

**2022**

**KHÓA LUẬN**

**TỐT NGHIỆP**

*Học để có nghề nghiệp*

*Học để có việc làm*

*Học để có khả năng tự học*

ĐỀ TÀI:

**THIẾT KẾ WEBSITE CỬA HÀNG QUẦN ÁO TRỰC TUYẾN**

*Họ và tên*: HOÀNG LÊ ANH TUẤN

*Ngành*: CÔNG NGHỆ THÔNG TIN

*Giảng viên hướng dẫn*: TH.S NGUYỄN DŨNG

HUẾ, THÁNG 9 NĂM 2022

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô của trường, đặc biệt là các thầy cô khoa công nghệ thông tin của trường đại học Phú Xuân đã giúp đỡ em về tài liệu tham khảo, để em có thể hoàn thành tốt Báo cáo tốt nghiệp.

Và em cũng xin chân thành cảm ơn thầy đã nhiệt tình hướng dẫn em trong việc lựa chọn đề tài nghiên cứu, hướng tiếp cận và giúp em chỉnh sửa những thiếu sót trong quá trình nghiên cứu.

Trong quá trình nghiên cứu, khó tránh khỏi sai sót, rất mong các Thầy, Cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài luận văn không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp từ phía Thầy, Cô để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn.

Em xin chân thành cảm ơn!

Huế, tháng 9 năm 2022

**Sinh viên**

**Hoàng Lê Anh Tuấn**

MỤC LỤC

[LỜI CẢM ƠN 2](#_Toc110954961)

[CHƯƠNG 1. TÌM HIỂU VÀ SỬ DỤNG ASP NET CORE 5](#_Toc110954962)

[1.1 Tổng quan về Asp net core 5](#_Toc110954963)

[1.2. Một số nội dung chính trong ASP.NET Core 6](#_Toc110954964)

[1.2.1 ASP.NET Core tích hợp được với nhiều client side framework 6](#_Toc110954965)

[1.2.2 Cơ chế Routing ASP.NET core 6](#_Toc110954966)

[1.3 Hiệu năng của ASP.NET Core 9](#_Toc110954967)

[1.4 Lý do sử dụng ASP.NET Core 10](#_Toc110954968)

[1.5 Sử dụng ASP.NET Core trong việc xây dựng web UI, web API và phát triển Client-side 13](#_Toc110954969)

[CHƯƠNG 2. TÌM HIỂU VÀ SỬ DỤNG MICROSFT SQLSERVER 13](#_Toc110954970)

[2.1 Tổng quan về cơ sở dữ liệu 13](#_Toc110954971)

[2.2 Tổng quan về SQL 17](#_Toc110954972)

[2.2.1 Nguồn gốc 17](#_Toc110954973)

[2.2.2 Tại sao sử dụng SQL và nó lại quan trọng 17](#_Toc110954974)

[2.2.3 Con đường sự nghiệp cho SQL Developer nói riêng và Database Developer 18](#_Toc110954975)

[2.3 Lý do sử dụng SQL 18](#_Toc110954976)

[CHƯƠNG 3. SỬ DỤNG API 19](#_Toc110954977)

[3.1 Tổng quan về API 19](#_Toc110954978)

[3.2 Cấu trúc của một ứng dụng Web API 20](#_Toc110954979)

[3.3 Một số nội dung chính trong ASP.NET Rest Api 26](#_Toc110954980)

[3.3.1 Cơ chế Model Binding: Truyền dữ liệu từ View lên Controller 26](#_Toc110954981)

[3.3.2 Request Life Cycle trong ASP.NET Core 29](#_Toc110954982)

[CHƯƠNG 4. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 32](#_Toc110954983)

[4.1 Xác định yêu cầu bài toán: 32](#_Toc110954984)

[4.2 Xác định các chức năng 32](#_Toc110954985)

[4.2.1 các yêu cầu chức năng: 32](#_Toc110954986)

[4.2.2. Các yêu cầu phi chức năng 33](#_Toc110954987)

[4.3 Đặc tả các chức năng của hệ thống: 34](#_Toc110954988)

[4.3.1 MS01 -Đăng nhập: 34](#_Toc110954989)

[4.3.2 MS02 -Đăng xuất 34](#_Toc110954990)

[4.3.3 MS03 -Quản lý loại sản phẩm: 35](#_Toc110954991)

[4.3.4 MS04 -Quản lý sản phẩm: 37](#_Toc110954992)

[4.3.5 MS06 - Quản lý đơn hàng 38](#_Toc110954993)

[4.3.6 MS07 - Quản lý khách hàng 40](#_Toc110954994)

[4.3.7 MS08 - Xem chi tiết sản phẩm 40](#_Toc110954995)

[4.3.8 MS09 - Tìm kiếm sản phẩm 41](#_Toc110954996)

[4.3.9 MS10 - Xem chi tiết giỏ hàng 42](#_Toc110954997)

[4.4. Đặc tả thiết kế hệ thống: 43](#_Toc110954998)

[4.4.1 Thiết kế và cách xử lý: 43](#_Toc110954999)

[4.4.1.3 Xem lịch sử khách hàng 45](#_Toc110955000)

[4.4.2 Thiết kế cơ sở dữ liệu: 56](#_Toc110955001)

[4.4.3 Thiết kế giao diện người dùng: 60](#_Toc110955002)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 69](#_Toc110955003)

[1. KẾT QUẢ ĐẶT ĐƯỢC 69](#_Toc110955004)

[2. HẠN CHẾ 69](#_Toc110955005)

[3. HƯỚNG PHÁT TRIỂN 70](#_Toc110955006)

# CHƯƠNG 1. TÌM HIỂU VÀ SỬ DỤNG ASP NET CORE

## 1**.1 Tổng quan về Asp net core**

ASP.NET đã được sử dụng từ nhiều năm để phát triển các ứng dụng web. Kể từ đó, framework này đã trải qua một sự thay đổi tiến hóa đều đặn và cuối cùng đã đưa chúng ta đến với ASP.NET Core 1.0 hậu duệ mới gần đây nhất của nó.

* ASP.NET Core không phải là phiên bản tiếp theo của ASP.NET 4.6
* Đó là một framework hoàn toàn mới, may mắn rằng nó một dự án side-by-side tương tự với mọi thứ mà chúng ta biết.
* Nó thực ra được viết lại trên framework ASP.NET 4.6 hiện tại những kích thước nhỏ hơn và nhiều modular hơn.

ASP.NET Core là một web framework mã nguồn và được tối ưu hóa cho cloud để phát triển các ứng dụng web chạy trên nhiều nền tảng như Windows, Linux và Mac. Hiện tại, nó bao gồm MVC framework được kết hợp các tính năng của MVC và Web API thành một web framework duy nhất.

* Các ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên .NET Framework hoàn chỉnh.
* Nó đã được thiết kế để cung cấp một framework tối ưu cho các ứng dụng để triển khai tới cloud hoặc chạy on-premises.
* Nó bao gồm những modular với các thành phần tối thiểu, do đó bạn giữ được tính linh hoạt trong quá trình xây dựng các giải pháp của mình.
* Bạn có thể phát triển và chạy các ứng dụng đa nền tảng từ ASP.NET Core trên Windows, Mac và Linux.

**Ưu điểm của ASP.NET Core**

ASP.NET Core đi kèm với những ưu điểm sau:

* ASP.NET Core có một số thay đổi kiến trúc dẫn đến modular framework nhỏ hơn.
* ASP.NET Core không còn dựa trên System.Web.dll. Nó dựa trên một tập hợp nhiều yếu tố của Nuget packages.
* Điều này cho phép bạn tối ưu ứng dụng của mình chỉ cần những NuGet packages cần thiết.
* Lợi ích của diện tích bề mặt ứng dụng nhỏ hơn thì bảo mật chặt chẽ hơn, giảm dịch vụ, cải thiện hiệu suất và giảm chi phí.
* Với ASP.NET Core, bạn có thể nhận được các cải tiến sau:
* Xây dựng và chạy các ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux.
* Được xây dựng trên **.NET CORE** , hỗ trợ side-by-side app versioning.
* Công cụ mới giúp đơn giản hóa việc phát triển web hiện đại.
* Liên kết đơn các web stack như Web UI và API Web.
* Cấu hình dựa trên môi trường đám mây sẵn có.
* Được xây dựng dựa trên cho DI (Dependency Injection).
* Tag Helpers làm cho các Razor makup trở nên tự nhiên hơn với HTML.
* Có khả năng host trên IIS hoặc self-host.

## 1.2. Một số nội dung chính trong ASP.NET Core

### 1.2.1 ASP.NET Core tích hợp được với nhiều client side framework

ASP.NET là một nền tảng ứng dụng web (web application framework) được phát triển và cung cấp bởi Microsoft, cho phép những người lập trình tạo ra những trang web động, những ứng dụng web và những dịch vụ web. Đây là sản phẩm tích hợp của các client side framework và phương thức-giao thức kết nối các thư viện khác một cách liên tục bao gồm HTML, CSS, bootstrap và JavaScript, cao cấp hơn ta có thể tích hợp Angular, api, reactjs, nodejs, vv…

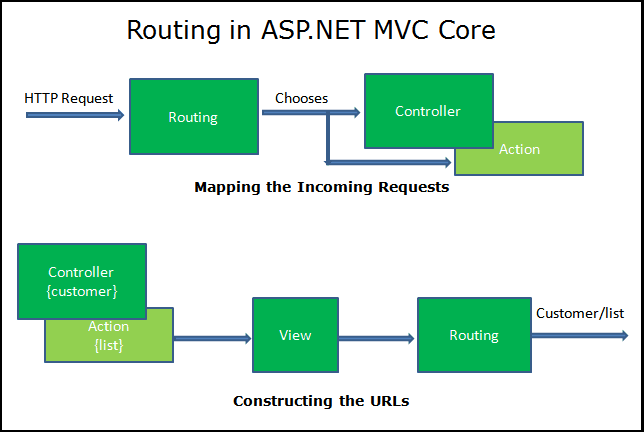
ASP.NET core là framework ngày càng trở nên phổ biến hàm với nhiều tính năng và các thư viện được hỗ trợ cho developer, điều đó khiến cho các giao diện web trở nên sinh động hơn. Mọi thứ mà chung là có thể làm được trên web ngày nay là Javascript có thể chạy được trên browser, cũng như chạy định server. Thật vậy, rõ ràng rằng từ xưa đến nay chúng ta vẫn quan niệm rằng lập trình trên phía server chỉ dùng được những ngôn ngữ như php, ruby, java,… nhưng từ khi ASP.NET core ra đời và thay thế ASP.NET MVC nó mang đến phương thức mới cho việc lập trình cho phía server cũng như client.

### 1.2.2 Cơ chế Routing ASP.NET core

Routing là một quá trình khi ASP.NET Core xem xét các URL request gửi đến và "chỉ đường" cho nó đến Controller Actions. Nó cũng được sử dụng để tạo ra URL đầu ra. Quá trình này được đảm nhiệm bởi Routing Middleware. Routing Middleware có sẵn trong thư viện ***Microsoft.AspNetCore.Routing***.

Routing có 2 trách nhiệm chính:

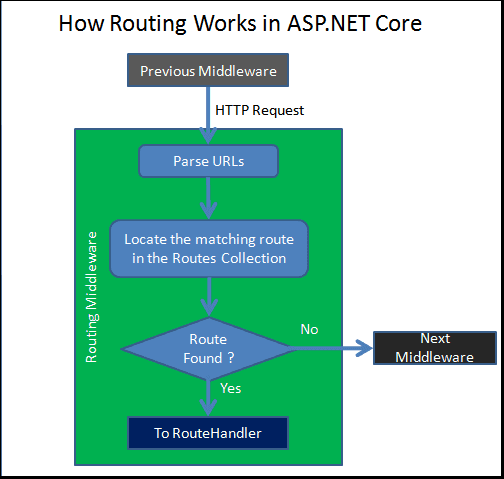
* Nó map request đến vào Controller Action.
* Tạo ra URL đầu ra tương ứng với Controller action.



*Hình 1.1 Cơ chế Routing Asp.net Core*

**Cách thức Routing làm việc**

Mô hình dưới đây mô tả cách làm việc của cơ chế Routing trong ASP.NET Core:



*Hình 1.2 Mô tả cách làm việc của cơ chế Routing*

Khi request đến thì Routing Middleware sẽ làm những việc sau đây:

* Phân tích URL.
* Tìm kiếm xem có cái Route nào match trong **RouteCollection.**
* Nếu Route tìm thấy thì đẩy nó sang **RouteHandler.**
* Nếu không tìm thấy Route nào thì bỏ qua và gọi middleware tiếp theo.

**Route là gì?**

Route tương tự như bản đồ. Chúng ta sử dụng bản đồ để đi đến điểm đích. Tương tự như thế, ứng dụng ASP.NET Core sử dụng Route để đến controller action.

Mỗi Route bao gồm các thông tin như tên, mẫu URL (URL pattern) hay còn gọi là template url, thông tin controller action mặc định và ràng buộc (constraints). URL pattern được so sánh với URL đến xem có đúng mẫu không. Một ví dụ của URL pattern là: {controller=Home}/{action=Index}/{id?}

Route được định nghĩa trong Microsoft.AspNetCore.Routing.

**Route Collection là gì?**

Route Collection là một tập hợp tất cả ác Route trong ứng dụng. Một ứng dụng sẽ lưu một tập hợp các route ở một nơi duy nhất trong bộ nhớ. Các Route này sẽ thêm vào collection khi ứng dụng khởi động. Route Module sẽ tìm kiếm một Route match với URL request đến trong mỗi một Route của Route Collection. Route Collection được định nghĩa trong **Microsoft.AspNetCore.Routing.**

**Route Handler là gì?**

Route Handler là một thành phần quyết định sẽ làm gì với Route. Khi cơ chế routing tìm được một Route thích hợp cho một request đến, nó sẽ gọi đến **RouteHandler**và gửi route đó cho **RouteHandler**xử lý. Route Handler là class triển khai từ interface **IRouteHandler**. Trong ASP.NET Core thì Route được xử lý bởi **MvcRouteHandler**.

**MVCRouteHandler**

Đây là Route Handler mặc định của ASP.NET Core MVC Middleware. **MVCRouteHandler**được đăng ký khi đăng ký MVC Middleware. Có thể ghi đè việc này bằng cách tự tạo cho mình một custom implementation của Route Handler. MVCRouteHandler được định nghĩa trong namespace: **Microsoft.AspnetCore.Mvc**

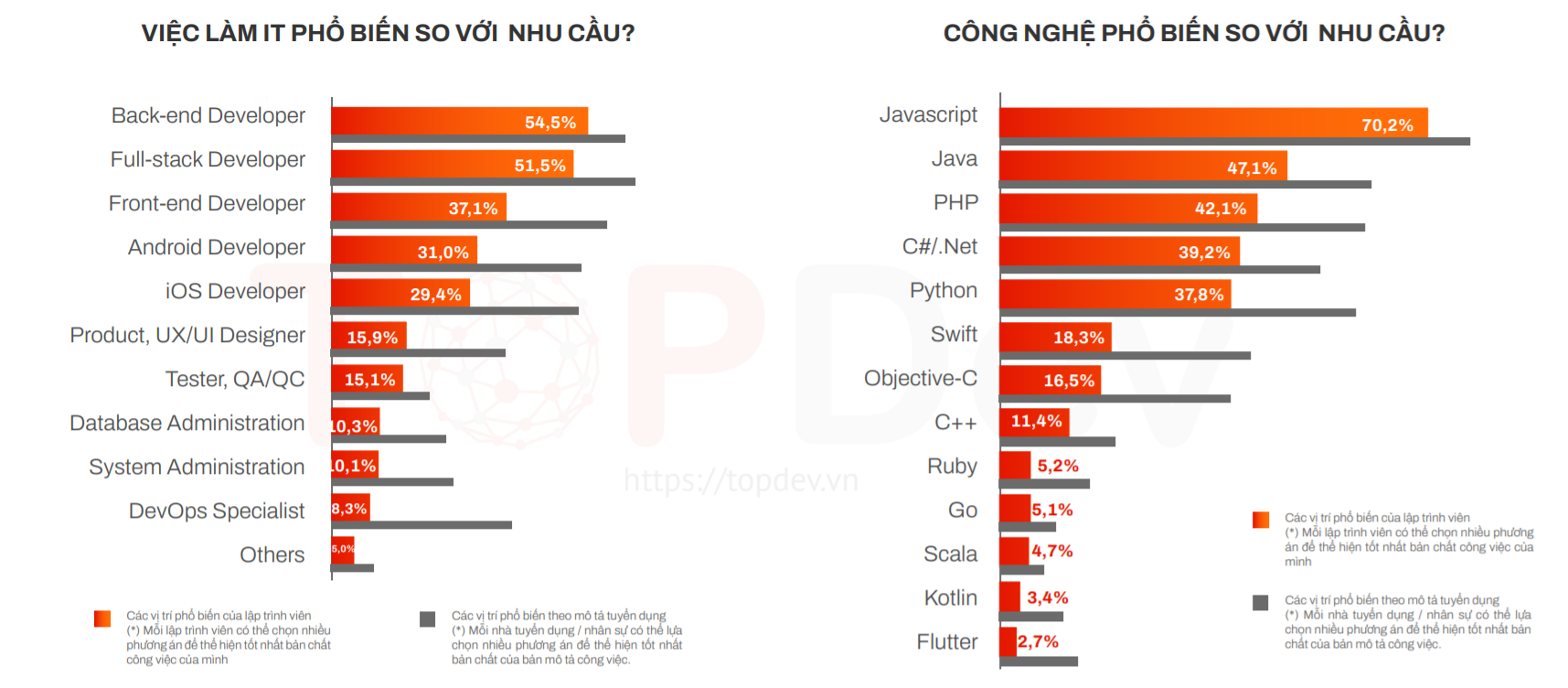
MVCRouteHandler có trách nhiệm gọi Controller Factory, sau đó nó sẽ tạo ra một thể hiện của Controller được ghi trong Route. Controller sẽ được nhận và nó sẽ gọi một Action Memthod và tạo ra View. Vậy là hoàn thành request.

*Chúng ta cần đăng ký Route sử dụng phương thức MapRoute của Routing Middleware. Routing Engine sau đó sẽ match URL đến với Route Collection và tìm ra một Route match với nó. Nó sẽ gọi RouteHandler (mặc định là MvcRouteHandler). MvcRouteHandler sau đó gọi Controller tương ứng trong Route.*

## 1.3 Hiệu năng của ASP.NET Core

Hiệu năng của ASP.NET Core so với các ngôn ngữ lập trình back-end khác có quan nhiều đến loại thao tác được thực hiện trong các bài kiểm tra. Mặc dù liên ASP.NET Core là ngôn ngữ khá quen thuộc và hầu như lâu đời sánh vai cùng với các ngôn ngữ khác, nhưng có một sự thật là ASP.NET Core cũng không quá vượt trội. ASP.NET Core sẽ phù hợp với các ứng dụng thường xuyên truy cập đến cơ sở dữ liệu. Đó là lý do tại sao mà nó đã được nhiều công ty áp dụng cho ứng dụng có số lượng người dùng lớn.

