ANALIZA BIEN EL CODIGO:  
  
DocenteDashboard.jsx:  
  
import React, { useState, useEffect } from "react";

import axios from "axios";

import { useNavigate } from "react-router-dom";

import { toast } from "react-toastify";

import ResourceList from "../docente/ResourceList";

import ResourceUpload from "../docente/ResourceUpload";

function Dashboard() {

  const handleUploadSuccess = (uploadedResource) => {

    toast.success("Recurso subido correctamente");

    setActiveSection("resources"); // Redirige a la lista de recursos

    // Si necesitas actualizar la lista de recursos, puedes hacerlo aquí

  };

  const [courses, setCourses] = useState([]);

  const [categories, setCategories] = useState([]);

  const userId = localStorage.getItem("userId"); // Asegúrate de guardar esto en el login

  const navigate = useNavigate();

  const [sidebarCollapsed, setSidebarCollapsed] = useState(false);

  const [activeSection, setActiveSection] = useState("welcome");

  const [isEditing, setIsEditing] = useState(false);

  const [user, setUser] = useState({

    nombre\_completo: "",

    correo: "",

    nombre\_rol: "",

    area\_interes: "",

    foto\_perfil: "",

    password: "",

  });

  const token = localStorage.getItem("token");

  // Carga los cursos y categorías al montar el componente

  useEffect(() => {

    const loadInitialData = async () => {

      try {

        const response = await axios.get(

          "http://localhost:3000/api/docente/datos-utiles",

          {

            headers: { Authorization: `Bearer ${token}` },

          }

        );

        setCourses(response.data.cursos);

        setCategories(response.data.categorias);

      } catch (error) {

        console.error("Error al cargar datos iniciales:", error);

      }

    };

    if (token) loadInitialData();

  }, [token]);

  useEffect(() => {

    if (!token) {

      navigate("/");

      return;

    }

    const loadProfile = async () => {

      try {

        const response = await axios.get("http://localhost:3000/api/profile", {

          headers: { Authorization: `Bearer ${token}` },

        });

        // Extrae solo el nombre del archivo si hay una URL completa

        let foto\_perfil = response.data.foto\_perfil;

        if (foto\_perfil && foto\_perfil.includes("/")) {

          foto\_perfil = foto\_perfil.split("/").pop();

        }

        setUser({

          nombre\_completo: response.data.nombre\_completo,

          correo: response.data.correo,

          nombre\_rol: response.data.nombre\_rol,

          area\_interes: response.data.area\_interes || "",

          foto\_perfil: foto\_perfil || "",

          password: "",

        });

      } catch (error) {

        console.error("Error al cargar perfil:", error);

        navigate("/");

      }

    };

    loadProfile();

  }, [token, navigate]);

  const toggleSidebar = () => {

    setSidebarCollapsed(!sidebarCollapsed);

  };

  const showSection = (section) => {

    setActiveSection(section);

    setIsEditing(false);

  };

  const handleEdit = async () => {

    if (isEditing) {

      try {

        // Primero actualizamos los datos del perfil

        await axios.put(

          "http://localhost:3000/api/profile",

          {

            nombre\_completo: user.nombre\_completo,

            correo: user.correo,

            password: user.password,

            area\_interes: user.area\_interes,

          },

          {

            headers: { Authorization: `Bearer ${token}` },

          }

        );

        // Luego actualizamos la foto si hay una seleccionada

        if (selectedFile) {

          const formData = new FormData();

          formData.append("profile\_image", selectedFile);

          const pictureResponse = await axios.put(

            "http://localhost:3000/api/profile/picture",

            formData,

            {

              headers: {

                Authorization: `Bearer ${token}`,

                "Content-Type": "multipart/form-data",

              },

            }

          );

          // Forzar una actualización completa del estado

          setUser((prev) => ({

            ...prev,

            foto\_perfil: pictureResponse.data.filename,

            password: "", // Limpiar contraseña

          }));

