TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**LƯU MINH NHIỀU - 52100456**

**BLOGPOTS**

**SỬ DỤNG ASP.NET CORE**

**BÁO CÁO CUỐI KỲ**

**CÔNG NGHỆ .NET**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**LƯU MINH NHIỀU - 52100456**

**BLOGPOTS**

**SỬ DỤNG ASP.NET CORE**

**BÁO CÁO CUỐI KỲ**

**CÔNG NGHỆ .NET**

Người hướng dẫn

**Dzoãn Xuân Thanh**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Trước tiên, em xin chân thành cảm ơn Thầy Dzoãn Xuân Thanh đã hướng dẫn và hỗ trợ em trong quá trình thực hiện báo cáo này. Những kiến thức và kinh nghiệm quý báu từ Thầy đã giúp em có cái nhìn sâu sắc hơn về công nghệ ASP.NET. Em cũng xin cảm ơn các bạn đã theo dõi và góp ý cho bài báo cáo của em. Hy vọng rằng những gì em trình bày hôm nay sẽ mang lại giá trị và góp phần vào sự phát triển chung của lớp. Một lần nữa, em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày 30 tháng 11 năm 2024*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Nhiều*

*Lưu Minh Nhiều*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của Dzoãn Xuân Thanh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 30 tháng 11 năm 2024*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Nhiều*

*Lưu Minh Nhiều*

**BLOGPOTS SỬ DỤNG ASP.NET CORE**

**TÓM TẮT**

(Dự án BlogPots được phát triển với công nghệ ASP.NET Core kết hợp với Entity Framework (EF) Core để tạo ra một nền tảng quản lý blog mạnh mẽ và linh hoạt. Hệ thống sử dụng PostgreSQL làm cơ sở dữ liệu, đảm bảo hiệu suất và khả năng mở rộng.

Phần giao diện người dùng được xây dựng bằng Vue.js, cho phép tạo ra trải nghiệm người dùng mượt mà và tương tác cao.

**BLOGPOTS USING ASP.NET CORE**

**ABSTRACT**

The BlogPots project is developed using ASP.NET Core combined with Entity Framework (EF) Core to create a powerful and flexible blog management platform. PostgreSQL is utilized as the database, ensuring performance and scalability.

The user interface is built with Vue.js, allowing for a smooth and highly interactive user experience.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ viii](#_Toc23329)

[DANH MỤC BẢNG BIỂU x](#_Toc21077)

[DANH MỤC CÁC CHỮ VIẾT TẮT xi](#_Toc17663)

[CHƯƠNG 1. MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI 1](#_Toc11741)

[1.1 Giới thiệu đề tài 1](#_Toc5025)

[1.1.1 Bối cảnh và lý do chọn đề tài 1](#_Toc6257)

[1.1.2 Mục tiêu của đề tài 1](#_Toc25116)

[1.2 Phạm vi thực hiện 2](#_Toc19831)

[1.3 Ý nghĩa thực tiễn 2](#_Toc69)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 2](#_Toc24427)

[2.1 Tổng quan về công nghệ sử dụng trong dự án 2](#_Toc6326)

[2.1.1 ASP.Net Core 2](#_Toc28747)

[2.1.2 Vue.js 2](#_Toc2077)

[2.1.3 PostgreSQL 3](#_Toc10791)

[2.2 Các vấn đề liên quan đến dự án 3](#_Toc30708)

[2.2.1 Kiến trúc hệ thống 3](#_Toc24954)

[2.2.2 RESTful API 3](#_Toc8213)

[2.2.3 Quản lý trạng thái và giao tiếp dữ liệu 4](#_Toc22257)

[2.2.4 Bảo mật trong ứng dụng 4](#_Toc28098)

[2.3 Công cụ và thư viện hỗ trợ 4](#_Toc31951)

[CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ 5](#_Toc1050)

[3.1 Đặc tả hệ thống 5](#_Toc1041)

[3.1.1 Yêu cầu chức năng 5](#_Toc26902)

[3.1.2 Yêu cầu phi chức năng 5](#_Toc22518)

[3.2 Phân tích và thiết kế hệ thống 5](#_Toc2942)

[3.2.1 Các tác nhân trong hệ thống: 5](#_Toc8864)

[3.2.2 Sơ đồ usecase tổng quát: 6](#_Toc29048)

[3.2.3 Các usecase trong hệ thống: 6](#_Toc3048)

[3.2.4 Đặc tả chi tiết các usecase: 7](#_Toc7326)

[3.3 Thiết kế cơ sở dữ liệu 20](#_Toc11162)

[3.3.1 Sơ đồ ERD 20](#_Toc22465)

[3.4 Thiết kế giao diện người dùng (UI) 20](#_Toc12970)

[3.5 Thiết kế API 21](#_Toc962)

[3.6 Công nghệ và thư viện hỗ trợ 21](#_Toc11060)

[CHƯƠNG 4. HIỆN THỰC DỰ ÁN 22](#_Toc25635)

[4.1 Program.cs 22](#_Toc27959)

[4.2 JWTServices.cs 25](#_Toc12612)

[4.3 BcryptService.cs 27](#_Toc5743)

[4.3.1 Authentication 28](#_Toc9577)

[4.3.2 Authorization 28](#_Toc1467)

[4.4 Cấu trúc dự án 36](#_Toc8949)

[4.5 Giao diện Client Side 39](#_Toc7917)

[4.5.1 Home 40](#_Toc17792)

[4.5.2 Blogs List 40](#_Toc513)

[4.5.3 Blog Detail 41](#_Toc31932)

[4.5.4 Post Detail 41](#_Toc30979)

[4.5.5 User Profile 42](#_Toc1847)

[4.5.6 Blog management: 43](#_Toc14895)

[4.5.7 Blog Create: 43](#_Toc5248)

[4.5.8 Blog Edit: 43](#_Toc18449)

[4.5.9 Post management: 44](#_Toc20803)

[4.5.10 Post Create 45](#_Toc18353)

[4.5.11 Post Edit 45](#_Toc30024)

[4.6 Tổng hợp Endpoint (Swagger): 46](#_Toc9486)

[CHƯƠNG 5. TỔNG KẾT VÀ ĐÁNH GIÁ 47](#_Toc27270)

[TÀI LIỆU THAM KHẢO 48](#_Toc24691)

# DANH MỤC HÌNH VẼ

[Hình 1 Sơ đồ usecase tổng quát 6](#_Toc28455)

[Hình 2 : Sơ đồ ERD 20](#_Toc22939)

[Hình 3 : Register UI 28](#_Toc19061)

[Hình 4 : RegisterPage.vue 29](#_Toc8031)

[Hình 5 : RegisterForm.vue 30](#_Toc7068)

[Hình 6 : Register.cs 31](#_Toc30293)

[Hình 7 : Login UI 32](#_Toc26730)

[Hình 8 : LoginPage.vue 33](#_Toc7627)

[Hình 9 : LoginForm.vue 34](#_Toc20133)

[Hình 10 : Login.cs 35](#_Toc9960)

[Hình 11 : Project Structure (1) 36](#_Toc5863)

[Hình 12 : Project Structure (2) 37](#_Toc15914)

[Hình 13 : Project Structure (3) 38](#_Toc27052)

[Hình 14 : Home UI 39](#_Toc10516)

[Hình 15 : Blogs List UI 40](#_Toc23210)

[Hình 16 : Blog Detail UI 40](#_Toc19488)

[Hình 17 : Post Detail UI 41](#_Toc15969)

[Hình 18 : User Profile UI 41](#_Toc19707)

[Hình 19 : Blog management 42](#_Toc32242)

[Hình 20 : Blog Create 42](#_Toc9457)

[Hình 21 : Blog Edit 43](#_Toc23860)

[Hình 22 : Post management 43](#_Toc14246)

[Hình 23 : Post Create UI 44](#_Toc26373)

[Hình 24 : Post Edit 45](#_Toc16904)

# DANH MỤC BẢNG BIỂU

[Bảng 1 : Tác nhân hệ thống 6](#_Toc16485)

[Bảng 2 : Usecase trong hệ thống 7](#_Toc13222)

[Bảng 3 : Usecase đăng ký tài khoản 9](#_Toc1919)

[Bảng 4 : Usecase đăng nhập 10](#_Toc29119)

[Bảng 5 : Usecase Xem và cập nhật thông tin người dùng 12](#_Toc17184)

[Bảng 6 : Usecase đăng xuất 13](#_Toc16961)

[Bảng 7 : Usecase xem danh sách bài viết 15](#_Toc31604)

[Bảng 8 : Usecase lọc bài viết theo chủ đề 16](#_Toc5084)

[Bảng 9 : Usecase tìm kiếm bài viết 18](#_Toc25271)

[Bảng 10 : Usecase Xem chi tiết bài viết 20](#_Toc27086)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| API | Application Programming Interface |
| SSR | Server Side Rendering |
| CSR | Client Side Rendering |
| SQL | Structure Query Language |
| EF | Entity Framework |

# MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

### Bối cảnh và lý do chọn đề tài

Với sự phát triển vượt bậc của công nghệ thông tin, việc chia sẻ kiến thức và thông tin qua các nền tảng trực tuyến trở nên quan trọng hơn bao giờ hết. Blogposts là một trong những hình thức phổ biến nhất để cá nhân hoặc tổ chức thể hiện ý tưởng, chia sẻ kinh nghiệm và giao lưu với cộng đồng. Tuy nhiên, để xây dựng một website blogposts chất lượng và hiện đại, việc lựa chọn công nghệ phù hợp là yếu tố then chốt.

Xây dựng và phát triển 1 dự án riêng cho bản thân với đề tài khác trong khoảng kinh nghiệm về 1 Software Development. Cùng với đó là được tiếp xúc với công nghệ mới như .Net hay các loại cơ sở dữ liệu, framework mới đối với bản thân như: PostgreSQL, VueJS…

Trong báo cáo này, tôi sẽ trình bày về dự án blog được phát triển bằng Client Side Rendering (CRS) ASP.NET Core API, sử dụng EF Core cho truy cập dữ liệu, PostgreSQL làm cơ sở dữ liệu và VueJS với Composition API cho giao diện người dùng.

### Mục tiêu của đề tài

· Xây dựng một website Blogposts với đầy đủ các tính năng cơ bản như đăng bài viết, chỉnh sửa, xóa, và hiển thị danh sách bài viết.

· Ứng dụng ASP.Net Core trong xây dựng backend để xử lý logic và quản lý dữ liệu.

· Kết hợp Vue.js cho giao diện frontend để tạo trải nghiệm người dùng mượt mà và thân thiện.

· Sử dụng PostgreSQL để lưu trữ dữ liệu một cách an toàn và hiệu quả.

· Tích hợp các thư viện và framework hỗ trợ để tăng cường hiệu suất và khả năng mở rộng của hệ thống.

## Phạm vi thực hiện

Dự án tập trung vào các chức năng chính sau:

· Quản lý bài viết (thêm, sửa, xóa, tìm kiếm).

· Phân quyền người dùng (user, owner).

· Giao diện thân thiện và dễ sử dụng cho cả người viết và người đọc.

· Hệ thống quản lý dữ liệu linh hoạt, đảm bảo tính bảo mật và hiệu quả.

## Ý nghĩa thực tiễn

Đề tài không chỉ cung cấp kiến thức chuyên sâu về công nghệ ASP.Net Core mà còn giúp sinh viên nâng cao kỹ năng lập trình, làm việc với các công cụ hiện đại trong phát triển ứng dụng web. Ngoài ra, sản phẩm hoàn thiện có thể được sử dụng làm nền tảng cho các ứng dụng thực tế khác trong tương lai.

# CƠ SỞ LÝ THUYẾT

## Tổng quan về công nghệ sử dụng trong dự án

### ASP.Net Core

ASP.Net Core là một framework phát triển ứng dụng web mã nguồn mở do Microsoft phát triển. Đây là một nền tảng đa nền tảng, giúp các lập trình viên xây dựng các ứng dụng web, API RESTful, và ứng dụng dịch vụ web mạnh mẽ.

Ưu điểm:

· Hiệu suất cao, khả năng mở rộng tốt.

· Hỗ trợ đa nền tảng (Windows, Linux, macOS).

· Hệ sinh thái phong phú với nhiều thư viện hỗ trợ mạnh mẽ.

· Dễ dàng tích hợp với các công nghệ khác như SignalR, Entity Framework Core.

### Vue.js

Vue.js là một framework JavaScript dùng để xây dựng giao diện người dùng (UI). Được biết đến với sự linh hoạt, đơn giản và khả năng tích hợp dễ dàng với các dự án khác.

Ưu điểm:

· Giao diện thân thiện, dễ học và sử dụng.

· Hỗ trợ quản lý trạng thái tốt thông qua Vuex.

· Hiệu suất cao, nhẹ và nhanh.

· Tương thích tốt với các hệ thống backend khác nhau.

### PostgreSQL

PostgreSQL là hệ quản trị cơ sở dữ liệu mã nguồn mở mạnh mẽ, được thiết kế để xử lý khối lượng dữ liệu lớn và các hoạt động giao dịch phức tạp.

Ưu điểm:

· Hỗ trợ các kiểu dữ liệu phong phú (JSON, XML, Array, v.v.).

· Đảm bảo tính toàn vẹn dữ liệu với ACID.

· Hiệu suất cao, phù hợp với cả ứng dụng nhỏ và hệ thống lớn.

· Dễ dàng mở rộng và tích hợp với các ứng dụng khác.

## Các vấn đề liên quan đến dự án

### Kiến trúc hệ thống

Dự án sử dụng kiến trúc **Client-Server**, trong đó:

* **Frontend (Vue.js)**: Đảm nhiệm việc hiển thị giao diện người dùng và tương tác trực tiếp với người dùng.
* **Backend (ASP.Net Core)**: Chịu trách nhiệm xử lý logic nghiệp vụ, quản lý dữ liệu, và cung cấp API cho frontend.
* **Database (PostgreSQL)**: Lưu trữ dữ liệu bài viết, người dùng, và các thông tin khác.

