

**TRƯỜNG ĐẠI HỌC TRÀ VINH**  
**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**



**ISO 9001:2015**

**NGUYỄN HỮU LUÂN**

**TÌM HIỂU API VÀ XÂY DỰNG ỨNG DỤNG**  
**BẢN DỤNG CỤ THỂ DỤC THỂ THAO**  
**TRÊN THIẾT BỊ DI ĐỘNG**

**KHÓA LUẬN TỐT NGHIỆP**  
**NGÀNH CÔNG NGHỆ THÔNG TIN**

**VĨNH LONG, NĂM 2025**

TRƯỜNG ĐẠI HỌC TRÀ VINH  
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

TÌM HIỂU API VÀ XÂY DỰNG ỨNG DỤNG  
BẢN DỤNG CỤ THỂ DỤC THỂ THAO  
TRÊN THIẾT BỊ DI ĐỘNG

KHÓA LUẬN TỐT NGHIỆP  
NGÀNH CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn : ThS Trịnh Quốc Việt

Sinh viên thực hiện: Nguyễn Hữu Luân

Mã số sinh viên: 117521003

Lớp: DA21TTC

VĨNH LONG, NĂM 2025

## LỜI CAM ĐOAN

Tôi xin cam đoan rằng khóa luận tốt nghiệp với đề tài “Tìm hiểu API và xây dựng ứng dụng bán dụng cụ thể dục thể thao trên thiết bị di động” là công trình nghiên cứu do chính tôi thực hiện dưới sự hướng dẫn của thầy Trịnh Quốc Việt. Tất cả các số liệu, kết quả và nội dung trình bày trong khóa luận này là trung thực và là sản phẩm của quá trình lao động học thuật nghiêm túc của cá nhân tôi. Mọi tài liệu tham khảo đều đã được trích dẫn rõ ràng và đầy đủ theo quy định. Tôi xin chịu hoàn toàn trách nhiệm về tính xác thực và nguyên bản của công trình nghiên cứu này.

*Vĩnh Long, ngày    tháng    năm 2025*

**Sinh viên thực hiện**

(Ký và ghi rõ họ tên)

**Nguyễn Hữu Luân**

## **LỜI CẢM ƠN**

Tôi xin chân thành gửi lời cảm ơn đến quý thầy cô Trường Kỹ thuật và Công nghệ nói chung và các thầy cô Khoa Công nghệ Thông tin nói riêng đã tạo điều kiện cho tôi có cơ hội thực hành, tiếp xúc để tôi có thể tránh được những vướng mắc và bỏ ngỡ trong môi trường công việc thời gian tới.

Tôi xin chân thành cảm ơn thầy Trịnh Quốc Việt, nhờ sự giúp đỡ tận tình và những chỉ bảo của thầy từ lúc bắt đầu cho tới lúc kết thúc đồ án mà tôi đã hoàn thành đúng thời hạn quy định và tích lũy được cho mình một lượng nền tảng kiến thức quý báu. Trong quá trình thực hiện đồ án còn những thiếu sót nên tôi mong nhận được đóng góp ý của quý thầy, cô để tôi có thể học tập thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn trong tương lai.

Tôi xin chân thành cảm ơn.

# MỤC LỤC

<b>LỜI CAM ĐOAN</b> .....	i
<b>LỜI CẢM ƠN</b> .....	ii
<b>MỤC LỤC</b> .....	iii
<b>DANH MỤC CHỮ VIẾT TẮT</b> .....	viii
<b>DANH MỤC BẢNG BIỂU</b> .....	ix
<b>DANH MỤC HÌNH ẢNH</b> .....	x
<b>TÓM TẮT</b> .....	xi
<b>1. CHƯƠNG 1 MỞ ĐẦU</b> .....	1
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu đề tài.....	2
1.3. Nội dung đề tài .....	2
1.4. Đối tượng và phạm vi nghiên cứu.....	2
1.5. Phương pháp nghiên cứu.....	3
1.6. Đóng góp của đề án.....	4
<b>2. CHƯƠNG 2 CƠ SỞ LÝ THUYẾT</b> .....	6
2.1. Công nghệ áp dụng vào ứng dụng.....	6
2.1.1. NodeJS .....	6
2.1.2. ExpressJS .....	8
2.1.3. API .....	10
2.1.4. RESTful API.....	12
2.1.5. Flutter .....	14
2.1.6. Ngôn ngữ Dart .....	18
2.1.7. MongoDB .....	19
2.2. Công nghệ phát triển .....	23
<b>3. CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU</b> .....	25
3.1. Mô tả bài toán.....	25
3.2. Phân tích chức năng .....	26
3.2.1. Quản trị viên .....	26
3.2.2. Người dùng .....	27
3.2.3. Yêu cầu phi chức năng.....	27
3.3. Sơ đồ use case .....	28

3.3.1. Sơ đồ use case tổng quát .....	28
3.3.2. Sơ đồ Use Case tìm kiếm sản phẩm.....	28
3.3.3. Sơ đồ Use Case quản lý giỏ hàng .....	29
3.3.4. Sơ đồ Use Case đặt hàng .....	29
3.3.5. Sơ đồ Use Case đánh giá sản phẩm .....	30
3.3.6. Sơ đồ Use Case quản lý đơn hàng .....	31
3.3.7. Sơ đồ Use Case quản lý tin tức .....	31
3.3.8. Sơ đồ Use Case quản lý sản phẩm .....	32
3.3.9. Sơ đồ Use Case quản lý thống kê .....	32
3.4. Sơ đồ hệ thống.....	33
3.5. Thiết kế dữ liệu .....	33
3.5.1. Mô hình quan niệm dữ liệu .....	33
3.5.2. Mô hình dữ liệu mức logic.....	34
3.5.3. Cấu trúc dữ liệu dưới dạng JSON .....	34
3.5.4. Dữ liệu mẫu.....	36
3.5.5. Các thực thể .....	38
4. CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU .....	40
4.1. Giao diện đăng nhập.....	40
4.2. Giao diện đăng ký .....	41
4.3. Giao diện người dùng.....	42
4.3.1. Giao diện trang chủ.....	42
4.3.2. Giao diện trang sản phẩm .....	43
4.3.3. Giao diện trang chi tiết sản phẩm .....	43
4.3.4. Giao diện trang giỏ hàng.....	44
4.3.5. Giao diện trang thanh toán.....	45
4.3.6. Giao diện trang chi tiết đơn hàng.....	46
4.4. Giao diện quản trị.....	47
4.4.1. Giao diện trang tổng quan.....	47
4.4.2. Giao diện thêm sản phẩm.....	48
4.4.3. Giao diện các đơn hàng.....	50
4.4.4. Giao diện duyệt trạng thái đơn hàng.....	50
5. CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....	52

5.1. Kết luận .....	52
5.1.1. Kết quả và đóng góp của đề án.....	52
5.1.2. Hạn chế .....	52
5.2. Hướng phát triển .....	53
<b>DANH MỤC TÀI LIỆU THAM KHẢO .....</b>	<b>54</b>

**BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP**  
(Dành cho người hướng dẫn)

Ngành: Công nghệ thông tin

Họ và tên người hướng dẫn: Trịnh Quốc Việt .....  
Đơn vị công tác: Khoa Công nghệ Thông tin, Trường Kỹ thuật và Công nghệ .....  
Họ và tên sinh viên: Nguyễn Hữu Luân ..... MSSV: 117521003 .....  
Tên đề tài: Tìm hiểu API và xây dựng ứng dụng bán dụng cụ thể dục thể thao trên thiết bị di động .....

**I. NHẬN XÉT**

1. Tinh thần, thái độ học tập, nghiên cứu của sinh viên:
- .....
- .....
- .....
- .....
- .....
2. Khả năng nghiên cứu khoa học:
- .....
- .....
- .....
- .....
- .....
- .....
3. Hình thức và nội dung của đồ án:
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....



.....

.....

.....

.....

.....

(Ký & ghi rõ họ tên)

**Trịnh Quốc Việt**

## DANH MỤC CHỮ VIẾT TẮT

Từ viết tắt	Ý nghĩa
<b>API</b>	Artificial Programming Interface (Giao diện lập trình ứng dụng)
<b>JWT</b>	Json Web Token
<b>HTTPS</b>	Hypertext Transfer Protocol Secure

## **DANH MỤC BẢNG BIỂU**

Bảng 2.1 Bảng giải thích cấu trúc thư mục quan trọng của Flutter Framework ...	17
Bảng 3.1 Thực thể role (vai trò người dùng).....	38
Bảng 3.2 Thực thể rating (sao đánh giá) .....	38
Bảng 3.3 Thực thể user (người dùng).....	38
Bảng 3.4 Thực thể products (sản phẩm).....	38
Bảng 3.5 Thực thể orders (đơn hàng).....	39
Bảng 3.6 Thực thể categories (danh mục).....	39

## DANH MỤC HÌNH ẢNH

Hình 2.1 Cấu trúc thư mục của Flutter .....	16
Hình 2.2 Mô hình MVC (Model-View-Controller) .....	23
Hình 3.1 Sơ đồ use case tổng quát.....	28
Hình 3.2 Sơ đồ Use Case tìm kiếm sản phẩm .....	28
Hình 3.3 Sơ đồ Use Case quản lý giỏ hàng .....	29
Hình 3.4 Sơ đồ Use Case đặt hàng .....	29
Hình 3.5 Sơ đồ Use Case đánh giá sản phẩm.....	30
Hình 3.6 Sơ đồ Use Case quản lý đơn hàng.....	31
Hình 3.7 Sơ đồ Use Case quản lý tin tức.....	31
Hình 3.8 Sơ đồ Use Case quản lý sản phẩm.....	32
Hình 3.9 Sơ đồ Use Case quản lý thống kê .....	32
Hình 3.10 Sơ đồ hệ thống.....	33
Hình 3.11 Mô hình quan niệm dữ liệu .....	33
Hình 3.12 Mô hình dữ liệu mức logic .....	34
Hình 4.1 Giao diện đăng nhập.....	40
Hình 4.2 Giao diện đăng ký.....	41
Hình 4.3 Giao diện trang chủ .....	42
Hình 4.4 Giao diện trang sản phẩm .....	43
Hình 4.5 Giao diện chi tiết sản phẩm .....	44
Hình 4.6 Giao diện trang giỏ hàng .....	45
Hình 4.7 Giao diện trang thanh toán .....	46
Hình 4.8 Giao diện chi tiết đơn hàng .....	47
Hình 4.9 Giao diện tổng quan.....	48
Hình 4.10 Giao diện thêm sản phẩm .....	49
Hình 4.11 Giao diện các đơn hàng .....	50
Hình 4.12 Giao diện duyệt trạng thái đơn hàng .....	51

## TÓM TẮT

Đề tài “Tìm hiểu API và xây dựng ứng dụng bán dụng cụ thể dục thể thao trên thiết bị di động” nghiên cứu vai trò của API trong phát triển ứng dụng di động và xây dựng một ứng dụng thương mại điện tử chuyên về dụng cụ thể dục thể thao. Mục tiêu là tìm hiểu các loại API (RESTful, GraphQL), cách tích hợp chúng và phát triển ứng dụng với giao diện thân thiện, hiệu quả.

Phương pháp nghiên cứu bao gồm phân tích tài liệu về API, thiết kế hệ thống với module frontend (giao diện tìm kiếm, giỏ hàng, thanh toán) và backend (API quản lý sản phẩm, đơn hàng, xác thực người dùng qua Auth/JWT). Ứng dụng được phát triển trên Android và IOS, tích hợp thanh toán (Apple Pay/Google Pay). Quá trình kiểm thử sử dụng Android Studio, Postman, đảm bảo tính ổn định và bảo mật.

Kết quả là ứng dụng hoàn chỉnh với các tính năng: hiển thị sản phẩm, tìm kiếm, giỏ hàng, thanh toán và quản lý tài khoản. API giúp tách biệt frontend-backend, dễ mở rộng và tích hợp dịch vụ bên thứ ba. Đề củng cố kiến thức về API, mang tính ứng dụng cao, hỗ trợ doanh nghiệp thể thao tiếp cận thị trường di động. Hướng phát triển tương lai có thể tích hợp AI gợi ý sản phẩm hoặc mở rộng sang nền tảng web, đóng góp vào kinh tế số Việt Nam.

# CHƯƠNG 1 MỞ ĐẦU

## 1.1. Lý do chọn đề tài

Việc lựa chọn đề tài “Tìm hiểu API và xây dựng ứng dụng bán dụng cụ thể dục thể thao trên thiết bị di động” xuất phát từ sự kết hợp giữa sở thích cá nhân, định hướng nghề nghiệp và tiềm năng phát triển của lĩnh vực công nghệ di động tại Việt Nam. Là một sinh viên ngành công nghệ thông tin, tôi luôn bị thu hút bởi các dự án phát triển ứng dụng thực tế, đặc biệt là những ứng dụng mang lại giá trị trực tiếp cho người dùng. Đề tài này không chỉ cho phép tôi khám phá sâu hơn về công nghệ API mà còn mang đến cơ hội xây dựng một sản phẩm cụ thể, phù hợp với xu hướng tiêu dùng hiện đại.

Một trong những lý do chính để chọn đề tài này là sự quan tâm cá nhân đến lĩnh vực thể dục thể thao. Tôi nhận thấy rằng, trong những năm gần đây, người dân Việt Nam ngày càng chú trọng đến việc rèn luyện sức khỏe, dẫn đến nhu cầu mua sắm dụng cụ thể thao tăng mạnh. Tuy nhiên, các ứng dụng di động chuyên biệt phục vụ thị trường này tại Việt Nam vẫn còn hạn chế, đặc biệt là những ứng dụng được tối ưu hóa với công nghệ hiện đại như API. Điều này đã thôi thúc tôi phát triển một ứng dụng giúp người dùng dễ dàng tiếp cận các sản phẩm thể thao chất lượng thông qua thiết bị di động.

Bên cạnh đó, API là một lĩnh vực mà tôi muốn nghiên cứu sâu hơn để nâng cao kỹ năng lập trình. Việc tìm hiểu cách API hoạt động, từ việc kết nối dữ liệu đến tích hợp các dịch vụ như thanh toán hay quản lý đơn hàng, sẽ giúp tôi nắm bắt tốt hơn các công nghệ tiên tiến trong phát triển ứng dụng. Đề tài này mang đến cơ hội thực hành thực tế, từ việc thiết kế giao diện đến triển khai các chức năng phức tạp, qua đó chuẩn bị hành trang vững chắc cho con đường sự nghiệp sau này.

Cuối cùng, tôi mong muốn tạo ra một sản phẩm không chỉ đáp ứng yêu cầu học thuật mà còn có tiềm năng ứng dụng thực tiễn, có thể trở thành nền tảng cho các dự án khởi nghiệp hoặc cải tiến trong tương lai. Với sự kết hợp giữa tính thực tiễn, đam mê cá nhân và cơ hội học hỏi, tôi tin rằng đề tài này sẽ là một bước đi ý nghĩa trong hành trình học tập và phát triển của mình.

## 1.2. Mục tiêu đề tài

Nghiên cứu và nắm vững khái niệm, đặc điểm và cách thức hoạt động của API trong phát triển ứng dụng di động.

Xây dựng ứng dụng di động bán dụng cụ thể dục thể thao với các chức năng cơ bản như hiển thị sản phẩm, quản lý giỏ hàng và thanh toán trực tuyến.

Tích hợp API để đảm bảo ứng dụng hoạt động hiệu quả, linh hoạt và dễ dàng mở rộng.

Tạo ra một sản phẩm thực tiễn, đáp ứng nhu cầu người dùng và củng cố kỹ năng lập trình di động.

## 1.3. Nội dung đề tài

Tìm hiểu khái niệm, vai trò và cách tích hợp API trong phát triển ứng dụng di động.

Phân tích yêu cầu và thiết kế các chức năng cho ứng dụng bán dụng cụ thể dục thể thao.

Phát triển ứng dụng với giao diện thân thiện, tích hợp API và cơ sở dữ liệu hiệu quả.

Kiểm thử, đánh giá ứng dụng và đề xuất hướng cải tiến cho sản phẩm.

## 1.4. Đối tượng và phạm vi nghiên cứu

### **Đối tượng nghiên cứu:**

Công nghệ API và cách ứng dụng trong phát triển ứng dụng di động.

Nhu cầu thị trường và hành vi người dùng đối với dụng cụ thể dục thể thao.

Các chức năng cốt lõi của ứng dụng bán hàng trên thiết bị di động.

Hiệu suất và trải nghiệm người dùng của ứng dụng được phát triển.

Các đối tượng như Model, View, Controller, Route... trong mô hình MVC.

Các thành phần chính của Flutter Framework.

Các công nghệ, ngôn ngữ lập trình có sử dụng để xây dựng ứng dụng di động

### **Phạm vi nghiên cứu:**

Phạm vi nghiên cứu tập trung vào tìm hiểu các loại API phổ biến và ứng dụng trong phát triển ứng dụng di động thương mại.

Nghiên cứu và ứng dụng Flutter Framework để xây dựng một ứng dụng bán dụng cụ thể dục thể thao.

Nghiên cứu thị trường dụng cụ thể dục thể thao tại Việt Nam, tập trung vào nhu cầu người dùng đô thị.

Kiểm thử ứng dụng trong môi trường mô phỏng và trên thiết bị thực tế với quy mô người dùng giới hạn.

### **1.5. Phương pháp nghiên cứu**

**Phương pháp lý thuyết:** Thu thập tài liệu về khái niệm API, quy trình phát triển ứng dụng di động, các công nghệ lập trình và tích hợp API, nguồn gốc lập trình và các công cụ, dịch vụ nhằm tăng lưu trữ dữ liệu thực hiện.

#### **Phương pháp thực nghiệm:**

##### *Phân tích và thiết kế:*

- Tiến hành khảo sát nhu cầu của người dùng mục tiêu (người tập thể dục, phòng gym, huấn luyện viên) thông qua bảng câu hỏi và phỏng vấn để xác định các chức năng cần thiết như tìm kiếm sản phẩm, giỏ hàng, thanh toán trực tuyến và đánh giá sản phẩm.

- Nghiên cứu thị trường dụng cụ thể dục thể thao để xác định các xu hướng phổ biến, sản phẩm bán chạy và các tính năng cạnh tranh của các ứng dụng tương tự.

