TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN KẾT THÚC HỌC PHẦN

CÔNG NGHỆ PHẦM MỀM

(MSHP: 220055)

TÊN ĐỀ TÀI

XÂY DỰNG ỨNG DỤNG WEB QUẢN LÝ CHI TIÊU CÁ NHÂN VỚI JAVA SPRING BOOT

Sinh viên thực hiện:

110122068	Võ Chí Hải	DA22TTD
110122105	Nguyễn Đỗ Thành Lộc	DA22TTD
110122099	Hoàng Tuấn Kiệt	DA22TTD

Giáo viên hướng dẫn: TS. Nguyễn Bảo Ân

Trà Vinh, tháng 6 năm 2025

TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN KẾT THÚC HỌC PHẦN

CÔNG NGHỆ PHẦM MỀM

(MSHP: 220055)

TÊN ĐỀ TÀI

XÂY DỰNG ỨNG DỤNG WEB QUẢN LÝ CHI TIÊU CÁ NHÂN VỚI JAVA SPRING BOOT

Sinh viên thực hiện:

110122068	Võ Chí Hải	DA22TTD
110122105	Nguyễn Đỗ Thành Lộc	DA22TTD
110122099	Hoàng Tuấn Kiệt	DA22TTD

Giáo viên hướng dẫn: TS. Nguyễn Bảo Ân

Trà Vinh, tháng 6 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày ... tháng ... năm 2025

Giáo viên hướng dẫn

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày ... tháng ... năm 2025

Thành viên hội đồng

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành bài báo cáo này, nhóm em xin gửi lời cảm ơn đến các Quý Thầy cô Trường kỹ thuật và công nghệ, Trường đại học Trà Vinh đã tạo cơ hội cho em được học tập, rèn luyện và tích lũy kiến thức, kỹ năng để thực hiện bài báo cáo này.

Đặc biệt, em xin gửi lời cảm ơn đến thầy Nguyễn Bảo Ân đã tận tình chỉ dẫn, theo dõi và đưa ra những lời khuyên bổ ích giúp em giải quyết được các vấn đề gặp phải trong quá trình nghiên cứu và hoàn thành đề tài một cách tốt nhất.

Do kiến thức của bản thân còn hạn chế và thiếu kinh nghiệm thực tiễn nên nội dung bài báo cáo khó tránh những thiếu sót. Nhóm chúng em rất mong nhận được sự góp ý, chỉ dạy thêm từ Quý Thầy cô.

Cuối cùng, em xin chúc Quý Thầy Cô luôn thật nhiều sức khỏe và đạt được nhiều thành công trong công việc.

MỤC LỤC

Danh mục hình ảnh và bảng biểu

CHƯƠNG 1. GIỚI THIỆU

1.1. Giới thiệu đề tài

Tên đề tài: Xây dựng ứng dụng web quản lý chi tiêu cá nhân sử dụng Spring Boot

Chủ đề: Phát triển ứng dụng hỗ trợ người dùng ghi chép, phân loại và phân tích các khoản thu – chi cá nhân thông qua giao diện web trực quan, thân thiện, áp dụng công nghệ Spring Boot.

1.2. Mục tiêu của ứng dụng

Ứng dụng hướng đến việc hỗ trợ người dùng:

Quản lý thu nhập và chi tiêu hàng ngày.

Hiển thị báo cáo thống kê theo tuần/tháng/năm bằng biểu đồ.

Đặt giới hạn chi tiêu theo tháng và cảnh báo khi vượt hạn mức.

1.3. Lý do chọn đề tài

Trong đời sống hiện đại, việc quản lý tài chính cá nhân ngày càng trở thành một kỹ năng quan trọng, đặc biệt đối với sinh viên, người đi làm và những người trẻ tuổi mới bắt đầu xây dựng cuộc sống độc lập. Với sự phát triển của công nghệ, nhu cầu số hóa trong quản lý chi tiêu không chỉ là một xu hướng mà còn là một giải pháp thiết yếu giúp người dùng kiểm soát tốt hơn thói quen tiêu dùng, từ đó hướng tới một lối sống tiết kiệm và ổn định về tài chính.

Tuy nhiên, trên thực tế, không phải ai cũng có thể duy trì việc ghi chép thu – chi một cách đều đặn bằng các phương pháp thủ công như sổ tay hay bảng tính Excel. Những cách này tuy đơn giản nhưng thiếu tính linh hoạt, khó đồng bộ và dễ gây nhàm chán. Mặt khác, nhiều ứng dụng quản lý chi tiêu hiện có trên thị trường thường tích hợp quá nhiều tính năng phức tạp, yêu cầu người dùng phải làm quen trong thời gian dài hoặc gặp rào cản ngôn ngữ (đa phần là tiếng Anh). Điều đó khiến người dùng phổ thông dễ nản và từ bỏ việc theo dõi tài chính cá nhân.

Chính vì vậy, nhóm đã quyết định lựa chọn đề tài "Ứng dụng quản lý chi tiêu cá nhân" với mục tiêu xây dựng một hệ thống đơn giản, dễ sử dụng, giao diện thân thiện, tập trung vào những chức năng thiết yếu nhất như: ghi chép thu chi, phân loại khoản mục, xem báo cáo thống kê và cảnh báo khi vượt ngân sách. Ứng dụng được xây dựng dưới dạng website để người dùng có thể dễ dàng truy cập từ mọi thiết bị có trình duyệt, đồng thời sử dụng Spring Boot – một framework mạnh mẽ, hiện đại và phù hợp với mô hình phát triển web chuyên nghiệp, giúp đảm bảo hiệu suất của hệ thống.

Bên cạnh đó, việc lựa chọn đề tài này còn xuất phát từ mong muốn củng cố và nâng cao kỹ năng lập trình web backend bằng Java – một trong những ngôn ngữ phổ biến nhất hiện nay trong lĩnh vực phát triển phần mềm doanh nghiệp. Việc áp dụng Spring Boot không chỉ giúp nhóm hiểu rõ hơn về cấu trúc và luồng xử lý của một ứng dụng web, mà còn rèn luyện được nhiều kỹ năng quan trọng như thiết kế API RESTful, quản lý cơ sở dữ liệu quan hệ, triển khai ứng dụng thực tế, cũng như xử lý các vấn đề liên quan đến bảo mật và xác thực người dùng.

CHƯƠNG 2. PHÂN TÍCH YÊU CẦU

2.1. Các chức năng chính của hệ thống

2.1.1. Quản lý người dùng

Đăng ký tài khoản: người dùng tạo tài khoản mới với các thông tin bao gồm: tên người dùng và mật khẩu.

Đăng nhập/đăng xuất: xác thực người dùng vào hệ thống.

Bảo mật thông tin người dùng bằng cách mã hóa mật khẩu (dùng BCrypt).

2.1.2. Ghi chép thu – chi

Giao diện đơn giản để nhập thông tin thu nhập hoặc khoản chi tiêu.

Các trường thông tin bao gồm: tên giao dịch (thu/chi), số tiền, ngày thực hiện.

Cho phép chỉnh sửa hoặc xóa giao dịch đã tạo

2.1.3. Xem thống kê và báo cáo tài chính

Thống kê tổng thu – chi theo tuần, tháng, năm.

Hiển thị số dư hiện tại của người dùng.

Biểu đồ hình tròn và cột thể hiện tỷ lệ các khoản chi theo danh mục.

Biểu đồ đường hiển thị biến động số dư theo thời gian.

2.2. Các yêu cầu phi chức năng

Bên cạnh các chức năng chính, hệ thống cũng cần đáp ứng các yêu cầu phi chức năng nhằm đảm bảo trải nghiệm người dùng và chất lượng phần mềm.

2.2.1.Hiệu năng

Hệ thống phải phản hồi nhanh (dưới 2 giây) cho các thao tác như ghi giao dịch, tìm kiếm, thống kê.

Cơ sở dữ liệu được tối ưu bằng chỉ mục (index) cho truy vấn nhanh.

2.2.2. Bảo mật

Mã hóa mật khẩu người dùng khi lưu vào cơ sở dữ liệu.

Sử dụng Spring Security để bảo vệ tài nguyên backend.

Kiểm tra dữ liệu đầu vào để tránh tấn công SQL Injection, XSS.

2.2.3. Tính mở rộng

Hệ thống được thiết kế theo mô hình MVC kết hợp với kiến trúc RESTful API, trong đó backend (sử dụng Spring Boot). Kiến trúc này giúp dễ dàng tách rời giữa phần xử lý dữ liệu và phần hiển thị giao diện, từ đó nâng cao tính linh hoạt, khả năng bảo trì và khả năng phát triển đa nền tảng.

2.2.4. Khả năng bảo trì

Viết mã sạch, dễ hiểu, có tài liệu chú thích rõ ràng.

Cấu trúc thư mục rõ ràng theo chuẩn Spring Boot: controller, service, repository, model, config.

Dễ nâng cấp, kiểm thử và chỉnh sửa khi cần thiết.

2.2.5.Tính thân thiện với người dùng

Giao diện đơn giản, dễ sử dụng với người không am hiểu công nghệ.

Thiết kế responsive, hiển thị tốt trên cả máy tính và điện thoại.

Sử dụng màu sắc, biểu tượng và biểu đồ hợp lý để minh họa thông tin chi tiêu.

CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể hệ thống

Hệ thống được xây dựng theo kiến trúc Monolithic, trong đó toàn bộ giao diện người dùng, xử lý nghiệp vụ và truy cập dữ liệu được tích hợp trong cùng một ứng dụng duy nhất. Kiến trúc này đơn giản trong triển khai và vận hành, phù hợp với các hệ thống có quy mô vừa và nhỏ, giúp tối ưu hiệu suất khi triển khai ban đầu và dễ dàng kiểm soát logic hệ thống trong một khối thống nhất.

Các thành phần chính bao gồm:

Frontend:

Giao diện người dùng được xây dựng bằng Thymeleaf – một template engine chạy phía server, cho phép render HTML động dựa trên dữ liệu trả về từ backend. Với Thymeleaf, các trang HTML được sinh ra trực tiếp từ server, giúp đồng bộ giữa giao diện và logic xử lý, đồng thời giảm độ phức tạp khi phát triển giao diện.

Để thiết kế giao diện, hệ thống sử dụng Tailwind CSS, một framework CSS hiện đại theo hướng utility-first. Tailwind hỗ trợ lập trình viên dễ dàng tạo nên các giao diện thân thiện, nhất quán, phù hợp với nhiều độ phân giải khác nhau mà không cần viết CSS thủ công quá nhiều. Giao diện hệ thống đạt tính responsive, cho phép hoạt động tốt trên cả máy tính và thiết bị di động.

