

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

PRACTICAS INICIALES

CATEDRÁTICO: INGA. FLORIZA FELIPA ÁVILA DE MEDINILLA



## **MANUAL TÉCNICO**

CHRISTIAN FERNANDO BOROR ROMERO

CARNÉ: 202307765

SECCIÓN: C

GUATEMALA, 19 DE SEPTIEMBRE DEL 2,024

# ÍNDICE

## Contenido

ÍNDICE .....	1
INTRODUCCIÓN .....	2
OBJETIVOS .....	3
1. GENERAL.....	3
2. ESPECÍFICOS.....	3
ALCANCES DEL SISTEMA.....	4
ESPECIFICACIÓN TÉCNICA.....	4
• REQUISITOS DE HARDWARE .....	4
• REQUISITOS DE SOFTWARE.....	4
LÓGICA DEL PROGRAMA .....	5
ANEXOS.....	21

## **INTRODUCCIÓN**

Este manual técnico describe el desarrollo y las funcionalidades de una aplicación web que permite a los estudiantes compartir opiniones sobre catedráticos y cursos de manera eficiente y segura. La aplicación está diseñada para facilitar la interacción entre los estudiantes, brindando un espacio donde puedan realizar publicaciones, comentarios y visualizar las evaluaciones de diferentes cursos y docentes, con el fin de generar un intercambio de experiencias y retroalimentación.

El sistema ha sido implementado utilizando tecnologías modernas, como React para el desarrollo del frontend, Node.js y Express para el backend, y MySQL como sistema de gestión de bases de datos.

## **OBJETIVOS**

### **1. GENERAL**

- 1.1. Proporcionar una guía técnica clara y detallada sobre el funcionamiento y la estructura del sistema desarrollado, para facilitar su mantenimiento y futuras mejoras.

### **2. ESPECÍFICOS**

- 2.1. Objetivo 1: Describir la arquitectura del sistema y las tecnologías utilizadas en su desarrollo.
- 2.2. Objetivo 2: Ofrecer información técnica que permita a los desarrolladores entender el código y las bases de datos asociadas al proyecto.
- 2.3. Objetivo 3: Mostrar capturas para una mayor referencia de los elementos utilizados en el programa.

## **ALCANCES DEL SISTEMA**

### **ESPECIFICACIÓN TÉCNICA**

#### **● REQUISITOS DE HARDWARE**

Minimo:

Ram: 128MB

Disco duro: 4Gb

Resolución de pantalla: 800 x 600

Procesador Intel Pentium III a 800 MHz.

Recomendable:

Ram: 256MB

Disco duro: 6GB

Resolucion de pantalla: 1024 x 768

#### **● REQUISITOS DE SOFTWARE**

Jav Oracle Version 8 (Recomendado)

#### **Windows**

Windows 7 en adelante

Windows Vista SP2

#### **Mac Os X**

Mac con Intel que ejecuta Mac OS X 10.7.3+, 10.8.3+, 10.9+

Privilegios de administrador para la instalación

Explorador de 64 bits

#### **Linux**

Oracle Linux 5.5+

Oracle Linux 6.x (32 bits), 6.x (64 bits)<sup>3</sup>

Oracle Linux 7.x (64 bits)<sup>3</sup> (7u67 y superiores)

Suse Linux Enterprise Server 10 SP2, 11.x

Exploradores: Firefox 3.6 y posterior

# LÓGICA DEL PROGRAMA

## BACKEND

**Accesos.js:** Utilizado para implementar la funcionalidad el inicio de sesión y el registro de un usuario.

```
const db = require('../config/db');
const jwt = require('jsonwebtoken');
const secretKey = 'clave'

// Maneja el registro de usuario
exports.registro = (req, res) => {
  const { name, lname, password, email } = req.body;
  const sql = 'INSERT INTO usuarios (name, lname, password, email) VALUES (?, ?, ?, ?)';
  db.query(sql, [name, lname, password, email], (err, results) => {
    if (err) {
      console.error('Error al registrar el usuario:', err);
      return res.status(500).json({ mensaje: 'Error al registrar el usuario' });
    }
    const carnet = results.insertId;
    res.status(201).json({ mensaje: `Usuario registrado exitosamente con carnet: ${carnet}` });
  });
};

// Maneja el inicio de sesión
exports.login = (req, res) => {
  const { carnet, password } = req.body;

  // Verificar usuario en la base de datos (esto ya lo tienes)
  const sql = 'SELECT * FROM usuarios WHERE carnet = ? AND password = ?';
  db.query(sql, [carnet, password], (err, results) => {
    if (err) {
      return res.status(500).json({ mensaje: 'Error en el servidor' });
    }
    if (results.length > 0) {
      // Generar el token JWT con el carnet del usuario
      const token = jwt.sign({ carnet }, secretKey, { expiresIn: '1h' });
      return res.status(200).json({ mensaje: 'Login exitoso', token });
    } else {
      return res.status(401).json({ mensaje: 'Credenciales incorrectas' });
    }
  });
};
```

