

ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN



KHOÁ LUẬN
TỐT NGHIỆP ĐẠI HỌC

Đề tài:
TÌM HIỂU ASP.NET CORE MVC ĐỂ XÂY
DỰNG ỨNG DỤNG ĐI CHỢ ONLINE

Sinh viên thực hiện : LƯƠNG VIỆT BẢO
Khóa : TINK42B – Hệ chính quy

Huế, 05 – 2022

ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN



KHOÁ LUẬN
TỐT NGHIỆP ĐẠI HỌC

Đề tài:
TÌM HIỂU ASP.NET CORE MVC ĐỂ XÂY
DỰNG ỨNG DỤNG ĐI CHỢ ONLINE

Sinh viên thực hiện : LƯƠNG VIỆT BẢO
Khóa : TINK42B – Hệ chính quy
Giáo viên hướng dẫn : NGUYỄN QUANG HÙNG

Huế, 05 – 2022

LỜI CẢM ƠN

Em xin chân thành cảm ơn các thầy cô giáo trong trường Đại học Khoa học Huế nói chung, các thầy cô trong Bộ môn Công nghệ phần mềm nói riêng đã dạy dỗ cho em kiến thức về các môn đại cương cũng như các môn chuyên ngành, giúp em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ em trong suốt quá trình học tập.

Đặc biệt, em xin chân thành cảm ơn thầy Nguyễn Quang Hưng đã giành nhiều thời gian tận tình hướng dẫn, định hướng, chỉ bảo em trong suốt quá trình làm khoá luận.

Em cũng xin chân thành cảm ơn bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên em trong suốt quá trình học tập và hoàn thành khoá luận tốt nghiệp.

Tuy nhiên, vì vốn kiến thức chuyên ngành lẫn kinh nghiệm cá nhân còn hạn hẹp nên việc xảy ra nhiều thiếu sót trong nội dung báo cáo là không thể tránh khỏi. Em rất mong nhận được sự góp ý, chỉ bảo từ quý thầy cô để hoàn thiện bài báo cáo này tốt hơn.

Một lần nữa, em xin chân thành cảm ơn!

Huế, tháng 5 năm 2022

Sinh viên thực hiện

Lương Viết Bảo

MỤC LỤC

MỤC LỤC	ii
DANH MỤC HÌNH ẢNH	v
CHƯƠNG 1: TỔNG QUAN VỀ ASP.NET CORE MVC	1
1.1 Giới thiệu về .NET Core	1
1.1.1 Định nghĩa về .NET Core	1
1.1.2 Lịch sử của .NET Core	1
1.1.3 Thành phần của .NET Core	2
1.1.4 Đặc điểm .NET Core.....	3
1.1.5 Ưu điểm của .NET Core	4
1.1.6 .NET Framework và .NET Core	5
1.2. Giới thiệu về ASP.NET Core.....	6
1.2.1 Tổng quan về ASP.NET Core.....	6
1.2.2 Cấu trúc của ASP.NET Core	7
1.2.3 Cách ASP.NET Core làm việc.....	11
1.2.4 Đặc điểm của ASP.NET Core.....	12
1.2.5 Ưu điểm của ASP.NET Core	13
1.2.6 ASP.NET và ASP.NET Core.....	14
1.2.7 Lợi ích khi sử dụng ASP.NET Core	15
1.3 Giới thiệu về mô hình MVC	17
1.3.1 Mô hình MVC.....	17
1.3.2 Cấu trúc của MVC	18
1.3.3 Luồng xử lý trong mô hình MVC	19
1.3.4 Nhược điểm và ưu điểm của mô hình MVC.....	20
1.3.5 Ứng dụng của mô hình MVC	21

1.4 Tổng quan về dự án ASP.NET Core MVC.....	21
1.4.1 Mẫu kiến trúc MVC trong ASP.NET Core	22
1.4.2 Cấu trúc một dự án ASP.NET Core MVC	24
1.4.3 Kết nối với hệ quản trị cơ sở dữ liệu Sql Server 2019	25
CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG ĐI CHỢ ONLINE.....	29
2.1 Phân tích thực trạng	29
2.2 Yêu cầu bài toán.....	30
2.3 Mô hình hóa yêu cầu.....	31
2.3.1 Xác định Actor.....	31
2.3.2 Xác định Use case.....	32
2.3.3 Sơ đồ Use case	33
2.3.4 Mô tả Use case	43
2.4 Phân tích và thiết kế hệ thống	49
2.4.1 Thiết kế cơ sở dữ liệu.....	49
2.4.2 Sơ đồ hoạt động (Activity Diagram)	50
2.4.3 Sơ đồ tuần tự (Sequence Diagram).....	55
2.5 Thiết kế giao diện hệ thống.....	63
2.5.1 Giao diện dành cho khách hàng.....	63
2.5.2 Giao diện dành cho admin	68
2.6 Cài đặt phần mềm lập trình và công nghệ sử dụng.....	70
2.6.1 Cài đặt phần mềm lập trình.....	70
2.6.2 Công nghệ sử dụng	70
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN TƯƠNG LAI	71
1.1 Kết luận	71
1.2 Hướng phát triển tương lai.....	72

TÀI LIỆU THAM KHẢO.....	73
-------------------------	----

DANH MỤC HÌNH ẢNH

Hình 1.1 Phiên bản phát hành của .NET Core	2
Hình 1.2 Cấu trúc .NET Core.....	3
Hình 1.3 Sơ đồ .NET Framework, .NET Core và Xamarin.....	5
Hình 1.4 Sự ra đời ASP.NET Core	7
Hình 1.5 Cấu trúc của ASP.NET Core.....	7
Hình 1.6 Cơ chế hoạt động của ASP.NET Core	11
Hình 1.7 Luồng xử lý trong mô hình MVC	19
Hình 1.8 Sơ đồ hoạt động ứng dụng kiến trúc MVC trong ASP.NET Core	22
Hình 1.9 Cấu trúc một dự án ASP.NET Core MVC.....	24
Hình 1.10 Bước 1 Cài đặt các gói thư viện Entity Framework.....	26
Hình 1.11 Bước 2 Cấu hình kết nối với cơ sở dữ liệu	26
Hình 1.12 Bước 3 Hiện thị màn hình Package Manager Console	27
Hình 1.13 Bước 4 Thêm lệnh kết nối cơ sở dữ liệu.....	27
Hình 1.14 Bước 5 Kết nối thành công cơ sở dữ liệu.....	28
Hình 2.1 Sơ đồ Use case tổng quát	33
Hình 2.2 Use case quản lý mua hàng	34
Hình 2.3 Use case quản lý giỏ hàng.....	34
Hình 2.4 Use case quản lý thông tin tài khoản.....	35
Hình 2.5 Use case quản lý đơn hàng.....	35
Hình 2.6 Use case đăng ký tài khoản	35
Hình 2.7 Use case quản lý sản phẩm.....	36
Hình 2.8 Use case quản lý danh mục sản phẩm.....	36
Hình 2.9 Use case quản lý đơn hàng.....	37
Hình 2.10 Use case quản lý khách hàng	37
Hình 2.11 Use case quản lý nhà cung cấp.....	38
Hình 2.12 Use case quản lý tài khoản truy cập hệ thống.....	38
Hình 2.13 Use case quản lý quyền truy cập hệ thống	39

Hình 2.14 Use case quản lý tỉnh/thành	39
Hình 2.15 Use case quản lý quận/huyện	40
Hình 2.16 Use case quản lý phường/xã.....	40
Hình 2.17 Use case quản lý tin tức	41
Hình 2.18 Use case quản lý page	41
Hình 2.19 Use case quản lý shipper.....	42
Hình 2.20 Use case quản lý quảng cáo	42
Hình 2.21 Use case quản lý thống kê.....	43
Hình 2.22 Cơ sở dữ liệu hệ thống	49
Hình 2.23 Sơ đồ hoạt động đăng nhập và đăng ký	50
Hình 2.24 Sơ đồ hoạt động quản lý mua hàng	51
Hình 2.25 Sơ đồ hoạt động quản lý giỏ hàng.....	52
Hình 2.26 Sơ đồ hoạt động quản lý tài khoản.....	52
Hình 2.27 Sơ đồ hoạt động quản lý đơn hàng.....	53
Hình 2.28 Sơ đồ hoạt động quản lý hệ thống.....	53
Hình 2.29 Sơ đồ hoạt động quản lý thống kê.....	54
Hình 2.30 Sơ đồ tuần tự đăng nhập và đăng ký	55
Hình 2.31 Sơ đồ tuần tự quản lý mua hàng.....	56
Hình 2.32 Sơ đồ tuần tự quản lý giỏ hàng	57
Hình 2.33 Sơ đồ tuần tự quản lý tài khoản.....	58
Hình 2.34 Sơ đồ tuần tự quản lý đơn hàng	59
Hình 2.35 Sơ đồ tuần tự quản lý hệ thống chức năng chính.....	60
Hình 2.36 Sơ đồ tuần tự quản lý hệ thống chức năng phụ.....	61
Hình 2.37 Sơ đồ tuần tự quản lý thống kê	62
Hình 2.38 Giao diện đăng ký khách hàng	63
Hình 2.39 Giao diện đăng nhập khách hàng	63
Hình 2.40 Giao diện cập nhật thông tin tài khoản	64
Hình 2.41 Giao diện quản lý đơn hàng	64

Hình 2.42 Giao diện thay đổi mật khẩu	65
Hình 2.43 Giao diện quản lý mua hàng	65
Hình 2.44 Giao diện thông tin sản phẩm	66
Hình 2.45 Giao diện quản lý giỏ hàng	66
Hình 2.46 Giao diện đặt hàng	67
Hình 2.47 Giao diện đặt hàng thành công.....	67
Hình 2.48 Giao diện đăng nhập admin	68
Hình 2.49 Giao diện quản lý thống kê	68
Hình 2.50 Giao diện quản lý hệ thống chức năng chính.....	69
Hình 2.51 Giao diện quản lý hệ thống chức năng phụ.....	69

CHƯƠNG 1: TỔNG QUAN VỀ ASP.NET CORE MVC

1.1 Giới thiệu về .NET Core

1.1.1 Định nghĩa về .NET Core

.NET Core là một nền tảng phát triển đa mục đích, mã nguồn mở được duy trì bởi Microsoft và cộng đồng .NET trên GitHub. Đó là nền tảng chéo (hỗ trợ Windows, macOS và Linux) và có thể được sử dụng để xây dựng các ứng dụng thiết bị, đám mây và IoT.

1.1.2 Lịch sử của .NET Core

Ngày 12 tháng 11 năm 2014, Microsoft đã công bố .NET Core, trong nỗ lực hỗ trợ đa nền tảng cho .NET trên các hệ điều hành Linux và macOS. Khởi đầu .NET Core áp dụng mô hình phát triển mã nguồn mở thông thường dưới sự quản lý của .NET Foundation, được Miguel de Icaza, nhân viên Microsoft phụ trách phát triển phần mềm của dự án, mô tả .NET Core là "phiên bản .NET được thiết kế lại dựa trên phiên bản đơn giản hóa của các thư viện lớp".

Phiên bản .NET Core 1.0 phát hành vào ngày 27 tháng 6 năm 2016, cho phép phát triển theo bộ công cụ Microsoft Visual Studio 2015 Update 3. Tiếp theo các phiên bản .NET Core 1.0.4 và .NET Core 1.1.1 được phát hành theo .NET Core Tools 1.0 và Visual Studio 2017 vào 7 tháng 3 năm 2017.

Phiên bản .NET Core 2.0 phát hành vào ngày 14 tháng 8 năm 2017, kèm với Visual Studio 2017 15.3, ASP .NET Core 2.0, và Entity Framework Core 2.0. Phiên bản .NET Core 2.1 phát hành ngày 30 tháng 5 năm 2018 với bản 2.1.30 là phiên bản Hỗ trợ dài hạn LTS. Phiên bản .NET Core 2.2 phát hành ngày 4 tháng 12 năm 2018.

Lịch sử các phiên bản và kế hoạch phát triển .NET Core:

Phiên bản	Ngày phát hành	Phát hành với	Phiên bản cuối	Ngày cập nhật cuối cùng	Hỗ trợ đến ngày ^[7]
.NET Core 1.0	2016-06-27 ^[8]	Visual Studio 2015 Update 3	1.0.16	14 tháng 5 năm 2019	Ngày 27 tháng 6 năm 2019
.NET Core 1.1	2016-11-16 ^[9]	Visual Studio 2017 Version 15.0	1.1.13	14 tháng 5 năm 2019	Ngày 27 tháng 6 năm 2019
.NET Core 2.0	2017-08-14 ^[10]	Visual Studio 2017 Version 15.3	2.0.9	10 tháng 7 năm 2018	Ngày 1 tháng 10 năm 2018
.NET Core 2.1	2018-05-30 ^[11]	Visual Studio 2017 Version 15.7	2.1.30 (LTS)	19 tháng 8 năm 2021	Ngày 21 tháng 8 năm 2021
.NET Core 2.2	2018-12-04 ^[12]	Visual Studio 2019 Version 16.0	2.2.8	19 tháng 11 năm 2019	Ngày 23 tháng 12 năm 2019
.NET Core 3.0	2019-09-23 ^[13]	Visual Studio 2019 Version 16.3	3.0.3	18 tháng 2 năm 2020	Ngày 3 tháng 3 năm 2020
.NET Core 3.1	2019-12-03 ^[14]	Visual Studio 2019 Version 16.4	3.1.22 (LTS)	14 tháng 12 năm 2021	Ngày 3 tháng 12 năm 2022
.NET 5	2020-11-10 ^[15]	Visual Studio 2019 Version 16.8	5.0.13	14 tháng 12 năm 2021	Ngày 8 tháng 5 năm 2022
.NET 6	2021-11-08 ^[16]	Visual Studio 2022 Version 17.0	6.0.1 (LTS)	14 tháng 12 năm 2021	Ngày 8 tháng 11 năm 2024
.NET 7	2022-11 (dự kiến)				Tháng 5 năm 2024 (dự kiến)
.NET 8	2023-11 (dự kiến)		(dự kiến là LTS)		Tháng 11 năm 2026 (dự kiến)

Hình 1.1 Phiên bản phát hành của .NET Core

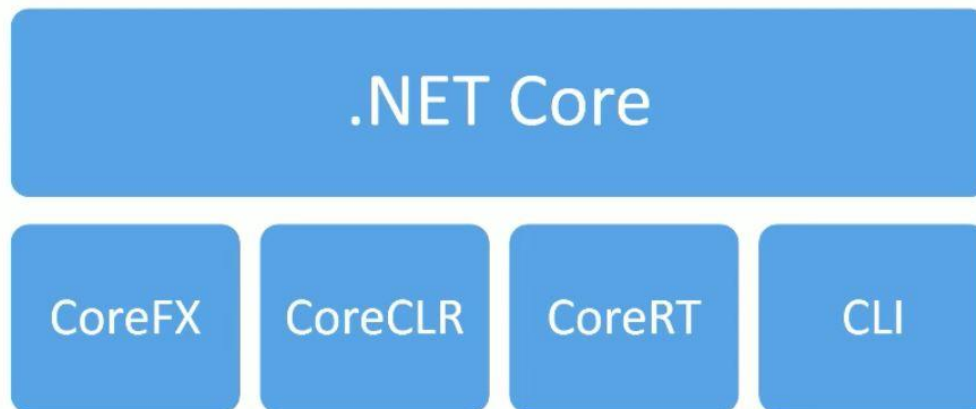
1.1.3 Thành phần của .NET Core

.NET Core bao gồm các nền tảng: .NET Compiler Roslyn, .NET Core framework CoreFX, .NET Core runtime CoreCLR, và ASP.NET Core.

Cấu trúc của .Net Core chi tiết:

- CoreFX: Nó được xem là nền tảng thư viện dành cho .NET Core.
- .NET Core runtime: cung cấp một hệ thống kiểu, tải lắp ráp, trình thu gom rác, interop gốc và các dịch vụ cơ bản khác. Các thư viện khung .NET Core cung cấp các kiểu dữ liệu nguyên thủy, các kiểu thành phần ứng dụng và các tiện ích cơ bản.
- ASP.NET Core runtime: cung cấp khung để xây dựng các ứng dụng kết nối internet, điện toán đám mây hiện đại, chẳng hạn như ứng dụng web, ứng dụng IoT và phụ trợ di động.
- .NET Core SDK và trình biên dịch ngôn ngữ (Roslyn và F #) cho phép trải nghiệm nhà phát triển .NET Core.

- Dotnet command, được sử dụng để khởi chạy các ứng dụng .NET Core và các lệnh CLI. Nó chọn thời gian chạy và lưu trữ thời gian chạy, cung cấp chính sách tải lắp ráp và khởi chạy các ứng dụng và công cụ.



Hình 1.2 Cấu trúc .NET Core

1.1.4 Đặc điểm .NET Core

- **Open-source Framework:** .NET Core là nền tảng nguồn mở được duy trì bởi Microsoft và có sẵn trên GitHub theo giấy phép MIT và Apache 2. Nó là một dự án .NET Foundation.
- **Cross-platform:** .NET Core chạy trên hệ điều hành Windows, macOS và Linux. Có thời gian chạy khác nhau cho mỗi hệ điều hành thực thi mã và tạo ra cùng một output.
- **Nhất quán giữa các kiến trúc:** Có thể thực thi mã giống nhau trong các kiến trúc tập lệnh khác nhau (bao gồm x64, x68 và RAM)
- **Hỗ trợ nhiều ngôn ngữ:** Có thể sử dụng ngôn ngữ lập trình C#, F# và Visual Basic để phát triển ứng dụng .NET Core.
- **Triển khai linh hoạt:** Các ứng dụng .NET Core có thể được triển khai song song (cài đặt toàn bộ người dùng hoặc toàn hệ thống). Nó cũng có thể triển khai với các Container Docker.

- **Khả năng tương thích:** .NET Core tương thích với .NET Framework và Mono API (thông qua .NET Standard).
- **Hỗ trợ bởi Microsoft:** Vì .NET Core được phát triển bởi Microsoft nên tài liệu cũng như các cập nhật, hỗ trợ được update thường xuyên. Từ đó, giúp người sử dụng có thể giải quyết vấn đề nhanh chóng.
- **Công cụ CLI:** .NET core bao gồm các công cụ CLI (Command – line – interface: Giao tiếp thông qua dòng lệnh) để phát triển và tích hợp liên tục.
- **Kiến trúc mô-đun:** .NET core hỗ trợ cách tiếp cận kiến trúc mô-đun bằng cách sử dụng các gói NuGet. Có các gói NuGet khác nhau cho các tính năng khác nhau có thể được thêm vào dự án .NET core nếu cần. Nhờ đó sẽ ít tiêu tốn dung lượng bộ nhớ, tăng hiệu suất và dễ bảo trì ứng dụng hơn.

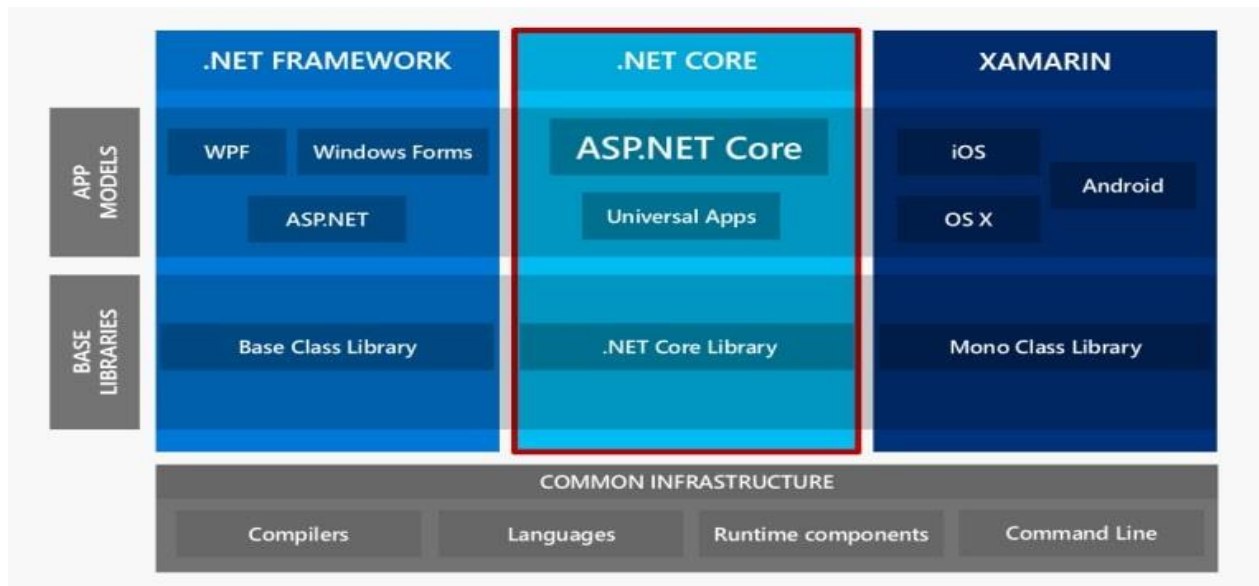
1.1.5 Ưu điểm của .NET Core

- .NET core có thể được sử dụng cho nhiều nền tảng khác nhau như Windows, Linux hay Mac OS. Khi ứng dụng nó với Visual Studio Code hoặc các công cụ khác sẽ giúp bạn phát triển ứng dụng trên các nền tảng khác nhau một cách dễ dàng.
- Với khả năng triển khai linh hoạt của mình, khi .NET core kết hợp với Container Docker hoặc Azure Kubernetes Service sẽ giúp tiết kiệm tài nguyên đáng kể.
- Với kiến trúc Mô đun, .NET Core ít tiêu tốn dung lượng bộ nhớ, tăng hiệu suất và dễ bảo trì ứng dụng hơn.
- Tốc độ thực thi nhanh và khả năng mở rộng ứng dụng cũng là một trong những ưu điểm lớn của .NET core

1.1.6 .NET Framework và .NET Core

NET Framework được giới thiệu năm 2000 và trở thành nền tảng chủ yếu trong việc phát triển sản phẩm .NET của Microsoft. Hai nền tảng .NET Core và .NET Framework có một số điểm khác biệt:

- .NET Core không hỗ trợ tất cả các mô hình .NET Framework. Cụ thể, .NET Core không hỗ trợ ASP.NET Web Forms và ASP.NET MVC nhưng nó hỗ trợ ASP.NET Core MVC. Bắt đầu từ .NET Core 3.0 cũng hỗ trợ Window Forms và WPF chỉ trên nền tảng Windows.
- .NET Core chứa một tập lớn các tập con của Thư viện lớp cơ sở .NET Framework (.NET Framework Base Class Library) nhưng có những yếu tố khác biệt như tên các assembly, kiểu,...
- .NET Framework chỉ hỗ trợ Windows và Windows Server trong khi đó .NET Core hỗ trợ thêm macOS và Linux.
- .NET Core là open source – điểm khác biệt cốt lõi với .NET Framework.



Hình 1.3 Sơ đồ .NET Framework, .NET Core và Xamarin

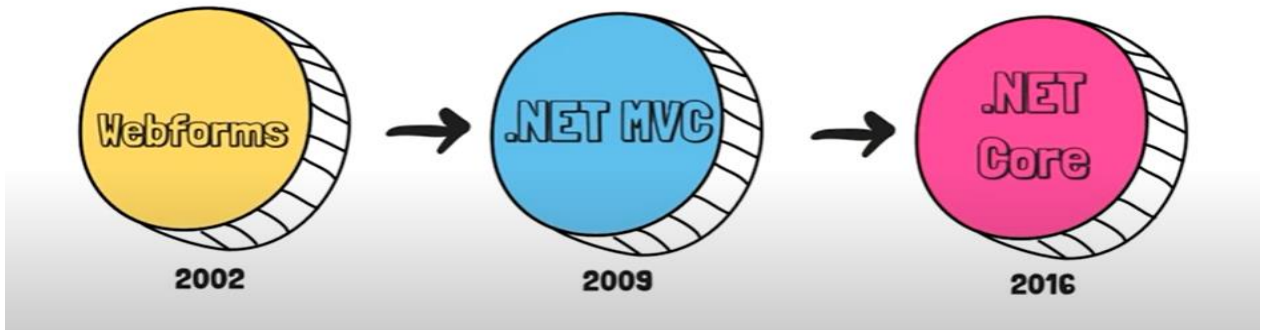
1.2. Giới thiệu về ASP.NET Core

1.2.1 Tổng quan về ASP.NET Core

ASP.NET Core là công nghệ phát triển ứng dụng web dựa trên hai nền tảng .NET Framework và .NET Core của Microsoft.

ASP.NET Core là phiên bản tiến hóa mới nhất của công nghệ ASP.NET, kế thừa những tính năng nổi trội từ ASP.NET cũ nhưng ASP.NET Core là nền tảng hoàn toàn mới, khác biệt so với ASP.NET trước đây.

ASP.NET xuất hiện lần đầu vào năm 2002 như là một phần của .NET Framework 1.0. Ứng dụng ASP.NET lúc này là ASP.NET Web Forms cho phép tạo các ứng dụng dựa trên giao diện đồ họa và mô hình hướng sự kiện. Theo thời gian, với việc các ứng dụng ASP.NET Web Forms trở nên cồng kềnh, khó bảo trì nên đến năm 2009, Microsoft cho ra đời phiên bản ASP.NET MVC nhằm tạo các ứng dụng web dựa trên mô hình MVC. Dù có nhiều ưu điểm nổi bật hơn ASP.NET Web Forms nhưng cả hai cùng dựa vào một nền tảng với tập tin trọng tâm *System.Web.dll*. Sự phụ thuộc này kế thừa một số ưu điểm của .NET Framework như tính tin cậy cao và nhiều tính năng thuận tiện cho sự phát triển các ứng dụng web hiện đại trên nền tảng Windows, đồng thời cũng mang lại một số hạn chế. Sự hạn chế đầu tiên là sự thay đổi chậm chạp về nội dung trong tập tin *System.Web.dll* ảnh hưởng lớn đến khả năng mở rộng của ASP.NET, bên cạnh đó, các ứng dụng web hiện đại đòi hỏi thực thi đa nền (cross platform) tức là không chỉ thực thi trên Windows mà còn trên Linux hay MacOS.



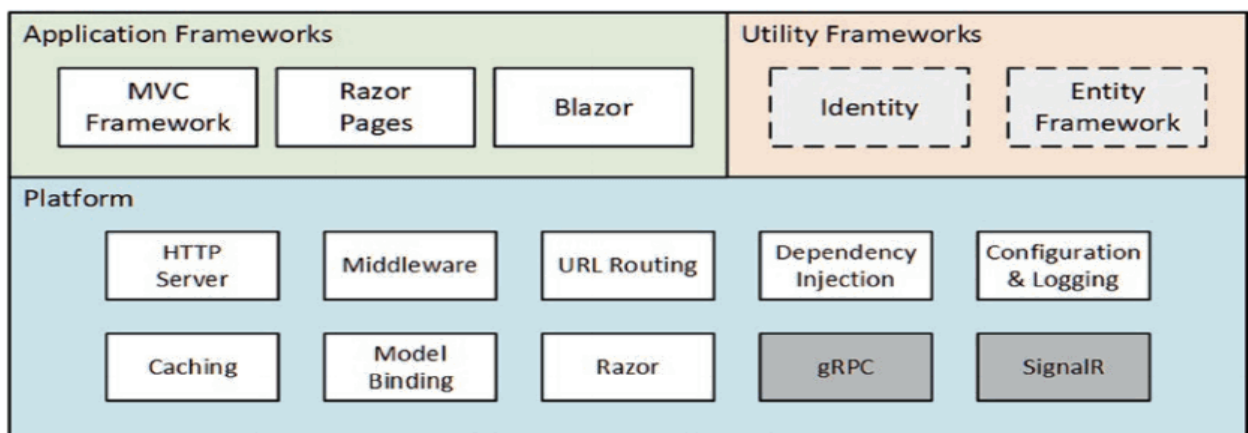
Hình 1.4 Sự ra đời ASP.NET Core

ASP.NET Core tức công nghệ ASP.NET kế thừa nền tảng cũ .NET Framework và phát triển thêm những tính năng mới trên nền tảng .NET Core ra đời và Microsoft mang đến ASP.NET Core có những sự tiến hóa và hoàn hảo mới.