Backend:

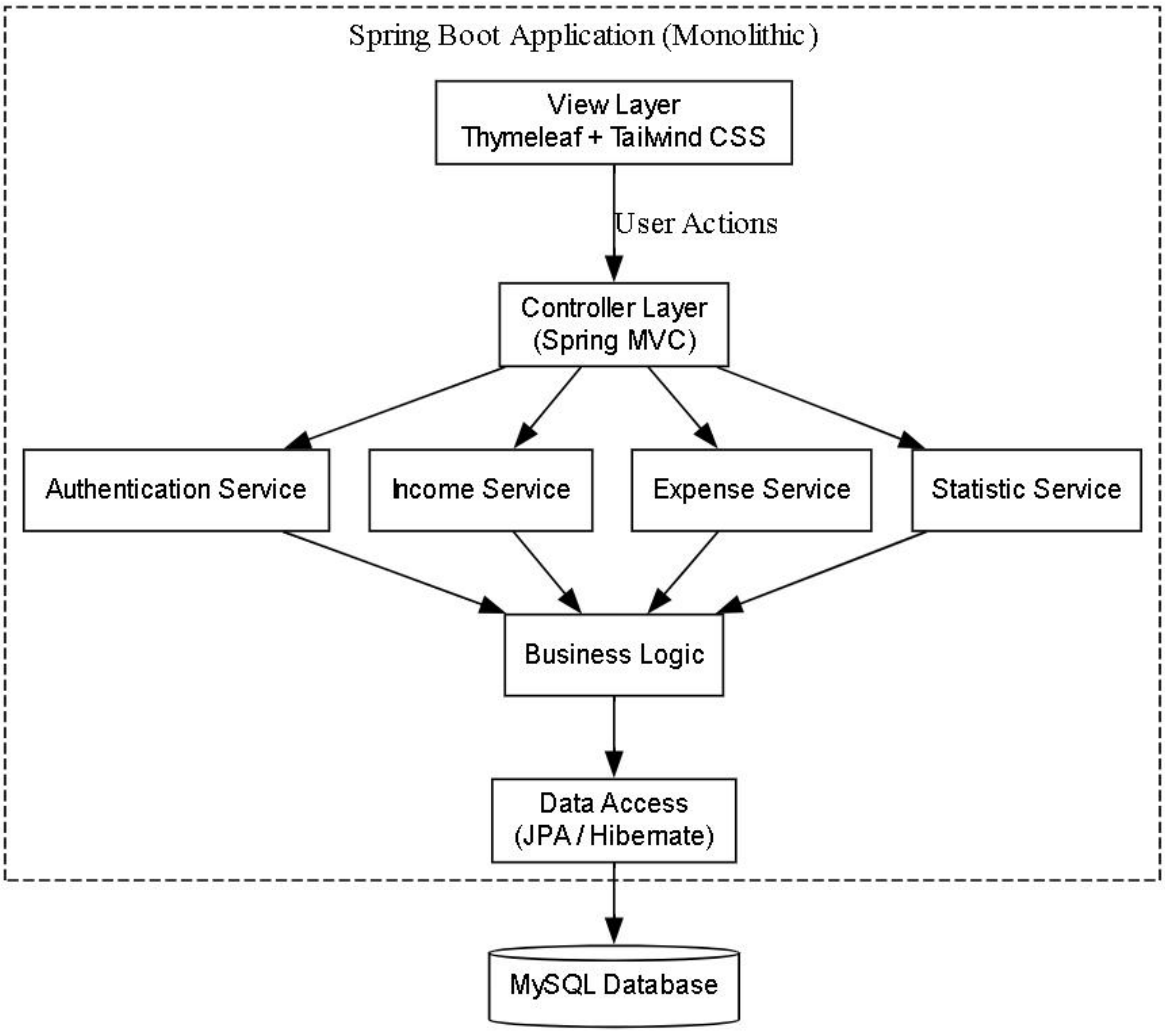
Phần backend được xây dựng với Spring Boot, một framework mạnh mẽ thuộc hệ sinh thái Java. Backend chịu trách nhiệm xử lý logic nghiệp vụ, thực hiện các chức năng như xác thực người dùng, quản lý phiên làm việc, tiếp nhận các yêu cầu từ giao diện người dùng và xử lý dữ liệu tương ứng.

Toàn bộ quy trình xử lý dữ liệu, tính toán, phân tích và phản hồi kết quả đều diễn ra ở backend. Backend và frontend được tích hợp chặt chẽ trong cùng một ứng dụng Monolithic, giúp giảm độ trễ và tăng độ đồng bộ giữa hai phần này.

Database:

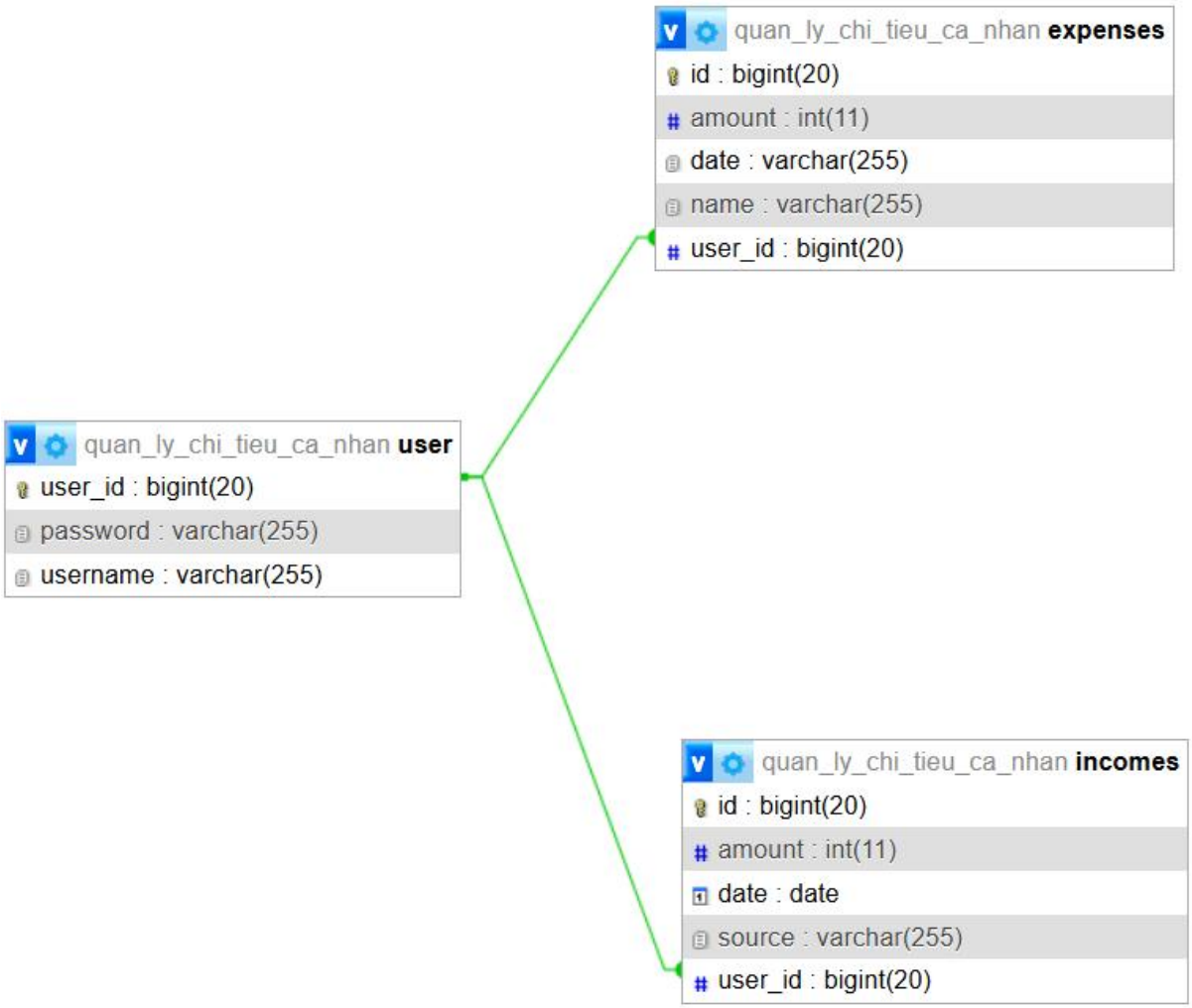
Hệ thống sử dụng MySQL làm hệ quản trị cơ sở dữ liệu chính. MySQL lưu trữ toàn bộ thông tin người dùng, dữ liệu thu chi cá nhân, danh mục các loại chi tiêu, cũng như các thiết lập cấu hình hệ thống.

3.2. Sơ Đồ Kiến Trúc Hệ Thống



Hình 1 Sơ đồ kiến trúc của hệ thống

3.3. Sơ đồ quan hệ dữ liệu



HÌnh 2 Sơ đồ quan hệ dữ liệu

3.4. Thiết kế API swagger mô tả các endpoint cấu trúc request respone

Endpoint	Phương thức	Mô tả	Yêu cầu (Request)	Phản hồi (Response)
/api/users/register	POST	Đăng ký người dùng bằng username và password	{ “username”: “user1”, “password”: “123”}	- **200 OK**: Đăng ký thành công. -
/api/users	GET	Hiển thị tất cả người dùng	- Không có request body.	- **200 OK**: Danh sách tất cả người dùng
/api/users/ {username}	GET	Hiển thị người dùng theo username	-Ko có request body	- **200 OK**: Người dùng theo username **-404 NOT FOUND:** Ko tìm thấy

Bảng 1 API user

Endpoint	**Phương thức**	**Mô tả**	**.Yêu cầu (Request)**	**Phản hồi (Response)**
/api/expenses	GET	Xem tất cả các khoản chi tiêu của người dùng	Không có request body.	**200 OK**: Thành công
/api/expenses/{id}	GET	Xem các khoản chi tiêu theo id	Không có request body.	**- 200 OK**: Thành công **-404 NOT FOUND:** Ko tìm thấy
/api/expenses/monthly	GET	Xem tổng các khoản chi tiêu theo từng tháng	Không có request body.	**- 200 OK**: Thành công
/api/expenses	POST	Tạo mới một chi tiêu	{ “expense”: “tiền điện”, “amount”: ”250000”, “date”: ”2025-06-10” }	**- 200 OK**: Thành công
/api/expenses	DELETE	Xóa một khoản chi tiêu	Không có request body	**- 200 OK**: Thành công

Bảng 2 API expense

Endpoint	**Phương thức**	**Mô tả**	**.Yêu cầu (Request)**	**Phản hồi (Response)**
/api/incomes	GET	Xem tất cả các khoản thu nhập	Không có request body.	**200 OK**: Thành công
/api/expenses/ {id}	GET	Xem các khoản thu nhập theo id	Không có request body.	**- 200 OK**: Thành công **-404 NOT FOUND:** Ko tìm thấy
/api/incomes	POST	Tạo mới một thu nhập	{ “source”: “tiền lương”, “amount”: ”7000000”, “date”: ”2025-06-30” }	**- 200 OK**: Thành công
/api/incomes	DELETE	Xóa một khoản thu nhập	Không có request body	**- 200 OK**: Thành công **-404 NOT FOUND:** Ko tìm thấy