### RESTful API

Dự án áp dụng chuẩn RESTful API để giao tiếp giữa frontend và backend. RESTful API mang lại các lợi ích như:

· Đơn giản, dễ sử dụng và dễ mở rộng.

· Cho phép frontend và backend hoạt động độc lập.

### Quản lý trạng thái và giao tiếp dữ liệu

Frontend sử dụng Vuex để quản lý trạng thái toàn cục, đảm bảo tính nhất quán dữ liệu trên toàn bộ ứng dụng. Backend sử dụng Entity Framework Core để giao tiếp với cơ sở dữ liệu, giúp xử lý dữ liệu dễ dàng hơn.

### Bảo mật trong ứng dụng

Authentication và Authorization:

· Sử dụng ASP.Net Core Identity để xác thực và phân quyền người dùng.

· Bảo vệ API: Áp dụng cơ chế bảo mật bằng JWT (JSON Web Token) để đảm bảo API chỉ được truy cập bởi người dùng hợp lệ.

## Công cụ và thư viện hỗ trợ

Dự án sử dụng nhiều công cụ và thư viện để tăng tốc độ phát triển và đảm bảo chất lượng:

· Frontend: Axios (gửi HTTP request), Vue Router (quản lý routing).

· Backend: Swagger(tài liệu hóa API), Serilog (logging).

· Database: pgAdmin.

# PHÂN TÍCH THIẾT KẾ

## Đặc tả hệ thống

### Yêu cầu chức năng

· **Quản lý bài viết:** Cho phép người dùng thêm, chỉnh sửa, xóa, và hiển thị bài viết.

· **Quản lý người dùng:**

* Đăng ký, đăng nhập và quản lý tài khoản.
* Phân quyền: Người dùng thường và quản trị viên.

· **Tìm kiếm và phân loại bài viết:** Hỗ trợ người dùng tìm kiếm bài viết theo từ khóa hoặc danh mục.

· **Giao diện người dùng:**

* Thân thiện, dễ sử dụng, hỗ trợ cả thiết bị di động và máy tính.
* Hiển thị danh sách bài viết và chi tiết từng bài.

### Yêu cầu phi chức năng

* **Hiệu suất:** Hệ thống phải hoạt động ổn định, có thời gian phản hồi nhanh.
* **Bảo mật:**
  + Bảo vệ thông tin người dùng.
  + Ngăn chặn các lỗ hổng bảo mật phổ biến như SQL Injection, XSS.
* **Khả năng mở rộng:** Thiết kế hệ thống linh hoạt, có thể bổ sung thêm chức năng trong tương lai.
* **Khả năng bảo trì:** Mã nguồn rõ ràng, dễ dàng bảo trì và nâng cấp.

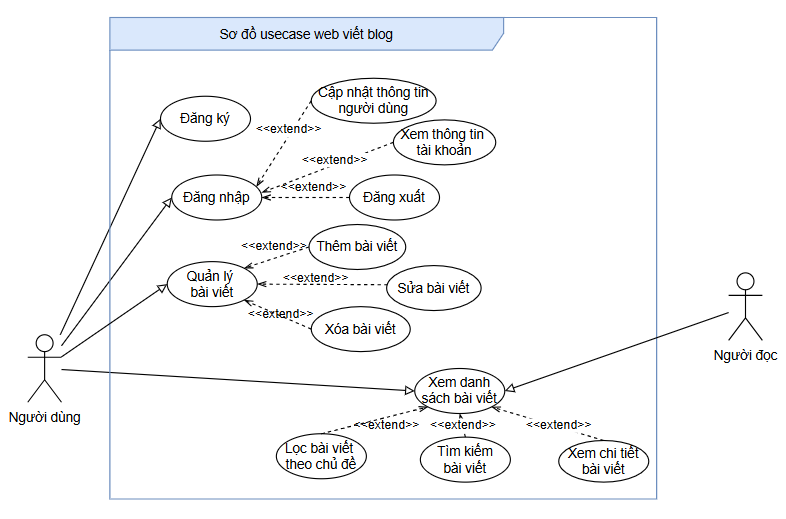
## Phân tích và thiết kế hệ thống

### Các tác nhân trong hệ thống:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tác nhân** | **Mô tả** |
| 1 | Người dùng | * Đăng ký tài khoản * Đăng nhập để có thể tạo bài viết * Quản lý các bài viết bao gồm: thêm, sửa, xóa bài viết |
| 2 | Người đọc | * Không cần đăng nhập vẫn có thể truy cập và đọc nội dung trên blog * Xem bài viết, tìm kiếm, xem chi tiết bài viết |

Bảng 1: Tác nhân hệ thống

### Sơ đồ usecase tổng quát:



Hình 1 Sơ đồ usecase tổng quát

### Các usecase trong hệ thống:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã UC** | **Tên use case** | **Actor** | **Mô tả** |
| UC1 | Đăng ký tài khoản | Người dùng | Người dùng thực hiện đăng ký tài khoản để có quyền tạo bài viết và sử dụng các chức năng khác của hệ thống. |
| UC2 | Đăng nhập | Người dùng | Khi muốn thực hiện quản lý tài khoản, quản lý bài viết thì người dùng phải đăng nhập để thực hiện các hành động |
| UC3 | Xem và cập nhật thông tin người dùng | Người dùng | Tác nhân muốn cập nhật thông tin hồ sơ của mình bao gồm đổi mật khẩu và đổi ảnh đại diện |
| UC4 | Đăng xuất | Người dùng | Các tác nhân đã đăng nhập muốn truy xuất khỏi hệ thống |
| UC5 | Quản lý bài viết | Người dùng | Người dùng tạo, chỉnh sửa, xóa hoặc đặt trạng bài viết (public, private) bài viết trên tài khoản |
| UC6 | Xem danh sách bài viết | Tất cả | Người đọc có thể xem danh sách các bài viết được public trên trang web |
| UC7 | Lọc bài viết theo chủ đề | Tất cả | Người đọc lọc danh sách bài viết theo các chủ đề hoặc tag trên hệ thống |
| UC8 | Tìm kiếm bài viết | Tất cả | Người đọc tìm kiếm bài viết dựa trên các từ khóa |
| UC9 | Xem chi tiết bài viết | Tất cả | Người đọc chọn 1 bài viết từ danh sách hoặc kết quả tìm kiếm để xem toàn bộ nội dung |

Bảng 2: Usecase trong hệ thống

### Đặc tả chi tiết các usecase:

#### Đăng ký tài khoản

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC1 | |
| **Use case** | Đăng ký tài khoản | |
| **Ngữ cảnh** | Người dùng muốn tạo tài khoản mới trên hệ thống để truy cập và sử dụng các chức năng của trang web | |
| **Mô tả** | Người dùng điền thông tin đăng ký tài khoản (gồm email, password) để hệ thống tạo và lưu trữ thông tin vào cơ sở dữ liệu. | |
| **Tác nhân** | Người dùng | |
| **Sự kiện kích hoạt** | Người dùng truy cập vào trang web | |
| **Điều kiện tiên quyết** | Người dùng chưa có tài khoản trên hệ thống | |
| **Kết quả** | Đăng ký tài khoản thành công | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng chọn mục Đăng ký | 1.1 Hệ thống hiển thị giao diện điền thông tin đăng ký (email, password) |
| 2. Người dùng nhập đầy đủ thông tin | 2.1 Hệ thống kiểm tra tính hợp lệ của thông tin:  - Email chưa tồn tại trong hệ thống  2.2 Hệ thống lưu thông tin vào cơ sở dữ liệu nếu hợp lệ  2.3 Hệ thống hiển thị thông báo “Đăng ký thành công” |
| **Luồng thay thế** | Hệ thống hiển thị thông báo lỗi (vd: email đã tồn tại hoặc sai định dạng, mật khẩu không hợp lệ)  Người dùng được yêu cầu sửa lại thông tin và thử lại | |
| **Ngoại lệ** | Email hoặc dữ liệu không thể lưu do lỗi hệ thống: => Hệ thống hiển thị thông báo lỗi chung "Có lỗi xảy ra, vui lòng thử lại sau".  Người dùng nhập thông tin không đầy đủ: => Hệ thống yêu cầu người dùng hoàn tất các trường bắt buộc. | |

Bảng 3: Usecase đăng ký tài khoản

#### Đăng nhập

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC2 | |
| **Use case** | Đăng nhập | |
| **Ngữ cảnh** | Người dùng cần đăng nhập vào hệ thống để truy cập và sử dụng các chức năng của trang web theo quyền hạn được cấp. | |
| **Mô tả** | Người dùng nhập thông tin tài khoản (email và mật khẩu). Hệ thống xác minh thông tin và cho phép truy cập nếu thông tin hợp lệ. | |
| **Tác nhân** | Người dùng | |
| **Sự kiện kích hoạt** | Người dùng chọn chức năng Đăng nhập từ giao diện hệ thống. | |
| **Điều kiện tiên quyết** | Người dùng đã có tài khoản trong hệ thống. | |
| **Kết quả** | Người dùng đăng nhập thành công và được chuyển đến trang chính hoặc giao diện phù hợp với quyền hạn của họ.  Nếu không thành công, hệ thống thông báo lỗi và yêu cầu nhập lại thông tin. | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng chọn mục Đăng nhập. | 1.1 Hệ thống hiển thị giao diện đăng nhập với các trường nhập liệu (username/email, password). |
| 2. Người dùng nhập thông tin tài khoản. | 2.1. Hệ thống kiểm tra tính hợp lệ của thông tin:  - Username/email tồn tại trong cơ sở dữ liệu.  - Mật khẩu đúng với tài khoản.  2.2. Nếu hợp lệ, hệ thống cho phép truy cập và chuyển đến trang chính. |
| **Luồng thay thế** | Thông tin không hợp lệ: - Hệ thống hiển thị thông báo lỗi (ví dụ: "Tên đăng nhập hoặc mật khẩu không đúng"). - Người dùng có thể thử lại hoặc chọn Quên mật khẩu để khôi phục. | |
| **Ngoại lệ** | Lỗi kết nối mạng hoặc hệ thống không phản hồi: - Hệ thống hiển thị thông báo "Không thể kết nối đến hệ thống, vui lòng thử lại sau".  Tài khoản bị khóa hoặc vô hiệu hóa: - Hệ thống hiển thị thông báo "Tài khoản của bạn đã bị khóa, vui lòng liên hệ hỗ trợ". | |

Bảng 4: Usecase đăng nhập

#### Xem và cập nhật thông tin người dùng

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC3 | |
| **Use case** | Xem và cập nhật thông tin người dùng | |
| **Ngữ cảnh** | Người dùng muốn xem hoặc chỉnh sửa các thông tin cá nhân của mình trên hệ thống, chẳng hạn như tên, email, mật khẩu, ảnh đại diện, hoặc thông tin bổ sung. | |
| **Mô tả** | Người dùng muốn xem hoặc chỉnh sửa các thông tin cá nhân của mình trên hệ thống, chẳng hạn như tên, email, mật khẩu, ảnh đại diện, hoặc thông tin bổ sung. | |
| **Tác nhân** | Người dùng | |
| **Sự kiện kích hoạt** | Người dùng chọn chức năng Xem/Cập nhật thông tin cá nhân từ giao diện trang web. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống.  Hệ thống có lưu thông tin cá nhân của người dùng trong cơ sở dữ liệu. | |
| **Kết quả** | Thông tin cá nhân của người dùng được hiển thị để xem.  Nếu có chỉnh sửa, thông tin được cập nhật thành công trong cơ sở dữ liệu. | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng chọn mục Thông tin cá nhân. | 1.1 Hệ thống hiển thị giao diện hiển thị thông tin hiện tại của người dùng (tên, email, v.v.). |
| 2. Người dùng chỉnh sửa thông tin cần thiết. | 2.1. Hệ thống kiểm tra tính hợp lệ của các trường thông tin mới (email, mật khẩu, v.v.)  2.2. Nếu hợp lệ, hệ thống lưu thông tin mới vào cơ sở dữ liệu.  2.3. Hệ thống hiển thị thông báo "Cập nhật thành công". |
| **Luồng thay thế** | Thông tin không hợp lệ:   * Hệ thống hiển thị thông báo lỗi (ví dụ: "Email không hợp lệ", "Mật khẩu quá ngắn"). * Người dùng được yêu cầu nhập lại thông tin chính xác. | |
| **Ngoại lệ** | Người dùng không nhập đủ thông tin bắt buộc: - Hệ thống hiển thị thông báo lỗi yêu cầu người dùng hoàn thành các trường bắt buộc.  Lỗi lưu thông tin vào cơ sở dữ liệu: - Hệ thống hiển thị thông báo "Có lỗi xảy ra, vui lòng thử lại sau". | |

Bảng 5: Usecase Xem và cập nhật thông tin người dùng

#### Đăng xuất

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC4 | |
| **Use case** | Đăng xuất | |
| **Ngữ cảnh** | Người dùng muốn thoát khỏi hệ thống sau khi hoàn thành các hoạt động để đảm bảo an toàn thông tin tài khoản. | |
| **Mô tả** | Người dùng thực hiện yêu cầu đăng xuất. Hệ thống xóa thông tin phiên làm việc hiện tại của người dùng và chuyển về trang chính hoặc trang đăng nhập. | |
| **Tác nhân** | Người dùng | |
| **Sự kiện kích hoạt** | Người dùng chọn chức năng Đăng xuất từ giao diện hệ thống. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống. | |
| **Kết quả** | Phiên làm việc của người dùng được kết thúc.  Người dùng được chuyển về trang chính hoặc trang đăng nhập. | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng chọn mục Đăng xuất. | 1.1. Hệ thống hiển thị thông báo xác nhận (nếu cần). |
| 2. Người dùng xác nhận muốn đăng xuất. | 2.1. Hệ thống xóa thông tin phiên làm việc của người dùng.  2.2. Hệ thống chuyển người dùng về trang chính hoặc trang đăng nhập. |
| **Luồng thay thế** | Người dùng không xác nhận đăng xuất (nếu có xác nhận): Hệ thống hủy yêu cầu đăng xuất và quay lại trạng thái trước đó. | |
| **Ngoại lệ** | Lỗi kết nối hoặc hệ thống không phản hồi:  Hệ thống hiển thị thông báo "Không thể xử lý yêu cầu đăng xuất, vui lòng thử lại sau". | |

