TRƯỜNG ĐẠI HỌC KIẾN TRÚC HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

----- ๑ 🕮 ๑ -----

CSHARP VÀ CÔNG NGHỆ DOTNET

Đề tài kết thúc môn học

Xây dựng mạng xã hội dành cho người viết lách

Giảng viên hướng dẫn: Nguyễn Thị Nguyệt

Nhóm thực hiện - Nhóm 1

Thành viên	Mã sinh viên
Nguyễn Văn Cường (Nhóm trưởng)	2055010026
Nguyễn Thị Diệu My	2055010176
Ngô Nguyễn Ngọc Minh	2055010170
Lê Quang Huy	2055010116
Đường Cảnh Dương	2055010048

Hà Nội, tháng 6 năm 2023

Lời cảm ơn

Trong quá trình học môn C#, chúng em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy cô và bạn bè từ những lab nhỏ cho tới hiện tại là đồ án hết môn. Chúng em coi đây là thành quả cho sự cố gắng của cả nhóm. chúng em xin gửi lời cảm ơn chân thành đến cô - Th.s Nguyễn Thị Nguyệt, giảng viên khoa Công nghệ thông tin đồng thời là giảng viên bộ môn này, người đã tận tình hướng dẫn, chỉ bảo chúng em trong quá trình xây dựng, hoàn thiện dự án.

Em cũng xin chân thành cảm ơn các thầy cô giáo trong trường Đại học Kiến trúc Hà Nội nói chung, các thầy cô trong Khoa Công nghệ thông tin nói riêng đã dạy cho chúng em kiến thức về các môn đại cương cũng như các môn cơ sở, giúp chúng em có được kiến thức nền tảng vững vàng và tạo điều kiện giúp đỡ chúng em trong suốt quá trình học tập.

Cuối cùng, xin chân thành cảm ơn những người bạn, những thành viên trong nhóm đã cùng nhau làm nên dự án này một cách tâm huyết nhất.

- *Trưởng nhóm*

Mục Lục

CHƯƠNG I: MỞ ĐẦU

1.1. Tổng quan

Blog ra đời lần đầu tiên vào năm 1994. Những người dùng blog sử dụng nó để chia sẻ những suy nghĩ cá nhân của bản thân trên internet. Họ viết những câu chuyện của chính họ, cuộc sống hằng ngày, những điều họ muốn chia sẻ với người khác, hầu như bất cứ điều gì họ muốn chia sẻ với người khác, họ bắt đầu một blog đơn giản và viết. Khi blog xuất bản nó online, mọi người tìm thấy blog của họ để học những điều mới và họ cũng giao tiếp với nhau để giải quyết vấn đề của nhau.

Đó là cách mà blog ra đời và bây giờ hầu như bất kỳ ai cũng có blog của riêng mình và sử dụng nó cho những mục đích riêng của họ. Blog giống như một trang web tin tức vậy, mỗi một bài Blog đều sẽ có một cấu trúc riêng biệt tùy vào mục đích của từng cá nhân người viết.Tuy nhiên, nó vẫn phải theo một quy chuẩn riêng nào đó để người đọc có thể nhận ra đó là một bài Blog. Hiện nay, có gần 600 triệu blog có sẵn trên internet, chủ yếu với các nền tảng nổi tiếng như Wordpress, Tumblr, Blogger, Wix, Squarepace.

1.2. Mục đích của đề tài

Việc chọn đề tài Mạng xã hội viết lách (Blog) - mô hình restful API để xây dựng 1 website là nơi sinh hoạt cho người dùng giao lưu, học hỏi hay thậm chí là kinh doanh.

1.3 Đối tượng, phạm vi áp dụng

Đề tài " Mạng xã hội viết lách (blog)" của chúng em được tạo ra hướng tới nhiều đối tượng như các bạn trẻ có sở thích viết lách, muốn chia sẻ cuộc sống thường nhật, hay những công ti với mục đích kinh doanh qua blog.

CHƯƠNG II: CƠ SỞ LÍ THUYẾT

2.1 Net core

NET Core là một nền tảng phát triển đa mục đích, mã nguồn mở được duy trì bởi Microsoft và cộng đồng .NET trên GitHub. Đó là nền tảng chéo (hỗ trợ Windows, macOS và Linux) và có thể được sử dụng để xây dựng các ứng dụng thiết bị, đám mây và IoT. NET core hỗ trợ các ngôn ngữ như C#, F# và 1 phần visual basic Net. Sau đây là một số đặc điểm nổi bật cũng như là tính ưu trội của NET Core:

Đa nền tảng: Chạy trên các hệ điều hành Windows, macOS và Linux.

Nhất quán trên các kiến trúc: có thể chạy mã nguồn của bạn với cùng một hành vi trên nhiều kiến trúc hệ thống, bao gồm x64, x86 và ARM.

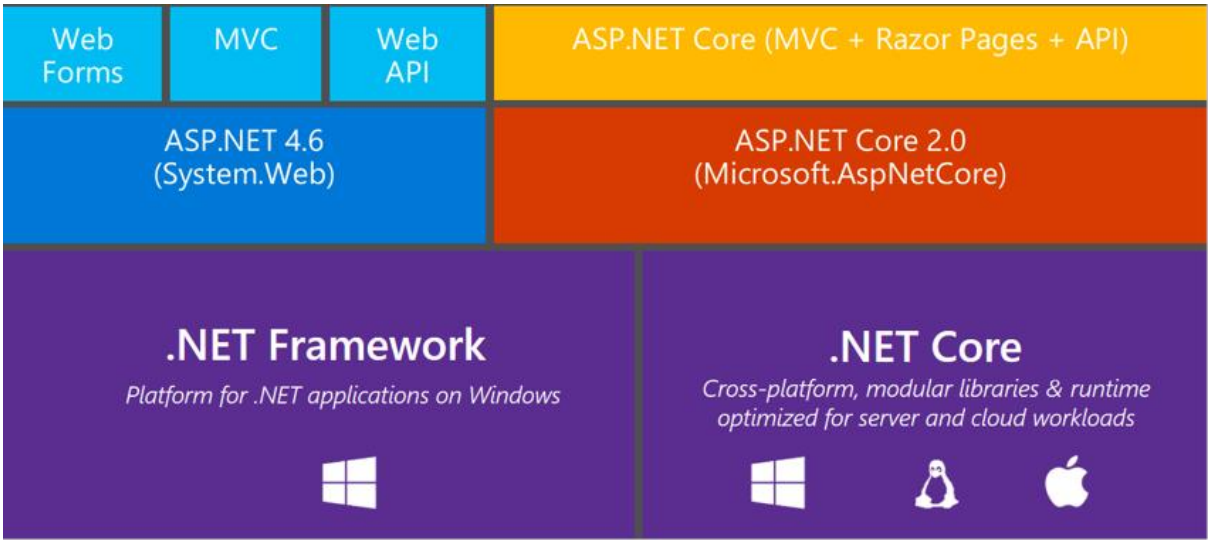
Các công cụ dòng lệnh: Bao gồm các công cụ dòng lệnh dễ sử dụng, có thể được sử dụng để phát triển cục bộ và trong các tình huống tích hợp liên tục.

Triển khai linh hoạt: có thể cài đặt song song (cài đặt toàn người dùng hoặc toàn hệ thống). Có thể được sử dụng với các container Docker

Tương thích: tương thích với .NET Framework, Xamarin và Mono, thông qua .NET Standard.

Nguồn mở: Nền tảng là nguồn mở, sử dụng giấy phép MIT và Apache 2. NET Core là một dự án .NET Foundation.

Được hỗ trợ bởi Microsoft: được Microsoft hỗ trợ.



Thành phần của bao gồm các nền tảng: .NET Compiler Roslyn, .NET Core framework CoreFX, .NET Core runtime CoreCLR, và ASP.NET Core.

- CoreFX: Nó được xem là nền tảng thư viện dành cho NET Core.
- CoreCLR: Đây là công cụ thực thi .Net trong .Net Core. Nó hỗ trợ thực hiện một số chức năng như thu gom và biên dịch rác thành mã máy.
- Net Core runtime: Sẽ cung cấp một kiểu hệ thống, tải lắp ráp, trình thu gom rác và các dịch vụ cơ bản khác.
- Net Core runtime: Cung cấp framework để việc xây dựng các ứng dụng hiện đại tối ưu, dựa trên đám mây, ứng dụng web, kết nối internet,…
- Net Core SDK và trình biên dịch ngôn ngữ (Roslyn và F#): giúp cho phép phát triển Net Core
- Lệnh dotnet: Lệnh dùng cho việc khởi chạy ứng dụng .NET Core và các lệnh CLI.

2.2 Restful API

Định nghĩa: API RESTful là một giao diện mà hai hệ thống máy tính sử dụng để trao đổi thông tin một cách an toàn qua internet. Hầu hết các ứng dụng kinh doanh

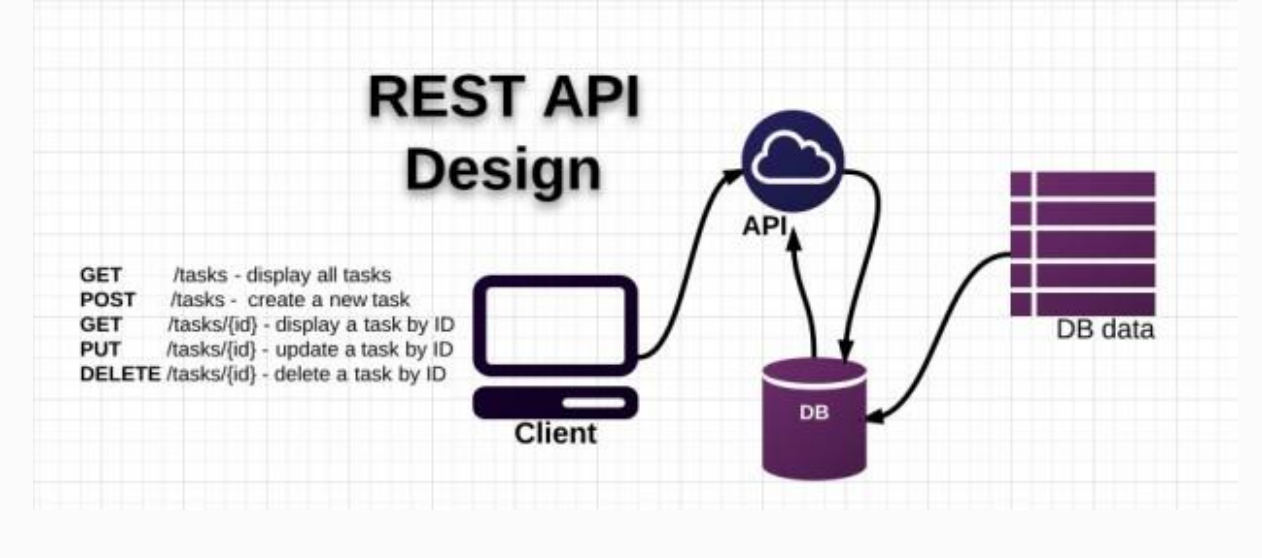
phải giao tiếp với các ứng dụng nội bộ và bên thứ ba khác để thực hiện tác vụ. Ví dụ: để tạo bảng lương hàng tháng, hệ thống báo cáo kế toán nội bộ phải chia sẻ dữ liệu với hệ thống ngân hàng của khách hàng để lập hóa đơn tự động và giao tiếp với ứng dụng bảng chấm công nội bộ. Các API RESTful hỗ trợ trao đổi thông tin này vì chúng tuân theo các tiêu chuẩn giao tiếp phần mềm bảo mật, đáng tin cậy và hiệu quả.

