Manual Tecnico - Desarrollo Web

Nombre: Angel Emanuel Rodriguez Corado
Carnet: 202404856

Índice

Introducción

En este manual se muestra los diferentes códigos usados para desarrollar el proyecto web, se muestra tanto el backend como el frontend para que cualquier usuario pueda ver como fue creado este proyecto, el Backend se trabajó con Node.js y el Frontend se trabajó con React

Backend

Routes

comentarios.js

```
const express = require('express')
const router = express.Router()
const db = require('../config/db')

// Obtener todos los comentarios de una publicación
router.get('/:id_publicacion', (req, res) => {
    const { id_publicacion } = req.params
    const query = `
        SELECT c.id_comentario, c.mensaje, c.fecha_creacion,
               u.id_usuario, u.nombres, u.apellidos
        FROM Comentarios c
        JOIN Usuarios u ON c.id_usuario = u.id_usuario
        WHERE c.id_publicacion = ?
        ORDER BY c.fecha_creacion ASC
    `
    db.query(query, [id_publicacion], (err, results) => {
        if (err) return res.status(500).json({ error: err })
        res.json(results)
    })
})

// Crear un nuevo comentario
router.post('/', (req, res) => {
    const { id_publicacion, id_usuario, mensaje } = req.body
    if (!id_publicacion || !id_usuario || !mensaje) {
        return res.status(400).json({ error: 'Datos incompletos' })
    }

    const query = `
        INSERT INTO Comentarios (id_publicacion, id_usuario, mensaje,
fecha_creacion)
        VALUES (?, ?, ?, NOW())
    `
    db.query(query, [id_publicacion, id_usuario, mensaje], (err, result) =>
{
        if (err) return res.status(500).json({ error: err })
```

```
        res.json({ mensaje: 'Comentario creado', id: result.insertId })
    })
})

module.exports = router
```

cursos.js

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

// Obtener todos los cursos
router.get('/', (req, res) => {
  const sql = `
    SELECT c.id_curso, c.nombre_curso, c.seccion,
           p.id_profesor, p.nombres AS profesor_nombres, p.apellidos AS
profesor_apellidos
    FROM Cursos c
    LEFT JOIN Profesores p ON c.id_profesor = p.id_profesor
  `;
  db.query(sql, (err, results) => {
    if (err) return res.status(500).json({ error: err });
    res.json(results);
  });
});

// Obtener curso por ID
router.get('/:id', (req, res) => {
  const { id } = req.params;
  const sql = `
    SELECT c.id_curso, c.nombre_curso, c.seccion,
           p.id_profesor, p.nombres AS profesor_nombres, p.apellidos AS
profesor_apellidos
    FROM Cursos c
    LEFT JOIN Profesores p ON c.id_profesor = p.id_profesor
    WHERE c.id_curso = ?
  `;
  db.query(sql, [id], (err, result) => {
    if (err) return res.status(500).json({ error: err });
```

```
    if (result.length === 0) return res.status(404).json({ mensaje: 'Curso
no encontrado' });
    res.json(result[0]);
  });
});

// Agregar curso
router.post('/', (req, res) => {
  const { nombre_curso, seccion, id_profesor } = req.body;
  if (!nombre_curso) return res.status(400).json({ error: 'El nombre del
curso es obligatorio' });

  db.query(
    'INSERT INTO Cursos (nombre_curso, seccion, id_profesor) VALUES (?, ?,
?)',
    [nombre_curso, seccion || null, id_profesor || null],
    (err, result) => {
      if (err) return res.status(500).json({ error: err });
      res.status(201).json({ mensaje: 'Curso agregado', id: result.insertId
});
    }
  );
});

// Actualizar curso
router.put('/:id', (req, res) => {
  const { id } = req.params;
  const { nombre_curso, seccion, id_profesor } = req.body;

  db.query(
    'UPDATE Cursos SET nombre_curso = ?, seccion = ?, id_profesor = ? WHERE
id_curso = ?',
    [nombre_curso, seccion || null, id_profesor || null, id],
    (err, result) => {
      if (err) return res.status(500).json({ error: err });
      if (result.affectedRows === 0) return res.status(404).json({ mensaje:
'Curso no encontrado' });
      res.json({ mensaje: 'Curso actualizado' });
    }
  );
});

// Eliminar curso
```

```
router.delete('/:id', (req, res) => {
  const { id } = req.params;
  db.query('DELETE FROM Cursos WHERE id_curso = ?', [id], (err, result) =>
{
    if (err) return res.status(500).json({ error: err });
    if (result.affectedRows === 0) return res.status(404).json({ mensaje:
'Curso no encontrado' });
    res.json({ mensaje: 'Curso eliminado' });
  });
});

module.exports = router;
```

forgotPassword.js

```
const express = require('express');
const router = express.Router();
const db = require('../config/db');
const bcrypt = require('bcrypt');

// Reestablecer contraseña
router.put('/', async (req, res) => {
    const { registro_academico, correo, nueva_contrasena } = req.body;

    if (!registro_academico || !correo || !nueva_contrasena) {
        return res.status(400).json({ error: 'Todos los campos son obligatorios'
});
    }

    try {
        // Verificar que el usuario exista
        db.query(
            'SELECT * FROM Usuarios WHERE registro_academico = ? AND correo = ?',
            [registro_academico, correo],
            async (err, result) => {
                if (err) return res.status(500).json({ error: err });
                if (result.length === 0) {
                    return res.status(404).json({ error: 'Usuario no encontrado o
datos incorrectos' });
```

```
                }

                // Hashear la nueva contraseña
                const hashedPassword = await bcrypt.hash(nueva_contrasena, 10);