Bảng 3 Api incom

3.5. Thiết kế giao diện (UI/UX)

3.5.1. Giao diện trang đăng nhập

Hình 3 Trang đăng nhập

3.5.2. Giao diện trang đăng ký

Hình 4 Trang đăng ký

3.5.3. Giao diện trang chủ

Hình 5 Trang chủ

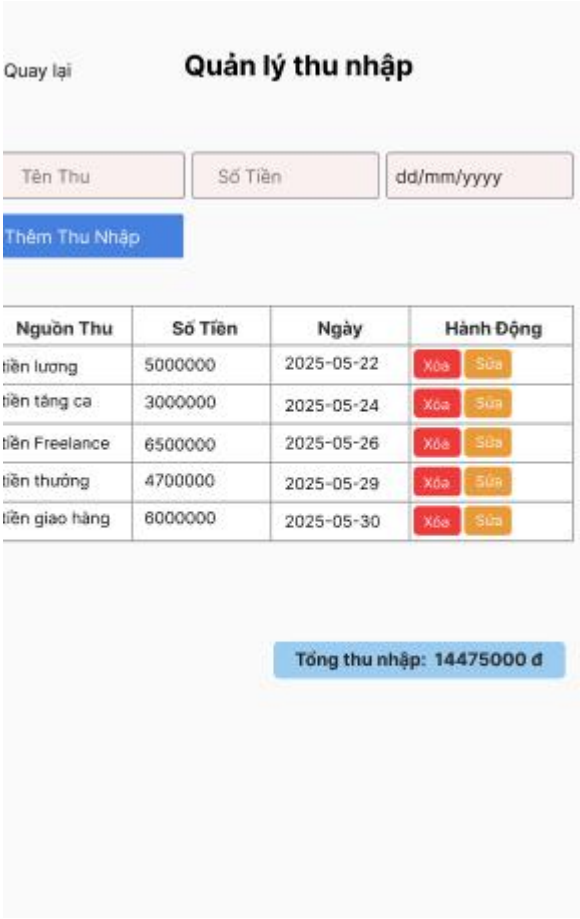
3.5.4. Giao diện menu

Hình 6 Menu

3.5.5 . Giao diện trang quản lý chi tiêu

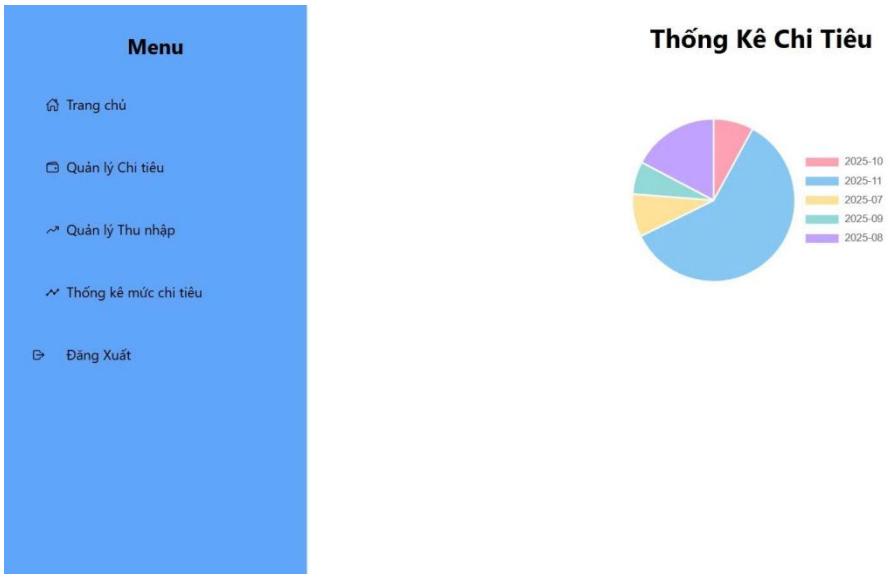
Hình 7 Trang quản lý chi tiêu

3.5.6. Giao diện trang quản lý thu nhập



Hình 8 Trang quản lý thu nhập

3.5.7. Trang thống kê chi tiêu



Hình 9 Trang thống kê chi tiêu

CHƯƠNG 4. TRIỂN KHAI CÁC CÔNG CỤ SỬ DỤNG

4.1. Danh sách các công cụ sử dụng trong dự án

4.1.1. Spring Boot

Mô tả: Spring Boot là một framework Java mạnh mẽ, giúp xây dựng các ứng dụng web một cách nhanh chóng và dễ dàng. Dự án sử dụng Spring Boot để xây dựng API backend, xử lý logic nghiệp vụ, và tương tác với cơ sở dữ liệu.

Vai trò trong dự án: Xử lý các yêu cầu từ phía người dùng, quản lý dữ liệu chi tiêu, và cung cấp API cho các tính năng của ứng dụng.

4.1.2. Tailwind CSS

Mô tả: Tailwind CSS là một framework CSS tiện lợi cho phép xây dựng giao diện người dùng (UI) theo cách linh hoạt và tùy chỉnh.

Vai trò trong dự án: Dùng để thiết kế giao diện người dùng cho ứng dụng, tạo các thành phần UI dễ dàng và đẹp mắt mà không cần viết quá nhiều CSS thủ công.

4.1.3. MySQL

Mô tả: MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở.

Vai trò trong dự án: Lưu trữ dữ liệu chi tiêu cá nhân, thông tin người dùng, và các thông tin liên quan đến các giao dịch tài chính.

4.1.4. Postman

Mô tả: Postman là một công cụ phát triển API phổ biến, dùng để kiểm thử các API và gửi yêu cầu HTTP.

Vai trò trong dự án: Sử dụng để kiểm thử API backend của ứng dụng, đảm bảo rằng các yêu cầu được thực thi chính xác và trả về kết quả đúng đắn

4.1.5. Docker

Mô tả: Docker là một nền tảng để phát triển, vận hành và phân phối các ứng dụng trong các container.

Vai trò trong dự án: Sử dụng Docker để đóng gói ứng dụng và các môi trường phát triển thành các container độc lập, giúp triển khai ứng dụng dễ dàng và đồng nhất trên các môi trường khác nhau.

4.1.6. Swagger

Mô tả: Swagger là một công cụ tạo tài liệu API tự động cho ứng dụng web.

Vai trò trong dự án: Tạo và duy trì tài liệu API cho backend, giúp các nhà phát triển khác hiểu rõ hơn về các endpoints và cách sử dụng chúng.

4.1.7. GitHub

Mô tả: GitHub là một nền tảng quản lý mã nguồn, hỗ trợ công cụ quản lý phiên bản Git.

Vai trò trong dự án: Quản lý mã nguồn của dự án, đồng bộ hóa công việc giữa các thành viên trong nhóm và theo dõi lịch sử thay đổi của mã nguồn.

4.1.8. JIRA

Mô tả: JIRA là công cụ quản lý dự án và theo dõi vấn đề của Atlassian.

Vai trò trong dự án: Quản lý và theo dõi tiến độ phát triển, các lỗi phần mềm, và các công việc trong suốt vòng đời của dự án.

4.1.9. GitHub Actions

Mô tả: GitHub Actions là một công cụ CI/CD tích hợp trong GitHub, cho phép tự động hóa các quy trình phát triển phần mềm.

Vai trò trong dự án: Cấu hình quy trình CI/CD để tự động hóa việc build, kiểm thử và triển khai ứng dụng mỗi khi có thay đổi mã nguồn.

4.2. Quy trình CI/CD với GitHub Actions

Quy trình CI/CD của dự án sử dụng GitHub Actions nhằm tự động hóa quá trình xây dựng, kiểm thử và triển khai ứng dụng:

4.2.1. CI (Continuous Integration):

Khi có thay đổi mã nguồn được đẩy lên repository trên GitHub, GitHub Actions sẽ tự động thực hiện các bước sau:

Xây dựng lại ứng dụng.

Kiểm thử mã nguồn (unit test, integration test).

Kiểm tra mã nguồn với các công cụ như SonarQube.

4.2.2. CD (Continuous Deployment):

Sau khi quá trình kiểm thử thành công, GitHub Actions sẽ tự động triển khai ứng dụng vào môi trường sản xuất hoặc staging. Quá trình này được thực hiện thông qua việc chạy các script Docker để xây dựng và triển khai ứng dụng.

4.3. Cấu hình Docker và quy trình triển khai ứng dụng

4.3.1. Cấu hình Docker:

Dockerfile: Một tệp Dockerfile đã được sử dụng để xây dựng container cho ứng dụng Spring Boot. Tệp này chứa các lệnh để tạo môi trường phù hợp, cài đặt các phụ thuộc và chạy ứng dụng trong container.

docker-compose.yml: Docker Compose được sử dụng để định nghĩa và chạy các ứng dụng Docker trong môi trường phát triển, bao gồm cả ứng dụng backend (Spring Boot) và cơ sở dữ liệu MySQL.

4.3.2. Quy trình triển khai ứng dụng:

Bước 1: Ứng dụng được đóng gói vào một Docker container.

Bước 2: Docker container được triển khai lên một môi trường staging hoặc production (có thể sử dụng các dịch vụ như AWS, DigitalOcean hoặc máy chủ riêng).

Bước 3: Sau khi triển khai, người dùng có thể truy cập ứng dụng web qua trình duyệt để thực hiện các chức năng quản lý chi tiêu cá nhân.

CHƯƠNG 5. QUẢN LÝ DỰ ÁN

5.1. Kế hoạch và tiến độ

5.1.1. Thiết lập dự án trên Jira

Tạo dự án mới với loại "Scrum software project".

Xác định backlog gồm các chức năng/feature cần triển khai.

Tạo các issue tương ứng: Story, Task, Bug, Sub-task.

5.1.2. Lập kế hoạch Sprint

Chọn các issue từ backlog để đưa vào sprint.

Ước lượng độ khó bằng Story Points.

Thiết lập thời gian Sprint (1 tuần, 2 tuần, 3 tuần).

5.1.3. Theo dõi tiến độ

Sử dụng Sprint Board để theo dõi trạng thái từng công việc: To Do,

In Progress, Done.

Kiểm tra biểu đồ Burndown Chart để đánh giá tiến độ thực tế so với kế hoạch.