        }

        // Recargar los datos del servidor para asegurar consistencia

        const refreshed = await axios.get("http://localhost:3000/api/profile", {

          headers: { Authorization: `Bearer ${token}` },

        });

        setUser({

          nombre\_completo: refreshed.data.nombre\_completo,

          correo: refreshed.data.correo,

          nombre\_rol: refreshed.data.nombre\_rol,

          area\_interes: refreshed.data.area\_interes || "",

          foto\_perfil: refreshed.data.foto\_perfil?.split("/").pop() || "",

          password: "",

        });

        alert(

          selectedFile ? "Perfil y foto actualizados" : "Perfil actualizado"

        );

        setSelectedFile(null);

      } catch (error) {

        console.error("Error:", error);

        alert(error.response?.data?.message || "Error al actualizar");

      }

    }

    setIsEditing(!isEditing);

  };

  const handleChange = (e) => {

    setUser({ ...user, [e.target.name]: e.target.value });

  };

  const handleLogout = () => {

    localStorage.removeItem("token");

    localStorage.removeItem("user");

    navigate("/");

  };

  // Foto de perfil

  const [selectedFile, setSelectedFile] = useState(null);

  const handleFileChange = (e) => {

    if (e.target.files && e.target.files[0]) {

      const file = e.target.files[0];

      // Validación básica del tipo de archivo

      if (!file.type.match("image.\*")) {

        alert("Por favor selecciona un archivo de imagen válido");

        return;

      }

      // Validación de tamaño (ejemplo: máximo 2MB)

      if (file.size > 2 \* 1024 \* 1024) {

        alert("La imagen no debe exceder los 2MB");

        return;

      }

      setSelectedFile(file);

      // Vista previa inmediata (opcional)

      const reader = new FileReader();

      reader.onload = (event) => {

        setUser((prev) => ({ ...prev, foto\_perfil: event.target.result }));

      };

      reader.readAsDataURL(file);

    }

  };

  return (

    <div className="d-flex flex-column vh-100">

      {/\* Navbar \*/}

      <nav className="navbar navbar-expand-lg navbar-light bg-white shadow-sm px-4 py-2">

        <button className="btn btn-outline-dark me-3" onClick={toggleSidebar}>

          ☰

        </button>

        <div className="ms-auto d-flex align-items-center dropdown">

          <div

            className="d-flex align-items-center"

            data-bs-toggle="dropdown"

            style={{ cursor: "pointer" }}

          >

            <img

              src={

                user.foto\_perfil

                  ? `http://localhost:3000/uploads/${

                      user.foto\_perfil

                    }?t=${Date.now()}`

                  : `https://ui-avatars.com/api/?name=${encodeURIComponent(

                      user.nombre\_completo || "U"

                    )}&background=random&rounded=true&size=40`

              }

              alt="avatar"

              className="rounded-circle border"

              width="40"

              height="40"

            />

            <span className="ms-2 fw-semibold text-dark">

              {user.nombre\_completo}

            </span>

          </div>

          <ul className="dropdown-menu dropdown-menu-end mt-2">

            <li>

              <button className="dropdown-item" onClick={handleLogout}>

                Cerrar sesión

              </button>

            </li>

          </ul>

        </div>

      </nav>

      <div className="d-flex flex-grow-1">

        {/\* Sidebar \*/}

        <div

          className={`bg-dark text-white sidebar shadow-sm ${

            sidebarCollapsed ? "collapsed" : ""

          }`}

          style={{ minWidth: "220px", transition: "all 0.3s" }}

        >

          <h5 className="text-center py-3 border-bottom border-secondary text-white">

            <i

              className="bi bi-folder2-open me-2"

              style={{ fontSize: "1.2rem" }}

            ></i>

            Menú

          </h5>

          <ul className="nav flex-column px-3">

            {[

              {

                icon: "bi-house-door",

                label: "Inicio",

                section: "welcome",

                color: "#FFC107",

              }, // amarillo

              {

                icon: "bi-journal-bookmark",

                label: "Mis recursos",

                section: "resources",

                color: "#0D6EFD",

              }, // azul

              {

                icon: "bi-cloud-arrow-up",

                label: "Subir recursos",

                section: "upload",

                color: "#198754",

              }, // verde

              {

                icon: "bi-share-fill",

                label: "Recursos Compartidos",

                section: "shared",

                color: "#6F42C1",

              }, // morado

              {

                icon: "bi-person",

                label: "Perfil",

                section: "profile",

                color: "#FD7E14",

              }, // naranja

            ].map((item) => (

              <li key={item.section} className="nav-item my-1">

                <a

                  href="#"

