**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

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

**ĐẶNG XUÂN PHÁT**

**PHÁT TRIỂN HỆ THỐNG CHUỖI CUNG ỨNG SỬ DỤNG SPRING BOOT VÀ NEXTJS**

**KHÓA LUẬN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

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

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

**ĐẶNG XUÂN PHÁT**

**PHÁT TRIỂN HỆ THỐNG CHUỖI CUNG ỨNG SỬ DỤNG SPRING BOOT VÀ NEXTJS**

**Mã số sinh viên: 2151050316**

**KHÓA LUẬN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Giảng viên hướng dẫn: TS.TRƯƠNG HOÀNG VINH**

**TP. HỒ CHÍ MINH, 2025**

|  |  |  |
| --- | --- | --- |
| TRƯỜNG ĐẠI HỌC MỞ  THÀNH PHỐ HỒ CHÍ MINH | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc** | |
| **KHOA CÔNG NGHỆ THÔNG TIN**  **\_\_\_\_\_\_** | |  |

**GIẤY XÁC NHẬN**

Tôi tên là: Đặng Xuân Phát

Ngày sinh: 11/11/2003 Nơi sinh: Kiên Giang

Chuyên ngành: Công Nghệ Thông Tin Mã sinh viên: 2151050316

Tôi đồng ý cung cấp toàn văn thông tin đồ án/ khóa luận tốt nghiệp hợp lệ về bản quyền cho Thư viện Trường Đại học Mở Thành phố Hồ Chí Minh. Thư viện Trường Đại học Mở Thành phố Hồ Chí Minh sẽ kết nối toàn văn thông tin đồ án/ khóa luận tốt nghiệp vào hệ thống thông tin khoa học của Sở Khoa học và Công nghệ Thành phố Hồ Chí Minh.

Ký tên

*(Ghi rõ họ và tên)*

…………………………

Ý KIẾN CHO PHÉP BẢO VỆ ĐỒ ÁN/ KHÓA LUẬN TỐT NGHIỆP

CỦA GIẢNG VIÊN HƯỚNG DẪN

**Giảng viên hướng dẫn:**

**Sinh viên thực hiện: Lớp:**

**Ngày sinh: Nơi sinh:**

**Tên đề tài:**

**Ý kiến của giảng viên hướng dẫn về việc cho phép sinh viên được bảo vệ đồ án/ khóa luận trước Hội đồng:**

*Thành phố Hồ Chí Minh, ngày … tháng … năm ……*

Người nhận xét

LỜI CẢM ƠN

Trong khoảng thời thực hiện đồ án tốt nghiệp cũng như là giai đoạn gần cuối trong chặng đường học tập tại Trường Đại Học Mở Thành Phố Hồ Chí Minh, em đã nhận được rất nhiều sự hỗ trợ tận tình từ thầy cô, bạn bè và gia đình.

Trước tiên, em xin gửi lời tri ân sâu sắc đến giảng viên hướng dẫn của em, thầy Trương Hoàng Vinh – Tiến sĩ và giảng viên khoa Công nghệ Thông tin trường Đại học Mở Thành phố Hồ Chí Minh. Em chân thành cảm ơn thầy đã đồng hành và hướng dẫn cho em có thể hoàn thành được tốt trong dự án đồ án này cũng như là giúp em có thể cải thiện được bản thân hơn và học hỏi được nhiều điều mới.

Ngoài ra, em cũng xin bày tỏ lòng biết ơn đến tất cả các giảng viên khoa Công nghệ Thông tin trường Đại học Mở Thành phố Hồ Chí Minh. Nhờ sự tận tâm của các thầy cô đã truyền tải kiến thức cho em mà em có được một nền tảng kiến thức vững chắc và kinh nghiệm quý báu để có thể áp dụng mà hoàn thành được môn đồ án này một cách trọn vẹn.

Thêm vào đó, em xin gửi lời cảm ơn chân thành đến gia đình và bạn bè, những người đã luôn đồng hành, động viên và hỗ trợ em trong suốt quá trình học tập cũng như giai đoạn thực hiện đồ án. Sự quan tâm và ủng hộ từ mọi người đã giúp em vượt qua những thử thách và khó khăn trong chặng đường vừa qua.

Em cũng nhận thấy bản thân còn rất nhiều thiếu sót về kinh nghiệm, rất mong nhận được đóng góp ý kiến từ thầy cô để em có thể cải thiện hơn cho bản thân.

Một lần nữa, em xin chân thành cảm ơn.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TÓM TẮT KHÓA LUẬN

Khóa luận của tôi tập trung vào phát triển một hệ thống quản lý chuỗi cung ứng hiện đại, với mục tiêu tối ưu hóa quy trình quản lý kho, theo dõi đơn hàng và sản xuất sản phẩm hiệu quả hơn cho các doanh nghiệp. Hệ thống được xây dựng trên nền tảng **Spring Boot** cho backend và **NextJS** cho frontend, đảm bảo tính hiệu suất cao và khả năng mở rộng linh hoạt. Đặc biệt, để cải thiện tốc độ xử lý, tôi tích hợp **Redis** như một giải pháp lưu trữ tạm thời (cache), giúp tối ưu hóa quá trình truy xuất dữ liệu.

Hệ thống này cung cấp một giải pháp toàn diện với các chức năng chính như quản lý nhà cung cấp, vật liệu, đơn hàng, kho, và báo cáo doanh thu. Các tính năng này không chỉ giúp doanh nghiệp tăng cường khả năng theo dõi nguồn lực, giảm thiểu rủi ro trong chuỗi cung ứng mà còn nâng cao khả năng ra quyết định thông qua các báo cáo chi tiết.

Để cải thiện hiệu quả công việc, tôi đã tích hợp thêm **trí tuệ nhân tạo (AI)** vào hệ thống. AI giúp dự báo doanh thu trong tương lai dựa trên dữ liệu lịch sử, giúp doanh nghiệp đưa ra các quyết định kinh doanh chính xác hơn. Đồng thời, tôi phát triển một **chatbot AI** có khả năng trả lời các câu hỏi cơ bản của khách hàng, giúp cải thiện trải nghiệm người dùng và tiết kiệm thời gian cho đội ngũ hỗ trợ khách hàng.

Về mặt kỹ thuật, tôi sử dụng **Spring Boot** để phát triển các API RESTful cho việc xử lý tác vụ backend, kết hợp với **Thymeleaf** cho giao diện quản trị phía máy chủ, giúp nội dung được xử lý trực tiếp từ máy chủ và cập nhật dữ liệu nhanh chóng. **NextJS** được sử dụng cho giao diện người dùng, mang lại trải nghiệm hiện đại, tốc độ tải trang nhanh và tối ưu hóa SEO. Sự kết hợp giữa **client-side rendering** và **server-side rendering** giúp cải thiện tốc độ và hiệu suất tổng thể của hệ thống.

Với việc tích hợp Redis, khả năng xử lý dữ liệu của hệ thống được nâng cao, giúp giảm thiểu thời gian truy xuất dữ liệu và hỗ trợ các hoạt động kinh doanh diễn ra mượt mà. Kết quả của khóa luận là một hệ thống quản lý chuỗi cung ứng linh hoạt, mạnh mẽ và có khả năng đáp ứng nhu cầu quản lý trong các doanh nghiệp hiện đại. Hệ thống này không chỉ giúp tối ưu hóa quy trình vận hành và chi phí mà còn mang lại lợi thế cạnh tranh lớn cho doanh nghiệp trong bối cảnh thị trường ngày càng khốc liệt.

ABSTRACT

My thesis focuses on the development of a modern supply chain management system aimed at optimizing warehouse management, order tracking, and product production processes to enhance business efficiency. The system is built on the **Spring Boot** framework for the backend and **NextJS** for the frontend, ensuring high performance and scalability. To improve speed, I integrated **Redis** as a caching solution, optimizing data retrieval time.

The system offers a comprehensive solution with essential features such as supplier management, material tracking, order management, inventory control, and revenue reporting. These features help businesses improve resource tracking, reduce supply chain risks, and enhance decision-making through detailed reports.

To further improve operational efficiency, I integrated **Artificial Intelligence (AI)** into the system. AI is used to forecast future revenue based on historical data, enabling businesses to make more accurate decisions. Additionally, I developed an **AI-powered chatbot** capable of answering basic customer queries, improving the user experience and saving time for the customer support team.

Technically, I used **Spring Boot** to develop RESTful APIs for backend operations, and **Thymeleaf** for server-side rendering of the admin interface, allowing quick updates of data directly from the server. **NextJS** was used for the frontend, providing a modern user interface with fast page loading and optimized SEO. The combination of **client-side rendering** and **server-side rendering** enhances the overall speed and performance of the system.

With the integration of Redis, the system's data handling capabilities are greatly enhanced, reducing data retrieval times and supporting seamless business operations. The result of this thesis is a flexible and robust supply chain management system that meets the management needs of modern enterprises. This system not only optimizes operational processes and costs but also provides a significant competitive advantage in the increasingly competitive market.

MỤC LỤC

[Ý KIẾN CHO PHÉP BẢO VỆ ĐỒ ÁN/ KHÓA LUẬN TỐT NGHIỆP 2](#_Toc194744023)

[CỦA GIẢNG VIÊN HƯỚNG DẪN 2](#_Toc194744024)

[LỜI CẢM ƠN 3](#_Toc194744025)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 4](#_Toc194744026)

[TÓM TẮT KHÓA LUẬN 5](#_Toc194744027)

[ABSTRACT 6](#_Toc194744028)

[MỤC LỤC 7](#_Toc194744029)

[DANH MỤC TỪ VIẾT TẮT 10](#_Toc194744030)

[DANH MỤC HÌNH VẼ 11](#_Toc194744031)

[DANH MỤC BẢNG 14](#_Toc194744032)

[MỞ ĐẦU 16](#_Toc194744033)

[Chương 1. TỔNG QUAN ĐỀ TÀI 17](#_Toc194744034)

[1.1. Giới thiệu đề tài 18](#_Toc194744035)

[1.2. Lý do chọn đề tài 18](#_Toc194744036)

[1.3. Mục tiêu và phạm vi đề tài 19](#_Toc194744037)

[1.4. Phương pháp nghiên cứu 21](#_Toc194744038)

[1.5. Bố cục báo cáo 22](#_Toc194744039)

[Chương 2. CƠ SỞ LÝ THUYẾT 23](#_Toc194744040)

[2.1. BackEnd Framework - Spring Boot 23](#_Toc194744041)

[2.1.1. Spring Boot là gì? 23](#_Toc194744042)

[2.1.2. Tại sao lại chọn Spring Boot 23](#_Toc194744043)

[2.1.3. Cách thực hiện tạo dự án với Spring Boot? 26](#_Toc194744044)

[2.1.4. Thymeleaf: 30](#_Toc194744045)

[2.1.5. Spring Security 32](#_Toc194744046)

[2.2. FrontEnd Framework-NextJS 38](#_Toc194744047)

[2.2.1. Giới thiệu về NextJS 38](#_Toc194744048)

[2.2.2. Tại sao nên sử dụng NextJS 39](#_Toc194744049)

[2.2.3. Cách cài đặt 1 dự án với Framework NextJS 44](#_Toc194744050)

[2.3. Các công cụ liên quan khác 47](#_Toc194744051)

[2.3.1. Redis và Docker 47](#_Toc194744052)

[2.3.2. PyTorch - Dự đoán doanh thu 50](#_Toc194744053)

[2.3.3. Rasa – ChatBot 55](#_Toc194744054)

[Chương 3. HỆ THỐNG QUẢN LÝ CHUỖI CUNG ỨNG 60](#_Toc194744055)

[3.1. Giới thiệu bài toán 60](#_Toc194744056)

[3.2. Phân tích hệ thống 60](#_Toc194744057)

[3.2.1. Sơ đồ use case đặt hàng từ nhà cung cấp 60](#_Toc194744058)

[3.2.2. Đặc tả use case đặt hàng từ nhà cung cấp 61](#_Toc194744059)

[3.2.3. Sơ đồ use case tạo đơn hàng sản xuất 63](#_Toc194744060)

[3.2.4. Đặc tả use case tạo đơn hàng sản xuất 63](#_Toc194744061)

[3.2.5. Use case mua hàng từ đại lý và theo dõi trạng thái đơn hàng 65](#_Toc194744062)

[3.2.6. Đặc tả use case mua hàng từ đại lý 66](#_Toc194744063)

[3.3. Thiết kế hệ thống 68](#_Toc194744064)

[3.3.1. Sơ đồ tuần tự (sequence diagram) 68](#_Toc194744065)

[3.3.2. Sơ đồ hoạt động (activity diagram) 72](#_Toc194744066)

[3.3.3. Thiết kế cơ sở dữ liệu 75](#_Toc194744067)

[3.4. Kiến trúc hệ thống 82](#_Toc194744068)

[3.4.1. Tổng quan hệ thống: 82](#_Toc194744069)

[3.5. Kết quả thực hiện 84](#_Toc194744070)

[3.5.1. Đăng nhập vào hệ thống: 84](#_Toc194744071)

[3.5.2. Đặt hàng từ nhà cung cấp 85](#_Toc194744072)

[3.5.3. Tạo đơn hàng sản xuất 86](#_Toc194744073)

[3.5.4. Tạo đơn hàng xuất cho đại lý 87](#_Toc194744074)

[3.5.5. Thống kê báo cáo **Error! Bookmark not defined.**](#_Toc194744075)

[3.5.6. Dự đoán doanh số sản phẩm 90](#_Toc194744076)

[3.5.7. Chatbot 92](#_Toc194744077)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 94](#_Toc194744078)

[4.1. Kết luận 94](#_Toc194744079)

[4.2. Hướng phát triển 95](#_Toc194744080)

[TÀI LIỆU THAM KHẢO 99](#_Toc194744081)

[PHỤ LỤC 100](#_Toc194744082)

DANH MỤC TỪ VIẾT TẮT

Danh mục từ viết tắt

|  |
| --- |
| QL: Quản lý |
| ĐH: Đơn hàng |
| NCC: Nhà cung cấp |
| KH: Khách hàng |
| SP: Sản phẩm |
| API: Application Programming Interface |
| CSR: Client-Side Rendering |
| SSR: Server-Side Rendering |

DANH MỤC HÌNH VẼ

[Hình 2.1 Mô hình chuỗi cung ứng 18](#_Toc194176153)

[Hình 2.1 Cây gia phả nhà Spring 23](#_Toc194176154)

[Hình 2.2 so sánh giữa Spring Boot và Spring truyền thống 24](#_Toc194176155)

[Hình 2.3 Trang tạo project Spring Boot **Error! Bookmark not defined.**](#_Toc194176156)

[Hình 2.4 cấu trúc dự án **Error! Bookmark not defined.**](#_Toc194176157)

[Hình 2.5 trang web đầu tiên khi truy cập vào đường **Error! Bookmark not defined.**](#_Toc194176158)

[Hình 2.6 Thực thể kho vận **Error! Bookmark not defined.**](#_Toc194176159)

[Hình 2.7 Luồng chạy của Spring boot 27](#_Toc194176160)

[Hình 2.8 Sơ đồ chứng thực tài khoản 33](#_Toc194176161)

[Hình 2.9 Giới thiệu về NextJS 38](#_Toc194176162)

[Hình 2.10 Tạo thư mục theo cấu trúc 39](#_Toc194176163)

[Hình 2.11 Quy trình tải trang với Server Side Rendering 41](#_Toc194176164)

[Hình 2.12: Quy trình tải trang với Client Side Rendering 43](#_Toc194176165)

[Hình 2.13 Cấu trúc dự án khi bắt đầu vào 45](#_Toc194176166)

[Hình 2.14 File cấu hình tailwind.config.ts 46](#_Toc194176167)

[Hình 2.15 File \_app.TS 46](#_Toc194176168)

[Hình 2.16 Luồng lấy dữ liệu khi có redis 48](#_Toc194176169)

[Hình 2.17 Dependency cần thiết cho redis 49](#_Toc194176170)

[Hình 2.18 Cấu hình redis 49](#_Toc194176171)

[Hình 2.19 Cấu trúc Rasa 56](#_Toc194176172)

[Hình 3.1 Sơ đồ use case về đặt hàng từ nhà cung cấp 61](#_Toc194176173)

[Hình 3.2 use case tạo đơn hàng sản xuất 63](#_Toc194176174)

[Hình 3.3 sơ đồ use case mua hàng từ đại lý và theo dõi trạng thái đơn hàng 66](#_Toc194176175)

[Hình 3.4 Sơ đồ tuần tự quy trình đặt hàng, xác nhận và cập nhật thanh toán **Error! Bookmark not defined.**](#_Toc194176176)

[Hình 3.5 sơ đồ quy trình cập nhật nguyên vật liệu và yêu cầu thống kê 68](#_Toc194176177)

[Hình 3.6 Sơ đồ quy trình đặt hàng của đại lý và xác nhận thanh toán **Error! Bookmark not defined.**](#_Toc194176178)

[Hình 3.7 sơ đồ lên lịch giao hàng và xem tình trạng đơn hàng **Error! Bookmark not defined.**](#_Toc194176179)

[Hình 3.8 Sơ đồ trạng thái đơn hàng 69](#_Toc194176180)

[Hình 3.9 Sơ đồ quy trình tạo đơn hàng sản xuất **Error! Bookmark not defined.**](#_Toc194176181)

[Hình 3.10 Cập nhật quy trình sản xuất. **Error! Bookmark not defined.**](#_Toc194176182)

[Hình 3.11 Sơ đồ trạng thái cho đơn hàng sản xuất 69](#_Toc194176183)

[Hình 3.12 sơ đồ quy trình cập nhật sản phẩm và thống kê sản phẩm 70](#_Toc194176184)

[Hình 3.13 Sơ đồ quy trình đánh giá đại lý và yêu cầu thống kê hiệu suất 71](#_Toc194176185)

[Hình 3.14 Sơ đồ hoạt động về quy trình mua hàng từ nhà cung cấp 72](#_Toc194176186)

[Hình 3.15 Sơ đồ hoạt động cho quy trình đặt hàng 73](#_Toc194176187)

[Hình 3.16 Sơ đồ hoạt động cho quy trình đặt hàng 74](#_Toc194176188)

[Hình 3.17 Các mối quan hệ ràng buộc cho nghiệp vụ mua hàng 75](#_Toc194176189)

[Hình 3.18 Các mối quan hệ ràng buộc giao đơn hàng 76](#_Toc194176190)

[Hình 3.19 Các mối quan hệ ràng buộc đánh giá cộng tác 77](#_Toc194176191)

[Hình 3.20 các mối quan hệ ràng buộc quy trình mua hàng từ nhà cung cấp 78](#_Toc194176192)

[Hình 3.21 các mối quan hệ ràng buộc quy trình tồn kho 79](#_Toc194176193)

[Hình 3.22 các mối quan hệ của quản lý đơn hàng sản xuất 81](#_Toc194176194)

[Hình 3.23 Luồng xử lý dữ liệu 82](#_Toc194176195)

[Hình 3.24 Luồng chứng thực người dùng 83](#_Toc194176196)

[Hình 3.25 Trang đăng nhập cho nhân viên và khách hàng 84](#_Toc194176197)

[Hình 3.26 Giao diện khi khách hàng đăng nhập thành công 84](#_Toc194176198)

[Hình 3.27 Giao diện khi đăng nhập với vai trò là nhân viên 85](#_Toc194176199)

[Hình 3.28 Giao diện trang đặt hàng từ nhà cung 85](#_Toc194176200)

[Hình 3.29 Mail được gửi khi hoàn tất đặt hàng từ nhà cung cấp 86](#_Toc194176201)

[Hình 3.30 Giao diện tạo đơn hàng sản xuất 87](#_Toc194176202)

[Hình 3.31 Giao diện giỏ hàng 88](#_Toc194176203)

[Hình 3.32 Giao diện xác nhận đơn hàng 88](#_Toc194176204)

[Hình 3.33 Giao diện thanh toán với cổng VnPay 89](#_Toc194176205)

[Hình 3.34 Giao diện sau khi thanh toán thành công 89](#_Toc194176206)

[Hình 3.35 Giao diện xem trạng thái đơn hàng **Error! Bookmark not defined.**](#_Toc194176207)

[Hình 3.36 Thống kê báo cáo tổng quan **Error! Bookmark not defined.**](#_Toc194176208)

[Hình 3.37 Biểu đồ so sánh giá trị thực với giá trị đánh giá 91](#_Toc194176209)

[Hình 3.38 Biểu đồ phân bổ giá trị sai số 91](#_Toc194176210)

[Hình 3.39 Tính năng tích hợp vào hệ thống 92](#_Toc194176211)

[Hình 3.40 Chat với AI 92](#_Toc194176212)

[Hình 3.41 Chat ở trang chủ nhân viên 93](#_Toc194176213)

DANH MỤC BẢNG

[Bảng 2.1 phương thức khởi chạy Spring Boot 25](#_Toc194176214)

[Bảng 2.2 Tệp tin chính của spring boot **Error! Bookmark not defined.**](#_Toc194176215)

[Bảng 2.3 Dependency cần thiết cho Spring Boot **Error! Bookmark not defined.**](#_Toc194176216)

[Bảng 2.4 Cấu hình port ứng dụng **Error! Bookmark not defined.**](#_Toc194176217)

[Bảng 2.5 Cấu hình kết nối cơ sở dữ liệu **Error! Bookmark not defined.**](#_Toc194176218)

[Bảng 2.6 Dependency để ánh xạ xuống MySQL **Error! Bookmark not defined.**](#_Toc194176219)

[Bảng 2.7 Class controller để xử lý dữ liệu 28](#_Toc194176220)

[Bảng 2.8 Code ở APIController 29](#_Toc194176221)

[Bảng 2.9 Code ở lớp service **Error! Bookmark not defined.**](#_Toc194176222)

[Bảng 2.10 code repository **Error! Bookmark not defined.**](#_Toc194176223)

[Bảng 2.11 Code Specification **Error! Bookmark not defined.**](#_Toc194176224)

[Bảng 2.12 Truy vấn dữ liệu với annotation Query **Error! Bookmark not defined.**](#_Toc194176225)

[Bảng 2.13 Code giao diện HTML với Thymeleaf **Error! Bookmark not defined.**](#_Toc194176226)

[Bảng 2.14 cập nhật thymeleaf ở pom.xml 31](#_Toc194176227)

[Bảng 2.15 code controller truyền dữ liệu 31](#_Toc194176228)

[Bảng 2.16 lặp danh sách dữ liệu với thymeleaf 32](#_Toc194176229)

[Bảng 2.17 Dependency cần thiết cho Spring Security 35](#_Toc194176230)

[Bảng 2.18 Entity class cho người dùng của hệ thống 35](#_Toc194176231)

[Bảng 2.19 ghi đè lớp lại lớp cha loadUserByUserName 36](#_Toc194176232)

[Bảng 2.20 Cấu hình Spring Security 37](#_Toc194176233)

[Bảng 2.21 cách sử dụng thẻ link 40](#_Toc194176234)

[Bảng 2.22 Câu lệnh bắt đầu cài đặt dự án NextJS 44](#_Toc194176235)

[Bảng 2.23 các lựa chọn để tạo cấu hình 45](#_Toc194176236)

[Bảng 2.24 Câu lệnh để khởi lại dự án 45](#_Toc194176237)

[Bảng 2.25 Câu lệnh cài đặt redis trên server 49](#_Toc194176238)

[Bảng 2.26 câu lệnh cài đặt Pytorch 50](#_Toc194176239)

[Bảng 2.27 Code thu thập dữ liệu 51](#_Toc194176240)

[Bảng 2.28 Code xây dựng mô hình hồi quy 52](#_Toc194176241)

[Bảng 2.29 Code hàm mất mát và tối ưu với optmizer 52](#_Toc194176242)

[Bảng 2.30 Code huấn luyện mô hình 53](#_Toc194176243)

[Bảng 2.31 Code dự đoán mô hình 53](#_Toc194176244)

[Bảng 2.32 code đơn giản ở file domain.yml 57](#_Toc194176245)

[Bảng 2.33 code đơn giản ở file nlu.yml 57](#_Toc194176246)

[Bảng 2.34 code đơn giản ở file rule.yml 58](#_Toc194176247)

[Bảng 2.35 code đơn giản với file config 59](#_Toc194176248)

[Bảng 2.36 cấu trúc json để gửi request 59](#_Toc194176249)

[Bảng 3.1 Đặc tả quy trình đặt hàng với nhà cung cấp 62](#_Toc194176250)

[Bảng 3.2 đặc tả use tạo đơn hàng sản xuất 65](#_Toc194176251)

[Bảng 3.3 Đặc tả use case mua hàng từ đại lý 67](#_Toc194176252)

MỞ ĐẦU

Trong môi trường kinh doanh ngày nay, quản lý chuỗi cung ứng hiệu quả đóng vai trò vô cùng quan trọng để duy trì lợi thế cạnh tranh. Nhiều doanh nghiệp đang đối mặt với các thách thức như quản lý kho không hiệu quả, theo dõi đơn hàng không chính xác và quy trình sản xuất chưa tối ưu. Để giải quyết những vấn đề này, khóa luận của tôi tập trung vào phát triển một hệ thống quản lý chuỗi cung ứng hiện đại, tận dụng các công nghệ tiên tiến.

Mục tiêu chính của khóa luận là xây dựng một giải pháp mạnh mẽ và có khả năng mở rộng, nhằm tối ưu hóa các chức năng quan trọng như quản lý nhà cung cấp, xử lý đơn hàng và theo dõi kho. Bằng cách tích hợp các tính năng cần thiết và sử dụng các framework hiện đại như Spring Boot và NextJS, hệ thống này hy vọng sẽ nâng cao hiệu quả vận hành và cải thiện khả năng ra quyết định cho các doanh nghiệp.

Khóa luận cũng tích hợp thêm trí tuệ nhân tạo (AI) để dự báo doanh thu bằng cách sử dụng PyTorch, cùng với việc phát triển một chatbot AI sử dụng Rasa để cải thiện trải nghiệm khách hàng. AI giúp dự báo doanh thu dựa trên dữ liệu lịch sử, hỗ trợ doanh nghiệp đưa ra các quyết định chính xác hơn. Chatbot AI được huấn luyện để trả lời các câu hỏi cơ bản của khách hàng, giúp tiết kiệm thời gian cho đội ngũ hỗ trợ và nâng cao chất lượng dịch vụ.

Phương pháp nghiên cứu sẽ sử dụng các API RESTful để đảm bảo sự giao tiếp liền mạch giữa frontend và backend, đồng thời triển khai server-side rendering để cập nhật dữ liệu nhanh chóng. Khóa luận sẽ tập trung vào việc kết hợp các công nghệ hiện đại và trí tuệ nhân tạo để xây dựng một hệ thống mạnh mẽ và hiệu quả.

Cấu trúc của khóa luận sẽ bắt đầu với cái nhìn tổng quan về các công nghệ được sử dụng, sau đó là phần thảo luận chi tiết về kiến trúc hệ thống, các bước thực hiện và cuối cùng là phần kết luận tóm tắt những phát hiện và hướng nghiên cứu trong tương lai.

# TỔNG QUAN ĐỀ TÀI

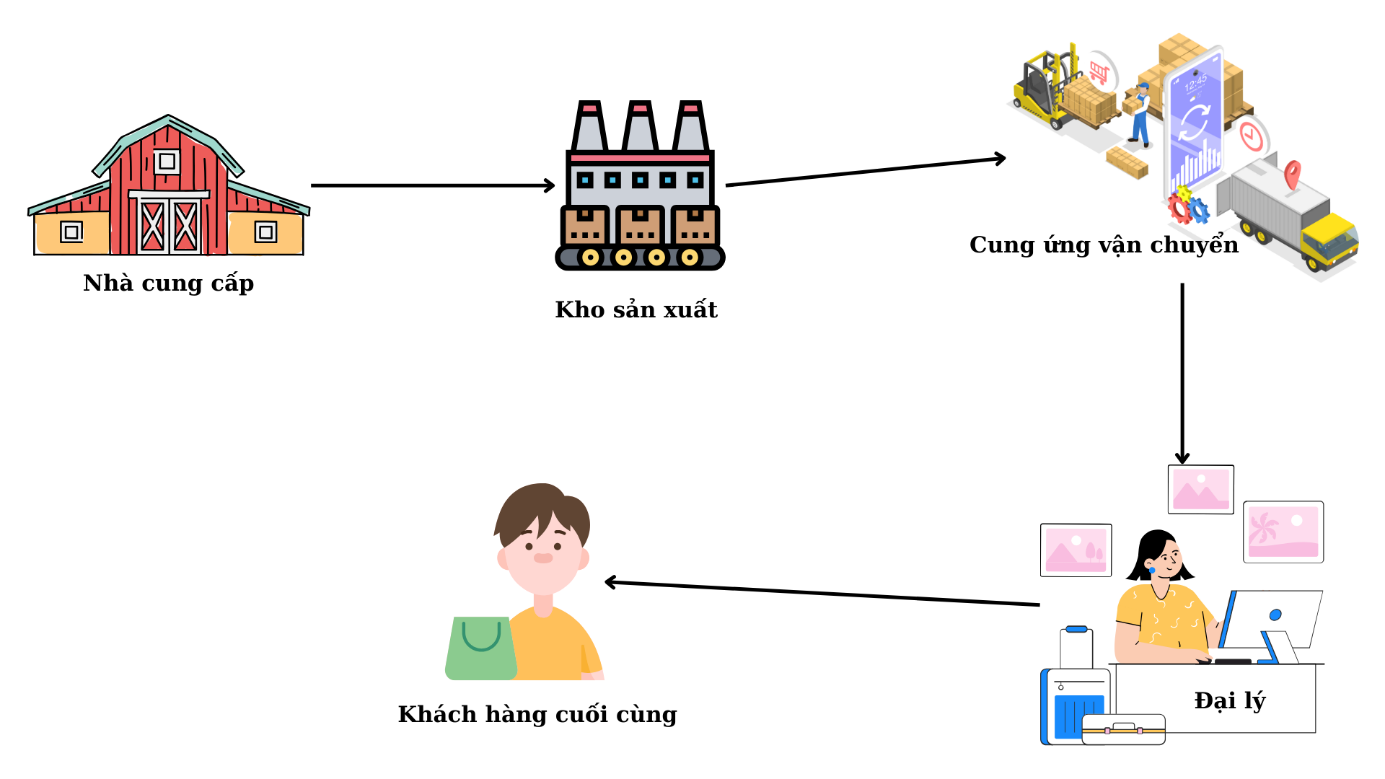
Trong bối cảnh cạnh tranh khốc liệt của thị trường hiện nay, các doanh nghiệp và cửa hàng luôn phải đối mặt với áp lực tối ưu hóa chi phí hoạt động, đặc biệt khi mở rộng hệ thống đại lý hoặc nhượng quyền thương hiệu. Một hệ thống quản lý chuỗi cung ứng hiệu quả chính là chìa khóa giúp doanh nghiệp kiểm soát toàn diện quá trình từ khâu nhập hàng từ nhà cung cấp đến phân phối sản phẩm cho các chi nhánh, đại lý, đảm bảo sự liên kết chặt chẽ giữa các khâu trong chuỗi cung ứng.

Việc áp dụng hệ thống quản lý chuỗi cung ứng không chỉ giúp doanh nghiệp giảm thiểu các chi phí không cần thiết mà còn tăng cường tính minh bạch và hiệu quả trong quá trình điều phối hàng hóa, quản lý kho bãi và tối ưu hóa luồng vận chuyển sản phẩm. Hệ thống này cho phép doanh nghiệp kiểm soát và giám sát mọi khâu từ nguyên vật liệu đầu vào đến sản phẩm đầu ra, đồng thời hỗ trợ ra quyết định dựa trên dữ liệu, nâng cao hiệu quả hoạt động.

Đề tài này tập trung phát triển một hệ thống quản lý chuỗi cung ứng hiện đại, không chỉ đáp ứng các yêu cầu về tối ưu hóa quy trình và chi phí mà còn tích hợp trí tuệ nhân tạo (AI) để dự báo doanh thu và cải thiện khả năng phản hồi của hệ thống qua chatbot AI. Hệ thống sẽ giúp doanh nghiệp có thể dự báo chính xác doanh thu trong tương lai, từ đó đưa ra các quyết định kịp thời và phù hợp. Đồng thời, chatbot AI sẽ hỗ trợ việc trả lời các câu hỏi cơ bản của khách hàng, giảm tải cho đội ngũ hỗ trợ và nâng cao trải nghiệm người dùng.

Hệ thống này sẽ là công cụ mạnh mẽ, giúp doanh nghiệp không chỉ duy trì sự ổn định mà còn nâng cao hiệu suất và khả năng cạnh tranh. Nhờ vào việc tích hợp các công nghệ tiên tiến, hệ thống này hứa hẹn mang lại lợi thế cạnh tranh bền vững cho doanh nghiệp trong bối cảnh thị trường luôn biến động.

## Giới thiệu đề tài



Hình . Mô hình chuỗi cung ứng

## Lý do chọn đề tài

Trước đây, em đã có cơ hội trải nghiệm môi trường kinh doanh tại quán cà phê và nhận thấy nhiều khó khăn trong việc quản lý nguồn cung cấp, vận hành và kiểm soát sản phẩm. Hầu hết các cửa hàng vẫn đang phụ thuộc nhiều vào các thao tác thủ công để quản lý hàng hóa, từ nguyên liệu đầu vào cho đến sản phẩm đầu ra, dẫn đến việc khó kiểm soát tồn kho, quản lý đơn hàng từ các nhà cung cấp, và phân phối sản phẩm đến các chi nhánh hoặc đại lý con. Điều này không chỉ gây lãng phí thời gian mà còn tạo ra nhiều sai sót, đặc biệt là khi quy mô cửa hàng mở rộng.