ASP.NET Core được biết đến với một số thay đổi mới mẻ lớn về kiến trúc, đó là kết quả từ việc học hỏi rất nhiều từ các Framework Module hóa khác. Hiện nay, ASP.NET Core không còn phụ thuộc cố định vào System.Web.dll nữa mà tập hợp các gói, các module hay còn được gọi là các Nuget Packages.

1.2.2 Cấu trúc của ASP.NET Core

Cấu trúc ASP.NET Core bao gồm 3 phần: Application Frameworks, Utility Frameworks, và Platform.



Hình 1.5 Cấu trúc của ASP.NET Core

- **Application Frameworks:** Chứa những cái tên có lẽ tương đối quen thuộc như MVC Framework, Razor Pages hay Blazor. Đây là những framework giúp xây dựng các dạng khác nhau của ứng dụng web.
- **Utility Frameworks:** Chứa những thứ cảm tưởng như không liên quan đến Asp.net Core: Identity và Entity Framework. Khối này chứa những framework hỗ trợ cho ứng dụng, bao gồm bảo mật và cơ sở dữ liệu.
- **Platform:** Là những gì tạo nên nền tảng chung nhất mà mọi loại ứng dụng Asp.net Core đều cần sử dụng đến.

1.2.2.1 Application Frameworks

Application Framework chứa các framework: MVC, Razor Pages, Blazor . Các framework này giúp giải quyết các loại vấn đề khác nhau, và giải quyết cùng một vấn đề theo những cách thức khác nhau.

Nhìn nhận theo một cách khác, đây là những mẫu kiến trúc khác nhau cho ứng dụng web mà Microsoft hỗ trợ. Có thể lựa chọn bất kỳ framework nào phù hợp với yêu cầu của dự án và sở thích cá nhân. Do vậy, chúng không loại trừ nhau mà còn tương trợ lẫn nhau. Trong quá trình phát triển ứng dụng, Có thể chuyển đổi từ framework này sang framework khác, hoặc kết hợp chúng.

- **MVC** là framework xây dựng theo nguyên lý phân chia nhiệm vụ (Separation of Concerns, SoC). MVC phân chia chức năng thành các phần của ứng dụng ra làm các nhóm độc lập, gọi là
- **Razor Pages** là một framework đơn giản hướng tới xây dựng các trang web tự thân độc lập. Mô hình của Razor Pages đơn giản hơn so với MVC.
- **Blazor** là framework dành cho phát triển ứng dụng đơn trang (SPA) nhưng sử dụng ngôn ngữ C# thay cho JavaScript chạy trên trình duyệt. Đây là framework mới nhất và đang nhận được nhiều sự quan tâm. Blazor bao gồm hai mô hình chính: Blazor Server và Blazor WebAssembly

1.2.2.2 Utility Frameworks

Utility Frameworks chứa hai framework (không bắt buộc) nhưng lại được sử dụng gần như trong mọi ứng dụng ASP.NET Core.

- **Entity Framework Core** là một ORM (Object-Relational Mapping) giúp ứng dụng tương tác với cơ sở dữ liệu. Entity Framework Core giúp ánh xạ (hai chiều) giữa các bảng cơ sở dữ liệu (ví dụ, SQL Server) với các domain class của ứng dụng. Entity Framework Core có thể xem là phiên bản hiện đại đa nền tảng của Entity Framework – ORM chính của .NET Framework cổ điển. Nếu đã biết Entity Framework có thể dễ dàng chuyển đổi sang Entity Framework Core.
- **Asp.net Core Identity** là framework dành cho xác thực (authentication) và xác minh quyền (authorization) người dùng trong ứng dụng.

1.2.2.3 Platform

Phần platform chứa nhiều thành phần cấp thấp cần thiết cho việc nhận và xử lý truy vấn HTTP, cũng như tạo ra các phản hồi phù hợp.

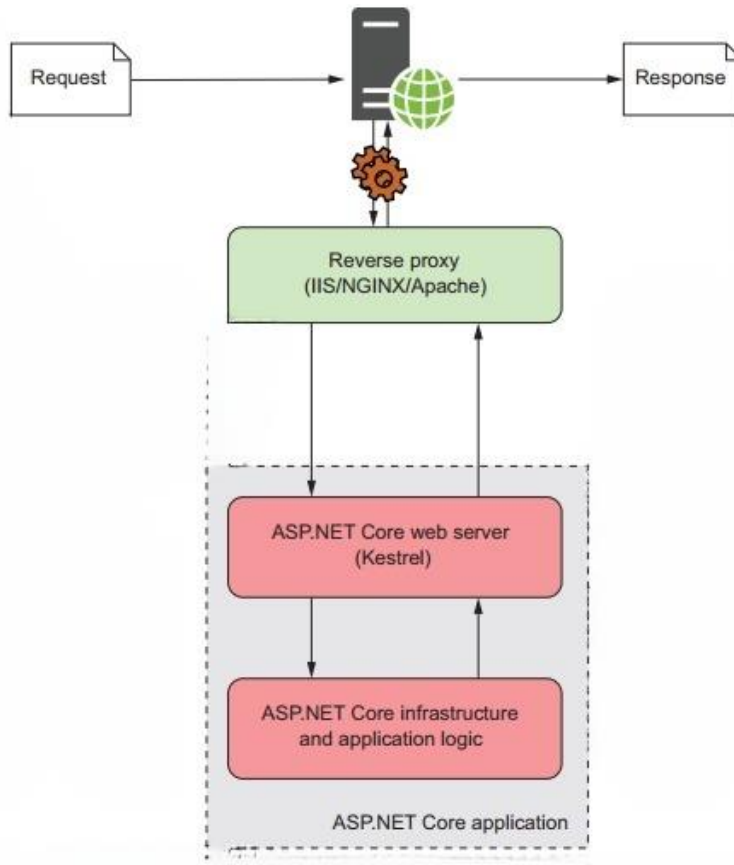
Mặc dù phần lớn thời gian trong quá trình phát triển ứng dụng làm việc chủ yếu với application framework, và bản thân các application framework cũng che đi rất nhiều phần của platform.

- **HTTP Server** còn gọi là built-in server với tên gọi Kestrel, có nhiệm vụ tiếp nhận truy vấn HTTP. Kestrel có thể hoạt động độc lập (tích hợp trong một ứng dụng khác) hoặc phối hợp với một web server thông thường (Apache, NGinx, IIS).
- **Middleware** là nhóm thành phần có nhiệm vụ xử lý truy vấn HTTP. Các middleware được xếp theo chuỗi. Khi truy vấn chạy qua mỗi khâu của chuỗi middle sẽ được xem xét xử lý.

- **URL Routing** là cơ chế ánh xạ chuỗi truy vấn HTTP sang thực thi một phương thức nào đó. Do vậy, mỗi URL (địa chỉ bạn gửi về server) sẽ tương ứng với thực thi một phương thức trên server.
- **Razor** là cơ chế sinh ra HTML từ dữ liệu và logic của chương trình. Razor được gọi là view engine trong Asp.net Core. Razor sử dụng loại cú pháp đặc biệt kết hợp giữa C# và HTML.
- **Model Binding** là cơ chế ánh xạ dữ liệu chứa trong truy vấn HTTP sang tham số của phương thức cần thực thi. Nhờ cơ chế Model Binding, việc xây dựng các phương thức trong Asp.net Core đơn giản như bất kỳ phương thức C# thông thường nào.
- **Dependency Injection** là cơ chế cho phép tự động sinh và chèn object vào một object khác. Asp.net Core xây dựng sẵn cơ chế này mà không cần đến một thư viện thứ ba (như Ninject, Unity).
- **Configuration & Logging** là cơ chế hỗ trợ cấu hình và lưu vết quá trình thực thi ứng dụng.
- **Caching** là cơ chế lưu tạm để tăng hiệu suất cho ứng dụng.
- **SignalR** là một framework riêng nhưng lại được sử dụng làm nền tảng cho ứng dụng Asp.net Core. SignalR cho phép tạo ra kênh truyền thông tốc độ cao (theo thời gian thực) và hai chiều giữa browser và web server. SignalR là nền tảng cho Blazor Server – loại ứng dụng đơn trang mà toàn bộ phần xử lý được đẩy về server và nhận lại kết quả (DOM) theo thời gian thực.
- **gRPC** là một chuẩn dành cho gọi hàm từ xa (Remote Procedure Call) đa nền tảng qua HTTP do Google (chữ g trong gRPC) phát triển. gRPC có thể hữu ích để các thành phần backend của ứng dụng (chạy trên server) trao đổi dữ liệu.

1.2.3 Cách ASP.NET Core làm việc

Người dùng từ trình duyệt web muốn gửi một yêu cầu đến ứng dụng ASP.NET Core cũng phải thông qua giao thức HTTP. Nhưng thay vì yêu cầu được gửi trực tiếp đến server (hay web server), nó phải thông qua một reverse-proxy server. Hiểu một cách đơn giản, một reverse-proxy server là một phần mềm có trách nhiệm nhận và chuyển tiếp yêu cầu từ người dùng đến server (hay web server). Các reverse-proxy server được công khai trên internet nhưng các web server thì chỉ có reverse-proxy server biết. Điều này làm tăng tính bảo mật và khả năng thực thi của các web server. Phần mềm reverse-proxy server trên Windows là IIS, trên Linux hay MacOS có thể là NGHINX hay Apache. Các bước thực thi được mô tả như hình sau đây:



Hình 1.6 Cơ chế hoạt động của ASP.NET Core

1. Yêu cầu được gửi đến reverse-proxy server trước khi đến web server thông qua HTTP Request
2. Yêu cầu được chuyển tiếp đến web server bởi reverse-proxy server
3. ASP.NET Core web server chuyển yêu cầu đến bộ phận xử lý
4. Yêu cầu được xử lý và phát sinh các nội dung hồi đáp đến trình duyệt
5. Kết quả xử lý được chuyển trở lại ASP.NET Core web server
6. ASP.NET Core web server chuyển kết quả đến reverse-proxy server
7. Kết quả hiển thị đến trình duyệt bởi HTTP Response

1.2.4 Đặc điểm của ASP.NET Core

Có thể thấy, ASP.NET Framework tồn tại bền bỉ trên hệ thống Windows từ lâu thì ASP.NET Core mới bắt đầu ra đời để hỗ trợ các tính năng cho .NET Framework như Windows Forms, WPF. ASP.NET Core có thể hoạt động trên bất kỳ nền tảng nào mà đảm bảo có thể khắc phục được những hạn chế của .NET Framework – chạy duy nhất trên nền tảng Windows. ASP.NET Core được thiết kế để tối ưu và cung cấp tốt cho các Development Framework triển khai trên đám mây hoặc chạy on-premise.

Về cơ bản, ASP.NET Core được tạo thành từ một thư viện bản Net. .NET Standard 2.0 có thể chạy ở bất cơ nào mà nó có thể hỗ trợ được. Đồng thời, ASP.NET Core trở thành một mã nguồn mở. Đây là sự thay đổi lớn và vô cùng quan trọng mà trước đây khó có lập trình viên nào nghĩ đến. Chính vì thế, ASP.NET Core thu hút đông đảo các lập trình viên sử dụng để xây dựng các trang web.

ASP.NET Core là một phiên bản thiết kế của ASP.NET 4.x có sự thay đổi kiến trúc giúp Framework nhẹ nhàng hơn và kết hợp cả tính Module nhiều hơn nữa.

Với một nền tảng có hiệu suất và tính tương thích hiệu quả, các lập trình viên vẫn trung thành xây dựng các web bằng ASP.NET Core.

1.2.5 Ưu điểm của ASP.NET Core

- ASP.NET Core được thay đổi một số kiến trúc nên dẫn đến Modular Framework nhỏ hơn.
- ASP.NET Core không còn phụ thuộc vào nền tảng system.web.dll, Framework này dựa trên một tập hợp nhiều yếu tố của Nuget Packages.
- Có thể tối ưu ứng dụng của mình dễ dàng thông qua những Nuget Packages cần thiết.
- Những ứng dụng web được thiết kế trên nền tảng ASP.NET Core sẽ ít tiêu hao dung lượng bộ nhớ, bảo mật chặt chẽ, tốc độ thực thi nhanh, hiệu năng hoạt động tốt và ổn định.
- ASP.NET Core giúp giảm dịch vụ, dễ bảo trì ứng dụng và tiết kiệm chi phí.
- Các ứng dụng ASP.NET Core được xây dựng và khởi tạo ở đa dạng các nền tảng như Windows, Mac và Linux.
- Được phát triển từ .NET Core, ASP.NET Core hỗ trợ chất lượng App Vesioning.
- Mang đến những công cụ và hàng loạt tính năng hiện đại, đơn giản hóa quy trình phát triển web.
- Tương thích với hệ thống xây dựng web UI và web APIs.
- Có thể tích hợp những client – side Frameworks hiện đại và những luồng phát triển.
- Cấu hình dựa trên mô hình đám mây có sẵn.
- Dependency Injection (DI) được xây dựng sẵn.
- Đa dạng cấu hình, thân thiện với nhiều môi trường.
- Có tính năng host trên IIS hoặc self-host trong Process của riêng mình.

- Chuyển thực thể, các thành phần module, dùng chung toàn bộ Nuget Package.
- Mã nguồn mở và tập trung vào cộng đồng.

1.2.6 ASP.NET và ASP.NET Core

ASP.NET và ASP.NET Core là 2 framework hỗ trợ các nhà phát triển các ứng dụng web dựa trên nền tảng .Net Framework khác nhau

- **ASP.NET** là một framework do Microsoft phát triển miễn phí trên nền tảng Window giúp hỗ trợ các nhà phát triển có thể tạo ra các websites và web applications dựa trên các ngôn ngữ C#, Visual Basic và HTML, CSS, JavaScript. Sau khi ra đời từ năm 2002, ASP.NET đã trải qua rất nhiều phiên bản, nó hỗ trợ nhiều framework con để tạo ra các ứng dụng web như : Web Forms, ASP.NET MVC, and ASP.NET Web Pages và nhiều công nghệ kèm theo để làm cho các ứng dụng viết trên nó trở lên đa dạng và linh động hơn.
- **ASP.NET Core** là một framework mã nguồn mở, được Microsoft và các nhà phát triển tạo ra nhằm cung cấp các ứng dụng web hỗ trợ đa nền tảng chứ không chỉ trên hệ điều hành Microsoft Windows thông thường như ASP.NET. Được giới thiệu từ năm 2016, ASP.NET Core đang được kỳ vọng tạo ra những ứng dụng mới có hiệu suất xử lý tốt hơn và ngày càng được hoàn thiện bởi sự chung sức của cộng đồng

ASP.NET	ASP.NET Core
Được xây dựng chỉ để dành cho Windows.	Được xây dựng dành cho cả Windows, Mac và Linux.
Có hiệu suất tốt	Có hiệu suất cao hơn cả ASP.NET 4x.
Có thể chạy được trên .Net Framework hay được gọi là Full .Net Framework.	Có thể chạy trên .Net Core và Full .Net Framework.

Asp.Net hỗ trợ Web Forms, Asp.Net MVC và ASP.NET web Pages.	Asp.Net Core hỗ trợ cho các trang Web MVC, Web API và Asp.Net được thêm vào ban đầu trong .Net Core 2.0, không hỗ trợ cho Web Forms.
Chỉ sử dụng IIS phụ thuộc vào System.web.dll.	Asp.Net Core không phụ thuộc vào IIS và System.web.dll.
Sử dụng ngôn ngữ C#, VB, WCF, WPF và WF.	Chỉ hỗ trợ ngôn ngữ C#, F# và VB trong thời gian ngắn, không hỗ trợ WCF, WPF và WF. Tuy nhiên, Asp.Net Core lại có thể hỗ trợ cho các thư viện WCF có sẵn.
Asp.Net MVC có thêm các ứng dụng như Web.config, Global.asax, Application Start.	Asp.Net Core đang hỗ trợ Appsettings.json, không hỗ trợ tệp Web.config và Global.asax.
Hỗ trợ vùng chứa không được đánh giá quá cao.	Hỗ trợ vùng chứa phù hợp cho các triển khai như Docker.
Tất cả các phiên bản chính thức đều được hỗ trợ.	Hỗ trợ Core từ Visual Studio 2015 cập nhật lần thứ 3
Người dùng cần biên dịch lại sau khi thay đổi mã.	Khi làm mới Core Browser sẽ tự động biên dịch và thực thi mã mà không cần phải dịch lại.

1.2.7 Lợi ích khi sử dụng ASP.NET Core

- **Hiệu suất nâng cao:** Công ty phát triển ASP.NET coi ứng dụng hoạt động tốt như thế nào là yếu tố chính trong khi lựa chọn khung để phát triển ứng dụng. Và trong trường hợp này, ASP.NET Core nhanh hơn nhiều so với ASP.NET MVC và đã cho thấy kết quả tuyệt vời so với các khung công tác

khác. Một lý do cho hiệu suất nhanh chóng của khuôn khổ là thực tế là hệ thống tự động tối ưu hóa các mã của nó để cải thiện hiệu suất.

- **Hỗ trợ đa nền tảng:** Cần đảm bảo rằng ứng dụng sẽ hoạt động tốt trên mọi nền tảng khi tiến hành phát triển ứng dụng. ASP.NET Core là đa nền tảng và chạy trên Windows, Linux, Mac và tất cả các thiết bị khác. Do đó, hệ thống cho phép các nhà phát triển chọn bất kỳ hệ điều hành nào để thuận tiện cho họ vì nó cực kỳ linh hoạt.
- **Ít mã hơn:** Chất lượng của mã xác định chất lượng của một ứng dụng. ASP.NET Core cho phép các nhà phát triển viết ít câu lệnh hơn. Do đó, cấu trúc mã trở nên dễ dàng hơn và ít phải viết mã hơn. Điều này làm cho việc phát triển ứng dụng dành cho thiết bị di động trở nên hiệu quả về mặt chi phí đối với các tổ chức. Ngoài ra, nó cũng cung cấp nhiều quyền kiểm soát hơn cho các nhà phát triển khi quy trình có liên quan và đơn giản hóa việc gỡ lỗi.
- **Bảo trì dễ dàng:** ASP.NET yêu cầu ít mã hơn và ít mã hơn sẽ dễ bảo trì hơn. Các nhà phát triển có thể dễ dàng tối ưu hóa mã trong ASP.NET Core và tiết kiệm thời gian bảo trì ứng dụng.
- **Hỗ trợ phát triển ứng dụng web dựa trên đám mây:** Một lợi ích khác của ASP.NET Core là nó cung cấp các kiểu phát triển ứng dụng khác nhau và ứng dụng web dựa trên đám mây. Vì vậy, cách tiếp cận này là phù hợp nhất cho các doanh nghiệp và doanh nghiệp đã sẵn sàng để mở rộng. Phát triển dựa trên đám mây cung cấp cho các ứng dụng web tính linh hoạt, khả năng truy cập, tích hợp dễ dàng hơn, bảo vệ dữ liệu và hơn thế nữa.
- **Mã nguồn mở:** ASP.NET Core là mã nguồn mở, có nghĩa là bất kỳ chuyên gia ASP.NET Core nào cũng có quyền truy cập vào mã khung. Tất cả các nhà phát triển .NET Core đều có thể cải tiến công nghệ và sửa đổi nó theo

nhu cầu phát triển ứng dụng của họ. Điều này giúp các nhà phát triển tạo ra các giải pháp web tốt nhất với ASP.NET Core.

- **Lưu trữ:** .NET Core cung cấp một máy chủ nội bộ cho mọi ứng dụng web ASP.NET Core theo mặc định. Nó cho phép chạy các ứng dụng ASP.NET Core trên Windows, Mac hoặc Linux. Hơn nữa, nó nhẹ và hỗ trợ lớp công bảo mật (SSL).
- **An ninh tốt hơn:** ASP.NET Core có một số tính năng tích hợp cho phép các nhà phát triển tạo các ứng dụng web an toàn hơn. Công nghệ này giúp duy trì việc thực thi, xác thực, ủy quyền và bảo vệ dữ liệu HTTPS dễ dàng hơn.
- **Phát triển nhanh chóng:** Phát triển nhanh có thể chứng minh là rất có lợi cho các dự án cần được quay vòng trong một thời gian ngắn, chẳng hạn như trong vòng hai đến ba tháng. Trong mô hình phát triển này, tập trung nhiều hơn vào các nhiệm vụ phát triển và tạo mẫu thay vì lập kế hoạch. ASP.NET Core linh hoạt và thích ứng với các thay đổi đồng thời giảm rủi ro tổng thể của dự án, mã hóa thủ công và lỗi đồng thời.
- **Khả năng di động tốt hơn:** ASP.NET Core có tính di động cao và tính di động có thể làm giảm đáng kể chi phí phát triển web. Khung cho phép di chuyển ứng dụng dễ dàng giữa các máy chủ. Ngoài ra, nhiều nhà phát triển có thể làm việc trên ứng dụng cùng một lúc.

1.3 Giới thiệu về mô hình MVC

1.3.1 Mô hình MVC

Mô hình MVC (MVC Design Pattern) là viết tắt của Model-View-Controller. Đó là một mẫu kiến trúc, mô hình lập trình phổ biến, được các lập trình viên sử dụng để tạo cấu trúc cho trang web, việc thiết kế phần mềm theo yêu cầu hay phát triển các ứng dụng. Theo góc nhìn rộng hơn, mô hình MVC được sử dụng để mô tả

quá trình làm web của đại đa số các ngôn ngữ lập trình phổ biến chẳng hạn như ngôn ngữ, C#, PHP, Ruby, Python hay JavaScript.

1.3.2 Cấu trúc của MVC

1.3.2.1 Models

Các đối tượng Models là một phần của ứng dụng, các đối tượng này thiết lập logic của phần dữ liệu của ứng dụng. Thông thường, các đối tượng model lấy và lưu trạng thái của model trong cơ sở dữ liệu (CSDL). Ví dụ như, một đối tượng Product (sản phẩm) sẽ lấy dữ liệu từ CSDL, thao tác trên dữ liệu và sẽ cập nhật dữ liệu trở lại vào bảng Products ở SQL Server.

Trong các ứng dụng nhỏ, model thường là chỉ là một khái niệm nhằm phân biệt hơn là được cài đặt thực thụ, ví dụ, nếu ứng dụng chỉ đọc dữ liệu từ CSDL và gửi chúng đến view, ứng dụng không cần phải có tầng model và các lớp liên quan. Trong trường hợp này, dữ liệu được lấy như là một đối tượng model (hơn là tầng model).

1.3.2.2 Controllers

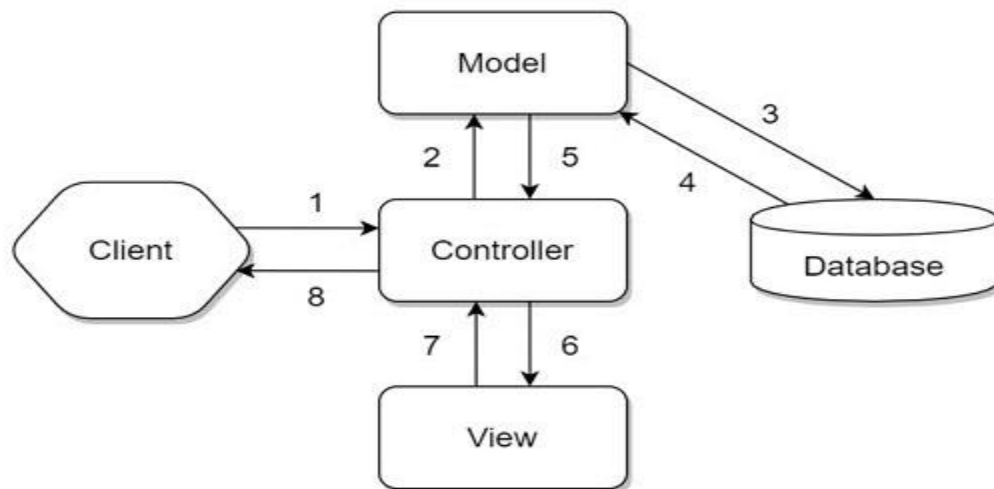
Controller là các thành phần dùng để quản lý tương tác người dùng, làm việc với model và chọn view để hiển thị giao diện người dùng. Trong một ứng dụng MVC, view chỉ được dùng để hiển thị thông tin, controller chịu trách nhiệm quản lý và đáp trả nội dung người dùng nhập và tương tác với người dùng. Ví dụ, controller sẽ quản lý các dữ liệu người dùng gửi lên (query-string values) và gửi các giá trị đó đến model, model sẽ lấy dữ liệu từ CSDL nhờ vào các giá trị này.

1.3.2.3 Views

Views là các thành phần dùng để hiển thị giao diện người dùng (UI). Thông thường, view được tạo dựa vào thông tin dữ liệu model. Ví dụ như, view dùng để cập nhật bảng Products sẽ hiển thị các hộp văn bản, drop-down list, và các check box dựa trên trạng thái hiện tại của một đối tượng Product.

1.3.3 Luồng xử lý trong mô hình MVC

- View sẽ hiển thị ra phần nhập form tiêu đề và nội dung. Người dùng sẽ nhập nội dung cần, sau đó gửi yêu cầu đến server, Controller sẽ tiếp nhận yêu cầu.
- Dữ liệu đầu vào sẽ được xử lý và quyết định luồng đi tiếp theo (trả về kết quả hay tương tác với database để lấy dữ liệu) thông qua bộ phận Controller. Trong trường hợp tương tác với database để lấy dữ liệu, Controller sẽ gửi thông báo đến Model để lấy dữ liệu đầu ra. Trong trường hợp còn lại, Controller sẽ trả về kết quả cho Client.
- Model tương tác với Database để truy xuất dữ liệu phù hợp với yêu cầu, sau đó database sẽ trả dữ liệu cho Model theo yêu cầu ban đầu.
- Sau đó, Model gửi trả về dữ liệu cho Controller xử lý
- Controller sẽ gửi thông báo đi kèm với dữ liệu phù hợp cho View, để View hiển thị dữ liệu phù hợp với theo yêu cầu
- Sau khi xử lý hiển thị dữ liệu, View trả kết quả (HTML, XML hoặc JSON...) về cho Controller
- Sau khi hoàn tất, Controller sẽ trả kết quả lại cho Client



Hình 1.7 Luồng xử lý trong mô hình MVC

1.3.4 Nhược điểm và ưu điểm của mô hình MVC

1.3.4.1 Ưu điểm

- Tạo mô hình chuẩn cho từng dự án, từ đó tiếp cận với ứng dụng dễ dàng hơn ngay cả với những người không có chuyên môn
- Hỗ trợ quá trình phát triển nhanh chóng nhờ các bộ phận hoạt động độc lập với nhau. Từ đó các lập trình viên dễ dàng hát triển, quản lý, vận hành, bảo trì trên từng bộ phận mà không làm ảnh hưởng đến toàn hệ thống, đồng thời dễ dàng kiểm soát được luồng xử lý của ứng dụng.
- Trình tự xử lý rõ ràng, dễ dàng kiểm tra, rà soát lỗi phần mềm trước khi thông tin trích xuất sau cùng hiển thị trước người dùng, từ đó đảm bảo chất lượng và độ uy tín của thông tin cao hơn
- Sở hữu bộ control ưu việt trên nền tảng các ngôn ngữ lập trình hiện đại như CSS, HTML, Javascript với nhiều hình thức khác nhau
- Có khả năng cung cấp đồng thời nhiều khung View lưu trữ dữ liệu. Nhờ đó tiết kiệm được diện tích bằng thông một cách tối ưu, đặc biệt trong trường hợp có nhiều yêu cầu được thực hiện thì kích thước càng tệp càng lớn
- Mô hình MVC truyền tải dữ liệu nhưng không định dạng lại dữ liệu, từ đó trạng thái dữ liệu được bảo tồn và sử dụng cho những lần sau này
- Mô hình có kết cấu tương đối đơn giản, dễ hiểu, xử lý những nghiệp vụ đơn giản

1.3.4.2 Nhược điểm

- Chỉ phù hợp với những dự án lớn, không thích hợp việc phát triển các ứng dụng nhỏ vì mô hình này yêu cầu người dùng phải lưu trữ một số lượng lớn các file dữ liệu khác nhau. Điều đó tạo nên sự cồng kềnh và phức tạp trong quá trình phát triển cũng như thời gian trung chuyển dữ liệu
- Phân chia công việc và tác vụ không đồng đều giữa các layer, vì phần Model phải đảm nhiệm hầu hết các tác vụ quan trọng
- Việc hiển thị của layer View phải phụ thuộc vào cả Controller và Model nên sự hỗ trợ cho quá trình kiểm thử không quá tốt bởi dù trên lý thuyết chúng

độc lập với nhau. Nếu không thể nhận yêu cầu và cũng không có dữ liệu được xử lý để hiển thị thì View cũng không hiển thị được gì. Để tiến hành kiểm thử trên View, chúng ta cần giả lập cả Controller và Model.

