Tổng kết báo cáo lần II Project 3

Họ tên: Phạm Minh Trường MSSV: 20215292

Nội dung

I. WEB API

1. Hash API

Chức năng chính: Tạo mã băm từ văn bản

Mô tả chi tiết

- Input:
  - Văn bản
  - Thuật toán mã hóa
- Output:
  - Mã băm
- Endpoint:
  - Endpoint: hash
  - Method : POST
- Process:

  Ta dùng thư viện class-validator để có thể kiểm tra dữ liệu đầu vào dễ hơn qua các Decorator : npm install class-validator

```
// định nghĩa dữ liệu đầu vào

import { IsIn, IsString } from 'class-validator';
export class HashTextDto {
  @IsString()
  text: string; // đoạn text cần băm

  @IsString()
  @IsIn(['md5', 'sha256', 'sha512']) // các loại thuật toán băm sử dụng
  algorithm: string;
}
```

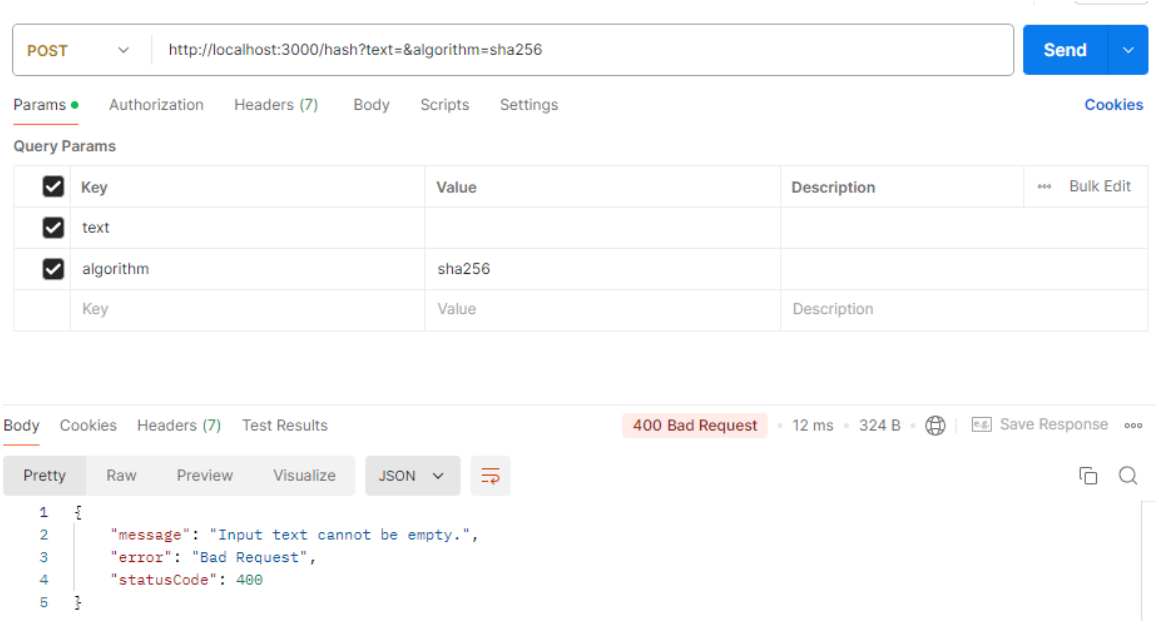
Ta dùng thư viện **crypto** để tạo chuỗi băm hệ hexa từ văn bản cần băm và loại băm.

ce365-api > src > hash > TS hash.service.ts > ...

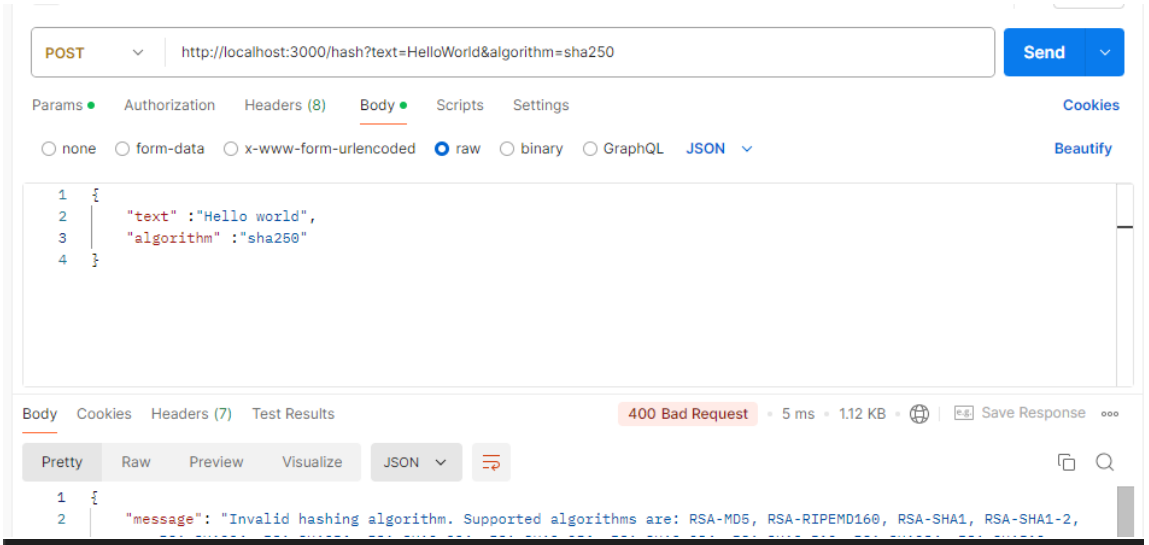
```ts
import { Injectable } from '@nestjs/common';
import * as crypto from 'crypto'; // thư viện crypto
@Injectable()
export class HashService {
  getHash(text: string, algorithm: string): string {
    // trả về mã hóa dưới dạng hexa
    return crypto.createHash(algorithm).update(text).digest('hex');
  }
}
```

- Error handle:
  - Chuỗi rỗng:

```ts
// Kiểm tra nếu chuỗi text bị thiếu hoặc rỗng
if (!text) {
  throw new BadRequestException('Input text cannot be empty.');
}
```

