

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



KHÓA LUẬN TỐT NGHIỆP
TÌM HIỂU MÔ HÌNH EVENT DRIVEN VÀ ỨNG DỤNG XÂY DỰNG
WEBISTE BÁN HÀNG

Giảng viên hướng dẫn: **ThS. PHẠM MINH ĐƯƠNG**

Sinh viên thực hiện: **THẠCH MINH LỰC**

Mã số sinh viên: **110119025**

Lớp: **DA19TTA**

Khoá: **2019 - 2023**

Trà Vinh, tháng 2 năm 2023

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



KHÓA LUẬN TỐT NGHIỆP
TÌM HIỂU MÔ HÌNH EVENT DRIVEN VÀ ỨNG DỤNG XÂY DỰNG
WEBISTE BÁN HÀNG

Giảng viên hướng dẫn: **ThS. PHẠM MINH ĐƯƠNG**

Sinh viên thực hiện: **THẠCH MINH LỰC**

Mã số sinh viên: **110119025**

Lớp: **DA19TTA**

Khoá: **2019 - 2023**

Trà Vinh, tháng 2 năm 2023

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

MỤC LỤC

CHƯƠNG 1: ĐẶT VẤN ĐỀ	1
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu và phạm vi nghiên cứu.....	2
1.2.1. Mục tiêu nghiên cứu.....	2
1.2.2. Phạm vi nghiên cứu.....	2
1.3. Ý nghĩa khoa học và thực tiễn.....	3
1.3.1. Ý nghĩa khoa học	3
1.3.2. Ý nghĩa thực tiễn.....	3
CHƯƠNG 2: CƠ SỞ LÝ LUẬN.....	4
2.1. Tổng quan về mô hình kiến trúc hướng sự kiện	4
2.1.1. Kiến trúc hướng sự kiện là gì ?	4
2.1.2. Sự phát triển của kiến trúc hướng sự kiện	4
2.2. Giới thiệu về Restful API.....	8
2.2.1. Restful API là gì ?.....	8
2.2.2. Các thành phần của Restful API	8
2.2.3. Restful API hoạt động như thế nào	9
2.3. Laravel Framework	9
2.3.1. Laravel là gì ?.....	9
2.3.2. Cơ chế hoạt động của Laravel.....	9
2.3.3. Cấu trúc thư mục của Laravel	10
2.3.4. Databases.....	11
2.3.5. Routing	12
2.3.6. Views.....	12
2.3.7. Controllers.....	13
2.3.8. Middleware	15
2.3.9. Requests	15
2.3.10. Response	16
2.4. VueJS	16
2.4.1. VueJS là gì ?	16

2.4.2. Data:	17
2.4.3. Directives	17
2.4.4. Methods.....	18
2.4.5. Computed	18
2.4.6. Watchers.....	19
2.4.7. Filters.....	19
2.4.8. Props.....	20
2.4.9. Events	20
2.4.10. Slot	20
2.5. Supervisor	21
2.5.1. Supervisor là gì	21
2.5.2. Cài đặt supervisor.....	22
2.5.3. Cấu hình supervisor để giám sát laravel queue.....	22
CHƯƠNG 3: ĐỐI TƯỢNG VÀ PHƯƠNG PHÁP NGHIÊN CỨU.....	23
3.1. Đối tượng nghiên cứu.....	23
3.2. Phương pháp nghiên cứu.....	23
3.3. Đạo đức nghiên cứu	23
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU VÀ THẢO LUẬN.....	24
4.1. Tổng quan về mô hình hàng đợi tin nhắn RabbitMQ	24
4.1.1. AMQP là gì ?	24
4.1.2. Sự tách biệt thực sự giữa bên public message và bên nhận thông tin.....	25
4.1.3. Các loại exchange trong RabbitMQ.....	25
4.1.4. Khi nào vào tại sao nên sử dụng RabbitMQ	29
4.1.5. Hướng dẫn cài đặt và sử dụng RabbitMQ	30
4.2. Giới thiệu các thành phần trong RabbitMQ Management Interface.....	31
4.2.1. Overview	31
4.2.2. Connections.....	33
4.2.3. Channels	33
4.2.4. Exchanges	33
4.2.5. Queue	34
4.2.6. Admin.....	35

4.3. Sử dụng RabbitMQ trên ứng dụng Laravel.....	35
4.4. Mô tả ứng dụng	39
4.5. Yêu cầu hệ thống.....	41
4.5.1. Yêu cầu chức năng	41
4.5.2. Yêu cầu phi chức năng.....	41
4.6. Mô hình CSDL	41
4.7. Mô hình xử lý	54
4.7.1. Usecase khách hàng	54
4.7.2. Usecase quản trị viên	55
4.8. Mô hình tuần tự.....	56
4.9. Mô tả hoạt động	58
4.9.1. Mô tả hoạt động của khách hàng	58
4.9.2. Mô tả hoạt động của quản trị viên.....	59
4.10. Cài đặt và thử nghiệm	60
4.10.1. Thiết kế giao diện.....	60
4.10.2. Hướng dẫn cài đặt	72
CHƯƠNG 5: KẾT LUẬN	73