- Đối với các ứng dụng phức tạp, đòi hỏi quy trình xử lý nghiệp vụ thì MVC cũng không phải là lựa chọn tối ưu

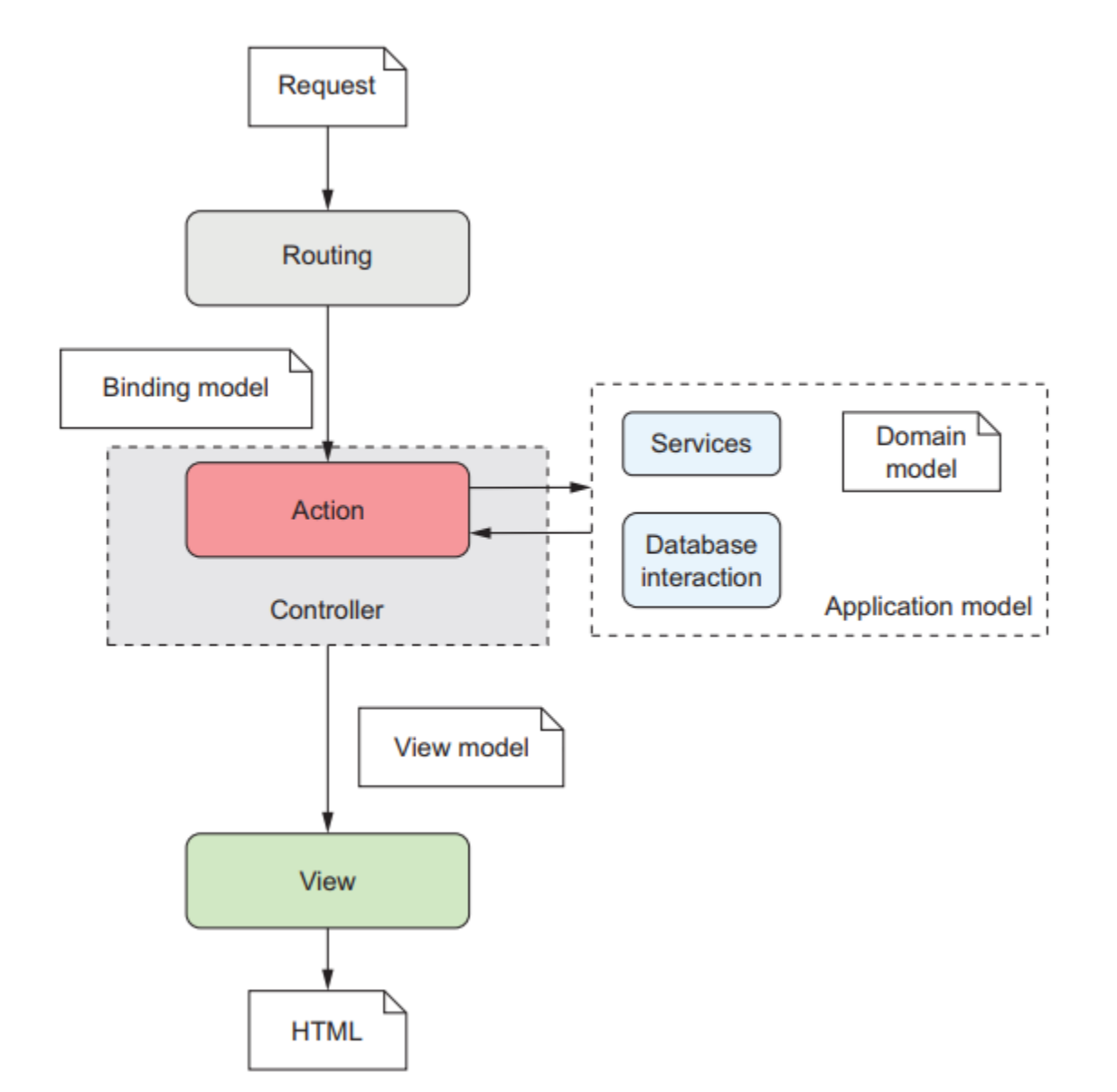
1.3.5 Ứng dụng của mô hình MVC

- ✓ MVC được ứng dụng phổ biến trong nhiều ngôn ngữ lập trình khác nhau, mà tiêu biểu có thể kể đến như ứng dụng ASP.NET MVC hay PHP MVC. Ngày nay có rất nhiều Framework, Source Code của mã nguồn mở của các website sử dụng kiến trúc lập trình MVC cho các ứng dụng của họ.
- ✓ Hệ thống model view controller sẽ cho phép phát triển toàn diện hệ thống front-end lẫn back-end mà không cần có sự can thiệp, chia sẻ, chỉnh sửa các tập tin trong khi một hoặc hai bên vẫn đang trong thao tác làm việc. Bên cạnh đó, việc vận hành quy trình MVC đơn giản cũng là yếu tố để khiến cấu trúc MVC được triển khai rộng rãi như hiện nay.
- ✓ Hiện nay mô hình MVC được áp dụng nhiều và phổ biến trên toàn Thế Giới, hầu như tất cả các công ty viết phần mềm, lập trình web app hiện nay đều phải ít nhất 1 lần ứng dụng mô hình MVC. Để việc triển khai dự án cho đội nhóm diễn ra nhanh chóng và chuyên nghiệp hơn thì nên áp dụng mô hình MVC rất tốt.

1.4 Tổng quan về dự án ASP.NET Core MVC

ASP.NET Core MVC là tên gọi của framework trong ASP.NET Core thực thi mô hình kiến trúc MVC. Framework này giúp phát triển nhiều loại ứng dụng khác nhau, từ ứng dụng web truyền thống đến ứng dụng đơn trang hoặc Web API.

1.4.1 Mẫu kiến trúc MVC trong ASP.NET Core

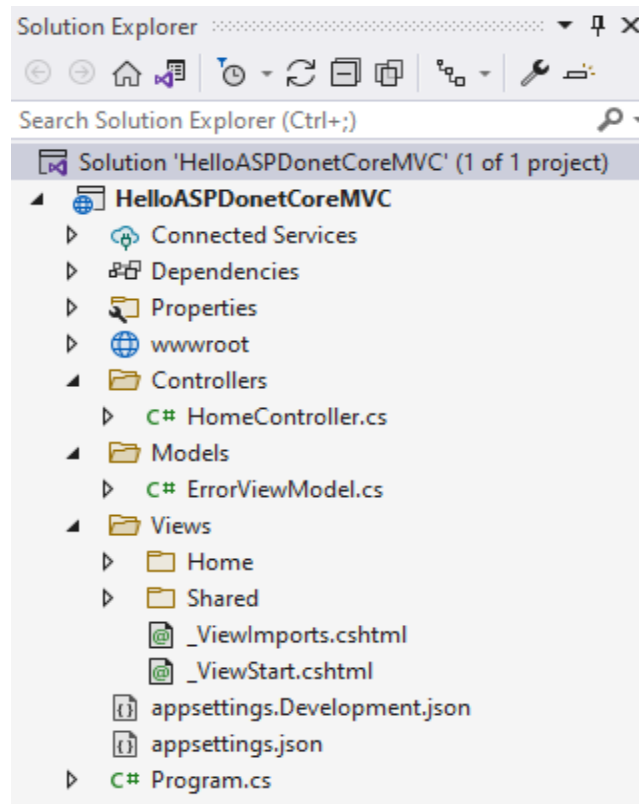


Hình 1.8 Sơ đồ hoạt động ứng dụng kiến trúc MVC trong ASP.NET Core

- Đây là một chu kỳ xử lý từ lúc nhận được truy vấn HTTP cho đến khi sinh ra HTML.
- Truy vấn HTTP sẽ được cơ chế routing ánh xạ sang một phương thức xác định gọi là action.

- Trong ASP.NET Core MVC, action là phương thức được thực thi để đáp ứng lại một truy vấn.
- Để thực thi, action cần đến dữ liệu đầu vào chứa trong truy vấn HTTP. Dữ liệu được trích ra từ truy vấn thông qua cơ chế model binding
- Binding model là một object đóng vai trò “thùng chứa” dữ liệu trích xuất ra từ truy vấn để cung cấp cho action. Binding model là kết quả hoạt động của cơ chế model binding và là tham số đầu vào cho action.
- Controller trong ASP.NET Core là class chứa các action có quan hệ nhất định.
- Action khi thực thi sẽ tương tác với các thành phần còn lại của ứng dụng như các dịch vụ, cơ sở dữ liệu.
- Với cách tiếp cận DDD (Domain-driven Design), phần dữ liệu nghiệp vụ được thể hiện qua các domain model
- Tất cả các thành phần dịch vụ, domain model, v.v., được gọi chung là application model
- Quá trình tương tác này sẽ sinh ra dữ liệu phục vụ cho hiển thị, gọi là view model.
- View model là object đơn giản chứa dữ liệu cần thiết để sinh ra giao diện. Thông thường view model là một biến thể của dữ liệu lấy được từ application model cùng với dữ liệu phụ trợ cho hiển thị (như tiêu đề, phân trang, v.v.).
- View trong ASP.NET Core MVC là các trang Razor chứa loại mã hỗn hợp C# + HTML theo cú pháp Razor. Vì vậy người ta cũng thường gọi view trong ASP.NET Core MVC là Razor view . Kết quả xử lý của Razor view là HTML.

1.4.2 Cấu trúc một dự án ASP.NET Core MVC



Hình 1.9 Cấu trúc một dự án ASP.NET Core MVC

Từ cửa sổ Solution Explorer, Có thể lướt qua cấu trúc tổng thể của một dự án ASP.NET Core MVC với một số thành phần sau:

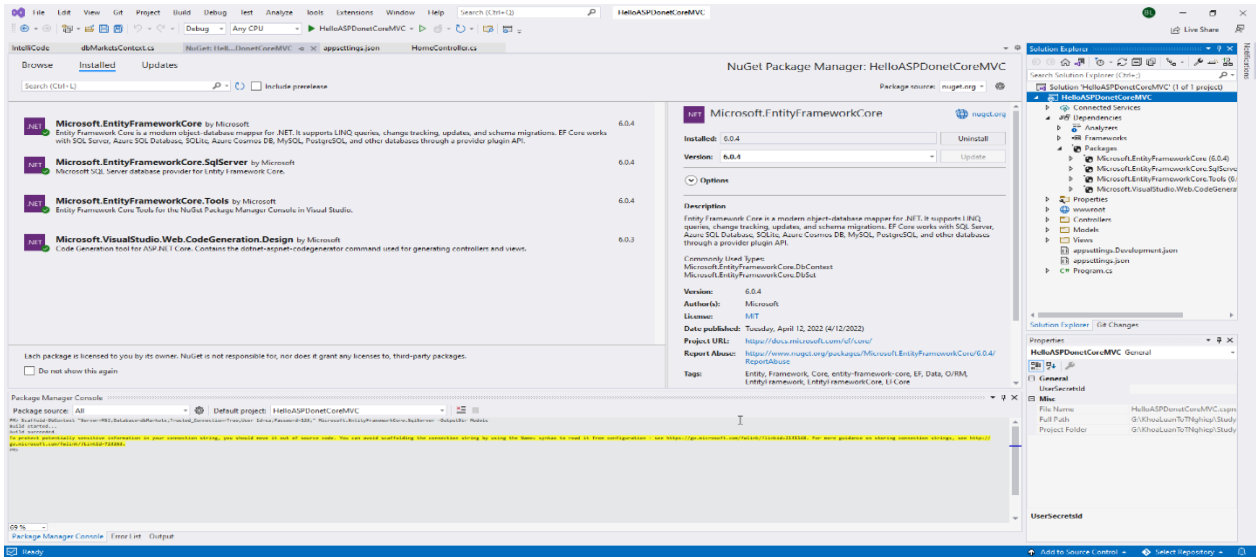
- Vị trí cao nhất trong cửa sổ Solution Explorer là thư mục gốc của dự án được lồng trong một thư mục solution. Visual Studio dùng khái niệm solution để làm việc với nhiều dự án, trong trường hợp này chỉ có một dự án. Tại thư mục này chúng ta sẽ tìm thấy một tập tin solution (có phần mở rộng **.sln**).
- Kế tiếp thư mục gốc dự án là tập tin quan trọng nhất của dự án, tập tin **HelloASPDnetCoreMVC.csproj**.
- Vì dự án ta đang dùng mô hình MVC nên các thư mục quan trọng tiếp theo của chúng ta sẽ là các thư mục **Controllers**, **Models** và **Views** chứa các tập tin dùng để xây dựng dự án.

- Thư mục **Properties** chứa tập tin **launchSettings.json** kiểm soát cách Visual Studio sẽ chạy và debug ứng dụng.
- Thư mục **wwwroot** là thư mục đặc biệt cho phép các trình duyệt web có thể truy cập trực tiếp đến nội dung bên trong nó như các tập tin CSS, JS, hình ảnh hay các tập tin HTML. Các trình duyệt sẽ không thể truy cập được các tập tin này nếu chúng ở bên thư mục **wwwroot**.
- **Properties** và **wwwroot** được xem như là hai thư mục đặc biệt, được đặt phía trên cửa sổ Solution Explorer, gần mục dự án và không tuân theo thứ tự các chữ cái. Phía trên hai thư mục này có hai mục đặc biệt hơn gọi là **Dependencies** và **Connected Services** chứa các thành phần phụ thuộc đến dự án như các gói NuGet, các dịch vụ từ xa, các thành phần hướng client.
- Tập tin **appsettings.json** cung cấp thông tin cấu hình ứng dụng tại thời điểm thực thi hay thời điểm biên dịch.
- Cuối cùng là tập tin **Program.cs** kiểm soát cấu hình và quá trình khởi động của ứng dụng tại thời điểm thực thi.

1.4.3 Kết nối với hệ quản trị cơ sở dữ liệu Sql Server 2019

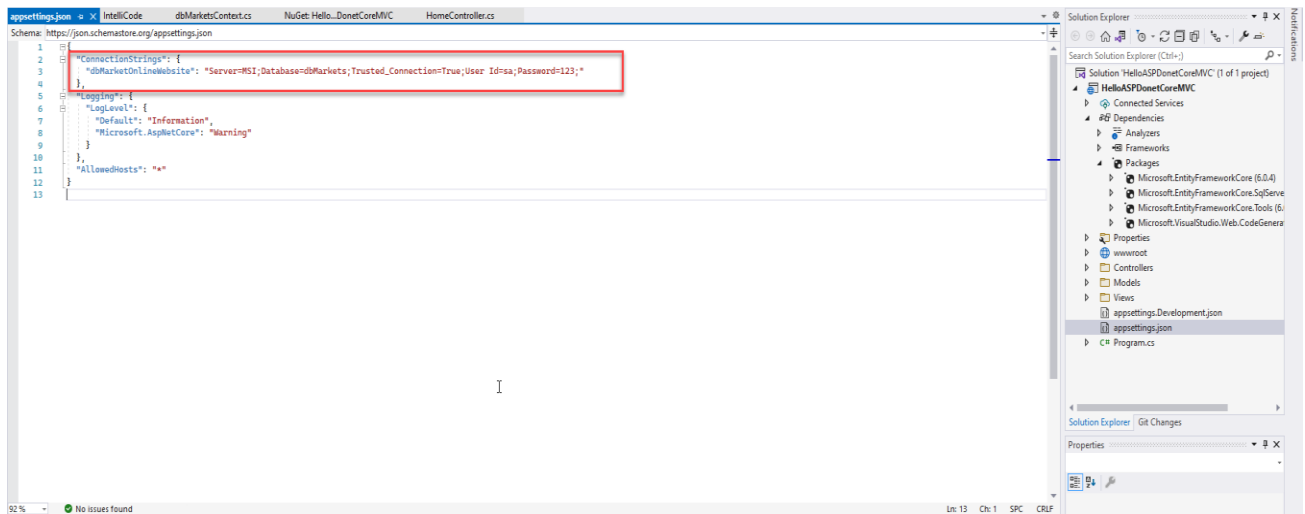
- **Bước 1:** Cài đặt các gói thư viện Entity Framework:
 - + Microsoft.EntityFrameworkCore.SqlServer
 - + Microsoft.EntityFrameworkCore.Tools
 - + Microsoft.VisualStudio.Web.CodeGeneration.Design
 - + Microsoft.EntityFrameworkCore

Tìm hiểu ASP.NET Core MVC để xây dựng ứng dụng đi chợ online



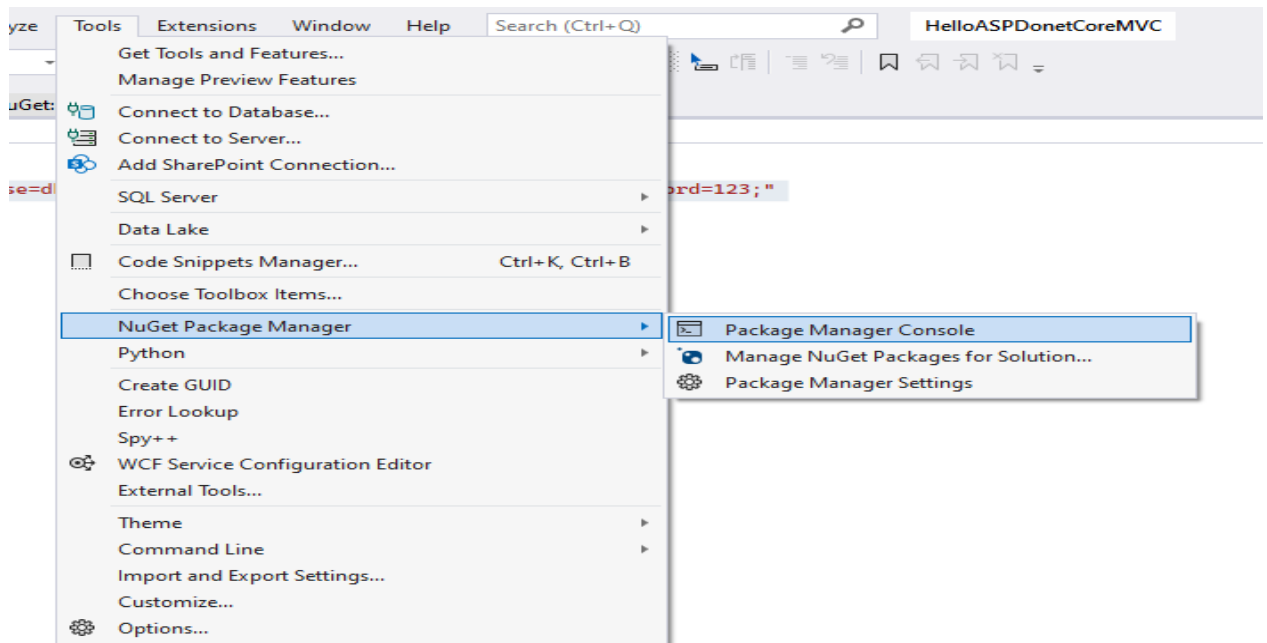
Hình 1.10 Bước 1 Cài đặt các gói thư viện Entity Framework

- **Bước 2:** Thêm đường dẫn kết nối với cơ sở dữ liệu “ "ConnectionStrings": { "dbMarketOnlineWebsite": "Server=MSI ; Database=dbMarkets ; Trusted_Connection=True ; User Id=sa;Password=123 ; " }, ” vào **appsettings.json**



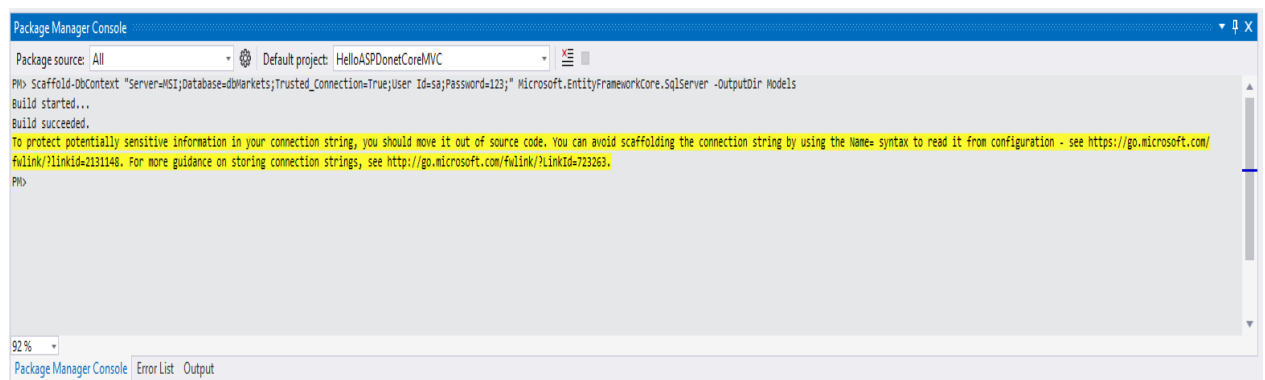
Hình 1.11 Bước 2 Cấu hình kết nối với cơ sở dữ liệu

- **Bước 3:** Chọn **Tools** -> Chọn **NuGet Package Manager** -> Chọn **Package Manager Console**



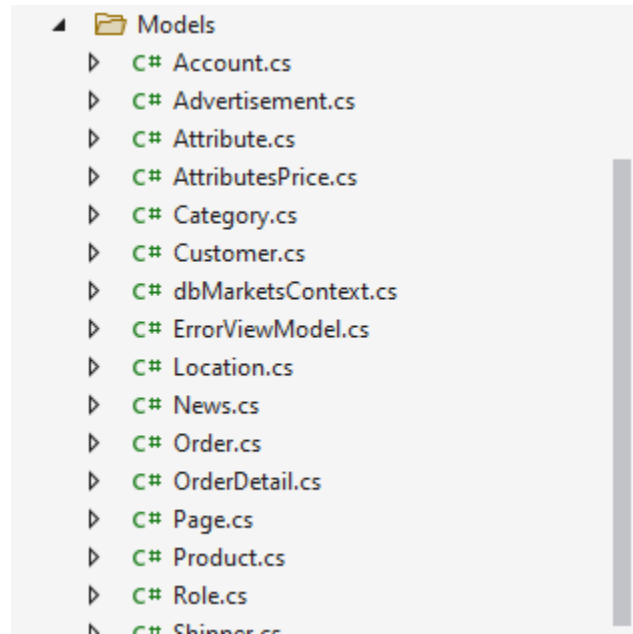
Hình 1.12 Bước 3 Hiển thị màn hình Package Manager Console

- **Bước 4:** Thêm lệnh “Scaffold-DbContext "Server=MSI ; Database=dbMarkets ; Trusted_Connection=True ; User Id=sa ; Password=123 ; " Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models” vào màn hình Console và nhấn **Enter**



Hình 1.13 Bước 4 Thêm lệnh kết nối cơ sở dữ liệu

- **Bước 5:** Sau khi kết nối thành công cơ sở dữ liệu Model sẽ tạo các lớp ánh xạ từ các bảng của database



Hình 1.14 Bước 5 Kết nối thành công cơ sở dữ liệu

CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG ĐI CHỢ ONLINE

2.1 Phân tích thực trạng

Những năm gần đây mua hàng trực tuyến đang dần phổ biến trong kỷ nguyên chuyển đổi số, các sàn thương mại điện tử cũng phát triển nhanh chóng để bắt kịp xu hướng mới. Vậy đâu là yếu tố giúp mua sắm online bùng nổ và trở nên phổ biến như vậy?

Đại dịch COVID-19 khiến cho nhiều hoạt động của con người gần như bị đóng băng. Tuy vậy, giãn cách xã hội và việc thường xuyên phải ở nhà lại mở ra cơ hội mới cho các hoạt động trực tuyến phát triển. Theo kết quả khảo sát do Facebook và Bain & Company hợp tác thực hiện, trong năm 2021, 5 hoạt động trên không gian trực tuyến được người tiêu dùng dành nhiều thời gian nhất chính là mạng xã hội, nhắn tin, xem video, thương mại điện tử và gửi email.

Theo đó, tỷ lệ người dùng internet tham gia mua sắm trực tuyến đã tăng từ 77% trong năm 2019 lên 88% trong năm 2020 (theo Sách trắng Thương mại điện tử năm 2021). Do tình hình dịch bệnh, người dân không được đi ra ngoài, mua sắm online dường như là "cứu cánh" giúp người tiêu dùng đáp ứng những nhu cầu của bản thân.

Mua hàng online có thể đáp ứng các nhu cầu tiêu dùng của khách hàng, đồng thời đảm bảo yếu tố an toàn, bảo vệ và nâng cao sức khỏe người tiêu dùng trong thời kỳ dịch bệnh. Khách hàng vẫn mua được món hàng mình ưng ý mà không phải di chuyển đến nhiều địa điểm mua sắm. Đối với những người không thích ra ngoài thì cách thức mua hàng online này như một phao cứu sinh giúp họ tự mua được đồ dùng cho bản thân mà không cần ra ngoài.

Không chỉ vậy, khi mọi người ở trong nhà một thời gian lâu dài có thể bị stress, mua sắm cũng là một cách thức giúp giảm đi những áp lực và cải thiện tâm

trạng hiệu quả hơn. Mua sắm trực tuyến không phải là một phương pháp tạm thời để đối phó với dịch bệnh mà có thể trở thành xu hướng khi các sàn thương mại điện tử kết hợp với các doanh nghiệp đem đến nhiều cải thiện vượt trội, mang đến trải nghiệm mua sắm hiện đại và thông minh hơn cho người tiêu dùng.

Từ thực trạng này, một website đi chợ online sẽ đáp nhu cầu mua sắm nhu yếu phẩm của người tiêu dùng. Đây là nơi mọi người có thể dễ dàng tìm kiếm những mặt hàng cần thiết trong cuộc sống nhưng vẫn đảm bảo chất lượng và uy tín.

2.2 Yêu cầu bài toán

Đây là website giúp người tiêu dùng mua sắm các sản phẩm phục vụ cho cuộc sống hằng ngày thay vì phải ra ngoài mua đi chợ, đi siêu thị tới những nơi đông đúc trong mùa dịch bệnh này để mua nhu yếu phẩm thì người dùng có thể tìm kiếm và mua trực tiếp các mặt hàng cần thiết trên website.

Người dùng phải đăng ký tài khoản để có thể mua hàng online. Sau khi đăng ký tài khoản thành công, người dùng có thể đăng nhập vào website bằng tài khoản đã tạo.

