

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



BÁO CÁO BÀI TẬP LỚN

Đề tài: Xây dựng website blog

Sinh viên thực hiện:	Vũ Trọng Khôi
Giảng viên hướng dẫn:	Ths Nguyễn Quốc Uy
Mã sinh viên	B22DCCN468
Lớp:	D22CQCN12-B
Khóa:	2022 – 2027

Hà Nội – 2025

LỜI CẢM ƠN

Trước tiên, em muốn gửi lời cảm ơn sâu sắc nhất đến thầy giáo Nguyễn Quốc Uy, người đã tận tình hướng dẫn cũng như góp ý cho em trong suốt quá trình nghiên cứu và thực hiện bài tập lớn để em có thể hoàn thành thật tốt bài tập này.

Em xin chân thành cảm ơn!

Hà Nội, tháng 5 năm 2025

Sinh viên

Vũ Trọng Khôi

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM

(Của giảng viên hướng dẫn)

[illegible]

..... **Điểm:**
 **(Bằng chữ:**)

Hà Nội, tháng 5 năm 2025

GIẢNG VIÊN HƯỚNG DẪN

NGUYỄN QUỐC UY

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM

(Của giảng viên phản biện)

[illegible]

..... **Điểm:**
 (Bằng chữ:)

Hà Nội, tháng 5 năm 2025

CÁN BỘ - GIẢNG VIÊN PHẢN BIỆT

GIỚI THIỆU ĐỀ TÀI

Trong kỷ nguyên số phát triển mạnh mẽ như hiện nay, việc chia sẻ kiến thức và học hỏi thông qua các nền tảng trực tuyến đã trở thành xu hướng phổ biến và được đông đảo người dùng đón nhận. Đặc biệt trong lĩnh vực công nghệ thông tin - một lĩnh vực không ngừng thay đổi và phát triển - thì nhu cầu tiếp cận nguồn tri thức chất lượng, cập nhật liên tục lại càng trở nên quan trọng hơn bao giờ hết. Việc xây dựng một nền tảng trực tuyến nhằm cung cấp thông tin, chia sẻ kinh nghiệm lập trình, cập nhật xu hướng công nghệ mới là rất cần thiết và mang ý nghĩa thực tiễn cao.

Trong những năm gần đây, cùng với sự bùng nổ của thương mại điện tử và mạng xã hội, các nền tảng blog cá nhân và website chuyên ngành về công nghệ cũng phát triển mạnh mẽ. Không chỉ các chuyên gia, mà cả sinh viên và lập trình viên trẻ tuổi cũng có xu hướng tìm kiếm và đóng góp nội dung chia sẻ kiến thức, kinh nghiệm thực tế trong học tập và làm việc. Tuy nhiên, các nền tảng hiện tại thường có phạm vi rộng, thiếu sự chuyên sâu hoặc phân mảnh thông tin, gây khó khăn cho người dùng trong việc tìm kiếm các chủ đề liên quan đến lập trình và công nghệ một cách hệ thống, có tổ chức. Từ thực tế đó, đề án tốt nghiệp này đề xuất xây dựng một hệ thống website blog chia sẻ kiến thức công nghệ với giao diện hiện đại, thân thiện và dễ sử dụng.

Hệ thống website mà đề án xây dựng nhằm cung cấp một không gian trực tuyến cho người dùng có thể đăng tải, tìm kiếm và đọc các bài viết liên quan đến nhiều chủ đề trong lĩnh vực công nghệ như lập trình web, phát triển phần mềm, bảo mật, trí tuệ nhân tạo, cơ sở dữ liệu... Bên cạnh đó, hệ thống cũng hỗ trợ các tính năng như tạo và quản lý bài viết, phân loại bài viết theo chuyên mục, tìm kiếm bằng từ khóa và giao diện đọc bài viết tối ưu trải nghiệm người dùng. Ngoài ra, người dùng có thể đăng ký tài khoản để theo dõi tác giả yêu thích, thích bài viết và tham gia bình luận để tạo nên một cộng đồng học hỏi và phát triển.

Về phía quản trị viên, hệ thống cung cấp giao diện quản lý nội dung giúp kiểm duyệt bài viết, quản lý người dùng, bài viết, danh mục và bình luận qua đó duy trì chất lượng và an toàn nội dung trên nền tảng.

Về mặt công nghệ, hệ thống được xây dựng với frontend sử dụng ReactJS - một trong những thư viện giao diện người dùng phổ biến và mạnh mẽ nhất hiện nay, backend sử dụng Spring Boot - một framework Java hiện đại, hỗ trợ xây dựng API REST hiệu quả, kết hợp với hệ quản trị cơ sở dữ liệu MySQL nhằm lưu trữ và quản lý dữ liệu một cách an toàn và ổn định. Việc lựa chọn các công nghệ trên không chỉ phù hợp với xu hướng phát triển phần mềm hiện đại mà còn đảm bảo khả năng mở rộng và hiệu suất của hệ thống.

Hệ thống được thiết kế với giao diện đẹp mắt, dễ sử dụng, hiệu năng ổn định, phù hợp với cả người mới học lập trình lẫn các lập trình viên có kinh nghiệm. Hệ thống này không chỉ là nơi chia sẻ tri thức, mà còn hướng tới việc xây dựng một cộng đồng công

nghệ gắn kết, nơi mọi người có thể học hỏi lẫn nhau và cùng phát triển kỹ năng.

Nội dung đồ án gồm bốn chương: chương 1 trình bày cơ sở lý thuyết và giới thiệu các công nghệ được sử dụng trong hệ thống; chương 2 là phân phân tích và thiết kế hệ thống, bao gồm các yêu cầu chức năng, sơ đồ usecase, luồng hoạt động và thiết kế cơ sở dữ liệu; chương 3 trình bày quá trình cài đặt và kiểm thử hệ thống, kèm theo các giao diện thực tế và kết quả kiểm thử; cuối cùng, chương 4 là phần tổng kết những kết quả đã đạt được, các hạn chế còn tồn tại và định hướng phát triển hệ thống trong tương lai.

CHƯƠNG I. CƠ SỞ LÝ THUYẾT

1.1. Công nghệ Frontend

1.1.1. Giới thiệu

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, được sử dụng rộng rãi trong việc xây dựng các giao diện người dùng (UI) tương tác cho các ứng dụng web và di động. ReactJS hoạt động dựa trên kiến trúc component-based (thành phần), cho phép chia nhỏ giao diện thành các phần riêng biệt, có thể tái sử dụng và dễ dàng quản lý. Kiến trúc này giúp việc phát triển ứng dụng trở nên linh hoạt, mở rộng thuận tiện và tối ưu hóa hiệu năng tổng thể.

Kết hợp với ReactJS, đồ án sử dụng Tailwind CSS - một framework CSS hiện đại theo phương pháp utility-first, giúp xây dựng giao diện người dùng nhanh chóng và linh hoạt thông qua các lớp tiện ích (utility classes). Việc sử dụng Tailwind giúp giảm thiểu việc viết CSS thuần thủ công, tăng khả năng kiểm soát thiết kế ngay trong file JSX và đẩy nhanh quá trình xây dựng giao diện đẹp mắt, nhất quán.

1.1.2. Tính năng nổi bật

ReactJS có những tính năng nổi bật sau:

- **JSX (JavaScript XML):** Cung cấp cú pháp kết hợp giữa HTML và JavaScript, giúp việc viết giao diện trực quan hơn. JSX không được trình duyệt hiểu trực tiếp mà sẽ được biên dịch qua Babel.
- **Virtual DOM:** React sử dụng Virtual DOM để giảm thiểu số lần thao tác trực tiếp lên DOM thực, từ đó cải thiện hiệu suất hiển thị và tốc độ cập nhật giao diện.
- **One-way Data Binding:** Dữ liệu được truyền theo một chiều từ component cha xuống component con, giúp kiểm soát luồng dữ liệu rõ ràng và giảm thiểu lỗi.
- **Component tái sử dụng:** Các phần giao diện được tổ chức thành các component có thể tái sử dụng, giúp dễ bảo trì và phát triển ứng dụng lớn một cách hiệu quả.

Tailwind CSS nổi bật với các đặc điểm sau:

- **Utility-first:** Thiết kế giao diện bằng cách kết hợp các lớp tiện ích nhỏ, giúp lập trình viên không phải tạo quá nhiều class riêng lẻ.
- **Tùy biến mạnh mẽ:** Cho phép mở rộng và cấu hình theme, màu sắc, kích thước... phù hợp với từng hệ thống.
- **Tích hợp dễ dàng với ReactJS:** Tailwind được thiết kế tương thích tốt với React, cho phép định nghĩa phong cách trực tiếp trong file JSX giúp rút ngắn thời gian phát triển và đảm bảo tính nhất quán.

-
-
- **Responsive Design:** Tailwind hỗ trợ thiết kế giao diện phản hồi (responsive) với hệ thống breakpoint rõ ràng, dễ sử dụng.

1.1.3. Ưu nhược điểm

Ưu điểm của ReactJS:

- **Hiệu suất cao:** Nhờ Virtual DOM và khả năng cập nhật tối ưu, React đảm bảo hiệu suất ứng dụng ổn định.
- **Tái sử dụng code:** Mỗi component đều có thể tái sử dụng và dễ dàng bảo trì.
- **Cộng đồng lớn:** Được sử dụng rộng rãi nên React có hệ sinh thái phong phú và cộng đồng hỗ trợ mạnh mẽ.

Ưu điểm của Tailwind CSS:

- **Tăng tốc độ phát triển:** Không cần viết nhiều CSS tùy chỉnh, chỉ cần kết hợp các class sẵn có.
- **Giao diện nhất quán:** Việc sử dụng các utility class đảm bảo sự nhất quán trong thiết kế toàn hệ thống.
- **Tối ưu hóa CSS:** Hỗ trợ loại bỏ CSS không dùng đến (tree-shaking) trong quá trình build, giảm dung lượng file.

Nhược điểm của ReactJS:

- **Chỉ xử lý phần giao diện:** React không bao gồm sẵn các chức năng như định tuyến, quản lý trạng thái hay kết nối backend, nên cần kết hợp thêm thư viện như React Router, Redux hoặc Context API.
- **Nhược điểm của Tailwind CSS:**
- **Khó tiếp cận ban đầu:** Với người mới, việc phải nhớ và kết hợp nhiều class có thể gây khó khăn.
- **Markup dài:** Các file JSX có thể trở nên dài và khó đọc nếu không tổ chức hợp lý.

Vì những ưu điểm nổi bật về hiệu suất, tính linh hoạt và khả năng mở rộng, đồ án đã lựa chọn sử dụng kết hợp ReactJS và Tailwind CSS để xây dựng giao diện người dùng cho hệ thống blog chia sẻ kiến thức công nghệ. Sự kết hợp này không chỉ giúp tạo ra giao diện hiện đại, thân thiện mà còn dễ bảo trì, dễ mở rộng và mang lại trải nghiệm người dùng tốt hơn.

1.2. Công nghệ Backend

1.2.1. Giới thiệu

Spring Boot là một phần mở rộng của Spring Framework - một trong những framework mạnh mẽ và phổ biến nhất trong lập trình Java. Spring Boot được thiết kế để

giúp lập trình viên xây dựng các ứng dụng Java-based backend một cách nhanh chóng và dễ dàng hơn bằng cách loại bỏ cấu hình phức tạp vốn có của Spring truyền thống. Spring Boot cung cấp khả năng cấu hình tự động, tích hợp sẵn các dependency phổ biến và hỗ trợ khởi tạo ứng dụng với một cấu trúc rõ ràng. Spring Boot đặc biệt phù hợp cho việc xây dựng RESTful API, hệ thống dịch vụ web hiện đại và được sử dụng rộng rãi trong các hệ thống thương mại điện tử, hệ thống blog, hệ thống quản lý nội dung và nhiều ứng dụng web khác.

1.2.2. Tính năng nổi bật

Spring Boot mang lại nhiều tính năng nổi bật:

- Cấu hình tự động (Auto Configuration): Spring Boot tự động thiết lập các thành phần phù hợp với các dependency có trong project, giúp rút ngắn thời gian cấu hình thủ công.
- Web Development: Spring Boot hỗ trợ phát triển ứng dụng web một cách dễ dàng, cho phép tạo ứng dụng HTTP tự chứa với các máy chủ nhúng như Tomcat hoặc Jetty. Mô-đun spring-boot-starter-web giúp khởi động ứng dụng nhanh chóng.
- Tích hợp RESTful API dễ dàng: Spring Boot cung cấp sẵn các annotation như `@RestController`, `@RequestMapping`, `@GetMapping`, v.v. để xây dựng API một cách nhanh chóng.
- Tích hợp dễ dàng với cơ sở dữ liệu: Hỗ trợ các công nghệ như JPA (Java Persistence API), Hibernate, JDBC... cùng khả năng kết nối dễ dàng với các hệ quản trị cơ sở dữ liệu như MySQL.
- Quản lý phụ thuộc bằng Maven hoặc Gradle: Giúp kiểm soát và quản lý thư viện một cách hiệu quả, rõ ràng.
- Bảo mật và phân quyền: Spring Security cho phép triển khai các cơ chế xác thực (authentication) và phân quyền (authorization) một cách linh hoạt.
- Tích hợp Spring Data JPA: Giúp đơn giản hóa việc truy vấn và thao tác dữ liệu bằng các interface và các annotation.

1.2.3. Ưu và nhược điểm

Ưu điểm của Spring Boot:

Spring Boot mang lại nhiều lợi ích thiết thực cho quá trình phát triển backend, đặc biệt trong các dự án web hiện đại:

- Tự khởi động ứng dụng một cách đơn giản: Spring Boot cho phép xây dựng các ứng dụng Spring độc lập, có thể chạy trực tiếp thông qua lệnh `java -jar`, không yêu cầu triển khai lên máy chủ ứng dụng như Tomcat dưới dạng file WAR.
- Hỗ trợ HTTP server nhúng: Các máy chủ như Tomcat, Jetty hay Undertow được tích hợp sẵn, giúp kiểm thử hoặc triển khai ứng dụng dễ dàng mà không cần cấu

hình thủ công môi trường server bên ngoài.

- Starter dependency cấu hình sẵn: Spring Boot cung cấp các bộ starter POM đã cấu hình sẵn, giúp đơn giản hóa quá trình khai báo dependency trong Maven hoặc Gradle, đồng thời đảm bảo tính tương thích giữa các thư viện.
- Sẵn sàng cho môi trường production: Hệ thống tích hợp các tính năng giám sát như kiểm tra trạng thái (health check), theo dõi chỉ số (metrics), cấu hình động (externalized configuration), giúp ứng dụng dễ dàng triển khai và vận hành trong môi trường thực tế.
- Loại bỏ cấu hình XML phức tạp: Spring Boot sử dụng cấu hình theo kiểu Java (Java-based configuration) và annotation, không cần đến các file cấu hình XML dài dòng như trong Spring truyền thống.
- Công cụ CLI tiện lợi: Spring Boot CLI (Command Line Interface) hỗ trợ xây dựng, chạy thử và kiểm tra ứng dụng nhanh chóng qua dòng lệnh, đặc biệt hữu ích trong giai đoạn phát triển ban đầu hoặc thử nghiệm nhanh một tính năng.
- Tích hợp nhiều plugin phát triển: Spring Boot hỗ trợ tốt các công cụ như Spring Tool Suite (STS), IntelliJ IDEA, VS Code, Maven, Gradle,... với nhiều plugin được tối ưu hóa cho quy trình phát triển.
- Giảm thiểu mã lặp (boilerplate): Nhờ vào cơ chế tự động hóa cấu hình và annotation, lập trình viên không cần viết lại các đoạn mã lặp đi lặp lại, từ đó tăng tốc độ phát triển, đồng thời giảm khả năng phát sinh lỗi cấu hình.
- Nâng cao hiệu suất và giảm thời gian phát triển: Với khả năng thiết lập nhanh chóng, quản lý dễ dàng và hệ sinh thái phong phú, Spring Boot giúp các dự án backend tiết kiệm được nhiều thời gian mà vẫn đảm bảo chất lượng và hiệu suất vận hành.

Nhược điểm của Spring Boot

- Mặc dù sở hữu nhiều ưu điểm vượt trội, Spring Boot vẫn tồn tại một số hạn chế nhất định:
- Kích thước ứng dụng có thể lớn: Do sử dụng các starter mặc định chứa nhiều dependency không thực sự cần thiết với tất cả dự án, kích thước file JAR khi build có thể bị tăng không cần thiết.
- Tối ưu hóa khó khăn trong dự án lớn: Với các dự án phức tạp hoặc yêu cầu tối ưu cao, việc phụ thuộc vào các cấu hình tự động có thể gây khó khăn trong việc kiểm soát chi tiết từng thành phần.
- Tiềm ẩn rủi ro từ cấu hình mặc định: Các giá trị cấu hình mặc định có thể không phù hợp với mọi tình huống, do đó nếu không kiểm soát kỹ, có thể dẫn đến lỗi hoặc lỗ hổng bảo mật.

