|  |  |
| --- | --- |
|  |  |
| HỌ VÀ TÊN SINH VIÊN: LÊ ĐẮC HẢI | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| ĐATN ĐẠI HỌC NGÀNH CÔNG NGHỆ THÔNG TIN |
|  |
| **XÂY DỰNG WEBSITE BÁN ĐỒNG HỒ ĐEO TAY**  **CHO CỬA HÀNG ĐẮC HẢI** |
|  |
|  |
| **CBHD: Ths. Nguyễn Lan Anh** |
| NGÀNH: CÔNG NGHỆ THÔNG TIN | **Sinh viên: Lê Đắc Hải** |
| **Mã số sinh viên: 2019603085** |
|  |
|  |
|  |
| Hà Nội – 2023 |
|  |

# MỞ ĐẦU

Trong những năm gần đây, cuộc cách mạng 4.0 và ngành công nghệ thông tin đang trên đà phát triển không ngừng, nó có sức ảnh hưởng rất lớn đến cuộc sống của chúng ta. Trong đó phải kể đến đó là thương mại điện tử. Với thương mại điện tử, chúng ta có thể mua sắm bất cứ thứ gì tại bất cứ nơi đâu với điều kiện rất đơn giản đó là 1 chiếc điện thoại thông minh có kết nối Internet. Từ những điều kiện thuận lợi trên, 1 sinh viên năm cuối ngành công nghệ thông tin với sở thích đồng hồ đeo tay như em đã nghĩ đến 1 công việc đó là thiết kế ra 1 trang web bán các loại đồng hồ đeo tay.

Vận dụng những kiến thức đã học và tìm hiểu cùng những công nghệ như .NET CORE 5, ReactJs, SQL Server, …để thiết kế 1 trang web với mục đích tiên quyết là hoàn thành tốt môn ĐATN, sau đó là định hướng phát triển trong tương lai, hoàn thiện hơn, có thể đưa vào sử dụng và hướng đến những khách hàng có công việc bận rộn, không có thời gian đến tận nơi, sờ tận tay những sản phẩm đồng hồ chất lượng mà vẫn có thể chọn mua những chiếc đồng hồ ưng ý nhất mọi lúc mọi nơi có thể trong giờ giải lao, giờ làm việc căng thẳng hoặc giờ nghỉ trưa.

Mục tiêu và kết quả dự kiến của đề tài là 1 trang web bán đồng hồ gồm có các chức năng cơ bản của phần quản trị, khách hàng. Các chức năng chi tiết hơn sẽ cùng tìm hiểu trong bài.

Lê Đắc Hải

# TỔNG QUAN VỀ ĐỀ TÀI

## Mục tiêu bài toán

Chương trình phải đảm bảo các yêu cầu sau:

* Giảm thiểu công sức, thời gian của khách hàng.
* Tìm kiếm nhanh chóng, thuận tiện, chính xác và hiệu quả.
* Lập báo cáo nhanh chóng, giúp nhân viên kinh doanh đưa ra những phương án kinh doanh hợp lý.

### Mục tiêu quản lý

* Theo dõi số lượng sản phẩm bán ra.
* Theo dõi số lượng tồn kho.
* Theo dõi doanh thu của cửa hàng.

### Đầu vào của hệ thống

* Thông tin về danh mục sản phẩm(các hãng đồng hồ).
* Thông tin về sản phẩm(đồng hồ).
* Thông tin về khách hàng.

### Đầu ra của hệ thống

* Thông tin chi tiết đơn hàng.
* Thông tin doanh thu của cửa hàng.

## Yêu cầu bài toán

### Đối với khách hàng

* Dễ hiểu, dễ sử dụng, tìm kiếm dễ dàng hiệu quả.
* Đảm bảo an toàn và bảo mật thông tin khách hàng.

### Đối với nhân viên

* Theo dõi số lượng tồn kho, đã hết.
* Thống kê được doanh số đã bán.
* Theo dõi được quá trình đặt hàng, giao hàng.
* Đưa ra phương án kinh doanh hợp lý.

### Đối với Admin

* Quản lý được người sử dụng.
* Theo dõi số lượng tồn kho, đã hết.
* Thống kê được doanh số đã bán.
* Theo dõi được quá trình đặt hàng, giao hàng.
* Đưa ra phương án kinh doanh hợp lý.

## Các chức năng của hệ thống

### Đối với khách hàng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Chức năng | Người dùng | Mô tả |
| 1 | Đăng ký | Khách hàng | Đăng ký tài khoản |
| 2 | Đăng nhập | Khách hàng | Đăng nhập hệ thống |
| 3 | Xem danh sách SP | Khách hàng | Cho phép xem danh sách sản phẩm |
| 4 | Xem chi tiết SP | Khách hàng | Cho phép xem thông tin chi tiết về sản phẩm |
| 5 | Comment | Khách hàng | Cho phép comment dưới sản phẩm |
| 6 | Thêm vào giỏ hàng | Khách hàng | Cho phép thêm sản phẩm vào giỏ hàng |
| 7 | Đặt hàng | Khách hàng | Cho phép đặt hàng |
| 8 | Hủy đơn hàng | Khách hàng | Cho phép hủy đơn hàng |
| 9 | Sửa thông tin cá nhân | Khách hàng | Cho phép sửa thông tin cá nhân |
| 10 | Xem giỏ hàng | Khách hàng | Cho phép xem giỏ hàng |

### Đối với nhân viên

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Chức năng | Người dùng | Mô tả |
| 1 | Đăng nhập | Nhân viên | Cho phép đăng nhập hệ thống |
| 2 | Xem danh sách KH | Nhân viên | Cho phép xem danh sách khách hàng |
| 3 | Quản lý DMSP | Nhân viên | Cho phép xem, thêm, sửa, xóa danh mục sản phẩm |
| 4 | Quản lý SP | Nhân viên | Cho phép thêm, sửa, xóa SP |
| 5 | Xem danh sách đơn hàng | Nhân viên | Cho phép xem danh sách đơn đặt hàng |
| 6 | Xác nhận đơn hàng | Nhân viên | Cho phép xác nhận đơn hàng |
| 7 | Xác nhận giao hàng | Nhân viên | Cho phép xác nhận giao hàng |