DANH MỤC CÁC BẢNG

Bảng 4-1. Chi tiết bảng loại khách hàng	42
Bảng 4-2. Chi tiết bảng khách hàng	42
Bảng 4-3. Chi tiết bản thương hiệu	43
Bảng 4-4. Chi tiết bảng danh mục	43
Bảng 4-5. Chi tiết bảng màu sắc	44
Bảng 4-6. Chi tiết bảng kích cỡ	44
Bảng 4-7. Chi tiết bảng mã giảm giá	45
Bảng 4-8. Chi tiết bảng giảm giá	45
Bảng 4-9. Chi tiết bảng sản phẩm	46
Bảng 4-10. Chi tiết bảng màu sắc sản phẩm	47
Bảng 4-11. Mô tả bảng kích cỡ sản phẩm	47
Bảng 4-12. Mô tả bảng biến thể sản phẩm	47
Bảng 4-13. Chi tiết bảng tags	48
Bảng 4-14. Chi tiết bảng liên hệ sản phẩm	48
Bảng 4-15. Chi tiết bảng tag sản phẩm	49
Bảng 4-16. Chi tiết bảng phương thức thanh toán	49
Bảng 4-17. Chi tiết bảng theo dõi đơn hàng	50
Bảng 4-18. Chi tiết bảng đơn hàng	50
Bảng 4-19. Chi tiết bảng chi tiết đơn hàng	51
Bảng 4-20. Chi tiết bảng danh mục bài viết	52
Bảng 4-21. Chi tiết bảng bài viết	52

DANH MỤC HÌNH ẢNH

Hình 2-1. Mô hình kiến trúc hướng sự kiện.....	6
Hình 2-2. Mô tả ứng dụng Laravel hoạt động.....	10
Hình 4-1. Cách hoạt động của producer và consumer	25
Hình 4-2. Cách hoạt động của Direct Exchange	26
Hình 4-3. Cách hoạt động của Fanout Exchange.....	27
Hình 4-4. Ví dụ về RabbitMQ xử lý file PDF	29
Hình 4-5. Giao diện tổng quan RabbitMQ.....	31
Hình 4-6. Giao diện Exchange RabbitMQ.....	34
Hình 4-7. Giao diện Queue RabbitMQ	34
Hình 4-8. Giao diện trang chủ.....	62
Hình 4-9. Giao diện chi tiết sản phẩm.	64
Hình 4-10. Giao diện danh sách sản phẩm.....	65
Hình 4-11. Giao diện đăng ký khách hàng.....	66
Hình 4-12. Giao diện giỏ hàng	67
Hình 4-13. Giao diện thanh toán	68
Hình 4-14. Giao diện quản trị	69
Hình 4-15. Giao diện thêm sản phẩm	70
Hình 4-16. Giao diện nhóm khách hàng	71

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

Tên viết tắt	Giải thích
API	Application Programming Interface
CSDL	Cơ sở dữ liệu
CSRF	Cross-site request forgery
JWT	JSON Web Token
JSON	JavaScript Object Notation
EDA	Event Driven Architecture
REST	Representational State Transfer
NPM	Node Package Manager
ORM	Object Relational Mapping

CHƯƠNG 1: ĐẶT VẤN ĐỀ

1.1. Lý do chọn đề tài

Với sự bùng nổ về các tiện ích và dịch vụ trên internet thì hiện nay xu hướng dịch vụ sẽ được triển khai trên các hệ thống số và các ứng dụng ngày càng gia tăng đáng kể. Sự thay đổi về xu hướng dịch vụ trên internet phát sinh ra yêu cầu bài toán về tắt nghẽn dữ liệu khi hệ thống phải đáp ứng phục vụ cho nhiều người dùng một lúc mà không ảnh hưởng đến chất lượng trải nghiệm của người dùng. Đối với các yêu cầu đó thì có nhiều đề xuất để xây dựng một hệ thống để đáp ứng đủ tải để phục vụ cho người dùng là việc làm hết sức cần thiết trong số đó việc triển khai hệ thống theo hướng sự kiện đang trở thành một mô hình được ưu tiên để giải quyết vấn đề này. Đối với việc triển khai một ứng dụng phần mềm việc triển khai truyền dữ liệu từ nơi này đến nơi khác là phần một phần cơ bản và cần phải đáp ứng. Messaging là một công nghệ quan trọng để kết nối dữ liệu trong môi trường này.

