

ÍNDICE

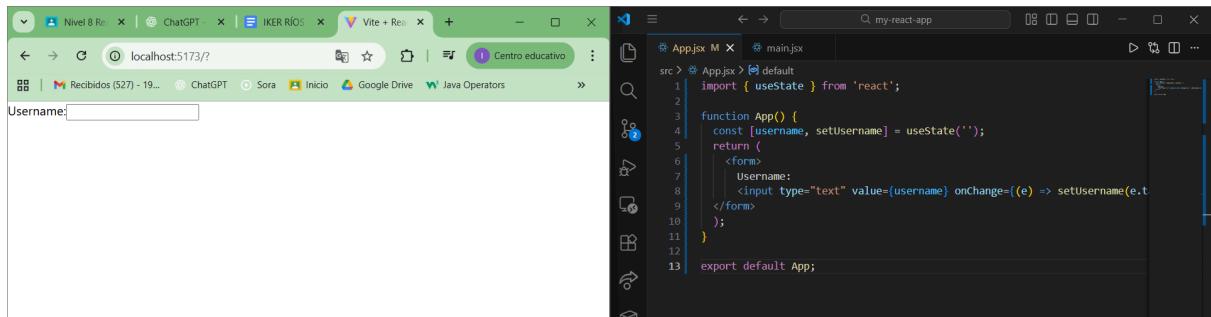
[Parte A: Input controlado \(Controlled Component\):](#)

[Parte B: Envío del formulario \(onSubmit\):](#)

[Parte C: Validación básica:](#)

[Miniretillo:](#)

Parte A: Input controlado (Controlled Component):

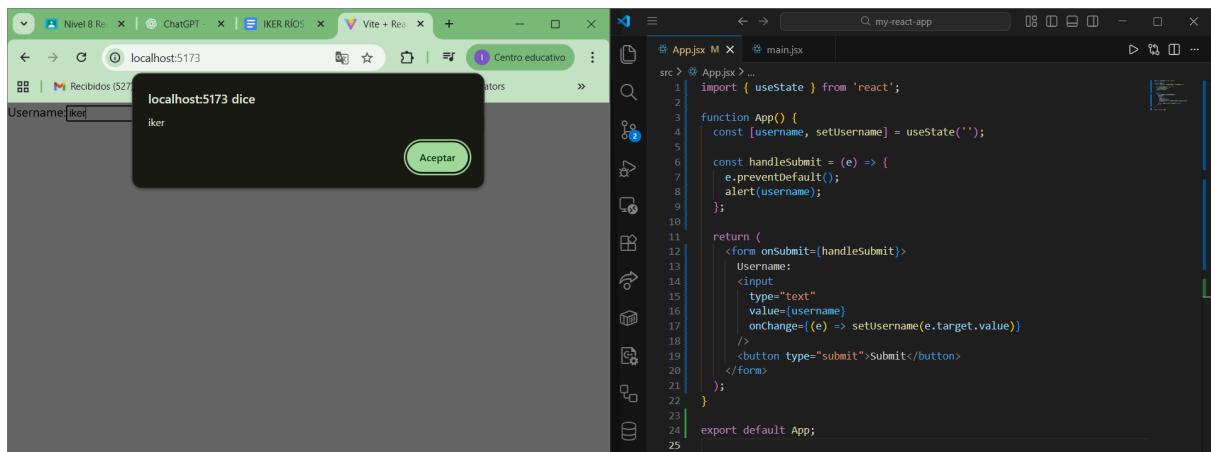


The screenshot shows a browser window at localhost:5173/ with a text input field containing "Username:". To the right is a code editor with the file `main.jsx` open, showing the following code:

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

(Captura de pantalla del código, elaboración propia)

Parte B: Envío del formulario (onSubmit):



