Mostrar panel'}
41       </button>
42
43       {visible && (
44         <p>Este es un texto secreto del panel <strong>modo</strong></p>
45       )}
46
47       <input
48         type="text"
49         placeholder="Escribe el modo"
50         value={modo}
51         onChange={(e) => setModo(e.target.value)}
52       />
53
54       <p>
55         Modo actual: <strong>{modo || 'ninguno'}</strong>
56       </p>
57     </div>
58   );
59 }
60
61 function App() {
62   return (
63     <>
64       <h1>Mini Panel de Control <strong>m</strong></h1>
65
66       <Panel />
67       <Contador />
68     </>
69   );
70 }
71
72 export default App;
73
```

(Ilustración 5. Captura de pantalla del código en cual se observa el botón que oculta y muestra un texto “oculto”, un botón que hace de contador (+/- 1), un input para escribir en el modo en el que nos encontramos, y un botón que cambia el texto del botón según el estado (true / false con un ternario. Elaboración propia.)



Mini Panel de Control 🎮

Panel

Este es un texto secreto del panel 🤫

KLARA

Modo actual: KLARA

Contador

Valor: 7

(Ilustración 6. Captura de pantalla de la App donde mostramos las funcionalidades previamente dichas, botón que oculta el panel, en este caso, lo muestra, nuestro modo actual, y el contador. Elaboración propia.)

PREGUNTAS

PREGUNTA 1: ¿Qué diferencia hay entre onClick y onSubmit?

onClick responde a un clic en un elemento, mientras que onSubmit se dispara cuando un formulario se envía, independientemente de cómo se haya enviado.

PREGUNTA 2: ¿Por qué usamos e.preventDefault() en un formulario?

Para evitar el comportamiento por defecto del navegador, que es recargar la página al enviar el formulario.

PREGUNTA 3: ¿Qué es una “entrada controlada” y por qué usamos value + onChange?

Es un input cuyo valor está controlado por el estado de React, lo que permite tener control total sobre los datos y mantener la interfaz sincronizada.

PREGUNTA 4: En tu mini-reto, que estado(s) manejas y qué evento(s) los actualizan?

Se manejan estados para visibles, contador y modo, actualizados mediante eventos de clic y eventos de cambio en inputs.