**Bước 0:** Cài đặt express

<https://expressjs.com/en/starter/generator.html>

Mở cmd/terminal của laptop

**npm install express –save => lưu lại**

**npm install -g express-generator => cài**

**express =>check->N**

**Bước 1:** Tạo project

**express --view=hbs tenproject**

**cd tenproject**

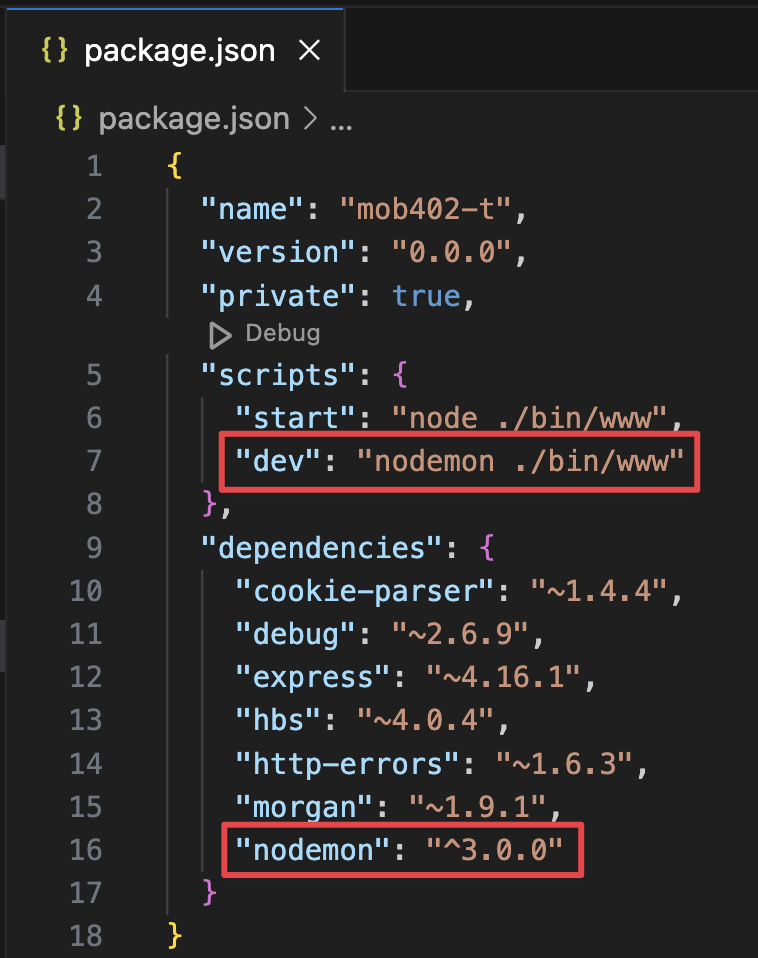
**npm install**

**[nếu có lỗi: npm audit fix --force]**

**Bước 2:** Cài thư viện nodemon

**npm i nodemon**

**Bước 3:** Setup **nodemon** trong **package.json**



Chạy project: **npm run dev**

**Một số lưu ý:**

Cấu hình **Route** trong **app.js**

A screen shot of a computer program

Description automatically generated

Cấu hình **port** của server trong **bin/www**

**A screen shot of a computer

Description automatically generated**

**View**

* **xuất ra giao diện views:**

router.**get**('/', function(req, res, next) {

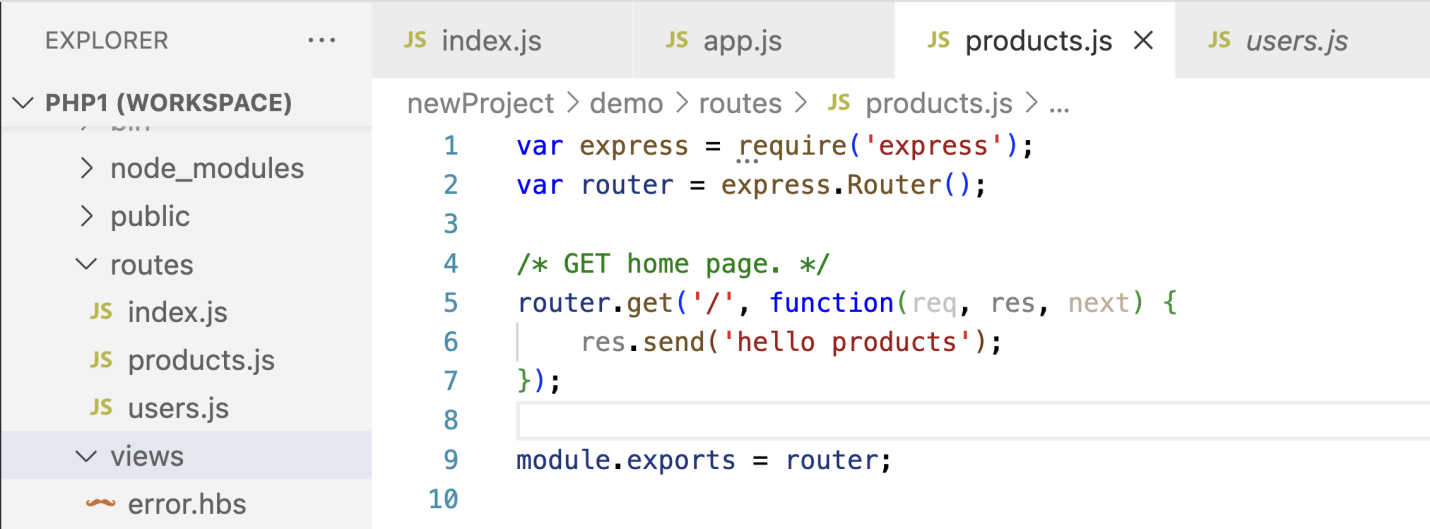
**res.render**('index', { title: 'Xin chào mọi người' });

});

**A screenshot of a computer

Description automatically generated**

**- xuất ra không cần views tạo file products.js:**

****

* **API**

router.**get**('/san-pham', function(req, res, next) {

**res.json**({ "name": 'Bánh Oreo', "price": 1000 });

});

**cài đặt mongo compas**

[**https://www.mongodb.com/try/download/community-kubernetes-operator**](https://www.mongodb.com/try/download/community-kubernetes-operator)

**Connect databse (Mongodb)**

**Bước 1:** Cài đặt thư viện

**npm i mongoose**

**Bước 2**: Tạo model

**Lưu ý: Db cần “s” model thì “hông”**

* **Category.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const ObjectId = Schema.ObjectId;

const **category** = new Schema({

id: { type: ObjectId }, // khóa chính

name: {

type: String, // kiểu dữ liệu

required: true, // bắt buộc phải có

unique: true, // không được trùng

trim: true, // bỏ khoảng trắng 2 đầu

minlength: 3, // độ dài tối thiểu

maxlength: 50, // độ dài tối đa

default: 'No name' // giá trị mặc định

},

});

module.exports = mongoose.models.**category** || mongoose.model(**'category'**, **category**);

**// category -----> categories**

* **Product.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const ObjectId = Schema.ObjectId;

