

LAB 4

MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ API Delete
- ✓ Multer (upload ảnh).
- ✓ Gửi mail
- ✓ Authentication - JWT (token) - create, refresh token, exp

NỘI DUNG

BÀI 1: THỰC HIỆN CHỨC NĂNG XÓA MỘT FRUIT

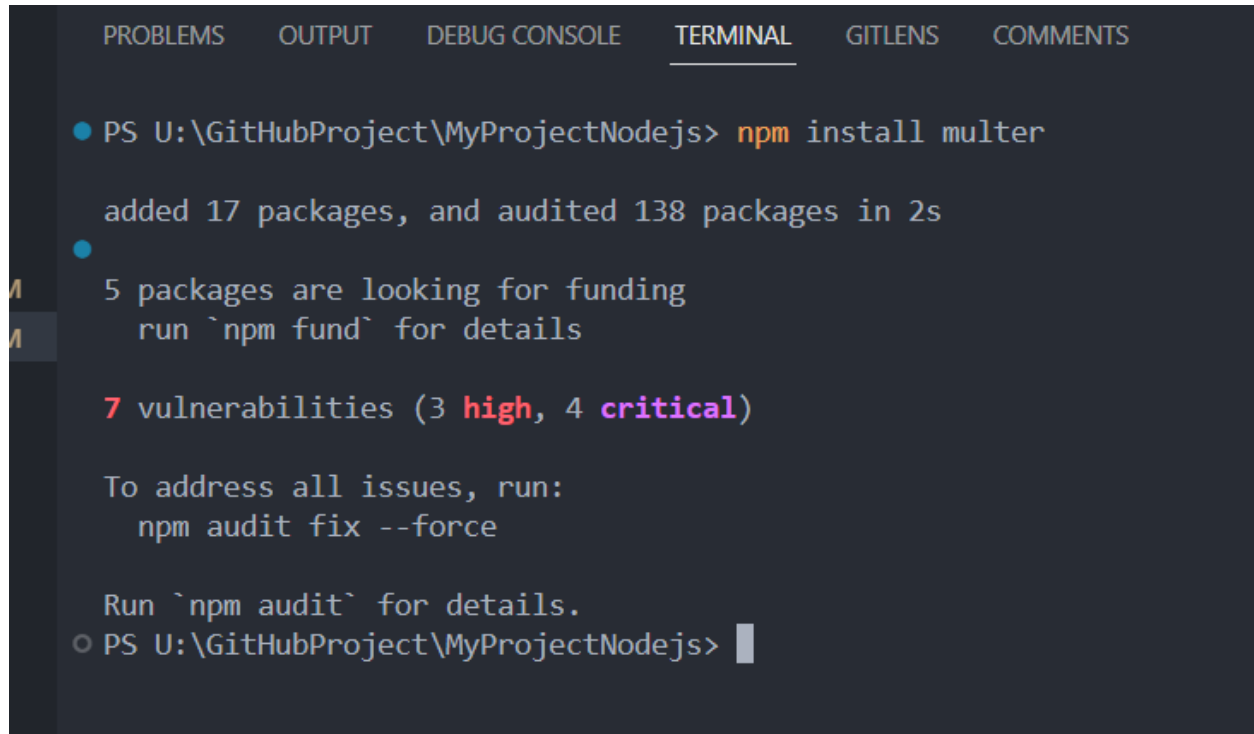
Tại file **api.js**

```
router.delete('/destroy-fruit-by-id/:id', async (req, res) => {
  try {
    const {id} = req.params
    const result = await Fruits.findByIdAndDelete(id);
    if(result)
    {
      //Nếu xóa thành công sẽ trả về thông item đã xóa
      res.json({
        "status" : 200,
        "messenger" : "Xóa thành công",
        "data" : result
      })
    }else
    {
      res.json({
        "status" : 400 ,
        "messenger" : "Lỗi, Xóa không thành công",
        "data" : []
      })
    }
  } catch (error) {
    console.log(error);
  }
})
```