1.3. Cơ sở dữ liệu

1.3.1. Giới thiệu

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được phát triển

ban đầu bởi công ty MySQL AB và hiện nay thuộc sở hữu của Oracle Corporation. Với khả năng xử lý mạnh mẽ, hiệu năng cao và tính ổn định, MySQL là một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất hiện nay, được sử dụng rộng rãi trong cả các ứng dụng quy mô nhỏ đến hệ thống lớn, bao gồm các website thương mại điện tử, mạng xã hội và hệ thống quản lý doanh nghiệp. MySQL sử dụng ngôn ngữ truy vấn có cấu trúc (SQL - Structured Query Language) để tương tác và quản lý dữ liệu. Nhờ vào tính chất quan hệ, dữ liệu được lưu trữ trong các bảng có mối liên kết chặt chẽ, giúp dễ dàng tổ chức, tìm kiếm và phân tích dữ liệu hiệu quả.

1.3.2. Tính năng nổi bật

MySQL cung cấp nhiều tính năng nổi bật giúp quản lý và vận hành cơ sở dữ liệu hiệu quả:

- Hiệu suất cao và ổn định: MySQL được tối ưu hóa cho tốc độ và độ tin cậy, có khả năng xử lý hàng triệu truy vấn mỗi ngày trong các hệ thống lớn.
- Tính linh hoạt cao: Hỗ trợ nhiều loại lưu trữ (storage engines) như InnoDB (mặc định), MyISAM, MEMORY,... giúp người dùng lựa chọn mô hình lưu trữ phù hợp với từng nhu cầu cụ thể.
- Bảo mật dữ liệu: MySQL hỗ trợ xác thực người dùng bằng mật khẩu được mã hóa, đồng thời cho phép phân quyền chi tiết đến từng cơ sở dữ liệu, bảng, hoặc cột.
- Tương thích với nhiều nền tảng: Có thể triển khai trên nhiều hệ điều hành như Windows, Linux, macOS,... đồng thời dễ dàng tích hợp với các công nghệ như PHP, Java, Python, .NET,...
- Hỗ trợ giao dịch (transaction): Với engine InnoDB, MySQL hỗ trợ các đặc tính của giao dịch ACID (Atomicity, Consistency, Isolation, Durability), đảm bảo tính toàn vẹn và nhất quán của dữ liệu trong các hệ thống yêu cầu độ tin cậy cao.
- Sao lưu và phục hồi: Cung cấp nhiều công cụ như mysqldump, mysqlhotcopy, hoặc tích hợp với các phần mềm sao lưu để bảo vệ dữ liệu.

1.3.3. Ưu và nhược điểm

Ưu điểm của MySQL

- Dễ học, dễ sử dụng: Với cú pháp SQL quen thuộc và cộng đồng người dùng lớn, việc học và vận hành MySQL trở nên thuận tiện ngay cả với người mới bắt đầu.
- Mã nguồn mở và miễn phí: MySQL được phát hành theo giấy phép GPL, phù hợp với cả mục đích học tập và thương mại.
- Khả năng mở rộng: Có thể xử lý dữ liệu với số lượng lớn, dễ dàng mở rộng theo chiều ngang hoặc chiều dọc tùy thuộc vào hạ tầng.
- Tích hợp tốt với Spring Boot: Thư viện JDBC và Spring Data JPA cho phép MySQL hoạt động mượt mà trong các ứng dụng backend sử dụng Spring Boot.

Nhược điểm của MySQL

- Chưa tối ưu cho phân tích dữ liệu lớn (Big Data): So với các hệ quản trị cơ sở dữ

liệu chuyên biệt như PostgreSQL, Oracle hoặc NoSQL như MongoDB, MySQL còn hạn chế khi xử lý các bài toán phức tạp về phân tích dữ liệu hoặc dữ liệu phi cấu trúc.

- Tính năng nâng cao còn hạn chế: Một số tính năng cao cấp như hỗ trợ toàn vẹn tham chiếu nâng cao, trigger phức tạp hoặc stored procedure vẫn còn đơn giản hơn so với các hệ quản trị cơ sở dữ liệu cao cấp khác.

1.4. Công nghệ khác

Ngoài các công nghệ chính như ReactJS, TailwindCSS ở frontend, Spring Boot ở backend và MySQL cho cơ sở dữ liệu, đề án còn tích hợp thêm một số công nghệ hỗ trợ khác nhằm nâng cao trải nghiệm người dùng và tối ưu hóa hệ thống:

1.4.1. Websocket

WebSocket là giao thức giao tiếp hai chiều giữa client và server thông qua một kết nối TCP duy nhất. WebSocket cho phép cập nhật dữ liệu theo thời gian thực mà không cần client liên tục gửi yêu cầu mới. Trong đề án, WebSocket được sử dụng để xử lý các chức năng cần cập nhật nhanh như thông báo hoặc các hành động tương tác thời gian thực.

1.4.2. Cloudinary

Cloudinary là nền tảng quản lý và lưu trữ đa phương tiện trên đám mây, hỗ trợ upload, chuyển đổi định dạng, tối ưu hóa và phân phối hình ảnh/video. Đề án sử dụng Cloudinary để lưu trữ và xử lý ảnh người dùng hoặc ảnh bài viết, giúp giảm tải cho server backend, đồng thời cải thiện tốc độ tải trang và chất lượng hiển thị hình ảnh trên giao diện web.

1.5. Tổng kết

Chương I đã trình bày tổng quan về các công nghệ chính được sử dụng trong quá trình xây dựng hệ thống chia sẻ kiến thức công nghệ, bao gồm ReactJS và TailwindVSS ở phía frontend, Spring Boot ở phía backend và MySQL làm hệ quản trị cơ sở dữ liệu. Các công nghệ này được lựa chọn nhằm đảm bảo hiệu năng, khả năng mở rộng và trải nghiệm người dùng tốt nhất cho hệ thống. Bên cạnh đó, chương này cũng giới thiệu một số công nghệ hỗ trợ như WebSocket để xử lý các chức năng thời gian thực, Cloudinary phục vụ quản lý và tối ưu hóa hình ảnh.

Việc kết hợp các công nghệ hiện đại này không chỉ giúp hệ thống vận hành ổn định mà còn góp phần nâng cao trải nghiệm người dùng, đảm bảo tính linh hoạt và dễ mở rộng cho các nhu cầu phát triển trong tương lai. Đây là cơ sở kỹ thuật quan trọng để triển khai các chức năng của hệ thống trong các chương tiếp theo.

CHƯƠNG II. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

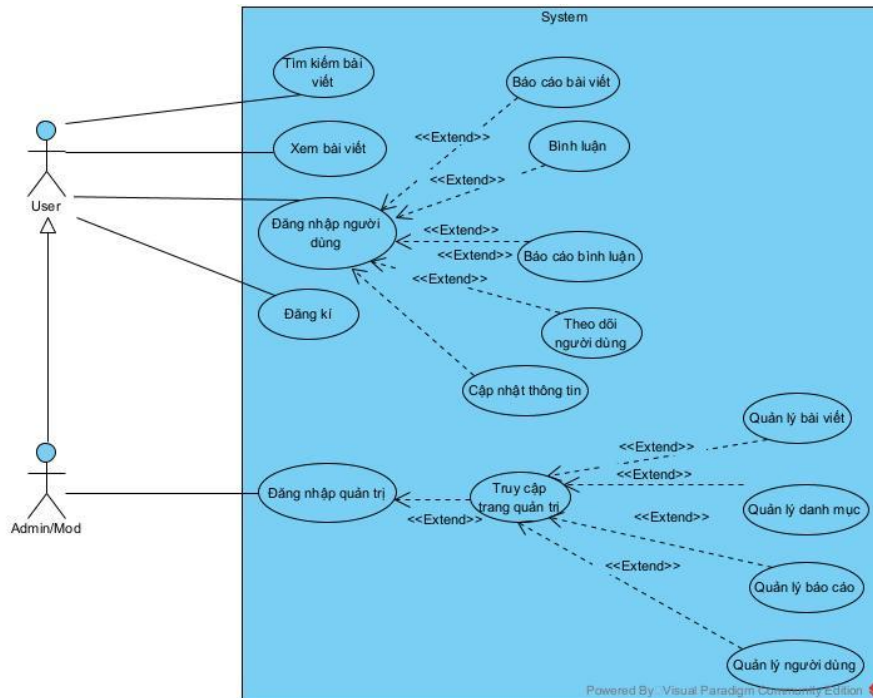
2.1. Tổng quan về nghiệp vụ

Hệ thống có 4 tác nhân chính là người dùng đã đăng nhập tài khoản, người dùng chưa đăng ký tài khoản (tài khoản khách), kiểm duyệt viên (mod) và quản trị viên (admin).

- Người dùng chưa đăng ký tài khoản là người dùng truy cập hệ thống chưa đăng ký tài khoản hay đăng nhập tài khoản. Sau khi đăng ký/ đăng nhập thành công sẽ trở thành user.
- Người dùng đã đăng ký tài khoản khác với người dùng chưa đăng ký tài khoản là có thể đăng bài, thích bài viết, bình luận, theo dõi người dùng.
- Admin là người quản trị toàn bộ hệ thống - người có tất cả các quyền thực thi các chức năng quản lý: danh mục, bài viết, bình luận, người dùng, báo cáo. Admin cũng có các quyền như người dùng.
- Mod là kiểm duyệt viên có quyền thực thi các chức năng như duyệt bài viết, duyệt báo cáo. Mod cũng có các quyền như người dùng

2.2. Phân tích yêu cầu chức năng

2.2.1. Tổng quan hệ thống

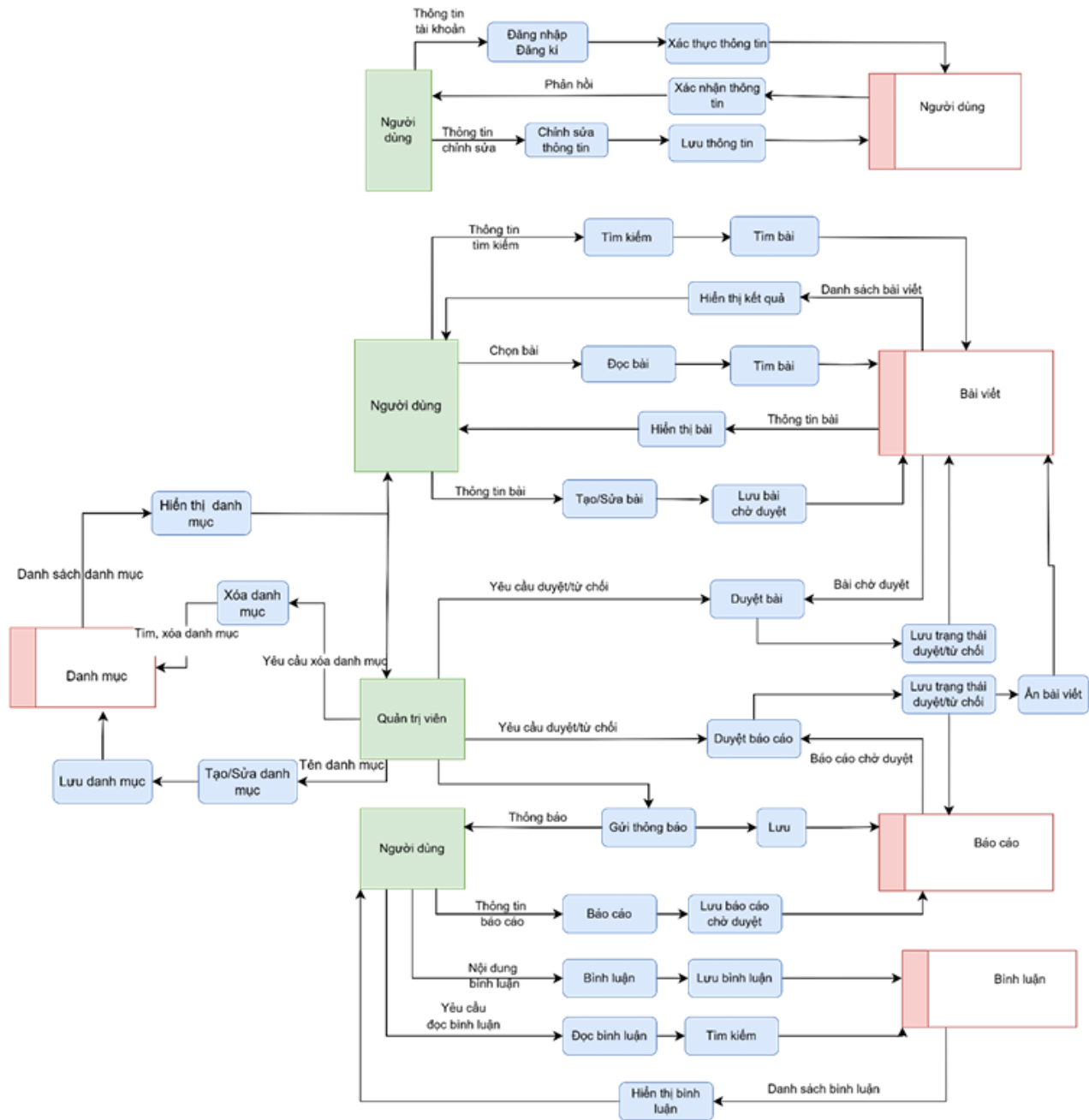


Hình 2. Usecase tổng quan hệ thống

Các usecase được mô tả chi tiết:

- Đăng kí: Usecase cho phép người dùng đăng kí tài khoản.
- Đăng nhập: Usecase cho phép người dùng đăng nhập vào hệ thống website
- Tìm kiếm bài viết: Usecase cho phép người dùng tìm kiếm bài viết theo tên, nội dung.
- Xem bài viết: Usecase cho phép người dùng xem chi tiết một bài viết cụ thể.
- Bình luận: Usecase cho phép người dùng bình luận vào một bài viết.
- Theo dõi người dùng: Usecase cho phép người dùng theo dõi một người dùng khác
- Cập nhật thông tin: Usecase cho phép người dùng cập nhật thông tin tài khoản (họ, tên, mô tả, số điện thoại,...)
- Báo cáo bài viết: Usecase cho phép người dùng báo cáo vi phạm của một bài viết.
- Báo cáo bình luận: Usecase cho phép người dùng báo cáo vi phạm của một bình luận.
- Truy cập trang quản trị: Usecase cho phép quản trị viên và kiểm duyệt viên truy cập vào trang quản trị.
- Quản lý bài viết: Usecase cho phép quản trị viên và kiểm duyệt viên thực hiện xem, tìm kiếm, duyệt, từ chối, xóa bài viết.
- Quản lý danh mục: Usecase cho phép quản trị viên thực hiện xem, tìm kiếm, thêm, xóa, sửa danh mục.
- Quản lý báo cáo: Usecase cho phép quản trị viên và kiểm duyệt viên thực hiện xem, từ chối, xử lý báo cáo bình luận/bài viết.
- Quản lý người dùng: Usecase cho phép quản trị viên thực hiện tìm kiếm, xem, khóa tài khoản người dùng.