Trong đó, **API** là : Giao diện lập trình ứng dụng (API) xác định các quy tắc mà bạn phải tuân theo để giao tiếp với các hệ thống phần mềm khác. Các nhà phát triển đưa ra hoặc tạo API để các ứng dụng khác có thể giao tiếp với ứng dụng của họ theo cách lập trình. Ví dụ: ứng dụng bảng chấm công đưa ra một API yêu cầu tên đầy đủ của nhân viên và phạm vi ngày. Khi nhận được thông tin này, bảng chấm công của nhân viên sẽ được xử lý nội bộ và trả về số giờ làm việc trong phạm vi ngày đó.

Bạn có thể coi API web như một cổng giữa client và tài nguyên trên web.

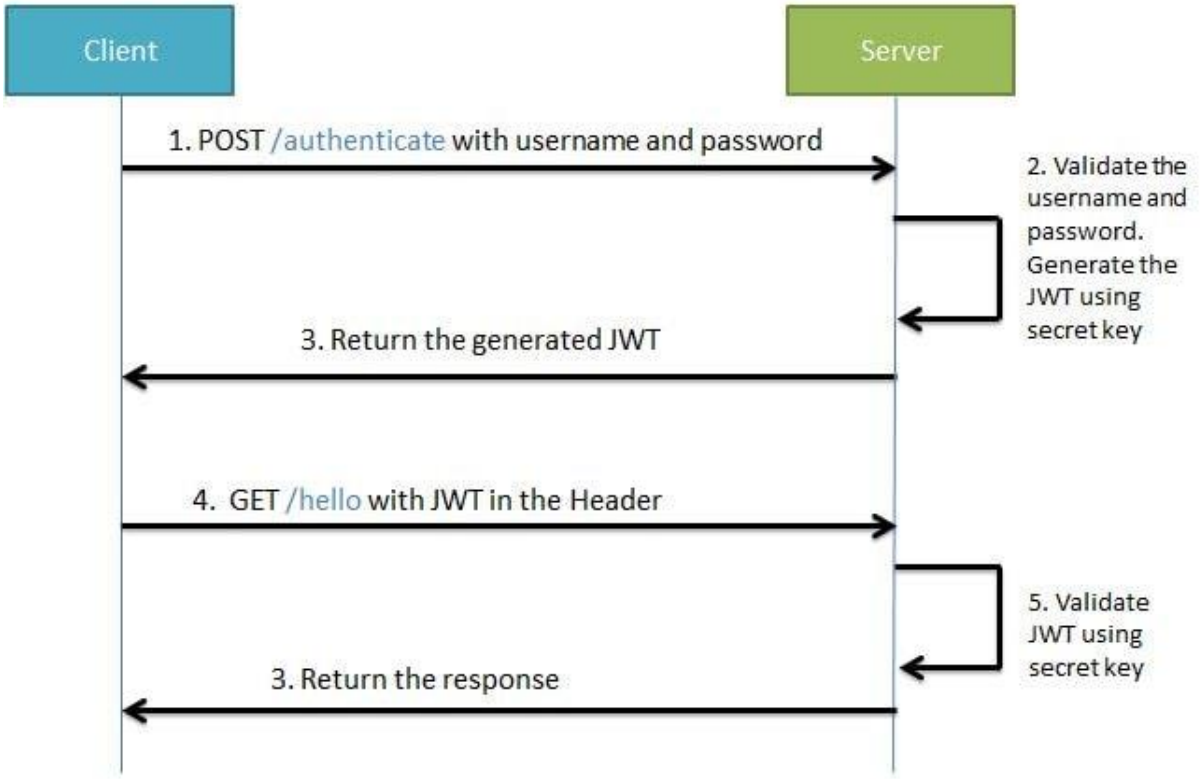
REST là: Chuyển trạng thái đại diện (REST) là một kiến trúc phần mềm quy định các điều kiện về cách thức hoạt động của API. REST ban đầu được tạo ra như một hướng dẫn để quản lý giao tiếp trên một mạng phức tạp như Internet. Bạn có thể sử dụng kiến trúc dựa trên REST để hỗ trợ giao tiếp hiệu suất cao và đáng tin cậy trên quy mô lớn. Bạn có thể dễ dàng triển khai và sửa đổi REST, mang lại khả năng hiển thị và tính di động đa nền tảng cho bất kỳ hệ thống API nào.

Các nhà phát triển API có thể thiết kế các API bằng cách sử dụng nhiều kiến trúc khác nhau. Các API tuân theo kiểu kiến trúc REST được gọi là API REST. Các dịch vụ web triển khai kiến trúc REST được gọi là dịch vụ web RESTful. Thuật ngữ API RESTful thường là chỉ các API web RESTful. Tuy nhiên, bạn có thể sử dụng các thuật ngữ API REST và API RESTful thay thế cho nhau.



2.3 Security - JWT

JSON Web Token (JWT) là một chuẩn mở (RFC 7519) để truyền thông tin an toàn giữa các bên như một đối tượng JSON. Thông tin này được xác minh và đáng tin cậy bởi chữ ký số. JWT được gán với một khoá bí mật (sử dụng thuật toán HMAC) hoặc một cặp khóa công khai/ khóa riêng sử dụng RSA hoặc ECDSA. Các thông tin trong chuỗi JWT được định dạng bằng Json.



Vậy khi nào thì nên sử dụng JWT:

● Xác thực (Authentication): Tình huống thường gặp nhất, khi người dùng đăng nhập, mỗi yêu cầu tiếp theo đều kèm theo chuỗi token JWT, cho phép người dùng có thể truy cập đường dẫn, dịch vụ và tài nguyên được phép ứng với token đó. Đăng nhập một lần cũng là một chức năng có sử dụng JWT một cách rộng rãi, bởi vì chuỗi JWT có kích thước đủ nhỏ để đính kèm trong yêu cầu và sử dụng ở nhiều hệ thống khác nhau.

● Trao đổi thông tin (Information Exchange): JSON Web Token cũng là một cách hữu hiệu và bảo mật để trao đổi thông tin giữa nhiều ứng dụng, bởi vì JWT phải được ký (bằng cặp khóa công khai và khóa riêng). Ngoài ra, chữ ký cũng được tính toán dựa trên nội dung của header và nội dung payload, nhờ đó, bạn có thể xác thực được nội dung là nguyên bản, chưa được chỉnh sửa hoặc can thiệp. Tuy nhiên, một lưu ý hết sức quan trọng là do cấu trúc của JWT đơn giản nên JWT có thể dễ dàng bị giải mã, do vậy, không nên dùng JWT để truyền các thông tin nhạy cảm.

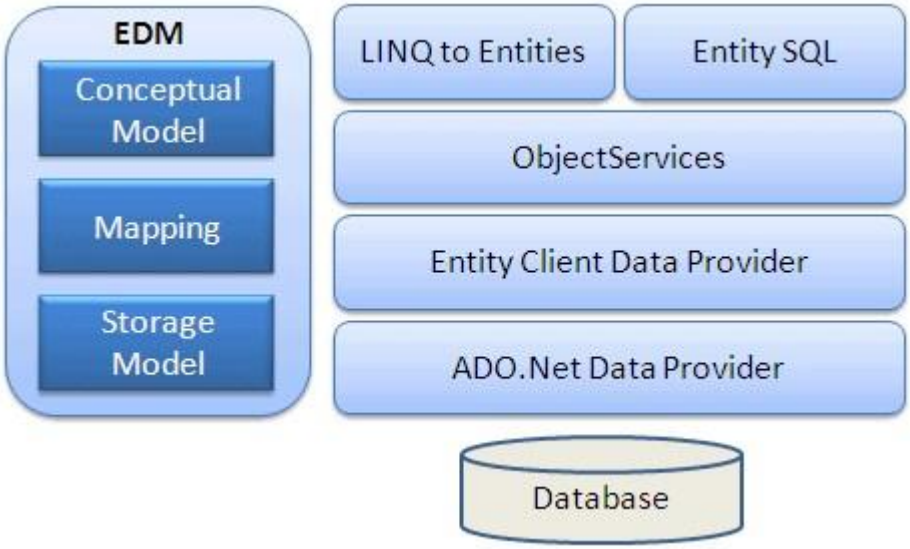
2.4 Entity Framework

Entity Framework ra đời nhằm hỗ trợ sự tương tác giữa các ứng dụng trên nền **tảng .NET** với các cơ sở dữ liệu quan hệ. Hay, Entity Frmework chính là công cụ giúp ánh xạ giữa các đối tượng trong ứng dựng, phần mềm của bạn với các bảng của một cơ sở dữ liệu quan hệ.

Emntity Framework giúp các nhà phát triển Web tương tác với dữ liệu quan hệ theo phương pháp hướng đối tượng với ít mã hơn so với các ứng dụng truyền thống. Lợi ích lớn nhất của nó là giúp lập trình viên giảm thiểu việc lập trình mã nguồn để thực hiện truy cập và tương tác với cơ sở dữ liệu.

Hiện nay, Entity framework là 1 framework mạnh để phát triển ứng dụng Web với sự hỗ trợ đông đảo của cộng đồng người dùng.

Cấu trúc cơ bản Entity Framework được mô tả đơn giản quá hình sau



CHƯƠNG III: PHÂN TÍCH VÀ THIẾT KẾ

3.1. Phân tích yêu cầu

3.1.1. Mô tả nghiệp vụ

Xây dựng website dành cho người thích viết lách. Bất kì ai cũng có thể đăng ký tài khoản và đăng nhập để tương tác lẫn nhau trong website. Hệ thống bao gồm những đối tượng như sau:

- Admin: Người có quyền và trách nhiệm cao nhất của toàn bộ hệ thống.
- Moderator: Người quản trị được admin giao quyền để quản lý các user khác
- User: Người dùng có tài khoản trong hệ thống có thể tương tác với các bài viết, bình luận và theo dõi những người dùng khác.
- Client: Người dùng khách. Đây là những người truy cập website nhưng không có tài khoản. Họ chỉ có quyền đọc một số bài viết nhất định và không có quyền tương tác.

3.1.2. Yêu cầu chức năng

- **Đăng nhập, Đăng ký, Đăng xuất**
  - Đăng nhập: Nếu người dùng đã có tài khoản trên hệ thống, có thể đăng nhập thông qua email và mật khẩu cá nhân.
  - Đăng ký: Nếu chưa có tài khoản, cần tạo tài khoản trên hệ thống bằng cách nhập các thông tin như email, họ, tên, mật khẩu. Sau khi hệ thống nhận được yêu cầu, người dùng sẽ nhận được một email xác nhận. Khi tạo tài khoản, mật khẩu của người dùng được mã hóa một chiều, ngay cả người quản trị hệ thống cũng không thể giải mã được mật khẩu của bất kỳ user nào.
  - Đăng xuất: Khi chọn đăng xuất, người dùng sẽ trở về giao diện khách, chỉ được phép đọc bài viết và không được tương tác.

