TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐOÀN THỊ DIỄM QUỲNH - 52100758**

**DƯƠNG THÁI TÍN - 52100761**

**XÂY DỰNG ỨNG DỤNG THƯƠNG MẠI ĐIỆN TỬ FULL-STACK TRÊN NỀN TẢNG WEB APP**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN**

**KỸ THUẬT PHẦN MỀM**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐOÀN THỊ DIỄM QUỲNH – 52100758**

**DƯƠNG THÁI TÍN - 52100761**

**XÂY DỰNG ÚNG DỤNG THƯƠNG MẠI ĐIỆN TỬ FULL-STACK TRÊN NỀN TẢNG WEB APP**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN**

**KỸ THUẬT PHẦN MỀM**

Người hướng dẫn

**Thầy DƯƠNG HỮU PHƯỚC**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến thầy Dương Hữu Phước, người đã tận tình hướng dẫn, hỗ trợ và đóng góp những ý kiến quý báu trong suốt quá trình nghiên cứu và phát triển dự án này. Nhờ có sự hướng dẫn của thầy, chúng em đã có cơ hội tiếp cận với các công nghệ hiện đại và áp dụng vào thực tiễn một cách hiệu quả.

Chúng em cũng xin gửi lời cảm ơn đến quý thầy cô trong khoa Công nghệ Thông tin đã cung cấp nền tảng kiến thức vững chắc, giúp chúng em có đủ năng lực để thực hiện đề tài này. Bên cạnh đó, chúng em xin cảm ơn các bạn cùng lớp, bạn bè đã hỗ trợ, động viên trong suốt quá trình thực hiện.

Cuối cùng, chúng em xin gửi lời cảm ơn đến gia đình, những người luôn đồng hành, động viên và tạo điều kiện thuận lợi để chúng em có thể hoàn thành dự án này một cách tốt nhất.

*TP. Hồ Chí Minh, ngày 14 tháng 2 năm 2025*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Đoàn Thị Diễm Quỳnh*

*Dương Thái Tín*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của Thầy Dương Hữu Phước. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 14 tháng 2 năm 2025*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Đoàn Thị Diễm Quỳnh*

*Dương Thái Tín*

**XÂY DỰNG ỨNG DỤNG THƯƠNG MẠI ĐIỆN TỬ FULL-STACK TRÊN NỀN TẢNG WEB APP**

**TÓM TẮT**

CHƯƠNG 1: Mở đầu và tổng quan đề tài

CHƯƠNG 2: Cơ sở lý thuyết

CHƯƠNG 3: Giao diện hệ thống và luồng xử lý chức năng

CHƯƠNG 4: Kết luận

**BUILDING A FULL-STACK E-COMMERCE APPLICATION ON A WEB PLATFORM**

**ABSTRACT**

CHAPTER 1: Introduction and Overview of the Topic

CHAPTER 2: Theoretical Basis

CHAPTER 3: System Interface and Functional Flow

CHAPTER 4: Conclusion

**MỤC LỤC**

[DANH MỤC HÌNH VẼ vii](#_Toc190539520)

[DANH MỤC BẢNG BIỂU viii](#_Toc190539521)

[DANH MỤC CÁC CHỮ VIẾT TẮT ix](#_Toc190539522)

[CHƯƠNG 1. MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI 1](#_Toc190539523)

[1.1 Lý do chọn đề tài 1](#_Toc190539524)

[1.2 Mục tiêu thực hiện đề tài 1](#_Toc190539525)

[1.2.1 Phạm vi 1](#_Toc190539526)

[1.2.2 Công nghệ sử dụng 1](#_Toc190539527)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 3](#_Toc190539528)

[2.1 Kiến trúc phần mềm 3](#_Toc190539529)

[2.1.1 Mô hình Client-Server 3](#_Toc190539530)

[2.1.2 Mô hình Model-View-Controller (MVC) 4](#_Toc190539531)

[2.1.3 Kiến trúc RESTful API 5](#_Toc190539532)

[2.2 Công nghệ phát triển 6](#_Toc190539533)

[2.2.1 Front-end 6](#_Toc190539534)

[2.2.2 Back-end 9](#_Toc190539535)

[2.3 Công nghệ phát triển 11](#_Toc190539536)

[2.3.1 Mô hình ERD (Entity-Relationship Diagram) 11](#_Toc190539537)

[2.3.2 Lược đồ quan hệ (Relational Schema) 12](#_Toc190539538)

[2.4 Phân thích và thiết kế hệ thống 14](#_Toc190539539)

[2.4.1 Use Case Diagram 14](#_Toc190539540)

[2.4.2 Đặc tả Use Case 16](#_Toc190539541)

[2.4.3 Lược đồ ERD 21](#_Toc190539542)

[2.4.4 Cơ sở dữ liệu 23](#_Toc190539543)

[2.4.5 Sequence Diagram 27](#_Toc190539544)

[2.5 Công nghệ bảo mật 33](#_Toc190539545)

[2.5.1 Xác thực và phân quyền người dùng 34](#_Toc190539546)

[2.5.2 Bảo mật dữ liệu 35](#_Toc190539547)

[2.5.3 Bảo mật giao tiêp dữ liệu 35](#_Toc190539548)

[CHƯƠNG 3. GIAO DIỆN HỆ THỐNG VÀ LUỒNG XỬ LÝ CHỨC NĂNG 38](#_Toc190539549)

[3.1 Trang chủ 38](#_Toc190539550)

[3.2 Đăng nhập 38](#_Toc190539551)

[3.3 Đăng kí 38](#_Toc190539552)

[CHƯƠNG 4. KẾT LUẬN 40](#_Toc190539553)

[4.1 Kết luận 40](#_Toc190539554)

[4.2 Hướng phát triển 40](#_Toc190539555)

[TÀI LIỆU THAM KHẢO 42](#_Toc190539556)

# DANH MỤC HÌNH VẼ

[Hình 2.1: Sơ đồ use-case của khách hàng 14](#_Toc190539502)

[Hình 2.2: Sơ đồ use-case của khách hàng thành viên 15](#_Toc190539503)

[Hình 2.3: Sơ đồ use-case của admin 16](#_Toc190539504)

[Hình 2.4: ERD mô tả lược đồ quan hệ 23](#_Toc190539505)

[Hình 2.5: Biểu đồ tuần tự đăng nhập 28](#_Toc190539506)

[Hình 2.6: Biểu đồ tuần tự đăng kí 28](#_Toc190539507)

[Hình 2.7: Biểu đồ tuần tự tìm kiếm sản phẩm 29](#_Toc190539508)

[Hình 2.8: Biểu đồ tuần tự đặt hàng 29](#_Toc190539509)

[Hình 2.9: Biểu đồ tuần tự xem thông tin sản phẩm 30](#_Toc190539510)

[Hình 2.10: Biểu đồ tuần tự bình luận 30](#_Toc190539511)

[Hình 2.11: Biểu đồ tuần tự hiển thị danh sách sản phẩm 31](#_Toc190539512)

[Hình 2.12: Biểu đồ tuần tự thêm sản phẩm 31](#_Toc190539513)

[Hình 2.13: Biểu đồ tuần tự xóa sản phẩm 32](#_Toc190539514)

[Hình 2.14: Biểu đồ tuần tự xem sản phẩm 32](#_Toc190539515)

[Hình 2.14: Biểu đồ tuần tự xoá người dùng 33](#_Toc190539516)

[Hình 2.14: Biểu đồ tuần tự xoá đơn hàng 33](#_Toc190539517)

[Hình 3.1: Giao diện trang đăng nhập 38](#_Toc190539518)

[Hình 3.2: Giao diện trang đăng kí 39](#_Toc190539519)

# DANH MỤC BẢNG BIỂU

[Bảng 2.1: Bảng đặc tả use-case xem thông tin điện thoại 17](#_Toc190533247)

[Bảng 2.2: Bảng đặc tả use-case tìm kiếm điện thoại 18](#_Toc190533248)

[Bảng 2.3: Bảng đặc tả use-case đánh giá điện thoại 18](#_Toc190533249)

[Bảng 2.4: Bảng đặc tả use-case đặt hàng 19](#_Toc190533250)

[Bảng 2.5: Bảng đặc tả use-case quản lý sản phẩm 20](#_Toc190533251)

[Bảng 2.6: Bảng đặc tả use-case quản lý đơn hàng 21](#_Toc190533252)

[Bảng 2.7: Bảng Users (Khách hàng) 24](#_Toc190533253)

[Bảng 2.8: Bảng Products (Sản phẩm) 24](#_Toc190533254)

[Bảng 2.9: Bảng Orders (Đơn hàng) 25](#_Toc190533255)

[Bảng 2.10: Bảng Order\_Items (Chi tiết đơn hàng) 25](#_Toc190533256)

[Bảng 2.11: Bảng Cart (Giỏ hàng) 26](#_Toc190533257)

[Bảng 2.12: Bảng Reviews (Đánh giá sản phẩm) 26](#_Toc190533258)

[Bảng 2.13: Bảng Categories (Danh mục sản phẩm) 26](#_Toc190533259)

[Bảng 2.14: Bảng Payments (Thanh toán) 27](#_Toc190533260)

# DANH MỤC CÁC CHỮ VIẾT TẮT

# MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI

## Lý do chọn đề tài

Trong thời đại kỷ nguyên số, thương mại điện tử (E-commerce) đã và đang trở thành một trong những lĩnh vực phát triển mạnh mẽ nhất, đóng vai trò quan trọng trong nền kinh tế số. Với sự bùng nổ của các nền tảng mua sắm trực tuyến như Shopee, Lazada, Tiki, ...., nhu cầu của người tiêu dùng và doanh nghiệp đối với các giải pháp thương mại điện tử ngày càng tăng. Vì vậy, việc nghiên cứu và phát triển một ứng dụng thương mại điện tử hoàn chỉnh trên nền tảng web full-stack không chỉ giúp nâng cao trải nghiệm người dùng mà còn góp phần thúc đẩy sự chuyển đổi số trong kinh doanh..

## Mục tiêu thực hiện đề tài

Xây dựng một website thương mại điện tử hoàn chỉnh với giao diện trực quan, thân thiện với người dùng.

Cung cấp đầy đủ các chức năng quan trọng của một hệ thống thương mại điện tử như quản lý sản phẩm, giỏ hàng, thanh toán và quản lý đơn hàng.

Ứng dụng các công nghệ hiện đại như Web API, ReactJS, PHP Laravel để đảm bảo hiệu suất, bảo mật và trải nghiệm người dùng tối ưu.

Tích hợp các phương thức thanh toán trực tuyến an toàn và tiện lợi như VNPAY, MoMo.

Phát triển hệ thống phân quyền linh hoạt, đảm bảo mỗi người dùng có trải nghiệm phù hợp với vai trò của mình.

### Phạm vi

Ứng dụng hoạt động trên môi trường web.

Hỗ trợ nhiều loại thiết bị: PC, laptop, tablet, mobile.

Đối tượng sử dụng: Người mua hàng, quản trị viên quản lý sản phẩm và đơn hàng.