- Người dùng đăng nhập vào hệ thống có thể thực hiện các chức năng:
 - Người dùng có thể xem, cập nhật thông tin cá nhân của mình
 - Người dùng có thể đổi mật khẩu, hoặc lấy lại mật khẩu khi quên mật khẩu
 - Người dùng quản lý đơn hàng
 - Người dùng có thể hủy đơn hàng
 - Người dùng có thể xem lịch sử chi tiết đơn hàng
 - Người dùng tìm kiếm, lọc và sắp xếp sản phẩm
 - Người quản lý giỏ hàng, cập nhật và thay đổi giỏ hàng, thanh toán, đặt hàng
 - Người dùng có thể xem tin thông tin chi tiết sản phẩm

- Quản trị hệ thống của website khi đăng nhập vào hệ thống:
 - Quản trị viên có thể quản lý mặt hàng phục vụ nhu cầu tiêu dùng của khách hàng
 - Quản trị viên có thể quản lý danh mục sản phẩm, phân loại từng mặt hàng
 - Quản trị viên có thể quản lý đơn hàng đặt hàng khách hàng, thông tin giao hàng, trạng thái đơn hàng
 - Quản trị viên có thể quản lý thông tin của khách hàng
 - Quản trị viên có thể quản lý tài khoản truy cập sử dụng hệ thống
 - Quản trị viên có thể quản lý quyền truy cập cho từng tài khoản sử dụng hệ thống
 - Quản trị viên có thể quản lý thông tin giao hàng trên khắp đất nước như: tỉnh/thành, quận/huyện, phường/xã
 - Quản trị viên có thể quản lý tin tức, blog, bài viết về sản phẩm
 - Quản trị viên có thể quản lý shipper cho việc giao hàng, nhập hàng
 - Quản trị viên có thể quản lý page đa dạng hóa hệ thống làm website có nhiều tính năng cũng như nghiệp vụ mới mẻ phù hợp hệ thống
 - Quản trị viên có thể quản lý quảng cáo phục vụ cho việc đầu tư mang lại lợi nhuận cũng vốn hóa duy trì và phát triển hệ thống

2.3 Mô hình hóa yêu cầu

2.3.1 Xác định Actor

Người sử dụng hệ thống:

- Khách hàng (người mua hàng)
- Admin (người quản trị hệ thống)

2.3.2 Xác định Use case

+ Use case cho khách hàng:

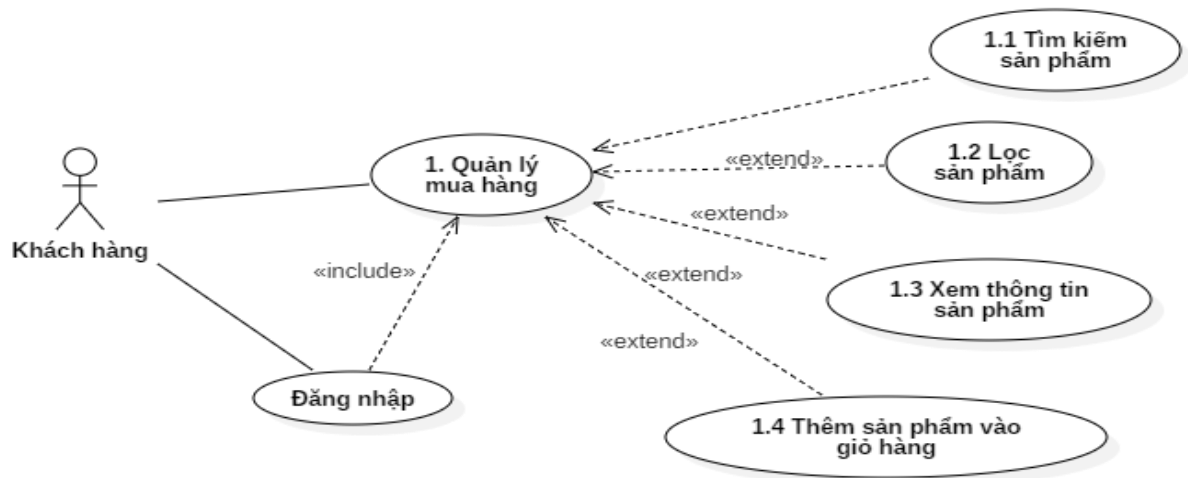
- Quản lý thông tin tài khoản: Cập nhật thông tin cá nhân, thay đổi mật khẩu
- Quản lý đơn hàng: Xem đơn hàng, hủy đơn hàng
- Quản lý giỏ hàng: Thêm sản phẩm vào giỏ hàng, cập nhật giỏ hàng, đặt hàng
- Quản lý mua hàng: Tìm kiếm, lọc sản phẩm, xem thông tin chi tiết sản phẩm

+ Use case cho Admin:

- Quản lý sản phẩm: Thêm, sửa, xóa, xem thông tin, tìm kiếm, sắp xếp sản phẩm
- Quản lý loại sản phẩm: Thêm, sửa, xóa, xem thông tin loại sản phẩm
- Quản lý đơn hàng: Thêm, sửa, xóa, xem thông tin, cập nhật trạng thái đơn hàng
- Quản lý thông tin khách hàng: Thêm, sửa, xóa, xem thông tin, tìm kiếm khách hàng
- Quản lý thông tin nhà cung cấp: Thêm, sửa, xóa, xem thông tin nhà cung cấp
- Quản lý tài khoản truy cập hệ thống: Thêm, sửa, xóa, xem thông tin tài khoản truy cập hệ thống
- Quản lý quyền truy cập: Thêm, sửa, xóa, xem thông tin quyền truy cập
- Quản lý tỉnh/thành: Thêm, sửa, xóa, xem thông tin tỉnh/thành
- Quản lý quận/huyện: Thêm, sửa, xóa, xem thông tin quận/huyện
- Quản lý phường/xã: Thêm, sửa, xóa, xem thông tin phường/xã
- Quản lý tin tức: Thêm, sửa, xóa, xem thông tin tin tức
- Quản lý shipper: Thêm, sửa, xóa, xem thông tin, tìm kiếm shipper
- Quản lý page: Thêm, sửa, xóa, xem tin thông tin page
- Quản lý quảng cáo: Thêm, sửa, xóa, xem thông tin quảng cáo
- Quản lý thống kê: Thông kê đơn hàng chưa xác nhận, sản phẩm bán chạy, doanh thu qua từng kỳ, người dùng hệ thống, thống kê bán hàng, thống kê số đơn hàng

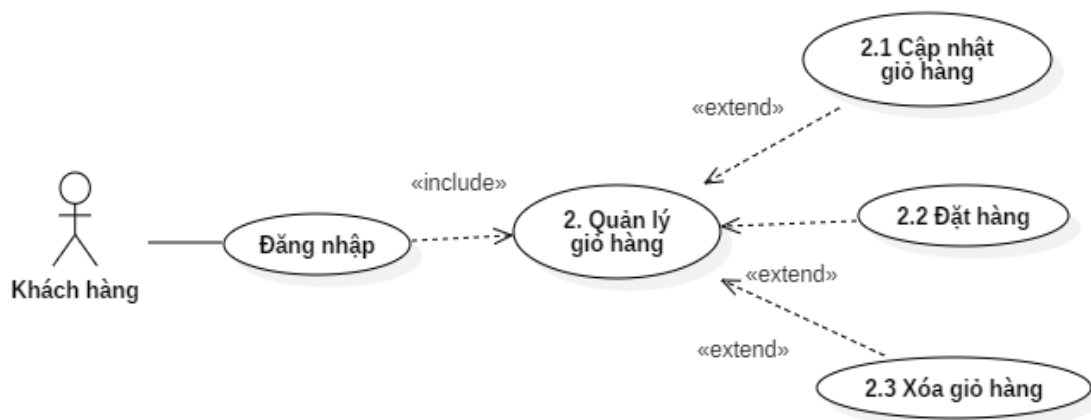
2.3.3.2 Use case khách hàng

❖ Quản lý mua hàng



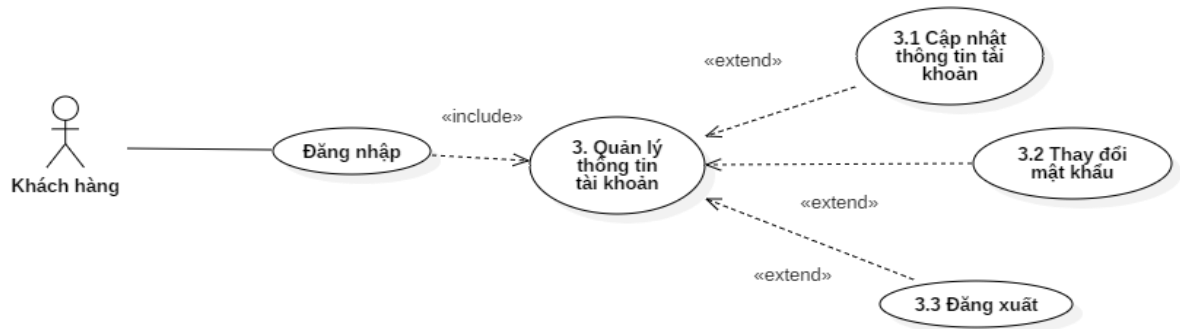
Hình 2.2 Use case quản lý mua hàng

❖ Quản lý giỏ hàng



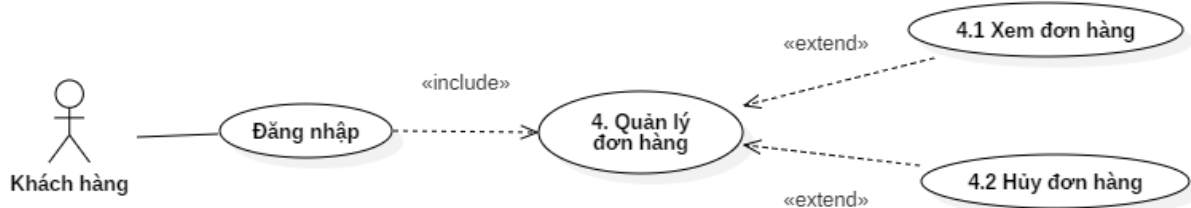
Hình 2.3 Use case quản lý giỏ hàng

❖ Quản lý thông tin tài khoản



Hình 2.4 Use case quản lý thông tin tài khoản

❖ Quản lý đơn hàng



Hình 2.5 Use case quản lý đơn hàng

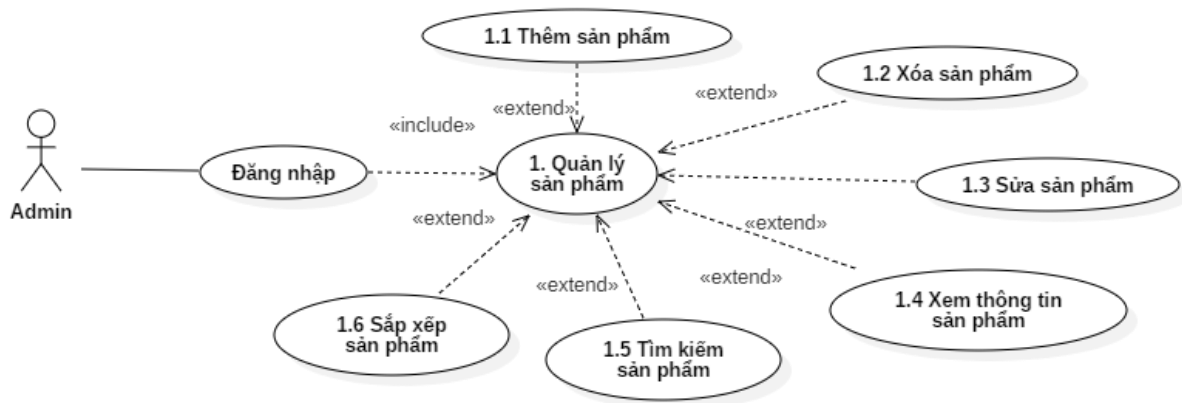
❖ Đăng ký tài khoản



Hình 2.6 Use case đăng ký tài khoản

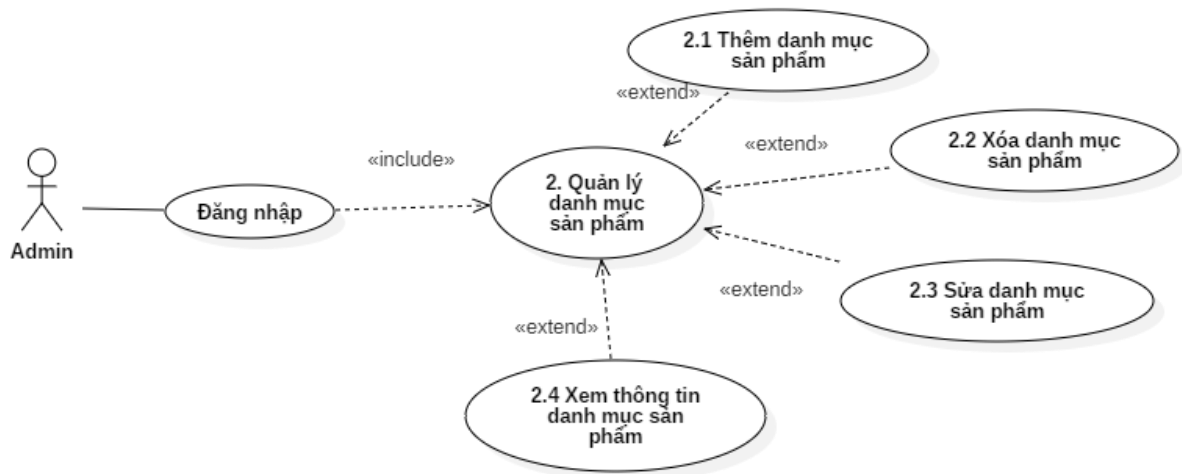
2.3.3.3 Use case của Admin

❖ Quản lý sản phẩm



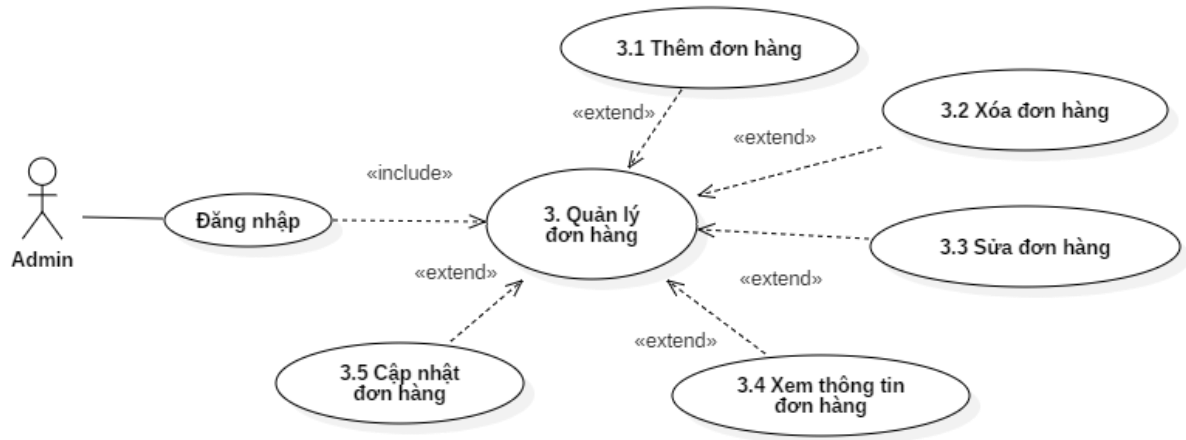
Hình 2.7 Use case quản lý sản phẩm

❖ Quản lý danh mục sản phẩm



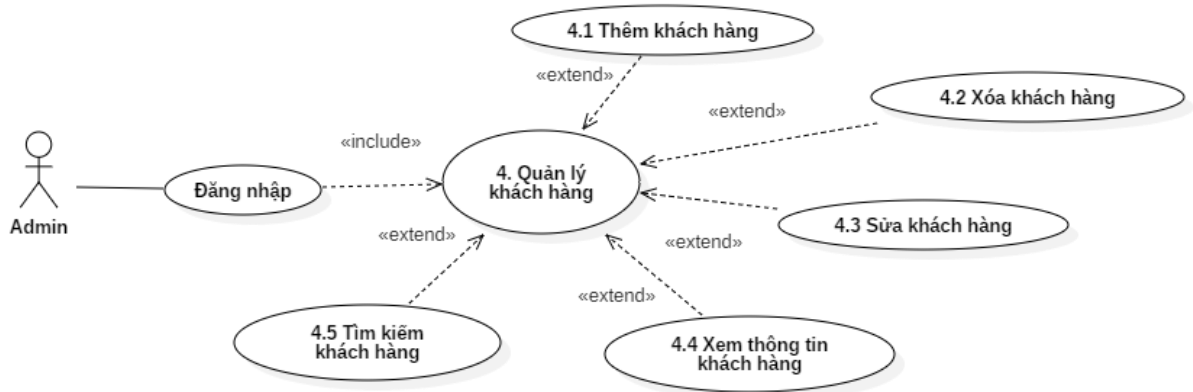
Hình 2.8 Use case quản lý danh mục sản phẩm

❖ Quản lý đơn hàng



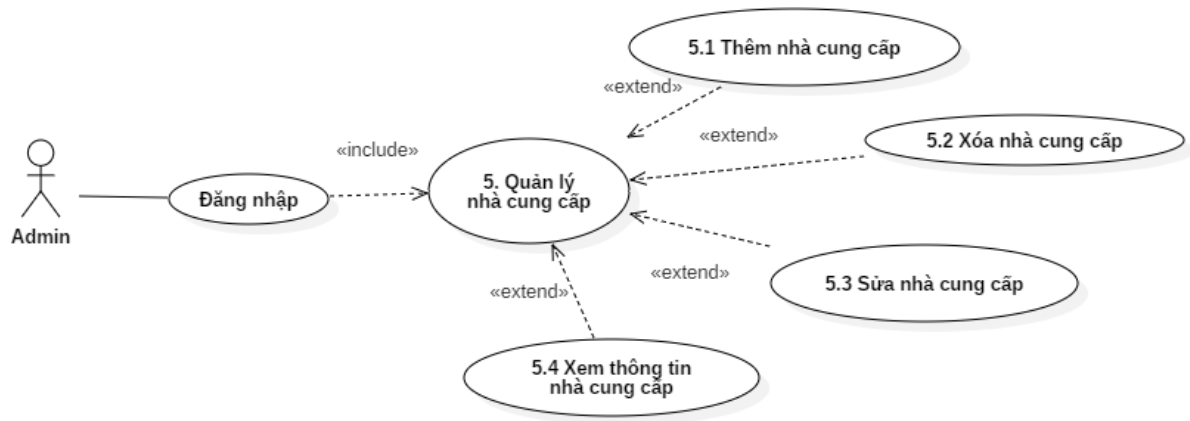
Hình 2.9 Use case quản lý đơn hàng

❖ Quản lý khách hàng



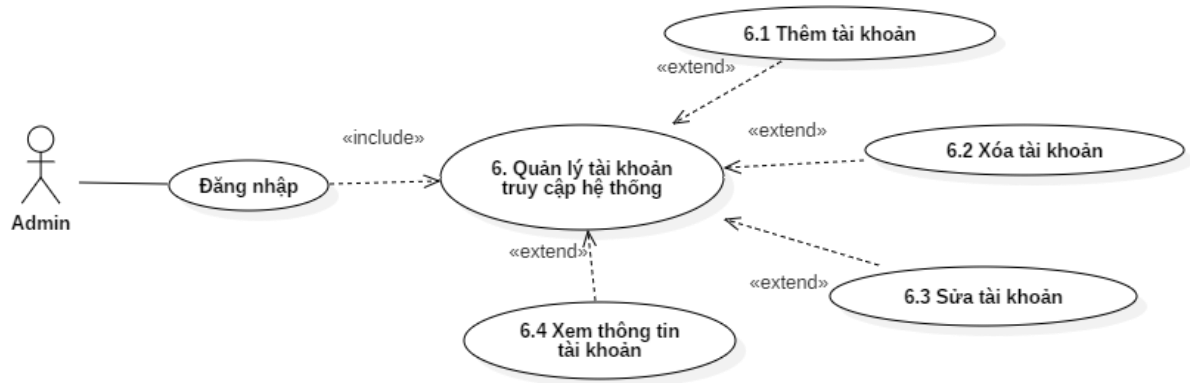
Hình 2.10 Use case quản lý khách hàng

❖ Quản lý nhà cung cấp



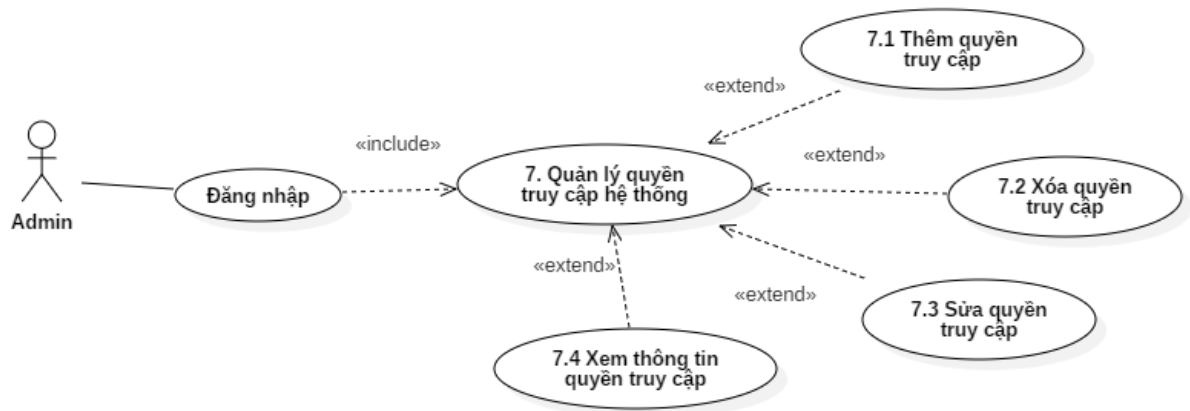
Hình 2.11 Use case quản lý nhà cung cấp

❖ Quản lý tài khoản truy cập hệ thống



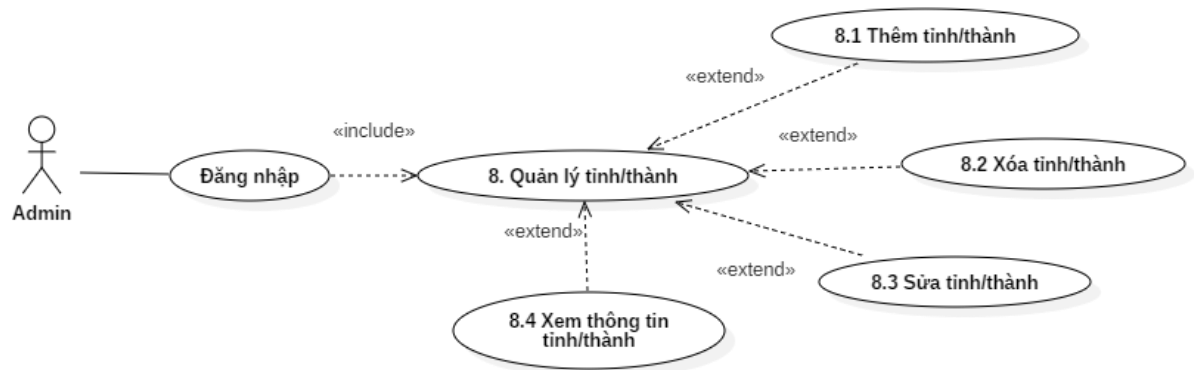
Hình 2.12 Use case quản lý tài khoản truy cập hệ thống

❖ Quản lý quyền truy cập hệ thống



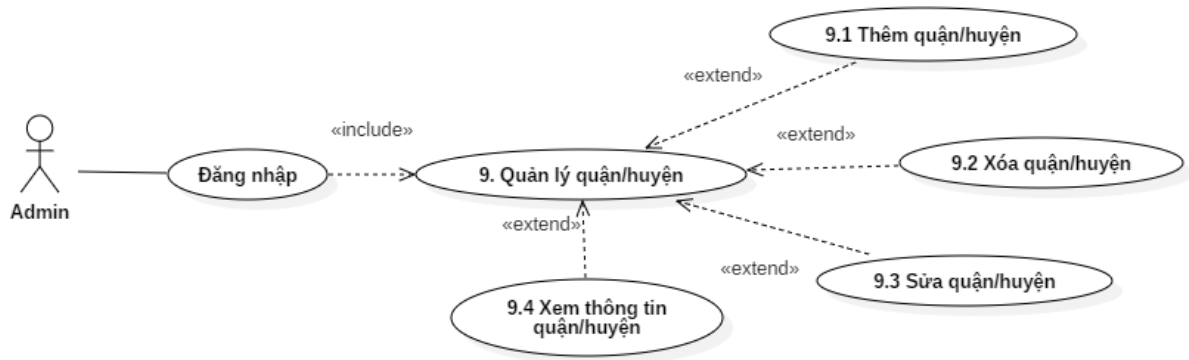
Hình 2.13 Use case quản lý quyền truy cập hệ thống