##### *Mô hình hóa hệ thống:*

- Sử dụng biểu đồ UML (Use Case Diagram, Class Diagram) để mô tả các chức năng, cấu trúc và luồng hoạt động của ứng dụng.

- Thiết kế cơ sở dữ liệu bằng biểu đồ ERD (Entity-Relationship Diagram) để xác định các thực thể chính như User, Product, Order, Payment và các mối quan hệ giữa chúng.



- Lựa chọn mô hình kiến trúc phù hợp, chẳng hạn như kiến trúc client-server, với ứng dụng di động đóng vai trò client và API đóng vai trò trung gian kết nối với cơ sở dữ liệu.

#### *Lập trình và tích hợp:*

- Sử dụng Flutter làm nền tảng phát triển ứng dụng đa nền tảng để đảm bảo tương thích với cả IOS và Android, giảm thời gian và chi phí phát triển.

- Sử dụng Node.js để xây dựng backend và API RESTful, hỗ trợ các chức năng như quản lý sản phẩm, xử lý đơn hàng và xác thực người dùng.

- Tích hợp các thư viện và công cụ hỗ trợ như Provider (Flutter) để quản lý trạng thái ứng dụng, các thư viện giao diện như Flutter Widgets để tăng tốc phát triển UI.

#### *Kiểm thử:*

- Kiểm tra sự tương tác giữa các module của ứng dụng và API, đảm bảo dữ liệu được truyền tải chính xác và các chức năng hoạt động liền mạch.

- Mô phỏng các kịch bản thực tế như người dùng thêm sản phẩm vào giỏ hàng, thực hiện thanh toán hoặc đăng nhập thất bại.

#### *Triển khai:*

- Sử dụng các nền tảng như TestFlight (IOS) hoặc Firebase App Distribution (Android) để triển khai ứng dụng cho một nhóm người dùng thử nghiệm.

- Thu thập phản hồi từ người dùng thử nghiệm về tính năng, giao diện và lỗi gặp phải thông qua các biểu mẫu phản hồi hoặc công cụ như Google Forms.

### **1.6. Đóng góp của đồ án**

Đồ án “Tìm hiểu API và xây dựng ứng dụng bán dụng cụ thể dục thể thao trên thiết bị di động” mang lại những đóng góp thiết thực cả về mặt lý thuyết và thực tiễn. Về lý thuyết, đồ án cung cấp cái nhìn sâu sắc về công nghệ API, vai trò của nó trong phát triển ứng dụng di động, cũng như cách tích hợp hiệu quả vào các hệ thống thương mại điện tử. Về thực tiễn, đồ án tạo ra một ứng dụng di động hoàn chỉnh, đáp ứng nhu cầu mua sắm dụng cụ thể dục thể thao của người dùng đô thị tại Việt Nam, với các chức năng như quản lý sản phẩm, giỏ hàng và thanh toán trực tuyến. Sản phẩm không

chỉ góp phần thúc đẩy xu hướng thương mại điện tử trong lĩnh vực thể thao mà còn là nền tảng để phát triển thêm các tính năng nâng cao trong tương lai. Đồng thời, đồ án giúp tác giả củng cố kiến thức lập trình, kỹ năng phân tích và thiết kế ứng dụng, tạo tiền đề cho các dự án công nghệ hoặc khởi nghiệp sau này.

## CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

### 2.1. Công nghệ áp dụng vào ứng dụng

#### 2.1.1. NodeJS

##### 2.1.1.1. Giới thiệu về NodeJS

NodeJS là một môi trường thực thi đa nền tảng và mã nguồn mở, cho phép phát triển các ứng dụng nhanh chóng và có khả năng mở rộng. Nó được xây dựng trên nền tảng “V8 JavaScript Engine” của Google Chrome. Cùng với đó là cấu trúc I/O non-block và mô hình event-driven (hướng sự kiện) giúp tăng hiệu suất và xử lý các ứng dụng thời gian thực một cách hiệu quả [8].

##### 2.1.1.2. Ứng dụng của NodeJS

**Ứng dụng thời gian thực:** Node.js phù hợp cho các app chat, thông báo đẩy nhờ hỗ trợ WebSocket và kiến trúc event-driven.

**API RESTful & GraphQL:** Thường dùng để xây dựng API nhanh, hiệu quả cho web và mobile.

**Single Page Application (SPA):** Hỗ trợ xử lý server tốt, thích hợp cho SPA cần tốc độ và hiệu suất.

**Ứng dụng IoT:** Nhẹ, xử lý nhanh, phù hợp với hệ thống thiết bị kết nối IoT.

**Truyền phát dữ liệu:** Hỗ trợ streaming tốt, dùng cho video, âm nhạc, hoặc xử lý file lớn.

**Thương mại điện tử:** Đáp ứng nhiều yêu cầu đồng thời, thích hợp cho web bán hàng.

**Máy chủ proxy:** Làm trung gian giữa client và các server/API khác hiệu quả.

**Ứng dụng đa nền tảng:** Kết hợp với Electron để tạo app desktop chạy trên nhiều hệ điều hành.

##### 2.1.1.3. Nguyên lý hoạt động của NodeJS

Node.js vận hành theo mô hình event-driven (hướng sự kiện) và sử dụng event loop (vòng lặp sự kiện) để xử lý các tác vụ bất đồng bộ một cách hiệu quả. Thay vì tạo nhiều luồng xử lý như mô hình đa luồng truyền thống (multi-threaded), Node.js chỉ

dùng một luồng đơn (single-threaded) để xử lý mọi yêu cầu – nhờ vào cơ chế non-blocking I/O.

Khi một tác vụ I/O như đọc file, truy vấn cơ sở dữ liệu hay gọi API được thực hiện, Node.js không chờ tác vụ hoàn tất. Thay vào đó, nó đăng ký callback rồi tiếp tục xử lý các yêu cầu khác. Khi tác vụ hoàn thành, kết quả được đưa vào event queue và chờ event loop gọi lại callback tương ứng khi call stack trống.

Node.js được xây dựng trên V8 Engine của Google Chrome – giúp biên dịch JavaScript trực tiếp thành mã máy, tăng tốc độ thực thi đáng kể.

*Cấu trúc hoạt động của Node.js gồm các thành phần chính:*

**Call Stack:** Nơi lưu trữ các lệnh gọi hàm đang thực thi.

**Event Queue:** Hàng đợi chứa các sự kiện chờ được xử lý.

**Event Loop:** Cơ chế liên tục kiểm tra và chuyển callback vào call stack khi sẵn sàng.

**Thread Pool (libuv):** Các tác vụ I/O nặng được chuyển xuống thread pool để xử lý song song và phản hồi lại qua callback.

Với mô hình này, Node.js đặc biệt phù hợp cho các ứng dụng hiệu suất cao, xử lý đồng thời lớn như: ứng dụng chat thời gian thực, streaming, hay API server hiện đại.

#### *2.1.1.4. Ưu nhược điểm của NodeJS*

**Ưu điểm:**

- Hiệu suất cao
- Xử lý thời gian thực tốt
- Sử dụng cùng một ngôn ngữ
- Hệ sinh thái phong phú
- Dễ dàng mở rộng

**Nhược điểm:**

- Hiệu suất kém với các tác vụ CPU nặng
- Không phù hợp với ứng dụng đơn luồng phức tạp

- Quản lý callback phức tạp
- Không có tính nhất quán
- Bảo mật

### 2.1.2. ExpressJS

#### 2.1.2.1. Khái niệm về ExpressJS

Express.js là một framework đơn giản được xây dựng trên nền tảng Node.js, ra đời với mục đích làm cho việc phát triển các ứng dụng web và API trở nên đơn giản, hiệu quả và dễ bảo trì hơn. Bằng cách cung cấp các công cụ tiện lợi để xử lý yêu cầu HTTP, quản lý định tuyến (routing), và thêm các chức năng khác vào ứng dụng thông qua middleware [10].

Express.js mang lại một cấu trúc rõ ràng cho việc xử lý yêu cầu và phản hồi, giúp lập trình viên dễ dàng kiểm soát luồng dữ liệu qua từng tầng logic. Bên cạnh đó, Express hỗ trợ mở rộng linh hoạt thông qua việc tích hợp các middleware bên ngoài – cho phép bổ sung các chức năng như xác thực, xử lý lỗi, ghi log, upload file, v.v.

Nhờ sự đơn giản, dễ tiếp cận và khả năng mở rộng cao, Express.js đã trở thành lựa chọn hàng đầu trong việc xây dựng các hệ thống web hiện đại – từ ứng dụng nhỏ gọn đến các kiến trúc RESTful API quy mô lớn.

#### 2.1.2.2. Tính năng của ExpressJS

**Khả năng định tuyến rõ ràng:** Express.js cung cấp hệ thống định tuyến URL hiệu quả, cho phép xây dựng cấu trúc ứng dụng rõ ràng và dễ bảo trì. Framework này hỗ trợ đầy đủ các phương thức HTTP như GET, POST, PUT và DELETE, giúp xử lý linh hoạt các yêu cầu từ phía client.

**Hỗ trợ xây dựng API hiệu quả:** Express.js được thiết kế tối ưu cho việc phát triển các API theo chuẩn RESTful hoặc GraphQL. Nhờ cú pháp đơn giản và dễ hiểu, việc tạo và quản lý các endpoint trở nên nhanh chóng và thuận tiện.

**Khả năng tích hợp cao:** Express.js hoạt động liền mạch với nền tảng Node.js và dễ dàng tích hợp với cơ sở dữ liệu cũng như các thư viện bên ngoài thông qua hệ sinh thái npm phong phú. Điều này giúp mở rộng chức năng ứng dụng một cách linh hoạt và hiệu quả.

### *2.1.2.3. Ưu nhược điểm của ExpressJS*

#### ***Ưu điểm:***

- Nhẹ và linh hoạt
- Dễ học và sử dụng
- Tốc độ phát triển nhanh
- Hiệu suất cao
- Hỗ trợ ứng dụng đa dạng

#### ***Nhược điểm:***

- Không phù hợp với dự án lớn
- Quá phụ thuộc vào middleware
- Cần viết nhiều mã
- Bảo mật không tích hợp sẵn
- Quản lý lỗi không hiệu quả

### *2.1.2.4. Nguyên lý hoạt động của ExpressJS*

ExpressJS là một framework ứng dụng web phổ biến hoạt động như một lớp trung gian giữa máy chủ NodeJS và các yêu cầu HTTP từ phía client. Khi có một yêu cầu được gửi đến, Express đảm nhận vai trò tiếp nhận, phân tích và điều phối luồng dữ liệu thông qua hệ thống một cách có tổ chức. Cụ thể, Express tiến hành phân tích phương thức HTTP, địa chỉ URL, tiêu đề và nội dung của yêu cầu để xác định tuyến đường phù hợp thông qua hệ thống định tuyến. Sau đó, hệ thống sẽ thực thi tuần tự các middleware liên quan, bao gồm cả middleware mặc định và các middleware do người dùng định nghĩa. Các middleware này có thể can thiệp, xử lý, kiểm tra hoặc kết thúc yêu cầu tùy vào chức năng được thiết lập. Sau quá trình xử lý, Express sẽ gửi phản hồi tương ứng về phía client.

Trong đó, middleware giữ vai trò then chốt trong cơ chế hoạt động của Express. Chúng tạo nên một chuỗi xử lý tuyến tính, cho phép dễ dàng mở rộng chức năng như xác thực người dùng, ghi log, kiểm tra dữ liệu đầu vào hoặc xử lý lỗi. Cơ chế này giúp

Express trở thành một công cụ linh hoạt, thích hợp cho việc xây dựng các ứng dụng web và API hiện đại với khả năng tùy biến cao.

### 2.1.3. API

#### 2.1.3.1. Khái niệm API

API (Application Programming Interface – Giao diện lập trình ứng dụng) là tập hợp các phương thức và giao thức cho phép các phần mềm hoặc hệ thống khác nhau có thể giao tiếp và trao đổi dữ liệu với nhau. API đóng vai trò như một cầu nối trung gian giữa các ứng dụng, giúp lập trình viên truy cập và sử dụng các chức năng có sẵn từ thư viện, nền tảng hoặc hệ thống bên ngoài mà không cần phải xây dựng lại từ đầu. Thông qua API, các ứng dụng có thể gửi yêu cầu, nhận phản hồi và thực hiện các thao tác như truy vấn cơ sở dữ liệu, xử lý thông tin, hoặc tích hợp với các dịch vụ thứ ba một cách linh hoạt và hiệu quả.

#### 2.1.3.2. Ứng dụng API

**Web API** là dạng API được sử dụng phổ biến trong các hệ thống web. Nó cho phép các ứng dụng phía client như trình duyệt hoặc ứng dụng di động kết nối, truy xuất hoặc cập nhật dữ liệu trên máy chủ thông qua giao thức HTTP. Phần lớn các website hiện đại đều tích hợp Web API để thực hiện các chức năng như đăng nhập, lấy dữ liệu sản phẩm, hoặc gửi biểu mẫu. Ví dụ điển hình là tính năng đăng nhập thông qua Google, Facebook, Twitter hay GitHub – các thao tác này thực chất là việc gọi đến Web API do các nền tảng đó cung cấp.

**API hệ điều hành** là tập hợp các hàm, phương thức và quy ước kết nối được hệ điều hành như Windows hoặc Linux công bố, nhằm hỗ trợ các nhà phát triển xây dựng phần mềm tương tác trực tiếp với tài nguyên hệ thống. Các API này có thể bao gồm thao tác với tập tin, bộ nhớ, quá trình (process), giao tiếp mạng và nhiều chức năng lõi khác. Thông qua tài liệu đặc tả API của hệ điều hành, lập trình viên có thể xây dựng các ứng dụng mạnh mẽ và tối ưu.

**API của thư viện hoặc framework** mô tả cách thức sử dụng các chức năng được đóng gói sẵn bởi thư viện phần mềm hoặc nền tảng phát triển. API ở đây quy định rõ các phương thức, tham số, giá trị trả về và cách thức hoạt động, giúp lập trình viên dễ dàng tích hợp và mở rộng. Một điểm mạnh của API là tính trừu tượng và khả năng

tương tác đa ngôn ngữ – ví dụ, một ứng dụng viết bằng PHP có thể gọi đến thư viện tạo file PDF được viết bằng C++, miễn là thư viện đó cung cấp API phù hợp.

#### 2.1.3.3. Phân loại API

**Open API (Public API):** Đây là loại API được cung cấp công khai cho mọi lập trình viên hoặc tổ chức bên ngoài sử dụng mà không bị giới hạn nghiêm ngặt. Chúng thường được sử dụng để tích hợp vào các ứng dụng bên thứ ba nhằm mở rộng tính năng. Ví dụ tiêu biểu gồm API dự báo thời tiết, API bản đồ của Google Maps hoặc API tìm kiếm từ YouTube.

**Internal API (Private API):** Là API chỉ được sử dụng trong phạm vi nội bộ của một tổ chức hoặc hệ thống. Nó cho phép các thành phần phần mềm trong cùng một kiến trúc trao đổi dữ liệu, phối hợp xử lý mà không để lộ ra bên ngoài. Loại API này thường giúp tăng cường bảo mật và kiểm soát truy cập dữ liệu nội bộ.

**Partner API:** API dành riêng cho các đối tác chiến lược hoặc khách hàng có thỏa thuận hợp tác với tổ chức cung cấp API. Việc truy cập Partner API thường yêu cầu cơ chế xác thực mạnh mẽ và quyền hạn truy cập được giới hạn theo vai trò hoặc hợp đồng đã ký kết. Đây là cách phổ biến để chia sẻ dữ liệu có chọn lọc giữa các tổ chức.

**Composite API:** Là loại API cho phép gộp nhiều thao tác API khác nhau vào một lần gọi duy nhất. Điều này giúp giảm số lượng request gửi đến máy chủ, đồng thời nâng cao hiệu suất hệ thống, đặc biệt hữu ích trong các ứng dụng có nhiều luồng xử lý dữ liệu liên quan cần thực hiện cùng lúc.

#### 2.1.3.4. Ưu nhược điểm của API

##### ***Ưu điểm:***

*Tăng khả năng tái sử dụng mã nguồn:*

Các chức năng được đóng gói dưới dạng API có thể dùng lại nhiều lần ở các dự án khác nhau, giúp tiết kiệm thời gian và công sức lập trình.

*Giảm chi phí phát triển và bảo trì:*

Thay vì xây dựng từ đầu, lập trình viên có thể tận dụng các API sẵn có (nội bộ hoặc bên ngoài), giúp rút ngắn thời gian phát triển và giảm chi phí vận hành hệ thống.



#### *Tích hợp hệ thống nhanh chóng:*

API cho phép kết nối và trao đổi dữ liệu giữa các ứng dụng, dịch vụ hoặc hệ thống khác nhau một cách linh hoạt, đặc biệt hữu ích trong các hệ thống đa nền tảng.

#### *Tăng khả năng mở rộng:*

Việc thiết kế hệ thống dưới dạng các thành phần kết nối qua API giúp dễ dàng mở rộng hoặc nâng cấp mà không ảnh hưởng đến toàn bộ hệ thống.

#### *Hỗ trợ phát triển ứng dụng hiện đại:*

API là nền tảng để xây dựng các ứng dụng web, mobile, hoặc IoT, đồng thời hỗ trợ các giao thức phổ biến như REST, GraphQL, gRPC,...

#### *Nhược điểm:*

##### *Yêu cầu đảm bảo bảo mật và phân quyền hợp lý:*

Việc mở API ra bên ngoài có thể khiến hệ thống dễ bị tấn công nếu không có các cơ chế xác thực, mã hóa và phân quyền người dùng phù hợp.

##### *Phụ thuộc vào bên thứ ba:*

Nếu sử dụng API từ các dịch vụ ngoài như Google, Facebook, các thay đổi về chính sách hoặc phiên bản API có thể ảnh hưởng đến hoạt động hệ thống của bạn.

##### *Khó khăn trong kiểm soát phiên bản:*

Khi API được cập nhật hoặc thay đổi, cần đảm bảo tính tương thích ngược với các hệ thống đang sử dụng trước đó, nếu không sẽ gây lỗi.

##### *Phải duy trì tài liệu kỹ lưỡng:*

API cần có tài liệu chi tiết để các lập trình viên khác có thể hiểu và sử dụng đúng cách. Việc thiếu tài liệu hoặc tài liệu lỗi thời dễ gây nhầm lẫn và sai sót.

