TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

LÝ THỊ HOÀNG MỸ - 51900137
GIẢN VIẾT ĐỨC – 51900586

XÂY DỰNG HỆ THỐNG QUẢN LÝ PHÂN PHỐI DƯỢC PHẨM

ÁP DỤNG KIẾN TRÚC MICROSERVICES TRÊN NỀN TẢNG JAVA SPRING BOOT

DỰ ÁN CNTT2

KỸ THUẬT PHẦN MỀM

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

LÝ THỊ HOÀNG MỸ - 51900137
GIẢN VIẾT ĐỨC – 51900586

XÂY DỰNG HỆ THỐNG QUẢN LÝ PHÂN PHỐI DƯỢC PHẨM ÁP DỤNG KIẾN TRÚC MICROSERVICES TRÊN NỀN TẢNG JAVA SPRING BOOT

DỰ ÁN CNTT2
KỸ THUẬT PHẦN MỀM

Người hướng dẫn
ThS. Võ Văn Thành

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Lời đầu tiên, chúng em muốn bày tỏ lòng biết ơn chân thành đến giảng viên hướng dẫn – ThS. Võ Văn Thành, thầy là người đã hướng dẫn tận tình và giải đáp những thắc mắc cho chúng em trong môn "Dự án Công nghệ thông tin 2". Nhờ vào những điều mà thầy đã định hướng và truyền đạt chúng em mới có thể hoàn thành dự án "***Xây dựng Hệ thống quản lý phân phối Dược phẩm áp dụng kiến trúc Microservices trên nền tảng Java Spring Boot***".

Tiếp theo, chúng em xin chân thành cảm ơn khoa Công Nghệ Thông Tin trường Đại học Tôn Đức Thắng đã cho chúng em cơ hội thực hiện môn học này. Đây là môn học hữu ích và quý báu trong thời gian chúng em học tập tại trường.

Cuối cùng, kiến thức là mênh mông và vô hạn nên dự án của chúng em khó tránh khỏi thiếu sót và hạn chế, mong rằng thầy sẽ góp ý để dự án trở nên hoàn thiện hơn.

Chúng em chân thành cảm ơn thầy!

TP. Hồ Chí Minh, ngày ... tháng … năm 20..

Tác giả

(Ký tên và ghi rõ họ tên)

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

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

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

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

TP. Hồ Chí Minh, ngày ... tháng ... năm 20..

Tác giả

(Ký tên và ghi rõ họ tên)

XÂY DỰNG HỆ THỐNG QUẢN LÝ PHÂN PHỐI DƯỢC PHẨM ÁP DỤNG KIẾN TRÚC MICROSERVICES TRÊN NỀN TẢNG JAVA SPRING BOOT

TÓM TẮT

Báo cáo này trình bày về quá trình phân tích, thiết kế và triển khai dự án xây dựng phần mềm quản lý phân phối dược phẩm áp dụng kiến trúc Microservices trên nền tảng Java Spring Boot. Dự án nhằm mục tiêu tạo ra một ứng dụng phần mềm hoàn chỉnh và ổn định để giải quyết nhu cầu cụ thể của doanh nghiệp phân phối dược phẩm.

Đầu tiên, báo cáo tập trung vào quá trình lập kế hoạch và phân tích yêu cầu của dự án. Qua việc tương tác, trao đổi và thảo luận với giảng viên hướng dẫn về các yêu cầu cụ thể của dự án để xác định lý do, mục tiêu, phương pháp nghiên cứu, quy trình phân phối dược phẩm trong thực tế và chức năng, nghiệp vụ cần phải có trong phần mềm.

Tiếp theo, báo cáo trình bày về các công nghệ, kiến trúc và framework xây dựng phần mềm, bao gồm cả kiến trúc Microservices và các công cụ, ngôn ngữ dùng để phát triển backend và frontend. Xác định các actor và use case trong hệ thống, đặc tả use case, vẽ activity diagram và thiết kế cơ sở dữ liệu cho từng service.

Sau đó, báo cáo đi sâu vào quá trình thực thi và triển khai hệ thống. Trong quá trình này, nhóm em đã thực hiện thiết kế giao diện người dùng bằng ReactJS và xây dựng backend và API bằng Java Spring Boot, Swagger, Eureka, PostgreSQL và một số công cụ khác. Sau đó thực hiện triển khai hệ thống, kết nối frontend và API đã được xây dựng.

Cuối cùng, báo cáo đánh giá kết quả của dự án và đề xuất các hướng phát triển trong tương lai.

BUILDING A PHARMACEUTICAL DISTRIBUTION MANAGEMENT SYSTEM TO APPLY MICROSERVICES ARCHITECTURE ON THE JAVA SPRING BOOT PLATFORM

ABSTRACT

This report presents the process of analysis, design and implementation of the pharmaceutical distribution management software project that applies Microservices architecture on the Java Spring Boot platform. The project aims to create a complete and stable software application to address the specific needs of the pharmaceutical distribution enterprises.

First, the report focuses on the planning process and analyzing the project's requirements. Through interacting, exchanging and discussing with lecturers guiding the specific requirements of the project to determine the reason, objectives, research methods, pharmaceutical distribution process in practice and functions , the business is required in the software.

Next, the report presented the technologies, architecture and framework to build software, including microservices architecture and tools and languages used to develop Backend and Frontend. Determine the actor and use case in the system, specify Use Case, draw Activity DiaGram and design database for each service.

After that, the report went into the process of implementation and deployment of the system. In this process, my group has designed user interface with ReactJs and built backend and API with Java Spring Boot, Swagger, Eureka, Postgresql and some other tools. Then implement the system, Frontend and API connectivity has been built.

Finally, the project results evaluate report and propose future development directions.

MỤC LỤC

DANH MỤC HÌNH VẼ ..... **x**

DANH MỤC BẢNG BIỂU ..... **xiv**

DANH MỤC CÁC CHỮ VIẾT TẮT ..... **xv**

CHƯƠNG 1. MỞ ĐẦU **1**

1.1 Lý do chọn đề tài 1

1.2 Mục tiêu thực hiện đề tài 2

1.3 Đối tượng và phạm vi nghiên cứu 2

1.3.1 Đối tượng nghiên cứu *2*

1.3.2 Phạm vi nghiên cứu *2*

1.4 Phương pháp nghiên cứu 3

1.4.1 Nghiên cứu lý thuyết *3*

1.4.2 Nghiên cứu thực nghiệm *3*

1.4.3 Phương pháp đánh giá *3*

1.5 Ý nghĩa nghiên cứu 4

CHƯƠNG 2. TỔNG QUAN **5**

2.1 Khảo sát thực trạng quản lý chuỗi cung ứng dược phẩm hiện nay 5

2.1.1 Tổng quan *5*

2.1.2 Thách thức *6*

2.2 Giải pháp 7

2.3 Quy trình lập kế hoạch thực hiện dự án 7

2.4 Chức năng nghiệp vụ 10

2.4.1 Chức năng người dùng *10*

2.4.2 Chức năng người quản trị *11*

2.4.3 Yêu cầu hệ thống *12*

2.4.4 Lựa chọn giải pháp và công nghệ *14*

2.5 Kết quả mong muốn đạt được 14

CHƯƠNG 3. TÌM HIỂU CÔNG NGHỆ THỰC HIỆN 16

3.1 Front-end 16

3.1.1 ReactJS *16*

3.1.2 Ưu điểm và nhược điểm của ReactJS *17*

3.2 Back-end 18

3.2.1 Java Spring Boot, Spring Cloud, Spring Security, Spring Data JPA *18*

3.2.2 Kiến trúc Microservices *19*

3.2.3 Rabbit MQ *20*

3.2.4 Eureka *21*

3.2.5 Redis *22*

3.3 Cơ sở dữ liệu 23

3.3.1 PostgreSQL *23*

3.3.2 Ưu điểm và nhược điểm của PostgreSQL *23*

3.4 API 24

3.4.1 Swagger và Swagger UI *24*

3.4.2 Ưu điểm và nhược điểm của Swagger và Swagger UI *26*

CHƯƠNG 4. THIẾT KẾ HỆ THỐNG 27

4.1 Cấu trúc của hệ thống 27

4.2 Sơ đồ cấu trúc của hệ thống 28

4.3 Mô tả nghiệp vụ 28
4.3.1 Super Admin 28
4.3.2 Admin 29
4.3.3 Warehouse Manager 30
4.3.4 Transportation Staff 30
4.3.5 User 30
4.3.6 Quy trình nhập hàng 31
4.3.7 Quy trình bán hàng 32
4.4 Sơ đồ Use case 33
4.4.1 Xác định các Use case 33
4.4.2 Sơ đồ Use case tổng quát 37
4.5 Đặc tả các Use case 39
4.5.1 Use case thêm người dùng 39
4.5.2 Use case chấp nhận lời mời và thêm thông tin người dùng 41
4.5.3 Use case quên mật khẩu 43
4.5.4 Use case tìm kiếm sản phẩm bằng filter 47
4.5.5 Use case thêm sản phẩm mới 49
4.5.6 Use case xem danh sách sản phẩm 51
4.5.7 Use case xem chi tiết sản phẩm 53
4.5.8 Use case sửa thông tin sản phẩm 55
4.5.9 Use case xóa sản phẩm 57
4.5.10 Use case thanh toán đơn hàng 59
4.6 Thiết kế cơ sở dữ liệu 62

4.6.1 Cơ sở dữ liệu của user service *62*

4.6.2 Cơ sở dữ liệu của product service *62*

4.6.3 Cơ sở dữ liệu của customer service *63*

4.6.4 Cơ sở dữ liệu của order service *64*

4.6.5 Cơ sở dữ liệu của warehouse service *65*

CHƯƠNG 5. THỰC THI HỆ THỐNG 66

5.1 Phần giao diện 66

5.1.1 Giao diện thêm người dùng *66*

5.1.2 Giao diện chấp nhận lời mời và thêm thông tin *66*

5.1.3 Giao diện quên mật khẩu *67*

5.1.4 Giao diện tìm kiếm và xem danh sách sản phẩm *69*

5.1.5 Giao diện thêm sản phẩm mới *69*

5.1.6 Giao diện xem chi tiết sản phẩm *70*

5.1.7 Giao diện thanh toán đơn hàng *70*

5.2 Phần hệ thống 71

5.2.1 User Service *71*

5.2.2 Product Service *73*

5.2.3 Customer Service *75*

5.2.4 Order Service *78*

5.2.5 Warehouse Service *81*

CHƯƠNG 6. TRIỂN KHAI HỆ THỐNG 84

6.1 Eureka 84

6.2 Triển khai trên Docker 84

6.3 Ngrok 85

6.4 Kết nối front-end vs API 85

CHƯƠNG 7. KẾT LUẬN 86

7.1 Kết luận 86

7.1.1 Những vấn đề đạt được 86

7.1.2 Những vấn đề chưa đạt được 86

7.1.3 Thuận lợi và khó khăn 86

7.2 Hướng phát triển trong tương lai 87

TÀI LIỆU THAM KHẢO 88

DANH MỤC HÌNH VẼ

Hình 3.1: Giao diện Facebook 16
Hình 3.2: Giao diện Instagram 17
Hình 3.3: Giao diện của Swagger và Swagger UI 25
Hình 3.4: Giao diện xem chi tiết API trên Swagger UI 25
Hình 4.1: Sơ đồ cấu trúc của hệ thống 28
Hình 4.2: Sơ đồ mô tả quy trình nhập hàng 31
Hình 4.3: Sơ đồ mô tả quy trình bán hàng 32
Hình 4.4: Sơ đồ Use Case tổng quát 1 37
Hình 4.5: Sơ đồ Use Case tổng quát 2 38
Hình 4.6: Sequence diagram của UC01 40
Hình 4.7: Activity diagram của UC01 41
Hình 4.8: Sequence diagram của UC02 42
Hình 4.9: Activity diagram của UC02 43
Hình 4.10: Sequence diagram của UC05 45
Hình 4.11: Activity diagram của UC05 46
Hình 4.12: Sequence diagram của UC18 48
Hình 4.13: Activity diagram của UC18 48
Hình 4.14: Sequence diagram của UC16 50
Hình 4.15: Activity diagram của UC16 51
Hình 4.16: Sequence diagram của UC17 52
Hình 4.17: Activity diagram của UC17 52
Hình 4.18: Sequence diagram của UC19 54

Hình 4.19: Activity diagram của UC19 54
Hình 4.20: Sequence diagram của UC20 56
Hình 4.21: Activity diagram của UC20 57
Hình 4.22: Sequence diagram của UC21 58
Hình 4.23: Activity diagram của UC21 59
Hình 4.24: Sequence diagram của UC30 60
Hình 4.25: Activity diagram của UC30 61
Hình 4.26: Cơ sở dữ liệu của User Service 62
Hình 4.27: Cơ sở dữ liệu của Product Service 62
Hình 4.28: Cơ sở dữ liệu của Product Service 63
Hình 4.29: Cơ sở dữ liệu của Order Service 64
Hình 4.30: Cơ sở dữ liệu của Warehouse Service 65
Hình 5.1: Giao diện thêm người dùng 66
Hình 5.2: Giao diện nhận lời mời qua mail 66
Hình 5.3: Giao diện thêm thông tin người dùng 67
Hình 5.4: Giao diện quên mật khẩu 1 67
Hình 5.5: Giao diện quên mật khẩu 2 68
Hình 5.6: Giao diện quên mật khẩu 3 68
Hình 5.7: Giao diện tìm kiếm và xem danh sách sản phẩm 69
Hình 5.8: Giao diện thêm sản phẩm mới 69
Hình 5.9: Giao diện xem chi tiết sản phẩm 70
Hình 5.10: Giao diện thanh toán đơn hàng 1 70
Hình 5.11: Giao diện thanh toán đơn hàng 2 71

Hình 5.12: API của User Service (hình 1)..... 71
Hình 5.13: API của User Service (hình 2)..... 72
Hình 5.14: API của User Service (hình 3)..... 72
Hình 5.15: API get user profile - Request..... 72
Hình 5.16: API get user profile - Response..... 73
Hình 5.17: API của Product Service..... 74
Hình 5.18: API delete product - Request..... 74
Hình 5.19: API delete product - Response..... 75
Hình 5.20: API của Customer Service (hình 1)..... 76
Hình 5.21: API của Customer Service (hình 2)..... 76
Hình 5.22: API get customer detail by id - Request..... 76
Hình 5.23: API get customer detail by id - Response..... 77
Hình 5.24: API của Order Service (hình 1)..... 78
Hình 5.25: API của Order Service (hình 2)..... 78
Hình 5.26: API của Order Service (hình 3)..... 79
Hình 5.27: API get all orders - Request..... 79
Hình 5.28: API get all orders - Response..... 80
Hình 5.29: API của Warehouse Service (hình 1)..... 81
Hình 5.30: API của Warehouse Service (hình 2)..... 81
Hình 5.31: API của Warehouse Service (hình 3)..... 82
Hình 5.32: API get all product in warehouse - Request..... 83
Hình 5.33: API get all product in warehouse - Response..... 83
Hình 6.1: Những Service được đăng ký bằng Eureka..... 84

Hình 6.2: Triển khai Redis trên Docker 84

Hình 6.3: Triển khai PostgresSQL trên Docker..... 84

Hình 6.4: Triển khai RabbitMQ trên Docker..... 84

Hình 6.5: Màn hình sau khi đã triển khai trên Docker..... 85

Hình 6.6: Chạy Ngrok để kết nối API với front-end..... 85

Hình 6.7: Config domain của back-end ở front-end 85

DANH MỤC BẢNG BIỂU

Bảng 2.1: Bảng tóm tắt quy trình lập kế hoạch thực hiện dự án 7

Bảng 2.2: Bảng tóm tắt yêu cầu về phần mềm 12

Bảng 4.1: Các use case trong hệ thống..... 33

Bảng 4.2: Đặc tả use case thêm người dùng..... 39

Bảng 4.3: Đặc tả use case chấp nhận lời mời và thêm thông tin người dùng 41

Bảng 4.4: Đặc tả use case quên mật khẩu 43

Bảng 4.5: Đặc tả use case tìm kiếm bằng filter 47

Bảng 4.6: Đặc tả use case thêm sản phẩm mới..... 49

Bảng 4.7: Đặc tả use case xem danh sách sản phẩm..... 51

Bảng 4.8: Đặc tả use case xem chi tiết sản phẩm 53

Bảng 4.9: Đặc tả use case sửa thông tin sản phẩm 55

Bảng 4.10: Đặc tả use case đặt xóa sản phẩm 57

Bảng 4.11: Đặc tả use case thanh toán đơn hàng..... 59

DANH MỤC CÁC CHỮ VIẾT TẮT

AI	Artificial Intelligence (Trí tuệ nhân tạo)
BE	Back end
EMA	European Medicines Agency (Cơ quan Dược phẩm Châu Âu)
FDA	Food and Drug Administration (Cục Quản lý Thực phẩm và Dược phẩm)
FE	Front end
GPS	Global Positioning System (Hệ thống Định vị Toàn cầu)
JSON	JavaScript Object Notation
JSX	JavaScript XML
JWT	JSON Web Token
OTP	One-Time Password
SEO	Search Engine Optimization
SQL	Structured Query Language
SSL	Secure Sockets Layer
UI	User Interface
URI	Uniform Resource Identifier
XML	eXtensible Markup Language

CHƯƠNG 1. MỞ ĐẦU

1.1 Lý do chọn đề tài

Trong bối cảnh hiện đại hóa xã hội, các hệ thống và phần mềm áp dụng thành tựu công nghệ thông tin tham gia vào mọi mặt của đời sống, điều này đã tạo ra những giá trị to lớn góp phần thúc đẩy xã hội tiến bộ và phát triển hơn. Vì vậy, việc xây dựng một hệ thống quản lý phân phối dược phẩm hiệu quả, đáp ứng tốt nhu cầu của doanh nghiệp là rất quan trọng.

Kiến trúc Microservices áp dụng sự phân tán dịch vụ, độc lập, nhằm mục đích triển khai và mở rộng riêng biệt. Điều này giúp tăng tính linh hoạt, dễ dàng bảo trì và nâng cấp từng phần của hệ thống mà không ảnh hưởng đến các phần khác. Nó dần trở thành một xu hướng được ưa chuộng trong việc phát triển và xây dựng các ứng dụng phân tán, đáp ứng tốt nhu cầu mở rộng và nâng cấp hệ thống.

Hệ thống này bao gồm các tính năng chính nhằm quản lý tốt các hoạt động trong quy trình phân phối dược phẩm như quản lý người dùng, quản lý sản phẩm, quản lý khách hàng, quản lý đơn hàng, quản lý kho, quản lý vận chuyển,... Xây dựng hệ thống với kiến trúc Microservice trên nền tảng Java Spring Boot giúp phát triển nhanh chóng, hiệu quả hơn, dễ dàng sửa đổi và nâng cấp.

Bên cạnh đó, đồ án sẽ đưa ra những vấn đề khó khăn đã gặp phải trong quá trình xây dựng, cũng như các giải pháp được đề xuất. Cuối cùng, sẽ đánh giá hiệu suất và tính khả thi của hệ thống, cùng với những đề xuất và hướng phát triển trong tương lai.

Bài báo cáo nhằm cung cấp một nghiên cứu thực tế về quá trình xây dựng phần mềm với kiến trúc Microservices trong xây dựng hệ thống quản lý phân phối dược phẩm bằng Java Spring Boot, qua đó nâng cao hiệu quả hoạt động kinh doanh và trải nghiệm của người dùng.

1.2 Mục tiêu thực hiện đề tài

Đề tài "Xây dựng hệ thống quản lý phân phối dược phẩm áp dụng kiến trúc Microservices trên nền tảng Java Spring Boot" có những mục tiêu chính sau đây:

- Xây dựng hệ thống với đầy đủ tính năng quản lý quy trình phân phối dược phẩm như quản lý người dùng, quản lý sản phẩm, quản lý khách hàng, quản lý đơn hàng, quản lý kho, quản lý vận chuyển.
- Xây dựng hệ thống dựa trên kiến trúc Microservices với nền tảng Java Spring Boot, chia hệ thống thành các service độc lập, tăng tính linh hoạt, đảm bảo khả năng bảo trì và mở rộng.
- Sau khi hoàn thành, thực hiện đánh giá khả năng áp dụng vào thực tiễn và hiệu suất của hệ thống, từ đó làm cơ sở để đề xuất hướng phát triển trong tương lai. Hỗ trợ nâng cao hiệu quả kinh doanh và trải nghiệm của người dùng.

1.3 Đối tượng và phạm vi nghiên cứu

1.3.1 Đối tượng nghiên cứu

Đối tượng nghiên cứu chính của đề tài là hệ thống quản lý phân phối dược phẩm, bao gồm các tính năng quản lý quy trình phân phối dược phẩm của doanh nghiệp.

Bên cạnh đó, đề tài cũng nghiên cứu về kiến trúc Microservices, nền tảng Java Spring Boot và các công nghệ, frameworks khác để áp dụng vào thiết kế và xây dựng hệ thống.

1.3.2 Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài bao gồm:

- Thiết kế kiến trúc hệ thống theo mô hình MicroServices, xác định các service cần thiết và sự tương tác qua lại giữa các service.

- Xây dựng và triển khai hệ thống với kiến trúc Microservices trên nền tảng Java Spring Boot, gồm các service sau: User Service, Product Service, Customer Service, Order Service, Warehouse Service, Shipment Service. Giao diện người dùng trên web và được hiện thực bằng ReactJS.
- Nghiên cứu các công nghệ và frameworks để giải quyết các vấn đề liên quan đến giao tiếp giữa các Service, phân vùng dữ liệu, tính bảo mật và tính mở rộng.

Có một điều cần lưu ý rằng phạm vi nghiên cứu của nhóm tập trung vào khía cạnh kỹ thuật liên quan đến việc xây dựng hệ thống theo kiến trúc Microservices trên nền tảng Java Spring Boot, các vấn đề về khía cạnh kinh doanh, vận hành, quản lý chỉ mang tính chất tham khảo không nằm trong phạm vi nghiên cứu.

1.4 Phương pháp nghiên cứu

1.4.1 Nghiên cứu lý thuyết

Lý thuyết được nghiên cứu bằng phương pháp tìm hiểu thông qua tài liệu, các công trình nghiên cứu liên quan đến kiến trúc Microservices, mô hình thiết kế phân tán, nền tảng Java Spring Boot và các công nghệ, frameworks, ngôn ngữ liên quan như: ReactJS, PostgreSQL, Swagger, RabbitMQ,...

1.4.2 Nghiên cứu thực nghiệm

Thiết kế kiến trúc hệ thống theo mô hình Microservices, xác định các service cần thiết và sự tương tác giữa các service.

Xây dựng, triển khai kiến trúc Microservices trên nền tảng Java Spring Boot cho các chức năng quản lý quy trình phân phối dược phẩm.

Áp dụng các công nghệ, framworks vào hệ thống để giải quyết các vấn đề về giao tiếp giữa các service, phân quyền, phân vùng dữ liệu và bảo mật hệ thống.

1.4.3 Phương pháp đánh giá

Kiểm tra hiệu suất xử lý của phần mềm và khả năng áp dụng vào thực tế, kiểm tra thời gian phản hồi, khả năng chịu tải, độ tin cậy, khả năng bảo trì và khả năng mở rộng.

Tiến hành đánh giá bằng cách triển khai và vận hành hệ thống trong môi trường thử nghiệm.

1.5 Ý nghĩa nghiên cứu

Dự án này tiếp cận hướng xây dựng hệ thống một cách hiện đại, ứng dụng kiến trúc Microservices giúp tăng tính linh hoạt khi xây dựng hệ thống và tăng khả năng bảo trì và mở rộng hệ thống.

Dự án là minh chứng cho khả năng áp dụng thực tế của kiến trúc Microservices vào một lĩnh vực của đời sống xã hội.

Giúp nhóm tăng thêm kinh nghiệm thực tiễn trong việc sử dụng kiến trúc Microservices, nền tảng Java Spring Boot và các công cụ, frameworks một cách hiệu quả.

Bên cạnh các ý nghĩa liên quan đến đối tượng nghiên cứu chính, thông qua dự án nhóm đã được tìm hiểu thêm về lĩnh vực phân phối dược phẩm, quy trình quản lý và vận hành trong lĩnh vực này. Điều này giúp nhóm có thêm hiểu biết về lĩnh vực mới và rèn luyện kỹ năng tìm hiểu, phân tích vấn đề.

CHƯƠNG 2. TỔNG QUAN

2.1 Khảo sát thực trạng quản lý chuỗi cung ứng dược phẩm hiện nay

2.1.1 Tổng quan

Các thành phần chính trong chuỗi cung ứng dược phẩm:

- Nhà sản xuất dược phẩm (Manufacturers): đây là thành phần sản xuất các loại dược phẩm, có thể là các công ty dược phẩm lớn (như Pfizer, Johnson & Johnson, Novartis,...) hoặc các công ty sản xuất dược phẩm nhỏ hơn.
- Nhà cung cấp nguyên liệu (Suppliers): cung cấp các nguyên liệu thô cho quá trình sản xuất dược phẩm, các nguyên liệu bao gồm các thành phần hoạt chất của dược phẩm, tá dược,...và vật liệu để đóng gói sản phẩm.
- Nhà phân phối (Distributors): đóng vai trò trung gian, phân phối sản phẩm từ nhà sản xuất đến các địa điểm bán lẻ (như nhà thuốc, bệnh viện,...).
- Nhà bán lẻ (Retailers), nhà thuốc (Pharmacy) và cơ sở y tế (Healthcare Providers): có thể kể đến các hiệu thuốc, chuỗi nhà thuốc, cửa hàng trực tuyến hoặc bệnh viện, phòng khám. Chịu trách nhiệm bán thuốc trực tiếp cho người tiêu dùng cuối cùng.

Quy trình phân phối dược phẩm diễn ra như sau:

Nghiên cứu và phát triển dược phẩm: thực hiện nghiên cứu, thử nghiệm lâm sàng để phát triển thuốc và xin cấp phép từ các cơ quan quản lý dược phẩm như FDA hoặc EMA.

Sản xuất: sản xuất nguyên liệu, phối chế thuốc và đóng gói sản phẩm. Giai đoạn này đòi hỏi kiểm tra chất lượng nghiêm ngặt để đảm bảo an toàn dược phẩm và đạt hiệu quả tốt.

Kiểm tra chất lượng dược phẩm: Kiểm tra chất lượng của các mẫu dược phẩm trước khi phân phối ra thị trường nhằm đảm bảo thuốc đáp ứng được các tiêu chuẩn và quy định an toàn dược phẩm.

Vận chuyển và bảo quản: thuốc cần được bảo quản trong điều kiện nhiệt độ và độ ẩm thích hợp để đảm bảo chất lượng. Quá trình vận chuyển cũng phải được quản lý để đảm bảo thuốc đến đúng thời hạn và không xảy ra sai sót.

Phân phối và bán lẻ: sản phẩm được nhà phân phối đưa từ nhà sản xuất đến các cửa hàng bán lẻ hoặc các cơ sở y tế, cần quản lý tốt hàng tồn kho để đảm bảo không bị thiếu hụt hoặc lãng phí.

2.1.2 Thách thức

Quy trình phân phối dược phẩm thủ công thường phải đối mặt với những thách thức, khó khăn từ tác động của môi trường hoặc từ công nghệ và con người. Đầu tiên phải kể đến là điều kiện bảo quản,đa số dược phẩm có điều kiện bảo quản tương đối bình thường như nhiệt độ phòng (trong khoảng 20 - 25 độ C), độ ẩm thấp và tránh ánh nắng trực tiếp, tuy nhiên có một số loại thuốc yêu cầu phải bảo quản ở điều kiện đặc biệt (2 - 8 độ C). Vì vậy, cần có một hệ thống tiến hành thống kê, quản lý và giám sát để dược phẩm luôn nằm trong điều kiện lưu trữ và vận chuyển thích hợp.

Tiếp theo, trong quá trình phân phối dược phẩm cần quản lý vận chuyển để giảm thiểu thời gian cho quá trình giao hàng và chi phí vận chuyển, đồng thời phải đảm bảo tính toàn vẹn của sản phẩm, tránh hư hỏng hoặc thất thoát. Vì vậy, cần tối ưu hóa tuyến đường và lịch trình vận chuyển trong quá trình phân phối.

Bên cạnh đó, các sản phẩm dược được phân phối cần đáp ứng các tiêu chuẩn nghiêm ngặt của cơ quan có thẩm quyền như FDA (Hoa Kỳ), EMA (Châu Âu) và các cơ quan tương đương của mỗi quốc gia. Phải đảm bảo khả năng truy xuất nguồn gốc của từng lô hàng, quản lý được thông tin về xuất xứ và quá trình xử lý của sản phẩm.

Cuối cùng, các tác động từ môi trường như thiên tai, dịch bệnh có thể làm gián đoạn chuỗi cung ứng, nhà phân phối cần có kế hoạch dự phòng và chiến lược quản lý rủi ro để giảm tác động từ thiên tai xuống mức thấp nhất.

2.2 Giải pháp

Đối mặt với các thách thức trên, có một số giải pháp có thể cân nhắc. Đầu tiên, với vấn đề về điều kiện bảo quản chúng ta có thể giải quyết bằng hệ thống quản lý sản phẩm và tích hợp hệ thống IoT để quản lý và cảnh báo khi lô hàng không được bảo quản trong điều kiện thích hợp.

Tiếp theo, về vấn đề quản lý vận chuyển nhằm giảm thời gian và chi phí đồng thời đảm bảo tính toàn vẹn của sản phẩm, chúng ta có thể tạo ra một phần mềm quản lý kho và quản lý vận chuyển tích hợp với hệ thống GPS để theo dõi và điều chỉnh lộ trình thích hợp. Có thể tích hợp AI hoặc machine learning để phân tích lịch sử và nhu cầu sản phẩm và điều chỉnh số lượng hàng trong kho để đảm bảo luôn có đủ hàng mà không bị dư thừa.

Trong việc tuân thủ quy định về an toàn và truy xuất nguồn gốc dược phẩm, chúng ta có thể sử dụng Blockchain để ghi lại tất cả các bước trong chuỗi cung ứng, từ lúc sản xuất đến lúc phân phối đến nhà thuốc và cơ sở y tế. Bên cạnh đó, cần có hệ thống có chức năng lưu trữ các chứng nhận chất lượng và kiểm định của từng lô hàng để đảm bảo tính hợp pháp và an toàn cho người sử dụng.

2.3 Quy trình lập kế hoạch thực hiện dự án

Bảng 2.1: Bảng tóm tắt quy trình lập kế hoạch thực hiện dự án

Giai đoạn	Công việc chính	Công việc chi tiết
1. Xác định yêu cầu và phạm vi của dự án	1.1 Khảo sát và thu thập yêu cầu	- Tìm hiểu tài liệu liên quan đến quy trình phân phối dược phẩm. - Thảo luận để hiểu rõ nhu cầu và yêu cầu cụ thể của hệ thống. - Tạo tài liệu yêu cầu chức năng cho hệ thống.
	1.2 Phân tích và xác định phạm vi dự án	- Phân tích yêu cầu và xác định phạm vi thực hiện của dự án.

Giai đoạn	Công việc chính	Công việc chi tiết
		- Tạo danh sách các chức năng chính và phụ. - Tạo tài liệu yêu cầu chi tiết.
2. Lập kế hoạch dự án	2.1 Lập nhóm dự án	- Xác định và phân công vai trò của các thành viên trong nhóm. - Lập kênh giao tiếp. - Thiết lập công cụ quản lý dự án (Github, Jira).
	2.2 Lập kế hoạch thời gian thực hiện dự án	- Xác định các mốc thời gian chính của dự án. - Sử dụng Jira để chia nhỏ và quản lý công việc thành các giai đoạn nhỏ.
3. Thiết kế kiến trúc hệ thống	3.1 Thiết kế hệ thống dựa trên kiến trúc Microservices	- Xác định các service chính của hệ thống. - Thiết kế các API cho từng service và cách giao tiếp giữa các service.
	3.2 Thiết kế cơ sở dữ liệu	- Thiết kế mô hình dữ liệu cho từng service. - Thiết kế các bảng dữ liệu và xác định mối quan hệ giữa chúng. - Thêm dữ liệu cho các bảng dữ liệu.
	3.3 Thiết kế giao diện người dùng (UI/UX)	- Xác định các component chính và giao diện người dùng. - Tạo Mockup cho các màn hình chính.
4. Phát triển hệ thống	4.1 Thiết lập môi trường	- Cài đặt và cấu hình môi trường cho phát triển backend và frontend. - Thiết lập các công cụ và framework hỗ trợ phát triển hệ thống.