Trong bối cảnh cạnh tranh ngày càng cao, việc tối ưu hóa chuỗi cung ứng trở thành một yếu tố sống còn cho các doanh nghiệp. Một hệ thống quản lý chuỗi cung ứng hiệu quả có thể giúp các doanh nghiệp kiểm soát tốt hơn quá trình từ nhà cung cấp đến phân phối, đảm bảo tính minh bạch và hiệu quả trong từng khâu. Từ những kinh nghiệm thực tiễn đã trải qua, kết hợp với kiến thức học tại trường, em nhận ra rằng việc phát triển một hệ thống quản lý chuỗi cung ứng hiện đại sẽ giúp giảm thiểu sai sót trong quá trình vận hành, tăng hiệu suất làm việc, và tạo lợi thế cạnh tranh cho các doanh nghiệp.

Lý do em chọn đề tài này là vì nó phản ánh đúng nhu cầu thực tế của các doanh nghiệp đang mở rộng quy mô và tìm kiếm những phương thức hiệu quả để quản lý chuỗi cung ứng. Hệ thống này không chỉ giúp doanh nghiệp giảm thiểu chi phí mà còn đảm bảo việc kiểm soát chất lượng sản phẩm từ nguồn cung cấp đến tay các đại lý, đồng thời tăng cường khả năng đáp ứng nhanh với các thay đổi của thị trường. Đặc biệt, việc tích hợp các công nghệ như **trí tuệ nhân tạo (AI)** để dự báo doanh thu và chatbot AI hỗ trợ khách hàng sẽ giúp hệ thống nâng cao hiệu quả, đưa ra các quyết định chính xác hơn, đồng thời tiết kiệm thời gian và nguồn lực. Chính nhờ khả năng quản lý chuỗi cung ứng toàn diện và áp dụng công nghệ hiện đại, các doanh nghiệp có thể gia tăng sự cạnh tranh, cải thiện dịch vụ, và tối ưu hóa quá trình kinh doanh một cách bền vững.

## Mục tiêu và phạm vi đề tài

Phát triển hệ thống chuỗi cung ứng từ việc quản lý các mặt hàng được nhập từ nhà cung cấp cho đến quy trình sản xuất sản phẩm, theo dõi đơn hàng được giao cho các đại lý, và quản lý quá trình vận chuyển hàng hóa giữa các kho hàng, đại lý. Hệ thống này sẽ tích hợp AI (NLP) để hỗ trợ việc theo dõi và tối ưu hóa chuỗi cung ứng, đồng thời giúp tự động hóa quy trình liên lạc với nhà cung cấp và phân phối. Bên cạnh đó, hệ thống sẽ hỗ trợ các phương thức thanh toán trực tuyến phổ biến hiện nay như VNPay, đồng thời kết hợp Redis để tăng cường hiệu suất xử lý dữ liệu và tối ưu tốc độ truy xuất thông tin.Quản lý nhà cung cấp: cho phép nhân viên quản lý cửa hàng thêm/sửa/ xóa thông tin của các nhà cung cấp, Lưu trữ và cũng như quản lý các thông tin cần thiết của nhà cung cấp, Phân tích và đánh giá để đưa ra cách để quản lý trong việc tối ưu lựa chọn nguồn hàng cho cửa hàng.

Quản lý vật liệu nhập: Cho phép quản lý thêm/sửa/xóa các thông tin về các vật liệu được nhập từ nhà cung cấp. Quản lý các loại vật liệu phục vụ sản xuất để có thể dễ dàng tìm kiếm vật liệu khi cần..

Quản lý đơn giá và thanh toán: Cho phép nhân viên quản lý cập nhật về giá của vật liệu, xem các lịch sử thay đổi giá vật liệu của nhà cung cấp cũng như xem các chi phí liên quan đơn hàng khi đặt từ nhà cung cấp đó có tối ưu hơn so với các nhà cung cấp khác trên thị trường hiện nay không và quản lý các phương thức thanh toán từ tiền mặt đến thanh toán trực tuyến cho tất cả đơn hàng.

Quản lý quy trình sản xuất: Quản lý tạo công thức để ra được một sản phẩm thì sẽ gồm những vật liệu gì nhằm để quản lý khi bán hàng cho khách hàng trực tiếp hay các giao hàng cho các đại lý con thì sẽ đã tốn bao nhiêu vật liệu, so sánh thông tin từ hệ thống và số liệu kiểm kho để nắm bắt được sự thiếu hụt về hàng hóa.

Quản lý sản phẩm: Cho phép quản lý của cửa hàng thêm/ sửa/ xóa các thông tin sản phẩm của cửa hàng, thống kê các sản phẩm bán chạy để khách hàng nắm bắt thông tin về sản phẩm để lựa chọn mua sản phẩm cũng như cho các nhà đại lý có thể lựa chọn lấy hàng về cho khách hàng của mình, cũng như cửa hàng tăng cường sản xuất các mặt hàng bán chạy.

Quản lý đơn hàng: Cho phép người dùng nhân viên tạo các đơn hàng nhập từ nhà cung cấp và đại lý cả nhân viên đều có thể tạo đơn hàng xuất, theo dõi các đơn hàng xuất và trạng thái đơn hàng, quản lý đơn hàng xuất và nhập cho cửa hàng có thể nắm thông tin về doanh thu hay số lượng thanh toán ở các phương thức để dễ dàng nắm bắt số liệu doanh thu và báo cáo, cung cấp tính năng xác nhận đơn hàng và hủy đơn hàng khi cần thiết.

Quản lý vận chuyển và giao hàng: Quản lý các đối tác vận chuyển của cửa đến các đại lý con và khách hàng, nhân viên quản lý sẽ lên lịch giao hàng từ địa điểm cửa hàng đến địa điểm yêu cầu cho đại lý, cập nhật vị trí hàng hóa trong ở kho nào của cửa hàng.

Quản lý khách hàng: Hệ thống sẽ lưu trữ thông tin của khách hàng cuối cùng cần sự trợ giúp với cửa hàng. Khách hàng có thể truy cập vào hệ thống để cung cấp thông tin khi cần hỗ trợ.

Quản lý kho hàng: Hệ thống quản lý kho hàng cho phép theo dõi vị trí, số lượng tồn kho, và tình trạng đơn hàng trong từng kho. Nhân viên có thể cập nhật trạng thái hàng hóa khi vận chuyển giữa các kho hoặc đến các đại lý con. Hệ thống cũng hỗ trợ tối ưu hóa không gian lưu trữ, điều phối hàng hóa hiệu quả và cung cấp báo cáo về hiệu suất kho vận, giúp doanh nghiệp giảm thiểu chi phí và nâng cao hiệu quả vận hành..

Thống kê báo cáo: Hệ thống sẽ cung cấp các thống kê và báo cáo để cửa hàng có cái nhìn tổng quan về doanh thu, số lượng tồn kho, Đơn hàng sản xuất, cũng như đánh giá hiệu quả của các đối tác (bao gồm đại lý, nhà vận chuyển và nhà cung cấp) cùng số lượng hàng tồn kho.

## Phương pháp nghiên cứu

Để thực hiện nghiên cứu và phát triển hệ thống quản lý chuỗi cung ứng hiệu quả, em sẽ áp dụng các bước nhằm quản lý các nghiệp vụ liên quan đến đơn hàng, vật liệu, sản phẩm, nhà cung cấp và các đối tác vận chuyển. Hệ thống sẽ cho phép tạo và cập nhật dữ liệu cho các đơn hàng, vật liệu, sản phẩm và nhà cung cấp, đồng thời tích hợp các chức năng tìm kiếm để dễ dàng truy xuất thông tin cần thiết. Bên cạnh đó, em sẽ phát triển hệ thống báo cáo chi tiết bao gồm báo cáo doanh thu, hiệu suất hoạt động của các nhà cung cấp, đối tác vận chuyển và số lượng hàng tồn kho, giúp cung cấp cái nhìn tổng quan về hiệu suất kinh doanh. Để phát triển hệ thống, em sẽ sử dụng **Spring Boot** cho backend, với dữ liệu được lưu trữ trong cơ sở dữ liệu **MySQL Workbench**, đảm bảo tính nhất quán và bảo mật. Để cải thiện hiệu suất và khả năng mở rộng, em sẽ tích hợp **Redis (NoSQL)** nhằm tối ưu hóa tốc độ truy xuất dữ liệu và tăng cường hiệu suất xử lý. **NextJS** sẽ được sử dụng để thiết kế giao diện người dùng, giúp mang lại trải nghiệm trực quan và thân thiện. Đặc biệt, hệ thống sẽ được tích hợp **AI** để dự báo doanh thu thông qua mô hình học sâu sử dụng **PyTorch**, giúp doanh nghiệp dự báo chính xác doanh thu trong tương lai dựa trên dữ liệu lịch sử và xu hướng thị trường. Đồng thời, em cũng sẽ phát triển một **chatbot AI** sử dụng **Rasa** để hỗ trợ trả lời các câu hỏi cơ bản từ khách hàng, giúp nâng cao trải nghiệm người dùng và giảm tải cho bộ phận hỗ trợ khách hàng. Với việc áp dụng các công nghệ hiện đại như vậy, hệ thống sẽ được tối ưu hóa để đáp ứng nhu cầu quản lý chuỗi cung ứng, góp phần nâng cao hiệu suất hoạt động và năng lực cạnh tranh cho doanh nghiệp.

## Bố cục báo cáo

Báo cáo sẽ có bố cục là 4 chương, nội dung như sau:

Chương 1. Tổng quan đề tài: đi vào giới thiệu về đề tài cũng như lý do em chọn đề tài, phạm vi nghiên cứu và các phương pháp được sử dụng trong quá trình phát triển hệ thống. Nội dung sẽ cung cấp cái nhìn tổng quát về hệ thống chuỗi cung ứng, các vấn đề tài nhắm tới, cũng như hướng đi của nghiên cứu.

Chương 2. Cơ sở lý thuyết: Chương này giới thiệu về các công nghệ được sử dụng trong phát triển hệ thống, bao gồm Spring Boot cho backend và NextJS cho frontend. Spring Boot được chọn vì khả năng xây dựng API mạnh mẽ, dễ mở rộng, trong khi NextJS hỗ trợ server-side rendering để tối ưu hóa tốc độ và SEO. Hệ thống cũng tích hợp AI với PyTorch để dự báo doanh thu và Rasa để phát triển chatbot AI, giúp tối ưu hóa quy trình quản lý chuỗi cung ứng và nâng cao trải nghiệm người dùng.

Chương 3. Phân tích thiết kế hệ thống: Mô tả chi tiết và phân tích chi tiết yêu cầu hệ thống cũng như nêu rõ các tính năng và các tác nhân của hệ thống với những tính năng nào cuối cùng là luồng đi của hệ thống giống người dùng có thể hiểu rõ và nắm bắt được hệ thống.

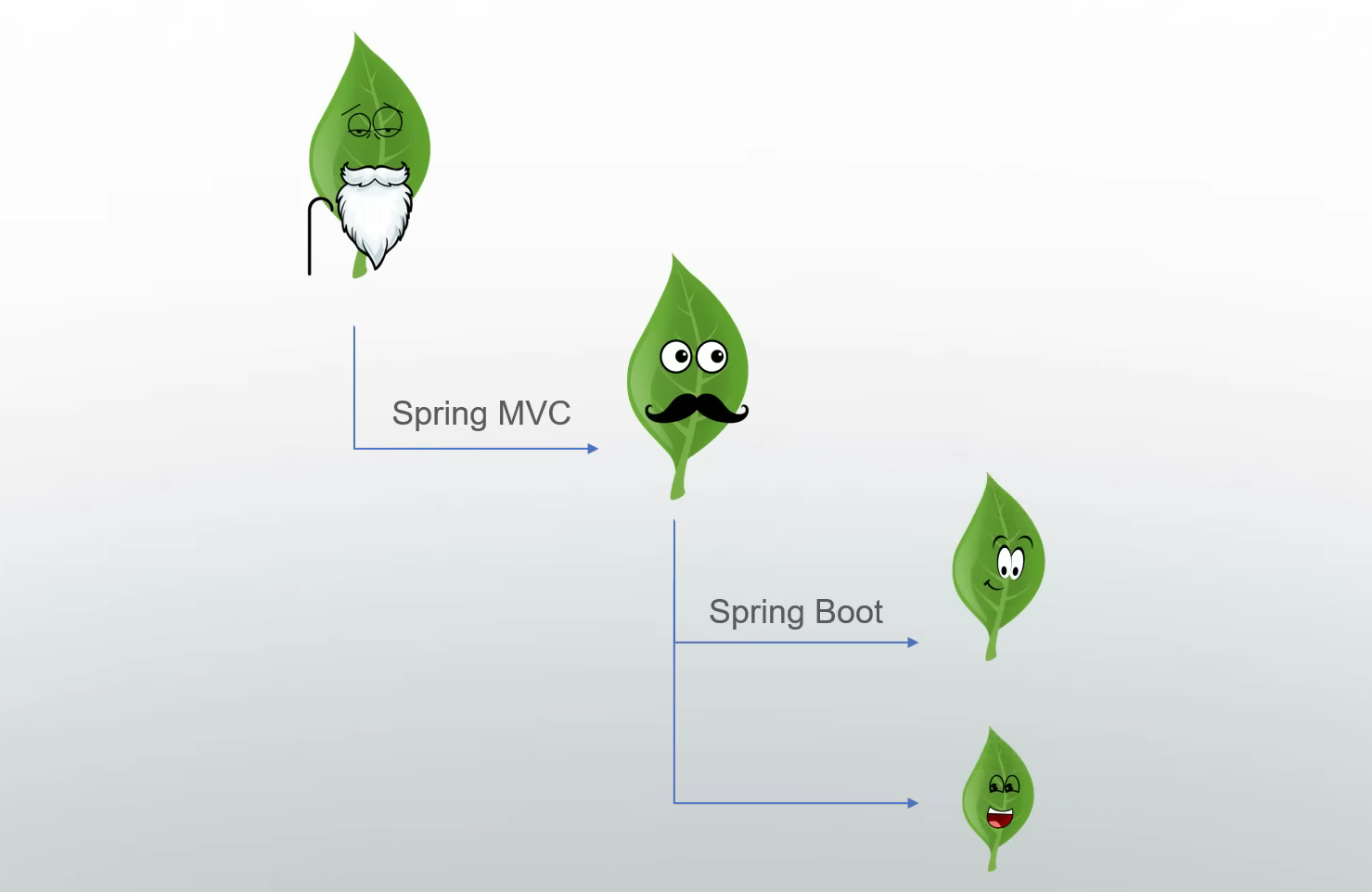
Chương 4. Kết luận và hướng phát triển: tổng kết lại toàn bộ quá trình phát triển hệ thống cũng như kết quả đạt được, chưa đạt được và các hạn chế còn tồn tại, chương này sẽ đánh giá tính hiệu quả của hệ thống chuỗi cung ứng mà em xây dựng bao gồm cái hiệu suất, bảo mật, và khả năng đáp ứng với thực tế. Đồng thời, em sẽ đề xuất các hướng phát triển tương lai để cải thiện hệ thống, như bổ sung tính năng mới, tối ưu hóa trải nghiệm người dùng, và mở rộng tích hợp với các công nghệ mới.

# CƠ SỞ LÝ THUYẾT

## BackEnd Framework - Spring Boot

### Spring Boot là gì?

Spring boot là một Framework mã nguồn mở được phát triển trên nền tảng Spring Framework, được sử dụng để thiết kế các ứng dụng các ứng dụng java nhanh chóng hơn, nó cung cấp nhiều tính năng hữu ích, giúp giảm thiểu các cấu hình phức tạp, Spring Boot cũng là xuất thân là Framework của đại gia đình Spring được phát hành chính thức vào năm 2014 tính cho đến thời điểm hiện nay Spring Boot chỉ đứng sau Spring MVC.



2

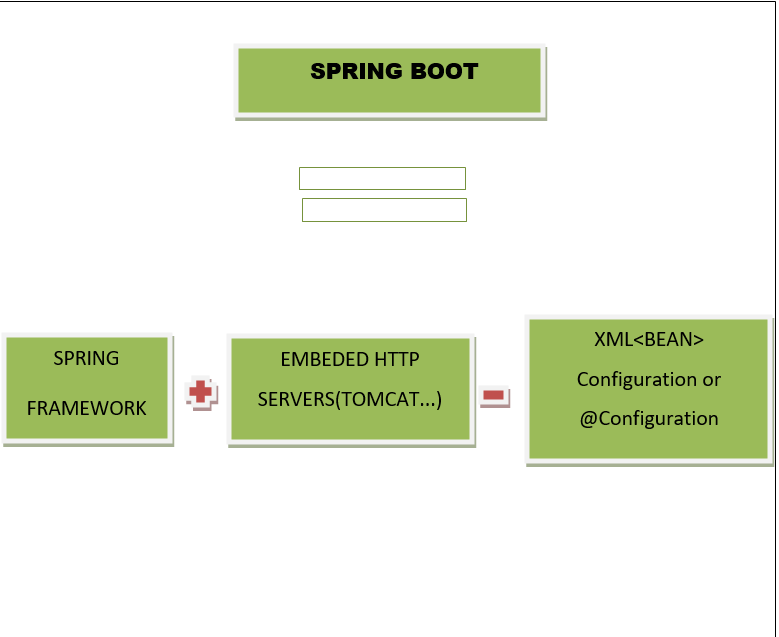
1

Hình . Cây gia phả nhà Spring

### Tại sao lại chọn Spring Boot

#### Spring Boot đơn giản hóa:

Nếu trước đây công đoạn khởi tạo một dự án diễn ra phức tạp với các Framework truyền thống có nhiều annotation hay các file xml phức tạp thì giờ đây Spring Boot là một Framework lựa chọn hàng đầu với sự tối ưu của nó mang lại với 1 cú nhấp chuột ta đã có thể tạo 1 file cấu hình và các dependency cơ bản hoặc ta có thể tùy chỉnh, Spring Boot hỗ trợ phát triển web theo mô hình microservices, xu hướng ngày càng phổ biến trong các công nghệ hiện đại, Spring Boot cung cấp khả năng tự động cấu hình với file .properties hoặc .yaml để cấu hình các configuration cần thiết và các thành phần như cơ sở dữ liệu, bảo mật, giúp ta tập trung vào nghiệp vụ hơn là cấu hình phức tạp.



Hình . so sánh giữa Spring Boot và Spring truyền thống

#### Những tính năng nổi bật của Spring Boot

Spring Boot đã khẳng định vị thế của mình là một Framework mạnh mẽ và linh hoạt nhất dành cho các nhà phát triển java nói riêng và các nhà phát triển web nói chung. Dưới đây là một số tính năng nổi bật của Spring Boot.

**Phát triển ứng dụng web dễ dàng:**

Spring Boot rất phù hợp cho việc phát triển các ứng dụng web nhờ vào module chuyên dụng như **spring-boot-starter-web**. Framework này hỗ trợ tích hợp các máy chủ HTTP nhúng như **Tomcat**, **Jetty** hoặc **Undertow**, giúp ứng dụng có thể chạy độc lập mà không cần cấu hình máy chủ ngoài. Nhờ đó, các ứng dụng có thể được khởi chạy một cách nhanh chóng mà không cần triển khai dưới dạng file WAR.

**SpringApplication – Khởi động ứng dụng dễ dàng:**

SpringApplication là một lớp quan trọng trong Spring Boot giúp khởi động ứng dụng một cách nhanh chóng và dễ dàng. Ta chỉ cần gọi phương thức tĩnh run() trong lớp main() để khởi chạy ứng dụng:

|  |
| --- |
| Java |
| @SpringBootApplication\*xuanphat11112003 public class HeThongChuoiCungUngApplication {  public static void main(String[] args) {  SpringApplication.run(HeThongChuoiCungUngApplication.class, args);  } } |

Bảng . phương thức khởi chạy Spring Boot

Điều này giúp đơn giản hóa quá trình khởi động, đồng thời quản lý vòng đời của ứng dụng một cách trực quan

**Quản lý sự kiện và Listener:**

Spring Boot hỗ trợ việc xử lý các Application Events để thực hiện các tác vụ khác nhau. Các Listener có thể được thêm vào để lắng nghe và xử lý các sự kiện của ứng dụng. Điều này giúp theo dõi và phản hồi các sự kiện xảy ra trong suốt vòng đời của ứng dụng một cách dễ dàng và hiệu quả.

**Cấu hình linh hoạt (Externalized Configuration):**

Spring Boot cho phép cấu hình ứng dụng theo nhiều cách khác nhau, đặc biệt hỗ trợ tốt cho việc thay đổi cấu hình ứng dụng theo từng môi trường (development, production, testing). Bạn có thể sử dụng các file cấu hình như .properties hoặc YAML để điều chỉnh cấu hình một cách dễ dàng mà không cần phải thay đổi mã nguồn.

**Bảo mật ứng dụng với Spring Security**

Spring Boot tích hợp sẵn **Spring Security**, giúp bảo vệ các ứng dụng web thông qua các phương pháp bảo mật tiêu chuẩn như **Basic Authentication** và **OAuth2**. Mặc định, Spring Boot tự động bảo vệ tất cả các endpoint HTTP, giúp bảo mật hệ thống một cách toàn diện.

**Cấu hình đơn giản với Starter Dependencies:**

Spring boot cung cấp các Start Dependencies giúp đơn giản hóa việc cấu hình phụ thuộc của dự án Maven. Thay vì phải cấu hình riêng lẻ, các start này bao gồm một tập hợp các thư viện cần thiết cho các mục đích phát triển web, dữ liệu, bảo mật v.v;

### Cách thực hiện tạo dự án với Spring Boot?

1. **Truy cập Spring Initializr:**

* Vào Spring Initializr, chọn các thông số như JDK, Maven/Gradle, và ngôn ngữ Java.
* Chọn các dependency như spring-boot-starter-web, spring-boot-starter-data-jpa, v.v.

1. **Truy cập Spring Initializr:**

* Vào Spring Initializr, chọn các thông số như JDK, Maven/Gradle, và ngôn ngữ Java.
* Chọn các dependency như spring-boot-starter-web, spring-boot-starter-data-jpa, v.v.

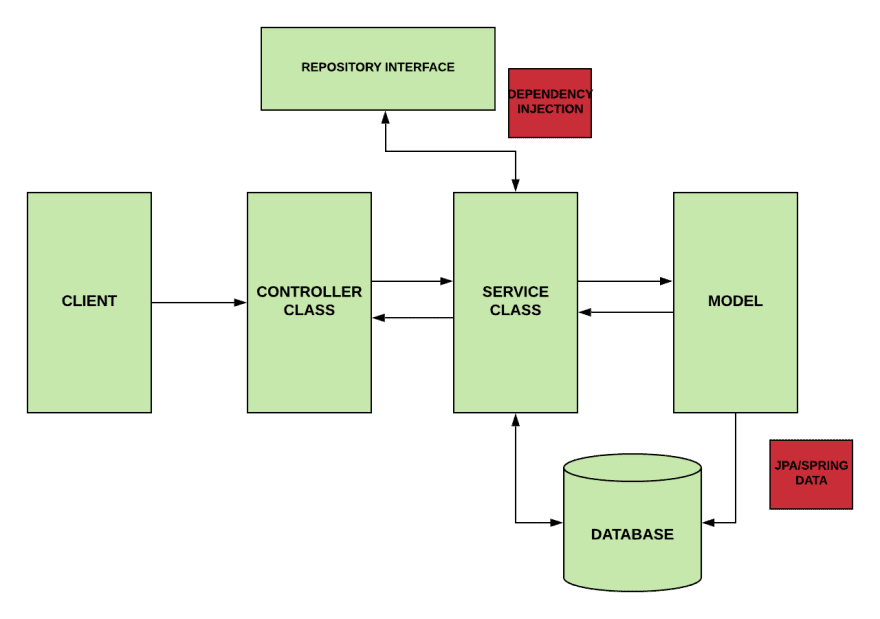
1. **Cấu hình cơ sở dữ liệu:**
   * Cấu hình cơ sở dữ liệu trong tệp application.properties:

|  |
| --- |
| Properties |
| spring.datasource.url=jdbc:mysql://localhost:3306/mydb  spring.datasource.username=root  spring.datasource.password=password |

Bảng . application.properties

1. **Chạy ứng dụng**:
   * Mở class chính với annotation @SpringBootApplication và chạy dự án trong IDE hoặc qua dòng lệnh bằng mvn spring-boot:run.

**Tiếp đến là quy trình làm việc của Spring Boot**:



Specification

JpaRepository

Hình . Luồng chạy của Spring boot

**Luồng của Spring Boot hoạt động như sau:**

1. **Client gửi HTTP request:**

* Khi người dùng tương tác với ứng dụng qua form hoặc URL, một HTTP request sẽ được gửi tới server.

2. **Controller xử lý request:**

* Controller sẽ nhận request và xử lý. Có 2 loại controller:
  + @Controller: Xử lý web request và trả về view (HTML, JSP).
  + @RestController: Dùng để xây dựng API RESTful và trả về dữ liệu dưới dạng JSON.

|  |
| --- |
| Java |
| @Controller  public class DXPController {  @GetMapping("/example")  public String handleRequestDXP(Model model) {  return "example"; // Trả về template Thymeleaf  }  } |

Bảng . Class controller để xử lý dữ liệu

3. **Controller xử lý request với annotations:**

* Controller sử dụng các annotations như @GetMapping, @PostMapping, @RequestMapping để xử lý loại request.

4**. Service xử lý logic nghiệp vụ:**

* Nếu cần xử lý logic nghiệp vụ, Controller sẽ gọi các Service được đánh dấu với @Service và được Autowired.

|  |
| --- |
| Java |
| @Service  public class CategoryService {  @Autowired  CategoryRepository categoryRepository;  public String doBusinessLogic() {  return categoryRepository.save(); // Logic nghiệp vụ  }  } |

Bảng . Code ở APIController

5. **Repository và kết nối với cơ sở dữ liệu:**

* Repository chịu trách nhiệm kết nối với cơ sở dữ liệu thông qua Spring Data JPA.

|  |
| --- |
| Java |
| @Repository  public interface CategoryRepository extends JpaRepository<Category, Long> {  List<Category> findByType(CategoryType type);  } |

Bảng . Code Repository

6. **Xử lý truy vấn phức tạp với Specification**:

* Sử dụng **Specification** để xử lý truy vấn phức tạp hoặc dùng @Query để viết HQL.

|  |
| --- |
| Java |
| @Query("SELECT ms FROM Materialstock ms WHERE ms.materialId.id = ?1 ORDER BY ms.inventoryId.entryDate ASC")  List<Materialstock> findByIdOrderByInventoryEntryDateAsc(int id); |

Bảng . Code Specification

7. **Trả lại dữ liệu hoặc view**:

* Cuối cùng, sau khi xử lý các yêu cầu, **Controller** sẽ trả về một view (HTML với Thymeleaf) hoặc dữ liệu dưới dạng JSON.

### Thymeleaf:

Thymeleaf là một template engine mạnh mẽ và nó được thiết kế để làm việc mượt mà với HTML, CSS, JavaScript, và các công nghệ web khác. Nó đặc biệt hữu ích trong các ứng dụng Spring Boot do sự tích hợp mạnh mẽ và dễ dàng với framework này. Một trong những ưu điểm lớn của Thymeleaf là nó cho phép lập trình viên có thể xây dựng giao diện web với các mẫu HTML "tĩnh" mà vẫn tương tác được với dữ liệu từ backend.Cài đặt dependency cần thiết để sử dụng thymeleaf:

Dưới đây là một số đặc điểm nổi bật của Thymeleaf khi kết hợp với Spring Boot:

1. **Tích hợp hoàn hảo với Spring Boot**: Thymeleaf là lựa chọn lý tưởng cho Spring Boot nhờ vào việc nó có thể kết hợp trực tiếp với Spring MVC, giúp dễ dàng hiển thị dữ liệu lên giao diện người dùng từ các controller của Spring.
2. **Cú pháp đơn giản và dễ hiểu**: Thymeleaf sử dụng cú pháp gần với HTML gốc, làm cho mã dễ đọc và bảo trì, kể cả đối với những lập trình viên mới.
3. **Hỗ trợ quốc tế hóa (I18N)**: Với Thymeleaf, việc xử lý đa ngôn ngữ trên giao diện người dùng cũng trở nên dễ dàng thông qua việc tích hợp với hệ thống thông điệp (MessageSource) của Spring Boot.
4. **Phân tách rõ ràng giữa logic và hiển thị**: Thymeleaf giúp tách biệt rõ ràng giữa logic của ứng dụng và phần hiển thị giao diện người dùng, đảm bảo nguyên tắc "Separation of Concerns" trong lập trình.
5. **Tính năng Fragment (khối mã có thể tái sử dụng)**: Thymeleaf cho phép bạn định nghĩa các fragment HTML có thể tái sử dụng, giúp giảm thiểu việc lặp lại mã và dễ dàng quản lý các thành phần giao diện như header, footer, menu...
6. **Hỗ trợ Spring Security**: Thymeleaf có các thuộc tính đặc biệt để hỗ trợ kiểm tra quyền truy cập và hiển thị nội dung dựa trên quyền của người dùng, tương thích tốt với Spring Security.

Ví dụ tích hợp Thymeleaf với Spring Boot:

* **Thêm Thymeleaf vào project**: Khi tạo một ứng dụng Spring Boot, Thymeleaf có thể được thêm vào thông qua dependency trong pom.xml

|  |
| --- |
| xml |
| <dependency> \*xuanphat11112003  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency> |

Bảng . cập nhật thymeleaf ở pom.xml

Ở mỗi file HTML thực hiện khai báo như sau để sử dụng thư viện thymeleaf

<html xmlns:th="http://www.thymeleaf.org">

Ta có thể tạo một controller và sử dụng model.addAttribute để gửi dữ liệu thymeleaf.

|  |
| --- |
| Java |
| @Controller \*xuanphat11112003  public class DXPController {  @GetMapping("/greeting")  public String greeting(Model model) {  model.addAttribute("name", "Dang Xuan Phat");  return "index"; // Tên file HTML (index.html) trong thư mục templates}} |

Bảng . code controller truyền dữ liệu

Lặp qua qua danh sách đối tượng được truyền từ controller bằng thymeleaf.

|  |
| --- |
| html |
| <ul>  <li th:each="item : ${items}" th:text="${item}"></li>  </ul> |

Bảng . lặp danh sách dữ liệu với thymeleaf

Các thuộc tính của thymeleaf:

* **th:text**: Hiển thị văn bản từ biến.
* **th:href**: Gán giá trị liên kết cho thẻ <a>.
* **th:if**: Điều kiện hiển thị.
* **th:each**: Lặp qua danh sách các phần tử.
* **th:value**: Gán giá trị cho thẻ <input>.
* **th:replace**: chèn hay thay thế một nội dung của template

vd : (<head th:replace="layer/head :: head"></head>).

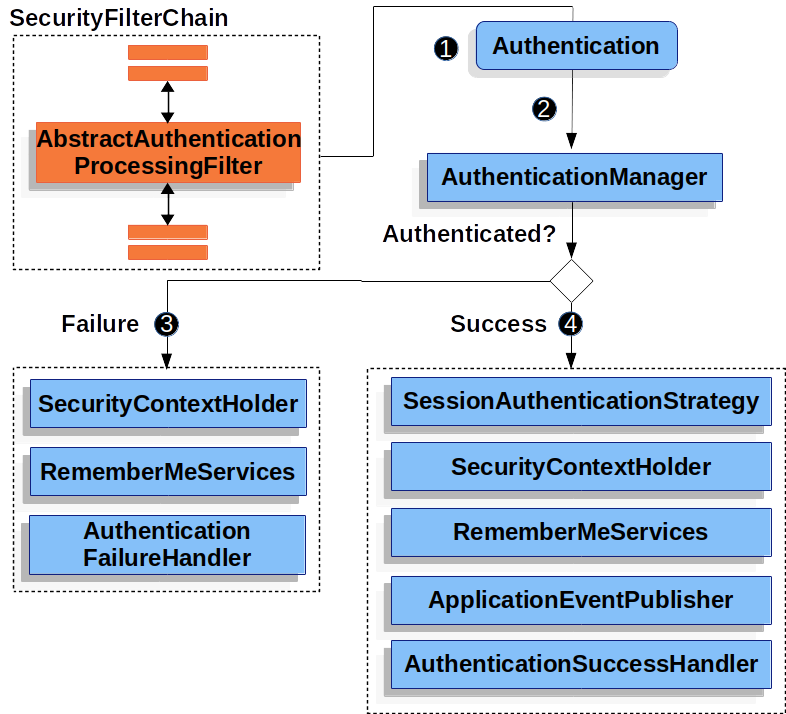
### Spring Security

Spring Security là một framework mạnh mẽ giúp bảo mật ứng dụng web, cung cấp các cơ chế xác thực, phân quyền và bảo vệ chống lại các cuộc tấn công phổ biến như CSRF, XSS. Nó dễ dàng tích hợp với Spring Boot và cho phép triển khai các giải pháp bảo mật toàn diện.

**Các thành phần chính trong Spring Security**:

* + - * **SecurityContextHolder:** Lưu trữ thông tin xác thực người dùng.
      * **Authentication:** Đại diện cho thông tin người dùng đã xác thực, có thể là thông tin đăng nhập hoặc người dùng đã xác thực.
      * **GrantedAuthority:** Các quyền của người dùng (chẳng hạn như vai trò).
      * **AuthenticationManager:** Quản lý quy trình xác thực người dùng.
      * **AuthenticationProvider:** Thực hiện các loại xác thực cụ thể (ví dụ: xác thực bằng username/password hoặc token JWT).
      * **AuthenticationEntryPoint:** Yêu cầu người dùng đăng nhập nếu chưa xác thực.
      * **AbstractAuthenticationProcessingFilter:** Cơ sở để xử lý các bộ lọc xác thực trong Spring Security.