##### *Hiệu suất có thể bị ảnh hưởng:*

Giao tiếp qua API thường phải thông qua mạng, có thể làm giảm hiệu suất nếu không được tối ưu tốt (như gọi nhiều request không cần thiết, không dùng cache,...).

## **2.1.4. RESTful API**

### *2.1.4.1. Khái niệm RESTful API*

RESTful API (Representational State Transfer API) là một kiểu thiết kế giao diện lập trình ứng dụng (API) tuân theo các nguyên tắc của kiến trúc REST. REST

hoạt động dựa trên giao thức HTTP, sử dụng các phương thức như GET, POST, PUT, DELETE để thực hiện các thao tác với tài nguyên (resource), được định danh bằng URL.

RESTful API có tính chất nhẹ, dễ triển khai, phù hợp với các ứng dụng web và mobile hiện đại. Dữ liệu thường được trả về ở định dạng JSON hoặc XML, giúp dễ dàng trao đổi giữa các hệ thống khác nhau.

#### *2.1.4.2. Các nguyên tắc của RESTful API*

RESTful API được xây dựng dựa trên một tập hợp các nguyên tắc cốt lõi, giúp đảm bảo tính hiệu quả và linh hoạt trong thiết kế.

Client-Server Architecture (Kiến trúc Client-Server)

Statelessness (Không lưu trạng thái)

Cacheable (Có thể lưu cache)

Layered System (Hệ thống phân lớp)

Uniform Interface (Giao diện thống nhất)

Code on Demand (Mã được tải về)

#### *2.1.4.3. Các thành phần chính của RESTful API*

Resource (Tài nguyên): Mỗi tài nguyên có một URL định danh duy nhất.

HTTP Methods:

GET: Lấy dữ liệu.

POST: Tạo mới dữ liệu.

PUT: Cập nhật toàn bộ dữ liệu.

PATCH: Cập nhật một phần.

DELETE: Xoá dữ liệu.

Status Code: Trạng thái phản hồi (200 OK, 404 Not Found, 500 Server Error...).

Headers & Body: Chứa metadata và nội dung dữ liệu gửi/nhận.

#### *2.1.4.4. Lợi ích của RESTful API*

RESTful API nổi bật nhờ sự đơn giản và tuân thủ các tiêu chuẩn phổ biến. Với việc sử dụng các phương thức HTTP cơ bản như GET, POST, PUT và DELETE, REST giúp thực hiện các thao tác CRUD một cách rõ ràng và dễ hiểu.

Một ưu điểm lớn của RESTful API là khả năng tương thích cao – bất kỳ nền tảng nào có thể gửi yêu cầu HTTP, từ trình duyệt web đến ứng dụng di động hay các dịch vụ backend, đều có thể tích hợp dễ dàng.

Kiến trúc REST cũng hỗ trợ khả năng mở rộng tốt, cho phép hệ thống xử lý lượng lớn yêu cầu mà không ảnh hưởng đến hiệu suất của các thành phần khác. Việc tách biệt rõ ràng giữa client và server giúp frontend và backend có thể phát triển độc lập, nâng cao tính linh hoạt trong quản lý và vận hành hệ thống.

Ngoài ra, RESTful API có thể trả dữ liệu ở nhiều định dạng như JSON, XML hay HTML, đáp ứng đa dạng nhu cầu sử dụng. Với khả năng tái sử dụng cao, các API được thiết kế theo chuẩn REST giúp tiết kiệm thời gian và chi phí khi mở rộng hoặc phát triển ứng dụng mới.

#### *2.1.4.5. Những hạn chế*

Mặc dù RESTful API mang lại nhiều lợi ích, nhưng vẫn tồn tại một số hạn chế nhất định. Trước hết, REST không thực sự phù hợp với những hệ thống có mối quan hệ dữ liệu phức tạp, do việc truy xuất các dữ liệu liên kết thường đòi hỏi nhiều lần gọi API, dẫn đến hiệu suất thấp. Bên cạnh đó, REST thiếu khả năng mô tả và truy vấn dữ liệu linh hoạt như GraphQL, khiến việc phát triển và kiểm thử đôi khi gặp khó khăn.

Ngoài ra, do REST không có một quy chuẩn thống nhất về cách đặt tên tài nguyên, các nhóm phát triển có thể áp dụng cách đặt tên khác nhau, gây thiếu nhất quán trong toàn bộ hệ thống.

### **2.1.5. Flutter**

#### *2.1.5.1. Giới thiệu về Flutter*

Flutter là một Mobile Framework của Google dùng để tạo giao diện chất lượng cao trên các hệ điều hành iOS và Android.

Flutter là một mã nguồn mở và hoàn toàn miễn phí.

Flutter là một khung nguồn mở do Google phát triển và hỗ trợ. Các nhà phát triển frontend và fullstack sử dụng Flutter để xây dựng giao diện người dùng (UI) của ứng dụng cho nhiều nền tảng chỉ với một nền mã duy nhất.

Tại thời điểm ra mắt vào năm 2018, Flutter chủ yếu hỗ trợ phát triển ứng dụng di động. Hiện nay, Flutter hỗ trợ phát triển ứng dụng trên sáu nền tảng: iOS, Android, web, Windows, MacOS và Linux.

#### *2.1.5.2. Quá trình phát triển*

Năm 2015, Flutter Framework được công bố bởi Google, hoạt động với ngôn ngữ Dart, làm nền tảng để phát triển Android.

Năm 2017, phiên bản Alpha đã được phát hành lần đầu tiên. Tại sự kiện I/O cùng năm Google đã quảng bá việc Flutter sẽ phát triển đa nền tảng.

Năm 2018, Flutter 1.0 được phát hành tại sự kiện Flutter Live và có sẵn SDK để các nhà phát triển sử dụng nhằm tạo ứng dụng dễ dàng hơn.

Năm 2019, tại Google I/O, Flutter 1.12 ra đời và Flutter dành cho các nền tảng Windows, macOS, Linux đã được phát hành.

Năm 2021, Flutter cập nhật phiên bản 2.0.6.

Năm 2022, Flutter cập nhật phiên bản 3.0.

Phiên bản mới nhất của Flutter là 3.22.2 tính đến ngày 1/6/2024

#### *2.1.5.3. Các thành phần chính của Flutter Framework*

SDK – Software Development Kit (Bộ công cụ phát triển phần mềm) là các công cụ giúp phát triển ứng dụng, gồm các công cụ để biên dịch mã nguồn dùng cho hai hệ điều hành là Android và iOS.

Framework – UI Library Based on Widgets (Thư viện giao diện người dùng dựa trên widgets) là tập hợp các phần tử giao diện có thể tái sử dụng như Button, Text Input, Slider giúp người dùng có thể sáng tạo giao diện ứng dụng theo nhu cầu cá nhân.

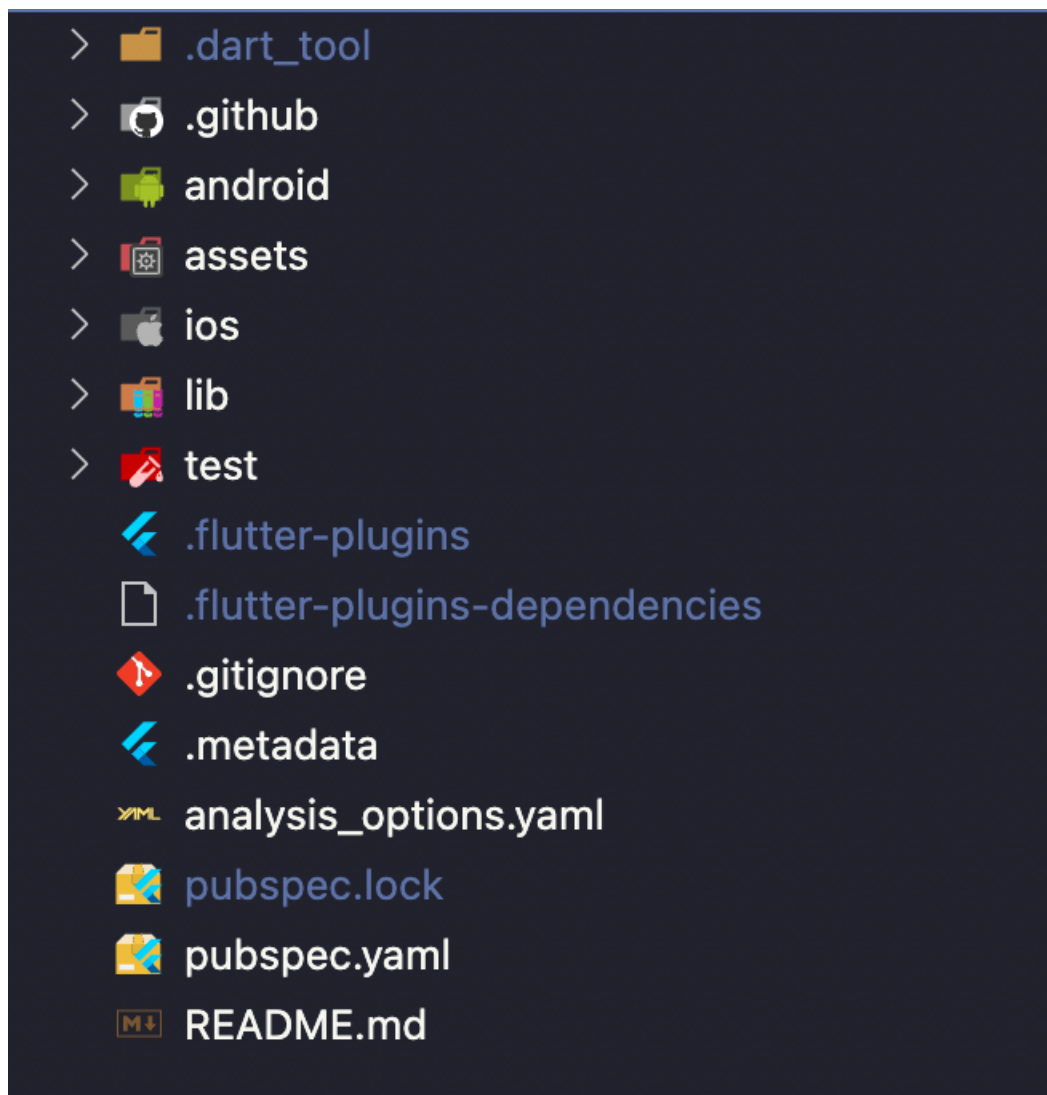
Dart là ngôn ngữ lập trình chính được sử dụng trong Flutter, được tối ưu hóa cho việc phát triển giao diện người dùng. Dart Platform bao gồm trình biên dịch (compiler) và máy ảo (Dart VM) để thực thi mã nguồn. Ngôn ngữ này hỗ trợ cả biên dịch trước (AOT - Ahead of Time) cho hiệu suất cao trên thiết bị di động và biên dịch tức thời (JIT - Just in Time) để tăng tốc phát triển. Dart còn cung cấp các thư viện cốt

lỗi mạnh mẽ, giúp xử lý các tác vụ như quản lý trạng thái, kết nối mạng và lưu trữ dữ liệu, đảm bảo ứng dụng hoạt động mượt mà và hiệu quả.

Flutter Engine là lõi của Flutter Framework, được viết bằng C++ và chịu trách nhiệm xử lý các tác vụ cấp thấp như render giao diện, xử lý sự kiện cảm ứng và quản lý vòng đời ứng dụng. Engine tích hợp Skia, một thư viện đồ họa 2D mạnh mẽ, để vẽ giao diện người dùng độc lập với nền tảng. Ngoài ra, Flutter Engine quản lý các kênh giao tiếp (Platform Channels) để kết nối với các API gốc của Android và iOS, cho phép ứng dụng truy cập các tính năng thiết bị như máy ảnh, GPS, hoặc thông báo đẩy, đảm bảo tính linh hoạt và hiệu suất cao.

#### 2.1.5.4. Cấu trúc thư mục Flutter

Cấu trúc thư mục của Flutter được tổ chức một cách rõ ràng cho việc phát triển ứng dụng theo các nền tảng khác nhau như iOS và Android.



Hình 2.1 Cấu trúc thư mục của Flutter

*Bảng 2.1 Bảng giải thích cấu trúc thư mục quan trọng của Flutter Framework*

STT	Đối tượng	Mô tả
1	android/	Chứa các tệp cấu hình và mã nguồn gốc của ứng dụng Android
2	ios/	Chứa các tệp cấu hình và mã nguồn gốc của ứng dụng iOS.
3	lib/	Thư mục này chứa tất cả các mã nguồn Dart của ứng dụng. File main.dart là tệp chính được chạy đầu tiên khi ứng dụng khởi động.
4	test/	Chứa các tệp kiểm thử cho ứng dụng.
5	web/, windows/, macos/, linux/	Các thư mục này chứa mã nguồn và cấu hình cho các nền tảng tương ứng nếu người lập trình đang phát triển ứng dụng đa nền tảng.
6	.dart_tool/	Thư mục chứa các công cụ Dart cần thiết cho dự án.
7	pubspec.yaml	Tệp này khai báo các dependencies của dự án, các asset, và các thông tin khác về dự án.
8	build.gradle	Tệp cấu hình Gradle dùng để build ứng dụng Android.
9	.vscode/	Thư mục này chứa các tệp cấu hình cho Visual Studio Code.
10	build/	Thư mục này được tạo tự động trong quá trình build ứng dụng và chứa các tệp tạm thời được tạo ra trong quá trình build. Không cần phải chỉnh sửa các tệp trong thư mục này.

#### **2.1.5.5. Đặt điểm nổi bật của Flutter Framework**

Tính năng Host Reload: Tính năng này hoạt động trong milliseconds để hiện thị giao diện tới lập trình viên. Sử dụng các widget có thể tùy chỉnh giúp lập trình viên tạo các giao diện chỉ trong vài phút. Ngoài ra, tính năng này còn hỗ trợ thêm fix bug mà không cần phải thông qua máy ảo, máy android hoặc iOS giúp tiết kiệm thời gian hơn.

Expressive and Flexible UI - Giao diện người dùng linh hoạt và biểu cảm: hỗ trợ nhiều các thành phần để xây dựng giao diện của Flutter vô cùng tiện lợi theo các phong cách như Material Design và Cupertino, hỗ trợ API chuyển động, smooth scrolling...

Native Performance – Hiệu xuất gốc: Các widget của flutter kết hợp các sự khác biệt của các nền tảng ví dụ như scrolling, navigation, icons, font để cung cấp một hiệu năng tốt nhất tới iOS và Android.

#### 2.1.5.6. Nhược điểm

Để sử dụng được Flutter, lập trình viên hoặc các nhà phát triển phải học thêm ngôn ngữ lập trình Dart vì ngôn ngữ lập trình này không phổ biến, do vậy sẽ gặp khó khăn hơn trong việc phát triển các ứng dụng trên thiết bị di động.

Cập nhật lên kho ứng dụng khó khăn hơn vì phải thông qua trung tâm ứng dụng như App Store hoặc Play Store.

#### 2.1.6. Ngôn ngữ Dart

Dart là ngôn ngữ lập trình hướng đối tượng đa mục đích mã nguồn mở, được đặt nền móng và phát triển bởi Google. Lập trình hướng đối tượng thể hiện qua việc hỗ trợ giao diện và lớp. Sự đa dạng của Dart thể hiện thông qua khả năng phát triển ứng dụng web, di động, máy chủ và máy tính để bàn.

##### 2.1.6.1. Lịch sử hình thành và các phiên bản

Trong lĩnh vực lập trình, Dart là một dự án tâm huyết của Lars Bak và Kasper Lund, được phát triển dưới sự quản lý của Google. Dart lần đầu tiên được giới thiệu tại hội nghị GOTO ở Đan Mạch vào ngày 10 tháng 10 năm 2011 và nhanh chóng cho thấy tiềm năng lớn của mình. Một cột mốc quan trọng là việc phát hành phiên bản Dart 1.0 vào ngày 14 tháng 11 năm 2013. Mặc dù ban đầu Dart nhận được những phản hồi trái chiều, nhưng điều này không thể cản trở sự phát triển vượt bậc của nó.

Giai đoạn này cũng đánh dấu sự chuyển đổi từ kế hoạch tích hợp máy ảo Dart vào Chrome sang việc biên dịch mã Dart thành JavaScript. Điều này rất quan trọng đối với phiên bản Dart 2.0, ra mắt vào tháng 8 năm 2018, mang đến nhiều cải tiến về ngôn ngữ, bao gồm cả hệ thống kiểu dữ liệu mới.

Dart có nhiều phiên bản khác nhau. Mới nhất chính là Dart 3.5.2, ra đời vào ngày 28 tháng 8 năm 2024.

##### 2.1.6.2. Đặc điểm nổi bật của ngôn ngữ Dart

Cú pháp Dart rõ ràng và ngắn gọn, công cụ đơn giản nhưng mạnh mẽ. Type-safe giúp xác định sớm các lỗi.

Dart tối ưu hóa việc biên dịch trước thời hạn để có được dự đoán hiệu suất cao và khởi động nhanh trên các thiết bị di động và web.

Dart biên dịch thành mã ARM và x86, để các ứng dụng di động của Dart có thể chạy tự nhiên trên iOS, Android và hơn thế nữa. Đối với các ứng dụng web, chuyển mã từ Dart sang JavaScript.

### **2.1.6.3. Nhược điểm**

Ngôn ngữ lập trình Dart là một ngôn ngữ lập trình khá mới nên hiện tại cộng đồng người sử dụng có quy mô nhỏ, chưa có nhiều tài liệu phục vụ cho việc học tập.

Dart chủ yếu được biết đến và sử dụng trong việc phát triển ứng dụng di động với Flutter. Việc sử dụng Dart ngoài lĩnh vực này vẫn còn hạn chế, làm giảm tính linh hoạt của nó so với các ngôn ngữ đa năng khác.

### **2.1.7. MongoDB**

#### *2.1.7.1. Khái niệm*

MongoDB là một phần mềm mã nguồn mở dùng để quản trị cơ sở dữ liệu NoSQL.

Hiện nay, có nhiều công ty toàn cầu sử dụng MongoDB để lưu trữ lượng dữ liệu “khổng lồ” của họ như Facebook, Nokia, eBay, Adobe, Google,...