                  onClick={() => showSection(item.section)}

                  className="nav-link text-white px-2 d-flex align-items-center gap-2"

                  style={{ transition: "0.2s", borderRadius: "5px" }}

                >

                  <i

                    className={`bi ${item.icon}`}

                    style={{ color: item.color }}

                  ></i>

                  {item.label}

                </a>

              </li>

            ))}

            <hr className="border-secondary my-2" />

            <li className="nav-item px-0">

              <a

                href="https://tubiblioteca.utp.edu.pe"

                className="nav-link text-white d-flex align-items-center gap-2"

              >

                <i className="bi bi-book" style={{ color: "#0DCAF0" }}></i>

                UTP+biblio

              </a>

            </li>

            <li className="nav-item px-0">

              <a

                href="#"

                className="nav-link text-white d-flex align-items-center gap-2"

              >

                <i

                  className="bi bi-question-circle"

                  style={{ color: "#DC3545" }}

                ></i>

                Ayuda

              </a>

            </li>

          </ul>

        </div>

        {/\* Main Content \*/}

        <div className="flex-grow-1 p-4 bg-light">

          {activeSection === "welcome" && (

            <div className="text-center mt-5">

              <h2

                className="fw-bold text-center"

                style={{

                  color: "#1B1F3B",

                  fontSize: "2rem",

                  borderBottom: "3px solid #1B1F3B",

                  display: "inline-block",

                  paddingBottom: "8px",

                }}

              >

                ¡Bienvenido, {user.nombre\_rol}!

              </h2>

              <p className="text-muted mt-2">Nos alegra tenerte de vuelta.</p>

            </div>

          )}

          {activeSection === "profile" && (

            <div className="container mt-5">

              <h2 className="mb-4 text-center fw-bold text-dark">

                <i

                  className="bi bi-person-circle me-2"

                  style={{ fontSize: "2rem" }}

                ></i>

                Perfil de Usuario

              </h2>

              <div

                className="card mx-auto shadow-sm rounded-4"

                style={{ maxWidth: "600px" }}

              >

                <div className="card-body">

                  <form>

                    {/\* campos del perfil \*/}

                    <div className="mb-3">

                      <label className="form-label">Foto de perfil</label>

                      <input

                        type="file"

                        className="form-control"

                        onChange={handleFileChange}

                        disabled={!isEditing}

                      />

                    </div>

                    <div className="mb-3">

                      <label className="form-label">Nombre</label>

                      <input

                        type="text"

                        className="form-control"

                        name="nombre\_completo"

                        value={user.nombre\_completo}

                        onChange={handleChange}

                        readOnly={!isEditing}

                      />

                    </div>

                    <div className="mb-3">

                      <label className="form-label">Correo</label>

                      <input

                        type="email"

                        className="form-control"

                        name="correo"

                        value={user.correo}

                        onChange={handleChange}

                        readOnly={!isEditing}

                      />

                    </div>

                    <div className="mb-3">

                      <label className="form-label">Nueva contraseña</label>

                      <input

                        type="password"

                        className="form-control"

                        name="password"

                        value={user.password}

                        onChange={handleChange}

                        placeholder="Escribe una nueva contraseña si deseas cambiarla"

                        disabled={!isEditing}

                      />

                    </div>

                    <div className="mb-3">

                      <label className="form-label">Rol (no editable)</label>

                      <input

                        type="text"

                        className="form-control"

                        value={user.nombre\_rol}

                        readOnly

                      />

                    </div>

                    <div className="mb-3">

                      <label className="form-label">Área de interés</label>

                      <input

                        type="text"

                        className="form-control"

                        name="area\_interes"

                        value={user.area\_interes}

                        onChange={handleChange}

                        readOnly={!isEditing}

                      />

                    </div>

                    <div className="text-center">

                      <button

                        type="button"

                        className="btn btn-dark px-4 py-2 rounded-3"

