**BỘ CÔNG THƯƠNG**

**TRƯỜNG CAO ĐẲNG CÔNG THƯƠNG TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CHUYÊN ĐỀ THỰC TẬP**

**XÂY DỰNG WEBSITE BÁN ĐIỆN THOẠI**

**(SỬ DỤNG REACTJS, ASP.NET CORE API, SQL SERVER)**

**Giảng viên hướng dẫn: Huỳnh Tấn Phát**

**Thực hiện:** Nguyễn Bách Đoan

MSSV: 21110190 Lớp: CCQ2111F

TP Thủ Đức, ngày 1 tháng 11 2023

# LỜI CẢM ƠN

Lời đầu tiên, xin trân trọng cảm ơn Thầy Huỳnh Tấn Phát đã tận tình hướng dẫn em trong quá trình học tập cũng như trong việc hoàn thành đồ án.

Xin chân thành cảm ơn các Thầy, Cô thuộc khoa Công Nghệ Thông Tin cũng như những khoa đang công tác tại trường Cao Đẳng Công Thương TP.HCM đã tận tình giảng dạy cho em trong thời gian học tập.

Xin chân thành cảm ơn ban giám hiệu trường Cao Đẳng Công Thương TP.HCM đã tạo điều kiện cho em cũng như các sinh viên khác hoàn thành tốt đồ án tốt nghiệp của mình.

Xin gửi lời cảm ơn đến gia đình, bạn bè, những người luôn sẵn sàng chia sẻ và giúp đỡ em trong học tập và cuộc sống.

Do giới hạn kiến thức và khả năng lý luận của bản thân còn thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của các Thầy, Cô để bài luận của em được hoàn thiện hơn. Xin chân thành cảm ơn!

TP Thủ Đức, ngày 11 tháng 06 năm 2021

Sinh viên thực hiện

Đoan

Nguyễn Bách Đoan

Mục lục

[LỜI CẢM ƠN 2](#_Toc150527734)

[DANH MỤC CÁC BẢNG 4](#_Toc150527735)

[DANH MỤC CÁC HÌNH, ĐỒ THỊ 6](#_Toc150527736)

[CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI 9](#_Toc150527737)

[1.1. Lí do chọn đề tài 9](#_Toc150527738)

[1.2. Mục tiêu của đề tài 9](#_Toc150527739)

[1.3. Dự kiến kết quả đạt được 9](#_Toc150527740)

[CHƯƠNG 2. CƠ SỞ LÍ THUYẾT 10](#_Toc150527741)

[2.1. RESTful Api 10](#_Toc150527742)

[2.1.1. Giới thiệu 10](#_Toc150527743)

[2.1.2. Cách hoạt động 11](#_Toc150527744)

[2.2. ASP.NET Core 11](#_Toc150527745)

[2.2.1. Giới thiệu về ASP.NET Core 11](#_Toc150527746)

[2.2.2. Cấu trúc của ASP.NET Core 12](#_Toc150527747)

[2.3. SQLServer 16](#_Toc150527748)

[2.3.1. Giới thiệu SQLServer 16](#_Toc150527749)

[2.3.2. Các thành phần của SQLServer 16](#_Toc150527750)

[2.4. React JS 17](#_Toc150527751)

[2.4.1. Giới thiệu về React JS 17](#_Toc150527752)

[2.4.2. Các thành phần của React JS 17](#_Toc150527753)

[2.5. TailwindCSS 19](#_Toc150527754)

[CHƯƠNG 3. KHẢO SÁT VÀ PHÂN TÍCH CHỨC NĂNG 20](#_Toc150527755)

[3.1. Khảo sát hệ thống 20](#_Toc150527756)

[3.2. Phân tích yêu cần hệ thống 20](#_Toc150527757)

[3.2.1. Yêu cầu chức năng 20](#_Toc150527758)

[3.2.2. Yêu cầu phi chức năng 20](#_Toc150527759)

[CHƯƠNG 4. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 21](#_Toc150527760)

[4.1. Danh sách Actor 21](#_Toc150527761)

[4.2. Danh sách Use Case 21](#_Toc150527762)

[4.3. Biểu đồ use case 22](#_Toc150527763)

[4.3.1. Biểu đồ use case sử dụng cho quản trị 22](#_Toc150527764)

[4.4. Biểu đồ hoạt động 23](#_Toc150527765)

[4.4.1. Phía quản trị 23](#_Toc150527766)

[4.5. Biểu đồ tuần tự 27](#_Toc150527767)

[4.5.1. Phía quản trị 27](#_Toc150527768)

[4.6. Thiết kế cơ sở dữ liệu 31](#_Toc150527769)

[4.6.1. Category 32](#_Toc150527770)

[4.6.2. Brand 32](#_Toc150527771)

[4.6.3. Product 32](#_Toc150527772)

[4.6.4. User 33](#_Toc150527773)

[4.6.5. Order 33](#_Toc150527774)

[4.6.6. OrderDetail 34](#_Toc150527775)

[4.6.7. Banner 34](#_Toc150527776)

[4.7. Thiết kế giao diện 34](#_Toc150527777)

[4.7.1. Phía quản trị 34](#_Toc150527778)

[CHƯƠNG 5. ĐÁNH GIÁ KẾT QUẢ 39](#_Toc150527779)

[5.1. Kết quả đạt được 39](#_Toc150527780)

[5.1.1. Ứng dụng công nghệ 39](#_Toc150527781)

[5.1.2. Chức năng đạt được 39](#_Toc150527782)

[5.2. Hạn chế của đề tài 39](#_Toc150527783)

[5.3. Hướng phát triển của đề tài 39](#_Toc150527784)

[TÀI LIỆU THAM KHẢO 40](#_Toc150527785)

# TỔNG QUAN VỀ ĐỀ TÀI

## Lí do chọn đề tài

Hiện nay, với sự phát triển mạnh mẽ của cuộc cách mạng công nghiệp 4.0, mà tiêu biểu là sự phát triển của công nghệ thông tin, việc ứng dụng công nghệ hiện đại vào các lĩnh vực kinh tế ngày càng trở nên phổ biến, bởi nó không chỉ giúp công việc đơn giản, hiệu quả mà còn vô cùng chính xác, mang lại lợi ích sản xuất cao.

Trong cuộc sống hối hả, tất bật thì việc sử dụng thời gian và quản lý thời gian hợp lý là vô cùng cần thiết. Chiếc điện thoại ngày càng trở thành vật dụng không thể thiếu đối với nhiều người. Không chỉ thế, với sự phát triển nhanh chóng của nền kinh tế nước nhà, mức sống của người dân ngày càng cao, nhu cầu mua sắm ngày càng tăng.

Việc kết hợp công nghệ thông tin vào việc bán hàng đã cho ra đời hình thức bán hàng mới – bán hàng online. Bán hàng online giúp cho chúng ta tiếp cận với nhiều khách hàng hơn. Cùng với đó là sự giảm chi phí mặt bằng do không cần quá nhiều không gian để trưng bày sản phẩm. Việc mua hàng online cũng đem lại nhiều lợi ích cho khách hàng. Chúng ta có thể mua sắm mà không cần phải đi ra ngoài. Việc đặt hàng cũng như lựa chọn sản phẩm diễn ra nhanh chóng và thuận tiện hơn.

Với những lợi ích trên nên em đã quyết định lựa chọn đề tài “Xây dựng website bán điện thoại”. Nhằm cung cấp cho người bán hàng những lợi ích khi kết hợp công nghệ thông tin vào việc bán hàng. Cũng như những lợi ích của khách hàng trong việc mua hàng online.

## Mục tiêu của đề tài

Ứng dụng công nghệ ASP.NET Core, ReactJS, SQLServer, … vào việc xây dựng website thương mại điện tử B2C.

## Dự kiến kết quả đạt được

Xây dựng được website bán hàng với các chức năng cơ bản như xem sản phẩm, đặt hàng, thanh toán, … và vận hành hệ thống trên môi trường thực tế.

# CƠ SỞ LÍ THUYẾT

## RESTful Api

### Giới thiệu

#### REST

Viết tắt của Representational State Transfer, là một kiến trúc phần mềm để kích hoạt các dịch vụ (service) qua Web. Được giới thiệu vào năm 2000 bởi Roy Fielding, một trong những tác giả của HTTP, REST được sử dụng để yêu cầu (request) các dịch vụ (service) và trả lại phản hồi (response) qua Web, thường được gọi là các Web service.

REST là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng.

#### RESTful

Là một ứng dụng web chạy theo cơ sở hạ tầng REST cung cấp khả năng tương tác giữa các hệ thống khác nhau trên toàn bộ mạng.

#### 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. Nói là giao diện nhưng thực tế không có giao diện nào để có thể nhìn được mà nó thông thường là những URL, function, … giúp các ứng dụng giao tiếp với nhau (sử dụng được những chức năng của nhau hoặc trao đổi thông tin với nhau).

Có thể chia API thành 3 loại:

* Web API: sử dụng trên các hệ thống web.
* API trên hệ điều hành: Windows hay Linux có rất nhiều API, họ cung cấp các tài liệu API là đặc tả các hàm, phương thức cũng như các giao thức kết nối. Nó giúp lập trình viên có thể tạo ra các phần mềm ứng dụng có thể tương tác trực tiếp với hệ điều hành.
* API của thư viện phần mềm hay framework: API mô tả và quy định các hành động mong muốn mà các thư viện cung cấp. Một API có thể có nhiều cách triển khai khác nhau và nó cũng giúp cho một chương trình viết bằng ngôn ngữ này có thể sử dụng thư viện được viết bằng ngôn ngữ khác.

