KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ NGÀNH

HỌC KỲ I, NĂM HỌC 2024 - 2025

NGHIÊN CỨU KIẾN TRÚC CLEAN ARCHITECTURE VỚI MICROSERVICE CHO HỆ THỐNG BÁN SẢN PHẨM CÔNG NGHỆ APPLE

Giáo viên hướng dẫn:
Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Trương Hoàng Hưng
MSSV: 110121027
Lớp: DA21TTA

Trà Vinh, 09 tháng 01 năm 2025

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ I, NĂM HỌC 2023 - 2024

NGHIÊN CỨU KIẾN TRÚC CLEAN ARCHITECTURE VỚI MICROSERVICES CHO HỆ THỐNG BÁN SẢN PHẨM CÔNG NGHỆ APPLE

Giáo viên hướng dẫn:
Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Trương Hoàng Hưng
MSSV: 110121027
Lớp: DA21TTA

Trà Vinh, 09 tháng 01 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ..... tháng ...... năm ......

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ….. tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên, em muốn gửi lời cảm ơn sâu sắc đến thầy Nguyễn Bảo Ân, giảng viên bộ môn CNTT - Trường Đại học Trà Vinh. Thầy là người đã tận tình hướng dẫn, giúp đỡ, em trong suốt quá trình thực hiện đồ án.

Sau thời gian học tập và tích lũy kiến thức chuyên môn từ các môn học, cũng như thực hành các kiến thức cơ bản. Em còn tìm hiểu thêm các kiến thức về lập trình Website cũng như ngôn ngữ lập trình C# giúp cho bản thân biết thêm một cách khách quan về lĩnh vực này. Và áp dụng các kiến thức đã học trên lớp cũng như các nguồn tài liệu bổ sung vào dự án "Nghiên cứu kiến trúc Clean Architecture với Microservice cho hệ thống bán sản phẩm công nghệ của Apple".

Do chưa có nhiều kinh nghiệm cũng như các kiến thức còn hạn chế nên đề tài còn nhiều thiếu sót mong nhận được góp ý của quý thầy để khắc phục và hoàn thiện đề tài tốt hơn. Em xin chân thành cảm ơn.

MỤC LỤC

MỞ ĐẦU 13
CHƯƠNG 1: TỔNG QUAN 15
1.1. Đặt vấn đề 15
1.2. Mục đích 15
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT 17
2.1. Kiến trúc Clean Architecture 17
2.1.1. Tổng quan 17
2.1.2. Ưu điểm 18
2.1.3. Nhược điểm 18
2.2. Kiến trúc Microservice 18
2.2.1. Đặc điểm 19
2.2.2. Thành phần 19
2.2.3. Ưu điểm 20
2.3. RESTful API 21
2.3.1. Nguyên tắc thiết kế của REST 21
2.3.2. Phương thức HTTP trong RESTful API 22
2.3.3. Các quy ước thiết kế RESTful API 22
2.4. Ngôn ngữ Backend 23
2.4.1. Ngôn ngữ C Sharp (C#) 23
2.4.2. ASP.NET CORE 23
2.4.3. Entity Framework Core 24
2.5. SQL Server 25
2.5.1. Khái niệm 25
2.5.2. Cấu trúc 25
2.6. MediatR và CQRS Pattern 26
2.7. Docker 27
2.7.1. Khái niệm 27
2.7.2. Lợi ích 28
2.8. Postman 28
2.8.1. Khái niệm 28
2.8.2. Tính năng 28

2.9. Github Action 30
2.9.1. Khái niệm 30
2.9.2. Các thành phần chính 30
CHƯƠNG 3: THỰC NGHIỆM.....31
3.1. Xây dựng hệ thống 31
3.1.1. Các Services trong hệ thống.....31
3.1.2. Kiến trúc dùng cho hệ thống 32
3.1.3. Github Action 33
3.2. Xây dựng cơ sở dữ liệu.....35
3.2.1. Thiết kế ERD.....35
3.3. Thiết kế bảng và cơ sở dữ liệu.....36
3.3.1. AppleAuth Database.....36
3.3.2. AppleCart Database.....36
3.3.3. AppleCategory Database.....37
3.3.4. AppleInventory Database 38
3.3.5. AppleOrder Database 39
3.3.6. AppleProduct Database 41
3.3.7. ApplePromotion Database.....44
3.3.8. AppleUser Database 45
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU 48
4.1. Quy ước chung cho API 48
4.2. Auth Service 48
4.2.1. Register API (User) 48
4.2.2. VertifyOTP API (User) 49
4.2.3. ResendOTP API (User) 49
4.2.4. Login API (User).....49
4.2.5. RefreshToken API (User).....50
4.2.6. ResetPassword API (User) 50
4.2.7. ChangePassword API (User).....50
4.3. Cart Service 51
4.3.1. CreateCart API (User) 51
4.3.2. DeleteCart Item API (User).....51

4.3.3. GetCartByUserId API (User) 52
4.4. Category Service 53
4.4.1. CreateCategory API (Admin) 53
4.4.2. UpdateCategory API (Admin) 54
4.4.3. GetAllCategories API (Admin) 54
4.4.4. GetAllCategoriesActived API (User) 55
4.4.5. GetCategoryById API (Admin) 56
4.5. Inventory Service 57
4.5.1. GetAllInventories API (Admin) 57
4.5.2. GetInventoryById API (Admin) 57
4.6. Order Service 58
4.6.1. CreateOrder API (User) 58
4.6.2. ChangeStatus API (Admin) 59
4.6.3. GetAllOrders API (Admin) 60
4.6.4. GetOrderById API (Admin, User) 62
4.6.5. GetOrderByUserId API (User) 63
4.7. Product Service 65
4.7.1. CreateProduct API (Admin) 65
4.7.2. UpdateProduct API (Admin) 66
4.7.3. CreateProductImage API (Admin) 66
4.7.4. UpdateProductImage API (Admin) 67
4.7.5. DeleteProductImage API (Admin) 67
4.7.6. CreateColor API (Admin) 68
4.7.7. UpdateColor API (Admin) 68
4.7.8. DeleteColor API (Admin) 68
4.7.9. GetAllProduct API (Admin) 69
4.7.10. GetAllProductActived API (User) 70
4.7.11. GetProductById API (User) 71
4.7.12. GetProductByCategoryId API (User) 72
4.7.13. GetProductByName API (User) 73
4.8. Promotion Service 74
4.8.1. CreatePromotion API (Admin) 74

4.8.2. UpdatePromotion API (Admin) 75
4.8.3. DeletePromotion API (Admin) 76
4.8.4. GetAllPromotions API (Admin) 76
4.8.5. GetPromotionById API (Admin) 77
4.9. User Service.....77
4.9.1. UpdateProfileUser API (User) 77
4.9.2. CreateUserAddress API (User) 78
4.9.3. UpdateUserAddress API (User) 78
4.9.4. DeleteUserAddress API (User) 79
4.9.5. GetProfileUserById API (User) 80
4.9.6. GetAllAddressByUserId API (User).....80
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 82
5.1. Kết quả đạt được.....82
5.2. Kết quả chưa đạt được.....82
5.3. Hướng phát triển.....82
DANH MỤC TÀI LIỆU THAM KHẢO 83

DANH MỤC HÌNH ẢNH

Hình 2.1. Kiến trúc Clean Architecture....17
Hình 2.2. Các thành phần của ASP.NET Core....24
Hình 2.3. Cấu trúc của SQL Server....26
Hình 3.1. Các Services trong hệ thống.....31
Hình 3.2. Kiến trúc của hệ thống.....32
Hình 3.3. Tự động tạo Pull Request....33
Hình 3.4. Tự động build các Image....34
Hình 3.5. Mô hình Diagram....35
Hình 3.6. Bảng usertoken....36
Hình 3.7. Bảng cart....37
Hình 3.8. Bảng cart_item....37
Hình 3.9. Bảng category....38
Hình 3.10. Bảng inventory....39
Hình 3.11. Bảng order....40
Hình 3.12. Bảng order_item....41
Hình 3.13. Bảng product....42
Hình 3.14. Bảng product_image....43
Hình 3.15. Bảng color....43
Hình 3.16. Bảng product_color....44
Hình 3.17. Bảng promotion....45
Hình 3.18. Bảng user....46
Hình 3.19. Bảng useraddress....47

DANH MỤC BẢNG BIỂU

Bảng 4.1. Quy ước chung dữ liệu đầu ra 48
Bảng 4.2. Register Request 48
Bảng 4.3. VertifiOTP Request 49
Bảng 4.4. ResendOTP Request 49
Bảng 4.5. Login Request 49
Bảng 4.6. RefreshToken Request 50
Bảng 4.7. ResetPassword Request 50
Bảng 4.8. ChangePassword Request 51
Bảng 4.9. CreateCart Request 51
Bảng 4.10. CartItemDTO 51
Bảng 4.11. DeleteCart Request 52
Bảng 4.12. GetCartByUserId Request 52
Bảng 4.13. CartFullDTO 52
Bảng 4.14. CartItemDTO 52
Bảng 4.15. CreateCategory Request 53
Bảng 4.16. UpdateCategory Request 54
Bảng 4.17. Category 55
Bảng 4.18. Category 55
Bảng 4.19. GetCategoryById Request 56
Bảng 4.20. Category 56
Bảng 4.21. Inventory 57
Bảng 4.22. GetInventoryById Request 58
Bảng 4.23. Inventory 58
Bảng 4.24. CreateOrder Request 58
Bảng 4.25. OrderItemDTO 59
Bảng 4.26. Change Status Request 59
Bảng 4.27. OrderFullDTO 60
Bảng 4.28. UserDTO 61
Bảng 4.29. UserAddressDTO 61
Bảng 4.30. PromotionDTO 61
Bảng 4.31. OrderItemDTO 61

Bảng 4.32. GetOrderById Request....62
Bảng 4.33. OrderFullDTO....62
Bảng 4.34. UserDTO....62
Bảng 4.35. PromotionDTO....63
Bảng 4.36. OrderItemDTO....63
Bảng 4.37. GetOrderByUserId Request....63
Bảng 4.38. OrderFullDTO....64
Bảng 4.39. UserDTO....64
Bảng 4.40. UserAddressDTO....64
Bảng 4.41. PromotionDTO....64
Bảng 4.42. OrderItemDTO....65
Bảng 4.43. CreateProduct Request....65
Bảng 4.44. UpdateProduct Request....66
Bảng 4.45. CreateProductImage Request....67
Bảng 4.46. UpdateProductImage Request....67
Bảng 4.47. DeleteProductImage Request....68
Bảng 4.48. CreateColor Request....68
Bảng 4.49. UpdateColor Request....68
Bảng 4.50. DeleteColor Request....69
Bảng 4.51. ProductFullDTO....69
Bảng 4.52. ColorDTO....69
Bảng 4.53. ProductFullDTO....70
Bảng 4.54. ColorDTO....70
Bảng 4.55. ProductImageDTO....71
Bảng 4.56. GetProductById Request....71
Bảng 4.57. ProductFullDTO....71
Bảng 4.58. ColorDTO....72
Bảng 4.59. ProductImageDTO....72
Bảng 4.60. GetProductByCategoryId Request....72
Bảng 4.61. ProductFullDTO....72
Bảng 4.62. ColorDTO....73
Bảng 4.63. ProductImageDTO....73

Bảng 4.64. GetProductByName Request 73
Bảng 4.65. ProductFullDTO.....73
Bảng 4.66. ColorDTO 74
Bảng 4.67. ProductImageDTO 74
Bảng 4.68. CreatePromotion Request 74
Bảng 4.69. UpdatePromotion Request 75
Bảng 4.70. DeletePromotion Request 76
Bảng 4.71. PromotionDTO.....76
Bảng 4.72. GetPromotionById Request 77
Bảng 4.73. PromotionDTO.....77
Bảng 4.74. UpdateProfileUser Request.....78
Bảng 4.75. CreateUserAddress Request.....78
Bảng 4.76. UpdateUserAddress Request.....79
Bảng 4.77. DeleteUserAddress Request.....79
Bảng 4.78. GetProfileUserById Request.....80
Bảng 4.79. UserDTO 80
Bảng 4.80. GetAllAddressByUserId Request 80
Bảng 4.81. UserAddress 80

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

1. Vấn đề nghiên cứu:

- Xây dựng kiến trúc phần mềm Clean Architecture với Microservice để tối ưu hóa quản lý và vận hành hệ thống.
- Tích hợp chức năng đăng nhập, đăng ký cho khách hàng.
- Cung cấp các API để hỗ trợ tìm kiếm, lọc sản phẩm theo nhu cầu khách hàng.
- Phát triển các dịch vụ quản lý sản phẩm, người dùng,… dành cho quản trị viên.

2. Hướng tiếp cận:

- Nghiên cứu mô hình Clean Architecture và kiến trúc Microservice để đảm bảo khả năng mở rộng và bảo trì dễ dàng.
- Xác định các chức năng cần thiết và phân tách chúng thành các module độc lập.
- Tập trung vào việc thiết kế API với các tiêu chuẩn RESTful để hỗ trợ các tính năng cốt lõi như quản lý người dùng, sản phẩm,….

3. Cách giải quyết:

- Xây dựng các service độc lập với các chức năng chính, bao gồm quản lý người dùng, sản phẩm, tìm kiếm,….
- Sử dụng công nghệ backend tiên tiến như .NET Core để triển khai kiến trúc Microservice.
- Áp dụng các phương pháp phát triển phần mềm hiện đại như CI/CD để nâng cao hiệu quả triển khai và vận hành hệ thống.

4. Kết quả đạt được:

- Một hệ thống backend tương đối hoàn chỉnh với kiến trúc Clean Architecture được triển khai theo mô hình Microservice.
- Cung cấp các API mạnh mẽ và dễ sử dụng để hỗ trợ các tính năng cần thiết của hệ thống bán sản phẩm công nghệ.

MỞ ĐẦU

1. Lý do chọn đề tài

Như chúng ta đã biết, các sản phẩm của Apple luôn thu hút sự quan tâm lớn từ thị trường, đặc biệt là những người yêu công nghệ và thiết kế. Hệ sinh thái của Apple bao gồm các sản phẩm nổi bật như IPhone, IPad, Macbook, Apple Watch và AirPods, vốn luôn được đánh giá cao về chất lượng và tính sáng tạo. Việc xây dựng một hệ thống bán hàng cho các sản phẩm này không chỉ tạo điều kiện để hiểu sâu hơn về thị trường mà còn là cơ hội để áp dụng các kiến thức về kiến trúc phần mềm hiện đại.

Với sự phát triển nhanh chóng của công nghệ, việc áp dụng các mô hình kiến trúc tiên tiến như **Clean Architecture** và triển khai các **Microservice** giúp hệ thống dễ dàng mở rộng, bảo trì và nâng cao hiệu suất. Chính vì vậy, em đã chọn đề tài **"Nghiên cứu kiến trúc Clean Architecture với Microservice cho hệ thống bán sản phẩm công nghệ của Apple"** để phát triển kỹ năng chuyên môn cũng như tạo ra một hệ thống backend hiệu quả.

2. Mục tiêu nghiên cứu

Mục tiêu của đề tài là xây dựng một hệ thống backend áp dụng kiến trúc Clean Architecture và mô hình Microservice, đáp ứng tương đối đầy đủ các tính năng cơ bản của một hệ thống bán hàng, bao gồm:

- Quản lý sản phẩm, người dùng và các giao dịch mua bán.
- Cung cấp API mạnh mẽ và dễ sử dụng cho các chức năng như tìm kiếm, lọc sản phẩm, và quản trị hệ thống.
- Đảm bảo tính bảo mật, hiệu suất cao, và khả năng mở rộng.
- Hệ thống cần được triển khai với quy trình phát triển hiện đại, hỗ trợ CI/CD và đảm bảo tính tương thích với nhiều công cụ và nền tảng khác nhau.

3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu chủ yếu là các sản phẩm công nghệ của Apple, bao gồm: IPhone, IPad, Macbook, Apple Watch và Airpods.

Phạm vi nghiên cứu: Tập trung vào xây dựng tầng backend cho hệ thống, bao gồm:

- Thiết kế và triển khai kiến trúc Clean Architecture và Microservice.
- Phát triển các API RESTful với sự hỗ trợ của Swagger và Postman để kiểm thử và tài liệu hóa.
- Tích hợp hệ thống với Docker để đóng gói và triển khai dễ dàng.
- Sử dụng GitHub Action để thiết lập CI/CD, tối ưu hóa quy trình phát triển.

