**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**ĐỒ ÁN MÔN HỌC**

**Thiết kế các WebAPI**

**cho ứng dụng truy vấn trực tuyến trên Office365**

**PHẠM MINH TRƯỜNG**

Truong.pm215292@sis.hust.edu.vn

**Ngành Công nghệ thông tin**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | ShapeThS. Nguyễn Đức Tiến Text Box |
| **Khoa:** | Kỹ thuật Máy tính |
| **Trường:** | Công nghệ Thông tin và Truyền thông |
| **HÀ NỘI, 1/2025** | |

Mục lục

[I. Mở đầu 3](#_Toc188083269)

[A. Giới thiệu 3](#_Toc188083270)

[B. Công cụ, ngôn ngữ sử dụng 3](#_Toc188083271)

[II. Triển khai hệ thống 7](#_Toc188083272)

[1. Hash API 7](#_Toc188083273)

[2. Qrcode API 12](#_Toc188083274)

[3. GetExchangeRate API 14](#_Toc188083275)

[4. Historical Rates API 17](#_Toc188083276)

[5. File uploader 19](#_Toc188083277)

[5.1. Mô tả chung : 19](#_Toc188083278)

[5.2. Chi tiết: 20](#_Toc188083279)

[6. Tài liệu mô tả cho các API bằng Swagger 32](#_Toc188083280)

[III. Tích hợp và kiểm thử 36](#_Toc188083281)

[1. Phương pháp kiểm thử 36](#_Toc188083282)

[2. Thư viện sử dụng để kiểm thử: Jest 36](#_Toc188083283)

[3. Chương trình kiểm thử 38](#_Toc188083284)

[3.1 Hash Api Test 38](#_Toc188083285)

[3.2 Qrcode API Test 41](#_Toc188083286)

[3.3 GER API Test 43](#_Toc188083287)

[3.4 File Api Test 48](#_Toc188083288)

[4. Kết quả kiểm thử 51](#_Toc188083289)

# Mở đầu

## Giới thiệu

Dự án này nhằm xây dựng một hệ thống quản lý API đa chức năng, tập trung vào việc phát triển các API tiện ích và tích hợp các dịch vụ bên thứ ba để giải quyết các bài toán thực tế. Mục tiêu chính của dự án là cung cấp các giải pháp lập trình hiệu quả, dễ sử dụng, và có khả năng mở rộng.

Hệ thống bao gồm các tính năng nổi bật như sau:

1. **Mã hóa và xử lý dữ liệu:**
   * Phát triển API cho phép người dùng mã hóa văn bản bằng các thuật toán băm như MD5, nhằm đảm bảo tính bảo mật và xác thực dữ liệu.
2. **Tạo QR Code:**
   * Xây dựng API để tạo mã QR từ các đoạn văn bản nhập vào, sử dụng các thư viện npm để tối ưu hóa thời gian phát triển.
3. **Tích hợp API tỷ giá hối đoái:**
   * Tích hợp các API từ Vietcombank để truy cập tỷ giá hối đoái thời gian thực và lịch sử, cung cấp dữ liệu chính xác cho các bài toán kinh doanh và tài chính.
4. **Quản lý tệp:**
   * Hỗ trợ tải lên, liệt kê, tải xuống tệp và mã hóa hình ảnh thành chuỗi Base64. Tích hợp giao diện kéo-thả tệp để cải thiện trải nghiệm người dùng.
5. **Kiểm thử và phát triển:**
   * Áp dụng phương pháp Test-Driven Development (TDD) để xây dựng các unittest, đảm bảo chất lượng và tính chính xác của hệ thống trước mỗi lần triển khai.

## Công cụ, ngôn ngữ sử dụng

**1. NestJS**

* **Loại công nghệ:** Framework phát triển backend.
* **Vai trò:**
  + NestJS cung cấp một kiến trúc module hóa, giúp tổ chức code rõ ràng và dễ bảo trì.
  + Hỗ trợ mạnh mẽ cho các tiêu chuẩn như Dependency Injection và Decorators.
  + Tích hợp dễ dàng với các công nghệ khác như TypeORM, Swagger, hoặc thư viện kiểm tra như Jest.
* **Lý do sử dụng:**
  + Phù hợp với các ứng dụng cần khả năng mở rộng.
  + Tích hợp mạnh mẽ với , giúp code dễ đọc và giảm lỗi.

**2. Swagger (OpenAPI)**

* **Loại công nghệ:** Công cụ tài liệu hóa API.
* **Vai trò:**
  + Tạo giao diện trực quan cho việc sử dụng và kiểm thử API.
  + Cung cấp tài liệu đầy đủ về các endpoint, tham số đầu vào, phản hồi và mã lỗi.
  + Hỗ trợ khả năng kiểm thử API trực tiếp từ trình duyệt.
* **Lý do sử dụng:**
  + Tăng tính minh bạch và hỗ trợ cho các nhà phát triển tích hợp API.
  + Tài liệu được tự động đồng bộ hóa với mã nguồn, giảm rủi ro tài liệu lỗi thời.

**3.**

* **Loại công nghệ:** Ngôn ngữ lập trình (superset của JavaScript).
* **Vai trò:**
  + Cung cấp các tính năng như kiểu tĩnh (static typing), giúp giảm lỗi runtime.
  + Tăng cường khả năng đọc hiểu và bảo trì code.
* **Lý do sử dụng:**
  + Phù hợp với dự án lớn và yêu cầu tính bảo trì cao.
  + Được hỗ trợ tốt trong NestJS và các công cụ liên quan.