Giai đoạn	Công việc chính	Công việc chi tiết
	4.2 Phát triển Backend	- Xây dựng các service theo kiến trúc Microservices. - Tạo và cấu hình các RESTful API cho từng service. - Kết nối và thao tác với cơ sở dữ liệu.
	4.3 Phát triển Frontend	- Xây dựng giao diện người dùng (bằng ReactJS). - Kết nối frontend với các API từ backend.
5. Kiểm thử	5.1 Kiểm thử đơn vị (Unit Testing)	- Kiểm thử đơn vị cho từng service và component.
	5.2 Kiểm thử tích hợp (Integration Testing)	- Kiểm thử các tích hợp giữa các service và giữa frontend với backend. - Sử dụng Swagger UI để kiểm thử API.
	5.3 Kiểm thử hệ thống (System Testing)	- Kiểm thử toàn bộ hệ thống để đảm bảo hoạt động đúng chức năng.
6. Triển khai và vận hành	6.1 Triển khai hệ thống	- Sử dụng Docker để container hóa các service và triển khai lên môi trường sử dụng.
	6.2 Vận hành và giám sát	- Giám sát và theo dõi hiệu suất của hệ thống. - Đảm bảo hệ thống luôn sẵn sàng và có thể mau chóng khôi phục khi có sự cố xảy ra.
7. Bảo trì và nâng cấp	7.1 Bảo trì hệ thống	- Xử lý các lỗi phát sinh và cập nhật hệ thống thường xuyên.

Giai đoạn	Công việc chính	Công việc chi tiết
		- Cập nhật tài liệu và hướng dẫn sử dụng khi có sự thay đổi hoặc cải tiến hệ thống.
	7.2 Cải tiến và mở rộng hệ thống	- Thu thập phản hồi từ người sử dụng để liên tục cải tiến và mở rộng hệ thống. - Thêm các tính năng mới và cải thiện, nâng cấp tính năng hiện có.
8. Quản lý dự án và báo cáo	8.1 Quản lý tiến độ	- Theo dõi tiến độ dự án và điều chỉnh kế hoạch khi cần thiết.
	8.2 Báo cáo	- Báo cáo tiến độ định kỳ cho các bên liên quan. - Họp thường xuyên để cập nhật và giải quyết vấn đề gặp phải.

2.4 Chức năng nghiệp vụ

2.4.1 Chức năng người dùng

Người dùng của hệ thống là những người sử dụng có role như user, warehouse manager và transportation staff. Những người dùng này có chức năng như sau:

- Đăng nhập: đây là chức năng chung của tất cả người dùng, họ sử dụng email và mật khẩu để truy cập vào hệ thống.
- Quản lý tài khoản: người dùng có thể cập nhật thông tin cá nhân của bản thân bao gồm tên, avatar, số điện thoại và giới tính.
- Xem danh sách và xem chi tiết sản phẩm: người dùng có thể duyệt qua danh sách sản phẩm và xem thông tin chi tiết của từng sản phẩm.
- Lọc sản phẩm: người dùng có thể lọc sản phẩm theo danh mục, giá cả và các tiêu chí khác. Màn hình sẽ hiển thị các sản phẩm phù hợp với các tiêu chí đã lọc.

- Tạo đơn hàng: người dùng có thể tạo đơn hàng mới và thêm sản phẩm vào đơn hàng hoặc xóa sản phẩm khỏi đơn hàng đang tạo.
- Theo dõi đơn hàng: người dùng có thể theo dõi trạng thái đơn hàng từ khi tạo đơn hàng đến khi giao hàng.
- Quản lý kho: đây là chức năng riêng của người quản lý kho, người quản lý có thể xem danh sách kho, tình trạng hàng trong kho và lịch sử nhập hàng/xuất hàng của kho.
- Vận chuyển đơn hàng: đây là chức năng riêng của nhân viên vận chuyển, họ là người vận chuyển đơn hàng đến nhà thuốc hoặc cơ sở y tế và quyết định trạng thái của đơn hàng (đang xử lý/ hoàn thành).

2.4.2 Chức năng người quản trị

Người quản trị của hệ thống là những người sử dụng có role như admin hoặc super admin, họ là những người nắm toàn bộ quyền trong hệ thống và có thể sử dụng tất cả chức năng của người dùng. Ngoài ra, họ có những chức năng quản lý mà người dùng không có:

Quản lý người dùng: người quản trị có quyền thêm người dùng mới vào hệ thống bằng cách gửi lời mời cho họ thông qua email cũng như phân quyền cho người dùng. Ngoài ra, người quản trị có quyền xem thông tin người dùng và quản lý quyền truy cập của họ (active/inactive).

Quản lý sản phẩm: người quản trị có tất cả các quyền liên quan đến sản phẩm bao gồm: thêm sản phẩm, xem chi tiết, chỉnh sửa thông tin, xem danh sách, lọc sản phẩm. Ngoại trừ việc xóa sản phẩm khỏi hệ thống, vì điều này sẽ ảnh hưởng đến đơn hàng liên quan đến sản phẩm bị xóa, thay vào đó sản phẩm sẽ được đặt trạng thái inactive.

Quản lý danh mục sản phẩm: tương tự người quản trị có quyền thêm, xem và chỉnh sửa danh mục sản phẩm.

Quản lý đơn hàng: người quản trị có quyền xem và quản lý tất cả các đơn hàng. Ngoài ra, người quản trị có quyền cập nhật trạng thái đơn hàng, từ khi tạo đơn đến khi giao hàng.

Quản lý doanh thu: người quản trị có quyền xem các báo cáo và thống kê liên quan đến doanh thu, số lượng đơn hàng và chi tiết các khoản thu chi.

2.4.3 Yêu cầu hệ thống

2.4.3.1 Yêu cầu về phần cứng

Để hệ thống có thể vận hành và xử lý tốt các tác vụ như mong muốn, phần cứng phải đạt được các yêu cầu như sau.

- Máy chủ (Server)
  - CPU: Intel Xeon hoặc tương đương.
  - RAM: 16GB trở lên.
  - Ổ cứng: SSD 1TB trở lên.
  - Kết nối mạng: Ethernet hoặc Wi-Fi tốc độ cao.
- Máy trạm (Client)
  - CPU: Intel Core i5 trở lên hoặc tương đương.
  - RAM: 16GB trở lên.
  - Ổ cứng: SSD 512GB trở lên.
  - Kết nối mạng: Ethernet hoặc Wi-Fi tốc độ cao.

2.4.3.2 Yêu cầu về phần mềm

Bảng 2.2: Bảng tóm tắt yêu cầu về phần mềm

Thành phần	Yêu cầu
Hệ điều hành máy chủ	Linux (Ubuntu, CentOS) hoặc Windows Server
Hệ điều hành máy trạm	Windows 10/11, macOS hoặc Linux
Cơ sở dữ liệu	PostgreSQL
Ngôn ngữ lập trình	Java Spring Boot - Java 17 (backend)

Thành phần	Yêu cầu
Công nghệ Frontend	ReactJS và SCSS
Caching	Redis
Message Broker	RabbitMQ
Framework	- Spring Boot 3.2.5 - Spring Cloud - Spring Security - Eurika - Spring Data JPA - Swagger - Feign Client

2.4.3.3 Yêu cầu về kết nối

Để đảm bảo truyền tải dữ liệu ổn định và mượt mà hệ thống cần được kết nối Internet tốc độ cao, bên cạnh đó băng thông phải đủ lớn để có thể đáp ứng nhiều người dùng truy cập cùng lúc.

2.4.3.4 Vấn đề bảo mật

Để đảm bảo vấn đề bảo mật và toàn vẹn dữ liệu, hệ thống phải đáp ứng các yêu cầu về bảo mật sau đây:

- Hệ thống phải được bảo vệ và có giải pháp chống xâm nhập. Người dùng chỉ có thể tham gia vào hệ thống khi được gửi lời mời và chấp nhận lời mời.
- Các cơ chế xác thực và phân quyền phải được triển khai để bảo vệ dữ liệu.
- Cần có cơ chế quản lý phiên làm việc.

2.4.3.5 Khả năng mở rộng

Hệ thống phải đảm bảo vấn đề mở rộng và nâng cấp khi lượng dữ liệu tăng lên.

Triển khai cơ chế phân phối tải giữa các dịch vụ (service).

Đảm bảo việc backup và phục hồi dữ liệu khi gặp trường hợp khẩn cấp.

2.4.3.6 Yêu cầu về hiệu năng

Thời gian phản hồi của hệ thống phải dưới 5 giây cho hầu hết các yêu cầu từ người dùng.

Hỗ trợ tối thiểu 500 người dùng đồng thời mà không gặp vấn đề về hiệu năng.

2.4.4 Lựa chọn giải pháp và công nghệ

Với mục đích xây dựng hệ thống quản lý phân phối dược phẩm bằng kiến trúc Microservices trên nền tảng Java Spring Boot đáp ứng tốt các yêu cầu về hiệu năng, tính bảo mật và khả năng mở rộng, nhóm đã cân nhắc và lựa chọn các giải pháp, kiến trúc, công nghệ và framework sau đây:

- Kiến trúc hệ thống: Microservices.
- Backend: Java Spring Boot - ngôn ngữ lập trình: Java 17.
- Frontend: ReactJS và SCSS.
- Cơ sở dữ liệu: PostgreSQL.
- Caching: Redis.
- Message Broker: RabbitMQ.
- Framework: Spring Boot 3.2.5, Spring Cloud, Spring Security, Eurika, Spring Data JPA.
- Other: Swagger, Feign Client.

2.5 Kết quả mong muốn đạt được

Việc xây dựng hệ thống quản lý phân phối dược phẩm bằng kiến trúc Microservices trên nền tảng Java Spring Boot kết hợp với các công nghệ và framework, đáp ứng các điều kiện và yêu cầu như trên nhằm đạt kết quả sau đây:

Về việc tăng hiệu quả quản lý cho quy trình phân phối dược phẩm, hệ thống cho phép người dùng quản lý tốt tất cả các thành phần trong chuỗi phân phối bao gồm người dùng, sản phẩm, khách hàng, đơn hàng, nhà kho, quá trình vận chuyển, doanh thu, tài liệu,...Giúp tối ưu hóa quá trình kinh doanh, giảm thiểu sai sót và mang lại hiệu quả cao.

Xây dựng hệ thống với giao diện người dùng hiện đại, thân thiện và dễ sử dụng, giúp người dùng mau chóng làm quen và thực hiện tốt các thao tác với hệ thống nhằm mang lại trải nghiệm tốt cho người dùng khi sử dụng phần mềm.

Xây dựng hệ thống theo kiến trúc Microservices cho phép hệ thống dễ dàng mở rộng và thay đổi theo nhu cầu. Mỗi service có thể được phát triển, triển khai và bảo trì một cách độc lập.

Hệ thống cho phép người quản trị mời người dùng mới mà không cho phép đăng ký tài khoản và xử dụng JWT để đảm bảo an toàn dữ liệu và tính bảo mật nhằm ngăn chặn các truy cập trái phép. Ngoài ra, sử dụng biện pháp mã hóa mật khẩu để bảo vệ thông tin người dùng.

Hệ thống sử dụng RabbitMQ giúp xử lý dữ liệu và quản lý message queue giữa các service trong hệ thống, đảm bảo khả năng xử lý dữ liệu theo thời gian thực. Ngoài ra, xử dụng Eureka để quản lý các service trong hệ thống, đảm bảo cho việc tự động phát hiện và kết nối giữa các service với nhau.

Với những yếu tố trên, nhóm em mong muốn sẽ xây dựng được hệ thống quản lý phân phối dược phẩm đáp ứng tốt các yêu cầu của khách hàng và doanh nghiệp về nghiệp vụ kinh doanh hiện tại, ngoài ra hệ thống phải sẵn sàng cho những thay đổi, phát triển và mở rộng trong tương lai, mang lại lợi ích và giải pháp tốt cho doanh nghiệp, từ đó học hỏi thêm nhiều kiến thức và kinh nghiệm bổ ích cho bản thân.

CHƯƠNG 3. TÌM HIỂU CÔNG NGHỆ THỰC HIỆN

3.1 Front-end

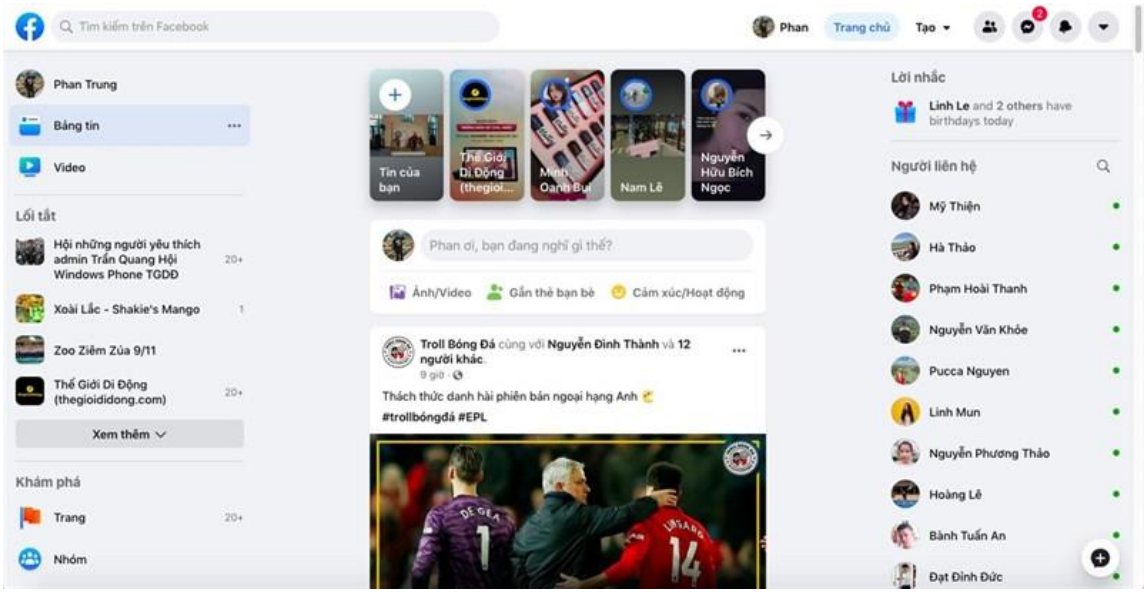
3.1.1 ReactJS

ReactJS là một thư viện JavaScrip mã nguồn mở được phát triển bởi Facebook, được sử dụng để xây dựng các giao diện người dùng UI cho các ứng dụng web. Thư viện này được giới thiệu lần đầu tiên vào năm 2013 và đã nhanh chóng trở thành một trong những công nghệ phổ biến trong việc hỗ trợ lập trình viên phát triển front-end.

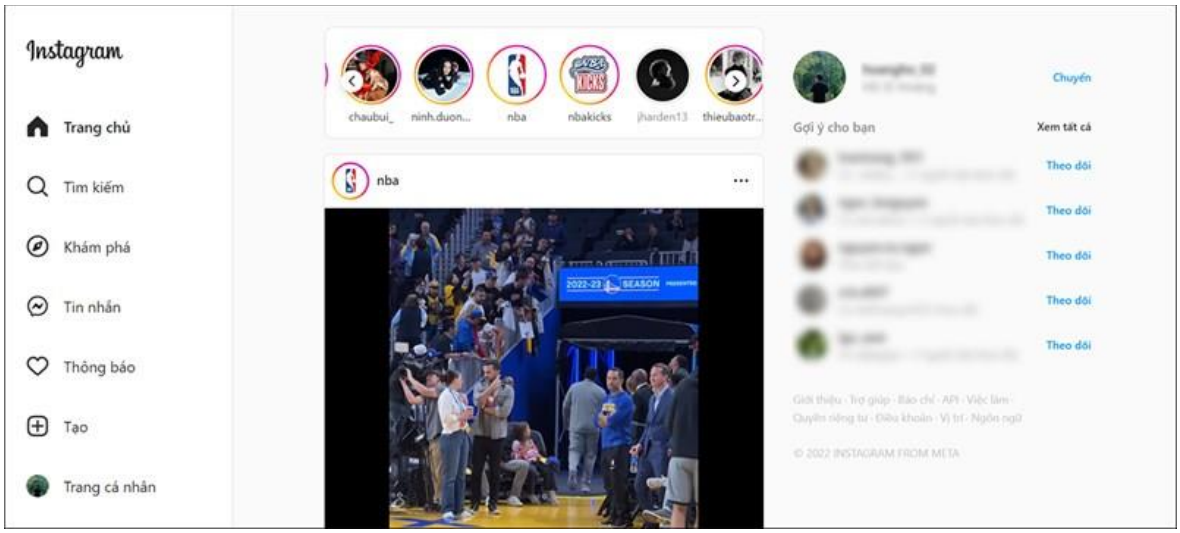
Thư viện ReactJS tập trung vào việc tạo ra các component có thể tái sử dụng giúp cho việc phát triển giao diện người dùng được thực hiện một cách dễ dàng và hiệu quả hơn. Đây là một công nghệ mang lại hiệu suất cao và trải nghiệm mượt mà.

Các ứng dụng nổi tiếng có giao diện web được phát triển bằng ReactJS có thể kể đến như Facebook (Hình 3.1), Instagram (Hình 3.2), Netflix, Tesla, Pinterest,...

ReactJS đã và đang trở thành một trong những công cụ mạnh mẽ và phổ biến nhất trong lĩnh vực phát triển frontend web và trở thành lựa chọn hàng đầu của các lập trình viên. Tuy nhiên, dù là công nghệ nào thì vẫn có ưu điểm và nhược điểm, vậy nên cần phải cân nhắc kỹ dựa trên đặc điểm của dự án để đảm bảo sự phù hợp và hiệu quả.



Hình 3.1: Giao diện Facebook



Hình 3.2: Giao diện Instagram

3.1.2 Ưu điểm và nhược điểm của ReactJS

3.1.2.1 Ưu điểm

ReactJS trở thành lựa chọn hàng đầu trong lĩnh vực phát triển frontend bởi những ưu điểm sau đây:

- ReactJS sử dụng Virtual DOM để cập nhật giao diện một cách tối ưu, ngay khi có sự thay đổi trong giao diện thì nó sẽ lập tức cập nhật những phần cần thiết, đặc điểm này giúp tăng hiệu suất làm việc.
- ReactJS cho phép xây dựng các UI từ các component nhỏ và có thể tái sử dụng, giúp mã nguồn trở nên rõ ràng, dễ bảo trì, phát triển giao diện nhanh chóng và hiệu quả.
- JavaScript có cú pháp mở rộng là JSX, nó cho phép viết HTML trực tiếp trong JavaScript, giúp mã nguồn trở nên dễ hiểu, trực quan và dễ sửa đổi hơn.
- ReactJS có thể giúp cải thiện khả năng SEO của các ứng dụng web.
- ReactJS có một cộng đồng phát triển lớn mạnh và năng động, bên cạnh đó có rất nhiều công cụ, tài liệu và thư viện nhằm hỗ trợ người dùng.

- Ngoài ra, ReactJS luôn được cập nhật các tính năng mới nhằm cải tiến hiệu suất dự án.

3.1.2.2 Nhược điểm

Bên cạnh những ưu điểm vượt trội của mình thì ReactJS cũng có những nhược điểm và hạn chế mà lập trình viên cần cân nhắc để lựa chọn phù hợp, các nhược điểm có thể kể đến như sau:

- ReactJS là một thư viện khá phức tạp để học hỏi và tiếp cận đối với những người mới bắt đầu lập trình frontend, nhất là khi cần phải nắm vững các khái niệm như component, state, props và lifecycle methods.
- ReactJS không phải là một framework toàn diện vì vậy lập trình viên phải tự lựa chọn và tích hợp thêm các công cụ khác như Redux (quản lý state), React Router (điều hướng),... để xây dựng nên giao diện hoàn chỉnh.
- ReactJS là thư viện có kích thước khá nhỏ gọn so với các framework khác, tuy nhiên, như đã nói trước đó thì để xây dựng giao diện hoàn chỉnh cần phải kết hợp với các công cụ và thư viện khác, nên khi hoàn thành thì kích thước của dự án sẽ rất lớn và ảnh hưởng đến tốc độ tải trang.

3.2 Back-end

3.2.1 Java Spring Boot, Spring Cloud, Spring Security, Spring Data JPA

3.2.1.1 Java Spring Boot

Java Spring Boot là một framework dựa trên nền tảng Spring Framework, được thiết kế để đơn giản hóa quá trình phát triển và triển khai các ứng dụng Spring. Spring Boot cung cấp các công cụ và tính năng giúp giảm thiểu cấu hình thủ công và cho phép các nhà phát triển tạo ra các ứng dụng Java mạnh mẽ một cách nhanh chóng và hiệu quả.

3.2.1.2 Spring Cloud

Spring Cloud là một tập hợp các công cụ giúp phát triển các hệ thống phân tán và microservices một cách dễ dàng. Nó được xây dựng trên nền tảng Spring Boot và cung cấp các giải pháp cho các vấn đề phổ biến trong phát triển và triển khai các ứng dụng phân tán, như quản lý cấu hình, khám phá dịch vụ, cân bằng tải, quản lý phiên bản, theo dõi, và xử lý lỗi.

3.2.1.3 Spring Security

Spring Security là một framework mạnh mẽ và linh hoạt được tích hợp vào Spring Framework, cung cấp các tính năng bảo mật toàn diện cho các ứng dụng Java. Nó giúp bảo vệ các ứng dụng khỏi các mối đe dọa bảo mật phổ biến bằng cách cung cấp các công cụ để xác thực (authentication) và phân quyền (authorization).

3.2.1.4 Spring Data JPA

Spring Data JPA là một phần của dự án Spring Data, cung cấp một cách tiếp cận tiện lợi và mạnh mẽ để làm việc với cơ sở dữ liệu bằng cách sử dụng Java Persistence API (JPA). Spring Data JPA giúp đơn giản hóa các thao tác cơ bản với cơ sở dữ liệu, chẳng hạn như truy vấn, lưu trữ và quản lý dữ liệu, bằng cách cung cấp một lớp trừu tượng trên JPA.

3.2.2 Kiến trúc Microservices

3.2.2.1 Kiến trúc Microservices là gì?

Kiến trúc Microservices (Microservices Architecture) là một phong cách thiết kế phần mềm trong đó một ứng dụng lớn được xây dựng từ các dịch vụ nhỏ, độc lập và có thể triển khai riêng lẻ. Mỗi dịch vụ thực hiện một chức năng cụ thể và giao tiếp với các dịch vụ khác thông qua các giao thức nhẹ, thường là HTTP hoặc messaging.

3.2.2.2 Đặc điểm của Kiến trúc Microservices:

- Độc lập triển khai (Independent Deployment)
- Độc lập phát triển (Independent Development)

- Mỗi dịch vụ là một đơn vị chức năng (Single Responsibility)
- Giao tiếp qua giao thức nhẹ (Lightweight Communication)
- Khả năng mở rộng (Scalability):
- Đa dạng công nghệ (Polyglot Programming)

3.2.2.3 Ưu điểm của Kiến trúc Microservices

- Tăng tính linh hoạt trong phát triển và triển khai
- Khả năng chịu lỗi cao
- Dễ dàng mở rộng
- Tối ưu hóa tài nguyên
- Đa dạng công nghệ

3.2.2.4 Nhược điểm của Kiến trúc Microservices:

- Quản lý phức tạp
- Giao tiếp giữa các dịch vụ có thể gây ra độ trễ và phức tạp
- Tăng cường giao tiếp giữa các microservices có thể gây ra độ trễ và phức tạp hơn trong việc quản lý giao tiếp.
- Phân tán dữ liệu giữa các microservices có thể gây ra thách thức trong việc duy trì tính nhất quán.
- Cần phải đảm bảo bảo mật giữa các microservices và quản lý quyền truy cập tốt hơn.

3.2.3 Rabbit MQ

3.2.3.1 Rabbit MQ là gì?

RabbitMQ là một hệ thống hàng đợi thông điệp mã nguồn mở được sử dụng để trao đổi và xử lý thông điệp giữa các hệ thống hoặc các thành phần của một hệ thống phân tán. Nó dựa trên giao thức AMQP (Advanced Message Queuing Protocol) và được viết bằng ngôn ngữ lập trình Erlang, nổi bật với hiệu suất cao và khả năng mở rộng.

3.2.3.2 Các khái niệm chính trong RabbitMQ:

1. Producer (Nhà sản xuất):

   Thành phần gửi thông điệp đến RabbitMQ. Producers thường là các ứng dụng hoặc dịch vụ tạo ra thông điệp và gửi chúng đến các hàng đợi.

2. Queue (Hàng đợi):

   Nơi lưu trữ các thông điệp trước khi chúng được tiêu thụ. Các hàng đợi trong RabbitMQ tuân theo mô hình FIFO (First In, First Out).

3. Consumer (Nhà tiêu thụ)

   Thành phần nhận và xử lý thông điệp từ RabbitMQ. Consumers có thể là các dịch vụ hoặc ứng dụng lấy thông điệp từ hàng đợi để xử lý.

4. Exchange (Bộ trao đổi)

   Thành phần chịu trách nhiệm định tuyến thông điệp từ Producers đến các hàng đợi dựa trên các quy tắc định tuyến. Có bốn loại exchange chính: Direct, Topic, Fanout, và Headers.

5. Binding (Ràng buộc)

   Quy tắc định tuyến giữa exchange và queue. Bindings xác định cách các thông điệp sẽ được định tuyến từ exchange đến hàng đợi.

3.2.4 Eureka

3.2.4.1 Eureka là gì?

Eureka là một dịch vụ khám phá và đăng ký (service discovery) mã nguồn mở của Netflix, được sử dụng rộng rãi trong các kiến trúc microservices. Nó cho phép các dịch vụ trong hệ thống đăng ký và khám phá lẫn nhau mà không cần phải cấu hình tĩnh các địa chỉ mạng.

3.2.4.2 Các khái niệm chính trong Eureka:

Eureka Server: Là trung tâm đăng ký dịch vụ, nơi các dịch vụ có thể đăng ký và tìm kiếm các dịch vụ khác. Eureka Server lưu trữ thông tin về tất cả các dịch vụ đã đăng ký và cung cấp API để các dịch vụ có thể đăng ký, hủy đăng ký và tìm kiếm.

Eureka Client: Là các dịch vụ đăng ký với Eureka Server. Mỗi Eureka Client sẽ gửi thông tin của mình đến Eureka Server và định kỳ gửi các gói tin "heartbeat" để cho Eureka Server biết rằng dịch vụ vẫn đang hoạt động.

Service Registry: Là nơi lưu trữ thông tin về tất cả các dịch vụ đã đăng ký. Service Registry chứa thông tin như tên dịch vụ, địa chỉ IP, cổng, và trạng thái của dịch vụ.

Instance: Là một bản sao của một dịch vụ. Mỗi instance có một định danh duy nhất và thông tin chi tiết về dịch vụ.

3.2.5 Redis

3.2.5.1 Redis là gì?

Redis (Remote Dictionary Server) là một cơ sở dữ liệu lưu trữ cấu trúc dữ liệu trong bộ nhớ (in-memory data structure store) mã nguồn mở, được sử dụng làm cơ sở dữ liệu, bộ nhớ đệm (cache), và môi trường trao đổi thông điệp (message broker). Redis nổi bật với hiệu suất cao và khả năng hỗ trợ nhiều cấu trúc dữ liệu khác nhau như chuỗi (strings), danh sách (lists), tập hợp (sets), băm (hashes), và nhiều cấu trúc khác.

3.2.5.2 Đặc điểm chính của Redis

In-memory Storage: Tất cả dữ liệu được lưu trữ trong bộ nhớ, cho phép truy cập và thao tác dữ liệu với tốc độ rất nhanh.

Đa dạng cấu trúc dữ liệu: Redis hỗ trợ nhiều loại cấu trúc dữ liệu khác nhau như strings, lists, sets, sorted sets, hashes, bitmaps, hyperloglogs, và streams.

Khả năng sao chép (Replication): Redis hỗ trợ sao chép dữ liệu giữa các instance để tăng cường độ tin cậy và khả năng chịu lỗi.

Khả năng mở rộng (Scalability): Redis hỗ trợ sharding và clustering, cho phép mở rộng hệ thống một cách linh hoạt và hiệu quả.

Hỗ trợ các kịch bản khác nhau: Redis có thể được sử dụng làm cơ sở dữ liệu chính, bộ nhớ đệm, message broker, và thậm chí là queue system.

Hỗ trợ giao dịch (Transactions): Redis hỗ trợ thực hiện các giao dịch atomically thông qua lệnh MULTI/EXEC.

Lua scripting: Hỗ trợ chạy các script Lua để thực hiện các thao tác phức tạp trên dữ liệu.

3.3 Cơ sở dữ liệu

3.3.1 PostgreSQL

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ đối tượng mã nguồn mở được phát triển từ năm 1986 tại Đại học California. Nó được biết đến bởi khả năng mở rộng, độ tin cậy cao và tuân thủ tiêu chuẩn SQL. PostgreSQL là một lựa chọn phổ biến cho các ứng dụng doanh nghiệp, web và các hệ thống dữ liệu lớn (big data).

PostgreSQL là một trong những hệ quản trị cơ sở dữ liệu mạnh mẽ và linh hoạt, phù hợp với nhiều loại ứng dụng, hệ thống và dự án. Tuy nhiên, PostgreSQL vẫn có những ưu điểm và nhược điểm, người dùng cần cân nhắc và lựa chọn phù hợp với đặc điểm của dự án.

3.3.2 Ưu điểm và nhược điểm của PostgreSQL

3.3.2.1 Ưu điểm

Để trở thành lựa chọn hàng đầu trong việc quản trị cở sở dữ liệu, PostgreSQL có những ưu điểm sau đây:

- PostgreSQL là một mã nguồn mở và miễn phí.
- Hỗ trợ đầy đủ các tiêu chuẩn của SQL, dễ dàng tương thích và chuyển đổi dữ liệu với các hệ quản trị cơ sở dữ liệu khác.
- PostgreSQL hỗ trợ cho nhiều kiểu dữ liệu, kể cả dữ liệu phức tạp như JSON, XML và hstore, tạo điều kiện thuận lợi cho việc lưu trữ và xử lý dữ liệu phi cấu trúc.
- PostgreSQL cho phép mở rộng bằng cách sử dụng extension và thêm các chức năng mới mà không cần sửa đổi mã nguồn cơ bản.

- PostgreSQL tuân thủ các thuộc tính ACID (Atomicity, Consistency, Isolation, Durability) đảm bảo tính nhất quán và độ tin cậy của dữ liệu.
- Cộng đồng phát triển lớn, tài liệu phong phú, dễ dàng tìm kiếm sự hỗ trợ và giải pháp khi gặp vấn đề cũng là một điểm cộng của PostgreSQL.
- Cuối cùng, PostgreSQL cung cấp các tính năng bảo mật mạnh mẽ như kiểm soát truy cập chi tiết, xác thực người dùng và hỗ trợ SSL.

3.3.2.2 Nhược điểm

Bên cạnh những ưu điểm thì PostgreSQL vẫn tồn tại một số nhược điểm sau đây:

- Đối với các hệ thống dữ liệu cực lớn hoặc yêu cầu hiệu suất cao thì có thể PostgreSQL không nhanh bằng một số hệ quản trị cơ sở dữ liệu khác như MySQL.
- Việc cấu hình và quản lý PostgreSQL có thể phức tạp hơn so với một số hệ quản trị cơ sở dữ liệu khác, điều này yêu cầu người quản trị phải có kiến thức và kinh nghiệm chuyên sâu.
- So với một số hệ quản trị cơ sở dữ liệu nhẹ thì PostgreSQL có thể yêu cầu nhiều tài nguyên hệ thống hơn về bộ nhớ và CPU.
- So với hệ quản trị cơ sở dữ liệu thương mại như Oracle hoặc SQL Server, PostgreSQL có ít tài liệu và sự hỗ trợ doanh nghiệp hơn.