const **product** = new Schema({

id: { type: ObjectId }, //khóa chính

name: { type: String },

price: { type: Number },

quantity: { type: Number },

image: { type: String },

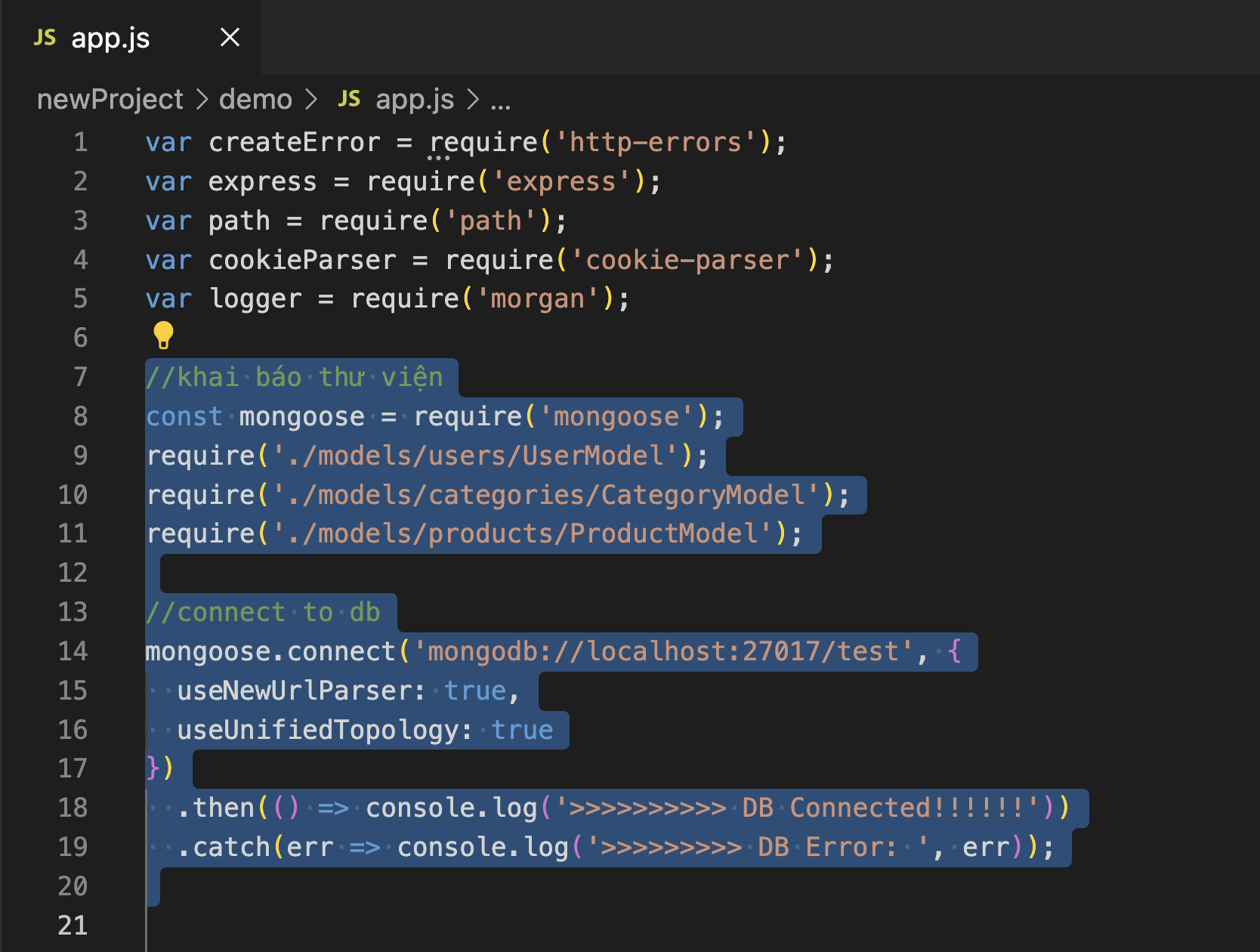
category: { type: ObjectId, ref: **'category'** } **//khóa ngoại**

});

module.exports = mongoose.models.**product** || mongoose.model(**'product**', **product**);

**// product -----> products**

**Bước 3**: Config mongoose trong **app.js**



//connect database

mongoose.connect('mongodb://**127.0.0.1:27017/test**', {

useNewUrlParser: true,

useUnifiedTopology: true

})

.then(() => console.log('>>>>>>>>>> DB Connected!!!!!!'))

