**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**NGUYỄN TUẤN KIỆT**

**HỆ THỐNG BÁN NÔNG SẢN TRỰC TUYẾN SỬ DỤNG KIẾN TRÚC MICROSERVICE VỚI .NET CORE VÀ NEXTJS**

**ĐỒ ÁN NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**TP. HỒ CHÍ MINH, 2025BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**NGUYỄN TUẤN KIỆT**

**HỆ THỐNG BÁN NÔNG SẢN TRỰC TUYẾN SỬ DỤNG KIẾN TRÚC MICROSERVICE VỚI .NET CORE VÀ NEXTJS**

**Mã số sinh viên: 2151013046**

**ĐỒ ÁN NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Giảng viên hướng dẫn: TS. LÊ VIẾT TUẤN**

**TP. HỒ CHÍ MINH, 2025**

LỜI CẢM ƠN

Trước tiên, em xin được gửi lời cảm ơn chân thành đến TS Lê Viết Tuấn – Giảng viên hướng dẫn đồ án ngành của em, đã tận tình giúp đỡ em trong quá trình hoàn thiện đồ án ngành. Những hướng dẫn, góp ý và nhận xét của thầy đã giúp em có thể nhận ra được bản thân mình còn nhiều thiếu sót khi thực hiện đồ án. Đồng thời cũng giúp em có thêm được nhiều kiến thức để có thể áp dụng vào những dự án sau này.

Đồ án này không chỉ là kết quả sau một thời gian dài học tập mà còn là niềm đam mê với ngành Công Nghệ Thông Tin khi mà có thể sử dụng những công nghệ hiện đại để có thể mang đến những sản phẩm công nghệ đến tay nhiều người. Em tin rằng những trải nghiệm và kiến thức khi thực hiện đồ án sẽ là tiền đề, là bước đệm vững chắc cho sự nghiệp sau này khi đi làm trong tương lai.

Do kiến thức và thời gian có hạn, sẽ còn nhiều sai sót khi thực hiện đồ án. Em mong được đón nhận những lời góp ý, nhận xét chân thành của quý thầy cô, đặc biệt là giảng viên hướng dẫn của em, nhằm hoàn thiện hơn đồ án của mình.

Cuối cùng em xin gửi lời cảm ơn đến quý thầy cô, bạn bè đã luôn đồng hành cùng em trong suốt quá trình học tập tại trường Đại học Mở Thành phố Hồ Chí Minh.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TÓM TẮT ĐỐ ÁN NGÀNH

Trong thời đại công nghệ số hiện nay, việc mua hàng thông qua Internet trở nên dễ dàng hơn, đặc biệt là với những người làm nông, việc mua hàng thông qua Internet sẽ giúp họ có thể tiết kiệm nhiều thời gian hơn trong việc mua hàng, đồng thời cũng giúp họ có thể tiếp cận nhiều hơn tới những sản phẩm hay hàng hóa mới xuất hiện trên thị trường. Tuy vậy trên nền tảng Internet hiện nay, những Website bán hàng nông sản không nhiều, nếu có thì sẽ phải sử dụng những trang Web Landing Page đơn giản hay những trang Web sử dụng công nghệ cũ hay thông qua những trang thương mại điện tử như Shopee, Lazada… Điều này khiến nhiều cửa hàng khó có thể đưa hàng hóa của mình tiếp cận người dùng trên Internet, làm cho việc hàng hóa mới khó có thể đến tay người tiêu dùng.

Với đồ án bán hàng nông sản này, người dùng có thể được tiếp cận hàng hóa mới, tiếp cận với người dùng, dễ dàng truy cập để mua hàng mà không thông qua bên thứ ba. Website cũng được sử dụng những công nghệ mới nhất, giúp tăng hiệu suất và độ ổn định cho hệ thống.

Hệ thống sẽ gồm ba vai trò là khách hàng, người sẽ sử dụng hệ thống để mua hàng, nhân viên sẽ phụ trách quản lý kho hàng và đơn hàng của người mua hàng và quản trị viên sẽ là người quản lý và duy trì hệ thống hoạt động.

Kết quả đạt được sẽ là một hệ thống mua hàng hoàn chỉnh, được xây dựng dựa trên kiến trúc Microservice, giúp đảm bảo được hệ thống sẽ luôn hoạt động, dễ dàng triển khai với các tính năng như mua hàng, cho hàng vào giỏ, lấy hàng trong giỏ để thanh toán, đăng ký và đăng nhập, tìm kiếm sản phẩm theo từ khóa. Điều này giúp hỗ trợ rất nhiều trong việc mua hàng của người tiêu dùng, đặc biệt là những người làm nông, giúp nhà nông có thể dễ dàng mua hàng và tiếp cận được nhiều hàng hóa mới.

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc197460137)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc197460138)

[TÓM TẮT ĐỐ ÁN NGÀNH 3](#_Toc197460139)

[DANH MỤC TỪ VIẾT TẮT 7](#_Toc197460144)

[DANH MỤC HÌNH VẼ 8](#_Toc197460157)

[DANH MỤC BẢNG 10](#_Toc197460158)

[MỞ ĐẦU 11](#_Toc197460159)

[Chương 1. TỔNG QUAN ĐỀ TÀI 12](#_Toc197460160)

[1.1. Giới thiệu đề tài 12](#_Toc197460161)

[1.2. Lý do chọn đề tài 12](#_Toc197460162)

[1.3. Mục tiêu và phạm vi đề tài 12](#_Toc197460163)

[1.4. Phương pháp nghiên cứu 13](#_Toc197460164)

[1.4.1. Khảo sát nghiệp vụ 13](#_Toc197460165)

[1.4.2. Lựa chọn công nghệ 13](#_Toc197460166)

[1.5. Bố cục báo cáo 13](#_Toc197460167)

[Chương 2. CƠ SỞ LÝ THUYẾT 15](#_Toc197460168)

[2.1. Giới thiệu tổng quan về Microservice 15](#_Toc197460169)

[2.1.1. Đặc điểm chính của Microservice 15](#_Toc197460170)

[2.1.2. Kiến trúc hệ thống 16](#_Toc197460171)

[2.2. Giới thiệu về ASP.NET Core 17](#_Toc197460172)

[2.2.1. Đặc điểm chính của ASP.NET Core 17](#_Toc197460173)

[2.2.2. Kiến trúc ASP.NET Core 18](#_Toc197460174)

[2.2.3. Cách sử dụng 19](#_Toc197460175)

[2.3. Giới thiệu về Next.js 24](#_Toc197460176)

[2.3.1. Đặc điểm chính của Next.js 24](#_Toc197460177)

[2.3.2. Kiến trúc NextJS 26](#_Toc197460178)

[2.3.3. Cách sử dụng 26](#_Toc197460179)

[2.4. Các công cụ và dịch vụ 29](#_Toc197460180)

[2.4.1. PostgreSQL 29](#_Toc197460181)

[2.4.2. MongoDB 29](#_Toc197460182)

[2.4.3. Redis 29](#_Toc197460183)

[Chương 3. PHÁT TRIỂN HỆ THỐNG BÁN NÔNG SẢN SỬ DỤNG KIẾN TRÚC MICROSERVICE 30](#_Toc197460184)

[3.1. Giới thiệu hệ thống 30](#_Toc197460185)

[3.2. Phân tích hệ thống 30](#_Toc197460186)

[3.2.1. Lược đồ Use Case 30](#_Toc197460187)

[3.2.2. Đặc tả Use Case 31](#_Toc197460188)

[3.3. Thiết kế cơ sở dữ liệu 35](#_Toc197460189)

[3.3.1. UserService 35](#_Toc197460190)

[3.3.2. CartService 35](#_Toc197460191)

[3.3.3. ProductService 35](#_Toc197460192)

[3.3.4. PaymentService 36](#_Toc197460193)

[3.3.5. Mối quan hệ giữa các bảng 36](#_Toc197460194)

[3.4. Kiến trúc hệ thống 37](#_Toc197460195)

[3.4.1. Sơ đồ tuần tự (Sequence Diagram) 37](#_Toc197460196)

[3.4.2. Sơ đồ hoạt động (Activity Diagram) 40](#_Toc197460197)

[3.5. Kiến trúc hệ thống 45](#_Toc197460198)

[3.5.1. User Service 45](#_Toc197460199)

[3.5.2. Cart Service 45](#_Toc197460200)

[3.5.3. Product Service 45](#_Toc197460201)

[3.5.4. Order Service 46](#_Toc197460202)

[3.5.5. Công cụ hỗ trợ 46](#_Toc197460203)

[3.6. Kết quả thực hiện 47](#_Toc197460204)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 51](#_Toc197460205)

[4.1. Kết luận 51](#_Toc197460206)

[4.1.1. Tổng quát 51](#_Toc197460207)

[4.1.2. Về ưu và nhược điểm 51](#_Toc197460208)

[4.1.3. Hạn chế 51](#_Toc197460209)

[4.2. Hướng phát triển 52](#_Toc197460210)

[TÀI LIỆU THAM KHẢO 53](#_Toc197460211)

DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| SQL | Structre Query Language |
| SSR | Server-Side Rendering |
| SSG | Static Side Generation |
| ASP | Active Server Page |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |

DANH MỤC HÌNH VẼ

[Hình 2.1: Hệ thống sử dụng kiến trúc Microservice. 16](#_Toc197460035)

[Hình 2.2: Kiến trúc Clean Architecture của ASP.NET Core 18](#_Toc197460036)

[Hình 2.3: Luồng đi của ASP.NET Core 19](#_Toc197460037)

[Hình 2.4: Giao diện Visual Studio Installer 20](#_Toc197460038)

[Hình 2.5: Giao diện đặt tên và vị trí dự án trong máy tính 20](#_Toc197460039)

[Hình 2.6: Giao diện thêm thông tin cho dự án 21](#_Toc197460040)

[Hình 2.7: Cấu trúc dự án ASP.NET Core 22](#_Toc197460041)

[Hình 2.8: Controller ASP.NET Core 23](#_Toc197460042)

[Hình 2.9: Chương trình ASP.NET Core đầu tiên 23](#_Toc197460043)

[Hình 2.10: NextJS Routing 24](#_Toc197460044)

[Hình 2.11: Route Handles của NextJS 25](#_Toc197460045)

[Hình 2.12: Kiến trúc ứng dụng của NextJS 26](#_Toc197460046)

[Hình 2.13: Folder Next.js. 27](#_Toc197460047)

[Hình 2.14: Tạo dự án Next.js 27](#_Toc197460048)

[Hình 2.15: Lựa chọn dự án Next.js 27](#_Toc197460049)

[Hình 2.16: Cấu trúc thư mục của Next.js 28](#_Toc197460050)

[Hình 2.17: Ứng dụng Next.js đầu tiên 28](#_Toc197460051)

[Hình 3.1: Lược đồ Use Case 31](#_Toc197460052)

[Hình 3.2: Lược đồ cơ sở dữ liệu 35](#_Toc197460053)

[Hình 3.3: Sơ đồ tuần tự tìm kiếm sản phẩm 37](#_Toc197460054)

[Hình 3.4: Sơ đồ tuần tự đặt hàng 38](#_Toc197460055)

[Hình 3.5: Sơ đồ tuần tự quản lý kho hàng 39](#_Toc197460056)

[Hình 3.6: Sơ đồ tuần tự quản lý sản phẩm 40](#_Toc197460057)

[Hình 3.7: Sơ đồ hoạt động cho chức năng quên mật khẩu 41](#_Toc197460058)

[Hình 3.8: Sơ đồ hoạt động cho chức năng mua hàng 42](#_Toc197460059)

[Hình 3.9: Sơ đồ hoạt động cho chức năng quản lý kho hàng 43](#_Toc197460060)

[Hình 3.10: Sơ đồ hoạt động quản lý hàng hóa 44](#_Toc197460061)

[Hình 3.11: Kiến trúc hệ thống 45](#_Toc197460062)

[Hình 3.12: Trang chủ website 47](#_Toc197460063)

[Hình 3.13: Trang sản phẩm 47](#_Toc197460064)

[Hình 3.14: Trang chứa danh mục 48](#_Toc197460065)

[Hình 3.15: Component Cart 48](#_Toc197460066)

[Hình 3.16: Trang hiển thị sản phẩm khi tìm kiếm theo từ khóa 49](#_Toc197460067)

[Hình 3.17: Trang thông tin sản phẩm 49](#_Toc197460068)

[Hình 3.18: Trang và Modal hiển thị thông tin 49](#_Toc197460069)

[Hình 3.19: Trang hiển thị form đăng nhập 50](#_Toc197460070)

[Hình 3.20: Trang hiển thị form đăng ký 50](#_Toc197460071)

DANH MỤC BẢNG

[Bảng 3.1: UseCase mua hàng 31](#_Toc194171857)

[Bảng 3.2: UseCase Thanh toán 32](#_Toc194171858)

[Bảng 3.3: UseCase quản lý kho hàng 33](#_Toc194171859)

[Bảng 3.4: UseCase quản lý sản phẩm 34](#_Toc194171860)

MỞ ĐẦU

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Trong thời đại công nghệ đang phát triển hiện nay, việc sử dụng công nghệ và Internet đã giúp cho ngành nông nghiệp có thể phục vụ được cho nhiều người hơn trong cuộc sống, đặc biệt với hệ thống thương mại điện tử (E-Commerce) đã giúp cho việc kết nối giữa nhà sản xuất và người tiêu dùng trở nên dễ dàng hơn. Tuy nhiên, thị trường nông sản hiện nay vẫn còn gặp nhiều vấn đề hạn chế có thể xảy đến như chuỗi cung ứng phức tạp, giá cả biến động hay tình trạng hàng hóa chưa được đảm bảo chất lượng…

Trước những thực trạng như vậy, việc xây dựng một nền tảng E-Commerce dành riêng cho nông sản không chỉ giúp cho nhà cung cấp tối ưu được quá trình phân phối và bán hàng mà còn đảm bảo được chất lượng sản phẩm, giúp cho người mua, đặc biệt là nhà nông có thể tiếp cận được thị trường rộng lớn và hàng hóa đa dạng hơn, không còn chỉ còn là những sản phẩm quen thuộc mà còn là những sản phẩm của những nhà cung cấp mới, giúp người tiêu dung có thể mua được sản phẩm với giá cả và chất lượng tốt nhất.

## Lý do chọn đề tài

Việc phát triển một cửa hàng trực tuyến mua nông sản giúp người dùng có thể dễ dàng mua hàng hóa thông qua nền tảng Website, giúp cho việc tiết kiệm thời gian trong việc mua và lựa chọn hàng hóa. Kèm với đó ở phía cửa hàng có thể giúp họ đưa sản phẩm đến gần hơn với người mua, đặc biệt là nhà nông khi họ có quá nhiều lựa chọn sản phẩm hay những sản phẩm mới trên thị trường.

## Mục tiêu và phạm vi đề tài

Website bán hàng nông sản giúp giải quyết về nhu cầu mua hàng hóa nông sản của nhà nông trong quá trình làm nông. Người làm nông có thể mua hàng hóa, tìm kiếm sản phẩm và được giới thiệu sản phẩm mới giúp họ có thể tiếp cận nhiều hơn tới sản phẩm mới nhất trên thị trường. Đối với người bán hàng thì sẽ giúp họ có thể tiếp cận nhiều tệp khách hàng hơn, giúp tăng doanh thu và số lượng bán ra. Sản phẩm sẽ được quản lý dễ dàng hơn bởi hệ thống.

## Phương pháp nghiên cứu

### Khảo sát nghiệp vụ

Để việc phát triển hệ thống bán nông sản, việc khảo sát và hiểu rõ những yêu cầu cũng như nghiệp vụ của hệ thống được đảm bảo hoạt động một cách hiệu quả. Bước đầu tiên là khảo sát trên các nền tảng mua hàng như các Website bán hàng trực tuyến như shopnongnghiep.com, nongnghieppho.vn hay vuonsaigon.vn, hoặc các gian hàng trên các sàn thương mại điện tử như Shopee, Lazada, Tiki để tìm hiểu cách các hệ thống này hoạt động cũng như nghiệp vụ giúp người mua hàng có thể dễ dàng chọn và mua hàng hóa, người bán có thể quản lý sản phẩm và hàng hóa. Kết quả của cuộc khảo sát cho thấy được những nghiệp vụ cốt lõi cần thiết để có thể tạo và quản lý hệ thống bao gồm có quản lý người dùng, quản lý sản phẩm, hàng hóa, quản lý đơn hàng và tích hợp các phương thức thanh toán phổ biến hiện nay. Những nghiệp vụ mà khảo sát mang lại chính là cốt lõi để có thể giúp cho việc phát triển và triển khai hệ thống trở nên dễ dàng và hiệu quả hơn.

### Lựa chọn công nghệ

ASP.NET Core: Triển khai theo kiến trúc Microservice giúp đáp ứng được khả năng nâng cấp, mở rộng và phát triển hệ thống.

NextJS: Hiển thị giao diện người dùng, linh hoạt và hiệu suất cao.

PostgreSQL: Lưu trữ dữ liệu có cấu trúc, đảm bảo tính nhất quán và ổn định của dữ liệu.

MongoDB: Lưu trữ dữ liệu phi cấu trúc, đảm bảo tính linh hoạt và truy vấn nhanh chóng.

Redis: Lưu trữ dữ liệu phi cấu trúc một cách nhanh chóng, giúp lưu dữ liệu tạm thời.

## Bố cục báo cáo

Chương 1: Giới thiệu tổng quan về đề tài, trình bày ngắn gọn về mục đích, lý do, mục tiêu và phương pháp nghiên cứu đề tài. Chương đầu tiên này sẽ tập trung vào việc giới thiệu qua về đề tài, dựa trên bối cảnh và nhu cầu thực tiễn trong việc mua hàng, đồng thời cũng giới thiệu qua những công nghệ cần thiết để phát triển hệ thống.

Chương 2: Cơ sở lý thuyết của hệ thống. Chương thứ hai sẽ giới thiệu rõ hơn về công nghệ được áp dụng trong đề tài, cụ thể với Backend, ASP.NET Core sẽ được lựa chọn để phát triển với kiến trúc Microservice và với Frontend thì Next.js sẽ là công nghệ được lựa chọn. Các cơ sở dữ liệu được sử dụng trong hệ thống bao gồm PostgreSQL để quản lý dữ liệu có tính nhất quán như thông tin người dùng hay hóa đơn, MongoDB và Redis sẽ quản lý cơ sở dữ liệu có tính linh hoạt, với MongoDB sẽ quản lý sản phẩm và hàng hóa, với Redis sẽ lưu dữ liệu nhanh như giỏ hàng.

Chương 3: Phát triển hệ thống bán nông sản sử dụng kiến trúc Microservice. Ở chương này sẽ đi sâu vào việc phát triển hệ thống, bao gồm kiến trúc hệ thống, các sơ đồ, luồng hoạt động của hệ thống và những kết quả đạt được sau khi áp dụng công nghệ vào hệ thống.

Chương 4: Kết luận và hướng phát triển. Với chương này sẽ đưa ra kết luận sau khi nghiên cứu, sử dụng và áp dụng các công nghệ được giới thiệu và nghiệp vụ, sơ đồ được triển khai để làm cơ sở đưa ra hướng phát triển của hệ thống trong tương lai.

# CƠ SỞ LÝ THUYẾT

## Giới thiệu tổng quan về Microservice

Microservice là một kiến trúc phần mềm, cho phép nhà phát triển có thể chia nhỏ hệ thống thành các dịch vụ (Service) tách biệt và độc lập, không cần phụ thuộc lẫn nhau mà vẫn đảm bảo việc kết nối hệ thống và hoạt động một cách chặt chẽ khi cần thiết.