2.2.2. Data Flow Diagram



Hình 3. Data Flow Diagram của hệ thống



Hình 4: Bảng cơ sở dữ liệu

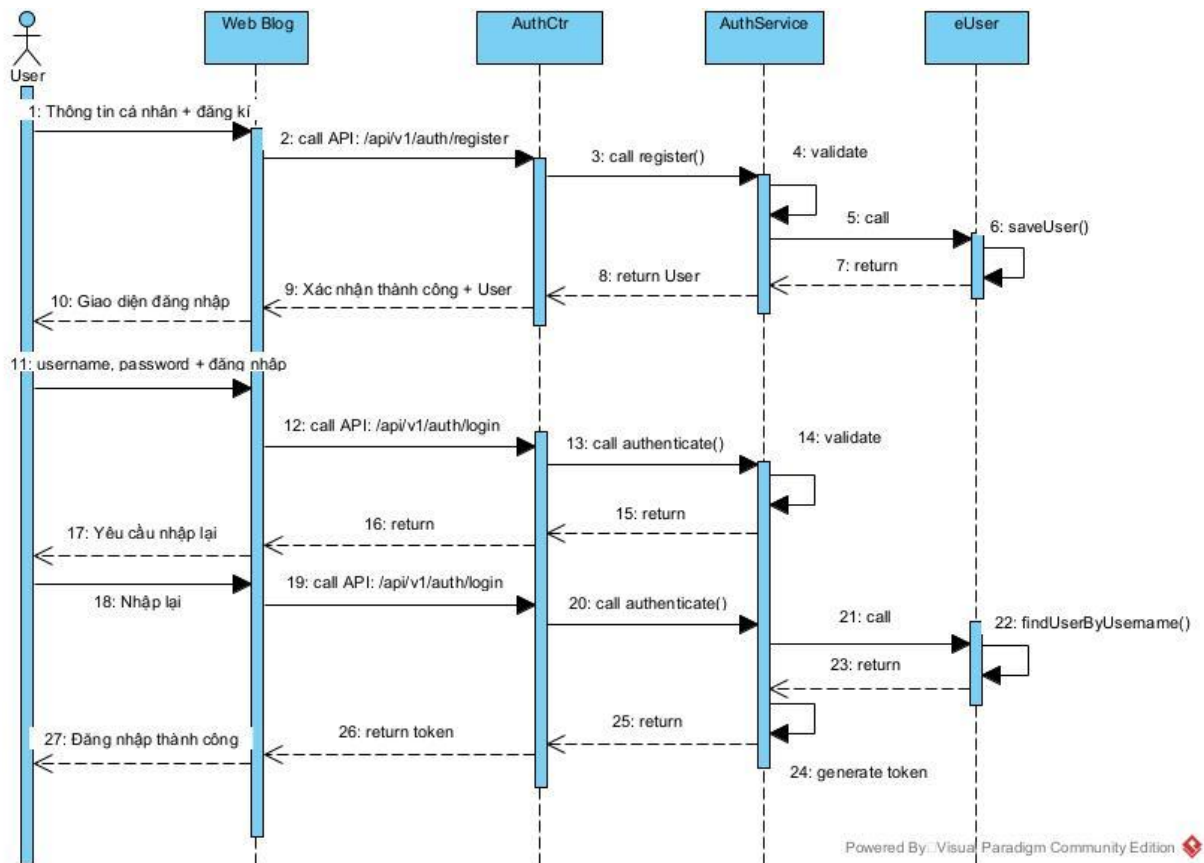
Tên bảng	Mô tả thuộc tính
Category: Danh mục	<ul style="list-style-type: none"> - Id: Mã danh mục - Name: Tên danh mục
Post: Bài viết	<ul style="list-style-type: none"> - Id: mã bài viết - Content: Nội dung bài viết - Description: Mô tả ngắn - Title: tiêu đề - Status: trạng thái duyệt - Thumbnail_url: đường dẫn của ảnh thumbnail - Views: lượt xem - Created_time: thời gian tạo - Modified_Time: thời gian chỉnh sửa - AuthorId: Mã tác giả - CategoryId: Mã người dùng
User: Người dùng	<ul style="list-style-type: none"> - Id: mã người dùng - Firstname: tên - Lastname: họ

	<ul style="list-style-type: none"> - Username: tên đăng nhập - Displayname: tên hiển thị - Phone: số điện thoại - Email: email - Introduction: giới thiệu bản thân - Avatar_url: đường dẫn ảnh đại diện - Password: mật khẩu
Follow: Theo dõi	<ul style="list-style-type: none"> - Id: mã lượt theo dõi - Created_time: thời gian bắt đầu theo dõi - FollowerId: người theo dõi - FollowedId: người được theo dõi
Roles: vai trò người dùng	<ul style="list-style-type: none"> - Id: mã vai trò - Name: tên vai trò (User, Admin, Mod)
User_Roles: Bảng liên kết người dùng với vai trò	<ul style="list-style-type: none"> - RoleId: mã vai trò - UserId: mã người dùng
Notification: Thông báo	<ul style="list-style-type: none"> - Id: mã thông báo - Title: tiêu đề thông báo - Content: nội dung thông báo - Redirect_url: đường dẫn khi bấm vào thông báo - Type: loại thông báo - Created_time: thời gian tạo thông báo - ReceiverId: mã người nhận thông báo
PostVote: Đánh giá bài viết	<ul style="list-style-type: none"> - Id: mã lượt vote - Type: loại vote (ví dụ: upvote/downvote) - AuthorId: mã người vote - PostId: mã bài viết được vote
Comment: Bình luận	<ul style="list-style-type: none"> - Id: mã bình luận - Content: nội dung bình luận

	<ul style="list-style-type: none"> - Created_time: thời gian tạo - Modified_time: thời gian chỉnh sửa - PostId: mã bài viết được bình luận - AuthorId: mã người bình luận - ParentId: mã bình luận cha
ReportComment: Báo cáo bình luận	<ul style="list-style-type: none"> - Id: mã báo cáo - Reason: lý do báo cáo - Status: trạng thái xử lý - Created_time: thời gian tạo báo cáo - CommentId: mã bình luận bị báo cáo (liên kết đến Comment) - AuthorId: mã người báo cáo (liên kết đến User)
ReportPost: Báo cáo bài viết	<ul style="list-style-type: none"> - Id: mã báo cáo - Reason: lý do báo cáo - Status: trạng thái xử lý - Created_time: thời gian tạo báo cáo - AuthorId: mã người báo cáo - PostId: mã bài viết bị báo cáo

2.4. Sequence Diagram

2.4.1. Đăng kí, đăng nhập



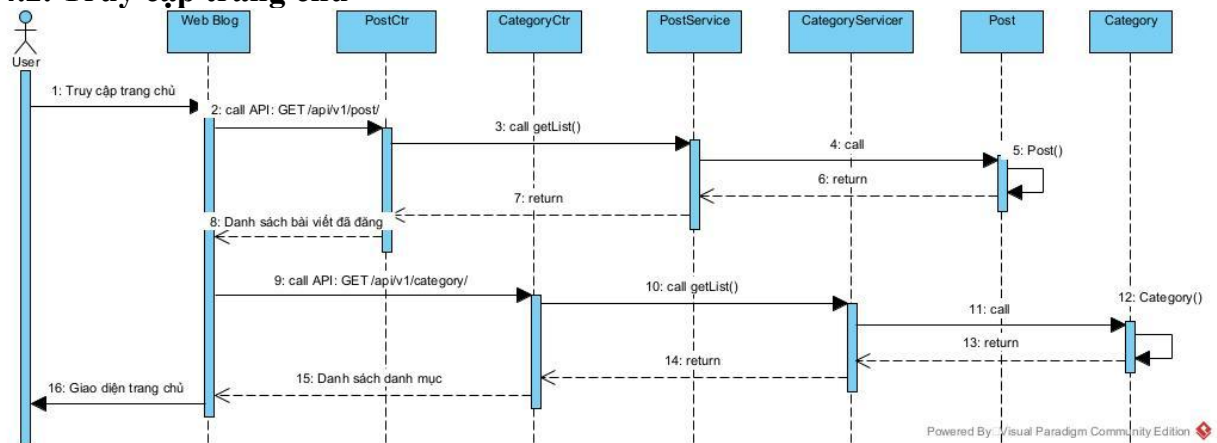
Hình 2.4.1. Sequence Diagram chức năng đăng kí, đăng nhập

Mô tả:

1. User gửi thông tin cá nhân và đăng ký đến Web Blog.
2. Web Blog gọi API /api/v1/auth/register tới AuthCtr.
3. AuthCtr gọi phương thức register() trên AuthService.
4. AuthService thực hiện kiểm tra dữ liệu (validate).
5. AuthService gọi phương thức để lưu thông tin.
6. eUser lưu thông tin người dùng (saveUser()).
7. eUser trả về kết quả lưu.
8. AuthService trả về thông tin User đã đăng ký thành công.
9. AuthCtr xác nhận đăng ký thành công và trả về thông tin User.
10. Web Blog giao kết quả đăng ký thành công cho User.
11. User gửi username, password và yêu cầu đăng nhập tới Web Blog.
12. Web Blog gọi API /api/v1/auth/login tới AuthCtr.
13. AuthCtr gọi phương thức authenticate() trên AuthService.
14. AuthService kiểm tra thông tin đăng nhập (validate).
15. AuthService trả về kết quả kiểm tra.
16. AuthCtr trả về thông tin đăng nhập.
17. User gửi yêu cầu đăng nhập lại tới Web Blog.
18. Web Blog gọi API /api/v1/auth/login tới AuthCtr.

19. AuthCtr gọi lại authenticate() trên AuthService.
20. AuthService gọi phương thức để tìm User theo username.
21. eUser thực hiện tìm kiếm (findUserByUsername()).
22. eUser trả về thông tin User.
23. AuthService trả về kết quả.
24. AuthService tạo token (generate token).
25. AuthService trả về token.
26. AuthCtr trả về token.
27. Website thông báo đăng nhập thành công.

2.4.2. Truy cập trang chủ

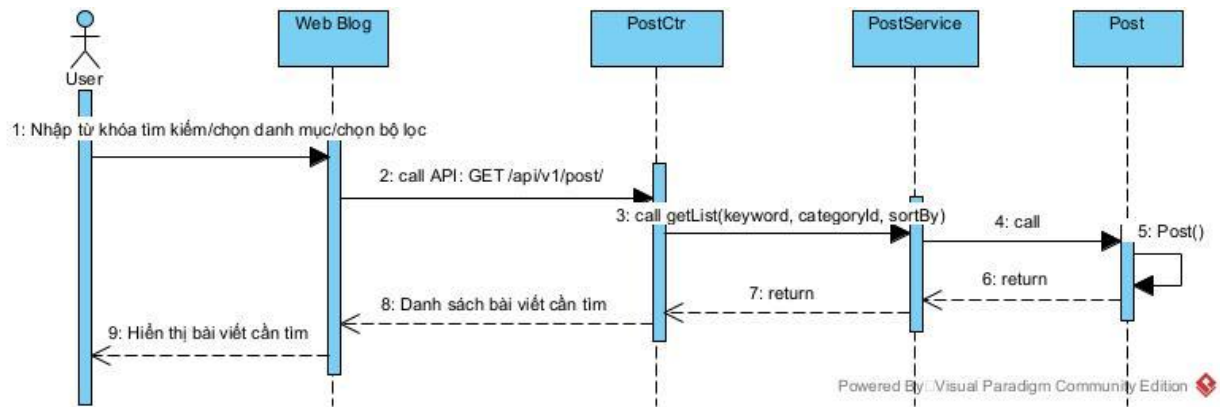


Hình 2.4.2. Sequence Diagram chức năng truy cập trang chủ

Mô tả:

1. User tạo yêu cầu truy cập trang chủ đến Web Blog.
2. Web Blog gọi API: GET /api/v1/post tới PostCtr.
3. PostCtr gọi getList() trên PostService.
4. PostService gọi đến Post yêu cầu khởi tạo
5. PostService khởi tạo Post.
6. Post trả về kết quả.
7. PostService trả về kết quả danh sách bài viết.
8. PostCtr trả về kết quả danh sách bài viết.
9. Web Blog gọi API: GET /api/v1/category tới CategoryCtr.
10. CategoryCtr gọi getList() trên CategoryService.
11. CategoryService gọi đến Category yêu cầu khởi tạo.
12. CategoryService khởi tạo Category
13. Category trả về kết quả.
14. CategoryService trả về kết quả.
15. CategoryCtr trả về kết quả.
16. Web hiển thị giao diện trang chủ

2.4.3. Tìm kiếm bài viết

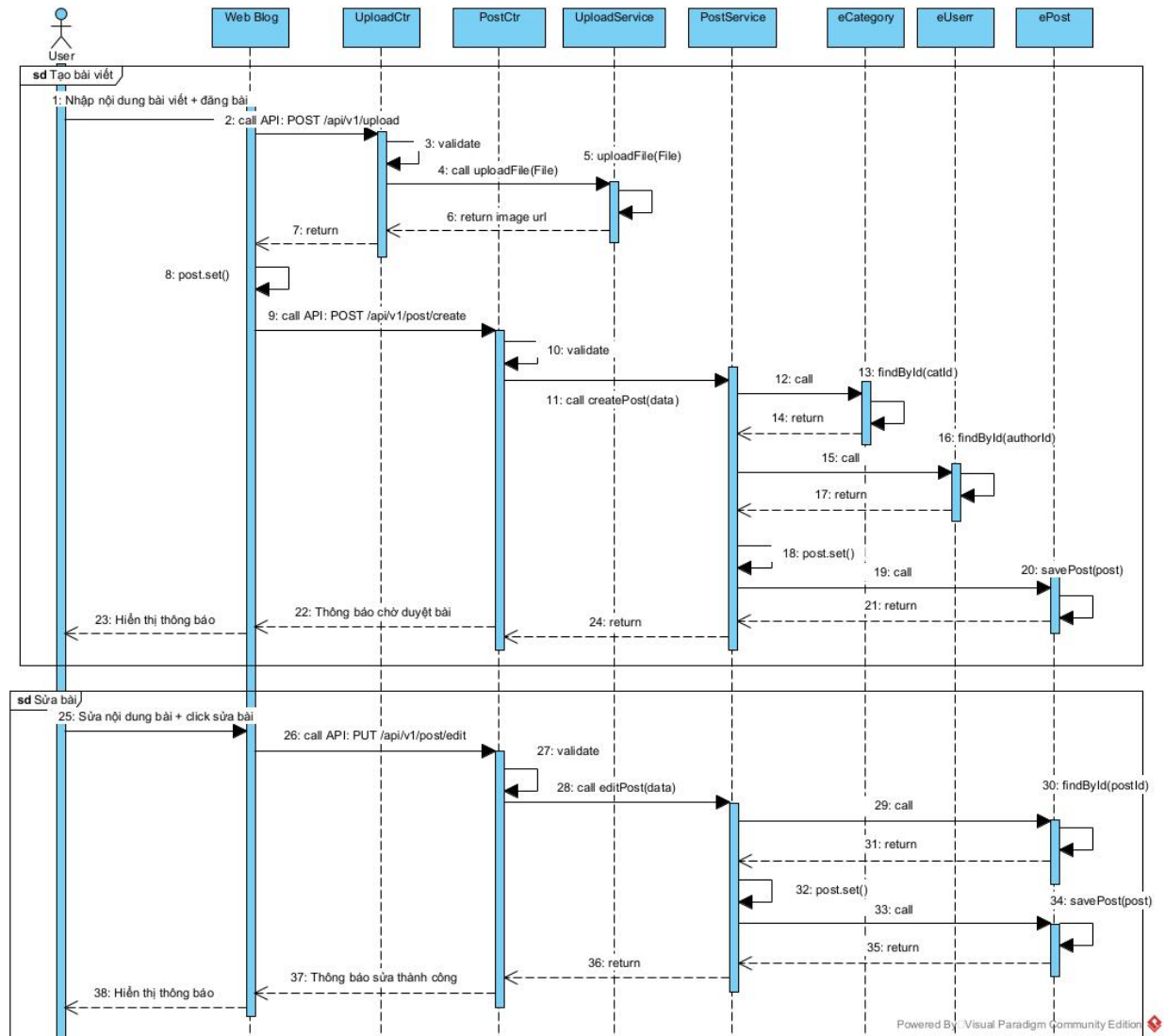


Hình 2.4.3. Sequence Diagram chức năng truy cập trang chủ

Mô tả:

1. User nhập từ khóa tìm kiếm/chọn danh mục/chọn bộ lọc.
2. Web Blog gọi API: GET /api/v1/post/ tới PostCtr.
3. PostCtr gọi getList(keyword, categoryId, sortBy) trên PostService.
4. PostService gọi Post yêu cầu khởi tạo
5. PostService khởi tạo Post.
6. Post trả về kết quả.
7. PostService trả về kết quả danh sách Post.
8. PostCtr trả về kết quả danh sách Post.
9. Website hiển thị kết quả

2.4.4. Tạo/Sửa/Xóa bài viết

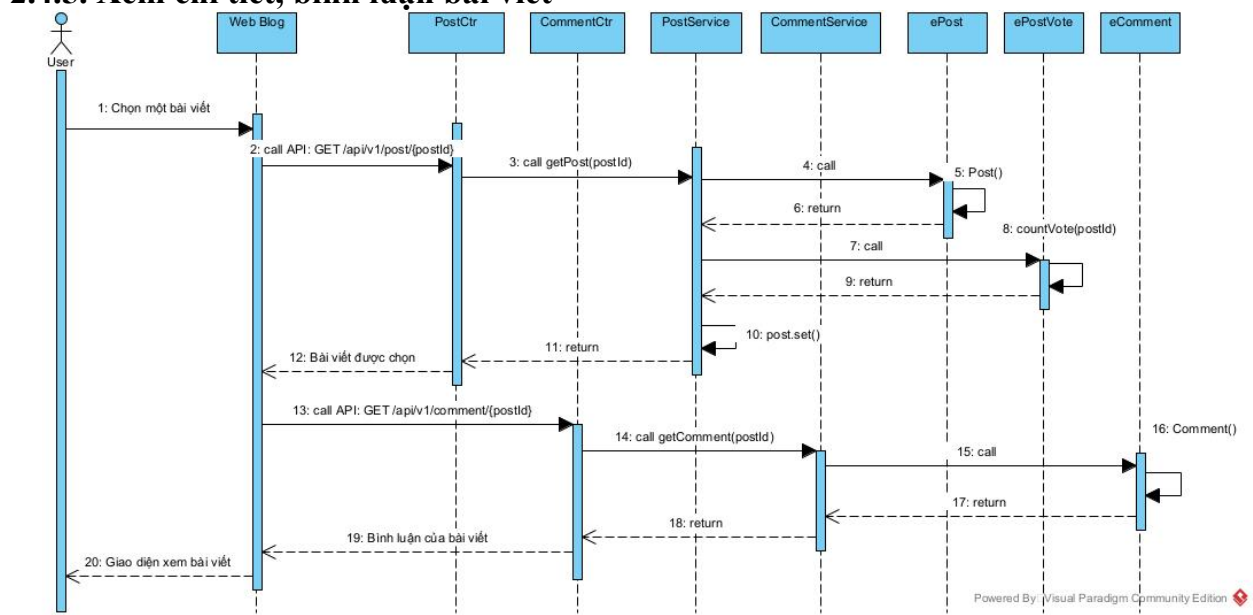


Hình 2.4.4. Sequence Diagram chức năng tạo bài

1. User nhập nội dung bài viết và đăng bài.
2. Web Blog gọi API: POST /api/v1/upload tới UploadCtr.
3. UploadCtr thực hiện kiểm tra dữ liệu (validate).
4. UploadCtr gọi uploadFile(file).
5. UploadService gọi uploadFile(file) thực hiện upload file ảnh.
6. UploadService trả về đường dẫn của ảnh.
7. UploadCtr trả về kết quả.
8. Post set thuộc tính thumbnailUrl.
9. Web Blog gọi API: POST /api/v1/post/create tới PostCtr.
10. PostCtr thực hiện kiểm tra dữ liệu (validate).
11. PostCtr gọi createPost(data) thực hiện tạo bài viết.
12. PostService gọi eCategory yêu cầu tìm
13. eCategory lấy danh mục theo id

14. eCategory trả về kết quả.
15. PostService gọi eUser yêu cầu tìm
16. eUser lấy user là tác giả bài viết.
17. eUser trả về kết quả.
18. Post.setAuthor(), Post.setCategory()
19. PostService gọi hàm lưu của ePost
20. ePost lưu bài viết (savePost()).
21. ePost trả về kết quả là bài viết mới tạo.
22. PostService trả về kết quả.
23. PostCtr trả về thông báo chờ duyệt bài.
24. Website hiển thị thông báo
25. User sửa nội dung bài và click sửa bài viết.
26. Web Blog gọi API: PUT /api/v1/post/edit tới PostCtr.
27. PostCtr thực hiện kiểm tra dữ liệu (validate).
28. PostCtr gọi editPost(data) thực hiện sửa bài.
29. PostService gọi ePost để lấy bài lên.
30. ePost gọi findById(postId).
31. ePost trả bài viết về cho PostService
32. Post được trả về set các thuộc tính thay đổi
33. PostService gọi ePost thực hiện lưu bài viết.
34. ePost thực hiện lưu bài viết mới được chỉnh sửa.
35. ePost trả về kết quả
36. PostService trả về kết quả
37. PostCtr trả về thông báo sửa thành công
38. Website hiển thị thông báo.