Tiếp theo để hiểu rõ hơn về luông cũng như kiến trúc hoạt động của Spring Security ta sẽ tiếp tục tìm hiểu sâu hơn về AbstractAuthenticationProcessingFilter:



Hình . Sơ đồ chứng thực tài khoản

(nguồn: [*https://docs.spring.io/spring-security/reference/index.html*](https://docs.spring.io/spring-security/reference/index.html))

1. **Người dùng gửi thông tin đăng nhập**:

* Khi người dùng gửi thông tin đăng nhập (ví dụ: username và password), lớp AbstractAuthenticationProcessingFilter sẽ tạo ra một đối tượng Authentication từ HttpServletRequest để xác thực. Kiểu của Authentication được tạo ra phụ thuộc vào lớp con của AbstractAuthenticationProcessingFilter.
* Ví dụ: ở lớp UsernamePasswordAuthenticationFilter sẽ tạo một đối tượng UsernamePasswordAuthenticationToken chứa username và password được gửi từ HttpServletRequest.

2. **Authentication được chuyển vào AuthenticationManager để xác thực**:

* Sau khi đối tượng Authentication được tạo, nó sẽ được chuyển vào AuthenticationManager để tiến hành xác thực.

3. **Nếu xác thực thất bại**:

* SecurityContextHolder sẽ bị xóa (clear).
* RememberMeServices.loginFail sẽ được gọi. Nếu tính năng "remember me" không được cấu hình, bước này sẽ không có hiệu lực (no-op).
* AuthenticationFailureHandler sẽ được gọi để xử lý thất bại.

4. **Nếu xác thực thành công**:

* SessionAuthenticationStrategy sẽ được thông báo về việc người dùng đã đăng nhập thành công.
* Đối tượng Authentication sẽ được thiết lập vào SecurityContextHolder. Nếu cần lưu SecurityContext để tự động thiết lập trong các request tiếp theo, phương thức SecurityContextRepository#saveContext cần được gọi một cách rõ ràng
* RememberMeServices.loginSuccess sẽ được gọi. Nếu tính năng "remember me" không được cấu hình, bước này cũng sẽ không có hiệu lực.
* ApplicationEventPublisher sẽ phát ra sự kiện InteractiveAuthenticationSuccessEvent để thông báo rằng việc xác thực thành công đã diễn ra.
* Kích hoạt AuthenticationSuccessHandler để xử lý thành công.

**Cài đặt Spring Security với Spring Boot**:

* Các Dependency cần thiết để tạo 1 xác thực người dùng với spring security:

|  |
| --- |
| xml |
| <dependency> \*xuanphat11112003  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-security</artifactId>  </dependency> |

Bảng . Dependency cần thiết cho Spring Security

* Tạo 1 một entity class user:

|  |
| --- |
| Java |
| @Getter \*xuanphat11112003 @Setter @NoArgsConstructor @Entity public class User implements Serializable {  private String username;  private String password;  private String lastName;  private String firstName;  private String address;  private String useRole; }; |

Bảng . Entity class cho người dùng của hệ thống

* Tiếp tục tạo 1 interface UserService extends UserDetailsService:

|  |
| --- |
| java |
| @Override \*xuanphat11112003 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  User u = this.userRepo.findByUsername(username);  if(u == null){  throw new UsernameNotFoundException("User Not Found");  }  Set<GrantedAuthority> authorities = new HashSet<>();  authorities.add(new SimpleGrantedAuthority(u.getUserRole().toString()));  return new org.springframework.security.core.userdetails.User(u.getUsername(),u.getPassword(),authorities); |

Bảng . ghi đè lớp lại lớp cha loadUserByUserName

* Sau đó tiếp tục cấu hình với config như ở kiến trúc trước đó ta sẽ dùng đối tượng SecurityFilterChain để cấu hình Security cho hệ thống:

|  |
| --- |
| Java |
| @Configuration  @EnableWebSecurity  public class SecurityConfig {  @Bean  public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  http.authorizeRequests().antMatchers("/public/\*\*").permitAll()  .anyRequest().authenticated()  .and().formLogin().loginPage("/login").permitAll()  .and().logout().permitAll();  return http.build();  }  @Bean  public PasswordEncoder passwordEncoder() {  return new BCryptPasswordEncoder();  }  @Bean  public InMemoryUserDetailsManager userDetailsService() {  UserDetails user = User.withUsername("user").password(passwordEncoder().encode("password")).roles("USER").build();  UserDetails admin = User.withUsername("admin").password(passwordEncoder().encode("admin")).roles("ADMIN").build();  return new InMemoryUserDetailsManager(user, admin);  }  } |

Bảng . Cấu hình Spring Security

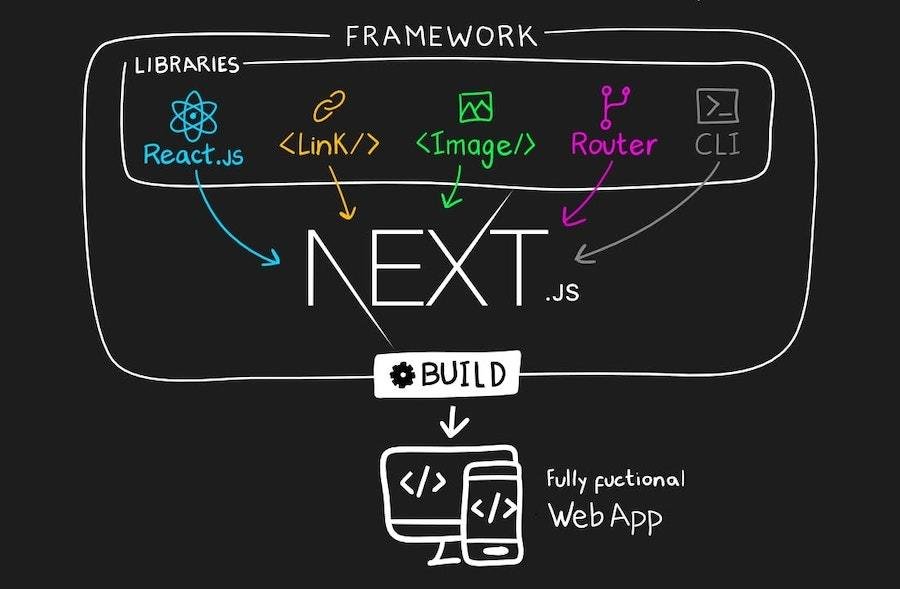
Như vậy Spring Security cũng là phần cuối đối với phần trình bày về Spring Boot thông qua chương này ta có thể hiểu được về Spring Boot là gì cũng như các tiện ích mà FrameWork mang lại cho dự án và hiểu được các cách hoạt động của Spring FrameWork để có thể dễ dàng bắt đầu tạo 1 dự án.

## FrontEnd Framework-NextJS

### Giới thiệu về NextJS

NextJS là một bước tiến vượt trội từ mô hình trang web tĩnh truyền thống, nơi mà việc chuyển trang chậm và tương tác kém gây ra sự cản trở trải nghiệm người dùng và khi React.js ra đời với mô hình (SPA) Single Page Application đã khắc phục được nhược điểm của trang web tĩnh nhưng lại gặp 1 thách thức về tính SEO. NextJS Phát triển bởi Vercel và ra đời như một giải pháp tối ưu, kết hợp những của ưu điểm của SPA và khả năng hiển thị tối ưu trên các công cụ tìm kiếm, giúp ứng dụng web vừa nhanh và mượt mà.

Ngoài ra NextJs không chỉ là Framework cho giao diện mà NextJS còn là cầu nối giữa giao diện và hệ thống (UI/UX) trong phát triển hệ thống web hiện đại, là một Framework lý tưởng giúp các lập trình viên xây dựng ứng dụng full-stack trở nên dễ dàng.



Hình . Giới thiệu về NextJS

(nguồn: <https://slides.com/rodrigoarriaran/next-js/fullscreen>)

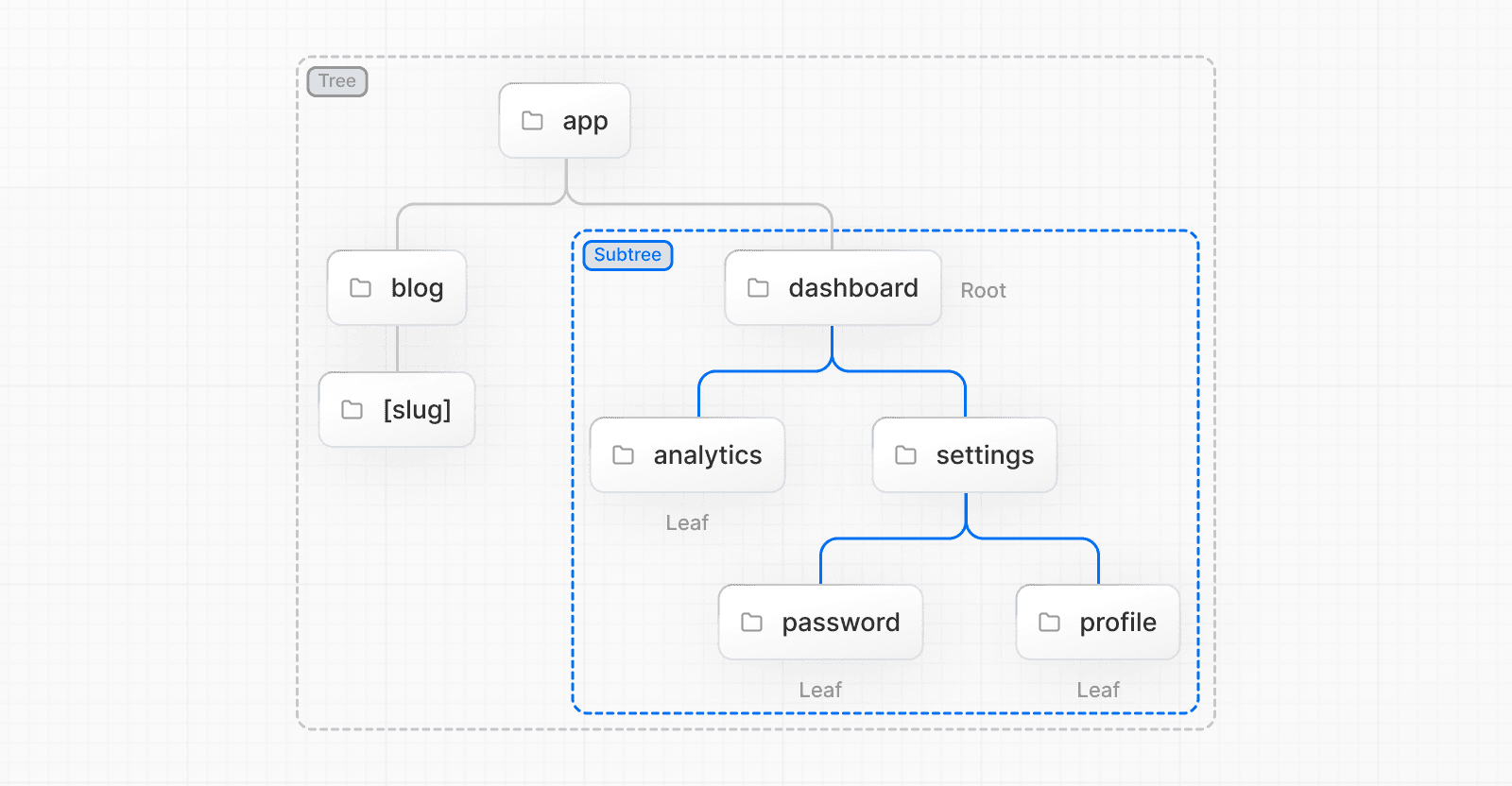
### Tại sao nên sử dụng NextJS

NextJS là một Framework hỗ trợ mạnh mẽ cho giao diện cùng với nhiều tính năng vượt trội dưới đây ta hãy cùng tìm hiểu qua về các tính năng của NextJS

#### Routing trong NextJS

* **Automatic Routing**: NextJS sẽ tự động tạo các router cho chúng ta, ta chỉ cần đặt tên thư mục hoặc các file theo đúng cấu trúc của NextJs

Ví dụ: ta sẽ có 1 đường dẫn http:localhost:3000/about và ta chỉ cần tạo theo đúng cấu trúc của NextJS là page/about.tsx thì NextJS sẽ tự động tạo các router phù hợp với đường dẫn mong muốn.



Hình . Tạo thư mục theo cấu trúc

(nguồn: <https://nextjs.org/docs/app/building-your-application/routing>)

* **Nested Routing**: sẽ giúp ta trong việc tạo các thư mục con để các router lồng nhau, nếu ta muốn với một đường dẫn /employee cho tất cả các trang dùng cho người dùng đăng nhập với quyền là nhân viên thì ta chỉ cần tạo 1 một folder employee và folder với tên đường dẫn thì sẽ có được một router truy cập mong muốn với tên file là page thì đường dẫn sẽ mặc định là employee/.
* **Dynamic Routes**: nếu ta muốn tạo ra các router động bằng cách sử dụng cặp dấu [] trong tên file tao chỉ cần xây dựng thư mục theo cấu trúc pages/employee/[slug].tsx thì NextJS sẽ tạo ra các router có đường dẫn /employee/employee-1.tsx hoặc /employee/employee-2.tsx với slug là giá trị bất kỳ mà ta truyền vào.
* **Link Component**: NextJS, việc chuyển trang nội bộ (internal navigation) có thể thực hiện dễ dàng và hiệu quả với component <Link />. Thay vì tải lại toàn bộ trang như cách truyền thống, <Link /> cho phép điều hướng giữa các trang với tốc độ nhanh hơn nhờ tận dụng tính năng client-side routing. Để sử dụng, ta chỉ cần import Link từ 'next/link' và truyền thuộc tính href với đường dẫn mong muốn.

Ví dụ:

|  |
| --- |
| Typescripts |
| import Link from 'next/link';  function HomePage() {  return (  <div>  <Link href="/employee">Trang nhân viên</Link>  </div>  );  } |

Bảng . cách sử dụng thẻ link

* **Query Parameter**: Để lấy dữ liệu động từ URL, dùng useRouter từ next/router và truy cập router.query. Ví dụ: với URL /employee?department=sales, ta có thể lấy giá trị department để sử dụng trực tiếp trong trang.

Ví dụ, để lấy query parameter từ URL /employee?department=sales:

|  |
| --- |
| Typescript |
| import { useRouter } from 'next/router';  function EmployeePage() {  const router = useRouter();  const { department } = router.query;  return ( <div> <h1> Phòng ban: {department} </h1> </div>);} |

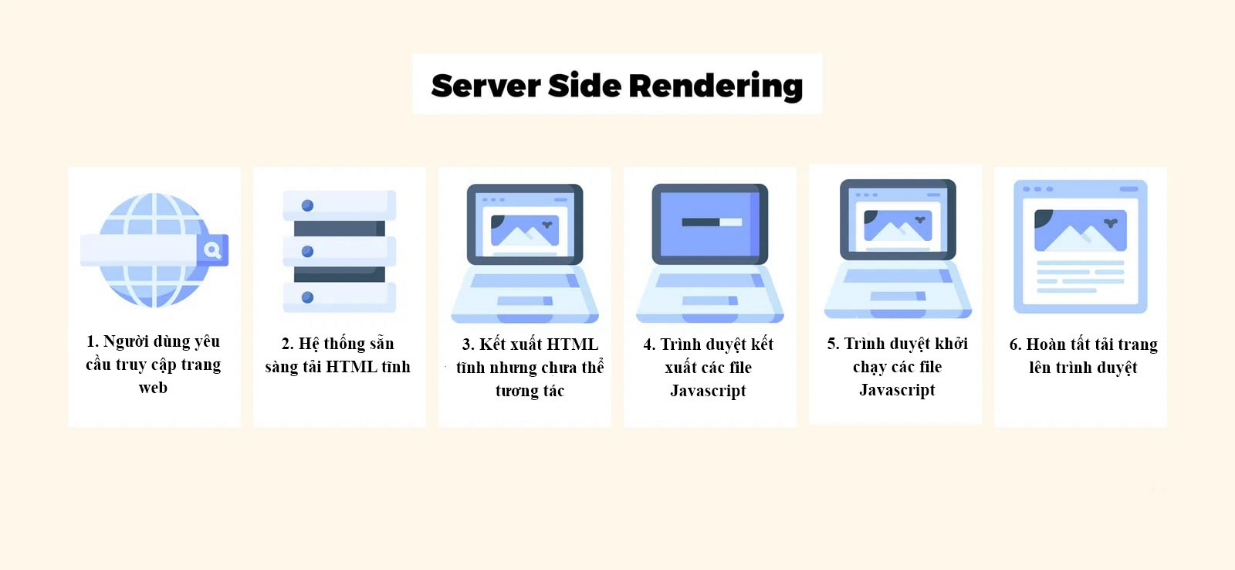
Bảng . code sử dụng router

#### Rendering trong NextJS

Server-side Rendering (SSR)

**Cơ chế hoạt động của SSR:**

* Khi người dùng gửi yêu cầu hiển thị trang web, trình duyệt sẽ gửi yêu cầu này tới server.
* Ở server sẽ nhận yêu cầu này và truy cập vào dữ liệu, render ra HTML.
* Trả HTML CSS JS về cho trình duyệt.
* Trình duyệt nhận được HTML thì sẽ bắt đầu tải xuống và render ra giao diện cho người dùng nhưng lúc này chưa thể tương tác vì chưa có JavaScripts.
* Tiếp tục trình duyệt sẽ tải Javascript của trang xuống và lúc này người dùng có thể tương tác được với trang bình thường.



Hình . Quy trình tải trang với Server Side Rendering

**Ưu điểm của Server-Side Rendering (SSR):**

* Tải trang lần đầu sẽ rất nhanh bởi vì toàn bộ dữ liệu được xử lý ở server, client chỉ cần nhận về và hiện thị cho người dùng.
* Tốt cho SEO vì dữ liệu được tải ở dạng HTML ta chỉ cần một cú click chuột vào bất kỳ vào một page nào đó sẽ thấy được điều này, giúp cho bot của google khi quét sẽ thấy được toàn bộ dữ liệu.
* Dễ dàng tích hợp cho các lập trình viên muốn lập trình trên 1 project mà vẫn có thể tạo ra cả Backend và Frontend.

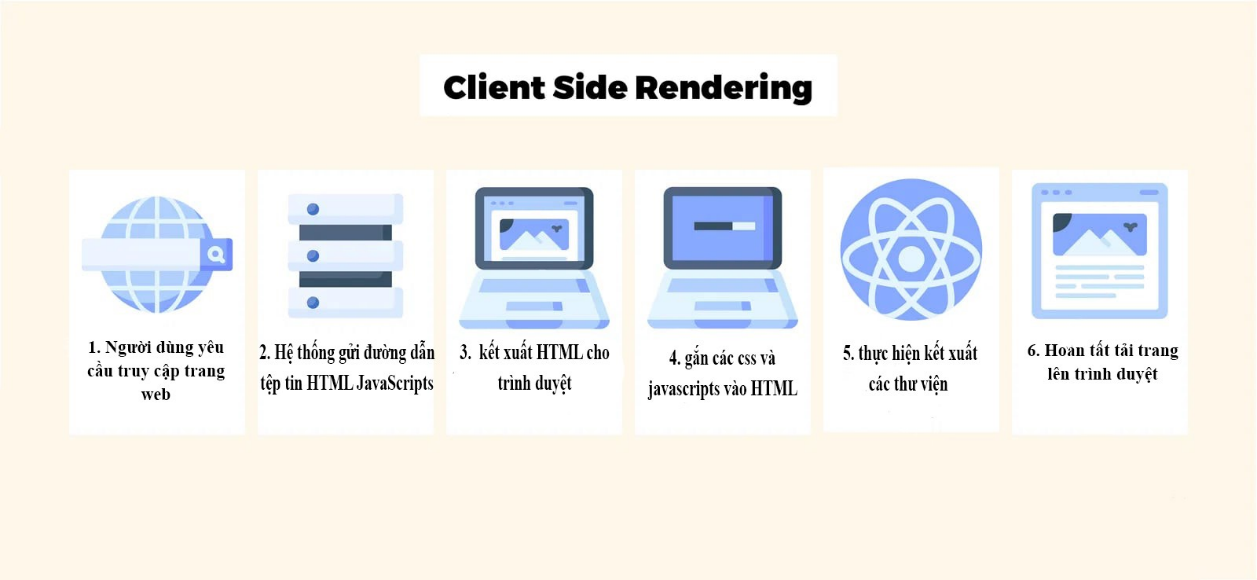
**Nhược điểm của Server-Side Rendering (SSR):**

* Server sẽ phải xử lý dữ liệu rất nhiều có thể dẫn đến quá tải, khi người dùng chuyển trang thì cả trang đều phải load lại mang lại tính trải nghiệm không tốt đối với người dùng.

Client-side Rendering (CSR)

**Cơ chế hoạt động của CSR:**

* Khi người dùng gửi yêu cầu truy cập vào trang web, trình duyệt sẽ gửi yêu cầu đến server.
* Server trả về một HTML tĩnh cơ bản (thường là một trang chứa JavaScript) mà không chứa dữ liệu trang.
* Trình duyệt tải xuống và thực thi JavaScript của trang.
* Javascript sẽ gửi yêu cầu đến server để lấy dữ liệu cần thiết (thông qua API hoặc các phương thức khác).
* Khi nhận được dữ liệu, Javascript sẽ render nội dung vào DOM, tạo ra giao diện người dùng hoàn chỉnh.
* Người dùng có thể tương tác với trang ngay khi JavaScript được tải xuống và chạy, mà không cần phải tải lại trang.



Hình .: Quy trình tải trang với Client Side Rendering

**Ưu điểm của Client-Side Rendering (CSR):**

* Chuyển dữ liệu cho client xử lý giúp giảm nhẹ việc hơn cho server nên vì vậy HTML render ở client hay có thể hiểu là Single Page Application (SPA).
* Trang chỉ load một lần duy nhất, khi người dùng muốn lấy dữ liệu mới từ server chỉ gọi đến server thông qua Ajax.
* Nên vì vậy trang web không phải lại nhiều khi người dùng chuyển trang giúp tăng được tính tương tác cho người dùng.

**Nhược điểm của Client-Side Rendering (CSR):**

* Tính SEO sẽ bị hạn chế bởi vì việc load trang lần sẽ chậm hơn so với SSR, Do trình duyệt phải tải toàn bộ HTML và Javascripts ở lần đầu tải trang
* Và lập trình viên phải tạo ra 2 dự án client và server riêng biệt.
* Trang web sẽ không thể chạy nếu tắt Javasript ở trình duyệt.
* Trang vẫn sẽ tải chậm nếu người dùng sử dụng thiết bị có cấu hình yếu.

Chính vì những nhược điểm của 2 cơ chế hoạt động mà NextJS ra đời, NextJS là sự kết hợp hoàn hảo của SSR và CSR để tạo nên trang web có trải nghiệm tuyệt vời nhưng vẫn đảm bảo được tính SEO cho trang web.

Ngoài ra NextJS còn hỗ trợ các tính năng giúp tối ưu hóa hiệu suất code như Code Splitting, Lazy Loading, Image Optimazon, … Tự động làm giao diện mới mà không cần load lại trang với cơ chế Fast Refresh, Có thể tự tạo file CSS cho mỗi trang nhằm giúp tránh xung đột, Hỗ trợ TypeScripts giúp code dễ nhìn và tăng tính rõ ràng hơn thuận tiện cho việc debug về sau.

NextJS sở hữu cho mình một cộng đồng người sử dụng đông đảo điêu này giúp việc tìm hiểu NextJS cho người mới bắt đầu sẽ trở nên dễ dàng vì có rất nhiều nguồn tài liệu phong phú và hữu ích, Nếu đã quen với ReactJS trước đó thì việc tìm hiểu về NextJS sẽ rất đơn giản vì NextJS được tích hợp tốt với ReactJS kết hợp tốt với hệ sinh thái của React.

### Cách cài đặt 1 dự án với Framework NextJS

Bước 1: chuẩn bị môi trường

Đầu tiên chúng ta có thể truy cập vào trang chính của NextJs.org để xem NextJS hiện tại đang hỗ trợ phiên Node.js bao nhiêu để thuận tiện cài đặt tránh trường hợp không khớp môi trường cài đặt.

Bước 2: tạo một dự mới với NextJS

Để tạo dự án ta mở terminal cd tới đường dẫn muốn lưu tập và chạy lệnh sau:

|  |
| --- |
| Terminal |
| npx create-next-app@latest |

Bảng . Câu lệnh bắt đầu cài đặt dự án NextJS

Sau khi chạy câu lệnh thành công ta sẽ thấy tiếp tục những dòng lệnh sau:

|  |
| --- |
| Terminal |
| **What is your project named? my-app**  Would you like to use TypeScript? No / Yes ----tùy chọn----  Would you like to use ESLint? No / Yes ----tùy chọn----  Would you like to use Tailwind CSS? No / Yes ----tùy chọn----  Would you like your code inside a `src/` directory? No / Yes ----tùy chọn----  Would you like to use App Router? (recommended) No / Yes ----tùy chọn----  Would you like to use Turbopack for `next dev`? No / Yes ----tùy chọn----  Would you like to customize the import alias (`@/\*` by default)? No / Yes  What import alias would you like configured? @/\* ----tùy chọn---- |

Bảng . các lựa chọn để tạo cấu hình

NextJS đưa ra những lựa chọn giúp người dùng nhanh chóng cài đặt chữ thư viện hỗ trợ cho dự án.

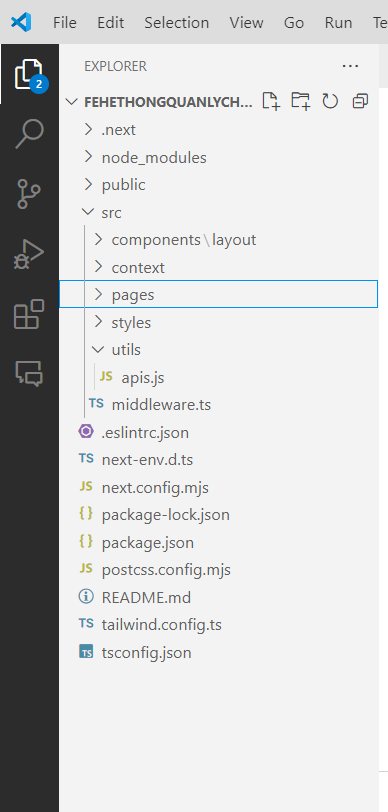
Sau khi tạo dự án bắt đầu mở dự án và chạy câu lệnh:

|  |
| --- |
| Terminal |
| npm run dev |

Bảng . Câu lệnh để khởi lại dự án

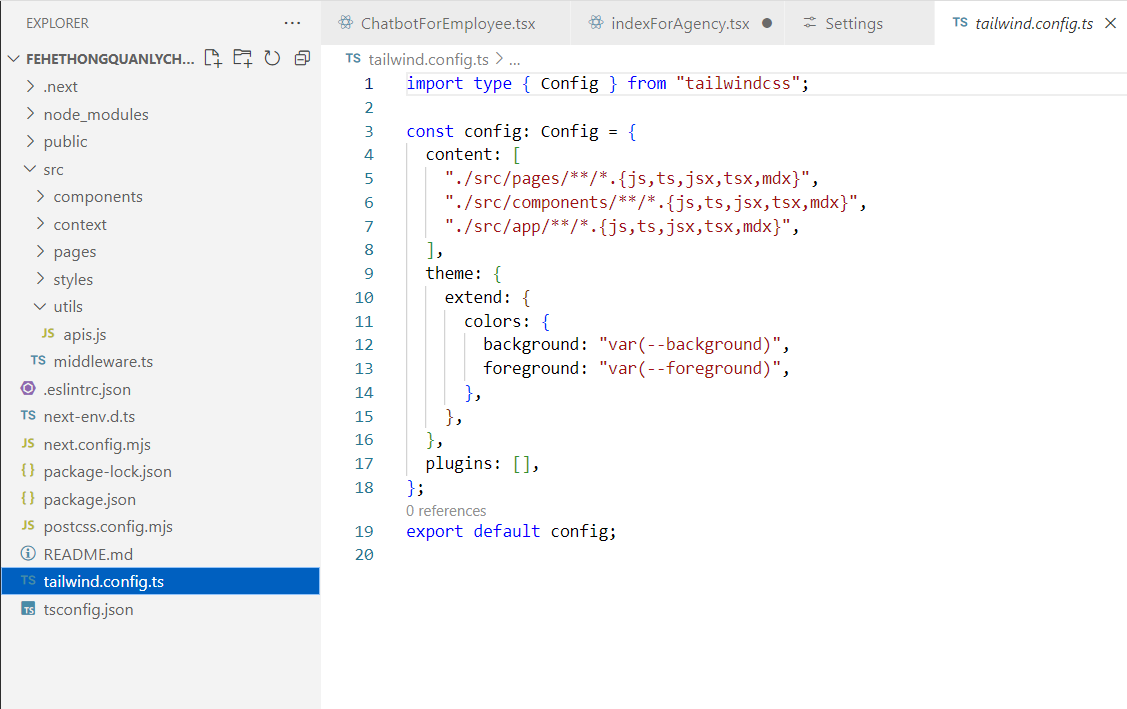
Sau khi chạy thành công, ta sẽ thấy xuất hiện đường dẫn ở trên terminal, thường sẽ là http://localhost:3000. ta truy cập vào đường dẫn này để thấy dự án của mình chạy trên trình duyệt.

Cấu trúc dự án khi mới tạo:



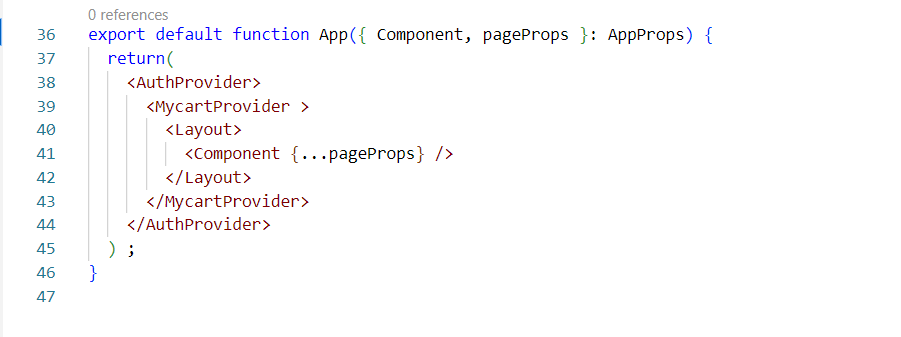
Hình . Cấu trúc dự án khi bắt đầu vào

* Để tạo cái router ta sẽ tiến hành tạo trong thư mục pages tạo một file hoặc folder để NextJS sẽ tự động tạo đường dẫn (ví dụ: src/pages/agency/index.tsx -> đường dẫn truy cập localhost:3000/agency ta truy cập vào đường dẫn này sẽ thấy dự án của mình phát triển)



Hình . File cấu hình tailwind.config.ts

* Tailwind hỗ trợ mạnh mẽ sẽ hỗ trợ mạnh mẽ cho dự án, với file cấu hình này sẽ tạo sẵn đường dẫn được hỗ trợ tailwind nếu ta tạo những file để phát triển ở ngoài đường dẫn được cấu hình mặc định ta chỉ cần thêm đường dẫn vào mảng content.



Hình . File \_app.TS

* Đây là nơi mà ta có thể tái tạo lên các Header Footer dùng cho tất cả các component.

NextJS là một công cụ mạnh mẽ là một Framework xây dựng dựa trên ReactJS được thiết kế tập trung để cải thiện hiệu suất về SEO cho dự án, được tích hợp sẵn hệ thống routing phù hợp cho phần Frontend của dự án hỗ trợ được tính seo cho trang web và đảm bảo được tính trải nghiệm cao với người dùng.

## Các công cụ liên quan khác

### Redis và Docker

**Redis là gì?**

Redis, viết tắt của REmote DIctionary Server, là một hệ thống lưu trữ dữ liệu kiểu key-value, hoạt động chủ yếu trong bộ nhớ (in-memory) và mã nguồn mở. Nó cho phép lưu trữ dữ liệu dưới nhiều cấu trúc khác nhau như chuỗi, danh sách, tập hợp, và hash. Một trong những điểm mạnh của Redis là khả năng truy xuất dữ liệu rất nhanh, nhờ vào việc dữ liệu được lưu trữ trực tiếp trong RAM thay vì trên đĩa cứng.

**Tại sao nên sử dụng Redis?**

Tốc độ nhanh: Redis cho phép truy xuất dữ liệu với thời gian rất ngắn, thường dưới 1 ms, nhờ vào việc lưu trữ thông tin trong bộ nhớ.

Đa dạng kiểu dữ liệu: Ngoài việc hỗ trợ key-value cơ bản, Redis còn cho phép sử dụng nhiều kiểu dữ liệu phức tạp như danh sách, tập hợp, và hash, giúp quản lý và truy xuất dữ liệu dễ dàng hơn.

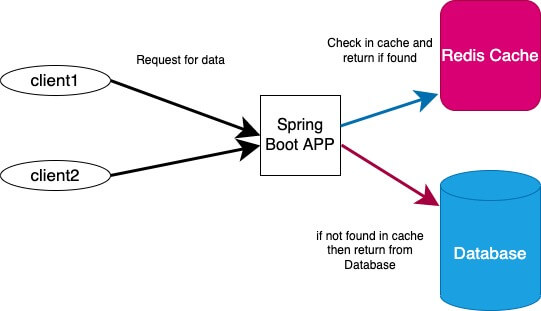
Khả năng mở rộng linh hoạt: Redis có tính năng clustering và replication, giúp người dùng dễ dàng mở rộng quy mô và tăng cường tính sẵn sàng của dữ liệu.

Chức năng caching hiệu quả: Redis thường được sử dụng như một bộ nhớ cache, giúp giảm tải cho các cơ sở dữ liệu chính và nâng cao hiệu suất của ứng dụng.

Hỗ trợ giao tiếp thời gian thực: Redis cung cấp cơ chế publish/subscribe, cho phép các ứng dụng trao đổi thông tin một cách nhanh chóng và hiệu quả.

