**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

TRƯỜNG ĐẠI HỌC THỦY LỢI



ĐỒ ÁN TỐT NGHIỆP

đề tài: XÂY DỰNG ỨNG DỤNG MUA BÁN ĐỒ ĂN TRÊN NỀN TẢNG IOS

|  |  |
| --- | --- |
| Sinh Viên: | Vũ Đức Đạt |
| Mã sinh viên: | 175A071221 |
| Lớp: | 59TH1 |

|  |  |
| --- | --- |
| NGƯỜI HƯỚNG DẪN: | ThS. NGUYỄN VĂN NAM |

HÀ NỘI, NĂM 2022

**HỌ VÀ TÊN: VŨ ĐỨC ĐẠT ĐỒ ÁN TỐT NGHIỆP HÀ NỘI, NĂM 2022**

|  |  |
| --- | --- |
| logo | TRƯỜNG ĐẠI HỌC THUỶ LỢI  **KHOA CÔNG NGHỆ THÔNG TIN**  BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP |

TÊN ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG MUA BÁN ĐỒ ĂN TRÊN NỀN TẢNG IOS

*Sinh viên thực hiện*: *Vũ Đức Đạt*

*Lớp*: *59TH1*

*Giáo viên hướng dẫn*: ThS. *Nguyễn Văn Nam*

**TÓM TẮT ĐỀ TÀI**

Ngày nay khoa học kỹ thuật trên thế giới nói chung, ở Việt Nam nói riêng đang trên đà phát triển mạnh mẽ và không ngừng nâng cao phát triển về mọi mặt. Đặc biệt là ngành công nghệ thông tin. Để đáp ứng nhu cầu ngày càng cao của con người thì hàng loạt các công ty về máy tính được ra đời và các ứng dụng công nghệ thông tin đang được áp dụng ngày một nhiều hơn.Với công nghệ hiện đại như ngày nay thì con người dần dần được giải phóng bởi những công việc nặng nhọc và thay vào đó là những ứng dụng của công nghệ thông tin ngày một hiện đại và kỹ xảo như các robot, máy tự động, các phần mềm quản lý và rất nhiều các ứng dụng đang được áp dụng ngoài thực tiễn. Đặc biệt trong thời gian dịch bệnh COVID diễn ra nghiêm trọng như hiện nay thì giải pháp phần mềm ngày càng phổ biến và cần thiết.

Ứng dụng “Mua bán đồ ăn” ra đời với mục đích chính là giúp khách hàng có thể đặt mua các thức ăn một cách nhanh chóng ngay tại nhà mà không cần phải đến tận cửa hàng để mua đồ ăn đặc biệt là trong mùa dịch bệnh này.

IOS là một hệ điều hành điện thoại di động được phát triển bởi Apple và phân phối độc quyền cho phần cứng của Apple. Hiện nay IOS đang chiếm lĩnh rất cao trong thị phần điện thoại thông minh.Từ những ý nghĩa thực tiễn đố , em đã chọn đề tài “XÂY DỰNG ỨNG DỤNG MUA BÁN ĐỒ ĂN TRÊN NỀN TẢNG IOS”

**CÁC MỤC TIÊU CHÍNH**

* Mục tiêu 1: Tìm hiểu và thực hiện Đồ án tốt nghiệp bằng Swift.
* Mục tiêu 2: Ứng dụng mua bán đồ ăn trên IOS.

**KẾT QUẢ DỰ KIẾN**

* Hoàn thành ứng dụng mua bán đồ ăn trên nền tảng IOS.
* Tổng hợp thông tin, viết báo cáo đồ án tốt nghiệp.

LỜI CAM ĐOAN

Em xin cam đoan đây là Đồ án tốt nghiệp của bản thân em xây dựng. Các kết quả trong Đồ án tốt nghiệp này là trung thực, và không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

|  |  |
| --- | --- |
|  |  |
|  |  |

**Lời cảm ơn**

Em xin cảm ơn chân thành đến toàn thể thầy cô trong trường Đại Học Thủy Lợi nói chung và các thầy cô trong khoa Công nghệ thông tin nói riêng, những người đã tận tình hướng dẫn, dạy dỗ và trang bị cho em những kiến thức bổ ích trong 4 năm vừa qua.

Em xin chân thành gửi lời cảm ơn sâu sắc đến các thầy cô khoa công nghệ thông tin đặc biệt là thầy giáo ThS Nguyễn Văn Nam, người đã tận tình hướng dẫn, trực tiếp chỉ bảo và tạo mọi điều kiện giúp đỡ em trong suốt quá trình làm đồ án tốt nghiệp.

Sau cùng em xin gửi lời cảm ơn chân thành tới những ngƣời bạn đã động viên, cổ vũ và đóng góp ý kiến trong quá trình học tập, nghiên cứu cũng như quá trình làm đồ án tốt nghiệp. Bên cạnh đó, do còn nhiều hạn chế về kiến thức và kinh nghiệm nên trong đồ án không tránh khỏi những thiếu sót, kính mong quý thầy cô chỉ bảo thêm.

***Em xin chân thành cảm ơn !***

MỤC LỤC

Contents

[CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI VÀ CƠ SỞ LÝ THUYẾT 1](#_Toc92122098)

[1.Tổng quan đề tài 1](#_Toc92122099)

[1.1. Bài toán đặt ra. 1](#_Toc92122100)

[1.2. Mục tiêu của đề tài: 2](#_Toc92122101)

[2. Cơ sở lý thuyết 2](#_Toc92122102)

[2.1. Swift 2](#_Toc92122103)

[2.2. Web API 4](#_Toc92122104)

[2.3. MongoDB 5](#_Toc92122105)

[2.4. Mô hình MVVM(Swift) 7](#_Toc92122106)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 10](#_Toc92122107)

[1. Thu thập và phân tích yêu cầu 10](#_Toc92122108)

[1.1. Thu thập dữ liệu.: 10](#_Toc92122109)

[1.2. Phân loại các yêu cầu 12](#_Toc92122110)

[2. Xác định Usecase và vẽ biểu đồ Usecase 14](#_Toc92122111)

[2.1. Liệt kê actor 14](#_Toc92122112)

[2.2. Xác định các Usecase 14](#_Toc92122113)

[2.3. Danh sách các Use case 15](#_Toc92122114)

[2.4. Lược đồ tổng quát của các màn Activity 17](#_Toc92122115)

[2.5. Xây dựng kịch bản cho Usecase 17](#_Toc92122116)

[2.6. Biểu đồ Usecase 22](#_Toc92122117)

[3. Biểu đồ hoạt động Activity 25](#_Toc92122118)

[4. Xây dựng biểu đồ lớp (Class Diagram) 27](#_Toc92122119)

[4.1. Xác định các lớp 27](#_Toc92122120)

[4.2. Bản vẽ Class diagram 28](#_Toc92122121)

[5. Biểu đồ trạng thái 29](#_Toc92122122)

[5.1. Đơn hàng 29](#_Toc92122123)

[5.2. Sản phẩm 29](#_Toc92122124)

[5.3. Người dùng 30](#_Toc92122125)

[5.4. Giỏ hàng 30](#_Toc92122126)

[5.5. Voucher 30](#_Toc92122127)

[5.6. Đơn vị vận chuyển 31](#_Toc92122128)

[5. Thiết kế cơ sở dữ liệu 31](#_Toc92122129)

[5.1. Xác định các bảng trong CSDL 31](#_Toc92122130)

[5.2. Mô hình quan hệ 38](#_Toc92122131)

[CHƯƠNG 3. KẾT QUẢ VÀ ĐÁNH GIÁ 39](#_Toc92122132)

[1. Ngôn ngữ lập trình và công nghệ sử dụng 39](#_Toc92122133)

[1.1. Lựa chọn công nghệ 39](#_Toc92122134)

[1.2. Swift 39](#_Toc92122135)

[1.3. CocoaPod 40](#_Toc92122136)

[1.4. ExpressJs 40](#_Toc92122137)

[1.5. ExpressJs 40](#_Toc92122138)

[2. Môi trường phát triển 40](#_Toc92122139)

[2.1. IDE (Integrated development environment) – Visual Studio Code 40](#_Toc92122140)

[2.2. Source control (Quản lý mã nguồn) – Git 41](#_Toc92122141)

[2.3. Xcode 41](#_Toc92122142)

[2.4. Nodejs 42](#_Toc92122143)

[2.3.MongoDB 42](#_Toc92122144)

[2.6. CocoaPods 49](#_Toc92122145)

[3. Giao diện Ứng dụng cho người dùng 51](#_Toc92122146)

[3.1. Màn hình đăng nhập 51](#_Toc92122147)

[3.2. Màn hình đăng ký 52](#_Toc92122148)

[3.3. Màn hình chính Của ứng dụng 53](#_Toc92122149)

[53](#_Toc92122150)

[3.4. Màn hình của cửa hàng 54](#_Toc92122151)

[3.5. Màn hình hiển thị khi người dùng nhấp vào một sản phẩm 55](#_Toc92122152)

[3.6. Màn hình trang giỏ hàng 56](#_Toc92122153)

[3.7. Màn hình thanh toán 57](#_Toc92122154)

[57](#_Toc92122155)

[3.8. Màn hình thanh toán. 58](#_Toc92122156)

[3.9. Màn hình thông tin cá nhân. 59](#_Toc92122157)

[4. Server 60](#_Toc92122158)

[6. Source Code 60](#_Toc92122159)

[Kết luận 79](#_Toc92122160)

[TÀI LIỆU THAM KHẢO 80](#_Toc92122161)

**DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ**

**API** Application programming interface

**LỜI NÓI ĐẦU**

Ngày nay khoa học kỹ thuật trên thế giới nói chung, ở Việt Nam nói riêng đang trên đà phát triển mạnh mẽ và không ngừng nâng cao phát triển về mọi mặt. Đặc biệt là ngành công nghệ thông tin.Để đáp ứng nhu cầu ngày càng cao của con người thì hàng loạt các công ty về máy tính được ra đời và các ứng dụng công nghệ thông tin đang được áp dụng ngày một nhiều hơn.Với công nghệ hiện đại như ngày nay thì con người dần dần được giải phóng bởi những công việc nặng nhọc và thay vào đó là những ứng dụng của công nghệ thông tin ngày một hiện đại và kỹ xảo như các robot, máy tự động, các phần mềm quản lý và rất nhiều các ứng dụng đang được áp dụng ngoài thực tiễn. Đặc biệt trong thời gian dịch bệnh COVID diễn ra nghiêm trọng như hiện nay thì giải pháp phần mềm ngày càng phổ biến và cần thiết.

Ứng dụng đặt đồ ăn online có lẽ đã là những phần mềm, những tiện ích quá quen thuộc với mỗi chúng ta rồi đúng không nào?

Chúng là một trong những sản phẩm tiêu biểu, gợi nhớ rằng mỗi chúng ta đang sống trong thời đại công nghệ hiện đại đến mức chỉ việc ngồi ở nhà và … hít thở thôi cũng có thể làm được “tất tần tật” mọi thứ trên đời. Những dịch vụ giao hàng tận tay đã thay thế những thủ tục rườm rà của hình thức giao gửi tận cửa bưu điện “lỗi thời”.

Thương mại điện tử thay thế các chợ truyền thống, giúp cho người người, nhà nhà đều có thể mua sắm tại gia mà chẳng cần đi đâu xa. Tương tự, các ứng dụng đặt món, giao thức ăn nhanh cũng đang trở thành một xu hướng đang phát triển “rầm rộ” và nhận được sự hưởng ứng “khủng” của người tiêu dùng nhờ những tiện ích không thể chối cãi của nó.

Ứng dụng “Mua bán đồ ăn” ra đời với mục đích chính là giúp khách hàng có thể đặt mua các thức ăn một cách nhanh chóng ngay tại nhà mà không cần phải đến tận cửa hàng để mua đồ ăn đặc biệt là trong mùa dịch bệnh này.

IOS là một hệ điều hành điện thoại di động được phát triển bởi Apple và phân phối độc quyền cho phần cứng của Apple. Hiện nay IOS đang chiếm lĩnh rất cao trong thị phần điện thoại thông minh.Từ những ý nghĩa thực tiễn đố , em đã chọn đề tài “XÂY DỰNG ỨNG DỤNG MUA BÁN DỒ ĂN TRÊN NỀN TẢNG IOS”

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI VÀ CƠ SỞ LÝ THUYẾT

## 1.Tổng quan đề tài

**Thực phẩm** hay còn gọi là thức ăn là một không thể thiếu để duy trì sự sống sức khỏe. Đặc biết trong xã hội ngày càng phát triển như hiện nay thì nhu cầu đó lại càng phát triển hơn nữa. Đi kèm với sự phát triển của công nghệ đặc biệt là ngành công nghiệp Hiện đại, cùng với sự ra đời của các phần mềm di động giúp cung cấp càng nhiều thông tin, lợi ích cho con người. Một trong số đó là nhu cầu ăn uống với giải pháp của các **Ứng dụng đặt đồ ăn.**

### 1.1. Bài toán đặt ra.

Về phía Người dùng: Họ sẽ truy cập vào ứng dụng tìm và đặt mua thứ mình muốn. Sau khi đặt mua, đơn hàng sẽ được chuyển đến hệ thống quản lý gọi điện xác nhận thông qua thông tin là số điện thoại của khách hàng. Khi xác nhận xong, cửa hàng sẽ lấy đủ sản phẩm được đặt trong đơn giao cho các đơn vị vận chuyển, người giao hàng của các đơn vị vận chuyển sẽ tiến hành giao hàng đến tận tay người mua theo địa chỉ được cung cấp trong đơn hàng. Khi nhận được hàng thì người mua sẽ tiến hành thanh toán trực tiếp cho người giao hàng.

Về phía ứng dụng: Hệ thống phải đảm bảo cho người dùng truy cập Ứng dụng dễ dàng lựa chọn các sản phẩm, xem các khuyến mãi, các sản phẩm được ghi nguồn gốc rõ ràng, quy trình sản xuất để khách hàng có thể an tâm chọn mua, việc tìm kiếm sản phẩm và mua hàng thuận tiện, dễ sử dụng cho cả những người có ít kiến thức về tin học. Hệ thống cũng giúp cho cửa hàng có thể nhận được các đơn hàng mà các khách hàng gửi đến và sau khi đã lấy đủ hàng theo đơn hàng, nhân viên giao hàng của các đơn vị vận chuyển có thể vận chuyển hàng tới cho khách và thu tiền để hoàn tất giao dịch của một đơn hàng.

Về phía server: Server sẽ cung cấp dữ liệu cho ứng dụng qua các API, Server có thể quản lý được các sản phẩm, cửa hàng, …

### 1.2. Mục tiêu của đề tài:

* Nghiên cứu các kỹ thuật, công nghệ phát triển Ứng dụng iOS
* Phân tích thiết kế hệ thống cho Ứng dụng bán đồ ăn
* Xây dựng thành công sản phẩm demo, cụ thể là Ứng dụng mua bán đồ ăn theo bản phân tích thiết kế hệ thống.
* Chạy thử nghiệm trên local.
* Phân tích đánh giá sản phẩm.

## 2. Cơ sở lý thuyết

### 2.1. Swift

Giới thiệu ngôn ngữ Swift

* Swift là một ngôn ngữ lập trình hướng đối tượng dành cho việc phát triển iOS và OS X, được giới thiệu bởi Apple tại hội nghị WWDC 2014. Swift được mong đợi sẽ tồn tại song song cùng Objective-C, ngôn ngữ lập trình hiện tại dành cho các hệ điều hành của Apple
* Các nhà thiết kế Swift đã lấy ý tưởng từ nhiều ngôn ngữ phổ biến khác như Objective-C, Rust, Haskell, Ruby, Python, C # và CLU.
* Swift được thiết kế là một ngôn ngữ có khả năng phòng chống lỗi cao

Đặc điểm

* Là một ngôn ngữ có khả năng chống lỗi cao khi có nhiều cú pháp lập trình an toàn
* Sử dụng cú pháp lập trình hiện đại, bên cạnh bộ cú pháp của Objective – C
* Đơn giản để đọc và hiểu
* Là ngôn ngữ tuyệt vời để phát triển ứng dụng trên iOS và OS X
* Swift đi kèm với một tính năng sân chơi nơi các lập trình viên Swift có thể viết mã của họ và thực thi nó để xem kết quả ngay lập tức
* Swift thống nhất được phần thủ tục và hướng đối tượng.

Ưu điểm so với Objective-C

* Swift chạy nhanh hơn, tương đương với C++
* Swift dễ đọc và dễ học hơn so với Objective-C (cú pháp hoàn toàn mới, ngắn gọn hơn rất nhiều)
* Các file trong Swift thống nhất nên bảo trì code đơn giản hơn.
* Compiler của Swift tốt hơn Objective-C
* Swift không sử dụng con trỏ giúp cho code an toàn hơn, và giúp lập trình viên lược bỏ phần khái niệm khó nhằn về con trỏ.
* Swift quản lý bộ nhớ tốt hơn
* Swift là mã nguồn mở: Lập trình viên có thể xem được mã nguồn, chỉnh sửa, vá lỗi

Nhược điểm so với Objective-C

* Nhiều API không làm việc với Swift
* Nhiều qui tắc khi khởi tạo, lựa chọn kiểu optional
* Một lượng lớn các project, phần mềm trước đó đều viết bằng Objective-C, do đó khi cần bảo trì, phát triển và tìm kiếm Objective-C vẫn dễ tìm kiếm hơn.

