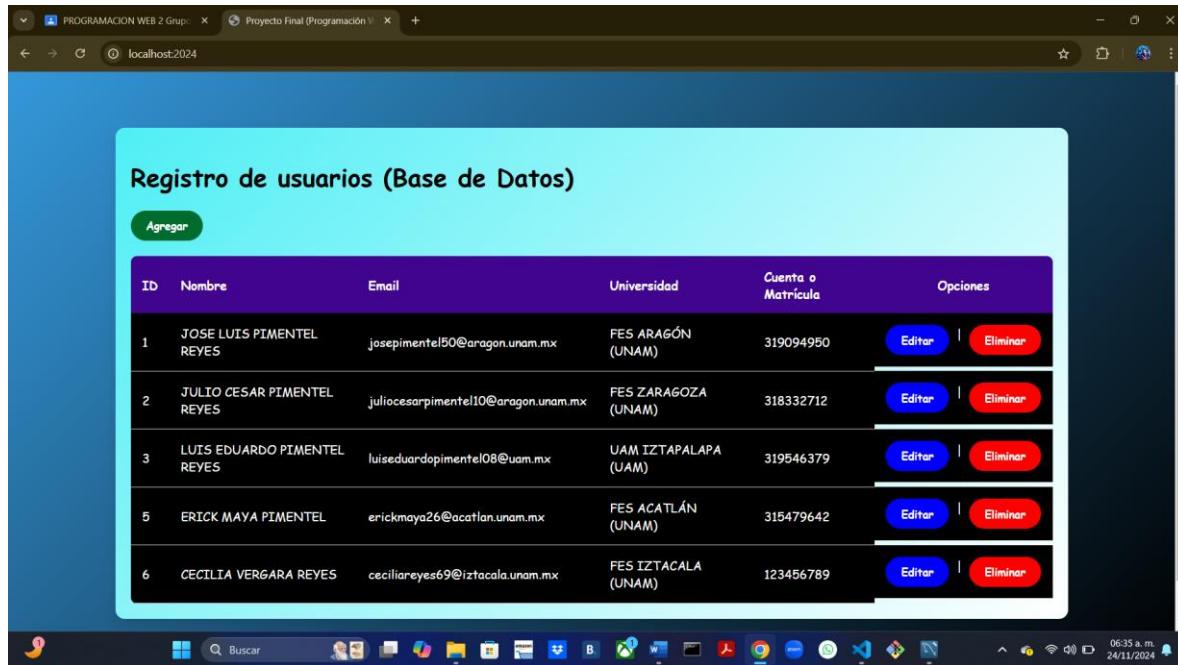
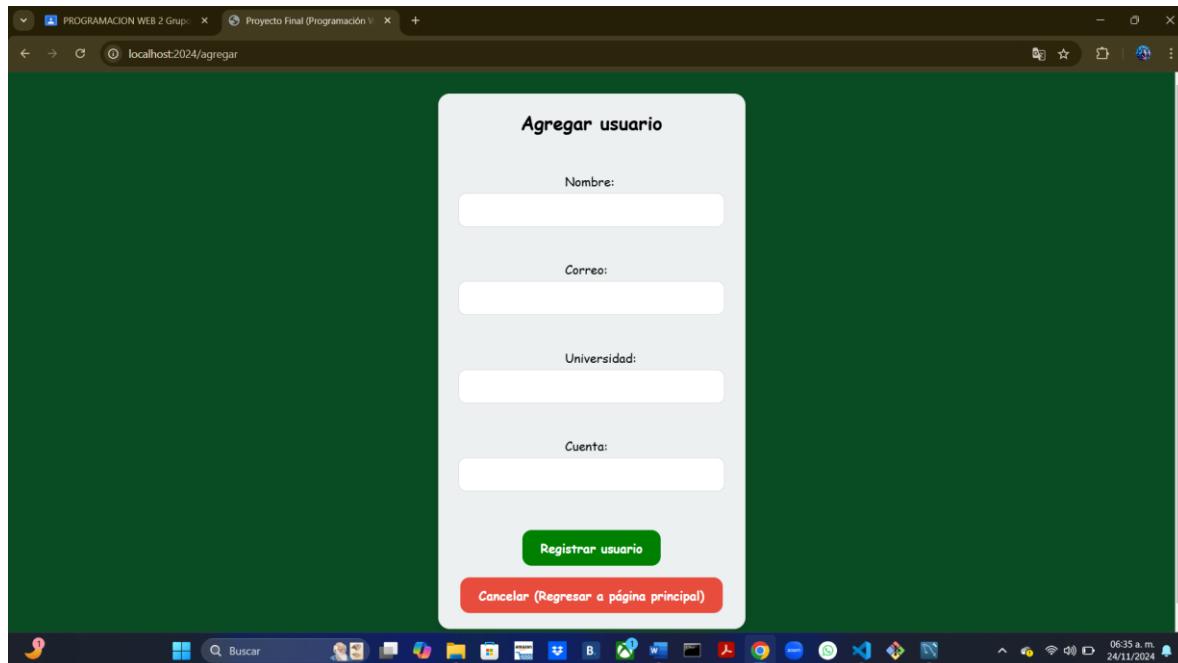


Visualización de mi Registro de Usuarios (CRUD).



ID	Nombre	Email	Universidad	Cuenta o Matrícula	Opciones
1	JOSE LUIS PIMENTEL REYES	josepimentel50@aragon.unam.mx	FES ARAÑÓN (UNAM)	319094950	<a href="#">Editor</a>   <a href="#">Eliminar</a>
2	JULIO CESAR PIMENTEL REYES	juliocesarpimentel10@aragon.unam.mx	FES ZARAGOZA (UNAM)	318332712	<a href="#">Editor</a>   <a href="#">Eliminar</a>
3	LUIS EDUARDO PIMENTEL REYES	luiseduardopimentel08@uam.mx	UAM IZTAPALAPA (UAM)	319546379	<a href="#">Editor</a>   <a href="#">Eliminar</a>
5	ERICK MAYA PIMENTEL	erickmaya26@acatlan.unam.mx	FES ACATLÁN (UNAM)	315479642	<a href="#">Editor</a>   <a href="#">Eliminar</a>
6	CECILIA VERGARA REYES	ceciliareyes69@iztacala.unam.mx	FES IZTACALA (UNAM)	123456789	<a href="#">Editor</a>   <a href="#">Eliminar</a>

Insertar o agregar un nuevo usuario a la tabla de usuarios de mi Base de Datos.



Agregar usuario

Nombre:

Correo:

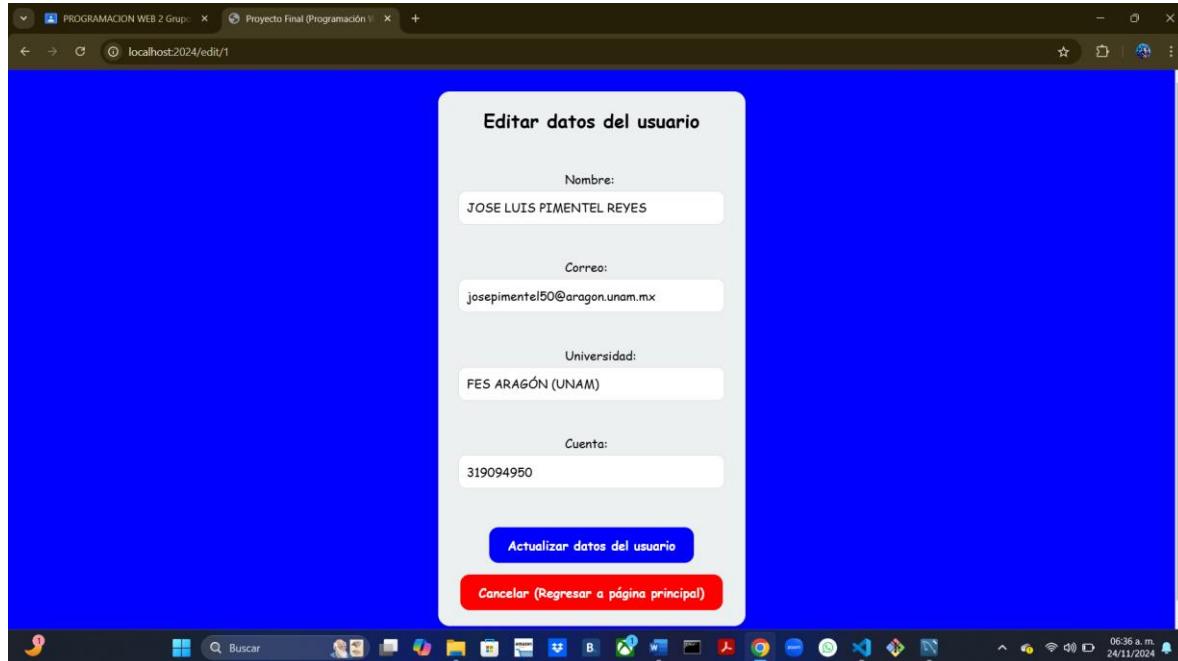
Universidad:

Cuenta:

[Registrar usuario](#)

[Cancelar \(Regresar a página principal\)](#)