Microservice ra đời nhằm để khắc phục những nhược điểm cố hữu mà kiến trúc nguyên khối (Monolithic) đang còn gặp phải như khó có thể mở rộng, khó bảo trì và thêm tính năng, không thể linh hoạt trong việc sửa đổi và phát triển các chức năng mới và nếu có xảy ra lỗi hệ thống thì có thể làm cho toàn bộ chương trình buộc phải dừng hoạt động.

Microservice giúp cho việc phát triển hệ thống trở nên dễ dàng thông qua việc phát triển theo từng module riêng lẻ, có thể tận dụng được nhiều cơ sở dữ liệu, các chức năng được chia nhỏ giúp cho việc phát triển trở nên độc lập, giúp linh hoạt trong việc phát triển và triển khai hệ thống một cách độc lập.

### Đặc điểm chính của Microservice

#### Độc lập

Các chức năng trong hệ thống được phát triển thành các dịch vụ riêng, cho phép phát triển hệ thống một cách dễ dàng, giúp việc phát triển hệ thống song song và tập trung vào một chức năng cụ thể trong dịch vụ.

#### Chịu lỗi

Các chức năng được phát triển và hoạt động một cách độc lập, khi có một dịch vụ gặp sự cố hay bị lỗi, các dịch vụ khác vẫn có thể hoạt động bình thường, giúp cho việc hệ thống hoạt động liên tục mà không phải dừng lại vì gặp sự cố hệ thống.

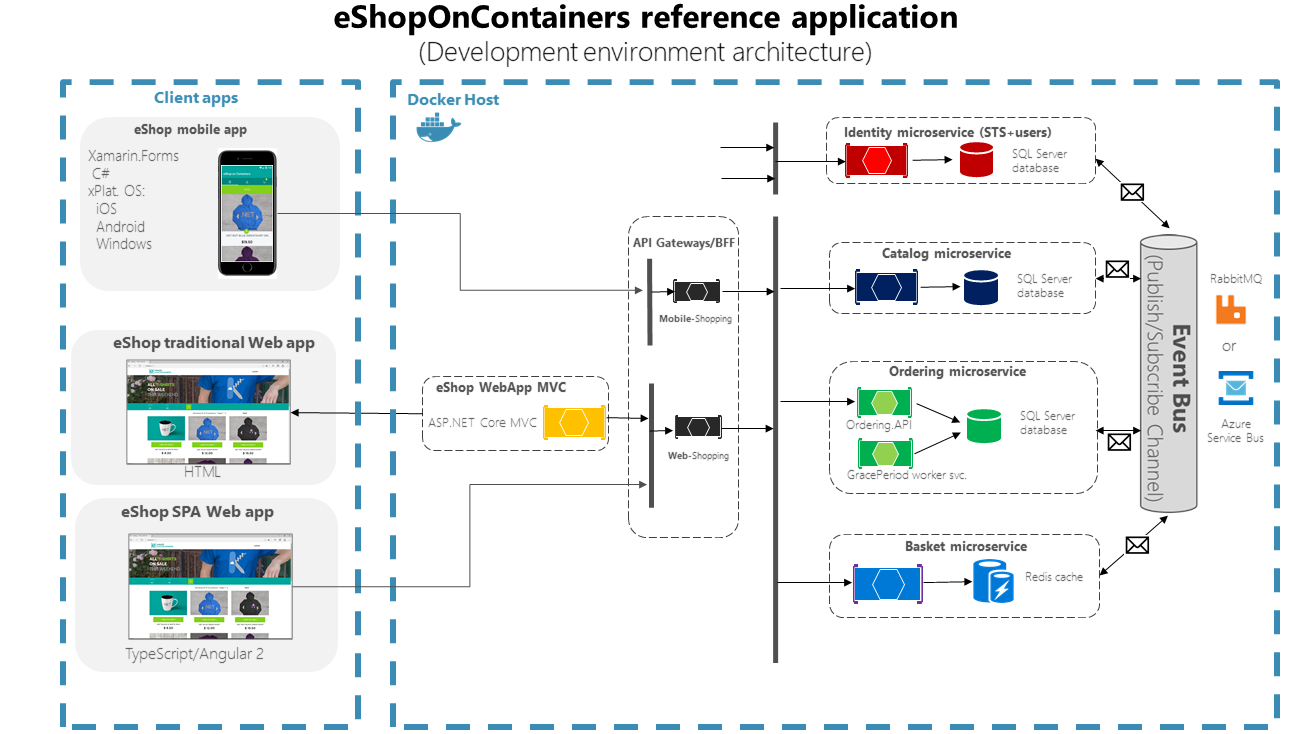
#### Mở rộng

Các dịch vụ trong hệ thống hoàn toàn có thể mở rộng mà không phải lo sẽ đụng vào các dịch vụ khác, giúp cải thiện việc phát triển phần mềm.

#### Linh hoạt

Các dịch vụ có thể được phát triển dựa trên nhiều công nghệ và công cụ khác nhau, không còn phụ thuộc vào một công nghệ được sử dụng trong hệ thống, giúp tận dụng tối đa sức mạnh và chức năng của công nghệ đó.

### Kiến trúc hệ thống



Hình .: Hệ thống sử dụng kiến trúc Microservice. Nguồn: “<https://learn.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/implement-api-gateways-with-ocelot>”

Hệ thống sử dụng kiến trúc Microservice sẽ bao gồm các Service khác nhau, hoạt động linh hoạt, các Service sẽ đảm nhận một chức năng cụ thể khác nhau, đồng thời sử dụng các cơ sử dữ liệu khác nhau để có thể lưu trữ dữ liệu linh hoạt. Các Service sẽ giao tiếp với nhau thông qua một số giao thức như Rest API, gRPC…

API Gateway là một cổng trung gian cho phép Client khi gửi request đến Service thì API Gateway sẽ tự động điều hướng đến Service được gửi request. Ngoài ra API Gateway cũng được tích hợp thêm bảo mật và cân bằng tải để giảm tải khi có quá nhiều request được gửi đến.

## Giới thiệu về ASP.NET Core

ASP.NET Core là một Framework mã nguồn mở và đa nền tảng được phát triển bởi Microsoft sử dụng ngôn ngữ C#. ASP.NET Core cho phép nhà phát triển có thể sử dụng để phát triển hệ thống khi cung cấp các công cụ giúp phát triển hệ thống một cách mạnh mẽ, đặc biệt là những hệ thống sử dụng kiến trúc Microservice.

### Đặc điểm chính của ASP.NET Core

#### High-performance

ASP.NET Core được thiết kế để cung cấp hiệu suất mạnh mẽ và tối ưu. Với việc sử dụng Web Server hiệu suất cao như Kestrel, ASP.NET Core có thể xử lý được lượng lớn những request với tốc độ nhanh chóng. Ngoài ra ASP.NET Core cũng hỗ trợ việc xử lý bất đồng bộ (asynchronous) giúp tận dụng tối đa tài nguyên hệ thống.

#### Dependency Injection (DI)

ASP.NET Core hỗ trợ Dependency Injection (DI) ngay từ ban đầu, giúp quản lý và tái sử dụng hệ thống một cách hiệu quả. Việc sử dụng DI giúp code dễ bảo trì và giảm sự phụ thuộc giữa các module trong ứng dụng.

#### Cross-platform

ASP.NET Core có thể chạy trên hiều hệ điều hành như Windows, MacOS và Linux. Điều này giúp cho lập trình viên có thể triển khai ứng dụng trên nhiều môi trường khác nhau mà không bị ràng buộc vào một hệ điều hành Windows như ASP.NET.

#### Flexible

ASP.NET Core sử dụng kiến trúc Middleware cho phép tùy chỉnh pipeline và xử lý yêu cầu HTTP bằng cách thêm, xóa hoặc thay đổi thứ tự các Middleware linh hoạt, giúp tối ưu hóa quy trình xử lý request và response của ứng dụng.

#### Cloud and Microservice

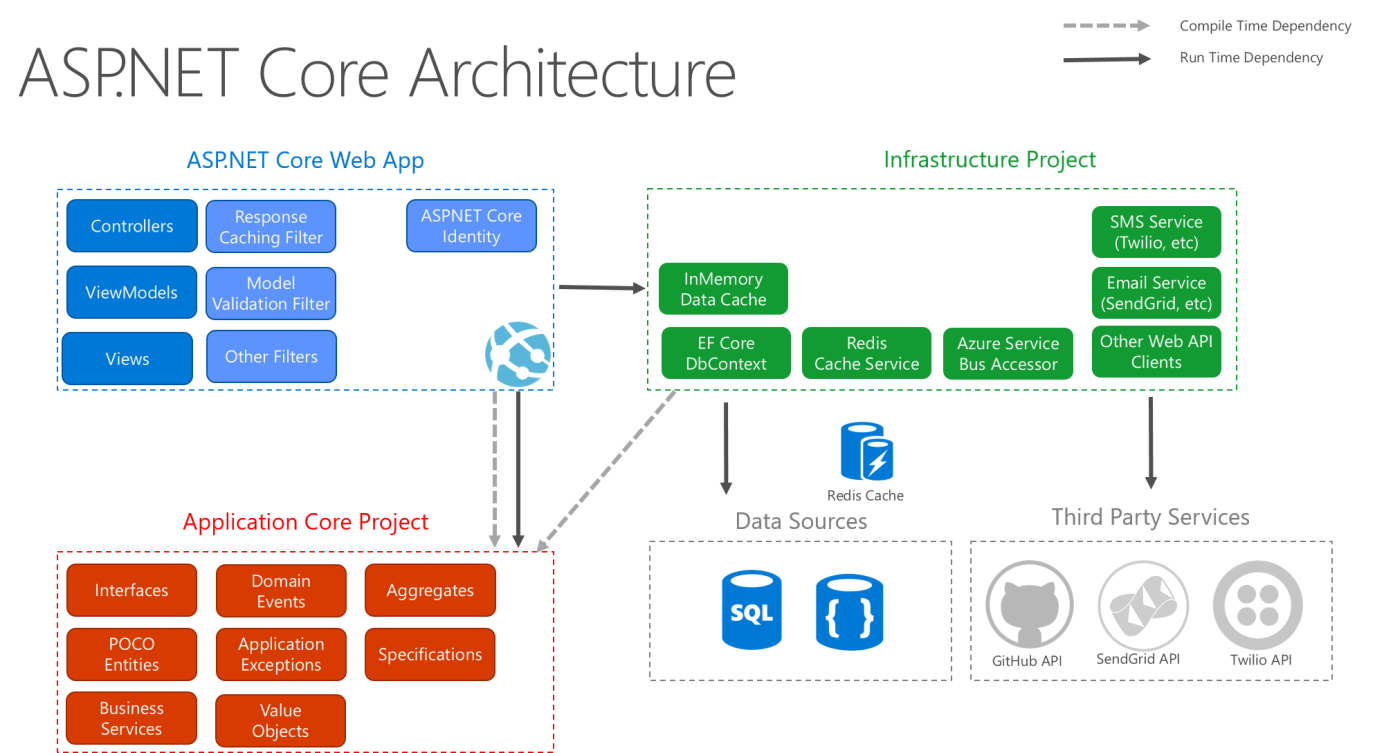
ASP.NET Core được thiết kế để triển khai trên Cloud như Azure và phù hợp với kiến trúc phần mềm Microservice. Với các công cụ như Rest API, Ocelot (API Gateway), Docker,… lập trình viên có thể dễ dàng xây dựng và phát triển phần mềm.

#### Production – ready

ASP.NET Core cung cấp bộ công cụ hỗ trợ mạnh mẽ cho việc đưa ứng dụng ra môi trường Production như:

* Serilog giúp quản lý logging một cách mạnh mẽ.
* Sử dụng Prometheus, Granfa để giám sát hệ thống.
* Hỗ trợ CI/CD để có thể tự động hóa quy trình triển khai ứng dụng.

### Kiến trúc ASP.NET Core

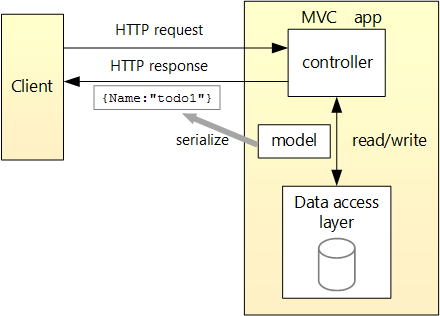


Hình .: Kiến trúc Clean Architecture của ASP.NET Core. Nguồn: “<https://learn.microsoft.com/vi-vn/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0>”

ASP.NET Core Web App (Lớp Presentation): Chứa các thành phần đại diện xử lý yêu cầu HTTP với các thành phần gồm Controllers để xử lý yêu cầu request, ViewModels giúp định dạnh dữ liệu hiển thị, Identity quản lý xác thực và phân quyền...

Application Core Project (Lớp Business Logic): Chứa các logic và quy tắc nghiệp vụ với các thành phần gồm Interfaces: Định nghĩa các interface cho dịch vụ, Specifications xử lý quy tắc nghiệp vụ, Business Service quản lý dịch vụ nghiệp vụ…

Infrastructure Project (Lớp Data Access): Chứa các thành phần giao tiếp với cơ sở dữ liệu, cache và dịch vụ bên thứ ba với các thành phần gồm EF Core DbContext giúp truy cập cơ sở dữ liệu, Azure Service Bus Accessor giao tiếp với Service bus…

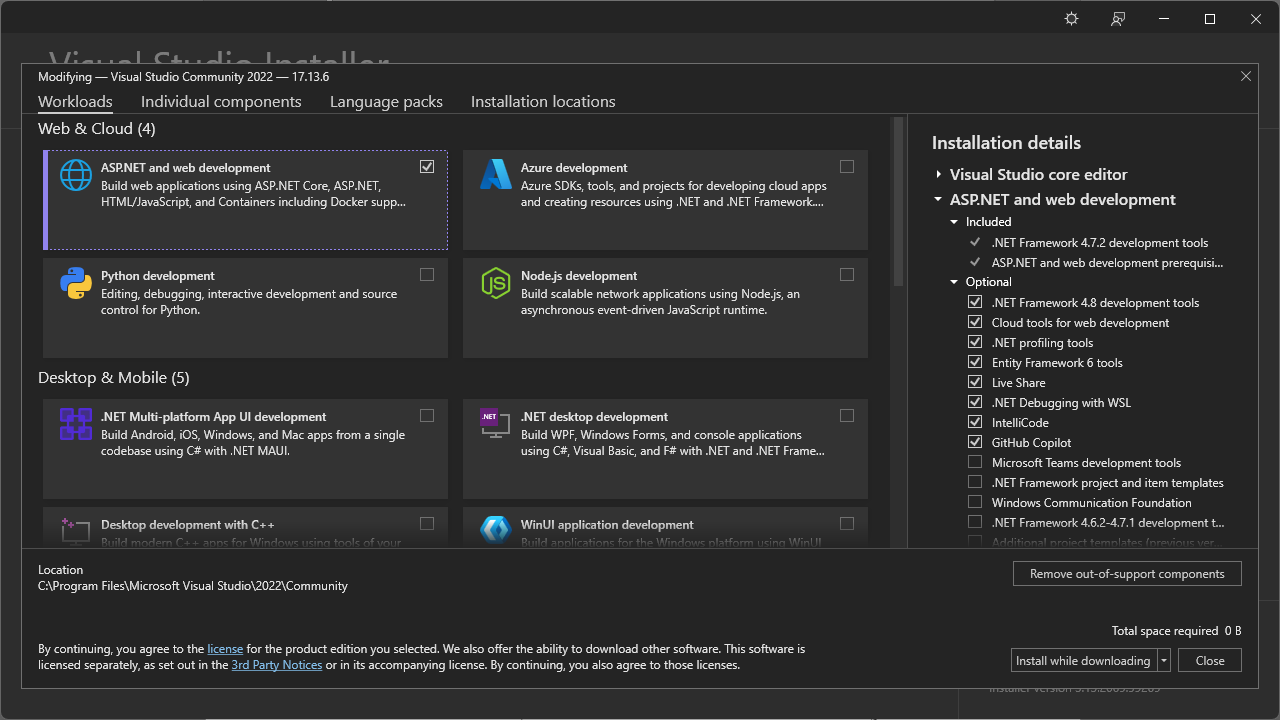


Hình .: Luồng đi của ASP.NET Core. Nguồn: <https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-9.0&tabs=visual-studio>

Người dùng (client) gửi HTTP Request (GET, POST, PUT, DELETE), ví dụ như POST một đối tượng có dạng {Name: “todo1”}. Request này sau đó sẽ được đưa đến controller tiếp nhận và xử lý. Controller sẽ có nhiệm vụ phân tích và ánh xạ dữ liệu vào model thông qua quá trình deserialize. Sau đó, controller sử dụng model để tương tác với tầng dữ liệu (Data Access Layer), thực hiện đọc/ghi dữ liệu từ cơ sở dữ liệu. Khi đã có kết quả xử lý thì dữ liệu sẽ được serialize thành định dạng JSON để trả ngược lại cho client thông qua HTTP Response. Luồng đi giúp đảm bảo việc tách biệt các vai trò rõ ràng giữa các thành phần trong hệ thống như giao tiếp, xử lý logic và truy xuất dữ liệu, giúp hệ thống dễ dàng mở rộng và bảo trì

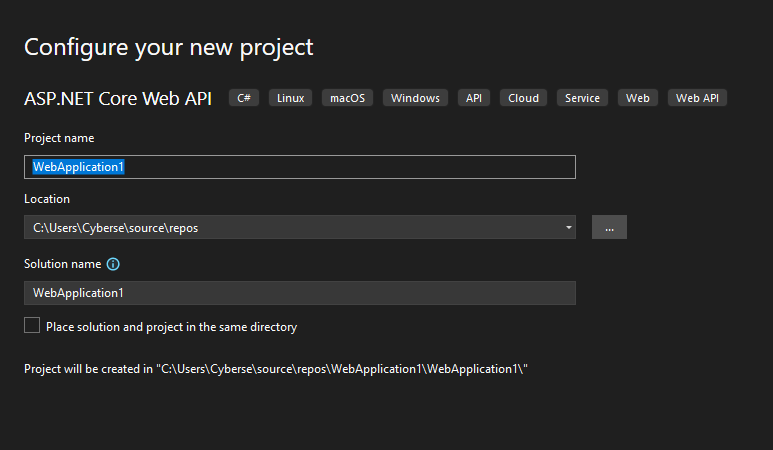
### Cách sử dụng

Một trong những cách phổ biến để tạo một ứng dụng ASP.NET Core là sử dụng Visual Studio 2022. Trước khi khởi tạo phải tải gói xây dựng Website với ASP.NET trên Visual Studio Installer



Hình .: Giao diện Visual Studio Installer

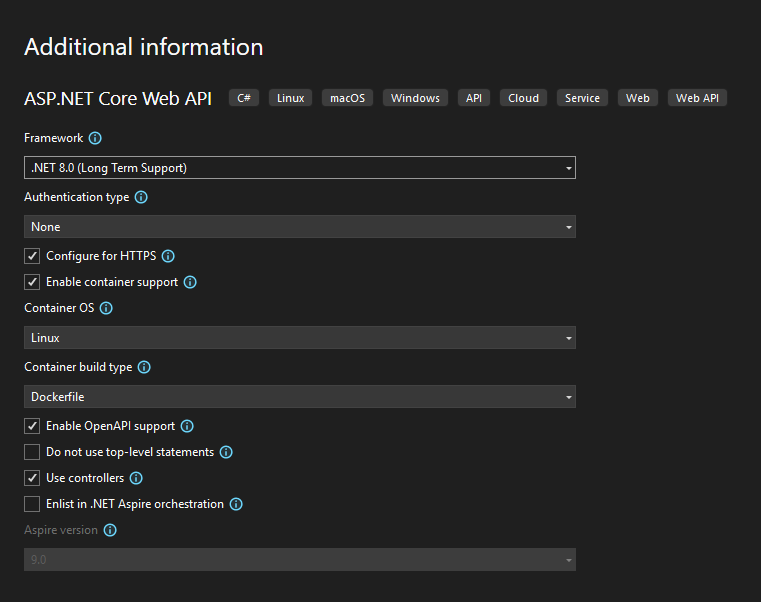
Sau khi cài đặt thì đã có thể bắt đầu tạo một dự án ASP.NET Core



Hình .: Giao diện đặt tên và vị trí dự án trong máy tính

Đây là bước đầu tiên để khởi tạo dự án, đặt tên cho dự án và đặt vị trí cho dự án, phần tên sẽ đặt tên cho file có đuôi .sln để khi người dùng quay lại dự án chỉ cần truy cập vào file đó.

