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

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)3

Oracle Linux 7.x (64 bits)3 (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.

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

exports.obtenerCursosAprobados = (req, res) => {
  const carnet = parseInt(req.params.carnet);

  if (isNaN(carnet)) {
    return res.status(400).json({ mensaje: 'Carnet inválido' });
  }

  const sql = `
    SELECT c.id_curso, c.nombre_curso, ca.fecha_aprobacion
    FROM cursos_aprobados ca
    JOIN cursos c ON ca.id_curso = c.id_curso
    WHERE ca.carnet = ?
  `;

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

    const totalCreditosSql = `
      SELECT SUM(c.creditos) as total_creditos
      FROM cursos_aprobados ca
      JOIN cursos c ON ca.id_curso = c.id_curso
      WHERE ca.carnet = ?
    `;

    db.query(totalCreditosSql, [carnet], (err, creditoResults) => {
      if (err) {
        console.error('Error al obtener créditos totales:', err);
        return res.status(500).json({ mensaje: 'Error al obtener créditos totales' });
      }

      res.status(200).json({
        cursos: results,
        totalCreditos: creditoResults[0].total_creditos || 0
      });
    });
  });
};
```

```
exports.agregarCursoAprobado = (req, res) => {
  const { id_curso } = req.body;
  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 fecha_aprobacion = new Date(); // Fecha actual

    const sql = `
      INSERT INTO cursos_aprobados (carnet, id_curso, fecha_aprobacion)
      VALUES (?, ?, ?)
    `;

    db.query(sql, [carnet, id_curso, fecha_aprobacion], (err, results) => {
      if (err) {
        console.error('Error al agregar curso aprobado:', err);
        return res.status(500).json({ mensaje: 'Error al agregar curso aprobado' });
      }
      res.json({ mensaje: 'Curso aprobado agregado exitosamente' });
    });
  });
};
```

```
// Función para obtener cursos aprobados y créditos totales del usuario
exports.obtenerCursosAprobadosYCreditos = (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 sqlCursos = `
      SELECT c.id_curso, c.nombre_curso, ca.fecha_aprobacion
      FROM cursos c
      JOIN cursos_aprobados ca ON c.id_curso = ca.id_curso
      WHERE ca.carnet = ?
      ORDER BY ca.fecha_aprobacion ASC
    `;

    db.query(sqlCursos, [carnet], (err, cursos) => {
      if (err) {
        console.error('Error al obtener cursos aprobados:', err);
        return res.status(500).json({ mensaje: 'Error al obtener cursos aprobados' });
      }

      const sqlCreditos = `
        SELECT SUM(c.creditos) AS total_creditos
        FROM cursos c
        JOIN cursos_aprobados ca ON c.id_curso = ca.id_curso
        WHERE ca.carnet = ?
      `;

      db.query(sqlCreditos, [carnet], (err, resultados) => {
        if (err) {
          console.error('Error al obtener créditos totales:', err);
          return res.status(500).json({ mensaje: 'Error al obtener créditos totales' });
        }