Để phân tích hiệu suất, chúng ta sẽ thực hiện các bài kiểm tra giữa ASP.NET Core và các ngôn ngữ lập trình phổ biến trong gần 3 thập kỷ nay. Bài kiểm tra này được thực hiện bởi TopDev về ASP.NET Core để thống kê khảo sát của mọi người để đưa ra kết luận của họ vào năm 2022. TopDev có hai nhóm cùng xây dựng một chức năng, một nhóm dựa trên vị trí việc làm của ngành IT theo nhu cầu và nhóm còn lại dựa trên công nghệ phổ biến theo nhu cầu. Kết quả của bài kiểm tra được thể hiện qua hai biểu đồ sau:



*Hình 1.3 Khảo sát việc làm và top ngôn ngữ phổ biến trong năm 2022*

Dù là bất cứ ngôn ngữ nào thì cũng sẽ có những thế mạnh nhất định. Công nghệ là luôn luôn đổi mới, sẽ luôn luôn phát triển nên đòi hỏi người học/người làm cũng thường xuyên cập nhật để phát triển theo.

## 1.4 Lý do sử dụng ASP.NET Core

***Hỗ trợ đa nền tảng & container***

Với việc giới thiệu .NET Core, giờ đây bạn có thể tạo các ứng dụng ASP.NET và triển khai chúng lên Windows, Linux và macOS, Microsoft và cộng đồng đã nỗ lực rất nhiều để biến Linux trở thành công dân hạng nhất để chạy ASP.NET.

Những container đang ăn những đám mây những ngày này. Docker, Kuberenetes và các công nghệ khác là tất cả các cơn thịnh nộ. ASP.NET Core cho phép các nhà phát triển sử dụng tất cả các công nghệ mới này. Microsoft Azure thậm chí còn hỗ trợ triển khai ứng dụng của bạn lên các thùng chứa và Kubernetes.

***Hiệu suất cao***

Một số người nói rằng hiệu suất là một tính năng quan trọng của phần mềm của bạn. Tôi có xu hướng đồng ý! Với sự ra đời của ASP.NET Core và máy chủ web Kestrel, ASP.NET được quảng cáo là một trong những khung ứng dụng web nhanh nhất hiện có.

 Công nghệ cung cấp đường ống tích hợp ASP.NET và IIS có tuổi đời khoảng 15 năm. Nó đã làm mọi thứ và mang theo rất nhiều hành lý. Máy chủ web Kestrel mới được thiết kế lại từ đầu để tận dụng các mô hình lập trình không đồng bộ, gọn nhẹ hơn và nhanh hơn!

***Không đồng bộ qua async/await***

ASP.NET có hỗ trợ tuyệt vời để sử dụng các mẫu lập trình không đồng bộ. Async hiện được triển khai trong tất cả các lớp .NET Framework phổ biến và hầu hết các thư viện của bên thứ ba. Hầu hết các ứng dụng hiện đại dành phần lớn thời gian và chu kỳ CPU của chúng để chờ truy vấn cơ sở dữ liệu, cuộc gọi dịch vụ web và các hoạt động I/O khác để hoàn thành.

Một trong những lý do ASP.NET Core nhanh hơn là việc sử dụng rộng rãi các mẫu không đồng bộ trong các khung MVC và Kestrel mới.

***Các khung API & Web API hợp nhất***

Trước ASP.NET Core, các nhà phát triển thường sử dụng các khung công tác MVC và Web API. MVC được thiết kế để tạo các ứng dụng web phục vụ HTML. API Web được thiết kế để tạo các dịch vụ RESTful bằng JSON hoặc XML.

Với ASP.NET Core, MVC và Web API đã được hợp nhất với nhau. Luôn luôn có rất nhiều sự trùng lặp với hai khung. MVC luôn có thể trả về dữ liệu JSON thay vì HTML. Kết hợp chúng là một động thái tốt và đơn giản hóa sự phát triển.

Với ASP.NET Core, chúng mở rộng khung MVC để cho phép đóng gói các khía cạnh của bộ điều khiển và mô hình của một trang cùng với ràng buộc hai chiều. Chúng là một sự thay thế cho WebForms trong khi sử dụng cú pháp Dao cạo quen thuộc.

***Nhiều môi trường và chế độ phát triển***

Nó cho phép lập trình viên dễ dàng phân biệt các phần của mã cho hành vi của họ trong quá trình phát triển, dàn dựng, sản xuất, v.v. Không có cách chuẩn nào để làm điều này trước ASP.NET Core.

***Tiêm phụ thuộc***

Một trong những tính năng mới tuyệt vời của ASP.NET Core được tích hợp trong phần phụ thuộc. Nó được sử dụng rất nhiều trong chính ASP.NET MVC. Đó là cách ưa thích mà những thứ như bối cảnh đăng nhập, bối cảnh cơ sở dữ liệu và những thứ khác được truyền vào bộ điều khiển MVC của lập trình viên.

***Bảo vệ giả mạo yêu cầu trên nhiều trang web (CSRF)***

ASP.NET Core có một khung tốt có sẵn để ngăn chặn các kiểu tấn công này. Nó tạo ra các mã thông báo chống giả mạo.

***Tự lưu trữ Ứng dụng Web***

Bạn có thể tạo một ứng dụng web ASP.NET tự lưu trữ theo nhiều cách khác nhau. Trong .NET 4.5 bạn có thể thực hiện nó bằng cách sử dụng Owin, Nancy hoặc WCF. Đối với Tiền tố, chúng tôi sử dụng API Web ASP.NET với Owin.

Với ASP.NET Core, lập trình viên cũng có thể sử dụng máy chủ web Kestrel tiêu chuẩn. Một trong những lợi thế lớn của .NET Core là ứng dụng web của lập trình viên về cơ bản là một ứng dụng console. IIS chỉ ngồi trước nó như một proxy ngược. Điều này có nghĩa là bạn cũng có thể triển khai ứng dụng của mình chỉ với kestrel cho các trường hợp sử dụng không dựa trên máy chủ, như Tiền tố.

***Bộ lọc hành động***

Một trong những tính năng tuyệt vời của ASP.NET là hỗ trợ các bộ lọc mở rộng. Điều này cho phép bạn thực hiện chức năng có thể được áp dụng cho toàn bộ bộ điều khiển hoặc hành động mà không cần sửa đổi chính hành động đó.

Các bộ lọc được sử dụng để chỉ định bộ đệm, xử lý lỗi, ủy quyền hoặc bất kỳ logic tùy chỉnh nào lập trình viên muốn triển khai.

***Bộ nhớ đệm đầu ra mở rộng***

Tính năng này cho phép ASP.NET lưu trữ đầu ra được tạo bởi một trang và phục vụ nội dung được lưu trong bộ nhớ cache này cho các yêu cầu trong tương lai. Nó lưu trữ dữ liệu không được cập nhật thường xuyên và xuất dữ liệu cụ thể đó từ một vị trí được lưu trữ.

ASP.NET giúp dễ dàng xác định thời gian bất kỳ yêu cầu nào sẽ được lưu trong bộ đệm thông qua các tiêu đề HTTP phổ biến. Nó cũng có hỗ trợ cho bộ nhớ đệm đầu ra trong bộ nhớ trên chính máy chủ web của bạn. Bạn thậm chí có thể sử dụng Redis hoặc các nhà cung cấp khác để xử lý bộ đệm ẩn đầu ra.

***Toàn cầu hóa và nội địa hóa***

ASP.NET giúp dễ dàng bản địa hóa ngày, số và văn bản trong ứng dụng web của lập trình viên. Nếu lập trình viên muốn ứng dụng của mình được sử dụng trên toàn cầu, việc bản địa hóa sẽ rất quan trọng đối với lập trình viên.

ASP.NET cho phép tùy chỉnh ứng dụng của bạn cho nhiều ngôn ngữ thông qua các tệp tài nguyên. Các tệp tài nguyên này được coi là kho lưu trữ trung tâm nơi lưu giữ tất cả các văn bản và các trang web có thể đọc tệp tài nguyên này và nhận nhãn. Có hai loại tài nguyên:

* Tài nguyên cục bộ - cụ thể cho một trang (nghĩa là sẽ có tệp tài nguyên cục bộ cho mỗi trang).
* Tài nguyên toàn cầu - chung cho toàn bộ trang web (nghĩa là một tệp tài nguyên được truy cập bởi tất cả các trang).

***OpenAPI vênh vang***

Nếu lập trình viên đang tạo các ứng dụng API, lập trình viên muốn chắc chắn rằng họ đang sử dụng Swagger. Nó giúp dễ dàng ghi lại và kiểm tra API của bạn.

ASP.NET có lịch sử cung cấp chức năng tích hợp khá giống với các dịch vụ web SOAP được tạo bằng WCF. Nếu bạn đang sử dụng API Web hoặc MVC cho API RESTful, lập trình viên chắc chắn muốn sử dụng Swagger.

*Cộng đồng lập trình viên rộng lớn*

Là một công nghệ phổ biến, ASP.ENT Core có một cộng đồng đông đảo lập trình viên trên toàn thế giới đang sử dụng. Điều này sẽ giúp quá trình tìm hiểu và sử dụng dễ dàng hơn bao giờ hết.

*Triển khai ứng dụng đơn giản và nhanh chóng*

Việc triển khai một ứng dụng Node.js lên các dịch vụ hỗ trợ hosting rất đơn giản. Ví dụ như somee.com hoặc hosting WINDOWS.

## 1.5 Sử dụng ASP.NET Core trong việc xây dựng web UI, web API và phát triển Client-side

**Xây dựng web UI và web API**

* Chúng ta có thể tạo ra các ứng dụng web có thể testing theo mô hình MVC(Model-View-Controller).
* Chúng ta có thể xây dựng HTTP services cái mà hỗ trợ nhiều định dạng và có đầy đủ hỗ trợ cho nội dung của dữ liệu trả về.
* Razor cung cấp một ngôn ngữ hiệu quả để tạo Views.
* Tag Helper cho phép code server side tham gia vào việc tạo và render phần tử HTML.
* Model Binding tự động ánh xạ dữ liệu từ HTTP request tới tham số của method action.
* Model Validation tự động thực hiện validate client và server.

*Phát triển client-site*

ASP.NET Core được thiết kế để tích hợp một cách liên tục với nhiều client-side frameworks, bao gồm HTML, CSS, Javascript và Bootstrap, cao cấp hơn là ReactJS, AngularJS, KnockoutJS, vv…

# CHƯƠNG 2. TÌM HIỂU VÀ SỬ DỤNG MICROSFT SQLSERVER

## 2.1 Tổng quan về cơ sở dữ liệu

**Cơ sở dữ liệu là gì ?**

Một cơ sở dữ liệu (CSDL) là một tập hợp các dữ liệu có liên quan với nhau, được lưu trữ trên máy tính, có nhiều người sử dụng và được tổ chức theo một mô hình. Hay nói cách khác, CSDL là một bộ các dữ liệu tác nghiệp được lưu trữ lại và được các hệ ứng dụng của một đơn vị cụ thể nào đó sử dụng.

**Mô hình cơ sở dữ liệu**

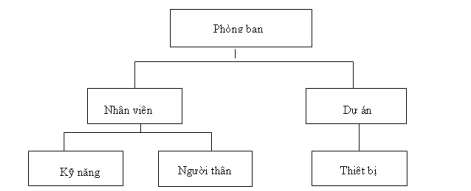
Mô hình dữ liệu là một khuôn dạng của dữ liệu cho phép người dùng nhìn thấy dữ liệu dưới cấu trúc thuật ngữ để diễn tả mà ta gọi là lược đồ (scheme). Nó cho ta biết cấu trúc của cơ sở dữ liệu, bao gồm hai thành phần:

* Hệ thống ký hiệu để mô tả dữ liệu.
* Tập hợp các phép toán thao tác trên dữ liệu đó.

*Mô hình phân cấp (Hierarchical model)*

Mô hình phân cấp được đưa ra vào những năm 60, trong mô hình này dữ liệu được tổ chức thành cấu trúc cây. Trong đó các nút (node) là tập các thực thể, các cành (edge) là các mối quan hệ giữa hai nút theo mối quan hệ nhẩt định, cứng nhắc. Hay nói cách khác:

* Là mô hình dữ liệu trong đó các bản ghi được sắp xếp theo cấu trúc top-down (tree).
* Một nút con chỉ có một nút cha -> chỉ có một đường truy nhập tới dữ liệu đó trước. Tập dữ liệu được tổ chức theo cấu trúc của mô hình dữ liệu phân cấp gọi là CSDL phân cấp.

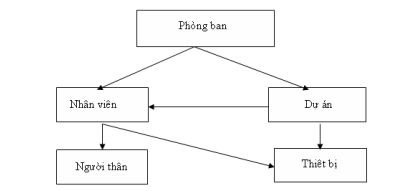


*Hình 2.1.1 Mô hình phân cấp (Hierarchical model)*

***Mô hình mạng (Network model)***

Mô hình mạng được sử dụng phổ biến vào cuối những năm 60, và được định nghĩa lại vào năm 1971. Trong mô hình này dữ liệu được tổ chức thành một đồ thị có hướng, trong đó các đỉnh là các thực thể, các cung là quan hệ giữa hai đỉnh, một kiểu bản ghi có thể liên kết với nhiều kiểu bản ghi khác.

Một con có thể có nhiều cha dẫn tới có thể có nhiều đường truy nhập đến một dữ liệu cho trước. Tập dữ liệu được tổ chức theo cấu trúc của mô hình dữ liệu mạng gọi là CSDL mạng.



*Hình 2.1.2* ***Mô hình mạng (Network model)***

Ưu điểm :

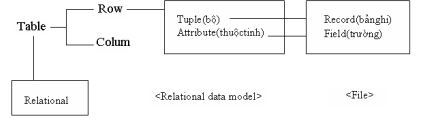
* Đơn giản.
* Có thể biểu diễn các ngữ nghĩa đa dạng với kiểu bản ghi và kiểu móc nối.
* Truy vấn thông qua phép duyệt đồ thị (navigation).

Nhược điểm :

* Số lượng các con trỏ lớn.
* Hạn chế trong biểu diễn ngữ nghĩa của các móc nối giữa các bản ghi.

***Mô hình dữ liệu quan hệ (Relational model)***

Mô hình này đượcc E.F Codd đưa vào đầu những năm 70, mô hình này dựa trên lý thuyết tập hợp và đại số quan hệ. Vì tính chất chặt chẽ của toán học về lí thuyết tập hợp nên mô hình này đã mô tả dữ liệu một cách rõ ràng, mềm dẻo và là mô hình thông dụng nhất hiện nay. Hầu hết các DBMS đều tổ chức dữ liệu theo mô hình dữ liệu quan hệ. Trong đó dữ liệu được tổ chức dưới dạng bảng, các phép toán thao tác trên dữ liệu dựa trên lý thuyết tập hợp của toán học. Tập dữ liệu được tổ chức theo cấu trúc của mô hình dữ liệu quan hệ thì được gọi là CSDL quan hệ.



*Hình 2.1.3* ***Mô hình dữ liệu quan hệ (Relational model)***

Ưu điểm :

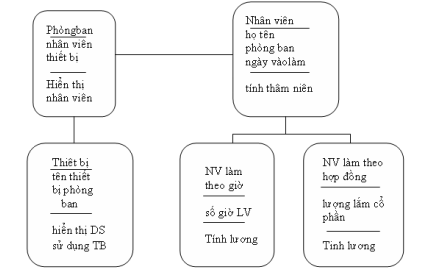
* Dựa trên lý thuyết tập hợp.
* Khả năng tối ưu hoá các xử lý phong phú.

Nhược điểm :

* Hạn chế trong biểu diễn ngữ nghĩa.
* Cấu trúc dữ liệu không linh hoạt.

***Mô hình dữ liệu hướng đối tượng (Object Oriented model)***

Là mô hình dữ liệu trong đó các thuộc tính dữ liệu và các phương thức thao tác trên các thuộc tính đó đều được đóng gói trong các cấu trúc gọi là đối tượng (object). Tập dữ liệu được tổ chức theo cấu trúc của mô hình dữ liệu hướng đối tượng gọi là CSDL hướng đối tượng.



*Hình 2.1.4* ***Mô hình dữ liệu hướng đối tượng***

Ưu điểm :

* Dễ dàng biểu diễn cái mà con người nhận thức từ thế giới thực.
* Biểu diễn ngữ nghĩa phong phú của các thực thể và quan hệ giữa các thực thể.

Nhược điểm :

* Không dễ dàng ánh xạ vào những cấu trúc lưu trữ trên máy tính.

**Ưu điểm của cơ sở dữ liệu**

* Giảm sự trùng lặp thông tin xuống mức tối thiểu nhất. Do đó đảm bảo thông tin có tính nhất quán và toàn vẹn dữ liệu.
* Đảm bảo dữ liệu có thể được truy xuất theo nhiều cách khác nhau.
* Nhiều người có thể sử dụng chung một cơ sở dữ liệu.

**Nhược điểm của cơ sở dữ liệu**

* Tính chủ quyền của dữ liệu.
* Tính bảo mật và quyền khai thác thông tin của người sử dụng.
* Tranh chấp dữ liệu.
* Cần đảm bảo an toàn dữ liệu khi có sự cố.

## 2.2 Tổng quan về SQL

**SQL là tên gọi tắt của cụm từ Structured Query Language**, là ngôn ngữ truy vấn có cấu trúc. Nó là một ngôn ngữ, là tập hợp các lệnh để tương tác với cơ sở dữ liệu. Dùng để lưu trữ, thao tác và truy xuất dữ liệu được lưu trữ trong một cơ sở dữ liệu quan hệ. Trong thực tế, SQL là ngôn ngữ chuẩn được sử dụng hầu hết cho hệ cơ sở dữ liệu quan hệ. Tất cả các hệ thống quản lý cơ sở dữ liệu quan hệ (RDMS) như MySQL, MS Access, Oracle, Postgres và SQL Server… đều sử dụng SQL làm ngôn ngữ cơ sở dữ liệu chuẩn.

Hầu như công ty nào lớn cũng cần xây dựng một hệ thống để lưu trữ cơ sở dữ liệu. Mọi thứ trong cơ sở dữ liệu này sẽ được diễn tả ra thành nhiều bảng, có mối quan hệ với nhau. Để truy vấn và lấy dữ liệu từ các bảng này nhằm tổng hợp thành thông tin nào đó, người ta dùng đến SQL thông qua các câu query.