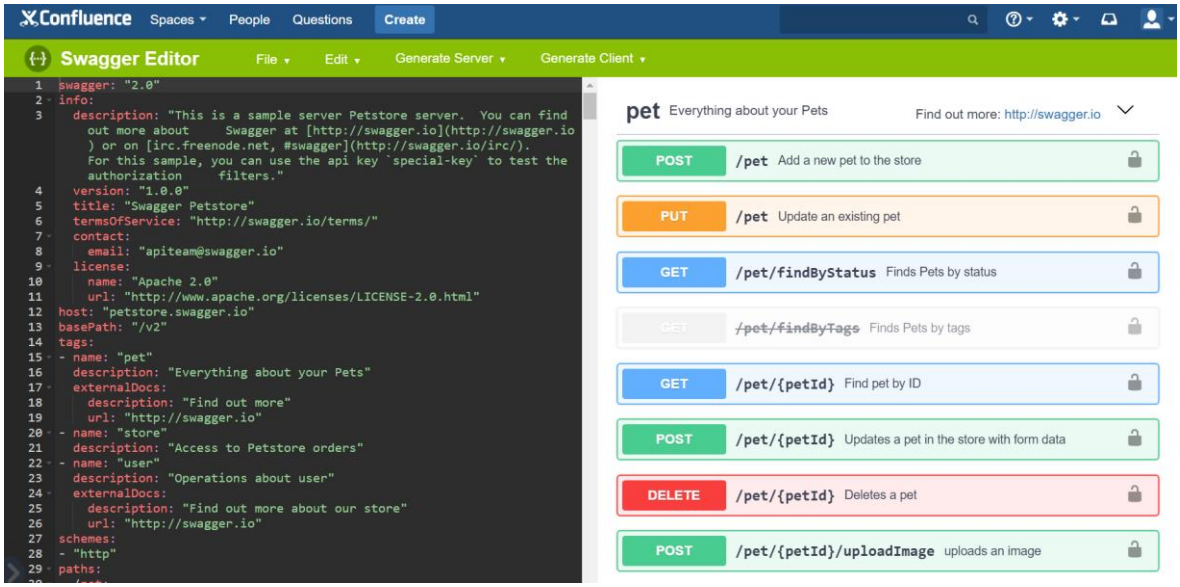
3.4 API

3.4.1 Swagger và Swagger UI

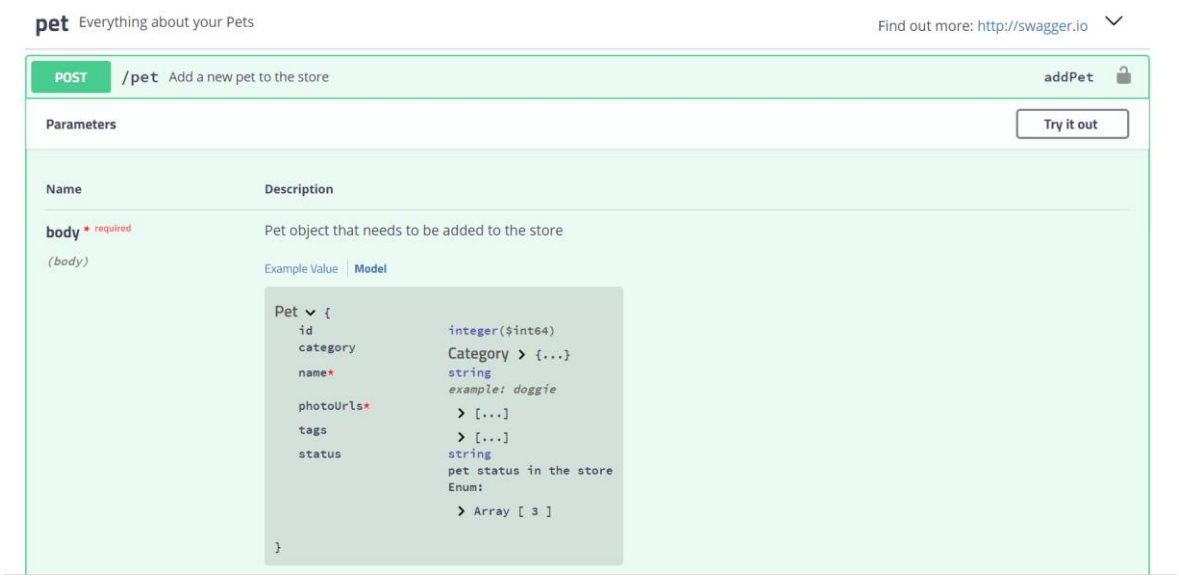
Swagger là một bộ công cụ mã nguồn mở được dùng để thiết kế, xây dựng, tài liệu hóa và sử dụng các API REST. Nó cung cấp một cách chuẩn hóa để mô tả cấu trúc API, cho phép các nhà phát triển dễ dàng hiểu và sử dụng API.

Swaggwr UI (Hình 3.3) là một phần của hệ sinh thái Swagger, cung cấp một giao diện người dùng để trực quan hóa và tương tác với API REST. Nó hiển thị tài

liệu API theo định dạng Swagger và cho phép người dùng thử nghiệm các endpoint của API trực tiếp từ trình duyệt (Hình 3.4).



Hình 3.3: Giao diện của Swagger và Swagger UI



Hình 3.4: Giao diện xem chi tiết API trên Swagger UI

3.4.2 Ưu điểm và nhược điểm của Swagger và Swagger UI

3.4.2.1 Ưu điểm

Swagger và Swagger UI được nhóm em lựa chọn cho việc thiết kế và sử dụng API vì những ưu điểm sau đây:

- Swagger tuân thủ OpenAPI Specification nên có thể tiêu chuẩn hóa tài liệu API và làm nó trở nên nhất quán và dễ hiểu.
- Swagger cho phép tự động tạo tài liệu API từ mã nguồn giúp tiết kiệm thời gian, công sức và giảm thiểu lỗi phát sinh so với việc viết tài liệu thủ công.
- Swagger UI cung cấp giao diện trực quan, dễ sử dụng để tương tác với API, có thể trực tiếp thử nghiệm từ trình duyệt mà không cần viết mã.
- Swagger hỗ trợ nhiều ngôn ngữ lập trình và framework khác nhau, dễ dàng tích hợp với nhiều dự án và hệ thống.
- Swagger UI hỗ trợ người dùng gửi yêu cầu và nhận phản hồi trực tiếp từ API, giúp việc kiểm thử và xác thực API dễ dàng hơn.
- Swagger có thể mở rộng và tích hợp với các công cụ khác như Postman giúp cải thiện hiệu suất và quy trình làm việc.

3.4.2.2 Nhược điểm

Bên cạnh các ưu điểm nổi bật thì Swagger và Swagger UI vẫn tồn tại một số nhược điểm như:

Đối với các dự án lớn và phức tạp, việc cấu hình và duy trì tài liệu API bằng Swagger có thể trở nên phức tạp và tốn thời gian.

Với các API lớn có thể ảnh hưởng đến hiệu suất xử lý khi tương tác với Swagger UI.

Hạn chế trong việc tùy chỉnh giao diện.

Swagger yêu cầu người dùng phải mô tả API chính xác và đầy đủ, nếu mô tả có thiếu sót hoặc không chính xác, có thể dẫn đến lỗi khi sử dụng API.

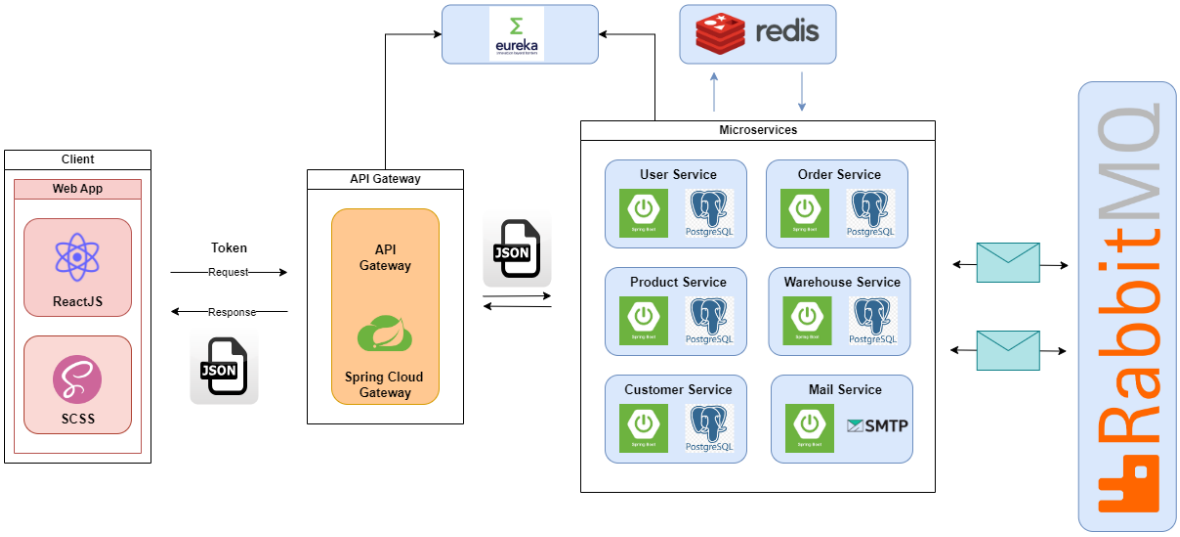
CHƯƠNG 4. THIẾT KẾ HỆ THỐNG

4.1 Cấu trúc của hệ thống

Hệ thống được thiết kế dựa trên một kiến trúc phân tán, linh hoạt, dễ dàng mở rộng và sửa đổi. Cấu trúc hệ thống bao gồm các thành phần chính như client (webapp), API Gateway và các service trong hệ thống phân tán. Bên cạnh các thành phần chính còn có các công nghệ và framework phục vụ cho việc giao tiếp giữa các service.

- Client: Webapp được xây dựng bằng ReactJS – cho người dùng thấy được một giao diện trực quan về hệ thống thông qua trình duyệt web. Bên cạnh ReactJS, HTML và SCSS cũng được sử dụng với mục đích mang lại trải nghiệm tốt hơn cho người dùng khi sử dụng hệ thống.
- API Gateway: Được xây dựng dựa trên Spring Cloud Gateway, đây là thành phần trung tâm của hệ thống, chịu trách nhiệm điều hướng các yêu cầu từ client đến các microservice tương ứng. Ngoài ra, thành phần này còn đảm nhiệm chức năng xác thực người dùng nhằm ngăn chặn các truy cập ẩn hoặc các truy cập từ người dùng chưa được thêm từ hệ thống.
- Microservices: Dựa trên Java Spring Boot và PostgreSQL, mỗi service chịu trách nhiệm cho một chức năng cụ thể và được phân chia dựa trên chức năng. Bên cạnh các công cụ chính là Java Spring Boot và Postgre phần này còn được xây dựng bởi Spring Feign Client.
- Các service sẽ giao tiếp với nhau bằng cách sử dụng message broker (rabbitMQ).
- Redis: được sử dụng để cache dữ liệu, giảm tải cho cơ sở dữ liệu, lưu trữ thông tin phiên (session), lưu trữ dữ liệu tạm thời (mã OTP).

4.2 Sơ đồ cấu trúc của hệ thống



Hình 4.1: Sơ đồ cấu trúc của hệ thống

4.3 Mô tả nghiệp vụ

Hệ thống được xây dựng với tên gọi là ERPPharmaceutical, được thiết kế dựa trên nghiệp vụ của doanh nghiệp phân phối dược phẩm. Các tác nhân của hệ thống bao gồm: Super Admin, Admin, Warehouse Manager, Transportation Staff và User. Tất cả tác nhân đều có chức năng đăng nhập vào hệ thống, tuy nhiên mỗi tác nhân sẽ có quyền hạn nhất định trong các chức năng quản lý của hệ thống.

4.3.1 Super Admin

Super Admin là quản lý cấp cao của hệ thống, có quyền quyết định toàn bộ nghiệp vụ, quản lý tất cả các chức năng và hoạt động của hệ thống:

- Quản lý người dùng: Thêm và phân quyền cho người dùng mới, xem, tìm kiếm và sửa thông tin người dùng, đặt trạng thái (active hay inactive) cho tài khoản của người dùng (Admin, Warehouse Manager, Transportation Staff và User).
- Quản lý danh mục sản phẩm: Thêm, xem, tìm kiếm, sửa thông tin và đặt trạng thái (active hay inactive) cho danh mục sản phẩm.

- Quản lý khách hàng: Thêm, xem, tìm kiếm, sửa thông tin và đặt trạng thái (active hay inactive) cho khách hàng của hệ thống. Lưu ý rằng khách hàng bao gồm nhà cung cấp sản phẩm (supplier) và nhà thuốc (pharmacy).
- Quản lý sản phẩm: Thêm, xem, tìm kiếm và sửa thông tin và đặt trạng thái (active hay inactive) cho sản phẩm trong hệ thống. Lưu ý rằng cần phải thêm nhà sản xuất của sản phẩm vào hệ thống trước, trong form thêm sản phẩm sẽ có trường lựa chọn nhà sản xuất.
- Quản lý đơn hàng: Tạo yêu cầu mua hàng/ bán hàng, tạo hóa đơn, thanh toán đơn hàng.
- Quản lý kho hàng: Chấp nhận yêu cầu nhập hàng vào kho/ xuất hàng khỏi kho, quản lý số lượng sản phẩm trong kho, lưu trữ thông tin nhập kho và xuất kho.

4.3.2 Admin

Admin là quản lý của hệ thống, có quyền quyết định vào đa số nghiệp vụ, quản lý hầu hết các chức năng và hoạt động của hệ thống, tuy nhiên sẽ có hạn chế so với Super Admin:

- Quản lý người dùng: Thêm và phân quyền cho người dùng mới, xem, tìm kiếm, và sửa thông tin người dùng (Admin, Warehouse Manager, Transportation Staff và User).
- Quản lý danh mục sản phẩm: Thêm, xem, tìm kiếm, sửa thông tin danh mục sản phẩm.
- Quản lý khách hàng: Thêm, xem, tìm kiếm, sửa thông tin khách hàng của hệ thống. Lưu ý rằng khách hàng bao gồm nhà cung cấp sản phẩm (supplier) và nhà thuốc (pharmacy).
- Quản lý sản phẩm: Thêm, xem, tìm kiếm, sửa thông tin sản phẩm trong hệ thống.

- Quản lý đơn hàng: Tạo yêu cầu mua hàng/ bán hàng, tạo hóa đơn, thanh toán đơn hàng.
- Quản lý kho hàng: Chấp nhận yêu cầu nhập hàng vào kho/ xuất hàng khỏi kho, quản lý số lượng sản phẩm trong kho, lưu trữ thông tin nhập kho và xuất kho.

4.3.3 Warehouse Manager

Warehouse Manager là người quản lý kho sản phẩm của hệ thống, có quyền quyết định và quản lý hầu hết các chức năng và hoạt động của kho cùng với sản phẩm nhập-xuất kho, tuy nhiên sẽ có hạn chế so với Super Admin và Admin:

- Quản lý người dùng: Xem thông tin cá nhân của người dùng.
- Quản lý sản phẩm: Xem và tìm kiếm thông tin sản phẩm trong hệ thống.
- Quản lý kho hàng: Chấp nhận yêu cầu nhập hàng vào kho/ xuất hàng khỏi kho, quản lý số lượng sản phẩm trong kho, lưu trữ thông tin nhập kho và xuất kho.

4.3.4 Transportation Staff

Transportation Staff là người vận chuyển đơn hàng, có quyền cập nhật trạng thái của đơn hàng trong quá trình vận chuyển:

- Quản lý người dùng: Xem thông tin cá nhân của người dùng.
- Quản lý sản phẩm: Xem và tìm kiếm thông tin sản phẩm trong hệ thống.
- Quản lý kho hàng: Chấp nhận yêu cầu nhập hàng vào kho/ xuất hàng khỏi kho, quản lý số lượng sản phẩm trong kho, lưu trữ thông tin nhập kho và xuất kho.

4.3.5 User

User là nhân viên tạo đơn hàng và thực hiện thanh toán đơn hàng (trường hợp khách hàng thanh toán bằng tiền mặt), có quyền cập nhật trạng thái thanh toán của đơn hàng:

- Quản lý người dùng: Xem thông tin cá nhân của người dùng.

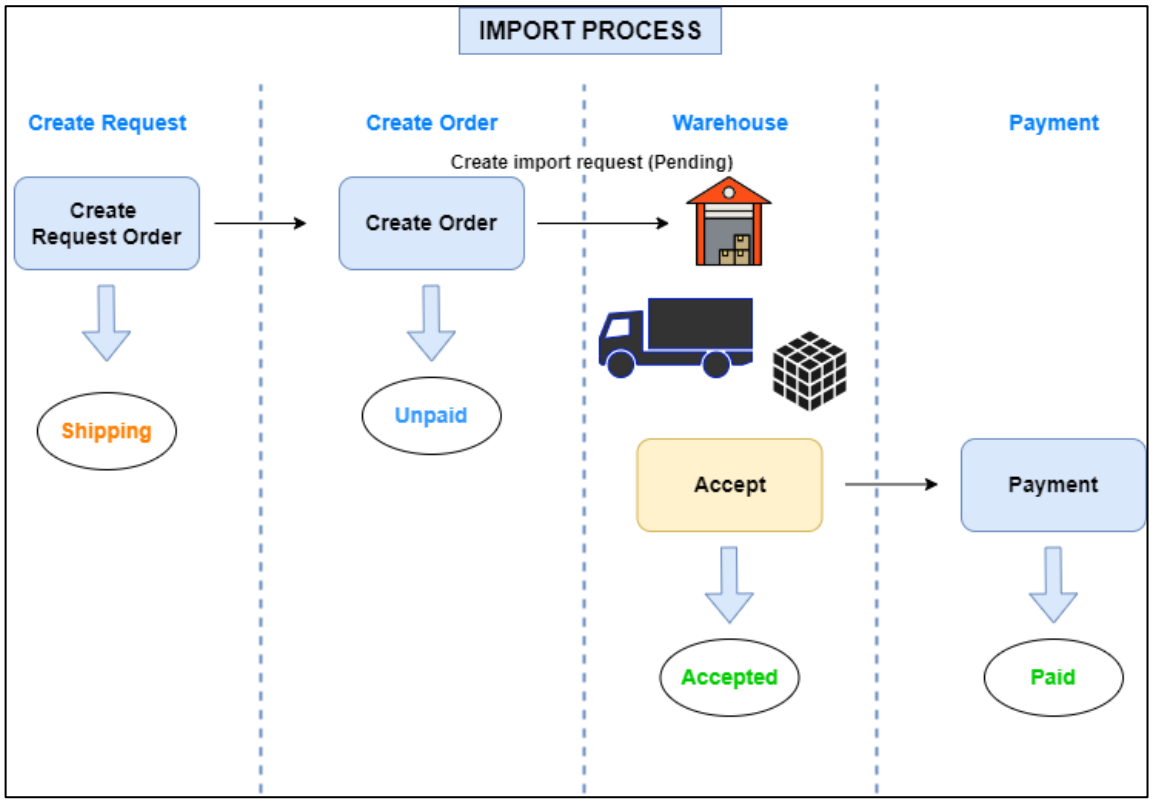
- Quản lý sản phẩm: Xem và tìm kiếm thông tin sản phẩm trong hệ thống.
- Quản lý đơn hàng: Tạo yêu cầu mua hàng/ bán hàng, tạo hóa đơn, thanh toán đơn hàng.

4.3.6 Quy trình nhập hàng

Trong quy trình nhập hàng, các service tham gia gồm Order Service và Warehouse Service. Quy trình được thực hiện như mô tả ở sơ đồ Hình 4.2, các bước chi tiết như sau:

Đầu tiên, tại Order Service sẽ thực hiện chức năng tạo yêu cầu mua hàng (Create Request Order) để tạo yêu cầu gửi về phía nhà sản xuất sẽ mua hàng.

Sau khi phía nhà sản xuất xác nhận đơn hàng thì sử dụng chức năng tạo đơn hàng (Create Order), đồng thời gửi cho Warehouse Service yêu cầu nhập kho để nhắc nhở sắp có đơn hàng mua.

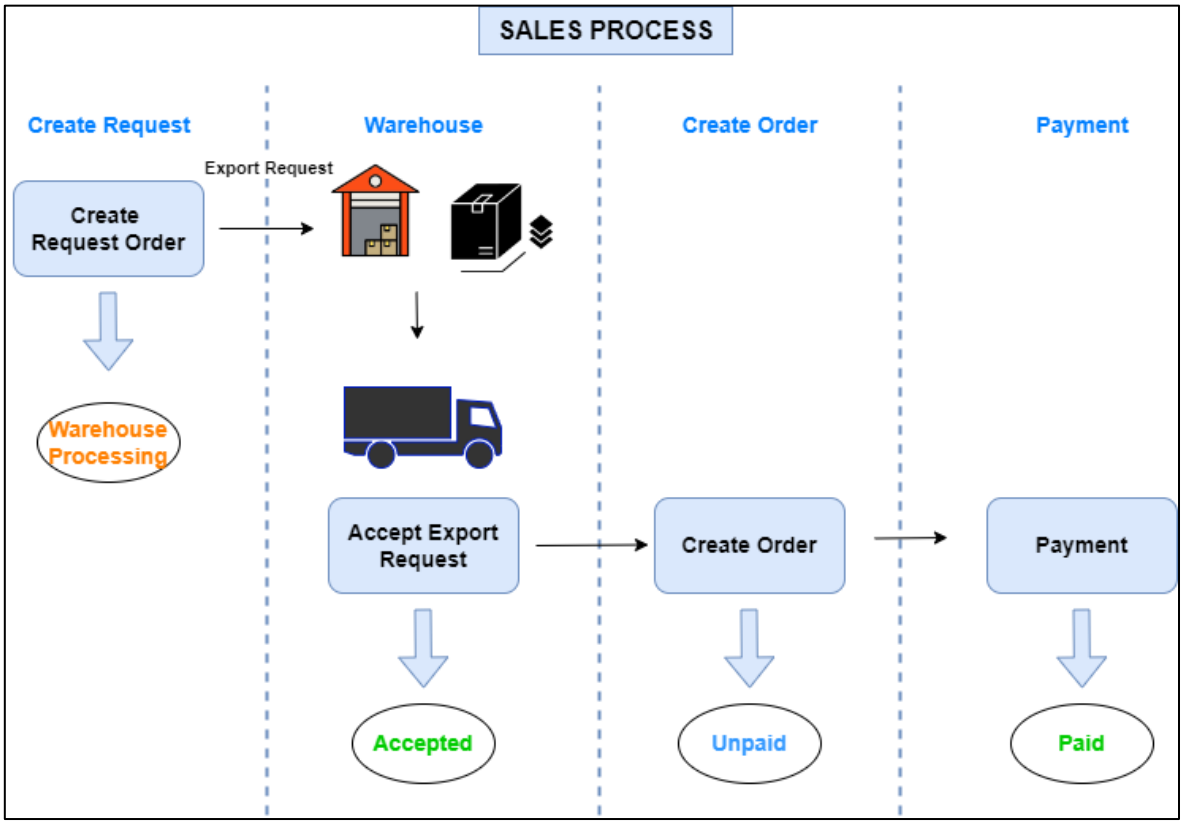


Hình 4.2: Sơ đồ mô tả quy trình nhập hàng

Sau khi kho nhận được hàng từ nhà sản xuất sẽ click button <Accept> trong trang chi tiết của import request để thông báo rằng đã nhận được hàng.

Cuối cùng, thực hiện thanh toán đơn hàng để hoàn tất quy trình.

4.3.7 Quy trình bán hàng



Hình 4.3: Sơ đồ mô tả quy trình bán hàng

Tương tự quy trình nhập hàng, quy trình bán hàng cũng có sự tham gia của Order Service và Warehouse Service. Quy trình được thực hiện như mô tả ở sơ đồ Hình 4.3, các bước chi tiết như sau:

Đầu tiên, tại Order Service sẽ thực hiện chức năng tạo yêu cầu bán hàng (Create Request Order) để gửi cho Warehouse Service yêu cầu xuất kho để nhắc nhở sắp có đơn hàng bán.

Sau khi kho thực hiện đóng gói sản phẩm và vận chuyển đến khách hàng sẽ click button <Accept> trong trang chi tiết của export request để thông báo rằng đã giao hàng thành công.

Cuối cùng, thực hiện thanh toán đơn hàng để hoàn tất quy trình.

4.4 Sơ đồ Use case

4.4.1 Xác định các Use case

Bảng 4.1: Các use case trong hệ thống

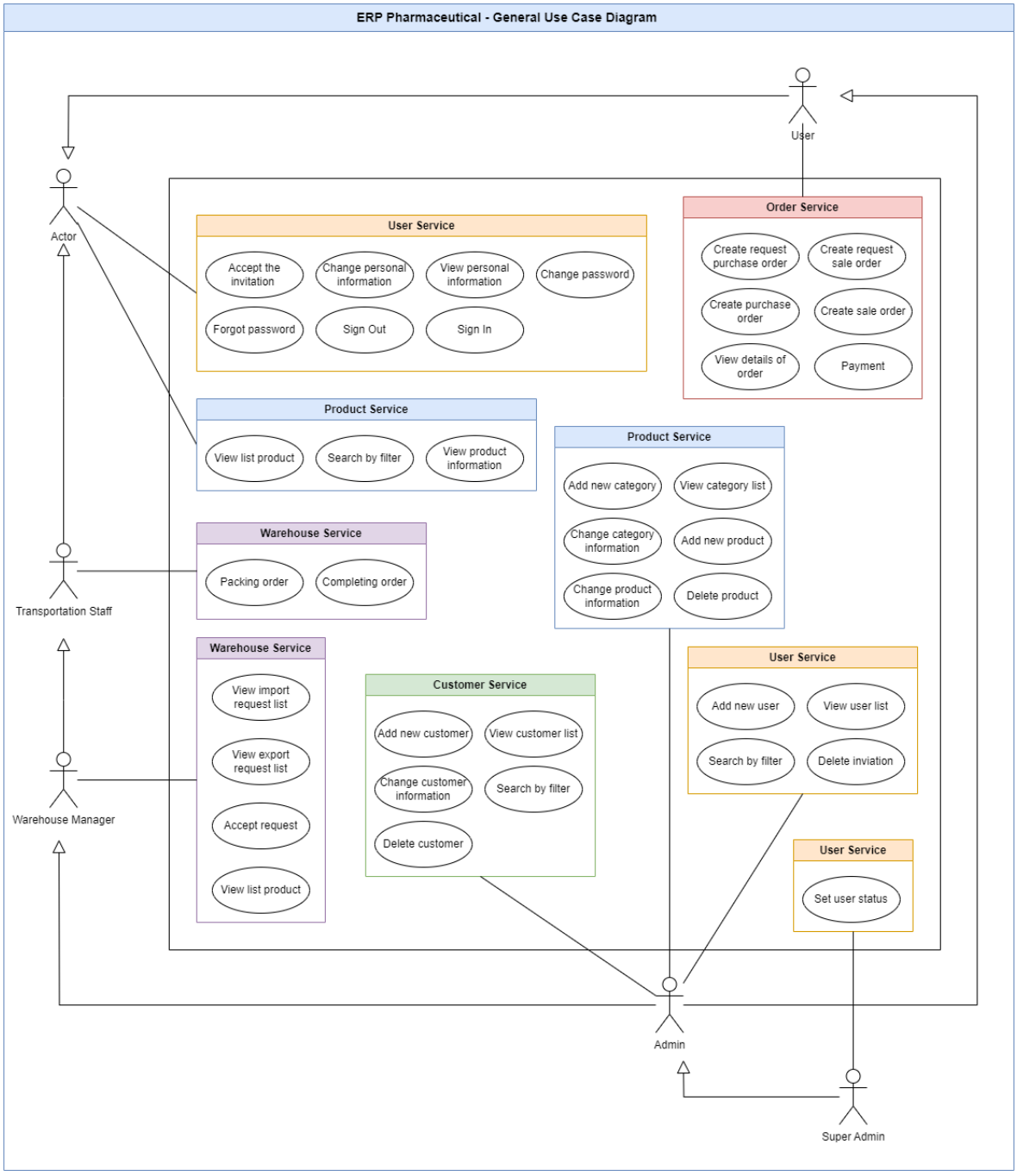
ID	Tên Use Case	Mô tả
UC01	Thêm người dùng	Super Admin hoặc Admin thêm người dùng và phân quyền, lời mời sẽ được gửi đến email của người dùng.
UC02	Chấp nhận lời mời và thêm thông tin người dùng	Người dùng được gửi lời mời tham gia hệ thống sẽ click vào link được gửi trong email và nhập các thông tin để tạo tài khoản người dùng.
UC03	Xem thông tin cá nhân	Người dùng xem thông tin tài khoản cá nhân.
UC04	Thay đổi thông tin cá nhân	Người dùng thực hiện thao tác thay đổi thông tin cá nhân.
UC05	Quên mật khẩu	Người dùng thực hiện việc đổi mật khẩu khi quên mật khẩu.
UC06	Đổi mật khẩu	Người dùng thực hiện việc đổi mật khẩu sau khi đã đăng nhập thành công vào hệ thống.
UC07	Đăng nhập	Người dùng đăng nhập bằng tài khoản đã được mời và hoàn thành bước thêm thông tin.

ID	Tên Use Case	Mô tả
UC08	Đăng xuất	Người dùng đăng xuất khỏi hệ thống.
UC09	Xem danh sách người dùng	Người dùng xem danh sách người dùng trong hệ thống.
UC10	Tìm kiếm người dùng bằng filter	Người dùng thực hiện tìm kiếm tài khoản người dùng bằng filter.
UC11	Đặt trạng thái (active hoặc inactive)	Người dùng thực hiện đặt trạng thái người dùng, trạng thái active tài khoản người dùng hoạt động bình thường, trạng thái inactive tài khoản người dùng không thể sử dụng các chức năng của hệ thống.
UC12	Thêm danh mục sản phẩm	Người dùng thực hiện thêm danh mục sản phẩm mới vào hệ thống.
UC13	Xem danh sách danh mục	Người dùng xem danh sách danh mục trong hệ thống.
UC14	Tìm kiếm danh mục sản phẩm bằng filter	Người dùng thực hiện tìm kiếm danh mục sản phẩm bằng filter.
UC15	Sửa thông tin danh mục	Người dùng thực hiện thao tác thay đổi thông tin danh mục.
UC16	Thêm sản phẩm mới	Người dùng thực hiện thêm sản phẩm mới vào hệ thống.
UC17	Xem danh sách sản phẩm	Người dùng xem danh sách sản phẩm trong hệ thống.
UC18	Tìm kiếm sản phẩm bằng filter	Người dùng thực hiện tìm kiếm sản phẩm bằng filter.