- **Trang chủ**
  - Người dùng được xem danh sách các bài viết mới nhất, phân trang cho tới bài cũ nhất, các bài viết được quản trị viên ghim lại.
  - Chọn vào một bài viết bất kỳ để xem chi tiết nội dung.
- **Trang bài viết**
  - Xem chi tiết nội dung bài viết, hiển thị số lượt xem của bài, upvote (thích) hoặc downvote (không thích) đối với bài viết.
  - Người dùng cũng có thể bình luận để tương tác với các user khác trong bài viết đó.
  - Truy cập trang cá nhân của tác giả.
  - Nếu là tác giả bài đăng, người dùng có thể chỉnh sửa nội dung bài viết của mình nhưng không được xóa (điều này chỉ được thực hiện bởi người quản trị).
- **Tìm kiếm bài viết**
  - Tại trang chủ của bài viết, người dùng có thể tìm kiếm bài viết dựa trên từ khóa của tiêu đề bài viết, một trang chứa danh sách kết quả những bài viết có kết quả tương tự sẽ hiện ra.
- **Viết bài**
  - Ở bất kì trang nào của website, người dùng cũng có thể truy cập trang trình soạn thảo văn bản để viết bài bằng cách chọn vào nút “Viết bài” trên thanh header của website
  - Tại đây có thể nhập tiêu đề bài viết, nội dung bài viết có thể kèm theo một hình ảnh hoặc không, trang web sẽ lấy một bức ảnh mặc định từ hệ thống để làm ảnh thumbnail (ảnh hiển thị).
  - Sau khi hoàn tất việc soạn thảo, bước tiếp theo người dùng được phép chọn một hoặc nhiều thể loại (categories) phù hợp cho nội dung bài viết

của mình kèm theo một mô tả ngắn (hoặc không, trang web sẽ tự động lấy một phần nhỏ nội dung bài viêt làm mô tả ngắn).

- Cuối cùng là chọn nút đăng bài, họ sẽ được chuyển tới trang của bài viết đó.

- **Bình luận về bài viết**
  - Khi truy cập vào một bài viết bất kỳ, user có thể để lại bình luận tại bài viết đó.
  - User chỉ có quyền chỉnh sửa bình luận và không có quyền xóa.
- **Chỉnh sửa thông tin cá nhân**
  - Những thông tin mà người dùng có thể chỉnh sửa cùng lúc như mô tả về bản thân, họ và tên, ngày sinh và giới tính.
  - Nếu muốn đổi mật khẩu, người dùng phải nhập mật khẩu hiện tại kèm theo mật khẩu mới và chọn lưu.
  - Với email, sau khi nhập một email thay đổi mới, họ cần xác thực lại địa chỉ tương tự như khi đăng ký tài khoản.
- **Truy cập trang cá nhân**
  - Tại đây sẽ hiển thị danh sách những bài viết của người dùng đó kèm theo ảnh đại diện, ảnh bìa và mô tả tiểu sử.
  - Đối với trang cá nhân của user khác, chúng ta có thể lựa chọn theo dõi (nếu chưa theo dõi) và bỏ theo dõi (nếu đang theo dõi) họ.
  - Đối với trang cá nhân của bản thân. Người dùng có thể cập nhật ảnh đại diện và ảnh bìa, chỉnh sửa tiểu sử (mô tả bản thân).
- **Trang quản trị**
  - Chỉ những người quản trị như Admin và Moderator mới có thể truy cập.

- Tại đây hiển thị danh sách các bài viết, thể loại bài viết và danh sách người dùng tương ứng với các thông tin như tên tiêu đề, tác giả, thể loại của bài viết, tên của thể loại bài viết và tên người dùng, email, vai trò.
- Bên cạnh đấy là các nút kèm theo các hành động như xóa hay ghim bài viết, chỉnh sửa hoặc xóa một thể loại bài viết, gán hoặc gỡ quyền đối với một thành viên quản trị (điều này chỉ được thực hiển bởi Admin – người có vai trò lớn nhất trong hệ thống).

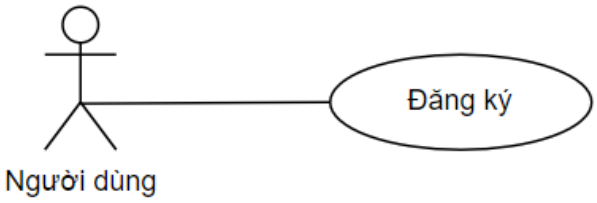
3.2. Phân tích hệ thống

3.2.1. Biểu đồ Use Case toàn hệ thống

Hình 1: Biểu đồ Use Case tổng quát

3.2.2. Chức năng đăng ký tài khoản

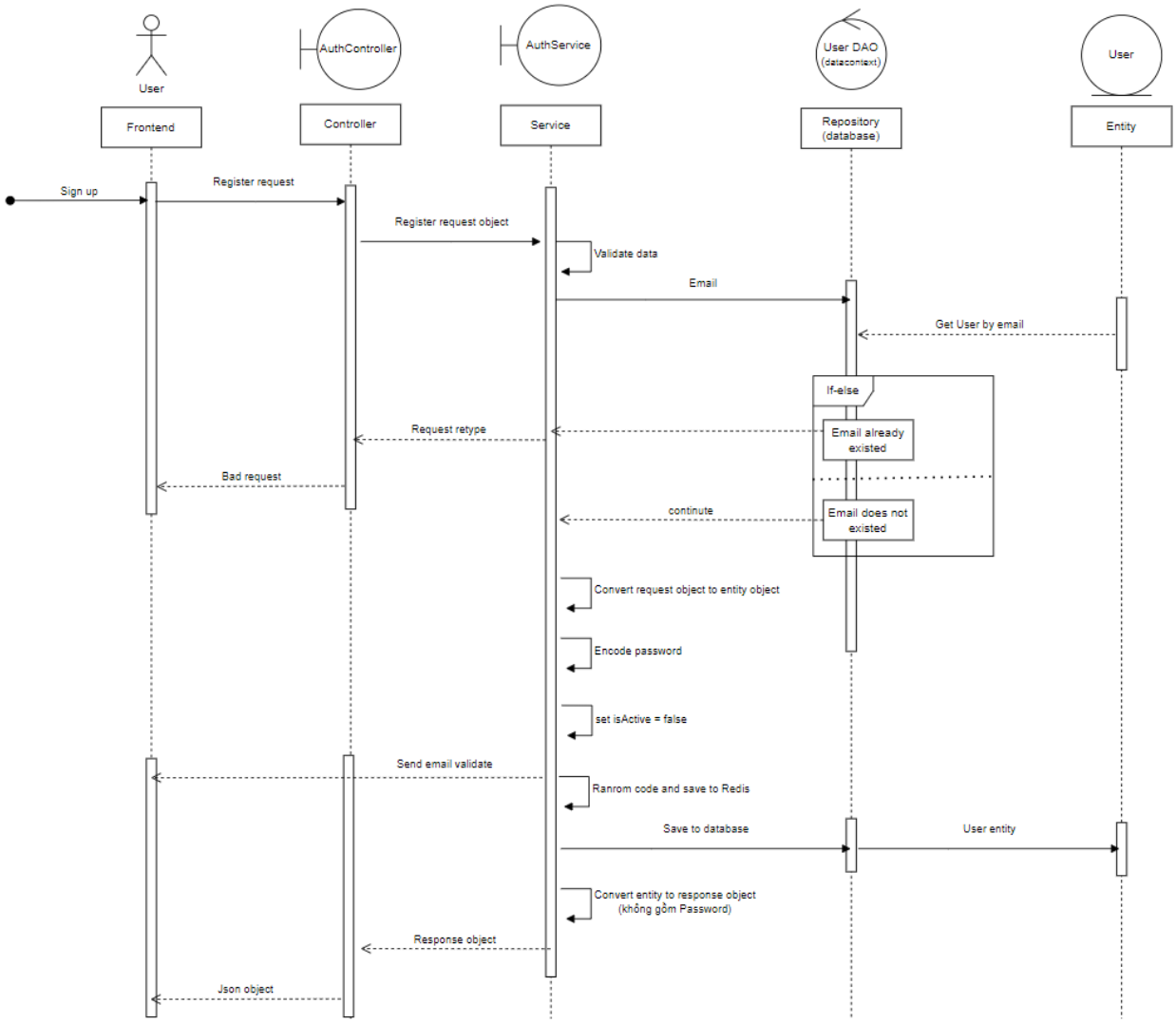
3.2.2.1. Biểu đồ Use Case đăng ký



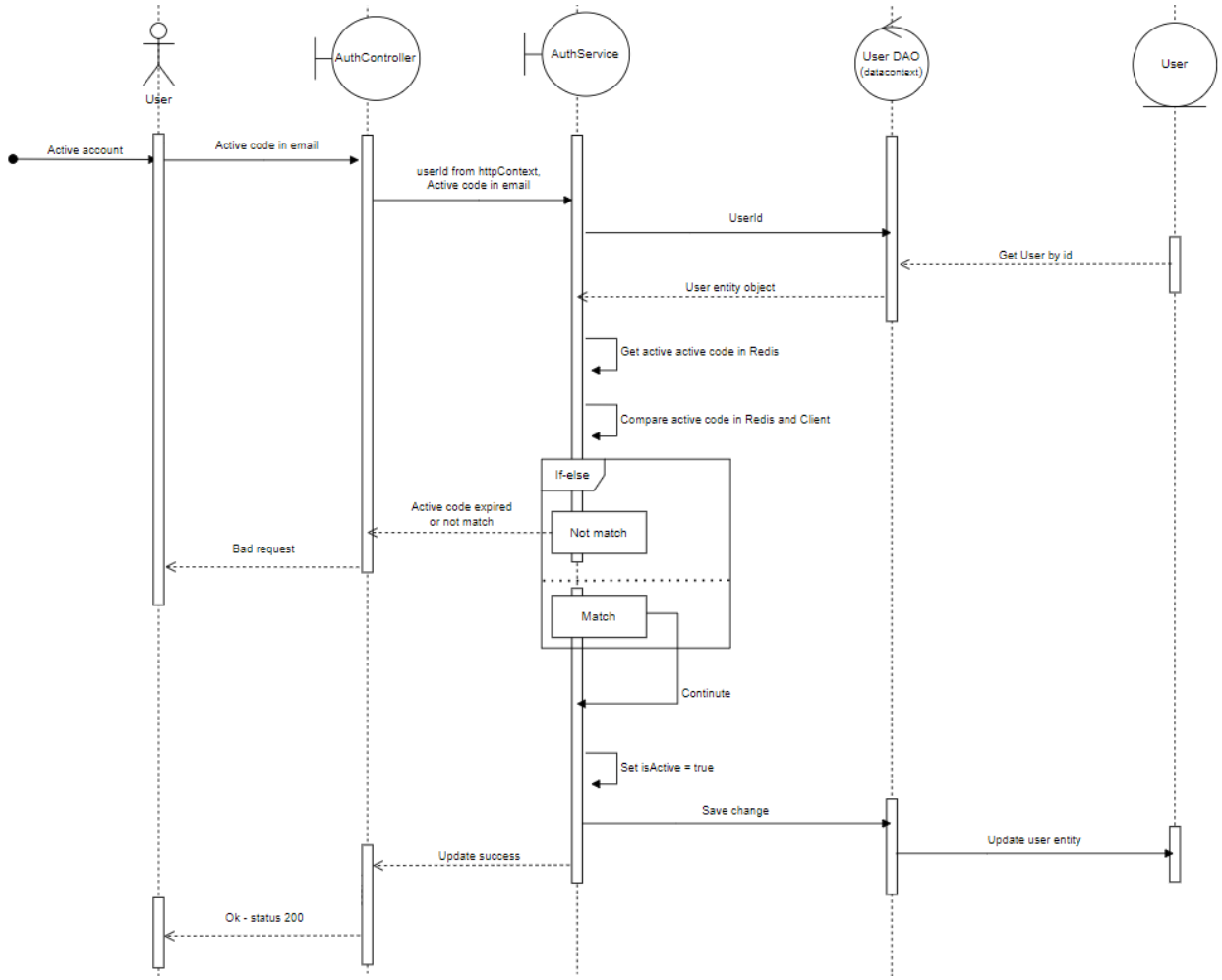
Hình 2: Use Case đăng ký

Tên Use Case	**Đăng ký tài khoản**
Tác nhân	Người dùng ngoài hệ thống
Mục đích	Xác định danh tính người dùng
Luồng tương tác	1. User chọn đăng ký 2. Nhập các thông tin 3. Hệ thống nhận, kiểm tra thông tin và phản hồi 4. Quay lại trang đăng nhập
Luồng tương tác ngoại lệ	3.1. Email bị trùng hoặc thiếu dữ liệu 3.2. Hệ thống yêu cầu người dùng nhập lại 3.3. Hệ thống tiếp tục kiểm tra thông tin và phản hồi lại

