

## Contenido

<b>Parte A: Input controlado (Controlled Component).....</b>	<b>2</b>
<b>Parte B: Envío del formulario (onSubmit).....</b>	<b>2</b>
<b>Parte C: Validación básica.....</b>	<b>3</b>
<b>Miniretillo (subimos el nivel chavales) .....</b>	<b>4</b>

## Parte A: Input controlado (Controlled Component)

```
1 import { useState } from 'react';
2 function App() {
3   const [username, setUsername] = useState('');
4   return (
5     <form>
6       Username:
7       <input
8         type="text"
9         value={username}
10        onChange={(e) => setUsername(e.target.value)}
11      />
12    </form>
13  );
14 }
15 export default App;
```

Ilustración 1. Input controlado. Elaboración propia

El valor del input está vinculado al estado username. Cada vez que el usuario escribe, el evento onChange actualiza el estado en tiempo real, manteniendo la interfaz y los datos sincronizados.

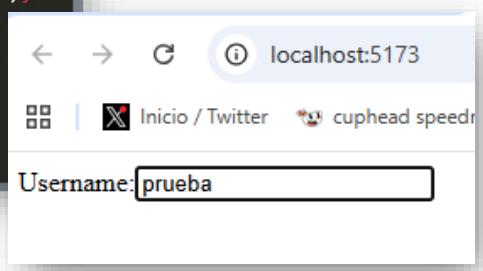


Ilustración 2. Visualización del resultado.  
Elaboración propia

## Parte B: Envío del formulario (onSubmit)

```
1 import { useState } from 'react';
2
3 function App() {
4   const [username, setUsername] = useState('');
5
6   // 1. Añade la función handleSubmit() para controlar el envío
7   const handleSubmit = (e) => {
8     e.preventDefault(); // Evita el refresco automático de la página
9     alert(username); // Muestra en una alerta el valor guardado en el estado
10  };
11
12  return (
13    // 2. Asigna la función al evento onSubmit y añade el botón
14    <form onSubmit={handleSubmit} style={{ padding: '20px' }}>
15      Username:
16      <input
17        type="text"
18        value={username}
19        onChange={(e) => setUsername(e.target.value)}
20      />
21      <button type="submit">Submit</button>
22    </form>
23  );
24 }
25
26 export default App;
```

Ilustración 3. handleSubmit. Elaboración propia

Añadir la función **handleSubmit** para controlar el envío del formulario, usando **e.preventDefault()** para que la página no se recargue. Ahora el botón de Submit muestra el valor actual del estado en una alerta, verificando que los datos se capturan correctamente.

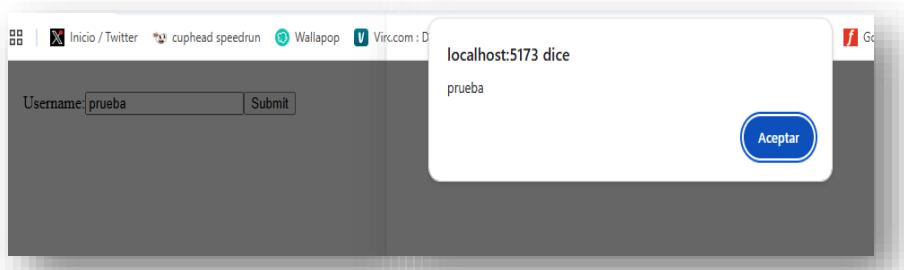


Ilustración 4. Visualización del resultado. Elaboración propia

## Parte C: Validación básica

```

1 import { useState } from 'react';
2
3 function App() {
4   const [username, setUsername] = useState('');
5   // 1. Estado adicional para el error
6   const [usernameError, setUsernameError] = useState('');
7
8   // 4. En handleSubmit, impide el envío si hay error
9   const handleSubmit = (e) => {
10     e.preventDefault();
11     if (usernameError) {
12       alert('No se puede enviar: el formulario contiene errores');
13     } else {
14       alert(username);
15     }
16   };
17
18   // 2. función handleUsername() que actualice el username y valide la
19   // longitud (mínimo 7 caracteres).
20   const handleUsername = (e) => {
21     const { value } = e.target;
22     setUsername(value);
23     if (value.length <= 6) {
24       setUsernameError('El username debe tener más de 6 caracteres');
25     } else {
26       setUsernameError('');
27     }
28   };
29
30   return (
31     <form onSubmit={handleSubmit}>
32       Username:
33       <input type="text" value={username} onChange={handleUsername} />
34       <p style={{ color: 'red' }}>{usernameError}</p>
35
36       <button>Submit</button>
37     </form>
38   );
39
40 export default App;

```

Implementación la **validación en tiempo real** mediante `handleUsername`, que comprueba que el nombre tenga al menos 7 caracteres antes de permitir el envío. Si el nombre es corto, se guarda un mensaje en el nuevo estado `usernameError` y se muestra automáticamente en rojo bajo el `input`.

