



**UNIVERSIDAD AUTÓNOMA DE CHIAPAS
FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN C-I**

**LICENCIATURA EN INGENIERÍA EN DESARROLLO Y
TECNOLOGÍAS DE
SOFTWARE**

MATERIA: TALLER DE DESARROLLO 3

DOCENTE: LUIS GUTIÉRREZ ALFARO

ACTIVIDAD:

Frontend: Formulario React con Validación y Consumo de API

**ALUMNO: JOSE RUBÉN CLEMENTE CORZO
100020762
5M**

10 DE OCTUBRE DEL 2025

TUXTLA GUTIÉRREZ, CHIAPAS

1 Introducción

Este pequeño sistema permite el registro de estudiantes mediante un formulario web, validando los datos en el frontend y enviándolos a un backend Express para su almacenamiento. El proceso incluye validaciones, comunicación por API REST, manejo de respuestas basado en lo propuesto como ejercicio.

Acceso al repositorio de github: <https://github.com/Ruben2216/wepApp.git>

2 Flujo General del Proceso

1. El usuario accede a la interfaz web y llena el formulario de registro.
2. El frontend valida los datos antes de permitir el envío.
3. Los datos se agrupan en un objeto JSON y se envían al backend mediante una petición HTTP POST.
4. El backend recibe los datos, los procesa y responde con un mensaje de confirmación.
5. El usuario recibe retroalimentación visual (alertas y mensajes en consola).

3 Frontend: Captura, Validación y Envío de Datos

3.1 Estructura del Formulario

El formulario está construido con React y utiliza componentes funcionales. Cada campo del formulario está vinculado a un estado local (`formData`), lo que permite capturar los valores ingresados por el usuario de manera controlada.

- **Campos:** Documento, Nombre, Apellido, Teléfono, Correo.
- **Validaciones:**
 - Todos los campos son obligatorios (`required`).
 - El correo se valida con `type="email"`.
 - El teléfono acepta sólo dígitos y máximo 10 caracteres (`type="tel"`, `pattern`, `maxLength`).
- **Retroalimentación:** El navegador impide el envío si algún campo no cumple las validaciones.

Formulario

Documento:

Nombre:

Apellido:

Teléfono:

Correo:

Registrar

Cancelar

Figure 1: Formulario de registro de estudiantes con validaciones nativas.

3.2 Manejo de Estado y Agrupación de Datos

Cada cambio en los inputs actualiza el estado `formData`. Al enviar el formulario, este estado se convierte directamente en un JSON para facilitar la comunicación con el backend.



Figure 2: Formulario de registro de estudiantes con validaciones nativas.

3.3 Envío de Datos y Manejo de Respuestas

Al enviar el formulario, se realiza una petición POST al backend. Se muestra en consola el JSON enviado y la respuesta recibida, lo que facilita la depuración y la verificación del flujo.

```
const handleSubmit = async (event) => {
  event.preventDefault();
  console.log("Datos a enviar:", formData);
  try {
    const response = await
fetch("http://localhost:4000/api/save", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(formData),
  });
    const result = await response.json();
    if (response.ok) {
      console.log("Datos enviados exitosamente:",
result);
      alert("Estudiante registrado exitosamente");
    } else {
      console.error("Error al enviar datos:", result);
    }
  } catch (error) {
    console.error("Error al enviar el formulario:",
error);
  }
};
```

Figure 3: Formulario de registro de estudiantes con validaciones nativas.

```
Datos a enviar:                               ListaEstudiantes.jsx:118
  {documento: 'INE', nombre: 'Jose Ruben', apellido: 'Clemente Corzo', telefono: '9612215796', correo: 'jose.clemente48@unach.mx'}
  apellido: "Clemente Corzo"
  correo: "jose.clemente48@unach.mx"
  documento: "INE"
  nombre: "Jose Ruben"
  telefono: "9612215796"
  [[Prototype]]: Object

Datos enviados exitosamente:                  ListaEstudiantes.jsx:128
  {message: 'Datos recibidos correctamente'}
  message: "Datos recibidos correctamente"
  [[Prototype]]: Object

> |
```

Figure 4: Consola mostrando el JSON enviado y la respuesta del backend.