Khi nhắc đến API thì hầu như đang nói tới Web API.

#### RESTful API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

### Cách hoạt động

* REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.
* GET (SELECT): Trả về một Resource hoặc một danh sách Resource
* POST (CREATE): Tạo mới một Resource
* PUT (UPDATE): Cập nhật thông tin cho Resource
* DELETE (DELETE): Xoá một Resource

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.



Hình 1 Cách hoạt động của RESTful API

## ASP.NET Core

### Giới thiệu về ASP.NET Core

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. Lần đầu tiên được đưa ra thị trường vào tháng 1 năm 2002 cùng với phiên bản 1.0 của.NET framework, là công nghệ nối tiếp của Microsoft's Active Server Pages (ASP). ASP.NET được biên dịch dưới dạng Common Language Runtime (CLR), cho phép những người lập trình viết mã ASP.NET với bất kỳ ngôn ngữ nào được hỗ trợ bởi.NET language.

ASP.NET Core là một nền tảng mã nguồn mở miễn phí kế thừa từ ASP.NET được phát triển bời Microsoft. Lần đầu tiên được được đưa ra thị trường vào tháng 6 năm 2016 cùng với phiên bản 1.0 của .NET Core.

Ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework. Nó được thiết kế để cung cấp và tối ưu development framework cho những dụng cái mà được triển khai trên đám mây (clound) hoặc chạy on-promise.

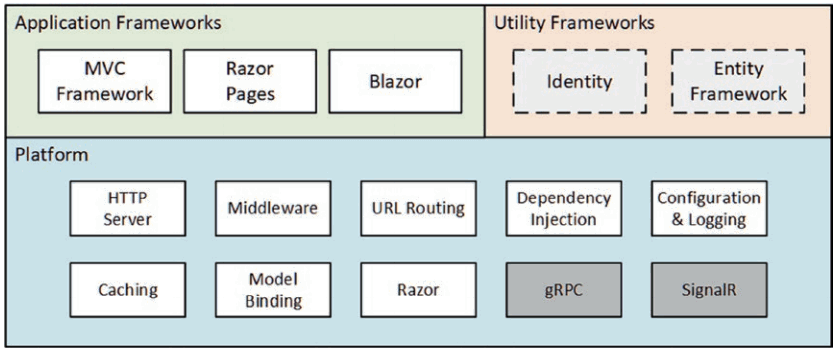
ASP.NET Core được sử dụng để phát triển website và có thể tương với nhiều trình duyệt khác nhau trên các hệ điều hành như Windows, Mac và Linux.

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **ASP.NET Core** | **ASP.NET** |
| **Lịch sử** | Phiên bản đầu tiên được giới thiệu vào tháng 6 năm 2016 cùng với bản .Net Core 1.0 | Ra đời vào tháng 1 năm 2002 cùng với bản .Net Framwork 1.0 |
| **Phát triển** | Microsoft | Microsoft |
| **Hệ điều hành** | Hỗ trợ trên các hệ điều hành Windows, macOS, hoặc Linux | Chỉ hỗ trợ Windows |
| **Công cụ phát triển** | Visual Studio, Visual Studio for Mac, và Visual Studio Code | Visual Studio |
| **Ngôn ngữ lập trình** | C#, F# | C#, VB, F# |
| **Các phiên bản đã phát hành** | 1.0, 1.1, 2.0, 2.1 long-term support, 2.2, 3.0, 3.1 long-term support, 5.0, 6.0, 7.0, 8.0 | 1.0, 1.1, 2.0, 3.0, 3.5, 3.5 Service Pack 1, 4.0, 5.0, 4.5.1, 4.6, 4.7, 4.7.1, 4.8 |

Bảng 1 So sánh ASP.NET Core và ASP.NET

### Cấu trúc của ASP.NET Core

ASP.NET Core là một hệ thống phức tạp dành cho phát triển nhiều loại ứng dụng khác nhau hoạt động trên web. Do vậy, ASP.NET Core bao gồm nhiều thành phần với vai trò khác nhau nhưng tương tác trong một khuôn khổ chung.



Hình 2 Cấu trúc của ASP.NET Core

#### Application Frameworks

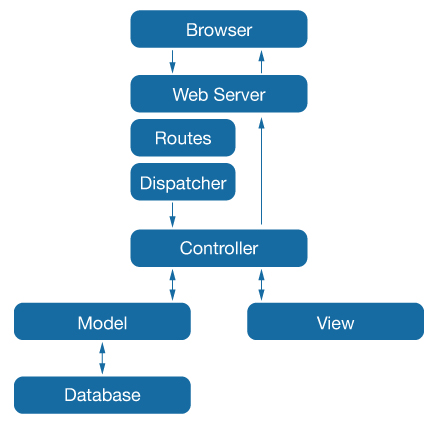
**MVC Framework:** xây dựng theo nguyên lý phân chia nhiệm vụ. MVC phân chia chức năng các thành phần của dứng dụng ra làm các nhóm độc lập, gọi là **M**odel, **V**iew và **C**ontroller. Trong ASP.NET Core, cùng với sự phổ biến của ứng dụng Single-Page, MVC không còn đóng vai trò một framework chính nữa.

MVC gồm 3 thành phần bao gồm:

* M là Model: nơi chứa nghiệp vụ tương tác với dữ liệu, nó bao gồm các class/function xử lý nhiều nghiệp vụ như kết nối database, truy vấn dữ liệu, thêm, xoá, sửa dữ liệu, …
* V là View: là nơi chứa giao diện, hiển thị dữ liệp giúp người dùng tương tác với hệ thống.
* C là Controller: là nơi tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó gồm các class/function xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu cần thiết nhờ các nghiệp vụ ở lớp model cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp View.

Sự tương tác giữa các thành phần:

* Controller tương tác với Model.
* Controller tương tác với View.
* Model và View không có sự tương tác với nhau mà nó tương tác thông qua Controller.



Hình 3 Mô hình MVC

**Razor Pages:** một trong các framework dành cho xây dựng ứng dụng web bên trên ASP.NET Core. Razor Pages cho phép trộn HTML và C# (gọi là cú pháp Razor) vào cùng một file (có đuôi cshtml) để dễ dàng tạo ra HTML theo logic của chương trình. Nhờ vậy dữ liệu HTML tạo ra “động” chứ không cố định như ở các trang web “tĩnh” thiết kế sẵn.

**Blazor:** một framework giúp phát triển ứng dụng đơn trang (SPA – Single Page Application) nhưng sử dụng ngôn ngữ C#.

#### Utility Frameworks

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

**Entity Framework Core:** giúp ứng dụng tương tác với cơ sở dữ liệu. Entity Frameworks core giúp ánh xạ (hai chiều) giữa các bảng cơ sở dữ liệu với các domain model.

**Identity:** là framework dành cho xác thực (authentication) và xác minh quyền (authorization) người dùng trong ứng dụng.

#### Platform

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

**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). Kestrel là một HTTP web server mã nguồn mở (open source), đa nền tảng (cross-platform), hướng sự kiện (event-driven) và bất đồng bộ (asynchronous I/O). Nó được phát triển để chạy ứng dụng ASP.NET Core trên bất cứ nền tảng nào. Nó được thêm vào mặc định trong ứng dụng ASP.NET Core.

**Middleware:** Một Middleware là một module code nó nhận yêu cầu gửi đến Request và trả về Response. Cụ thể trong ASP.NET Core, middleware có thể:

* Nhận một HTTP Request gửi đến và phát sinh ra HTTP Response để trả về.
* Nhận một HTTP Request gửi đến, thi hành một số tác vụ (có thể là sửa đổi HTTP Request), sau đó chuyển đến một middleware khác.
* Nhận HTTP Response, sửa nó và chuyển đến một Middleware khác.

**URL Routing:** cơ chế ánh xạ chuỗi truy vấn HTTP sang thực thi một phương thức nào đó. Do vậy, mỗi URL sẽ tương ứng với thực thi một phương thức trên server.

**Dependency Injection:** 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.

**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.

**Model Binding:** Trong ứng dụng ASP.NET Core thì các trang Razor Page hoặc các Controller (MVC) cần các dữ liệu gửi bởi truy vấn, từ các thông tin gửi đến như chuỗi query của url, tham số route, dữ liệu post của Form ... ứng dụng có thể đọc được những dữ liệu này, ta có thể hoàn toàn trích xuất được những dữ liệu này bằng cách sử dụng các phương thức cung cấp bởi đối tượng HttpRequest (thuộc tính Request của Controller hoặc của PageModel).

ASP.NET Core cung cấp một cơ chế giúp bạn tự động quá trình lấy dữ liệu trên gọi là Model Biding, nó hoạt động tự động qua mấy bước chính:

* Lấy các dữ liệu từ các nguồn gửi đến (Route data, Form data, chuỗi Query, File tải lên).
* Tiến hành chuyển và thiết lập dữ liệu đó cho Razor Page hoặc Controller.
* Chuyển đổi chuỗi dữ liệu nhận được thành các kiểu phù hợp với .NET.