Lịch sử phát triển

* **Swift 1.2** là bản cập nhật lớn đầu tiên cho Swift, tinh chỉnh các tính năng ngôn ngữ khác nhau và cải thiện các tính năng khác.
* **Swift 2** là bản cập nhật lớn thứ hai cho Swift, giới thiệu các ngoại lệ được kiểm tra và nhiều cải tiến lớn khác.
* **Swift 2.2** là một bản cập nhật nhỏ cho Swift 2.0, không dùng một số cú pháp trước khi loại bỏ nó trong Swift 3.
* **Swift 3** là bản cập nhật lớn thứ ba cho Swift và đã mang đến một số lượng lớn các thay đổi về cách đặt tên để làm cho ngôn ngữ dễ viết và tự nhiên hơn để viết.
* **Swift 4** là bản cập nhật lớn thứ tư cho Swift và tập trung chủ yếu vào các tính năng mới giúp ngôn ngữ trở nên biểu cảm hơn.
* **Swift 4.2** chính thức là một bản cập nhật nhỏ cho Swift 4.0, nhưng trên thực tế đã cung cấp rất nhiều tính năng mới đã lấp đầy khoảng trống trước khi Swift 5 được phát hành.
* **Swift 5** là bản cập nhật lớn thứ năm cho Swift và tập trung vào việc làm cho ngôn ngữ ổn định trong thời gian dài.
* **Swift 5.1** nghe có vẻ như một bản cập nhật nhỏ, nhưng trên thực tế được tích hợp nhiều tính năng để cho phép SwiftUI hoạt động trơn tru
* **Swift 5.2** là bản cập nhật hoàn thiện hơn các về các thành phần, tập trung vào Swift UI

### 2.2. Web API

**API** là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của **Application Programming Interface** – giao diện lập trình ứng dụng. API cung cấp khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng

**Web API** là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp, trao đổi dữ liệu qua lại. Dữ liệu được Web API trả lại thường dạng [JSON](https://topdev.vn/blog/json-la-gi/) hoặc XML thông qua giao thức HTTP hoặc HTTPS.

* **Ưu điểm**
* Web API được sử dụng hầu hết trên các ứng dụng desktop, ứng dụng mobile và ứng dụng website.
* Linh hoạt với các định dạng dữ liệu khi trả về client: Json, XML hay định dạng khác.
* Nhanh chóng xây dựng HTTP service: URI, request/response headers, caching, versioning, content formats và có thể host trong ứng dụng hoặc trên IIS.
* Mã nguồn mở, hỗ trợ chức năng RESTful đầy đủ, sử dụng bởi bất kì client nào hỗ trợ XML, Json.
* Hỗ trợ đầy đủ các thành phần MVC như: routing, controller, action result, filter, model binder, IoC container, dependency injection, unit test.
* Giao tiếp hai chiều được xác nhận trong các giao dịch, đảm bảo độ tin cậy cao.
* **Nhược điểm**

Do web API còn khá mới nên chưa thể đánh giá nhiều về nhược điểm của mô hình nay. Tuy nhiên, có hai nhược điểm dễ dàng nhận thấy:

* Web API chưa hoàn toàn phải là RESTful service, mới chỉ hỗ trợ mặc định GET, POST
* Để sử dụng hiệu quả cần có kiến thức chuyên sâu, có kinh nghiệm backend tốt
* Tốn thời gian và chi phí cho việc phát triển, nâng cấp và vận hành
* Có thể gặp vấn đề về bảo mật khi hệ thống bị tấn công nếu không giới hạn điều kiện kỹ.

### 2.3. MongoDB

Hiện nay có khá nhiều hệ quản trị cơ sở dữ liệu dạng NoSQL, một trong số đó phải kể đến MongoDB. Bài viết sau sẽ giúp mọi người có được cái nhìn sơ lược về database này.

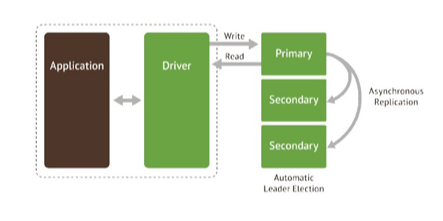
KHÁI NIỆM CƠ BẢN VỀ MONGODB?

MongoDB là một mã nguồn mở và là một tập tài liệu dùng cơ chế NoSQL để truy vấn, nó được viết bởi ngôn ngữ C++. Chính vì được viết bởi C++ nên nó có khả năng tính toán với tốc độ cao chứ không giống như các hệ quản trị CSDL hiện nay. Mỗi một table (bảng dữ liệu) trong SQL sử dụng thì trong MongoDB gọi là collection (tập hợp) Mỗi một record (bản ghi) trong MongoDB được gọi là document (tài liệu). Một bản ghi của MongoDB được lưu trữ dưới dạng document (tài liệu), nó được ghi xuống với cấu trúc field (trường) và value (giá trị). Nó giống như là một đối tượng JSON có dạng như sau:



Điều đó giúp bạn có thể dễ dàng ép kiểu sang array (mảng) để lập trình các ứng dụng một cách dễ dàng hơn. Nói một cách dễ hiểu thì mỗi một bản ghi của MongoDB là một mảng dữ liệu riêng biệt bao gồm các cặp key, value khác nhau do đó cách lưu trữ của MongoDB là phi cấu trúc dữ liệu.

MONGODB HOẠT ĐỘNG NHƯ THẾ NÀO?

****

MongoDB hoạt động dưới một tiến trình ngầm service luôn mở một cổng (Cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.

Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên “\_id” thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định được tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau. Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.

Mỗi khi có một truy vấn dữ liệu, bản ghi được cache (ghi đệm) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng. Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

ƯU VÀ NHƯỢC ĐIỂM CỦA MONGODB LÀ GÌ?

Ưu điểm:

* Dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ.
* Dữ liệu được caching (ghi đệm) lên RAM, hạn chế truy cập vào ổ cứng nên tốc độ đọc và ghi cao.

 Nhược điểm:

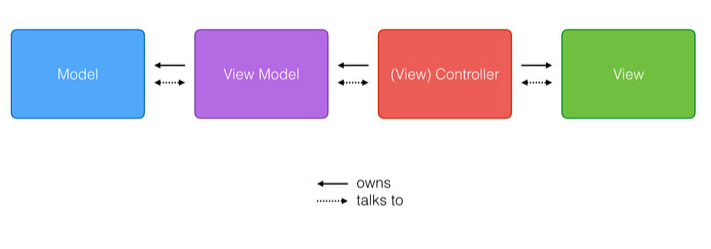
* Không ràng buộc, toàn vẹn nên không ứng dụng được cho các mô hình giao dịch yêu cầu độ chính xác cao.
* Không có cơ chế transaction (giao dịch) để phục vụ các ứng dụng ngân hàng.
* Dữ liệu được caching, lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn.
* Như đã giới thiệu ở trên, mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao.\

### 2.4. Mô hình MVVM(Swift)

Hầu hết ứng dụng iOS đều được viết dựa trên **MVC pattern (Model-View-Controller)**. Ưu điểm của **MVC** là đơn giản và dễ dàng sử dụng. Nhưng đôi khi sử dụng **MVC** trong một số trường hợp thì nó cũng gặp nhiều vấn đề, chưa hoàn hảo lắm nên **MVVM** là một sự thay đổi hấp dẫn.

Trong project làm bằng **MVC** pattern, bạn sẽ phải quyết định viết đoạn code ở chỗ nào. Nên viết ở Model, View và thường xuyên để hết ở **Controller**. Điều này gây ra vấn để là làm cho **controller** quá lớn đối với các màng hình phước tạp và rất khó để test và quản lý. Đối với **MVVM** thì làm việc này tốt hơn và việc add thêm một lớp layer **ViewModel** ở giữa. **ViewModel** sẽ chuyển đổi thông tin từ lớp model và truyền cho thông tin cho view cần dùng. **Controller** sẽ không còn chịu trách nhiệm về phần này nửa.

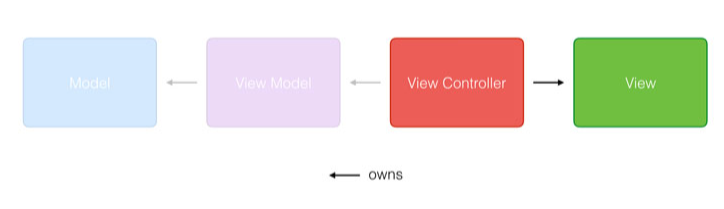
Sơ đồ làm phân luồng của **MVVM** được chia thành 4 lớp:



Khi làm việc với **MVVM** thì cần phải hiểu **6 rule**, coi **6 rule** như một hướng dẫn sử dụng cũng được.

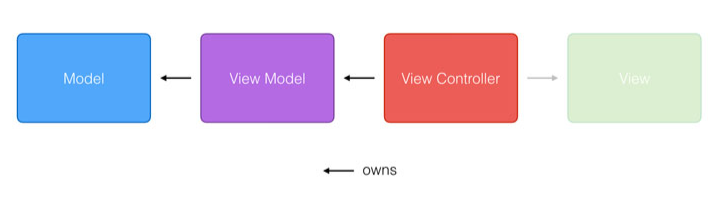
**Rule 1**

**View** không được biết thông tin gì về **Controller** chứa nó. Nó chỉ hiểu **thông tin vừa đủ** để hiện thị thông tin mà **controller** đưa cho nó thôi.



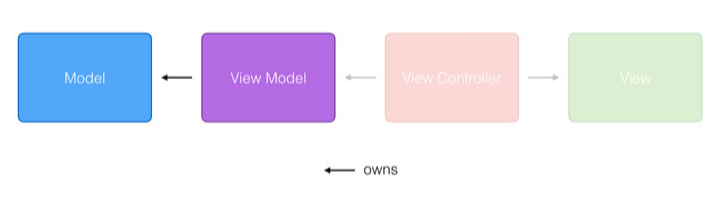
**Rule 2**

**Controller** không được biết gì về **Model**. Đấy là nguyên nhân tạo ra **MVVM** từ **MVC**



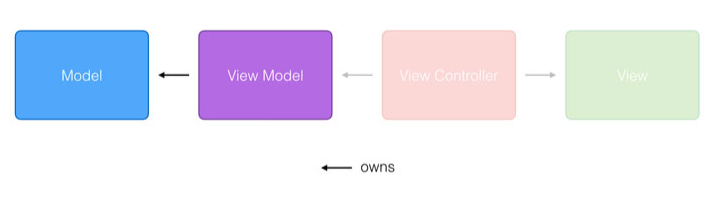
**Rule 3**

**Model** không được biết về **View Model** chưa nó.



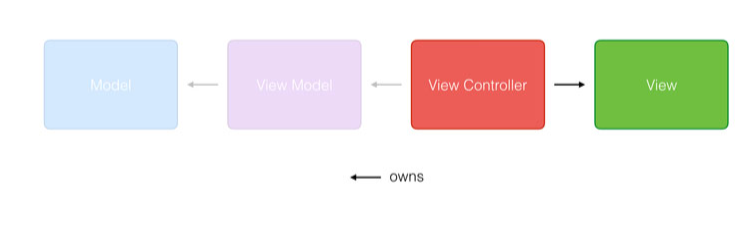
**Rule 4**

**ViewModel** là nơi chứa **model**. Còn đối với **MVC** thì **Controller** làm việc này.



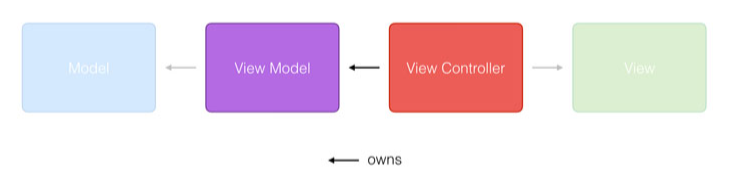
**Rule 5**

**Controller** chứa **view**.



**Rule 6**

Và cuối cùng, **Controller** chứa **ViewModel**. Controller sẽ tháo tác với một hoặc nhiều **ViewModel**



CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 1. Thu thập và phân tích yêu cầu

### 1.1. Thu thập dữ liệu.:

Phần lớn các yêu cầu đến từ khảo sát thực tế và thảo luận của các nhà phát triển ta có:

Bảng 1 Bảng danh sách các yêu cầu

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Vai trò | Phương pháp thu thập | Kết quả |
|  | Người dùng | Phỏng vấn, phân vai | * Sử dụng ngôn ngữ tiếng Anh và tiếng Việt * Menu sắp xếp khoa học, bố cục Ứng dụng rõ ràng * Giao diện đơn giản, có thể được dễ dàng sử dụng kể cả đối với người có ít khả năng về công nghệ. * Ứng dụng không bị giật lag trong quá trình sử dụng, đặc biệt là quá trình mua hàng và thanh toán * Có chức năng đăng ký, đăng nhập, đăng xuất * Có chức năng theo dõi đơn hàng, xem các đơn hàng đã mua theo trạng thái * Có thể hủy đơn hàng khi đơn hàng chưa được xác nhận * Có chức năng bình luận, đánh giá sản phẩm * Có chức năng quản lý giỏ hàng để khách hàng có thể thêm các sản phẩm vào giỏ trước khi đặt mua * Có thể chọn các đơn vị vận chuyển khác nhau. * Có chức năng tìm kiếm theo tên sản phẩm * Trang thanh toán hiển thị rõ ràng, rành mạch có chức năng sử dụng phiếu giảm giá * Có tính năng gửi phản hồi từ người mua hàng. * Giao hàng theo thông tin mà người dùng cung cấp trong thông tin của tài khoản * Sản phẩm phải có nguồn gốc xuất xứ rõ ràng * Sản phẩm phải có các hình ảnh mô tả thực tế. * Ứng dụng phải hiển thị địa chỉ cửa hàng * Có thể thanh toán bằng nhiều hình thức như: thanh toán trực tiếp, thanh toán qua chuyển khoản ngân hàng, thanh toán qua các ví điện tử, tiền ảo… * Có chức năng chỉnh sửa thông tin tài khoản * Có chức năng thêm ảnh avatar vào tài khoản * Có chức năng đổi mật khẩu |
|  | Quản trị | Phỏng vấn | * Người quản trị có thể thêm sản phẩm mới vào danh sách sản phẩm của ứng dụng, ẩn sản phẩm, cập nhật giá bán, khuyến mại của các sản phẩm * Người quản trị có thể xem doanh thu, xem thống kê về dữ liệu bán hàng. * Có khả năng lưu trữ một số lượng lớn dữ liệu về người mua, sản phẩm, đơn hàng. * Có thể xác nhận người mua đã thanh toán bằng hình thức chuyển khoản. * Có thể theo dõi được tình trạng đơn hàng. * Có thể xem được lịch sử bán hàng. * Có thể ẩn được những đơn hàng mà bị hủy hoặc đơn hàng đã hoàn thành. * Có thể thêm sửa xóa các danh mục và hãng Ứng dụng * Màn hình quản lý có bố cục rõ ràng, dễ thao tác * Có giao diện thân thiện, màu sắc dịu nhẹ không quá nổi bật |

### 1.2. Phân loại các yêu cầu

Bảng 2 Bảng phân loại các yêu cầu

|  |  |
| --- | --- |
| **Yêu cầu chức năng** | **Yêu cầu phi chức năng** |
| * Có thể ẩn được những đơn hàng mà bị hủy hoặc đơn hàng đã hoàn thành. * Có thể thêm sửa xóa các danh mục trên Ứng dụng * Trên các màn hình nhập liệu, sẽ chỉ ra các trường có tính chất bắt buộc. * Người quản trị có thể thêm sản phẩm mới vào danh sách sản phẩm của trang ứng dụng, ẩn sản phẩm, cập nhật giá bán, khuyến mại của các sản phẩm * Người quản trị có thể xem doanh thu, xem thống kê về dữ liệu bán hàng. * Có khả năng lưu trữ một số lượng lớn dữ liệu về người mua, sản phẩm, đơn hàng. * Có thể xác nhận người mua đã thanh toán bằng hình thức chuyển khoản. * Có thể theo dõi được tình trạng đơn hàng. * Có thể xem được lịch sử bán hàng. * Có chức năng đăng ký, đăng nhập, đăng xuất * Có chức năng theo dõi đơn hàng, xem các đơn hàng đã mua theo trạng thái * Có thể hủy đơn hàng khi đơn hàng chưa được xác nhận * Có chức năng bình luận, đánh giá sản phẩm * Có chức năng quản lý giỏ hàng để khách hàng có thể thêm các sản phẩm vào giỏ trước khi đặt mua * Có thể chọn các đơn vị vận chuyển khác nhau * Có chức năng tìm kiếm theo tên sản phẩm * Trang thanh toán hiển thị rõ ràng, rành mạch có chức năng sử dụng phiếu giảm giá nếu có * Có tính năng gửi phản hồi từ người mua hàng. * Sản phẩm phải có các hình ảnh mô tả thực tế. * Ứng dụng hiển thị địa chỉ cửa hàng * Có thể thanh toán bằng nhiều hình thức như: thanh toán trực tiếp, thanh toán qua chuyển khoản ngân hàng, thanh toán qua các ví điện tử, tiền ảo… * Có chức năng chỉnh sửa thông tin tài khoản * Có chức năng thêm ảnh avatar vào tài khoản * Có chức năng đổi mật khẩu | * Máy chủ phải có dung lượng lớn để lưu trữ data * Người dùng có thể truy cập từ tất cả các thiết bị có kết nối với Internet thông qua Ứng dụng * Màn hình ứng dụng có bố cục rõ ràng, dễ thao tác * Có giao diện thân thiện, màu sắc dịu nhẹ không quá nổi bật * Giao hàng theo thông tin mà người dùng cung cấp trong thông tin của tài khoản * Các sản phẩm phải có nguồn gốc xuất xứ rõ ràng * Sử dụng ngôn ngữ tiếng Anh và tiếng Việt * Menu sắp xếp khoa học, bố cục Ứng dụng rõ ràng * Giao diện đơn giản, có thể được dễ dàng sử dụng kể cả đối với người có ít khả năng về công nghệ. * Ứng dụng không bị giật lag trong quá trình sử dụng, đặc biệt là quá trình mua hàng và thanh toán |

## 

## 2. Xác định Usecase và vẽ biểu đồ Usecase

### 2.1. Liệt kê actor

-Xét theo hệ thống chúng ta nhận thấy:

Những người muốn mua đồ ăn vào ứng dụng để xem và tìm kiếm các thông tin .Những người này gọi là **Khách hàng tiềm năng.**

Những người đã đặt hàng vào ứng dụng để xem đơn hàng, thực hiện thanh toán,… gọi là **Khách hàng.**

Về phía đơn các cửa hàng sẽ do **Quản trị viên** quản lý các cửa hàng , quản lý các sản phẩm,đơn hàng trực tiếp qua cơ sở dữ liệu.