BÀI 2: THÊM THÔNG TIN MỘT FRUIT VÀ UPLOAD HÌNH ẢNH CHO FRUIT ĐÓ THÔNG QUA MULTER

Bước 1: Thêm thư viện

`npm install multer`



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS  COMMENTS

● PS U:\GitHubProject\MyProjectNodejs> npm install multer

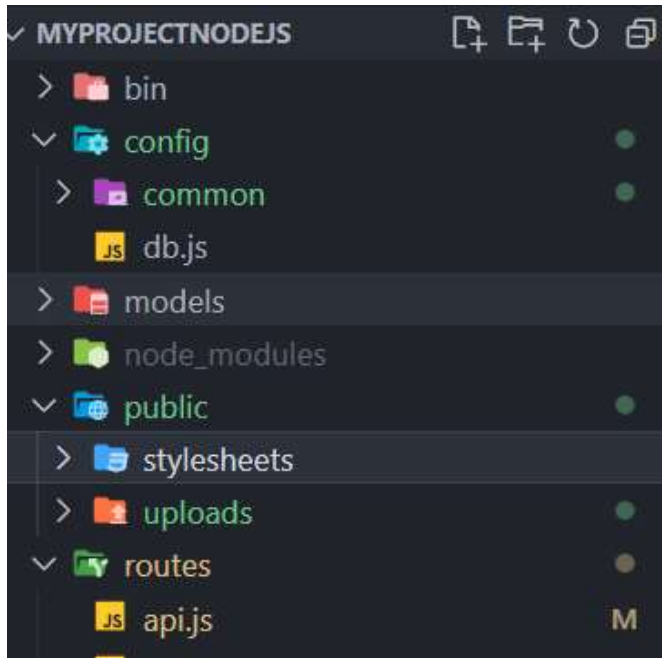
added 17 packages, and audited 138 packages in 2s
●
5 packages are looking for funding
  run `npm fund` for details

7 vulnerabilities (3 high, 4 critical)

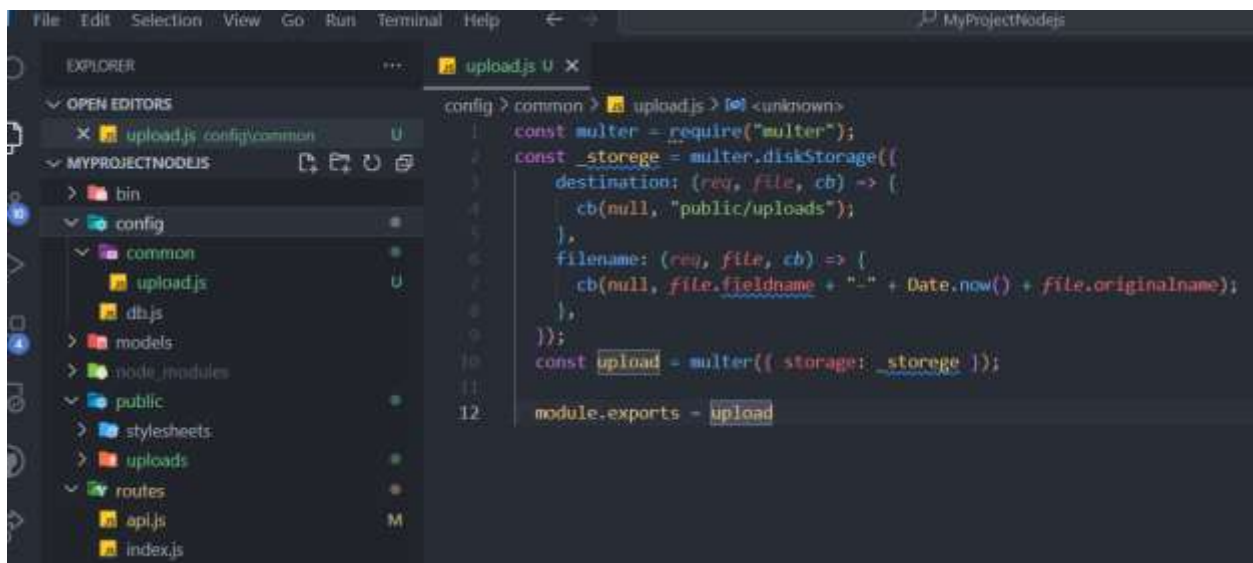
To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
○ PS U:\GitHubProject\MyProjectNodejs> 
```

Bước 2: Tạo folder đặt tên **uploads** trong folder **public** (các file tải lên sẽ nằm ở thư mục này (**public/uploads**))



Bước 3: Cấu hình upload. Tạo 1 file **upload.js** (config/common/upload.js)



Bước 4: Viết api, mở file api.js