**Razor:** cơ chế sinh ra HTML từ dữ liệu và logic của hương trình. Razor được gọi là view engine trong ASP.NET Core. Razor sử dụng loại cú pháp đặc biệt kết hợp giữ C# và HTML.

**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.

**SignalR:** làm 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** và **SignalR** hiếm khi phải sử dụng đến, trừ trong những tình huống đặc biệt. Chúng được đưa vào đây chủ yếu vì đóng vai trò nền tảng cho một số trường hợp đặc biệt.

## SQLServer

### Giới thiệu SQLServer

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ được phát triển bởi Microsoft. Là một máy chủ cơ sở dữ liệu, nó là một sản phẩm phần mềm có chức năng chính là lưu trữ và truy xuất dữ liệu theo yêu cầu của các ứng dụng phần mềm khác. Có thể chạy trên cùng một máy tính hoặc trên một máy tính khác trên mạng (bao gồm cả Internet).

Phần mềm được Microsoft phát triển dựa trên RDBMS. Đây cũng là một hệ quản trị cơ sở dữ liệu quan hệ đối tượng (ORDBMS). SQLServer có thể sử dụng giao diện dòng lệnh và giao diện GUI.

SQLServer hỗ trợ ngôn ngữ SQL (trước đây là SEQUEL – ngôn ngữ truy vấn tiếng anh có cấu trúc) – vốn là sản phẩm của IBM.

### Các thành phần của SQLServer

SQLServer hoạt động theo mô hình khách – chủ, do đó sẽ có 2 thành phần là Workstation và Server.

Workstation được cài trên bất kì thiết bị nào hoặc trên thiết bị của người vận hành máy chủ. Đây là các giao diện phần mềm để tương tác với Server, ví dụ như SSMS, SSCM, Prifiler, BIDS, SQLEM, …

Server được cài trên máy chủ tập trung. Nó chính là các dịch vụ như SQL Server, SQL Server Agent, SSIS, SSAS, SSRS, SQL Browser, SQL Full Text Search, …

## React JS

### Giới thiệu về React JS

React (Hay ReactJS, React.js) là một thư viện Javascript mã nguồn mở để xây dựng các thành phần giao diện có thể tái sử dụng. Nó được tạo ra bởi Jordan Walke, một kỹ sư phần mềm tại Facebook. Người bị ảnh hưởng bởi XHP (Một nền tảng thành phần HTML cho PHP). React lần đầu tiên được triển khai cho ứng dụng Newsfeed của Facebook năm 2011, sau đó được triển khai cho Instagram.com năm 2012. Nó được mở mã nguồn (open-sourced) tại JSConf US tháng 5 năm 2013.

React được sử dụng rộng rãi và có hệ sinh thái đa dạng phong phú. UI tất nhiên là quan trọng, nhưng không phải là tất cả. Để phát triển ứng dụng hoàn chỉnh, một mình React Js là không làm được tất cả, ReactJS sẽ cần kết hợp thêm:

* HTML/CSS: nếu làm ứng dụng web.
* Flux/Redux: là một kiến trúc giúp tổ chức code rõ ràng và sạch sẽ.

Nó được sử dụng bởi các công ty lớn, đã thành lập và các công ty mới thành lập như: Netflix, Airbnb, Instagram và New York Times, ... ReactJS mang lại nhiều lợi thế cho các lập trình viên, khiến nó trở thành một lựa chọn tốt hơn so với các Framework khác như Angular.

### Các thành phần của React JS

#### JSX

Trong React, thay vì thường xuyên sử dụng JavaScript để thiết kế bố cục trang web thì sẽ dùng JSX. JSX được đánh giá là sử dụng đơn giản hơn JavaScript và cho phép trích dẫn HTML cũng như việc sử dụng các cú pháp thẻ HTML để render các subcomponent. JSX tối ưu hóa code khi biên soạn, vì vậy nó chạy nhanh hơn so với code JavaScript tương đương.

Các đoạn mã được return trong component trông sẽ giống HTML nhưng thực chất đó là một đoạn jsx, jsx cho phép chúng ta kết nối giữa HTML và JavaScript trong cùng một source.

#### Component

Hoạt động giống như các hàm và trả về các thành phần HTML.

Các component là thành phần độc lập và có thể tái sử dụng lại nhiều lần.

Các component thực hiện công việc giống như các function trong JavaScript nhưng chúng độc lập và nhiệm vụ chính là trả về HTML thông qua hàm render.

Có 2 loại component:

* Stateless Component (functional component): thực chất là 1 function có nhiệm vụ return về một đoạn mã HTML hiển thị ra giao diện. Stateless không thể sử dụng được state và component lifecycle.
* Stateful Component (class component): là một class có chứa phương thức render. Khi component được gọi, hàm render sẽ chạy và trả về nội dung HTML bên trong. Statefull sử dụng được cả state và lifecycle.

#### State

Trong React, State được dùng để biểu diễn trạng thái của Component. Tức là giá trị của State có thể thay đổi được, trái ngược hoàn toàn với Props là 1 giá trị bất biến.

#### Props

Props là thuộc tính mặc định của component để nhận dữ liệu từ component cha truyền từ ngoài vào và nó không thể bị chỉnh sửa bên trong component con.

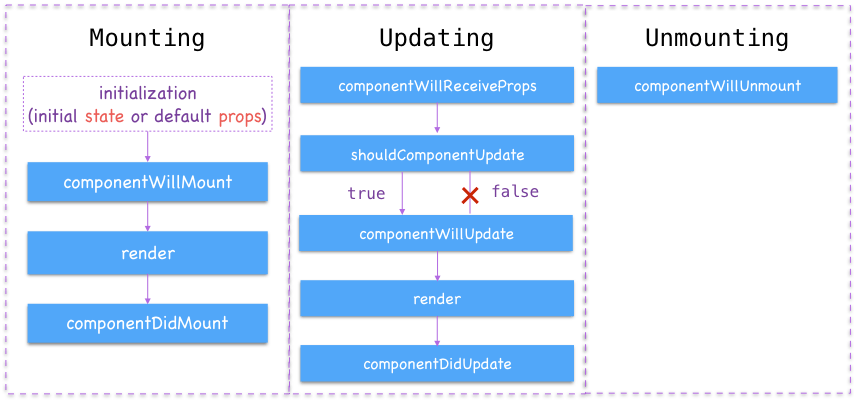
#### React Lifecycle (Vòng đời component)

Các phương thức trong React Component Lifecycle có thể chia ra làm 3 pha chính là: Mounting, Updating và UnMounting.

**Mounting (hay còn gọi là Creation):** bao gồm các phương thức sẽ được khởi chạy lần lượt khi component được khởi tạo và chỉ sử dụng được với stateful.

**Updating:** Updating là giai đoạn khi React Component cần cập nhật giao diện mỗi khi props hoặc state của nó thay đổi.

**UnMounting:** Giai đoạn này sẽ clear tất cả những thứ liên quan đến component này trước khi remove nó.



Hình 4 Vòng đời component

## TailwindCSS

Tailwind css là một utility-first CSS framework nó hỗ trợ phát triển xây dựng nhanh chóng giao diện người dùng, nó cũng có điểm chung giống như Bootstrap & điểm làm nó nổi bật hơn cả đó là chúng ta có thể tùy biến phát triển css theo cách mà chúng ta định nghĩa ra.

Tailwind CSS đã dựng sẵn các class CSS và cả các class responsive và nhiều tiện ích khác mà chúng ta có thể sử dụng dễ dàng vào HTML và không cần viết một dòng CSS nào. Ngược lại, cách làm này lại không có chúng ta những class dựng sẵn theo kiểu của Bootstrap (đã dựng sẵn các thành phần), mà mọi thứ chúng ta bắt buộc phải tự dựng hoàn toàn bằng các class tiện ích này.

**Ưu điểm:**

* Viết CSS custom cực kì ít.
* Không cần viết 1 dòng css nào mà chỉ cần thêm class để tạo giao diện, không cần phải đau đầu suy nghĩ tên class để đặt cho các div.
* Các tên class của tailwind rất khoa học, dễ hiểu và dễ sử dụng, mỗi class là một thuộc tính.
* TailwindCSS giúp cải thiện hiệu suất vì giảm thiểu các việc trùng lặp thuộc tính.
* Có nhiều plugin hỗ trợ loại bỏ các class thừa không sử dụng, giảm thiểu việc có mặt trong việc khai báo ở class HTML.

**Nhược điểm:**

* Số lượng class cho mỗi tag có thể rất nhiều.
* Phải vững kiến thức về CSS.
* Không đủ hết 100% thuộc tính CSS.

# KHẢO SÁT VÀ PHÂN TÍCH CHỨC NĂNG

## Khảo sát hệ thống

Admin: là người quản trị web, admin đăng nhập tài khoản có toàn quyền sử dụng hệ thống.

## Phân tích yêu cần hệ thống

### Yêu cầu chức năng

#### Yêu cầu phía quản trị

Quản lý các đơn hàng.

Quản lý doanh thu.

Quản lý sản phẩm.

Quản lý danh mục sản phẩm.

Quản lý thương hiệu sản phẩm.

Quản lý tin tức.

### Yêu cầu phi chức năng

Giao diện: dễ sử dụng, trực quan, thân thiện với người dùng.