### 2.2.1 Nguồn gốc

**SQL** bắt đầu từ những năm 1970, khi các kỹ sư của **IBM** là **Donald Chamberlin** và **Raymond Boyce** thiết kế phiên bản đầu tiên để tương tác và lấy dữ liệu được lưu trữ trong hệ thống cơ sở dữ liệu của công ty. Họ gọi nó là SEQUEL, mặc dù sau đó họ buộc phải thay đổi nó do các vấn đề bản quyền.

### 2.2.2 Tại sao sử dụng SQL và nó lại quan trọng

**SQL** được sử dụng phổ biến vì nó có các ưu điểm sau:

* Cho phép truy cập dữ liệu trong các hệ thống quản lý cơ sở dữ liệu quan hệ.
* Cho phép mô tả dữ liệu.
* Cho phép xác định dữ liệu trong cơ sở dữ liệu và thao tác dữ liệu đó.
* Cho phép nhúng trong các ngôn ngữ khác sử dụng mô-đun SQL, thư viện và trình biên dịch trước.
* Cho phép tạo và thả các cơ sở dữ liệu và bảng.
* Cho phép tạo chế độ view, thủ tục lưu trữ, chức năng trong cơ sở dữ liệu.
* Cho phép thiết lập quyền trên các bảng, thủ tục và view.

### 2.2.3 ****Con đường sự nghiệp cho SQL Developer nói riêng và Database Developer****

Hiện tại có 2 hướng phát triển như sau:

* Hướng đi thiên về kỹ thuật: có thể trở thành Data Architect – là người thiết kế ra cơ sở dữ liệu cho doanh nghiệp. Hiện nay, Data Scientist cũng là vị trí đang rất cần trong thời đại AI & Bigdata. Ngoài kiến thức về SQL, lập trình viên còn phải biết về xác suất thống kê và ngôn ngữ lập trình khác, thường là Python.
* Hướng đi về quản lý: đích đến sẽ là Business Analyst hoặc Project Manager.

## 2.3 Lý do sử dụng SQL

*Khai thác dữ liệu*

Biết sử dụng SQL sẽ cho phép lập trình viên khai thác dữ liệu với hiệu quả cao hơn. Chỉ cần sử dụng các truy vấn cơ bản là bạn có thể xác định được dữ liệu cụ thể theo yêu cầu của người dùng.

*Nhu cầu lập trình viên SQL cao*

Khi bạn là backend developer, khả năng sử dụng SQL là yêu cầu bắt buộc. Cho nên biết sử dụng và sử dụng thành thạo để tối ưu query db là một ưu điểm cực kỳ tốt.

*Dữ liệu có ở mọi nơi*

Mọi ứng dụng trên internet để cần lưu trữ dữ liệu. Ngay cả các ứng dụng desktop, laptop và mobile cũng cần phải lưu dữ liệu. Khi bạn đăng ký tài khoản trên ứng dụng web, lập trình viên đang cung cấp thông tin cho ứng dụng.

*Thao tác với dữ liệu dễ dàng*

Với [SQL](https://topdev.vn/viec-lam-it/sql-kt28), việc muốn thêm, sửa, đọc hay xóa dữ liệu trở lên dễ dàng hơn bao giờ hết. Ví dụ, nếu các lập trình viên muốn lấy thông tin người dùng với email chứa gmail trong đó bạn có thể chạy một câu lệnh duy nhất sau với MySQL.

mysql> select \* from users where email like '%gmail%'

*SQL được sử dụng và hỗ trợ bởi nhiều công ty lớn*

Mặc dù SQL là ngôn ngữ khá đơn giản, nhưng nắm vững được nó là một kỹ năng là một điều vô cùng cần thiết.

Tất cả các công ty lớn về công nghệ trên thế giới hiện nay như Microsoft, IBM, Oracle… đều hỗ trợ việc phát triển ngôn ngữ này và SQL được rất nhiều công ty lớn sử dụng.

*SQL giúp công việc lập trình dễ dàng hơn*

Nếu không xài DB để lưu trữ dữ liệu, lập trình viên có thể lưu dữ liệu trên một file trên server. Tuy nhiên thử tưởng tượng trong ứng dụng của lập trình viên có rất nhiều loại thông tin cần lưu và bạn quyết định sử dụng nhiều file để lưu mỗi loại dữ liệu.

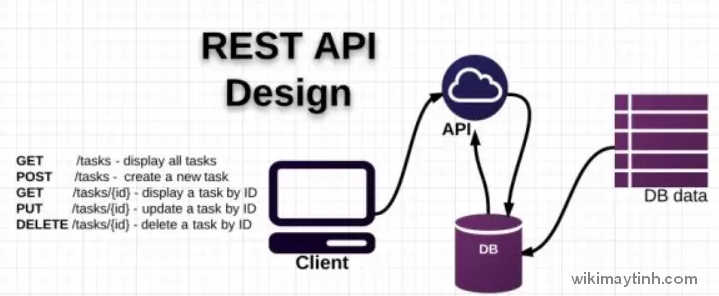
# CHƯƠNG 3. SỬ DỤNG API

## 3.1 Tổng quan về API

**API** là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của Application Programming Interface – giao diện lập trình ứng dụng. API cung cấp khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng.

**Restful api là gì ?**

* Restful API được hiểu là một tiêu chuẩn được sử dụng để thết kế API cho các ứng dụng website. Nhằm để quản lý các resource web một cách dễ dàng hơn. Có thể nói Restful là một trong những kiểu thiết kế API được các lập trình viên sử dụng khá phổ biến.
* Theo đó Rest quy định cách sử dụng http method như GET, POST, PUT, DELETE… và cách định dạng URL cho các ứng dụng web.
* API cũng có thể liên quan đến khung phần mềm : Khung có thể dựa trên một số thư viện triển khai một số API. Nhưng không giống như việc sử dụng API thông thường, quyền truy cập vào hành vi được xây dựng trong khung trung gian bằng cách mở rộng nội dung của nó với các lớp mới cắm vào khung chính nó.
* **Web API** là một trong những công nghệ mới của Microsoft dùng để xây dựng dịch vụ thành phần phân tán. Web API là mô hình dùng để hỗ trợ MVC bao gồm: routing, controller, action result, filter, loc container, model binder, unit test, injection. Bên cạnh đó nó còn hỗ trợ restful đầy đủ các phương thức: Get/Post/put/delete dữ liệu.



*Hình 3.1 Hình minh họa rest api*

**Ưu điểm của API**

* Kết nối mọi lúc nhờ vào Internet.
* Giao tiếp hai chiều phải được xác nhận trong các giao dịch.
* Vì giao tiếp là API hai chiều nên thông tin rất đáng tin cậy.
* Cung cấp cấp trải nghiệm thân thiện với người.
* Cung cấp giải pháp phát triển khi các nhà phát triển tìm thấy cách sử dụng mới để trao đổi API.
* Cấu hình đơn giản khi được so sánh với WCF.
* Mã nguồn mở.
* Hỗ trợ chức năng RESTful một cách đầy đủ.
* Hỗ trợ đầy đủ các thành phần MVC như: routing, controller, action result, filter, model binder, IoC container, dependency injection, unit test.
* Khả năng trình diễn cao.

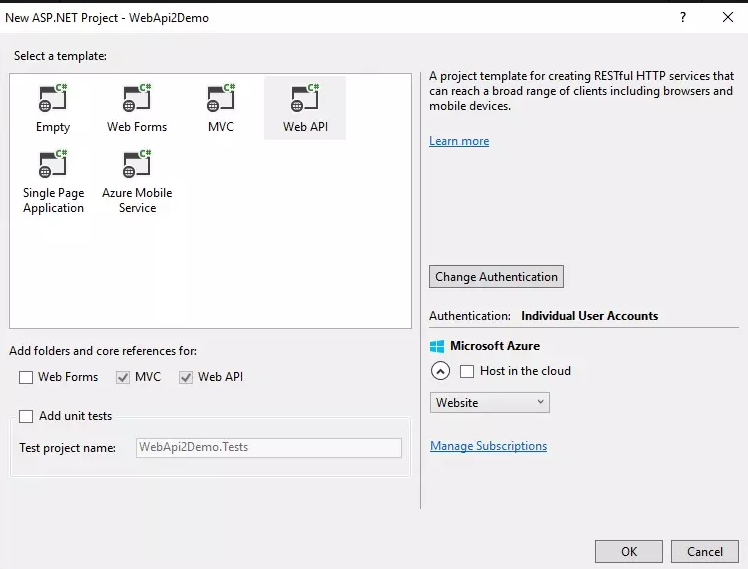
## 3.2 Cấu trúc của một ứng dụng Web API

Công nghệ sử dụng:

* Visual studio 2019/Visual Studio 2022
* .Net core 3.1 trở lên
* Web Api 2.0
* Entity framework 6.0
* Knockout js hoặc các framework khác nếu có

Tạo project

* Mở Visual studio
* Chọn New project
* Tại cửa sổ dialog chọn [Asp.net](http://asp.net/) web applicaltion
* Tiếp đến chọn Web API

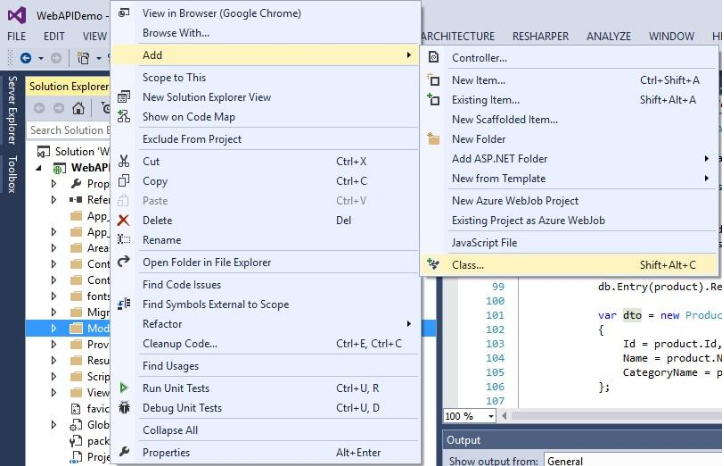


*Hình 3.2.1.1 chọn Web Api để khởi tạo project*

*Tạo data models*

Model là tập hợp các đối tượng chứa dữ liệu của ứng dụng có thể chứa thêm cả các logic nữa. Model chia làm một số loại dựa trên công dụng và nơi chúng sử dụng. Có 3 loại mục đích chính:

* Domain Model
* View Model
* Edit Model

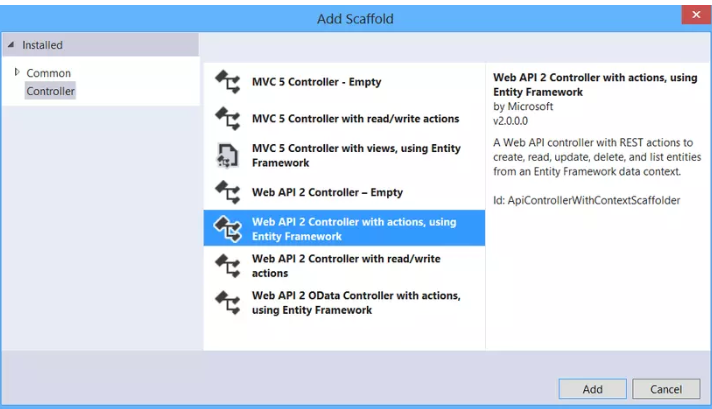
****

*Hình 3.2.1.2 Tạo dữ liệu(Data models)*

Ví dụ hai lớp Category và Product

*Tạo controllers*

* Click chuột phải tại thư mục Controller , chọn "Add" , chọn "Controller"
* Tại cửa sổ "Add Scaffold" chúng ta chọn "Web API 2 Controller with actions, using Entity Framework".

**

*Hình 3.2.1.3 Tạo controller*

Tại cửa sổ "Add controller"

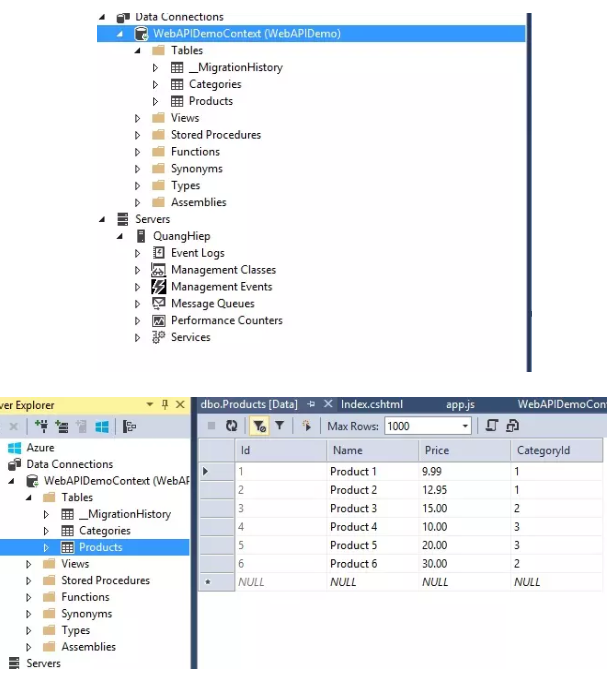
* Tại Model class dropdownlist, chọn Category class.
* Tích chọn “Use async controller actions”.
* Trong mục "Controller Name" tên controller tự động được điền vào.

***Làm việc với Entity framework***

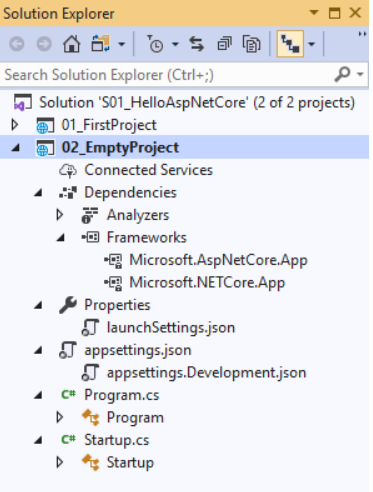
Trong demo này tôi sử dụng Code First Migrations để init data và thao tác với Sql server. Trước tiên cần enable migration, để làm điều này chúng ta thực hiện như sau: Chọn Tool, chọn Nuget Package Manager , chọn Package Manage Console.

Trong cửa sổ Package Manage Console gõ lệnh “Add-Mgration AdddemoTodatabase”, sau đó gõ tiếp “Update-database” và cuối cùng một file migarion được tạo ra với code được tự động generate.

Cuối cùng sau khi code first thành công ta có một database hoàn chỉnh:



*Hình 3.2.1.4 Database sau khi code first*

**

*Hình 3.2.1.5 Cấu trúc project*

Trong đó:

* Connected service: danh sách các dịch vụ online mà project sử dụng, ví dụ WFC Web Service, các dịch vụ đám mây Azure,…
* Dependencies: danh sách các gói nuget package.
* Properties: chứa các thông tin cấu hình của project.
* Appsettings.json: cấu hình các hoạt động của ứng dụng, như connection string của database.
* Program.cs: cấu hình infractrucure cho ứng dụng.
* Startup.cs: chứa các phương thức cấu hình của ứng dụng, ví dụ sử dụng các middleware trong pipeline, Dependency Injection, Logging.
* wwwroot: chứa CSS, image, Javascript.

## 3.3 Một số nội dung chính trong ASP.NET Rest Api

### 3.3.1 Cơ chế Model Binding: Truyền dữ liệu từ View lên Controller

Trong cơ chế **Model Binding** của ASP.NET Core chúng ta sẽ học cách làm sao để truyền dữ liệu từ View lên Controller. Chúng ta cũng sẽ tìm hiểu về **Model Binding** và cơ chế hoạt động của nó. ASP.NET Core cho phép chúng ta bind dữ liệu từ nhiều nguồn khác nhau như HTML Form sử dụng **[FromForm]**, từ giá trị route **[FromRoute]**, từ query string **[FromQuery]**, từ  body của request**[FromBody]** và từ Header của request **[FromHeader].**

*Model Binding*

Model Binding là cơ chế map dữ liệu được gửi qua HTTP Request vào các tham số của action method trong Controller. HTTP Request có thể chứa dữ liệu từ nhiều định dạng. Dữ liệu có thể chứa trong HTML Form. Nó có thể là một phần của route value hoặc trên query string hay có thể là một body của request.

Cơ chế ASP.NET Core model binding cho phép chúng ta dễ dàng bind các giá trị này vào các tham số của action method. Các tham số này có thể là kiểu nguyên thủy hoặc kiểu đối tượng phức tạp.

*Lấy dữ liệu từ Form Data trong Controller*

Chúng ta đã tạo một form cơ bản cho phép nhận một đối tượng Data. Khi người dùng click nút **Submit**thì dữ liệu sẽ được post lên phương thức **Create**trên Controller. Trong project đó chúng ta cũng tạo một **DataEditModel**class chứa chi tiết của sản phẩm cần được tạo hoặc chỉnh sửa.

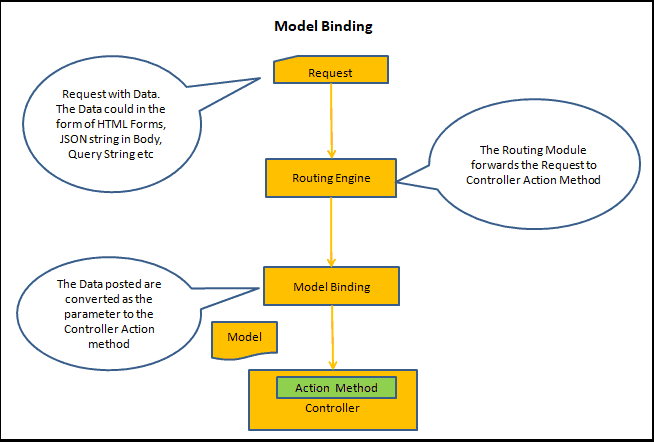
Sau đó ta tạo một form gồm các *field* cần thiết để hiển thị dữ liệu. Kế tiếp ta tạo phương thức method cho **DataController**.

Một submit của form trên, các giá trị trong form sẽ tự động được map vào đối tượng **DataEditModel**trong Action method của controller:

public IActionResult **Create**(DataEditModel **model**)

Cơ chế này tự động xảy ra đằng sau được gọi là Model Binding. Model Binder sẽ tìm các trường tương ứng giữa tham số **ProductEditModel**với trường trong form, route value hoặc query string...

