

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC SAO ĐỎ



ĐỒ ÁN TỐT NGHIỆP

Ngành: Công nghệ thông tin

TÊN ĐỀ TÀI: NGHIÊN CỨU FRAMEWORK ĐA NỀN TẢNG
FLUTTER, XÂY DỰNG ỨNG DỤNG BÁN HÀNG TRÊN ĐIỆN THOẠI
DI ĐỘNG

Họ và tên sinh viên: Nguyễn Tuấn Anh

Lớp, khoá: DK10-CNTT

Giảng viên hướng dẫn: Ths. Phạm Thị Hường

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả đưa ra trong đồ án tốt nghiệp này là các kết quả thu được trong quá trình nghiên cứu, thực nghiệm của tôi dưới sự hướng dẫn của Ths Phạm Thị Hường, không sao chép bất kỳ kết quả nghiên cứu nào của các tác giả khác.

Nội dung nghiên cứu có tham khảo và sử dụng một số thông tin, tài liệu từ các nguồn tài liệu đã được liệt kê trong danh mục các tài liệu tham khảo.

Nếu sai tôi xin chịu mọi hình thức kỷ luật theo quy định.

Hải Dương, ngày 10 tháng 11 năm 2023

Sinh viên thực hiện

Nguyễn Tuấn Anh

ĐẠI HỌC SAO ĐỎ

LỜI CẢM ƠN

Trước hết, em xin bày tỏ lòng biết ơn chân thành đến cô Phạm Thị Hường, người đã tận tâm và kiên nhẫn dành thời gian giúp đỡ em từ giai đoạn lên ý tưởng cho đến khi hoàn thiện sản phẩm. Từ cung cấp phản hồi hữu ích, làm giàu thêm kiến thức và kỹ năng cùng sự chỉ dẫn, đồng hành và sự hỗ trợ không ngừng là nguồn động viên giúp em vượt qua những thách thức và phát triển kỹ năng chuyên môn để hoàn thành đồ án của mình.

Em xin bày tỏ lòng biết ơn đến gia đình và người thân, người luôn ủng hộ, khích lệ và chia sẻ cùng em trong mọi thời kỳ. Cảm ơn tất cả mọi người đã đồng hành cùng em trên hành trình này. Sự hỗ trợ của mọi người làm cho em trở nên hoàn thiện hơn trong tương lai.

Xin trân trọng cảm ơn!

Hải Dương, ngày 10 tháng 11 năm 2023

Sinh viên thực hiện

Nguyễn Tuấn Anh

MỤC LỤC

	Trang
LỜI CAM ĐOAN	i
MỤC LỤC	iii
DANH MỤC CÁC HÌNH VẼ	vi
MỞ ĐẦU	1
1.1. Tổng quan về kiến trúc di động	3
1.1.1. Kiến trúc hệ thống	3
1.1.2. Kiến trúc ứng dụng Android	3
1.1.3. Kiến trúc ứng dụng iOS	5
1.2. Tổng quan về công nghệ đa nền tảng	6
1.2.1. Định nghĩa về công nghệ đa nền tảng	6
1.2.2. Ưu và nhược điểm của lập trình đa nền tảng	7
1.2.3. Một số công cụ hỗ trợ lập trình đa nền tảng	7
1.3. Tổng quan về NodeJs và NPM	18
1.3.1. Giới thiệu	18
1.3.2. Đặc điểm của Node.js	18
1.3.3. Các thành phần trong Node.js	19
1.4. Kết luận chương 1	19
CHƯƠNG 2. PHÂN TÍCH, THIẾT KẾ HỆ THỐNG	21
2.1. Phân tích yêu cầu	21
2.2. Phân tích các đối tượng và chức năng hệ thống	22
2.2.1. Đối tượng hệ thống	22
2.2.2. Mô tả chức năng chính	23
2.2.3. Mô tả yêu cầu thiết bị và phần mềm	24
2.3. Đề xuất dữ liệu	25
2.4. Phân tích một số quy trình trong thiết kế ứng dụng	31
2.4.1. Quy trình đăng ký tài khoản	31
2.4.2. Quy trình đăng nhập	32

2.4.3. Quy trình xem chi tiết sản phẩm	32
2.4.4. Quy trình thêm sản phẩm vào giỏ hàng	33
2.4.5. Quy trình đặt mua sản phẩm	33
2.4.6. Quy trình tìm kiếm sản phẩm	34
2.4.7. Quy trình xác nhận đơn hàng	34
2.5. Kết luận chương 2	35
CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG BÁN HÀNG TRÊN ĐIỆN THOẠI DI ĐỘNG	36
3.1. Thiết lập API	36
3.1.1. Thiết lập cơ sở dữ liệu MongoDB	36
3.1.2. Thiết lập tạo API	40
3.2. Xây dựng giao diện ứng dụng	41
3.3. Hoàn thiện logic	45
3.3.1. Chức năng đăng nhập/đăng ký	45
3.3.2. Chức năng hiển thị sản phẩm tại trang chủ	48
3.3.3. Chức năng tìm kiếm sản phẩm	49
3.3.4. Chức năng thêm giỏ hàng	51
3.3.5. Chức năng thêm địa chỉ	52
3.4. Kết quả thực nghiệm và đánh giá	53
3.4.1. Kết quả thực nghiệm	53
3.4.2. Đánh giá ứng dụng	63
3.5. Kết luận chương 3	65
KẾT LUẬN	66
1. Kết quả đạt được	66
2. Kiến nghị	66
TÀI LIỆU THAM KHẢO	67

DANH MỤC CÁC BẢNG

Bảng 2.1. Chức năng của quản trị và khách hàng	22
Bảng 2.2. Bảng dữ liệu users	25
Bảng 2.3. Bảng dữ liệu categories	26
Bảng 2.4. Bảng dữ liệu products	27
Bảng 2.5. Bảng dữ liệu product_infos	27
Bảng 2.6. Bảng dữ liệu orders	28
Bảng 2.7. Bảng dữ liệu myorder	29
Bảng 2.8. Bảng dữ liệu carts	30
Bảng 3.1. Bảng cấu trúc ứng dụng	44
Bảng 3.2. Đánh giá về yêu cầu phi chức năng	63
Bảng 3.3. Đánh giá về chức năng ứng dụng	64

DANH MỤC CÁC HÌNH VẼ

	Trang
Hình 1.1. Kiến trúc Android	4
Hình 1.2. Kiến trúc iOS	5
Hình 1.3. Kiến trúc Flutter	9
Hình 1.4. Minh họa Widget Flutter	10
Hình 1.5. Cấu trúc của NodeJs	19
Hình 2.1. Hình mô tả dữ liệu users trên cơ sở dữ liệu MongoDB	26
Hình 2.2. Hình mô tả dữ liệu categories trên cơ sở dữ liệu MongoDB	26
Hình 2.3. Hình mô tả dữ liệu products trên cơ sở dữ liệu MongoDB	27
Hình 2.4. Hình mô tả dữ liệu product_infos trên cơ sở dữ liệu MongoDB	28
Hình 2.5. Hình mô tả dữ liệu orders trên cơ sở dữ liệu MongoDB	29
Hình 2.6. Hình mô tả dữ liệu myorders trên cơ sở dữ liệu MongoDB	30
Bảng 2.8. Bảng dữ liệu carts	30
Hình 2.7. Hình mô tả dữ liệu carts trên cơ sở dữ liệu MongoDB	31
Hình 2.8. Quy trình đăng ký tài khoản	31
Hình 2.9. Quy trình đăng nhập	32
Hình 2.10. Quy trình xem chi tiết sản phẩm	32
Hình 2.11. Quy trình thêm sản phẩm vào giỏ hàng	33
Hình 2.12. Quy trình đặt mua hàng	33
Hình 2.13. Quy trình tìm kiếm sản phẩm	34
Hình 2.14. Quy trình xác nhận đơn hàng	34
Hình 3.1. Tạo tên project mới	36
Hình 3.2. Đặt tên project	37
Hình 3.3. Tạo cơ sở dữ liệu	37
Hình 3.4. Chọn lưu trữ đám mây cho cơ sở dữ liệu	38
Hình 3.5. Chọn kiểu kết nối	38
Hình 3.6. Chọn kết nối tới ứng dụng	39

Hình 3.7. Tạo mã kết nối	39
Hình 3.8. Cấu trúc dự án Flutter	43
Hình 3.9. Dữ liệu của ứng dụng	53
Hình 3.10. Giao diện đăng ký ứng dụng	54
Hình 3.11. Giao diện đăng nhập ứng dụng	54
Hình 3.12. Thêm địa chỉ thành công	55
Hình 3.13. Giao diện trang chủ ứng dụng	56
Hình 3.14. Danh sách sản phẩm	57
Hình 3.15. Chi tiết thông tin sản phẩm	58
Hình 3.16. Quản lý quá trình order sản phẩm người bán hàng	59
Hình 3.17. Lịch sử các sản phẩm đã mua	60
Hình 3.18. Danh sách sản phẩm đang bán	61
Hình 3.19. Danh sách sản phẩm đã được mua	62
Hình 3.20. Biểu đồ phân tích bán hàng	63

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Ngày nay những chiếc điện thoại di động thông minh đang dần mạnh mẽ và tiện lợi hơn, nó có thể thay thế những nhu cầu của con người trước đây trên các thiết bị vi tính. Theo thống kê hiện có hơn 1 triệu ứng dụng trong các cửa hàng ứng dụng lớn là App Store và Google Play, các doanh nghiệp trên toàn thế giới cũng đã nhận ra tầm quan trọng của ứng dụng di động trong công việc kinh doanh. Chính nền tảng Internet đã giúp cho nhiều hình thức giao dịch, phương pháp quản lý mới ra đời để thích ứng nền tảng và thời điểm trong đó có thương mại điện tử dạng app trên điện thoại. Với dạng app có nhiều ưu điểm trong quản lý tìm kiếm khách hàng mới, tăng trải nghiệm cho khách hàng hiện tại, tăng số lượng hàng hóa tiêu thụ. App trên di động giúp tương tác cao hơn với người dùng, theo dõi và nắm bắt nhu cầu thị trường và truyền thông bát chấp không gian và thời gian, tiết kiệm chi phí quảng cáo, dễ sử dụng và quản lý. Do vậy, việc thiết kế một ứng dụng bán hàng trên điện thoại di động là nhu cầu cần thiết. Tuy nhiên, việc chọn nền tảng nào để xây dựng nhằm mang lại lợi ích về thời gian và chi phí mà còn tạo ra một môi trường phát triển hiệu quả và dễ quản lý là vấn đề mà người lập trình cần cân nhắc. Lập trình đa nền (cross-platform development) và framework Flutter của Google đóng vai trò quan trọng trong việc xây dựng ứng dụng bán hàng trên điện thoại. Đồng thời, việc chia sẻ mã nguồn giữa các nền tảng trong Flutter giúp giảm sự phân tán và tăng tính duy trì của mã nguồn. Việc này làm cho quá trình phát triển và bảo trì ứng dụng trở nên hiệu quả và linh hoạt hơn. Tóm lại, sự kết hợp giữa lập trình đa nền và framework Flutter làm cho việc xây dựng ứng dụng bán hàng trên điện thoại trở nên linh hoạt, hiệu quả và ít tốn kém. Xuất phát từ đó, nhóm đề tài xây dựng ứng dụng bán hàng trên điện thoại di động nhằm khai thác các ưu điểm của nền tảng này.

2. Mục tiêu nghiên cứu

- Nghiên cứu công nghệ đa nền tảng Flutter, ngôn ngữ lập trình Dart và các công nghệ phát triển ứng dụng di động.
- Nghiên cứu nghiệp vụ bài toán bán hàng nói chung và bán hàng có sử dụng công nghệ nói riêng.

- Thiết kế giao diện, xử lý logic cho ứng dụng bán hàng dựa trên công nghệ phát triển ứng dụng di động đa nền tảng Flutter và biên dịch lên thiết bị di động.

3. Đối tượng nghiên cứu

- Framework đa nền tảng Flutter, kỹ thuật lập trình Web, kỹ thuật lập trình di động, ngôn ngữ lập trình Dart và cơ sở dữ liệu phi quan hệ NoSQL Mongo.
- Quy trình, nghiệp vụ bài toán bán hàng trực tuyến.

4. Phạm vi nghiên cứu

Đồ án nghiên cứu, xây dựng ứng dụng quản lý bán hàng trên điện thoại di động dựa trên Framework Flutter, cơ sở dữ liệu Mongo và các mặt hàng thuộc các danh mục: Điện thoại, sản phẩm thiết yếu, sản phẩm gia dụng, sản phẩm giáo dục, thời trang.

5. Phương pháp nghiên cứu

Phương pháp nghiên cứu tài liệu: Nghiên cứu các tạp chí khoa học, đồ án và tài liệu chuyên ngành.

Phương pháp thực nghiệm: Thiết kế, xây dựng, chạy thử nghiệm ứng dụng và đánh giá kết quả.

6. Ý nghĩa khoa học và thực tiễn của đồ án

Ý nghĩa khoa học: Đề xuất được phương pháp quản lý bán hàng trên điện thoại di động dựa trên kỹ thuật lập trình đa nền giúp tối ưu hóa quá trình phát triển bằng cách cho phép ứng dụng chạy trên nhiều hệ điều hành khác nhau mà không cần phải viết mã nguồn riêng lẻ cho từng nền tảng. Điều này tiết kiệm thời gian và nguồn lực, đồng thời giúp duy trì tính nhất quán giữa các phiên bản ứng dụng trên Windows, iOS và Android. Framework Flutter, với sự hỗ trợ mạnh mẽ từ ngôn ngữ lập trình Dart, cung cấp một cách tiếp cận hiệu quả cho việc phát triển ứng dụng di động giúp tăng tốc quá trình phát triển và kiểm thử.

Ý nghĩa thực tiễn: Xây dựng ứng dụng bán hàng trên điện thoại di động dựa trên kỹ thuật lập trình đa nền đảm bảo các yêu cầu nghiệp vụ của bài toán bán hàng và sử dụng công nghệ tối ưu, hiện đại.

7. Kết cấu của đồ án

Đồ án trình bày gồm 3 chương:

Chương 1. Cơ sở lý thuyết: Trình bày sơ lược về lịch sử nghiên cứu, các kỹ thuật và công nghệ làm cơ sở triển khai thực hiện đồ án; hướng đề xuất xây dựng ứng dụng bán hàng trên điện thoại mà đồ án cần thực hiện.

Chương 2. Phân tích, thiết kế hệ thống: Trình bày các bước phân tích, thiết kế ứng dụng bán hàng trên điện thoại.

Chương 3. Xây dựng ứng dụng bán hàng trên điện thoại trình bày các bước thiết kế phần mềm hệ thống với các chức năng được phân tích ở chương 2 và đánh giá kết quả triển khai thực nghiệm.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về kiến trúc di động

1.1.1. Kiến trúc hệ thống

Thiết bị di động hiện tại còn lại hai hệ điều hành di động phổ biến là iOS và Android. iOS hoạt động trên thiết bị iPhone và iPad, dùng ngôn ngữ Objective - C hoặc Swift với XCode hoặc các Framework đa nền tảng (Unity, Xamarin, Flutter...) và được Apple quản lý. Với hệ điều hành Android có sự đa dạng về mẫu mã và chủng loại của cả phần cứng và hệ điều hành, lập trình bằng Java dùng Android Studio hoặc các bên đa nền tảng khác (Unity, Xamarin, Flutter...). android có mã nguồn mở, các nhà sản xuất khác có thể tùy chỉnh được (như Samsung, HTC, Sony...)

Cả hai hệ điều hành, đều sử dụng chung một số mẫu kiến trúc: Layers (ba thành phần khác nhau), MVC (Model – View – Controller) hay MVVM (Model – View – ViewModel).

Model – View – Controller phân công ra:

- Tầng Model định danh ra những gì cần trả về cho người dùng
- Tầng Controller đưa ra các yêu cầu truy vấn tài nguyên
- Tầng View hiển thị lại các dữ liệu theo một chuẩn dễ hiểu cho người dùng.

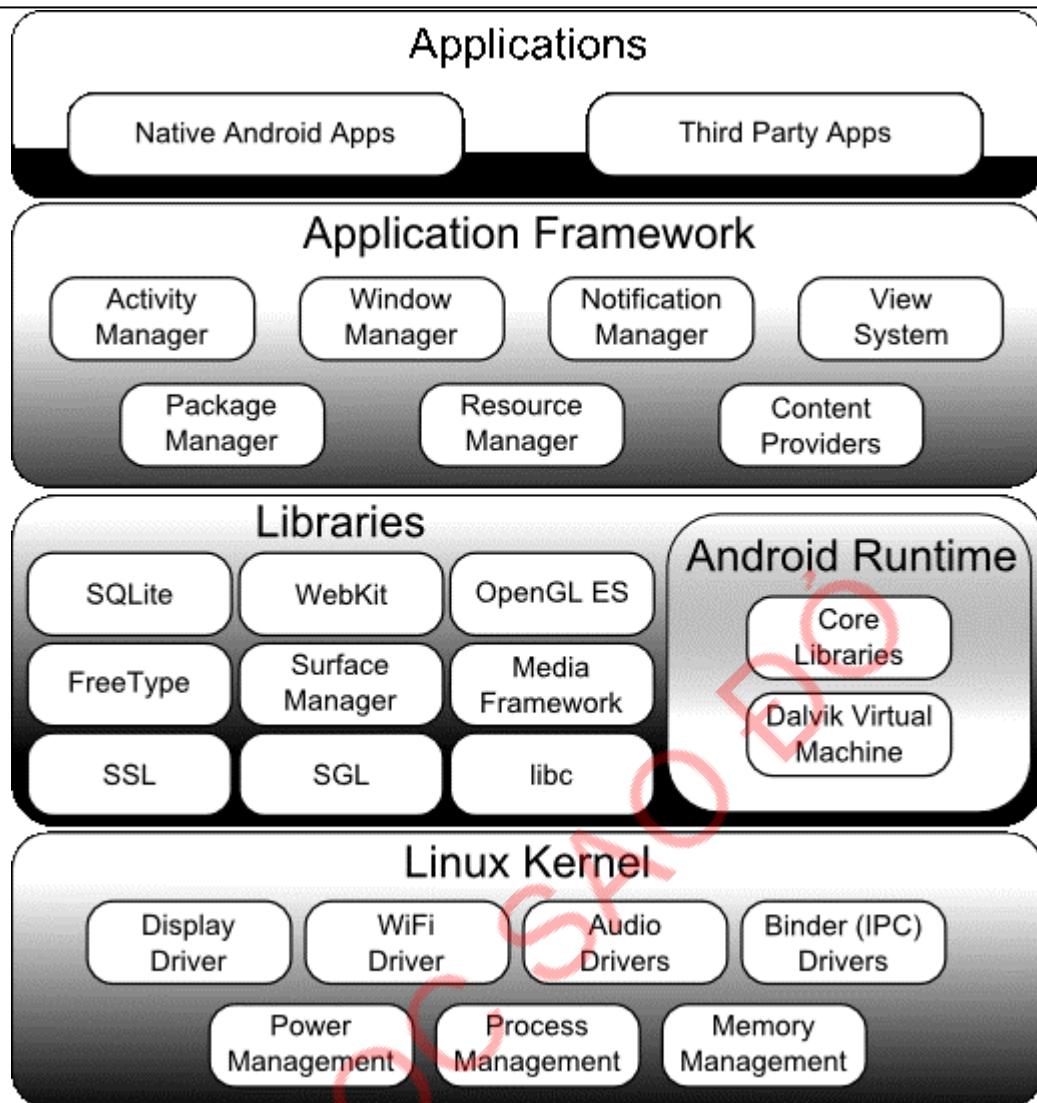
Model – View – ViewModel khá giống với MVC: ViewModel: Chứa các model và chuẩn bị các dữ liệu quan sát cho View. Nó cung cấp các cơ chế để truyền dữ liệu từ View sang Model.

Một cập nhật bên đối tượng của một lớp thuộc nhóm View sẽ cập nhật dữ liệu bên đối tượng của lớp thuộc nhóm Model và ngược lại (Data Binding)

Kiến trúc MVC, MVVM có thể áp dụng cho các ứng dụng không có kết nối Internet còn loại kiến trúc Client – Server để cập đến các ứng dụng có truy cập đến máy tính ở bên ngoài, qua HTTP Request hoặc Socket hoặc Webservice

1.1.2. Kiến trúc ứng dụng Android

Android được cấu trúc theo hình thức một hàng đợi phần mềm (software stack) bao gồm các ứng dụng (applications), hệ điều hành (operating system), môi trường thực thi (run-time environment), middleware, các dịch vụ (services), các thư viện (libraries). Hình sau minh họa kiến trúc của Android với các tầng (layer) và các thành phần tương ứng trong mỗi tầng trong một Android software stack:



Hình 1.1. Kiến trúc Android

Linux Kernel:

Cung cấp một mức trùu tượng giữa các thiết bị phần cứng và các tầng trên của stack. Dựa trên Linux phiên bản 2.6, kernel cung cấp đa nhiệm, các dịch vụ hệ thống mức thấp như bộ nhớ, tiến trình, các drivers như wifi, audio,...