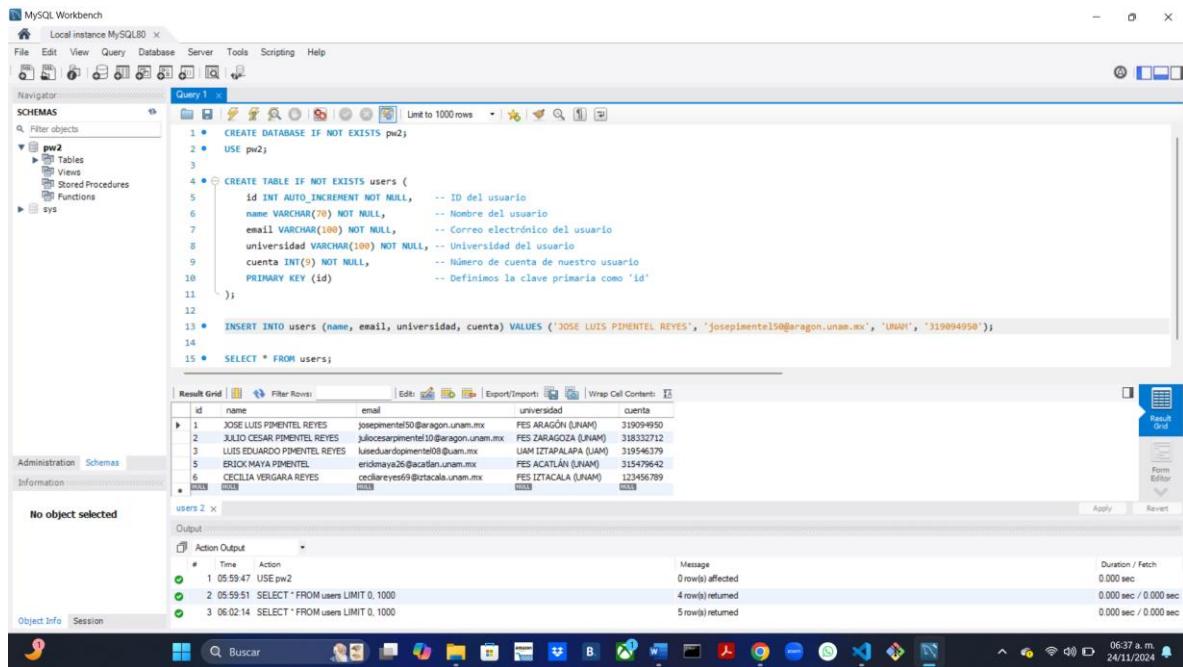
Interfaz para poder editar datos del usuario y mostrarlos en la tabla de usuarios de mi Base de Datos.



Eliminación de usuario a través del botón Eliminar.

Registro de usuarios (Base de Datos)					
ID	Nombre	Email	Universidad	Cuenta o Matrícula	Opciones
1	JOSE LUIS PIMENTEL REYES	josepimentel50@aragon.unam.mx	FES ARAGÓN (UNAM)	319094950	<a href="#">Editor</a>   <a href="#">Eliminar</a>
2	JULIO CESAR PIMENTEL REYES	juliocesar.pimentel10@aragon.unam.mx	FES ZARAGOZA (UNAM)	318332712	<a href="#">Editor</a>   <a href="#">Eliminar</a>
3	LUIS EDUARDO PIMENTEL REYES	luiseduardopimentel08@uam.mx	UAM IZTAPALAPA (UAM)	319546379	<a href="#">Editor</a>   <a href="#">Eliminar</a>
5	ERICK MAYA PIMENTEL	erickmaya26@acatlan.unam.mx	FES ACATLÁN (UNAM)	315479642	<a href="#">Editor</a>   <a href="#">Eliminar</a>

Observación de los usuarios registrados antes de eliminar al último usuario de la Base de Datos (Vista desde Workbench para comprobar la conexión de la Base de Datos).



```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
Filter objects
Schemas
pw2
Tables
Views
Stored Procedures
Functions
sys
Query 1
CREATE DATABASE IF NOT EXISTS pw2;
USE pw2;
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT NOT NULL, -- ID del usuario
    name VARCHAR(70) NOT NULL, -- Nombre del usuario
    email VARCHAR(100) NOT NULL, -- Correo electrónico del usuario
    universidad VARCHAR(100) NOT NULL, -- Universidad del usuario
    cuenta INT(9) NOT NULL, -- Número de cuenta de nuestro usuario
    PRIMARY KEY (id) -- Definimos la clave primaria como 'id'
);
INSERT INTO users (name, email, universidad, cuenta) VALUES ('JOSE LUIS PIMENTEL REYES', 'josepimentel50@aragon.unam.mx', 'UNAM', '319094950');
SELECT * FROM users;

```

Result Grid | Filter Rows | Edit | Export/Import | Wrap Cell Content:

	id	name	email	universidad	cuenta
▶	1	JOSE LUIS PIMENTEL REYES	josepimentel50@aragon.unam.mx	FES ARAGÓN (UNAM)	319094950
▶	2	JULIO CESAR PIMENTEL REYES	julioscesarpimentel10@aragon.unam.mx	FES ZARAGOZA (UNAM)	318332712
▶	3	LUIS EDUARDO PIMENTEL REYES	luiseduardopimentel08@unam.mx	UAM IZTAPALAPA (UAM)	319546379
▶	4	ERICK MAYA PIMENTEL	erickmaya26@ecatlan.unam.mx	FES ACATLÁN (UNAM)	315479642
▶	5	CECILIA VARGERA REYES	ceciliareyes69@tzascal.unam.mx	FES IZTACALCA (UNAM)	123456789
▶	6				
▶	7				
▶	8				
▶	9				
▶	10				
▶	11				
▶	12				
▶	13				
▶	14				
▶	15				

users 2 x

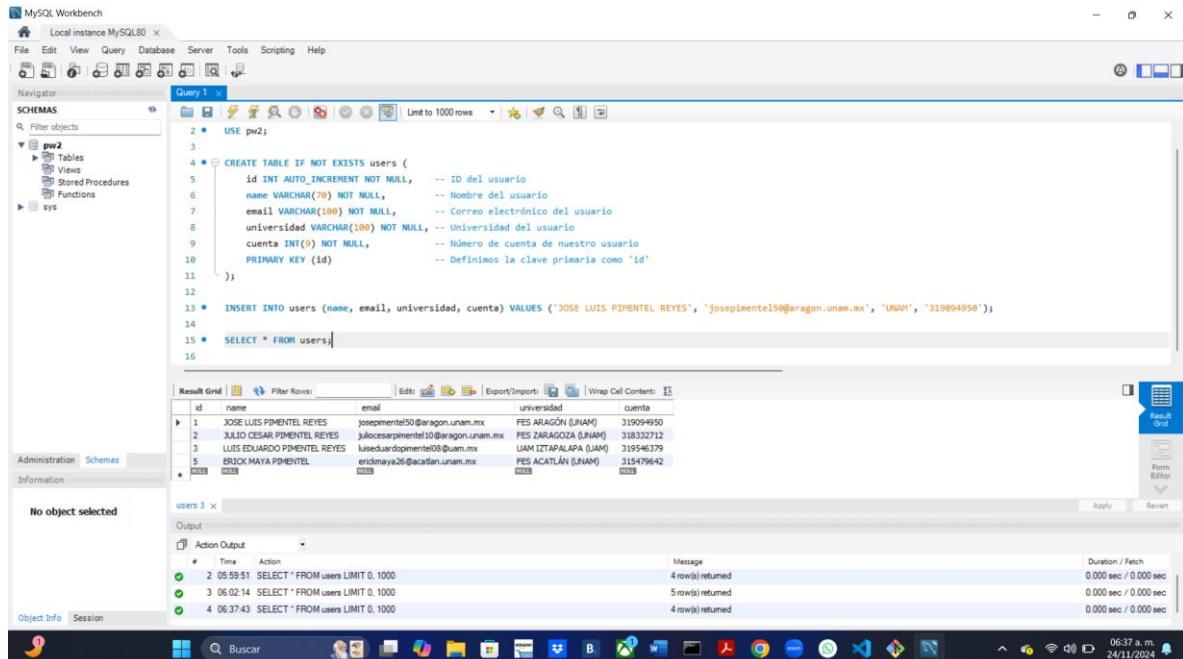
Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	05:59:47	USE pw2	0 row(s) affected	0.000 sec
2	05:59:51	SELECT * FROM users LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
3	06:02:14	SELECT * FROM users LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Observación de los usuarios registrados después de eliminar al último usuario de la Base de Datos (Vista desde Workbench para comprobar la conexión de la Base de Datos).



```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
Filter objects
Schemas
pw2
Tables
Views
Stored Procedures
Functions
sys
Query 1
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT NOT NULL, -- ID del usuario
    name VARCHAR(70) NOT NULL, -- Nombre del usuario
    email VARCHAR(100) NOT NULL, -- Correo electrónico del usuario
    universidad VARCHAR(100) NOT NULL, -- Universidad del usuario
    cuenta INT(9) NOT NULL, -- Número de cuenta de nuestro usuario
    PRIMARY KEY (id) -- Definimos la clave primaria como 'id'
);
INSERT INTO users (name, email, universidad, cuenta) VALUES ('JOSE LUIS PIMENTEL REYES', 'josepimentel50@aragon.unam.mx', 'UNAM', '319094950');
SELECT * FROM users;

```

Result Grid | Filter Rows | Edit | Export/Import | Wrap Cell Content:

	id	name	email	universidad	cuenta
▶	1	JOSE LUIS PIMENTEL REYES	josepimentel50@aragon.unam.mx	FES ARAGÓN (UNAM)	319094950
▶	2	JULIO CESAR PIMENTEL REYES	julioscesarpimentel10@aragon.unam.mx	FES ZARAGOZA (UNAM)	318332712
▶	3	LUIS EDUARDO PIMENTEL REYES	luiseduardopimentel08@unam.mx	UAM IZTAPALAPA (UAM)	319546379
▶	4	ERICK MAYA PIMENTEL	erickmaya26@ecatlan.unam.mx	FES ACATLÁN (UNAM)	315479642
▶	5				
▶	6				
▶	7				
▶	8				
▶	9				
▶	10				
▶	11				
▶	12				
▶	13				
▶	14				
▶	15				

users 3 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	05:59:51	SELECT * FROM users LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
3	06:02:14	SELECT * FROM users LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
4	06:37:43	SELECT * FROM users LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Código SQL para crear la Base de Datos pw2 en referencia a la materia Programación Web 2, así mismo, se crea la tabla de usuarios que llevará el registro y las acciones que realicen los usuarios (agregar, editar y eliminar).

