Voila tout la registration code :

Models/user.js

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

}, { 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

};

Utils/mailer.js

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.");

    }

}

module.exports = { transporter, sendRegistrationEmail,sendValidationSuccessEmail };

controllers/registerController.js

const { User,valid\_Enregistrement,valid\_connexion,valid\_mise\_a\_jour,valid\_activation\_compte,valid\_code\_validation} = require("../models/user");

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

const bcrypt = require("bcrypt");

const { sendRegistrationEmail,sendValidationSuccessEmail } = require("../utils/mailer");

require("dotenv").config();

const crypto = require("crypto");

//Post register user

/\*\*

 \* @file registerController.js

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

 \* @route /api/userRegister/register

 \* @requires express

 \* @requires bcrypt

 \* @requires express-async-handler

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

 \*/

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

    // 1. Validation

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

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

    // 2. Vérifier si l'utilisateur existe déjà

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

    if (user) return res.status(400).send({ message: "This user is already registered" });

    // 3. Hacher le mot de passe

    const salt = await bcrypt.genSalt(10);

    const hashedPassword = await bcrypt.hash(req.body.password, salt);

    // 4. Générer un code de validation à 6 chiffres

    const validationCode = crypto.randomInt(100000, 999999).toString() // Code à 6 chiffres

    const validationCodeExpires = Date.now() +  5 \* 60 \* 1000; // Expire dans 5 minutes

    //Generer un code de 6 chiffre

    // 4. Créer le nouvel utilisateur

    const newUser = new User({

        nom: req.body.nom,

        email: req.body.email,

        password: hashedPassword,

        role: req.body.role || 'employe',

        statut: 'en\_attente',

        validationCode, // Ajouter le code

        validationCodeExpires, // Ajouter la date d'expiration

        emailVerified: false

    });

    const savedUser = await newUser.save();

   // 6. Envoyer l'email avec le code

    try {

        await sendRegistrationEmail(savedUser.email, savedUser.nom, validationCode);

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

    } catch (err) {

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

        return res.status(500).send({ message: "Erreur lors de l'envoi de l'email de confirmation." });

    }

    // 6. Préparer la réponse sans le mot de passe

    const userResponse = {

        \_id: savedUser.\_id,

        nom: savedUser.nom,

        email: savedUser.email,

        role: savedUser.role,

        statut: savedUser.statut,

        emailVerified: savedUser.emailVerified,

        createdAt: savedUser.createdAt

    };

    res.status(201).json({

        message: "Compte créé avec succès. En attente d'activation par le RH.",

        user: userResponse,

    });

});

//post validation du code de comfirmation

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

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

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

    const { email, code } = req.body;

    // 2. Trouver l'utilisateur

    const user = await User.findOne({ email });

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

    // 3. Vérifier le code et l'expiration

    if (user.validationCode !== code) {

        return res.status(400).send({ message: "Code de validation incorrect." });

    }

    if (user.validationCodeExpires < Date.now()) {

        return res.status(400).send({ message: "Code de validation expiré." });

    }

    // 4. Mettre à jour emailVerified, laisser statut en\_attente

    user.emailVerified=true;

    user.validationCode=undefined;

    user.validationCodeExpires = undefined;

    await user.save();

    //envoyer le mail de validation reussite

    try{

        await sendValidationSuccessEmail(user.email, user.nom);

    }catch(error){

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

        return res.status(500).send({ message: "Email vérifié, mais erreur lors de l'envoi de l'email de confirmation." });

    }

    // 5. Réponse

    res.status(200).send({ message: "Email vérifié avec succès. Votre compte est en attente d'approbation par un responsable RH." });

})

module.exports = { createUser ,validateUserCode};

routes/registrationRoute.js

const express = require("express");

const router = express.Router();

const {createUser,validateUserCode} = require("../controllers/registerController")

// Route pour l'enregistrement

router.post("/Register",createUser);

// Route pour valider le code

router.post("/validate-code", validateUserCode);

module.exports = router;

module.exports = router ;