**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: PHÂN TÍCH VÀ XÂY DỰNG HỆ THỐNG**

**ĐẶT ĐỒ ĂN TRỰC TUYẾN**

**CHO SINH VIÊN NỘI TRÚ TẠI KÝ TÚC XÁ UTC2**

Giảng viên hướng dẫn: ThS. TRẦN PHONG NHÃ

Sinh viên thực hiện: NGUYỄN HOÀNG MINH CHÂU

Lớp : CQ.57.CNTT

Khoá :57

Tp. Hồ Chí Minh, năm 2022

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: PHÂN TÍCH VÀ XÂY DỰNG HỆ THỐNG**

**ĐẶT ĐỒ ĂN TRỰC TUYẾN**

**CHO SINH VIÊN NỘI TRÚ TẠI KÝ TÚC XÁ UTC2**

Giảng viên hướng dẫn: ThS. TRẦN PHONG NHÃ

Sinh viên thực hiện: NGUYỄN HOÀNG MINH CHÂU

Lớp : CQ.57.CNTT

Khoá :57

Tp. Hồ Chí Minh, năm 2022

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI **CỘNG HÒA XÃ HỘI CHỦ NGHIÃ VIỆT NAM**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH** Độc lập – Tự do – Hạnh phúc

**NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP**

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-------\*\*\*-------

**Mã sinh viên: 5751071002 Họ tên SV: NGUYỄN HOÀNG MINH CHÂU**

**Khóa: 57** **Lớp:** **CQ.57.CNTT**

1. **Tên đề tài**

**“Phân tích và xây dựng hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại ký túc xá UTC2”**

1. **Mục đích, yêu cầu**

**Mục đích:**

* Phân tích bài toán và xây dựng một hệ thống quản lý đặt đồ ăn trực tuyến cho sinh viên đang ở nội trú tại khu ký túc xá UTC2.

**Yêu cầu:**

* Tìm hiểu và phân tích bài toán về nghiệp vụ bán đồ ăn tại khu ký túc xá dựa trên dữ liệu thực tế thu thập được
* Nghiên cứu về cách xây dựng một trang web đặt và bán đồ ăn trực tuyến thực tế
* Tìm hiểu về Spring Boot để xây dựng máy chủ xử lý (Back-End)
* Tìm hiểu về Angular để xây dựng một trang web bán hàng trực tuyến (Front-End)
* Tìm hiểu về giao thức Restful API để xây dựng cầu nối giao tiếp giữ máy chủ xử lý (Back-End) và trang web bán hàng (Front-End)
* Áp dụng để giải quyết bài toán đặt ra và xây dựng thành công một hệ thống riêng cho khu ký túc xá

1. **Nội dung và phạm vi đề tài**

**Nội dung đề tài:**

* Giới thiệu tổng quan về Spring Boot, Restful API và Angular
* Giới thiệu về mô hình quản lý bán hàng trực tuyến
* Nghiên cứu, phân tích bài toán thực tế và xây dựng hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại khu ký túc xá UTC2

**Phạm vi đề tài:**

* Nghiên cứu mô hình quản lý bán hàng trực truyến và áp dụng vào bài toán thực tế cho sinh viên nội trú tại khu ký túc xá UTC2

1. **Công nghệ, công cụ và ngôn ngữ lập trình**

**Công nghệ và ngôn ngữ lập trình:**

* Spring Boot
* Java
* Angular
* Restful API
* HTML5
* CSS/SCSS
* JavaScript
* TypeScript
* Tích hợp thanh toán bằng ví Momo

**Công cụ:**

* Eclipse IDE
* Visual Studio Code
* Postman
* Ứng dụng thanh toán ví Momo (môi trường test/môi trường thực tế)

1. **Các kết quả chính dự kiến sẽ đạt được và ứng dụng**

* Hoàn chỉnh cuốn báo cáo đề tài
* Hiểu rõ về Spring Boot, Restful API và Angular
* Nắm rõ được mô hình và nghiệp vụ của hệ thống bán hàng trực tuyến và có thể áp dụng được mô hình này cho bất kỳ bài toán nào khác có liên quan
* Hoàn chỉnh tìm hiểu và phân tích hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại khu ký túc xá UTC2
* Xây dựng được website demo cho hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại khu tý túc xác UTC2

1. **Giáo viên và cán bộ hướng dẫn**

Họ tên: Trần Phong Nhã

Đơn vị công tác: Bộmôn Công NghệThông Tin - Trường đại học Giao thông vận tải phân hiệu tại thành phố Hồ Chí Minh

Điện thoại: 0906 761 014 Email: tpnha@utc2.edu.vn

|  |  |
| --- | --- |
| **Ngày … tháng … năm 2022**  **Trưởng BM Công nghệ Thông tin** | **Đã giao nhiệm vụ TKTN**  **Giáo viên hướng dẫn** |
|  | **Trần Phong Nhã** |
| Đã nhận nhiệm vụ TKTN  Sinh viên: Nguyễn Hoàng Minh Châu  Điện thoại: 0967 065 246 | Ký tên:  Email: minhchauk1k@gmail.com |

**LỜI CẢM ƠN**

Lời nói đầu tiên, tôi xin kính gửi lời cảm ơn chân thành nhất tới Quý thầy cô trong Bộ môn Công Nghệ Thông Tin, cũng như Ban Giám Hiệu Trường Đại học Giao thông Vận tải phân hiệu tại Thành phố Hồ Chí Minh, đã cho phép tôi thực hiện đề tài tốt nghiệp **“Phân tích và xây dựng hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại ký túc xá UTC2”**.

Để hoàn thành nhiệm vụ được giao này, ngoài sự nỗ lực học hỏi không ngừng của bản thân còn có sự hướng dẫn tận tình của các giảng viên trong 4 năm vừa qua, đặc biệt hơn hết nhờ có giảng viên **Trần Phong Nhã**, người thầy đã hướng dẫn cho tôi những hướng đi, truyền đạt cho tôi những kiến thức, kỹ năng để tôi có thể hoàn thành đề tài tốt nghiệp này.

Mặc dù đã cố gắng hết sức để hoàn thành đề tài, nhưng chắc chắn rằng sẽ khó tránh khỏi những thiếu sót. Tôi rất mong nhận được những sự đánh giá, góp ý của Quý thầy cô để tôi có thể rút ra cho mình những bài học, kinh nghiệm quý báu.

Sau cùng, tôi cũng không biết nói gì hơn ngoài kính chúc Quý thầy cô trong Bộ môn Công Nghệ Thông Tin và đặc biệt là thầy **Trần Phong Nhã** thật dồi dào sức khỏe và ngày càng gặt hái được nhiều thành công hơn nữa trong cuộc sống cũng như trong sự nghiệp giảng dạy của mình.

Tôi xin chân thành cảm ơn!

|  |  |
| --- | --- |
|  | **TP. Hồ Chí Minh, ngày ... tháng ... năm 2022**  **Sinh viên thực hiện**  **Nguyễn Hoàng Minh Châu** |

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

|  |
| --- |
| **Tp. Hồ Chí Minh, ngày … tháng … năm 2022**  **Giảng viên hướng dẫn**  **ThS. Trần Phong Nhã** |

**MỤC LỤC**

[DANH MỤC CHỮ VIẾT TẮT i](#_Toc106803553)

[BẢNG BIỂU, SƠ ĐỒ VÀ HÌNH ẢNH ii](#_Toc106803554)

[MỞ ĐẦU iii](#_Toc106803555)

[Lý do chọn đề tài iii](#_Toc106803556)

[Mục tiêu nghiên cứu iv](#_Toc106803557)

[Đối tượng và phạm vi nghiên cứu iv](#_Toc106803558)

[Phương pháp nghiên cứu iv](#_Toc106803559)

[Tính thiết thực của đề tại iv](#_Toc106803560)

[Cấu trúc nội dung báo cáo vi](#_Toc106803561)

[Kết luận tổng quan vi](#_Toc106803562)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 1](#_Toc106803563)

[1.1 Java [1][4] 1](#_Toc106803564)

[1.1.1 Giới thiệu về Java 1](#_Toc106803565)

[1.1.2 Lịch sử hình thành 2](#_Toc106803566)

[1.1.3 Đặc điểm của Java [13] 3](#_Toc106803567)

[1.2 Spring Boot [3][7] 5](#_Toc106803568)

[1.2.1 Giới thiệu về Spring Boot 5](#_Toc106803569)

[1.2.2 Định nghĩa về Micro Service 6](#_Toc106803570)

[1.2.3 Các tính năng nổi bật chính của Spring Boot 9](#_Toc106803571)

[1.3 Angular [8] 10](#_Toc106803572)

[1.3.1 Giới thiệu về Angular 10](#_Toc106803573)

[1.3.2 Các đặc điểm và tính năng nổi bật của Angular 12](#_Toc106803574)

[1.3.3 Lợi ích của Angular 15](#_Toc106803575)

[1.4 Giới thiệu về API [5] 16](#_Toc106803576)

[1.5 MySQL [16] 17](#_Toc106803577)

[1.5.1 Giới thiệu về MySQL 17](#_Toc106803578)

[1.5.2 Đặc điểm của MySQL 18](#_Toc106803579)

[CHƯƠNG 2. PHÂN TÍCH VÀ TRIỂN KHAI VẤN ĐỀ 19](#_Toc106803580)

[2.1 Bài toán thực tế 19](#_Toc106803581)

[2.1.1 Tiến hành phân tích 20](#_Toc106803582)

[2.1.2 Hướng giải quyết vấn đề 23](#_Toc106803583)

[2.1.3 Đưa ra kết luận 26](#_Toc106803584)

[2.2 Xây dựng và triển khai hệ thống 28](#_Toc106803585)

[2.2.1 Thiết kế cơ sở dữ liệu 28](#_Toc106803586)

[2.2.2 Thiết kế sơ đồ Use Case tổng quát 37](#_Toc106803587)

[2.2.3 Thiết kế phần xử lý Back-End 39](#_Toc106803588)

[2.2.4 Thiết kế phần xử lý Front-End 51](#_Toc106803589)

[CHƯƠNG 3: ỨNG DỤNG HỆ THỐNG VÀO MÔ HÌNH BÀI TOÁN THỰC TẾ 57](#_Toc106803590)

[3.1 Quy trình tạo đơn hàng của người dùng 57](#_Toc106803591)

[3.1.1 Người dùng chưa đăng nhập 57](#_Toc106803592)

[3.1.2 Người dùng đã đăng nhập 59](#_Toc106803593)

[3.2 Quy trình thanh toán của người dùng 60](#_Toc106803594)

[3.2.1 Thanh toán bằng tiền mặt 60](#_Toc106803595)

[3.2.2 Thanh toán bằng ví điện tử Momo 61](#_Toc106803596)

[3.3 Quy trình xử lý đơn hàng của chủ cửa hàng 65](#_Toc106803597)

[3.3.1 Không quản lý theo lô 65](#_Toc106803598)

[3.3.2 Có quản lý theo lô 68](#_Toc106803599)

[3.4 Quy trình cập nhật trạng thái thông tin sản phẩm của chủ cửa hàng 71](#_Toc106803600)

[3.4.1 Không sử dụng biến hệ thống 71](#_Toc106803601)

[3.4.2 Có sử dụng biến hệ thống 72](#_Toc106803602)

[KẾT LUẬN VÀ KIẾN NGHỊ 74](#_Toc106803603)

[Kết quả đạt được 74](#_Toc106803604)

[Các vấn đề còn tồn đọng 75](#_Toc106803605)

[Đề xuất hướng phát triển 75](#_Toc106803606)

[TÀI LIỆU THAM KHẢO 76](#_Toc106803607)

# DANH MỤC CHỮ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Từ viết tắt** | **Ý nghĩa** | **Ghi chú** |
| 1 | CNTT | Công nghệ thông tin |  |
| 2 | HMAC | Hash-based Message Authentication Code | Thuật toán mã hóa xác thực tin nhắn |
| 3 | UTC2 | University of Transport and Communications 2 | Tên viết tắt của trường |
| 4 | URL | Uniform Resource Locator | Đường dẫn tài nguyên |
| 5 | JPA | Java Persistence API | Một đặc tả của Java cho việc ánh xạ giữa các đối tượng Java với cơ sở dữ liệu |
| 6 | DB | Database | Cơ sở dữ liệu |
| 7 | API | Application Programming Interface | Giao diện lập trình ứng dụng |
| 8 | OOP | Object Oriented Programming | Lập trình hướng đối tượng |
| 9 | JVM | Java Virtual Machine | Máy ảo Java |
| 10 | JDK | Java Development Kit | Một tập hợp những công cụ phần mềm giúp phát triển ứng dụng Java |
| 11 | SQL | Structed Query Language | Ngôn ngữ truy vấn có cấu trúc |

# BẢNG BIỂU, SƠ ĐỒ VÀ HÌNH ẢNH

[Hình 1. 1 Giới thiệu về Java 1](#_Toc106803538)

[Hình 1. 2 Lịch sử phát triển Java 2](#_Toc106803539)

[Hình 1. 3 Spring Framework 5](#_Toc106803540)

[Hình 1. 4 Spring Boot 5](#_Toc106803541)

[Hình 1. 5 Ví dụ về mô hình kiến trúc của Micro Service 6](#_Toc106803542)

[Hình 1. 6 Ví dụ về mô hình hoạt động độc lập 7](#_Toc106803543)

[Hình 1. 7 Ví dụ về mô hình hoạt động chuyên biệt 8](#_Toc106803544)

[Hình 1. 8 Quá trình phát triển của Angular 11](#_Toc106803545)

[Hình 1. 9 Ví dụ về lập trình với Angular 11](#_Toc106803546)

[Hình 1. 10 Ví dụ về một Component 13](#_Toc106803547)

[Hình 1. 11 Ví dụ về Binding trong Angular 14](#_Toc106803548)

[Hình 1. 12 Ví dụ về Binding một chiều 14](#_Toc106803549)

[Hình 1. 13 Ví dụ về Binding hai chiều 14](#_Toc106803550)

[Hình 1. 14 Ví dụ về ứng dụng trang đơn và ứng dụng truyền thống 15](#_Toc106803551)

[Hình 1. 15 Giới thiệu về MySQL 17](#_Toc106803552)

[Hình 2. 1 Áp lực từ việc quản lý quá nhiều thông tin 13](#_Toc106799418)

[Hình 2. 2 Cung cấp thông tin cho người sử dụng 16](#_Toc106799419)

[Hình 2. 3 Ghi nhận và xác nhận đơn hàng 17](#_Toc106799420)

[Hình 2. 4 Quản lý địa điểm giao hàng 18](#_Toc106799421)

[Hình 2. 5 Thống kê báo cáo, kiểm soát luồng tiền 19](#_Toc106799422)

[Hình 2. 6 Mô hình phát triển kinh doanh hiện đại 20](#_Toc106799423)

# MỞ ĐẦU

## Lý do chọn đề tài

Những năm gần đây, cuộc cách mạng công nghiệp 4.0 (cách mạng công nghiệp lần thứ 4) đang nổi lên với sự phát triển mạng Internet cộng thêm các phương thức thanh toán trực tuyến (không sử dụng tiền mặt) đang nổi lên và trở thành xu thế được chính phủ và nhà nước khuyến khích người dân sử dụng, nhằm quản lý nền kinh tế dễ dàng cũng như tranh lạm phát xảy ra.

Ngoài ra, trong hai nằm vừa qua, toàn thế giới đã phải hứng chịu một đại dịch toàn cầu đó là Covid-19, gây thiệt hại nặng nề, kinh khủng đến nền kinh tế toàn cầu cũng như về sức khỏe trong cộng đồng.

Trong tình hình diễn biến phức tạp đó, chính phủ/nhà nước đã đề ra hướng giải quyết đó là “Cách ly cộng đồng – Hạn chế tiếp xúc tối đa cũng như không tụ tập nơi đông người” nhằm mục tiêu kiểm soát và hạn chế tình hình dịch bệnh diễn ra.

Qua đó mà các nhu cầu như mua sắm đồ ăn, nhu yếu phẩm, … trên mô hình đặt hàng và thanh toán trực tuyến đã cũng tăng nhanh một cách chóng mặt trên các mô hình quen thuộc tại các trang thương mại điện tử như Shopee, Grab, Tiki … Như chúng ta đều biết, các mô hình thương mại điện tử điển hình như các trang web bán hàng trực tuyến này không quá xa lạ cũng như đã từ lâu về trước. Mặc dù chúng đã duy trì và tồn tại được lâu như vậy, thế nhưng có khá ít người biết được lý do cũng như lợi ích to lớn không thể đong đếm hết được mà chúng mang lại cho chúng ta, mà cụ thể hơn là trong tình hình thực tiễn là dịch bệnh đang diễn ra khá phúc tạp như này.

Và để làm rõ các vấn đề nêu trên đó, nên tôi đã quyết định thực hiện nghiên cứu đề tài **“Phân tích và xây dựng hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại ký túc xá UTC2”** và chọn vấn đề này làm đề tài thực hiện cho đồ án tốt nghiệp của mình.

## Mục tiêu nghiên cứu

Sau khi đã xác định được đề tài, tôi đã đặt ra các mục tiêu nghiên cứu như sau:

- Nắm rõ được mô hình và nghiệp vụ của hệ thống bán hàng trực tuyến và có thể áp dụng được mô hình này cho bất kỳ bài toán nào khác có liên quan.

- Hoàn chỉnh tìm hiểu và phân tích hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại khu ký túc xá UTC2.

- Xây dựng được website demo cho hệ thống đặt đồ ăn trực tuyến cho sinh viên nội trú tại khu tý túc xác UTC2.

## Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Mô hình bán đồ ăn/sản phẩm khác tại căn tin khu ký túc xá UTC2, các website bán hàng trực tuyến khác đang phổ biến hiện nay.

Phạm vi nghiên cứu: Mô hình các trang web bán hàng trực tuyến để áp dụng cho nội bộ các sinh viên đang ở nội trú tại ký túc xá UTC2.

## Phương pháp nghiên cứu

Chủ động tiến hành thu thập và phân tích các tài liệu, thông tin liên quan đến đề tài. Từ đó, lựa chọn phương hướng giải quyết vấn đề, tìm hiểu thuật toán và các nghiệp vụ riêng, sau đó ứng dụng các thuật toán, nghiệp vụ này vào hệ thống.

## Tính thiết thực của đề tại

Việc xây dựng được mô hình, hệ thống tốt cho trang web bán hàng trực tuyến sẽ giúp ích rất nhiều cho các cá nhân và doanh nghiệp:

***Đối với cá nhân:***

Người sử dụng (người có nhu cầu mua hàng): sẽ dễ dàng tiếp cận được những sản phẩm, mặt hàng hoặc dịch vụ mà mình đang có nhu cầu một cách thuận tiện và đơn giản nhất. Ngoài ra còn theo dõi được thông tin về sản phẩm (tình trạng sản phẩm còn hàng hoặc đã bán hết, giá cả, chương trình khuyến mãi, …) mà không cần phải tốn công đích thân trực tiếp đến tận nơi ở khu căn tin để tham khảo. Bên cạnh đó, việc hệ thống cho phép thanh toán trực tuyến cũng giúp cho việc mở rộng thêm các phương thức thanh toán khác nhau, người sử dụng có thể chọn thanh toán theo phương thức thuận tiện với mình nhất, ngoài ra mô hình thanh toán trực tiếp giúp ta nắm được rõ ràng, chân thực và cái nhìn khách quan nhất về nguồn tiền hiện có cũng như giảm thiểu công sức về việc quản lý tiền bạc.

Người quản lý (chủ cửa hàng): sẽ dễ dàng cung cấp thông tin, tình trạng về các sản phẩm/dịch vụ hiện có trong cửa hàng, giúp thông tin tiếp cận người mua hàng một cách nhanh chóng, kịp thời mà không cần phải mất công thông báo, hoặc gọi điện xác nhận, … Không những thế, hệ thống giúp cho việc chuẩn bị các đơn hàng một cách rõ ràng và cụ thể nhất (tránh thiếu sót sản phẩm/dịch vụ và các rủi ro xảy ra trong quá trình chuẩn bị đơn hàng cho người mua) thông qua việc xem trực tiếp chi tiết danh sách những sản phẩm mà người dùng đã đặt hàng từ đơn hàng mà hệ thống đã ghi nhận được. Ngoài ra, hệ thống còn giúp cho người quản lý nắm bắt được các địa điểm giao hàng cụ thể mà người mua hàng đã cung cấp, qua đó mà lựa chọn hình thức giao hàng/vận chuyển hợp lý và phù hợp nhất với cửa hàng. Khi sử dụng hệ thống, người quản lý có thể theo dõi được trạng thái làm việc/kinh doanh của doanh nghiệp, cũng như các thống kê báo cáo về doanh thu, các sản phẩm được ưa thích, … thống kê đơn hàng hiện có trong ngày, trong tuần, giữa các tuần … qua đó đưa ra hoặc đề xuất các chiến lược kinh doanh phù hợp với tình hình hiện tại của cửa hàng/doanh nghiệp.

***Đối với doanh nghiệp:***

Hiện nay trên thị trường có xuất hiện nhiều môi trường hỗ trợ lập ra một cửa hàng/trang web cho phép doanh nghiệp có thể tiến hành kinh doanh trực tuyến như Shopee, Lazada, Tiki, Foody … , tuy nhiên hiện nay các mô hình này có phụ phí kinh doanh khá cao nên các doanh nghiệp nhỏ và vừa khó có thể tiếp cận hoặc còn quan ngại để tiếp cận. Không những thế, một số cửa hàng/doanh nghiệp còn có riêng cho mình một hệ thống nghiệp vụ riêng của họ, mà các mô hình kinh doanh trực tuyến từ các môi trường trên không thể đáp ứng hết được, vậy nên một số doanh nghiệp/cửa hàng dù rất muốn có thể kinh doanh trực tuyến song lại không thể thực hiện trong thời điểm này được. Do đó việc phân tích và xây dựng ra một hệ thống quản lý và kinh doanh trực tuyến riêng cho doanh nghiệp/cửa hàng cũng chính là một trong những việc ưu tiên hàng đầu, cần phải thực hiện. Việc xây dựng nên được một hệ thống như thế sẽ giúp ích rất nhiều cũng như thỏa mãn những vấn đề này cho các doanh nghiệp/cửa hàng có mong muốn kinh doanh trực tuyến nhưng chưa thể làm được.

## Cấu trúc nội dung báo cáo

Đồ án được chia thành các chương như sau:

Chương 1: Cơ sở lý thuyết

Chương 2: Phân tích và triển khai vấn đề

Chương 3: Ứng dụng hệ thống vào mô hình bài toán thực tế

## Kết luận tổng quan

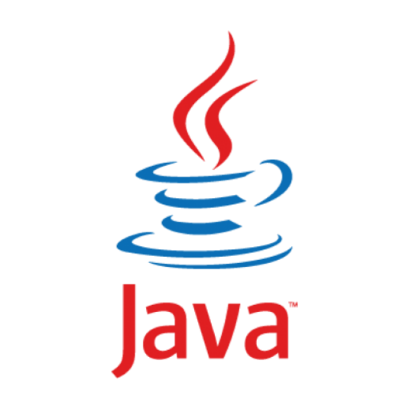
Trong phần này, tôi đã giới thiệu khái quát tổng quan những vấn đề mà đề tài nghiên cứu.

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## 1.1 Java [1][4]

### 1.1.1 Giới thiệu về Java

Java là một ngôn ngữ lập trình hướng đối tượng (OOP) và dựa trên các lớp (class). Khác với phần lớn ngôn ngữ lập trình thông thường, thay vì biên dịch mã nguồn thành mã máy hoặc thông dịch mã nguồn khi chạy, Java được thiết kế để biên dịch mã nguồn thành bytecode, bytecode sau đó sẽ được môi trường thực thi (runtime environment) chạy.



Hình 1. 1 Giới thiệu về Java

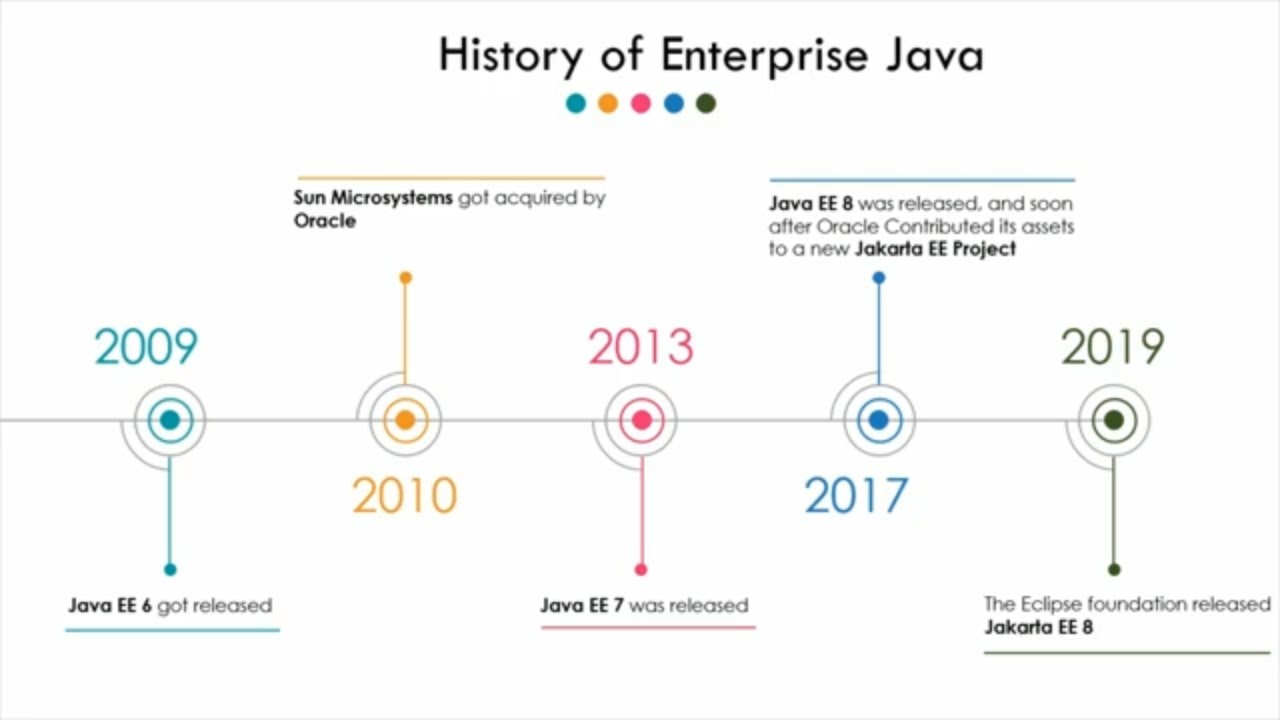
Trước đây, Java chạy chậm hơn những ngôn ngữ dịch thẳng ra mã máy như C và C++, nhưng sau này nhờ công nghệ "biên dịch tại chỗ" - Just in time compilation, khoảng cách này đã được thu hẹp, và trong một số trường hợp đặc biệt Java có thể chạy nhanh hơn. Java chạy nhanh hơn những ngôn ngữ thông dịch như Python, Perl, PHP gấp nhiều lần. Java chạy tương đương so với C#, một ngôn ngữ khá tương đồng về mặt cú pháp và quá trình dịch/chạy.

Cú pháp Java được vay mượn nhiều từ C & C++ nhưng có cú pháp hướng đối tượng đơn giản hơn và ít tính năng xử lý cấp thấp hơn. Do đó việc viết một chương trình bằng Java dễ hơn, đơn giản hơn, đỡ tốn công sửa lỗi hơn.

Trong Java, hiện tượng rò rỉ bộ nhớ hầu như không xảy ra do bộ nhớ được quản lý bởi Java Virtual Machine (JVM) bằng cách tự động "dọn dẹp rác". Người lập trình không phải quan tâm đến việc cấp phát và xóa bộ nhớ như C, C++. Tuy nhiên khi sử dụng những tài nguyên mạng, file IO, database (nằm ngoài kiểm soát của JVM) mà người lập trình không đóng (close) các streams thì rò rỉ dữ liệu vẫn có thể xảy ra.