                        onClick={handleEdit}

                      >

                        {isEditing ? "Guardar Cambios" : "Editar Perfil"}

                      </button>

                    </div>

                  </form>

                </div>

              </div>

            </div>

          )}

          {activeSection === "upload" && (

            <ResourceUpload

              courses={courses}

              categories={categories}

              onUploadSuccess={handleUploadSuccess}

            />

          )}

          {activeSection === "resources" && userId && (

            <ResourceList

              userId={userId}

              courses={courses}

              categories={categories}

              onDelete={handleDeleteResource}

              onUpdate={handleUpdateResource}

            />

          )}

          {activeSection === "shared" && userId && (

            <SharedResouces userId={userId} />

          )}

        </div>

      </div>

      {/\* Estilo sidebar colapsado \*/}

      <style>{`

        .sidebar.collapsed {

          margin-left: -220px;

        }

        .sidebar {

          transition: all 0.3s ease;

        }

        .nav-link:hover {

          background-color: rgba(255, 255, 255, 0.1);

        }

      `}</style>

    </div>

  );

}

export default Dashboard;

ResourceList.jsx:

// src/components/docente/ResourceList.jsx

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { confirmAlert } from 'react-confirm-alert';

import 'react-confirm-alert/src/react-confirm-alert.css';

const ResourceList = ({ userId }) => {

  const [resources, setResources] = useState([]);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState('');

  const [editingId, setEditingId] = useState(null);

  const [editForm, setEditForm] = useState({

    titulo: '',

    descripcion: '',

    id\_categoria: '',

    id\_curso: ''

  });

  const token = localStorage.getItem('token');

  useEffect(() => {

    const fetchResources = async () => {

      try {

        setLoading(true);

        setError(null);

        const response = await axios.get(

          'http://localhost:3000/api/docente/recursos',

          {

            headers: {

              'Authorization': `Bearer ${token}`,

              'Cache-Control': 'no-cache'

            }

          }

        );

        if (!response.data || !Array.isArray(response.data)) {

          throw new Error('Formato de respuesta inválido');

        }

        setResources(response.data);

      } catch (err) {

        console.error('Error al cargar recursos:', {

          error: err,

          response: err.response

        });

        setError(err.response?.data?.message ||

                'Error al cargar los recursos. Intente recargar la página.');

      } finally {

        setLoading(false);

      }

    };

    fetchResources();

  }, [token, userId]);

  const handleDelete = async (resourceId) => {

    try {

      await axios.delete(`http://localhost:3000/api/docente/eliminar-recurso/${resourceId}`, {

        headers: { Authorization: `Bearer ${token}` }

      });

      setResources(resources.map(resource =>

        resource.id\_recurso === id ? {

          ...resource,

          titulo: editForm.titulo,

          descripcion: editForm.descripcion,

          id\_categoria: editForm.id\_categoria,

          id\_curso: editForm.id\_curso,

          nombre\_categoria: resources.find(r => r.id\_categoria === editForm.id\_categoria)?.nombre\_categoria || resource.nombre\_categoria,

          nombre\_curso: resources.find(r => r.id\_curso === editForm.id\_curso)?.nombre\_curso || resource.nombre\_curso

        } : resource

      ));

      setEditingId(null);

    } catch (err) {

      console.error('Error al editar recurso:', err);

      setError('No se pudo actualizar el recurso');

    }

  };

  const formatDate = (dateString) => {

    const options = { year: 'numeric', month: 'long', day: 'numeric' };

    return new Date(dateString).toLocaleDateString('es-ES', options);

  };

  if (loading) return <div className="text-center mt-4">Cargando...</div>;

  if (error) return <div className="alert alert-danger">{error}</div>;

  return (

    <div className="container mt-4">

      <h2 className="mb-4">Mis Recursos</h2>

      {resources.length === 0 ? (

        <div className="alert alert-info">No has subido ningún recurso aún.</div>

      ) : (

        <div className="table-responsive">

          <table className="table table-striped">

            <thead>

              <tr>

                <th>Título</th>

                <th>Descripción</th>

                <th>Curso</th>

                <th>Categoría</th>

                <th>Tipo</th>

                <th>Fecha</th>

                <th>Acciones</th>

              </tr>

            </thead>

            <tbody>

              {resources.map(resource => (

                <tr key={resource.id\_recurso}>

                  <td>

                    {editingId === resource.id\_recurso ? (

                      <input

                        type="text"