Android Runtime (ART):

Khi thực thi một ứng dụng trong Android Studio, Android sẽ được biên dịch thành một mã bytecode trung gian (định dạng DEX). Khi ứng dụng được tải lên thiết bị, ART sẽ sử dụng một tiến trình gọi là Ahead-of-Time (AOT) để chuyển mã bytecode thành định dạng mã để bộ xử lý có thể hiểu được. Định dạng mã này được gọi là Executable and Linkable Format (ELF). Mỗi lần ứng dụng Android tải lên thiết bị, phiên bản thực thi ELF đã tạo lần đầu sẽ chạy mà không cần phải chuyển sang mã bytecode, do đó, ứng dụng thực thi nhanh hơn và làm tăng tuổi thọ pin của các thiết bị. Cách tiếp cận AOT ngược với cách biên dịch Just-in-Time (JIT) trong các phiên bản Andorid cũ hơn. Biên dịch JIT sẽ chuyển mã bytecode sang máy ảo mỗi lần ứng dụng

Android được tải lên thiết bị, do đó, sẽ xử lý chậm hơn rất nhiều so với cách biên dịch AOT.

Libraries (thư viện Android):

Android cung cấp một tập các thư viện đa dạng dùng trong đồ họa, cơ sở dữ liệu, giao tiếp mạng,... Các thư viện chủ yếu dự trên Java và C/C++. Một số thư viện chủ yếu gồm: android.app, android.database, android.net, android.opengl,...

Application Framework:

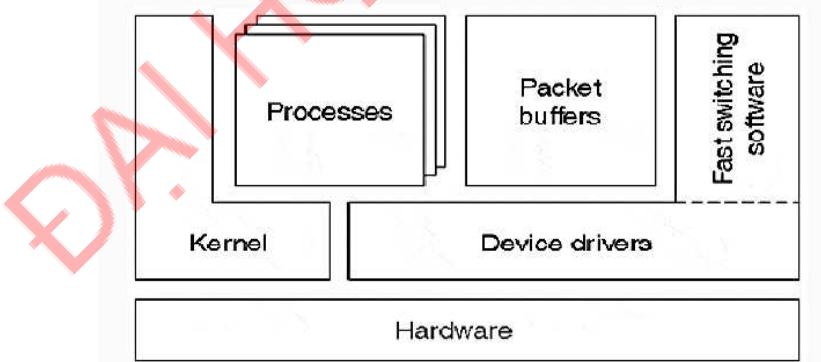
Application Framework là một tập các dịch vụ định hình nền môi trường trong đó các ứng dụng Android thực thi và được quản lý. Một số framework chủ yếu trong Android như Activity Manager, Content Providers, Resource Manager,....

Applications:

Là tầng cao nhất trong stack gồm các ứng dụng Android và các ứng dụng bên thứ ba do người dùng cài đặt trên thiết bị.

1.1.3. Kiến trúc ứng dụng iOS

iOS viết tắt của từ Internetwork Operating System, là một hệ điều hành hoạt động trên phần cứng của router Cisco, nó điều khiển hoạt động định tuyến và chuyển mạch của một router. Trên hệ điều hành iOS gồm có 3 phần: aaaa-bbbb-cccc trong đó aaaa là dòng sản phẩm áp dụng hệ điều hành này, bbbb là các chức năng của iOS, cc là định dạng file iOS, nơi iOS chạy. Giao diện người dùng của iOS dựa trên cơ sở thao tác bằng tay trên màn hình cảm ứng của các thiết bị Apple.



Hình 1.2. Kiến trúc iOS

Tiến trình (process): Là những tuyến riêng lẻ kết hợp với dữ liệu để thực hiện tác vụ như duy trì hệ thống, chuyển mạch gói dữ liệu, thực hiện giao thức định tuyến,...

Nhân (Kernel): Cung cấp dịch vụ cơ bản của hệ thống tùy thuộc vào iOS như quản lý bộ nhớ, lập lịch các tiến trình,... Nó cung cấp quản lý tài nguyên phần cứng (CPU, bộ nhớ) cho các tiến trình.

Bộ đệm gói (Packet buffer): Cung cấp các bộ đệm toàn cục và kết hợp với các chức năng quản lý bộ đệm để lưu trữ gói dữ liệu đang được chuyển mạch.

Trình điều khiển thiết bị (Device driver): Là chức năng điều khiển giao tiếp giữa phần cứng và thiết bị ngoại vi, giao tiếp giữa các tiến trình iOS, kernel và phần cứng. Chúng giao tiếp với phần mềm chuyển mạch nhanh (fast switching soft).

Phần mềm chuyển mạch nhanh (fast switching soft): Có chức năng chuyển mạch gói dữ liệu cao.

1.2. Tổng quan về công nghệ đa nền tảng

1.2.1. Định nghĩa về công nghệ đa nền tảng

Đa nền tảng là một thuật ngữ để chỉ các phần mềm hay phương thức điện toán được vận hành cùng nhau trên nhiều nền tảng. Như vậy, một phần mềm được gọi là đa nền tảng khi và chỉ khi nó có khả năng hoạt động trên nhiều hơn một hệ điều hành hay kiến trúc máy tính. Trong thời đại công nghệ số hiện nay, để phục vụ cho các đối tượng sử dụng nhiều nền tảng khác nhau đến từ iOS, Android hay Windows Phone, ta cần phải tạo ra các ứng dụng đa nền tảng (cross-platform hay multi-platform).

Để phát triển các ứng dụng dành cho thiết bị di động, Android cung cấp một framework gốc dựa trên ngôn ngữ Java và Kotlin, trong khi iOS cung cấp một framework dựa trên ngôn ngữ Objective - C/Swift. Cứ mỗi một nền tảng, lập trình viên phải sử dụng một ngôn ngữ tương ứng để tạo ra chương trình. Điều này gây bất tiện nếu muốn phát triển cùng 1 ứng dụng trên nhiều nền tảng khác nhau. Vì vậy, cần hai ngôn ngữ và framework khác nhau để phát triển ứng dụng cho cả hai hệ điều hành. Ngày nay, để khắc phục sự phức tạp này, có một số framework đã được giới thiệu hỗ trợ cả hệ điều hành cùng với các ứng dụng dành cho máy tính để bàn. Những loại framework này được gọi là công cụ phát triển đa nền tảng.

Trên thị trường hiện nay, có khá nhiều các ứng dụng đa nền tảng đang được sử dụng, nhưng thực tế chúng chỉ có 2 loại nhất định:

- Loại có thể chạy trực tiếp trên các nền tảng mà không cần đến sự hỗ trợ của trình biên dịch hay thông dịch. Những ứng dụng này thường được tạo ra bởi những ngôn ngữ thông dịch hay đã được dịch sẵn qua mã bytecode.

- Loại còn lại được tạo ra khá đơn giản, nhưng đòi hỏi chương trình đã có bước thiết kế hoặc biên dịch phù hợp với các nền tảng mà nó phục vụ.

Ứng dụng di động đa nền tảng (cross-platform mobile apps) là thuật ngữ để chỉ những ứng dụng di động được thiết kế chạy trên nhiều nền tảng di động khác nhau như: iOS, Android, Windows Phone, BlackBerry, WebOS... mà không cần phải lập trình nhiều lần cho từng nền tảng.

1.2.2. Ưu và nhược điểm của lập trình đa nền tảng

Ưu điểm của lập trình đa nền tảng:

- Sản phẩm được tạo ra từ lập trình đa nền tảng sẽ nhanh chóng tiếp cận được đến nhiều đối tượng người dùng hơn.
- Tiết kiệm chi phí cho chủ sở hữu khi thực hiện.
- Tiết kiệm thời gian công sức cho các lập trình viên trong việc xây dựng và phát triển ứng dụng phục vụ nhu cầu người dùng.
- Đa phần các ứng dụng được tạo ra bởi lập trình đa nền tảng đều ứng dụng những công nghệ hiện đại và tiên tiến, mang đến những trải nghiệm tốt nhất cho người dùng.
- Thuận tiện hơn trong việc triển khai và duy trì hoạt động hóa ứng dụng trên các nền tảng thiết bị.
- Các đoạn code xây dựng chương trình có thể tái sử dụng lại nhiều lần.
- Chương trình được xây dựng với cấu trúc liền mạch, tuân thủ chặt chẽ theo mô hình MVC hay MVVM, giúp chúng không phụ thuộc vào OS.

Nhược điểm của lập trình đa nền tảng:

- Việc lập trình đa nền tảng sẽ khó khai thác triệt để thư viện trong những nền tảng lớn, phổ biến là Android và iOS.
- Khó khăn trong việc đảm bảo chất lượng cho ứng dụng khi hoạt động trên các nền tảng khác nhau. Có thể là về giao diện hiển thị, công cụ hay ngôn ngữ trong nền tảng API và một số yêu cầu xử lý từ các bên liên quan.
- Bị giới hạn trong khả năng linh hoạt so với các ứng dụng được xây dựng dành riêng cho một nền tảng nhất định.

1.2.3. Một số công cụ hỗ trợ lập trình đa nền tảng

1.2.3.1. Flutter [1]

a. Định nghĩa

Framework phát triển đa nền tảng có khả năng viết một code và có thể triển khai trên nhiều nền tảng khác nhau (Android, iOS và máy tính để bàn), giúp tiết kiệm rất nhiều thời gian và nỗ lực phát triển của các nhà phát triển.

Có một số công cụ có sẵn để phát triển đa nền tảng, bao gồm các công cụ dựa trên web. Mỗi framework này có mức độ thành công khác nhau trong ngành công nghiệp di động. Gần đây, một framework công tác mới đã được giới thiệu trong họ phát triển đa nền tảng có tên là Flutter được phát triển từ Google. Flutter là một bộ công cụ giao diện người dùng để tạo các ứng dụng nhanh, đẹp, được biên dịch nguyên bản cho thiết bị di động, web và máy tính để bàn với một ngôn ngữ lập trình và cơ sở code duy nhất. Flutter miễn phí và code nguồn mở. Ban đầu, Flutter được phát triển từ

Google và bây giờ được quản lý theo tiêu chuẩn ECMA. Ứng dụng Flutter sử dụng ngôn ngữ lập trình Dart để tạo ứng dụng. Flutter chủ yếu được tối ưu hóa cho các ứng dụng di động 2D có thể chạy trên cả nền tảng Android và iOS. Chúng ta cũng có thể sử dụng Flutter để xây dựng các ứng dụng đầy đủ tính năng, bao gồm máy ảnh, bộ nhớ, vị trí địa lý, mạng, SDK của bên thứ ba,...Flutter khác với các framework khác vì nó không sử dụng WebView cũng như các widget OEM (Original Equipment Manufacturer) đi kèm với thiết bị. Thay vào đó, Flutter sử dụng công cụ kết xuất hiệu suất cao của riêng mình để vẽ các widget. Flutter cũng triển khai hầu hết các hệ thống như hoạt ảnh, cử chỉ và widget bằng ngôn ngữ lập trình Dart cho phép các nhà phát triển đọc, thay đổi, thay thế hoặc loại bỏ mọi thứ một cách dễ dàng.

b. Các tính năng của Flutter [2]

Flutter cung cấp các phương pháp dễ dàng và đơn giản để bắt đầu xây dựng các ứng dụng dành cho thiết bị di động và máy tính để bàn với bộ thiết kế material design và widget phong phú. Trong đồ án này đề cập về các tính năng chính của Flutter để phát triển framework di động.

- **Mã nguồn mở (Open Source):** Flutter là một framework code nguồn mở và miễn phí để phát triển các ứng dụng di động.
- **Đa nền tảng (Cross platform):** Tính năng này cho phép Flutter viết code một lần, duy trì và có thể chạy trên các nền tảng khác nhau nhằm tiết kiệm thời gian, công sức và tiền bạc của các nhà phát triển.
- **Tải lại ngay (Hot Reload):** Bất cứ khi nào nhà phát triển thực hiện thay đổi trong code, thì những thay đổi này có thể được nhìn thấy ngay lập tức với **Hot Reload**. Có nghĩa là những thay đổi hiển thị ngay lập tức trong chính ứng dụng và đây là một tính năng rất tiện dụng, cho phép nhà phát triển sửa các lỗi ngay lập tức.
- **Các tính năng và SDK gốc có thể truy cập (Accessible Native Features and SDKs):** Tính năng này cho phép quá trình phát triển ứng dụng dễ dàng thông qua code gốc của Flutter, tích hợp bên thứ ba và các API nền tảng. Do đó, có thể dễ dàng truy cập SDK trên cả hai nền tảng.
- **Code tối thiểu (Minimal code):** Ứng dụng Flutter được phát triển bởi ngôn ngữ lập trình Dart, sử dụng biên dịch JIT và AOT để cải thiện thời gian khởi động tổng thể, hoạt động và tăng tốc hiệu suất. JIT nâng cao hệ thống phát triển và làm mới giao diện người dùng mà không cần nỗ lực thêm vào việc xây dựng hệ thống mới.
- **Widget:** Framework Flutter cung cấp các widget có khả năng phát triển các thiết kế cụ thể có thể tùy chỉnh.

c. Kiến trúc của Flutter

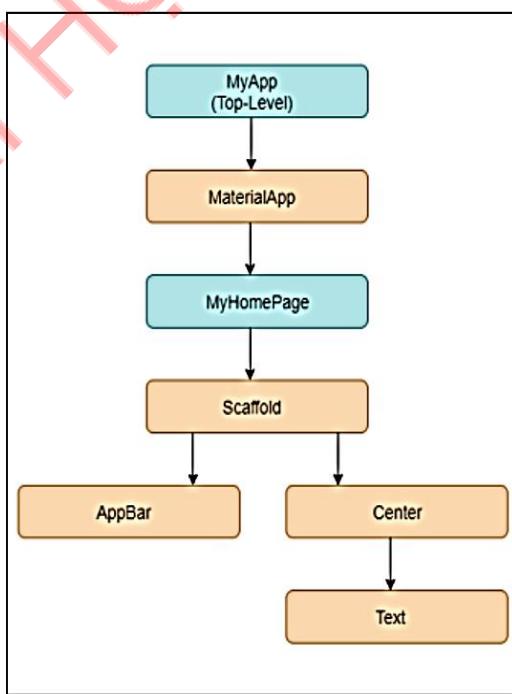
Kiến trúc Flutter chủ yếu bao gồm bốn thành phần: Flutter Engine, thư viện nền tảng (Foundation Library), Widgets và Design Specific Widgets.

Flutter Engine: Là một công để giúp chạy các ứng dụng di động chất lượng cao và cơ bản dựa trên ngôn ngữ C++. Flutter Engine triển khai các thư viện lõi Flutter bao gồm animation và đồ họa, tệp và mạng I / O, kiến trúc plugin, hỗ trợ trợ năng và thời gian chạy dart để phát triển, biên dịch và chạy các ứng dụng Flutter. Phải sử dụng thư viện đồ họa mã nguồn mở của Google, Skia, để hiển thị đồ họa cấp thấp.

Thư viện nền tảng (Foundation Library): Chứa tất cả các gói cần thiết cho các khôi build cơ bản để viết một ứng dụng Flutter. Các thư viện này được viết bằng ngôn ngữ Dart.

Widget: Trong Flutter, mọi thứ đều là một widget, đó là khái niệm cốt lõi của Framework. Widget trong Flutter về cơ bản là một thành phần giao diện người dùng ảnh hưởng và kiểm soát chế độ xem và giao diện của ứng dụng. Widget đại diện cho một mô tả bất biến về một phần của giao diện người dùng, bao gồm đồ họa, văn bản, hình dạng và Animation được tạo bằng các Widget. Các widget tương tự như các thành phần React.

Trong Flutter, bản thân ứng dụng là một Widget chứa nhiều Widget con. Điều đó có nghĩa rằng ứng dụng là tiện ích con cấp cao nhất và giao diện người dùng được xây dựng bằng cách sử dụng một hoặc nhiều tiện ích con giúp tạo một giao diện người dùng phức tạp rất dễ dàng.



Hình 1.3. Kiến trúc Flutter

Thiết kế các widget cụ thể:

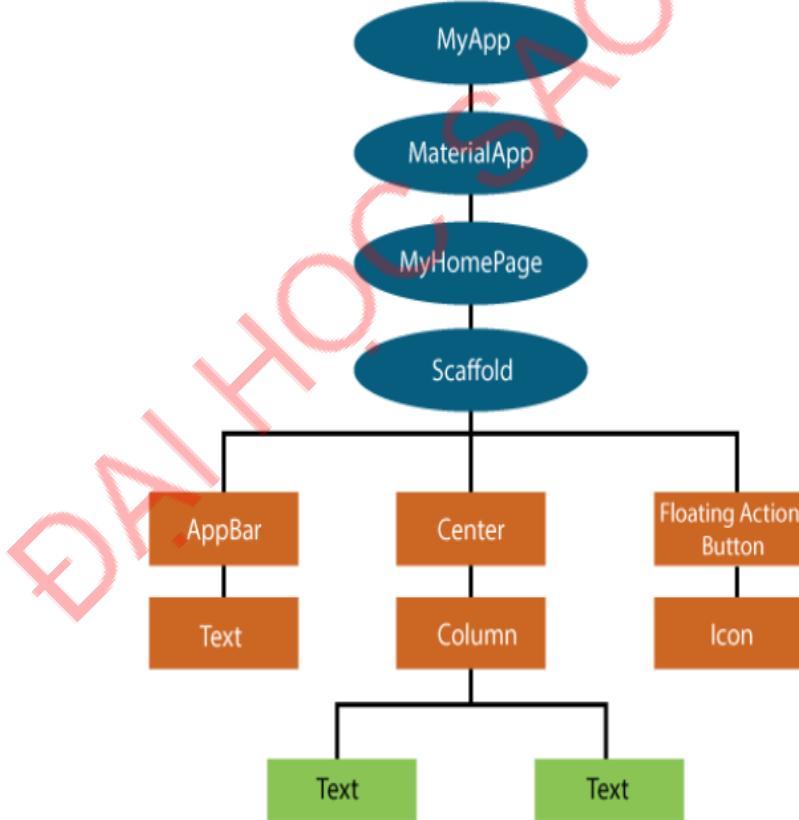
Framework Flutter có hai bộ widget phù hợp với các ngôn ngữ thiết kế cụ thể. Đây là Material Design cho ứng dụng Android và Cupertino Style cho ứng dụng IOS.

Một số loại Widgets của Flutter thường gặp [3]:

Widget Flutter:

Bất cứ khi nào viết mã để xây dựng đối tượng nào trong Flutter, đối tượng đó sẽ nằm trong một widget. Mục đích chính là xây dựng ứng dụng từ các widget, giúp mô tả cách độ xem ứng dụng như thế nào với cấu hình và trạng thái hiện tại của chúng.

Khi thực hiện bất kỳ thay đổi nào trong code, Widget con sẽ xây dựng lại mô tả của nó bằng cách tính toán sự khác biệt của Widget con hiện tại và trước đó để xác định những thay đổi tối thiểu hiển thị trong giao diện người dùng của ứng dụng. Các Widget được lồng vào nhau để xây dựng ứng dụng, có nghĩa là thư mục gốc của ứng dụng là một Widget và tất cả các cách nhìn xuống cũng là một Widget.



Hình 1.4. Minh họa Widget Flutter

Widget hiển thị:

Các widget hiển thị có liên quan đến dữ liệu đầu vào và đầu ra của người dùng. Một số loại quan trọng của Widget con này là:

Text

Text (văn bản) là một widget con trong Flutter cho phép hiển thị một chuỗi Text với một dòng duy nhất trong ứng dụng. Tùy thuộc vào các ràng buộc về bố cục, có thể ngắt chuỗi trên nhiều dòng hoặc tất cả có thể được hiển thị trên cùng một dòng. Nếu không chỉ định bất kỳ kiểu nào cho widget Text, nó sẽ sử dụng kiểu lớp TextStyleDefault gần nhất.

Button

Button (nút) là phần tử điều khiển đồ họa cung cấp cho người dùng kích hoạt một sự kiện như thực hiện hành động, lựa chọn, tìm kiếm,... Chúng có thể được đặt ở bất kỳ đâu trong giao diện người dùng như hộp thoại, biểu mẫu, thẻ, thanh công cụ.

Các nút là các Widget Flutter, là một phần của thư viện material design. Flutter cung cấp một số loại nút có hình dạng, kiểu dáng và tính năng khác nhau.

Image

Widget con này giữ hình ảnh có thể tìm nạp hình ảnh từ nhiều nguồn như từ thư mục nội dung hoặc trực tiếp từ URL. Nó cung cấp nhiều hàm tạo để tải hình ảnh, được đưa ra dưới đây:

- + Hình ảnh (Image): Đây là một trình tài hình ảnh chung, được sử dụng bởi ImageProvider .
 - + asset: Tải hình ảnh từ thư mục tài sản dự án.
 - + Tệp (file): Tải hình ảnh từ thư mục hệ thống.
 - + Bộ nhớ (memory): Tải hình ảnh từ bộ nhớ.
 - + Mạng (network): Tải hình ảnh từ mạng

Widget ẩn:

Các Widget vô hình có liên quan đến cách bố trí và kiểm soát các Widget. Widget ẩn cung cấp việc kiểm soát cách các Widget thực sự hoạt động và cách chúng sẽ hiển thị trên màn hình.

d. Layout trong Flutter

Các bước để bố trí Widget như sau:

Bước 1: Chọn một bố cục Widget.