### 1.1.2 Lịch sử hình thành

Java được khởi đầu bởi James Gosling và bạn đồng nghiệp ở Sun Microsystems năm 1991. Ban đầu ngôn ngữ này được gọi là Oak (có nghĩa là cây sồi; do bên ngoài cơ quan của ông Gosling có trồng nhiều loại cây này), họ dự định ngôn ngữ đó thay cho C++, nhưng các tính năng giống Objective C. Không nên lẫn lộn Java với JavaScript, hai ngôn ngữ đó chỉ giống tên và loại cú pháp như C. Công ty Sun Microsystems đang giữ bản quyền và phát triển Java thường xuyên. Tháng 04/2011, công ty Sun Microsystems tiếp tục cho ra bản JDK 1.6.24.



Hình 1. 2 Lịch sử phát triển Java

Java được tạo ra với tiêu chí "Viết (code) một lần, thực thi khắp nơi" ("Write Once, Run Anywhere" (WORA)). Chương trình ứng dụng viết bằng Java có thể chạy trên mọi nền tảng (platform) khác nhau thông qua một môi trường thực thi với điều kiện có môi trường thực thi thích hợp hỗ trợ nền tảng đó. Môi trường thực thi của Sun Microsystems hiện hỗ trợ Sun Solaris, Linux, Mac OS, FreeBSD & Windows. Ngoài ra, một số công ty, tổ chức cũng như cá nhân khác cũng phát triển môi trường thực thi Java cho những hệ điều hành khác như BEA, IBM, HP.... Trong đó đáng nói đến nhất là IBM Java Platform hỗ trợ Windows, Linux, AIX & z/OS.

Những chi tiết về ngôn ngữ, máy ảo và API của Java được giữ bởi Cộng đồng Java (do Sun quản lý). Java được tạo ra vào năm 1991 do một số kỹ sư ở Sun, bao gồm ông James Gosling, một phần của Dự án Xanh (Green Project). Java được phát hành vào năm 1994, rồi nó trở nên nổi tiếng khi Netscape tuyên bố tại hội thảo SunWorld năm 1995 là trình duyệt Navigator của họ sẽ hỗ trợ Java. Về sau Java được hỗ trợ trên hầu hết các trình duyệt như Internet Explorer (Microsoft), Firefox (Mozilla), Safari (Apple), ...

Java được sử dụng chủ yếu trên môi trường NetBeans và Oracle. Sau khi Oracle mua lại công ty Sun Microsystems năm 2009-2010, Oracle đã mô tả họ là "người quản lý công nghệ Java với cam kết không ngừng để bồi dưỡng một cộng đồng tham gia và minh bạch".

### 1.1.3 Đặc điểm của Java [13]

***Ưu điểm:***

+ Ưu điểm được nhắc đến đầu tiên, cũng chính là niềm tự hào đặc trưng của JAVA: Cross – Platform. Vậy Cross – Platform có nghĩa là bạn có thể viết mã nguồn trên một IDE ở một máy tính và thực thi chương trình ở bất cứ máy tính sử dụng hệ điều hành nào. Với điều kiện máy tính thực thi chương trình phải có hệ thống trình biên-thông dịch riêng được gọi là Java Virtual Machine (JVM), JVM sẽ có nhiều phiên bản dành riêng cho từng hệ điều hành (Windows, Linus và MAC OS).

+ Ưu điểm thứ 2, Java là ngôn ngữ hướng đối tượng – phù hợp với xu thế phát triển. “Chương trình phải được thiết kế trước khi xây dựng”, không lập trình viên nào có thể thực hiện toàn bộ một chương trình ngay từ đầu. Công việc đầu tiên phải thiết kế phân tích chia nhỏ các phần công việc độc lập, tiếp tục chia các phần nhỏ thành nhỏ hơn nữa. Với cơ chế đặc trưng “sử dụng lại”(kế thừa) , “mở rộng” (đa hình) trong kiến trúc OOP, lập trình viên có thể đầu tư phân tích và kiến tạo cấu trúc của toàn bộ một chương trình (Software Engineering) sao cho phù hợp với tính chất công việc.

+ Ưu điểm thứ 3, công nghệ mở và phát triển không ngừng. Cạnh tranh với các công nghệ trong .NET, cộng đồng công nghệ trong JAVA cũng không kém cạnh và miễn phí đường đường chính chính.

+ Các ưu điểm khác, phát triển mạnh hướng network, di động (phổ biến hiện nay là Android, ngoài ra còn có J2ME), an toàn trong công tác quản lí tài nguyên cấp phát động (cơ chế Garbage Collection – tương đồng trong C#), bảo mật ở mức quản lí chương trình với nhiều mức. Khi Java lần đầu tiên ra mắt đã gây nhiều ấn tượng với khả năng viết applet, là một một ứng dụng cho phép nhúng trên trang web, tuy nhiên về sau dần trở nên mờ nhạt vì nhược điểm thiếu an toàn bảo mật.

***Nhược điểm:***

+ Bảo mật sourcecode không tốt.

+ Tốc độ khá chậm, nhưng vẫn chấp nhận được.

## 1.2 Spring Boot [3][7]

### 1.2.1 Giới thiệu về Spring Boot

Spring Boot là một Framework (có thể tạm hiểu là một thư viện hỗ trợ) mã nguồn mở dựa trên ngôn ngữ lập trình Java nhằm mục đích tạo ra một Micro Service (có thể tạm hiểu là một mô hình máy chủ mini dùng để xử lý dữ liệu). Đây là một dự án được phát triển bởi nhiều người sử dụng khác nhau (Pivotal Team) trong hệ sinh thái Spring framework.



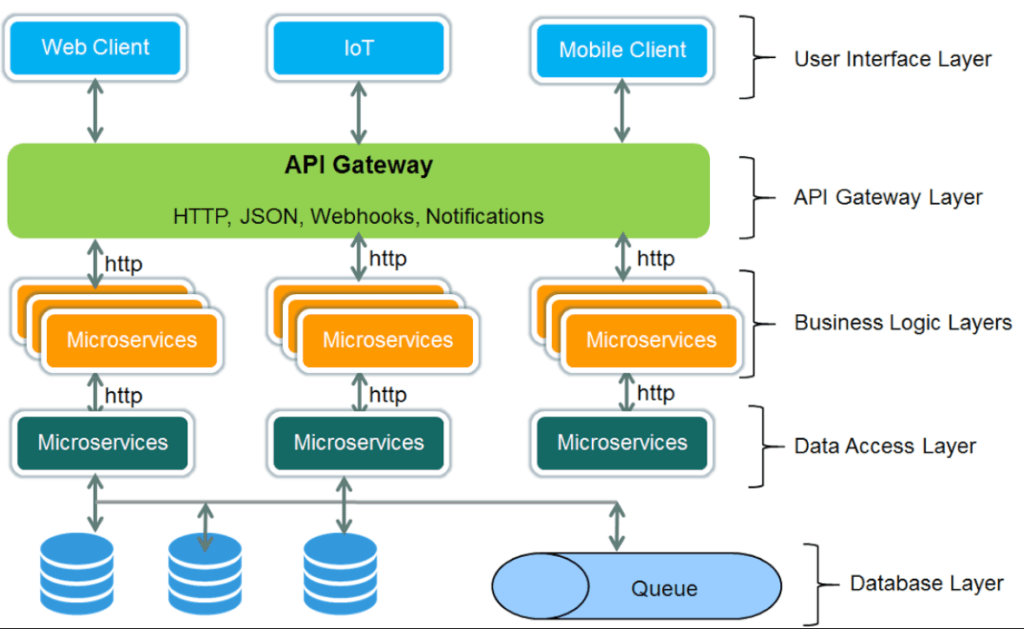
Hình 1. 3 Spring Framework

Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring một cách thuận tiện, đơn giản và nhanh chóng, và chỉ cần tập trung vào việc phát triển business (về mặt nghiệp vụ khách hàng/kinh doanh) cho ứng dụng.



Hình 1. 4 Spring Boot

### 1.2.2 Định nghĩa về Micro Service



Hình 1. 5 Ví dụ về mô hình kiến trúc của Micro Service

Sơ đồ trên là một thể hiện của mô hình Micro Service. Trên thực tế mô hình Micro Service được áp dụng vào các sản phẩm có rất nhiều biến thể, nên sẽ rất khó để có một mô hình phù hợp và chính xác cho từng bài toán.

Theo kiến trúc trên, một ứng dụng được chia thành một bộ các Micro Service, mỗi Micro Service thực chất là một service có thể được triển khai và chạy độc lập. Chúng tách biệt về mặt mã nguồn, về hoạt động và dữ liệu.

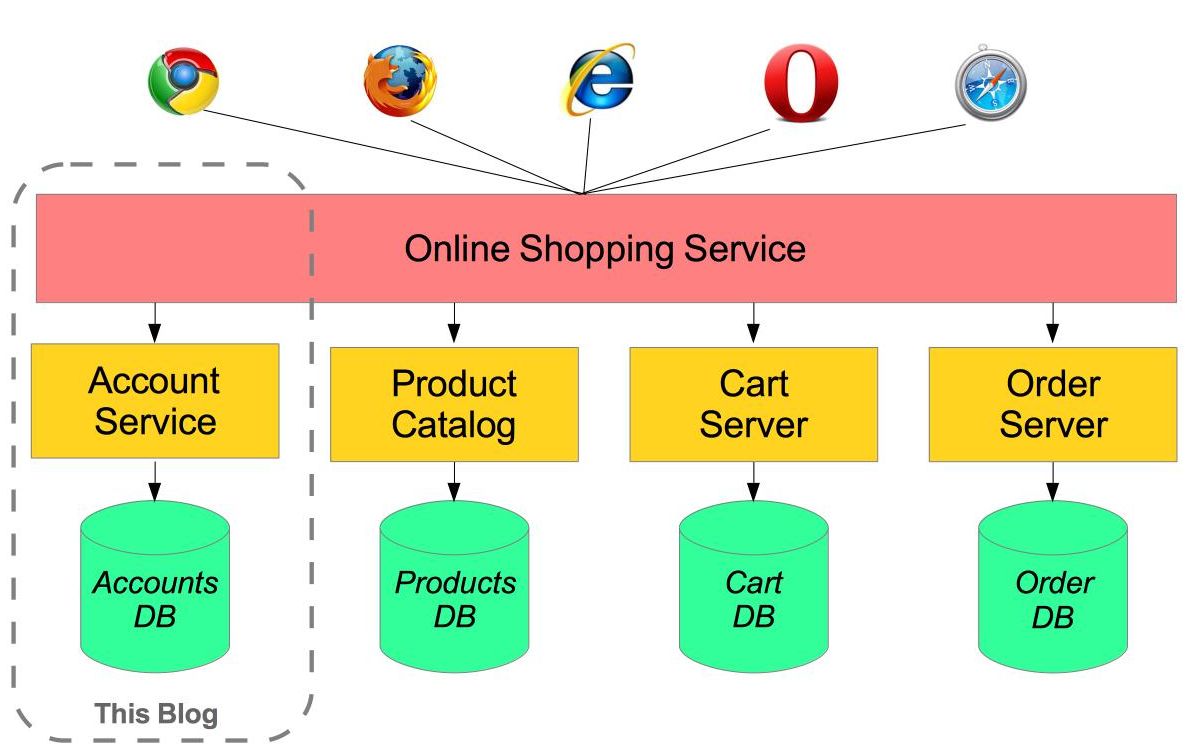
Mỗi Micro Service có nơi chứa dữ liệu của riêng của nó và chỉ có nó có quyền truy cập vào vùng dữ liệu này. Do các Micro Service là độc lập, chúng không giao tiếp trực tiếp với nhau mà qua một thành phần trung gian được gọi là API gateway.

Do đó vai trò của API gateway rất quan trọng trong mô hình Micro Service. Nó là điểm đến và đi của mọi yêu cầu hay phản hồi.

#### 1.2.2.1 Tính độc lập

Các Micro Service hoạt động tách biệt nhau trong hệ thống, do vậy việc build một Micro Service cũng độc lập với việc build các Micro Service khác.

Mỗi Micro Service là một dịch vụ chuyên biệt, có thể hoạt động độc lập.



Hình 1. 6 Ví dụ về mô hình hoạt động độc lập

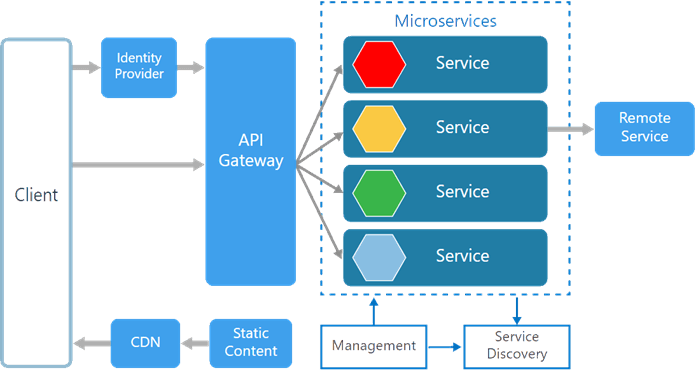
Do tính tách biệt này mà mỗi Micro Service đều dễ dàng thay thế và mở rộng.

Hơn thế nữa, nó còn giúp việc phát triển các Micro Service linh động hơn, các Micro Service có thể được phát triển bởi các team khác nhau, dùng các ngôn ngữ khác nhau và tiến độ phát triển dự án cũng nhanh hơn do không có sự phụ thuộc giữa các team.

Mỗi team có thể chủ động quản lý phần việc riêng của mình.

#### 1.2.2.2 Tính chuyên biệt

Thông thường mỗi Micro Service đại diện cho một tính năng mà các công ty/ doanh nghiệp muốn cung cấp tới người dùng.



Hình 1. 7 Ví dụ về mô hình hoạt động chuyên biệt

Hoặc cũng có thể hiểu nó là các dịch vụ riêng biệt của một ứng dụng hay trang web riêng biệt với nhau mà người dùng truy cập.

#### 1.2.2.3 Phòng chống xảy ra lỗi

Kiến trúc Micro Service sinh ra là để dành cho các hệ thống từ lớn đến vô cùng lớn.

Nó áp dụng phương pháp “***chia để trị***”, phương pháp này giúp việc áp dụng các công cụ, kỹ thuật cho việc giám sát, phòng chống lỗi phần mềm, lỗi hệ thống hiệu quả.

Khi một thành phần trong hệ thống bị lỗi, nó có thể được thay thế bằng các thành phần dự phòng một cách dễ dàng, trong quá trình thay thế thành phần bị lỗi, các thành phần khác vẫn hoạt động bình thường.

Do vậy hoạt động của toàn bộ hệ thống sẽ không hoặc ít bị gián đoạn.

### 1.2.3 Các tính năng nổi bật chính của Spring Boot

- ***Dễ dàng triển khai***: Server Tomcat, Jetty hoặc Undertow có thể được nhúng trực tiếp ngay trong file JAR build ra và có thể chạy ở bất kì đâu mà java chạy được (không cần triển khai tệp WAR).

- ***Tự động cấu hình***: Hệ thống đã tự động cấu hình thay lập trình viên, bạn chỉ cần viết code và tiến hành chạy hệ thống là được (đơn giản hóa quá trình xây dựng cấu hình của lập trình viên).

- ***Dễ dàng mở trộng***: Dựa trên các Annotation để tự động cấu hình cho các Java Bean, cấu hình XML và cấu hình Database Transactions (phiên giao dịch cơ sở dữ liệu). Trong Spring Boot, mọi thứ đều được cấu hình tự động, không cần cấu hình thủ công.

- ***Tiết kiệm thời gian và chi phí***: Vì dựa trên mục tiêu đơn giản hóa và hỗ trợ tối đa cho các lập trinh viên, nên quá trình từ khi bắt đầu đến khi tạo ra sản phẩm tiêu tốn rất ít chi phí và tài nguyên (tiền bạc và thời gian) nhưng năng suất sẽ được cải thiện rõ rệt.

## 1.3 Angular [8]

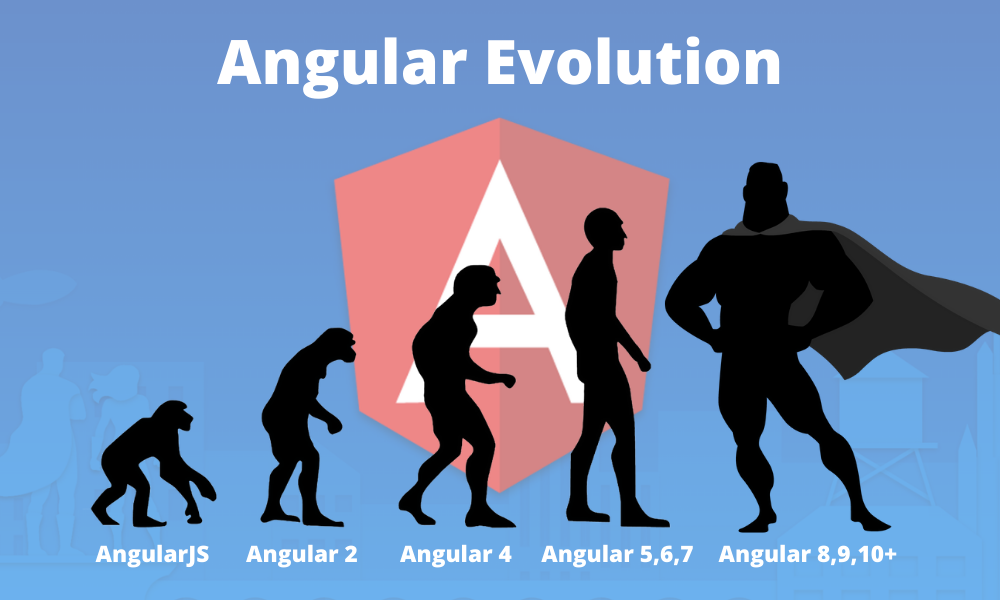
### 1.3.1 Giới thiệu về Angular

Angular được xem là một open source (mã nguồn mở) hay một JavaScript frameworks miễn phí chuyên dụng cho công việc thiết kế giao diện web (Front-end). Angular được phát triển từ những năm 2009 và được duy trì bởi Google.

Trên thực tế, Angular sở hữu hai phiên bản hoàn toàn khác nhau như sau: Angular phiên bản 1.x hay còn được gọi là **AngularJS** và Angular phiên bản 2.x (hiện nay là các phiên bản **từ Angular 2 đến Angular 10+**) phiên bản này được gọi với cái tên chính thức là **Angular.**

Phiên bản bản Angular 2.x ra đời nhằm thay thế AngularJS với các khái niệm mới nhằm đơn giản hóa và tối ưu cho quá trình phát triển sử dụng framework này.

Angular 2.x thay đổi hoàn toàn so với AngularJS, phiên bản này hoàn toàn được viết bằng Typescript. Ngoài ra nó còn nhanh hơn “người anh em tiền nhiệm” AngularJS khá nhiều khi được sử dụng, phiên bản này cũng hỗ trợ đa nền tảng trên đa trình duyệt, giúp cho cấu trúc code được tổ chức đơn giản và dễ sử dụng hơn.



Hình 1. 8 Quá trình phát triển của Angular

Theo như thống kê, frameworks này được xem là frameworks Front-End mạnh mẽ nhất và được ưa dùng bởi các lập trình viên chuyên xử lý về phần giao diện web ở level cao.

Lý do là, Angular giúp cho các lập trình viên xây dựng các ứng dụng trang đơn (Single-Page Application) bằng cách sử dụng HTML và TypeScript một cách nhanh hơn.

Có thể lấy ví dụ cụ thể và đơn giản như sau:

“*Bình thường nếu chỉ làm bằng tay, bạn sẽ làm ra được 10 cây viết/ngày nhưng nếu sử dụng công cụ máy móc, bạn sẽ làm ra được 100, thậm chí 1000 cây viết/ngày*.”



Hình 1. 9 Ví dụ về lập trình với Angular

Angular đối với các lập trình viên cũng như công cụ ở trên vậy, nếu không sử dụng framework này thì người lập trình sẽ mất rất nhiều thời gian để hoàn thiện giao diện web.

### 1.3.2 Các đặc điểm và tính năng nổi bật của Angular

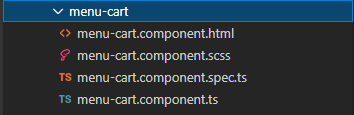
#### 1.3.2.1 Đặc điểm

***Components:***

Khi chúng ta thực hiện code, Angular sẽ quản lý ứng dụng của chúng ta bằng các **Components** khác nhau (có thể hiển đơn giản là một khối code trong app Angular)

Mỗi Components thường gồm một tập hợp ba file khác nhau:

* File TypeScript: với đuôi là **.ts**, file này được xem như một **Controller** dùng để điều khiển và xử lý các hoạt động/sự kiện xảy ra trên một đơn vị Components.
* File HTML: với đuôi là **.html**, file này được xem như phần **View** hoặc giao diện hiển thị ra bên ngoài của một đơn vị Components mà người dùng có thể thấy được trực tiếp trên màn hình khi sử dụng ứng dụng.
* File CSS/SCSS: với đuôi là **.css/.scss**, file này được xem như một chiếc “**cọ vẽ**”, thứ sẽ giúp cho phần View/giao diện hiển thị của chúng ta trở nên đặc sắc và sinh động hơn.



Hình 1. 10 Ví dụ về một Component

Các components là độc lập với nhau và được định nghĩa độc lập với hệ thống. Nó có thể được cài vào hoặc tháo ra khỏi hệ thống dễ dàng.

Một component cũng có thể to lớn như là cả 1 màn hình chứa nhiều control hoặc một nhóm nhiều màn hình. Tức là là một component cũng có thể chứa và gọi được nhiều component khác nối vào.

#### 1.3.2.2 Các tính năng nổi bật

***Binding:***

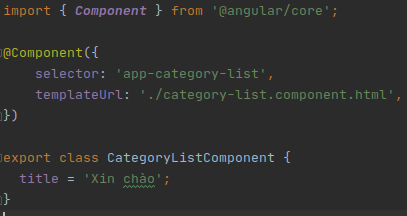
Angular có cách code Binding (kết nối giữa giao diện HTML và Data hiện có trong file xử lý) dữ liệu theo kiểu 2 chiều.

Nghĩa là khi giao diện HTML thay đổi dữ liệu thì biến Data trong file xử lý (file TypeScript) sẽ ngay lập tức nhận được giá trị vừa thay đổi đó

Và ngược lại, khi giá trị Data trong file xử lý thay đổi thì ngay lập tức màn hình giao diện HTML cũng sẽ thay đổi theo.

Để bind một chuỗi ra ngoài màn hình HTML thì rất đơn giản, chỉ cần sử dụng 2 dấu ngoặc nhọn **{{**TenBien**}}** là có thể làm được.

Ví dụ chúng ta có một Component đơn giản như sau:



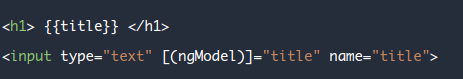
Hình 1. 11 Ví dụ về Binding trong Angular

Biến **title** có giá trị là “Xin chào” tại file xử lý, chúng ta chỉ cần thực hiện cú pháp như ảnh phía dưới thì giá trị “Xin chào” sẽ được hiển thị lên màn hình ứng dụng được chạy.

https://images.viblo.asia/full/0959a4c9-ca65-43f0-90fd-0835db00b59d.png

Hình 1. 12 Ví dụ về Binding một chiều

***To-way binding:***



Hình 1. 13 Ví dụ về Binding hai chiều

Với từ khóa **ngModel** là từ khóa của Angular. Khi chúng ta viết **[(ngModel)]** chúng ta sẽ gắn chặt giá trị của <input> của HTML với biến **title**.

Dẫn đến người dùng gõ bất kì gì vào ô <input> thì thẻ <h1> sẽ nhận giá trị tương ứng của **title**.

### 1.3.3 Lợi ích của Angular

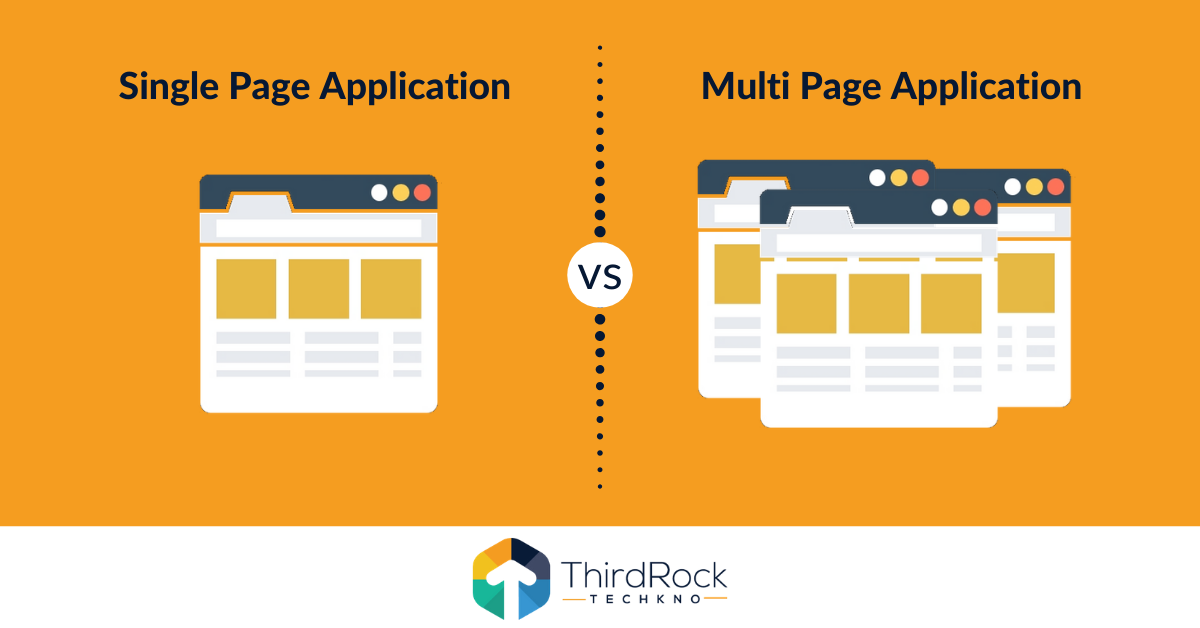
Angular được “chống lưng” bởi Google, giúp cho lập trình viên có cảm giác được đảm bảo. Mặc nhiên, họ sẽ ám thị rằng framework này khó mà bị “khai tử”, vì vậy cứ yên tâm sử dụng.

Cộng đồng người dùng lớn nên nếu có thắc mắc gì cũng sẽ nhanh chóng được giải đáp.

Bằng việc quản lý bằng các Component nên khi code chúng ta có thể tái sử dụng lại bằng cách gọi lại hoặc clone ra một Component mới, điều này giúp tối ưu được việc code với phương hướng “Code ít hơn, nhưng hiệu quả nhiều hơn”

Giúp phát triển Ứng dụng trang đơn (Single-page Application). Đây là ứng dụng chạy trên browser mà không bắt buộc phải tải lại trang khi sử dụng.

Ngoài ra nhờ việc Binding mà Angular cung cấp thì trang web khi sử dụng sẽ tạo cảm giác mượt mà, trơn tru hơn.



Hình 1. 14 Ví dụ về ứng dụng trang đơn và ứng dụng truyền thống

## 1.4 Giới thiệu về API [5]

Một API là một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu dịch vụ có thể được tạo ra từ các chương trình máy tính khác, và/hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng.

Chẳng hạn, một chương trình máy tính có thể (và thường là phải) dùng các hàm API của hệ điều hành để xin cấp phát bộ nhớ và truy xuất tập tin. Nhiều loại hệ thống và ứng dụng hiện thực API, như các hệ thống đồ họa, cơ sở dữ liệu, mạng, dịch vụ web, và ngay cả một số trò chơi máy tính.

Đây là ứng dụng hệ thống cung cấp đầy đủ các chức năng và các tài nguyên mà các lập trình viên có thể rút ra từ đó để tạo nên các tính năng giao tiếp người- máy như: các trình đơn kéo xuống, tên lệnh, hộp hội thoại, lệnh bàn phím và các cửa sổ. Một trình ứng dụng có thể sử dụng nó để yêu cầu và thi hành các dịch vụ cấp thấp do hệ điều hành của máy tính thực hiện.