5.2. Phân công nhiệm vụ của từng thành viên trong nhóm

Họ và tên	Vai trò	Nhiệm vụ
Võ Chí Hải	Scrum Master, Developer, DevOps	Quản lý Sprint, theo dõi tiến độ trên Jira Phát triển Backend và Frontend. Triển khai ứng dụng (Deploy lên Docker)
Nguyễn Đỗ Thành Lộc	Designer	Thiết kế UI/UX với Figma
Hoàng Tuấn Kiệt	Tester	Viết test case, kiểm thử chức năng

Bảng 4 Bảng phân công công việc

CHƯƠNG 6. KIỂM THỬ

6.1. Chiến lược kiểm thử

Chiến lược kiểm thử là kế hoạch tổng thể xác định cách thức tiến hành kiểm thử phần mềm nhằm đảm bảo chất lượng sản phẩm. Đối với ứng dụng web quản lý chi tiêu cá nhân, chiến lược kiểm thử được thiết kế như sau:

6.1.1 Mục tiêu

Đảm bảo các API backend hoạt động đúng logic nghiệp vụ (quản lý tài khoản, ghi nhận chi tiêu, xem báo cáo, v.v).

Phát hiện và xử lý lỗi sớm trong quá trình phát triển.

Tự động hóa kiểm thử để rút ngắn thời gian kiểm thử thủ công.

6.1.2 Phạm vi kiểm thử

Chỉ tập trung vào kiểm thử **API backend** (RESTful API).

Không kiểm thử frontend (UI) ở giai đoạn này.

Bao gồm cả các tình huống hợp lệ và không hợp lệ.

6.1.3 Loại kiểm thử áp dụng

Kiểm thử chức năng (Functional Testing): Đảm bảo các API trả về đúng kết quả theo yêu cầu.

Kiểm thử hồi quy (Regression Testing): Kiểm tra lại các chức năng cũ sau khi thêm/chỉnh sửa tính năng mới.

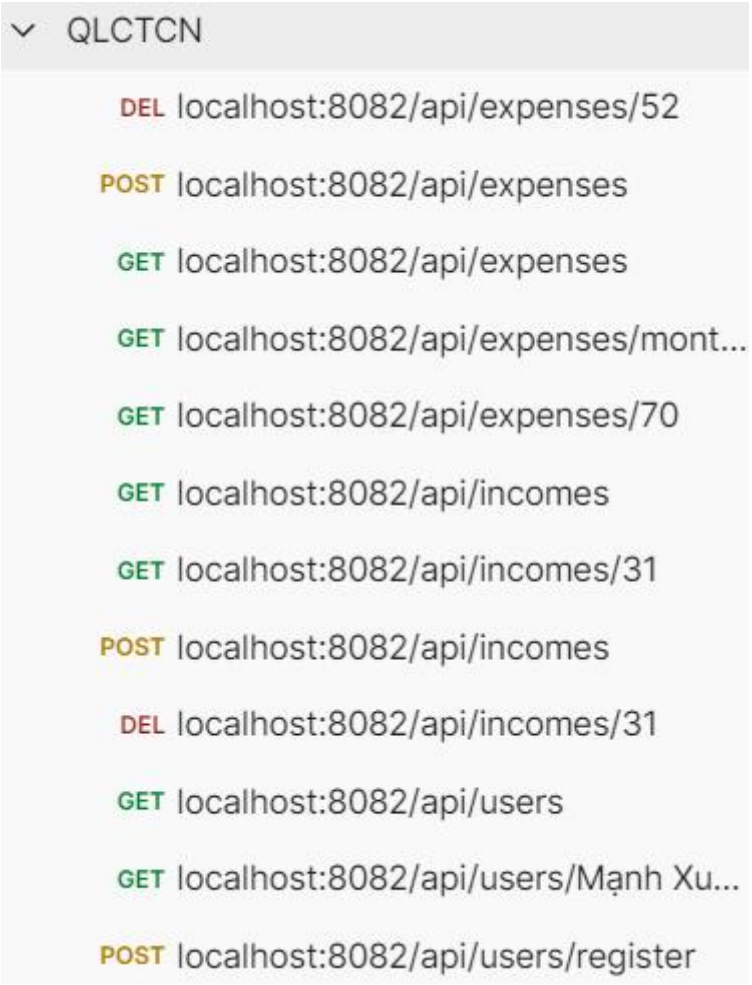
Kiểm thử tích hợp (Integration Testing): Kiểm tra luồng xử lý giữa các module (ví dụ: đăng nhập rồi mới thêm chi tiêu).

Kiểm thử tự động (Automated Testing): Tự động chạy test mỗi khi có thay đổi code bằng GitHub Actions.

6.2. Công cụ sử dụng và kết quả kiểm thử API

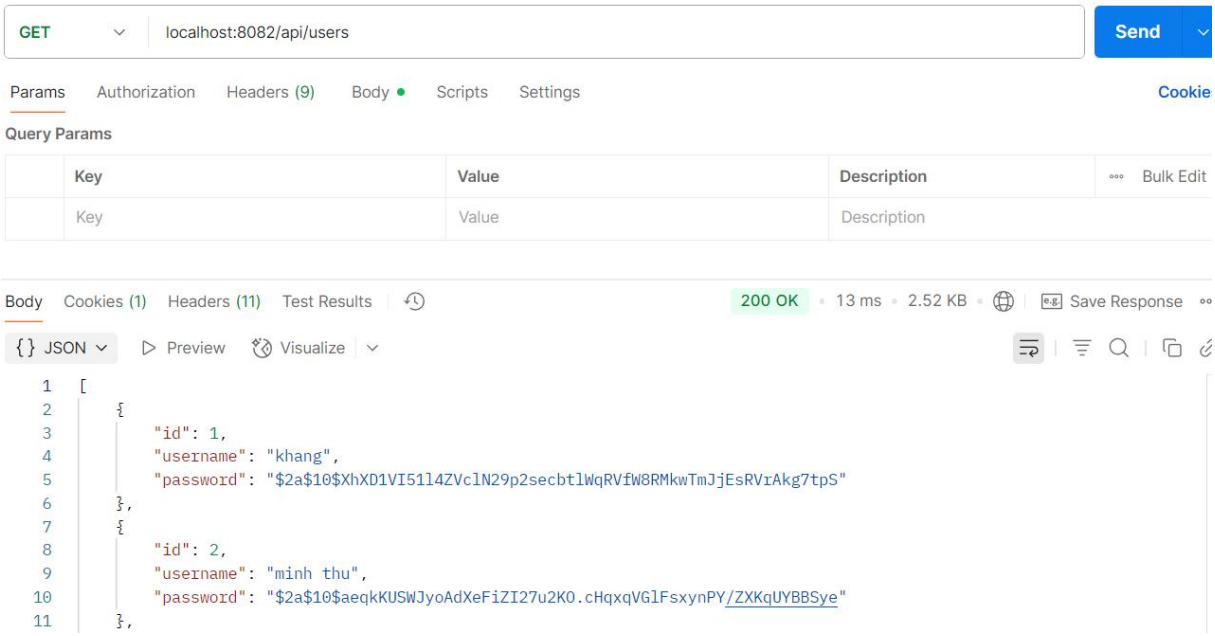
Tên công cụ	Mục đích
Postman	Tạo và chạy các request API thủ công, viết test case, kiểm tra phản hồi

Bảng 5 Công cụ kiểm thử API



Hình 10 Các endpoint trong dự án

6.2.1 kết quả kiểm thử API



Hình 11 API hiển thị tất cả user

Endpoint: http://localhost:8082/api/users

Phương thức: GET

Mục tiêu: Truy xuất danh sách tất cả người dùng đã đăng ký trong hệ thống

Kết quả trả về: Danh sách người dùng ở dạng JSON, mỗi người gồm:

id: mã người dùng

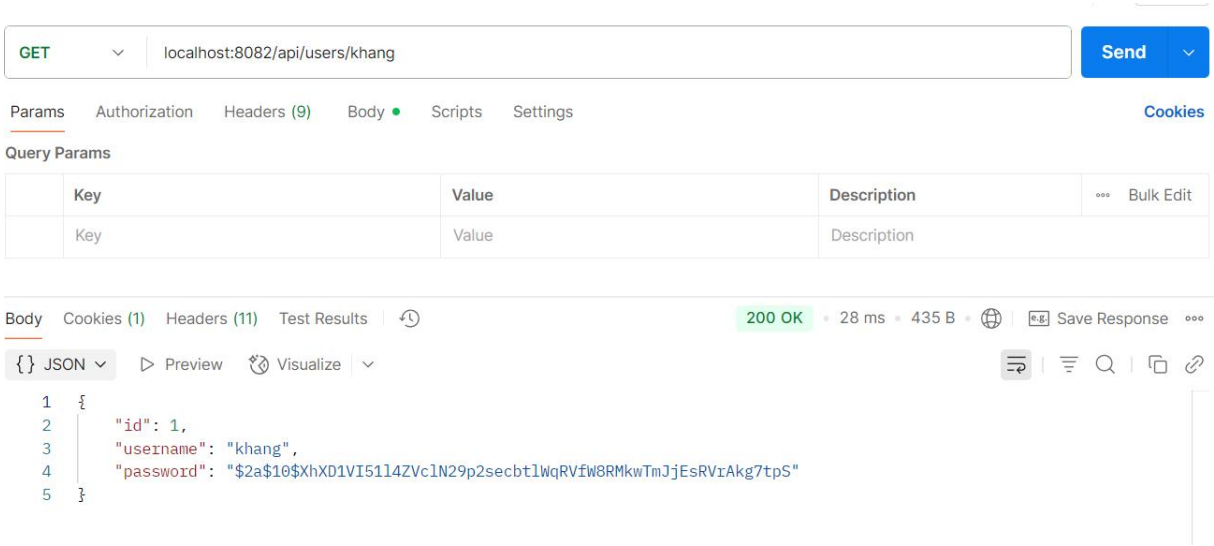
username: tên tài khoản

password: chuỗi đã mã hóa (bằng bcrypt, định dạng $2a$...)

API hoạt động đúng, trả về danh sách người dùng từ cơ sở dữ liệu.

Trường password được mã hóa bằng bcrypt → thể hiện bạn đã xử lý bảo mật tối thiểu khi lưu.

Không có xác thực hoặc kiểm tra quyền → đơn giản hóa để tập trung vào chức năng chính.