**Comentarios.js:** Utilizado para la funcionalidad de crear un comentario y obtener los comentarios dependiendo del id de la publicación de la cual se desea ver los comentarios.

```
const db = require('../config/db');
const jwt = require('jsonwebtoken');
const secretKey = 'clave';

// Función para crear un comentario
exports.crearComentario = (req, res) => {
  const token = req.headers.authorization && req.headers.authorization.split(' ')[1];

  if (!token) {
    return res.status(401).json({ mensaje: 'Token no proporcionado' });
  }

  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(403).json({ mensaje: 'Token inválido' });
    }

    const carnet = decoded.carnet; // Obtener el carnet desde el token
    const { id_publicacion, comentario } = req.body;

    if (!id_publicacion || !comentario) {
      return res.status(400).json({ mensaje: 'ID de la publicación y comentario son obligatorios' });
    }

    const sql = 'INSERT INTO comentarios (id_publicacion, carnet, comentario) VALUES (?, ?, ?)';
    db.query(sql, [id_publicacion, carnet, comentario], (err, results) => {
      if (err) {
        console.error('Error al crear el comentario:', err);
        return res.status(500).json({ mensaje: 'Error al crear el comentario' });
      }

      res.status(201).json({ mensaje: 'Comentario creado exitosamente' });
    });
  });
};

// Función para obtener los comentarios de una publicación
exports.obtenerComentarios = (req, res) => {
  const { id_publicacion } = req.params;

  if (!id_publicacion) {
    return res.status(400).json({ mensaje: 'ID de la publicación es obligatorio' });
  }

  const sql = `
    SELECT c.comentario, c.fecha_creacion, c.carnet
    FROM comentarios c
    WHERE c.id_publicacion = ?
    ORDER BY c.fecha_creacion ASC
  `;

  db.query(sql, [id_publicacion], (err, results) => {
    if (err) {
      console.error('Error al obtener los comentarios:', err);
      return res.status(500).json({ mensaje: 'Error al obtener los comentarios' });
    }

    res.json({ comentarios: results });
  });
};
```

**Cursos.js:** Se aplica la lógica relacionada para los cursos, como obtener los cursos aprobados de un usuario que se busca en el frontend, obtener los cursos aprobados del usuario logeado y la funcionalidad para agregar un curso aprobado para el usuario en su perfil.

```
1  const db = require('../config/db');
2  const jwt = require('jsonwebtoken');
3  const secretKey = 'clave'
4
5  exports.obtenerCursosAprobados = (req, res) => {
6    const carnet = parseInt(req.params.carnet);
7
8    if (isNaN(carnet)) {
9      return res.status(400).json({ mensaje: 'Carnet inválido' });
10   }
11
12   const sql = `
13     SELECT c.id_curso, c.nombre_curso, ca.fecha_aprobacion
14     FROM cursos_aprobados ca
15     JOIN cursos c ON ca.id_curso = c.id_curso
16     WHERE ca.carnet = ?
17   `;
18
19   db.query(sql, [carnet], (err, results) => {
20     if (err) {
21       console.error('Error al obtener cursos aprobados:', err);
22       return res.status(500).json({ mensaje: 'Error al obtener cursos aprobados' });
23     }
24
25     const totalCreditoSql = `
26       SELECT SUM(c.creditos) as total_creditos
27       FROM cursos_aprobados ca
28       JOIN cursos c ON ca.id_curso = c.id_curso
29       WHERE ca.carnet = ?
30     `;
31
32     db.query(totalCreditoSql, [carnet], (err, creditoResults) => {
33       if (err) {
34         console.error('Error al obtener créditos totales:', err);
35         return res.status(500).json({ mensaje: 'Error al obtener créditos totales' });
36       }
37
38       res.status(200).json({
39         cursos: results,
40         totalCredito: creditoResults[0].total_creditos || 0
41       });
42     });
43   });
44 }
```

```
46 exports.agregarCursoAprobado = (req, res) => {
47   const { id_curso } = req.body;
48   const token = req.headers.authorization && req.headers.authorization.split(' ')[1];
49
50   if (!token) {
51     return res.status(401).json({ mensaje: 'Token no proporcionado' });
52   }
53
54   jwt.verify(token, secretKey, (err, decoded) => {
55     if (err) {
56       return res.status(403).json({ mensaje: 'Token inválido' });
57     }
58
59     const carnet = decoded.carnet; // Obtener el carnet desde el token
60     const fecha_aprobacion = new Date(); // Fecha actual
61
62     const sql = `
63       INSERT INTO cursos_aprobados (carnet, id_curso, fecha_aprobacion)
64       VALUES (?, ?, ?)
65     `;
66
67     db.query(sql, [carnet, id_curso, fecha_aprobacion], (err, results) => {
68       if (err) {
69         console.error('Error al agregar curso aprobado:', err);
70         return res.status(500).json({ mensaje: 'Error al agregar curso aprobado' });
71       }
72       res.json({ mensaje: 'Curso aprobado agregado exitosamente' });
73     });
74   });
75 }
```



```

77 // Función para obtener cursos aprobados y créditos totales del usuario
78 exports.obtenerCursosAprobadosYCreditos = (req, res) => {
79   const token = req.headers.authorization && req.headers.authorization.split(' ')[1];
80
81   if (!token) {
82     return res.status(401).json({ mensaje: 'Token no proporcionado' });
83   }
84
85   jwt.verify(token, secretKey, (err, decoded) => {
86     if (err) {
87       return res.status(403).json({ mensaje: 'Token inválido' });
88     }
89
90     const carnet = decoded.carnet; // Obtener el carnet desde el token
91
92     const sqlCursos = `
93       SELECT c.id_curso, c.nombre_curso, ca.fecha_aprobacion
94       FROM cursos c
95       JOIN cursos_aprobados ca ON c.id_curso = ca.id_curso
96       WHERE ca.carnet = ?
97       ORDER BY ca.fecha_aprobacion ASC
98     `;
99
100    db.query(sqlCursos, [carnet], (err, cursos) => {
101      if (err) {
102        console.error('Error al obtener cursos aprobados:', err);
103        return res.status(500).json({ mensaje: 'Error al obtener cursos aprobados' });
104      }
105
106      const sqlCreditos = `
107        SELECT SUM(c.creditos) AS total_creditos
108        FROM cursos c
109        JOIN cursos_aprobados ca ON c.id_curso = ca.id_curso
110        WHERE ca.carnet = ?
111      `;
112
113      db.query(sqlCreditos, [carnet], (err, resultados) => {
114        if (err) {
115          console.error('Error al obtener créditos totales:', err);
116          return res.status(500).json({ mensaje: 'Error al obtener créditos totales' });
117        }
118
119        res.json({
120          cursos: cursos,
121          total_creditos: resultados[0].total_creditos
122        });
123      });
124    });
125  });
126 }