.catch(err => console.log('>>>>>>>>> DB Error: ', err));

**API đơn giản:**

* Lấy danh sách sản phẩm

var modelProduct = require('../models/Product');

//http://localhost:3000/products/list-product

router.get('/list-product', **async** function(req, res, next) {

var data = **await** modelProduct**.find()**;

res.json(data);

});

* Chi tiết sản phẩm theo id

//http://localhost:3000/products/detail-by-id?id=64aa82dac3db275e94364cc0

router.get('/detail-by-id', **async** function(req, res, next) {

**var id = req.query.id;**

var data = **await** modelProduct.**find({\_id: id})**;

res.json(data);

});

//http://localhost:3000/products/detail-find-by-id?id=64aa82dac3db275e94364cc0

router.get('/detail-find-by-id', async function(req, res, next) {

**var id = req.query.id;**

var data = await modelProduct.**findById(id)**;

res.json(data);

});

* Lấy 1 số thông tin của một bảng

Ví dụ: Lấy tên, giá của sản phẩm có trong danh sách

//http://localhost:3000/products/list-product-name-and-price

router.get('/list-product-name-and-price', async function(req, res, next) {

var data = await modelProduct.**find({},'name price')**;

res.json(data);

});

* Lấy thông tin khóa ngoại

//http://localhost:3000/products/list-product-with-category

router.get('/list-product-with-category', async function(req, res, next) {

var data = await modelProduct.**find().populate('category')**;

res.json(data);

});

* Truy vấn có lồng điều kiện

**Ví dụ 1:**

/\*\*

\*

\* select \* from products where name like '%query%'

\* and price > 1000 and price < 2000

\* or quantity < 100

\*/

var query = {

// **gt**: lớn hơn (greater than), **lt**: nhỏ hơn (less than)

// **Ví dụ:** and price > 1000 AND price < 2000

price: { $gt: 1000, $lt: 2000 },

// **gte**: lớn hơn hoặc bằng (greater than or equal to), **lte**: nhỏ hơn hoặc bằng (less than or equal to)

// **Ví dụ:** and quantity <= 100

// quantity: { $lte: 100 },

// **Ví dụ:** or quantity <= 100 or quantity > 200

$or: [ { quantity: { $lte: 100 } }, { price: { $gt: 200 } } ],

// **Ví dụ:**

// $regex: regular expression

// $options: i: ignore case

// tìm kiếm theo tên sản phẩm có chứa keyword

// name: { $regex: keyword, $options: 'i' }

// **Ví dụ:** tìm kiếm theo tên

name: keyword,

}

var product = await modelProduct.find(query);

**Ví dụ 2:**

var products = await modelProduct

.find({},'name price category') // lấy 2 field name và price

.populate('category') // lấy thông tin category

.sort({ price: -1 }) // sắp xếp giảm dần theo giá;

.skip(0) // bỏ qua 2 sản phẩm

.limit(2) // lấy 2 sản phẩm

* Một số toán tử truy vấn so sánh trong MongoDB

**$eq** trả về các document trong đó giá trị bằng với một giá trị được chỉ định

cú pháp : { <field>: { $eq: <value> } }

**$gt** trả về các document trong đó giá trị của trường một giá trị được chỉ định

cú pháp : {field: {$gt: value} }

**$gte** trả về các document trong đó giá trị của trường lớn hơn hoặc bẳng 1 giá trị được chỉ định

cú pháp : {field: {$gte: value} }

**$in** trả về các document trong đó giá trị nằm trong mảng được chỉ định

cú pháp : { field: { $in: [<value1>, <value2>, ... <valueN> ] } }

**$lt** so sánh các giá trị nhỏ hơn giá trị được chỉ định

cú pháp : {field: {$lt: value} }

**$lte** so sánh các giá trị nhỏ hơn một giá trị được chỉ định

cú pháp : { field: { $lte: value} }

**$ne** so sánh các giá trị không bằng giá trị được chỉ định

cú pháp : {field: {$ne: value} }

**$nin** lấy ra các giá trị không có trong mảng được chỉ định cú pháp : { field: { $nin: [ <value1>, <value2> ... <valueN> ]} }

* Truy vấn nhiều điều kiện trong MongoDB

MongoBD có hỗ trợ chúng ta truy vẫn nhiều diều kiện trong 1 lần khai báo với các toán tử AND, OR như trong SQL.