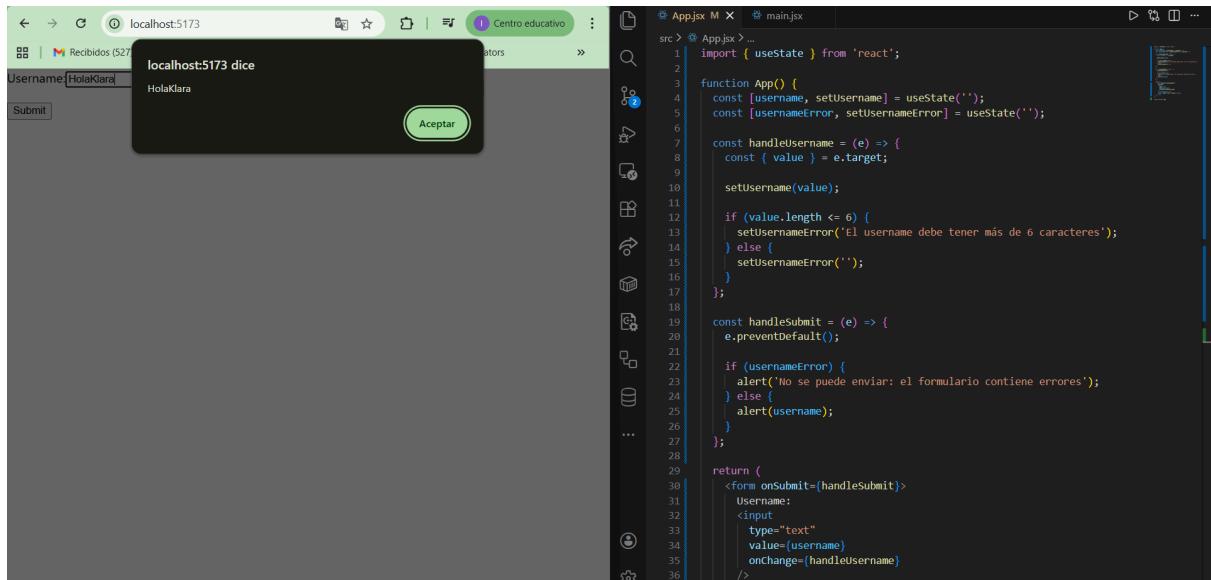
The screenshot shows a browser window at localhost:5173 with a text input field containing "Username: iker". A modal dialog box is displayed with the text "localhost:5173 dice" and "iker" above a "Aceptar" button. To the right is a code editor with the file `main.jsx` open, showing the following code:

```
1 import { useState } from 'react';
2
3 function App() {
4   const [username, setUsername] = useState('');
5
6   const handleSubmit = (e) => {
7     e.preventDefault();
8     alert(username);
9   };
10
11   return (
12     <form onSubmit={handleSubmit}>
13       Username:
14         <input
15           type="text"
16           value={username}
17           onChange={(e) => setUsername(e.target.value)} />
18       <button type="submit">Submit</button>
19     </form>
20   );
21 }
22
23
24 export default App;
```

(Captura de pantalla del código, elaboración propia)

Parte C: Validación básica:

(Captura de pantalla del código, elaboración propia)



The screenshot shows a browser window at localhost:5173 with a text input field containing "Username: HolaKlara". A modal dialog box is displayed with the text "localhost:5173 dice" and "HolaKlara" above a "Aceptar" button. To the right is a code editor with the file `main.jsx` open, showing the following code:

```
1 import { useState } from 'react';
2
3 function App() {
4   const [username, setUsername] = useState('');
5   const [usernameError, setUsernameError] = useState('');
6
7   const handleUsername = (e) => {
8     const { value } = e.target;
9
10     setUsername(value);
11
12     if (value.length <= 6) {
13       setUsernameError('El username debe tener más de 6 caracteres');
14     } else {
15       setUsernameError('');
16     }
17   };
18
19   const handleSubmit = (e) => {
20     e.preventDefault();
21
22     if (usernameError) {
23       alert('No se puede enviar: el formulario contiene errores');
24     } else {
25       alert(username);
26     }
27   };
28
29
30   return (
31     <form onSubmit={handleSubmit}>
32       Username:
33         <input
34           type="text"
35           value={username}
36           onChange={handleUsername} />
37     </form>
38   );
39 }
40
41
42 export default App;
```