4. Phương pháp nghiên cứu

Hệ thống được phát triển dựa trên các công nghệ tiên tiến:

- Nền tảng: Sử dụng .NET Core với ngôn ngữ C# để xây dựng backend, đảm bảo tính linh hoạt và hiệu suất cao.
- Cơ sở dữ liệu: SQL Server để lưu trữ và quản lý dữ liệu, tối ưu hóa hiệu năng truy vấn.
- Đóng gói và triển khai: Docker được sử dụng để tạo môi trường triển khai nhất quán và dễ dàng mở rộng.
- Quy trình phát triển: CI/CD được tích hợp với GitHub Actions để tự động hóa quá trình kiểm thử và triển khai.
- Kiểm thử và tài liệu hóa: Sử dụng Swagger để tài liệu hóa API và Postman để kiểm thử chức năng.

Phương pháp nghiên cứu bao gồm các giai đoạn: thiết kế kiến trúc, phát triển các Microservice, kiểm thử chức năng API, và triển khai toàn bộ hệ thống lên môi trường thực tế.

CHƯƠNG 1: TỔNG QUAN

1.1. Đặt vấn đề

Trong thời buổi xã hội đang không ngừng phát triển, xu hướng thương mại điện tử ngày càng phát triển. Mọi việc giờ đây thật đơn giản, chỉ cần có một chiếc máy tính hay chỉ với một chiếc điện thoại thông minh có kết nối internet, việc mua bán, trao đổi thương mại trở nên thật dễ dàng hơn bao giờ hết với tất cả mọi người chỉ với một vài cái bấm chuột.

Thương mại điện tử không chỉ giúp xóa bỏ rào cản về không gian và thời gian mà còn tạo điều kiện để các doanh nghiệp tối ưu hóa quy trình bán hàng, cải thiện hiệu quả kinh doanh. Đặc biệt, các sản phẩm công nghệ của Apple luôn được săn đón bởi sự độc đáo, chất lượng cao và thiết kế tinh tế, điều này tạo ra cơ hội lớn để xây dựng một hệ thống hỗ trợ kinh doanh các sản phẩm này thông qua nền tảng số.

Bên cạnh đó, việc xây dựng một hệ thống backend với các công nghệ tiên tiến như **.NET Core**, **SQL Server**, kết hợp với các công cụ **Docker**, **GitHub Actions** và tài liệu hóa API bằng **Swagger**, không chỉ đáp ứng nhu cầu kinh doanh mà còn tối ưu hóa hiệu suất, tính bảo mật và khả năng mở rộng của hệ thống.

Trên cơ sở đó, đề tài **"Nghiên cứu kiến trúc Clean Architecture với Microservice cho hệ thống bán sản phẩm công nghệ của Apple"** được thực hiện nhằm giải quyết nhu cầu phát triển một hệ thống backend hiệu quả, hiện đại, đáp ứng yêu cầu kinh doanh sản phẩm của Apple đến khách hàng trên toàn quốc và quốc tế.

1.2. Mục đích

Mục đích của đề tài là xây dựng một hệ thống backend cho website bán hàng công nghệ của Apple, đảm bảo các yêu cầu chính sau:

- Tính năng hướng đến người dùng: Cung cấp các API dễ sử dụng, tương thích với các thiết bị thông minh như điện thoại, máy tính bảng và laptop.
- Tối ưu hiệu năng: Hệ thống cần đảm bảo tốc độ xử lý nhanh, chính xác, đáp ứng đồng thời nhu cầu của cả người mua và người bán.
- Chức năng cho người dùng: Hỗ trợ đăng ký, đăng nhập tài khoản, quản lý tài khoản, giỏ hàng, đơn hàng.
- Chức năng cho quản trị viên: Quản lý sản phẩm, danh mục, đơn hàng,….

- Công nghệ tiên tiến: Ứng dụng các công nghệ như .NET Core để xây dựng hệ thống backend, SQL Server để quản lý dữ liệu, Docker để triển khai nhanh chóng, và sử dụng GitHub Actions để tự động hóa quy trình kiểm thử và triển khai.

Hệ thống không chỉ đáp ứng đầy đủ chức năng cơ bản của một website thương mại điện tử mà còn đảm bảo tính bảo mật, khả năng mở rộng và tương tác hiệu quả với khách hàng.

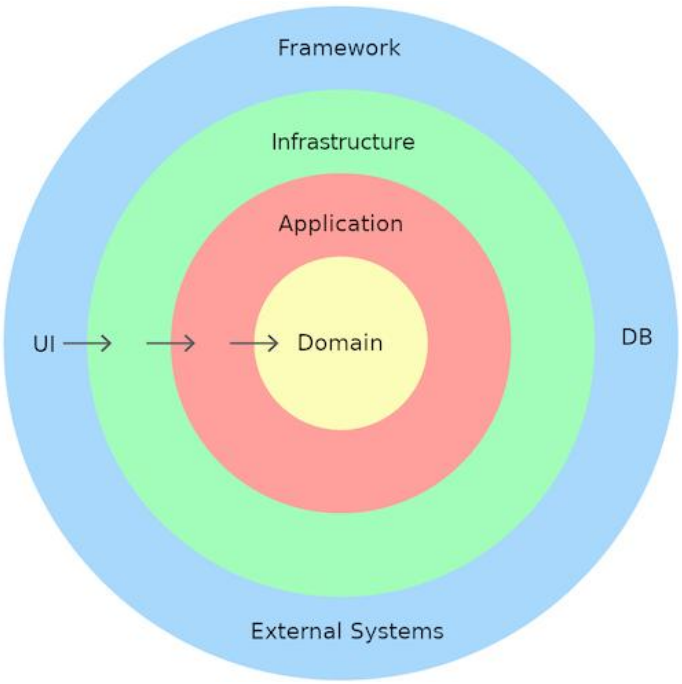
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Kiến trúc Clean Architecture

2.1.1. Tổng quan

Clean Architecture được xây dựng dựa trên tư tưởng "độc lập" kết hợp với các nguyên lý thiết kế hướng đối tượng (đại diện tiêu biểu là Dependency Inversion). Độc lập ở đây nghĩa là việc project không bị phụ thuộc vào framework và các công cụ sử dụng trong quá trình kiểm thử.

Kiến trúc của Clean Architecture thường chia thành 4 layer với một quy tắc phụ thuộc. Các layer bên trong không phụ thuộc bất kỳ điều gì về các layer bên ngoài. Điều này có nghĩa là nó có quan hệ phụ thuộc nên "hướng" vào bên trong.



Hình 2.1. Kiến trúc Clean Architecture

Domain/Entities: Chứa các đối tượng nghiệp vụ cốt lõi.

Application/Use case: Chứa logic nghiệp vụ, thể hiện các chức năng của hệ thống.

Interface Adapter: Là nơi chuyển đổi dữ liệu giữa các lớp.

Framework and Drivers: Là nơi chưa các công cụ, framework.

2.1.2. Ưu điểm

Chia để trị rất hiệu quả trong ứng dụng lớn: Trong Clean Architecture thì code tầng nào thì ở đúng tầng nấy. Hạn chế được việc "code ở đâu cũng là code, chạy được là được". Nếu làm tốt được các bài toán nhỏ thì không có bài toán lớn nào không giải quyết được.

Dễ bảo trì và mở rộng: Việc tìm kiếm bug và lỗi logic sẽ trở nên dễ dàng và nhanh hơn, file code sẽ không nhiều vì chỉ làm đúng việc của nó. Vì các tầng độc lập với nhau thông qua các Interfaces nên việc mở rộng hoặc thay đổi các tầng sẽ không ảnh hưởng tới nhau. Điều này hạn chế các breaking change cũng như phải viết lại code (refactoring).

Dễ kiểm thử: Các logic business của các tầng trong Clean Architecture chính là các Unit Test cần được kiểm thử rất cẩn thận. Vì sự độc lập thông qua Interfaces nên các mock test rất dễ triển khai. Việc này được thực hiện thông qua implement lại để coverage được tất cả các trường hợp.

2.1.3. Nhược điểm

Cồng kềnh và phức tạp: Điều dễ thấy nhất là Clean Architecture không hề dễ sử dụng, phải viết nhiều lớp (class/object) hơn. Trong trường hợp ứng dụng của bạn quá đơn giản, ít tính năng, vòng đời ngắn thì chọn lựa kiến trúc này có thể mang lại những rắc rối không cần thiết.

Tính trừu tượng cao: Vấn đề này gọi là indirect. Trừu tượng càng cao thì tiện cho các developers nhưng sẽ gây ảnh hưởng không nhỏ tới tốc độ thực thi (performance). Ngoài ra cũng không thể code nhanh, vội vã "mì ăn liền" được mà phải tạo đủ các Interfaces.

Khó tuyển người: Sử dụng Clean Architecture sẽ cần tuyển dụng developer thấu hiểu về kiến trúc này. Nguyên tắc Dependency Inversion rất dễ bị xâm phạm vì sự hạn chế kiến thức, sự bất cẩn hoặc vì thời gian cần triển khai tính năng quá ít.

2.2. Kiến trúc Microservice

Kiến trúc Microservices là một phương pháp phát triển ứng dụng trong đó ứng dụng được chia thành các dịch vụ nhỏ, độc lập, mỗi dịch vụ thực hiện một chức năng

cụ thể và giao tiếp với nhau thông qua các giao thức mạng. Mỗi microservice có thể được triển khai, phát triển và duy trì độc lập.

2.2.1. Đặc điểm

Phân tách theo chức năng: Mỗi microservice thực hiện một chức năng riêng biệt của hệ thống, ví dụ như quản lý người dùng, xử lý thanh toán, hay quản lý đơn hàng. Điều này giúp giảm độ phức tạp và dễ dàng bảo trì.

Độc lập và tách biệt: Các microservices hoạt động độc lập, có thể triển khai, phát triển và bảo trì riêng biệt mà không ảnh hưởng đến các dịch vụ khác trong hệ thống. Các dịch vụ này có thể chạy trên các máy chủ hoặc container khác nhau.

Giao tiếp qua API/Message Broker: Các microservices giao tiếp với nhau thông qua API, thường là RESTful API hoặc gRPC, ngoài ra còn có thể giao tiếp với nhau qua Message Broker như RabbitMQ hay Kafka. Điều này giúp các dịch vụ có thể tương tác mà không cần chia sẻ thông tin trực tiếp, từ đó giảm sự phụ thuộc vào nhau.

Mã nguồn và cơ sở dữ liệu riêng biệt: Mỗi dịch vụ có mã nguồn và cơ sở dữ liệu riêng. Việc này giúp đảm bảo rằng mỗi microservice có thể sử dụng loại cơ sở dữ liệu phù hợp và không phụ thuộc vào các dịch vụ khác.

Khả năng mở rộng: Các microservices có thể mở rộng độc lập. Khi một dịch vụ gặp phải tải cao, chỉ cần mở rộng dịch vụ đó mà không phải mở rộng toàn bộ hệ thống.

Tự động triển khai và quản lý: Microservices hỗ trợ các công cụ CI/CD (Continuous Integration/Continuous Deployment) giúp tự động hóa quá trình kiểm thử, xây dựng và triển khai các dịch vụ.

Tính chịu lỗi cao: Với tính phân tán của các dịch vụ, khi một dịch vụ gặp sự cố, các dịch vụ khác vẫn có thể hoạt động bình thường. Điều này giúp đảm bảo tính khả dụng của toàn bộ hệ thống.

2.2.2. Thành phần

Các Microservices độc lập: Mỗi microservice thực hiện một chức năng cụ thể, chẳng hạn như quản lý người dùng, xử lý đơn hàng, thanh toán, hay thông báo.

API Gateway: Là điểm vào duy nhất cho các yêu cầu từ client. API Gateway giúp phân phối các yêu cầu đến đúng microservice, xử lý bảo mật, load balancing, và logging.

Service Discovery: Giúp các microservices tự động phát hiện và giao tiếp với nhau mà không cần phải biết trước về địa chỉ của các dịch vụ. Các công cụ như Consul, Eureka hỗ trợ tính năng này.

Message Broker: Được sử dụng để hỗ trợ giao tiếp bất đồng bộ giữa các microservices. Ví dụ như Kafka, RabbitMQ giúp các dịch vụ gửi và nhận thông điệp mà không cần đồng bộ thời gian.

Cơ sở dữ liệu phân tán: Mỗi microservice có thể có cơ sở dữ liệu riêng, điều này giúp giảm sự phụ thuộc vào các dịch vụ khác và tối ưu hóa hiệu suất.

Quản lý giao dịch: Khi một giao dịch liên quan đến nhiều microservices, cần có các giải pháp như Saga pattern hoặc Event-Driven Architecture để đảm bảo tính toàn vẹn của giao dịch trong môi trường phân tán.

2.2.3. Ưu điểm

Linh hoạt và dễ bảo trì: Việc tách các chức năng thành các microservices giúp giảm độ phức tạp, dễ dàng bảo trì, thay đổi hoặc thay thế một dịch vụ mà không ảnh hưởng đến các dịch vụ khác.

Khả năng mở rộng: Microservices cho phép mở rộng độc lập từng dịch vụ khi cần thiết, giúp tối ưu hóa tài nguyên và giảm chi phí.

Triển khai độc lập: Các microservices có thể được triển khai độc lập mà không làm gián đoạn hệ thống, giúp giảm downtime và cải thiện hiệu quả triển khai.

Tính khả dụng cao: Vì mỗi dịch vụ có thể hoạt động độc lập, nếu một dịch vụ gặp sự cố, các dịch vụ còn lại vẫn tiếp tục hoạt động bình thường.

Lựa chọn công nghệ linh hoạt: Các microservices có thể sử dụng các công nghệ khác nhau (ngôn ngữ lập trình, cơ sở dữ liệu, framework) phù hợp với yêu cầu của từng dịch vụ.

2.3. RESTful API

RESTful API là một kiểu API dựa trên các nguyên tắc của REST (Representational State Transfer), sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để quản lý tài nguyên.

Việc sử dụng RESTful API giúp hệ thống dễ dàng giao tiếp với các dịch vụ khác thông qua HTTP, tạo ra các điểm truy cập dễ hiểu và thống nhất cho các thành phần của hệ thống, đảm bảo tính độc lập giữa các phần giao tiếp

2.3.1. Nguyên tắc thiết kế của REST

REST là một kiểu kiến trúc, không phải là một giao thức hay công nghệ cụ thể. Các nguyên tắc của REST giúp xây dựng các API dễ sử dụng và dễ bảo trì. Một RESTful API cần tuân theo các nguyên tắc chính sau:

- Client-Server Architecture (Kiến trúc máy khách - máy chủ): Hệ thống được chia thành hai thành phần chính là máy khách (client) và máy chủ (server). Máy khách chịu trách nhiệm thực hiện yêu cầu và hiển thị dữ liệu, trong khi máy chủ cung cấp dịch vụ xử lý và lưu trữ dữ liệu.
- Stateless (Không trạng thái): RESTful API phải không có trạng thái (stateless), tức là mỗi yêu cầu từ máy khách phải chứa đủ thông tin cần thiết để máy chủ xử lý mà không cần dựa vào các thông tin từ các yêu cầu trước đó. Điều này giúp API dễ dàng mở rộng và linh hoạt hơn trong xử lý các yêu cầu song song.
- Uniform Interface (Giao diện thống nhất): Một trong những nguyên tắc quan trọng của REST là cung cấp giao diện thống nhất. Điều này có nghĩa là tất cả các tài nguyên phải được định dạng và truy cập thông qua một giao diện chung, với các quy ước cụ thể, ví dụ như URL để định danh tài nguyên, và sử dụng các phương thức HTTP để thao tác.
- Cacheable (Có khả năng lưu trữ): RESTful API phải hỗ trợ việc lưu trữ tạm (cache) để giảm tải máy chủ và tăng hiệu suất của hệ thống. Máy khách hoặc proxy có thể lưu trữ các phản hồi để tái sử dụng chúng trong những yêu cầu sau, nếu dữ liệu không thay đổi.
- Layered System (Hệ thống phân lớp): Kiến trúc REST có thể phân lớp thành nhiều tầng khác nhau (ví dụ như lớp proxy, lớp bảo mật, lớp xử lý ứng dụng,

v.v.). Việc sử dụng hệ thống phân lớp giúp tăng tính bảo mật và khả năng mở rộng của hệ thống.