Bước 2: Tạo một Widget hiển thị.

Bước 3: Thêm Widget hiển thị vào Widget layout.

Bước 4: Thêm Widget bố cục vào trang muốn hiển thị.

Các loại Widget layout

Widget đơn:

Widget layout con duy nhất là một loại widget, có thể chỉ có một widget bên trong widget bố cục mẹ. Các widget này cũng có thể chứa chức năng bố cục đặc biệt. Flutter cung cấp nhiều widget con để làm cho giao diện người dùng của ứng dụng trở nên hấp dẫn. Nếu sử dụng các widget này một cách thích hợp, nó có thể tiết kiệm thời gian và làm cho code ứng dụng dễ đọc hơn.

Widget đa:

Widget đa là một loại widget chứa nhiều hơn một widget con bên trong và cách bố trí của các widget này là duy nhất.

e. Quản lý State

Như đã tìm hiểu ở phần widget, mọi thứ trong ứng dụng flutter đều là một widget. Có thể phân loại widget này thành hai loại, một là widget không trạng thái (Stateless widget) và một là widget có trạng thái (Stateful widget). Widget không trạng thái không có bất kỳ trạng thái bên trong nào, có nghĩa là một khi nó được xây dựng, chúng ta không thể thay đổi hoặc sửa đổi cho đến khi chúng được khởi tạo lại. Mặt khác, widget Stateful là động và có trạng thái, có nghĩa là có thể sửa đổi một cách dễ dàng trong suốt vòng đời của nó mà không cần khởi động lại.

State (Trạng thái):

Trạng thái là thông tin có thể đọc được khi widget được tạo và có thể thay đổi hoặc sửa đổi trong suốt thời gian tồn tại của ứng dụng. Nếu muốn thay đổi widget, cần cập nhật đối tượng trạng thái, có thể được thực hiện bằng cách sử dụng hàm setState() có sẵn cho các widget Stateful. Hàm setState() cho phép thiết lập các thuộc tính của đối tượng trạng thái kích hoạt vẽ lại giao diện người dùng.

Quản lý state (trạng thái) là một trong những quy trình phổ biến và cần thiết nhất trong vòng đời của một ứng dụng. Theo tài liệu chính thức, Flutter mang tính chất khai báo. Điều đó có nghĩa là Flutter xây dựng giao diện người dùng bằng cách phản ánh trạng thái hiện tại của ứng dụng. Trong Flutter, quản lý trạng thái phân thành hai loại khái niệm:

- **Trạng thái tức thời:** Trạng thái này còn được gọi là trạng thái giao diện người dùng. Nó là một loại trạng thái có liên quan đến widget cụ thể, hoặc có thể nói rằng đó là một trạng thái chứa trong một widget duy nhất. Trong loại trạng thái này, người dùng không cần sử dụng các kỹ thuật quản lý trạng thái. Ví dụ phổ biến của trạng thái này là Text Field.

- **Trạng thái ứng dụng:** Khác với trạng thái tức thời. Đó là một loại trạng thái muốn chia sẻ trên các phần khác nhau của ứng dụng và muốn giữ lại giữa các phiên của người dùng. Trạng thái còn được gọi là trạng thái ứng dụng hoặc trạng thái chia sẻ. Một số ví dụ về trạng thái này là tùy chọn người dùng, thông tin đăng nhập, thông báo trong ứng dụng mạng xã hội, giỏ hàng trong ứng dụng thương mại điện tử, trạng thái đã đọc / chưa đọc của các bài báo trong ứng dụng tin tức,...

f. Navigator và Routing

Điều hướng và định tuyến (Navigation and Routing) là niệm cốt lõi của tất cả các ứng dụng di động, cho phép người dùng di chuyển giữa các trang khác nhau. Mọi ứng dụng di động đều chứa một số màn hình để hiển thị các loại thông tin khác nhau. Ví dụ một ứng dụng có thể có màn hình chứa nhiều sản phẩm khác nhau. Khi người dùng chạm vào sản phẩm đó sẽ hiển thị thông tin chi tiết về sản phẩm.

Trong Flutter, các màn hình và trang được gọi là các **tuyến** và các tuyến này chỉ là một widget. Trong Android, một tuyến tương tự như Activity, trong khi trong iOS, nó tương đương với ViewController.

Trong bất kỳ ứng dụng di động nào, điều hướng đến các trang khác nhau xác định quy trình làm việc của ứng dụng và cách xử lý điều hướng được gọi là **định tuyến**. Flutter cung cấp một lớp định tuyến cơ bản MaterialPageRoute và hai phương thức Navigator.push() và Navigator.pop() cho biết cách điều hướng giữa hai tuyến. Các bước điều hướng trong ứng dụng như sau.

Bước 1: Tạo hai tuyến.

Bước 2: Điều hướng đến một tuyến bằng cách sử dụng phương thức Navigator.push().

Bước 3: Điều hướng đến tuyến đầu tiên bằng cách sử dụng phương thức Navigator.pop().

1.2.3.2. React native [4]

React Native là một Framework phát triển ứng dụng di động đa nền tảng, cho phép xây dựng các ứng dụng di động bằng việc sử dụng ngôn ngữ lập trình JavaScript và sử dụng các thành phần giao diện được tái sử dụng. Framework này do cộng đồng Facebook và mã nguồn mở phát triển. Điểm nổi bật của React Native là khả năng chia sẻ mã nguồn giữa các nền tảng khác nhau như iOS và Android. Thay vì viết mã cho từng nền tảng một cách riêng biệt, người dùng có thể sử dụng một cơ sở mã nguồn chung để xây dựng ứng dụng trên cả hai hệ điều hành.

Cách hoạt động của React Native dựa trên nguyên tắc “learn once, write anywhere”. Người dùng sẽ sử dụng các thành phần giao diện được cung cấp bởi React Native để xây dựng giao diện người dùng của ứng dụng. React Native sẽ chuyển đổi

các thành phần này thành mã đặc tả cho nền tảng cụ thể, giúp hiển thị giao diện trên các thiết bị khác nhau.

Hoạt động của React Native:

React Native hoạt động bằng cách kết hợp sự mạnh mẽ của React – một thư viện giao diện người dùng dựa trên JavaScript. Bên cạnh đó, React Native còn sử dụng các thành phần native để hiển thị giao diện trên các nền tảng di động khác nhau. Cách thức React Native hoạt động như sau:

Khai báo Giao diện với JSX: Sử dụng JSX (JavaScript XML) để định nghĩa giao diện của ứng dụng. JSX cho phép viết mã tương tự HTML để tạo ra các thành phần giao diện như nút, văn bản, hình ảnh,...

Xây dựng Components: Xây dựng các thành phần giao diện bằng cách sử dụng các thành phần được cung cấp bởi React Native. Các thành phần này bao gồm các thành phần cơ bản như View, Text, Image,

Virtual DOM: React Native sử dụng một cơ chế gọi là Virtual DOM để theo dõi sự thay đổi trong giao diện. Virtual DOM là một phiên bản ảo của cây giao diện người dùng thật, được duyệt và cập nhật một cách hiệu quả để giảm thiểu việc tương tác trực tiếp với DOM thật trên các thiết bị di động.

Bridge và Native Modules: Khi các thay đổi trong Virtual DOM xảy ra, React Native sử dụng một thành phần được gọi là “Bridge” để giao tiếp với các API native của từng nền tảng (iOS và Android). Nếu cần, người dùng có thể sử dụng Native Modules để viết mã native cho các tính năng đặc biệt hoặc tối ưu hiệu suất.

Render thành phần Native: Sau khi React Native tạo ra các mã tương ứng với giao diện, nó sẽ sử dụng các thành phần native của từng nền tảng để hiển thị giao diện. Điều này giúp ứng dụng có hiệu suất gần như các ứng dụng native.

Hot Reloading và điều khiển trạng thái: React Native hỗ trợ tính năng hot reloading, cho phép người dùng thấy kết quả ngay lập tức khi thay đổi mã nguồn.

Ưu điểm:

- Có thể tái sử dụng code: React Native cho phép người dùng tái sử dụng code trong phát triển các ứng dụng đa nền tảng.

- Cộng đồng người dùng lớn: React Native được đánh giá là một trong những Framework có cộng đồng người dùng rất lớn trên toàn thế giới.

- Tính ổn định và tối ưu: Được phát triển bởi Facebook, React Native có hiệu năng ổn định khá cao. Mã React Native giúp đơn giản hóa quá trình xử lý dữ liệu. Trải nghiệm người dùng tốt hơn khi so sánh với ứng dụng Hybrids

Nhược điểm:

- Yêu cầu Native code.
- Hiệu năng kém hơn so với Native App.
- Bảo mật chưa thực sự tốt: Do dùng JavaScript, người dùng sẽ bị ảnh hưởng bởi những đặc điểm của JavaScript như dễ làm dễ sai dẫn đến khó duy trì sau.
- Tùy biến chưa thật sự tốt ở một số module.
- Không thích hợp cho các ứng dụng cần năng lực tính toán cao.

1.2.3.3. Ionic [5]

Ionic là một framework dùng để phát triển ứng dụng hybrid dựa trên HTML5. Một ứng dụng hybrid là một ứng dụng di động được xây dựng bằng các công nghệ phát triển web như HTML5, CSS, JavaScript (sự kết hợp giữa Angular JS và Cordova), được tạo bởi Max Lynk, Ben Sperry và Adam Bradley vào năm 2013. Các ứng dụng này sử dụng các thành phần hiển thị nội dung website (các trình duyệt ẩn như UIWebView trên iOS hay Webview trên Android và các thiết bị khác) để hiển thị các đoạn mã HTML. Ionic là một framework rất mạnh để viết các ứng dụng Hybrid, các đặc điểm có thể kể đến là:

- Ứng dụng hybrid đem lại nhiều lợi thế như khả năng hiển thị nội dung trên tất cả các thiết bị di động, cũng như tận dụng tối đa các tính năng khác của thiết bị di động như GPS, camera, danh sách liên lạc,... Bên cạnh đó, thời gian và chi phí dùng để tạo nên một ứng dụng hybrid cũng thấp hơn so với các ứng dụng di động thông thường.
- Ionic như là một bộ khung front-end giúp kiểm soát hình ảnh và trải nghiệm trên ứng dụng. Giống như “Bootstrap for Native”, nhưng với sự hỗ trợ của một lượng lớn các thành phần di động, hiệu ứng chuyển động mượt mà và thiết kế đẹp hơn.
- Không như các framework khác, Ionic có những thành phần giao diện và cách bố trí mang phong cách giống phong cách thiết kế mặc định trên thiết bị di động.
- Ionic là một framework HTML5 nên nó cần bộ đóng gói (wrapper) như Cordova hay PhoneGap để có thể chạy được như một ứng dụng di động. Cordova là bộ đóng gói mặc định trong Ionic framework.

Các tính năng của Ionic:

- Ionic sử dụng Angular JS để cung cấp các cấu trúc ứng dụng, trong khi bản thân ionic tập trung chính vào giao diện người dùng. Nói cách có sự phối hợp giữa sức mạnh của Angular JS và giao diện đẹp của Ionic UI.
- Ionic cung cấp một tập các Angular directives (các phần tử HTML tùy biến) để làm thành phần cho nó, do đó sử dụng các tiện ích gọn để viết mã HTML. Ngoài các

directives, Ionic còn sử dụng thêm vào các thành phần khác như Angular touch recognizers, view animation login, HTML sanitation và asynchronous communication.

- Việc xây dựng ứng dụng dựa trên nền Angular JS đòi hỏi mã nguồn phải có khả năng mở rộng cao để bổ sung các tính năng mới. Tuy nhiên với Ionic, người dùng có thể tái sử dụng các chức năng trong ứng dụng trên các nền tảng khác nhau đồng thời vẫn có thể tùy chỉnh giao diện người dùng cho mỗi nền tảng riêng biệt. Các thành phần trong ionic như danh sách, slide,... chính là các directives (các thuộc tính của thẻ HTML dùng trong Angular JS) đó chính là lý do khiến Ionic và Angular JS kết hợp tốt với nhau.

Giao diện người dùng:

Ionic dựa trên sự tồn tại của HTML5 và CSS3 để cung cấp những trải nghiệm nhanh chóng. Ionic không có giao diện người dùng sinh động, nhưng nó cung cấp giao diện một cách nhanh chóng và nhất quán. Sass là một ngôn ngữ mở rộng của CSS – Cascading Style Sheets – cho phép Ionic thêm các biến số và khả năng lồng cú pháp để mở rộng sự xuất hiện của giao diện ứng dụng. Ngoài ra, Ionic còn được đóng gói thêm các thư viện icon nguồn mở, khoảng 440 icon.

Giao diện dựng sẵn Widgets:

Ionic cung cấp một loạt các thành phần giao diện thiết kế sẵn. Các thành phần của Ionic đơn giản và mạnh mẽ. Chúng là các phần tử HTML phức hợp, được gọi là các directives, Ionic cũng cung cấp các Controller để bổ sung cho cấu hình và tương tác. Ionic cung cấp các khái niệm đơn giản mà có thể được kết hợp để cung cấp giao diện người dùng phong phú.

Ưu điểm:

- Dễ học, thời gian phát triển nhanh, có thể sử dụng các kỹ năng từ lập trình web.
- Đa nền tảng.
- Khả năng truy cập đến các tính năng của thiết bị và hệ điều hành như bluetooth, camera, cảm biến..
- Dễ dàng thiết kế giao diện cho các thiết bị có kích cỡ khác nhau.
- Việc sử dụng AngularJS làm core giúp phần xử lý UI linh động hơn so với javascript hay thư viện Jquery.
- Việc sử dụng AngularJS làm core cũng mang lại lợi thế lớn so với các framework cho ứng dụng hybrid khác.
- Ionic cung cấp đầy đủ các thành phần trong giao diện người dùng như Pull-to-Refresh, Infinite-loader, tabs,...

Nhược điểm:

- Vẫn còn đang trong giai đoạn phát triển.

- Hiệu năng vẫn chưa cao và ổn định.
- Cộng đồng phát triển ứng dụng còn chưa đông.

1.2.3.4. Xamarin [6]

Xamarin là một framework để xây dựng các ứng dụng di động đa nền tảng được tạo ra bởi hãng phần mềm di động cùng tên thành lập vào ngày 16 tháng 5 năm 2011 bởi các kỹ sư đã tạo ra Mono, Mono cho Android và MonoTouch. Xamarin sử dụng ngôn ngữ C# để viết các ứng dụng chạy trên Mac, Android, ý tưởng triển khai các ứng dụng đa nền tảng sử dụng Common Language Infrastructure (CLI) và Common Language Specifications (Microsoft .NET). Với cơ sở mã nguồn mở của C#, các lập trình viên có thể sử dụng các công cụ Xamarin để viết các ứng dụng native Android, iOS và Windows với giao diện người dùng native và sự chia sẻ code trên nhiều nền tảng, bao gồm Windows và macOS.

Ưu điểm:

Chia sẻ code ở mọi nơi: Khi viết một ứng dụng sử dụng bộ công cụ của Xamarin thì sử dụng một lớp trừu tượng phía trên các SDK thực sự của iOS và Android, nghĩa là thu được kết quả là một ứng dụng native hoàn toàn với giao diện dùng native trên mỗi nền tảng. Khi tạo ứng dụng trên Xamarin, sử dụng cùng ngôn ngữ, API và cấu trúc dữ liệu để chia sẻ trung bình 75% code trên tất cả các nền tảng điện thoại di động. Logic ứng dụng này có thể dễ dàng chia sẻ trên nhiều nền tảng giúp giảm đáng kể chi phí và thời gian phát triển ứng dụng di động.

Performance như native: Không giống như phương pháp kết hợp truyền thống dựa trên các công nghệ web, một ứng dụng đa nền tảng được xây dựng với Xamarin cũng có thể xem vào hàng native. Các số liệu performances là tương đương khi so sánh với các số liệu performance của Java, Kotlin cho Android và Objective-C hoặc Swift cho ứng dụng phát triển ứng dụng iOS native. Hơn thế nữa, performance liên tục được cải thiện để phù hợp hoàn toàn với tiêu chuẩn của lập trình native. Nền tảng Xamarin cung cấp là giải pháp để kiểm thử và theo dõi hoạt động của ứng dụng. Xamarin Test Cloud kết hợp với công cụ Xamarin Test Recorder cho phép chạy các UI test tự động và xác định các vấn đề về performance trước khi ứng dụng release.

Mỗi số ưu điểm khác như khả năng lựa chọn UI layout, tích hợp Oauth, tích hợp REST APIs từ xa, công nghệ dẫn đường và xử lý tín hiệu thời gian thực cho ứng dụng định vị, tích hợp mạng xã hội, có bộ cơ sở dữ liệu SQLite nhúng, thư viện XAML cho phép xây dựng một loạt ứng dụng và có hỗ trợ data binding.

Nhược điểm:

Ứng dụng thực hiện chậm hơn và yêu cầu nhiều dung lượng hơn trên thiết bị: Ứng dụng Xamarin lớn hơn, nặng hơn so với ứng dụng native. So sánh với ứng dụng

native nó chiếm nhiều hơn vài Mb so với Java/Objective-C tương ứng. Kích thước của một ứng dụng code bằng xamarin là 3Mb, trong khi code bằng Objective C chỉ chiếm 172 Kb, càng sử dụng nhiều API, càng nhiều lưu trữ bị chiếm trên thiết bị.

Nhược điểm của AOT Compiler: Xamarin Forms cũng có những khuyết điểm AOT compiler. Nó không được compile các đoạn code gọn gàng như Xcode, không hỗ trợ sinh code tự động, thời gian Visual Studio build code lâu.

Cộng đồng hỗ trợ ít: Cộng đồng Xamarin ít hơn so với cộng đồng của riêng iOS hay Android.

1.3. Tổng quan về NodeJs và NPM

1.3.1. Giới thiệu

NodeJS là một nền tảng Server side được xây dựng dựa trên Javascript Engine (V8 Engine). Node.js được phát triển bởi Ryan Dahl năm 2009, là một nền tảng dựa vào Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. Node.js sử dụng các phần phát sinh các sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.

NodeJs là một mã nguồn mở, đa nền tảng cho phát triển các ứng dụng phía Server và các ứng dụng liên quan đến mạng. Ứng dụng Node.js được viết bằng Javascript và có thể chạy trong môi trường Node.js trên hệ điều hành như hệ điều hành Window, Linux...

Node.js cũng cung cấp cho chúng ta các module Javascript đa dạng, có thể đơn giản hóa sự phát triển của các ứng dụng web sử dụng Node.js với các phần mở rộng

1.3.2. Đặc điểm của Node.js

- **Không đồng bộ và phát sinh sự kiện (Event Driven):** Tất cả các APIs của thư viện Node.js đều không đồng bộ, nghĩa là không blocking (khóa). Nó rất cần thiết vì Node.js không bao giờ đợi một API trả về dữ liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về Sự kiện của Node.js giúp Server nhận phản hồi từ các API gọi trước đó.

- **Chạy nhanh:** Dựa trên V8 Javascript Engine của Google Chrome, thư viện Node.js rất nhanh trong các quá trình thực hiện code.

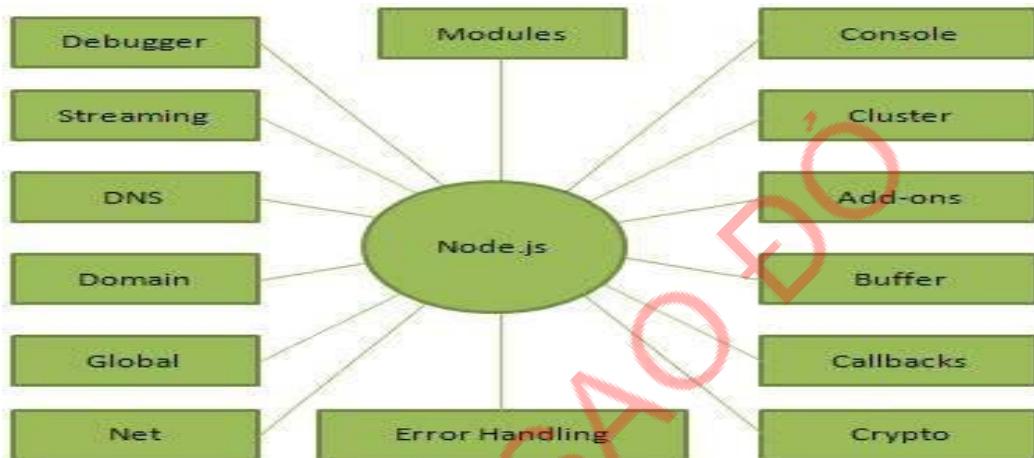
- **Các tiến trình đơn giản nhưng hiệu năng cao:** Node.js sử dụng một mô hình luồng đơn (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp Server trả lại các phản hồi với một cách không khóa và tạo cho Server hiệu quả cao ngược lại với

các cách truyền thống tạo ra một số lượng luồng hữu hạn để quản lý request. Nodejs sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.

- **Không đệm:** Ứng dụng Node.js không lưu trữ các dữ liệu buffer.
- **Có giấy phép:** Node.js được phát hành dựa vào MIT License.