Tính bền vững của dữ liệu: Mặc dù hoạt động chủ yếu trong bộ nhớ, Redis cũng có các phương thức để lưu trữ dữ liệu trên đĩa, đảm bảo rằng thông tin không bị mất đi.

**Redis hoạt động như thế nào?**



Dữ liệu có lưu ở cache

Yêu cầu dữ liệu

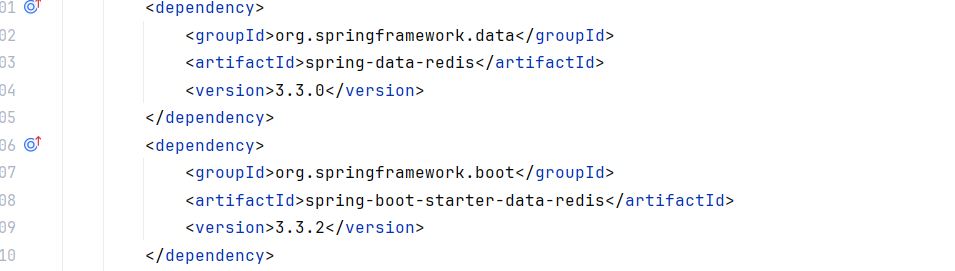
Dữ liệu không lưu ở cahche

Hình . Luồng lấy dữ liệu khi có redis

Khi một yêu cầu truy xuất dữ liệu được gửi đến ứng dụng Spring Boot, ứng dụng này sẽ ưu tiên kiểm tra xem dữ liệu đó đã có sẵn trong bộ nhớ đệm Redis chưa. Nếu dữ liệu đã được lưu trữ trong Redis, ứng dụng sẽ nhanh chóng trả về kết quả cho người dùng mà không cần truy cập vào cơ sở dữ liệu chính. Quá trình này diễn ra cực kỳ nhanh chóng do đặc tính truy xuất dữ liệu trong bộ nhớ của Redis.

Ngược lại, nếu dữ liệu chưa có trong Redis, ứng dụng sẽ tiến hành truy vấn cơ sở dữ liệu để lấy dữ liệu cần thiết. Sau đó, dữ liệu này sẽ được lưu vào Redis để phục vụ cho các yêu cầu truy xuất tiếp theo. Việc lưu trữ dữ liệu vào Redis giúp giảm thiểu thời gian truy cập cơ sở dữ liệu, đặc biệt là đối với những dữ liệu được truy xuất thường xuyên.

Các dependency cần thiết để cài redis cho spring boot:



Hình . Dependency cần thiết cho redis

Tiếp tục là tạo server cho redis:

Docker là một trong những server ảo dễ dàng cài đặt và tích hợp nhất

Tiến hành chạy 2 câu lệnh sau khi đã cài đặt docker thành công về máy có thể chạy trên cmd hoặc docker desktop.

|  |
| --- |
| bash |
| docker pull redis  docker run --name my-redis -d -p 6379:6379 redis |

Bảng . Câu lệnh cài đặt redis trên server

Cấu hình redis trong spring boot:



Hình . Cấu hình redis

Trong hệ thống quản lý chuỗi cung ứng, Redis đóng vai trò là một lớp bộ nhớ đệm hiệu quả, giúp tăng tốc độ truy xuất dữ liệu, giảm tải cho cơ sở dữ liệu và cải thiện trải nghiệm người dùng. Việc tích hợp Redis vào hệ thống là một giải pháp tối ưu để nâng cao hiệu suất và đáp ứng nhu cầu ngày càng cao của các doanh nghiệp trong việc quản lý chuỗi cung ứng một cách hiệu quả.

### PyTorch - Dự đoán doanh thu

**Pytorch là gì?**

PyTorch là một thư viện học sâu (deep learning) mã nguồn mở được phát triển bởi Facebook. Nó cung cấp các công cụ để xây dựng và huấn luyện các mô hình học máy (machine learning) và học sâu (deep learning), đặc biệt là trong các bài toán phân loại, hồi quy, nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, và nhiều ứng dụng khác. PyTorch hỗ trợ tính toán trên GPU, cho phép thực hiện các phép toán nhanh chóng và hiệu quả.

**Tại sao lại dùng PyTorch?**

PyTorch được chọn trong hệ thống của bạn vì khả năng linh hoạt và mạnh mẽ trong việc xây dựng các mô hình học sâu. Mặc dù PyTorch không chỉ được thiết kế để dự đoán doanh thu, nhưng nó rất phù hợp cho các bài toán hồi quy (regression), trong đó dự đoán giá trị liên tục như doanh thu là một ứng dụng điển hình. PyTorch cung cấp các công cụ đơn giản để huấn luyện các mô hình dự báo dựa trên dữ liệu lịch sử, giúp doanh nghiệp dự đoán doanh thu trong tương lai. Với khả năng tối ưu hóa và hỗ trợ tính toán trên GPU, PyTorch giúp giảm thiểu thời gian huấn luyện cho các mô hình phức tạp.

**Cách sử dụng Pytorch?**

***Cài đặt Pytorch:***

|  |
| --- |
| *CMD* |
| pip install torch torchvision |

Bảng . câu lệnh cài đặt Pytorch

***Tải dữ liệu:***

Trước khi xây dựng mô hình, bạn cần chuẩn bị dữ liệu cho việc huấn luyện. Giả sử bạn có một tập dữ liệu lịch sử doanh thu, mỗi mẫu có thể bao gồm các thông số như thời gian (tháng, quý), chi phí quảng cáo, số lượng sản phẩm bán ra, v.v.

|  |
| --- |
| python |
| import torch  from torch.utils.data import Dataset, DataLoader  import pandas as pd  data = pd.read\_csv('doanh\_thu.csv')  class RevenueDataset(Dataset):  def \_\_init\_\_(self, dataframe):  self.data = dataframe  self.features = torch.tensor(dataframe[['chi\_phi\_quang\_cao', 'so\_luong\_ban']].values, dtype=torch.float32)  self.targets = torch.tensor(dataframe['doanh\_thu'].values, dtype=torch.float32)  def \_\_len\_\_(self):  return len(self.data)  def \_\_getitem\_\_(self, idx):  return self.features[idx], self.targets[idx]  dataset = RevenueDataset(data)  dataloader = DataLoader(dataset, batch\_size=32, shuffle=True) |

Bảng . Code thu thập dữ liệu

***Xây dựng mô hình hồi quy với Pytorch***

Bây giờ bạn sẽ xây dựng một mô hình hồi quy để dự đoán doanh thu. Mô hình này có thể bao gồm một số lớp mạng nơ-ron (neural network) cơ bản.

|  |
| --- |
| python |
| import torch.nn as nn  import torch.optim as optim  class RevenueModel(nn.Module):  def \_\_init\_\_(self):  super(RevenueModel, self).\_\_init\_\_()  # Mạng nơ-ron với 2 lớp ẩn  self.fc1 = nn.Linear(2, 64)  self.fc2 = nn.Linear(64, 32)  self.fc3 = nn.Linear(32, 1)  def forward(self, x):  x = torch.relu(self.fc1(x))  x = torch.relu(self.fc2(x))  x = self.fc3(x)  return x  model = RevenueModel() |

Bảng . Code xây dựng mô hình hồi quy

***Định nghĩa hàm mất mát và Bộ tối ưu***

|  |
| --- |
| python |
| #xác định hàm mất mát và bộ tối ưu hóa optmizer  criterion = nn.MSELoss()  optimizer = optim.Adam(model.parameters(), lr=0.001) |

Bảng . Code hàm mất mát và tối ưu với optmizer

***Huấn luyện mô hình:*** Trong mỗi vòng lặp (epoch), bạn sẽ tính toán mất mát, thực hiện quá trình lan truyền ngược (backpropagation) và cập nhật các trọng số của mô hình.

|  |
| --- |
| python |
| epochs = 100  for epoch in range(epochs):  for features, targets in dataloader:  optimizer.zero\_grad() # Đặt lại gradient trước mỗi bước tối ưu hóa  outputs = model(features)  loss = criterion(outputs, targets.view(-1, 1))  loss.backward()  optimizer.step()  if (epoch+1) % 10 == 0:  print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}') |

Bảng . Code huấn luyện mô hình

***Dự đoán***

* + Sau khi huấn luyện xong, ta có thể sử dụng mô hình đã huấn luyện để dự đoán doanh thu cho các dữ liệu mới:

|  |
| --- |
| python |
| with torch.no\_grad():  sample\_input = torch.tensor([[50000, 1500]], dtype=torch.float32  predicted\_revenue = model(sample\_input)  print(f'Dự đoán doanh thu: {predicted\_revenue.item()}') |

Bảng . Code dự đoán mô hình

***Mô tả tổng quan về thuật toán sử dụng***

Trong bài toán dự báo doanh số, dữ liệu đầu vào là một chuỗi thời gian (sales\_history) thể hiện doanh số các kỳ trước đó. Mục tiêu là dự đoán giá trị doanh số cho kỳ tiếp theo (next\_sales).  
Để giải quyết bài toán này, nhóm nghiên cứu đã sử dụng mô hình học sâu kết hợp giữa **mạng nơ-ron hồi tiếp (LSTM)** và **cơ chế chú ý (Attention Mechanism)**, được thiết kế riêng nhằm tối ưu hóa khả năng khai thác thông tin theo thứ tự thời gian.

***Cấu trúc thuật toán***

**LSTM (Long Short-Term Memory)LSTM** là một dạng mạng hồi tiếp cải tiến, cho phép mô hình ghi nhớ thông tin dài hạn trong chuỗi thời gian mà không gặp phải hiện tượng mất mát thông tin (vanishing gradient) như các RNN truyền thống.  
LSTM được trang bị ba cổng điều khiển:

* **Forget gate**: quyết định những thông tin nào cần loại bỏ.
* **Input gate**: xác định thông tin mới nào nên thêm vào bộ nhớ.
* **Output gate**: chọn lọc phần bộ nhớ nào được xuất ra làm kết quả bước thời gian hiện tại.

Trong mô hình triển khai, một mạng **LSTM nhiều lớp (5 layers)** với **512 nút ẩn (hidden units)** được sử dụng để học các xu hướng phức tạp từ dữ liệu doanh số.

***Attention Mechanism***

Mặc dù LSTM đã giúp lưu giữ thông tin, nhưng khi chuỗi dữ liệu dài hoặc có nhiều yếu tố tác động, mô hình có thể bị "loãng" thông tin.  
Để khắc phục, nhóm nghiên cứu tích hợp thêm một **cơ chế Attention** ngay sau LSTM, nhằm:

* Tính toán **mức độ quan trọng** của từng bước thời gian.
* Tạo ra một **vector ngữ cảnh (context vector)** gói gọn các thông tin có giá trị cao nhất cho việc dự đoán.

Điều này giúp mô hình "tập trung" vào những kỳ kinh doanh đặc biệt (ví dụ: lễ Tết, đợt sale lớn) thay vì đối xử mọi thời điểm như nhau.

### Rasa – ChatBot

**Rasa là gì?**

Rasa là một nền tảng mã nguồn mở để phát triển chatbot AI và các hệ thống xử lý ngôn ngữ tự nhiên (NLP). Rasa giúp xây dựng chatbot có khả năng hiểu và phản hồi các câu hỏi của người dùng một cách tự động. Nó cho phép lập trình viên xây dựng các chatbot có khả năng hiểu ngữ cảnh, nhận diện ý định (intents) và thực hiện các hành động (actions) phù hợp dựa trên các cuộc trò chuyện với người dùng. Rasa hỗ trợ cả Rasa NLU (Natural Language Understanding) để xử lý ngôn ngữ và Rasa Core để quản lý cuộc hội thoại.

**Tại sao lại dùng Rasa?**  
Rasa được lựa chọn vì tính năng mở rộng và khả năng tùy chỉnh cao, cho phép xây dựng các chatbot phức tạp với khả năng hiểu ngữ cảnh của người dùng. Điều này giúp chatbot không chỉ trả lời các câu hỏi cơ bản mà còn có thể tham gia vào các cuộc trò chuyện dài và phức tạp, tạo ra một trải nghiệm người dùng tự nhiên hơn. Rasa cũng dễ dàng tích hợp với các hệ thống hiện có và có thể hoạt động offline, giúp đảm bảo bảo mật và tính linh hoạt khi triển khai trong môi trường doanh nghiệp.

**Cách sử dụng Rasa?**

***Cài đặt Rasa:***

|  |
| --- |
| cmd |
| pip install rasa |

Bảng . Câu lệnh tải framework rasa

**Khởi tạo dự án mới**:

* + Sau khi cài đặt xong, ta sẽ cần tạo một dự án mới. Rasa cung cấp lệnh rasa init để tạo một dự án mẫu, bao gồm các tệp cấu hình cơ bản cần thiết để xây dựng và huấn luyện một chatbot.

|  |
| --- |
| cmd |
| Would you like to create a new project? [Y/n] |

Bảng . Xác nhận tạo dự án mới

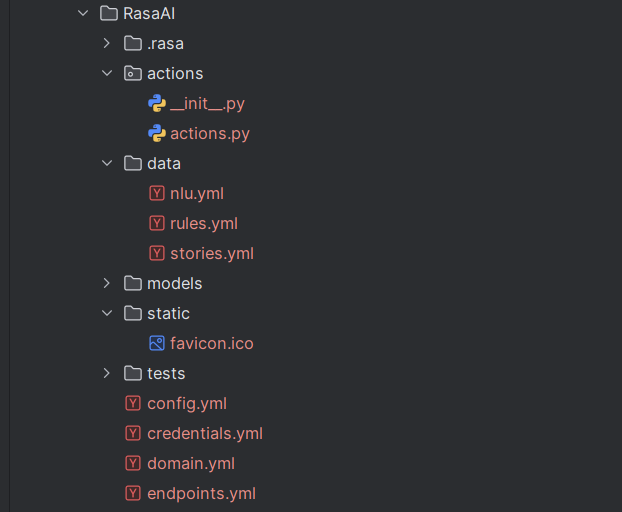
* + Chọn **Y** để tạo dự án mới và tiếp tục với cấu hình mặc định.
  + Sau khi tạo dự án mới, Rasa sẽ hỏi có muốn huấn luyện một mô hình khởi tạo hay không. Mô hình này sẽ được sử dụng để kiểm tra nhanh hệ thống và giúp ta bắt đầu xây dựng chatbot ngay lập tức.

|  |
| --- |
| cmd |
| Do you want to train an initial model? [Y/n] |

Bảng . Xác nhận train AI lập tức

* + Chọn **Y** để huấn luyện mô hình và tạo các tệp cấu hình cho dự án.

***Cấu trúc mô hình sao khi tạo thành công:***

******

Hình . Cấu trúc Rasa

***Cấu hình rasa cơ bản:***

**File domain.yml**: Đây là nơi ta định nghĩa các intents, entities, actions, và responses mà bot của bạn sẽ sử dụng.

|  |
| --- |
| yaml |
| version: "2.0"  intents:  - greet  - goodbye  responses:  utter\_greet:  - text: "Chào bạn!"  utter\_goodbye:  - text: "Tạm biệt!" |

Bảng . code đơn giản ở file domain.yml

**File data/nlu.yml**: Đây là nơi ta định nghĩa các ví dụ cho các intents (dữ liệu huấn luyện cho NLU).

|  |
| --- |
| yaml |
| version: "2.0"  nlu:  - intent: greet  examples: |  - Chào  - Xin chào  - Hello |

Bảng . code đơn giản ở file nlu.yml

**File data/rules.yml:** Đây là nơi ta định nghĩa các quy tắc cho hành vi của bot.

|  |
| --- |
| yaml |
| version: "2.0"  rules:  - rule: Greet when the user says hello  steps:  - intent: greet  - action: utter\_greet  - rule: Say goodbye when the user says goodbye  steps:  - intent: goodbye  - action: utter\_goodbye |

Bảng . code đơn giản ở file rule.yml

**File config.yml**: Đây là nơi ta định nghĩa cấu hình cho mô hình học máy (các pipeline NLU và policies).

|  |
| --- |
| yaml |
| version: "2.0"  language: "vi"  pipeline:  - name: WhitespaceTokenizer  - name: RegexEntityExtractor  - name: CountVectorsFeaturizer  - name: DIETClassifier  - name: EntitySynonymMapper  - name: ResponseSelector  - name: FallbackClassifier  policies:  - name: RulePolicy |

Bảng . code đơn giản với file config

***Run rasa với API***

* + rasa run --enable-api --cors "\*" --port 5006
  + rasa run actions
  + cấu trúc json gửi xuống rasa:

|  |
| --- |
| json |
| [{"sender": "user",  "message": "Chào"}] |

Bảng . cấu trúc json để gửi request

# HỆ THỐNG QUẢN LÝ CHUỖI CUNG ỨNG

## Giới thiệu bài toán

Quản lý chuỗi cung ứng trong doanh nghiệp hiện đại là một nhiệm vụ phức tạp, đòi hỏi sự liên kết chặt chẽ giữa các bộ phận như nhà cung cấp, kho hàng, vận chuyển và khách hàng. Doanh nghiệp thường đối mặt với các thách thức như theo dõi hàng tồn kho, quản lý đặt hàng từ nhà cung cấp, điều phối vận chuyển và xử lý đơn hàng khách hàng một cách hiệu quả.

Hiện tại, quá trình này thường sử dụng phương pháp thủ công hoặc hệ thống không liên kết, dẫn đến thiếu thông tin chính xác, chậm trễ giao hàng, và khó khăn trong việc theo dõi đơn hàng, làm tăng chi phí và giảm sự hài lòng của đại lý.

Để giải quyết vấn đề, một hệ thống quản lý chuỗi cung ứng hiệu quả cần được xây dựng để tự động hóa quy trình, từ đặt hàng với nhà cung cấp, quản lý tồn kho đến vận chuyển sản phẩm đến khách hàng. Hệ thống này giúp tối ưu hóa quy trình, giảm sai sót và tăng cường khả năng cạnh tranh.

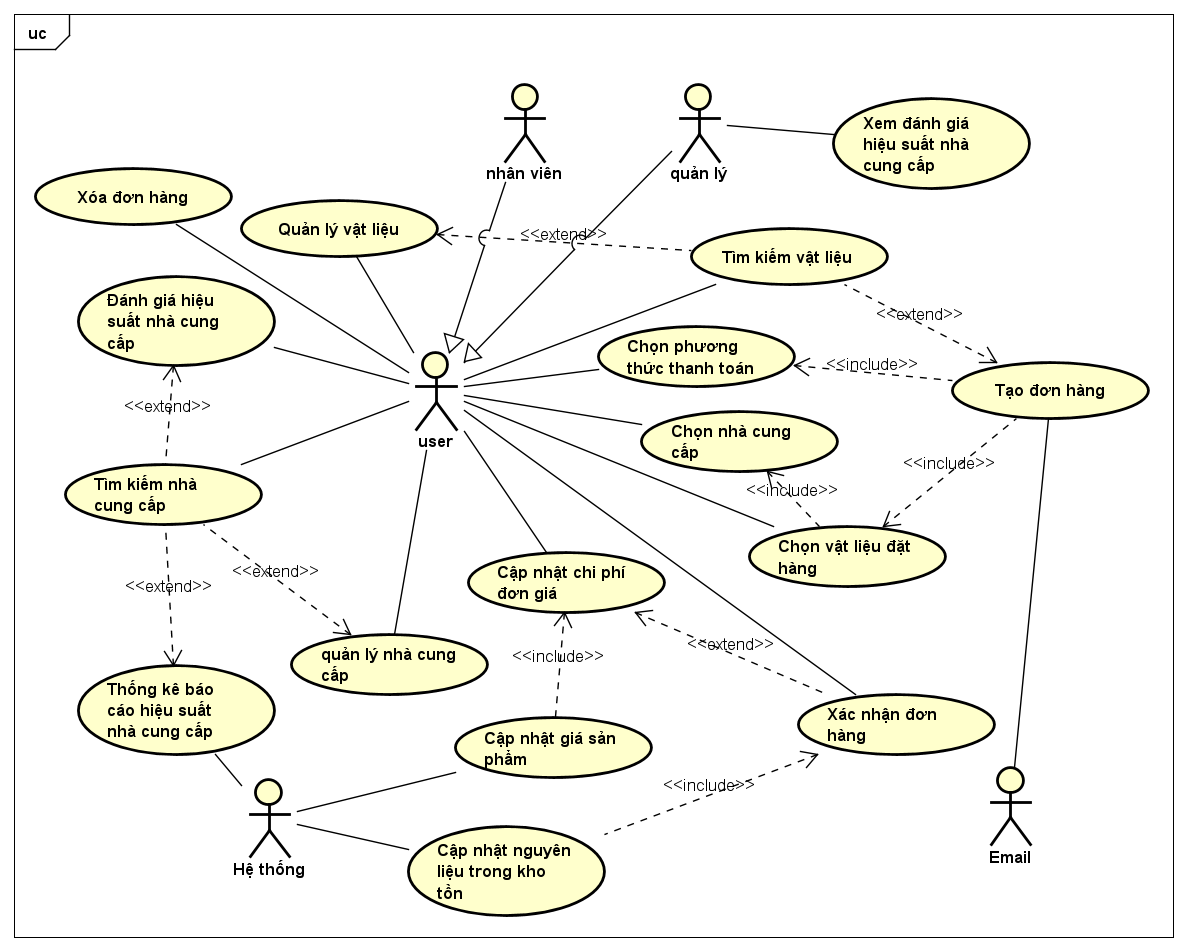
Phần báo cáo này sẽ phân tích hệ thống quản lý chuỗi cung ứng với các chức năng chính như đặt hàng, quản lý tồn kho, và theo dõi trạng thái đơn hàng, thông qua các sơ đồ và mô hình minh họa sự vận hành và tương tác với các tác nhân liên quan.

## Phân tích hệ thống

### Sơ đồ use case đặt hàng từ nhà cung cấp

Quy trình tạo đơn hàng nhập từ nhà cung cấp yêu cầu hệ thống quản lý chuỗi cung ứng đảm bảo các bước như chọn nhà cung cấp, vật liệu cần nhập và phương thức thanh toán được thực hiện chính xác và hiệu quả. Use case này mô tả cách người dùng tương tác với hệ thống để tạo đơn hàng, từ việc chọn nhà cung cấp, tìm kiếm và thêm vật liệu, đến xác nhận và hoàn tất đơn hàng.

Mục tiêu của sơ đồ use case là phân tích các hành động và luồng công việc giữa người dùng và hệ thống, giúp nhận diện các bước chính trong quá trình đặt hàng, các trường hợp ngoại lệ và phản hồi của hệ thống. Sơ đồ cũng hình dung sự tương tác giữa nhà cung cấp, hệ thống, quản lý và nhân viên kho.



Hình . Sơ đồ use case về đặt hàng từ nhà cung cấp

### Đặc tả use case đặt hàng từ nhà cung cấp

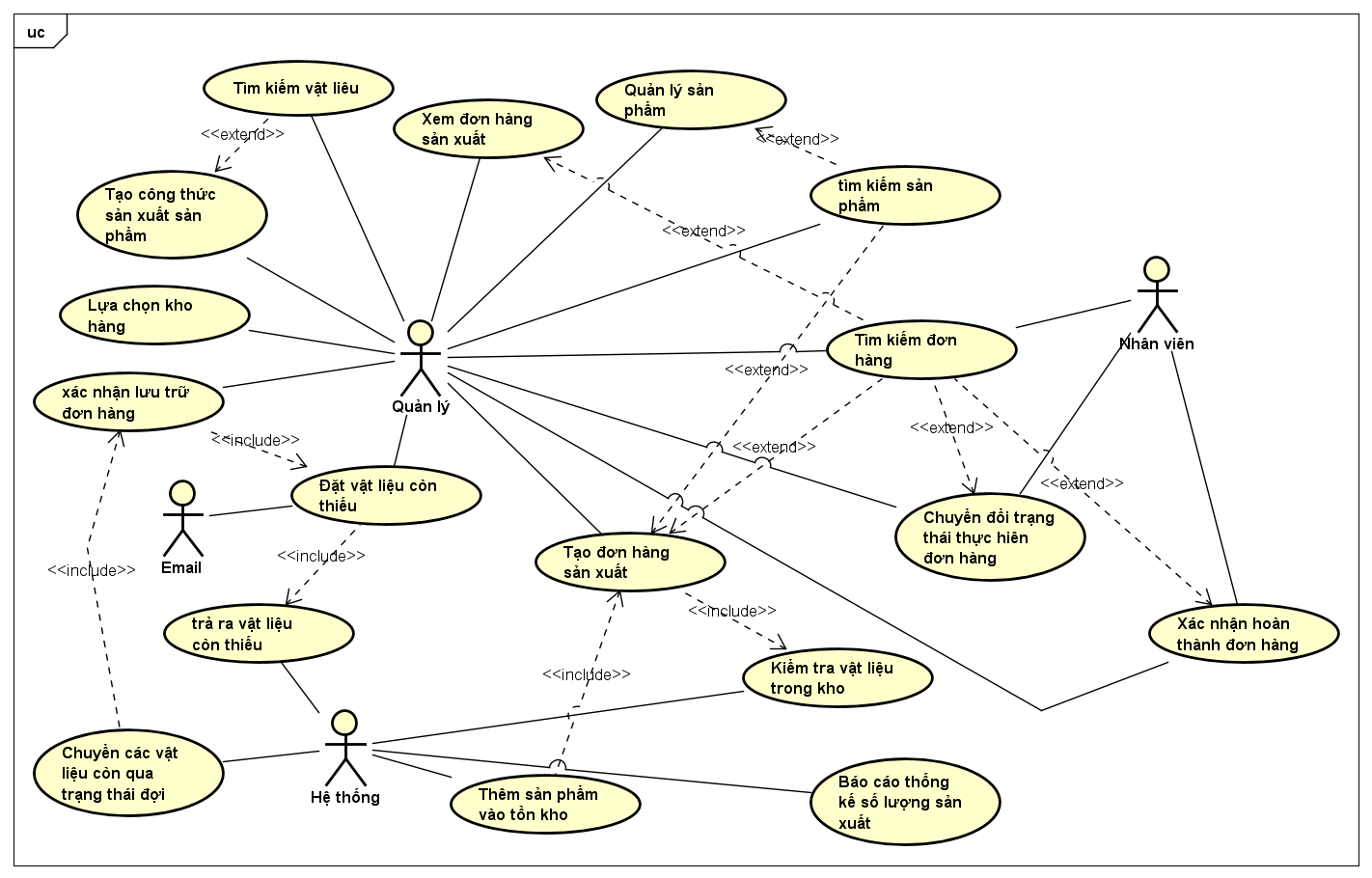
|  |  |
| --- | --- |
| **Tên use case** | Đặt đơn hàng từ nhà cung cấp |
| **Mô tả** | Use case mô tả quá trình đặt hàng từ quản lý và tiếp nhận đơn hàng từ nhà cung cấp |
| **Actor chính** | Quản lý, hệ thống |
| **Actor phụ** | Email |
| **Pre-conditions**  **(Tiền điều kiện)** | Quản lý phải đăng và xác thực thì mới được vào hệ thống để tạo đơn hàng |
| **Post-conditions**  **(Hậu điều kiện)** | Thông báo kết quả hoàn thành thanh toán và xác nhận đơn hàng để hệ thống cập nhật dữ liệu |
| **Luồng hoạt động** | Nhân viên hoặc quản lý có thể thêm sửa xóa nguyên vật liệu, truy cập vào trang đặt hàng.  Nhân viên hoặc quản lý chọn nhà cung cấp, để hiển thị vật liệu từ nhà cung cấp được chọn để tiến hành đặt hàng, có thể tìm kiếm theo tên hoặc danh mục.  sau khi nhân viên hoặc quản lý hoàn tất đơn hàng và bấm đặt đơn hàng thì đơn hàng sẽ xuất ra bảng pdf và gửi qua email cho nhà cung cấp.  sau khi hoàn tất đơn hàng nhập từ nhà cung cấp nhân viên có thể xác nhận đơn hàng, nhân viên lựa chọn cập nhật đơn giá để thay đổi giá sản phẩm.  tiếp tục nhân viên quản lý tiến hành xác nhận đơn hàng chọn kho hàng để hệ thống cập nhật vật liệu trong đơn hàng vào kho hàng.  Sau khi xác nhận đơn hàng nhân viên quản lý có thể bắt đầu tiến hành đánh giá nhà cung cấp thông qua đơn hàng đã được đặt.  Nhân viên quản lý truy cập vào trang thống kê báo cáo đánh giá để xem đánh giá hiệu suất |
| **Luồng thay thế** | Nếu là nhân viên xác nhận đơn hàng thì hệ thống sẽ tự cập nhật vào kho hàng mà nhân viên đang công tác |
| **Luồng ngoại lệ** | Trong trường hợp gặp lỗi khi tạo đơn hàng nhân viên hoặc quản lý tiến hành đặt hàng lại |

Bảng . Đặc tả quy trình đặt hàng với nhà cung cấp

### Sơ đồ use case tạo đơn hàng sản xuất

Sơ đồ use case tạo đơn hàng sản xuất mô tả quy trình lập đơn hàng để bắt đầu quá trình sản xuất trong hệ thống quản lý chuỗi cung ứng. Đây là một bước quan trọng để đảm bảo việc sản xuất diễn ra theo đúng kế hoạch và đáp ứng được yêu cầu từ các đơn hàng, khách hàng hoặc nhu cầu tồn kho.

Use case này sẽ mô tả các tác nhân chính, chẳng hạn như quản lý, nhân viên kho và hệ thống, cùng các tương tác giữa họ trong quá trình tạo đơn hàng sản xuất. Các bước bao gồm lập kế hoạch sản xuất, xác nhận thông tin sản phẩm, kiểm tra các vật liệu cần thiết để tạo sản phẩm. Sơ đồ sẽ giúp làm rõ cách hệ thống hỗ trợ người dùng trong việc lập và quản lý đơn hàng sản xuất một cách hiệu quả.



Hình . use case tạo đơn hàng sản xuất

### Đặc tả use case tạo đơn hàng sản xuất

|  |  |
| --- | --- |
| **Tên use case** | Tạo đơn hàng sản xuất |
| **Mô tả** | Use case mô tả quy trình tạo đơn hàng sản xuất cho chuỗi cung ứng |
| **Actor chính** | Quản lý, nhân viên, khách hàng, hệ thống |
| **Actor phụ** | Email |
| **Pre-conditions**  **(Tiền điều kiện)** | Quản lý phải đăng nhập và xác thực thì mới được vào hệ thống để tạo đơn hàng |
| **Post-conditions**  **(Hậu điều kiện)** | Vật liệu cần thiết phải đủ trong kho thì mới tạo đơn hàng thành công |
| **Luồng hoạt động** | Quản lý truy cập vào hệ thống nếu tiến hành tạo đơn hàng xuất.  Chọn sản phẩm cần đặt hàng và chọn kho hàng để sản xuất.  Sau khi xác nhận đơn hàng hệ thống sẽ tiến hành kiểm tra các vật liệu trong kho có đủ để tạo sản phẩm không.  Hệ thống trả ra các vật liệu còn thiếu.  Quản lý chọn đặt hàng và chọn phương thức thanh toán cho các đơn hàng từ nhiều nhà cung cấp  Hệ thống sẽ ghi nhận và gửi mail file pdf cho tất cả các nhà cung cấp cho đơn hàng.  Quản lý xác nhận lưu trữ đơn hàng  Hệ thống sẽ cập nhật các vật liệu đợi cho đến khi đơn hàng từ nhà cung cấp về và đủ.  Quản lý tiến hành xác nhận tạo đơn hàng, để chuyển đổi trạng thái cho nhân viên nhận thực hiện  10. Nhân viên xác nhận các trạng thái trong quá trình thực hiện đơn hàng  11. Sau khi nhân viên xác nhận hoàn thành đơn hàng hệ thống sẽ cập nhật sản phẩm vào kho tồn sản phẩm |
| **Luồng thay thế** | Nếu vật liệu đủ thì quản lý có thể tiến hành đặt và trạng thái đợi nhân viên thực hiện |
| **Luồng ngoại lệ** | Trong trường hợp sản phẩm chưa có công thức quản lý phải tạo công thức cho sản phẩm để tiến hành đặt hàng |

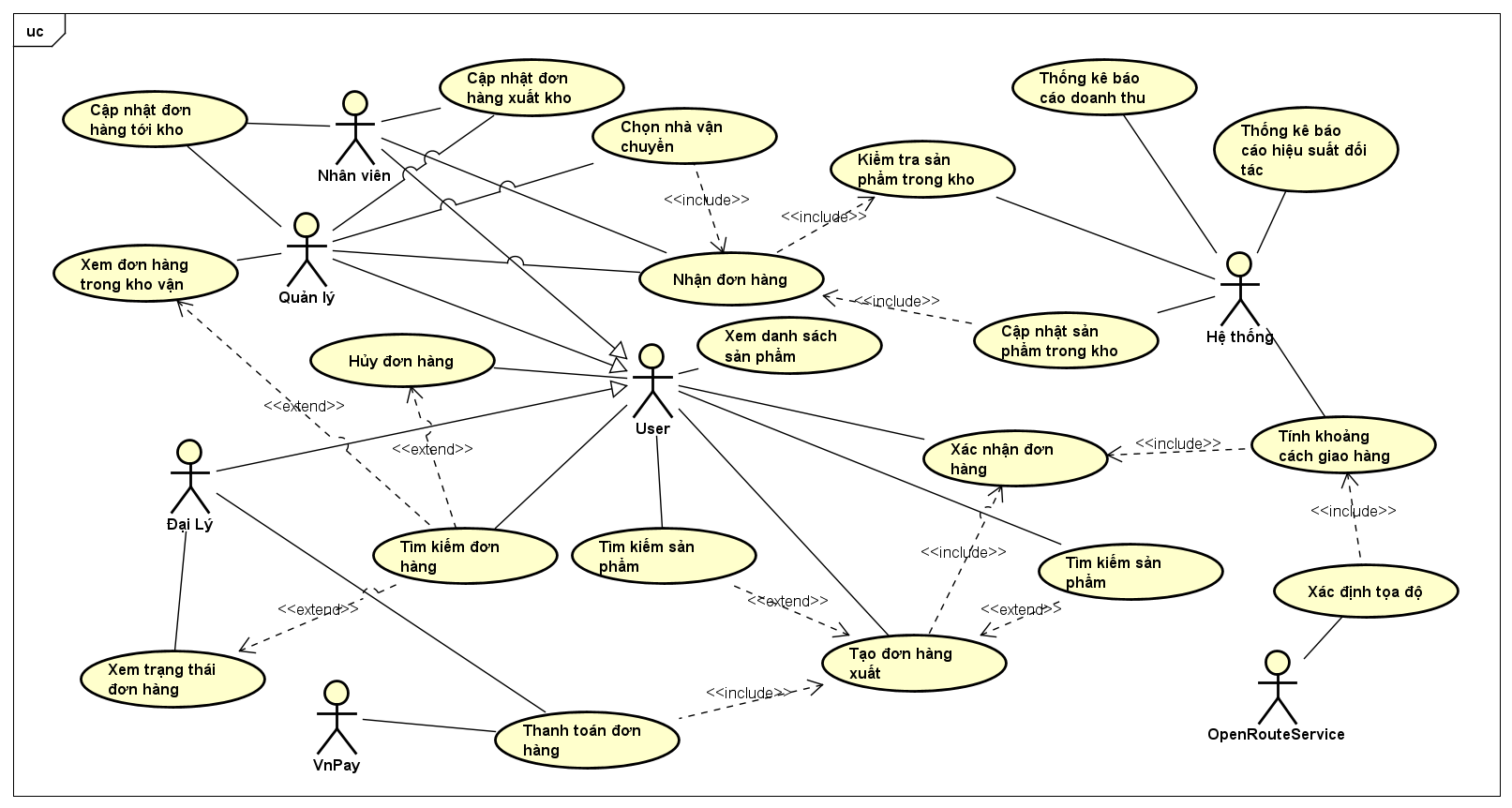
Bảng . đặc tả use tạo đơn hàng sản xuất

### Use case mua hàng từ đại lý và theo dõi trạng thái đơn hàng

Trong quy trình mua hàng từ đại lý, hệ thống quản lý chuỗi cung ứng hỗ trợ người dùng trong việc tìm kiếm và chọn sản phẩm, thực hiện thanh toán, cũng như theo dõi trạng thái của đơn hàng. Use case này mô tả cách hệ thống giúp người dùng thực hiện các tác vụ liên quan đến mua hàng từ đại lý, từ khâu tìm kiếm sản phẩm, đến việc xác nhận đơn hàng và hoàn tất giao dịch.

