const mongoose = require("mongoose");

const Joi = require("joi");

// ✅ Schéma Mongoose optimisé

const userSchema = new mongoose.Schema({

    nom: { type: String, required: true, minlength: 5 },

    email: { type: String, required: true, minlength: 5, unique: true, lowercase: true },

    password: { type: String, required: true, minlength: 8 },

    role: { type: String, enum: ['employe', 'rh'], default: 'employe' },

    competence: [String],

    statut: {

        type: String,

        enum: ['en\_attente', 'actif', 'inactif'],

        default: 'en\_attente'

    },

    validationCode: { type: String },

    validationCodeExpires: { type: Date },

    emailVerified: { type: Boolean, default: false }, // Nouveau champ pour vérifier l'email

    enLigne: { type: Boolean, default: false }

}, { timestamps: true });

// ✅ Validation pour l'enregistrement

function valid\_Enregistrement(obj) {

    const schema = Joi.object({

        nom: Joi.string().min(5).required(),

        email: Joi.string().email().min(5).required(),

        password: Joi.string().min(8).required(),

        role: Joi.string().valid('employe', 'rh').default('employe')

    });

    return schema.validate(obj);

}

// ✅ Validation pour la connexion

function valid\_connexion(obj) {

    const schema = Joi.object({

        email: Joi.string().email().min(5).required(),

        password: Joi.string().min(8).required()

    });

    return schema.validate(obj);

}

// ✅ Validation pour la mise à jour d'un utilisateur

function valid\_mise\_a\_jour(obj) {

    const schema = Joi.object({

        nom: Joi.string().min(5),

        email: Joi.string().email().min(5),

        password: Joi.string().min(8),

        role: Joi.string().valid('employe', 'rh'),

        statut: Joi.string().valid('en\_attente', 'actif', 'inactif')

    });

    return schema.validate(obj);

}

// ✅ Fonction simplifiée pour valider un compte utilisateur

function valid\_activation\_compte(obj) {

    const schema = Joi.object({

        statut: Joi.string().valid('actif', 'inactif').required()

    });

    return schema.validate(obj);

}

//✅ Fonction simplifiée  pour valider le code de comfirmation

function valid\_code\_validation(obj){

    const schema = Joi.object({

        email: Joi.string().email().min(5).required(),

        code: Joi.string().length(6).required(),

    });

    return  schema.validate(obj);

}

const User = mongoose.model("User", userSchema);

module.exports = {

    User,

    valid\_Enregistrement,

    valid\_connexion,

    valid\_mise\_a\_jour,

    valid\_activation\_compte,valid\_code\_validation

};

    const {User,valid\_connexion,valid\_code\_validation} = require("../models/user")

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

    const bcrypt = require("bcrypt");

    require("dotenv").config();

    const crypto = require("crypto");

    const { invalid } = require("joi");

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

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

    //Post login user

    /\*\*

     \* @file loginController.js

     \* @description Gère l'enregistrement des utilisateurs (route POST /login)

     \* @route /api/userLogin/login

     \* @requires express

     \* @requires bcrypt

     \* @requires express-async-handler

     \* @requires ../models/user (modèle Mongoose + fonctions de validation)

     \*/

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

        // 1. Validation des données

        const { error } = valid\_connexion(req.body);

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

        // 2. Recherche de l'utilisateur

        let user = await User.findOne({ email: req.body.email });

        if (!user) return res.status(400).json({ message: "Utilisateur non trouvé" });

        user.enLigne = true ;

        user.save();

        // 3. Vérification du mot de passe

        const isPasswordMatch = await bcrypt.compare(req.body.password, user.password);

        if (!isPasswordMatch) return res.status(400).json({ message: "Mot de passe invalide" });

        // Récupérer IP et User-Agent

        const ip = req.ip || req.connection.remoteAddress || "IP non disponible";

        const userAgent = req.headers['user-agent'] || "Agent utilisateur non disponible";

        // Envoyer mail notification

        sendLoginNotification(user.email, ip, userAgent).catch(console.error);

        // ✅ Réponse en cas de succès (ex: sans token pour l'instant)

        const token = generateToken(user);

        res.json({

            message: "Connexion réussie",

            token,

            user: {

                id: user.\_id,

                nom: user.nom,

                email: user.email,

                role: user.role,

                statut: user.statut,

                enLigne : user.enLigne

            }

        });

    })

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

      const user = await User.findById(req.user.id);

        if (!user) {

            return res.status(404).json({ message: "Utilisateur non trouvé" });

        }

        if (!user.enLigne) {

            return res.status(400).json({ message: "Ce compte est déjà déconnecté" });

        }

        user.enLigne = false;

        await user.save();

        res.json({

            message: "Déconnexion réussie",

            user: {

                id: user.\_id,

                email: user.email,

                nom: user.nom,

                enLigne: user.enLigne

            }

        });

    })

    module.exports = {loginUser,logoutUser}

const express = require("express")

const router = express.Router();

const {loginUser,logoutUser}= require("../controllers/loginController")

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

// Route POST pour la connexion

router.post("/login", loginUser);

// Route POST pour la déconnexion (protégée par token)

router.post("/logout", verifyToken, logoutUser);