- Code on Demand (Tùy chọn mã theo yêu cầu): Trong một số trường hợp, máy chủ có thể cung cấp mã thực thi (ví dụ như JavaScript) cho máy khách, cho phép máy khách có khả năng tự thực hiện các thao tác nhất định mà không cần yêu cầu máy chủ

2.3.2. Phương thức HTTP trong RESTful API

- **GET**: Lấy dữ liệu từ máy chủ. Thường được sử dụng để truy xuất tài nguyên mà không làm thay đổi trạng thái của tài nguyên.
- **POST**: Tạo mới một tài nguyên trên máy chủ. Thường được sử dụng để gửi dữ liệu tới máy chủ để tạo tài nguyên mới.
- **PUT**: Cập nhật toàn bộ thông tin của một tài nguyên. Thường được sử dụng để thay thế hoặc cập nhật toàn bộ thông tin của tài nguyên.
- **PATCH**: Cập nhật một phần thông tin của tài nguyên. Thường được sử dụng để thay đổi một số thuộc tính của tài nguyên thay vì toàn bộ.
- **DELETE**: Xóa một tài nguyên. Thường được sử dụng để xóa tài nguyên khỏi máy chủ.
- **OPTIONS:** Lấy thông tin về các phương thức HTTP được hỗ trợ bởi một tài nguyên. Thường được sử dụng để kiểm tra quyền truy cập hoặc hỗ trợ CORS (Cross-Origin Resource Sharing).
- **HEAD:** Tương tự như GET nhưng không trả về phần thân (body) của phản hồi, chỉ trả về tiêu đề (header). Thường được sử dụng để kiểm tra sự tồn tại của tài nguyên hoặc kiểm tra metadata mà không cần lấy toàn bộ dữ liệu.

2.3.3. Các quy ước thiết kế RESTful API

Một số quy ước cần tuân theo khi thiết kế RESTful API:

Đặt tên tài nguyên:

Nên sử dụng số nhiều cho các tài nguyên để biểu thị danh sách, ví dụ /users thay vì /user.

URI chỉ nên biểu thị tài nguyên, không nên biểu thị hành động. Ví dụ: thay vì /getUser, ta sử dụng GET /users/{id}.

Sử dụng mã trạng thái HTTP: RESTful API nên trả về mã trạng thái HTTP để biểu thị kết quả của yêu cầu:

- **200 OK**: Thành công.
- **201 Created**: Tài nguyên mới đã được tạo thành công.
- **400 Bad Request**: Lỗi từ phía người dùng, chẳng hạn gửi sai tham số.
- **401 Unauthorized**: Người dùng không có quyền truy cập.
- **403 Forbidden:** Cấm truy cập vào một chức năng nào đó trong hệ thống.
- **404 Not Found**: Không tìm thấy tài nguyên yêu cầu.
- **500 Internal Server Error**: Lỗi máy chủ.

2.4. Ngôn ngữ Backend

2.4.1. Ngôn ngữ C Sharp (C#)

C# là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư dẫn đầu là Andres Hejlsberg và Scott Wiltamuth của Microsoft vào năm 2000. Nó là một ngôn ngữ được xây dựng dựa trên nền tảng của C++ và Java. Nhờ vậy mà chúng hầu như đều có cấu trúc chương trình khá giống nhau. Một số cải tiến của nó đó chính là cấu trúc được rút gọn sao cho dễ nhớ và đơn giản hơn.

Ngôn ngữ lập trình C# được các chuyên gia nhận xét là loại ngôn ngữ thuần hướng các đối tượng. So với những loại ngôn ngữ lập trình khác thì C# sử dụng lượng từ khóa ít hơn rất nhiều. Nhờ đặc điểm này mà các lập trình viên đều thuận tiện hơn rất nhiều cho việc xây dựng lên các đối tượng dành riêng cho mình

C# được thiết kế cho các ngôn ngữ chung cơ sở hạ tầng (Common Language Infrastructure – CLI), trong đó bao gồm các mã (Executable Code) và môi trường thực thi (Runtime Environment) cho phép sử dụng các ngôn ngữ cao cấp khác nhau trên đa nền tảng máy tính và kiến trúc khác nhau.

2.4.2. ASP.NET CORE

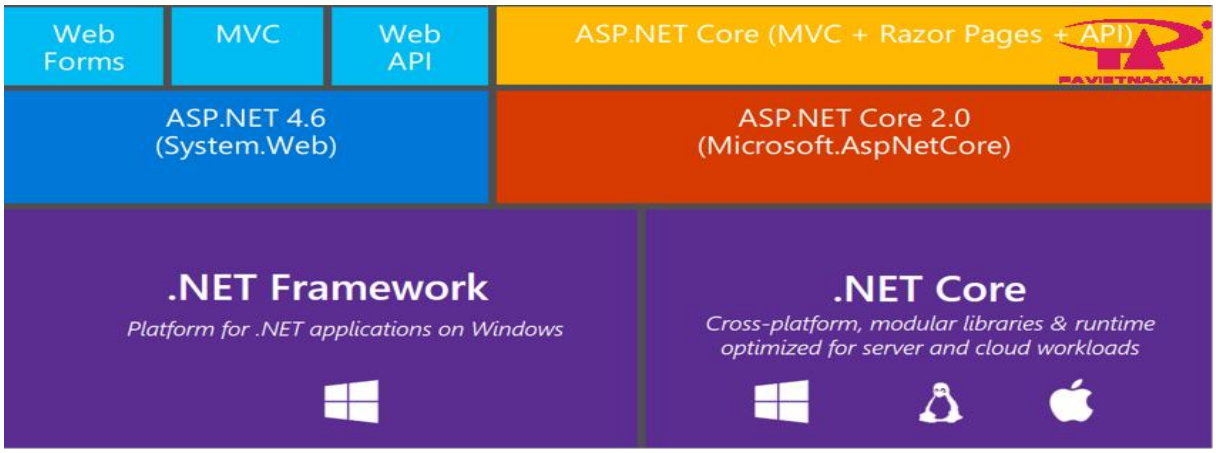
ASP.NET Core là một open-source mới và framework đa nền tảng (cross-platform) cho việc xây dựng những ứng dụng hiện tại dựa trên kết nối đám mây, giống

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

ASP.NET Core bao gồm các thành phần theo hướng module nhằm tối thiểu tài nguyên và chi phí phát triển, như vậy bạn giữ lại được sự mềm giẻo trong việc xây dựng giải pháp của bạn. Bạn có thể phát triển và chạy những ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux.

Các thành phần chính của nền tảng ASP.NET cơ bản gồm:

- Ngôn ngữ: ASP.NET sử dụng nhiều ngôn ngữ lập trình khác nhau như VB.NET và C#.
- Thư viện: ASP.NET có bộ thư viện chuẩn bao gồm các giao diện, các lớp và kiểu giá trị. Bộ thư viện này có thể sử dụng lại cho quá trình phát triển ASP.NET và xây dựng các chức năng của hệ thống.
- Thời gian chạy ngôn ngữ chung (CLR): CLR – Common Language Runtime được sử dụng để thực hiện các hoạt động mã. Các hoạt động này sẽ thực hiện xử lý các ngoại lệ và thu gom rác. [10]



Hình 2.2. Các thành phần của ASP.NET Core

2.4.3. Entity Framework Core

Entity Framework Core là phiên bản mới của Entity Framework sau EF 6.x. Nó là mã nguồn mở, nhẹ, có thể mở rộng và là phiên bản đa nền tảng của công nghệ truy cập dữ liệu Entity Framework.

Entity Framework là một framework Object/Relational Mapping (O/RM - ánh xạ quan hệ/đối tượng). Đây là một cải tiến của ADO.NET, cung cấp cho các nhà phát triển một cơ chế tự động để truy cập và lưu trữ dữ liệu trong cơ sở dữ liệu.

EF Core được dự định sẽ sử dụng với các ứng dụng .NET Core. Tuy nhiên, nó cũng có thể được sử dụng với các ứng dụng dựa trên .NET Framework 4.5+ tiêu chuẩn.

2.5. SQL Server

2.5.1. Khái niệm

SQL server là một dạng hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System - Viết tắt là RDBMS). Nó được phát triển bởi gã khổng lồ trong làng công nghệ Microsoft vào năm 1989 và vẫn được sử dụng rộng rãi đến ngày nay.

Có thể khi thấy chữ server thì nhiều bạn sẽ có những hiểu nhầm ở đây. Chúng ta sẽ cùng làm rõ về khái niệm SQL server bằng cách xem qua những đặc tính dưới đây của nó nhé:

- Thứ nhất, server là một thiết bị phần cứng nhưng SQL server thì về bản chất nó là một sản phẩm phần mềm. Nó được các kỹ sư của Microsoft xây dựng và phát triển tử cách đây gần 30 năm rồi. Vì là sản phẩm phần mềm nên nó được cài trên các thiết bị phần cứng như server.
- Thứ hai, nó có chức năng chính là lưu trữ và truy xuất dữ liệu theo yêu cầu của các ứng dụng phần mềm khác. Chúng ta sẽ lưu trữ dữ liệu vào đó và sử dụng các câu lệnh để tìm kiếm dữ liệu khi cần.
- Thứ ba, nó sử dụng câu lệnh SQL (Transact-SQL) để trao đổi dữ liệu giữa máy khách (máy Client) và máy cài SQL Server.

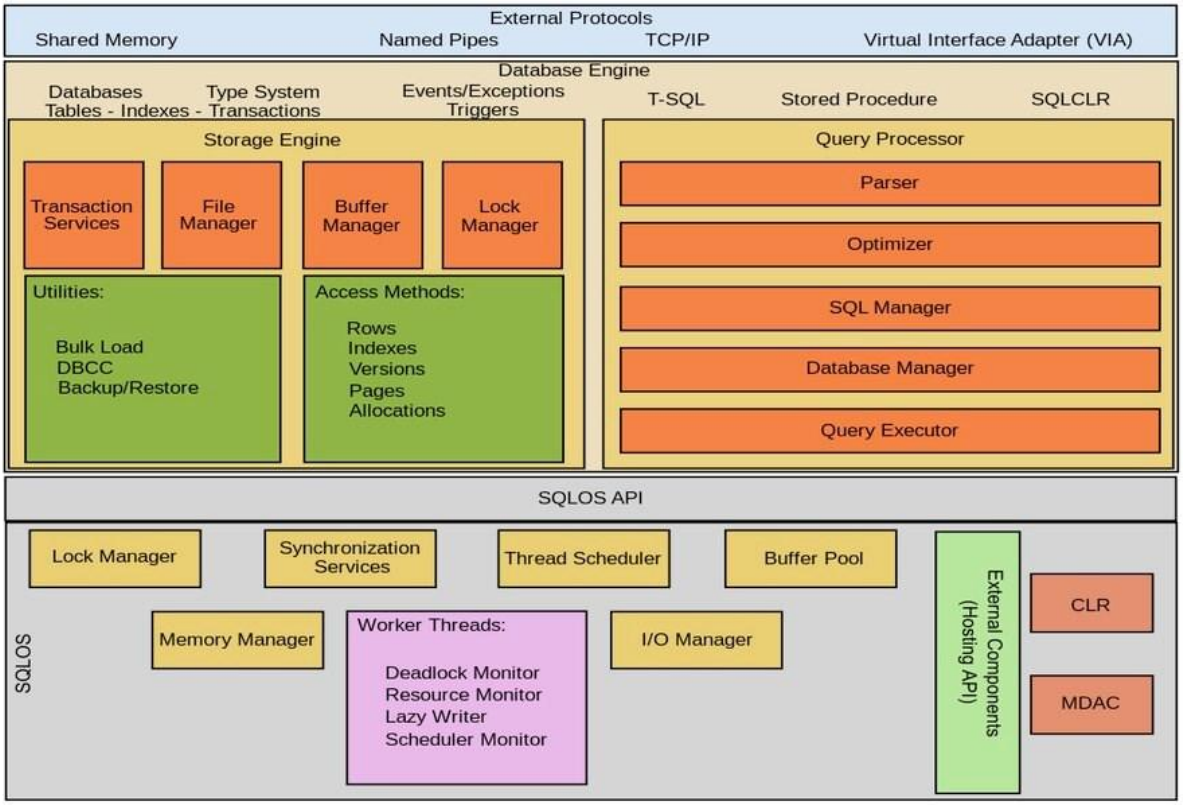
2.5.2. Cấu trúc

SQLOS: là viết tắt của hệ điều hành SQL server. Đây là tầng cuối cùng trong kiến trúc tổng thể của SQL server. Tại đây sẽ chịu trách nhiệm xử lý các nhiệm vụ như quản lý bộ nhớ, lên lịch nhiệm vụ, khoá dữ liệu nhằm tránh các xung đột ngoài ý muốn có thể xảy ra mỗi khi thực hiện các thao tác cập nhật.

Database engine: đây là một công cụ có chức năng quản lý việc lưu trữ, xử lý và bảo mật dữ liệu. Trong đây sẽ bao gồm rất nhiều các công cụ khác nhau như một

công cụ lưu trữ quản lý các tệp, bảng, trang, chỉ mục, bộ đệm dữ liệu và giao dịch cơ sở dữ liệu.

External protocol: đây là các giao thức được sử dụng để giao tiếp với Database engine. Nó bao gồm TCP/IP hay VIA (Virtual Interface Adapter),...



Hình 2.3. Cấu trúc của SQL Server

2.6. MediatR và CQRS Pattern

MediatR là một thư viện giúp thực hiện CQRS (Command Query Responsibility Segregation) và giảm thiểu sự phụ thuộc cứng giữa các lớp trong hệ thống bằng cách sử dụng Mediator Pattern. Trong Clean Architecture, MediatR được sử dụng để tách biệt rõ ràng giữa các hành động và truy vấn, từ đó giúp hệ thống dễ dàng mở rộng và bảo trì.

Một số đặc điểm chính của MediatR:

- **CQRS Pattern:** CQRS là một mẫu kiến trúc tách biệt giữa Command (các lệnh để thay đổi dữ liệu) và Query (các truy vấn để lấy dữ liệu). MediatR giúp triển khai CQRS một cách dễ dàng bằng cách chia các lớp xử lý nghiệp vụ thành các lớp CommandHandler và QueryHandler. Điều này giúp hệ thống có tính linh hoạt cao hơn, đặc biệt khi mở rộng các yêu cầu nghiệp vụ.

- **Giảm sự phụ thuộc giữa các lớp:** MediatR cho phép các lớp có thể giao tiếp với nhau thông qua Mediator mà không cần phải tham chiếu trực tiếp đến nhau. Các thành phần chỉ cần đăng ký hoặc lắng nghe các lệnh (command) hoặc truy vấn (query) mà chúng quan tâm. Điều này giúp hệ thống ít bị thay đổi hơn khi cần bổ sung hoặc điều chỉnh các chức năng.
- **Pipeline Behaviors:** MediatR cũng hỗ trợ khái niệm Pipeline Behaviors, giúp chèn các logic xử lý trước hoặc sau khi thực hiện các Handler. Ví dụ, có thể thêm các chức năng như logging, validation, hoặc authorization cho các lệnh và truy vấn mà không cần thay đổi code xử lý chính.

2.7. Docker

2.7.1. Khái niệm

Docker là một nền tảng mã nguồn mở giúp tự động hóa việc triển khai ứng dụng dưới dạng các container, đảm bảo các ứng dụng chạy nhất quán trong bất kỳ môi trường nào. Docker sử dụng công nghệ container hóa, cho phép đóng gói mọi thứ cần thiết để chạy ứng dụng vào một đơn vị nhỏ gọi là "container" (bao gồm mã nguồn, thư viện, và các phụ thuộc).

Các thành phần chính trong Docker:

Docker Engine: Là phần mềm chạy trên hệ điều hành và quản lý các container. Docker Engine bao gồm:

- **Docker Daemon** (docker service): Chạy nền, chịu trách nhiệm quản lý các container.
- **Docker CLI**: Giao diện dòng lệnh, cho phép người dùng tương tác với Docker Daemon.
- **Docker API**: Cung cấp các phương thức để tương tác với Docker thông qua HTTP.

Docker Container: Là môi trường cách ly mà ứng dụng chạy trong đó. Mỗi container có thể chứa một ứng dụng và các phụ thuộc của nó (thư viện, môi trường runtime, v.v.). Containers chạy trên cùng một hệ điều hành, nhưng tách biệt với nhau.