### 2.2. Xác định các Usecase

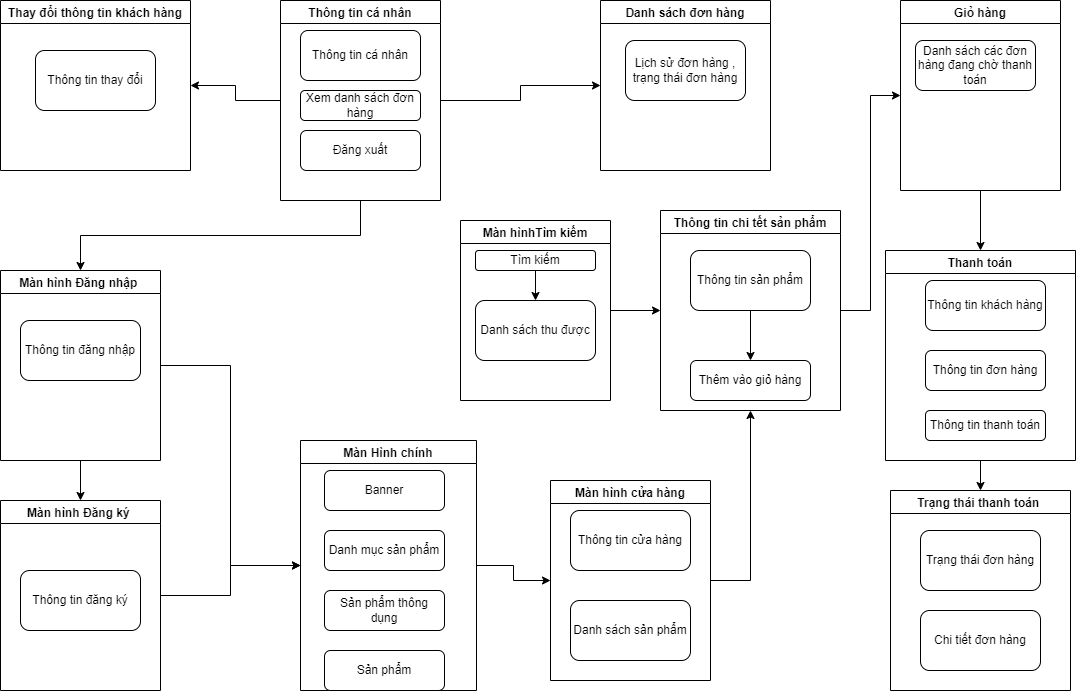
* **Khánh hàng tiềm năng**
  + Có thể bỏ qua chức năng Đăng nhập,Đăng ký.
  + Xem và tìm kiếm sản phẩm theo tên của sản phẩm.
  + Xem chi tiết sản phẩm được chọn.
  + Xem thông tin của Cửa hàng.
  + Thêm sản phẩm vào giỏ hàng.
* **Khách hàng**
  + Đăng nhập, Đăng ký
  + Quản lý giỏ hàng
    - Thêm xóa sản phẩm trong giỏ hàng.
  + Thanh toán
  + Thay đổi thông tin cá nhân
  + Xem danh sách đơn hàng và trạng thái đơn hàng
* **Quản trị viên**
  + Đăng nhập vào cơ sở dữ liệu
  + Quản lý các cửa hàng
    - Xem danh sách
    - Thêm mới hoặc thay đổi thông tin cửa hàng
    - Xóa cửa hàng
  + Quản lý sản phẩm
    - Xem danh sách sản phẩm
    - Thêm mới hoặc thay đổi thông tin sản phẩm
    - Xóa sản phẩm
    - Lựa chọn các sản phẩm để hiển thị trên màn hình chính
  + Quản lý danh mục
    - Xem danh sách danh mục sản phẩm
    - Thêm mới hoặc thay đổi thông tin danh mục
    - Xóa danh mục
  + Quản lý đơn hàng
    - Xem danh sách đơn hàng

### 2.3. Danh sách các Use case

Bảng 4 Bảng liệt kê các use case

|  |  |
| --- | --- |
| 1. Đăng nhập | Người dùng đăng nhập vào hệ thống bằng tài khoản riêng của mình và thực hiện các công việc cần làm. |
| 1. Đăng ký | Người dùng có thể đăng ký tài khoản trên hệ thống để sử dụng các chức năng của Ứng dụng: thêm sản phẩm vào giỏ hàng, đặt hàng, chỉnh sửa thông tin cá nhân, bình luận đánh giá sản phẩm… |
| 1. Quản lý người dùng | Quản trị viên theo dõi danh sách người dung trong hệ thống (không có quyền xóa hoặc thay đổi thông tin tài khoản) |
| 1. Quản lý danh mục sản phẩm | Quản trị viên thêm, xóa danh mục sản phẩm |
| 1. Quản lý danh sách thương hiệu | Quản trị viên thêm, xóa danh sách cửa hàng, đối tác, nhà cung cấp |
| 1. Quản lý đơn hàng | Quản trị viên có thể chọn một đơn hàng, phê duyệt đơn hàng,hủy đơn hàng, chuyển đổi trạng thái của đơn hàng, ẩn đơn hàng khi đơn đã bị hủy hoặc hoàn thành |
| 1. Tìm kiếm sản phẩm | Người dùng có thể tìm kiếm sản phẩm theo tên của sản phẩm |
| 1. Quản lý giỏ hàng | Người dùng là người thêm, sửa, xóa trong giỏ hàng. |
| 1. Đánh giá sản phẩm | Người dùng có thể đánh giá sản phẩm bằng cách bình luận hoặc đánh giá bằng sao |
| 1. Đặt hàng | Người dùng thêm sản phẩm vào giỏ hàng để tiến hành thanh toán |

### 2.4. Lược đồ tổng quát của các màn Activity



### 2.5. Xây dựng kịch bản cho Usecase

**Kịch bản Usecase đăng nhập**

|  |  |
| --- | --- |
| Tên use case | Đăng nhập |
| Tên Actor | Khách hàng |
| Mức | 1 |
| Tiền điền kiện |  |
| Đảm bảo tối thiểu | Khách hàng truy cập vào ứng dụng |
| Đảm bảo thành công |  |
| Kích hoạt | Khách hàng yêu cầu chức năng đăng nhập |
| **Hành động tác nhân** | **Phản ứng hệ thống** |
| Khách hàng chọn vào chức năng đăng nhập | Hệ thống chuyển sang màn hình đăng nhập.  Hệ thống sẽ hiển thị form đăng nhập cho khách hàng. |
| Khách hàng chọn chức năng đăng ký | Hệ thống sẽ chuyển sang màn hình đăng ký,  Để khách hàng có thể đăng ký tài khoản |
| Khách hàng chọn chức năng bỏ qua đăng nhập. | Hệ thống sẽ bỏ qua chức năng đăng nhập và chuyển đến màn hình chính của ứng dụng.Tuy nhiên người dùng không thể thực hiện được chức năng thanh toán và quản lý đơn hàng, thông tin cá nhân. |
| **Ngoại lệ** |  |
| Nhập dữ liệu | Tài khoản đăng nhập là dạng mail. Nếu khách hàng nhập sai hệ thống sẽ thông báo cho khách hàng.  Nếu khách hàng nhập trống tài khoản hoặc mật khẩu. Hệ thống sẽ hiển thị thông báo cho khách hàng. |

***Bảng 2.1 Kịch bản Usecase đăng nhập***

**Kịch bản Usecase đăng ký**

|  |  |
| --- | --- |
| Tên use case | Đăng ký |
| Tên Actor | Khách hàng |
| Mức | 1 |
| Tiền điền kiện |  |
| Đảm bảo tối thiểu | Khách hàng truy cập vào ứng dụng |
| Đảm bảo thành công |  |
| Kích hoạt | Khách hàng yêu cầu chức năng đăng ký |
| **Hành động tác nhân** | **Phản ứng hệ thống** |
| Khách hàng chọn vào chức năng đăng ký | Hệ thống chuyển sang màn hình đăng ký.  Hệ thống sẽ hiển thị form đăng ký cho khách hàng. |
| Khách hàng chọn chức năng đăng nhập | Hệ thống sẽ chuyển đến màn hình đăng nhập cho khách hàng.Để khách hàng đăng nhập |
| Khách hàng chọn chức năng bỏ qua đăng ký. | Hệ thống sẽ bỏ qua chức năng đăng nhập và chuyển đến màn hình chính của ứng dụng.Tuy nhiên người dùng không thể thực hiện được chức năng thanh toán và quản lý đơn hàng, thông tin cá nhân. |
| Khách hàng chọn đăng ký tài khoản | Hệ thống sẽ đăng ký tài thoản của khách hàng. Và tiến hành đăng nhập luôn theo thông tin đã đăng ký sau đó chuyển đến màn hình chính. |
| **Ngoại lệ** |  |
| Nhập dữ liệu | Tài khoản đăng nhập là dạng mail. Nếu khách hàng nhập sai hệ thống sẽ thông báo cho khách hàng.  Mật khẩu nhập lại không khớp với mật khẩu hệ thống thống báo cho khách hàng.  Nếu khách hàng nhập trống tài khoản hoặc mật khẩu. Hệ thống sẽ hiển thị thông báo cho khách hàng. |

***Bảng 2.2 Kịch bản Usecase* đăng ký**

**Kịch bản Usecase màn hình chính.**

|  |  |
| --- | --- |
| Tên use case | Xem màn hình chính |
| Tên Actor | Khách hàng |
| Mức | 1 |
| Tiền điền kiện |  |
| Đảm bảo tối thiểu | Khách hàng truy cập vào ứng dụng |
| Đảm bảo thành công |  |
| Kích hoạt | Khách hàng yêu cầu chức năng xem màn hình chính |
| **Hành động tác nhân** | **Phản ứng hệ thống** |
| Khách hàng yêu cầu chức năng lựa chọn vào các banner quảng cáo của các cửa hàng | Hệ thống chuyển sang màn hình của nhà hàng |
| Khách hàng yêu cầu chức năng lựa chọn vào các danh mục | Hệ thống sẽ chuyển sang màn hình xem danh sách các sản phẩm theo danh mục mà khách hàng đã lựa chọn |
| Khách hàng chọn vào một sản phẩm trên màn hình chính | Hệ thống chuyển sang màn hình của nhà hàng chứa sản phẩm và hiện đè một màn hình thông tin chi tiết của sản phẩm. Người dùng có thể thêm sản phẩm đó vào giỏ hàng. |

***Bảng 2. 3 Kịch bản Usecase trang chủ***

***Kịch bản Usecase tìm kiếm theo tên sản phẩm***

|  |  |
| --- | --- |
| Tên use case | Tìm kiếm theo tên sản phẩm |
| Tên Actor | Khách hàng |
| Mức | 2 |
| Tiền điền kiện |  |
| Đảm bảo tối thiểu | Khách hàng truy cập vào ứng dụng |
| Đảm bảo thành công |  |
| Kích hoạt | Khách hàng yêu cầu chức năng |
| **Hành động tác nhân** | **Phản ứng hệ thống** |
| Khách hàng yêu cầu chức năng tìm kiếm theo từ khóa | Hệ thống chọn lọc những sản phẩm có liên quan đến từ khóa  Hiển thị những sản phẩm được chọn lọc |
| **Ngoại lệ** | Nếu không có sản phẩm liên quan đến từ khóa , hệ thống không hiển thị sản phẩm nào |

***Bảng 2. 4 Kịch bản Usecase tìm kiếm sản phẩm***

***Kịch bản Usecase đặt hàng***

|  |  |
| --- | --- |
| Tên use case | Quản lý đặt hàng |
| Tên Actor | Khách hàng |
| Mức | 1 |
| Tiền điền kiện |  |
| Đảm bảo tối thiểu | Khách hàng lựa chọn mặt hàng muốn đặt |
| Đảm bảo thành công |  |
| Kích hoạt | Khách hàng yêu cầu chức năng đặt hàng |
| **Hành động tác nhân** | **Phản ứng hệ thống** |
| Khách hàng yêu cầu chức đặt hàng | Hệ thống sẽ ghi nhận số lượng, mặt hàng của cửa hàng mà khách hàng đã thêm vào giỏ hàng.  Hệ thống sẽ hiển thị đầy đủ thông tin về sản phẩm. |
| **Ngoại Lệ** | Khách hàng chỉ có thể chọn thanh toán khi đã đăng nhập. |

***Bảng 2.5 Kịch bản Usecase đặt hàng***

***Kịch bản Usecase quản lý giỏ hàng***

|  |  |
| --- | --- |
| Tên use case | Quản lý giỏ hàng |
| Tên Actor | Khách hàng |
| Mức | 1 |
| Tiền điền kiện |  |
| Đảm bảo tối thiểu |  |
| Đảm bảo thành công |  |
| Kích hoạt | Khách hàng yêu cầu chức năng quản lý giỏ hàng |
| **Hành động tác nhân** | **Phản ứng hệ thống** |
| Khách hàng yêu cầu chức năng xem giỏ hàng | Hệ thống chuyển đến màn hình quản lý giỏ hàng |
| Khách hàng yêu cầu tiếp chức năng làm mới giỏ hàng | Dữ liệu cũ sẽ được xóa đi , giỏ hàng sẽ được làm mới |
| Khách hàng yêu cầu chức năng thanh toán | Hệ thống sẽ chuyển đến màn hình thanh toán |
| **Ngoại Lệ** | Khách hàng chỉ có thể chọn thanh toán khi đã đăng nhập. |

***Bảng 2.6 Kịch bản Usecase quản lí giỏ hàng***

***Kịch bản Usecase Thanh toán***

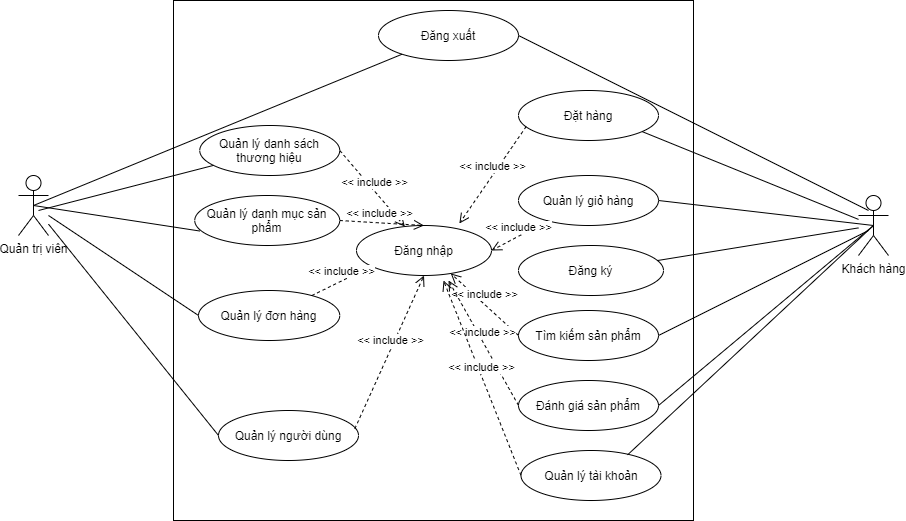
|  |  |
| --- | --- |
| Tên use case | Thanh toán |
| Tên Actor | Khách hàng |
| Mức | 1 |
| Tiền điền kiện | Khách hàng phải đăng nhập ứng dụng |
| Đảm bảo tối thiểu | Khách hàn chọn thanh toán trong giỏ hàng |
| Đảm bảo thành công | Giỏ hàng được chọn thanh toán phải có sản phẩm. |
| Kích hoạt | Khách hàng yêu cầu chức năng thanh toán |
| **Hành động tác nhân** | **Phản ứng hệ thống** |
| Khách hàng yêu cầu chức năng thanh toán | Hệ thống chuyển tới màn hình thanh toán để thực hiện chức năng thanh toán.  Hệ thống đưa ra màn hình cho người dùng đầy đủ các thông tin về bản thân, địa chỉ , số điện thoại lấy từ thông tin người dùng đã đăng nhập.  Hệ thống đưa ra màn hình các thông tin chi tiết về các sản phẩm , số lượng giá thành của từng sản phẩm trong giỏ hàng của người dùng.  Hệ thống đưa ra hình thức thành toán cho khách hàng và hiển thị hóa đơn tổng tiền cho khách hàng |
| Khách hàng yêu cầu chức năng thay đổi địa thông tin nhận hàng | Hệ thống sẽ chuyển đến màn hình thay đổi thông tin địa chỉ. Tại đây khách hàng có thể thay đổi thông tin của mình như tên, Số điện thoại nhận hàng, địa chỉ nhận hàng. |
| Khách hàng yêu cầu chức năng thêm sản phẩm. | Hệ thống sẽ chuyển đến màn hình của cửa hàng để lựa chọn sản phẩm thêm vào giỏ. |
| Khách hàng yêu cầu chức năng xóa sản phẩm. | Hệ thống sẽ cập nhật lại thông tin của đơn hàng và tính toán lại tiền. |
| Khách hàng yêu cầu chức năng thanh toán | Đơn hàng của khách hàng sẽ được chuyển đến server để xử lý,sau khi thành công sẽ chuyển sang màn hình hiển thị trạng thái của đơn hàng. |
| **Ngoại lệ** |  |
| Thanh toán | Sever xử lý đơn hàng nếu thất bại Hệ thống sẽ báo lại cho khách hàng để khách hàng thực hiện lại. |