                // Actualizar contraseña
                db.query(
                    'UPDATE Usuarios SET contrasena = ? WHERE registro_academico
= ? AND correo = ?',
                    [hashedPassword, registro_academico, correo],
                    (err2) => {
                        if (err2) return res.status(500).json({ error: err2 });
                        res.json({ mensaje: 'Contraseña reestablecida
correctamente' });
                    }
                );
            }
        );
    } catch (error) {
        res.status(500).json({ error: 'Error al reestablecer contraseña' });
    }
});

module.exports = router;
```

profesores.js

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

// Obtener todos los profesores
router.get('/', (req, res) => {
  db.query('SELECT * FROM Profesores', (err, results) => {
    if (err) return res.status(500).json({ error: err });
    res.json(results);
  });
```

```
});

// Obtener profesor por ID
router.get('/:id', (req, res) => {
  const { id } = req.params;
  db.query('SELECT * FROM Profesores WHERE id_profesor = ?', [id], (err, result)
=> {
    if (err) return res.status(500).json({ error: err });
    if (result.length === 0) return res.status(404).json({ mensaje: 'Profesor no
encontrado' });
    res.json(result[0]);
  });
});

// Agregar profesor
router.post('/', (req, res) => {
  const { nombres, apellidos } = req.body;
  if (!nombres || !apellidos) return res.status(400).json({ error: 'Todos los
campos son obligatorios' });

  db.query(
    'INSERT INTO Profesores (nombres, apellidos) VALUES (?, ?)',
    [nombres, apellidos],
    (err, result) => {
      if (err) return res.status(500).json({ error: err });
      res.status(201).json({ mensaje: 'Profesor agregado', id: result.insertId
});
    }
  );
});

// Actualizar profesor
router.put('/:id', (req, res) => {
  const { id } = req.params;
  const { nombres, apellidos } = req.body;

  db.query(
    'UPDATE Profesores SET nombres = ?, apellidos = ? WHERE id_profesor = ?',
    [nombres, apellidos, id],
```

```
    (err, result) => {
      if (err) return res.status(500).json({ error: err });
      if (result.affectedRows === 0) return res.status(404).json({ mensaje:
'Profesor no encontrado' });
      res.json({ mensaje: 'Profesor actualizado' });
    }
  );
});

// Eliminar profesor
router.delete('/:id', (req, res) => {
  const { id } = req.params;
  db.query('DELETE FROM Profesores WHERE id_profesor = ?', [id], (err, result) =>
{
    if (err) return res.status(500).json({ error: err });
    if (result.affectedRows === 0) return res.status(404).json({ mensaje:
'Profesor no encontrado' });
    res.json({ mensaje: 'Profesor eliminado' });
  });
});

module.exports = router;
```

publicaciones.js

```
// routes/publicaciones.js
const express = require('express')
const router = express.Router()
const db = require('../config/db')

// Obtener todas las publicaciones, más recientes primero
router.get('/', (req, res) => {
    const query = `
    SELECT
        p.id_publicacion,
        p.mensaje,
        p.fecha_creacion,
```

```
        u.id_usuario, u.nombres, u.apellidos,
        c.id_curso, c.nombre_curso,
        pr.id_profesor, pr.nombres AS nombre_profesor, pr.apellidos AS
apellido_profesor
    FROM Publicaciones p
    JOIN Usuarios u ON p.id_usuario = u.id_usuario
    LEFT JOIN Cursos c ON p.id_curso = c.id_curso
    LEFT JOIN Profesores pr ON p.id_profesor = pr.id_profesor
    ORDER BY p.fecha_creacion DESC
    `
    db.query(query, (err, results) => {
    if (err) return res.status(500).json({ error: err })
    res.json(results)
    })
})

// Crear una nueva publicación
router.post('/', (req, res) => {
    const { id_usuario, id_curso, id_profesor, mensaje } = req.body

    if (!id_usuario || !mensaje || (!id_curso && !id_profesor)) {
    return res.status(400).json({ error: 'Datos incompletos' })
    }

    const query = `
    INSERT INTO Publicaciones (id_usuario, id_curso, id_profesor, mensaje,
fecha_creacion)
    VALUES (?, ?, ?, ?, NOW())
    `
    db.query(query, [id_usuario, id_curso || null, id_profesor || null, mensaje],
(err, result) => {
    if (err) return res.status(500).json({ error: err })
    res.json({ mensaje: 'Publicación creada', id: result.insertId })
    })
})

module.exports = router
```

usuarios.js

```
const express = require('express')
const router = express.Router()
const db = require('../config/db')
const bcrypt = require('bcrypt')

// Obtener todos los usuarios
router.get('/', (req, res) => {
  db.query('SELECT * FROM Usuarios', (err, results) => {
    if (err) return res.status(500).json({ error: err })
    res.json(results)
  })
})

// Obtener usuario por ID
router.get('/:id', (req, res) => {
  const { id } = req.params
  db.query('SELECT * FROM Usuarios WHERE id_usuario = ?', [id], (err, result) =>
{
    if (err) return res.status(500).json({ error: err })
    if (result.length === 0) return res.status(404).json({ mensaje: 'Usuario no
encontrado' })
    res.json(result[0])
  })
})

// Registrar nuevo usuario
router.post('/', async (req, res) => {
  const { registro_academico, nombres, apellidos, correo, contrasena } = req.body

  if (!registro_academico || !nombres || !apellidos || !correo || !contrasena) {
    return res.status(400).json({ error: 'Todos los campos son obligatorios' })
  }

  try {
    const hashedPassword = await bcrypt.hash(contrasena, 10)
    db.query(
```

```
      'INSERT INTO Usuarios (registro_academico, nombres, apellidos, correo,
contrasena) VALUES (?, ?, ?, ?, ?)',
      [registro_academico, nombres, apellidos, correo, hashedPassword],
      (err, result) => {
        if (err) return res.status(500).json({ error: err })
        res.json({ mensaje: 'Usuario registrado', id: result.insertId })
      }
    )
  } catch (error) {
    res.status(500).json({ error: 'Error al registrar usuario' })
  }
})

// Actualizar usuario
router.put('/:id', async (req, res) => {
  const { id } = req.params
  const { registro_academico, nombres, apellidos, correo, contrasena } = req.body

  if (!registro_academico || !nombres || !apellidos || !correo || !contrasena) {
    return res.status(400).json({ error: 'Todos los campos son obligatorios' })
  }

  try {
    const hashedPassword = await bcrypt.hash(contrasena, 10)
    db.query(
      'UPDATE Usuarios SET registro_academico = ?, nombres = ?, apellidos = ?,
correo = ?, contrasena = ? WHERE id_usuario = ?',
      [registro_academico, nombres, apellidos, correo, hashedPassword, id],
      (err) => {
        if (err) return res.status(500).json({ error: err })
        res.json({ mensaje: 'Usuario actualizado', id })
      }
    )
  } catch (error) {
    res.status(500).json({ error: 'Error al actualizar usuario' })
  }
})

// Eliminar usuario
```

```
router.delete('/:id', (req, res) => {
  const { id } = req.params
  db.query('DELETE FROM Usuarios WHERE id_usuario = ?', [id], (err) => {
    if (err) return res.status(500).json({ error: err })
    res.json({ mensaje: 'Usuario eliminado' })
  })
})

// Login de usuario
router.post('/login', (req, res) => {
    const { registro_academico, contrasena } = req.body;

    if (!registro_academico || !contrasena) {
        return res.status(400).json({ mensaje: 'Registro académico y contraseña
son requeridos' });
    }

    db.query(
        'SELECT * FROM Usuarios WHERE registro_academico = ?',
        [registro_academico],
        async (err, result) => {
            if (err) return res.status(500).json({ error: err });
            if (result.length === 0) {
                return res.status(401).json({ mensaje: 'Usuario no encontrado'
});
            }

            const usuario = result[0];
            const match = await bcrypt.compare(contrasena, usuario.contrasena);
            if (!match) return res.status(401).json({ mensaje: 'Contraseña
incorrecta' });

            // Eliminamos la contraseña antes de enviar la respuesta
            delete usuario.contrasena;
            res.json({ mensaje: 'Login exitoso', usuario });
        }
    );
});
```

```
// Obtener usuario por registro_academico
router.get('/registro/:registro_academico', (req, res) => {
  const { registro_academico } = req.params
  db.query(
    'SELECT * FROM Usuarios WHERE registro_academico = ?',
    [registro_academico],
    (err, result) => {
      if (err) return res.status(500).json({ error: err })
      if (result.length === 0) {
        return res.status(404).json({ mensaje: 'Usuario no encontrado' })
      }
      res.json(result[0])
    }
  )
})


module.exports = router
```