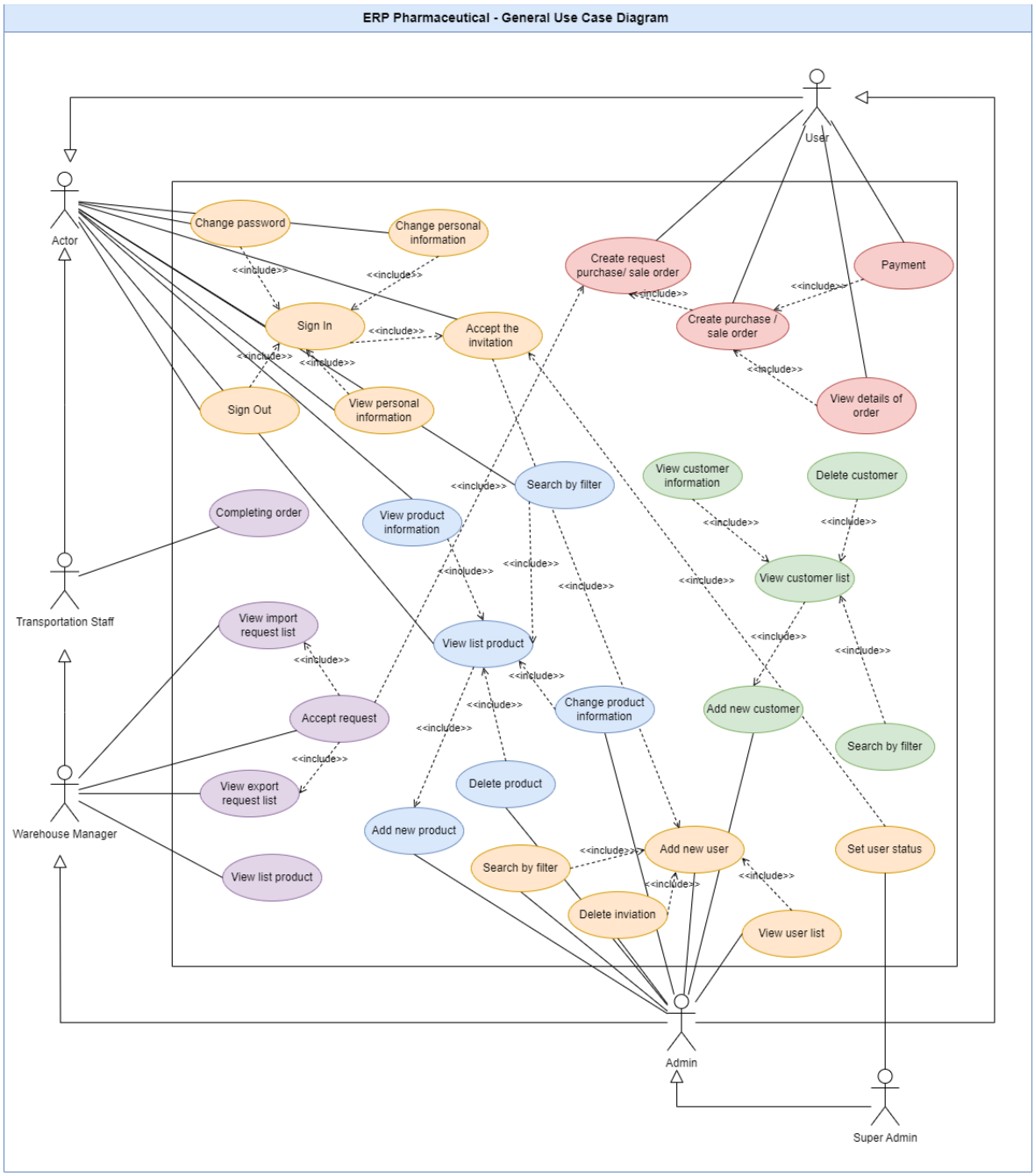
ID	Tên Use Case	Mô tả
UC19	Xem chi tiết sản phẩm	Người dùng xem thông tin chi tiết của sản phẩm trong hệ thống.
UC20	Sửa thông tin sản phẩm	Người dùng thực hiện thao tác thay đổi thông tin sản phẩm.
UC21	Xóa sản phẩm	Người dùng thực hiện thao tác xóa sản phẩm trong hệ thống.
UC22	Thêm khách hàng mới	Người dùng thực hiện thêm khách hàng mới vào hệ thống.
UC23	Xem danh sách khách hàng	Người dùng xem danh sách khách hàng trong hệ thống, lưu ý khách hàng bao gồm nhà sản xuất và nhà thuốc.
UC24	Tìm kiếm khách hàng bằng filter	Người dùng thực hiện tìm kiếm khách hàng bằng filter.
UC25	Xem thông tin chi tiết của khách hàng	Người dùng xem thông tin chi tiết của khách hàng trong hệ thống.
UC26	Sửa thông tin khách hàng	Người dùng thực hiện thao tác thay đổi thông tin khách hàng.
UC27	Xóa khách hàng	Người dùng thực hiện thao tác xóa khách hàng trong hệ thống.
UC28	Tạo yêu cầu mua hàng	Người dùng thực hiện thao tác tạo yêu cầu mua hàng cho nhà sản xuất dược phẩm.
UC29	Tạo đơn mua hàng	Người dùng thực hiện thao tác tạo hóa đơn mua hàng khi nhận được sản phẩm. Đồng thời tạo

ID	Tên Use Case	Mô tả
		yêu cầu nhập hàng cho quản lý kho.
UC30	Thanh toán đơn hàng	Người dùng thực hiện thao tác thanh toán cho đơn hàng.
UC31	Chấp nhận nhập hàng	Người dùng thực hiện thao tác chấp nhận đơn hàng vào kho và lưu lại thông tin đơn hàng.
UC32	Tạo yêu cầu bán hàng	Người dùng thực hiện thao tác tạo yêu cầu bán hàng cho quản lý kho để thực hiện đóng gói và vận chuyển sản phẩm.
UC33	Tạo đơn bán hàng	Người dùng thực hiện thao tác tạo hóa đơn bán hàng khi có khách hàng muốn mua sản phẩm. Đồng thời tạo yêu cầu đóng gói sản phẩm cho quản lý kho.
UC34	Chấp nhận xuất hàng	Người dùng thực hiện thao tác chấp nhận đóng gói đơn hàng và chuyển đơn hàng đến khách hàng.
UC35	Xem danh sách sản phẩm trong kho	Người dùng xem danh sách sản phẩm trong kho, thông tin về số lượng, ngày nhập hàng gần nhất và ngày xuất hàng gần nhất.
UC36	Tìm kiếm sản phẩm trong kho bằng filter	Người dùng thực hiện tìm kiếm sản phẩm trong kho bằng filter.

4.4.2 Sơ đồ Use case tổng quát



Hình 4.4: Sơ đồ Use Case tổng quát 1



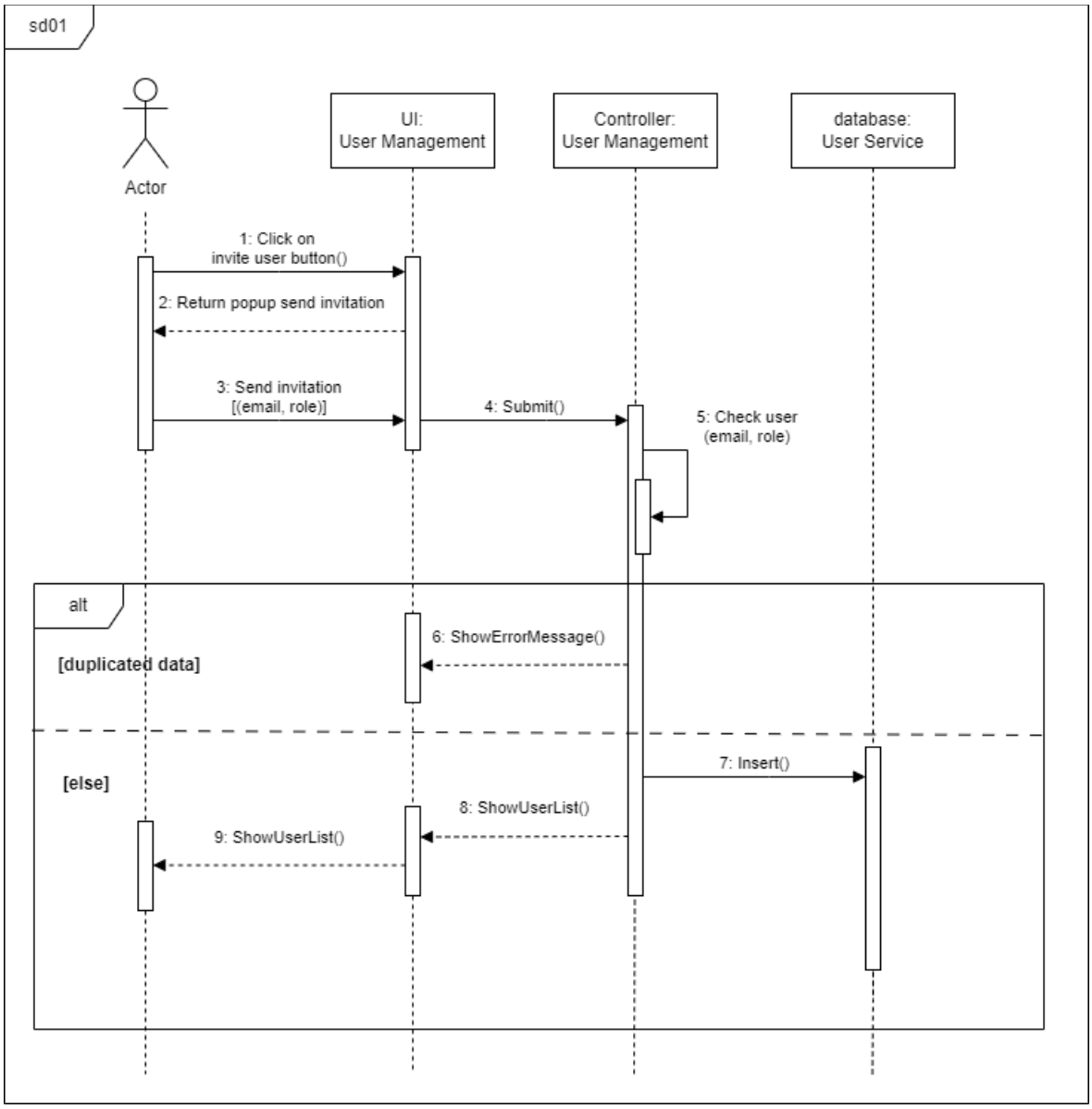
Hình 4.5: Sơ đồ Use Case tổng quát 2

4.5 Đặc tả các Use case

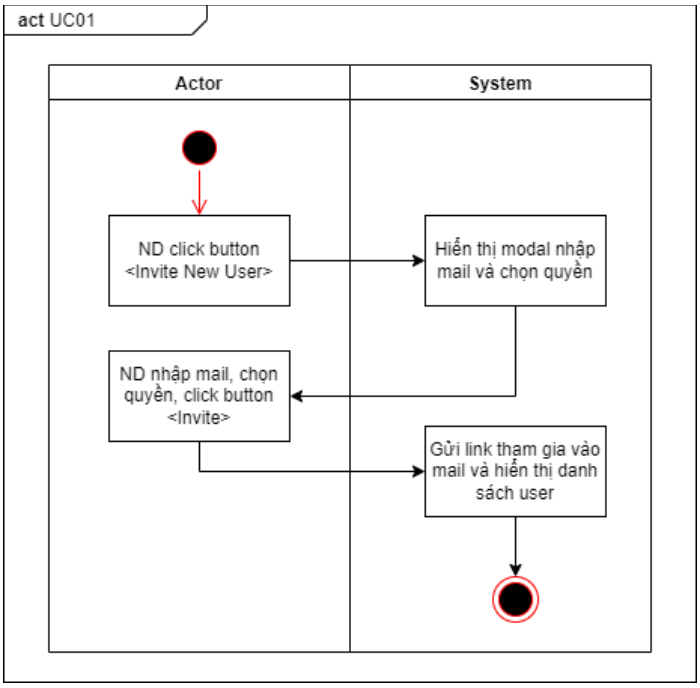
4.5.1 Use case thêm người dùng

Bảng 4.2: Đặc tả use case thêm người dùng

Mã use case	UC01	
Tên use case	Thêm người dùng	
Ngữ cảnh	Trên website khi đã đăng nhập thành công vào hệ thống.	
Mô tả	Người dùng thực hiện thêm người dùng vào hệ thống bằng cách gửi mail và phân quyền.	
Tác nhân	Admin và Super Admin	
Sự kiện kích hoạt	Người dùng click vào button <Invite new user>	
Điều kiện tiên quyết	Người dùng đã đăng nhập vào website thành công.	
Kết quả	Gửi lời mời thành công.	
Luồng sự kiện	**Actor**	**System**
	1. Người dùng click vào button <Invite new user>	1.1. Hệ thống hiển thị modal nhập mail và chọn quyền
	2. Người dùng nhập mail và chọn quyền, sau đó click button <Invite>	2.1. Hệ thống gửi link tham gia hệ thống vào mail được nhập 2.2. Hệ thống hiển thị danh sách user.
Ngoại lệ	Không có	



Hình 4.6: Sequence diagram của UC01



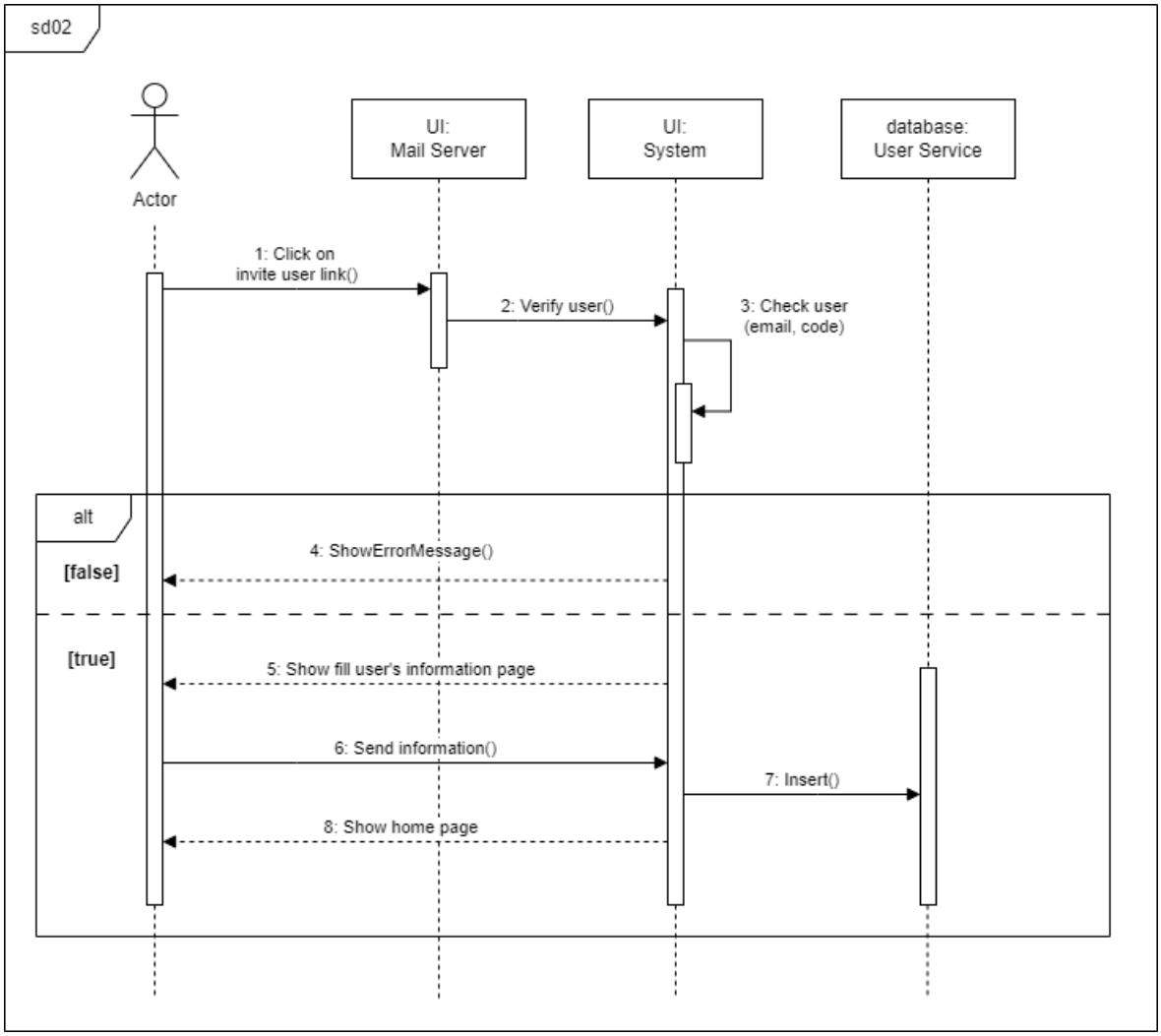
Hình 4.7: Activity diagram của UC01

4.5.2 Use case chấp nhận lời mời và thêm thông tin người dùng

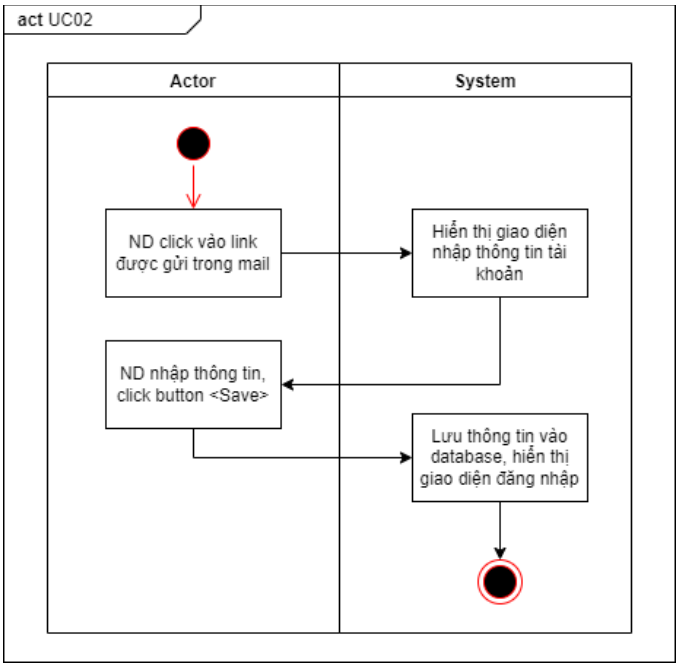
Bảng 4.3: Đặc tả use case chấp nhận lời mời và thêm thông tin người dùng

Mã use case	UC02	
Tên use case	Chấp nhận lời mời và thêm thông tin người dùng	
Ngữ cảnh	Trên giao diện mail đã nhận, người dùng click vào link nhập thông tin user.	
Mô tả	Người dùng thực hiện thêm thông tin và click button <Save> để tham gia vào hệ thống.	
Tác nhân	New User	
Sự kiện kích hoạt	Người dùng click vào link được gửi trong mail.	
Điều kiện tiên quyết	Người dùng đã được Super Admin hoặc Admin của hệ thống gửi lời mời.	
Kết quả	Tham gia vào hệ thống thành công.	
Luồng sự kiện	**Actor**	**System**

	1. Người dùng click vào link được gửi trong mail	1.1. Hệ thống hiển thị giao diện nhập thông tin tài khoản.
	2. Người dùng nhập thông tin, sau đó click button <Save>	2.1. Hệ thống lưu thông tin người dùng vào database. 2.2. Hệ thống hiển thị giao diện đăng nhập thành công.
Ngoại lệ	Không có	



Hình 4.8: Sequence diagram của UC02



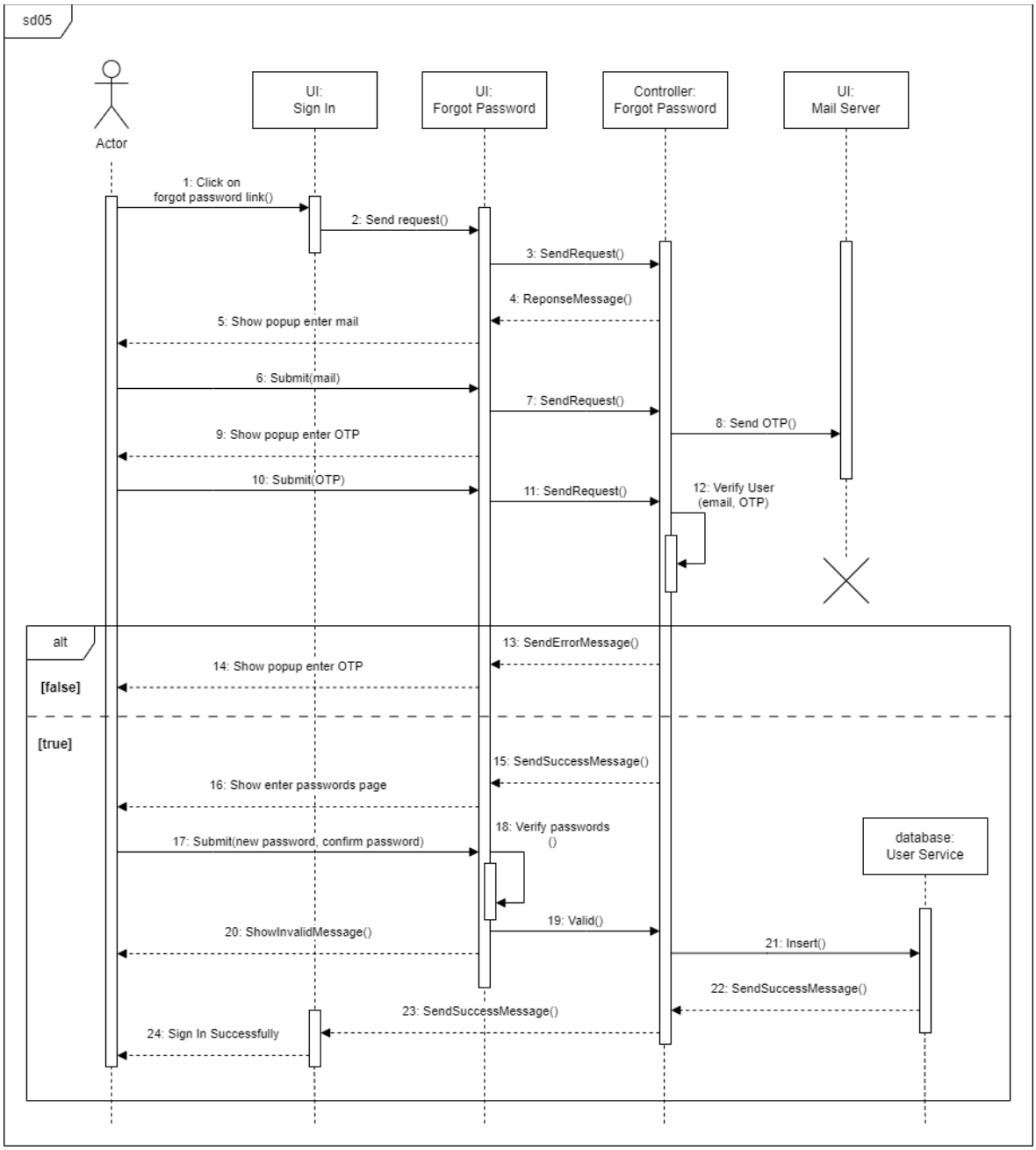
Hình 4.9: Activity diagram của UC02

4.5.3 Use case quên mật khẩu

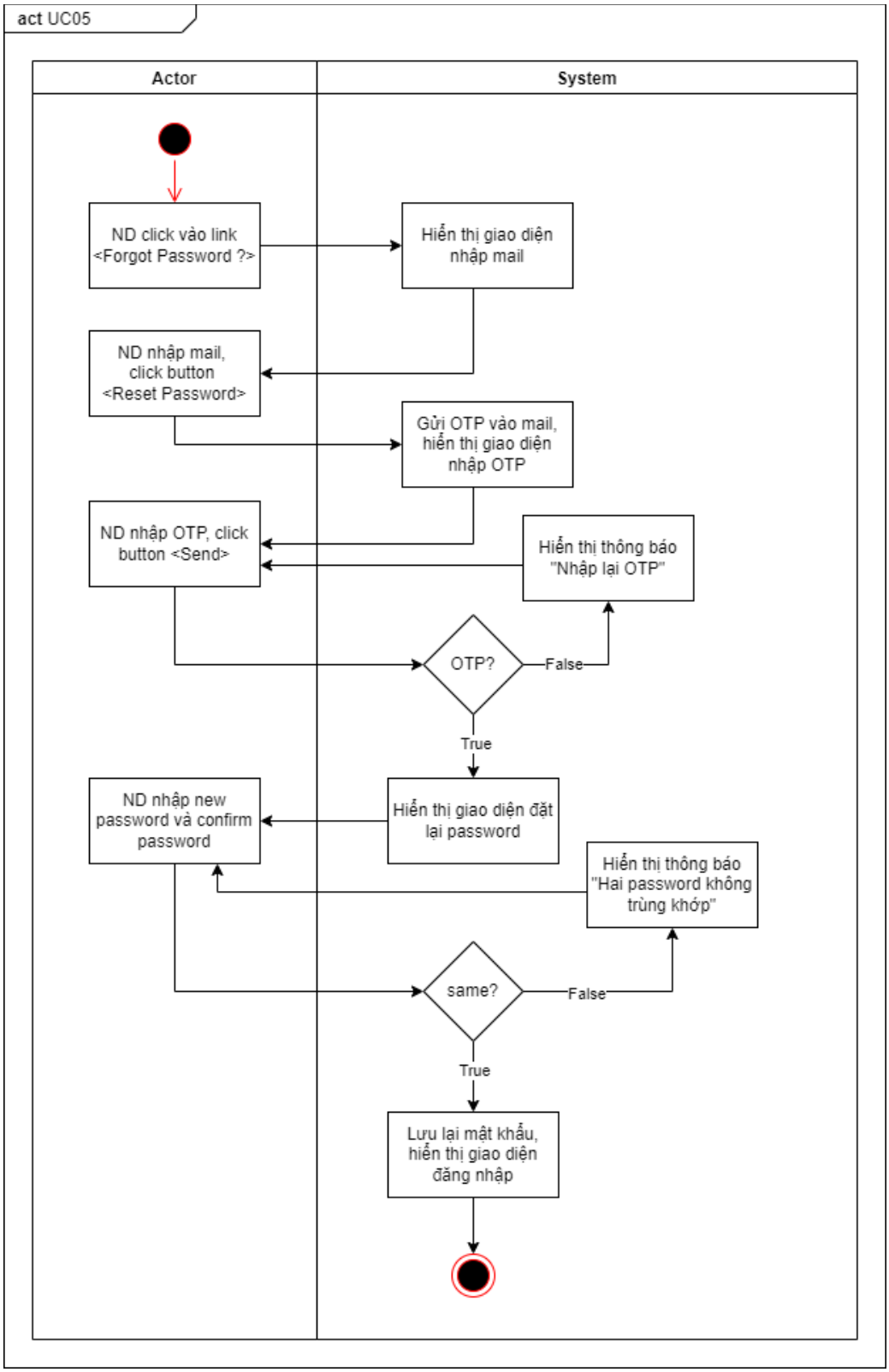
Bảng 4.4: Đặc tả use case quên mật khẩu

Mã use case	UC05
Tên use case	Quên mật khẩu
Ngữ cảnh	Trên giao diện đăng nhập, người dùng click vào link <Forgot password ?>
Mô tả	Người dùng sử dụng chức năng quên mật khẩu để nhập mail đã tham gia vào hệ thống, nhận OTP được gửi vào mail, nhập OTP đã nhận và nhập mật khẩu mới.
Tác nhân	User, Transportation Staff, Warehouse Manager, Admin và Super Admin
Sự kiện kích hoạt	Người dùng click vào link <Forgot password ?>
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đã tham gia vào hệ thống.
Kết quả	Người dùng đặt mật khẩu mới thành công.

	Actor	System
Luồng sự kiện	1. Người dùng click vào link <Forgot password ?> trong form đăng nhập.	1.1. Hệ thống hiển thị giao diện nhập mail.
	2. Người dùng nhập mail và click button <Reset Password>	2.1. Hệ thống gửi OTP vào mail. 2.2. Hệ thống hiển thị giao diện nhập OTP
	3. Người dùng nhập OTP và click button <Send>	3.1. Hệ thống kiểm tra OTP: 3.1.1. OTP đúng: Hệ thống hiển thị giao diện đặt lại password 3.1.2. OTP sai: Hệ thống thông báo "Nhập lại OTP"
	4. Người dùng nhập password mới và xác nhận password.	4.1. Hệ thống kiểm tra hai password: 4.1.1. Giống nhau: hệ thống lưu lại mật khẩu và hiển thị giao diện đăng nhập 4.1.2 Khác nhau: Yêu cầu nhập lại
Ngoại lệ	- Người dùng nhập mail chưa tham gia vào hệ thống. - Người dùng nhập sai OTP. - Người dùng nhập hai password không giống nhau.	