3.2.2.2. Biểu đồ tuần tự đăng ký (sequence)



Hình 3.1: Biểu đồ sequence chức năng đăng ký (Bước tạo tài khoản)

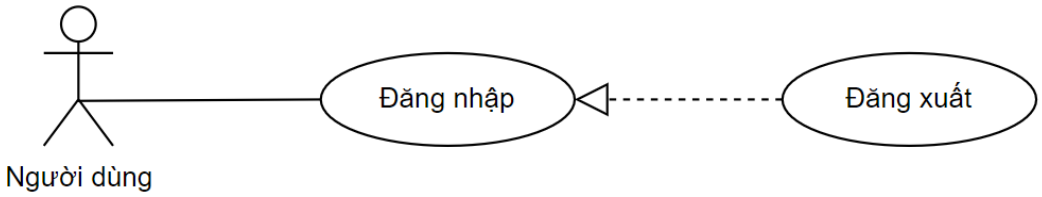


Hình 3.2: : Biểu đồ sequence chức năng đăng ký (Bước kính hoạt tài khoản)

3.2.3. Chức năng đăng nhập

3.2.3.1. Biểu đồ Use Case đăng nhập

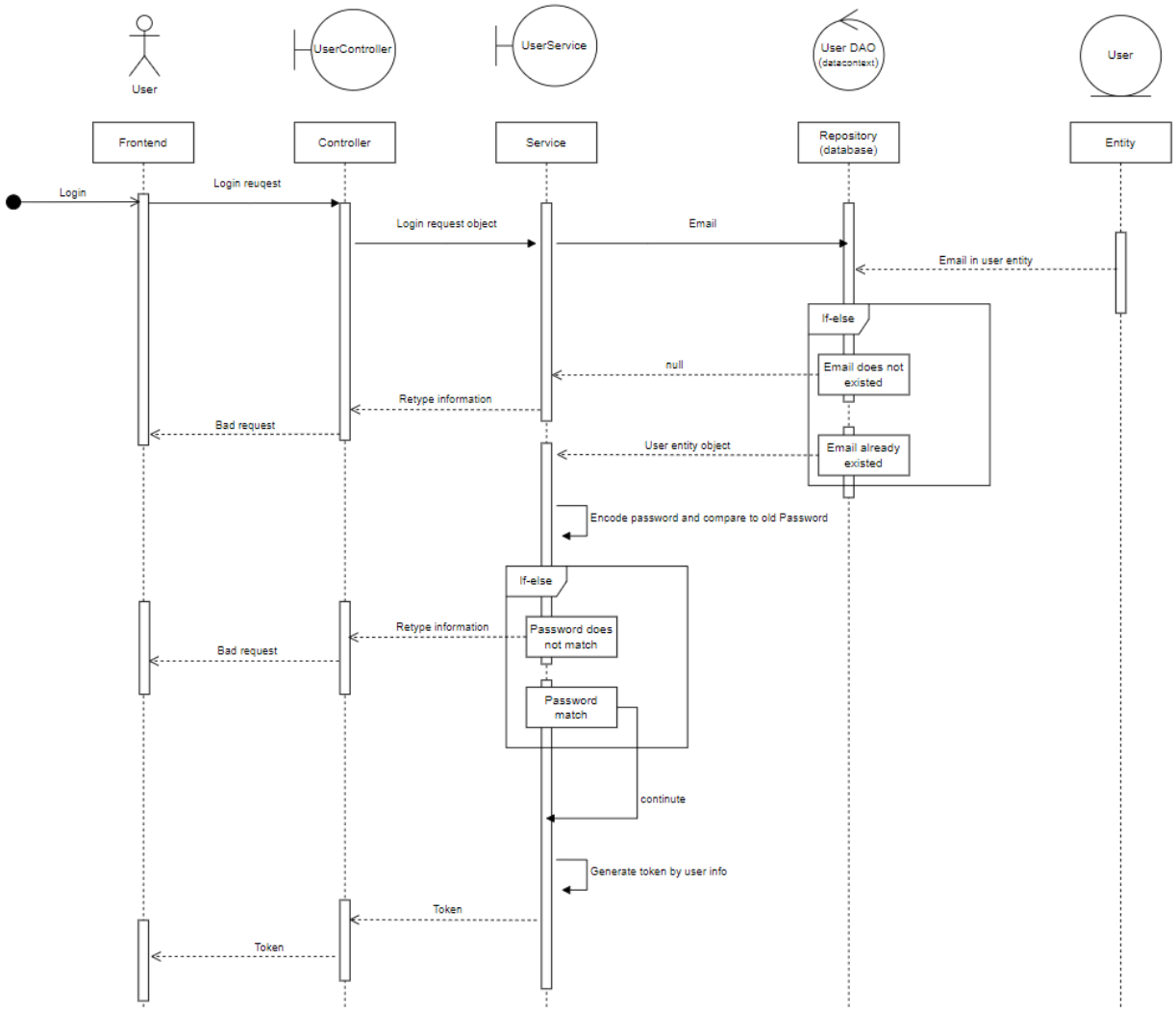
Use Case này cho phép User thực thiện chức năng đăng nhập vào hệ thống hoặc đăng xuất.



Hình 4: Use Case đăng nhập, đăng xuất

Tên Use Case	**Đăng nhập**
Tác nhân	Người dùng trong hệ thống
Mục đích	Xác thực người dùng
Luồng tương tác	1. Người dùng chọn đăng nhập 2. Người dùng nhập email và mật khẩu 3. Hệ thống nhận, kiểm tra thông tin và phản hồi trạng thái 4. Chuyển tới trang chủ
Luồng tương tác ngoại lệ	3.1. Sai một trong hai thông tin 3.2. Hệ thống yêu cầu người dùng nhập lại

3.2.3.2. Biểu đồ tuần tự đăng nhập (sequence)

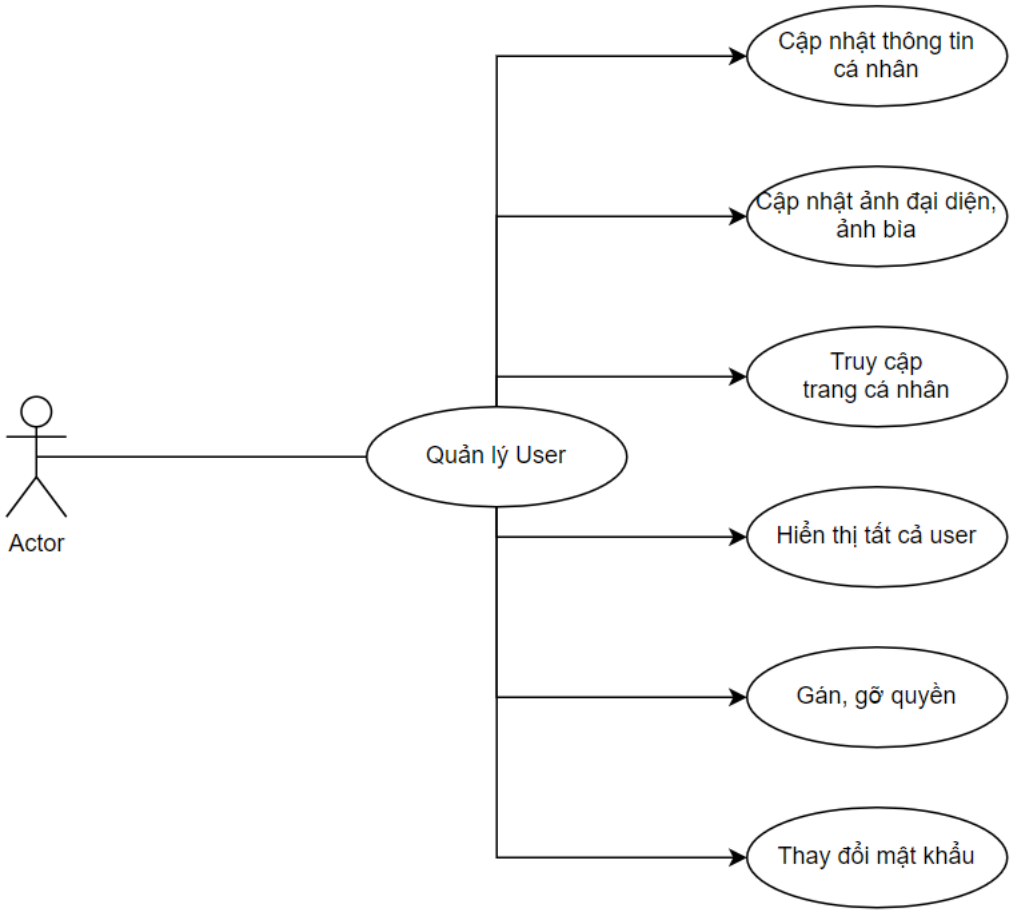


Hình 5: Biểu đồ sequence chức năng đăng nhập

3.2.4. Chức năng liên quan đến User

3.2.4.1. Biểu đồ Use Case đối với User

Use Case này cho phép User thực hiện các chức năng như cập nhật thông tin cá nhân, cập nhật ảnh bìa và ảnh đại diện, truy cập trang cá nhân, thay đổi mật khẩu. Đối với Admin và Moderator có thể hiển thị danh sách người dùng và gán, gỡ quyền đối với Admin.



Hình 6: Các Use Case đối với User

Tên Use Case	**Cập nhật thông tin**
Tác nhân	Người dùng trong hệ thống

Mục đích	Thay đổi thông tin cá nhân
Luồng tương tác	1. Người dùng truy cập trang thông tin 2. Người dùng điền các trường dữ liệu có thể thay thế 3. Chọn cập nhật 4. Hệ thống xác nhận cập nhật thành công
Luồng tương tác ngoại lệ	3.1. Thông tin không phù hợp, hệ thống không cập nhật và yêu cầu nhập lại 3.2. Tiếp tục xác nhận lại thông tin sau khi nhập

Tên Use Case	**Cập nhật ảnh đại diện, ảnh bìa**
Tác nhân	Người dùng trong hệ thống
Mục đích	Thay đổi ảnh đại diện, ảnh bìa thiển thị trang cá nhân
Luồng tương tác	1. Người dùng truy cập trang cá nhân 2. Chọn nút chỉnh sửa ảnh 3. Chọn một bức ảnh trong máy 4. Trang web tự động cập nhật lại
Luồng tương tác ngoại lệ	3.1. Ảnh không đúng định dạng 3.2. Hệ thống từ chối cập nhật 3.2. Người dùng chọn lại ảnh mới