❖ Quản lý tỉnh/thành



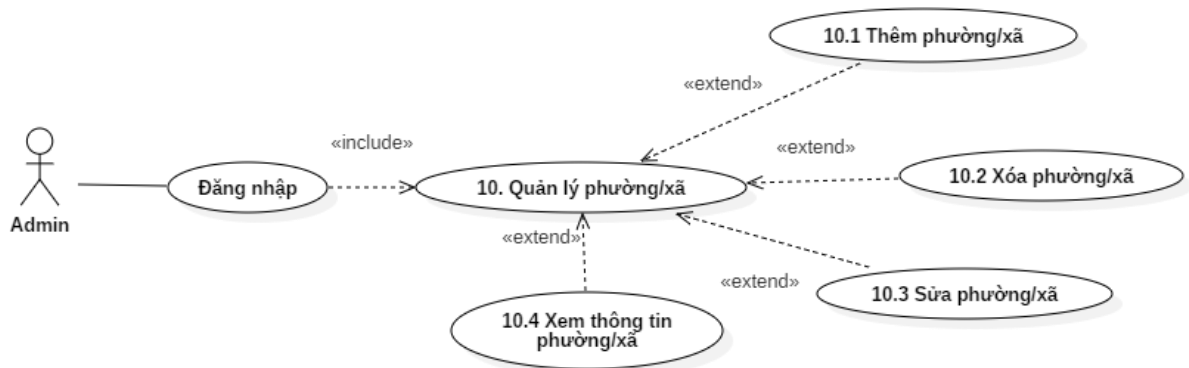
Hình 2.14 Use case quản lý tỉnh/thành

❖ Quản lý quận/huyện



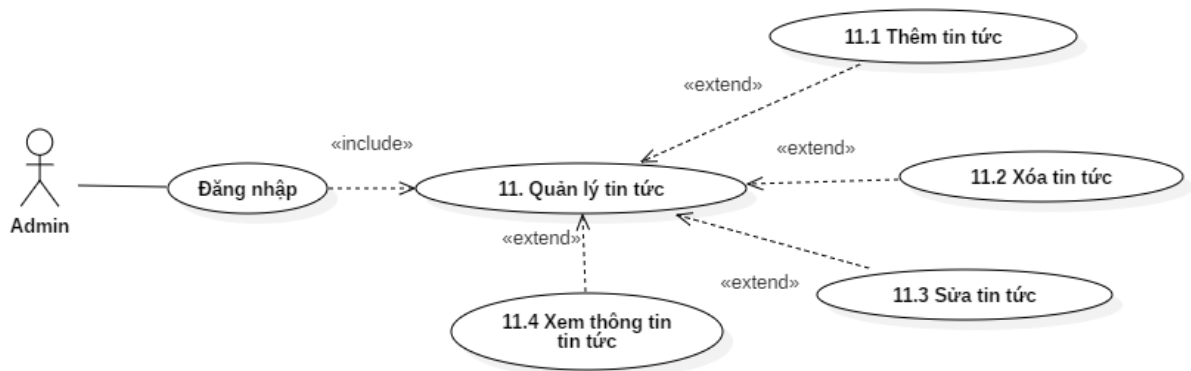
Hình 2.15 Use case quản lý quận/huyện

❖ Quản lý phường/xã



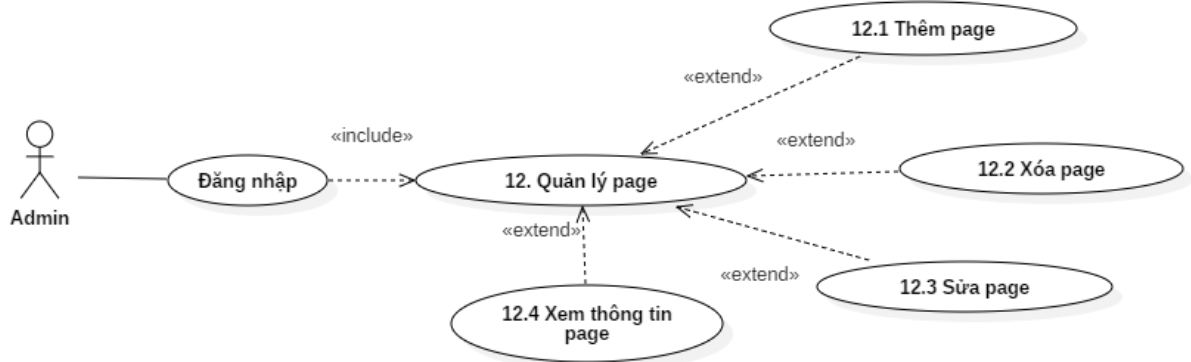
Hình 2.16 Use case quản lý phường/xã

❖ Quản lý tin tức



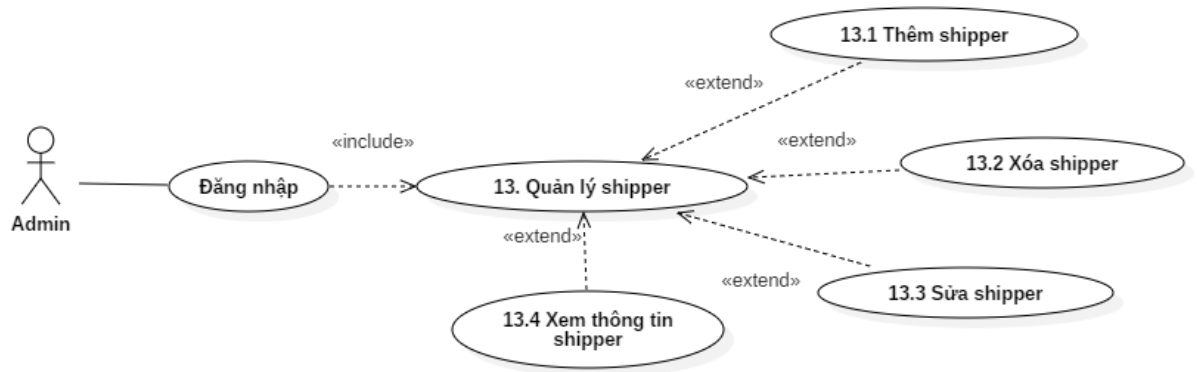
Hình 2.17 Use case quản lý tin tức

❖ Quản lý page



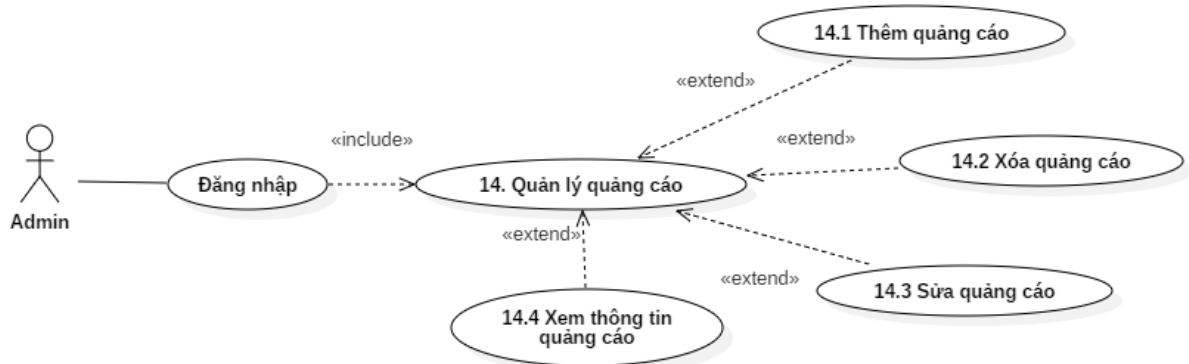
Hình 2.18 Use case quản lý page

❖ Quản lý shipper



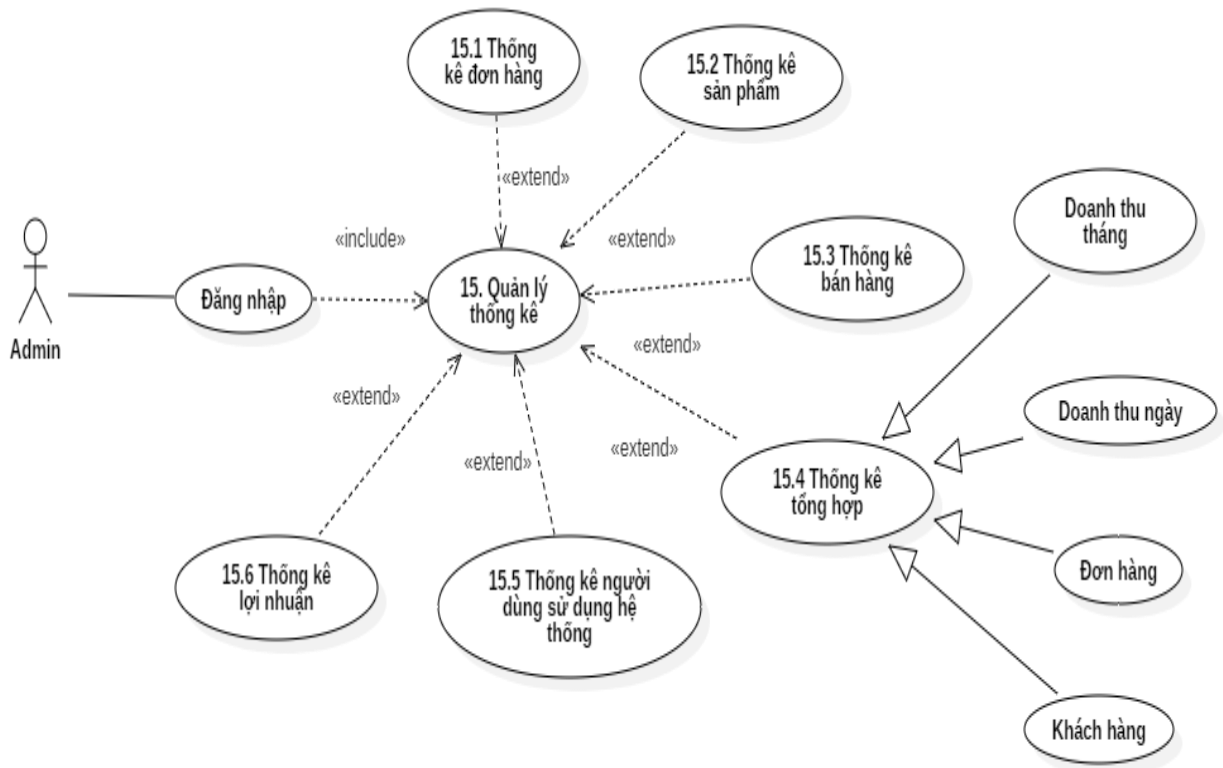
Hình 2.19 Use case quản lý shipper

❖ Quản lý quảng cáo



Hình 2.20 Use case quản lý quảng cáo

❖ Quản lý thống kê



Hình 2.21 Use case quản lý thống kê

2.3.4 Mô tả Use case

STT	Use case	Actor	Mục đích
	Đăng ký	Khách hàng	Khách hàng chưa có thông tin tại hệ thống thì phải đăng ký tài khoản để đăng nhập
	Đăng nhập	Khách hàng	Khách hàng đã có thông tin tài khoản dùng để đăng nhập vào hệ thống
1	Quản lý mua hàng	Khách hàng	Khách hàng sử dụng các chức năng liên quan đến việc mua hàng
1.1	Tìm kiếm sản phẩm	Khách hàng	Khách hàng nhập thông tin sản phẩm cần tìm kiếm chọn chức năng tìm kiếm hệ thống sẽ sản phẩm nếu tìm thấy không thì ngược lại
1.2	Lọc sản phẩm	Khách hàng	Khách hàng sắp xếp các sản phẩm đang hiển thị theo giá giảm tăng dần, khuyến mãi...

1.3	Xem thông tin sản phẩm	Khách hàng	Khách hàng click vào sản phẩm trả về thông tin chi tiết của sản phẩm
1.4	Thêm sản phẩm vào giỏ hàng	Khách hàng	Khách hàng thêm sản phẩm vào giỏ hàng
2	Quản lý giỏ hàng	Khách hàng	Khách hàng sử dụng các chức năng liên quan đến giỏ hàng
2.1	Cập nhật giỏ hàng	Khách hàng	Khách hàng cập nhật số lượng sản phẩm tại giỏ hàng
2.2	Thanh toán	Khách hàng	Khách hàng click thanh toán sau đó tiến hành nhập thông tin địa chỉ nhận hàng để muốn đặt hàng
2.3	Xóa giỏ hàng	Khách hàng	Khách hàng muốn xóa một hoặc tất cả sản phẩm tại giỏ hàng
3	Quản lý thông tin tài khoản	Khách hàng	Khách hàng sử dụng các chức năng liên quan quản lý tài khoản
3.1	Cập nhật thông tin tài khoản	Khách hàng	Khách hàng thay đổi thông tin cá nhân của mình
3.2	Thay đổi mật khẩu	Khách hàng	Khách hàng thay đổi mật khẩu hiện tại
3.3	Đăng xuất	Khách hàng	Khách hàng click đăng xuất thoát khỏi hệ thống
4	Quản lý đơn hàng	Khách hàng	Khách hàng sử dụng các chức năng liên quan quản lý đơn hàng
4.1	Xem đơn hàng	Khách hàng	Khách hàng xem lịch sử đơn đặt hàng
4.2	Hủy đơn hàng	Khách hàng	Khách hàng có thể hủy đơn nếu đơn đó chưa được vận chuyển
	Đăng ký tài khoản	Admin	Admin chưa có thông tin tại hệ thống thì phải đăng ký tài khoản để đăng nhập
	Đăng nhập	Admin	Admin đã có thông tin tài khoản dùng để đăng nhập vào hệ thống và sử dụng theo quyền truy cập của mình
1	Quản lý sản phẩm	Admin	Admin sử dụng các chức năng liên quan quản lý sản phẩm
1.1	Thêm sản phẩm	Admin	Admin thêm thông tin sản phẩm vào hệ thống
1.2	Xóa sản phẩm	Admin	Admin xóa thông tin sản phẩm khỏi hệ thống
1.3	Sửa sản phẩm	Admin	Admin cập nhật thông tin sản phẩm
1.4	Xem thông tin sản phẩm	Admin	Admin xem thông tin chi tiết của sản phẩm

1.5	Tìm kiếm sản phẩm	Admin	Admin nhập tên sản phẩm hệ thống trả về kết quả tìm kiếm
1.6	Sắp xếp sản phẩm	Admin	Admin sắp xếp danh sách sản phẩm thuộc loại sản phẩm
2	Quản lý danh mục sản phẩm	Admin	Admin sử dụng các chức năng liên quan quản lý danh mục sản phẩm
2.1	Thêm danh mục sản phẩm	Admin	Admin thêm thông tin danh mục sản phẩm vào hệ thống
2.2	Xóa danh mục sản phẩm	Admin	Admin xóa thông tin danh mục sản phẩm khỏi hệ thống
2.3	Sửa danh mục sản phẩm	Admin	Admin cập nhật thông tin danh mục sản phẩm
2.4	Xem thông tin danh mục sản phẩm	Admin	Admin xem thông tin chi tiết của danh mục sản phẩm
3	Quản lý đơn hàng	Admin	Admin sử dụng các chức năng liên quan quản lý đơn hàng
3.1	Thêm đơn hàng	Admin	Admin thêm thông tin đơn hàng vào hệ thống
3.2	Xóa đơn hàng	Admin	Admin xóa thông tin đơn hàng khỏi hệ thống
3.3	Sửa đơn hàng	Admin	Admin cập nhật thông tin đơn hàng
3.4	Xem thông tin đơn hàng	Admin	Admin xem thông tin chi tiết của đơn hàng
3.5	Cập nhật đơn hàng	Admin	Admin thay đổi trạng thái của đơn hàng
4	Quản lý khách hàng	Admin	Admin sử dụng các chức năng liên quan quản lý khách hàng
4.1	Thêm khách hàng	Admin	Admin thêm thông tin khách hàng vào hệ thống
4.2	Xóa khách hàng	Admin	Admin xóa thông tin khách hàng khỏi hệ thống
4.3	Sửa khách hàng	Admin	Admin cập nhật thông tin khách hàng
4.4	Xem thông tin khách hàng	Admin	Admin xem thông tin chi tiết của khách hàng
4.5	Tìm kiếm khách hàng	Admin	Admin nhập tên khách hàng hệ thống trả về kết quả tìm kiếm
5	Quản lý nhà cung cấp	Admin	Admin sử dụng các chức năng liên quan quản lý nhà cung cấp

5.1	Thêm nhà cung cấp	Admin	Admin thêm thông tin nhà cung cấp vào hệ thống
5.2	Xóa nhà cung cấp	Admin	Admin xóa thông tin nhà cung cấp khỏi hệ thống
5.3	Sửa nhà cung cấp	Admin	Admin cập nhật thông tin nhà cung cấp
5.4	Xem thông tin nhà cung cấp	Admin	Admin xem thông tin chi tiết của nhà cung cấp
6	Quản lý tài khoản truy cập hệ thống	Admin	Admin sử dụng các chức năng liên quan quản lý tài khoản truy cập hệ thống
6.1	Thêm tài khoản truy cập hệ thống	Admin	Admin thêm thông tin tài khoản truy cập hệ thống vào hệ thống
6.2	Xóa tài khoản truy cập hệ thống	Admin	Admin xóa thông tin tài khoản truy cập hệ thống khỏi hệ thống
6.3	Sửa tài khoản truy cập hệ thống	Admin	Admin cập nhật thông tin tài khoản truy cập hệ thống
6.4	Xem thông tin tài khoản truy cập hệ thống	Admin	Admin xem thông tin chi tiết của tài khoản truy cập hệ thống
7	Quản lý quyền truy cập hệ thống	Admin	Admin sử dụng các chức năng liên quan quản lý quyền truy cập hệ thống
7.1	Thêm quyền truy cập hệ thống	Admin	Admin thêm thông tin quyền truy cập hệ thống vào hệ thống
7.2	Xóa quyền truy cập hệ thống	Admin	Admin xóa thông tin quyền truy cập hệ thống khỏi hệ thống
7.3	Sửa quyền truy cập hệ thống	Admin	Admin cập nhật thông tin quyền truy cập hệ thống
7.4	Xem thông tin tài khoản truy cập hệ thống	Admin	Admin xem thông tin chi tiết quyền truy cập hệ thống

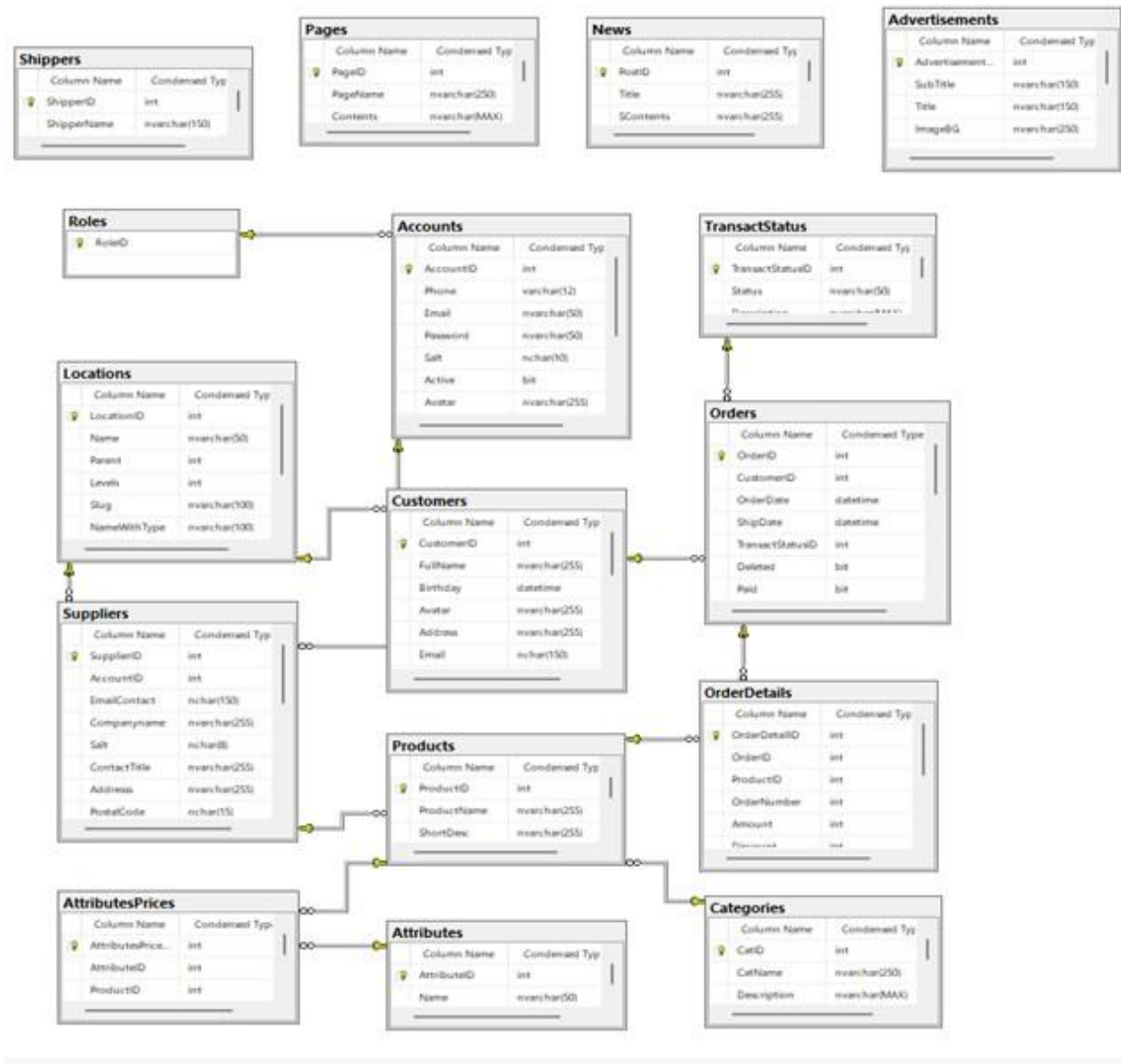
8	Quản lý tỉnh/thành	Admin	Admin sử dụng các chức năng liên quan quản lý tỉnh/thành
8.1	Thêm tỉnh/thành	Admin	Admin thêm thông tin tỉnh/thành hệ thống vào hệ thống
8.2	Xóa tỉnh/thành	Admin	Admin xóa thông tin tỉnh/thành hệ thống khỏi hệ thống
8.3	Sửa tỉnh/thành	Admin	Admin cập nhật thông tin tỉnh/thành
8.4	Xem thông tin tỉnh/thành	Admin	Admin xem thông tin chi tiết của tỉnh/thành
9	Quản lý quận/huyện	Admin	Admin sử dụng các chức năng liên quan quản lý quận/huyện
9.1	Thêm quận/huyện	Admin	Admin thêm thông tin quận/huyện hệ thống vào hệ thống
9.2	Xóa quận/huyện	Admin	Admin xóa thông tin quận/huyện hệ thống khỏi hệ thống
9.3	Sửa quận/huyện	Admin	Admin cập nhật thông tin quận/huyện
9.4	Xem thông tin quận/huyện	Admin	Admin xem thông tin chi tiết của quận/huyện
10	Quản lý phường/xã	Admin	Admin sử dụng các chức năng liên quan quản lý phường/xã
10.1	Thêm phường/xã	Admin	Admin thêm thông tin phường/xã hệ thống vào hệ thống
10.2	Xóa phường/xã	Admin	Admin xóa thông tin phường/xã hệ thống khỏi hệ thống
10.3	Sửa phường/xã	Admin	Admin cập nhật thông tin phường/xã
10.4	Xem thông tin phường/xã	Admin	Admin xem thông tin chi tiết của phường/xã
11	Quản lý tin tức	Admin	Admin sử dụng các chức năng liên quan quản lý tin tức
11.1	Thêm tin tức	Admin	Admin thêm thông tin tin tức hệ thống vào hệ thống
11.2	Xóa tin tức	Admin	Admin xóa thông tin tin tức hệ thống khỏi hệ thống
11.3	Sửa tin tức	Admin	Admin cập nhật thông tin tin tức
11.4	Xem thông tin tin tức	Admin	Admin xem thông tin chi tiết của tin tức

12	Quản lý page	Admin	Admin sử dụng các chức năng liên quan quản lý page
12.1	Thêm page	Admin	Admin thêm thông tin page hệ thống vào hệ thống
12.2	Xóa page	Admin	Admin xóa thông tin page hệ thống khỏi hệ thống
12.3	Sửa page	Admin	Admin cập nhật thông tin page
12.4	Xem thông tin page	Admin	Admin xem thông tin chi tiết của page
13	Quản lý shipper	Admin	Admin sử dụng các chức năng liên quan quản lý shipper
13.1	Thêm shipper	Admin	Admin thêm thông tin shipper hệ thống vào hệ thống
13.2	Xóa shipper	Admin	Admin xóa thông tin shipper hệ thống khỏi hệ thống
13.3	Sửa shipper	Admin	Admin cập nhật thông tin shipper
13.4	Xem thông tin shipper	Admin	Admin xem thông tin chi tiết của shipper
14	Quản lý quảng cáo	Admin	Admin sử dụng các chức năng liên quan quản lý quảng cáo
14.1	Thêm quảng cáo	Admin	Admin thêm thông tin quảng cáo hệ thống vào hệ thống
14.2	Xóa quảng cáo	Admin	Admin xóa thông tin quảng cáo hệ thống khỏi hệ thống
14.3	Sửa quảng cáo	Admin	Admin cập nhật thông tin quảng cáo
14.4	Xem thông tin quảng cáo	Admin	Admin xem thông tin chi tiết của quảng cáo
15	Quản lý thống kê	Admin	Admin sử dụng các chức năng liên quan quản lý thống kê
15.1	Thống kê đơn hàng	Admin	Admin thống kê danh sách đơn hàng chưa được xác nhận (đơn hàng mới đặt mới nhất)
15.2	Thống kê sản phẩm	Admin	Admin thống kê được top những sản phẩm hiển thị trang chủ
15.3	Thống kê bán hàng	Admin	Admin biết được số lượng người dùng online và offline qua từng tháng
15.4	Thống kê tổng hợp	Admin	Admin biết được tổng doanh thu tháng, ngày, số lượng đơn hàng đã bán, số lượng khách hàng sử dụng hệ thống

15.5	Thống kê người sử dụng hệ thống	Admin	Admin thống kê được số lượng nhà cung cấp, khách hàng, người quản trị viên
15.6	Thống kê lợi nhuận	Admin	Admin thống kê tổng tất các chi phí thuê, tiền đơn hàng, tiền doanh thu bán hàng và tiền lãi nhận được

2.4 Phân tích và thiết kế hệ thống

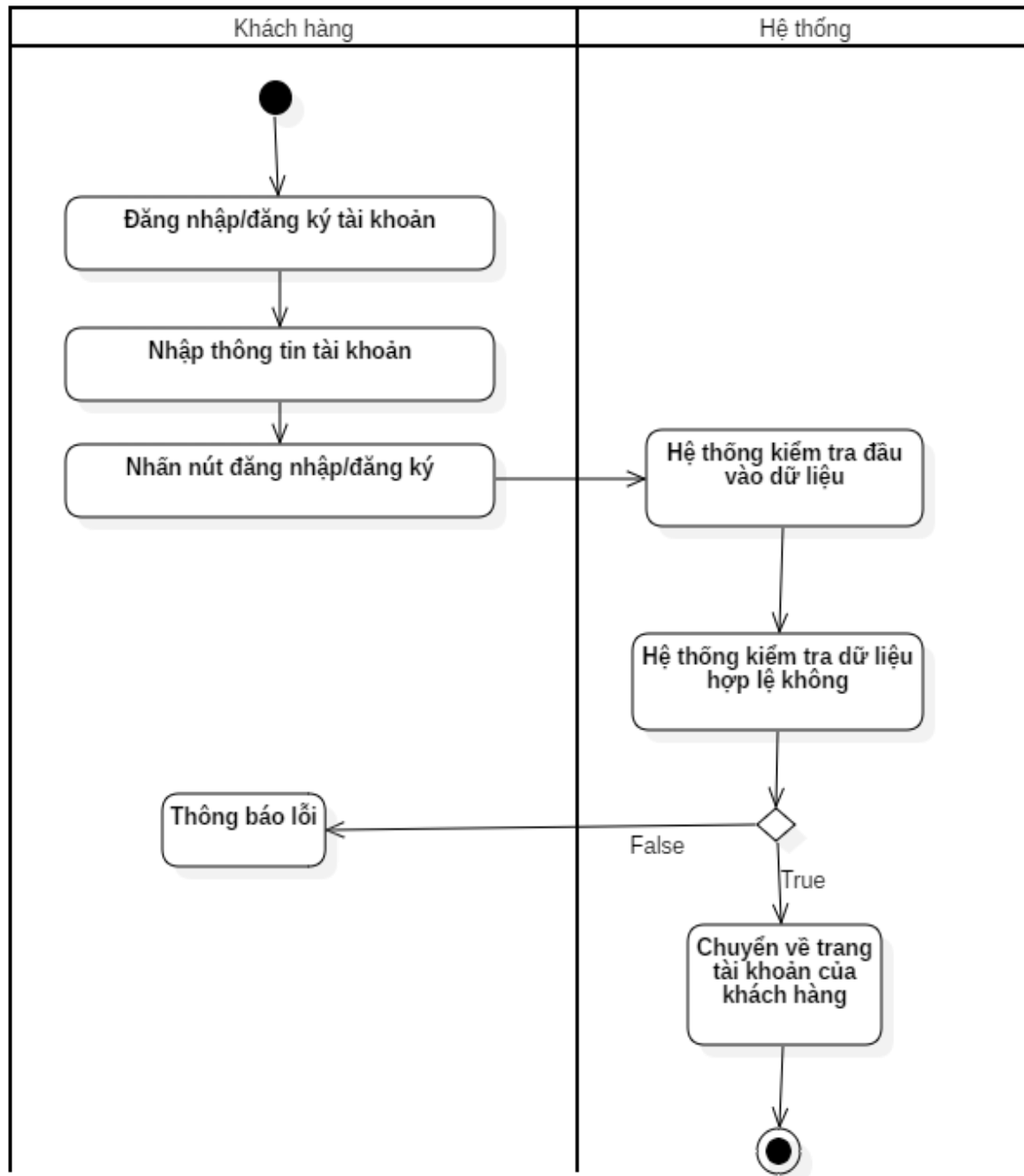
2.4.1 Thiết kế cơ sở dữ liệu



Hình 2.22 Cơ sở dữ liệu hệ thống

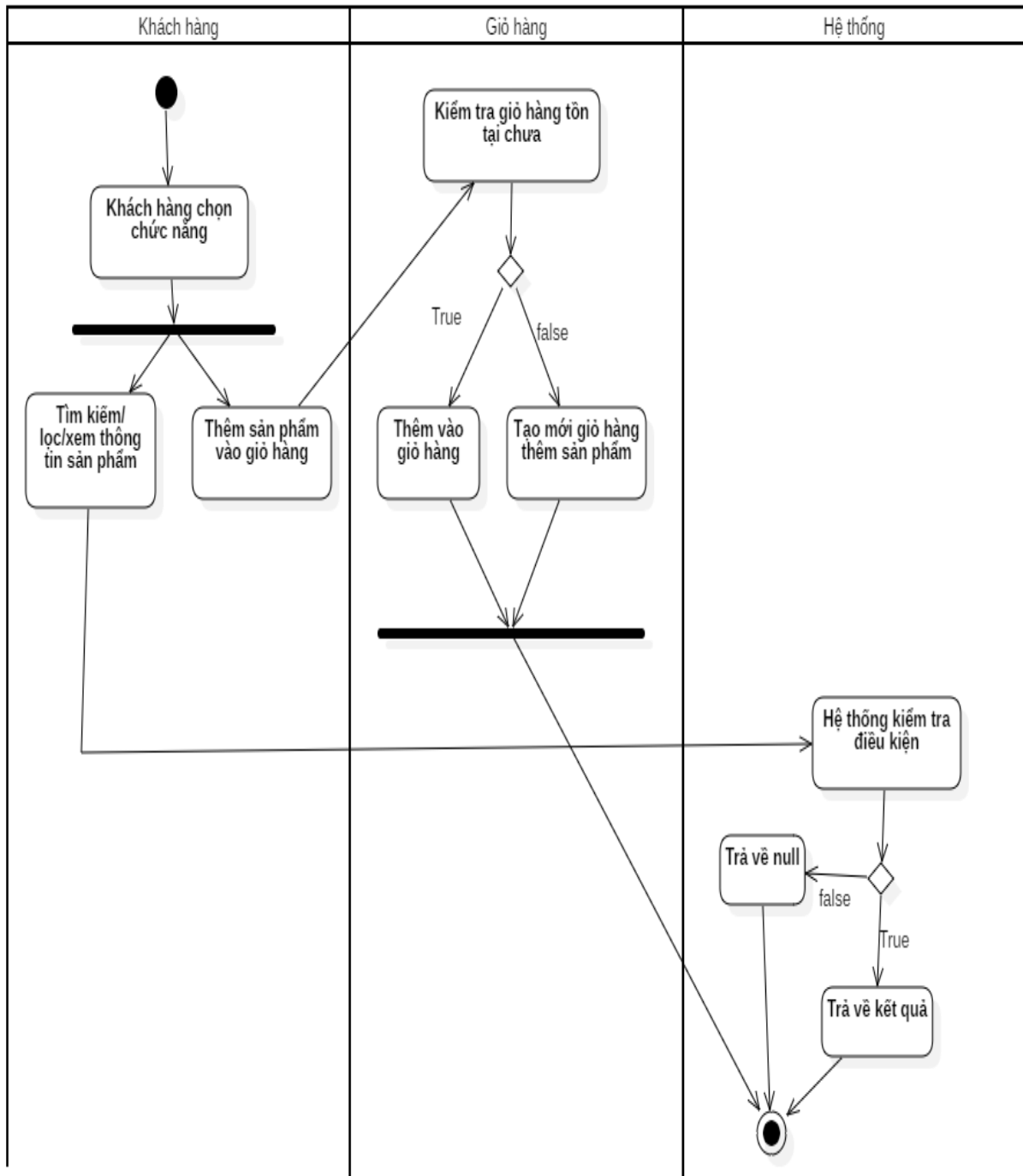
2.4.2 Sơ đồ hoạt động (Activity Diagram)

