KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**Báo Cáo Kết Thúc Môn Công Nghệ Phần Mềm**

**Học Kỳ II Năm Học 2024-2025**

**THIẾT KẾ ỨNG DỤNG WEB QUẢN LÝ CHI TIÊU CÁ NHÂN**

**BẰNG SPRING BOOT**

|  |  |
| --- | --- |
|  | *Sinh viên thực hiện:*  Võ Chí Hải-110122068-DA22TTD  Nguyễn Đỗ Thành Lộc-110122105-DA22TTD  Hoàng Tuấn Kiệt-110122099-DA22TTD |

**Giáo viên hướng dẫn : Nguyễn Bảo Ân**

***Trà Vinh, Tháng 2 Năm 2025***

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**Báo Cáo Kết Thúc Môn Công Nghệ Phần Mềm**

**Học Kỳ II Năm Học 2024-2025**

**THIẾT KẾ ỨNG DỤNG WEB QUẢN LÝ CHI TIÊU CÁ NHÂN**

**BẰNG SPRING BOOT**

|  |
| --- |
| *Sinh viên thực hiện:*  Võ Chí Hải-110122068-DA22TTD  Nguyễn Đỗ Thành Lộc-110122105-DA22TTD  Hoàng Tuấn Kiệt-110122099-DA22TTD |

**Giáo viên hướng dẫn : Nguyễn Bảo Ân**

***Trà Vinh, Tháng 2 Năm 2025***

*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, em xin gửi lời cảm ơn đến các Quý Thầy cô Khoa 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 Giảng viên hướng dẫn 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. 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**

[CHƯƠNG 1. GIỚI THIỆU 1](#_Toc27448)

[1.1. Giới thiệu đề tài 1](#_Toc28699)

[1.2. Mục tiêu của ứng dụng 1](#_Toc30031)

[1.3. Lý do chọn đề tài 1](#_Toc9415)

[CHƯƠNG 2. PHÂN TÍCH YÊU CẦU 3](#_Toc10518)

[2.1. Các chức năng chính của hệ thống 3](#_Toc13366)

[2.1.1Quản lý người dùng 3](#_Toc6687)

[2.1.2.Ghi chép thu – chi 3](#_Toc9275)

[2.1.3.Quản lý danh mục chi tiêu 3](#_Toc7509)

[2.1.4.Xem thống kê và báo cáo tài chính 3](#_Toc19181)

[2.1.5.Đặt hạn mức chi tiêu 4](#_Toc6627)

[2.2. Các yêu cầu phi chức năng 4](#_Toc32373)

[2.2.1.Hiệu năng 4](#_Toc4687)

[2.2.2.Tính khả dụng 4](#_Toc12349)

[2.2.3.Bảo mật 4](#_Toc12147)

[2.2.4.Tính mở rộng 4](#_Toc13952)

[2.2.5.Khả năng bảo trì 5](#_Toc9274)

[2.2.6.Tính thân thiện với người dùng 5](#_Toc24473)

[CHƯƠNG 3. KIẾN TRÚC HỆ THỐNG 6](#_Toc14411)

[3.1. Kiến trúc tổng thể hệ thống 6](#_Toc1147)

[3.2. Sơ Đồ Kiến Trúc Hệ Thống 6](#_Toc30984)

[3.3. Sơ đồ ERD 7](#_Toc15417)

[3.4. Thiết kế api swagger mô tả các endpoint cấu trúc request respone 7](#_Toc19339)

[2.3.2.Cơ sở dữ liệu 10](#_Toc8334)

[2.3.3.Công cụ Hỗ trợ 10](#_Toc12171)

[Container: 10](#_Toc21430)

[Docker Image: 10](#_Toc16963)

[Dockerfile: 11](#_Toc14883)

[Docker Compose: 11](#_Toc18909)

[Docker Hub: 12](#_Toc11988)

[3.5. Kiến trúc Ứng dụng 12](#_Toc5358)

**Danh mục hình ảnh và bảng biểu**

*[Hình 1 Sở đồ kiến trúc của hệ thống 1](#_Toc30666)* [6](#_Toc30666)

*[HÌnh 2 Sở đồ ERD](#_Toc32214)* [7](#_Toc32214)

*[Bảng 1 Mô tả các API và Endpoint](#_Toc19802)* [9](#_Toc19802)

# GIỚI THIỆU

## 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.

## Mục tiêu của ứng dụng

Ứng dụng hướng đến việc hỗ trợ người dùng:

Quản lý thu nhập và chi tiêu hàng ngày.

Phân loại các khoản chi theo danh mục (ăn uống, đi lại, học tập…).

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.