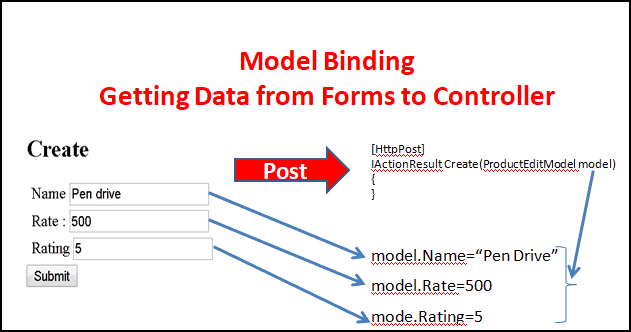
*Cách làm việc của Cơ chế Model Binding*



*Hình 3.3.1.1 Cơ chế làm việc của Model binding*

Khi người dùng click vào nút Submit thì một request Post được gửi lên server với Form Data, QueryString, Route Parameter... MVCRouteHandler của Routing Engine sẽ xử lý request đến và có trách nhiệm gọi action method tương ứng. Model Bindler sẽ được kích hoạt trước khi action method được gọi. Nó tìm dữ liệu thỏa mãn trong form data, querys tring và request parameter trong HTTP Request. Sau đso nó sẽ binding các giá trị vào tham số của action method qua tên.

Ví dụ, trường "**name**" trong form sẽ được map vào thuộc tính "**Name**" trong **ProductEditModel**. Rate trong form sẽ được map vào thuộc tính Rate...



*Hình 3.3.1.2 Map thuộc tính data của Model binding*

Để Model binding làm việc đúng:

* Thuộc tính Name phải match với Request Data
* Cấc thuộc tính phải đặt public set

*Model Binder*

Model Binder có trách nhiệm gán dữ liệu vào các tham số của action method. Model Binder được tạo mởi model binder provider. Model binder phải được implement inteface **IModelBinderProvider**. Nghĩa là bạn có thể tạo một Model Binder của riêng mình hoặc mở rộng nó bằng cách triển khai inteface **IModelBinderProvider**. Custom model binder phải được đăng ký trong **ModelBinderProviders**trong **Startup.cs**.

services.AddMvc(options =>

{

options.ModelBinderProviders.Add(new CustomModelBinderProvider());

});

*ModelState*

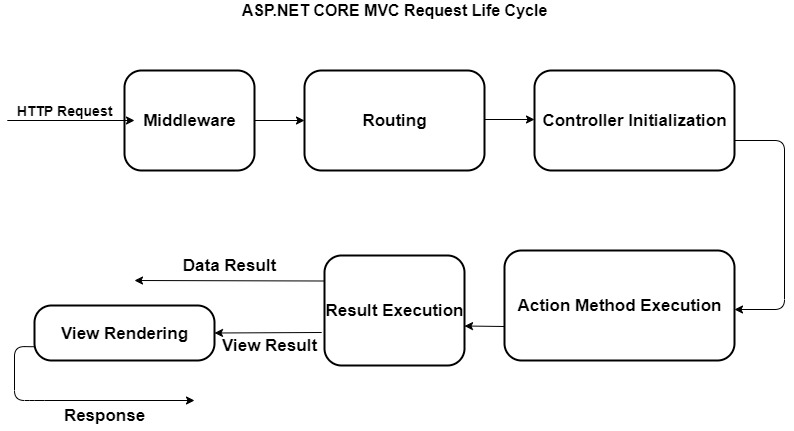
Nếu Model binder không thành công trong việc bind dữ liệu từ Request vào thuộc tính model tương ứng, nó sẽ không đưa ra bất cứ thông báo lỗi nào. Nhưng nó sẽ update đối tượng ModelState với danh sách lỗi và set thuộc tính IsValid là false.

Vì thế kiểm tra **ModelState.IsValid** sẽ cho chúng ta thấy quá trình binding có thành công hay không.

Ví dụ: Trong ví dụ trên khi click nút **submit**mà không nhập bất cứ dữ liệu gì trên form thì kết quả sẽ ra validate thất bại và vì thế **ModelState.IsValid** sẽ là **false**.

### 3.3.2 Request Life Cycle trong ASP.NET Core

Requets life cycle ( vòng đời request ) trong ASP.NET Core MVC là một chuỗi những sự kiện, quá trình hoặc những thành phần tương tác với nhau để giải quyết và xử lý HTTP request và trả về response cho client .



*Hình 3.3.2.1 Biểu đồ thể hiện phương thức trong một Request Life Cycles*

*Request Pipeline*

Request Pipeline ( đường ống request ) là chính sách được mở màn từ khi request được giải quyết và xử lý tới khi trả về một response. Request đến từ trình duyệt đi qua request pipeline và quay trở lại khi giải quyết và xử lý xong để trả về response cho client. Các thành phần đơn lẻ tạo nên request pipeline này được gọi là middleware .

*Middleware*

Middleware là thành phần tạo thành request pipeline để giải quyết và xử lý request và tạo ra response phản hồi lại client. Mỗi một middleware thao tác với những request nhận được từ middleware trước nó. Nó cũng hoàn toàn có thể quyết định hành động gọi middleware tiếp theo trong pipeline hoặc trả về response cho middleware ngay trước nó .

*Routing*

Routing là một middleware được tiến hành trong MVC framework. Routing middleware quyết định hành động cách một request được gửi đến hoàn toàn có thể được ánh xạ tới Controller và Action Method nào nhờ sự trợ giúp của những thông số kỹ thuật tuyến .

*Khởi tạo Controller*

Ở quá trình này của request lifecycle trong ASP.NET MVC Core, quy trình khởi tạo và thực thi Controller diễn ra. Controller có nghĩa vụ và trách nhiệm giải quyết và xử lý những request được gửi đến. Controller lựa chọn Action Method thích hợp trên cơ sở những mẫu định tuyến được phân phối .

*Thực thi Action Method*

Sau khi Controller được khởi tạo, Action Method được thực thi và trả về Result ( tác dụng ).

*Thực thi Result*

Trong quá trình này của request lifecycle trong ASP.NET MVC Core, Result tức là response được tạo cho request HTTP bắt đầu, sẽ được thực thi. Nếu Action Method trả về có dạng ViewResult, MVC view engine sẽ render và trả về phản hồi HTML. Nếu tác dụng không phải là ViewResult thì Action Method sẽ tạo ra phản hồi của riêng nó.

**3.4 Lý do sử dụng Rest API để tích hợp đồ án ASP.NET Core**

Web API thường sở hữu vô số ưu điểm vượt trội tương tự như của REST API, nhờ vậy giúp nó nhận được nhiều hơn sự tin tưởng từ nhiều nhà phát triển và nhà lập trình website. Các ưu điểm nổi bật mà Web API sở hữu sẽ là:

*Tinh linh hoạt cao*

Dễ viết, dễ tin chỉnh, hơn so với framework WCF, Web service trước đây.

*Tính khả dụng*

Hỗ trợ nâng cao trải nghiệm người dùng nhờ UX rất thân thiện. Hỗ trợ tốt mọi tính năng cũng như thành phần của HTTP. Có khả năng bảo mật cao, xác nhận được 2 chiều ngay khi có request. Các dữ liệu trả về luôn ở nhiều định dạng khác nhau từ XML cho đến JSON. Sẽ Host được nhiều loại client khác nhau từ ứng dụng nền web cho đến những phần mềm của desktop.

*Cộng đồng hỗ trợ mạnh*

Dotnet Core hiện đang có một cộng đồng hỗ trợ mạnh trên github.Dotnet hiện đang có 29140 star đánh giá và hơn 1439 lượt xem, có hơn 1000 người thường xuyên đóng góp. Ngoài ra còn có cộng đồng, nhóm lớn trên facebook, Stackoverflow và Quora.

# CHƯƠNG 4. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 4.1 Xác định yêu cầu bài toán:

* Do nhu cầu của khách hàng ngày càng tăng và công nghệ thông tin ngày càng phát triển đặc biệt là thương mại điện tử.
* Website cửa hàng quần áo trực tuyên Electro Shopping ra đời nhằm mục đích đáp ứng nhu cầu đặt các trang phục, phụ kiện qua mạng ngày càng lớn của người dân và giúp cho các cửa hàng phát triển nhanh hơn, giúp cho chủ cửa hàng có thể quản lý cửa hàng của mình một cách tốt hơn.

## 4.2 Xác định các chức năng

### 4.2.1 các yêu cầu chức năng:

|  |  |  |
| --- | --- | --- |
| **STT** | **MÃ CHỨC NĂNG** | **TÊN CHỨC NĂNG** |
| 1 | MS01 | Đăng nhập |
| 2 | MS02 | Đăng xuất |
| 3 | MS03 | Quản lý loại sản phẩm |
| 4 | RQ04 | Quản lý sản phẩm |
| 5 | MS05 | Hiển thị phản hồi của khách hàng(chỉ có hiển thị tại client-site) |
| 6 | MS06 | Quản lý đơn hàng |
| 7 | MS07 | Quản lý khách hàng |
| 8 | MS08 | Xem chi tiết sản phẩm |
| 9 | MS09 | Tìm kiếm sản phẩm |
| 10 | MS10 | Xem chi tiết giỏ hàng |

### 4.2.2. Các yêu cầu phi chức năng

Yêu cầu hiệu năng

* Yêu cầu về độ thực thủ: Tốc độ xử lý và tính toán nhanh (Các thuật toán sắp xếp, tìm kiếm được thiết kế tối ưu).
* Yêu cầu về giao diện: Giao diện phải thân thiện với người dùng, để dùng thao tác.

Yêu cầu độ tin cậy

* Đáp ứng được nhu cầu của khách hàng
* Đảm bảo đáp ứng được các yêu cầu của người sử dụng. - Ôn định, thời gian làm việc không gặp lỗi quá lớn.
* Cho kết quả chính xác.
* Đảm bảo website có ít khiếm khuyết về lập trình
* Kiểm soát được các ngoại lệ.
* Lỗi logic tối thiểu.

Yêu cầu bảo mật

* Đảm bảo tính duy nhất của tên đăng nhập người sử dụng.
* Chỉ có người quản trị hệ thống mới có quyền cập nhật các thông tin về
* sản phẩm và người sử dụng.
* Chịu được tấn công của tin tặc và có thể khắc phục ngay về trạng thái
* trước khi bị tấn công.

## 4.3 Đặc tả các chức năng của hệ thống:

### 4.3.1 MS01 -Đăng nhập:

|  |  |
| --- | --- |
| Mã yêu cầu | MS01 |
| Tên yêu cầu | Đăng nhập |
| Đối tượng sử dụng | Tất cả các nhóm người dùng |
| Mực ưu tiên | cao |
| Tiền điều kiện | Không |
| Nội dung | Cho phép người dùng thực hiện đăng nhập vào hệ thống |
| Cách xử lý | Bước 1: Người dùng nhập tên đăng nhập và mật khẩu và nhấn đăng nhập  Bước 2: Hệ thống kiểm tra dữ liệu người dùng vừa nhập vào. Nếu thành công hệ thống chuyển đến trang làm việc tiếp theo, Ngược lại hệ thống gửi một thông báo đến người dùng. |
| Kết quả | Đăng nhập thành công hoặc thất bại |
| Ghi chú |  |

### 4.3.2 MS02 -Đăng xuất

|  |  |
| --- | --- |
| Mã yêu cầu | MS02 |
| Tên yêu cầu | Đăng xuất |
| Đối tượng sử dụng | Tất cả nhóm người dùng |
| Mực ưu tiên | cao |
| Tiền điều kiện | Người dùng đã đăng nhập thành công trước đó |
| Nội dung | Cho phép người dùng thực hiện đăng xuất ra khỏi hệ thống, kết thúc phiên làm việc |
| Cách xử lý | Bước 1: Người dùng chọn Đăng xuất  Bước 2: Hệ thống đăng xuất, chuyển người dùng trở lại trang chủ của website. Nếu người dùng đăng nhập vào trang chủ của trang quản lý thì chuyển lại giao diện đăng nhập |
| Kết quả | Đăng xuất thành công |
| Ghi chú |  |

### 4.3.3 MS03 -Quản lý loại sản phẩm:

|  |  |
| --- | --- |
| Mã yêu cầu | MS03 |
| Tên yêu cầu | Quản lý loại sản phẩm |
| Đối tượng sử dụng | Quản trị viên |
| Mực ưu tiên | cao |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Nội dung | Trong chức năng này người dùng có thể : thêm mới , cập nhật, xóa các loại sản phẩm. |
| Cách xử lý | *Thêm mới*  Bước 1: Người dùng nhập đầy đủ các loại thông tin: tên loại sản phẩm  Bước 2: Người dùng nhấp chọn nút “Thêm”  Bước 3: Hệ thống kiểm tra sự đầy đủ và hợp lệ của các thông tin người dùng vừa nhập vào. Nếu thông tin là đầy đủ và hợp lệ thì hệ thống tiến hành thêm thông tin vào CSDL và gửi thông báo “Thêm thành công" đến người dùng. Ngược lại nếu thông tin chưa đầy đủ hoặc không hợp lệ thì hệ thống gửi một thông bảo chính xác lỗi đến người dùng.  *Cập nhật*  Bước 1: Hệ thống hiển thị tất cả loại sản phẩm từ CSDL.  Bước 2: Người dùng tìm loại sản phẩm muốn cập nhật lại thông tin. Sau đó click icon “sửa"  Bước 3: Hệ thống chuyển người dùng lên đầu trang. Tại đây người dùng có thể cập nhật lại thông tin của loại sản phẩm vừa chọn ở Bước 2. Sau đó nhấn nút "Cập nhật"  Bước 4: Hệ thống kiểm tra sự đầy đủ và hợp lệ của các thông tin người dùng vừa mới cập nhật lại. Nếu thông tin là đầy đủ và hợp lệ thì hệ thống tiến hành cập nhật thông tin vào CSDL và gửi thông báo “Cập nhật thành công" đến người dùng. Ngược lại nếu thông tin chưa đầy đủ hoặc không hợp lệ thì hệ thống gửi một thông báo chính xác lỗi đến người dùng.  *Xóa*  Bước 1: Hệ thống hiển thị tất cả loại sản phẩm từ CSDL  Bước 2: Người dùng tìm loại sản phẩm muốn xóa  Bước 3: Người dùng nhấp chọn Xóa để tiến hành  xóa loại sản phẩm đã chọn.  Bước 4: Hệ thống hiển thị một thông báo Xác nhận xóa đến người dùng.  Bước 5: Nếu người dùng tiếp tục nhấn nút Ok. Hệ thống sẽ kiểm tra tính hợp lệ loại sản phẩm người dùng muốn xóa Nếu hợp lệ thì hệ thống tiến hành xóa bỏ hoàn toàn các thông tin của loại sản phẩm ra khỏi CSDL và thông báo “Xóa thành công” đến người dùng. Ngược lại nếu thông tin không hợp lệ thì hệ thống gửi một thông báo chính xác lỗi đến người dùng. |
| Kết quả | Thêm mới, cập nhật, xóa, thành công hoặc thất bại |
| Ghi chú | Bắt buộc phải có thông báo xác nhận khi người dùng chọn Xóa loại sản phẩm.  Tất cả các trưởng thông tin không được bỏ trống.  Hỗ trợ thêm chức năng Tìm kiếm loại sản phẩm để thuận tiện hơn cho người dùng. |

### 4.3.4 MS04 -Quản lý sản phẩm:

|  |  |
| --- | --- |
| Mã yêu cầu | MS04 |
| Tên yêu cầu | Quản lý sản phẩm |
| Đối tượng sử dụng | Quản trị viên |
| Mực ưu tiên | cao |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Nội dung | Trong chức năng này người dùng có thể : thêm mới , cập nhật, xóa các sản phẩm. |
| Cách xử lý | *Thêm mới*  Bước 1: Người dùng nhập đầy đủ vào các trường thông tin: Tên sản phẩm, Mô tả khái quát sản phẩm, Mô tả chi tiết sản phẩm, Giá sản phẩm, Giảm giá, Tên loại sản phẩm, Ảnh.  Bước 2: Người dùng nhấp chọn nút “Thêm”  Bước 3: Hệ thống kiểm tra sự đầy đủ và hợp lệ của các thông tin người dùng vừa nhập vào. Nếu thông tin là đầy đủ và hợp lệ thì hệ thống tiến hành thêm thông tin vào CSDL và gửi thông báo “Thêm thành công” đến người dùng. Ngược lại nếu thông tin chưa đầy đủ hoặc không hợp lệ thì hệ thống gửi một thông báo chính xác lỗi đến người dùng.  *Cập nhật*  Bước 1: Hệ thống hiển thị tất cả sản phẩm từ CSDL.  Bước 2: Người dùng tìm sản phẩm muốn cập nhật lại thông tin. Sau đó click “chọn"  - Bước 3: Hệ thống chuyển người dùng lên đầu trang. Tại đây người dùng có thể cập nhật lại thông tin của sản phẩm vừa chọn ở Bước 2.  Bước 4: Hệ thống kiểm tra sự đầy đủ và hợp lệ của các thông tin người dùng vừa mới cập nhật lại. Nếu thông tin là đầy đủ và hợp lệ thì hệ thống tiến hành cập nhật thông tin vào CSDL và gửi thông báo "Cập nhật thành công” đến người dùng. Ngược lại nếu thông tin chưa đầy đủ hoặc không hợp lệ thì hệ thống gửi một thông báo chính xác lỗi đến người dùng.  Xóa  Bước 1: Hệ thống hiển thị tất cả sản phẩm từ CSDL  Bước 2: Người dùng tìm sản phẩm muốn xóa  Bước 3: Người dùng nhấp chọn “Xóa" để tiến hành xóa sản phẩm đã chọn. Bước 4: Hệ thống hiển thị một thông báo Xác nhận  xóa đến người dùng.  Bước 5: Nếu người dùng tiếp tục nhấn nút Ok. Hệ thống sẽ kiểm tra tính hợp lệ sản phẩm người dùng muốn xóa Nếu hợp lệ thì hệ thống tiến hành xóa bỏ hoàn toàn các thông tin của sản phẩm ra khỏi CSDL và thông báo “Xóa thành công” đến người dùng. Ngược lại nếu thông tin không hợp lệ thì hệ thống gửi một thông báo chính xác lỗi đến người dùng.  *Làm mới*  Bước 1: Người dùng chọn nút "Làm mới" để làm trắng ô nhập dữ liệu |
| Kết quả | Thêm mới, cập nhật, xóa, thành công hoặc thất bại |
| Ghi chú | Hát buộc phải có thông báo xác nhận khi người dùng chọn Xóa sản phẩm.  Tất cả các trường thông tin không được bỏ trống.  Hỗ trợ thêm chức năng Tim kiếm sản phẩm để thuận tiện hơn cho người dùng |