Sau khi hoàn thành bước đầu tiên, bước thứ hai sẽ là thêm thông tin cho dự án



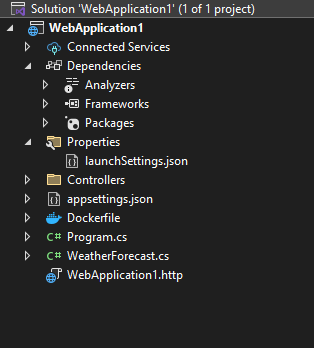
Hình .: Giao diện thêm thông tin cho dự án

Tại đây ta có thể lựa chọn phiên bản cho Framework, có thể là .NET 8 hoặc mới nhất là .NET 9

Với Authentication Type sẽ cấu hình xác thực trong ứng dụng nhằm để bảo mật ứng dụng, giúp bảo mật hệ thống. Mặc dù mặc định là None nhưng vẫn có thể được cấu hình thủ công để xác thực người dùng.

Với Container, ta có thể sử dụng với phiên bản Container bao gồm Windows hoặc Linux, và sử dụng Dockerfile để phù hợp cho việc sử dụng Docker. Hiện tại Docker đã được sử dụng phổ biến nên việc khởi tạo dự án kèm theo Dockerfile giúp người dùng có thể dễ dàng tạo dự án và triển khai lên Docker.

Sau khi điền đầy đủ thông tin sẽ bắt đầu tạo dự án.



Hình .: Cấu trúc dự án ASP.NET Core

Thành phần chính:

Dependencies: Chứa thông tin framework, thông tin Packages cài đặt từ NuGet.

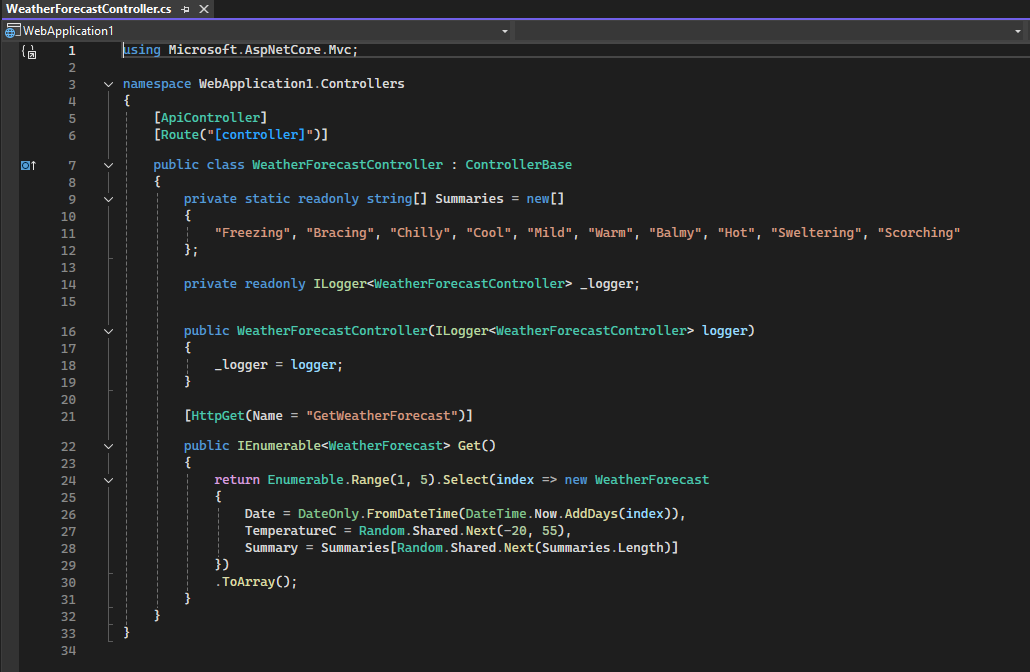
Properties: Chứa thông tin cấu hình host và port của các HTTP trong dự án.

Controllers: Chứa controller của tầng controller trong dự án.

appsetting.json: Chứa các thông tin setting như thông tin cơ sở dữ liệu, các API Key của ứng dụng bên thứ ba, JWT,…

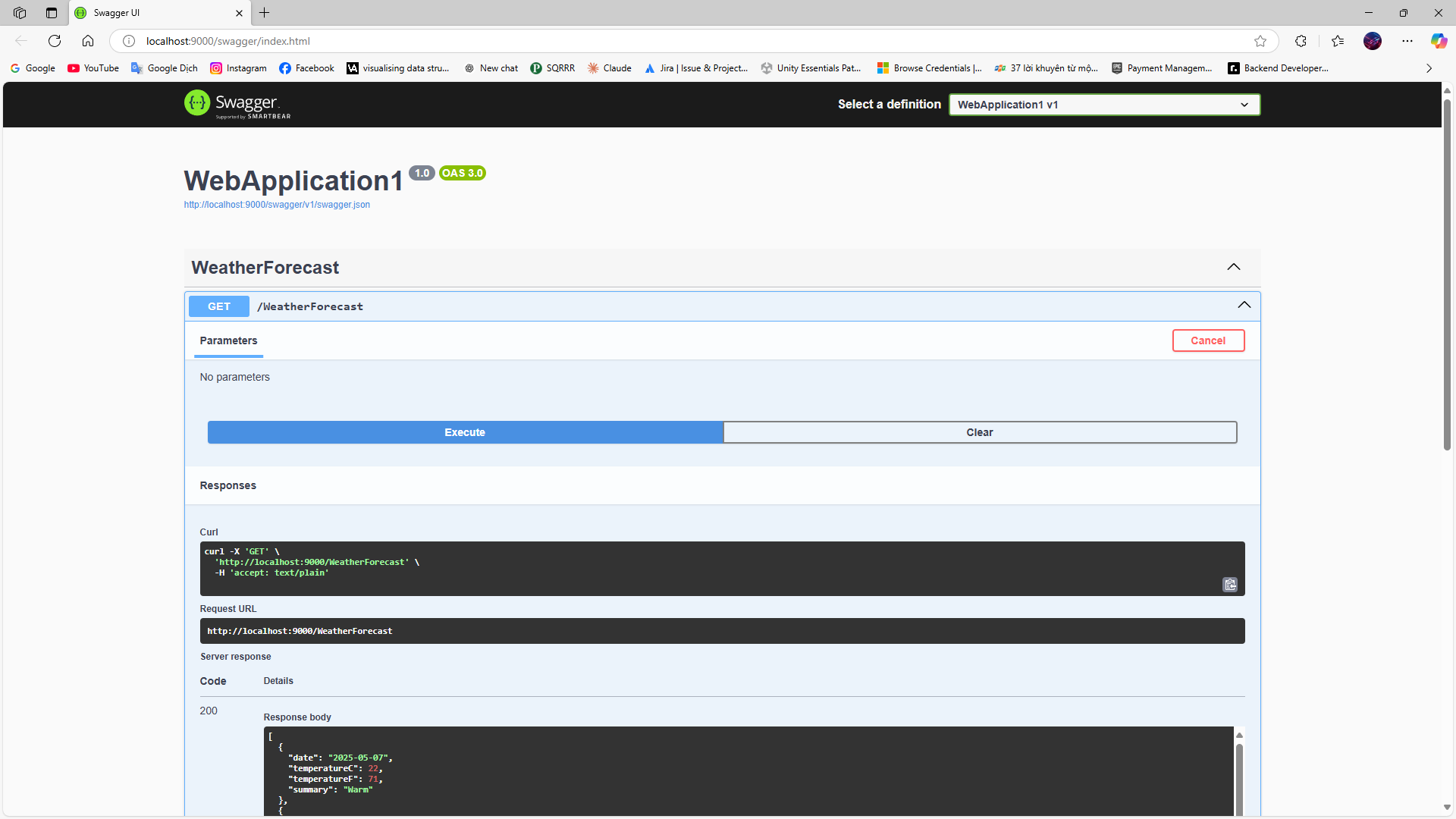
Program.cs: Chứa cấu hình chung của dự án, bao gồm Dependency Injection, Authentication và Authorization, Swagger,…

Trong Controllers đã được tạo sẵn Controllers để có thể dễ dàng chạy thử hệ thống.



Hình .: Controller ASP.NET Core

Chạy thử dự án và sử dụng API trực tiếp với Swagger trên ***localhost:9000:***



Hình .: Chương trình ASP.NET Core đầu tiên

## Giới thiệu về Next.js

Next.js là một Framework dành riêng cho React được sử dụng phổ biến để xây dựng và phát triển ứng dụng web hiện đại với khả năng Server-Side Rendering (SSR), Static Site Generation (SSG), tối ưu Search Engine Optimization (SEO) và hỗ trợ API Routes. Framework được phát triển bởi Vercel và được sử dụng rộng rãi để phát triển Website tối ưu hiệu suất.

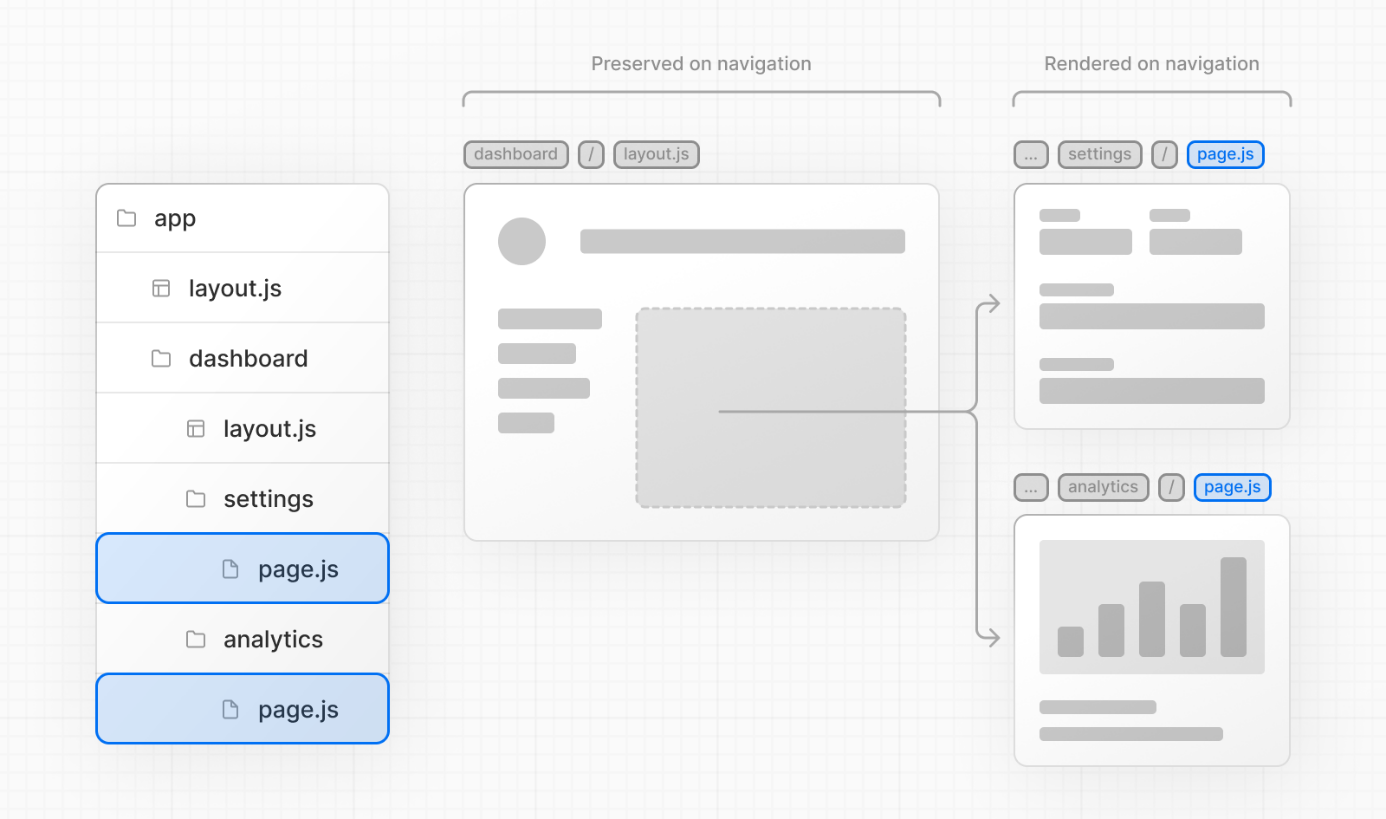
### Đặc điểm chính của Next.js

#### Hybrid Rendering (SSR & SSG)

Next.js hỗ trợ cả Server-Site Rendering, cho phép render nội dung trang trên server mỗi khi có request và Static Site Generation cho phép tạo trang tĩnh sẵn có khi build, giúp tăng tốc độ tải trang nhanh và tối ưu hiệu suất hơn.

#### Routing

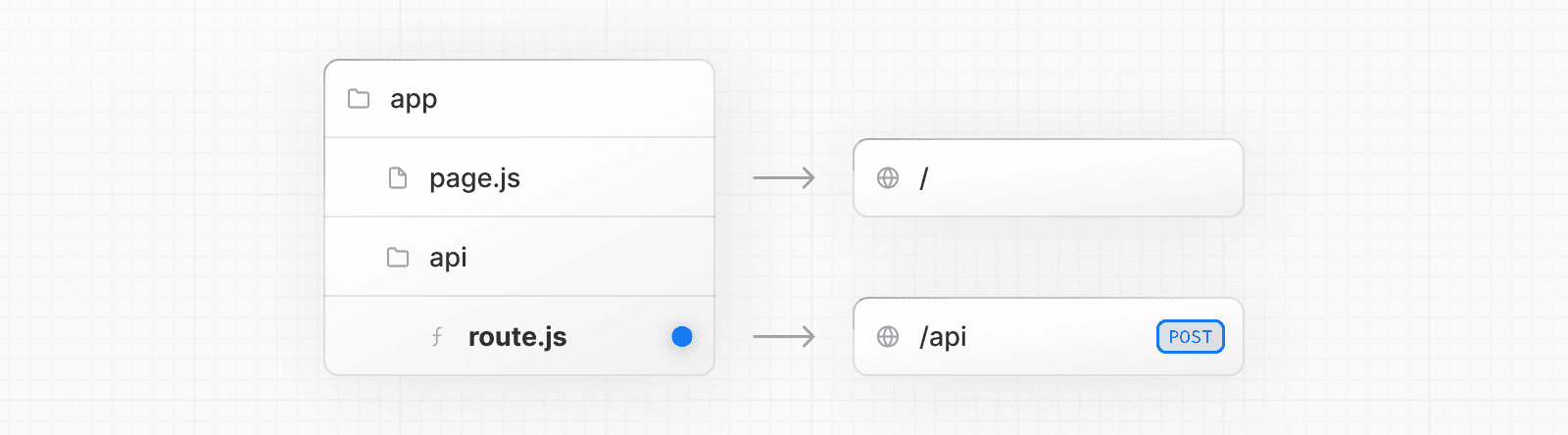
Next.js sử dụng hệ thống routing dựa trên file cho phép các trang là một file mà không cần cấu hình thủ công như React Route. Điều này giúp việc phát triển giao diện với Next trở nên dễ dàng và nhanh chóng hơn



Hình .: NextJS Routing. Nguồn: “https://nextjs.org/docs/app/building-your-application/routing/linking-and-navigating”

#### API Routes

Next.js cho phép tạo các API Backend ngay trong cùng dự án React thông qua thư mục như utils/APIs.tsx, giúp việc xử lý Backend trở nên nhẹ nhàng mà không cần một Server riêng.



Hình .: Route Handles của NextJS. Nguồn: “https://nextjs.org/docs/app/building-your-application/routing/route-handlers”

#### Search Engine Optimization (SEO)

Next.js tối ưu tốt SEO, cho phép hỗ trợ SSR và SSG để các công cụ tìm kiếm index nội dung trở nên dễ dàng hơn. Đồng thời với Image Optimization, việc tìm kiếm hình ảnh trở nên dễ dàng hơn với next/image giúp tối ưu hình ảnh.

#### Tailwind CSS

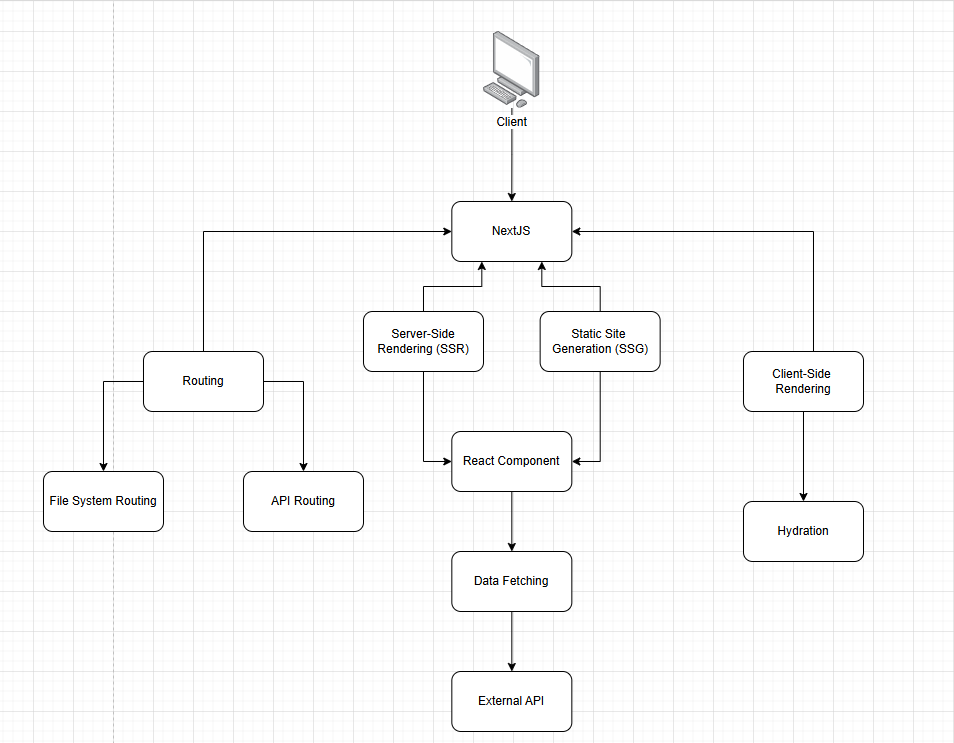
Next.js hỗ trợ Tailwind CSS giúp cho việc styling giao diện trở nên dễ dàng hơn, đồng thời có thể dễ dàng tích hợp các thư viện UI phổ biến như Chakra UI, Material UI.