```
const Upload = require('../config/common/upload');
router.post('/add-fruit-with-file-image', Upload.array('image', 5), async (req, res) => {
    //Upload.array('image', 5) => up nhiều file tối đa là 5
    //upload.single('image') => up load 1 file
    try {
        const data = req.body; // Lấy dữ liệu từ body
        const {files} = req //files nếu upload nhiều, file nếu upload 1 file
        const urlsImage =
        files.map((file) => `${req.protocol}://${req.get("host")}/uploads/${file.filename}`)
        //url hình ảnh sẽ được lưu dưới dạng: http://localhost:3000/upload/filename
        const newfruit = new Fruits({
            name: data.name,
            quantity : data.quantity,
            price : data.price,
            status : data.status,
            image : urlsImage, /* Thêm url hình */
            description : data.description,
            id_distributor : data.id_distributor
        }); //Tạo một đối tượng mới
        const result = await newfruit.save(); //Thêm vào database
        if(result)
            // Nếu thêm thành công result !null trả về dữ liệu
            res.json({
                "status" : 200,
                "messenger" : "Thêm thành công",
                "data" : result
            })
        }else
            // Nếu thêm không thành công result null, thông báo không thành công
            res.json({
                "status" : 400 ,
                "messenger" : "Lỗi, thêm không thành công",
                "data" : []
            })
        }
    } catch (error) {
        console.log(error);
    }
});
```

The screenshot shows a REST client interface with a POST request to `http://localhost:3000/api/add-distributor`. The request body is a JSON object with the following fields:

| Key | Value | Description |
|--|--------------------------|-------------|
| <input checked="" type="checkbox"/> name | Tan Mỹ | |
| <input checked="" type="checkbox"/> quantity | 100 | |
| <input checked="" type="checkbox"/> price | 5000 | |
| <input checked="" type="checkbox"/> status | 1 | |
| <input checked="" type="checkbox"/> image | 3 files selected | |
| <input checked="" type="checkbox"/> description | ngon | |
| <input checked="" type="checkbox"/> id_distributor | 640c7414be38e1f0171131ee | |

The response body is a JSON object:

```

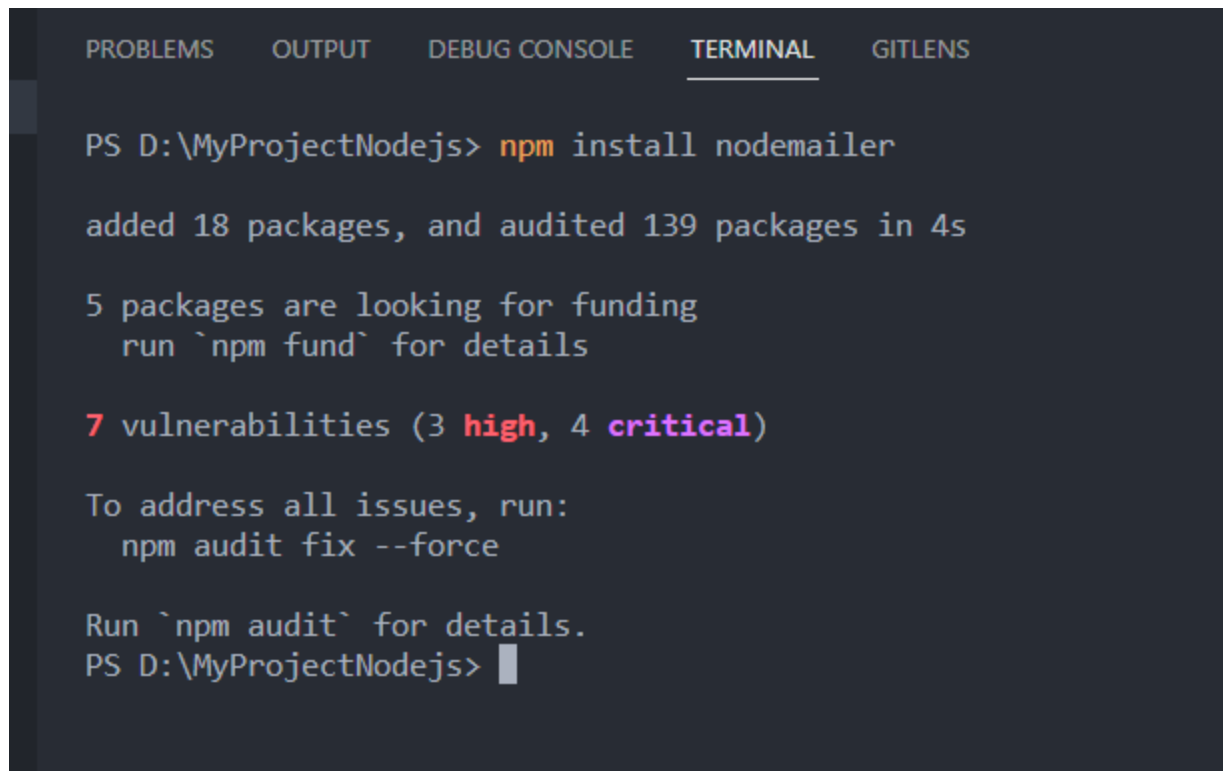
1 {
2   "status": 200,
3   "message": "Thêm thành công",
4   "data": {
5     "name": "Tan Mỹ",
6     "quantity": 100,
7     "price": 5000,
8     "status": 1,
9     "image": [
10      "http://localhost:3000/uploads/image-169217414238716188858642148_tan-bi-my-bad-dalicious-kizw-36-44.jpg",
11      "http://localhost:3000/uploads/image-169217414238716188858642148_Tan-1.jpg",
12      "http://localhost:3000/uploads/image-169217414238716188858642148.jpg"
13    ],
14     "description": "ngon",
15     "id_distributor": "640c7414be38e1f0171131ee",
16     "_id": "640c7414be38e1f0171131ee"
17   }
18 }
  
```

The status bar at the bottom indicates a successful response with status 200 OK.

BÀI 3: GỬI EMAIL ĐĂNG KÝ TÀI KHOẢN THÀNH CÔNG

Bước 1: Thêm thư viện

`npm install nodemailer`



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS D:\MyProjectNodejs> npm install nodemailer

added 18 packages, and audited 139 packages in 4s

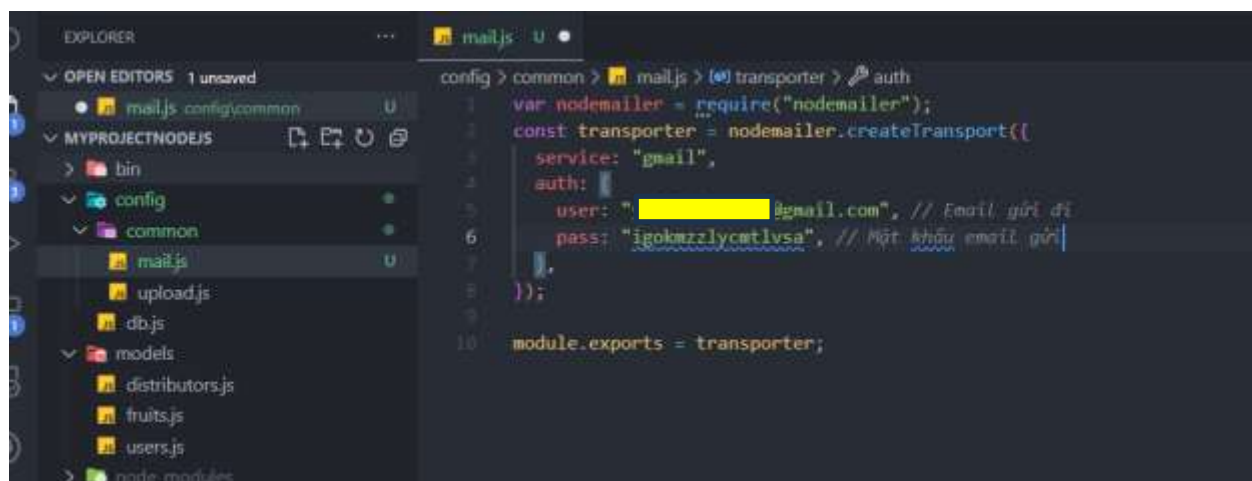
5 packages are looking for funding
  run `npm fund` for details

7 vulnerabilities (3 high, 4 critical)

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\MyProjectNodejs> 
```

