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 vii](#_Toc7149)

[DANH MỤC BẢNG BIỂU viii](#_Toc2371)

[DANH MỤC CÁC CHỮ VIẾT TẮT ix](#_Toc11059)

[CHƯƠNG 1. MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI 1](#_Toc13689)

[1.1 Lý do chọn đề tài 1](#_Toc31889)

[1.2 Kiến trúc tổng thể 1](#_Toc4865)

[1.3 Sơ đồ ERD (Tạm Thời) 2](#_Toc29757)

[1.4 Khác biệt giữa Client Side Rendering và Server Side Rendering 2](#_Toc27600)

[CHƯƠNG 2. CHI TIẾT DỰ ÁN 3](#_Toc22641)

[2.1 Auth: Authentication and Authorization 3](#_Toc26867)

[2.1.1 Authentication 3](#_Toc752)

[2.1.2 Authorization 3](#_Toc9819)

[2.2 Cấu trúc dự án 13](#_Toc11297)

[2.3 Giao diện Client Side 15](#_Toc15917)

[2.3.1 Home (Route: /) 16](#_Toc929)

[2.3.2 Blogs List (Route: /blogs) 17](#_Toc20336)

[2.3.3 Blog Detail (Tạm Thời) (Route: /blogs/[blogId, blogname, blogtitle…]) 18](#_Toc10930)

[2.3.4 User Profile (Tạm Thời) (Route: /user/[userId]/[Route~]) 21](#_Toc25422)

[2.3.5 Post Create (Tạm Thời) (Route: blog/[…]/post/create) 21](#_Toc1792)

[CHƯƠNG 3. TỔNG KẾT 23](#_Toc27976)

[CHƯƠNG 4. KẾT LUẬN 24](#_Toc22660)

[4.1 Kết luận 24](#_Toc17925)

[4.2 Hướng phát triển 24](#_Toc5011)

[TÀI LIỆU THAM KHẢO 25](#_Toc15784)

# DANH MỤC HÌNH VẼ

[Hình 1 : Sơ đồ ERD 2](#_Toc26073)

[Hình 2 : Register UI 4](#_Toc15202)

[Hình 3 : RegisterPage.vue 5](#_Toc11869)

[Hình 4 : RegisterForm.vue 6](#_Toc23105)

[Hình 5 : Register.cs 7](#_Toc26930)

[Hình 6 : Login UI 8](#_Toc16807)

[Hình 7 : LoginPage.vue 9](#_Toc25101)

[Hình 8 : LoginForm.vue 10](#_Toc25241)

[Hình 9 : Login.cs 11](#_Toc5810)

[Hình 10 : JWTServices.cs 11](#_Toc21138)

[Hình 11 : BcryptService.cs 12](#_Toc16148)

[Hình 12 : Project Structure (1) 14](#_Toc13512)

[Hình 13 : Project Structure (2) 14](#_Toc8382)

[Hình 14 : Project Structure (3) 15](#_Toc18608)

[Hình 15 : Home UI 17](#_Toc12657)

[Hình 16 : Blogs List UI 18](#_Toc16836)

[Hình 17 : Blog Detail UI 21](#_Toc19727)

[Hình 18 : User Profile UI 21](#_Toc19588)

[Hình 19 : Post Create UI 22](#_Toc695)

# DANH MỤC BẢNG BIỂU

[Bảng 4.1: Thống kê kiểu thực thể trong tập VLSP 2016 4](#_Toc142677565)

# 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

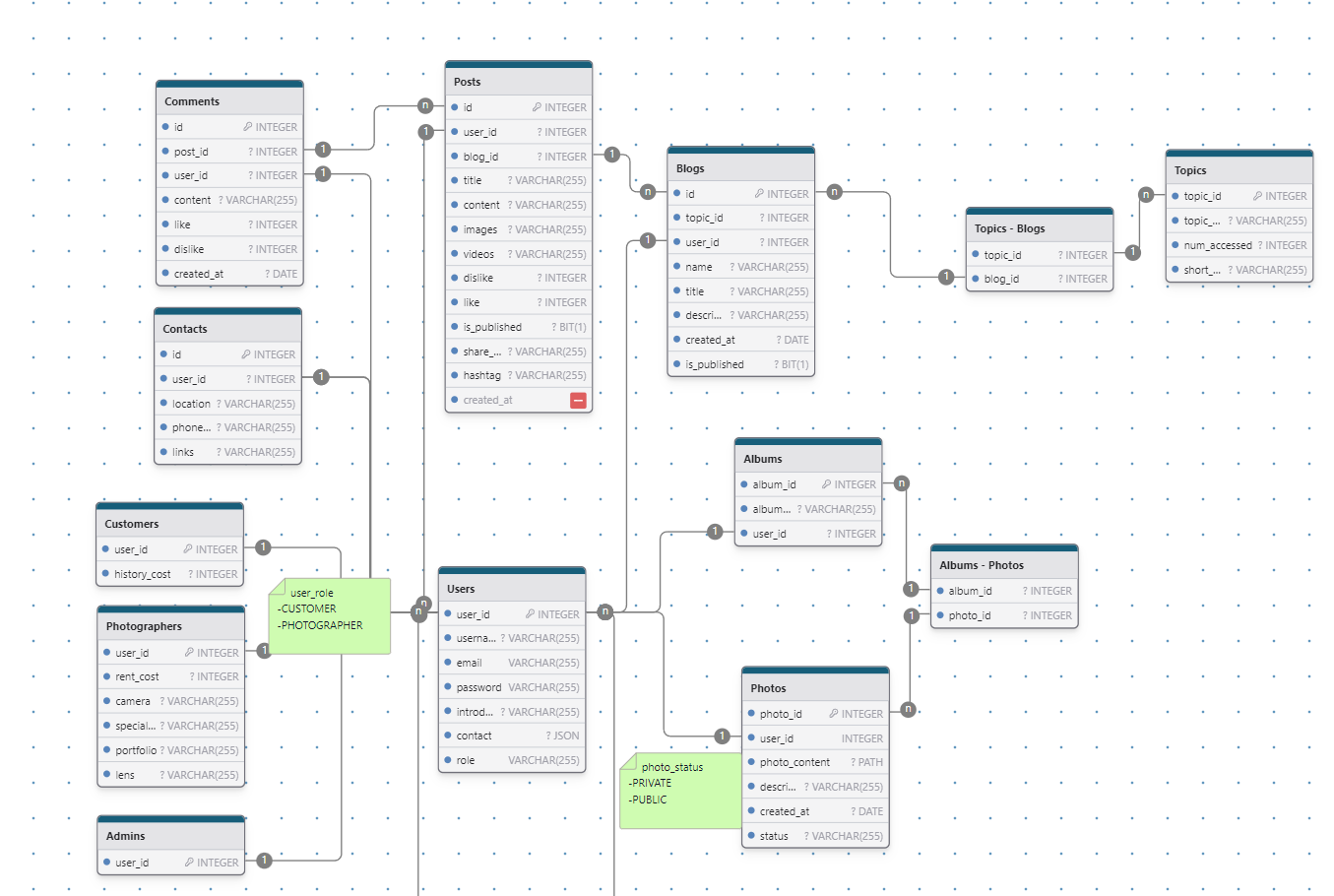
* **Hiệu suất:** Hệ thống phải hoạt động ổn định, có thời gian phản hồi nhanh.
* **Bảo mật:**
  + Bảo vệ thông tin người dùng.
  + Ngăn chặn các lỗ hổng bảo mật phổ biến như SQL Injection, XSS.
* **Khả năng mở rộng:** Thiết kế hệ thống linh hoạt, có thể bổ sung thêm chức năng trong tương lai.
* **Khả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