Ngoài ra, người dùng có thể theo dõi trạng thái của đơn hàng thông qua hệ thống, biết được đơn hàng đang ở đâu thông qua tính năng xác định tọa độ và tính toán khoảng cách giao hàng. Hệ thống còn tích hợp các tác nhân khác như VnPay để xử lý thanh toán và OpenRouteService để hỗ trợ tính toán lộ trình giao hàng.

Mục tiêu của sơ đồ use case này là cung cấp cái nhìn rõ ràng về các bước cần thiết để hoàn thành một giao dịch mua hàng từ đại lý, đồng thời hỗ trợ theo dõi trạng thái đơn hàng sau khi giao dịch hoàn tất.



Hình . sơ đồ use case mua hàng từ đại lý và theo dõi trạng thái đơn hàng

### Đặc tả use case mua hàng từ đại lý

|  |  |
| --- | --- |
| **Tên use case** | Mua hàng của đại lý và theo dõi trạng thái đơn hàng |
| **Mô tả** | Use case này mô tả quá trình đại lý mua hàng từ hệ thống, từ khâu đặt hàng, thanh toán đến theo dõi trạng thái đơn hàng và cập nhật kho hàng. |
| **Actor chính** | Quản lý, nhân viên, đại lý, hệ thống |
| **Actor phụ** | VnPay, OpenRouteService |
| **Pre-conditions**  **(Tiền điều kiện)** | User phải đăng và xác thực thì mới được vào hệ thống để tạo đơn hàng  Hệ thống có đầy đủ thông tin về sản phẩm và kho hàng. |
| **Post-conditions**  **(Hậu điều kiện)** | Thông báo kết quả hoàn thành thanh toán, xác nhận đơn hàng và cập nhật trạng thái đơn hàng cho đại lý và hệ thống cập nhật dữ liệu.  Cập nhật trạng thái đơn hàng cho đại lý.  Cập nhật dữ liệu kho. |
| **Luồng hoạt động** | User truy cập vào hệ thống để đặt hàng.  Khi lựa chọn sản phẩm và hoàn thành đơn hàng user bấm xác nhận hệ thống sẽ tự tính sản phẩm hiện trong kho và khoảng cách để đưa ra giá cho chi phí.  sau khi đặt hàng xong đại lý sẽ phải thanh toán trực tuyến qua VnPay hoặc Thanh toán.  sau khi hoàn tất thanh toán nhân viên cửa hàng sẽ xác nhận hóa đơn để hệ thống cập nhật số lượng kho.  tiếp đó quản lý sẽ lên lịch giao hàng và chọn nhà vận chuyện thích hợp  khi đơn hàng vận chuyển đến các kho nhân viên tại kho sẽ cập nhật trạng thái đơn hàng cho người dùng có thể theo dõi  đại lý có thể theo dõi quá trình đơn hàng được vận chuyển và quản lý có thể xem kho hàng có bao nhiêu đơn hàng trong kho đó.  sau khi nhận hàng đại lý cũng có thể truy cập vào hỗ trợ khách hàng để đưa ra ý kiến nếu muốn |
| **Luồng thay thế** | Nếu khi xác nhận đơn hàng mà không đủ sản phẩm thì quản lý phải tiến hành tạo đơn sản xuất để đủ điều kiện xác nhận |
| **Luông ngoại lệ** | Trong trường hợp nếu đặt đơn hàng thất bại hệ thống sẽ thông báo cho khách hàng order lại |

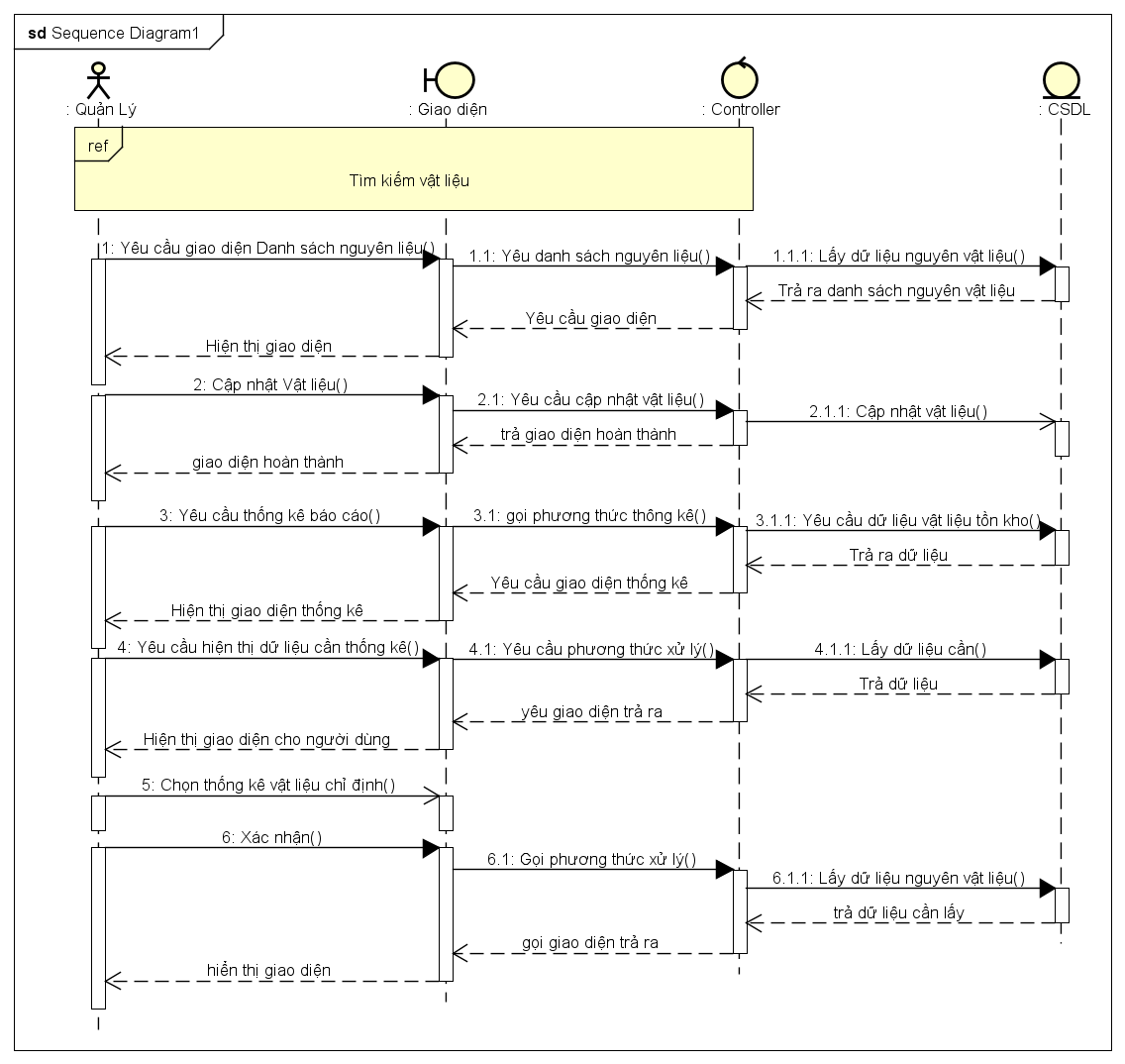
Bảng . Đặc tả use case mua hàng từ đại lý

## Thiết kế hệ thống

### Sơ đồ tuần tự (sequence diagram) và Sơ đồ trạng thái (State diagram)

#### Quản lý nguyên vật liệu và yêu cầu thống kê

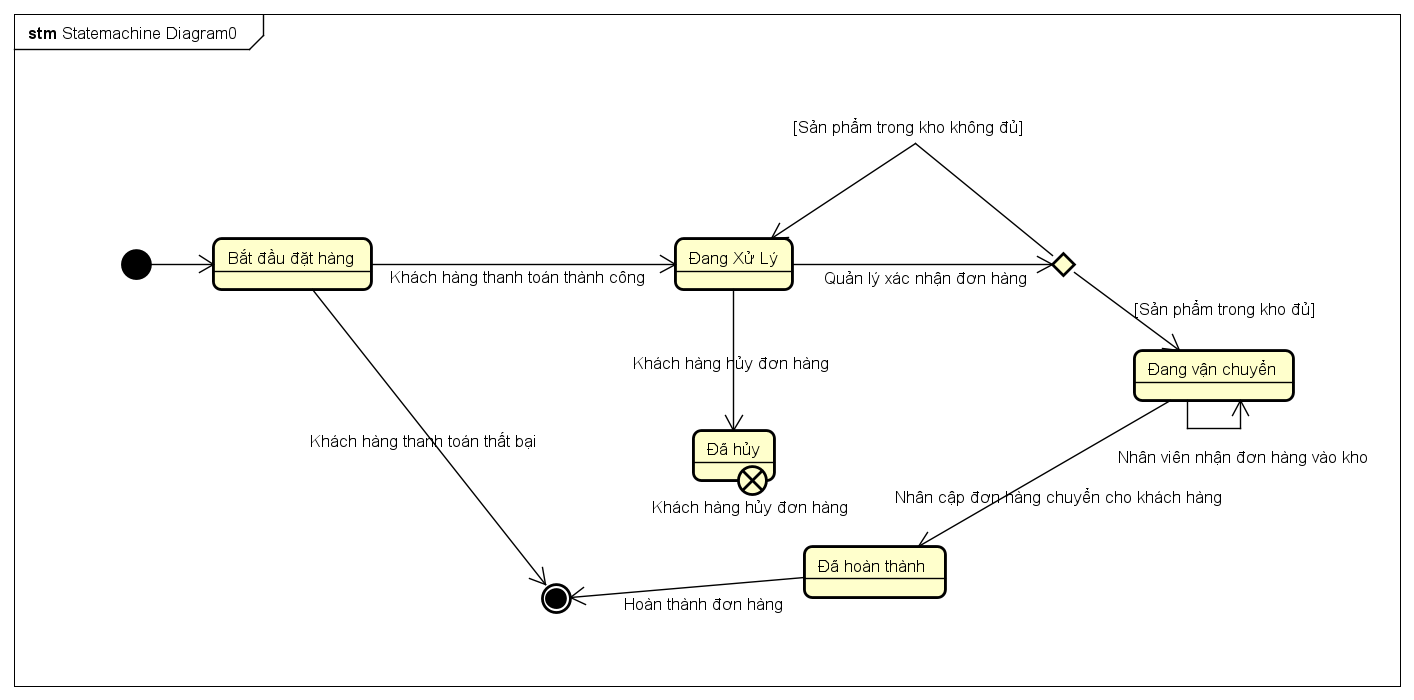
Sơ đồ tuần tự về quản lý nguyên vật liệu và thống kê sẽ giúp chúng ta hiểu rõ hơn về quy trình hoạt động của hệ thống. Bằng cách trực quan hóa các bước từ việc yêu cầu danh sách nguyên vật liệu đến việc tạo báo cáo thống kê, sơ đồ sẽ cho thấy một cái nhìn tổng quan về cách thức các thành phần trong hệ thống tương tác với nhau.



Hình . sơ đồ quy trình cập nhật nguyên vật liệu và yêu cầu thống kê

#### lên lịch giao hàng cho đại lý và xem trạng thái

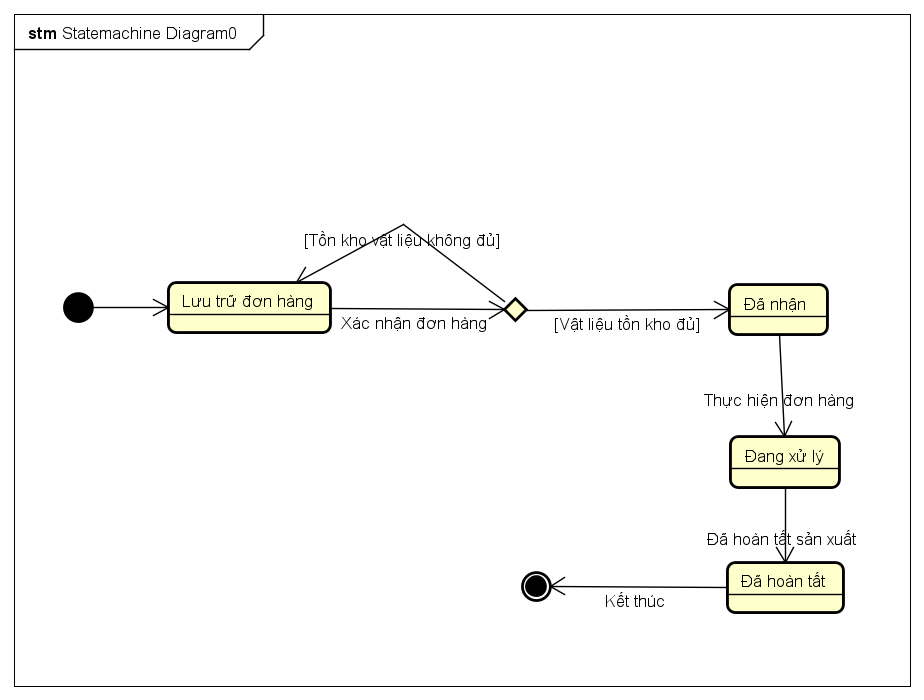
Và để hiểu rõ hơn về trạng thái của đơn hàng từ lúc bắt đầu tạo cho đến lúc kết thúc đơn hàng sẽ diễn ra như thế nào dưới đây là sơ đồ đưa ra rõ quy trình về trạng thái thay đổi của đơn hàng khi được tác động:



Hình . Sơ đồ trạng thái đơn hàng

#### Quản lý quy trình sản xuất

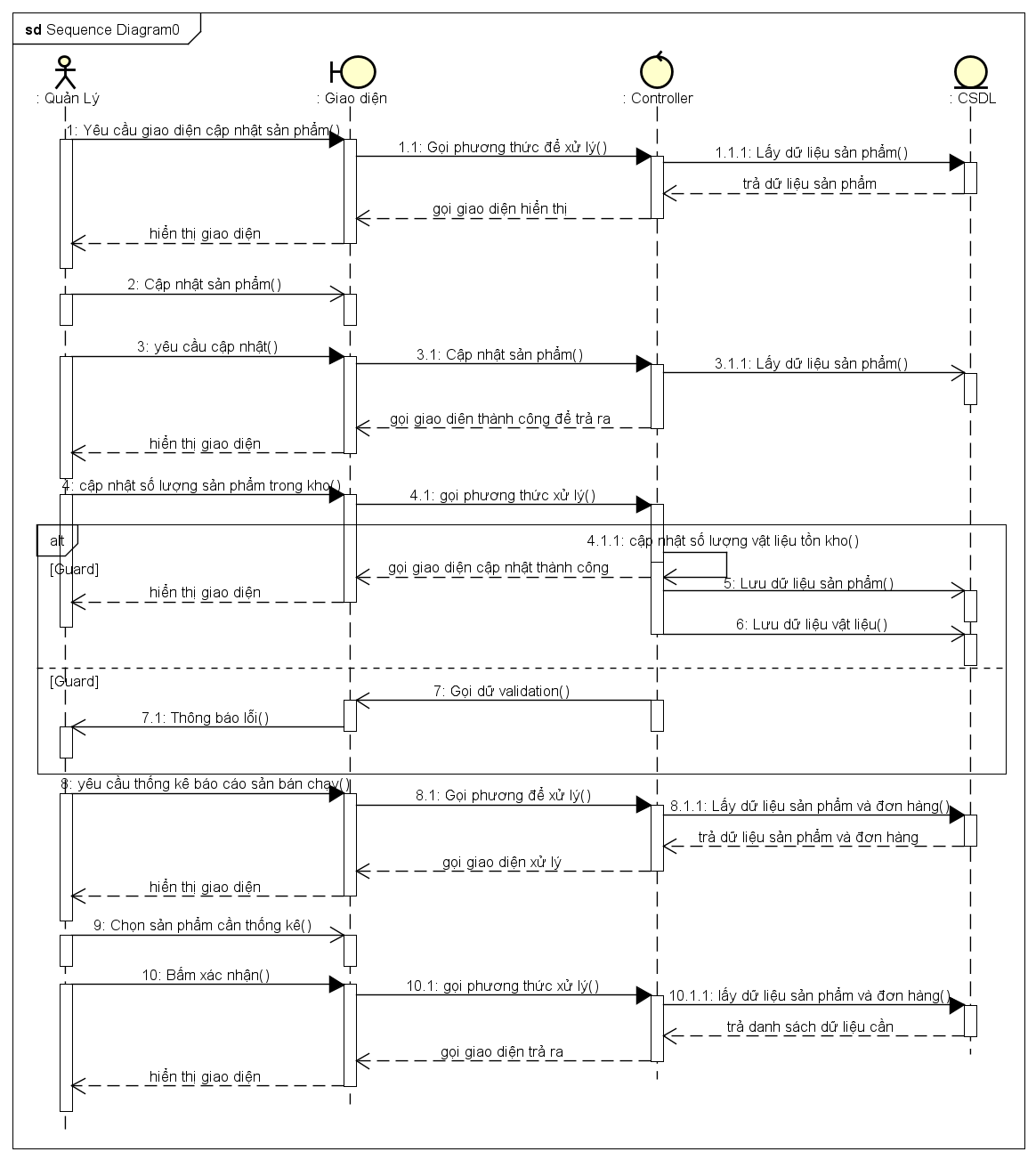
Để hiểu hơn về trạng thái của đơn hàng sản xuất, dưới đây là sơ đồ trạng thái của đơn hàng sản xuất, đơn hàng sẽ ở trạng thái lưu trữ để đổi trạng thái cho đơn hàng lưu trữ thì vật liệu trong kho phải đảm bảo đủ, và nhân viên hoặc quản lý sẽ cập nhật trạng thái dựa trên tình hình sản xuất.



Hình . Sơ đồ trạng thái cho đơn hàng sản xuất

#### Quản lý sản phẩm và thống kê báo cáo doanh thu

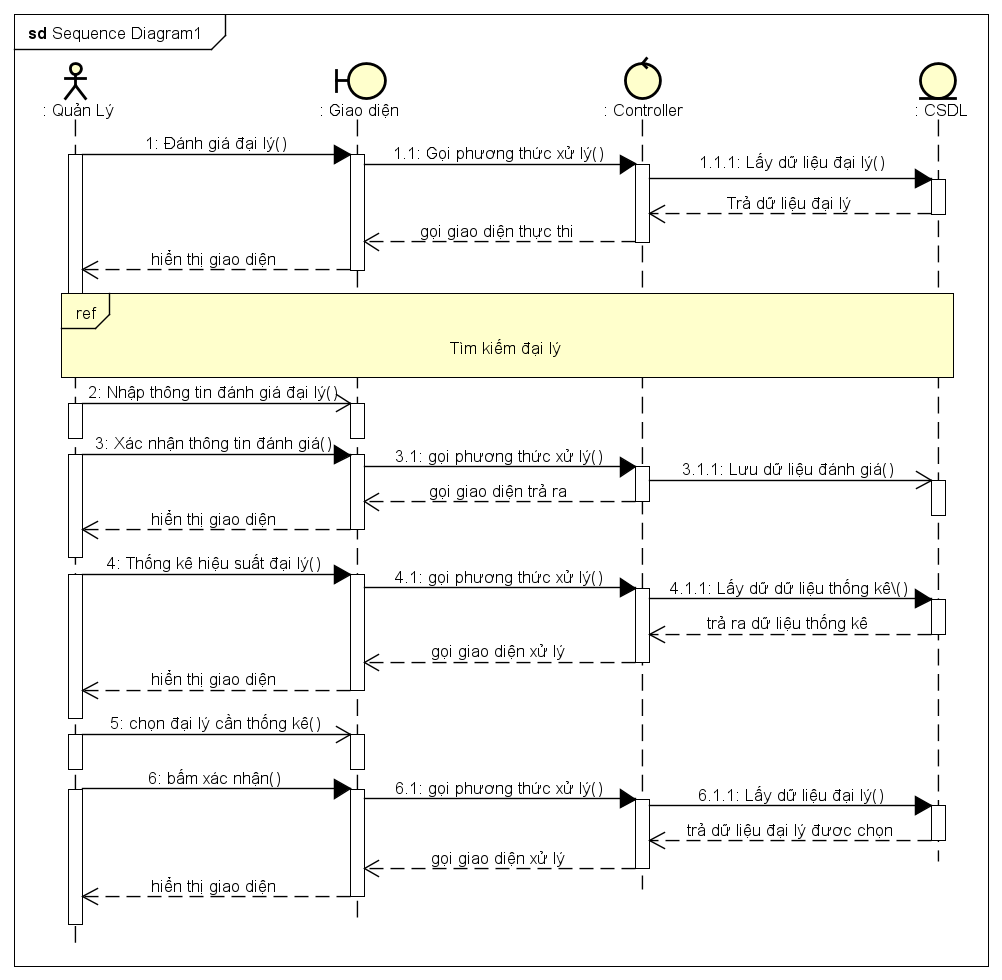
Tiếp đến, dưới đây với sơ đồ tuần tự này sẽ trình bày một cách chi tiết quy trình quản lý sản phẩm và thống kê doanh thu, bắt đầu từ việc cập nhật thông tin sản phẩm cho đến khi tạo ra báo cáo doanh thu. Sơ đồ này cho phép chúng ta dễ dàng hình dung các hoạt động, các thực thể liên quan, và mối liên hệ giữa chúng trong toàn bộ quy trình.



Hình . sơ đồ quy trình cập nhật sản phẩm và thống kê sản phẩm

#### Đánh giá đại lý yêu cầu thống kê hiệu suất

Sơ đồ này sẽ mô tả chi tiết từng bước trong quy trình, từ thu thập dữ liệu đến đánh giá cuối cùng, đồng thời minh họa rõ ràng sự tương tác giữa các thành phần trong hệ thống, giúp ta nắm bắt toàn bộ quá trình thực hiện một cách trực quan và hệ thống.

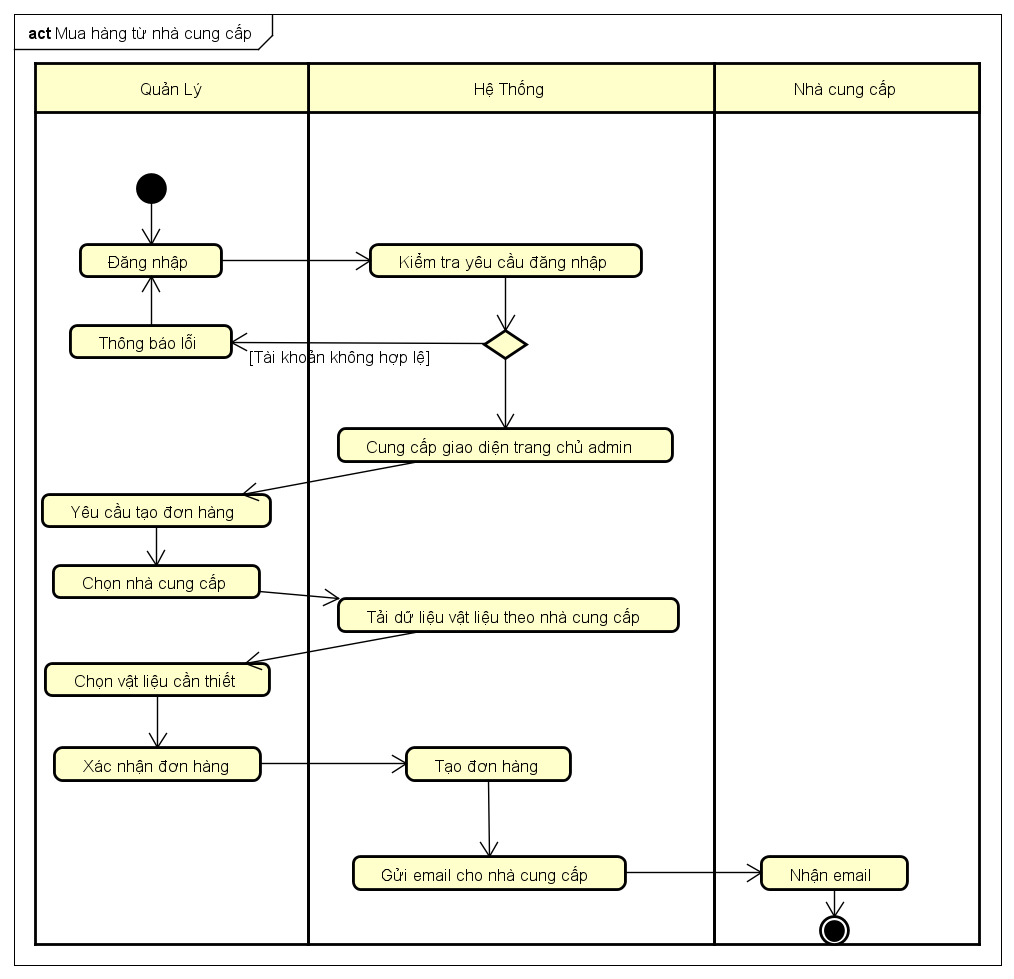


Hình . Sơ đồ quy trình đánh giá đại lý và yêu cầu thống kê hiệu suất

### Sơ đồ hoạt động (activity diagram)

#### Chức năng đặt hàng từ nhà cung cấp

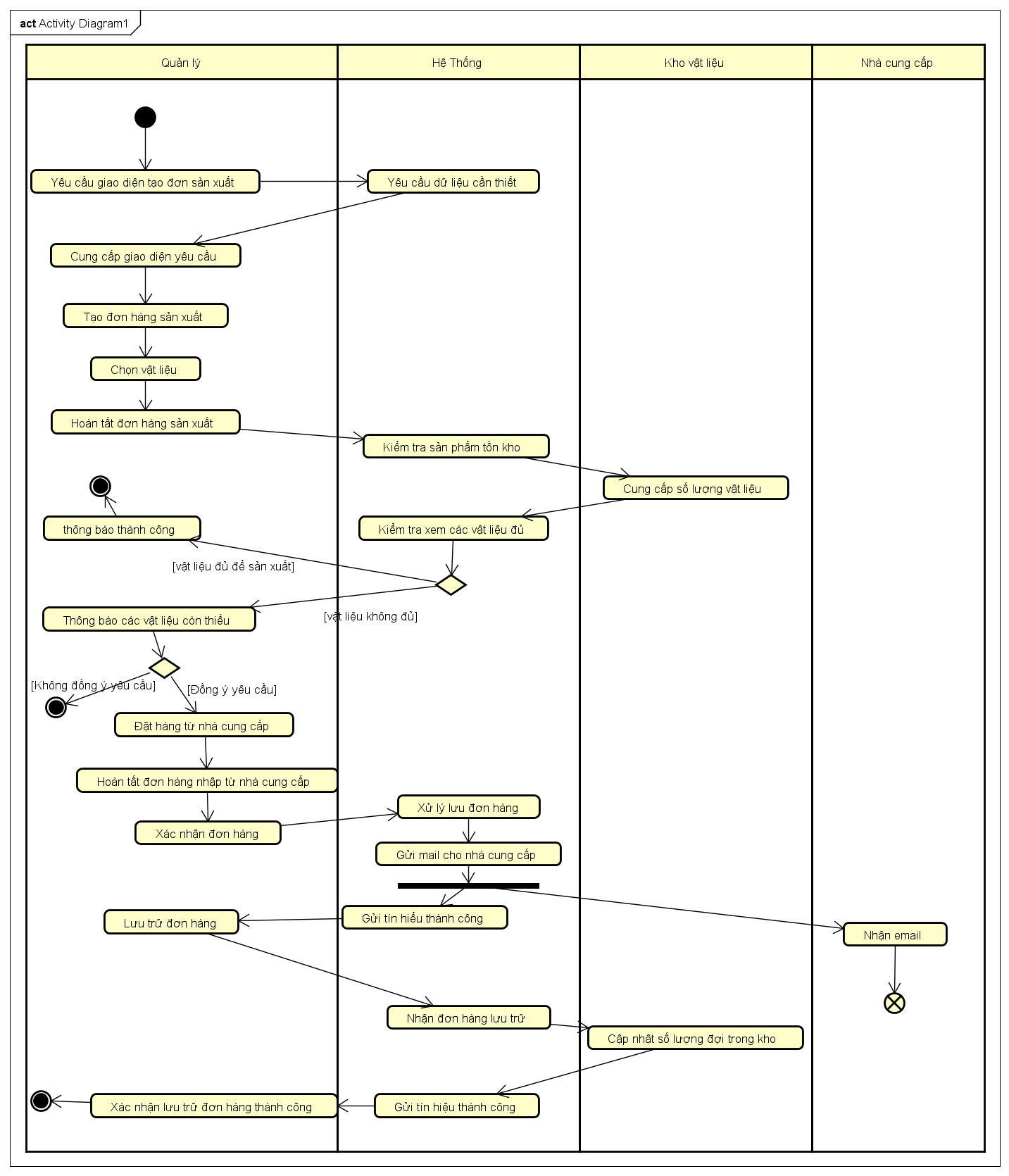
Sơ đồ hoạt động dưới đây sẽ nêu rõ hơn các bước về quy trình xử lý đơn hàng nhập từ nhà cung cấp, Qua sơ đồ này ta có thể hình dung rõ ràng hơn về các hoạt động chính, Mỗi bước trong quy trình đều được thể hiện bằng các hình khối và mũi tên, tạo thành một luồng công việc logic và dễ theo dõi.



