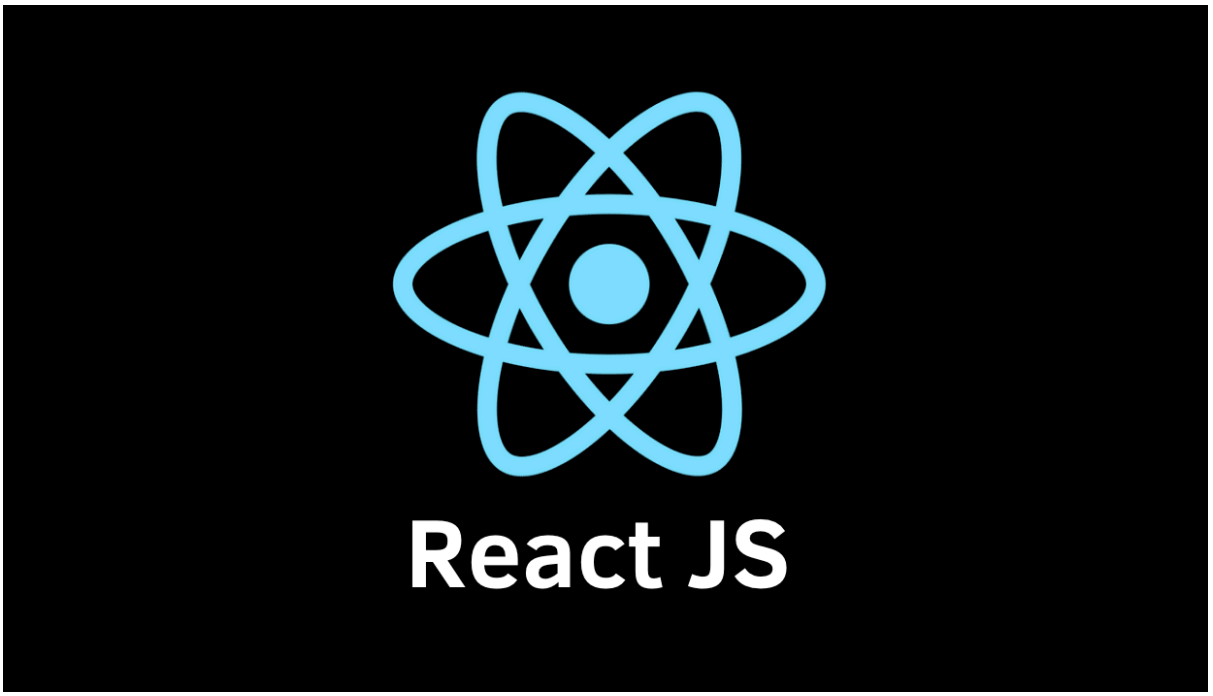
Desarrollo web en entorno cliente - Nivel 8 React

Pablo Valero López *26/01/2026*

2ºDAW



Objetivo

● Crear un formulario en React usando estado (useState) como fuente única de los datos.
● Gestionar cambios de inputs con onChange y actualizar el estado.
● Controlar el envío con onSubmit y evitar el refresco con preventDefault().
● Aplicar validación básica y mostrar mensajes de error en la UI.

Parte A: Input controlado (Controlled Component)

En la siguiente imagen se muestra el código de un input controlado.

Imagen 1: Código de input controlado. Elaboración propia.

En la siguiente imagen se muestra el resultado del input controlado.

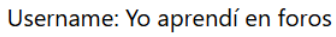


Imagen 2: Resultado de input controlado. Elaboración propia.

Parte B: Envío del formulario (onSubmit)

En la siguiente imagen se muestra el código del formulario.

```
import { useState } from 'react';
import './style.css';

function App() {

    const [username, setUsername] = useState('');
    const handleSubmit = (e) => {
        e.preventDefault();
        alert(username);
        };

    return (
            <form className='centrar' onSubmit={handleSubmit}>
                Username:
                <input
                    type="text"
                    value={username}
                    onChange={(e) => setUsername(e.target.value)}
                />
                <button>Submit</button>
            </form>
        );
}
export default App;
```

Imagen 3: Código de formulario. Elaboración propia.

En la siguiente imagen se muestra el resultado del formulario.

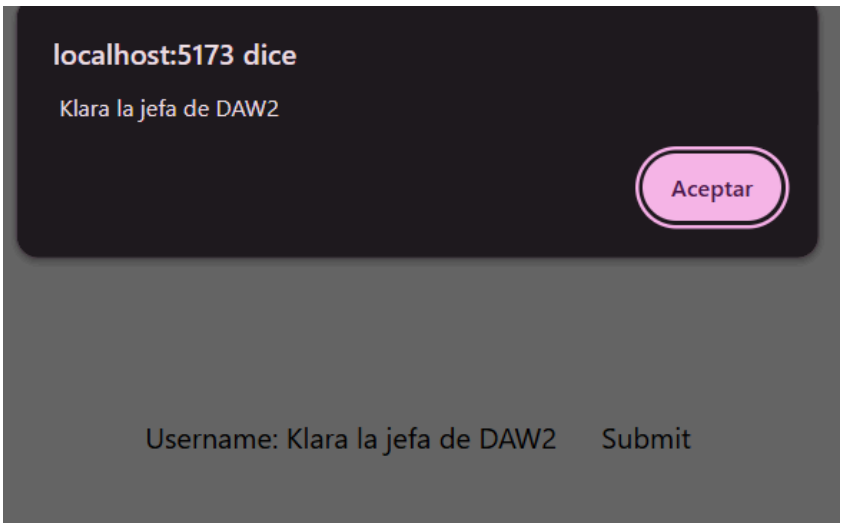


Imagen 4: Resultado de formulario. Elaboración propia.