## Phân tích và thiết kế hệ thống

…

## Thiết kế cơ sở dữ liệu

### Sơ đồ ERD



Hình 1: 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();

        });

});

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();

### 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.

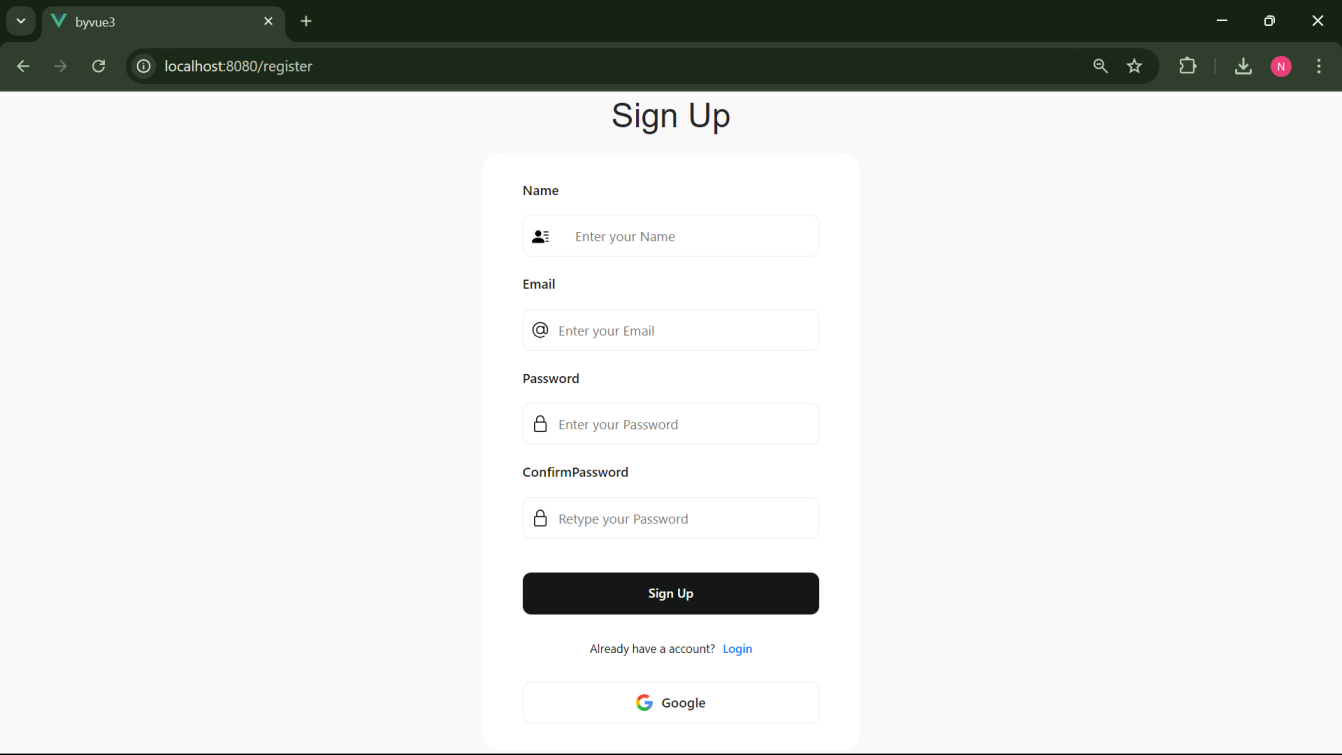
- Server sẽ giải mã token và kiểm tra thông tin người dùng, bao gồm vai trò và quyền hạn của họ.

Quy định quyền truy cập:

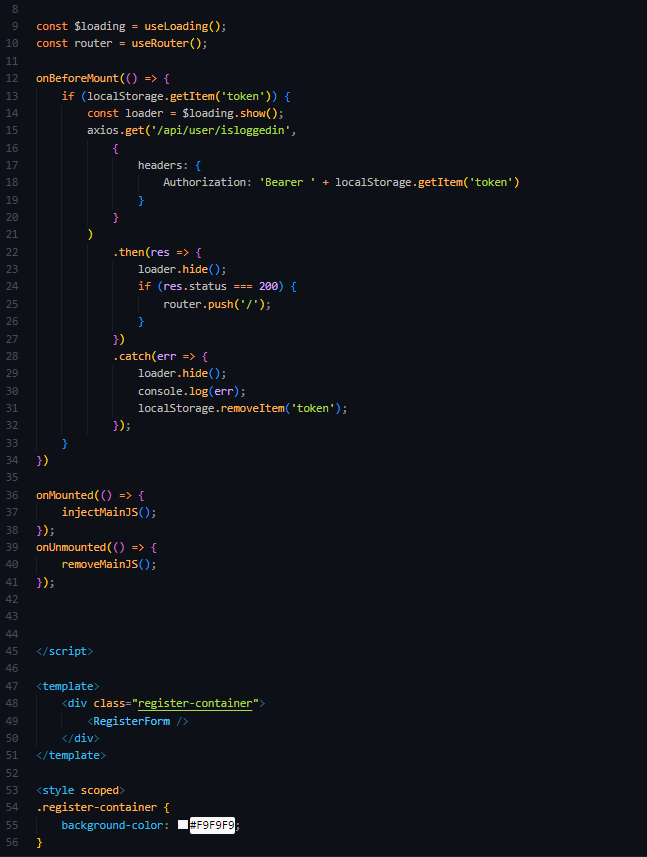
- Dựa trên vai trò của người dùng (ví dụ: quản trị viên, người dùng thường), hệ thống sẽ quyết định quyền truy cập vào các tài nguyên khác nhau.

- 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.