1.3.3. Các thành phần trong Node.js

Sơ đồ dưới đây mô tả các thành phần quan trọng của Node.js:



Hình 1.5. Cấu trúc của NodeJs

NPM viết tắt của Node Package Manager là một công cụ (chương trình) quản lý các thư viện lập trình Javascript cho Node.js, công cụ này là thật sự cần thiết cho thế giới mã nguồn mở. Trong cộng đồng Javascript, các lập trình viên chia sẻ hàng trăm nghìn các đoạn code giúp cho các dự án mới tránh phải viết lại các thành phần cơ bản, các thư viện lập trình hay thậm chí cả các framework. Mỗi đoạn code này có thể phụ thuộc vào rất nhiều các mã nguồn mở khác, công cụ quản lý thư viện NPM ra đời nhằm giảm công sức trong việc quản lý các thư viện này.

1.4. Kết luận chương 1

Chương 1 trình bày về công nghệ đa nền tảng và kiến trúc ứng dụng di động đưa ra một cái nhìn tổng quan về những khía cạnh quan trọng của phát triển ứng dụng hiện đại. Công nghệ đa nền tảng được đề xuất như một phương pháp linh hoạt, giúp giảm chi phí phát triển và tăng cường khả năng triển khai trên nhiều hệ điều hành và thiết bị khác nhau. Ngoài ra, chương đi sâu vào các công nghệ và framework quan trọng như Node.js và NPM, cung cấp môi trường và công cụ quản lý mã nguồn mở cho phát triển ứng dụng mạng. Các framework như React Native, Xamarin, Flutter và Ionic cung cấp các phương pháp tiếp cận khác nhau cho việc xây dựng ứng dụng di động đa nền tảng. Sự linh hoạt của React Native, tính gần như native của Xamarin, hiệu suất cao của Flutter và dễ học của Ionic đều được nhấn mạnh như những lựa chọn

quan trọng cho các nhà phát triển ứng dụng di động. Chương này là nguồn thông tin quan trọng giúp độc giả hiểu rõ hơn về các xu hướng và công nghệ đang làm thay đổi cảnh quan phát triển ứng dụng di động ngày nay.

DÀI HỌC SAO ĐỎ

CHƯƠNG 2. PHÂN TÍCH, THIẾT KẾ HỆ THỐNG

2.1. Phân tích yêu cầu

2.1.1. Phát biểu bài toán

Mục tiêu bài toán đặt ra là xây dựng được ứng dụng bán hàng trên điện thoại thông minh sử dụng Framework đa nền tảng Flutter thỏa mãn các mục tiêu đề ra. Chức năng cơ bản gồm:

- Xem sản phẩm
- Giỏ hàng
- Thanh toán và đặt hàng
- Quản lý tài khoản
- Thông báo và cập nhật
- Đánh giá và nhận xét
- Theo dõi vận chuyển
- Hỗ trợ trực tuyến

Các yêu cầu phi chức năng đề xuất như:

- Trải nghiệm người dùng (UI/UX)
- Hiệu suất
- Lưu trữ dữ liệu
- Độ bảo mật

2.1.2. Quy trình tương tác

Quy trình chỉ ra quá trình tương tác giữa các thành phần trong hệ thống. Các yêu cầu HTTP có thể được sử dụng để giao tiếp giữa Flutter và API, trong khi MongoDB cung cấp cơ sở dữ liệu không có cấu trúc cho ứng dụng. Đồng thời, các lớp xử lý và lớp dịch vụ có thể được thêm vào để quản lý logic nghiệp vụ và tương tác giữa các thành phần. Quy trình tương tác giữa Flutter, API, và MongoDB cho ứng dụng bán hàng trên điện thoại có thể được mô tả như sau:

Flutter App:

- *Chức năng giao diện người dùng (UI)*: Flutter xây dựng giao diện người dùng cho ứng dụng di động, hiển thị danh sách sản phẩm, giỏ hàng, ...
- *Xử lý sự kiện*: Flutter xử lý sự kiện người dùng như nhấn nút, thêm sản phẩm vào giỏ hàng, thực hiện thanh toán và gửi các yêu cầu đến API.

API (Application Programming Interface):

- *Quản lý sản phẩm và người dùng*: API cung cấp các endpoint để quản lý thông tin sản phẩm, người dùng, đơn hàng,...
- *Xử lý yêu cầu*: API xử lý các yêu cầu từ ứng dụng Flutter, thực hiện các tác vụ như đọc và ghi vào cơ sở dữ liệu, xác thực người dùng và trả lại dữ liệu cho ứng dụng.

MongoDB:

- *Lưu trữ dữ liệu*: MongoDB lưu trữ dữ liệu như thông tin sản phẩm, thông tin người dùng, lịch sử đơn hàng,...

- *Kết nối với API*: API sử dụng MongoDB để đọc và ghi dữ liệu. Các thao tác CRUD (Create, Read, Update, Delete) được thực hiện thông qua API để tương tác với cơ sở dữ liệu.

Quá trình tương tác:

- *Xem sản phẩm và giỏ hàng*: Ứng dụng Flutter gửi yêu cầu lấy danh sách sản phẩm và thông tin giỏ hàng đến API. API truy xuất dữ liệu từ MongoDB và trả về kết quả cho ứng dụng.

- *Thanh toán và đặt hàng*: Khi người dùng thực hiện thanh toán, ứng dụng Flutter gửi yêu cầu đặt hàng và thanh toán đến API. API xác nhận đơn hàng, cập nhật trạng thái và thông tin liên quan trong MongoDB.

2.2. Phân tích các đối tượng và chức năng hệ thống**2.2.1. Đối tượng hệ thống**

Thông qua nghiên cứu mục tiêu, các tài liệu liên quan, đồ án đã phân tích yêu cầu bài toán như sau:

Bảng 2.1. Chức năng của quản trị và khách hàng

Đối tượng	Chức năng
Quản trị	Quản lý tài khoản và đăng nhập: Bảo vệ ứng dụng bằng cách yêu cầu xác thực đăng nhập.
	Quản lý thông tin cửa hàng: Thay đổi thông tin liên quan đến cửa hàng như địa chỉ, số điện thoại và logo,...
	Quản lý sản phẩm (thêm, sửa, xóa sản phẩm).
	Quản lý danh mục sản phẩm (thêm, sửa, xóa danh mục sản phẩm).
	Quản lý khách hàng.
	Quản lý đơn hàng: Xem đơn hàng, xác nhận đơn hàng.
	Quản lý nội dung trang chủ
	Quản lý tiến trình đặt mua sản phẩm: Giai đoạn đã đặt đơn, đã gửi hàng, đang giao hàng, đã giao hàng.

Đối tượng	Chức năng
Khách hàng	Đăng ký tài khoản trên ứng dụng.
	Đăng nhập ứng dụng bằng tài khoản đã đăng ký.
	Xem các sản phẩm đang bán (còn hàng).
	Xem các sản phẩm đã mua.
	Xem lịch sử các sản phẩm đã mua
	Xem chi tiết các sản phẩm.
	Đặt mua sản phẩm.
	Xem các đơn đặt hàng.

2.2.2. Mô tả chức năng chính

- Khách hàng:
 - **Đăng ký tài khoản:** Là một trong những tính năng quan trọng của hệ thống. Khi người dùng đăng ký tài khoản cần phải thực hiện nhập đầy đủ các thông tin được hiển thị trên màn hình ứng dụng. Sau đó, người dùng nhấn nút “Đăng ký” thì hệ thống sẽ kiểm tra thông tin đăng ký và cập nhật vào hệ thống.
 - **Đăng nhập:** Sau khi đăng ký thành công, người dùng có thể sử dụng tài khoản đăng nhập vào ứng dụng bằng cách nhập đúng Email và Password và chọn “Đăng nhập”. Nếu có sai sót màn hình sẽ thông báo thất bại, và ngược lại sẽ chuyển đến màn hình chính.
 - **Xem sản phẩm theo danh mục:** Nếu người dùng chọn danh mục sản phẩm, màn hình sẽ hiển thị tất cả các sản phẩm theo danh mục đã chọn.
 - **Xem chi tiết sản phẩm:** Khi người dùng chọn vào sản phẩm thì màn hình chi tiết sản phẩm sẽ hiển thị cho kết quả tên sản phẩm, giá sản phẩm, mô tả sản phẩm, trạng thái sản phẩm (còn hoặc hết).
 - **Thêm sản phẩm vào giỏ hàng:** Trên trang xem chi tiết sản phẩm sẽ có ô Spinner để người dùng chọn số lượng sản phẩm mong muốn, và khi người dùng nhấn vào nút “Thêm vào giỏ hàng” thì sản phẩm sẽ được đưa vào giỏ hàng.
 - **Đặt mua sản phẩm:** Khi người dùng chọn vào giỏ hàng thì sản phẩm sẽ được thêm vào giỏ hàng, giỏ hàng sẽ hiển thị tên sản phẩm giá sản phẩm, số lượng sản phẩm, tổng từng sản phẩm, tổng giỏ hàng. Thông tin khách hàng, đơn hàng sẽ được quản lý trên hệ thống quản trị.

- **Xem đơn hàng:** Người dùng xem các thông tin của sản phẩm người dùng chọn mua đều nằm trong giỏ hàng, tổng tiền từng sản phẩm, và tổng tiền đơn hàng đều được hiển thị.
- **Xem lịch sử mua hàng:** Người dùng xem lịch sử các sản phẩm đã mua và trạng thái của giao dịch: Đang giao, đã đặt hàng,...
- Quản trị:
 - **Quản lý danh mục sản phẩm:** Thêm, sửa, xóa các sản phẩm trong cửa hàng hiện có.
 - **Quản lý sản phẩm:** Thêm, sửa, xóa các sản phẩm trong cửa hàng hiện có.
 - **Quản lý khách hàng:** Xem thông tin của khách hàng.
 - **Quản lý đơn hàng:** Xem thông tin của đơn hàng mà khách hàng đã đặt mua, đồng thời xác nhận đơn hàng cho khách hàng.
 - **Quản lý tài khoản và đăng nhập:** Bảo vệ ứng dụng bằng cách yêu cầu xác thực đăng nhập.
 - **Quản lý thông tin cửa hàng:** Thay đổi thông tin liên quan đến cửa hàng như địa chỉ, số điện thoại và logo,...
 - **Quản lý nội dung trang chủ.**

2.2.3. Mô tả yêu cầu thiết bị và phần mềm

Yêu cầu thiết bị:

Phần cứng: Bất kỳ máy tính hoặc laptop nào có thể chạy môi trường phát triển Flutter. Ít nhất 8GB RAM được khuyến nghị để đảm bảo hiệu suất ổn định khi xây dựng và chạy ứng dụng Flutter.

Yêu cầu hệ điều hành: Windows, macOS hoặc Linux.

Kết nối Internet ổn định: Để cài đặt các gói phần mềm, tải về dependencies.

Thiết bị di động có thể kết nối và một số yêu cầu khác như USB debugging được bật.

Yêu cầu phần mềm:

Flutter SDK:

Cài Đặt Flutter: Tải và cài đặt Flutter SDK từ trang chính thức của Flutter: [Flutter SDK](#)

Editor/IDE: Chọn một trình soạn thảo mã nguồn hoặc môi trường phát triển tích hợp (IDE) để phát triển ứng dụng Flutter. Một số lựa chọn phổ biến bao gồm: Visual Studio Code, Android Studio, IntelliJ IDEA. Trong đồ án này sử dụng IDE Android Studio.

MongoDB:

Cài Đặt MongoDB: Tải và cài đặt MongoDB từ trang chính thức của MongoDB: MongoDB Community Edition

API Server: Có thể sử dụng một số ngôn ngữ và framework để xây dựng API Server, ví dụ Node.js với Express, Python với Flask hoặc Django, Java với Spring Boot. Trong đồ án này sử dụng Node.js.

Yêu cầu khác:

Tài khoản mongodb Atlas (Nếu cần): Nếu không cài đặt MongoDB trực tiếp trên máy tính, có thể đăng ký tài khoản MongoDB Atlas để sử dụng MongoDB trong môi trường đám mây.

2.3. Đè xuất dữ liệu

Về lưu trữ dữ liệu, bài toán đặt ra là cần lưu trữ thông tin các sản phẩm, thông tin về người dùng và tình trạng mua bán hàng trên ứng dụng. Trong đồ án này sử dụng MongoDB để quản lý dữ liệu.

Để quản lý khách hàng mua sản phẩm cần liên hệ và chăm sóc sau bán hàng, thông tin được thể hiện trong bảng 2.2.

Bảng 2.2. Bảng dữ liệu users

STT	Tên trường	Kiểu dữ liệu	Mô tả
1.	id	String	Mã người dùng.
2.	name	String	Tên người dùng.
3.	password	String	Mật khẩu của người dùng.
4.	email	String	Địa chỉ email người dùng.
5.	type	String	Loại người dùng.
6.	token	String	Mã thông báo để xác thực người dùng sau khi đăng nhập.
7.	cart	List	Giỏ hàng của người dùng.

Dữ liệu về users được lưu trong MongoDB như sau:

```

{
  "_id": ObjectId("6555d5fd1632e59081fb7532"),
  "name": "anh123",
  "email": "anh123@gmail.com",
  "password": "$2a$08$tEnxRz0a2298pgd6A7tUC.P./jix319V1/ExGTC5mdxMdaQLPo9Cu",
  "address": "a,b,1 - 23123",
  "type": "user",
  "cart": Array (empty)
}
{
  "_id": ObjectId("6555ed249e9e030bc9b30dd5"),
  "name": "anh1",
  "email": "anh12345@gmail.com"
}

```

Hình 2.1. Hình mô tả dữ liệu users trên cơ sở dữ liệu MongoDB

Để quản lý danh mục sản phẩm trên MongoDB cần có các thông tin thể hiện trong bảng 2.3.

Bảng 2.3. Bảng dữ liệu categories

STT	Tên trường	Kiểu dữ liệu	Mô tả
1.	id	int	Mã danh mục sản phẩm.
2.	category	string	Tên danh mục sản phẩm.

Dữ liệu về categories được lưu trong MongoDB như sau:

Hình 2.2. Hình mô tả dữ liệu categories trên cơ sở dữ liệu MongoDB

Để quản lý sản phẩm trên ứng dụng cần các thông tin thể hiện trong bảng 2.4.

Bảng 2.4. Bảng dữ liệu products

STT	Tên trường	Kiểu dữ liệu	Mô tả
1.	id	String	Mã sản phẩm được tạo tự động
2.	name	String	Tên sản phẩm
3.	description	String	Mô tả sản phẩm
4.	quantity	double	Số lượng sản phẩm
5.	images	List	Chuỗi là link hình ảnh
6.	category	String	Loại sản phẩm
7.	price	double	Giá sản phẩm
8.	rating	List	Đánh giá sản phẩm

Dữ liệu về products được lưu trong MongoDB như sau:

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with databases like test, carts, categories, myorders, orders, product_infos, products, and users. The 'products' database is selected. On the right, the main area shows the 'test.products' collection. It displays storage details (STORAGE SIZE: 4KB, LOGICAL DATA SIZE: 0B, TOTAL DOCUMENTS: 0, INDEXES TOTAL SIZE: 4KB), navigation tabs (Find, Indexes, Schema Anti-Patterns, Aggregation, Search Indexes), a query input field ('Type a query: { field: 'value' }'), and a results section ('QUERY RESULTS: 0'). A large red stamp 'ĐÃ TÌM THẤY' is overlaid across the entire interface.

Hình 2.3. Hình mô tả dữ liệu products trên cơ sở dữ liệu MongoDB

Chi tiết sản phẩm trên ứng dụng cần các thông tin thể hiện trong bảng 2.5.

Bảng 2.5. Bảng dữ liệu product_infos

STT	Tên trường	Kiểu dữ liệu	Mô tả
1.	id	String	Mã sản phẩm được tạo tự động

STT	Tên trường	Kiểu dữ liệu	Mô tả
2.	name	String	Tên sản phẩm
3.	description	String	Mô tả sản phẩm
4.	quantity	double	Số lượng sản phẩm
5.	images	List	Chuỗi là link hình ảnh
6.	category	String	Loại sản phẩm
7.	price	double	Giá sản phẩm
8.	rating	List	Đánh giá sản phẩm

Dữ liệu về product_infos được lưu trong MongoDB như sau:

The screenshot shows the MongoDB Compass interface with the database 'test' selected. Under the 'test' namespace, the 'product_infos' collection is highlighted. The interface includes a sidebar with other collections like carts, categories, myorders, orders, products, and users. The main area shows the collection details: storage size 36KB, logical data size 2.75KB, total documents 8, and indexes total size 36KB. It features tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. An 'INSERT DOCUMENT' button is at the top right. A search bar at the top says 'Type a query: { field: 'value' }'. Below it, a 'QUERY RESULTS: 1-8 OF 8' section lists two documents. The first document has an _id of '6564a79ff2a44280a92c1a34', name 'Áo Hoodie Đỏ', description 'Phong cách thời trang đường phố', images [1], quantity 100, price 12, category 'Essentials', and ratings [empty]. The second document has an _id of '6564a7e3f2a44280a92c1a45', name 'Balo du lịch', and images [empty]. Buttons for Reset, Apply, and Options are at the bottom right of the results table.

Hình 2.4. Hình mô tả dữ liệu product_infos trên cơ sở dữ liệu MongoDB

Việc quản lý đơn mua hàng của khách hàng gồm các thông tin trong bảng 2.6.

Bảng 2.6. Bảng dữ liệu orders

STT	Tên trường	Kiểu dữ liệu	Mô tả
1.	id	String	Mã đặt hàng.
2.	userId	String	Mã của người dùng trong đơn hàng.
3.	products	List	Sản phẩm trong đơn hàng.

STT	Tên trường	Kiểu dữ liệu	Mô tả
4.	quantity	List	Danh sách chứa các giá trị số nguyên, đại diện cho số lượng tương ứng của từng sản phẩm trong danh sách.
5.	address	String	Địa chỉ giao hàng của đơn hàng.
6.	orderedAt	int	Thời điểm đặt hàng.
7.	status	int	Trạng thái của đơn hàng.
8.	totalPrice	double	Tổng giá trị của đơn hàng.

Dữ liệu về orders được lưu trong MongoDB như sau:

Hình 2.5. Hình mô tả dữ liệu orders trên cơ sở dữ liệu MongoDB

Chi tiết việc mua hàng của khách hàng gồm chi tiết đơn đặt hàng thể hiện trong bảng 2.7.

Bảng 2.7. Bảng dữ liệu myorder

STT	Tên trường	Kiểu dữ liệu	Mô tả
1.	id	String	Mã đặt hàng được tạo ra từ khi người tiêu dùng đặt đơn.
2.	products	List	Sản phẩm trong đơn hàng.
3.	totalprices	double	Tổng giá trị của đơn hàng.

STT	Tên trường	Kiểu dữ liệu	Mô tả
4.	address	String	Địa chỉ giao hàng của đơn hàng.
5.	userid	String	Mã của người dùng trong đơn hàng.
6.	orderedAt	int	Thời điểm đặt hàng.
7.	status	int	Trạng thái của đơn hàng.

Dữ liệu về myorders được lưu trong MongoDB như sau:

Hình 2.6. Hình mô tả dữ liệu myorders trên cơ sở dữ liệu MongoDB

Chi tiết việc mua hàng của khách hàng gồm chi tiết đơn đặt hàng thể hiện trong bảng 2.8.

Bảng 2.8. Bảng dữ liệu carts

STT	Tên trường	Kiểu dữ liệu	Mô tả
1.	userId	String	Mã của người dùng trong giỏ hàng.
2.	product	List	Sản phẩm trong giỏ hàng.
3.	quantity	List	Số lượng tương ứng của từng sản phẩm trong giỏ hàng.

