BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

NGÔ VĂN LÂU

XÂY DỰNG HỆ THỐNG GIAO DỊCH BẤT ĐỘNG SẢN SỬ DỤNG ASP.NET CORE VÀ NEXT.JS

ĐỒ ÁN NGÀNH

NGÀNH CÔNG NGHỆ THÔNG TIN

TP. HỒ CHÍ MINH, 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

NGÔ VĂN LÂU

XÂY DỰNG HỆ THỐNG GIAO DỊCH BẤT ĐỘNG SẢN SỬ DỤNG ASP.NET CORE VÀ NEXT.JS

Mã số sinh viên: 2151053034

ĐỒ ÁN NGÀNH

NGÀNH CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn: TRƯƠNG HOÀNG VINH

TP. HỒ CHÍ MINH, 2025

LỜI CẢM ƠN

Đầu tiên em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn của em là thầy Trương Hoàng Vinh, nhờ những hiểu biết và sự hướng dẫn tận tình của thầy đã giúp em hoàn thành đồ án tốt nghiệp và học được những kiến thức mới cho sự nghiệp sau này của em. Em xin chúc thầy sẽ tiếp tục thành công trên con đường giảng dạy của mình.

Em xin gửi lời cảm ơn khoa Công nghệ thông tin đã giảng dạy và tổ chức môn Đồ án tốt nghiệp giúp em ứng dụng những kiến thức đã học ở trường đồng thời cũng học được các kiến thức mới và áp dụng thực hiện và hoàn thành đồ án. Em xin chúc khoa Công nghệ thông tin sẽ phát triển hơn nữa.

Em xin gửi lời cảm ơn đến trường Đại học Mở Thành phố Hồ Chí Minh đã tổ chức giảng dạy và tạo điều kiện tổ chức học tập cho em cũng như những sinh viên của trường để học tập, nghiên cứu và phát triển. Em xin chúc trường không ngừng phát triển và mở rộng hơn nữa.

Cuối cùng em xin chân thành cảm ơn bạn bè và các anh chị đã giúp đỡ, hỗ trợ tận tình giúp em trong quá trình phát triển và hoàn thành đồ án. Em xin chúc anh chị và các bạn thành công trên con đường sự nghiệp của mình.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TÓM TẮT ĐỒ ÁN NGÀNH

Đồ án "Xây dựng hệ thống giao dịch bất động sản sử dụng ASP.NET Core và Next.js" hướng tới việc thiết kế và triển khai một nền tảng web hiện đại, hỗ trợ người dùng đăng tin, tìm kiếm và giao dịch bất động sản trực tuyến. Hệ thống được phát triển theo kiến trúc Microservices, kết hợp frontend bằng Next.js và backend bằng ASP.NET Core, cho phép phân tách rõ ràng giữa các chức năng như quản lý người dùng, bất động sản, thanh toán, ...

Các công nghệ được áp dụng bao gồm: PostgreSQL, Redis, RabbitMQ, gRPC, Docker, PayPal, giúp đảm bảo hiệu suất, khả năng mở rộng và khả năng triển khai thực tế. Ngoài ra, hệ thống còn tích hợp thanh toán qua PayPal và có khả năng quản lý hình ảnh thông qua MinIO.

Về chức năng, hệ thống cung cấp đầy đủ các tính năng cốt lõi: đăng ký, đăng nhập, thêm tin bất động sản, tìm kiếm, đánh giá và bình luận, thanh toán trực tuyến. Đây là bước đầu quan trọng trong việc hiện thực hóa một nền tảng giao dịch bất động sản minh bạch, dễ sử dụng và linh hoạt – có tiềm năng phát triển và thương mại hóa trong tương lai.

ABSTRACT

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

MỞ ĐẦU

Sự phát triển của công nghệ và nhu cầu ngày càng tăng trong việc tìm kiếm, mua bán, cho thuê bất động sản đã thúc đẩy việc xây dựng các hệ thống giao dịch trực tuyến thông minh và tiện lợi. Tuy nhiên, nhiều nền tảng hiện nay vẫn còn thiếu hụt các tính năng hỗ trợ thanh toán, đánh giá minh bạch và trải nghiệm người dùng tối ưu.

Xuất phát từ thực tế đó, đồ án "Xây dựng hệ thống giao dịch bất động sản sử dụng ASP.NET Core và Next.js" này được thực hiện với mục tiêu xây dựng một hệ thống giao dịch bất động sản hoàn chỉnh – cho phép người bán dễ dàng đăng tải thông tin, quản lý bất động sản, còn người mua có thể tìm kiếm, lọc và đặt cọc/thuê bất động sản một cách nhanh chóng. Hệ thống sử dụng kiến trúc microservices hiện đại, đảm bảo khả năng mở rộng và vận hành ổn định. Đồng thời áp dụng các công nghệ hiện đại như ASP.NET Core, Next.js, PostgreSQL, Redis, RabbitMQ, gRPC, Docker, PayPal để phát triển giúp hệ thống đáp ứng được các yêu cầu nghiệp vụ, dễ triển khai, chịu tải tốt và dễ dàng mở rộng.

Chương 1. TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu đề tài

Hiện nay, nhu cầu tìm kiếm, mua bán hoặc cho thuê bất động sản đang ngày càng tăng cùng với sự phát triển mạnh mẽ của công nghệ, mạng Internet và sự phổ biến của các thiết bị công nghệ như máy tính, laptop và đặc biệt là điện thoại thông minh, người dùng có xu hướng chuyển sang các nền tảng số để tiếp cận thông tin nhanh chóng, minh bạch và hiệu quả. Chính điều này đã thúc đẩy em phát triển một hệ thống quản lý và giao dịch bất động sản hoàn chỉnh, hiện đại, dễ sử dụng và đặc biệt là đáp ứng được các nhu cầu của người dùng như đăng tin bán hoặc cho thuê bất động sản, tìm kiếm nhanh chóng và chính xác, giao diện trực quan và dễ sử dụng, hỗ trợ thanh toán trực tuyến, đánh giá và bình luận trong bối cảnh các hệ thống hiện tại thiếu thân thiện với người dùng hoặc chưa áp dụng các công nghệ hiện đại.

1.2. Lý do chọn đề tài

Lý do chọn đề tài xuất phát từ nhu cầu tìm kiếm một hệ thống trực quan dễ sử dụng và đáp ứng các yêu cầu của một hệ thống quản lý và giao dịch trực tuyến bất động sản của người mua hoặc thuê và chủ sở hữu bất động sản trong bối cảnh phát triển mạnh mẽ của Internet và sự phổ biến của các thiết bị điện tử như máy tính, laptop và đặc biệt là điện thoại thông minh đồng thời các hệ thống hiện tại chưa đáp ứng được các yêu cầu đó.

Việc thực hiện đề tài này sẽ tạo ra một hệ thống đáp ứng được các yêu cầu hiện tại của người dùng đồng thời giúp chủ sở hữu bất động sản tiếp cận khách hàng một cách nhanh chóng, dễ dàng và tiết kiệm thời gian. Không những thế, khi lựa chọn đề tài này còn giúp em học tập, nghiên cứu và áp dụng các kiến thức đã học và công nghệ mới vào việc phát triển hệ thống.

Qua đó, em không chỉ có thể tạo ra một sản phẩm có giá trị thực tiễn giải quyết được nhu cầu thị trường mà còn tạo cho em một kinh nghiệm phong phú trong việc phát triển phần mềm công nghệ, nâng cao kỹ năng công nghệ góp phần vào sự nghiệp sau này của em.

1.3. Mục tiêu và phạm vi đề tài

Mục tiêu đề tài là xây dựng một hệ thống quản lý và giao dịch bất động sản trực tuyến với giao diện người dùng thân thiện và dễ sử dụng đồng thời đáp ứng được các yêu cầu của người mua hoặc thuê và chủ sở hữu bất động sản, giải quyết được các hạn chế mà các hệ thống hiện có đang gặp phải. Hệ thống sẽ được thiết kế với kiến trúc Microservices đảm bảo chịu tải tốt, độc lập và dễ phát triển đồng thời đảm bảo khả năng mở rộng sau này. Trong đó sẽ chia làm phần Front-end (giao diện người dùng) và phần Back-end theo kiến trúc Microservices triển khai API và xử lý các logic nghiệp vụ. Ngoài ra hệ thống còn áp dụng các công nghệ hiện đại vào việc xây dựng và triển khai hệ thống giúp hệ thống linh hoạt và mạnh mẽ. Từ đó giải quyết được các yêu cầu của các chủ sở hữu bất động sản như dễ dàng đăng tin, quản lý thông tin về các bất động sản của mình và cung cấp các tính năng tìm kiếm, giao dịch, thanh toán trực tuyến thuận tiện cho người mua hoặc thuê.

Phạm vi đề tài bao gồm việc xây dựng hệ thống cho phép các chủ sở hữu bất động sản đăng tải và quản lý thông tin, hình ảnh, mô tả về các bất động sản của họ. Ngoài ra, các báo cáo thống kê về doanh thu và hiệu quả giao dịch cũng sẽ được tích hợp, giúp các chủ sở hữu dễ dàng theo dõi và tối ưu hóa hoạt động kinh doanh của mình. Đối với người mua hoặc thuê, hệ thống cung cấp tính năng tìm kiếm, mua hoặc thuê bất động sản, cho phép thanh toán trực tuyến và xem thời hạn thuê của các bất động sản đã thuê. Bên cạnh đó, hệ thống cũng hỗ trợ tính năng đánh giá, bình luận về các bất động sản mà khách hàng đã trải nghiệm.

1.4. Phương pháp nghiên cứu

Để xây dựng hệ thống quản lý bất động sản, em áp dụng phương pháp nghiên cứu kết hợp giữa lý thuyết và thực tiễn, nhằm đảm bảo phát triển một hệ thống hiệu quả, dễ sử dụng và phù hợp với nhu cầu thực tế của người dùng. Các phương pháp nghiên cứu sẽ bao gồm các bước sau: Thu thập yêu cầu, phân tích và thiết kế hệ thống, lựa chọn công nghệ và cuối cùng là phát triển và triển khai hệ thống.

Đầu tiên là thu thập yêu cầu, ở bước này em sẽ tiến hành thu thập các yêu cầu của người dùng cuối như người mua hoặc thuê, chủ sở hữu bất động sản thông qua các việc như phỏng vấn, khảo sát và nghiên cứu tài liệu từ đó thu được các yêu cầu nghiệp

vụ mà hệ thống cần có để đáp ứng và phát triển. Ngoài ra em còn thực hiện việc khảo sát bằng việc dùng thử các hệ thống hiện có từ đó tìm ra các điểm mạnh và hạn chế để áp dụng và khắc phục trong hệ thống của em.

Tiếp theo là bước phân tích và thiết kế hệ thống, sau khi em đã có đầy đủ yêu cầu em sẽ tiến hành phân tích và thiết kế hệ thống. Trong đó việc đầu tiên em sẽ mô hình hóa yêu cầu bằng cách xây dựng các mô hình nghiệp vụ (use cases), sơ đồ tuần tự (sequence diagrams) để xác định các chức năng chính. Sau đó là xây dựng sơ đồ lớp (class diagrams) và lựa chọn cơ sở dự liệu phù hợp để đáp ứng yêu cầu, khả năng mở rộng và hiệu suất.

Bước tiếp theo là lựa chọn công nghệ, trong quá trình nghiên cứu em lựa chọn các công nghệ phù hợp với mục tiêu và yêu cầu của hệ thống. Các công nghệ bao gồm:

- Back-end: ASP.NET Core và Microservices để xây dựng các dịch vụ độc lập, dễ dàng bảo trì và mở rộng.
- Front-end: Next.js giúp tối ưu hóa trải nghiệm người dùng với khả năng render cả client-side và server-side, hỗ trợ SEO và tốc độ tải trang.
- Cơ sở dữ liệu: PostgreSQL làm cơ sở dữ liệu cho các dữ liệu có cấu trúc.
- Các công nghệ khác: PayPal cho việc thanh toán trực tuyến, Redis cho bộ nhớ cache phân tán, RabbitMQ cho hệ thống messaging, gRPC cho giao tiếp giữa các dịch vụ, Nginx cho api gateway, load balancer và tối ưu hóa hiệu suất, Docker cho việc đóng gói và triển khai hệ thống.

Cuối cùng là phát triển và triển khai, ở bước này em sẽ tiến hành theo các bước nhỏ cho từng service của kiến trúc Microservices. Đầu tiên là phát triển hệ thống sẽ tuân theo phương pháp Agile, trong đó các tính năng được phát triển theo từng sprint. Tiếp theo là kiểm thử với các bài kiểm thử sẽ được thực hiện để đảm bảo tính ổn định, hiệu suất và bảo mật của hệ thống. Các kiểm thử bao gồm kiểm thử đơn vị (unit test), kiểm thử tích hợp (integration test) và kiểm thử hệ thống (system test). Sau cùng là triển khai hệ thống trên môi trường đám mây (cloud) thông qua Docker để đảm bảo tính sẵn sàng cao, khả năng mở rộng và dễ dàng quản lý.

1.5. Bố cục báo cáo

Chương 1: Tổng quan đề tài, mục tiêu của chương này là trình bày tổng quan về đề tài, lý do chọn đề tài, mục tiêu và phạm vi thực hiện, phương pháp nghiên cứu được sử dụng và các công nghệ hỗ trợ.

Chương 2: Cơ sở lý thuyết, mục tiêu của chương này là trình bày các kiến thức nền tảng về công nghệ được sử dụng trong hệ thống như ASP.NET, Next.js, Entity Framework, JWT, Fluent Validation, Serilog, Microservices, cũng như các công cụ hỗ trợ như PostgreSQL, Redis, RabbitMQ, gRPC, Nginx, Docker, PayPal, …

Chương 3: Hệ thống giao dịch bất động sản , mục tiêu của chương này là trình bày chi tiết về bài toán, phân tích hệ thống qua các sơ đồ use case, sơ đồ tuần tự, sơ đồ lớp, kiến trúc tổng thể và các chức năng chính đã triển khai như đăng ký, đăng nhập, đăng tin, tìm kiếm và thanh toán bất động sản.

Chương 4: Kết luận và hướng phát triển, mục tiêu của chương này là tổng kết quá trình thực hiện đồ án, các kết quả đạt được, đồng thời đề xuất các hướng mở rộng và nâng cao hệ thống trong tương lai.

Chương 2. CƠ SỞ LÝ THUYẾT

2.1. ASP.NET Core

2.1.1. Giới thiệu về ASP.NET Core

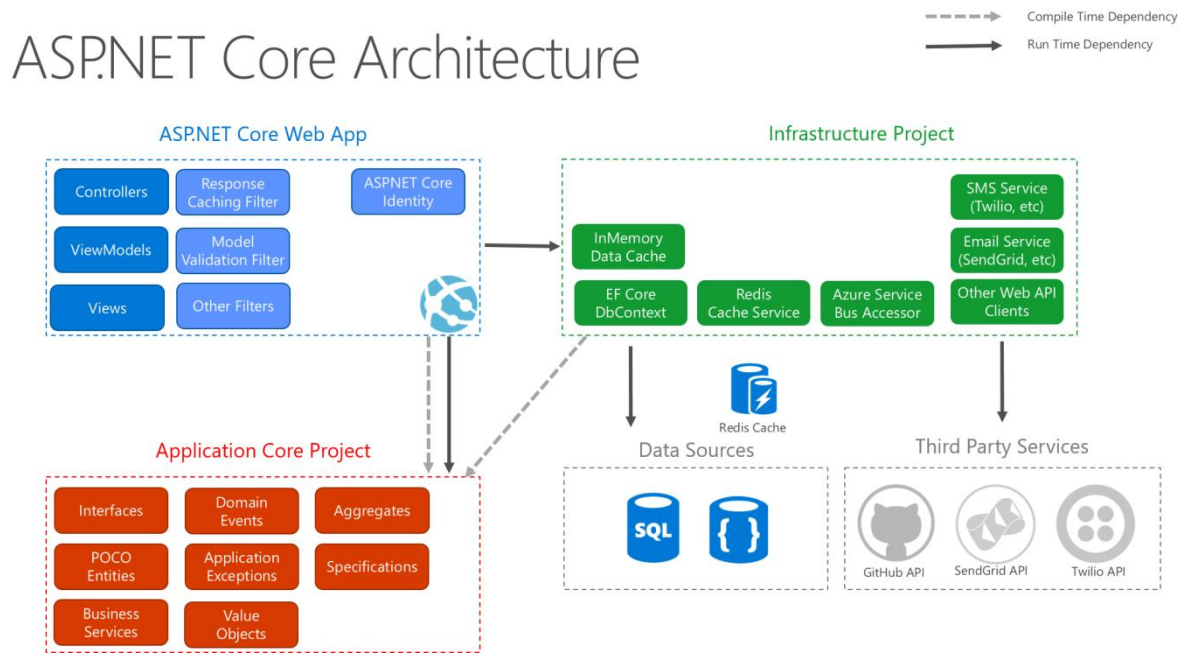
ASP.NET Core là một framework mạnh mẽ phát triển ứng dụng web mã nguồn mở, đa nền tảng do Microsoft phát triển, kế thừa và mở rộng từ ASP.NET truyền thống. Là một framework đa nền tảng nên chạy được trên Windows, MacOS và Linux. Framework cung cấp một môi trường phát triển mạnh mẽ với các công cụ và thư viện sẵn có, giúp tối ưu hóa quá trình phát triển và triển khai các ứng dụng. ASP.NET Core hỗ trợ xây dựng các loại ứng dụng khác nhau như: web app, web API, microservices, real-time app (SignalR) và cloud-based app. Ngoài ra với tính năng mở rộng và khả năng tích hợp các công nghệ khác một cách nhanh chóng và dễ dàng, ASP.NET Core là một lựa chọn lý tưởng cho việc xây dựng các hệ thống quy mô lớn, dễ bảo trì và có khả năng mở rộng trong tương lai.

Hình 2.1: ASP.NET Core

2.1.2. Tính năng nổi bật của ASP.NET Core

ASP.NET Core là một framework hết sức mạnh mẽ cho việc phát triển ứng dụng, nó mang trong mình các tính năng vô cùng nổi bật và được đánh giá là một trong những web framework có hiệu suất cao nhất theo các benchmark (TechEmpower, …). Đầu tiên là tính đa nền tảng, ASP.NET Core chạy được trên Windows, MacOS và Linux từ giúp các nhà phát triển linh hoạt trong việc triển và

triển khai. Bên cạnh, nó còn hỗ trợ cross-platform development – phù hợp với DevOps và triển khai container. Ngoài ra ASP.NET Core được đánh giá là framework có hiệu suất cao và nhẹ là nhờ sử dụng Kestrel – một lightweight web server tích hợp sẵn tối ưu cho hiệu suất đồng thời framework được tổ chức theo kiến trúc mô-dun hóa (Modular Architecture), chỉ cần nạp những thành phần cần thiết thông qua NuGet giúp giảm dung lượng và tăng hiệu suất từ đó giúp dễ dàng phát triển ứng dụng. Không chỉ có thế, ASP.NET Core còn hỗ trợ Dependency Injection (DI) mặc định, Middleware pipeline rõ ràng, dễ kiểm soát dòng xử lý HTTP. Bảo mật vả xác thực cũng vô cùng mạnh mẽ với các cơ chế xác thực JWT, cookie, OAuth2 được tích hợp sẵn đồng thời hỗ trợ Identity framework để quản lý user và quyền hạn. Việc triển khai cũng rất dễ dàng, linh hoạt với việc ASP.NET Core tích hợp tốt với Docker, Azure, Kubernetes và hỗ trợ CI/CD pipelines, logging, cấu hình linh hoạt qua appsettings.json, biến môi trường,… Đặc biệt là ASP.NET Core là một framework mã nguồn mở với cộng đồng vô cùng mạnh mẽ để hỗ trợ khi gặp khó khăn.