                        className="form-control form-control-sm"

                        name="titulo"

                        value={editForm.titulo}

                        onChange={handleEditChange}

                      />

                    ) : (

                      resource.titulo

                    )}

                  </td>

                  <td>

                    {editingId === resource.id\_recurso ? (

                      <textarea

                        className="form-control form-control-sm"

                        name="descripcion"

                        value={editForm.descripcion}

                        onChange={handleEditChange}

                        rows="2"

                      />

                    ) : (

                      resource.descripcion || 'Sin descripción'

                    )}

                  </td>

                  <td>

                    {editingId === resource.id\_recurso ? (

                      <select

                        className="form-select form-select-sm"

                        name="id\_curso"

                        value={editForm.id\_curso}

                        onChange={handleEditChange}

                      >

                        {resources.map(r => (

                          <option key={r.id\_curso} value={r.id\_curso}>

                            {r.nombre\_curso}

                          </option>

                        ))}

                      </select>

                    ) : (

                      resource.nombre\_curso

                    )}

                  </td>

                  <td>

                    {editingId === resource.id\_recurso ? (

                      <select

                        className="form-select form-select-sm"

                        name="id\_categoria"

                        value={editForm.id\_categoria}

                        onChange={handleEditChange}

                      >

                        {resources.map(r => (

                          <option key={r.id\_categoria} value={r.id\_categoria}>

                            {r.nombre\_categoria}

                          </option>

                        ))}

                      </select>

                    ) : (

                      resource.nombre\_categoria

                    )}

                  </td>

                  <td>{resource.tipo\_archivo}</td>

                  <td>{formatDate(resource.fecha\_subida)}</td>

                  <td>

                    {editingId === resource.id\_recurso ? (

                      <>

                        <button

                          className="btn btn-success btn-sm me-1"

                          onClick={() => saveEdit(resource.id\_recurso)}

                        >

                          Guardar

                        </button>

                        <button

                          className="btn btn-secondary btn-sm"

                          onClick={cancelEditing}

                        >

                          Cancelar

                        </button>

                      </>

                    ) : (

                      <>

                        <button

                          className="btn btn-primary btn-sm me-1"

                          onClick={() => startEditing(resource)}

                        >

                          Editar

                        </button>

                        <button

                          className="btn btn-danger btn-sm"

                          onClick={() => handleDelete(resource.id\_recurso)}

                        >

                          Eliminar

                        </button>

                      </>

                    )}

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      )}

    </div>

  );

};

export default ResourceList;

docenteController.js:

const db = require('../models/db');

const multer = require('multer');

const path = require('path');

const fs = require('fs');

// Configuración de Multer para recursos académicos

const storageRecursos = multer.diskStorage({

  destination: function (req, file, cb) {

    const uploadDir = path.resolve(\_\_dirname, '..', 'uploads', 'recursos');

    console.log('Ruta absoluta donde se guardará:', uploadDir);

    if (!fs.existsSync(uploadDir)) {

      fs.mkdirSync(uploadDir, { recursive: true });

    }

    cb(null, uploadDir);

  },

  filename: function (req, file, cb) {

    const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() \* 1E9);

    cb(null, uniqueSuffix + path.extname(file.originalname));

  }

});

const uploadRecurso = multer({

  storage: storageRecursos,

  fileFilter: (req, file, cb) => {

    const allowedTypes = [

      'application/pdf',

      'application/vnd.openxmlformats-officedocument.wordprocessingml.document',

      'application/msword' // Añadir soporte para DOC antiguo

    ];

    if (allowedTypes.includes(file.mimetype)) {

      cb(null, true);

    } else {

      cb(new Error('Solo se permiten archivos PDF o DOCX'), false);

    }

  },

  limits: {

    fileSize: 25 \* 1024 \* 1024, // Aumentado a 25MB para margen

    files: 1,

    parts: 10 // Limitar partes del formulario

  }

});

exports.uploadRecurso = uploadRecurso;

// Obtener recursos del docente

exports.getRecursosDocente = async (req, res) => {

  try {

    const docenteId = req.user.id;

    const query = `

      SELECT r.\*, c.nombre\_curso, cat.nombre\_categoria

      FROM recursos r

      JOIN cursos c ON r.id\_curso = c.id\_curso

      JOIN categorias cat ON r.id\_categoria = cat.id\_categoria

      WHERE r.id\_docente = ?