**4. Jest**

* **Loại công nghệ:** Thư viện kiểm thử tự động.
* **Vai trò:**
  + Kiểm thử các module backend (Unit Test) và đảm bảo các chức năng hoạt động đúng như mong đợi.
  + Hỗ trợ tính năng mock (giả lập) để kiểm thử trong môi trường cô lập.
* **Lý do sử dụng:**
  + Tích hợp liền mạch với NestJS.
  + Cung cấp báo cáo chi tiết về kết quả kiểm thử.

**5. Multer**

* **Loại công nghệ:** Middleware cho xử lý file upload.
* **Vai trò:**
  + Hỗ trợ xử lý file tải lên từ client.
  + Lưu trữ file vào hệ thống hoặc đọc nội dung file để xử lý thêm.
* **Lý do sử dụng:**
  + Dễ dàng tích hợp với NestJS.
  + Hiệu quả trong việc quản lý file.

**6. Node.js**

* **Loại công nghệ:** Nền tảng backend.
* **Vai trò:**
  + Xử lý các yêu cầu HTTP và cung cấp các endpoint API.
  + Cung cấp môi trường runtime cho NestJS.
* **Lý do sử dụng:**
  + Hiệu suất cao nhờ cơ chế xử lý bất đồng bộ (non-blocking I/O).
  + Phổ biến trong việc xây dựng API RESTful.

**7. Axios**

* **Loại công nghệ:** Thư viện HTTP client.
* **Vai trò:**
  + Gửi các yêu cầu HTTP đến các API bên thứ ba, ví dụ: lấy tỷ giá hối đoái từ nguồn bên ngoài.
* **Lý do sử dụng:**
  + Cú pháp đơn giản và tích hợp dễ dàng.
  + Hỗ trợ mạnh mẽ cho các yêu cầu đồng bộ và xử lý lỗi.

**8. QRCode**

* **Loại công nghệ:** Thư viện tạo mã QR.
* **Vai trò:**
  + Tạo mã QR từ chuỗi văn bản được cung cấp bởi người dùng.
* **Lý do sử dụng:**
  + Nhẹ, dễ sử dụng, và tích hợp liền mạch với Node.js.
  + Trả về mã QR ở định dạng Base64 hoặc hình ảnh.

**9. Crypto (Node.js Built-in Library)**

* **Loại công nghệ:** Thư viện mã hóa tích hợp trong Node.js.
* **Vai trò:**
  + Xử lý mã hóa dữ liệu như tạo mã băm (hash) bằng các thuật toán MD5, SHA256,...
* **Lý do sử dụng:**
  + An toàn và đáng tin cậy nhờ sự hỗ trợ trực tiếp từ Node.js.
  + Hiệu quả trong các tác vụ mã hóa.

**10. Express**

* **Loại công nghệ:** Framework web (tích hợp trong NestJS).
* **Vai trò:**
  + Xử lý các yêu cầu HTTP và cung cấp khả năng định tuyến cho API.
* **Lý do sử dụng:**
  + Được tích hợp tự động trong NestJS.
  + Nhẹ và dễ sử dụng cho các ứng dụng nhỏ hoặc lớn.

**11. File System (fs - Node.js Built-in Library)**

* **Loại công nghệ:** Thư viện tích hợp trong Node.js.
* **Vai trò:**
  + Quản lý file trên hệ thống, bao gồm lưu trữ, đọc, hoặc ghi file.
* **Lý do sử dụng:**
  + Tích hợp tốt với Node.js, không cần cài đặt thêm thư viện bên ngoài.
  + Phù hợp để xây dựng các chức năng tải file và xử lý file.

**12. xml2js**

* **Loại công nghệ:** Thư viện chuyển đổi XML sang JSON.
* **Vai trò:**
  + Phân tích dữ liệu từ các API trả về định dạng XML, như API tỷ giá hối đoái.
* **Lý do sử dụng:**
  + Dễ dàng chuyển đổi XML thành đối tượng JSON để xử lý trong code.

# Triển khai hệ thống

## 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

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

* Error handle:
  + Chuỗi rỗng: Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

    Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động

* + Thuật toán không đúng hoặc không được hỗ trợ: Ảnh có chứa văn bản, phần mềm, Biểu tượng máy tính, Trang web

    Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Mô tả được tạo tự động

TestAPI:

Ảnh có chứa văn bản, phần mềm, ảnh chụp màn hình, số

Mô tả được tạo tự động

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

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, phần mềm, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

## Qrcode API

1. Chức năng chính: Chuyển text thành ảnh Qrcode
2. Mô tả chi tiết:
   * Input: Văn bản
   * Output: Ảnh QR Code
   * Process

Service:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Controller:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* 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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

## GetExchangeRate API

* 1. 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)
  2. 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

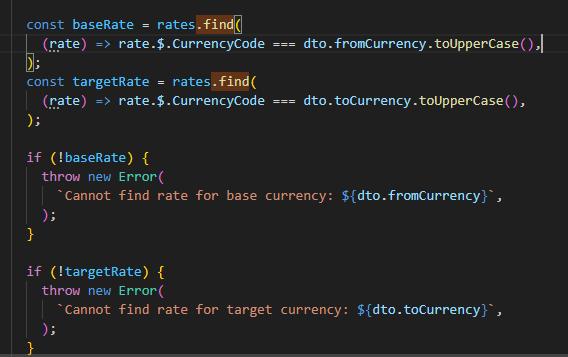
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Mô tả được tạo tự động