Có rất nhiều giải pháp để đơn giản hóa việc phát triển một ứng dụng và trong số đó đề xuất giải pháp sử dụng mô hình để chuẩn hóa có thể tái sử dụng các luồng xử lý trong hệ thống được rất nhiều hệ thống đang sử dụng là mô hình sự kiện sử dụng hàng đợi để xử lý các sự kiện đưa vào. Trong quá trình phát triển ứng dụng mà sử dụng mô hình này thì nhà phát triển chỉ cần quan tâm đến core logic của họ còn việc định tuyến dữ liệu thì được hệ thống bên thứ 3 chịu trách nhiệm. Ngoài việc xác định và định tuyến hướng đi của dữ liệu thì mô hình còn một số tính năng quan trọng như về khả năng chịu lỗi hệ thống, đảm bảo dữ liệu và khả năng phân tán dữ liệu để xử lý.

Trong mô hình hướng sự kiện các dữ liệu bản ghi trở thành các sự kiện và công việc truyền các bản ghi dữ liệu - bản ghi - từ hệ thống này sang hệ thống khác sẽ được thực hiện thông qua một hệ thống trung gian. Ngược lại với các kết nối trực tiếp, nơi người gửi biết được người nhận và kết nối trực tiếp đến từng người nhận, các giải pháp messaging sẽ tách rời việc gửi dữ liệu khỏi việc xử lý dữ liệu. Người gửi không cần biết người nhận nào sẽ thấy dữ liệu đó hoặc khi nào họ sẽ thấy dữ liệu đó.

Với những lợi thế đó mô hình hướng sự kiện được ứng dụng ngày càng rộng rãi và giúp ích rất nhiều cho các nhà phát triển cũng như xử lý được các yêu cầu bài toán phức tạp mà việc xử lý truyền thống không thể thực hiện được. Đối với các hệ thống cần phải đáp ứng được nhu nhập truy cập của hàng chục người và hàng nghìn người cùng một lúc thì khi các sự kiện phát sinh thì cần phải có hệ thống để mà xử lý. Việc xử lý này rất quan trọng bởi nó ảnh hưởng rất lớn đến toàn bộ hệ thống khi mà các trung tâm trung gian này là nơi để mà tiếp nhận và phân phối các sự kiện đến nơi để mà xử lý.

Hiện nay có khá nhiều phần mềm để quản lý các tin nhắn sự kiện trong số đó có như RabbitMQ, Apache Kafka, ActiveMQ, SQS (AWS), Redis Pub/Sub,... Mỗi loại message broker sẽ có những ưu nhược điểm riêng tùy theo từng nhu cầu mục đích và chi phí phải bỏ ra.

1.2. Mục tiêu và phạm vi nghiên cứu

1.2.1. Mục tiêu nghiên cứu

- Nghiên cứu về mô hình kiến trúc hướng sự kiện
- Tổng quan về hàng đợi tin nhắn RabbitMQ (các khái niệm và mô hình)
- Tìm hiểu Restful API của Laravel, xây dựng được API giao tiếp Vue với lại Laravel
- Tìm hiểu quy trình quản lý nghiệp vụ website bán hàng
- Nghiên cứu cài đặt và phát triển Website bằng Laravel và VueJS

1.2.2. Phạm vi nghiên cứu

- Tìm hiểu về mô hình hướng sự kiện và ứng dụng mô hình hàng đợi tin nhắn để xây dựng hệ thống theo hướng sự kiện.
- Tìm hiểu cách sử dụng và tạo Restful API bằng Laravel Framework trả dữ liệu để VueJS có thể sử dụng.
- Tìm hiểu nguyên lý xây dựng website bán hàng trực tuyến và tích hợp hệ thống thanh toán, vận chuyển, xác thực.

1.3. Ý nghĩa khoa học và thực tiễn

1.3.1. Ý nghĩa khoa học

- Giúp sinh viên hiểu được quy trình hoạt động của công nghệ web. Hiểu được cách thiết kế truy xuất CSDL từ phía backend (Laravel) và giao tiếp với lại Frontend (VueJS).
- Tìm hiểu được cách thiết kế giao diện website từ HTML, CSS, Bootstrap. Nâng cao tư duy xây dựng và phát triển hệ thống.