        res.json({
          cursos: cursos,
          total_creditos: resultados[0].total_creditos
        });
      });
    });
  });
};
```

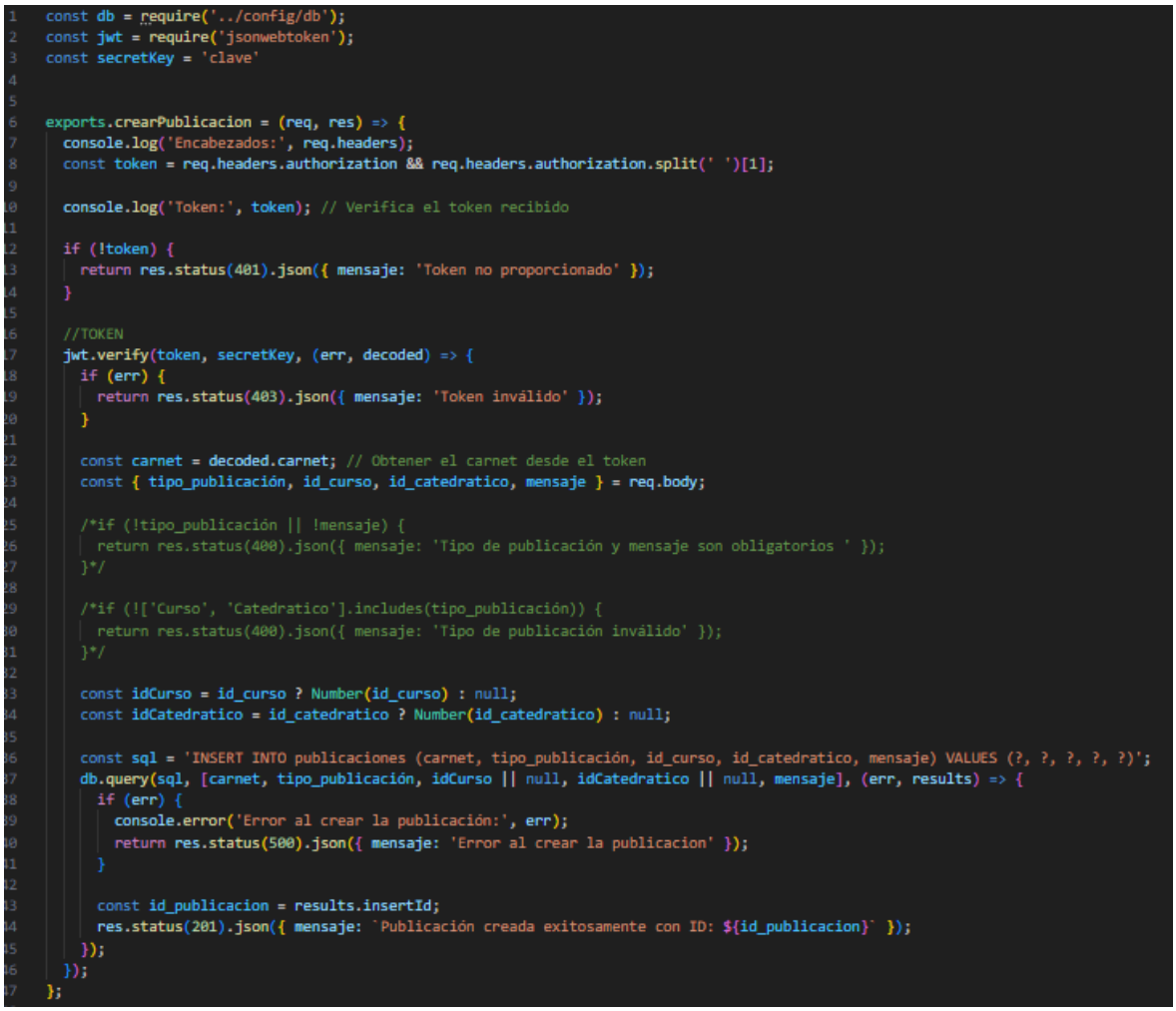
Gets.js: funciones para obtener los cursos y catedráticos para las combobox cuando se desea agregar un comentario.

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

// Obtener todos los cursos
exports.getCursos = (req, res) => {
    const sql = 'SELECT * FROM cursos';
    db.query(sql, (err, results) => {
        if (err) {
            console.error('Error al obtener cursos:', err);
            return res.status(500).json({ mensaje: 'Error al obtener cursos' });
        }
        res.status(200).json({ cursos: results });
    });
};

// Obtener todos los catedráticos
exports.getCatedraticos = (req, res) => {
    const sql = 'SELECT * FROM catedratico';
    db.query(sql, (err, results) => {
        if (err) {
            console.error('Error al obtener catedráticos:', err);
            return res.status(500).json({ mensaje: 'Error al obtener catedráticos' });
        }
        res.status(200).json({ catedraticos: results });
    });
};
```

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.