* 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” 
  + Tìm tỷ giá cho loại tiền tệ so với VND qua trường Buy của ExrateList Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

    Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

    Mô tả được tạo tự động
  + -> Kết quả trả về là baseBuyRate/targetBuyRate
* Error Handle:
  + Không tìm thấy tỷ giá: Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

    Mô tả được tạo tự động
  + Không tìm thấy loại tiền tệ 
  + Lỗi khi kết nối API : Ảnh có chứa ảnh chụp màn hình, văn bản, Phông chữ

    Mô tả được tạo tự động
* Ví dụ:

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

Mô tả được tạo tự động

## Historical Rates API

1. 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.
2. 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

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Mô tả được tạo tự động

* 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 Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

    Mô tả được tạo tự động
  + Ngày không đúng định dạng

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Ví dụ :

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Mô tả được tạo tự động

* + Ngày trong tương lai Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

    Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

    Mô tả được tạo tự động

Ví dụ:

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Mô tả được tạo tự động

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

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Mô tả được tạo tự động

## File uploader

### 5.1. Mô tả chung :

File uploader cho phép người dùng:

* + 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.

### 5.2. Chi tiết:

#### 5.2.1. 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.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Định nghĩa đến URL của phần backend: Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, đồng hồ

Mô tả được tạo tự động

#### Các hàm chính:

* 1. fetchFIles : Lấy danh sách file đã upload

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, Phông chữ

Mô tả được tạo tự động

* Gửi yêu cầu get đến API: /file/list
* Nếu thành công thì cập nhật danh sách file
  1. handleFileUpload : xử lý upload file

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

Mô tả được tạo tự động

* Input và Upload :

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Hiển thị danh sách file

  {/\* 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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

Mô tả được tạo tự động

* DownloadẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

  Mô tả được tạo tự động

#### 5.2.2 Backend

* 1. Cấu trúc thư mục

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, số

Mô tả được tạo tự động

* 1. 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
  1. 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).toLowerCase())

        ) {

          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

* 1. FileService

Thư mục lưu trữ file của sever là trong thư mục uploads, và định nghĩa path đến 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
  1. 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);

},

}),

}))

## Tài liệu mô tả cho các API bằng Swagger

Em đã viết mô tả cho các API bằng swagger

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

**Tóm Tắt Các Endpoint**

**Nhóm Hash**

* **Endpoint:** /hash (POST)
* **Mục đích:** Tạo mã băm từ văn bản và thuật toán được cung cấp.
* **Chi tiết:**
  + **Input:**

json

{

"text": "Hello World",

"algorithm": "sha256"

}

* + **Output (201):**

json

{

"hash": "5d41402abc4b2a76b9719d911017c592"

}

* + **Lỗi (400):** Dữ liệu không hợp lệ (thiếu text hoặc algorithm).

**Nhóm QRCode**

* **Endpoint:** /qrcode/generate (GET)
* **Mục đích:** Tạo mã QR từ văn bản đầu vào.
* **Chi tiết:**
  + **Input (Query):**
    - data: Văn bản để tạo mã QR.
    - Ví dụ: Hello World.
  + **Output (200):**

json

{

"qrCode": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA..."

}

* + **Lỗi (400):** Dữ liệu đầu vào không hợp lệ.

**Nhóm Exchange Rates**

1. **Endpoint:** /get-exchange-rate/current-date (GET)
   * **Mục đích:** Lấy tỷ giá hối đoái hiện tại.
   * **Input (Query):**
     + fromCurrency: Mã tiền tệ gốc (VD: USD).
     + toCurrency: Mã tiền tệ đích (VD: VND).
     + date: Ngày cần lấy tỷ giá (YYYY-MM-DD).
   * **Output (200):**

json

{

"fromCurrency": "USD",

"toCurrency": "VND",

"rate": 23000

}

* + **Lỗi (400):** Thiếu thông tin hoặc định dạng ngày không hợp lệ.

1. **Endpoint:** /get-exchange-rate/historical (GET)
   * **Mục đích:** Lấy tỷ giá hối đoái lịch sử.
   * **Input (Query):**
     + Tương tự như endpoint trên.
   * **Output (200):**

json

{

"date": "2024-01-01",

"fromCurrency": "USD",

"toCurrency": "EUR",

"rate": 0.85

}

* + **Lỗi (400):** Ngày không hợp lệ hoặc không có dữ liệu.

**Nhóm File**

1. **Endpoint:** /file/upload (POST)
   * **Mục đích:** Upload một tệp.
   * **Input:** Tệp tin (multipart/form-data).
   * **Output (201):**

json

{

"message": "File uploaded successfully",

"filePath": "/uploads/example.txt"

}

* + **Lỗi (400):** Quá trình upload thất bại.

1. **Endpoint:** /file/list (GET)
   * **Mục đích:** Lấy danh sách các tệp đã upload.
   * **Output (200):**

json

[

"example1.txt",

"example2.png",

"example3.pdf"

]

1. **Endpoint:** /file/download/{filename} (GET)
   * **Mục đích:** Tải tệp đã upload về máy.
   * **Input:**
     + filename: Tên tệp cần tải.
   * **Output (200):** Tệp được tải thành công.
   * **Lỗi (404):** Không tìm thấy tệp.
2. **Endpoint:** /file/upload-image (POST)
   * **Mục đích:** Upload ảnh và trả về chuỗi Base64.
   * **Input:** Ảnh (jpg, png, jpeg).
   * **Output (201):**

json

{

"message": "Image uploaded successfully",

"base64": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA..."