Hệ giao tiếp lập trình ứng dụng giúp ích rất nhiều cho người sử dụng vì nó cho phép tiết kiệm được nhiều thời gian tìm hiểu các chương trình mới, do đó khích lệ mọi người dùng nhiều ứng dụng hơn.

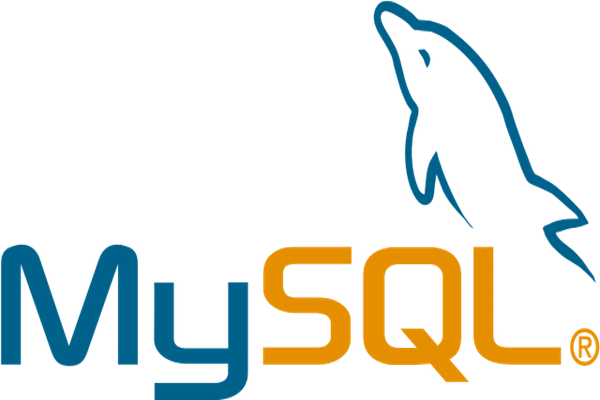
Một trong các mục đích chính của một API là cung cấp khả năng truy xuất đến một tập các hàm hay dùng — ví dụ: hàm để vẽ các cửa sổ hay các icon trên màn hình.

## 1.5 MySQL [16]

### 1.5.1 Giới thiệu về MySQL

MySQL là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở (Relational Database Management System, viết tắt là RDBMS) hoạt động theo mô hình client-server.

RDBMS là một phần mềm hay dịch vụ dùng để tạo và quản lý các cơ sở dữ liệu (Database) theo hình thức quản lý các mối liên hệ giữa chúng.



Hình 1. 15 Giới thiệu về MySQL

MySQL là một trong số các phần mềm RDBMS.

RDBMS và MySQL thường được cho là một vì độ phổ biến quá lớn của MySQL. Các ứng dụng web lớn nhất như Facebook, Twitter, YouTube, Google, và Yahoo! đều dùng MySQL cho mục đích lưu trữ dữ liệu.

Kể cả khi ban đầu nó chỉ được dùng rất hạn chế nhưng giờ nó đã tương thích với nhiều hạ tầng máy tính quan trọng như Linux, macOS, Microsoft Windows, và Ubuntu.

### 1.5.2 Đặc điểm của MySQL

#### 1.5.2.1 Linh hoạt và dễ dùng

Bạn có thể sửa source code để đáp ứng nhu cầu của bạn mà không phải thanh toán thêm bất kỳ chi phí nào.

Quá trình cài đặt cũng rất đơn giản và thường không quá 30 phút.

#### 1.5.2.2 Hiệu năng cao

Nhiều server clusters sử dụng MySQL.

Bất kể bạn lưu trữ dữ liệu lớn của các trang thương mại điện tử hoặc những hoạt động kinh doanh nặng nề liên quan đến công nghệ thông tin, MySQL cũng có thể đáp ứng được với tốc độ cao, mượt mà.

#### 1.5.2.3 An toàn

An toàn dữ liệu luôn là vấn đề quan trọng nhất khi chọn phần mềm RDBMS.

Với hệ thống phân quyền truy cập và quản lý tài khoản, MySQL đặt tiêu chuẩn bảo mật rất cao.

Việc mã hóa thông tin đăng nhập và chứng thực từ host đều khả dụng.

# CHƯƠNG 2. PHÂN TÍCH VÀ TRIỂN KHAI VẤN ĐỀ

## 2.1 Bài toán thực tế

Theo như khoảng thời gian tôi còn ở nội trú trong khu ký túc xá tại UTC2, tôi nhận thấy rằng việc quản lý tại khu căng tin nội trú tại đây còn chưa thực hiện được tốt.

Cụ thể là để đặt hàng một sản phẩm/dịch vụ tại đây, người có nhu cầu mua hàng (các bạn sinh viên) phải đến tận nơi để kiểm tra, xác nhận và thu thập thông tin về những sản phẩm/dịch vụ hiện đang sẵng sàng được bày bán. Sau đó mới tiến hành chọn sản phẩm/dịch vụ, kế đến là đặt mua và thanh toán tại chỗ, và cuối cùng là tự vận chuyển về khu vực nội trú của mình.

Hoặc buộc bạn phải biết được số điện thoại (cách thức liên lạc gián tiếp) của người chủ căng tin, sau đó mỗi lần muốn đặt một sản phẩm/dịch vụ thì bạn phải trực tiếp gọi điện để xác nhận rằng sản phẩm/dịch vụ mà mình muốn hiện có đang sẵng sàng để bán. Và nếu sản phẩm/dịch vụ đó vẫn còn có thể đặt hàng được thì mới tiến hành đặt hàng và chờ nhân viên của căng tin vận chuyển đến, cuối cùng là tiến hành thanh toán và kết thúc quá trình mua hàng.

Ngoài ra, quá trình quản lý và ghi nhận lại đơn hàng từ người có nhu cầu mua hàng (các bạn sinh viên) đã đặt hàng, hoàn toàn được người chủ căng tin quản lý bằng sổ ghi giấy (cụ thể là mỗi lần có đơn hàng mới thì người chủ căng tin phải ghi chú lại thủ công bằng tay vào giấy/vở ghi chép, sau đó dựa theo ghi chú mà xử lý nghiệp vụ với đơn hàng).

Việc quản lý giấy tờ này được lặp lại mỗi ngày và với số lượng rất lớn đơn đặt hàng, điều này dễ dẫn đến việc thiếu sót hoặc nhầm lẫn trong quá trình ghi nhận đơn hàng từ người có nhu cầu mua hàng (các bạn sinh viên) đã đặt, khiến cho quy trình làm việc thiếu an toàn và hiệu quả/chưa được tối ưu và chuyên nghiệp.

### 2.1.1 Tiến hành phân tích



Hình 2. 1 Áp lực từ việc quản lý quá nhiều thông tin

Vì căng tin là địa điểm kinh doanh dịch vụ ăn uống và cung cấp các dịch vụ khác là chủ yếu cho các bạn sinh viên ở nội trú tại khu ký túc xá. Với số lượng công việc mỗi ngày rất lớn như vậy thì việc quản lý, tính toán chi phí sẽ dễ phát sinh nhiều thiếu sót/nhầm lẫn không mong muốn, chính vì vậy mà doanh nghiệp phải luôn tìm cách để hạn chế các vấn đề và tình trạng đó một cách triệt để cũng như tối ưu và thiết thực nhất.

#### 2.1.1.1 Vấn đề về cung cấp thông tin cho người sử dụng

Theo như các vấn đề đã nêu ra ở trên, việc cung cấp thông tin để người sử dụng có thể nắm bắt được một cách rõ ràng và trực quan nhất về sản phẩm/dịch vụ từ phía căng tin đến với người có nhu cầu mua hàng (các bạn sinh viên ở nội trú) vẫn còn gặp nhiều vấn đề và bất cập cần phải giải quyết.

***Ví dụ về việc không nắm bắt được thông tin***:

Vì không thể nắm được thông tin sản phẩm/dịch vụ → Sinh viên cần phải trực tiếp đến tận nơi để xác nhận (đối với các bạn ở xa khu căng tin/không thể trực tiếp đến được) → Tiêu tốn thời gian và sức lực (khiến cho sinh viên ngại hoặc không muốn đặt hàng) → Sinh viên ở xa sẽ hạn chế đặt mua ở căng tin → Hiệu quả kinh doanh không tốt/bị kém.

Vì không thể nắm được thông tin sản phẩm → Sinh viên cần phải gọi điện cho chủ căng tin để tìm hiểu về thông tin sản phẩm/dịch vụ (tiêu tốn cước phí cuộc gọi) → Chủ căng tin phải yêu cầu bộ phận chuẩn bị kiểm tra tình trạng của sản phẩm/dịch vụ đó (vì chính chủ căng tin cũng không thể nắm bắt được, làm cho việc quản lý và quy trình làm việc chưa khoa học gây lãng phí thời gian và công sức) → Sau khi căng tin đã chuẩn bị xong đơn hàng thì mới tiến hành giao hàng (tuy nhiên sinh viên đã đặt hàng hiện tại không biết được đơn mình đặt đang ở trạng thái nào, có được xử lý hay chưa) → Khi nhân viên giao hàng cho sinh viên, phải thực hiện gọi điện lại cho bạn sinh viên đó để thông báo giao hàng (tiêu tốn cước phí cuộc gọi) → Tuy đã đặt hàng và thanh toán thành công vẫn tiêu tốn một lượng tài nguyên khá lớn → Hiệu quả kinh doanh không tốt/bị kém.

#### 2.1.1.2 Vấn đề về việc ghi nhận và xác nhận đơn hàng

Việc ghi lại đơn hàng bằng phương thức ghi sổ giấy thủ công sẽ gặp đôi rủi ro về mặt thông tin như: có thể ghi sai hoặc thiếu sản phẩm/dịch vụ, sai hoặc nhầm lẫn về địa chỉ giao, … do phải liên tục thực hiện với số lượng lớn đơn hàng và thao tác lặp đi lặp lại.

Không ít trường hợp vì ghi đơn hàng thủ công bằng tay vào giấy có sai sót dẫn đến sinh viên hủy đơn, không muốn nhận hàng hoặc giảm thiện cảm/lòng tin của sinh viên đối với căng tin/cách làm việc của căng tin.

Về mặt doanh nghiệp, những rủi ro này là thứ không đáng có và cần phải hạn chế xảy ra vì nó ảnh hưởng trực tiếp đến bộ mặt cũng như uy tín của doanh nghiệp.

Không kể đến việc tiêu tốn kinh phí cho việc mua giấy bút, sổ sách để ghi nhận đơn hàng, mà nó còn vô tình gây ra áp lực cho người ghi nhận vì họ phải luôn cố nhớ đơn hàng nào, của ai và trạng thái ra sao, … để thực hiện quy trình đặt và giao hàng cho đến tận khi đơn hàng hoàn thành. Tiêu hao rất nhiều tài nguyên về tiền bạc và nguồn lực con người.

#### 2.1.1.3 Vấn đề về quản lý địa điểm giao hàng

Hiện tại theo kiến thức mà tôi tìm hiểu được từ các ứng dụng, mua bán sản phẩm trực tuyến, các dịch vụ đặt đồ ăn nhanh trực tuyến khác như ShopeeFood, BAEMIN, … họ luôn sử dụng một thuật toán để tối ưu các địa điểm giao hàng.

Sao cho trong một lần giao hàng trong một khu vực cụ thể có thể tiết kiệm được tối đa chi phí vận chuyển mà năng suất giao hàng lại được tối ưu nhất.

Cụ thể là tránh các trường hợp nhân viên giao các đơn hàng nhỏ lẻ/số lượng ít/lẻ tẻ ở các khu vực tách xa nhau, dẫn đến qui trình giao hàng không được tối ưu, gây thiệt hại lớn về nhân lực (thiệt thòi cho phía nhân viên giao hàng) cũng như về mặt tài chính (chi phí vận chuyển và chuẩn bị, chi phí quản lý).

Bên cạnh đó, cần phải đảm bảo được đơn hàng không bị chậm trễ, cố tình ứ đọng. Tình trạng đơn hàng phải liên tục được cập nhật và thông báo đến phia người mua hàng để tránh gây mất uy tín cũng như giảm thiện cảm đối với phía người mua hàng.

#### 2.1.1.4 Vấn đề về thống kê báo cáo, kiểm soát luồng tiền

Như đã được nêu ở trên, việc quản lý đơn hàng bằng sổ sách giấy ở thời điểm hiện tại có xảy ra nhiều rủi ro khác nhau như:

- Nhầm lẫn hoặc sai sót trong quá trình viết thủ công

- Vấn đề chữ viết khó đọc hoặc định dạng chưa đồng nhất

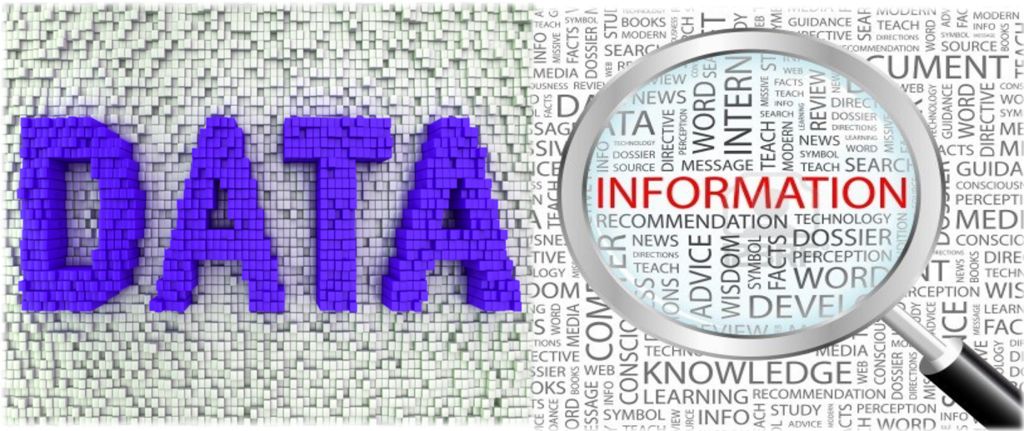
- Khó thống kê để tổng hợp và báo cáo lại nguồn thu/chi để đề xuất ra các phương hướng cũng như giải pháp kinh doanh hợp lý trong thời điểm hiện tại

- Nếu có xảy ra mất sổ sách hoặc hư hao thì sẽ ảnh hưởng đến quá trình tổng hợp và báo cáo thu chi, ngoài ra còn tiêu tốn không gian để lưu trữ lại

- Vì là viết tay thủ công nên khó có thể kiểm soát được những luồng tiền thất thoát hoặc luồng tiền ngoài ý muốn. Kiểm soát tài chính không minh bạch và rõ ràng.

### 2.1.2 Hướng giải quyết vấn đề

#### 2.1.2.1 Vấn đề về cung cấp thông tin cho người sử dụng



Hình 2. 2 Cung cấp thông tin cho người sử dụng

Xây dựng hệ thống cho phép sinh viên (người sử dụng) theo dõi được thông tin của sản phẩm/dịch vụ một cách nhanh chóng, chân thực và khách quan nhất.

Xây dựng hệ thống cho phép sinh viên (người sử dụng) theo dõi được thông tin cũng như trạng thái đơn hàng mà mình đã đặt hàng.

Xây dựng hệ thống cho phép chủ căng tin theo dõi và cập nhật được thông tin cũng như trạng thái của sản phẩm/dịch vụ

#### 2.1.2.2 Vấn đề về việc ghi nhận và xác nhận đơn hàng



Hình 2. 3 Ghi nhận và xác nhận đơn hàng

Xây dựng hệ thống cho phép sinh viên (người sử dụng) đặt hàng trực tuyến những sản phẩm/dịch vụ mà mình muốn. Từ thông tin đặt hàng mà sinh viên (người sử dụng) đó cung cấp, hệ thống sẽ tự tổng hợp lại và tạo ra đơn đặt hàng.

Xây dựng hệ thống cho phép tự động lưu trữ và thông báo đơn hàng đến chủ căng tin để tiến hành xử lý chuẩn bị đơn hàng.

Xây dựng hệ thống cho phép chủ căng tin cập nhật tình trạng đơn hàng theo các nghiệp vụ yêu cầu.

Xây dựng hệ thống cho phép chủ căng tin theo dõi được danh sách các đơn hàng cần phải xử lý/đơn hàng theo trạng thái cụ thể.

#### 2.1.2.3 Vấn đề về quản lý địa điểm giao hàng



Hình 2. 4 Quản lý địa điểm giao hàng

Xây dựng hệ thống cho phép chủ căng tin quản lý được địa chỉ giao hàng cố định, từ đó hạn chế được tình trạng địa chỉ giao hàng bị sai sót hoặc nhầm lẫn từ phía người sử dụng.

Xây dựng hệ thống cho phép chủ căng tin quản lý một lần giao hàng trong một khu vực cụ thể để tiết kiệm được tối đa chi phí vận chuyển và năng suất giao hàng lại được tối ưu nhất.

Xây dựng hệ thống cho phép chủ căng tin và nhân viên giao hàng theo dõi được danh sách các đơn hàng cần phải giao theo khu vực hoặc theo đơn hàng từ đó tiến hành giao hàng theo địa chỉ đã được mô tả trong đơn hàng.

#### 2.1.2.4 Vấn đề về thống kê báo cáo, kiểm soát luồng tiền



Hình 2. 5 Thống kê báo cáo, kiểm soát luồng tiền

Xây dựng hệ thống cho phép chủ căng tin xem được thông tin về doanh thu, các thống kê báo cáo, được tổng hợp một cách minh bạch và rõ ràng nhất nhằm kiểm soát hiệu quả luồng tiền.

Xây dựng hệ thống cho phép sinh viên (người sử dụng) thanh toán trực tuyến qua bên thứ ba, cũng như vẫn phải đảm bảo được sự minh bạch và rõ ràng.

Xây dựng hệ thống có khả năng tự động tổng hợp và báo cáo lại doanh thu cũng như tình trạng các sản phẩm dựa trên thông tin từ các đơn hàng đã được đặt, qua đó thể hiện lại nội dung cho chủ căng tin nắm bắt được, nhằm đưa ra chiến lược kinh doanh phù hợp.

### 2.1.3 Đưa ra kết luận

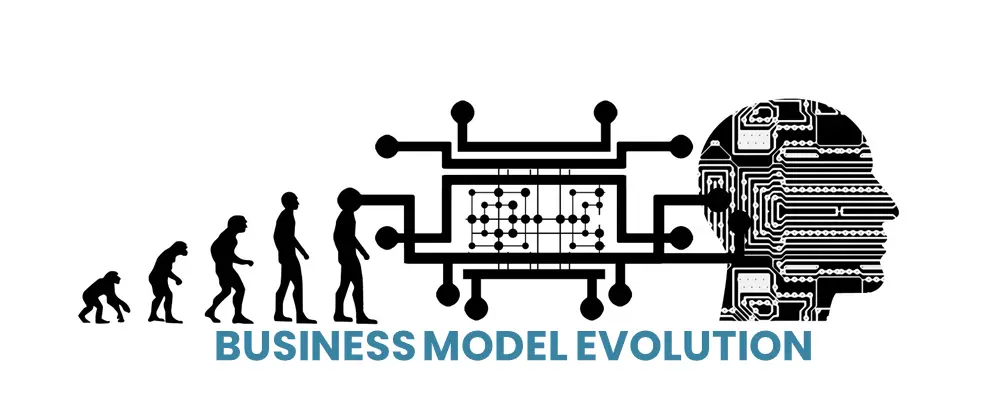
Từ những phân tích về bài toán thực tế trên của khu căng tin cho sinh viên nội trú tại ký túc xá UTC2 thì ta có thể dễ dàng thấy được rằng việc xây dựng một hệ thống/mô hình kinh doanh trực tuyến là rất cần thiết và không hề thừa thải trong giai đoạn công nghệ phát triển như hiện nay (thời đại kỹ thuật số và tự động hóa 4.0):

- Việc xây dựng một hệ thống quản lý sẽ tiết kiệm được nhiều thời gian, tiền bạc cũng như nguồn lực về con người.

- Một hệ thống quản lý tốt sẽ giúp cho người chủ kinh doanh có cái nhìn bao quát hết được các hoạt động cũng như tình hình kinh doanh của chính doanh nghiệp, qua đó mà đề ra các giải pháp hoặc phương hướng phát triển tiếp theo cho doanh nghiệp.

- Ngoài ra, việc xây dựng một hệ thống quản lý như trên sẽ giúp cho doanh nghiệp mở ra nhiều hướng kinh doanh và phát triển hơn, dễ dàng tiếp cận với người sử dụng hơn, cũng như đẩy mạnh hiệu quả kinh doanh. Khi mà chỉ cần bỏ một khoảng đầu tư nhỏ nhưng sẽ mang lại một hiệu quả vô cùng lớn.

- Một hệ thống mới, một mô hình quản lý mới, một tầm nhìn mới, cũng như những hướng phát triển mới.



Hình 2. 6 Mô hình phát triển kinh doanh hiện đại

## 2.2 Xây dựng và triển khai hệ thống

### 2.2.1 Thiết kế cơ sở dữ liệu

#### 2.2.1.1 Mục tiêu đề ra

1. Quản lý thông tin địa chỉ

2. Quản lý thông tin phân loại của sản phẩm

3. Quản lý thông tin đơn hàng

4. Quản lý thông tin đơn hàng giao hàng theo khu vực

5. Quản lý thông tin phân quyền người dùng

6. Quản lý thông tin của sản phẩm

7. Quản lý thông tin người dùng

#### 2.2.1.2 Xác định thực thể và các thuộc tính của chúng

Sau đây là một số thực thể tiêu biểu của hệ thống, những thực thể khác có thể tham khảo thêm tại phần mã nguồn của đề tài.

Đơn hàng gồm các thuộc tính:

* Mã đơn hàng
* Chi tiết đơn hàng
* Trạng thái đơn hàng
* Loại hình hoặc phương thức thanh toán
* Số điện thoại người nhận hàng
* Tổng tiền của đơn hàng
* Tổng số lượng sản phẩm có trong đơn hàng
* Tên người nhận hàng
* Người tạo đơn hàng
* Địa chỉ giao hàng
* Ghi chú
* Thời gian tạo đơn hàng

Thông tin của sản phẩm gồm các thuộc tính:

* Mã sản phẩm
* Tên sản phẩm
* Mô tả sản phẩm
* Mức giảm giá
* Phân loại sản phẩm
* Giá thành sản phẩm
* Trạng thái của sản phẩm
* Thông tin hiển thị của sản phẩm (hình ảnh/url)

#### 2.2.1.3 Xác định các ràng buộc toàn vẹn về nghiệp vụ của hệ thống

R1: Một địa chỉ có thể có nhiều người cùng ở và một người dùng cũng có thể có nhiều địa chỉ khác nhau

R2: Mỗi đơn hàng khi được tạo phải quản lý được người tạo và thời gian tạo đơn hàng

R3: Một người dùng có thể có nhiều phân quyền khác nhau

R4: Mỗi đơn hàng khi lập ra phải có tối thiểu một chi tiết đơn hàng (tối thiểu một sản phẩm được chọn)

R5: Mỗi đơn hàng giao theo khu vực phải có tối thiểu một đơn hàng được giao

R6: Nhiều đơn hàng khác nhau có thể cùng giao về cùng một khu vực

R7: Mỗi sản phẩm có duy nhất một mã

R8: Mỗi người dùng có duy nhất một tên đăng nhập và mã người dùng

R9: Trong mỗi đơn hàng bắt buộc phải có địa chỉ giao hàng và số điện thoại người nhận

R10: Người dùng không thể đặt hàng sản phẩm/dịch vụ nếu sản phẩm/dịch vụ đó đã bán hết

R11: Người dùng khi thanh toán đơn hàng có thể chọn phương thức thanh toán trực tuyến, tuy nhiên nếu thanh toán thất bại thì đơn hàng sẽ được tính là thanh toán trực tiếp bằng tiền mặt

R12: Người dùng không thể đăng nhập hệ thống nếu tài khoản của họ đã bị quản trị viên vô hiệu hóa (khóa)

R13: Người dùng khi tạo giỏ hàng thì không cần đăng nhập, tuy nhiên khi tiến hành thanh toán thì bắt buộc phải đăng nhập

R14: Ngày lập đơn hàng không được quá ngày hiện tại

R15: Người lập đơn hàng phải tồn tại trong hệ thống và tài khoản không bị khóa

R16: Một người dùng có thể tạo được nhiều đơn hàng hoặc không tạo đơn hàng nào

R17: Người dùng không thể đặt hàng nếu ngoài khung giờ phục vụ của hệ thống, tuy nhiên vẫn xem được các thông tin trước đó

#### 2.2.1.4 Xây dựng thành bảng dữ liệu trong cơ sở dữ liệu

Sau đây là một số Table (bảng dữ liệu) tiêu biểu trong database của hệ thống, những Table khác có thể tham khảo thêm tại phần mã nguồn của đề tài.

##### 2.2.1.4.1 Bảng đơn hàng (Orders)



Hình 3. 1 Table Đơn hàng

|  |  |  |
| --- | --- | --- |
| **STT** | **Cột** | **Giá trị thể hiện của cột** |
| 1 | id | Là khóa chính của table |
| 2 | address | Địa chỉ giao hàng |
| 3 | cancel\_reasons | Lý do hủy đơn |
| 4 | create\_date | Ngày khởi tạo đơn hàng |
| 5 | create\_user | Người khởi tạo đơn hàng |
| 6 | is\_cancel | Đơn hàng đã bị hủy hay chưa |
| 7 | is\_completed | Đơn hàng đã hoàn thành hay chưa |
| 8 | is\_paid | Đơn hàng đã được thanh toán hay chưa |
| 9 | note | Ghi chú cho đơn hàng |
| 10 | order\_code | Mã đơn hàng |
| 11 | order\_status | Trạng thái của đơn hàng |
| 12 | pay\_type | Phương thức thanh toán |
| 13 | phone\_number | Số điện thoại người nhận |
| 14 | total\_amount | Tổng tiền của đơn hàng phải thanh toán |
| 15 | total\_qty | Tổng sản phẩm có trong đơn hàng |
| 16 | update\_date | Ngày cập nhật đơn hàng |
| 17 | update\_user | Người cập nhật đơn hàng |
| 18 | user\_display\_name | Tên của người nhận hàng |

Bảng 3. 1 Table Đơn hàng

Bảng này được lập ra với mục tiêu là “**Quản lý thông tin đơn hàng**”, là thứ sẽ được sinh ra khi mà người dùng tiến hành đăng ký khởi tạo đơn hàng từ hệ thống trong quá trình đặt món và tiến hành thanh toán.

##### 2.2.1.4.2 Bảng sản phẩm (Products)



Hình 3. 2 Table Sản phẩm

|  |  |  |
| --- | --- | --- |
| **STT** | **Cột** | **Giá trị thể hiện của cột** |
| 1 | id | Là khóa chính của table |
| 2 | category | Phân loại sản phẩm |
| 3 | create\_date | Ngày khởi tạo |
| 4 | create\_user | Người khởi tạo |
| 5 | description | Mô tả của sản phẩm |
| 6 | discount\_number | Giá trị giảm giá trừ trực tiếp vào giá sản phẩm |
| 7 | discount\_percent | Giá trị giảm giá sản phẩm tính theo phần trăm |
| 8 | discount\_type | Hình thức giảm giá |
| 9 | final\_price | Giá sau khi đã giảm giá |
| 10 | is\_deleted | Sản phẩm này đã bị xóa hay chưa |
| 11 | is\_discount | Sản phẩm này có đang giảm giá hay không |
| 12 | is\_inventory | Sản phẩm này còn hàng để bán hay không |
| 13 | price | Giá gốc của sản phẩm trước khi giảm giá |
| 14 | product\_code | Mã sản phẩm |
| 15 | product\_name | Tên sản phẩm |
| 16 | type | Phân nhóm hiển thị cho sản phẩm |
| 17 | update\_date | Ngày chỉnh sửa sản phẩm |
| 18 | update\_user | Người chỉnh sửa sản phẩm |
| 19 | url\_avatar | Ảnh hiển thị của sản phẩm |

Bảng 3. 2 Table Sản phẩm

Bảng này được lập ra với mục tiêu là “**Quản lý thông tin sản phẩm**”, là thứ sẽ cung cấp thông tin trực quan và rõ ràng nhất khi người dùng muốn xem thông tin về sản phẩm/dịch vụ hiện có trên hệ thống. Bên cạnh đó người quản lý cũng theo dõi được tổng quan tình trạng của sản phẩm bên trong doanh nghiệp của mình.