Bảng 6: Usecase đăng xuất

#### Xem danh sách bài viết

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC6 | |
| **Use case** | Xem danh sách bài viết | |
| **Ngữ cảnh** | Người dùng muốn xem danh sách các bài viết có sẵn trên hệ thống để tìm kiếm hoặc duyệt nội dung theo nhu cầu. | |
| **Mô tả** | Người dùng truy cập và Danh sách bài viết hiển thị trên hệ thống | |
| **Tác nhân** | Tất cả | |
| **Sự kiện kích hoạt** | Người dùng chọn chức năng Danh sách bài viết từ giao diện trang web. | |
| **Điều kiện tiên quyết** | Hệ thống đã có bài viết được lưu trữ trong cơ sở dữ liệu. | |
| **Kết quả** | Danh sách bài viết được hiển thị với các thông tin cơ bản (ví dụ: tiêu đề, tác giả, ngày đăng, lượt xem).  Người dùng có thể nhấp vào từng bài viết để xem chi tiết. | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng chọn mục Danh sách bài viết. | 1.1. Hệ thống lấy dữ liệu bài viết từ cơ sở dữ liệu.  1.2. Hệ thống hiển thị danh sách bài viết theo tiêu chí mặc định hoặc đã được sắp xếp. |
| 2. Người dùng duyệt danh sách và chọn bài viết. | 2.1. Hệ thống điều hướng người dùng đến trang chi tiết bài viết được chọn. |
| **Luồng thay thế** | Không có bài viết nào trong hệ thống:  Hệ thống hiển thị thông báo "Hiện tại chưa có bài viết nào". | |
| **Ngoại lệ** | Lỗi tải dữ liệu từ cơ sở dữ liệu:  Hệ thống hiển thị thông báo lỗi "Không thể tải danh sách bài viết, vui lòng thử lại sau". | |

Bảng 7: Usecase xem danh sách bài viết

#### Lọc bài viết theo chủ đề

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC7 | |
| **Use case** | Lọc bài viết theo chủ đề | |
| **Ngữ cảnh** | Người dùng muốn tìm kiếm các bài viết liên quan đến một chủ đề hoặc tag cụ thể để dễ dàng tiếp cận nội dung phù hợp. | |
| **Mô tả** | Người dùng chọn một chủ đề hoặc tag từ danh sách hoặc nhập vào thanh tìm kiếm. Hệ thống hiển thị danh sách các bài viết tương ứng với chủ đề hoặc tag được chọn. | |
| **Tác nhân** | Tất cả | |
| **Sự kiện kích hoạt** | Người dùng chọn một chủ đề, tag từ giao diện hoặc nhập vào thanh tìm kiếm. | |
| **Điều kiện tiên quyết** | Hệ thống có các bài viết được gắn tag hoặc phân loại theo chủ đề.  Người dùng đã truy cập vào trang web và chọn chức năng lọc. | |
| **Kết quả** | Danh sách bài viết được hiển thị, chỉ bao gồm các bài liên quan đến chủ đề hoặc tag đã chọn.  Người dùng có thể nhấp vào từng bài viết để xem chi tiết. | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng chọn chủ đề hoặc tag cần lọc. | 1.1. Hệ thống truy vấn cơ sở dữ liệu để lấy các bài viết liên quan đến chủ đề hoặc tag đã chọn.  1.2. Hệ thống hiển thị danh sách các bài viết tương ứng. |
| 2. Người dùng duyệt danh sách và chọn bài viết. | 2.1. Hệ thống điều hướng đến trang chi tiết bài viết được chọn. |
| **Luồng thay thế** | Lỗi kết nối hoặc cơ sở dữ liệu không phản hồi:  Hệ thống hiển thị thông báo "Có lỗi xảy ra, vui lòng thử lại sau". | |
| **Ngoại lệ** | Lỗi tải dữ liệu từ cơ sở dữ liệu:  Hệ thống hiển thị thông báo lỗi "Không thể tải danh sách bài viết, vui lòng thử lại sau". | |

Bảng 8: Usecase lọc bài viết theo chủ đề

#### Tìm kiếm bài viết

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC8 | |
| **Use case** | Tìm kiếm bài viết | |
| **Ngữ cảnh** | Người dùng muốn tìm kiếm bài viết dựa trên từ khóa để nhanh chóng tiếp cận nội dung mong muốn. | |
| **Mô tả** | Người dùng nhập từ khóa tìm kiếm vào thanh công cụ tìm kiếm trên trang web. Hệ thống phân tích từ khóa, tìm các bài viết liên quan trong cơ sở dữ liệu và hiển thị danh sách kết quả phù hợp. | |
| **Tác nhân** | Tất cả | |
| **Sự kiện kích hoạt** | Người dùng nhập từ khóa và nhấn nút Tìm kiếm. | |
| **Điều kiện tiên quyết** | Hệ thống có các bài viết được lưu trữ trong cơ sở dữ liệu.  Từ khóa tìm kiếm phải phù hợp với nội dung hoặc metadata của bài viết (tiêu đề, mô tả, tag, v.v.). | |
| **Kết quả** | Hệ thống hiển thị danh sách các bài viết phù hợp với từ khóa tìm kiếm.  Người dùng có thể nhấp vào bài viết để xem chi tiết. | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng nhập từ khóa vào thanh tìm kiếm. | 1.1. Hệ thống nhận từ khóa và thực hiện truy vấn cơ sở dữ liệu để tìm kiếm bài viết liên quan.  1.2. Hệ thống hiển thị danh sách kết quả tìm kiếm, bao gồm tiêu đề, mô tả ngắn và các thông tin cơ bản khác. |
| 2. Người dùng duyệt danh sách và chọn bài viết. | 2.1. Hệ thống điều hướng đến trang chi tiết bài viết được chọn. |
| **Luồng thay thế** | Không tìm thấy kết quả phù hợp:  Hệ thống hiển thị thông báo "Không tìm thấy bài viết nào liên quan đến từ khóa bạn nhập". | |
| **Ngoại lệ** | Người dùng không nhập từ khóa hoặc nhập từ khóa không hợp lệ:  Hệ thống hiển thị thông báo yêu cầu nhập từ khóa hợp lệ (ví dụ: "Vui lòng nhập từ khóa để tìm kiếm").  Lỗi truy vấn hoặc cơ sở dữ liệu không phản hồi:  Hệ thống hiển thị thông báo "Có lỗi xảy ra, vui lòng thử lại sau". | |

Bảng 9: Usecase tìm kiếm bài viết

#### Xem chi tiết bài viết

|  |  |  |
| --- | --- | --- |
| **Mã Use Case** | UC9 | |
| **Use case** | Xem chi tiết bài viết | |
| **Ngữ cảnh** | Người dùng muốn xem nội dung chi tiết của một bài viết đã được chọn từ danh sách bài viết hoặc kết quả tìm kiếm. | |
| **Mô tả** | Người dùng nhấp vào tiêu đề hoặc ảnh đại diện của một bài viết từ danh sách hoặc kết quả tìm kiếm để truy cập vào trang chi tiết của bài viết đó. Hệ thống hiển thị toàn bộ nội dung của bài viết cùng các thông tin liên quan như tác giả, ngày đăng, và các bài viết liên quan khác. | |
| **Tác nhân** | Tất cả | |
| **Sự kiện kích hoạt** | Người dùng chọn một bài viết từ danh sách hoặc kết quả tìm kiếm. | |
| **Điều kiện tiên quyết** | Người dùng đã truy cập vào hệ thống và có quyền xem bài viết.  Bài viết đã được hệ thống lưu trữ trong cơ sở dữ liệu và có thông tin đầy đủ. | |
| **Kết quả** | Hệ thống hiển thị chi tiết bài viết bao gồm nội dung, tiêu đề, tác giả, ngày đăng, tag, và các bài viết liên quan.  Người dùng có thể duyệt qua nội dung hoặc tương tác với các bài viết liên quan, để lại bình luận (nếu có tính năng bình luận). | |
| **Luồng sự kiện** | **Actor** | **System** |
| 1. Người dùng chọn bài viết từ danh sách hoặc kết quả tìm kiếm. | 1.1. Hệ thống tải chi tiết bài viết từ cơ sở dữ liệu và hiển thị nội dung đầy đủ. |
| 2. Người dùng duyệt qua nội dung bài viết. | 2.1. Hệ thống hiển thị các thông tin bổ sung (tác giả, ngày đăng, tag, bài viết liên quan, v.v.).  2.2. Nếu có tính năng bình luận, hệ thống hiển thị phần bình luận dưới bài viết (nếu có). |
| **Luồng thay thế** | Không có bài viết trong cơ sở dữ liệu:  Hệ thống hiển thị thông báo "Bài viết không tồn tại hoặc đã bị xóa". | |
| **Ngoại lệ** | Lỗi tải bài viết:  Hệ thống hiển thị thông báo "Có lỗi xảy ra, vui lòng thử lại sau". | |

Bảng 10: Usecase Xem chi tiết bài viết

## Thiết kế cơ sở dữ liệu

### Sơ đồ ERD



Hình 2: Sơ đồ ERD

## Thiết kế giao diện người dùng (UI)

· **Trang chính**: Hiển thị danh sách bài viết với các thông tin cơ bản như tiêu đề, tác giả, ngày đăng.

· **Trang chi tiết bài viết**: Hiển thị nội dung bài viết và các thông tin liên quan.

· **Trang quản lý bài viết (Owner)**: Quản trị viên có thể thêm, chỉnh sửa, hoặc xóa bài viết.

· **Trang đăng nhập/đăng ký**: Người dùng có thể tạo tài khoản hoặc đăng nhập vào hệ thống.

## Thiết kế API

Các API chính được thiết kế theo chuẩn RESTful

· **/api/auth/**

· **POST /register**: Đăng kí tài khỏan.

· **POST /login**: Đăng nhập và lấy token JWT.

· **/api/post**:

· **GET /**: lấy danh sách các bài viết.

· **POST /**: Thêm bài viết mới.

· **PUT /{id}**: chỉnh sửa bài viết

· **DELETE /{id}**: xóa bài viết

· **GET /getpushedlishposts**: lấy tất cả các bài post được pushlish.

· **/api/blog**:

· …các route

## Công nghệ và thư viện hỗ trợ

· **Backend**: ASP.Net Core, Entity Framework Core (ORM), BCrypt, Swagger, DotNetEnv.

· **Frontend**: Vue.js, Vue Router, Axios, vue-loading-overlay, vue-toastification, CKEditor5.

· **Database**: PostgreSQL.

# HIỆN THỰC DỰ ÁN

## Program.cs

using BackEnd.Configs.Enviroments;

using BackEnd.Database;

using BackEnd.Database.Tables;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Identity;

using Microsoft.Extensions.FileProviders;

using Microsoft.Extensions.Options;

using Microsoft.OpenApi.Models;

using System;

var builder = WebApplication.CreateBuilder(args);

var env = new EnviromentVariables();

//bonus config

//Addconsole logging

builder.Logging.AddConsole();

//Configures

//PostgreSQL

builder.Services.ConfigureDatabase(env.GetConnectionString());

//JWT

builder.Services.ConfigureJwt(env.GetJwtKey());

// Add CORS services to the container.

builder.Services.AddCors(*options* =>

{

    options.AddPolicy("AllowLocalhost",

*builder* =>

        {

            builder.WithOrigins("http://localhost:8080")

                   .AllowAnyHeader()

                   .AllowAnyMethod()

                   .WithExposedHeaders("X-Pagination");

        });

});

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(*option* =>

{

    option.SwaggerDoc("v1", new OpenApiInfo { Title = "Demo API", Version = "v1" });

    option.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

    {

        In = ParameterLocation.Header,

        Description = "Please enter a valid token",

        Name = "Authorization",

        Type = SecuritySchemeType.ApiKey,

        BearerFormat = "JWT",

        Scheme = "Bearer"

    });

    option.AddSecurityRequirement(new OpenApiSecurityRequirement()

    {

        {

            new OpenApiSecurityScheme

            {

                Reference = new OpenApiReference

                {

                    Type = ReferenceType.SecurityScheme,

                    Id = "Bearer"

                },

                Scheme = "oauth2",

                Name = "Bearer",

                In = ParameterLocation.Header,

            },

            new List<string>()

        }

    });

});

// accept lowercase urls

builder.Services.AddRouting(*options* => options.LowercaseUrls = true);

// Serialization of enums as strings in JSON responses

builder.Services.AddControllers().AddJsonOptions(*option* =>

{

    option.JsonSerializerOptions.Converters.Add(new System.Text.Json.Serialization.JsonStringEnumConverter());

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

//app.UseHttpsRedirection();

app.MapControllers();

app.UseCors("AllowLocalhost");

// Serve static files from the /uploads directory

app.UseStaticFiles(new StaticFileOptions

{

    FileProvider = new PhysicalFileProvider(

        Path.Combine(builder.Environment.ContentRootPath, "uploads")),

    RequestPath = "/uploads"

});

app.UseAuthentication();

app.UseAuthorization();

app.Run();

## JWTServices.cs

using BackEnd.Configs.Enviroments;

using BackEnd.Database.Tables;

using BackEnd.DTO;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace BackEnd.Servicers

{

    public class JWTServices

    {

        public static string GenerateJwtToken(UserDTO *user*)

        {

            // get jwt key from enviroment variables

            var env = new EnviromentVariables();

            var tokenHandler = new JwtSecurityTokenHandler();

            var key = Encoding.ASCII.GetBytes(env.GetJwtKey());

            // create claims from the token handler and add them to the token handler

            var claims = new List<Claim>

            {

                new Claim(ClaimTypes.NameIdentifier, user.UID),

                    //email bonus - unique email of user

                    new Claim(ClaimTypes.Email, user.Email)

            };

            // add roles to the claims

            foreach (var role in user.Roles)

            {

                claims.Add(new Claim(ClaimTypes.Role, role));

            }

            // create the token descriptor

            var tokenDescriptor = new SecurityTokenDescriptor

            {

                Subject = new ClaimsIdentity(claims),

                Expires = DateTime.UtcNow.AddHours(1),

                Audience = "http://localhost:5144",

                Issuer = "http://localhost:5144",

                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)

            };

            // create the token

            var token = tokenHandler.CreateToken(tokenDescriptor);

            return tokenHandler.WriteToken(token);

        }

    }

}

## BcryptService.cs

namespace BackEnd.Servicers

{

    public class BcryptService

    {

        public static string HashPassword(string *password*)

        {

            return BCrypt.Net.BCrypt.HashPassword(password);

        }

        public static bool Verify(string *password*, string *hash*)

        {

            return BCrypt.Net.BCrypt.Verify(password, hash);

        }

    }

}

### Authentication

Authentication (xác thực) là quá trình xác định danh tính của người dùng khi họ đăng nhập vào ứng dụng. Trong dự án này, tôi đã áp dụng các biện pháp sau để đảm bảo tính an toàn và bảo mật:

- Mã hóa mật khẩu:

Khi người dùng đăng ký tài khoản, mật khẩu của họ được mã hóa bằng phương pháp bcrypt trước khi lưu trữ vào cơ sở dữ liệu.