}

* + **Lỗi (400):** Upload thất bại.

# Tích hợp và kiểm thử

## Phương pháp kiểm thử

Trong dự án này, em đã áp dụng phương pháp **Test-Driven Development (TDD)**. Quá trình TDD được thực hiện theo các bước sau:

* **Viết unittest trước khi phát triển chức năng.**
* **Phát triển mã nguồn để vượt qua bài kiểm thử.**
* **Chạy lại kiểm thử tự động.**
* **Tích hợp kiểm thử.**

## Thư viện sử dụng để kiểm thử: Jest

Jest là một thư viện kiểm thử JavaScript phổ biến, được phát triển bởi Facebook. Jest chủ yếu được sử dụng để viết và chạy các bài kiểm thử cho các ứng dụng JavaScript và .

**Cách Jest được sử dụng trong dự án**

1. **Tạo môi trường kiểm thử**:
   * beforeEach được dùng để thiết lập lại môi trường trước mỗi test case, đảm bảo các test độc lập với nhau.
   * Test.createTestingModule trong NestJS hỗ trợ mô phỏng (mock) module để kiểm thử.

beforeEach(async () => {

const module: TestingModule = await Test.createTestingModule({

providers: [HashService],

}).compile();

hashService = module.get<HashService>(HashService);

});

1. **Viết các test case**:
   * Các test case được định nghĩa bằng it (một từ khóa Jest để định nghĩa test case).
   * Mỗi test case xác định một kỳ vọng cụ thể (expect) và kiểm tra kết quả thực tế so với kỳ vọng.

Ví dụ:

it('should generate a correct hash for MD5', () => {

const text = 'Hello';

const algorithm = 'md5';

const expectedHash = crypto

.createHash(algorithm)

.update(text)

.digest('hex');

const result = hashService.getHash(text, algorithm);

expect(result).toBe(expectedHash);

});

1. **Xử lý lỗi và ngoại lệ**:
   * Jest hỗ trợ kiểm tra các lỗi và ngoại lệ bằng cách sử dụng toThrowError. Điều này được dùng để đảm bảo các trường hợp lỗi (như đầu vào không hợp lệ hoặc lỗi thuật toán) được xử lý đúng cách.

Ví dụ:

it('should throw an error if text is null', () => {

expect(() => {

hashService.getHash(null, 'md5');

}).toThrowError(new BadRequestException('Input text cannot be empty.'));

});

1. **Mocking**:
   * Sử dụng jest.spyOn để mock một function của thư viện bên ngoài (crypto.createHash) nhằm kiểm tra cách ứng dụng xử lý lỗi ngoại lệ.

Ví dụ:

jest.spyOn(crypto, 'createHash').mockImplementation(() => {

throw new Error('Unexpected crypto error');

});

expect(() => {

hashService.getHash('Hello', 'md5');

}).toThrowError(

new BadRequestException(

'Failed to generate hash: Unexpected crypto error',

),

);

## Chương trình kiểm thử

### Hash Api Test

**1. Test Case: Service Initialization**

* **Mô tả:** Kiểm tra xem HashService có được định nghĩa và khởi tạo đúng hay không.
* **Kỳ vọng:** hashService phải được định nghĩa.

**Mã nguồn:**

it('should be defined', () => {

expect(hashService).toBeDefined();

});

**2. Test Case: MD5 Hash Generation**

* **Mô tả:** Kiểm tra xem phương thức getHash có tạo ra chuỗi băm MD5 đúng cho một đoạn văn bản không.
* **Đầu vào:** Text = "Hello", Algorithm = "md5".
* **Kỳ vọng:** Chuỗi băm phải trùng với kết quả MD5 mong đợi.

**Mã nguồn:**

it('should generate a correct hash for MD5', () => {

const text = 'Hello';

const algorithm = 'md5';

const expectedHash = crypto

.createHash(algorithm)

.update(text)

.digest('hex');

const result = hashService.getHash(text, algorithm);

expect(result).toBe(expectedHash);

});

**3. Test Case: SHA-256 Hash Generation**

* **Mô tả:** Kiểm tra xem phương thức getHash có tạo ra chuỗi băm SHA-256 đúng cho một đoạn văn bản không.
* **Đầu vào:** Text = "Hello", Algorithm = "sha256".
* **Kỳ vọng:** Chuỗi băm phải trùng với kết quả SHA-256 mong đợi.

**Mã nguồn:**

it('should generate a correct hash for SHA-256', () => {

const text = 'Hello';

const algorithm = 'sha256';

const expectedHash = crypto

.createHash(algorithm)

.update(text)

.digest('hex');

const result = hashService.getHash(text, algorithm);

expect(result).toBe(expectedHash);

});

**4. Test Case: Null Input Text**

* **Mô tả:** Kiểm tra xem phương thức getHash có ném lỗi khi đầu vào là null không.
* **Đầu vào:** Text = null, Algorithm = "md5".
* **Kỳ vọng:** Ném lỗi BadRequestException với thông báo "Input text cannot be empty."