1.3.2. Ý nghĩa thực tiễn

- Trước sự phát triển ngày càng mạnh mẽ thương mại điện tử thì việc mua sắm ngày càng trở nên phổ biến. Việc ứng dụng mô hình hướng sự kiện và áp dụng mô hình hàng đợi tin nhắn sẽ giúp cho hệ thống thương mại điện tử và một số hệ thống cần đáp ứng lượng xử lý cao có thể đáp ứng được hàng ngàn thậm chí hàng triệu phản hồi đơn hàng từ hệ thống mà không ảnh hưởng đến trải nghiệm chất lượng của người dùng. Do đó việc nghiên cứu mô hình hướng sự kiện sẽ là đề tài góp phần đưa ra ý tưởng hướng tiếp cận mới mà có thể là sẽ giải quyết được các bài toán mà mô hình truyền thống sẽ gặp phải.
- Thông qua đề tài này sẽ làm cơ sở tri thức để sinh viên có thể làm việc tại các công ty phát triển phần mềm.

CHƯƠNG 2: CƠ SỞ LÝ LUẬN

2.1. Tổng quan về mô hình kiến trúc hướng sự kiện

2.1.1. Kiến trúc hướng sự kiện là gì ?

Sự kiện là sự thay đổi trạng thái của một đối tượng khi có các tác nhân hoặc các hoạt động được tác động vào. Một sự kiện phải bao gồm đầu vào là những yêu cầu thay đổi trạng thái của đối tượng. Còn đầu ra chính là trạng thái của đối tượng sau khi được sửa đổi. Một sự kiện có thể trải qua nhiều sự biến đổi để tạo ra được trạng thái đích và sự kiện này cũng có thể là đầu vào của sự kiện khác tạo ra các mắt xích sự kiện đến khi mà đạt được đến trạng thái đích. Trong mô hình hướng sự kiện, đầu vào của sự kiện có thể là dữ liệu thô được đưa vào xử lý sau một loạt quá trình lưu trữ, phân tích, tổng kết thì các sự kiện này sẽ đưa ra được kết quả mà ta yêu cầu. Trong quá trình luân chuyển các sự kiện thì các sự kiện này sẽ được lưu trữ và có thể được dịch chuyển đến một nơi nào khác cần cho quá trình phân tích, các dữ liệu cũng có thể được theo dõi và được quản lý để tránh việc mất mát hay là sai sót dữ liệu.

Kiến trúc hướng sự kiện là một phương pháp thiết kế hệ thống được xây dựng để ghi lại, truyền tải và xử lý các sự kiện thông qua một kiến trúc tách rời. Điều này có nghĩa là các hệ thống không cần biết về nhau để chia sẻ thông tin và hoàn thành nhiệm vụ. Kiến trúc này thay thế kiến trúc “request/response” truyền thống, nơi các dịch vụ sẽ phải đợi phản hồi trước khi chúng có thể chuyển sang tác vụ tiếp theo. Luồng của kiến trúc hướng sự kiện được điều hành bởi các sự kiện và nó được thiết kế để phản hồi chúng hoặc thực hiện một số hành động để phản hồi lại một sự kiện.

2.1.2. Sự phát triển của kiến trúc hướng sự kiện

Trong vài năm qua, xu hướng từ tập trung vào dữ liệu ở trạng thái nghỉ (kiến trúc hướng dịch vụ) sang tập trung vào các sự kiện (kiến trúc hướng sự kiện) trở thành ngày càng phổ biến khi mà lượng truy cập người dùng ngày càng gia tăng.

Đối với các mô hình truyền thống dữ liệu sẽ được tập trung và xử lý ở một chỗ ở đây có thể được gọi là trung tâm dữ liệu khi xử lý các sự kiện xảy ra thứ họ quan tâm nhất chính là sự toàn vẹn và đúng đắn của dữ liệu, đổi lại thứ họ phải đánh đổi chính là

hiệu năng và tốc độ của ứng dụng, các phát sinh về sự cố sẽ diễn ra ngay lập tức sau khi mà sự kiện được đưa vào xử lý và trong quá trình đó có một hoạt động nào đó phát sinh ra lỗi. Trong mô hình hướng sự kiện, các dữ liệu cũng là một phần không thể nào thiếu, nhưng các sự kiện hay các thay đổi là quan trọng nhất, thứ họ quan tâm là sự kiện sẽ được diễn ra như thế nào và sẽ được xử lý như thế nào. Tuy nhiên, trong một ứng dụng phần mềm thì hai mô hình này vẫn được sử dụng cùng nhau cả hai mô hình cùng nhau hỗ trợ để đảm bảo được mục đích của từng ứng dụng.