### Công nghệ sử dụng

**Front-end:** ReactJS, Redux để quản lý trạng thái ứng dụng.

**Back-end:** PHP Laravel với kiến trúc MVC.

**Cơ sở dữ liệu**: MySQL.

**Thanh toán trực tuyến:** VNPAY, MoMo.

**Xác thực và bảo mật:** JWT Authentication, mã hóa mật khẩu bằng bcrypt.

# CƠ SỞ LÝ THUYẾT

## Kiến trúc phần mềm

### Mô hình Client-Server

Mô hình Client-Server là nền tảng chính trong hệ thống. Đây là mô hình mạng phân tán trong đó Client (máy khách) gửi yêu cầu đến Server (máy chủ) để truy xuất hoặc xử lý dữ liệu. Server tiếp nhận yêu cầu, xử lý nghiệp vụ và trả về kết quả. Mô hình này giúp hệ thống hoạt động linh hoạt, dễ mở rộng và tối ưu hiệu suất.

Kiến trúc ba lớp (Three-tier Architecture) được áp dụng để tách biệt giữa giao diện người dùng (Frontend), xử lý nghiệp vụ (Backend) và lưu trữ dữ liệu (Database). Lớp Frontend chịu trách nhiệm hiển thị và xử lý tương tác với người dùng, lớp Backend xử lý các quy trình logic, còn lớp Database lưu trữ và quản lý dữ liệu. Việc tách biệt này giúp tăng tính bảo mật, dễ dàng bảo trì và nâng cấp hệ thống.

RESTful API được sử dụng để giao tiếp giữa Frontend và Backend theo mô hình Client-Server. API tuân theo các phương thức HTTP như GET, POST, PUT, DELETE để trao đổi dữ liệu một cách hiệu quả. Điều này giúp hệ thống linh hoạt, có thể mở rộng và tích hợp với nhiều nền tảng khác.

Về bảo mật, hệ thống sử dụng JWT (JSON Web Token) để xác thực và phân quyền người dùng. Khi đăng nhập, Server cấp một token để Client sử dụng trong các yêu cầu tiếp theo, đảm bảo tính bảo mật. Ngoài ra, mật khẩu người dùng được băm bằng bcrypt để bảo vệ dữ liệu, sử dụng HTTPS để mã hóa thông tin truyền tải và hạn chế SQL Injection bằng cách sử dụng ORM (Eloquent trong Laravel).

Cơ sở dữ liệu quan hệ (Relational Database) với hệ quản trị MySQL được áp dụng để tổ chức và quản lý dữ liệu theo mô hình bảng. Các bảng như users, products, cart, … giúp lưu trữ dữ liệu có cấu trúc rõ ràng, hỗ trợ truy vấn nhanh chóng và đảm bảo tính nhất quán dữ liệu. Laravel sử dụng Eloquent ORM giúp thao tác với dữ liệu dễ dàng và an toàn hơn so với SQL thuần.

Các lý thuyết trên là nền tảng quan trọng giúp hệ thống thương mại điện tử hoạt động hiệu quả, đảm bảo tính bảo mật, dễ bảo trì và có thể mở rộng trong tương lai.

### Mô hình Model-View-Controller (MVC)

Mô hình MVC (Model - View - Controller) là một kiến trúc phần mềm phổ biến trong phát triển ứng dụng web, giúp tổ chức mã nguồn một cách rõ ràng, dễ bảo trì và mở rộng. MVC chia ứng dụng thành ba phần chính:

* **Model:** Chịu trách nhiệm quản lý dữ liệu, thực hiện các thao tác với cơ sở dữ liệu như truy vấn, thêm, sửa, xóa dữ liệu. Trong Laravel, Model sử dụng Eloquent ORM để tương tác với cơ sở dữ liệu một cách dễ dàng và bảo mật.
* **View**: Quản lý giao diện người dùng, hiển thị dữ liệu từ Model và nhận tương tác từ người dùng. Trong ReactJS, View được xây dựng bằng các component giúp hiển thị thông tin một cách động và tối ưu trải nghiệm người dùng.
* **Controller:** Xử lý logic nghiệp vụ, nhận yêu cầu từ người dùng qua HTTP request, gọi Model để lấy dữ liệu và trả về View để hiển thị. Trong Laravel, Controller giúp điều hướng request, xử lý logic và gọi API khi cần thiết.

***Lợi ích của mô hình MVC:***

* Tách biệt rõ ràng giữa các thành phần, giúp mã nguồn dễ bảo trì, nâng cấp và mở rộng.
* Cải thiện khả năng tái sử dụng mã nguồn, giúp tiết kiệm thời gian phát triển.
* Hỗ trợ làm việc nhóm tốt hơn, khi các lập trình viên có thể làm việc độc lập trên từng phần của hệ thống mà không ảnh hưởng đến nhau.
* Dễ kiểm thử (Unit Testing) vì các thành phần được tách biệt, giúp phát hiện và sửa lỗi nhanh chóng.

Trong dự án thương mại điện tử, mô hình MVC giúp quản lý dữ liệu người dùng, sản phẩm, đơn hàng một cách rõ ràng. Laravel đảm nhận vai trò backend với Model xử lý dữ liệu, Controller xử lý logic và View hiển thị thông tin, còn ReactJS đảm nhận phần View trên giao diện người dùng, giúp tối ưu trải nghiệm và tăng hiệu suất hệ thống.

### Kiến trúc RESTful API

RESTful API là một kiểu kiến trúc thiết kế API dựa trên nguyên tắc REST (Representational State Transfer), được sử dụng phổ biến trong các hệ thống web để giao tiếp giữa client và server. RESTful API tuân thủ các nguyên tắc sau:

* **Client - Server**: Kiến trúc REST tách biệt client (giao diện người dùng) và server (xử lý dữ liệu), giúp hệ thống linh hoạt và dễ mở rộng.
* **Stateless** (Không trạng thái): Mỗi request từ client đến server đều độc lập và không lưu trạng thái. Điều này giúp server không cần nhớ thông tin của request trước đó, giúp tăng hiệu suất và mở rộng hệ thống dễ dàng.
* **Cacheable** (Có thể lưu trữ cache): REST hỗ trợ caching để giảm tải cho server, giúp cải thiện hiệu suất bằng cách lưu trữ dữ liệu response tạm thời.
* **Uniform Interface** (Giao diện đồng nhất): REST API sử dụng các phương thức HTTP chuẩn:

*GET*: Lấy dữ liệu từ server.

*POST*: Gửi dữ liệu mới lên server.

*PUT*: Cập nhật dữ liệu hiện có.

*DELETE*: Xóa dữ liệu trên server.

* **Layered System** (Hệ thống phân lớp): Kiến trúc REST có thể có nhiều tầng (layers), giúp tăng tính bảo mật và khả năng mở rộng.
* **Code on Demand** (Tùy chọn tải mã lệnh): Server có thể gửi mã thực thi (như JavaScript) đến client để mở rộng chức năng, nhưng đây là tùy chọn.

***Ưu điểm của RESTful API:***

* Dễ hiểu và sử dụng do tuân theo các quy tắc HTTP chuẩn.
* Tương thích với nhiều nền tảng khác nhau như web, mobile, desktop.
* Tăng hiệu suất hệ thống nhờ caching và kiến trúc phân lớp.
* Dễ mở rộng và bảo trì do tách biệt giữa client và server.
* RESTful API trong dự án thương mại điện tử

Trong hệ thống thương mại điện tử, RESTful API giúp xử lý các chức năng như:

* Lấy danh sách sản phẩm (GET /api/products).
* Thêm sản phẩm vào giỏ hàng (POST /api/cart/add).
* Cập nhật số lượng sản phẩm trong giỏ hàng (PUT /api/cart/update).
* Xóa sản phẩm khỏi giỏ hàng (DELETE /api/cart/remove).
* Xác nhận và tạo đơn hàng (POST /api/orders).

RESTful API giúp frontend (ReactJS) và backend (Laravel) giao tiếp hiệu quả, đảm bảo dữ liệu đồng bộ và nâng cao trải nghiệm người dùng.

## Công nghệ phát triển

### Front-end

#### ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở do Facebook phát triển, dùng để xây dựng giao diện người dùng (UI) cho ứng dụng web và mobile. React hoạt động theo kiến trúc component-based, trong đó UI được chia thành các component độc lập, có thể tái sử dụng, giúp tối ưu hóa quá trình phát triển và bảo trì ứng dụng. React sử dụng Virtual DOM để cải thiện hiệu suất, giúp cập nhật giao diện nhanh chóng bằng cách so sánh sự thay đổi trong DOM ảo trước khi cập nhật vào DOM thật. Cơ chế One-Way Data Binding giúp dữ liệu chỉ chảy theo một hướng từ state đến UI, làm tăng tính kiểm soát và dự đoán của ứng dụng. JSX, một cú pháp mở rộng của JavaScript, giúp lập trình viên viết mã HTML ngay trong JavaScript, làm cho việc phát triển UI trở nên dễ dàng hơn. React hỗ trợ state và props để quản lý dữ liệu động trong component, trong đó state là dữ liệu riêng của component, còn props giúp truyền dữ liệu giữa các component. React Hooks, từ phiên bản 16.8, cho phép sử dụng state và các tính năng React trong functional components mà không cần dùng class components. Một số hooks phổ biến như useState, useEffect, useContext giúp quản lý state, hiệu ứng, và dữ liệu dùng chung trong ứng dụng. React Router hỗ trợ điều hướng trong ứng dụng Single Page Application (SPA) bằng cách thay đổi URL mà không cần tải lại trang. Redux là một thư viện giúp quản lý state toàn cục, giúp lưu trữ và chia sẻ trạng thái giữa các component một cách hiệu quả. React được sử dụng rộng rãi trong các ứng dụng thương mại điện tử để xây dựng giao diện người dùng mượt mà, quản lý sản phẩm, giỏ hàng, thanh toán và đơn hàng thông qua tích hợp với backend qua RESTful API.

#### Redux

Redux là một thư viện quản lý trạng thái ứng dụng phổ biến trong JavaScript, thường được sử dụng với React để duy trì và quản lý dữ liệu một cách hiệu quả. Redux hoạt động theo mô hình Flux architecture, với nguyên tắc state global duy nhất giúp đảm bảo dữ liệu nhất quán trong toàn bộ ứng dụng.

Redux sử dụng Store, nơi chứa toàn bộ state của ứng dụng. Các thành phần giao diện không trực tiếp thay đổi state mà sử dụng Action, là các đối tượng JavaScript mô tả ý định thay đổi state. Các action này được xử lý bởi Reducer, là các hàm thuần túy nhận state hiện tại và action, sau đó trả về state mới. Điều này giúp đảm bảo state luôn nhất quán và dễ theo dõi.

Cơ chế Dispatch cho phép gửi action đến store để kích hoạt reducer và cập nhật state. Khi state thay đổi, React components sẽ tự động cập nhật theo thông qua useSelector và useDispatch, giúp tối ưu hóa hiệu suất của ứng dụng.

Redux hoạt động theo nguyên tắc Immutable State, nghĩa là thay vì thay đổi trực tiếp state, nó tạo ra một bản sao mới với dữ liệu cập nhật, giúp giảm thiểu lỗi trong ứng dụng và cải thiện khả năng debug.

Redux cũng hỗ trợ Middleware như Redux Thunk hoặc Redux Saga, giúp xử lý các tác vụ bất đồng bộ như gọi API hoặc thực hiện logic phức tạp trước khi cập nhật state.

Trong các ứng dụng thương mại điện tử, Redux thường được sử dụng để quản lý dữ liệu người dùng, giỏ hàng, đơn hàng và sản phẩm, đảm bảo trải nghiệm mượt mà và nhất quán giữa các trang.