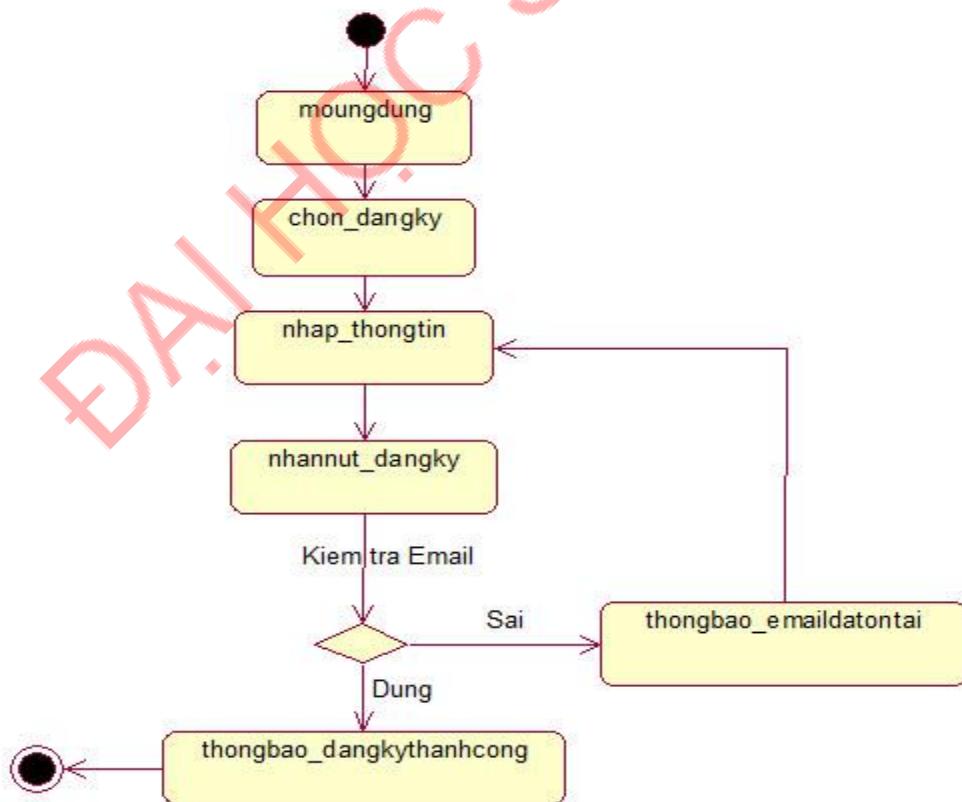
Dữ liệu về carts được lưu trong MongoDB như sau:

Hình 2.7. Hình mô tả dữ liệu carts trên cơ sở dữ liệu MongoDB

2.4. Phân tích một số quy trình trong thiết kế ứng dụng

2.4.1. Quy trình đăng ký tài khoản

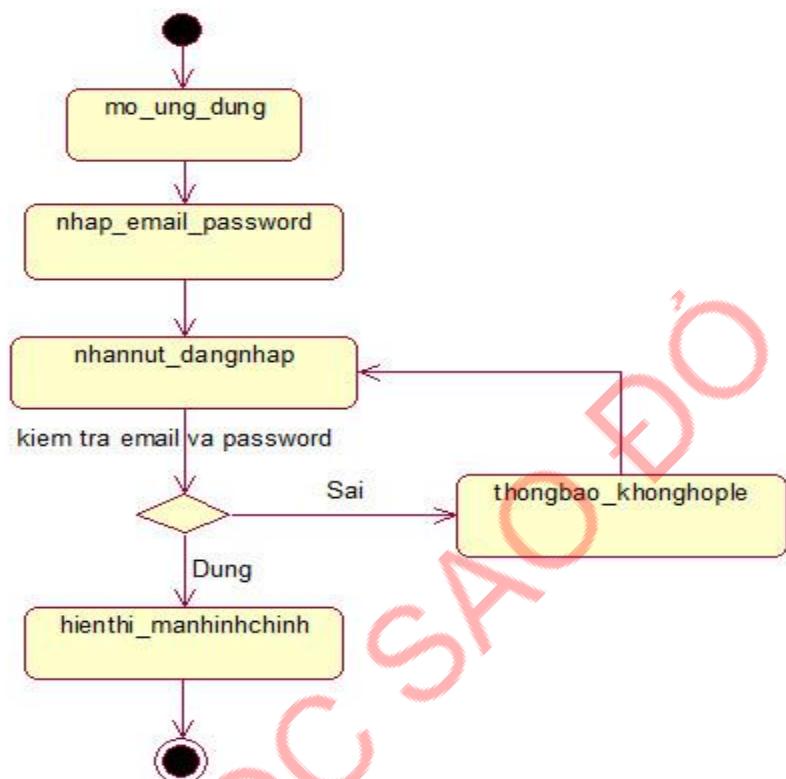
Hoạt động đăng ký diễn ra khi người dùng mở ứng dụng và chọn nút “Đăng ký”, sau đó nhập đầy đủ thông tin theo màn hình hiển thị, rồi nhấn nút “Đăng ký”, hệ thống sẽ kiểm tra Email trong cơ sở dữ liệu, hiển thị thông báo và kết thúc hoạt động.



Hình 2.8. Quy trình đăng ký tài khoản

2.4.2. Quy trình đăng nhập

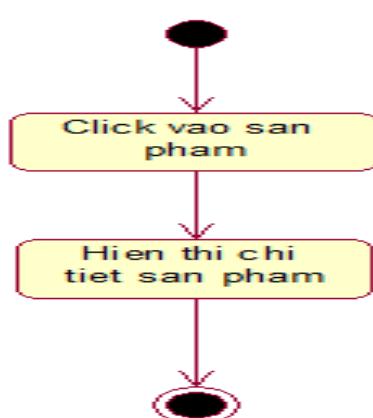
Hoạt động đăng nhập diễn ra khi người dùng đã có tài khoản và mở ứng dụng, người dùng nhập đúng Email và Password, sau đó nhấn nút “Đăng nhập”, hệ thống sẽ kiểm tra, nếu Email và Password đúng thì sẽ chuyển tới màn hình chính và kết thúc hoạt động, nếu sai hoặc bỏ trống thì sẽ thông báo và người dùng thực hiện nhập lại.



Hình 2.9. Quy trình đăng nhập

2.4.3. Quy trình xem chi tiết sản phẩm

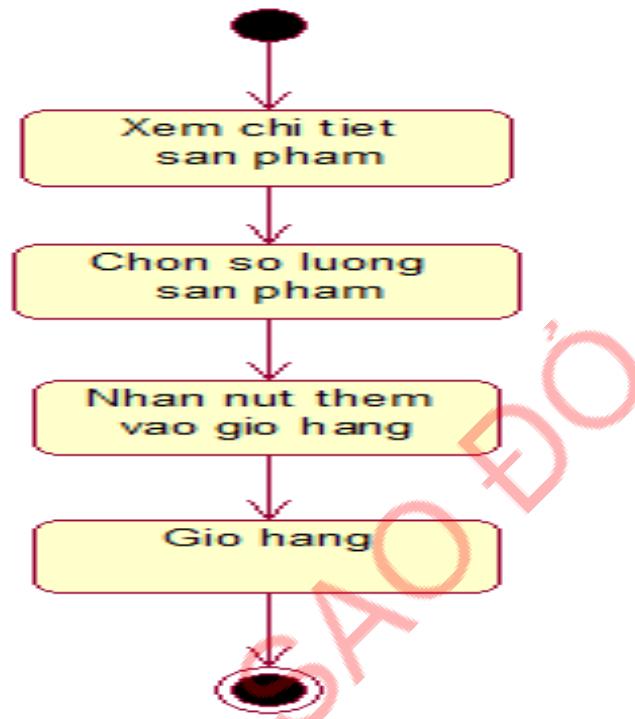
Khi người dùng click vào sản phẩm, thì màn hình sẽ hiển thị chi tiết sản phẩm đó và kết thúc hoạt động.



Hình 2.10. Quy trình xem chi tiết sản phẩm

2.4.4. Quy trình thêm sản phẩm vào giỏ hàng

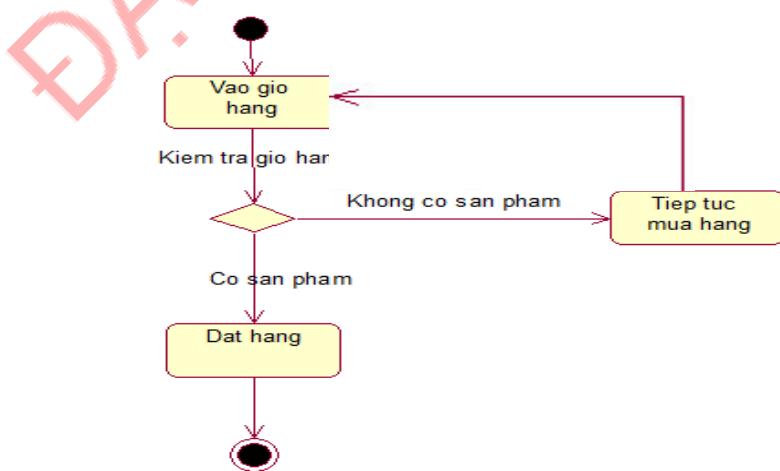
Hoạt động thêm sản phẩm vào giỏ hàng diễn ra khi người dùng xem chi tiết sản phẩm, chọn số lượng sản phẩm và nhấn vào nút “Thêm vào giỏ hàng”, sản phẩm vừa được thêm sẽ chuyển tới giỏ hàng và kết thúc hoạt động.



Hình 2.11. Quy trình thêm sản phẩm vào giỏ hàng

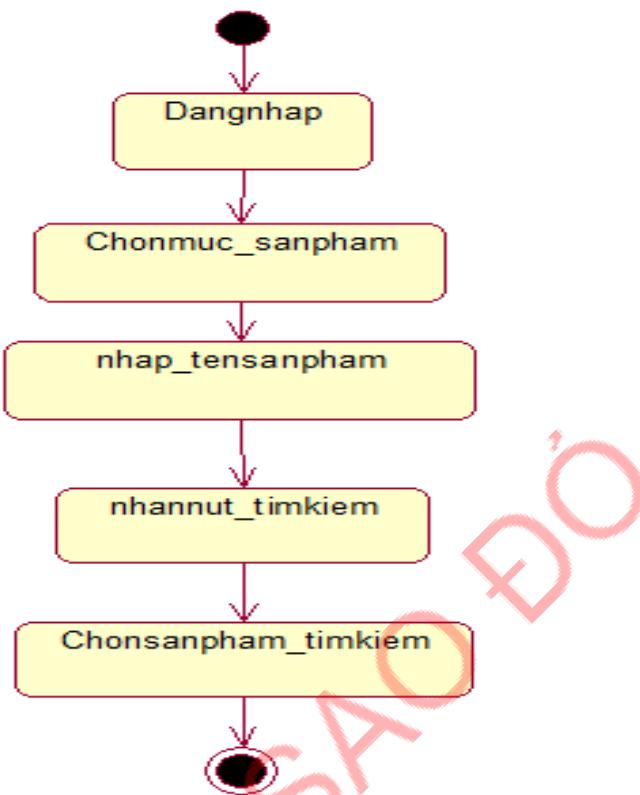
2.4.5. Quy trình đặt mua sản phẩm

Hoạt động đặt hàng diễn ra khi người dùng vào giỏ hàng và muốn đặt hàng, nếu giỏ hàng trống thì phải tiếp tục mua hàng để có sản phẩm trong giỏ hàng, nếu có sản phẩm trong giỏ hàng thì bấm nút đặt hàng, kết thúc hoạt động.



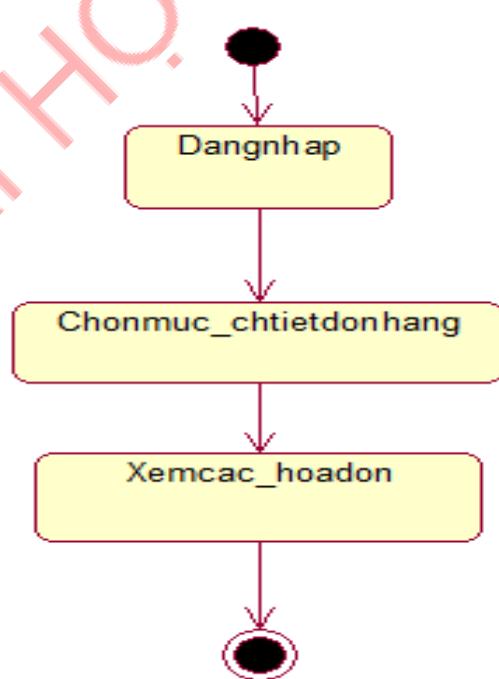
Hình 2.12. Quy trình đặt mua hàng

2.4.6. Quy trình tìm kiếm sản phẩm



Hình 2.13. Quy trình tìm kiếm sản phẩm

2.4.7. Quy trình xác nhận đơn hàng



Hình 2.14. Quy trình xác nhận đơn hàng

2.5. Kết luận chương 2

Chương 2 của đề tài tập trung vào việc phân tích cấu trúc tổng quan của dự án, bao gồm cả các khía cạnh về phần cứng, phần mềm, đối tượng, và chức năng. Những thông tin này cung cấp một cái nhìn toàn diện về kết cấu và yêu cầu cơ bản của ứng dụng, giúp xác định phạm vi và hướng phát triển tiếp theo. Đối với phần cứng, xác định yêu cầu cơ bản như máy tính, CPU, RAM và kết nối Internet. Những yêu cầu này nhấn mạnh sự cần thiết của một môi trường phát triển ổn định và đủ mạnh mẽ để hỗ trợ quá trình phát triển và kiểm thử. Đối với phần mềm, sử dụng Flutter, điều này không chỉ giúp đơn giản hóa quá trình phát triển đa nền tảng mà còn đảm bảo khả năng mở rộng và bảo trì; kết hợp với việc sử dụng MongoDB làm cơ sở dữ liệu cho ứng đem lại sự linh hoạt và khả năng mở rộng tốt.

ĐẠI HỌC SAO ĐỎ

CHƯƠNG 3. XÂY DỰNG ỦNG DỤNG BÁN HÀNG TRÊN ĐIỆN THOẠI DI ĐỘNG

3.1. Thiết lập API

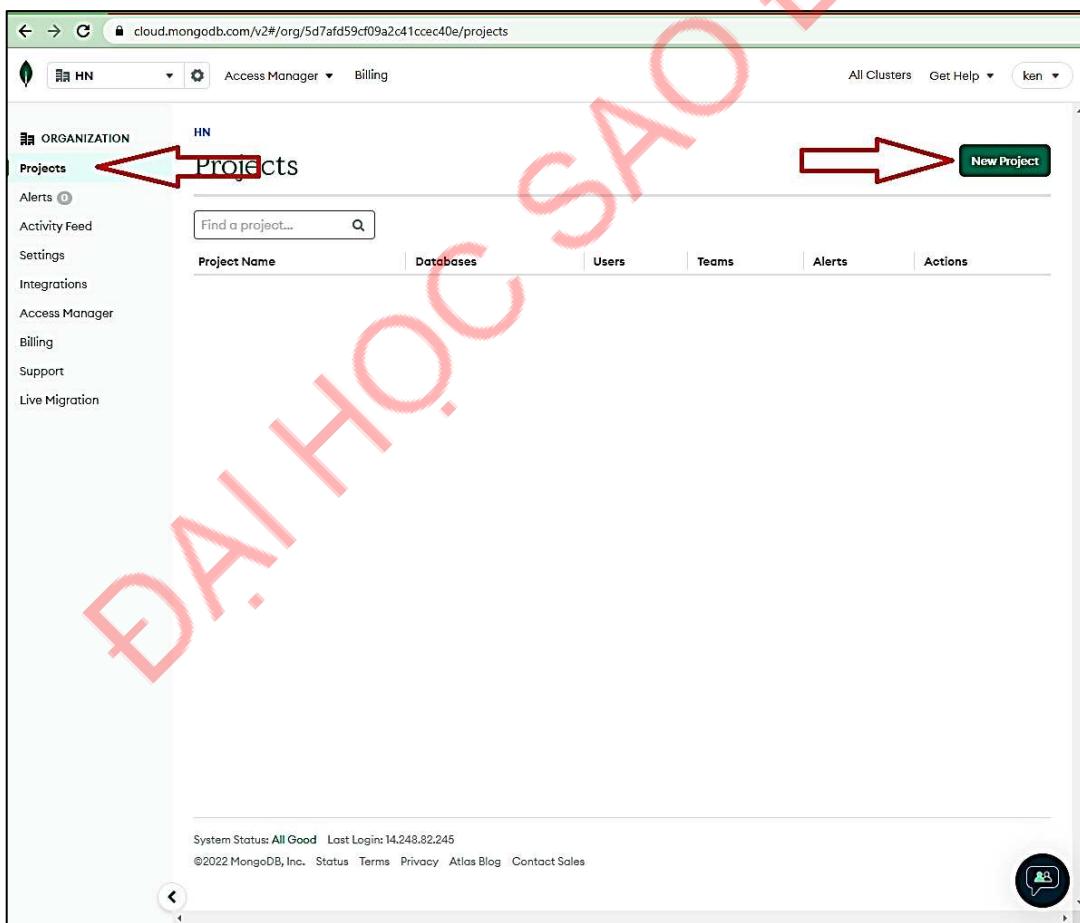
3.1.1. Thiết lập cơ sở dữ liệu MongoDB

MongoDB Atlas, một dịch vụ quản lý cơ sở dữ liệu MongoDB dựa trên đám mây sử dụng NoSQL. Nó giúp đơn giản hóa việc triển khai, quản lý và mở rộng cơ sở dữ liệu MongoDB mà không đòi hỏi người quản trị cơ sở dữ liệu phải quản lý các tác vụ hạ tầng hệ thống phức tạp.

Bước 1: Đăng ký tài khoản

Truy cập trang chủ: <https://www.mongodb.com/> để đăng ký sau đó đăng nhập tài khoản người dùng.

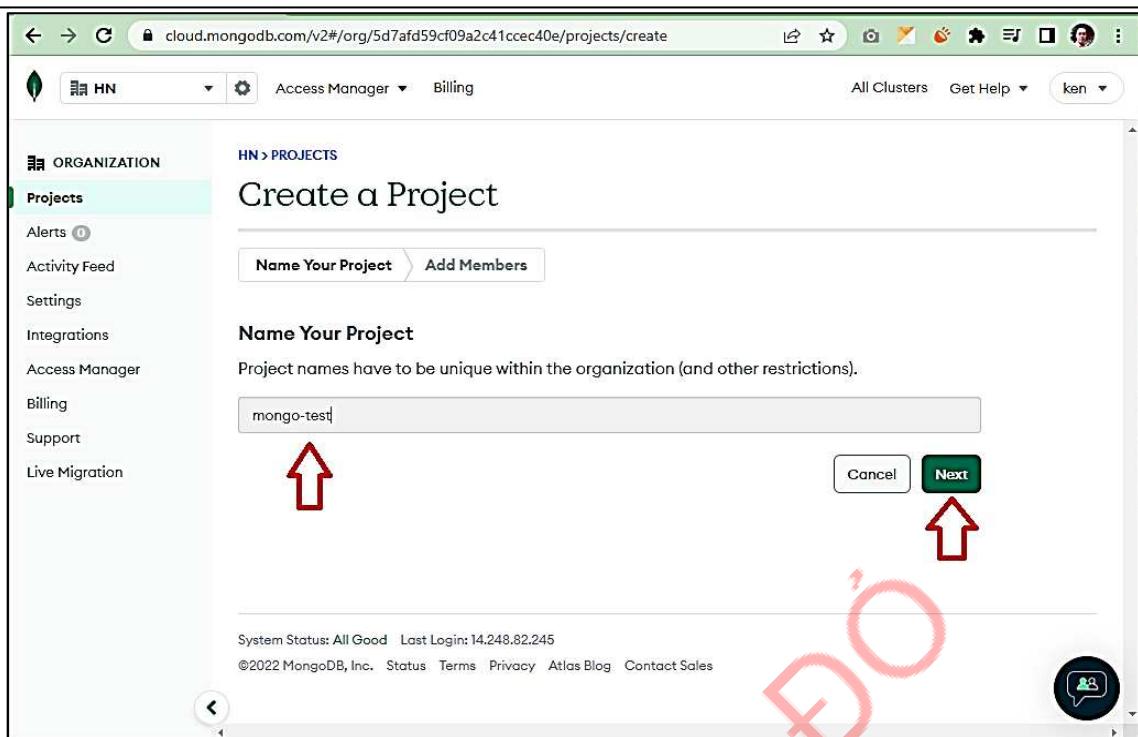
Bước 2: Tạo tên Project mới



Hình 3.1. Tạo tên project mới

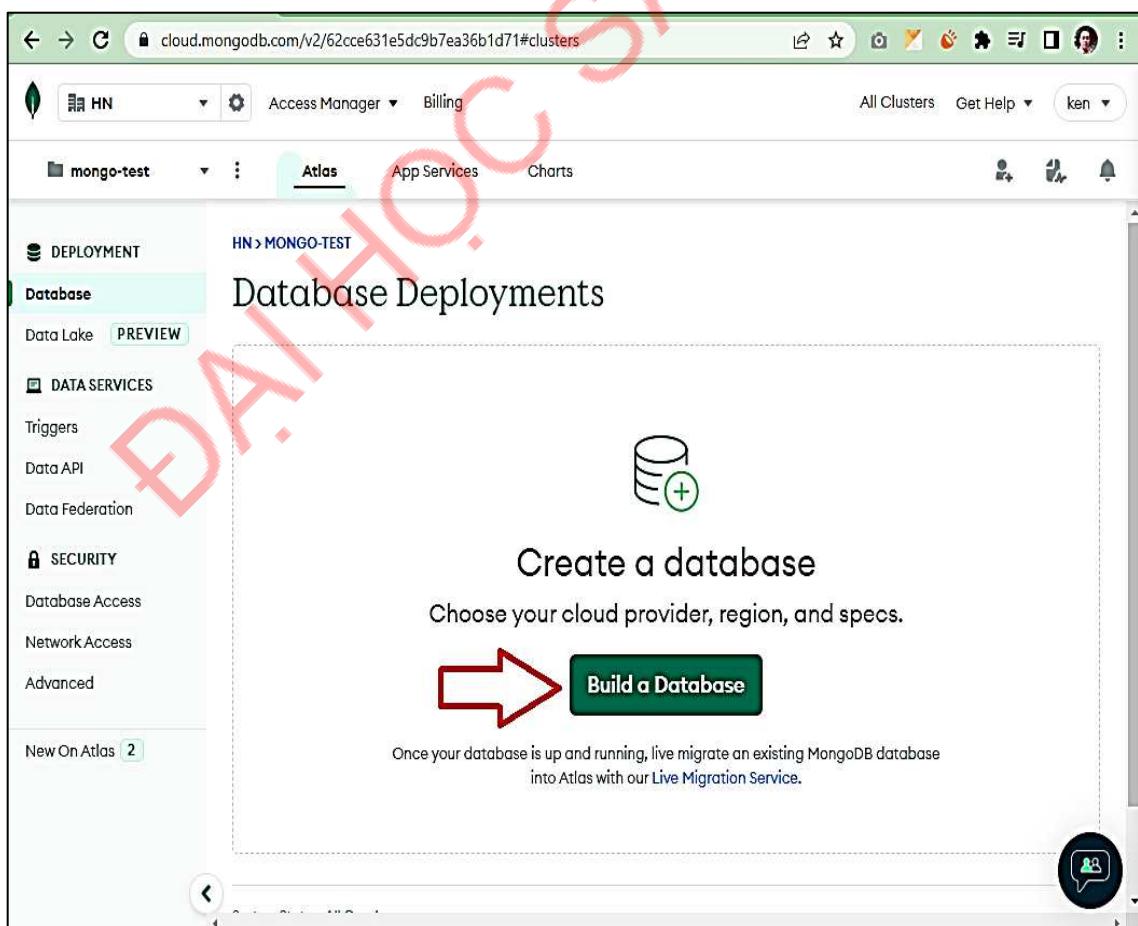
Bước 3: Đặt tên project

Đặt tên cho dự án chính là tên của cơ sở dữ liệu lưu trữ cho ứng dụng sau này.