Hình 12 API Hiển thị user theo id

Endpoint: http://localhost:8082/api/users/khang

Phương thức: GET

Mục tiêu: Truy xuất thông tin chi tiết của người dùng có username là khang

Kết quả trả về: Một đối tượng JSON chứa:

id: mã người dùng (1)

username: "khang"

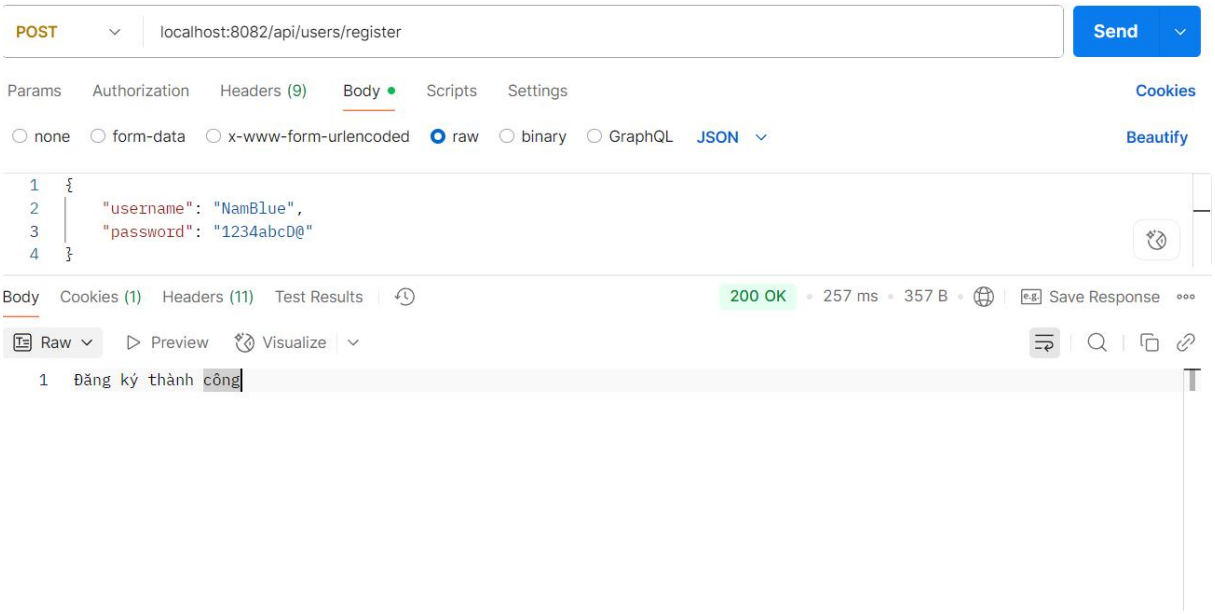
password: chuỗi mật khẩu đã được mã hóa bằng bcrypt (không phải mật khẩu gốc)

API hoạt động đúng khi trả về đúng thông tin người dùng theo tham số path username.

Trường password vẫn được mã hóa để đảm bảo không lộ mật khẩu thật.

Ứng dụng không phân quyền, nên API này không yêu cầu xác thực.

Mặc dù trả về mật khẩu mã hóa có thể không cần thiết trong ứng dụng cá nhân, nhưng vẫn đảm bảo an toàn thông tin gốc.



Hình 13 API đăng ký tài khoản

Endpoint: http://localhost:8082/api/users/register

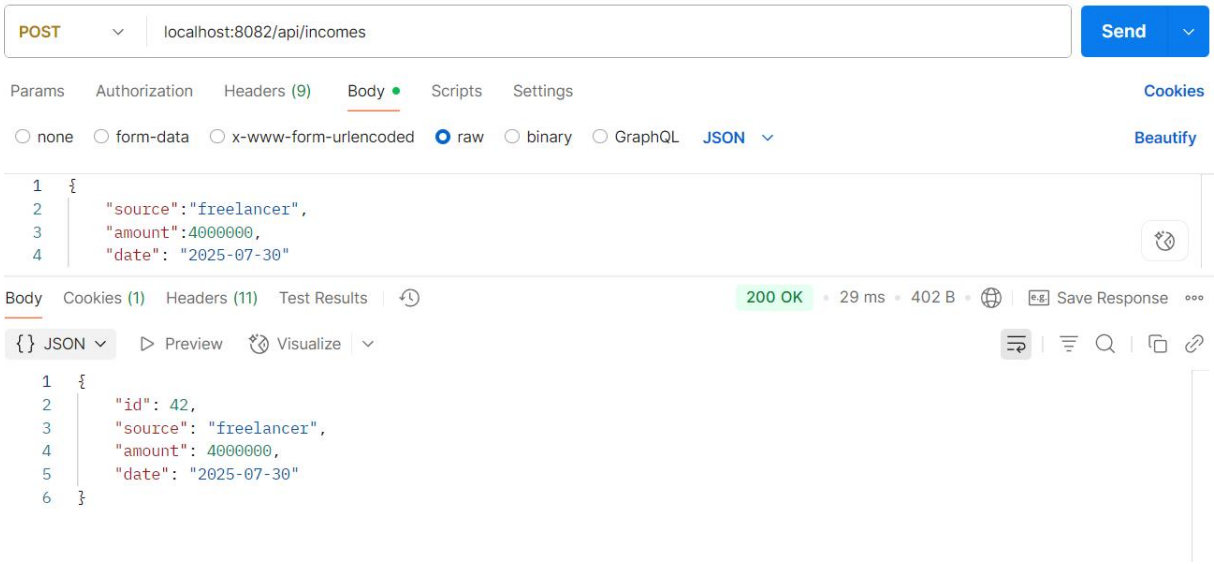
Phương thức: POST

Dữ liệu gửi (Body JSON):

Mật khẩu người dùng được xử lý mã hóa trước khi lưu, đảm bảo bảo mật dữ liệu nhạy cảm.

API không yêu cầu xác thực để đăng ký tài khoản mới.

Ứng dụng hướng đến quản lý chi tiêu cá nhân, do đó việc đăng ký tài khoản là bước đầu tiên để sử dụng các chức năng khác.



Hình 14 API tạo mới thu nhập

Endpoint: http://localhost:8082/api/incomes

Phương thức: POST

Dữ liệu gửi (Body JSON):

Mục tiêu: Thêm mới một khoản thu nhập với nguồn thu, số tiền và ngày nhận

Kết quả trả về:

Đối tượng JSON của khoản thu nhập vừa tạo, bao gồm:

id: 41 (mã định danh khoản thu nhập)

source: "freelancer"

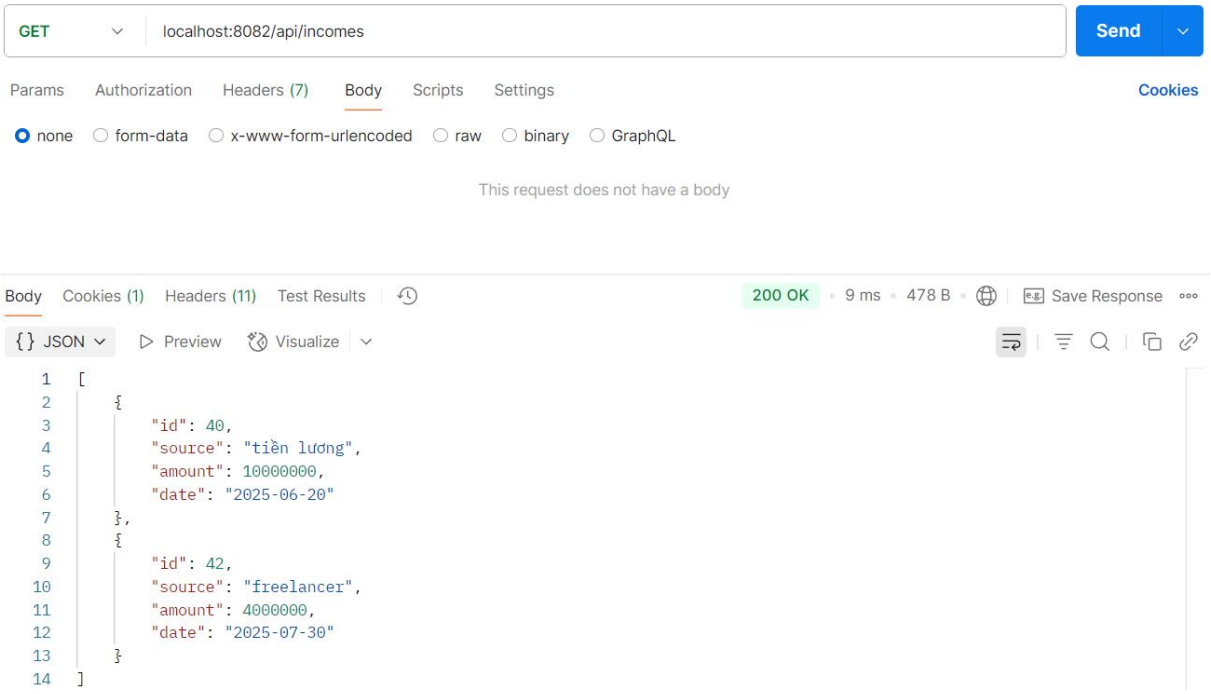
amount: 4000000

date: "2025-07-30

API thực hiện đúng chức năng thêm dữ liệu mới vào cơ sở dữ liệu.

Dữ liệu trả về cho phép xác nhận thông tin đã được lưu chính xác.

Không yêu cầu xác thực trong ứng dụng cá nhân này.



Hình 15 API hiển thị tất cả thu nhập

Endpoint: http://localhost:8082/api/incomes

Phương thức: GET

Mục tiêu: Truy xuất toàn bộ các khoản thu nhập đã được thêm vào hệ thống

Kết quả trả về:

Một mảng JSON gồm các đối tượng khoản thu nhập, mỗi khoản bao gồm:

id: mã định danh khoản thu nhập

source: nguồn thu nhập (ví dụ: “tiền lương”, “freelancer”)

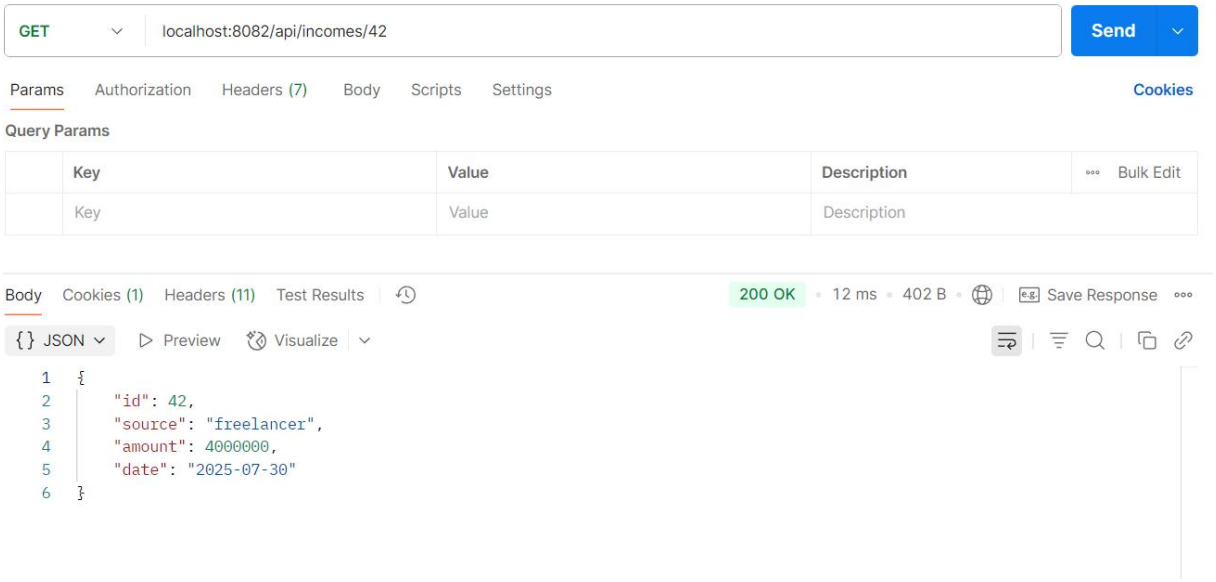
amount: số tiền thu nhập

date: ngày nhận thu nhập

API hoạt động ổn định, trả về đầy đủ danh sách khoản thu nhập cho người dùng.