***Bảng 2.7 Kịch bản Usecase thanh toán***

### 2.6. Biểu đồ Usecase

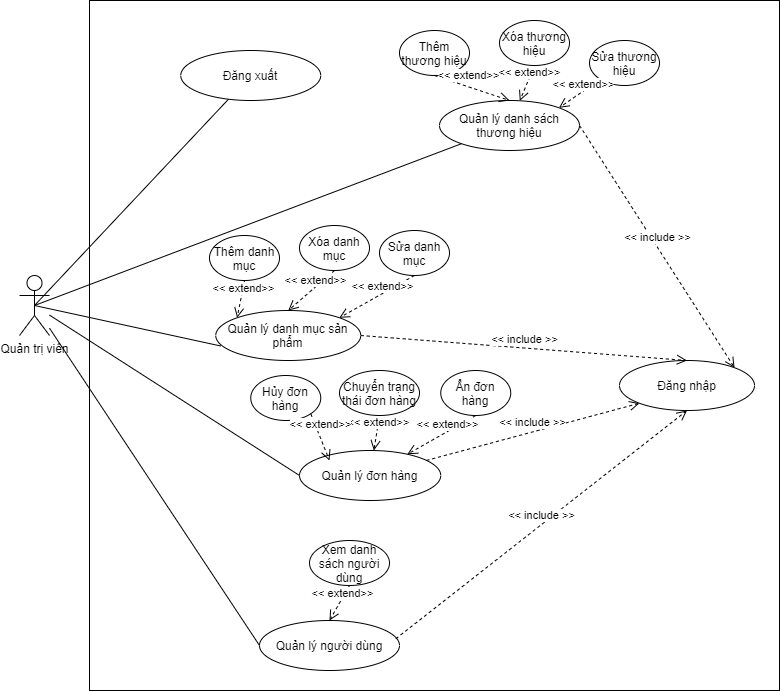
Biểu đồ use case biểu diễn các chức năng của hệ thống bao gồm một tập hợp các tác nhân, các use case và mối quan hệ giữa chúng. Mỗi use case mô tả một chức năng hệ thống cần phải có xét từ quan điểm người sử dụng.

#### 2.6.1. Biểu đồ Use case tổng quát



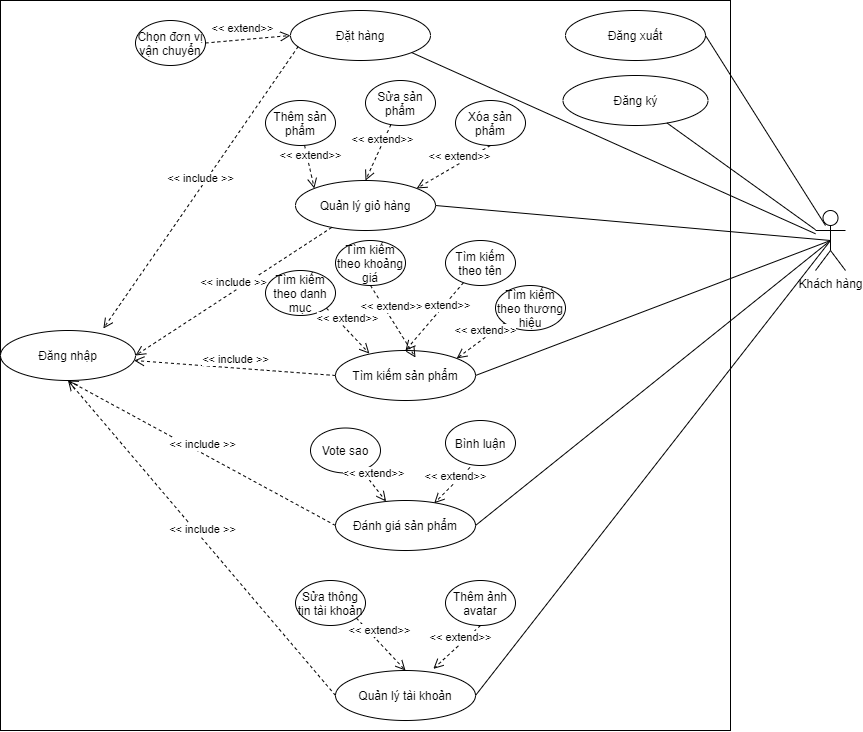
Hình 4 Biểu đồ Use case tổng quát

#### 2.6.2. Biểu đồ Use case đối với quản trị viên



Hình 5 Biểu đồ Use case đối với quản trị viên

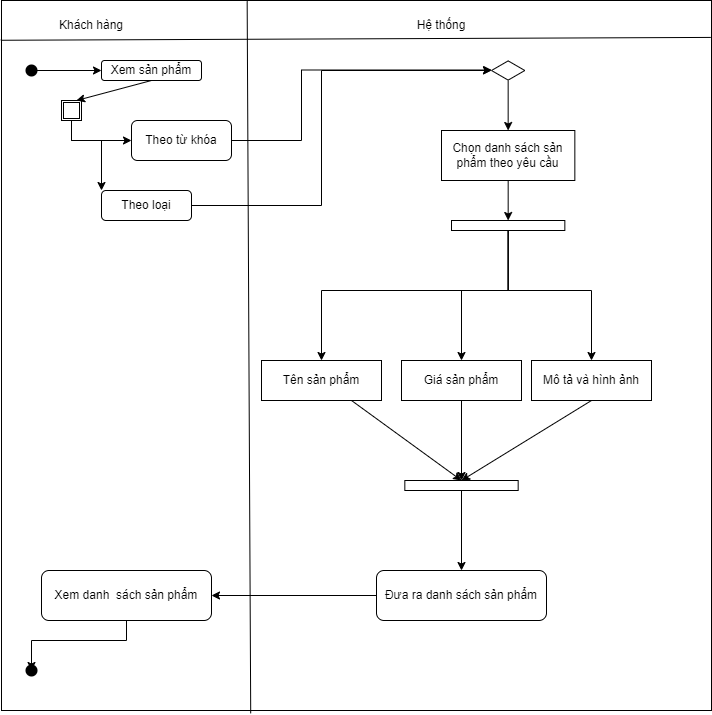
#### 2.6.3. Biểu đồ Use case đối với Người dùng



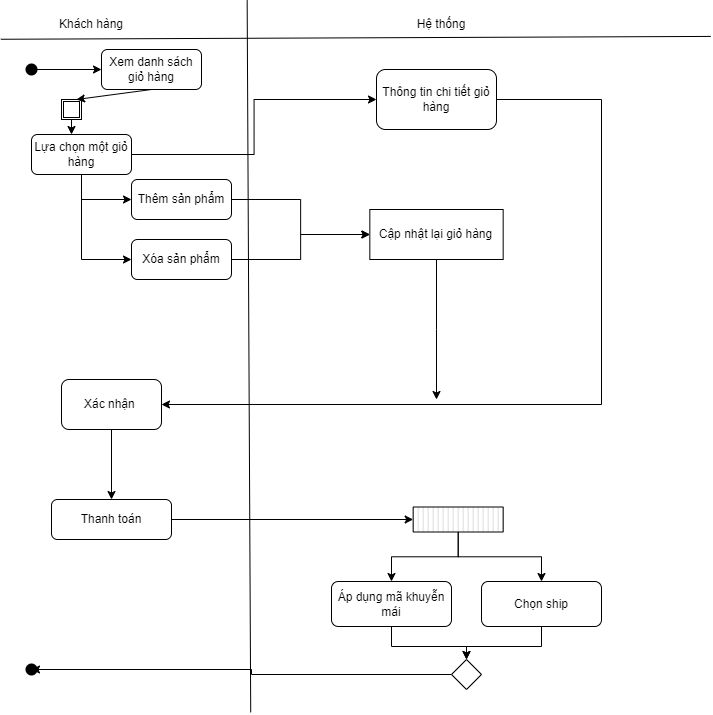
Hình 6 Biểu đồ Use case đối với khách hàng

## 3. Biểu đồ hoạt động Activity

**Xem sản phẩm**



**Quản lý giỏ hàng và thanh toán**

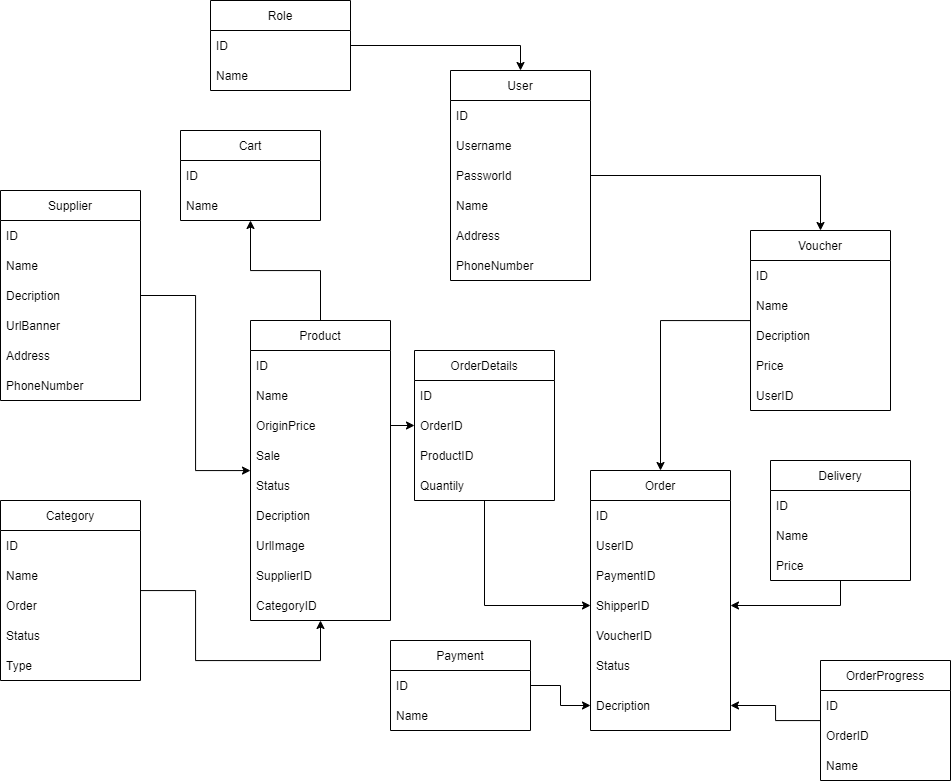


## 4. Xây dựng biểu đồ lớp (Class Diagram)

### 4.1. Xác định các lớp

* User (tài khoản) có các thuộc tính: ID, Username, Password, Name, Gender, Email, Date of birth, Address, Phone number, Avatar, RoleID
* Products (Sản phẩm) có các thuộc tính: ID, Code, Name, Origin price, Sale, Status, Slug, Description, ImageUrl, SupplierID,CategoryID
* Cart (giỏ hàng) có các thuộc tính: ID, User ID, Product ID, Quantity, Status
* Order (đơn hàng) có các thuộc tính: ID, User ID, Payment method ID, Delivery method ID, Order Progress ID, Into money, Transport fee, Status, Description
* Category (Danh mục) có các thuộc tính: ID, Name,
* Supplier (Nhà cung cấp) có các thuộc tính: ID, Name, Description
* Delivery method (Đơn vị vận chuyển) có các thuộc tính: ID, Name, Money
* Order detail (Chi tiết đơn hàng) có các thuộc tính: ID, Order ID, Product ID, Quantity, Total money,status
* Payment method (Phương thức thanh toán) có các thuộc tính: ID, Name
* Voucher (Phiếu giảm giá) có các thuộc tính: ID, Name, Sale, Slug, User ID, Condition
* Role Có các thuộc tính: ID,Name

### 4.2. Bản vẽ Class diagram

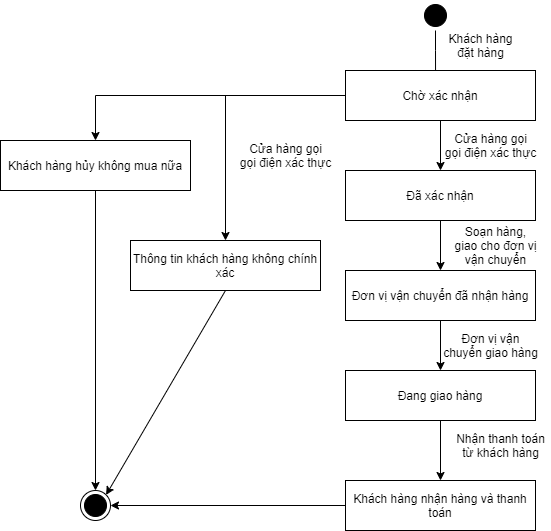


Hình 7 Bản vẽ Class diagram

## 5. Biểu đồ trạng thái

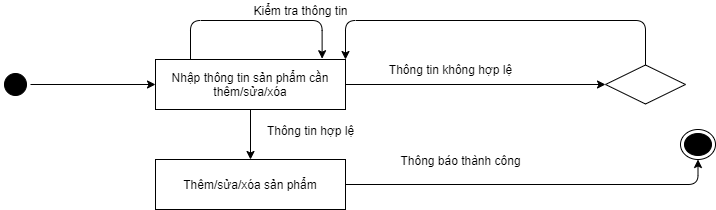
Biểu đồ trạng thái được sử dụng để mô tả các trạng thái và sự chuyển tiếp giữa các trạng thái khi có các sự kiện tác động tới một đối tượng.