Hình 4.10: Sequence diagram của UC05

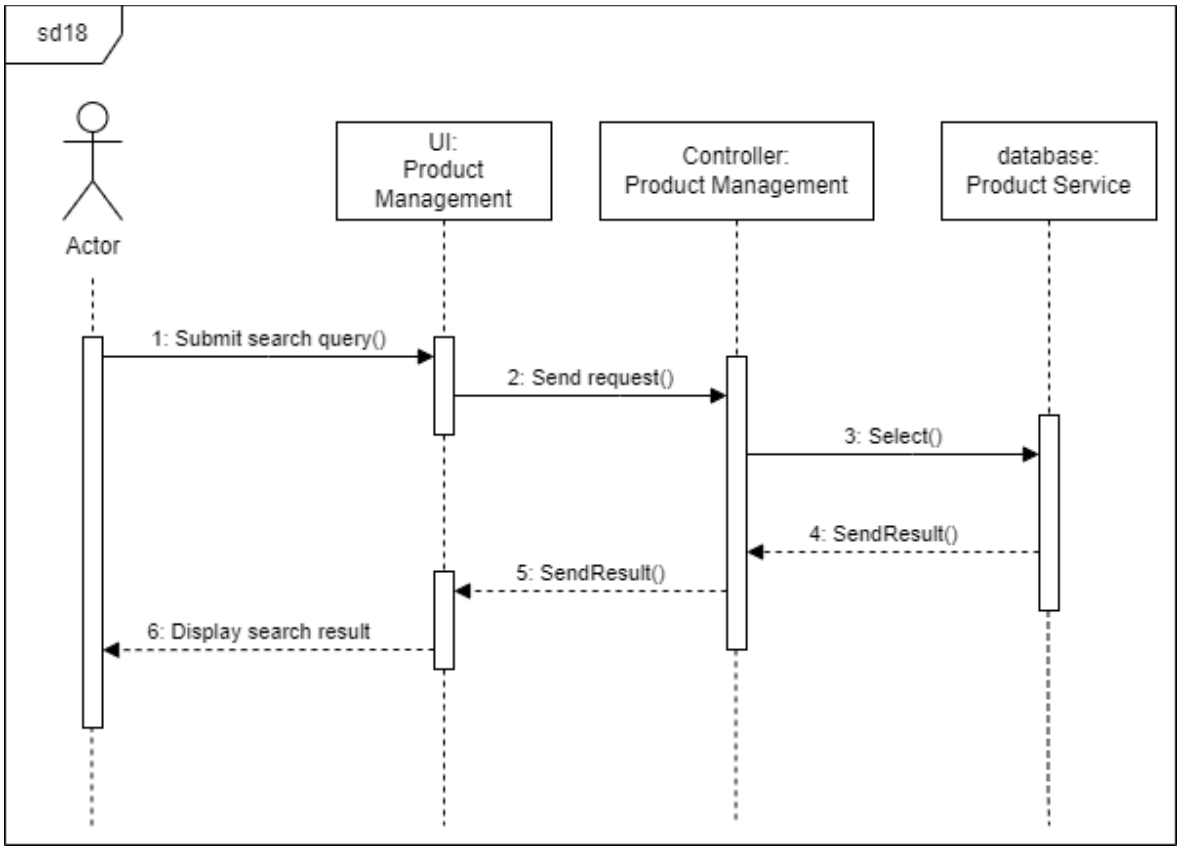


Hình 4.11: Activity diagram của UC05

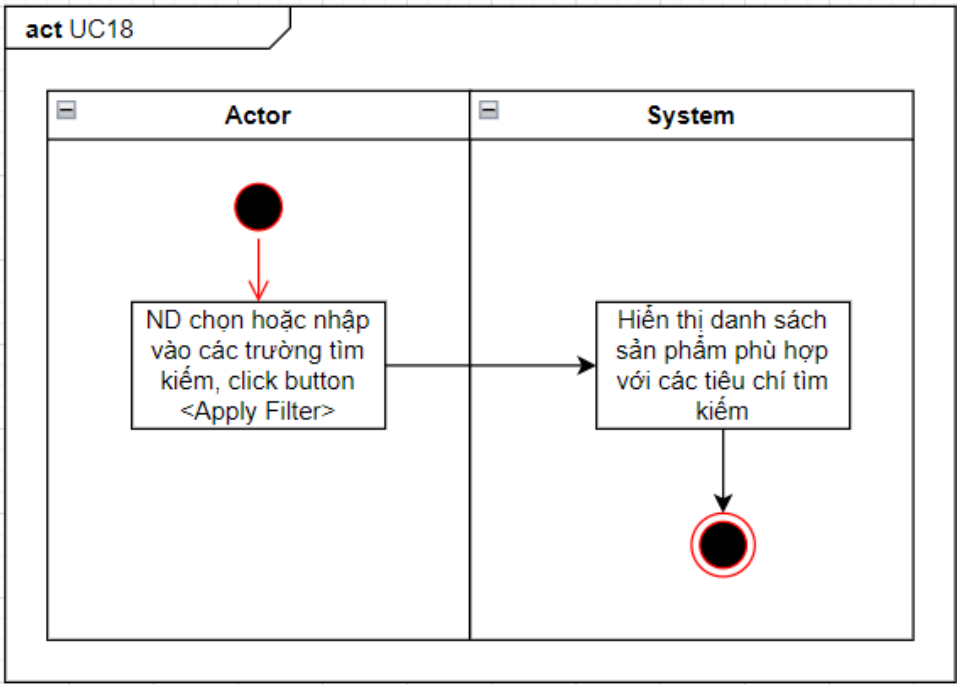
4.5.4 Use case tìm kiếm sản phẩm bằng filter

Bảng 4.5: Đặc tả use case tìm kiếm bằng filter

Mã use case	UC18	
Tên use case	Tìm kiếm sản phẩm bằng filter	
Ngữ cảnh	Trên giao diện sản phẩm, người dùng chọn hoặc nhập vào các trường tìm kiếm như: category, manufacturer, min price, max price, search, sắp xếp và click vào button <Apply Filter> để thực hiện tìm kiếm.	
Mô tả	Người dùng sử dụng chức năng tìm kiếm bằng filter để tìm và lọc ra các sản phẩm phù hợp với nhu cầu.	
Tác nhân	User, Transportation Staff, Warehouse Manager, Admin và Super Admin	
Sự kiện kích hoạt	Người dùng click vào button <Apply Filter>	
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đăng nhập thành công vào hệ thống.	
Kết quả	Người dùng tìm sản phẩm thành công.	
Luồng sự kiện	**Actor**	**System**
	1. Người dùng chọn hoặc nhập vào các trường tìm kiếm và click vào button <Apply Filter>	1.1. Hệ thống hiển thị danh sách sản phẩm phù hợp với các tiêu chí tìm kiếm
Ngoại lệ	Không có	



Hình 4.12: Sequence diagram của UC18



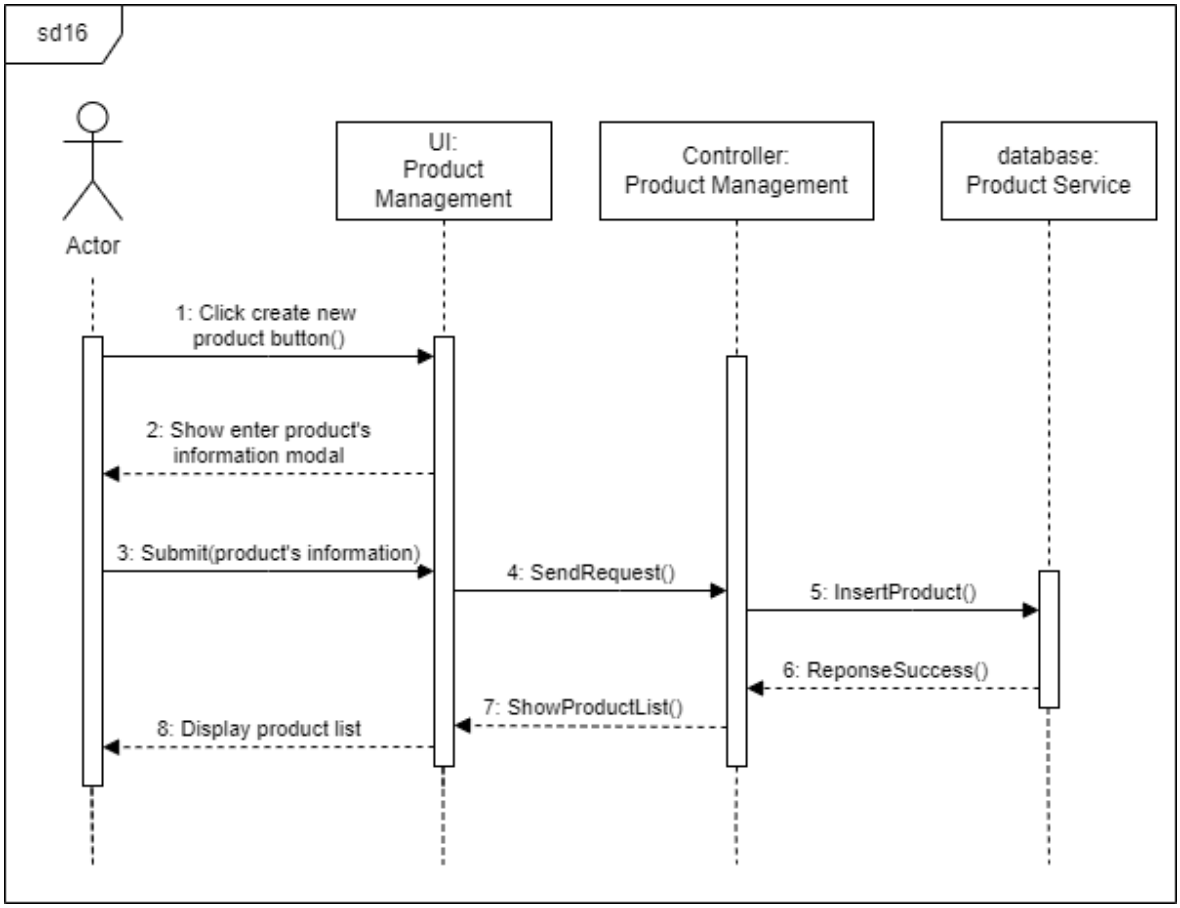
Hình 4.13: Activity diagram của UC18

4.5.5 Use case thêm sản phẩm mới

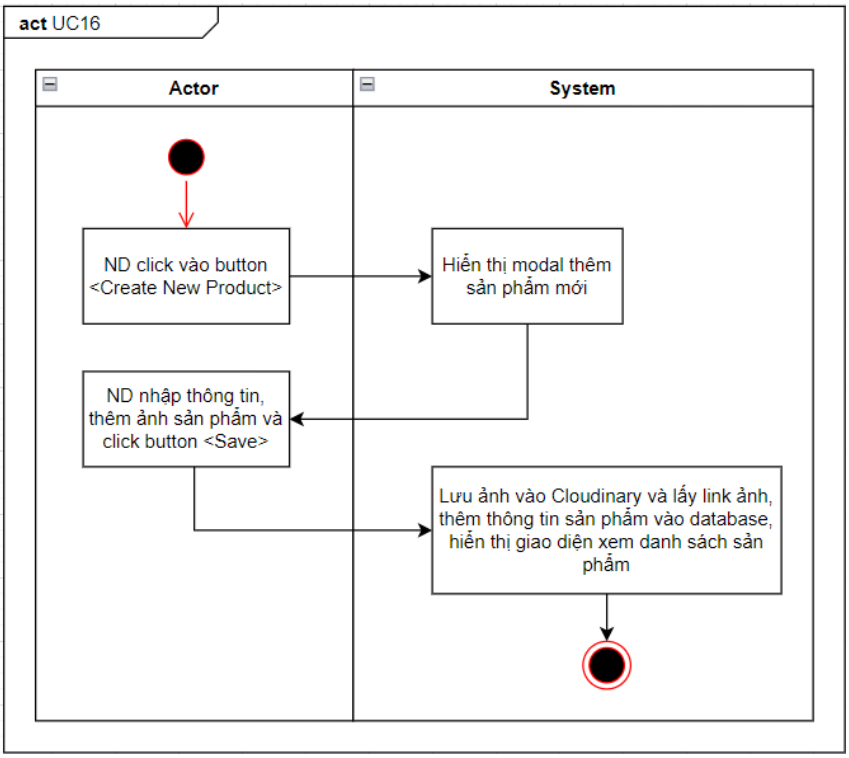
Bảng 4.6: Đặc tả use case thêm sản phẩm mới

Mã use case	UC16	
Tên use case	Thêm sản phẩm mới	
Ngữ cảnh	Trên giao diện quản lý sản phẩm, người dùng click vào button <Create New Product> để thực hiện thêm sản phẩm mới.	
Mô tả	Người dùng sử dụng chức năng thêm sản phẩm mới để thêm sản phẩm vào hệ thống.	
Tác nhân	Admin và Super Admin	
Sự kiện kích hoạt	Người dùng click vào button < Create New Product >	
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đăng nhập thành công vào hệ thống.	
Kết quả	Người dùng thêm sản phẩm mới thành công.	
Luồng sự kiện	**Actor**	**System**
	1. Người dùng click vào button < Create New Product >	1.1. Hệ thống hiển thị modal thêm sản phẩm mới.
	2. Người dùng thêm ảnh sản phẩm, nhập thông tin sản phẩm và click button <Save>	2.1. Hệ thống lưu ảnh sản phẩm tại Cloudinary và trả về đường link xem ảnh. 2.2. Hệ thống lưu thông tin sản phẩm vào database. 2.3 Hệ thống hiển thị giao diện xem danh sách sản phẩm sau khi đã cập nhật database.

Ngoại lệ	Người dùng không thêm ảnh hoặc nhập thiếu thông tin sản phẩm.



Hình 4.14: Sequence diagram của UC16



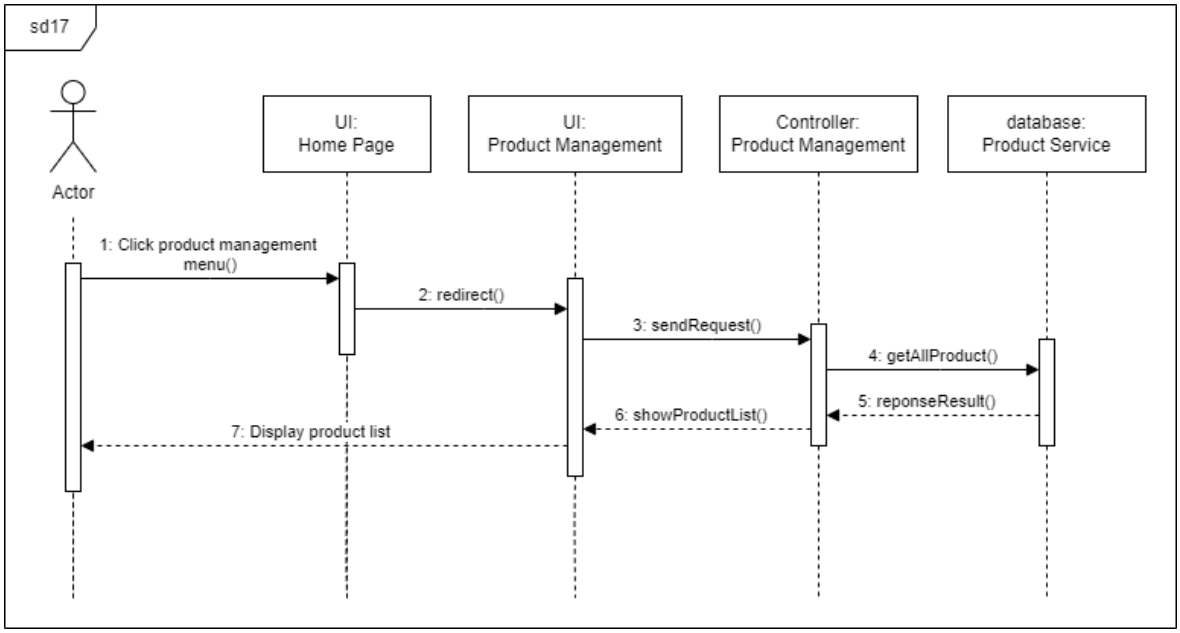
Hình 4.15: Activity diagram của UC16

4.5.6 Use case xem danh sách sản phẩm

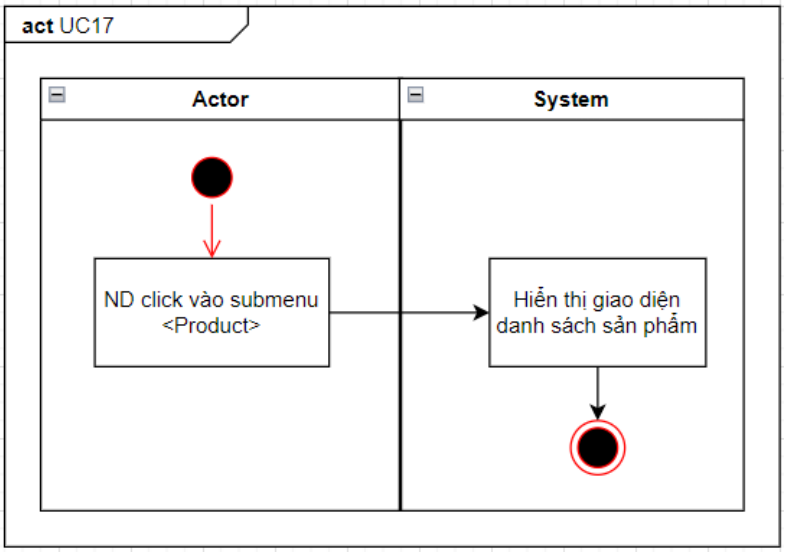
Bảng 4.7: Đặc tả use case xem danh sách sản phẩm

Mã use case	UC17
Tên use case	Xem danh sách sản phẩm
Ngữ cảnh	Trên website khi đã đăng nhập thành công vào hệ thống
Mô tả	Người dùng xem danh sách các sản phẩm hiện có trong hệ thống
Tác nhân	Admin, Super Admin, Warehouse Manager, Transportation Staff, User
Sự kiện kích hoạt	Người dùng click vào button < Product >
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đăng nhập thành công vào hệ thống.
Kết quả	Hiển thị danh sách sản phẩm.

	Actor	System
Luồng sự kiện	1. Người dùng click vào < Product > tại thanh menu	1.1. Hệ thống hiển thị danh sách sản phẩm
Ngoại lệ	Không có	



Hình 4.16: Sequence diagram của UC17

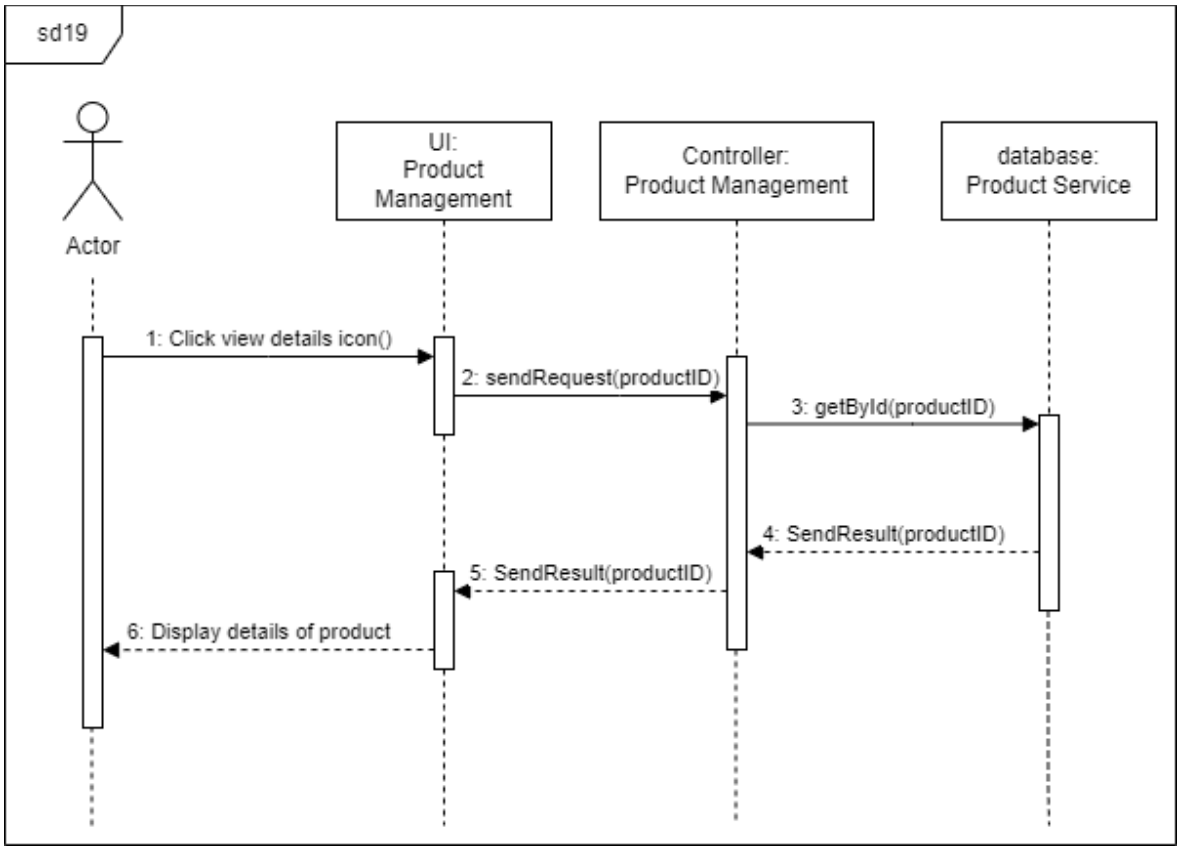


Hình 4.17: Activity diagram của UC17

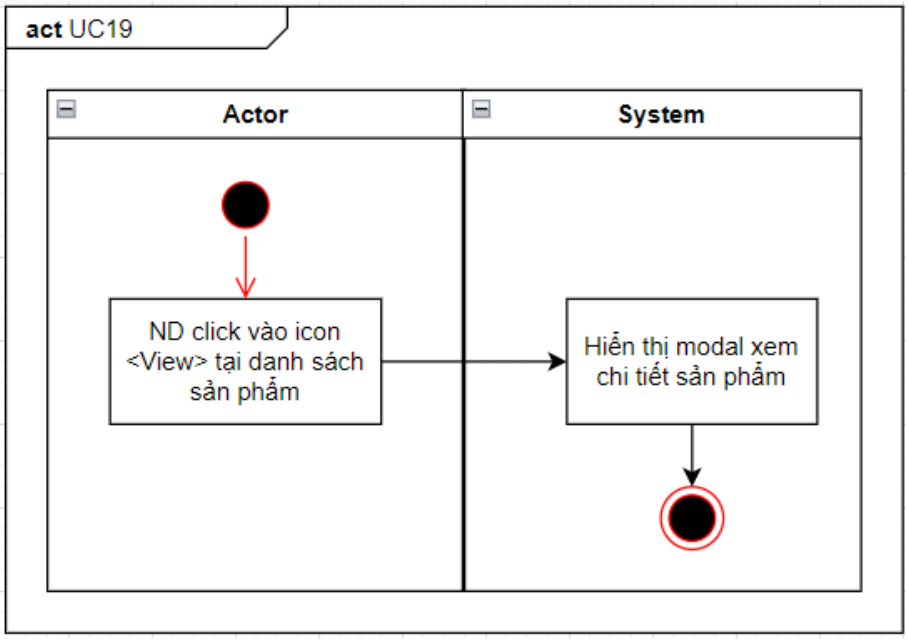
4.5.7 Use case xem chi tiết sản phẩm

Bảng 4.8: Đặc tả use case xem chi tiết sản phẩm

Mã use case	UC19	
Tên use case	Xem chi tiết sản phẩm	
Ngữ cảnh	Trên website khi đã đăng nhập thành công vào hệ thống	
Mô tả	Người dùng xem chi tiết sản phẩm hiện có trong hệ thống	
Tác nhân	Admin, Super Admin, Warehouse Manager, Transportation Staff, User	
Sự kiện kích hoạt	Người dùng click vào một sản phẩm bất kỳ trong danh sách sản phẩm	
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đăng nhập thành công vào hệ thống.	
Kết quả	Hiển thị chi tiết sản phẩm.	
Luồng sự kiện	**Actor**	**System**
	1. Người dùng click vào icon Xem chi tiết tại sản phẩm trong danh sách sản phẩm	1.1. Hệ thống hiển thị chi tiết sản phẩm
Ngoại lệ	Không có	



Hình 4.18: Sequence diagram của UC19

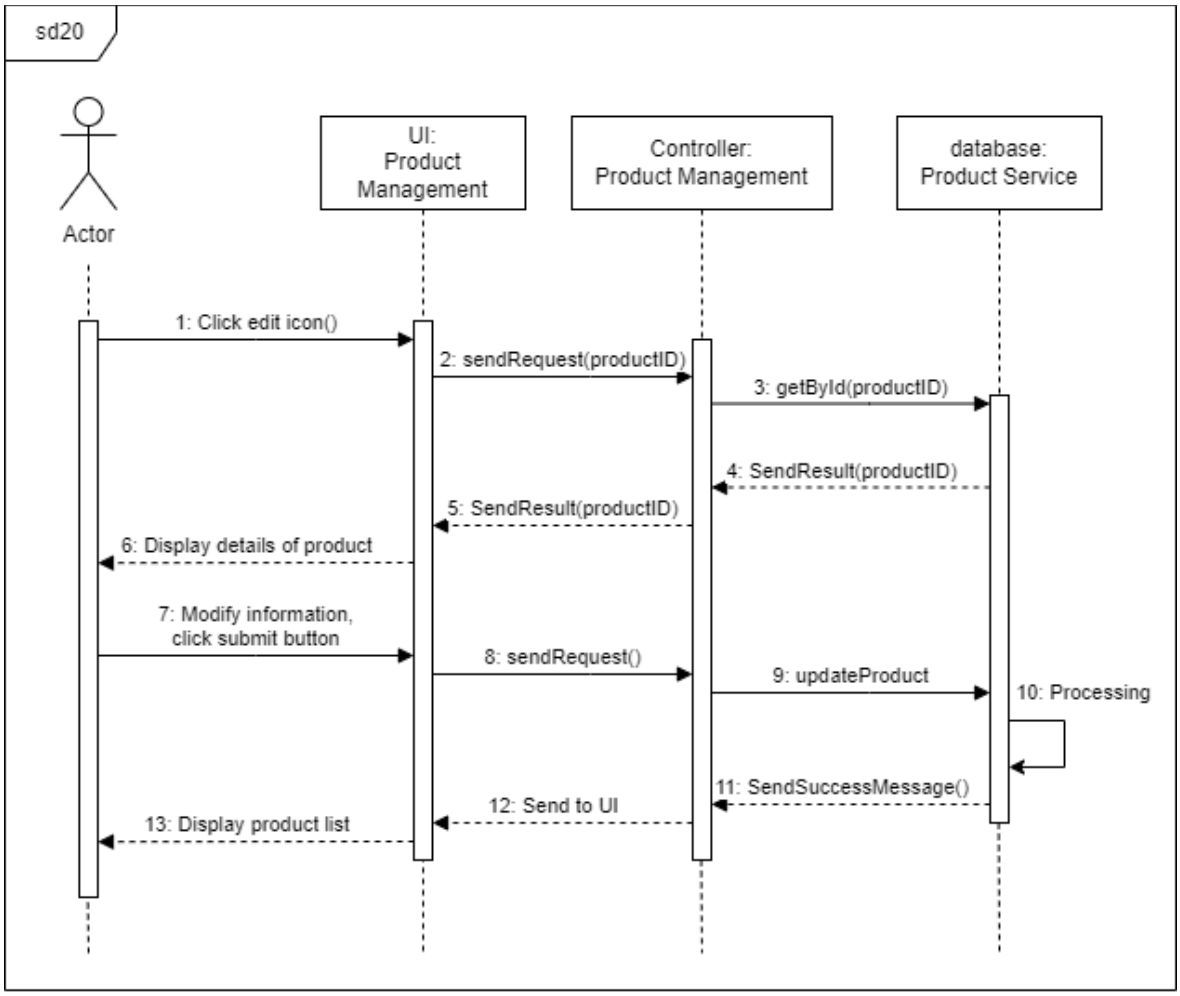


Hình 4.19: Activity diagram của UC19

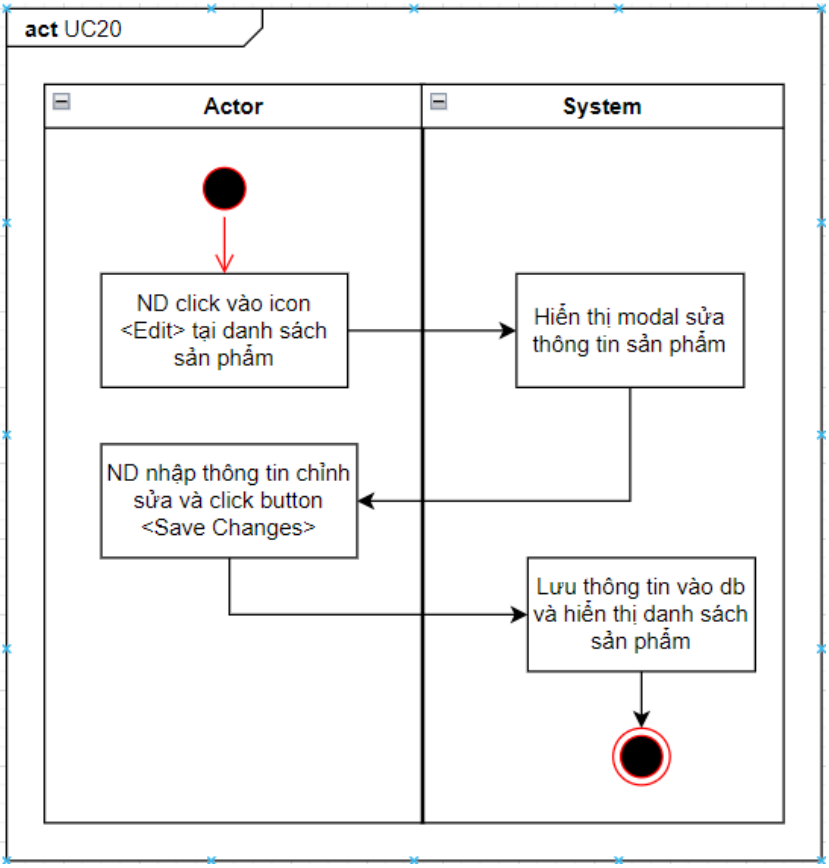
4.5.8 Use case sửa thông tin sản phẩm

Bảng 4.9: Đặc tả use case sửa thông tin sản phẩm

Mã use case	UC20	
Tên use case	Sửa thông tin sản phẩm	
Ngữ cảnh	Trên website khi đã đăng nhập thành công vào hệ thống	
Mô tả	Người dùng thực hiện sửa thông tin sản phẩm trong hệ thống	
Tác nhân	Admin và Super Admin	
Sự kiện kích hoạt	Người dùng click vào một sản phẩm bất kỳ trong danh sách sản phẩm	
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đăng nhập thành công vào hệ thống.	
Kết quả	Thông tin sản phẩm được cập nhật	
Luồng sự kiện	**Actor**	**System**
	1. Người dùng click vào danh mục Management và click vào Product	1.1. Hệ thống hiển thị danh sách sản phẩm.
	2. Người dùng click icon <Edit>	2.1. Hệ thống hiển thị chi tiết thông tin sản phẩm cần chỉnh sửa
	3. Người dùng chỉnh sửa thông tin sản phẩm và click button <Save Changes>	3.1. Hệ thống lưu dữ liệu vào database và trả về danh sách sản phẩm sau khi đã chỉnh sửa
Ngoại lệ	Không có	



Hình 4.20: Sequence diagram của UC20



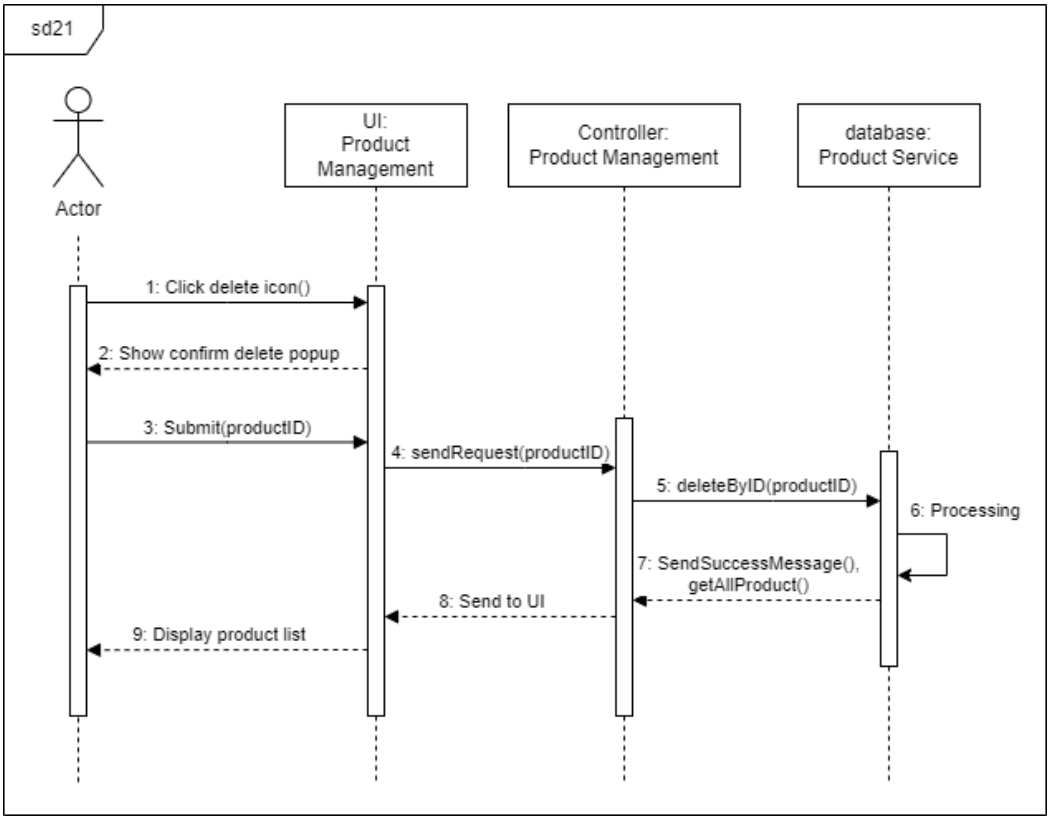
Hình 4.21: Activity diagram của UC20

4.5.9 Use case xóa sản phẩm

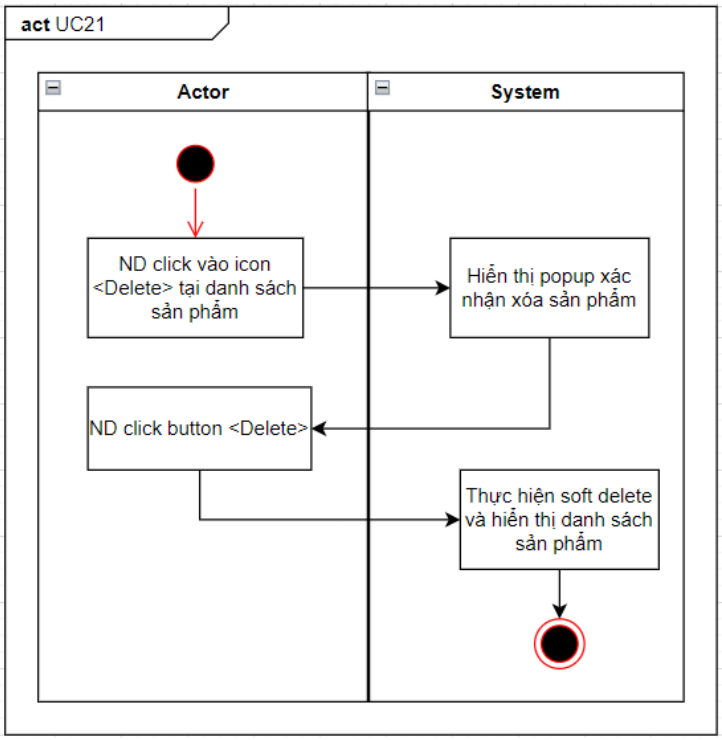
Bảng 4.10: Đặc tả use case đặt xóa sản phẩm

Mã use case	UC21
Tên use case	Xóa sản phẩm
Ngữ cảnh	Trên website khi đã đăng nhập thành công vào hệ thống
Mô tả	Người dùng thực hiện xóa sản phẩm trong hệ thống
Tác nhân	Admin và Super Admin
Sự kiện kích hoạt	Người dùng click vào icon <Delete> trên trang danh sản phẩm
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đăng nhập thành công vào hệ thống.