##### 2.2.1.4.3 Bảng chi tiết đơn hàng (Order Details)



Hình 3. 3 Table Chi tiết đơn hàng

|  |  |  |
| --- | --- | --- |
| **STT** | **Cột** | **Giá trị thể hiện của cột** |
| 1 | id | Là khóa chính của table |
| 2 | qty | Số lượng đặt hàng cho sản phẩm |
| 3 | product\_id | Id của sản phẩm được đặt |

Bảng 3. 3 Table Chi tiết đơn hàng

Bảng này được lập ra giúp cho việc “**Quản lý thông tin đơn hàng**”, thông qua bảng này ta có thể dễ dàng quản lý trong một đơn hàng khi được lập ra sẽ có những sản phẩm/dịch vụ nào được người dùng chọn mua, và mỗi sản phẩm/dịch vụ đó được đặt mua với số lượng cụ thể là bao nhiêu trong mỗi đơn hàng.

#### 2.2.1.5 Tổng quát quá trình thiết kế

Gồm có **8 table** quản lý chính được tạo ra từ việc phân tích mục tiêu đã đề ra trước đó gồm:

1. Addresses
2. Categories
3. Order\_details
4. Order\_lots
5. Orders
6. Products
7. Roles
8. Users

Ngoài ra còn sử dụng thêm 1 table để quản lý hệ thống:

1. System\_parameters

Khi sử dụng Spring Boot để cấu hình cơ sở dữ liệu thì hệ thống sẽ tự sinh ra cho ta **4 table phụ** dùng để quản lý dựa theo **8 table chính** ở trên một cách tự động hóa mà không cần chúng ta can thiệp gì:

1. Order\_lots\_details
2. Orders\_details
3. Users\_addresses
4. Users\_roles

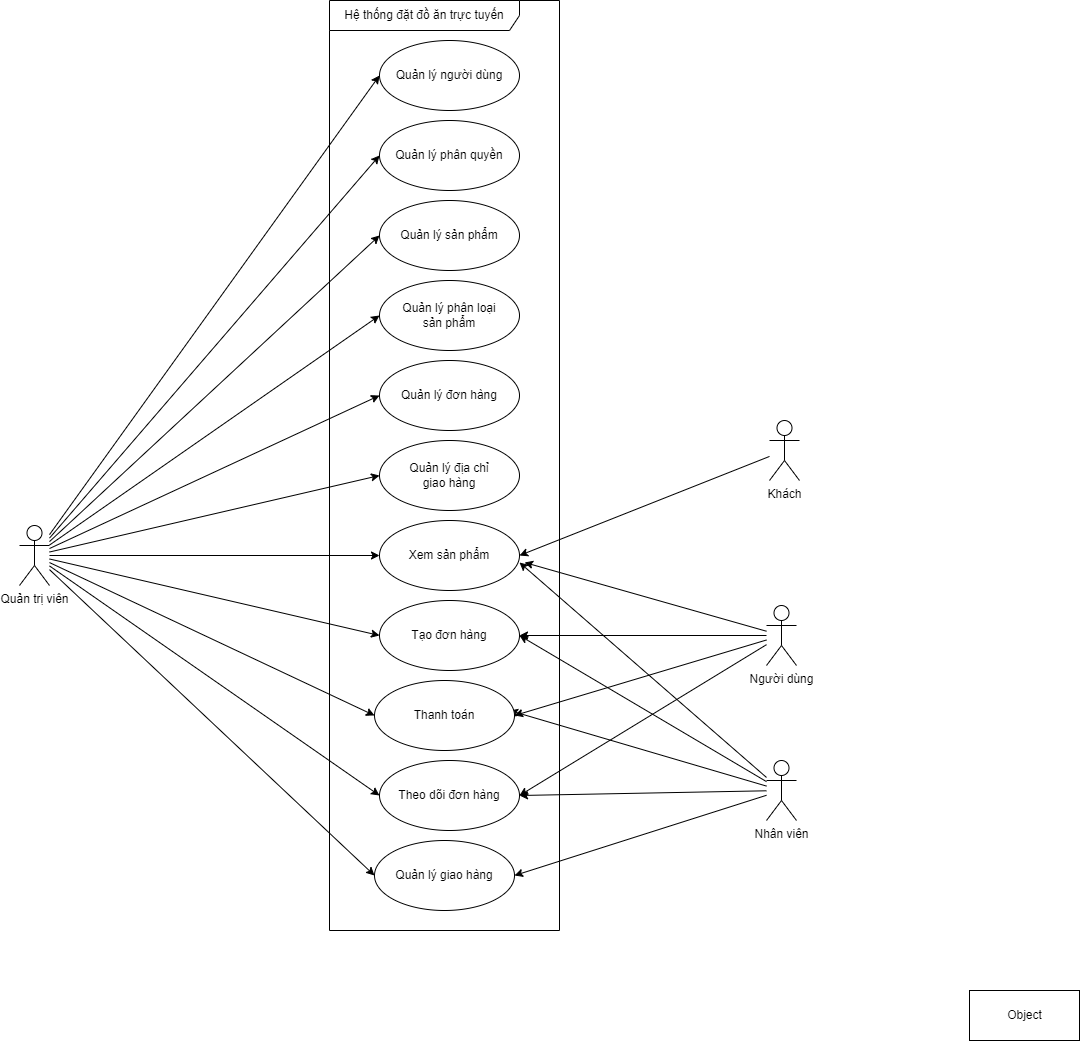
#### 2.2.1.6 Kết quả đạt được

Sau quá trình đánh giá và phân tích, ta đã xây dựng được mô hình cơ sở dữ liệu có thể đáp ứng được 100% mục tiêu đã đề ra ở phía trên.

Thêm vào đó ta đã tạo thêm một table riêng để cấu hình chung cho toàn bộ hệ thống của mình, điều này khiến cho việc mở trộng hệ thống sau này dễ dàng và thuận tiện hơn rất nhiều.

Vì sử dụng Spring Boot nên việc cấu hình và quản trị cơ sở dữ liệu sẽ trở nên đơn giản hơn bao giờ hết, đây cũng là một trong những ưu điểm/điểm mạnh của framework này.

### 2.2.2 Thiết kế sơ đồ Use Case tổng quát



Hình 3. 8 Sơ đô Use Case tổng quát

Dựa theo sơ đồ Use Case được lập ra ở trên ta có thể thấy hệ thống được phân cấp ra thành **4 thành phần** (4 cấp bậc) có thể sử dụng và truy cập:

- ***Quản trị viên***: người có thể truy cập và quản lý tất cả chức năng/tính năng của hệ thống.

- ***Khách***: người truy cập vào hệ thống nhưng chưa đăng nhập, chỉ có thể xem một số thống tin tổng quan về doanh nghiệp và sản phảm, có thể tiến hành đặt hàng tuy nhiên sẽ bị yêu cầu đăng nhập khi tiến hành thanh toán (không thể tạo đơn hàng).

- ***Người dùng***: người truy cập vào hệ thống sau khi đã đăng nhập, có thể xem thông tin tổng quan về doanh nghiệp và sản phẩm, có thể tiến hành đặt hàng và thanh toán (có thể tạo đơn hàng), có thể theo dõi các đơn hàng mình đã đặt và quản lý được thông tin cá nhân của tài khoản.

- ***Nhân viên (nhân viên giao hàng)***: người truy cập vào hệ thống sau khi đã đăng nhập, có thể xem thông tin tổng quan về doanh nghiệp và sản phẩm, có thể tiến hành đặt hàng và thanh toán (có thể tạo đơn hàng), có thể theo dõi các đơn hàng mình đã đặt và quản lý được thông tin cá nhân của tài khoản, có thể theo dõi và quản lý được các đơn hàng đang trong trạng thái cần xử lý vận chuyển.

### 2.2.3 Thiết kế phần xử lý Back-End

#### 2.2.3.1 Thiết kế Model

Trong **JPA – Hibernate**, một Entity (thực thể) là một Java class tương ứng với một table trong database, mỗi thuộc tính của Entity tương ứng với một dòng trong bảng.

**JPA** yêu cầu một Entity class phải cung cấp đầy đủ các thuôc tính dữ liệu để ánh xạ giữa Entity class và table trong database như: tên cột, kiểu dữ liệu, khoá chính, khoá ngoại, … ngoài ra còn có thêm nhiều vấn đề khác.

Sau đây là một số Entity class tiêu biểu của hệ thống, những Entity class khác có thể tham khảo thêm tại phần mã nguồn của đề tài.

**Product**

|  |
| --- |
| package springboot.model;  import java.io.Serializable;  import java.math.BigDecimal;  import java.util.Date;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  import lombok.AllArgsConstructor;  import lombok.Data;  import lombok.Getter;  import lombok.NoArgsConstructor;  import lombok.Setter;  *@Entity*  *@Data*  *@Setter*  *@Getter*  *@NoArgsConstructor*  *@AllArgsConstructor*  *@Table*(name = "products")  *@SuppressWarnings*("serial")  public class Product implements Serializable {  *@Id*  *@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)  *@Column*(nullable = false, updatable = false)  private Long id;  *@Column*(nullable = false, updatable = false)  private String productCode;  *@Column*(nullable = false)  private String productName;  *@Column*(nullable = false)  private BigDecimal price;  private BigDecimal finalPrice;  private BigDecimal discountNumber;  private BigDecimal discountPercent;  private String urlAvatar;  private String description;  private String category;  private String type;  private String discountType;  private boolean isDeleted;  private boolean isDiscount;  private boolean isInventory = true;  *@Column*(nullable = false, updatable = false)  private String createUser;  private String updateUser;  *@Column*(nullable = false, updatable = false)  private Date createDate;  private Date updateDate;  } |

Bảng 3. 10 Product

#### 2.2.3.2 Thiết kế Repository

**Repository** trong JPA có thể hiểu là một kho lưu trữ khép kín, trong đó chứa các phương thức xử lý như lưu trữ, truy xuất hoặc tìm kiếm dữ liệu, … từ database thông qua các Entity class đã được khai báo từ phần Model ở trên.

Sau đây là một số Repository tiêu biểu của hệ thống, những Repository khác có thể tham khảo thêm tại phần mã nguồn của đề tài.

**OrderRepository**

|  |
| --- |
| package springboot.repository;  import java.util.Date;  import java.util.List;  import java.util.Optional;  import org.springframework.data.domain.Pageable;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.data.jpa.repository.Query;  import org.springframework.data.repository.query.Param;  import org.springframework.stereotype.Repository;  import springboot.model.Order;  *@Repository*  public interface OrderRepository extends JpaRepository<Order, Long> {  Optional<Order> findById(Long id);  void deleteById(Long id);  List<Order> findByOrderStatusIn(List<String> value);  List<Order> findByOrderStatus(String value);  List<Order> findByOrderStatusInAndIsCompleted(List<String> value, boolean isCompleted);  List<Order> findByOrderStatusAndIsCompleted(String value, boolean isCompleted);  List<Order> findByCreateDateGreaterThanEqual(Date today);  *@Query*("select o from Order o where o.createUser = :user AND o.createDate >= :dateFrom AND o.createDate < :dateTo")  List<Order> findByCreateUserAndCreateDateBetween(*@Param*("user") String user, *@Param*("dateFrom") Date dateFrom,  *@Param*("dateTo") Date dateTo);  *@Query*("select o from Order o where o.createUser = :user AND o.orderStatus = :status AND o.createDate >= :dateFrom AND o.createDate < :dateTo")  List<Order> findByCreateUserAndOrderStatusAndCreateDateBetween(*@Param*("user") String user,  *@Param*("status") String status, *@Param*("dateFrom") Date dateFrom, *@Param*("dateTo") Date dateTo);  *@Query*("select o from Order o where o.createDate >= :today AND o.isCompleted = false AND (o.orderStatus = 'WAITFORPAY' OR o.orderStatus = 'PAID')")  List<Order> getJustPaid(*@Param*("today") Date today);  *@Query*("select o from Order o where o.createDate >= :today AND o.isCompleted = false AND o.orderStatus = 'PREPARING'")  List<Order> getJustRepaired(*@Param*("today") Date today);  *@Query*("select o from Order o where o.createDate >= :today AND o.isCompleted = false AND o.orderStatus = 'DELIVERY'")  List<Order> getJustDelivered(*@Param*("today") Date today);  *@Query*("select o from Order o where o.isCompleted = :isCompleted AND o.createDate >= :dateFrom AND o.createDate < :dateTo")  List<Order> findByIsCompletedAndCreateDateBetween(*@Param*("isCompleted") boolean isCompleted,  *@Param*("dateFrom") Date dateFrom, *@Param*("dateTo") Date dateTo);  *@Query*("select o from Order o where o.payType = :payType AND o.createUser = :createUser AND o.createDate >= :today ORDER BY o.orderCode DESC")  List<Order> getNewestOrderMomoByUserName(*@Param*("payType") String payType, *@Param*("createUser") String createUser,  *@Param*("today") Date today, Pageable pageable);  } |

Bảng 3. 11 OrderRepository

#### 2.2.3.3 Thiết kế Service

**Service** là nơi để xử lý các nghiệp vụ chính của hệ thống, mỗi Service đảm nhận một vai trò và chức năng khác nhau trong mỗi tính năng mà hệ thống có thể cung cấp được cho người sử dụng.

Sau đây là một số Service tiêu biểu của hệ thống, những Service khác có thể tham khảo thêm tại phần mã nguồn của đề tài.

**ProductService**

|  |
| --- |
| package springboot.service;  import java.math.BigDecimal;  import java.util.ArrayList;  import java.util.Date;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import lombok.RequiredArgsConstructor;  import lombok.extern.slf4j.Slf4j;  import springboot.enums.MConst;  import springboot.exception.EntityNotFoundException;  import springboot.model.Category;  import springboot.model.Product;  import springboot.repository.ProductRepository;  *@Service*  *@Slf4j*  *@RequiredArgsConstructor*  *@Transactional*  public class ProductService {  *@Autowired*  private final ProductRepository productRepo;  *@Autowired*  private final CommonService commonService;  *@Autowired*  private final CategoryService categoryService;  public Product add(Product product) {  // tạo ProductCode  if (product.getProductCode() == null || product.getProductCode().isEmpty()) {  String createCode = "PD\_" + String.*format*("%05d", productRepo.count() + 1);  product.setProductCode(createCode);  }  // tạo FinalPrice  if (product.isDiscount()) {  switch (product.getDiscountType()) {  case MConst.***NUMBER***:  product.setFinalPrice(product.getPrice().subtract(product.getDiscountNumber()));  product.setDiscountPercent(BigDecimal.***ZERO***);  break;  case MConst.***PERCENT***:  BigDecimal hundred = new BigDecimal(100);  BigDecimal discount = product.getPrice().multiply(product.getDiscountPercent());  BigDecimal discountHundred = discount.divide(hundred);  product.setFinalPrice(product.getPrice().subtract(discountHundred));  product.setDiscountNumber(BigDecimal.***ZERO***);  break;  }  } else {  product.setFinalPrice(product.getPrice());  product.setDiscountPercent(BigDecimal.***ZERO***);  product.setDiscountNumber(BigDecimal.***ZERO***);  }  product.setCreateDate(new Date());  product.setCreateUser(commonService.getCurrentUser());  // thêm mới Category nếu chưa tồn tại  addCategory(product.getCategory(), product.getCategory(), product.getType());  ***log***.info("Added new Product: {}", product.getProductName());  return productRepo.save(product);  }  private void addCategory(String key, String value, String type) {  if (!categoryService.isExist(key, value, type)) {  categoryService.add(new Category(key, value, type));  }  }  public List<Product> getProducts() {  try {  return productRepo.findAll();  } catch (Exception e) {  ***log***.error("Error: {}", e.getMessage());  return new ArrayList<>();  }  }  public List<Product> getByTypeAndIsDelete(String type, boolean isDelete) {  try {  return productRepo.findByTypeAndIsDeleted(type, isDelete);  } catch (Exception e) {  ***log***.error("Error: {}", e.getMessage());  return new ArrayList<>();  }  }  public List<Product> getByIsDelete(boolean isDelete) {  try {  return productRepo.findByIsDeleted(isDelete);  } catch (Exception e) {  ***log***.error("Error: {}", e.getMessage());  return new ArrayList<>();  }  }  public Product update(Product product) {  Product entity = findById(product.getId());  product.setCreateUser(entity.getCreateUser());  product.setCreateDate(entity.getCreateDate());  product.setUpdateDate(new Date());  product.setUpdateUser(commonService.getCurrentUser());  // tạo FinalPrice  if (product.isDiscount()) {  switch (product.getDiscountType()) {  case MConst.***NUMBER***:  product.setFinalPrice(product.getPrice().subtract(product.getDiscountNumber()));  product.setDiscountPercent(BigDecimal.***ZERO***);  break;  case MConst.***PERCENT***:  BigDecimal hundred = new BigDecimal(100);  BigDecimal discount = product.getPrice().multiply(product.getDiscountPercent());  BigDecimal discountHundred = discount.divide(hundred);  product.setFinalPrice(product.getPrice().subtract(discountHundred));  product.setDiscountNumber(BigDecimal.***ZERO***);  break;  }  } else {  product.setFinalPrice(product.getPrice());  product.setDiscountPercent(BigDecimal.***ZERO***);  product.setDiscountNumber(BigDecimal.***ZERO***);  }  // thêm mới Category nếu chưa tồn tại  addCategory(product.getCategory(), product.getCategory(), product.getType());  ***log***.info("Updated Product: {} by {}", new Object[] { entity.getProductCode(), entity.getUpdateUser() });  return productRepo.save(product);  }  public Product findById(Long id) {  return productRepo.findById(id)  .orElseThrow(() -> new EntityNotFoundException("Product with id: " + id + "was not found!"));  }  public Product findByProductCode(String code) {  return productRepo.findByProductCode(code)  .orElseThrow(() -> new EntityNotFoundException("Product with code: " + code + "was not found!"));  }  public void softDeleteById(Long id) {  Product entity = findById(id);  entity.setUpdateDate(new Date());  entity.setUpdateUser(commonService.getCurrentUser());  entity.setDeleted(true);  ***log***.info("Soft delete a Product: {}", entity.getProductCode());  productRepo.save(entity);  }  public void deleteById(Long id) {  productRepo.deleteById(id);  }  } |

Bảng 3. 15 ProductService

#### 2.2.3.4 Thiết kế Controller

**Controller** là nơi giao tiếp với Front-End thông qua các API đã được định nghĩa riêng cho từng nghiệp vụ.

Sau đây là một số Controller tiêu biểu của hệ thống, những Controller khác có thể tham khảo thêm tại phần mã nguồn của đề tài.

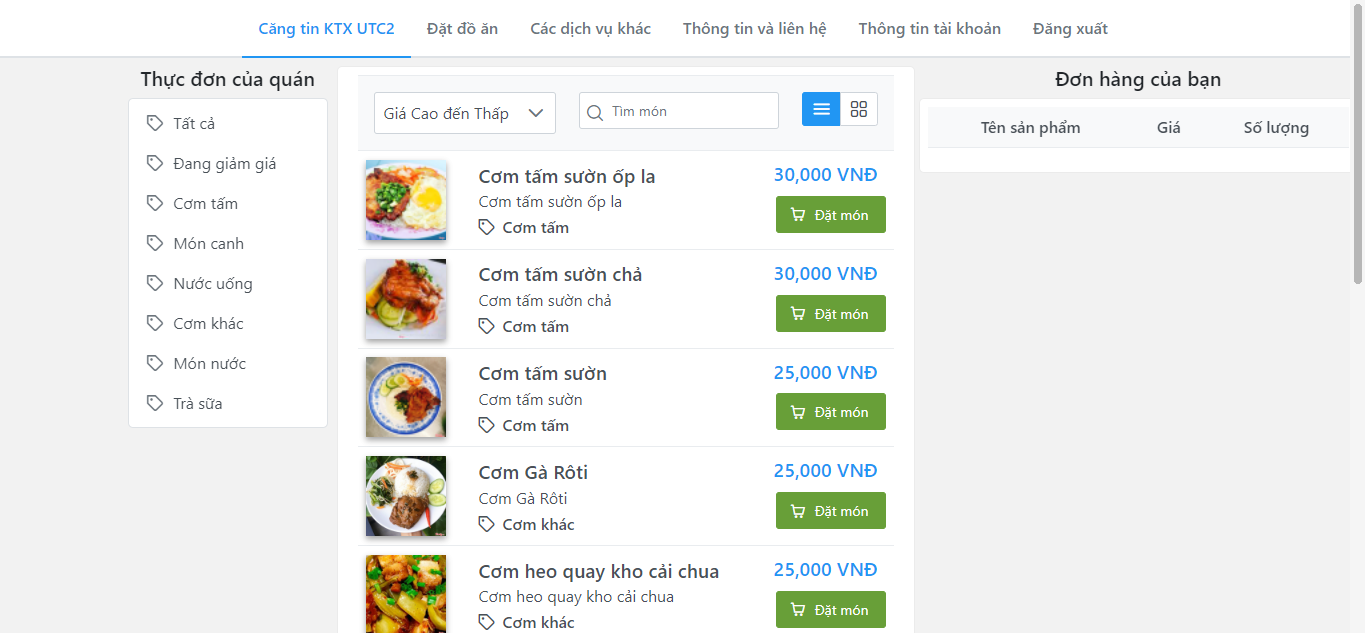
**OrderController**

|  |
| --- |
| package springboot.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.HttpStatus;  import org.springframework.http.ResponseEntity;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.PutMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  import lombok.RequiredArgsConstructor;  import springboot.model.Order;  import springboot.model.OrderLot;  import springboot.service.OrderLotService;  import springboot.service.OrderService;  *@RestController*  *@RequiredArgsConstructor*  *@RequestMapping*("/order")  public class OrderController {  *@Autowired*  private final OrderService orderService;  *@Autowired*  private final OrderLotService lotService;  *@PostMapping*("/add")  public ResponseEntity<Order> add(*@RequestBody* Order order) {  Order newOrder = orderService.add(order);  return new ResponseEntity<>(newOrder, *HttpStatus*.***CREATED***);  }  *@GetMapping*("/all")  public ResponseEntity<List<Order>> getOrders() {  List<Order> orders = orderService.getOrders();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@GetMapping*("/all/user/status={status}&dateFrom={dateFrom}&dateTo={dateTo}")  public ResponseEntity<List<Order>> getOrdersOfUser(*@PathVariable*("status") String status,  *@PathVariable*("dateFrom") String dateFrom, *@PathVariable*("dateTo") String dateTo) {  List<Order> orders = orderService.getOrdersOfUser(status, dateFrom, dateTo);  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@GetMapping*("/all/just/paid")  public ResponseEntity<List<Order>> getOrdersJustPaid() {  List<Order> orders = orderService.getOrdersJustPaid();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@GetMapping*("/all/just/repaired")  public ResponseEntity<List<Order>> getOrdersJustRepaired() {  List<Order> orders = orderService.getOrdersJustRepaired();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@GetMapping*("/all/just/delivered")  public ResponseEntity<List<Order>> getOrdersJustDelivered() {  List<Order> orders = orderService.getOrdersJustDelivered();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@PutMapping*("update/status/id={id}&status={status}")  public ResponseEntity<Order> updateStatusByIdAndStatus(*@PathVariable*("id") Long id,  *@PathVariable*("status") String status) {  Order updatedOrder = orderService.updateStatusByIdAndStatus(id, status);  return new ResponseEntity<>(updatedOrder, *HttpStatus*.***OK***);  }  *@PutMapping*("update/delivery")  public ResponseEntity<OrderLot> deliveryOrders(*@RequestBody* List<Long> idList) {  OrderLot lot = orderService.deliveryOrders(idList);  return new ResponseEntity<>(lot, *HttpStatus*.***OK***);  }  *@PutMapping*("update/complete")  public ResponseEntity<Order> completeOrders(*@RequestBody* List<Long> idList) {  orderService.completeOrders(idList);  return new ResponseEntity<>(*HttpStatus*.***OK***);  }  *@GetMapping*("/lot/all")  public ResponseEntity<List<OrderLot>> getOrderLots() {  List<OrderLot> orders = lotService.getOrderLots();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@GetMapping*("/lot/all/incompleted")  public ResponseEntity<List<OrderLot>> getOrderLotsIncompleted() {  List<OrderLot> orders = lotService.getOrderLotsIncompleted();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@GetMapping*("/lot/all/just/repaired")  public ResponseEntity<List<OrderLot>> getOrderLotsJustRepaired() {  List<OrderLot> orders = lotService.getOrderLotsJustRepaired();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  *@PostMapping*("/lot/add")  public ResponseEntity<OrderLot> add(*@RequestBody* OrderLot orderLot) {  OrderLot newLot = lotService.add(orderLot);  return new ResponseEntity<>(newLot, *HttpStatus*.***CREATED***);  }  *@PutMapping*("/lot/update/delivery")  public ResponseEntity<OrderLot> deliveryLot(*@RequestBody* Long id) {  OrderLot lot = lotService.deliveryLot(id);  return new ResponseEntity<>(lot, *HttpStatus*.***OK***);  }  *@GetMapping*("/report/week")  public ResponseEntity<List<List<Order>>> getOrdersReportThisWeek() {  List<List<Order>> result = orderService.getOrdersReportWeek();  return new ResponseEntity<>(result, *HttpStatus*.***OK***);  }  *@GetMapping*("/report/last/week")  public ResponseEntity<List<List<Order>>> getOrdersReportLastWeek() {  List<List<Order>> result = orderService.getOrdersReportLastWeek();  return new ResponseEntity<>(result, *HttpStatus*.***OK***);  }  *@GetMapping*("/report/today")  public ResponseEntity<List<Order>> getOrdersReportToday() {  List<Order> result = orderService.getOrdersReportToday();  return new ResponseEntity<>(result, *HttpStatus*.***OK***);  }  *@GetMapping*("/update/momo")  public ResponseEntity<List<Order>> updateMomoStatus() {  List<Order> orders = orderService.updateMomoStatus();  return new ResponseEntity<>(orders, *HttpStatus*.***OK***);  }  } |