#### Rendering

NextJS hỗ trợ 2 phương thức render chính bao gồm:

* **Server-Side Rendering (SSR):** Tăng tốc độ tải trang, hỗ trợ SEO tốt và cập nhật dữ liệu một cách nhanh chóng, phù hợp cho Website thay đổi liên tục
* **Static Site Generation (SSG):** Do có HTML được tải lên từ trước nên tốc độ tải trang sẽ rất nhanh, giúp tăng trải nghiệm người dùng, phù hợp cho Website ít thay đổi

### Kiến trúc NextJS



Hình .: Kiến trúc ứng dụng của NextJS

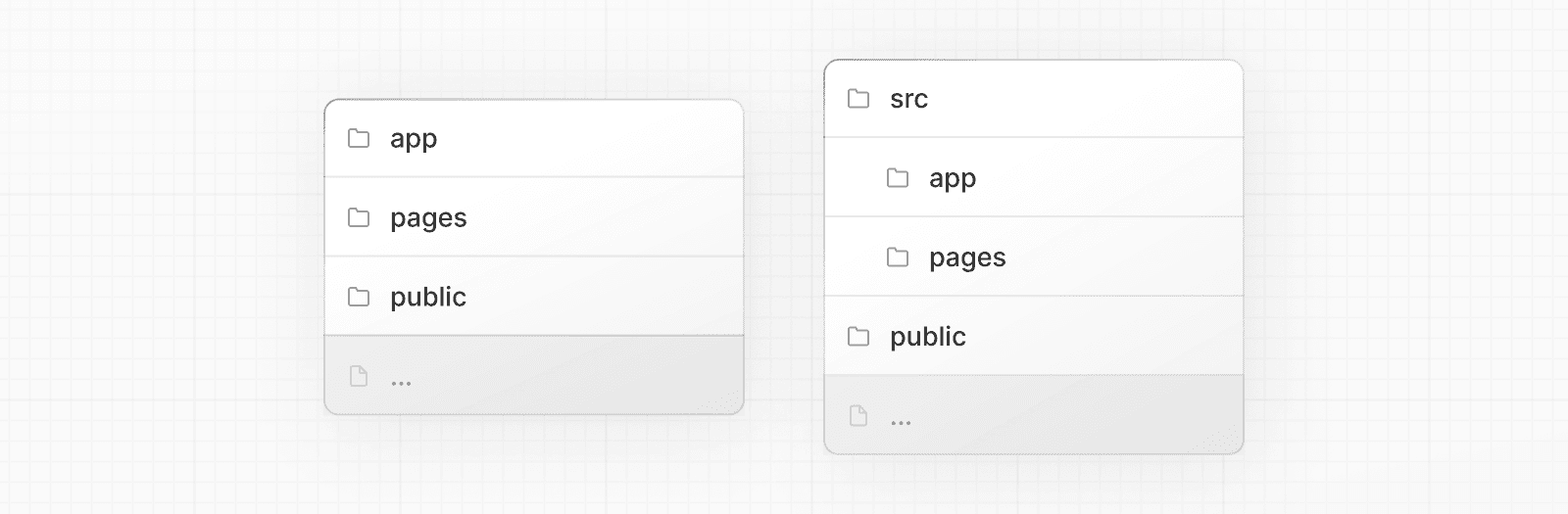
NextJS sử dụng Routing theo từng thư mục để định tuyến, mỗi thư mục sẽ chứa page và trở thành một Route, ví dụ như thư mục product chứa page.tsx, tương ứng như Route sẽ là localhost:3000/product. Ngoài ra, NextJS cũng cho phép xây dựng và sử dụng các endpoint trong cùng một cấu trúc thư mục.

NextJS hỗ trợ các cơ chế Rendering mạnh mẽ cho cả Client và Server, tiêu biểu như Server-Site Rendering cho phép cập nhật nội dung HTML liên tục và nhanh chóng, kết hợp với Static Side Generation để tải trang HTML đã có sẵn tại thời điểm build.

Với Client-Side Rendering ở phía client, Server sẽ được giảm tải khi Server chỉ cần gửi dữ liệu cần thiết thì Client sẽ xử lý Render.

### Cách sử dụng

Next.js sẽ có hai cách để tổ chức file và folder trong dự án



Hình .: Folder Next.js. Nguồn: <https://nextjs.org/docs/app/getting-started/project-structure>

Mặc dù có hai cách tổ chức, nhưng cấu trúc chung sẽ bao gồm của Next.js sẽ bao gồm:

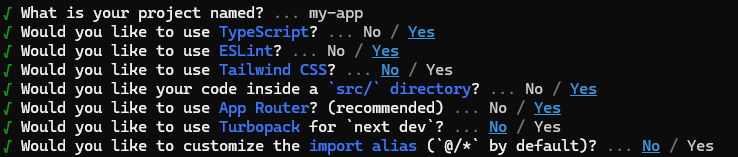
* src/ (hoặc chỉ có app và pages): sẽ chứa mã nguồn của hệ thống
* public/ : lưu trữ các tài nguyên tĩnh của dự án như hình ảnh…

Để khởi tạo dự án Next.js, ta cần cài đặt Node.js, sau đó sử dụng terminal và sử dụng lệnh để tạo dự án:



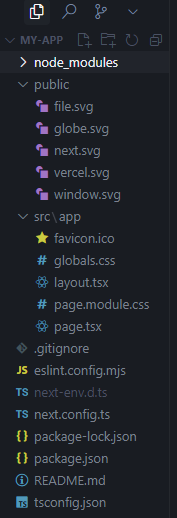
Hình .: Tạo dự án Next.js

Chọn các yêu cầu sử dụng trong dự án



Hình .: Lựa chọn dự án Next.js

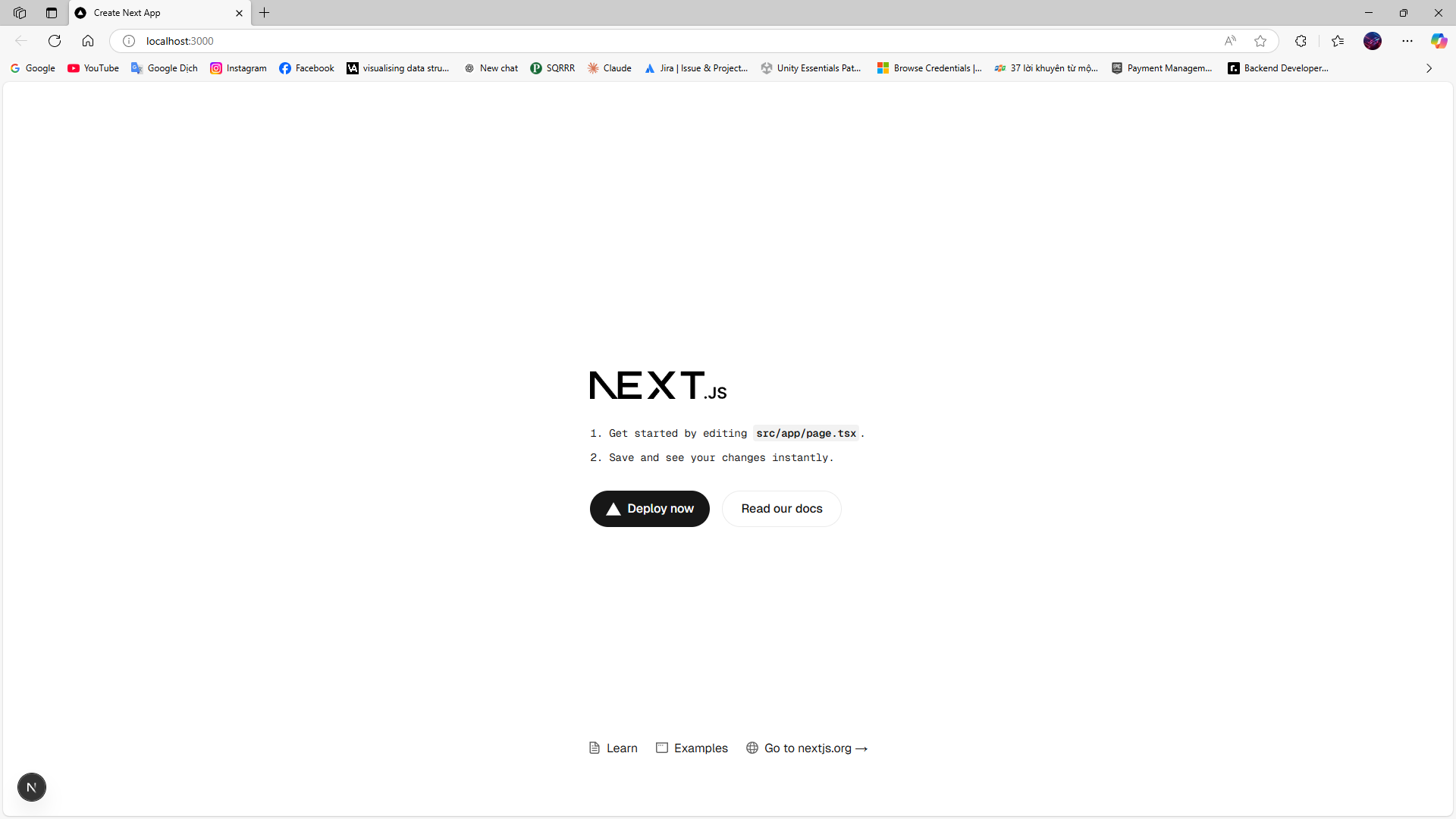
Sau khi lựa chọn hoàn tất thì sẽ có cấu trúc thư mục đơn giản như sau:



Hình .: Cấu trúc thư mục của Next.js

Ta sử dụng lệnh npm install hoặc yarn để cài đặt các dependency cần thiết.

Để chạy ứng dụng ta dùng lệnh npm run dev hoặc yarn run dev:



Hình .: Ứng dụng Next.js đầu tiên

## Các công cụ và dịch vụ

### PostgreSQL

PostgreSQL là một cơ sở dữ liệu quan hệ, nổi bật với khả năng mở rộng, bảo mật và nhất quán dữ liệu, giúp cho việc lưu trữ và mở rộng dữ liệu với PostgreSQL trở nên linh hoạt và dễ dàng hơn.  
PostgreSQL sẽ là nơi để lưu trữ những dữ liệu mang tính nhất quán như thông tin người dùng, thông tin hóa đơn để chặt chẽ hơn trong việc quản lý dữ liệu.

### MongoDB

MongoDB là một cơ sở dữ liệu không quan hệ, cho phép lưu trữ dữ liệu linh hoạt, tốc độ truy vấn cao. MongoDB lưu trữ dữ liệu theo dạng documents (BSON), một dạng dữ liệu tương tự như JSON.

Nhờ sự linh hoạt vốn có, MongoDB giúp cho việc lưu trữ những thông tin linh hoạt như thông tin sản phẩm trở nên dễ dàng hơn, truy vấn cũng trở nên nhanh chóng hơn.

### Redis

Redis là một cơ sở dữ liệu không quan hệ, cho phép lưu trữ dữ liệu trên RAM, giúp việc ghi, đọc và xóa dữ liệu trở nên nhanh chóng dưới dạng key-value.

Với tốc độ đọc ghi và xóa dữ liệu nhanh chóng, Redis giúp cho việc lưu trữ dữ liệu tạm thời như thông tin giỏ hàng trở nên dễ dàng hơn, xóa cũng trở nên nhanh chóng.

# PHÁT TRIỂN HỆ THỐNG BÁN NÔNG SẢN SỬ DỤNG KIẾN TRÚC MICROSERVICE

## Giới thiệu hệ thống

Hệ thống thương mại điện tử bán nông sản là một hệ thống không chỉ giúp người tiêu dùng, đặc biệt là nhà nông có thể mua hàng một cách dễ dàng ngay tại nhà với việc mua hàng trên Website, giúp tiết kiệm thời gian và chi phí đi lại.

Hệ thống giúp người tiêu dùng tiếp cận được những sản phẩm nông sản chất lượng, an toàn và rõ nguồn gốc. Đồng thời người tiêu dùng cũng được tiếp cận những sản phẩm mới nhất trên thị trường, giảm thiểu sự phụ thuộc vào một kênh bán hàng hay chỉ những sản phẩm đã xuất hiện trên thị trường.

## Phân tích hệ thống

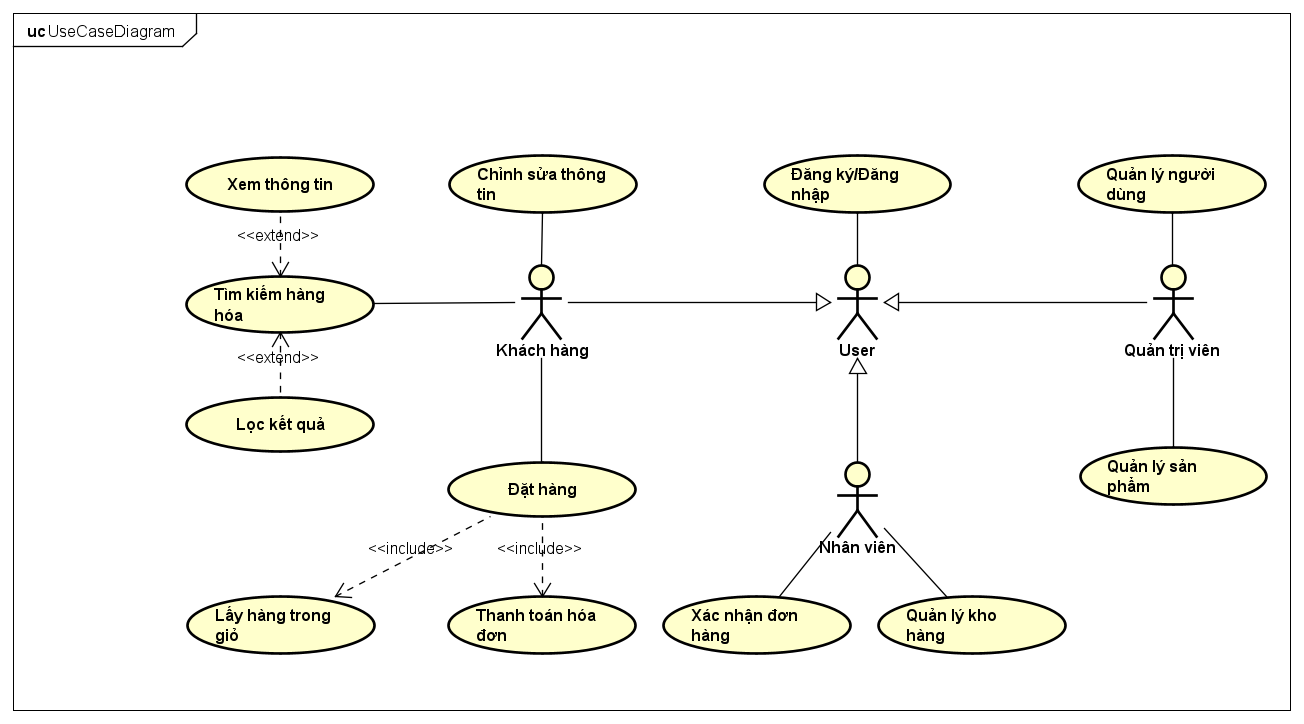
### Lược đồ Use Case

Hệ thống sẽ bao gồm 3 actor chính:

* Admin: Vai trò quản lý người dùng và hệ thống.
* Staff: Vai trò quản lý kho hàng và đơn hàng.
* Customer: Vai trò mua hàng và thanh toán.

Các chức năng chính trong hệ thống:

* Mua hàng: Hàng hóa được khách hàng lựa chọn và đưa vào giỏ hàng.
* Thanh toán: Lấy hàng hóa trong giỏ của khách hàng và tạo hóa đơn thanh toán.
* Quản lý kho hàng: Hàng hóa được quản lý số lượng bởi nhân viên.
* Quản lý sản phẩm: Các sản phẩm sẽ được quản trị viên xem xét trước khi đưa vào hệ thống bán hàng.



Hình .: Lược đồ Use Case

### Đặc tả Use Case

|  |  |
| --- | --- |
| UseCase Id | UC-01 |
| Tên UseCase | Mua hàng |
| Mô tả | Khách hàng lựa chọn hàng hóa và đưa vào giỏ hàng |
| Actor chính | Khách hàng |
| Actor phụ | Không có |
| Tiền điều kiện | Khách hàng đã truy cập vào Website |
| Hậu điều kiện | Hiển thị danh sách sản phẩm đã đưa vào giỏ |
| Luồng hoạt động | Khách hàng nhập tên sản phẩm vào thanh tìm kiếm.  Hệ thống tìm kiếm sản phẩm dựa vào từ khóa.  Hệ thống hiển thị danh sách sản phẩm.  Khách hàng chọn sản phẩm và thêm vào giỏ hàng. |
| Luồng thay thế | (2a) Nếu ô tìm kiếm trống, hàng hóa sẽ hiển thị gợi ý trên giao diện.  (2b) Nếu không tìm thấy sản phẩm theo từ khóa sẽ thông báo không tìm thấy sản phẩm. |
| Luồng ngoại lệ | Không có |

Bảng .: UseCase mua hàng

|  |  |
| --- | --- |
| UseCase Id | UC-02 |
| Tên UseCase | Đặt hàng và thanh toán. |
| Mô tả | Khách hàng lấy hàng trong giỏ sau đó đặt hàng và thanh toán. |
| Actor chính | Khách hàng. |
| Actor phụ | Không có. |
| Tiền điều kiện | Khách hàng đã đăng nhập hoặc điền thông tin giao hàng vào hệ thống. |
| Hậu điều kiện | Đơn hàng đã được tạo hóa đơn và thanh toán thành công. |
| Luồng hoạt động | 1. Khách hàng truy cập vào giỏ hàng. 2. Khách hàng kiểm tra thông tin hàng hóa trong giỏ. 3. Khách hàng nhấn nút đặt hàng. 4. Hệ thống tạo đơn hàng và yêu cầu thanh toán hóa đơn. 5. Khách hàng được chuyển đến màn hình thanh toán. 6. Thanh toán hóa đơn và kết thúc. |
| Luồng thay thế | (3a) Nếu khách hàng chưa đăng nhập hoặc chưa điền thông tin thì sẽ báo yêu cầu đăng nhập hoặc nhập thông tin. |
| Luồng ngoại lệ | (4a) Nếu hệ thống lỗi sẽ lưu lại giỏ hàng và hiển thị thông báo lỗi |