Kết quả	Thông tin sản phẩm được cập nhật	
Luồng sự kiện	**Actor**	**System**
	1. Người dùng click vào danh mục Management và click vào Product 2. Người dùng click icon <Delete> 3. Người dùng click vào button <Delete>	1.1. Hệ thống hiển thị danh sách sản phẩm. 2.1. Hệ thống hiển thị popup Confirm Delete 3.1. Hệ thống ẩn sản phẩm và trả về danh sách sản phẩm sau khi đã xóa.
Ngoại lệ	Không có	



Hình 4.22: Sequence diagram của UC21



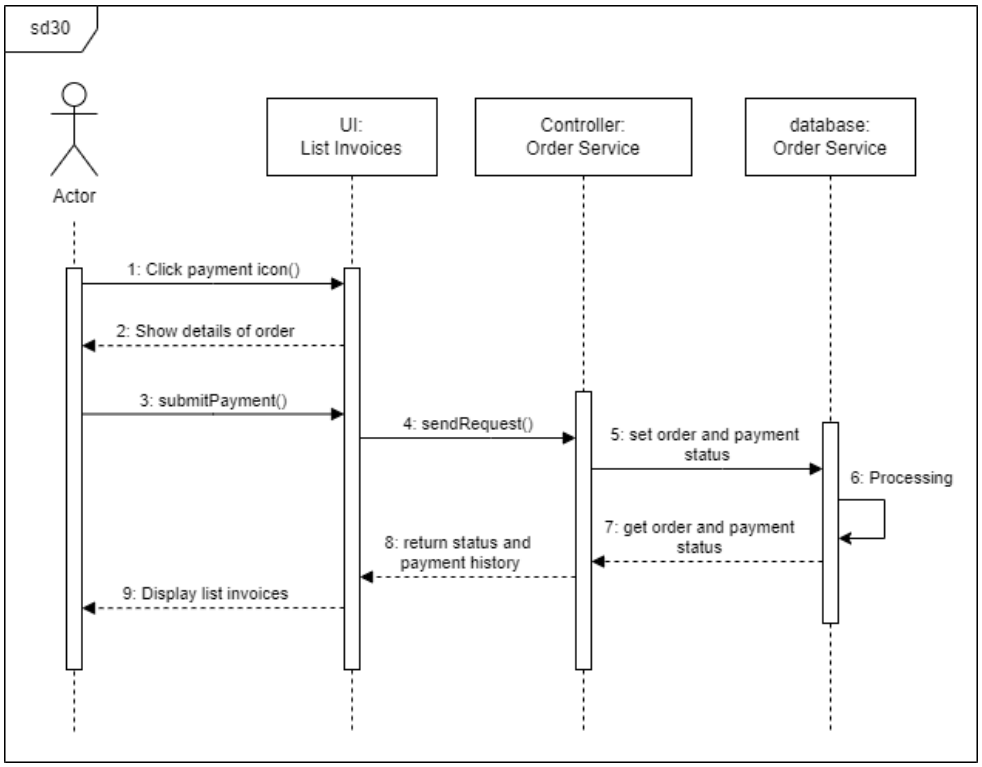
Hình 4.23: Activity diagram của UC21

4.5.10 Use case thanh toán đơn hàng

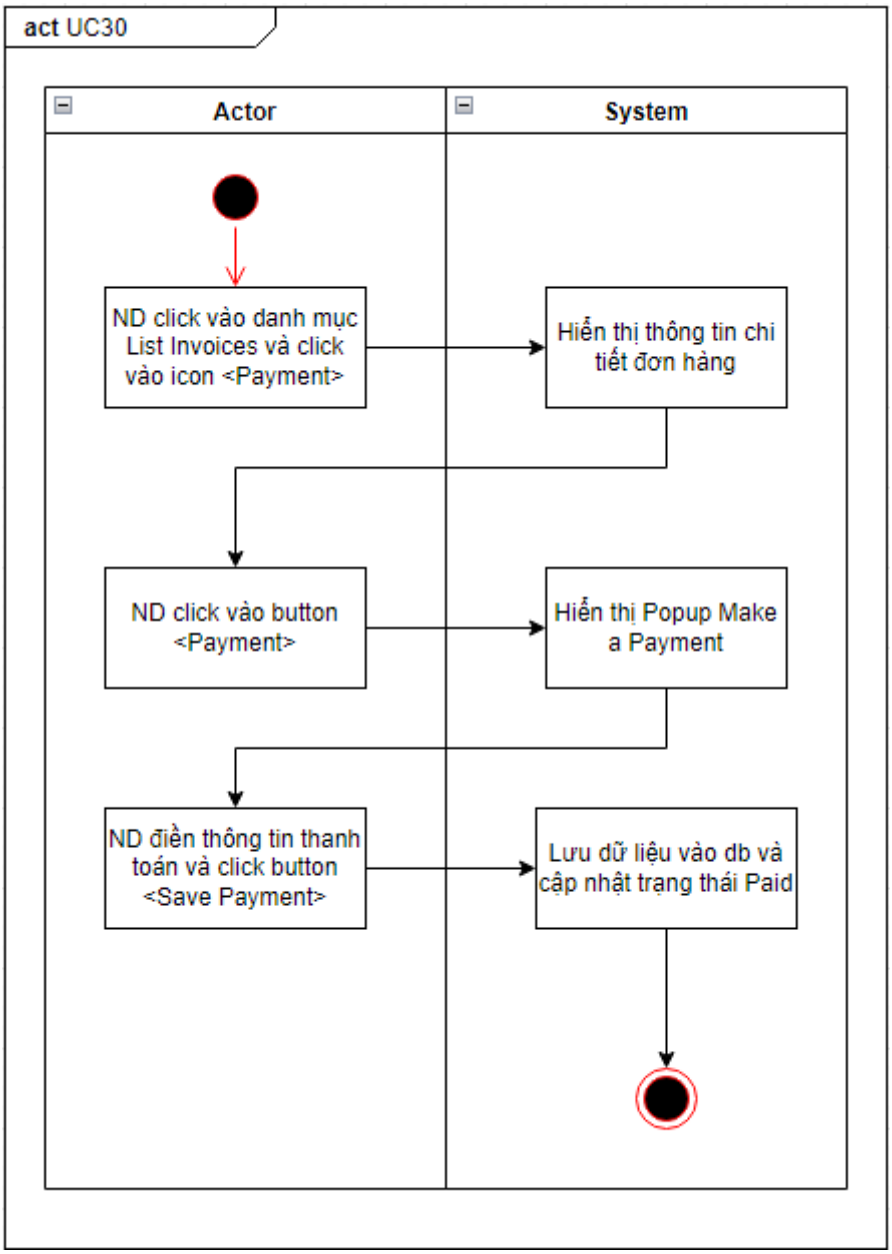
Bảng 4.11: Đặc tả use case thanh toán đơn hàng

Mã use case	UC30	
Tên use case	Thanh toán đơn hàng	
Ngữ cảnh	Trên website khi đã đăng nhập thành công vào hệ thống	
Mô tả	Người dùng thực hiện thanh toán đơn hàng trong hệ thống	
Tác nhân	Admin, Super Admin, User	
Sự kiện kích hoạt	Người dùng click vào icon <Payment> tại danh sách hóa đơn	
Điều kiện tiên quyết	- Người dùng truy cập thành công vào website. - Người dùng đăng nhập thành công vào hệ thống.	
Kết quả	Thanh toán đơn hàng thành công	
Luồng sự kiện	**Actor**	**System**

	1. Người dùng click vào danh mục List Invoices và click vào icon <Payment> 2. Người dùng click button <Payment> 3. Người dùng điền thông tin thanh toán và click button <Save Payment>	1.1. Hệ thống hiển thị trang thông tin chi tiết đơn hàng. 2.1. Hệ thống hiển thị popup Make a Payment 3.1. Hệ thống lưu dữ liệu vào database và kiểm tra số tiền thanh toán: 3.1.1. Nếu số tiền thanh toán nhỏ hơn số tiền trong hóa đơn thì vẫn giữ trạng thái "Unpaid" 3.1.2. Ngược lại, chuyển trạng thái đơn hàng thành "Paid"
Ngoại lệ	Không có	



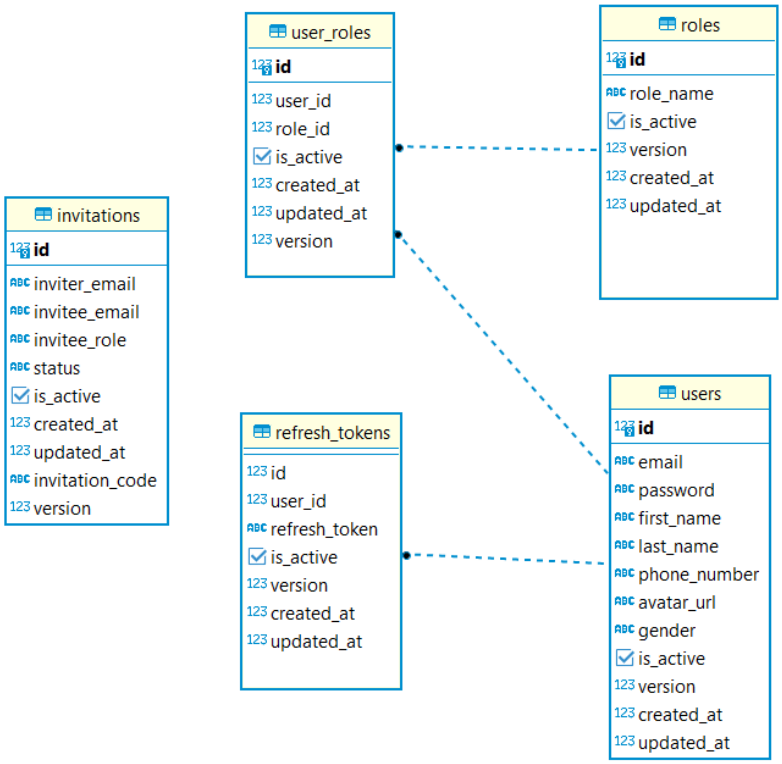
Hình 4.24: Sequence diagram của UC30



Hình 4.25: Activity diagram của UC30

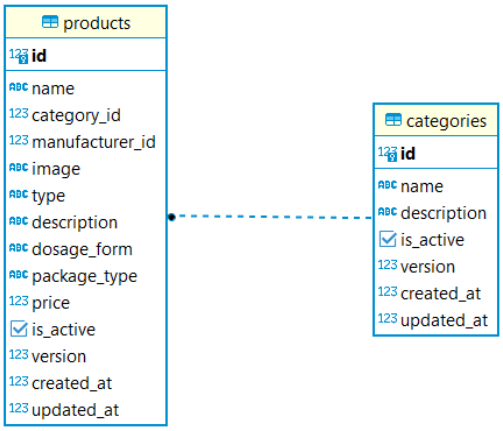
4.6 Thiết kế cơ sở dữ liệu

4.6.1 Cơ sở dữ liệu của user service



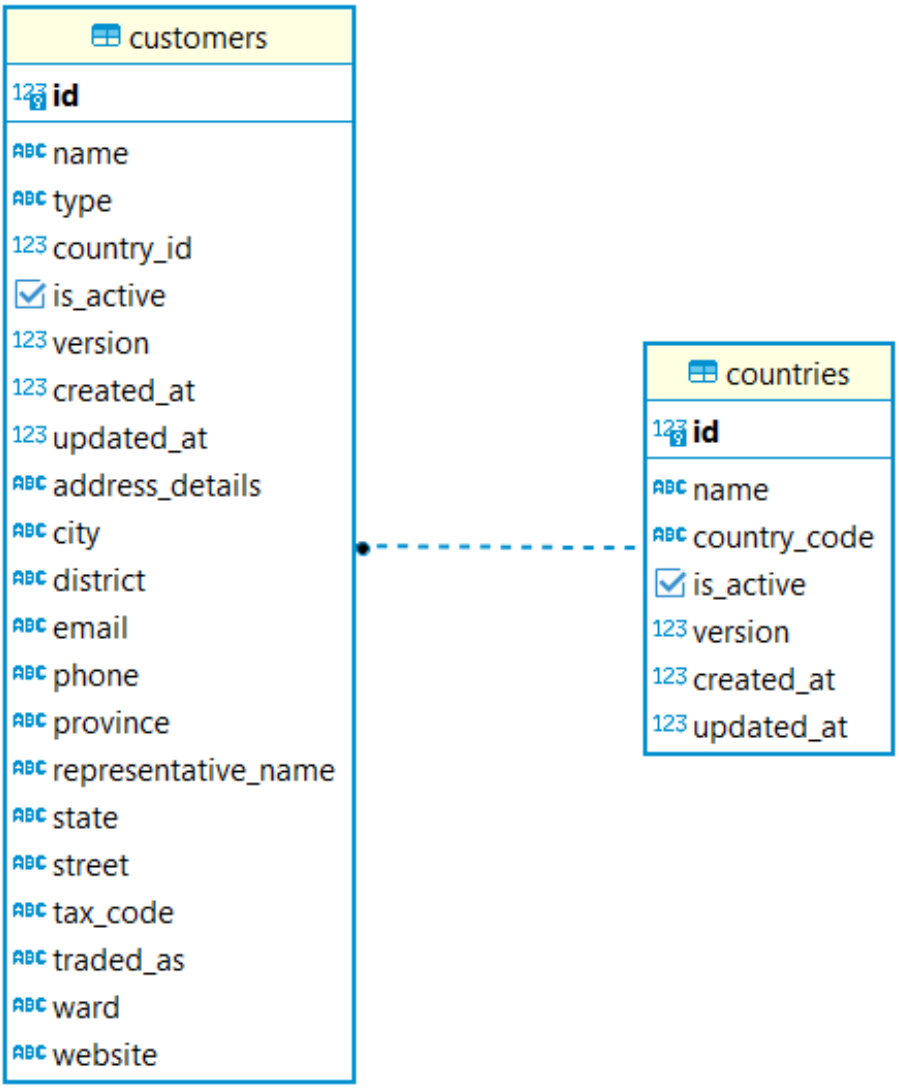
Hình 4.26: Cơ sở dữ liệu của User Service

4.6.2 Cơ sở dữ liệu của product service



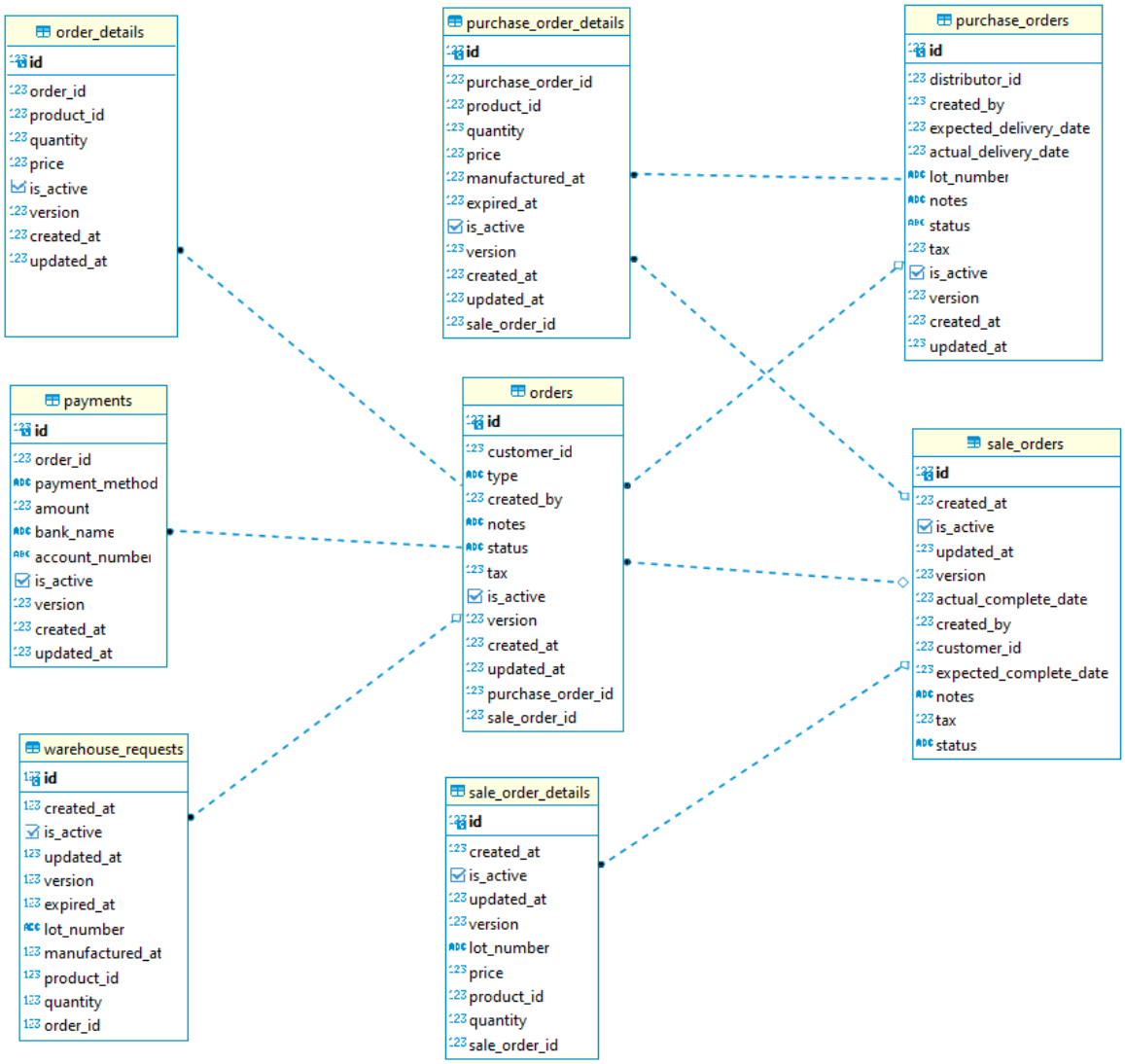
Hình 4.27: Cơ sở dữ liệu của Product Service

4.6.3 Cơ sở dữ liệu của customer service



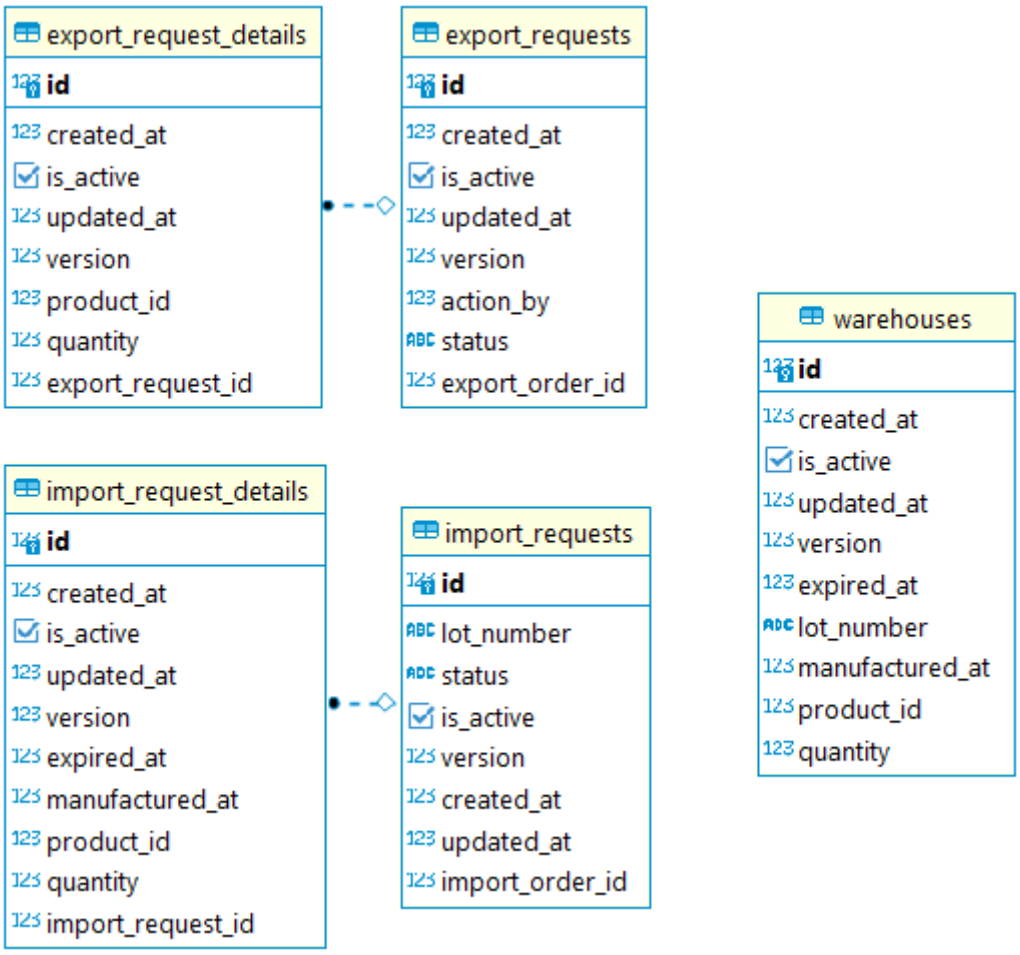
Hình 4.28: Cơ sở dữ liệu của Product Service