### Đối với admin

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Chức năng | Người dùng | Mô tả |
| 1 | Đăng nhập | Admin | Cho phép đăng nhập hệ thống |
| 2 | Quản lý nhân viên | Admin | Cho phép xem, thêm, sửa, xóa nhân viên |
| 3 | Quản lý DMSP | Admin | Cho phép xem, thêm, sửa, xóa danh mục sản phẩm |
| 4 | Quản lý SP | Admin | Cho phép thêm, sửa, xóa SP |
| 5 | Quản lý đơn hàng | Admin | Cho phép xem danh sách, xác nhận và giao đơn đặt hàng |
| 6 | Thống kê | Admin | Cho phép thống kê doanh thu theo năm |

## Công nghệ và kỹ thuật sử dụng

### .Net core API(.Net 5)

#### .Net Core là gì?

[ASP.NET](http://asp.net/) Core là một nền tảng mã nguồn mở, cross-platform framework cho việc xây dựng và kết nối các ứng dụng web hiện đại chẳng hạn như ứng dụng Web, IoT, Backend-WebAPI.

[ASP.NET](http://asp.net/) Core cung cấp một kiến trúc để tối ưu hóa việc xây dựng các ứng dụng đám mây (microsoft azure) hoặc các ứng dụng độc lập. Bạn có thế phát triển ứng dụng [ASP.NET](http://asp.net/) Core đa nên tảng(Window, Mac, Linux), với mã nguồn mở tại [Github](https://github.com/aspnet/home).

#### Ưu điểm của .Net core so với .Net framework

.NET Core hỗ trợ xây dựng ứng dụng đa nền tảng (window, linux, MacOS), .NET Framework chỉ hỗ trợ trên window.

.NET Core ra đời sau nên Microsoft ra đời các phiên bản mới liên tục, cập nhật theo từng năm (tương lai), .NET Framework ra đời lâu và Microsoft ngừng cập nhật phiên bản(lớn) mới.

.NET Core có nhanh và nhẹ hơn .NET Framework

.NET Core không cần tải toàn bộ framework, chỉ cần cài bộ SDK và khi cần package nào thfi tải xuống nên nhẹ hơn. .NET Framework phải cài toàn bộ framework đầy đủ thư viện lên window nên rất nặng.

.NET Core là mã nguồn mở nên các nhà phát triển (developers) có thể đóng góp để gúp mã nguồn tốt hơn.

### DI(Dependency Injection)

#### Dependency Injection (DI) là gì?

Là một design pattern được [ASP.Net](http://asp.net/) hỗ trợ. Đây là một kỹ thuật để hiện thực hóa Inversion of Control Pattern (có thể coi nó là một design pattern riêng cũng được). Các module phụ thuộc (dependency) sẽ được inject vào module cấp cao. Có thể hiểu 1 cách đơn giản như sau:

* Các module không giao tiếp trực tiếp với nhau, mà thông qua interface. Module cấp thấp sẽ implement interface, module cấp cao sẽ gọi module cấp thấp thông qua interface.
* Ví dụ: Để giao tiếp với database, ta có interface IDatabase, các module cấp thấp là XMLDatabase, SQLDatabase. Module cấp cao là CustomerBusiness sẽ chỉ sử dụng interface IDatabase.
* Việc khởi tạo các module cấp thấp sẽ do DI Container thực hiện. Ví dụ: Trong module CustomerBusiness, ta sẽ không khởi tạo IDatabase db = new XMLDatabase(), việc này sẽ do DI Container thực hiện. Module CustomerBusiness sẽ không biết gì về module XMLDatabase hay SQLDatabase.
* Việc Module nào gắn với interface nào sẽ được config trong code hoặc trong file XML.
* DI được dùng để làm giảm sự phụ thuộc giữa các module, dễ dàng hơn trong việc thay đổi module, bảo trì code và testing.

=> Đó là lý do tại sao cần sử dụng Dependency Injection

#### Ưu điểm

* Giảm sự kết dính giữa các module
* Code dễ bảo trì, dễ thay thế module
* Rất dễ test và viết Unit Test
* Dễ dàng thấy quan hệ giữa các module (Vì các dependency đều được inject vào constructor)

#### Nhược điểm

* Khái niệm DI khá “khó tiêu”, các developer mới sẽ gặp khó khăn khi học
* Sử dụng interface nên đôi khi sẽ khó debug, do không biết chính xác module nào được gọi
* Các object được khởi tạo toàn bộ ngay từ đầu, có thể làm giảm performance
* Làm tăng độ phức tạp của code

### JWT(Json Web Token)

#### **JSON Web Token là gì?**

JSON Web Mã (JWT) là một chuẩn mở (RFC 7519) định nghĩa một cách nhỏ gọn và khép kín để truyền một cách an toàn thông tin giữa các bên dưới dạng đối tượng JSON. Thông tin này có thể được xác minh và đáng tin cậy vì nó có chứa chữ ký số. JWTs có thể được ký bằng một thuật toán bí mật (với thuật toán HMAC) hoặc một public / private key sử dụng mã hoá RSA.

Một ví dụ về JWT Token:

“eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjEzODY4OTkxMzEsImlzcyI6ImppcmE6MTU0ODk1OTUiLCJxc2giOiI4MDYzZmY0Y2ExZTQxZGY3YmM5MGM4YWI2ZDBmNjIwN2Q0OTFjZjZkYWQ3YzY2ZWE3OTdiNDYxNGI3MTkyMmU5IiwiaWF0IjoxMzg2ODk4OTUxfQ.uKqU9dTB6gKwG6jQCuXYAiMNdfNRw98Hw\_IWuA5MaMo”

Thoạt trông phức tạp là thế nhưng nếu hiểu, cấu trúc của một JWT chỉ đơn giản như sau:

<base64-encoded header>.< base64-encoded header payload>.< base64-encoded header signature>.

Nói một cách khác, JWT là sự kết hợp (bởi dấu .) một Object Header dưới định dạng JSON được encode base64, một payload object dưới định dạng JSOn được encode base64 và một Signature cho URI cũng được mã hóa base64 nốt.

Trong đó:

**Header**

Header bao gồm hai phần chính: loại token (mặc định là JWT - Thông tin này cho biết đây là một Token JWT) và thuật toán đã dùng để mã hóa (HMAC SHA256 - HS256 hoặc RSA).

