

// src/config/db.js

import pg from 'pg';

import dotenv from 'dotenv';

dotenv.config();

const { Pool } = pg;

export const pool = new Pool({

  host:     process.env.DB\_HOST,

  port:     process.env.DB\_PORT,

  user:     process.env.DB\_USER,

  password: process.env.DB\_PASSWORD,

  database: process.env.DB\_NAME,

});

export async function testDB() {

  try {

    const client = await pool.connect();

    console.log('✅ Conexión exitosa a PostgreSQL');

    client.release();

  } catch (err) {

    console.error('❌ Error conectando a la BD:', err);

    process.exit(1);

  }

}

// src/controllers/auth.controller.js

import bcrypt from "bcryptjs";

import jwt from "jsonwebtoken";

import { pool } from "../config/db.js";

import { transporter } from "../service/mail.service.js";

const JWT\_SECRET = process.env.JWT\_SECRET || "un\_secreto\_muy\_seguro\_para\_jwt";

const saltRounds = 10;

export async function login(req, res) {

  const { email, password } = req.body;

  try {

    const { rows } = await pool.query(

      `SELECT u.id\_usuario, u.nombre, u.apellido, u.contraseña, r.nombre AS categoria

      FROM usuarios u

      JOIN roles r ON u.rol\_id = r.id\_rol

      WHERE u.email = $1`, [email]);

    if (!rows.length)

      return res.status(401).json({ message: "Credenciales inválidas" });

    const user = rows[0];

    const match = await bcrypt.compare(password, user.contraseña);

    if (!match)

      return res.status(401).json({ message: "Credenciales inválidas" });

    const token = jwt.sign(

      { id: user.id\_usuario, categoria: user.categoria },

      JWT\_SECRET,

      { expiresIn: "1h" }

    );

    res.json({

      token,

      user: {

        id: user.id\_usuario,

        nombre: user.nombre,

        apellido: user.apellido,

        categoria: user.categoria,

      },

    });

  } catch (err) {

    console.error("💥 login:", err);

    res.status(500).json({ message: "Error del servidor" });

  }

}

export async function forgotPassword(req, res) {

  const { email } = req.body;

  try {

    const { rows } = await pool.query(

      "SELECT id\_usuario, nombre FROM usuarios WHERE email=$1 AND activo=true",

      [email]

    );

    if (!rows.length)

      return res

        .status(200)

        .json({ message: "Si el email está registrado se enviará un enlace." });

    const user = rows[0];

    const resetToken = jwt.sign({ id: user.id\_usuario }, JWT\_SECRET, {

      expiresIn: "1h",

    });

    await pool.query(

      "UPDATE tokensrecuperacion SET usado=true WHERE usuario\_id=$1 AND usado=false",

      [user.id\_usuario]

    );

    await pool.query(

      "INSERT INTO tokensrecuperacion (usuario\_id, token, fecha\_creacion, fecha\_expiracion, usado) VALUES ($1,$2,NOW(),NOW()+INTERVAL '1 hour',false)",

      [user.id\_usuario, resetToken]

    );

    const resetUrl = `http://localhost:5173/reset-password?token=${resetToken}`;

    await transporter.sendMail({

      from: process.env.EMAIL\_USER,

      to: email,

      subject: "Restablecer contraseña • Diario Virtual",

      html: `<p>Hola ${user.nombre}, haz clic aquí para restablecer tu contraseña:</p><a href="${resetUrl}">${resetUrl}</a>`,

    });

    res.json({ message: "Si el email está registrado se enviará un enlace." });

  } catch (err) {

    console.error("💥 forgotPassword:", err);

    res.status(500).json({ message: "Error del servidor" });

  }

}

export async function resetPassword(req, res) {

  const { token, newPassword } = req.body;

  try {

    const { id } = jwt.verify(token, JWT\_SECRET);

    const { rows } = await pool.query(

      "SELECT \* FROM tokensrecuperacion WHERE token=$1 AND usuario\_id=$2 AND usado=false AND fecha\_expiracion>NOW()",