4.6.4 Cơ sở dữ liệu của order service



Hình 4.29: Cơ sở dữ liệu của Order Service

4.6.5 Cơ sở dữ liệu của warehouse service

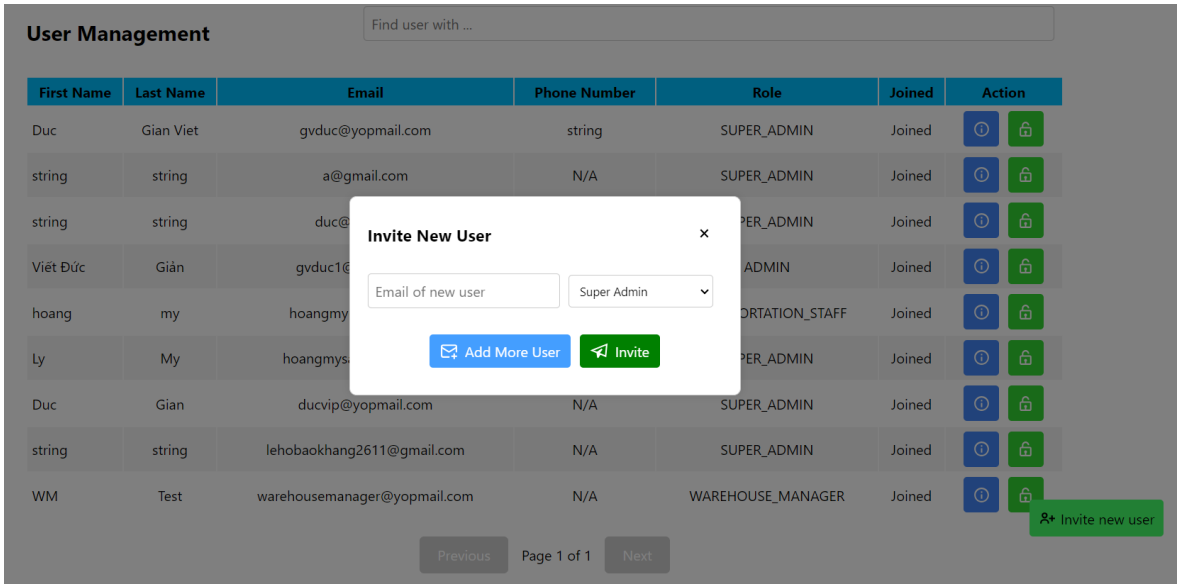


Hình 4.30: Cơ sở dữ liệu của Warehouse Service

CHƯƠNG 5. THỰC THI HỆ THỐNG

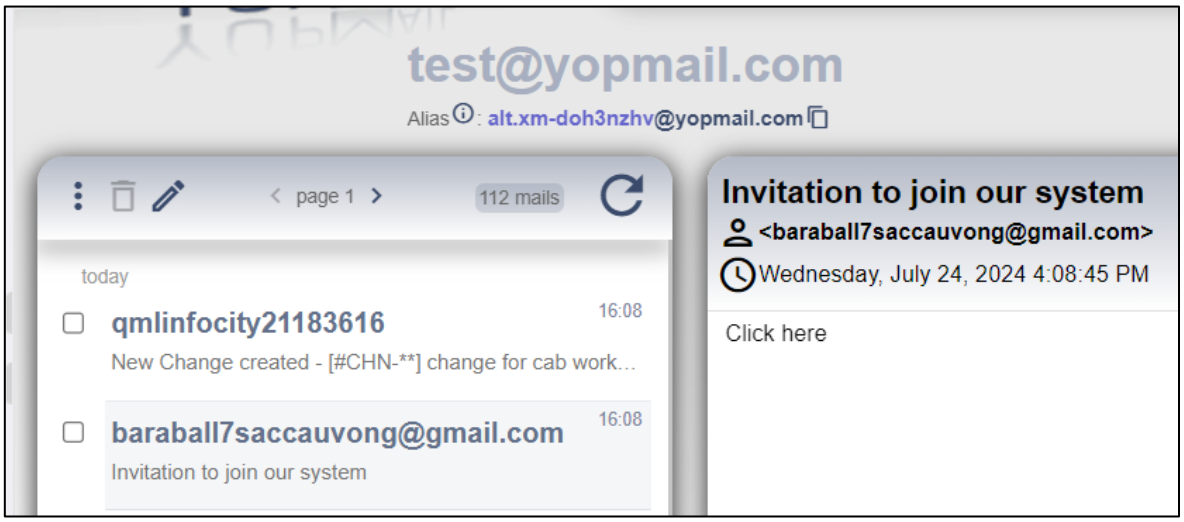
5.1 Phần giao diện

5.1.1 Giao diện thêm người dùng

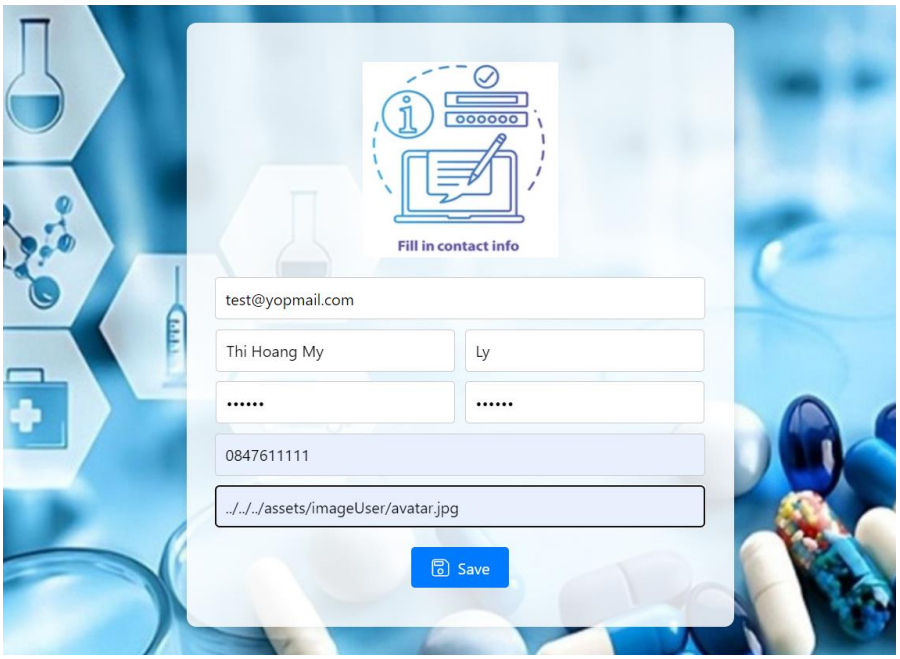


Hình 5.1: Giao diện thêm người dùng

5.1.2 Giao diện chấp nhận lời mời và thêm thông tin

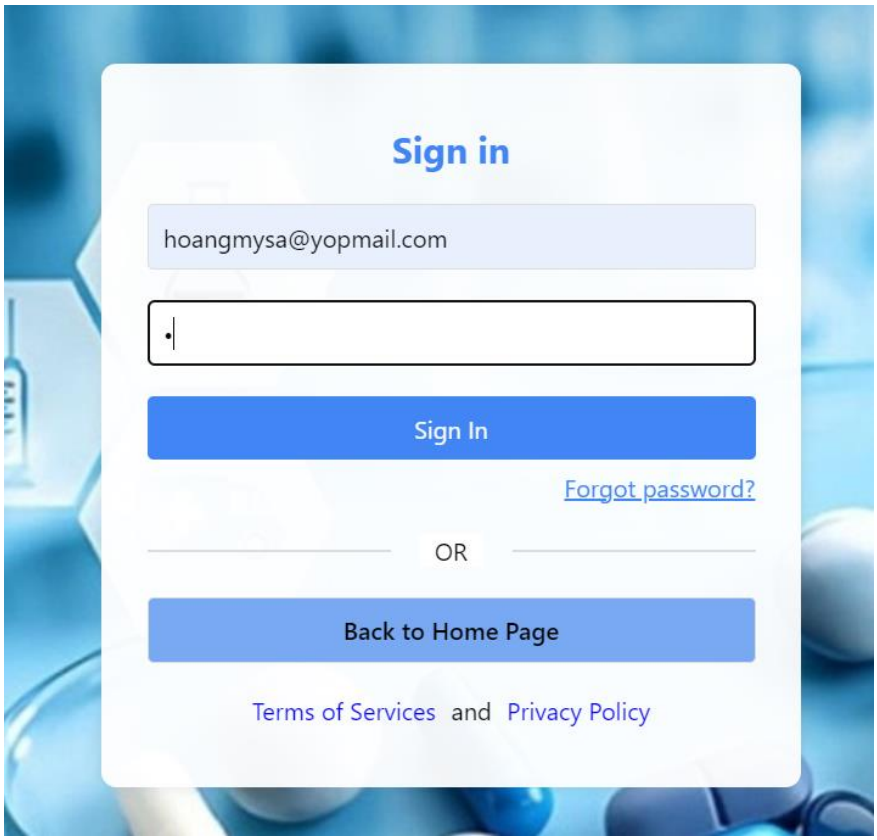


Hình 5.2: Giao diện nhận lời mời qua mail

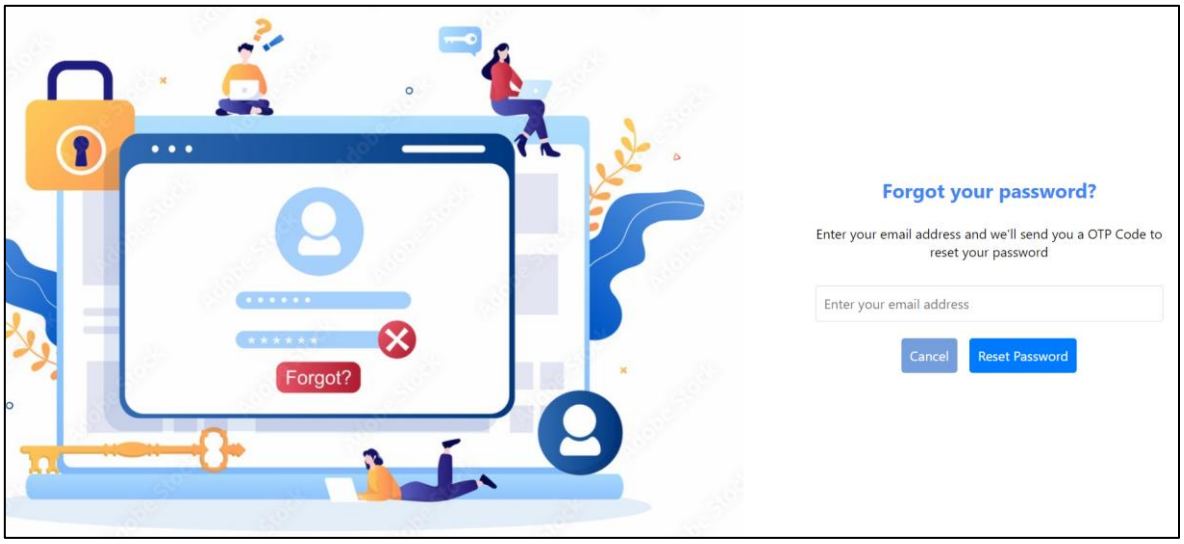


Hình 5.3: Giao diện thêm thông tin người dùng

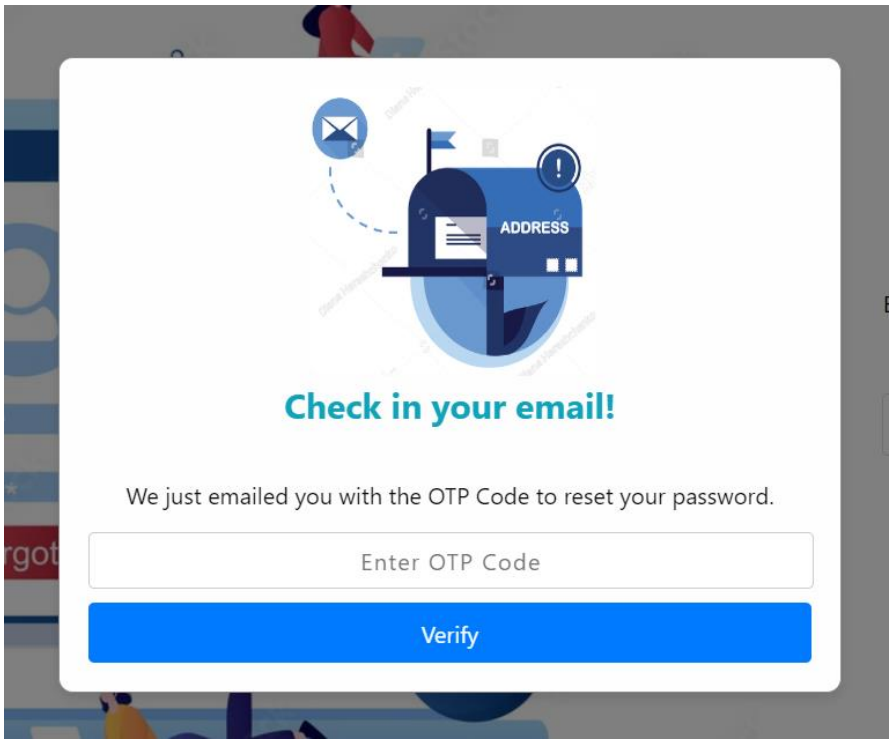
5.1.3 Giao diện quên mật khẩu



Hình 5.4: Giao diện quên mật khẩu 1

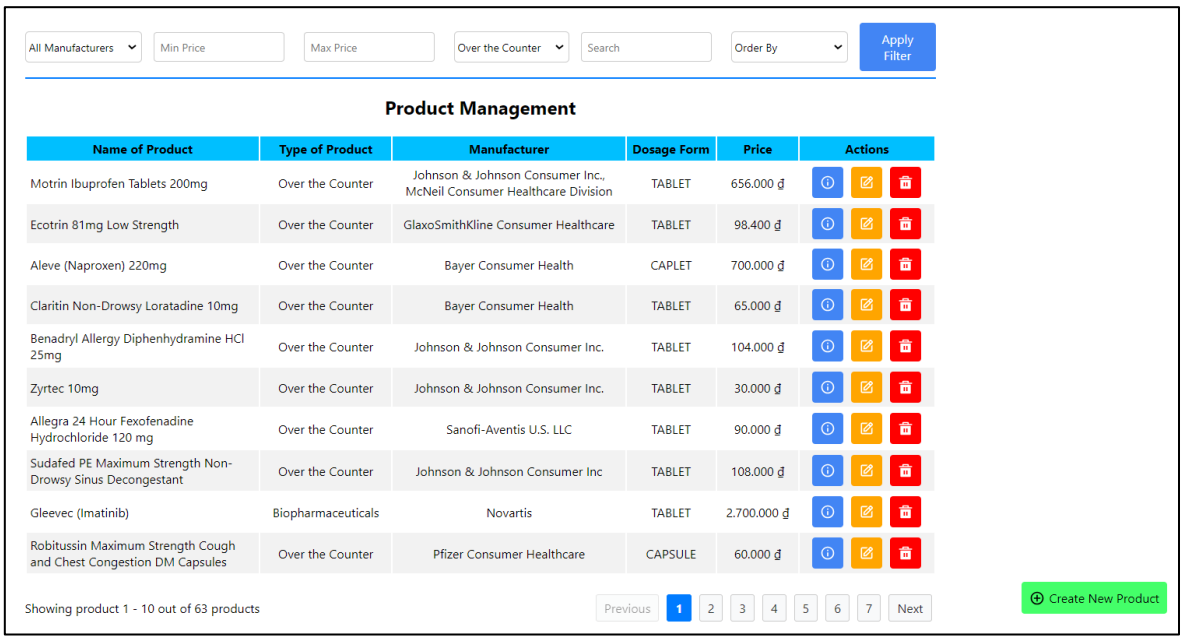


Hình 5.5: Giao diện quên mật khẩu 2



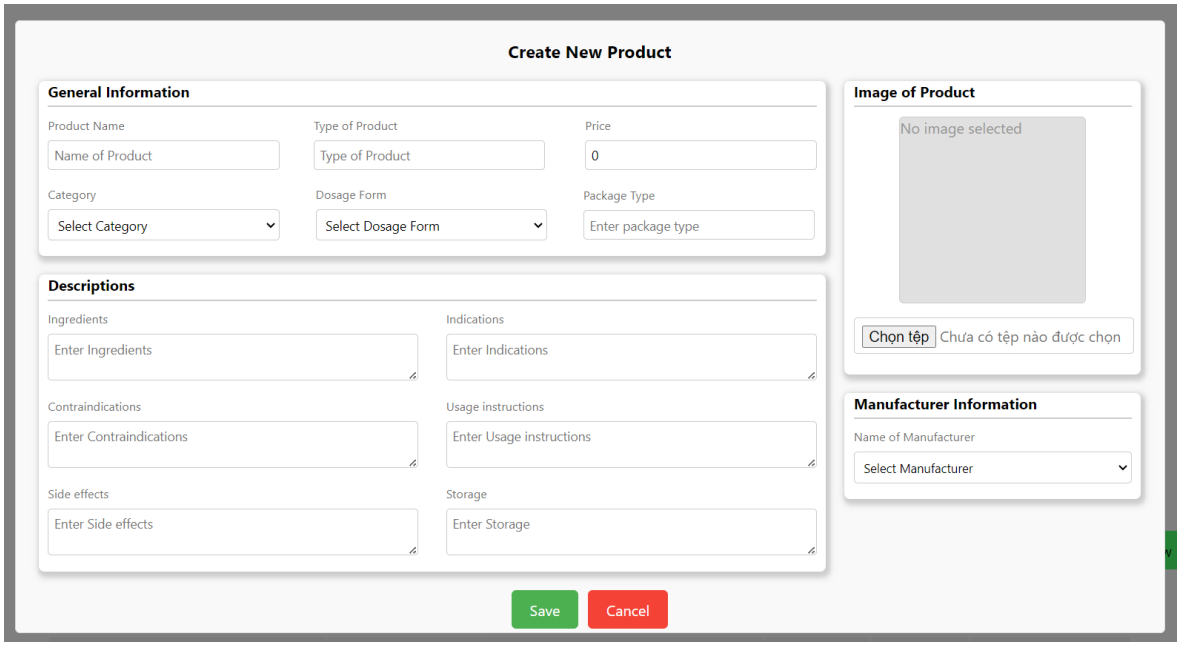
Hình 5.6: Giao diện quên mật khẩu 3

5.1.4 Giao diện tìm kiếm và xem danh sách sản phẩm



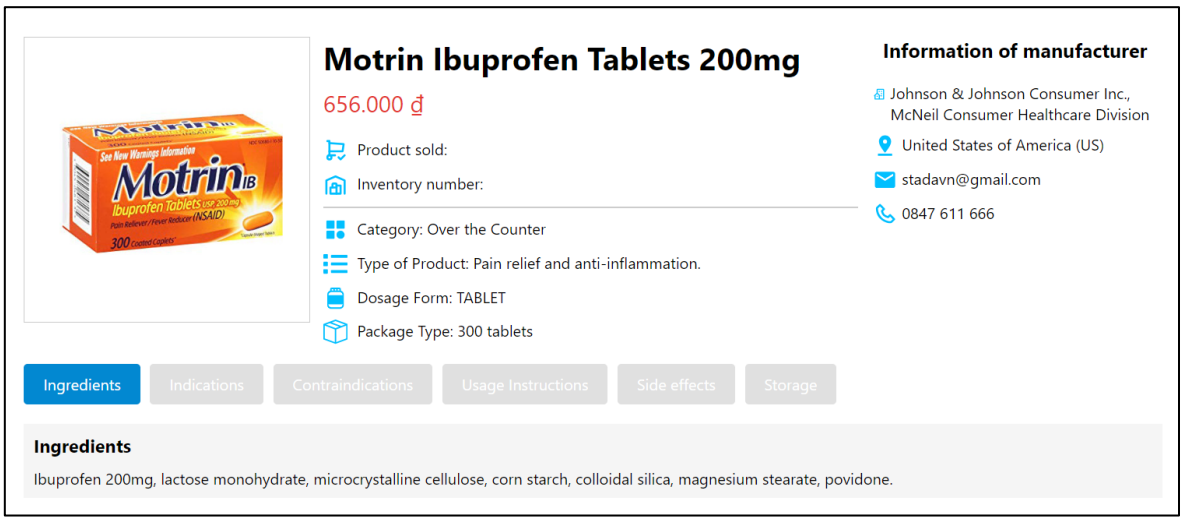
Hình 5.7: Giao diện tìm kiếm và xem danh sách sản phẩm

5.1.5 Giao diện thêm sản phẩm mới



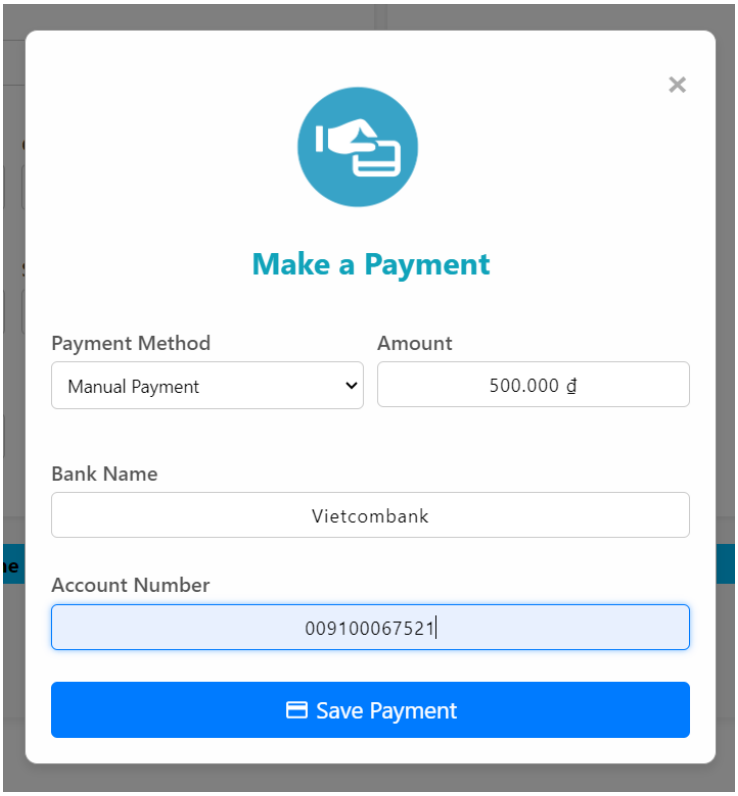
Hình 5.8: Giao diện thêm sản phẩm mới

5.1.6 Giao diện xem chi tiết sản phẩm

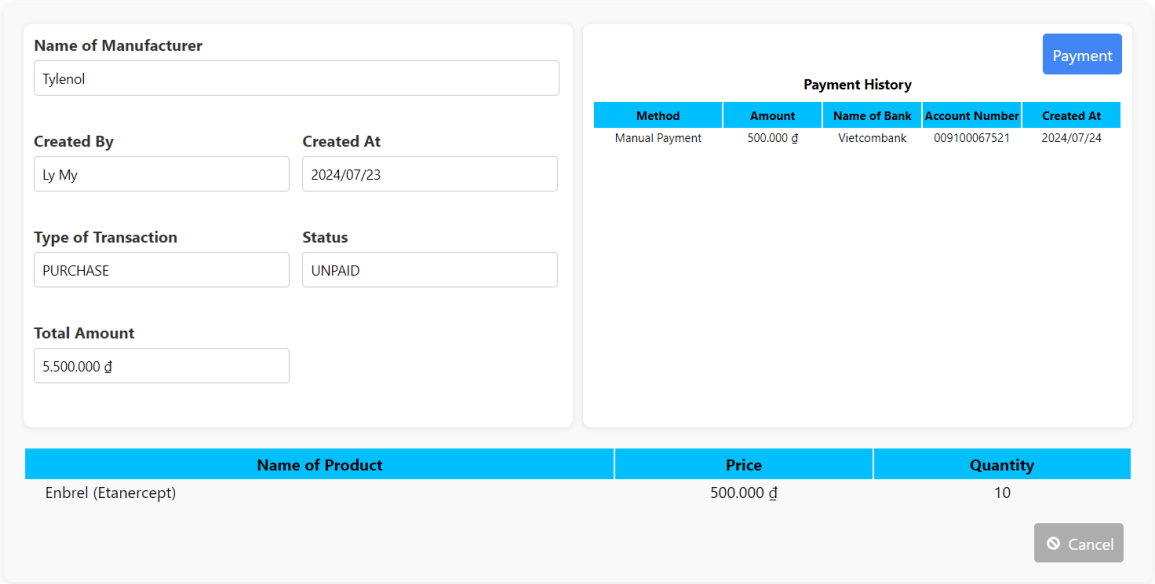


Hình 5.9: Giao diện xem chi tiết sản phẩm

5.1.7 Giao diện thanh toán đơn hàng



Hình 5.10: Giao diện thanh toán đơn hàng 1

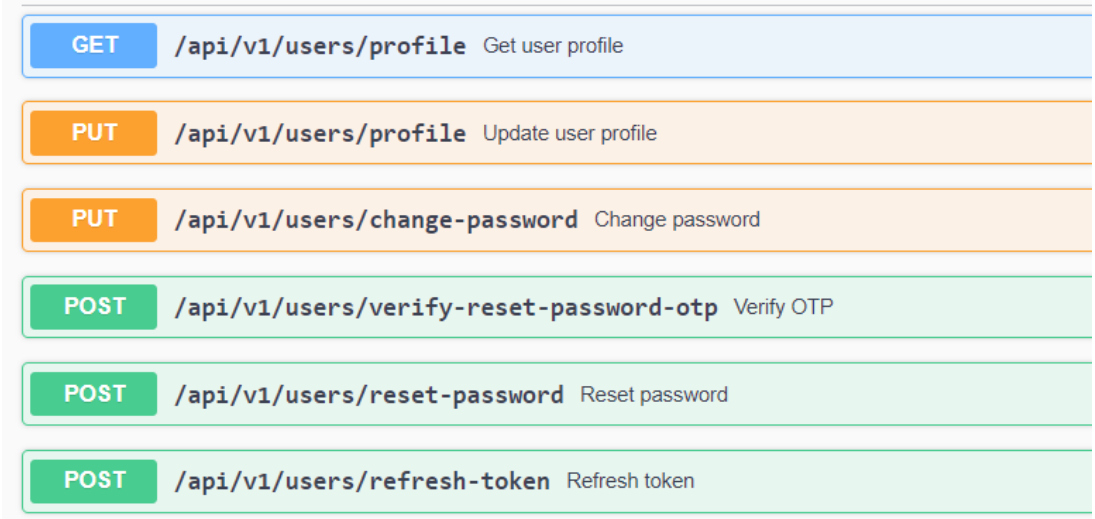


Hình 5.11: Giao diện thanh toán đơn hàng 2

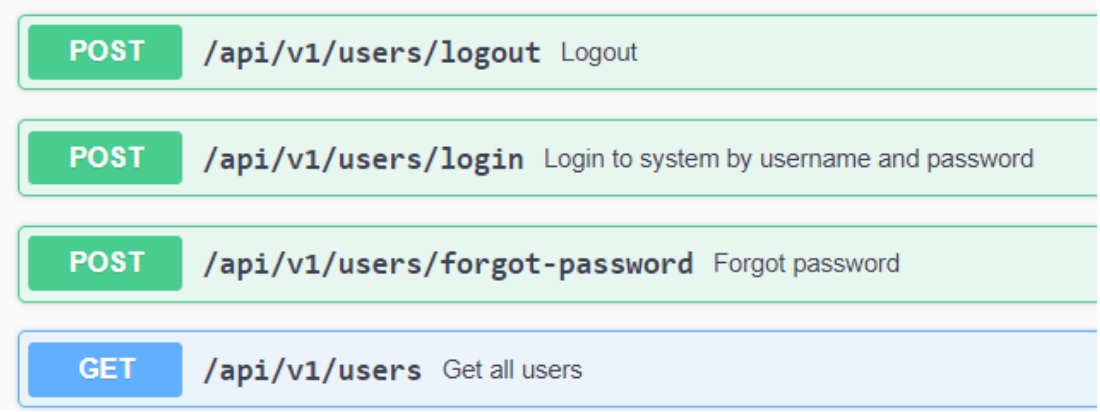
5.2 Phần hệ thống

5.2.1 User Service

Service này quản lý thông tin người dùng và phân quyền. Các chức năng chính bao gồm: Đăng nhập, quên mật khẩu, thay đổi mật khẩu, cập nhật thông tin cá nhân, mời người dùng, chấp nhận lời mời, xóa lời mời.



Hình 5.12: API của User Service (hình 1)



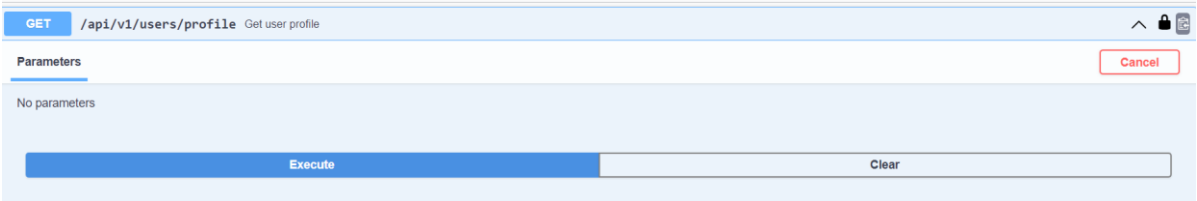
Hình 5.13: API của User Service (hình 2)

Hình 5.14: API của User Service (hình 3)

Ví dụ cụ thể về API get user profile:

Request:

- URI: /api/v1/users/profile
- Header: Cần phải chứa jwt trong header “Authorization”



Hình 5.15: API get user profile - Request

Response:

Hình 5.16: API get user profile - Response

Luồng thực hiện diễn ra như sau:

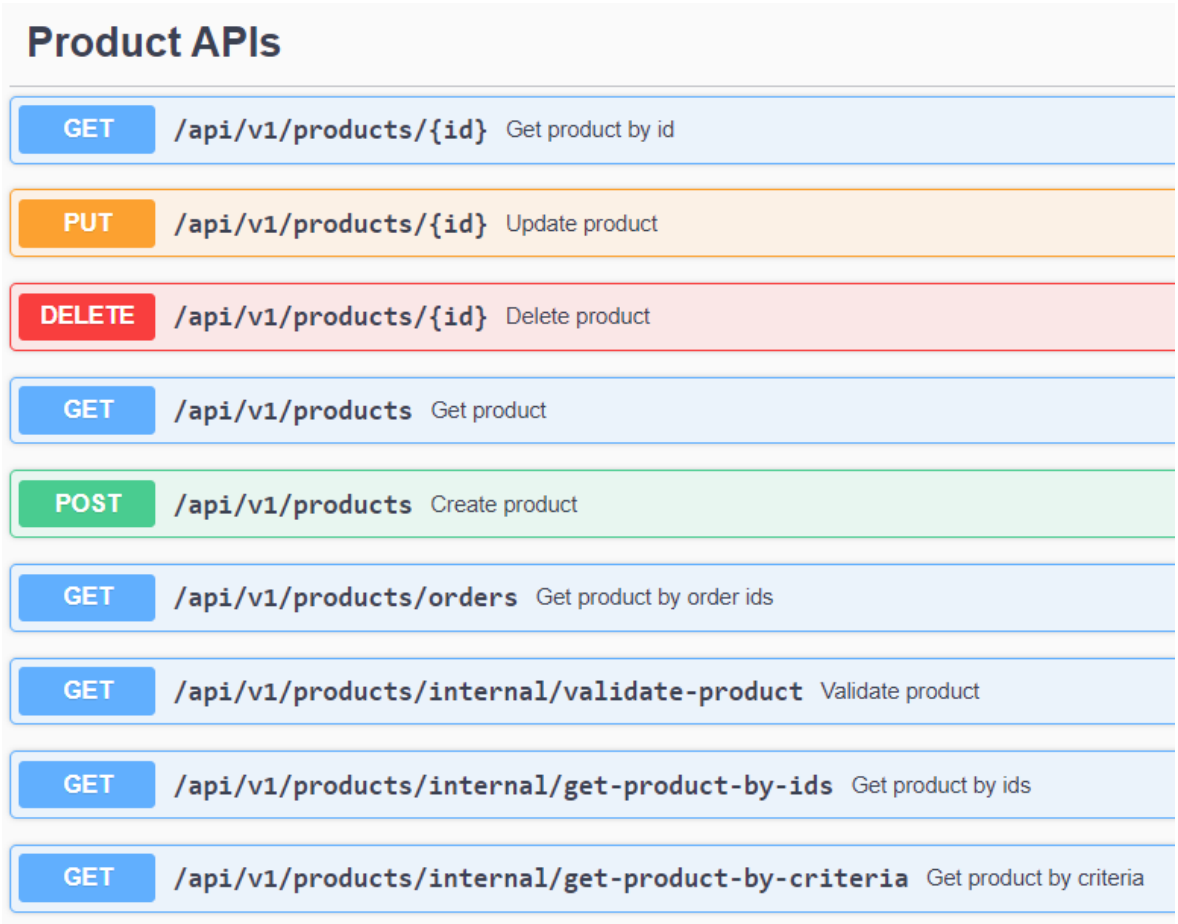
- User chọn xem thông tin người dùng tại FE.
- Request được gửi đến API Gateway, API Gateway sẽ redirect về user service.
- Tầng interceptor ở user service parse jwt từ header và set vào SecurityContextHolder.
- Request sẽ tiếp tục đến tầng controller, controller sẽ gọi đến tầng facade.
- Ở tầng facade, tiến hành lấy thông tin người dùng từ database và build response để trả về cho FE.
- FE nhận response và hiển thị thông tin người dùng lên UI.

5.2.2 Product Service

Service này sẽ quản lý thông tin về danh mục sản phẩm và sản phẩm.

Các chức năng chính như:

- Thêm mới sản phẩm
- Sửa thông tin sản phẩm
- Xem chi tiết sản phẩm
- Xóa sản phẩm (soft delete)

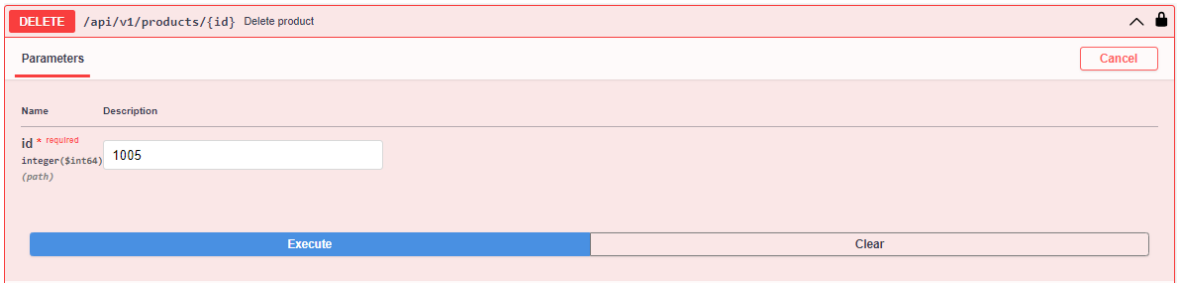


Hình 5.17: API của Product Service

Ví dụ về API Delete Product:

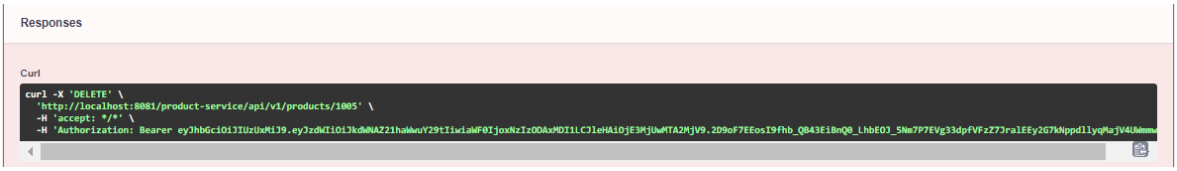
Request:

- URI: /api/v1/products/{id}
- Header: Cần phải chứa jwt trong header “Authorization”
- Path variable: {id} – product id



Hình 5.18: API delete product - Request

Response:



Hình 5.19: API delete product - Response

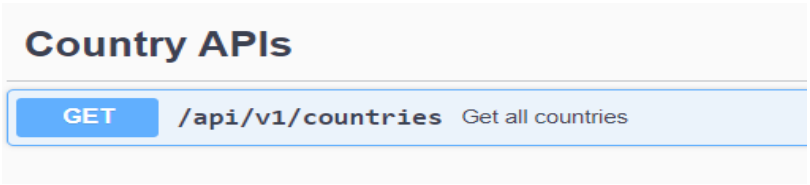
Luồng thực hiện diễn ra như sau:

- User click vào icon Delete tại danh sách sản phẩm hiển thị bởi FE.
- Request sẽ được gửi tới API gateway, sau đó được redirect tới product service.
- Tầng interceptor ở user service parse jwt từ header và set vào SecurityContextHolder.
- Request sẽ tiếp tục tới tầng Controller, tại đây phương thức deleteProduct trong ProductController nhận yêu cầu và gọi tới productFacade.deleteProduct(id) để thực hiện việc xóa sản phẩm.
- Tại tầng Façade sẽ điều phối yêu cầu tới Service để xử lý logic nghiệp vụ.
- Service tương tác với Respository để thực hiện soft delete (vẫn giữ thông tin tại database nhưng không hiển thị tại FE).
- Client nhận phản hồi thành công từ API.

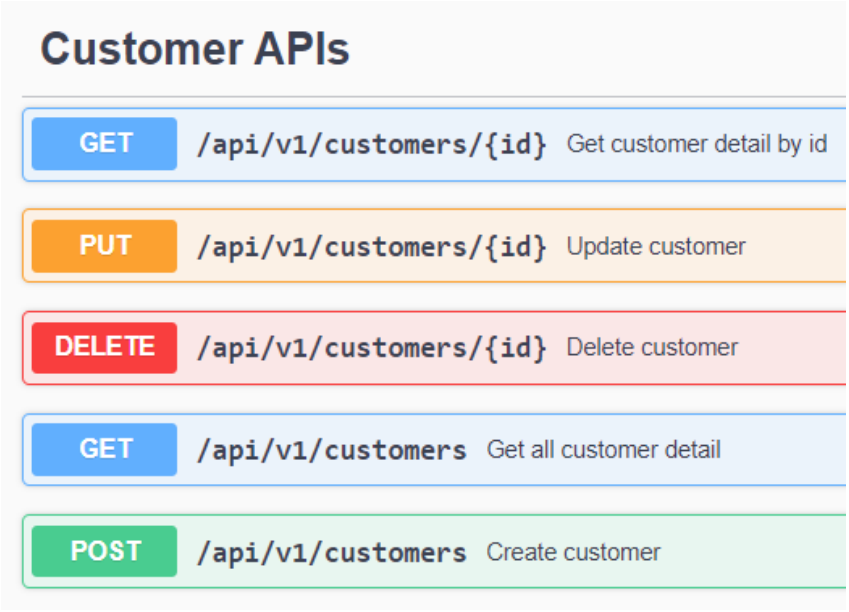
5.2.3 Customer Service

Service này sẽ quản lý thông tin khách hàng như:

- Thêm mới khách hàng
- Xem chi tiết khách hàng
- Sửa thông tin khách hàng
- Xóa thông tin khách hàng.