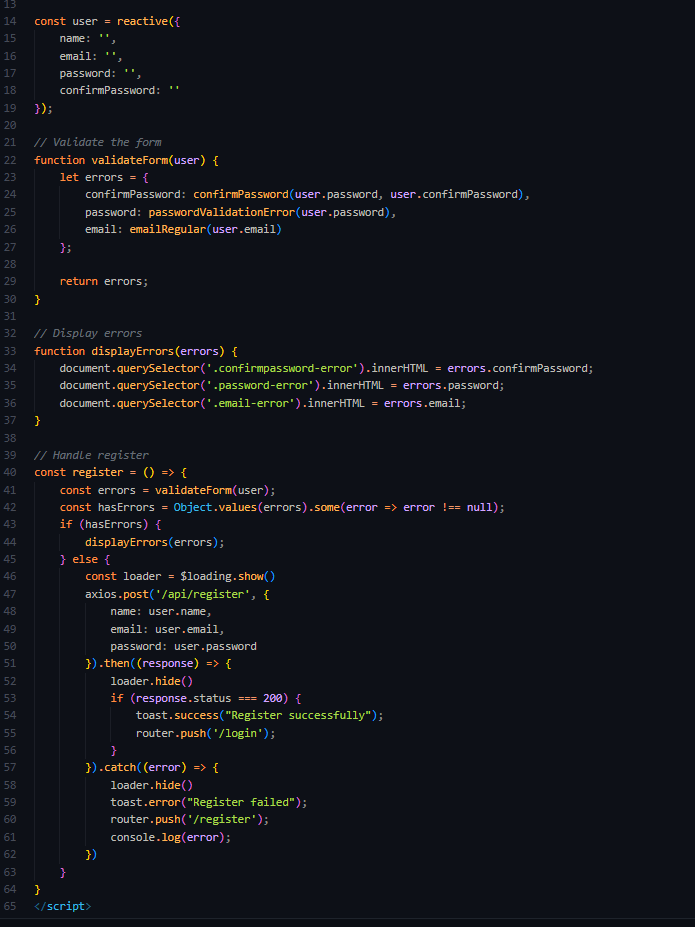
Register - FE:



Hình 2: Register UI

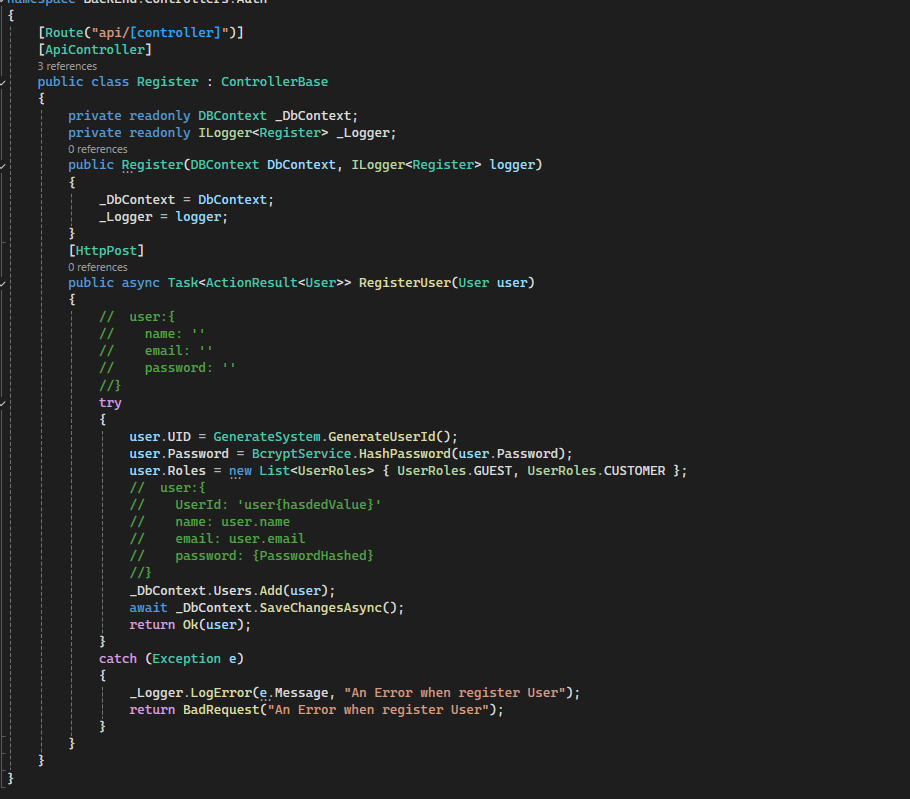


Hình 3: RegisterPage.vue



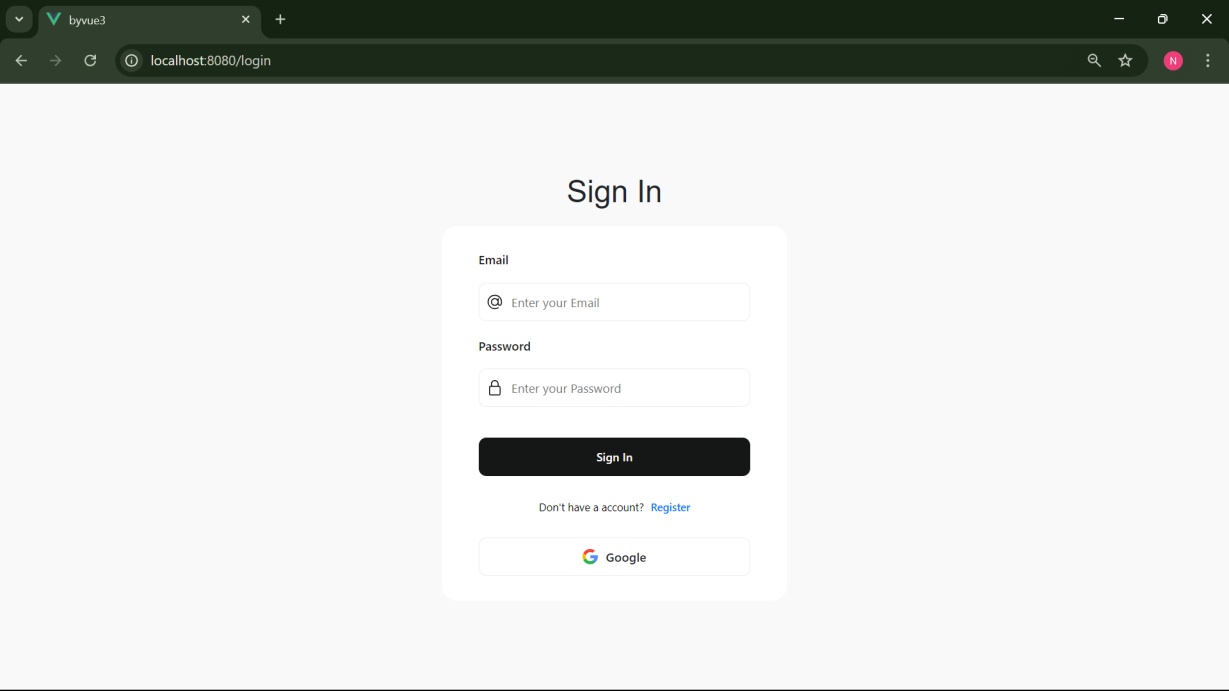
Hình 4: RegisterForm.vue

Register - BE:

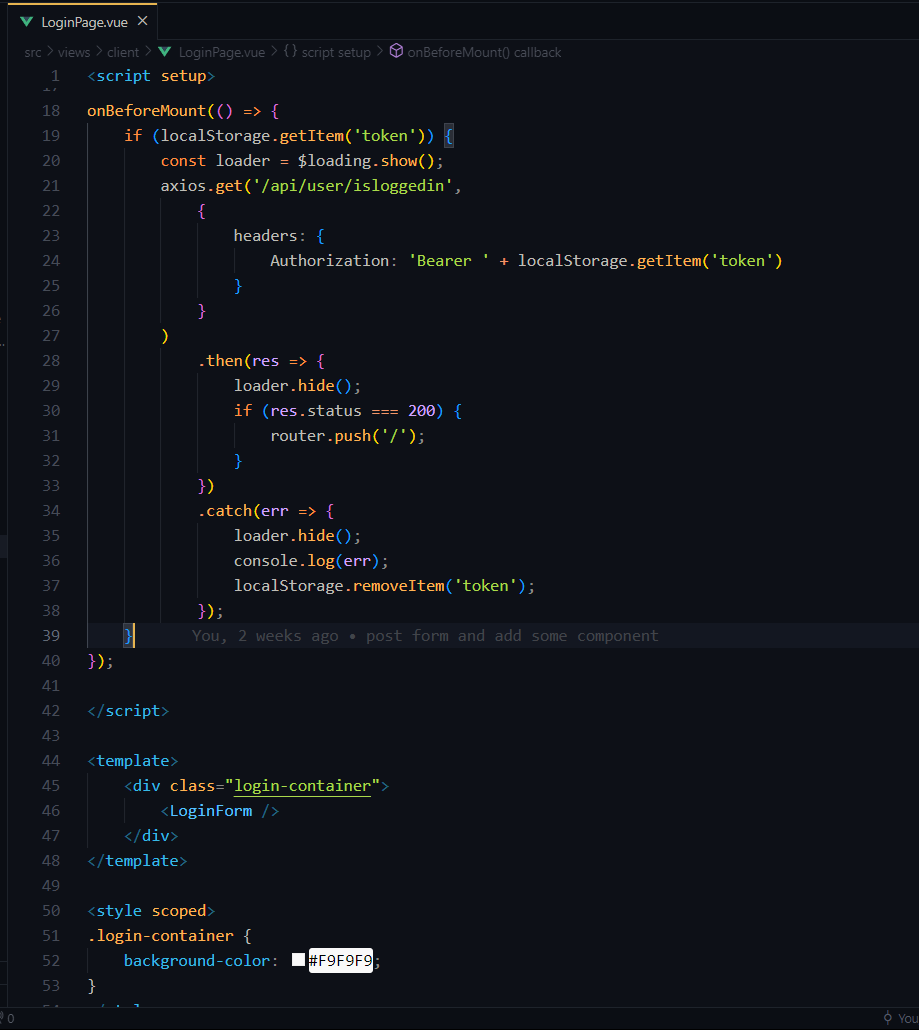


Hình 5: Register.cs

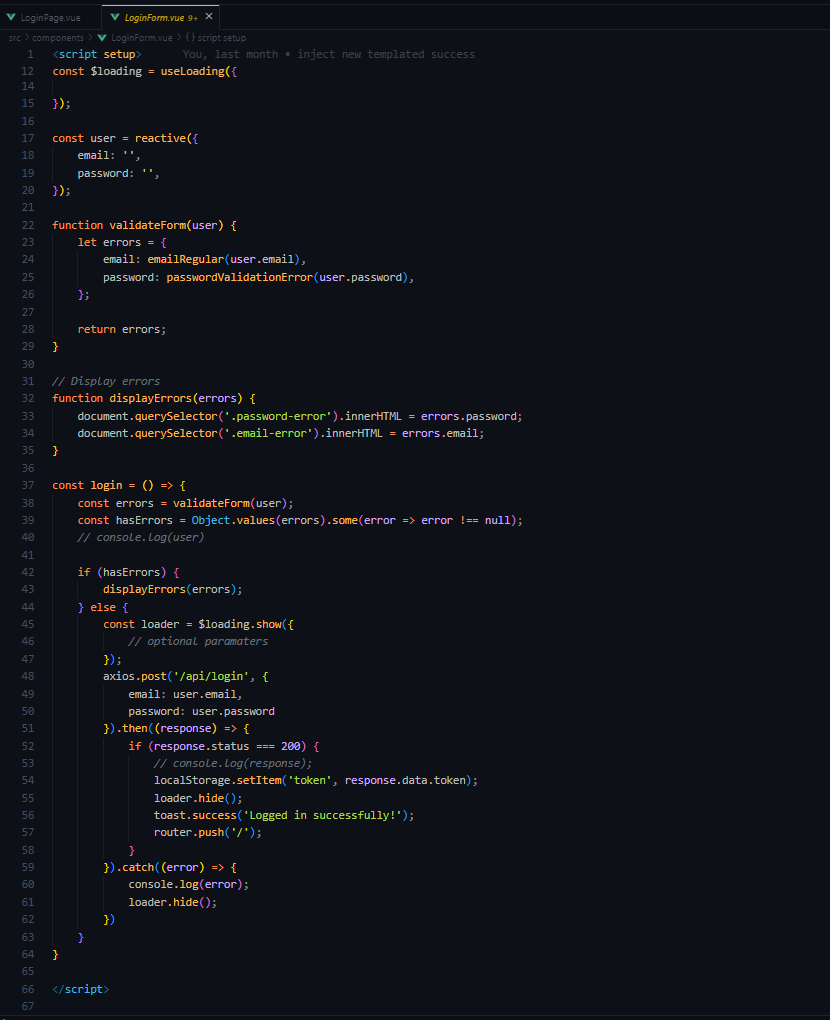
Login - FE:



Hình 6: Login UI

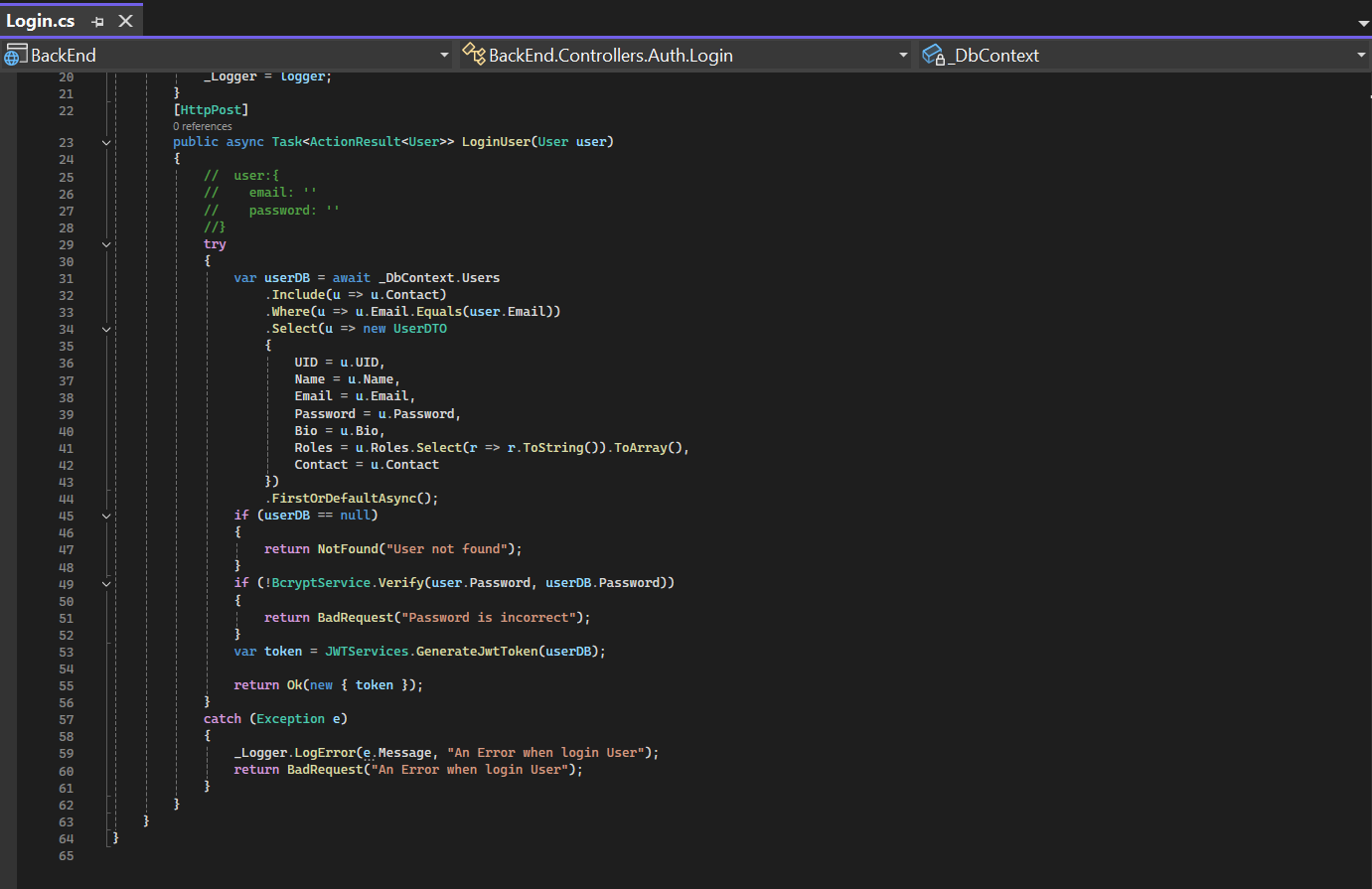


Hình 7: LoginPage.vue



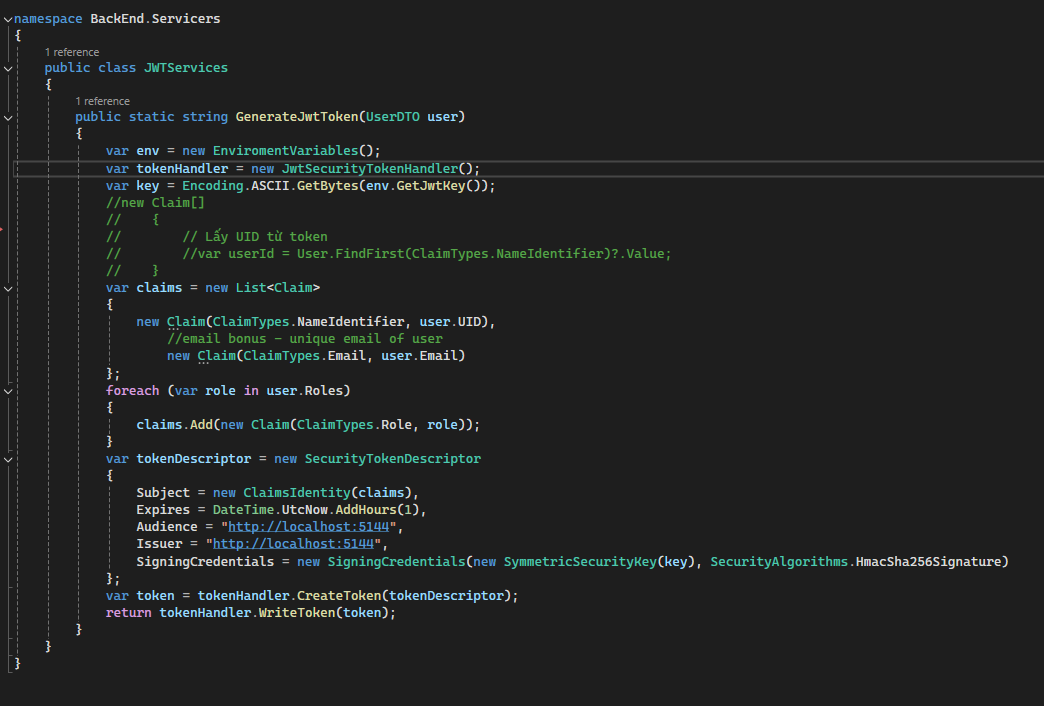
Hình 8: LoginForm.vue

Login - BE:

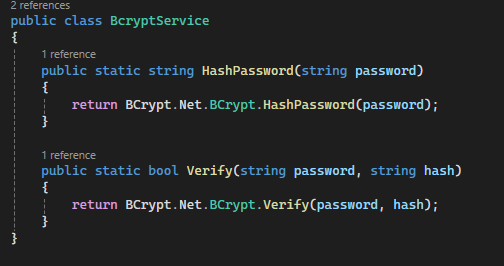


Hình 9: Login.cs

Services Supported:



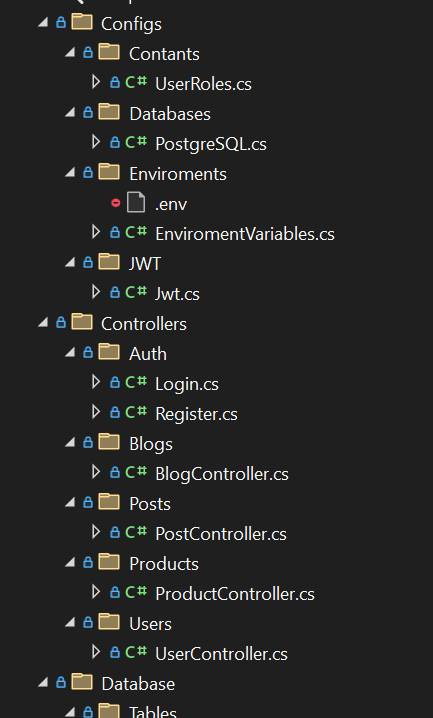
Hình 10: JWTServices.cs



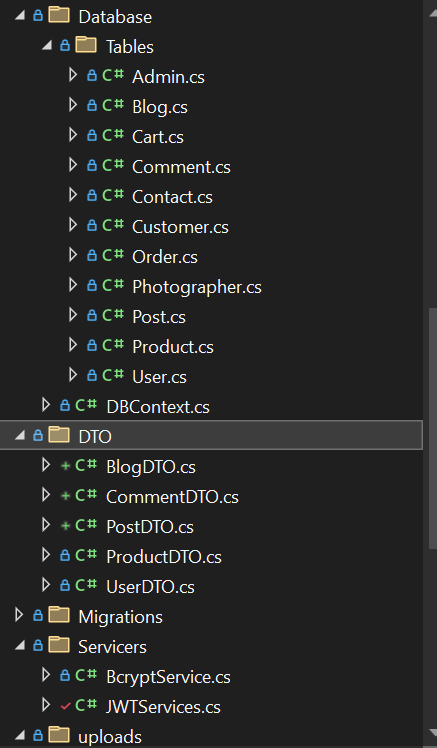
Hình 11: BcryptService.cs

## Cấu trúc dự án

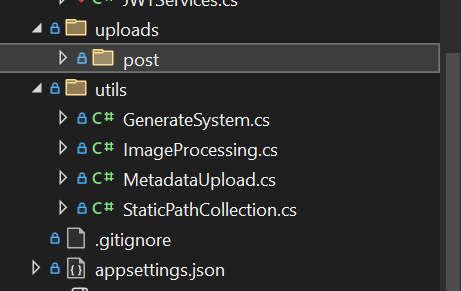
Dự án blog của tôi được phát triển theo kiến trúc microservices, cho phép ứng dụng trở nên linh hoạt, dễ mở rộng và bảo trì hơn. Dưới đây là các thành phần chính trong cấu trúc microservices của dự án:



Hình 12: Project Structure (1)



Hình 13: Project Structure (2)

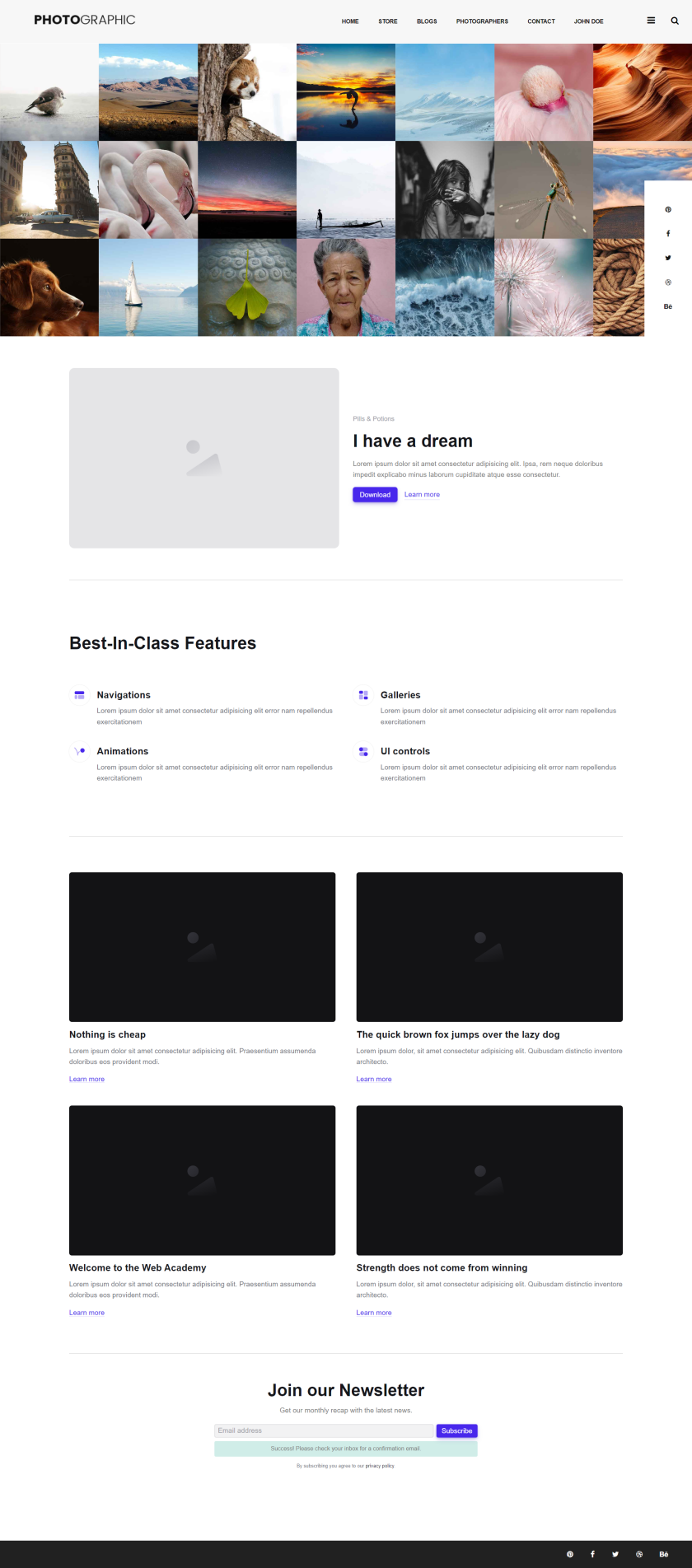


Hình 14: Project Structure (3)

## Giao diện Client Side

Vì đang trong giai đoạn phát triển nên tôi sẽ chỉ demo về UI của VueJS.

