Práctica 4: CRUD de Usuarios con Node.js, MySQL y HTML (Modelo MVC)

Objetivos de aprendizaje

- Comprender el patrón MVC (Modelo-Vista-Controlador).
- Conectarse a una base de datos MySQL desde Node.js.
- Crear una aplicación CRUD (Crear, Leer, Actualizar y Eliminar).
- Utilizar formularios HTML para interactuar con el servidor.
- Estructurar un proyecto profesional en Node.js.

Requisitos

- MySQL instalado y configurado.
- Node.js y NPM instalados.
- Visual Studio Code.
- Navegador Web.
- Paquetes: express, mysql2, body-parser, ejs.

Estructura del proyecto

```
crud-mvc/
├── app.js
├── config/
│   └── db.js
├── controllers/
│   └── userController.js
├── models/
│   └── userModel.js
├── public/
│   └── styles.css
├── routes/
│   └── userRoutes.js
├── views/
│   ├── index.ejs
│   ├── form.ejs
├── package.json
```

Paso 1: Crear la base de datos y tabla

En tu cliente de MySQL o phpMyAdmin, ejecuta:

```
CREATE DATABASE crud_node;

USE crud_node;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  correo VARCHAR(100) NOT NULL
);
```

Paso 2: Inicializar el proyecto

```
mkdir crud-mvc
cd crud-mvc
npm init -y
npm install express mysql2 body-parser ejs
```

Paso 3: Configurar la base de datos

config/db.js

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '', // coloca tu contraseña si tienes una
  database: 'crud_node'
});

connection.connect(err => {
  if (err) throw err;
  console.log('Conectado a la base de datos MySQL.');
```

```
});

module.exports = connection;
```

Paso 4: Crear el modelo

models/userModel.js

```
const db = require('../config/db');

const User = {
  getAll: callback => {
    db.query('SELECT * FROM users', callback);
  },

  getById: (id, callback) => {
    db.query('SELECT * FROM users WHERE id = ?', [id], callback);
  },

  create: (data, callback) => {
    db.query('INSERT INTO users SET ?', data, callback);
  },

  update: (id, data, callback) => {
    db.query('UPDATE users SET ? WHERE id = ?', [data, id], callback);
  },

  delete: (id, callback) => {
    db.query('DELETE FROM users WHERE id = ?', [id], callback);
  }
};

module.exports = User;
```

Paso 5: Crear el controlador

controllers/userController.js

```
const User = require('../models/userModel');

exports.index = (req, res) => {
  User.getAll((err, rows) => {
    if (err) throw err;
    res.render('index', { users: rows });
  });
};

exports.form = (req, res) => {
  res.render('form', { user: null });
};

exports.create = (req, res) => {
  const { nombre, correo } = req.body;
  User.create({ nombre, correo }, err => {
    if (err) throw err;
    res.redirect('/');
  });
};

exports.edit = (req, res) => {
  const id = req.params.id;
  User.getById(id, (err, rows) => {
    if (err) throw err;
    res.render('form', { user: rows[0] });
  });
};
```

```
exports.update = (req, res) => {
  const id = req.params.id;
  const { nombre, correo } = req.body;
  User.update(id, { nombre, correo }, err => {
    if (err) throw err;
    res.redirect('/');
  });
};

exports.delete = (req, res) => {
  const id = req.params.id;
  User.delete(id, err => {
    if (err) throw err;
    res.redirect('/');
  });
};
```

Paso 6: Crear las rutas

routes/userRoutes.js

```
const express = require('express');
const router = express.Router();
const userController = require('../controllers/userController');

router.get('/', userController.index);
router.get('/nuevo', userController.form);
router.post('/crear', userController.create);
router.get('/editar/:id', userController.edit);
router.post('/actualizar/:id', userController.update);
router.get('/eliminar/:id', userController.delete);

module.exports = router;
```

Paso 7: Crear las vistas HTML con EJS

views/index.ejs

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>Usuarios</title>
</head>
<body>
 <h1>Lista de Usuarios</h1>
 <a href="/nuevo">Nuevo Usuario</a>
 <table border="1">
  <tr>
   <th>ID</th><th>Nombre</th><th>Correo</th><th>Acciones</th>
  </tr>
  <% users.forEach(user => { %>
   <tr>
    <td><%= user.id %></td>
    <td><%= user.nombre %></td>
    <td><%= user.correo %></td>
    <td>
     <a href="/editar/<%= user.id %>">Editar</a>
     <a href="/eliminar/<%= user.id %>" onclick="return
confirm('¿Eliminar?')">Eliminar</a>
    </td>
   </tr>
  <% }) %>
 </table>
</body>
</html>
```

views/form.ejs

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title><%= user ? 'Editar' : 'Nuevo' %> Usuario</title>
</head>
<body>
  <h1><%= user ? 'Editar' : 'Nuevo' %> Usuario</h1>
  <form method="POST" action="<%= user ? '/actualizar/' + user.id : '/crear' %>">
    <input type="text" name="nombre" placeholder="Nombre" value="<%= user ? user.nombre : '' %>" required><br><br>
    <input type="email" name="correo" placeholder="Correo" value="<%= user ? user.correo : '' %>" required><br><br>
    <button type="submit">Guardar</button>
  </form>
</body>
</html>
```

Paso 8: Crear el archivo principal

app.js

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const userRoutes = require('./routes/userRoutes');

app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static('public'));

app.use('/', userRoutes);

app.listen(3000, () => {
```

```
  console.log('Servidor ejecutándose en http://localhost:3000');
});
```

Paso 9: Ejecutar la aplicación

node app.js

Y abre tu navegador en: http://localhost:3000

Resultado final

- Puedes **crear**, **listar**, **editar** y **eliminar** usuarios.
- El sistema está **conectado a MySQL**.
- Sigue el **modelo MVC**.
- Tiene una **interfaz web sencilla pero funcional**.

Actividad propuesta: Crear la estructura base para tu proyecto final y subir al menos un avance en lo que respecta a backend.