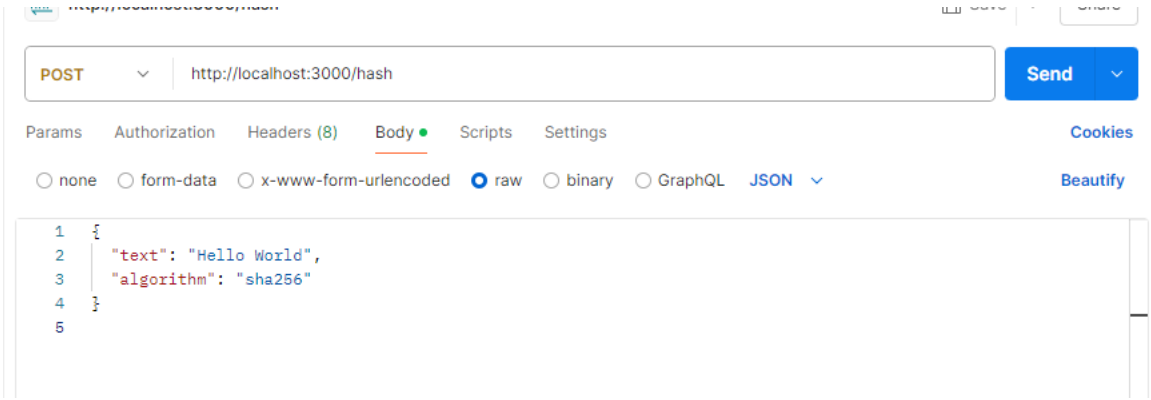


- Thuật toán không đúng hoặc không được hỗ trợ:

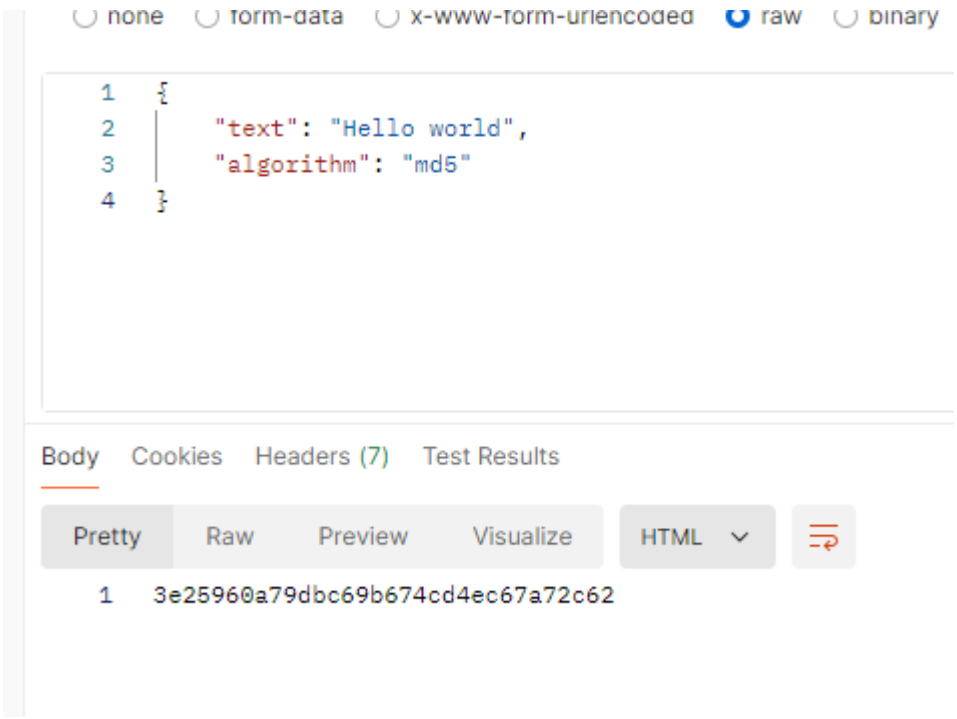


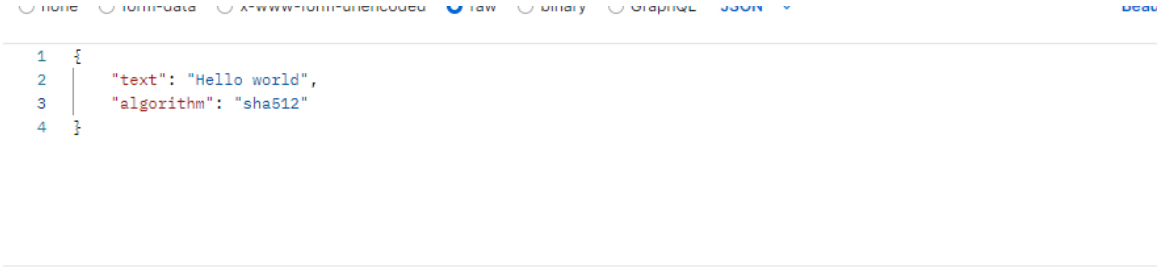

```
// Kiểm tra nếu thuật toán không hợp lệ
if (!crypto.getHashes().includes(algorithm)) {
  throw new BadRequestException(
    `Invalid hashing algorithm. Supported algorithms are: ${crypto
      .getHashes()
      .join(', ')}`,
  );
}
```

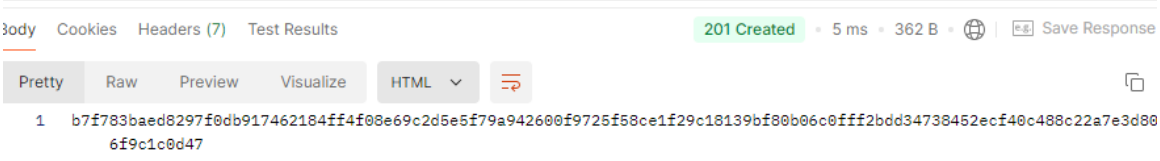
TestAPI:

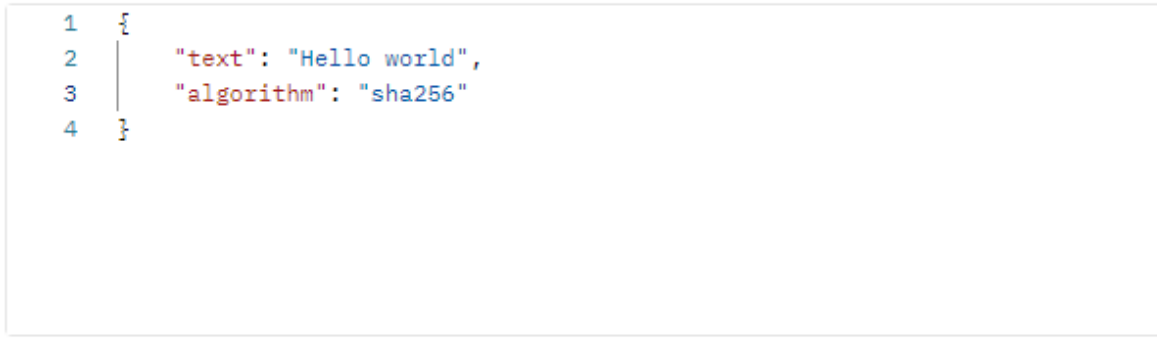


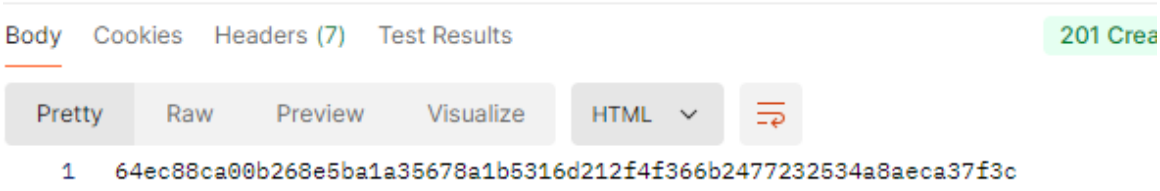
Gửi 1 HTTP Post đến module hash, truyền vào dữ liệu như trên và được kết quả