Hình 2.2 Kiến trúc ASP.NET Core

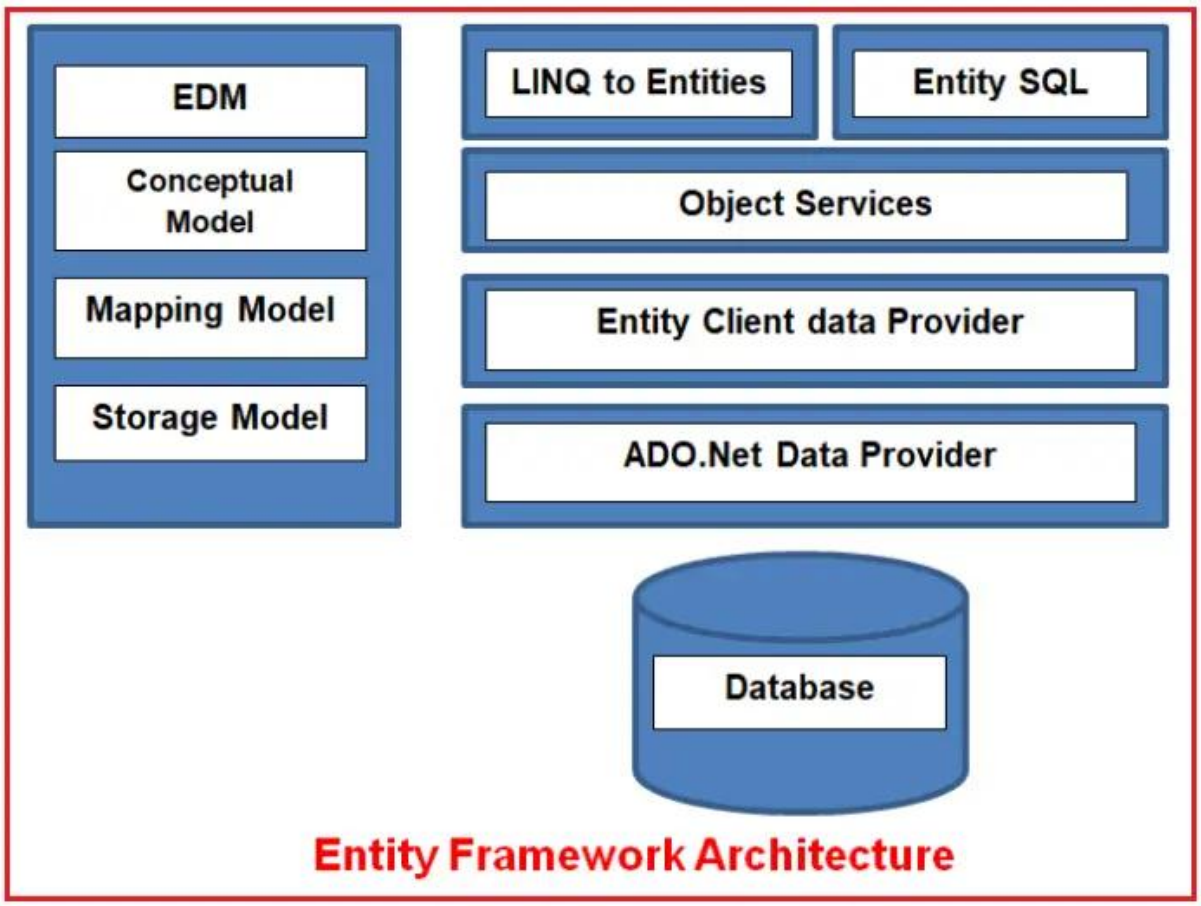
2.1.3. Entity Framework

Entity Framework (EF) là một thư viện ORM (Object-Relational Mapping) mạnh mẽ trong ASP.NET, cho phép lập trình viên làm việc với cơ sở dữ liệu thông qua các đối tượng trong C# thay vì phải viết các câu lệnh SQL thủ công. Bằng cách ánh xạ giữa các lớp trong mã nguồn và các bảng trong cơ sở dữ liệu, EF giúp đơn giản

hóa quá trình truy vấn, thêm, sửa và xóa dữ liệu, từ đó nâng cao hiệu quả lập trình và giảm thiểu lỗi.

Entity Framework hỗ trợ hai phương pháp tiếp cận chính khi làm việc với dữ liệu: Code-First và Database-First. Với phương pháp Code-First, lập trình viên định nghĩa các lớp trong mã nguồn trước, sau đó EF sẽ tự động sinh ra cơ sở dữ liệu dựa trên các lớp đó. Ngược lại, phương pháp Database-First cho phép sử dụng cơ sở dữ liệu đã tồn tại để tạo ra các lớp C# tương ứng, giúp dễ dàng tích hợp với hệ thống hiện có.

Việc sử dụng EF mang lại nhiều lợi ích như giảm thiểu sự phụ thuộc vào SQL thủ công, cải thiện khả năng bảo trì mã nguồn, đồng thời giúp lập trình viên tập trung vào việc xây dựng logic nghiệp vụ thay vì xử lý các chi tiết phức tạp của cơ sở dữ liệu.



Hình 2.3 Kiến trúc của Entity Framework

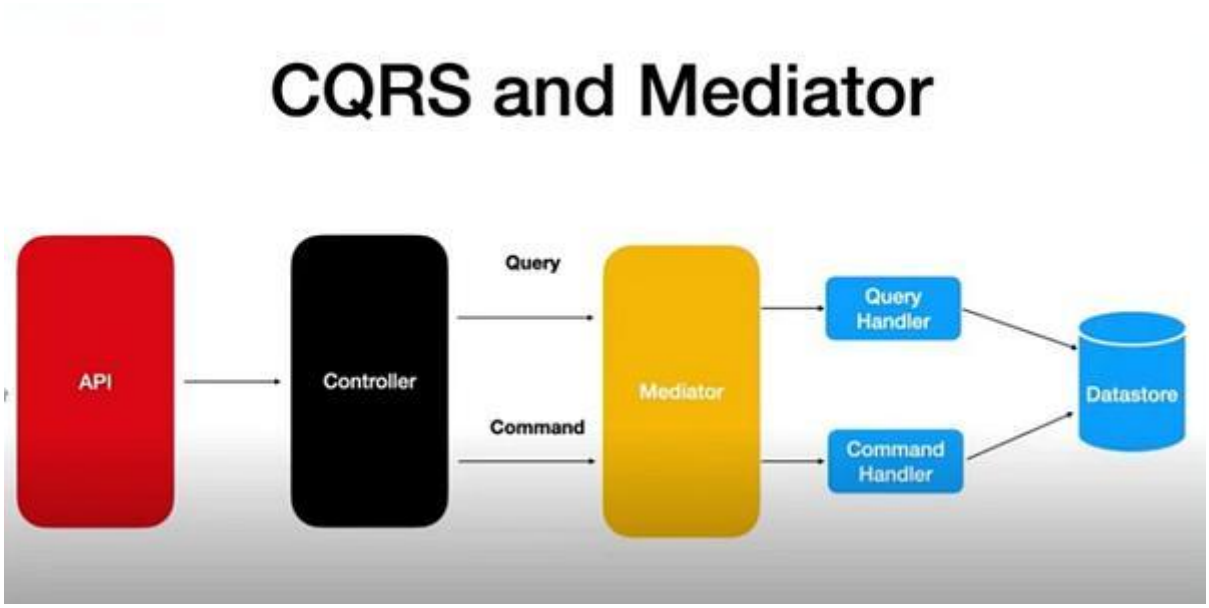
2.1.4. Mediator Pattern và CQRS

Đầu tiên là Mediator Pattern, đây là một pattern giúp giảm sự phụ thuộc giữa các thành phần trong hệ thống bằng cách đóng vai trò trung gian điều phối giao tiếp.

Thư viện MediatR là thư viện phổ biến nhất cho việc triển khai Mediator Pattern trong ứng dụng ASP.NET Core.

Tiếp theo là CQRS (Command and Query Responsibility Segregation) là một mô hình phân tách trách nhiệm của thao tác ghi (Command) và thao tác đọc (Query). Trong đó thao tác ghi (Command) đảm nhiệm xử lý các hành động thay đổi dữ liệu như create, update hoặc delete. Ngược lại thì thao tác đọc (Query) đảm nhiệm xử lý các hành động lấy dữ liệu đồng thời không làm thay đổi trang thái của hệ thống.

Việc kết hợp Mediator với CQRS giúp mã nguồn dễ tổ chức, tách biệt rõ ràng các luồng xử lý, tăng khả năng mở rộng, dễ test và bảo trì, giảm coupling giữa controller/service với logic nghiệp vụ đồng thời giúp mã nguồn có kiến trúc rõ ràng theo nguyên tắc SRP (Single Responsibility Principle) và dễ dàng mở rộng cho các tính năng như logging, validation, authorization nhờ pipeline behaviors của MediatR.



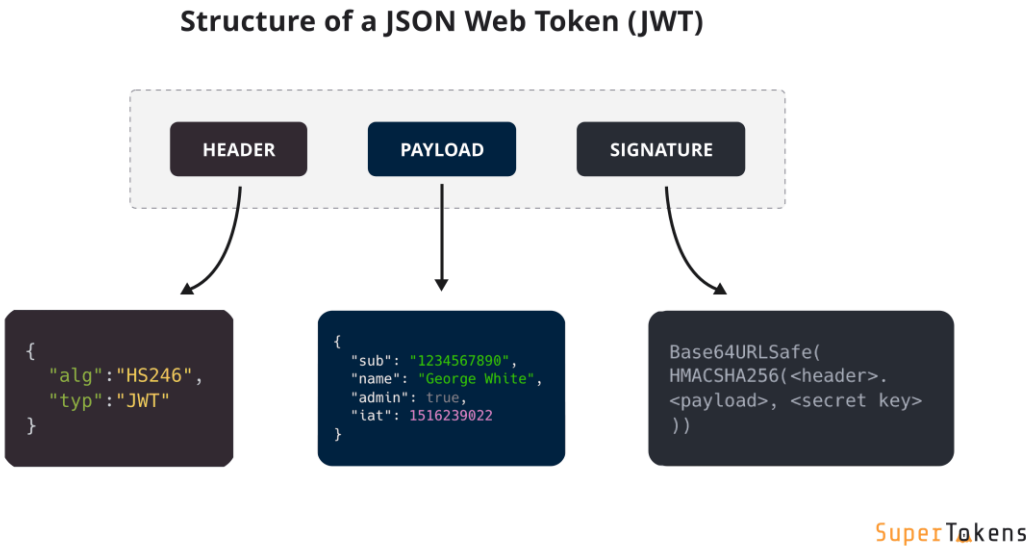
Hình 2.4 Kiến trúc của kết hợp Mediator với CQRS pattern

2.1.5. JWT

JWT (JSON Web Token) là một chuẩn mở (RFC 7519) dùng để truyền thông tin giữa các bên dưới dạng JSON đã được mã hóa và ký số để đảm bảo tính toàn vẹn và xác thực. JWT có ba thành phần chính:

- Header: chứa các thuật toán mã hóa như HS256, RS256, …
- Payload: chứa thông tin người dùng (claims).
- Signature: dùng để xác thực token không bị sửa đổi.

Quy trình hoạt động của JWT trong ASP.NET Core như sau: Khi người dùng đăng nhập thành công thì hệ thống ASP.NET Core sẽ tạo một JWT chứa các thông tin xác thực và gửi về client. Client sẽ lưu token trong cookie hoặc local storage và gửi lại hệ thống thông qua header Authorization trong các request sau. Với mỗi request hệ thống sẽ xác thực token thông qua middleware.



Hình 2.5 Kiến trúc JWT (nguồn: SuperTokens)

2.1.6. Fluent Validation

Fluent Validation là một thư viện mã nguồn mở trong ASP.NET Core giúp validate dữ liệu bằng cách định nghĩa các validation rule một cách rõ ràng và dễ dàng theo cú pháp fluent interface. Ưu điểm của Fluent Validation là cú pháp mạch lạc, dễ mở rộng, tăng khả năng tái sử dụng và test unit dễ dàng đồng thời tách biệt rõ validation logic khỏi controller/service giúp tuân thủ nguyên lý SRP (Single Responsibility Principle).

Để sử dụng trong ASP.NET Core bước đầu tiên thông qua NuGet cài đặt thư viện FluentValidation.AspNetCore, bước tiếp theo là tạo lớp validator kế thừa AbstractValidator<T>, cuối cùng là cấu hình trong Program.cs hoặc Startup.cs để tích hợp vào pipeline của ASP.NET Core sau đó khi cần sử dụng chỉ cần khai báo IValidator<T> trong constructor những nơi cần sử dụng.

2.1.7. Serilog

Serilog là một thư viện ghi log linh hoạt và mạnh mẽ dành cho ASP.NET Core đặc biệt nổi bật với khả năng ghi log có cấu trúc. Đồng thời hỗ trợ ghi log dưới dạng JSON giúp dễ dàng tích hợp với các hệ thống giám sát như ETK, Seq, Grafana, …

Serilog được sử dụng phổ biến với các ưu điểm nổi bật như dễ cấu hình và mở rộng, hỗ trợ nhiều sink như console, file, database, Elasticsearch, Seq, … Ngoài ra còn cung cấp nhiều level log như Verbose, Debug, Information, Warning, Error, Fatal, đặc biệt là có thể nhúng metadata vào log rất hữu ích cho việc truy vết lỗi phức tạp.

Để tích hợp Serilog đầu tiên thông qua NuGet cài đặt thư viện Serilog.AspNetCore, Serilog.Sinks.Console, Serilog.Sinks.File, … Sau đó thông qua UseSerilog() cấu hình logger trong Program.cs. Từ đó ta có thể sử dụng thông qua ILogger<T> đăng kí trong constructor hoặc trực tiếp Log.Information(…).

2.1.8. Microservices trong ASP.NET

Microservices là một kiến trúc phát triển phần mềm, trong đó ứng dụng được chia thành các dịch vụ nhỏ, độc lập, có thể phát triển và triển khai riêng biệt. Mỗi microservice thực hiện một nhiệm vụ cụ thể và có thể được phát triển, triển khai và bảo trì mà không ảnh hưởng đến các dịch vụ khác trong hệ thống.

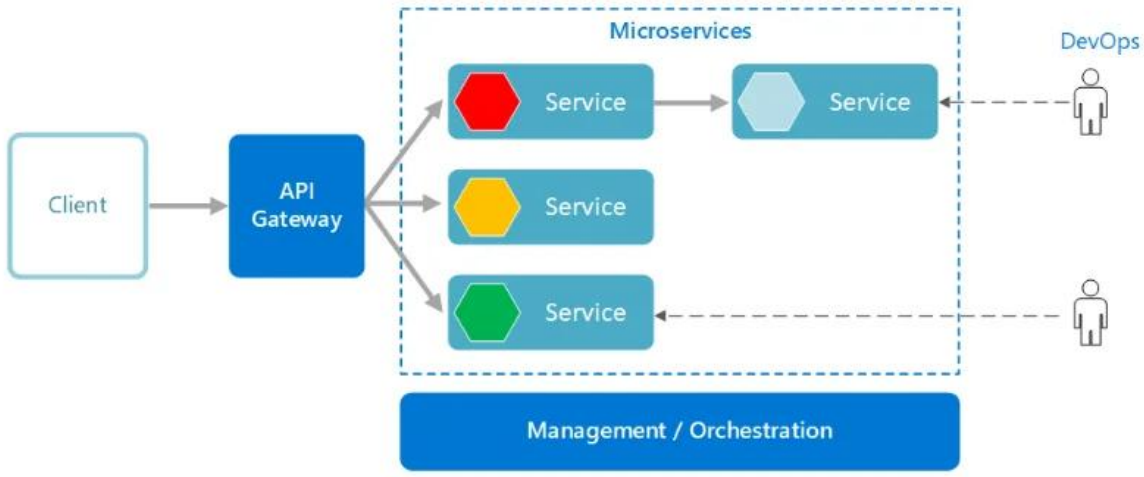
Ưu điểm của kiến trúc Microservices là dễ dàng mở rộng theo module, tăng khả năng triển khai độc lập, tránh ảnh hưởng lẫn nhau từ đó giúp cho ứng dụng chịu tải tốt, linh hoạt và dễ dàng mở rộng. Tuy nhiên bên cạnh các ưu điểm thì kiến trúc Microservices phải đối mặt với các thách thức như sự phức tạp trong quản lý các service hoặc lỗi liên service ngoài ra còn phải giải quyết các vấn đề như distributed transaction, consistency và observability.

Khi triển khai một hệ thống Microservices thường sẽ đi kèm theo một số thành phần:

- API Gateway: được dùng cho định tuyến, xác thực, rate limiting, …
- Service Discovery: được dùng cho định vị service và một số dịch vụ phổ biến cho việc đó như Consul, Eureka, …
- Configuration Management: được dùng để lưu và quản lý các cấu hình và một số dịch vụ phổ biến cho việc đó như Azure App Configuration, Consul.

- Monitoring và Logging: được dùng để giám sát hoạt động của hệ thống bằng cách tích hợp Serilog, Prometheus, Grafana, ELK Stack, …
- Communication: Các service trong kiến trúc Microservices thường được giao tiếp với nhau thông qua REST API, gRPC hoặc các message broker như RabbitMQ, Kafka.

Kiến trúc Microservices được ASP.NET Core hỗ trợ tốt nhờ khả năng cross-platform, hiệu suất cao và dễ dàng tích hợp các công nghệ hiện đại cho việc triển khai và container hóa như Docker, Kubernetes, … Ngoài ra ASP.NET Core còn hỗ trợ Dependency Injection, Middleware, gRPC, minimal API giúp dễ dàng phát triển các service riêng biệt.



Hình 2.6: Kiến trúc microservices

2.2. Next.js

2.2.1. Giới thiệu Next.js

Next.js là một framework mã nguồn mở được phát triển bởi Vercel và được xây dựng trên nền tảng React. Next.js được dùng để phát triển các ứng dụng web hiện đại với khả năng render phía server (SSR), tạo trang tĩnh (SSG) và tương tác client-side.