```

**Gets.js:** funciones para obtener los cursos y catedráticos para las combobox cuando se desea agregar un comentario.

```

1  const db = require('../config/db');
2
3  // Obtener todos los cursos
4  exports.getCursos = (req, res) => {
5    const sql = 'SELECT * FROM cursos';
6    db.query(sql, (err, results) => {
7      if (err) {
8        console.error('Error al obtener cursos:', err);
9        return res.status(500).json({ mensaje: 'Error al obtener cursos' });
10     }
11     res.status(200).json({ cursos: results });
12   });
13 }
14
15 // Obtener todos los catedráticos
16 exports.getCatedraticos = (req, res) => {
17   const sql = 'SELECT * FROM catedratico';
18   db.query(sql, (err, results) => {
19     if (err) {
20       console.error('Error al obtener catedráticos:', err);
21       return res.status(500).json({ mensaje: 'Error al obtener catedráticos' });
22     }
23     res.status(200).json({ catedraticos: results });
24   });
25 }

```

**Publicaciones.js:** Contiene las funcionalidades relacionadas a las publicaciones como crear una publicación, hacer una consulta a sql para obtener todas las publicaciones para mostrarlas en la ventana principal con sus respectivas filtraciones.

```
1  const db = require('../config/db');
2  const jwt = require('jsonwebtoken');
3  const secretKey = 'clave'
4
5
6  exports.crearPublicacion = (req, res) => {
7    console.log('Encabezados:', req.headers);
8    const token = req.headers.authorization && req.headers.authorization.split(' ')[1];
9
10   console.log('Token:', token); // Verifica el token recibido
11
12   if (!token) {
13     return res.status(401).json({ mensaje: 'Token no proporcionado' });
14   }
15
16   //TOKEN
17   jwt.verify(token, secretKey, (err, decoded) => {
18     if (err) {
19       return res.status(403).json({ mensaje: 'Token inválido' });
20     }
21
22     const carnet = decoded.carnet; // Obtener el carnet desde el token
23     const { tipo_publicación, id_curso, id_catedratico, mensaje } = req.body;
24
25     /*if (!tipo_publicación || !mensaje) {
26       return res.status(400).json({ mensaje: 'Tipo de publicación y mensaje son obligatorios ' });
27     }*/
28
29     /*if (!['Curso', 'Catedratico'].includes(tipo_publicación)) {
30       return res.status(400).json({ mensaje: 'Tipo de publicación inválido' });
31     }*/
32
33     const idCurso = id_curso ? Number(id_curso) : null;
34     const idCatedratico = id_catedratico ? Number(id_catedratico) : null;
35
36     const sql = 'INSERT INTO publicaciones (carnet, tipo_publicación, id_curso, id_catedratico, mensaje) VALUES (?, ?, ?, ?, ?)';
37     db.query(sql, [carnet, tipo_publicación, idCurso || null, idCatedratico || null, mensaje], (err, results) => {
38       if (err) {
39         console.error('Error al crear la publicación:', err);
40         return res.status(500).json({ mensaje: 'Error al crear la publicación' });
41       }
42
43       const id_publicacion = results.insertId;
44       res.status(201).json({ mensaje: 'Publicación creada exitosamente con ID: ${id_publicacion}' });
45     });
46   });
47 }
```

```

49 exports.obtenerPublicaciones = (req, res) => {
50   const { filtroCurso, filtroCatedratico, nombreCurso, nombreCatedratico } = req.query;
51
52   let sql = `
53     SELECT p.id_publicacion, p.carnet, p.tipo_publicación, p.id_curso, p.id_catedratico, p.mensaje, p.fecha_creacion, c.nombre_curso, cat.nombre_catedratico
54     FROM publicaciones p
55     LEFT JOIN cursos c ON p.id_curso = c.id_curso
56     LEFT JOIN catedratico cat ON p.id_catedratico = cat.id_catedratico
57     WHERE 1=1
58   `;
59
60   let params = [];
61
62   // Filtrar por curso (tiene que recibir id)
63   if (filtroCurso) {
64     const idCurso = parseInt(filtroCurso, 10); // Parsear a entero
65     sql += " AND p.id_curso = ?";
66     params.push(idCurso);
67   }
68
69   // Filtrar por catedrático (tiene que recibir id)
70   if (filtroCatedratico) {
71     const idCatedratico = parseInt(filtroCatedratico, 10); // Parsear a entero
72     sql += " AND p.id_catedratico = ?";
73     params.push(idCatedratico);
74   }
75
76   // Filtrar por nombre del curso
77   if (nombreCurso) {
78     sql += " AND c.nombre_curso LIKE ?";
79     params.push(`%${nombreCurso}%`);
80   }
81
82   // Filtrar por nombre del catedrático
83   if (nombreCatedratico) {
84     sql += " AND cat.nombre_catedratico LIKE ?";
85     params.push(`%${nombreCatedratico}%`);
86   }
87
88   // Ordenar por fecha de creación (más recientes primero)
89   sql += " ORDER BY p.fecha_creacion DESC";
90
91   db.query(sql, params, (err, results) => {
92     if (err) {
93       console.error('Error al obtener publicaciones:', err);
94       return res.status(500).json({ mensaje: 'Error al obtener las publicaciones' });
95     }
96
97     res.json({ publicaciones: results });
98   });
99 };