Tốc độ xử lý: hệ thống xử lý nhanh, chính xác.

Tương thích: tương thích với nhiều trình duyệt và thiết bị.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## Danh sách Actor

|  |  |  |
| --- | --- | --- |
| STT | Tên Actor | Ý nghĩa |
| 1 | Admin | * Là người quản trị website |

Bảng 2 Danh sách Actor

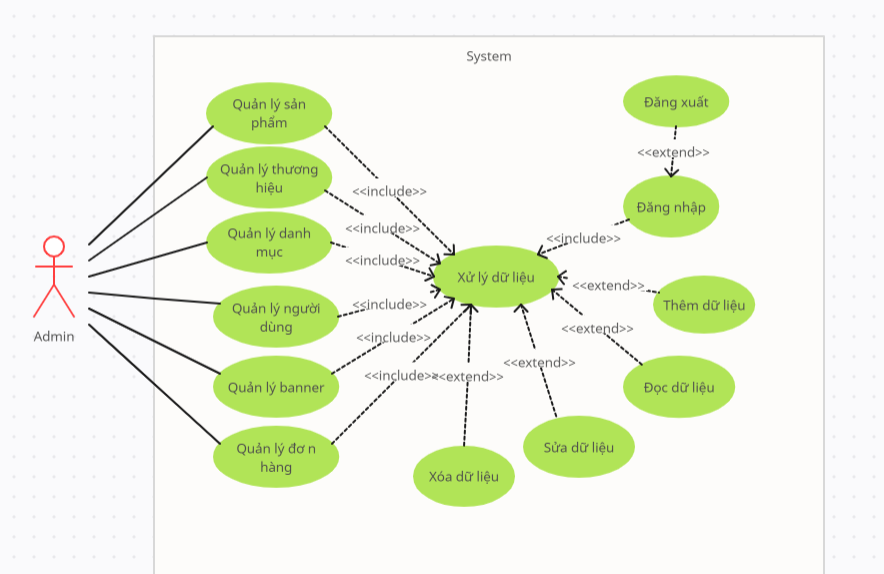
## Danh sách Use Case

|  |  |  |
| --- | --- | --- |
| ID | Tên Use Case | Mô tả |
| UC1 | Đăng nhập | Khách hàng đăng nhập vào website. |
| UC2 | Đăng xuất | Khách hàng đăng xuất khỏi hệ thống |
| UC3 | Quản lý tài khoản | Khách hàng quản lý tài khoản đã đăng ký |
| UC4 | Quản lý thông tin | Khách hàng quản lý thông tin cá nhân |
| UC5 | Sửa thông tin | Khách hàng chỉnh sửa thông tin cá nhân |
| UC6 | Quản lý đơn hàng | Khách hàng quản lý đơn hàng đã đặt |
| UC7 | Tìm kiếm sản phẩm | Khách hàng tìm kiếm sản phẩm theo chủ đề |
| UC8 | Quản lý sản phẩm | Quản trị quản lý sản phẩm |
| UC9 | Quản lý thương hiệu | Quản trị quản lý thương hiệu |
| UC10 | Quản lý danh mục | Quản trị quản lý danh mục |
| UC11 | Quản lý banner | Quản trị quản lý banner |
| UC12 | Quản lý đơn hàng | Quản trị quản lý đơn đặt hàng |
| UC13 | Quản lý người dùng | Quản trị quản lý người dùng |
| UC14 | Quản lý trang | Quản trị quản lý trang |