Hình . Sơ đồ hoạt động về quy trình mua hàng từ nhà cung cấp

#### Chức năng tạo đơn hàng sản xuất

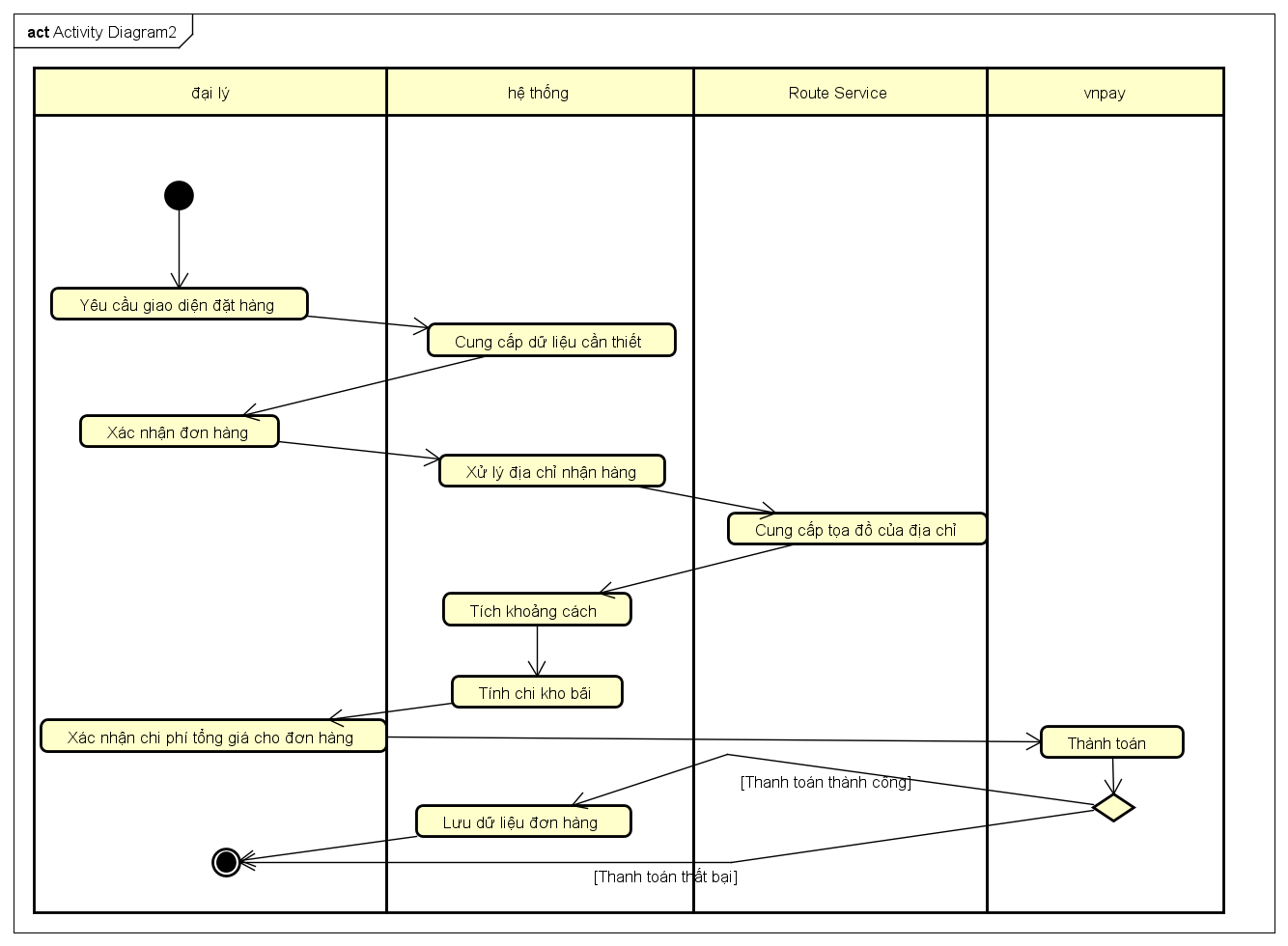
Sau khi xác nhận đơn hàng từ nhà cung cấp, người dùng có thể tạo nên những công thức theo các vật liệu được lưu trên hệ thống và tiến hành lên đơn hàng để sản xuất sản phẩm, dưới đây sẽ là sơ đồ hoạt động mô tả chi tiết hơn về quy trình tạo đơn sản xuất nhằm sản xuất sản phẩm.



Hình . Sơ đồ hoạt động cho quy trình đặt hàng

#### Chức năng xuất đơn hàng cho khách hàng

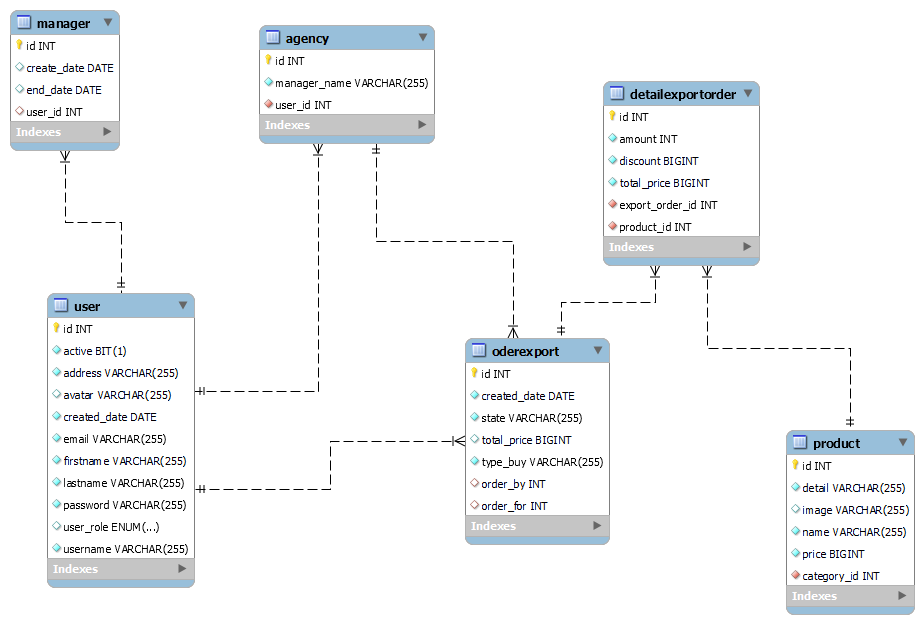
Quy trình đặt hàng được hỗ trợ đại lý đặt hàng và thanh toán một cách dễ dàng trên hệ thống, giúp rút ngắn thời gian xử lý đơn hàng và giảm thiểu lỗi. Sơ đồ hoạt động này minh họa chi tiết các bước thực hiện, từ khi khách hàng gửi yêu cầu đến khi đơn hàng được xác nhận và chuyển đến bộ phận giao hàng.



Hình . Sơ đồ hoạt động cho quy trình đặt hàng

### Thiết kế cơ sở dữ liệu

#### Thông tin tạo đơn hàng xuất cho đại lý



Hình . Các mối quan hệ ràng buộc cho nghiệp vụ mua hàng

1. Mối quan hệ giữa các bảng **OrderExport** 1-N **DetailExportorder\_product** và **DetailExport\_Material**:

* Đối với các nghiệp vụ tạo đơn hàng xuất cho người mua hàng sẽ có loại người mua hàng đó là khách hàng chỉ mua sản phẩm hoặc là đại lý có thể mua cả sản phẩm và vật liệu nên hai bảng chi tiết đơn hàng sẽ cùng nối vào đơn hàng để dễ dàng lưu thông tin.
* Cũng như hai bảng **Material** và **Product** khi tạo đơn hàng thì một đơn hàng có thể có nhiều sản phẩm hoặc vật liệu còn sản phẩm và vật liệu cũng có thể nằm trong nhiều đơn hàng.

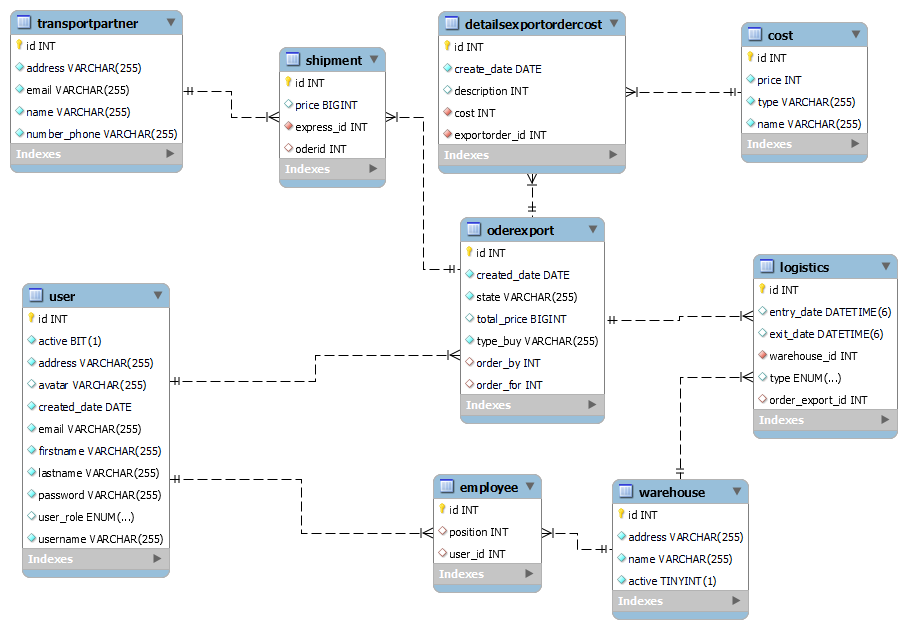
1. Mối quan hệ giữa các bảng **OrderExport** 1-1 **User**:

* Với mỗi đơn hàng sẽ được tạo bởi 1 User có thể là Manager hoặc là Admin nhằm lưu thông dễ dàng xác định đơn hàng được order từ ai nếu có vấn đề xảy ra.

1. Mối quan hệ giữa các bảng **Agency** 1-1 **User**:

* Đại lý vẫn có thể truy cập vào hệ thống để tạo đơn hàng phụ thuộc vào quyền của User nên hai bảng lưu thông tin của đại lý hay customer sẽ khóa ngoại đến User như kế thừa để đảm bảo tính toàn vẹn thông tin.

#### Thông tin nghiệp vụ giao hàng



Hình . Các mối quan hệ ràng buộc giao đơn hàng

1. Mối quan hệ giữa các bảng **Shipment** N-1 **OrderExport và TransportPartner**:

* Khi tạo đơn hàng xuất cho đại lý hay khách hàng đặt hàng có cần giao hàng thì ta phải thực hiện nghiệp vụ như cập nhật đơn hàng hay chọn nhà vận chuyển để giao hàng, nên Shipment sẽ lưu thông tin đó và thông tin của đơn hàng và nhà vận chuyển bởi vì một nhà vận chuyển có thể giao nhiều đơn hàng, một nhà vận chuyển thì chỉ chuyển một đơn hàng nên ta sẽ ràng buộc unique cho hai trường khóa ngoại của đơn hàng, ở bảng TransportPartner vẫn để lưu thông tin nhà vận chuyển.

2. Mối quan hệ giữa bảng **Cost** N-N **OrderExport**:

* Khi tạo một đơn hàng sẽ có các chi phí khi vận chuyển hàng đến đại lý như các chi phí vận chuyển hay là các chi phí kho bãi hoặc là chi phí vận chuyển nhanh nên nếu khi tạo đơn hàng với loại chi thì phải cần thực hiện tạo thêm hóa đơn cho chi phí đơn hàng giao, một đơn hàng có thể có 1 hoặc 2 chi phí, 1 chi phí có thể dành cho nhiều đơn hàng, nên sẽ có xuất hiện bảng phụ **DetailexportOrderCost** để hỗ trợ lưu thông tin.

3. Mối quan hệ giữa bảng **Logistic N-1 OrderExport, Logistics N-1 Warehouse:**

* Logistic là bảng để lưu thông tin kho vận sẽ lưu thông tin đơn hàng đang được vận chuyển đến các kho hàng nào lưu thông tin ngày đến ngày xuất để đại lý có thể dễ dàng theo dõi được đơn hàng của mình hiện tại đang ở kho hàng nào.

#### Thông tin hợp tác và đánh giá với các đối tác



Hình . Các mối quan hệ ràng buộc đánh giá cộng tác

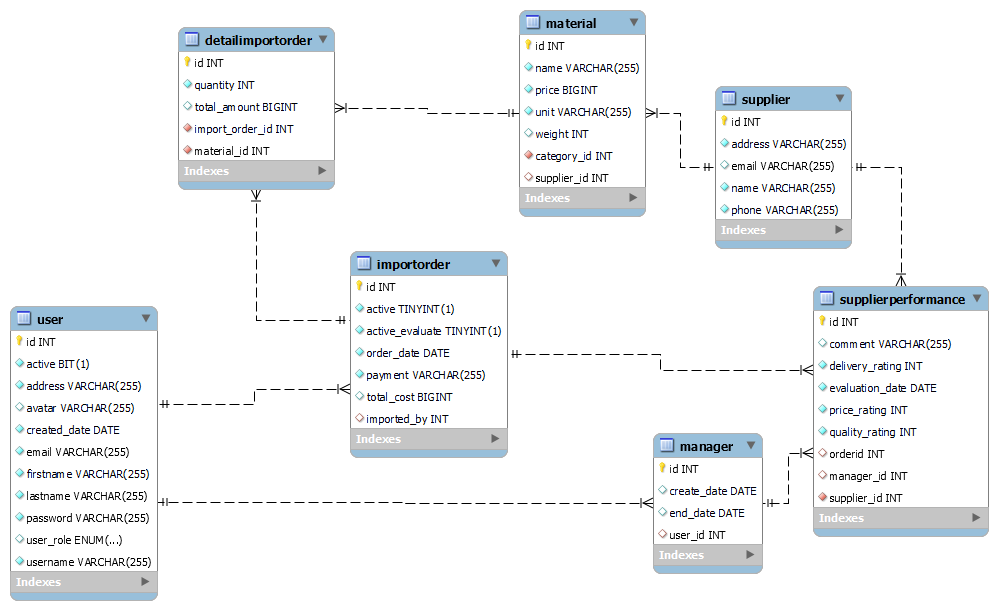
1. Mối quan hệ giữa các bảng **Partner 1-N Agency, Partner 1-N TransportPartner**:

* Agency (Đại lý) hay TransportPartner (Nhà Vận Chuyển) đều là những đối tác quan trọng của hệ thống bảng Partner (Đối tác) lưu trữ giữ liệu một nhiều đến Agency và Transportpartner nhưng để tránh tình trạng bảng bị null khi join các bảng lại với nhau thì em xin ra thêm 2 bảng phụ.
* **PartnerAgency 1-N Agency, Partner 1-1 PartnerAgency:** để lưu trữ thông tin của đại lý với mối quan với một đại lý có thể gia hạn hợp đồng đối tác nhiều lần.
* **PartnerTransport 1-N Agency, Partner 1-1 PartnerAgency:** tương tự như Agency (Đại lý) 1 nhà vận chuyển cũng có thể ký hợp đồng đối tác nhiều lần

2. Mối quan hệ giữa các bảng **PerformanceValuation** N-1 **User, Partner**:

* Khi đánh giá cộng tác sẽ có lưu thông tin người đánh giá là nhân viên của cửa hàng và những thông tin đánh giá sẽ lưu vào bảng PerformanceValutaion 1 bảng đánh giá sẽ đánh giá 1 cộng tác, 1 cộng tác sẽ có nhiều bảng đánh giá từ nhân viên cửa hàng.

#### Thông tin tạo đơn hàng nhập và đánh giá nhà cung cấp



Hình . các mối quan hệ ràng buộc quy trình mua hàng từ nhà cung cấp

1. Mối quan hệ giữa các bảng **DetailImportOrder** N-1 **ImportOrder, Material, DetailImportOrder** N-1 User:

* Khi tạo đơn hàng từ nhà cung cấp thì một đơn hàng sẽ có nhiều vật liệu và 1 vật liệu có thể nằm ở nhiều hóa đơn, nên khi đó sẽ xin ra bảng phụ để lưu thông tin chi tiết đơn hàng, và thông tin chi tiết đơn hàng sẽ lưu lại thông tin người tạo ra đơn hàng.

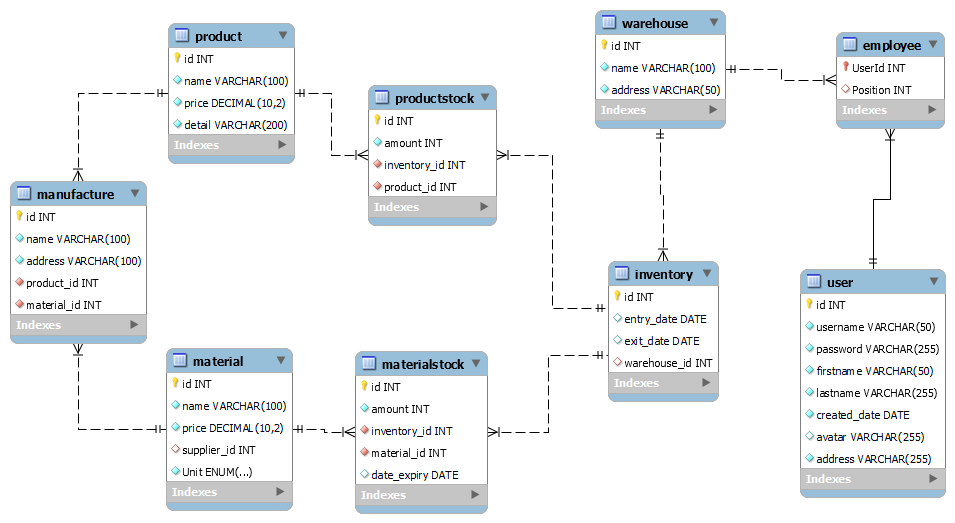
2. Mối quan hệ giữa các bảng **SupplierPerformance** N-1 **Supplier, Manager**:

* Sau khi nhận đơn hàng quản lý sẽ đánh giá nhà cung cấp và thông tin lưu cho bảng đánh giá nhà cung cấp là thông tin quản lý và nhà cung cấp do quản lý đánh giá dựa trên đơn hàng, 1 quản lý có thể đánh giá nhiều nhà cung cấp và 1 nhà cung cấp có thể bị đánh giá nhiều lần.

3. Mối quan hệ giữa bảng **MaterialPrice** N-1 **Material**:

* Mỗi khi hết hạn hợp đồng hoặc là trong quá trình ký kết hợp trình sẽ có 1 ngày để thông báo cho cửa hàng là vật liệu có giá thay đổi khi đó MaterialPrice sẽ lưu thông tin và giá vật liệu thay đổi để xem sự chênh lệch về giá sau khi thay đổi, nên 1 đơn giá sẽ lưu nhiều vật liệu.

#### Thông tin các sản phẩm tồn kho và kho hàng



Hình . các mối quan hệ ràng buộc quy trình tồn kho

1. Mối quan hệ giữa các bản **Product, Material** N-N **Inventory** N-1 **WareHouse:**

* Quá trình cũng là một phần quan trọng của doanh nghiệp đảm bảo theo dõi số lượng để nắm bắt tình hình đặt hàng từ nhà cung cấp sao cho phù hợp một tồn kho có thể chứa nhiều vật liệu hay sản phẩm, một sản phẩm có thể được chứa trong nhiều tồn kho theo ngày nhập vào và ngày hết hạn của sản phẩm.
* Mỗi tồn kho sẽ thuộc của 1 kho hàng nhất định và 1 kho hàng thì có thể có nhiều tồn kho.

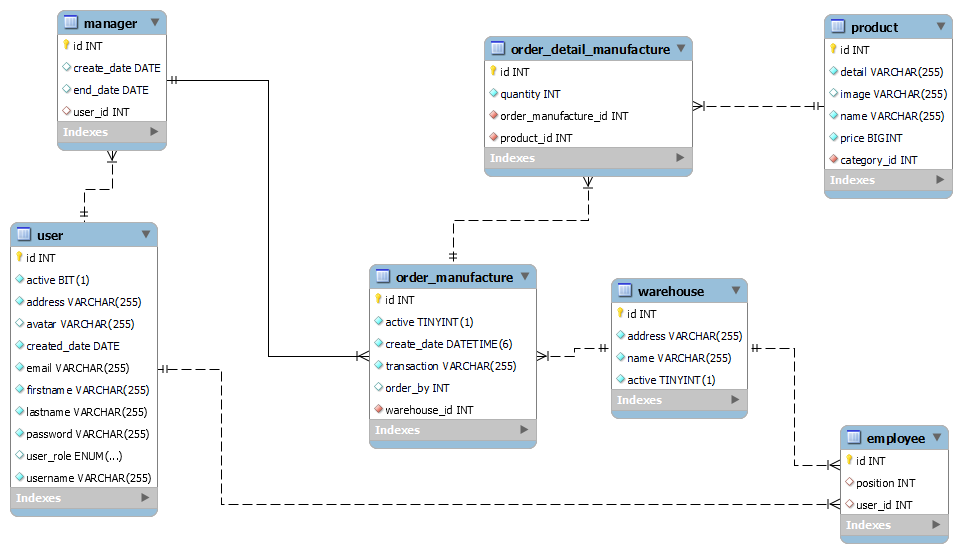
2. Mối quan hệ giữa các bảng **Manufacture** N-1 **Product, Material**:

* Khi xác nhận đơn hàng nhập từ nhà cung cấp thì hệ thống sẽ tự động lấy thông tin hóa đơn để nhập vào kho hàng và cũng như vậy khi một sản phẩm được thêm vào tồn kho hay được tạo ra thì sẽ dựa trên Manufacture như một bảng lưu thông tin công thức để tạo ra một sản phẩm để trừ vật liệu vào trong tồn kho, vì vậy một một sản phẩm tạo ra. bởi nhiều vật liệu một vật liệu cũng dùng để tạo ra cho các sản phẩm sẽ sinh ra bảng phụ là Manufacture để lưu thông tin tạo ra sản phẩm.

3. Mối quan hệ giữa bảng **Employee** N-1 **WareHouse:**

* Nhân viên kho hàng có nhiệm vụ cập nhật trạng thái đơn hàng khi đến kho trong quá trình vận chuyển kho hàng và một kho hàng có nhiều nhân viên quản lý, một nhân viên chỉ quản lý một kho hàng,

#### Thông tin quản lý đơn sản xuất



Hình . các mối quan hệ của quản lý đơn hàng sản xuất

1. Mối quan hệ giữa bản **Product N-N OrderManufacture:**

* Để đảm bảo được sản phẩm trong kho luôn tồn tại đủ sản phẩm nhằm đáp ứng kịp thời cho đại lý một cách nhanh nhất, OrderManufacture lưu trữ thông tin sản xuất các sản phẩm 1 đơn hàng sản xuất có nhiều sản phẩm và 1 nhiều sản phẩm cũng có thể nằm ở nhiều đơn hàng sản xuất nên **DetailOrderManufacture** có nhiệm vụ để lưu trữ thông tin đơn.

2. Mối quan hệ giữa bảng **OrderManufacture** **N-1 Manager:**

* Quản lý sẽ người trực tiếp quản lý (Manager) việc sản xuất các sản phẩm và mỗi đơn hàng sản xuất (OrderManufacture) sẽ lưu trữ thông tin của một quản lý tạo ra đơn hàng đó, một quản lý có thể tạo ra nhiều đơn hàng sản xuất

3. Mối quan hệ giữa các **OrderManufacture N-1 WareHouse, Employee N-1 Warehouse:**

* Mỗi một đơn hàng sản xuất (OrderManufacture) sẽ được chọn để sản xuất tại 1 kho nào đó do quản lý sẽ là người lựa chọn và mỗi kho hàng (WareHouse) và mỗi kho hàng sẽ được có nhiều đơn hàng sản xuất, nhân viên (Employee) làm việc ở tại kho hàng sẽ có trách nhiệm tiếp nhận và cập nhật trạng thái đơn hàng sản xuất để quản lý có thể nắm bắt và mỗi một nhân viên sẽ làm tại 1 kho hàng và một kho hàng có nhiều nhân viên.

## Kiến trúc hệ thống

### Tổng quan hệ thống:



Hình . Luồng xử lý dữ liệu

1. Khi người dùng gửi yêu cầu truy cập vào trang web NextJS sẽ nhận yêu cầu và bắt đầu tải trang HTML tĩnh đồng thời gửi tín hiệu xuống Spring Boots để lấy dữ liệu.

2. Khi nhận được yêu cầu Request Spring Boots sẽ tiến hành Filter để lấy dữ liệu với dạng json và chuyển về dạng request object để tiến hành xử lý nếu có lỗi thì sẽ gửi lại thông báo cho controller gửi lại tín hiệu lỗi cho client.

3. Tiếp tục đó controller sẽ gọi lớp service để tiến hàng lấy cái dữ liệu yêu cầu.

4. Trước khi gọi Repository để xử lý Spring Boot sẽ kiểm tra Cache Redis có chứa dữ liệu với đúng với keyword theo yêu cầu không nếu có thì sẽ lấy dữ liệu từ Redis và tiến hành trả dữ liệu.

5. Service sẽ thông qua lớp Repository để yêu cầu lấy dữ liệu thông quan JpaRepository hoặc truy vấn với điều kiện phức tạp với Specification để lấy dữ liệu.

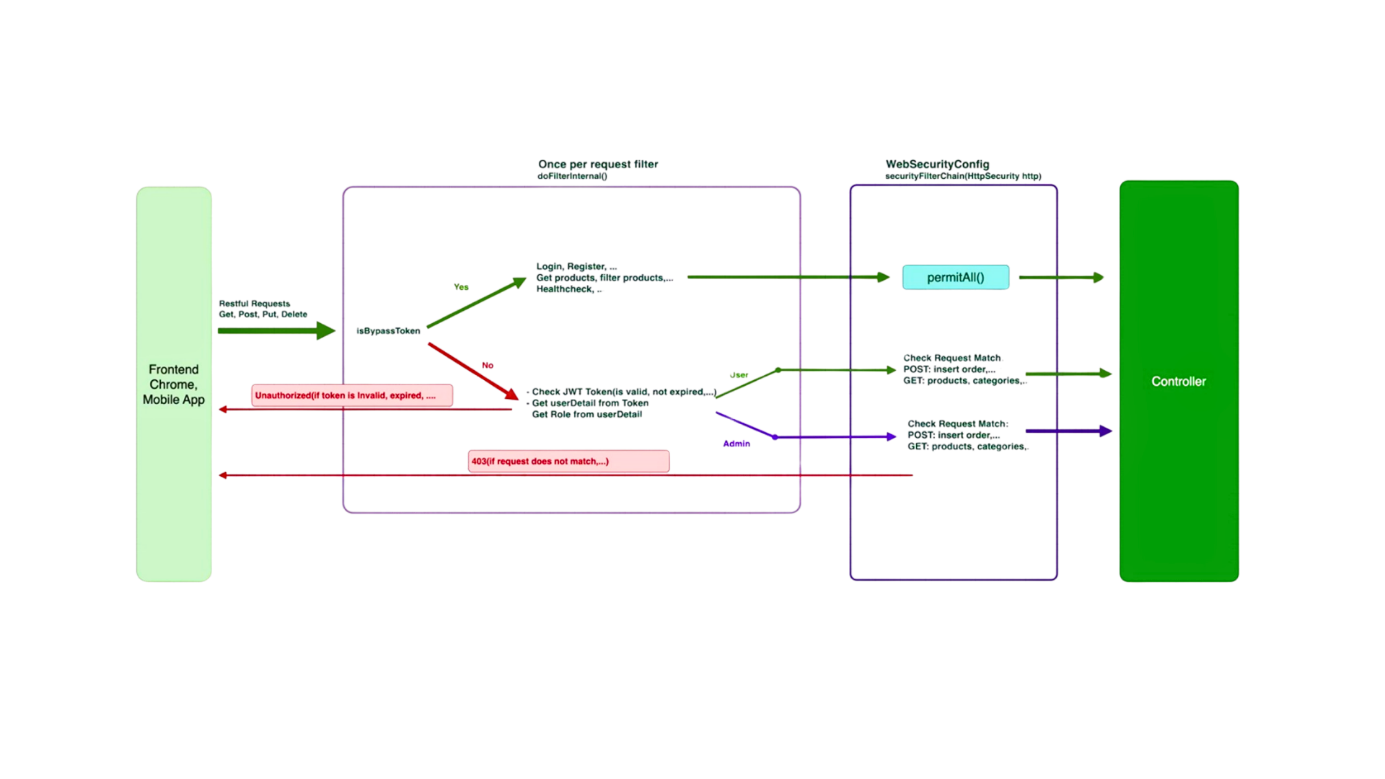
6. Sau khi hoàn tất lấy dữ liệu Controller sẽ gửi tín hiệu và dữ liệu lên lại cho Client.

7. Lúc này Client ở NextJS sẽ bắt đầu nhận dữ liệu và hoàn tất ServerSide Rendering

8. ClientSide Rendering sẽ tiến hành tải HTML và CSS ở client và gắn các JavaScripts để bắt đầu tương tác.

9. Và cuối cùng hydration sẽ bắt đầu kết xuất HTML nhận ở ServerSide Rendering và ClientSide Rendering và tải 1 trang hoàn tất đầy đủ tương tác cho người dùng.

Chứng thực khi người dùng gửi yêu cầu:

****

Hình . Luồng chứng thực người dùng

1. Người dùng gửi các yêu cầu RESTful (GET, POST, PUT, DELETE) từ ứng dụng frontend.

2. Mỗi yêu cầu sẽ đi qua một bộ lọc (filter) được cấu hình để xử lý các yêu cầu một lần.

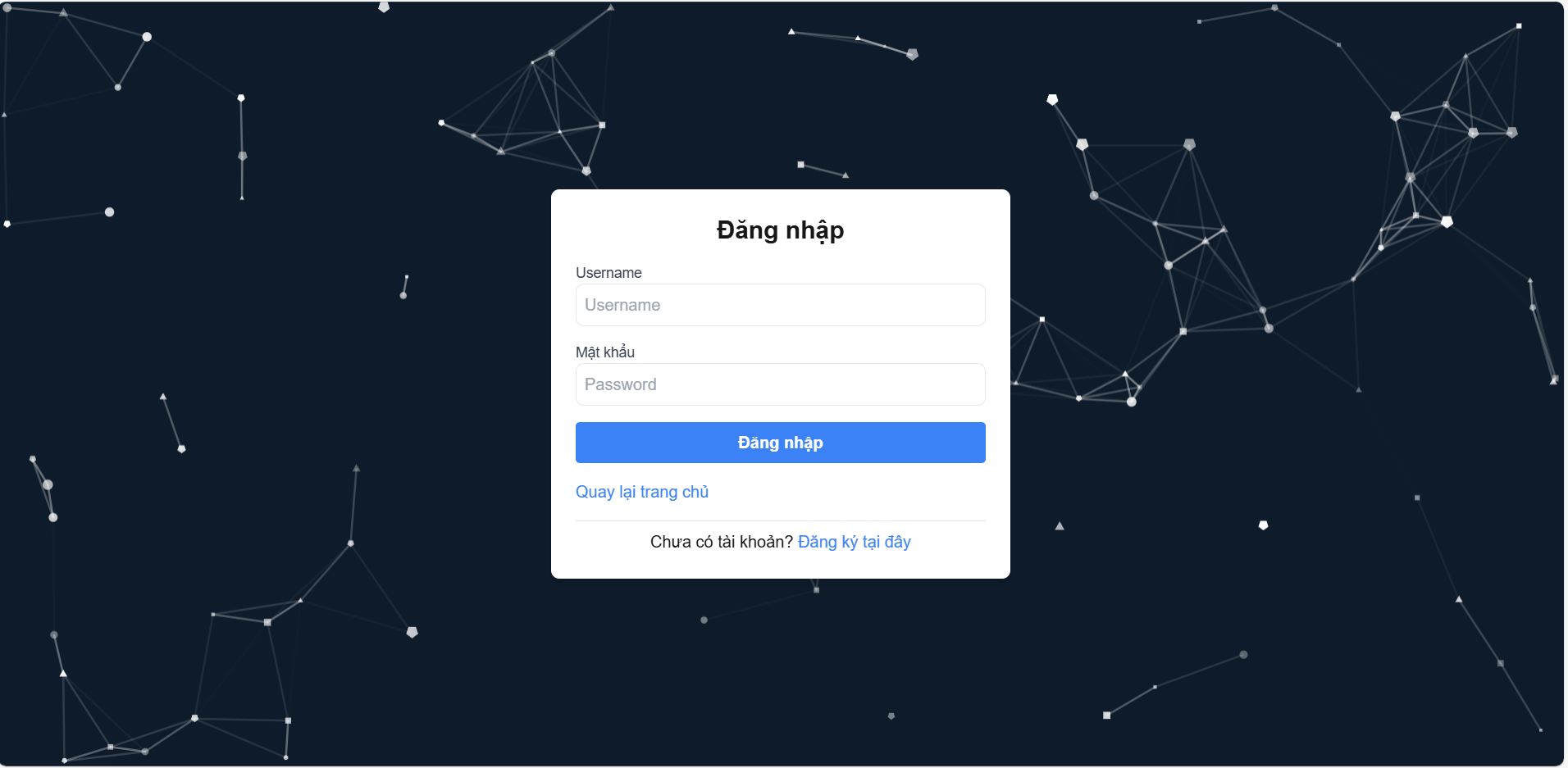
3. Bộ lọc này sẽ kiểm bắt parse token ra để kiểm tra ROLE của người dùng thông qua UserDetail có quyền với yêu cầu hay không.

4. Nếu tất cả các kiểm tra đều hợp lệ, yêu cầu sẽ được chuyển đến Controller để xử lý và trả về phản hồi cho người dùng, Nếu token không hợp lệ hoặc yêu cầu không khớp với điều kiện bảo mật, hệ thống sẽ trả về mã lỗi 403 hoặc thông báo không được phép truy cập.