      [token, id]

    );

    if (!rows.length)

      return res.status(400).json({ message: "Enlace inválido o expirado" });

    const hash = await bcrypt.hash(newPassword, saltRounds);

    await pool.query("UPDATE usuarios SET contraseña=$1 WHERE id\_usuario=$2", [

      hash,

      id,

    ]);

    await pool.query(

      "UPDATE tokensrecuperacion SET usado=true WHERE token=$1",

      [token]

    );

    res.json({ message: "Contraseña restablecida" });

  } catch (err) {

    console.error("💥 resetPassword:", err);

    res.status(500).json({ message: "Error del servidor" });

  }

}

// src/routes/auth.routes.js

import { Router } from 'express';

import { login, forgotPassword, resetPassword } from '../controllers/auth.controller.js';

const router = Router();

router.post('/login', login);

router.post('/forgot-password', forgotPassword);

router.post('/reset-password', resetPassword);

export default router;

// src/app.js

import express from 'express';

import cors from 'cors';

import authRoutes from './routes/auth.routes.js';

import userRoutes from './routes/user.routes.js';

import errorHandler from './middlewares/error.middleware.js';

import articleRoutes from './routes/article.routes.js';

const app = express();

app.use(cors());

app.use(express.json());

app.get('/', (\_req, res) => res.send('Backend Diario Virtual funcionando 👌'));

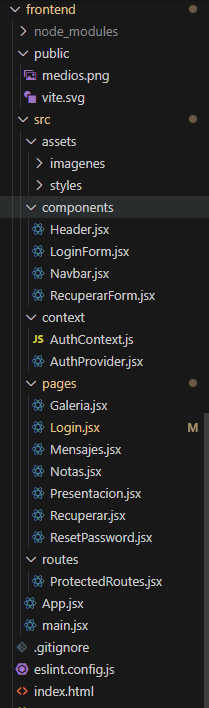
app.use('/api/auth', authRoutes);

app.use('/api', userRoutes);

app.use('/api/articles',articleRoutes);

app.use(errorHandler);   // siempre al final

export default app;



import { useState, useContext } from 'react';

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

import LoginForm from '../components/LoginForm';

import '../assets/styles/login.css';

import AuthContext from '../context/AuthContext.js';

function Login() {

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

  const navigate = useNavigate();

  const { login } = useContext(AuthContext);

  const handleLogin = async (credentials) => {

    try {

      console.log('🔄 Enviando credenciales:', credentials);

      const response = await fetch('http://localhost:5000/api/auth/login', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify({

          email: credentials.email,

          password: credentials.password

        }),

      });

      const data = await response.json();

      console.log('📡 Respuesta del servidor:', data);

      if (response.ok) {

        console.log('✅ Login exitoso:', data.user);

        // Actualizar contexto (y localStorage dentro del provider)

        login(data.user);

        // Redirigir según el rol del usuario

        const rol = data.user.categoria;

        console.log('Redirigiendo según el rol:', rol);

        if (rol === 'Periodista') {

          navigate('/Notas');

        } else if (rol === 'Fotografo') {

          navigate('/galeria');

        } else if (rol === 'Editor') {

          navigate('/editor');

        } else {

          navigate('/Dashboard'); // ruta por defecto

        }

      } else {

        setError(data.message || 'Credenciales incorrectas');

      }

    } catch (error) {

      console.error('💥 Error al intentar login:', error);

      setError('Error de conexión con el servidor');

    }

  };

  return (

    <div className="login-page">

      <div className="login-container">

        {error && <div className="error-message">{error}</div>}

        <LoginForm onLogin={handleLogin} />

      </div>

    </div>

  );