Dockerfile: Là một tập tin văn bản chứa các hướng dẫn để tạo Docker image. Dockerfile xác định môi trường và cách cài đặt các phần mềm cần thiết trong container.

Docker Image: Là mẫu (template) để tạo ra containers. Mỗi image có thể chứa toàn bộ hệ điều hành, phần mềm, ứng dụng, cấu hình... Image là phần không thay đổi khi container được tạo ra từ nó.

Docker Hub: Là kho lưu trữ chứa các Docker image. Người dùng có thể tải xuống các image đã có sẵn hoặc tải lên các image tùy chỉnh.

2.7.2. Lợi ích

- Start và stop các container trong vài giây.
- Khởi chạy docker trên nhiều hệ thống.
- Dễ dàng thiết lập môi trường làm việc. Chỉ cần config 1 lần duy nhất và không bao giờ phải cài đặt lại các dependencies.

2.8. Postman

2.8.1. Khái niệm

Postman là một công cụ phổ biến và mạnh mẽ dành cho việc phát triển và kiểm thử API (Application Programming Interface). Nó giúp các lập trình viên và kỹ sư phần mềm dễ dàng kiểm tra, phát triển và tự động hóa các cuộc gọi API, đồng thời cung cấp một giao diện người dùng trực quan để thực hiện các yêu cầu HTTP và xử lý các phản hồi một cách nhanh chóng. Postman hỗ trợ nhiều tính năng, từ việc gửi các yêu cầu đơn giản đến việc kiểm tra, tài liệu hóa và tự động hóa các kiểm thử API.

2.8.2. Tính năng

Gửi các yêu cầu HTTP: Postman hỗ trợ tất cả các phương thức HTTP phổ biến như GET, POST, PUT, DELETE, PATCH, OPTIONS, và HEAD. Có thể dễ dàng cấu hình các tiêu đề HTTP, tham số truy vấn, dữ liệu đầu vào và nhiều tùy chọn khác trong giao diện người dùng của Postman.

Hỗ trợ JSON và XML: Khi làm việc với các API trả về dữ liệu dưới dạng JSON hoặc XML, Postman giúp dễ dàng đọc và phân tích cú pháp dữ liệu, cũng như hiển thị nó dưới dạng cây (tree view) hoặc bảng (table view) để dễ dàng kiểm tra.

Tạo và lưu trữ Collections: Postman cho phép tổ chức các yêu cầu API vào các Collection, giúp nhóm các yêu cầu API có liên quan lại với nhau. Bạn có thể tạo thư mục, chia sẻ hoặc xuất các Collection để hợp tác với nhóm hoặc tổ chức.

Environments: Postman hỗ trợ tạo các môi trường để dễ dàng chuyển đổi giữa các cấu hình khác nhau, ví dụ như môi trường phát triển (development), môi trường thử nghiệm (staging), hoặc môi trường sản xuất (production).

Kiểm thử API (Testing): Postman hỗ trợ viết các kiểm thử tự động bằng cách sử dụng JavaScript. Bạn có thể thêm các script để kiểm tra các phản hồi API (ví dụ, kiểm tra mã trạng thái HTTP, kiểm tra giá trị trả về, v.v.).

Ví dụ: Kiểm tra xem mã trạng thái HTTP có phải là 200 hay không, hay kiểm tra xem dữ liệu trả về có chứa một số giá trị cụ thể hay không.

Mock Servers: Postman cho phép bạn tạo Mock Servers để mô phỏng phản hồi của API mà không cần phải thực sự gọi đến backend. Điều này hữu ích trong việc phát triển khi backend chưa sẵn sàng hoặc để thử nghiệm các yêu cầu API.

Chia sẻ và cộng tác: Postman cung cấp tính năng chia sẻ Collections và Environments với các thành viên trong nhóm, giúp cộng tác hiệu quả khi làm việc trên các dự án phát triển API. Cộng tác trong thời gian thực với các tính năng như Comments, có thể ghi chú vào các yêu cầu API để giải thích hoặc yêu cầu sửa đổi.

Giới thiệu về Monitor (Giám sát): Postman có thể được sử dụng để thiết lập các tác vụ giám sát API tự động. Có thể lập lịch các cuộc gọi API để kiểm tra tính ổn định và hiệu suất của API theo thời gian.

Tạo tài liệu API (Documentation): Có thể tự động tạo tài liệu cho API của mình từ các yêu cầu và mô tả trong Postman, giúp dễ dàng chia sẻ thông tin API với các thành viên trong nhóm hoặc đối tác bên ngoài.

Tích hợp với CI/CD: Postman hỗ trợ tích hợp với các công cụ CI/CD (Continuous Integration/Continuous Deployment) để tự động hóa các kiểm thử API trong quy trình phát triển phần mềm. Có thể chạy các kiểm thử API của Postman trong các pipeline CI/CD thông qua Newman, công cụ dòng lệnh của Postman

2.9. Github Action

2.9.1. Khái niệm

GitHub Actions là một nền tảng CI/CD tích hợp sẵn trên GitHub, cho phép tự động hóa toàn bộ quy trình phát triển phần mềm từ xây dựng, kiểm thử, đến triển khai. Với GitHub Actions, bạn có thể tạo các workflow (quy trình làm việc) để tự động thực hiện các tác vụ khi có các sự kiện cụ thể xảy ra trong repository của bạn.

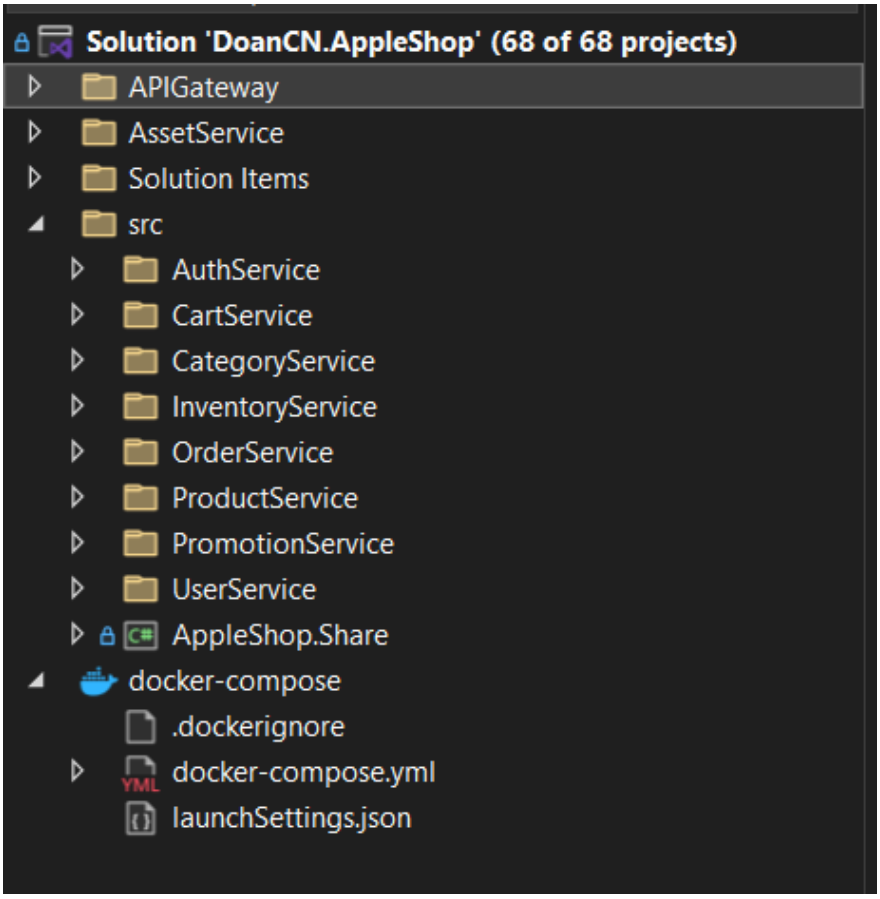
2.9.2. Các thành phần chính

- Workflow: Là một tập hợp các quy trình tự động hóa được định nghĩa trong repository GitHub, giúp quản lý và thực hiện các công việc như build, test, và deploy phần mềm. Workflows được cấu hình thông qua các tệp YAML và lưu trữ trong thư mục .github/workflows của repository
- Jobs: Là các tập hợp các bước (steps) cần thực hiện trong workflow. Mỗi job chạy trên một runner riêng biệt, có thể chạy song song hoặc theo thứ tự phụ thuộc lẫn nhau.
- Runners: Là các máy chủ hoặc máy ảo nơi các jobs trong workflow được thực thi. GitHub cung cấp runners mặc định (hosted runners) và bạn cũng có thể tự thiết lập runners của riêng mình (self-hosted runners).
- Actions: Là các nhiệm vụ (tasks) được thực thi trong một workflow. Một action có thể là một lệnh shell hoặc một phần mềm được đóng gói sẵn giúp thực hiện các nhiệm vụ phổ biến.
- Events: Là các sự kiện kích hoạt workflows. GitHub cung cấp nhiều sự kiện khác nhau, chẳng hạn như khi có push, pull request, hoặc issue mới.

CHƯƠNG 3: THỰC NGHIỆM

3.1. Xây dựng hệ thống

3.1.1. Các Services trong hệ thống

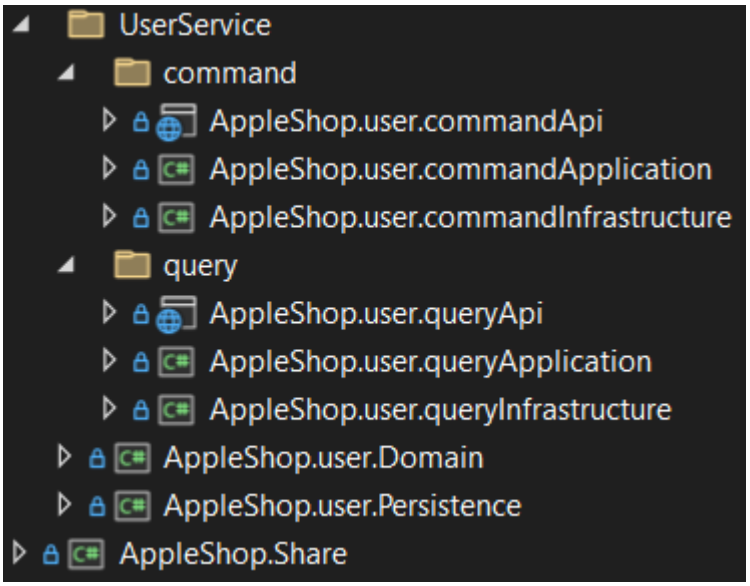


Hình 3.1. Các Services trong hệ thống.

- APIGateway: Nhận request từ người dùng và điều hướng đến service cụ thể để xử lý và trả response về cho người dùng.
- AssetService: Dùng để lưu trữ hình ảnh cho các service.
- AuthService: Dùng để xác thực người dùng trước khi request đến các service con để xử lý. Ngoài ra, một số service sẽ không yêu cầu xác thực.
- CartService: Dùng để xử lý các tác vụ liên quan đến giỏ hàng của người dùng.
- CategoryService: Dùng để xử lý các tác vụ liên quan đến danh mục.
- InventoryService: Dùng để quản lý kho hàng của sản phẩm.
- OrderService: Dùng để xử lý các tác vụ liên quan đến đặt hàng sản phẩm.

- ProductService: Dùng để xử lý các tác vụ liên quan đến sản phẩm.
- PromotionService: Dùng để xử lý các tác vụ liên quan đến các mã giảm giá.
- UserService: Dùng để quản lý các thông tin liên quan đến người dùng.
- SolutionItem: Là thư mục chứa các file cấu hình cho dự án như .gitignore, readme.md,….

3.1.2. Kiến trúc dùng cho hệ thống



Hình 3.2. Kiến trúc của hệ thống.

Hệ thống áp dụng CQRS Pattern và dựa trên kiến trúc Clean Architecture để xây dựng, về cơ bản hệ thống sẽ bao gồm 5 project trong đó mỗi command/query sẽ bao gồm 3 project, 2 project sẽ dùng chung trong mỗi service và 1 project dùng chung cho tất cả các service nằm trong hệ thống. Chức năng của các project:

- Share: Chứa các file common cho hệ thống, các dịch vụ mà tất cả các service có thể dùng chung. Ngoài ra, đây là nơi đảm nhiệm việc giao tiếp giữa các service với nhau.
- Domain: Chứa các thực thể (Entities), các Interfaces của hệ thống.
- Persistence: Chứa các file chủ yếu dùng để giao tiếp với cơ sở dữ liệu, các repository.
- Infrastructure: Chứa các consumers (Masstransit) để xử lý các message được gửi đi từ các service khác. Ngoài ra, nếu áp dụng các dịch vụ từ bên ngoài (Internet Banking,…) sẽ được đăng ký tại tầng này.

- Application: Chứa các usercase và xử lý các request từ người dùng bằng cách sử dụng thư viện MediatR, đảm nhiệm việc validate cho các request trước khi được lưu vào cơ sở dữ liệu.
- Api: Định nghĩa các API bằng cách sử dụng MinimalAPI và đưa ra các đầu API để ứng dụng vào FrontEnd, Mobile,….

3.1.3. Github Action

- Ứng dụng Github Action để tự động hóa việc tạo Pull Request, merge nhánh dev vào nhánh main và tiến hành build ra các Image sau đó được push tự động lên Docker Hub.

- Việc build ra các Image sẽ sử dụng cache của Docker để tránh việc build lại các Service không có cập nhật.

```yaml
create-pull-request:
  runs-on: ubuntu-latest
  permissions:
    contents: write
    pull-requests: write
  steps:
    # Step 1: Checkout repository to work with dev branch
    - name: Checkout repository
      uses: actions/checkout@v4

    # Step 2: Create Pull Request from dev to main
    - name: Create Pull Request
      id: cpr
      uses: peter-evans/create-pull-request@v7
      with:
        token: ${{ secrets.PAT }}
        commit-message: Update report
        committer: github-actions[bot] <41898282+github-actions[bot]@users.noreply.github.com>
        author: ${{ github.actor }} <${{ github.actor_id }}+${{ github.actor }}@users.noreply.github.com>
        signoff: false
        branch: main
        delete-branch: false
        title: 'Update report'
        body: |
          Update report
          - Updated with *today's* date
          - Auto-generated by [create-pull-request][1]

          [1]: https://github.com/hungg523
        labels: |
          automated pr
        milestone: 1
        draft: false
```