### 5.1. Đơn hàng



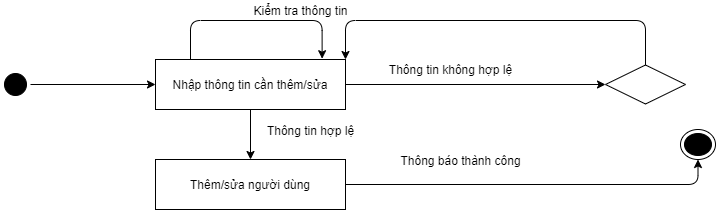
Hình 8 Biểu đồ trạng thái đơn hàng

### 5.2. Sản phẩm



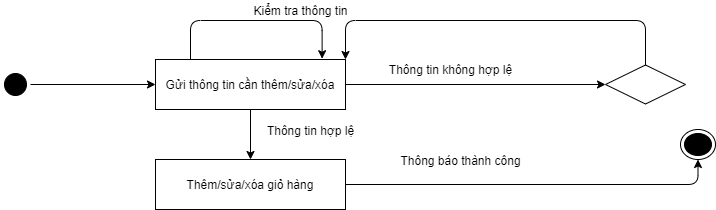
Hình 9 Biểu đồ trạng thái sản phẩm

### 5.3. Người dùng



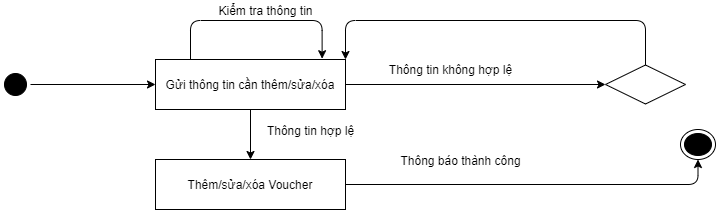
Hình 10 Biểu đồ trạng thái người dùng

### 5.4. Giỏ hàng



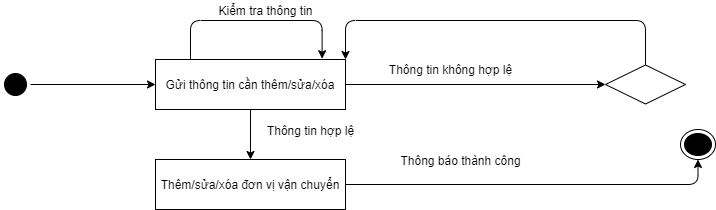
Hình 11 Biểu đồ trạng thái giỏ hàng

### 5.5. Voucher



Hình 13 Biểu đồ trạng thái voucher

### 5.6. Đơn vị vận chuyển



Hình 15 Biểu đồ trạng thái đơn vị vận chuyển

## 5. Thiết kế cơ sở dữ liệu

### 5.1. Xác định các bảng trong CSDL

#### 5.1.1. Bảng ROLE

- Bảng ROLE lưu danh sách kiểu tài khoản như: user, admin

Bảng 5 Bảng ROLE (vai trò của tài khoản)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|  | Id | int | not null, primary key | Id của role, khóa chính |
|  | name | varchar(20) | not null | Tên của role |

#### 5.1.2. Bảng USER

*-* Bảng USER lưu danh sách người dùng bao gồm thông tin tài khoản và thông tin cá nhân

Bảng 6 Bảng USER (người dùng)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | id | int | not null, primary key | Id của người dùng, khóa chính |
| 2 | roleId | int | not null, foreign key (roleId) references Role(id) | Id của role, khóa ngoại |
| 3 | username | nvarchar(50) | not null | Tên đăng nhập của người dung |
| 4 | password | nvarchar(30) | not null | Mật khẩu đăng nhập của tài khoản |
| 5 | Name | nvarchar(100) | not null | Họ tên người dùng |
| 6 | Phone\_number | char(10) |  | Số điện thoại |
| 7 | Avatar | nvarchar(255) |  | Ảnh đại điện |

#### 5.1.3. Bảng VOUCHER

- Bảng VOUCHER lưu các mã giảm giá

Bảng 7 Bảng VOUCHER (phiếu giảm giá)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|  | id | int | not null, primary key | Id của phiếu |
|  | name | nvarchar(20) | not null | Tên của phiếu |
|  | sale | int | Not null | Giá trị giảm giá |
|  | slug | int | Not null | Từ khóa tìm kiếm |
|  | userId | int | not null | Khách hàng được áp dụng |
|  | condition | int |  | Điều kiện áp dụng |

#### 5.1.4. Bảng PAYMENT\_MEDTHOD

- Bảng PAYMENT\_METHOD lưu danh sách phương thức thanh toán

Bảng 8 Bảng PAYMENT\_MEDTHOD (Phương thức thanh toán)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|  | id | int | not null, primary key | Id của phương thức thanh toán |
|  | name | nvarchar(30) | not null | Tên trạng phương thức thanh toán |

#### 5.1.5. Bảng ORDER\_PROGRESS

- Bảng ORDER\_PROGRESS lưu danh sách tình trạng đơn hàng

Bảng 9 Bảng ORDER\_PROGRESS(Tình trạng đơn hàng)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|  | Id | int | not null, primary key | Id của tình trạng đơn |
|  | Progress | nvarchar | Not null | Tình trạng đơn hàng |

#### 5.1.6. Bảng PRODUCT

- Bảng PRODUCT lưu danh sách các sản phẩm bao gồm các thông tin cơ bản

Bảng 10 Bảng PRODUCT (sản phẩm)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | not null, primary key | Id của sản phẩm, khóa chính |
| 2 | name | nvarchar(Max) | not null | Tên của sản phẩm |
| 3 | Code | nchar(30) | not null | Mã sản phẩm |
| 4 | originPrice | int | not null | Giá của sản phẩm |
| 5 | Sale | int |  | Giá trị khuyến mại (%) |
| 6 | Status | nvarchar(30) |  | Trạng thái của sản phẩm |
| 7 | description | nvarchar(Max) | not null | Mô tả sản phẩm |
| 8 | Img | nvarchar(Max) | not null | Ảnh mô tả |
| 9 | CategoryId | int | Not null, foreign key(categoryId) references Category(id) | Id của danh mục sản phẩm |
| 10 | SupplierId | int | Not null, foreign key(supplierId) references supplier(id) | Id của nhà cung cấp |

#### 5.1.7. Bảng ORDER\_DETAIL

- Bảng ORDER\_DETAIL lưu thông tin chi tiết của đơn hàng

Bảng 11 Bảng ORDER\_DETAIL (chi tiết đơn hàng)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|  | Id | int | Not null, primary key | Id của chi tiết đơn hàng |
|  | orderId | int | Not null, foreign key (orderId) references Order (id) | Id của đơn hàng |
|  | productId | int | Not null, foreign key (orderId) references Order (id) | Id của sản phẩm |
|  | Quantity | int | Not null | Số lương sản phẩm |
|  | totalMoney | money | Not null | Tổng tiền |

#### 5.1.8. Bảng ORDER

- Bảng ORDER lưu danh sách các đơn hàng

Bảng 12 Bảng ORDER (Đơn hàng)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | Not null, primary key | Id của đơn hàng |
| 2 | userId | int | Not null, foreign key(userId) references User(id) | Id của chủ đơn hàng |
| 3 | paymentMethodId | int | Not null, foreign key(paymentMethodId) references PaymentMedthod(id) | Id của phương thức thanh toán |
| 4 | orderProgressId | int | Not null, foreign key(orderProgressId) references OrderProgress (id) | Id của tình trạng đơn hàng |
| 5 | intoMoney | money | Not null | Tổng tiền đơn hàng |
| 6 | transportFee | money | Not null | Phí vận chuyển |
| 7 | Status | int |  | Trạng thái đơn hàng |

#### 5.1.9. Bảng DELIVERY\_METHOD

- Bảng DELIVERY\_METHOD lưu danh sách đơn vị vận chuyển

#### Bảng 13 Bảng DELIVERY\_METHOD(Đơn vị vận chuyển)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | Not null, primary key | Id của đơn vị vận chuyển |
| 2 | Name | nvarchar(50) | Not null | Tên của đơn vị vận chuyển |
| 3 | Price | money | Not null | Phí vận chuyển |

#### 5.1.10. Bảng Supplier

- Bảng Supplier lưu danh sách các Nhà cung cấp

Bảng 14 Bảng SUPPLIER(Nhà cung cấp)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | Not null, primary key | Id |
| 2 | Name | nvarchar(50) | Not null | Tên |
| 3 | Active | bit | Not null | Tình trạng kích hoạt |
| 4 | Description | nvarchar(max) |  | Mô |
| 5 | BannerUrl | nvarchar(50) | Not null | Banner url của nhà cung cấp |
| 6 | Address | nvarchar(50) | Not null | Địa chỉ |
| 7 | PhoneNumber | nvarchar(50) | Not null | Số điện thoại |

#### 5.1.11. Bảng CATEGORY

- Bảng CATEGORY lưu các danh mục sản phẩm

Bảng 15 Bảng CATEGORY (Danh mục)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | Not null, primary key | Id của danh mục |
| 2 | Name | nvarchar(max) | Not null | Tên của danh mục |
| 3 | IconUrl | nvarchar(50) |  | Icon Url |
| 4 | Type | int | Not null | Loại danh mục |
| 5 | Status | int | Not null | Trạng thái |

#### 5.1.12. Bảng CART

- Bảng CART thông tin giỏ hàng của khách hàng

Bảng 16 Bảng CART (Giỏ hàng)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | Not null, primary key |  |
| 2 | userId | int | Not null, foreign key(userId) references User(id) | Id của khách hàng |
| 3 | productId | int | Not null, foreign key(productId) references Product(id) | Id của sản phẩm |
| 4 | Quanity | int | Not null | Số lượng |
| 5 | Status | nvarchar(30) |  | Tình trạng |

#### 5.1.13. Bảng VOUCHER\_OF\_USER

- Bảng VOUCHER\_OF\_USER lưu thông tin voucher của khách hàng

Bảng 17 Bảng VOUCHER\_OF\_USER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | Not null, primary key |  |
| 3 | voucherId | int | Not null, foreign key(voucherId) references Voucher(id) | Id của voucher |
| 4 | userId | int | Not null, foreign key(userId) references User(id) | Id của khách hàng |

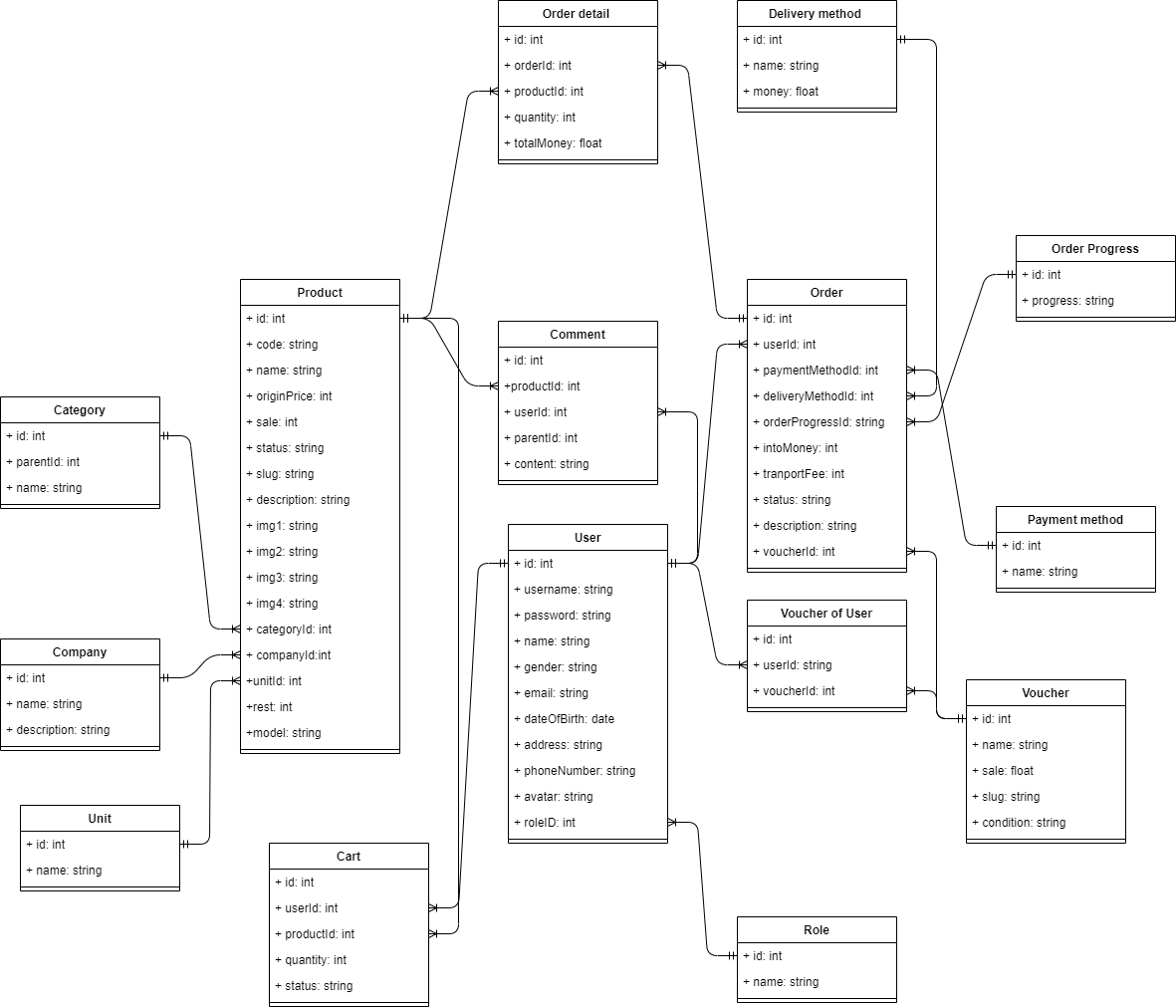
#### 5.1.14. Bảng COMMENT

- Bảng COMMENT lưu thông tin bình luận của khách hàng vể sản phẩm

Bảng 18 Bảng COMMENT (Bình luận)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
| 1 | Id | int | Not null, primary key | Id của bình luận |
| 2 | userId | int | Not null, foreign key(userId) references User(id) | Id của người bình luận |
| 3 | productId | int | Not null, foreign key(productId) references Product(id) | Id của sản phẩm được bình luận |
| 4 | ParentId | int | foreign key(parentId) references Comment (id) | Id của bình luận cha |
| 5 | Content | nvarchar(max) | Not null | Nội dung bình luận |

### 5.2. Mô hình quan hệ



Hình 19 Mô hình quan hệ trong CSDL

CHƯƠNG 3. KẾT QUẢ VÀ ĐÁNH GIÁ

## 1. Ngôn ngữ lập trình và công nghệ sử dụng

Sau khi thu thập yêu cầu và phân tích thiết kế chúng ta sẽ bắt đầu phát triển sản phẩm. Một sản phẩm phần mềm chất lượng cần nắm rõ yêu cầu, đọc kỹ bản thiết kế và chia nhỏ từng phần công việc nếu không dự án sẽ càng ngày càng khó thực hiện, chậm tiến độ, thậm chí sai yêu cầu và phải làm lại từ đầu.

### 1.1. Lựa chọn công nghệ

Phần ứng dụng: **Swift** là ngôn ngữ lập trình mới được phát triển bởi Apple Inc với mục đích hỗ trợ lập trình viên trong việc phát triển các ứng dụng chạy trên các hệ điều hành như iOS, OSX và watchOS.

Phần server được viết bằng Nodejs sử dụng thư viện Express và các API được viết theo chuẩn RESTful API và hệ quản trị cơ sở dữ liệu em dùng là SQL server

Phần quản lý dữ liệu: MongoDB là một cơ sở dữ liệu mã nguồn mở và là cơ sở dữ liệu NoSQL(\*) hàng đầu, được hàng triệu người sử dụng. MongoDB được viết bằng C++.

### 1.2. Swift

Swift là một ngôn ngữ lập trình hướng đối tượng dành cho việc phát triển iOS và macOS, watchOS, tvOS, Linux, và z/OS. được giới thiệu bởi Apple tại hội nghị WWDC 2014. Swift được mong đợi sẽ tồn tại song song cùng Objective-C, ngôn ngữ lập trình hiện tại dành cho các hệ điều hành của Apple. Swift được thiết kế để hoạt động với các framework Cocoa và Cocoa Touch của Apple và phần lớn mã Objective-C hiện có được viết cho các sản phẩm của Apple. Nó được biên dịch với trình biên dịch LLVM và đã được đưa vào Xcode kể từ phiên bản 6, phát hành năm 2014. Trên các nền tảng của Apple, nó sử dụng thư viện runtime Objective-C cho phép mã C, Objective-C, C++ và Swift cùng chạy trong một chương trình.

### 1.3. CocoaPod

### 1.4. ExpressJs

**Expressjs** là một framework được xây dựng trên nền tảng của **Nodejs**. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. **Expressjs** hỗ trợ các method HTTP và midleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.

*Tổng hợp một số chức năng chính của Expressjs như sau:*

* Thiết lập các lớp trung gian để trả về các HTTP request.
* Define router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL.
* Cho phép trả về các trang HTML dựa vào các tham số.

### 1.5. ExpressJs

## 2. Môi trường phát triển

Để phát triển **Ứng dụng mua bán đồ ăn** trong đồ án tốt nghiệp này, em đã sử dụng máy tính của mình là Windows CPU Intel Core i5-9400F, RAM 16Gb, ổ cứng SSD 12Gb, chạy hệ điều hành Windows 10 sử dụng để lập trình server bằng NodeJS và quản lý dữ liệu, macbook i5 2017 để lập trình bằng ngôn ngữ swift Và cài thêm các môi trường phát triển sau:

### 2.1. IDE (Integrated development environment) – Visual Studio Code

* IDE - Môi trường phát triển tích hợp là một ứng dụng phần mềm cung cấp các phương tiện toàn diện cho các lập trình viên máy tính để phát triển phần mềm. Một IDE thường bao gồm ít nhất một trình soạn thảo mã nguồn, xây dựng các công cụ tự động hóa và một trình gỡ lỗi.
* **Visual Studio Code** là một trình soạn thảo mã nguồn do Microsoft tạo cho Windows, Linux và macOS. Các tính năng bao gồm hỗ trợ gỡ lỗi, tô sáng cú pháp, hoàn thành mã thông minh, đoạn mã, tái cấu trúc mã và được nhúng kèm Git. Trong khảo sát dành cho nhà phát triển trên Stack Overflow 2019, Visual Studio Code được xếp hạng là công cụ môi trường dành cho nhà phát triển phổ biến nhất, với 50,7% trong số 87.317 người trả lời rằng họ sử dụng nó.

### 2.2. Source control (Quản lý mã nguồn) – Git

* Git là tên gọi là một hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay (Distributed Version Control System – DVCS). DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository), mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa chính. Và một máy tính khác (nếu họ có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia.
* Lợi ích khi dùng Git:
  + Git dễ sử dụng, an toàn và nhanh chóng.
  + Quản lý source code dễ dàng chuyên nghiệp
  + Có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều bằng việc kết hợp các phân nhánh (branch).
  + Hạn chế được lỗi xảy ra trong quá trình code trong 1 team
  + Khi gặp lỗi có thể dễ dàng Backup lại phiên bản trước
  + Code không giới hạn khoảng cách giữa các thành viên trong team, bạn có thể làm việc ở bất cứ đâu vì chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa.
  + Dễ dàng trong việc deployment sản phẩm.