```

**Usuarios.js:** Incluye las funcionalidades para buscar un usuario según su carnet y obtener un usuario para mostrar su información donde sea requerido.

```

1  const db = require('../config/db');
2  const jwt = require('jsonwebtoken');
3  const secretKey = 'clave';
4
5  exports.buscarUsuario = (req, res) => {
6    const carnet = parseInt(req.params.carnet);
7
8    if (isNaN(carnet)) {
9      return res.status(400).json({ mensaje: 'Carnet inválido' });
10   }
11
12   const sql = `
13     SELECT * FROM usuarios WHERE carnet = ?
14   `;
15
16   db.query(sql, [carnet], (err, results) => {
17     if (err) {
18       console.error('Error al buscar el usuario:', err);
19       return res.status(500).json({ mensaje: 'Error al buscar el usuario' });
20     }
21
22     if (results.length === 0) {
23       return res.status(404).json({ mensaje: 'Usuario no encontrado' });
24     }
25
26     res.status(200).json({ usuario: results[0] });
27   });
28 };

```

```

exports.obtenerUsuario = (req, res) =>{
  const token = req.headers.authorization && req.headers.authorization.split(' ')[1];

  if (!token) {
    return res.status(401).json({ mensaje: 'Token no proporcionado' });
  }

  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(403).json({ mensaje: 'Token inválido' });
    }

    const carnet = decoded.carnet; // Obtener el carnet desde el token
    console.log(carnet)
    const sql = `
      SELECT u.carnet, u.name, u.lname, u.email
      FROM usuarios u
      WHERE u.carnet = ?
    `;

    db.query(sql, [carnet], (err, results) => {
      if (err) {
        console.error('Error al obtener el usuario:', err);
        return res.status(500).json({ mensaje: 'Error al obtener el usuario' });
      }
      res.json({ datos: results });
    });
  });
};

```

## Rutas del backend

```

const express = require('express')
const router = express.Router()
const {login, registro,} = require('../controllers/accesos')
const {crearPublicacion, obtenerPublicaciones} = require('../controllers/publicaciones')
const { getCursos, getCatedraticos } = require('../controllers/gets');
const { buscarUsuario, obtenerUsuario } = require('../controllers/usuarios')
const { obtenerCursosAprobados, agregarCursoAprobado, obtenerCursosAprobadosYCreditos } = require('../controllers/cursos')
const { crearComentario, obtenerComentarios } = require('../controllers/comentarios')

//GET
router.get('/getCursos', getCursos);
router.get('/getCatedraticos', getCatedraticos);
router.get('/getPublicaciones', obtenerPublicaciones)
router.get('/buscarUsuario/:carnet', buscarUsuario)
router.get('/cursosAprobados/:carnet', obtenerCursosAprobados)
router.get('/obtenerComentarios/:id_publicacion', obtenerComentarios)
router.get('/obtenerUsuario', obtenerUsuario)
router.get('/cursosAprobadosYCreditos', obtenerCursosAprobadosYCreditos);

//Post
router.post('/registro', registro)
router.post('/login', login)
router.post('/crearPubli', crearPublicacion)
router.post('/crearComentario', crearComentario)
router.post('/agregarCursoAprobado', agregarCursoAprobado)