Bảng 3. 18 OrderController

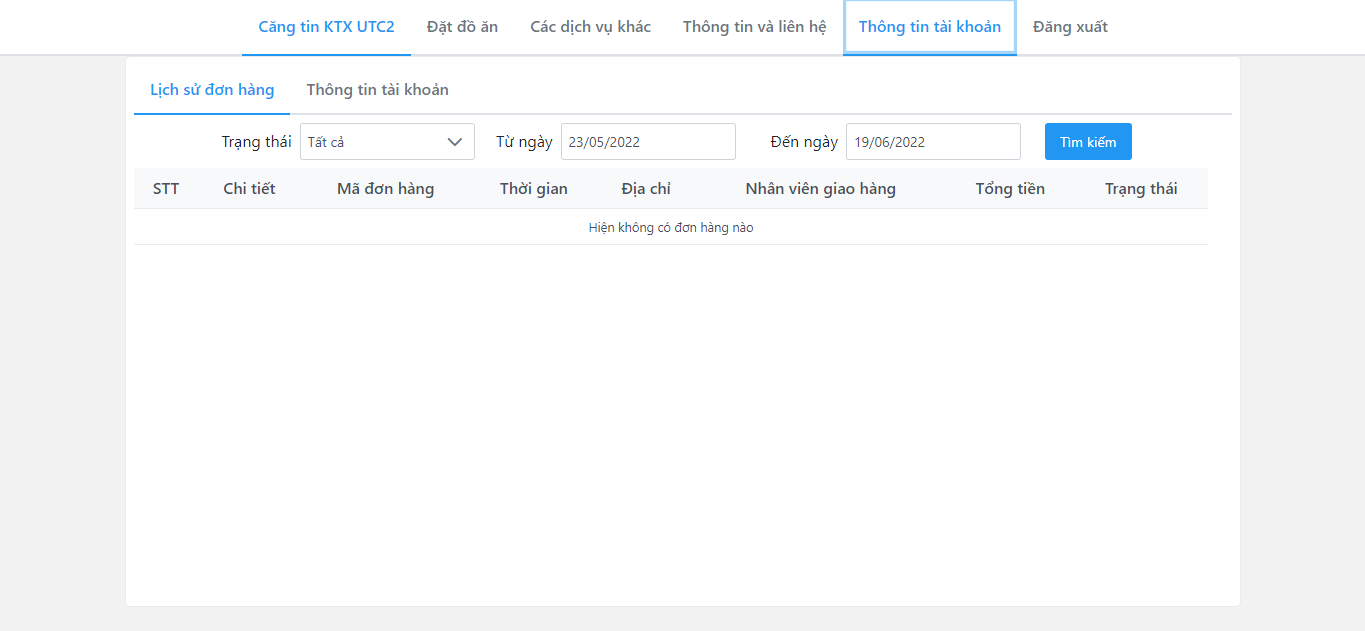
### 2.2.4 Thiết kế phần xử lý Front-End

#### 2.2.4.1 Giao điện xem sản phẩm



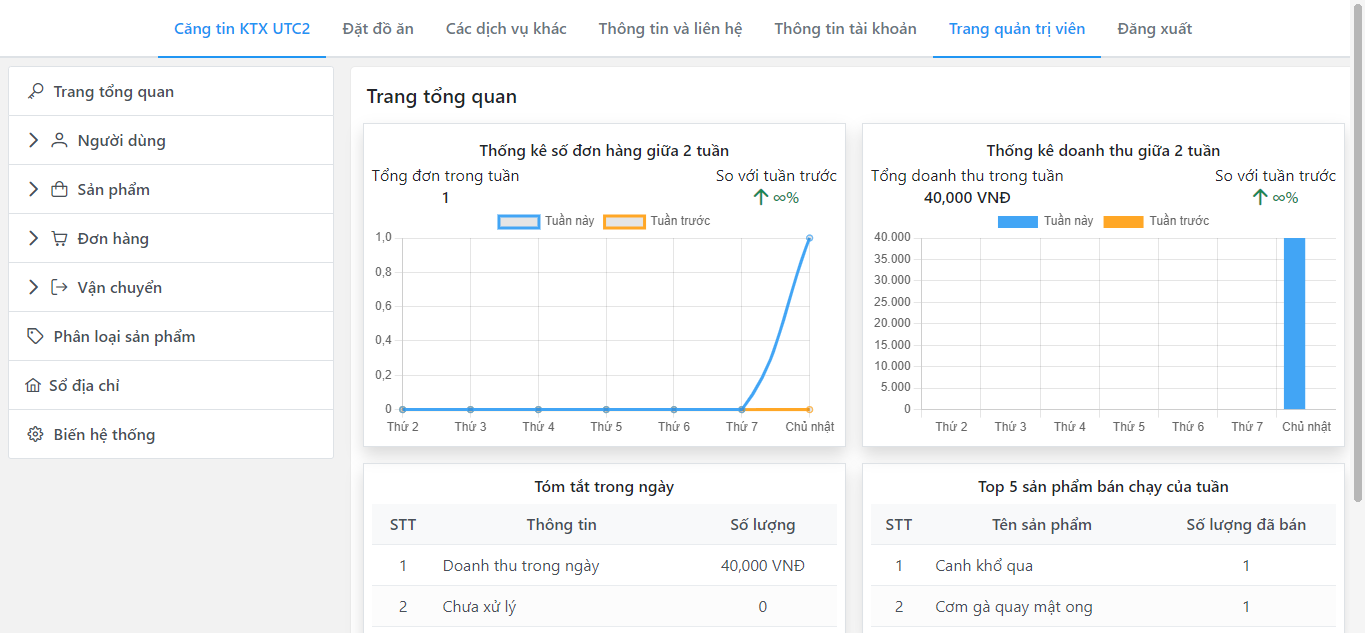
Hình 3. 9 Giao diện xem sản phẩm (đồ ăn)

#### 2.2.4.3 Giao diện xem thông tin đơn hàng đã đặt

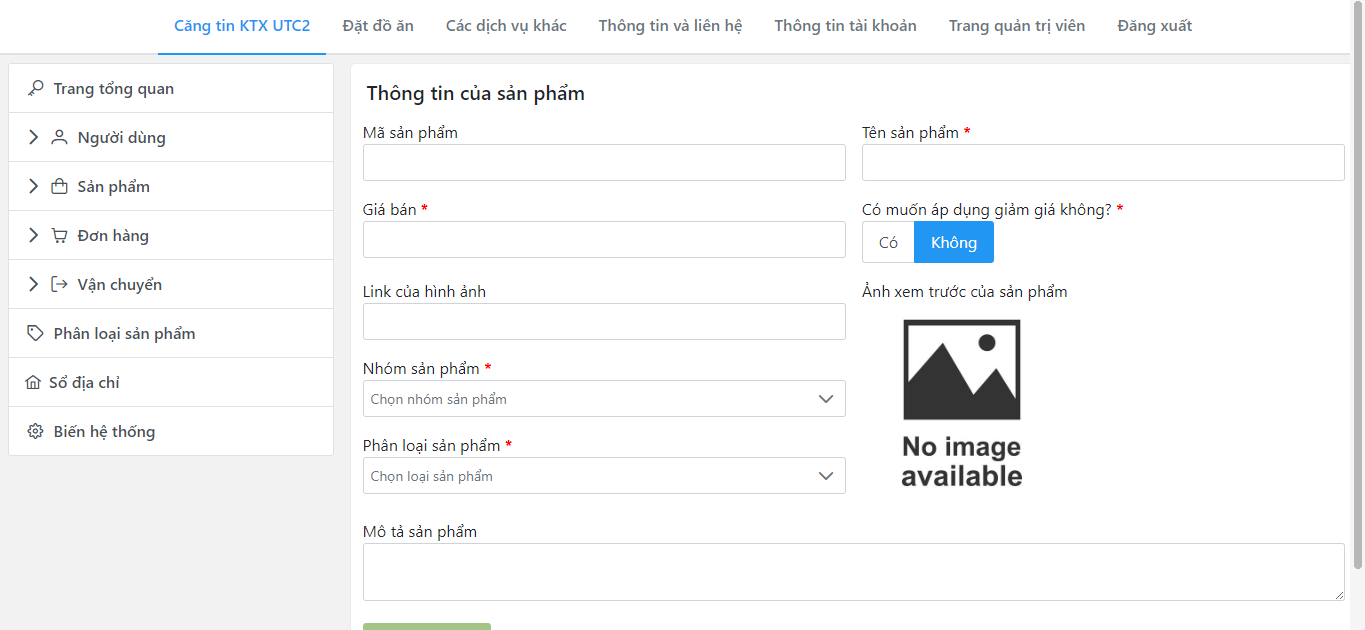


Hình 3. 12 Giao diện xem thông tin đơn hàng đã đặt

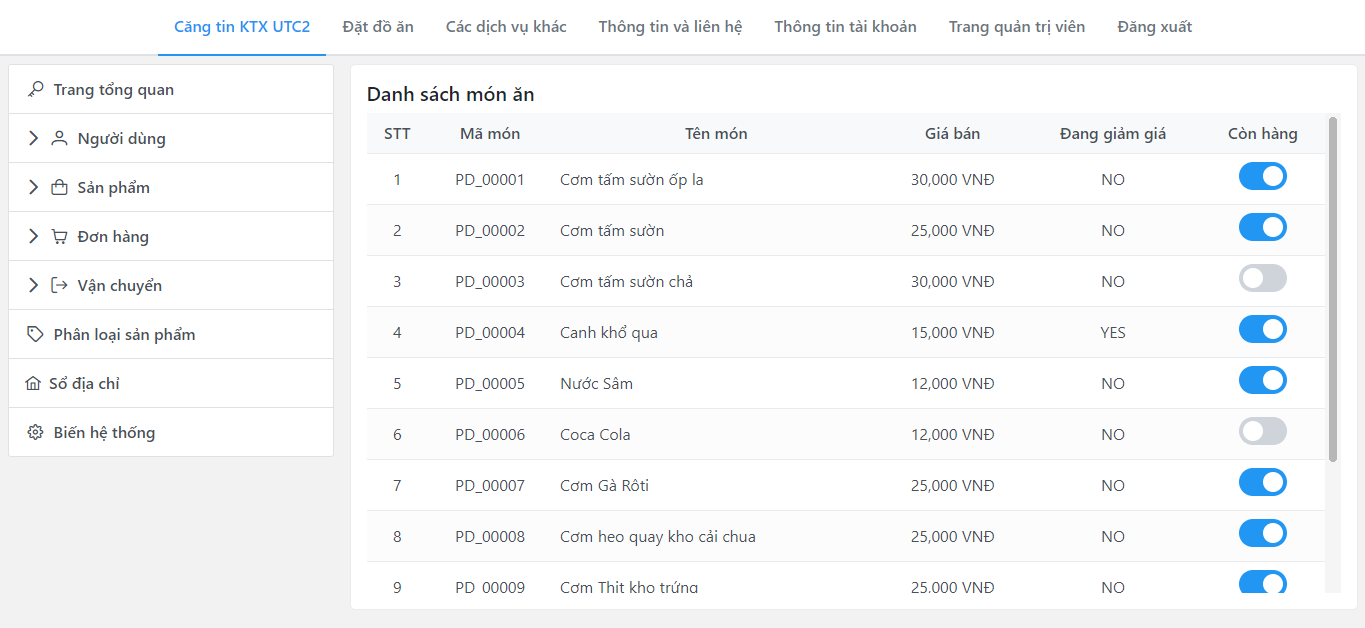
#### 2.2.4.7 Giao diện trang quản trị viên



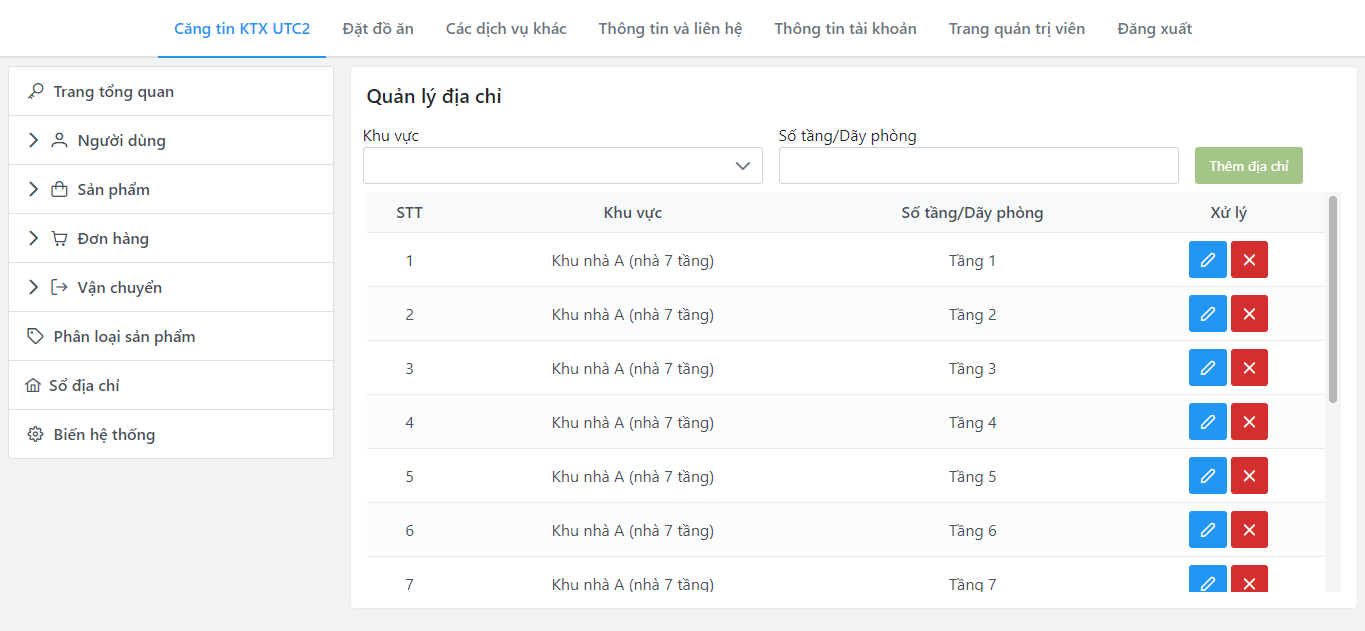
Hình 3. 16 Giao diện trang quản trị viên



Hình 3. 18 Giao diện Thêm sản phẩm



Hình 3. 19 Giao diện Danh sách món ăn



Hình 3. 27 Giao diện Quản lý địa chỉ giao hàng

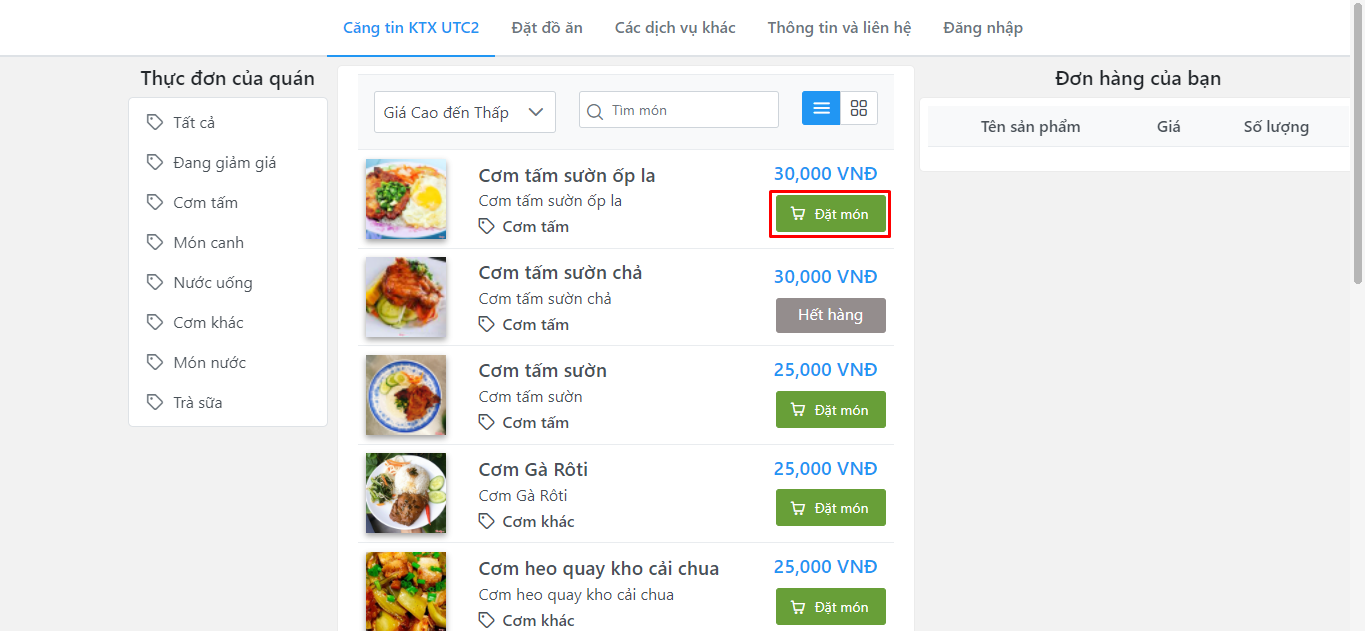
#### 2.2.4.8 Tích hợp thanh toán Momo [6]

|  |
| --- |
| import { HttpClient, HttpHeaders } from "@angular/common/http";  import { Injectable } from "@angular/core";  import jsSHA from "jssha";  import { Observable } from "rxjs";  import { environment } from "src/environments/environment";  @Injectable({      providedIn: 'root'  })  export class MomoService {      private localServerURL = 'http://localhost:4200';      private partnerCode = "MOMO";      private accessKey = "F8BBA842ECF85";      private secretkey = "K951B6PE1waDMi640xX08PD3vg6EkVlz";      private redirectUrl = `${this.localServerURL}/checkout`;      private ipnUrl = this.redirectUrl;      private requestType = "captureWallet";      private extraData = "";      constructor(          private http: HttpClient      ) { }      private createFixCors() {          let \_headers = new HttpHeaders();          \_headers = new HttpHeaders().set('Content-Type', 'application/json');          \_headers = \_headers.set('Access-Control-Allow-Origin', '\*');          \_headers = \_headers.set('Access-Control-Allow-Credentials', 'true');          \_headers = \_headers.set('Access-Control-Allow-Methods', 'GET, POST, OPTIONS');          \_headers = \_headers.set('Access-Control-Allow-Headers', 'Origin, Content-Type, Accept, X-Custom-Header, Upgrade-Insecure-Requests');          const options = {              headers: \_headers          }          return options;      }      private createSignature(secretkey: string, rawSignature: string) {          const shaObj = new jsSHA("SHA-256", "TEXT", {              hmacKey: { value: secretkey, format: "TEXT" },          });          shaObj.update(rawSignature);          return shaObj.getHash("HEX");      }      public verifyIPNSignatureTest(data: any) {          const rawSignature = "accessKey=" + this.accessKey + "&amount=" + data.amount + "&extraData=" + data.extraData + "&message=" + data.message + "&orderId=" + data.orderId + "&orderInfo=" + data.orderInfo + "&orderType=" + data.orderType + "&partnerCode=" + data.partnerCode + "&payType=" + data.payType + "&requestId=" + data.requestId + "&responseTime=" + data.responseTime + "&resultCode=" + data.resultCode + "&transId=" + data.transId;          const signature = this.createSignature(this.secretkey, rawSignature);          return signature == data.signature ? true : false;      }      public verifyIPNSignature(data: any) {          const accessKey = "TGWTJShO0mPx6usU";          const secretkey = "INN28GZCXV6RpTgqz3TyDnea6fDAQF7A";          const rawSignature = "accessKey=" + accessKey + "&amount=" + data.amount + "&extraData=" + data.extraData + "&message=" + data.message + "&orderId=" + data.orderId + "&orderInfo=" + data.orderInfo + "&orderType=" + data.orderType + "&partnerCode=" + data.partnerCode + "&payType=" + data.payType + "&requestId=" + data.requestId + "&responseTime=" + data.responseTime + "&resultCode=" + data.resultCode + "&transId=" + data.transId;          const signature = this.createSignature(secretkey, rawSignature);          return signature == data.signature ? true : false;      }      public momoPaymentTest(totalAmount: any): Observable<any> {          const requestId = this.partnerCode + new Date().getTime();          const orderId = requestId;          const orderInfo = "Thanh toán với Momo";          const amount = totalAmount;          const rawSignature = "accessKey=" + this.accessKey + "&amount=" + amount + "&extraData=" + this.extraData + "&ipnUrl=" + this.ipnUrl + "&orderId=" + orderId + "&orderInfo=" + orderInfo + "&partnerCode=" + this.partnerCode + "&redirectUrl=" + this.redirectUrl + "&requestId=" + requestId + "&requestType=" + this.requestType;          const signature = this.createSignature(this.secretkey, rawSignature);          const requestBody = JSON.stringify({              partnerCode: this.partnerCode,              accessKey: this.accessKey,              requestId: requestId,              amount: amount,              orderId: orderId,              orderInfo: orderInfo,              redirectUrl: this.redirectUrl,              ipnUrl: this.ipnUrl,              extraData: this.extraData,              requestType: this.requestType,              signature: signature,              lang: 'vi'          });          return this.http.post<any>(`/api/test/momo`, requestBody, this.createFixCors());      }      public momoPayment(param: any): Observable<any> {          // mã doanh nghiệp thật sự          const partnerCode = "MOMOAALK20220518";          const accessKey = "TGWTJShO0mPx6usU";          const secretkey = "INN28GZCXV6RpTgqz3TyDnea6fDAQF7A";          // phần sinh ra theo order          const requestId = new Date().getTime();          const orderId: any = requestId;          const orderInfo = "Thanh toán với Momo";          // set 1 ngàn cho đỡ tốn tiền :v          const amount = 1000;          const rawSignature = "accessKey=" + accessKey + "&amount=" + amount + "&extraData=" + this.extraData + "&ipnUrl=" + this.ipnUrl + "&orderId=" + orderId + "&orderInfo=" + orderInfo + "&partnerCode=" + partnerCode + "&redirectUrl=" + this.redirectUrl + "&requestId=" + requestId + "&requestType=" + this.requestType;          const signature = this.createSignature(secretkey, rawSignature);          const requestBody = JSON.stringify({              partnerCode: partnerCode,              accessKey: accessKey,              requestId: requestId,              amount: amount,              orderId: orderId,              orderInfo: orderInfo,              redirectUrl: this.redirectUrl,              ipnUrl: this.ipnUrl,              extraData: this.extraData,              requestType: this.requestType,              signature: signature,              lang: 'vi'          });          return this.http.post<any>(`/api/momo`, requestBody, this.createFixCors());      }  } |