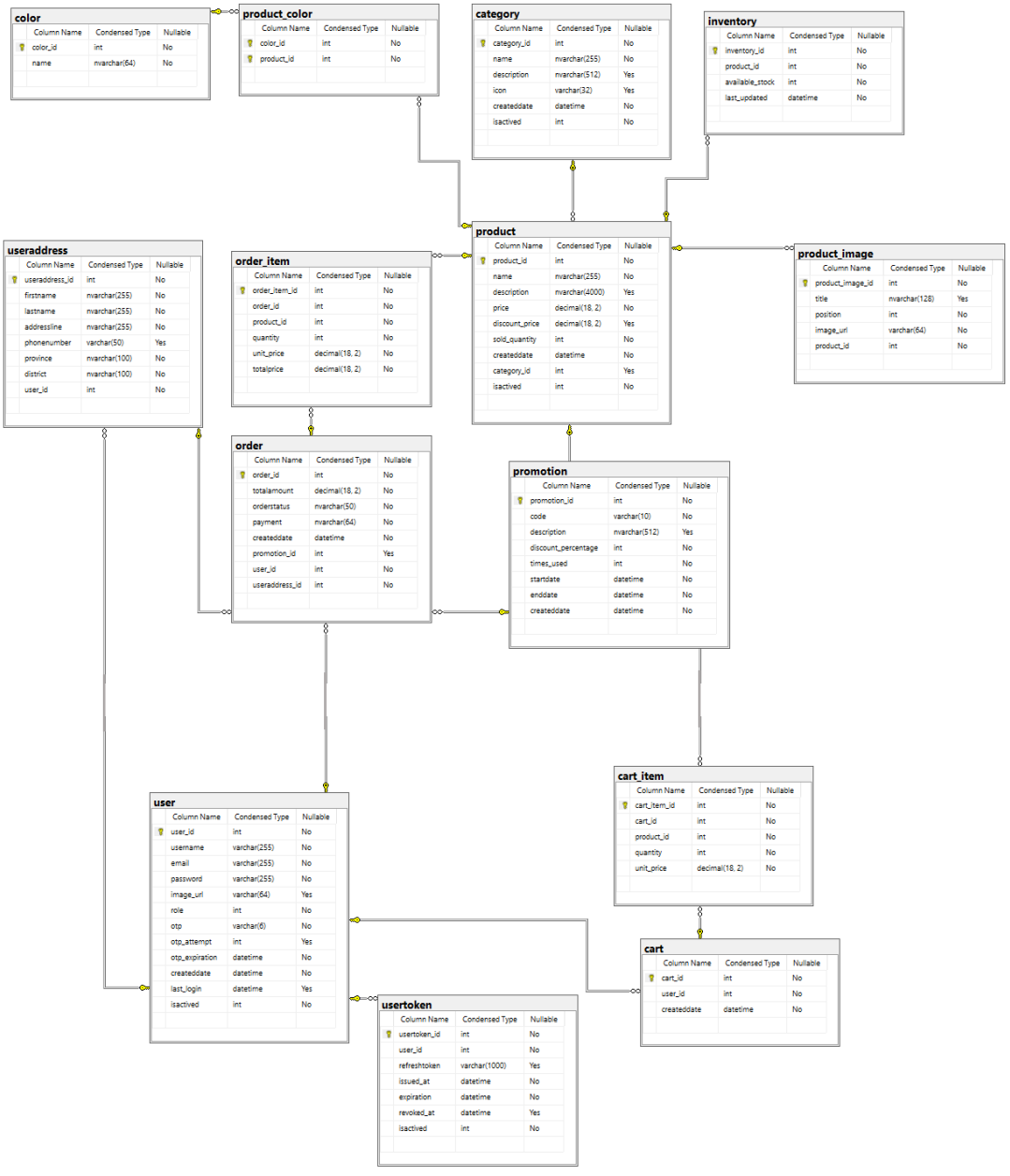
Hình 3.3. Tự động tạo Pull Request

```yaml
build-and-push:
  runs-on: ubuntu-latest
  strategy:
    matrix:
      service:
        - name: assetservice
        - name: auth-command
        - name: cart-command
        - name: cart-query
        - name: category-command
        - name: category-query
        - name: inventory-command
        - name: inventory-query
        - name: order-command
        - name: order-query
        - name: product-command
        - name: product-query
        - name: promotion-command
        - name: promotion-query
        - name: user-command
        - name: user-query
  env:
  steps:
      # Checkout the repository
    - name: Checkout repository
    - name: Set up Docker Buildx
    - name: Cache Docker layers
    - name: Log in to Docker Hub
    - name: Build Docker Images
      uses: docker/build-push-action@v5
      with:

      # Push Docker images
    - name: Push Docker Images
      uses: docker/build-push-action@v5
      with:
          context: .
          file: ${{ matrix.service.path }}
          push: true
          tags: ${{ env.DOCKER_USERNAME }}/doan:${{ matrix.service.tag }}

      # Prune Docker builder cache
    - name: Prune Docker builder cache
      run: docker builder prune -f
```

Hình 3.4. Tự động build các Image

3.2. Xây dựng cơ sở dữ liệu

3.2.1. Thiết kế Diagram



Hình 3.5. Mô hình Diagram

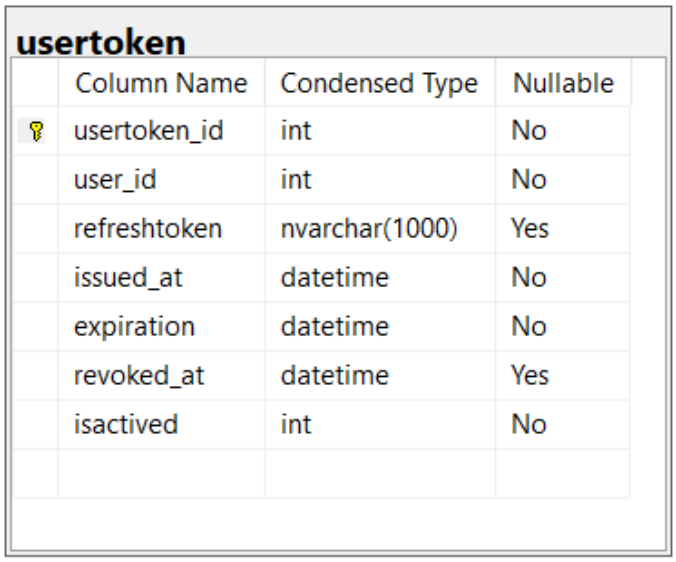
3.3. Thiết kế bảng và cơ sở dữ liệu

3.3.1. AppleAuth Database

AppleAuth được thiết kế dùng cho AuthSerice gồm 1 bảng: usertoken.

Bảng **usertoken** dùng để lưu trữ thông tin liên quan đến các token dành cho việc xác thực, bao gồm các trường:

- usertoken_id: Khóa chính của bảng.
- user_id: Khóa ngoại liên kết với bảng **user**.
- refreshtoken: Lưu trữ refreshtoken của người dùng khi đăng nhập.
- issued_at: Ghi lại thời gian mà token được khởi tạo.
- expiration: Thời gian hết hạn của token.
- revoked_at: Ghi lại thời gian mà người dùng đăng xuất.
- isactived: Trạng thái của token.



usertoken

Column Name	Condensed Type	Nullable
usertoken_id	int	No
user_id	int	No
refreshtoken	nvarchar(1000)	Yes
issued_at	datetime	No
expiration	datetime	No
revoked_at	datetime	Yes
isactived	int	No

Hình 3.6. Bảng usertoken

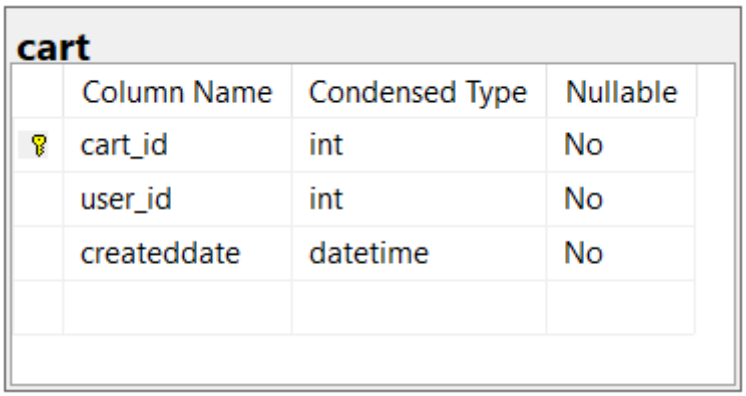
3.3.2. AppleCart Database

AppleCart được thiết kế cho CartService bao gồm 2 bảng: cart, cart_item.

Bảng **cart** dùng lưu trữ thông liên quan đến giỏ hàng của người dùng, bao gồm các trường:

- cart_id: khóa chính của bảng.

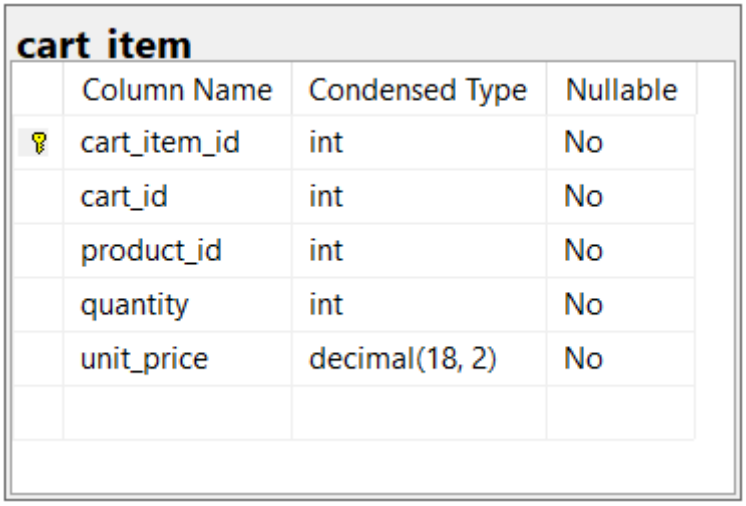
- user_id: khóa ngoại liên kết với bảng **user**.
- createddate: ghi lại ngày giỏ hàng được khởi tạo.



Hình 3.7. Bảng cart

Bảng **cart_item** dùng để lưu trữ các sản phẩm mà người dùng đã thêm vào giỏ hàng của họ, bao gồm các trường:

- cart_item_id: khóa chính của bảng.
- cart_id: khóa ngoại liên kết với bảng **cart**.
- pròuct_id: khóa ngoại liên kết với bảng **product**.
- quantity: số lượng sản phẩm được thêm vào giỏ hàng.
- unit_price: đơn giá của sản phẩm.



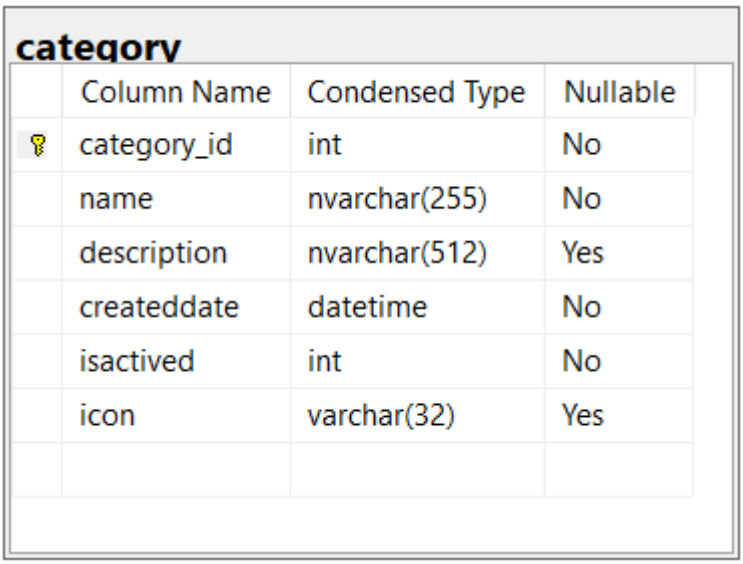
Hình 3.8. Bảng cart_item

3.3.3. AppleCategory Database

AppleCategory được thiết kế cho CategoryService bao gồm 1 bảng: category.

Bảng **category** dùng để lưu trữ các danh mục của sản phẩm, bao gồm các trường:

- category_id: khóa chính của bảng.
- name: tên danh mục.
- description: mô tả của danh mục.
- createddate: ngày tạo danh mục.
- isactived: trạng thái của danh mục.
- icon: icon đại diện cho danh mục.



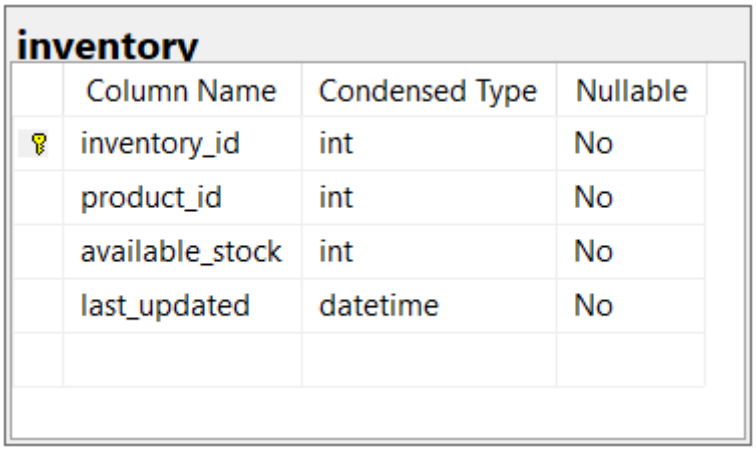
Hình 3.9. Bảng category

3.3.4. AppleInventory Database

AppleInventory được thiết kế cho InventoryService bao gồm 1 bảng: inventory.

Bảng **inventory** dùng để lưu trữ số lượng sản phẩm còn trong kho hàng, bao gồm các trường:

- inventory_id: khóa chính của bảng.
- product_id: khóa ngoại liên kết với bảng **product**.
- available_stock: số lượng sản phẩm còn lại trong kho.
- last_updated: ngày cập nhật số lượng sản phẩm.



inventory

	Column Name	Condensed Type	Nullable
🔑	inventory_id	int	No
	product_id	int	No
	available_stock	int	No
	last_updated	datetime	No

Hình 3.10. Bảng inventory

3.3.5. AppleOrder Database

AppleOrder được thiết kế dùng cho OrderService bao gồm 2 bảng: order, order_item.

Bảng **order** dùng để lưu trữ các thông tin liên quan đến việc đặt hàng của người dùng, bao gồm các trường:

- order_id: khóa chính của bảng.
- totalamount: tổng giá tiền của đơn hàng.
- orderstatus: trạng thái của đơn hàng.
- payment: phương thức thanh toán của đơn hàng.
- createddate: ngày đặt hàng.
- promotion_id: khóa ngoại liên kết với bảng **promotion**.
- user_id: khóa ngoại liên kết với bảng **user**.
- useraddress_id: khóa ngoại liên kết với bảng **useraddress**.

order

	Column Name	Condensed Type	Nullable
🔑	order_id	int	No
	totalamount	decimal(18, 2)	No
	orderstatus	nvarchar(50)	No
	payment	nvarchar(64)	No
	createddate	datetime	No
	promotion_id	int	Yes
	user_id	int	No
	useraddress_id	int	No

Hình 3.11. Bảng order

Bảng **order_item** dùng để lưu các sản phẩm mà người dùng đã đặt mua, bao gồm các trường:

- order_item_id: khóa chính của bảng.
- order_id: khóa ngoại liên kết với bảng **order**.
- product_id: khóa ngoại liên kết với bảng **product**.
- quantity: số lượng sản phẩm đặt mua.
- unit_price: đơn giá của sản phẩm được đặt.
- totalprice: tổng tiền của sản phẩm được đặt.

order_item

Column Name	Condensed Type	Nullable
order_item_id	int	No
order_id	int	No
product_id	int	No
quantity	int	No
unit_price	decimal(18, 2)	No
totalprice	decimal(18, 2)	No

Hình 3.12. Bảng order_item

3.3.6. AppleProduct Database

AppleProduct được thiết kế cho ProductService bao gồm 4 bảng: product, product_image, color, product_color.

Bảng **product** dùng để lưu trữ các thông tin của sản phẩm, bao gồm các trường:

- product_id: khóa chính của bảng.
- name: tên sản phẩm.
- description: mô tả của sản phẩm.
- price: giá của sản phẩm.
- createddate: ngày tạo sản phẩm.
- category_id: khóa ngoại liên kết với bảng **category**.
- isactived: trạng thái của sản phẩm.
- discount_price: giá giảm của sản phẩm.
- sold_quantity: số lượng sản phẩm đã bán.

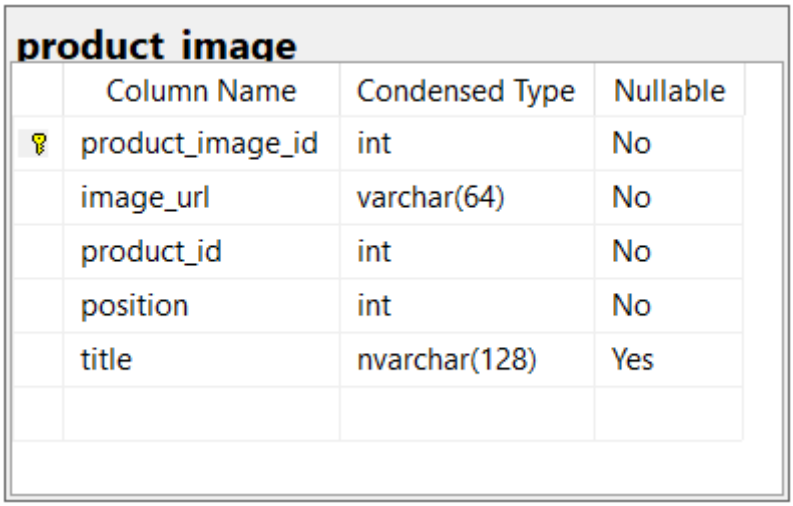
product

Column Name	Condensed Type	Nullable
product_id	int	No
name	nvarchar(255)	No
description	nvarchar(4000)	Yes
price	decimal(18, 2)	No
createddate	datetime	No
category_id	int	Yes
isactived	int	No
discount_price	decimal(18, 2)	Yes
sold_quantity	int	No

Hình 3.13. Bảng product

Bảng **product_image** dùng để lưu trữ các thông tin liên quan đến hình ảnh của sản phẩm, bao gồm các trường:

- product_image_id: khóa chính của bảng.
- image_url: đường dẫn đến vị trí lưu ảnh.
- product_id: khóa ngoại liên kết đến bảng **product**.
- position: vị trí của hình ảnh.
- title: tiêu đề của ảnh.