Bảng 3 Danh sách use case

## Biểu đồ use case

### Biểu đồ use case sử dụng cho quản trị

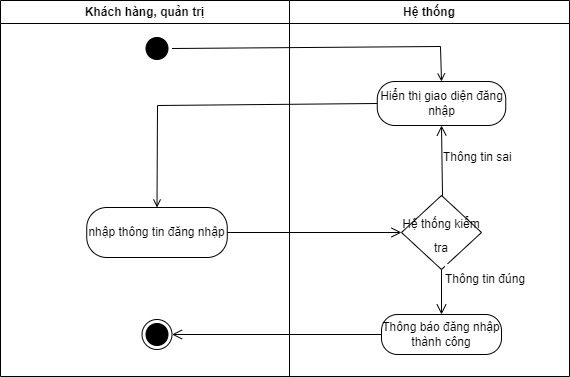


Hình 5 Biểu đồ use case sử dụng cho quản trị

## Biểu đồ hoạt động

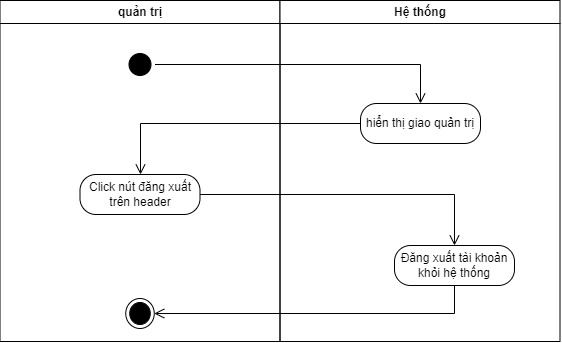
### Phía quản trị

#### Đăng nhập



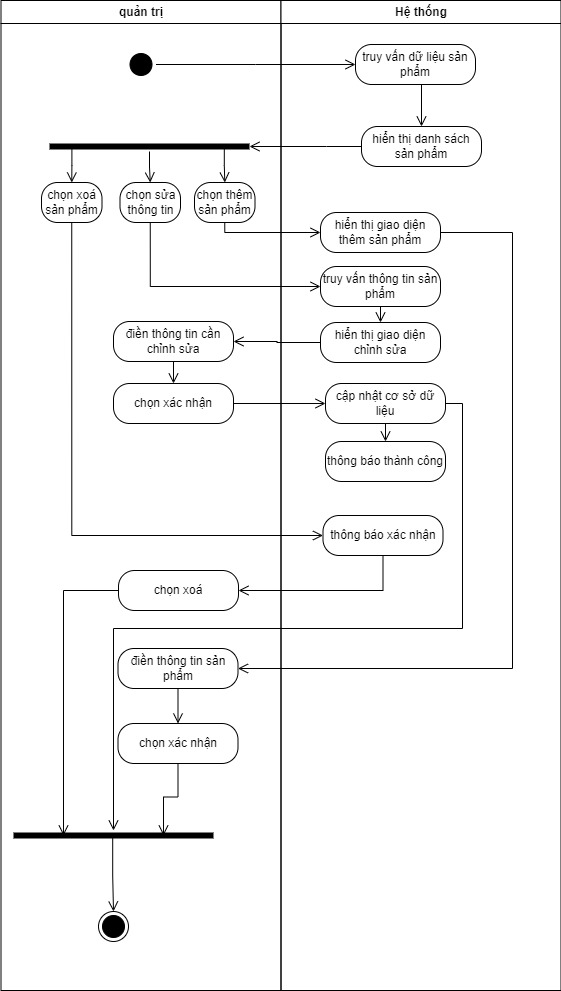
Hình 17 Biểu đồ đăng nhập của quản trị

#### Đăng xuất



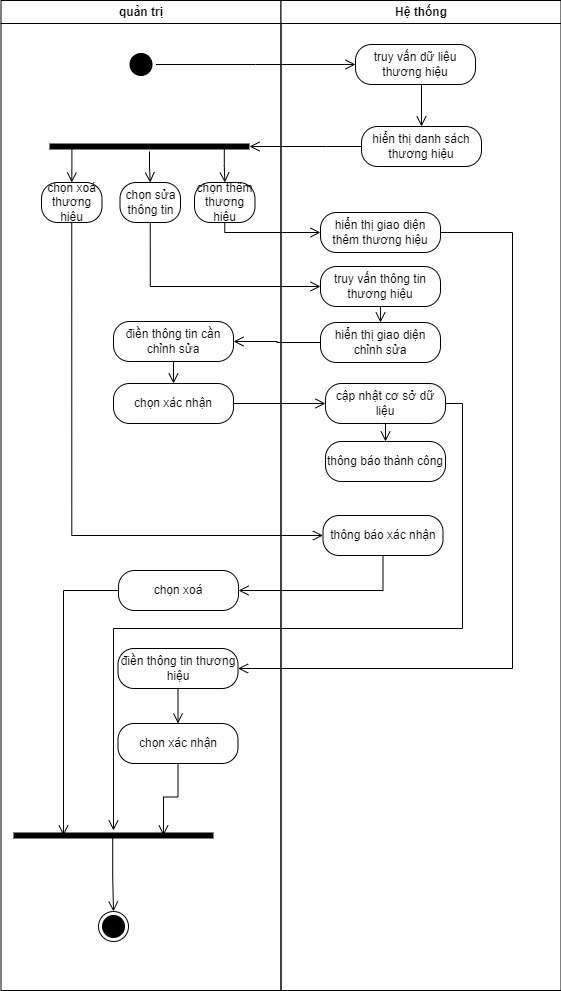
Hình 18 Biểu đồ đăng xuất của quản trị

#### Biểu đồ hoạt động quản lý sản phẩm



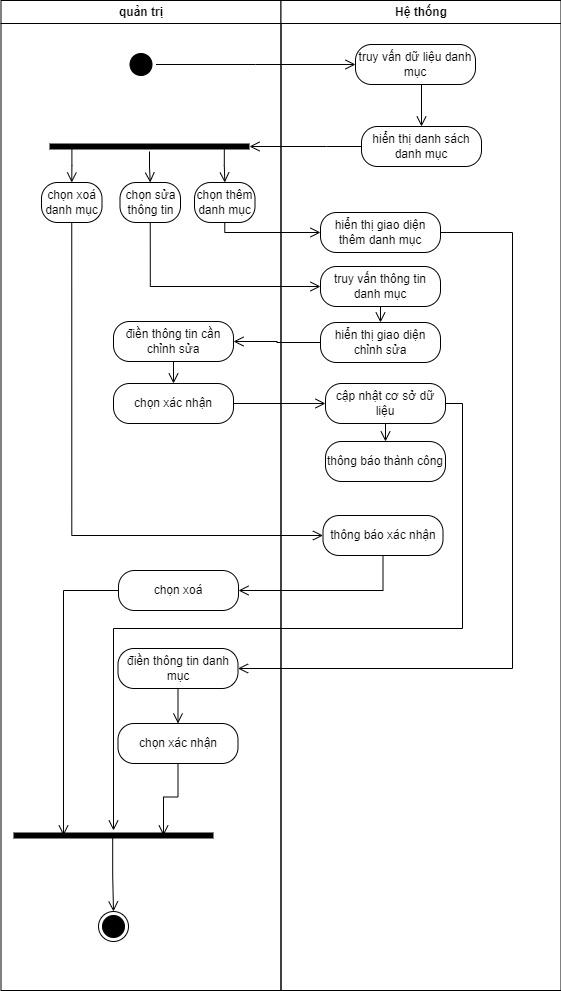
Hình 19 Biểu đồ hoạt động quản lý sản phẩm

#### Biểu đồ hoạt động quản lý thương hiệu



Hình 20 Biểu đồ hoạt động quản lý thương hiệu

#### Biểu đồ hoạt động quản lý danh mục

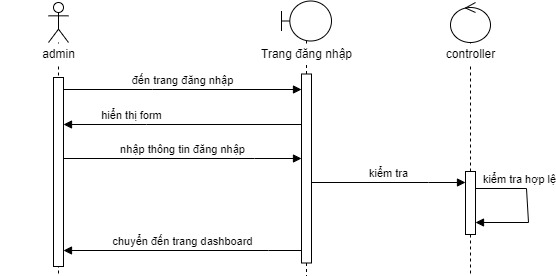


Hình 21 Biểu đồ hoạt động quản lý danh mục

## Biểu đồ tuần tự

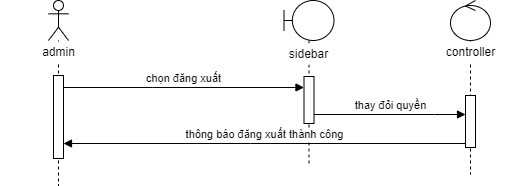
### Phía quản trị

#### Đăng nhập



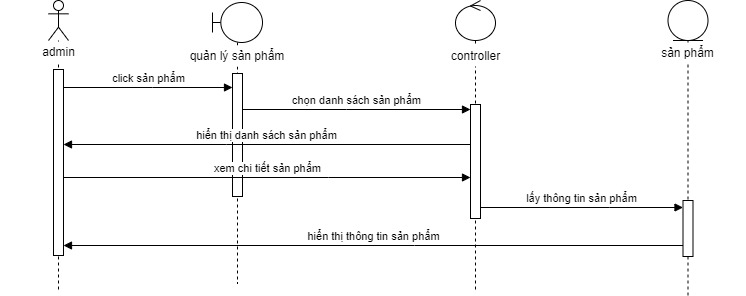
Hình 33 Biểu đồ tuần tự đăng nhập của quản trị

#### Đăng xuất



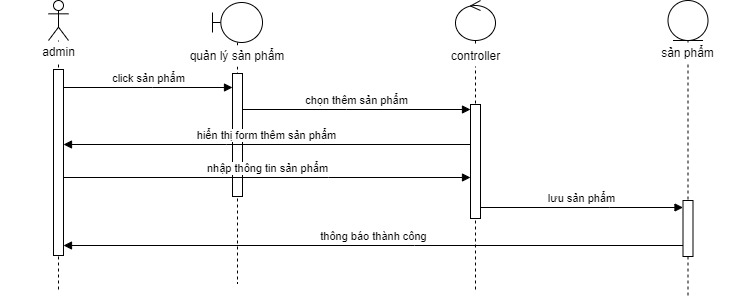
Hình 34 Biểu đồ tuần tự đăng xuất của quản trị

#### Xem thông tin sản phẩm



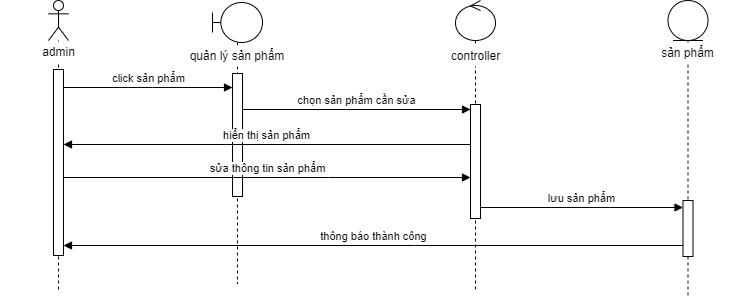
Hình 35 Biểu đồ tuần tự xem thông tin sản phẩm của quản trị

#### Thêm mới sản phẩm



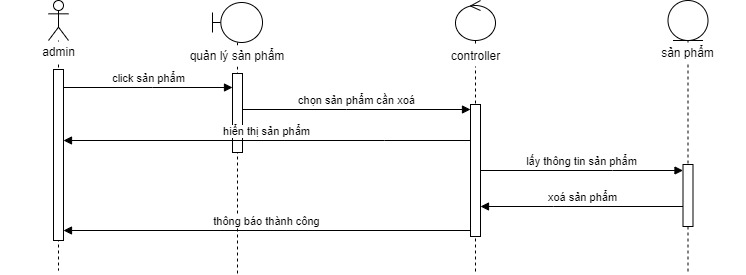
Hình 36 Biểu đồ tuần tự thêm sản phẩm

#### Cập nhật sản phẩm



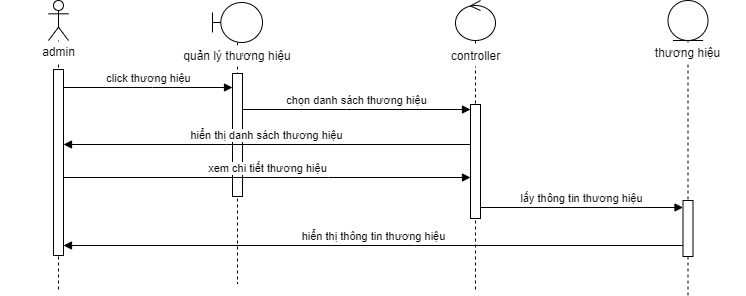
Hình 37 Biểu đồ tuần tự cập nhật sản phẩm

#### Xoá sản phẩm



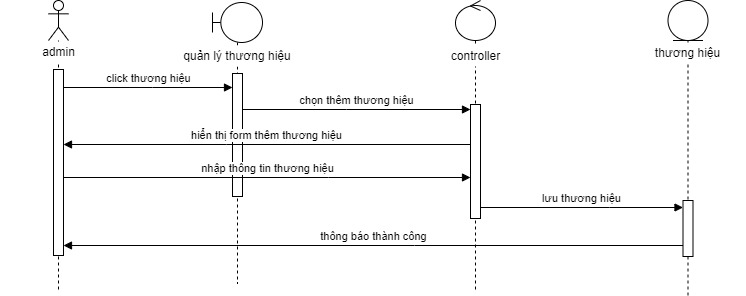
Hình 38 Biểu đồ tuần tự xoá sản phẩm

#### Xem thông tin thương hiệu



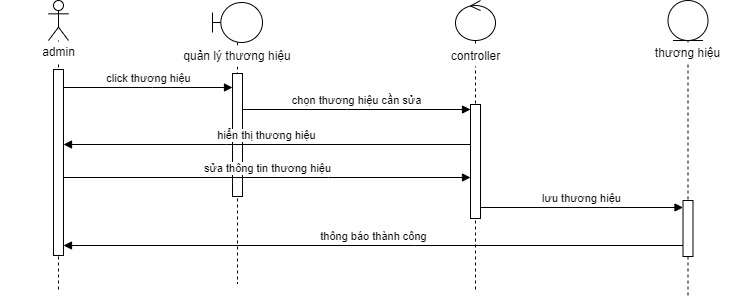
Hình 39 Biểu đồ tuần tự xem thông tin thương hiệu

#### Thêm mới thương hiệu



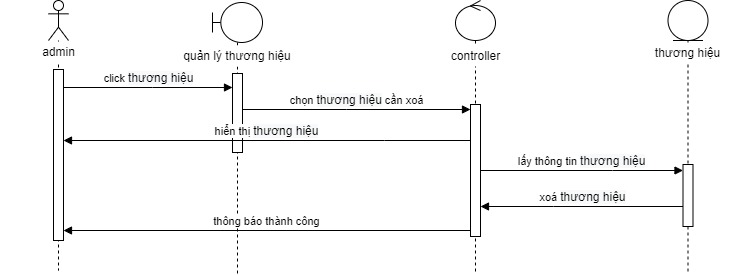
Hình 40 Biểu đồ tuần tự thêm thương hiệu