## 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 (hoặc cá nhân) 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 và tính mở rộng của hệ thống.

Bên cạnh đó, việc 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 giúp sinh viên không chỉ hiểu rõ hơn về cấu trúc và luồng xử lý trong một ứng dụng web, mà còn rèn luyện được các kỹ năng như thiết kế API RESTful, quản lý cơ sở dữ liệu quan hệ, triển khai ứng dụng thực tế, xử lý bảo mật và xác thực người dùng.

# PHÂN TÍCH YÊU CẦU

## Các chức năng chính của hệ thống

## 2.1.1Quản lý người dùng

Đăng ký tài khoản: người dùng có thể tạo tài khoản mới với tên người dùng và mật khẩu.

Đăng nhập/đăng xuất: xác thực người dùng và cho phép đăng nhập 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: loại giao dịch (thu/chi), số tiền, danh mục, ghi chú, ngày thực hiện.

Hỗ trợ phân loại khoản mục theo danh mục như: Ăn uống, Giao thông, Giải trí, Học tập, Khác...

Cho phép chỉnh sửa hoặc xóa giao dịch đã tạo

### 2.1.3.Quản lý danh mục chi tiêu

Cho phép người dùng tùy chỉnh danh mục chi tiêu theo nhu cầu (thêm, sửa, xóa danh mục).

Hệ thống cung cấp sẵn một số danh mục mặc định.

Các danh mục được sử dụng để thống kê và hiển thị biểu đồ.

### 2.1.4.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.

### 2.1.5.Đặt hạn mức chi tiêu

Người dùng có thể thiết lập hạn mức chi tiêu cho từng tháng.

Khi tổng chi tiêu vượt quá hạn mức, hệ thống sẽ gửi cảnh báo.

Giao diện hiển thị phần trăm mức tiêu dùng so với hạn mức đã đặt.

## 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.Tính khả dụng

Hệ thống cần hoạt động ổn định 24/7 trên môi trường triển khai (Heroku).

Có thể xử lý nhiều người dùng đồng thời.

### 2.2.3.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.

### 2.2.4.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) được xây dựng hoàn toàn độc lập và giao tiếp với frontend thông qua các API chuẩn. 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.

Backend tách biệt với frontend, có thể phát triển phiên bản di động sau này.

### 2.2.5.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.

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

### 2.2.6.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.

# KIẾN TRÚC HỆ THỐNG

## Kiến trúc tổng thể hệ thống

Hệ thống được xây dựng theo mô hình Client - Server với các thành phần chính:

Client: Giao diện người dùng (Web/App) cho phép người dùng tương tác với hệ thống.

Server (API): Là nơi xử lý logic, xác thực người dùng, xử lý dữ liệu và phản hồi các yêu cầu từ client.

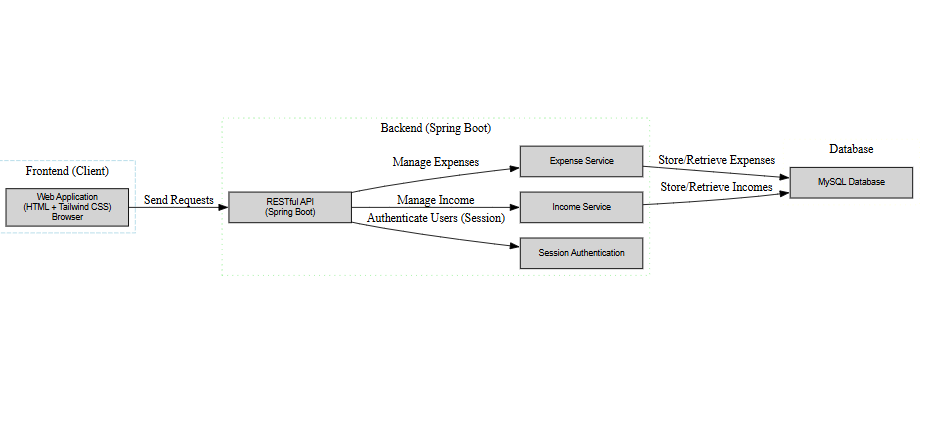
Cơ sở dữ liệu (Database): Lưu trữ toàn bộ dữ liệu liên quan đến người dùng.

Frontend: Được xây dựng với HTML, CSS (dùng Tailwind CSS để làm đẹp và responsive).

Backend: Spring Boot xử lý API và logic nghiệp vụ.