The screenshot shows a Windows desktop environment. In the center is a Visual Studio Code window titled 'Proyecto Final'. The left sidebar shows a project structure with files like 'bd.sql', 'app.js', 'index.ejs', 'add.ejs', 'edit.ejs', 'agregar.css', 'editar.css', 'estilos.css', 'particulars.js', 'pages.js', and 'views'. The main editor area contains the following SQL code:

```
CREATE DATABASE IF NOT EXISTS pw2; -- pw2 hace referencia a la materia (Programación Web 2)
USE pw2;
-- Creamos tabla 'users'
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT NOT NULL,
    name VARCHAR(70) NOT NULL,
    email VARCHAR(100) NOT NULL,
    universidad VARCHAR(100) NOT NULL,
    cuenta INT(9) NOT NULL,
    PRIMARY KEY (id)
);
-- Insertamos un usuario
INSERT INTO users (name, email, universidad, cuenta) VALUES ('JOSE LUIS PIMENTEL REYES', 'josepimentel50@aragon.unam.mx', 'UNAM', '319094950');
```

Below the code editor is a terminal window titled 'Historial restaurado' showing the command 'Node app.js' being run. The output indicates a successful connection to the database.

Este código en **Node.js** utiliza el framework **Express.js** para configurar un enrutador que define varias rutas específicas del servidor.

Se importa el módulo **express** y se crea una instancia de un enrutador con **express.Router()**. Además, se utiliza **path** para manejar rutas de archivos de forma compatible con diferentes sistemas operativos.

1. **Rutas para renderizar vistas:** La **ruta/edit** renderiza la vista **edit.ejs** ubicada en la carpeta **views**. La **ruta/agregar** renderiza la vista **add.ejs** en la misma carpeta. Esto permite mostrar páginas dinámicas generadas en el servidor al cliente.
2. **Rutas para servir archivos CSS:** Las rutas **/styles**, **/estilosagregar** y **/estiloseditar** envían al cliente los archivos CSS (**estilos.css**, **estilosadd.css**, **estilosedit.css**) almacenados en la carpeta **public/css**. Esto permite aplicar estilos específicos a las vistas en función de la página solicitada.
3. **Exportación del enrutador:** Finalmente, el enrutador se exporta con **module.exports = router** para que pueda ser integrado en el servidor principal.

En resumen, este código organiza las rutas del servidor para gestionar tanto vistas dinámicas como la entrega de recursos estáticos de manera eficiente.

The screenshot shows the Visual Studio Code interface with the following details:

- Explorador (File Explorer):** Shows the project structure under "PROYECTO FINAL". It includes files like app.js, bd.sql, agregar.css, editar.css, styles.css, routes.js, pages.js, and various CSS and JS files for views.
- Code Editor:** Displays the content of the pages.js file, which contains Express.js route definitions for creating, reading, updating, and deleting users from a MySQL database.
- Terminal:** Shows the command "node app.js" being run, and the output indicates a successful connection to the MySQL database at "localhost:2024".
- Bottom Status Bar:** Provides information such as the current file (app.js), line count (Lín. 26), column count (col. 1), encoding (UTF-8), and date (24/11/2024).

Este código es una aplicación en Node.js con Express.js que implementa un sistema CRUD (Crear, Leer, Actualizar y Eliminar) para gestionar usuarios desde una base de datos MySQL.

Las funcionalidades principales incluyen:

- Configuración del servidor: Usa express, mysql2, y body-parser (los cuales son requisitos indispensables establecidos previamente para el desarrollo de este proyecto).
- Conexión a mi Base de Datos.
- Operaciones CRUD:
  - Leer: Obtiene todos los usuarios de la tabla users y los muestra en una vista llamada index.ejs.
  - Crear: Agrega un nuevo usuario verificando que el campo "cuenta" tenga 9 dígitos.
  - Actualizar: Muestra un formulario de edición para un usuario específico y actualiza su información en la base de datos.
  - Eliminar: Borra un usuario específico basado en su ID.
- Inicio del servidor: Ejecuta el servidor en el puerto 2024 (referente al año presente) y muestra un mensaje de confirmación en la consola si es que se logra la conexión; si no es el caso se muestra un mensaje de error en la conexión.

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const pageRoutes = require('./routes/pages');

const app = express();
const port = 2024;
app.use('/', pageRoutes);
app.use(express.static(path.join(__dirname, 'public')));

app.use(bodyParser.urlencoded({ extended: false }));
app.set('view engine', 'ejs');

// Conexión a la BD
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '123456',
  database: 'pw2',
  port: '3306'
});

db.connect((err) => {
  if(err)
    console.log('No se pudo realizar la conexión a la Base de Datos');
});
```

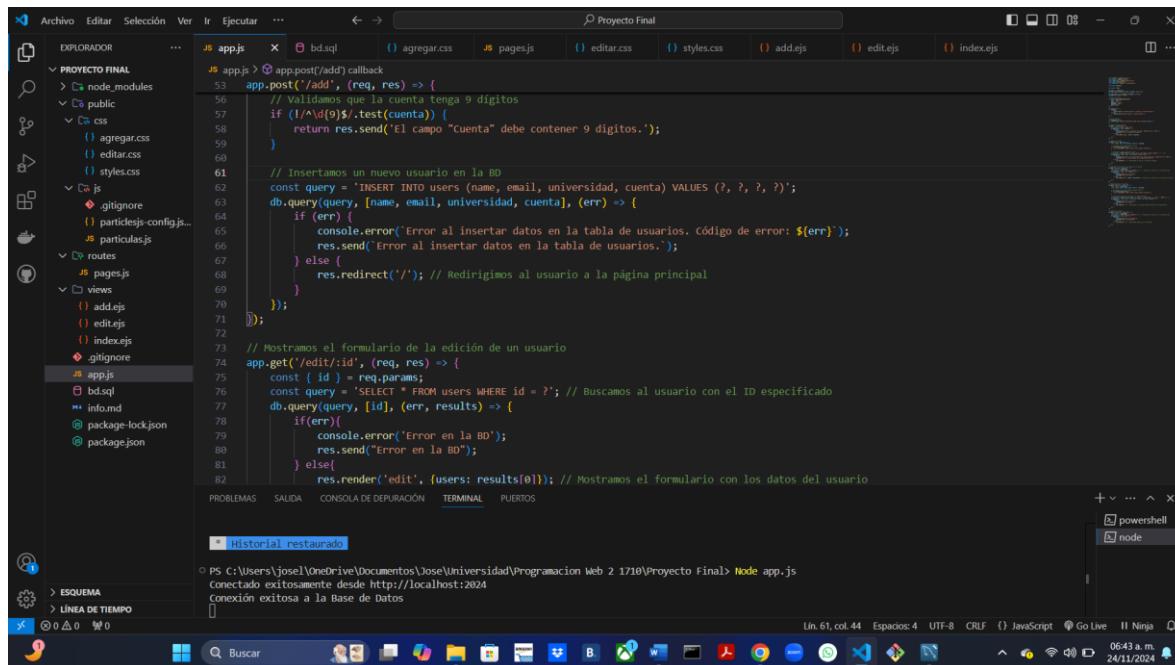
PS C:\Users\jose\OneDrive\Documentos\Jose\Universidad\Programación Web 2 1710\Proyecto Final> Node app.js  
Conectado exitosamente desde http://localhost:2024  
Conexión exitosa a la Base de Datos

```
db.connect((err) => {
  if(err)
    console.log('Conexión exitosa a la Base de Datos');
  else{
    console.log('Inicia servidor');
    app.listen(port, ()=>{
      console.log('Conectado exitosamente desde http://localhost:' + port);
    });
  }
});
```

```
// Mostrar tabla de usuarios
app.get('/', (req, res) => {
  const query = 'SELECT * FROM users';
  db.query(query, (err, results) => {
    if(err){
      console.error('Error al recuperar los datos. Código de error: ' + err);
      res.send('Error al recuperar los datos');
    } else{
      res.render('index', {users: results});
    }
  });
});
```

```
// Agregar un usuario
app.post('/add', (req, res) => {
  const { name, email, universidad, cuenta } = req.body;
```

PS C:\Users\jose\OneDrive\Documentos\Jose\Universidad\Programación Web 2 1710\Proyecto Final> Node app.js  
Conectado exitosamente desde http://localhost:2024  
Conexión exitosa a la Base de Datos



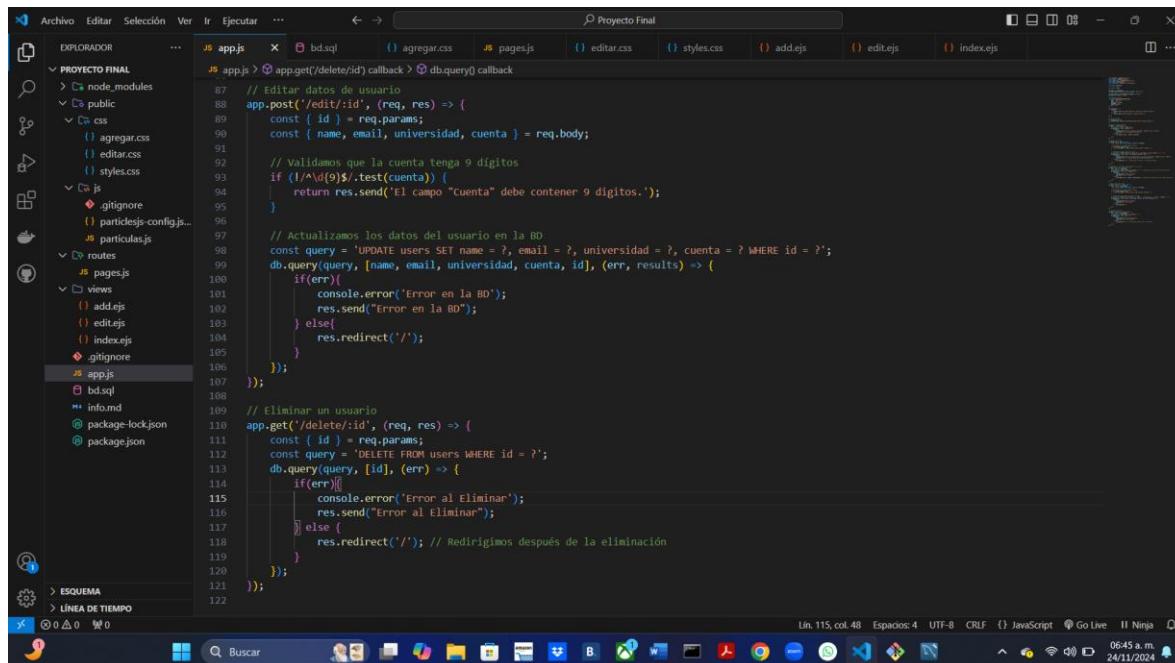
```

    app.post('/add', (req, res) => {
        // Validamos que la cuenta tenga 9 dígitos
        if (!/^d{9}$/.test(cuenta)) {
            return res.send('El campo "Cuenta" debe contener 9 dígitos.');
        }