Bảng .: UseCase Thanh toán

|  |  |
| --- | --- |
| UseCase Id | UC-03 |
| Tên UseCase | Quản lý kho hàng |
| Mô tả | Nhân viên kiểm tra tồn kho và nhập hàng mới vào kho |
| Actor chính | Nhân viên |
| Actor phụ | Không có |
| Tiền điều kiện | Nhân viên đã đăng nhập vào hệ thống |
| Hậu điều kiện | Dữ liệu kho hàng đã được nhân viên cập nhật |
| Luồng hoạt động | 1. Nhân viên đăng nhập vào hệ thống. 2. Nhân viên truy cập vào chức năng quản lý kho hàng. 3. Nhân viên kiểm tra danh sách hàng hóa trong ngày và số lượng tồn kho. 4. Kiểm tra và cập nhật thông tin về số lượng hàng hóa trong kho. 5. Hệ thống lưu lại cập nhật kho hàng mới nhất. |
| Luồng thay thế | (4a) Nếu hàng hóa đủ thì sẽ không cần nhập thêm hàng hóa trong kho |
| Luồng ngoại lệ | (5a) Nếu hệ thống lỗi thì sẽ hiển thị thông báo và yêu cầu thử lại. |

Bảng .: UseCase quản lý kho hàng

|  |  |
| --- | --- |
| UseCase Id | UC-04 |
| Tên UseCase | Quản lý sản phẩm |
| Mô tả | Quản trị viên quản lý thông tin sản phẩm trong kho hàng |
| Actor chính | Quản trị viên |
| Actor phụ | Không có |
| Tiền điều kiện | Quản trị viên đã đăng nhập hệ thống |
| Hậu điều kiện | Danh sách sản phẩm được quản lý và cập nhật thông tin mới nhất |
| Luồng hoạt động | 1. Quản trị viên truy cập chức năng quản lý sản phẩm. 2. Quản trị viên có thể thêm, sửa hoặc xóa thông tin sản phẩm. 3. Kiểm tra dữ liệu được quản trị viên thêm, sửa hoặc xóa có hợp lệ không. 4. Cập nhật danh sách sản phẩm mới nhất. |
| Luồng thay thế | (2a) Nếu thông tin không có gì thay đổi thì quản trị viên có thể thoát ra mà không phải cập nhật gì. |
| Luồng ngoại lệ | (3a) Nếu dữ liệu không hợp lệ sẽ thông báo lỗi kèm chi tiết. |

Bảng .: UseCase quản lý sản phẩm

## Thiết kế cơ sở dữ liệu

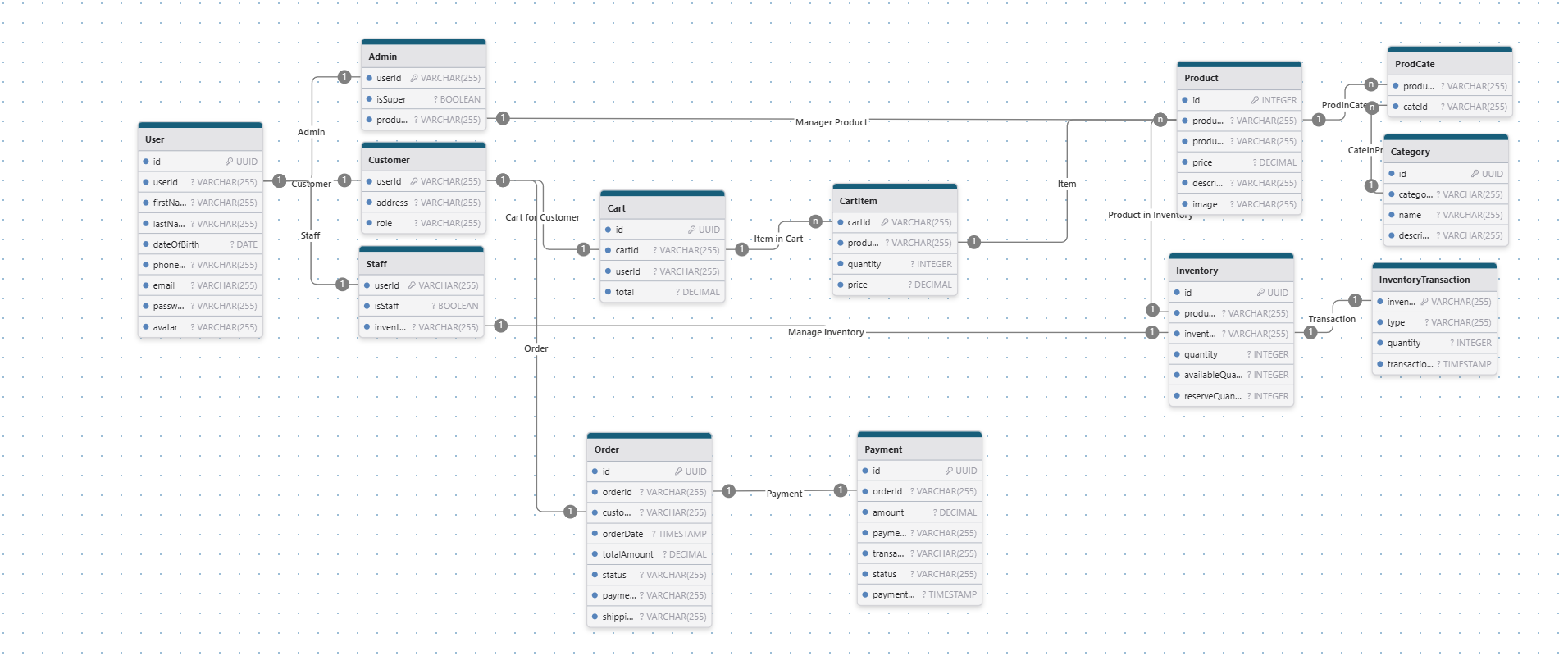
Thông tin về lược đồ cơ sở dữ liệu bao gồm:

ProductService

PaymentService

CartService

UserService



Hình .: Lược đồ cơ sở dữ liệu

Hệ thống sẽ gồm 4 dịch vụ chính với 4 cơ sở dữ liệu khác nhau bao gồm:

### UserService

User: Chứa các thông tin cơ bản liên quan đến User.

Admin: Thông tin quản trị viên.

Staff: Thông tin nhân viên.

Customer: Thông tin khách hàng.

### CartService

Cart: Chứa thông tin giỏ hàng thuộc một khách hàng.

CartItem: Chứa thông tin hàng hóa được thêm vào giỏ.

### ProductService

Product: Chứa thông tin hàng hóa.

Category: Chứa thông tin danh mục.

Inventory: Chứa thông tin kho chứa hàng hóa.

InventoryTransaction: Chứa thông tin về thời gian nhập xuất hàng hóa.

### PaymentService

Order: Chứa thông tin hóa đơn sau khi mua hàng và tiến hành thanh toán.

Payment: Chứa thông tin thanh toán thông qua ngân hàng và ví điện tử

### Mối quan hệ giữa các bảng

User – Admin, Staff, Customer (1 – 1): Một người dùng sẽ tương ứng với một thông tin của User, các Role sẽ có một vài điểm khác biệt nhưng không nhiều.

User – Cart (1 – 1): Một người dùng sẽ có một giỏ hàng cho mình, hàng hóa sẽ được xóa sau khi thanh toán.

Cart – CartItems (1 – N): Một giỏ hàng có thể chứa nhiều hàng hóa và khi người dùng chọn lấy hàng hóa sẽ chỉ lấy hàng đó vào một giỏ hàng của người dùng.

CartItems – Product (1 – 1): Một đồ vật trong giỏ hàng có thể là nhiều sản phẩm, một sản phẩm chỉ thuộc một đồ vật trong sản phẩm.

Product – Category (N – N): Một sản phẩm có thể thuộc nhiều danh mục và một danh mục có thể chứa nhiều sản phẩm.

Product – Inventory (N – 1): Một kho hàng có thể chứa nhiều sản phẩm, một sản phẩm chỉ thuộc một kho hàng.

Inventory – InventoryTransaction (1 – N): Một giao dịch chỉ liên quan tới một kho hàng và một kho hàng sẽ có nhiều giao dịch.

User – Order (1 – N): Một khách hàng sẽ thuộc có nhiều hóa đơn và một hóa đơn sẽ chỉ thuộc về một khách hàng.

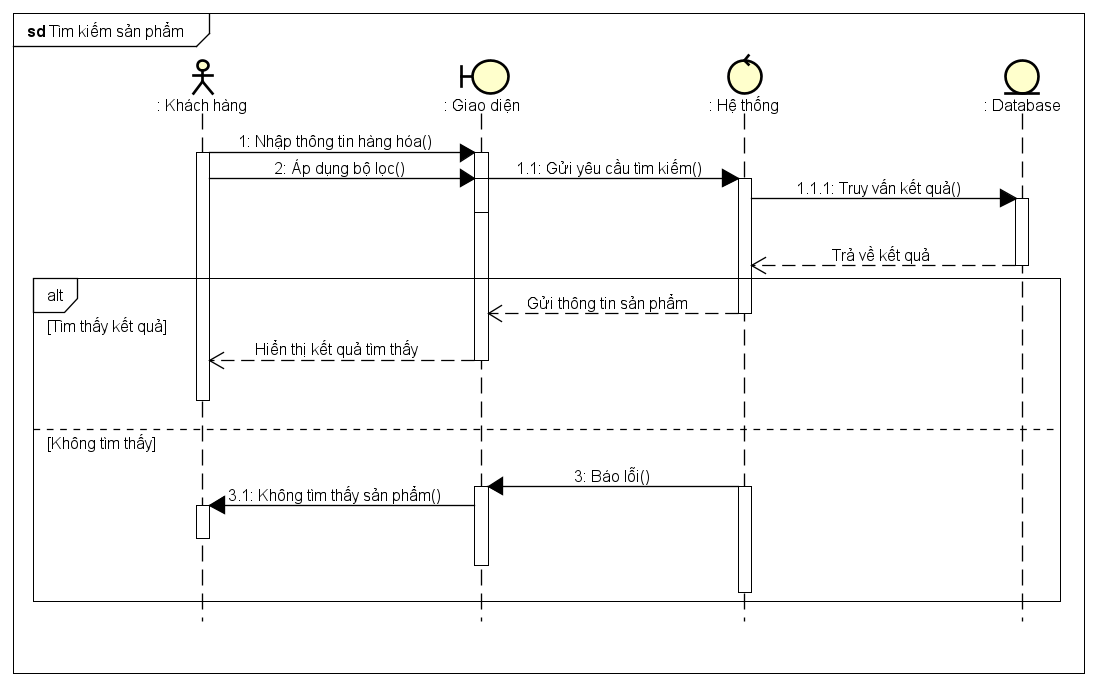
Order – Payment (1 – 1): Một hóa đơn sẽ có một phương thức thanh toán và một phương thức thanh toán sẽ được ghi nhận vào một hóa đơn.

## Kiến trúc hệ thống

### Sơ đồ tuần tự (Sequence Diagram)

Sơ đồ tuần tự này sẽ hiển thị luồng đi dữ liệu của các chức năng có trong hệ thống

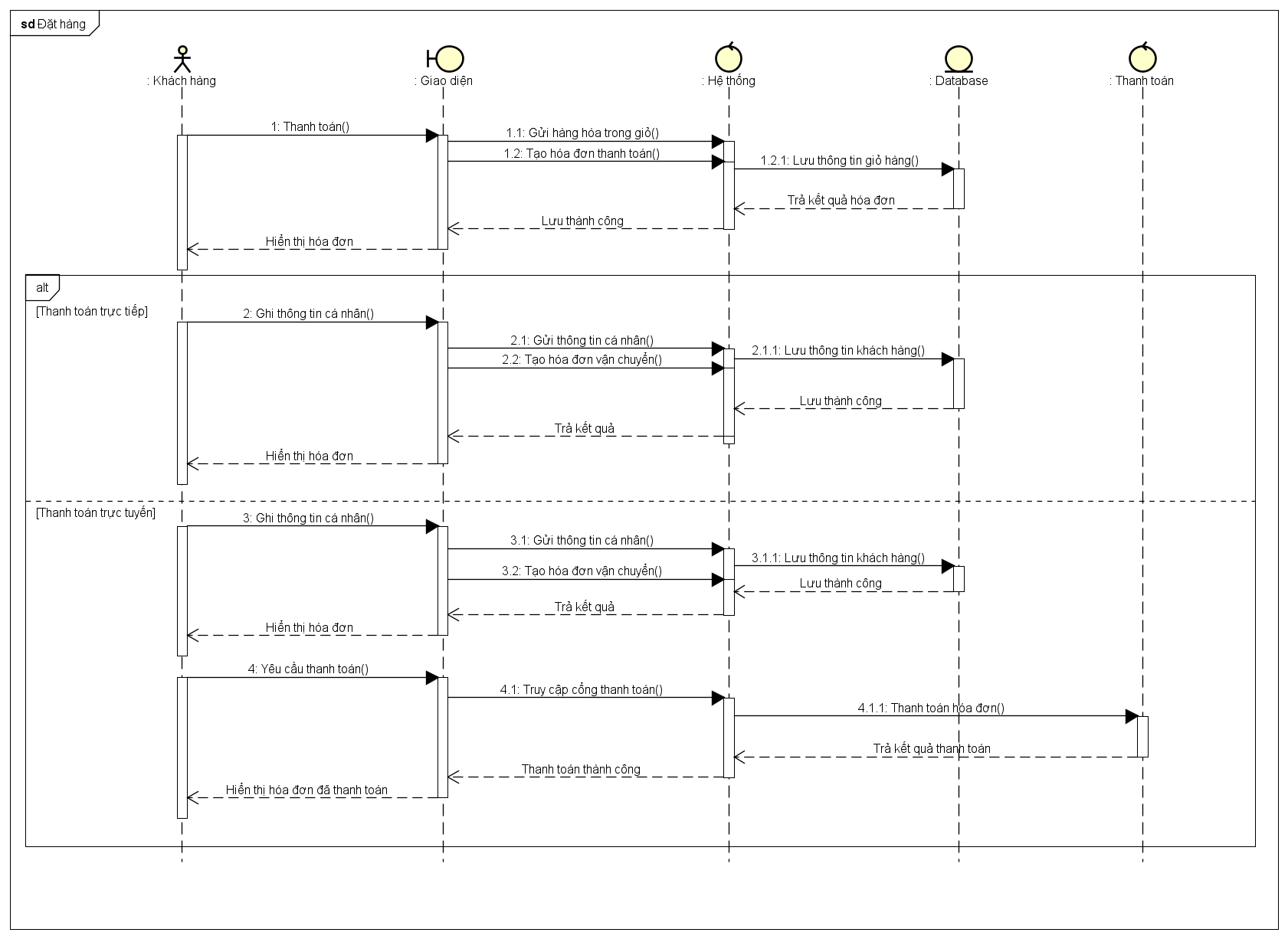
#### Tìm kiếm sản phẩm



Hình .: Sơ đồ tuần tự tìm kiếm sản phẩm

Trong sơ đồ này khi người dùng nhập thông tin tìm kiếm thì từ khóa tìm kiếm sẽ được xử lý, sau đó sẽ trả về kết quả tìm kiếm, với hai trường hợp là tìm thấy thì sẽ trả kết quả và không tìm thấy sẽ báo lỗi.

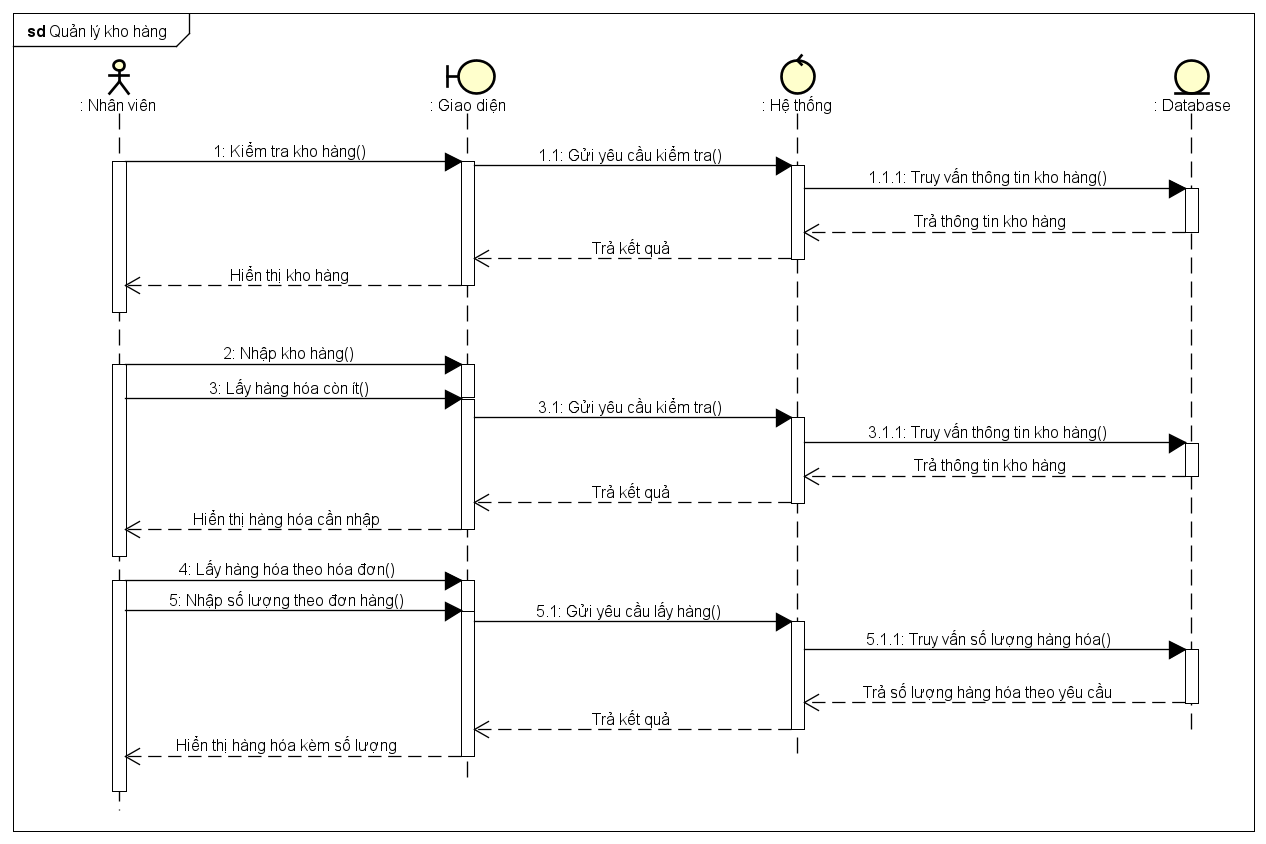
#### Đặt hàng



Hình .: Sơ đồ tuần tự đặt hàng

Trong sơ đồ này người dùng sẽ thanh toán, sau khi lấy hết hàng hóa trong giỏ sẽ đưa vào thanh toán, khách hàng có thể lựa chọn thanh toán trực tiếp hoặc trực tuyến, khi thanh toán trực tiếp sẽ hiển thị hóa đơn và khách hàng trả tiền mặt bên ngoài, với thanh toán trực tuyến sẽ gọi đến ngân hàng hoặc ví điện tử để thanh toán hóa đơn.