#### Cập nhật thương hiệu



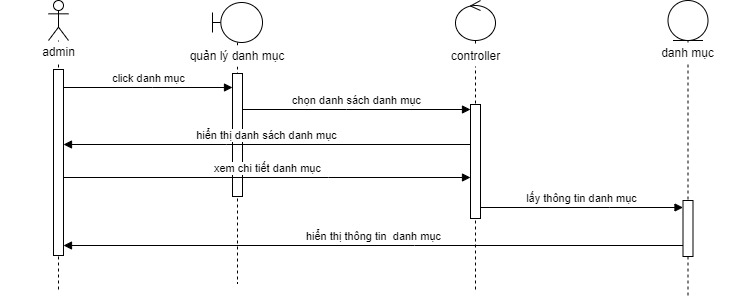
Hình 41 Biểu đồ tuần tự cập nhật thương hiệu

#### Xoá thương hiệu



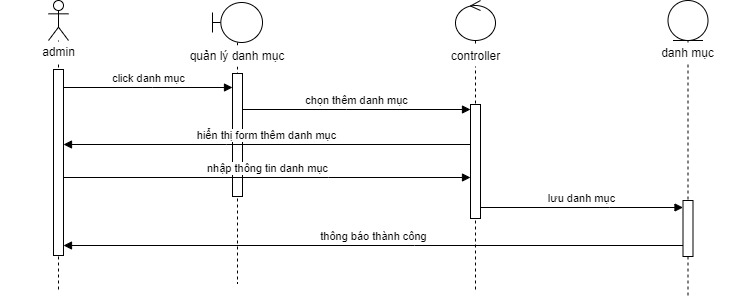
Hình 42 Biểu đồ tuần tự xoá thương hiệu

#### Xem thông tin danh mục



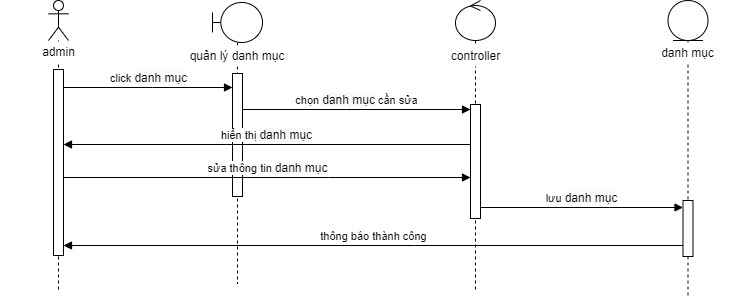
Hình 43 Biểu đồ tuần tự xem thông tin danh mục

#### Thêm danh mục



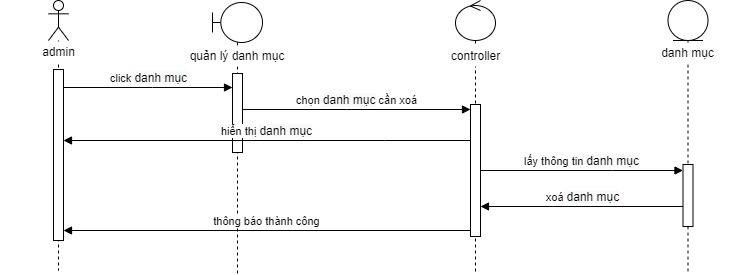
Hình 44 Biểu đồ tuần tự thêm danh mục

#### Sửa danh mục



Hình 45 Biểu đồ tuần tự sửa danh mục

#### Xoá danh mục



Hình 46 Biểu đồ tuần tự xoá danh mục

## Thiết kế cơ sở dữ liệu

### Category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Loại** | **Kiểu** | **Ràng buộc** |
| **1** | Id | Khoá chính | int | not null |
| **2** | Name |  | nvarchar | not null |
| **3** | Slug |  | nvarchar | not null |
| **4** | Parent\_Id |  | int | not null |
| **5** | Sort\_order |  | int | not null |
| **6** | Status |  | int | not null |
| **7** | CreatedAt |  | datetime2 | not null |

Bảng 4 Danh mục

### Brand

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Loại** | **Kiểu** | **Ràng buộc** |
| **1** | Id | Khoá chính | int | not null |
| **2** | Name |  | nvarchar | not null |
| **3** | Slug |  | nvarchar | not null |
| **4** | Image |  | nvarchar | not null |
| **5** | Sort\_order |  | int | not null |
| **6** | Status |  | int | not null |
| **7** | CreatedAt |  | datetime2 | not null |

Bảng 5 Thương hiệu

### Product

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Loại** | **Kiểu** | **Ràng buộc** |
| **1** | Id | Khoá chính | int | not null |
| **2** | Name |  | nvarchar | not null |
| **3** | Slug |  | nvarchar | not null |
| **4** | Price |  | int | not null |
| **5** | Price\_Sale |  | int | not null |
| **6** | Qty |  | int | not null |
| **7** | Detail |  | nvarchar | not null |
| **8** | BrandId | Khoá ngoại | int | not null |
| **9** | CategoryId | Khoá ngoại | int | not null |
| **10** | Image |  | nvarchar | not null |
| **11** | Status |  | int | not null |
| **12** | CreatedAt |  | datetime2 | not null |

Bảng 6 Sản phẩm

### User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Loại** | **Kiểu** | **Ràng buộc** |
| **1** | Id | Khoá chính | int | not null |
| **2** | Role |  | int | not null |
| **3** | Username |  | nvarchar | not null |
| **4** | Password |  | nvarchar | not null |
| **5** | Status |  | int | not null |
| **6** | CreatedAt |  | datetime2 | not null |

Bảng 7 Người dùng

### Order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Loại** | **Kiểu** | **Ràng buộc** |
| **1** | Id | Khoá chính | int | not null |
| **2** | Name |  | nvarchar | not null |
| **3** | Description |  | nvarchar | not null |
| **4** | Phone |  | int | not null |
| **5** | CreatedAt |  | datetime2 | not null |
| **6** | Status |  | int | not null |

Bảng 11 Đơn hàng

### OrderDetail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Loại** | **Kiểu** | **Ràng buộc** |
| **1** | Id | Khoá chính | int | not null |
| **2** | OrderId | Khoá ngoại | int | not null |
| **3** | ProductId | Khoá ngoại | int | not null |
| **4** | Qty |  | int | not null |
| **5** | CreatedAt |  | datetime2 | not null |

Bảng 12 Chi tiết đơn hàng

### Banner

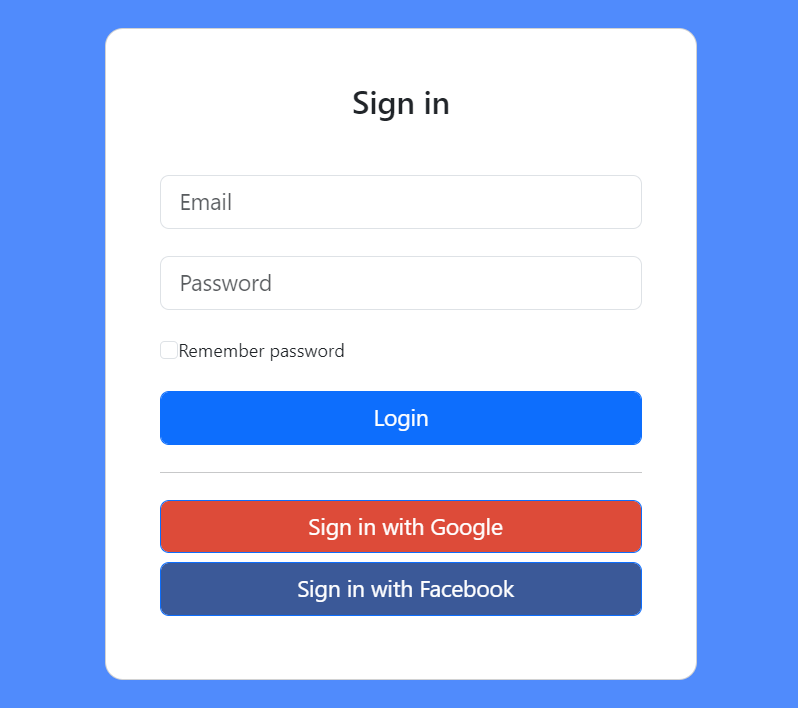
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Loại** | **Kiểu** | **Ràng buộc** |
| **1** | Id | Khoá chính | char | not null |
| **2** | Name |  | nvarchar | not null |
| **3** | Link |  | nvarchar | not null |
| **4** | Image |  | nvarchar | not null |
| **6** | Status |  | int | not null |
| **7** | CreatedAt |  | datetime2 | not null |