.env

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=281205
DB_NAME=db_facultad
DB_PORT=3306
```

index.js

```
require('dotenv').config();
const express = require('express');
const cors = require('cors'); // ✅ Importamos cors
const app = express();
const db = require('./config/db'); // mi pool de conexiones

// Importamos las rutas
```

```
const usuariosRoutes = require('./routes/usuarios');
const profesoresRoutes = require('./routes/profesores');
const cursosRoutes = require('./routes/cursos');
const forgotPasswordRoutes = require('./routes/forgotPassword');
const publicacionesRoutes = require('./routes/publicaciones');
const comentariosRouter = require('./routes/comentarios')

//Habilitar CORS para React
app.use(cors({
    origin: 'http://localhost:3000', // puerto donde corre tu React
    credentials: true
}));

app.use(express.json());

// Ruta de prueba a la base de datos
app.get('/test-db', (req, res) => {
    db.query('SELECT NOW() AS fecha', (err, result) => {
        if (err) {
            console.error('Error en la consulta:', err);
            return res.status(500).json({ error: 'Error en la consulta', details:
err });
        }
        res.json({ mensaje: 'Conexión exitosa con la base de Datos', resultado:
result });
    });
});

// Rutas
app.use('/usuarios', usuariosRoutes);
app.use('/profesores', profesoresRoutes);
app.use('/cursos', cursosRoutes);
app.use('/forgot-password', forgotPasswordRoutes);
app.use('/publicaciones', publicacionesRoutes);
app.use('/comentarios', comentariosRouter)


// Iniciar servidor
const PORT = process.env.PORT || 5000;
```

```
app.listen(PORT, () => {
    console.log(`Servidor corriendo en http://localhost:${PORT}`);
});
```

Frontend

src

App.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Login from './Login.jsx';
import Register from './Register.jsx';
import ForgotPassword from './ForgotPassword.jsx';
import PantallaInicial from './PantallaInicial.jsx'
import CrearPublicacion from './CrearPublicacion.jsx';
import Comentarios from './Comentarios.jsx'
import Perfil from './Perfil.jsx';
import './App.css';

function App() {
  return (
    <Router>
      <div className="App">
        <Routes>
          <Route path="/" element={<Login />} />
          <Route path="/register" element={<Register />} />
          <Route path="/forgot-password" element={<ForgotPassword />} />
          <Route path="/inicio" element={<PantallaInicial />} />
          <Route path="/crear-publicacion" element={<CrearPublicacion />} />
          <Route path="/publicacion/:id" element={<Comentarios />} />
          <Route path="/perfil/:registro" element={<Perfil />} />
        </Routes>
      </div>
```

```
    </Router>
  );
}

export default App;
```

Comentarios.jsx

```
import React, { useState, useEffect } from 'react'
import { useParams, useNavigate } from 'react-router-dom'
import axios from 'axios'
import { API_URL } from './config'
import './Comentarios.css'

function Comentarios() {
  const { id } = useParams()
  const navigate = useNavigate()

  const [comentarios, setComentarios] = useState([])
  const [mensaje, setMensaje] = useState('')
  const [error, setError] = useState('')
  const [exito, setExito] = useState('')
  const usuario = JSON.parse(localStorage.getItem('usuario'))

  const fetchComentarios = async () => {
    try {
      const res = await axios.get(`${API_URL}/comentarios/${id}`)
      setComentarios(res.data)
    } catch (err) {
      console.error(err)
    }
  }

  useEffect(() => {
    fetchComentarios()
  }, [])
```

```
const handleSubmit = async (e) => {
  e.preventDefault()
  setError('')
  setExito('')

  if (!mensaje.trim()) {
    setError('Escribe un mensaje')
    return
  }

  try {
    await axios.post(`${API_URL}/comentarios`, {
      id_publicacion: id,
      id_usuario: usuario.id_usuario,
      mensaje
    })
    setExito('Comentario agregado correctamente')
    setMensaje('')
    fetchComentarios()
  } catch (err) {
    console.error(err)
    setError(err.response?.data?.error || 'Error al agregar comentario')
  }
}

return (
  <div className="comentarios-page">
    <header className="header-comentarios">
      <h2>Comentarios de la Publicación</h2>
      <button className="boton-regresar" onClick={() => navigate('/inicio')}>
        Regresar
      </button>
    </header>

    <main className="contenido-comentarios">
      {comentarios.length === 0 && <p>No hay comentarios aún.</p>}
      {comentarios.map(c => (
        <div key={c.id_comentario} className="comentario-card">
          <div className="comentario-header">
```

```
              <span className="usuario">{c.nombres} {c.apellidos}</span>
              <span className="fecha">{new
Date(c.fecha_creacion).toLocaleString()}</span>
            </div>
            <div className="mensaje">{c.mensaje}</div>
          </div>
        ))}

        <form className="form-comentario" onSubmit={handleSubmit}>
          {error && <p className="error">{error}</p>}
          {exito && <p className="exito">{exito}</p>}
          <textarea
            placeholder="Escribe un comentario..."
            value={mensaje}
            onChange={(e) => setMensaje(e.target.value)}
          />
          <button type="submit">Agregar Comentario</button>
        </form>
      </main>
    </div>
  )
}

export default Comentarios
```