#### *2.1.7.2. Công dụng của Mongo DB*

MongoDB giúp các tổ chức lưu trữ lượng lớn dữ liệu trong khi vẫn hoạt động nhanh chóng. Ngoài lưu trữ dữ liệu, MongoDB còn được sử dụng trong các trường hợp sau:

Tích hợp một lượng lớn dữ liệu đa dạng.

Mô tả các cấu trúc dữ liệu phức tạp, biến hoá.

Cung cấp dữ liệu cho các ứng dụng hiệu suất cao.

Hỗ trợ các ứng dụng đám mây lai và đa đám mây..

Hỗ trợ phương pháp phát triển Agile

Thay vì sử dụng các table và row như trong cơ sở dữ liệu quan hệ, vì là cơ sở dữ liệu NoSQL, MongoDB được tạo thành từ collection và document. Document được tạo thành từ các cặp khóa-giá trị (là đơn vị dữ liệu cơ bản của MongoDB). Còn collection, tương đương với table trong SQL, là nơi chứa các bộ document.



### 2.1.7.3. Các thuật ngữ của MongoDB

#### **\_id**

\_id là một trường bắt buộc trong mọi document của MongoDB. \_id được sử dụng để đại diện cho tính duy nhất của một document trong một collection. Trường \_id hoạt động giống như khóa chính (primary key) của document.

\_id là một số thập lục phân 12 byte đảm bảo tính duy nhất của mọi document. Bạn có thể cung cấp \_id trong khi chèn document. Trong 12 byte này:

4 byte đầu tiên đại diện cho thời điểm hiện tại (dựa trên hệ giây của Unix Epoch);

3 byte tiếp theo cho id máy;

2 byte tiếp theo cho process id của máy chủ MongoDB;

3 byte cuối cùng là giá trị gia tăng đơn giản.

#### **Document**

Document là đơn vị lưu trữ dữ liệu cơ bản trong cơ sở dữ liệu MongoDB. Document mang vai trò tương tự như row trong các hệ thống cơ sở dữ liệu quan hệ truyền thống.

Document là một cách để sắp xếp và lưu trữ dữ liệu dưới dạng một tập hợp các cặp field-value. Document trong MongoDB không cần phải có cùng một bộ field hoặc cấu trúc với các document khác trong cùng một collection.

Đồng thời, các field chung trong document của một collection có thể chứa các loại dữ liệu khác nhau.

#### **Collection**

Collection là một tập hợp các document MongoDB. Collection tương tự như table trong hệ thống cơ sở dữ liệu quan hệ. Các collection có tính chất schema less, do đó các document trong cùng một collection có thể có các trường khác nhau.

Thông thường, một collection chứa các document có mục đích tương tự hoặc liên quan với nhau.

#### **Database**

Trong MongoDB, database là một container vật lý chứa tập hợp các collection. Một database có thể chứa 0 collection hoặc nhiều collection.

Một phiên bản máy chủ MongoDB có thể lưu trữ nhiều database và không có giới hạn về số lượng database có thể được lưu trữ trên một phiên bản, nhưng giới hạn ở không gian bộ nhớ ảo có thể được phân bổ bởi hệ điều hành.

#### **2.1.7.4. MongDB hoạt động như thế nào ?**

MongoDB là một máy chủ cơ sở dữ liệu và dữ liệu được lưu trữ trong các cơ sở dữ liệu này. Hay nói cách khác, môi trường MongoDB cung cấp một máy chủ có thể khởi động và sau đó tạo nhiều cơ sở dữ liệu trên đó bằng MongoDB.

Trong máy chủ MongoDB có thể tạo nhiều cơ sở dữ liệu và nhiều collection.

Cách cơ sở dữ liệu MongoDB chứa các collection cũng giống như cách cơ sở dữ liệu MySQL chứa các table.

Bên trong collection, chúng ta có document. Các document này chứa dữ liệu mà người dùng muốn lưu trữ trong cơ sở dữ liệu MongoDB và một collection có thể chứa nhiều document. Đồng thời, với tính chất schema-less (không cần một cấu trúc lưu trữ dữ liệu), document này không nhất thiết phải giống với document khác.

Các document được tạo bằng cách sử dụng các field (trường). Các field là các cặp khóa-giá trị trong document, giống như các column trong cơ sở dữ liệu quan hệ. Giá trị của các field có thể là bất kỳ loại dữ liệu BSON nào như double, string, boolean,...

MongoDB lưu trữ dữ liệu ở định dạng BSON document. Ở đây, BSON là đại diện cho định dạng mã hoá nhị phân của các tài liệu JSON (chữ B trong BSON là viết tắt của Binary). Hay nói cách khác, trong phần backend, máy chủ MongoDB chuyển đổi dữ liệu JSON thành dạng nhị phân, được gọi là BSON, và BSON này có thể được lưu trữ và truy vấn hiệu quả hơn. Kích thước tối đa của BSON document là 16 MB.

Trong MongoDB document, được phép lưu trữ dữ liệu lồng nhau. Việc lồng dữ liệu này cho phép tạo các mối quan hệ phức tạp giữa dữ liệu và lưu trữ chúng trong cùng một document, giúp cho quá trình làm việc và tìm nạp dữ liệu hiệu quả hơn so với SQL.

### 2.1.7.5. Ưu điểm và khuyết điểm của MongoDB

#### *Ưu điểm:*

Không schema: Giống như các cơ sở dữ liệu NoSQL khác, MongoDB không yêu cầu các schema được xác định trước.

MongoDB lưu trữ bất kỳ loại dữ liệu nào: Điều này cho phép người dùng linh hoạt tạo số lượng trường trong document theo nhu cầu, và giúp việc mở rộng cơ sở dữ liệu MongoDB trở nên dễ dàng hơn so với cơ sở dữ liệu quan hệ truyền thống.

Hướng document: Một trong những ưu điểm của việc sử dụng document là các đối tượng này ánh xạ tới các kiểu dữ liệu gốc trong một số ngôn ngữ lập trình. Việc có các document được nhúng cũng làm giảm nhu cầu kết nối cơ sở dữ liệu, điều này có thể làm giảm chi phí.

Khả năng mở rộng: Kiến trúc mở rộng theo chiều ngang của MongoDB giúp bạn tạo ra một ứng dụng có thể xử lý được lưu lượng truy cập tăng đột biến khi doanh nghiệp của bạn phát triển. Ngoài ra, việc phân chia dữ liệu (sharding) cho phép cơ sở dữ liệu phân phối dữ liệu trên một cụm máy. MongoDB cũng hỗ trợ tạo vùng dữ liệu dựa trên shard key.

Hỗ trợ bên thứ ba: MongoDB hỗ trợ một số công cụ lưu trữ và cung cấp API công cụ lưu trữ có thể cắm được (pluggable storage engine API) cho phép các bên thứ ba phát triển công cụ lưu trữ dữ liệu riêng.

Linh hoạt lưu trữ tệp dung lượng lớn: MongoDB phát triển hệ thống tệp riêng GridFS, gần giống với hệ thống tệp phân tán Hadoop. Việc sử dụng hệ thống tệp nhằm để lưu trữ các tệp vượt qua kích thước giới hạn của BSON (16 MB cho mỗi document).

#### *Khuyết điểm:*

Tính liên tục: Với chiến lược chuyển đổi dự phòng tự động, người dùng chỉ có thể thiết lập một node master trong cụm MongoDB. Nếu node master bị lỗi, một node khác sẽ tự động chuyển đổi thành master mới. Quá trình chuyển đổi này đảm bảo tính liên tục, nhưng không diễn ra tức thời mà có thể mất tới một phút.

Giới hạn ghi: Node master duy nhất của MongoDB cũng làm giới hạn lại tốc độ ghi dữ liệu vào cơ sở dữ liệu. Việc ghi dữ liệu phải được ghi trên node master và việc ghi thông tin mới vào cơ sở dữ liệu bị giới hạn bởi khả năng của node master đó.

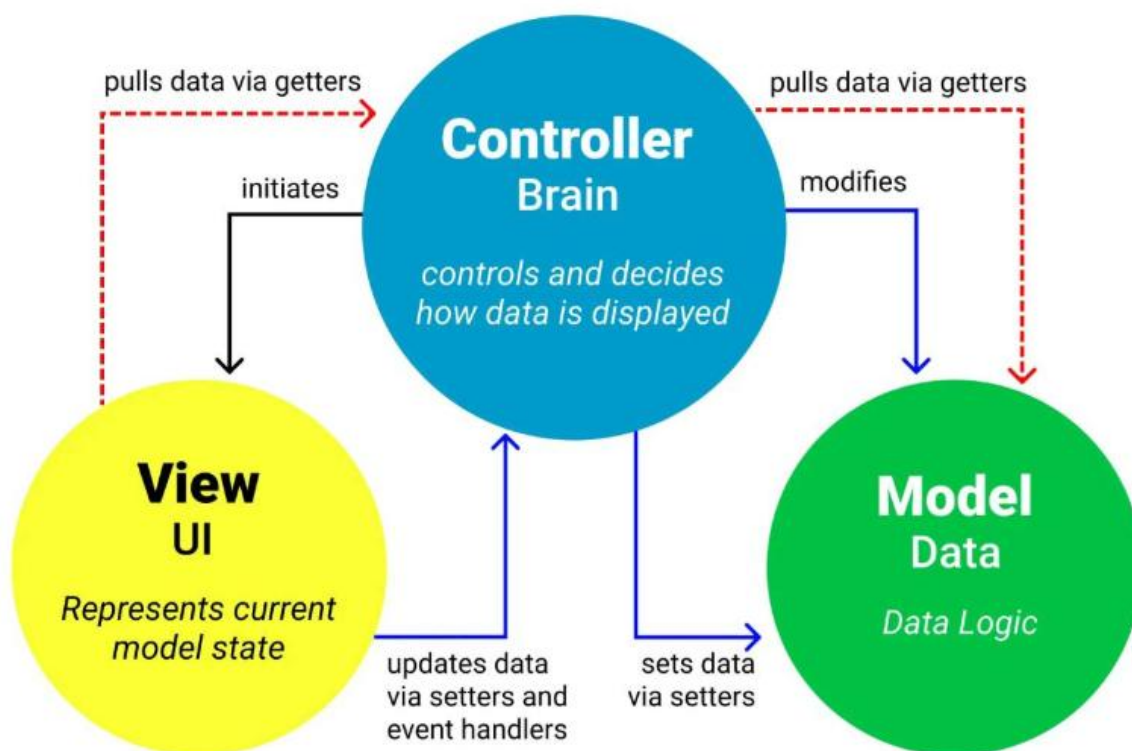
Tính nhất quán của dữ liệu: MongoDB không cung cấp tính toàn vẹn tham chiếu đầy đủ thông qua việc sử dụng các ràng buộc khóa ngoại (foreign-key), điều này có thể ảnh hưởng đến tính nhất quán của dữ liệu.

Bảo mật: Tính năng xác thực người dùng không được mặc định bật trong cơ sở dữ liệu MongoDB. Để bảo mật hệ thống trước các cuộc tấn công của tin tặc, bạn có thể thủ công thiết lập các cài đặt chặn những kết nối lạ và không an toàn.

## 2.2. Công nghệ phát triển

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

Mô hình MVC là một kiến trúc phần mềm phổ biến được sử dụng để phát triển ứng dụng web. Mô hình này giúp tách biệt các phần của ứng dụng, qua đó cải thiện khả năng bảo trì, mở rộng và quản lý mã nguồn. Mô hình MVC bao gồm ba thành phần chính:



Hình 2.2 Mô hình MVC (Model-View-Controller)

#### Model (Mô hình):

Model đại diện cho dữ liệu và logic nghiệp vụ của ứng dụng. Chịu trách nhiệm quản lý dữ liệu, bao gồm việc truy xuất, lưu trữ và xử lý thông tin từ cơ sở dữ liệu.

Model tương tác với cơ sở dữ liệu để thực hiện các thao tác như thêm, sửa, xóa và truy vấn dữ liệu. Khi có thay đổi trong dữ liệu, mô hình sẽ thông báo cho các thành phần khác (thường là View) để cập nhật giao diện người dùng.

### **View (Giao diện):**

View người dùng chịu trách nhiệm hiển thị thông tin cho người dùng. Nhận dữ liệu từ model và trình bày chúng một cách trực quan.

View không chứa logic nghiệp vụ, chỉ hiển thị dữ liệu và nhận các tương tác từ người dùng (như nhấp chuột, nhập liệu). Khi người dùng thực hiện hành động, View sẽ gửi yêu cầu đến Controller để xử lý.

### **Controller (Bộ điều khiển):**

Controller là cầu nối giữa Model và View. Nhận các yêu cầu từ người dùng thông qua View, xử lý các yêu cầu đó (có thể bao gồm việc gọi đến Model để lấy dữ liệu) và cập nhật View tương ứng.

Controller nhận được yêu cầu từ View, sẽ thực hiện các thao tác cần thiết, như truy xuất dữ liệu từ Model, sau đó gửi dữ liệu đó đến View để hiển thị cho người dùng.

### **Lợi ích của mô hình MVC**

Mô hình MVC giúp tách biệt rõ ràng giữa dữ liệu, giao diện và logic nghiệp vụ, từ đó giúp dễ dàng bảo trì và mở rộng ứng dụng. Với việc tách biệt các thành phần, việc kiểm thử từng phần của ứng dụng trở nên dễ dàng hơn. Nhiều lập trình viên có thể làm việc đồng thời trên các phần khác nhau của ứng dụng mà không gây xung đột, ví dụ, một người có thể làm việc trên View trong khi người khác làm việc trên Model.

## CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU

### 3.1. Mô tả bài toán

Trong bối cảnh công nghệ số hóa phát triển mạnh mẽ, thương mại điện tử trên nền tảng di động đang trở thành xu hướng chủ đạo, đặc biệt trong lĩnh vực thể dục thể thao, một ngành hàng ngày càng được quan tâm do nhu cầu nâng cao sức khỏe cộng đồng. Tuy nhiên, tại Việt Nam thị trường dụng cụ thể dục thể thao vẫn tồn tại nhiều hạn chế: các cửa hàng truyền thống gặp khó khăn trong việc tiếp cận khách hàng trực tuyến, thiếu ứng dụng di động chuyên biệt hỗ trợ mua sắm tiện lợi, dẫn đến mất cơ hội cạnh tranh với các nền tảng lớn như Shopee hay Lazada. Người tiêu dùng thường phải tìm kiếm sản phẩm qua nhiều kênh phân tán, gặp vấn đề về thông tin sản phẩm không đầy đủ, quy trình thanh toán phức tạp và thiếu tính cá nhân hóa. Hơn nữa, việc quản lý kho hàng, đơn đặt hàng và dữ liệu khách hàng thủ công khiến doanh nghiệp nhỏ lẻ tốn kém thời gian và dễ mắc lỗi. Bài toán đặt ra là cần xây dựng một ứng dụng di động hỗ trợ bán dụng cụ thể dục thể thao, tích hợp API để kết nối dữ liệu hiệu quả, nhằm nâng cao trải nghiệm người dùng và tối ưu hóa quy trình kinh doanh.

Cụ thể, ứng dụng cần đáp ứng các yêu cầu chính: Giao diện thân thiện trên thiết bị di động, hỗ trợ hiển thị danh mục sản phẩm đa dạng như máy chạy bộ, tạ tay, dụng cụ yoga, với tính năng tìm kiếm, lọc theo giá cả, thương hiệu và đánh giá, quản lý giỏ hàng, thanh toán trực tuyến an toàn qua các cổng như ApplePay và Google Pay, tích hợp quản lý tài khoản người dùng, bao gồm đăng ký, đăng nhập, theo dõi đơn hàng, phía backend cần xử lý dữ liệu thời gian thực, quản lý cơ sở dữ liệu sản phẩm, đơn hàng và người dùng, đồng thời đảm bảo bảo mật thông tin qua các cơ chế xác thực như JWT. Việc sử dụng API là yếu tố cốt lõi để tách biệt frontend và backend, cho phép ứng dụng dễ dàng mở rộng, tích hợp dịch vụ bên thứ ba như lưu trữ hình ảnh trên Cloudinary.

Dự án sử dụng Flutter làm framework phát triển frontend, nhờ khả năng cross-platform (hỗ trợ cả Android và IOS từ một codebase duy nhất), dựa trên ngôn ngữ Dart và hệ thống widget phong phú để tạo giao diện mượt mà, responsive. Flutter giúp giảm thời gian phát triển và chi phí, đồng thời tích hợp dễ dàng với các API qua thư viện như http hoặc Dio. Phía backend, Node.js được chọn nhờ tính linh hoạt, hiệu suất cao trong xử lý yêu cầu đồng thời và hệ sinh thái phong phú với Express.js để xây

dựng RESTful API. Node.js kết hợp với cơ sở dữ liệu MongoDB (qua Mongoose) để lưu trữ dữ liệu không cấu trúc, hỗ trợ các tính năng như upload hình ảnh sản phẩm và xử lý đơn hàng thời gian thực. Việc kết nối giữa Flutter và Node.js qua API đảm bảo dữ liệu được trao đổi an toàn, với các endpoint như GET /products để lấy danh sách sản phẩm, POST /orders để tạo đơn hàng, và PUT /users để cập nhật thông tin người dùng.

Tổng thể, bài toán không chỉ tập trung vào việc xây dựng ứng dụng mà còn nghiên cứu sâu về API để đảm bảo tính khả dụng, bảo mật và hiệu suất. Giải quyết bài toán này sẽ góp phần thúc đẩy thương mại điện tử trong lĩnh vực thể dục thể thao, mang lại lợi ích cho cả doanh nghiệp và người tiêu dùng, đồng thời khẳng định vai trò của công nghệ hiện đại trong kinh tế số. Dự án dự kiến triển khai qua các giai đoạn: phân tích yêu cầu, thiết kế hệ thống, lập trình, kiểm thử và đánh giá, với trọng tâm là tích hợp API để ứng dụng hoạt động ổn định trên thiết bị di động thực tế.