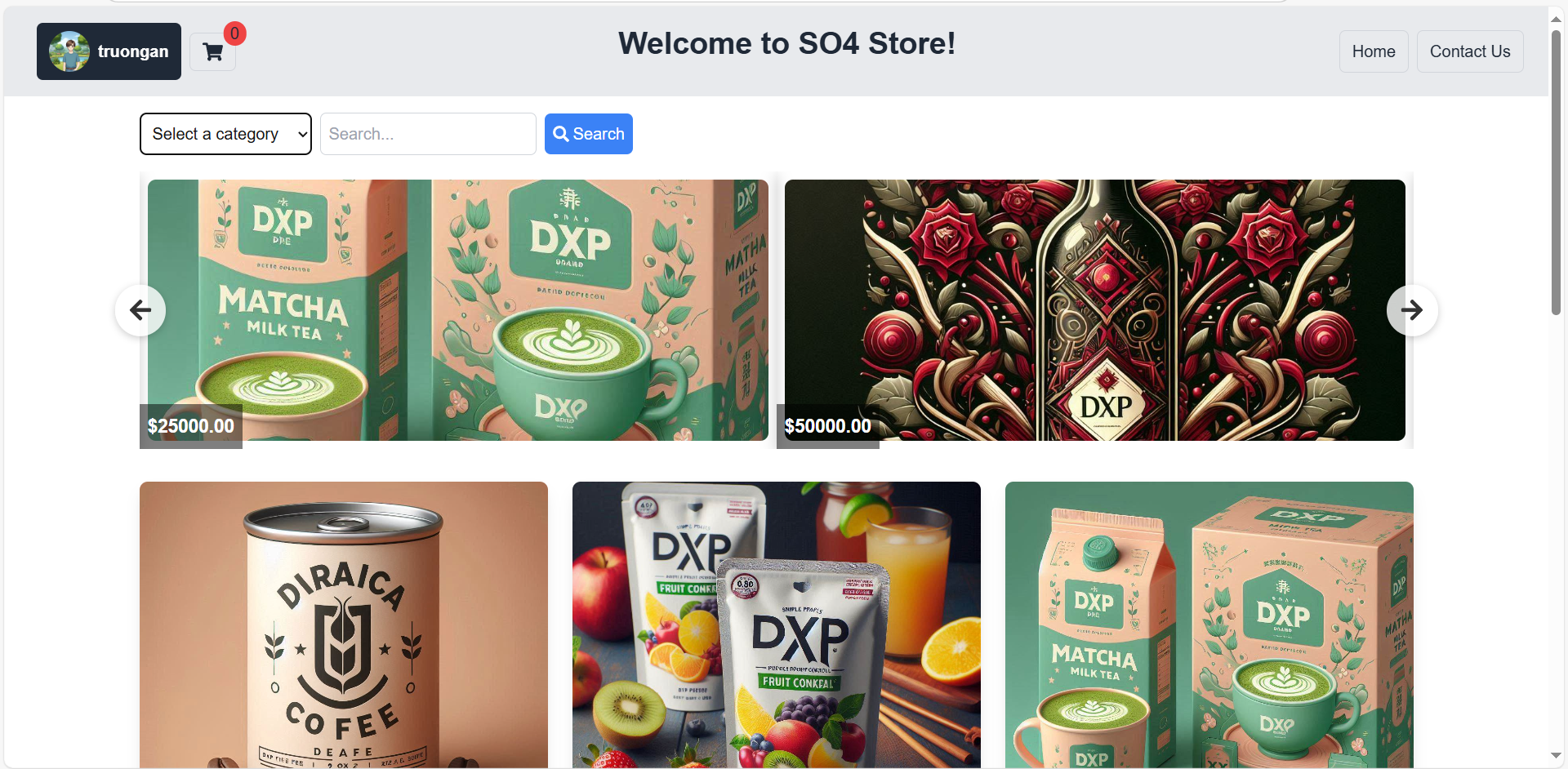
## Kết quả thực hiện

### Đăng nhập vào hệ thống:

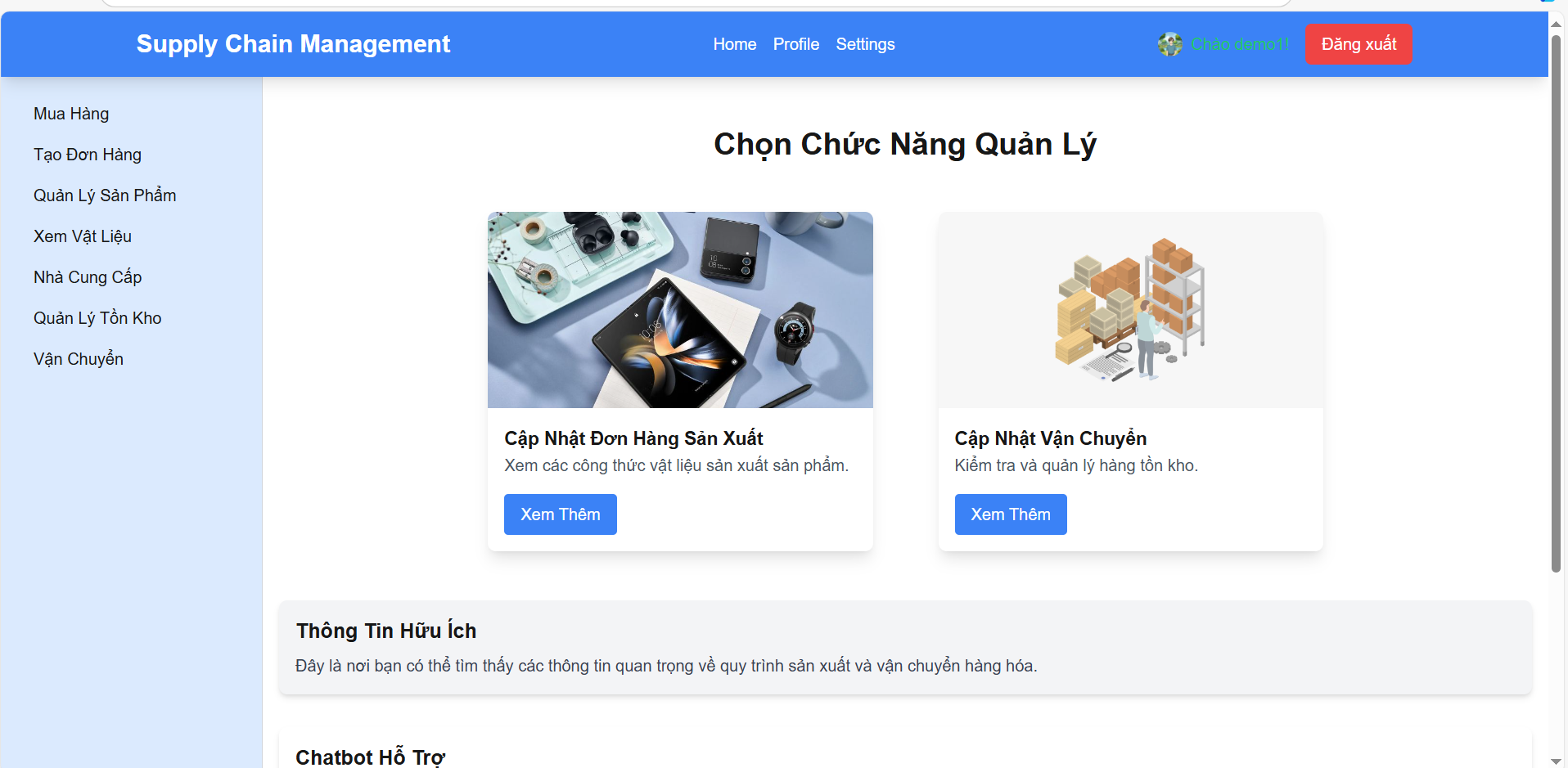
Hệ thống sẽ có 2 trang dành cho nhân viên và khách hàng, khi người dùng đăng nhập hệ thống sẽ tự động chuyển người dùng đến trang của khách hàng hoặc nhân viên dựa trên quyền.



Hình . Trang đăng nhập cho nhân viên và khách hàng



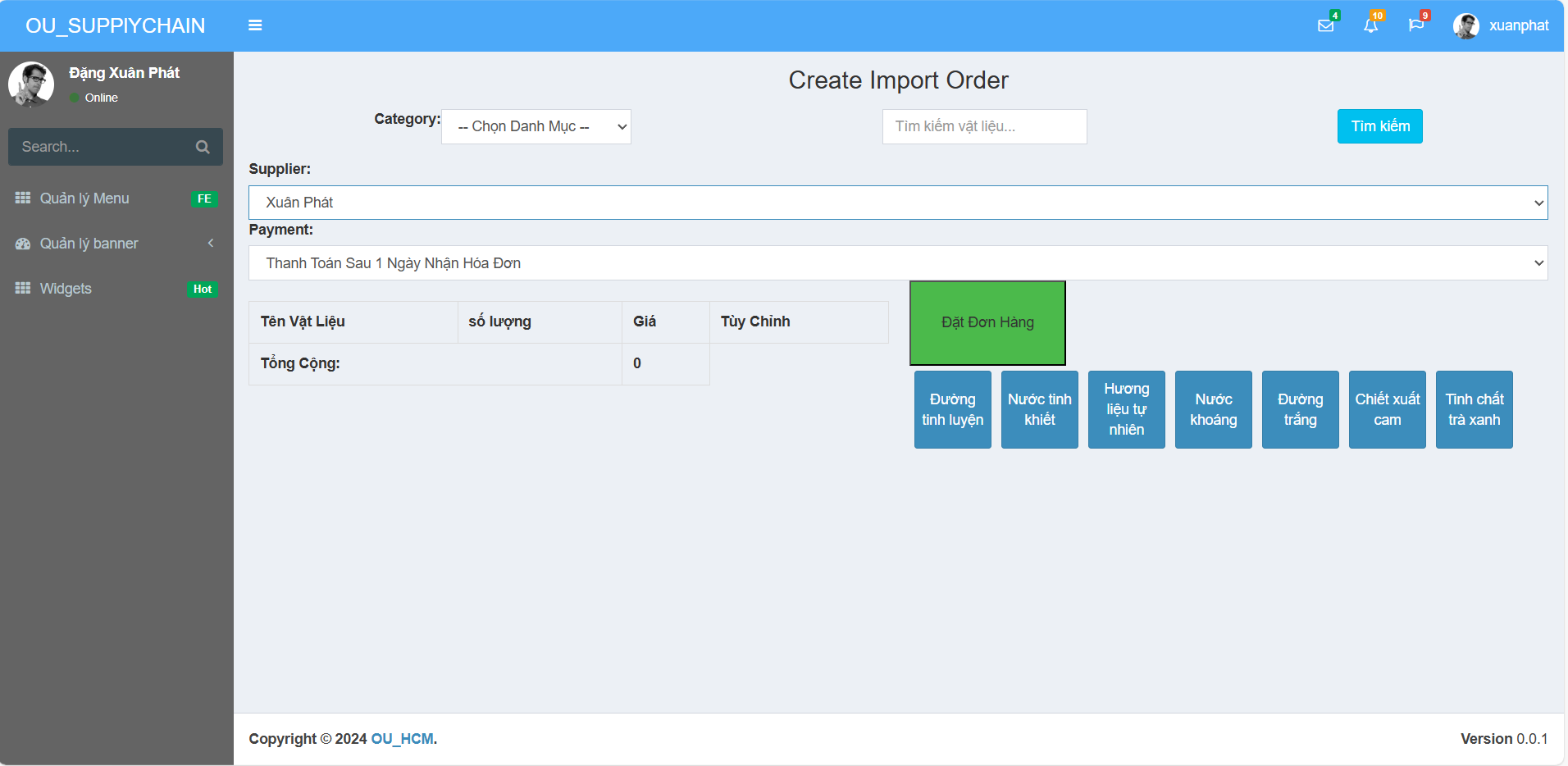
Hình . Giao diện khi khách hàng đăng nhập thành công



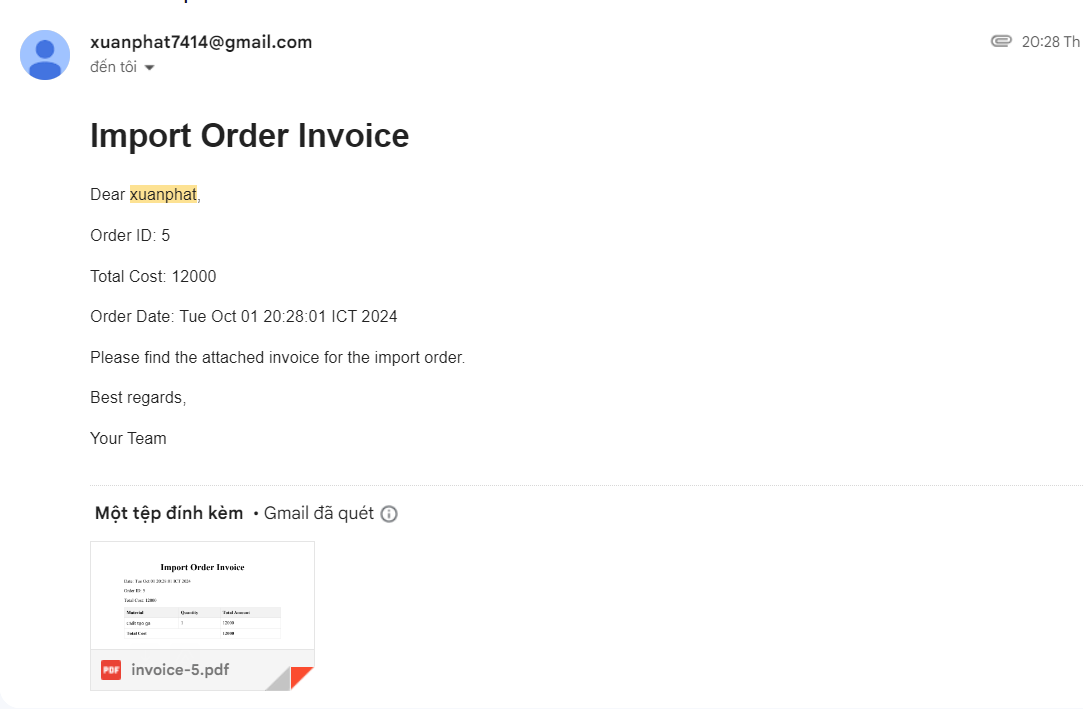
Hình . Giao diện khi đăng nhập với vai trò là nhân viên

### Đặt hàng từ nhà cung cấp

Khi truy cập vào ứng dụng để yêu cầu đặt hàng các nguyên liệu từ nhà cung cấp người dùng là (Admin, Quản lý) sẽ phải chọn nhà cung cấp thì mới hiện sản phẩm để có thể thực hiện chọn, đảm bảo được mỗi một đơn hàng chỉ được tạo từ một nhà cung cấp, sau khi xác nhận đơn hàng hệ thống sẽ tự động gửi mail đến cho nhà cung cấp.



Hình . Giao diện trang đặt hàng từ nhà cung

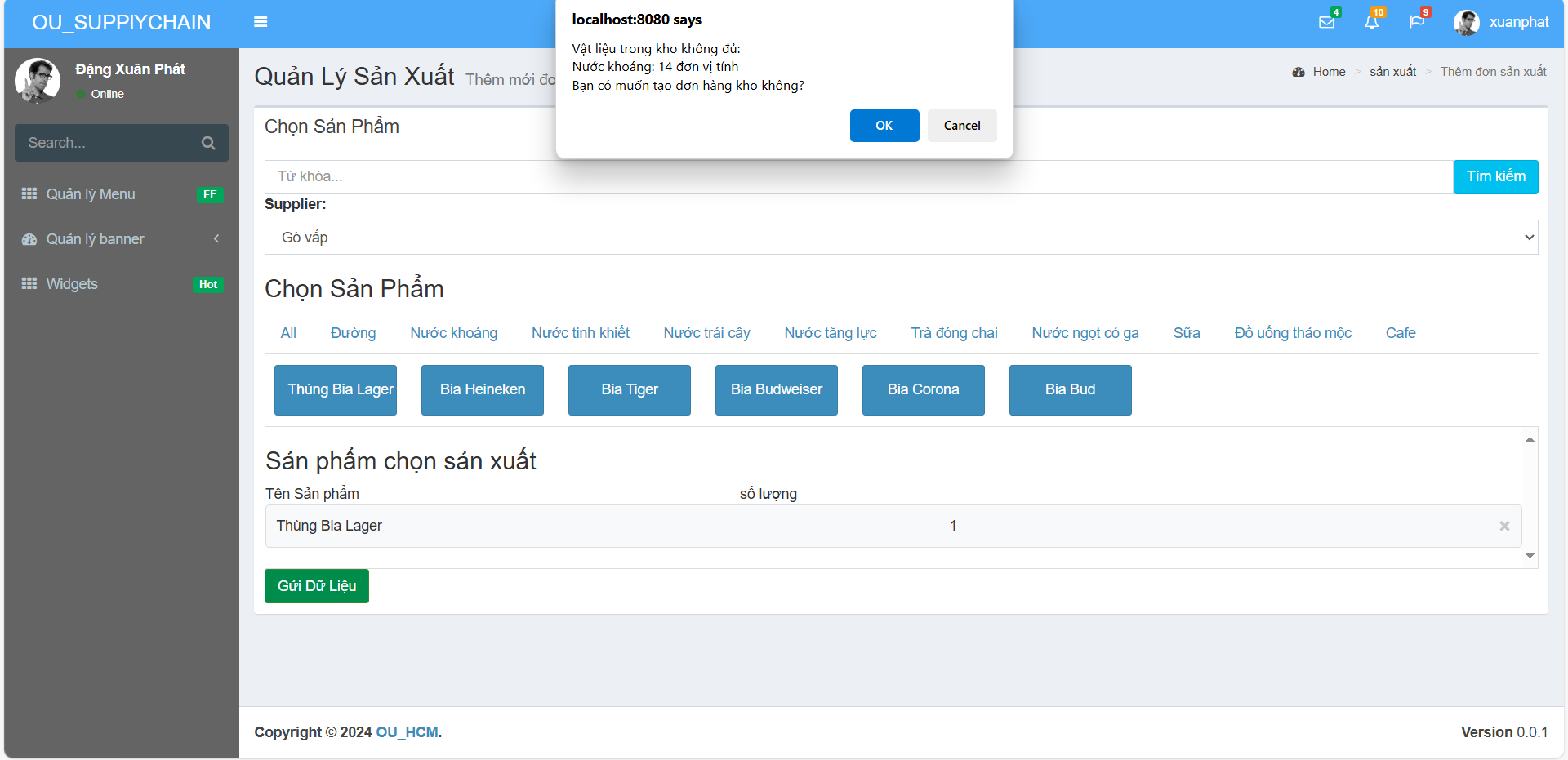


Hình . Mail được gửi khi hoàn tất đặt hàng từ nhà cung cấp

### Tạo đơn hàng sản xuất

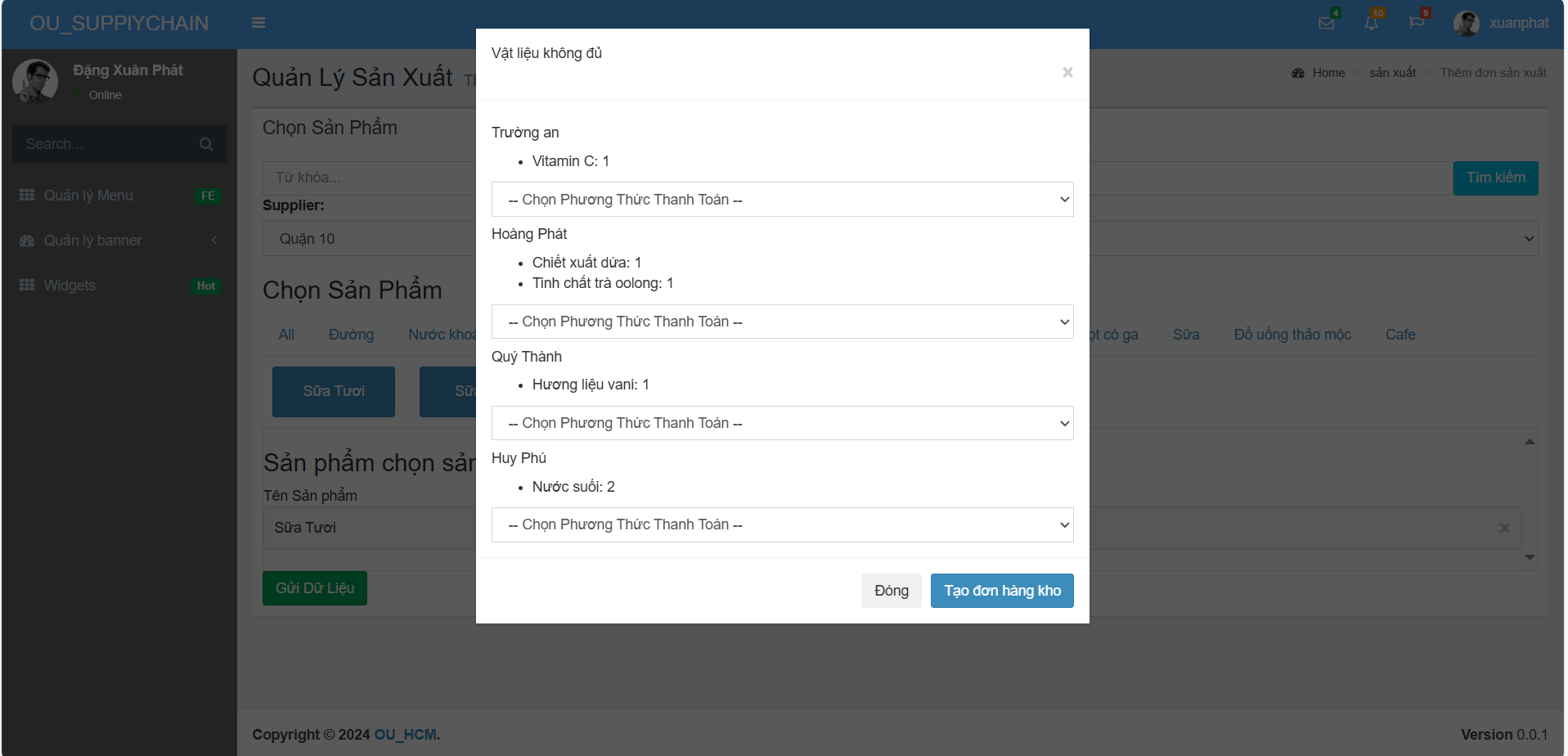
Hệ thống cho phép người dùng (Admin, Quản lý) tạo các đơn hàng sản xuất để yêu cầu xuống các kho hàng để tiến hành sản xuất các sản phẩm, sau khi tạo đơn sản xuất hệ thống sẽ kiểm tra các vật liệu để tạo ra sản phẩm hiện tồn trong kho có đủ không nếu không đủ, hệ thống sẽ hỗ trợ người dùng đặt đơn hàng từ nhiều hàng cung cấp nếu muốn và sẽ lưu trữ đơn hàng đồng thời giữ lại các vật liệu đủ cho đơn hàng để tránh xung đột khi quản lý tạo các đơn hàng tiếp theo.

Khi tạo đơn sản xuất hệ thống sẽ tiến hành kiểm tra sản phẩm thông qua công thức được tạo và lưu trên hệ thống để kiểm tra vào kho hàng xem mặt hàng cho đủ không nếu thiếu sẽ xuất ra thống báo như sau:



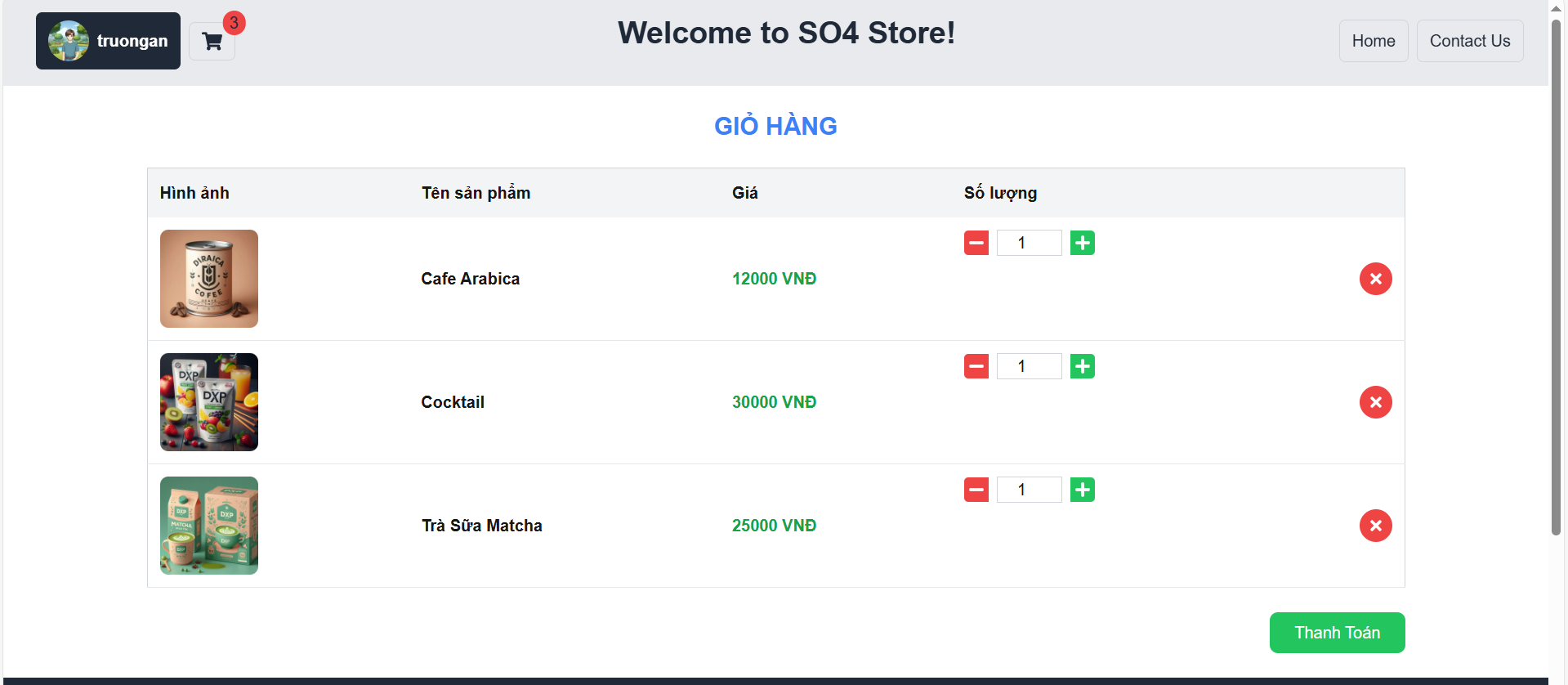
Hình . Giao diện tạo đơn hàng sản xuất

Hệ thống hỗ trợ cho người dùng (Admin, Quản lý) đặt hàng từ nhiều nhà cung cấp và lưu trữ đơn hàng và lưu trữ các vật liệu đủ lại:

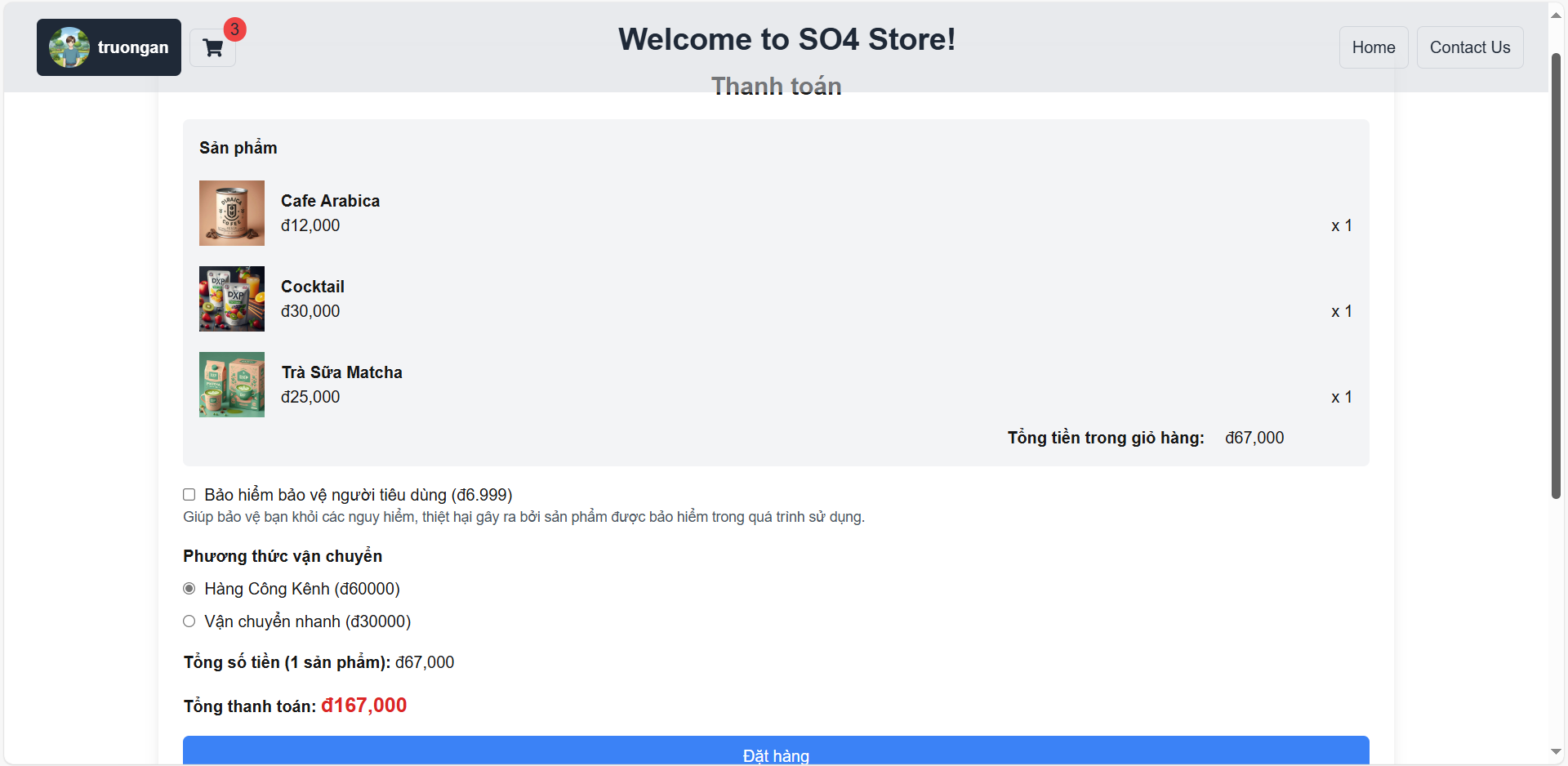


### Tạo đơn hàng xuất cho đại lý

Sau khi đăng nhập với quyền là đại lý, người dùng sẽ tiến hành chọn các sản phẩm để đặt hàng sau khi hoàn tất đơn hàng hệ thống sẽ tự tính khoảng cách đến địa chỉ của đại lý và tính các sản phẩm hiện có trong kho để tính các chi phí liên quan về vận chuyển và kho bãi, người dùng cho thể chọn các phương thức giao hàng khác nhau nếu muốn, sau khi xác nhận các chi phí người dùng sẽ chuyển tới trang để thanh toán trực tuyến với VnPay.