Tên Use Case	**Hiển thị trang cá nhân**
Tác nhân	Người dùng trong hệ thống

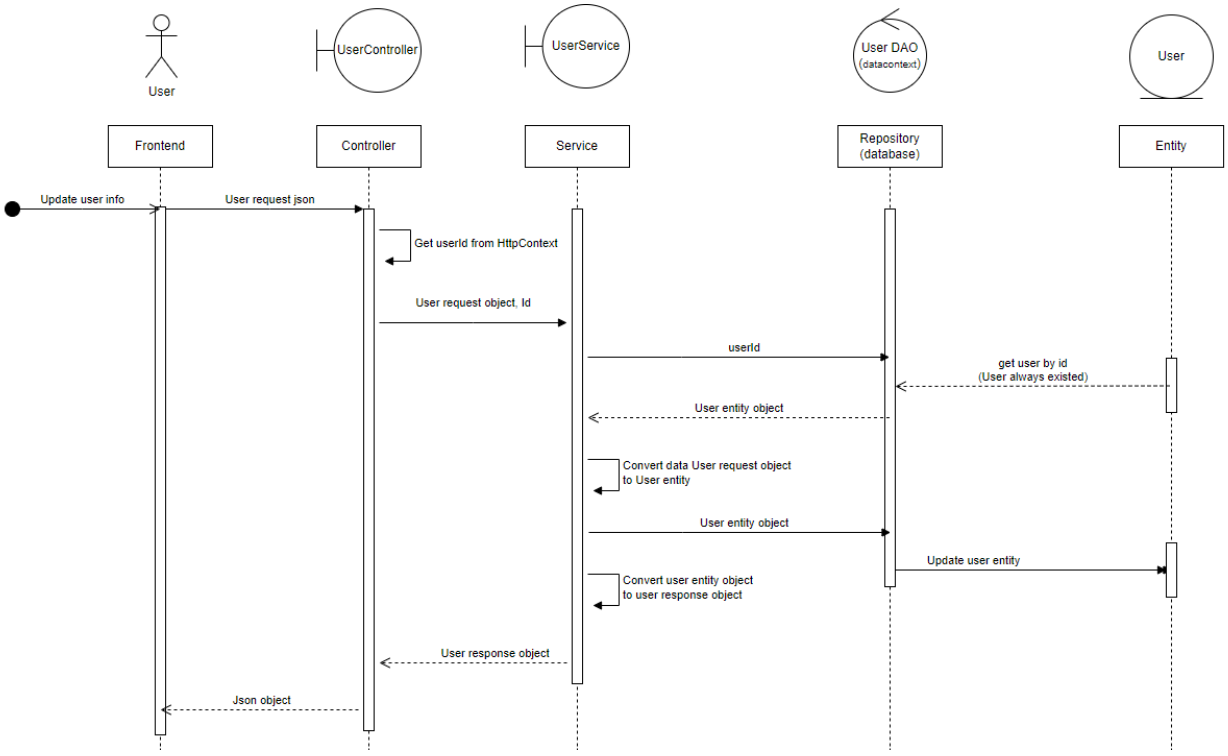
Mục đích	Hiển thị bài đăng, thông người dùng
Luồng tương tác	1. Người dùng trong hệ thống chọn truy cập trang cá nhân của bất kỳ ai 2. Hệ thống đưa tới trang người dùng đó và hiển thị thông tin
Luồng tương tác ngoại lệ	1.1. Người dùng truy cập trang cá nhân không tồn tại từ URL 1.2. Trang web không trả ra gì cả :D

Tên Use Case	**Hiển thị tất cả Users**
Tác nhân	Người quản trị, admin
Mục đích	Hiển thị, gán quyền với các users
Luồng tương tác	1. Trên thanh header trang web, chọn ảnh nhỏ cá nhân. 2. Người quản trị chọn truy cập trang admin. 3. Hệ thống trả ra danh sách bảng các users đang hoạt động.
Luồng tương tác ngoại lệ	2.1. User cố gắng truy cập thông qua URL 2.2. Hệ thống từ chối yêu cầu, quay lại trang chủ

Tên Use Case	**Gán quyền, gỡ quyền**
Tác nhân	Admin

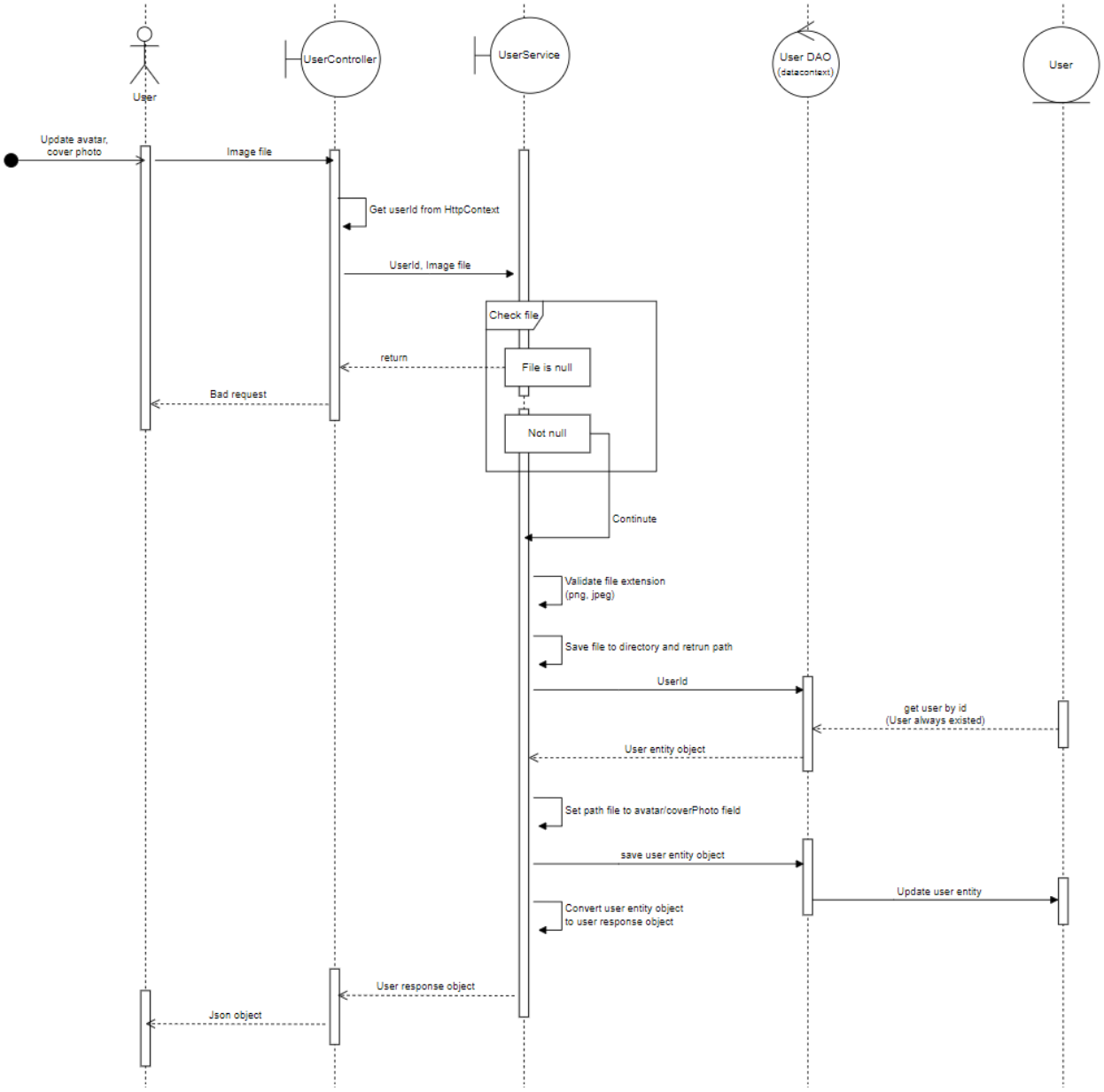
Mục đích	Giao vai trò quản trị cho user khác
Luồng tương tác	1. Admin trỏ vào ảnh cá nhân trên header 2. Chọn truy cập trang admin 3. Hệ thống trả ra danh sách các users 4. Admin chọn gán/gỡ quyền đối với một user bất kỳ
Luồng tương tác ngoại lệ	4.1. Vai trò moderator cố gắng thực hiện thao tác 4.2. Bị từ chối yêu cầu

1.2.4.2. Biểu đồ tuần tự cập nhật thông tin cá nhân



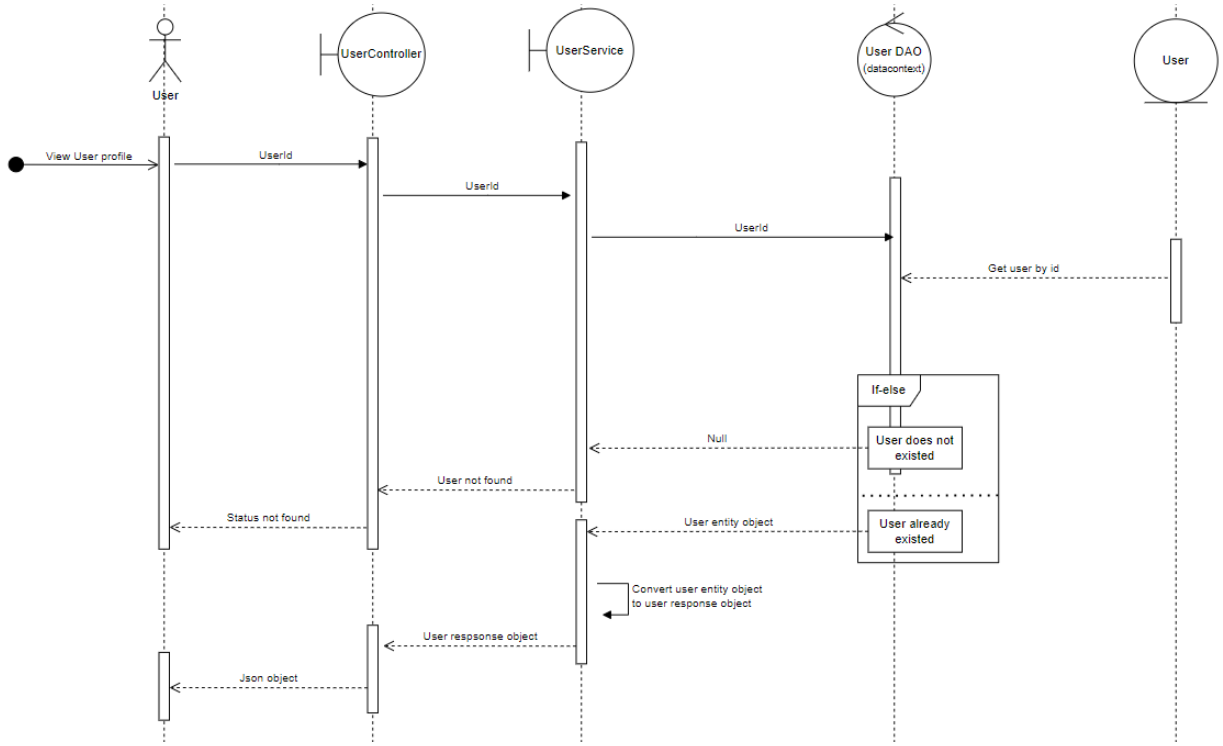
Hình 7: Sequence cập nhật thông tin cá nhân