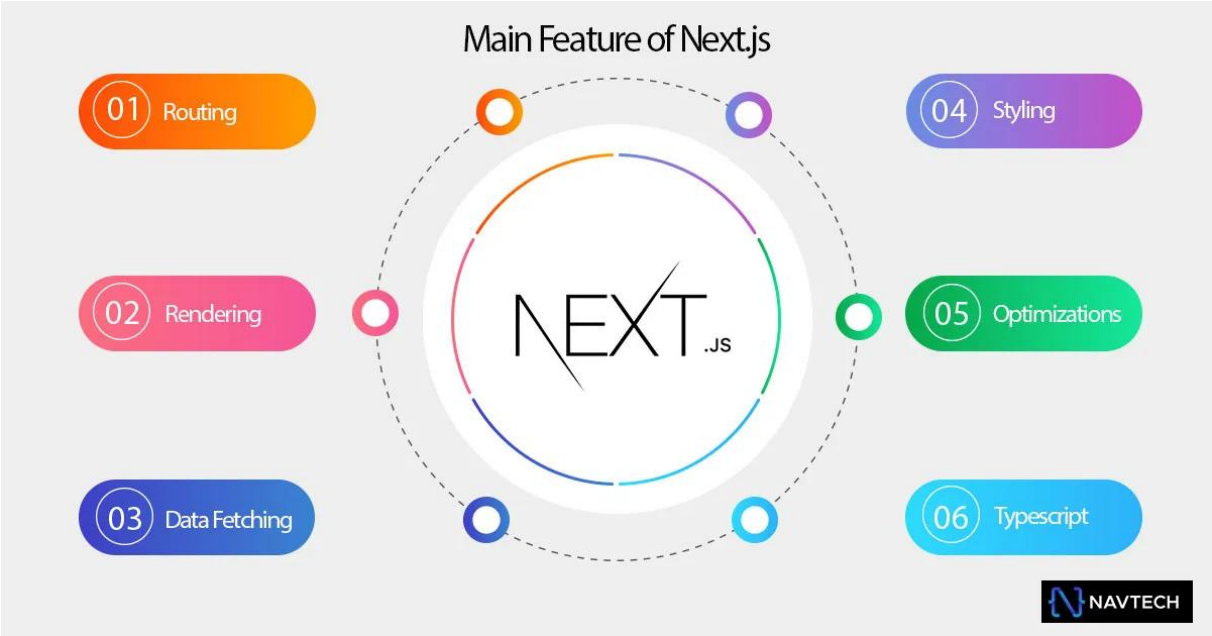
Điểm nổi bật của Next.js là hỗ trợ đa dạng các chiến lược rendering như Client-Side Rendering (CSR), Server-Side Rendering (SSR), Static Site Generation (SSG), Incremental Static Regeneration (ISR). Bên cạnh đó cấu trúc thư mục của Next.js vô cùng rõ ràng và hỗ trợ routing bằng hệ thống file-based routing giúp đơn giản hóa cho việc định tuyến. Đồng thời việc tích hợp Typescript, Tailwind CSS, … trong Next.js

cũng hết sức đơn giản. Và một điểm nổi bật nữa của Next.js là nó hỗ trợ fullstack, dễ dàng xây dựng API trực tiếp thông qua các file trong /api. Bên cạnh Next.js thường xuyên được hỗ trợ và cập nhật bởi nhà phát triển là Vercel đồng thời tài liệu chi tiết và cộng đồng nhà phát triển cũng vô cùng mạnh.

Từ những điều trên, Next.js rất phù hợp phát triển các ứng dụng Single Page Application cần SEO (Search Engine Optimization), các hệ thống yêu cầu tốc độ tải nhanh và khả năng mở rộng cao như các trang thương mại điện tử, blog, …

2.2.2. Các tính năng nổi bật của Next.js

Next.js được Vercel phát triển với rất nhiều tính năng nổi bật, đầu tiên không thể không kể đến đó là Hybrid Rendering, Next.js hỗ trợ và cho phép chọn phương thức render phù hợp cho từng trang như Server-Side Rendering (SSR), Static Site Generation (SSG), Client-Side Rendering (CSR), và Incremental Static Regeneration (ISR) giúp tối ưu hiệu suất và trải nghiệm người dùng. Tiếp theo là File-based Routing, đây là một hệ thống định tuyến theo cấu trúc thư mục, không cần cấu hình phức tạp gì chỉ cần trong các thư mục có chứa file với tên page.jsx hoặc page.tsx nếu sử dụng typescript thì sẽ trở thành định tuyến nếu sử dụng App Router còn với Page Router thì mỗi file trong thư mục pages/ sẽ tự động trở thành một route. Ngoài ra Next.js còn tích hợp API Routes cung cấp khả năng xây dựng API nhanh chóng và dễ dàng ngay trong dự án mà không cần tạo server riêng biệt. Không những thế, Next.js còn tối ưu hóa hình ảnh tự dộng, giảm kích thước ảnh và tối ưu định dạng ảnh phù hợp theo kích thước màn hình. Đồng thời còn hỗ trợ CSS linh hoạt và module hóa như CSS Modules, Tailwind CSS, SCSS, Styled-component dễ dàng, giúp quản lý CSS hiệu quả tránh xung đột. Dễ dàng tích hợp TypeScript với cấu hình đơn giản, giúp tăng độ an toàn và dễ bảo trì mã nguồn. Đặc biệt là hỗ trợ tốt cho phát triển ứng dụng quy mô lớn với việc tích hợp với hệ thống CMS, GraphQL, các dịch vụ cloud, CI/CD pipelines, và deploy dễ dàng lên Vercel hoặc các nền tảng khác dễ dàng và nhanh chóng.



Hình 2.7 Các tính năng của Next.js

2.2.3. Hướng dẫn tạo project Next.js

B1: Cài đặt Node.js tại trang Node.js — Run JavaScript Everywhere (nodejs.org).

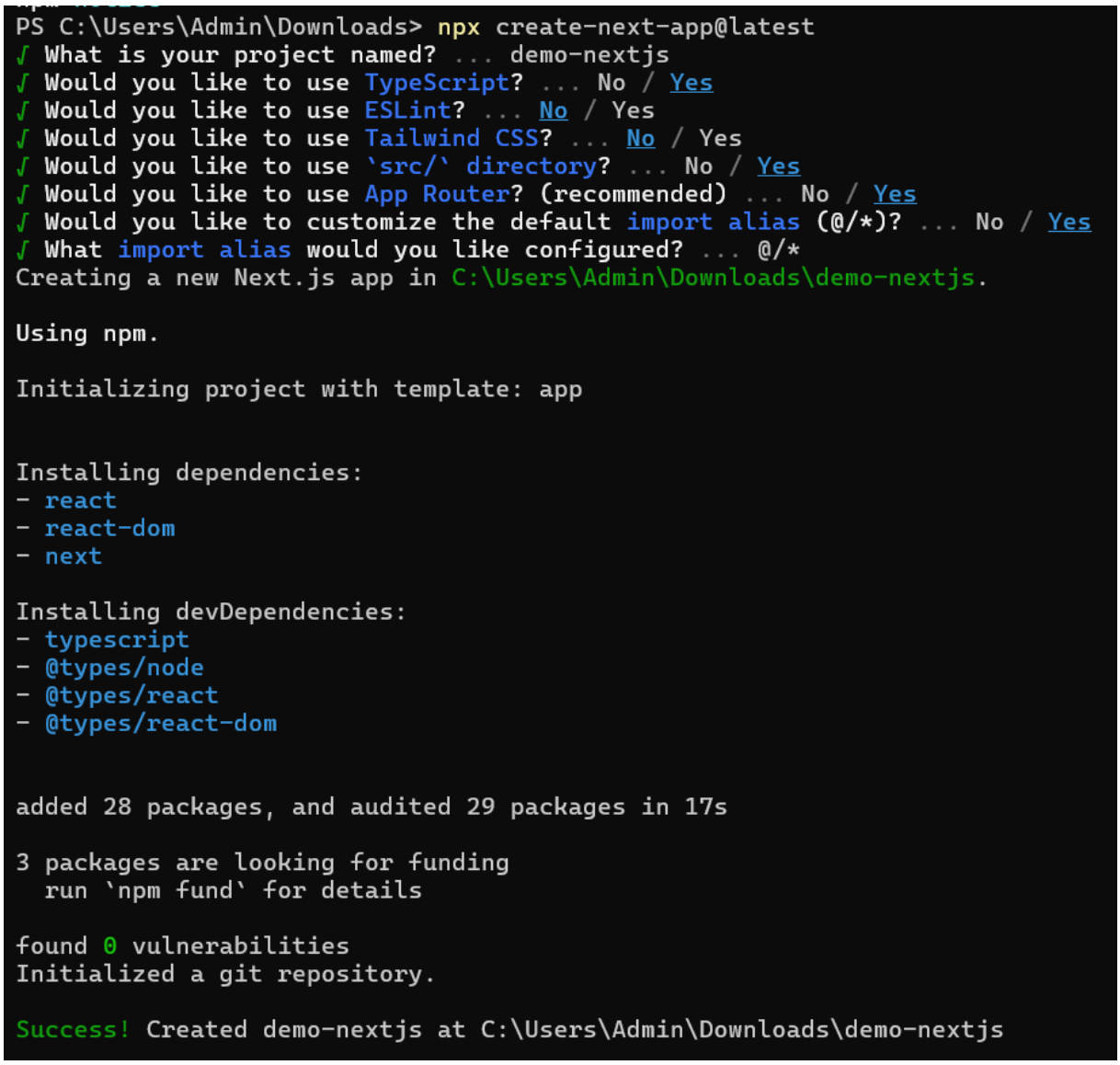
B2: Mở terminal ở nơi muốn tạo project.

B3: Gõ lệnh “npx create-next-app@latest”.

B4: Nhập tên project và lựa chọn cấu hình mong muốn.

B5: di chuyển vào project Next.js vừa tạo.

B6: Gõ lệnh “npm run dev” để chạy project.



Hình 2.8 Quá trình tạo project Next.js

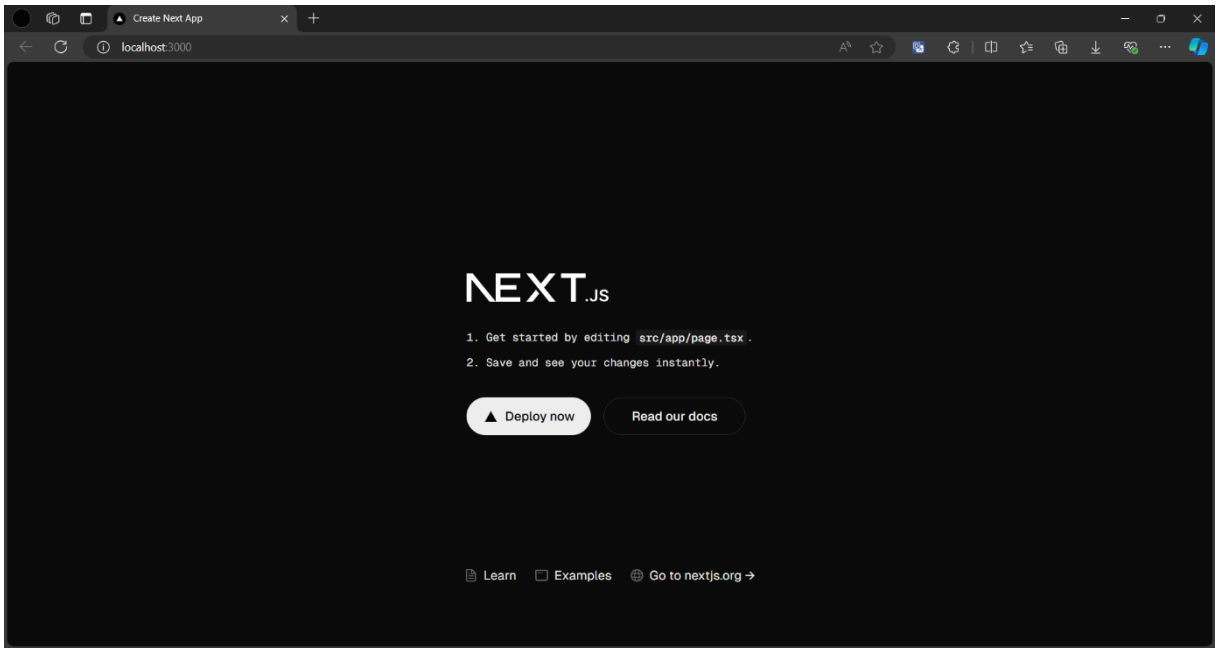
```
PS C:\Users\Admin\Downloads> cd .\demo-nextjs\
PS C:\Users\Admin\Downloads\demo-nextjs> npm run dev

> demo-nextjs@0.1.0 dev
> next dev

  ▲ Next.js 14.2.8
  - Local:        http://localhost:3000

 ✓ Starting...
 ✓ Ready in 2.3s
 ○ Compiling / ...
 ✓ Compiled / in 3.8s (555 modules)
 GET / 200 in 4029ms
 ○ Compiling /favicon.ico ...
 ✓ Compiled /favicon.ico in 1575ms (304 modules)
 GET /favicon.ico 200 in 1788ms
```

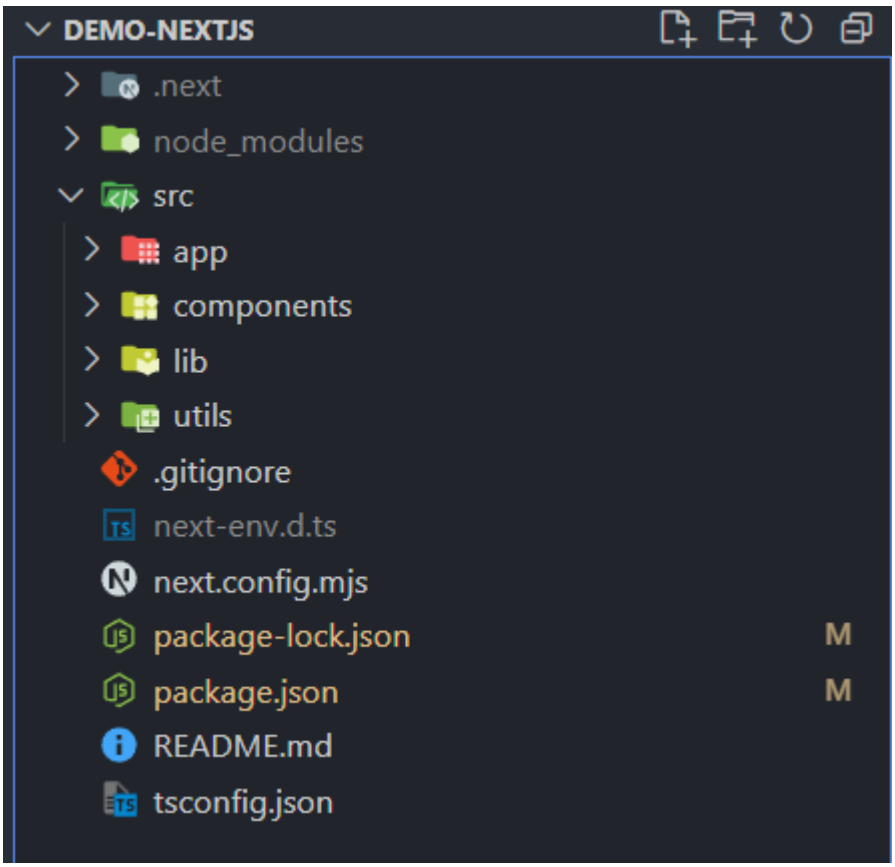
Hình 2.9 Chạy project Next.js thành công



Hình 2.10 Giao diện khi chạy project Next.js thành công

2.2.4. Cấu trúc thư mục Next.js

Hình ảnh sau đây mô tả cấu trúc thư mục của một project Next.js:



Hình 2.11 Cấu trúc thư mục của project Next.js

src/ là nơi chứa toàn mã nguồn của project Next.js:

- app/ là nơi quản lý tất cả các route của ứng dụng.
- components/ là nơi chứa các component dùng chung cho ứng dụng.
- lib/ là nơi chứa các thư viện và third-party của ứng dụng.
- utils/ là nơi chứa các hàm tiện ích dùng chung trong dự án.

next.config.mjs là file chứa cấu hình của Next.js.

package.json là file chứa các dependencies và scripts của project.

2.3. Các dịch vụ và công nghệ hỗ trợ

2.3.1. PostgreSQL

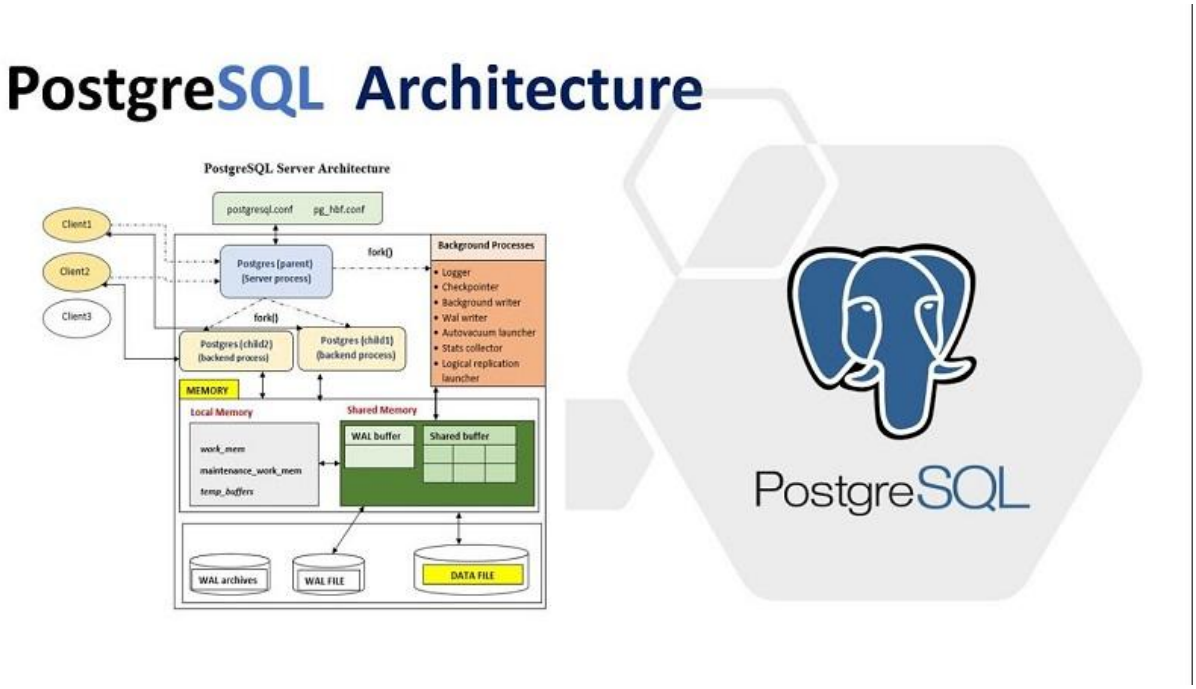
PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở phổ biến, mạnh mẽ, và được sử dụng rộng rãi trong cả ứng dụng nhỏ lẫn lớn. Đồng thời hỗ trợ nhiều loại dữ liệu và truy vấn phức tạp.

Điểm nổi bật của PostgreSQL là tuân thủ mạnh mẽ các chuẩn SQL, hỗ trợ đầy đủ các tính năng như truy vấn phức tạp, view, trigger, stored procedure, indexing.

Ngoài ra nhờ tối ưu hóa truy vấn thông minh và các tính năng indexing nâng cao mà PostgreSQL đạt được hiệu suất tốt. Đặc biệt là khả năng mở rộng cao nhờ hỗ trợ các hàm tùy chỉnh (UDF), các kiểu dữ liệu tuy biến.

Ngoài ra PostgreSQL còn nổi bật với việc hỗ trợ kiểu dữ liệu đa dạng với kiểu dữ liệu cơ bản như integer, text, boolean và kiểu dữ liệu nâng cao như JSON, JSONB, Array, UUID, GIS/spatial data. Không chỉ có thế, PostgreSQL còn có tính bảo mật và độ tin cậy cao nhờ vào các tính năng bảo mật mạnh mẽ như mã hóa dữ liệu, phân quyền người dùng linh hoạt, xác thực truy cập đồng thời có khả năng khôi phục dữ liệu tốt thông qua các cơ chế sao lưu, phục hồi và nhân bản.