}

export default Login;

import { useState, useEffect } from 'react';

import '../assets/styles/notas.css';

function Notas() {

  const [notas, setNotas] = useState([]);

  useEffect(() => {

    // Simulamos que cada nota tiene una URL de archivo asociada

    const notasSimuladas = [

      {

        id: 1,

        titulo: "Entrevista con Ministro",

        estado: "Borrador",

        fechaCreacion: "2025-07-01",

        fechaModificacion: "2025-07-15",

        url\_archivo: "/archivos/archivo1.pdf", // En /public/archivos/nota1.pdf

      },

      {

        id: 2,

        titulo: "Informe sobre Educación",

        estado: "Listo",

        fechaCreacion: "2025-06-20",

        fechaModificacion: "2025-07-05",

        url\_archivo: "/archivos/Tp 9.docx", // En /public/archivos/nota2.pdf

      },

    ];

    setNotas(notasSimuladas);

  }, []);

  const actualizarEstado = (idNota, nuevoEstado) => {

    const nuevasNotas = notas.map((nota) =>

      nota.id === idNota ? { ...nota, estado: nuevoEstado } : nota

    );

    setNotas(nuevasNotas);

  };

  const verArchivo = (url) => {

    window.open(url, '\_blank');

  };

  return (

    <div className="contenedor-notas">

      <h2>Mis Notas</h2>

      <table className="tabla-notas">

        <thead>

          <tr>

            <th>Título</th>

            <th>Estado</th>

            <th>Fecha de creación</th>

            <th>Última modificación</th>

            <th>Acciones</th>

          </tr>

        </thead>

        <tbody>

          {notas.map((nota) => (

            <tr key={nota.id}>

              <td>{nota.titulo}</td>

              <td>

                <select

                  value={nota.estado}

                  onChange={(e) => actualizarEstado(nota.id, e.target.value)}

                >

                  <option value="Borrador">Borrador</option>

                  <option value="Listo">Listo</option>

                  <option value="En revisión">En revisión</option>

                </select>

              </td>

              <td>{nota.fechaCreacion}</td>

              <td>{nota.fechaModificacion}</td>

              <td>

                <button className="btn-descargar">📥 Descargar</button>

                <button

                  className="btn-ver"

                  onClick={() => verArchivo(nota.url\_archivo)}

                >

                  👁 Ver

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

}

export default Notas;

// src/App.jsx (o donde tengas tus rutas)

import React from 'react';

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

import Login from './pages/Login';

import Recuperar from './pages/Recuperar';

import ResetPassword from './pages/ResetPassword';

import Presentacion from './pages/Presentacion';

import Navbar from './components/Navbar';

import ProtectedRoute from './routes/ProtectedRoutes';

import Notas from './pages/Notas';

import Galeria from './pages/Galeria';

import Mensajes from './pages/Mensajes';

function App() {

  return (

     <>

      <Navbar /> {/\* visible siempre \*/}

      <Routes>

        {/\* Define tus rutas aquí sin envolverlas en Router \*/}

        <Route path="/" element={<Presentacion />} />

        <Route path="/login" element={<Login />} />

        <Route path="/recuperar" element={<Recuperar />} />

        <Route path="/reset-password" element={<ResetPassword />} />

        <Route element={<ProtectedRoute allow={['Periodista']} />}>

          <Route path="/notas" element={<Notas />} />

        </Route>

        <Route element={<ProtectedRoute allow={['fotografo']} />}>

          <Route path="/galeria" element={<Galeria />} />

        </Route>

        <Route element={<ProtectedRoute allow={['periodista', 'fotografo', 'editor']} />}>

          <Route path="/mensajes" element={<Mensajes />} />

        </Route>

        <Route path="/no-autorizado" element={<h2>No autorizado</h2>} />

      </Routes>

    </>

  );

}

export default App;

import { Navigate, Outlet } from 'react-router-dom';

import { useContext } from 'react';

import AuthContext from '../context/AuthContext.js';

export default function ProtectedRoute({ allow }) {

  const { usuario } = useContext(AuthContext);

  if (!usuario)         return <Navigate to="/login" replace />;

  if (!allow.includes(usuario.categoria))

                        return <Navigate to="/no-autorizado" replace />;

  return <Outlet />;

}