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


exports.crearPublicacion = (req, res) => {
  console.log('Encabezados:', req.headers);
  const token = req.headers.authorization && req.headers.authorization.split(' ')[1];

  console.log('Token:', token); // Verifica el token recibido

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

  //TOKEN
  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 { tipo_publicación, id_curso, id_catedratico, mensaje } = req.body;

    /*if (!tipo_publicación || !mensaje) {
      return res.status(400).json({ mensaje: 'Tipo de publicación y mensaje son obligatorios ' });
    }*/

    /*if (!['Curso', 'Catedratico'].includes(tipo_publicación)) {
      return res.status(400).json({ mensaje: 'Tipo de publicación inválido' });
    }*/

    const idCurso = id_curso ? Number(id_curso) : null;
    const idCatedratico = id_catedratico ? Number(id_catedratico) : null;

    const sql = 'INSERT INTO publicaciones (carnet, tipo_publicación, id_curso, id_catedratico, mensaje) VALUES (?, ?, ?, ?, ?)';
    db.query(sql, [carnet, tipo_publicación, idCurso || null, idCatedratico || null, mensaje], (err, results) => {
      if (err) {
        console.error('Error al crear la publicación:', err);
        return res.status(500).json({ mensaje: 'Error al crear la publicacion' });
      }

      const id_publicacion = results.insertId;
      res.status(201).json({ mensaje: `Publicación creada exitosamente con ID: ${id_publicacion}` });
    });
  });
};
```

```
exports.obtenerPublicaciones = (req, res) => {
  const { filtroCurso, filtroCatedratico, nombreCurso, nombreCatedratico } = req.query;

  let sql = `
    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
    FROM publicaciones p
    LEFT JOIN cursos c ON p.id_curso = c.id_curso
    LEFT JOIN catedratico cat ON p.id_catedratico = cat.id_catedratico
    WHERE 1=1
  `;

  let params = [];

// Filtrar por curso (tiene que recibir id)
if (filtroCurso) {
  const idCurso = parseInt(filtroCurso, 10); // Parsear a entero
  sql += " AND p.id_curso = ?";
  params.push(idCurso);
}

// Filtrar por catedrático (tiene que recibir id)
if (filtroCatedratico) {
  const idCatedratico = parseInt(filtroCatedratico, 10); // Parsear a entero
  sql += " AND p.id_catedratico = ?";
  params.push(idCatedratico);
}

  // Filtrar por nombre del curso
  if (nombreCurso) {
      sql += " AND c.nombre_curso LIKE ?";
      params.push(`%${nombreCurso}%`);
  }

  // Filtrar por nombre del catedrático
  if (nombreCatedratico) {
      sql += " AND cat.nombre_catedratico LIKE ?";
      params.push(`%${nombreCatedratico}%`);
  }

  // Ordenar por fecha de creación (más recientes primero)
  sql += " ORDER BY p.fecha_creacion DESC";

  db.query(sql, params, (err, results) => {
      if (err) {
          console.error('Error al obtener publicaciones:', err);
          return res.status(500).json({ mensaje: 'Error al obtener las publicaciones' });
      }

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

Ususarios.js: Incluye las funcionalidades para buscar un usuario según su carnet y obtener un usuario para mostrar su información donde sea requerrido.

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

exports.buscarUsuario = (req, res) => {
  const carnet = parseInt(req.params.carnet);

  if (isNaN(carnet)) {
    return res.status(400).json({ mensaje: 'Carnet inválido' });
  }

  const sql = `
    SELECT * FROM usuarios WHERE carnet = ?
  `;

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

    if (results.length === 0) {
      return res.status(404).json({ mensaje: 'Usuario no encontrado' });
    }

    res.status(200).json({ usuario: results[0] });
  });
};
```

```
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

```
const express = require('express')
const cors = require('cors')

const MyApp = express();

const bodyParser = require('body-parser');

const corsOptions = {
    origin: 'http://localhost:3000'
};

MyApp.use(cors(corsOptions));
MyApp.use(bodyParser.json({ limit: '15mb' }));

MyApp.use(express.json())
MyApp.use(cors())

//--------
const Router = require('./routers/routes')
MyApp.use(Router)
//--------

const port = 5000;

MyApp.listen(port, () =>{
    console.log(`The API is listening at http://localhost:${port}`);
});
```