#### Axios

Axios là một thư viện HTTP Client dựa trên Promise, được sử dụng để gửi yêu cầu HTTP từ trình duyệt hoặc môi trường Node.js. Nó giúp thực hiện các request bất đồng bộ đến server một cách dễ dàng, đồng thời hỗ trợ quản lý dữ liệu trả về một cách hiệu quả.

Axios hỗ trợ nhiều phương thức HTTP như GET, POST, PUT, DELETE, giúp gửi và nhận dữ liệu từ API một cách linh hoạt. Một trong những ưu điểm lớn của Axios so với fetch API là nó tự động chuyển đổi dữ liệu JSON, giúp xử lý phản hồi từ server dễ dàng hơn.

Axios cung cấp Interceptors, cho phép xử lý yêu cầu (request) và phản hồi (response) trước khi chúng được gửi hoặc nhận, giúp tối ưu hóa bảo mật và quản lý lỗi. Nó cũng hỗ trợ hủy request (Cancel Token), giúp giảm tải tài nguyên khi thực hiện các thao tác như tìm kiếm hoặc cập nhật dữ liệu theo thời gian thực.

Một tính năng quan trọng khác của Axios là cấu hình mặc định (Global Config), giúp định nghĩa trước các headers, token xác thực, base URL... giúp đơn giản hóa quá trình gửi request. Ngoài ra, Axios cũng hỗ trợ timeout, giúp giới hạn thời gian chờ phản hồi từ server để tránh treo ứng dụng.

Trong các ứng dụng thương mại điện tử, Axios thường được sử dụng để kết nối giữa Front-end (ReactJS) và Back-end (Laravel API), giúp lấy dữ liệu sản phẩm, giỏ hàng, đơn hàng và thực hiện xác thực người dùng.

### Back-end

#### Laravel (PHP Framework)

Laravel là một framework PHP phổ biến, được thiết kế để xây dựng các ứng dụng web một cách nhanh chóng và hiệu quả. Nó tuân theo kiến trúc MVC (Model-View-Controller), giúp tổ chức mã nguồn một cách rõ ràng, dễ bảo trì và mở rộng. Laravel cung cấp nhiều tính năng mạnh mẽ hỗ trợ lập trình viên trong việc phát triển ứng dụng, bao gồm quản lý routing, truy vấn cơ sở dữ liệu, xác thực người dùng, xử lý lỗi và gửi email.

Một trong những đặc điểm nổi bật của Laravel là Eloquent ORM, một hệ thống ORM mạnh mẽ giúp tương tác với cơ sở dữ liệu thông qua các mô hình thay vì truy vấn SQL thuần túy. Laravel cũng hỗ trợ Migration, giúp quản lý cấu trúc cơ sở dữ liệu một cách dễ dàng và đồng bộ trên nhiều môi trường làm việc.

Laravel cung cấp hệ thống Middleware giúp kiểm soát truy cập và bảo mật ứng dụng. Hệ thống Blade Template Engine giúp tạo giao diện linh hoạt, dễ sử dụng và tối ưu hiệu suất. Ngoài ra, Laravel hỗ trợ API RESTful, giúp xây dựng các ứng dụng web tương tác với client một cách hiệu quả.

Bảo mật trong Laravel được đảm bảo nhờ các tính năng như mã hóa dữ liệu, bảo vệ CSRF (Cross-Site Request Forgery), kiểm tra đầu vào và cơ chế bảo mật authentication tích hợp sẵn.

Trong các ứng dụng thương mại điện tử, Laravel thường được sử dụng để phát triển Back-end, xử lý dữ liệu giỏ hàng, quản lý sản phẩm, đơn hàng, và xác thực người dùng thông qua hệ thống API kết nối với Front-end ReactJS.

#### My SQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS - Relational Database Management System) mã nguồn mở phổ biến, được sử dụng rộng rãi để lưu trữ và quản lý dữ liệu cho các ứng dụng web. MySQL hoạt động theo mô hình Client-Server, trong đó Client gửi truy vấn SQL và Server xử lý truy vấn, sau đó trả về kết quả.

MySQL sử dụng SQL (Structured Query Language) làm ngôn ngữ chính để thao tác với dữ liệu. Các thao tác cơ bản trong MySQL bao gồm:

* Tạo cơ sở dữ liệu và bảng (CREATE DATABASE, CREATE TABLE).
* Chỉnh sửa dữ liệu (INSERT, UPDATE, DELETE).
* Truy vấn dữ liệu (SELECT, WHERE, ORDER BY, JOIN).
* Quản lý quyền truy cập (GRANT, REVOKE).

Một trong những đặc điểm quan trọng của MySQL là tính nhất quán và đảm bảo toàn vẹn dữ liệu, nhờ các ràng buộc Primary Key, Foreign Key, Unique, Not Null. MySQL cũng hỗ trợ các cơ chế ACID (Atomicity, Consistency, Isolation, Durability) để đảm bảo an toàn dữ liệu trong các giao dịch (Transactions).

Hệ thống lưu trữ dữ liệu của MySQL có nhiều Storage Engine, phổ biến nhất là:

* InnoDB: Hỗ trợ ràng buộc khóa ngoại (Foreign Key), giao dịch ACID, phù hợp với các hệ thống yêu cầu tính toàn vẹn dữ liệu cao.
* MyISAM: Hiệu suất cao, không hỗ trợ giao dịch, thường dùng trong các ứng dụng đọc dữ liệu nhiều.

MySQL có thể tích hợp với nhiều ngôn ngữ lập trình như PHP, Python, Java và hoạt động hiệu quả với các framework như Laravel. Trong một ứng dụng thương mại điện tử, MySQL được sử dụng để lưu trữ và quản lý thông tin người dùng, sản phẩm, giỏ hàng, đơn hàng và hỗ trợ truy vấn dữ liệu nhanh chóng.

#### JWT (JSON Web Token)

JWT (JSON Web Token) là một tiêu chuẩn mở được sử dụng để truyền tải thông tin một cách an toàn giữa các bên dưới dạng đối tượng JSON. JWT thường được sử dụng trong quá trình xác thực và phân quyền của các ứng dụng web và API. JWT bao gồm ba phần chính: Header, Payload và Signature. Header chứa thông tin về thuật toán mã hóa và loại token. Payload chứa dữ liệu liên quan đến người dùng hoặc hệ thống, bao gồm các claims như thông tin người dùng, thời gian hết hạn, quyền hạn. Signature được tạo bằng cách mã hóa Header và Payload với một khóa bí mật hoặc sử dụng cặp khóa công khai - khóa riêng tư để đảm bảo tính toàn vẹn của token.

JWT hoạt động theo cơ chế khi người dùng đăng nhập, hệ thống kiểm tra thông tin và nếu hợp lệ, JWT sẽ được tạo và gửi về client. Client lưu trữ token này và gửi kèm trong các yêu cầu tiếp theo đến server thông qua tiêu đề Authorization. Server nhận token và kiểm tra chữ ký, nếu hợp lệ thì tiếp tục xử lý yêu cầu. JWT có nhiều ưu điểm như bảo mật cao do sử dụng chữ ký số, không cần lưu trạng thái giúp giảm tải cho server, hiệu suất cao vì không cần truy vấn cơ sở dữ liệu nhiều lần và phù hợp với các hệ thống phân tán như microservices. Tuy nhiên, JWT cũng có một số nhược điểm như không thể thu hồi khi đã cấp phát, kích thước lớn do chứa nhiều thông tin và không thể cập nhật thông tin người dùng sau khi đã phát hành.

JWT được ứng dụng rộng rãi trong các hệ thống bảo mật và xác thực như xác thực người dùng thay thế session, phân quyền để xác định quyền truy cập của người dùng, giao tiếp giữa các microservices để xác thực các yêu cầu trong hệ thống phân tán và bảo mật API RESTful bằng cách xác thực yêu cầu từ phía client. Trong dự án thương mại điện tử này, JWT được sử dụng để xác thực và phân quyền người dùng trên hệ thống, giúp đảm bảo chỉ những người dùng hợp lệ mới có thể thực hiện các hành động như mua hàng, quản lý giỏ hàng và xử lý đơn hàng.

## Công nghệ phát triển

### Mô hình ERD (Entity-Relationship Diagram)

Mô hình ERD (Entity-Relationship Diagram) là một phương pháp trực quan để mô tả cấu trúc dữ liệu của một hệ thống. ERD giúp biểu diễn các thực thể (entities), thuộc tính (attributes) của chúng và mối quan hệ (relationships) giữa các thực thể đó. Mô hình này thường được sử dụng trong giai đoạn thiết kế cơ sở dữ liệu để đảm bảo dữ liệu được tổ chức một cách hợp lý và hiệu quả.

Trong ERD, thực thể (entity) là một đối tượng cụ thể trong hệ thống, có thể là con người, sản phẩm, đơn hàng, khách hàng, v.v. Mỗi thực thể có một tập thuộc tính mô tả các đặc điểm của nó. Thực thể có thể được phân loại thành thực thể chính (strong entity) và thực thể yếu (weak entity), trong đó thực thể yếu không thể tồn tại độc lập và phụ thuộc vào thực thể chính. Mỗi thực thể có một khóa chính (primary key) để định danh duy nhất.

Quan hệ (relationship) trong ERD thể hiện sự liên kết giữa các thực thể. Các mối quan hệ có thể là một-một (1:1), một-nhiều (1:N) hoặc nhiều-nhiều (M:N). Mỗi quan hệ có thể có thuộc tính riêng và thường được biểu diễn bằng các đường nối giữa các thực thể. Trong hệ thống thương mại điện tử, một khách hàng có thể thực hiện nhiều đơn hàng (1:N), và một đơn hàng có thể chứa nhiều sản phẩm (N:M).

Các thuộc tính (attributes) trong ERD mô tả đặc điểm của thực thể hoặc quan hệ. Có các loại thuộc tính như thuộc tính đơn giản (simple attributes), thuộc tính phức hợp (composite attributes), thuộc tính đa trị (multivalued attributes) và thuộc tính dẫn xuất (derived attributes). Ví dụ, thuộc tính "tuổi" có thể là thuộc tính dẫn xuất từ ngày sinh.

Lợi ích của mô hình ERD là giúp xác định rõ các thực thể, mối quan hệ và thuộc tính của hệ thống, từ đó tối ưu hóa thiết kế cơ sở dữ liệu. ERD giúp phát hiện sớm các vấn đề về thiết kế dữ liệu, đảm bảo dữ liệu không bị dư thừa hoặc thiếu sót. Ngoài ra, ERD còn giúp lập trình viên, nhà phân tích và các bên liên quan dễ dàng hiểu và triển khai hệ thống.

Trong dự án thương mại điện tử này, ERD được sử dụng để thiết kế cơ sở dữ liệu, xác định các thực thể quan trọng như Users (Người dùng), Products (Sản phẩm), Orders (Đơn hàng), Cart (Giỏ hàng), Order\_Items (Chi tiết đơn hàng) … và các mối quan hệ giữa chúng. Điều này giúp đảm bảo hệ thống có cấu trúc dữ liệu hợp lý, hỗ trợ tốt cho các chức năng như mua hàng, quản lý đơn hàng và thanh toán.

### Lược đồ quan hệ (Relational Schema)

Mô hình dữ liệu quan hệ (Relational Data Model) là một mô hình tổ chức dữ liệu sử dụng các bảng (relations) để lưu trữ thông tin. Mỗi bảng bao gồm các hàng (records/tuples) và cột (attributes/fields), trong đó mỗi hàng đại diện cho một bản ghi dữ liệu và mỗi cột biểu diễn một thuộc tính của dữ liệu. Mô hình này được sử dụng rộng rãi trong các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) như MySQL, PostgreSQL, SQL Server.