      ORDER BY r.fecha\_subida DESC

    `;

    const [recursos] = await db.promise().query(query, [docenteId]);

    res.json(recursos);

  } catch (error) {

    console.error('Error al obtener recursos:', error);

    res.status(500).json({ message: 'Error al obtener recursos' });

  }

};

// Modifica la función subirRecurso en docenteController.js

exports.subirRecurso = async (req, res) => {

  try {

    console.log('Iniciando subida de recurso...'); // Log de depuración

    console.log('Cuerpo de la solicitud:', req.body); // Ver qué datos llegan

    console.log('Archivo recibido:', req.file); // Ver el archivo recibido

    const docenteId = req.user.id;

    const { titulo, descripcion, id\_curso, id\_categoria } = req.body;

    if (!req.file) {

      console.error('No se recibió archivo');

      return res.status(400).json({ message: 'Debes subir un archivo' });

    }

    // Validar tipos de archivo permitidos

    const allowedTypes = ['application/pdf', 'application/vnd.openxmlformats-officedocument.wordprocessingml.document'];

    if (!allowedTypes.includes(req.file.mimetype)) {

      console.error('Tipo de archivo no permitido:', req.file.mimetype);

      return res.status(400).json({ message: 'Solo se permiten archivos PDF o DOCX' });

    }

    const tipoArchivo = req.file.mimetype === 'application/pdf' ? 'PDF' : 'DOCX';

    const query = `

      INSERT INTO recursos

      (titulo, descripcion, archivo\_url, tipo\_archivo, id\_docente, id\_curso, id\_categoria)

      VALUES (?, ?, ?, ?, ?, ?, ?)

    `;

    console.log('Ejecutando query con:', [titulo, descripcion, `recursos/${req.file.filename}`, tipoArchivo, docenteId, id\_curso, id\_categoria]);

    const [result] = await db.promise().query(query, [

      titulo,

      descripcion,

      `recursos/${req.file.filename}`,

      tipoArchivo,

      docenteId,

      id\_curso,

      id\_categoria

    ]);

    // Agregar al docente como autor

    await db.promise().query(

      'INSERT INTO autores\_recursos (id\_recurso, id\_autor) VALUES (?, ?)',

      [result.insertId, docenteId]

    );

    console.log('Recurso subido exitosamente con ID:', result.insertId);

    res.status(201).json({

      message: 'Recurso subido exitosamente',

      id\_recurso: result.insertId

    });

  } catch (error) {

    console.error('Error detallado al subir recurso:', error);

    res.status(500).json({

      message: 'Error al subir recurso',

      error: error.message,

      stack: error.stack

    });

  }

};

// Obtener recursos compartidos con el docente

exports.getRecursosCompartidos = async (req, res) => {

  try {

    const docenteId = req.user.id;

    const query = `

      SELECT r.\*, c.nombre\_curso, cat.nombre\_categoria,

             u.nombre\_completo AS nombre\_autor

      FROM recursos r

      JOIN cursos c ON r.id\_curso = c.id\_curso

      JOIN categorias cat ON r.id\_categoria = cat.id\_categoria

      JOIN autores\_recursos ar ON r.id\_recurso = ar.id\_recurso

      JOIN usuarios u ON ar.id\_autor = u.id\_usuario

      WHERE r.id\_docente != ? AND ar.id\_autor = ?