### **3.2. Phân tích chức năng**

#### **3.2.1. Quản trị viên**

Quản trị viên (admin) chịu trách nhiệm quản lý hệ thống và đảm bảo hoạt động của ứng dụng bán dụng cụ thể dục thể thao diễn ra hiệu quả. Các chức năng chính bao gồm:

Quản lý sản phẩm: Thêm, sửa, xóa thông tin sản phẩm (máy chạy bộ, tạ tay, dụng cụ yoga,...), bao gồm tên, mô tả, giá cả, hình ảnh và trạng thái tồn kho.

Quản lý đơn hàng: Xem danh sách đơn hàng, cập nhật trạng thái (chờ xử lý, đang giao, hoàn thành) và xử lý yêu cầu hoàn hàng hoặc hủy đơn.

Quản lý người dùng: Kiểm tra, kích hoạt hoặc vô hiệu hóa tài khoản người dùng, xem lịch sử mua hàng của khách hàng để hỗ trợ chăm sóc.

Quản lý khuyến mãi: Tạo, sửa, xóa các chương trình khuyến mãi (giảm giá, mã ưu đãi) và gửi thông báo.

Báo cáo thống kê: Xem báo cáo doanh thu, sản phẩm bán chạy và hành vi người dùng để hỗ trợ ra quyết định kinh doanh.

### **3.2.2. Người dùng**

Người dùng là khách hàng sử dụng ứng dụng để mua sắm dụng cụ thể dục thể thao. Các chức năng chính bao gồm:

**Tìm kiếm và lọc sản phẩm:** Tìm kiếm sản phẩm theo từ khóa, lọc theo danh mục, giá cả, hoặc thương hiệu.

**Xem chi tiết sản phẩm:** Xem thông tin chi tiết, hình ảnh, đánh giá và nhận xét về sản phẩm.

**Quản lý giỏ hàng:** Thêm, xóa, chỉnh sửa số lượng sản phẩm trong giỏ hàng trước khi thanh toán.

**Thanh toán trực tuyến:** Thực hiện thanh toán qua các cổng tích hợp như Apple Pay hoặc Google Pay, đảm bảo an toàn và tiện lợi.

**Quản lý tài khoản và đơn hàng:** Đăng ký, đăng nhập, cập nhật thông tin cá nhân, theo dõi trạng thái đơn hàng và nhận thông báo khuyến mãi.

### **3.2.3. Yêu cầu phi chức năng**

**Hiệu suất:** Ứng dụng phải phản hồi trong vòng 2 giây cho các thao tác cơ bản như tìm kiếm, tải danh sách sản phẩm, API backend xử lý tối đa 100 yêu cầu đồng thời.

**Bảo mật:** Dữ liệu người dùng được mã hóa qua HTTPS, xác thực bằng JWT; thanh toán tuân thủ các quy chuẩn an toàn.

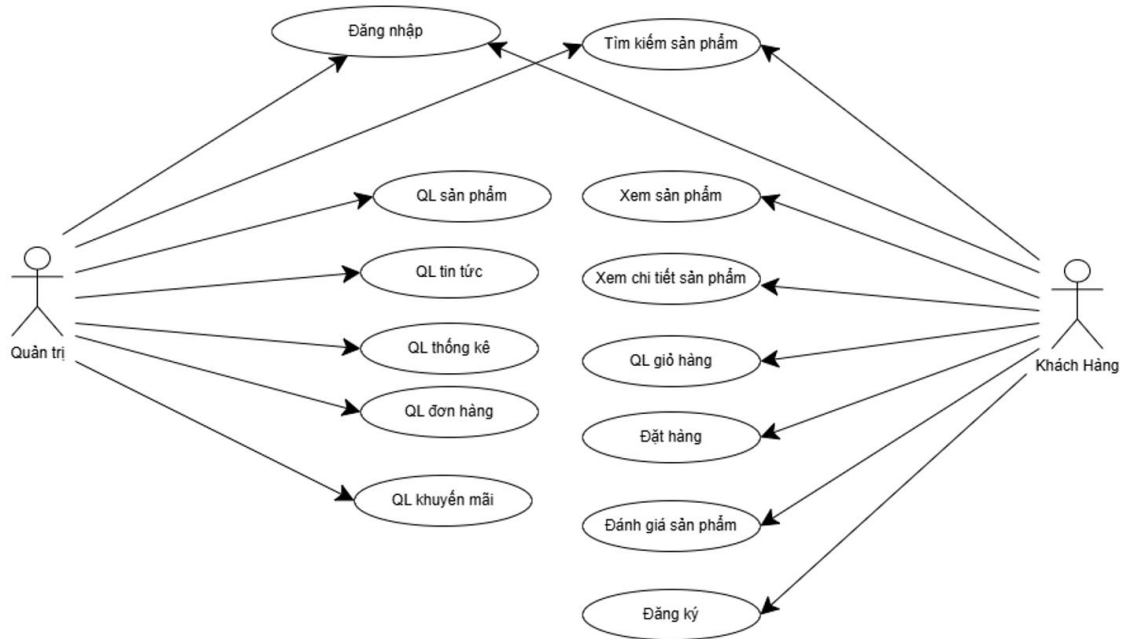
**Khả năng mở rộng:** Hệ thống hỗ trợ thêm tính năng mới (ví dụ: gợi ý sản phẩm bằng AI) mà không cần thay đổi cấu trúc chính.

**Tính tương thích:** Ứng dụng hoạt động mượt mà trên Android (từ phiên bản 8.0) và iOS (từ phiên bản 12.0) với giao diện responsive trên các kích thước màn hình.



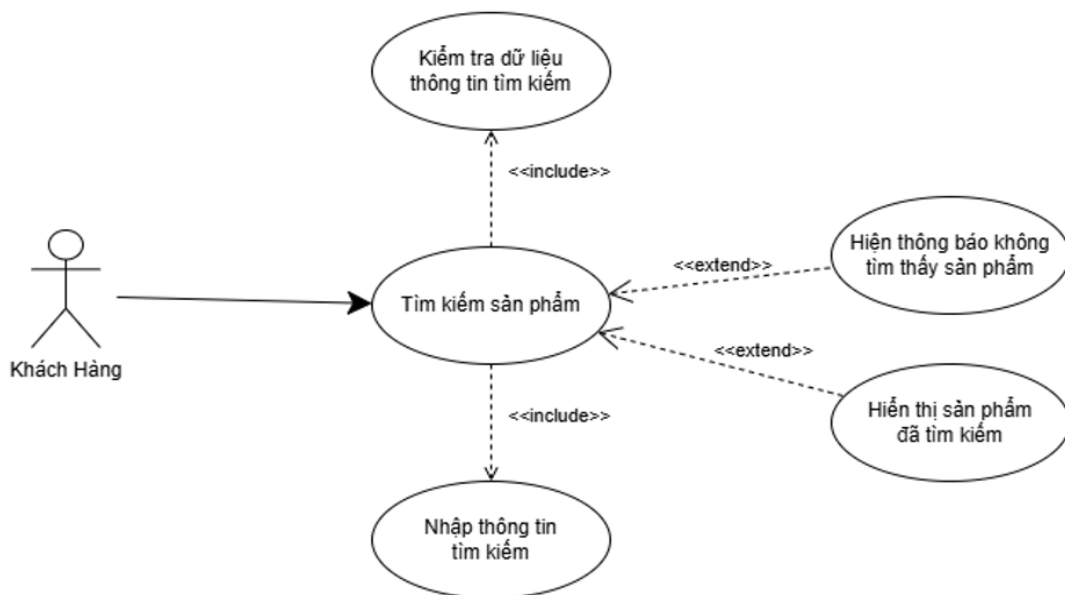
### 3.3. Sơ đồ use case

#### 3.3.1. Sơ đồ use case tổng quát



Hình 3.1 Sơ đồ use case tổng quát

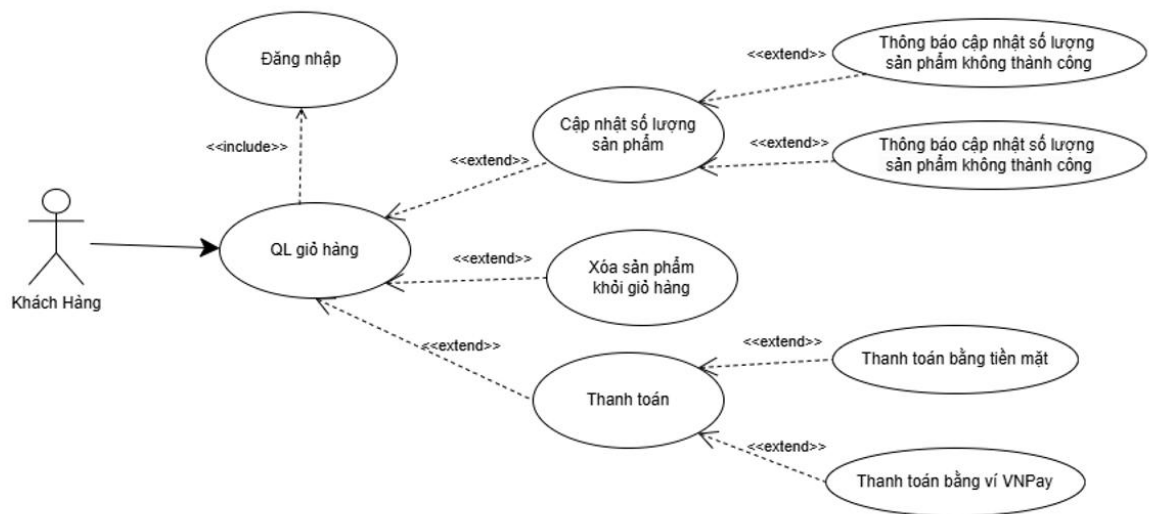
#### 3.3.2. Sơ đồ Use Case tìm kiếm sản phẩm



Hình 3.2 Sơ đồ Use Case tìm kiếm sản phẩm

*Mô tả:* Tác nhân khách hàng có thể tìm kiếm sản phẩm trên hệ thống bằng cách nhập thông tin sản phẩm muốn tìm.

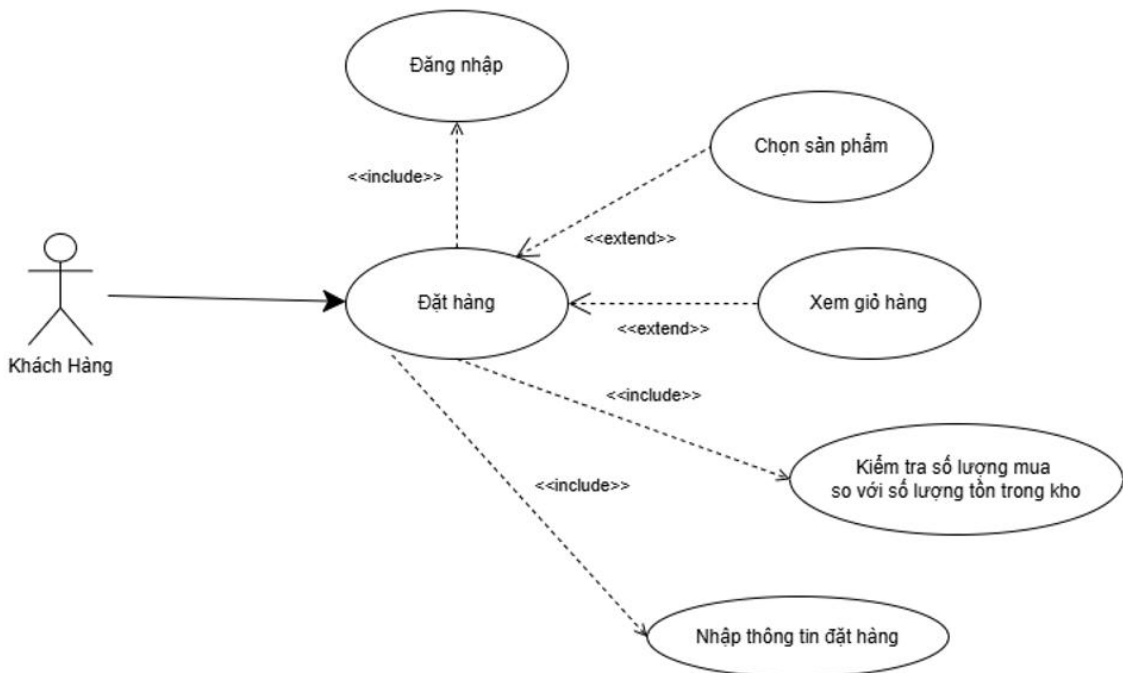
### 3.3.3. Sơ đồ Use Case quản lý giỏ hàng



Hình 3.3 Sơ đồ Use Case quản lý giỏ hàng

*Mô tả:* tác nhân khách hàng cần đăng nhập để sử dụng chức năng “Quản lý giỏ hàng”. Các hành động bao gồm: cập nhật số lượng sản phẩm (có thể thành công hoặc không thành công), xóa sản phẩm khỏi giỏ hàng, và thanh toán (với các tùy chọn thanh toán bằng tiền mặt hoặc Apple Pay và Google Pay).

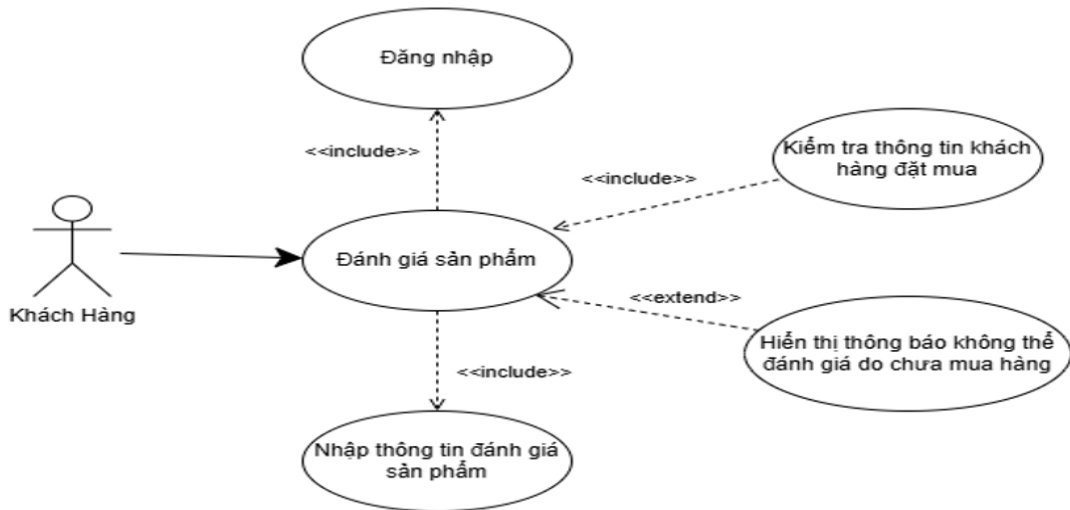
### 3.3.4. Sơ đồ Use Case đặt hàng



Hình 3.4 Sơ đồ Use Case đặt hàng

*Mô tả:* tác nhân khách hàng cần đăng nhập để sử dụng chức năng “Đặt hàng”. Các bước quan trọng bao gồm: kiểm tra số lượng mua so với tồn kho và nhập thông tin đặt hàng. Ngoài ra, chức năng này để khách hàng chọn sản phẩm hoặc xem giỏ hàng trước khi đặt hàng.

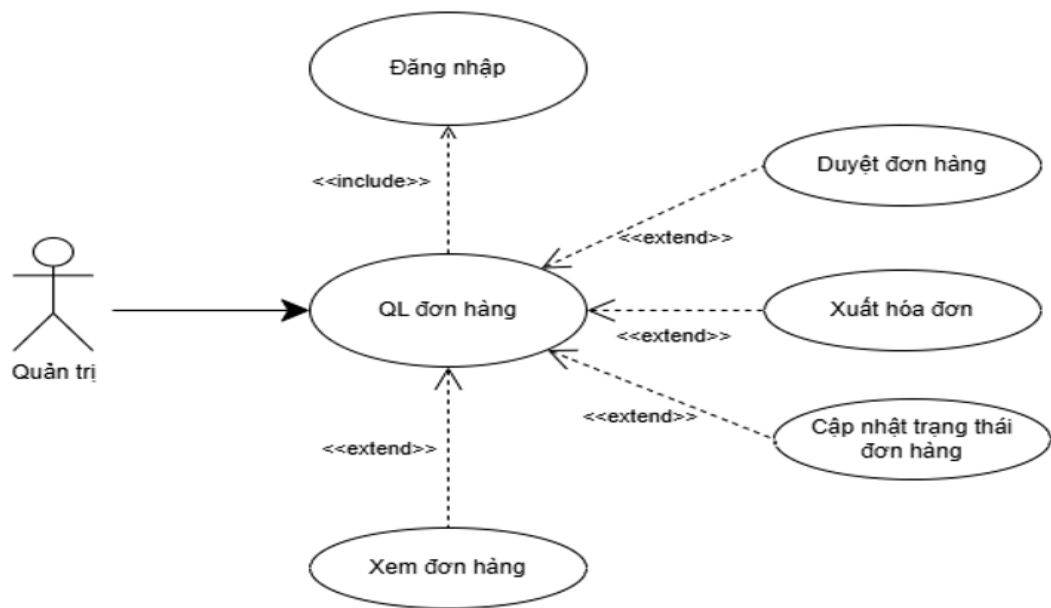
### 3.3.5. Sơ đồ Use Case đánh giá sản phẩm



Hình 3.5 Sơ đồ Use Case đánh giá sản phẩm

*Mô tả:* tác nhân khách hàng có thể thực hiện hành động “Đánh giá sản phẩm”. Để thực hiện đánh giá, khách hàng cần phải đăng nhập và nhập thông tin đánh giá sản phẩm. Các trường hợp kiểm tra thông tin khách hàng đã đặt mua sản phẩm (Kiểm tra thông tin khách hàng đặt mua) và hiển thị thông báo nếu khách hàng chưa mua hàng (Hiện thị thông báo không thể đánh giá do chưa mua hàng).