        // Insertamos un nuevo usuario en la BD
        const query = 'INSERT INTO users (name, email, universidad, cuenta) VALUES (?, ?, ?, ?)';
        db.query(query, [name, email, universidad, cuenta], (err) => {
            if (err) {
                console.error('Error al insertar datos en la tabla de usuarios. Código de error: ${err}');
                res.send('Error al insertar datos en la tabla de usuarios.');
            } else {
                res.redirect('/');
            }
        });
    });

    // Mostramos el formulario de la edición de un usuario
    app.get('/edit/:id', (req, res) => {
        const { id } = req.params;
        const query = 'SELECT * FROM users WHERE id = ?'; // Buscamos al usuario con el ID especificado
        db.query(query, [id], (err, results) => {
            if (err) {
                console.error('Error en la BD');
                res.send('Error en la BD');
            } else {
                res.render('edit', { users: results[0] }); // Mostramos el formulario con los datos del usuario
            }
        });
    });
}

```



```

    app.post('/edit/:id', (req, res) => {
        const { id } = req.params;
        const { name, email, universidad, cuenta } = req.body;

        // Validamos que la cuenta tenga 9 dígitos
        if (!/^d{9}$/.test(cuenta)) {
            return res.send('El campo "Cuenta" debe contener 9 dígitos.');
        }

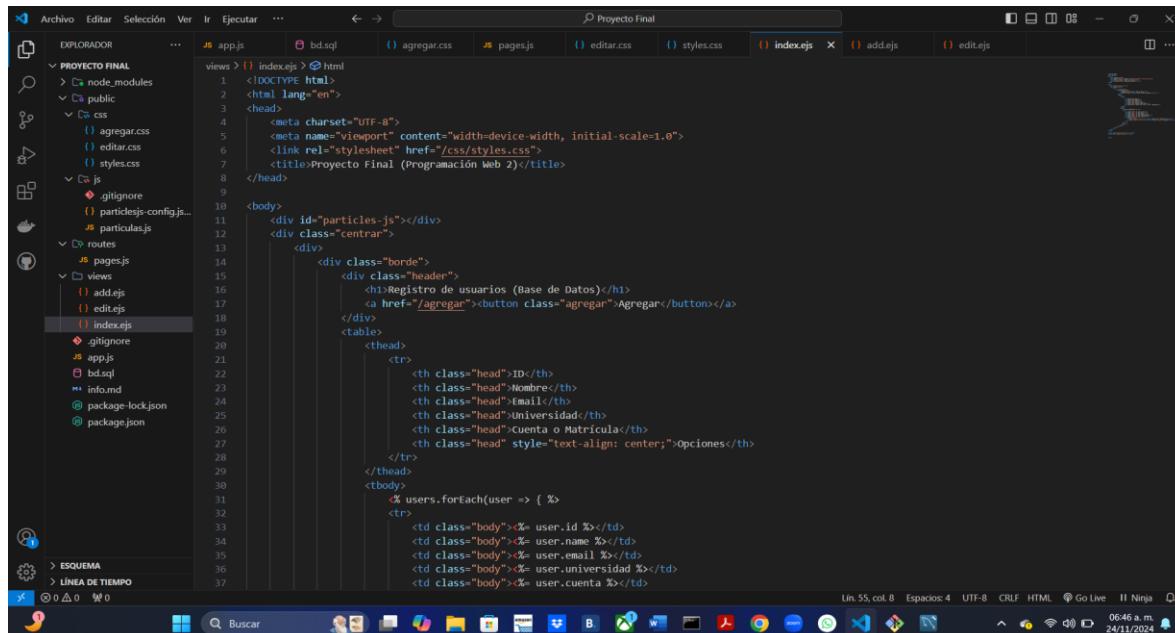
        // Actualizamos los datos del usuario en la BD
        const query = 'UPDATE users SET name = ?, email = ?, universidad = ?, cuenta = ? WHERE id = ?';
        db.query(query, [name, email, universidad, cuenta, id], (err, results) => {
            if (err) {
                console.error('Error en la BD');
                res.send('Error en la BD');
            } else {
                res.redirect('/');
            }
        });
    });

    // Eliminar un usuario
    app.get('/delete/:id', (req, res) => {
        const { id } = req.params;
        const query = 'DELETE FROM users WHERE id = ?';
        db.query(query, [id], (err) => {
            if (err) {
                console.error('Error al Eliminar');
                res.send('Error al Eliminar');
            } else {
                res.redirect('/');
            }
        });
    });
}

```

Este código HTML dinámico con EJS muestra un registro de usuarios conectado a una base de datos.  
 Incluye:

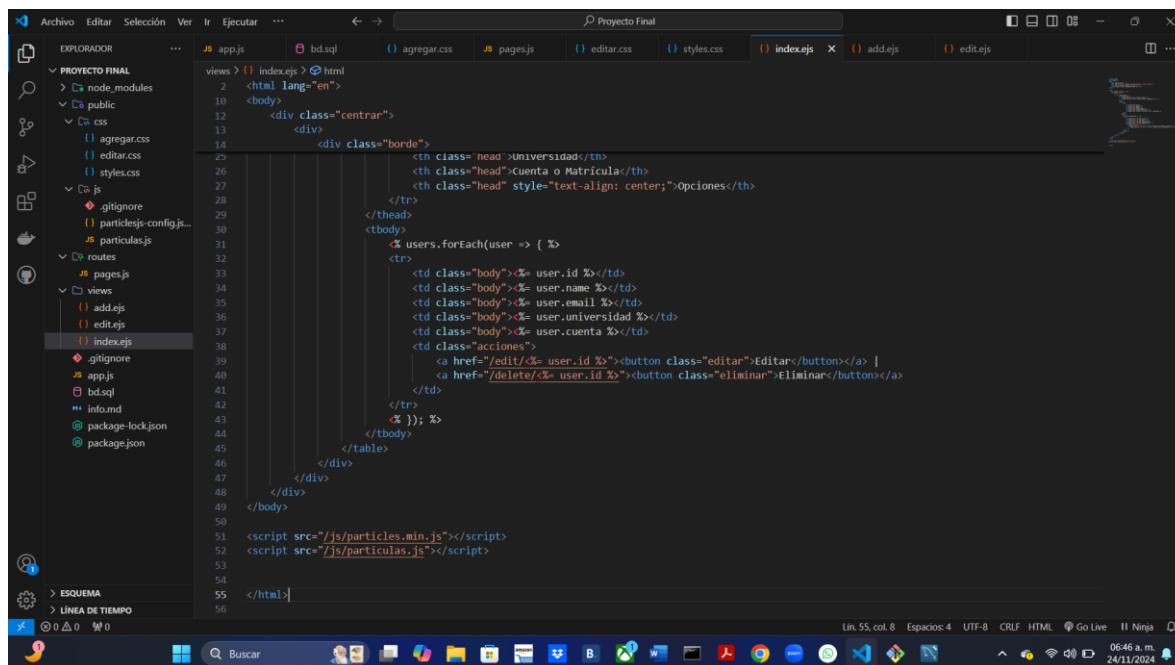
- Botón para **agregar** usuarios (agregar).
- Una tabla que lista usuarios (ID, Nombre, Email, Universidad, Cuenta) con datos generados dinámicamente.
- Botones para **editar** (edit) y **eliminar** (delete) cada usuario.
- Scripts (particles.js) para animar el fondo.



```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/styles.css">
    <title>Proyecto Final (Programación Web 2)</title>
</head>
<body>
    <div id="particulas-js"></div>
    <div class="centrar">
        <div class="borde">
            <div class="header">
                <h1>Registro de usuarios (Base de Datos)</h1>
                <a href="/agregar"><button class="agregar">Añadir</button></a>
            </div>
            <table>
                <thead>
                    <tr>
                        <th class="head">ID</th>
                        <th class="head">Nombre</th>
                        <th class="head">Email</th>
                        <th class="head">Universidad</th>
                        <th class="head">Cuenta o Matrícula</th>
                        <th class="head" style="text-align: center;">Opciones</th>
                    </tr>
                </thead>
                <tbody>
                    <% users.forEach(user => { %>
                    <tr>
                        <td class="body"><% user.id %></td>
                        <td class="body"><% user.name %></td>
                        <td class="body"><% user.email %></td>
                        <td class="body"><% user.universidad %></td>
                        <td class="body"><% user.cuenta %></td>
                    </tr>
                    <% }); %>
                </tbody>
            </table>
        </div>
    </div>
</body>

```