Một ví dụ đơn giản cho thấy được những lợi ích tuyệt vời của mô hình sự kiện được ứng dụng trong thực tế: điều gì sẽ xảy ra khi mà chúng ta đi vào nhà hàng thường thức món ăn, đối với mô hình hướng dịch vụ các nhà hàng yêu cầu khách hàng phải xếp thành một hàng đợi đến khi nào tới lượt của mình thì sẽ đến quầy phục vụ tiến hành đặt đồ ăn sau đó sẽ tiến hành thanh toán với người phục vụ, sau khi đã thanh toán xong người khách hàng vẫn phải đứng đợi người phục vụ đưa giấy đặt hàng cho bên đầu bếp để chế biến theo đơn hàng. Sau khi các món trong đơn hàng được làm xong thì người phục vụ sẽ đưa đồ ăn đến quầy phục vụ để đưa cho khách hàng. Lúc này khách hàng đã nhận được đồ ăn của mình sẽ tìm đến một bàn ăn nào đó để thưởng thức món ăn. Lúc này người phục vụ sẽ sẵn sàng tiếp nhận khách hàng tiếp theo và các quy trình vẫn sẽ diễn ra như vậy đến khi nào phục vụ đến hết người cuối cùng. Rõ ràng ta thấy các hoạt động này sẽ phải tốn rất nhiều thời gian và chi phí cho mỗi lần phục vụ, ứng dụng mô hình hướng sự kiện, khi mà khách hàng đến họ cũng sẽ được bắt vào đứng đợi ở trong hàng sau đó khi nào mà khách hàng tới lượt của mình thì người khách hàng đó sẽ đến quầy phục vụ tiến hành đặt đồ ăn. Sau khi đã nhận được đơn hàng từ phía khách hàng người phục vụ sẽ đưa thông tin món ăn trong đơn hàng cho bếp, bếp sẽ tiến hành làm món ăn ở cùng thời điểm này người phục vụ sẽ hướng dẫn khách hàng ngồi vào bàn chờ. Lúc này quầy phục vụ đã bị trống đã sẵn sàng nhận đơn hàng từ phía khách hàng tiếp theo, cũng trong quá trình này người phục vụ cũng có thể nhận yêu cầu thanh toán từ phía khách hàng. Mặc dù các hoạt động này diễn ra không liên tục nhưng mà phải đảm bảo được hết tất cả các sự kiện phải xảy ra. Đây cũng chính là ý tưởng của mô hình hàng đợi tin nhắn được nghiên cứu trong cuốn báo cáo này

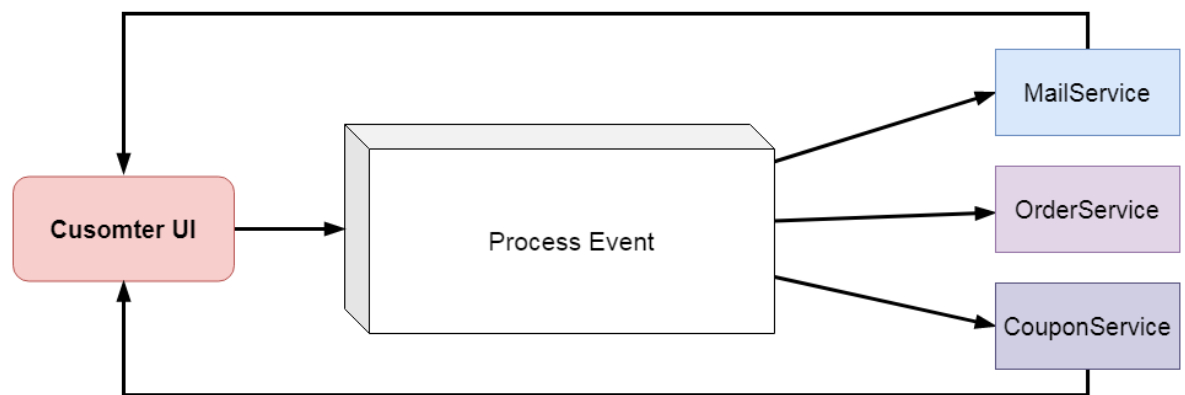
Các sự kiện trong thiết kế hệ thống có những đặc điểm chung dưới đây:

- Chúng là một sự kiện cho thể hiện những sự kiện diễn ra trước đó.
- Các sự kiện này tồn tại bất biến và không thể thay đổi.
- Chúng có thể được lưu trữ vĩnh viễn và có thể truy cập bất cứ khi nào.

Kiến trúc hướng sự kiện là một tin nhắn xác nhận một hành động đã được thực hiện. Thông báo có thể là một xác nhận đơn giản về hành động hoặc chứa dữ liệu bổ sung về sự kiện mà các hệ thống phân sử dụng để áp dụng logic nghiệp vụ của chúng.

Kiến trúc hướng sự kiện hoạt động như thế nào?

Các thành phần của kiến trúc hướng sự kiện có thể bao gồm ba phần: nhà sản xuất (producer), bên nhận thông tin, nhà môi giới (consumer). Consumer có thể là tùy chọn, đặc biệt khi bạn có một producer và một bên nhận thông tin duy nhất đang giao tiếp trực tiếp với nhau và producer chỉ gửi các sự kiện cho bên nhận thông tin. Ta có thể tương tác giữa các đối tượng khác nhau bên trong 1 ứng dụng, giữa 1 đối tượng và đối tượng bên ngoài thông qua sự kiện và đáp ứng sự kiện.