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.

```jsx
import React, { useState } from "react";
import '../Styles/Styles.css'
import { useNavigate, Link } from "react-router-dom";

function NavBar() {
    const [carnetBuscado, setCarnetBuscado] = useState('');
    const navigate = useNavigate();

    const handleLogout = () => {
        localStorage.removeItem('token');
        navigate('/login');
    };

    const handleBuscarUsuario = async (e) => {
        e.preventDefault();
        try {
            const response = await fetch(`http://localhost:5000/buscarUsuario/${carnetBuscado}`);
            if (response.ok) {
                const result = await response.json();
                navigate(`/perfil/${carnetBuscado}`);
            } else {
                alert('Usuario no encontrado');
            }
        } catch (error) {
            console.error('Error al buscar usuario:', error);
            alert('Error al buscar usuario');
        }
    };
```

```jsx
    return (
        <div className="container-navbar">
            <div className="left-container-navbar">
                <h1 style={{ color: "white" }}>USAC</h1>
                <h1>.</h1>
                <ul className="link-list">
                    <li className="link-list-item">
                        <Link className="link" to="/crearP"> Crear publicación</Link>
                    </li>
                </ul>
            </div>
            <div className="right-container-navbar">
                <form onSubmit={handleBuscarUsuario} className="form-inline">
                    <input
                        type="text"
                        className="form-control mr-sm-2"
                        placeholder="Buscar usuario por carnet"
                        value={carnetBuscado}
                        onChange={(e) => setCarnetBuscado(e.target.value)}
                        required
                    />
                    <button className="btn btn-outline-info my-2 my-sm-0" type="submit">
                        Buscar
                    </button>
                </form>
                <button className="btn btn-outline-info" onClick={handleLogout}>
                    Logout
                </button>
                <h1>.</h1>
                <Link to="/perfil">
                    <button className="btn btn-outline-info">
                        Perfil
                    </button>
                </Link>
            </div>
        </div>
    )
}

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();
    };
```

```
    return (
        <div>
            <NavBar />

            <div className="container mt-5">
                <h2>Publicaciones</h2>

                <form onSubmit={aplicarFiltros} className="mb-4">
                    <div className="form-group">
                        <label>Filtrar por Curso</label>
                        <input
                            type="text"
                            className="form-control"
                            value={filtroCurso}
                            onChange={(e) => setFiltroCurso(e.target.value)}
                        />
                    </div>
                    <div className="form-group">
                        <label>Filtrar por Catedrático</label>
                        <input
                            type="text"
                            className="form-control"
                            value={filtroCatedratico}
                            onChange={(e) => setFiltroCatedratico(e.target.value)}
                        />
                    </div>
                    <div className="form-group">
                        <label>Buscar por nombre del curso</label>
                        <input
                            type="text"
                            className="form-control"
                            value={nombreCurso}
                            onChange={(e) => setNombreCurso(e.target.value)}
                        />
                    </div>
                    <div className="form-group">
                        <label>Buscar por nombre del catedrático</label>
                        <input
                            type="text"
                            className="form-control"
                            value={nombreCatedratico}
                            onChange={(e) => setNombreCatedratico(e.target.value)}
                        />
                    </div>
                    <button type="submit" className="btn btn-primary">Aplicar Filtros</button>
                </form>

                {error && <p className="text-danger">{error}</p>}
```

```
                <div className="list-group">
                    {publicaciones.map((pub) => (
                        <div key={pub.id_publicacion} className="list-group-item">
                            <h5>{pub.tipo_publicación === 'Curso' ? pub.nombre_curso : pub.nombre_catedratico}</h5>
                            <p>Publicación de: {pub.carnet}</p>
                            <p>{pub.mensaje}</p>
                            <small className="text-muted">Fecha de creación: {new Date(pub.fecha_creacion).toLocaleString()}</small>
                            <button
                                onClick={() => navigate(`/comentarios/${pub.id_publicacion}`)}
                                className="btn btn-info mt-2"
                            >
                                Ver Comentarios
                            </button>
                        </div>
                    ))}
                </div>
            </div>
        </div>
    );
}

export default Home;
```