1. Qrcode API

a) Chức năng chính: Chuyển text thành ảnh Qrcode

b) Mô tả chi tiết:

- Input: Văn bản
- Output: Ảnh QR Code
- Process

Service:

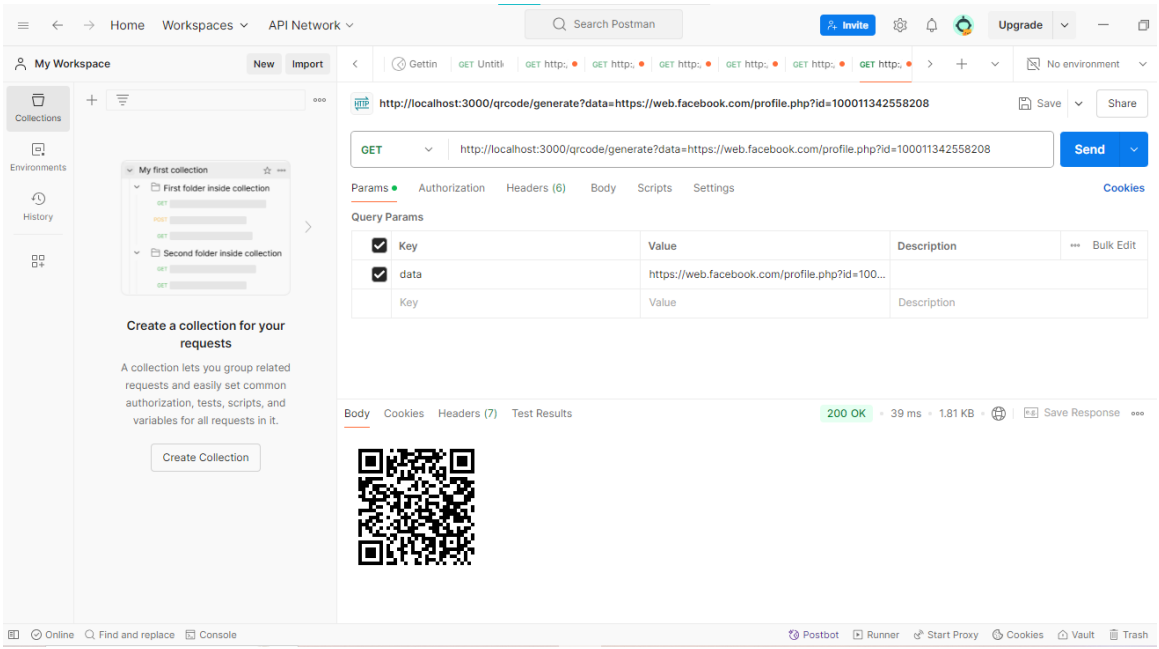
```
office365-api > src > qrcode > TS qrcode.service.ts > ...
1  import { Injectable } from '@nestjs/common';
2  import * as QRCode from 'qrcode';
3
4  @Injectable()
5  export class QrCodeService {
6    async generateQRCode(data: string): Promise<Buffer> {
7      const qrCodeImage = await QRCode.toBuffer(data); // Data to buffer: mã nhị phân của ảnh
8      return qrCodeImage;
9    }
10 }
11
```

Controller:

```
office365-api > src > qrcode > TS qrcode.controller.ts > ...
1  import { Controller, Query, Get, Res } from '@nestjs/common';
2  import { QrCodeService } from './qrcode.service';
3  import { Response } from 'express';
4
5  @Controller('qrcode')
6  export class QrCodeController {
7    constructor(private readonly qrCodeService: QrCodeService) {}
8
9    @Get('generate')
10   async generate(@Query('data') data: string, @Res() res: Response) { // lấy data từ query string trả về
11     const qrCode = await this.qrCodeService.generateQRCode(data);
12     // Đặt header cho nội dung hình ảnh là image/png
13     res.setHeader('Content-Type', 'image/png');
14     res.send(qrCode);
15   }
16 }
17
```

- Khi user gửi yêu cầu GET đến endpoint: /qrcode/generate gồm dữ liệu truy vấn (data)
- Controller nhận yêu cầu lấy dữ liệu data và gọi đến Service;
- Service tạo mã QR và trả về 1 Buffer chứa ảnh và controller thiết lập tiêu đề là image/png

Test API: đoạn text sẽ là link đến Facebook



2. GetExchangeRate API

a. Chức năng chính: Lấy tỷ giá hối đoái hiện tại giữa 2 loại tiền tệ từ API của Vietcombank (https://portal.vietcombank.com.vn/Usercontrols/TVPortal.TyGia/pXML.aspx)

b. Mô tả chi tiết:

- Input: Cấu trúc DTO gồm fromCurrency: mã tiền tệ gốc và toCurrency là mã tiền tệ đích
- Trong chức năng này, mặc định date là ngày hiện tại

```
c > get-exchange-rate > dto > TS ger-dto.ts > ...
1   export class getExchangeRateDto {
2     fromCurrency: string;
3     toCurrency: string;
4     date?: string; // ngày lấy tỉ giá
5   }
6
```

- Output: Trả về tỷ giá chuyển đổi giữa fromCurrency và toCurrency theo kiểu number
- Endpoint:
  - Endpoint: get-exchange-rate/current-date
  - Phương thức HTTP: GET

- Process:
  - Chuyển dữ liệu trả về từ API của Vietcombank từ XML sang JSON nhờ thư viện “xml2js”

```
import { parseStringPromise } from 'xml2js';
```

  - Tìm tỷ giá cho loại tiền tệ so với VND qua trường Buy của ExrateList

```
const baseRate = rates.find(
  (rate) => rate.$.CurrencyCode === dto.fromCurrency.toUpperCas
);
const targetRate = rates.find(
  (rate) => rate.$.CurrencyCode === dto.toCurrency.toUpperCase(
);
```

```
const baseBuyRate = parseFloat(baseRate.$.Buy.replace(',', ''))
const targetBuyRate = parseFloat(targetRate.$.Buy.replace(',',
```

  - -> Kết quả trả về là baseBuyRate/targetBuyRate
- Error Handle:
  - Không tìm thấy tỷ giá:

```
const rates = data.ExrateList.Exrate;

if (!rates || rates.length === 0) {
  throw new Error('No exchange rates found in API response');
}
```