### 4.3.5 MS06 - Quản lý đơn hàng

|  |  |
| --- | --- |
| Mã yêu cầu | MS06 |
| Tên yêu cầu | Quản lý đơn hàng |
| Đối tượng sử dụng | Quản trị viên, khách hàng |
| Mực ưu tiên | cao |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Nội dung | Trong chức năng này người dùng có thể. Xem khách hàng và Xem đơn hàng. |
| Cách xử lý | *Cập nhật tình trạng đơn hàn*g  Bước 1: Hệ thống hiển thị danh sách các đơn hàng từ CSDL.  (Đối với quản trị viên)  Bước 2: Người dùng lựa chọn một đơn hàng từ danh  sách để Cập nhật tình trạng đơn hàng.  Bước 3: Trên màn hình quản lý đem hàng người dùng chọn nút "Duyệt đơn" hoặc “Hủy đơn” để xác nhận đơn hàng.  Bước 4: Hệ thống cập nhật lại CSDL cho thay đổi của tình trạng đơn hàng và gửi một thông báo xác nhận “Cập nhật thành công" đến người dùng.  *Hủy đơn hàng*  Bước 1: Hệ thống hiển thị danh sách các đơn hàng từ CSDL  Bước 2: Người dùng tìm một đơn hàng từ danh sách để Huỷ.  Bước 3: Hệ thống hiển thị một thông báo xác nhận “Bạn có muốn hủy đơn hàng này”. Nếu người dùng nhấn nút Có thể hệ thống tiến tinh Hủy bỏ toàn bộ các thông tin của một (nhiều) đơn hàng ra khỏi CSDL và gửi một thông báo xác nhận “hủy đơn hàng thành công” đến người dùng |
| Kết quả | Cập nhật tình trạng đơn hàng, hủy đơn hàng thành công hay thất bại. |
| Ghi chú | Bắt buộc phải có thông báo xác nhận đối với thao tác Hủy đơn hàng  Xây dựng thêm chức năng tìm kiếm đơn hàng để hỗ trợ người dùng thuận tiện hơn. |

### 4.3.6 MS07 - Quản lý khách hàng

|  |  |
| --- | --- |
| Mã yêu cầu | MS07 |
| Tên yêu cầu | Quản lý khách hàng |
| Đối tượng sử dụng | Quản trị viên |
| Mực ưu tiên | cao |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Nội dung | Cho phép người dùng là quản trị viên thực hiện các chức năng: Cập nhật tình trạng đơn hàng, hủy đơn hàng. |
| Cách xử lý | *Xem khách hàng*  Bước 1: Hệ thống hiển thị tất cả Khách hàng từ CSDL.  Bước 2: Người dùng chỉ xem khách hàng, và không có nút xóa, chỉ có khách hàng mới có quyền thay đổi thông tin cá nhân của tài khoản.  *Xem lịch sử mua hàng của khách hàng*  Bước 1: Hệ thống hiển thị tất cả Khách hàng từ CSDL.  Bước 2: Người dùng chọn một khách hàng muốn xem lịch sử và nhấp vào nút Lịch sử tương ứng.  Bước 3: Hệ thống truy vấn CSDL và hiển thị toàn bộ lịch sử của khách hàng đến người dùng. Các lịch sử được hiển thị bao gồm: Đơn đặt hàng và Tài khoản. |
| Kết quả | Xem chi tiết khách hàng, xem lịch sử của khách hàng khi giao dịch trong shop. |
| Ghi chú | Không |

### 4.3.7 MS08 - Xem chi tiết sản phẩm

|  |  |
| --- | --- |
| Mã yêu cầu | MS08 |
| Tên yêu cầu | Xem chi tiết sản phẩm |
| Đối tượng sử dụng | khách hàng |
| Mực ưu tiên | cao |
| Tiền điều kiện | Không |
| Nội dung | Cho phép người dùng là khách hàng xem chi tiết thông tin 1 sản phẩm |
| Cách xử lý | Bước 1: Người dùng nhấp vào tên hoặc nhấp vào ánh đại diện tương ứng với sản phẩm muốn xem thông tin.  Bước 2: Hệ thống chuyển người dùng đến trang Chi tiết sản phẩm. Tại đây thông tin đầy đủ về sản phẩm được hiển thị đầy đủ đến người dùng: Tên sản phẩm, giả sản phẩm, mô tả ngắn về sản phẩm, thông tin chí tiết... |
| Kết quả | Xem chi tiết sản thành công hoặc thất bại |
| Ghi chú | Không |

### 4.3.8 MS09 - Tìm kiếm sản phẩm

|  |  |
| --- | --- |
| Mã yêu cầu | MS09 |
| Tên yêu cầu | Tìm kiếm sản phẩm |
| Đối tượng sử dụng | khách hàng |
| Mực ưu tiên | cao |
| Tiền điều kiện | Không |
| Nội dung | Cho phép người dùng là khách hàng thực hiện tìm kiếm sản phẩm theo loại sản phẩm giá... |
| Cách xử lý | Bước 1: Tử giao diện Trang chủ người dùng chọn "Sản phẩm" từ Menu. Hệ thống sẽ chuyển người dùng đến trang xem sản phẩm Tại đây người dùng chọn tìm kiếm theo loại sản phẩm hoặc tìm kiếm theo giá sản phẩm..  Bước 2: Hệ thống truy vấn CSDL và trả về kết quả tìm kiếm những sản phẩm mà người dùng nhập vào ở bước 1 hoặc trả về “Không có sản phẩm nào giống với tử khỏa” nếu không tìm thấy. |
| Kết quả | Tìm kiếm sản phẩm thành công hoặc thất bại |
| Ghi chú | Không |

### 4.3.9 MS10 - Xem chi tiết giỏ hàng

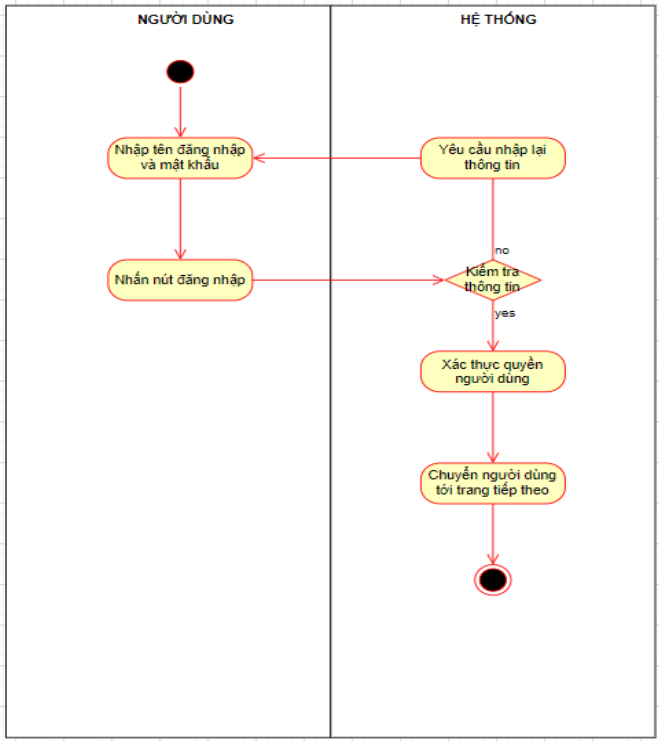
|  |  |
| --- | --- |
| Mã yêu cầu | MS10 |
| Tên yêu cầu | Xem chi tiết giỏ hàng |
| Đối tượng sử dụng | khách hàng |
| Mực ưu tiên | cao |
| Tiền điều kiện | Không |
| Nội dung | Cho phép người dùng là khách hàng thực hiện các chức năng sau: Thêm sản phẩm vào giỏ hàng, cập nhật số lượng sản phẩm trong gió hàng, xóa sản phẩm trong gió hàng, thanh toán. |
| Cách xử lý | *Thêm sản phẩm vào giỏ hàng*  Bước 1: Người dùng chọn một sản phẩm cần mua và bán nút "Thêm vào giỏ hàng"(nếu người dùng chưa đăng nhập hệ thống sẽ chuyển người dùng đến trang Đăng nhập để đăng nhập và tiếp tục).  Bước 2: Hệ thống lưu sản phẩm người dùng vừa thêm vào giỏ hàng  *Cập nhật số lượng sản phẩm trong giỏ hàng*  Bước 1: Người dùng chọn số lượng muốn cập nhật lại vào ô số lượng...  Bước 2: Hệ thống cập nhật lại số lượng của sản phẩm tương ứng trong gió hàng.  *Xóa sản phẩm trong giỏ hàng*  Bước 1: Người dùng chọn một sản phẩm cần xóa ra khỏi giỏ hàng và bấm vào nút Xóa được hiển thị dưới dạng icon “Xóa").  Bước 2: Hệ thống xóa sản phẩm tương ứng ra khỏi giỏ hàng.  *Thanh toán*  Bước 1: Từ trang Giỏ hàng người dùng bấm nút Thanh toán để tiến hành đặt hàng.  Bước 2: Hệ thống chuyển người dùng đến trang Thanh toán để đặt hàng  Bước 3: Người dùng điền đầy đủ (hoặc cập nhật lại) thông tin giao hàng và xem lại đơn hàng trước khi Thanh toán.  Bước 4: Người dùng bấm nút “Thanh toán”.  Bước 5: Hệ thống kiểm tra thông tin thanh toán ( Thông báo nếu có lỗi xảy ra) và lưu thông tin vào csdl |
| Kết quả | Người dùng mua hàng, cập nhật số lượng, xóa sản phẩm trong giỏ hàng, thanh toán đơn hàng thành công hoặc thất bại |
| Ghi chú | Người dùng phải đăng nhập trước khi thực hiện các chức năng |

## 4.4. Đặc tả thiết kế hệ thống:

### 4.4.1 Thiết kế và cách xử lý:

#### 4.4.1.1 Đăng nhập:

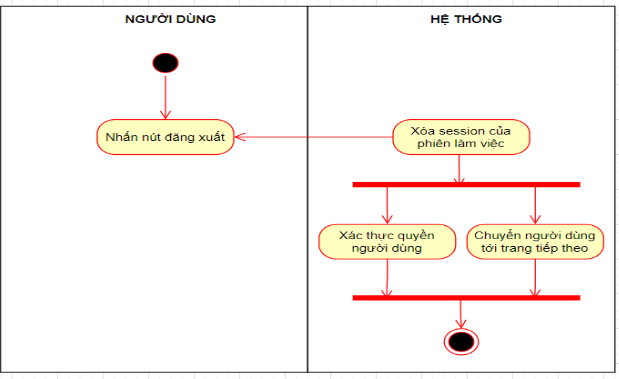
* Mục đích: cho phép người dùng đăng nhập vào hệ thống.
* Cách xử lý:



*Hình 4.1 Biểu đồ hoạt động chức năng Đăng nhập*

#### 4.4.1.2. Đăng xuất

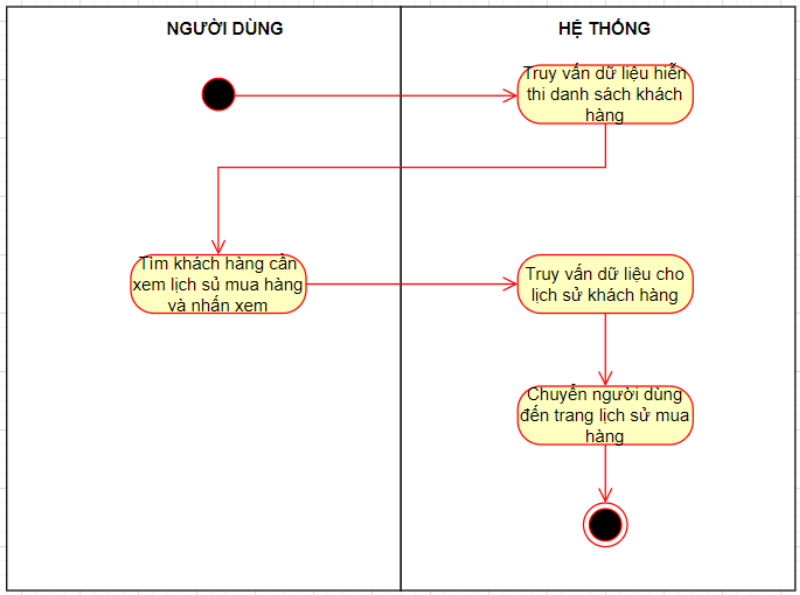
* Mục đích: Cho phép người dùng thực hiện đăng xuất khi kết thúc phiên làm việc.



*Hình 4.2 Biểu đồ chức năng đăng xuất*

#### 4.4.1.3 Xem lịch sử khách hàng:

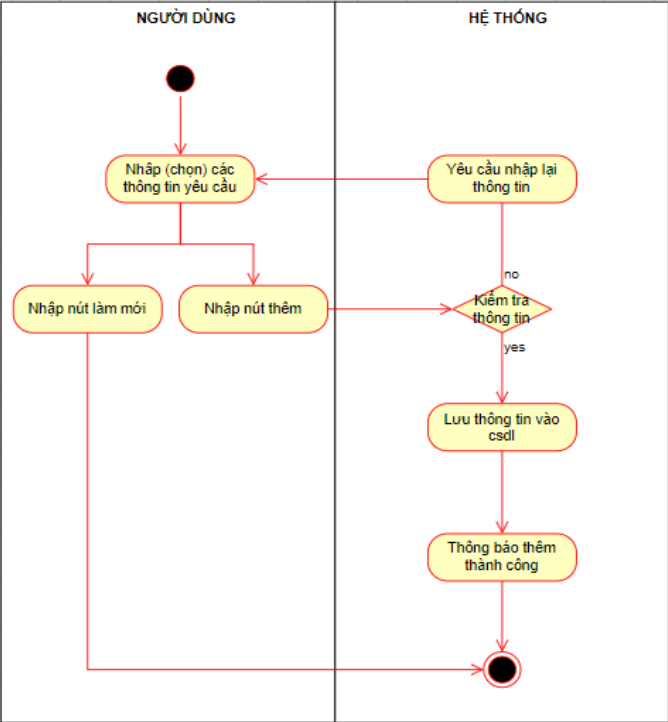
* Mục đích: Cho phép quản trị viên xem lịch sử mua hàng của khách hàng
* Cách xử lý:



*Hình 4.3 Biểu đồ hoạt động xem lịch sử mua hàng của khách hàng*

#### 4.4.1.4 Thêm mới sản phẩm:

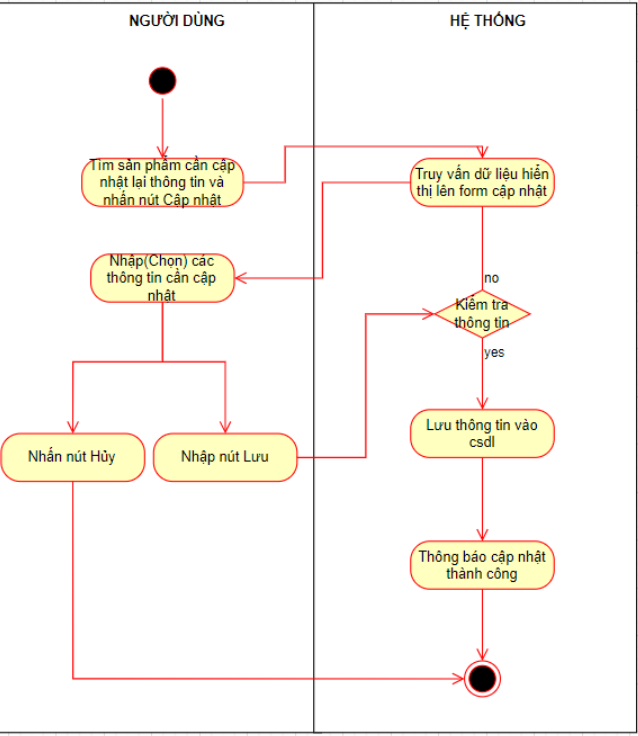
* Mục đích: Cho phép quản trị viên thêm sản phẩm
* Cách xử lý:



*Hình 4.4 Biểu đồ thêm mới sản phẩm*

#### 4.4.1.5 Cập nhật thông tin sản phẩm:

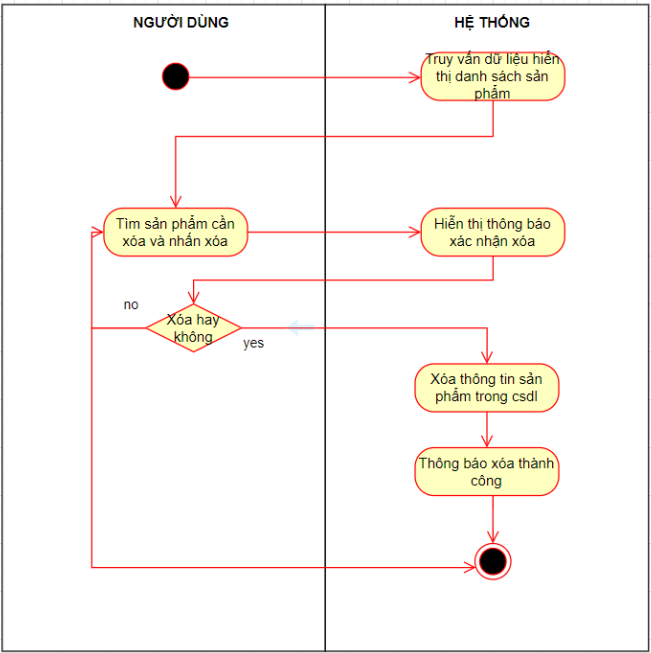
* Mục đích: Cho phép quản trị viên cập nhật sản phẩm
* Cách xử lý:



*Hình 4.5 Biểu đồ cập nhật sản phẩm*

#### 4.4.1.6 Xóa sản phẩm:

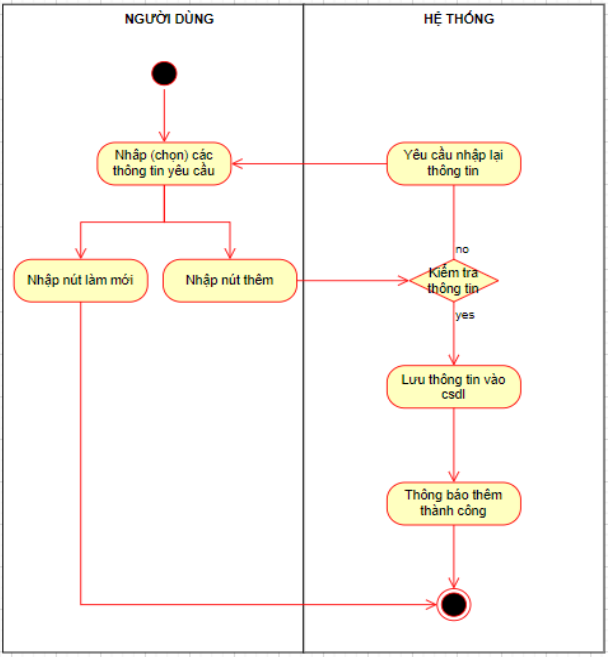
* Mục đích: Cho phép quản trị viên xóa sản phẩm
* Cách xử lý:



*Hình 4.6 Biểu đồ xóa sản phẩm*

#### 4.4.1.7 Thêm mới loại sản phẩm:

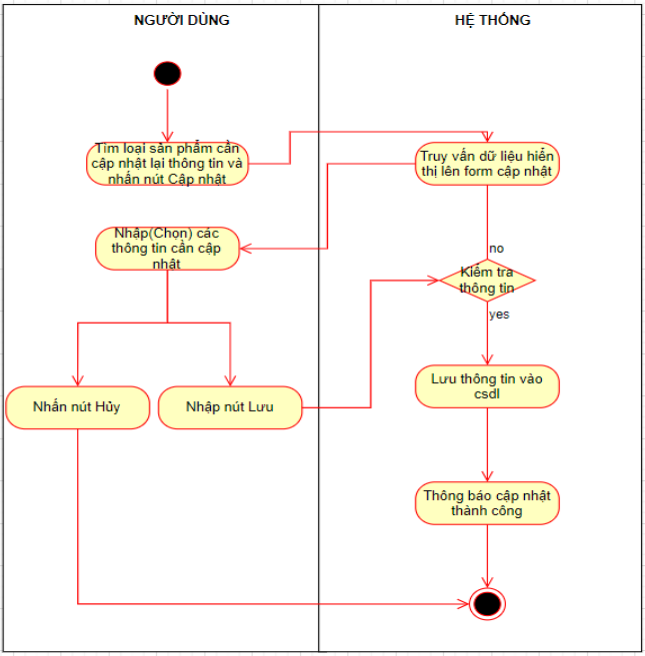
* Mục đích: Cho phép quản trị viên thêm mới loại sản phẩm
* Cách xử lý:



*Hình 4.7 Biểu đồ thêm mới loại sản phẩm*

#### 4.4.1.8 Cập nhật loại sản phẩm:

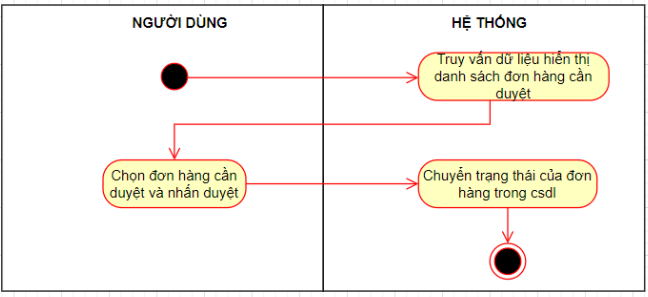
* Mục đích: Cho phép người dùng là quản trị viên thêm mới loại sản phẩm
* Cách xử lý:



*Hình 4.8 Biểu đồ cập nhật loại sản phẩm*

#### 4.4.1.9 Xác nhận tình trạng đơn hàng:

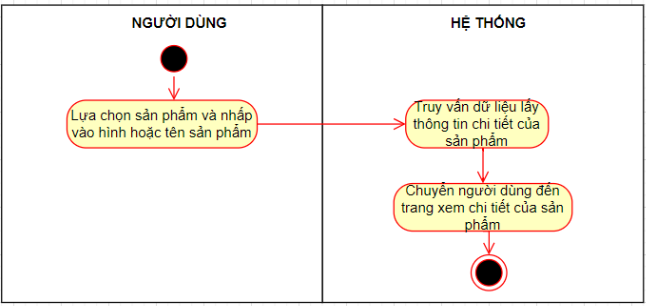
* Mục đích: Cho phép quản trị viên duyệt đơn hàng của khách hàng về sản phẩm.
* Cách xử lý:



*Hình 4.9 Biểu đồ duyệt đơn hàng*

#### 4.4.1.10 Xem chi tiết sản phẩm:

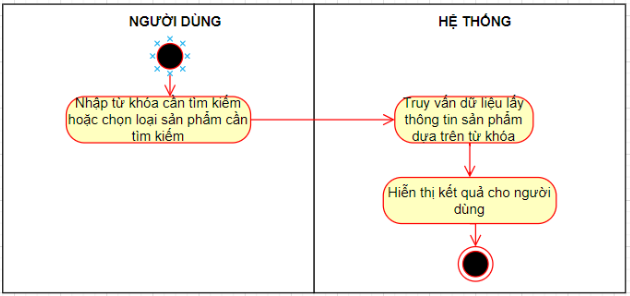
* Mục đích: Cho phép khách hàng xem chi tiết của sản phẩm
* Cách xử lý:



*Hình 4.10 Biểu đồ xem chi tiết sản phẩm*

#### 4.4.1.11 Tìm kiếm sản phẩm:

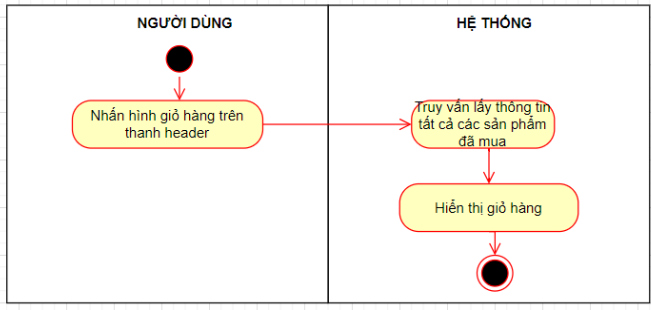
* Mục đích: Cho phép khách hàng tìm kiếm sản phẩm
* Cách xử lý:

**

*Hình 4.11 Biểu đồ tìm kiếm sản phẩm*

#### 4.4.1.12 Xem chi tiết giỏ hàng:

* Mục đích: Cho phép khách hàng xem lại các sản phẩm đã thêm vào giỏ hàng
* Cách xử lý:



#### 4.4.1.13 Xóa giỏ hàng:

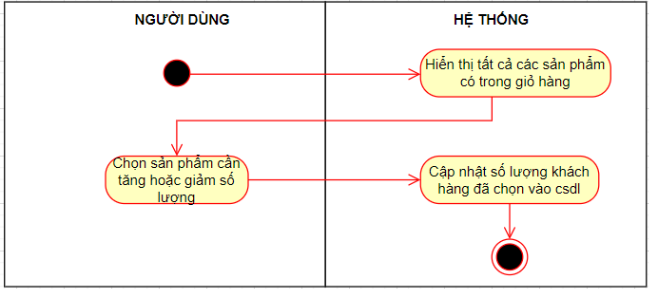
* Mục đích: Cho phép khách hàng xóa các sản phẩm đã thêm vào giỏ hàng
* Cách xử lý:



*Hình 4.13 Biểu đồ xóa giỏ hàng*

#### 4.4.1.14 Cập nhật số lượng sản phẩm trong giỏ hàng:

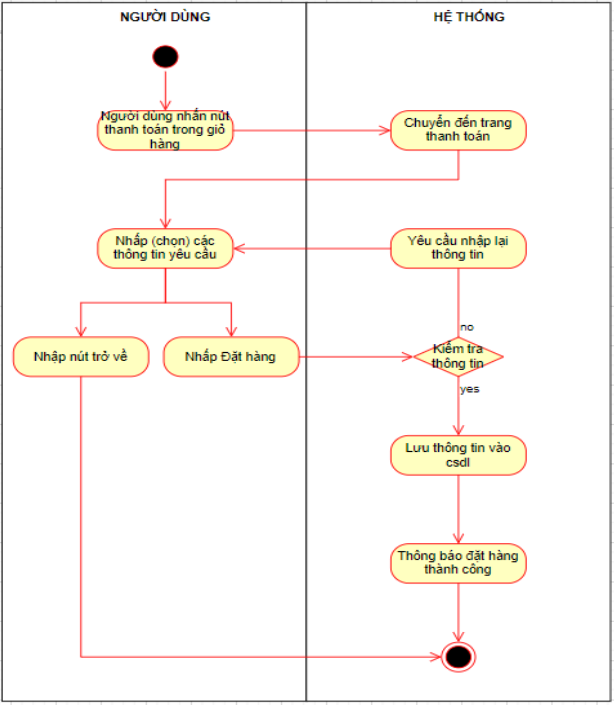
* Mục đích: Cho phép khách hàng cập nhật các sản phẩm đã thêm vào giỏ hàng
* Cách xử lý:

****

*Hình 4.14 Biểu đồ cập nhật sản phẩm trong giỏ hàng*

#### 4.4.1.15 Đặt hàng:

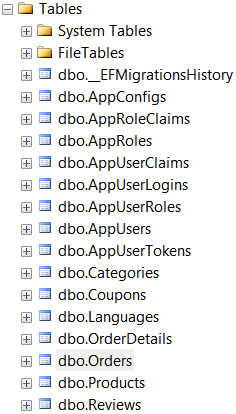
* Mục đích: Cho phép khách hàng đặt hàng các sản phẩm đã thêm vào giỏ hàng
* Cách xử lý:

**

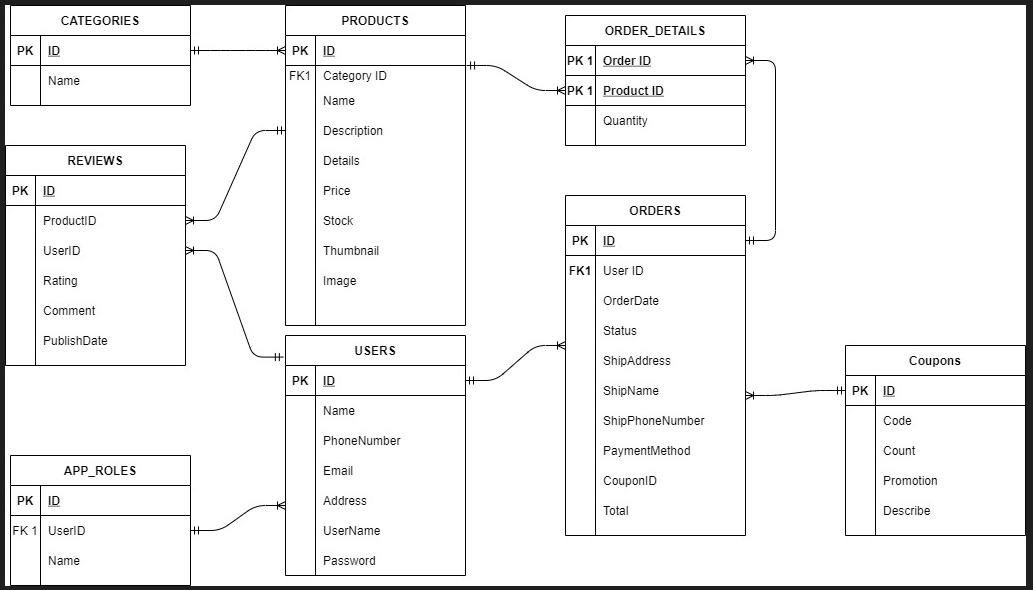
*Hình 4.15 Biểu đồ đặt sản phẩm*

### 4.4.2 Thiết kế cơ sở dữ liệu:

Sau khi code first thành công ta được cơ sở dữ liệu đầy dủ

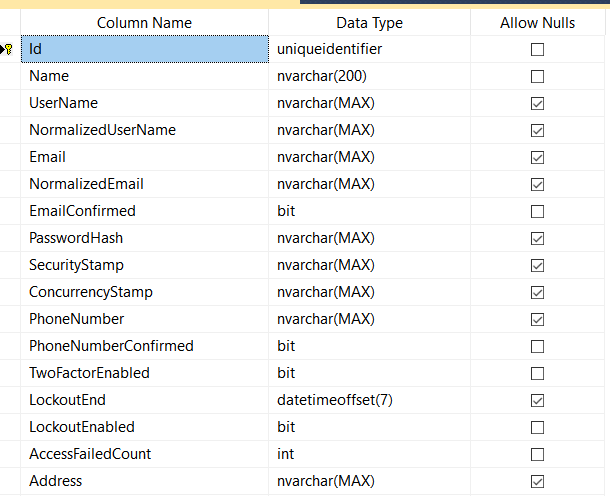


#### 4.2.2.1 Lược đồ dữ liệu quan hệ:



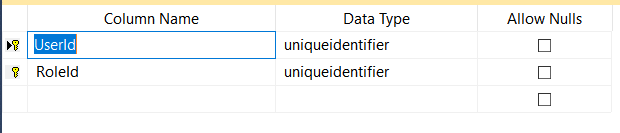
*Hình 4.16 Lược đồ dữ liệu quan hệ*

Bảng User:



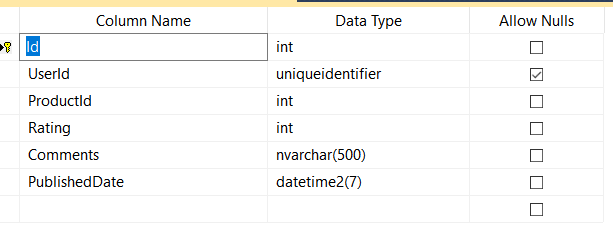
*Hình 4.17 Bảng user*

Bảng Roles:



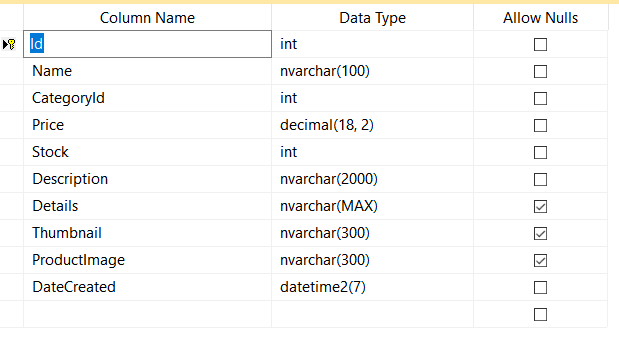
*Hình 4.18 Bảng Role*

Bảng Reviews:



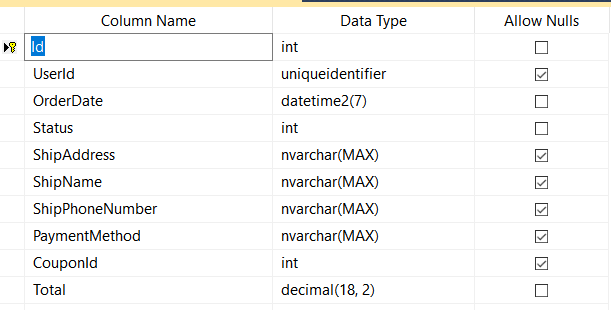
*Hình 4.19 Bảng Reviews*

Bảng Products:



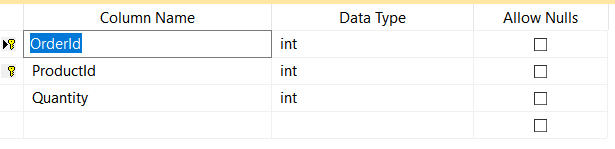
*Hình 4.20 Bảng Products*

Bảng Orders sản phẩm:



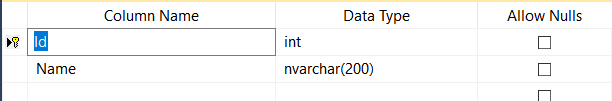
*Hình 4.21 Bảng Orders*

Bảng OrderDetails:



*Hình 4.22 Bảng OrderDetails*

Bảng Categories:



*Hình 4.23 Bảng Categories*

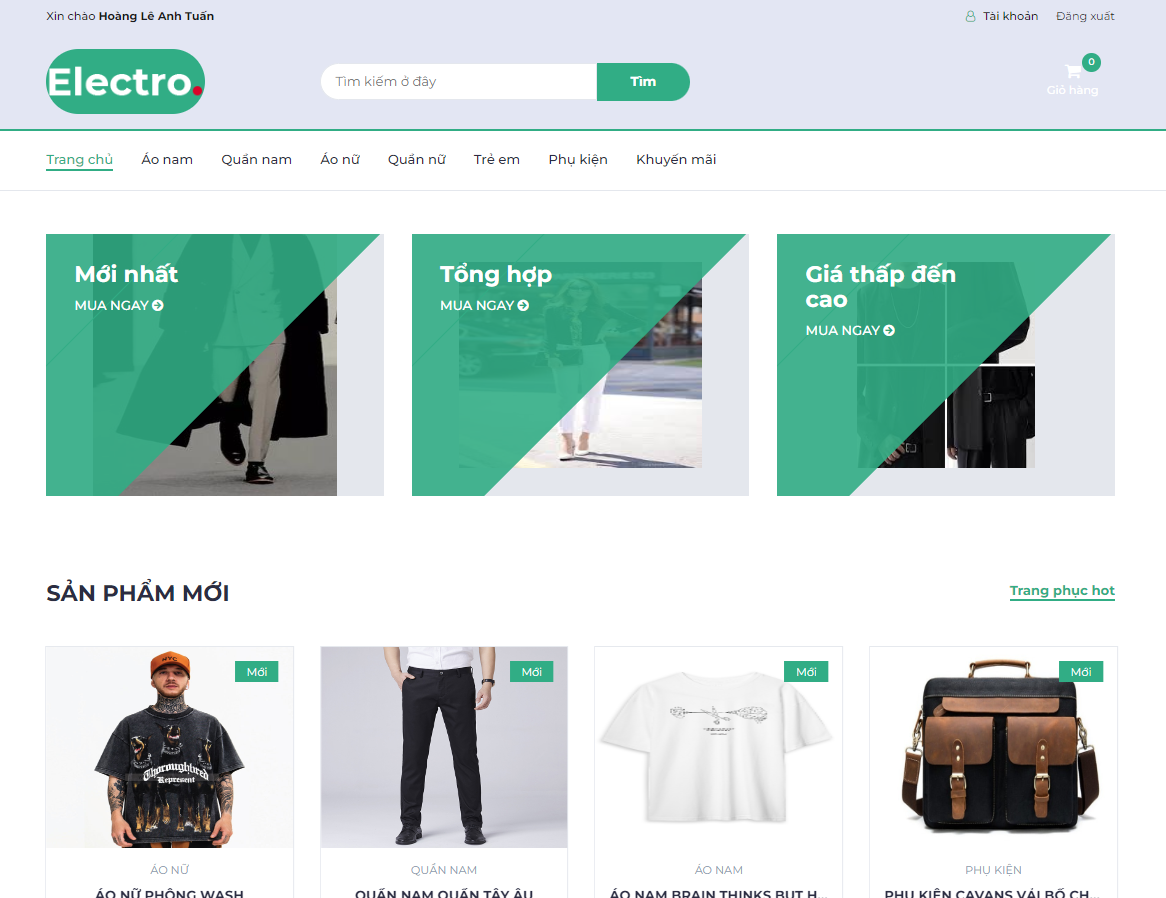
Bảng Khuyến mãi(Coupons):