**Payload**

Payload chứa các claims. Claims là một các biểu thức về một thực thể (chẳng hạn user) và một số metadata phụ trợ. Có 3 loại claims thường gặp trong Payload: reserved, public và private claims.

**Signature**

Chữ ký Signature trong JWT là một chuỗi được mã hóa bởi header, payload cùng với một chuỗi bí mật theo nguyên tắc sau:

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload), secret)

Do bản thân Signature đã bao gồm cả header và payload nên Signature có thể dùng để kiểm tra tính toàn vẹn của dữ liệu khi truyền tải.

#### Khi nào nên sử dụng JWT

Một trong những tình huống ứng dụng JWT thường gặp, đó là:

Authentication: Tình huống thường gặp nhất, khi user logged in, mỗi request tiếp đó đều kèm theo chuỗi token JWT, cho phép người dùng có thể truy cập đường dẫn, dịch vụ và tài nguyên được phép ứng với token đó. Single Sign On cũng là một chức năng có sử dụng JWT một cách rộng rãi, bởi vì chuỗi JWT có kích thước đủ nhỏ để đính kèm trong request và sử dụng ở nhiều hệ thống thuộc các domain khác nhau.

Information Exchange: JSON Web Token cũng là một cách hữu hiệu và bảo mật để trao đổi thông tin giữa nhiều ứng dụng, bởi vì JWT phải được ký (bằng cặp public / private key), bạn có thể chắc rằng người gửi chính là người mà họ nói rằng họ là (nói tóm tắt hơn là không hoặc khó để mạo danh bằng JWT), ngoài ra, chữ ký cũng được tính toán dựa trên nội dung của header và nội dung payload, nhờ đó, bạn có thể xác thực được nội dung là nguyên bản, chưa được chỉnh sửa hoặc can thiệp. Tuy nhiên, một lưu ý hết sức quan trọng là do cấu trúc của JWT đơn giản nên JWT có thể dễ dàng bị decode, do vậy, bạn không nên dùng JWT để transfer các thông tin nhạy cảm.

### ReactJS

#### Reactjs là gì?

Reactjs là một thư viện **Javascript** mã nguồn mở hỗ trợ xây dựng các thành phần giao diện nhanh gọn và tiện lợi. Bình thường các lập trình viên sẽ nhúng javascript vào code HTML thông qua các attribute như AngularJS nhưng với Reactjs làm việc như một thư viện cho phép **nhúng HTML vào javascript thông qua JSX.** Qua đó bạn có thể dễ dàng lồng các đoạn HTML vào trong JSX làm cho các component dễ hiểu và dễ sử dụng hơn.

Trong Reactjs thường dùng javascript để thiết kế bố cục cho trang web, nhưng nhược điểm là cấu trúc khá là khó. Thay vào đó sử dụng JSX và nhúng các đoạn HTML vào javascript, ta thấy cú pháp dễ hiểu hơn và JSX cũng có thể tối ưu code khi biên soạn. Vừa dễ cho người lập trình mà vừa tiện cho việc biên dịch.

#### Thành phần cơ bản của Reactjs là gì?

Thành phần cơ bản của React được gọi là **components**. Syntax để viết HTML sử dụng Javascript để render. Bạn có thể tạo ra một component bằng các gọi phương thức createClass của đối tượng React, điểm bắt đầu khi tiếp cận với thư viện này. Có thể lồng nhiều component vào nhau thông qua lệnh return của phương thức render.

Trong một chương trình thì có rất nhiều các component, để đơn giản việc quản lý các component đó người ta sử dụng redux, **redux giống như 1 cái kho chứa các component** và khi dùng component nào thì chỉ cần gọi nó ra.

Virtual DOM không được tạo ra bởi Reactjs nhưng lại được sử dụng rất nhiều. Đây là một chuẩn của W3C được dùng để truy xuất code HTML hoặc XML. Các **Virtual DOM** sẽ được tạo ra khi chạy chương trình, đó là nơi chưa các component. Sử dụng DOM sẽ tiết kiệm được hiệu suất làm việc, khi có thay đổi gì Reactjs đều tính toán trước và việc còn lại chỉ là thực hiện chúng lên DOM.

#### Lợi ích khi sử dụng Reactjs

Làm gì cũng phải có nguyên nhân, người dùng sử dụng Reactjs ngày càng nhiều thì Reactjs phải mang lại những lợi ích tốt.

* ReactJS giúp cho việc viết các đoạn code Javascript sẽ trở nên dễ dàng hơn vì nó sử dụng một cú pháp đặc biệt đó chính là cú pháp JSX. Thông qua JSX cho phép nhúng code HTML và Javascript.
* ReactJS cho phép Developer phá vỡ những cấu tạo UI phức tạp thành những component độc lập. Dev sẽ không phải lo lắng về tổng thể ứng dụng web, giờ đây Developer dễ dàng chia nhỏ các cấu trúc UI/UX phức tạp thành từng component đơn giản hơn.
* Đi kèm với ReactJS là rất nhiều các công cụ phát triển giúp cho việc debug code một cách dễ dàng hơn.
* Một trong những ưu điểm nữa của ReactJS đó là sự thân thiện với SEO. Hầu như các JS Frameworks không thân thiện với các tìm kiếm mặc dù đã được cải thiện nhiều nhưng dưới sự hỗ trợ của các render dữ liệu trả về dưới dạng web page giúp cho SEO chuẩn hơn.