```

## App.js

```
1  const express = require('express')
2  const cors = require('cors')
3
4  const MyApp = express();
5
6  const bodyParser = require('body-parser');
7
8  const corsOptions = {
9    |   origin: 'http://localhost:3000'
10  | };
11
12  MyApp.use(cors(corsOptions));
13  MyApp.use(bodyParser.json({ limit: '15mb' }));
14
15  MyApp.use(express.json())
16  MyApp.use(cors())
17
18  //-----
19  const Router = require('./routers/routes')
20  MyApp.use(Router)
21  //-----
22
23  const port = 5000;
24
25  MyApp.listen(port, () =>{
26    |   console.log(`The API is listening at http://localhost:${port}`);
27  | });
```

## FRONTEND

**Navbar.js:** Barra utilizada en la pestaña principal para poder acceder a las demás funcionalidades de la aplicación como crear publicación, ver perfil, buscar usuario o cerrar sesión.

```
1 import React, { useState } from "react";
2 import '../Styles/Styles.css'
3 import { useNavigate, Link } from "react-router-dom";
4
5 function NavBar() {
6   const [carnetBuscado, setCarnetBuscado] = useState('');
7   const navigate = useNavigate();
8
9   const handleLogout = () => {
10     localStorage.removeItem('token');
11     navigate('/login');
12   };
13
14   const handleBuscarUsuario = async (e) => {
15     e.preventDefault();
16     try {
17       const response = await fetch('http://localhost:5000/buscarUsuario/${carnetBuscado}');
18       if (response.ok) {
19         const result = await response.json();
20         navigate('/perfil/${carnetBuscado}');
21       } else {
22         alert('Usuario no encontrado');
23       }
24     } catch (error) {
25       console.error('Error al buscar usuario:', error);
26       alert('Error al buscar usuario');
27     }
28   };
29 }
```

```
30
31 return (
32   <div className="container-navbar">
33     <div className="left-container-navbar">
34       <h1 style={{ color: "white" }}>USAC</h1>
35       <ul className="link-list">
36         <li className="link-list-item">
37           <Link className="link" to="/crearP"> Crear publicación</Link>
38         </li>
39       </ul>
40     </div>
41     <div className="right-container-navbar">
42       <form onSubmit={handleBuscarUsuario} className="form-inline">
43         <input
44           type="text"
45           className="form-control mr-sm-2"
46           placeholder="Buscar usuario por carnet"
47           value={carnetBuscado}
48           onChange={(e) => setCarnetBuscado(e.target.value)}
49           required
50         />
51         <button className="btn btn-outline-info my-2 my-sm-0" type="submit">
52           Buscar
53         </button>
54       </form>
55       <button className="btn btn-outline-info" onClick={handleLogout}>
56         Logout
57       </button>
58       <h1></h1>
59       <Link to="/perfil">
60         <button className="btn btn-outline-info">
61           Perfil
62         </button>
63       </Link>
64     </div>
65   </div>
66 )
67
68 export default NavBar;
```

## Home.js: Ventana principal que se abre cuando se inicia sesión

```
import './Styles/Styles.css';
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import NavBar from './NavBar';

function Home() {
  const [publicaciones, setPublicaciones] = useState([]);
  const [filtroCurso, setFiltroCurso] = useState('');
  const [filtroCatedratico, setFiltroCatedratico] = useState('');
  const [nombreCurso, setNombreCurso] = useState('');
  const [nombreCatedratico, setNombreCatedratico] = useState('');
  const [error, setError] = useState('');
  const navigate = useNavigate();

  useEffect(() => {
    obtenerPublicaciones();
  }, []);

  // Función para obtener las publicaciones desde el backend
  const obtenerPublicaciones = () => {
    let query = `?filtroCurso=${filtroCurso}&filtroCatedratico=${filtroCatedratico}&nombreCurso=${nombreCurso}&nombreCatedratico=${nombreCatedratico}`;

    fetch('http://localhost:5000/getPublicaciones${query}')
      .then(res => res.json())
      .then(data => {
        setPublicaciones(data.publicaciones);
      })
      .catch(err => {
        console.error('Error al obtener publicaciones:', err);
        setError('Error al obtener las publicaciones');
      });
  };

  // Función para aplicar filtros
  const aplicarFiltros = (e) => {
    e.preventDefault();
    obtenerPublicaciones();
  };
}
```

```
40   return (
41     <div>
42       <NavBar />
43
44       <div className="container mt-5">
45         <h2>Publicaciones</h2>
46
47         <form onSubmit={aplicarFiltros} className="mb-4">
48           <div className="form-group">
49             <label>Filtrar por Curso</label>
50             <input
51               type="text"
52               className="form-control"
53               value={filtroCurso}
54               onChange={(e) => setFiltroCurso(e.target.value)}
55             />
56           </div>
57           <div className="form-group">
58             <label>Filtrar por Catedrático</label>
59             <input
60               type="text"
61               className="form-control"
62               value={filtroCatedratico}
63               onChange={(e) => setFiltroCatedratico(e.target.value)}
64             />
65           </div>
66           <div className="form-group">
67             <label>Buscar por nombre del curso</label>
68             <input
69               type="text"
70               className="form-control"
71               value={nombreCurso}
72               onChange={(e) => setNombreCurso(e.target.value)}
73             />
74           </div>
75           <div className="form-group">
76             <label>Buscar por nombre del catedrático</label>
77             <input
78               type="text"
79               className="form-control"
80               value={nombreCatedratico}
81               onChange={(e) => setNombreCatedratico(e.target.value)}
82             />
83           </div>
84           <button type="submit" className="btn btn-primary">Aplicar Filtros</button>
85         </form>
86
87         {error && <p className="text-danger">{error}</p>}
```

```

88
89     <div className="list-group">
90       {publicaciones.map((pub) => (
91         <div key={pub.id_publicacion} className="list-group-item">
92           <h5>{pub.tipo_publicación === 'Curso' ? pub.nombre_curso : pub.nombre_catedratico}</h5>
93           <p>Publicación de: {pub.carnet}</p>
94           <p>{pub.mensaje}</p>
95           <small className="text-muted">Fecha de creación: {new Date(pub.fecha_creacion).toLocaleString()}</small>
96           <button
97             onClick={() => navigate("/comentarios/${pub.id_publicacion}")}
98             className="btn btn-info mt-2"
99           >
100             Ver Comentarios
101           </button>
102         </div>
103       ))}
104     </div>
105   </div>
106 </div>
107 );
108 }
109
110 export default Home;
111