Lược đồ quan hệ (Relational Schema) là một cách biểu diễn cấu trúc của mô hình dữ liệu quan hệ bằng cách mô tả các bảng, thuộc tính, khóa chính (primary key), khóa ngoại (foreign key) và mối quan hệ giữa các bảng. Một hệ thống thương mại điện tử thường có các lược đồ quan hệ chính như:

* Users (Người dùng): Chứa thông tin về khách hàng hoặc quản trị viên với các thuộc tính như id, name, email, password, role, created\_at, updated\_at.
* Products (Sản phẩm): Lưu thông tin sản phẩm với các thuộc tính như id, name, price, sale\_price, type, image, amount, description, created\_at, updated\_at.
* Orders (Đơn hàng): Chứa thông tin về đơn hàng với các thuộc tính như id, user\_id (khóa ngoại liên kết với bảng Users), total\_price, status, shipping\_address, created\_at, updated\_at.
* Cart (Giỏ hàng): Lưu trữ các sản phẩm mà người dùng đã thêm vào giỏ hàng nhưng chưa thanh toán, với các thuộc tính như id, user\_id, product\_id, quantity, created\_at, updated\_at.
* Order\_Items (Chi tiết đơn hàng): Liên kết giữa Orders và Products để lưu thông tin về các sản phẩm có trong đơn hàng, gồm các thuộc tính id, order\_id (khóa ngoại liên kết với Orders), product\_id (khóa ngoại liên kết với Products), quantity, price, created\_at, updated\_at.

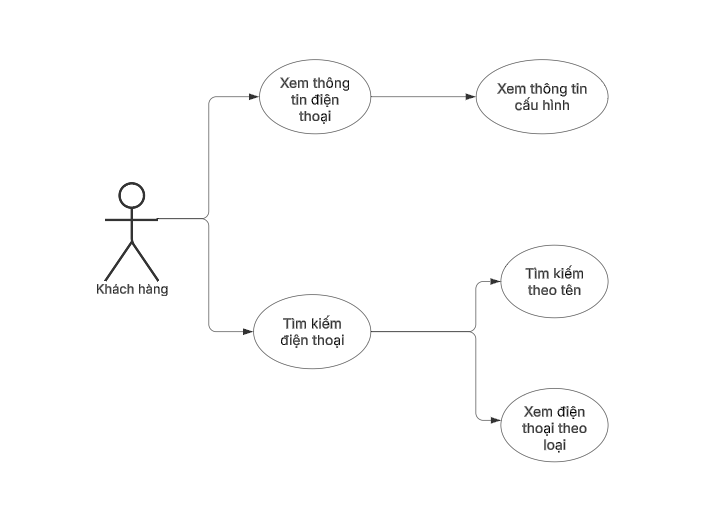
Các bảng trong mô hình quan hệ liên kết với nhau thông qua khóa ngoại (foreign key), đảm bảo tính toàn vẹn dữ liệu và hỗ trợ thực thi các ràng buộc giữa các thực thể. Mô hình dữ liệu quan hệ giúp tối ưu hóa việc lưu trữ, đảm bảo truy xuất dữ liệu nhanh chóng và tránh dư thừa dữ liệu thông qua chuẩn hóa cơ sở dữ liệu (database normalization).

Trong dự án thương mại điện tử này, lược đồ quan hệ được sử dụng để tổ chức dữ liệu một cách logic và hỗ trợ các chức năng như quản lý sản phẩm, giỏ hàng, đơn hàng, thanh toán, đảm bảo hệ thống hoạt động ổn định và có khả năng mở rộng trong tương lai.

## Phân thích và thiết kế hệ thống

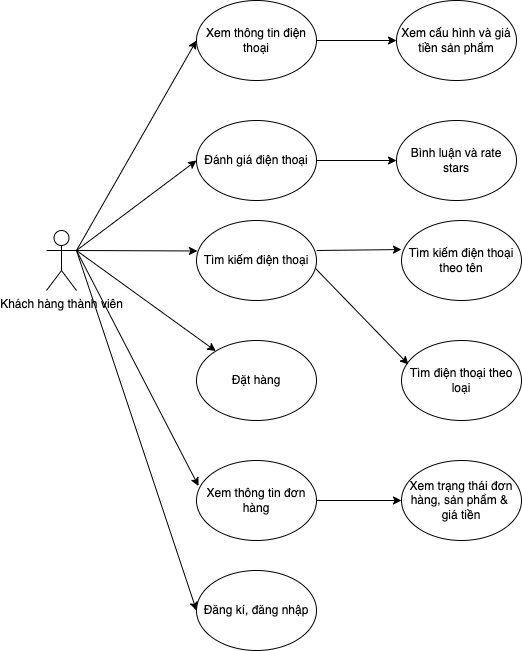
### Use Case Diagram

#### Sơ đồ use-case của khách hàng



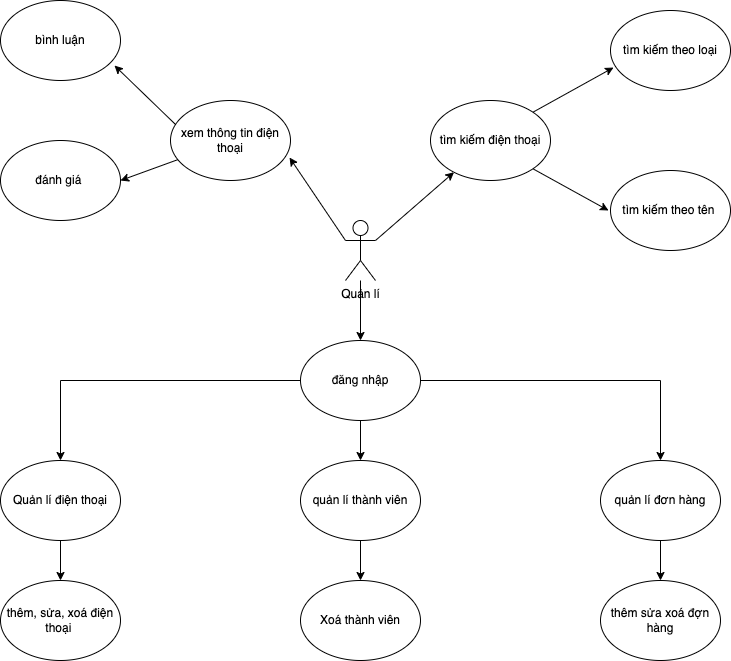
Hình .: Sơ đồ use-case của khách hàng

#### Sơ đồ use-case của khách hàng thành viên



Hình .: Sơ đồ use-case của khách hàng thành viên

#### Sơ đồ use-case của admin



Hình .: Sơ đồ use-case của admin

### Đặc tả Use Case

#### Xem thông tin điện thoại

|  |  |
| --- | --- |
| ***Tên Use Case*** | ***Xem thông tin điện thoại*** |
| Mô tả | Cho phép người dùng xem thông tin chi tiết về điện thoại |
| Tác nhân | Khách hàng, Khách hàng thành viên |
| Tiền điều kiện | Người dùng truy cập vào trang web |
| Luồng sự kiện chính | 1. Người dùng chọn một sản phẩm điện thoại  2. Hệ thống hiển thị thông tin chi tiết về điện thoại (cấu hình, giá, hình ảnh, mô tả) |
| Luồng sự kiện phụ | 1a. Nếu sản phẩm không tồn tại, hệ thống hiển thị thông báo lỗi |

Bảng .: Bảng đặc tả use-case xem thông tin điện thoại

#### Tìm kiếm điện thoại

|  |  |
| --- | --- |
| ***Tên Use Case*** | ***Tìm kiếm điện thoại*** |
| Mô tả | Cho phép người dùng tìm kiếm điện thoại theo tiêu chí cụ thể |
| Tác nhân | Khách hàng, Khách hàng thành viên |
| Tiền điều kiện | Người dùng truy cập vào trang web |
| Luồng sự kiện chính | 1. Người dùng chọn một sản phẩm điện thoại  2. Hệ thống hiển thị thông tin chi tiết về điện thoại (cấu hình, giá, hình ảnh, mô tả) |
| Luồng sự kiện phụ | 1a. Người dùng có thể tìm kiếm theo tên hoặc loại điện thoại  2a. Nếu không có kết quả phù hợp, hệ thống hiển thị thông báo |

Bảng .: Bảng đặc tả use-case tìm kiếm điện thoại

#### Đánh giá điện thoại

|  |  |
| --- | --- |
| ***Tên Use Case*** | ***Đánh giá điện thoại*** |
| Mô tả | Cho phép người dùng đánh giá điện thoại |
| Tác nhân | Khách hàng thành viên |
| Tiền điều kiện | Người dùng đã đăng nhập vào hệ thống |
| Luồng sự kiện chính | 1. Người dùng chọn sản phẩm muốn đánh giá  2. Người dùng nhập nội dung đánh giá và số sao  3. Hệ thống lưu đánh giá và hiển thị công khai |
| Luồng sự kiện phụ | 1a. Nếu người dùng chưa đăng nhập, hệ thống yêu cầu đăng nhập trước khi đánh giá |

Bảng .: Bảng đặc tả use-case đánh giá điện thoại

#### Đặt hàng

|  |  |
| --- | --- |
| ***Tên Use Case*** | ***Đặt hàng*** |
| Mô tả | Cho phép khách hàng đặt mua sản phẩm |
| Tác nhân | Khách hàng thành viên |
| Tiền điều kiện | Người dùng đã thêm sản phẩm vào giỏ hàng và đăng nhập vào hệ thống |
| Luồng sự kiện chính | 1. Người dùng vào giỏ hàng và chọn sản phẩm cần mua  2. Hệ thống hiển thị thông tin sản phẩm, tổng tiền và hình thức thanh toán  3. Người dùng xác nhận đơn hàng  4. Hệ thống lưu đơn hàng và hiển thị trạng thái |
| Luồng sự kiện phụ | 1a. Nếu giỏ hàng trống, hệ thống hiển thị thông báo |

Bảng .: Bảng đặc tả use-case đặt hàng

#### Quản lý sản phẩm (Admin)

|  |  |
| --- | --- |
| ***Tên Use Case*** | ***Quản lý sản phẩm*** |
| Mô tả | Quản trị viên có thể thêm, sửa, xóa sản phẩm |
| Tác nhân | Quản trị viên |
| Tiền điều kiện | Người dùng có quyền Admin và đăng nhập vào hệ thống |
| Luồng sự kiện chính | 1. Admin vào trang quản trị sản phẩm  2. Admin chọn thêm mới, chỉnh sửa hoặc xóa sản phẩm  3. Hệ thống cập nhật dữ liệu vào cơ sở dữ liệu |
| Luồng sự kiện phụ | 2a. Nếu sản phẩm đã tồn tại, hệ thống yêu cầu cập nhật thay vì thêm mới |

Bảng .: Bảng đặc tả use-case quản lý sản phẩm

#### Quản lý đơn hàng (Admin)

|  |  |
| --- | --- |
| ***Tên Use Case*** | ***Quản lý đơn hàng*** |
| Mô tả | Quản trị viên có thể xem, sửa trạng thái đơn hàng |
| Tác nhân | Quản trị viên |
| Tiền điều kiện | Admin đã đăng nhập vào hệ thống |
| Luồng sự kiện chính | 1. Admin vào trang quản lý đơn hàng  2. Admin xem danh sách đơn hàng  3. Admin có thể thay đổi trạng thái đơn hàng (đang xử lý, đang giao, hoàn thành, hủy đơn)  4. Hệ thống cập nhật trạng thái vào cơ sở dữ liệu |
| Luồng sự kiện phụ | 3a. Nếu đơn hàng đã bị hủy, admin không thể chỉnh sửa trạng thái |