### Thư viện axios

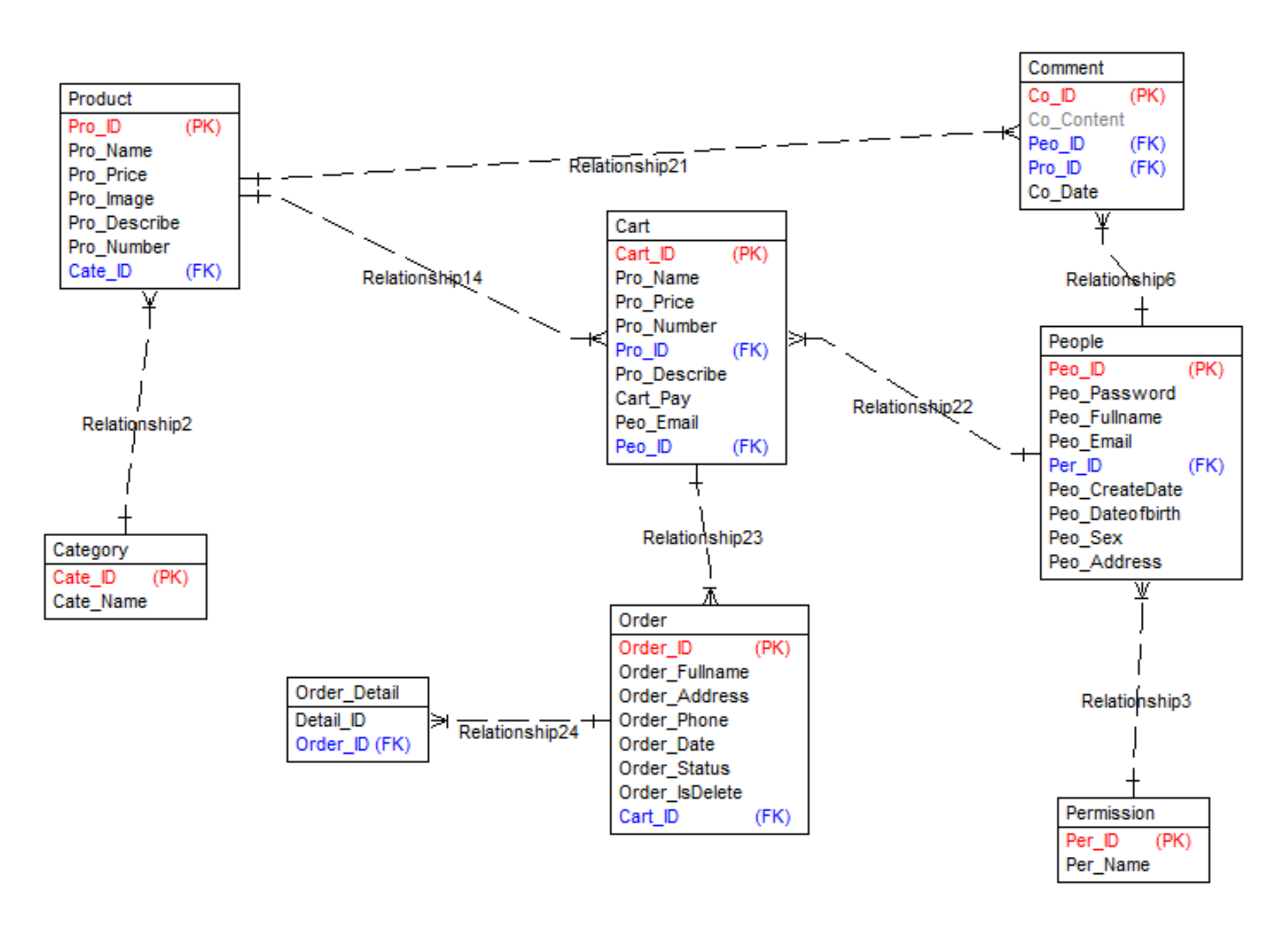
Axios là gì? Axios là một thư viện HTTP Client dựa trên Promise. Cơ bản thì nó cung cấp một API cho việc xử lý XHR (XMLHttpRequests).

Các alias cho việc tạo ra các request chính với phương thức tương ứng:

* axios.get(url[, config])
* axios.delete(url[, config])
* axios.post(url[, data[, config]])
* axios.put(url[, data[, config]])