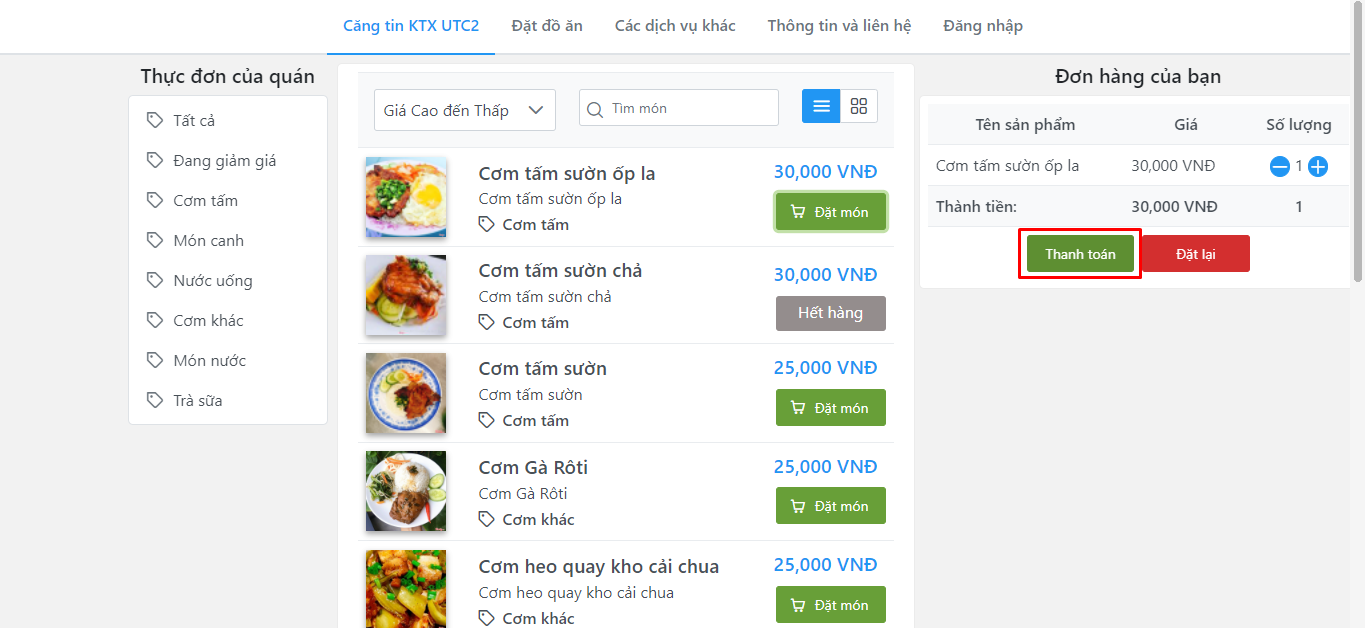
# CHƯƠNG 3: ỨNG DỤNG HỆ THỐNG VÀO MÔ HÌNH BÀI TOÁN THỰC TẾ

## 3.1 Quy trình tạo đơn hàng của người dùng

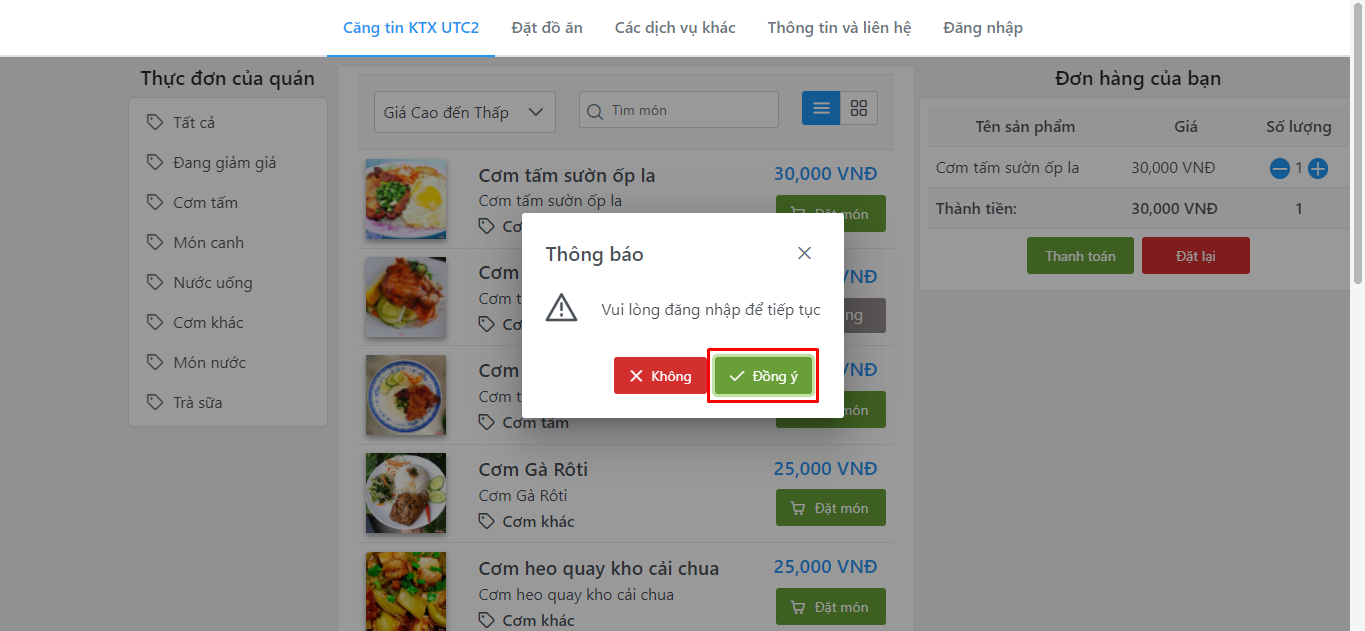
### 3.1.1 Người dùng chưa đăng nhập



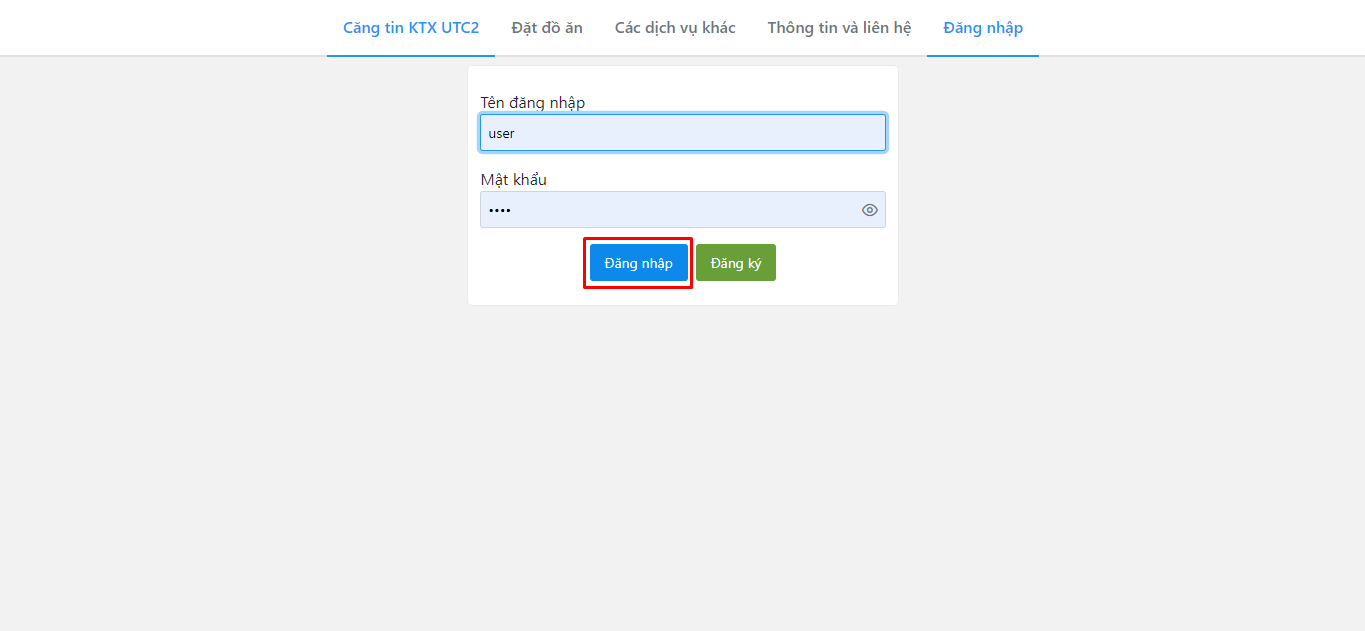
Hình 4. 1 Đặt món khi chưa đăng nhập



Hình 4. 2 Đến trang thanh toán khi chưa đăng nhập

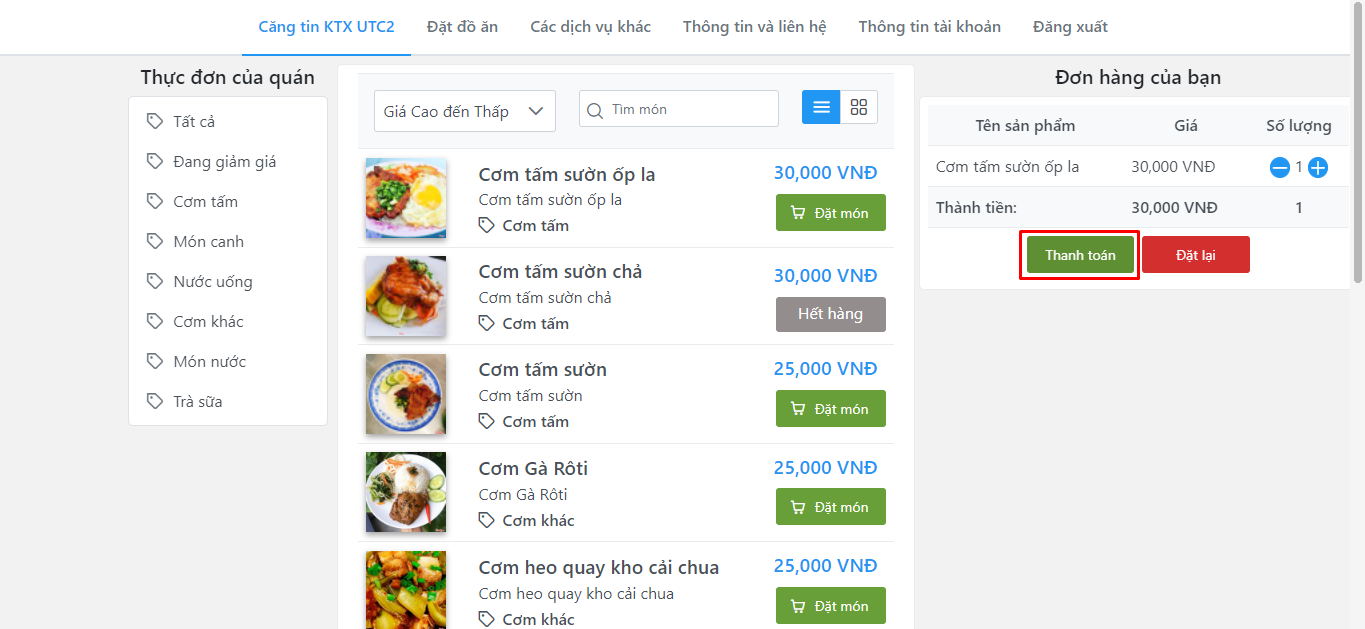


Hình 4. 3 Thông báo yếu câu đăng nhập

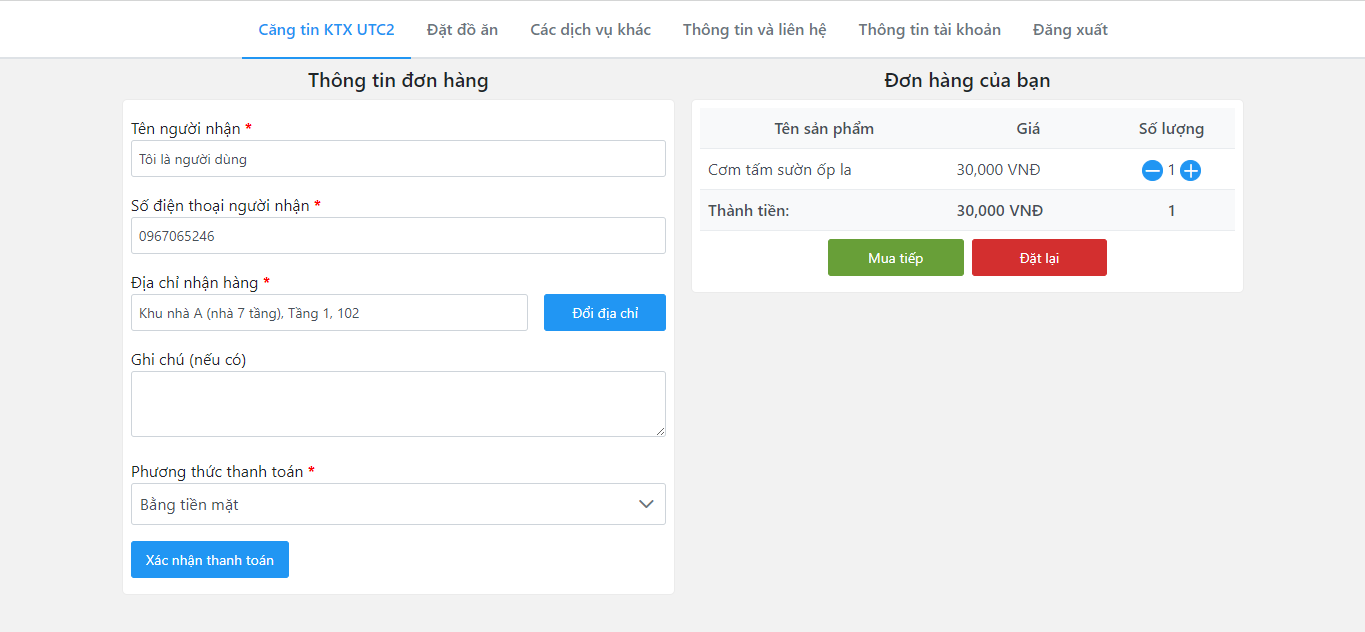


Hình 4. 4 Đăng nhập tài khoản vào hệ thống

### 3.1.2 Người dùng đã đăng nhập



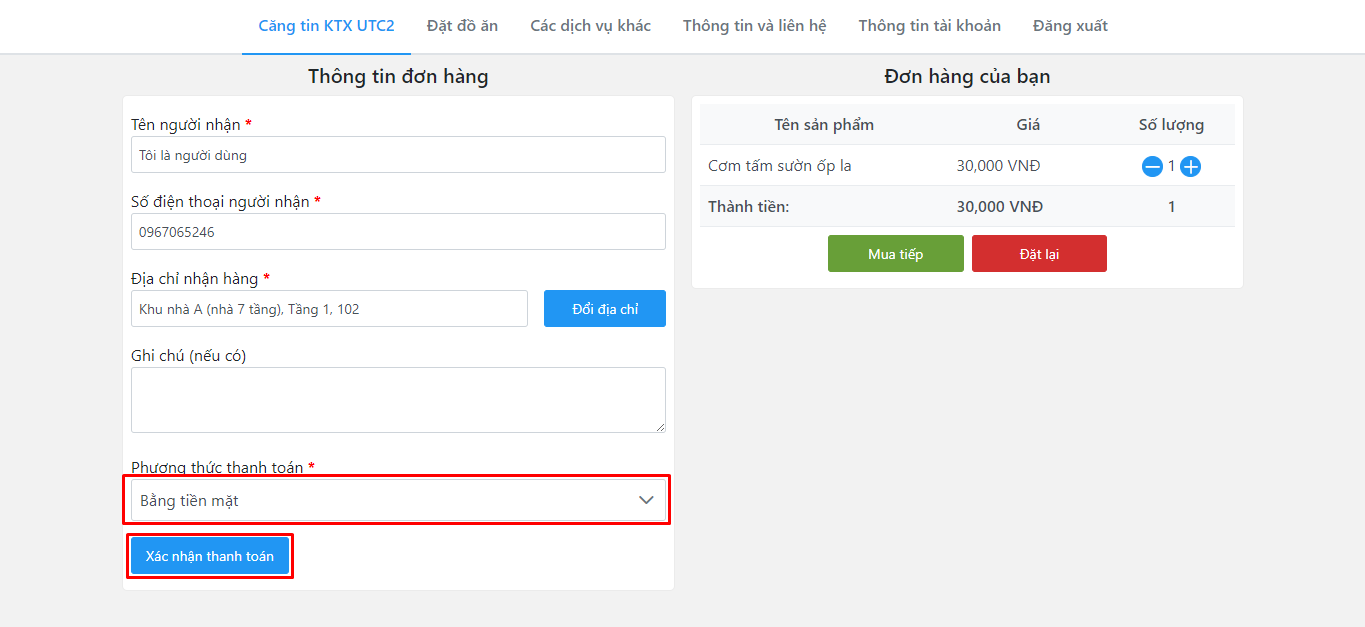
Hình 4. 5 Người dùng đã đăng nhập thành công và muốn thanh toán



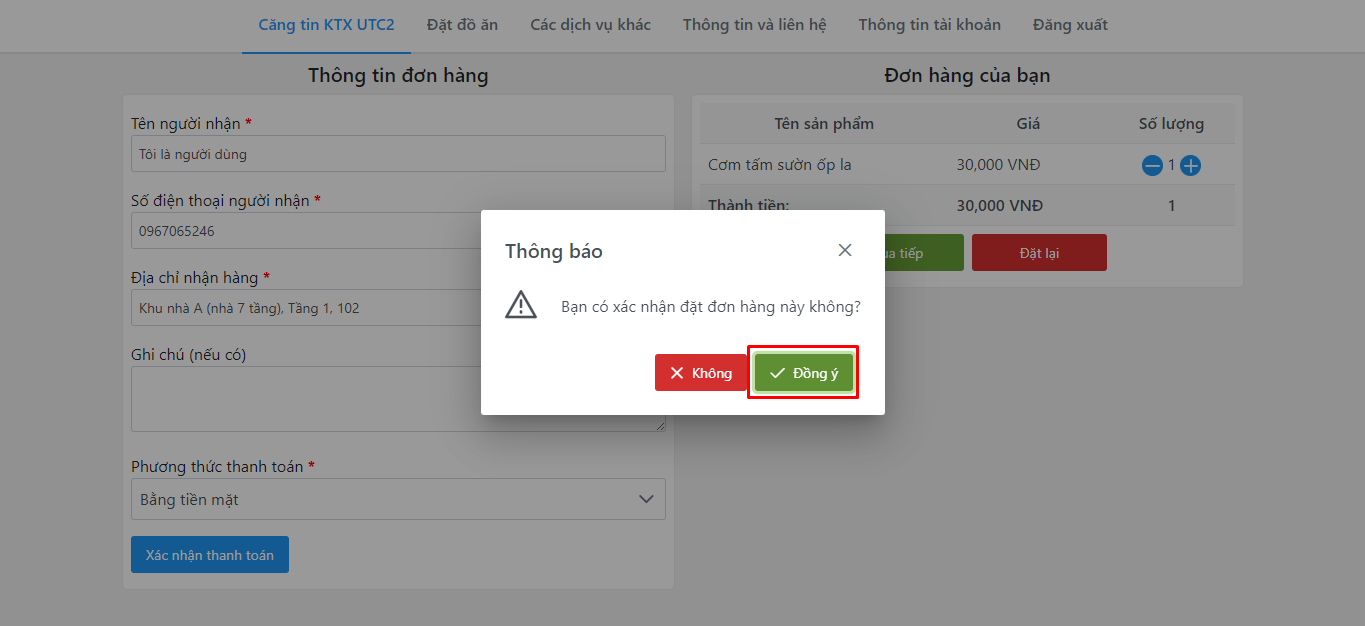
Hình 4. 6 Người dùng tiến hành thanh toán

## 3.2 Quy trình thanh toán của người dùng

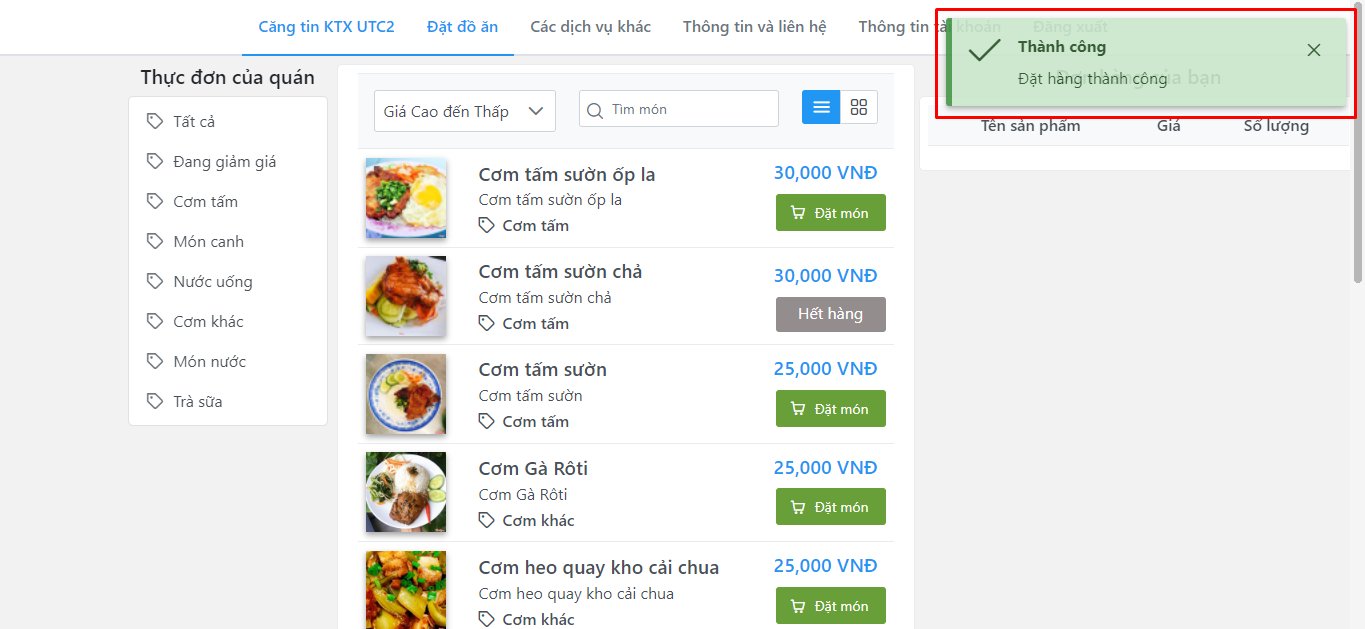
### 3.2.1 Thanh toán bằng tiền mặt



Hình 4. 7 Chọn phương thức thanh toán tiền mặt

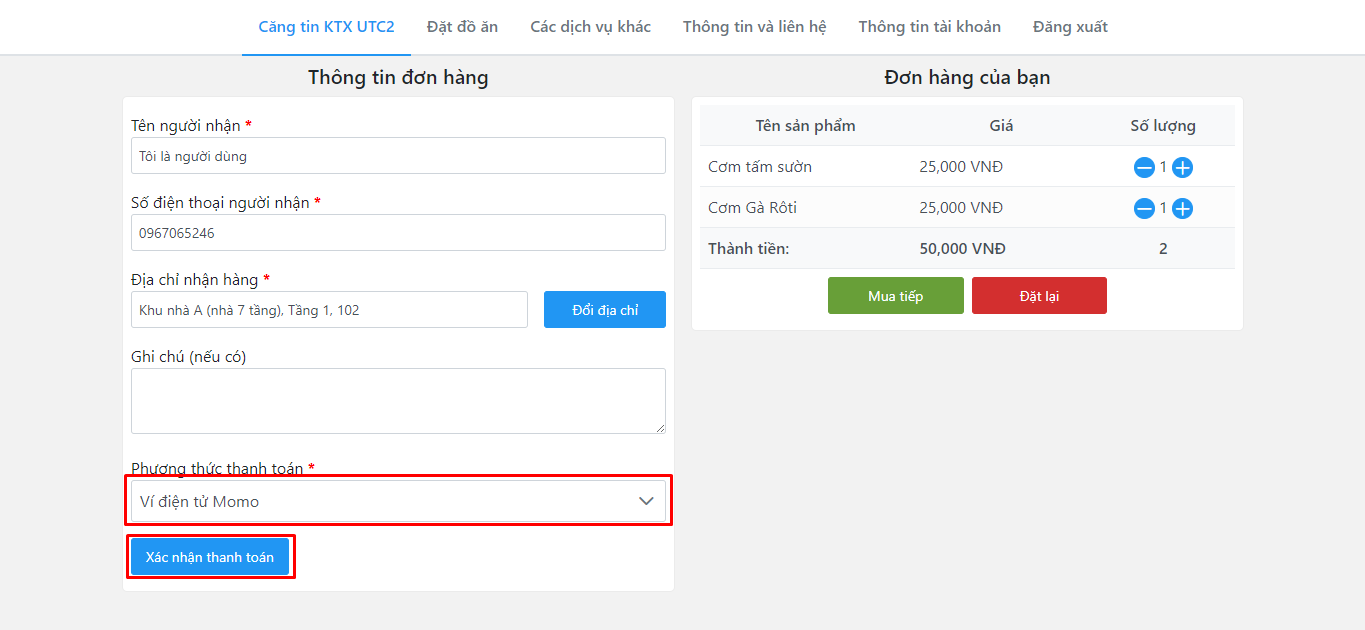


Hình 4. 8 Xác nhận thanh toán bằng tiền mặt

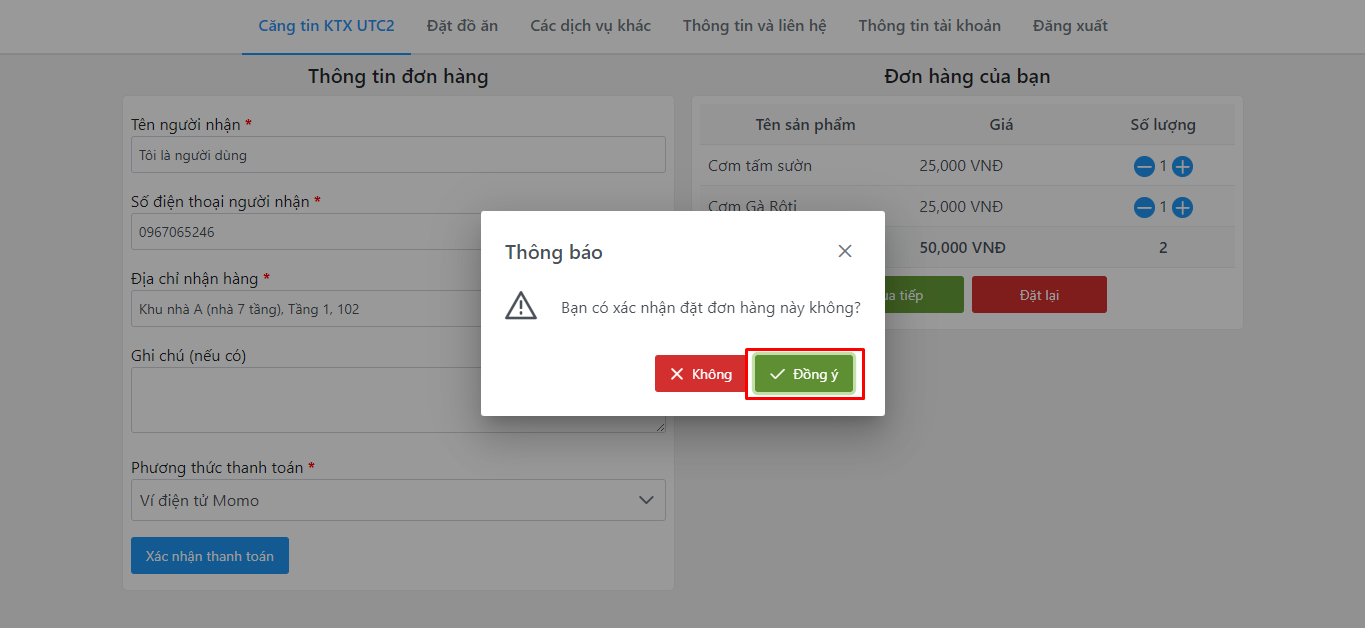


Hình 4. 9 Đặt đơn hàng thành công bằng tiền mặt

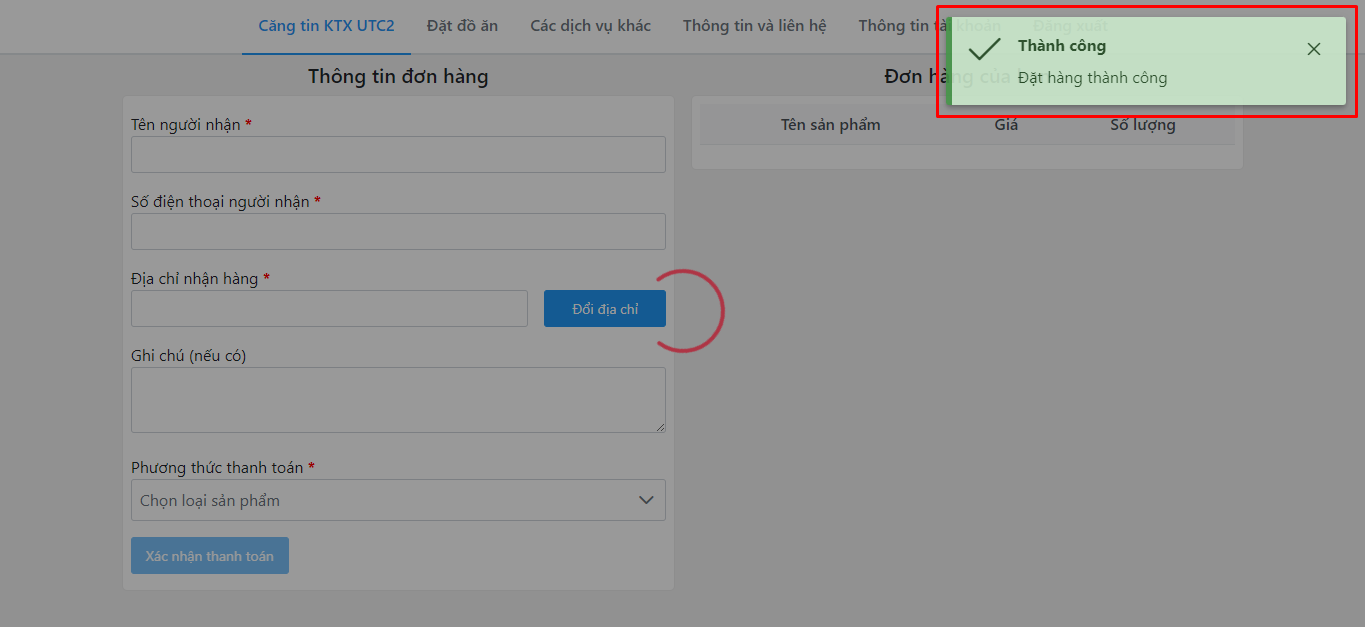
### 3.2.2 Thanh toán bằng ví điện tử Momo



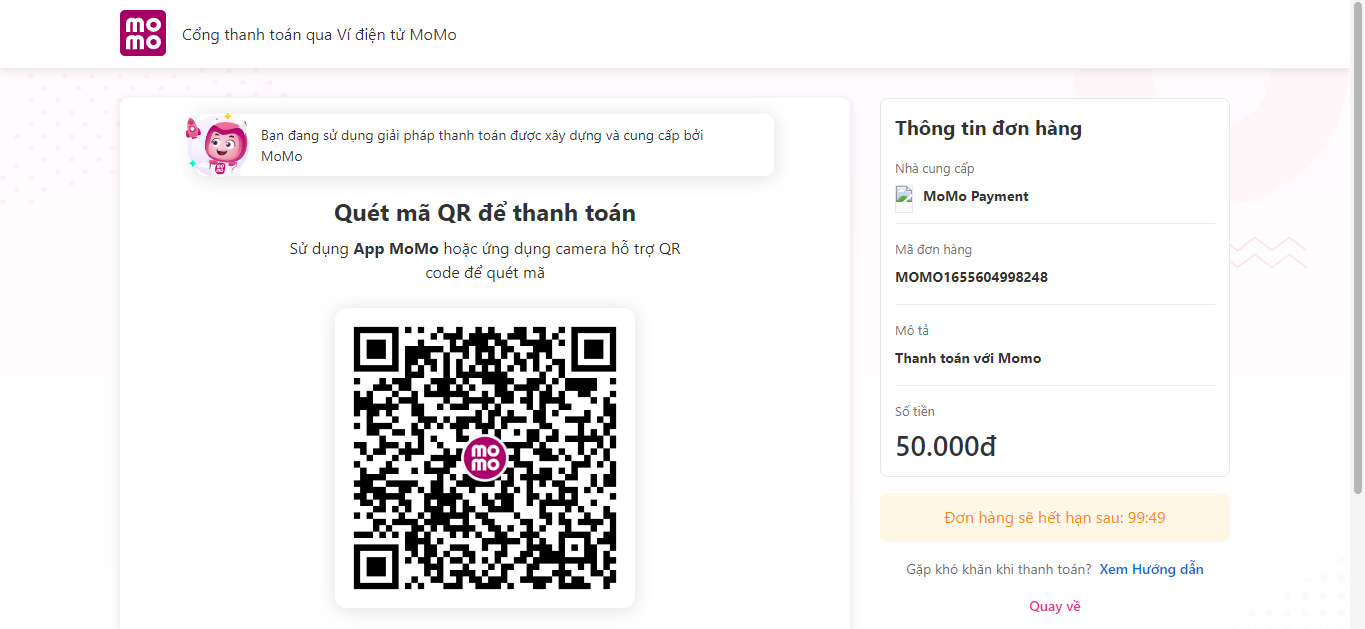
Hình 4. 10 Chọn phương thức thanh toán bằng ví điện tử Momo