Bằng cách này, ngay cả khi cơ sở dữ liệu bị xâm nhập, thông tin mật khẩu vẫn được bảo vệ, giảm thiểu rủi ro cho người dùng.

- Tạo JSON Web Token (JWT):

Khi người dùng đăng nhập thành công, hệ thống sẽ tạo một JWT.

Token này chứa thông tin người dùng và được mã hóa để đảm bảo tính toàn vẹn.

Token sẽ được gửi lại cho người dùng và được sử dụng trong các yêu cầu tiếp theo để xác thực danh tính.

### Authorization

Phân quyền là quá trình xác định quyền truy cập của người dùng vào các tài nguyên trong ứng dụng. Trong dự án này, tôi đã thực hiện phân quyền như sau:

Kiểm tra quyền truy cập:

- Mỗi khi người dùng gửi yêu cầu đến server, JWT sẽ được gửi kèm theo.

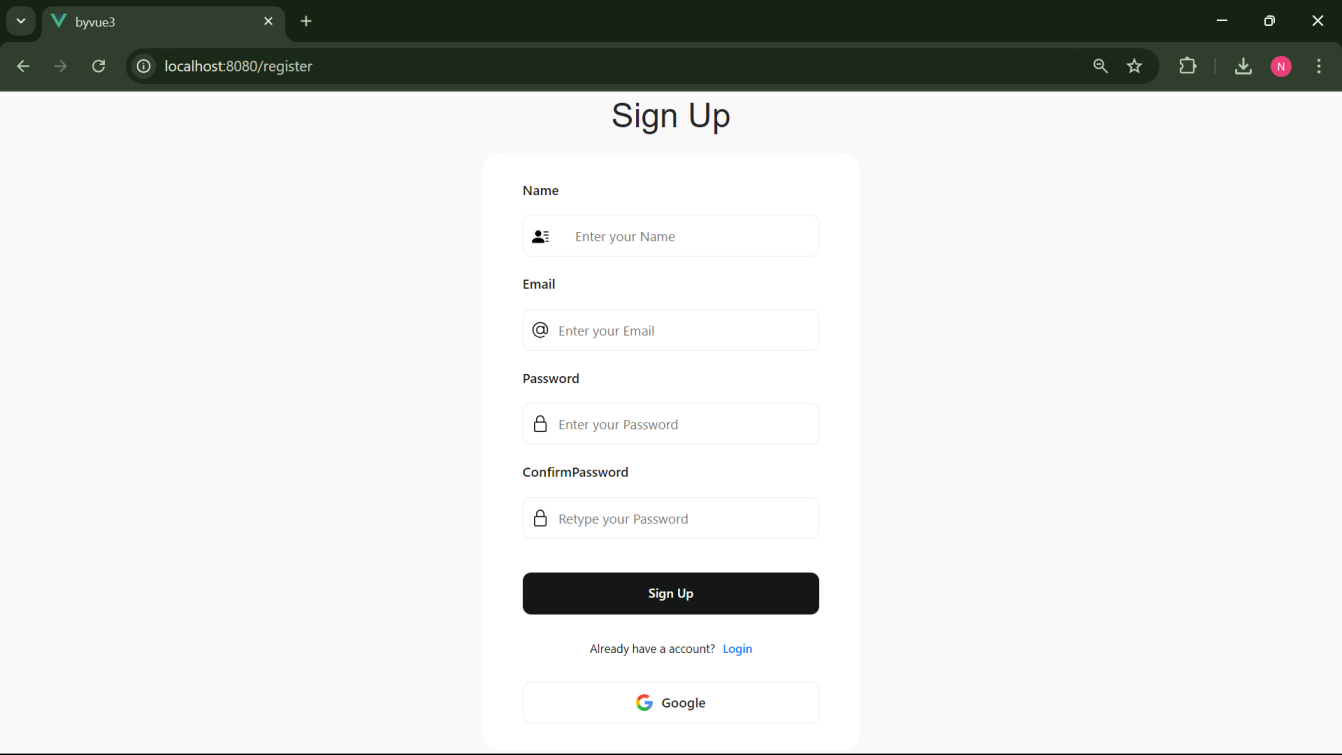
- Server sẽ giải mã token và kiểm tra thông tin người dùng, bao gồm vai trò và quyền hạn của họ.

Quy định quyền truy cập:

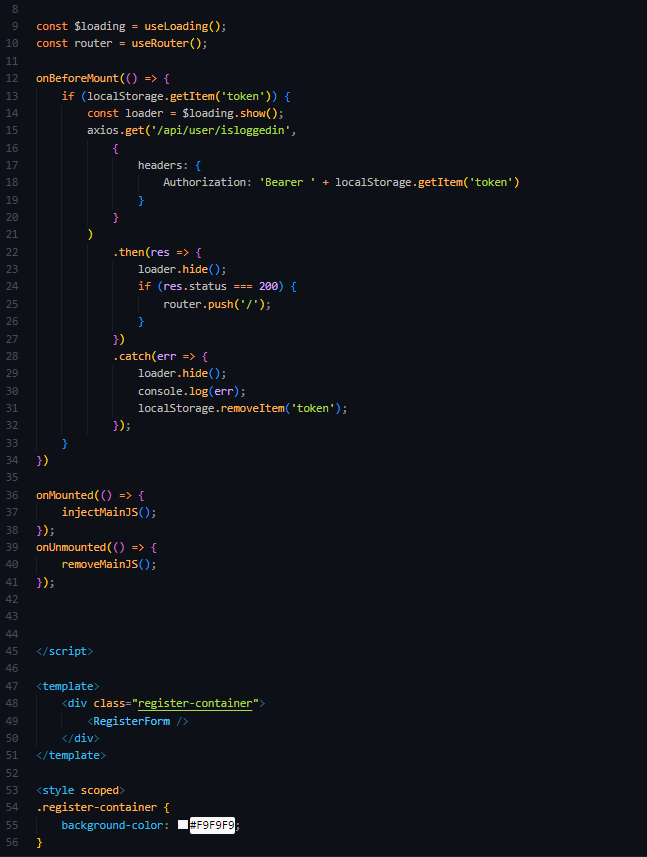
- Dựa trên vai trò của người dùng (ví dụ: quản trị viên, người dùng thường), hệ thống sẽ quyết định quyền truy cập vào các tài nguyên khác nhau.

- Chỉ những người dùng có quyền hợp lệ mới có thể thực hiện các thao tác nhất định như tạo, chỉnh sửa hoặc xóa bài viết.

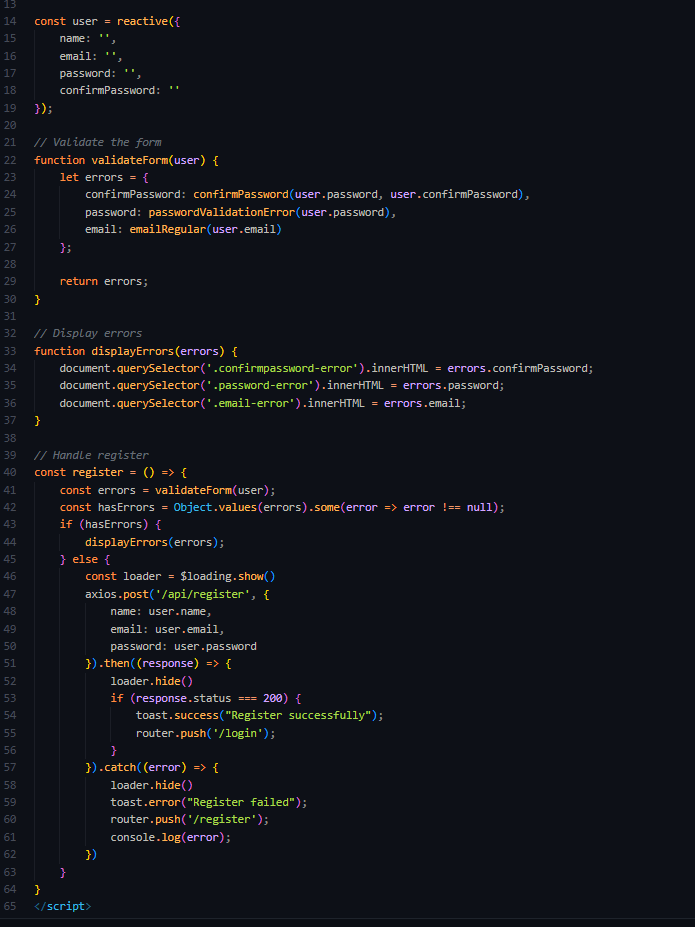
Register - FE:



Hình 3: Register UI

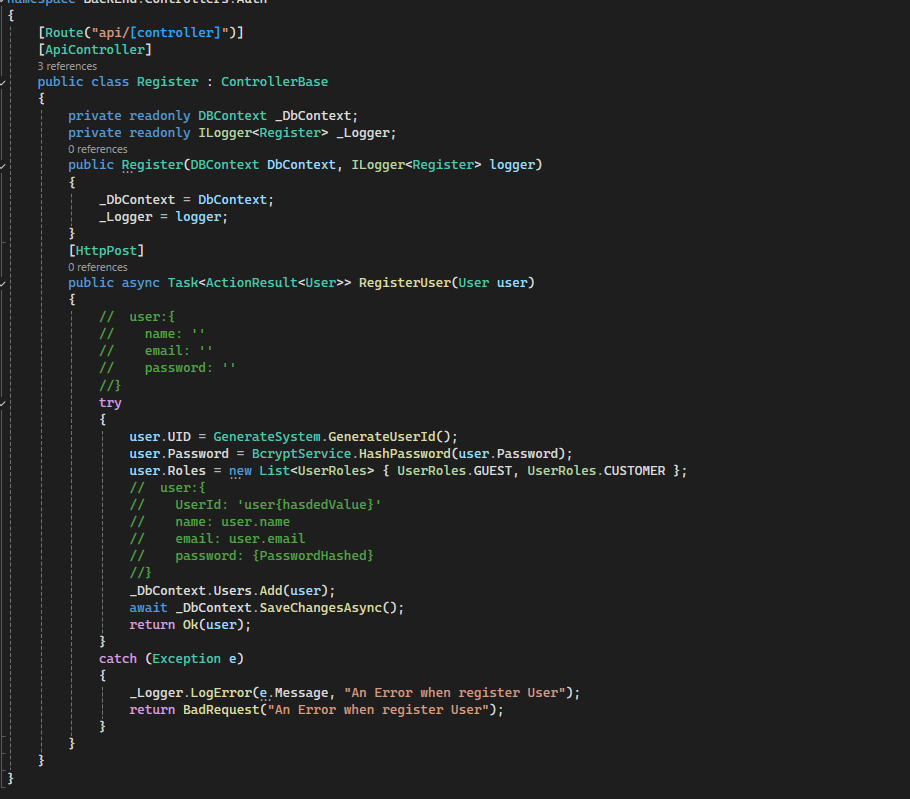


Hình 4: RegisterPage.vue



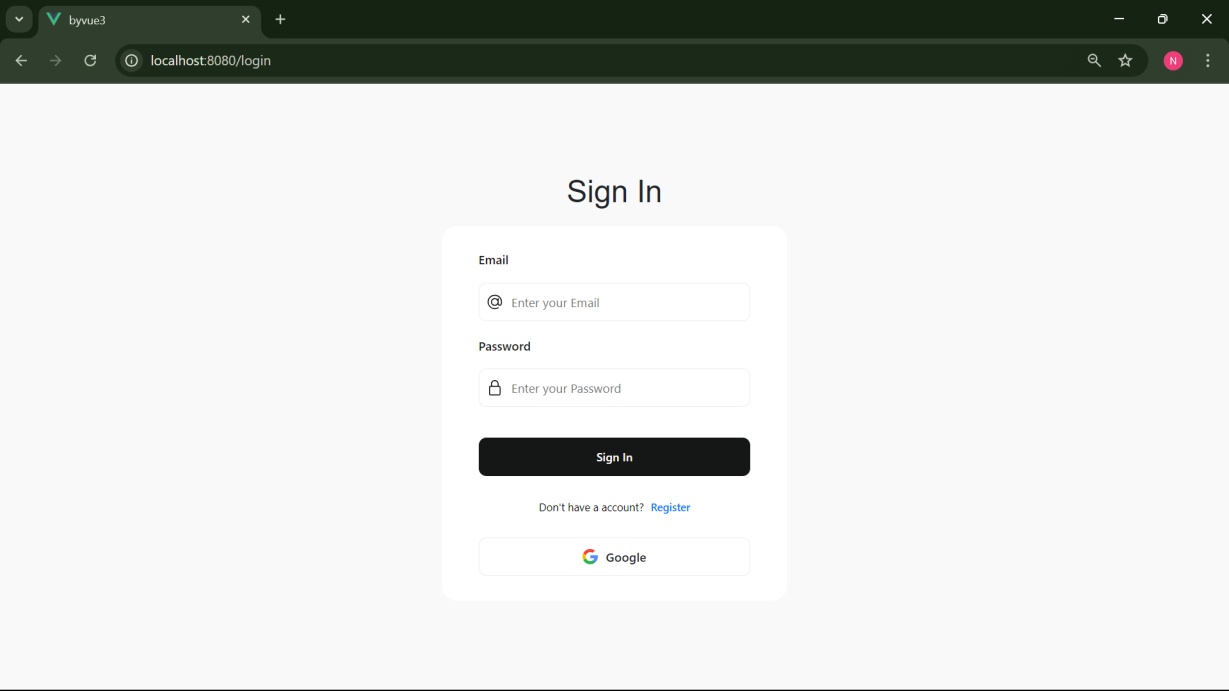
Hình 5: RegisterForm.vue

Register - BE:

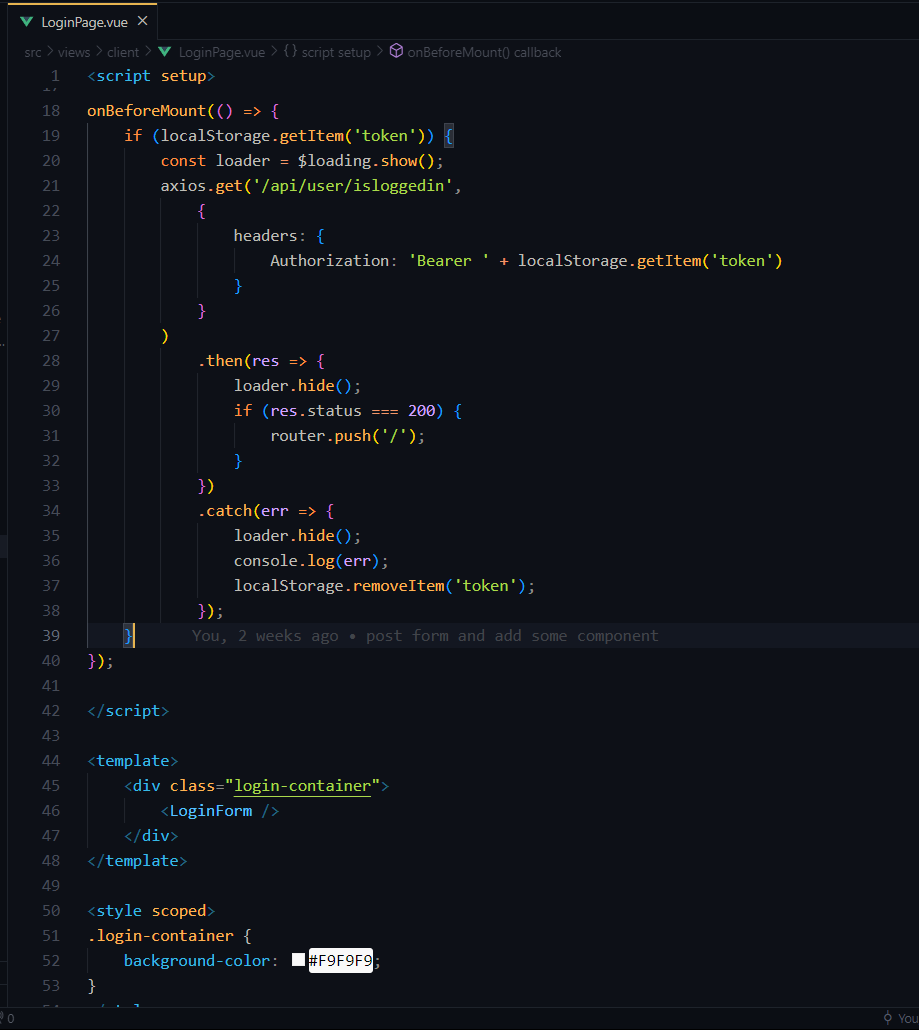


Hình 6: Register.cs

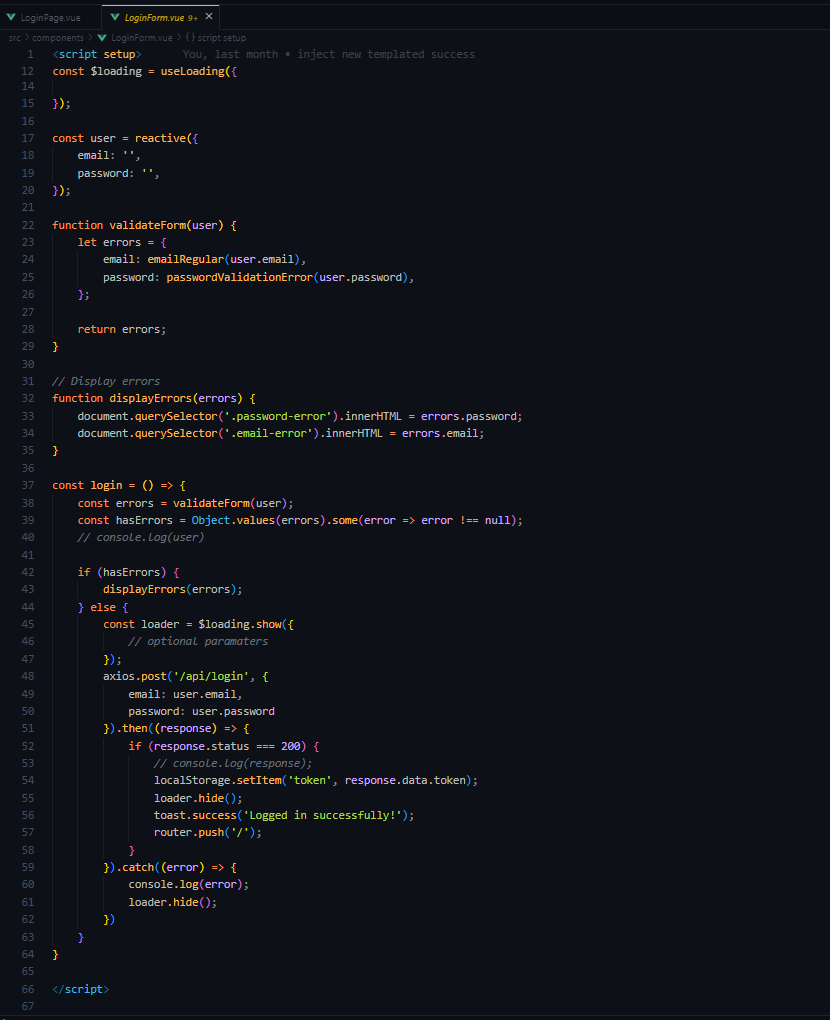
Login - FE:



Hình 7: Login UI

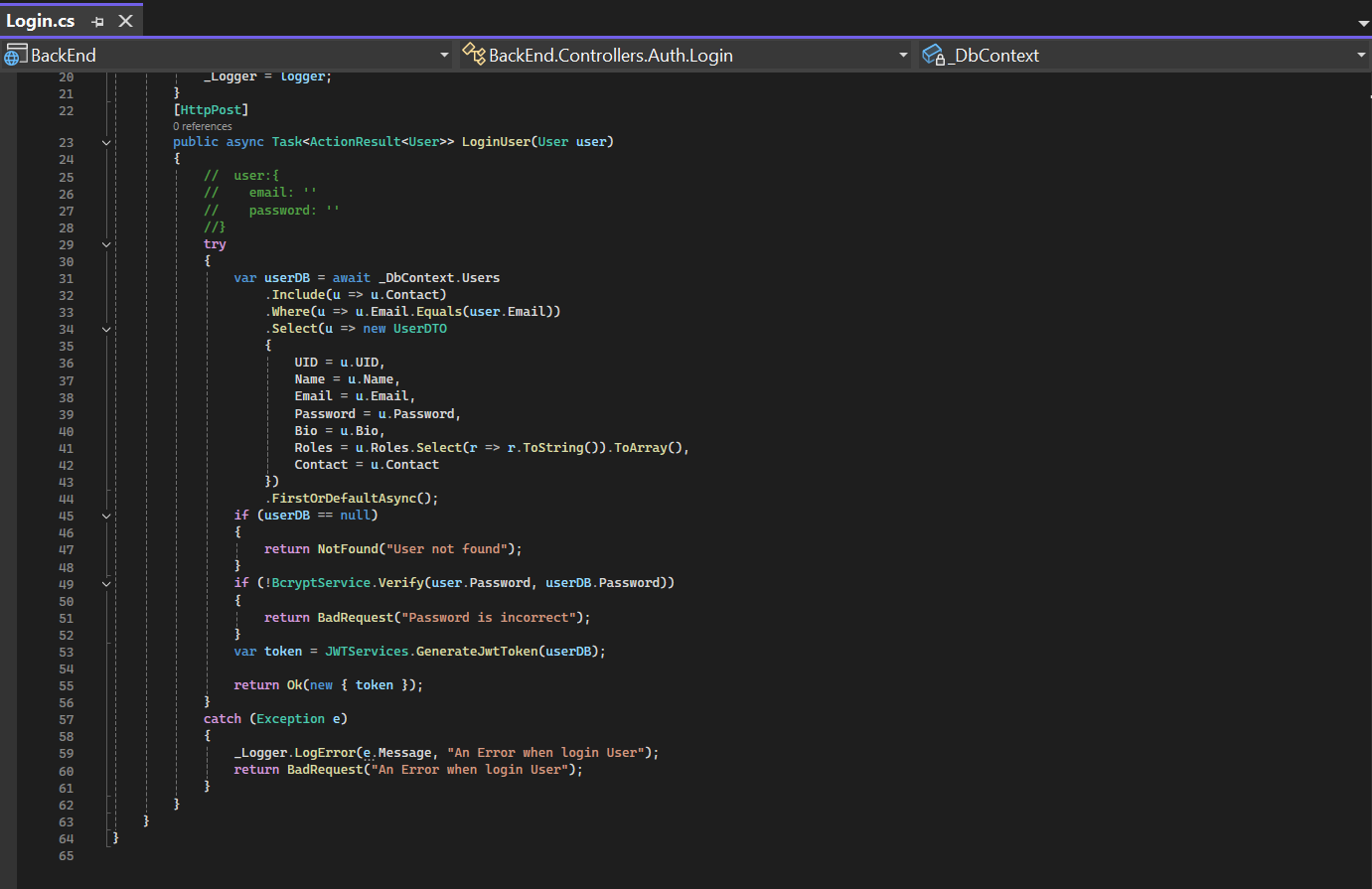


Hình 8: LoginPage.vue



Hình 9: LoginForm.vue

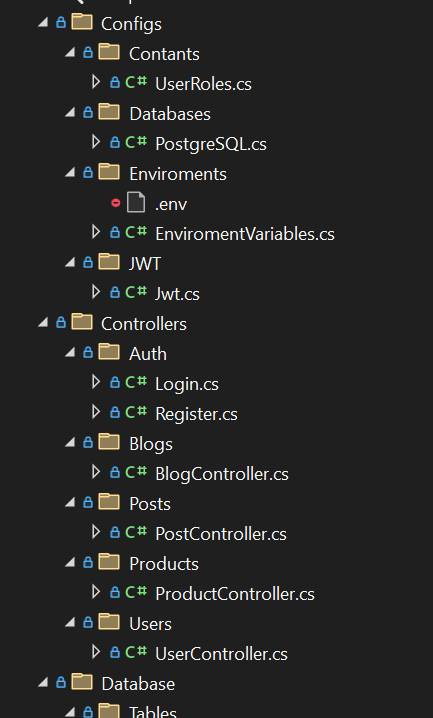
Login - BE:



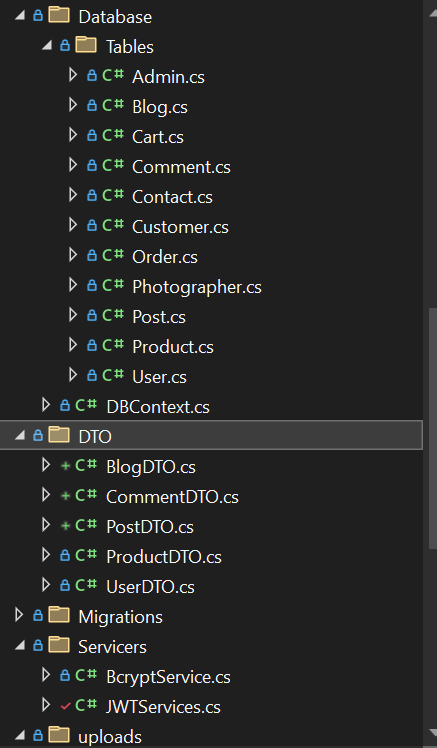
Hình 10: Login.cs

## Cấu trúc dự án

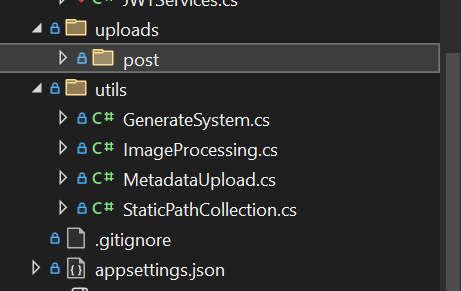
Dự án blog của tôi được phát triển theo kiến trúc microservices, cho phép ứng dụng trở nên linh hoạt, dễ mở rộng và bảo trì hơn. Dưới đây là các thành phần chính trong cấu trúc microservices của dự án:



Hình 11: Project Structure (1)