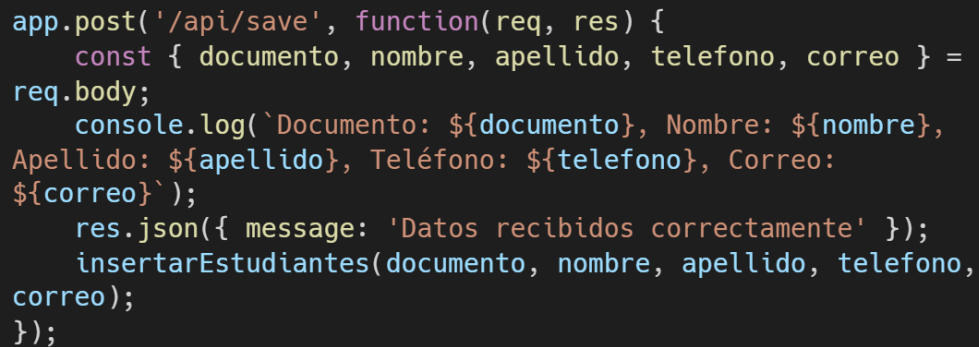
```
Documento: INE, Nombre: Jose Ruben, Apellido: Clemente Corzo, Teléfono: 9612215796, Correo: jose.clemente48@unach.mx
{
  documento: 'INE',
  nombre: 'Jose Ruben',
  apellido: 'Clemente Corzo',
  telefono: '9612215796',
  correo: 'jose.clemente48@unach.mx'
}
Estudiante insertado exitosamente null
□
```

Figure 5: Respuesta del backend desde consola de Visual Studio Code

4 Backend: Recepción, Procesamiento y Respuesta

4.1 Recepción de Datos

El backend está implementado con Express y expone el endpoint `/api/save` para recibir los datos. Utiliza `express.json()` para procesar el cuerpo de la petición.



```
app.post('/api/save', function(req, res) {  
  const { documento, nombre, apellido, telefono, correo } =  
  req.body;  
  console.log(`Documento: ${documento}, Nombre: ${nombre},  
  Apellido: ${apellido}, Teléfono: ${telefono}, Correo:  
  ${correo}`);  
  res.json({ message: 'Datos recibidos correctamente' });  
  insertarEstudiantes(documento, nombre, apellido, telefono,  
  correo);  
});
```

Figure 6: End point a donde se dirige el método Post desde el frontend

4.2 Almacenamiento y Respuesta

Los datos recibidos se almacenan en una base de datos PostgreSQL mediante la función `insertarEstudiantes`. El backend responde con un mensaje JSON que confirma la recepción exitosa.

```
async function insertarEstudiantes(
  documento,
  nombre,
  apellido,
  telefono,
  correo
) {
  try {
    await dbConnection.none(
      "INSERT INTO estudiantes (documento, nombre,
apellido, telefono, correo) VALUES ($1, $2, $3, $4, $5)",
      [documento, nombre, apellido, telefono, correo]
    );
    console.log("Estudiante insertado exitosamente");
  } catch (error) {
    console.error("Error al insertar estudiante:", error);
  }
}
```

5 Mensajes de Éxito y Manejo de Errores

- **Frontend:** El usuario recibe alertas visuales y mensajes en consola confirmando el registro o informando de errores.
- **Backend:** Responde con un JSON que indica el éxito o el error en la operación.

6 Resumen y Justificación Técnica

Este flujo garantiza:

- Validación robusta y amigable para el usuario.
- Comunicación eficiente y segura entre frontend y backend.
- Retroalimentación clara en cada paso del proceso.
- Facilidad de mantenimiento y extensión futura.

6.1 Estructura de Componentes (Frontend)

El frontend implementa una arquitectura modular basada en componentes React

- **ListaEstudiantes.jsx:** Componente principal que maneja el estado y la lógica de negocio.
- **EstudianteForm.jsx:** Componente presentacional para el formulario de registro.
- **EstudianteTable.jsx:** Componente presentacional para mostrar la lista de estudiantes.
- **Formulario.jsx:** Componente adicional (legacy) para otros tipos de registro.