Từ những điều trên, PostgreSQL được sử dụng nhiều vào các lĩnh vực như web, phân tích dữ liệu, ứng dụng di động, tài chính và viễn thông. Bên cạnh đó, nó còn dễ dàng tích hợp với nhiều ngôn ngữ lập trình và framework phổ biến.



Hình 2.12 Kiến trúc của PostgreSQL

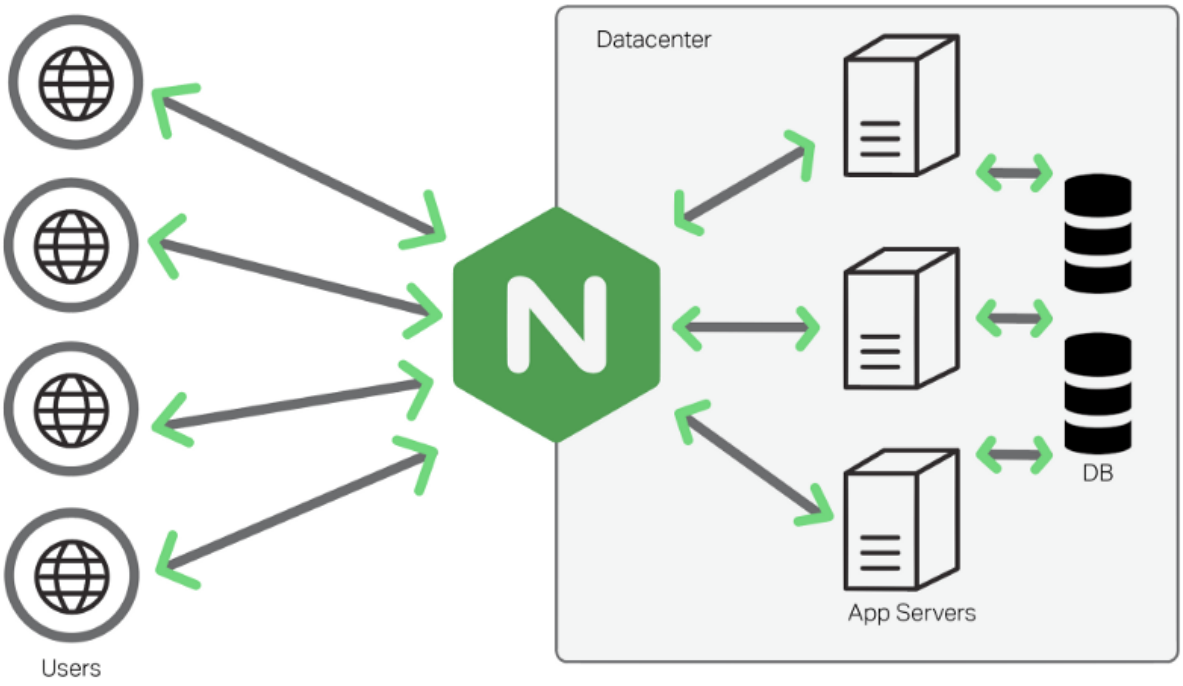
2.3.2. Nginx

Nginx là một web server mã nguồn mở với hiệu suất cao và được sử dụng rộng rãi. Nó có thể đảm nhiệm đồng thời các vai trò như load balancer, reverse proxy, HTTP cache và hỗ trợ streaming media.

Điểm nổi bật của Nginx là khả năng xử lý lượng lớn truy cập đồng thời mà vẫn đảm bảo hiệu suất vượt trội so với các web server khác. Không chỉ như thế, Nginx còn

tiêu thụ tài nguyên hệ thống như CPU, RAM vô cùng thấp giúp tiết kiệm chi phí và tăng khả năng mở rộng. Còn về bảo mật thì Nginx hỗ trợ SSL/TLS mạnh mẽ, chống các cuộc tấn công DDoS và khả năng giới hạn lưu lượng truy cập (rate limiting). Ngoài ra nó còn có khả năng tích hợp module bảo mật như ModSecurity giúp chống các cuộc tấn công web hiệu quả.

Trong kiến trúc web, Nginx đóng vai trò là một Reverse Proxy hỗ trợ định tuyến các request HTTP/HTTPS tới các dịch vụ backend một cách linh hoạt và an toàn hoặc là một Load Balancing giúp phân phối các yêu cầu đồng đều giữa các máy chủ, giúp tăng khả năng chịu tải và độ sẵn sàng. Ngoại ra để giảm tải cho server và cải thiện tốc độ phản hồi bằng cách cache các nội dung tĩnh với Nginx đóng vai trò là HTTP Caching.



Hình 2.13 Chức năng của Nginx

2.3.3. RabbitMQ

RabbitMQ là một phần mềm mã nguồn mở với vai trò là hệ thống message broker. Với việc sử dụng giao thức AMQP (Advanced Message Queuing Protocol), nó cho phép các ứng dụng trao đổi thông tin một cách bất đồng bộ, giúp giảm độ trễ và tăng hiệu suất của toàn hệ thống.

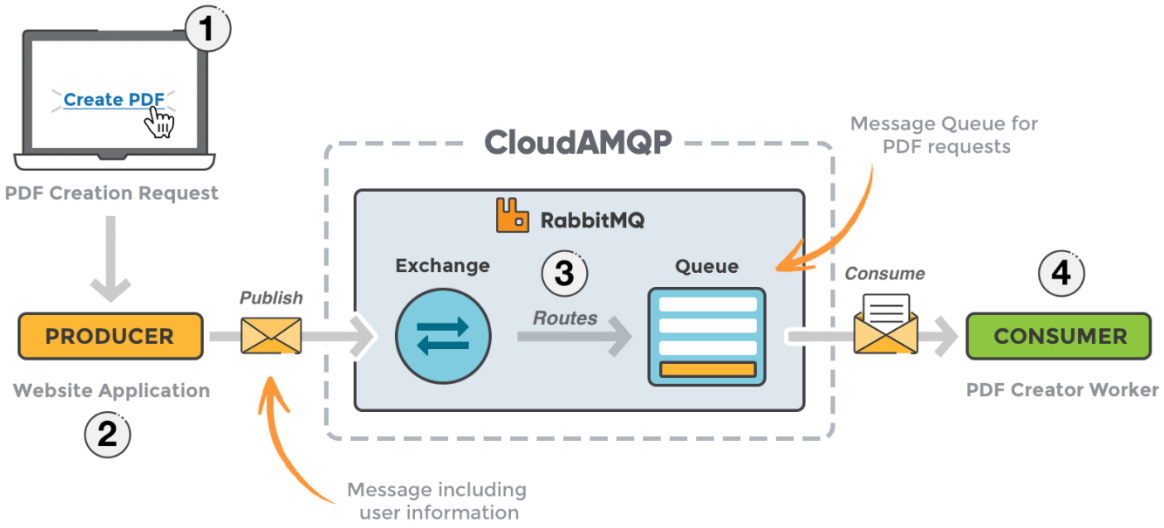
Điểm nổi bật của RabbitMQ là ngoài giao thức AMQP nó còn hỗ trợ các giao thức khác như MQTT, STOMP và HTTP. RabbitMQ cung cấp độ tin cậy và độ bền

cao với các tính năng như lưu trữ thông điệp (message persistance), quản lý hàng đợi và đảm bảo giao nhận thông điệp (delivery acknowledgment).

Kiến trúc của RabbitMQ bao gồm Exchanges, Queues, Binding và Routing keys nhờ vào đó mà nó linh hoạt trong việc định tuyến thông điệp thông qua các loại exchange như direct, topic, fanout và headers từ đó phù hợp với nhiều kịch bản sử dụng.

Khả năng mở rộng và quản lý là điều đáng được nhắc đến ở RabbitMQ với việc dễ dàng mở rộng theo ngang thông qua clustering và federation đồng thời cung cấp tính năng quản lý tập trung với giao diện quản trị dễ dùng là RabbitMQ Management Plugin.

Nhờ vào các điều trên mà RabbitMQ được ứng dụng rất phổ biến trong kiến trúc Microservices, đặc biệt là các hệ thống yêu cầu xử lý thông điệp theo thời gian thực. Ngoài ra nó còn được sử dụng rộng rãi trong các ứng dụng web, tài chính, logistics, IoT và hệ thống thông báo thời gian thực.



Hình 2.14 Luồng hoạt động của RabbitMQ

2.3.4. PayPal

PayPal là một nền tảng thanh toán điện tử toàn cầu, cho phép cá nhân và doanh nghiệp gửi và nhận tiền qua Internet một cách an toàn và nhanh chóng. Các chức năng chính của PayPal là hỗ trợ thanh toán trực tuyến cho các hệ thống thương mại điện tử

đồng thời cho phép liên kết với thẻ tín dụng, thẻ ghi nợ hoặc tài khoản ngân hàng để thực hiện thanh toán một cách dễ dàng.

Bên cạnh đó PayPal còn nổi bật với tính bảo mật cao và tin cậy nhờ cung cấp lớp bảo mật cao với mã hóa mạnh mẽ, xác thực hai lớp (2FA), và cơ chế phòng chống gian lận ngoài ra người dùng không cần chia sẻ thông tin thẻ trực tiếp với người bán, giúp tăng mức độ an toàn khi giao dịch. Đặc biệt là dễ dàng tích hợp vào các hệ thống qua API do PayPal cung cấp ngoài ra còn hỗ trợ các SDK cho nhiều nền tảng như JavaScript, Node.js, .NET, PHP, Java và mobile (iOS/Android).

Từ những điều trên mà PayPal được sử dụng rộng rãi bởi các nền tảng lớn như eBay, Shopify, và hàng triệu website bán hàng toàn cầu. Ngoài ra, PayPal còn cung cấp các giải pháp dành cho doanh nghiệp như thanh toán định kỳ (subscription), hóa đơn, và hỗ trợ giao dịch quốc tế.

Hình 2.15 Các tính năng của PayPal

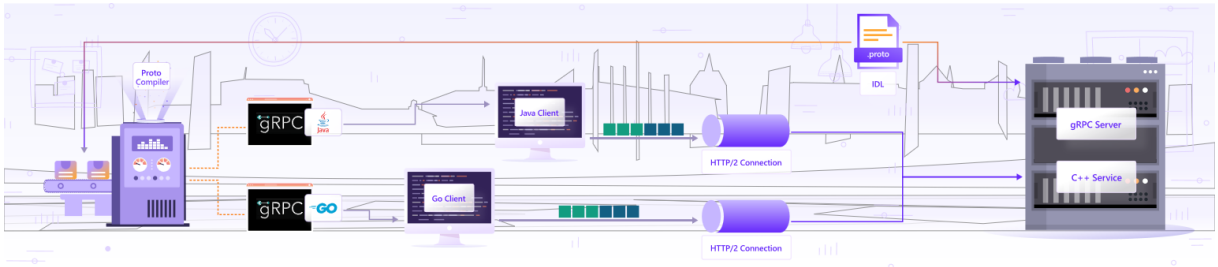
2.3.5. gRPC

gRPC là một framework mã nguồn mở sở hữu hiệu năng cao và được phát triển bởi Google nhằm mục đích xây dựng và kết nối các dịch vụ phân tán. gRPC sử dụng giao thức HTTP/2 để truyền tải và dữ liệu được định nghĩa bởi Protocol Buffers (protobuf) từ đó nó có một hiệu suất và tốc độ truyền dữ liệu cao. Bên cạnh đó nó còn hỗ trợ mạnh mẽ giao thức streaming hai chiều (bi-directional streaming) thích hợp cho các hệ thống xử lý dữ liệu thời gian thực.

Về cơ chế hoạt động thì gRPC sử dụng file proto để định nghĩa interface rõ ràng giúp việc phát triển, bảo trì và mở rộng các dịch vụ trở nên dễ dàng hơn. Ngoài ra việc tự động sinh mã nguồn từ file proto hỗ trợ đa ngôn ngữ lập trình giúp tiết kiệm thời gian phát triển và giảm lỗi khi tích hợp.

Đặc biệt trong kiến trúc Microservices thì gRPC hỗ trợ trong việc giao tiếp chặt chẽ giữa các service mang lại khả năng tương thích và giảm thiểu lỗi runtime từ đó tiết kiệm tài nguyên mạng và tăng hiệu năng nhờ dữ liệu nhỏ gọn, khả năng nén và truyền tải hiệu quả.

Với các ưu điểm tuyệt vời của gRPC mà nó được ứng dụng trong kiến trúc Microservices làm cầu nối giao tiếp và các hệ thống đòi hỏi hiệu suất cao như hệ thống phân tán, game, streaming, dịch vụ đám mây, … Đặc biệt nó được các công ty lớn như Google, Netflix tin cậy sử dụng trong các project của họ.



Hình 2.16 Cách hoạt động của gRPC

2.3.6. Docker

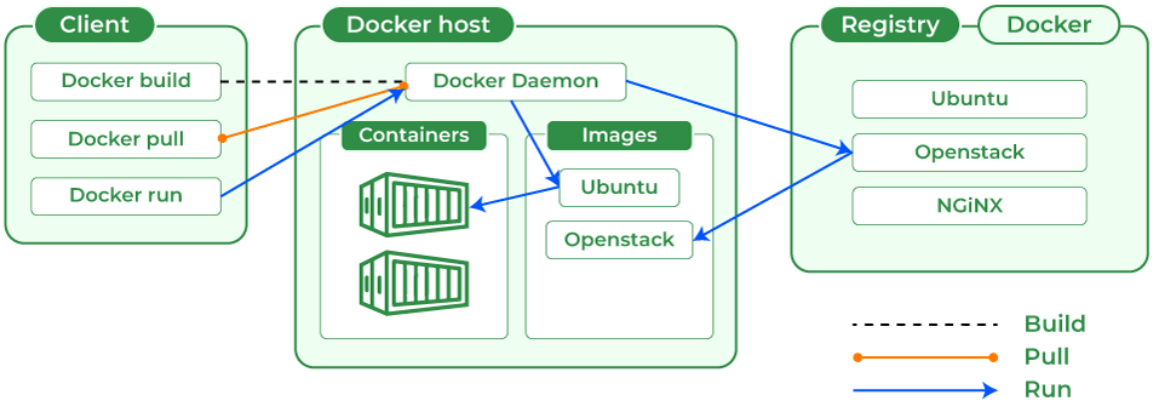
Docker là một nền tảng mã nguồn mở dùng để triển khai, vận hành và quản lý các ứng dụng dưới dạng container. Trong đó container giúp đóng gói ứng dụng cùng môi trường thực thi của nó một cách độc lập và thống nhất trên mọi hệ thống. Điểm nổi bật của Docker là khả năng cô lập ứng dụng và các thành phần liên quan khỏi hệ thống bên ngoài giúp giảm thiểu xung đột và tăng tính nhất quán từ đó đạt được hiệu suất cao, khởi động nhanh và sử dụng tài nguyên một cách tối ưu so với máy ảo.

Docker bao gồm:

- Image: là bản mẫu chứa mã nguồn ứng dụng và các phụ thuộc cần thiết.
- Container: là phiên bản thực thi cụ thể được tạo ra từ Image.
- Registry: là nơi lưu trữ và chia sẻ các Image ví dụ như Docker Hub.

Việc sử dụng Docker mang lại sự đơn giản hóa việc triển khai và phân phối ứng dụng, từ môi trường development đến môi trường production từ đó giảm thiểu sai sót do sự khác biệt môi trường. Ngoài ra việc sử dụng Docker còn tăng tốc độ phát triển phần mềm, giảm chi phí hạ tầng và quản trị hệ thống, dễ dàng mở rộng quy mô.

Docker thường được sử dụng trong việc phát triển hệ thống Microservices, CI/CD và DevOps và được nhiều công ty lớn như Google, Amazon, Netflix sử dụng rộng rãi để vận hành các dịch vụ quy mô lớn một cách nhanh chóng và hiệu quả.



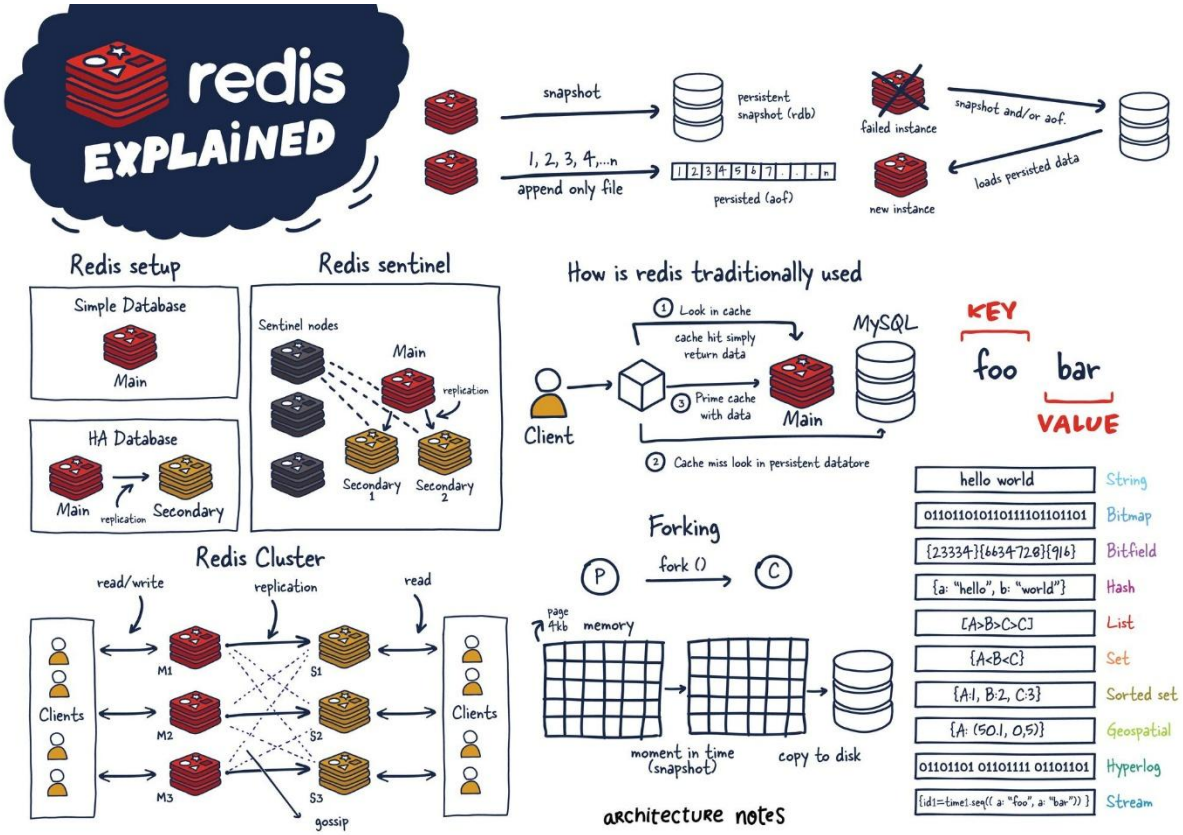
Hình 2.17 Kiến trúc của Docker

2.3.7. Redis

Redis (Remote Dictionary Server) là một cơ sở dữ liệu NoSQL mã nguồn mở hoạt động trên bộ nhớ in-memory và nổi bật với hiệu suất cao và độ trễ thấp. Đồng thời, Redis được sử dụng phổ biến cho việc cache, message broker hoặc cơ sở dữ liệu thời gian thực. Điểm nổi bật ở Redis là lưu trữ trực tiếp trên bộ nhớ RAM, giúp tốc độ đọc/ghi cực nhanh ngoài ra nó còn hỗ trợ nhiều cấu trúc dữ liệu linh hoạt như String, Hash, List, Set, Sorted Set, Stream, Bitmaps, HyperLogLog và Geospatial Indexes.