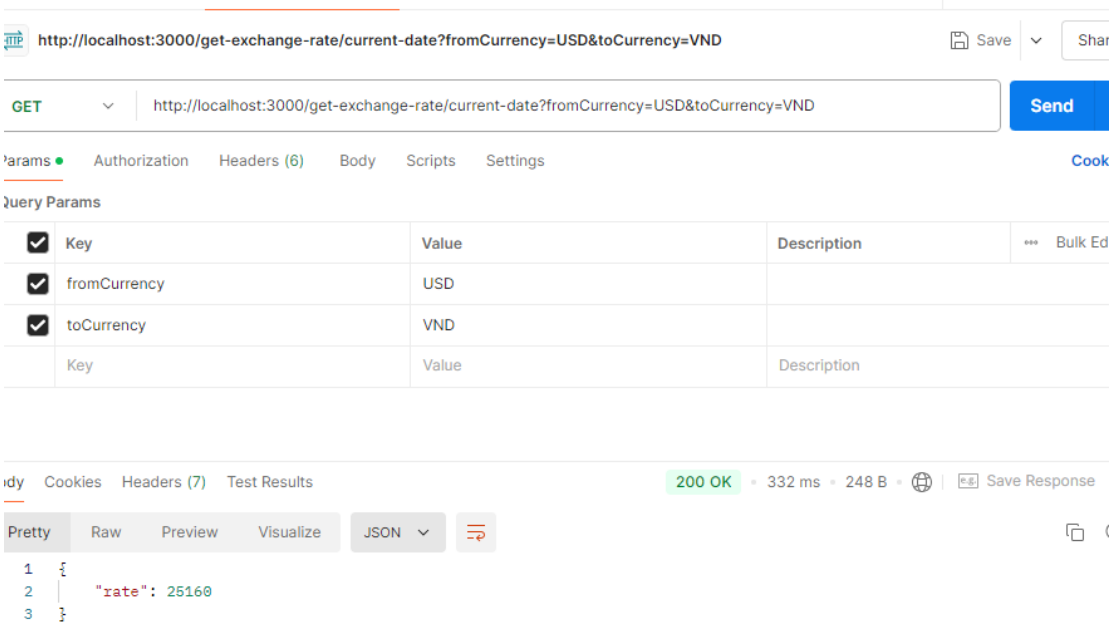
- Không tìm thấy loại tiền tệ

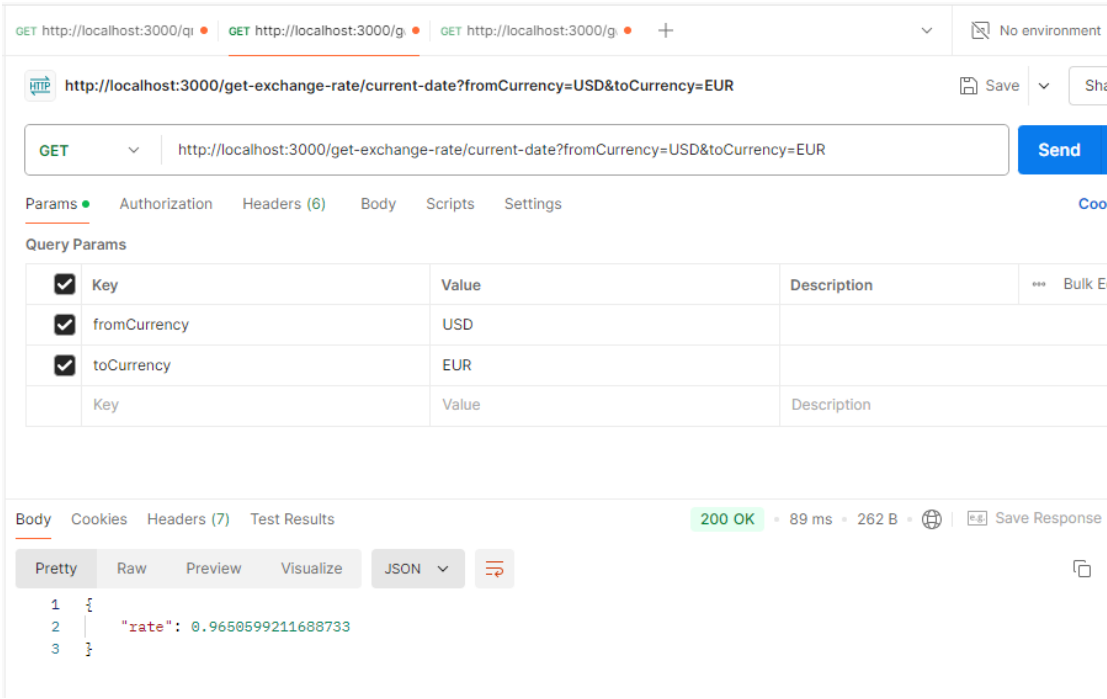
```
const baseRate = rates.find(
  (rate) => rate.$.CurrencyCode === dto.fromCurrency.toUpperCa
);
const targetRate = rates.find(
  (rate) => rate.$.CurrencyCode === dto.toCurrency.toUpperCase
);

if (!baseRate) {
  throw new Error(
    `Cannot find rate for base currency: ${dto.fromCurrency}`,
  );
}

if (!targetRate) {
  throw new Error(
    `Cannot find rate for target currency: ${dto.toCurrency}`,
  );
}
```

- Lỗi khi kết nối API :

```
} catch (error) {
  throw new Error(`Failed to fetch exchange rate: ${error.messag
}
```

- Ví dụ:





3. Historical Rates API

a) Chức năng chính: Lấy tỷ giá hối đoái lịch sử giữa 2 loại tiền tệ vào 1 ngày cụ thể từ link : https://www.vietcombank.com.vn/api/exchangerates?date=2024-10-08 ngày có thể tùy chỉnh theo format đó.

b) Mô tả chi tiết:

- Input: Cấu trúc DTO gồm fromCurrency: mã tiền tệ gốc và toCurrency là mã tiền tệ đích, date: ngày lấy tỷ giá

```
c > get-exchange-rate > dto > TS ger-dto.ts > ...
1   export class getExchangeRateDto {
2     fromCurrency: string;
3     toCurrency: string;
4     date?: string; // ngày lấy tỉ giá
5   }
6
```

- Output: Trả về tỷ giá chuyển đổi giữa fromCurrency và toCurrency theo kiểu number
- Process: tương tự với Process của lấy tỷ giá theo ngày hiện tại, thêm phần xử lý kiểm tra ngày đầu vào

- Endpoint:
  - Endpoint: get-exchange-rate/historical
  - Phương thức: GET
- Xử lý lỗi với Date:
  - Ngày không được cung cấp