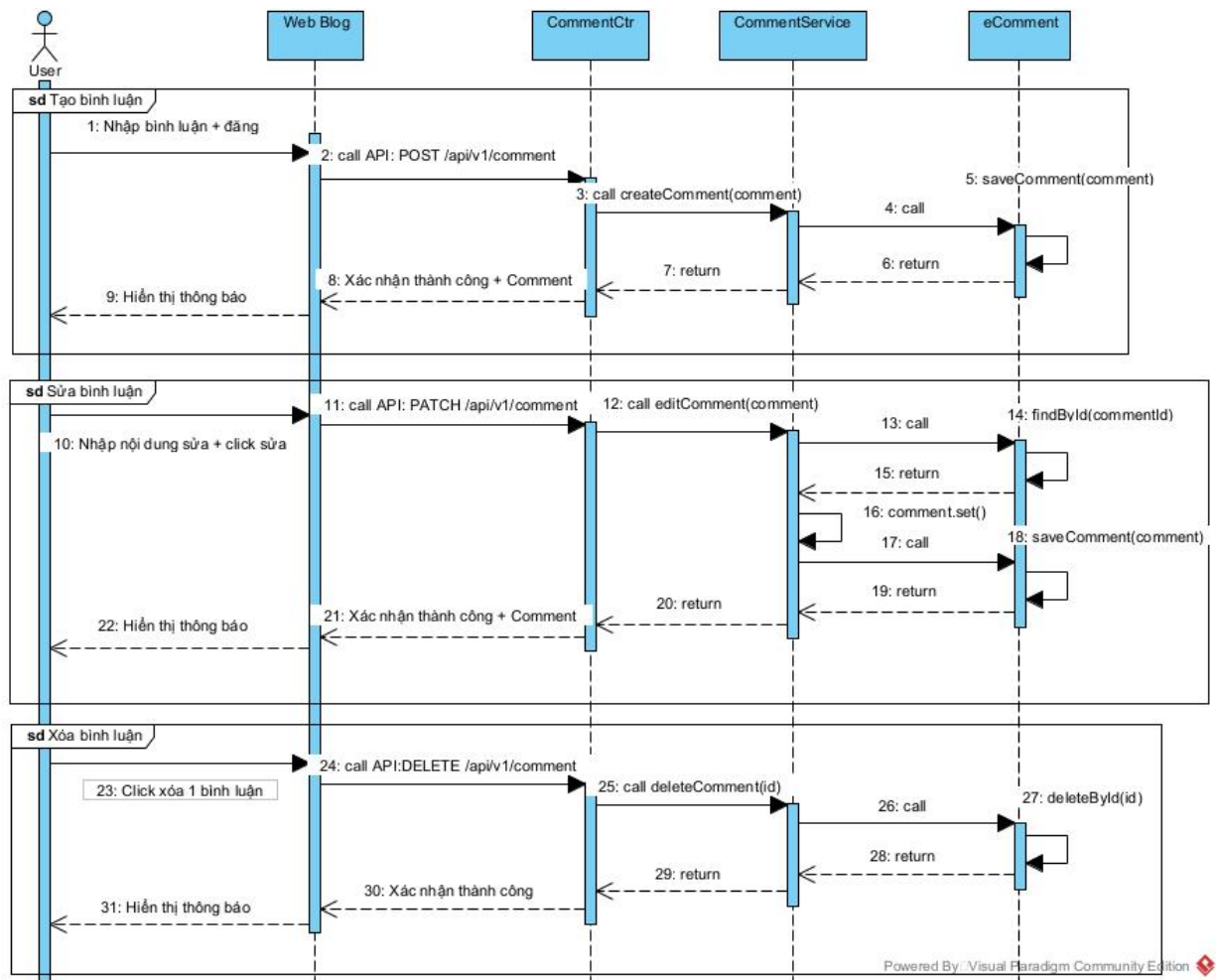
2.4.5. Xem chi tiết, bình luận bài viết



Hình 2.4.5. Sequence Diagram chức năng xem chi tiết

1. User chọn một bài viết.
2. Web Blog gọi API: GET /api/v1/post/postId tới PostCtr.
3. PostCtr gọi getPost(postId) trên PostService lấy bài viết theo id.
4. PostService gọi ePost thực hiện lấy bài viết.
5. ePost khởi tạo bài viết.
6. ePost trả về kết quả
7. PostService gọi countVote(postId) trên ePostVote .
8. ePostVote thực hiện lấy số vote của bài viết.
9. ePostVote trả về kết quả
10. Post set thuộc tính countVote
11. PostService trả về kết quả bài viết.
12. PostCtr trả về bài viết được chọn.
13. Web Blog gọi API: GET /api/v1/comment/postId tới CommentCtr.
14. CommentCtr gọi getComment(postId) trên CommentService.
15. CommentService gọi eComment để lấy danh sách Comment.
16. eComment khởi tạo các Comment
17. eComment trả về kết quả
18. CommentService trả về kết quả.
19. CommentCtr trả về danh sách bình luận của bài viết.
20. Website hiển thị giao diện xem bài viết.

2.4.6. Tạo/Sửa/Xóa bình luận

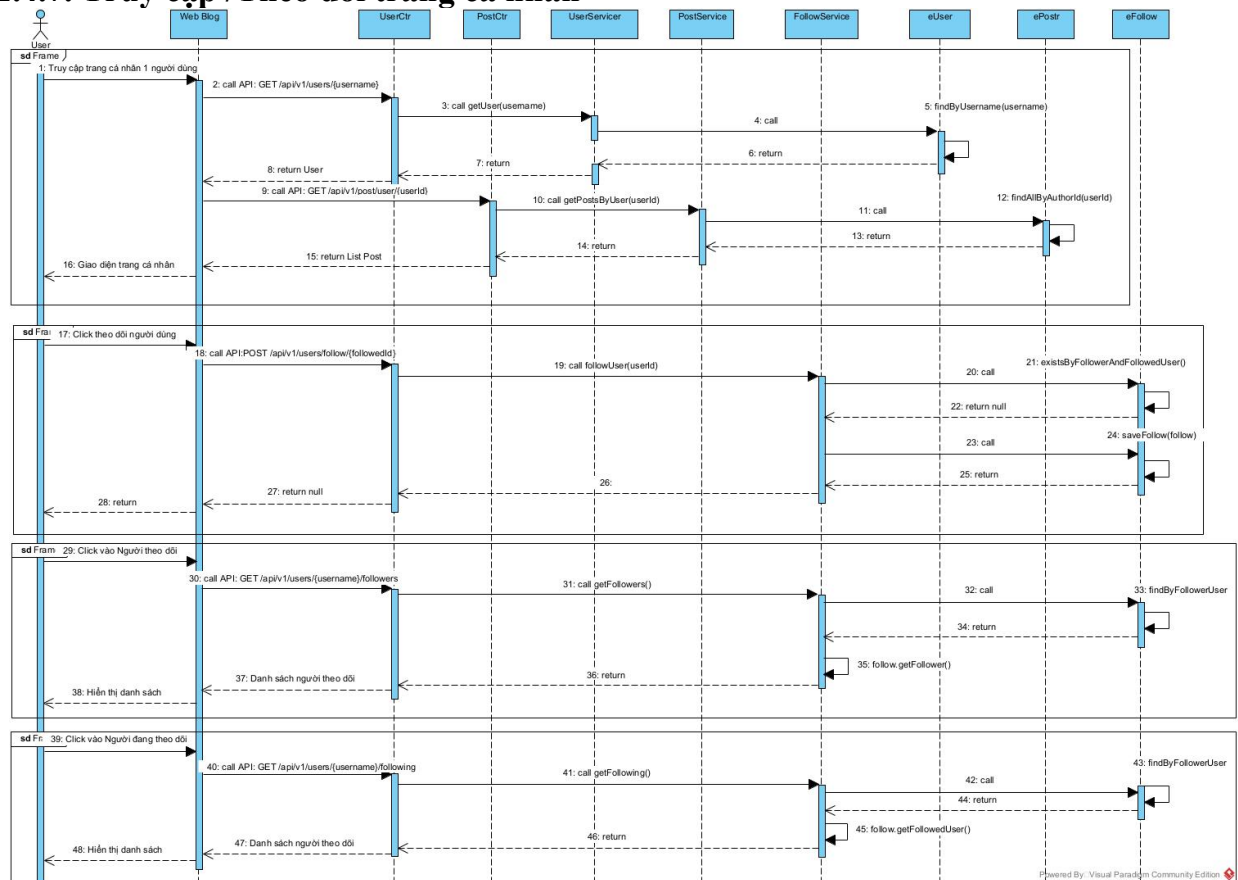


Hình 2.4.6. Sequence Diagram chức năng bình luận

1. User nhập bình luận và đăng bình luận.
2. Web Blog gọi API: POST /api/v1/comment tới CommentCtr.
3. CommentCtr gọi createComment(comment) trên CommentService để tạo bình luận.
4. CommentService gọi eComment thực hiện lưu bình luận.
5. eComment lưu bình luận.
6. eComment trả về.
7. CommentService trả về kết quả bình luận đã tạo
8. CommentCtr trả về xác nhận thành công bình luận.
9. Website hiển thị thông báo.
10. User nhập nội dung sửa và click sửa bình luận.
11. Web Blog gọi API: PATCH /api/v1/comment tới CommentCtr.
12. CommentCtr gọi editComment(comment) trên CommentService để sửa bình luận.
13. CommentService gọi eComment để lấy bình luận theo id.
14. eComment khởi tạo bình luận.

15. eComment trả về kết quả.
16. Commenttt trả về set các thuộc tính thay đổi.
17. CommentService gọi eComment thực hiện lưu bình luận.
18. eComment lưu bình luận đã sửa.
19. eComment trả về kết quả.
20. CommentService trả về kết quả bình luận đã sửa.
21. CommentCtr xác nhận thành công bình luận đã sửa.
22. Website hiển thị thông báo.
23. User click xóa bình luận.
24. Web Blog gọi API: DELETE /api/v1/comment tới CommentCtr.
25. CommentCtr gọi deleteComment(id) trên CommentService để xóa bình luận.
26. CommentService gọi eComment thực hiện xóa bình luận theo id.
27. eComment xóa bình luận.
28. eComment trả về kết quả.
29. CommentService trả về kết quả xóa bình luận.
30. CommentCtr xác nhận thành công.
31. Website hiển thị thông báo.

2.4.7. Truy cập /Theo dõi trang cá nhân



Hình 2.4.7. Sequence Diagram chức năng trang cá nhân

SD Lấy thông tin người dùng

1. User thử lấy thông tin người dùng.
2. Web Blog gọi API: GET /api/v1/users/username tới UserCtr.
3. UserCtr gọi getUser(username) trên UserService để lấy thông tin người dùng.
4. UserService gọi eUser thực hiện tìm người dùng theo username.
5. eUser khởi tạo thông tin người dùng.
6. eUser trả về kết quả.
7. UserService trả về kết quả thông tin người dùng.
8. UserCtr xác nhận thành công thông tin người dùng.
9. Website hiển thị thông tin người dùng.

SD Theo dõi người dùng

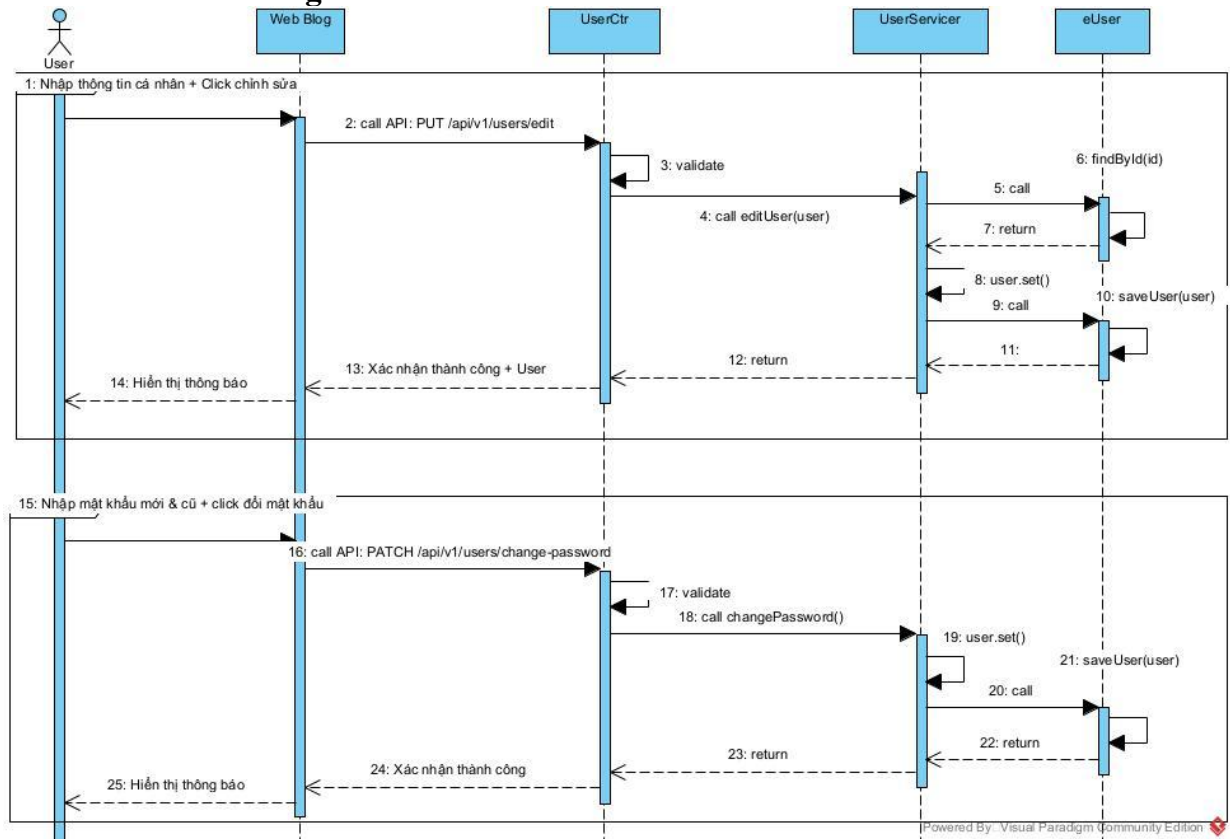
10. User click theo dõi người dùng
11. Web Blog gọi API: POST /api/v1/users/follow/beingFollowed tới UserCtr.
12. UserCtr gọi followUser(followed) trên UserService để thực hiện theo dõi.
13. UserService gọi FollowService để kiểm tra và xử lý theo dõi.
14. FollowService gọi eFollow thực hiện kiểm tra người theo dõi và người được theo dõi.
15. eFollow khởi tạo thông tin theo dõi.
16. eFollow trả về kết quả.
17. FollowService gọi eFollow thực hiện lưu thông tin theo dõi.
18. eFollow lưu thông tin theo dõi (save()).eFollow trả về kết quả.
FollowService trả về kết quả theo dõi.
UserService trả về kết quả theo dõi.
19. UserCtr xác nhận thành công theo dõi.
20. Website hiển thị thông báo.