CrearPublicacion.jsx

```
import React, { useState, useEffect } from 'react'
import axios from 'axios'
import { API_URL } from './config'
import { useNavigate } from 'react-router-dom'
import './CrearPublicacion.css';

function CrearPublicacion({ onNuevaPublicacion }) {
  const [tipo, setTipo] = useState('Profesor')
  const [profesores, setProfesores] = useState([])
  const [cursos, setCursos] = useState([])
```

```
const [seleccion, setSeleccion] = useState('')
const [contenido, setContenido] = useState('')
const [mensajeError, setMensajeError] = useState('')
const [mensajeExito, setMensajeExito] = useState('')

const navigate = useNavigate()
const usuario = JSON.parse(localStorage.getItem('usuario'))

useEffect(() => {
  axios.get(`${API_URL}/profesores`)
    .then(res => setProfesores(res.data))
    .catch(err => console.error(err))

  axios.get(`${API_URL}/cursos`)
    .then(res => setCursos(res.data))
    .catch(err => console.error(err))
}, [])

const handleSubmit = async (e) => {
  e.preventDefault()
  setMensajeError('')
  setMensajeExito('')

  if (!seleccion || !contenido.trim()) {
    setMensajeError('Todos los campos son obligatorios')
    return
  }

  try {
    const data = {
      id_usuario: usuario.id_usuario,
      id_curso: tipo === 'Curso' ? seleccion : null,
      id_profesor: tipo === 'Profesor' ? seleccion : null,
      mensaje: contenido
    }

    await axios.post(`${API_URL}/publicaciones`, data)
    setMensajeExito('Publicación creada correctamente')
    setContenido('')
```

```
      setSeleccion('')
      onNuevaPublicacion() // actualizar PantallaInicial
    } catch (err) {
      console.error(err)
      setMensajeError(err.response?.data?.error || 'Error al crear publicación')
    }
  }

  const handleRegresar = () => {
    navigate('/inicio') // redirige a Pantalla Inicial
  }

  return (
    <div className="crear-publicacion">
      <h3>Crear Publicación</h3>
      {mensajeError && <p className="error">{mensajeError}</p>}
      {mensajeExito && <p className="exito">{mensajeExito}</p>}

      <form onSubmit={handleSubmit}>
        <label>Tipo:</label>
        <select value={tipo} onChange={e => { setTipo(e.target.value);
setSeleccion('') }}>
          <option value="Profesor">Profesor</option>
          <option value="Curso">Curso</option>
        </select>

        {tipo === 'Profesor' && (
          <>
            <label>Profesor:</label>
            <select value={seleccion} onChange={e =>
setSeleccion(e.target.value)}>
              <option value="">--Seleccione--</option>
              {profesores.map(p => (
                <option key={p.id_profesor} value={p.id_profesor}>
                  {p.nombres} {p.apellidos}
                </option>
              ))}
            </select>
          </>
```

```
        )}

        {tipo === 'Curso' && (
          <>
            <label>Curso:</label>
            <select value={seleccion} onChange={e =>
setSeleccion(e.target.value)}>
              <option value="">--Seleccione--</option>
              {cursos.map(c => (
                <option key={c.id_curso} value={c.id_curso}>
                  {c.nombre_curso}
                </option>
              ))}
            </select>
          </>
        )}

        <label>Mensaje:</label>
        <textarea
          value={contenido}
          onChange={e => setContenido(e.target.value)}
          placeholder="Escribe tu publicación..."
        />

        <div className="botones">
          <button type="submit">Publicar</button>
          <button type="button" onClick={handleRegresar}>Regresar</button>
        </div>
      </form>
    </div>
  )
}

export default CrearPublicacion
```

ForgotPassword.jsx

```jsx
import React, { useState } from 'react'
import axios from 'axios'
import { API_URL } from './config'

function ForgotPassword() {
  const [registro, setRegistro] = useState('')
  const [correo, setCorreo] = useState('')
  const [nuevaContrasena, setNuevaContrasena] = useState('')
  const [mensaje, setMensaje] = useState('')
  const [error, setError] = useState('')

  const handleReset = async (e) => {
    e.preventDefault()
    setMensaje('')
    setError('')

    try {
      const response = await axios.put(`${API_URL}/forgot-password`, {
        registro_academico: registro,
        correo,
        nueva_contrasena: nuevaContrasena
      })

      setMensaje(response.data.mensaje)
      setRegistro('')
      setCorreo('')
      setNuevaContrasena('')
    } catch (err) {
      console.error('Error al reestablecer contraseña:', err)
      if (err.response && err.response.data && err.response.data.mensaje) {
        setError(err.response.data.mensaje)
      } else {
        setError('No se pudo conectar con el servidor')
      }
    }
  }
```

```
  return (
    <div className="form-container">
      <h2>Reestablecer Contraseña</h2>
      <form onSubmit={handleReset}>
        <input
          type="text"
          placeholder="Registro Académico"
          value={registro}
          onChange={(e) => setRegistro(e.target.value)}
          required
        />
        <input
          type="email"
          placeholder="Correo electrónico"
          value={correo}
          onChange={(e) => setCorreo(e.target.value)}
          required
        />
        <input
          type="password"
          placeholder="Nueva Contraseña"
          value={nuevaContrasena}
          onChange={(e) => setNuevaContrasena(e.target.value)}
          required
        />
        <button type="submit">Reestablecer</button>
      </form>
      {mensaje && <p style={{ color: 'green' }}>{mensaje}</p>}
      {error && <p style={{ color: 'red' }}>{error}</p>}
    </div>
  )
}

export default ForgotPassword
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Login.jsx