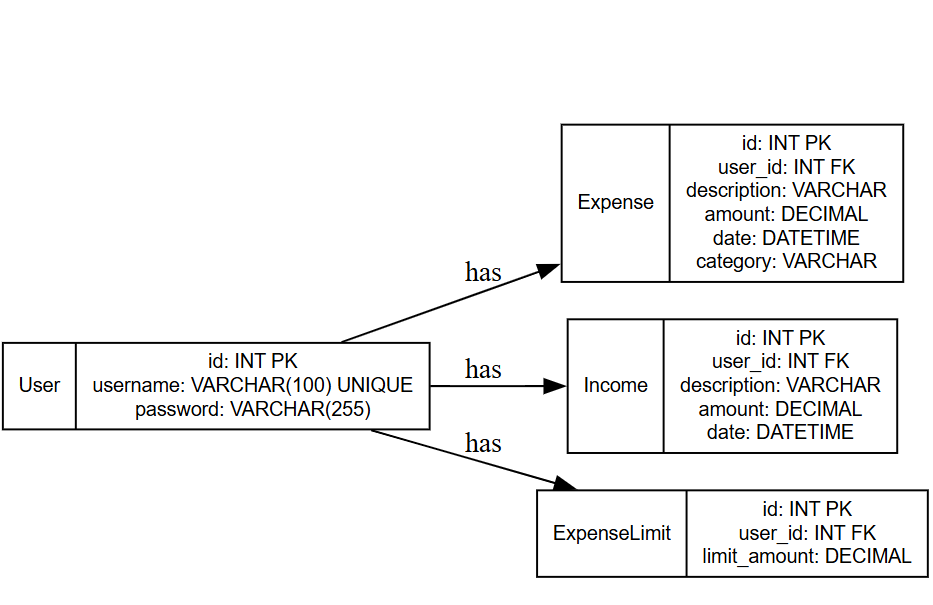
Database: MySQL lưu trữ dữ liệu người dùng, chi tiêu và thu nhập

## Sơ Đồ Kiến Trúc Hệ Thống



*Hình 1 Sở đồ kiến trúc của hệ thống 1*

## Sơ đồ ERD



*HÌnh 2 Sở đồ ERD*

## Thiết kế api swagger mô tả các endpoint cấu trúc request respone

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | Phương thức | Mô tả | Tham số | Yêu cầu  (Request) | Phản hồi (Response) |
| /auth/login | POST | Đăng nhập người dùng bằng username và password | username: Tên người dùng password: Mật khẩu | Json{  "username": "user1",  "password": "password123"} | - **200 OK**:  Đăng nhập thành công. - **401 Unauthorized**:  Thông tin đăng nhập  không hợp lệ. |
| /expenses | GET | Lấy danh sách chi tiêu của người dùng | - Không có tham số (Yêu cầu xác thực người dùng qua đăng nhập) | - Không có request body. | - **200 OK**:  Danh sách chi tiêu  của người dùng.  json[  { "id": 1,  "description":  "Mua cà phê",  "amount": 50.000  "date":  "2023-05-01",  "category": "Ăn uống" } ] |
| /expenses | POST | Thêm mới một giao dịch chi tiêu | - description: Mô tả chi tiêu - amount: Số tiền chi tiêu - category: Danh mục chi tiêu | json<br>{  "description": "Mua cà phê",  "amount": 50.00,  "category": "Ăn uống"} | **201 Created**:  Chi tiêu đã được  tạo thành công. json{ "id": 1 }  **400 Bad Request**:  Thông tin chi tiêu  không hợp lệ. |
| /expenses/  {expenseId} | PUT | Cập nhật một giao dịch chi tiêu | expenseId: ID của chi tiêu | json{"description": "Mua trà sữa",  "amount": 55.00,  "category":  "Ăn uống"} | **200 OK**  : Chi tiêu  đã được  cập nhật  thành  công. **400 Bad Request**:  Dữ liệu  không hợp  lệ.  **404 Not Found**:  Không  tìm thấy  chi tiêu  với ID  này. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| /expenses/  {expenseId} | DELETE | Xóa một giao dịch chi tiêu theo ID | expenseId: ID của chi tiêu cần xóa | Không có request body. | **204 No Content**:  Xóa chi tiêu  thành công. **404 Not Found**:  Không tìm thấy  chi tiêu với ID này. |
| /expenselimit | GET | Lấy giới hạn chi tiêu của người dùng | Không có tham số (Yêu cầu xác thực người dùng qua đăng nhập) | Không có request body. | **200 OK**:  Trả về giới hạn chi  tiêu của người dùng  json{ "limit\_amount"  : 1000.00 } **401 Unauthorized**:  Không có quyền  truy cập. |
| /expenselimit | PUT | Cập nhật giới hạn chi tiêu của người dùng | limit\_  amount: Số tiền giới hạn chi tiêu | json{  "limit\_amout: 1500.00} | - **200 OK**:  Cập nhật giới hạn  chi tiêu thành công. - **400 Bad Request**:  Thông tin giới hạn  chi tiêu không  hợp lệ. |