Bảng .: Bảng đặc tả use-case quản lý đơn hàng

### Lược đồ ERD

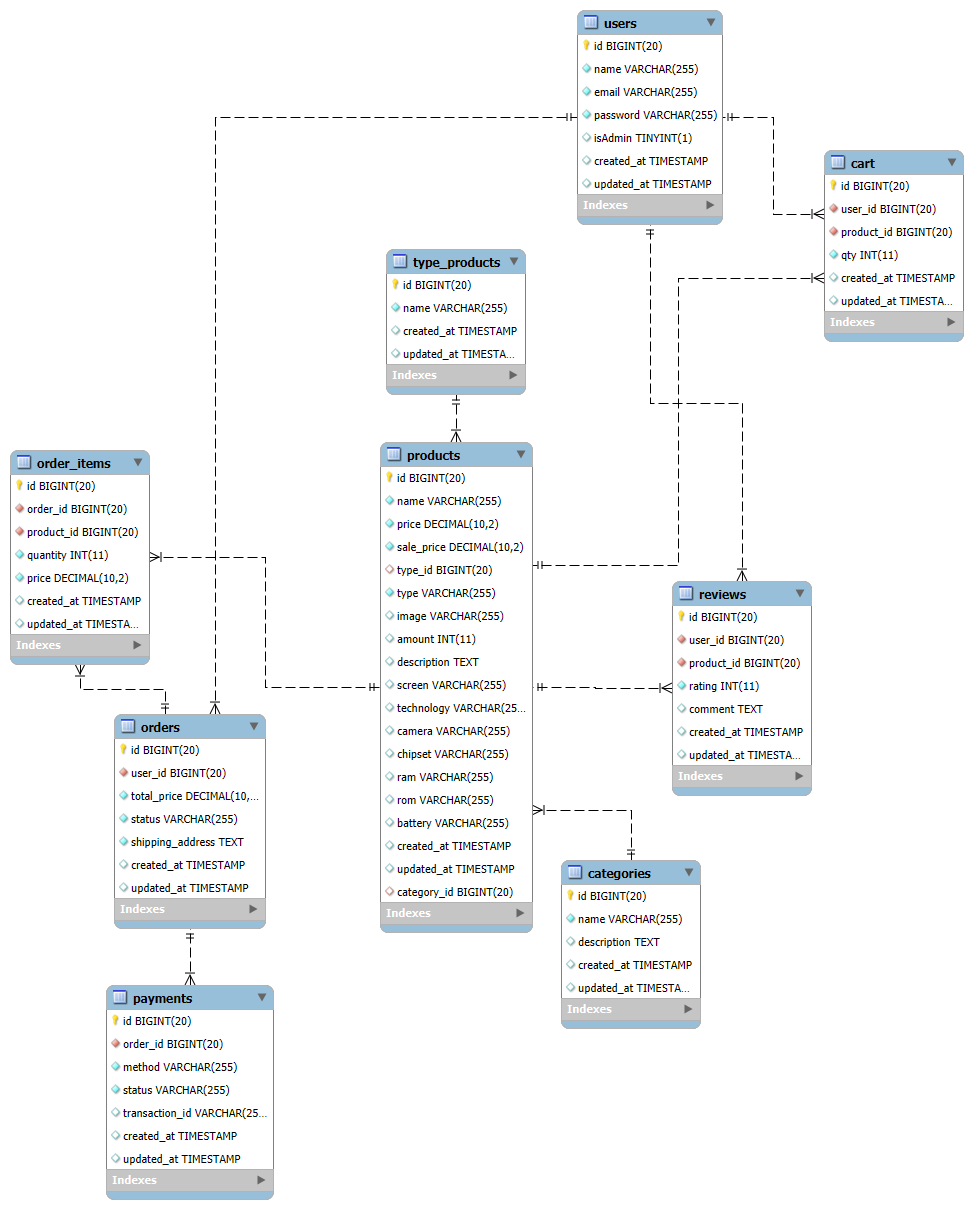
#### Thực thể chính

* **Users (Người dùng)**: Lưu thông tin về người dùng, bao gồm id, name, email, password, và isAdmin (phân quyền quản trị).
* **Products (Sản phẩm)**: Chứa thông tin về sản phẩm, bao gồm id, name, price, sale\_price, type, image, description, và amount (số lượng tồn kho).
* **Orders (Đơn hàng)**: Lưu thông tin về đơn hàng của người dùng với các thuộc tính id, user\_id (liên kết với bảng Users), total\_price, status, shipping\_address, payment\_method, created\_at, và updated\_at.
* **Order\_Items (Chi tiết đơn hàng)**: Chứa thông tin các sản phẩm có trong đơn hàng, gồm id, order\_id (liên kết với Orders), product\_id (liên kết với Products), quantity, và price.
* **Cart (Giỏ hàng)**: Lưu thông tin giỏ hàng của người dùng, gồm id, user\_id (liên kết với Users), created\_at, và updated\_at.
* **Reviews (Đánh giá sản phẩm)**: Chứa thông tin đánh giá sản phẩm của người dùng, gồm id, user\_id (liên kết với Users), product\_id (liên kết với Products), rating, comment, và created\_at.
* **Categories (Danh mục sản phẩm)**: Lưu thông tin phân loại sản phẩm, gồm id, name, description, và created\_at.
* **Payments (Thanh toán)**: Lưu thông tin giao dịch thanh toán, gồm id, order\_id (liên kết với Orders), method (phương thức thanh toán như VNPAY, MoMo), status, transaction\_id, và created\_at.

#### Quan hệ giữa các thực thể

* **Users - Orders**: Một người dùng có thể có nhiều đơn hàng (1:N).
* **Orders - Order\_Items**: Một đơn hàng có nhiều sản phẩm (1:N).
* **Products - Order\_Items**: Một sản phẩm có thể xuất hiện trong nhiều đơn hàng (1:N).
* **Users - Cart**: Một người dùng chỉ có một giỏ hàng (1:1).
* **Cart - Products**: Một giỏ hàng có thể chứa nhiều sản phẩm, và một sản phẩm có thể nằm trong nhiều giỏ hàng (N:M).
* **Users - Reviews**: Một người dùng có thể đánh giá nhiều sản phẩm (1:N).
* **Products - Reviews**: Một sản phẩm có thể có nhiều đánh giá từ nhiều người dùng (1:N).
* **Categories - Products**: Một danh mục có nhiều sản phẩm (1:N).
* **Orders - Payments**: Một đơn hàng có một thanh toán tương ứng (1:1).

#### ERD mô tả lược đồ quan hệ



Hình .: ERD mô tả lược đồ quan hệ

### Cơ sở dữ liệu

#### Bảng Users (Khách hàng)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| name | *VARCHAR(255)* | Họ và tên người dùng |
| email | *VARCHAR(255)* | Địa chỉ email (duy nhất) |
| password | *VARCHAR(255)* | Mật khẩu đã mã hóa |
| isAdmin | *BOOLEAN* | Xác định quyền admin (true/false) |
| created\_at | *TIMESTAMP* | Thời gian tạo tài khoản |
| updated\_at | *TIMESTAMP* | Thời gian cập nhật |

Bảng .: Bảng Users (Khách hàng)

#### Bảng Products (Sản phẩm)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| name | *VARCHAR(255)* | Tên sản phẩm |
| price | *DECIMAL(10,2)* | Giá sản phẩm |
| sale\_price | *DECIMAL(10,2)* | Giá khuyến mãi (nếu có) |
| type | *VARCHAR(255)* | Loại sản phẩm |
| category\_id | *BIGINT (FK)* | Liên kết với Categories |
| image | *VARCHAR(255)* | Đường dẫn ảnh sản phẩm |
| amount | *INT* | Số lượng tồn kho |
| description | *TEXT* | Mô tả sản phẩm |
| created\_at | *TIMESTAMP* | Thời gian tạo |
| updated\_at | *TIMESTAMP* | Thời gian cập nhật |

Bảng .: Bảng Products (Sản phẩm)

#### Bảng Orders (Đơn hàng)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| user\_id | *BIGINT (FK)* | Liên kết với bảng Users |
| total\_price | *DECIMAL(10,2)* | Tổng giá trị đơn hàng |
| status | *VARCHAR(50)* | Trạng thái đơn hàng (pending, shipped, delivered) |
| payment\_method | *VARCHAR(50)* | Phương thức thanh toán (VNPAY, MoMo) |
| shipping\_address | *TEXT* | Địa chỉ giao hàng |
| created\_at | *TIMESTAMP* | Thời gian tạo |
| updated\_at | *TIMESTAMP* | Thời gian cập nhật |

Bảng .: Bảng Orders (Đơn hàng)

#### Bảng Order\_Items (Chi tiết đơn hàng)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| order\_id | *BIGINT (FK)* | Liên kết với bảng Orders |
| product\_id | *BIGINT (FK)* | Liên kết với bảng Products |
| quantity | *INT* | Số lượng sản phẩm trong đơn hàng |
| price | *DECIMAL(10,2)* | Giá sản phẩm tại thời điểm đặt hàng |
| created\_at | *TIMESTAMP* | Thời gian tạo |
| updated\_at | *TIMESTAMP* | Thời gian cập nhật |

Bảng .: Bảng Order\_Items (Chi tiết đơn hàng)

#### Bảng Cart (Giỏ hàng)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| user\_id | *BIGINT (FK)* | Liên kết với bảng Users |
| product\_id | *BIGINT (FK)* | Liên kết với bảng Products |
| quantity | *INT* | Số lượng sản phẩm trong giỏ hàng |
| created\_at | *TIMESTAMP* | Thời gian tạo |
| updated\_at | *TIMESTAMP* | Thời gian cập nhật |

Bảng .: Bảng Cart (Giỏ hàng)

#### Bảng Reviews (Đánh giá sản phẩm)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| user\_id | *BIGINT (FK)* | Liên kết với bảng Users |
| product\_id | *BIGINT (FK)* | Liên kết với bảng Products |
| rating | *INT* | Điểm đánh giá (1-5 sao) |
| comment | *TEXT* | Nội dung đánh giá |
| created\_at | *TIMESTAMP* | Thời gian tạo |

Bảng .: Bảng Reviews (Đánh giá sản phẩm)

#### Bảng Categories (Danh mục sản phẩm)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| name | *VARCHAR(255)* | Tên danh mục |
| description | *TEXT* | Mô tả danh mục |
| created\_at | *TIMESTAMP* | Thời gian tạo |

Bảng .: Bảng Categories (Danh mục sản phẩm)

#### Bảng Payments (Thanh toán)

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Mô tả** |
| id | *BIGINT (PK)* | Khóa chính |
| order\_id | *BIGINT (FK)* | Liên kết với bảng Orders |
| method | *VARCHAR(50)* | Phương thức thanh toán (VNPAY, MoMo) |
| status | *VARCHAR(50)* | Trạng thái thanh toán (pending, success, failed) |
| transaction\_id | *VARCHAR(255)* | Mã giao dịch |
| created\_at | *TIMESTAMP* | Thời gian tạo |