Dữ liệu trả về theo đúng định dạng JSON tiêu chuẩn, dễ dàng xử lý cho phần front-end.

Ứng dụng không yêu cầu xác thực trong giai đoạn hiện tại.



Hình 16 API hiển thị thu nhập theo id

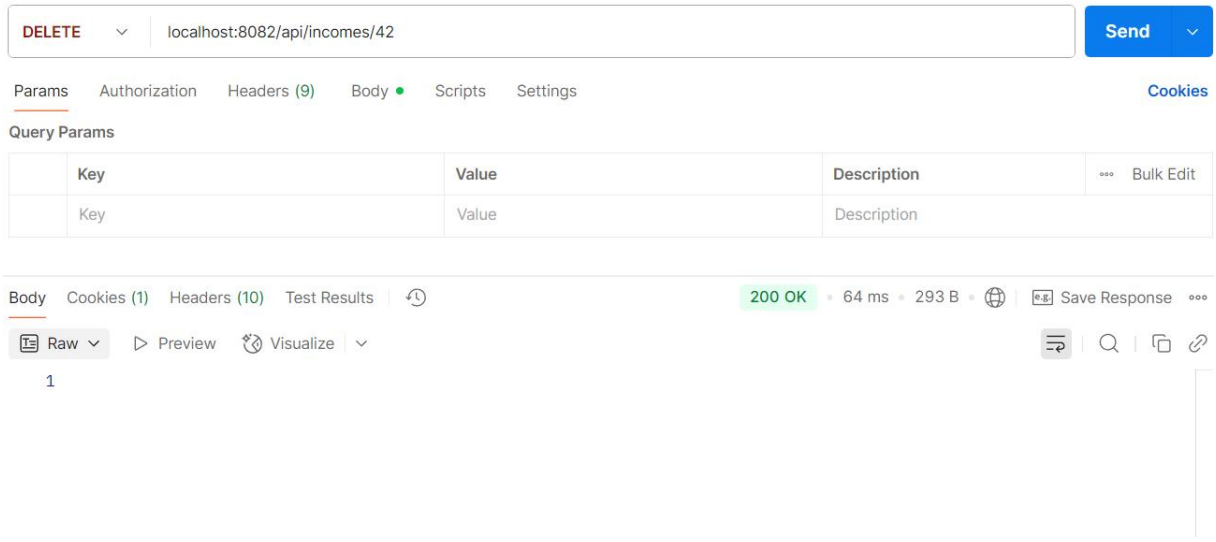
Endpoint: http://localhost:8082/api/expenses/42

Phương thức: GET

Mục tiêu: Truy vấn thông tin chi tiết của một khoản chi tiêu theo id.
với id = 78, hệ thống sẽ trả về thông tin tương ứng với khoản chi có mã số 78.

Trường hợp không tìm thấy id, hệ thống có thể trả về mã lỗi 404 Not Found.

Endpoint này thường được sử dụng trong chức năng "xem chi tiết" trên giao diện người dùng.



Hình 17 API xóa thu nhập

Endpoint: http://localhost:8082/api/incomes/42

Phương thức: DELETE

Mục tiêu: Xóa khoản thu nhập có id = 41 khỏi hệ thống

Kết quả trả về:

HTTP status: 200 OK *(hoặc* 204 No Content *nếu không trả về dữ liệu)*

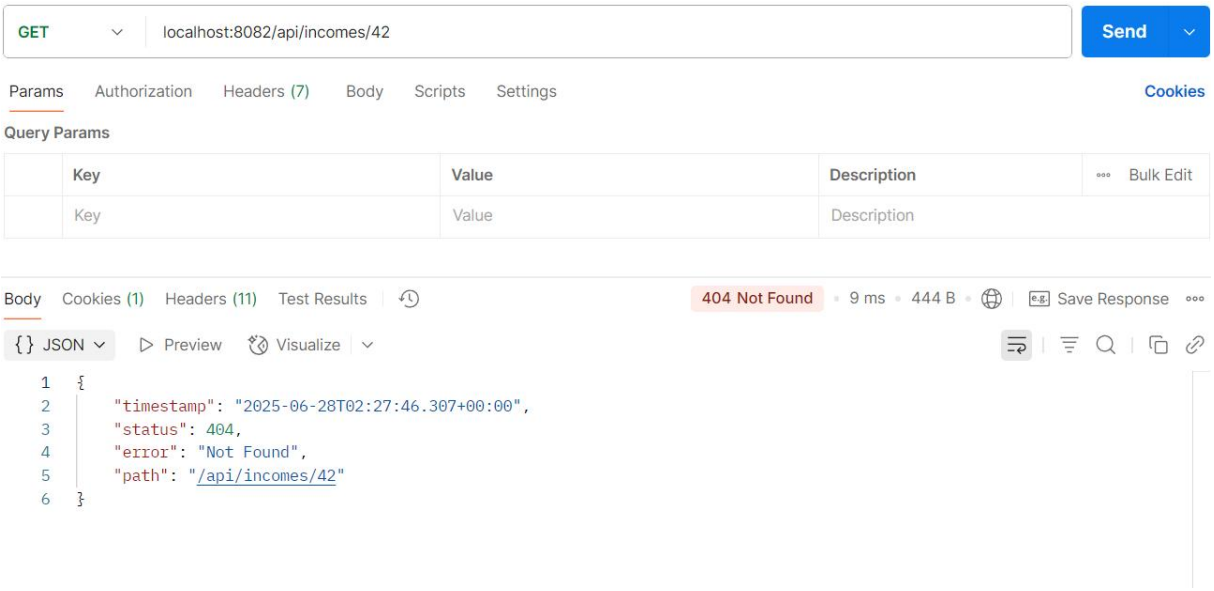
Xác nhận rằng khoản thu nhập đã bị xóa thành công khỏi cơ sở dữ liệu

API thực hiện đúng chức năng xóa dữ liệu theo ID.

Sau khi xóa, nếu thực hiện lại GET /api/incomes, khoản có id = 41 sẽ không còn xuất hiện.

Không có xác thực hoặc phân quyền — phù hợp với ứng dụng cá nhân đơn giản.

Nếu ID không tồn tại, hệ thống nên trả về HTTP 404

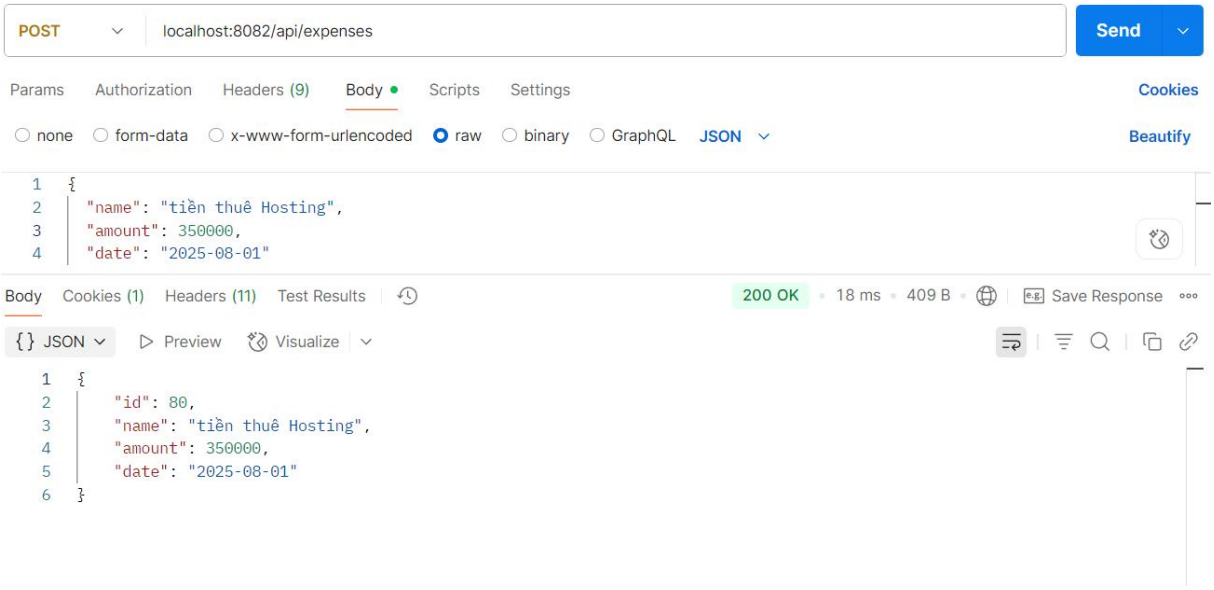


Hình 18 API thu nhập không tìm thấy

Đây là kết quả **đúng kỳ vọng**: sau khi khoản có id = 41 đã bị xóa, hệ thống không còn tìm thấy nên trả về lỗi 404.

Thông báo lỗi rõ ràng, có timestamp, mã lỗi (status = 404), loại lỗi (Not Found) và đường dẫn gây lỗi (/api/incomes/41).

Việc xử lý lỗi đúng giúp tăng độ tin cậy và dễ debug cho ứng dụng.