```

<!DOCTYPE html>
<html lang="en">
<body>
    <div class="centrar">
        <div class="borde">
            <thead>
                <tr>
                    <th class="head">Universidad</th>
                    <th class="head">Cuenta o Matrícula</th>
                    <th class="head" style="text-align: center;">Opciones</th>
                </tr>
            </thead>
            <tbody>
                <% users.forEach(user => { %>
                <tr>
                    <td class="body"><% user.id %></td>
                    <td class="body"><% user.name %></td>
                    <td class="body"><% user.email %></td>
                    <td class="body"><% user.universidad %></td>
                    <td class="body"><% user.cuenta %></td>
                    <td class="acciones">
                        <a href="/edit/<% user.id %>"><button class="editar">Editar</button></a> | 
                        <a href="/delete/<% user.id %>"><button class="eliminar">Eliminar</button></a>
                    </td>
                </tr>
                <% }); %>
            </tbody>
        </div>
    </div>
</body>
<script src="/js/particulas.min.js"></script>
<script src="/js/particulas.js"></script>

```

### Este código agrega usuarios a la Base de Datos.

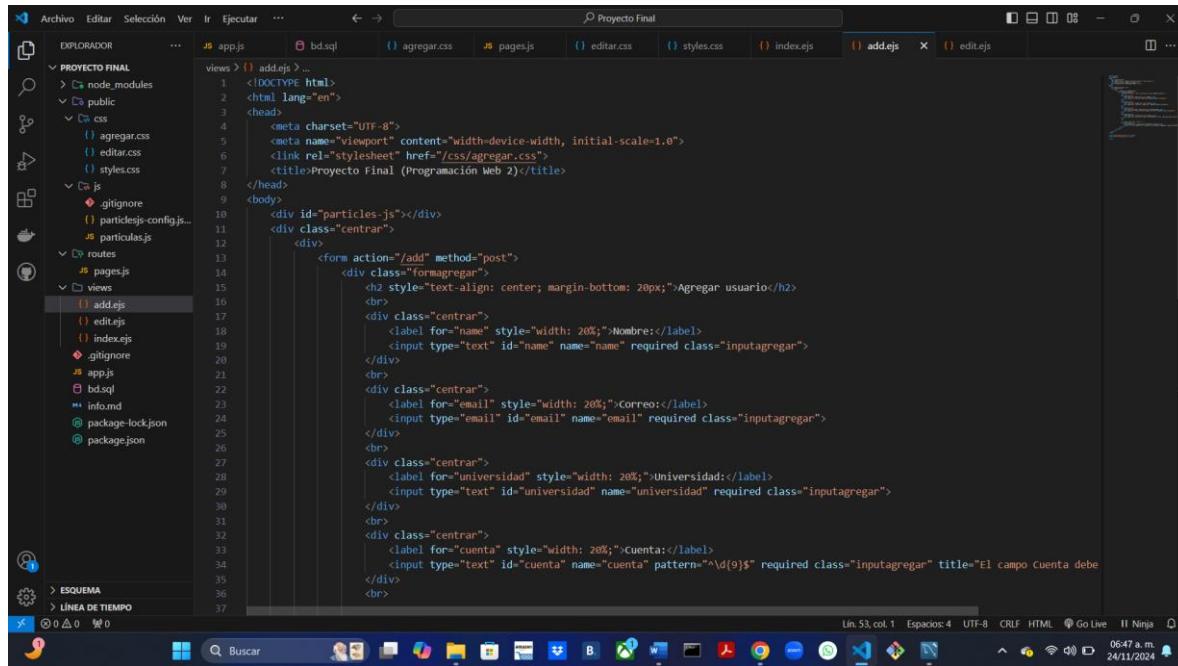
Permite enviar un formulario con datos del usuario (nombre, correo, universidad y cuenta) al servidor mediante una solicitud POST al endpoint /add. La cuenta debe contener exactamente 9 dígitos (validado con pattern).

### Componentes principales:

- Formulario de usuario: Campos de entrada para nombre, correo, universidad y cuenta con validación básica.
- Botones: Uno para enviar los datos al servidor y otro para cancelar (redirige a la página principal).

### Estilo y scripts:

- Archivo CSS (agregar.css) para el diseño.
- Animaciones de fondo con particles.js.



The screenshot shows a Windows desktop environment. On the left, there is a file explorer window titled "PROYECTO FINAL" containing various project files: app.js, bd.sql, agregar.css, pages.js, editar.css, styles.css, index.ejs, add.ejs, edit.ejs, .gitignore, particlesjs-config.js, and particlesjs.js. The "views" folder is expanded, showing add.ejs, edit.ejs, index.ejs, and .gitignore. The "routes" folder is also expanded, showing pages.js. In the center, there is a code editor window titled "add.ejs" which contains the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/agregar.css">
    <title>Proyecto Final (Programación Web 2)</title>
</head>
<body>
    <div id="particles-js"></div>
    <div class="centrar">
        <form action="/add" method="post">
            <div class="formagregar">
                <h2 style="text-align: center; margin-bottom: 20px;">Agregar usuario</h2>
                <br>
                <div class="centrar">
                    <label for="name" style="width: 20%;>Nombre:</label>
                    <input type="text" id="name" name="name" required class="inputagregar">
                </div>
                <br>
                <div class="centrar">
                    <label for="email" style="width: 20%;>Correo:</label>
                    <input type="email" id="email" name="email" required class="inputagregar">
                </div>
                <br>
                <div class="centrar">
                    <label for="universidad" style="width: 20%;>Universidad:</label>
                    <input type="text" id="universidad" name="universidad" required class="inputagregar">
                </div>
                <br>
                <div class="centrar">
                    <label for="cuenta" style="width: 20%;>Cuenta:</label>
                    <input type="text" id="cuenta" name="cuenta" pattern="\d{9}" required class="inputagregar" title="El campo Cuenta debe contener 9 dígitos.">
                </div>
            <br>
        </div>
    </form>
</div>
```

The screenshot shows a code editor window titled "Proyecto Final". The left sidebar displays a project structure with files like app.js, bd.sql, agregar.css, editar.css, styles.css, index.ejs, add.ejs, edit.ejs, .gitignore, particulas.js, routes, pages.js, and views. The main pane shows the content of the "add.ejs" file:

```
<html lang="en">
<body>
    <div class="centrar">
        <form action="/add" method="post">
            <div>
                <br>
                <div class="centrar">
                    <label for="cuenta" style="width: 20%;>xCuenta:</label>
                    <input type="text" id="cuenta" name="cuenta" pattern="\d{9}" required class="inputagregar" title="El campo Cuenta debe de tener 9 dígitos.">
                </div>
                <br>
                <div class="botones">
                    <button type="submit" class="agregar">Registrar usuario</button>
                    <a href="/"><button type="button" class="eliminar">Cancelar (regresar a la página principal)</button></a>
                </div>
            </div>
        </form>
    </div>
</body>
</html>
```

Below the code editor is a taskbar with various icons, and the system tray shows the date and time as 24/11/2024 06:47 a.m.

Este código **edita los datos de un usuario** en la Base de Datos.

Básicamente modifica los datos de un usuario específico mediante un formulario que realiza una solicitud POST al **endpoint /edit/:id**, donde **:id** es el identificador del usuario. Los campos incluyen validaciones, como que el número de cuenta debe tener exactamente 9 dígitos.

#### Componentes principales:

- Formulario de edición:
- Campos pre-rellenados con la información del usuario (name, email, universidad, cuenta) utilizando EJS para pasar datos desde el servidor.
- Validaciones incluidas (pattern) para asegurar que los datos sean correctos.

#### Botones:

- Actualizar: Envía los datos actualizados al servidor.
- Cancelar: Redirige a la página principal sin realizar cambios.

#### Estilo y scripts:

- Archivo CSS (editar.css) para personalizar el diseño de la página.
- Animaciones de fondo con particulas.js.

The screenshot shows a code editor window titled "Proyecto Final". The left sidebar displays a project structure under "EXPLORADOR" for a "PROYECTO FINAL" named "Programación Web 2". The "views" folder contains "edit.ejs", which is currently selected and shown in the main editor area. The code in "edit.ejs" is an HTML form for editing user data, including fields for name, email, and university, with validation patterns and placeholder text. The status bar at the bottom indicates "Lin. 37, col. 8 Espacios: 2 UTF-8 CRLF HTML Go Live II Ninja".