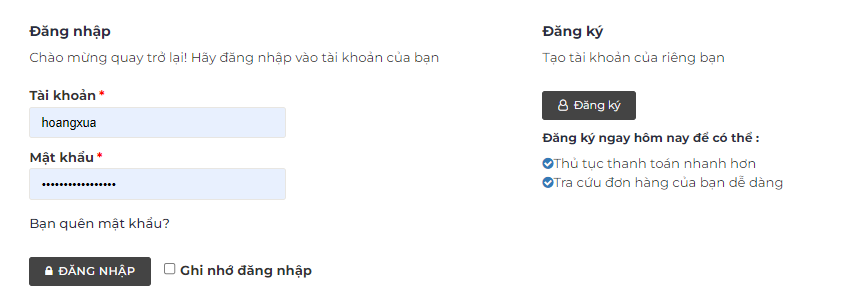
*Hình 4.24 Bảng Coupons*

### 4.4.3 Thiết kế giao diện người dùng:

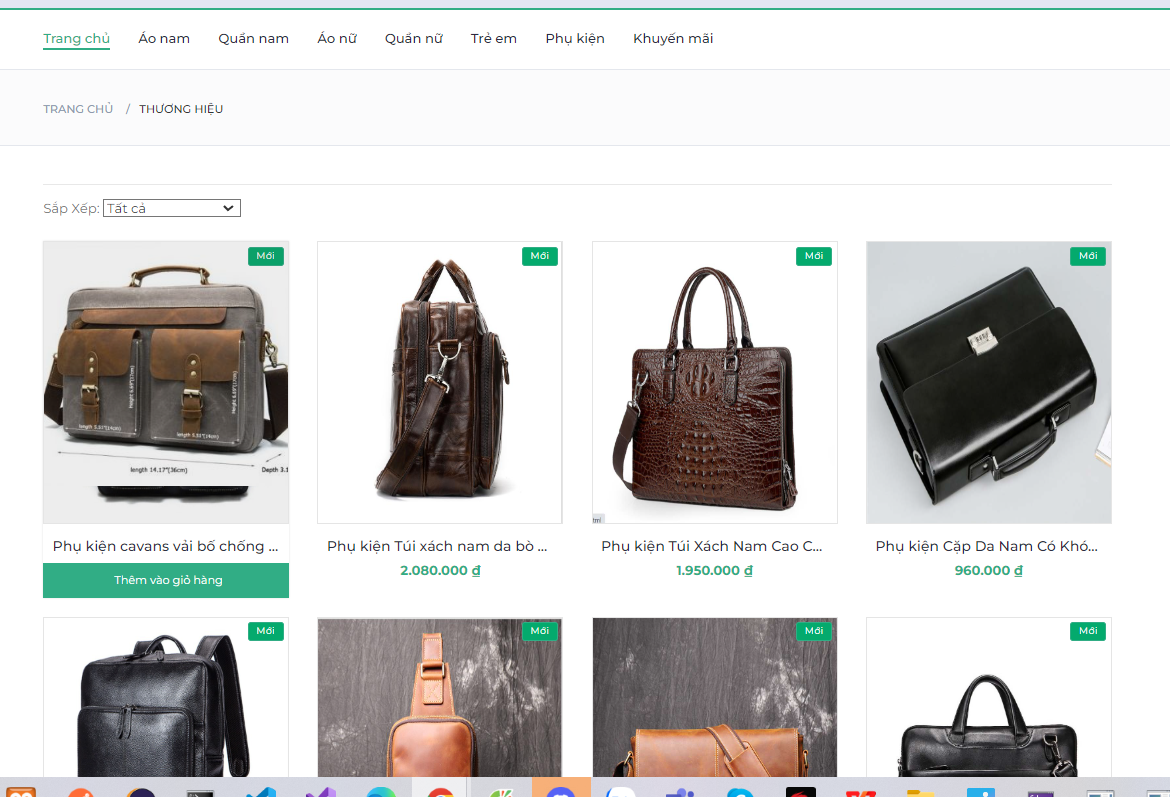
#### 4.3.3.1: Giao diện khách hàng:



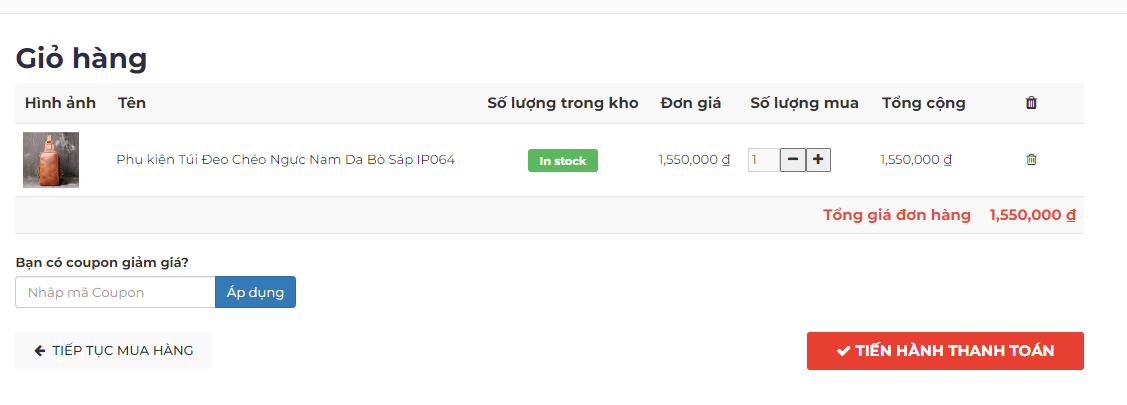
*Hình 4.25 Giao diện trang chủ*

**

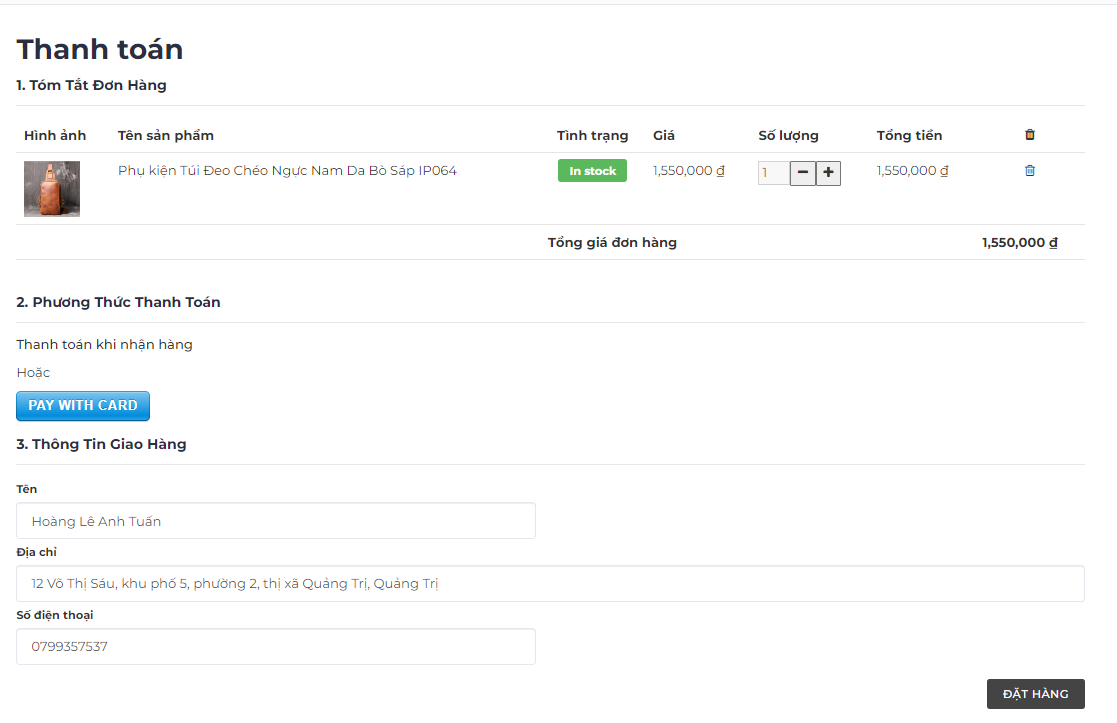
*Hình 4.26 Giao diện đăng nhập, đăng ký*

**

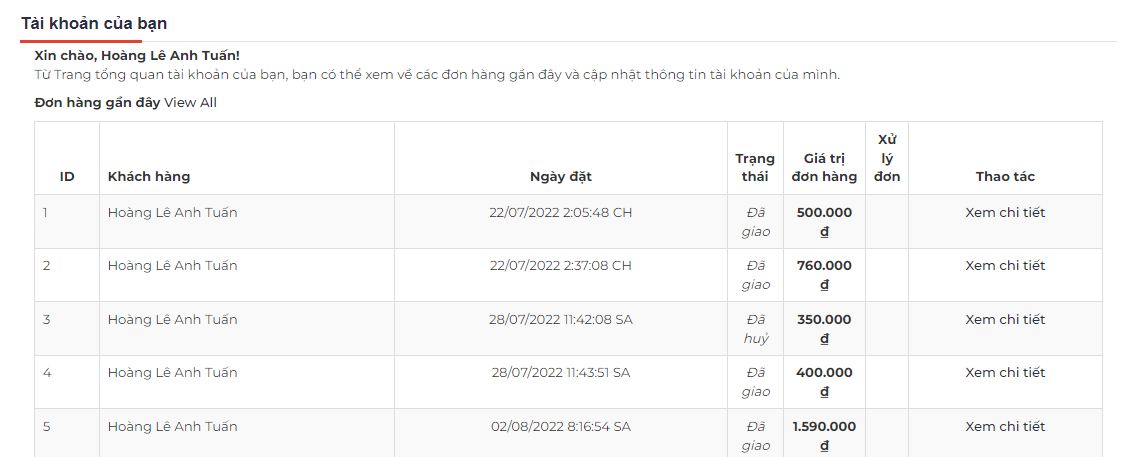
*Hình 4.27 Giao diện đặt hàng*

**

*Hình 4.28 Giao diện giỏ hàng*

**

*Hình 4.29 Giao diện đặt hàng*

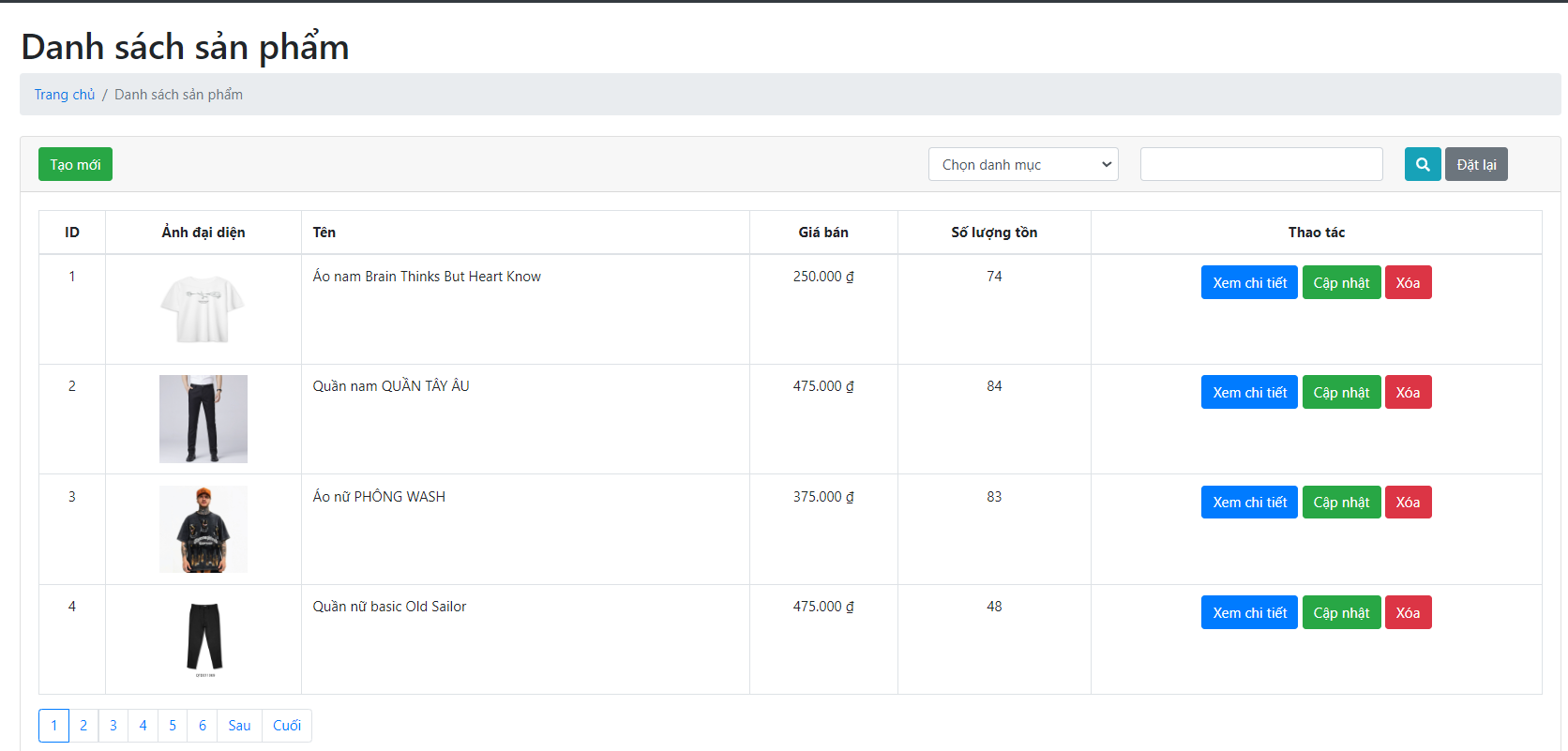
**

*Hình 4.30 Giao diện lịch sử mua hàng*

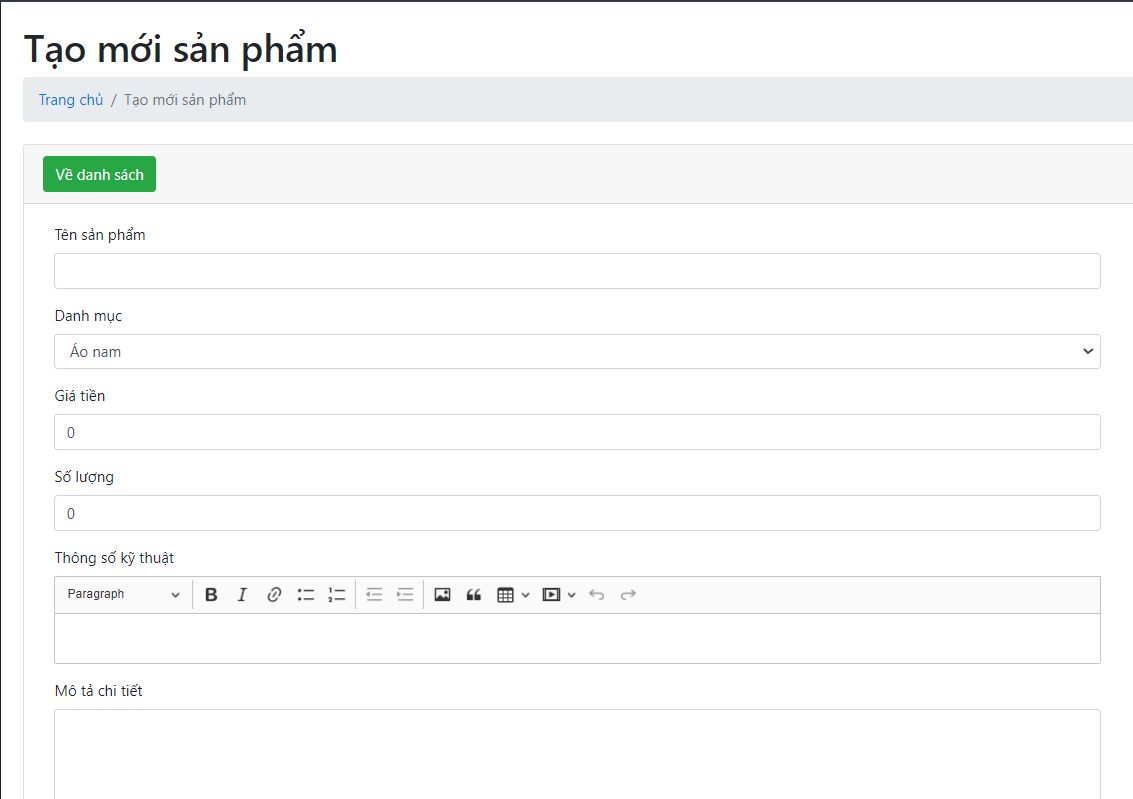
#### 4.3.3.2: Giao diện quản lý:



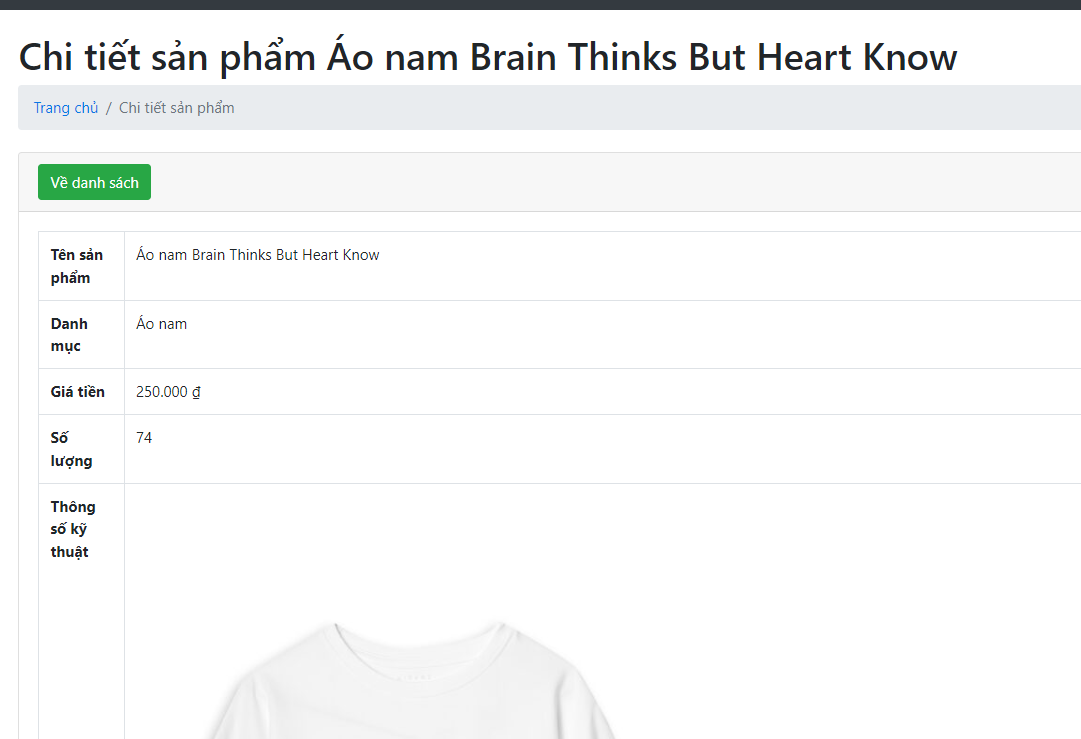
*Hình 4.31 Giao diện trang chủ trang quản lý*