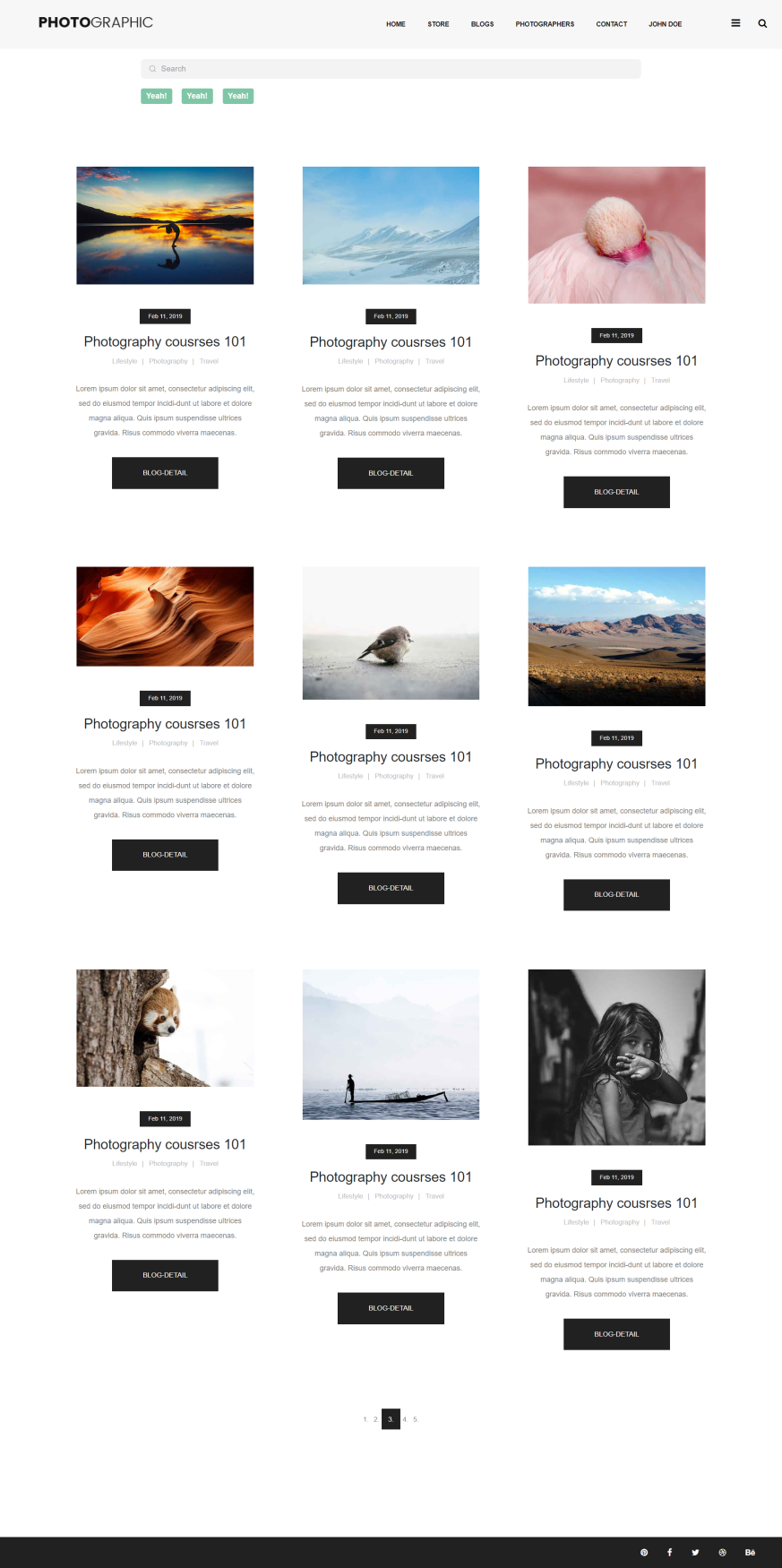
### Home (Route: /)



Hình 15: Home UI

### Blogs List (Route: /blogs)

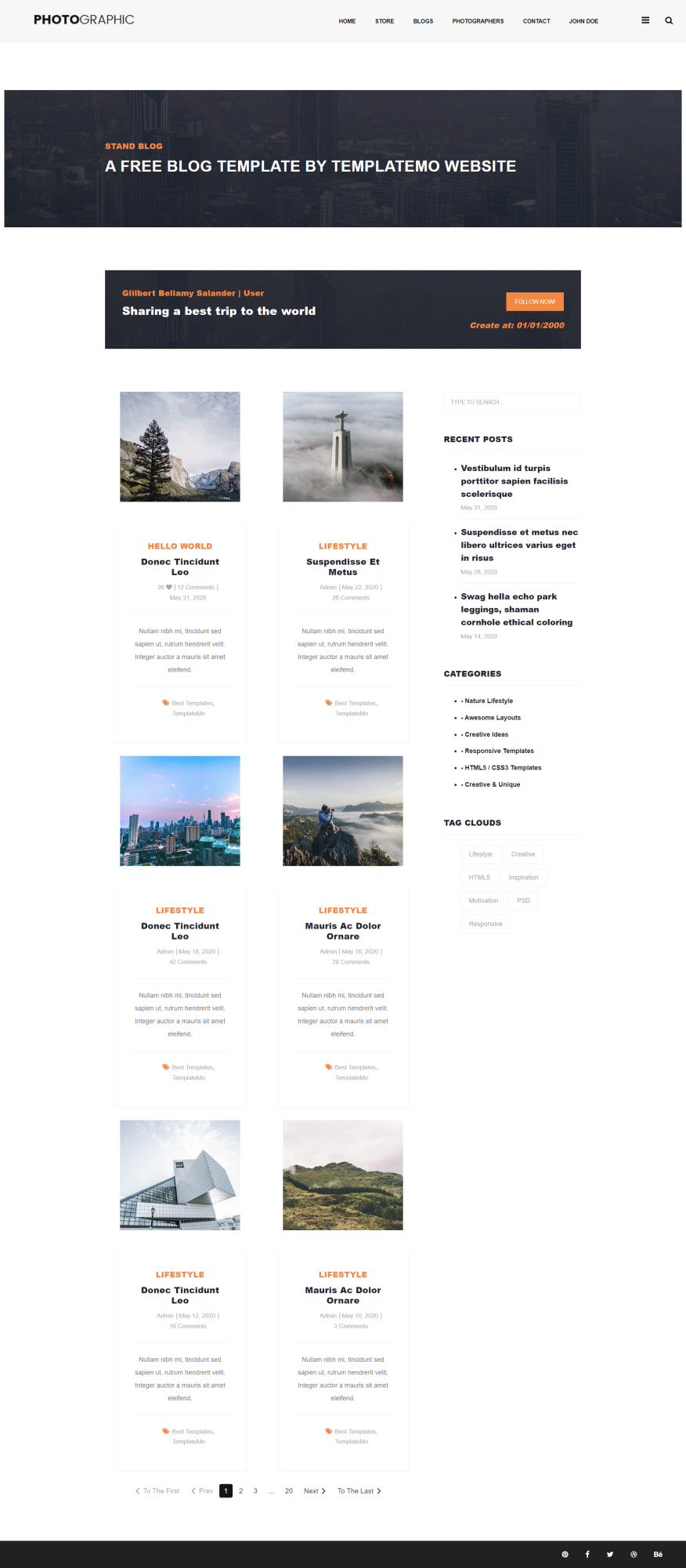
Route truy cập List Blog chung được public của các user có trong hệ thống. Tương tự như Route Discovery (Khám phá) trong những hệ sinh thái tương tự.