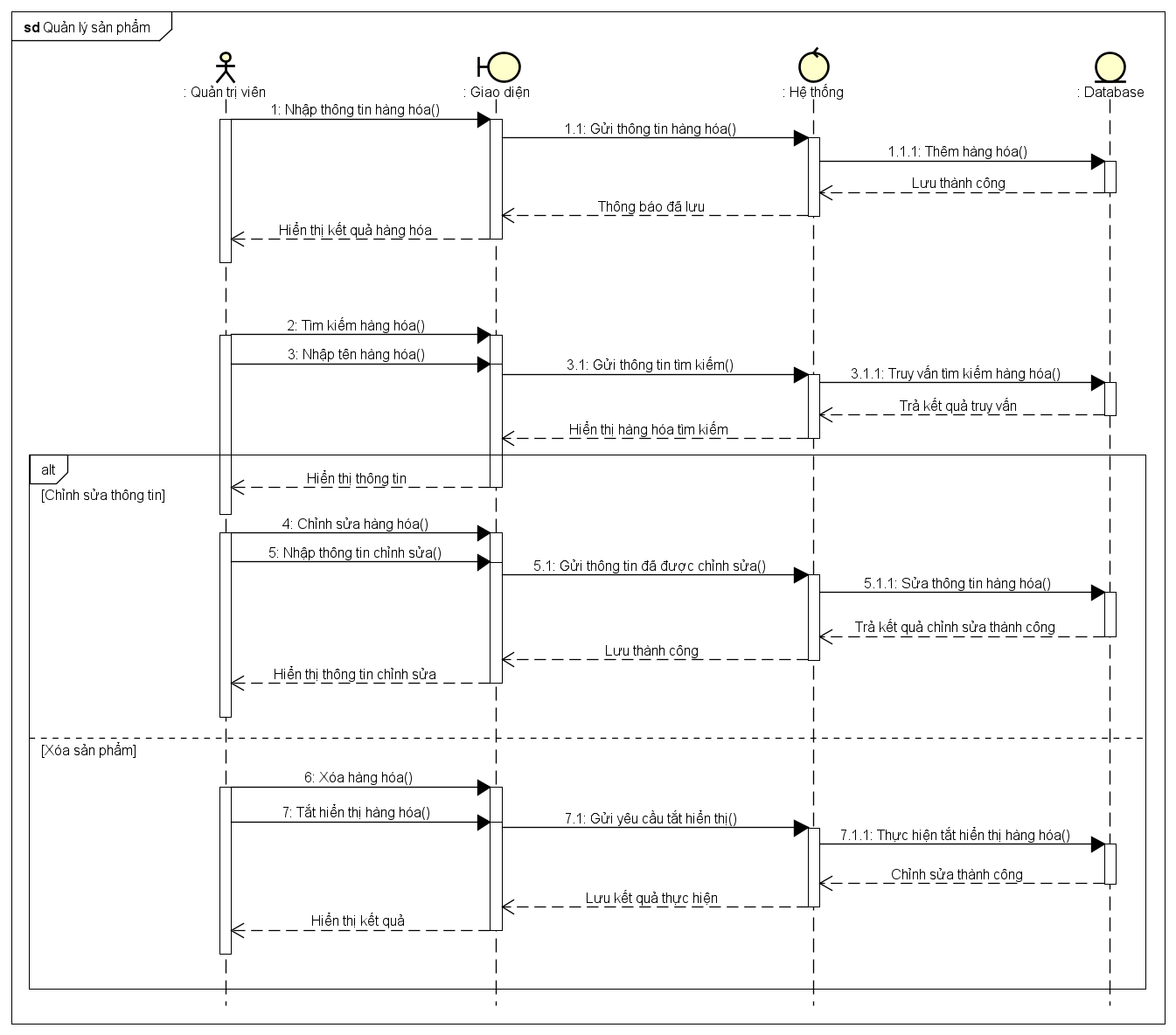
#### Quản lý kho hàng



Hình .: Sơ đồ tuần tự quản lý kho hàng

Trong sơ đồ này, khi nhập hàng thì nhân viên sẽ ghi nhận hàng hóa vừa nhập hàng vào kho và khi lấy hàng để giao cho khách hàng sẽ nhập lượng hàng hóa để bán hàng.

#### Quản lý sản phẩm



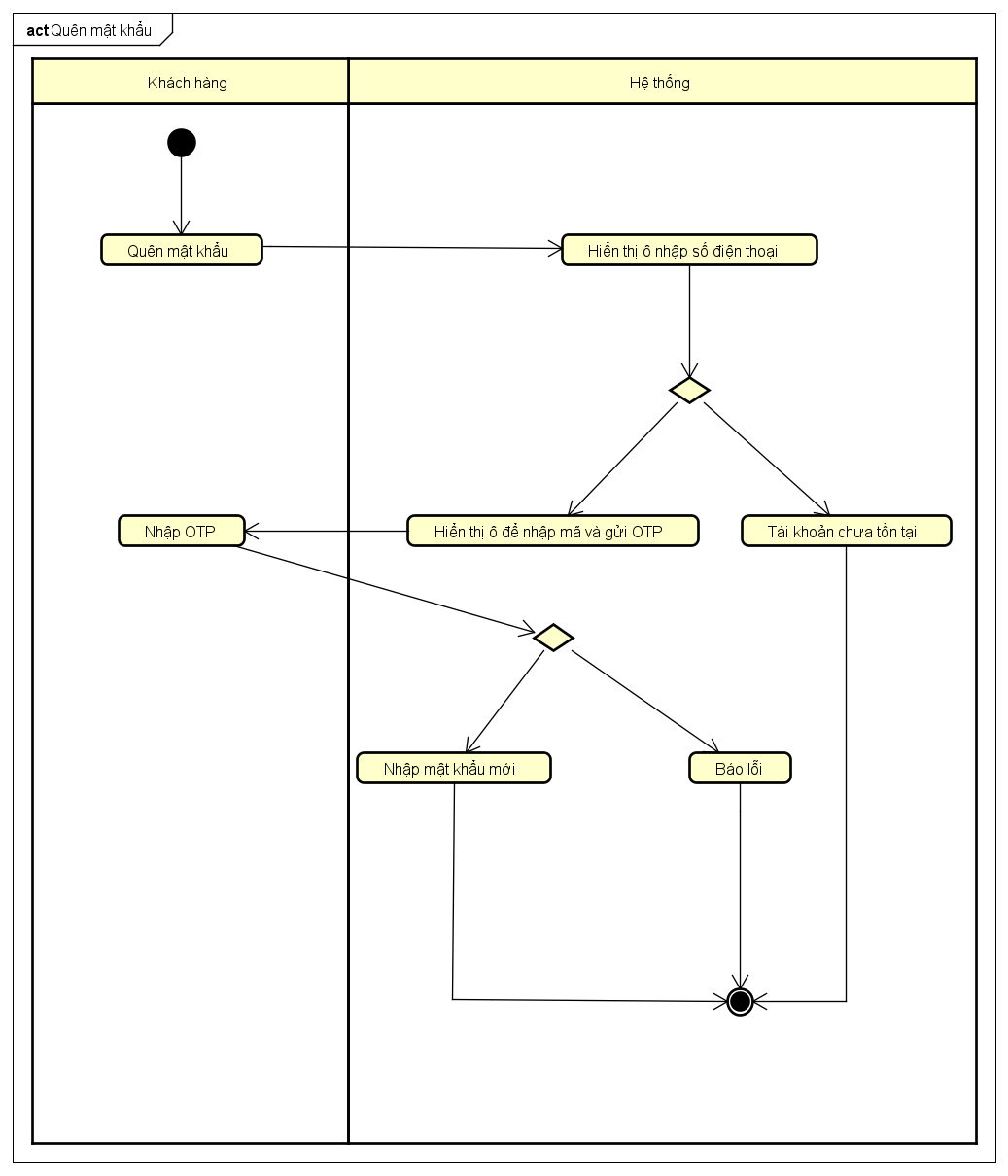
Hình .: Sơ đồ tuần tự quản lý sản phẩm

Sơ đồ này sẽ sử dụng cho chức năng quản lý sản phẩm, người quản trị sẽ nhập thông tin hàng hóa, vì hàng hóa sẽ tùy thuộc vào việc có thể nhập và bán nên phải xem xét trước khi đưa vào hệ thống bán hàng.

### Sơ đồ hoạt động (Activity Diagram)

Sơ đồ hoạt động này sẽ hiển thị cách hoạt động của từng chức năng có trong hệ thống

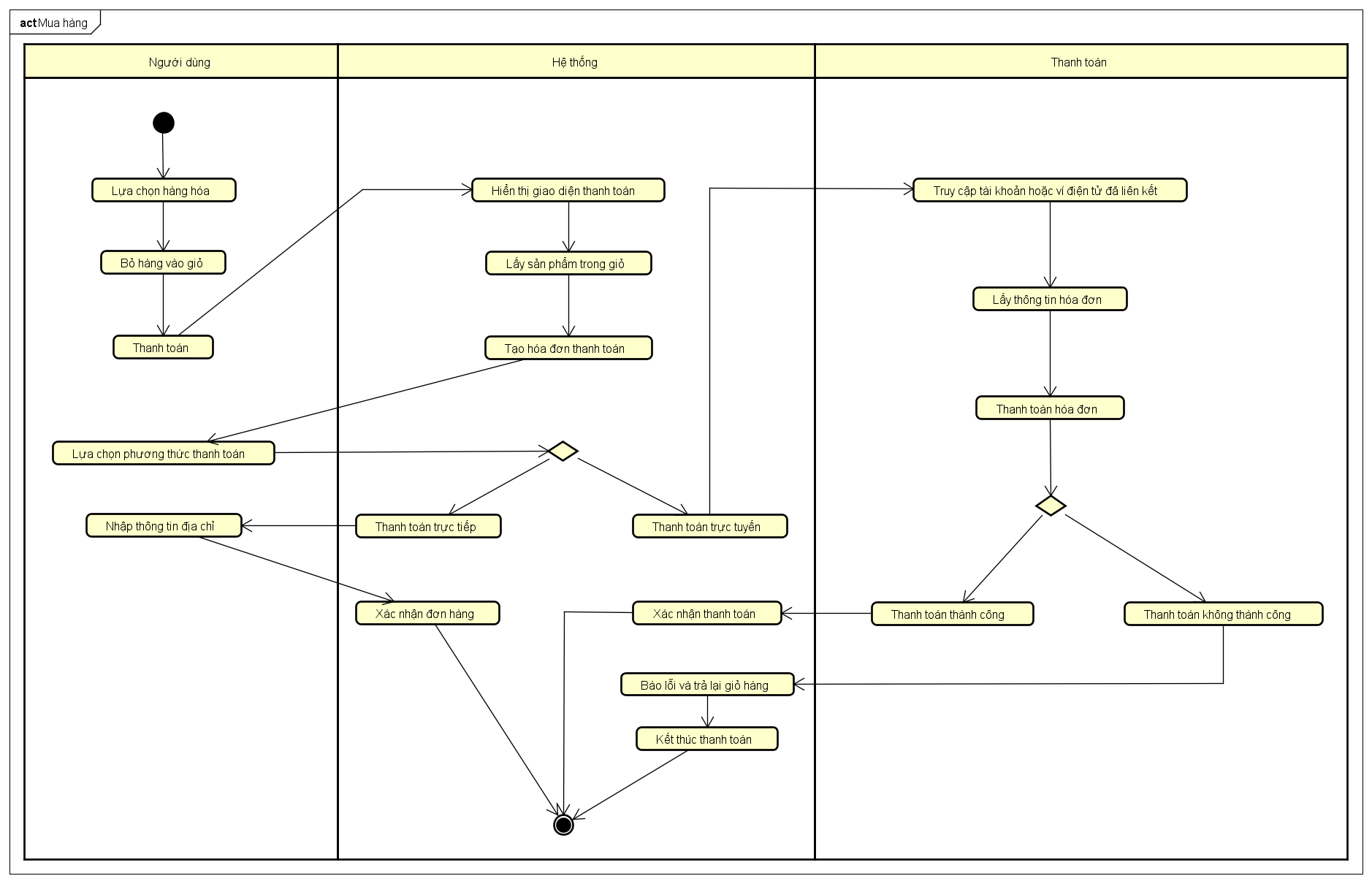
#### Quên mật khẩu



Hình .: Sơ đồ hoạt động cho chức năng quên mật khẩu

Chức năng này cho phép người dùng khi quên mật khẩu sẽ truy cập để có thể lấy thông lại thông tin của mình.

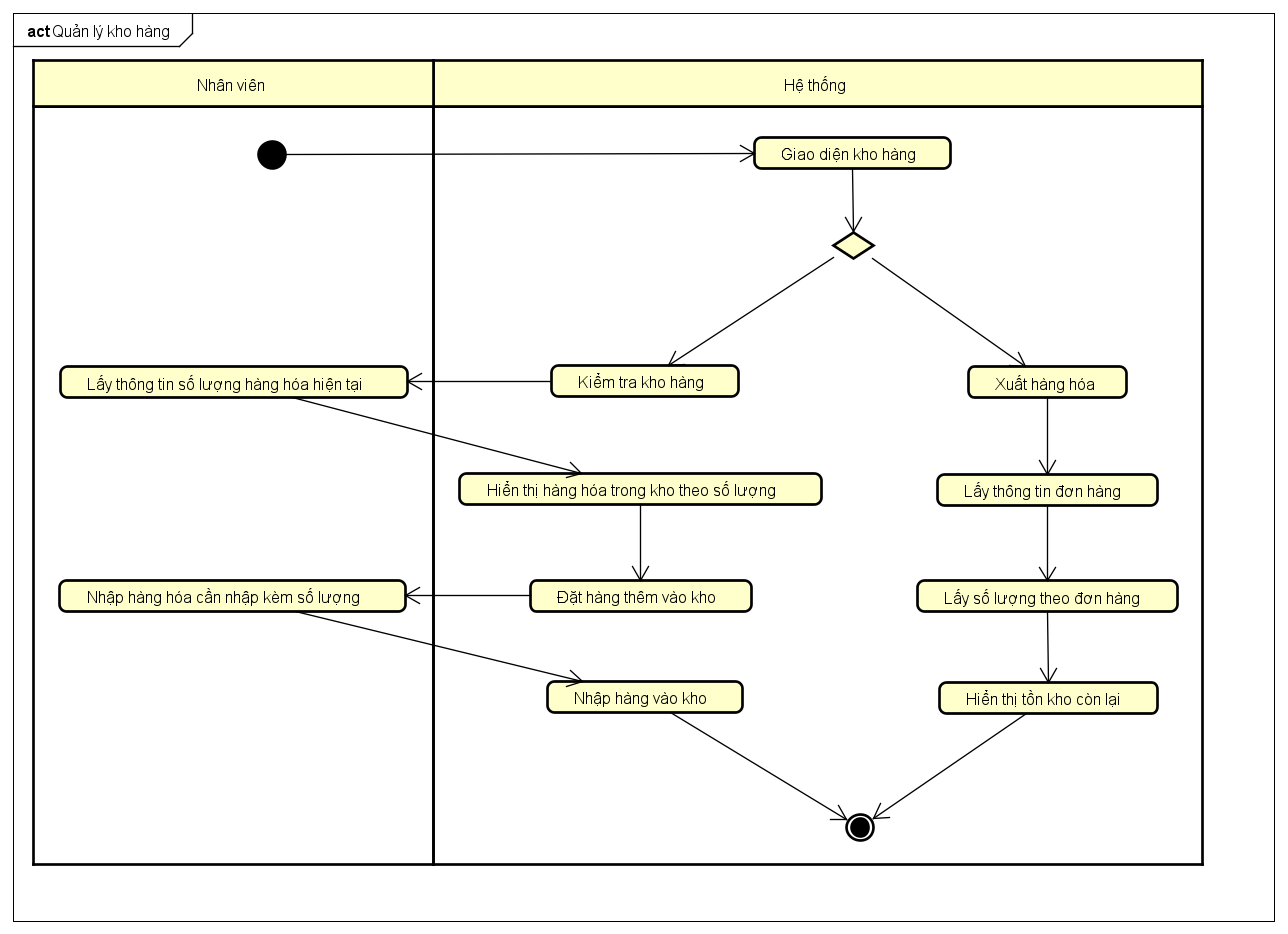
#### Mua hàng



Hình .: Sơ đồ hoạt động cho chức năng mua hàng

Sơ đồ này sẽ hiển thị cách hoạt động của chức năng mua hàng, lấy hết sản phẩm khi người dùng bỏ hàng vào giỏ sẽ hiển thị hóa đơn với sản phẩm kèm số lượng được đưa vào giỏ hàng, sau đó sẽ được tính giá tiền để tiến hành thanh toán.

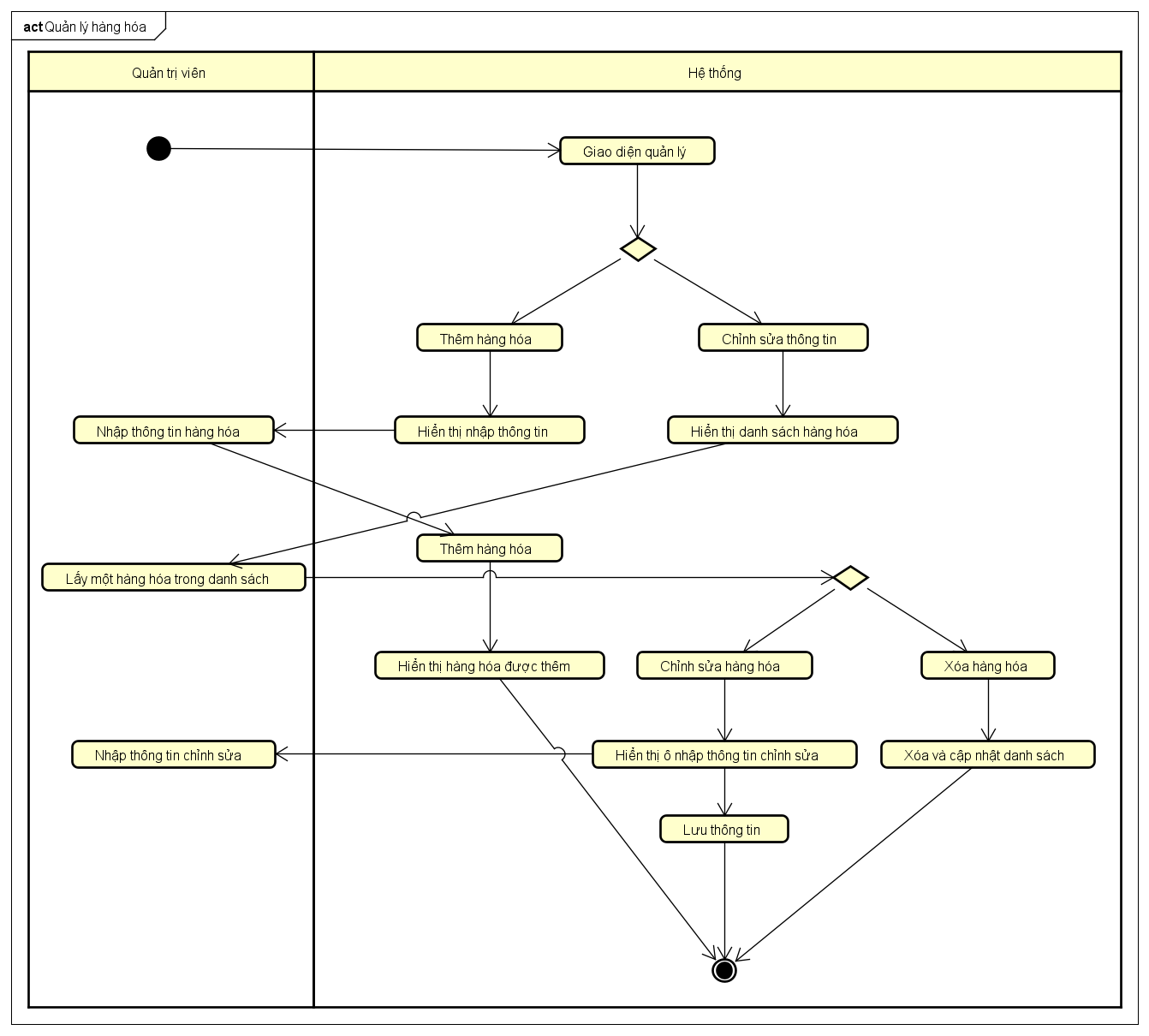
#### Quản lý kho hàng



Hình .: Sơ đồ hoạt động cho chức năng quản lý kho hàng

Sơ đồ này sẽ hiển thị cách hoạt động của chức năng quản lý kho hàng, kho hàng sẽ được quản lý bởi nhân viên. Có thể lấy hàng hoặc nhập hàng để quản lý số lượng.