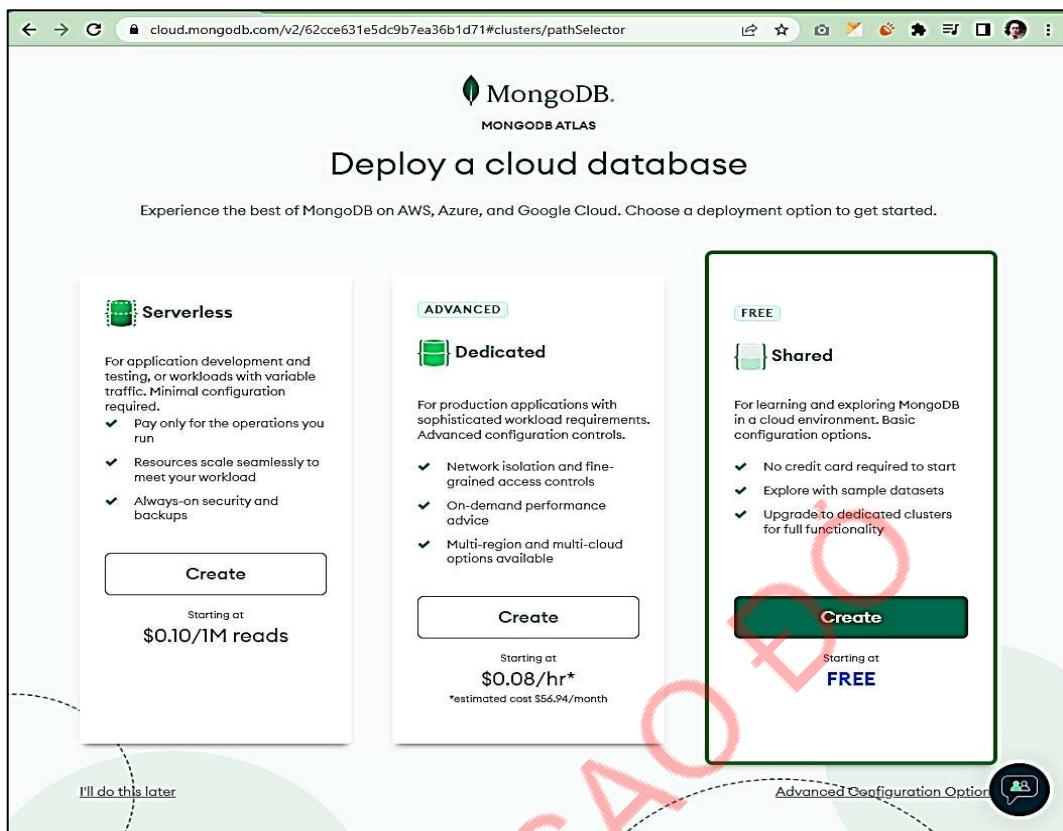
Hình 3.2. Đặt tên project

Bước 4: Tạo cơ sở dữ liệu



Hình 3.3. Tạo cơ sở dữ liệu

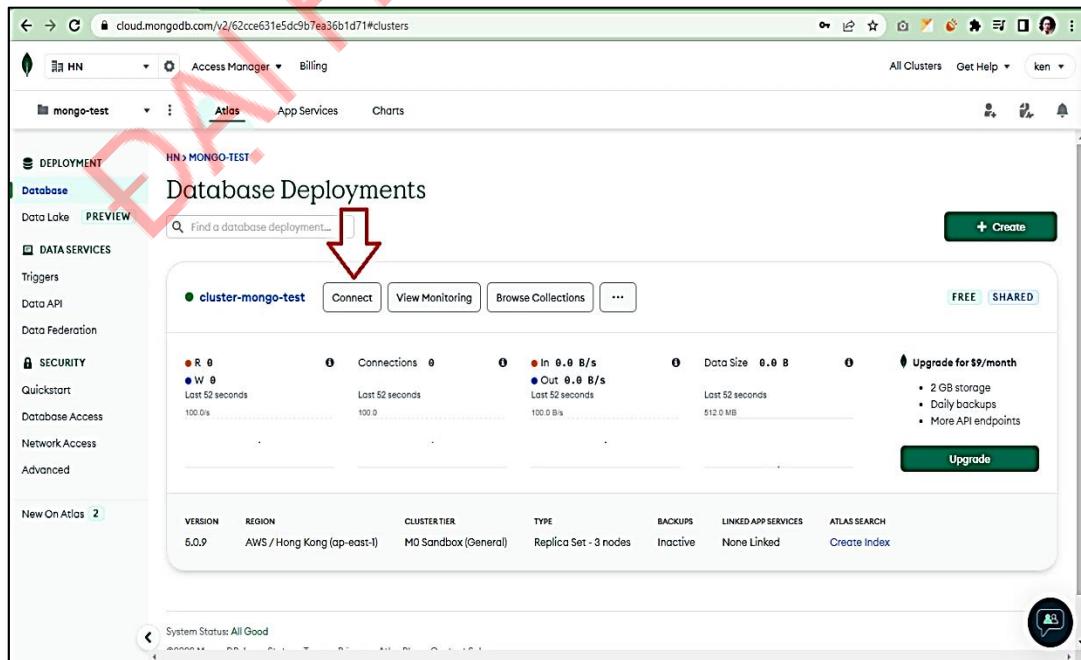
Bước 5: Chọn lưu trữ đám mây cho cơ sở dữ liệu



Hình 3.4. Chọn lưu trữ đám mây cho cơ sở dữ liệu

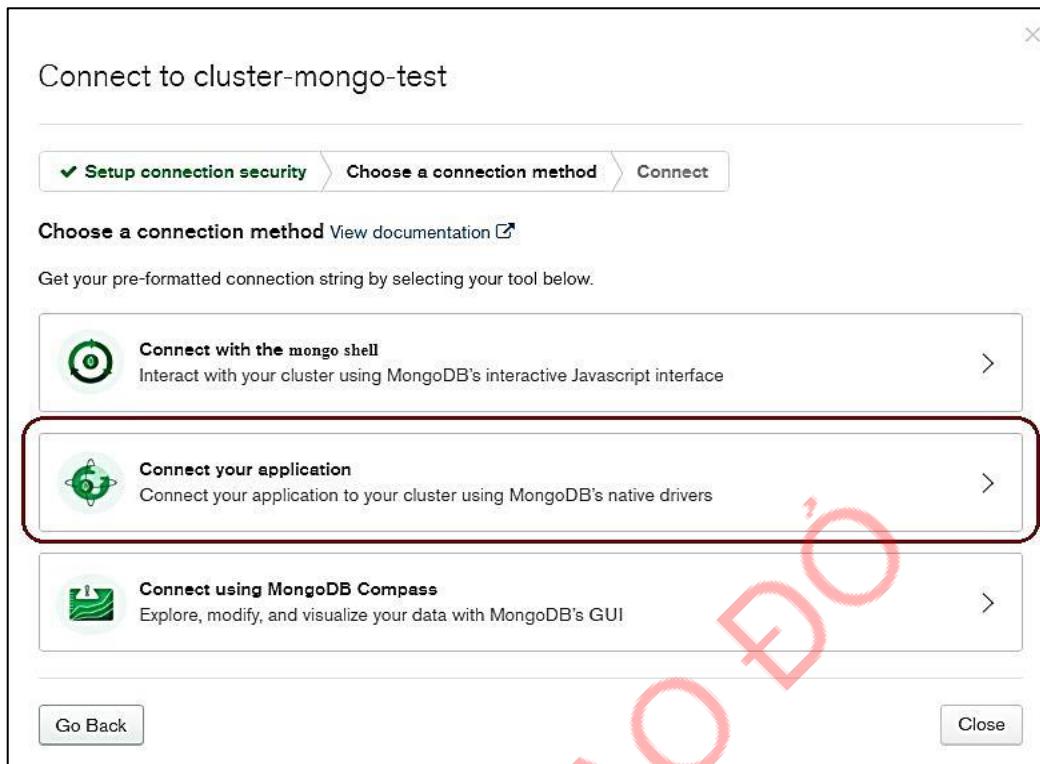
Bước 6: Chọn kiểu kết nối

Chọn kiểu kết nối dữ liệu là phương thức kết nối giữa thiết bị app với cơ sở dữ liệu đám mây.



Hình 3.5. Chọn kiểu kết nối

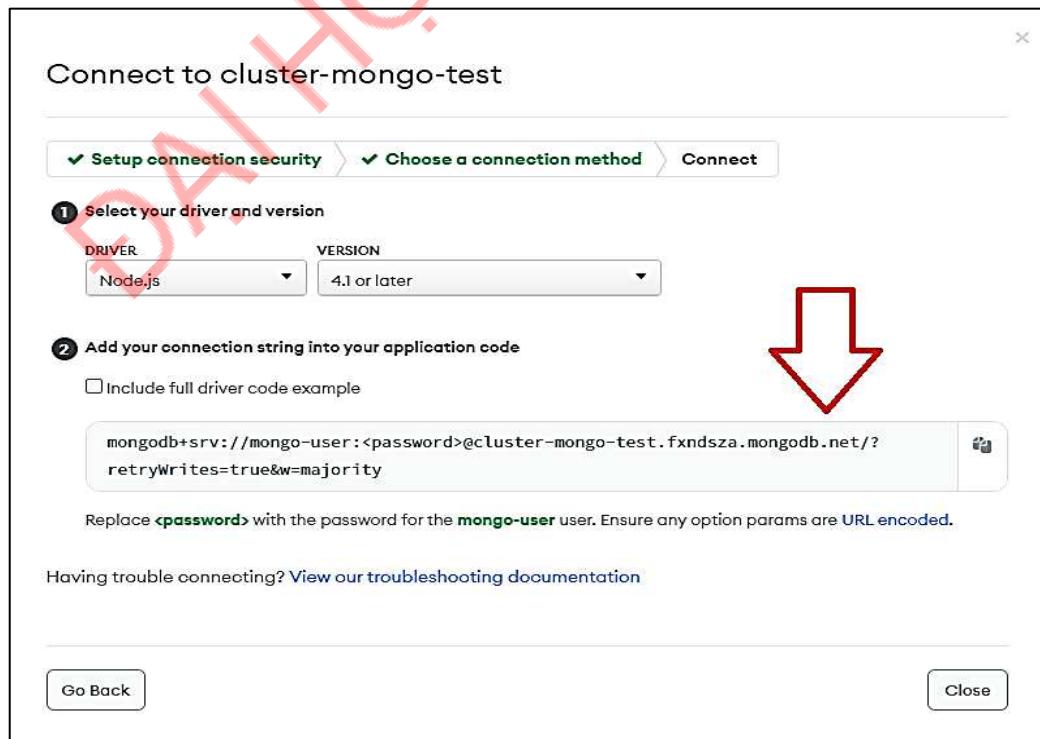
Bước 7: Chọn kết nối tới application



Hình 3.6. Chọn kết nối tới ứng dụng

Bước 8: Tạo mã kết nối

Chọn driver là ngôn ngữ sử dụng để kết nối tới mongodb and version chính là phiên bản của nodejs và copy đoạn mã kết nối được tự động sinh ra.



Hình 3.7. Tạo mã kết nối

3.1.2. Thiết lập tạo API

Trước tiên để kết nối tới cơ sở dữ liệu và tạo API cần cài đặt Node.js: Truy cập trang chính thức của Node.js tại <https://nodejs.org/> để tải và cài đặt Node.js.

Cài đặt MongoDB: Truy cập trang chính thức của MongoDB để tải và cài đặt MongoDB từ <https://www.mongodb.com/try/download/community>.

Bước 1: Khởi tạo dự án NodeJS

Khởi tạo một dự án Node.js mới và tạo một tệp package.json. Tại thư mục của dự án sử dụng command sau đó sử dụng lệnh:

```
npm init -y
```

Bước 2: Thiết lập các gói (thư viện) cần thiết

Express: Là Framework web tối giản cho Node.js nhằm đơn giản hóa quá trình tạo API mạnh mẽ và ứng dụng web.

Mongoose: Công cụ mô hình đối tượng MongoDB được thiết kế để làm việc trong môi trường không đồng bộ. Mongoose cung cấp một giải pháp dựa trên schema để mô hình dữ liệu ứng dụng.

HTTP: Là thành phần trong các module core của Node.js và cung cấp chức năng để tạo máy chủ và người dùng HTTP.

jsonwebtoken: Thư viện để tạo và xác minh các JSON Web Tokens (JWT). JWT thường được sử dụng cho việc xác thực và trao đổi thông tin giữa các bên.

bryptjs: Thư viện để mã hóa mật khẩu. Thường được sử dụng để lưu trữ mật khẩu người dùng bằng cách mã hóa trước khi lưu trữ trong cơ sở dữ liệu.

```
npm install express mongoose  
npm install http  
npm install jsonwebtoken  
npm install bcryptjs
```

Bước 3: Kết nối tới cơ sở dữ liệu

Sử dụng kết nối đã tạo trong MongoDB thực hiện kết nối với như sau:

```
const mongo = require('mongoose');  
  
const db = mongo.connect("mongodb+srv://<username>:<password>@cluster0.oek8j85.mongodb.net/?retryWrites=true&w=majority");  
  
db.on("error", console.error.bind(console, "MongoDB connection error:"));  
  
db.once("open", () => {  
    console.log("Connected successfully to server");  
});
```

```

    console.log("Kết nối thành công");
}).catch((e) => {
    console.log(e);
});

```

Bước 4: Thiết lập các đường dẫn truy cập API (Routes và Middleware)

Các đường dẫn điều hướng truy cập dữ liệu khi gọi lên API.

```

const authRouter = require('./routes/auth.js');
const adminRoute = require('./routes/admin.js');
const productRouter = require('./routes/product.js');
const userRouter = require('./routes/user.js');

app.use(express_import.json());
app.use(authRouter);
app.use(adminRoute);
app.use(productRouter);
app.use(userRouter);

```

Bước 5: Tạo kết nối trên Localhost

Ứng dụng sẽ lắng nghe dữ liệu trên cổng kết nối 3000 và xuất thông báo nếu kết nối thành công.

```

app.listen('3000', "0.0.0.0", () => {
    console.log('Đã kết nối tại cổng ' + '3000');
});

```

3.2. Xây dựng giao diện ứng dụng

my_flutter_app/

|-- lib/: Chứa mã nguồn của ứng dụng Flutter.

 | |-- **main.dart**: File chính của ứng dụng

 | |-- **screens/**: Chứa các widget tương ứng với các màn hình khác nhau của ứng dụng

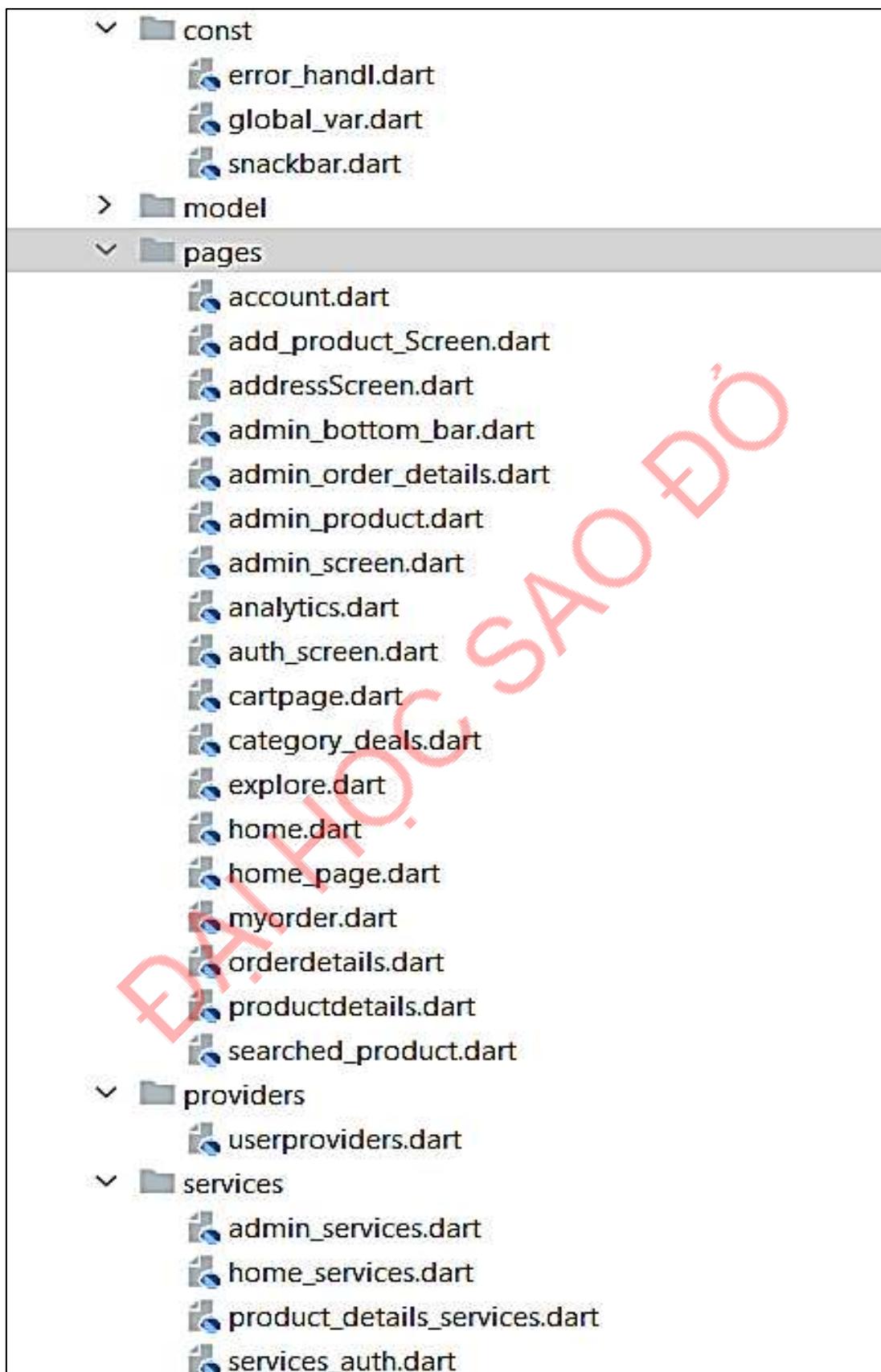
 | | |-- **home.dart**

 | | |-- **account.dart**

 | | |-- **myoder.dart**

```
| | |-- productdetail.dart
| | |-- add_product_screen.dart
| | |-- addressScreen.dart
| | |-- admin_order_details.dart
| | |-- ...
| |-- widgets/: Chứa các widget nhỏ và tái sử dụng được sử dụng trong các màn hình
| | |-- cart_product.dart
| | |-- cart_suntotal.dart
| | |-- ...
| |-- models/: Thư mục chứa các lớp mô hình để đại diện cho dữ liệu
| | |-- product.dart
| | |-- user.dart
| | |-- order.dart
| | |-- sales.dart
| | |-- rating.dart
| |-- services/: Chứa các dịch vụ, chẳng hạn như giao tiếp với API, xác thực, và quản lý cơ sở dữ liệu
| | |-- admin_services.dart
| | |-- home_services.dart
| | |-- product_detailse_services.dart
| | |-- services_auth.dart
| |-- images/
| | |-- books.jpg
| | |-- laptop.jpg
| | |-- fashion.jpg
| | |-- ...
|-- test/
|-- android/: Chứa mã nguồn native cho Android
|-- ios/: Chứa mã nguồn native cho iOS.
```

-- pubspec.yaml: Chứa các thông tin về dependencies (thư viện, gói) và các cấu hình khác của dự án.