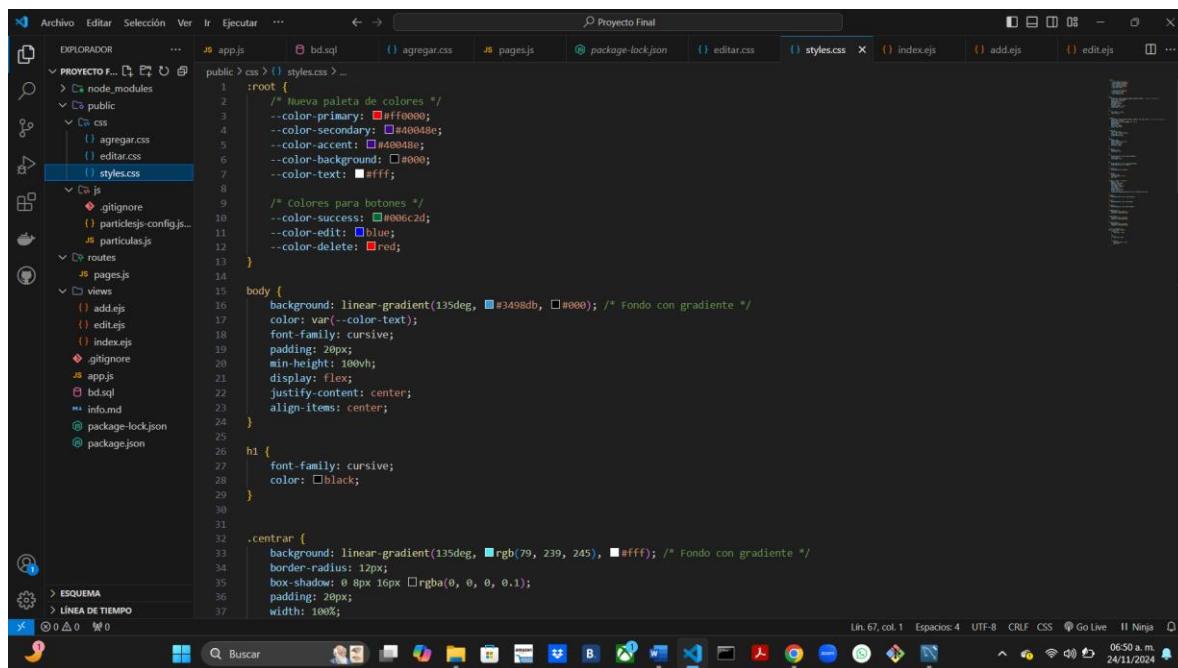
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/editar.css">
    <title>Proyecto Final (Programación Web 2)</title>
</head>
<body>
    <div id="particles-js"></div>
    <div class="centrar">
        <div>
            <form action="/edit/<% users.id%>" method="post">
                <div class="formeditar">
                    <h2 style="text-align: center; margin-bottom: 20px;">Editar datos del usuario</h2>
                    <br>
                    <div class="centrar">
                        <label for="name" style="width: 20%;">Nombre:</label>
                        <input type="text" id="name" name="name" required class="inputeditar" value="<%= users.name %>">
                    </div>
                    <br>
                    <div class="centrar">
                        <label for="email" style="width: 20%;">Correo:</label>
                        <input type="email" id="email" name="email" required class="inputeditar" value="<%= users.email %>">
                    </div>
                    <br>
                    <div class="centrar">
                        <label for="universidad" style="width: 20%;">Universidad:</label>
                        <input type="text" id="universidad" name="universidad" required class="inputeditar" value="<%= users.universidad %>">
                    </div>
                    <br>
                    <div class="centrar">
                        <label for="cuenta" style="width: 20%;">Cuenta:</label>
                        <input type="text" id="cuenta" name="cuenta" required class="inputeditar" value="<%= users.cuenta %>" pattern="\d{9}" title="El campo Cuenta debe contener 9 dígitos">
                    </div>
                </div>
                <div class="centrar">
                    <button type="submit" class="editar">Actualizar datos del usuario</button>
                    <a href="/"><button type="button" class="eliminar">Cancelar (Regresar a la página principal)</button></a>
                </div>
            </form>
        </div>
    </div>
    <script src="/js/particulas.js"></script>
    <script src="/js/particulas.js"></script>
</body>
</html>
```

This screenshot shows the same code editor window as the previous one, but the "edit.ejs" file now contains more code, specifically adding a "botones" section with a "cancelar" button. The status bar at the bottom indicates "Lin. 54, col. 1 Espacios: 2 UTF-8 CRLF HTML Go Live II Ninja".

```
<div class="centrar">
    <div>
        <form action="/edit/<% users.id%>" method="post">
            <div class="formeditar">
                <h2 style="text-align: center; margin-bottom: 20px;">Editar datos del usuario</h2>
                <br>
                <div class="centrar">
                    <label for="name" style="width: 20%;">Nombre:</label>
                    <input type="text" id="name" name="name" required class="inputeditar" value="<%= users.name %>">
                </div>
                <br>
                <div class="centrar">
                    <label for="email" style="width: 20%;">Correo:</label>
                    <input type="email" id="email" name="email" required class="inputeditar" value="<%= users.email %>">
                </div>
                <br>
                <div class="centrar">
                    <label for="universidad" style="width: 20%;">Universidad:</label>
                    <input type="text" id="universidad" name="universidad" required class="inputeditar" value="<%= users.universidad %>">
                </div>
                <br>
                <div class="centrar">
                    <label for="cuenta" style="width: 20%;">Cuenta:</label>
                    <input type="text" id="cuenta" name="cuenta" required class="inputeditar" value="<%= users.cuenta %>" pattern="\d{9}" title="El campo Cuenta debe contener 9 dígitos">
                </div>
                <br>
                <div class="botones">
                    <button type="submit" class="editar">Actualizar datos del usuario</button>
                    <a href="/"><button type="button" class="eliminar">Cancelar (Regresar a la página principal)</button></a>
                </div>
            </div>
        </div>
    </div>
    <script src="/js/particules.min.js"></script>
    <script src="/js/particulas.js"></script>
</body>
</html>
```

La hoja de CSS styles define animaciones y transiciones que mejoran la experiencia visual de la página, logrando efectos dinámicos y estéticos. Los elementos clave son:

1. **Fondo con gradiente:** El cuerpo de la página (body) y el contenedor principal (.centrar) tienen fondos con gradientes diagonales, proporcionando un diseño moderno y atractivo.
2. **Efectos en botones:** Los **botones** (agregar, editar, eliminar) **cambian su color de fondo al pasar el cursor** (hover) utilizando transiciones suaves (transition: background-color 0.3s ease). Incorporan un efecto de elevación visual, desplazándose ligeramente hacia arriba (transform: translateY(-2px)) cuando se pasa el cursor, logrando un efecto interactivo y de respuesta inmediata.
3. **Resaltado en filas de tabla:** Al **pasar el cursor sobre las filas de la tabla** (tr:hover td), su fondo **cambia de color a un tono gris oscuro**, destacando la fila seleccionada de forma clara y elegante.
4. **Sombra en el contenedor:** El contenedor principal incluye una sombra ligera (box-shadow), aportando un efecto de profundidad que lo resalta sobre el fondo.



```
public > css > styles.css > ...
1  :root {
2    /* Nueva paleta de colores */
3    --color-primary: #ff0000;
4    --color-secondary: #40048e;
5    --color-accent: #40048e;
6    --color-background: #0000;
7    --color-text: #ffff;
8
9    /* Colores para botones */
10   --color-success: #006c2d;
11   --color-edit: blue;
12   --color-delete: red;
13 }
14
15 body {
16   background: linear-gradient(135deg, #3498db, #000); /* Fondo con gradiente */
17   color: var(--color-text);
18   font-family: cursive;
19   padding: 20px;
20   min-height: 100vh;
21   display: flex;
22   justify-content: center;
23   align-items: center;
24 }
25
26 h1 {
27   font-family: cursive;
28   color: black;
29 }
30
31
32 .centrar {
33   background: linear-gradient(135deg, #rgb(79, 239, 245), #fff); /* Fondo con gradiente */
34   border-radius: 12px;
35   box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
36   padding: 20px;
37   width: 100%;
```

```
public > css > styles.css > ...
32 .centrar {
33   max-width: 1200px;
34   margin: 20px auto;
35 }
36
37 .titulo h1 {
38   font-size: 2rem;
39   color: var(--color-primary);
40   text-align: center;
41   margin-bottom: 20px;
42 }
43
44 table {
45   width: 100%;
46   border-collapse: collapse;
47   border-radius: 8px;
48   overflow: hidden;
49   margin-top: 20px;
50 }
51
52 th, td {
53   padding: 15px;
54   text-align: left;
55 }
56
57 th {
58   background-color: var(--color-secondary);
59   color: white;
60   font-weight: bold;
61 }
62
63 td {
64   background-color: var(--color-background);
65   border-bottom: 1px solid #d3d3d3;
66 }
```

```
public > css > styles.css > ...
73 .tr:hover td {
74   background-color: #d3d3d3;
75 }
76
77 .acciones {
78   display: flex;
79   justify-content: center;
80   gap: 10px;
81 }
82
83 .agregar, .editar, .eliminar {
84   border: none;
85   padding: 10px 20px;
86   font-size: 14px;
87   font-family: cursive;
88   font-weight: bold;
89   color: white;
90   border-radius: 50px;
91   cursor: pointer;
92   transition: background-color 0.3s ease, transform 0.2s ease;
93 }
94
95 .agregar {
96   background-color: var(--color-success);
97 }
98
99 .editar {
100   background-color: var(--color-edit);
101 }
102
103 .eliminar {
104   background-color: var(--color-delete);
105 }
106
107 .agregar:hover {
108   background-color: #2ecc71;
109   transform: translateY(-2px);
110 }
```

The screenshot shows a code editor interface with a dark theme. The left sidebar displays a project structure with files like app.js, bd.sql, agregar.css, pages.js, package-lock.json, editar.css, styles.css, index.ejs, add.ejs, edit.ejs, and particulas.js. The main editor area shows a portion of the styles.css file:

```
public > css > styles.css > ...
107 .agregar:hover {
108 }
109 .editar:hover {
110   background-color: #2980b9;
111   transform: translateY(-2px);
112 }
113 .eliminar:hover {
114   background-color: #e0392b;
115   transform: translateY(-2px);
116 }
117 /* Estilo responsive */
118 @media (max-width: 768px) {
119   .titulo h1 {
120     font-size: 1.5rem;
121   }
122   table {
123     font-size: 14px;
124   }
125   .acciones {
126     flex-direction: column;
127     gap: 5px;
128   }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
```

The status bar at the bottom indicates the code is in Line 67, Column 1, with 4 spaces, using UTF-8 encoding, and includes links for Go Live, Ninja, and the date/time (06:51 a.m. 24/11/2024).

Este código en CSS corresponde al sistema de agregar usuarios a la Base de Datos. Sus principales características son:

#### Diseño:

- Utilice la **función clamp()** para tamaños dinámicos en fuentes, márgenes, rellenos y elementos, ajustándose a diferentes tamaños de pantalla.

#### Animaciones y transiciones:

- Los botones (agregar y eliminar) al igual que en el CSS anterior cambian de color suavemente al pasar el cursor, con una transición de 0.3 segundos.
- Los campos de entrada destacan al ser enfocados, cambiando el borde y añadiendo un efecto de sombra.

#### Efectos visuales:

- Usa colores definidos en **:root** para consistencia en toda la página, con fondos contrastantes y colores llamativos para botones.
- Aplica sombras (box-shadow) y bordes redondeados para resaltar elementos como formularios y botones, creando un diseño moderno.