Redis vô cùng mạnh mẽ là nhờ các tính năng như Pub/Sub giúp trao đổi thông điệp thời gian thực giữa các ứng dụng, hỗ trợ persistence thông qua Snapshot (RDB) và Append-only file (AOF) giúp bảo vệ dữ liệu khi gặp sự cố. Ngoài ra Redis còn nổi bật với khả năng mở rộng và độ tin cậy với việc hỗ trợ clustering, sharding để mở rộng theo chiều ngang và tăng khả năng chịu tải đồng thời cung cấp các cơ chế nhân bản và phục hồi lỗi đảm bảo tính sẵn sàng cao.

Nhờ những điều trên mà Redis được sử dụng cho các việc cache dữ liệu nhằm giảm tải cơ sở dữ liệu và cải thiện hiệu suất của hệ thống. Bên cạnh đó Redis còn được dùng cho việc quản lý session, lưu trữ trạng thái người dùng nhanh chóng. Đặc biệt là sử dụng Redis trong hệ thống leaderboard, phân tích thời gian thực, các tác vụ liên quan tới địa lý, IoT và gaming.



Hình 2.18 Giải thích Redis

2.3.8. MinIO

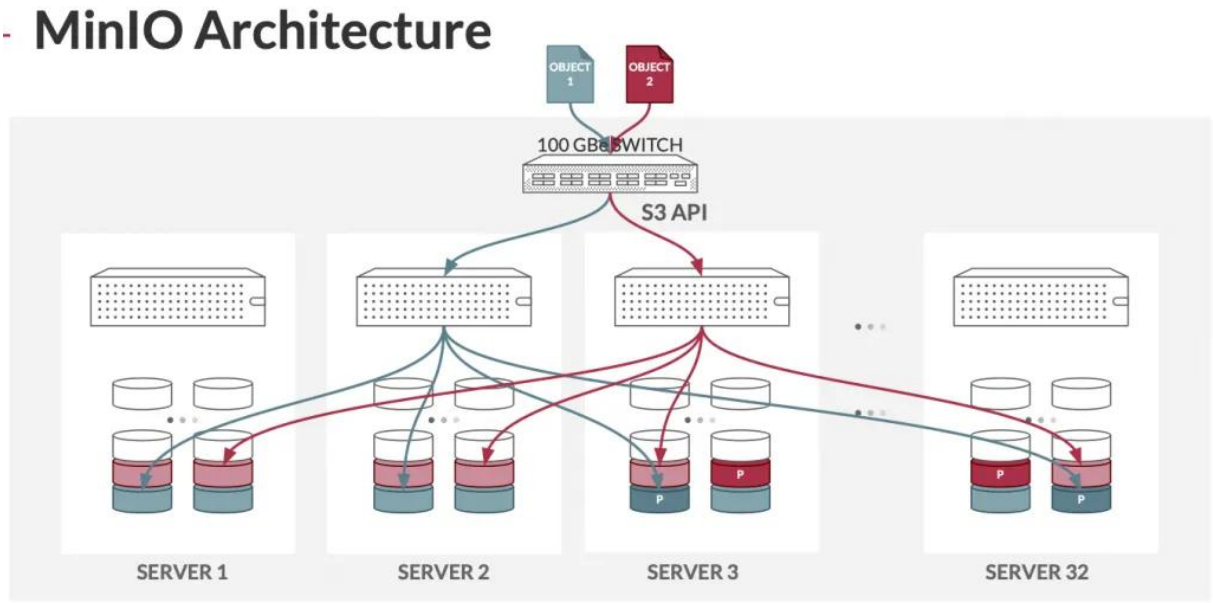
MinIO là một hệ thống lưu trữ đối tượng mã nguồn mở tương thích với Amazon S3 API. Đặc biệt nó được thiết kế để xử lý các yêu cầu lưu trữ d

ữ liệu phân tán với dung lượng lớn, phù hợp với các hệ thống hiện đại và cloud-native.

Điểm nổi bật của MinIO là có tính mở rộng cao, dễ dàng triển khai theo cụm để tăng dung lượng và độ bền dữ liệu ngoài ra nó còn có một hiệu suất vượt trội, tối ưu cho các tác vụ đòi hỏi thông lượng và độ trễ thấp đặc biệt là cho streaming và xử lý dữ liệu lớn. Bên cạnh đó MinIO còn hỗ trợ giao thức Amazon S3 giúp dễ dàng tích hợp vào các hệ thống có sử dụng chuẩn S3 ngoài ra không thể không đề cập đến khả năng

bảo mật mạnh mẽ, hỗ trợ mã hóa dữ liệu cả khi lưu trữ và khi truyền tải. Không chỉ có thế MinIO còn có giao diện quản trị dễ sử dụng, giúp quản lý dữ liệu, chính sách truy cập, giám sát hiệu suất và dung lượng hiệu quả. Nó còn hỗ trợ các cơ chế sao lưu, khôi phục và phục hồi lỗi đảm bảo dữ liệu an toàn, toàn vẹn và luôn sẵn sàng.

Nhờ đó MinIO thường được sử dụng làm kho lưu trữ dữ liệu tập trung, backup dữ liệu, lưu trữ file cho các hệ thống như web, video streaming hoặc phân tích dữ liệu. Đặc biệt nó còn dễ dàng tích hợp với các nền tảng container hóa như Kubernetes, Docker và phù hợp với kiến trúc Microservices.



Hình 2.19 Kiến trúc của MinIO

Chương 3. HỆ THỐNG GIAO DỊCH BẤT ĐỘNG SẢN

3.1. Giới thiệu bài toán

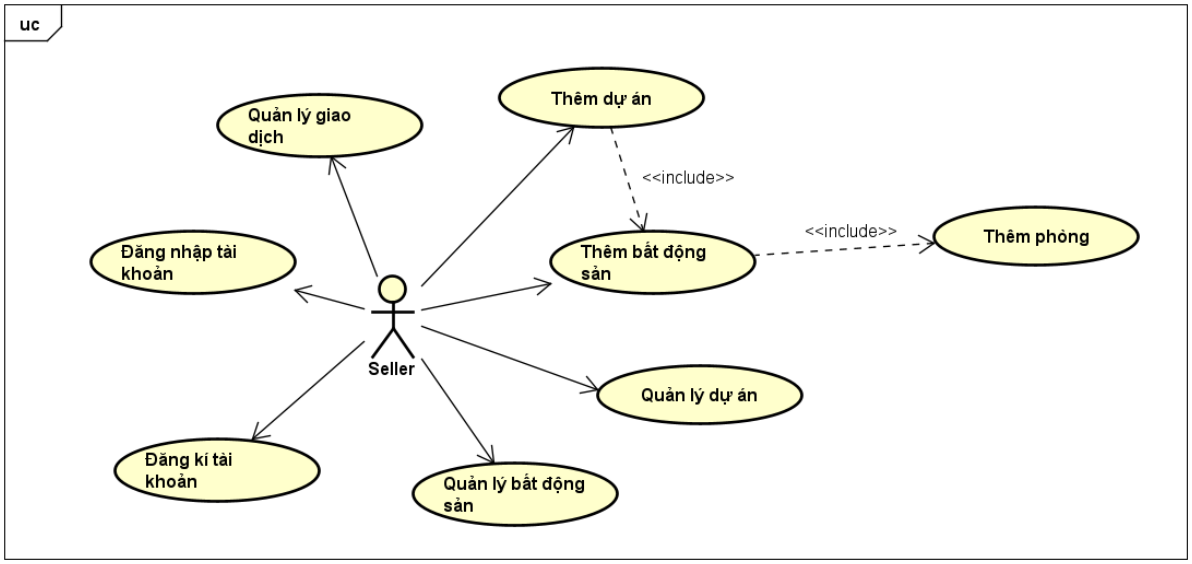
Mục tiêu của bài toán là xây dựng một hệ thống giao dịch bất động sản hoàn chỉnh đồng thời áp dụng được các công nghệ mới cũng triển khai, tích hợp các tính năng hiện đại vào hệ thống. Từ đó tạo ra được hệ thống đáp ứng nhu cầu dễ sử dụng, linh hoạt trong việc đăng thông tin bất động sản bán hoặc cho thuê cũng như cung cấp cho chủ sở hữu một nơi tiếp cận khách hàng nhanh chóng. Đồng thời người mua có một nơi thuận tiện cho việc tìm kiếm và mua hoặc thuê bất động sản.

3.2. Phân tích hệ thống

3.2.1. Sơ đồ use case

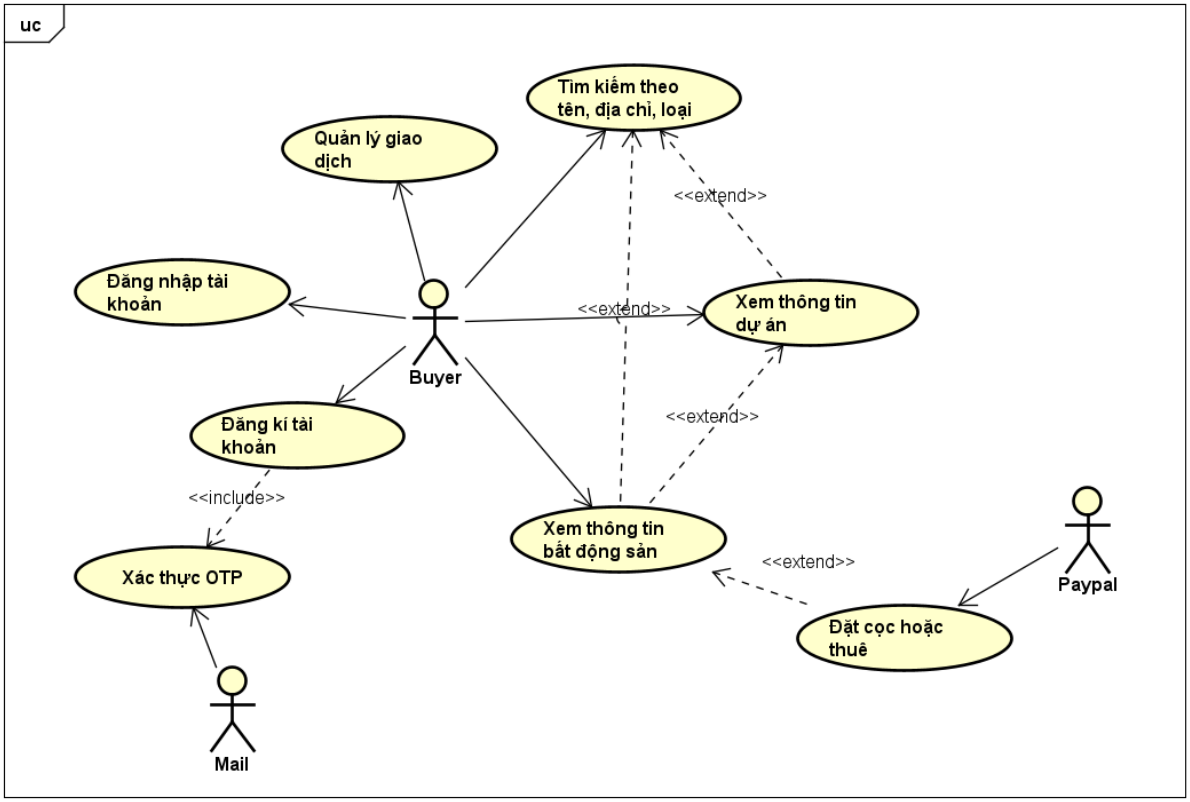
Hệ thống có 2 actor chính là seller và buyer. Và dưới đây là sơ đồ usecase của 2 actor trên mô tả các chức năng của hệ thống và sự tương tác của các actor với hệ thống thông qua các tính năng đó:

a) Sơ đồ usecase của actor seller mô tả tương tác của actor seller với hệ thống:



Hình 3.1 Sơ đồ usecase của actor seller

b) Sơ đồ usecase của actor buyer mô tả tương tác của actor buyer với hệ thống:



Hình 3.2 Sơ đồ usecase của actor seller

3.2.2. Đặc tả use case chức năng thêm bất động sản

Dưới đây là bảng đặc tả use case chức năng thêm bất động sản:

Usecase ID	UC-1.1
Tên usecase	Thêm bất động sản
Mô tả	Usecase này cho phép actor seller thêm thông tin của bất động sản lên hệ thống cho hệ thống hiển thị thông tin ấy lên hệ thống.
Actor chính	Seller
Tiền điều kiện	Seller đã có tài khoản và đăng nhập thành công hệ thống.
Hậu điều kiện	Thông báo thành công và hiện thông tin bất động sản vừa thêm.

Luồng hoạt động	1. Seller đăng nhập vào hệ thống. 2. Chọn quản lý bất động sản trong dashboard 3. Chọn nút thêm bất động sản 4. Nhập đầy đủ thông tin bất động sản cần thêm. 5. Chọn nút lưu. 6. Hệ thống sẽ kiểm tra các thông tin và cập nhật cơ sở dữ liệu nếu kiểm tra hợp lệ. 7. Hệ thống sẽ trả về thông báo thành công và hiện danh sách bất động sản
Luồng thay thế	Ở bước 1 nếu seller đăng nhập thất bại thì thông báo lỗi và cần đăng nhập lại.
Luồng ngoại lệ	Trong quá trình thực hiện nếu xảy ra lỗi thì hệ thống sẽ thông báo lỗi.

Bảng 3.1 Bảng đặc tả chức năng thêm bất động sản

3.2.3. Đặc tả use case đặt cọc hoặc thuê bất động sản

Dưới đây là bảng đặc tả use case chức năng đặt cọc hoặc thuê bất động sản:

Usecase ID	UC-1.2
Tên usecase	Đặt cọc hoặc thuê bất động sản
Mô tả	Usecase này cho phép buyer đặt cọc hoặc thuê bất động sản
Actor chính	Customer
Actor phụ	Paypal

Tiền điều kiện	Seller đã có tài khoản và đăng nhập thành công hệ thống và có tài khoản paypal. Có thông tin thông tin bất động sản và chưa được ai đặt cọc hoặc thuê.
Hậu điều kiện	Thông báo giao dịch của paypal.
Luồng hoạt động	1. Seller đăng nhập vào hệ thống. 2. Chọn bất động sản. 3. Chọn nút đặt cọc hoặc thuê. 4. Chọn nút thanh toán qua paypal. 5. Hệ thống sẽ chuyển qua trang thanh toán của paypal. 6. Chọn nút thanh toán. 7. Hệ thống sẽ thông báo kết quả giao dịch và sẽ lưu vào cơ sở dữ liệu nếu thanh toán thành công.
Luồng thay thế	Ở bước 1 nếu seller đăng nhập thất bại thì thông báo lỗi và cần đăng nhập lại.
Luồng ngoại lệ	Trong quá trình thực hiện nếu xảy ra lỗi thì hệ thống sẽ thông báo lỗi.

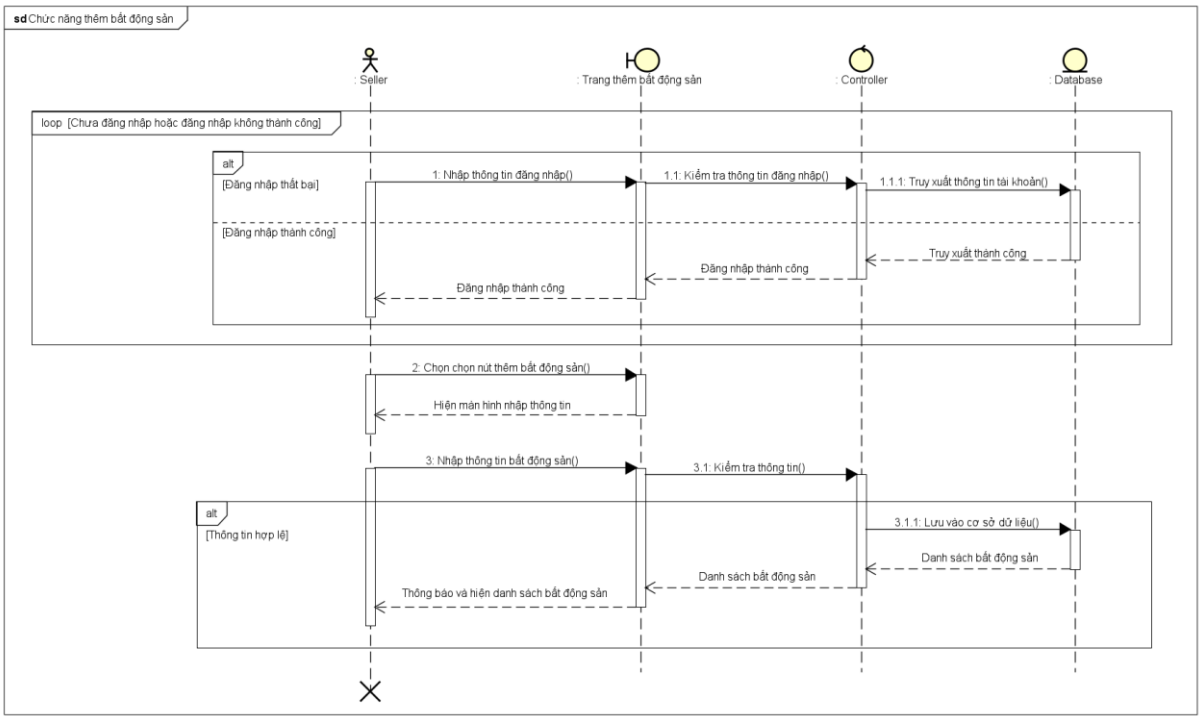
Bảng 3.2 Bảng đặt cọc hoặc thuê bất động sản

3.3. Thiết kế hệ thống

3.3.1. Sơ đồ tuần tự (Sequence Diagram)

a) Chức năng thêm bất động sản

Sơ đồ dưới đây mô tả quy trình thực hiện luồng hoạt động của chức năng thêm bất động sản:

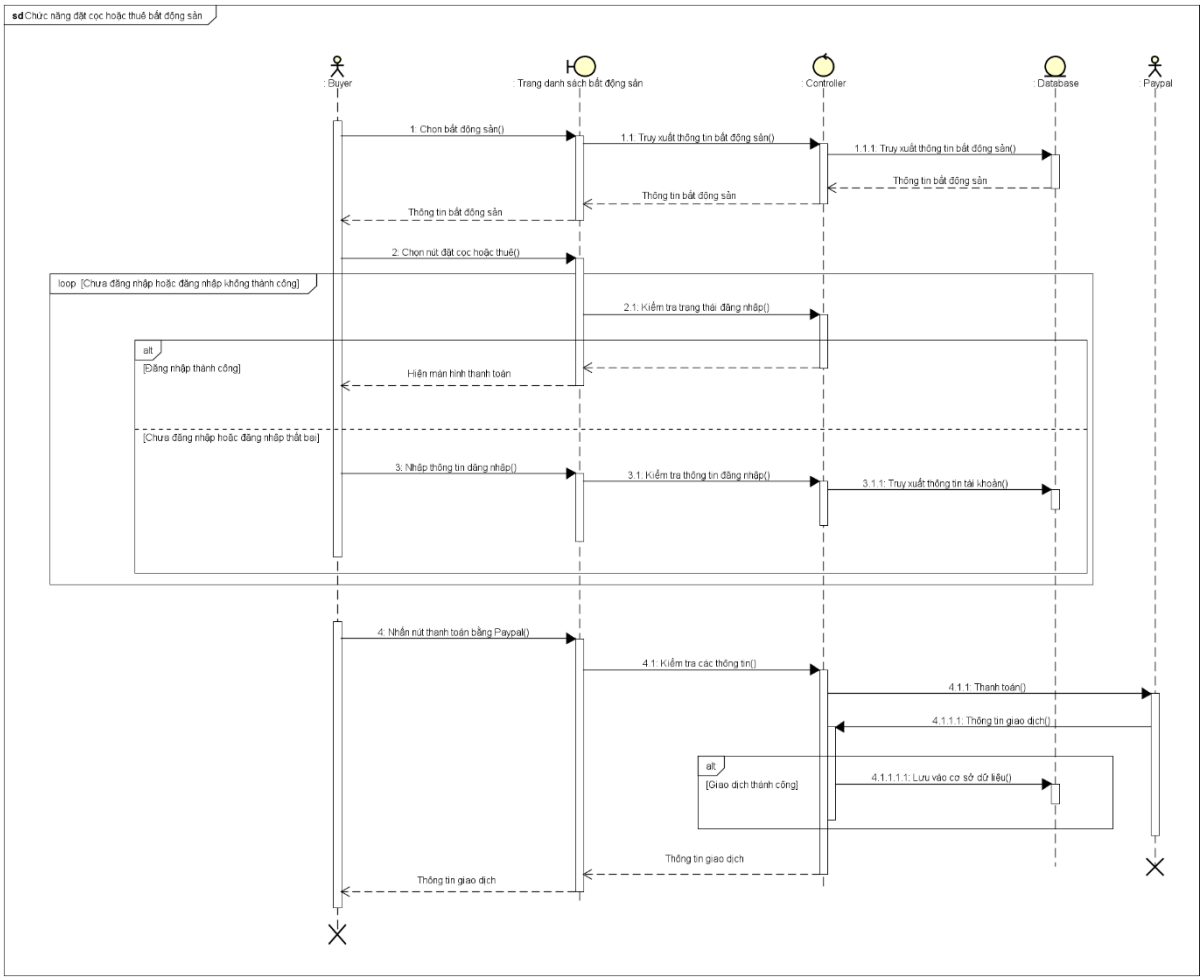


Hình 3.3 Sơ đồ tuần tự chức năng thêm bất động sản

Sơ đồ tuần tự mô tả quá trình Seller thực hiện thêm bất động sản vào hệ thống. Ban đầu, Seller cần đăng nhập sau đó thông tin đăng nhập được xác thực thông qua Controller và Database. Nếu xác thực thành công, Seller tiếp tục chọn chức năng thêm bất động sản và nhập thông tin bất động sản. Sau khi gửi thông tin, Controller sẽ kiểm tra, lưu dữ liệu vào Database và trả về danh sách bất động sản đã cập nhật để hiển thị danh sách bất động sản và kết thúc quá trình.

b) Chức năng đặt cọc hoặc thuê bất động sản

Sơ đồ dưới đây mô tả quy trình thực hiện luồng hoạt động của chức năng đặt cọc hoặc thuê bất động sản:

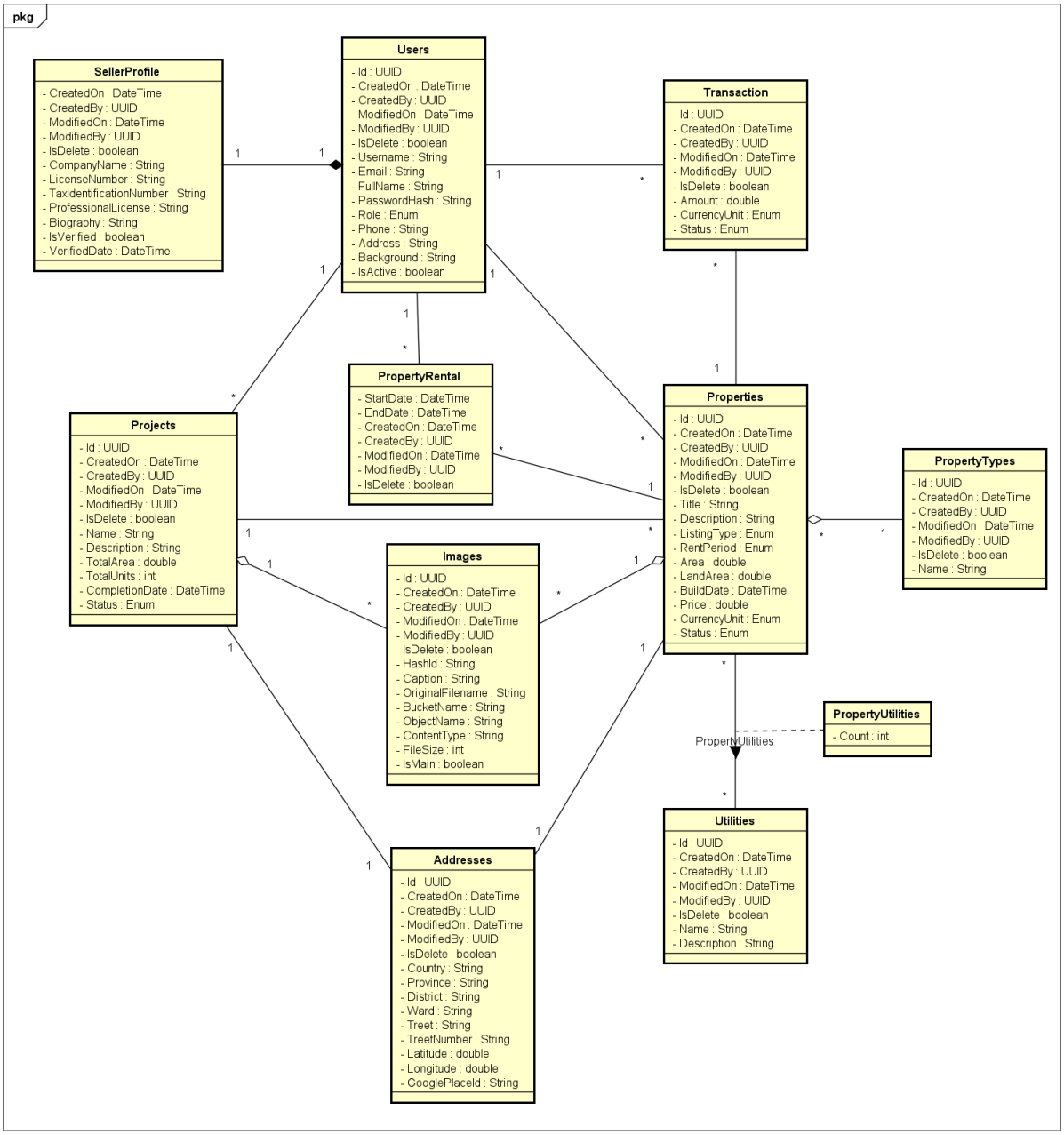


Hình 3.4: Sơ đồ tuần tự chức năng đặt cọc hoặc thuê bất động sản

Sơ đồ trình tự mô tả quá trình Buyer thực hiện đặt cọc hoặc thuê bất động sản. Buyer bắt đầu bằng cách chọn bất động sản từ trang danh sách. Thông tin bất động sản sẽ được lấy từ Controller và Database, sau đó hiển thị thông tin này cho Buyer. Khi Buyer chọn nút đặt cọc hoặc thuê, hệ thống sẽ kiểm tra trạng thái đăng nhập. Nếu chưa đăng nhập, Buyer phải tiến hành nhập thông tin đăng nhập và hệ thống sẽ xác thực thông tin qua Controller và Database. Khi đăng nhập thành công, màn hình thanh toán hiển thị để người dùng tiến hành thanh toán thông qua Paypal. Thông tin thanh toán được Controller xử lý và ghi lại vào Database nếu giao dịch thành công. Cuối cùng, hệ thống gửi thông tin giao dịch đến Buyer và kết thúc quá trình.

3.3.2. Sơ đồ lớp (Class Diagram)

Sơ đồ dưới đây là sơ đồ lớp mô tả các thuộc tính của các class và các quan hệ giữa các class với nhau:



Bảng 3.3 Sơ đồ lớp của hệ thống

Dưới đây là mô tả chi tiết các mối quan hệ trong sơ đồ class:

Users và SellerProfile (1-1): Mỗi người dùng (Users) chỉ sở hữu duy nhất một hồ sơ người bán (SellerProfile). Hồ sơ người bán này cung cấp thông tin bổ sung như tên công ty, giấy phép kinh doanh, và trạng thái xác thực. Ngược lại, mỗi hồ sơ người bán chỉ thuộc về một người dùng duy nhất.

Users và Projects (1-n): Một người dùng có thể sở hữu hoặc quản lý nhiều dự án bất động sản (Projects). Mỗi dự án sẽ có thông tin riêng như tên, mô tả, diện tích, số

lượng đơn vị, và ngày hoàn thành dự kiến. Tuy nhiên, mỗi dự án chỉ được tạo và quản lý bởi một người dùng duy nhất.

Users và Properties (1-n): Người dùng có khả năng sở hữu nhiều bất động sản (Properties), mỗi bất động sản có thông tin cụ thể về tiêu đề, diện tích, giá, loại giao dịch và trạng thái. Ngược lại, mỗi bất động sản chỉ được quản lý hoặc sở hữu bởi một người dùng duy nhất.

Users và Transaction (1-n): Một người dùng có thể thực hiện nhiều giao dịch bất động sản (Transaction). Mỗi giao dịch bao gồm các thông tin như số tiền giao dịch, loại tiền tệ và trạng thái giao dịch. Tuy nhiên, mỗi giao dịch chỉ được liên kết với một người dùng cụ thể.

Properties và PropertyTypes (n-1): Mỗi bất động sản thuộc về một loại bất động sản cụ thể (PropertyTypes), chẳng hạn như căn hộ, biệt thự hay nhà phố. Một loại bất động sản có thể bao gồm nhiều bất động sản khác nhau, nhưng mỗi bất động sản chỉ thuộc một loại duy nhất.

Properties và PropertyRental (1-n): Một bất động sản có thể được cho thuê nhiều lần, thông qua nhiều hợp đồng thuê khác nhau (PropertyRental). Mỗi lần thuê đều chứa thông tin về ngày bắt đầu, ngày kết thúc và trạng thái hợp đồng. Tuy nhiên, mỗi hợp đồng thuê chỉ liên quan đến một bất động sản cụ thể.

Properties và Transaction (1-n): Một bất động sản có thể tham gia vào nhiều giao dịch khác nhau, mỗi giao dịch phản ánh việc mua bán hoặc cho thuê bất động sản đó. Tuy nhiên, mỗi giao dịch chỉ liên quan đến một bất động sản cụ thể.

Properties và Images (1-n): Mỗi bất động sản có thể có nhiều hình ảnh minh họa (Images), giúp hiển thị các đặc điểm trực quan của bất động sản. Ngược lại, mỗi hình ảnh chỉ thuộc về duy nhất một bất động sản cụ thể.

Properties và Utilities thông qua PropertyUtilities (n-n): Bất động sản có thể liên kết với nhiều tiện ích (Utilities), như phòng bếp, phòng ngủ, ... Một tiện ích có thể áp dụng cho nhiều bất động sản, tạo ra mối quan hệ nhiều-nhiều thông qua bảng trung gian là PropertyUtilities.

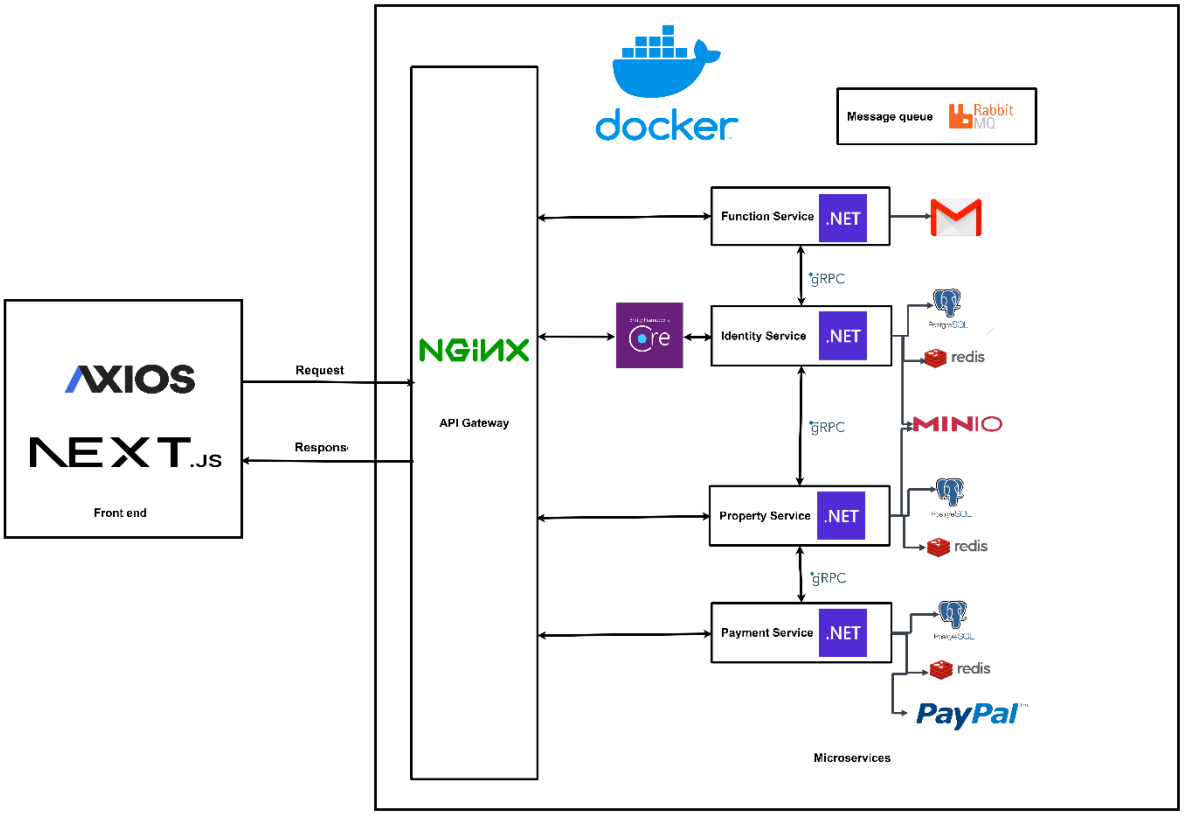
Projects và Images (1-n): Một dự án bất động sản có thể có nhiều hình ảnh minh họa đi kèm, thể hiện rõ hơn về thiết kế và các đặc điểm của dự án. Tuy nhiên, mỗi hình ảnh này chỉ thuộc về một dự án cụ thể.

Projects và Addresses (1-1): Mỗi dự án bất động sản sẽ có duy nhất một địa chỉ (Addresses), xác định rõ vị trí địa lý của dự án. Một địa chỉ chỉ áp dụng cho duy nhất một dự án.

Properties và Addresses (1-1): Mỗi bất động sản đều có địa chỉ duy nhất của mình, chỉ rõ nơi tọa lạc của bất động sản đó. Một địa chỉ cụ thể chỉ thuộc về một bất động sản duy nhất, đảm bảo sự chính xác và rõ ràng về vị trí.

3.4. Kiến trúc hệ thống

Hình ảnh dưới đây là kiến trúc của hệ thống buôn bán hoặc cho thuê bất động sản mô tả tổng quan cách mà các thành phần được cấu trúc và tương tác với nhau:



Hình 3.5 Kiến trúc hệ thống buôn bán hoặc cho thuê bất động sản

Kiến trúc hệ thống trên được thiết kế theo mô hình microservices hiện đại, sử dụng Docker để triển khai và quản lý toàn bộ dịch vụ. Phía giao diện người dùng

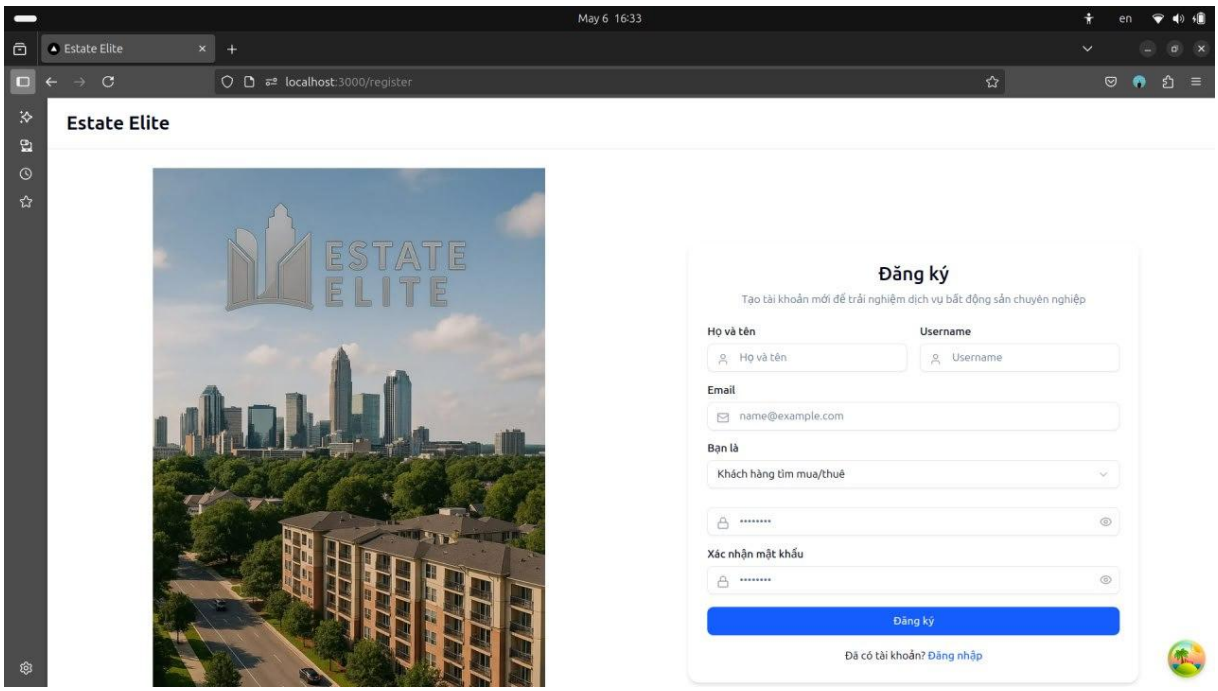
(frontend) được xây dựng bằng Next.js, sử dụng thư viện Axios để gửi yêu cầu đến API Gateway thông qua Nginx. API Gateway đóng vai trò điều phối các yêu cầu đến từng dịch vụ riêng biệt trong hệ thống.