Hình 19 API tạo mới chi tiêu

Endpoint: http://localhost:8082/api/expenses

Phương thức: POST

Dữ liệu gửi (Body JSON):

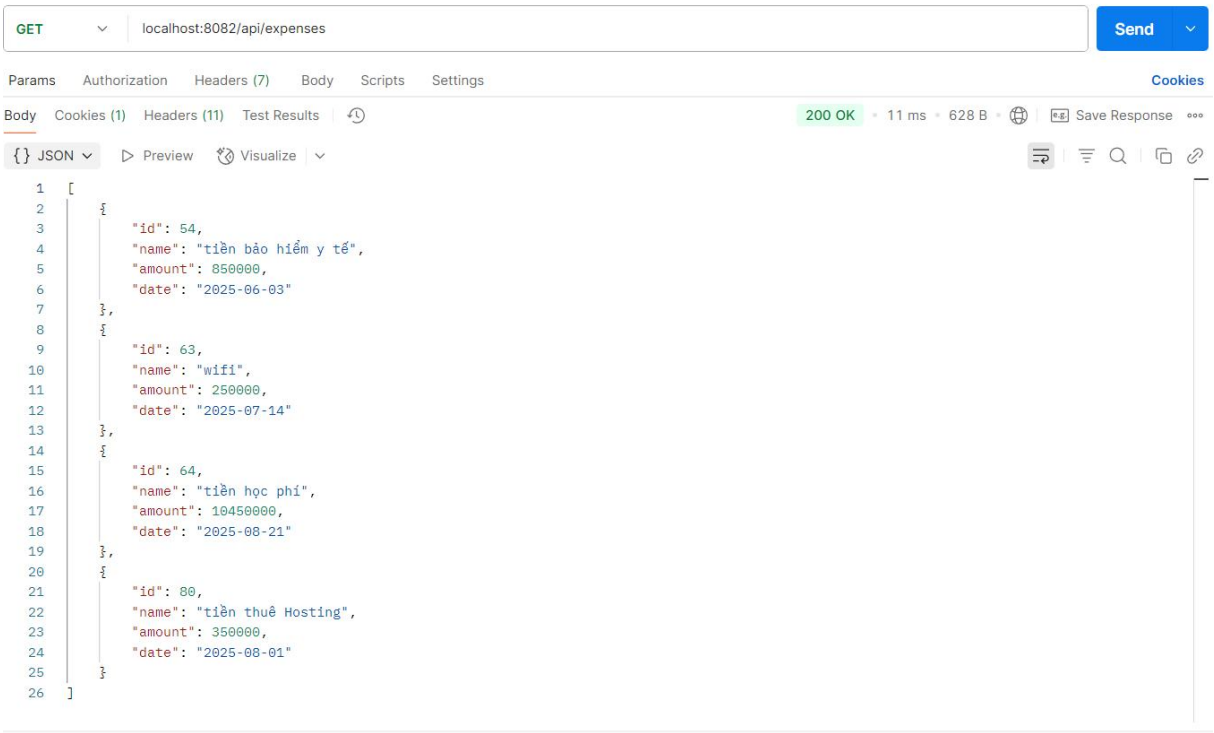
Mục tiêu: Thêm một khoản chi tiêu với nội dung, số tiền và ngày phát sinh

API hoạt động đúng, trả về thông tin khoản chi tiêu vừa được thêm vào, bao gồm id được tự động tạo.

Các trường thông tin trả về khớp với dữ liệu đã gửi.

Ngày được định dạng đúng chuẩn ISO (YYYY-MM-DD).

Ứng dụng hiện không yêu cầu xác thực, phù hợp với mục tiêu cá nhân và đơn giản hóa quy trình kiểm thử.



Hình 20 API hiển thị tất cả chi tiêu

Endpoint: http://localhost:8082/api/expenses

Phương thức: GET

Mục tiêu: Truy xuất toàn bộ các khoản chi tiêu đã được thêm vào hệ thống

Kết quả trả về: Một mảng JSON các khoản chi tiêu, mỗi khoản bao gồm:

id: mã định danh khoản chi

name: tên khoản chi (ví dụ: “tiền bảo hiểm y tế”, “wifi”)

amount: số tiền đã chi

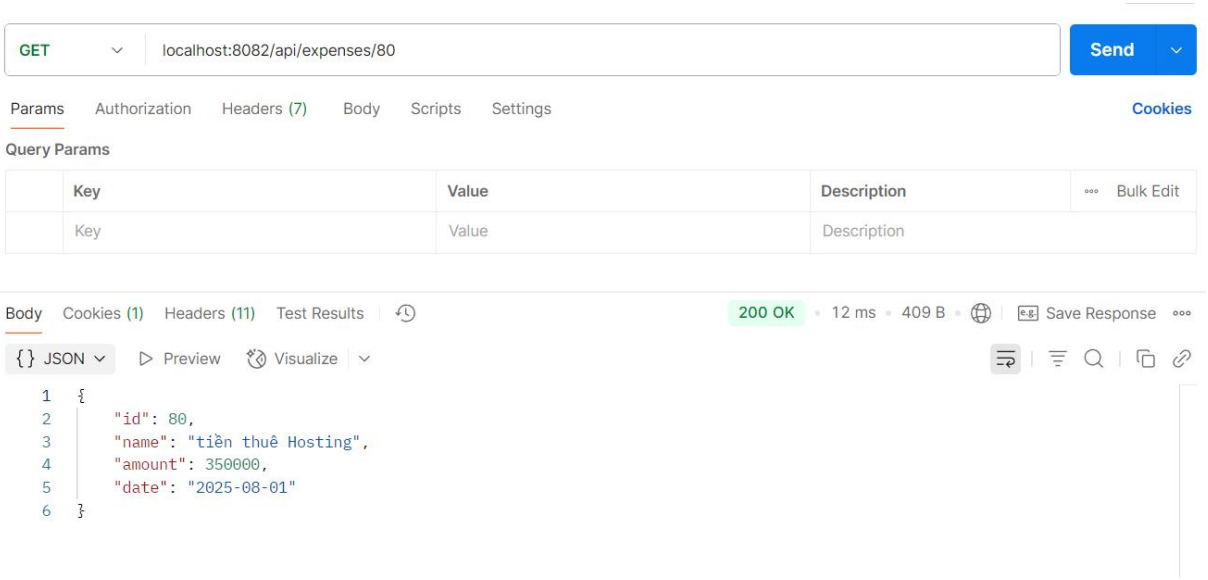
date: ngày chi (định dạng YYYY-MM-DD)

API phản hồi đúng dữ liệu đã thêm, thể hiện hệ thống lưu trữ và truy vấn chính xác.

Các khoản chi tiêu hiển thị đầy đủ thông tin, sắp xếp theo thứ tự id hoặc theo thứ tự thêm (tùy vào backend).

Dữ liệu phản hồi định dạng JSON chuẩn, thuận tiện xử lý ở phía frontend.

Không yêu cầu xác thực – phù hợp với ứng dụng đơn giản dành cho cá nhân.



Hình 21 API hiển thị chi tiêu theo id

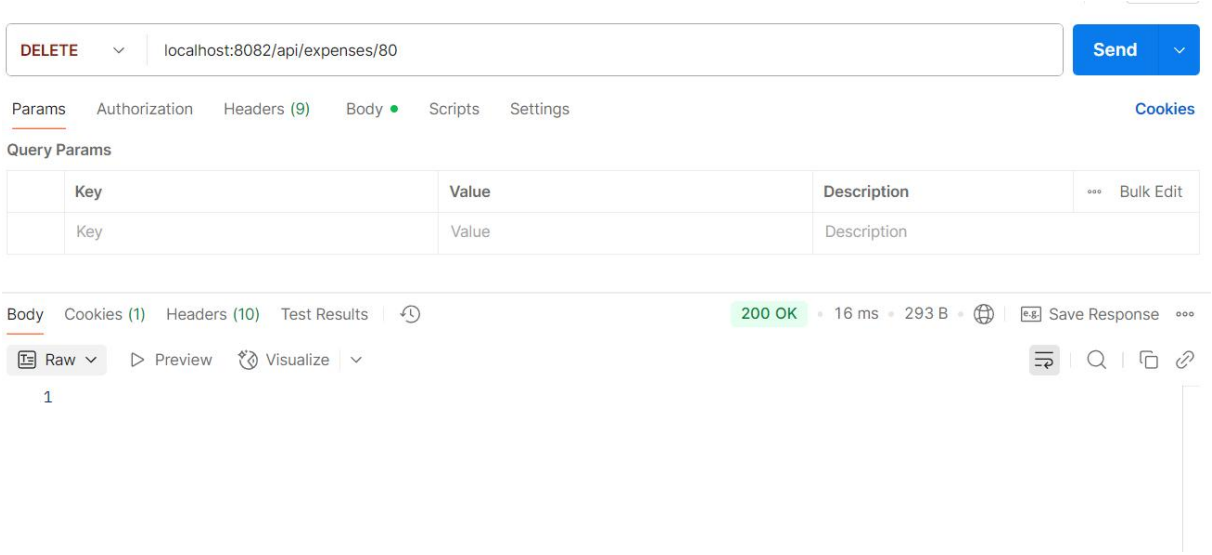
Endpoint: http://localhost:8082/api/expenses/78

Phương thức: GET

Mục tiêu: Truy vấn thông tin chi tiết của một khoản chi tiêu theo id.
với id = 78, hệ thống sẽ trả về thông tin tương ứng với khoản chi có mã số 78.

Trường hợp không tìm thấy id, hệ thống có thể trả về mã lỗi 404 Not Found.

Endpoint này thường được sử dụng trong chức năng "xem chi tiết" trên giao diện người dùng.



Hình 22 API xóa chi tiêu

Endpoint:/api/expenses/{id}

Phương thức:DELETE

Kiểu dữ liệu gửi đi:

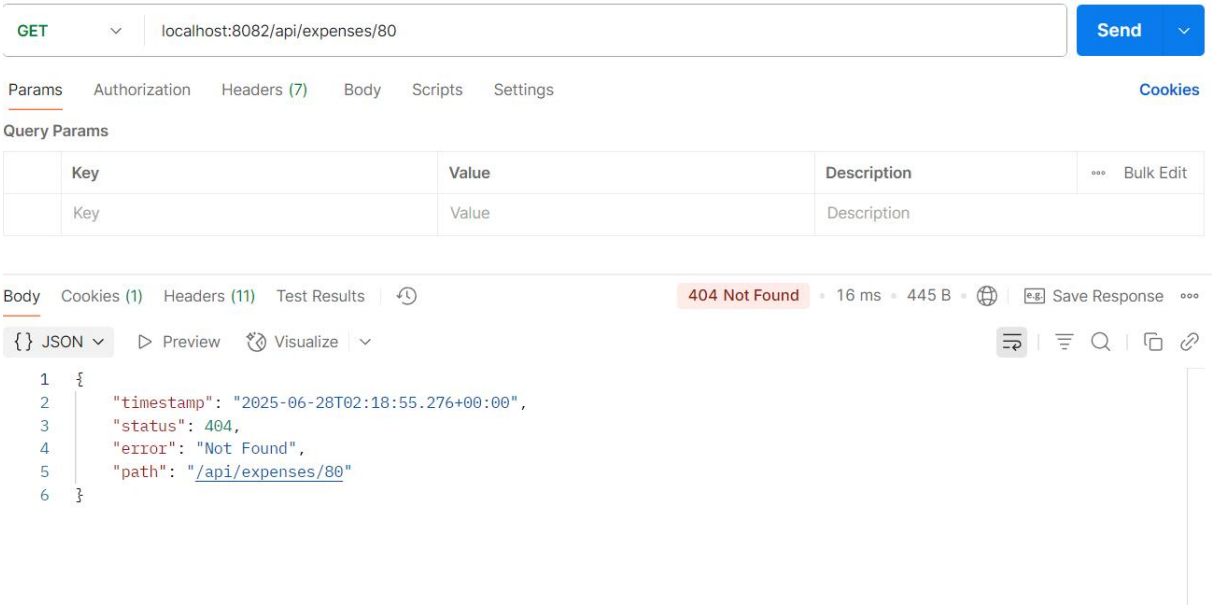
Không có body.
Tham số duy nhất là id được truyền trực tiếp trên URL để xác định khoản chi cần xoá.