```
public > css > () agregar.css > ...
1   :root {
2     /* Colores generales */
3     --color-primary: #1a1a1a;
4     --color-secondary: #34495e;
5     --color-accent: #3498db;
6     --color-background: #ecf0f1;
7     --color-text: #0000;
8   }
9   /* Colores para botones */
10  :button {
11    box-sizing: border-box;
12    margin: 0;
13    padding: 0;
14  }
15  body, button, input, table, th, td, form {
16    font-family: cursive;
17    font-size: clamp(14px, 1.5vw, 16px);
18    color: var(--color-text);
19  }
20  body {
21    background-color: #1a1a1a;
22    color: #3498db;
23    min-height: 100vh;
24    padding: clamp(6px, 5vw, 22px);
25    position: relative;
26    overflow-x: hidden;
27  }
28  #particles-js {
29    height: 100vh;
30    width: 100%;
31  }
```

```
#particles-js {
38   position: fixed;
39   top: 0;
40   left: 0;
41   z-index: -1;
42 }
43 .centrar {
44   display: flex;
45   flex-direction: column;
46   justify-content: center;
47   align-items: center;
48   gap: clamp(4px, 2vw, 2px);
49   width: min(100%, 1200px);
50   margin: 0 auto;
51   padding: clamp(4px, 1.5vw, 6px);
52 }
53 .borde {
54   background: var(--color-background);
55   padding: 15px;
56   border-radius: 8px;
57 }
58 table {
59   width: 100%;
60   border-collapse: separate;
61   border-spacing: 0;
62 }
63 th, td {
64   padding: clamp(6px, 1.2vw, 10px) clamp(8px, 1.5vw, 16px);
65   border-bottom: 1px solid #ddd;
66   word-wrap: break-word;
67   max-width: 300px;
68 }
```

Code editor showing the 'agregar.css' file from a project named 'PROYECTO FINAL'. The code defines styles for a form input field and its placeholder text.

```
public > css > agregar.css > ...
  th {
    background-color: var(--color-secondary);
    font-weight: 600;
    text-align: left;
  }
  .agregar, .eliminar {
    min-width: clamp(100px, 15vw, 120px);
    padding: clamp(8px, 1.5vw, 12px) clamp(16px, 2vw, 24px);
    font-size: clamp(14px, 1.5vw, 16px);
    font-weight: 600;
    color: #fff;
    border: none;
    border-radius: clamp(8px, 1.5vw, 12px);
    cursor: pointer;
    transition: 0.3s ease;
    text-align: center;
    white-space: nowrap;
  }
  .agregar {
    background-color: var(--color-success);
  }
  .eliminar {
    background-color: var(--color-delete);
  }
  .agregar:hover {
    background-color: #rgb(6, 165, 6);
  }
  .eliminar:hover {
    background-color: #c0392b;
  }
  .formagregar {
    width: 100px;
    max-width: min(90vw, 400px);
    border-radius: clamp(10px, 2vw, 15px);
    border: 1px solid #ddd;
    box-shadow: 0 6px 8px rgba(0, 0, 0, 0.1);
    margin: 0 auto;
  }
  .inputagregar {
    width: 100%;
    padding: clamp(8px, 1.2vw, 10px);
    border: 1px solid #ddd;
    border-radius: clamp(6px, 1.2vw, 10px);
    font-size: clamp(14px, 1.5vw, 16px);
    margin-bottom: clamp(8px, 1.2vw, 10px);
    transition: border-color 0.3s ease;
  }
  .inputagregar:focus {
    outline: none;
    border-color: #5eabb8;
    box-shadow: 0 0 3px rgba(94, 75, 139, 0.1);
  }
  .botones {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: clamp(6px, 1.5vw, 15px);
    width: 100%;
    margin-top: clamp(8px, 1.2vw, 12px);
  }
  @media screen and (max-width: 760px) {
```

Code editor showing the 'agregar.css' file from a project named 'PROYECTO FINAL'. The code includes a media query for screens up to 760px wide, which changes the button layout to a single column.

```
public > css > agregar.css > ...
  .formagregar {
    background-color: var(--color-background);
    padding: clamp(12px, 2vw, 20px);
    width: 100px;
    max-width: min(90vw, 400px);
    border-radius: clamp(10px, 2vw, 15px);
    border: 1px solid #ddd;
    box-shadow: 0 6px 8px rgba(0, 0, 0, 0.1);
    margin: 0 auto;
  }
  .inputagregar {
    width: 100%;
    padding: clamp(8px, 1.2vw, 10px);
    border: 1px solid #ddd;
    border-radius: clamp(6px, 1.2vw, 10px);
    font-size: clamp(14px, 1.5vw, 16px);
    margin-bottom: clamp(8px, 1.2vw, 10px);
    transition: border-color 0.3s ease;
  }
  .inputagregar:focus {
    outline: none;
    border-color: #5eabb8;
    box-shadow: 0 0 3px rgba(94, 75, 139, 0.1);
  }
  .botones {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: clamp(6px, 1.5vw, 15px);
    width: 100%;
    margin-top: clamp(8px, 1.2vw, 12px);
  }
  @media screen and (max-width: 760px) {
```

The screenshot shows a code editor window with the title "Proyecto Final". The left sidebar displays a file tree for a project named "PROYECTO FINAL". The "css" folder contains several files: "agregar.css", "editar.css", "styles.css", ".gitignore", "particulajs-config.js", and "particulajs.js". The main editor area shows the content of "agregar.css". The CSS code includes media queries for screens up to 760px and 480px, defining styles for buttons, forms, tables, and input fields. The code editor interface includes tabs for other files like "app.js", "bd.sql", "pages.js", "package-lock.json", "editar.css", "index.ejs", "add.ejs", and "edit.ejs". The bottom status bar shows the line number (197), column (1), and date/time (24/11/2024 06:54 a.m.).

```
public > css > agregar.css > ...
146 @media screen and (max-width: 760px) {
147   .botones {
148     gap: 8px;
149   }
150
151   .formagregar {
152     max-width: 100%;
153     padding: clamp(10px, 1.5vw, 15px);
154   }
155 }
156
157 @media screen and (max-width: 480px) {
158   body {
159     padding: 8px;
160   }
161   .botones {
162     flex-direction: column;
163     gap: 6px;
164   }
165   .agregar, .eliminar {
166     width: 100%;
167   }
168   table {
169     display: block;
170     overflow-x: auto;
171   }
172   td, th {
173     min-width: 120px;
174     padding: 8px;
175   }
176   .formagregar {
177     padding: 12px;
178   }
179   .inputagregar {
180     font-size: 16px;
181     margin-bottom: 8px;
182   }
183 }
184
185 @media screen and (max-width: 320px) {
186   .formagregar {
187     padding: 8px;
188   }
189   .inputagregar {
190     padding: 6px;
191     margin-bottom: 6px;
192   }
193   .agregar, .eliminar {
194     padding: 6px 12px;
195   }
196 }
```

This screenshot is nearly identical to the one above, showing the same code editor window for "agregar.css". The only difference is in the line numbers, which have increased from 197 to 197. The CSS code remains the same, defining styles for different screen widths and various UI components.

```
public > css > agregar.css > ...
157 @media screen and (max-width: 480px) {
158   .botones {
159     gap: 8px;
160   }
161   .agregar, .eliminar {
162     width: 100%;
163   }
164   table {
165     display: block;
166     overflow-x: auto;
167   }
168   td, th {
169     min-width: 120px;
170     padding: 8px;
171   }
172   .formagregar {
173     padding: 12px;
174   }
175   .inputagregar {
176     font-size: 16px;
177     margin-bottom: 8px;
178   }
179 }
180
181 @media screen and (max-width: 320px) {
182   .formagregar {
183     padding: 8px;
184   }
185   .inputagregar {
186     padding: 6px;
187     margin-bottom: 6px;
188   }
189   .agregar, .eliminar {
190     padding: 6px 12px;
191   }
192 }
```

Este CSS corresponde al sistema de actualización o edición de datos por parte del usuario. Sus principales características son:

**1. Diseño:**

- De igual manera que la hoja de estilos anterior, utilice **clamp()** para adaptarse a diferentes tamaños de pantalla, asegurando que el texto, los márgenes y los botones cambien de tamaño según la resolución.

**2. Interactividad y transiciones:**

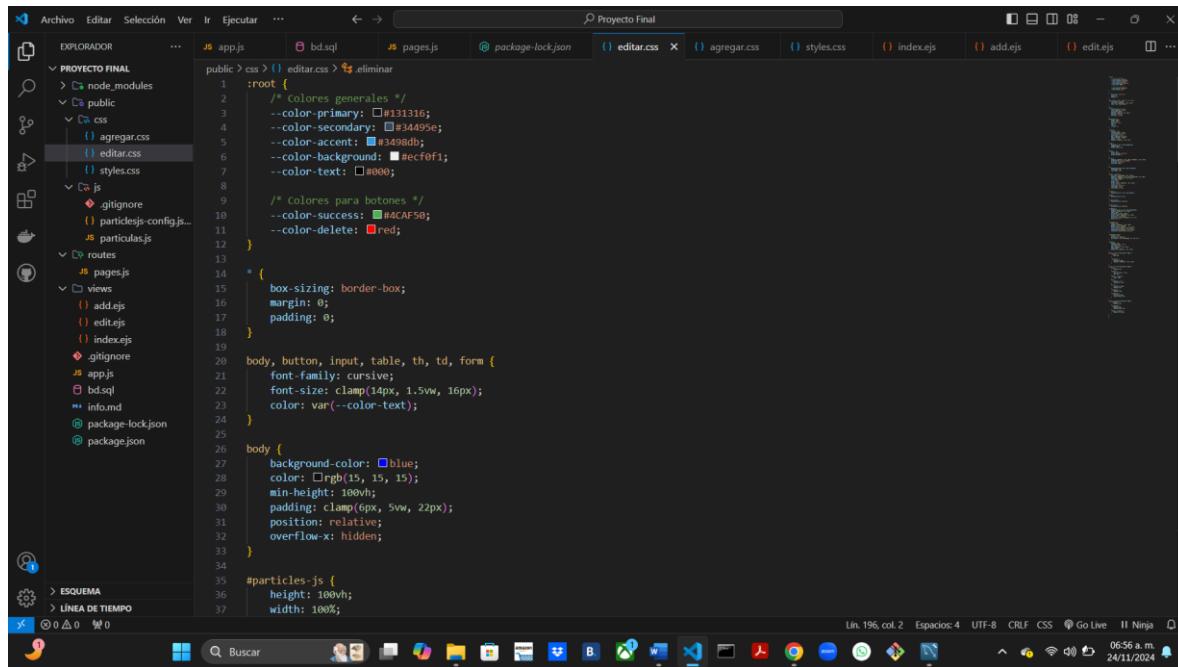
- Los botones (editar y eliminar) tienen transiciones de color y cambio de fondo con un efecto suave cuando se pasa el cursor sobre ellos, mejorando la experiencia del usuario (exactamente lo mismo que las páginas anteriores).

**3. Estética visual:**

- Se define una paleta de colores con tonos oscuros y un fondo claro, creando un contraste llamativo entre el texto y los elementos interactivos.
- Los botones tienen un diseño redondeado, y los formularios y tablas están diseñados con bordes suaves y sombras sutiles, proporcionando un aspecto moderno y accesible.

**4. Accesibilidad:**

- Los botones están diseñados para ser lo suficientemente grandes y con una transición suave al pasar el mouse, facilitando la interacción en dispositivos táctiles.