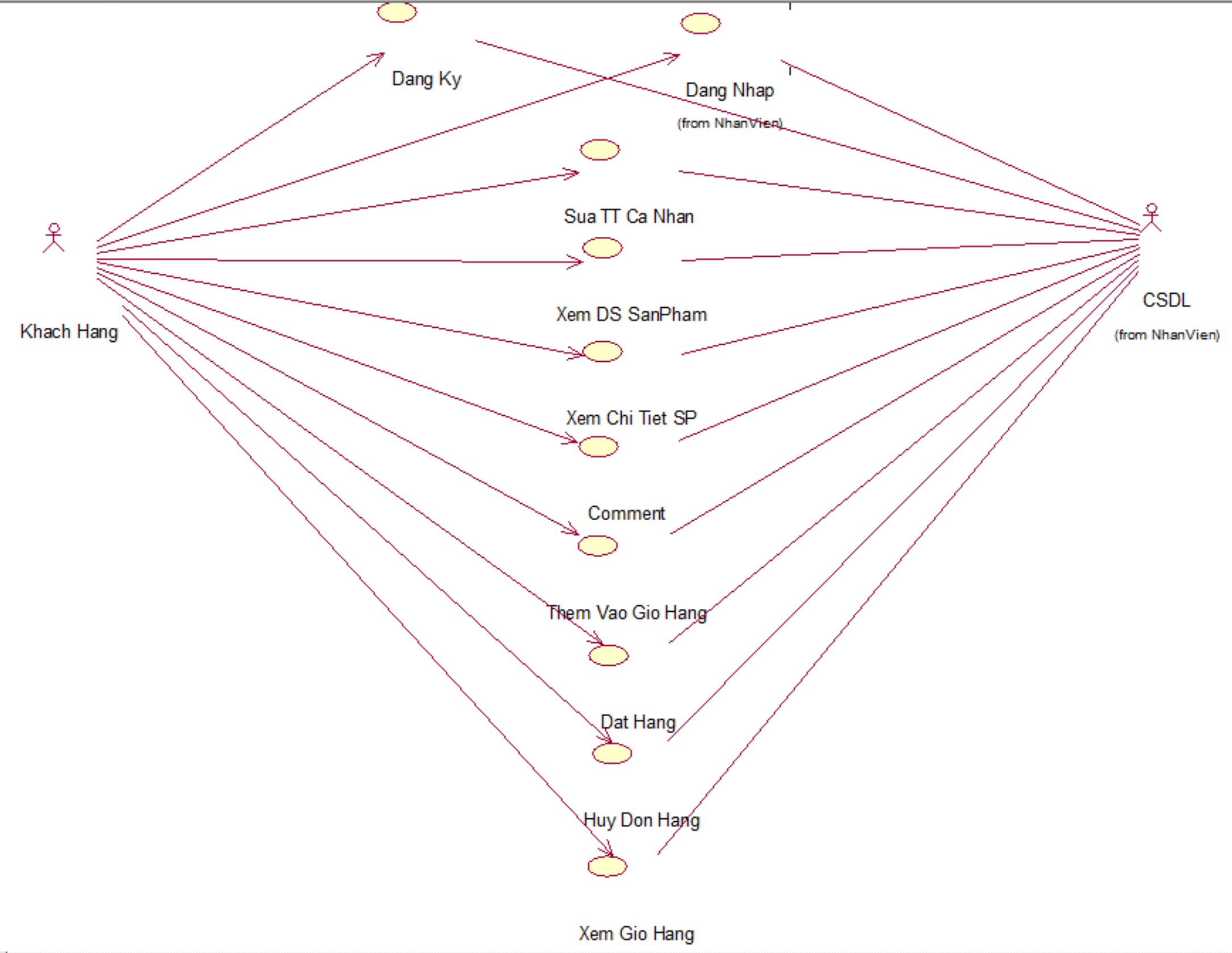
# PHÂN TÍCH THIẾT KẾ HỆ THỐNG VÀ CSDL

## Biểu đồ ERD

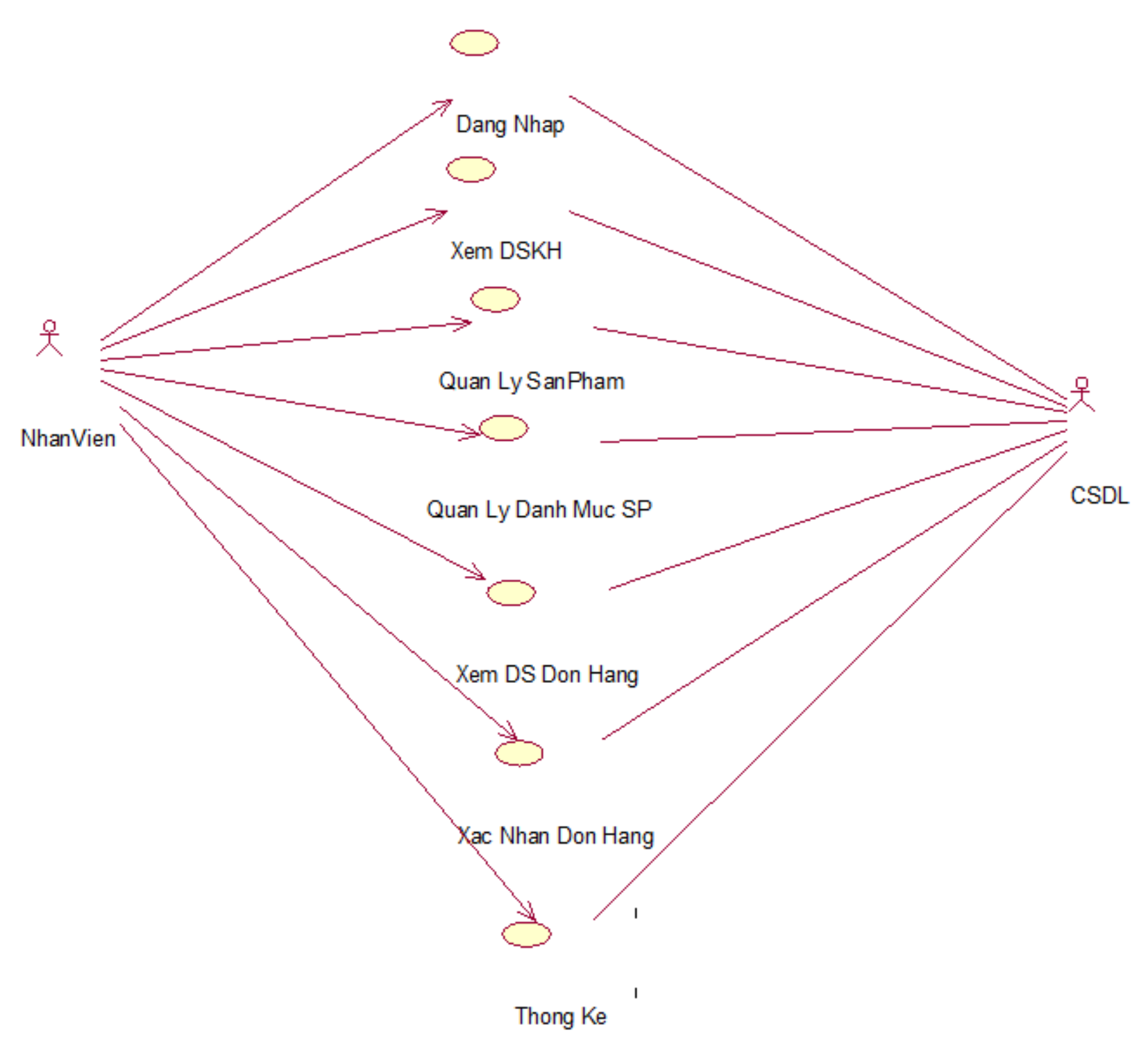


## Biểu đồ Use case

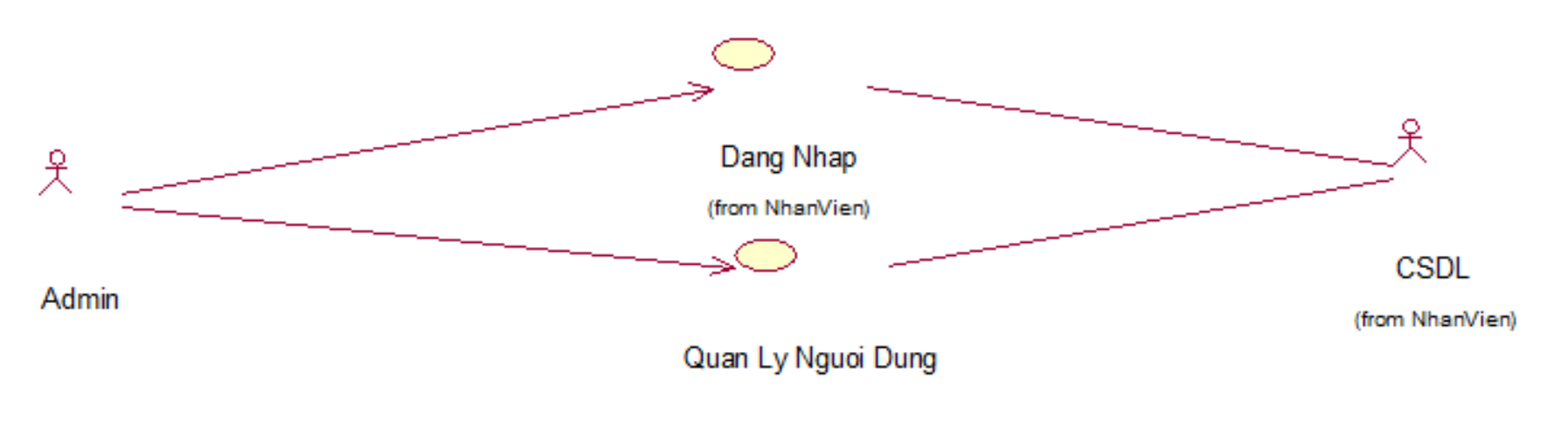
### Khách hàng



### Nhân viên



### Admin



## Biểu đồ tuần tự

### Khách hàng

#### Use case Comment



#### Use case Đăng ký



#### Use case Sửa thông tin cá nhân



#### Use case Xem DSSP



#### Use case Xem chi tiết SP



#### Use case Thêm vào giỏ hàng



#### Use case Xem giỏ hàng



#### Use case Đặt hàng



#### User case Hủy đơn hàng



### Nhân viên

#### Use case Đăng nhập



#### Use case Quản lý danh mục sản phẩm



#### Use case Quản lý sản phẩm



#### Use case Xem danh sách khách hàng



#### Use case Xem danh sách đơn hàng



#### Use case Xác nhận đơn hàng



#### Use case Thống kê



### Admin

#### Use case Quản lý người dùng



## Phân tích đặc tả Use case

### Khách hàng

#### Use case Đăng ký

Tên use case: Đăng kí

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào nút “Đăng kí” ở trang chủ. Hệ thống sẽ hiển thị form Đăng kí lên màn hình.
* Khách hàng nhập đầy đủ thông tin theo hướng dẫn, sau đó kích vào nút “Tạo tài khoản”. Hệ thống sẽ kiểm tra và lưu thông tin khách hàng vào bảng TaiKhoan.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.
* Tại bước 2 trong luồng cơ bản, nếu khách không nhập đầy đủ thông tin, hệ thống thông báo thiếu thông tin cần nhập và yêu cầu khách hàng nhập lại
* Tại bước 2 trong luồng cơ bản, tên tài khoản trùng với tên tài khoản đã tồn tại, hệ thống sẽ đưa ra thông báo “ Tên tài khoản đã tồn tại” và yêu cầu khách hàng nhập lại.

Điểm mở rộng: Không có

Tiền điều kiện: Không có

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Comment

Tên use case: Comment

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào ô comment và nhập thông tin comment và kích vào nút “Đăng”. Hệ thống sẽ thêm thông tin vào bảng COMMENT.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Sửa thông tin cá nhân

Tên use case: Sửa thông tin cá nhân

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào nút “Sửa thông tin” ở trang chủ. Hệ thống sẽ hiển thị form sửa thông tin cá nhân lên màn hình.
* Khách hàng nhập đầy đủ thông tin theo hướng dẫn, sau đó kích vào nút “Lưu”. Hệ thống sẽ kiểm tra và lưu thông tin khách hàng vào bảng PEOPLE.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.
* Tại bước 2 trong luồng cơ bản, nếu khách không nhập đầy đủ thông tin, hệ thống thông báo thiếu thông tin cần nhập và yêu cầu khách hàng nhập lại.