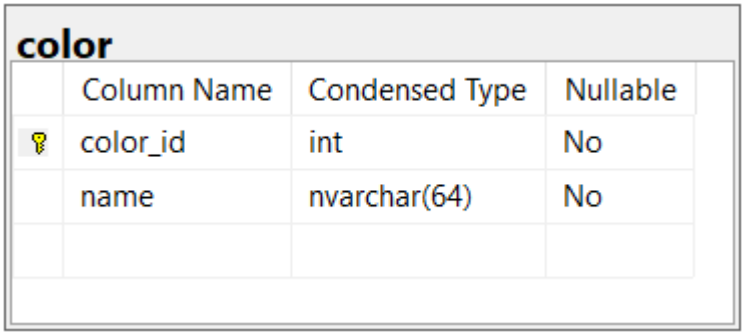
product_image

Column Name	Condensed Type	Nullable
product_image_id	int	No
image_url	varchar(64)	No
product_id	int	No
position	int	No
title	nvarchar(128)	Yes

Hình 3.14. Bảng product_image

Bảng **color** dùng để lưu trữ các thông tin liên quan đến màu sắc của sản phẩm, bao gồm các trường:

- color_id: khóa chính của bảng.
- name: tên gọi của màu sắc.



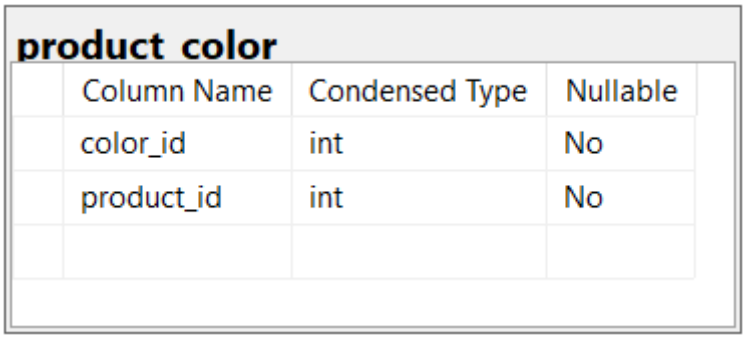
color

Column Name	Condensed Type	Nullable
color_id	int	No
name	nvarchar(64)	No

Hình 3.15. Bảng color

Bảng **product_color** dùng để lưu trữ các thông tin liên quan đến sản phẩm có bao nhiêu màu, bao gồm các trường:

- product_id: khóa ngoại liên kết đến bảng **product**.
- color_id: khóa ngoại liên kết đến bảng **color**.



product_color

Column Name	Condensed Type	Nullable
color_id	int	No
product_id	int	No

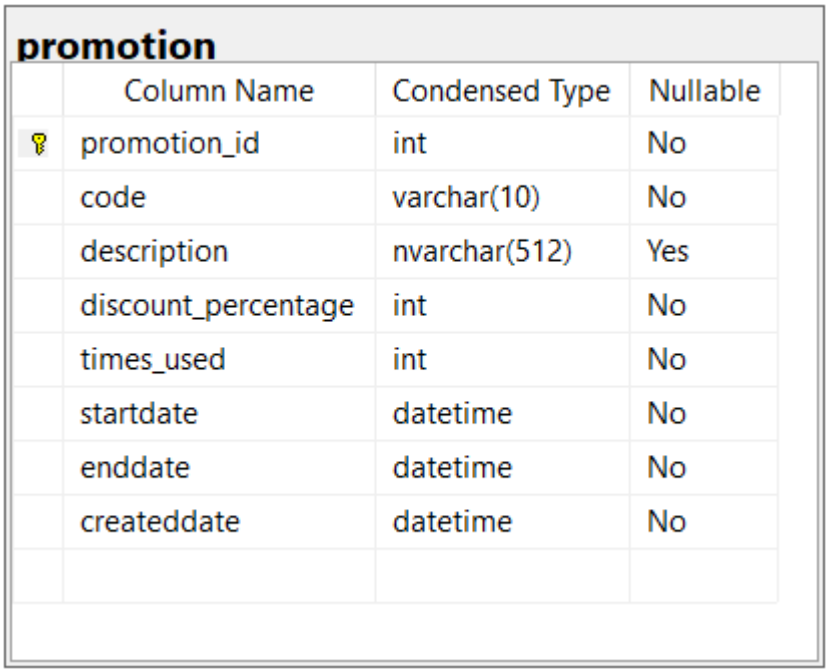
Hình 3.16. Bảng product_color

3.3.7. ApplePromotion Database

ApplePromotion được thiết kế cho PromotionService bao gồm 1 bảng: promotion.

Bảng **promotion** dùng để lưu trữ các thông tin liên quan đến mã giảm giá, bao gồm các trường:

- promotion_id: khóa chính của bảng.
- code: đoạn mã của phiếu giảm giá.
- description: mô tả của phiếu giảm giá.
- discount_percentage: phần trăm giá được giảm khi sử dụng phiếu giảm giá.
- times_used: số lần sử dụng của phiếu giảm giá.
- startdate: ngày phiếu giảm giá có hiệu lực.
- enddate: ngày phiếu giảm giá hết hạn.
- createddate: ngày tạo phiếu giảm giá.



promotion

Column Name	Condensed Type	Nullable
promotion_id	int	No
code	varchar(10)	No
description	nvarchar(512)	Yes
discount_percentage	int	No
times_used	int	No
startdate	datetime	No
enddate	datetime	No
createddate	datetime	No

Hình 3.17. Bảng promotion

3.3.8. AppleUser Database

AppleUser được thiết kế cho UserService bao gồm 2 bảng: user, useraddress.

Bảng **user** dùng để lưu trữ các thông tin liên quan đến của người dùng, bao gồm các trường:

- user_id: khóa chính của bảng.
- username: tên người dùng.
- email: địa chỉ mail của người dùng.
- password: mật khẩu của người dùng đã được mã hóa.
- otp: mã otp khi đăng ký tài khoản.
- otp_attemp: số lần nhập thử otp, tối đa 5 lần.
- otp_expiration: thời gian hết hạn của otp.
- createddate: ngày tạo otp.
- last_login: thời gian đăng nhập gần nhất.
- isactived: trạng thái tài khoản của người dùng
- role: quyền của người dùng.

- image_url: đường dẫn ảnh đại diện của người dùng.

user

	Column Name	Condensed Type	Nullable
🔑	user_id	int	No
	username	varchar(255)	No
	email	varchar(255)	No
	password	varchar(255)	No
	otp	varchar(6)	No
	otp_attempt	int	Yes
	otp_expiration	datetime	No
	createddate	datetime	No
	last_login	datetime	Yes
	isactived	int	No
	role	int	No
	image_url	varchar(64)	Yes

Hình 3.18. Bảng user

Bảng **useraddress** dùng để lưu trữ các thông tin liên quan đến địa chỉ nhận hàng của người dùng, bao gồm các trường:

- useraddress_id: khóa chính của bảng.
- firstname: họ của người nhận.
- lastname: tên của người nhận.
- addressline: địa chỉ cụ thể của người nhận (bao gồm phường/xã/thị trấn).
- phonenumber: số điện thoại của người nhận.
- provice: tỉnh/thành phố nhận hàng.
- district: quận/huyện nhận hàng.
- user_id: khóa ngoại liên kết với bảng **user**.

useraddress

	Column Name	Condensed Type	Nullable
🔑	useraddress_id	int	No
	firstname	nvarchar(255)	No
	lastname	nvarchar(255)	No
	addressline	nvarchar(255)	No
	phonenumber	varchar(50)	Yes
	province	nvarchar(100)	No
	district	nvarchar(100)	No
	user_id	int	No

Hình 3.19. Bảng useraddress

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Quy ước chung cho API

Bảng 4.1. Quy ước chung dữ liệu đầu ra

Field name	Value	Description	Data type	Nullable
isSuccess	true, false	- Khi thành công sẽ trả về true - Khi thất bại sẽ trả về false	bool	
statusCode	Mã trạng thái	- 200 trạng thái thành công - 400 các lỗi liên quan đến validate request của người dùng - 401 xác thực thất bại - 404 không tìm thấy thực thể - 500 lỗi liên quan đến server	int	
data	{id: 1,…}	Dữ liệu data trả về	JSON	x
errors	Danh sách các lỗi	Sẽ trả về danh sách các lỗi		x

- Đường dẫn API <url>: https://localhost:7001

- Khi cập nhật nếu giá trị là NULL field sẽ giữ lại giá trị cũ.

- Hiển thị ảnh bằng đường dẫn sau: https://localhost:7001/assets/image-icon-url

4.2. Auth Service

4.2.1. Register API (User)

- API: <url>/auth/register

- Mô tả: Dùng để đăng ký tài khoản mới.

Input:

Bảng 4.2. Register Request

Field name	Value	Description	Data type	Max lenght	Nullable
email	example@ex.com	Địa chỉ email của người dùng	string	255	
password	P@sswork123!	Mật khẩu	string	255	
confirmPassword	P@sswork123!	Nhập lại mật khẩu	string	255	

Output: None

4.2.2. VertifyOTP API (User)

- API: <url>/auth/vertify-otp

- Mô tả: Dùng để xác thực OTP của người dùng.

Input:

Bảng 4.3. VertifiOTP Request

Field name	Value	Description	Data type	Max lenght	Nullable
email	example@ex.com	Địa chỉ email của người dùng	string	255	
otp	ABC123	Mã otp của người dùng	string	6	

Output: None

4.2.3. ResendOTP API (User)

- API: <url>/auth/resend-otp

- Mô tả: Dùng để gửi lại OTP khi OTP cũ đã quá hạn hoặc khi người dùng không nhận được OTP.

Input:

Bảng 4.4. ResendOTP Request

Field name	Value	Description	Data type	Max lenght	Nullable
email	example@ex.com	Địa chỉ email của người dùng	string	255	

Output: None

4.2.4. Login API (User)

- API: <url>/auth/login

- Mô tả: Dùng để đăng nhập.

Input:

Bảng 4.5. Login Request

Field name	Value	Description	Data type	Max lenght	Nullable

email	example@ex.com	Địa chỉ email của người dùng	string	255	
password	P@sswork123!	Mật khẩu	string	255	

Ouput: None

4.2.5. RefreshToken API (User)

- API: <url>/auth/refresh-token

- Mô tả: Dùng để tạo lại access token khi hết phiên đăng nhập.

Input:

Bảng 4.6. RefreshToken Request

Field name	Value	Description	Data type	Max lenght	Nullable
refreshToken	pyMiDh1f8eyPz…	Địa chỉ email của người dùng	string	255	

Output: None

4.2.6. ResetPassword API (User)

- API: <url>/auth/reset-password

- Mô tả: Dùng để tạo lại mật khẩu khi quên, mật khẩu mới sẽ được gửi qua email.

Input:

Bảng 4.7. ResetPassword Request

Field name	Value	Description	Data type	Max lenght	Nullable
email	example@ex.com	Địa chỉ email của người dùng	string	255	

Output: None

4.2.7. ChangePassword API (User)

- API: <url>/auth/change-password

- Mô tả: Dùng để thay đổi mật khẩu của người dùng.

Input:

Bảng 4.8. ChangePassword Request

Field name	Value	Description	Data type	Max lenght	Nullable
newPassword	P@sswork123!	Mật khẩu mới	string	255	
confirmPassword	P@sswork123!	Nhập lại mật khẩu	string	255	

Output: None

4.3. Cart Service

4.3.1. CreateCart API (User)

- API: <url>/cart/create-cart

- Mô tả: Dùng để thêm sản phẩm vào giỏ hàng, nếu sản phẩm đã có sẵn sẽ cộng dồn số lượng.

Input:

Bảng 4.9. CreateCart Request

Field name	Value	Description	Data type	Max lenght	Nullable
userId	1,2,3,..	Id của người dùng	int		
cartItems		Danh sách sản phẩm thêm vào giỏ hàng	CartItemDTO		

Bảng 4.10. CartItemDTO

Field name	Value	Description	Data type	Max lenght	Nullable
productId	1,2,3,..	Id của sản phẩm	int		
quantity	10,25,…	Số lượng sản phẩm cần thêm vào giỏ hàng	int		

Output: None

4.3.2. DeleteCart Item API (User)

- API: <url>/cart/delete-cart-item/{cartId}/{productId}

- Mô tả: Dùng để loại bỏ sản phẩm ra khỏi giỏ hàng.

Input:

Bảng 4.11. DeleteCart Request

Field name	Value	Description	Data type	Max lenght	Nullable
cartId	1,2,3,..	Id của giỏ hàng	int		
productId	1,2,3,..	Id của sản phẩm cần loại bỏ	int		

Output:

4.3.3. GetCartByUserId API (User)

- API: <url>/carts/get-cart-by-user-id/{id}

- Mô tả: Dùng để xem tất cả sản phẩm có trong giỏ hàng của người dùng.

Input:

Bảng 4.12. GetCartByUserId Request

Field name	Value	Description	Data type	Max lenght	Nullable
userId	1,2,3,..	Id của người dùng	int		

Output:

Bảng 4.13. CartFullDTO

Field name	Value	Description	Data type	Nullable
cartId	1,2,3,..	Id của giỏ hàng	int	
cartItems		Danh sách các sản phẩm có trong giỏ hàng của người dùng	CartItemDTO	
totalPrice	100000, 500000,…	Tổng tiền của giỏ hàng	decimal	

Bảng 4.14. CartItemDTO

Field name	Value	Description	Data type	Nullable
productId	1,2,3,..	Id của sản phẩm	int	

name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	x
quantity	10,25,…	Số lượng sản phẩm cần thêm vào giỏ hàng	int	
unitPrice	50000, 100000,…	Giá của sản phẩm	decimal	
totalPrice	100000, 500000,…	Tổng tiền của sản phẩm	decimal	

4.4. Category Service

4.4.1. CreateCategory API (Admin)

- API: <url>/category/create-category

- Mô tả: Dùng để thêm mới một danh mục

Input:

Bảng 4.15. CreateCategory Request

Field name	Value	Description	Data type	Max lenght	Nullable
name	Iphone, Ipad,…	Tên danh mục	string	255	
iconData	…	Chuỗi base64 của ảnh	string		x
description	Mô tả của danh mục	Mô tả của danh mục	string	max	x
isactived	0: chưa kích hoạt	Trạng thái của	int		

	1: đã kích hoạt	danh mục, mặc định là 0			

Output: None

4.4.2. UpdateCategory API (Admin)

- API: <url>/category/update-category/{id}

- Mô tả: Dùng để cập nhật danh mục dựa vào id.

Input:

Bảng 4.16. UpdateCategory Request

Field name	Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	Id của danh mục	int		
name	Iphone, Ipad,…	Tên danh mục	string	255	x
iconData	…	Chuỗi base64 của ảnh	string		x
description	Mô tả của danh mục	Mô tả của danh mục	string	max	x
isactived	0: chưa kích hoạt 1: đã kích hoạt	Trạng thái của danh mục, mặc định là 0	int		x

Output: None

4.4.3. GetAllCategories API (Admin)

- API: <url>/get-all-categories

- Mô tả: Dùng để xem tất cả danh mục có trong cơ sở dữ liệu.

Input: None

Output:

Bảng 4.17. Category

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của danh mục	int	
name	Iphone, Ipad,…	Tên danh mục	string	
iconData	…	Chuỗi base64 của ảnh	string	x
description	Mô tả của danh mục	Mô tả của danh mục	string	x
createdDate	01/06/2025	Ngày tạo danh mục	datetime	
isactived	0: chưa kích hoạt 1: đã kích hoạt	Trạng thái của danh mục	int	

4.4.4. GetAllCategoriesActived API (User)

- API: <url>/categories/get-all-categories-actived

- Mô tả: Dùng để xem tất cả danh mục đã được kích hoạt có trong cơ sở dữ liệu.