Bảng 2.14: Bảng Payments (Thanh toán)

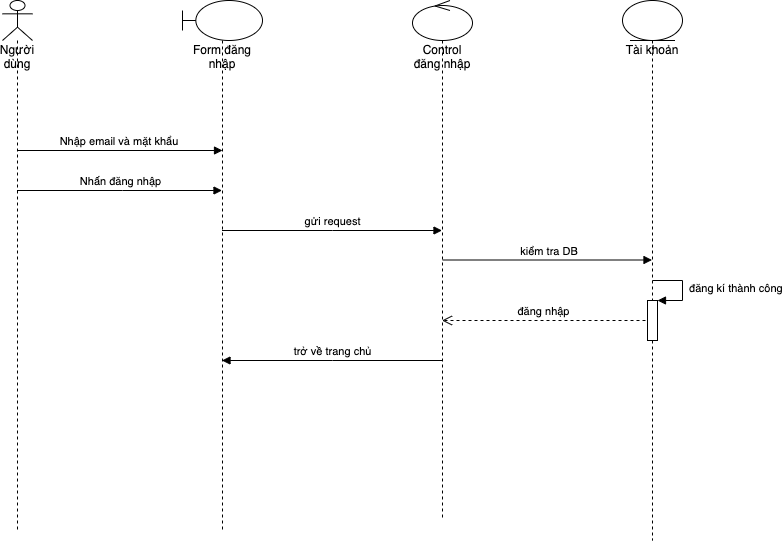
Bảng .: Bảng Payments (Thanh toán)

#### Các ràng buộc khóa ngoại

* Orders.user\_id → Users.id
* Order\_Items.order\_id → Orders.id
* Order\_Items.product\_id → Products.id
* Cart.user\_id → Users.id
* Cart.product\_id → Products.id
* Reviews.user\_id → Users.id
* Reviews.product\_id → Products.id
* Products.category\_id → Categories.id
* Payments.order\_id → Orders.id

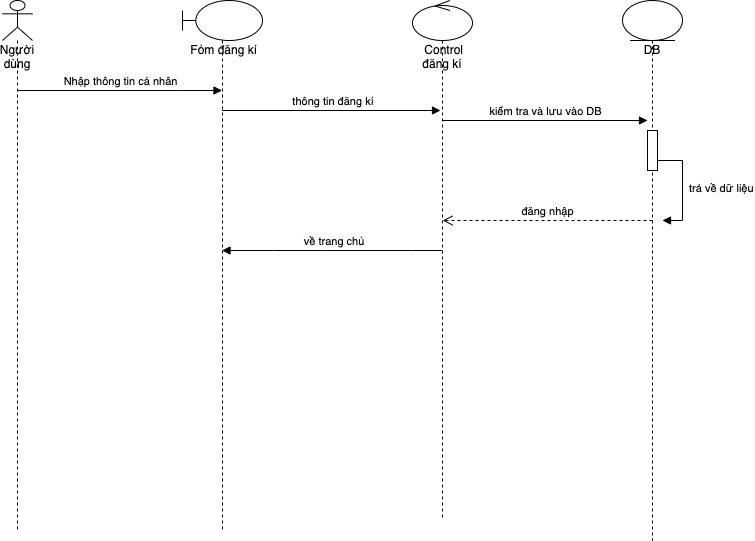
### Sequence Diagram

#### Biểu đồ tuần tự đăng nhập



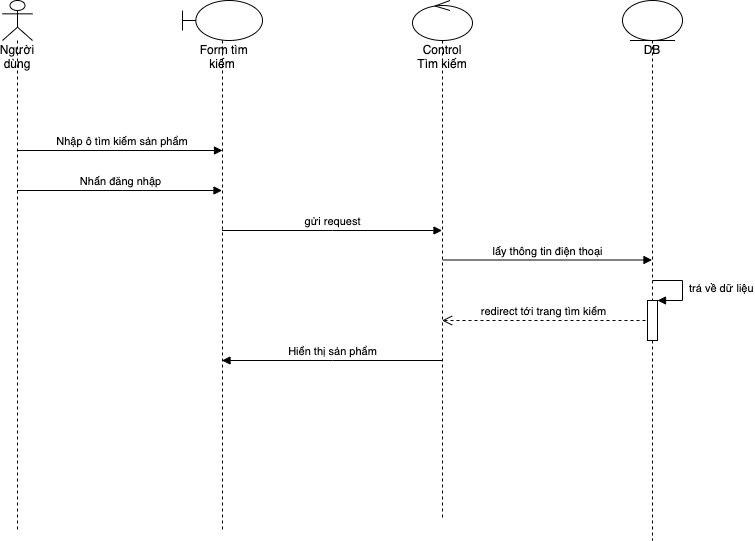
Hình .: Biểu đồ tuần tự đăng nhập

#### Biểu đồ tuần tự đăng kí



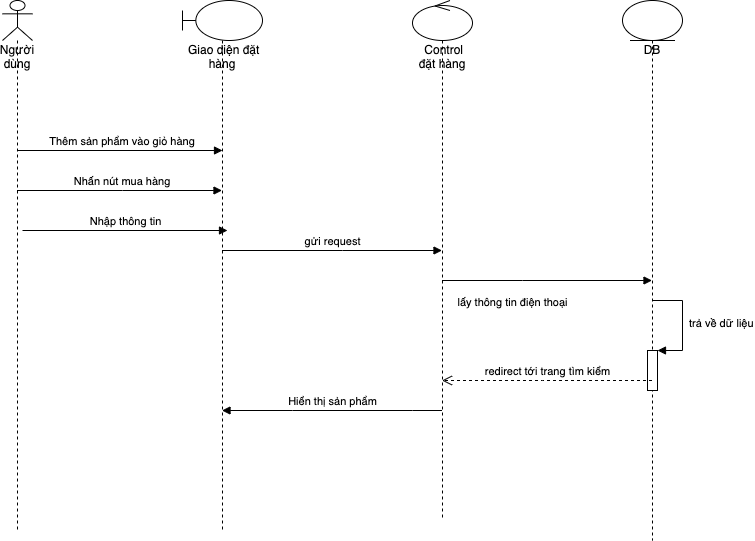
Hình .: Biểu đồ tuần tự đăng kí

#### Biểu đồ tuần tự tìm kiếm sản phẩm



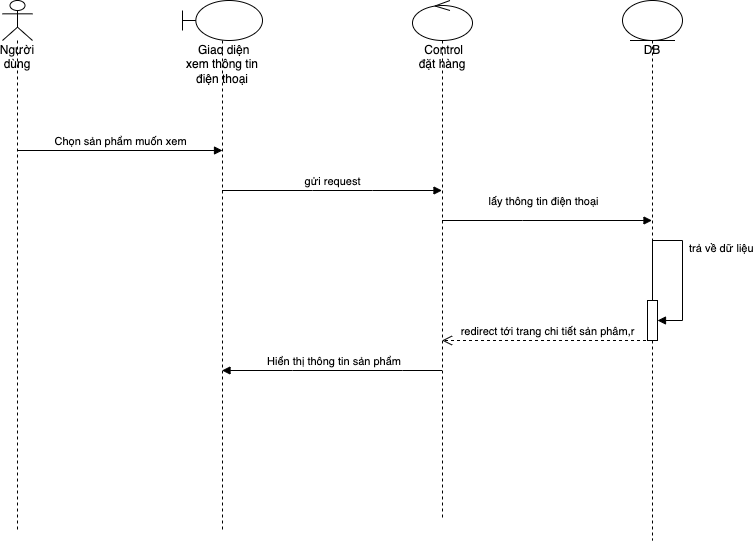
Hình .: Biểu đồ tuần tự tìm kiếm sản phẩm

#### Biểu đồ tuần tự đặt hàng



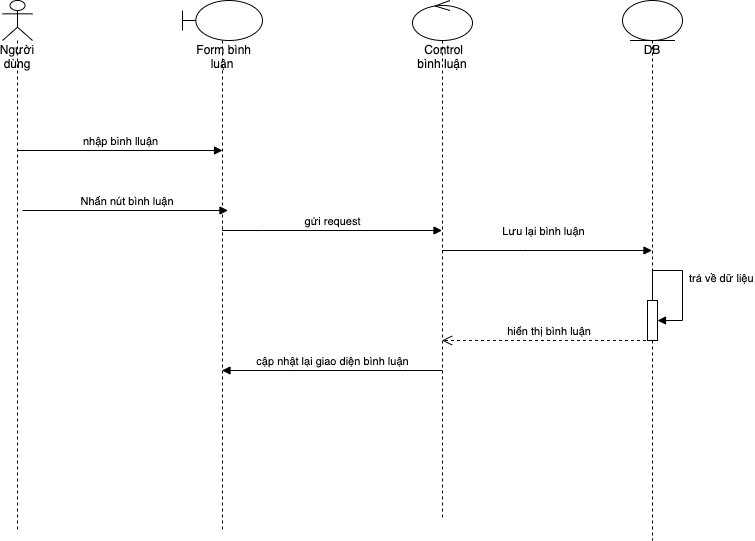
Hình .: Biểu đồ tuần tự đặt hàng

#### Biểu đồ tuần tự xem thông tin sản phẩm



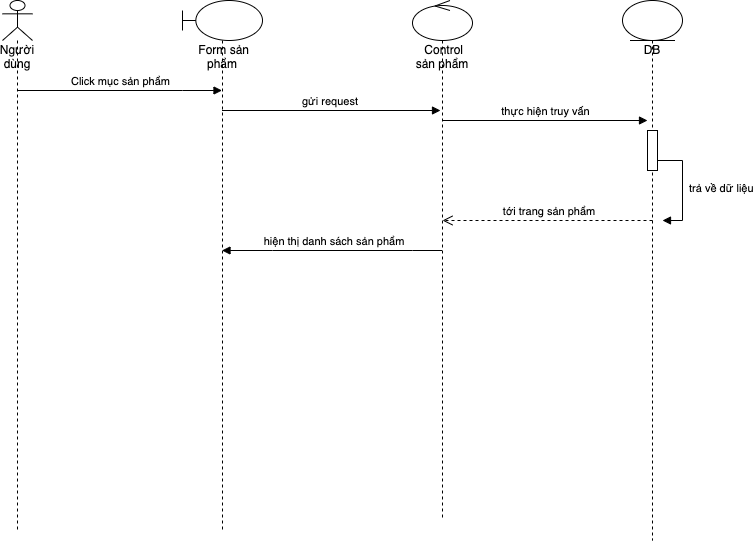
Hình .: Biểu đồ tuần tự xem thông tin sản phẩm