### 2.3. Xcode

Xcode là bộ phát triển phần mềm tích hợp được Apple phát triển chạy trên hệ điều hành Mac để các lập trình viên có thể phát triển phần mềm chạy trên hệ điều hành Mac và iOS. Phiên bản đầu tiên của Xcode được phát hành vào năm 2003 và phiên bản ổn định hiện tại là 12.5.1 được phát hành vào năm ngày 21 tháng 6 năm 2021. Xcode được phát hành miễn phí cho người dùng Mac download thông qua chợ ứng dụng App Store.

### 2.4. Nodejs

NodeJS được phát triển từ Javascript vào năm 2009 bởi Ryan Dahl. NodeJS có cách thức hoạt động chủ yếu trên Server sử dụng để xây dựng cho các ứng dụng realtime. NodeJS dùng mô hình I/O lập trình dựa theo sự kiện non-blocking.

Chính vì vậy, NodeJS tương đối gọn nhẹ, hiệu quả và là một công cụ hoàn hảo dành cho mọi ứng dụng chuyên sâu về dữ liệu dựa theo khoảng thời gian thực khi chạy trên những thiết bị phân tán. NodeJs thường xuyên được dùng cho mục đích xây dựng một số ứng dụng như:  Ad Server, Websocket server, Fast File Upload Client, RESTful API, Cloud Services, Any Real-time Data Application.

### MongoDB

**MongoDB** là một database hướng tài liệu (document), một dạng NoSQL database. Vì thế, MongoDB sẽ tránh cấu trúc table-based của relational database để thích ứng với các tài liệu như JSON có một schema rất linh hoạt gọi là BSON. MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ các các kích cỡ và các document khác nhau. Các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

**MongoDB** lần đầu ra đời bởi MongoDB Inc., tại thời điểm đó là thế hệ 10, vào tháng Mười năm 2007, nó là một phần của sản phẩm PaaS (Platform as a Service) tương tự như Windows Azure và Google App Engine. Sau đó nó đã được chuyển thành nguồn mở từ năm 2009.

Cài Đặt

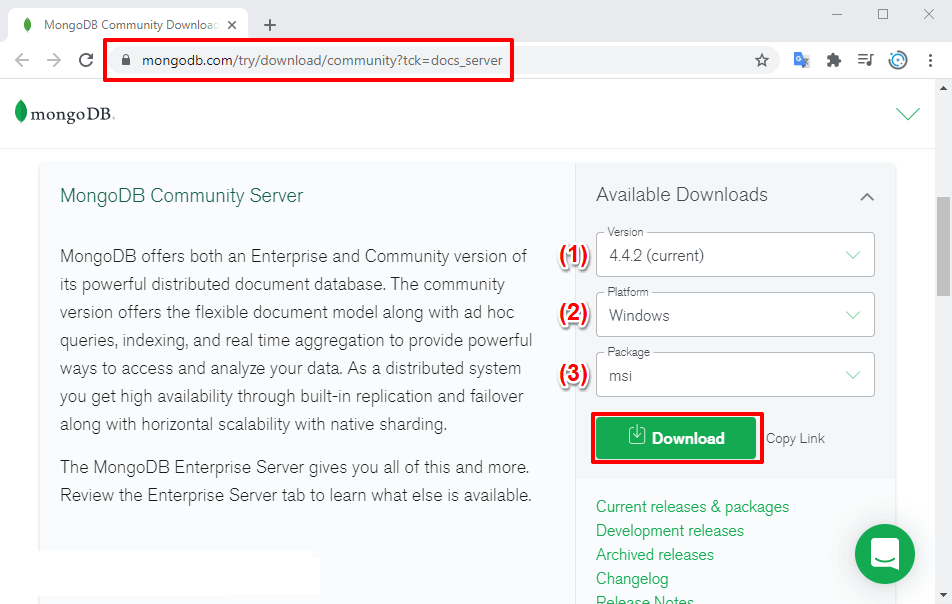
**Bước 1**: Đầu tiên các bạn vào đường dẫn sau để download file cài đặt của MongoDB tại đây:

https://www.mongodb.com/try/download/community?tck=docs\_server

(1) Là phiên bản hiện tại của Mongodb, phiên bản cộng đồng (Free).

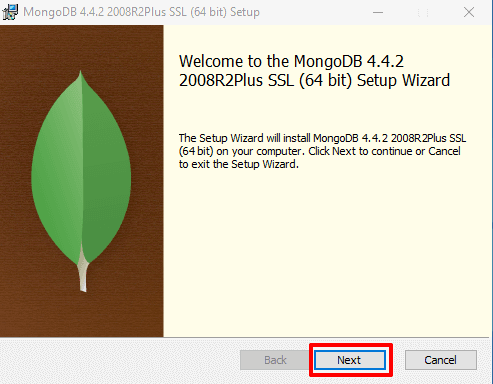
(2) Hệ điều hành (ở đây chúng ta chọn Windows).

(3) Loại file cài đặt (ở đây chúng ta chọn file \*.msi).

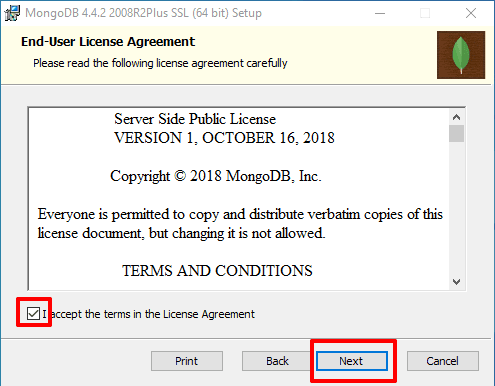


**Bước 2**: Các bạn có thể bấm vào biểu tượng mũi tên bên cạnh để chọn các option phù hợp => rồi bấm vào nút Download để tải file cài đặt về.

Sau khi tải file cài đặt về, các bạn nháy đúp chuột vào file để bắt đầu quá trình cài đặt => và bấm Next để chuyển sang bước tiếp theo.



**Bước 3:** Ở bước này các bạn tích vào I accept the terms in the License Agreement => sau đó bấm vào Next để sang bước tiếp theo.

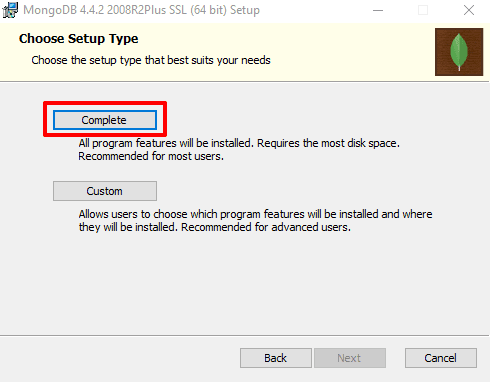


**Bước 4**: Ở đây sẽ có hai option đó là hoặc bạn bấm vào Complete hoặc Custom (Tùy chọn).

Nếu bạn chọn Complete thì tất cả các chức năng sẽ được cài đặt và tất nhiên là sẽ chiếm nhiều dung lượng ổ cứng hơn.

Còn nếu bạn chọn Custom thì bạn sẽ chỉ chọn những chức năng bạn muốn cài đặt và tùy thuộc vào chức năng bạn chọn dung lượng ổ cứng cài đặt sẽ giảm đi.

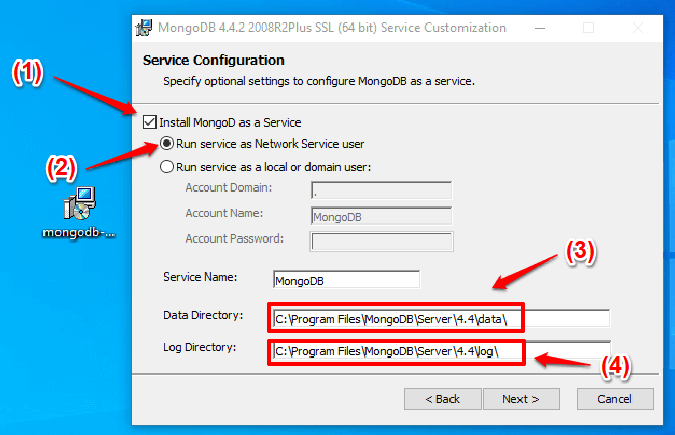
=> Chọn Complete.



**Bước 5:** Tiếp đến bước cấu hình service.

* (1) Cài đặt MongoDB như là một service. Nó giống như những service bình thường của hệ điều hành. Bạn có thể bật/tắt được.
* (2) Chạy service với vai trò là người sử dụng mạng dịch vụ.
* (3) Nơi lưu trữ Database.
* (4) Nơi chứa file log (các server chạy đều có file log, các bạn có thể xem log file xem server đã chạy các service nào, gặp lỗi ở đâu…)

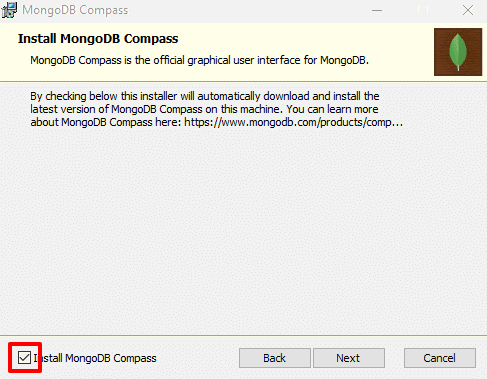
=> Sau đó các bạn bấm Next để tiếp tục.



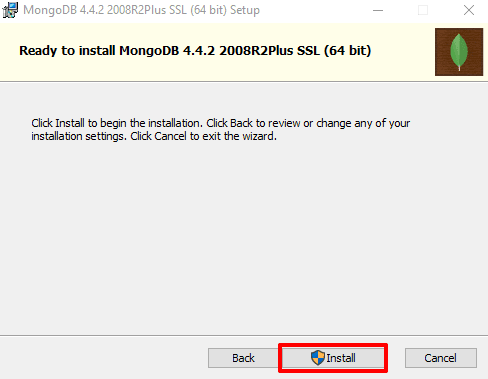
**Bước 6:** Bước tiếp theo này thì các bạn sẽ lựa chọn là có cài MongoDB Compass hay không.

Cho bạn nào chưa biết thì MongoDB Compass là một công cụ, chính xác hơn là một giao diện người dùng giúp cho việc thao tác với dữ liệu trong Mongodb trực quan hơn.

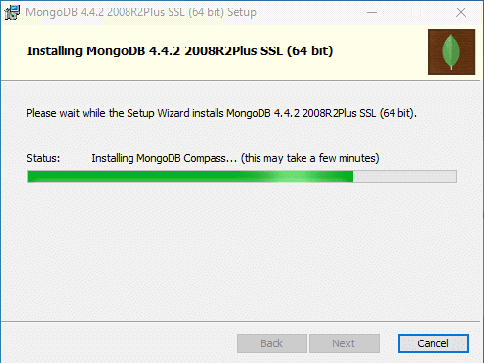
=> Bạn tích và ô Install MongoDB Compass để cài đặt => và bấm Next để đến bước cài đặt.



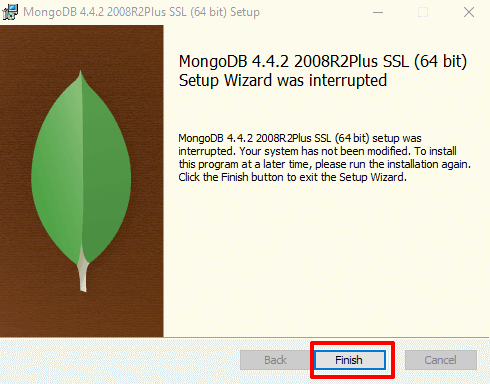
**Bước 7:** Đến bước này thì quá trình cấu hình đã xong và bắt đầu cài đặt. Các bạn bấm vào Install để bắt đầu quá trình cài đặt.



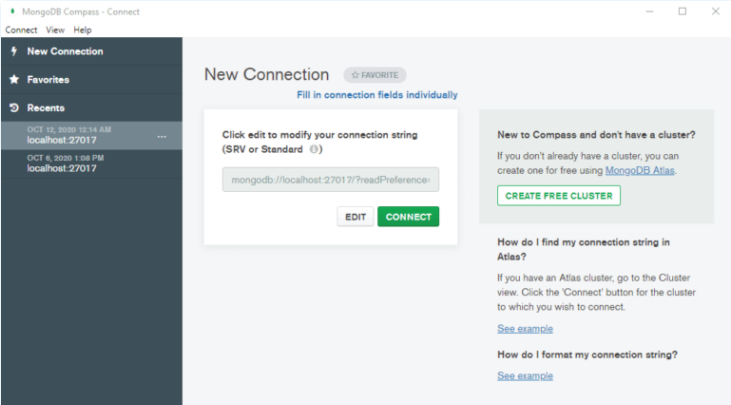
Quá trình cài đặt đang diễn ra, nhanh thì có thể 2-3 phút là xong, chậm có thể mất 5-7 phút tùy thuộc vào kết nối mạng và cấu hình máy nữa.



**Bước 8:** Sau khi cài đặt xong các bạn bấm vào Finish để hoàn tất quá trình cài đặt.



Sau khi cài đặt các bạn chạy Mongodb Compass để vào diện như bên dưới. Đây chính là giao diện chúng ta sẽ làm việc với mongodb.

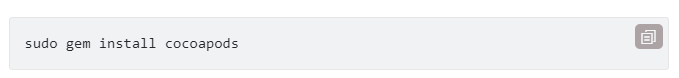


### 2.6. CocoaPods

CocoaPods là người quản lý phụ thuộc cho các dự án Cacao Swift và Objective-C. Nó có hơn 87 nghìn thư viện và được sử dụng trong hơn 3 triệu ứng dụng. CocoaPods có thể giúp bạn mở rộng quy mô dự án của mình một cách thanh lịch.

Cài đặt:

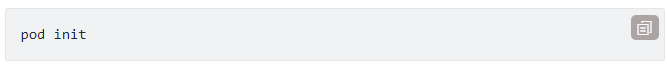
CocoaPods được xây dựng bằng Ruby và có thể cài đặt bằng Ruby mặc định có sẵn trên macOS. Trên OSX, mở terminal và chạy đoạn mã:



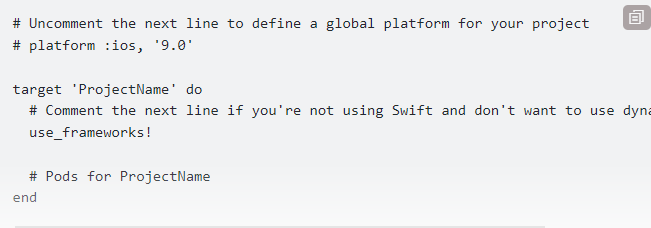
Sau đó Enter => Nhập mật khẩu của laptop => Enter để tiến hành cài đặt.

Tạo Podfile

Tiếp theo, chúng ta tạo mới một project, mở terminal của project . Tiếp theo chạy câu lệnh:



để khởi tạo Podfile. Sau khi tạo xong Podfile, chúng ta sẽ có một file với tên là "Podfile" có chứa nội dung:



Trong file này chúng ta sẽ tiến hành thêm các lib vào.

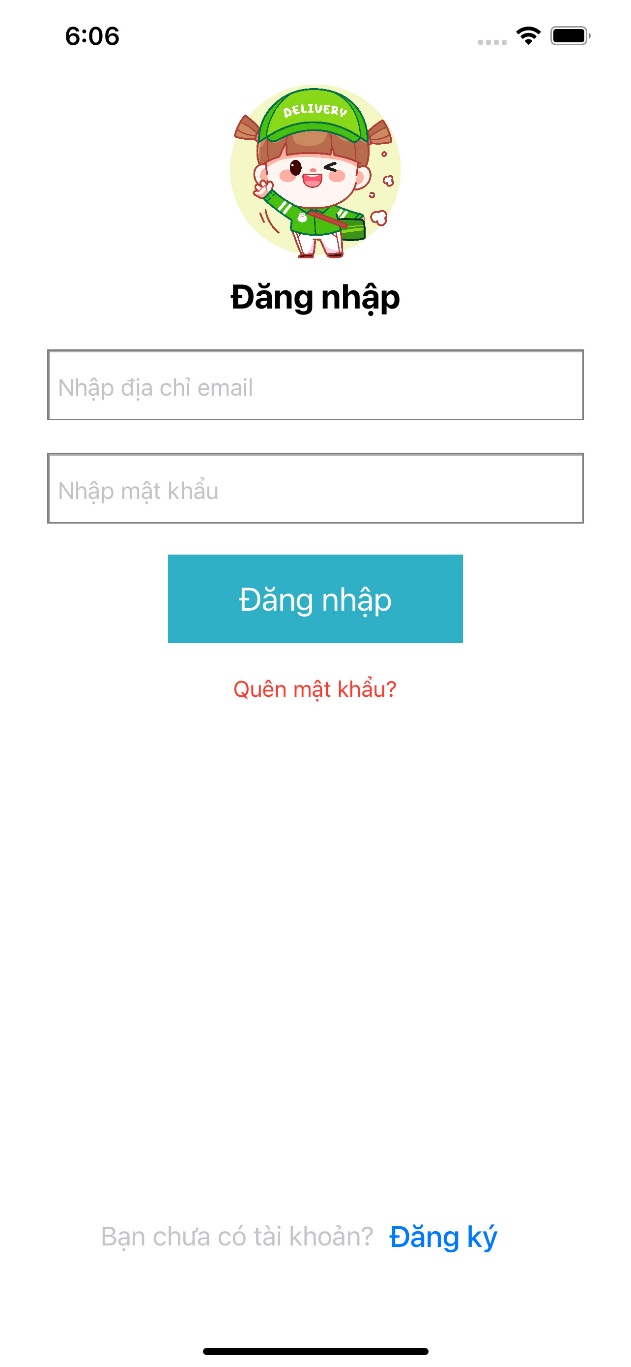
Cài đặt và cập nhật

Để tiến hành cài đặt các thư viện mà chúng ta thêm vào, trên terminal chúng ta tiếp tục gõ:



## 3. Giao diện Ứng dụng cho người dùng

### 3.1. Màn hình đăng nhập



* Màn hình đăng nhập bao gồm:
  + Các trường nhập dữ liệu: Email, Mật khẩu.
  + Nút “Đăng nhập”.
  + Nút “Quên mật khẩu”.
  + Nút “Đăng ký”. Để người dùng chuyển đến trang đăng ký
  + Xác thực khi người dùng để trống.