**

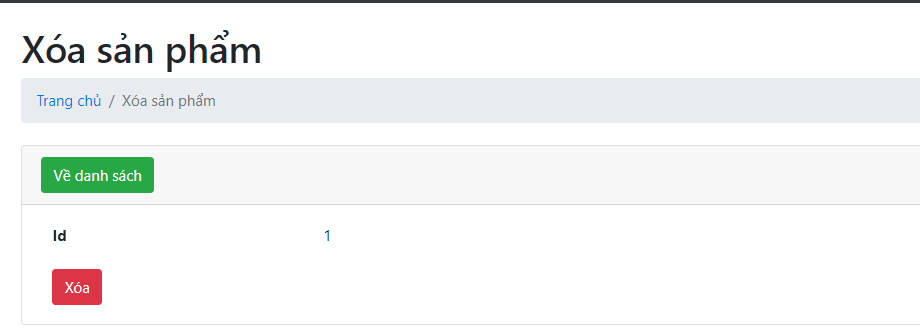
*Hình 4.32 Giao diện trang quản lý sản phẩm*

**

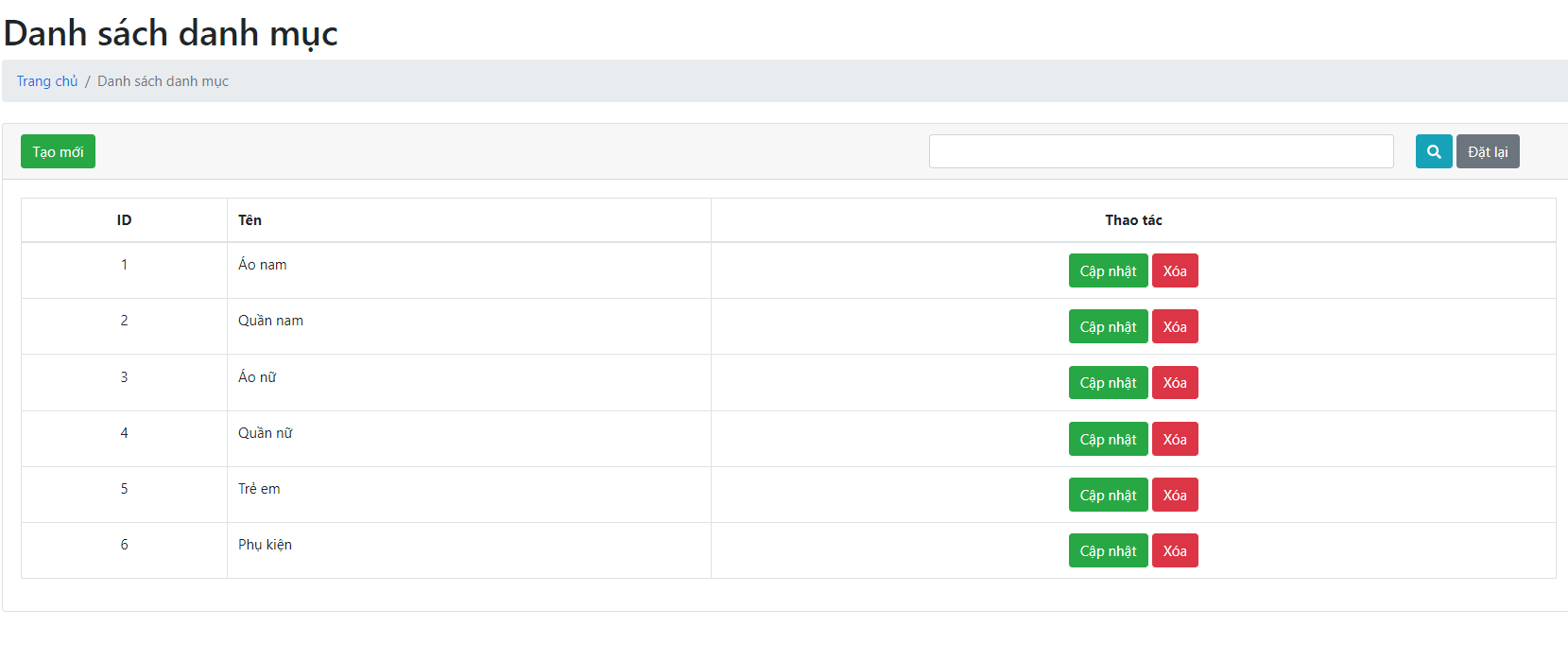
*Hình 4.33 Thêm sản phẩm*

**

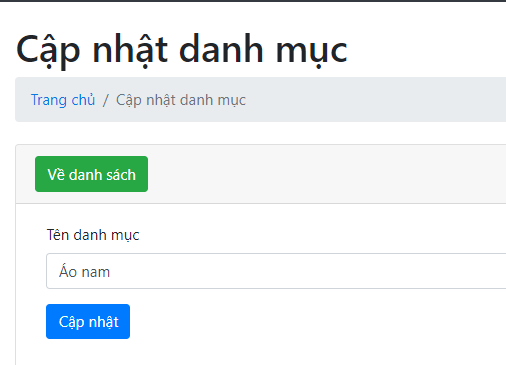
*Hình 4.34 Xem chi tiết sản phẩm*

**

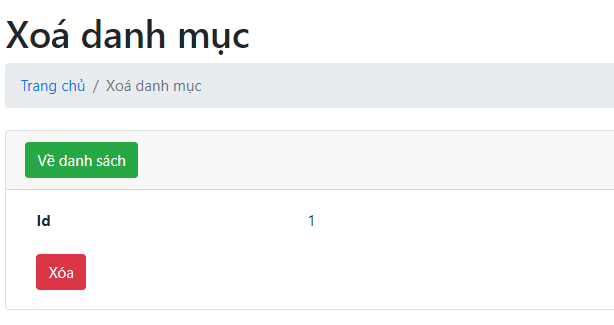
*Hình 4.35 Xóa sản phẩm*

**

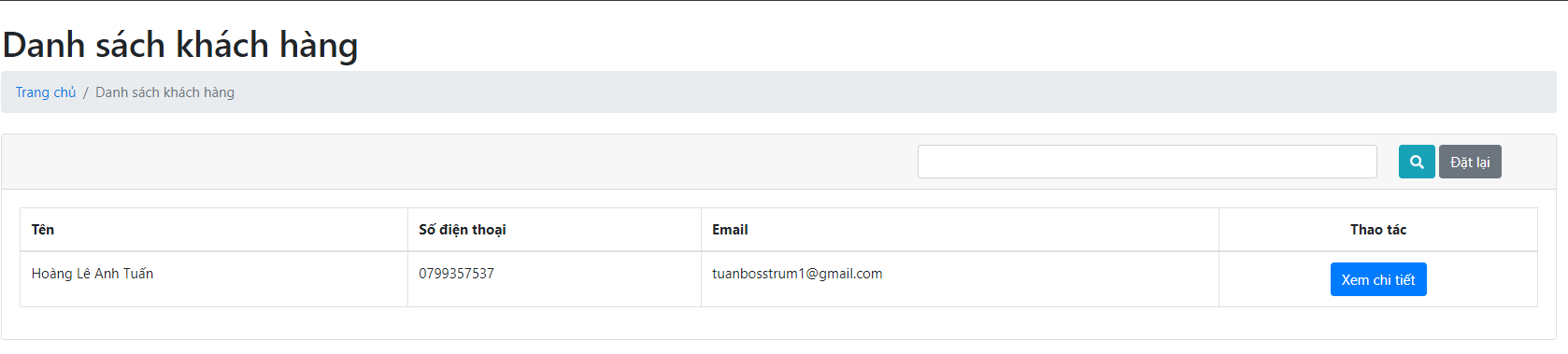
*Hình 4.36 Giao diện trang quản lý danh mục sản phẩm*

**

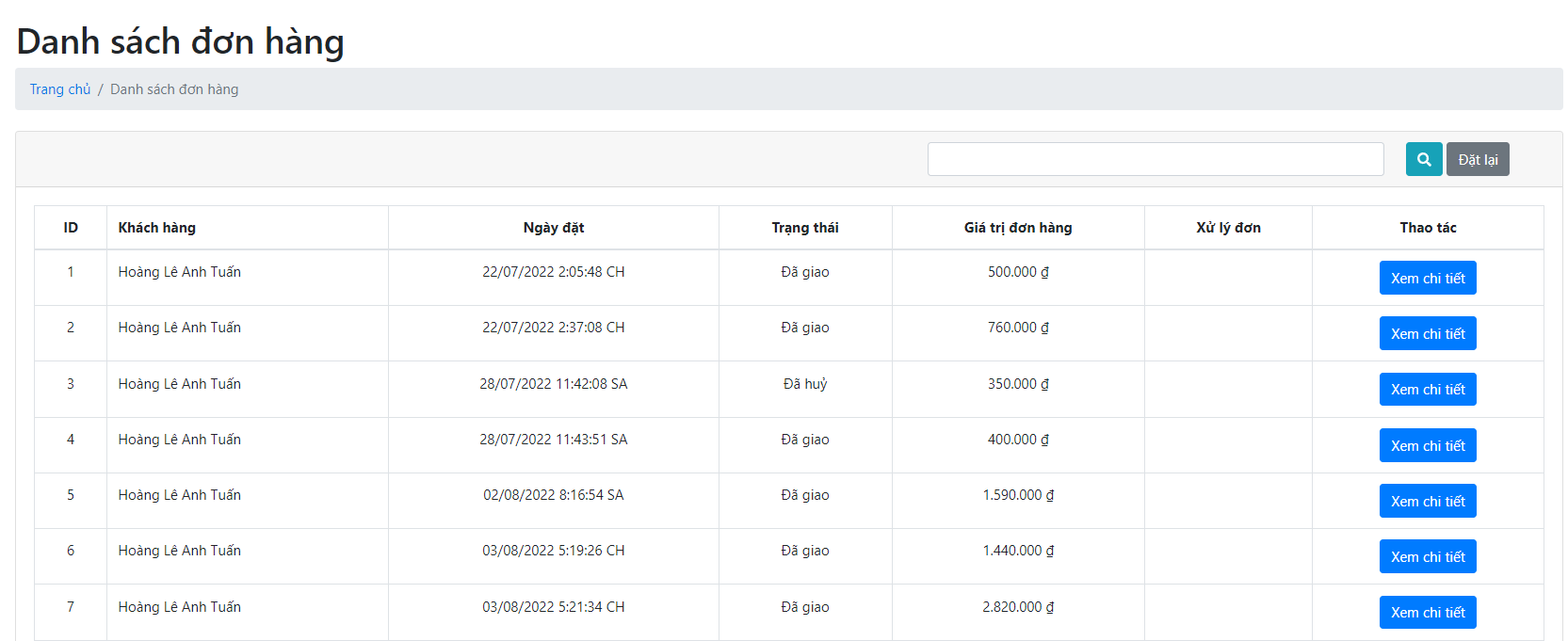
*Hình 4.37 Cập nhật danh mục sản phẩm*

**

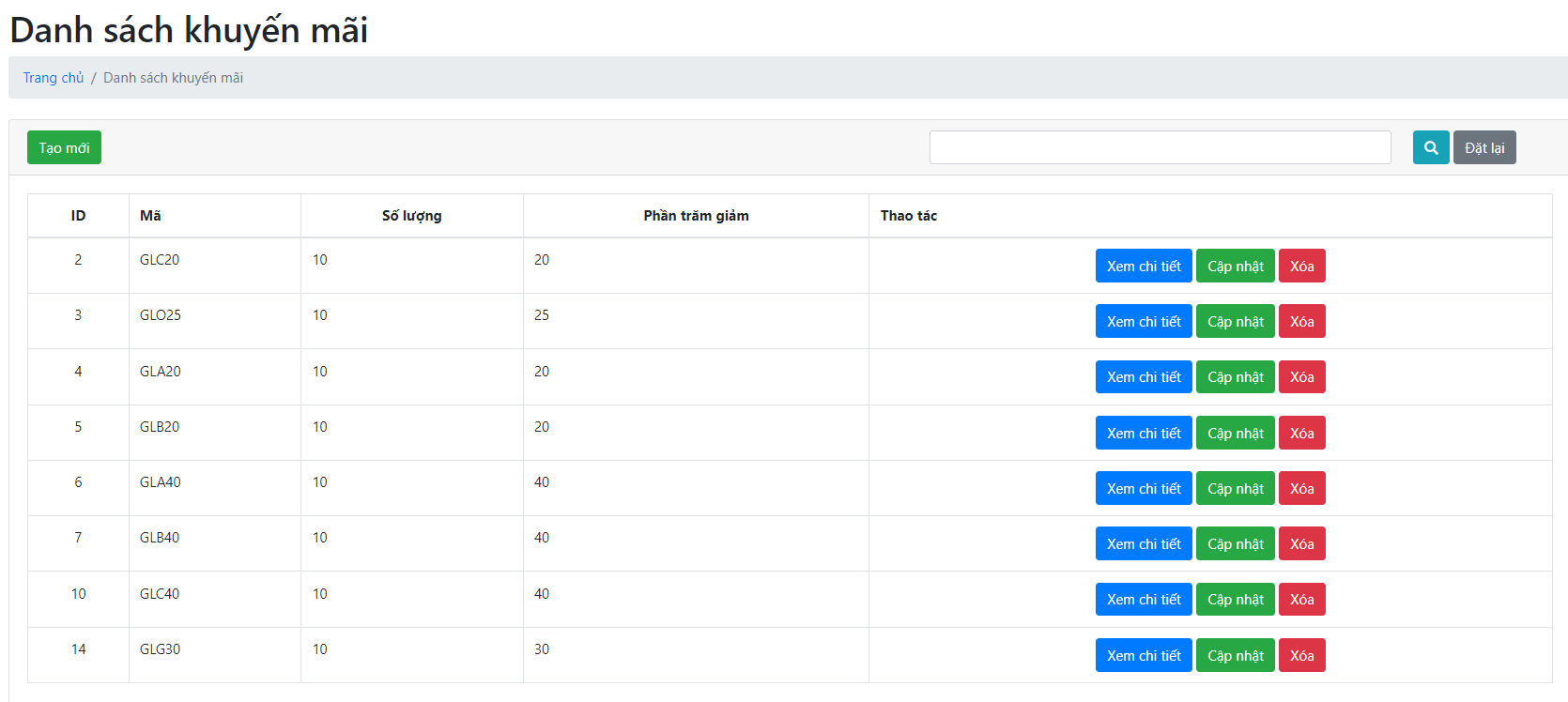
*Hình 4.38 Xóa danh mục sản phẩm*

**

*Hình 4.39 Giao diện danh sách khách hàng*

**

*Hình 4.40 Giao diện đơn hàng đã đặt*

**

*Hình 4.41 Giao diện mã khuyến mãi sản phẩm*

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 1. KẾT QUẢ ĐẶT ĐƯỢC

Trong quá trình thực hiện đề tài em đã cố gắng hết sức để tìm hiểu nghiệp vụ, quy trình bán hàng của nhà hàng và một số web khác như Hidanz, Navy shop để xây dựng và cài đặt chương trình nhưng vì thời gian có hạn nên chưa thể giải quyết được toàn bộ các vấn đề. Em mong nhận được sự cảm thông của quý thầy cô.

*Về công nghệ:*

Tìm hiểu và nắm bắt được các công cụ thiết kế web,

Biết được các thiết kế web động cũng như cách tổ chức cơ sở dữ liệu.

Hiểu sâu hơn về Asp.net core, các kiến thức hỗ trợ trong quá trình xây dựng hệ thống CSS, HTML, JavaScript, …

Tích lũy được kinh nghiệm về xây dựng kế hoạch cho việc phát triển một phần mềm hoàn chỉnh, cách bố trị và quản lý thời gian sao cho hợp lý.

*Về ứng dụng:*

Xây dựng thành công hệ thống “Xây dựng website cửa hàng quần áo trực tuyến bằng ASP.NET Core” đáp ứng được nhu cầu mua – bán của một cửa hàng, giúp cho hoạt động mua bán ở cửa hàng diễn ra nhanh chóng và tiện lợi hơn.

Giao diện thân thiện với người dùng.

Giới thiệu các sản phẩm của cửa hàng đến với khách hàng trong và ngoài nước.

Cho phép tra cứu mặt hàng khi khách hàng có nhu cầu đặt hàng.

Cho phép khách hàng thực hiện việc đặt hàng qua mạng.

Tiếp nhận và xử lý đơn hàng của khách hàng.

Cập nhật: sản phẩm, nhập hàng...

Thống kê các mua bán theo nhiều tiêu chí.

## 2. HẠN CHẾ

Khả năng làm việc thực tế còn nhiều hạn chế nhất là trong quá trình tiếp xúc, thu thập dữ liệu từ cửa hàng thực tế.

Thời gian thực hiện còn khá ít nên một số vấn đề vẫn chưa được giải quyết hoặc giải quyết được tối ưu nhất.

Hệ thống mới chỉ đáp ứng nhu cầu của một cửa hàng quy mô nhỏ, khả năng đáp ứng cho các cửa hàng lớn quy tắc nghiệp vụ phức tạp còn hạn chế.

Một số chức năng của hệ thống chỉ đáp ứng một phần nào yêu cầu của người dùng, chưa có tính sáng tạo, đột phá trong cách giải quyết vấn đề.

Khi đưa vào hoạt động thực tế trên host hệ thống chưa đáp ứng được nhu cầu về tốc độ tải.

## 3. HƯỚNG PHÁT TRIỂN

Hoàn thiện thêm các chức năng đang còn hạn chế của hệ thống.

Cho phép khách hàng mua hàng và thanh toán trực tiếp qua mạng thông qua thì ATM.

Hoàn thiện hệ thống hơn để khi đưa vào vận hành thực tế có thể vận hành tốt nhất đáp ứng đủ các tiêu chí về web hiện nay.

Xây dựng và mở rộng hệ thống cho phép các công ty, trường học có thể mua hàng với giá tốt nhất trực tiếp qua mạng.

*Tài liệu tham khảo:*

[*https://www.w3schools.com/html/*](https://www.w3schools.com/html/)

[*https://www.youtube.com/c/MyskillsVn*](https://www.youtube.com/c/MyskillsVn)

[*https://tedu.com.vn/khoa-hoc/lam-du-an-voi-aspnet-core-31-34.html*](https://tedu.com.vn/khoa-hoc/lam-du-an-voi-aspnet-core-31-34.html)

*group nhóm ASP Net core trên facebook:*

*https://www.facebook.com/groups/1863063413955695*