Bước 2: Cấu hình gửi mail



```
EXPLORER  ...  mailjs: U

OPEN EDITORS  1 unsaved
  mailjs config:common  U

MYPROJECTNODEJS
  bin
  config
    common
      mailjs  U
      upload.js
      db.js
  models
    distributors.js
    fruits.js
    users.js
  node_modules

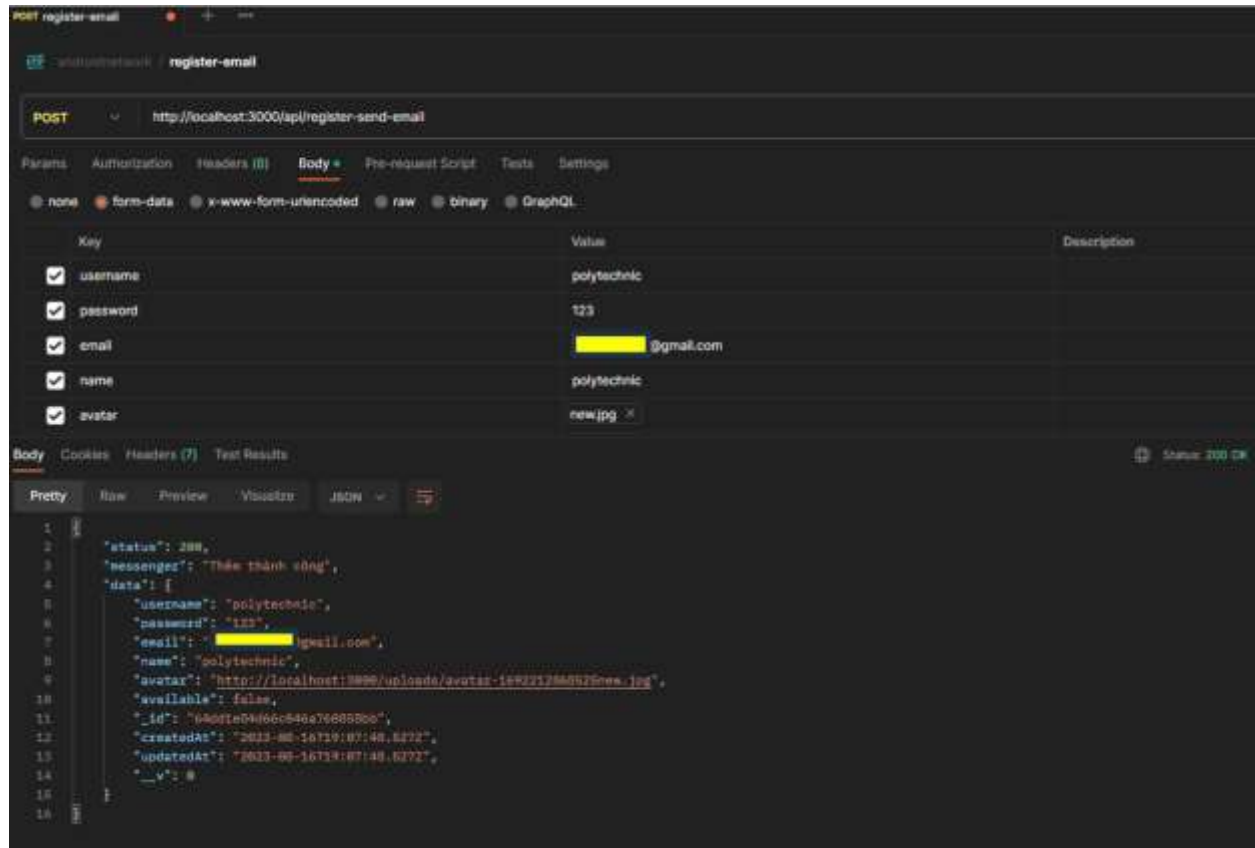
config > common > mailjs > transporter > auth
1  var nodemailer = require("nodemailer");
2  const transporter = nodemailer.createTransport({
3    service: "gmail",
4    auth: {
5      user: " [REDACTED]@gmail.com", // Email gửi đi
6      pass: "igokazzlycatlvs", // Mật khẩu email gửi
7    },
8  });
9
10 module.exports = transporter;
```