SD Lấy danh sách người theo dõi/người được theo dõi

21. User click vào danh sách người theo dõi.
22. Web Blog gọi API: GET /api/v1/users/username/followers tới UserCtr.
23. UserCtr gọi getFollowers() trên UserService để lấy danh sách người theo dõi.
24. UserService gọi FollowService để lấy danh sách.
25. FollowService gọi eFollow thực hiện tìm danh sách người theo dõi.
26. eFollow khởi tạo danh sách người theo dõi.
27. eFollow trả về kết quả.
28. FollowService trả về kết quả danh sách người theo dõi.
29. UserService trả về kết quả danh sách người theo dõi.
30. UserCtr xác nhận thành công danh sách người theo dõi.
31. Website hiển thị danh sách người theo dõi.
32. User click vào danh sách người được theo dõi.
33. Web Blog gọi API: GET /api/v1/users/username/following tới UserCtr.

34. UserCtr gọi getFollowing() trên UserService để lấy danh sách người được theo dõi.
35. UserService gọi FollowService để lấy danh sách.
36. FollowService gọi eFollow thực hiện tìm danh sách người được theo dõi.
37. eFollow khởi tạo danh sách người được theo dõi.
38. eFollow trả về kết quả.
39. FollowService trả về kết quả danh sách người được theo dõi.
40. UserService trả về kết quả danh sách người được theo dõi.
41. UserCtr xác nhận thành công danh sách người được theo dõi.
42. Website hiển thị danh sách người được theo dõi.

2.4.8. Chỉnh sửa thông tin cá nhân



Hình 2.4.8. Sequence Diagram chức năng chỉnh sửa thông tin

SD Chỉnh sửa thông tin người dùng

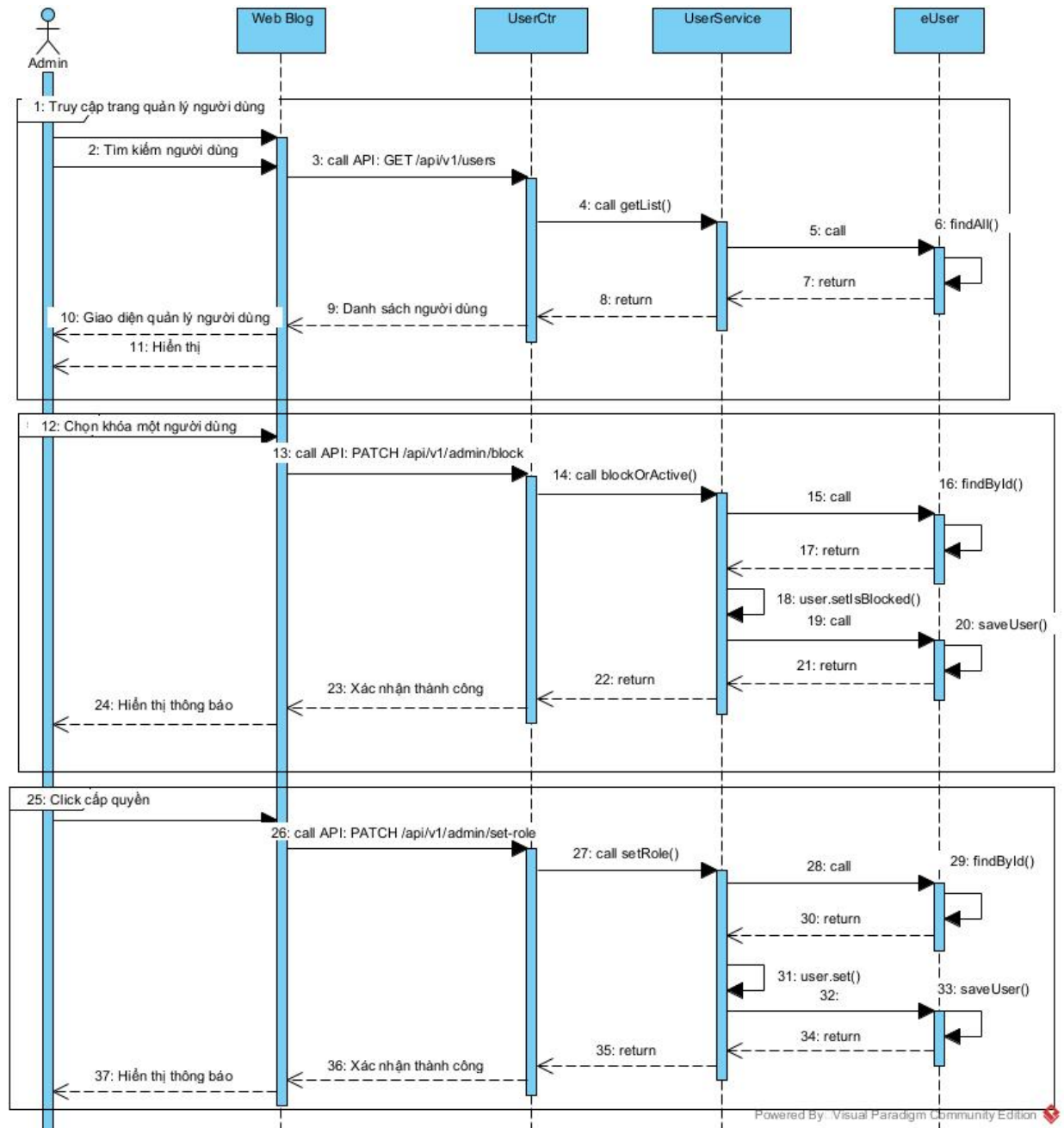
1. User nhập thông tin cá nhân và click chỉnh sửa.
2. Web Blog gọi API: PUT /api/v1/user/edit tới UserCtr.
3. UserCtr thực hiện kiểm tra dữ liệu (validate).
4. UserCtr gọi editUser(user) trên UserService để chỉnh sửa thông tin.
5. UserService gọi eUser thực hiện tìm người dùng theo id.

-
6. eUser khởi tạo người dùng.
 7. eUser trả về kết quả.
 8. User set các thuộc tính chỉnh sửa.
 9. UserService gọi eUser thực hiện lưu thông tin người dùng.
 10. eUser lưu thông tin người dùng (saveUser()).
 11. eUser trả về kết quả.
 12. UserService trả về kết quả thông tin đã chỉnh sửa.
 13. UserCtr xác nhận thành công thông tin người dùng.
 14. Website hiển thị thông báo.

SD Đổi mật khẩu

15. User nhập mật khẩu mới và click đổi mật khẩu.
16. Web Blog gọi API: PATCH /api/v1/users/change-password tới UserCtr.
17. UserCtr thực hiện kiểm tra dữ liệu (validate).
18. UserCtr gọi changePassword() trên UserService để đổi mật khẩu.
19. User set mật khẩu mới
20. UserService gọi eUser thực hiện lưu thông tin người dùng.
21. eUser lưu thông tin người dùng (saveUser()).
22. eUser trả về kết quả.
23. UserService trả về kết quả mật khẩu đã đổi.
24. UserCtr xác nhận thành công đổi mật khẩu.
25. Website hiển thị thông báo.

2.4.9. Quản lý người dùng



Hình 2.4.9. Sequence Diagram chức năng quản lý người dùng

SD Lấy danh sách người dùng

1. Admin truy cập trang quản lý người dùng.
2. Web Blog gọi API: GET /api/v1/users tới UserCtr.
3. UserCtr gọi getList() trên UserService để lấy danh sách người dùng.
4. UserService gọi eUser thực hiện lấy tất cả người dùng.
5. eUser khởi tạo danh sách người dùng.
6. eUser trả về kết quả.
7. UserService trả về kết quả danh sách người dùng.
8. UserCtr xác nhận thành công danh sách người dùng.

9. Website hiển thị danh sách người dùng.

SD Khóa người dùng

10. Admin chọn khóa người dùng.

11. Web Blog gọi API: PATCH /api/v1/admin/block tới UserCtr.

12. UserCtr gọi blockActive() trên UserService để khóa người dùng.

13. UserService gọi eUser thực hiện tìm người dùng theo id.

14. eUser khởi tạo người dùng.

15. eUser trả về kết quả.

16. UserService gọi User để cập nhật trạng thái khóa.

17. User.setIsBlocked().

18. UserService gọi eUser thực hiện lưu thông tin người dùng.

19. eUser lưu thông tin người dùng (saveUser()).

20. eUser trả về kết quả.

21. UserService trả về kết quả khóa người dùng.

22. UserCtr xác nhận thành công khóa người dùng.

23. Website hiển thị thông báo.

SD Gán quyền

24. Admin click gán quyền.

Web Blog gọi API: PATCH /api/v1/admin/set-role tới UserCtr.

25. UserCtr gọi setRole() trên UserService để gán quyền.

26. UserService gọi eUser thực hiện tìm người dùng theo id.

27. eUser khởi tạo người dùng.

28. eUser trả về kết quả.

29. UserService gọi User để cập nhật quyền.

30. User.set().

31. UserService gọi eUser thực hiện lưu thông tin người dùng.

32. eUser lưu thông tin người dùng (saveUser()).

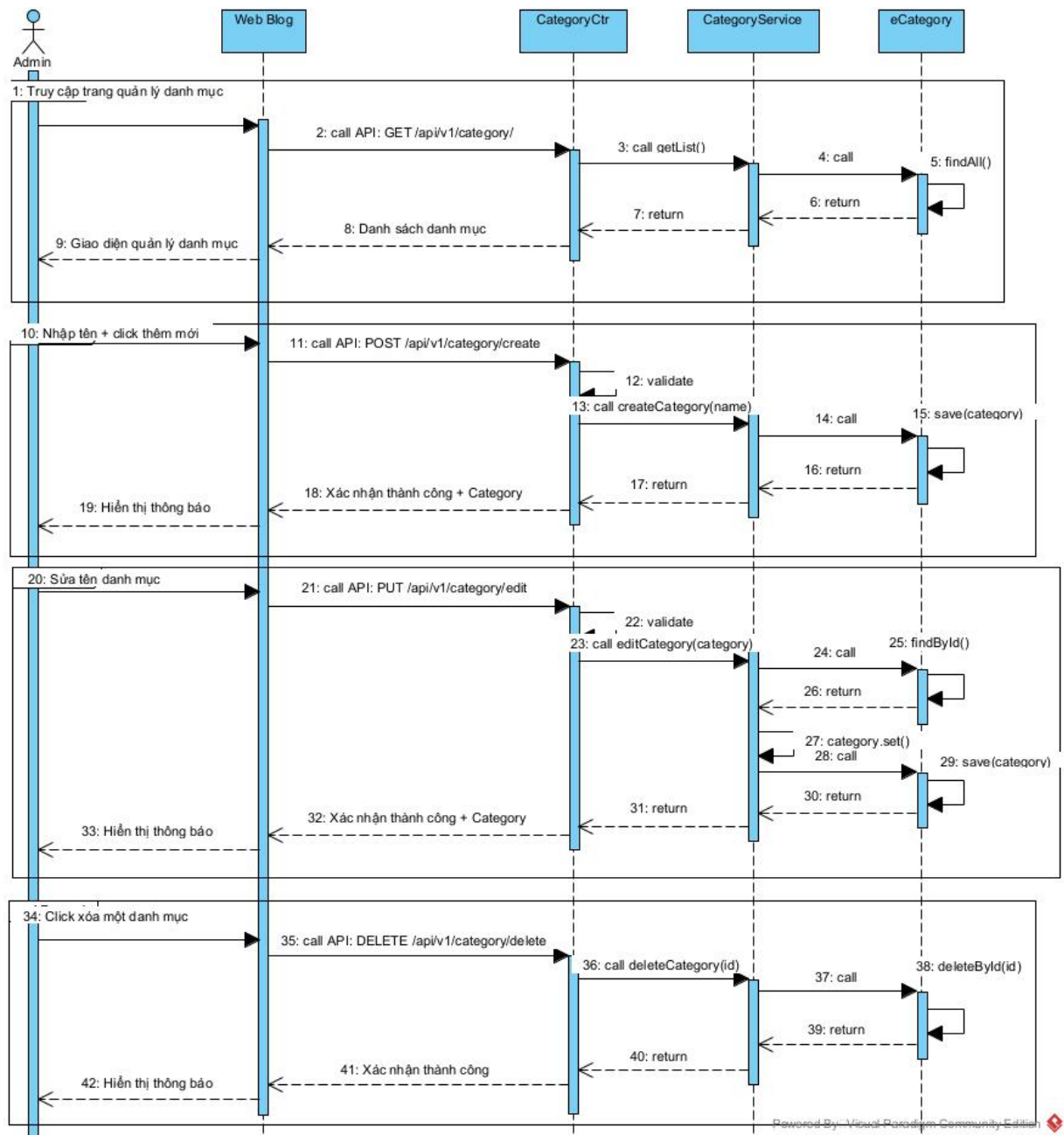
33. eUser trả về kết quả.

34. UserService trả về kết quả gán quyền.

35. UserCtr xác nhận thành công gán quyền.

36. Website hiển thị thông báo.

2.4.10. Quản lý danh mục



Hình 2.4.10. Sequence Diagram chức năng quản lý danh mục

SD Lấy danh sách danh mục

1. Admin thử lấy danh sách danh mục.
2. Web Blog gọi API: GET /api/v1/category tới CategoryCtr.
3. CategoryCtr gọi getList() trên CategoryService để lấy danh sách danh mục.
4. CategoryService gọi eCategory thực hiện lấy tất cả danh mục.
5. eCategory khởi tạo danh sách danh mục.
6. eCategory trả về kết quả.
7. CategoryService trả về kết quả danh sách danh mục.
8. CategoryCtr xác nhận thành công danh sách danh mục.

9. Website hiển thị danh sách danh mục.

SD Tạo danh mục

10. Admin nhập tên và click thêm mục.

11. Web Blog gọi API: POST /api/v1/category/create tới CategoryCtr.

12. CategoryCtr thực hiện kiểm tra dữ liệu (validate).

13. CategoryCtr gọi createCategory(name) trên CategoryService để tạo danh mục.

14. CategoryService gọi eCategory thực hiện lưu danh mục.

15. eCategory khởi tạo danh mục.

16. eCategory trả về kết quả.

17. CategoryService trả về kết quả danh mục đã tạo.

18. CategoryCtr xác nhận thành công danh mục.

19. Website hiển thị thông báo.

SD Sửa danh mục

20. Admin sửa tên danh mục.

21. Web Blog gọi API: PUT /api/v1/category/edit tới CategoryCtr.

22. CategoryCtr thực hiện kiểm tra dữ liệu (validate).

23. CategoryCtr gọi editCategory(category) trên CategoryService để sửa danh mục.

24. CategoryService gọi eCategory thực hiện tìm danh mục theo id.

25. eCategory khởi tạo danh mục.

26. eCategory trả về kết quả.

27. CategoryService gọi Category để cập nhật danh mục.

28. Category.set().

29. CategoryService gọi eCategory thực hiện lưu danh mục.

30. eCategory lưu danh mục (saveCategory()).

31. eCategory trả về kết quả.

32. CategoryService trả về kết quả danh mục đã sửa.

33. CategoryCtr xác nhận thành công danh mục đã sửa.

34. Website hiển thị thông báo.

SD Xóa danh mục

35. Admin click xóa mục danh mục.

36. Web Blog gọi API: DELETE /api/v1/category/delete tới CategoryCtr.

37. CategoryCtr gọi deleteCategory(id) trên CategoryService để xóa danh mục.

38. CategoryService gọi eCategory thực hiện xóa danh mục theo id.

39. eCategory xóa danh mục (deleteById()).

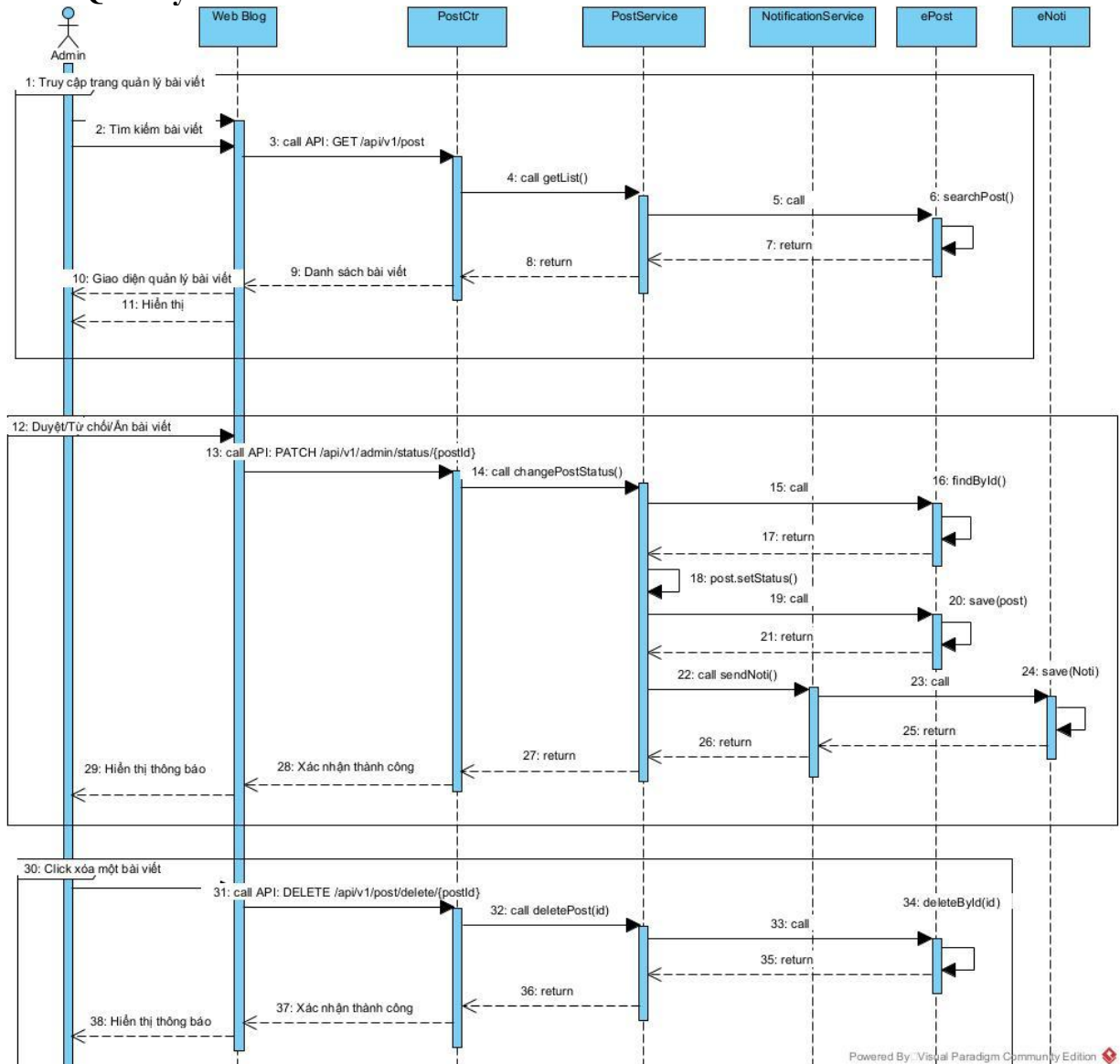
eCategory trả về kết quả.