Ilustración 5. Validación básica. Elaboración propia

Username:

El username debe tener más de 6 caracteres

Ilustración 7. Visualización de la validación.  
Elaboración propia

Username:

Ilustración 6. Visualización del resultado. Elaboración propia

## Miniretillo (subimos el nivel chavales)

**Miniretillo - Formulario Completo**

Username:

El username debe tener más de 6 caracteres

Email:

El email debe contener @ y .

Password:

La password debe tener al menos 8 caracteres

Ilustración 8. Visualización de las 3 validaciones. Elaboración propia

**Miniretillo - Formulario Completo**

Username:

Email:

Password:

Ilustración 9. Visualización del resultado con el botón Submit. Elaboración propia

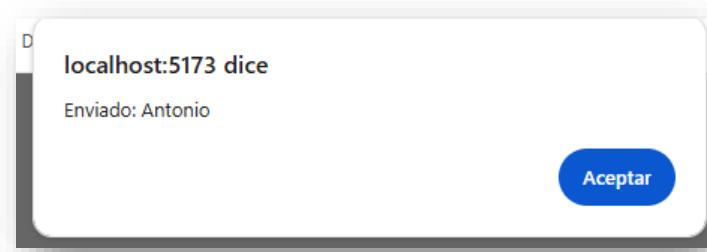
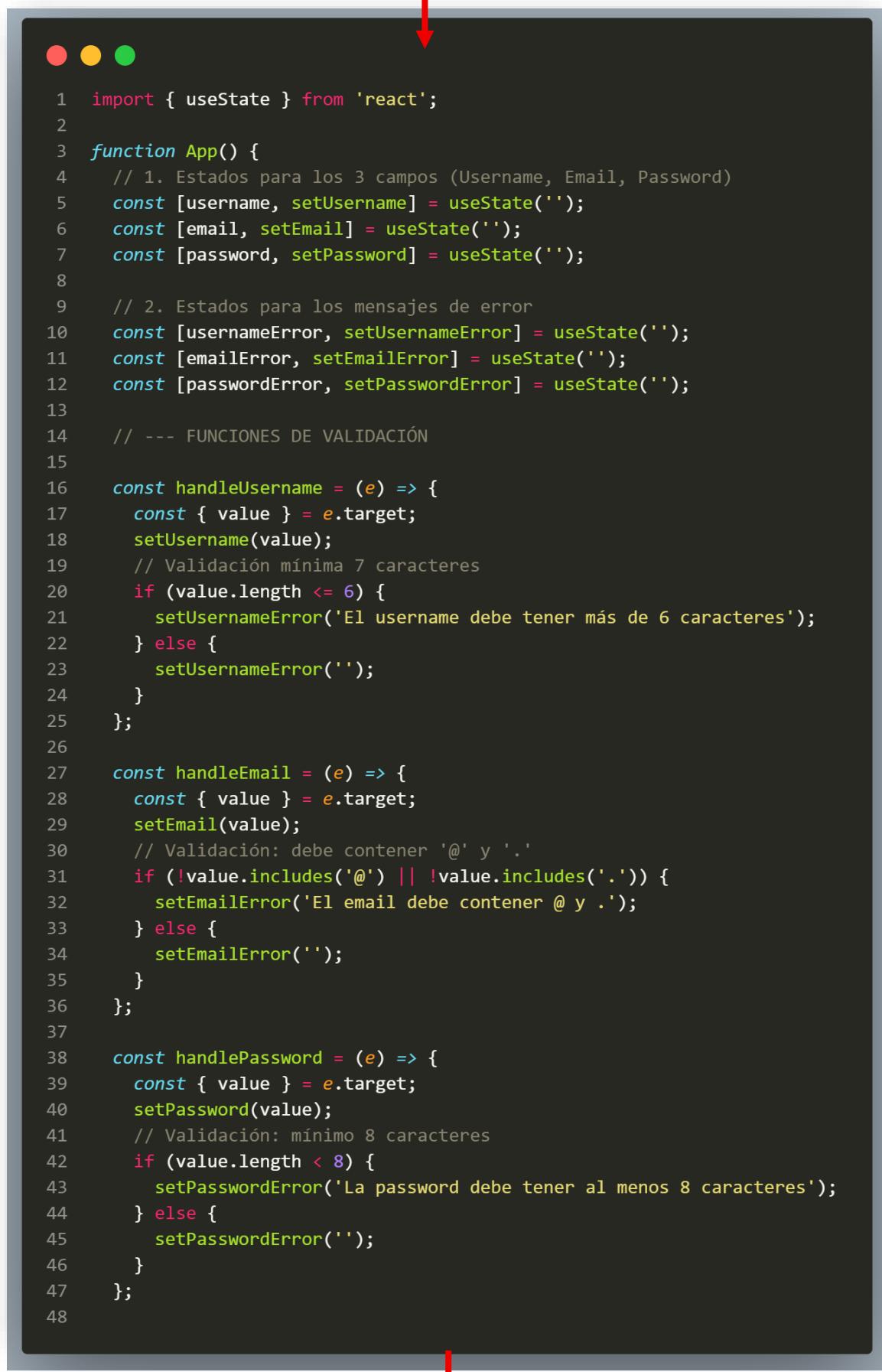