Hình 16: Blogs List UI

### Blog Detail (Tạm Thời) (Route: /blogs/[blogId, blogname, blogtitle…])

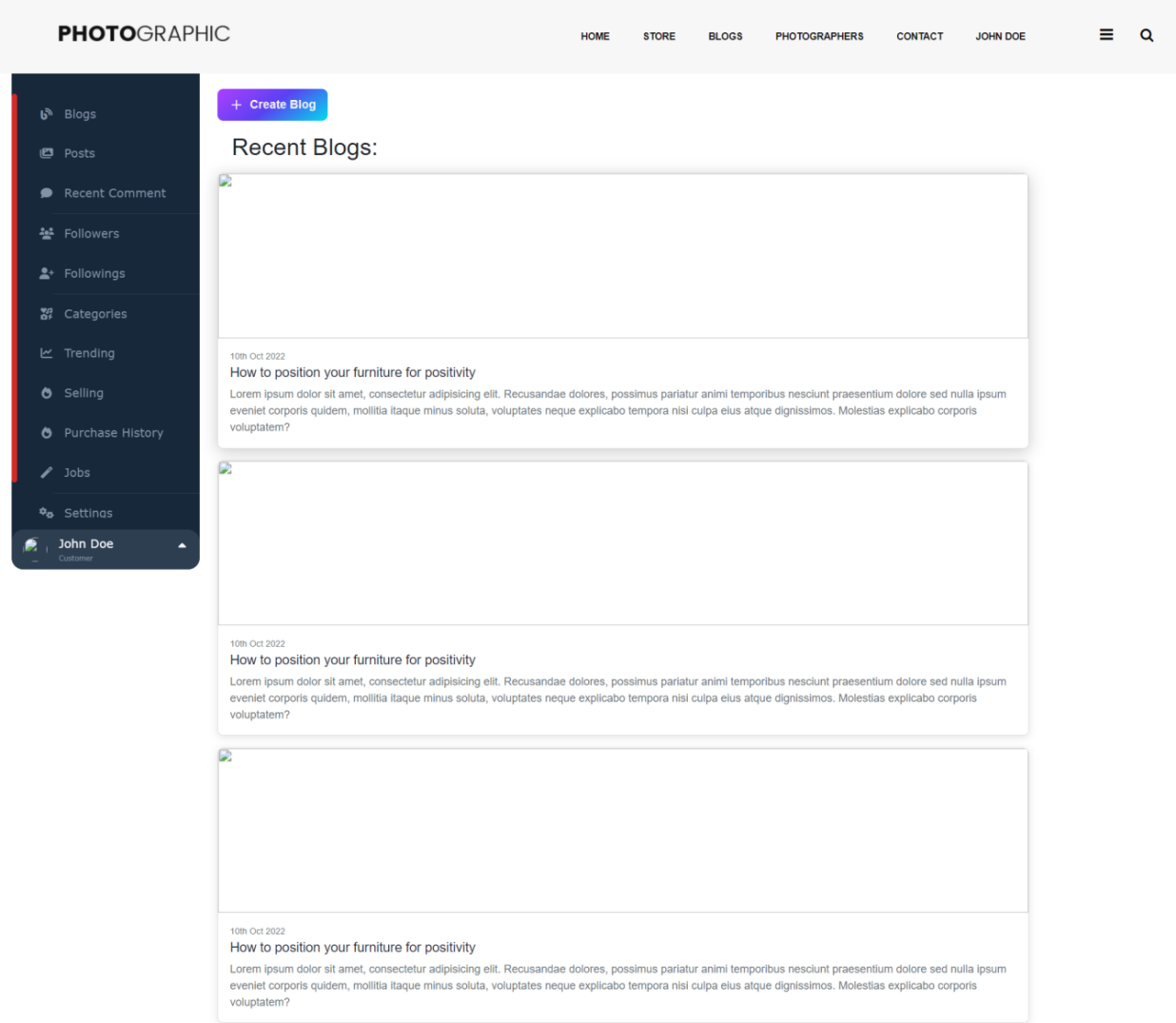
Route về chi tiết của 1 bài blog bao gồm các bài post được public thuộc bài blog đó. Bên trong Route […] là những slug sẽ được custom riêng tùy vào tình huống để đảm bảo việc SEO là hợp lý trong quá trình phát triển.



Hình 17: Blog Detail UI

### User Profile (Tạm Thời) (Route: /user/[userId]/[Route~])

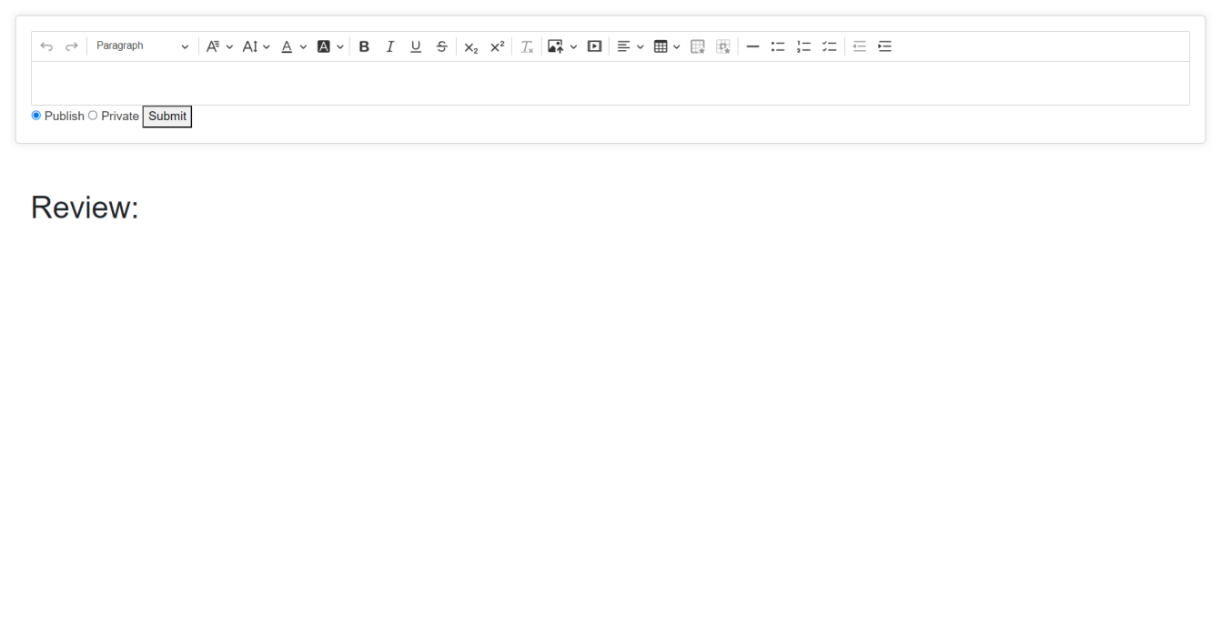
Route này dùng để người dùng quản lý trang cá nhân của bản thân bao gồm: các blogs, các posts, comment,… cùng với đó là các chức năng khác của một user có trong hệ thống được tương tác.



Hình 18: User Profile UI

### Post Create (Tạm Thời) (Route: blog/[…]/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 19: Post Create UI

# TỔNG KẾT

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.

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.