Mục tiêu:

Xoá một khoản chi tiêu cụ thể theo id.
Ví dụ: Xoá khoản chi có mã số 80 khỏi cơ sở dữ liệu.

Khi người dùng xác nhận xoá trên giao diện, hệ thống sẽ gửi yêu cầu DELETE đến endpoint này.

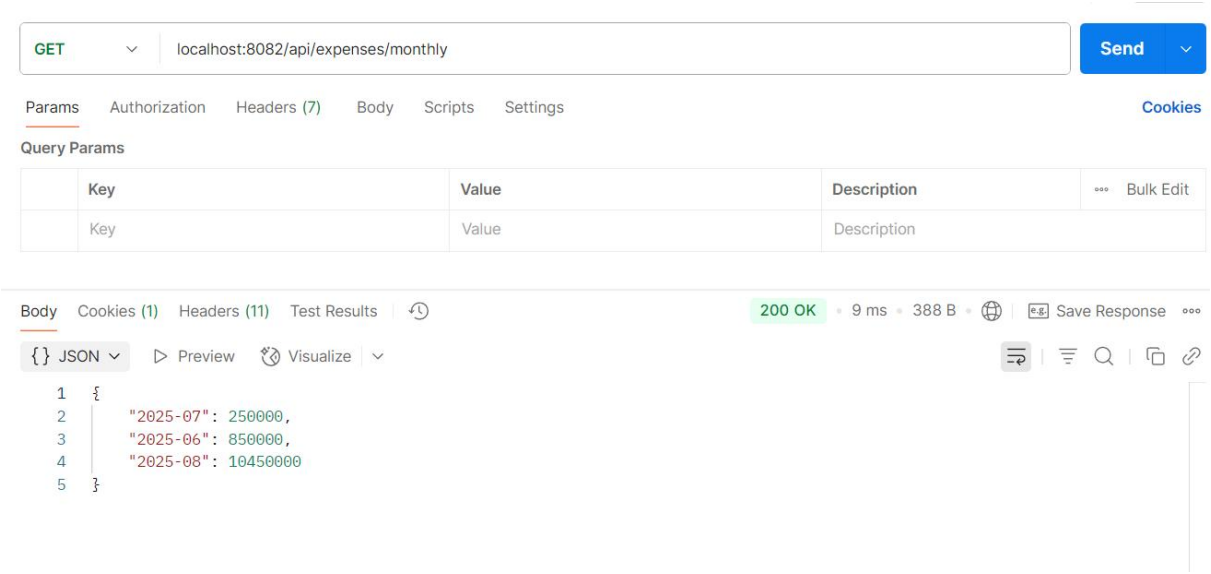
Nếu không tìm thấy khoản chi với id đã cho, hệ thống có thể trả về mã lỗi 404 Not Found.



Hình 23 API chi tiêu không tìm thấy

Lỗi này để thông báo cho người dùng rằng khoản chi yêu cầu không tồn tại.

Có thể xảy ra với các yêu cầu GET, DELETE, PUT khi tham số id không hợp lệ hoặc không tìm thấy.



Hình 24 API tổng chi tiêu từng tháng

Endpoint:/api/expenses/monthly
Phương thức:GET

Mục tiêu: Truy xuất tổng số tiền chi tiêu của từng tháng.
API trả về dữ liệu dạng JSON, trong đó mỗi khóa là tháng (YYYY-MM), giá trị là tổng số tiền chi trong tháng đó.

API này phục vụ cho chức năng thống kê chi tiêu theo tháng, giúp người dùng có cái nhìn tổng quan về mức chi tiêu của mình.

Có thể dùng dữ liệu này để vẽ biểu đồ hoặc báo cáo tài chính hàng tháng.

Nếu không có dữ liệu chi trong tháng nào đó, tháng đó sẽ không xuất hiện trong kết quả trả về.

CHƯƠNG 7. ĐÁNH GIÁ VÀ KẾT LUẬN

7.1. Những khó khăn gặp phải trong quá trình thực hiện

Hạn chế của kiến trúc monolithic trong mở rộng và bảo trì

Kiến trúc monolithic có ưu điểm là đơn giản về mặt triển khai và phát triển ban đầu, tuy nhiên khi ứng dụng phát triển lớn hơn, việc bảo trì và mở rộng có thể gặp khó khăn do tất cả các thành phần gắn kết chặt chẽ trong một khối duy nhất. Điều này làm giảm tính linh hoạt trong việc cập nhật, mở rộng hoặc sửa lỗi mà không ảnh hưởng đến toàn bộ hệ thống.

Thiết kế cơ sở dữ liệu phù hợp với yêu cầu quản lý chi tiêu cá nhân

Mặc dù ứng dụng dành cho cá nhân với dữ liệu không quá lớn, việc thiết kế bảng, quan hệ giữa các bảng để tối ưu truy vấn và dễ dàng mở rộng là một thách thức, đặc biệt khi cần đảm bảo tính toàn vẹn dữ liệu.

Bảo mật thông tin cá nhân trong ứng dụng đơn người dùng

Việc bảo vệ dữ liệu cá nhân, bao gồm thông tin đăng nhập và dữ liệu chi tiêu tài chính, đòi hỏi áp dụng các biện pháp bảo mật như mã hóa mật khẩu, quản lý phiên đăng nhập an toàn, tránh rò rỉ dữ liệu dù ứng dụng không có nhiều người dùng.

Thiết kế giao diện đơn giản, thân thiện và dễ sử dụng

Giao diện cần được xây dựng sao cho người dùng cá nhân có thể dễ dàng nhập liệu, theo dõi và phân tích chi tiêu mà không gặp khó khăn trong thao tác. Việc cân bằng giữa đầy đủ tính năng và sự đơn giản là thử thách trong thiết kế UX/UI

Quản lý và kiểm soát phiên bản mã nguồn trong dự án cá nhân

Mặc dù dự án có quy mô nhỏ, việc quản lý mã nguồn khoa học vẫn rất cần thiết để tránh lỗi phát sinh do thay đổi code và giúp việc bảo trì, nâng cấp trở nên dễ dàng hơn.

7.2. Bài học rút ra và đề xuất cải thiện trong tương lai

Lựa chọn kiến trúc phù hợp theo quy mô dự án

Đối với ứng dụng quy mô nhỏ hoặc cá nhân, kiến trúc monolithic là lựa chọn hợp lý nhờ sự đơn giản trong phát triển và triển khai. Tuy nhiên, nếu dự định mở rộng hoặc phát triển ứng dụng phức tạp hơn, nên cân nhắc áp dụng kiến trúc microservices hoặc modular để tăng tính linh hoạt và khả năng bảo trì.

Tối ưu hóa thiết kế cơ sở dữ liệu và truy vấn

Thiết kế cơ sở dữ liệu cần đảm bảo nguyên tắc chuẩn hóa, giảm thiểu dữ liệu dư thừa và tối ưu truy vấn nhằm nâng cao hiệu suất, đặc biệt khi dữ liệu phát triển về lâu dài.

Đảm bảo bảo mật ngay từ đầu

Áp dụng các biện pháp bảo mật cơ bản và chuẩn mực như mã hóa mật khẩu, sử dụng HTTPS, quản lý phiên đăng nhập an toàn, để bảo vệ dữ liệu cá nhân và tăng độ tin cậy của ứng dụng.

Phát triển giao diện người dùng dựa trên trải nghiệm thực tế

Tập trung xây dựng giao diện đơn giản, trực quan, hỗ trợ người dùng dễ dàng nhập liệu và quản lý thông tin. Có thể áp dụng các phương pháp test UX để cải thiện trải nghiệm người dùng.

Quản lý mã nguồn và quy trình phát triển chuyên nghiệp

Dù là dự án cá nhân, việc sử dụng hệ thống quản lý mã nguồn (Git) và áp dụng quy trình kiểm thử (unit test) sẽ giúp tăng chất lượng phần mềm, giảm thiểu lỗi và thuận tiện cho việc bảo trì, nâng cấp trong tương lai.

CHƯƠNG 8. PHỤ LỤC

8.1. Hướng dẫn cài đặt và chạy ứng dụng

8.1.1. Cài đặt môi trường

Cài đặt JDK

Truy cập: trang chính thức của Oracle:

https://www.oracle.com/java/technologies/downloads/#java21

-Chọn hệ điều hành tương ứng (Windows/macOS/Linux).

-Tải xuống và cài đặt JDK 21.

8.1.2. Cài đặt XAMPP

Truy cập trang: https://www.apachefriends.org/download.html

Tải phiên bản phù hợp với hệ điều hành và cài đặt

Mở **XAMPP Control Panel**, bật:

Apache

MySQL

8.1.3. Cài đặt cơ sở dữ liệu

Truy cập: http://localhost/phpmyadmin

Chọn **Database** → Nhập tên: quan_ly_chi_tieu_ca_nhan → Bấm **Create**.

Chọn **Import** → Tải lên file quan_ly_chi_tieu_ca_nhan.sql từ thư mục dự án → Bấm **Go** để import.

8.1.4. Cài đặt Postman

Truy cập trang chính thức: https://www.postman.com/downloads/

Tải bản dành cho hệ điều hành đang sử dụng (Windows/macOS/Linux)

Cài đặt theo hướng dẫn và đăng nhập (có thể dùng Google để đăng nhập nhanh)

8.1.5. Clone Ứng Dụng & Truy Cập

git clone https://github.com/haivoDA22TTD/CNPM_QLCTCN_Spring-Boot

cd CNPM_QLCTCN_Spring-Boot

8.1.6. Cài Đặt Thư Viện Phụ Thuộc

Chạy lệnh sau để cài đặt toàn bộ các thư viện cần thiết của dự án:

```
mvn clean install
```

8.1.7. Khởi Động Ứng Dụng

Sử dụng Maven để chạy ứng dụng Spring Boot:

```
mvn spring-boot:run
```

8.1.8. Ứng dụng sẽ chạy tại địa chỉ:

http://localhost:8082

8.2. Link GitHub và repository

https://github.com/haivoDA22TTD/CNPM_QLCTCN_Spring-Boot