### 3.2. Màn hình đăng ký



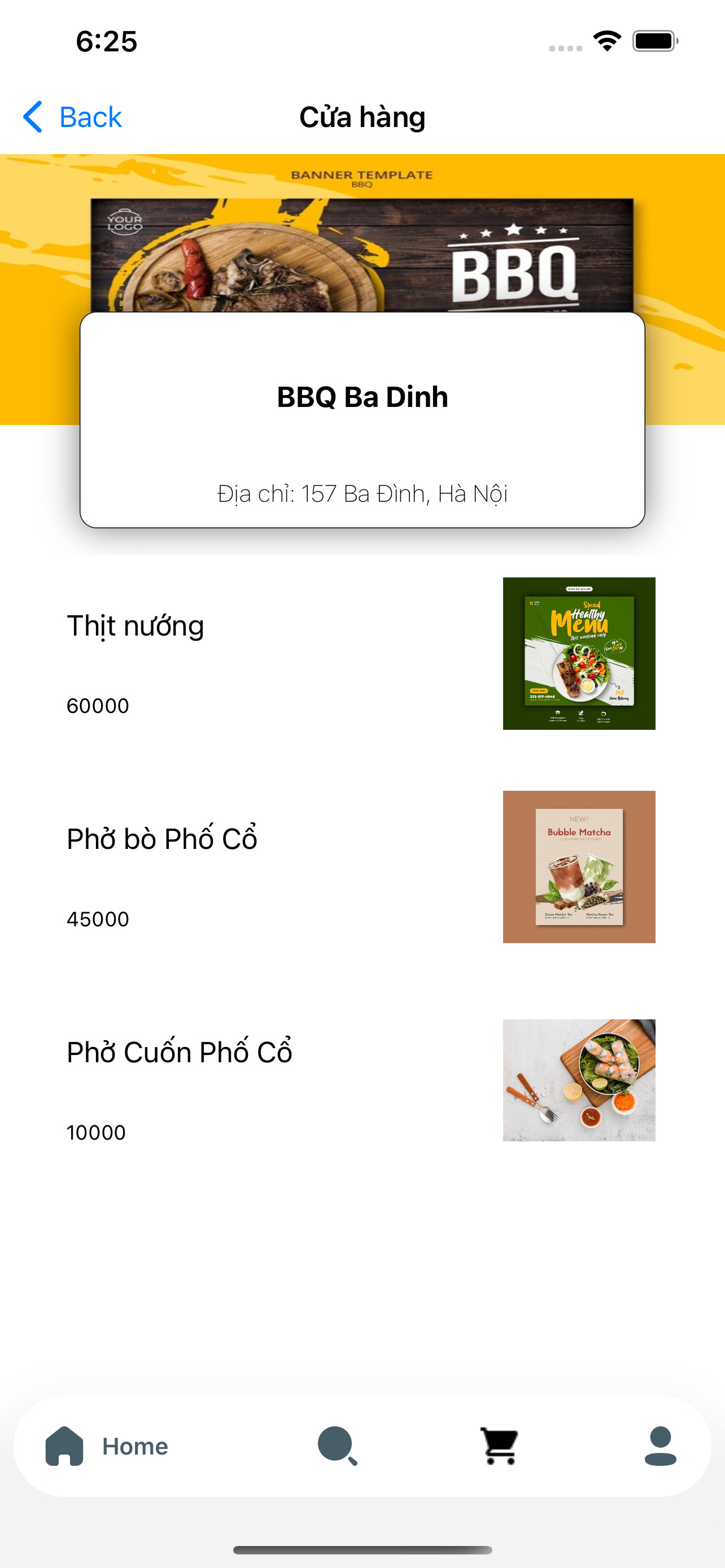
* Màn hình đăng ký tài khoản gồm các trường:
  + Các trường nhập dữ liệu: Email, Họ và tên, Mật khẩu, Nhập lại mật khẩu, Địa chỉ , Số điện thoại
  + Nút “Đăng ký”
  + Xác thực khi người dùng để trống
  + Nút “Đăng nhập” Để chuyển về trang đăng nhập khi người dùng đã có tài khoản và muốn đăng nhập.

### 3.3. Màn hình chính Của ứng dụng

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\410d543615a6dff886b7.jpg

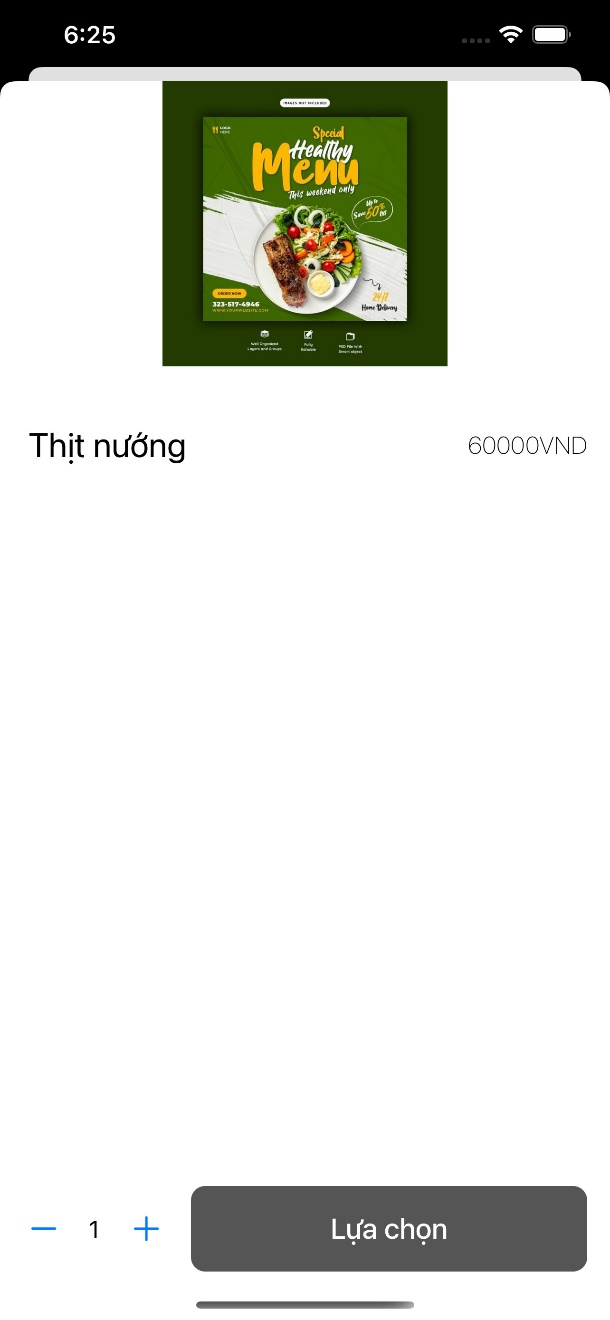
* Màn hình chính của ứng dụng gồm các trường:
  + Phần Banner ở đây sẽ để các banner quảng cáo khuyễn mãi của các cửa hàng
  + Phần Danh mục sản phẩm:
  + Mục các sản phẩm thông dụng
  + Mục sản phẩm thông thường
  + Thanh Tapbar điều hướng tới các màn Tìm kiếm, giỏ hàng,trang cá nhân

### 3.4. Màn hình của cửa hàng



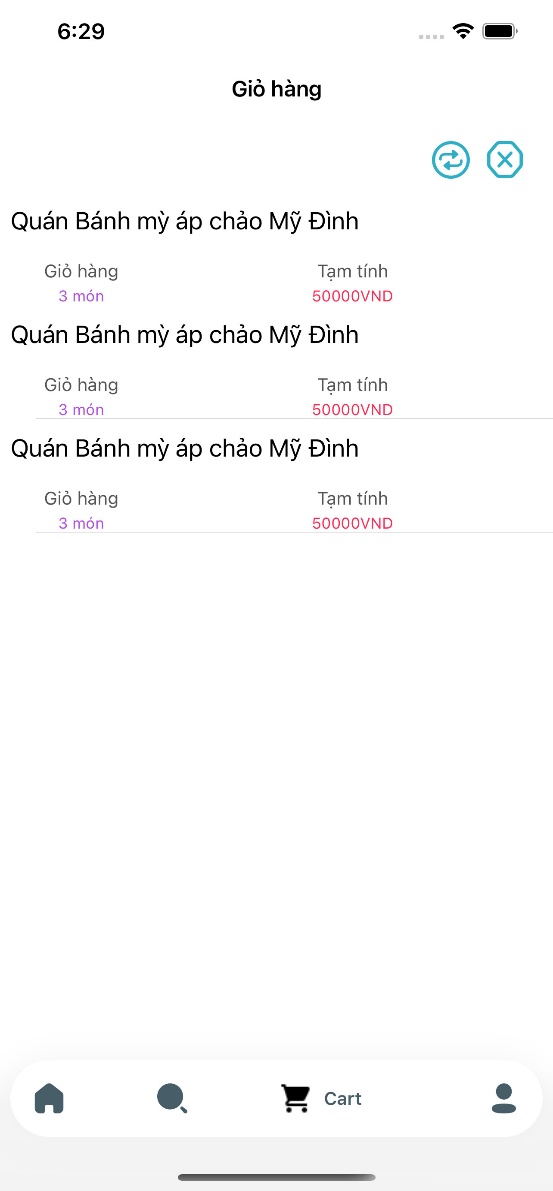
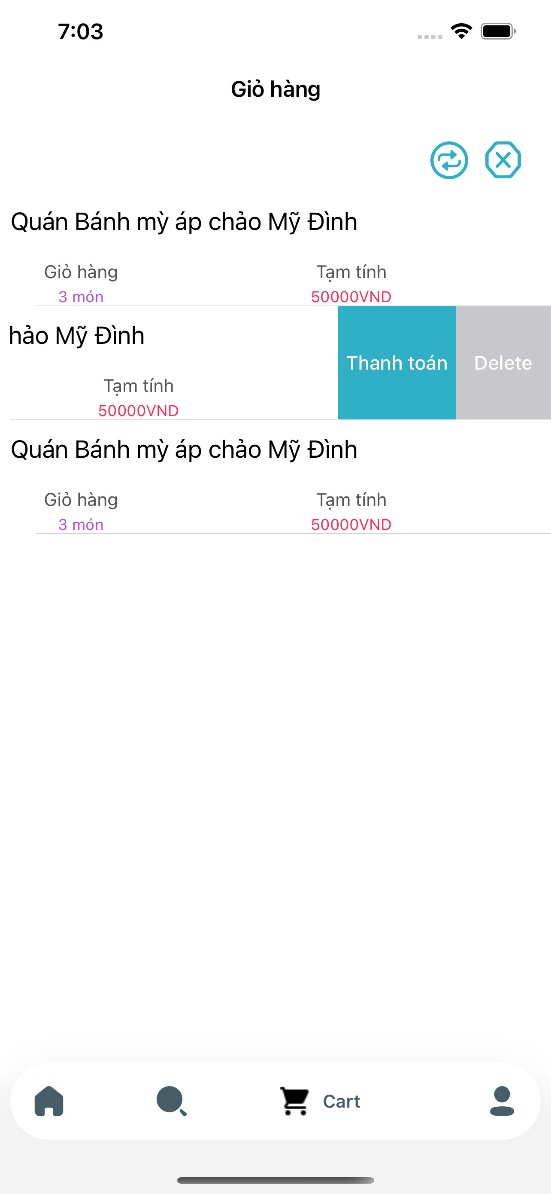
* Màn hình của cửa hàng gồm các trường:
  + Thôn tin về cửa hàng
  + Phần Danh sách sản phẩm:
  + Lựa chọn vài sản phẩm sẽ chuyển đến màn hình mua sản phẩm

### 3.5. Màn hình hiển thị khi người dùng nhấp vào một sản phẩm



* Màn hình của sản phẩm gồm các trường:
  + Thôn tin chi tiết về sản phẩm
  + Chọn số lượng sản phẩm
  + Nút “Lựa chọn” sẽ lưu lại sản phẩm vào giỏ hàng và chuyển đến trang thanh toán.

### 3.6. Màn hình trang giỏ hàng

* Màn hình của trang giỏ hàng gồm các trường:
  + Thông tin chi tiết về đơn hàng ,lượng sản phẩm, thành tiền.
  + Có thể lựa chọn đơn hàng và thực hiện thanh toán hoặc xóa đơn hàng đó đi

### 3.7. Màn hình thanh toán

### 4407bb29fab930e769a8

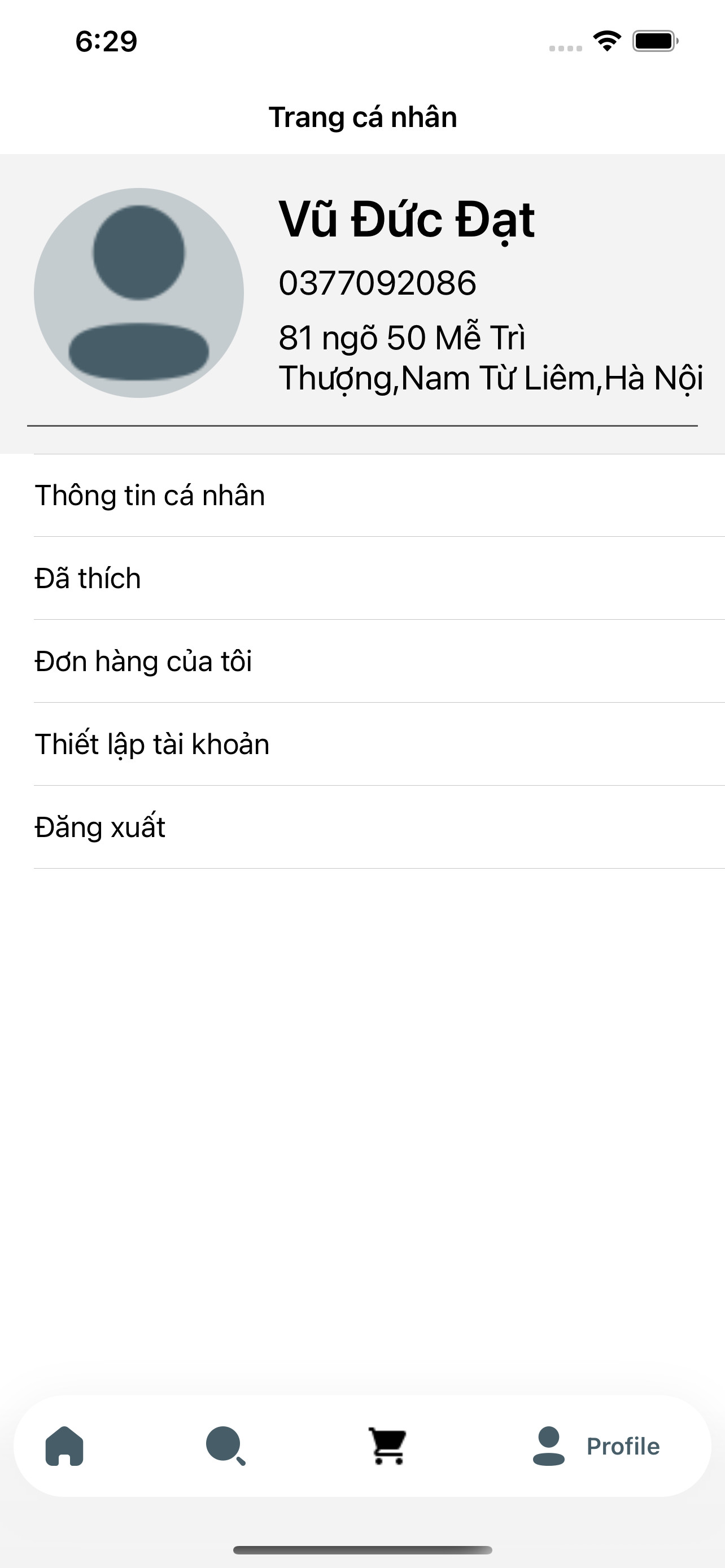
* Màn hình của trang thanh toán của đơn hàng:
  + Thông tin chi tiết về đơn hàng ,lượng sản phẩm, thành tiền.
  + Thay đổi địa chỉ nhận hàng.
  + Chọn phương thức thanh toán.
  + Chọn ship.
  + Thêm món từ cửa hàng.
  + Nút “Thanh toán” Thực hiện xác thực đơn hàng và chuyển sang màn hình trạng thái đơn hàng.