40. CategoryService trả về kết quả xóa danh mục.

41. CategoryCtr xác nhận thành công.

42. Website hiển thị thông báo.

2.4.11. Quản lý bài viết



Hình 2.4.11. Sequence Diagram chức năng quản lý bài viết

SD Lấy danh sách bài viết

1. Admin thử lấy danh sách bài viết.
2. Web Blog gọi API: GET /api/v1/post tới PostCtr.
3. PostCtr gọi getList() trên PostService để lấy danh sách bài viết.
4. PostService gọi ePost thực hiện tìm kiếm bài viết.
5. ePost khởi tạo danh sách bài viết.
6. ePost trả về kết quả.
7. PostService trả về kết quả danh sách bài viết.
8. PostCtr xác nhận thành công danh sách bài viết.
9. Website hiển thị danh sách bài viết.

SD Thay đổi trạng thái bài viết

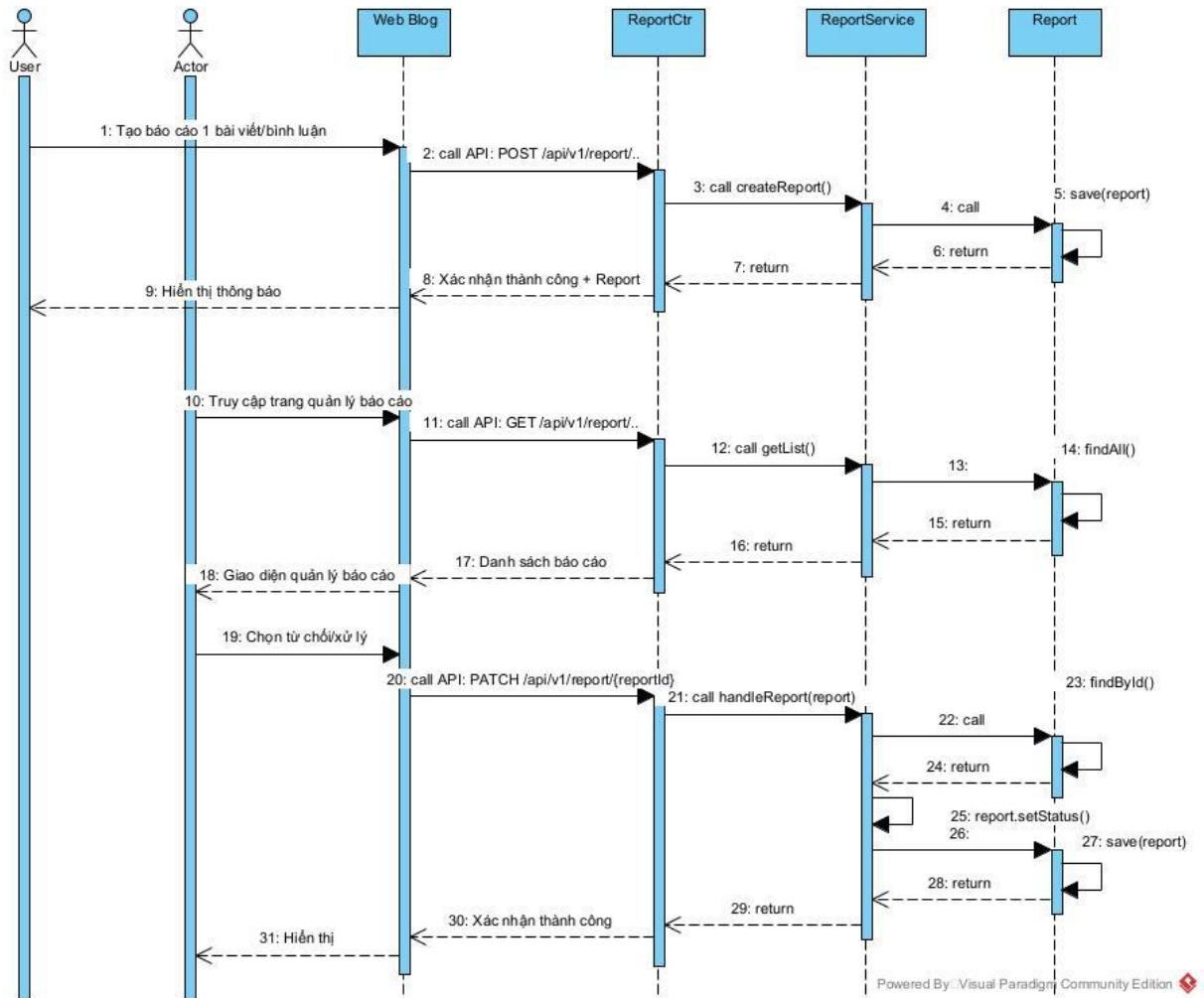
10. Admin chọn thay đổi trạng thái bài viết.

-
11. Web Blog gọi API: PATCH /api/v1/admin/status/postId tới PostCtr.
 12. PostCtr gọi changeStatus() trên PostService để thay đổi trạng thái.
 13. PostService gọi ePost thực hiện tìm bài viết theo id.\
 14. ePost khởi tạo bài viết.
 15. ePost trả về kết quả.
 16. PostService gọi Post để cập nhật trạng thái.
 17. Post.setStatus().
 18. PostService gọi ePost thực hiện lưu bài viết.
 19. ePost lưu bài viết (save()).
 20. ePost trả về kết quả.
 21. PostService gọi NotificationService để gửi thông báo.
 22. NotificationService gọi eNoti thực hiện gửi thông báo.
 23. eNoti khởi tạo thông báo.
 24. eNoti trả về kết quả.
 25. NotificationService trả về kết quả thông báo.
 26. PostService trả về kết quả thay đổi trạng thái.
 27. PostCtr xác nhận thành công thay đổi trạng thái.
 28. Website hiển thị thông báo.

SD Xóa bài viết

29. Admin click xóa mục bài viết.
30. Web Blog gọi API: DELETE /api/v1/post/delete/postId tới PostCtr.
31. PostCtr gọi deletePost(id) trên PostService để xóa bài viết.
32. PostService gọi ePost thực hiện xóa bài viết theo id.
33. ePost xóa bài viết (deleteById()).
34. ePost trả về kết quả.
35. PostService trả về kết quả xóa bài viết.
36. PostCtr xác nhận thành công.
37. Website hiển thị thông báo.

2.4.12. Quản lý báo cáo



Hình 2.4.12. Sequence Diagram chức năng quản lý báo cáo

SD Lấy danh sách bài viết

1. Admin thử lấy danh sách bài viết.
2. Web Blog gọi API: GET /api/v1/post tới ReportCtr.
3. ReportCtr gọi getList() trên ReportService để lấy danh sách bài viết.
4. ReportService gọi ePost thực hiện tìm kiếm bài viết.
5. ePost khởi tạo danh sách bài viết.
6. ePost trả về kết quả.
7. ReportService trả về kết quả danh sách bài viết.
8. ReportCtr xác nhận thành công danh sách bài viết.
9. Website hiển thị danh sách bài viết.

SD Thay đổi trạng thái bài viết

10. Admin chọn thay đổi trạng thái bài viết.
11. Web Blog gọi API: PATCH /api/v1/admin/status/postId tới ReportCtr.
12. ReportCtr gọi changePost() trên ReportService để thay đổi trạng thái.

-
13. ReportService gọi ePost thực hiện tìm bài viết theo id.
 14. ePost khởi tạo bài viết.
 15. ePost trả về kết quả.
 16. ReportService gọi Post để cập nhật trạng thái.
 17. Post.setStatus().
 18. ReportService gọi ePost thực hiện lưu bài viết.
 19. ePost lưu bài viết (save()).
 20. ePost trả về kết quả.
 21. ReportService gọi NotificationService để gửi thông báo.
 22. NotificationService gọi eNoti thực hiện gửi thông báo.
 23. eNoti khởi tạo thông báo.
 24. eNoti trả về kết quả.
 25. NotificationService trả về kết quả thông báo.
 26. ReportService trả về kết quả thay đổi trạng thái.
 27. ReportCtr xác nhận thành công thay đổi trạng thái.
 28. Website hiển thị thông báo.

SD Xóa bài viết

29. Admin click xóa mục bài viết.
30. Web Blog gọi API: DELETE /api/v1/post/delete/postId tới ReportCtr.
31. ReportCtr gọi deletePost(id) trên ReportService để xóa bài viết.
32. ReportService gọi ePost thực hiện xóa bài viết theo id.
33. ePost xóa bài viết (deleteById()).
34. ePost trả về kết quả.
35. ReportService trả về kết quả xóa bài viết.
36. ReportCtr xác nhận thành công.
37. Website hiển thị thông báo.

CHƯƠNG 3. THIẾT KẾ CHI TIẾT

3.1. API xác thực người dùng.

STT	Nhiệm vụ	Http Method	API
1	Đăng ký tài khoản	POST	/api/v1/auth/register
2	Đăng nhập	POST	/api/v1/auth/login
3	Đăng xuất	POST	/api/v1/auth/logout
4	Xác minh email	POST	/api/v1/auth/verify-email
5	Đặt lại mật khẩu	PUT	/api/v1/auth/reset-password
6	Làm mới token	POST	/api/v1/auth/refresh
7	Kiểm tra token	POST	/api/v1/auth/introspect

3.2. API bài viết

STT	Nhiệm vụ	Http Method	API
1	Tạo bài viết mới	POST	/api/v1/post/create
2	Chỉnh sửa bài viết	PUT	/api/v1/post/edit
3	Xóa bài viết	DELETE	/api/v1/post/delete/{postId}
4	Lấy danh sách bài viết	GET	/api/v1/post/
5	Lấy danh sách bài viết của User cụ thể	GET	/api/v1/post/user/{userId}
6	Lấy chi tiết một bài viết	GET	/api/v1/post/{postId}
7	Thay đổi trạng thái bài viết	PATCH	/api/v1/admin/status/{postId}
8	Like/Dislike bài viết	POST	/api/v1/post/{postId}/{voteType}

3.3. API bình luận

STT	Nhiệm vụ	Http Method	API
1	Tạo một bình luận mới	POST	/api/v1/comment/
2	Xóa một bình luận	DELETE	/api/v1/comment/
3	Chỉnh sửa nội dung của một bình luận	PATCH	/api/v1/comment/
4	Lấy danh sách bình luận cấp 1 của bài viết	GET	/api/v1/comment/{postId}
5	Lấy danh sách bình luận con (trả lời) của một bình luận cha	GET	/api/v1/comment/children/{parentId}

3.4. API danh mục

STT	Nhiệm vụ	Http Method	API
1	Tạo danh mục	POST	/api/v1/category/create
2	Chỉnh sửa danh mục	PUT	/api/v1/category/edit
3	Xóa danh mục	DELETE	/api/v1/category/delete
4	Lấy danh sách danh mục	GET	/api/v1/category/

3.5. API người dùng/admin

STT	Nhiệm vụ	Http Method	API
1	Lấy thông tin cá nhân của người dùng	GET	/api/v1/users/{username}
2	Lấy danh sách người dùng	GET	/api/v1/users/
3	Lấy danh sách người mà username đang theo dõi.	GET	/api/v1/users/{username}/following

5	Lấy danh sách người đang theo dõi username.	GET	/api/v1/users/{username}/followers
6	Theo dõi một người dùng khác	POST	/api/v1/users/follow/{followedId}
7	Hủy theo dõi một người dùng	POST	/api/v1/users/unfollow/{followedId}
8	Cập nhật thông tin cá nhân của người dùng	PUT	/api/v1/users/edit
9	Thay đổi mật khẩu cho người dùng	PATCH	/api/v1/users/change-password
10	Cấp/Hủy quyền kiểm duyệt viên	PATCH	/api/v1/admin/set-role
11	Khóa tài khoản người dùng	PATCH	/api/v1/admin/block

3.6. API báo cáo

STT	Nhiệm vụ	Http Method	API
1	Tạo báo cáo bài viết	POST	/api/v1/report/post/
2	Tạo báo cáo bình luận	POST	/api/v1/report/comment/
3	Lấy danh sách báo cáo bài viết	GET	/api/v1/report/post/
4	Lấy danh sách báo cáo bình luận	GET	/api/v1/report/comment/
5	Xử lý báo cáo bài viết	PATCH	/api/v1/report/post/{reportId}
6	Xử lý báo cáo bình luận	PATCH	/api/v1/report/comment/{reportId}

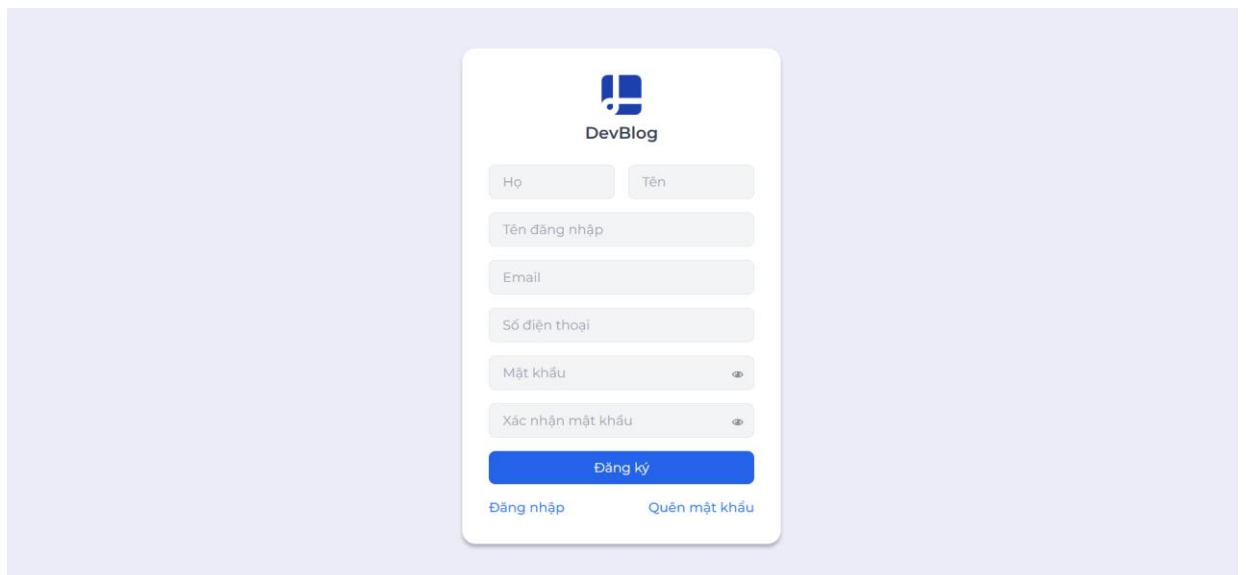
3.7. API khác

STT	Nhiệm vụ	Http Method	API
1	Upload ảnh bài viết lên	POST	/api/v1/upload

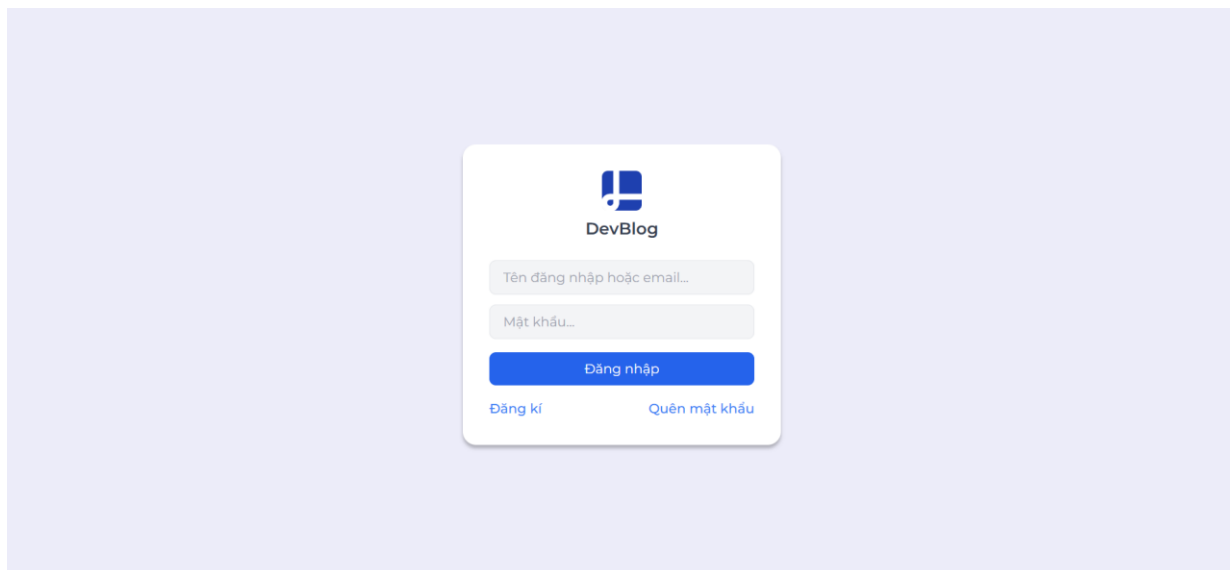
	Cloudinary		
2	Gửi mã otp cho email người dùng	POST	/api/v1/email/send-otp
3	Gửi thông báo cho người dùng	POST	/api/v1/notification
4	Lấy danh sách thông báo của người dùng	GET	/api/v1/notification/{userId}
5	Đánh dấu các thông báo là đã đọc	PATCH	/api/v1/notification/mark-as-read/{userId}