Ilustración 10. Visualización del resultado enviado. Elaboración propia

Ampliación el formulario a tres campos (username, email y password), creando estados para guardar los datos y otros para los mensajes de error. Funciones que validan en tiempo real: el email debe llevar '@' y '.', y la contraseña al menos 8 caracteres. Los errores aparecen automáticamente en rojo debajo de cada input para avisar al usuario. Por último, bloqueo del botón de 'Submit' para que no se pueda pulsar si falta algún dato o si hay errores, asegurando que el formulario sea correcto.



```
1 import { useState } from 'react';
2
3 function App() {
4     // 1. Estados para los 3 campos (Username, Email, Password)
5     const [username, setUsername] = useState('');
6     const [email, setEmail] = useState('');
7     const [password, setPassword] = useState('');
8
9     // 2. Estados para los mensajes de error
10    const [usernameError, setUsernameError] = useState('');
11    const [emailError, setEmailError] = useState('');
12    const [passwordError, setPasswordError] = useState('');
13
14    // --- FUNCIONES DE VALIDACIÓN
15
16    const handleUsername = (e) => {
17        const { value } = e.target;
18        setUsername(value);
19        // Validación mínima 7 caracteres
20        if (value.length <= 6) {
21            setUsernameError('El username debe tener más de 6 caracteres');
22        } else {
23            setUsernameError('');
24        }
25    };
26
27    const handleEmail = (e) => {
28        const { value } = e.target;
29        setEmail(value);
30        // Validación: debe contener '@' y '.'
31        if (!value.includes('@') || !value.includes('.')) {
32            setEmailError('El email debe contener @ y .');
33        } else {
34            setEmailError('');
35        }
36    };
37
38    const handlePassword = (e) => {
39        const { value } = e.target;
40        setPassword(value);
41        // Validación: mínimo 8 caracteres
42        if (value.length < 8) {
43            setPasswordError('La password debe tener al menos 8 caracteres');
44        } else {
45            setPasswordError('');
46        }
47    };
48}
```

Ilustración 11. Miniretillo. Elaboración propia



```
1  const handleSubmit = (e) => {
2      e.preventDefault();
3      if (usernameError || emailError || passwordError) {
4          alert('No se puede enviar: el formulario contiene errores');
5      } else {
6          alert(`Enviado: ${username}`);
7      }
8  };
9
10 // Deshabilitar el botón Submit
11 const esInvalido = usernameError || emailError || passwordError
12 || !username || !email || !password;
13
14 return (
15     <form onSubmit={handleSubmit} style={{ padding: '20px' }}>
16         <h2>Miniretillo - Formulario Completo</h2>
17
18         {/* Campo Username */}
19         <div>
20             Username:
21             <input type="text" value={username} onChange={handleUsername} />
22             <p style={{ color: 'red' }}>{usernameError}</p> /* Error en rojo */
23         </div>
24
25         {/* Campo Email */}
26         <div>
27             Email:
28             <input type="text" value={email} onChange={handleEmail} />
29             <p style={{ color: 'red' }}>{emailError}</p> /* Error en rojo */
30         </div>
31
32         {/* Campo Password */}
33         <div>
34             Password:
35             <input type="password" value={password} onChange={handlePassword} />
36             <p style={{ color: 'red' }}>{passwordError}</p> /* Error en rojo */
37         </div>
38
39         {/* Botón deshabilitado si existe algún error */}
40         <button disabled={esInvalido}>Submit</button>
41     </form>
42 );
43 }
44
45 export default App;
```

Ilustración 12. Continuación del código. Elaboración propia