Hình 2-1. Mô hình kiến trúc hướng sự kiện

Các trường hợp sử dụng kiến trúc theo hướng sự kiện

- **Sao chép dữ liệu:** Một sự kiện có thể được chia sẻ cho nhiều dịch vụ khác nhau.
- **Xử lý song song:** Một sự kiện có thể kích hoạt nhiều hoạt động để xử lý cùng một lúc.
- **Giám sát thời gian thực:** Các hệ thống có thể tạo ra các sự kiện cho các thay đổi đối với trạng thái của chúng và ta hoàn toàn có thể giám sát được chúng thông

qua một số dịch vụ để phát hiện ra điểm bất thường từ đó đưa ra các phương hướng giải quyết mới.

- **Khả năng tương tác:** Do bên gửi thông tin và bên nhận thông tin là hoàn toàn tách biệt nên ta hoàn toàn có thể viết bằng nhiều loại ngôn ngữ khác nhau mà không phụ thuộc vào nhau.
- **Dự phòng:** Khi có một dịch vụ trong hệ thống sự cố, các sự kiện có thể được duy trì và định tuyến đến các dịch vụ nằm trong hệ thống.
- **Microservices:** EDA thường được sử dụng trong hệ thống microservices để chia sẻ thông tin giữa các hệ thống.

Lợi ích của kiến trúc theo hướng sự kiện

- **Khả năng chịu lỗi cao:** Việc tách rời bên gửi thông tin và bên tiếp nhận thông tin nên mang lại cho EDA khả năng chịu lỗi cao vì lỗi trong một dịch vụ sẽ không ảnh hưởng đến các dịch vụ còn lại.
- **Lưu vào bộ đệm:** do tính chất không đồng bộ của EDA, không cần đợi các bên tiếp nhận thông tin bắt kịp nếu các sự kiện được tạo với tốc độ khác với tốc độ chúng được tiêu thụ.
- **Khả năng mở rộng:** mỗi dịch vụ có thể được mở rộng một cách độc lập để phù hợp nhất với nhu cầu của doanh nghiệp.
- **Tiết kiệm chi phí:** Việc chuyển các quy trình sang kiến trúc hướng sự kiện loại bỏ nhu cầu của những quy trình lặp đi lặp lại trước đây và thay vào đó sử dụng một luồng dữ liệu liên tục để xử lý khi nó được tạo ra. Cách tiếp cận xử lý này sẽ cho phép doanh nghiệp đạt đến tốc độ phản hồi có thể dự đoán trước và khả năng mở rộng xử lý, làm giảm thiểu chi phí hạ tầng cần phải xử lý định kỳ một lượng lớn dữ liệu. Do đó, điều này làm giảm chi phí vận hành, cải thiện hiệu quả kinh doanh, năng suất và lợi nhuận. EDA cũng tiêu thụ ít băng thông hơn.
- **Nhanh nhẹn:** Các sự kiện được tự động lọc và định tuyến mà không cần tạo mã tùy chỉnh dài dòng.
- **Vị trí tập trung:** Luồng sự kiện hoạt động như một vị trí tập trung, nơi các thay đổi chính sách và kiểm tra có thể được thực hiện.

- **Nâng cao trải nghiệm khách hàng:** EDA là một phương pháp tiếp cận đưa trải nghiệm người dùng trở thành thứ quan trọng nhất mà cũng phải đảm bảo được sự ổn định và chính xác trong quá trình vận hành.

2.2. Giới thiệu về Restful API

2.2.1. Restful API là gì ?

Restful API là một trong những tiêu chuẩn trong quá trình phát triển phần mềm cho phép lập trình viên tạo và quản lý các tài nguyên để giao tiếp từ phía server và phía client. Dữ liệu truyền qua thông qua các giao thức HTTP, HTTPS thường sẽ là các dữ liệu dạng thô, json, binary text.

2.2.2. Các thành phần của Restful API

API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà ta cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.

REST (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE,... đến một URL để xử lý dữ liệu.

RESTful API là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các loại tài nguyên. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile...) khác nhau giao tiếp với nhau. Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE...) và cách định dạng các URL cho ứng dụng web để quản các tài nguyên hệ thống. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE...) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

2.2.3. Restful API hoạt động như thế nào

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

- **GET** (SELECT): Trả về một resource hoặc một danh sách resource.
- **POST** (CREATE): Tạo mới một resource.
- **PUT** (UPDATE): Cập nhật thông tin cho resource.
- **DELETE** (DELETE): Xoá một resource.

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.