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

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

function Login() {
    const [carnet, setCarnet] = useState('');
    const [password, setPassword] = useState('');

    const Navegador = useNavigate();

    const handleSubmit = (event) => {
        event.preventDefault();

        // Verificación de datos de administrador
        if (carnet === "202307765" && password === "admin") {

            alert("Inicio de sesión como administrador exitoso");
            Navegador('/admin'); // Redirige a la página de admin (PENDIENTE)
            return;
        }

        // Datos del usuario
        const dataJson = {
            carnet: carnet,
            password: password
        };

        fetch('http://localhost:5000/login', {
            method: 'POST',
            body: JSON.stringify(dataJson),
            headers: {
                'Content-Type': 'application/json'
            }
        })
        .then((response) => {
            if (!response.ok) {
                throw new Error(`Error: ${response.statusText}`);
            }
            return response.json();
        })
        .then((data) => {
            if (data.mensaje === 'Login exitoso') {
                localStorage.setItem('token', data.token);
                console.log(data.datos);
                alert(`Bienvenido`);
                Navegador('/home');
            } else {
                alert(data.mensaje);
            }
        })
        .catch((error) => {
            console.error('Error:', error);
            alert('Hubo un problema con el inicio de sesión');
        });
    };
```

```jsx
    return (
        <div className="login-background">
            <div className="container-fluid h-100">
                <div className="row align-items-center h-100">
                    <div className="col-md-6 mx-auto">
                        <div className="card">
                            <div className="card-body">
                                <h2 className="card-title text-center mb-4">Inicio de sesión</h2>
                                <form onSubmit={handleSubmit}>
                                    <div className="form-floating">
                                        <input
                                            type="text"
                                            className="form-control"
                                            id="floatInput"
                                            placeholder="Ingrese su carnet"
                                            onChange={(e) => setCarnet(e.target.value)}
                                            value={carnet}
                                        />
                                        <label>Carnet</label>
                                    </div>
                                    <div className="form-floating">
                                        <input
                                            type="password"
                                            className="form-control"
                                            id="floatInput"
                                            placeholder="Ingrese su contraseña"
                                            onChange={(e) => setPassword(e.target.value)}
                                            value={password}
                                        />
                                        <label>Contraseña</label>
                                    </div>
                                    <div className="text-center">
                                        <button type="submit" className="btn btn-primary">Iniciar sesión</button>
                                        <h6><Link className="link text-skyblue" to="/register"> Registrarse </Link></h6>
                                    </div>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    );
}

export default Login;
```

CrearPubli.js: Formulario para crear una publicación.

```jsx
function CrearPublicacion() {
  const [tipoPublicacion, setTipoPublicacion] = useState('');
  const [idCurso, setIdCurso] = useState('');
  const [idCatedratico, setIdCatedratico] = useState('');
  const [mensaje, setMensaje] = useState('');
  const [cursos, setCursos] = useState([]);
  const [catedraticos, setCatedraticos] = useState([]);
  const [error, setError] = useState('');
  const navigate = useNavigate();

  useEffect(() => {
    // Fetch de cursos
    fetch('http://localhost:5000/getCursos')
      .then((res) => res.json())
      .then((data) => setCursos(data.cursos))
      .catch((err) => console.error(err));

    // Fetch de catedraticos
    fetch('http://localhost:5000/getCatedraticos')
      .then((res) => res.json())
      .then((data) => setCatedraticos(data.catedraticos))
      .catch((err) => console.error(err));
  }, []);

  const handleSubmit = async (e) => {
    e.preventDefault();

    const publicacionData = {
      tipo_publicación: tipoPublicacion,
      id_curso: tipoPublicacion === 'Curso' ? idCurso : null,
      id_catedratico: tipoPublicacion === 'Catedratico' ? idCatedratico : null,
      mensaje,
    };

    console.log('Datos enviados al backend:', publicacionData);

    const token = localStorage.getItem('token'); // Obtener el token
    console.log('Token:', token);

    try {
      const response = await fetch('http://localhost:5000/crearPubli', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${token}`,
        },
        body: JSON.stringify(publicacionData),
      });

      const result = await response.json();
      if (response.ok) {
        navigate('/home');
      } else {
        setError(result.mensaje);
      }
    } catch (error) {
      setError('Error al crear la publicación');
```