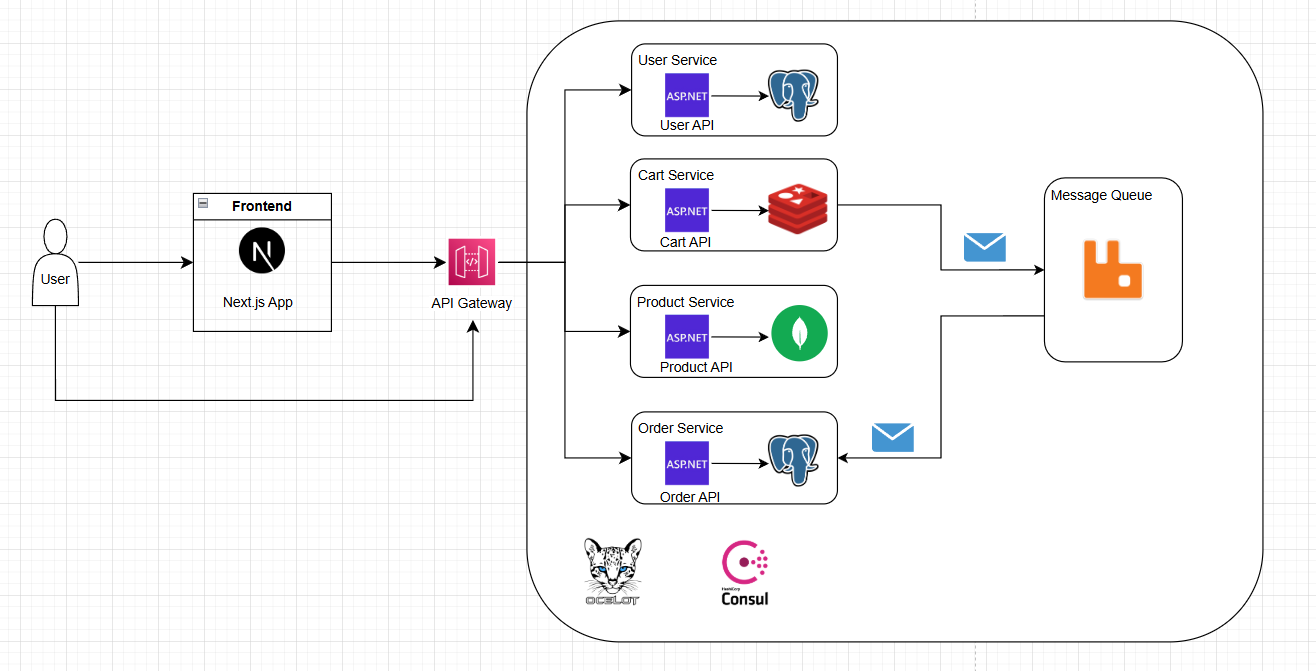
#### Quản lý sản phẩm



Hình .: Sơ đồ hoạt động quản lý hàng hóa

Sơ đồ hoạt động này sẽ hiển thị cách hàng hóa được quản trị viên quản lý, nhằm để quản lý được thông tin hàng hóa có trong cửa hàng, đảm bảo hàng hóa sẽ không bị quá nhiều, dư thừa hay không chất lượng.

## Kiến trúc hệ thống



Hình .: Kiến trúc hệ thống

Tuân thủ theo kiến trúc Microservice với mỗi chức năng sẽ tách biệt với từng Service, với từng Database riêng biệt và được kết nối với nhau thông qua API Gateway.

### User Service

Quản lý thông tin người dùng, bao gồm thông tin cá nhân, đăng nhập, đăng ký và bảo mật thông tin người dùng.

User Service giúp quản trị viên có thể quản lý người dùng nhanh chóng, đồng thời cũng giúp cho quản trị viên có thể đăng ký thông tin cho nhân viên.

### Cart Service

Quản lý giỏ hàng của mỗi người dùng trong hệ thống, giúp người dùng có thể lấy hàng hóa và thanh toán hóa đơn.

Cart Service sẽ ghi nhận hàng hóa người dùng thêm vào, sau khi đã mua, người dùng có thể chọn thanh toán và hàng hóa sẽ đến Order Service để tiến hành thanh toán

### Product Service

Quản lý hàng hóa và số lượng hàng hóa có trong hệ thống, đảm bảo hàng hóa có thông tin đúng như thông tin trên hàng hóa thật và số lượng luôn được đảm bảo để luôn có hàng hóa.

Product Service sẽ vừa hiển thị thông tin hàng hóa, vừa là nơi ghi nhận thông tin kho hàng để khi khách hàng thanh toán hàng hóa, hàng hóa trong kho sẽ tự động được lấy ra theo như thông tin hóa đơn của Order Service để giảm số lượng hàng hóa trong kho.

### Order Service

Quản lý thanh toán hàng hóa và phương thức thanh toán của người dùng, giúp người dùng sau khi mua hàng sẽ có hóa đơn và minh bạch trong việc mua hàng.

Order Service sẽ lấy trực tiếp thông tin hàng hóa trong Cart Service và tiến hành tạo hóa đơn, tính giá trị đơn hàng và xuất hóa đơn thanh toán. Nếu sử dụng thanh toán trực tuyến, Order Service sẽ hiển thị QR thông tin thanh toán như Momo hoặc VNPAY kèm giá tiền để thanh toán đơn hàng.

### Công cụ hỗ trợ

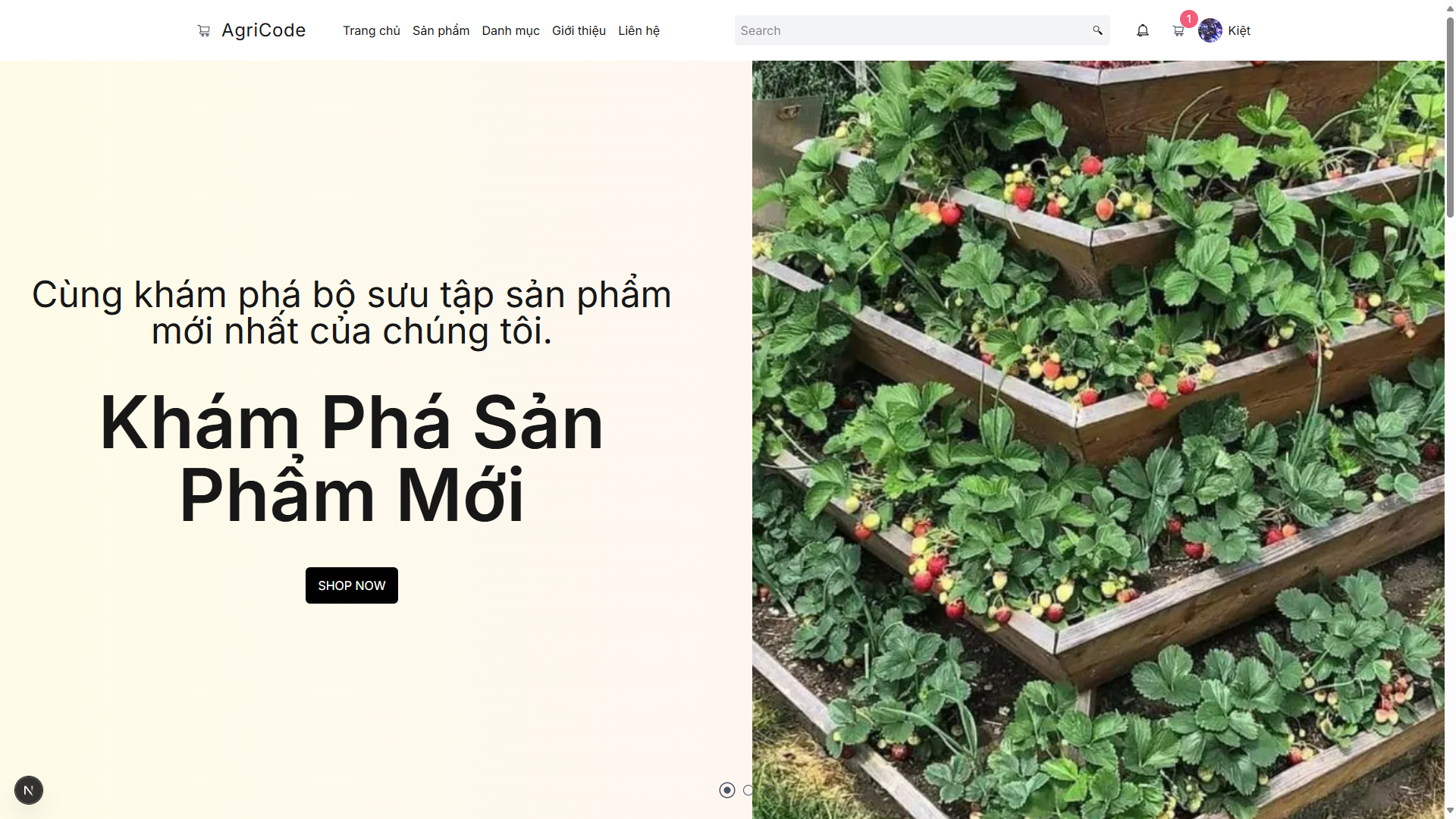
RabbitMQ: Đóng vai trò như một Message Broker giúp truyền thông tin giao tiếp với nhau trong hệ thống và phân tán Message tới dịch vụ tương ứng.

API Gateway: Điều hướng các API vào chung một cổng duy nhất và điều hướng Request của người dùng đến với các dịch vụ được gọi.

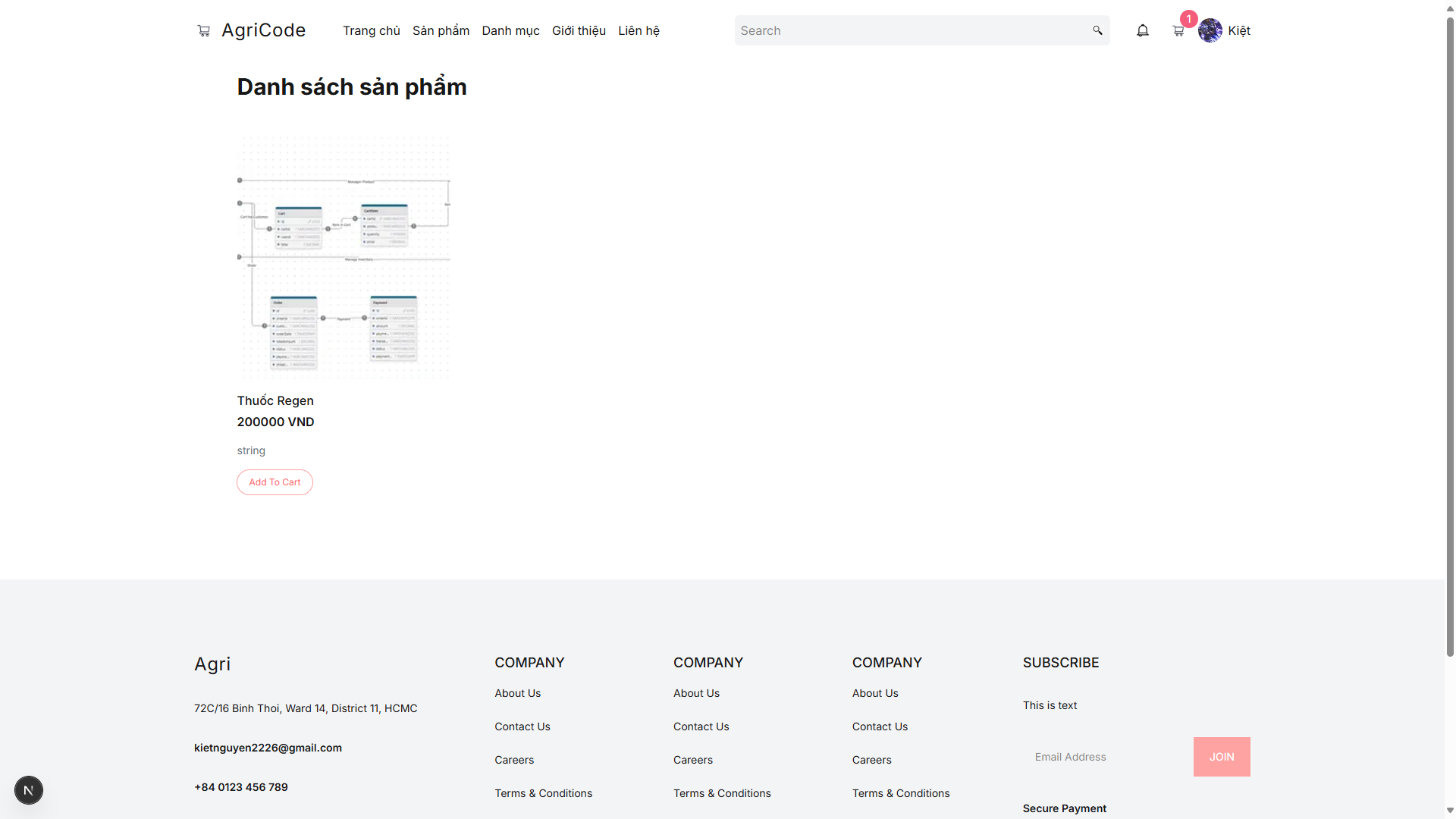
Service Discovery: Phát hiện dịch vụ trong hệ thống, đồng thời cân bằng tải.

Next.js: Hiển thị UI để tương tác với người dùng và hệ thống.