Các dịch vụ backend được phát triển bằng .NET, giao tiếp nội bộ với nhau thông qua gRPC nhằm tối ưu hiệu suất. Trong đó, Identity Service đảm nhiệm xác thực người dùng, quản lý tài khoản và lưu trữ dữ liệu bằng PostgreSQL và Redis, đồng thời tích hợp Entity Framework Core để tương tác với cơ sở dữ liệu. Functional Service xử lý các tác vụ nền như gửi email và tương tác với RabbitMQ để quản lý hàng đợi tác vụ bất đồng bộ. Property Service quản lý thông tin bất động sản, kết nối với PostgreSQL, Redis và sử dụng MinIO để lưu trữ hình ảnh. Payment Service xử lý các giao dịch thanh toán, tích hợp với Redis, PostgreSQL và cổng thanh toán PayPal.

3.5. Chức năng hệ thống

3.5.1. Trang đăng ký

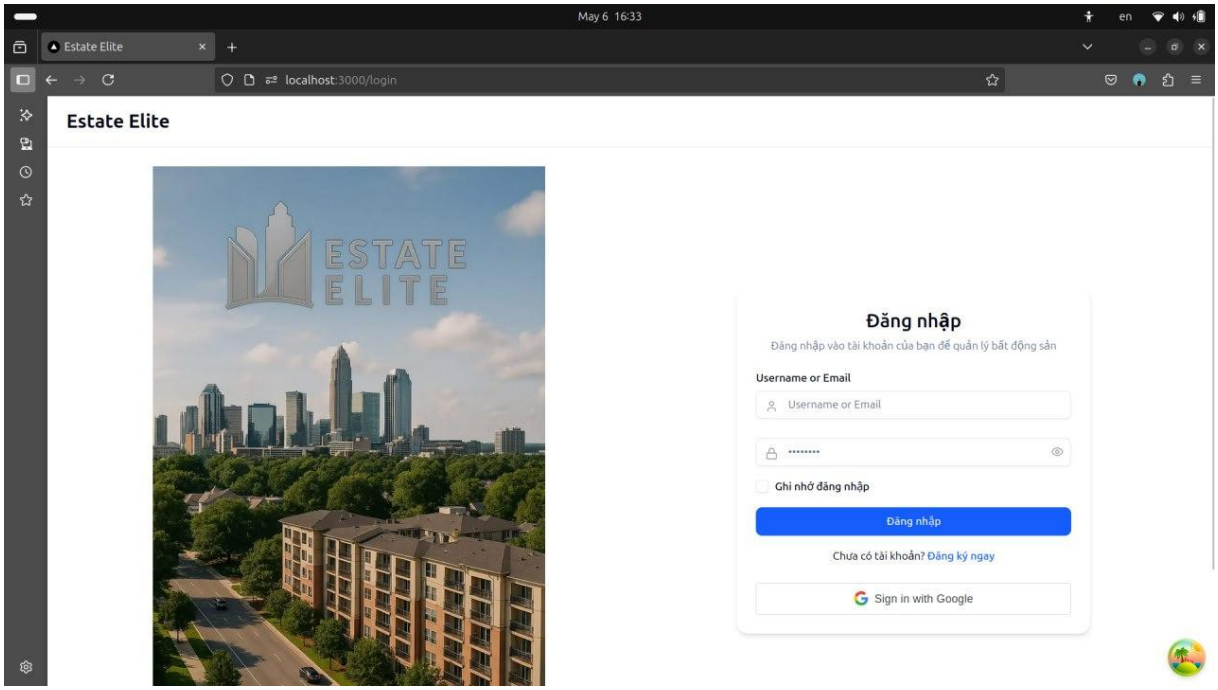
Trang đăng ký tài khoản cho phép người dùng có thể dễ dàng tạo tài khoản bằng cách nhập đầy đủ thông tin như họ tên, email, tên đăng nhập, mật khẩu và vai trò (ví dụ: khách hàng tìm mua/thuê). Giao diện được thiết kế hiện đại, thân thiện, giúp người dùng nhanh chóng truy cập và trải nghiệm dịch vụ môi giới bất động sản chuyên nghiệp.



Hình 3.6 Trang đăng ký

3.5.2. Trang đăng nhập

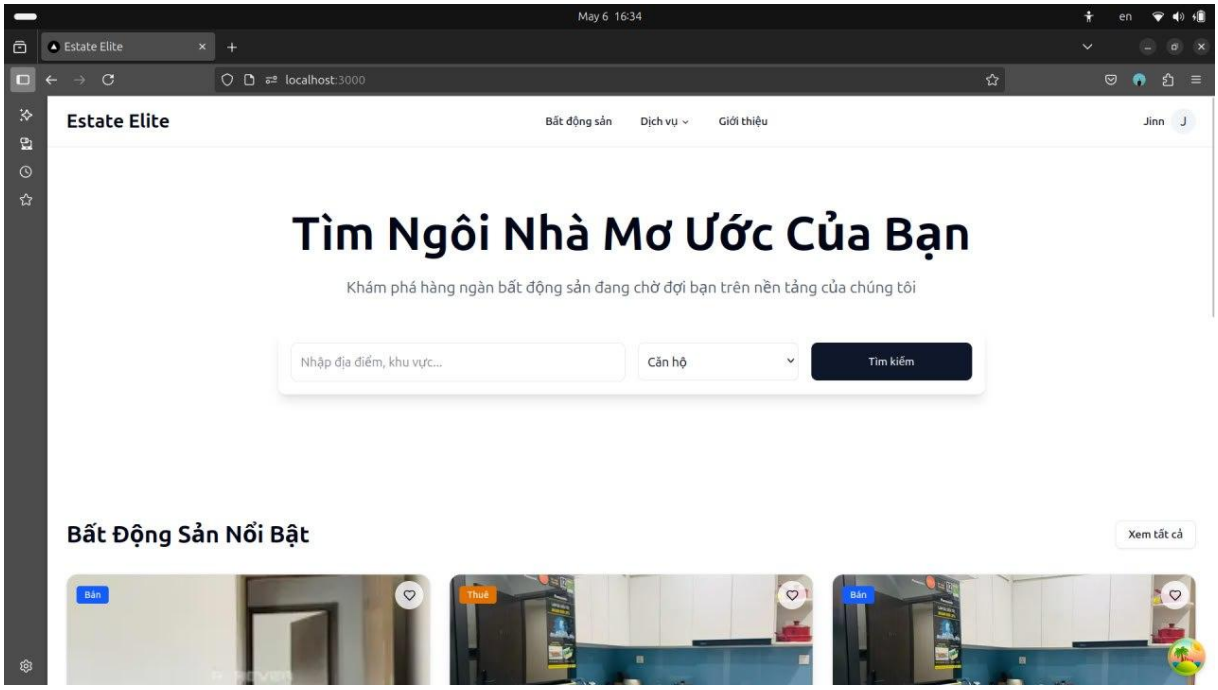
Trang đăng nhập cho phép người dùng có thể truy cập hệ thống bằng tài khoản đã đăng ký. Giao diện đơn giản, cho phép nhập tên người dùng hoặc email và mật khẩu, kèm tùy chọn "Ghi nhớ đăng nhập" giúp tăng trải nghiệm người dùng trong những lần truy cập sau. Ngoài ra còn hỗ trợ đăng nhập nhanh qua tài khoản Google.



Hình 3.7 Trang đăng nhập

3.5.3. Trang chủ của hệ thống

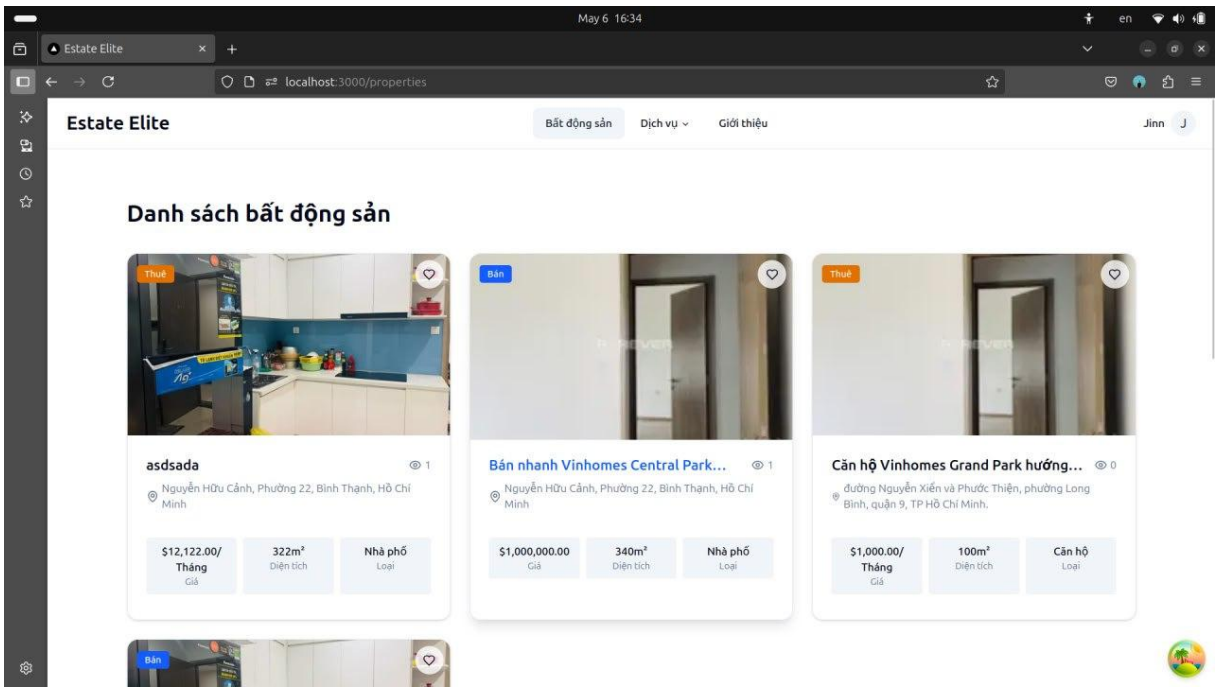
Trang chủ hệ thống cung cấp giao diện tìm kiếm bất động sản trực quan, cho phép người dùng nhập địa điểm, khu vực, chọn loại bất động sản và nhấn “Tìm kiếm” để khám phá hàng ngàn sản phẩm đang chờ giao dịch. Đồng thời hiển thị danh sách các bất động sản nổi bật ngay trên trang chủ.



Hình 3.8 Trang chủ hệ thống

3.5.4. Trang danh sách bất động sản

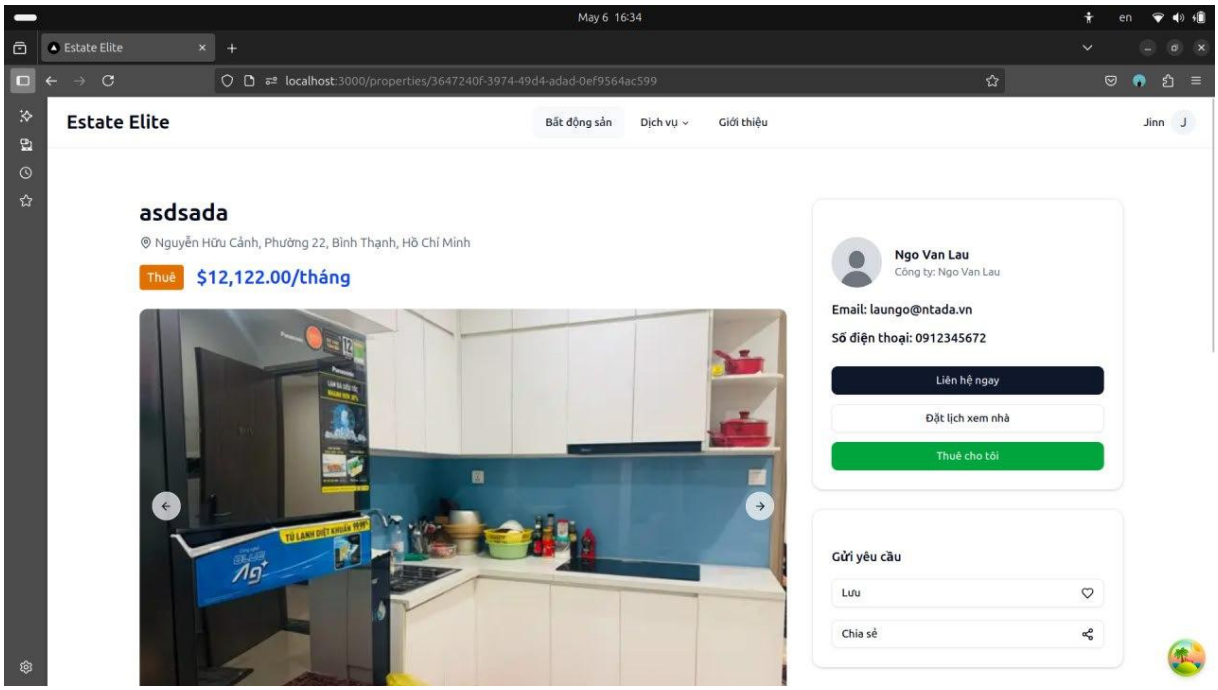
Trang danh sách bất động sản liệt kê danh sách bất động sản với mỗi bất động sản sẽ hiển thị đơn giản bao gồm địa chỉ, giá thuê/bán, hình ảnh minh họa. Người dùng có thể nhấn vào để xem chi tiết bất động sản.



Hình 3.9 Trang danh sách bất động sản

3.5.5. Trang chi tiết bất động sản

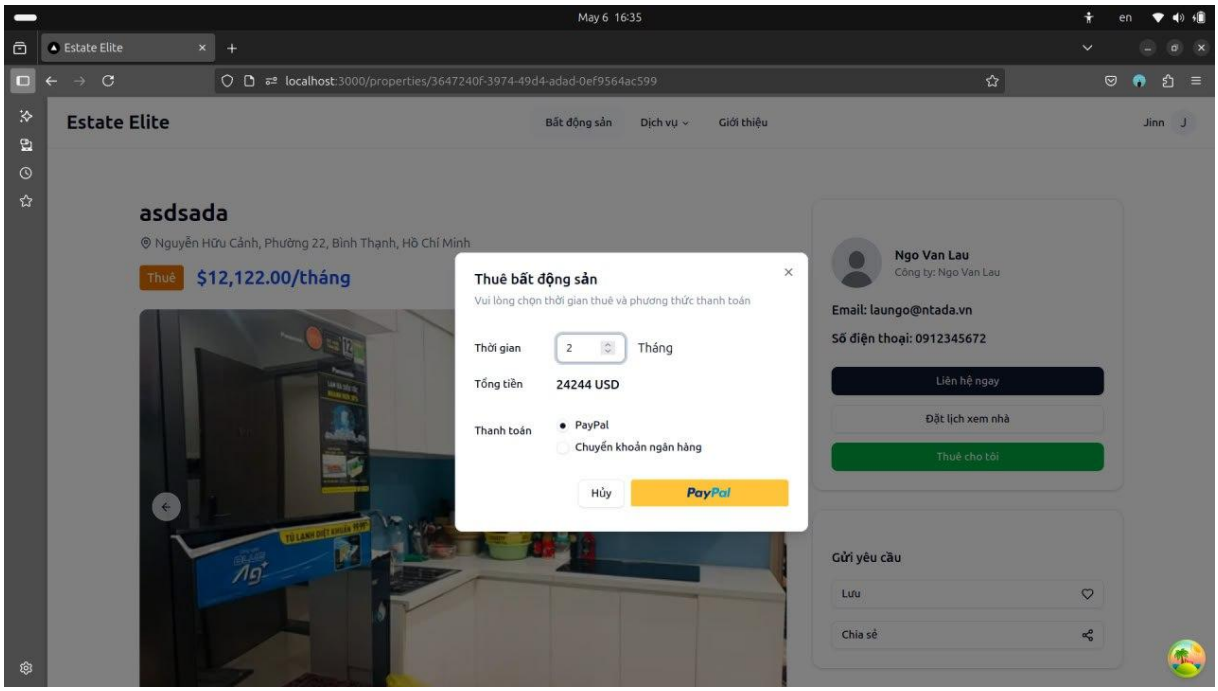
Trang chi tiết bất động sản hiển thị thông tin cụ thể về bất động sản đó như thông tin chủ sở hữu, địa chỉ, giá thuê/bán, hình ảnh minh họa. Ngoài ra người dùng có thể liên hệ với chủ sở hữu, đặt lịch xem nhà hoặc mua/thuê bất động sản. Đồng thời người dùng có thể lưu bất động sản lại để xem sau hoặc chia sẻ nó với mọi người.



Hình 3.10 Trang chi tiết bất động sản

3.5.6. Chức năng thuê và thanh toán

Chức năng thuê và thanh toán cho phép người dùng chọn thời gian thuê sau đó hệ thống tự động tính tiền. Đồng thời cung cấp hai hình thức thanh toán là Paypal hoặc chuyển khoản ngân hàng.