Bảng 15 Banner

## Thiết kế giao diện

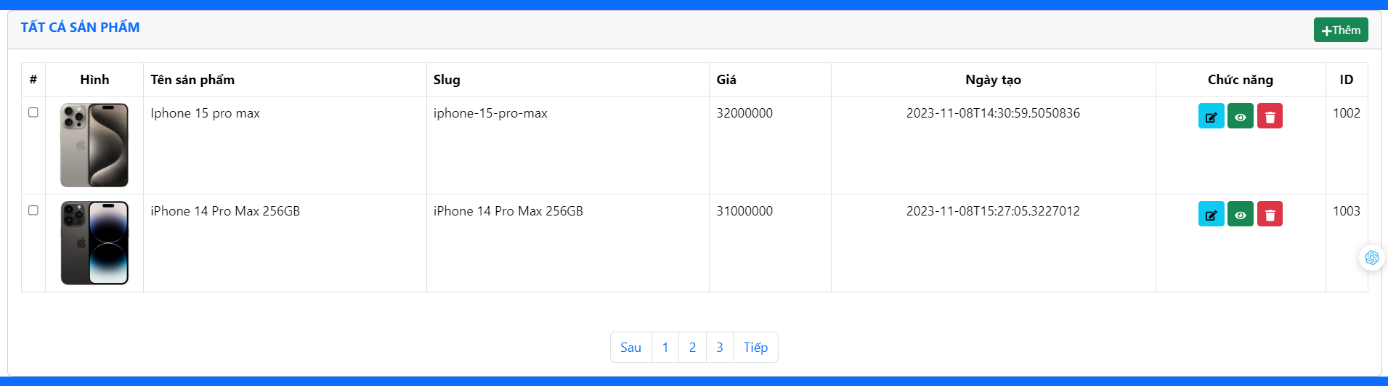
### Phía quản trị

#### Giao diện đăng nhập



Hình 69 Giao diện đăng nhập của quản trị

#### Danh sách sản phẩm



Hình 74 Danh sách sản phẩm

#### Thêm sản phẩm



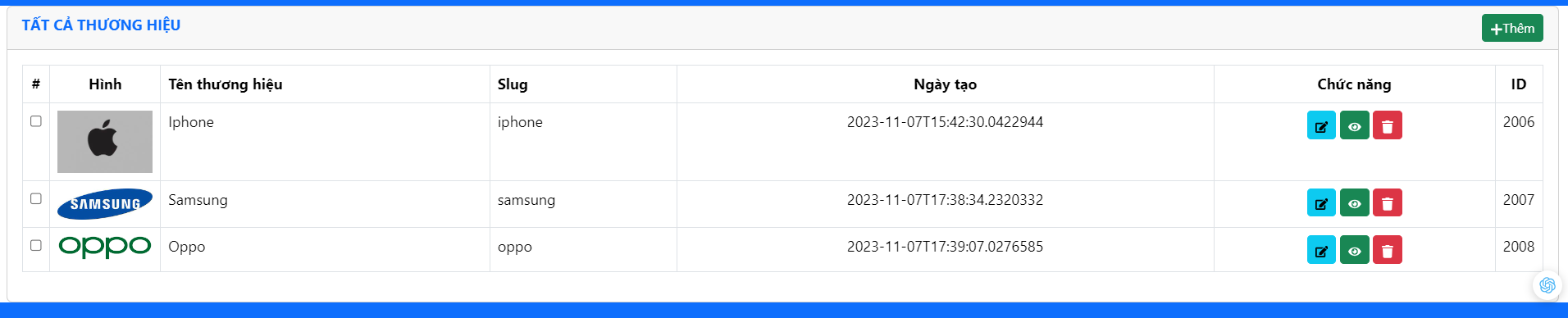
Hình 76 Thêm sản phẩm

#### Cập nhật sản phẩm



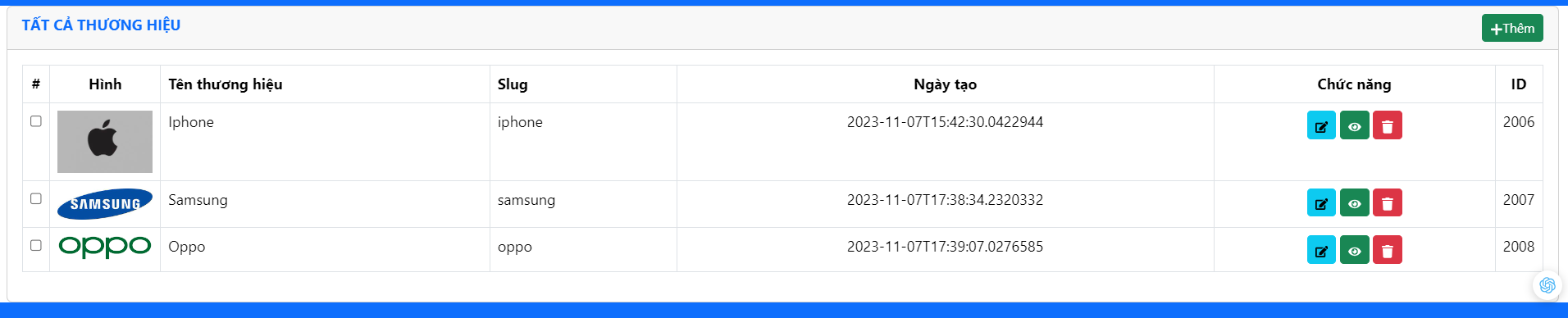
Hình 77 Cập nhật sản phẩm

#### Danh sách thương hiệu



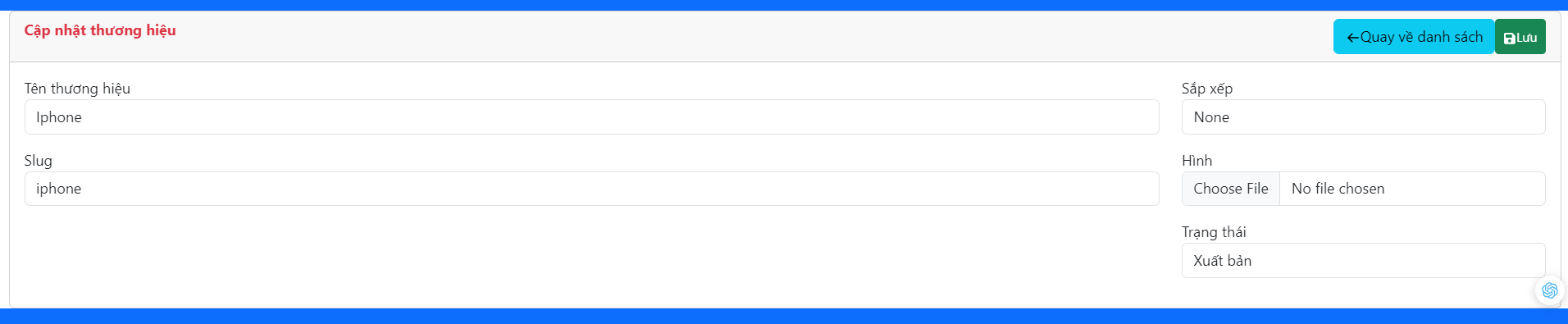
Hình 78 Danh sách thương hiệu

#### Thêm thương hiệu



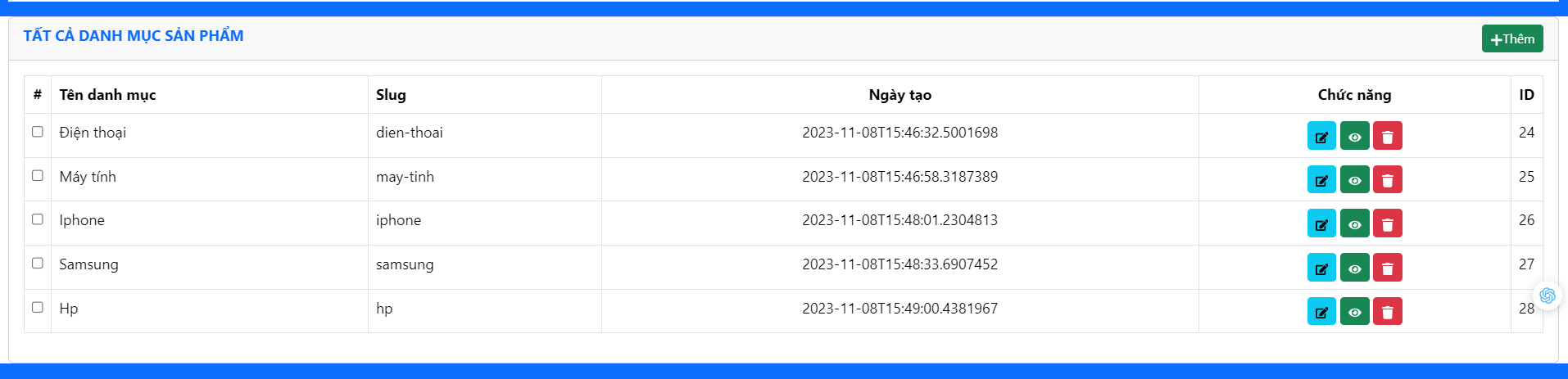
Hình 79 Thêm thương hiệu

#### Cập nhật thương hiệu



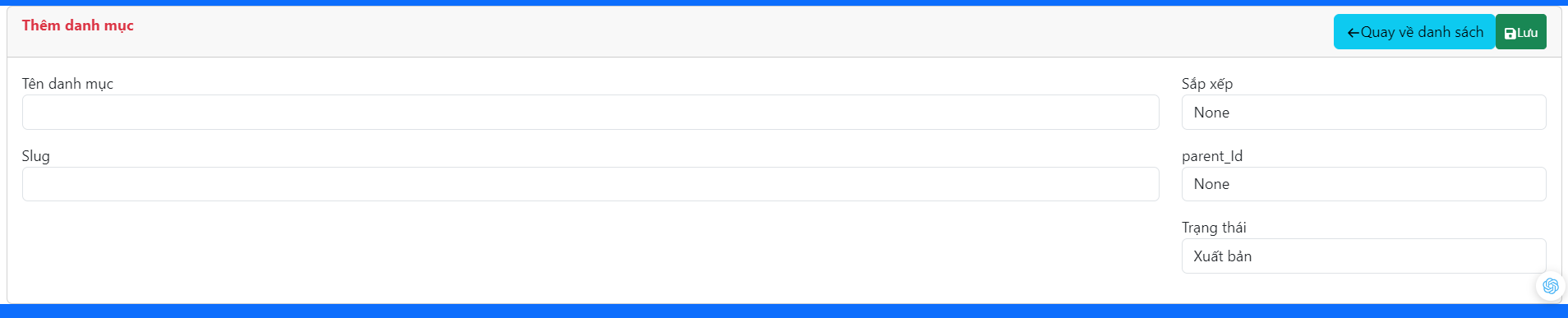
Hình 80 Cập nhật thương hiệu

#### Danh sách danh mục



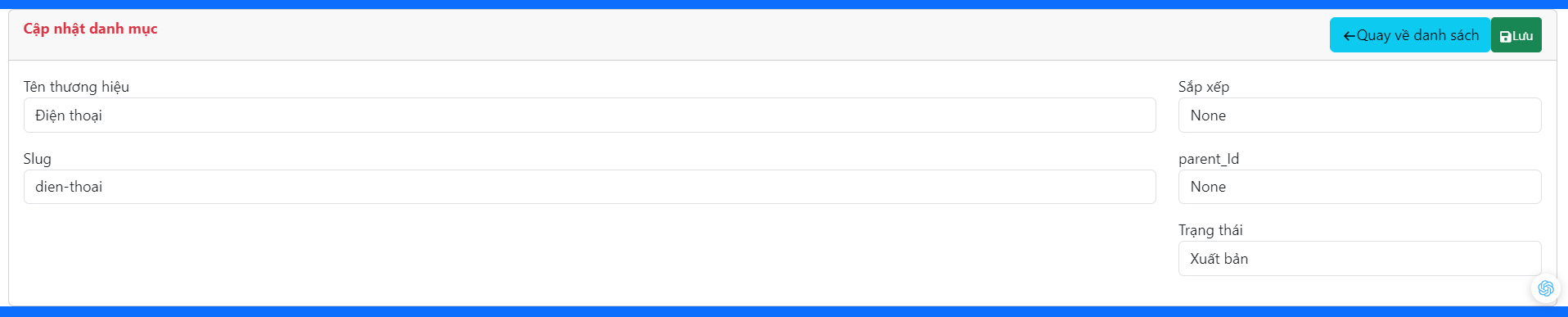
Hình 81 Danh sách danh mục

#### Thêm danh mục



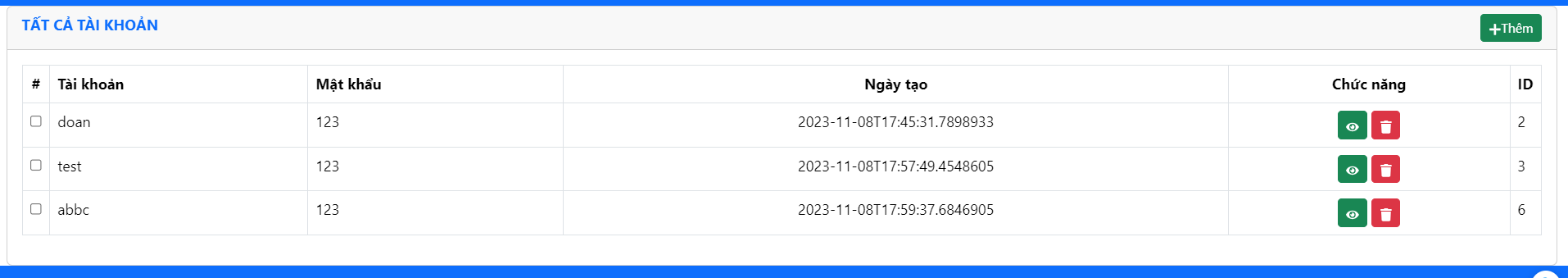
Hình 82 Thêm danh mục

#### Cập nhật danh mục



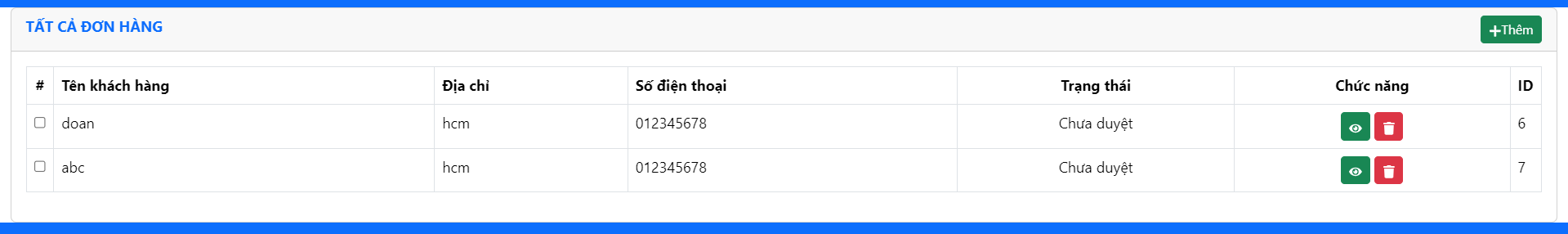
Hình 83 Cập nhật danh mục

#### Danh sách người dùng



Hình 87 Danh sách người dùng

#### Giao diện đặt hàng



Hình 88 Giao diện đặt hàng

# ĐÁNH GIÁ KẾT QUẢ

## Kết quả đạt được

### Ứng dụng công nghệ

Áp dụng thành công các công nghệ ReactJS, TailwindCSS, ASP.NET Core, SQLServer… trong việc xây dựng hệ thống website.

### Chức năng đạt được

**Đối với người quản trị:**

* Xem thông tin của các đề mục cần quản lý như: sản phẩm, thương hiệu, danh mục, đơn đặt hàng…
* Có thể sửa đổi, cập nhật các dữ liệu.
* Xem đơn hàng trong ngày.

## Hạn chế của đề tài

Giao diện chưa bắt mắt kém thu hút người xem.

Hệ thống website chưa được bảo mật.

Một vài chức năng còn lỗi vặt.

## Hướng phát triển của đề tài

Hướng phát triển của Website là tiến đến một website thương mại điện tử với đầy đủ các chức năng thanh toán tiền qua mạng.