#### Biểu đồ tuần tự bình luận



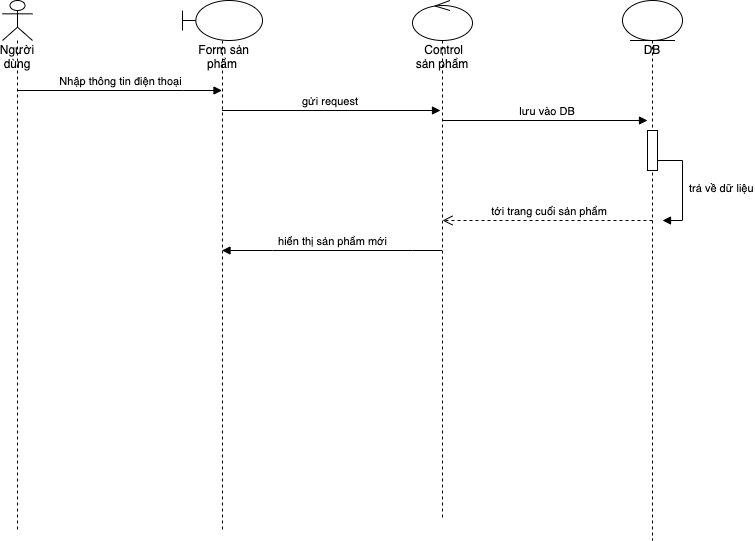
Hình .: Biểu đồ tuần tự bình luận

#### Biểu đồ tuần tự hiển thị danh sách sản phẩm



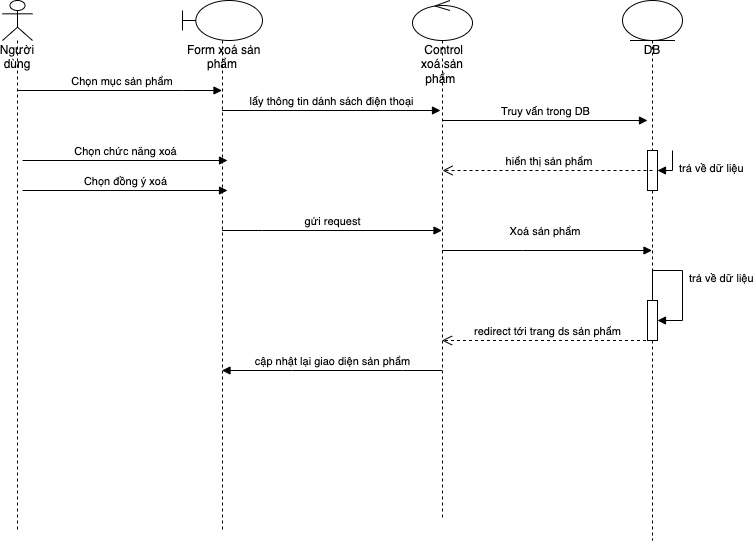
Hình .: Biểu đồ tuần tự hiển thị danh sách sản phẩm

#### Biểu đồ tuần tự thêm sản phẩm



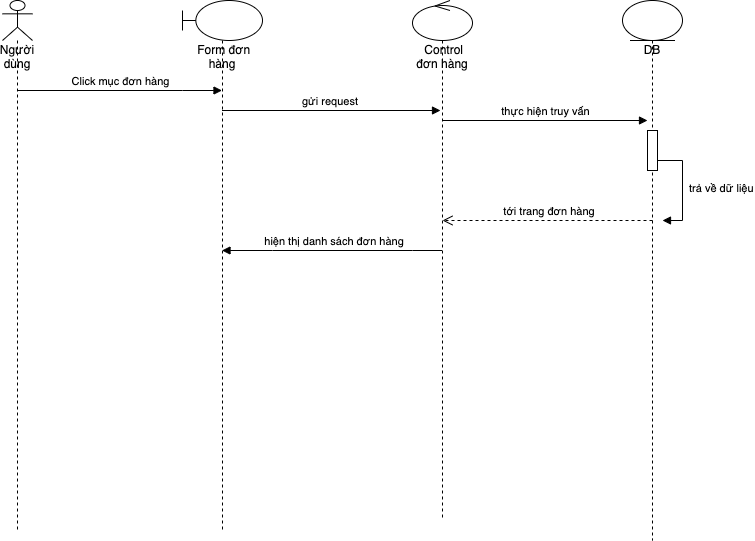
Hình .: Biểu đồ tuần tự thêm sản phẩm

#### Biểu đồ tuần tự xóa sản phẩm



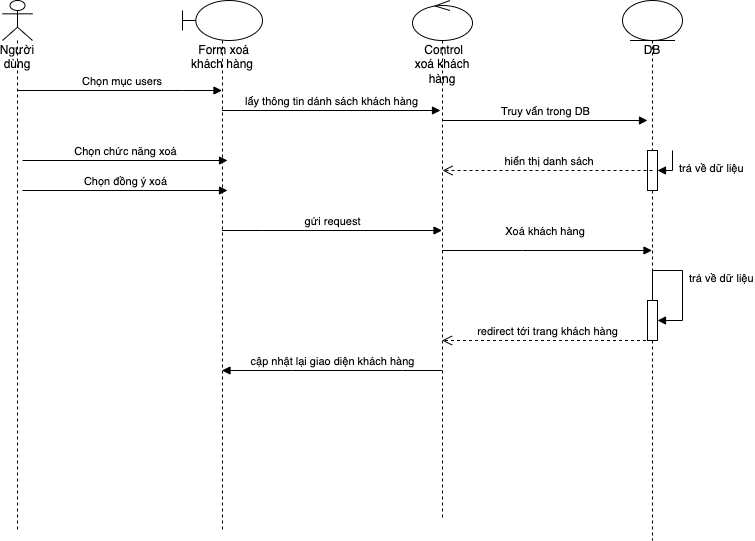
Hình .: Biểu đồ tuần tự xóa sản phẩm

#### Biểu đồ tuần tự xem sản phẩm



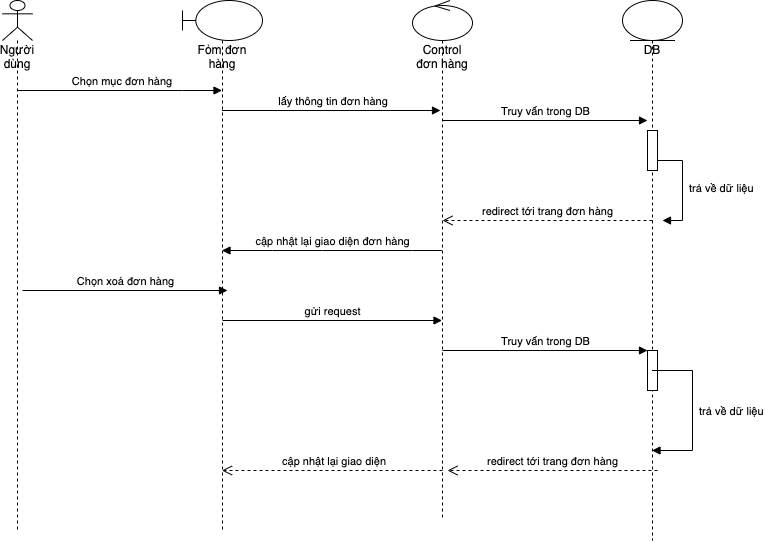
Hình .: Biểu đồ tuần tự xem sản phẩm

#### Biểu đồ tuần tự xoá người dùng



Hình .: Biểu đồ tuần tự xoá người dùng

#### Biểu đồ tuần tự xoá đơn hàng



Hình .: Biểu đồ tuần tự xoá đơn hàng

## Công nghệ bảo mật

### Xác thực và phân quyền người dùng

Laravel Sanctum và JWT (JSON Web Token) đều là các phương pháp xác thực trong Laravel, mỗi phương pháp có ưu điểm riêng và phù hợp với từng trường hợp cụ thể.

Laravel Sanctum là một hệ thống xác thực API nhẹ dành cho các ứng dụng Laravel, cung cấp phương thức xác thực đơn giản mà không cần cấu hình phức tạp như OAuth. Sanctum sử dụng token dựa trên cookie và bearer token để bảo vệ các route API, giúp dễ dàng quản lý phiên đăng nhập trong ứng dụng SPA (Single Page Application) hoặc ứng dụng di động. Sanctum hoạt động bằng cách lưu token trên trình duyệt và sử dụng cơ chế bảo vệ CSRF (Cross-Site Request Forgery), giúp tăng tính bảo mật cho ứng dụng.

JWT (JSON Web Token) là một tiêu chuẩn xác thực không trạng thái (stateless authentication), sử dụng token để truyền tải thông tin người dùng mà không cần lưu trạng thái trên server. JWT bao gồm ba phần: Header, Payload và Signature, giúp bảo mật và xác thực dữ liệu được gửi qua mạng. Laravel có thể sử dụng thư viện tymon/jwt-auth để triển khai JWT. Phương thức này phù hợp với các ứng dụng yêu cầu xác thực API không phụ thuộc vào phiên (session), giúp tối ưu hóa hiệu suất và khả năng mở rộng.

Khi lựa chọn giữa Sanctum và JWT, cần xem xét nhu cầu của ứng dụng:

* Sanctum phù hợp cho ứng dụng SPA hoặc ứng dụng Laravel thuần túy, nơi có thể sử dụng cơ chế xác thực session-based và cookie.
* JWT phù hợp với các ứng dụng đa nền tảng (mobile, web, API độc lập) cần xác thực không trạng thái để duy trì tính linh hoạt.

Laravel cũng cung cấp Passport, một hệ thống OAuth2 đầy đủ chức năng, phù hợp với các ứng dụng lớn hơn, yêu cầu xác thực phức tạp.

Việc lựa chọn Sanctum hay JWT phụ thuộc vào kiến trúc ứng dụng và yêu cầu bảo mật, trong đó Sanctum dễ thiết lập và sử dụng hơn, trong khi JWT linh hoạt hơn cho hệ thống API không trạng thái.

### Bảo mật dữ liệu

Bcrypt là một thuật toán băm mật khẩu được thiết kế để bảo vệ thông tin đăng nhập khỏi các cuộc tấn công như brute-force và rainbow table. Bcrypt sử dụng một hàm băm chậm và có khả năng tự động thêm salt để đảm bảo rằng ngay cả khi hai mật khẩu giống nhau được băm, chúng vẫn tạo ra kết quả khác nhau. Bcrypt còn cho phép điều chỉnh cost factor, giúp tăng độ phức tạp của quá trình băm theo thời gian nhằm nâng cao tính bảo mật.

Middleware trong Laravel là một lớp trung gian giúp kiểm soát và xử lý các request trước khi chúng được chuyển đến Controller. Middleware có thể được sử dụng để kiểm tra xác thực, kiểm soát quyền truy cập, xử lý CORS, giới hạn tốc độ request, ghi log và thực hiện nhiều tác vụ khác. Laravel cung cấp một số middleware mặc định như xác thực người dùng (auth), giới hạn request (throttle), và xử lý CORS (cors). Ngoài ra, Laravel cho phép tạo middleware tùy chỉnh để thực hiện các logic bảo mật và điều hướng luồng dữ liệu phù hợp với yêu cầu của ứng dụng.

### Bảo mật giao tiêp dữ liệu

#### CORS (Cross-Origin Resource Sharing)

CORS (Cross-Origin Resource Sharing) là một cơ chế bảo mật của trình duyệt cho phép hoặc từ chối các yêu cầu HTTP được gửi từ một nguồn khác với nguồn gốc của trang web. Mặc định, trình duyệt sẽ chặn các yêu cầu AJAX từ một domain khác để ngăn chặn các cuộc tấn công bảo mật như Cross-Site Request Forgery (CSRF).

CORS hoạt động bằng cách thêm các HTTP headers vào phản hồi của máy chủ để chỉ định các nguồn gốc được phép truy cập tài nguyên. Nếu không có header CORS hoặc nếu nguồn gốc của yêu cầu không được máy chủ cho phép, trình duyệt sẽ chặn yêu cầu.

Một số header quan trọng trong CORS:

* Access-Control-Allow-Origin: Xác định domain nào được phép truy cập tài nguyên.
* Access-Control-Allow-Methods: Xác định các phương thức HTTP (GET, POST, PUT, DELETE, v.v.) được phép.
* Access-Control-Allow-Headers: Xác định các headers được phép gửi trong yêu cầu.
* Access-Control-Allow-Credentials: Cho phép hoặc từ chối gửi thông tin xác thực (cookies, tokens) trong yêu cầu.