Bước 3: Viết api. Mở file api.js

```
const Users = require('../models/users');
const Transporter = require('../config/common/mail')
router.post('/register-send-email', Upload.single('avatar'), async (req, res) => {
  try {
    const data = req.body;
    const {file} = req
    const newUser = Users({
      username: data.username,
      password: data.password,
      email: data.email,
      name: data.name,
      avatar: `${req.protocol}://${req.get("host")}/uploads/${file.filename}`,
      //url avatar http://localhost:3000/uploads/filename
    })
    const result = await newUser.save()
    if(result)
    { //Gửi mail
      const mailOptions = {
        from: "dinhnt24@fpt.edu.vn", //email gửi đi
        to: result.email, // email nhận
        subject: "Đăng ký thành công", //subject
        text: "Cảm ơn bạn đã đăng ký", // nội dung mail
      };
      // Nếu thêm thành công result !null trả về dữ liệu
      await Transporter.sendMail(mailOptions); // gửi mail
      res.json({
        "status": 200,
        "messenger": "Thêm thành công",
        "data": result
      })
    }
  } else
  { // Nếu thêm không thành công result null, thông báo không thành công
    res.json({
      "status": 400,
      "messenger": "Lỗi, thêm không thành công",
      "data": []
    })
  }
} catch (error) {
  console.log(error);
}
```



```
}
})
```



Mail được gửi về địa chỉ email tương ứng



BÀI 4: JWT (Đăng nhập trả về token, refreshToken)

Bước 1: Thêm thư viện

`npm install jsonwebtoken`

Bước 2: Viết api Login, mở file `api.js`

```
const JWT = require('jsonwebtoken');
const SECRETKEY = "FPTPOLYTECHNIC"
router.post('/login', async (req, res) => {
  try {
    const {username, password} = req.body;
    const user = await Users.findOne({username, password})
    if(user)
    {
      //Token người dùng sẽ sử dụng gửi lên trên header mỗi lần muốn gọi api
      const token = JWT.sign({id: user._id}, SECRETKEY, {expiresIn: '1h'});
      //Khi token hết hạn, người dùng sẽ call 1 api khác để lấy token mới
      //Lúc này người dùng sẽ truyền refreshToken lên để nhận về 1 cặp token, refreshToken mới
      //Nếu cả 2 token đều hết hạn người dùng sẽ phải thoát app và đăng nhập lại
      const refreshToken = JWT.sign({id: user._id}, SECRETKEY, {expiresIn: '1d'})
      //expiresIn thời gian token
      res.json({
        "status" : 200,
        "messenger" : "Đăng nhập thành công",
        "data" : user,
        "token" : token,
        "refreshToken" : refreshToken
      })
    }
  } else {
    // Nếu thêm không thành công result null, thông báo không thành công
    res.json({
      "status" : 400 ,
      "messenger" : "Lỗi, đăng nhập không thành công",
      "data" : []
    })
  }
} catch (error) {
  console.log(error);
}
})
```