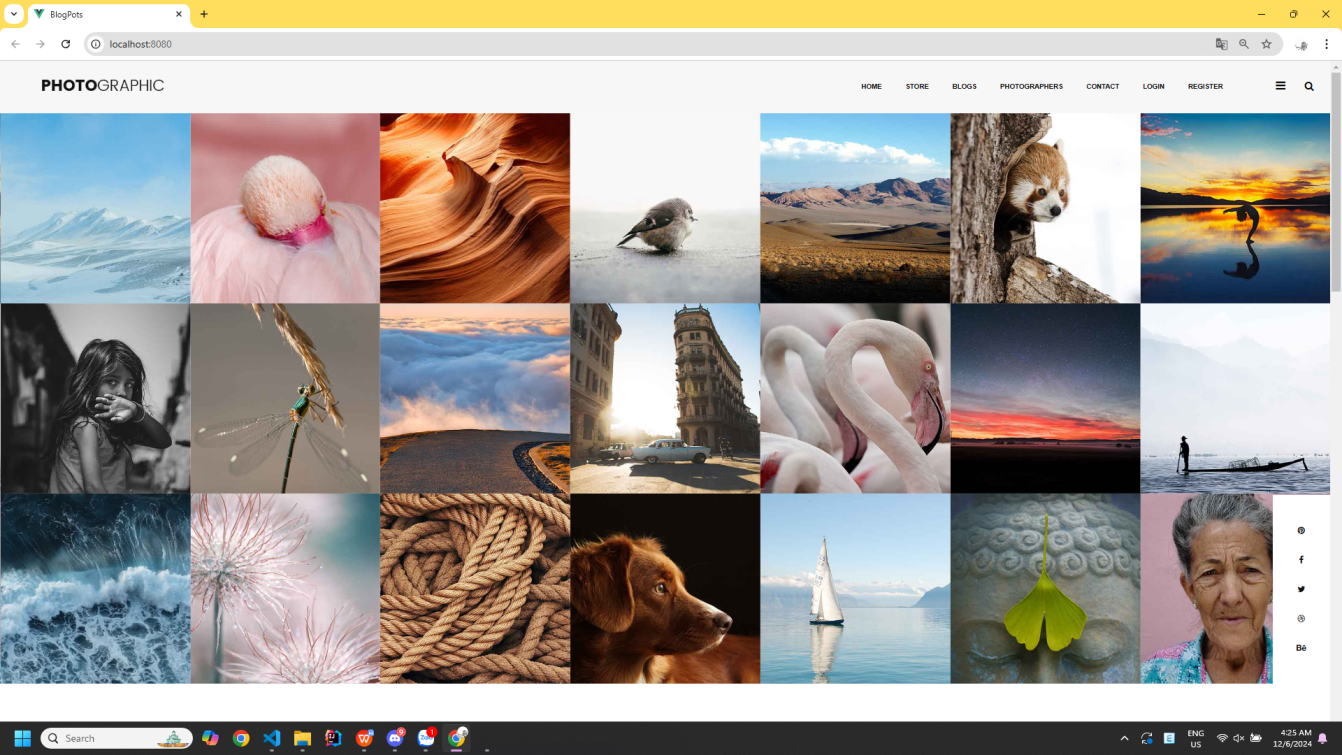
Hình 12: Project Structure (2)



Hình 13: Project Structure (3)

## Giao diện Client Side

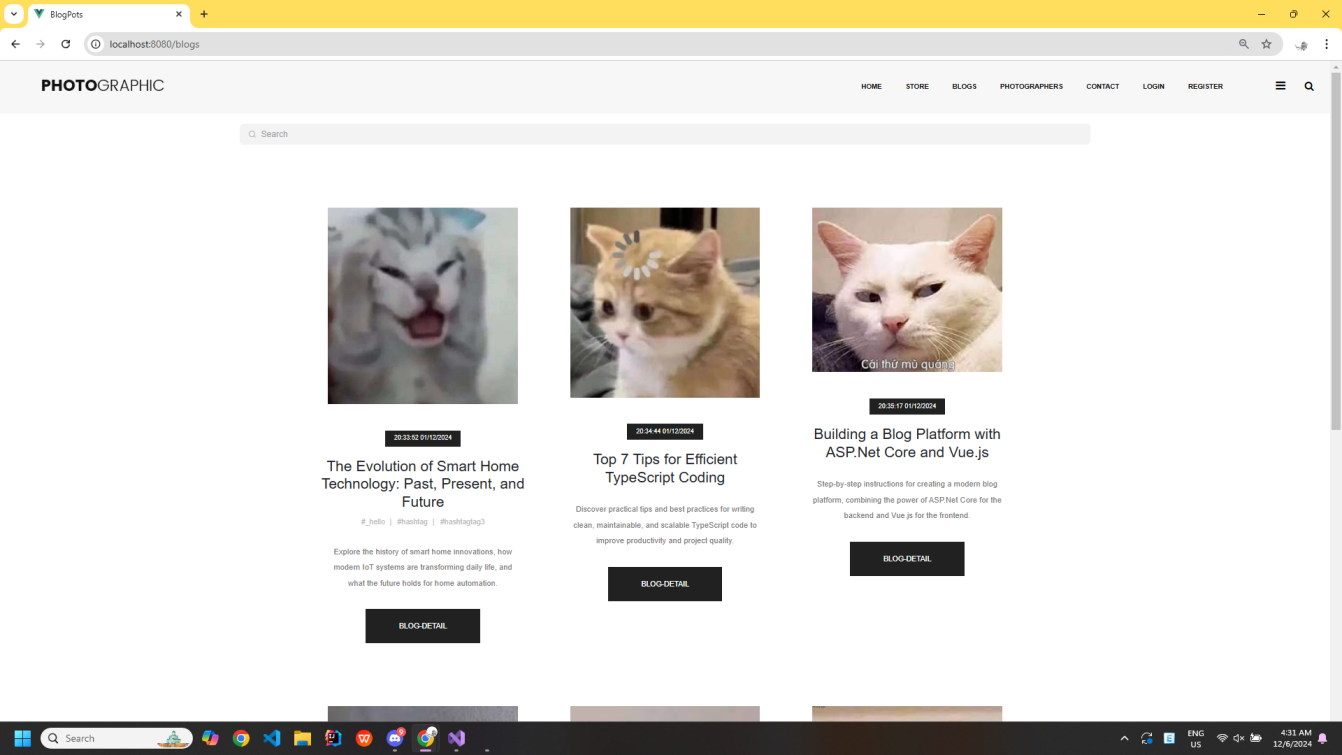
### Home



Hình 14: Home UI

### Blogs List

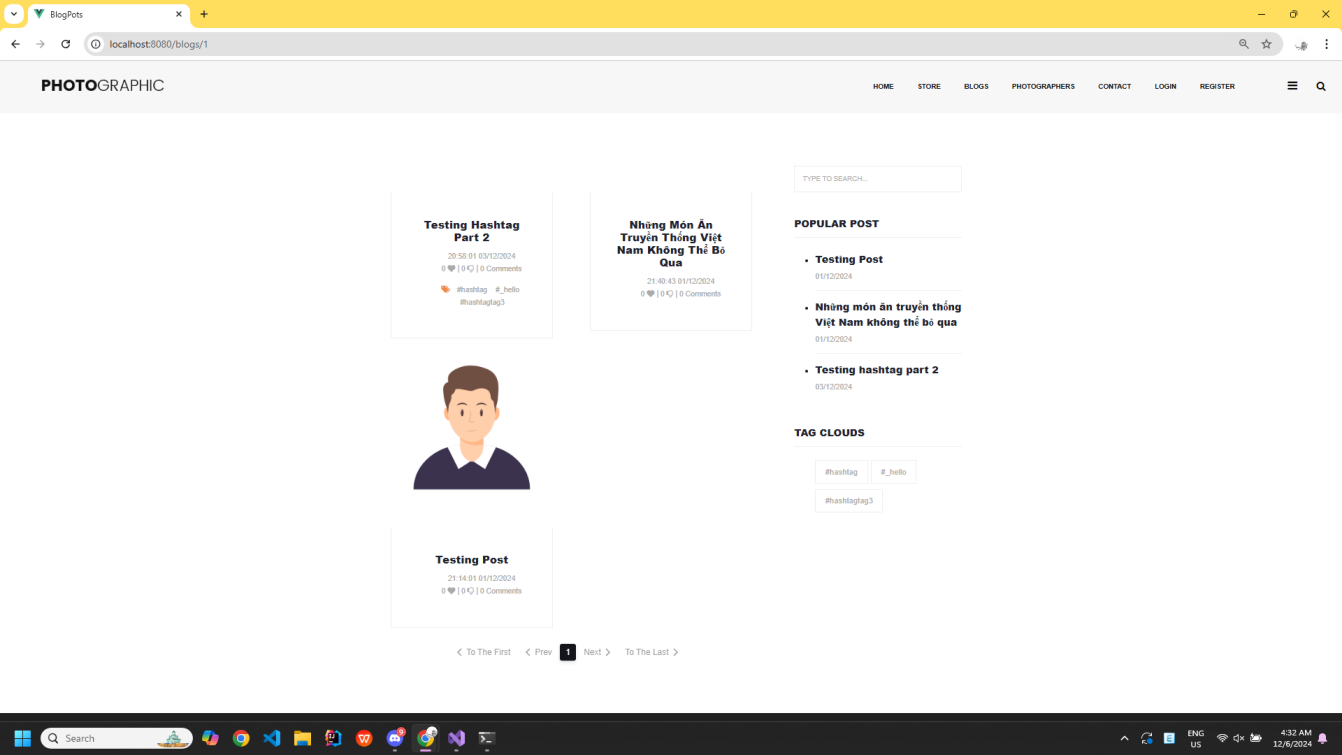
Route truy cập List Blog chung được public của các user có trong hệ thống. Tương tự như Route Discovery (Khám phá) trong những hệ sinh thái tương tự.



Hình 15: Blogs List UI

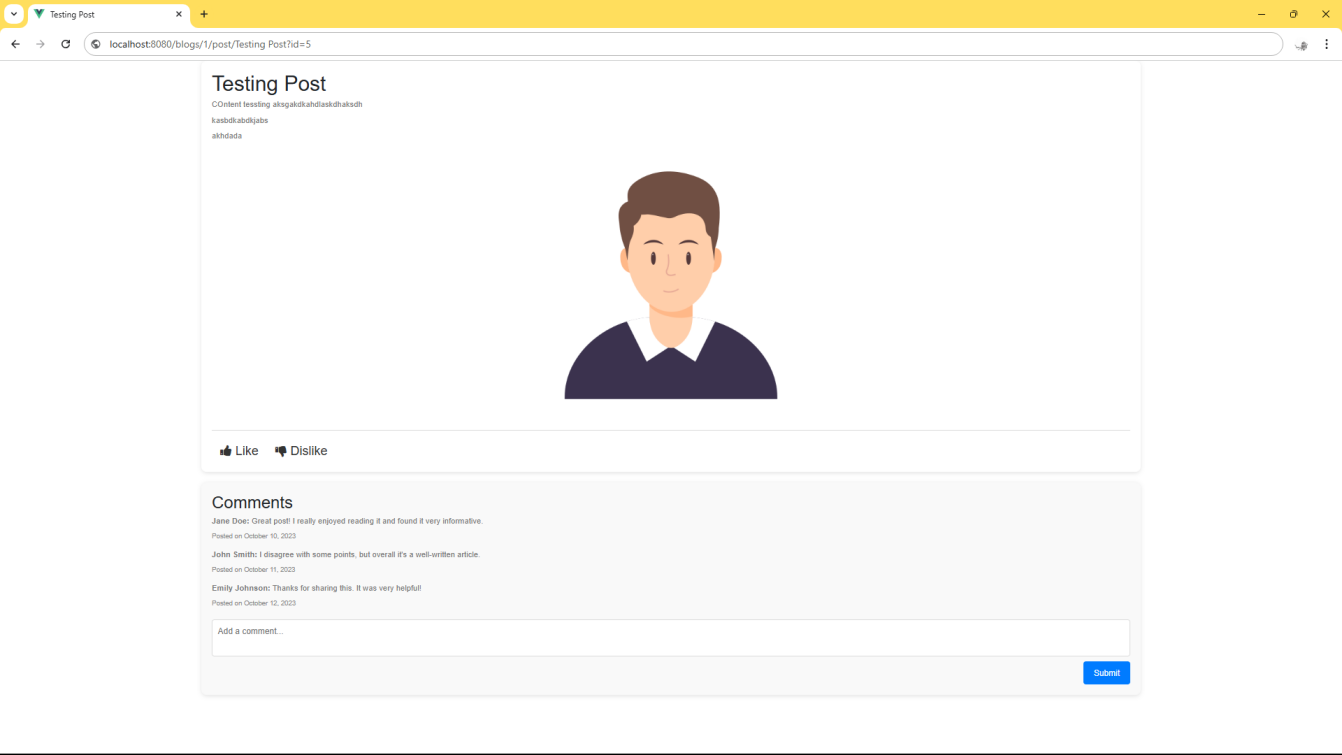
### Blog Detail

Route về chi tiết của 1 bài blog bao gồm các bài post được public thuộc bài blog đó. Bên trong Route […] là những slug sẽ được custom riêng tùy vào tình huống để đảm bảo việc SEO là hợp lý trong quá trình phát triển.