* Tại bước 2 trong luồng cơ bản, tên tài khoản trùng với tên tài khoản đã tồn tại, hệ thống sẽ đưa ra thông báo “ Tên tài khoản đã tồn tại” và yêu cầu khách hàng nhập lại.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Xem DSSP

Tên use case: Xem DSSP

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào danh mục sản phẩm trên thanh menu. Hệ thống sẽ lấy danh sách sản phẩm theo danh mục lên màn hình.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Không có

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Xem chi tiết SP

Tên use case: Xem chi tiết SP

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào sản phẩm bất kỳ. Hệ thống sẽ lấy thông tin chi tiết sản phẩm lên màn hình.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Không có

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Thêm vào giỏ hàng

Tên use case: Thêm vào giỏ hàng

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào nút “Thêm vào giỏ hàng” trong phần chi tiết sản phẩm. Hệ thống sẽ lấy thông tin sản phẩm đưa vào bảng CART.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Xem giỏ hàng

Tên use case: Xem giỏ hàng

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào icon cart trên thanh menu. Hệ thống sẽ lấy và hiển thị danh sách sản phẩm trong giỏ hàng lên màn hình.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Đặt hàng

Tên use case: Đặt hàng

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào nút “Đặt hàng” trên 1 sản phẩm trong giỏ hàng, hệ thống sẽ hiển thị 1 form nhập thông tin nhận hàng.
* Khách hàng nhập thông tin nhận hàng vào form và kích nút “Đặt”.
* Hệ thống sẽ lấy thông tin và đưa vào bảng ORDER.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.
* Tại bước 2 luồng cơ bản, khi khách hàng không nhập đầy đủ thông tin thì hệ thống sẽ báo lỗi và nhập lại.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Hủy đơn hàng

Tên use case: Hủy đơn hàng

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào nút “Hủy” trên 1 sản phẩm trong giỏ hàng. Hệ thống sẽ lấy sửa thông tin bảng ORDER.

Use case kết thúc

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

### Nhân viên

#### Use case Đăng nhập

Tên use case: Đăng nhập

Luồng cơ bản:

* Use case bắt đầu khi khách hàng kích vào nút “Đăng nhập” trên trang chủ. Hệ thống sẽ hiển thị form đăng nhập lên màn hình.
* Khách hàng nhập email và mật khẩu sau đó kích nút “Đăng nhập”. Hệ thống sẽ kiểm tra và đưa khách hàng đến trang chủ. Use case kết thúc.