Hình 4. 11 Xác nhận đặt hàng bằng phương thức thanh toán Momo

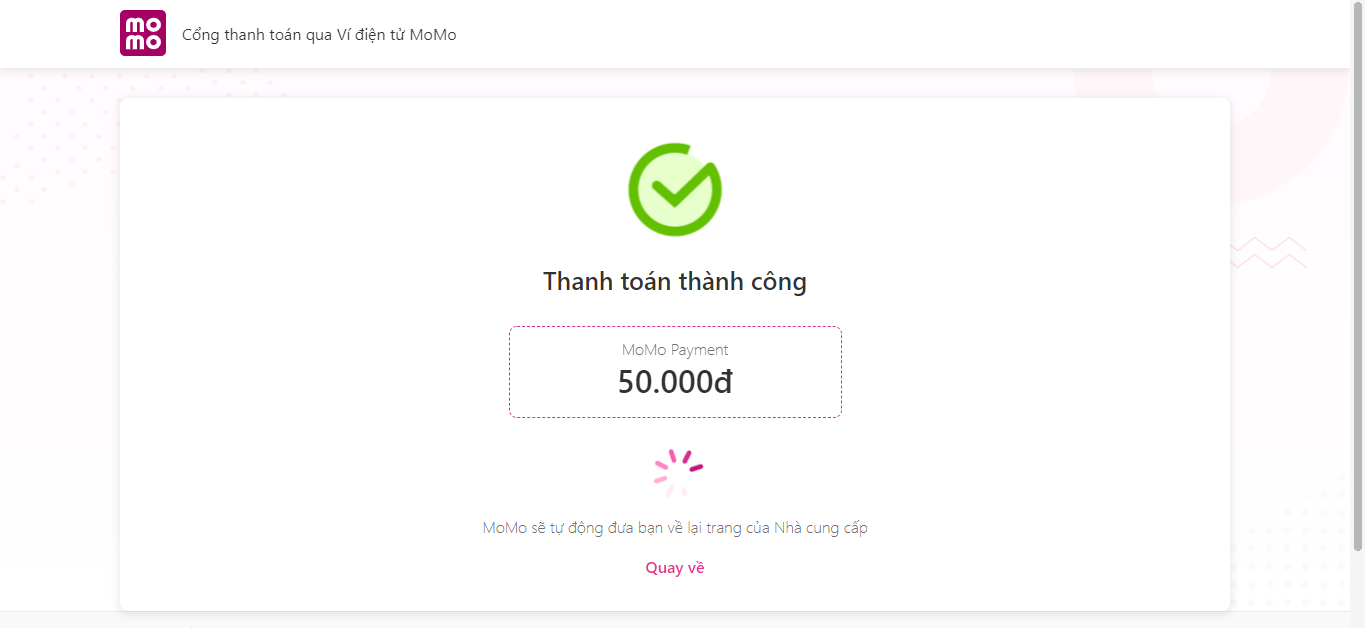


Hình 4. 12 Đặt hàng thành công bằng phương thức thanh toán Momo



Hình 4. 13 Chuyển đến trang thanh toán của Momo

#### 3.2.2.1 Thanh toán thành công

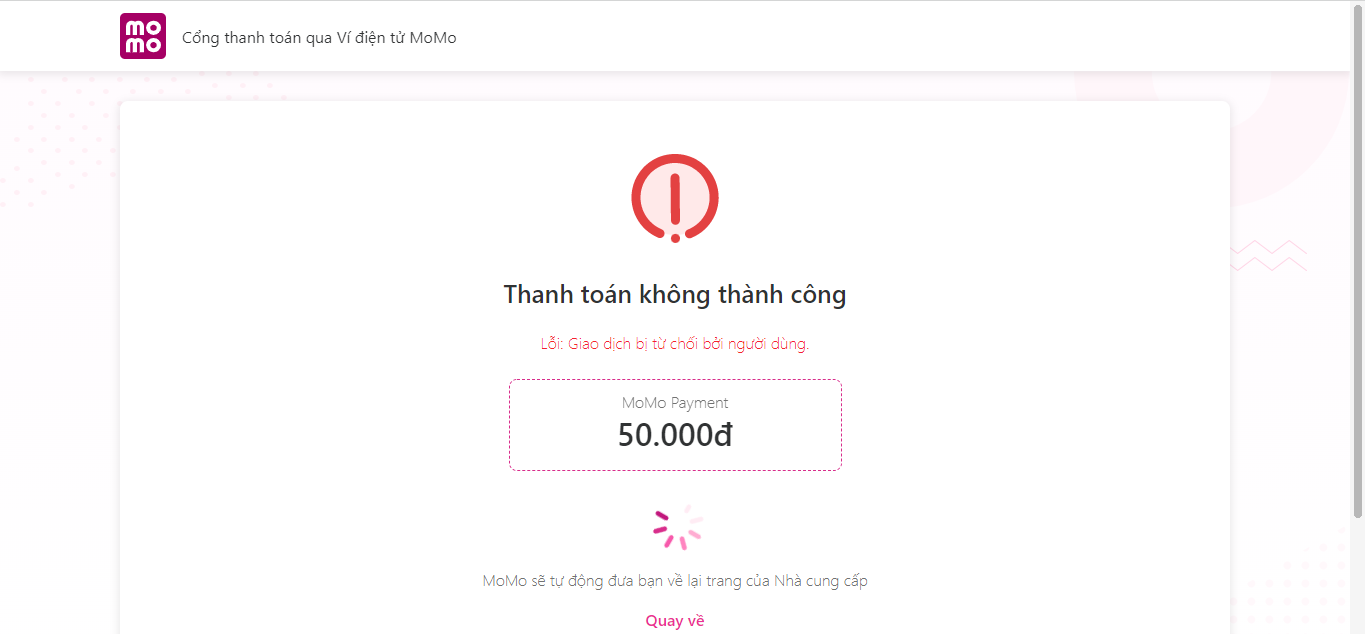


Hình 4. 14 Thanh toán Momo thành công

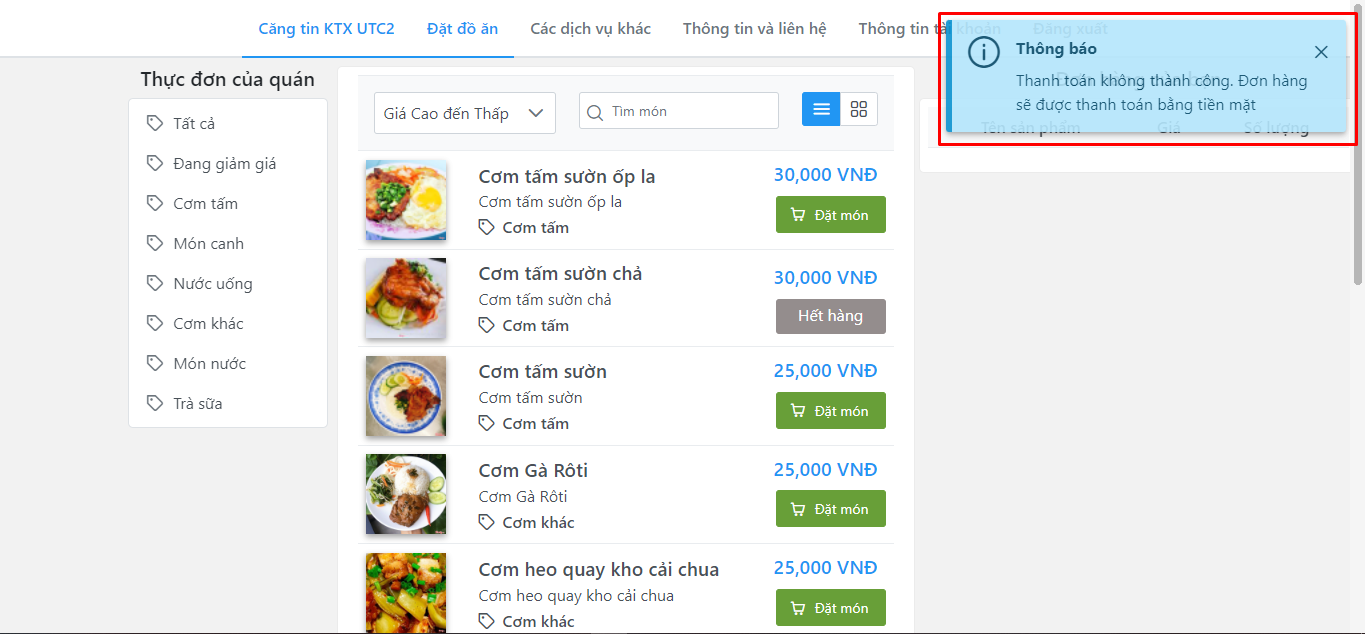


Hình 4. 15 Thông báo thanh toán Momo thành công

#### 3.2.2.2 Thanh toán thất bại



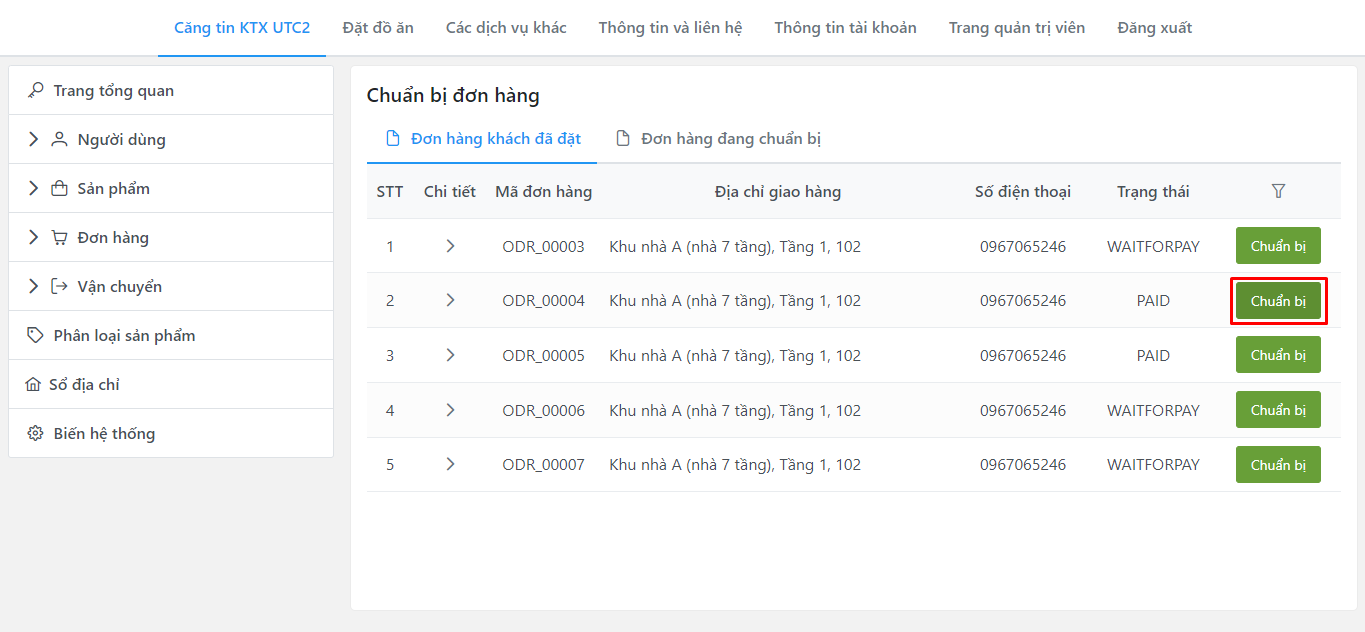
Hình 4. 16 Thanh toán Momo thất bại



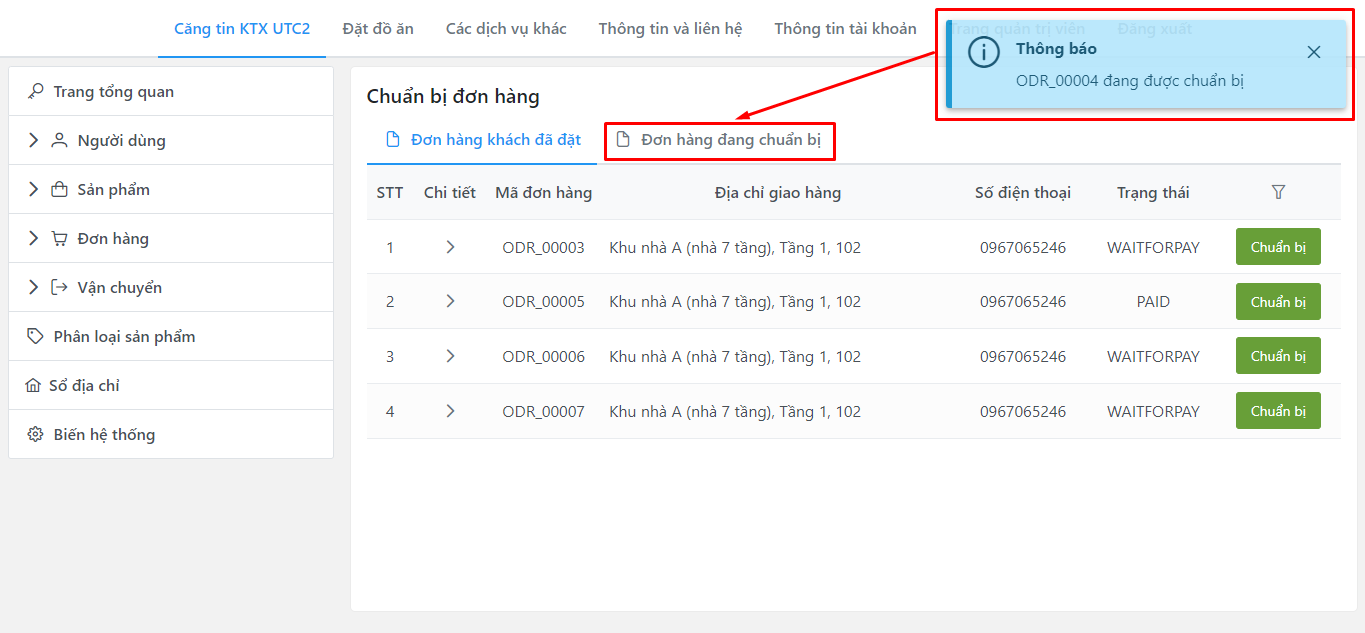
Hình 4. 17 Thông báo đơn hàng đã thánh toán Momo thất bại

## 3.3 Quy trình xử lý đơn hàng của chủ cửa hàng

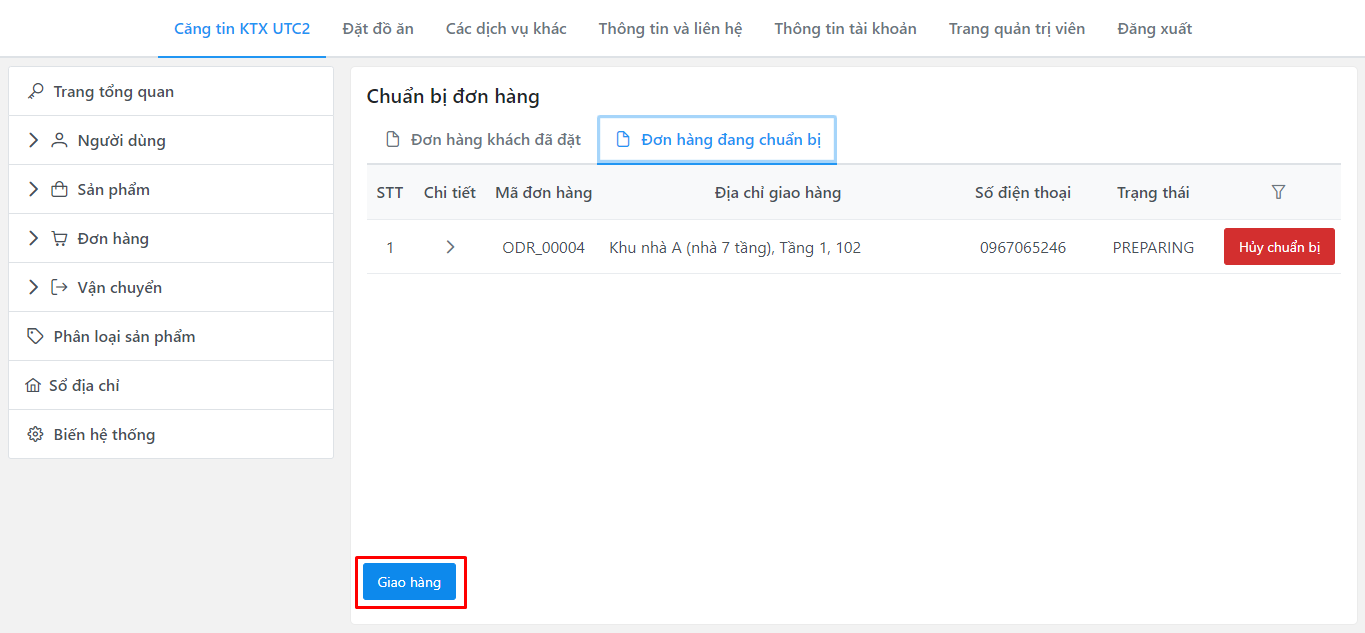
### 3.3.1 Không quản lý theo lô



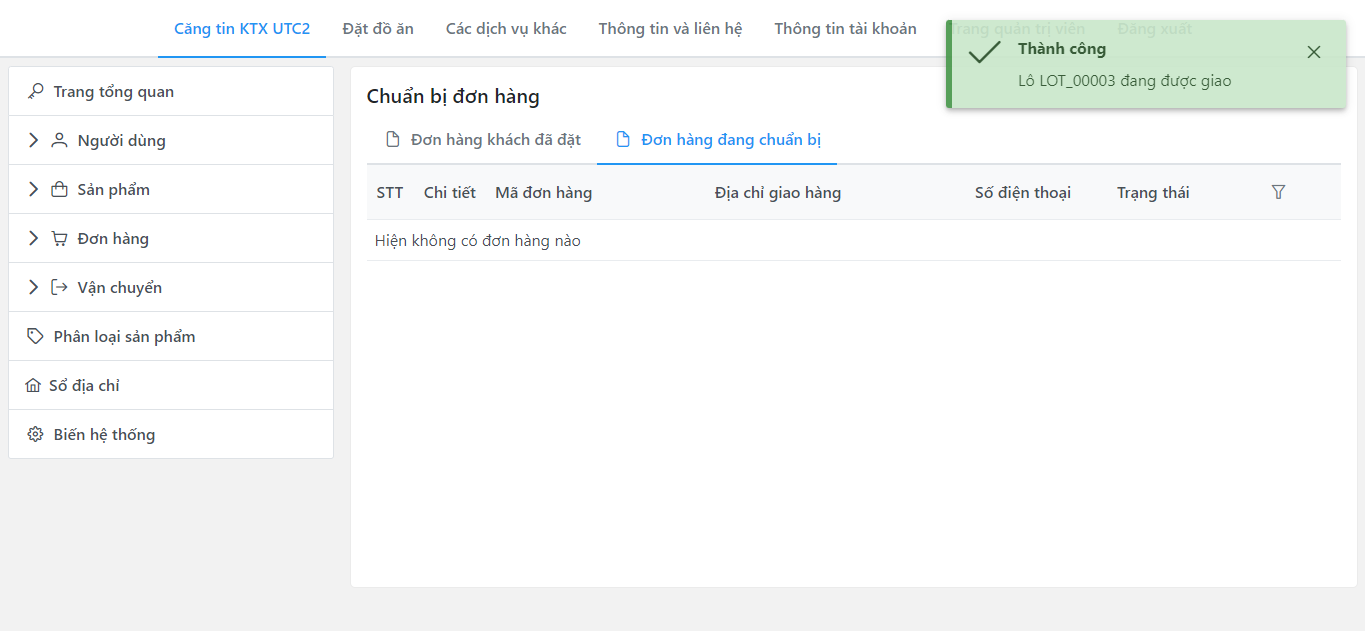
Hình 4. 18 Chuẩn bị đơn hàng



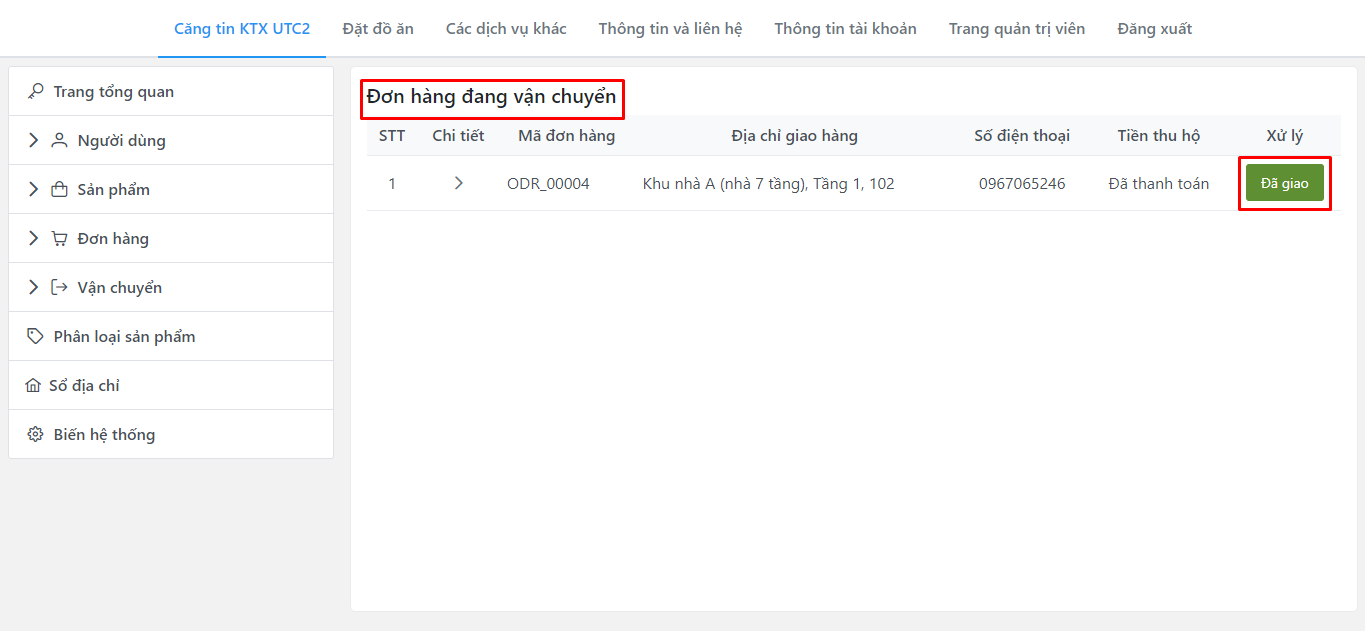
Hình 4. 19 Thông báo chuẩn bị cho đơn hàng thành công