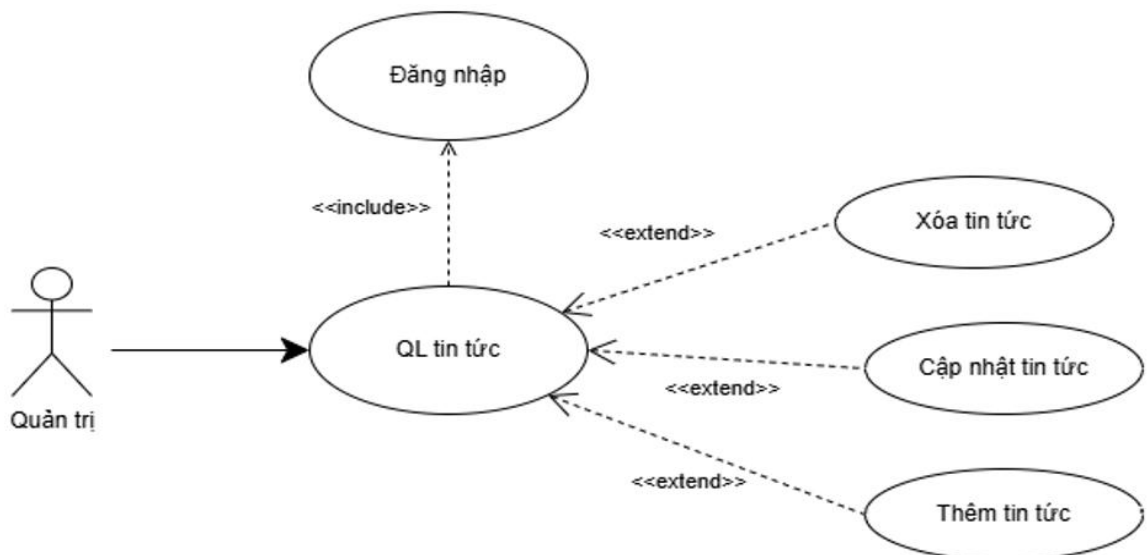
### 3.3.6. Sơ đồ Use Case quản lý đơn hàng



Hình 3.6 Sơ đồ Use Case quản lý đơn hàng

*Mô tả:* tác nhân quản trị có thể thực hiện các chức năng sau: duyệt đơn hàng, xuất hóa đơn, cập nhật trạng thái đơn hàng và xem các đơn hàng. Người quản trị cần phải đăng nhập vào hệ thống trước khi có thể thực hiện các chức năng này.

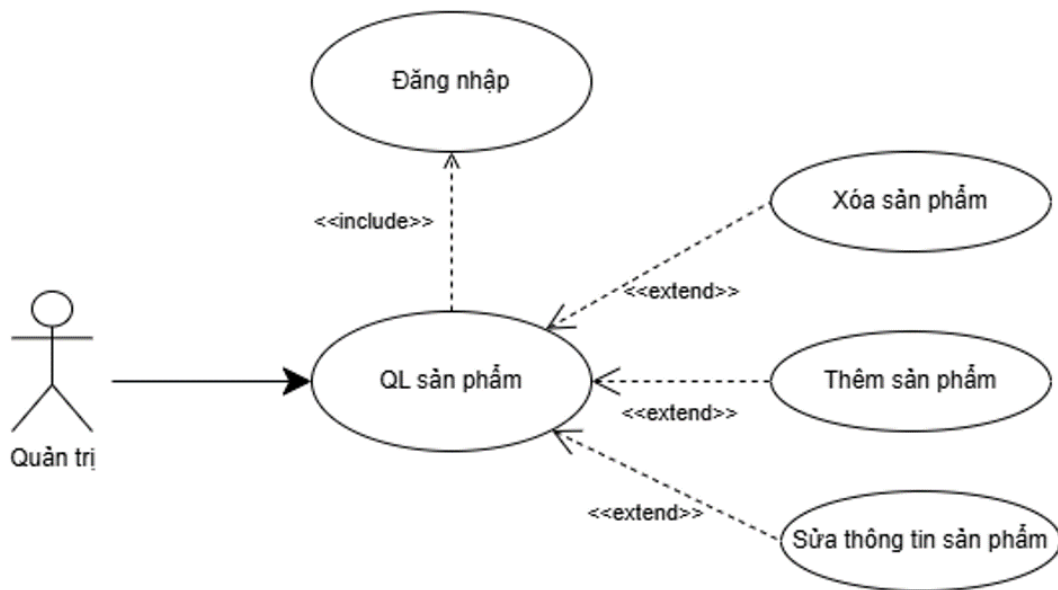
### 3.3.7. Sơ đồ Use Case quản lý tin tức



Hình 3.7 Sơ đồ Use Case quản lý tin tức

*Mô tả:* tác nhân người quản trị cần đăng nhập và thực hiện các chức năng như thêm tin tức, cập nhật tin tức và xóa tin tức.

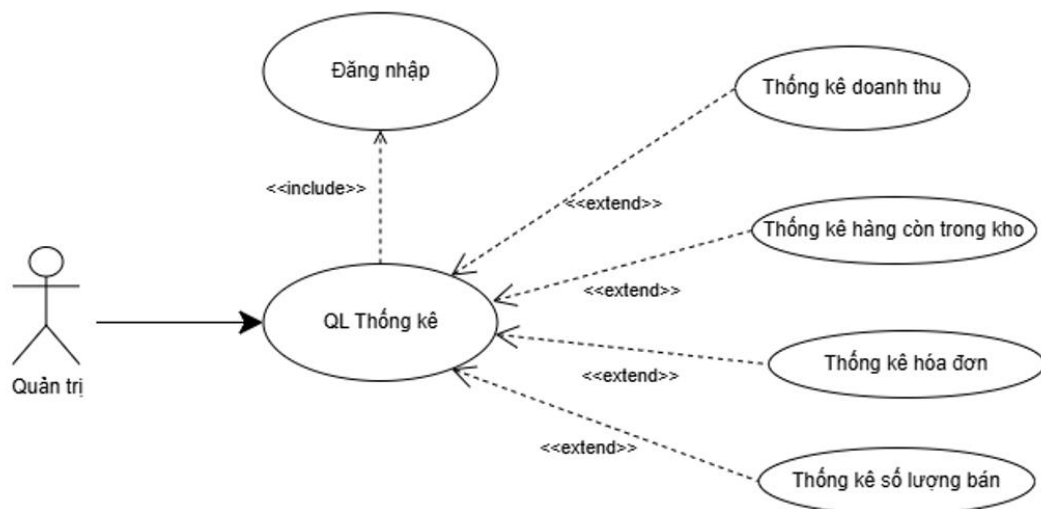
### 3.3.8. Sơ đồ Use Case quản lý sản phẩm



Hình 3.8 Sơ đồ Use Case quản lý sản phẩm

*Mô tả:* tác nhân quản trị thực hiện các chức năng liên quan đến quản lý sản phẩm. Quản trị cần phải thực hiện đăng nhập trước khi có thể truy cập vào các chức năng bao gồm xóa sản phẩm, thêm sản phẩm và sửa thông tin sản phẩm.

### 3.3.9. Sơ đồ Use Case quản lý thống kê

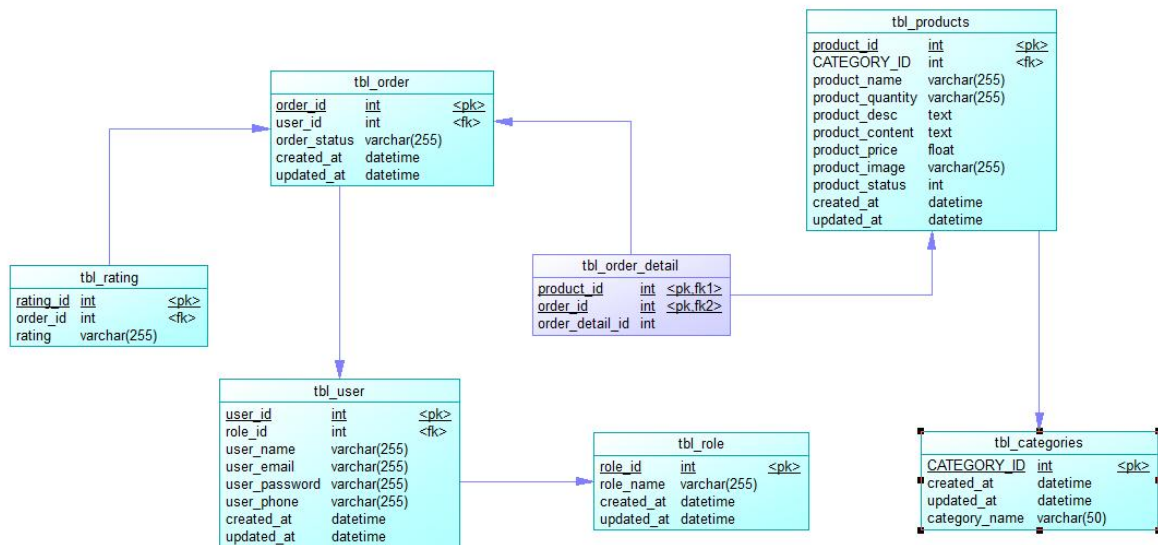


Hình 3.9 Sơ đồ Use Case quản lý thống kê

*Mô tả:* tác nhân quản trị cần thực hiện thao tác đăng nhập để có thể quản lý thống kê. Khi quản lý thống kê, quản trị có thể thực hiện các chức năng cụ thể như thống kê doanh thu, thống kê hàng còn trong kho, thống kê hóa đơn và thống kê số lượng bán.



### 3.5.2. Mô hình dữ liệu mức logic



Hình 3.12 Mô hình dữ liệu mức logic

### 3.5.3. Cấu trúc dữ liệu dưới dạng JSON

#### Đối tượng users

```
{  
  "name": "",  
  "email": "",  
  "password": "",  
  "address": "",  
  "type": "",  
  "cart": [  
    {  
      "product": {},  
      "quantity":  
    }  
  ]  
}
```

### **Đối tượng products**

```
{  
  "product_id": "",  
  "name": "",  
  "description": "",  
  "images": [],  
  "quantity": 0,  
  "price": 0,  
  "category": "",  
  "ratings": []  
}
```

### **Đối tượng orders**

```
{  
  "order_id": "",  
  "products": [  
    {  
      "product": {},  
      "quantity": 0  
    }  
  ],  
  "totalPrice": 0,  
  "address": "",  
  "userId": "",  
  "orderedAt": 0,  
  "status": 0  
}
```



### 3.5.4. Dữ liệu mẫu

#### Đối tượng users

```
{  
  "user_id": "6896fe2cb9e682f59cc123fc",  
  "name": "Tien",  
  "email": "Tien@gmail.com",  
  "password":  
"$2b$06$KjO7o9fwWcN9Vz8WocBNienPgwfHiiaMtQUMyz47NmkgGbVP0  
pBv6",  
  "address": "73, Kiên Thị Nhẫn, Châu Thành, Trà Vinh - 67000",  
  "type": "user",  
  "cart": []  
}
```

#### Đối tượng products

```
{  
  "product_id": "689cb6cedf3fa6b1f1fff388",  
  "name": "Thảm yoga",  
  "description": "Mô tả chi tiết sản phẩm - Thảm tập yoga định tuyến Chính  
Hãng Domik,  
  "images": [  
    "https://res.cloudinary.com/dngxulpam/image/upload/v1755100877/Th%E1%B  
A%A3m%20yoga/v89fidbjoxtkio1xrlzn.jpg"  
  ],  
  "quantity": 99,  
  "price": 108000,  
  "category": "Yoga",  
  "ratings": []  
}
```

```
}
```

### Đối tượng orders

```
{
```

```
  "order_id": "689cb7c6df3fa6b1f1fff47b",
```

```
  "products": [
```

```
    {
```

```
      "product": {
```

```
        "product_id": "689cb6cedf3fa6b1f1fff388",
```

```
        "name": "Thảm yoga",
```

```
        "description": "Mô tả chi tiết sản phẩm - Thảm tập yoga định tuyến  
Chính Hãng Domik,
```

```
        "images": [
```

```
        "https://res.cloudinary.com/dngxulpam/image/upload/v1755100877/Th%E1%BA%A3m%20yoga/v89fidbjoxtkio1xrlzn.jpg"
```

```
        ],
```

```
        "quantity": 99,
```

```
        "price": 108000,
```

```
        "category": "Yoga",
```

```
        "ratings": []
```

```
      },
```

```
      "quantity": 1
```

```
    }
```

```
  ],
```

```
  "totalPrice": 108000,
```

```
  "address": "73, Kiên Thị Nhân, Châu Thành, Trà Vinh - 67000",
```

```
  "userId": "6896fe2cb9e682f59cc123fc",
```

```

    "orderedAt": 1755101126557,

    "status": 1

}

```

### 3.5.5. Các thực thể

*Bảng 3.1 Thực thể role (vai trò người dùng)*

Thuộc tính	Mô tả	Kiểu dữ liệu
role_id	ID vai trò	String
role_name	Số sao đánh giá	String
created_at	Thời gian tạo	Timestamp
updated_at	Thời gian cập nhật	Timestamp

*Bảng 3.2 Thực thể rating (sao đánh giá)*

Thuộc tính	Mô tả	Kiểu dữ liệu
user_ID	ID người dùng	String
rating	Số sao đánh giá	String

*Bảng 3.3 Thực thể user (người dùng)*

Thuộc tính	Mô tả	Kiểu dữ liệu
user_ID	ID người dùng	String
name	Tên người dùng	String
email	Email người dùng	String
password	Mật khẩu người dùng	String
address	Địa chỉ người dùng	String
type	Vai trò người dùng	String

*Bảng 3.4 Thực thể products (sản phẩm)*

Thuộc tính	Mô tả	Kiểu dữ liệu
product_ID	ID sản phẩm	String
name	Tên sản phẩm	String

description	Mô tả sản phẩm	String
images	Hình ảnh sản phẩm	String
quantity	Số lượng sản phẩm	Number
price	Giá sản phẩm	Number

*Bảng 3.5 Thực thể orders (đơn hàng)*

Thuộc tính	Mô tả	Kiểu dữ liệu
order_ID	ID đơn hàng	String
totalPrice	Tổng tiền đơn hàng	String
address	Địa chỉ giao hàng	String
orderedAt	Thời điểm đặt hàng	String
status	Trạng thái đơn hàng	String

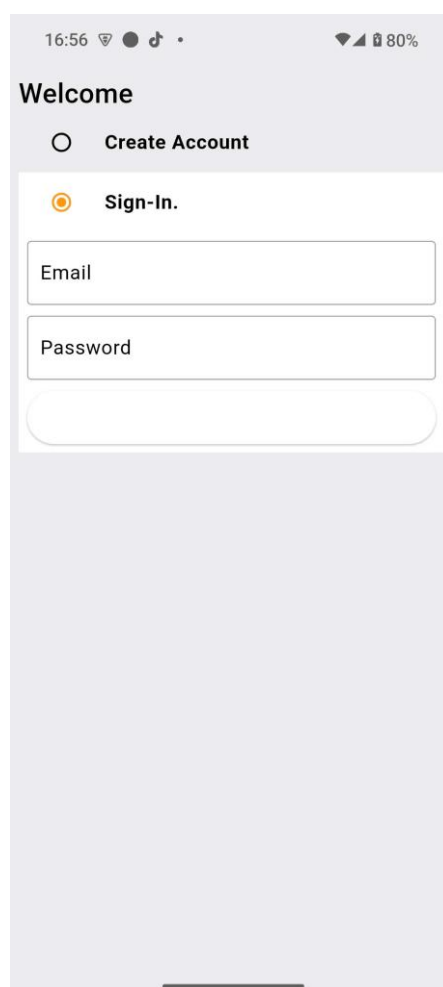
*Bảng 3.6 Thực thể categories (danh mục)*

Tên cột	Mô tả	Kiểu dữ liệu
category_id	ID của danh mục	String
category_name	Tên của danh mục	String
created_at	Thời gian tạo danh mục	Timestamp
updated_at	Thời gian cập nhật danh mục	Timestamp

## CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

### 4.1. Giao diện đăng nhập

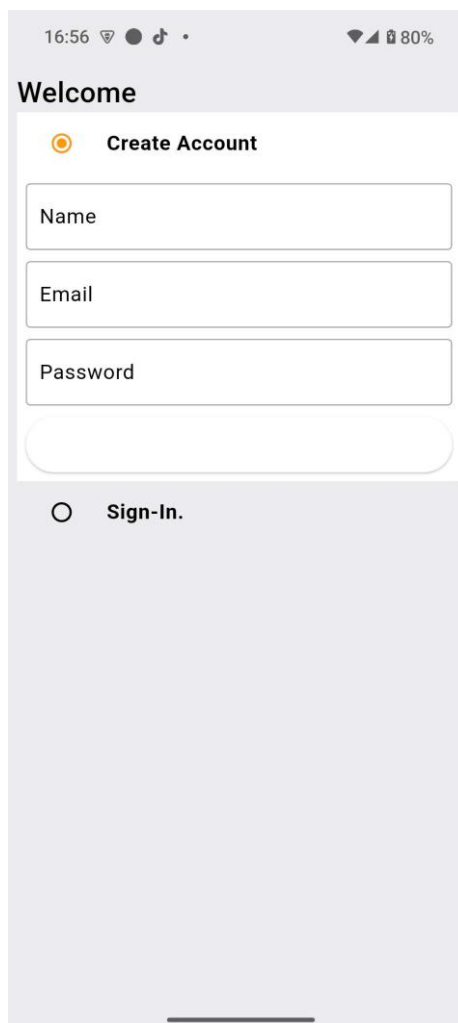
Giao diện đăng nhập của ứng dụng bán dụng cụ thể thao được thiết kế để người dùng truy cập nhanh chóng vào hệ thống và sử dụng các tính năng mua sắm cá nhân hóa. Người dùng có thể đăng nhập bằng email và mật khẩu. Khi truy cập, người dùng nhập thông tin vào các trường email và mật khẩu, sau đó nhấn nút “Đăng nhập” để vào hệ thống, được xử lý qua API POST /login tới backend Node.js sử dụng xác thực JWT. Nếu thông tin đăng nhập sai, một SnackBar sẽ hiển thị thông báo lỗi ngay lập tức để chỉnh sửa. Người dùng chưa có tài khoản có thể nhấn liên kết “Đăng ký” để chuyển sang trang tạo tài khoản mới. Giao diện sử dụng Flutter với bố cục responsive, màu sắc thân thiện, đảm bảo đăng nhập dễ dàng, an toàn, giúp người dùng nhanh chóng tiếp cận các tính năng như tìm kiếm sản phẩm, quản lý giỏ hàng, thanh toán trực tuyến và theo dõi đơn hàng.