*Bảng 1 Mô tả các API và Endpoint*

Spring Boot là một framework phát triển ứng dụng Java, giúp đơn giản hóa quá trình phát triển ứng dụng web. Một số tính năng nổi bật của Spring Boot gồm có:

**Khởi tạo Dự án Nhanh chóng**: Cung cấp cấu hình mặc định cho các ứng dụng, giúp tiết kiệm thời gian.

**Tích hợp Dễ dàng với các Thư viện**: Hỗ trợ tích hợp các thư viện khác như Spring Data JPA, Spring Security.

**Quản lý Cấu hình**: Cung cấp các công cụ để quản lý cấu hình ứng dụng một cách linh hoạt.

### 2.3.2.Cơ sở dữ liệu

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến, được sử dụng để lưu trữ dữ liệu trong ứng dụng. Một số ưu điểm của MySQL gồm:

**Hiệu suất Cao**: MySQL có khả năng xử lý các truy vấn lớn một cách nhanh chóng.

**Dễ Dàng Tích Hợp**: Dễ dàng tích hợp với Spring Boot thông qua JPA (Java Persistence API).

**Quản lý Dữ liệu Linh hoạt**: Hỗ trợ các loại dữ liệu phong phú và cho phép thực hiện các thao tác CRUD (Create, Read, Update, Delete) dễ dàng.

### 2.3.3.Công cụ Hỗ trợ

**Postman**: Dùng để kiểm thử và bắt API, giúp kiểm tra các chức năng của ứng dụng một cách nhanh chóng và hiệu quả.

**Jira**: Quản lý dự án và theo dõi tiến độ phát triển, giúp nhóm làm việc phối hợp và xử lý các vấn đề phát sinh.

**Docker**: Triển khai ứng dụng trong môi trường container, đảm bảo tính nhất quán và dễ dàng trong việc quản lý môi trường phát triển và sản xuất.

Docker là một công cụ rất mạnh mẽ giúp đóng gói, triển khai và chạy các ứng dụng trong các container. Thay vì cài đặt trực tiếp ứng dụng trên hệ điều hành, Docker giúp tạo ra môi trường cách ly cho ứng dụng, giúp dễ dàng triển khai và di chuyển ứng dụng giữa các máy khác nhau mà không gặp phải vấn đề về môi trường hoặc phụ thuộc.

Để hiểu rõ hơn về Docker, có thể chia nó thành những phần chính sau:

### ****Container****:

Một container là một môi trường cách ly trong đó ứng dụng của bạn chạy. Nó giống như một máy ảo nhẹ nhưng không yêu cầu hệ điều hành riêng biệt, mà chia sẻ kernel của hệ điều hành gốc.

Container giúp bảo đảm rằng ứng dụng chạy trong môi trường nhất quán, bất kể môi trường máy chủ là gì.

### ****Docker Image****:

Docker Image là một bản sao của hệ thống file và ứng dụng cần thiết để chạy ứng dụng trong container. Một Docker Image thường được xây dựng từ một file cấu hình gọi là Dockerfile.

Khi bạn tạo một container từ một Docker Image, nó sẽ giống hệt với môi trường bạn đã cấu hình trong Image.

### ****Dockerfile****:

Dockerfile là một file chứa các lệnh để tạo Docker Image. Ví dụ như cài đặt phần mềm, thiết lập môi trường, sao chép các tệp từ máy của bạn vào container, v.v.

### ****Docker Compose****:

Docker Compose là một công cụ giúp bạn định nghĩa và chạy nhiều container Docker cùng một lúc. Thường dùng khi bạn cần chạy các dịch vụ như web server, database, caching server, v.v. trong cùng một môi trường.

### ****Docker Hub****:

Docker Hub là một dịch vụ trực tuyến cho phép bạn lưu trữ và chia sẻ Docker Images. Nó giống như một kho chứa các ứng dụng, nơi bạn có thể tải các image sẵn có hoặc tải lên các image của riêng mình.

## Kiến trúc Ứng dụng

Ứng dụng sẽ sử dụng kiến trúc RESTful, trong đó:

**REST API**: Cung cấp một tập các endpoint cho phép giao tiếp giữa giao diện và máy chủ, giúp thực hiện các thao tác như tạo, đọc, cập nhật và xóa dữ liệu.

**Service Layer**: Chứa logic xử lý nghiệp vụ, tách biệt việc xử lý dữ liệu và giao tiếp với cơ sở dữ liệu.