**$and**

Thực hiện một logic AND hoạt động trên một mảng của một hoặc nhiều biểu thức và chọn ra các tài liệu đáp ứng tấy cả các biểu thức trong mảng. cú pháp :

{ $and: [ { <expression1> }, { <expression2> } , ... , { <expressionN> } ] }

Ví dụ: bạn muốn lấy ra tất cả user có giới tính là nam. Bạn có thể sử dụng câu lệnh sau:

db.users.find({$and: [{gender: 'Male'}, {email: 'alex@gmail.com'}]});

**$or**

Ví dụ: bạn muốn lấy ra user là nữ hoặc trên 18 tuổi.

db.user.find({$or: [{gender: 'Female'}, {age: {$gt: 18}}]});

**$not**

Ví dụ: bạn muốn lấy ra các user lớn hơn 15 tuổi. Nghĩa là lấy các user nhỏ hơn hoặc bằng 15 tuổi.

db.users.find({age: {$not: {$gt: '15'}}});

**$nor**

Ví dụ: bạn muốn lấy ra các user không phải nữ và không lớn hơn 15 tuổi.

db.user.find({$nor: [{gender: 'Female'}, {age: {$gt: '20'}}]});

* **Regular Expressions**

Regular Expressions là một chức năng nâng cao trong truy vấn dữ liệu. Ví dụ chúng ta muốn tìm những user mà có tên bắt đầu bởi chữ 'A' regular expression sẽ giúp chúng ta làm được điều này.

db.users.find({'name': /(A.\*)/i});

'name': /(A.\*)/i

//i chỉ định thực hiện regex case-insensitive

(A.\*) chỉ định name phải bắt đầu bằng chữ A. Tương tự như query like 'A%' trong SQL.

Ngoài ra, có thể sử dụng $regex để tìm như sau:

db.user.find({name: {$regex: 'A', $options: '$i'}});

Nếu không sử dụng regex case-insensitive. db.user.find({name: {$regex: 'A'}});

* Thêm

router.post("/add", async function (req, res, next) {

try {

const { name, price, quantity, image, category } = req.body;

const newProduct = {

name,

price,

quantity,

image,

category,

};

await modelProduct.create(newProduct);

res.json({ status: 1, message: "Thêm sản phẩm thành công" });

} catch (err) {

res.json({ status: 0, message: "Thêm sản phẩm thất bại" });

}

});

* Sửa

router.post("/edit", async function (req, res, next) {

try {

const { id, name, price, quantity, image, category } = req.body;

var item = await modelProduct.findById(id);

if (item) {

item.name = name ? name : item.name;

item.price = price ? price : item.price;

item.quantity = quantity ? quantity : item.quantity;

item.image = image ? image : item.image;

item.category = category ? category : item.category;

await item.save();

res.json({ status: 1, message: "Sửa sản phẩm thành công" });

}

} catch (err) {

res.json({ status: 0, message: "Sửa sản phẩm thất bại" });

}

});

* Xóa

router.get("/delete", async function (req, res, next) {

try {

var id = req.query.id;

await modelProduct.findByIdAndDelete(id);

res.json({ status: 1, message: "Xóa sản phẩm thành công" });

} catch (err) {

res.json({ status: 0, message: "Xóa sản phẩm thất bại", err: err });

}

});

router.delete('/:id', async function (req, res, next) {

try {

let { id } = req.params;

await ProductController.remove(id);

res.json({ status: true });

} catch (error) {

res.json({ status: false });

}

});

**Upload file**

**Bước 1:** Cài đặt thư viện Multer

**npm i multer**

**Bước 2:** Tạo file **Upload.js** để setup đường dẫn lưu file và cách đặt tên cho file khi được upload

const multer = require('multer');

const storage = multer.diskStorage({

destination: function (req, file, cb) {

cb(null, './public/images/')

},

filename: function (req, file, cb) {

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

cb(null, file.fieldname + '-' + uniqueSuffix + '-' + file.originalname);

}

});

module.exports = multer({ storage: storage });

**Bước 3:**

**Upload single file**

router.post('/upload', [upload.single('image')],

async (req, res, next) => {

try {

const { file } = req;

if (!file) {

return res.json({ status: 0, link : "" });

} else {

const url = `http://192.168.1.13:3000/images/${file.filename}`;

return res.json({ status: 1, url : url });

}

} catch (error) {

console.log('Upload image error: ', error);

return res.json({status: 0, link : "" });