Hình 16: Blog Detail UI

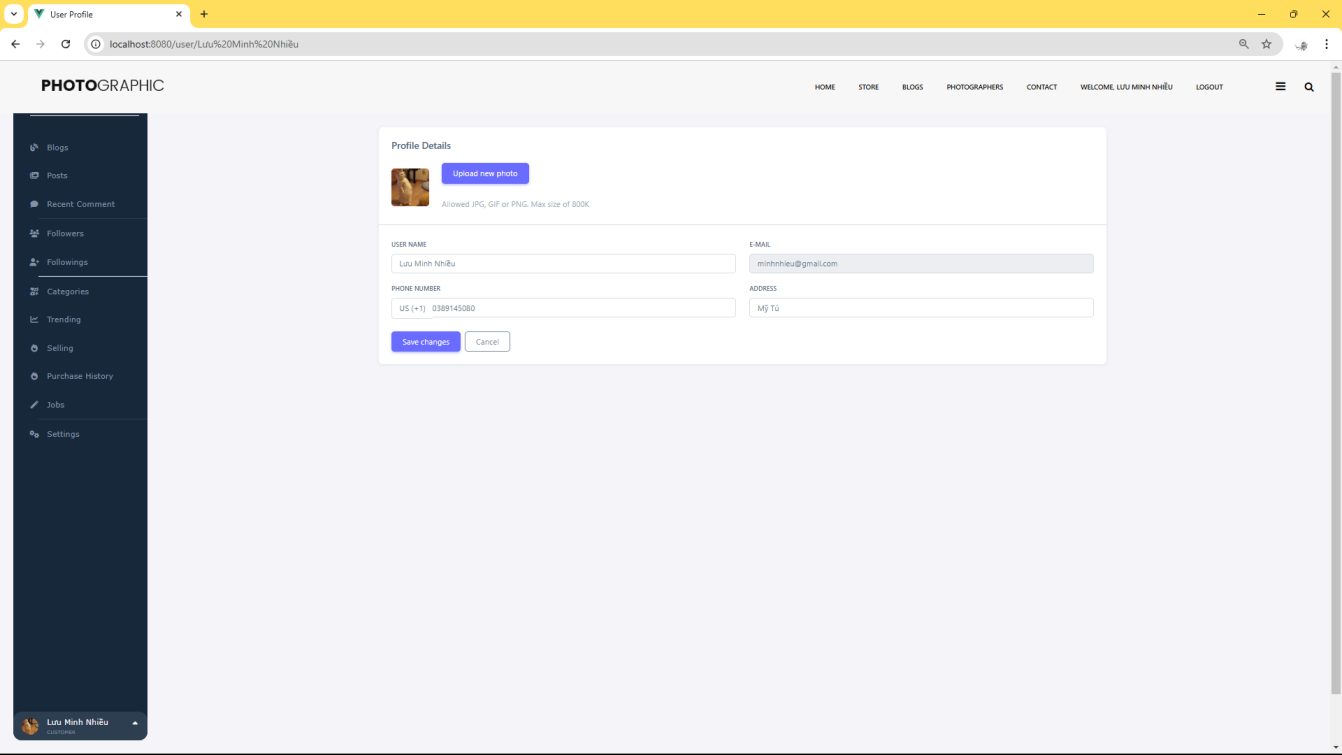
### Post Detail



Hình 17: Post Detail UI

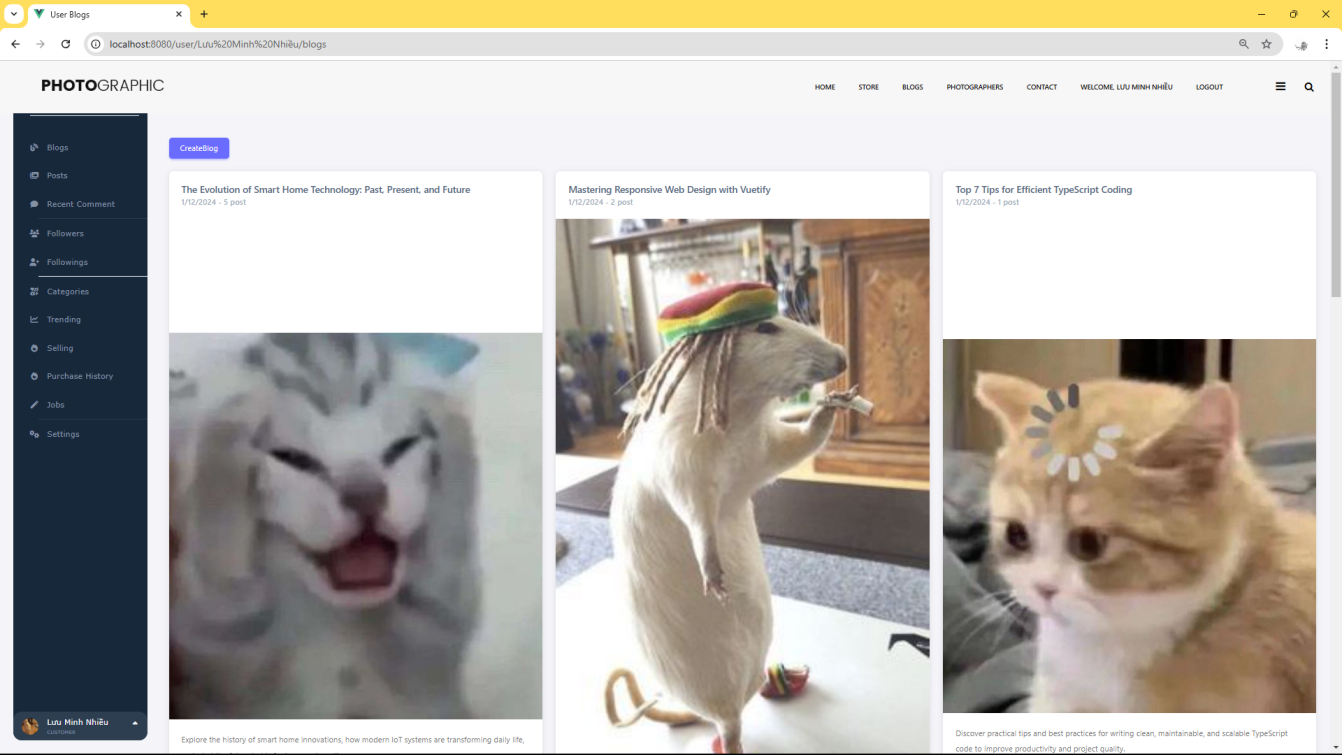
### User Profile

Route này dùng để người dùng quản lý trang cá nhân của bản thân bao gồm: các blogs, các posts, comment,… cùng với đó là các chức năng khác của một user có trong hệ thống được tương tác.



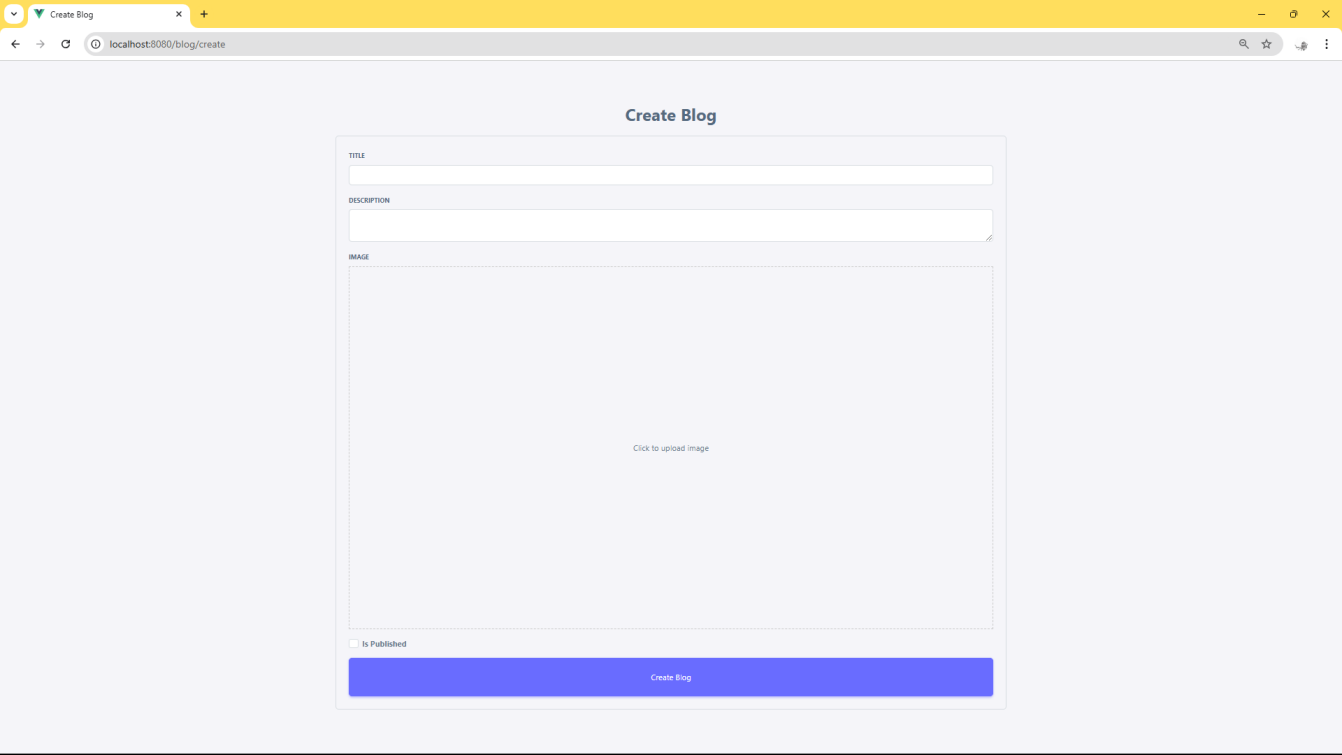
Hình 18: User Profile UI

### Blog management:



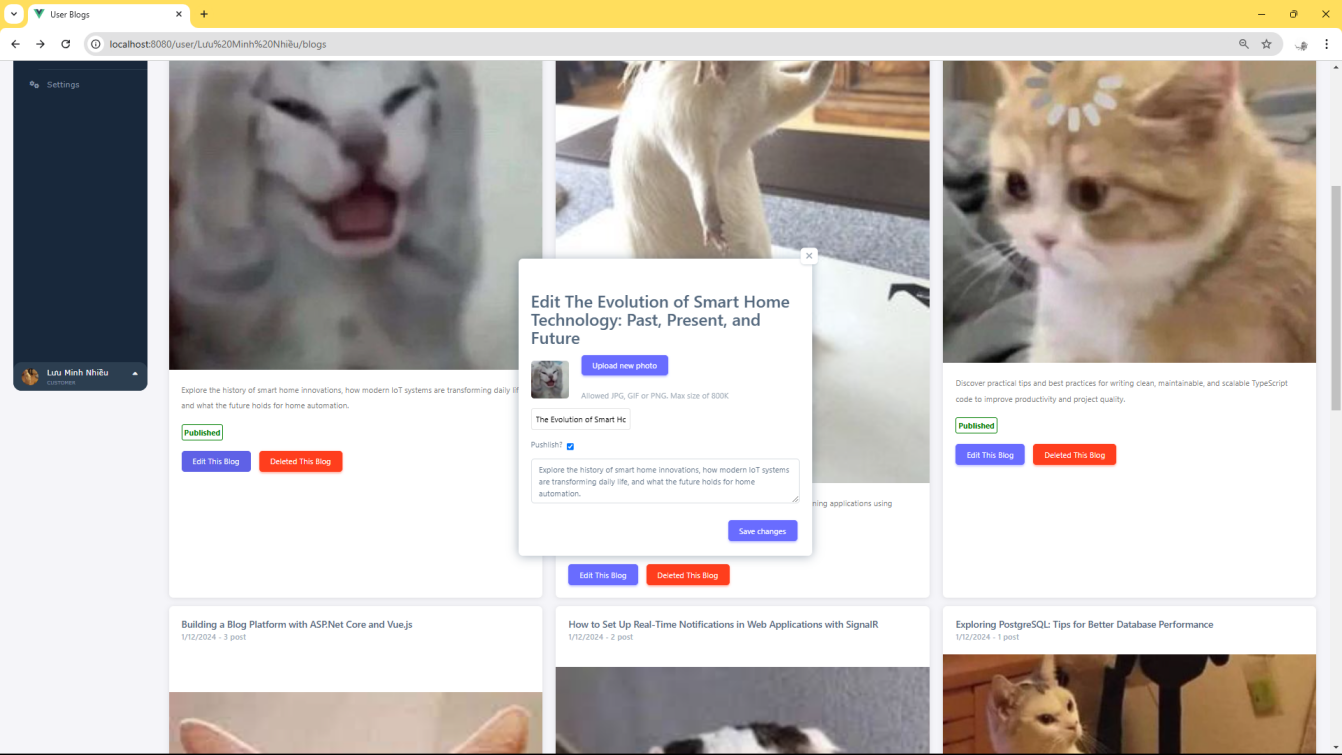
Hình 19: Blog management

### Blog Create:



Hình 20: Blog Create

### Blog Edit:



Hình 21: Blog Edit

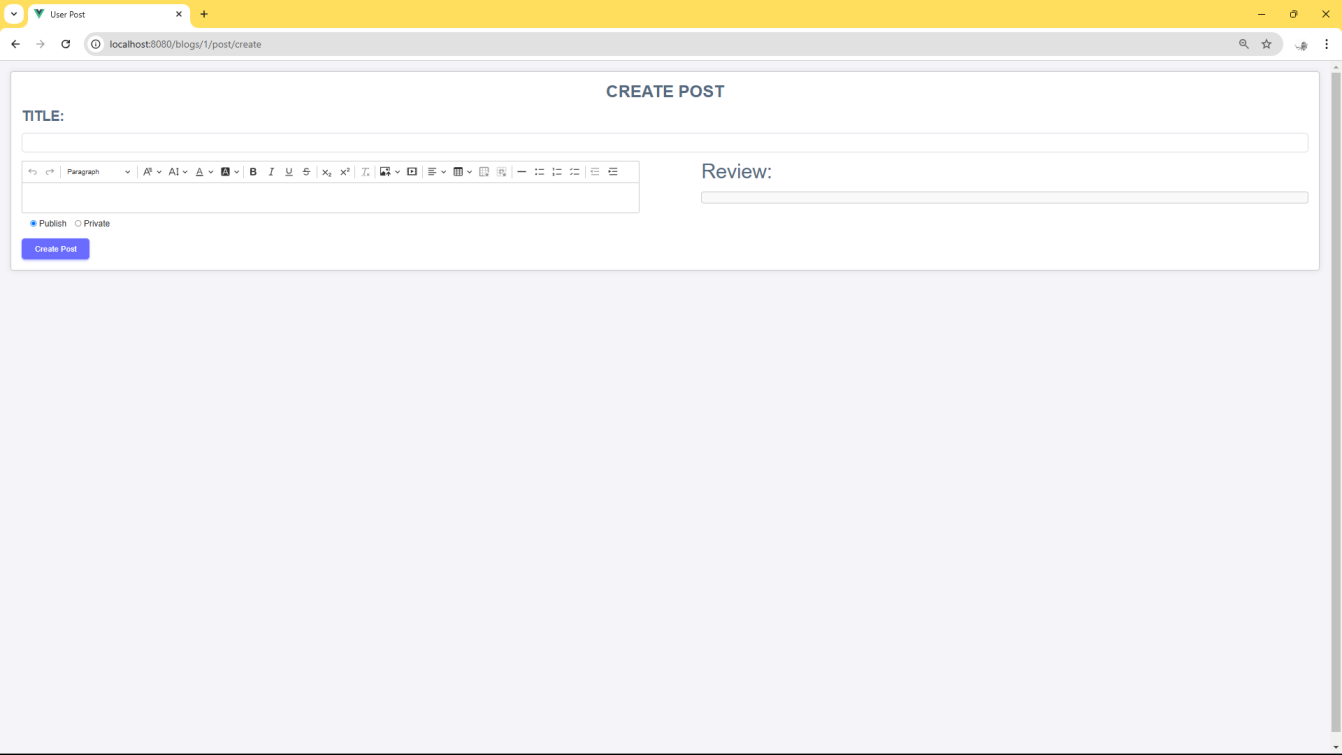
### Post management:



Hình 22: Post management

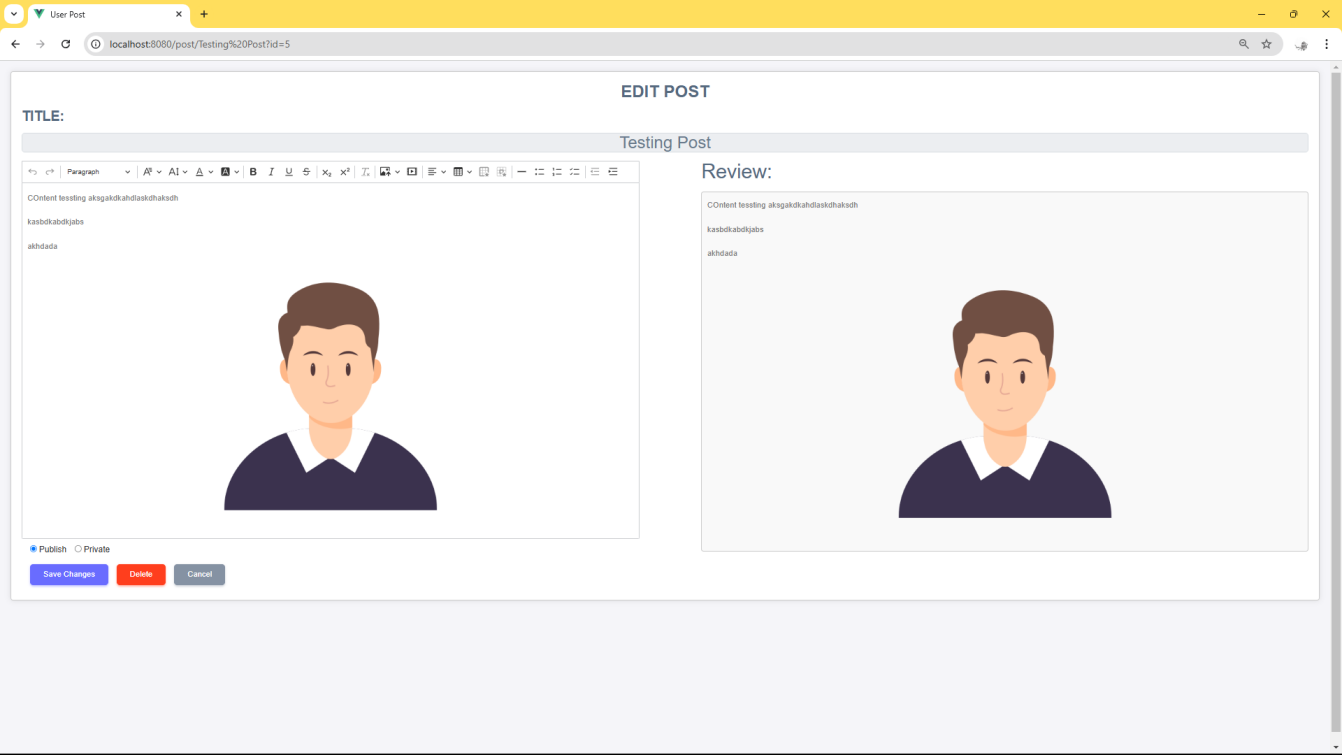
### Post Create

Route dùng để tạo Post trong 1 blog cụ thể, những gì người dùng tạo ra trong frame sẽ được render ra đúng như những gì người dùng thấy (What you see is what you get).



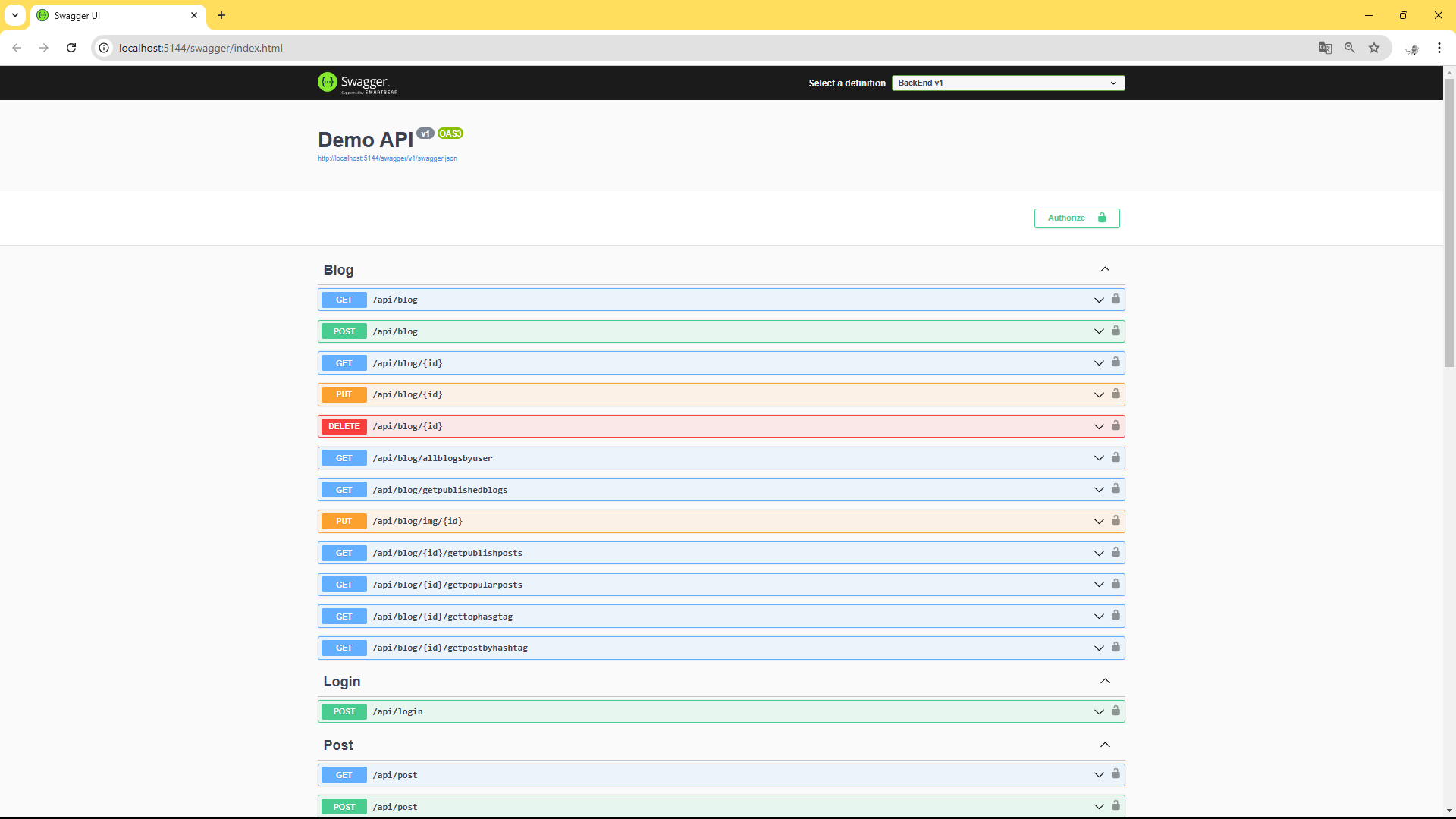
Hình 23: Post Create UI

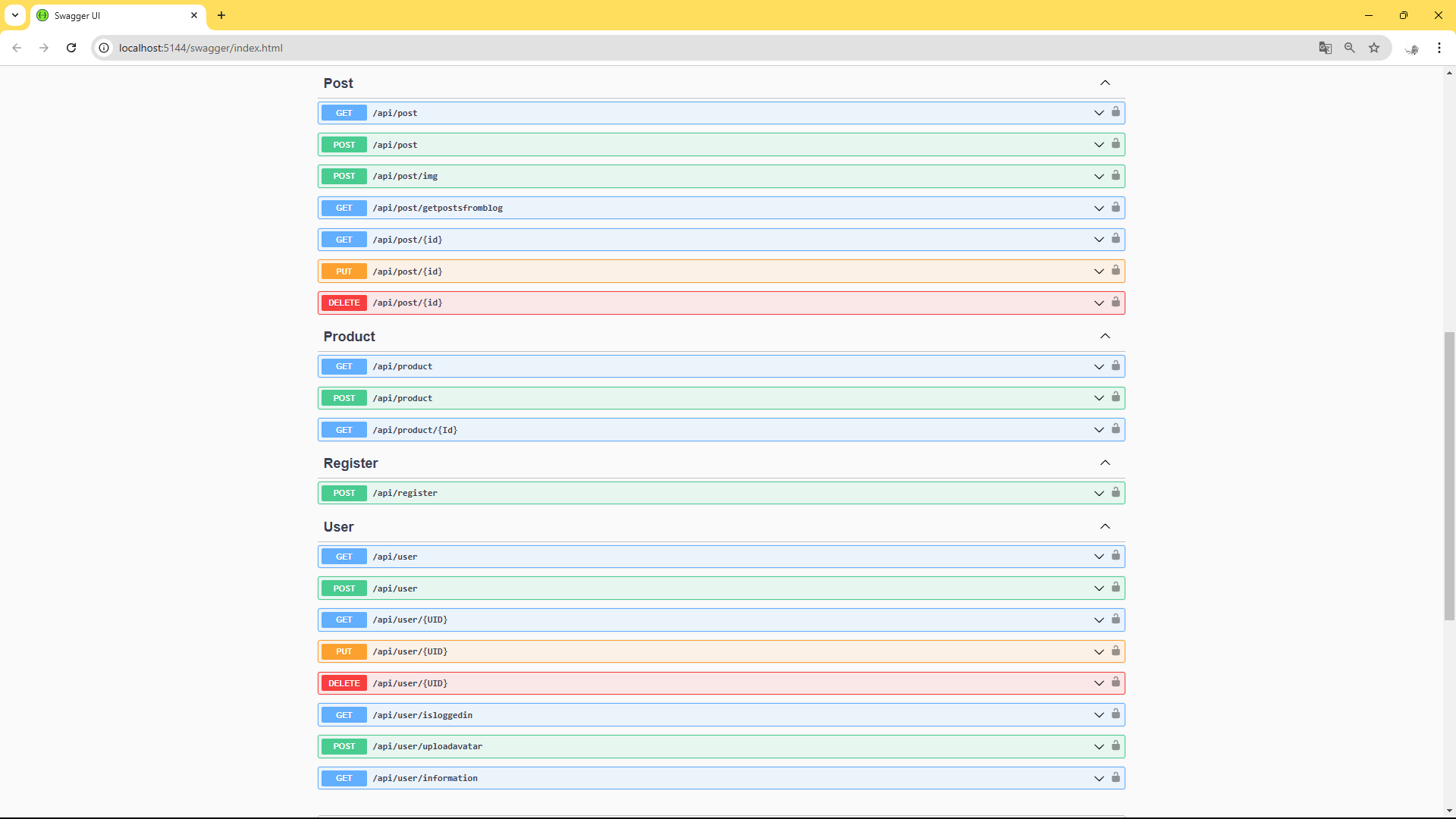
### Post Edit



Hình 24: Post Edit

## Tổng hợp Endpoint (Swagger):





# TỔNG KẾT VÀ ĐÁNH GIÁ

Dự án Blogpots Web sử dụng ASP.NET Core API cho phép xây dựng một hệ thống quản lý bài viết blog với các chức năng CRUD (Create, Read, Update, Delete) thông qua API. Phần backend được triển khai bằng ASP.NET Core API để quản lý dữ liệu từ cơ sở dữ liệu (PostgreSQL) thông qua EFCore, trong khi phần frontend sử dụng các công nghệ HTML, CSS, SASS và Javascript hoặc các framework hiện đại (Vue JS) để tạo ra giao diện người dùng thân thiện.

Sự tích hợp giữa API và giao diện người dùng đảm bảo tính tương tác cao, mang lại trải nghiệm mượt mà cho người dùng khi truy cập và quản lý nội dung trên web.

Dự án này có quy mô rất lớn và yêu cầu nhiều thời gian, công sức để hoàn thành toàn diện. Tuy nhiên, do thời gian hạn chế và công việc bận rộn, tôi chưa thể dành đủ thời gian như mong muốn để hoàn thiện mọi khía cạnh của đề tài. Vì vậy, tôi chỉ có thể hoàn thành phần lớn các yêu cầu cơ bản của dự án, vẫn còn một số chi tiết và tính năng chưa được triển khai đầy đủ. Mặc dù vậy, tôi cam kết sẽ tiếp tục phát triển, cải tiến và bổ sung thêm các phần còn thiếu để dự án đạt được chất lượng tốt nhất trong thời gian tới.

Link github: https://github.com/So-Much/ASPdotNet

# TÀI LIỆU THAM KHẢO

Tiếng Việt

…

Tiếng Anh

https://learn.microsoft.com/fr-fr/aspnet/core/?view=aspnetcore-8.0&WT.mc\_id=dotnet-35129-website.