2.3. Laravel Framework

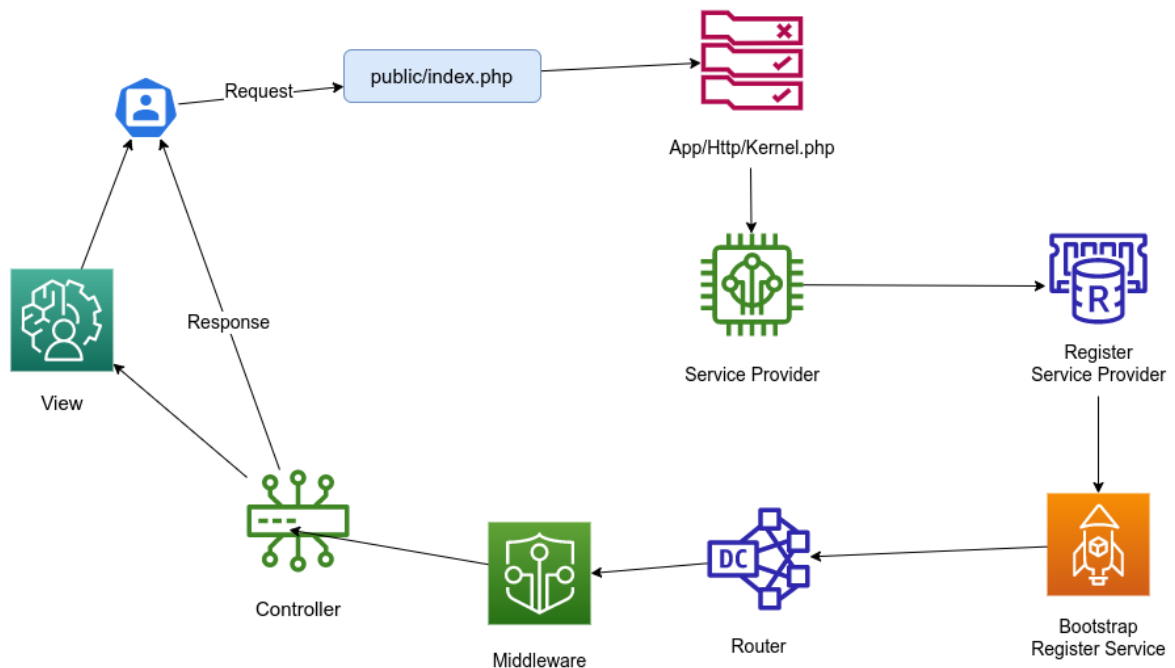
2.3.1. Laravel là gì ?

Laravel là một framework mã nguồn mở miễn phí của PHP, được tạo ra bởi Taylor Otwell (một lập trình viên kỳ cựu chuyển từ .NET sang PHP) giúp xây dựng những ứng dụng theo mô hình MVC (Model - View - Controller). Laravel xây dựng cấu trúc thư mục có tính khoa học cao, các tài liệu hướng dẫn và cộng đồng hỗ trợ và phát triển cực kỳ mạnh mẽ. Đối với người vừa bắt đầu tìm hiểu về Framework thì Laravel chính là một nguồn tài liệu để có thể mà phát triển bản thân theo hướng lập trình website.

2.3.2. Cơ chế hoạt động của Laravel

Khi truy cập một ứng dụng Laravel thì một yêu cầu sẽ được gửi về server. Đầu tiên khi vào trong ứng dụng Laravel thì server sẽ tiến hành tải các service provider đã được đăng ký trước sau đó boot chúng lên. Các yêu cầu sau đó được định tuyến đến đường dẫn như

yêu cầu và các yêu cầu này sẽ được xử lý ở phía bên trong controller. Lúc này controller sẽ đảm nhiệm nhiệm vụ xử lý tương tác với CSDL và trả về kết quả cho người dùng.



Hình 2-2. Mô tả ứng dụng Laravel hoạt động

2.3.3. Cấu trúc thư mục của Laravel

Thư mục	Chức năng
app	Chứa controllers, models, views và assets của ứng dụng.
bootstrap	Chứa một số file của bootstrap framework và một số file cấu hình nạp tự động.
config	Chứa các file cấu hình của hệ thống
database	Chứa file dùng để migrate các bảng và seed dữ liệu.
public	Thư mục public dùng cho việc lưu trữ những file như css, javascripts, file images và những file khác. Nó cũng chứa file khởi động index.php.

resources	Chứa các view, các asset và các file ngôn ngữ.
routes	Thư mục routes chứa các tuyến đường (route) đã định nghĩa của ứng dụng. Mặc định các file: API.php, web.php, channels.php và console.php được kết nối với Laravel.
test	Chứa feature test, unit test
vendor	Chứa toàn bộ code của bên thứ ba. Chứa plugin cài thêm cho ứng dụng và các composer package
storage	Thư mục storage được sử dụng để lưu trữ file tạm thời cho những dịch vụ Laravel khác nhau như session, cache, biên dịch template views. Thư mục này có thể ghi lại bởi web server và được duy trì bởi Laravel mà không cần can thiệp vào
.env	File chứa các cấu hình biến môi trường
artisan	file thực hiện lệnh của Laravel.
composer.json composer.lock composer-setup.php	File của Composer.
package.js	file package.js, chứa các package cần dùng cho projects.
phpunit.xml	file phpunit.xml, xml của phpunit dùng để testing project.
webpack.mix.js	file webpack.mix.js, file dùng để build các webpack.