Hình 3.8. Cấu trúc dự án Flutter

Cấu trúc ứng dụng được liệt kê trong bảng sau:

Bảng 3.1. Bảng cấu trúc ứng dụng

Nội dung	Chi tiết
Service	Admin_service
	Home_service
	Product_detail_service
	Service_auth
Model	Oder
	User
	Rating
	Orders
	Sales
Page	Product
	Account
	Add_product
	AddressScreen
	admin_product
	Admin_screen
	analysict
	Cartpage
	Myorder
	Home

Nội dung	Chi tiết
	bottomnav
	Button
	Caraoselimage
	Cart_product
	Cart_subtotal
	Charts
	Field
	Order
	Start
widgets	Search_product

3.3. Hoàn thiện logic

3.3.1. Chức năng đăng nhập/đăng ký

Chức năng chính trong phần này gồm: Hàm đăng ký (signup) và đăng nhập (signin) và xác thực người dùng (auth_service). Auth được sử dụng để xác định hành động đăng ký hay đăng nhập. Khi tài khoản đã được đăng nhập trên ứng dụng, sẽ tạo ra mã auth để xác thực người dùng đang sử dụng thiết bị hiện tại, do đó người dùng sẽ không cần đăng nhập lại lần tiếp theo khi vào ứng dụng. Hàm đăng nhập và đăng ký tài khoản sẽ lấy thông tin người dùng nhập và gửi về server thông qua API. Với câu lệnh `http.Response res = await http.post(Uri.parse('$uri/api/signup'))` trong đó uri là đường dẫn của hosting truy cập và gói dữ liệu gửi đi dưới dạng Json. Máy chủ sẽ nhận gói dữ liệu Json và giải mã sau đó xử lý logic cho đăng nhập nếu thành công sẽ trả dữ liệu lại cho thiết bị người dùng.

```
Auth _auth = Auth.signup;
final _signUpFormkey = GlobalKey<FormState>();
final _signInFormkey = GlobalKey<FormState>();
final auth_service auth = auth_service();
final TextEditingController email = TextEditingController();
final TextEditingController username = TextEditingController();
final TextEditingController pass = TextEditingController();
```

```
void signupuser() {  
    auth.signupuser(  
        context: context,  
        email: email.text,  
        password: pass.text,  
        name: username.text);  
}  
void signinuser() {  
    auth.signinuser(  
        context: context, email: email.text, password: pass.text, name: "");  
}  
@override  
Widget build(BuildContext context) {  
    void dispose() {  
        super.dispose();  
        email.dispose();  
        username.dispose();  
        pass.dispose();  
    }  
// HÀM SEVICE_AUTH  
Void signupuser(  
    {  
        required BuildContext context,  
        required String email,  
        required String pasword,  
        required String name}) async {  
    try {  
        User user = User(  
            id: "",  
            email: email,  
            name: name,  
            password: password,  
            address: "",  
            type: "",  
            cart: []);  
    }  
}
```

```
http.Response res = await http  
    .post(Uri.parse('$uri/api/signup'), body: user.toJson(), headers: {  
    'Content-Type': 'application/json; charset=UTF-8',  
});  
  
print(res.statusCode);  
// ignore: use_build_context_synchronously  
  
httpsError(  
    response: res,  
    context: context,  
    onSucces: () {  
    snackbar(context, "Account Creation Success");  
},  
) catch (e) {  
    snackbar(context, e.toString());  
}  
}  
  
void signinuser(  
{  
    required BuildContext context,  
    required String email,  
    required String pasword,  
    required String name}) async {  
try {  
    http.Response res = await http.post(Uri.parse('$uri/api/signin'),  
        body: jsonEncode({'email': email, 'password': pasword}),  
        headers: {  
        'Content-Type': 'application/json; charset=UTF-8',  
    },);  
    print(res.statusCode);  
}
```

```

httpsError(
    response: res,
    context: context,
    onSucces: () async {
        SharedPreferences prefs = await SharedPreferences.getInstance();
        Provider.of<UserProvider>(context, listen: false).setUser(res.body);
        prefs.setString("x-auth-token", jsonDecode(res.body)['token']);
        prefs.setString("x-auth-mail", jsonDecode(res.body)['email']);
        prefs.setString("x-auth-pass", jsonDecode(res.body)['password']);
        Navigator.pushNamedAndRemoveUntil(
            context, home.routeName, (route) => false);
    });
} catch (e) {
    snackbar(context, e.toString());
}
}

```

3.3.2. Chức năng hiển thị sản phẩm tại trang chủ

Khi vào các phần tại trang chủ chương trình gửi yêu cầu HTTP GET đến API để lấy danh sách sản phẩm theo danh mục Future<List<Product>> fetchCategoryProducts().

Sử dụng http.get: Phương thức get của gói HTTP được sử dụng để gửi yêu cầu GET đến API với đường dẫn được xây dựng dựa trên uri và category.

Sau khi gửi yêu cầu lên máy chủ sẽ trả về dữ liệu phản hồi dạng Json. Phân tích dữ liệu JSON nhận được từ API thành danh sách sản phẩm. Phương thức httpsError: Hàm httpsError được gọi để kiểm tra và xử lý phản hồi từ API. Nếu thành công (onSucces), dữ liệu JSON nhận được từ API được phân tích và chuyển đổi thành danh sách các đối tượng Product.

Phương thức snackbar: Nếu có lỗi trong quá trình gửi yêu cầu hoặc xử lý dữ liệu, hàm snackbar được gọi để hiển thị thông báo lỗi.

```

{required BuildContext context, required String category}) async {
List<Product> productList = [];

```

```

try {
    http.Response res = await http.get(
        Uri.parse(
            '$uri/api/cat-product?category=$category'), // by $category the query will be
        done from database
        headers: <String, String>{
            'Content-Type': 'application/json;charset=UTF-8'
        },
    );
    httpsError(
        response: res,
        context: context,
        onSucess: () {
            List temp = jsonDecode(res.body);
            temp.forEach((element) {
                productList.add(Product.fromJson(jsonEncode(element)));
            });
        });
    } catch (e) {
        snackbar(context, e.toString());
    }
    return productList;
}

```

3.3.3. Chức năng tìm kiếm sản phẩm

Khi người dùng có yêu cầu truy vấn sản phẩm chương trình kiểm tra dữ liệu người dùng nhập sẽ gọi tới API await http.get(Uri.parse('\$uri/admin/get-product')). Phản hồi nhận từ phía người dùng danh sách các sản phẩm. Sau đó chương trình xử lý file dữ liệu và hiển thị dữ liệu lên giao diện người dùng.

```

Future<List<Product>> fetchSearchProducts(
    {required BuildContext context, required String query}) async {

```

```
List<Product> productList = [];  
try {  
    http.Response res;  
    if (query == "" || query == " ") {  
        res = await http.get(  
            Uri.parse(  
                '$uri/admin/get-product'), headers: <String, String>{  
                    'Content-Type': 'application/json; charset=UTF-8'  
                },  
            );  
    } else {  
        res = await http.get(  
            Uri.parse(  
                '$uri/api/search-product/$query'), headers: <String, String>{  
                    'Content-Type': 'application/json; charset=UTF-8'  
                },  
            );  
    }  
    httpsError(  
        response: res,  
        context: context,  
        onSucces: () {  
            List temp = jsonDecode(res.body);  
            temp.forEach((element) {  
                productList.add(Product.fromJson(jsonEncode(element)));  
            });  
        });  
    }  
}
```

```
catch (e)  
{  
    Snackbar(context, e.toString());  
}  
  
return productList;  
}
```

3.3.4. Chức năng thêm giỏ hàng

Chương trình gửi yêu cầu HTTP POST đến API để thêm sản phẩm vào giỏ hàng các trường dữ liệu như idproduct, iduser. Sử dụng http.post để gửi yêu cầu POST đến API với thông tin dạng Json và thêm vào cơ sở dữ liệu. Sau đó hiển thị thông báo "Added to cart" nếu mọi thứ diễn ra thành công.

Provider.of<UserProvider>: Sử dụng Provider.of để lấy instance của UserProvider. listen: false được sử dụng để không lắng nghe thay đổi trong UserProvider ngay sau khi nó được tạo ra.

```
void addCart({  
    required BuildContext context,  
    required Product product,  
}) async {  
    try {  
        final userProvider = Provider.of<UserProvider>(context, listen: false);  
        http.Response res = await http.post(  
            Uri.parse(  
                '$uri/user/add-to-cart'), headers: <String, String>{  
                'Content-Type': 'application/json; charset=UTF-8'  
            },  
            body:  
                jsonEncode({'id': product.id, 'user_id': userProvider.user.id}));  
        httpsError(  
            response: res,  
            context: context,
```

```
onSuccess: () {  
    var temp =  
        userProvider.user.copyWith(cart: jsonDecode(res.body)['cart']);  
    userProvider.setUserFrommodel(temp);  
    snackbar(context, "Added to cart");  
});  
} catch (e) {  
    snackbar(context, e.toString());  
}  
}
```

3.3.5. Chức năng thêm địa chỉ

```
void saveAddress({  
    required BuildContext context,  
    required String address,  
}) async {  
    try {  
        final userProvider = Provider.of<UserProvider>(context, listen: false);  
        http.Response res = await http.post(  
            Uri.parse('$uri/user/save-user-address'),  
            headers: <String, String>{  
                'Content-Type': 'application/json; charset=UTF-8'  
            },  
            body: jsonEncode(  
                {'address': address, 'user_id': userProvider.user.id});  
        httpsError(  
            response: res,  
            context: context,  
            onSuccess: () {  
                //...  
            },  
            onFailure: (error) {  
                //...  
            },  
            onProgress: (progress) {  
                //...  
            },  
            onError: (error) {  
                //...  
            },  
        );  
    } catch (e) {  
        snackbar(context, e.toString());  
    }  
}
```

```

var temp = userProvider.user

    .copyWith(address: jsonDecode(res.body)['address']);

    userProvider.setUserFrommodel(temp);

    snackbar(context, "Address Updated");

});

} catch (e) {

    snackbar(context, e.toString());

}

}

```

3.4. Kết quả thực nghiệm và đánh giá

3.4.1. Kết quả thực nghiệm

Kết quả cơ sở dữ liệu như sau:

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with various service icons and a 'Deployment' section highlighted. The main area shows a database named 'test' with a collection named 'myorders'. Below the collection name, there are several document names: carts, categories, orders, product_infos, products, and users. In the center, there's a search bar with 'test.myorders' and a 'Find' button. Below the search bar, it says 'QUERY RESULTS: 1-6 OF 6'. Two documents are listed:

```

_id: ObjectId('6555f1ca9e030bc9b30e4a')
  products: Array (1)
    totalPrice: 49
    address: "a,v,d - c"
    userId: "6555f1899e030bc9b30e2c"
    orderedAt: 1700131274276
    status: 4
    __v: 0

_id: ObjectId('6564a2def2a44280a92c1944')
  products: Array (1)
    totalPrice: 799

```

Hình 3.9. Dữ liệu của ứng dụng

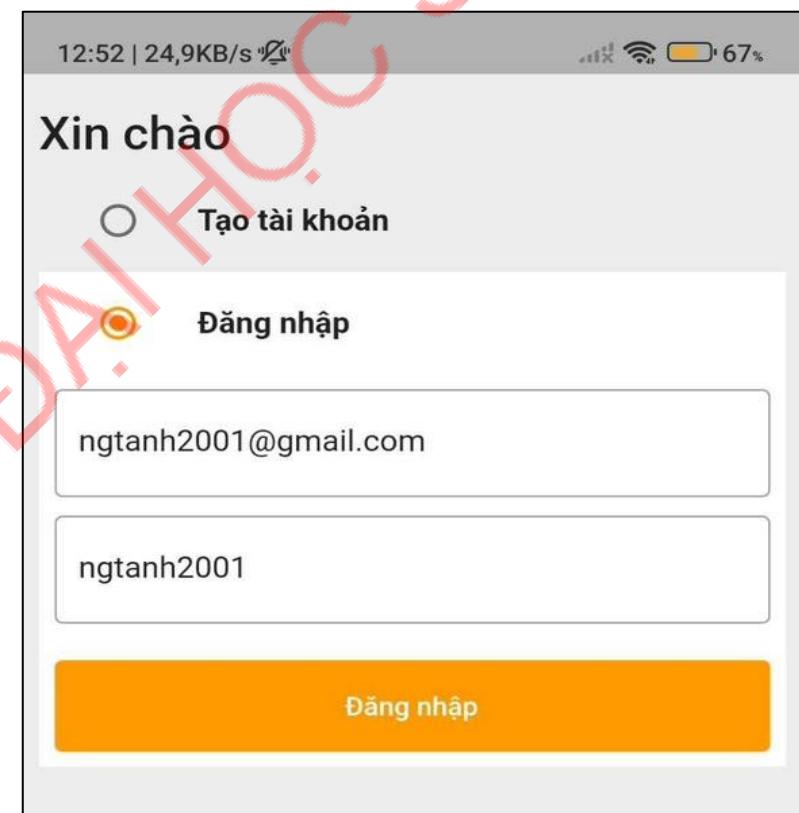
Để sử dụng ứng dụng người dùng phải tải và cài đặt ứng dụng trên điện thoại thông minh hệ điều hành Android, iOS. Sau đó phải đăng ký tài khoản với địa chỉ thư điện tử để vào ứng dụng. Ứng dụng có 2 tùy chọn: Đăng ký và đăng nhập dành cho người dùng chưa có tài khoản hoặc đã có tài khoản.

Đối với đăng ký, người dùng nhập các thông tin là địa chỉ email, tên người dùng, mật khẩu. Khi đã có tài khoản đăng ký hệ thống, người dùng được thực hiện chức năng đăng nhập.

Thông tin đăng nhập gồm địa chỉ email, mật khẩu đã đăng ký trên hệ thống và đăng nhập để thực hiện chức năng nhận xét sản phẩm, đặt mua sản phẩm.



Hình 3.10. Giao diện đăng ký ứng dụng



Hình 3.11. Giao diện đăng nhập ứng dụng

Tại giao diện thêm địa chỉ người dùng là khách mua hàng, các thông tin mà khách hàng nhập để cửa hàng căn cứ giao hàng. Các thông tin này là địa chỉ chính xác và có các tùy chọn: Lấy thông tin đăng ký mà khách hàng đã cung cấp cho hệ thống hoặc nhập địa chỉ mới. Sau khi nhập đủ các thông tin, khách hàng cần xác nhận để cập nhật thông tin cho giai đoạn giao hàng.



Hình 3.12. Thêm địa chỉ thành công

Giao diện chính hiển thị thông tin cho phép khách hàng tìm kiếm sản phẩm theo tên sản phẩm trong danh mục các hàng hóa tại cửa hàng. Gồm các loại danh mục: Sản phẩm điện thoại, sản phẩm thiết yếu, sản phẩm gia dụng, sản phẩm giáo dục và sản phẩm thời trang. Trong mỗi nhóm sản phẩm trên chứa danh sách các sản phẩm theo danh mục đã sắp xếp.

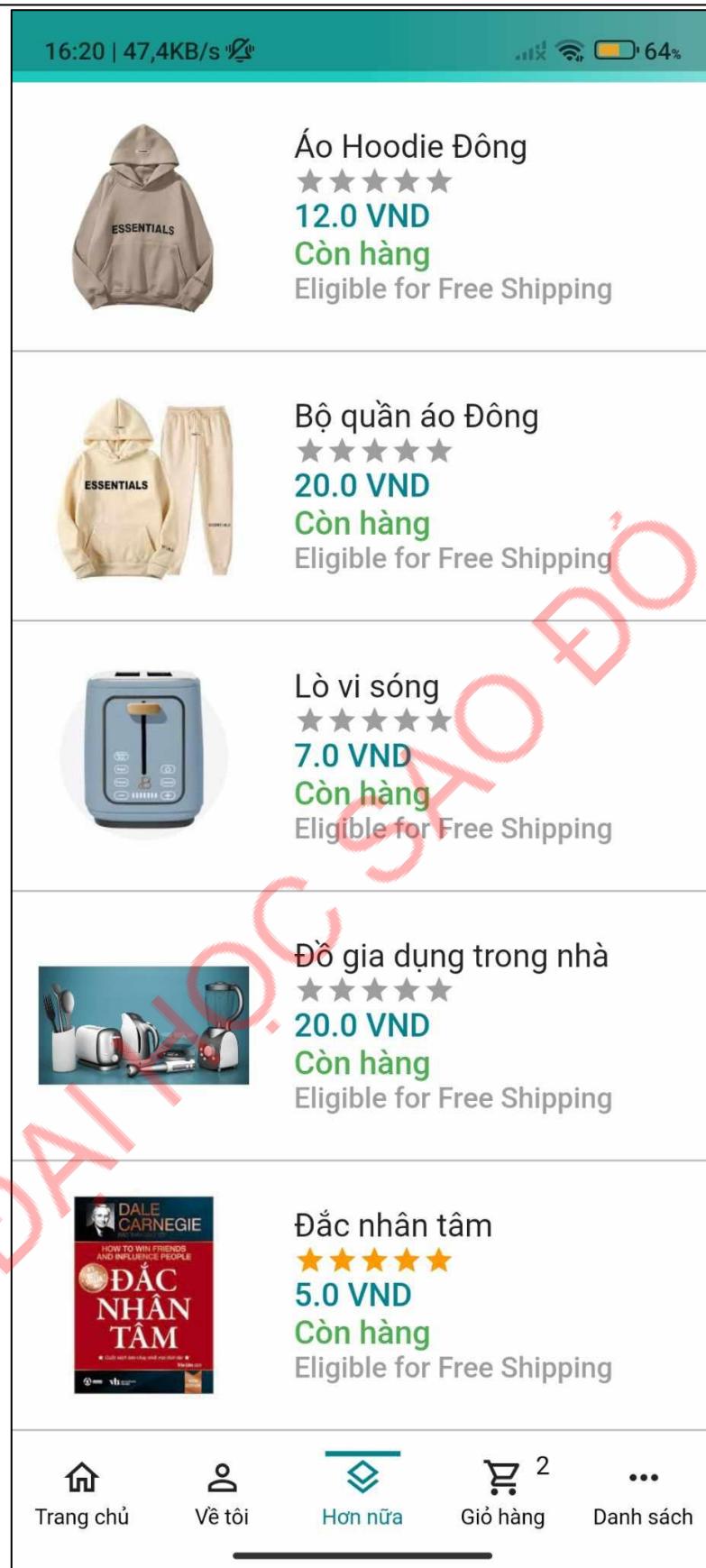
Tại giao diện chính còn hiển thị sản phẩm bán chạy trong ngày (được tính theo số lượng hàng đặt trong ngày nhiều nhất). Hiển thị slider các thông tin mới nhất như khuyến mãi, giảm giá và các quảng cáo khác.



Hình 3.13. Giao diện trang chủ ứng dụng

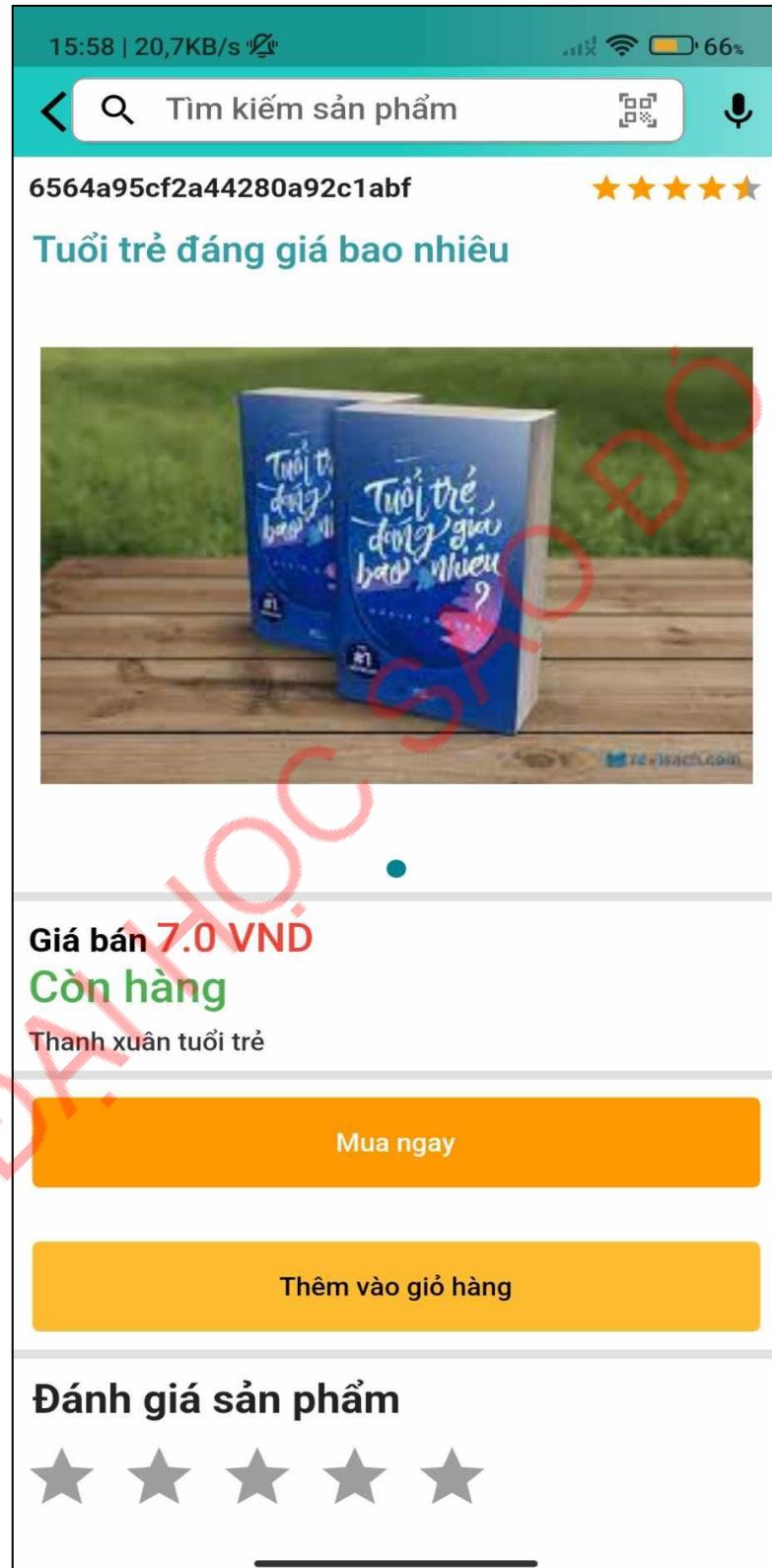
Phía dưới cùng của giao diện chính hiển thị các chức năng của người dùng, như giỏ hàng chứa thông tin về giỏ hàng là các sản phẩm đã đặt mua cùng trạng thái của quy trình đang thực hiện. Mục danh sách chứa các thông tin cho phép người dùng xem lịch sử mua hàng, đăng xuất ứng dụng, xem tiến trình xử lý đơn hàng.