```
public :root {
    /* Colores generales */
    --color-primary: #131316;
    --color-secondary: #343495e;
    --color-accent: #3498d8;
    --color-background: #ecf0f1;
    --color-text: #0000;
}

/* Colores para botones */
--color-success: #4CAF50;
--color-delete: red;

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body, button, input, table, th, td, form {
    font-family: cursive;
    font-size: clamp(14px, 1.5vw, 16px);
    color: var(--color-text);
}

body {
    background-color: blue;
    color: #rgb(15, 15, 15);
    min-height: 100vh;
    padding: clamp(6px, 5vw, 22px);
    position: relative;
    overflow-x: hidden;
}

#particles-js {
    height: 100vh;
    width: 100%;
```

Code editor screenshot showing the CSS for a table header and body. The code defines styles for th and td elements, including padding, border-collapse, border-spacing, and background-color.

```
public > css > editar.css > eliminar
35 #particulas {
36   position: fixed;
37   top: 0;
38   left: 0;
39   z-index: -1;
40 }
41 .centrar {
42   display: flex;
43   flex-direction: column;
44   justify-content: center;
45   align-items: center;
46   gap: clamp(4px, 2vw, 2px);
47   width: min(100%, 1200px);
48   margin: 0 auto;
49   padding: clamp(4px, 1.5vw, 6px);
50 }
51 .borde {
52   background: var(--color-background);
53   padding: 15px;
54   border-radius: 8px;
55 }
56 table {
57   width: 100%;
58   border-collapse: separate;
59   border-spacing: 0;
60 }
61 th, td {
62   padding: clamp(6px, 1.2vw, 10px) clamp(8px, 1.5vw, 16px);
63   border-bottom: 1px solid #ddd;
64   word-wrap: break-word;
65   max-width: 300px;
66 }
67 th {
68   padding: clamp(6px, 1.2vw, 10px) clamp(8px, 1.5vw, 16px);
69   border-bottom: 1px solid #ddd;
70   word-wrap: break-word;
71   max-width: 300px;
72 }
```

Code editor screenshot showing the CSS for table cells and form controls. The code defines styles for th, td, .eliminar, .editar, .eliminar:hover, .editar:hover, .formeditar, and .formeliminar elements.

```
public > css > editar.css > eliminar
74 th {
75   background-color: var(--color-secondary);
76   font-weight: 600;
77   text-align: left;
78 }
79 .editar, .eliminar {
80   min-width: clamp(100px, 15vw, 120px);
81   padding: clamp(8px, 1.5vw, 12px) clamp(16px, 2vw, 24px);
82   font-size: clamp(14px, 1.5vw, 16px);
83   font-weight: 600;
84   color: #fff;
85   border: none;
86   border-radius: clamp(8px, 1.5vw, 12px);
87   cursor: pointer;
88   transition: 0.3s ease;
89   text-align: center;
90   white-space: nowrap;
91 }
92 .eliminar {
93   background-color: var(--color-delete);
94 }
95 .editar {
96   background-color: #blue;
97 }
98 .eliminar:hover {
99   background-color: #c0392b;
100 }
101 .editar:hover {
102   background-color: #2980b9;
103 }
104 .eliminar:active {
105   background-color: #e67e22;
106 }
107 .editar:active {
108   background-color: #2980b9;
109 }
110 .formeditar {
111   background-color: var(--color-background);
112   padding: clamp(12px, 2vw, 20px);
```

```
public > css > editar.css > eliminar
  .eliminar {
    .editar:hover {
      .formeditar {
        background-color: var(--color-background);
        padding: clamp(12px, 2vw, 20px);
        width: 100%;
        max-width: min(90vw, 400px);
        border-radius: clamp(10px, 2vw, 15px);
        border: 1px solid #ddd;
        box-shadow: 0 6px 8px rgba(0, 0, 0, 0.1);
        margin: 0 auto;
      }
    }
  }

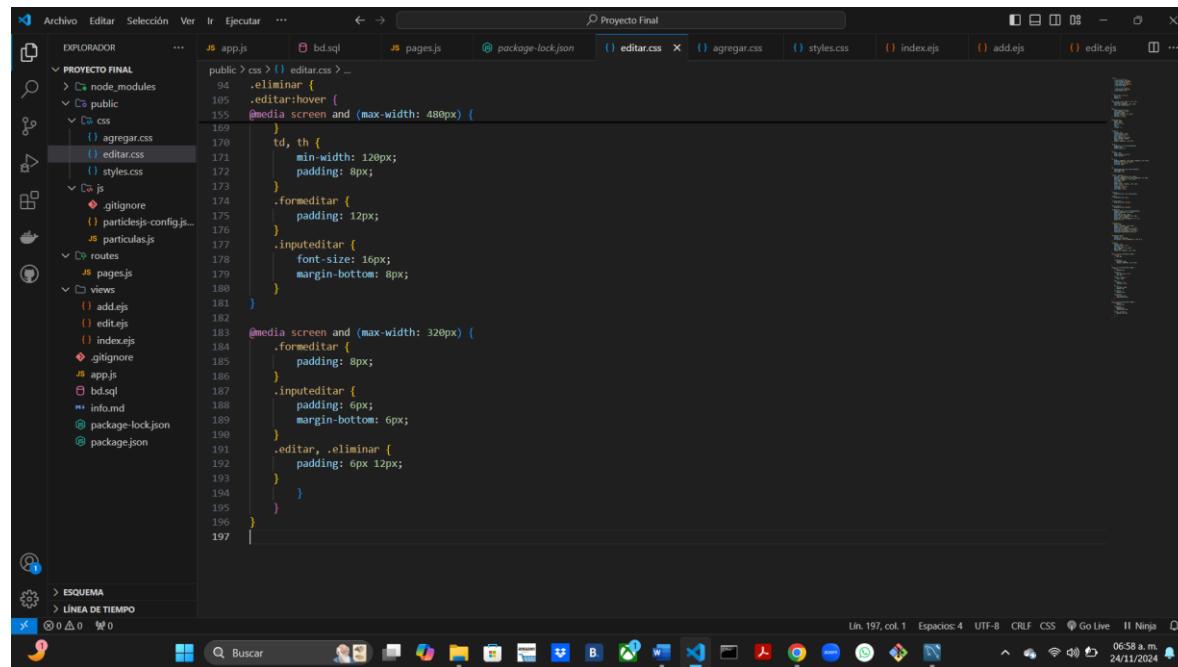
  .inputeditar {
    width: 100%;
    padding: clamp(8px, 1.2vw, 10px);
    border: 1px solid #ddd;
    border-radius: clamp(6px, 1.2vw, 10px);
    font-size: clamp(14px, 1.5vw, 16px);
    margin-bottom: clamp(8px, 1.2vw, 10px);
    transition: border-color 0.3s ease;
  }

  .inputeditar:focus {
    outline: none;
    border-color: #5e40b0;
    box-shadow: 0 0 0 3px rgba(94, 75, 139, 0.1);
  }

  .botones {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: clamp(6px, 1.5vw, 15px);
    width: 100%;
    margin-top: clamp(8px, 1.2vw, 12px);
  }
```

The screenshot shows a code editor interface with the following details:

- Project Structure:** The left sidebar displays the project structure:
  - PROYECTO FINAL
  - node\_modules
  - public
    - css
      - agregar.css
      - editar.css
      - styles.css
    - js
      - participlejs-config.js
      - particulas.js
    - routes
      - pages.js
    - views
      - add.ejs
      - edit.ejs
      - index.ejs
      - .gitignore
  - bd.sql
  - infomd
  - package-lock.json
  - package.json
- Code Editor:** The main area shows the content of the `editar.css` file, which contains CSS rules for a table editor. The code includes media queries for screens up to 480px wide, defining styles for rows, columns, and individual cells.
- Bottom Bar:** The bottom bar includes navigation icons (back, forward, search), a file browser, and system status indicators (battery, signal, volume).
- Right Panel:** A vertical panel on the right side shows a tree view of the project files and a preview of the current file's content.



```
public > css > editar.css > ...
  94 .eliminar {
  95   .editar:hover {
  96     @media screen and (max-width: 480px) {
  97       td, th {
  98         min-width: 120px;
  99         padding: 8px;
 100       }
 101       .formeditar {
 102         padding: 12px;
 103       }
 104       .inputeditar {
 105         font-size: 16px;
 106         margin-bottom: 8px;
 107       }
 108     }
 109     @media screen and (max-width: 320px) {
 110       .formeditar {
 111         padding: 8px;
 112       }
 113       .inputeditar {
 114         padding: 6px;
 115         margin-bottom: 6px;
 116       }
 117       .editar, .eliminar {
 118         padding: 6px 12px;
 119       }
 120     }
 121   }
 122 }
```