```jsx
return (
  <form onSubmit={handleSubmit}>
    <div className="form-group">
      <label htmlFor="tipoPublicacion">Tipo de Publicación</label>
      <select
        id="tipoPublicacion"
        value={tipoPublicacion}
        onChange={(e) => setTipoPublicacion(e.target.value)}
        className="form-control"
        required
      >
        <option value="">Selecciona un tipo</option>
        <option value="Curso">Curso</option>
        <option value="Catedratico">Catedrático</option>
      </select>
    </div>

    {tipoPublicacion === 'Curso' && (
      <div className="form-group">
        <label htmlFor="idCurso">Curso</label>
        <select
          id="idCurso"
          value={idCurso}
          onChange={(e) => setIdCurso(e.target.value)}
          className="form-control"
          required
        >
          <option value="">Selecciona un curso</option>
          {cursos.map((curso) => (
            <option key={curso.id_curso} value={curso.id_curso}>
              {curso.nombre_curso}
            </option>
          ))}
        </select>
      </div>
    )}

    {tipoPublicacion === 'Catedratico' && (
      <div className="form-group">
        <label htmlFor="idCatedratico">Catedrático</label>
        <select
          id="idCatedratico"
          value={idCatedratico}
          onChange={(e) => setIdCatedratico(e.target.value)}
          className="form-control"
          required
        >
          <option value="">Selecciona un catedrático</option>
          {catedraticos.map((catedratico) => (
            <option key={catedratico.id_catedratico} value={catedratico.id_catedratico}>
              {catedratico.nombre_catedratico}
            </option>
          ))}
        </select>
      </div>
    )}
```

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

```jsx
import React, { useState, useEffect } from "react";
import { useParams } from "react-router-dom";
import NavBar from './NavBar';

function Comentarios() {
    const { id_publicacion } = useParams();
    const [comentarios, setComentarios] = useState([]);
    const [nuevoComentario, setNuevoComentario] = useState('');
    const [error, setError] = useState('');

    useEffect(() => {
        obtenerComentarios();
    }, [id_publicacion]);

    // Función para obtener los comentarios desde el backend
    const obtenerComentarios = () => {
        fetch(`http://localhost:5000/obtenerComentarios/${id_publicacion}`)
            .then((res) => res.json())
            .then((data) => {
                setComentarios(data.comentarios);
            })
            .catch((err) => {
                console.error('Error al obtener comentarios:', err);
                setError('Error al obtener los comentarios');
            });
    };

    // Función para enviar un comentario
    const enviarComentario = () => {
        const token = localStorage.getItem('token');  // Suponiendo que el token está en localStorage
        fetch(`http://localhost:5000/crearComentario`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            },
            body: JSON.stringify({ id_publicacion, comentario: nuevoComentario })
        })
        .then(res => res.json())
        .then(() => {
            obtenerComentarios();  // Actualizar los comentarios después de enviar uno nuevo
            setNuevoComentario('');  // Limpiar el campo de texto
        })
        .catch(err => {
            console.error('Error al enviar comentario:', err);
        });
    };
```

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