Input: None

Output:

Bảng 4.18. Category

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của danh mục	int	
name	Iphone, Ipad,…	Tên danh mục	string	
iconData	…	Chuỗi base64	string	x

		của ảnh		
description	Mô tả của danh mục	Mô tả của danh mục	string	x
createdDate	01/06/2025	Ngày tạo danh mục	datetime	
isactived	1: đã kích hoạt	Trạng thái của danh mục	int	

4.4.5. GetCategoryById API (Admin)

- API: <url>/categories/get-category-by-id/{id}

- Mô tả: Dùng để xem danh mục cụ thể dựa vào id.

Input:

Bảng 4.19. GetCategoryById Request

Field name	Value	Description	Data type	Max lenght	Nullable
categoryId	1,2,3,..	Id của danh mục	int		

Output:

Bảng 4.20. Category

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của danh mục	int	
name	Iphone, Ipad,…	Tên danh mục	string	
iconData	…	Chuỗi base64 của ảnh	string	x
description	Mô tả của danh mục	Mô tả của danh mục	string	x

createdDate	01/06/2025	Ngày tạo danh mục	datetime	
isactived	0: chưa kích hoạt 1: đã kích hoạt	Trạng thái của danh mục	int	

4.5. Inventory Service

4.5.1. GetAllInventories API (Admin)

- API: <url>/inventories/get-all-inventories

- Mô tả: Dùng để xem thông tin kho hàng của tất cả sản phẩm.

Input: None

Output:

Bảng 4.21. Inventory

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của kho sản phẩm	int	
productId	1,2,3,…	Id của sản phẩm	int	
stock	10,50,…	Số lượng sản phẩm còn trong kho	int	
lastUpdated	01/06/2025	Ngày cập nhật kho gần nhất	datetime	

4.5.2. GetInventoryById API (Admin)

- API: <url>/ inventories /get-inventory-by-id/{id}

- Mô tả: Dùng để xem danh mục cụ thể dựa vào id.

Input:

Bảng 4.22. GetInventoryById Request

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của kho sản phẩm	int	

Output:

Bảng 4.23. Inventory

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của kho sản phẩm	int	
productId	1,2,3,…	Id của sản phẩm	int	
stock	10,50,…	Số lượng sản phẩm còn trong kho	int	
lastUpdated	01/06/2025	Ngày cập nhật kho gần nhất	datetime	

4.6. Order Service

4.6.1. CreateOrder API (User)

- API: <url>/order/create-order

- Mô tả: Dùng để tạo mới một đơn hàng.

Input:

Bảng 4.24. CreateOrder Request

Field name	Value	Description	Data type	Max lenght	Nullable

userId	1,2,3,..	Id của người dùng	int		
userAddressId	1,2,3,..	Id địa chỉ của người dùng	int		
promotionCode	APPLESHOP	Mã giảm giá	string		x
orderItems		Danh sách sản phẩm đã mua	OrderItemDTO		

Bảng 4.25. OrderItemDTO

Field name	Value	Description	Data type	Max lenght	Nullable
productId	1,2,3,..	Id của sản phẩm	int		
quantity	10,25,…	Số lượng sản phẩm cần mua	int		

Output: None

4.6.2. ChangeStatus API (Admin)

- API: <url>/order/change-status/{id}

- Mô tả: Dùng để thay đổi trạng thái của người dùng.

Input:

Bảng 4.26. Change Status Request

Field name	Value	Description	Data type	Max lenght	Nullable
id	1,2,3,..	Id của giỏ	int		
orderStatus	0: Pending 1: ACCEPT,	Trạng thái của sản phẩm	int		x

	2: SHIPPING,..				

Output: None

4.6.3. GetAllOrders API (Admin)

- API: <url>/orders/get-all-orders

- Mô tả: Dùng để xem danh sách tất cả đơn hàng của người dùng.

Input: None

Output:

Bảng 4.27. OrderFullDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của đơn hàng	int	
users		Thông tin của người dùng	UserDTO	
status	SUCCESS,…	Trạng thái của đơn hàng	string	
userAddresses		Địa chỉ của người đặt hàng	UserAddressDTO	
promotions		Thông tin của phiếu giảm giá đã áp dụng cho đơn hàng	PromotionDTO	x
orderItems		Danh sách các sản phẩm đã mua	OrderItemDTO	
totalAmout	100000, 200000,…	Tổng tiền của đơn hàng	decimal	

Bảng 4.28. UserDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của người dùng	int	
email	example@ex.com	Email của người dùng	string	

Bảng 4.29. UserAddressDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id địa chỉ của đặt hàng	int	
fullName	Dương Trung H	Tên của người đặt hàng	string	
phoneNumber	0123498756	Số điện thoại của người đặt hàng	string	
address	126 Nguyễn Thiện Thành, Phường 5, Trà Vinh	Địa chỉ của người đặt hàng	string	

Bảng 4.30. PromotionDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của phiếu giảm giá	int	
discountPercentage	20,50,…	Phần trăm giá được giảm	int	
Description	Mô tả của phiếu giảm giá	Mô tả của phiếu giảm giá	string	

Bảng 4.31. OrderItemDTO

Field name	Value	Description	Data type	Nullable
productId	1,2,3,..	Id của sản phẩm	int	
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	

description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	x
quantity	10,25,…	Số lượng sản phẩm cần thêm vào giỏ hàng	int	
unitPrice	50000, 100000,…	Giá của sản phẩm	decimal	
totalPrice	100000, 500000,…	Tổng tiền của sản phẩm	decimal	

4.6.4. GetOrderById API (Admin, User)

- API: <url>/orders/get-order-by-id/{id}

- Mô tả: Dùng để xem đơn hàng dựa vào id.

Input:

Bảng 4.32. GetOrderById Request

Field name	Value	Description	Data type	Max lenght	Nullable
id	1,2,3,..	Id của giỏ hàng	int		

Output:

Bảng 4.33. OrderFullDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của đơn hàng	int	
users		Thông tin của người dùng	UserDTO	
status	SUCCESS,…	Trạng thái của đơn hàng	string	
userAddresses		Địa chỉ của người đặt hàng	UserAddressDTO	
promotions		Thông tin của phiếu giảm giá đã áp dụng cho đơn hàng	PromotionDTO	x
orderItems		Danh sách các sản phẩm đã mua	OrderItemDTO	
totalAmout	100000, 200000,…	Tổng tiền của đơn hàng	decimal	

Bảng 4.34. UserDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của người dùng	int	
email	example@ex.com	Email của người dùng	string	

UserAddressDTO:

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id địa chỉ của đặt hàng	int	
fullName	Dương Trung H	Tên của người đặt hàng	string	
phoneNumber	0123498756	Số điện thoại của người đặt hàng	string	
address	126 Nguyễn Thiện Thành, Phường 5, Trà Vinh	Địa chỉ của người đặt hàng	string	

Bảng 4.35. PromotionDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của phiếu giảm giá	int	
discountPercentage	20,50,…	Phần trăm giá được giảm	int	
Description	Mô tả của phiếu giảm giá	Mô tả của phiếu giảm giá	string	

Bảng 4.36. OrderItemDTO

Field name	Value	Description	Data type	Nullable
productId	1,2,3,..	Id của sản phẩm	int	
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	x
quantity	10,25,…	Số lượng sản phẩm cần thêm vào giỏ hàng	int	
unitPrice	50000, 100000,…	Giá của sản phẩm	decimal	
totalPrice	100000, 500000,…	Tổng tiền của sản phẩm	decimal	

4.6.5. GetOrderByUserId API (User)

- API: <url>/orders/get-order-by-user-id/{id}

- Mô tả: Dùng để xem những đơn hàng mà người dùng đã đặt

Input:

Bảng 4.37. GetOrderByUserId Request

Field name	Value	Description	Data type	Max lenght	Nullable
id	1,2,3,..	Id của người	int		

		dùng			

Output:

Bảng 4.38. OrderFullDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của đơn hàng	int	
users		Thông tin của người dùng	UserDTO	
status	SUCCESS,…	Trạng thái của đơn hàng	string	
userAddresses		Địa chỉ của người đặt hàng	UserAddressDTO	
promotions		Thông tin của phiếu giảm giá đã áp dụng cho đơn hàng	PromotionDTO	x
orderItems		Danh sách các sản phẩm đã mua	OrderItemDTO	
totalAmout	100000, 200000,…	Tổng tiền của đơn hàng	decimal	

Bảng 4.39. UserDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của người dùng	int	
email	example@ex.com	Email của người dùng	string	

Bảng 4.40. UserAddressDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id địa chỉ của đặt hàng	int	
fullName	Dương Trung H	Tên của người đặt hàng	string	
phoneNumber	0123498756	Số điện thoại của người đặt hàng	string	
address	126 Nguyễn Thiện Thành, Phường 5, Trà Vinh	Địa chỉ của người đặt hàng	string	

Bảng 4.41. PromotionDTO

Field name	Value	Description	Data type	Nullable
id	1,2,3,…	Id của phiếu giảm giá	int	
discountPercentage	20,50,…	Phần trăm giá được giảm	int	
Description	Mô tả của phiếu giảm	Mô tả của phiếu giảm giá	string	

	giá			

Bảng 4.42. OrderItemDTO

Field name	Value	Description	Data type	Nullable
productId	1,2,3,..	Id của sản phẩm	int	
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	x
quantity	10,25,…	Số lượng sản phẩm cần thêm vào giỏ hàng	int	
unitPrice	50000, 100000,…	Giá của sản phẩm	decimal	
totalPrice	100000, 500000,…	Tổng tiền của sản phẩm	decimal	

4.7. Product Service

4.7.1. CreateProduct API (Admin)

- API: <url>/product/create-product

- Mô tả: Dùng để thêm mới một sản phẩm

Input:

Bảng 4.43. CreateProduct Request

Field name	Value	Description	Data type	Max lenght	Nullable
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	255	
description	Mô tả về sản phẩm	Mô tả về sản phẩm	string	4000	x
colorIds	1,2,3,…	Danh sách id về màu của sản phẩm	List<int>		x
price	1000000, 2000000,…	Giá của sản phẩm, mặc định là 0	decimal		
discountPrice	1000000, 2000000,…	Giá của sản phẩm	decimal		x
stockQuantity	10,25,50,…	Số lượng sản phẩm thêm vào kho, mặc định là 0	int		x
categoryId	1,2,3,…	Id của danh mục được liên kết	int		x

isActived	0: Chưa kích hoạt 1: Đã kích hoạt	Trạng thái của sản phẩm, mặc định là 0	int		

Output: None

4.7.2. UpdateProduct API (Admin)

- API: <url>/product/update-product/{id}

- Mô tả: Dùng để cập nhật sản phẩm có trong cơ sở dữ liệu dựa vào id.

Input:

Bảng 4.44. UpdateProduct Request

Field name	Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id của sản phẩm			
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	255	x
description	Mô tả về sản phẩm	Mô tả về sản phẩm	string	4000	x
colorIds	1,2,3,…	Danh sách id về màu của sản phẩm	List<int>		x
price	1000000, 2000000,…	Giá của sản phẩm, mặc định là 0	decimal		x
discountPrice	1000000, 2000000,…	Giá của sản phẩm	decimal		x
stockQuantity	10,25,50,…	Số lượng sản phẩm thêm vào kho, mặc định là 0	int		x
categoryId	1,2,3,…	Khóa chính của danh mục được liên kết	int		x
isActived	0: Chưa kích hoạt 1: Đã kích hoạt	Trạng thái của sản phẩm, mặc định là 0	int		x

Output: None

4.7.3. CreateProductImage API (Admin)

- API: <url>/product/create-product-image

- Mô tả: Dùng để thêm hình ảnh cho sản phẩm

Input:

Bảng 4.45. CreateProductImage Request

Field name	Value	Description	Data type	Max lenght	Nullable
title	Tiêu đề hình ảnh của sản phẩm	Tiêu đề hình ảnh của sản phẩm	string	128	x
imageData	data:image/png; base64,iVBO…	Chuỗi base64 của ảnh	string		x
position	0,1,2,3,…	Vị trí hiển thị của hình ảnh, mặc định là 0	int		
productId	1,2,3,…	Id của sản phẩm được liên kết	int		

Output: None

4.7.4. UpdateProductImage API (Admin)

- API: <url>/product/update-product-image

- Mô tả: Dùng để cập nhật hình ảnh có trong cơ sở dữ liệu dựa vào id.

Input:

Bảng 4.46. UpdateProductImage Request

Field name	Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	Id ảnh của sản phẩm	int		
title	Tiêu đề hình ảnh của sản phẩm	Tiêu đề hình ảnh của sản phẩm	string	128	x
imageData	data:image/png; base64,iVBO…	Chuỗi base64 của ảnh	string		x
position	0,1,2,3,…	Vị trí hiển thị của hình ảnh, mặc định là 0	int		x
productId	1,2,3,…	Id của sản phẩm được liên kết	int		x

Output: None

4.7.5. DeleteProductImage API (Admin)

- API: <url>/product/delete-product-image/{id}

- Mô tả: Dùng để xóa ảnh của sản phẩm dựa vào id.

Input:

Bảng 4.47. DeleteProductImage Request

Field name	Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	Id ảnh của sản phẩm	int		

Output: None

4.7.6. CreateColor API (Admin)

- API: <url>/product/create-color

- Mô tả: Dùng để thêm mới màu cho sản phẩm.

Input:

Bảng 4.48. CreateColor Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
name	đỏ, xanh, vàng,…	Tên màu sắc của sản phẩm	string	64	

Output: None

4.7.7. UpdateColor API (Admin)

- API: <url>/product/update-color/{id}

- Mô tả: Dùng để cập nhật màu sắc dựa vào id.

Input:

Bảng 4.49. UpdateColor Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	Id của màu sắc	int		
name	đỏ, xanh, vàng,…	Tên màu sắc của sản phẩm	string	64	x

Output: None

4.7.8. DeleteColor API (Admin)

- API: <url>/product/delete-color/{id}

- Mô tả: Dùng để xóa màu sắc có trong cơ sở dữ liệu dựa vào id.

Input:

Bảng 4.50. DeleteColor Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	Id của màu sắc	int		

Output: None

4.7.9. GetAllProduct API (Admin)

- API: <url>/products/get-all-products

- Mô tả: Dùng để xem toàn bộ danh sách các sản phẩm có trong cơ sở dữ liệu

Input: None

Output:

Bảng 4.51. ProductFullDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của sản phẩm	int	
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
Description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	
colors		Danh sách các màu của sản phẩm	ColorDTO	
price	1000000,2000000,…	Giá của sản phẩm	decimal	
discountPrice	1000000,2000000,…	Giá giảm của sản phẩm	decimal	x
stock	10,25,50,…	Số lượng sản phẩm có trong kho	int	
isActived	0: Chưa kích hoạt 1: Đã kích hoạt	Trạng thái của sản phẩm	int	
images		Danh sách hình của sản phẩm	ProductImageDTO	

Bảng 4.52. ColorDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của màu sắc	int	
name	Đỏ, Xanh, Vàng,…	Tên của màu sắc	string	

ProductImageDTO:

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của hình ảnh	int	
title	Tiêu đề của hình ảnh	Tiêu đề của hình ảnh	string	
url	<url>/assets/image.png	Đường dẫn ảnh	string	
position	0,1,2,…	Vị trí hiển thị của ảnh	int	

4.7.10. GetAllProductActived API (User)

- API: <url>/products/get-all-products-actived

- Mô tả: Dùng để xem toàn bộ danh sách các sản phẩm đã được kích hoạt có trong cơ sở dữ liệu

Input: None

Output:

Bảng 4.53. ProductFullDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của sản phẩm	int	
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
Description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	
colors		Danh sách các màu của sản phẩm	ColorDTO	
price	1000000,2000000,…	Giá của sản phẩm	decimal	
discountPrice	1000000,2000000,…	Giá giảm của sản phẩm	decimal	x
stock	10,25,50,…	Số lượng sản phẩm có trong kho	int	
isActived	1: Đã kích hoạt	Trạng thái của sản phẩm	int	
images		Danh sách hình của sản phẩm	ProductImageDTO	

Bảng 4.54. ColorDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của màu sắc	int	

name	Đỏ, Xanh, Vàng,…	Tên của màu sắc	string	

Bảng 4.55. ProductImageDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của hình ảnh	int	
title	Tiêu đề của hình ảnh	Tiêu đề của hình ảnh	string	
url	<url>/assets/image.png	Đường dẫn ảnh	string	
position	0,1,2,…	Vị trí hiển thị của ảnh	int	

4.7.11. GetProductById API (User)

- API: <url>/products/get-product-by-id/{id}

- Mô tả: Dùng để xem chi tiết một sản phẩm dựa vào id.

