GUÍA PRÁCTICA 02: Formularios en React con TypeScript

Objetivo: Aprender a crear formularios controlados en React, gestionar el estado con useState, manejar eventos y mostrar mensajes personalizados.

Requisitos

Antes de comenzar, asegúrate de:

- Tener instalado **Node.js** y **Visual Studio Code**
- Haber creado un proyecto con Vite + React + TypeScript

npm create vite@latest mi-formulario-app -- --template react-ts

cd mi-formulario-app

npm install

npm run dev

1. ¿Qué es un formulario controlado?

Un **formulario controlado** en React es aquel en el que **React controla el valor de los campos** mediante el estado (useState).
Cada vez que el usuario escribe algo, se actualiza el estado y el valor del campo.

2. Formulario básico

Creamos un formulario que captura el nombre y el mensaje de un usuario.

```
// src/components/FormularioContacto.tsx

import { useState, FormEvent } from 'react';


export default function FormularioContacto() {
 const [nombre, setNombre] = useState('');
 const [mensaje, setMensaje] = useState('');
 const [enviado, setEnviado] = useState(false);


 const manejarEnvio = (e: FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  console.log(`Nombre: ${nombre}, Mensaje: ${mensaje}`);
  setEnviado(true);
  setNombre('');
  setMensaje('');
 };
```

```
return (
 <div>
  <h2>Formulario de Contacto</h2>

  {enviado && <p style={{ color: 'green' }}>¡Gracias por tu mensaje!</p>}

  <form onSubmit={manejarEnvio}>
   <div>
    <label htmlFor="nombre">Nombre:</label><br />
    <input
     id="nombre"
     type="text"
     value={nombre}
     onChange={(e) => setNombre(e.target.value)}
     required
    />
   </div>
   <div>
    <label htmlFor="mensaje">Mensaje:</label><br />
    <textarea
     id="mensaje"
     value={mensaje}
     onChange={(e) => setMensaje(e.target.value)}
     required
    />
   </div>
   <button type="submit">Enviar</button>
  </form>
 </div>
);
```

}

3. ¿Cómo funciona este formulario?

Elemento	Función
useState	Crea variables de estado para guardar el contenido de los campos.
onChange	Escucha los cambios en los campos y actualiza el estado.
onSubmit	Controla el envío del formulario y previene el comportamiento por defecto del navegador.
required	Valida que el usuario no deje campos vacíos.

4. Agregar el formulario a tu App

```
// src/App.tsx

import FormularioContacto from './components/FormularioContacto';


function App() {
 return (
  <div style={{ padding: '20px' }}>
   <h1>Mi App con React + TypeScript</h1>
   <FormularioContacto />
  </div>
 );
}


export default App;
```

5. Mejora: Mostrar resumen personalizado

Vamos a mostrar un resumen del mensaje que se envió.

Modifica el componente así:

```
{enviado && (
 <div style={{ marginTop: '10px', background: '#e0ffe0', padding: '10px' }}>
  <p>Gracias, <strong>{nombre}</strong>.</p>
  <p>Tu mensaje fue: "{mensaje}"</p>
 </div>
```

)}

---

6. Conceptos clave

Término	Significado
useState	Hook que permite manejar datos internos del componente
onChange	Evento que se activa al escribir en un campo
onSubmit	Evento que se dispara al enviar un formulario
Componente controlado	Un campo cuyo valor es gestionado por React (con useState)

7. Actividad Final

Reto práctico: Crea un formulario de registro de estudiante

El formulario debe tener:

- Nombre completo
- Correo electrónico
- Carrera o especialidad
- Comentarios

Requisitos:

- Todos los campos deben ser requeridos (required)
- Mostrar un resumen al final, tipo:

¡Gracias Juan Pérez!

Hemos registrado tu correo: juan@mail.com

Carrera: Computación

Comentarios: Me gusta React.