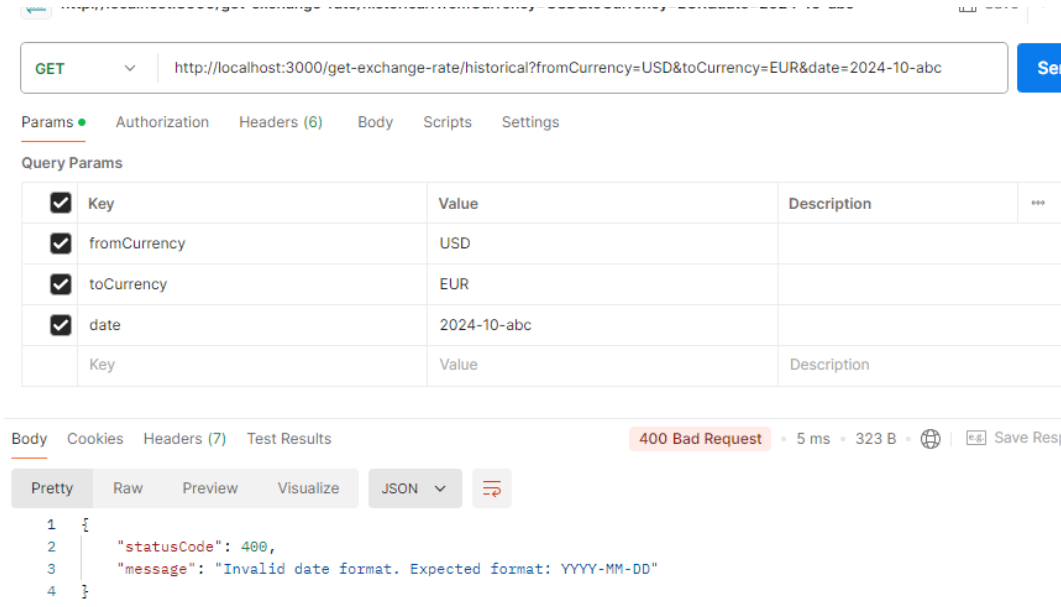
```
// Kiểm tra nếu ngày không được cung cấp
if (!date) {
  throw new HttpException(
    'Date is required to fetch historical exchange rates',
    HttpStatus.BAD_REQUEST,
  );
}
```

  - Ngày không đúng định dạng

```
// Kiểm tra ngày có hợp lệ hay không
if (!this.isValidDate(date)) {
  throw new HttpException(
    `Invalid date format. Expected format: YYYY-MM-DD`,
    HttpStatus.BAD_REQUEST,
  );
}
```

```
private isValidDate(date: string): boolean {
  // Kiểm tra xem chuỗi có phải định dạng ngày hợp lệ không
  const parsedDate = Date.parse(date);
  return !isNaN(parsedDate) && /^\d{4}-\d{2}-\d{2}$/.test(date);
}
```

Ví dụ :

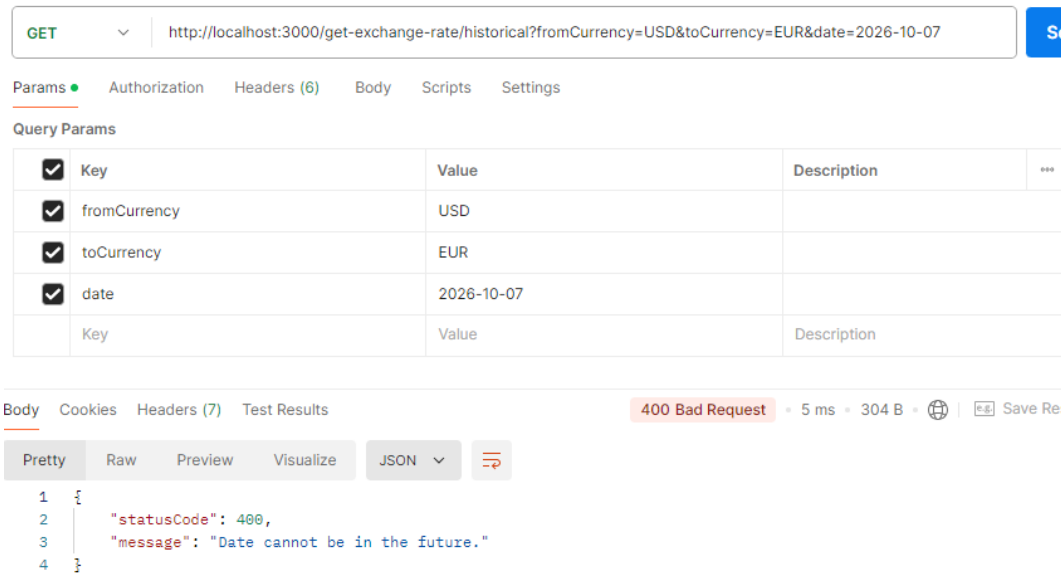


- Ngày trong tương lai

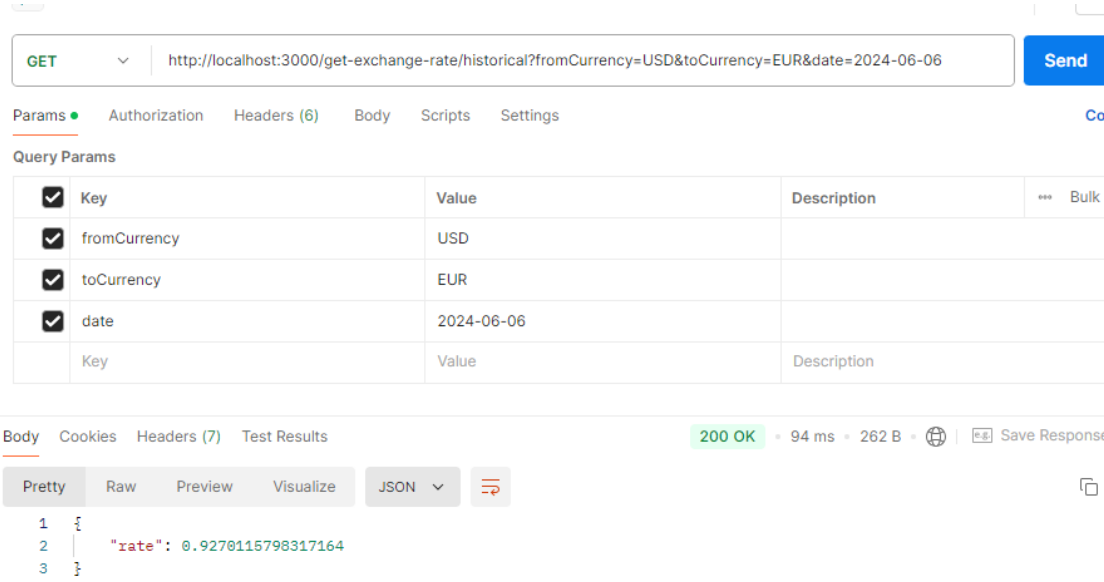
```
// Kiểm tra nếu ngày là ngày trong tương lai
if (this.isFutureDate(date)) {
  throw new HttpException(
    `Date cannot be in the future.`,
    HttpStatus.BAD_REQUEST,
  );
}
```

```
private isFutureDate(date: string): boolean {
  const today = new Date();
  const inputDate = new Date(date);
  return inputDate > today;
}
```

Ví dụ:



- Kết quả : chuyển từ USD sang EUR ngày 6/6/2024



II. File uploader

1. Mô tả chung :

File uploader cho phép người dùng:

- Upload các file thông thường (PDF, Word, hình ảnh, v.v.).