Hình 4.1 Giao diện đăng nhập

## 4.2. Giao diện đăng ký

Giao diện đăng ký của ứng dụng bán dụng cụ thể dục thể thao được thiết kế để hỗ trợ người dùng dễ dàng tạo tài khoản mới và nhanh chóng tham gia vào hệ thống mua sắm. Người dùng có thể đăng ký bằng email và mật khẩu. Khi truy cập, người dùng nhập thông tin vào các trường bao gồm họ tên, email, mật khẩu, sau đó nhấn nút “Đăng ký” để gửi dữ liệu qua API POST /register tới backend Node.js, sử dụng xác thực JWT. Trong trường hợp thông tin nhập vào không hợp lệ (như email đã tồn tại hoặc mật khẩu không khớp), một SnackBar sẽ hiển thị thông báo lỗi ngay lập tức để người dùng chỉnh sửa. Người dùng đã có tài khoản có thể nhấn liên kết “Đăng nhập” để chuyển sang trang đăng nhập. Giao diện được xây dựng bằng Flutter, sử dụng bố cục responsive với nền trắng, đảm bảo trải nghiệm trực quan, an toàn, giúp người dùng yên tâm khi tạo tài khoản để sử dụng các tính năng như tìm kiếm sản phẩm, quản lý giỏ hàng, thanh toán.

The image is a screenshot of a mobile application's 'Welcome' screen. At the top, the status bar shows the time 16:56, signal strength, and battery level at 80%. The app's title 'Welcome' is displayed in a bold, black font. Below the title, there is a section for 'Create Account' with a yellow circular icon containing a person silhouette. This section contains four input fields: 'Name', 'Email', 'Password', and a fourth empty field. Below these fields, there is a 'Sign-In' option with a radio button and the text 'Sign-In.'.

Hình 4.2 Giao diện đăng ký

### 4.3. Giao diện người dùng

#### 4.3.1. Giao diện trang chủ

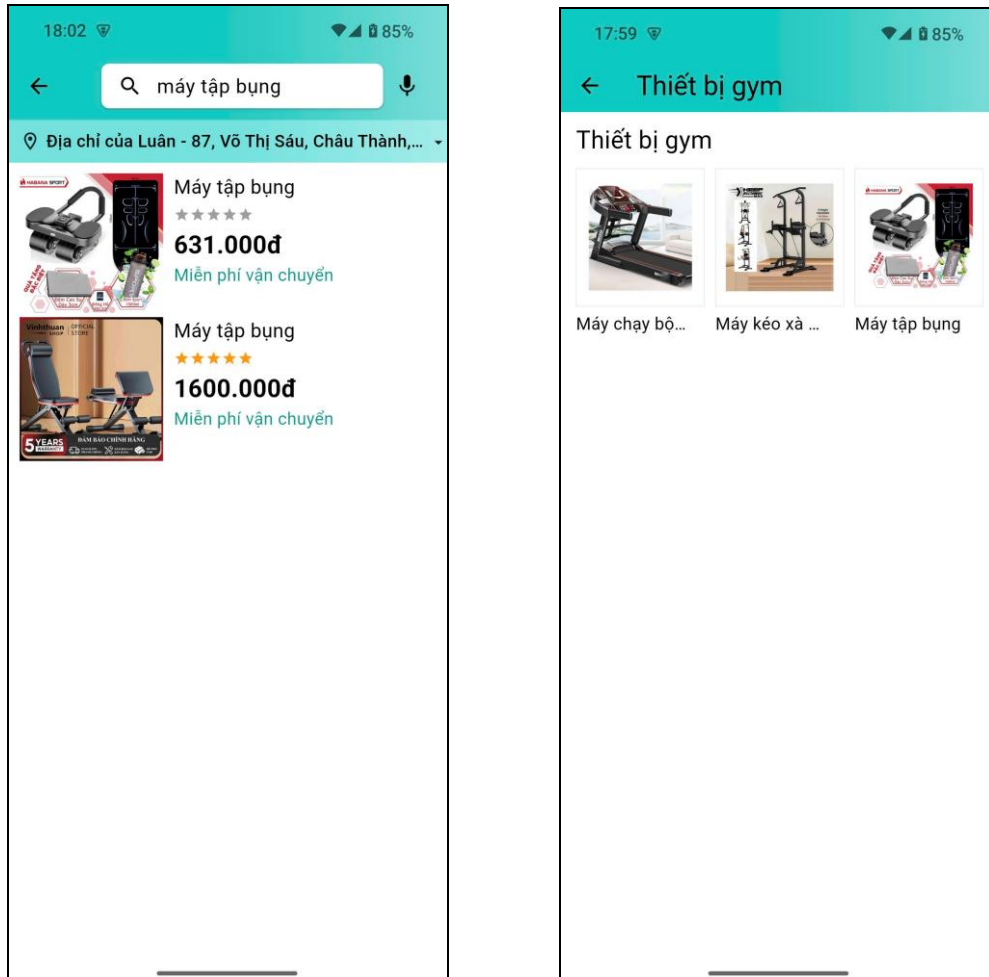
Giao diện trang chủ của ứng dụng bán dụng cụ thể dục thể thao được thiết kế bằng Flutter, mang lại trải nghiệm trực quan và thân thiện trên cả Android và iOS. Ở đầu trang, thanh tìm kiếm “Tìm kiếm sản phẩm” với biểu tượng micro, đặt trên nền xanh nhạt, cho phép người dùng nhập từ khóa nhanh chóng. Dưới đó, thanh địa chỉ hiển thị vị trí của người dùng đã từng cung cấp. Tiếp theo là danh mục sản phẩm với các biểu tượng như “Thiết bị gym”, “Dụng cụ thể dục tại nhà”, “Yoga”, “Thể thao ngoài trời” và “Phụ kiện liên quan”, được sắp xếp ngang với màu sắc nổi bật. Phần giữa trang nổi bật với banner quảng cáo, hiển thị hình ảnh vận động viên, thông tin giảm giá để thu hút người dùng. Dưới banner, phần sản phẩm bán chạy và các tùy chọn giao hàng. Thanh điều hướng dưới cùng gồm các biểu tượng “Trang chủ”, “Tài khoản”, và “Giỏ hàng”, giúp người dùng dễ dàng chuyển đổi giữa các chức năng. Giao diện responsive, được kiểm thử trên thiết bị thực tế, đảm bảo hiệu suất cao và giao diện mượt mà.



Hình 4.3 Giao diện trang chủ

### 4.3.2. Giao diện trang sản phẩm

Giao diện trang sản phẩm được thiết kế đơn giản nhưng đầy đủ các thông tin cần thiết với các hình ảnh sản phẩm phong phú nhằm thu hút sự chú ý của khách hàng. Trang chủ bao gồm chức năng tìm kiếm, hiển thị các sản phẩm nổi bật và sao đánh giá sản phẩm.



Hình 4.4 Giao diện trang sản phẩm

### 4.3.3. Giao diện trang chi tiết sản phẩm

Giao diện trang chi tiết sản phẩm của ứng dụng bán dụng cụ thể dục thể thao, phát triển bằng Flutter, mang lại trải nghiệm trực quan với hình ảnh sản phẩm, mã sản phẩm, đánh giá sao, và thông tin cơ bản. Tiêu đề nổi bật cùng logo thương hiệu, kèm mô tả ngắn gọn về đặc điểm và giá cả. Thanh điều hướng dưới cùng hỗ trợ di chuyển dễ dàng giữa các chức năng. Giao diện responsive, kết nối mượt mà với backend Node.js qua API, đảm bảo an toàn và tiện lợi.

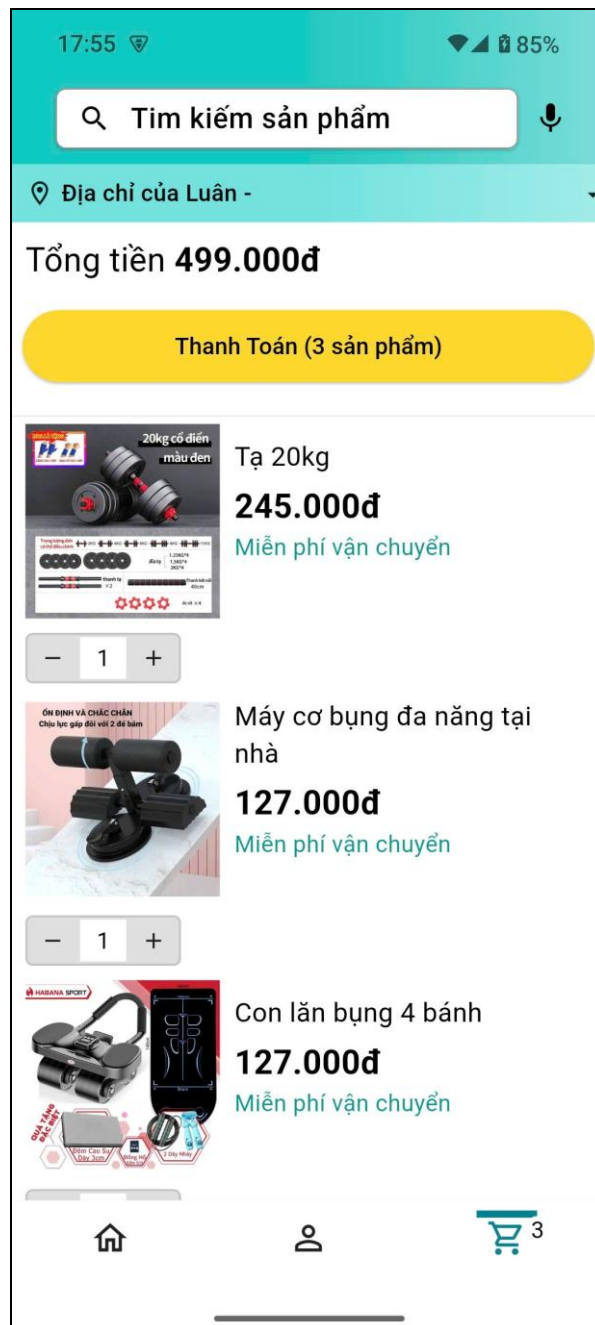




Hình 4.5 Giao diện chi tiết sản phẩm

#### 4.3.4. Giao diện trang giỏ hàng

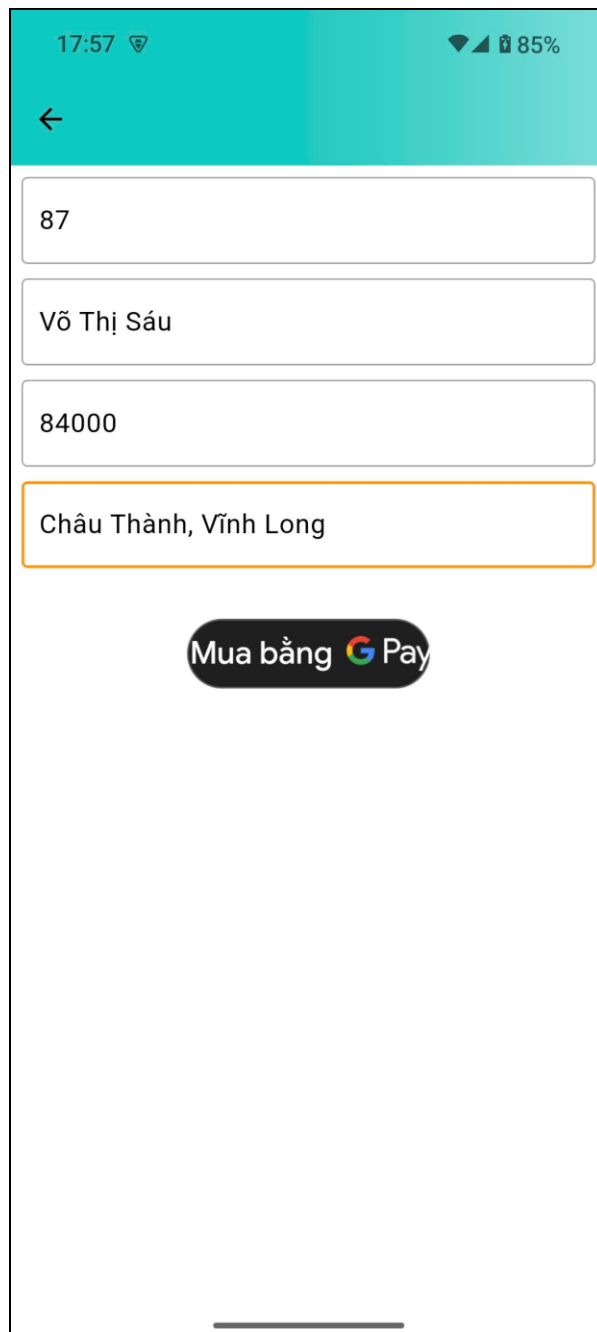
Giao diện trang giỏ hàng của ứng dụng bán dụng cụ thể dục thể thao được thiết kế để cung cấp cái nhìn tổng quan và dễ quản lý cho người dùng. Phần đầu hiển thị danh sách các sản phẩm đã chọn, bao gồm hình ảnh, tên, số lượng và giá cả, cùng tùy chọn chỉnh sửa số lượng sản phẩm hoặc xóa sản phẩm. Tổng giá trị đơn hàng được hiển thị nổi bật, kèm nút thanh toán để chuyển sang bước thanh toán. Thông tin giao hàng và khuyến mãi như miễn phí vận chuyển được hiển thị ngắn gọn, cho phép người dùng cập nhật địa chỉ.



Hình 4.6 Giao diện trang giỏ hàng

#### 4.3.5. Giao diện trang thanh toán

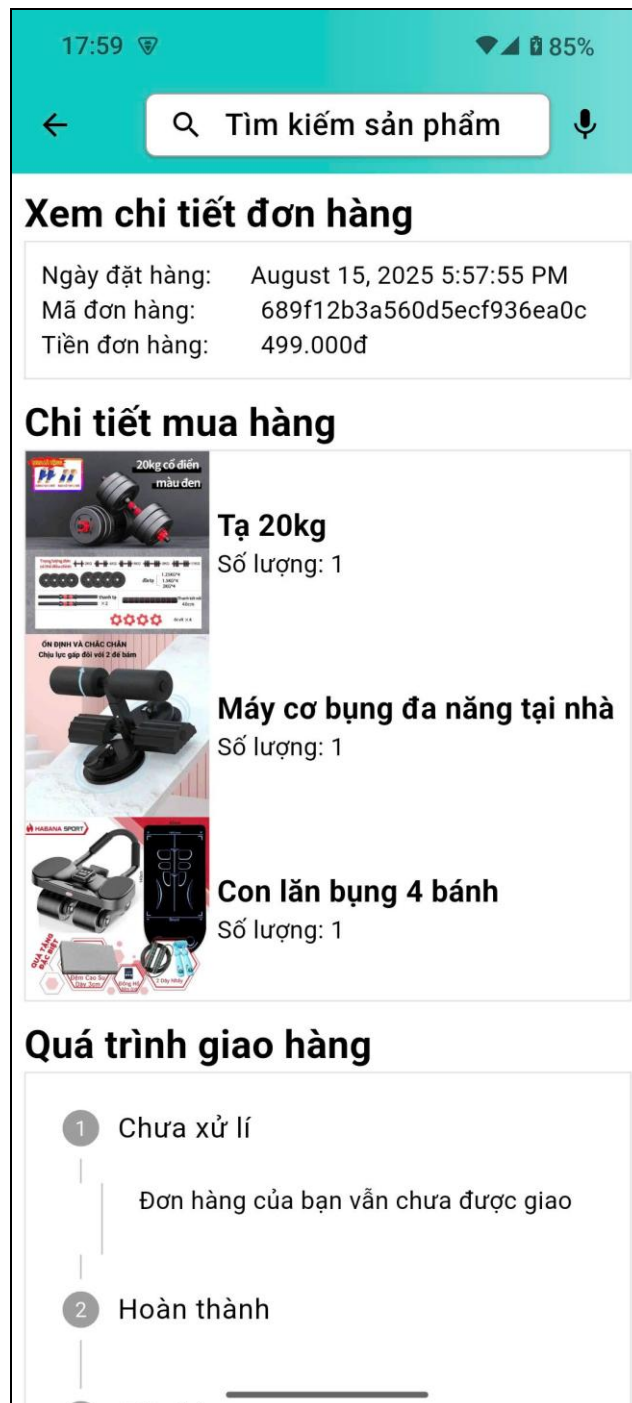
Giao diện trang thanh toán của ứng dụng với giao diện đơn giản giúp người dùng dễ dàng thanh toán đơn hàng của họ, với các trường nhập địa chỉ chỗ ở hoặc địa điểm nhận hàng mà người dùng muốn nhận hàng. Những người dùng đã từng mua hàng thì địa chỉ nhận hàng mà họ đã nhập trước đó sẽ được lưu lại và cho phép người dùng thay đổi địa chỉ trong lần mua hàng tiếp theo. Sau khi nhập đầy đủ các thông tin cần thiết, người dùng có thể thanh toán qua Google Pay hoặc Apple Pay tùy vào hệ điều hành trên thiết bị của họ.



Hình 4.7 Giao diện trang thanh toán

#### 4.3.6. Giao diện trang chi tiết đơn hàng

Giao chi tiết đơn hàng trong ứng dụng cho phép người dùng theo dõi đầy đủ thông tin về từng đơn mua. Giao diện hiển thị rõ mã đơn hàng, ngày đặt, trạng thái xử lý và phương thức thanh toán. Người dùng có thể xem danh sách sản phẩm đã mua kèm hình ảnh, số lượng, đơn giá và tổng tiền. Ngoài ra, trang còn cung cấp thông tin người nhận như họ tên, số điện thoại, địa chỉ giao hàng. Hệ thống hỗ trợ chức năng theo dõi tiến trình đơn hàng (chờ xác nhận, đang giao, đã giao). Nhờ đó, người dùng quản lý đơn hàng dễ dàng, minh bạch và tiện lợi.