Bước 3 : Kiểm tra token mỗi lần gọi api/get-list-fruit (ở lab 3)

```

206   });
207
208   router.get('/get-list-fruit', async (req, res, next) => {
209     const authHeader = req.headers['authorization']
210     //Authorization thêm từ khóa `Bearer token`
211     //nên sẽ xử lý cắt chuỗi
212     const token = authHeader && authHeader.split(' ')[1]
213     //Nếu không có token sẽ trả về 401
214     if (token == null) return res.sendStatus(401)
215     let payload;
216     JWT.verify(token, SECRETKEY, (err, _payload) => {
217       //Kiểm tra token, nếu token ko đúng, hoặc hết hạn
218       //Trả status code 403
219       //Trả status hết hạn 401 khi token hết hạn
220       if(err instanceof JWT.TokenExpiredError) return res.sendStatus(401)
221       if (err) return res.sendStatus(403)
222       //Nếu đúng sẽ log ra dữ liệu
223       payload = _payload;
224     })
225     console.log(payload);
226     try {
227       const data = await Fruits.find().populate('id_distributor');
228       res.json({
229         "status" : 200,
230         "messenger" : "Danh sách fruit",
231         "data" : data
232       })
233     } catch (error) {
234       console.log(error);
235     }
236   })
237

```

PROBLEMS 100 OUTPUT DEBUG CONSOLE **TERMINAL** GITLENS

```

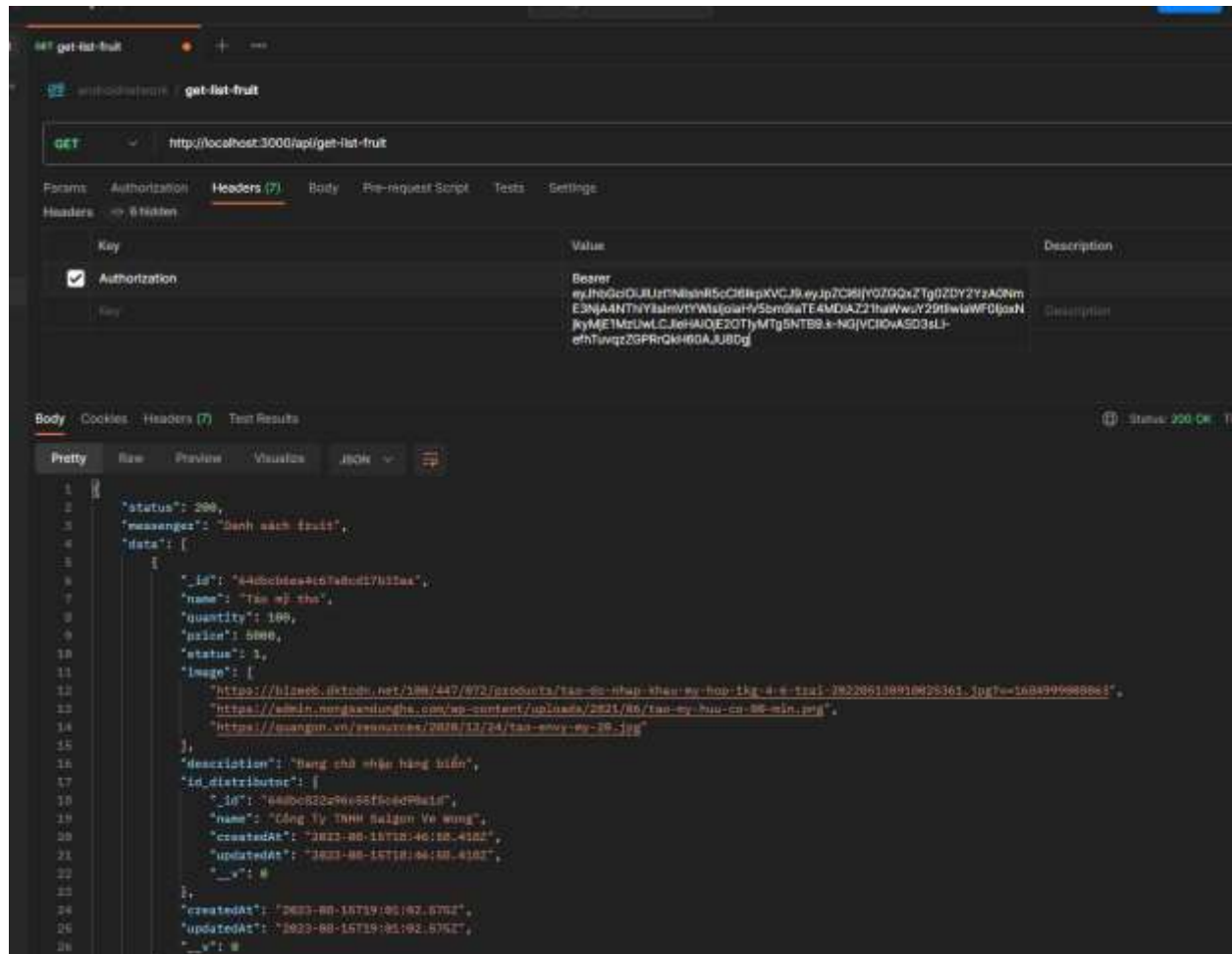
PS D:\MyProjectNodejs> npm run dev

> myprojectnodejs@0.0.0 dev
> nodemon ./bin/www

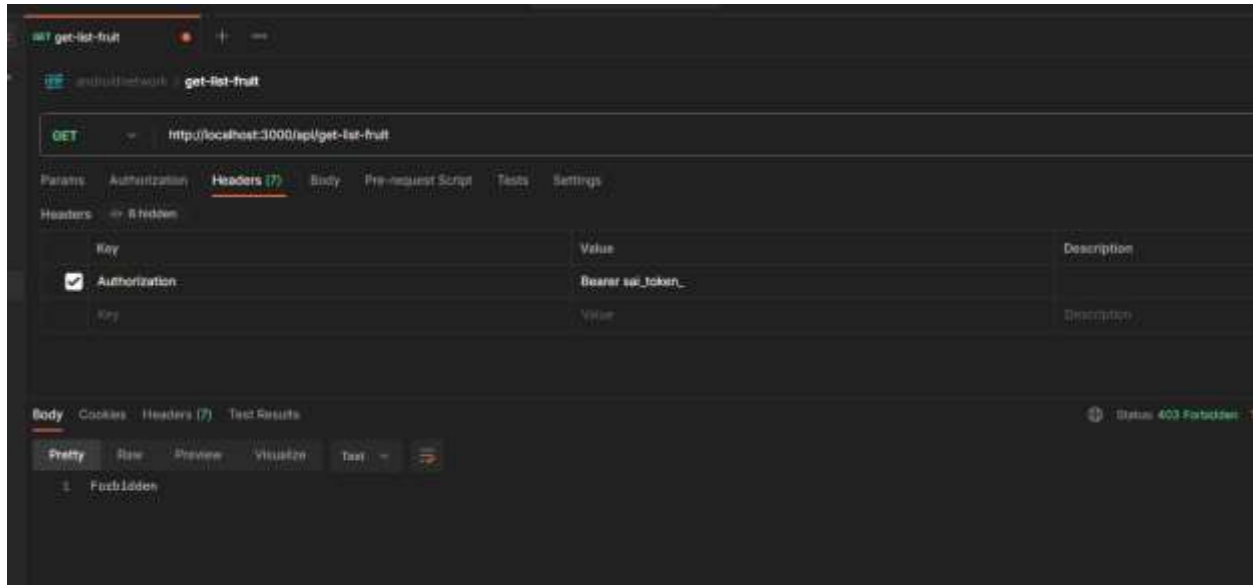
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./bin/www`
connect success
POST /api/login 200 21.873 ms - 735
{ id: '64dd1e84d66c046a760858bb', iat: 1692217904, exp: 1692221504 }
GET /api/get-list-fruit 200 16.111 ms - 2822

```

*Lưu ý : token ở Authorization = "Bearer mytoken"



*Nếu sai token ta sẽ nhận được kết quả như hình bên dưới



*** YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

--- Hết ---