```
import React, { useState } from 'react'
import { useNavigate, Link } from 'react-router-dom'
import axios from 'axios'
import { API_URL } from './config'

function Login() {
    const [registro, setRegistro] = useState('')
    const [contrasena, setContrasena] = useState('')
    const [error, setError] = useState('')
    const navigate = useNavigate()

    const handleLogin = async (e) => {
        e.preventDefault()
        setError('') // limpiar error previo

        try {
```

```
            const response = await axios.post(`${API_URL}/usuarios/login`, {
                registro_academico: registro,
                contrasena
            })

            // Login exitoso
            console.log('Login exitoso:', response.data)
            localStorage.setItem('usuario',
JSON.stringify(response.data.usuario))
            setRegistro('')
            setContrasena('')
            navigate('/inicio') // Redirigir a la página principal
        } catch (err) {
            console.error('Error login:', err)

            if (err.response && err.response.data && err.response.data.mensaje) {
                // Mostrar el mensaje exacto que envía el backend
                setError(err.response.data.mensaje)
            } else if (err.request) {
                // No hubo respuesta del servidor
                setError('No se pudo conectar con el servidor')
            } else {
                // Otro tipo de error
                setError('Ocurrió un error inesperado')
            }
        }
    }

    return (
    <div className="login-container">
        <div className="form-container">
            <h2>Inicio de Sesión</h2>
            <form onSubmit={handleLogin}>
                <input
                    type="text"
                    placeholder="Registro Académico"
                    value={registro}
                    onChange={(e) => setRegistro(e.target.value)}
                    required
```

```
                />
                <input
                    type="password"
                    placeholder="Contraseña"
                    value={contrasena}
                    onChange={(e) => setContrasena(e.target.value)}
                    required
                />
                <button type="submit">Ingresar</button>
            </form>
            {error && <p style={{ color: 'red' }}>{error}</p>}
            <div className="links">
                <Link to="/register">Registrar usuario</Link>
                <Link to="/forgot-password">¿Olvidó contraseña?</Link>
            </div>
        </div>
    </div>
    )
}

export default Login
```

PantallaInicial.jsx

```
import React, { useState, useEffect } from 'react'
import axios from 'axios'
import { useNavigate } from 'react-router-dom'
import { API_URL } from './config'
import './PantallaInicial.css'

function PantallaInicial() {
  const [publicaciones, setPublicaciones] = useState([])
  const [todasPublicaciones, setTodasPublicaciones] = useState([])
  const [usuario, setUsuario] = useState(null)
  const [filtro, setFiltro] = useState('')
  const [valorBusqueda, setValorBusqueda] = useState('')
  const [registroBusqueda, setRegistroBusqueda] = useState('')
```

```
const [errorRegistro, setErrorRegistro] = useState('')
const navigate = useNavigate()

useEffect(() => {
  const usuarioLogueado = JSON.parse(localStorage.getItem('usuario'))
  if (!usuarioLogueado) {
    navigate('/')
    return
  }
  setUsuario(usuarioLogueado)

  const fetchPublicaciones = async () => {
    try {
      const response = await axios.get(`${API_URL}/publicaciones`)
      setPublicaciones(response.data)
      setTodasPublicaciones(response.data)
    } catch (error) {
      console.error('Error al cargar publicaciones:', error)
    }
  }

  fetchPublicaciones()
}, [navigate])

const handleLogout = () => {
  localStorage.removeItem('usuario')
  navigate('/')
}

const verComentarios = (id_publicacion) => {
  navigate(`/publicacion/${id_publicacion}`)
}

const aplicarFiltro = () => {
  if (!filtro) {
    setPublicaciones(todasPublicaciones)
    return
  }
```

```
    let filtradas = todasPublicaciones

    switch (filtro) {
      case 'curso':
        filtradas = todasPublicaciones.filter(pub =>
pub.nombre_curso?.toLowerCase().includes(valorBusqueda.toLowerCase()))
        break
      case 'catedratico':
        filtradas = todasPublicaciones.filter(pub =>
          `${pub.nombre_profesor}
${pub.apellido_profesor}`.toLowerCase().includes(valorBusqueda.toLowerCase())
        )
        break
      case 'nombreCurso':
        filtradas = todasPublicaciones.filter(pub =>
pub.nombre_curso?.toLowerCase() === valorBusqueda.toLowerCase())
        break
      case 'nombreCatedratico':
        filtradas = todasPublicaciones.filter(pub =>
          `${pub.nombre_profesor} ${pub.apellido_profesor}`.toLowerCase() ===
valorBusqueda.toLowerCase()
        )
        break
      default:
        filtradas = todasPublicaciones
    }

    setPublicaciones(filtradas)
  }

  const limpiarFiltros = () => {
    setFiltro('')
    setValorBusqueda('')
    setPublicaciones(todasPublicaciones)
  }

  const buscarUsuario = async () => {
    if (!registroBusqueda.trim()) {
      setErrorRegistro('Ingrese un registro académico para buscar')
```

```
      return
    }
    try {
      const response = await
axios.get(`${API_URL}/usuarios/registro/${registroBusqueda}`)
      navigate(`/perfil/${response.data.registro_academico}`)
    } catch (error) {
      console.error('Error al buscar usuario:', error)
      setErrorRegistro('Usuario no encontrado')
    }
  }

  return (
    <div className="pantalla-inicial">
      <header className="header">
        <h1>{usuario ? `Bienvenido, ${usuario.nombres}` : 'Bienvenido'}</h1>
        <nav>
          <ul>
            <li>Inicio</li>
            <li onClick={() => navigate('/crear-publicacion')}>Crear
Publicación</li>
            <li onClick={() =>
navigate(`/perfil/${usuario.registro_academico}`)}>Perfil</li>
            <li>Cursos Aprobados</li>
            <li onClick={handleLogout}>Salir</li>
          </ul>
        </nav>
      </header>