}

});

**Upload multiple file**

router.post('/uploads', [upload.array('image', 9)],

async (req, res, next) => {

try {

const { files } = req;

if (!files) {

return res.json({ status: 0, link : [] });

} else {

const url = [];

for (const singleFile of files) {

url.push(`http://192.168.1.13:3000/images/${singleFile.filename}`);

}

return res.json({ status: 1, url : url });

}

} catch (error) {

console.log('Upload image error: ', error);

return res.json({status: 0, link : [] });

}

});

**Gửi mail**

**Bước 1:** Cài đặt thư viện Nodemailer

**npm i nodemailer**

**Bước 2:** Cấu hình mail

const nodemailer = require("nodemailer");

const transporter = nodemailer.createTransport({

host: "smtp.gmail.com",

port: 465,

secure: true,

auth: {

user: **'email\_cua\_ban**',

pass: '**mat\_khau\_ung\_dung**'

}

});

module.exports = { transporter };

**Bước 3:** Gửi mail

router.post("/send-mail", async function(req, res, next){

try{

const {to, subject, content} = req.body;

const mailOptions = {

from: "**abc <email address>",**

to: to,

subject: subject,

html: content

};

await sendMail.transporter.sendMail(mailOptions);

res.json({ status: 1, message: "Gửi mail thành công"});

}catch(err){

res.json({ status: 0, message: "Gửi mail thất bại"});

}

});

**Xác thực bằng JWT**

**Bước 1:** Cài đặt thư viện Nodemailer

**npm i** jsonwebtoken

**npm i bcrypt --save**

**Bước 2:** tạo file middleware/authen.js

const jwt = require('jsonwebtoken');

const authen = async (req, res, next) => {

try {

if (req.headers.authorization) {

const token = req.headers.authorization.split(' ')[1];

const data = jwt.verify(token, 'shhhhh');

req.user = data.user;

next();

}else{

res.status(401).json({ error: 'Not authoried!!' });

}

} catch (error) {

res.status(401).json({ error: error.message });

}

}

module.exports = authen;

**Bước 3:** register:

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const authen = require('../middleware/authen');

router.post('/api/register', async function (req, res, next) {

try {

const { name, email, password, age, confirm\_password } = req.body;

if(password !== confirm\_password) {

throw new Error('Dữ liệu không chính xác');

}

const salt = bcrypt.genSaltSync(10);

const hash = bcrypt.hashSync(password, salt);

const user = new UserModel({ name, email, password: hash, age });

await user.save();

res.status(200).json({ user });

} catch (error) {

console.log(error);

res.status(414).json({ user: { name: nulll, email: null, } });

}

});

**Bước 4:** login

router.post('/api/login', async function (req, res, next) {

try {

const { email, password } = req.body;

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

// kiểm tra password đã mã hóa

if (user && bcrypt.compareSync(password, user.password)) {

const access\_token = jwt.sign({ user }, 'shhhhh', { expiresIn: 1 \* 60 });

const refresh\_token = jwt.sign({ user }, 'shhhhh', { expiresIn: 90 \* 24 \* 60 \* 60 });

// access token là chuỗi ngẫu nhiên, dùng để xác thực người dùng

// refresh token là chuỗi ngẫu nhiên, dùng để lấy lại access token

res.status(200).json({ user, access\_token, refresh\_token });

}

else {

res.status(401).json({ error: 'Sai email hoặc mật khẩu' });

}

} catch (error) {

res.status(401).json({ error: error.message });

}

});

}

router.post('/api/refresh-token', async function (req, res, next) {

try {

let { refresh\_token } = req.body;

const data = jwt.verify(refresh\_token, 'shhhhh');

const access\_token = jwt.sign({ user: data.user }, 'shhhhh', { expiresIn: 1 \* 60 });

refresh\_token = jwt.sign({ user: data.user }, 'shhhhh', { expiresIn: 90 \* 24 \* 60 \* 60 });

res.status(200).json({ user: data.user, access\_token, refresh\_token });

} catch (error) {

res.status(414).json({ error: error.message });

}

});