**Mã nguồn:**

it('should throw an error if text is null', () => {

expect(() => {

hashService.getHash(null, 'md5');

}).toThrowError(new BadRequestException('Input text cannot be empty.'));

});

**5. Test Case: Empty Input Text**

* **Mô tả:** Kiểm tra xem phương thức getHash có ném lỗi khi đầu vào là chuỗi rỗng không.
* **Đầu vào:** Text = "", Algorithm = "md5".
* **Kỳ vọng:** Ném lỗi BadRequestException với thông báo "Input text cannot be empty."

**Mã nguồn:**

it('should throw an error if text is empty', () => {

expect(() => {

hashService.getHash('', 'md5');

}).toThrowError(new BadRequestException('Input text cannot be empty.'));

});

**6. Test Case: Invalid Algorithm**

* **Mô tả:** Kiểm tra xem phương thức getHash có ném lỗi khi thuật toán băm không được hỗ trợ không.
* **Đầu vào:** Text = "Hello", Algorithm = "unsupported-algorithm".
* **Kỳ vọng:** Ném lỗi BadRequestException với thông báo "Invalid hashing algorithm."

**Mã nguồn:**

it('should throw an error if the algorithm is invalid', () => {

const invalidAlgorithm = 'unsupported-algorithm';

expect(() => {

hashService.getHash('Hello', invalidAlgorithm);

}).toThrowError(

new BadRequestException(

`Invalid hashing algorithm. Supported algorithms are: ${crypto

.getHashes()

.join(', ')}`,

),

);

});

**7. Test Case: Unexpected Crypto Error**

* **Mô tả:** Kiểm tra xem phương thức getHash có xử lý lỗi bất ngờ từ thư viện crypto không.
* **Đầu vào:** Text = "Hello", Algorithm = "md5".
* **Kỳ vọng:** Ném lỗi BadRequestException với thông báo "Failed to generate hash: Unexpected crypto error."

**Mã nguồn:**

it('should throw an error if crypto throws an unexpected error', () => {

jest.spyOn(crypto, 'createHash').mockImplementation(() => {

throw new Error('Unexpected crypto error');

});

expect(() => {

hashService.getHash('Hello', 'md5');

}).toThrowError(

new BadRequestException(

'Failed to generate hash: Unexpected crypto error',

),

);

});

### Qrcode API Test

**QrCodeController Test Cases**

1. **Test Case: Kiểm tra khởi tạo controller**
   * **Mô tả:** Xác minh rằng QrCodeController được định nghĩa đúng.
   * **Mã nguồn:**

it('should be defined', () => {

expect(qrCodeController).toBeDefined();

});

1. **Test Case: Tạo QR Code thành công và gửi phản hồi**
   * **Mô tả:** Kiểm tra nếu phương thức generateQRCode tạo ra QR Code và gửi đúng dữ liệu phản hồi.
   * **Đầu vào:** Data = "Test Data".
   * **Kỳ vọng:**
     + Header phản hồi được đặt là Content-Type: image/png.
     + Phương thức res.send được gọi với buffer QR Code.
     + generateQRCode của service được gọi với đúng dữ liệu.
   * **Mã nguồn:**

it('should generate a QR Code and send it in response', async () => {

const data = 'Test Data';

const mockBuffer = Buffer.from('mock-qr-code');

jest.spyOn(qrCodeService, 'generateQRCode').mockResolvedValue(mockBuffer);

const res = {

setHeader: jest.fn(),

send: jest.fn(),

} as unknown as Response;

await qrCodeController.generateQRCode(data, res);

expect(res.setHeader).toHaveBeenCalledWith('Content-Type', 'image/png');

expect(res.send).toHaveBeenCalledWith(mockBuffer);

expect(qrCodeService.generateQRCode).toHaveBeenCalledWith(data);

});

1. **Test Case: Ném lỗi khi dữ liệu rỗng**
   * **Mô tả:** Xác minh rằng lỗi được ném nếu dữ liệu truyền vào trống.
   * **Đầu vào:** Data = "".
   * **Kỳ vọng:** Ném lỗi Data is required to generate a QR Code.
   * **Mã nguồn:**

it('should throw an error if data is empty', async () => {

const res = {

setHeader: jest.fn(),

send: jest.fn(),

} as unknown as Response;

await expect(qrCodeController.generateQRCode('', res)).rejects.toThrowError(

'Data is required to generate a QR Code.',

);

});

**QrCodeService Test Cases**

1. **Test Case: Kiểm tra khởi tạo service**
   * **Mô tả:** Xác minh rằng QrCodeService được định nghĩa đúng.
   * **Mã nguồn:**

it('should be defined', () => {

expect(qrCodeService).toBeDefined();

});

1. **Test Case: Tạo QR Code thành công**
   * **Mô tả:** Xác minh rằng generateQRCode tạo ra QR Code đúng cho dữ liệu hợp lệ.
   * **Đầu vào:** Data = "Test Data".
   * **Kỳ vọng:** Trả về buffer QR Code chính xác.
   * **Mã nguồn:**

it('should generate a QR Code for valid input', async () => {

const data = 'Test Data';

const mockBuffer = Buffer.from('mock-qr-code');

jest.spyOn(QRCode, 'toBuffer').mockResolvedValue(mockBuffer);

const result = await qrCodeService.generateQRCode(data);

expect(result).toBe(mockBuffer);

expect(QRCode.toBuffer).toHaveBeenCalledWith(data);

});