Khi một yêu cầu CORS được gửi, nếu nó là một simple request (chỉ sử dụng các phương thức như GET, POST không có header tùy chỉnh), trình duyệt sẽ gửi trực tiếp yêu cầu đến máy chủ. Nếu là một preflight request (yêu cầu có header tùy chỉnh hoặc sử dụng các phương thức như PUT, DELETE), trình duyệt sẽ gửi một yêu cầu OPTIONS trước để kiểm tra xem máy chủ có cho phép hay không.

Trong dự án thương mại điện tử, CORS thường được sử dụng khi frontend (ReactJS) và backend (Laravel) chạy trên các domain khác nhau. Nếu CORS không được cấu hình đúng, các yêu cầu API từ frontend sẽ bị trình duyệt chặn. Laravel hỗ trợ CORS thông qua middleware hoặc package như barryvdh/laravel-cors, cho phép định cấu hình các domain được phép truy cập API.

#### HTTPS

HTTPS (HyperText Transfer Protocol Secure) là phiên bản bảo mật của HTTP, được sử dụng để truyền tải dữ liệu trên internet một cách an toàn bằng cách mã hóa thông tin trao đổi giữa trình duyệt và máy chủ web. HTTPS kết hợp giao thức HTTP với TLS (Transport Layer Security) hoặc SSL (Secure Sockets Layer) để bảo vệ tính toàn vẹn và bảo mật dữ liệu trong quá trình truyền tải.

HTTPS đảm bảo ba yếu tố quan trọng:

* Mã hóa (Encryption): Mọi dữ liệu truyền giữa trình duyệt và máy chủ đều được mã hóa, giúp ngăn chặn các cuộc tấn công đánh cắp dữ liệu (man-in-the-middle).
* Xác thực (Authentication): HTTPS sử dụng chứng chỉ số (SSL/TLS Certificate) để xác minh danh tính của website, giúp người dùng tránh bị lừa đảo bởi các trang web giả mạo (phishing).
* Toàn vẹn dữ liệu (Data Integrity): Dữ liệu không thể bị thay đổi hoặc giả mạo trong quá trình truyền tải mà không bị phát hiện.

HTTPS hoạt động bằng cách thiết lập một kết nối bảo mật qua TLS/SSL, trong đó:

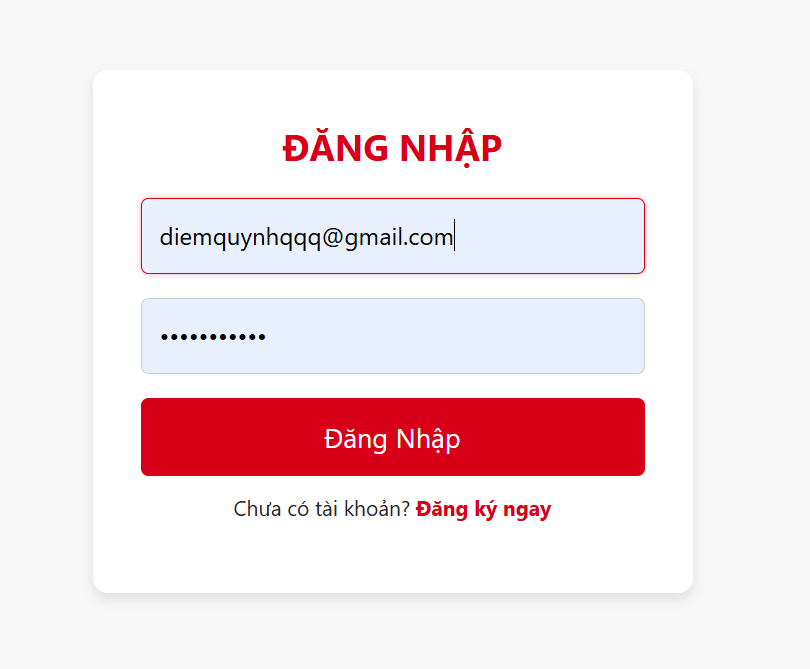
* Trình duyệt gửi yêu cầu đến máy chủ để thiết lập kết nối an toàn.
* Máy chủ gửi chứng chỉ SSL/TLS chứa thông tin mã hóa.
* Trình duyệt xác minh chứng chỉ thông qua các cơ quan chứng nhận (CA – Certificate Authority).
* Nếu hợp lệ, một phiên làm việc an toàn được thiết lập, cho phép trao đổi dữ liệu được mã hóa.

Trong các ứng dụng thương mại điện tử, HTTPS đóng vai trò quan trọng trong bảo vệ thông tin người dùng, tài khoản, thanh toán trực tuyến và đảm bảo rằng các giao dịch được thực hiện một cách an toàn. Việc sử dụng HTTPS không chỉ giúp tăng độ tin cậy của website mà còn là một yếu tố quan trọng trong SEO, do Google ưu tiên xếp hạng các trang web sử dụng HTTPS.

# GIAO DIỆN HỆ THỐNG VÀ LUỒNG XỬ LÝ CHỨC NĂNG

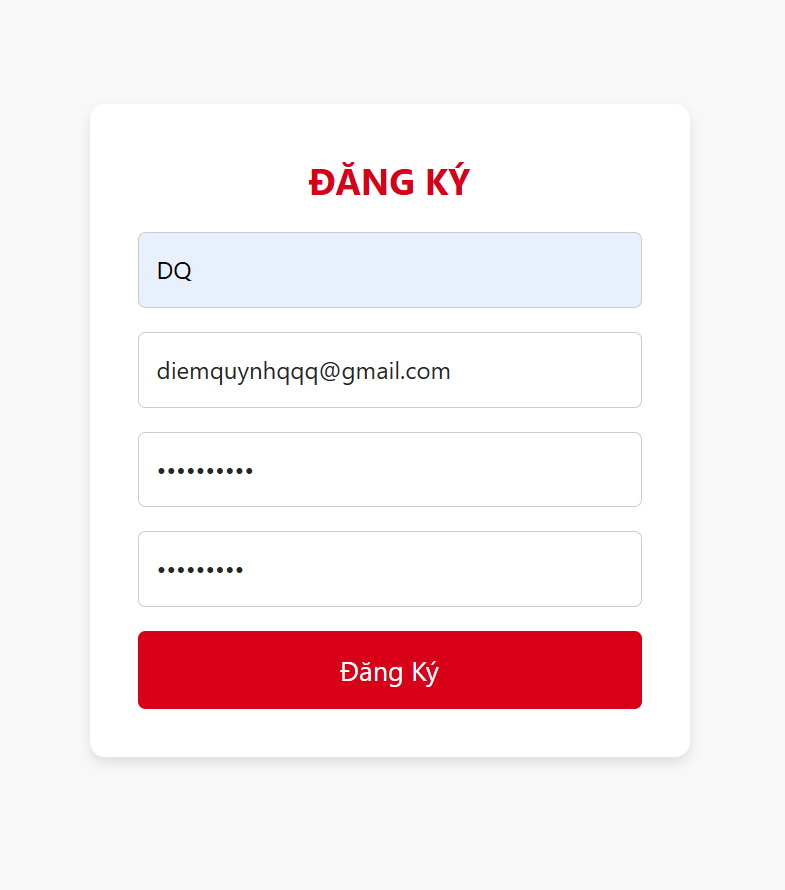
## Trang chủ

## Đăng nhập



Hình .: Giao diện trang đăng nhập

## Đăng kí



Hình .: Giao diện trang đăng kí

# KẾT LUẬN

## Kết luận

Dự án Xây dựng ứng dụng thương mại điện tử full-stack trên nền tảng web app đã được hoàn thành với đầy đủ các tính năng cốt lõi, bao gồm quản lý sản phẩm, giỏ hàng, thanh toán trực tuyến, quản lý đơn hàng và phân quyền người dùng. Ứng dụng được phát triển dựa trên mô hình Client-Server với Front-end sử dụng ReactJS và Back-end sử dụng PHP Laravel, đảm bảo tính tương tác mượt mà, tốc độ xử lý nhanh và khả năng mở rộng trong tương lai.

Trong quá trình thực hiện, nhóm em đã áp dụng các công nghệ tiên tiến như RESTful API, Redux, JWT Authentication, HTTPS, CORS Policy nhằm nâng cao tính bảo mật, tối ưu hóa hiệu suất và cải thiện trải nghiệm người dùng.

Tuy nhiên, bên cạnh những kết quả đạt được, hệ thống vẫn còn một số hạn chế nhất định như chưa hỗ trợ đa ngôn ngữ, chưa có tính năng AI để gợi ý sản phẩm thông minh, chưa tích hợp chatbot để hỗ trợ khách hàng. Những hạn chế này sẽ là động lực để nhóm tiếp tục nghiên cứu và phát triển trong các giai đoạn tiếp theo.

## Hướng phát triển

Dựa trên kết quả hiện tại, hệ thống có thể được mở rộng theo một số hướng phát triển sau:

* Cải thiện trải nghiệm người dùng (UI/UX): Tối ưu giao diện trên các thiết bị di động, nâng cấp giao diện hiện đại hơn.
* Tích hợp thêm phương thức thanh toán: Mở rộng hỗ trợ PayPal, Stripe bên cạnh VNPAY và MoMo.
* Nâng cao bảo mật: Áp dụng các kỹ thuật bảo mật nâng cao như OAuth2, Xác thực hai yếu tố (2FA), CAPTCHA để chống tấn công brute-force.
* Hệ thống gợi ý sản phẩm thông minh: Sử dụng Machine Learning để đề xuất sản phẩm phù hợp dựa trên hành vi người dùng.
* Tích hợp chatbot hỗ trợ khách hàng: Áp dụng AI Chatbot để tư vấn, giải đáp thắc mắc của khách hàng tự động.
* Phát triển phiên bản mobile app: Xây dựng ứng dụng trên React Native hoặc Flutter để hỗ trợ người dùng di động.
* Tối ưu hiệu suất và khả năng mở rộng: Sử dụng Redis cache, tối ưu database với indexing để cải thiện tốc độ xử lý và mở rộng hệ thống khi có nhiều người dùng.

# TÀI LIỆU THAM KHẢO

Tiếng Việt

<https://codelearn.io/sharing/tim-hieu-ve-mo-hinh-client-server>

<https://viblo.asia/p/tim-hieu-ve-restful-api-OeVKBNYyKkW>

<https://viblo.asia/p/axios-la-gi-xay-dung-ung-dung-khai-thac-tin-su-dung-vuejs-va-axios-bWrZn76rlxw>

<https://viblo.asia/p/chi-tiet-ve-laravel-la-gi-mot-framework-pho-bien-hien-nay-38X4EgDzLN2>

<https://viblo.asia/p/mysql-co-ban-924lJW8X5PM>

<https://200lab.io/blog/jwt-la-gi>

<https://cuuduongthancong.com/atc/33/mo-hi%CC%80nh-du%CC%83-lieu-quan-he--(relational-data-model)---ths-thai-bao-tran---dh-cong-nghe-thong-tin>

<https://viblo.asia/p/cors-la-gi-Qbq5Q0j3lD8>

Tiếng Anh

<https://legacy.reactjs.org/>

<https://redux.js.org/>

<https://vi.wikipedia.org/wiki/HTTPS>