```

**Login.js:** Funcionalidad para que el usuario acceda a la red social con sus credenciales.

```

1  import React, { useState } from "react";
2  import { useNavigate, Link } from "react-router-dom";
3  import '../Styles/Styles.css';
4
5  function Login() {
6    const [carnet, setCarnet] = useState('');
7    const [password, setPassword] = useState('');
8
9    const Navegador = useNavigate();
10
11    const handleSubmit = (event) => {
12      event.preventDefault();
13
14      // Verificación de datos de administrador
15      if (carnet === "282387765" && password === "admin") {
16
17        alert("Inicio de sesión como administrador exitoso");
18        Navegador('/admin'); // Redirige a la página de admin (PENDIENTE)
19        return;
20      }
21
22      // Datos del usuario
23      const dataJson = {
24        carnet: carnet,
25        password: password
26      };
27
28      fetch('http://localhost:5000/login', {
29        method: 'POST',
30        body: JSON.stringify(dataJson),
31        headers: {
32          'Content-Type': 'application/json'
33        }
34      })
35      .then((response) => {
36        if (!response.ok) {
37          throw new Error("Error: ${response.statusText}");
38        }
39        return response.json();
40      })
41      .then((data) => {
42        if (data.mensaje === 'Login exitoso') {
43          localStorage.setItem('token', data.token);
44          console.log(data.datos);
45          alert('Bienvenido');
46          Navegador('/home');
47        } else {
48          alert(data.mensaje);
49        }
50      })
51      .catch((error) => {
52        console.error('Error:', error);
53        alert('Hubo un problema con el inicio de sesión');
54      });
55    };
56

```



```

57     return (
58       <div className="login-background">
59         <div className="container-fluid h-100">
60           <div className="row align-items-center h-100">
61             <div className="col-md-6 mx-auto">
62               <div className="card">
63                 <div className="card-body">
64                   <h2 className="card-title text-center mb-4">Inicio de sesión</h2>
65                   <form onSubmit={handleSubmit}>
66                     <div className="form-floating">
67                       <input
68                         type="text"
69                         className="form-control"
70                         id="floatInput"
71                         placeholder="Ingresa su carnet"
72                         onChange={(e) => setCarnet(e.target.value)}
73                         value={carnet}
74                       />
75                       <label>Carnet</label>
76                     </div>
77                     <div className="form-floating">
78                       <input
79                         type="password"
80                         className="form-control"
81                         id="floatInput"
82                         placeholder="Ingresa su contraseña"
83                         onChange={(e) => setPassword(e.target.value)}
84                         value={password}
85                       />
86                       <label>Contraseña</label>
87                     </div>
88                     <div className="text-center">
89                       <button type="submit" className="btn btn-primary">Iniciar sesión</button>
90                       <h6><Link className="link text-skyblue" to="/register"> Registrarse </Link></h6>
91                     </div>
92                   </form>
93                 </div>
94             </div>
95           </div>
96         </div>
97       </div>
98     );
99   }
100 }
101
102 export default Login;