Hình 5.20: API của Customer Service (hình 1)

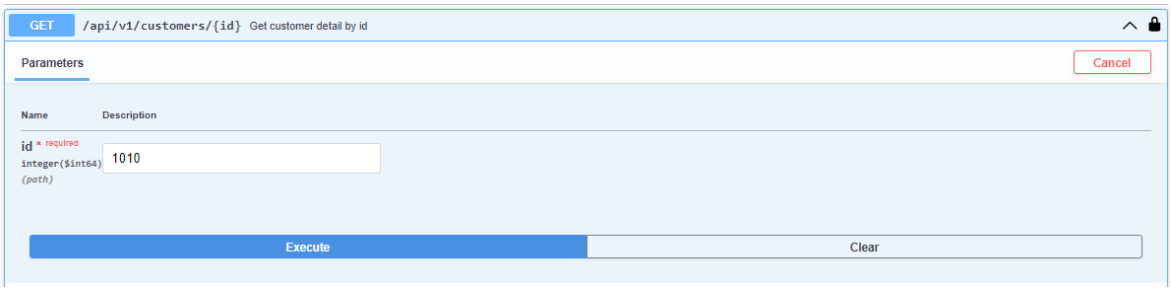


Hình 5.21: API của Customer Service (hình 2)

Ví dụ API Get customer detail by id:

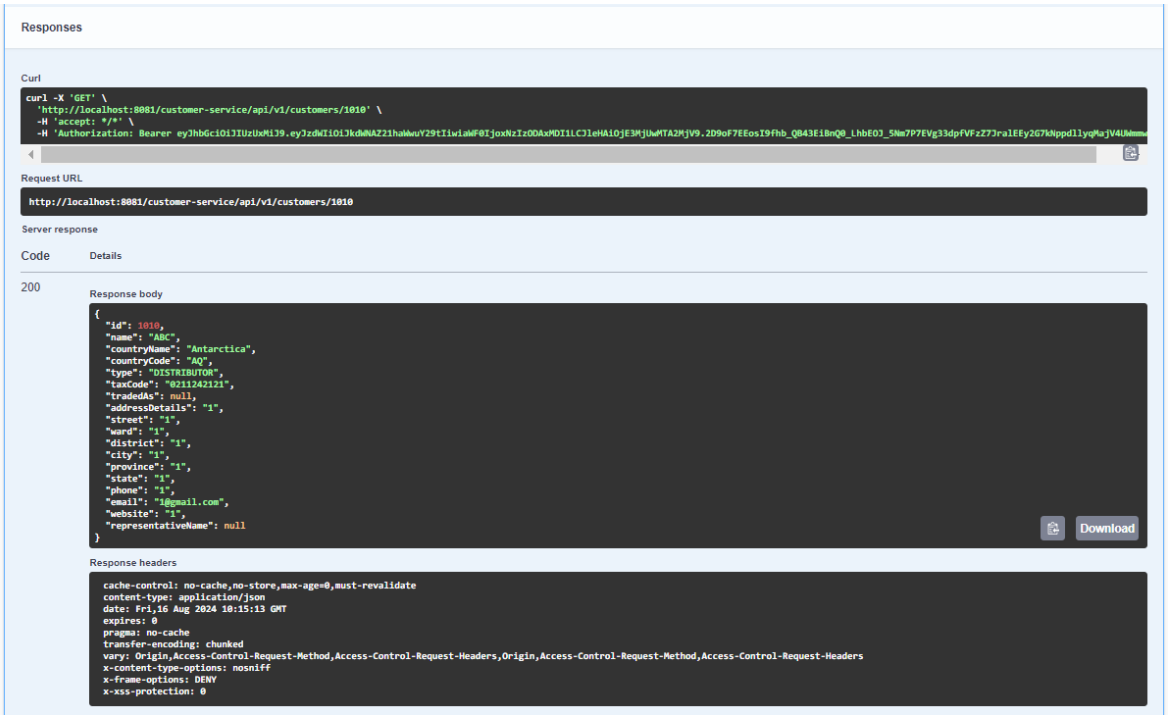
Request:

- URI: /api/v1/customers/{id}
- Header: Cần phải chứa jwt trong header "Authorization"
- Path variable: {id} – customer id



Hình 5.22: API get customer detail by id - Request

Response:



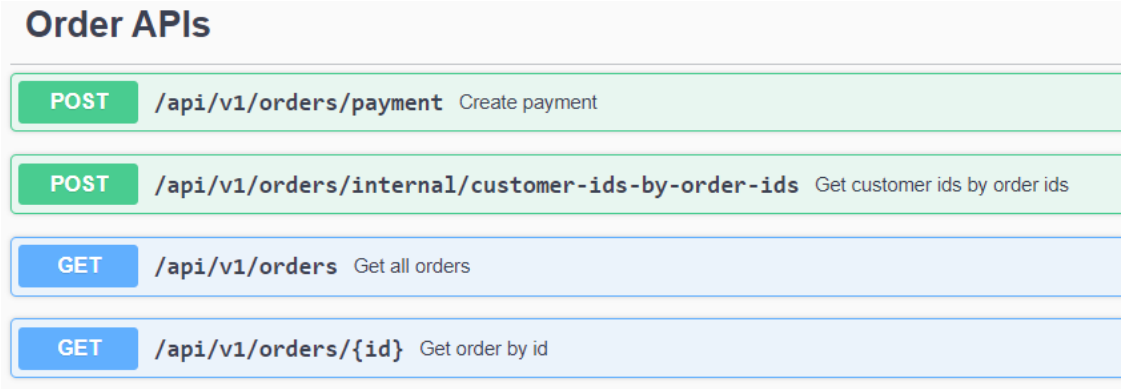
Hình 5.23: API get customer detail by id - Response

Luồng thực hiện diễn ra như sau:

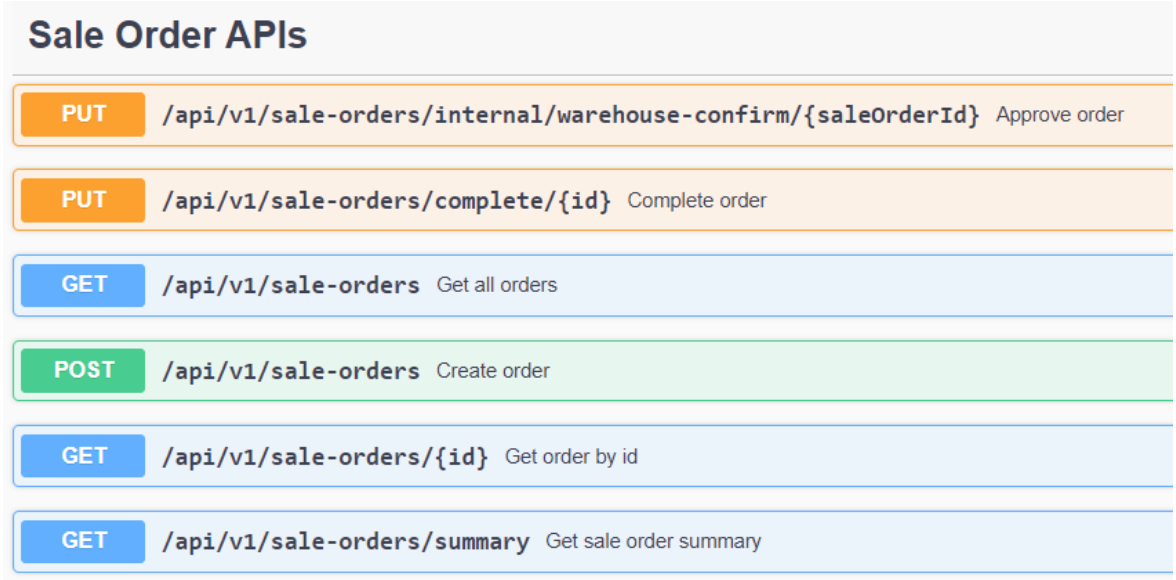
- User click vào icon View detail tại danh sách khách hàng tại FE.
- Request sẽ được gửi tới API gateway, sau đó được redirect tới customer service.
- Tầng interceptor ở user service parse jwt từ header và set vào SecurityContextHolder.
- Request sẽ chạy tới tầng Controller, tại đây sẽ nhận yêu cầu và gọi tới Service.
- Service sẽ tương tác với Repository để tìm kiếm khách hàng trong database. Service chuyển đổi thực thể customer thành response và trả về Controller.
- Client nhận phản hồi thành công từ API với thông tin chi tiết khách hàng.

5.2.4 Order Service

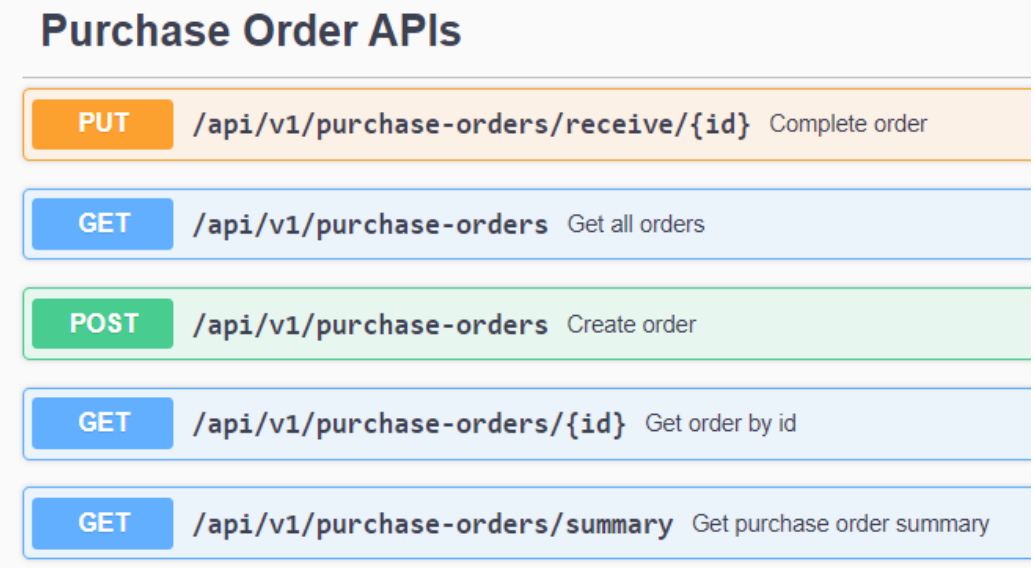
Service này sẽ quản lý tất cả những đơn hàng nhập về, bán ra và thanh toán, cụ thể bao gồm: Tạo yêu cầu đơn hàng nhập, tạo yêu cầu bán hàng, tạo hóa đơn, thanh toán đơn hàng.



Hình 5.24: API của Order Service (hình 1)



Hình 5.25: API của Order Service (hình 2)

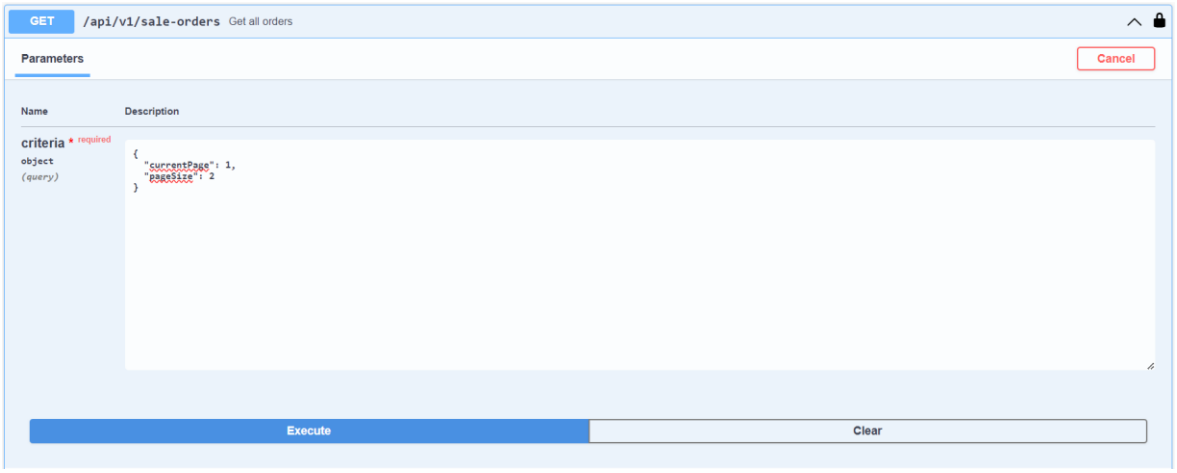


Hình 5.26: API của Order Service (hình 3)

Ví dụ về API get all orders:

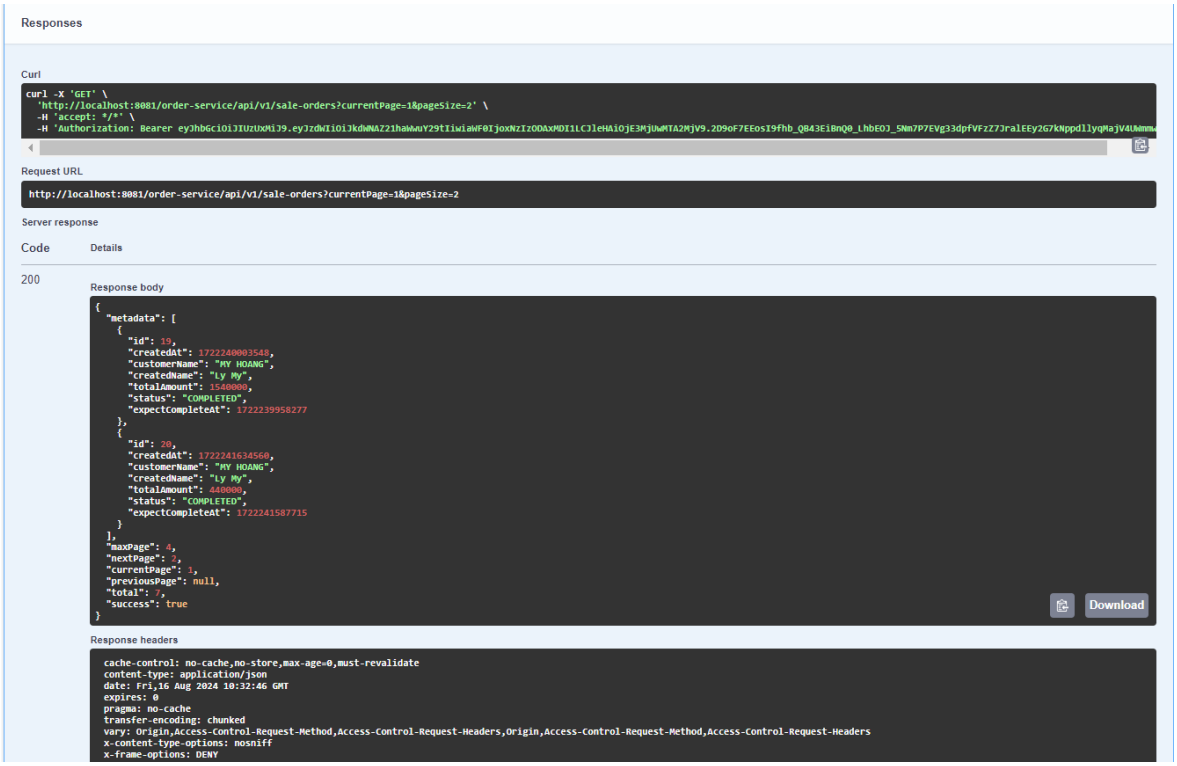
Request:

- URI: /api/v1/sale-orders
- Header: Cần phải chứa jwt trong header “Authorization”
- Request param: currentPage, pageSize



Hình 5.27: API get all orders - Request

Respone:



Hình 5.28: API get all orders - Response

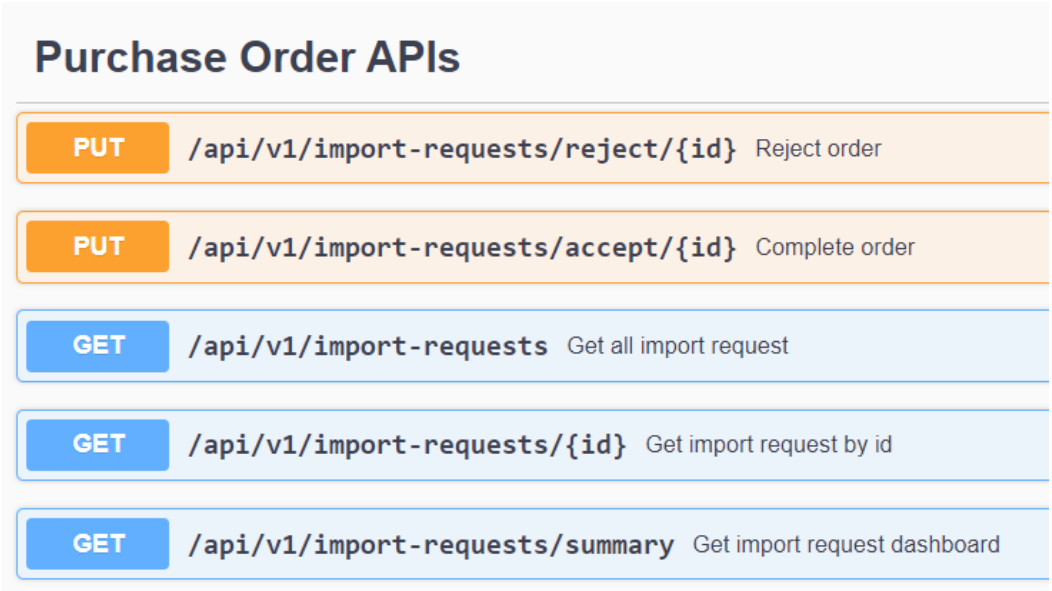
Luồng thực hiện diễn ra như sau:

- User click vào submenu List Invoices trong menu Order tại FE.
- Request sẽ được gửi tới API gateway, sau đó được redirect tới order service.
- Tầng interceptor ở order service parse jwt từ header và set vào SecurityContextHolder.
- Request sẽ chạy tới tầng Controller, tại đây sẽ nhận yêu cầu và gọi tới Service.
- Service tương tác với Respository để lấy tất cả đơn hàng từ database.
- Service chuyển đổi danh sách order thành danh sách orderResponse và trả về cho Controller.
- Client nhận phản hồi thành công từ API với danh sách các đơn hàng.

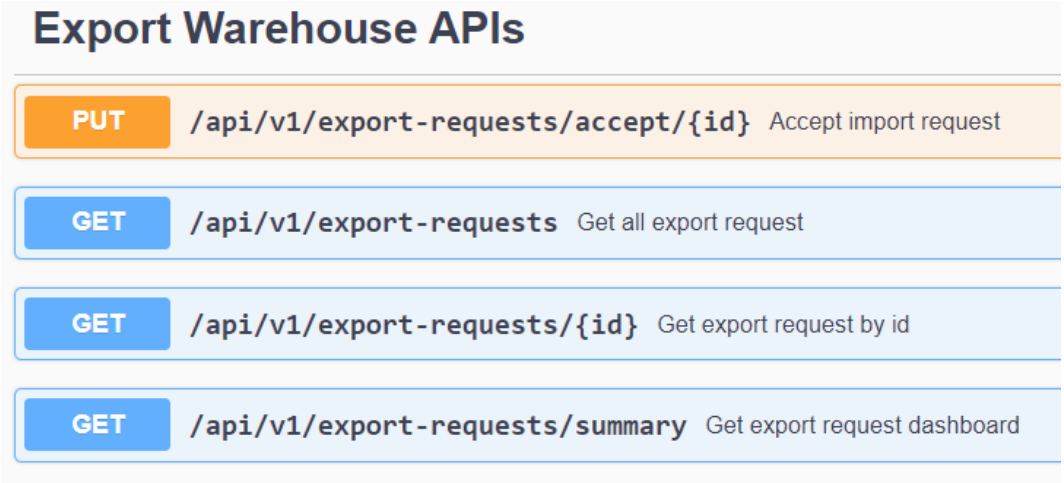
5.2.5 Warehouse Service

Service này sẽ quản lý quy trình nhập hàng, xuất hàng, số lượng hàng còn lại trong kho và thông tin nhập, xuất hàng.

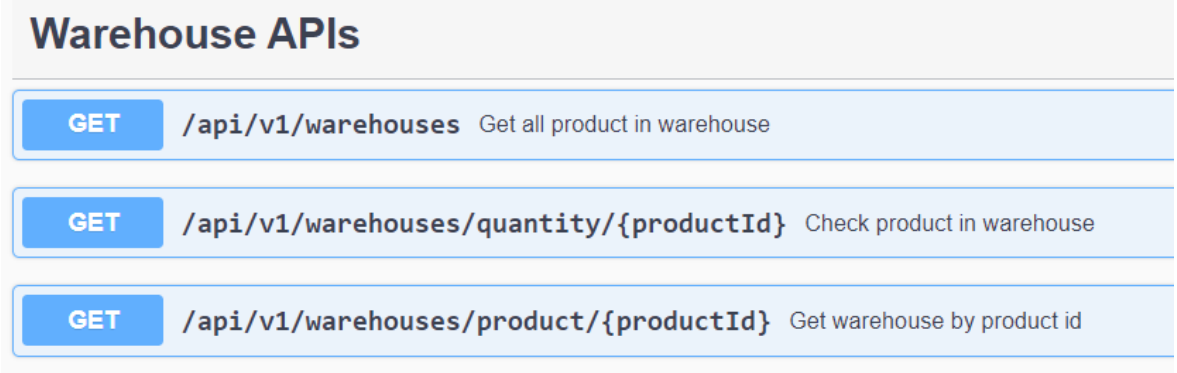
Các chức năng chính bao gồm: Xem danh sách yêu cầu nhập hàng và xuất hàng, nhận yêu cầu từ phía order, thông báo giao hàng thành công, xem danh sách sản phẩm trong kho, kiểm tra số lượng đơn hàng còn trong kho.



Hình 5.29: API của Warehouse Service (hình 1)



Hình 5.30: API của Warehouse Service (hình 2)



Hình 5.31: API của Warehouse Service (hình 3)

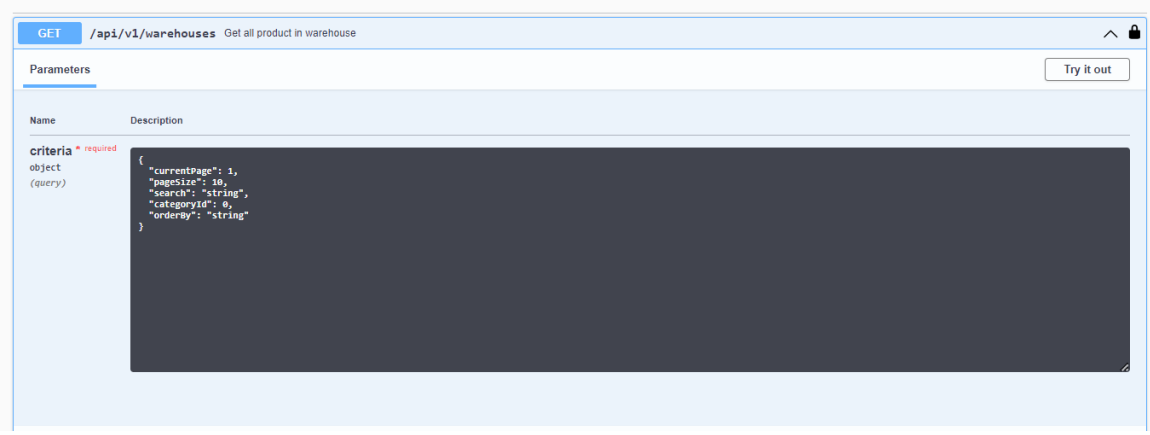
Ví dụ về API get all product in warehouse:

Luồng thực hiện diễn ra như sau:

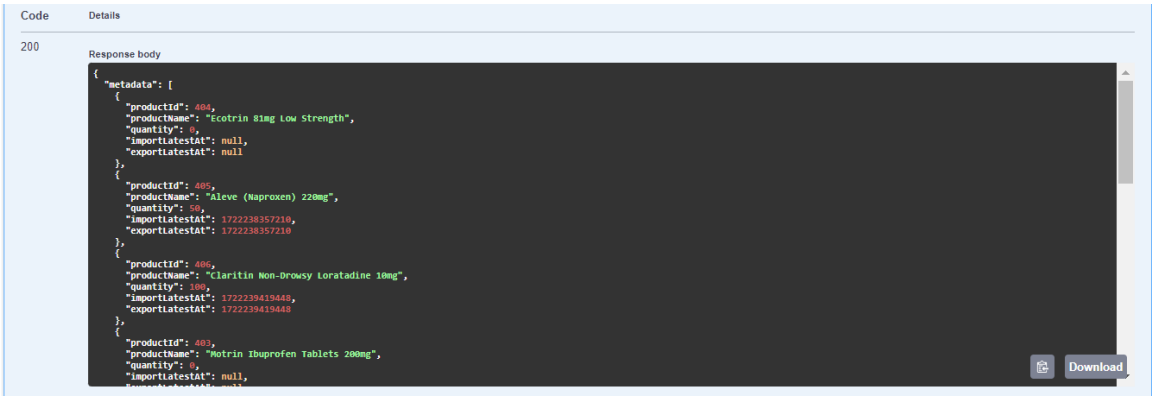
- User click vào submenu List of Product trong menu Warehouse tại FE.
- Request sẽ được gửi tới API gateway, sau đó được redirect tới warehouse service.
- Tầng interceptor ở user service parse jwt từ header và set vào SecurityContextHolder.
- Request sẽ chạy tới tầng Controller, tại đây sẽ nhận yêu cầu và gọi tới Service.
- Tại tầng Service sẽ gọi tới tầng Repository và tại đây sẽ tương tác trực tiếp với database để get all thông tin các product có trong warehouse.
- Map response từ list entities lấy được từ tầng service và trả kết quả cho FE thông qua API.

Request:

- URI: /api/v1/warehouses
- Header: Cần phải chứa jwt trong header “Authorization”
- Request param: currentPage, pageSize, search, category, orderBy



Hình 5.32: API get all product in warehouse - Request



Hình 5.33: API get all product in warehouse - Response

CHƯƠNG 6. TRIỂN KHAI HỆ THỐNG

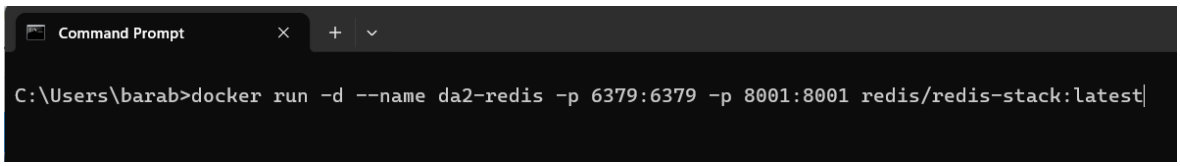
6.1 Eureka

Instances currently registered with Eureka

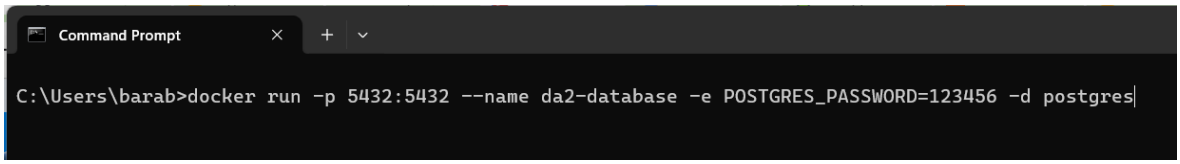
Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - DESKTOP-FVPLPUD:api-gateway:8081
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-FVPLPUD:customer-service:8200
MAIL-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-FVPLPUD:mail-service:587
ORDER-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-FVPLPUD:order-service:8300
PRODUCT-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-FVPLPUD:product-service:8100
USER-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-FVPLPUD:user-service:8000
WAREHOUSE-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-FVPLPUD:warehouse-service:8400

Hình 6.1: Những Service được đăng ký bằng Eureka

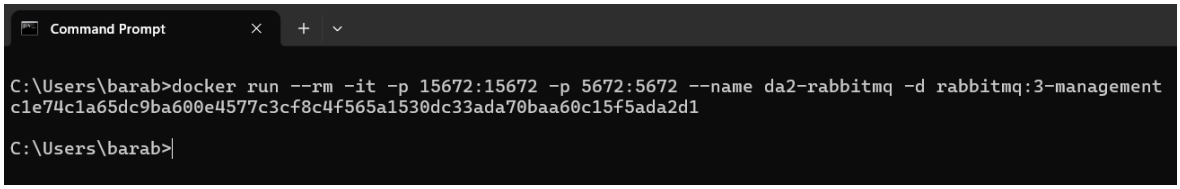
6.2 Triển khai trên Docker



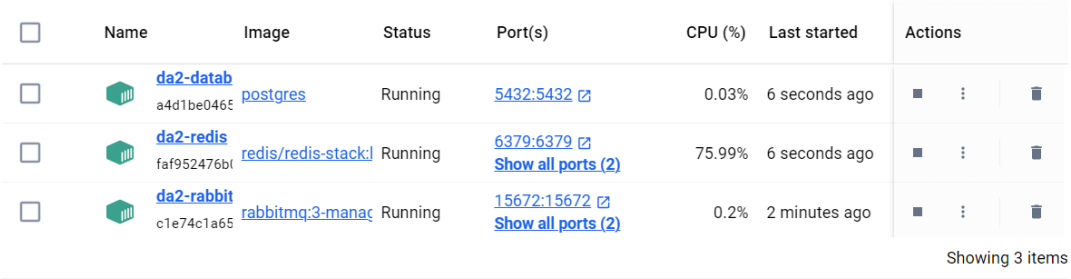
Hình 6.2: Triển khai Redis trên Docker



Hình 6.3: Triển khai PostgresSQL trên Docker



Hình 6.4: Triển khai RabbitMQ trên Docker



Hình 6.5: Màn hình sau khi đã triển khai trên Docker

6.3 Ngrok

```
ngrok

Found a bug? Let us know: https://github.com/ngrok/ngrok

Session Status                online
Account                       vduc301 (Plan: Free)
Update                        update available (version 3.13.0, Ctrl-U to update)
Version                       3.11.0
Region                        Asia Pacific (ap)
Latency                       26ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://7d19-171-224-241-12.ngrok-free.app -> http://localhost:8081

Connections                   ttl     opn     rt1     rt5     p50     p90
                              286     0       0.00    0.00    21.28   108.76

HTTP Requests
-------------

16:28:30.127 +07 POST    /user-service/api/v1/users/logout 401 Unauthorized
16:28:28.465 +07 OPTIONS /user-service/api/v1/users/logout 200 OK
16:28:28.537 +07 POST    /user-service/api/v1/users/logout 401 Unauthorized
16:05:27.865 +07 GET     /v3/api-docs/warehouse-service    200 OK
16:03:25.203 +07 GET     /v3/api-docs/order-service        200 OK
15:59:33.981 +07 GET     /v3/api-docs/customer-service     200 OK
15:53:53.332 +07 GET     /v3/api-docs/product-service      200 OK
15:48:10.245 +07 GET     /v3/api-docs/user-service         200 OK
15:45:58.062 +07 GET     /v3/api-docs/order-service        200 OK
15:45:50.668 +07 GET     /v3/api-docs/warehouse-service    200 OK
```

Hình 6.6: Chạy Ngrok để kết nối API với front-end

6.4 Kết nối front-end vs API

```
.env 1M, M
.env
1 REACT_APP_API_BASE_URL=https://7d19-171-224-241-12.ngrok-free.app
2
```

Hình 6.7: Config domain của back-end ở front-end

CHƯƠNG 7. KẾT LUẬN

7.1 Kết luận

7.1.1 Những vấn đề đạt được

Sau quá trình nghiên cứu, tìm hiểu và xây dựng Phần mềm Quản lý phân phối Dược phẩm với kiến trúc Microservices trên nền tảng Java Spring Boot, nhóm đã đạt được những thành tựu sau đây:

Ứng dụng được các công cụ, công nghệ, framework và kiến trúc như Microservices, Eureka, PostgreSQL, Java Spring Boot, Spring Cloud Gateway, mail SMTP, Redis, RabbitMQ, Swagger, ReactJS,...vào hệ thống.

Thực hiện được đầy đủ các chức năng cơ bản để quản lý quy trình phân phối dược phẩm như: quản lý danh mục sản phẩm, quản lý sản phẩm, quản lý khách hàng, quản lý đơn hàng, quản lý kho và thanh toán đơn hàng.

7.1.2 Những vấn đề chưa đạt được

Bên cạnh những điều đã đạt được thì nhóm có một số vấn đề chưa đạt được như sau:

- Chưa hoàn thiện những chức năng nâng cao như thanh toán qua mã QR, thanh toán qua ngân hàng, quản lý doanh thu, quản lý hợp đồng và tài liệu.
- Chưa xây dựng được kênh kết nối với nhà sản xuất dược phẩm và nhà thuốc.
- Chưa xây dựng được phần mềm dành cho khách hàng trên ứng dụng di động.
- Chưa ứng dụng được machine learning và AI vào hệ thống.

7.1.3 Thuận lợi và khó khăn

Những thuận lợi trong quá trình thực hiện dự án:

- Các công nghệ, framework và kiến trúc như ReactJS, Java Spring Boot, RabbitMQ, PostgreSQL và Microservices khá phổ biến, có nguồn tài

liệu phong phú cũng như cộng đồng hỗ trợ lớn mạnh là điểm thuận lợi lớn cho việc tìm hiểu các vấn đề kỹ thuật và áp dụng vào dự án.

- Các công nghệ như ReactJS, Java Spring Boot mang đến cho người dùng trải nghiệm tốt, mượt mà và hiệu suất xử lý cao.

Bên cạnh những điểm thuận lợi, nhóm cũng gặp những khó khăn trong khi thực hiện dự án:

- Hệ thống Microservices và một số công nghệ khác đòi hỏi người sử dụng phải có sự hiểu biết và kinh nghiệm để có thể tích hợp chúng vào hệ thống và phối hợp với nhau.
- Việc sử dụng Microservices tuy là một phương án tốt trong xây dựng hệ thống phân tán nhưng đồng thời nó cũng phát sinh ra vấn đề về bài toán đồng bộ dữ liệu giữa các service với nhau.
- Kích thước của dự án khá lớn và gây áp lực cho những thiết bị có cấu hình yếu.

7.2 Hướng phát triển trong tương lai

Dựa trên kết quả hiện tại của dự án, nhóm em đã đưa ra một số ý tưởng để phát triển phần mềm trong tương lai:

- Xây dựng kênh kết nối với nhà sản xuất và nhà thuốc.
- Xây dựng các service quản lý doanh thu, quản lý hợp đồng và tài liệu.
- Tích hợp cổng thanh toán qua tài khoản ngân hàng, MOMO và các dịch vụ thanh toán điện tử khác.
- Xây dựng phần mềm dành cho nhà thuốc trên ứng dụng di động.
- Tích hợp machine learning và AI vào hệ thống nhằm phân tích dữ liệu sản phẩm, thông báo sản phẩm sắp hết hạn hoặc số lượng sản phẩm cần nhập kho. Lập danh sách các sản phẩm cần nhập tùy theo nhu cầu (ví dụ: dịch Covid-19, dịch cúm,...).

TÀI LIỆU THAM KHẢO

Tiếng Anh

Sam Newman. (2015). Building Microservices.

React - A JavaScript library for building user interfaces. Retrieved 10 May 2024, from https://legacy.reactjs.org/

Spring Boot. Retrieved 10 May 2024, from https://spring.io/projects/spring-boot

Spring | Microservices. Retrieved 10 May 2024, from https://spring.io/microservices

Messaging with RabbitMQ. Retrieved 10 May 2024, from https://spring.io/guides/gs/messaging-rabbitmq