#### LICENCIATURA EN INFORMÁTICA UNIVERSIDAD AUTÓNOMA DE SINALOA



#### Nombre:

Ochoa Leòn Natalia

#### Maestro:

José Manuel Cazarez Alderete

#### Materia:

Desarrollo web lado del servidor

#### Actividad:

Actividad final: CRUD completo

Grupo:

2-1

Culiacán Sinaloa, 30 de mayo de 2025

Aplicación web (front y back)

#### CÓDIGO DE LAS RUTAS Y COMPONENTES VISUALES

#### **BACK** (rutas)

Este es el servidor backend que define las rutas que se comunican con la base de datos.

#### app.js

```
//CRUD

// CREATE
app.post('/empleados', async (req, res) => {
    const { nombre, telefono, fecha_de_nacimiento, sueldo } = req.body;
    const data = await empleados.create({ nombre, telefono, fecha_de_nacimiento, sueldo });
    res.send(data);
});
```

Esta ruta usa el método POST y se usa para crear un nuevo empleado, se llama cuando el usuario envía el formulario en el frontend.

```
//READ
app.get('/empleados', async (req, res) => {
    const data = await empleados.findAll();
    res.send(data);
});
```

Esta ruta usa el método GET y se usa para leer todos los empleados de la base de datos y se llama automáticamente al cargar el frontend.

```
//UPDATE
app.put('/empleados/:id', async (req, res) => {
   const { nombre, telefono, fecha_de_nacimiento, sueldo } = req.body;
   const { id } = req.params;
   const data = await empleados.update({ nombre, telefono, fecha_de_nacimiento, sueldo }, { where: { id } });
   res.send(data);
});
```

En esta ruta PUT se edita y actualiza un empleado ya existente

```
//DELETE
app.delete('/empleados/:id', async (req, res) => {
   const { id } = req.params;
   const data = await empleados.destroy({ where: { id } });
   res.send({ eliminado: data });
});
```

Esta ruta DELETE elimina un empleado por ID. Se activa cuando el usuario confirma la eliminación desde el frontend.

#### conexion.js

```
const { Sequelize } = require('sequelize');

// Conexión a base de datos SQLite
const conexion = new Sequelize({
   dialect: 'sqlite',
   storage: './basedatos.sqlite'
});

module.exports = conexion;
```

Este archivo se encarga de conectar Sequelize con la base de datos SQLite.La base de datos se guarda en un archivo local llamado **basedatos.sqlite** de la cual nos permite trabajar con bases de datos usando JavaScript sin escribir SQL directamente.

#### empleados.js

```
const { DataTypes } = require('sequelize');
const sequelize = require('../conexion');

const empleados = sequelize.define('empleados', {
    id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
    nombre: { type: DataTypes.STRING },
    telefono: { type: DataTypes.STRING },
    fecha_de_nacimiento: { type: DataTypes.DATEONLY },
    sueldo: { type: DataTypes.FLOAT }
}, {
    timestamps: false
});

module.exports = empleados;
```

Este archivo define el modelo empleado con Sequelize, cada empleado tiene id, nombre, teléfono, fecha de nacimiento y sueldo y se usa en el app.js para guardar, buscar, etc.

```
    PS C:\Users\Natilla\Documents\Desarrollo web\ProyectoFinalCRUDcompleto> node app.js http://localhost:3000
    Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='empleados'; Executing (default): PRAGMA INDEX_LIST(`empleados`)
    Base de datos conectada y tabla empleados creada
    PS C:\Users\Natilla\Documents\Desarrollo web\ProyectoFinalCRUDcompleto>
```

Backend funcionando

#### **FRONT** (componentes visuales)

#### EmpleadoForm.jsx

Esta función envía los datos del formulario al backend, si hay un empleado seleccionado, lo actualiza, si no, lo crea.

#### EmpleadoList.jsx

```
const handleChange = e => setForm({ ...form, [e.target.name]: e.target.value });

const handleSubmit = async (e) => {
    e.preventDefault();
    if (empleadoseleccionado) {
        await axios.put(`http://localhost:3000/empleados/${empleadoSeleccionado.id}`, form);
        setEmpleadoSeleccionado(null);
    } else {
        await axios.post('http://localhost:3000/empleados', form);
    }
    setForm({ nombre: '', telefono: '', fecha_de_nacimiento: '', sueldo: '' });
    fetchEmpleados();
};

return (
    <form onSubmit={handleSubmit} className="formulario">
         <h2>{ (Apy-{empleadoSeleccionado ? 'Editar empleado' : 'Agregar nuevo empleado'}</h2>
         <input name="nombre" placeholder="Nombre" value={form.nombre} onChange={handleChange} required />
         <input name="recha_de_nacimiento" type="date" value={form.telefono} onChange={handleChange} required />
         <input name="fecha_de_nacimiento" type="date" value={form.telefono} onChange={handleChange} required />
         <input name="sueldo" placeholder="Sueldo" type="number" value={form.sueldo} onChange={handleChange} required />
         </form>
    );
}
```

Cada fila en la lista muestra un empleado. Tiene botones para editar y eliminar, que usan funciones conectadas al backend.



#### **AGREGAR**



#### **EMPLEADO EN LISTA AGREGADO**



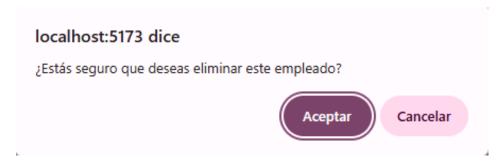
#### **ACTUALIZAR EMPLEADO**



#### **EMPLEADO ACTUALIZADO**



#### ALERTA DE ELIMINAR UN EMPLEADO



#### LISTA ACTUALIZADA

Lista de empleados

#### **REPOSITORIO DE GITHUB:**

https://github.com/NataliaOchoa/ProyectoFinalCRUD.git