- Upload ảnh và nhận lại mã hóa Base64 của ảnh.
- Tải về các file đã upload.
- Xem danh sách các file đã upload trực tiếp từ giao diện.

Hệ thống được thiết kế với frontend (React) và backend(NestJS) giao tiếp thông qua HTTP API.

2. Chi tiết:

2.1 Frontend Code:

React State:

- uploadedFiles: Lưu danh sách file đã upload, được lấy từ API /file/list.
- base64: Lưu mã Base64 của ảnh sau khi upload thành công.
- selectedFile: Lưu file được chọn để upload.
- isLoading: Quản lý trạng thái loading, ngăn người dùng nhấn nút khi đang tải.
- errorMessage: Lưu thông báo lỗi khi xảy ra sự cố.

```
const App = () => {
  const [uploadedFiles, setUploadedFiles] = useState([]); // Danh sách file
  const [base64, setBase64] = useState(""); // Base64 của ảnh
  const [selectedFile, setSelectedFile] = useState(null); // File được chọn
  const [isLoading, setIsLoading] = useState(false); // Trạng thái đang tải
  const [errorMessage, setErrorMessage] = useState(""); // Lỗi
```

Định nghĩa đến URL của phần backend:

```
const API_URL = "http://localhost:3000/file"; // URL Backend
```

Các hàm chính:

a. fetchFIles : Lấy danh sách file đã upload

```
// Fetch danh sách file từ server
const fetchFiles = async () => {
  try {
    setIsLoading(true);
    const response = await axios.get(`${API_URL}/list`);
    setUploadedFiles(response.data); // Cập nhật danh sách file
    console.log(response.data);
  } catch (error) {
    setErrorMessage(
      error.response?.data?.message || "Failed to fetch files."
    );
  } finally {
    setIsLoading(false);
  }
};
```

- Gửi yêu cầu get đến API: /file/list
- Nếu thành công thì cập nhật danh sách file

b. handleFileUpload : xử lý upload file

```
const handleFileUpload = async () => {
  if (!selectedFile) {
    setErrorMessage("Please select a file to upload.");
    return;
  }

  const formData = new FormData();
  formData.append("file", selectedFile); // Key phải là "file"

  try {
    setIsLoading(true);
    await axios.post(`${API_URL}/upload`, formData); // Gửi file lên server
    setSelectedFile(null); // Reset file được chọn
    fetchFiles(); // Refresh danh sách file
  } catch (error) {
    setErrorMessage(
      error.response?.data?.message || "Failed to upload file."
    );
  } finally {
    setIsLoading(false);
  }
};
```

- Kiểm tra file được chọn hay chưa trước khi upload

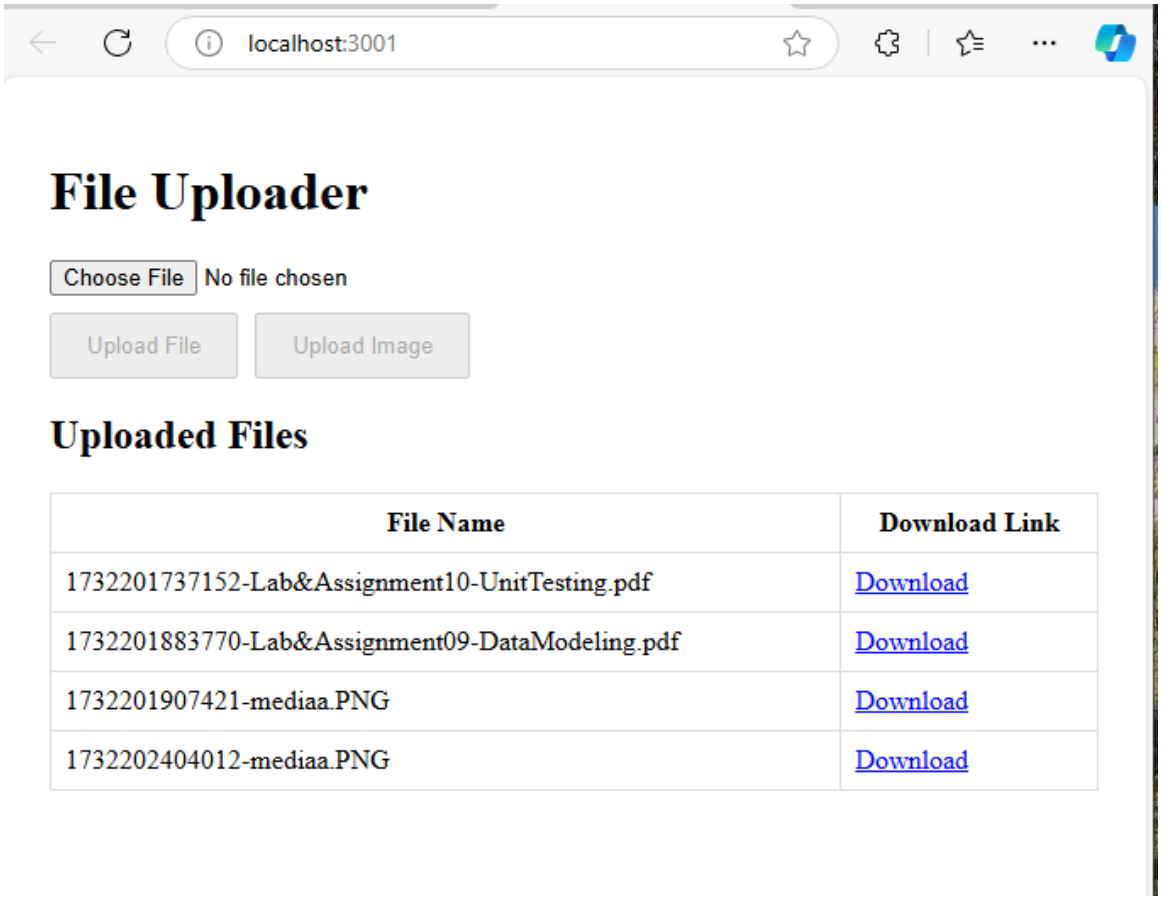
- Sử dụng FormData để gửi file qua key “file”
- Gửi yêu cầu Post đến API /file/upload
- Upload thành công thì gọi getchFiles để cập nhật lại danh sách file đã upload

c. handleImageUpliad : Upload ảnh và hiển thị mã Base 64 của ảnh

```
// Upload ảnh và lấy base64
const handleImageUpload = async () => {
  if (!selectedFile) {
    setErrorMessage("Please select an image to upload.");
    return;
  }

  const formData = new FormData();
  formData.append("image", selectedFile); // Key phải là "image"

  try {
    setIsLoading(true);
    const response = await axios.post(`${API_URL}/upload-image`, formData); // Gửi ảnh lên server
    setBase64(response.data.base64); // Lưu mã base64 trả về
  } catch (error) {
    setErrorMessage(
      error.response?.data?.message || "Failed to upload image."
    );
  } finally {
    setIsLoading(false);
  }
};
```