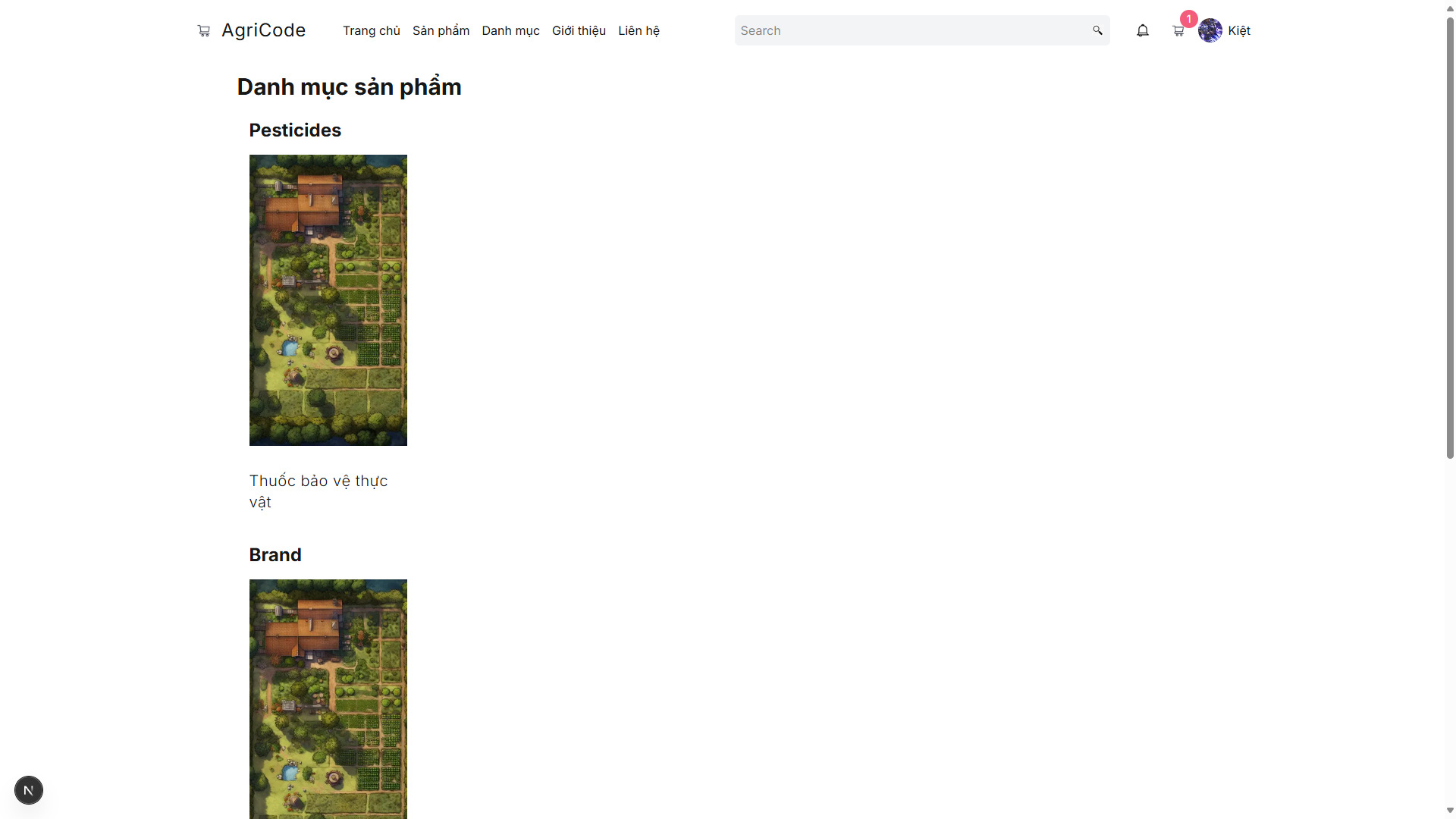
## Kết quả thực hiện



Hình .: Trang chủ website



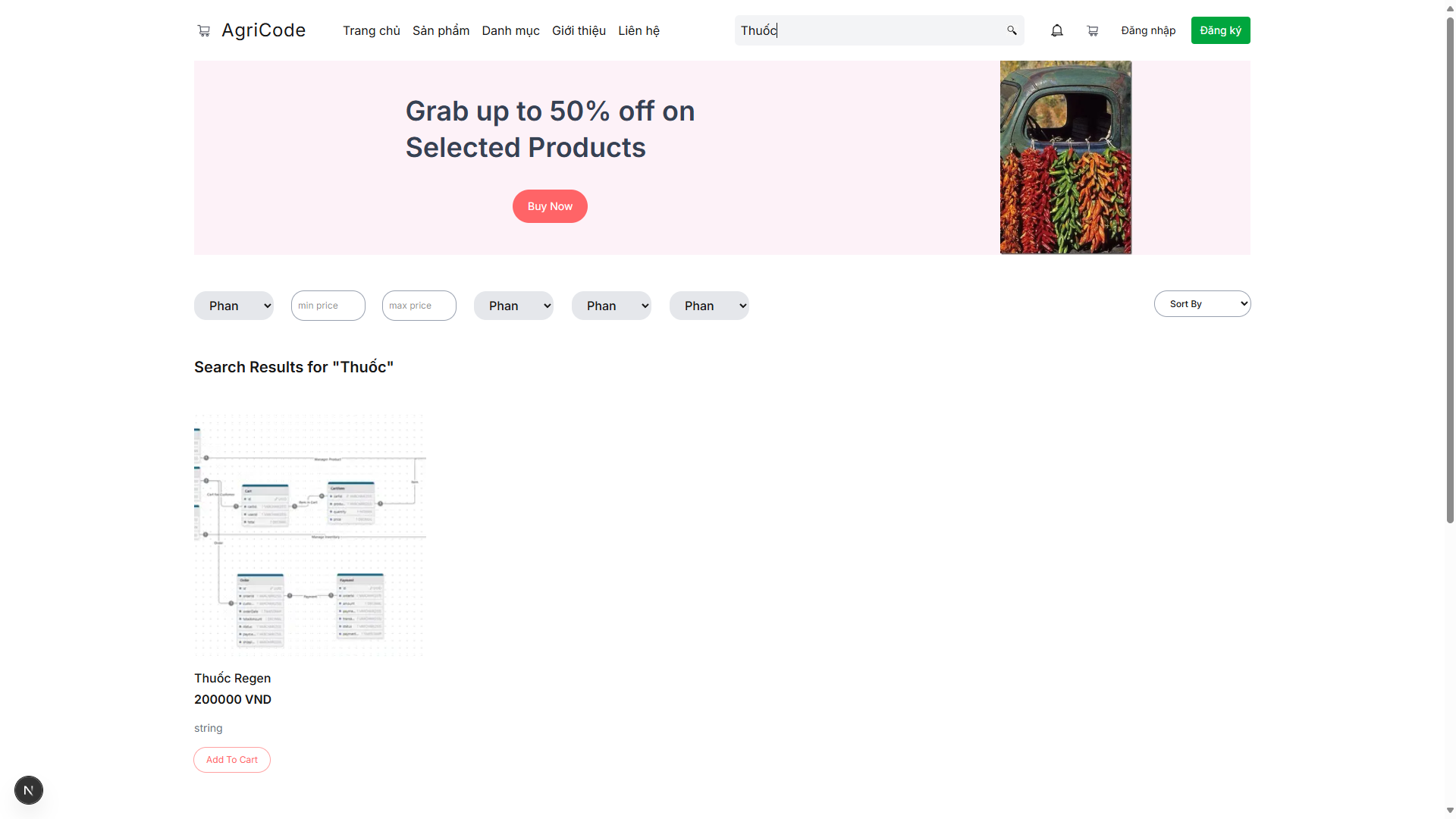
Hình .: Trang sản phẩm



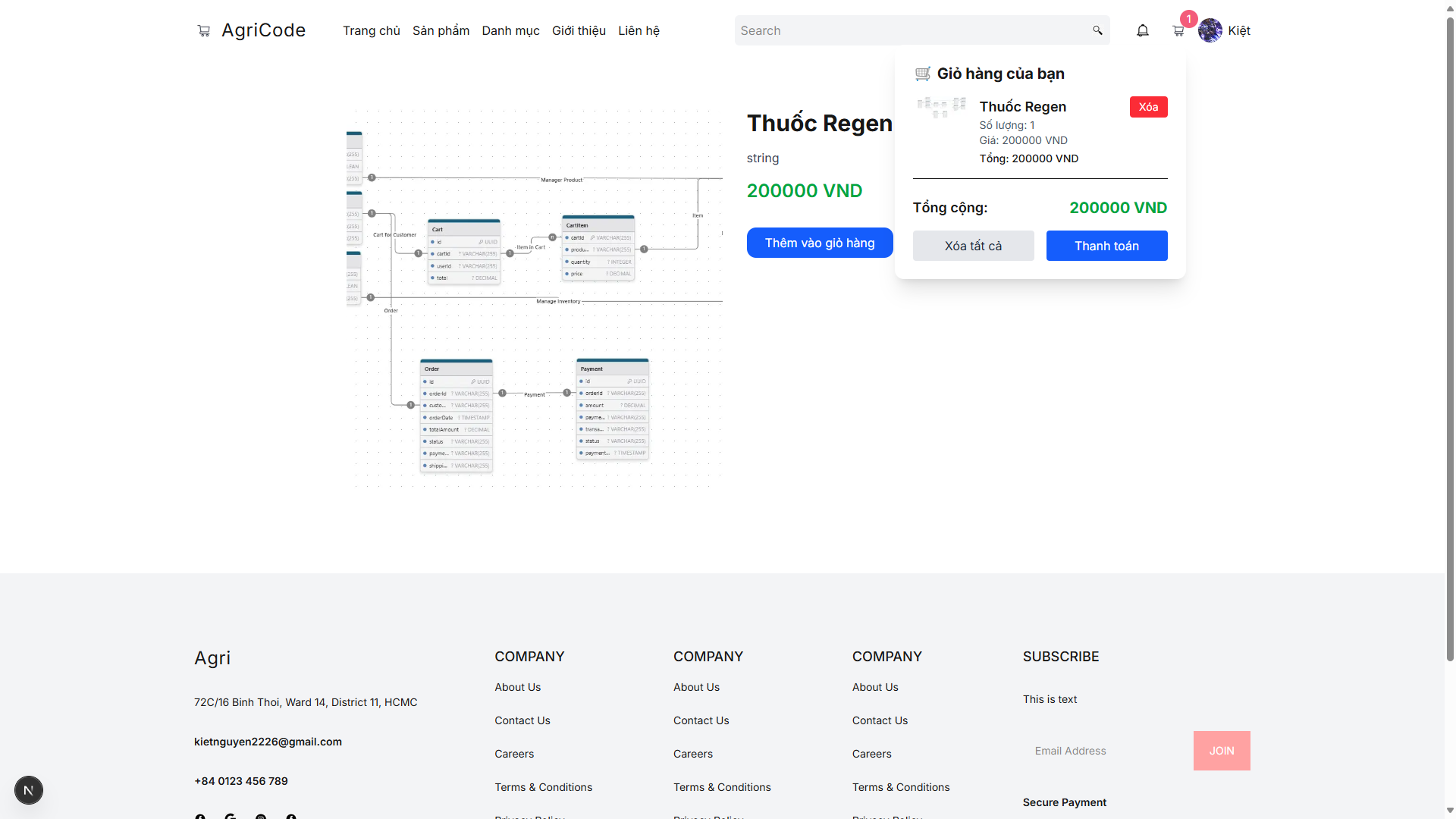
Hình .: Trang chứa danh mục



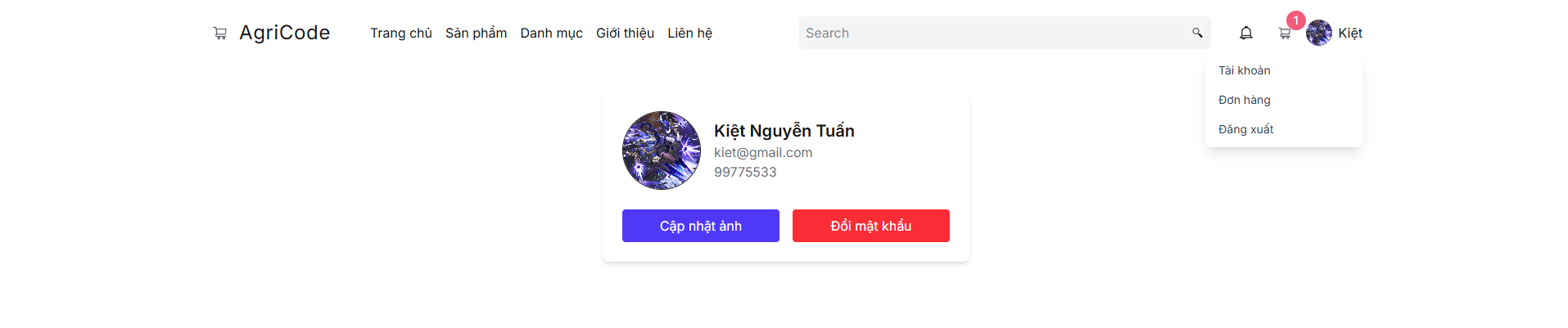
Hình .: Component Cart



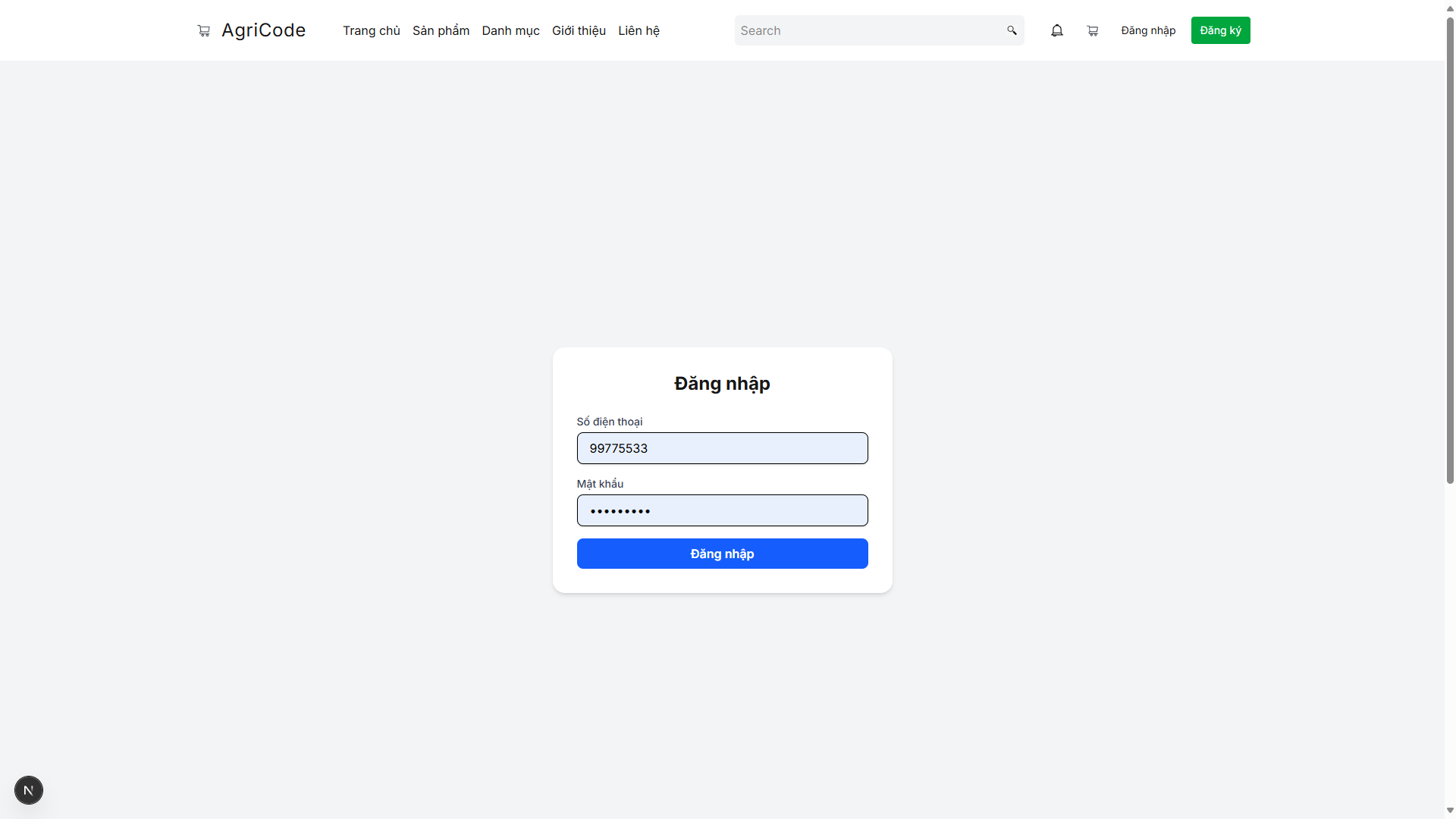
Hình .: Trang hiển thị sản phẩm khi tìm kiếm theo từ khóa



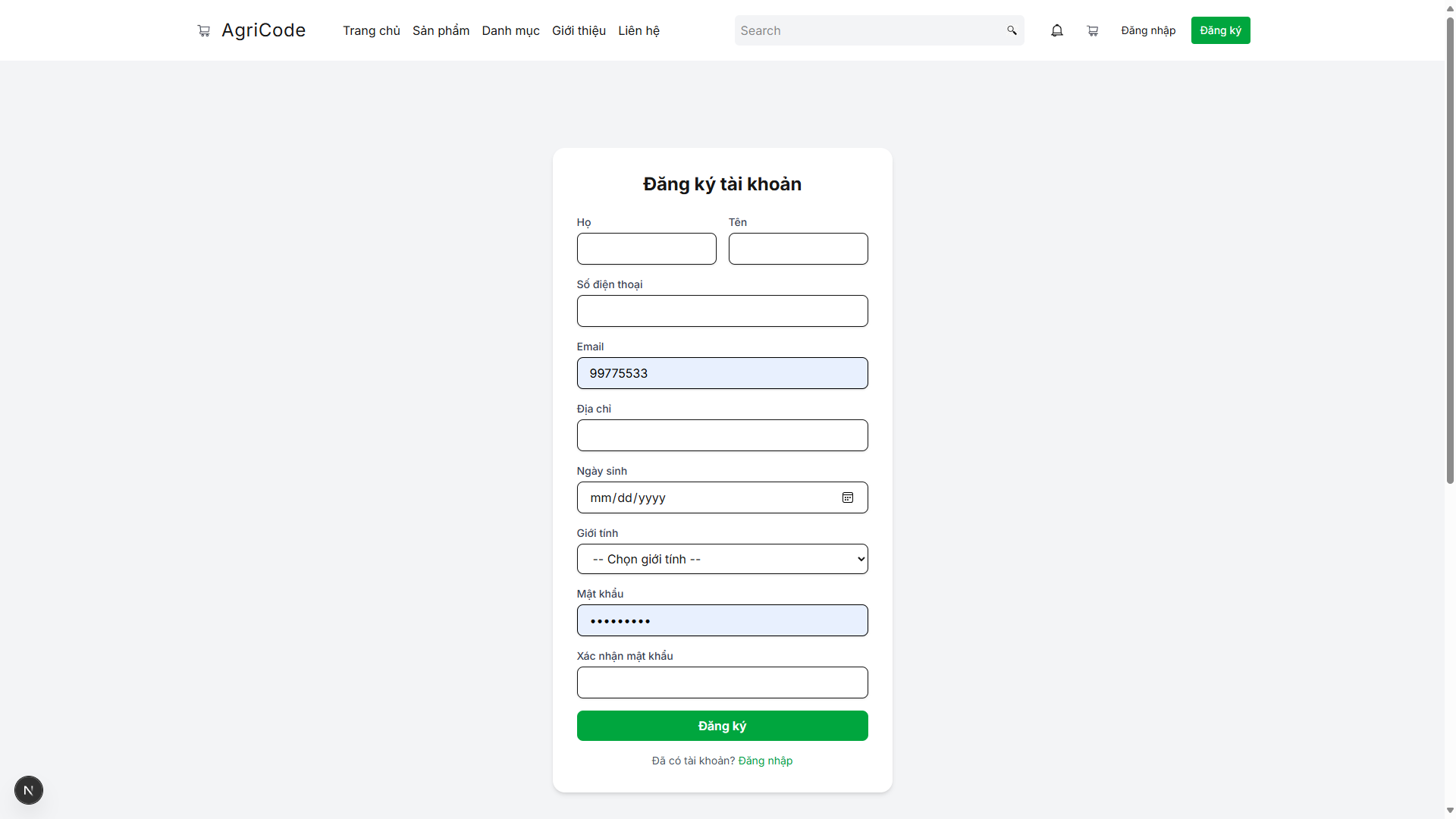
Hình .: Trang thông tin sản phẩm



Hình .: Trang và Modal hiển thị thông tin



Hình .: Trang hiển thị form đăng nhập



Hình .: Trang hiển thị form đăng ký

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

### Tổng quát

Sau quá trình phát triển và triển khai hệ thống bán nông sản, em đã đưa ra được những nhận xét tổng quan về hệ thống cũng như những ưu và nhược điểm của hệ thống.

**Đánh giá tổng quan về hệ thống:** Hệ thống bán hàng nông sản được xây dựng trên nền tảng Microservice với ASP.NET Core và NextJS, sử dụng các dịch vụ như API Gateway, Service Discovery và Message Queue. Nhờ vào việc phân chia hệ thống thành các Service độc lập, hệ thống có thể dễ dàng nâng cấp, bảo trì và mở rộng mà không ảnh hưởng đến các thành phần khác hay tổng thể hệ thống.

### Về ưu và nhược điểm

**Về ưu điểm:** Hệ thống đã tuân thủ theo quy tắc Microservice, các Service được phá triển độc lập nên có thể dễ dàng mở rộng và nâng cấp, các dịch vụ được chia nhỏ nên cũng dễ dàng quản lý, đồng thời có thể gắn nhiều loại cơ sở dữ liệu như PostgreSQL hay MongoDB cũng như nhiều dịch vụ bên thứ ba như Payment, API Gateway…

**Về nhược điểm:** Do hệ thống có nhiều Service nên việc quản lý cũng trở nên phức tạp hơn, đòi hỏi phải quan sát hệ thống liên tục để khi có lỗi sẽ phải sửa ngay, đồng thời do chạy nhiều dịch vụ cùng một lúc, máy chủ có thể sẽ bị dừng đột ngột nếu các Service lấy quá nhiều tài nguyên, kèm với đó cũng phải đảm bảo tính toàn vẹn của dữ liệu khi các Services giao tiếp với nhau trong hệ thống.

### Hạn chế

Do thời gian phát triển hệ thống có hạn, dự án còn gặp phải nhiều vấn đề liên quan đến hiệu năng, những lỗi chưa được xử lý kịp thời, những Service chưa thể được sử dụng hay vấn đề bảo mật đang còn gặp phải trong hệ thống, làm cho tổng quan hệ thống chưa được hoạt động tối ưu.

Dự án hiện tại chỉ được chạy trên môi trường của nhà phát triển, tức là chạy cục bộ, không được triển khai ở môi trường thực tế, nên việc không có lượng lớn người dùng trong hệ thống để kiểm tra và biết được độ chịu tải của hệ thống như thế nào.

## Hướng phát triển

Hệ thống sẽ còn cần cải thiện nhiều để tối ưu hơn trong việc triển khai như việc tích hợp thêm nhiều Service cần thiết, tích hợp CI/CD để triển khai và kiểm thử hệ thống đảm bảo luôn hoạt động ổn định, hệ thống theo giõi giúp kiểm tra hệ thống hoạt động ổn định hay có thể phát hiện được Service nào lỗi và dừng hoạt động.

Ngoài ra các dịch vụ Cloud như Microsoft Azure cũng là một lựa chọn vừa tối ưu cho hệ thống và cũng tối ưu cho việc triển khai thực tế sản phẩm nhằm giúp cho hệ thống có thể được triển khai và đến tay nhiều người dùng hơn.

TÀI LIỆU THAM KHẢO

[1] Microsoft Learn, “Overview of ASP.NET Core”, 19/06/2024. [Trực tuyến]. Địa chỉ: <https://learn.microsoft.com/vi-vn/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0>

[2] Vercel Inc, “Next.js Docs”, 10/04/2025. [Trực tuyến]. Địa chỉ: <https://nextjs.org/docs>

[3] Microsoft Learn, “Implement API Gateways with Ocelot”, 03/01/2023. [Trực tuyến]. Địa chỉ: <https://learn.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/implement-api-gateways-with-ocelot>