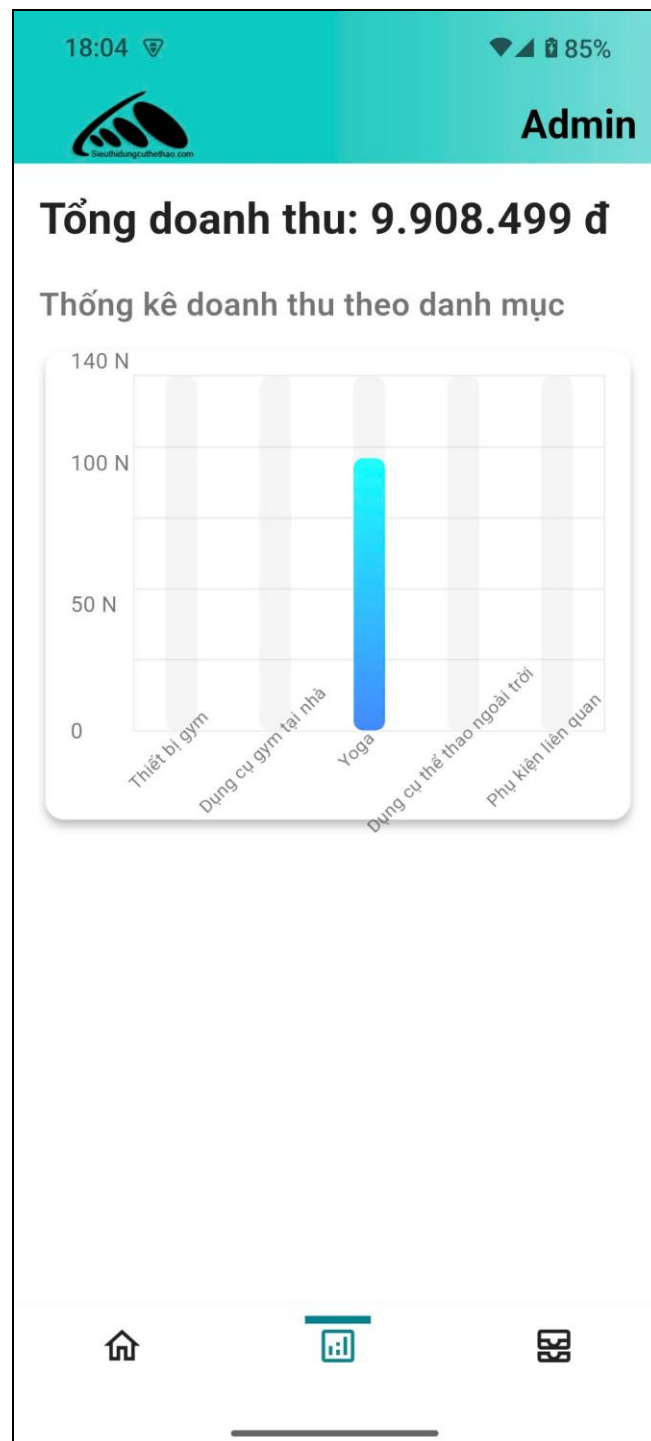
Hình 4.8 Giao diện chi tiết đơn hàng

## 4.4. Giao diện quản trị

### 4.4.1. Giao diện trang tổng quan

Giao diện tổng quan của quản trị viên trong ứng dụng đóng vai trò như bảng điều khiển trung tâm, giúp quản trị viên theo dõi toàn bộ hoạt động bán hàng. Giao diện cung cấp các chỉ số quan trọng như tổng doanh thu, số lượng đơn hàng, sản phẩm. Các thông tin được trực quan hóa bằng biểu đồ và bảng số liệu để dễ dàng phân tích. Ngoài ra, admin có thể nhanh chóng theo dõi trạng thái đơn hàng mới, đơn chờ xử lý

và đơn đã hoàn thành. Trang cũng hỗ trợ quản lý sản phẩm và người dùng một cách tập trung. Nhờ đó, admin có cái nhìn tổng thể, kịp thời đưa ra quyết định quản lý hiệu quả.

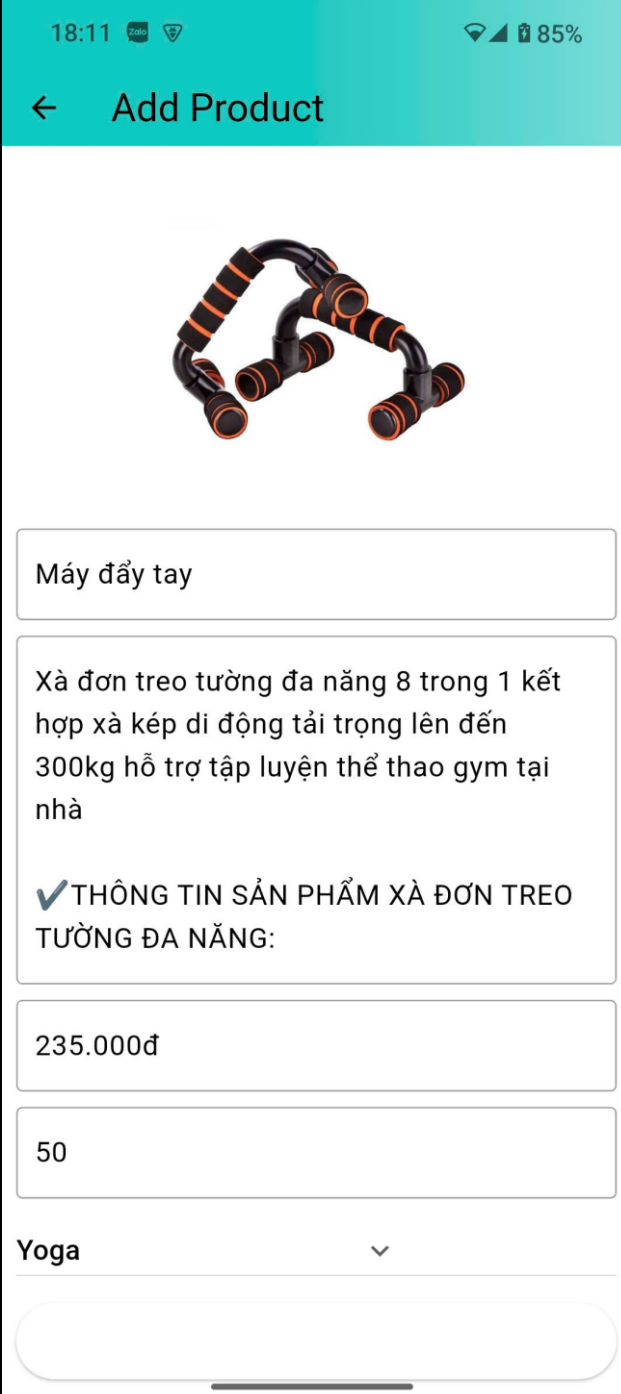


Hình 4.9 Giao diện tổng quan

#### 4.4.2. Giao diện thêm sản phẩm


Giao diện bao gồm các trường nhập thông tin cơ bản như tên sản phẩm, mô tả, giá bán, số lượng tồn kho và danh mục. Ngoài ra, admin có thể tải lên hình ảnh minh

họa nhằm giúp sản phẩm hiển thị trực quan hơn trên ứng dụng. Hệ thống cũng hỗ trợ các tùy chọn như gắn nhãn khuyến mãi, phân loại theo thương hiệu hoặc tình trạng hàng. Sau khi hoàn tất nhập liệu, admin chỉ cần nhấn nút lưu để cập nhật sản phẩm vào kho dữ liệu. Nhờ đó, việc quản lý và bổ sung sản phẩm trở nên nhanh chóng, chính xác và thuận tiện.



18:11 85%

← Add Product



Máy đẩy tay

Xà đơn treo tường đa năng 8 trong 1 kết hợp xà kép di động tải trọng lên đến 300kg hỗ trợ tập luyện thể thao gym tại nhà

✓ THÔNG TIN SẢN PHẨM XÀ ĐƠN TREO TƯỜNG ĐA NĂNG:

235.000đ

50

Yoga

Hình 4.10 Giao diện thêm sản phẩm

#### 4.4.3. Giao diện các đơn hàng

Giao diện các đơn hàng hiển thị các đơn hàng đã người dùng được đặt để quản trị viên dễ dàng quản lý xem quá trình xử lý của các đơn hàng đã giao đến người dùng hay chưa để có các đáp ứng tốt cho trải nghiệm khách hàng trong quá trình mua hàng.

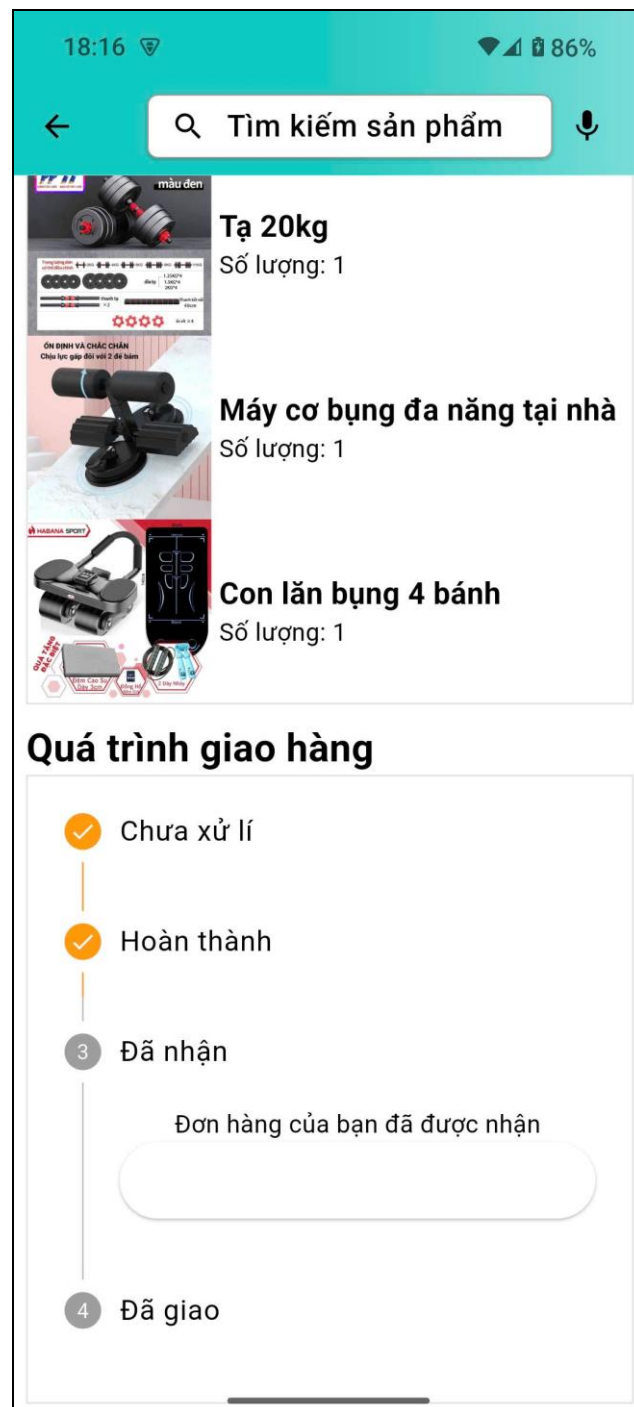


Hình 4.11 Giao diện các đơn hàng

#### 4.4.4. Giao diện duyệt trạng thái đơn hàng

Giao diện duyệt trạng thái đơn hàng của quản trị viên cho phép kiểm soát và xử lý các đơn hàng trong hệ thống. Giao diện hiển thị danh sách chi tiết đơn, bao gồm mã

đơn hàng, tên khách hàng, ngày đặt và giá trị đơn. Quản trị viên có thể lựa chọn thay đổi trạng thái đơn hàng như: chờ xác nhận, đang xử lý, đang giao, đã hoàn thành. Các thao tác được hỗ trợ bằng nút bấm hoặc menu chọn, giúp quá trình duyệt nhanh chóng và chính xác. Ngoài ra, hệ thống còn cung cấp thông báo khi trạng thái được cập nhật để đảm bảo khách hàng nắm bắt kịp thời.



Hình 4.12 Giao diện duyệt trạng thái đơn hàng



## CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 5.1. Kết luận

#### 5.1.1. Kết quả và đóng góp của đồ án

Đồ án đã đạt được mục tiêu đề ra là tìm hiểu, nghiên cứu và ứng dụng API vào việc xây dựng một ứng dụng bán dụng cụ thể dục thể thao trên thiết bị di động. Trong quá trình thực hiện, tôi đã nắm vững kiến thức về cách thiết kế và triển khai API để kết nối giữa giao diện người dùng và cơ sở dữ liệu, đồng thời áp dụng vào việc xây dựng các chức năng cơ bản của một ứng dụng thương mại điện tử như: đăng ký/dăng nhập, quản lý sản phẩm, giỏ hàng, đặt hàng, theo dõi trạng thái đơn hàng và quản trị hệ thống. Ứng dụng bước đầu đã thể hiện khả năng hoạt động ổn định, giao diện thân thiện, hỗ trợ người dùng tìm kiếm và mua sắm dụng cụ thể dục thể thao một cách thuận tiện.

Về mặt đóng góp, đồ án không chỉ mang ý nghĩa trong phạm vi học tập mà còn có giá trị thực tiễn. Một mặt, nó giúp sinh viên củng cố và mở rộng kiến thức lập trình di động, cơ sở dữ liệu cũng như kỹ năng làm việc với API. Mặt khác, sản phẩm có thể được xem như một mô hình tham khảo cho các ứng dụng thương mại điện tử quy mô nhỏ, đặc biệt trong lĩnh vực dụng cụ thể thao vốn đang ngày càng phát triển. Ngoài ra, việc kết hợp lý thuyết và thực hành đã rèn luyện cho tôi khả năng tư duy hệ thống, làm việc và giải quyết vấn đề, từ đó tạo nền tảng cho những nghiên cứu và phát triển ứng dụng trong tương lai.

#### 5.1.2. Hạn chế

Mặc dù đã đạt được những kết quả nhất định, đồ án vẫn còn tồn tại một số hạn chế. Thứ nhất, do giới hạn về thời gian và nguồn lực, các chức năng của ứng dụng mới chỉ dừng lại ở mức cơ bản, chưa triển khai đầy đủ các tính năng nâng cao như gợi ý sản phẩm theo nhu cầu, tích hợp ví điện tử hay nhiều chương trình khuyến mãi. Thứ hai, giao diện người dùng còn đơn giản, chưa được tối ưu hóa hoàn toàn về trải nghiệm và tính thẩm mỹ, đặc biệt trên các thiết bị di động có nhiều kích thước màn hình khác nhau. Thứ ba, khả năng mở rộng và tính bảo mật của hệ thống chưa được kiểm thử ở quy mô lớn, vì vậy chưa đảm bảo đáp ứng tốt khi số lượng người dùng tăng cao. Ngoài ra, quá trình kiểm thử ứng dụng chủ yếu thực hiện trong môi trường

giả lập và trên một số thiết bị nhất định, nên tính đa dạng và độ tin cậy chưa thật sự toàn diện.

## **5.2. Hướng phát triển**

Trong tương lai, ứng dụng có thể được mở rộng và hoàn thiện hơn nhằm đáp ứng tốt hơn nhu cầu thực tế của người dùng. Trước hết, cần bổ sung các tính năng nâng cao như gợi ý sản phẩm thông minh dựa trên hành vi mua sắm, tích hợp các phương thức thanh toán điện tử phổ biến và hỗ trợ đa dạng hình thức vận chuyển. Bên cạnh đó, việc cải tiến giao diện người dùng theo hướng hiện đại, thân thiện và tối ưu trải nghiệm trên nhiều loại thiết bị di động sẽ giúp ứng dụng trở nên hấp dẫn hơn. Về mặt kỹ thuật, hệ thống cần được mở rộng về khả năng xử lý, nâng cao tính bảo mật dữ liệu và tối ưu hiệu năng để phục vụ lượng người dùng lớn. Ngoài ra, cần tích hợp thêm các công cụ quản lý cho nhà bán hàng như báo cáo doanh thu, thống kê đơn hàng, nhằm tăng tính thực tiễn. Với những định hướng này, ứng dụng hứa hẹn có thể trở thành một nền tảng thương mại điện tử chuyên biệt cho lĩnh vực dụng cụ thể dục thể thao, có khả năng áp dụng trong thực tiễn và tiếp tục phát triển lâu dài.

## DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Nguyễn Hoàng Duy Thiện (2013). Tài liệu giảng dạy môn Lập trình thiết bị di động. Trường Đại học Trà Vinh.
- [2] Nguyễn Hà Giang (2018). *Lập trình trên thiết bị di động*. Giáo trình giảng dạy, Trường Đại học Công nghệ TP.HCM HUTECH.
- [3] Ngô Bá Hùng và Đoàn Hòa Minh (2016). *Giáo trình Lập trình cho thiết bị di động*. Giáo trình giảng dạy, Trường Đại học Cần Thơ.
- [4] C. L. V. Tiến, "Mô hình MVC là gì? Ví dụ và cách ứng dụng MVC vào lập trình," vietnix.vn, 10 9 2024. [Online], trang web: <https://vietnix.vn/tim-hieu-mo-hinh-mvc-la-gi/>. [Ngày truy cập 10 08 2025].
- [5] Flutter, trang web: <https://flutter.dev/> . [Ngày truy cập 10 08 2025].
- [6] Dart là gì? Giới thiệu cơ bản về ngôn ngữ lập trình Dart, trang web: <https://200lab.io/blog/tu-hoc-ngon-dart-nhung-dieu-can-biet-truoc-khi-bat-dau/> . [Ngày truy cập 11 08 2025].
- [7] MongoDB là gì ? Định nghĩa và hiểu rõ A-Z về MongoDB, trang web: <https://itviec.com/blog/mongodb-la-gi/> . [Ngày truy cập 11 08 2025].
- [8] NodeJS là gì? NodeJS có phải ngôn ngữ lập trình, trang web: <https://codegym.vn/blog/nodejs-la-gi-nodejs-co-phai-la-ngon-ngu-lap-trinh>. [Ngày truy cập 12 08 2025].
- [9] RESTful API là gì? Các nguyên tắc của RESTful API, trang web: <https://stringee.com/vi/blog/post/restful-api-la-gi/>. [Ngày truy cập 12 08 2025].
- [10] ExpressJS là gì?, trang web: <https://fptshop.com.vn/tin-tuc/danh-gia/express-js-la-gi-174976>. [Ngày truy cập 12 08 2025].