The screenshot shows a browser window at localhost:5173. A modal dialog box displays the message "localhost:5173 dice" followed by "No se puede enviar: el formulario contiene errores". Below the modal, the text "El username debe tener más de 6 caracteres" is visible. A "Aceptar" button is at the bottom right of the modal. To the right of the browser is a code editor showing the file main.jsx. The code defines a function App() that handles a form submission. It checks if the username length is less than or equal to 6, setting an error message. It also prevents default form behavior and shows an alert if there are errors. The code uses useState from react.

```

import { useState } from 'react';
const [username, setUsername] = useState('');
const [usernameError, setUsernameError] = useState('');

const handleUsername = (e) => {
  const { value } = e.target;
  setUsername(value);
  if (value.length <= 6) {
    setUsernameError('El username debe tener más de 6 caracteres');
  } else {
    setUsernameError('');
  }
};

const handleSubmit = (e) => {
  e.preventDefault();
  if (usernameError) {
    alert('No se puede enviar: el formulario contiene errores');
  } else {
    alert(username);
  }
};

return (
  <form onSubmit={handleSubmit}>
    Username:
    <input type="text" value={username} onChange={handleUsername}>
    </input>
    <p>{usernameError}</p>
  </form>
);

```

(Captura de pantalla del código, elaboración propia)

Miniretillo:

The screenshot shows a browser window at localhost:5173. The form has three fields: Username (ikerRios), Email (with red validation text "Email no válido"), and Password (with red validation text "Mínimo 8 caracteres"). A "Submit" button is at the bottom. To the right is a code editor for main.jsx. The code defines a function App() that creates a form with three input fields. Each field has an onChange handler that updates state variables (username, email, password). The form also has a submit button. Red validation messages are generated using conditional styling based on the error state of each input field.

```

function App() {
  <form onSubmit={handleSubmit}>
    <div>
      <label>Username</label><br />
      <input type="text" value={username} onChange={handleUsername}>
      <p style={{ color: 'red' }}>{usernameError}</p>
    </div>
    <div>
      <label>Email</label><br />
      <input type="text" value={email} onChange={handleEmail}>
      <p style={{ color: 'red' }}>{emailError}</p>
    </div>
    <div>
      <label>Password</label><br />
      <input type="password" value={password} onChange={handlePassword}>
      <p style={{ color: 'red' }}>{passwordError}</p>
    </div>
    <button type="submit" disabled={usernameError || emailError || passwordError}>
      Submit
    </button>
  </form>
}

export default App;

```

(Captura de pantalla del código, elaboración propia)

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

function App() {
  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const [usernameError, setUsernameError] = useState('');
  const [emailError, setEmailError] = useState('');
  const [passwordError, setPasswordError] = useState('');

  const handleUsername = (e) => {
    const value = e.target.value;
    setUsername(value);

    if (value.length < 7) {
      setUsernameError('Mínimo 7 caracteres');
    } else {
      setUsernameError('');
    }
  };

  const handleEmail = (e) => {
    const value = e.target.value;
    setEmail(value);

    if (!value.includes('@') || !value.includes('.')) {
      setEmailError('Email no válido');
    } else {
      setEmailError('');
    }
  };

  const handlePassword = (e) => {
    const value = e.target.value;
    setPassword(value);

    if (value.length < 8) {
      setPasswordError('Mínimo 8 caracteres');
    } else {
      setPasswordError('');
    }
  };
}
```

```

const handleSubmit = (e) => {
  e.preventDefault();

  if (usernameError || emailError || passwordError) {
    alert('Hay errores en el formulario');
    return;
  }

  alert('Formulario enviado');
};

return (
  <form onSubmit={handleSubmit}>
    <div>
      <label>Username</label><br />
      <input type="text" value={username} onChange={handleUsername}>
    </div>
    <p style={{ color: 'red' }}>{usernameError}</p>
  </div>

  <div>
    <label>Email</label><br />
    <input type="text" value={email} onChange={handleEmail}>
    <p style={{ color: 'red' }}>{emailError}</p>
  </div>

  <div>
    <label>Password</label><br />
    <input type="password" value={password}>
    <input type="text" value={password} onChange={handlePassword}>
    <p style={{ color: 'red' }}>{passwordError}</p>
  </div>

  <button
    type="submit"
    disabled={usernameError || emailError || passwordError}>
    Submit
  </button>
</form>
);
}

```

```
export default App;
```