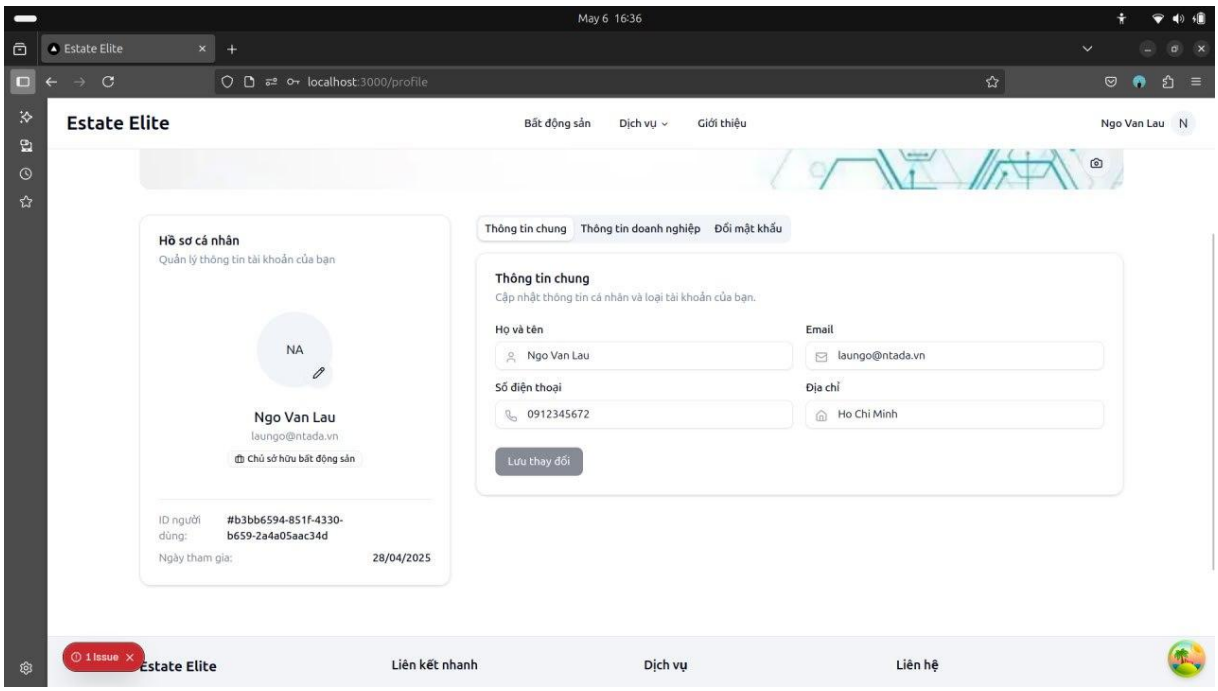


Hình 3.11 Chức năng thuê và thanh toán

3.5.7. Trang hồ sơ cá nhân

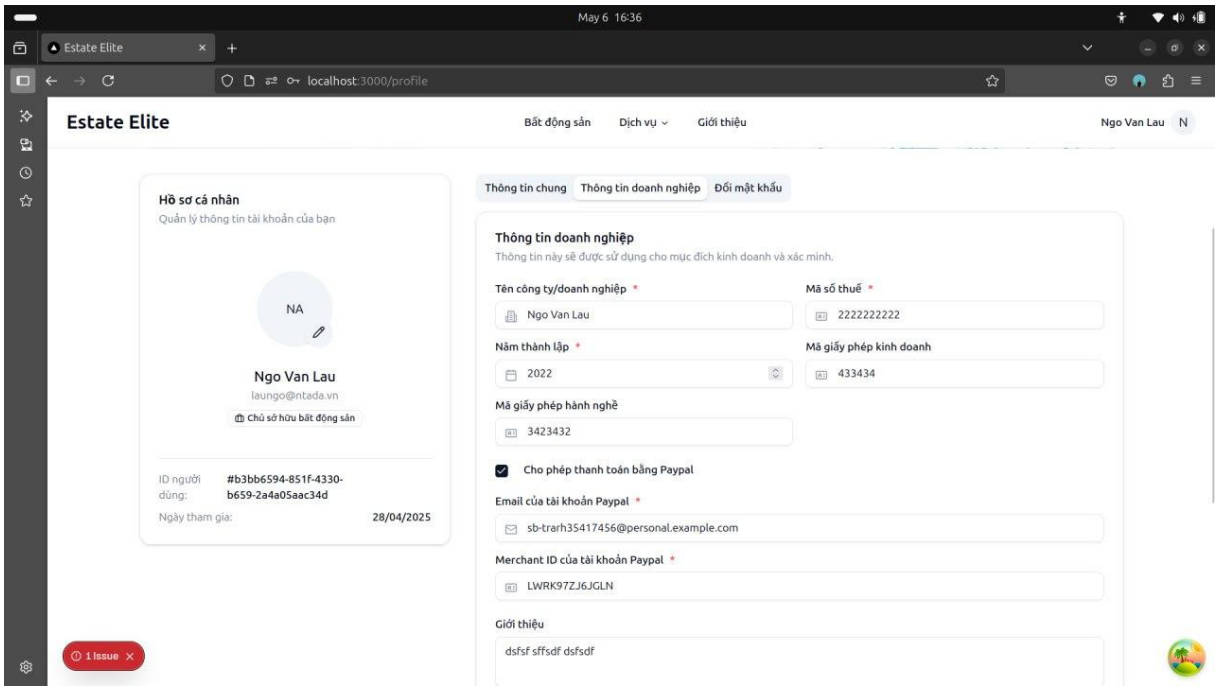
Trang hồ sơ gồm ba tab là thông tin chung, thông tin doanh nghiệp và đổi mật khẩu. Trong đó:

Tab thông tin chung cho phép người dùng quản lý thông tin cá nhân như họ tên, email, số điện thoại hoặc địa chỉ và người dùng có thể cập nhật.



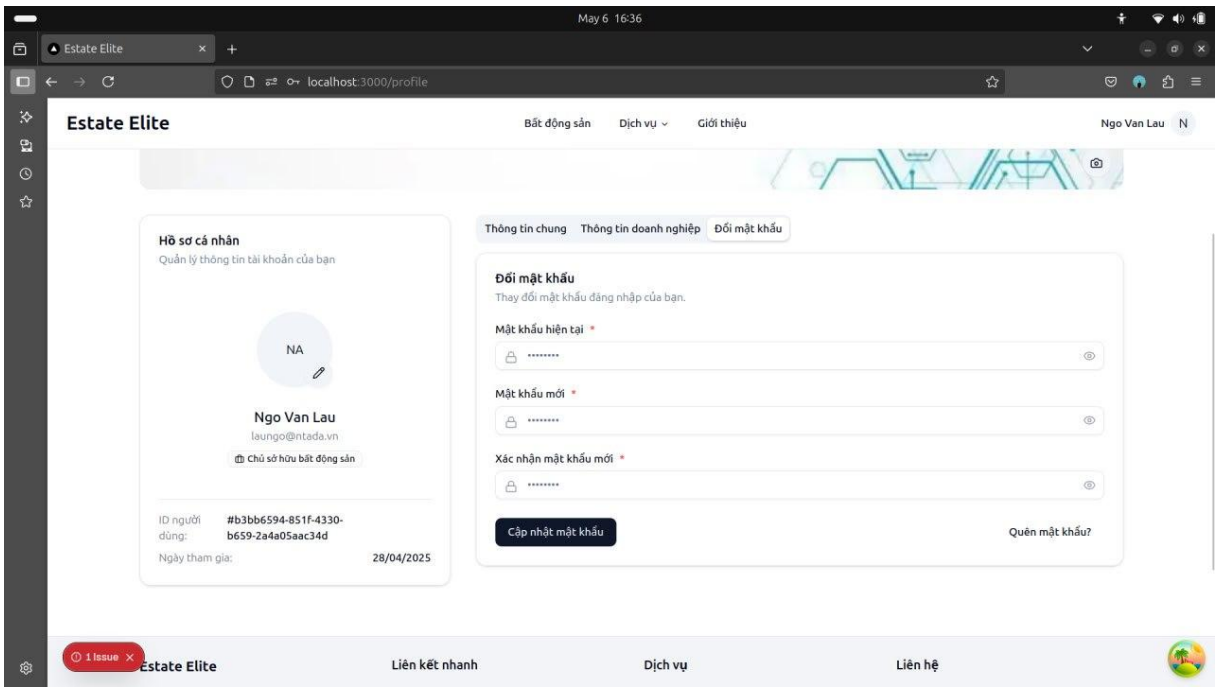
Hình 3.12 Tab thông tin chung

Tab thông tin doanh nghiệp cho phép chủ sở hữu bất động sản quản lý thông tin pháp lý, giấy phép doanh nghiệp và tích hợp tùy chọn thanh toán PayPal với thông tin Merchant ID và email PayPal.



Hình 3.13 Tab thông tin doanh nghiệp

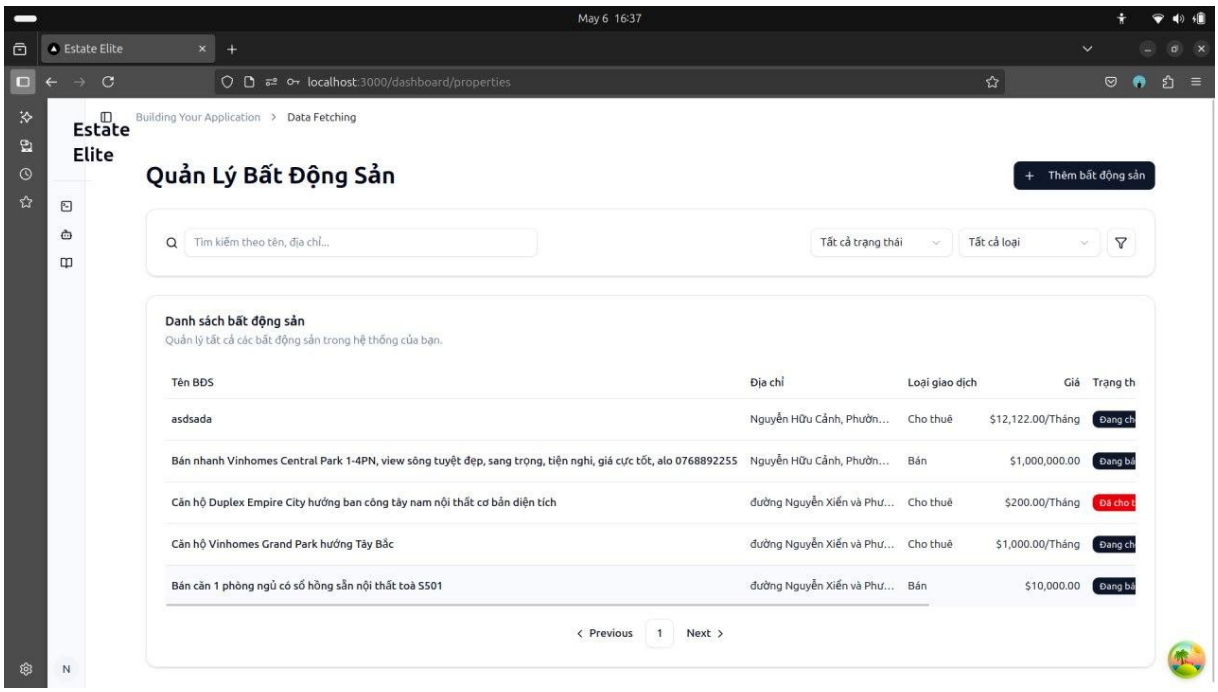
Tab đổi mật khẩu cho phép người dùng thay đổi mật khẩu bằng cách nhập mật khẩu cũ và nhập mật khẩu mới.



Hình 3.14 Tab đổi mật khẩu

3.5.8. Trang quản lý bất động sản

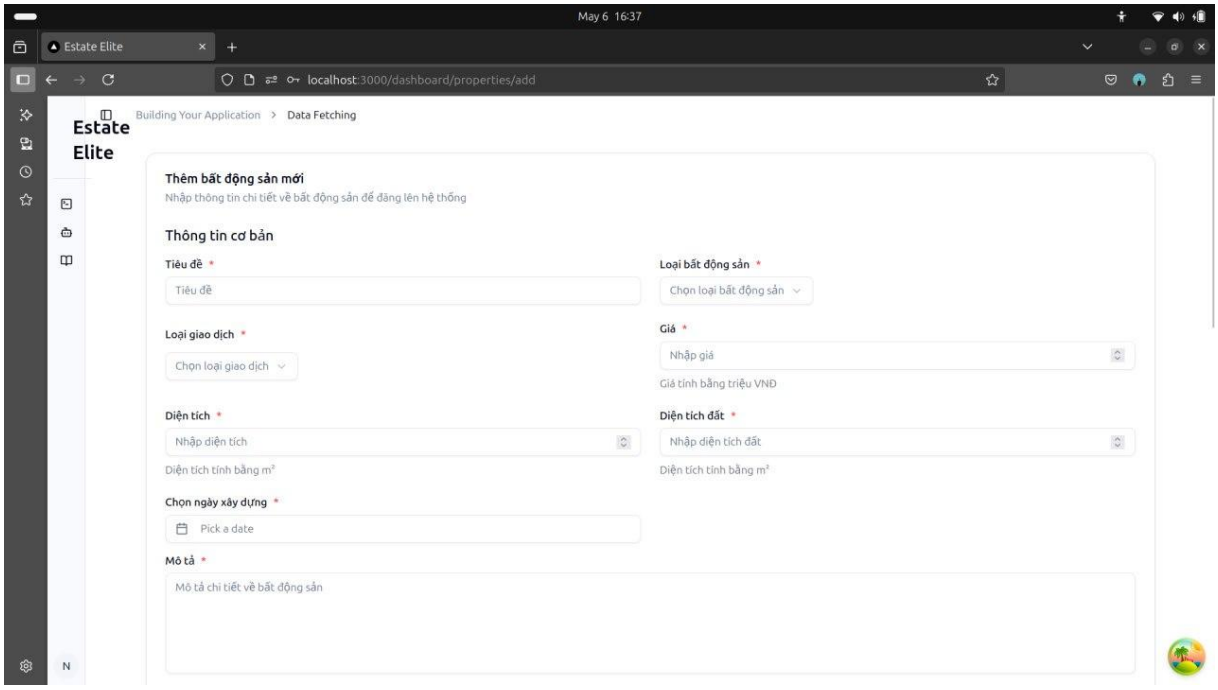
Trang quản lý bất động sản dành cho chủ sở hữu quản lý bất động sản của họ đã đăng tải. Trong đây hỗ trợ chức năng tìm kiếm nhanh chóng theo tên hoặc địa chỉ, lọc theo trạng thái hoặc loại giao dịch. Ngoài ra cung cấp các chức năng thêm, sửa hoặc xóa bất động sản.



Hình 3.15 Trang quản lý bất động sản

3.5.9. Chức năng thêm bất động sản

Chức năng dành cho phép chủ sở hữu bất động sản đăng tải thông tin bất động sản lên hệ thống. Thông tin bao gồm tiêu đề, loại giao dịch (bán, thuê), loại bất động sản (nhà phố, căn hộ,...), giá bán/thuê, diện tích nhà, diện tích đất, ngày xây dựng, mô tả chi tiết, …



Hình 3.16 Chức năng thêm bất động sản

Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Kết luận

Xuyên suốt quá trình thực hiện đề tài em đã vận dụng được các kiến thức, công nghệ đã học vào thực tế và không chỉ có vậy em còn học hỏi, tiếp thu và trau dồi nhiều kiến thức mới. Đầu tiên phải kể đến đó là em đã áp dụng được quy trình phát triển phần mềm vào cho việc thực hiện đề tài này. Thực hiện đầy đủ các giai đoạn bao gồm: Thu thập yêu cầu, phân tích và thiết kế hệ thống, lựa chọn công nghệ và cuối cùng là phát triển và triển khai.

Về khía cạnh kĩ thuật thì em đã áp dụng kiến trúc Microservices vào hệ thống giúp hệ thống chức năng được phân tách rõ ràng, các dịch vụ được độc lập phát triển mà không ảnh hưởng đến các dịch vụ khác từ đó hệ thống đạt được khả năng chịu tải tốt, dễ bảo trì, dễ mở rộng và phát triển lâu dài. Bên cạnh đó hệ thống em cũng tích hợp nhiều công nghệ mới và hiện đại. Phần back-end sử dụng ASP.NET Core và gRPC phù hợp cho việc phát triển kiến trúc Microservices đồng thời hệ thống đạt được hiệu năng cao và dễ dàng mở rộng. Phần front-end sử dụng Next.js cùng với Axios có thể triển khai được Server-side Rendering giúp tăng tốc độ tải trang và cải thiện SEO. Còn cơ sở dữ liệu sử dụng PostgreSQL và MinIO để lưu trữ media file. Ngoài ra, hệ thống còn tích hợp Message Broker sử dụng RabbitMQ để xử lý thông điệp bất đồng bộ giữa các service từ đó tăng hiệu năng và độ tin cậy, tích hợp Redis để cache dữ liệu tạm thời giúp tăng tốc độ phản hồi và tích hợp Docker để triển khai toàn bộ hệ thống để đảm bảo tính nhất quán trong các môi trường và dễ dàng phân phối.

Về mặt chức năng của hệ thống em đã triển khai được các chức năng như đăng kí, đăng nhập người dùng, quản lý thông tin cá nhân, đăng tin bất động sản với đầy đủ thông tin (hình ảnh, mô tả, trạng thái), tìm kiếm và lọc bất động sản theo các tiêu chí khác nhau và đặc biệt là tích hợp thanh toán trực tuyến bằng PayPal.

Tổng kết lại, qua quá trình thực hiện đề tài em đã áp dụng được các kiến thức, công nghệ đã học ở trường vào thực tế đồng thời được học hỏi và trau dồi các kiến thức công nghệ mới. Bên cạnh đó em còn rèn luyện được kỹ năng phân tích, thiết kế và triển khai hệ thống. Từ đó tạo nền tảng vững chắc cho sự phát triển của con đường sự nghiệp của em trong tương lai.

4.2. Hướng phát triển

Đề tài mặc dù đã hoàn thiện các chức năng cơ bản đáp ứng được các yêu cầu của một hệ thống quản lý và giao dịch bất động sản tuy nhiên để tiến xa hơn trong thương mại hóa, nâng cao chất lượng dịch vụ và mở rộng quy mô, đề tài cần được phát triển theo nhiều định hướng chiến lược quan trọng.

Đầu tiên, do sự phát triển mạnh mẽ và không ngừng bên cạnh những lợi ích của trí tuệ nhân tạo (AI) và máy học (Machine Learning) việc tích hợp vào hệ thống là định hướng hàng đầu. Hệ thống cần triển khai các chức năng như gợi ý bất động sản cá nhân hóa (dựa trên hành vi, lịch sử và vị trí), chức năng phát hiện gian lận, hành vi bất thường trong giao dịch.

Tiếp theo là phát triển ứng dụng di động đa nền tảng giúp mở rộng phạm vi tiếp cận, nâng cao trải nghiệm người dùng trong bối cảnh sự phổ biến của điện thoại thông minh. Các công nghệ có thể sử dụng là Flutter hoặc React Native.

Một trong những định hướng không thể thiếu là tăng cường bảo mật bằng việc tích hợp xác thực hai lớp (2FA) để tăng cường sự an toàn của tài khoản người dùng.

Cuối cùng là xây dựng các chức năng thống kê và báo cáo nâng cao giúp cho chủ sở hữu bất động sản kiểm soát và đưa ra các quyết định kinh doanh chiến lược.

Các định hướng phát triển trên sẽ giúp hệ thống hoàn thiện hơn về tính năng và trải nghiệm góp phần hoàn thiện hệ thống hiện tại. Đặc biệt là tạo nền tảng vững chắc để phát triển thành một sản phẩm thực sự tiềm năng trên thị trường bất động sản số trong và ngoài nước.

TÀI LIỆU THAM KHẢO

[1] Microsoft, “ASP.NET Core Documentation,” Microsoft Docs, 2024. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/.

[2] Vercel, “Next.js Documentation,” Next.js, 2024. [Online]. Available: https://nextjs.org/docs.

[3] PostgreSQL Global Development Group, “PostgreSQL Documentation,” PostgreSQL, 2024. [Online]. Available: https://www.postgresql.org/docs/.

[4] Redis Inc., “Redis Documentation,” Redis, 2024. [Online]. Available: https://redis.io/documentation.

[5] RabbitMQ, “RabbitMQ Documentation,” RabbitMQ, 2024. [Online]. Available: https://www.rabbitmq.com/documentation.html.

[6] gRPC Authors, “gRPC Documentation,” gRPC, 2024. [Online]. Available: https://grpc.io/docs/.

[7] Docker Inc., “Docker Documentation,” Docker, 2024. [Online]. Available: https://docs.docker.com/.

[8] PayPal, “PayPal Developer Documentation,” PayPal, 2024. [Online]. Available: https://developer.paypal.com/docs/.

[9] MinIO Inc., “MinIO Documentation,” MinIO, 2024. [Online]. Available: https://docs.min.io/.

[10] Nginx Inc., “Nginx Documentation,” Nginx, 2024. [Online]. Available: https://nginx.org/en/docs/.

[11] J. Skeet, *C# in Depth*, 4th ed. Manning Publications, 2019.

[12] J. Palermo, J. Bogard, and J. McCracken, *Entity Framework Core in Action*. Manning Publications, 2021.

[13] J. Smith, “Fluent Validation Documentation,” FluentValidation, 2024. [Online]. Available: https://fluentvalidation.net/.

[14] Serilog Contributors, “Serilog Documentation,” Serilog, 2024. [Online]. Available: https://serilog.net/.

PHỤ LỤC