Giao diện danh sách sản phẩm chứa các thông tin về tên sản phẩm, hình ảnh sản phẩm, giá bán sản phẩm, đánh giá của khách hàng về sản phẩm, tình trạng sản phẩm (còn hàng, hết hàng) để khách hàng nhận biết.



Hình 3.14. Danh sách sản phẩm

Giao diện chi tiết sản phẩm hiển thị hiển thị thông tin chi tiết người đánh giá sản phẩm (điều này là cần thiết để tạo sự tin tưởng cho khách hàng), hiển thị ảnh cụ thể sản phẩm, giá bán sản phẩm, tình trạng sản phẩm (còn hàng, hết hàng), mô tả sản phẩm, chọn chức năng mua hàng và thêm vào giỏ hàng.



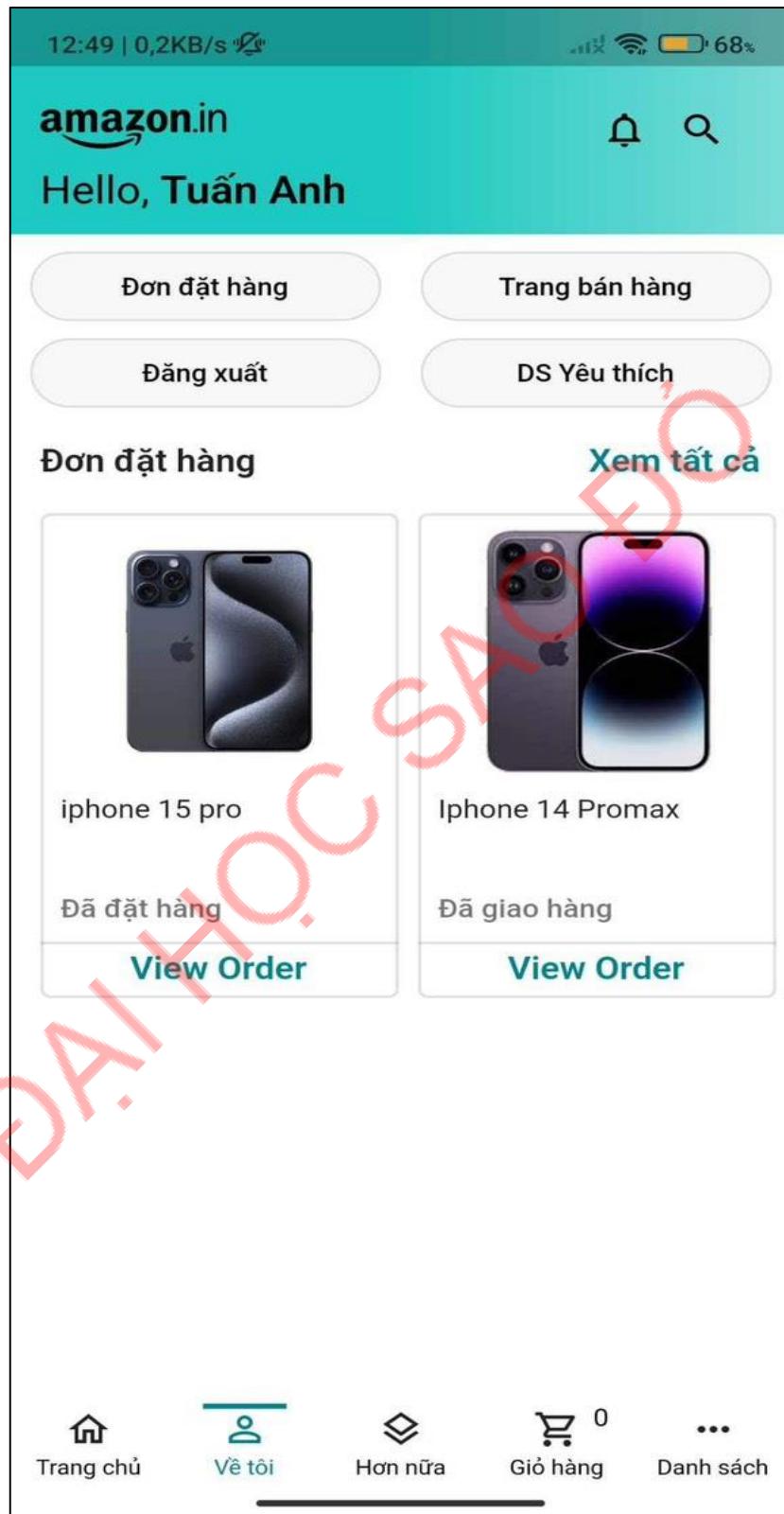
Hình 3.15. Chi tiết thông tin sản phẩm

Tại giao diện quản trị xử lý đơn hàng của khách hàng, thông tin gồm: Ngày giờ đặt hàng, mã đơn hàng, số tiền đã mua, tóm tắt đơn hàng, tiến trình xử lý của khách hàng và phản hồi của quản trị (đã đặt đơn, đã gửi đi, đang giao hàng, đã giao hàng).



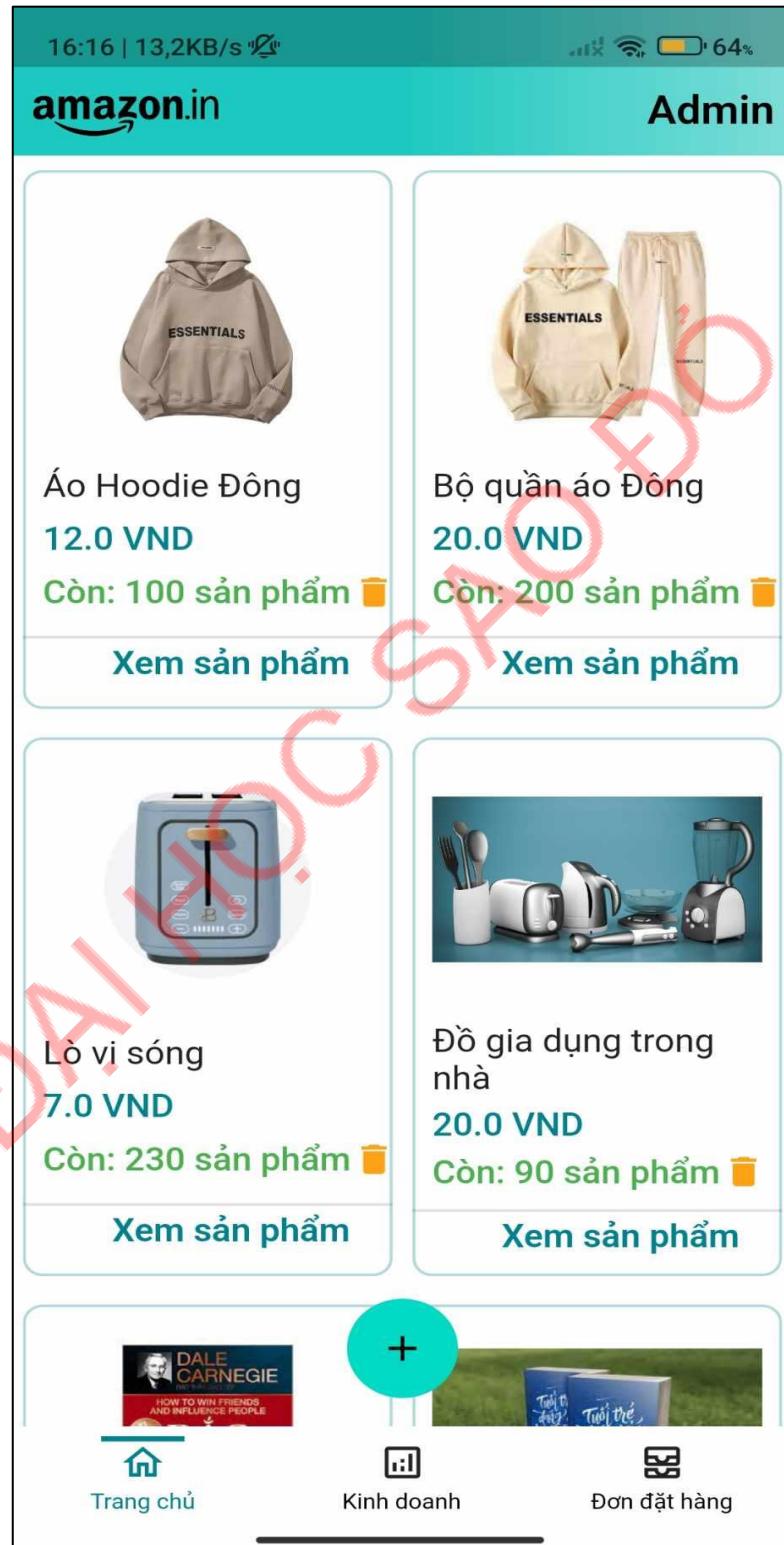
Hình 3.16. Quản lý quá trình đặt mua sản phẩm người bán hàng

Đối với khách hàng cũng có chức năng riêng, gồm xem đơn đặt hàng, xem danh sách sản phẩm yêu thích, xem trang bán hàng của hệ thống. Trong mỗi đơn hàng đã đặt, khách hàng xem được tình trạng của từng đơn hàng.



Hình 3.17. Lịch sử các sản phẩm đã mua

Giao diện quản lý sản phẩm của quản trị cho phép hiển thị thông tin tên sản phẩm, giá thành sản phẩm, số lượng sản phẩm còn đến thời điểm hiện tại để quản trị có hướng bổ sung, xử lý. Tại giao diện quản trị còn hiển thị thống kê tình hình kinh doanh mặt hàng nào để có phương hướng đề xuất, tình trạng các đơn hàng của khách để xác nhận và xử lý.



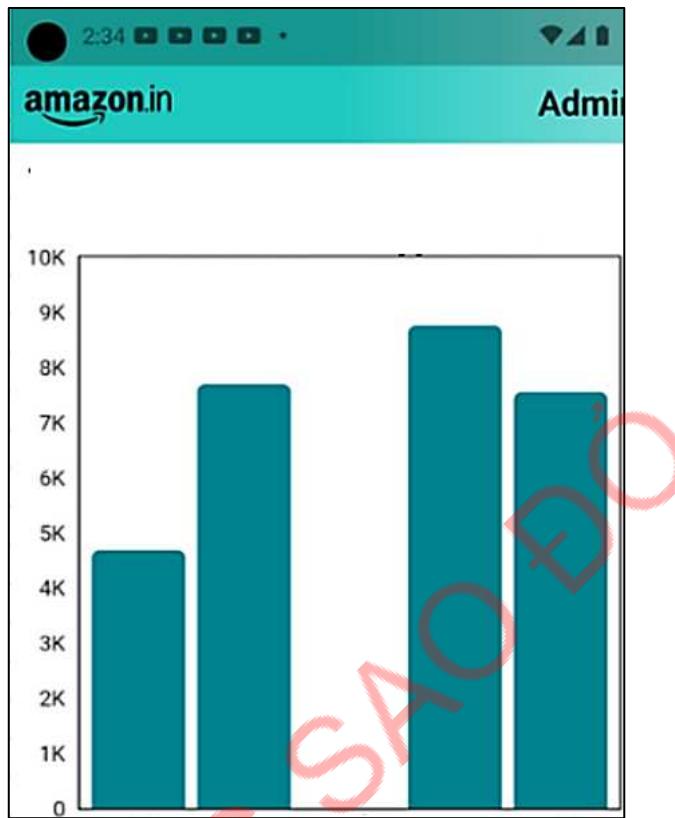
Hình 3.18. Danh sách sản phẩm đang bán

Phần xem đơn đặt hàng của quản trị, quản trị xem được tình trạng của từng đơn hàng, mỗi đơn hàng có thông tin về sản phẩm cần đặt, tổng tiền, tình trạng đã nhận hàng, đã gửi hàng,... và chi tiết về đơn hàng.



Hình 3.19. Danh sách sản phẩm đã được mua

Giao diện thống kê doanh thu của cửa hàng hiển thị dạng biểu đồ các mặt hàng đã bán. Biểu đồ trực quan mặt hàng nào bán được nhiều giúp quản trị có định hướng kinh doanh cho thời gian tiếp theo.



Hình 3.20. Biểu đồ phân tích bán hàng

3.4.2. Đánh giá ứng dụng

Ứng dụng bán hàng trên điện thoại di động được xây dựng bằng Framework đa nền tảng Flutter, sử dụng cơ sở dữ liệu phi quan hệ No SQL MongoDB và tích hợp API được xem xét, đánh giá dựa trên một số tiêu chí sau:

Bảng 3.2. Đánh giá về yêu cầu phi chức năng

Tiêu chí	Nội dung đánh giá
Trải nghiệm người dùng (UI/UX)	Linh hoạt trên nhiều thiết bị, đồng nhất giữa các nền tảng.
Hiệu suất	Hiệu suất tốt nhờ vào việc biên dịch thành mã máy, giảm độ trễ nhằm tối ưu hóa vấn đề hiệu suất.
Tích hợp API	Sử dụng API giúp kết nối ứng dụng với các nguồn dữ liệu và tính năng khác nhau. Flutter có thư viện mạnh mẽ cho việc xử lý HTTP requests.
Lưu trữ dữ liệu với mongodb	MongoDB cung cấp cơ sở dữ liệu linh hoạt, có khả năng mở rộng và có thể thích ứng với cấu trúc dữ liệu thay đổi.

Tiêu chí	Nội dung đánh giá
Độ bảo mật	Việc sử dụng HTTPS, mã hóa dữ liệu và cơ chế xác thực đúng cách có thể đảm bảo mức độ bảo mật cao cho ứng dụng.

Bảng 3.3. Đánh giá về chức năng ứng dụng

Chức năng cơ bản	Chi tiết đánh giá
Xem sản phẩm	Hiển thị danh sách sản phẩm với hình ảnh, giá cả và mô tả chi tiết.
	Hỗ trợ tìm kiếm và lọc sản phẩm theo danh mục, giá, đánh giá, và các tiêu chí khác.
Giỏ hàng	Cho phép người dùng thêm sản phẩm vào giỏ hàng và xem tổng số tiền.
	Hiển thị thông tin chi tiết về sản phẩm trong giỏ hàng
Thanh toán và đặt hàng	Chưa cung cấp nhiều phương thức thanh toán an toàn như thẻ tín dụng, ví điện tử, chuyển khoản.
	Xác nhận thông tin đặt hàng và gửi xác nhận đơn hàng đến email hoặc ứng dụng
Quản lý tài khoản	Đăng ký tài khoản cá nhân và đăng nhập an toàn
	Xem lịch sử đơn hàng và theo dõi tình trạng vận chuyển linh hoạt.
Thông báo và cập nhật	Gửi thông báo về ưu đãi đặc biệt, sự kiện mới, và cập nhật sản phẩm.
	Thông báo đơn hàng, thanh toán và giao hàng.
Đánh giá và nhận xét	Cho phép người dùng đánh giá về sản phẩm.
	Hiển thị đánh giá để người dùng khác có thể tham khảo trước khi mua.
Theo dõi vận chuyển	Cung cấp thông tin về tình trạng vận chuyển và theo dõi vị trí đơn hàng.
	Thông báo khi đơn hàng đã được giao thành công
Hỗ trợ trực tuyến	Cung cấp tính năng chat trực tuyến hoặc hỗ trợ qua điện thoại để giải đáp thắc mắc của người dùng.

3.5. Kết luận chương 3

Chương 3 trình bày về quá trình thực hiện xây dựng ứng dụng bán hàng trên điện thoại di động, sử dụng API, MongoDB, NodeJS và biên dịch ứng dụng trên các nền tảng khác nhau. Cụ thể, cách tích hợp API để tương tác linh hoạt với dữ liệu và chức năng từ các nguồn khác nhau, tạo ra một hệ thống mạnh mẽ và mở rộng. Sự linh hoạt này không chỉ tăng cường khả năng mở rộng của ứng dụng mà còn giúp tối ưu hóa trải nghiệm người dùng thông qua các tính năng và dịch vụ đa dạng. Với việc lưu trữ và quản lý dữ liệu MongoDB có tính linh hoạt và khả năng mở rộng đã giúp xử lý lượng dữ liệu lớn mà không làm giảm hiệu suất hệ thống. Kết hợp sử dụng backend bằng NodeJS để xây dựng ứng dụng.

ĐẠI HỌC SAO ĐỎ

KẾT LUẬN

1. Kết quả đạt được

Sau quá trình tìm hiểu và triển khai đồ án “Nghiên cứu Framework đa nền tảng Flutter, xây dựng ứng dụng bán hàng trên điện thoại di động” đã thực hiện được đầy đủ những yêu cầu đặt ra, các kết quả có thể kể đến đó là:

- Về mặt lý thuyết: Tìm hiểu được một số Framework phát triển ứng dụng đa nền tảng nói chung và Framework Flutter nói riêng trong việc xây dựng ứng dụng di động đa nền; cách thức lưu trữ và quản lý dữ liệu bằng MongoDB; nghiên cứu về cách API được sử dụng để giao tiếp giữa ứng dụng di động và cơ sở dữ liệu. Kết quả của nghiên cứu còn cung cấp hướng dẫn và thông tin hữu ích cho những người quan tâm đến việc phát triển ứng dụng di động đa nền tảng sử dụng công nghệ hiện đại.

Về mặt thực tiễn: Xây dựng được ứng dụng bán hàng trên điện thoại đảm bảo được các yêu cầu đặt ra trong phần mục tiêu và nội dung nghiên cứu. Ứng dụng đã hoàn thành đúng với thiết kế ban đầu. Kết quả thực hiện đúng với điều kiện cần cũng như điều kiện đủ để tạo ra sản phẩm. Giao diện ứng dụng dễ sử dụng, trực quan và không có nhiều thao tác phức tạp, giúp cho người dùng dễ dàng sử dụng, với đầy đủ các chức năng. Sự tích hợp linh hoạt của API, khả năng lưu trữ mạnh mẽ của MongoDB, tốc độ và linh hoạt của NodeJS cùng với việc biên dịch hiệu quả trên nền tảng đã tạo ra một ứng dụng mạnh mẽ, đáp ứng đầy đủ yêu cầu và đồng thời mang lại trải nghiệm tốt cho người dùng trên mọi thiết bị.

2. Kiến nghị

Để ứng dụng có thể hoạt động hiệu quả hơn, cần có sự kết hợp về giao diện chuyên nghiệp, dễ sử dụng với các chức năng có thể nghiên cứu và phát triển của đề tài như sau: Phát triển các chức năng chưa có tại phiên bản hiện tại như tích hợp thêm chức thanh toán qua ngân hàng,...

TÀI LIỆU THAM KHẢO

- [1] Mark Clow (2019), Learn Google Flutter Fast: 65 Example Apps, ebook.
- [2] <https://docs.flutter.dev/release/archive?tab=macos>. Truy cập 05/11/2023.
- [3] <https://flutter.dev/docs/get-started/install/windows>. Truy cập 05/11/2023.
- [4] <https://wiki.tino.org/react-native-la-gi/>. Truy cập 26/10/2023.
- [5] <https://viblo.asia/p/tong-quan-ve-ionic-framework-DljMbVqZMVZn>. Truy cập 12/10/2023.
- [6] <https://bizfly.vn/techblog/xamarin-la-gi.html>. Truy cập 12/10/2023.

DÀI HỌC SAO ĐỎ