      {/* Buscador de usuario */}
      <div className="buscador-perfil">
        <input
          type="text"
          placeholder="Buscar usuario por registro académico"
          value={registroBusqueda}
          onChange={(e) => {
            setRegistroBusqueda(e.target.value)
            setErrorRegistro('')
          }}
```

```
        />
        <button onClick={buscarUsuario} className="btn-aplicar">Buscar</button>
        {errorRegistro && <p className="error-busqueda">{errorRegistro}</p>}
      </div>

      {/* Barra de filtros */}
      <div className="buscador-perfil">
        <select value={filtro} onChange={(e) => setFiltro(e.target.value)}>
          <option value="">-- Selecciona un filtro --</option>
          <option value="curso">Filtrar por Curso</option>
          <option value="catedratico">Filtrar por Catedrático</option>
          <option value="nombreCurso">Nombre exacto del Curso</option>
          <option value="nombreCatedratico">Nombre exacto del
Catedrático</option>
        </select>
        <input
          type="text"
          placeholder="Ingrese valor a buscar"
          value={valorBusqueda}
          onChange={(e) => setValorBusqueda(e.target.value)}
        />
        <button className="btn-aplicar" onClick={aplicarFiltro}>Aplicar</button>
        <button className="btn-limpiar" onClick={limpiarFiltros}>Limpiar
Filtros</button>
      </div>

      <main className="contenido">
        {publicaciones.length === 0 ? (
          <p>No hay publicaciones aún.</p>
        ) : (
          publicaciones.map((pub) => (
            <div key={pub.id_publicacion} className="publicacion-card">
              <div className="publicacion-header">
                <span className="usuario">{pub.nombres} {pub.apellidos}</span>
                <span className="fecha">{new
Date(pub.fecha_creacion).toLocaleString()}</span>
              </div>
              <div className="mensaje">{pub.mensaje}</div>
              <div className="publicacion-footer">
```

```
                Curso: {pub.nombre_curso || 'N/A'} | Profesor:
{pub.nombre_profesor ? `${pub.nombre_profesor} ${pub.apellido_profesor}` : 'N/A'}
              </div>
              <button
                className="boton-comentarios"
                onClick={() => verComentarios(pub.id_publicacion)}
              >
                Ver Comentarios
              </button>
            </div>
          ))
        )}
      </main>
    </div>
  )
}

export default PantallaInicial
```

Perfil.jsx

```
import React, { useState, useEffect } from 'react'
import { useParams, useNavigate } from 'react-router-dom'
import axios from 'axios'
import { API_URL } from './config'
import './Perfil.css'

function Perfil() {
  const { registro } = useParams() // se obtiene de la ruta /perfil/:registro
  const [usuario, setUsuario] = useState(null)
  const [editMode, setEditMode] = useState(false)
  const [formData, setFormData] = useState({
    registro_academico: '',
    nombres: '',
    apellidos: '',
    correo: '',
    contrasena: ''
```

```
  })
  const navigate = useNavigate()

  useEffect(() => {
    const fetchUsuario = async () => {
      try {
        const response = await
axios.get(`${API_URL}/usuarios/registro/${registro}`)
        setUsuario(response.data)
        setFormData({
          registro_academico: response.data.registro_academico,
          nombres: response.data.nombres,
          apellidos: response.data.apellidos,
          correo: response.data.correo,
          contrasena: '' // vacío para que el usuario ingrese una nueva si quiere
        })

        // Si el perfil es del usuario logueado → habilitar edición
        const usuarioLogueado = JSON.parse(localStorage.getItem('usuario'))
        if (usuarioLogueado && usuarioLogueado.registro_academico === registro) {
          setEditMode(true)
        }
      } catch (error) {
        console.error('Error al obtener usuario:', error)
        alert('No se encontró el usuario.')
        navigate('/inicio')
      }
    }

    fetchUsuario()
  }, [registro, navigate])

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value })
  }

  const handleUpdate = async () => {
    try {
      await axios.put(`${API_URL}/usuarios/${usuario.id_usuario}`, formData)
```

```
      alert('Perfil actualizado correctamente')
      navigate('/inicio')
    } catch (error) {
      console.error('Error al actualizar perfil:', error)
      alert('Hubo un error al actualizar el perfil')
    }
  }

  if (!usuario) {
    return <p className="perfil-cargando">Cargando perfil...</p>
  }

  return (
    <div className="perfil-container">
      <h2>Perfil de Usuario</h2>

      <div className="perfil-form">
        <label>Registro Académico</label>
        <input
          type="text"
          name="registro_academico"
          value={formData.registro_academico}
          disabled
        />

        <label>Nombres</label>
        <input
          type="text"
          name="nombres"
          value={formData.nombres}
          onChange={handleChange}
          disabled={!editMode}
        />

        <label>Apellidos</label>
        <input
          type="text"
          name="apellidos"
          value={formData.apellidos}
```

```
          onChange={handleChange}
          disabled={!editMode}
        />

        <label>Correo</label>
        <input
          type="email"
          name="correo"
          value={formData.correo}
          onChange={handleChange}
          disabled={!editMode}
        />

        {editMode && (
          <>
            <label>Nueva Contraseña</label>
            <input
              type="password"
              name="contrasena"
              value={formData.contrasena}
              onChange={handleChange}
              placeholder="Ingrese nueva contraseña"
            />
          </>
        )}

        <div className="perfil-actions">
          <button onClick={() => navigate('/inicio')} className="btn-regresar">
            Regresar
          </button>
          {editMode && (
            <button onClick={handleUpdate} className="btn-guardar">
              Guardar Cambios
            </button>
          )}
        </div>
      </div>
    </div>
  )
```

```
}

export default Perfil
```

Register.jsx

```
import React, { useState } from 'react'
import { useNavigate } from 'react-router-dom'
import axios from 'axios'
import { API_URL } from './config'

function Register() {
    const [registro, setRegistro] = useState('')
    const [nombres, setNombres] = useState('')
    const [apellidos, setApellidos] = useState('')
    const [correo, setCorreo] = useState('')
    const [contrasena, setContrasena] = useState('')
    const [mensaje, setMensaje] = useState('')
    const [error, setError] = useState('')
    const navigate = useNavigate()

    const handleRegister = async (e) => {
        e.preventDefault()
        setMensaje('')
        setError('')

        try {
            const response = await axios.post(`${API_URL}/usuarios`, {
                registro_academico: registro,
                nombres,
                apellidos,
                correo,
                contrasena
            })

            setMensaje(response.data.mensaje || 'Usuario registrado
correctamente')
```

```
            // Limpiar campos
            setRegistro('')
            setNombres('')
            setApellidos('')
            setCorreo('')
            setContrasena('')