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 };

const nodemailer = require("nodemailer");

require("dotenv").config();

const transporter = nodemailer.createTransport({

    service: "gmail",

    auth: {

        user: process.env.EMAIL\_USER,

        pass: process.env.EMAIL\_PASS,

    },

});

// Fonction pour envoyer l'email de confirmation d'inscription

async function sendRegistrationEmail(email, nom, validationCode) {

    const mailOption = {

        from: process.env.EMAIL\_USER,

        to: email,

        subject: "Confirmez votre adresse email",

        html: `

        <div style="font-family: Arial, sans-serif; max-width: 600px; margin: 0 auto;">

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

            <p>Votre compte a été créé avec succès. Veuillez vérifier votre adresse email en utilisant le code suivant :</p>

            <h3 style="color: #007bff;">${validationCode}</h3>

            <p>Ce code est valide pendant 24 heures. Une fois votre email vérifié, un responsable RH examinera votre compte pour l'activer.</p>

            <p>Cordialement,</p>

            <p>L'équipe RH</p>

        </div>

        `,

    };

    try {

        await transporter.sendMail(mailOption);

        console.log(`Email envoyé à ${email} avec le code ${validationCode}`);

    } catch (error) {

        console.error("Erreur lors de l'envoi de l'email :", error);

        throw new Error("Erreur lors de l'envoi de l'email de confirmation.");

    }

}

   // Fonction pour envoyer l'email de validation réussie

   async function sendValidationSuccessEmail(email, nom) {

    const mailOption = {

        from: process.env.EMAIL\_USER,

        to: email,

        subject: "Validation de votre adresse email réussie",

        html: `

        <div style="font-family: Arial, sans-serif; max-width: 600px; margin: 0 auto;">

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

            <p>Votre adresse email a été vérifiée avec succès !</p>

            <p>Votre compte est actuellement en attente d'approbation par un responsable RH. Vous recevrez une notification une fois votre compte activé.</p>

            <p>Cordialement,</p>

            <p>L'équipe RH</p>

        </div>

        `,

    };

    try {

        await transporter.sendMail(mailOption);

        console.log(`Email de validation réussie envoyé à ${email}`);

    } catch (error) {

        console.error("Erreur lors de l'envoi de l'email de validation réussie :", error);

        throw new Error("Erreur lors de l'envoi de l'email de validation réussie.");

    }

   }

    //detecter le login

    async function sendLoginNotification(email,ip,userAgent){

        const mailOptions = {

        from: process.env.EMAIL\_USER,

        to: email,

        subject: "Nouvelle connexion détectée",

        text: `Bonjour,

               Nous avons détecté une nouvelle connexion à votre compte depuis un appareil avec les informations suivantes :

               Adresse IP : ${ip}

               Appareil/Navigateur : ${userAgent}

               Date et heure : ${new Date().toLocaleString()}

               Si ce n'était pas vous, merci de vérifier la sécurité de votre compte.

               Cordialement,

               L'équipe de support`,

        };

    return transporter.sendMail(mailOptions);

}

    // Email de validation pour changement d'email

    async function sendValidationUpdate(email,code){

        const mailOptions = {

            from: process.env.EMAIL\_USER,

            to: email,

            subject: "Code de validation pour la modification de votre email",

            html: `

                <h2>Validation de votre nouvelle adresse email</h2>

                <p>Votre code de validation est: <strong>${code}</strong></p>

                <p>Ce code expirera dans 24 heures.</p>

                <p>Si vous n'avez pas demandé de changement d'email, veuillez ignorer ce message.</p>`

            };

            return transporter.sendMail(mailOptions);

    }

    //Email de Confirmation de validation du update

    async function sendValidationConfirmation(email){

        const mailOptions = {

            from: process.env.EMAIL\_USER,

            to: email,

            subject: "Votre email a été validé avec succès",

            text: `

                Validation d'email réussie !

                Bonjour,

                Votre adresse email a été validée avec succès et est maintenant activée sur votre compte.

                Vous pouvez désormais utiliser cette adresse email pour vous connecter à votre compte et recevoir toutes nos communications.

                ✅ Email validé avec succès

                Date de validation: ${new Date().toLocaleString('fr-FR')}

                Si vous n'êtes pas à l'origine de cette action, veuillez contacter immédiatement le support.

                Cordialement,

                L'équipe de support`

            };

            return transporter.sendMail(mailOptions);

    }

   //Email de Confirmation de

async function sendStatusActif(email, status) {

    const subject = status === 'actif'

        ? "Votre compte a été activé"

        : "Votre compte a été désactivé";

    const text = status === 'actif'

        ? `Bonjour, votre compte a été activé avec succès. Vous pouvez maintenant vous connecter.`

        : `Bonjour, votre compte a été désactivé. Contactez l'administration pour plus d'informations.`;

    const mailOptions = {

        from: process.env.EMAIL\_USER,

        to: email,

        subject: subject,

        text: text

    };

    return transporter.sendMail(mailOptions);

}

module.exports = { transporter,

    sendRegistrationEmail,

    sendValidationSuccessEmail,

    sendLoginNotification,

    sendValidationUpdate ,

    sendValidationConfirmation ,

    sendStatusActif };