- Gửi file ảnh qua key “image” đến API /file/upload-image
- Nhận chuỗi Base64 từ server và hiển thị lên front-end

d. Giao diện hiển thị



- Input và Upload :

```jsx
{/* Input chọn file */}
<input
  type="file"
  onChange={(e) => setSelectedFile(e.target.files[0])}
  style={{ display: "block", marginBottom: "10px" }}
/>

{/* Nút hành động */}
<div style={{ display: "flex", gap: "10px" }}>
  <button
    onClick={handleFileUpload}
    disabled={!selectedFile || isLoading}
    style={{ padding: "10px 20px" }}
  >
    {isLoading ? "Uploading..." : "Upload File"}
  </button>
  <button
    onClick={handleImageUpload}
    disabled={!selectedFile || isLoading}
    style={{ padding: "10px 20px" }}
  >
    {isLoading ? "Uploading..." : "Upload Image"}
  </button>
</div>
```

- Hiển thị danh sách file

```jsx
{/* Danh sách file */}
    <h2 style={{ marginTop: "20px" }}>Uploaded Files</h2>
    {isLoading ? (
      <p>Loading...</p>
    ) : uploadedFiles.length === 0 ? (
      <p>No files uploaded yet.</p>
    ) : (

     <table style={{ width: "100%", borderCollapse: "collapse" }}>
        <thead>
          <tr>
            <th style={{ border: "1px solid #ddd", padding: "8px" }}>
              File Name
            </th>
            <th style={{ border: "1px solid #ddd", padding: "8px" }}>
              Download Link
```

```
            </th>
          </tr>
        </thead>
        <tbody>
          {uploadedFiles.map((file) => (
            <tr key={file.name}>
              <td style={{ border: "1px solid #ddd", padding: "8px" }}>
                {file.name}
              </td>
              <td style={{ border: "1px solid #ddd", padding: "8px" }}>
                <a
                  href={`http://localhost:3000${file.url}`}
                  download
                  style={{ color: "blue" }}
                >
                  Download
                </a>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    )}
```

- Hiển thị Base64 :
    - Với file ảnh được chọn, user nhấn vào nút Upload Image thì tự động hiển thị mã base64

File Uploader

Choose File chan-dung-k...424759.png

Upload File | Upload Image

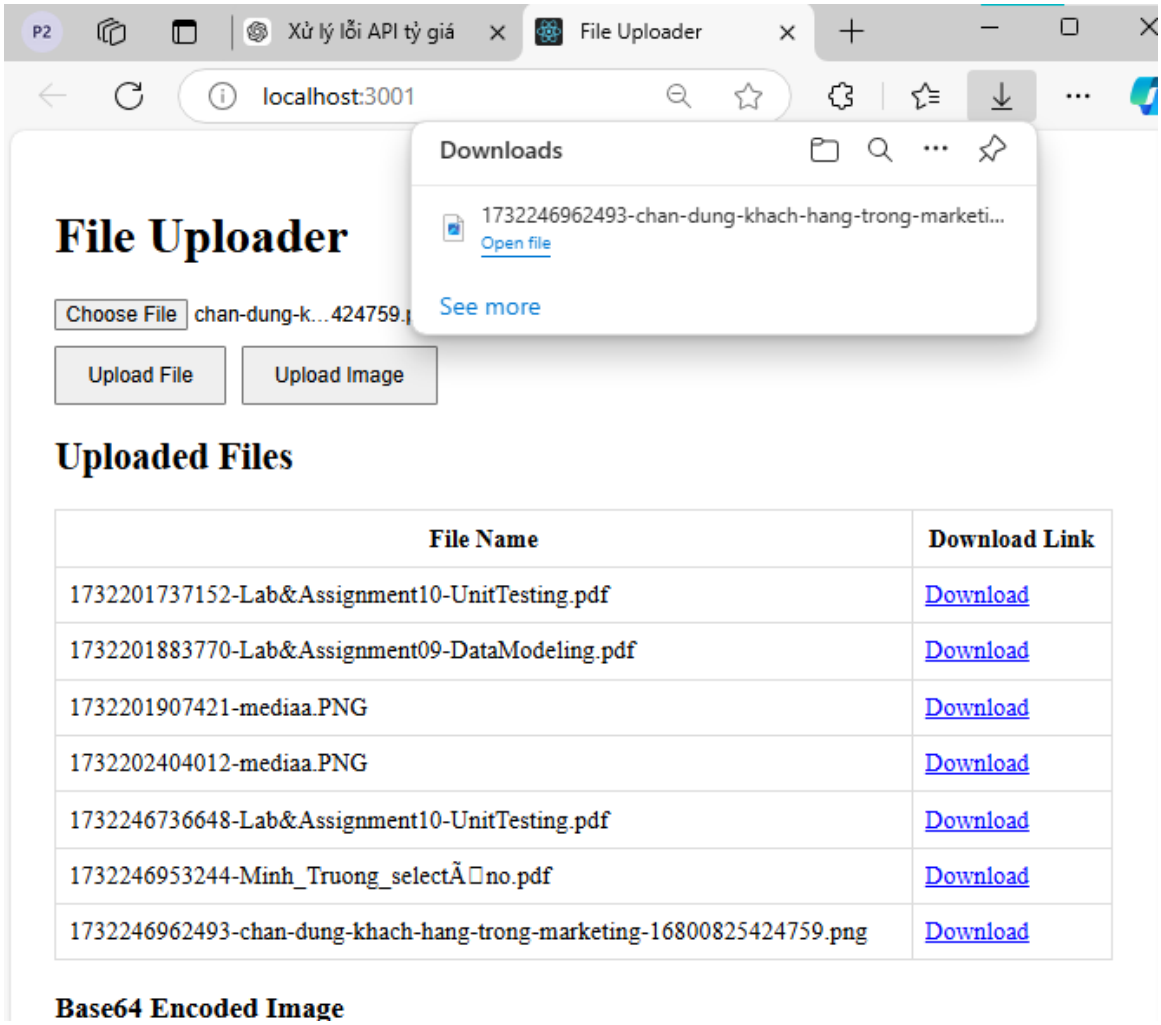
Uploaded Files

File Name	Download Link
1732201737152-Lab&Assignment10-UnitTesting.pdf	Download
1732201883770-Lab&Assignment09-DataModeling.pdf	Download
1732201907421-mediaa.PNG	Download
1732202404012-mediaa.PNG	Download
1732246736648-Lab&Assignment10-UnitTesting.pdf	Download
1732246953244-Minh_Truong_selectÃ□no.pdf	Download
1732246962493-chan-dung-khach-hang-trong-marketing-16800825424759.png	Download