2.4.2.1 Đăng nhập và đăng ký



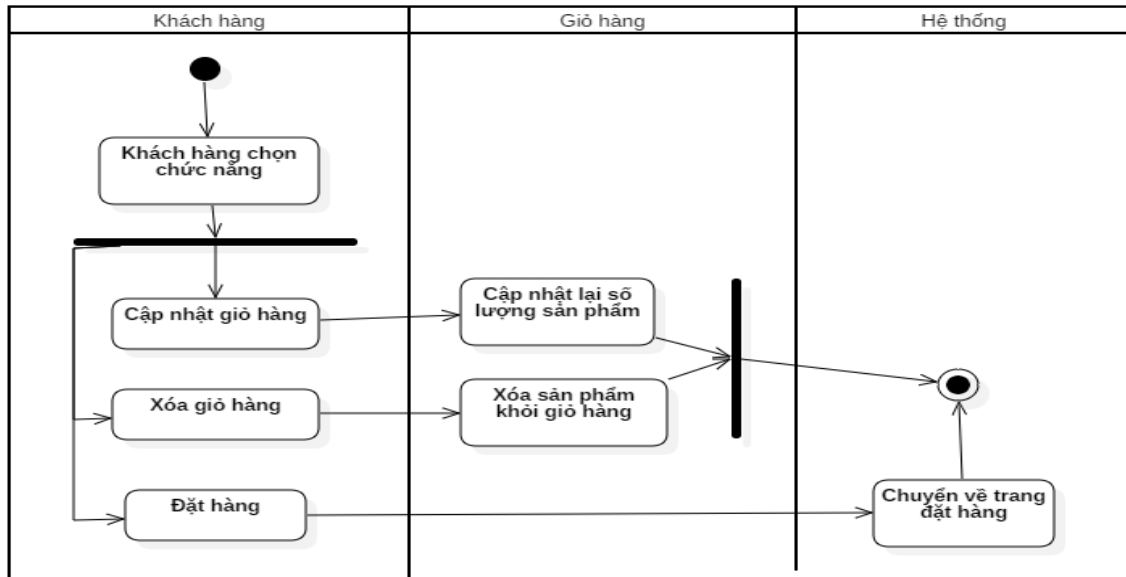
Hình 2.23 Sơ đồ hoạt động đăng nhập và đăng ký

2.4.2.2 Quản lý mua hàng



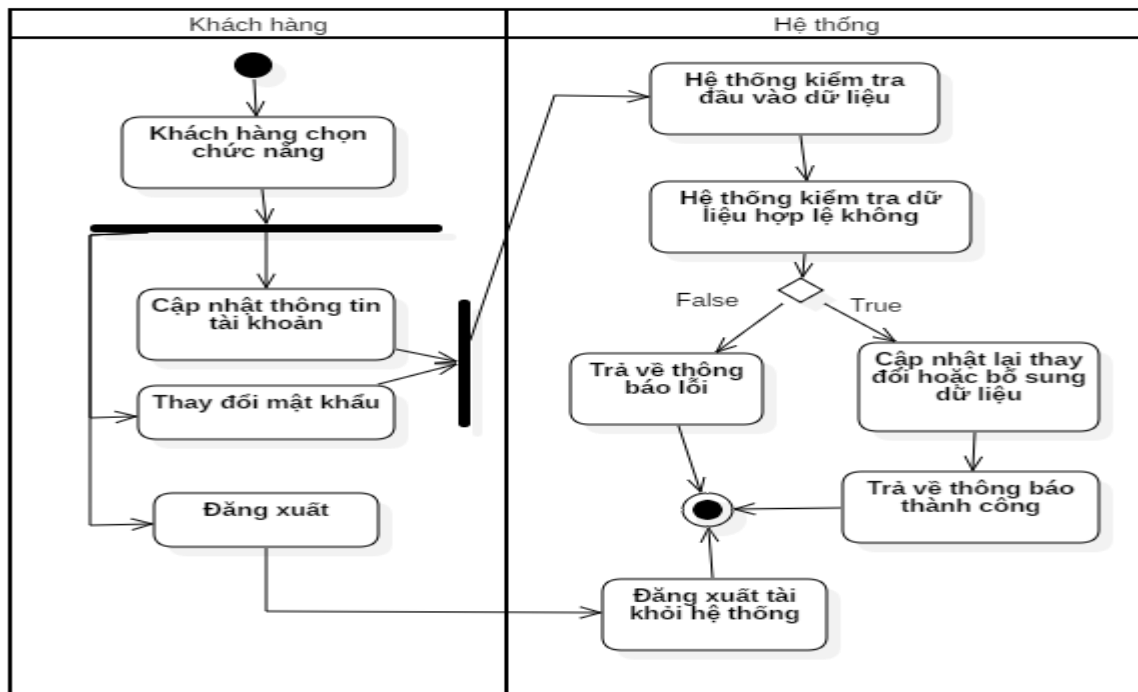
Hình 2.24 Sơ đồ hoạt động quản lý mua hàng

2.4.2.3 Quản lý giỏ hàng



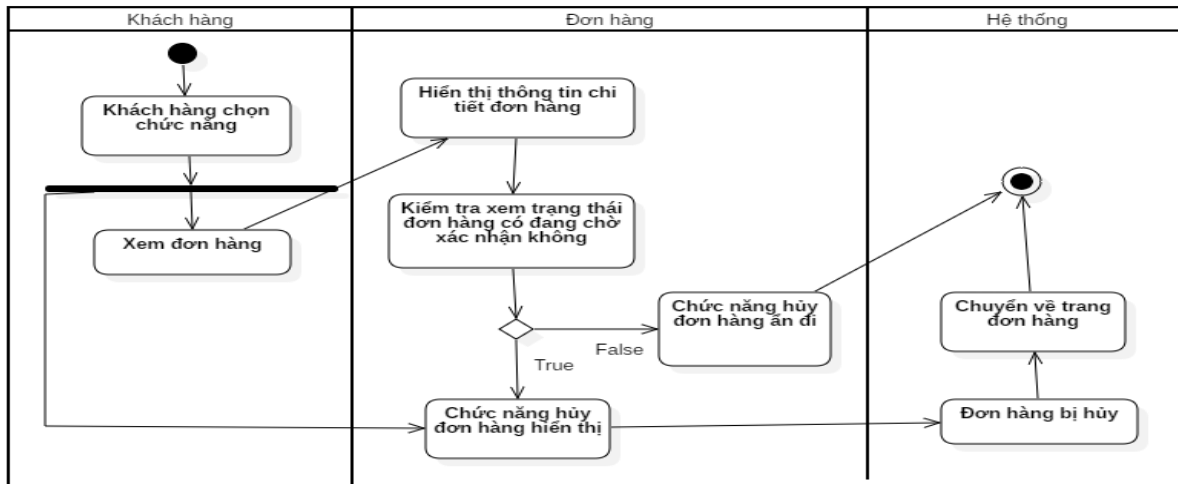
Hình 2.25 Sơ đồ hoạt động quản lý giỏ hàng

2.4.2.4 Quản lý tài khoản



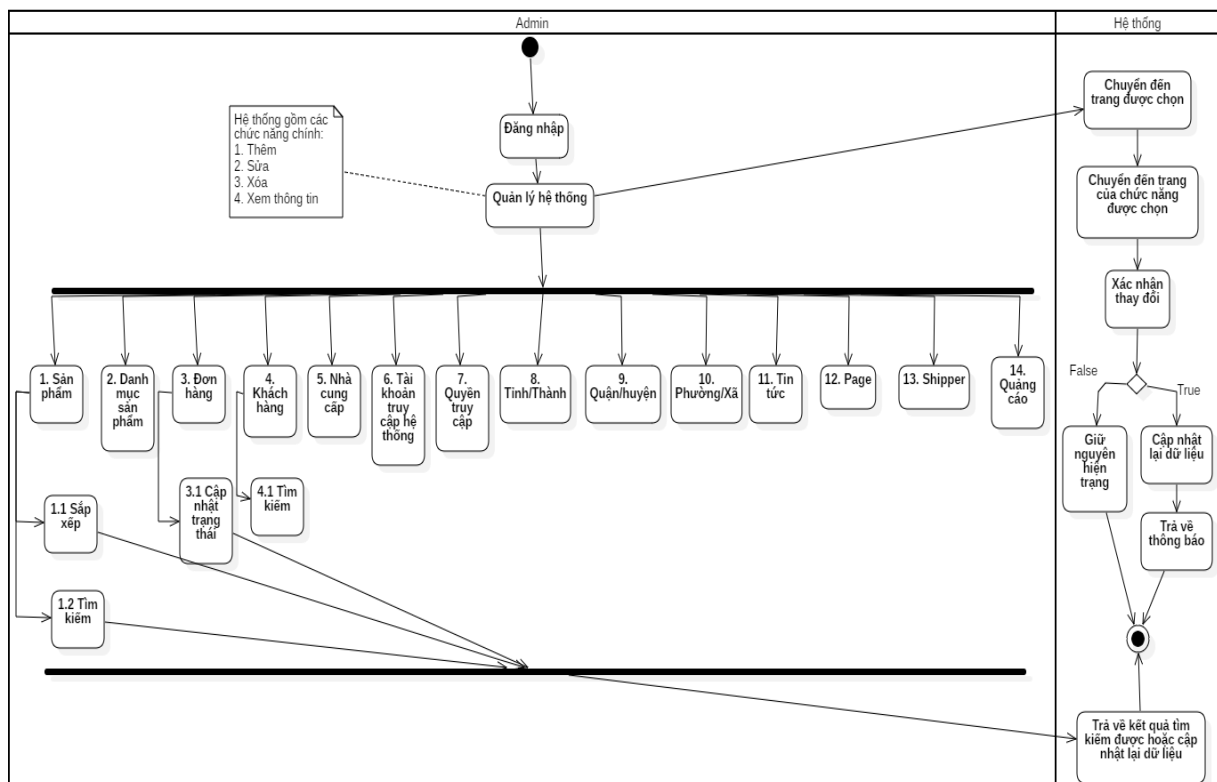
Hình 2.26 Sơ đồ hoạt động quản lý tài khoản

2.4.2.5 Quản lý đơn hàng



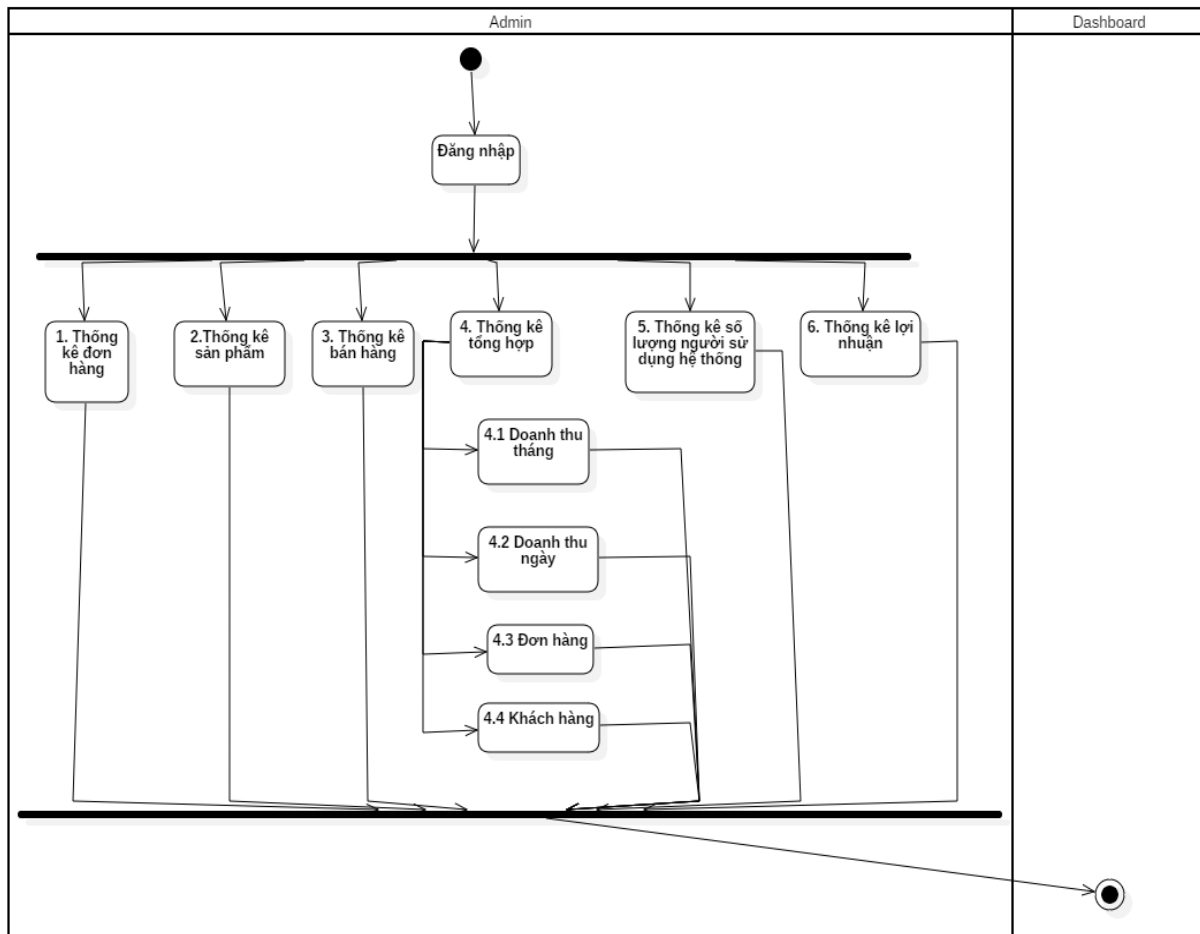
Hình 2.27 Sơ hoạt động quản lý đơn hàng

2.4.2.6 Quản lý hệ thống



Hình 2.28 Sơ đồ hoạt động quản lý hệ thống

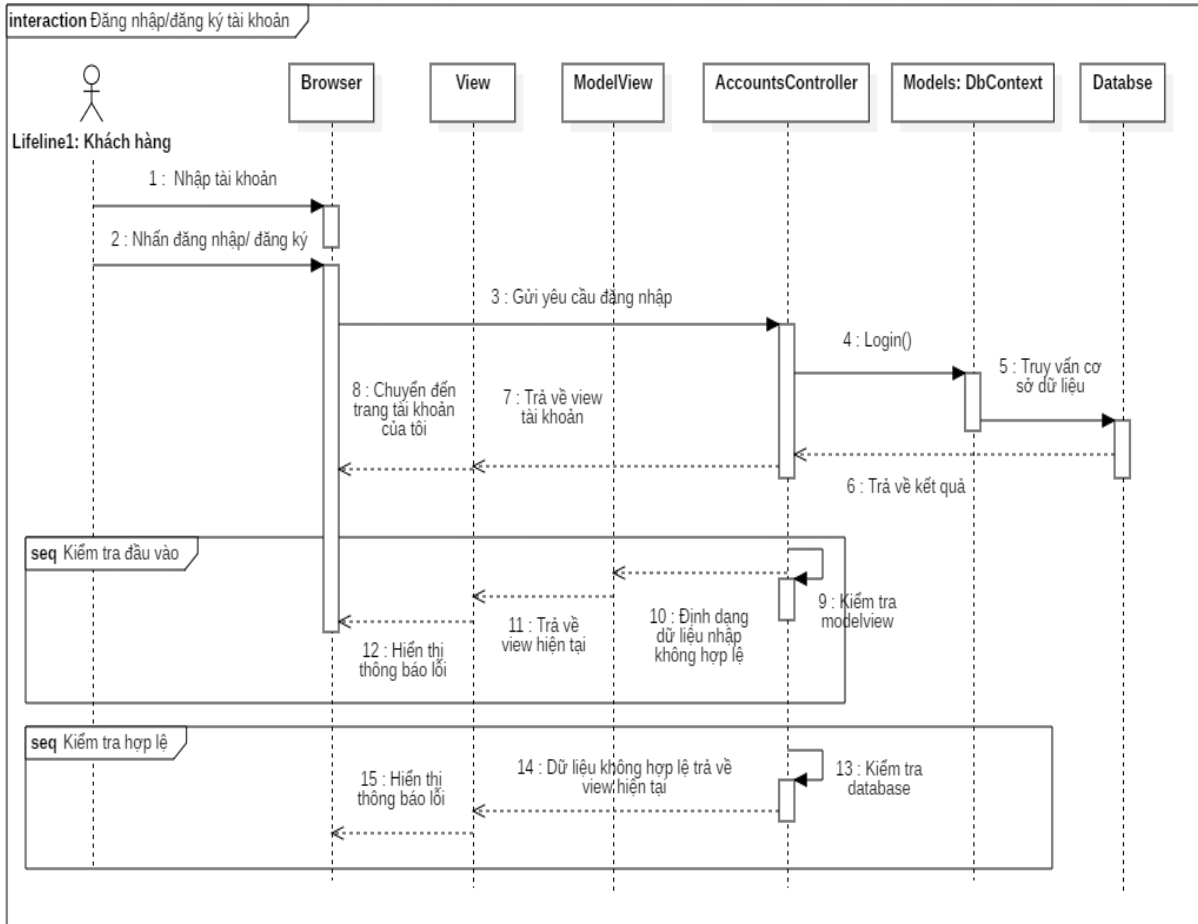
2.4.2.7 Quản lý thống kê



Hình 2.29 Sơ hoạt động quản lý thống kê

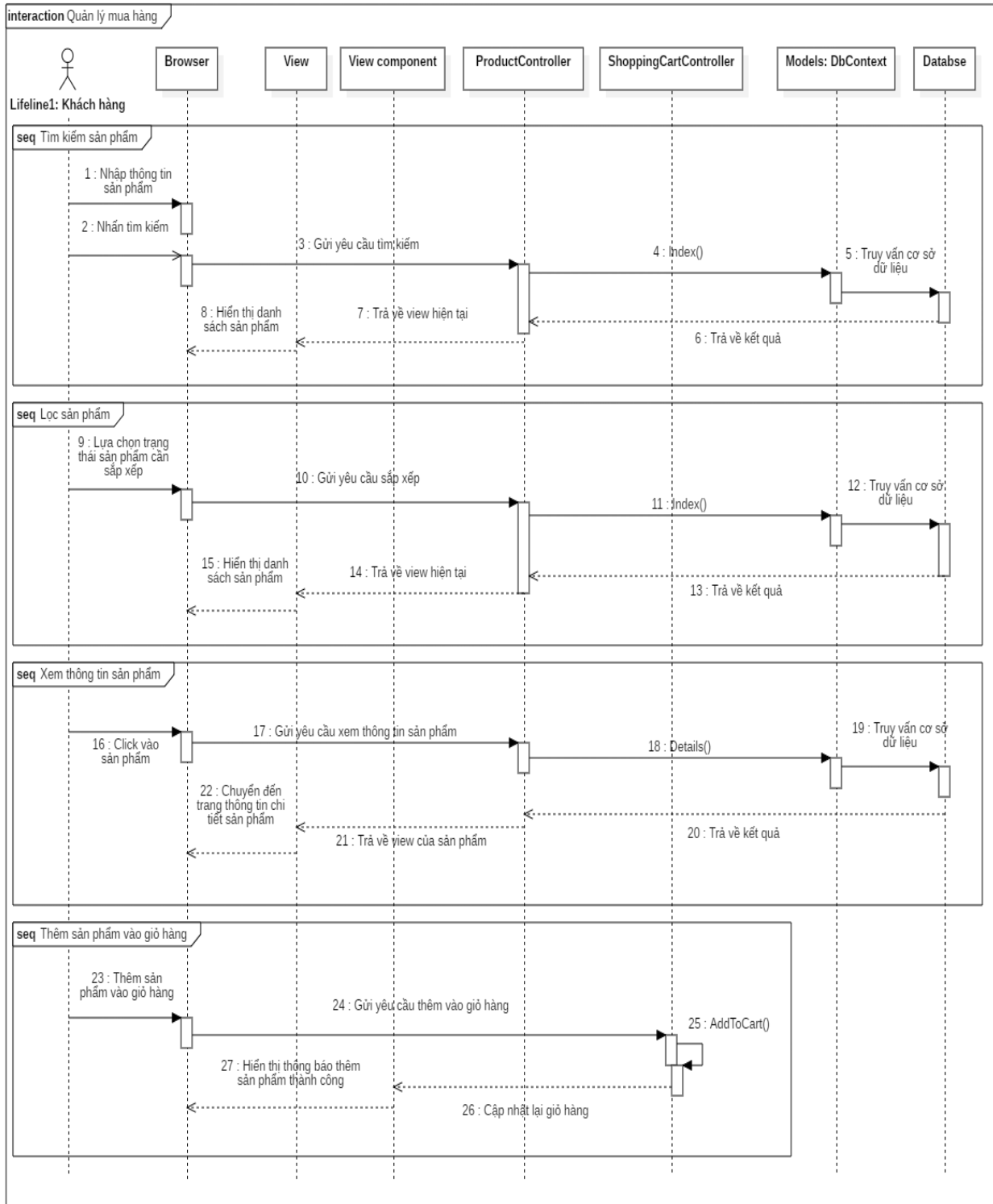
2.4.3 Sơ đồ tuần tự (Sequence Diagram)

2.4.3.1 Đăng nhập và đăng ký tài khoản



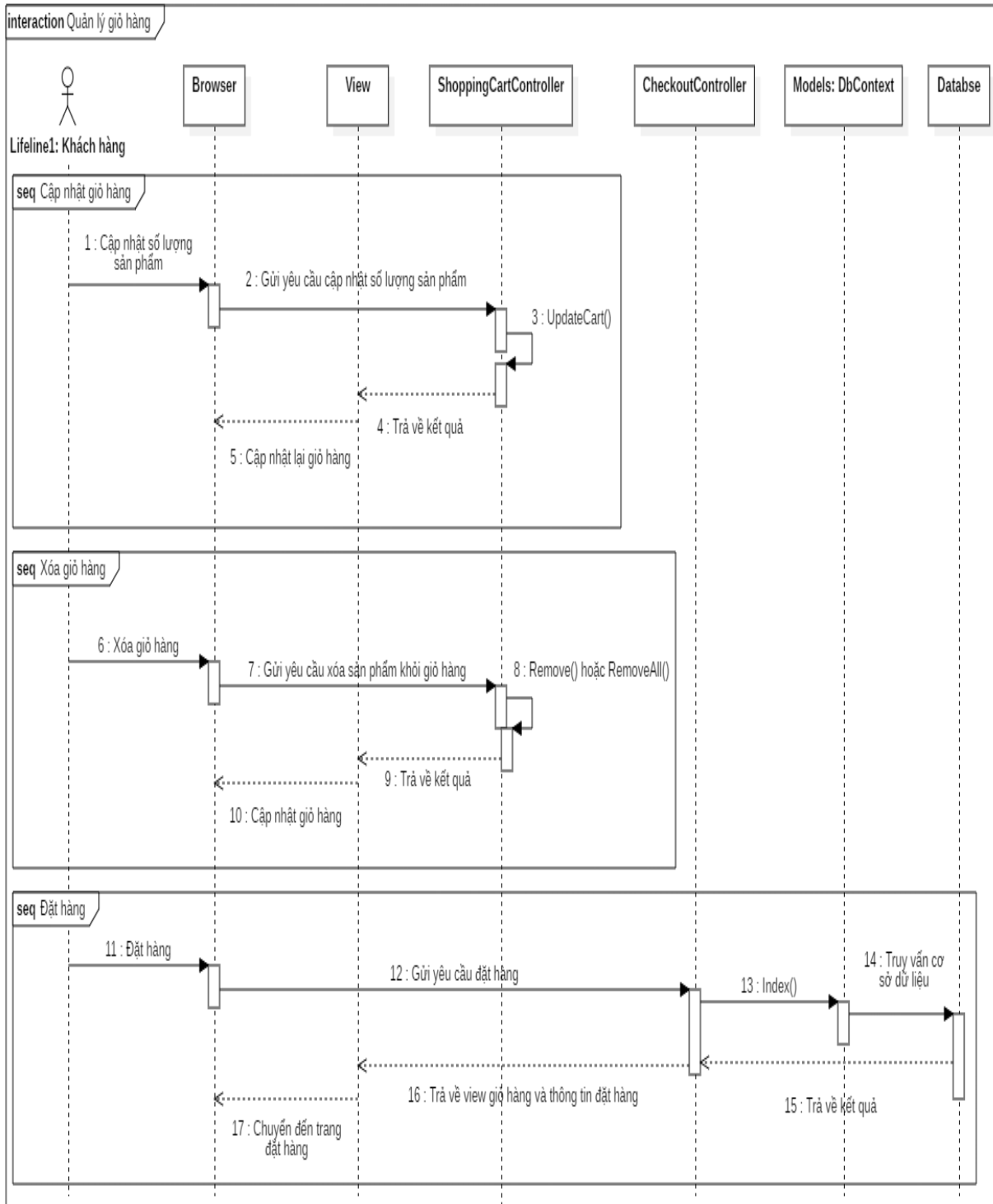
Hình 2.30 Sơ đồ tuần tự đăng nhập và đăng ký