Giao diện chưa thu hút được người dùng. Trong tương lai cần thiết kế giao diện đẹp, thân thiện với người dùng, có nhiều chức năng.

Trong thời gian sau này cần bổ sung các chức năng về kiểm kê: Thống kê số lần truy cập database, thống kê giá xuất nhập, tồn kho, thống kê doanh thu, xử lý hóa đơn tự động.

Mặc dù đã cố gắng hoàn chỉnh các yêu cầu nhưng bài báo cáo còn rất nhiều thiếu sót mong nhận được sự chỉ bảo hướng dẫn của các thầy cô khác giúp đỡ xem xét, đề xuất thêm các ý kiến cũng như bổ sung các vấn đề phục vụ cho việc xây dựng Website để em có thể hoàn chỉnh hơn. Em xin cảm ơn các quý thầy cô.

# TÀI LIỆU THAM KHẢO

[1] React Tutorial. [Trực tuyến]. Địa chỉ: https://reactjs.org. [Truy cập ngày 12/03/2022]

[2] ASP.NET Core Tutorial. [Trực tuyến]. Địa chỉ: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio. [Truy cập ngày 12/03/2022]

[3] TailwindCSS Documention. [Trực tuyến]. Địa chỉ: https://tailwindcss.com. [Truy cập ngày 20/03/2022]

[4] Entity Framework Core Tutorial. [Trực tuyến]. Địa chỉ: https://www.entityframeworktutorial.net. [Truy cập ngày 25/03/2022]

[5] HTML CSS Tutorial. [Trực tuyến]. Địa chỉ: https://www.w3schools.com. [Truy cập ngày 12/03/2022]

[6] RESTful API. [Trực tuyến]. Địa chỉ: https://topdev.vn/blog/restful-api-la-gi/. [Truy cập ngày 12/03/2022]

[7] Three Layer [Trực tuyến]. Địa chỉ: https://topdev.vn/blog/mo-hinh-3-lop-la-gi/. [Truy cập ngày 12/03/2022]

[8] f8 Học Lập Trình Để Đi Làm [Trực tuyến]. Địa chỉ: https://fullstack.edu.vn/. [Truy cập ngày 12/03/2022]

[9] Learn ASP.NET Web API [Trực tuyến]. Địa chỉ: https://www.tutorialsteacher.com/webapi. [Truy cập ngày 20/03/2022]