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.
* Tại bước 2 luồng cơ bản, khi khách hàng nhập sai thông tin tài khoản, hệ thống sẽ thông báo lỗi và nhập lại.

Điểm mở rộng: Không có

Tiền điều kiện: Không có

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Quản lý danh mục sản phẩm

Tên use case: Quản lý danh mục sản phẩm

Luồng cơ bản:

1. Thêm danh mục sản phẩm

* Use case bắt đầu khi nhân viên kích vào nút “Thêm danh mục”. Hệ thống sẽ hiển thị form nhập thông tin danh mục lên màn hình.
* Nhân viên nhập thông tin sau đó kích nút “Thêm”. Hệ thống sẽ đưa thông tin vào bảng CATEGORY. Use case kết thúc.

1. Sửa danh mục sản phẩm

* Use case bắt đầu khi nhân viên chọn danh mục và kích vào nút “Sửa”. Hệ thống sẽ hiển thị form sửa thông tin danh mục lên màn hình.
* Nhân viên nhập thông tin cần sửa sau đó kích nút “Sửa”. Hệ thống sẽ cập nhật thông tin vào bảng CATEGORY. Use case kết thúc.

1. Xóa danh mục sản phẩm

* Use case bắt đầu khi nhân viên chọn danh mục và kích vào nút “Xóa”. Hệ thống sẽ hiển thị thông báo xác nhận xóa lên màn hình.
* Nhân viên kích nút “Xóa”. Hệ thống sẽ xóa thông tin trong bảng CATEGORY. Use case kết thúc.

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Quản lý sản phẩm

Tên use case: Quản lý sản phẩm

Luồng cơ bản:

1. Thêm sản phẩm

* Use case bắt đầu khi nhân viên kích vào nút “Thêm sản phẩm”. Hệ thống sẽ hiển thị form nhập thông tin sản phẩm lên màn hình.
* Nhân viên nhập thông tin sau đó kích nút “Thêm”. Hệ thống sẽ đưa thông tin vào bảng PRODUCT. Use case kết thúc.

1. Sửa sản phẩm

* Use case bắt đầu khi nhân viên chọn sản phẩm và kích vào nút “Sửa”. Hệ thống sẽ hiển thị form sửa thông tin sản phẩm lên màn hình.
* Nhân viên nhập thông tin cần sửa sau đó kích nút “Sửa”. Hệ thống sẽ cập nhật thông tin vào bảng PRODUCT. Use case kết thúc.

1. Xóa sản phẩm

* Use case bắt đầu khi nhân viên chọn sản phẩm và kích vào nút “Xóa”. Hệ thống sẽ hiển thị thông báo xác nhận xóa lên màn hình.
* Nhân viên kích nút “Xóa”. Hệ thống sẽ xóa thông tin trong bảng PRODUCT. Use case kết thúc.

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Xem danh sách khách hàng

Tên use case: Xem danh sách khách hàng

Luồng cơ bản:

* Use case bắt đầu khi nhân viên kích vào nút “Khách hàng” trên thanh menu. Hệ thống sẽ lấy danh sách khách hàng và hiển thị lên màn hình.

Use case kết thúc.

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Xem danh sách đơn hàng

Tên use case: Xem danh sách đơn hàng

Luồng cơ bản:

* Use case bắt đầu khi nhân viên kích vào nút “Đơn hàng” trên thanh menu. Hệ thống sẽ lấy danh sách đơn hàng và hiển thị lên màn hình.

Use case kết thúc.

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Xác nhận đơn hàng

Tên use case: Xác nhận đơn hàng

Luồng cơ bản:

* Use case bắt đầu khi nhân viên kích vào nút “Đơn hàng” trên thanh menu. Hệ thống sẽ lấy danh sách đơn hàng và hiển thị lên màn hình.
* Nhân viên kích vào nút “Xác nhận” đơn hàng. Hệ thống sẽ cập nhật thông tin vào bảng ORDER.

Use case kết thúc.

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

#### Use case Thống kê

Tên use case: Thống kê

Luồng cơ bản:

* Use case bắt đầu khi nhân viên chọn tháng và kích vào nút “Thống kê” trên thanh menu. Hệ thống sẽ lấy danh sách sản phẩm đã bán, doanh thu đạt được và hiển thị lên màn hình.

Use case kết thúc.

Luồng rẽ nhánh:

* Tại bất kì bước nào của quá trình thực hiện, nếu không kết nối được với cơ sở dữ liệu thì hiển thị thông báo lỗi và kết thúc use case.

Điểm mở rộng: Không có

Tiền điều kiện: Đăng nhập

Hậu điều kiện: Không có

Yêu cầu đặc biệt: Không có

### Admin

## Thiết kế CSDL

### Bảng Permission

* Per\_ID(PK, uniqueidentifier, not null): mã quyền, sinh tự động.
* Per\_Name(nvarchar(50), not null): tên quyền.

### Bảng People

* Peo\_ID(PK, uniqueidentifier, not null): Mã người dùng, sinh tự động.
* Peo\_Fullname(nvarchar(50), not null): họ tên người dùng.
* Peo\_Email(varchar(30), not null, unique): địa chỉ Email.
* Peo\_Password(varchar(30), not null): mật khẩu.
* Peo\_Address(nvarchar(50), not null): địa chỉ người dùng.
* Peo\_Dateofbirth(date, not null): ngày sinh.
* Peo\_Sex(nvarchar(20)): giới tính.
* Peo\_CreateDate(date, not null): ngày đăng ký.
* Per\_ID(FK, uniqueidentifier, not null): mã quyền.

### Bảng Category

* Cate\_ID(PK, uniqueidentifier, not null): mã danh mục, sinh tự động.
* Cate\_Name(nvarchar(50), not null): tên danh mục.
* Cate\_Describe(nvarchar(max), not null): mô tả.
* Cate\_Count(int not null): số lượng sản phẩm của danh mục.

### Bảng Product

* Pro\_ID(PK, uniqueidentifier, not null): mã sản phẩm, sinh tự động.
* Pro\_Name(nvarchar(50), not null): tên sản phẩm.
* Pro\_Price(decimal(18,4), not null): giá sản phẩm.
* Pro\_Image(varchar(20), not null): hình ảnh.
* Pro\_Describe(nvarchar(max), not null): mô tả.
* Pro\_Number(int, not null): số lượng còn.
* Cate\_ID(FK, uniqueidentifier, not null): mã danh mục.