Input:

Bảng 4.56. GetProductById Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	Id của sản phẩm	int		

Output:

Bảng 4.57. ProductFullDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của sản phẩm	int	
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
Description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	
colors		Danh sách các màu của sản phẩm	ColorDTO	
price	1000000,2000000,…	Giá của sản phẩm	decimal	
discountPrice	1000000,2000000,…	Giá giảm của sản phẩm	decimal	x
stock	10,25,50,…	Số lượng sản phẩm có trong kho	int	
isActived	1: Đã kích hoạt	Trạng thái của	int	

		sản phẩm		
images		Danh sách hình của sản phẩm	ProductImageDTO	

Bảng 4.58. ColorDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của màu sắc	int	
name	Đỏ, Xanh, Vàng,…	Tên của màu sắc	string	

Bảng 4.59. ProductImageDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của hình ảnh	int	
title	Tiêu đề của hình ảnh	Tiêu đề của hình ảnh	string	
url	<url>/assets/image.png	Đường dẫn ảnh	string	
position	0,1,2,…	Vị trí hiển thị của ảnh	int	

4.7.12. GetProductByCategoryId API (User)

- API: <url>/products/get-product-by-category-id/{id}

- Mô tả: Dùng để lọc sản phẩm theo từng danh mục.

Input:

Bảng 4.60. GetProductByCategoryId Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	Id của danh mục	int		

Output:

Bảng 4.61. ProductFullDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của sản phẩm	int	
name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
Description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	
colors		Danh sách các màu của sản phẩm	ColorDTO	
price	1000000,2000000,…	Giá của sản	decimal	

		phẩm		
discountPrice	1000000,2000000,…	Giá giảm của sản phẩm	decimal	x
stock	10,25,50,…	Số lượng sản phẩm có trong kho	int	
isActived	1: Đã kích hoạt	Trạng thái của sản phẩm	int	
images		Danh sách hình của sản phẩm	ProductImageDTO	

Bảng 4.62. ColorDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của màu sắc	int	
name	Đỏ, Xanh, Vàng,…	Tên của màu sắc	string	

Bảng 4.63. ProductImageDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của hình ảnh	int	
title	Tiêu đề của hình ảnh	Tiêu đề của hình ảnh	string	
url	<url>/assets/image.png	Đường dẫn ảnh	string	
position	0,1,2,…	Vị trí hiển thị của ảnh	int	

4.7.13. GetProductByName API (User)

- API: <url>/products/get-product-by-name/{name}

- Mô tả: Dùng để tìm kiếm các sản phẩm theo tên tương đối.

Input:

Bảng 4.64. GetProductByName Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
name	Iphone 11, Iphone 12,…	Tên của sản phẩm cần tìm	string		

Output:

Bảng 4.65. ProductFullDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của sản phẩm	int	

name	Iphone 11, Iphone 12,…	Tên của sản phẩm	string	
Description	Mô tả của sản phẩm	Mô tả của sản phẩm	string	
colors		Danh sách các màu của sản phẩm	ColorDTO	
price	1000000,2000000,…	Giá của sản phẩm	decimal	
discountPrice	1000000,2000000,…	Giá giảm của sản phẩm	decimal	x
stock	10,25,50,…	Số lượng sản phẩm có trong kho	int	
isActived	1: Đã kích hoạt	Trạng thái của sản phẩm	int	
images		Danh sách hình của sản phẩm	ProductImageDTO	

Bảng 4.66. ColorDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của màu sắc	int	
name	Đỏ, Xanh, Vàng,…	Tên của màu sắc	string	

Bảng 4.67. ProductImageDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	Id của hình ảnh	int	
title	Tiêu đề của hình ảnh	Tiêu đề của hình ảnh	string	
url	<url>/assets/image.png	Đường dẫn ảnh	string	
position	0,1,2,…	Vị trí hiển thị của ảnh	int	

4.8. Promotion Service

4.8.1. CreatePromotion API (Admin)

- API: <url>/promotion/create-promotion

- Mô tả: Dùng để thêm mới mã giảm giá.

Input:

Bảng 4.68. CreatePromotion Request

Field name	Example Value	Description	Data	Max	Nullable

			type	lenght	
code	APPLESHOP	Mã giảm giá	string	10	
description	Mô tả về mã giảm giá	Mô tả về mã giảm giá	string	512	x
discountPercentage	10, 25,…	Phần trăm giá mã giảm giá được giảm, mặc định là 0	int		
timesUsed	5, 10,…	Sô lần phiếu giảm giá được sử dụng, mặc định là 0	int		
startDate	01/07/2025	Ngày mã giảm giá có hiệu lực	datetime		
endDate	01/07/2025	Ngày mã giảm giá hết hiệu lực	datetime		

Output: None

4.8.2. UpdatePromotion API (Admin)

- API: <url>/promotion/update-promotion/{id}

- Mô tả: Dùng để cập nhật mã giảm giá có trong cơ sở dữ liệu dựa vào id.

Input:

Bảng 4.69. UpdatePromotion Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id của mã giảm giá	int		
code	APPLESHOP	Mã giảm giá	string	10	
description	Mô tả về mã giảm giá	Mô tả về mã giảm giá	string	512	x
discountPercentage	10, 25,…	Phần trăm giá mã giảm giá được giảm, mặc định là 0	int		
timesUsed	5, 10,…	Sô lần phiếu giảm giá được sử dụng, mặc định là 0	int		
startDate	01/07/2025	Ngày mã giảm giá có hiệu lực	datetime		
endDate	01/07/2025	Ngày mã giảm giá hết hiệu lực	datetime		

Output: None

4.8.3. DeletePromotion API (Admin)

- API: <url>/promotion/delete-promotion/{id}

- Mô tả: Dùng để xóa mã giảm giá thông qua id.

Input:

Bảng 4.70. DeletePromotion Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id của mã giảm giá	int		

Output: None

4.8.4. GetAllPromotions API (Admin)

- API: <url>/promotions/get-all-promotions

- Mô tả: Dùng để xem toàn bộ mã giảm giá có trong cơ sở dữ liệu.

Input: None

Output:

Bảng 4.71. PromotionDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	id của mã giảm giá	int	
code	APPLESHOP	Mã giảm giá	string	
description	Mô tả về mã giảm giá	Mô tả về mã giảm giá	string	x
discountPercentage	10, 25,…	Phần trăm giá mã giảm giá được giảm, mặc định là 0	int	
timesUsed	5, 10,…	Sô lần phiếu giảm giá được sử dụng, mặc định là 0	int	
startDate	01/07/2025	Ngày mã giảm giá có hiệu lực	datetime	
endDate	01/07/2025	Ngày mã giảm giá hết hiệu lực	datetime	

4.8.5. GetPromotionById API (Admin)

- API: <url>/promotions/get-promotion-by-id/{id}

- Mô tả: Dùng để xem chi tiết mã giảm giá dựa vào id.

Input:

Bảng 4.72. GetPromotionById Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id của mã giảm giá	int		

Output:

Bảng 4.73. PromotionDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	id của mã giảm giá	int	
code	APPLESHOP	Mã giảm giá	string	
description	Mô tả về mã giảm giá	Mô tả về mã giảm giá	string	x
discountPercentage	10, 25,…	Phần trăm giá mã giảm giá được giảm, mặc định là 0	int	
timesUsed	5, 10,…	Sô lần phiếu giảm giá được sử dụng, mặc định là 0	int	
startDate	01/07/2025	Ngày mã giảm giá có hiệu lực	datetime	
endDate	01/07/2025	Ngày mã giảm giá hết hiệu lực	datetime	

4.9. User Service

4.9.1. UpdateProfileUser API (User)

- API: <url>/user/update-profile-user

- Mô tả: Dùng để cập nhật thông tin của người dùng.

Input:

Bảng 4.74. UpdateProfileUser Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id của người dùng	int		
userName	hung523	Tên người dùng	string	255	x
imageData	data:image/png; base64,iVBO…	Chuỗi base64 của ảnh	string		x

Output: None

4.9.2. CreateUserAddress API (User)

- API: <url>/user/create-user-address

- Mô tả: Dùng để thêm mới địa chỉ nhận hàng của người dùng.

Input:

Bảng 4.75. CreateUserAddress Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
firstName	Trương	Họ của người nhận hàng	string	255	
lastName	Hoàng Hưng	Tên của người dùng	string	255	
addressLine	60 Vang Nhứt, Phước Hảo	Địa chỉ của người nhận hàng	string	255	
phoneNumber	0124678953	Số điện thoại của người nhận hàng	string	20	
province	Châu Thành	Quận/Huyện của người nhận hàng	string	100	
district	Trà Vinh	Tỉnh/Thành Phố của người nhận hàng	string	100	
userId	1,2,3,…	id của người dùng cần liên kết	int		

Output: None

4.9.3. UpdateUserAddress API (User)

- API: <url>/user/update-user-address/{id}

- Mô tả: Dùng để cập nhật địa chỉ nhận hàng dựa vào id.

Input:

Bảng 4.76. UpdateUserAddress Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id địa chỉ của người nhận hàng	int		
firstName	Trương	Họ của người nhận hàng	string	255	x
lastName	Hoàng Hưng	Tên của người dùng	string	255	x
addressLine	60 Vang Nhứt, Phước Hảo	Địa chỉ của người nhận hàng	string	255	x
phoneNumber	0124678953	Số điện thoại của người nhận hàng	string	20	x
province	Châu Thành	Quận/Huyện của người nhận hàng	string	100	x
district	Trà Vinh	Tỉnh/Thành Phố của người nhận hàng	string	100	x
userId	1,2,3,…	id của người dùng cần liên kết	int		x

Output: None

4.9.4. DeleteUserAddress API (User)

- API: <url>/user/delete-user-address/{id}

- Mô tả: Dùng để xóa địa chỉ nhận hàng dựa vào id.

Input:

Bảng 4.77. DeleteUserAddress Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id địa chỉ của người nhận hàng	int		

Output: None

4.9.5. GetProfileUserById API (User)

- API: <url>/users/get-profile-user-by-id/{id}

- Mô tả: Dùng để xem thông của người dùng dựa vào id.

Input:

Bảng 4.78. GetProfileUserById Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id địa chỉ của người dùng	int		

Output:

Bảng 4.79. UserDTO

Field name	Example Value	Description	Data type	Nullable
id	1,2,3,…	id của người dùng	int	
userName	hung523	Tên người dùng	string	
imageUrl	<url>/assets/image.png	Đường dẫn ảnh	string	x

4.9.6. GetAllAddressByUserId API (User)

- API: <url>/users/get-all-address-by-user-id/{id}

- Mô tả: Dùng để xem tất cả địa chỉ nhận nhận dựa vào id của người dùng.

Input:

Bảng 4.80. GetAllAddressByUserId Request

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id địa chỉ của người dùng	int		

Output:

Bảng 4.81. UserAddress

Field name	Example Value	Description	Data type	Max lenght	Nullable
id	1,2,3,…	id địa chỉ của người nhận hàng	int		
firstName	Trương	Họ của người nhận hàng	string	255	x

lastName	Hoàng Hưng	Tên của người dùng	string	255	x
addressLine	60 Vang Nhứt, Phước Hảo	Địa chỉ của người nhận hàng	string	255	x
phoneNumber	0124678953	Số điện thoại của người nhận hàng	string	20	x
province	Châu Thành	Quận/Huyện của người nhận hàng	string	100	x
district	Trà Vinh	Tỉnh/Thành Phố của người nhận hàng	string	100	x
userId	1,2,3,…	id của người dùng cần liên kết	int		x

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được

Về chức năng: Hệ thống website đã hoàn thiện phần lớn các chức năng cơ bản của một trang thương mại điện tử, cụ thể:

- Người dùng có thể tạo tài khoản, đăng nhập và sử dụng các chức năng của hệ thống, xem sản phẩm, thêm sản phẩm vào giỏ hàng, đơn hàng,…
- Người quản trị có thể quản lý được sản phẩm, danh mục phiếu giảm giá,….

Về kiến thức và kỹ năng: Học hỏi và áp dụng các công nghệ như .NET Core 8, mô hình Microservice, kiến trúc Clean Architecture, Github Action. Ngoài ra, thông qua đề tài còn giúp em cải thiện các kỹ năng mềm như: tìm kiếm thông tin, giải quyết vấn đề, viết báo cáo và tài liệu hệ thống.

5.2. Kết quả chưa đạt được

Song song với các kết quả đạt được bên trên thì hệ thống cũng còn nhiều thiếu sót về nhiều mặt như là:

Về chức năng:

- Hỗ trợ thanh toán trực tuyến, thông báo các thời điểm khuyến mãi.
- Theo dõi sản phẩm, so sánh sản phẩm, đề xuất sản phẩm, và hệ thống hỗ trợ trực tuyến.

Về bảo mật: hệ thống mới chỉ phát triển mức độ bảo mật cơ bản của JWT, chưa có phân quyền nâng cao.

5.3. Hướng phát triển

Trong tương lai, định hướng phát triển thêm của em sẽ là tiếp tục phát triển, hoàn thiện tối ưu các chức năng đang ở mức sơ khai hoặc chưa được tốt. Bên cạnh đó em sẽ tiếp tục phát triển thêm các chức năng mới như là: hỗ trợ đa ngôn ngữ, theo dõi sản phẩm, so sánh sản phẩm, thông báo thời gian thực, hệ thống nhắn tin và trả lời tự động, tích hợp các hình thức thanh toán online, đưa các đầu API lên giao diện và kiểm thử hiệu năng thực tế,...

Việc cải thiện hiệu năng cũng vô cùng quan trọng trong tương lai khi mà lượng người dùng và dữ liệu ngày một nhiều nên. Giải pháp trong tương lai là tối ưu lại toàn bộ câu truy vấn, sử dụng NoSql để tăng hiệu suất và tốc độ đọc,...

DANH MỤC TÀI LIỆU THAM KHẢO

Admin. (2019, 03 9). *Entity Framework là gì?* Retrieved from netcore: https://netcore.vn/bai-viet/kien-thuc-entity-framwork/entity-framework-la-gi

cloud.z. (2024, 01 22). *Từ A-Z về Microservices và một lưu ý khi thiết kế Microservices*. Retrieved from cloud.z: https://cloud.z.com/vn/news/microservices/#:~:text=%C4%91i%E1%BB%83m%20c%E1%BB%A7a%20Microservices-,Microservices%20l%C3%A0%20m%E1%BB%99t%20ki%E1%BA%BFn%20tr%C3%BAc%20ph%E1%BA%A7n%20m%E1%BB%81m%20m%C3%A0%20trong%20%C4%91%C3%B3,d%E1%BB%85%20d%C3%A0ng%20the

CodeGym. (2024, 07 24). *C# là gì? Tìm hiểu về ngôn ngữ lập trình C#.* Retrieved from CODEGYM: https://codegym.vn/blog/c-la-gi-tim-hieu-ve-ngon-ngu-lap-trinh-c/

Liên, N. (2024). *Postman là gì? Tìm hiểu thành phần, tính ứng dụng và cơ sở chức năng của Postman*. Retrieved from fptshop: https://fptshop.com.vn/tin-tuc/danh-gia/postman-la-gi-168171

nội, H. (2024, 6 11). *GitHub Actions là gì? Công cụ CI/CD hiệu quả cho Devops*. Retrieved from 200Lab: https://200lab.io/blog/github-action-la-gi?srsltid=AfmBOoo8mehx5lQmonF2PagQJJSUm612_as4Mo9cUPSmextL8PaUFMSN

Thương, P. H. (2021, 9 04). *Triển khai CQRS với thư viện MediatR*. Retrieved from thuongph: https://thuongph.com/2021/09/04/trien-khai-cqrs-voi-thu-vien-mediatr/

Training, S. E. (2020, 02 14). *Docker là gì ? Kiến thức cơ bản về Docker*. Retrieved from VIBLO: https://viblo.asia/p/docker-la-gi-kien-thuc-co-ban-ve-docker-maGK7qeelj2

Trần, V. (2022, 5 25). *Clean Architecture là gì - Ưu nhược và cách dùng hợp lý*. Retrieved from 200Lab: https://200lab.io/blog/clean-architecture-uu-nhuoc-va-cach-dung-hop-ly

viettelidc. (2024, 9 23). *SQL là gì? Cách download và cài đặt SQL Server từ A - Z.* Retrieved from viettelidc: https://viettelidc.com.vn/tin-tuc/toan-tap-ve-sql-server-cho-nguoi-moi-bat-dau