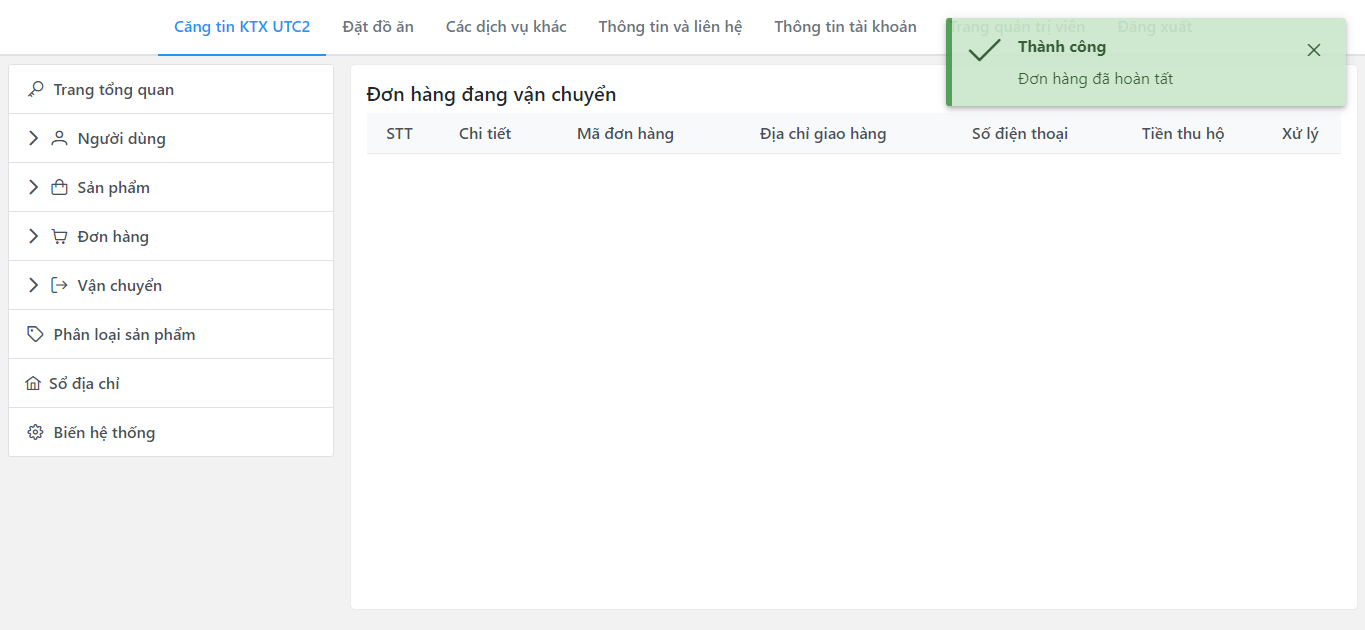
Hình 4. 20 Tiến hành giao hàng



Hình 4. 21 Thông báo tiến hành giao hàng thành công

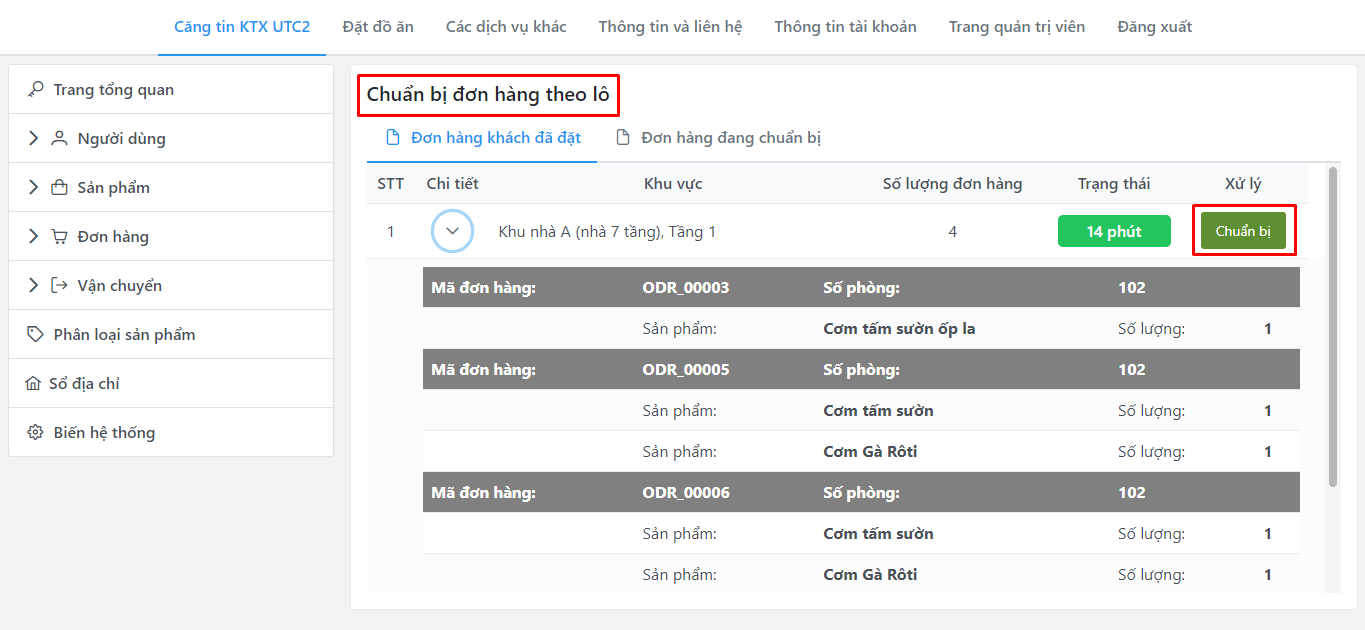


Hình 4. 22 Xác nhận đã giao hàng

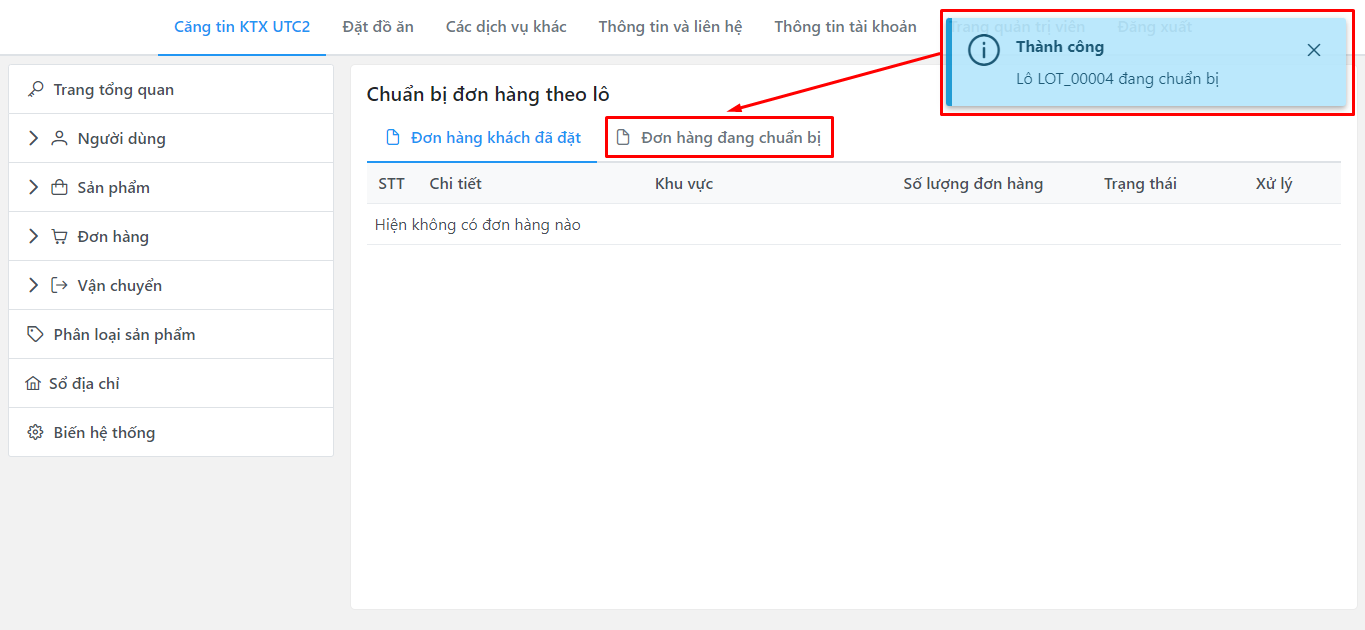


Hình 4. 23 Thông báo hoàn tất quá trình xử lý đơn hàng

### 3.3.2 Có quản lý theo lô



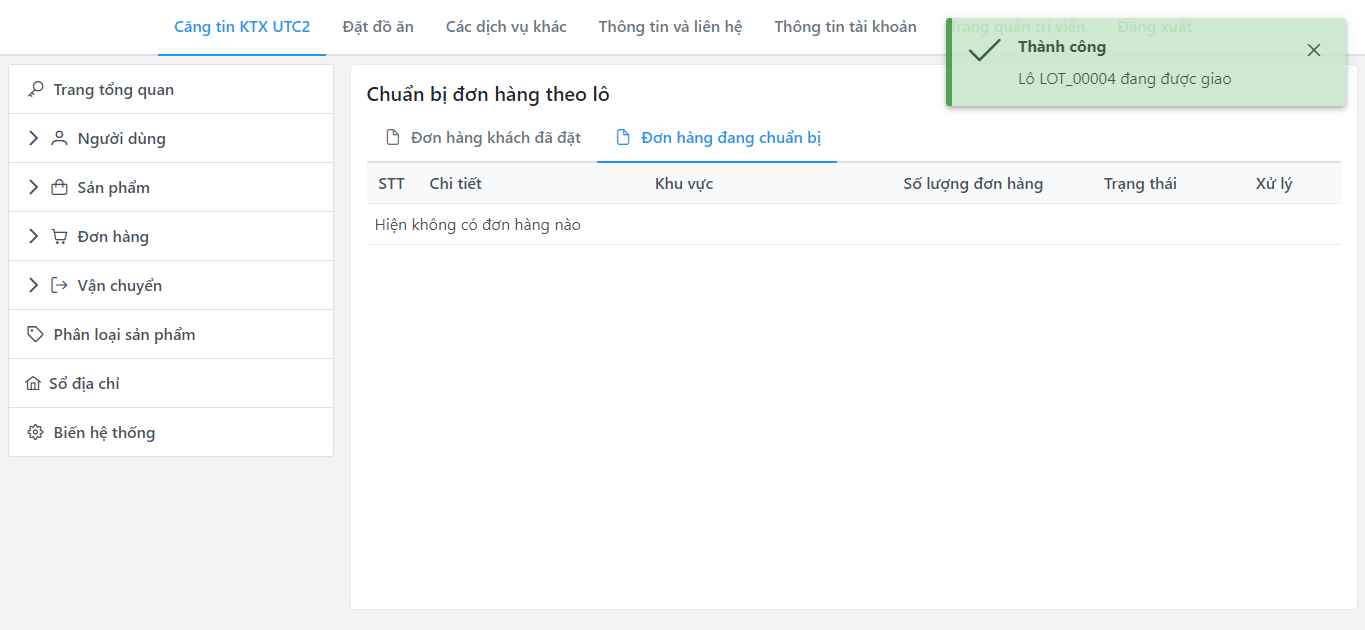
Hình 4. 24 Chuẩn bị đơn hàng theo lô



Hình 4. 25 Thông báo chuẩn bị đơn hàng theo lô thành công



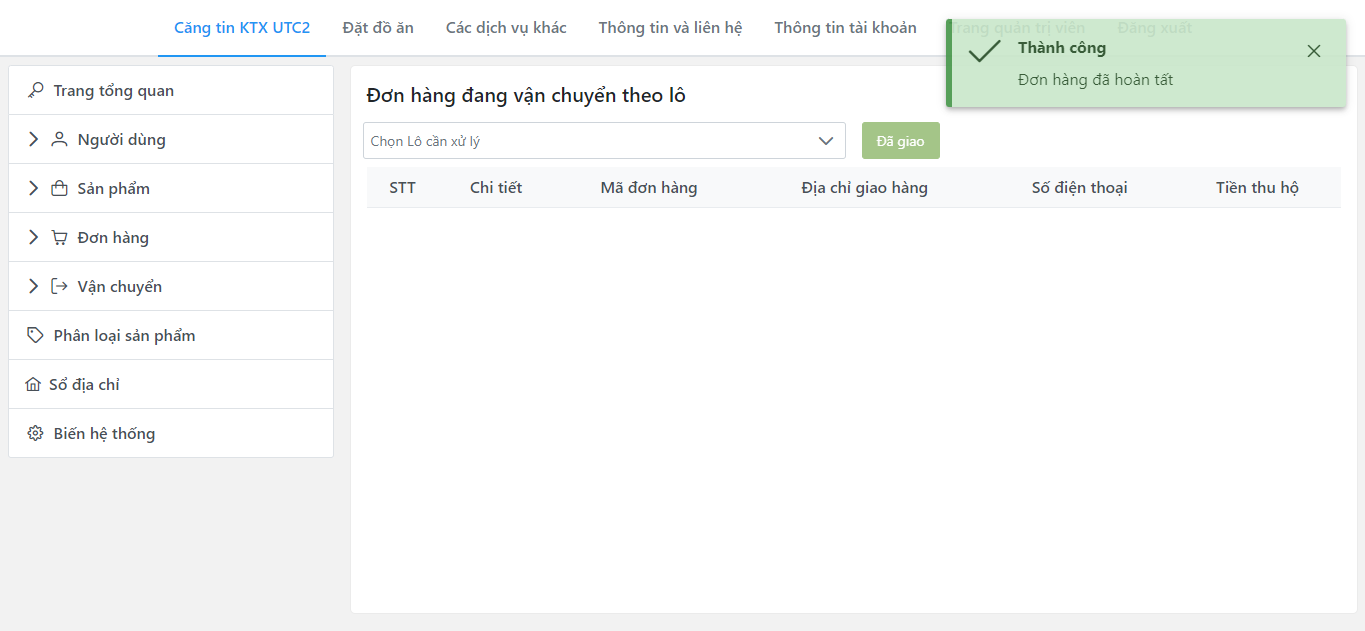
Hình 4. 26 Giao hàng theo lô



Hình 4. 27 Thông báo đã tiến thành giao hàng theo lô



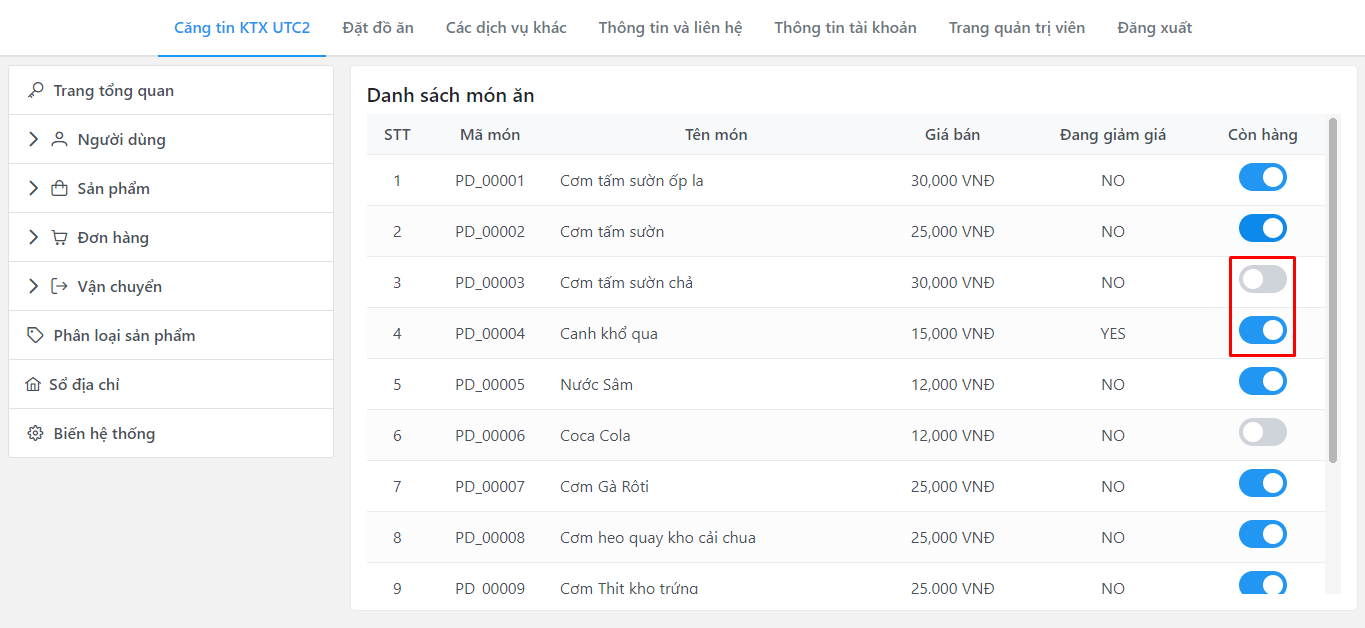
Hình 4. 28 Xác nhận giao hàng theo lô



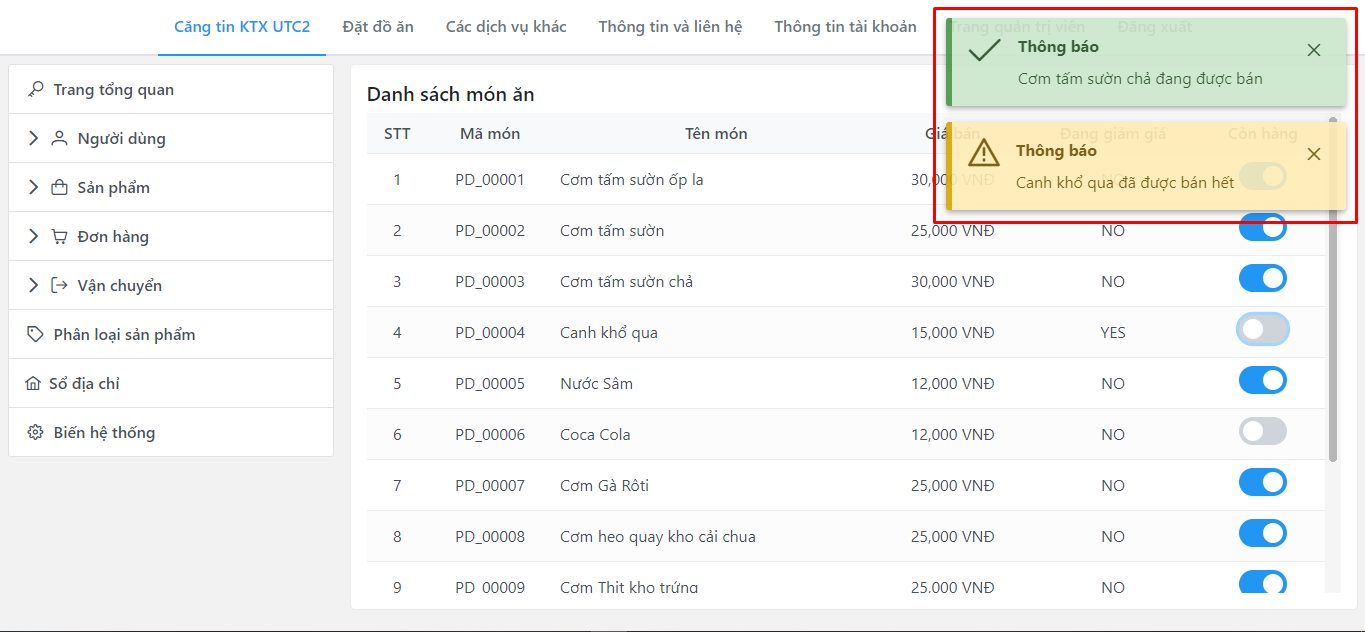
Hình 4. 29 Thông báo hoàn tất xử lý đơn hàng theo lô

## 3.4 Quy trình cập nhật trạng thái thông tin sản phẩm của chủ cửa hàng

### 3.4.1 Không sử dụng biến hệ thống

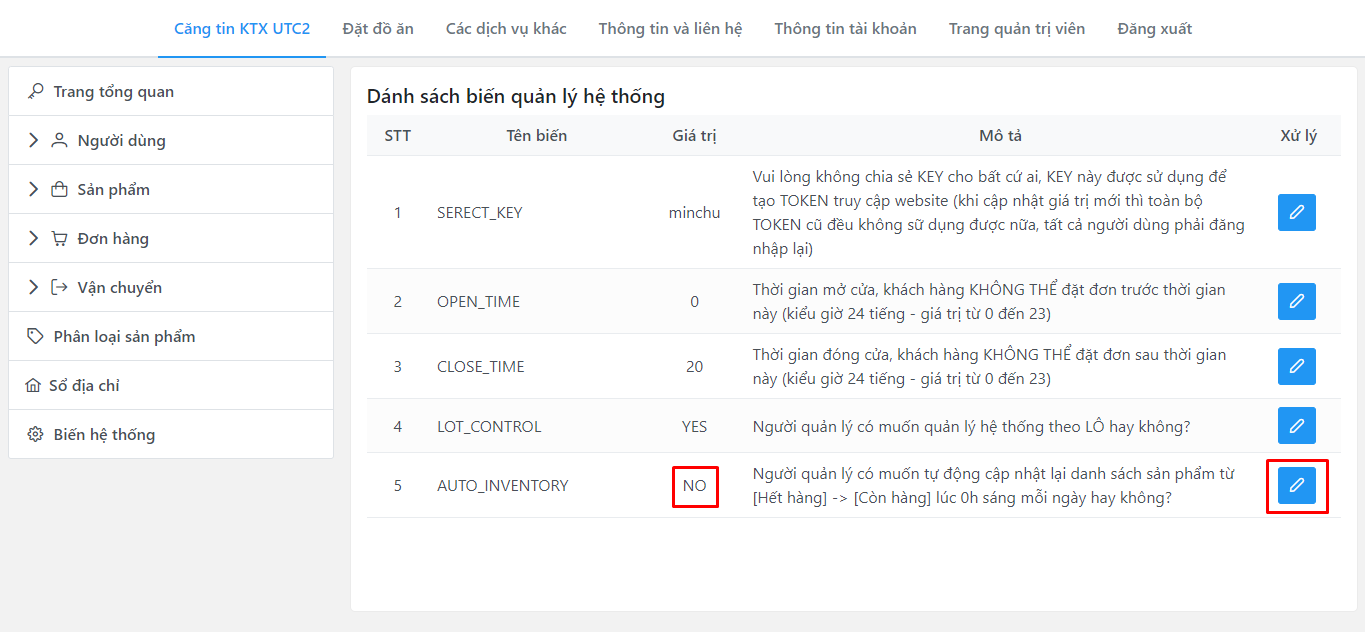


Hình 4. 30 Cập nhật trạng thái sản phẩm không dùng biến hệ thống

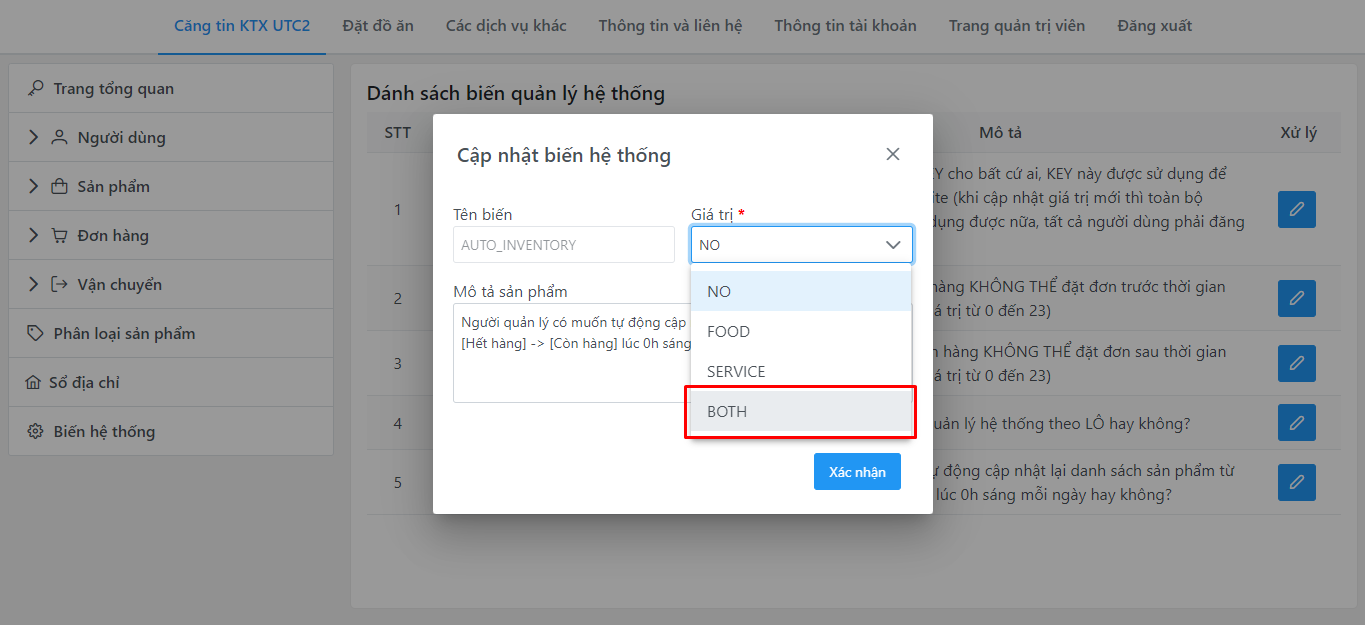


Hình 4. 31 Thông báo cập nhật trạng thái thành công

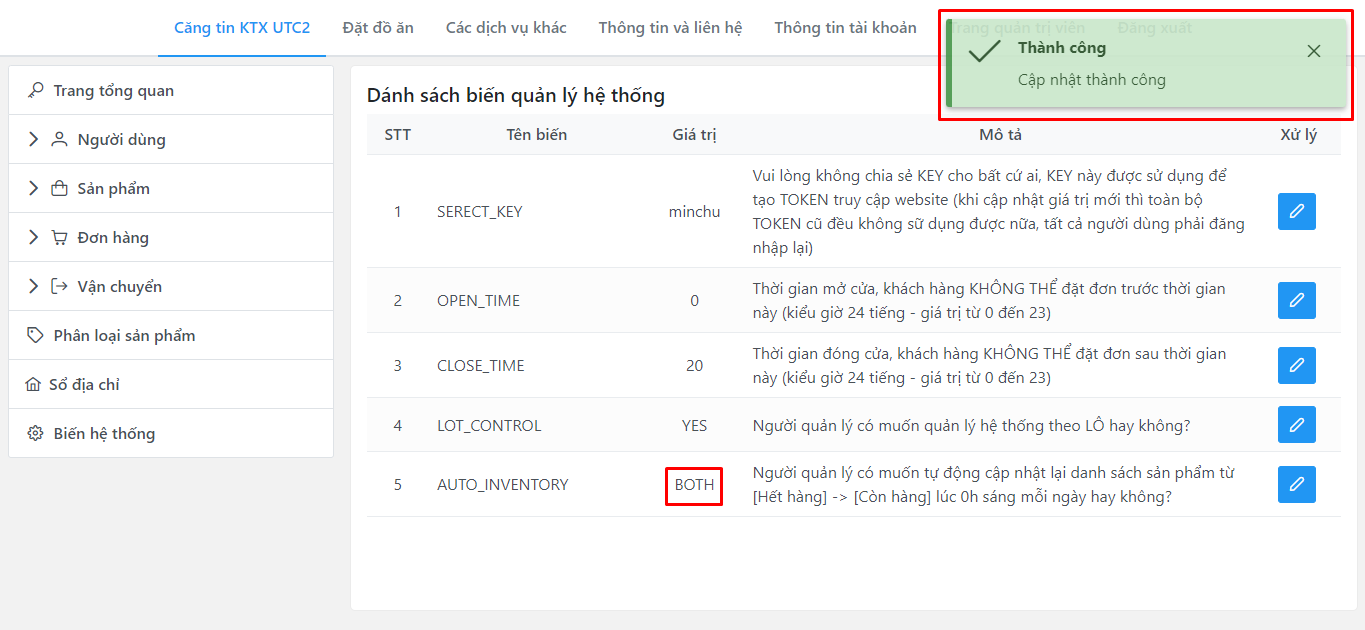
### 3.4.2 Có sử dụng biến hệ thống



Hình 4. 32 Cập nhật quản lý theo biến hệ thống



Hình 4. 33 Lựa chọn đối tượng cần quản lý



Hình 4. 34 Thông báo cập nhật thành công biến quản lý hệ thống

Theo như thông tin đã được nêu ở trên, đúng 0h sáng mỗi ngày thì hệ thống sẽ tự động cập nhật trạng thái cho danh sách đối tượng sản phẩm mà người chủ căng tin đã lựa chọn.

# KẾT LUẬN VÀ KIẾN NGHỊ

## Kết quả đạt được

Sau một thời gian nghiên cứu và thực hiện, tôi đã sử dụng ngôn ngữ lập trình Java (cụ thể là Spring Boot) và sự hỗ trợ của một vài thư viện để xử lý dữ liệu cũng như logic của hệ thông, hoàn thiện phần Back-End cho hệ thống đặt hàng trực tuyến.

Bên cạnh đó, là sử dụng MySQL để xây dựng cơ sở dữ liệu nhằm phục vụ cho việc lưu trữ cũng như quản lý dữ liệu ghi nhận được từ hệ thống.

Và để tạo giao diện thân thiện cho người sử dụng tôi đã sử dụng framework Angular cùng thư viên kèm theo là PrimeNG nhằm phục vụ cho mục đích này, hoàn thiện phần Font-End cho hệ thống đặt hàng trực tuyến. [9]

Kèm theo đó là tích hợp thành công phương thức thanh toán Momo cũng như tìm hiểu được cách sử dụng HMAC để xác thực tin nhắn mã hóa, giúp người sử dụng có nhiều lựa chọn và cảm thấy thoải mái hơn về mặt thanh toán đơn hàng.

Vậy nên, kết quả lớn nhất mà đề tài này đã đạt được là ta có thể áp dụng thực tiễn ngay tại khu căng tin trong ký túc xá UTC2 của nhà trường.

Giúp cho chủ căng tin và dễ dàng quản lý doanh nghiệp của mình hơn, bên cạnh đó là hạn chế tối đa sai sót trong lúc ghi nhận đơn hàng cũng như khó khăn trong việc tra cứu thông tin đối với các bạn sinh viên đang ở nội trú có nhu cầu đặt đồ ăn trực tuyến mà chưa thể thực hiện được.

Là ứng dụng gốc nên trong quá trình sử dụng không bị bên thứ 3 can thiệp, sửa đổi, đánh cắp thông tin, dữ liệu quan trọng (nhờ hệ thống đăng nhập khi muốn sử dụng các chức năng của ứng dụng).

Ngoài ra vì đây là ứng dụng tự phát triển riêng theo mô hình nên không phải mất phí dịch vụ duy trì doanh nghiệp như khi ta đăng ký mở các shop kinh doanh trực tuyến trên Tiki, Lazada, Shopee, … phần phụ phí này tương đối đáng kể, tuy nhiên khi sử dụng hệ thống này thì việc đó hoàn toàn không xảy ra.

Đề tài này có thể dễ dàng mở rộng tùy vào nhu cầu của mỗi cơ quan, công ty, doanh nghiệp hoặc cá nhân (vì phần lõi – core – của mô hình đã được hoàn thiện với đầy đủ chức năng cần thiết).

## Các vấn đề còn tồn đọng

Vì thực hiện trong khoảng thời gian khá ngắn nên phần Front-End chỉ hỗ trợ cho các thiết bị có kích thước màn hình lớn (kích thước trang web trên máy tính/máy tính bảng) nên chưa hỗ trợ hoàn toàn cho các thiết bị di động được.

Tính năng của hệ thống chưa được đa dạng như các nền tảng hiện có khác, nên cần được phát triển thêm.

## Đề xuất hướng phát triển

Như có nhắc đến ở phần nhược điểm ở trên rằng phần Front-End chưa hỗ trợ tốt cho các tiết bị di dộng, tuy nhiên phần Back-End được xây dựng dựa trên Restful API (hỗ trợ đa nền tảng và thiết bị dựa trên việc gọi API và nhận lại dữ liệu mong muốn) nên chỉ cần xây dựng giao diện hỗ trợ cho thiết bị di động là có thể sử dụng ngay được mà không cần phải xây dựng lại phần Back-End.

Xây dựng thêm một số tính năng khác như cho phép người dùng điểm danh mỗi ngày để tích lũy điểm thưởng khi mua hàng, …

Tích hợp thêm nhiều phương thức thanh toán trực tuyến khác nữa, ngoài ra tích hợp thêm internet banking để người dùng thoải mái hơn trong việc lựa chọn phương thức thanh toán mà phù hợp với mình nhất.

# TÀI LIỆU THAM KHẢO

[1] Herbert Schildt, Java: A Beginner’s Guide, Eighth Edition 8th Edition. (McGraw-Hill - 11/2018)

[2] Baron Schwartz (Author), Peter Zaitsev (Author), Vadim Tkachenko (Author), High Performance MySQL: Optimization, Backups, and Replication Third Edition. (O'Reilly Media, Inc - 3/2012)

[3] Craig Walls, Spring Boot in Action 1st Edition (Manning Publications - 12/2015)

[4] Giới thiệu về Java

<https://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)>

[5] Giới thiệu về API

<https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng>

[6]. Hướng dẫn tích hợp thanh toán Momo

Ngày truy cập (05/06/2022)

[https://developers.momo.vn/v2/#/](https://developers.momo.vn/v2/%23/)

<https://github.com/momo-wallet/payment/blob/master/nodejs/MoMo.js>

[7]. Tài liệu về Spring Boot

<https://start.spring.io/>

<https://docs.spring.io/spring-boot/docs/2.7.0/reference/htmlsingle/>

[8]. Tài liệu về Angular

<https://angular.io/docs>

[9]. PrimeNG framework cho Angular:

<https://www.primefaces.org/primeng/setup>

[10]. Hướng dẫn lập trình Spring Boot và Angular

<https://youtu.be/Gx4iBLKLVHk>

[11]. Tham khảo về lập trình Angular

Ngày truy cập (20/04/2022)

<https://viblo.asia/p/co-ban-ve-router-trong-angular-2-63vKjn7yK2R>

[12]. Hướng dẫn tạo mối quan hệ giữa các Model

Ngày truy cập (28/04/2022)

<https://codejava.net/frameworks/spring/jpa-join-query-for-like-search-examples>

[13] Đặc điểm của Java

Ngày truy cập (15/04/2022)

<https://anphanhv.wordpress.com/2015/03/21/tong-quan-ngon-ngu-java/>

[14]. Tham khảo giao diện và chức năng

<https://shopeefood.vn/>

[15]. Tham khảo giao diện trang quản trị viên

<https://adminlte.io/>

[16]. Tham khảo giao diện trang quản trị viên

<https://www.hostinger.vn/huong-dan/mysql-la-gi/>