1. **Test Case: Xử lý lỗi khi tạo QR Code thất bại**
   * **Mô tả:** Xác minh rằng lỗi được ném khi việc tạo QR Code thất bại.
   * **Đầu vào:** Data = "Test Data".
   * **Kỳ vọng:** Ném lỗi Failed to generate QR Code.
   * **Mã nguồn:**

it('should throw an error if QRCode generation fails', async () => {

const data = 'Test Data';

jest

.spyOn(QRCode, 'toBuffer')

.mockRejectedValue(new Error('QR Code error'));

await expect(qrCodeService.generateQRCode(data)).rejects.toThrowError(

'Failed to generate QR Code: QR Code error',

);

});

### GER API Test

**Test Cases cho GerController**

1. **Test Case: Controller được định nghĩa**
   * **Mô tả:** Kiểm tra xem Controller được khởi tạo thành công.
   * **Kỳ vọng:** Controller phải được định nghĩa.
   * **Mã nguồn:**

it('should be defined', () => {

expect(controller).toBeDefined();

});

1. **Test Case: Trả về tỷ giá chính xác**
   * **Mô tả:** Kiểm tra nếu Controller gọi Service để trả về tỷ giá cho đầu vào hợp lệ.
   * **Đầu vào:** fromCurrency: 'USD', toCurrency: 'VND'.
   * **Kỳ vọng:**
     + Phương thức getExchangeRate được gọi với đúng DTO.
     + Controller trả về tỷ giá chính xác.
   * **Mã nguồn:**

it('should return exchange rate for valid input', async () => {

const dto: getExchangeRateDto = { fromCurrency: 'USD', toCurrency: 'VND' };

const mockRate = 23000;

jest.spyOn(gerService, 'getExchangeRate').mockResolvedValue(mockRate);

const result = await controller.getCurrentRate(dto);

expect(result.rate).toBe(mockRate);

expect(gerService.getExchangeRate).toHaveBeenCalledWith(dto);

});

1. **Test Case: Xử lý lỗi từ Service**
   * **Mô tả:** Kiểm tra nếu Controller ném lỗi đúng khi Service gặp lỗi.
   * **Đầu vào:** fromCurrency: 'USD', toCurrency: 'VND'.
   * **Kỳ vọng:** Controller ném lỗi "Currency not found".
   * **Mã nguồn:**

it('should throw an error if service throws', async () => {

const dto: getExchangeRateDto = { fromCurrency: 'USD', toCurrency: 'VND' };

jest

.spyOn(gerService, 'getExchangeRate')

.mockRejectedValue(new Error('Currency not found'));

await expect(controller.getCurrentRate(dto)).rejects.toThrowError(

'Currency not found',

);

});

**Test Cases cho GerService**

1. **Test Case: Service được định nghĩa**
   * **Mô tả:** Kiểm tra xem Service được khởi tạo thành công.
   * **Kỳ vọng:** Service phải được định nghĩa.
   * **Mã nguồn:**

it('should be defined', () => {

expect(service).toBeDefined();

});

1. **Test Case: Xử lý khi không có dữ liệu từ API**
   * **Mô tả:** Kiểm tra nếu Service xử lý lỗi khi không nhận được dữ liệu từ API.
   * **Đầu vào:** fromCurrency: 'USD', toCurrency: 'EUR'.
   * **Kỳ vọng:** Ném lỗi "No response data received from API".
   * **Mã nguồn:**

it('should throw an error if API response is empty', async () => {

const dto: getExchangeRateDto = { fromCurrency: 'USD', toCurrency: 'EUR' };

(lastValueFrom as jest.Mock).mockResolvedValue({ data: null });

await expect(service.getExchangeRate(dto)).rejects.toThrowError(

'No response data received from API',

);

});

1. **Test Case: Trả về tỷ giá chính xác**
   * **Mô tả:** Kiểm tra nếu Service trả về đúng tỷ giá khi API phản hồi dữ liệu hợp lệ.
   * **Đầu vào:** fromCurrency: 'USD', toCurrency: 'EUR'.
   * **Kỳ vọng:** Tỷ giá được tính đúng.
   * **Mã nguồn:**

it('should return the correct exchange rate for valid currencies', async () => {

const dto: getExchangeRateDto = { fromCurrency: 'USD', toCurrency: 'EUR' };

const mockApiResponse = `

<ExrateList>

<Exrate CurrencyCode="USD" Buy="25221.00" />

<Exrate CurrencyCode="EUR" Buy="27852.00" />

</ExrateList>

`;

(lastValueFrom as jest.Mock).mockResolvedValue({ data: mockApiResponse });

(parseStringPromise as jest.Mock).mockResolvedValue({

ExrateList: {

Exrate: [

{ $: { CurrencyCode: 'USD', Buy: '25221.00' } },

{ $: { CurrencyCode: 'EUR', Buy: '27852.00' } },

],

},

});

const rate = await service.getExchangeRate(dto);

expect(rate).toBeCloseTo(25221.0 / 27852.0);

});

1. **Test Case: Không tìm thấy đơn vị tiền tệ**
   * **Mô tả:** Kiểm tra nếu Service ném lỗi khi không tìm thấy đơn vị tiền tệ trong API.
   * **Đầu vào:** fromCurrency: 'USD', toCurrency: 'XYZ'.
   * **Kỳ vọng:** Ném lỗi "Cannot find rates for the specified currencies".
   * **Mã nguồn:**

it('should throw an error if currency is not found', async () => {

const dto: getExchangeRateDto = { fromCurrency: 'USD', toCurrency: 'XYZ' };

const mockApiResponse = `

<ExrateList>

<Exrate CurrencyCode="USD" Buy="25221.00" />

</ExrateList>

`;