### Bảng Comment

* Co\_ID(PK, uniqueidentifier, not null): mã comment, sinh tự động.
* Co\_Content(nvarchar(max), not null): nội dung comment.
* Co\_Date(date, not null): ngày comment.
* Peo\_ID(FK, uniqueidentifier, not null): mã người dùng.
* Pro\_ID(FK, uniqueidentifier, not null): mã sản phẩm.

### Bảng Cart

* Cart\_ID(PK, uniqueidentifier, not null): mã giỏ hàng, sinh tự động.
* Pro\_Name(nvarchar(50), not null): tên sản phẩm.
* Pro\_Price(decimal(18,4), not null): giá sản phẩm.
* Pro\_Describe(nvarchar(max), not null): mô tả.
* Pro\_Number(int, not null): số lượng.
* Cart\_Pay(decimal(18,4), not null): giá tiền.
* Peo\_ID(FK, uniqueidentifier, not null): mã người dùng.
* Pro\_ID(FK, uniqueidentifier, not null): mã sản phẩm.

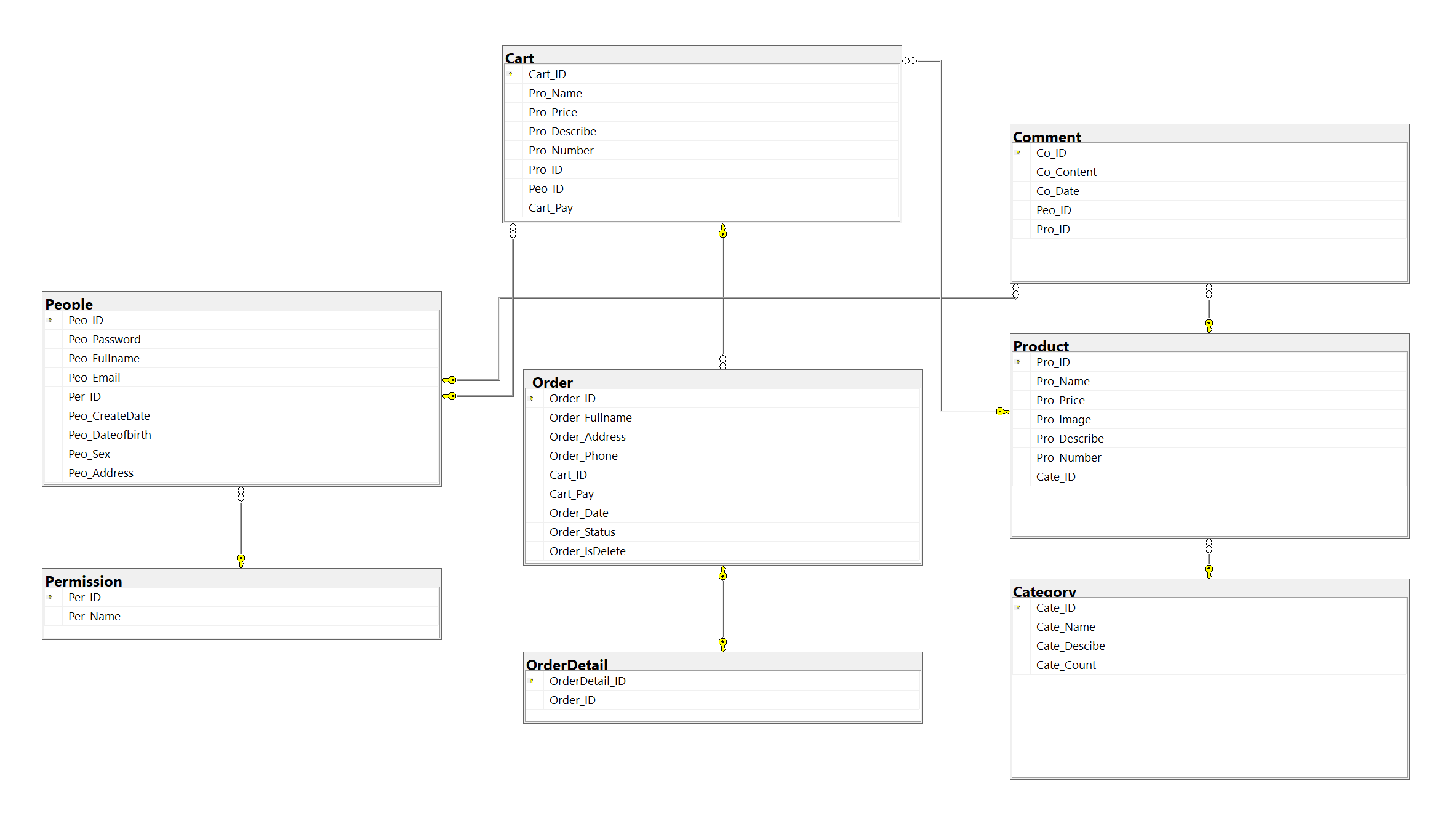
### Bảng Order

* Order\_ID(PK, uniqueidentifier, not null): mã đơn đặt hàng, sinh tự động.
* Order\_Fullname(nvarchar(50), not null): tên người nhận hàng.
* Order\_Address(nvarchar(50), not null): địa chỉ nhận hàng.
* Order\_Phone(decimal(18,0), not null): số điện thoại nhận hàng.
* Order\_Date(date, not null): ngày đặt hàng.
* Order\_Status(nvarchar(50), not null): trạng thái đơn hàng.
* Order\_IsDelete(varchar(20), not null) true or false.
* Cart\_ID(FK, uniqueidentifier, not null): mã giỏ hàng.

### Bảng OrderDetail

* OrderDetail\_ID(PK, uniqueidentifier, not null): mã chi tiết đơn hàng, sinh tự động.
* Order\_ID(FK, uniqueidentifier, not null, unique): mã đơn đặt hàng.

### Diagram



# PHÁT TRIỂN ỨNG DỤNG

# TỔNG KẾT

# TÀI LIỆU THAM KHẢO