            // Redirigir al login después de 2 segundos
            setTimeout(() => {
                navigate('/')
            }, 2000)
        } catch (err) {
            console.error('Error al registrar usuario:', err)
            if (err.response && err.response.data && err.response.data.error) {
                setError(err.response.data.error)
            } else {
                setError('No se pudo conectar con el servidor')
            }
        }
    }

    return (
    <div className="login-container">
        <div className="form-container">
            <h2>Registrar Usuario</h2>
            <form onSubmit={handleRegister}>
                <input
                    type="text"
                    placeholder="Registro Académico"
                    value={registro}
                    onChange={(e) => setRegistro(e.target.value)}
                    required
                />
                <input
                    type="text"
                    placeholder="Nombres"
                    value={nombres}
                    onChange={(e) => setNombres(e.target.value)}
                    required
```

```
                />
                <input
                    type="text"
                    placeholder="Apellidos"
                    value={apellidos}
                    onChange={(e) => setApellidos(e.target.value)}
                    required
                />
                <input
                    type="email"
                    placeholder="Correo electrónico"
                    value={correo}
                    onChange={(e) => setCorreo(e.target.value)}
                    required
                />
                <input
                    type="password"
                    placeholder="Contraseña"
                    value={contrasena}
                    onChange={(e) => setContrasena(e.target.value)}
                    required
                />
                <button type="submit">Registrar</button>
            </form>
            {mensaje && <p style={{ color: 'green' }}>{mensaje}</p>}
            {error && <p style={{ color: 'red' }}>{error}</p>}
        </div>
    </div>
    )
}

export default Register
```

CSS

App.css

```css
.login-container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
  margin: 0;
  padding: 0;
}

.App {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.form-container {
  background-color: #fff;
  padding: 30px 40px;
  border-radius: 8px;
  box-shadow: 0 0 15px rgba(0,0,0,0.1);
  width: 350px;
}

.form-container h2 {
  text-align: center;
  margin-bottom: 20px;
}

.form-container form {
```

```
  display: flex;
  flex-direction: column;
}

.form-container input {
  padding: 10px;
  margin-bottom: 15px;
  border-radius: 5px;
  border: 1px solid #ccc;
}

.form-container button {
  padding: 10px;
  border: none;
  border-radius: 5px;
  background-color: #007bff;
  color: #fff;
  font-weight: bold;
  cursor: pointer;
}

.form-container button:hover {
  background-color: #0056b3;
}

.form-container .links {
  display: flex;
  justify-content: space-between;
  margin-top: 10px;
}

.form-container .links a {
  text-decoration: none;
  color: #007bff;
}

.form-container .links a:hover {
  text-decoration: underline;
}
```

Comentarios.css

```css
/* Reset básico para este componente */
.comentarios-page * {
  box-sizing: border-box;
  font-family: Arial, sans-serif;
}

/* Contenedor principal */
.comentarios-page {
  width: 100%;
  min-height: 100vh;
  background-color: #f5f5f5;
  padding: 0;
  margin: 0;
}

/* Header */
.header-comentarios {
  width: 100%;
  background-color: #4a90e2;
  color: white;
  padding: 20px 40px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  position: sticky;
  top: 0;
  z-index: 100;
  box-shadow: 0 3px 6px rgba(0,0,0,0.1);
}

.header-comentarios h2 {
  font-size: 1.5rem;
}
```

```css
.boton-regresar {
  padding: 8px 16px;
  background-color: white;
  color: #4a90e2;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-weight: bold;
  transition: background-color 0.2s;
}

.boton-regresar:hover {
  background-color: #357ab8;
  color: white;
}

/* Contenido */
.contenido-comentarios {
  max-width: 900px;
  margin: 30px auto;
  padding: 0 20px;
}

/* Tarjetas de comentarios */
.comentario-card {
  background-color: white;
  padding: 15px 20px;
  border-radius: 10px;
  margin-bottom: 15px;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}

.comentario-header {
  display: flex;
  justify-content: space-between;
  margin-bottom: 5px;
  font-size: 0.85rem;
  color: #555;
}
```

```css
.mensaje {
  font-size: 0.95rem;
  color: #333;
}

/* Formulario */
.form-comentario {
  display: flex;
  flex-direction: column;
  margin-top: 20px;
}

.form-comentario textarea {
  resize: vertical;
  min-height: 60px;
  padding: 10px;
  border-radius: 6px;
  border: 1px solid #ccc;
  margin-bottom: 10px;
}

.form-comentario button {
  width: fit-content;
  padding: 8px 16px;
  border-radius: 6px;
  border: none;
  background-color: #4a90e2;
  color: white;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.2s;
}

.form-comentario button:hover {
  background-color: #357ab8;
}

/* Mensajes */
```

```css
.error {
  color: red;
  margin-bottom: 10px;
}

.exito {
  color: green;
  margin-bottom: 10px;
}
```

CrearPublicacion.css

```css
.crear-publicacion {
  background-color: white;
  padding: 20px;
  margin-bottom: 20px;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}

.crear-publicacion h3 {
  margin-bottom: 15px;
}

.crear-publicacion form {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.crear-publicacion label {
  font-weight: bold;
}

.crear-publicacion select,
.crear-publicacion textarea {
  padding: 8px;
```

```
  border-radius: 6px;
  border: 1px solid #ccc;
  width: 100%;
}

.crear-publicacion textarea {
  resize: vertical;
  min-height: 80px;
}

.crear-publicacion button {
  padding: 10px;
  background-color: #4a90e2;
  color: white;
  font-weight: bold;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  transition: background-color 0.2s;
}

.crear-publicacion button:hover {
  background-color: #357ab8;
}

.error {
  color: red;
  margin-bottom: 10px;
}

.exito {
  color: green;
  margin-bottom: 10px;
}

.crear-publicacion .botones {
  margin-top: 15px;
  display: flex;
  gap: 10px;
```

