const mongoose = require("mongoose");

const Joi = require("joi");

// ✅ Schéma Mongoose Reservation

const ReservationSchema = new mongoose.Schema({

    formation: {

        type: mongoose.Schema.Types.ObjectId,

        ref: "Formation", // 🔥 Référence par le nom du modèle

        required: true

    },

    employe: {

        type: mongoose.Schema.Types.ObjectId,

        ref: "User", // 🔥 Référence à ton modèle User

        required: true

    },

    status: {

        type: String,

        enum: ["en\_attente", "confirmee", "annulee"],

        default: "en\_attente"

    },

    dateReservation: {

        type: Date,

        default: Date.now // 🔥 sans ()

    }

}, { timestamps: true });

// ✅ Validation avec Joi

function ValidationDemandeReservation(obj) {

    const schema = Joi.object({

        formation: Joi.string().required(),

    });

    return schema.validate(obj);

}

// ✅ Validation avec Joi

function ValidationReservation(obj) {

    const schema = Joi.object({

        status: Joi.string().valid("en\_attente", "confirmee", "annulee").required()

    });

    return schema.validate(obj); // ⚠️ Assure-toi de RETURN

}

const Reservation = mongoose.model("Reservation", ReservationSchema);

module.exports = {

    Reservation,

    ValidationDemandeReservation,

    ValidationReservation

};

const {Reservation,ValidationDemandeReservation,ValidationReservation} =require("../models/reservation") ;

const asyncHandler = require("express-async-handler")

const {sendReservationStatusEmail} = require("../utils/mailer");

require("dotenv").config();

/\*\*

 \* @desc    Ajouter une nouvelle réservation

 \* @route   POST /api/reservations

 \* @access  Employé

 \* @desc Demander de reserver un formation a l admin

 \*/

const addReservation = asyncHandler(async(req,res)=>{

    // Validation des données entrantes

    const { error } = ValidationDemandeReservation(req.body);

    if (error) {

       return res.status(400).json({ message: error.details[0].message });

    }

    //verify token and get user id

    const employeId = req.user.id; // Assure-toi que le middleware d'authentification est en place

    if(!employeId || req.user.role !=='employe') {

        return res.status(401).json({message:"Non autorisé"});

    }

    // Créer une nouvelle réservation

    const newReservation = new Reservation({

        formation: req.body.formation,

        employe: employeId,

        status: "en\_attente", // Statut par défaut

        dateReservation: Date.now() // Par défaut, la date actuelle

    })

    // Sauvegarder la réservation dans la base de données

    await newReservation.save();

    res.status(201).json({message:"Demande de réservation envoyée avec succès", reservation:newReservation })

})

/\*\*

 \* @desc    Mettre à jour le statut d'une réservation par l'admin

 \* @route   PUT /api/reservations/:id

 \* @access  Admin

 \* @desc L'admin peut confirmer ou annuler une réservation

 \*/

const updateReservationStatus = asyncHandler(async(req,res)=>{

    // Validation des données entrantes

    const {error}=ValidationReservation(req.body)

    if(error) return res.status(400).json({message:error.details[0].message});

    //verify token

    if(req.user.role !=='admin') {

        return res.status(401).json({message:"Non autorisé"});

    }

    // Trouver la réservation par ID et peupler employe et formation

    const reservation = await Reservation.findById(req.params.id)

        .populate('employe','email nom')

        .populate('formation',  'titre');

    if(!reservation){

        return res.status(404).json({message:"Réservation non trouvée"})

    }

    // Mettre à jour le statut

    reservation.status = req.body.status;

    await reservation.save();

    // Envoyer un email à l'employé pour l'informer du changement de statut

    await sendReservationStatusEmail(

        reservation.employe.email,

        reservation.employe.nom,

        reservation.formation.titre,

        reservation.status

    );

    res.status(200).json({message:"Statut de la réservation mis à jour avec succès", reservation})

})

module.exports = {

    addReservation,

    updateReservationStatus

};

const express = require('express');

const router = express.Router();

const { addReservation, updateReservationStatus } = require('../controllers/reservationController');

const { verifyToken } = require('../middleware/verifyToken');

// Route pour ajouter une nouvelle réservation (employé)

router.post('/', verifyToken, addReservation);

// Route pour mettre à jour le statut d'une réservation (admin)

router.put('/:id', verifyToken, updateReservationStatus);

module.exports = router;

const jwt = require("jsonwebtoken");

require("dotenv").config();

function verifyToken(req, res, next) {

    const token = req.query.token || req.headers['x-access-token'];

    if (!token) {

        return res.status(403).json({ message: "Aucun token fourni." });

    }

    try {

        const decoded = jwt.verify(token, process.env.JWT\_SECRET\_KEY);

        req.user = decoded; // Stocke les données décodées dans `req.user`

        next();

    } catch (error) {

        return res.status(401).json({ message: "Token invalide ou expiré." });

    }

}

function generateToken(user) {

    return jwt.sign(

        { message : "login reussie",

          id: user.\_id, email: user.email, role: user.role },

          process.env.JWT\_SECRET\_KEY,

          { expiresIn: "7d" } //expiration dans 7 days

    );

}

module.exports = { verifyToken, generateToken };

async function sendReservationStatusEmail(email, nom, formation, status) {

    const statusMessage = status === 'confirmee'

        ? `Votre réservation pour la formation "${formation}" a été confirmée.`

        : `Votre réservation pour la formation "${formation}" a été annulée.`;

    const html = `

        <div>

            <h2 style="color: #333;">Bonjour ${nom},</h2>

            <p>${statusMessage}</p>

            <p>Cordialement,</p>

            <p>L'équipe admin</p>

        </div>

    `;

    // Envoi de l'email

    return sendMail(email, "Mise à jour de votre réservation", html);

}

async function sendMail(to, subject, html) {

    try {

        await transporter.sendMail({

            from: `"Support" <${process.env.EMAIL\_USER}>`,

            to,

            subject,

            html,

        });

        console.log(`📧 Email envoyé à ${to}`);

        return true;

    } catch (err) {

        console.error("Erreur envoi email:", err);

        throw new Error("Erreur lors de l'envoi du mail");

    }

}