### 3.8. Màn hình thanh toán.



* Màn hình của trạng thái của đơn hàng:
  + Thông tin chi tiết về đơn hàng ,lượng sản phẩm, thành tiền.
  + Hủy đơn khi đơn chưa xác nhận.
  + Trạng thái của đơn hàng.

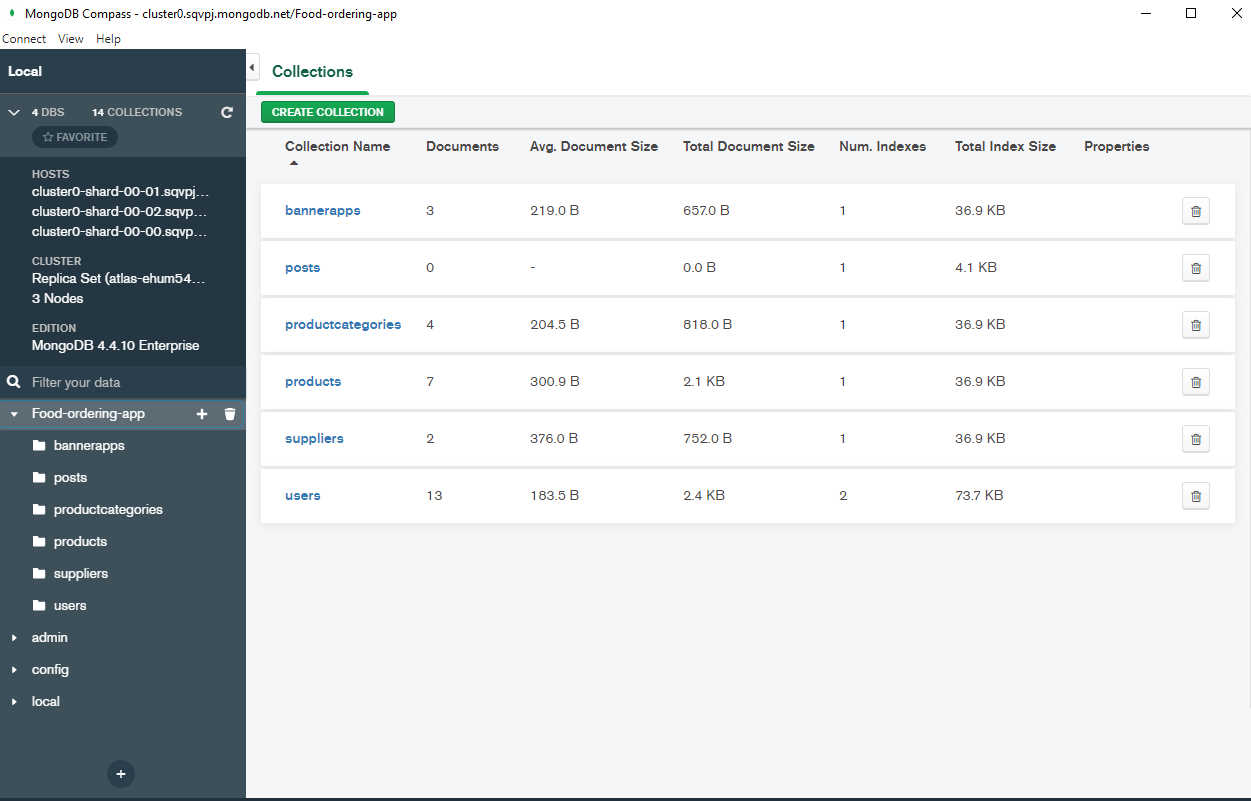
### 3.9. Màn hình thông tin cá nhân.



* Màn hình của trang cá nhân:
  + Thông tin của tài khoản
  + Thay đổi thông tin.
  + Các quán đã thích.
  + Xem danh sách đơn hàng
  + Thiết lập tài khoản
  + Đăng xuất.

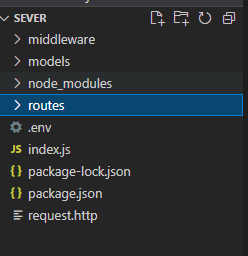
## 4. Server

Server quản lý dữ liệu trực tiếp bằng MongoDB:



## Source Code

* LinkGit: <https://github.com/Vuducdat1997/OrderingFood.git>
  1. **Source Code Server(NodeJS)**

****

Index.js

require('dotenv').config()

const express = require('express')

const mongoose = require('mongoose')

    // const auth = require('registry-auth-token')

const authRouter = require('./routes/auth')

const postRouter = require('./routes/post')

const supplierRouter = require('./routes/supplier')

const productRouter = require('./routes/product')

const categoryRouter = require('./routes/productCategory')

const bannerRouter = require('./routes/banner')

const connectDB = async() => {

    try {

        await mongoose.connect(`mongodb+srv://${process.env.DB\_USERNAME}:${process.env.DB\_PASSWORD}@cluster0.sqvpj.mongodb.net/Food-ordering-app?retryWrites=true&w=majority`, {

            useNewUrlParser: true,

            useUnifiedTopology: true

        })

        console.log('MongoDB connected')

    } catch (error) {

        console.log(error)

        process.exit(1)

    }

}

connectDB()

const app = express()

app.use(express.json())

app.use('/api/auth', authRouter)

app.use('/api/posts', postRouter)

app.use('/api/supplier', supplierRouter)

app.use('/api/product', productRouter)

app.use('/api/category', categoryRouter)

app.use('/api/banner', bannerRouter)

const PORT = 5000

app.listen(PORT, () => console.log(`Server started on port ${PORT}`))

auth.js

const express = require('express')

const router = express.Router()

const argon2 = require('argon2')

const User = require('../models/User')

const jwt = require('jsonwebtoken')

//@route POST api/auth/register

//@desc Register user

// @access public

router.post('/register', async(req, res) => {

    const { username, password, address, phoneNumber } = req.body

    //simple validation

    if (!username || !password)

        return res

            .status(400)

            .json({ success: false, message: 'Missing username or password' })

    try {

        //check for existing user

        const user = await User.findOne({ username })

        if (user)

            return res.status(400).json({

                success: false,

                message: 'Username already taken'

            })

        //all good

        const hashedPassword = await argon2.hash(password)

        const newUser = new User({ username, password: hashedPassword, address, phoneNumber })

        await newUser.save()

        //Return token

        const accessToken = jwt.sign({ userId: newUser.\_id },

            process.env.ACCESS\_TOKEN\_SECRET

        )

        res.json({ success: true, message: 'User created successly!', accessToken })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: 'Internal server error!' })

    }

});

//@route POST api/auth/login

//@desc Login user

// @access public

router.post('/login', async(req, res) => {

    const { username, password } = req.body

        //simple validation

    if (!username || !password)

        return res

            .status(400)

            .json({ success: false, message: 'Missing username or password' })

    try {

        //check for existing user

        const user = await User.findOne({ username })

        if (!user)

            return res.status(400).json({ success: false, message: 'Incorrect username or password' })

        //username found

        const passwordValid = await argon2.verify(user.password, password)

        if (!passwordValid)

            return res.status(400).json({ success: false, message: 'Incorrect username or password' })

        //all good

        //Return token

        const accessToken = jwt.sign({ userId: user.\_id },

            process.env.ACCESS\_TOKEN\_SECRET

        )

        res.json({ success: true, message: 'User logged in successly!', accessToken })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: 'Internal server error!' })

    }

})

module.exports = router

product.js

const express = require('express')

const router = express.Router()

const verifyToken = require('../middleware/auth')

const Product = require('../models/Product')

//@route Get api/product

//@desc Get product

//@access public

router.get('/', async(req, res) => {

    try {

        const product = await Product.find({})

        res.json({ success: true, product: product })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: "Interval server error" })

    }

})

//@route Get api/product/popular

//@desc Get popular

//@access public

router.get('/popular', async(req, res) => {

    try {

        const product = await Product.find({ type: "2" })

        res.json({ success: true, product: product })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: "Interval server error" })

    }

})

// //@route Get api/supplier

// //@desc Get supplier By id

// //@access Private

router.post('/getProduct', async(req, res) => {

    try {

        const { id } = req.body

        const product = await Product.findById(id)

        res.json({ success: true, product: product })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: "Interval server error" })

    }

})

//@route POST api/product

//@desc Create product

//@access Private

router.post('/', async(req, res) => {

    const { name, title, price, url, type, order, supplierId, categoryId } = req.body

    //simple validation

    if (!title || !name || !price || !url || !type || !order || !supplierId)

        return res.status(400).json({ success: false, message: 'Value is require' })

    try {

        const newProduct = new Product({

            name,

            title,

            price,

            url,

            type,

            order,

            supplierId,

            categoryId

        })

        await newProduct.save()

        res.json({ success: true, message: 'Save Success!', product: newProduct })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: 'Internal server error!' })

    }

})

module.exports = router

supplier.js

const express = require('express')

const router = express.Router()

const verifyToken = require('../middleware/auth')

const Supplier = require('../models/Supplier')

const Product = require('../models/Product')

    //@route Get api/supplier

    //@desc Get supplier

    //@access Private

router.get('/', async(req, res) => {

    try {

        const supplier = await Supplier.find({})

        res.json({ success: true, suppliers: supplier })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: "Interval server error" })

    }

})

//@route Get api/supplier

//@desc Get supplier By id

//@access Private

router.post('/getSupplier', async(req, res) => {

    try {

        const { id } = req.body

        const supplier = await Supplier.findById(id)

        const product = await Product.find({ supplierId: id })

        res.json({ success: true, supplier: supplier, product: product })

    } catch (error) {

        console.log(error)

        res.status(500).json({ success: false, message: "Interval server error" })

    }

})

//@route POST api/supplier

//@desc Create supplier

//@access Private

router.post('/', async(req, res) => {

    const { name, title, address, contact, bannerUrl, iconUrl } = req.body

    //simple validation

    if (!title || !name || !address || !contact || !bannerUrl || !iconUrl)

        return res.status(400).json({ success: false, message: 'Value is require' })

    try {

        const newSupplier = new Supplier({

            name,

            title,

            address,

            address,

            contact,

            bannerUrl,

            iconUrl

        })

        await newSupplier.save()

        res.json({ success: true, message: 'Save Success!', supplier: newSupplier })

    } catch (error) {

        console.log(error)

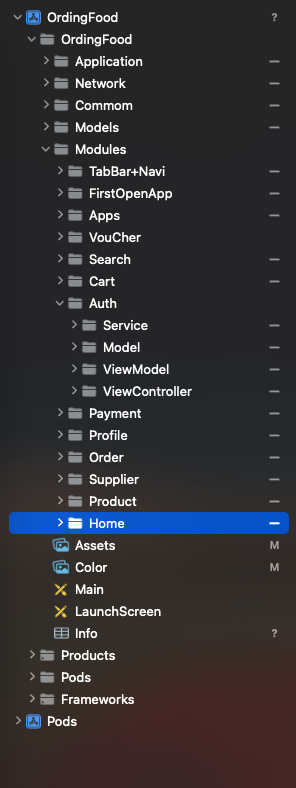
        res.status(500).json({ success: false, message: 'Internal server error!' })

    }

})

module.exports = router

* 1. **Source Code Ứng dụng**



***AuthAPI.swift***

import UIKit

import Moya

enum AuthAPI {

case login(cre: UserCredential)

case register(cre: RegisterCredential)

}

extension AuthAPI: TargetType {

var baseURL: URL {

guard let url = URL(string: "http://192.168.0.106:5000") else { fatalError() }

return url

}

var path: String {

switch self {

case .login:

return "/api/auth/login"

case .register:

return "api/auth/register"

}

}

var method: Moya.Method {

switch self {

case .login:

return .post

case .register:

return .post

}

}

var sampleData: Data {

Data()

}

var task: Task {

switch self {

case .login(let cre):

return .requestJSONEncodable(cre)

case .register(let cre):

return .requestJSONEncodable(cre)

}

}

var headers: [String : String]? {

return ["Content-type": "application/json"]

}

}

***AuthService.swift***

import Foundation

import AVFoundation

class AuthService: NetworkService {

func login(user: UserCredential, completion: @escaping (Result<LoginResponse,BaseError>) -> Void) {

client.request(MultiTarget(AuthAPI.login(cre: user)), type: LoginResponse.self) { result in

completion(result)

}

}

func register(register: RegisterCredential, completion: @escaping (Result<LoginResponse,BaseError>) -> Void) {

client.request(MultiTarget(AuthAPI.register(cre: register)), type: LoginResponse.self) { result in

completion(result)

}

}

}

***AuthViewModel.swift***

import Foundation

class AuthViewModel {

var authService = AuthService()

var cre : UserCredential

init(cre : UserCredential) {

self.cre = cre

}

var responIsLoading: ((Bool) -> ())?

var responseSuccess: ((String) -> ())?

var responseFail: ((String) -> ())?

func login(completion: @escaping (LoginResponse?) -> ()) {

responIsLoading?(true)

authService.login(user: cre) { [weak self] result in

print("Datvd \(result)")

switch result {

case .success(let respon):

switch respon.success {

case false:

completion(nil)

self?.responseFail?("Đăng nhập thất bại")

case true:

completion(respon)

self?.responseSuccess?("Đăng nhập thành công")

default:

completion(nil)

self?.responseFail?("Đăng nhập thất bại")

}

case .failure:

completion(nil)

self?.responseFail?("Đăng nhập thất bại")

}

self?.responIsLoading?(false)

}

}

}

***UserCredential.swift***

import Foundation

struct UserCredential: Codable {

var username: String?

var password: String?

enum Codingkeys: String, CodingKey {

case username,password

}

}

***LoginResponse.swift***

import Foundation

struct LoginResponse: Codable {

var success: Bool?

var message: String?

var accessToken: String?

enum CodingKeys: String, CodingKey {

case success, message, accessToken

}

}

***LoginViewController.swift***

import UIKit

class LoginViewController: UIViewController {

@IBOutlet weak var emailTextField: UITextField!

@IBOutlet weak var passwordTextField: UITextField!

var authViewModel: AuthViewModel!

override func viewDidLoad() {

super.viewDidLoad()

}

@IBAction func loginClick(\_ sender: Any) {

print(#function)

let email = emailTextField.text ?? "dat123"

let pass = passwordTextField.text ?? "123456"

authViewModel = AuthViewModel(cre: UserCredential(username: email, password: pass))

authViewModel.login { [self] respon in

self.authViewModel.responseFail = { string in

//Hien that bai

let alert = UIAlertController(title: "Thông báo", message: "Sai tài khoản hoặc mật khẩu", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Huỷ", style: .default, handler: nil))

navigationController?.present(alert, animated: true)

print(string)

}

self.authViewModel.responIsLoading = { bool in

//Show animation

print(bool)

}

guard let respon = respon else { return }

self.authViewModel.responseSuccess = { string in

//Save access Token

guard let accessToken = respon.accessToken else {return}

UserDefaults.standard.set(accessToken, forKey: UserDefaults.Keys.ACCESS\_TOKEN)

//Check IsLogin

UserDefaults.standard.set(true, forKey: UserDefaults.Keys.IS\_LOGIN\_APP)

//OpenTabbar

openTabbarView()

}

}

}

@IBAction func forgetPassword(\_ sender: Any) {

print(#function)

}

@IBAction func skipClick(\_ sender: Any){

UserDefaults.standard.register(defaults: [UserDefaults.Keys.ACCOUNT\_LOGIN : []])

openTabbarView()

}

func openTabbarView() {

let vc = TabBarController() as UIViewController

vc.navigationItem.hidesBackButton = true

navigationController?.pushViewController(vc, animated: true)

}

@IBAction func registerClick(\_ sender: Any) {

print(#function)

pushToNibViewController(RegisterViewController.self)

}

}

## Kết luận

**1. Kết quả đạt được của đồ án**

- Trong thời gian thực hiện đồ án em đã tìm hiểu về Swift và NodeJs từ đó sử dụng kết hợp với các kiến thức nền tảng cùng với sự trợ giúp của thầy Nguyễn Văn Nam em đã viết ra được Ứng dụng mua bán đồ ăn

- Ứng dụng đã có những chức năng cơ bản như:

+ Đăng nhập, đăng ký, tìm kiếm sản phẩm, quản lý giỏ hàng, mua hàng, quản lý đơn hàng, đánh giá sản phẩm

-Thông qua Ứng dụng, người tiêu dùng sẽ dễ dàng hơn trong việc đặt mua các thực phẩm thức từ cửa hàng mà không phải đến tận cửa hàng,cửa hàng sẽ có thêm nguồn khách hàng tiềm năng, tăng khả năng tiếp cận của các sản phẩm từ cửa hàng tới người tiêu dùng.

**2. Những hạn chế**

- Ứng vẫn còn hạn chế về giao diện

- Trong khoảng thời gian ngắn để thực hiện đề tài, em mới chỉ xây dựng được những chức năng cơ bản của Ứng dụng

**3. Hướng phát triển**

- Chỉnh sửa về giao diện để đẹp hơn.

- Thêm các chức năng, animation để giúp người dùng không bị nhàm chán khi sử dụng ứng dụng.

- Thêm chức năng thanh toán online

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | "Swift 5.5,". [Online]. Available: https://docs.swift.org/swift-book/. |
| [2] | " Swift Knowledge Base " [Online]. Available: https://www.hackingwithswift.com/example-code. |
| [3] | "api," 2021. [Online]. Available: https://expressjs.com/. |
| [4] | "MongoDB Doc,". [Online]. Available: https://docs.mongodb.com/. |