1.2.4.3. Biểu đồ tuần tự cập nhật ảnh đại diện, ảnh bìa



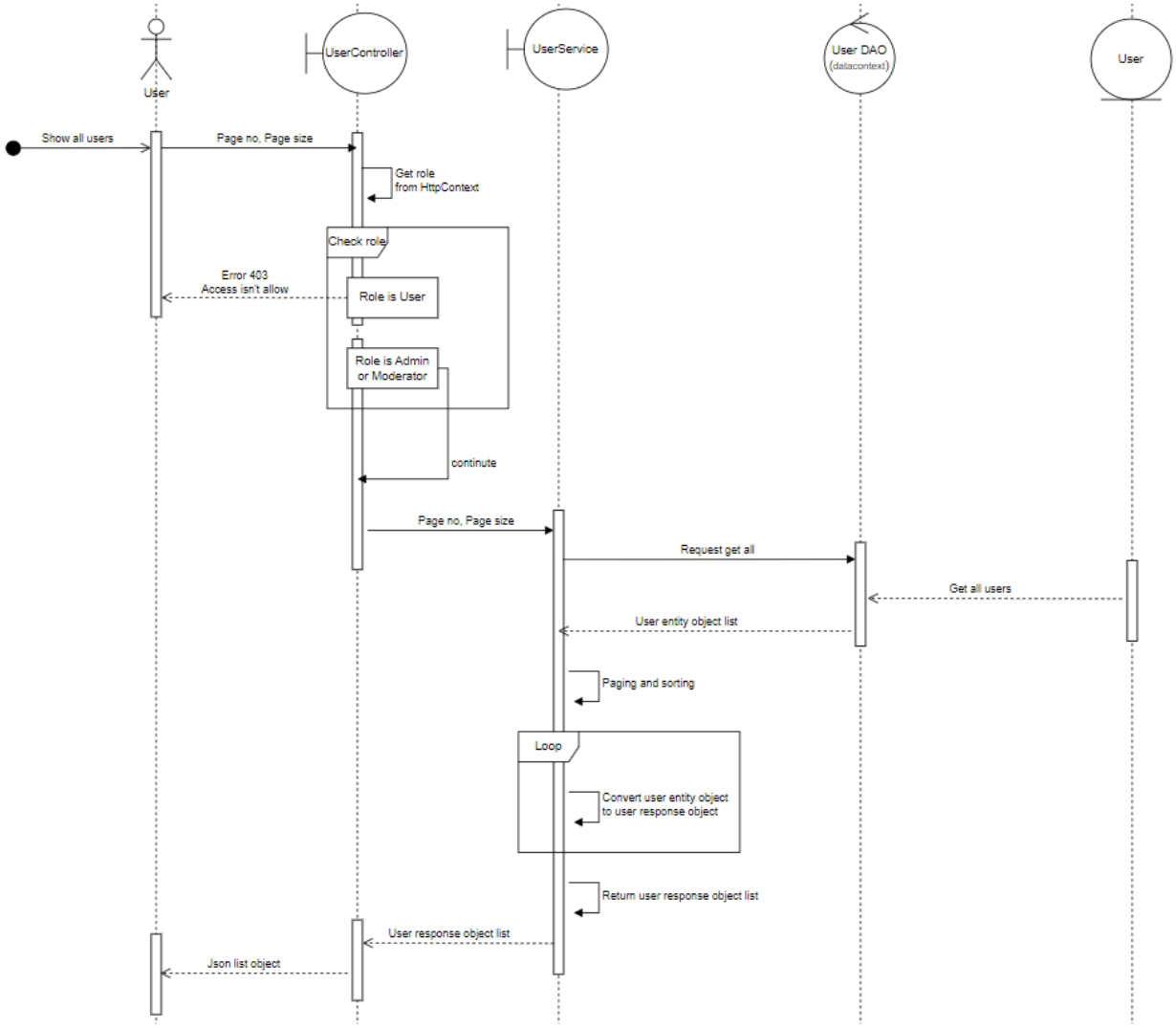
Hình 8: Sequence cập nhật ảnh đại diện, ảnh bìa