```

**CrearPubli.js:** Formulario para crear una publicación.

```

4 function CrearPublicacion() {
5   const [tipoPublicacion, setTipoPublicacion] = useState('');
6   const [idCurso, setIdCurso] = useState('');
7   const [idCatedratico, setIdCatedratico] = useState('');
8   const [mensaje, setMensaje] = useState('');
9   const [cursos, setCursos] = useState([]);
10  const [catedraticos, setCatedraticos] = useState([]);
11  const [error, setError] = useState('');
12  const navigate = useNavigate();
13
14  useEffect(() => {
15    // Fetch de cursos
16    fetch('http://localhost:5000/getCursos')
17      .then((res) => res.json())
18      .then((data) => setCursos(data.cursos))
19      .catch((err) => console.error(err));
20
21    // Fetch de catedraticos
22    fetch('http://localhost:5000/getCatedraticos')
23      .then((res) => res.json())
24      .then((data) => setCatedraticos(data.catedraticos))
25      .catch((err) => console.error(err));
26  }, []);
27
28  const handleSubmit = async (e) => {
29    e.preventDefault();
30
31    const publicacionData = {
32      tipo_publicacion: tipoPublicacion,
33      id_curso: tipoPublicacion === 'Curso' ? idCurso : null,
34      id_catedratico: tipoPublicacion === 'Catedratico' ? idCatedratico : null,
35      mensaje,
36    };
37
38    console.log('Datos enviados al backend:', publicacionData);
39
40    const token = localStorage.getItem('token'); // Obtener el token
41    console.log('Token:', token);
42
43    try {
44      const response = await fetch('http://localhost:5000/crearPubli', {
45        method: 'POST',
46        headers: {
47          'Content-Type': 'application/json',
48          'Authorization': 'Bearer ${token}',
49        },
50        body: JSON.stringify(publicacionData),
51      });
52
53      const result = await response.json();
54      if (response.ok) {
55        navigate('/home');
56      } else {
57        setError(result.mensaje);
58      }
59    } catch (error) {
60      setError('Error al crear la publicación');
61    }
62  };
63 }

```

```

64 return (
65   <form onSubmit={handleSubmit}>
66     <div className="form-group">
67       <label htmlFor="tipoPublicacion">Tipo de Publicación</label>
68       <select
69         id="tipoPublicacion"
70         value={tipoPublicacion}
71         onChange={(e) => setTipoPublicacion(e.target.value)}
72         className="form-control"
73         required
74       >
75         <option value="">Selecciona un tipo</option>
76         <option value="Curso">Curso</option>
77         <option value="Catedratico">Catedrático</option>
78       </select>
79     </div>
80
81     {tipoPublicacion === 'Curso' && (
82       <div className="form-group">
83         <label htmlFor="idCurso">Curso</label>
84         <select
85           id="idCurso"
86           value={idCurso}
87           onChange={(e) => setIdCurso(e.target.value)}
88           className="form-control"
89           required
90         >
91           <option value="">Selecciona un curso</option>
92           {cursos.map((curso) => (
93             <option key={curso.id_curso} value={curso.id_curso}>
94               {curso.nombre_curso}
95             </option>
96           ))}
97         </select>
98       </div>
99     )}
100
101     {tipoPublicacion === 'Catedratico' && (
102       <div className="form-group">
103         <label htmlFor="idCatedratico">Catedrático</label>
104         <select
105           id="idCatedratico"
106           value={idCatedratico}
107           onChange={(e) => setIdCatedratico(e.target.value)}
108           className="form-control"
109           required
110         >
111           <option value="">Selecciona un catedrático</option>
112           {catedraticos.map((catedratico) => (
113             <option key={catedratico.id_catedratico} value={catedratico.id_catedratico}>
114               {catedratico.nombre_catedratico}
115             </option>
116           ))}
117         </select>
118       </div>
119     )}
120   </form>

```

**Comentarios.js:** Formulario para agregar un comentario a la publicación deseada y que muestra los comentarios de la misma.

```
1  import React, { useState, useEffect } from "react";
2  import { useParams } from "react-router-dom";
3  import NavBar from './NavBar';
4
5  function Comentarios() {
6    const { id_publicacion } = useParams();
7    const [comentarios, setComentarios] = useState([]);
8    const [nuevoComentario, setNuevoComentario] = useState('');
9    const [error, setError] = useState('');
10
11    useEffect(() => {
12      obtenerComentarios();
13    }, [id_publicacion]);
14
15    // Función para obtener los comentarios desde el backend
16    const obtenerComentarios = () => {
17      fetch('http://localhost:5000/obtenerComentarios/${id_publicacion}')
18        .then((res) => res.json())
19        .then((data) => {
20          setComentarios(data.comentarios);
21        })
22        .catch((err) => {
23          console.error('Error al obtener comentarios:', err);
24          setError('Error al obtener los comentarios');
25        });
26    };
27
28    // Función para enviar un comentario
29    const enviarComentario = () => {
30      const token = localStorage.getItem('token'); // Suponiendo que el token está en localStorage
31      fetch('http://localhost:5000/crearComentario', {
32        method: 'POST',
33        headers: {
34          'Content-Type': 'application/json',
35          'Authorization': `Bearer ${token}`
36        },
37        body: JSON.stringify({ id_publicacion, comentario: nuevoComentario })
38      })
39        .then(res => res.json())
40        .then(() => {
41          obtenerComentarios(); // Actualizar los comentarios después de enviar uno nuevo
42          setNuevoComentario(''); // Limpiar el campo de texto
43        })
44        .catch(err => {
45          console.error('Error al enviar comentario:', err);
46        });
47    };
48  }
```