Base64 Encoded Image

iVBORw0KGgoAAAANSUhEUgAAA1gAAAFRCAYAAACogdOJAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcw

- Download



2.2 Backend

a. Cấu trúc thư mục

```
src/
├── file/
│   ├── FileController.ts  # Điều phối yêu cầu HTTP
│   ├── FileService.ts      # Xử lý logic nghiệp vụ
│   ├── FileModule.ts       # Khởi tạo module file
├── main.ts                 # Điểm vào chính của ứng dụng
uploads/                    # Nơi lưu file được upload
```

b. Tệp cấu hình chính main.ts:

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  // Kích hoạt CORS
  app.enableCors({
    origin: 'http://localhost:3001', // Địa chỉ frontend
    methods: 'GET,HEAD,PUT,PATCH,POST,DELETE', // Các phương thức HTTP
được phép
    credentials: true,
  });

  await app.listen(3000); // Lắng nghe backend trên cổng 3000
}
bootstrap();
```

- Khởi tạo ứng dụng NestJS
- Bật CORS(định nghĩa giao thức, nguồn gốc và header nào được phép giao tiếp với nó) để cho phép frontend ở localhost :3001 gửi yêu cầu HTTP

c. FileController

```
import {
  Post,
  Get,
  Param,
  Res,
  UploadedFile,
  UseInterceptors,
  Controller,
} from '@nestjs/common';
import { FileService } from './FileService';
import { FileInterceptor } from '@nestjs/platform-express';
import { diskStorage } from 'multer';
import { extname } from 'path';

@Controller('file')
export class FileController {
  constructor(private readonly fileService: FileService) {}
```

```
// Upload File Thường
@Post('upload')
@UseInterceptors(
  FileInterceptor('file', {
    storage: diskStorage({
      destination: './uploads', // Đường dẫn lưu file
      filename: (req, file, cb) => {
        // Tạo tên file duy nhất với timestamp
        const uniqueName = `${Date.now()}-${file.originalname}`;
        cb(null, uniqueName);
      },
    }),
  }),
)
uploadFile(@UploadedFile() file: Express.Multer.File) {
  console.log('Uploaded file:', file); // Kiểm tra log file
  return this.fileService.handleFileUpload(file);
}

// Danh sách File
@Get('list')
listFiles() {
  return this.fileService.listUploadedFiles();
}

// Tải File về
@Get('download/:filename')
downloadFile(@Param('filename') filename: string, @Res() res) {
  return this.fileService.downloadFile(filename, res);
}

// Upload File Ảnh và Trả về Base64
@Post('upload-image')
@UseInterceptors(
  FileInterceptor('image', {
    storage: diskStorage({
      destination: './uploads',
      filename: (req, file, cb) => {
        // Tạo tên file duy nhất với timestamp
        const uniqueName = `${Date.now()}-${file.originalname}`;
        cb(null, uniqueName);
      },
    }),
    fileFilter: (req, file, cb) => {
      // Lọc chỉ cho phép upload ảnh (jpg, png, jpeg)
```

```
        const allowedExtensions = ['.jpg', '.jpeg', '.png'];
        if (
          !allowedExtensions.includes(extname(file.originalname).toLowerCa
se())
        ) {
          return cb(new Error('Only image files are allowed!'), false);
        }
        cb(null, true);
      },
    }),
  )
  uploadImage(@UploadedFile() file: Express.Multer.File) {
    console.log('Uploaded image:', file); // Kiểm tra log file
    return this.fileService.uploadImage(file);
  }
}
```

Định nghĩa các endpoint API đại diện cho từng chức năng cụ thể

d. FileService

Thư mục lưu trữ file của sever là trong thư mục uploads, và định nghĩa path đến uploads:

```
  private readonly uploadPath = join(__dirname, '..', '..', 'uploads');
```

Các hàm chính:

- handleFileUpload:

```
  // Xử lý Upload File Thường
  handleFileUpload(file: Express.Multer.File) {
    if (!file) {
      throw new Error('File upload failed: No file received');
    }

    // Xây dựng đường dẫn file chính xác
    const filePath = join(this.uploadPath, file.filename);
    console.log('File saved at:', filePath);

    return { filename: file.filename, path: filePath };
  }
```

- listUploadFIles

```
 // Trả về danh sách File
  listUploadedFiles() {
    try {
```

```
      const files = fs.readdirSync(this.uploadPath); // Đọc danh sách file
từ thư mục uploads
      return files.map((file) => ({
        name: file,
        url: `/file/download/${file}`, // Tạo URL tải xuống
      }));
    } catch (error) {
      console.error('Error reading files:', error.message);
      throw new Error('Could not list uploaded files');
    }
  }
```

- downloadFile

```
// Xử lý Tải File về
  downloadFile(filename: string, res: Response) {
    try {
      // Xây dựng đường dẫn chính xác tới file
      const filePath = join(this.uploadPath, filename);
      console.log('Downloading file from:', filePath);

      // Kiểm tra nếu file không tồn tại
      if (!fs.existsSync(filePath)) {
        console.error('File not found:', filePath);
        res.status(404).send('File not found');
        return;
      }

      // Thiết lập header để trình duyệt tải file xuống
      res.setHeader(
        'Content-Disposition',
        `attachment; filename="${filename}"`,
      );
      res.setHeader('Content-Type', 'application/octet-stream');

      // Gửi file về cho client
      res.sendFile(filePath);
    } catch (error) {
      console.error('Error during file download:', error.message);
      res.status(500).send('An error occurred while downloading the
file');
    }
  }
```

Chức năng: Tìm file trong thư mục uploads -> nếu file tồn tại trả về file với header : “Content-Disposition” để tải xuống, trả về HTTP 404 nếu không tải được.

- uploadImage

```
// Xử lý Upload Ảnh và Trả về Base64
  uploadImage(file: Express.Multer.File) {
    if (!file) {
      throw new Error('File upload failed: No file received');
    }

    try {
      // Xây dựng đường dẫn tới file
      const filePath = join(this.uploadPath, file.filename);
      console.log('Image saved at:', filePath);

      const imageBuffer = fs.readFileSync(filePath); // Đọc file từ disk
      const base64 = imageBuffer.toString('base64'); // Chuyển sang base64

      return { base64 };
    } catch (error) {
      console.error('Error processing image:', error.message);
      throw new Error('Could not process the image');
    }
  }
```

Chức năng:

- Nhận file từ front-end
- Lưu vào uploads
- Đọc nội dung ảnh và trả về mã Base64

e. Middleware Muller

Xử lý file từ request, định nghĩa địa chỉ lưu file cũng như thông tin file

```
@UseInterceptors(FileInterceptor('file', {
 storage: diskStorage({
  destination: './uploads',
  filename: (req, file, callback) => {
   const uniqueName = `${Date.now()}-${file.originalname}`;
   callback(null, uniqueName);
  },
 }),
}))
```