CHƯƠNG 4. KẾT QUẢ

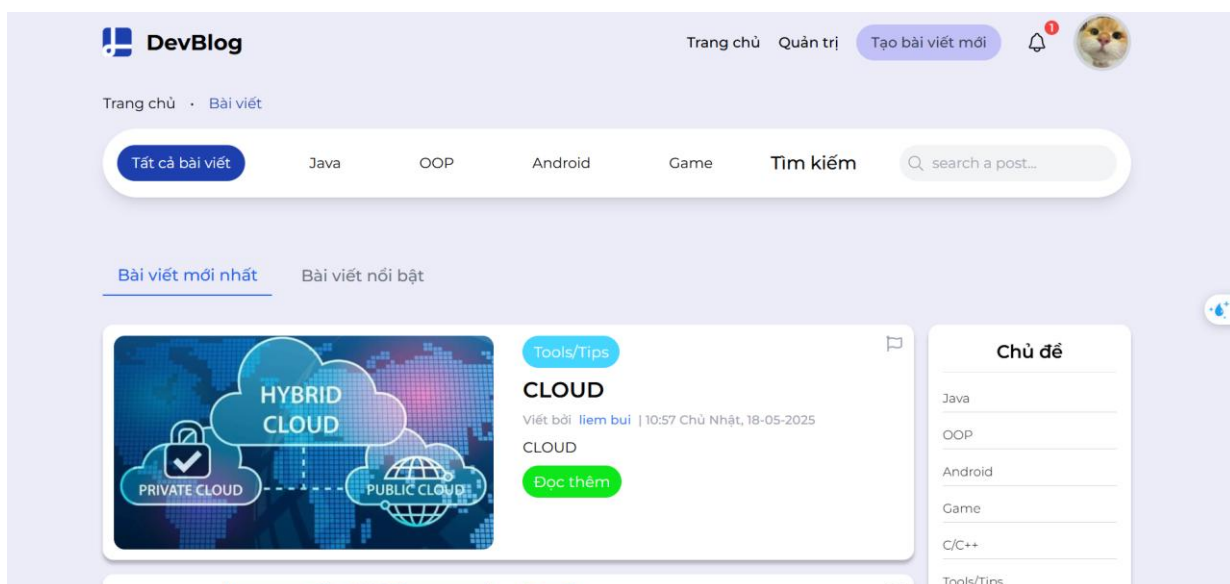
4.1. Giao diện người dùng



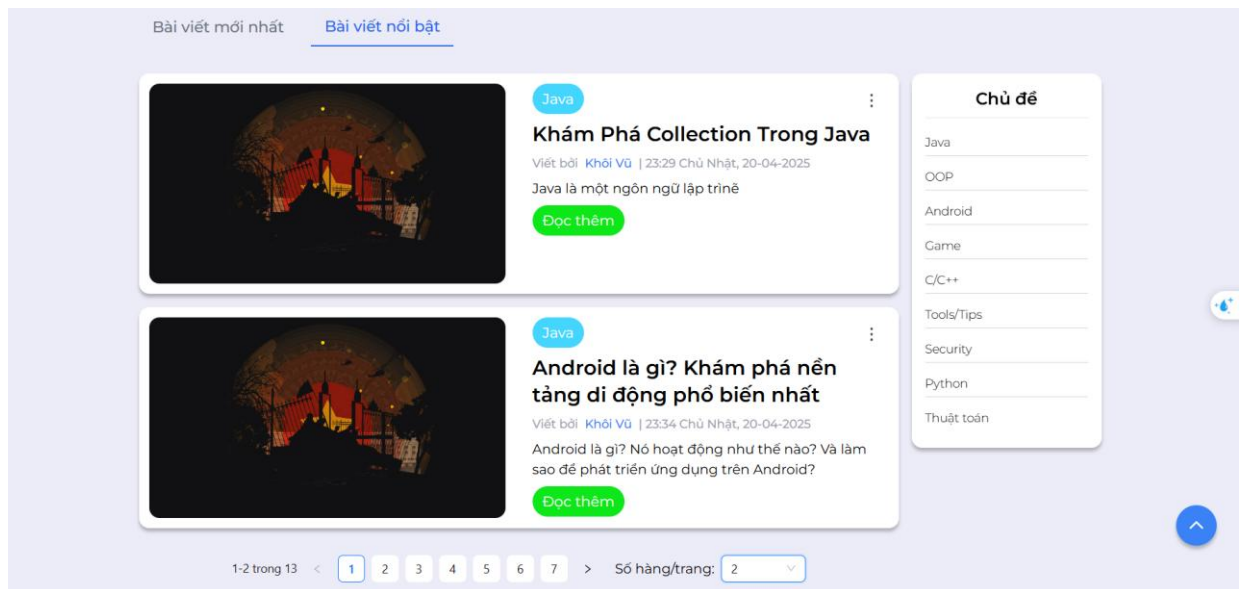
Hình 4.1.1. Giao diện đăng kí



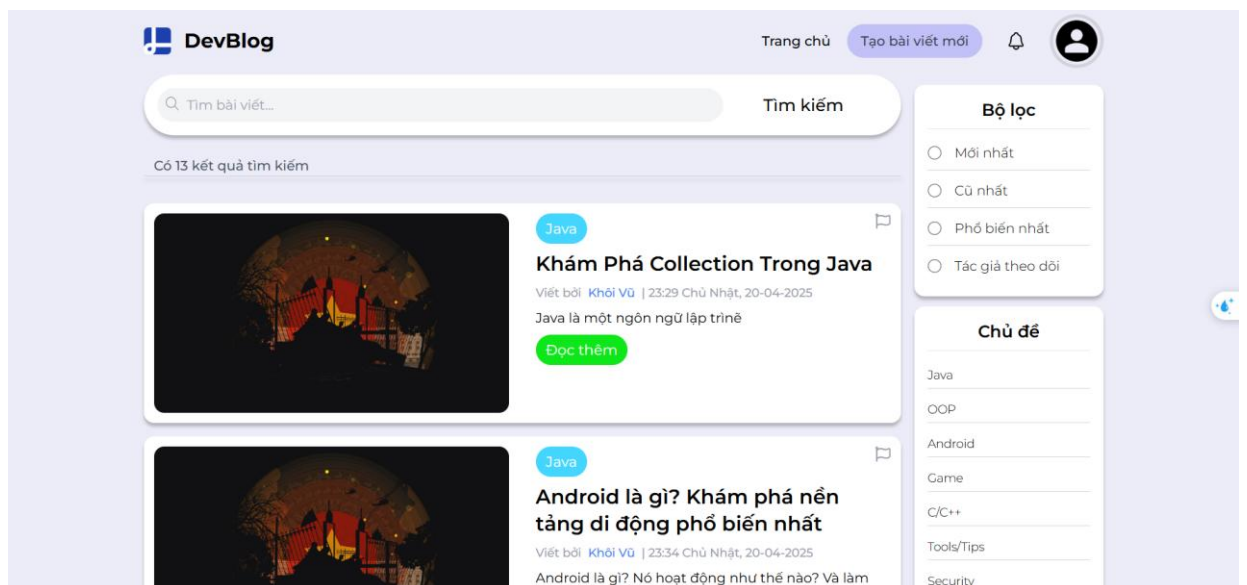
Hình 4.1.2. Giao diện đăng nhập




Hình 4.1.3. Trang chủ (1)



Hình 4.1.4. Trang chủ (2)