```
}

.crear-publicacion button {
  padding: 10px 20px;
  border-radius: 6px;
  border: none;
  cursor: pointer;
  background-color: #4a90e2;
  color: white;
  font-weight: bold;
  transition: background-color 0.2s;
}

.crear-publicacion button:hover {
  background-color: #357ab8;
}

.crear-publicacion button[type="button"] {
  background-color: #888;
}

.crear-publicacion button[type="button"]:hover {
  background-color: #555;
}
```

index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
```

```
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

Login.css

```
/* src/Login.css */
.login-container {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 8px;
    background-color: #f5f5f5;
    text-align: center;
}

.login-container input {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border-radius: 4px;
    border: 1px solid #ccc;
}

.login-container button {
    width: 100%;
    padding: 10px;
    border: none;
    background-color: #007bff;
    color: white;
    font-weight: bold;
    border-radius: 4px;
    cursor: pointer;
}

.login-container button:hover {
```

```
    background-color: #0056b3;
}

.login-container p {
    margin-top: 15px;
    color: red;
}
```

PantallaInicial.css

```
/* Reset básico */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: Arial, sans-serif;
  background-color: #f2f4f7;
}

/* Pantalla Inicial */
.pantalla-inicial {
  width: 100%;
  min-height: 100vh;
}

/* Header */
.header {
  width: 100%;
  background-color: #4a90e2;
  color: white;
  padding: 20px 40px;
  display: flex;
  justify-content: space-between;
  align-items: center;
```

```
  position: sticky;
  top: 0;
  z-index: 100;
  box-shadow: 0 3px 6px rgba(0,0,0,0.1);
}

.header h1 {
  font-size: 1.8rem;
}

.header nav ul {
  list-style: none;
  display: flex;
  gap: 20px;
}

.header nav ul li {
  cursor: pointer;
  padding: 8px 15px;
  border-radius: 5px;
  transition: background-color 0.3s;
  background-color: #fff;
  color: #4a90e2;
  font-weight: 500;
}

.header nav ul li:hover {
  background-color: #357ab8;
  color: #fff;
}

/* Contenido */
.contenido {
  max-width: 900px;
  margin: 30px auto;
  padding: 0 20px;
}

/* Publicaciones */
```

```css
.publicacion-card {
  background-color: white;
  padding: 20px;
  margin-bottom: 20px;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
  transition: transform 0.2s;
}

.publicacion-card:hover {
  transform: translateY(-3px);
}

.publicacion-header {
  display: flex;
  justify-content: space-between;
  margin-bottom: 10px;
  font-size: 14px;
  color: #555;
}

/* Botón Ver Comentarios */
.boton-comentarios {
  display: inline-block;
  padding: 8px 14px;
  margin-top: 10px;
  background-color: #4a90e2;
  color: white;
  border: none;
  border-radius: 6px;
  font-size: 14px;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.3s, transform 0.2s;
}

.boton-comentarios:hover {
  background-color: #357ab8;
  transform: scale(1.05);
```

```css
}

.boton-comentarios:active {
  background-color: #2a5d91;
  transform: scale(0.98);
}


.mensaje {
  font-size: 1rem;
  margin-bottom: 10px;
}

.publicacion-footer {
  margin-top: 10px;
  font-size: 13px;
  color: #888;
}

.buscador-perfil {
  margin: 20px auto;
  text-align: center;
}

.buscador-perfil form {
  display: flex;
  justify-content: center;
  gap: 10px;
}

.buscador-perfil input {
  padding: 8px 12px;
  border: 1px solid #ccc;
  border-radius: 6px;
  width: 250px;
}

.buscador-perfil button {
  padding: 8px 16px;
```

```
  border: none;
  border-radius: 6px;
  background: #007bff;
  color: white;
  cursor: pointer;
  font-weight: bold;
}

.buscador-perfil button:hover {
  background: #0056b3;
}

.error-busqueda {
  margin-top: 10px;
  color: red;
  font-size: 14px;
}

.buscador-perfil {
  margin: 20px auto;
  text-align: center;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  gap: 10px;
}

.buscador-perfil select,
.buscador-perfil input {
  padding: 8px 12px;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 14px;
}

.buscador-perfil .btn-aplicar {
  padding: 8px 16px;
  border: none;
  border-radius: 6px;
```

```css
  background: #007bff;
  color: white;
  cursor: pointer;
  font-weight: bold;
  transition: background 0.3s;
}

.buscador-perfil .btn-aplicar:hover {
  background: #0056b3;
}

.buscador-perfil .btn-limpiar {
  padding: 8px 16px;
  border: none;
  border-radius: 6px;
  background: #6c757d;
  color: white;
  cursor: pointer;
  font-weight: bold;
  transition: background 0.3s;
}

.buscador-perfil .btn-limpiar:hover {
  background: #5a6268;
}



/* Responsive */
@media (max-width: 600px) {
  .header {
    flex-direction: column;
    align-items: flex-start;
  }

  .header nav ul {
    flex-direction: column;
    gap: 10px;
    margin-top: 10px;
```

```
  }

  .contenido {
    padding: 0 10px;
  }
}
```

Perfil.css

```
.perfil-container {
  max-width: 600px;
  margin: 30px auto;
  padding: 20px;
  background: #f8f9fa;
  border-radius: 12px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
  font-family: Arial, sans-serif;
}

.perfil-container h2 {
  text-align: center;
  margin-bottom: 20px;
  color: #333;
}

.perfil-form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

.perfil-form label {
  font-weight: bold;
  color: #555;
}

.perfil-form input {
```

```
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 8px;
  font-size: 14px;
}

.perfil-form input:disabled {
  background-color: #e9ecef;
  cursor: not-allowed;
}

.perfil-actions {
  display: flex;
  justify-content: space-between;
  margin-top: 20px;
}

.btn-regresar, .btn-guardar {
  padding: 10px 16px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
}

.btn-regresar {
  background: #6c757d;
  color: white;
}

.btn-regresar:hover {
  background: #5a6268;
}

.btn-guardar {
  background: #28a745;
  color: white;
}
```

```css
.btn-guardar:hover {
  background: #218838;
}

.perfil-cargando {
  text-align: center;
  margin-top: 40px;
  font-size: 18px;
  color: #666;
}
```