2.4.3.2 Quản lý mua hàng



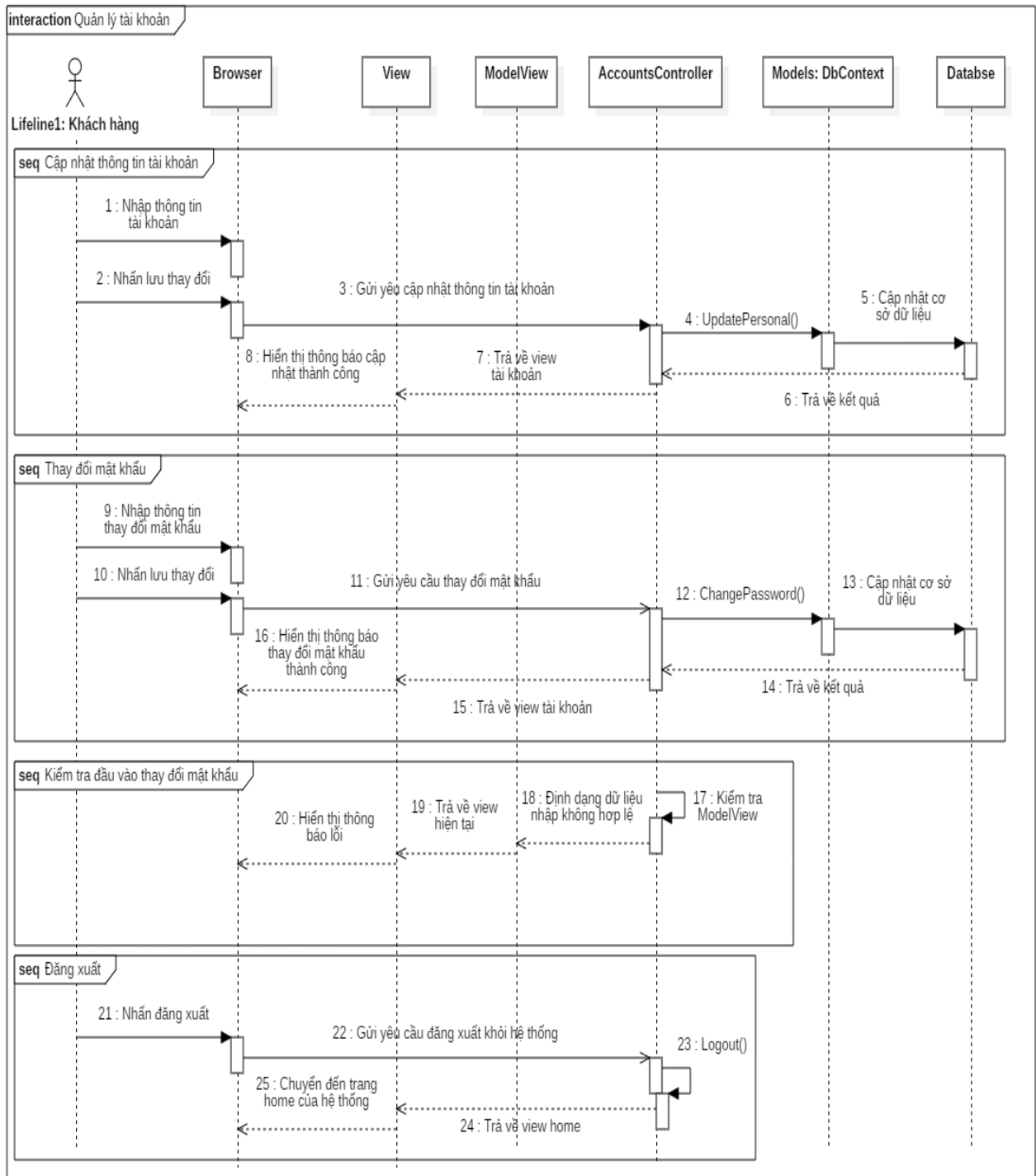
Hình 2.31 Sơ đồ tuần tự quản lý mua hàng

2.4.3.3 Quản lý giỏ hàng



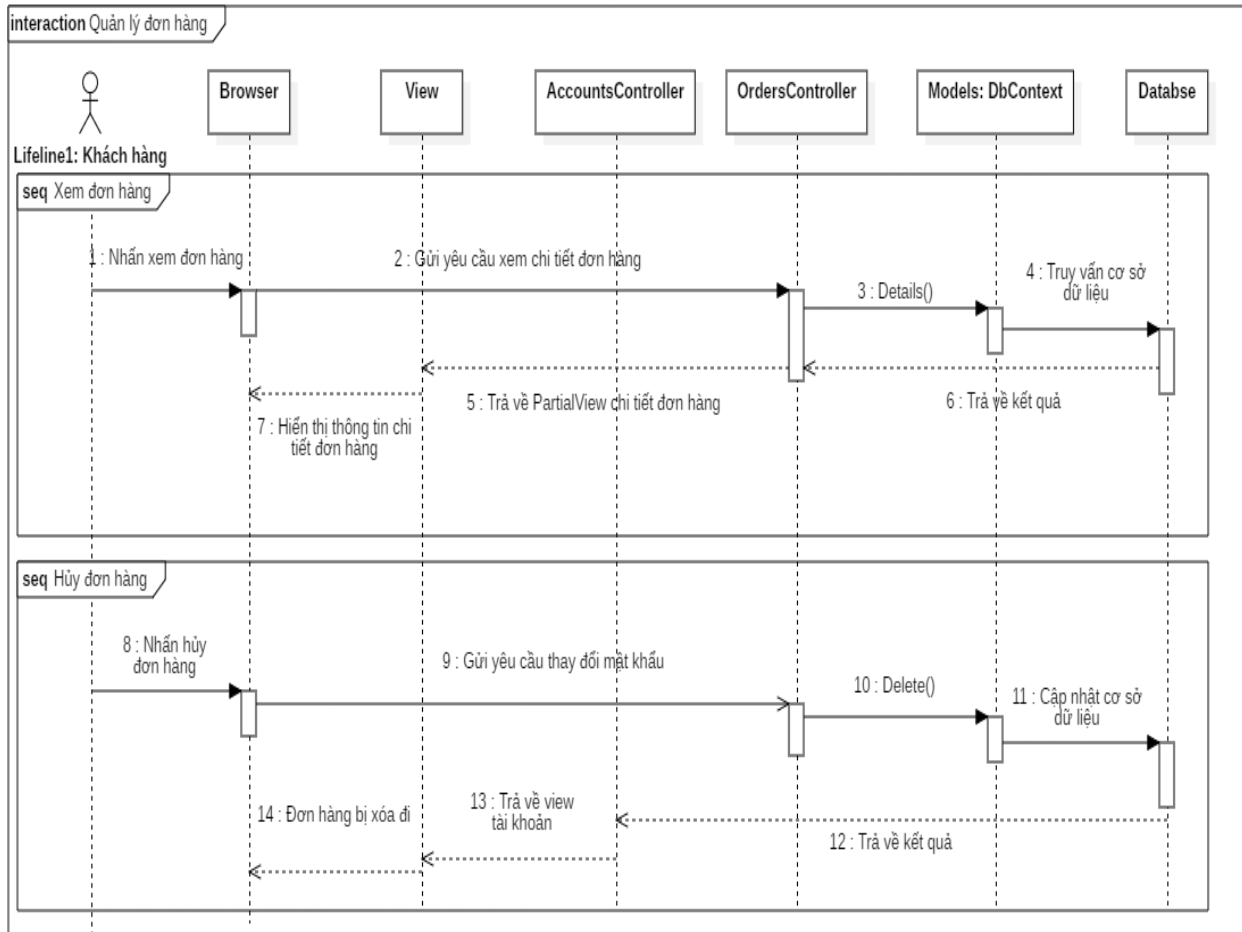
Hình 2.32 Sơ đồ tuần tự quản lý giỏ hàng

2.4.3.4 Quản lý tài khoản



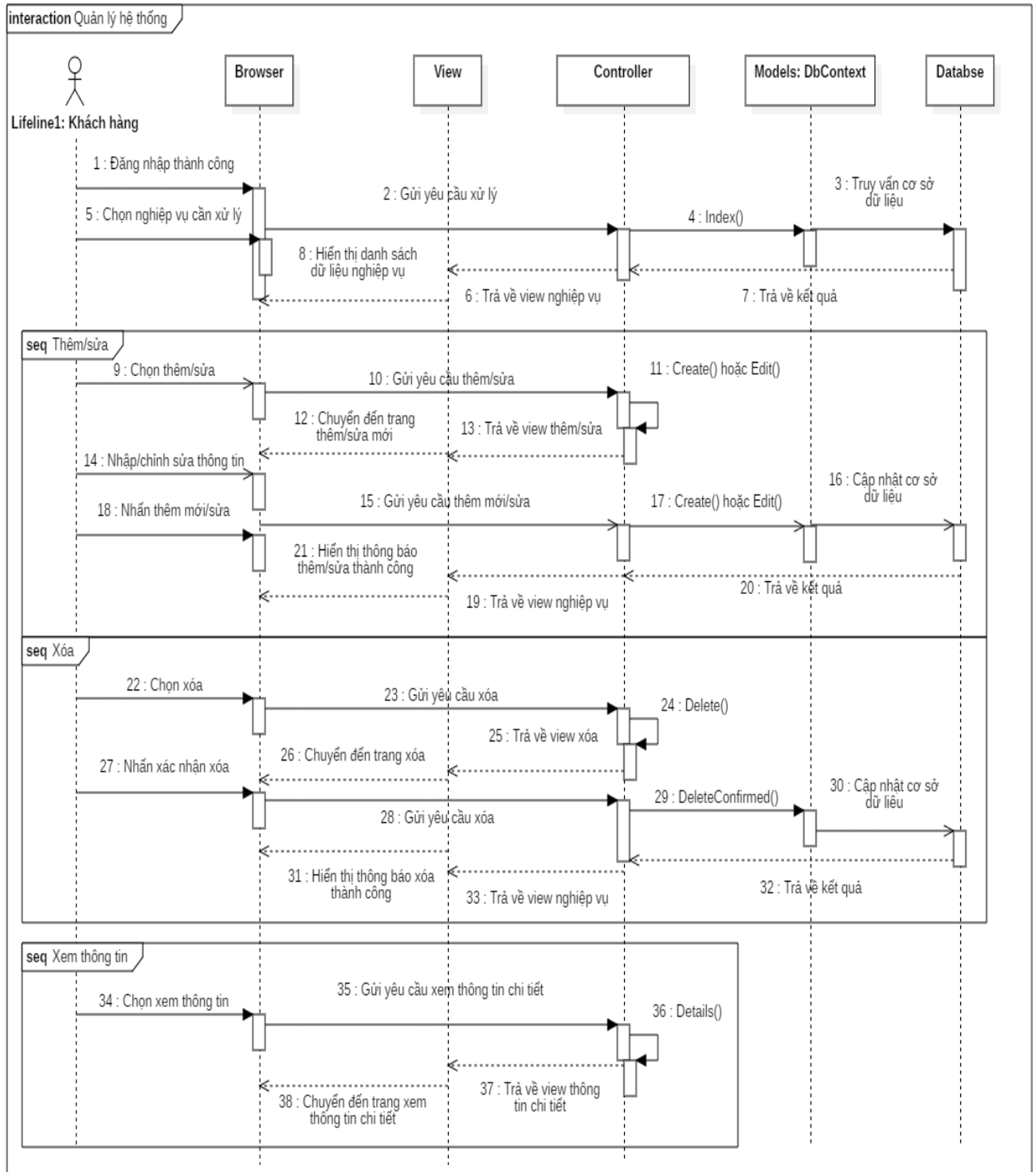
Hình 2.33 Sơ đồ tuần tự quản lý tài khoản

2.4.3.5 Quản lý đơn hàng

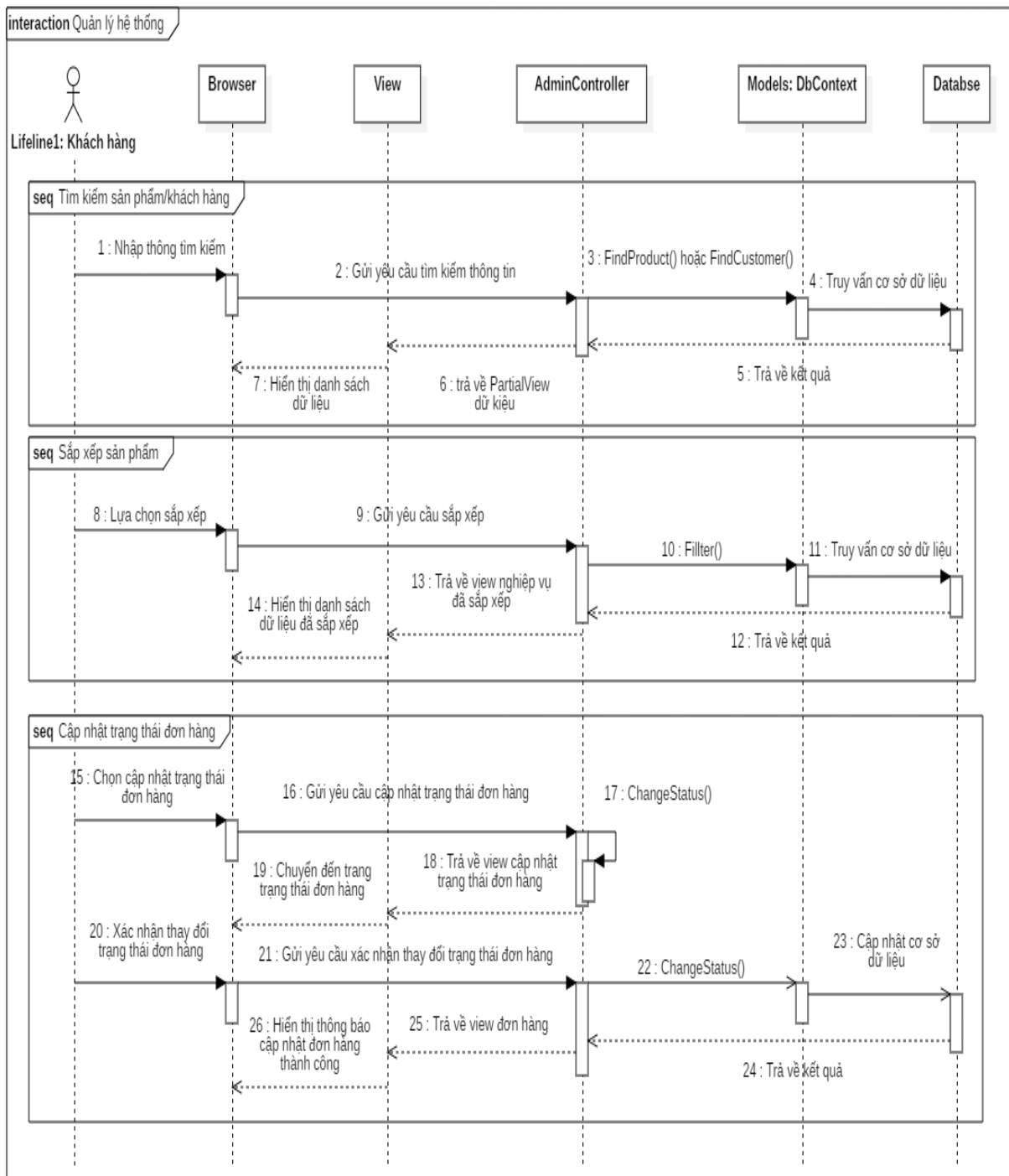


Hình 2.34 Sơ đồ tuần tự quản lý đơn hàng

2.4.3.6 Quản lý hệ thống

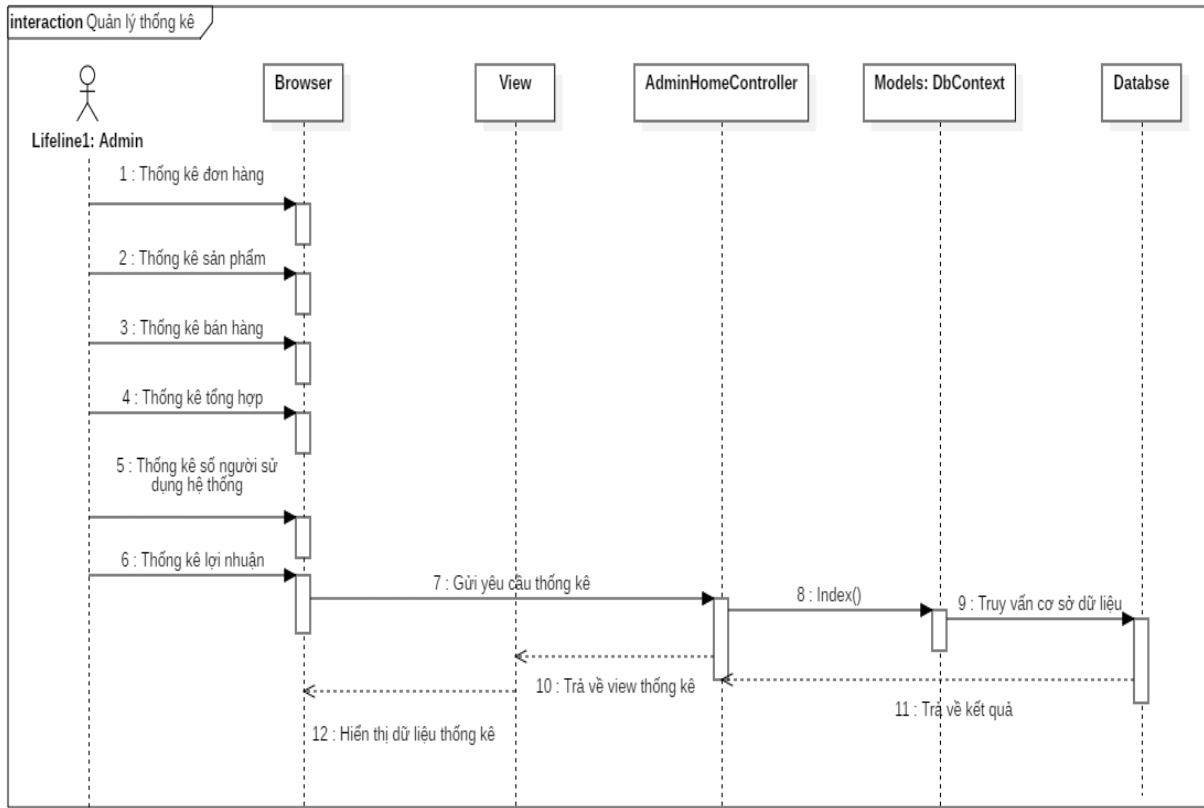


Hình 2.35 Sơ đồ tuần tự quản lý hệ thống chức năng chính



Hình 2.36 Sơ đồ tuần tự quản lý hệ thống chức năng phụ

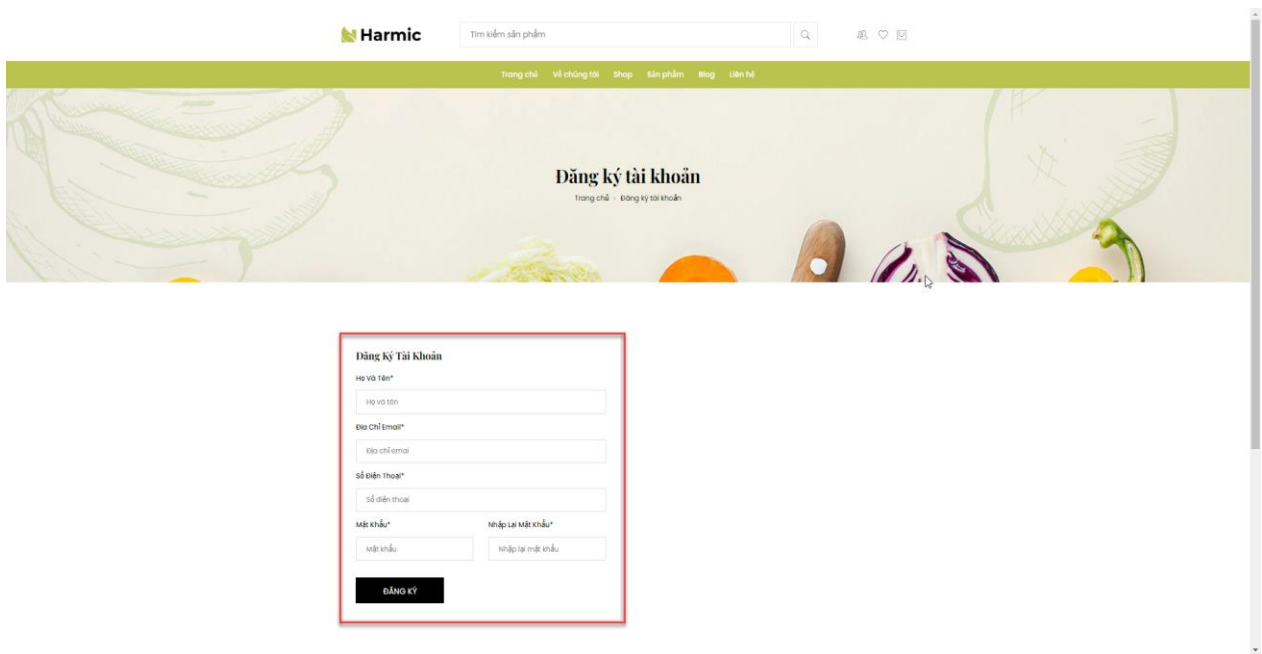
2.4.3.7 Quản lý thống kê



Hình 2.37 Sơ đồ tuần tự quản lý thống kê

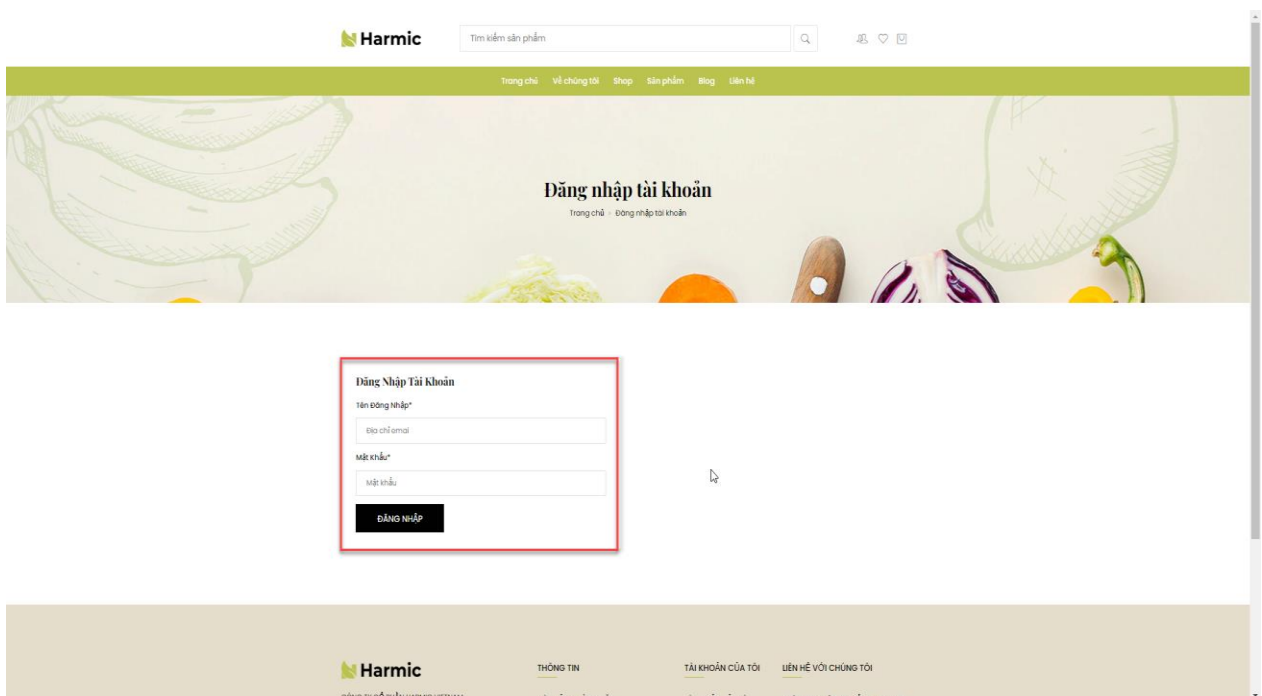
2.5 Thiết kế giao diện hệ thống

2.5.1 Giao diện dành cho khách hàng



The screenshot displays the Harmic website's registration page. The header features the Harmic logo, a search bar, and navigation links. The main banner area has a background image of vegetables and the text "Đăng ký tài khoản" (Register account) with a link to "Đăng ký tài khoản" (Register account). Below the banner, a registration form is highlighted with a red border. The form includes fields for "Họ và tên*" (Full name), "Email*", "Số điện thoại*" (Phone number), and "Mật khẩu*" (Password) with a "Nhập lại mật khẩu*" (Repeat password) field. A "Đăng ký" (Register) button is at the bottom of the form.

Hình 2.38 Giao diện đăng ký khách hàng



The screenshot displays the Harmic website's login page. The header is identical to the registration page. The main banner area has the text "Đăng nhập tài khoản" (Login account) with a link to "Đăng nhập tài khoản" (Login account). Below the banner, a login form is highlighted with a red border. The form includes fields for "Tên đăng nhập*" (Username) and "Mật khẩu*" (Password). A "Đăng nhập" (Login) button is at the bottom of the form. The footer contains the Harmic logo, contact information, and links to "THÔNG TIN", "TÀI KHOẢN CỦA TÔI", and "LIÊN HỆ VỚI CHÚNG TÔI".

Hình 2.39 Giao diện đăng nhập khách hàng

Trang chủ | Về chúng tôi | Shop | Sản phẩm | Blog | Liên hệ

Tài khoản của tôi

Home > Tài khoản của tôi

THÔNG TIN TÀI KHOẢN

DANH SÁCH ĐƠN HÀNG

THAY ĐỔI MẬT KHẨU

ĐĂNG XUẤT

Họ và Tên*

Aaaa

Email*

baol@gmail.com

Số điện thoại*

111111112

Ngày sinh nhật

05/10/2022

LƯU THAY ĐỔI

Harmic

THÔNG TIN | TÀI KHOẢN CỦA TÔI | LIÊN HỆ VỚI CHÚNG TÔI

Hình 2.40 Giao diện cập nhật thông tin tài khoản

Tài khoản của tôi

Home > Tài khoản của tôi

THÔNG TIN TÀI KHOẢN

DANH SÁCH ĐƠN HÀNG

THAY ĐỔI MẬT KHẨU

ĐĂNG XUẤT

DANH SÁCH ĐƠN HÀNG

Mã hóa đơn	Ngày mua hàng	Ngày ship hàng	Trạng thái	Tổng tiền	
#1010	11/05/2022 01:28:57		Chờ xác nhận	3,000 VNĐ	Xem đơn hàng

THÔNG TIN ĐƠN HÀNG: ID#1010

Tên người nhận hàng: Aaaa

Số điện thoại: 111111112

Địa chỉ giao hàng: test

Tổng tiền: 3,000 VNĐ

Phương/Xã: Xuân Phú

Quận/Huyện: Thành Phố Huế

Tỉnh/Thành: Huế

Phương thức thanh toán: Chuyển khoản ngân hàng

Trạng thái đơn hàng: Chờ xác nhận

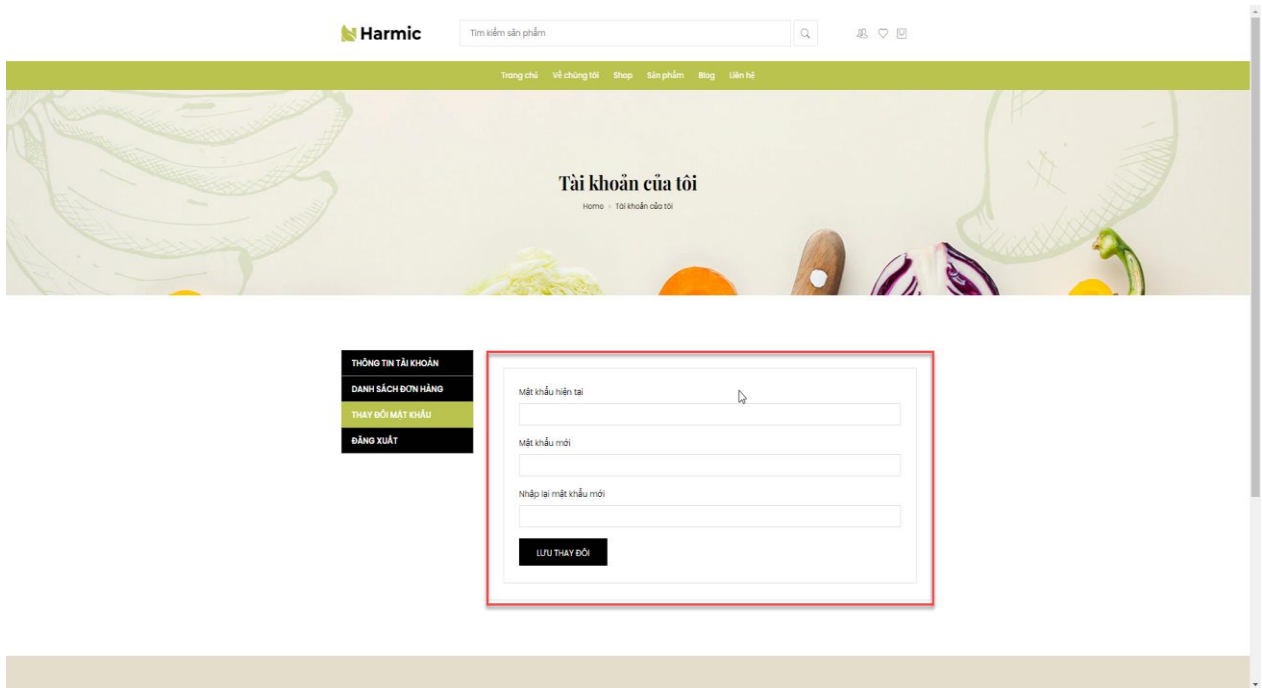
Ngày đặt hàng: 11/05/2022 01:28:57

Ngày giao hàng:

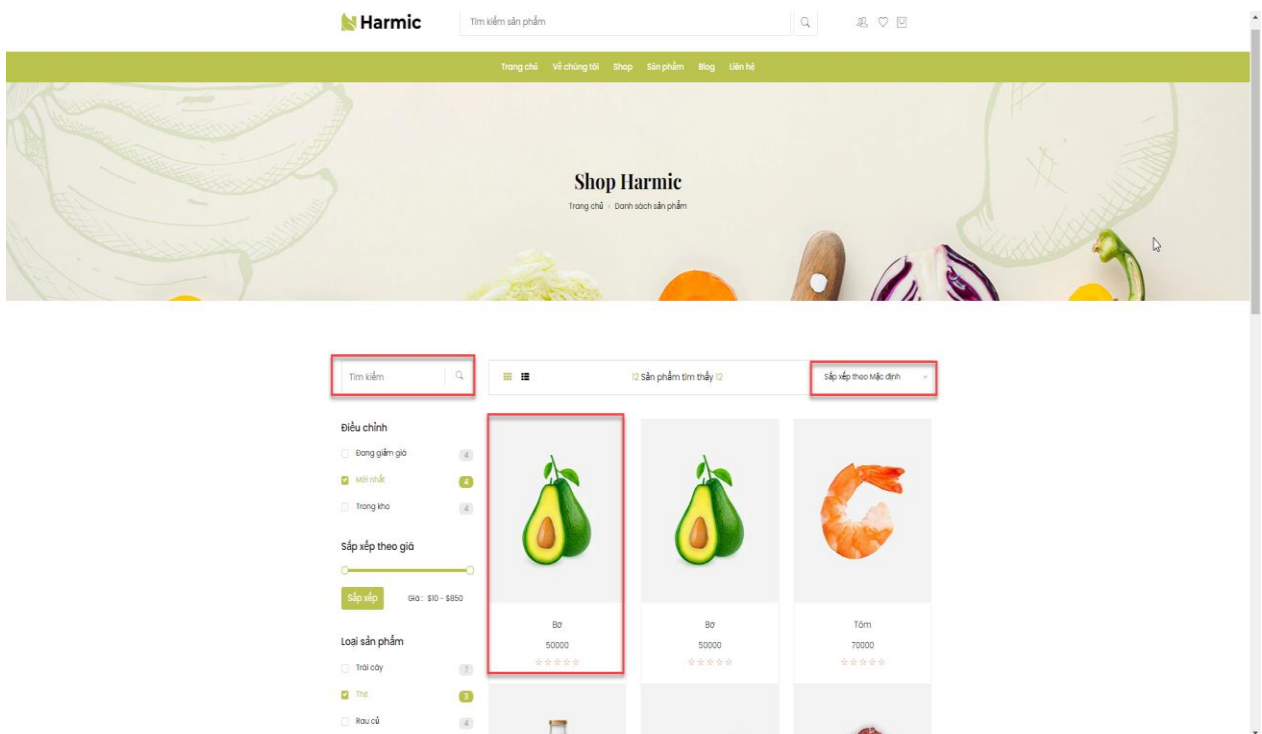
#	Sản phẩm	Số lượng	Đơn giá	Thành tiền
1	Táo	1	3,000 VNĐ	3,000 VNĐ

Hủy đơn hàng

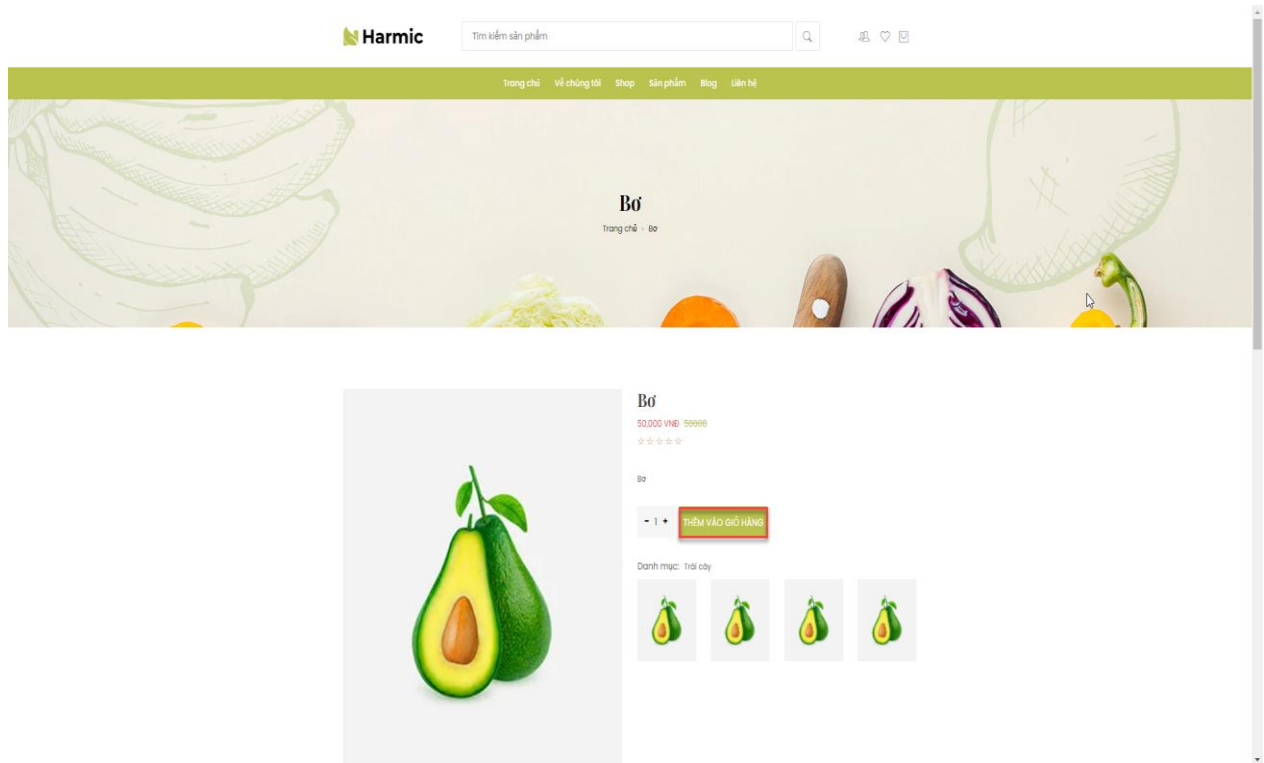
Hình 2.41 Giao diện quản lý đơn hàng



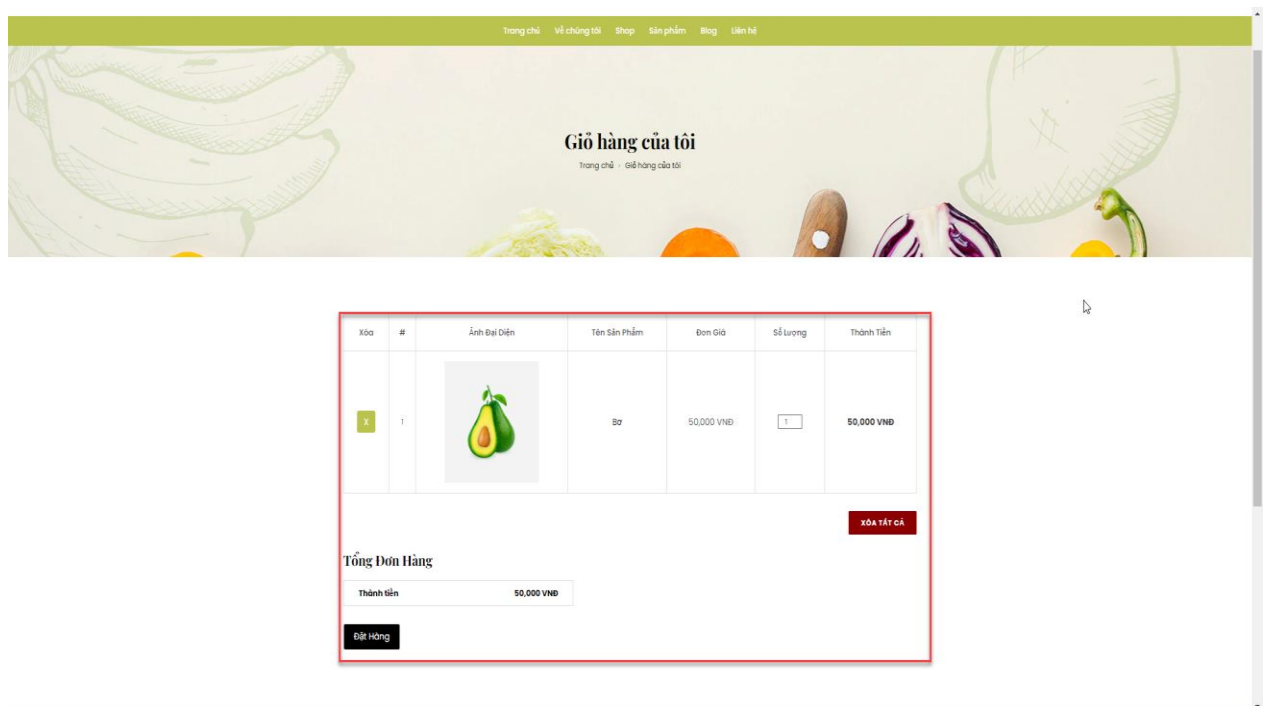
Hình 2.42 Giao diện thay đổi mật khẩu



Hình 2.43 Giao diện quản lý mua hàng



Hình 2.44 Giao diện thông tin sản phẩm



Hình 2.45 Giao diện quản lý giỏ hàng

Trang chủ Về chúng tôi Shop Sản phẩm Blog Liên hệ

THÔNG TIN MUA HÀNG

Home > THÔNG TIN MUA HÀNG

THÔNG TIN GIAO HÀNG

Họ và Tên*
Aaaa

Số điện thoại*
111111112

Email*
baol@gmail.com

Địa chỉ nhận hàng*
test

Tên/Thành*
Tỉnh/Thành

Quận/Huyện*
Quận/Huyện

Phường/Xã*
Phường/Xã

Ghi chú
Nội dung ghi chú

THÔNG TIN ĐƠN HÀNG

#	TÊN SẢN PHẨM	THÀNH TIỀN
1	Bơ x 1	50,000 VND

SỐ TIỀN PHẢI THANH TOÁN 50,000 VND

Chọn phương thức thanh toán*

Chọn phương thức thanh toán

ĐẶT HÀNG

Hình 2.46 Giao diện đặt hàng

Trang chủ Về chúng tôi Shop Sản phẩm Blog Liên hệ

Đặt hàng thành công

THÔNG TIN ĐƠN HÀNG

Mã đơn hàng: ID#1011

Tên người nhận hàng: Aaaa

Số điện thoại: 111111112

Địa chỉ: test

Phường: Xuân Phú

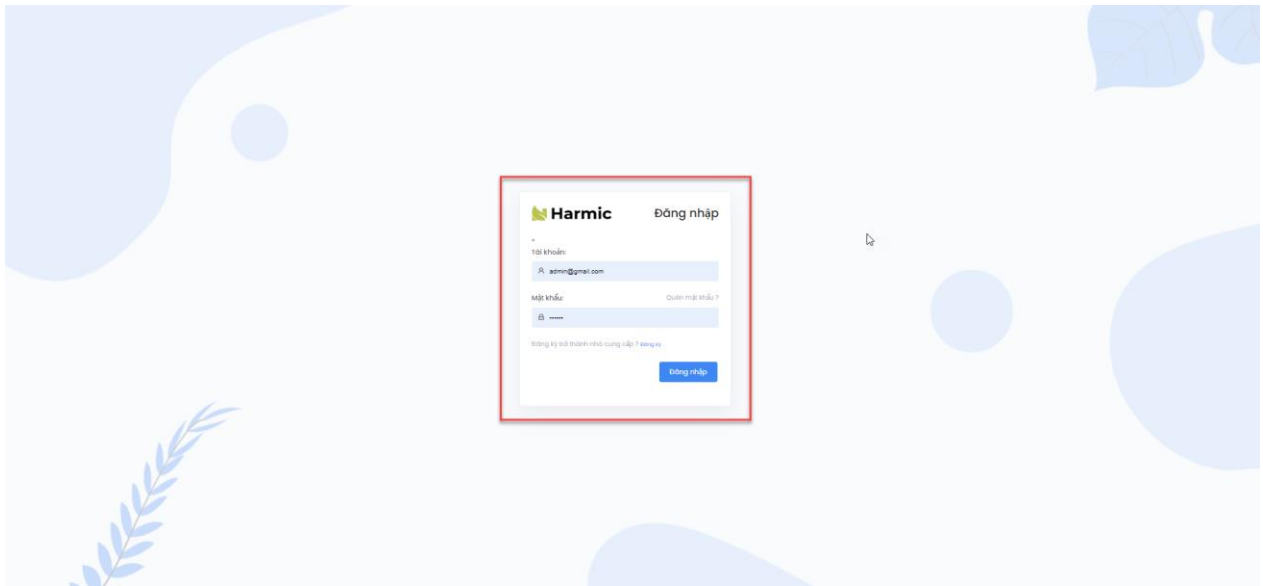
Quận/Huyện: Thành Phố Huế

Tỉnh/Thành: Huế

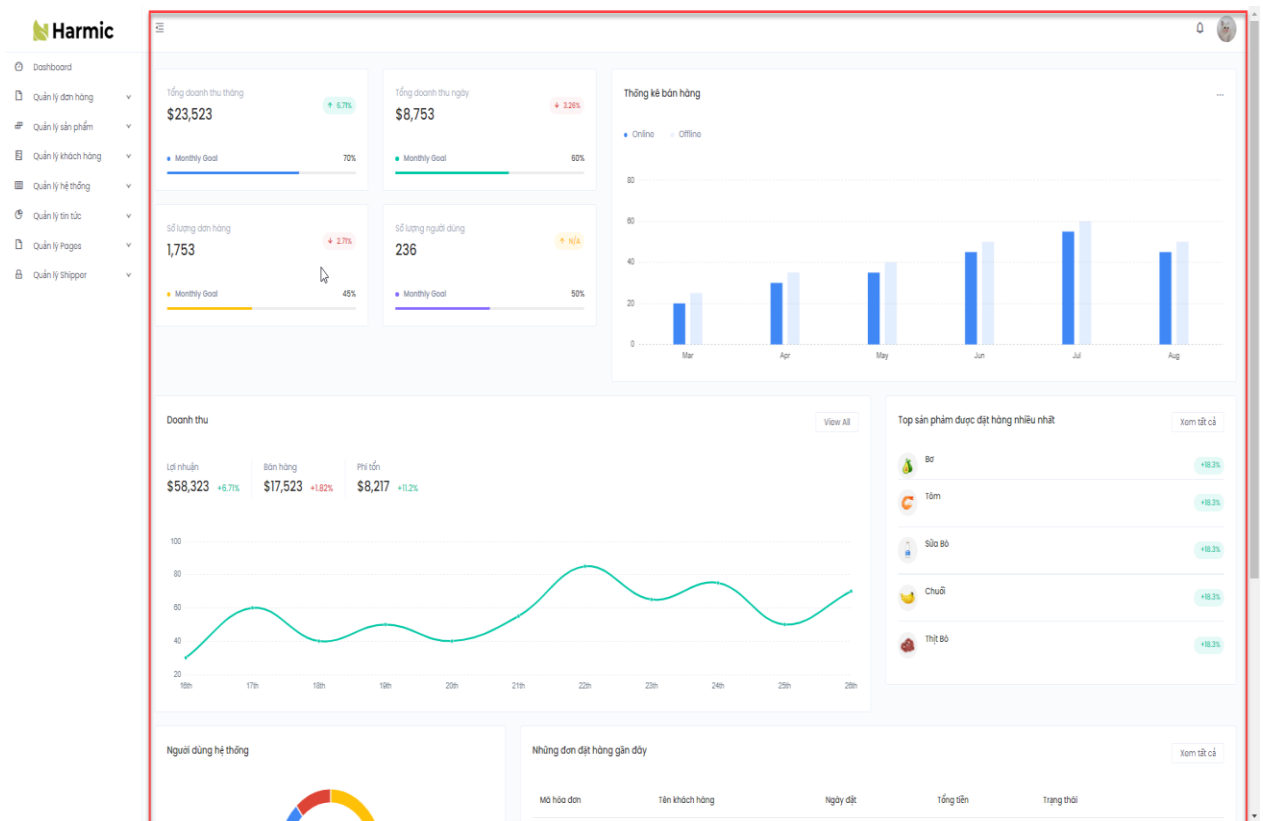
TIẾP TỤC MUA HÀNG

Hình 2.47 Giao diện đặt hàng thành công

2.5.2 Giao diện dành cho admin

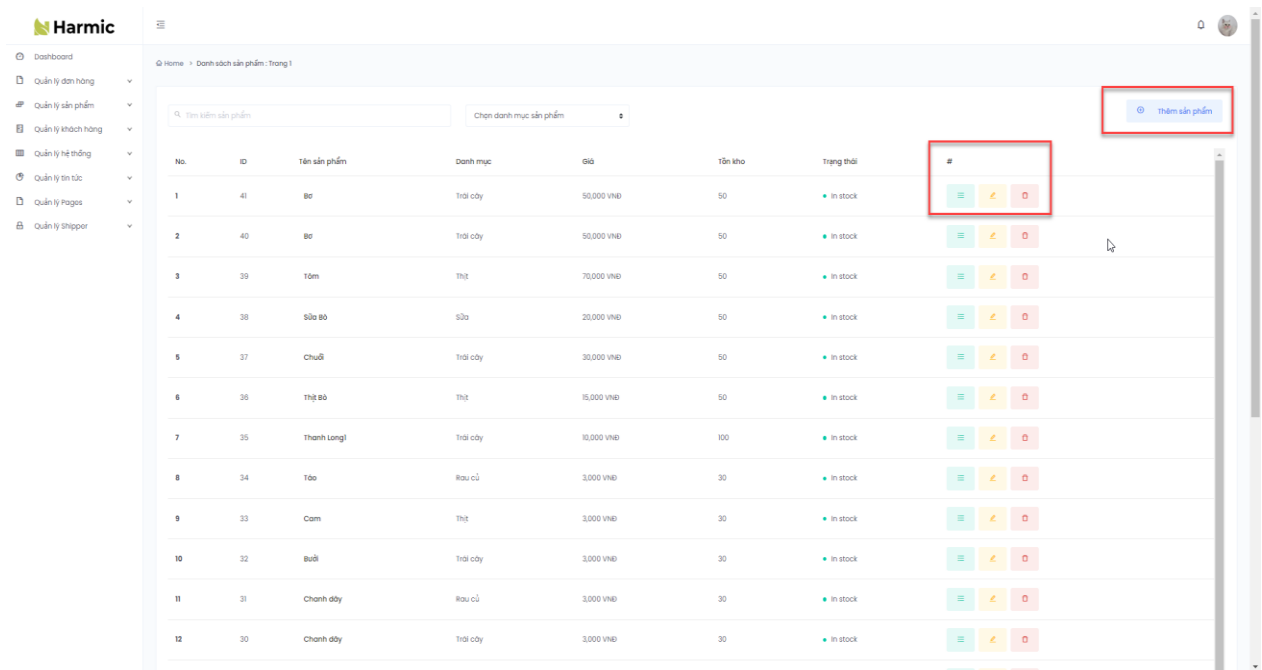


Hình 2.48 Giao diện đăng nhập admin

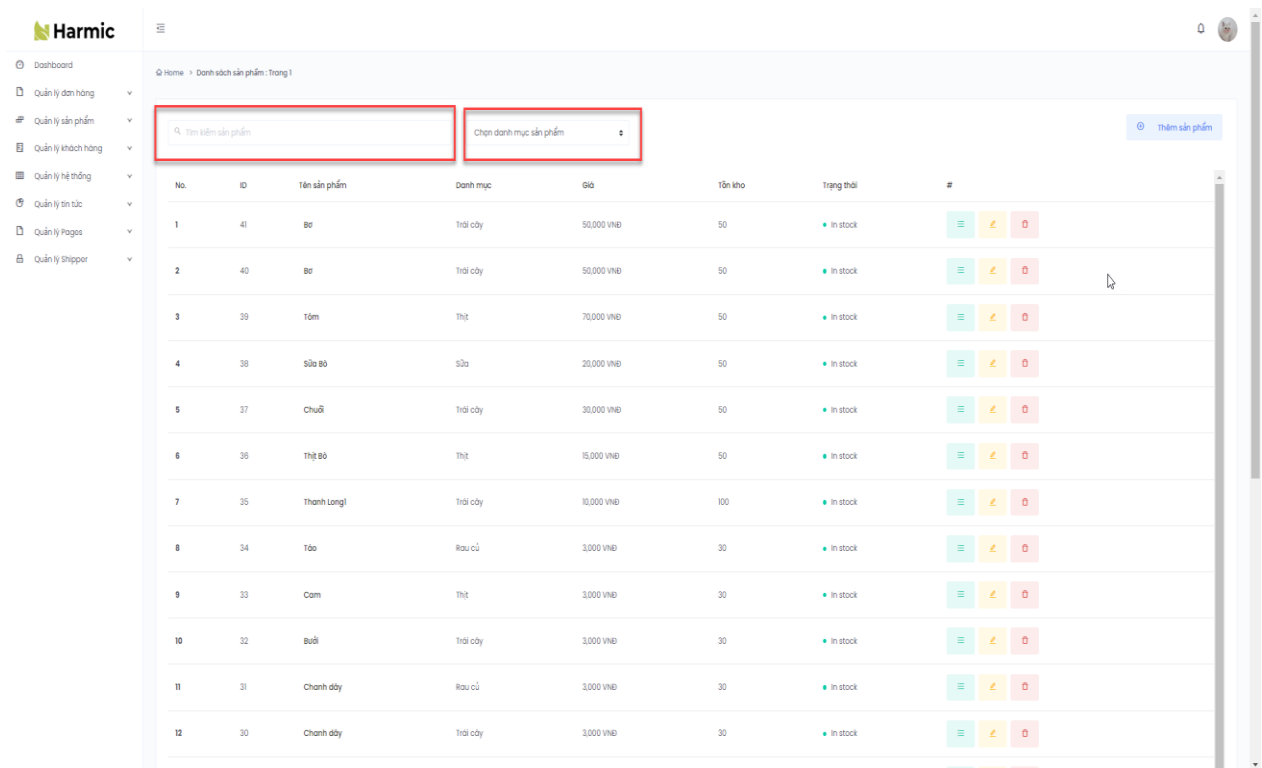


Hình 2.49 Giao diện quản lý thống kê

Tìm hiểu ASP.NET Core MVC để xây dựng ứng dụng đi chợ online



Hình 2.50 Giao diện quản lý hệ thống chức năng chính



Hình 2.51 Giao diện quản lý hệ thống chức năng phụ

2.6 Cài đặt phần mềm lập trình và công nghệ sử dụng

2.6.1 Cài đặt phần mềm lập trình

2.6.1.1 Cài đặt phần mềm

- Visual Studio 2022 - Enterprise
- SQL Server 2019

2.6.1.2 Yêu cầu hệ thống

- Đối với Visual Studio 2022
 - ✓ Yêu cầu bộ xử lý 2,6 GHz hoặc nhanh hơn
 - ✓ Yêu cầu tối thiểu RAM 4GB trở lên
 - ✓ Yêu cầu 10GB dung lượng đĩa cứng trở lên
- Đối với Sql Server 2019
 - ✓ Yêu cầu bộ xử lý tối thiểu 1,4 GHz hoặc nhanh hơn
 - ✓ Hệ điều hành win 10 TH1 1507 Trở lên
 - ✓ Yêu cầu tối thiểu RAM 4GB trở lên
 - ✓ Yêu cầu 10GB dung lượng đĩa cứng trở lên

2.6.2 Công nghệ sử dụng

- Ngôn ngữ lập trình C# 10.0, Linq.
- Hệ quản trị cơ sở dữ liệu SQL Server 2019.
- Framework ASP.NET Core MVC 6, Entity Framework 6.
- HTML, SCSS, CSS, ,Bootstrap, JQuery, AJAX.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN TƯƠNG LAI

1.1 Kết luận

Không thể không tin học hóa: Việc tin học hóa hệ thống website đi chợ online là vô cùng hữu ích và cần thiết trong thời gian hiện tại, thời điểm công nghệ thông tin phát triển nhanh và chiếm một vai trò quan trọng trong cuộc sống thường ngày.

Với Sự xuất hiện dịch bệnh covid-19 là cơ hội và thách thức đối với việc tin học hóa hệ thống website đi chợ online là quá phù hợp. Có thể hỗ trợ trong việc mua bán sản phẩm tiêu dùng cả trong trận chiến phòng chống dịch bệnh covid-19.

Hệ thống website đi chợ online không những đem lại lợi ích cho nhà cung cấp với khách hàng cho việc trao đổi mua bán thuận lợi hơn trong nghiệp vụ của hệ thống. Người dùng sẽ được trải nghiệm những dịch vụ tốt nhất dựa vào tin học hóa.

Qua dự án, em hiểu hơn về ASP.NET Core MVC, Bootstrap, HTML, CSS và JavaScript... biết cách sử dụng các công cụ hỗ trợ xây dựng web nhằm thực hiện xây dựng, thiết kế thuận tiện và tiết kiệm thời gian, công sức trong quá trình thực hiện dự án.

❖ Kết quả đã đạt được:

- ✓ Hiểu được khái quát về MVC, các lợi ích của ASP.NET Core MVC mang lại
- ✓ Hiểu được các bước cơ bản để tạo một website bằng ASP.NET Core MVC.
- ✓ Hiểu rõ cách thức hoạt động của một dự án làm bằng Framework MVC cũng như nắm được mô hình của nó.
- ✓ Xây dựng thành công một website thương mại điện tử được viết trên nền tảng ASP.NET Core MVC.

1.2 Hướng phát triển tương lai

Tuy hệ thống đã được hoàn thành nhưng chỉ với một khoản thời gian ngắn thực hiện dự án thì không thể nói là nó đã hoàn thiện. Dự án xây dựng trang website đi chợ online còn những thiếu sót, sự cố tồn tại đồng thời cũng là những tiềm năng để phát triển và hoàn thiện.

❖ Hướng phát triển một số chức năng của hệ thống đó là :

- Nâng cấp phần tương tác giữa khách hàng và nhà cung cấp
- Nâng cao chất lượng sản phẩm qua đánh giá của từng khách hàng phải công khai đến người dùng tiêu dùng.
- Người tiêu dùng có thể bình luận tương tác với nhau nếu muốn
- Khách hàng sẽ có nhiều nghiệp vụ mới mẻ hơn thuận lợi hơn trong việc mua các sản phẩm tại hệ thống
- Hệ thống sẽ mở rộng cho từng nhu cầu mục đích sử dụng người dùng
- Hệ thống tối ưu được tốc độ cũng như bảo mật hơn áp dụng công nghệ mới tốt hơn được nâng cấp trong tương lai đem lại cho sự tin tưởng an toàn cho khách hàng

Những ý kiến nêu trên là những hướng phát triển dự án này trong thời gian tới và em rất mong những được những nhận xét và góp ý của các thầy, cô giáo về những ý kiến trên để có thể có được những nhìn nhận, kinh nghiệm về việc phát triển các dự án sau này.

TÀI LIỆU THAM KHẢO

- <https://docs.microsoft.com/vi-vn/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-6.0&tabs=visual-studio>
- <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/view-components?view=aspnetcore-6.0>
- <https://docs.microsoft.com/en-us/dotnet/api/system.security.claims.claimsidentity?view=net-6.0>
- <https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/areas?view=aspnetcore-6.0>
- <https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-6.0>
- <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio>
- <https://docs.microsoft.com/en-us/aspnet/core/security/authorization/claims?view=aspnetcore-6.0>
- <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-6.0>
- <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/mvc-music-store/mvc-music-store-part-3>
- <https://github.com/aspnetcorehero/ToastNotification>
- <https://github.com/hieudole/PagedList.Core.Mvc>