1.2.4.4. Biểu đồ tuần tự hiển thị trang cá nhân



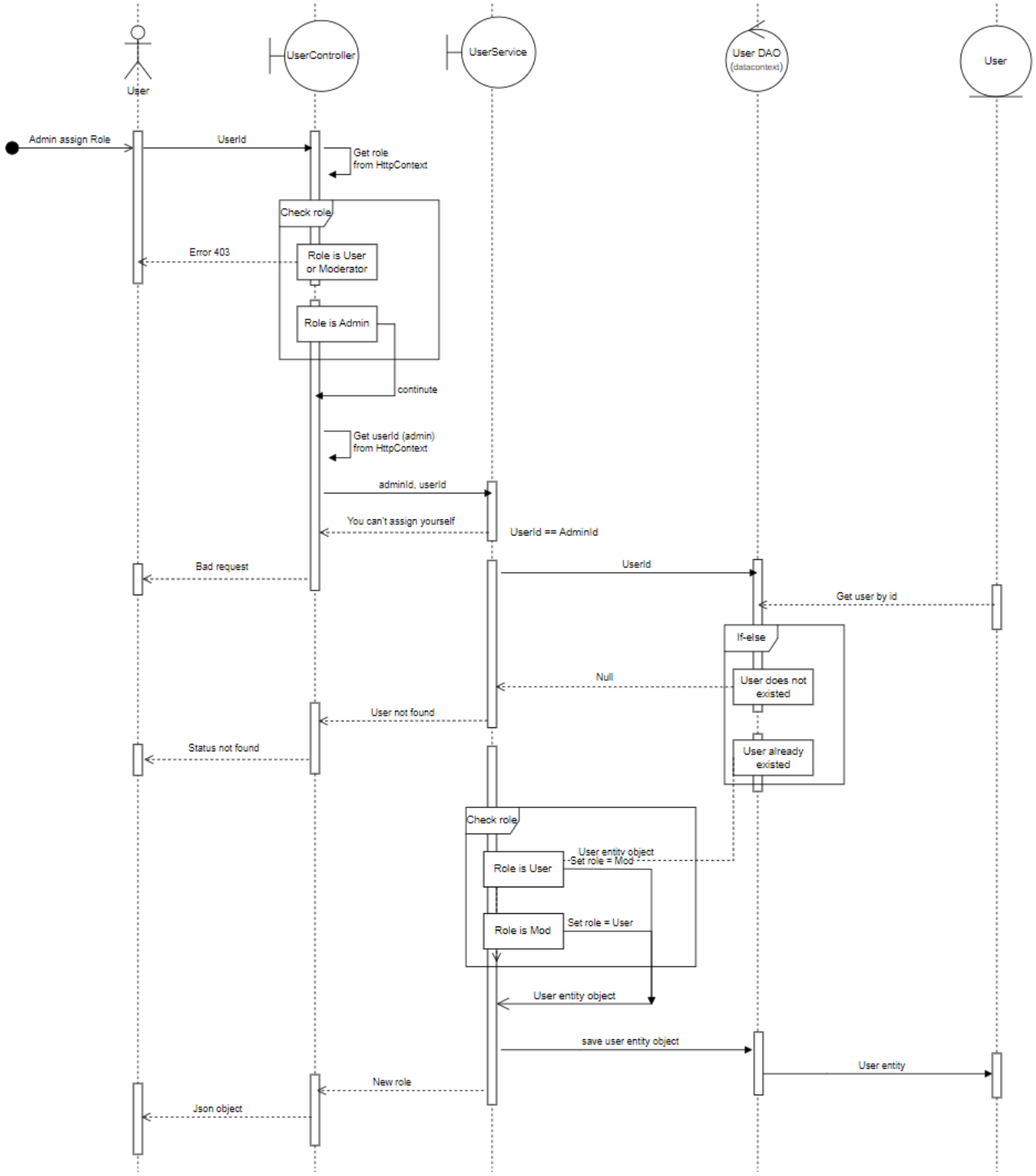
Hình 9: Sequence hiển thị trang cá nhân

1.2.4.5. Biểu đồ tuần tự hiển thị tất cả Users



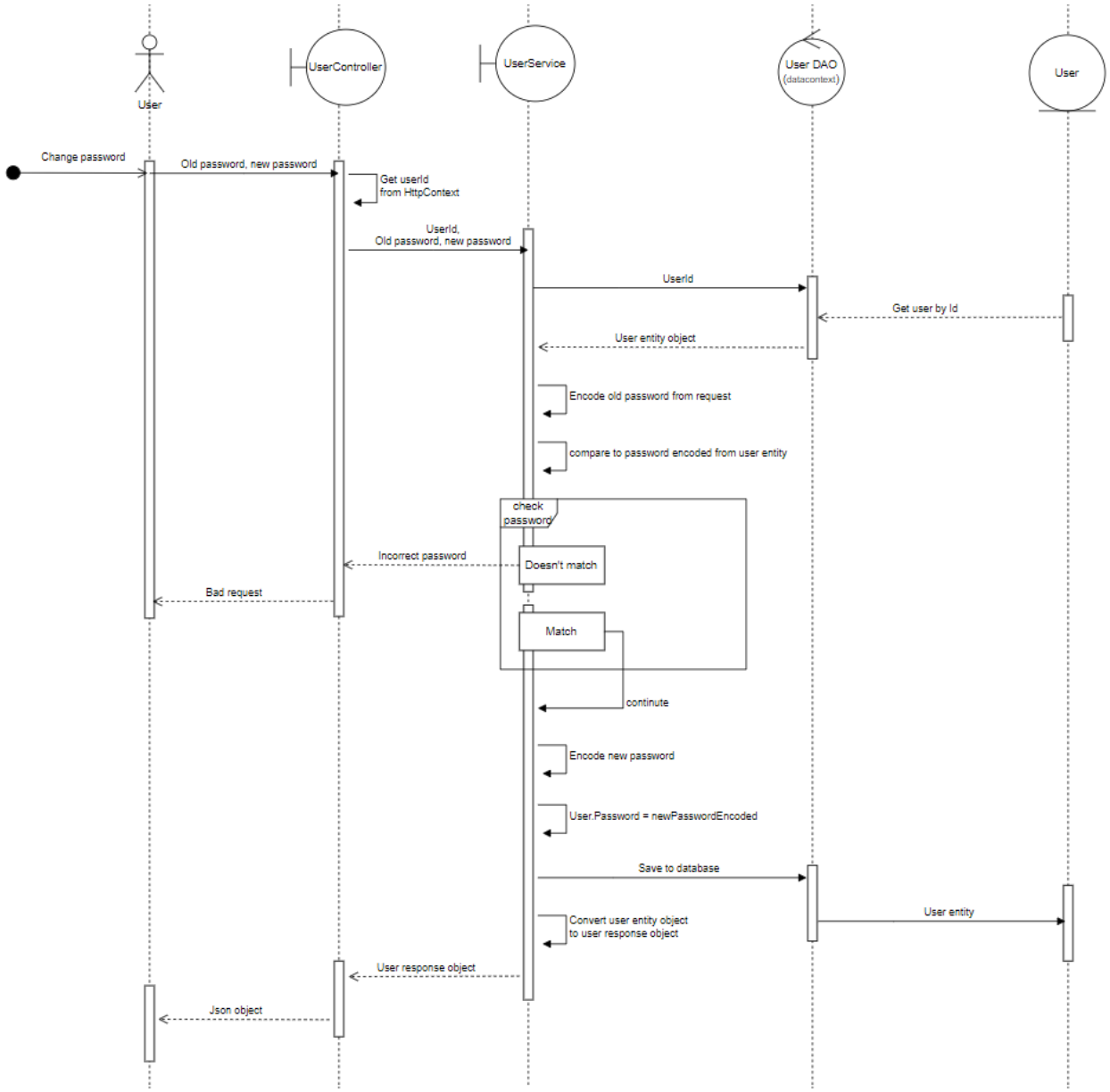
Hình 10: Sequence hiển thị tất cả Users

1.2.4.6. Biểu đồ tuần tự gán quyền cho user



Hình 11: Sequence gán quyền cho quản trị cho User

3.2.4.7 Biểu đồ tuần tự thay đổi mật khẩu

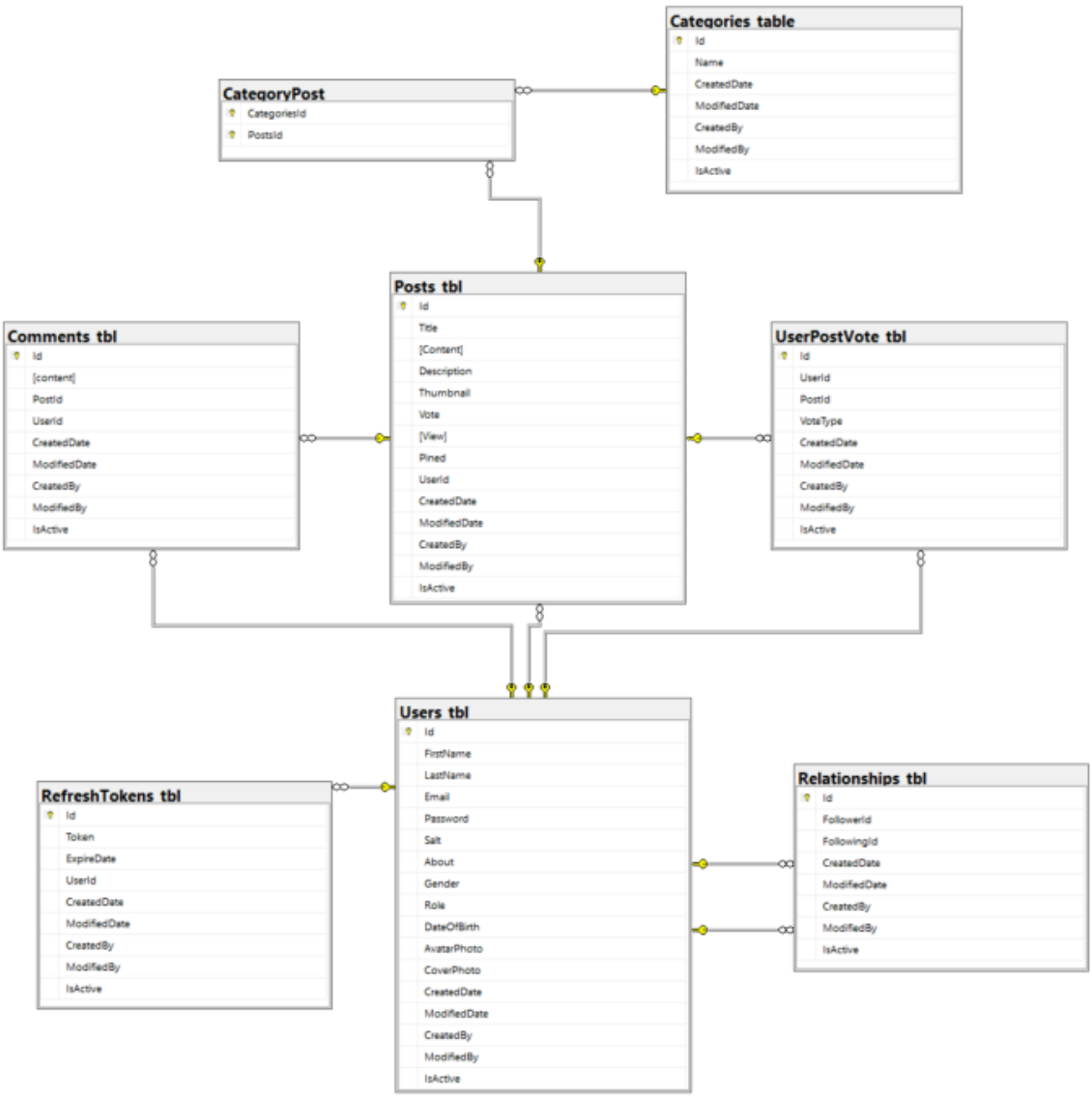


Hình 12: Sequence thay đổi mật khẩu

1.3. Thiết kế hệ thống

4.3.1. Thiết kế cơ sở dữ liệu

4.3.1.1. Tổng quan mô hình thực thể



Hình 13: Sơ đồ mô hình liên kết cơ sở dữ liệu

4.3.1.2. Bảng User

Bảng User dùng để lưu trữ thông tin người dùng

Tên trường	**Kiểu dữ liệu**	**Mô tả**
id	int	Khóa chính
first_name	nvarchar(255)	Tên
last_name	nvarchar(255)	Họ
email	nvarchar(255)	Tên đăng nhập
password	nvarchar(max)	Mật khẩu
salt	nvarchar(255)	Tham gia mã hóa mật khẩu
about	nvarchar(max)	Mô tả về người dùng
gender	varchar (enum)	Giới tính
role	varchar (enum)	Vai trò
data_of_birth	datetime	Ngày sinh
avatar_photo	varchar(max)	Đường dẫn ảnh đại diện
cover_photo	varchar(max)	Đường dẫn ảnh bìa
created_date	datetime	Thời gian tạo tài khoản
modified_date	datetime	Lần chỉnh sửa cuối
created_by	nvarchar(255)	Tạo bởi
modified_by	nvarchar(255)	Chỉnh sửa bởi
is_active	bit	Trạng thái của đối tượng

4.3.1.3. Bảng Relationship

Bảng Relationship dùng để lưu trữ ràng buộc trạng thái theo dõi (hoặc không) của các user với nhau

Tên trường	**Kiểu dữ liệu**	**Mô tả**
id	int	Khóa chính
follower_id	int	Khóa ngoại
following_id	int	Khóa ngoại
created_date	datetime	Thời gian tạo
modified_date	datetime	Lần chỉnh sửa cuối
created_by	nvarchar(255)	Tạo bởi
modified_by	nvarchar(255)	Chỉnh sửa bởi
is_active	bit	Trạng thái của đối tượng

4.3.1.4. Bảng RefreshToken

Bảng RefreshToken dùng để lưu trữ token dài hạn cho phía người dùng. Khi một accessToken kết hết, một token mới sẽ được sinh ra bằng đoạn mã refreshToken này.

Tên trường	**Kiểu dữ liệu**	**Mô tả**
id	int	Khóa chính
token	nvarchar(max)	refresh token
expire_date	datetime	Thời điểm hết hạn
user_id	int	Khóa ngoại
created_date	datetime	Thời gian tạo
modified_date	datetime	Lần chỉnh sửa cuối

created_by	nvarchar(255)	Tạo bởi
modified_by	nvarchar(255)	Chỉnh sửa bởi
is_active	bit	Trạng thái của đối tượng

3.3.1.5. Bảng Post

Bảng Post chứa thông tin về mỗi bài viết của trang web

Tên trường	**Kiểu dữ liệu**	**Mô tả**
id	int	Khóa chính
title	nvarchar(max)	Tiêu đề bài viết
content	nvarchar(max)	Nội dung bài viết
description	nvarchar(max)	Mô tả bài viết
thumbnail	nvarchar(max)	Đường dẫn ảnh bài viết
vote	int	Tổng số lượt thích
view	int	Tổng số lượt xem
pined	bit	Trạng thái ghim bài viết
user_id	int	Khóa ngoại
created_date	datetime	Thời gian tạo
modified_date	datetime	Lần chỉnh sửa cuối
created_by	nvarchar(255)	Tạo bởi
modified_by	nvarchar(255)	Chỉnh sửa bởi
is_active	bit	Trạng thái của đối tượng

3.3.1.6. Bảng UserPostVote

Bảng UserPostVote lưu lại các hành động upvote (thích) hoặc downvote (không thích) của các users đối với các bài viết.

Tên trường	**Kiểu dữ liệu**	**Mô tả**
id	int	Khóa chính
user_id	int	Khóa ngoại
post_id	int	Khóa ngoại
vote_type	nvarchar (enum)	Hành động up hoặc down
created_date	datetime	Thời gian tạo
modified_date	datetime	Lần chỉnh sửa cuối
created_by	nvarchar(255)	Tạo bởi
modified_by	nvarchar(255)	Chỉnh sửa bởi
is_active	bit	Trạng thái của đối tượng

3.3.1.7. Bảng Comment

Bảng Comment chứa các thông tin về bình luận của mỗi user đối với mỗi bài viết khác nhau.

Tên trường	**Kiểu dữ liệu**	**Mô tả**
id	int	Khóa chính
user_id	int	Khóa ngoại
post_id	int	Khóa ngoại
content	nvarchar (max)	Nội dung bình luận
created_date	datetime	Thời gian tạo
modified_date	datetime	Lần chỉnh sửa cuối
created_by	nvarchar(255)	Tạo bởi
modified_by	nvarchar(255)	Chỉnh sửa bởi
is_active	bit	Trạng thái của đối tượng

3.3.1.8. Bảng Category

Bảng category chứa thông tin về thể loại của bài viết

Tên trường	**Kiểu dữ liệu**	**Mô tả**
id	int	Khóa chính
name	nvachar(255)	Tên thể loại
created_date	datetime	Thời gian tạo
modified_date	datetime	Lần chỉnh sửa cuối
created_by	nvarchar(255)	Tạo bởi
modified_by	nvarchar(255)	Chỉnh sửa bởi
is_active	bit	Trạng thái của đối tượng

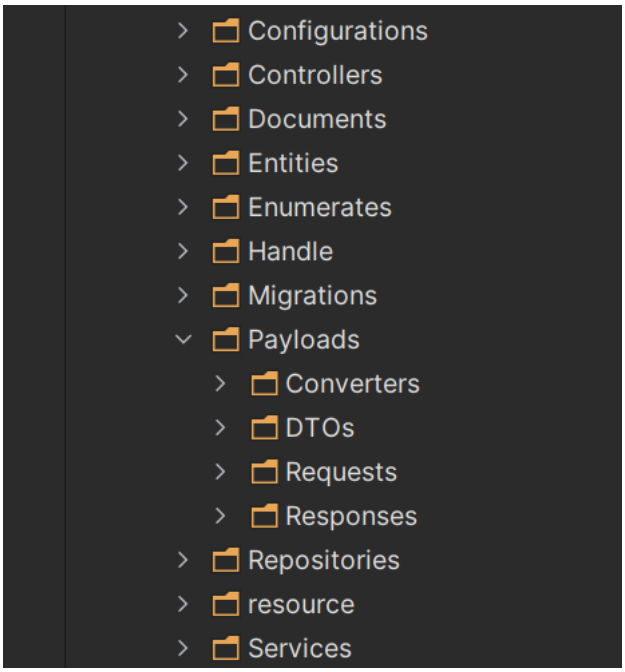
3.3.1.9. Bảng PostCategory

Bảng PostCategory chứa thông tin liên kêt giữa hai bảng Post và Category

Tên trường	**Kiểu dữ liệu**	**Mô tả**
category_id	int	Khóa ngoại
post_id	int	Khóa ngoại

CHƯƠNG IV: XÂY DỰNG ỨNG DỤNG

4.1. Cấu trúc chung của dự án



Hình 14: Cấu trúc phân lớp các tầng của dự án

4.2. Các tầng xử lý chính

Dự án được chia thành nhiều tầng xử lý, tuy nhiên sẽ có 4 tầng chính thông suốt trong quá trình xủ lý các yêu cầu của người dùng

4.2.1. Tầng Controller

Tầng này có nhiệm vụ nhận yêu cầu từ phía người dùng, xác thực bảo mật của request để xem người truy cập vào tài nguyên là ai và có vai trò gì trong hệ thống. Nếu thỏa mãn sẽ tiếp tục cho dữ liệu đi xuống tầng bên dưới.

Khi dữ liệu được trả ra từ server, tầng này cũng có nhiệm vụ đưa những dữ liệu đó cho phía client. Trong trường hợp này, dữ liệu mà phía controller nhận và trả được định nghĩa dưới dạng JSON object.