(lastValueFrom as jest.Mock).mockResolvedValue({ data: mockApiResponse });

(parseStringPromise as jest.Mock).mockResolvedValue({

ExrateList: {

Exrate: [{ $: { CurrencyCode: 'USD', Buy: '25221.00' } }],

},

});

await expect(service.getExchangeRate(dto)).rejects.toThrowError(

'Failed to fetch exchange rate: Cannot find rates for the specified currencies',

);

});

1. **Test Case: Xử lý đơn vị tiền tệ gốc là VND**
   * **Mô tả:** Kiểm tra nếu Service xử lý đúng tỷ giá khi đơn vị tiền tệ gốc là VND.
   * **Đầu vào:** fromCurrency: 'VND', toCurrency: 'USD'.
   * **Kỳ vọng:** Tỷ giá được tính chính xác.
   * **Mã nguồn:**

it('should handle VND as base currency correctly', async () => {

const dto: getExchangeRateDto = { fromCurrency: 'VND', toCurrency: 'USD' };

const mockApiResponse = `

<ExrateList>

<Exrate CurrencyCode="USD" Buy="25221.00" />

</ExrateList>

`;

(lastValueFrom as jest.Mock).mockResolvedValue({ data: mockApiResponse });

(parseStringPromise as jest.Mock).mockResolvedValue({

ExrateList: {

Exrate: [{ $: { CurrencyCode: 'USD', Buy: '25221.00' } }],

},

});

const rate = await service.getExchangeRate(dto);

expect(rate).toBeCloseTo(1 / 25221.0);

});

### File Api Test

#### I. Test Các Trường Hợp Thành Công

1. **Xử lý Upload File Thành Công**
   * **Mục tiêu:** Kiểm tra nếu Service xử lý upload tệp và trả về đúng thông tin tệp.
   * **Kỳ vọng:** Trả về tên tệp và đường dẫn chính xác.
   * **Mã:**

it('should handle file upload successfully', () => {

const mockFile: Express.Multer.File = {

fieldname: 'file',

originalname: 'example.txt',

encoding: '7bit',

mimetype: 'text/plain',

size: 1234,

destination: './uploads',

filename: 'example.txt',

path: './uploads/example.txt',

buffer: Buffer.from('test content'),

stream: new Readable(),

};

const result = service.handleFileUpload(mockFile);

expect(result).toEqual({

filename: mockFile.filename,

path: join(\_\_dirname, '..', '..', 'uploads', mockFile.filename),

});

});

1. **Tải File Thành Công**
   * **Mục tiêu:** Kiểm tra nếu Service tải tệp về thành công và gửi đúng phản hồi.
   * **Kỳ vọng:** Gửi tệp về client với tiêu đề Content-Disposition.
   * **Mã:**

it('should download file successfully', () => {

const mockRes = {

setHeader: jest.fn(),

sendFile: jest.fn(),

status: jest.fn().mockReturnThis(),

send: jest.fn(),

} as unknown as Response;

jest.spyOn(fs, 'existsSync').mockReturnValue(true);

const filename = 'example.txt';

const filePath = join(\_\_dirname, '..', '..', 'uploads', filename);

service.downloadFile(filename, mockRes);

expect(mockRes.setHeader).toHaveBeenCalledWith(

'Content-Disposition',

`attachment; filename="${filename}"`,

);

expect(mockRes.sendFile).toHaveBeenCalledWith(filePath);

});

1. **Upload Ảnh và Trả về Chuỗi Base64**
   * **Mục tiêu:** Kiểm tra nếu Service đọc nội dung ảnh và trả về chuỗi Base64 chính xác.
   * **Kỳ vọng:** Trả về chuỗi Base64 từ nội dung ảnh.
   * **Mã:**

it('should upload image and return base64 string', () => {

const mockFile: Express.Multer.File = {

fieldname: 'image',

originalname: 'example.png',

encoding: '7bit',

mimetype: 'image/png',

size: 1234,

destination: './uploads',

filename: 'example.png',

path: './uploads/example.png',

buffer: Buffer.from('test content'),

stream: new Readable(),

};

jest.spyOn(fs, 'readFileSync').mockReturnValue(Buffer.from('test content'));

const result = service.uploadImage(mockFile);

expect(result).toEqual({

base64: Buffer.from('test content').toString('base64'),

});

});

#### II. Test Các Trường Hợp Lỗi

1. **Upload Tệp Thất Bại**
   * **Mục tiêu:** Xác minh nếu Service ném lỗi khi không nhận được tệp.
   * **Kỳ vọng:** Ném lỗi File upload failed: No file received.
   * **Mã:**

it('should throw error if file upload fails', () => {

expect(() => {

service.handleFileUpload(null);

}).toThrowError('File upload failed: No file received');

});

1. **Tải File Không Tồn Tại**
   * **Mục tiêu:** Xác minh nếu Service trả về lỗi 404 khi tệp không tồn tại.
   * **Kỳ vọng:** Trả về phản hồi 404 với thông báo File not found.
   * **Mã:**

it('should return 404 if file is not found during download', () => {

const mockRes = {

setHeader: jest.fn(),

sendFile: jest.fn(),

status: jest.fn().mockReturnThis(),

send: jest.fn(),

} as unknown as Response;

jest.spyOn(fs, 'existsSync').mockReturnValue(false);

const filename = 'nonexistent.txt';

service.downloadFile(filename, mockRes);

expect(mockRes.status).toHaveBeenCalledWith(404);

expect(mockRes.send).toHaveBeenCalledWith('File not found');