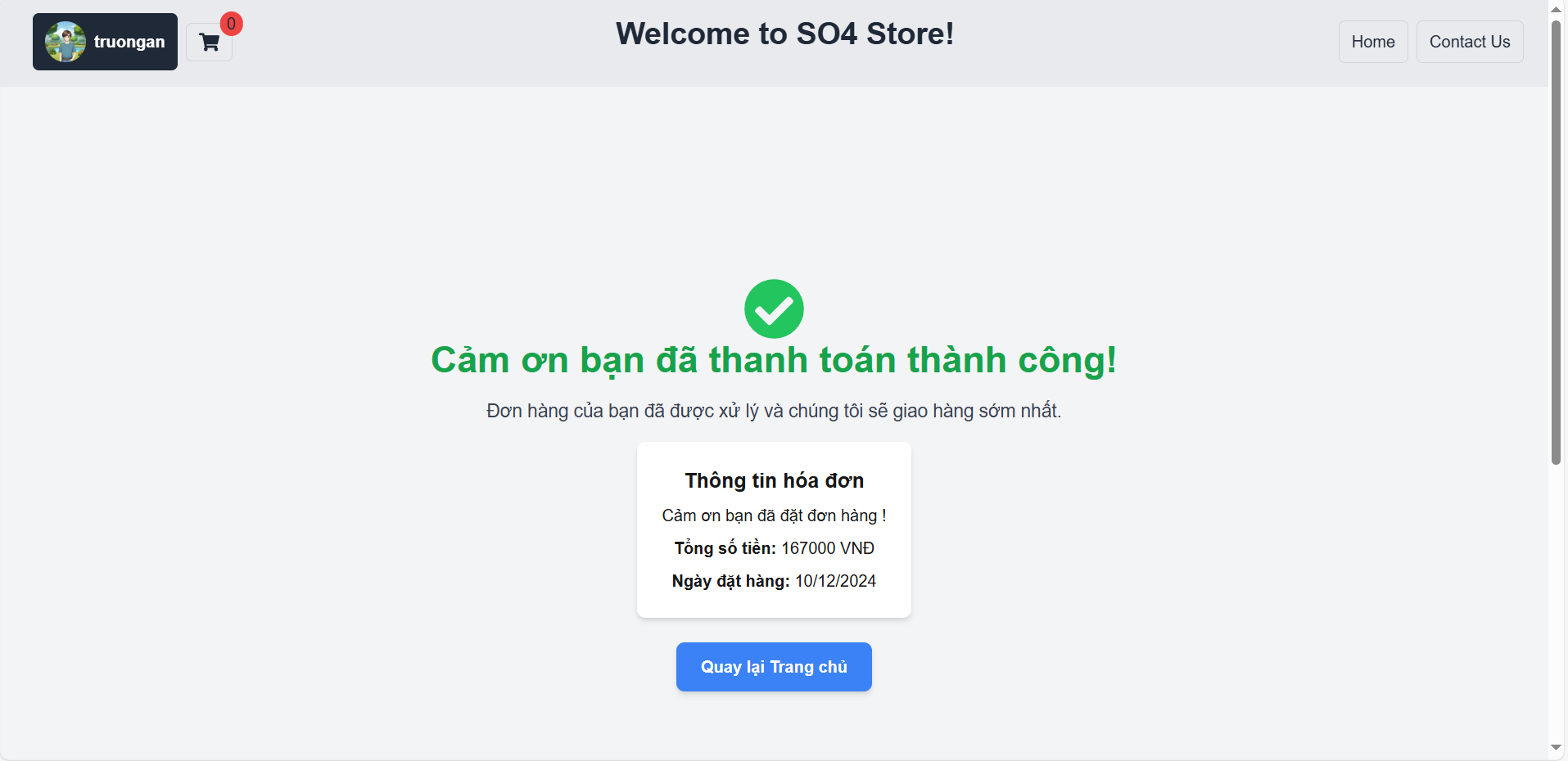
Hình . Giao diện giỏ hàng



Hình . Giao diện xác nhận đơn hàng

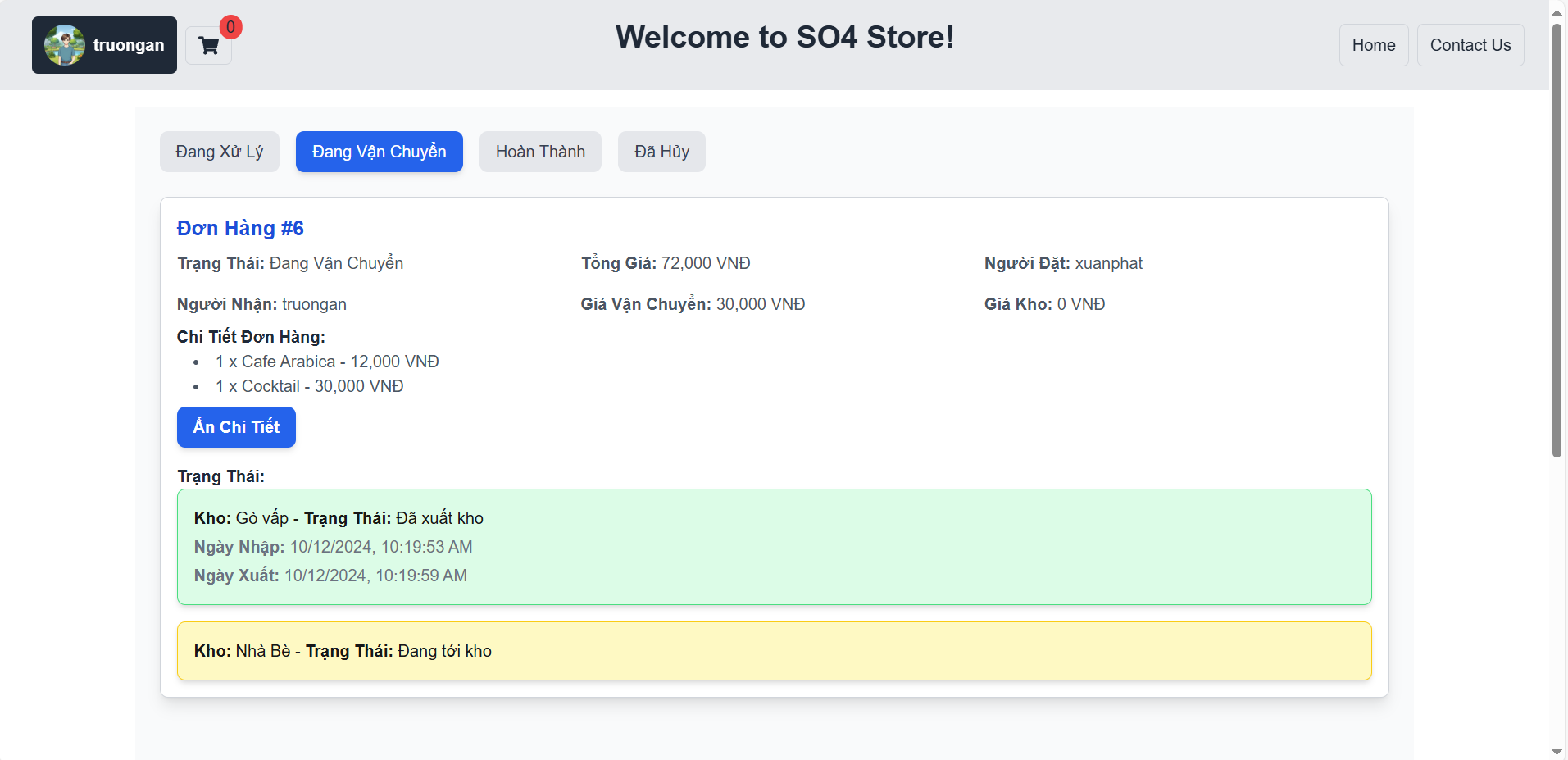


Hình . Giao diện thanh toán với cổng VnPay



Hình . Giao diện sau khi thanh toán thành công

Sau khi tạo hoàn tất hóa đơn đại lý có thể truy cập để xem trạng thái đơn hàng hiện đang ở đâu nhằm nắm bắt thông tin có thể nhận hàng kịp lúc:



Hình . xem chi tiết đơn hàng

### Dự đoán doanh số sản phẩm

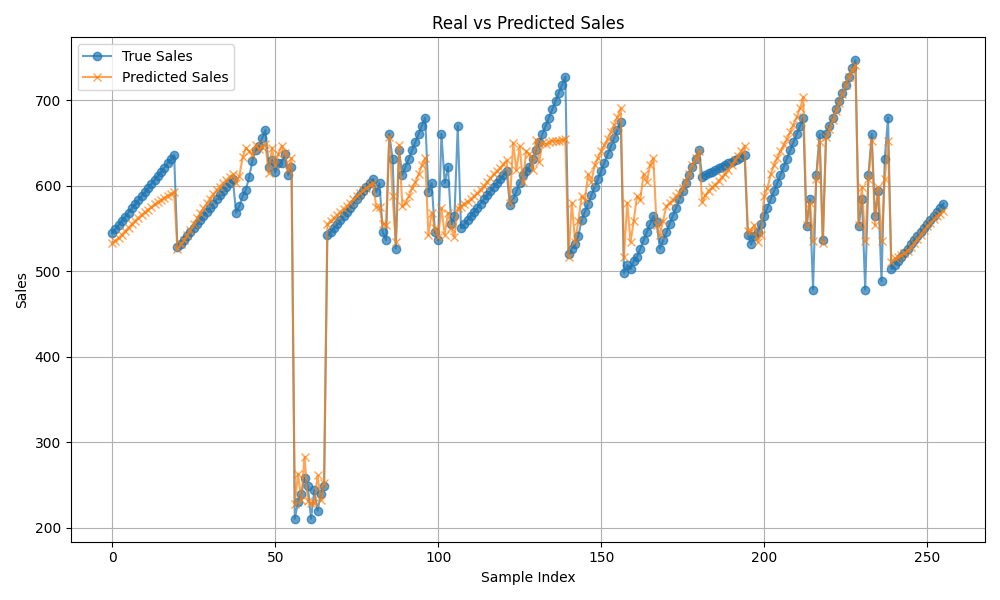
**Kết quả đánh giá mô hình:**

* + **Test Loss: 0.0637**
  + **Accuracy: 74.61% (với chỉ số khắt khe là 10%)**

**Báo cáo Đánh giá:**

* + **MAE (19.30):** Trung bình mỗi giá trị dự đoán sai lệch khoảng 19.30 đơn vị so với giá trị thực tế.
  + **RMSE (26.42):** Mô hình có sai số căn bậc hai là 26.42, chỉ ra rằng các sai số lớn hơn có ảnh hưởng đáng kể đến kết quả.
  + **R² Score (0.9065):** Mô hình giải thích được khoảng 90.65% sự biến động trong dữ liệu. Đây là một chỉ số rất tốt, cho thấy mô hình của bạn đang hoạt động khá hiệu quả trong việc dự đoán doanh số.
* Vì mô hình này thuộc loại **hồi quy**, vì đầu ra của mô hình dự đoán một giá trị liên tục, đó là doanh số sản phẩm tại một thời điểm nào đó.

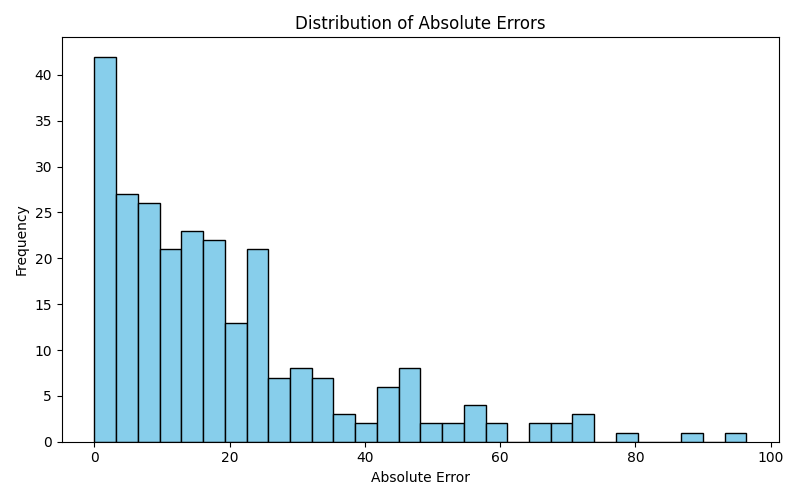
**Biểu đồ so sánh giữa Doanh số thực tế và Dự đoán:**

* Hình ảnh đầu tiên thể hiện sự so sánh giữa doanh số thực tế và dự đoán từ mô hình. Các điểm biểu diễn doanh số thực tế (chấm màu xanh) và dự đoán (dấu sao màu cam) cho thấy độ chính xác của mô hình trong việc dự đoán doanh số qua các chỉ số cụ thể. 

Hình . Biểu đồ so sánh giá trị thực với giá trị đánh giá

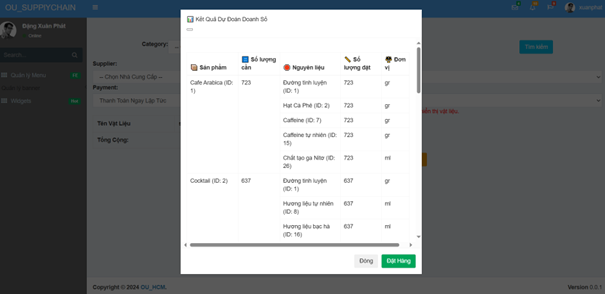
**Phân bố sai số tuyệt đối (Absolute Error):**

* Hình ảnh thứ hai mô tả sự phân bố của sai số tuyệt đối giữa giá trị dự đoán và giá trị thực tế. Biểu đồ cho thấy phần lớn các sai số nằm trong phạm vi thấp, chứng tỏ mô hình đã dự đoán khá chính xác.



Hình . Biểu đồ phân bổ giá trị sai số

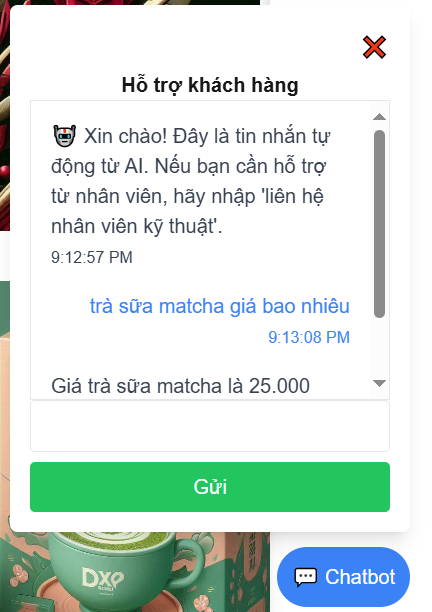
**Dự toán chi tiết sản phẩm trong hệ thống:** Hình ảnh cuối cùng cho thấy giao diện hệ thống sử dụng để theo dõi và dự đoán doanh số sản phẩm. Các thông tin chi tiết về nguyên liệu và số lượng đặt hàng được hiển thị rõ ràng để cung cấp cái nhìn tổng quan về quá trình quản lý doanh thu.



Hình . Tính năng tích hợp vào hệ thống

### Chatbot

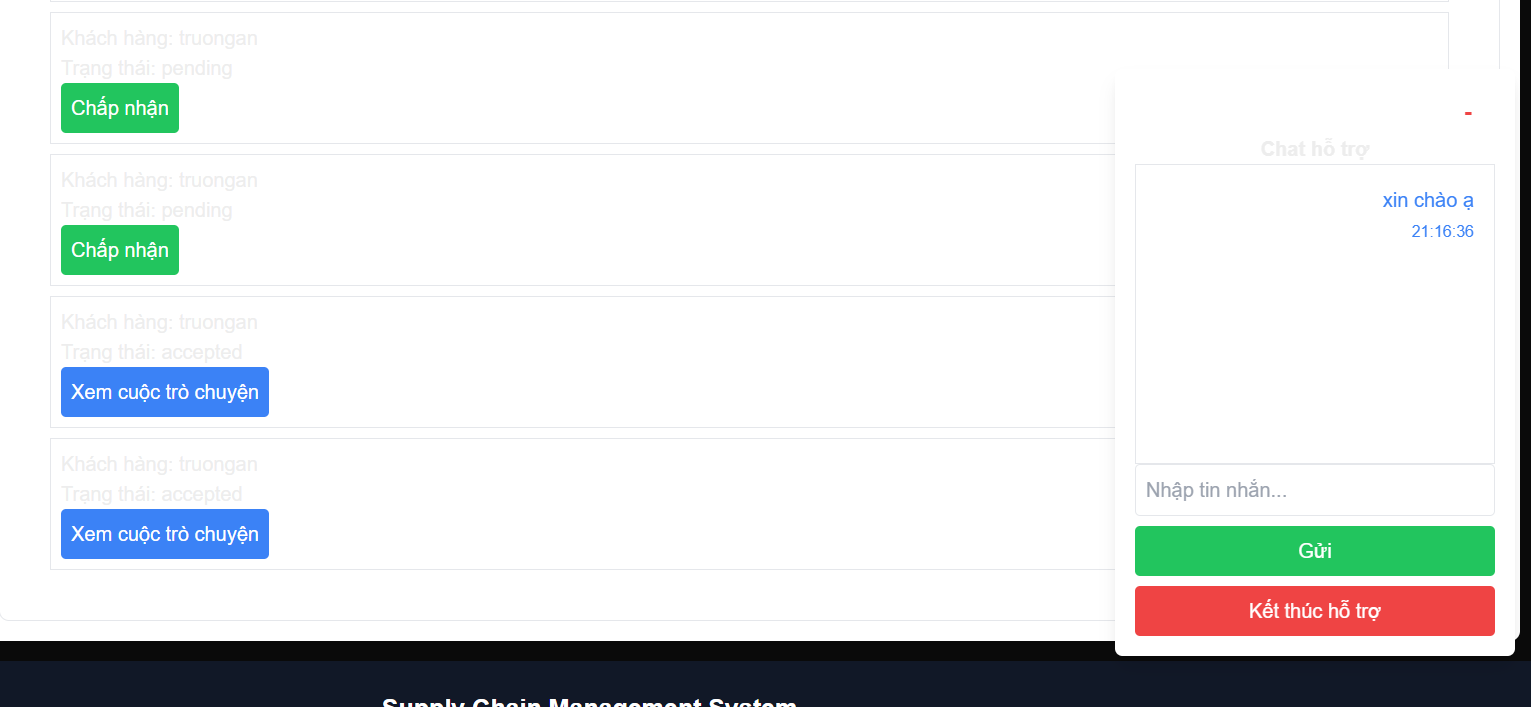
Hệ thống tích hợp Chatbot với AI được train với framework Rasa sẽ trả lời với các câu hỏi cơ bản về giá sản phẩm hay thông tin chi tiết khác.



Hình . Chat với AI

Nếu các câu hỏi liên quan đến kỹ thuật chatbot sẽ hiển thị form cho người dùng lựa chọn để kết nối với hệ thống.

Hệ thống tích hợp form thông báo cho nhân viên biết rằng là có khách hàng đang cần hỗ trợ kỹ thuật và một nhân viên có thể tư vấn cho nhiều khách hàng cùng lúc, hội thoại sẽ được kết thúc và xóa khi bấm vào nút kết thúc hỗ tr



Hình . Chat ở trang chủ nhân viên

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Trong quá trình nghiên cứu và phát triển, tôi đặt mục tiêu xây dựng một hệ thống quản lý chuỗi cung ứng toàn diện nhằm tối ưu hóa các chức năng quan trọng trong doanh nghiệp. Mục tiêu chính là giảm thiểu các công việc thủ công, nâng cao hiệu suất vận hành và hỗ trợ doanh nghiệp thích ứng nhanh với những thay đổi trong môi trường kinh doanh.

Sau 10 tuần thực hiện, hệ thống đã được hoàn thiện ở phiên bản đầu tiên. Phần backend sử dụng công nghệ Spring Boot, giúp đảm bảo tính ổn định và mở rộng linh hoạt. Phần frontend phát triển bằng NextJS, hỗ trợ đồng thời Server-Side Rendering (SSR) và Client-Side Rendering (CSR), giúp cải thiện tốc độ tải trang, tăng trải nghiệm người dùng và hỗ trợ tối ưu hóa SEO.

Để đảm bảo môi trường triển khai đồng nhất và tiện lợi, hệ thống đã tích hợp Docker. Redis cũng được sử dụng để tối ưu tốc độ truy xuất dữ liệu, giảm thời gian phản hồi cho người dùng.

Ngoài ra, hệ thống còn tích hợp các công nghệ trí tuệ nhân tạo (AI) tiên tiến, cụ thể:

Mô hình dự báo doanh thu bằng PyTorch sử dụng kiến trúc LSTM kết hợp Attention, cho phép dự đoán nhu cầu sản phẩm một cách chính xác, từ đó hỗ trợ lên kế hoạch sản xuất và nhập kho.

Chatbot Rasa giúp tự động hóa việc hỗ trợ khách hàng, trả lời nhanh các câu hỏi cơ bản và hỗ trợ theo dõi đơn hàng.

Bên cạnh đó, hệ thống cũng đã hoàn thiện chức năng quản lý đơn hàng, cho phép khách hàng theo dõi trạng thái vận chuyển đơn hàng trực tiếp trên bản đồ, giúp nâng cao trải nghiệm và tăng tính minh bạch trong giao dịch.

Quy trình sản xuất sản phẩm cũng được tối ưu hóa thông qua việc phân tích đơn hàng dự đoán trước số lượng sản xuất cần thiết, giảm thiểu tình trạng tồn kho dư thừa hoặc thiếu hụt.

Thông qua dự án này, tôi đã có cơ hội tiếp cận và áp dụng nhiều công nghệ mới, đồng thời trau dồi kỹ năng thiết kế hệ thống phức tạp và triển khai dự án thực tế một cách hiệu quả.

## Hướng phát triển

Trong tương lai, hệ thống có thể được mở rộng thêm nhiều chức năng nhằm tăng cường hiệu quả vận hành và trải nghiệm khách hàng.

### Tích hợp AI nhận diện hình ảnh trong kiểm kê kho

Hiện nay việc kiểm kê kho chủ yếu dựa vào việc ghi chép và nhập liệu thủ công, gây tốn thời gian và dễ xảy ra sai sót. Để cải thiện điều này, hệ thống có thể tích hợp thêm AI nhận diện hình ảnh.

Cụ thể, nhân viên kho có thể chụp ảnh báo cáo kiểm kê, hệ thống sẽ tự động nhận diện số liệu và cập nhật vào cơ sở dữ liệu. Việc này sẽ giúp tiết kiệm thời gian nhập liệu, giảm tỷ lệ sai sót và tăng tốc độ cập nhật tồn kho.

Các công nghệ có thể áp dụng bao gồm EasyOCR, PaddleOCR hoặc xây dựng mạng nơ-ron convolutional (CNN) chuyên xử lý phiếu kiểm kê.

### Quản lý hụt hàng và thất thoát kho

Hụt hàng trong kho do thất thoát hoặc hao hụt là vấn đề phổ biến. Để giải quyết, hệ thống có thể triển khai tính năng AI giám sát số lượng sản phẩm qua ảnh định kỳ, tự động so sánh với số liệu hệ thống và cảnh báo khi có chênh lệch lớn.

Điều này sẽ giúp doanh nghiệp kiểm soát kho hàng chặt chẽ hơn, giảm thiểu tổn thất và nâng cao độ tin cậy trong vận hành.

### Hiển thị kho dự kiến vận chuyển cho khách hàng

Một trong những nâng cấp quan trọng cho trải nghiệm khách hàng là cho phép khách hàng xem trước các kho trung chuyển mà đơn hàng sẽ đi qua. Khi đặt hàng, khách hàng có thể lựa chọn nhận hàng trực tiếp tại kho gần nhất nếu có nhu cầu.

Điều này giúp giảm thời gian vận chuyển, tiết kiệm chi phí giao hàng, đồng thời tăng mức độ hài lòng của khách hàng.

Để thực hiện, hệ thống cần tích hợp công nghệ xác định vị trí địa lý, ví dụ như sử dụng Google Maps API hoặc thuật toán Haversine để tìm kho gần nhất theo vị trí khách hàng.

### Phân tích báo cáo tồn kho bằng hình ảnh cho khách hàng

Đối với các khách hàng lớn thường xuyên kiểm kê kho, hệ thống có thể hỗ trợ chức năng cho phép khách hàng chụp ảnh phiếu kiểm kê. Hệ thống sẽ tự động phân tích hình ảnh, trích xuất dữ liệu và điền vào báo cáo điện tử.

Chức năng này giúp khách hàng tiết kiệm thời gian nhập liệu thủ công, tăng tính chính xác và tự động hóa quy trình báo cáo kho hàng.

### Phân quyền mở rộng quản lý kho hàng

Hiện tại việc điều hành kho chủ yếu tập trung tại trung tâm. Để tăng tính linh hoạt, hệ thống sẽ phát triển thêm chức năng phân quyền cho các kho hàng địa phương, cho phép quản lý các hoạt động như:

* Đặt vật liệu nội bộ.
* Yêu cầu luân chuyển hàng hóa giữa các kho.
* Theo dõi trực tiếp số lượng hàng tồn.

Điều này giúp từng kho hàng có thể chủ động hơn trong việc vận hành, đồng thời giảm tải cho bộ phận quản lý trung tâm.

### Nâng cấp quản lý đơn hàng và bản đồ theo dõi

Trong giai đoạn tiếp theo, hệ thống sẽ tiếp tục tối ưu tính năng theo dõi đơn hàng theo thời gian thực. Ngoài việc hiển thị trạng thái đơn hàng, hệ thống sẽ bổ sung các chức năng:

* Hiển thị trực tiếp vị trí hiện tại của đơn hàng trên bản đồ.
* Thông báo tự động cho khách hàng khi đơn hàng thay đổi trạng thái.
* Dự đoán thời gian giao hàng dựa trên dữ liệu lịch sử và tình trạng giao thông thực tế.

Những tính năng này sẽ giúp khách hàng cảm thấy yên tâm và chủ động hơn trong việc nhận hàng, đồng thời tăng uy tín cho doanh nghiệp.

### Tối ưu quy trình sản xuất dựa trên dữ liệu dự báo

Dựa trên dữ liệu dự báo doanh số bằng AI, hệ thống có thể tối ưu quy trình sản xuất bằng cách:

* Lập kế hoạch sản xuất theo nhu cầu thực tế.
* Phân bổ nguồn lực hiệu quả hơn, tránh sản xuất dư thừa.
* Rút ngắn thời gian sản xuất và giảm tồn kho lưu kho lâu ngày.

## Thử thách và khó khăn

Trong quá trình triển khai các tính năng mới của hệ thống, ngoài những lợi ích đáng kể, cũng sẽ có không ít thử thách và khó khăn mà đội ngũ phát triển cần phải đối mặt. Dưới đây là tổng hợp các **lợi ích** và **thách thức** trong việc mở rộng các tính năng của hệ thống.

**Lợi ích:**

1. **Tăng hiệu quả công việc**: Việc tự động hóa các công đoạn như kiểm kê kho, dự báo doanh thu hay quản lý đơn hàng sẽ giảm thiểu tối đa các công việc thủ công, giúp nhân viên và doanh nghiệp tiết kiệm thời gian và nguồn lực.
2. **Cải thiện trải nghiệm người dùng**: Việc cho phép khách hàng chọn kho nhận hàng gần nhất, theo dõi trạng thái đơn hàng qua bản đồ, và tự động hóa báo cáo kho sẽ tạo ra một trải nghiệm liền mạch, dễ sử dụng, và tiện lợi hơn cho khách hàng.
3. **Nâng cao tính chính xác và giảm thiểu sai sót**: Việc tích hợp AI trong việc nhận diện hình ảnh báo cáo kho và kiểm kê kho sẽ giúp giảm thiểu các lỗi do nhập liệu thủ công, từ đó cải thiện độ chính xác của các báo cáo và quản lý kho hàng.
4. **Tăng tính linh hoạt và tự chủ trong vận hành**: Việc phân quyền quản lý kho và cho phép các kho tự quản lý các hoạt động của mình sẽ giúp hệ thống trở nên linh hoạt hơn và giảm tải cho bộ phận trung tâm, từ đó tăng cường khả năng phản ứng nhanh chóng với các tình huống phát sinh.

**Thách thức:**

1. **Đảm bảo chất lượng hình ảnh và dữ liệu**: Hệ thống nhận diện hình ảnh sẽ gặp phải vấn đề về chất lượng hình ảnh (mờ, sai sót, ánh sáng không đủ). Để giải quyết vấn đề này, cần phải phát triển thêm các thuật toán tiền xử lý hình ảnh mạnh mẽ.
2. **Tính chính xác trong nhận diện số liệu**: Khi sử dụng OCR (nhận diện ký tự quang học) hoặc AI để nhận diện số liệu từ hình ảnh, hệ thống cần phải có khả năng xử lý các trường hợp không chuẩn, như các chữ viết tay không rõ ràng hay các báo cáo có định dạng phức tạp.
3. **Khó khăn trong việc duy trì tính cập nhật và chính xác của dữ liệu thời gian thực**: Việc theo dõi và cập nhật dữ liệu kho hàng, trạng thái đơn hàng trên bản đồ đòi hỏi hệ thống phải hoạt động chính xác trong thời gian thực. Điều này đòi hỏi cơ sở hạ tầng vững chắc và khả năng quản lý dữ liệu hiệu quả.
4. **Chi phí và thời gian phát triển**: Việc phát triển các tính năng AI, như nhận diện hình ảnh và phân tích dữ liệu từ hình ảnh, sẽ đòi hỏi bộ dữ liệu lớn và nhiều thời gian để huấn luyện mô hình. Điều này có thể ảnh hưởng đến tiến độ và chi phí phát triển.
5. **Quản lý sự phân tán của dữ liệu kho hàng**: Với hệ thống phân quyền quản lý kho, việc đồng bộ hóa và quản lý dữ liệu giữa các kho hàng và bộ phận trung tâm là một thách thức lớn. Cần phải có các công cụ đồng bộ hóa mạnh mẽ để đảm bảo tính nhất quán và độ chính xác của dữ liệu.

## Tổng kết:

Thông qua việc phát triển hệ thống quản lý chuỗi cung ứng và các hướng phát triển trong tương lai, tôi nhận thấy rằng hệ thống có thể giúp doanh nghiệp tối ưu hóa quy trình làm việc, tăng cường khả năng phản hồi và cải thiện trải nghiệm khách hàng. Việc ứng dụng công nghệ AI và các công cụ quản lý hiện đại sẽ không chỉ giúp giảm thiểu chi phí, mà còn nâng cao năng suất và độ chính xác trong các hoạt động kho vận.

TÀI LIỆU THAM KHẢO

[1] NextJS. "NextJS Documentation." [Online]. Available: https://nextjs.org/. [Accessed: Oct. 12, 2024].

[2] Spring Boot. "Spring Boot Projects." [Online]. Available: https://spring.io/projects/spring-boot. [Accessed: Oct. 12, 2024].

[3] Docker. "Get Started with Docker." [Online]. Available: https://www.docker.com/get-started/. [Accessed: Oct. 12, 2024].

[4] Thymeleaf. "Thymeleaf Documentation." [Online]. Available: https://www.thymeleaf.org/. [Accessed: Oct. 12, 2024].

[5] Techmaster. "Bài viết về công nghệ." [Online]. Available: https://techmaster.vn/posts?category=buvqnin0k7qv59jeq3sg. [Accessed: Oct. 12, 2024].

PHỤ LỤC

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Số hiệu bảng** | **Tên bảng** | **Các trường** | **Mô tả** | **Khóa chính** | **Khóa ngoại** | **Trang** |
| Bảng 3.1 | User | Id: int (primary key), Name: char(40), Role: char(20) | Lưu thông tin người dùng | Id | Không có | 71 |
| Bảng 3.2 | OrderExport | OrderId: int (primary key), CustomerId: int | Lưu thông tin đơn hàng xuất | Id | CustomerId tham chiếu đến User (Id) | 71 |
| Bảng 3.3 | Agency | AgencyId: int (primary key), UserId: int | Lưu thông tin đại lý | Id | UserId tham chiếu đến User (Id) | 72 |
| Bảng 3.4 | Material | MaterialId: int (primary key), Name: char(50), Stock: int | Lưu thông tin nguyên vật liệu | Id | Không có | 72 |
| Bảng 3.5 | Product | ProductId: int (primary key), Name: char(50), Price: float | Lưu thông tin sản phẩm | Id | Không có | 73 |
| Bảng 3.6 | DetailExport  Order | OrderId: int, ProductId: int, Quantity: int | Chi tiết đơn hàng xuất, bao gồm sản phẩm và vật liệu | Id | OrderId tham chiếu đến OrderExport  (OrderId), ProductId tham chiếu đến Product  (ProductId) | 73 |
| Bảng 3.7 | Cost | CostId: int (primary key), OrderId: int, Amount: float | Lưu thông tin chi phí cho đơn hàng | Id | OrderId tham chiếu đến OrderExport  (OrderId) | 74 |
| Bảng 3.8 | Transport  Partner | PartnerId: int (primary key), Name: char(50) | Lưu thông tin các đối tác vận chuyển | Id | Không có | 74 |
| Bảng 3.9 | Performance  Valuation | ValuationId: int (primary key), AgencyId: int, Score: float | Đánh giá hiệu suất hoạt động của đại lý | Id | AgencyId tham chiếu đến Agency  (AgencyId) | 75 |
| Bảng 3.10 | Inventory | InventoryId: int (primary key), MaterialId: int, Quantity: int | Quản lý tồn kho các nguyên vật liệu | Id | MaterialId tham chiếu đến Material  (MaterialId) | 75 |
| Bảng 3.11 | OrderDetails | DetailId: int (primary key), OrderId: int, Item: char(50) | Chi tiết từng mục trong đơn hàng | Id | OrderId tham chiếu đến OrderExport  (OrderId) | 76 |
| Bảng 3.12 | Supplier  Performance | PerformanceId: int (primary key), SupplierId: int, ManagerId: int | Đánh giá hiệu suất nhà cung cấp | Id | SupplierId tham chiếu đến Agency  (AgencyId), ManagerId tham chiếu đến User (Id) | 76 |
| Bảng 3.13 | MaterialPrice | PriceId: int (primary key), MaterialId: int, Price: float, Date: date | Lưu lịch sử giá của vật liệu | Id | MaterialId tham chiếu đến Material  (MaterialId) | 77 |
| Bảng 3.14 | Detail  ImportOrder | ImportOrderId: int, MaterialId: int, Quantity: int | Chi tiết các đơn hàng nhập | Id | ImportOrderId tham chiếu đến OrderExport  (OrderId), MaterialId tham chiếu đến Material  (MaterialId) | 77 |
| Bảng 3.15 | Warehouse | WarehouseId: int (primary key), Name: char(50), Location: char(100) | Thông tin kho lưu trữ | Id | Không có | 78 |
| Bảng 3.16 | Stock  Transaction | TransactionId: int (primary key), MaterialId: int, WarehouseId: int, Quantity: int | Giao dịch kho hàng | Id | MaterialId tham chiếu đến Material  (MaterialId), WarehouseId tham chiếu đến Warehouse  (WarehouseId) | 78 |
| Bảng 3.17 | Customer  Order | OrderId: int (primary key), CustomerId: int, Status: char(20) | Lưu thông tin các đơn hàng của khách hàng | id | CustomerId tham chiếu đến User (Id) | 78 |
| Bảng 3.18 | OrderPayment | PaymentId: int (primary key), OrderId: int, Amount: float, PaymentMethod: char(20) | Thông tin thanh toán đơn hàng | Id | OrderId tham chiếu đến CustomerOrder (OrderId) | 78 |
| Bảng 3.19 | Delivery  Schedule | ScheduleId: int (primary key), OrderId: int, DeliveryDate: date, Status: char(20) | Lịch trình giao hàng | Id | OrderId tham chiếu đến CustomerOrder (OrderId) | 78 |
| Bảng 3.20 | Warehouse  Inventory | InventoryId: int (primary key), WarehouseId: int, MaterialId: int, Quantity: int | Tồn kho theo từng kho | Id | WarehouseId tham chiếu đến Warehouse (WarehouseId), MaterialId tham chiếu đến Material (MaterialId) | 78 |
| Bảng 3.21 | Employee | EmployeeId: int (primary key), WarehouseId: int, Name: char(40) | Nhân viên kho hàng | Id | WarehouseId tham chiếu đến Warehouse (WarehouseId) | 78 |
| Bảng 3.22 | Order  Manufacture | ManufactureId: int (primary key), ProductId: int, Quantity: int, Status: char(20) | Lưu thông tin đơn sản xuất | Id | ProductId tham chiếu đến Product (ProductId) | 78 |
| Bảng 3.23 | DetailOrder  Manufacture | ManufactureId: int, MaterialId: int, Quantity: int | Chi tiết nguyên vật liệu cho đơn sản xuất | Id | ManufactureId tham chiếu đến OrderManufacture (ManufactureId), MaterialId tham chiếu đến Material (MaterialId) | 78 |
| Bảng 3.24 | Partner | PartnerId: int (primary key), Name: char(50), Type: char(20) | Lưu thông tin đối tác | Id | Không có | 78 |
| Bảng 3.25 | Partner  Agency | PartnerId: int, AgencyId: int | Quan hệ đối tác - đại lý | Id | PartnerId tham chiếu đến Partner (PartnerId), AgencyId tham chiếu đến Agency (AgencyId) | 78 |
| Bảng 3.26 | Partner  Transport | PartnerId: int, TransportId: int | Quan hệ đối tác - nhà vận chuyển | Id | PartnerId tham chiếu đến Partner  (PartnerId), TransportId tham chiếu đến TransportPartner  (PartnerId) | 78 |
| Bảng 3.27 | Logistic | LogisticId: int (primary key), OrderId: int, WarehouseId: int, DeliveryDate: date | Lưu thông tin về giao hàng | Id | OrderId tham chiếu đến CustomerOrder  (OrderId), WarehouseId tham chiếu đến Warehouse  (WarehouseId) | 78 |