Ví dụ cho chức năng chỉnh sửa bài viết

```
[HttpPut]
[Route("/api/post/{id}")]
[Authorize (AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
public IActionResult update(int id, PostRequest request, [FromQuery] List<int>
categories){

    int userId = Convert.ToInt32(HttpContext.User.FindFirst("Id").Value);

    ResponseObject<PostDTO> responseObject = postService.UpdatePost(userId,
id, request, categories);

    if (responseObject.Data == null) {
        return NotFound(responseObject);
    }
    return Ok(responseObject);
}
```

4.2.2. Tầng Service

Sau khi nhận được dữ liệu yêu cầu từ phía Controller, Service sẽ có nhiệm vụ xử lý logic chung và phân tán các dữ liệu cần xử lý riêng biệt cho các tầng trung gian khác để xử lý.

Tầng này đóng vai trò là cầu nối trung gian của cả hệ thống

Ví dụ cho chức năng chỉnh sửa bài viết

```
public ResponseObject<PostDTO> UpdatePost(int userId, int postId, PostRequest
request, List<int> categoryIds){

    var postToUpdate = dataContext.Posts
        .Include(entity => entity.User)
        .Include(entity => entity.Categories)
        .FirstOrDefault(post => post.Id.Equals(postId));

    if (!postToUpdate.User.Id.Equals(userId)) {
        return responseObject.responseError(StatusCodes.Status404NotFound,
            "Post doesn't belong to you", null);
    }

    if (postToUpdate == null){
        return responseObject.responseError(StatusCodes.Status404NotFound,
"Post not found or", null);
    }

    Post postTemporary = dataContext.Posts
        .Where(post => post.User.Id.Equals(userId) && post.IsActive == false)
        .OrderBy(post => post.Id)
        .LastOrDefault();  //nullable -> ok

    //if update thumbnail -> get thumbnail from temporary post and delete
itself
```

```
    if (postTemporary != null) {
        postToUpdate.Thumbnail = postTemporary.Thumbnail;
        dataContext.Posts.Remove(postTemporary);
    }

    postConverter.requestToEntity(request, postToUpdate);

    var updatedCategoryList = categoryIds
        .Select(id => dataContext.Categories
            .FirstOrDefault(c => c.Id.Equals(id)))
        .ToList();

    postToUpdate.Categories.Clear();
    postToUpdate.Categories = updatedCategoryList;

    postToUpdate.ModifiedDate = DateTime.Now;
    dataContext.SaveChanges();
    var updatedPostDto = postConverter.entityToDto(postToUpdate);
    return responseObject.responseSuccess("Updated successfully",
updatedPostDto);
}
```

4.2.3. Tầng Converter

Đối với những dữ liệu nhạy cảm như mật khẩu người dùng hay chỉ đơn giản làm giảm bớt sự thừa thãi hoặc thiếu dữ liệu cho mỗi lần nhận và trả dữ liệu. Converter sẽ có nhiệm vụ chuyển đổi các dữ liệu từ đối tượng này qua đối tượng khác để đảm bảo tính "đủ" khi giao tiếp giữa client và server

Ví dụ chuyển đổi thông tin người dùng để trả ra cho client

```
public UserDTO entityToDto(User entity) {

    HttpContext context = httpContextAccessor.HttpContext;
    string baseUrl =
$"{context.Request.Scheme}://{context.Request.Host}/image/";

    return new UserDTO {
        Id = entity.Id,
        FirstName = entity.FirstName,
        LastName = entity.LastName,
        Email = entity.Email,
        About = entity.About,
        Gender = entity.Gender,
        Role = entity.Role,
        DateOfBirth = entity.DateOfBirth,
        AvatarPhoto = baseUrl + entity.AvatarPhoto,
//http://localhost:8080/...
        CoverPhoto = baseUrl + entity.CoverPhoto,
        CreatedDate = entity.CreatedDate,
        CreatedBy = entity.CreatedBy,
        ModifiedDate = entity.ModifiedDate,
```

```
            ModifiedBy = entity.ModifiedBy,
            isActive = entity.IsActive
        };
}
```

Trong trường hợp này, nếu không tiến hành chuyển dổi, dữ liệu hiên thị ra cho người dùng sẽ là một đối tượng chứa cả thông tin nhạy cảm như mật khẩu. Vậy nên hàm chuyển đổi này sẽ có vai trò lọc những dữ liệu đó ở lại và chỉ lấy những trường thông tin có thể công khai.

4.2.4. Tầng Repository

Repository hay DataContext là một tầng mà dữ liệu sẽ tương tác trực tiếp với database. Tầng này đã được .Net thiết kế phần lớn, nhiệm vụ của lập trình viên chỉ là thực thi và cấu hình những đối tượng cần lưu vào trong cơ sở dữ liệu.

```
public DbSet<User> Users { get; set; }

public DbSet<Post> Posts { get; set; }

public DbSet<Comment> Comments { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Relationship> Relationships { get; set; }

public DbSet<RefreshToken> RefreshTokens { get; set; }

public DbSet<UserPostVote> UserPostVotes { get; set; }
```

4.3. Kiểm thử:

4.3.1. Chức năng đăng ký :

Kịch bản kiểm thử : Người dùng nhập vào các loại thông tin được yêu cầu như Họ tên , tên đăng nhập , địa chỉ email , mật khẩu và khâu nhập lại mật khẩu , nếu nhập không đúng yêu cầu sẽ đưa ra cảnh báo nhập lại.

Kết quả kiểm thử : Đăng ký thành công , thông tin đăng ký được lưu vào cơ sở dữ liệu.

4.3.2. Chức năng đăng nhập :

Kịch bản kiểm thử : Người dùng sử dụng thông tin tên đăng nhập (hoặc địa chỉ email) kèm mật khẩu đã đăng ký trước đó tiến hành đăng nhập vào hệ thống , nếu sai thông tin hệ thống sẽ thông báo đăng nhập thất bại.

Kết quả kiểm thử : Đăng nhập thành công , hệ thống cho phép người dùng trải nghiệm đầy đủ các chức năng của trang web.

4.3.3. Chỉnh sửa profile cá nhân :

Kịch bản kiểm thử : Người dùng truy cập vào " thông tin tài khoản " trong profile cá nhân , sau đó tiến hành chỉnh sửa và cập nhật các thông tin như ngày tháng năm sinh , họ tên , giới tính , email , mô tả sau đó chọn " Lưu thông tin".

Kết quả kiểm thử : Người dùng nhận được thông báo cập nhật thông tin thành công và dữ liệu sẽ được cập nhật vào cơ sở dữ liệu.

4.3.4. Chỉnh sửa ảnh đại diện , ảnh bìa trong profile cá nhân:

Kịch bản kiểm thử : Người dùng trọn mục "bài viết của bạn" trong phần tài khoản , ở đây sẽ hiển thị bài viết cá nhan cũng như ảnh bìa , ảnh cá nhân của người dùng (nếu là tài khoản mới thì sẽ có sẵn ảnh mặc định dành cho người dùng). Người dùng sử dụng "edit" ở trên ảnh cá nhân cũng như ảnh bìa sau đó upload ảnh lên từ thiết bị đang sử dụng , sau khi upload người dùng ấn "Open".

Kết quả kiểm thử : Ảnh cá nhân và ảnh bìa được lưu thành công , người dùng sẽ thấy ảnh được hiển thị trên trang cá nhân và mini avatar.

4.3.5. Tạo bài viết :

Kịch bản kiểm thử : Người dùng sử dụng chức năng tạo bài viết trên giao diện website , yêu cầu nhập vào tiêu đề bài viết và nội dung có thể kèm theo hình ảnh , sau khi hoàn thiện người dùng ấn vào " tiếp theo " và chọn tag thể loại của bài viết sau đó chọn vào " tạo bài viết ".

Kết quả kiểm thử : bài viết được tạo thành công và hiển thị trên giao diện của website.

4.3.6. Chỉnh sửa bài viết :

Kịch bản kiểm thử : Người dùng sau khi tạo bài viết , bên phải giao diện bài viết đã tạo sẽ có một nút menu bao gồm "Xóa bài viết" và "Chỉnh sửa bài viết". Người dùng chọn "Chỉnh sửa bài viết" sau đó tiến hành chỉnh sửa , sau khi chỉnh sửa ấn vào " Tiếp theo" để lưu bài viết.

Kết quả kiểm thử : Người dùng sẽ nhận được thông báo chỉnh sửa thành công.

4.3.7. Tìm kiếm bài viết :

Kịch bản kiểm thử : Người dùng nhập tên tiêu đề bài viết cần tìm vào thanh tìm kiếm có sẵn trên website.

Kết quả kiểm thử : Bài viết người dùng cần tìm được hiển thị trên giao diện website.

4.3.8. Comment , chỉnh sửa comment :

Kịch bản kiểm thử : Người dùng sau khi tạo bài viết , có thể bình luận bên dưới bài viết , tương tự như bài viết người dùng có thể chỉnh sửa comment của mình ở nút menu của mục comment.

Kết quả kiểm thử : Comment hiển thị thành công , người dùng có thể chỉnh sửa comment và hiển thị comment sau khi chỉnh sửa.

4.3.9. Xóa bài viết & comment (đối với admin):

Kịch bản kiểm thử : Admin sử dụng chức năng xóa bài viết trong mục menu của bài viết và comment , sau đó chọn vào " Xóa bài viết " và " Xóa bình luận ".

Kết quả kiểm thử : Xóa bài viết và bình luận thành công , người dùng sẽ không thấy bài viết và bình luận xuất hiện trên giao diện website nữa.

4.3.10. Tạo category đối với admin:

Kịch bản kiểm thử : Admin sử dụng " Chức năng admin" trong mục tài khoản , tại đây sẽ có danh sách các bài viết , người dùng và các tag category. Admin chọn " Tạo mới một thể loại" , nhập tên thể loại cần tạo sau đó chọn " Thêm".

Kết quả kiểm thử : Category được thêm thành công vào tag category.

4.3.11. Quản lý bài viết :

Kịch bản kiểm thử : Admin sử dụng "chức năng admin" trong mục tài khoản , tại đây sẽ có danh sách các bài viết , người dùng và các tag category. Danh sách các bài biết sẽ được hiển thị , gồm 2 chức năng chinh " Xóa bài viết" và "Ghim bài viết", admin có thể thao tác sử dụng "Xóa" hoặc "Pin".

Kết quả kiểm thử: Bài viết xóa thành công và ghim thành công , sau khi xóa bài viết sẽ không hiển thị trên website , sau khi ghim bài viết sẽ được hiển thị trong mục được ghim trên website.

CHƯƠNG V: TÀI LIỆU THAM KHẢO

https://codegym.vn/blog/2020/06/29/net-core-la-gi-tong-quan-ve-net-core/?fbclid=IwAR1c9AQ78eXj1i8P7kraX3C-M3vakVAQxqVkGYrxv-V0Jn0pAEuHNeFK6EY

https://aws.amazon.com/vi/what-is/restful-api/?fbclid=IwAR2uVuu10_y2RmANSO1_PjV0AAgDrfCK_-LJszZpEm4uTNQrWQSRH-kEn5s

https://viblo.asia/p/code-vi-du-json-web-token-cho-restful-api-voi-spring-security-jwt-vyDZOzBkKwj?fbclid=IwAR2x2VnmwAAvUPxMuRu_s0qQ1g6y205VAoyw6A7OF1s63MuGofbHKM1YdHA

https://chiasekinang.com/entity-framework-la-gi/?fbclid=IwAR1mt2tqEe0WDoQfs9HUB6_08KV-UdrzcC52TauBVnqEPlMBMOgZMJUvrag

https://stackoverflow.com/