      ORDER BY r.fecha\_subida DESC

    `;

    const [recursos] = await db.promise().query(query, [docenteId, docenteId]);

    res.json(recursos);

  } catch (error) {

    console.error('Error al obtener recursos compartidos:', error);

    res.status(500).json({ message: 'Error al obtener recursos compartidos' });

  }

};

// Obtener cursos y categorías para dropdowns

exports.getCursosYCategorias = async (req, res) => {

  try {

    const [cursos] = await db.promise().query('SELECT \* FROM cursos ORDER BY nombre\_curso');

    const [categorias] = await db.promise().query('SELECT \* FROM categorias ORDER BY nombre\_categoria');

    res.json({ cursos, categorias });

  } catch (error) {

    console.error('Error al obtener datos:', error);

    res.status(500).json({ message: 'Error al obtener datos' });

  }

};

// Eliminar recurso (solo si es del docente)

exports.eliminarRecurso = async (req, res) => {

  try {

    const docenteId = req.user.id;

    const { id\_recurso } = req.params;

    // Verificar que el recurso pertenece al docente

    const [recurso] = await db.promise().query(

      'SELECT id\_docente FROM recursos WHERE id\_recurso = ?',

      [id\_recurso]

    );

    if (recurso.length === 0) {

      return res.status(404).json({ message: 'Recurso no encontrado' });

    }

    if (recurso[0].id\_docente !== docenteId) {

      return res.status(403).json({ message: 'No autorizado para eliminar este recurso' });

    }

    // Eliminar (ON DELETE CASCADE se encargará de autores\_recursos)

    await db.promise().query(

      'DELETE FROM recursos WHERE id\_recurso = ?',

      [id\_recurso]

    );

    res.json({ message: 'Recurso eliminado exitosamente' });

  } catch (error) {

    console.error('Error al eliminar recurso:', error);

    res.status(500).json({ message: 'Error al eliminar recurso' });

  }

};

// Actualizar recurso existente

exports.actualizarRecurso = async (req, res) => {

  try {

    const docenteId = req.user.id;

    const { id\_recurso } = req.params;

    const { titulo, descripcion } = req.body;

    // Verificar que el recurso pertenece al docente

    const [recurso] = await db.promise().query(

      'SELECT id\_docente FROM recursos WHERE id\_recurso = ?',

      [id\_recurso]

    );

    if (recurso.length === 0) {

      return res.status(404).json({ message: 'Recurso no encontrado' });

    }

    if (recurso[0].id\_docente !== docenteId) {

      return res.status(403).json({ message: 'No autorizado para editar este recurso' });

    }

    // Actualizar el recurso

    await db.promise().query(

      'UPDATE recursos SET titulo = ?, descripcion = ? WHERE id\_recurso = ?',

      [titulo, descripcion, id\_recurso]

    );

    // Obtener el recurso actualizado con información adicional

    const [updatedRecurso] = await db.promise().query(`

      SELECT r.\*, c.nombre\_curso, cat.nombre\_categoria

      FROM recursos r

      JOIN cursos c ON r.id\_curso = c.id\_curso

      JOIN categorias cat ON r.id\_categoria = cat.id\_categoria

      WHERE r.id\_recurso = ?

    `, [id\_recurso]);

    res.json({

      message: 'Recurso actualizado exitosamente',

      recurso: updatedRecurso[0]

    });

  } catch (error) {

    console.error('Error al actualizar recurso:', error);

    res.status(500).json({ message: 'Error al actualizar recurso' });

  }

};

docenteRoutes.js:

const express = require('express');

const router = express.Router();

const docenteController = require('../controllers/docenteController');

const { verifyToken, isDocente } = require('../middlewares/authMiddleware');

// Middleware para verificar token y rol de docente

router.use(verifyToken);

router.use(isDocente);

// Rutas para recursos del docente

router.get('/recursos', docenteController.getRecursosDocente);

router.get('/recursos-compartidos', docenteController.getRecursosCompartidos);

router.get('/datos-utiles', docenteController.getCursosYCategorias);

// Ruta para subir recurso (con middleware de multer)

router.post('/subir-recurso',

  docenteController.uploadRecurso.single('archivo'),

  docenteController.subirRecurso

);

// Ruta para eliminar recurso

router.delete('/eliminar-recurso/:id\_recurso', docenteController.eliminarRecurso);

// Ruta para actualizar recursos

router.put('/recurso/:id\_recurso', docenteController.actualizarRecurso);

module.exports = router;