**Perfil.js:** Muestra el perfil del usuario el cual se ingresó desde el buscador de la página principal.

```
5 function Perfil() {
6   const [usuario, setUsuario] = useState(null);
7   const [cursos, setCursos] = useState([]);
8   const [totalCreditos, setTotalCreditos] = useState(0);
9   const [idCurso, setIdCurso] = useState('');
10  const [mensaje, setMensaje] = useState('');
11  const navigate = useNavigate();
12
13  useEffect(() => {
14    const fetchUsuario = async () => {
15      try {
16        const response = await fetch('http://localhost:5000/obtenerUsuario', {
17          headers: {
18            'Authorization': 'Bearer ${localStorage.getItem('token')}'
19          }
20        });
21        if (response.ok) {
22          const result = await response.json();
23          setUsuario(result.datos[0]); // Suponiendo que hay un solo usuario
24        } else {
25          alert('No se pudo obtener la información del usuario');
26        }
27      } catch (error) {
28        console.error('Error al obtener el usuario:', error);
29      }
30    };
31
32    const fetchCursosAprobadosYCreditos = async () => {
33      try {
34        const response = await fetch('http://localhost:5000/cursosAprobadosYCreditos', {
35          headers: {
36            'Authorization': 'Bearer ${localStorage.getItem('token')}'
37          }
38        });
39        if (response.ok) {
40          const result = await response.json();
41          setCursos(result.cursos);
42          setTotalCreditos(result.total_creditos);
43        } else {
44          alert('No se pudo obtener los cursos aprobados y créditos totales');
45        }
46      } catch (error) {
47        console.error('Error al obtener cursos aprobados y créditos:', error);
48      }
49    };
50  });
```

```
51 fetchUsuario();
52 fetchCursosAprobadosYCreditos();
53 }, []);
54
55 const handleAgregarCurso = async (e) => {
56   e.preventDefault();
57   try {
58     const response = await fetch('http://localhost:5000/agregarCursoAprobado', {
59       method: 'POST',
60       headers: {
61         'Content-Type': 'application/json',
62         'Authorization': 'Bearer ${localStorage.getItem('token')}'
63       },
64       body: JSON.stringify({ id_curso: idCurso })
65     });
66     if (response.ok) {
67       setMensaje('Curso aprobado agregado exitosamente');
68       setIdCurso(''); // Limpiar el campo de entrada
69       // Volver a obtener los cursos aprobados y créditos totales
70       const result = await fetch('http://localhost:5000/cursosAprobadosYCreditos', {
71         headers: {
72           'Authorization': 'Bearer ${localStorage.getItem('token')}'
73         }
74       });
75       const data = await result.json();
76       setCursos(data.cursos);
77       setTotalCreditos(data.total_creditos);
78     } else {
79       alert('Error al agregar curso aprobado');
80     }
81   } catch (error) {
82     console.error('Error al agregar curso aprobado:', error);
83     alert('Error al agregar curso aprobado');
84   }
85 };
86
87 if (!usuario) return <div className="alert alert-info">Cargando...</div>;
88
```

```

89     return (
90       <div className="container mt-4">
91         <h1 className="mb-4">Perfil de {usuario.name} {usuario.lname}</h1>
92         <div className="card mb-4">
93           <div className="card-body">
94             <p><strong>Email:</strong> {usuario.email}</p>
95             <p><strong>Carnet:</strong> {usuario.carnet}</p>
96           </div>
97         </div>
98         <h2 className="mb-4">Cursos Aprobados</h2>
99         {cursos.length === 0 ? (
100           <div className="alert alert-info">No tiene cursos aprobados.</div>
101         ) : (
102           <table className="table table-striped">
103             <thead>
104               <tr>
105                 <th>ID del Curso</th>
106                 <th>Nombre del Curso</th>
107                 <th>Fecha de Aprobación</th>
108               </tr>
109             </thead>
110             <tbody>
111               {cursos.map((curso) => (
112                 <tr key={curso.id_curso}>
113                   <td>{curso.id_curso}</td>
114                   <td>{curso.nombre_curso}</td>
115                   <td>{new Date(curso.fecha_aprobacion).toLocaleDateString()}</td>
116                 </tr>
117               ))}
118             </tbody>
119           </table>
120         )}
121         <h3 className="mt-4">Créditos Totales</h3>
122         <p>{totalCreditos} créditos</p>
123         <form onSubmit={handleAgregarCurso} className="mb-4">
124           <div className="form-group">
125             <label htmlFor="idCurso">ID del Curso</label>
126             <input
127               type="text"
128               id="idCurso"
129               className="form-control"
130               placeholder="Ingrese el ID del curso"
131               value={idCurso}
132               onChange={(e) => setIdCurso(e.target.value)}
133               required
134             />
135           </div>
136           <button type="submit" className="btn btn-primary mt-2">Agregar Curso Aprobado</button>
137         </form>
138         {mensaje && <div className="alert alert-success">{mensaje}</div>}
139       </div>
140     );
141   }
142
143   export default Perfil;

```

## ANEXOS

### Diagrama Entidad Relación