Parte C: Validación básica

En la siguiente imagen se muestra el código de la validación básica

```jsx
import { useState } from 'react';
import './style.css';

function App() {
    const [username, setUsername] = useState('');
    const handleSubmit = (e) => {
        e.preventDefault();
        if (usernameError) {
            alert('No se puede enviar: el formulario contiene errores');
        } else {
            alert(username);
        }
        };
    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('');
    }
};
    return (
            <form className='centrar' onSubmit={handleSubmit}>
                Username:
                <input
                    type="text"
                    value={username}
                    onChange={handleUsername}
                />
                <button>Submit</button>
                <p>{usernameError}</p>
            </form>
        );
}
export default App;
```

Imagen 5: Código de formulario. Elaboración propia.

En las siguientes imágenes se muestran los resultados del formulario básico

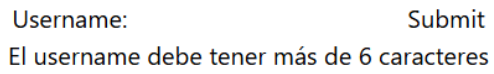


Imagen 6: Resultado de formulario. Elaboración propia.

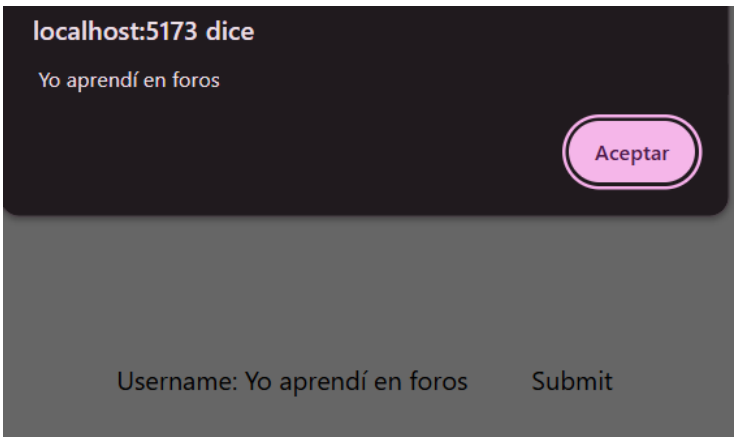


Imagen 7: Resultado de formulario. Elaboración propia.

Miniretillo (subimos el nivel chavales)

En las siguientes imágenes se muestra el código del miniretillo.

```
import { useState } from 'react';
import './style.css';

function App() {
    const [username, setUsername] = useState('');
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const handleSubmit = (e) => {
        e.preventDefault();
        if (usernameError || emailError || passwordError) {
            alert('No se puede enviar: el formulario contiene errores');
        } else {
            alert(`${username}, ${email}, ${password}`);
        }
        };
        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 [emailError, setEmailError] = useState('');
        const handleEmail = (e) => {
            const { value } = e.target;
            setEmail(value);
            if (!value.includes('@') || !value.includes('.')) {
                setEmailError('El email debe contener @ y un .');
            } else {
                setEmailError('');
            }
```

Imagen 8: Código de miniretillo. Elaboración propia.

```
const [passwordError, setPasswordError] = useState('');
const handlePassword = (e) => {
    const { value } = e.target;
    setPassword(value);
    if (value.length<8){
        setPasswordError('La contraseña debe tener mínimo 8 caracteres.');
    }else{
        setPasswordError('');
    }
}
```

Imagen 9: Código de miniretillo. Elaboración propia.

```
return (
        <form className='centrar' onSubmit={handleSubmit}>
            Username:
            <input
                type="text"
                value={username}
                onChange={handleUsername}
            />
            Email:
            <input
                type="email"
                value={email}
                onChange={handleEmail}
            />
            Password:
            <input
                type="password"
                value={password}
                onChange={handlePassword}
            />
            <button>Submit</button>
            <p className='error'>{usernameError}</p>
            <p className='error'>{emailError}</p>
            <p className='error'>{passwordError}</p>
        </form>
    );
}
export default App;
```

Imagen 10: Código de miniretillo. Elaboración propia.

En las siguientes imágenes se muestra el resultado del miniretillo.

Username:Klaefa Email: Password:••••• Submit

El username debe tener más de 6 caracteres

El email debe contener @ y un .

La contraseña debe tener mínimo 8 caracteres.

Imagen 11: Resultado de miniretillo. Elaboración propia.

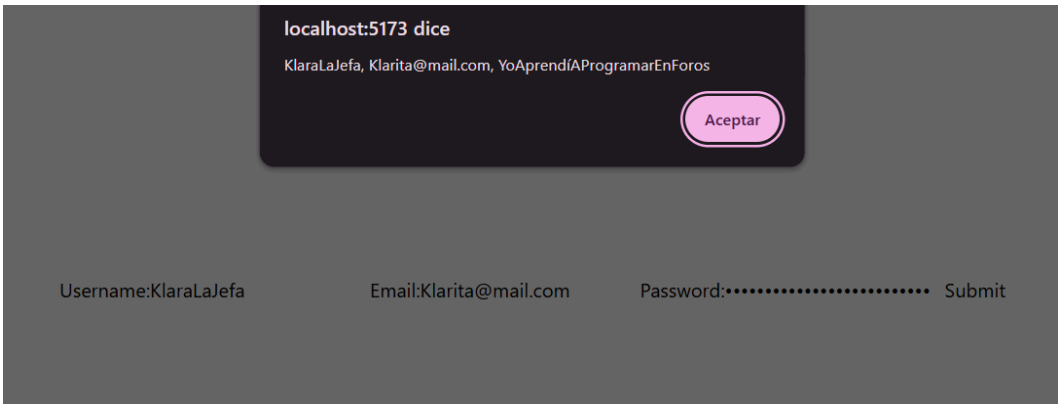


Imagen 12: Resultado de miniretillo. Elaboración propia.