});

1. **Upload Ảnh Thất Bại**
   * **Mục tiêu:** Xác minh nếu Service ném lỗi khi không nhận được ảnh.
   * **Kỳ vọng:** Ném lỗi File upload failed: No file received.
   * **Mã:**

it('should throw error if image upload fails', () => {

expect(() => {

service.uploadImage(null);

}).toThrowError('File upload failed: No file received');

});

#### III. Mô Phỏng Thư Viện fs

* **Mock Thư Viện fs:**
  + Mock các phương thức existsSync, readFileSync, readdirSync để kiểm soát kết quả trả về.
  + Ví dụ:

jest.spyOn(fs, 'existsSync').mockReturnValue(true);

jest.spyOn(fs, 'readFileSync').mockReturnValue(Buffer.from('test content'));

## Kết quả kiểm thử

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, số

Mô tả được tạo tự động

Hình 1 Minh họa kết quả chạy các trường hợp kiểm thử

**1. Tổng Quan Kết Quả**

* **Tổng số bộ test (Test Suites):** 7
* **Tổng số test case:** 36
* **Kết quả:**
  + **Passed:** 36/36
  + **Failed:** 0/36
* **Thời gian chạy:** 13.133 giây

**2. Kết Quả Chi Tiết Từng Module**

**FileService**

* **Mô tả:** Module này chịu trách nhiệm xử lý upload tệp, tải tệp, liệt kê danh sách tệp đã upload, và xử lý ảnh thành chuỗi Base64.
* **Kết quả:**
  + **Passed Test Cases:**
    1. Xử lý upload tệp thành công.
    2. Ném lỗi nếu không nhận được tệp khi upload.
    3. Tải tệp về thành công.
    4. Ném lỗi 404 nếu tệp không tồn tại khi tải về.
    5. Xử lý ảnh và trả về chuỗi Base64 thành công.
    6. Ném lỗi nếu upload ảnh thất bại.
  + **Thời gian:** 18.252 giây.

**QrCodeService**

* **Mô tả:** Module này tạo mã QR từ dữ liệu văn bản được cung cấp.
* **Kết quả:**
  + **Passed Test Cases:**
    1. Tạo mã QR từ dữ liệu hợp lệ.
    2. Ném lỗi nếu mã QR không được tạo thành công.
  + **Thời gian:** 11.9 giây.

**QrCodeController**

* **Mô tả:** Xử lý các yêu cầu HTTP liên quan đến mã QR, giao tiếp với QrCodeService.
* **Kết quả:**
  + **Passed Test Cases:**
    1. Xử lý yêu cầu tạo mã QR thành công.
    2. Ném lỗi nếu dữ liệu đầu vào không hợp lệ.
  + **Thời gian:** 12.358 giây.

**GerService**

* **Mô tả:** Truy xuất tỷ giá hối đoái hiện tại thông qua API.
* **Kết quả:**
  + **Passed Test Cases:**
    1. Trả về tỷ giá chính xác cho đơn vị tiền tệ hợp lệ.
    2. Ném lỗi khi không nhận được dữ liệu từ API.
    3. Ném lỗi khi đơn vị tiền tệ không hợp lệ.
    4. Xử lý đơn vị tiền tệ gốc hoặc đích là VND.
  + **Thời gian:** 11.938 giây.

**GerController**

* **Mô tả:** Xử lý các yêu cầu HTTP liên quan đến tỷ giá hối đoái, giao tiếp với GerService.
* **Kết quả:**
  + **Passed Test Cases:**
    1. Trả về tỷ giá cho yêu cầu hợp lệ.
    2. Ném lỗi khi dịch vụ gặp lỗi.
  + **Thời gian:** 12.349 giây.

**HistoricalExchangeRateService**

* **Mô tả:** Truy xuất tỷ giá hối đoái lịch sử từ API dựa trên ngày cụ thể.
* **Kết quả:**
  + **Passed Test Cases:**
    1. Trả về tỷ giá chính xác cho ngày hợp lệ.
    2. Ném lỗi khi ngày không hợp lệ hoặc không được cung cấp.
    3. Ném lỗi khi ngày là tương lai.
    4. Ném lỗi khi không có dữ liệu từ API.
  + **Thời gian:** 11.945 giây.

**3. Console Output Quan Trọng**

* **FileService:**
  + Tệp tải về không tồn tại:

mathematica

File not found: C:\...\uploads\nonexistent.txt

* + Ảnh lưu thành công:

mathematica

Image saved at: C:\...\uploads\example.png

* **GerService:**
  + Không nhận được dữ liệu từ API:

javascript

Error fetching exchange rate: No response data received from API

* + Không tìm thấy đơn vị tiền tệ:

arduino

Error fetching exchange rate: Cannot find rates for the specified currencies

**4. Nhận Xét**

* **Điểm mạnh:**
  + Tất cả các module đều vượt qua các bài kiểm thử.
  + Xử lý lỗi và ngoại lệ tốt, đảm bảo các thông báo lỗi rõ ràng.
* **Điểm cần cải thiện:**
  + Cần giảm số lần log thông tin thừa từ console trong các trường hợp thành công.