```
function Perfil() {
  const [usuario, setUsuario] = useState(null);
  const [cursos, setCursos] = useState([]);
  const [totalCreditos, setTotalCreditos] = useState(0);
  const [idCurso, setIdCurso] = useState('');
  const [mensaje, setMensaje] = useState('');
  const navigate = useNavigate();

  useEffect(() => {
    const fetchUsuario = async () => {
      try {
        const response = await fetch('http://localhost:5000/obtenerUsuario', {
          headers: {
            'Authorization': `Bearer ${localStorage.getItem('token')}`
          }
        });
        if (response.ok) {
          const result = await response.json();
          setUsuario(result.datos[0]); // Suponiendo que hay un solo usuario
        } else {
          alert('No se pudo obtener la información del usuario');
        }
      } catch (error) {
        console.error('Error al obtener el usuario:', error);
      }
    };

    const fetchCursosAprobadosYCreditos = async () => {
      try {
        const response = await fetch('http://localhost:5000/cursosAprobadosYCreditos', {
          headers: {
            'Authorization': `Bearer ${localStorage.getItem('token')}`
          }
        });
        if (response.ok) {
          const result = await response.json();
          setCursos(result.cursos);
          setTotalCreditos(result.total_creditos);
        } else {
          alert('No se pudo obtener los cursos aprobados y créditos totales');
        }
      } catch (error) {
        console.error('Error al obtener cursos aprobados y créditos:', error);
      }
    };
```

```
    fetchUsuario();
    fetchCursosAprobadosYCreditos();
  }, []);

  const handleAgregarCurso = async (e) => {
    e.preventDefault();
    try {
      const response = await fetch('http://localhost:5000/agregarCursoAprobado', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${localStorage.getItem('token')}`
        },
        body: JSON.stringify({ id_curso: idCurso })
      });
      if (response.ok) {
        setMensaje('Curso aprobado agregado exitosamente');
        setIdCurso(''); // Limpiar el campo de entrada
        // Volver a obtener los cursos aprobados y créditos totales
        const result = await fetch('http://localhost:5000/cursosAprobadosYCreditos', {
          headers: {
            'Authorization': `Bearer ${localStorage.getItem('token')}`
          }
        });
        const data = await result.json();
        setCursos(data.cursos);
        setTotalCreditos(data.total_creditos);
      } else {
        alert('Error al agregar curso aprobado');
      }
    } catch (error) {
      console.error('Error al agregar curso aprobado:', error);
      alert('Error al agregar curso aprobado');
    }
  };

  if (!usuario) return <div className="alert alert-info">Cargando...</div>;
```

```jsx
  return (
    <div className="container mt-4">
      <h1 className="mb-4">Perfil de {usuario.name} {usuario.lname}</h1>
      <div className="card mb-4">
        <div className="card-body">
          <p><strong>Email:</strong> {usuario.email}</p>
          <p><strong>Carnet:</strong> {usuario.carnet}</p>
        </div>
      </div>
      <h2 className="mb-4">Cursos Aprobados</h2>
      {cursos.length === 0 ? (
        <div className="alert alert-info">No tiene cursos aprobados.</div>
      ) : (
        <table className="table table-striped">
          <thead>
            <tr>
              <th>ID del Curso</th>
              <th>Nombre del Curso</th>
              <th>Fecha de Aprobación</th>
            </tr>
          </thead>
          <tbody>
            {cursos.map((curso) => (
              <tr key={curso.id_curso}>
                <td>{curso.id_curso}</td>
                <td>{curso.nombre_curso}</td>
                <td>{new Date(curso.fecha_aprobacion).toLocaleDateString()}</td>
              </tr>
            ))}
          </tbody>
        </table>
      )}
      <h3 className="mt-4">Créditos Totales</h3>
      <p>{totalCreditos} créditos</p>
      <form onSubmit={handleAgregarCurso} className="mb-4">
        <div className="form-group">
          <label htmlFor="idCurso">ID del Curso</label>
          <input
            type="text"
            id="idCurso"
            className="form-control"
            placeholder="Ingrese el ID del curso"
            value={idCurso}
            onChange={(e) => setIdCurso(e.target.value)}
            required
          />
        </div>
        <button type="submit" className="btn btn-primary mt-2">Agregar Curso Aprobado</button>
      </form>
      {mensaje && <div className="alert alert-success">{mensaje}</div>}
    </div>
  );
}

export default Perfil;
```

ANEXOS

Diagrama Entidad Relación