Hình 4.1.5. Giao diện tìm kiếm bài viết

 **DevBlog**


Trang chủ

Tạo bài viết mới





Tạo bài viết mới

 Thêm ảnh bìa

Tiêu đề bài viết

Chọn một chủ đề

Chọn danh mục ▾

Mô tả ngắn


Normal


⌵


B


I


U











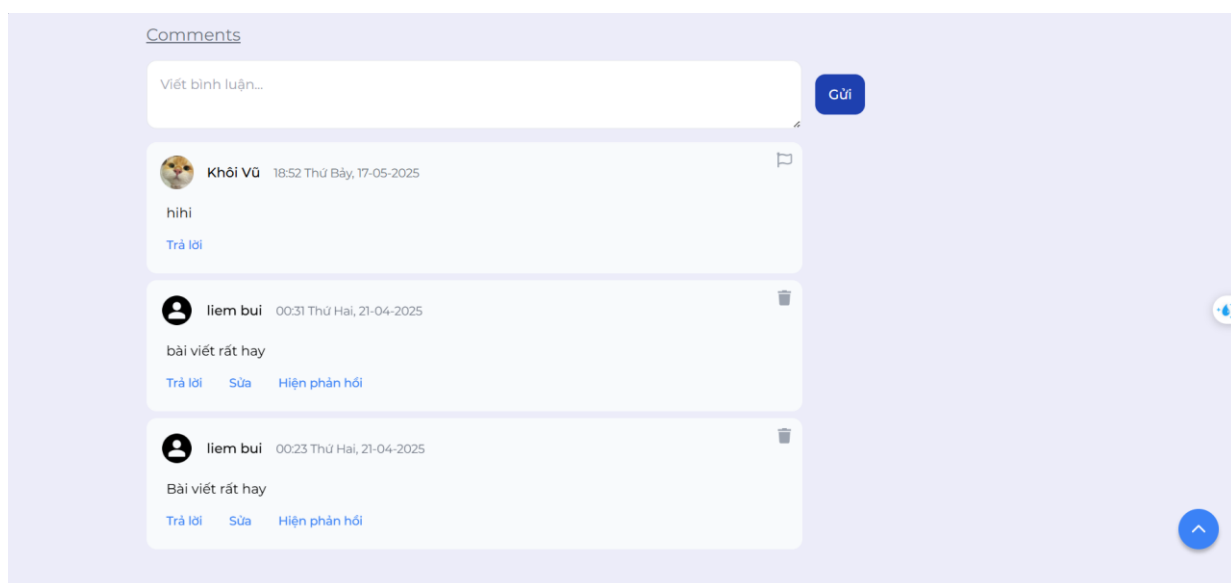
T_x

Đăng bài

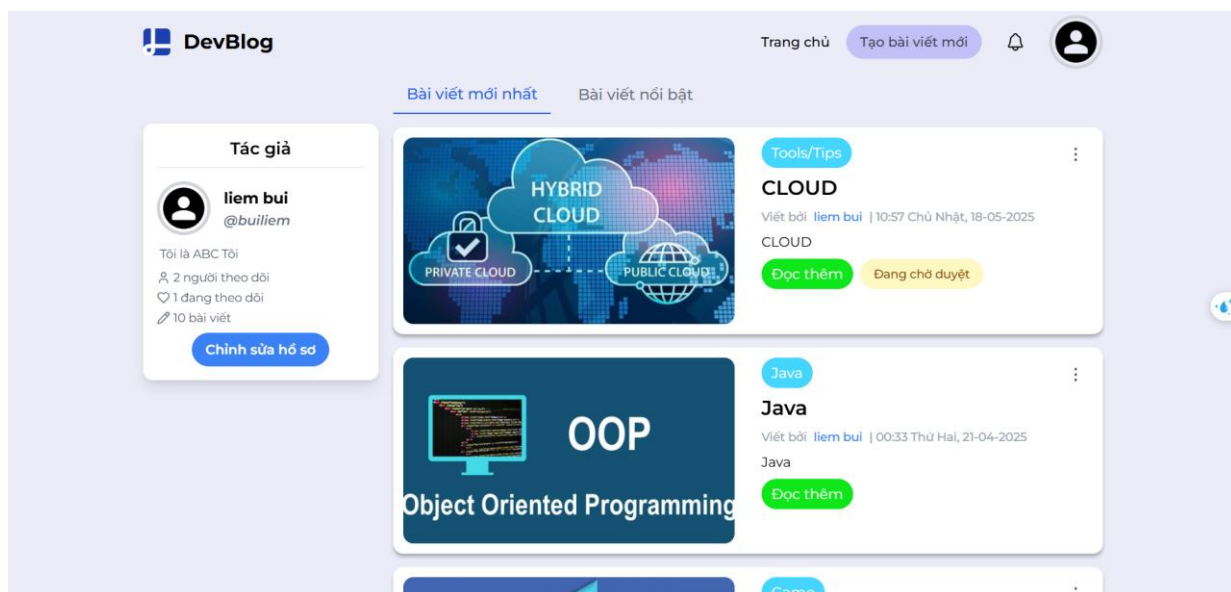
Hình 4.1.6. Giao diện tạo bài viết



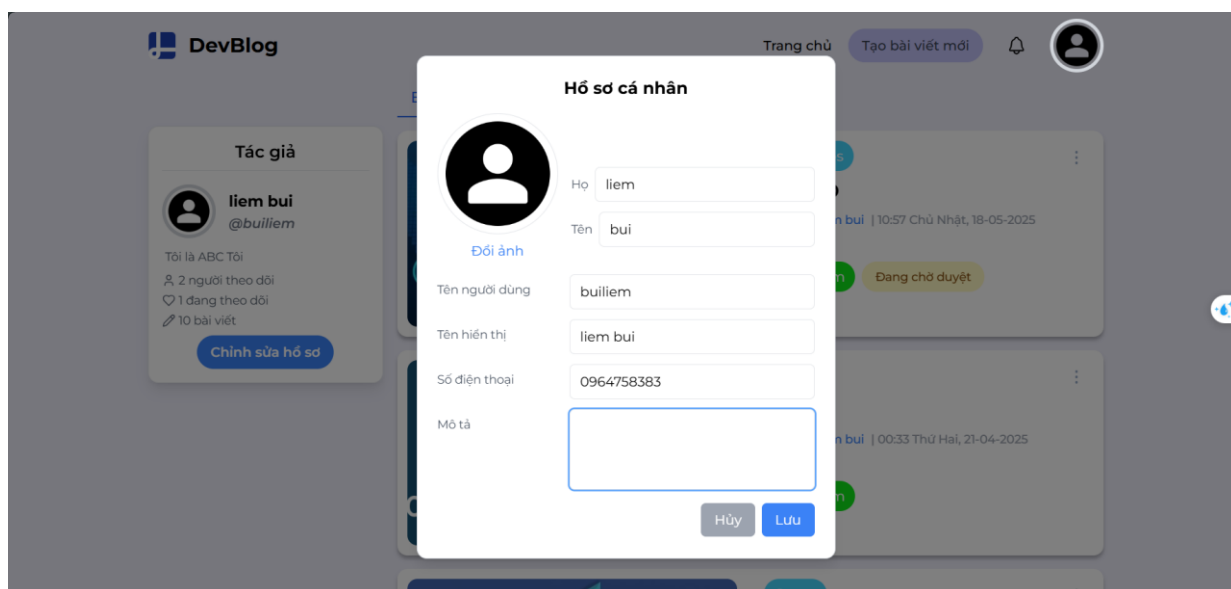
Hình 4.1.7. Giao diện bài viết



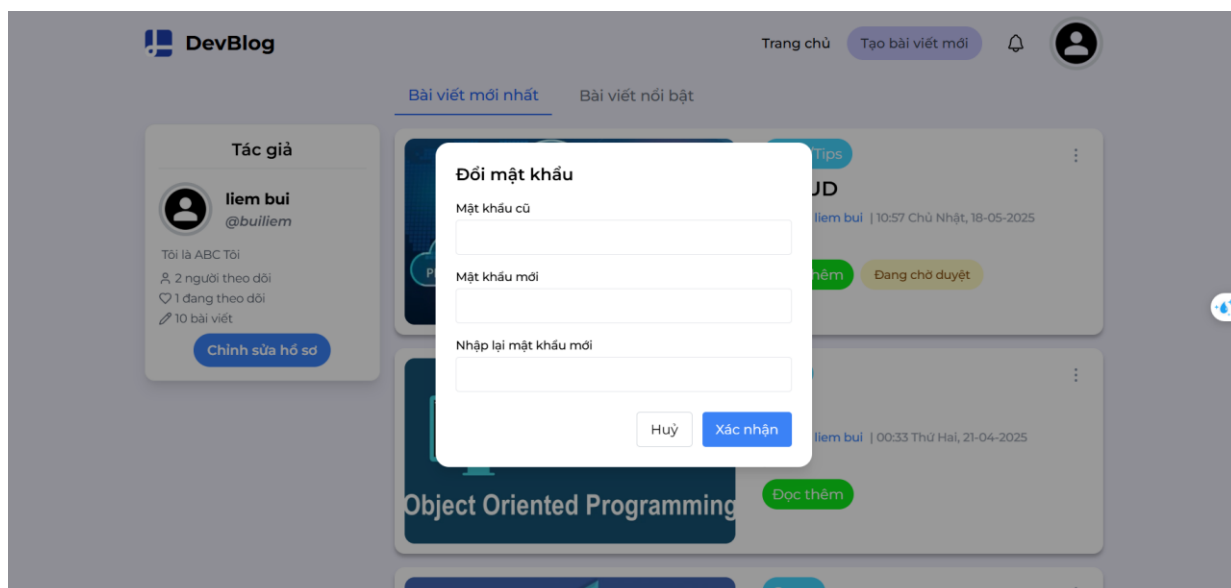
Hình 4.1.8. Giao diện bình luận



Hình 4.1.9. Giao diện trang cá nhân

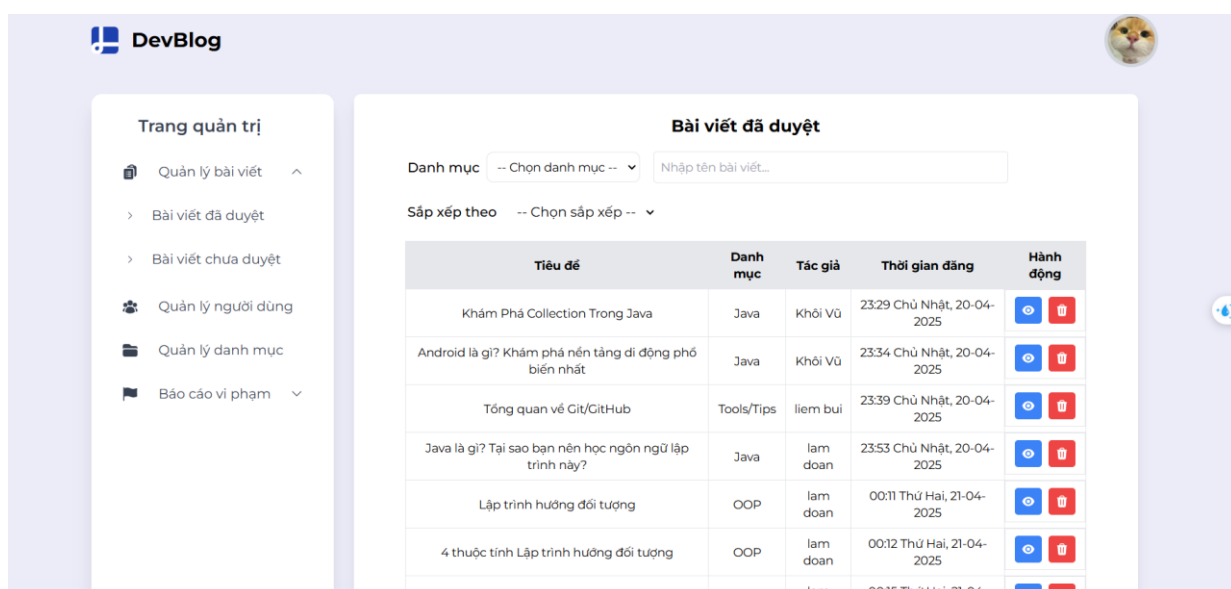


Hình 4.1.10. Giao diện chỉnh sửa hồ sơ

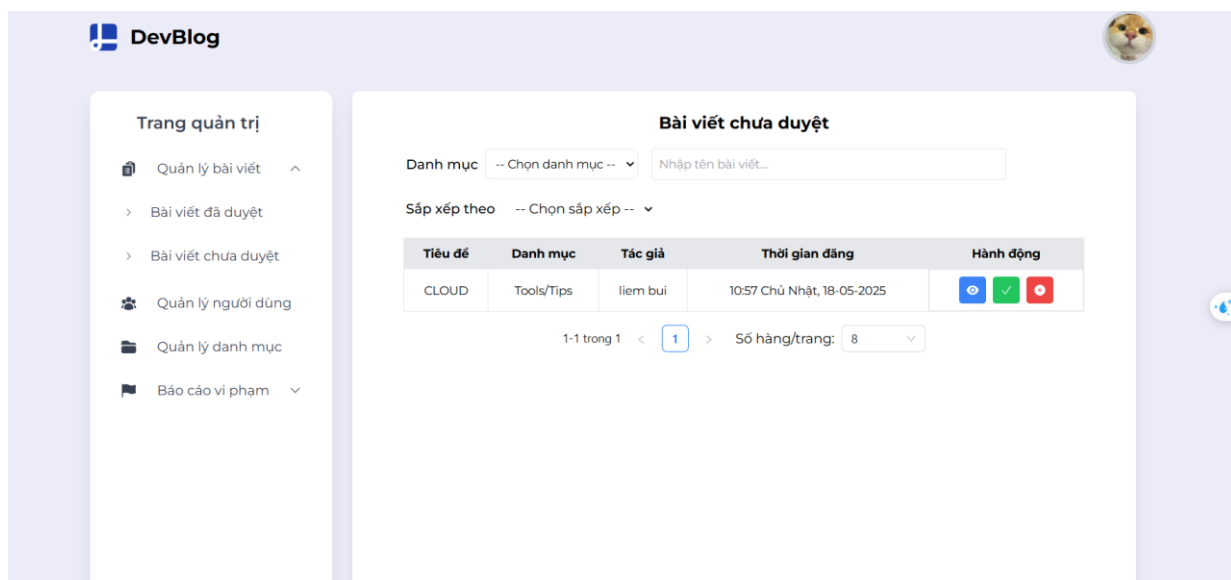


Hình 4.1.11. Giao diện đổi mật khẩu

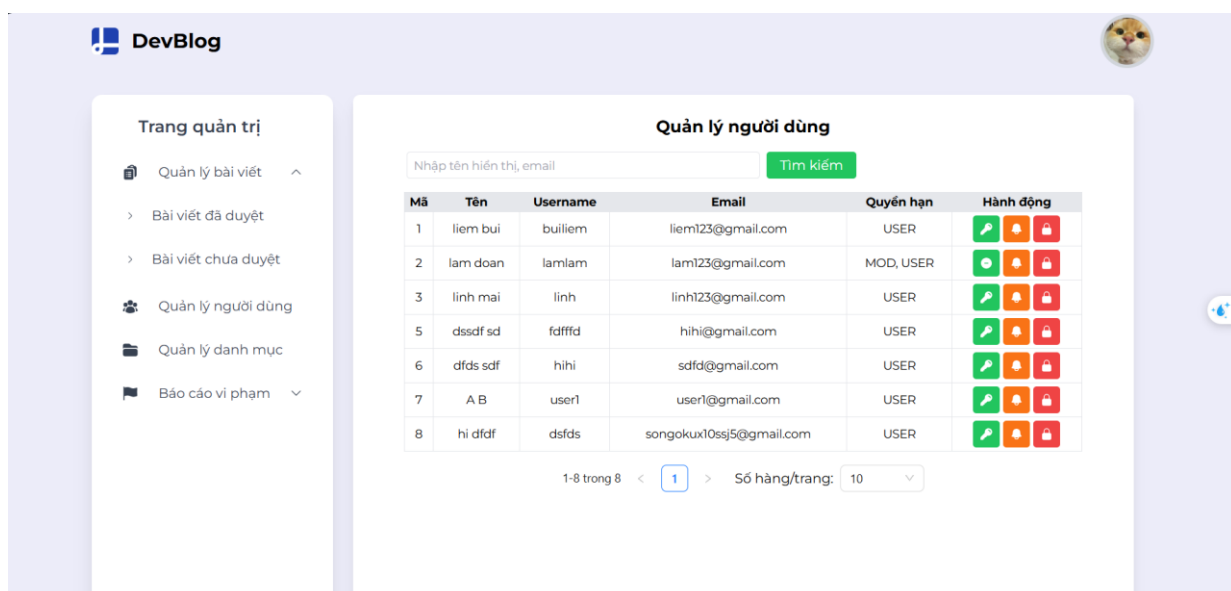
4.2. Giao diện trang quản trị



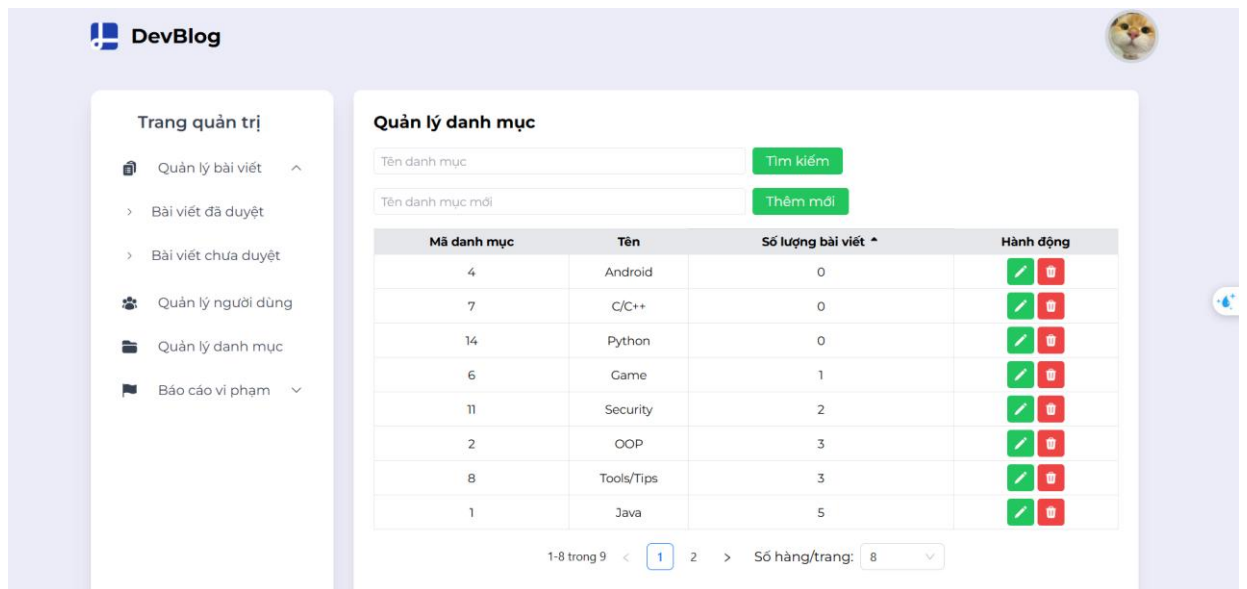
Hình 4.2.1. Giao diện quản lý bài viết đã duyệt



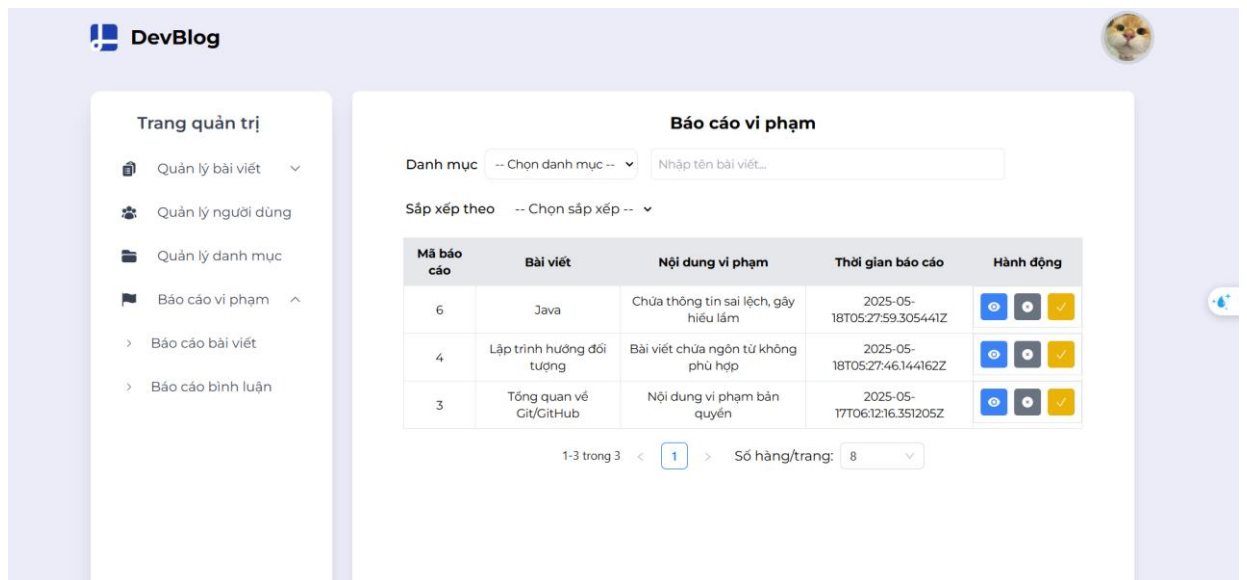
Hình 4.2.2. Giao diện quản lý bài viết chưa duyệt



Hình 4.2.3. Giao diện quản lý người dùng



Hình 4.2.4. Giao diện quản lý danh mục



Hình 4.2.5. Giao diện quản lý báo cáo bài viết

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Sau quá trình nghiên cứu và xây dựng, đề tài **Website Blog** đã đạt được những mục tiêu cơ bản đặt ra ban đầu. Cụ thể:

- Đã xây dựng được một hệ thống WebBlog với các chức năng chính như: đăng ký, đăng nhập, tạo bài viết, chỉnh sửa, xóa bài viết, hiển thị danh sách bài viết theo phân trang, tìm kiếm bài viết, bình luận và tương tác.
- Giao diện được thiết kế thân thiện, dễ sử dụng, đảm bảo trải nghiệm người dùng tốt trên cả máy tính và thiết bị di động.
- Hệ thống backend sử dụng Spring Boot giúp tổ chức mã nguồn rõ ràng, dễ mở rộng; frontend sử dụng ReactJS giúp xây dựng giao diện động, hiệu quả.
- Dữ liệu người dùng và bài viết được quản lý chặt chẽ qua hệ quản trị cơ sở dữ liệu MySQL.
- Đã áp dụng một số kiến thức như phân tầng kiến trúc 3 layers, RESTful API, bảo mật thông tin người dùng, xác thực phiên đăng nhập.

Qua đề tài, cá nhân đã nâng cao được kỹ năng lập trình web, quản lý dự án, làm việc nhóm và hiểu sâu hơn về cách một ứng dụng web thực tế được xây dựng, vận hành.

5.2. Hướng phát triển

Mặc dù hệ thống WebBlog đã đáp ứng được các yêu cầu cơ bản, tuy nhiên vẫn còn nhiều khả năng mở rộng và hoàn thiện thêm trong tương lai:

- Tích hợp xác thực nâng cao: Sử dụng tích hợp OAuth2 (Google, Facebook) để tăng tính bảo mật và tiện lợi.
- Tối ưu hiệu năng: Áp dụng cache (Redis), lazy loading hoặc SSR (Server-Side Rendering) để cải thiện tốc độ tải và trải nghiệm người dùng.
- Hệ thống quản trị (admin): Phát triển trang quản trị để kiểm duyệt bài viết, kiểm soát người dùng, thống kê truy cập, báo cáo vi phạm.
- Tối ưu SEO: Cải thiện cấu trúc đường dẫn (URL thân thiện), thêm metadata, sitemap và sử dụng SSR để giúp website thân thiện hơn với công cụ tìm kiếm.
- Ứng dụng AI/ML: Gợi ý bài viết liên quan, phân tích cảm xúc bình luận, lọc spam bằng các mô hình học máy.
- Triển khai trên nền tảng thực tế: Đưa sản phẩm lên hosting/cloud (AWS, Render, Vercel...) và theo dõi hoạt động thực tế qua công cụ như Google Analytics, Log monitoring...

TÀI LIỆU THAM KHẢO

- [1] Pivotal Software. (2024). *Spring Boot Reference Documentation*. Truy cập từ <https://docs.spring.io/spring-boot/>
- [2] ReactJS. (2024). *React – A JavaScript library for building user interfaces*. Truy cập từ <https://reactjs.org/>
- [3] MySQL. (2024). *MySQL 8.0 Reference Manual*. Truy cập từ <https://dev.mysql.com/doc/>
- [4] Mozilla Developer Network. (2024). *HTML, CSS, and JavaScript Guides*. Truy cập từ <https://developer.mozilla.org/>
- [5] Robert C. Martin. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.