2.3.4. Databases

Laravel làm cho việc kết nối tới các database và thực thi các query cực kì đơn giản với nhiều database backend thông qua sử dụng raw SQL, query builder, và Eloquent ORM. Hiện tại, Laravel hỗ trợ sẵn bốn database sau:

- MySQL
- Postgres
- SQLite
- SQL Server

Một trong những công cụ để hỗ trợ truy vấn mạnh mẽ của Laravel là Eloquent ORM. Eloquent ORM đi kèm với Laravel cung cấp một API đơn giản và tiện lợi cho giao tiếp với database. Mỗi database table sẽ có một "Model" tương ứng để tương tác với table đó. Model cho phép query dữ liệu trong table, cũng như chèn thêm sửa xóa các dữ liệu mới.

2.3.5. Routing

Tập tin định nghĩa của Laravel được đặt ở thư mục routes mặc định khi tạo dự án, Laravel sẽ tạo ra 4 file route:

- Api.php: Thường để quản lý định tuyến các yêu cầu xử lý liên quan đến api
- Web.php: Thường được dùng để quản lý định tuyến để render ra các view
- Channels.php: Thường dùng để quản lý định tuyến broadcast, socket
- Console.php: Thường dùng để quản lý định tuyến các console

Các phương thức được sử dụng nhiều nhất trong Laravel

- Route::get(url, [controller, method]): Sử dụng để lấy các tài nguyên mà hệ thống trả về.
- Route::post(url, [controller, method]): Sử dụng để thêm một tài nguyên mới.
- Route::put(url, [controller, method]): Sử dụng để thay đổi dữ liệu của một tài nguyên.
- Route::delete(url, [controller, method]): Sử dụng để xóa một tài nguyên.

2.3.6. Views

Views sẽ chứa nội dung định dạng HTML cho phép hiển thị giao diện người dùng. Khi có yêu cầu truy cập từ phía người dùng thì từ controller sẽ xác định được đường dẫn từ đó sẽ hiển thị giao diện đúng với lại đường dẫn được yêu cầu. Laravel sử dụng Blade Engine để render ra giao diện người dùng, blade cung cấp nhiều macro cho phép người

lập trình viết các cú pháp hiển thị một cách ngắn gọn và tái sử dụng được nhiều code hơn. Một ví dụ đơn giản về blade view:

```
<html>
<head>
</head>
<body>
    <h1> Hello {{ $name }} </h1>
</body>
</html>
```

Ở ví dụ trên biến \$name sẽ được truyền từ phía controller sau đó sẽ được hiển thị ở trên giao diện người dùng. Ta có thể định nghĩa một route đơn giản để hiển thị blade view trên như sau:

```
Route::get('/', function() {
    return view('index', ['name' => 'world']);
});
```

Giao diện sẽ được hiển thị như sau:



Với blade engine ta hoàn toàn có thể làm nhiều thứ với các macro mặc định như @extends, @if, @foreach,...

2.3.7. Controllers

Controller được xem là trung tâm xử lý logic ứng dụng Laravel, tất cả các yêu cầu từ phía người dùng sẽ được đưa đến controller sau đó được xử lý và phản hồi lại cho phía bên người dùng. Controller cho phép nhóm các HTTP request liên quan đến logic do đó sẽ làm cho mã nguồn sẽ trở nên dễ đọc và dễ hiểu. Các controller sẽ được đặt ở trong thư mục app/Http/Controllers

Ví dụ về một controller

```

<?php

namespace App\Http\Controllers;

class HomeController extends Controller {
    public function index() {
        return view('index', ['name' => 'world']);
    }
}

```

Những resource controller làm cho việc xây dựng các RESTful controller xung quanh các nguồn tài nguyên trở nên dễ dàng hơn. Ví dụ như, ta muốn tạo một controller xử lý những HTTP request liên quan đến sản phẩm được lưu trữ trong ứng dụng. Sử dụng câu lệnh Artisan make:controller, chúng ta có thể nhanh chóng tạo ra controller:

```
php artisan make:controller SanPhamController --resource
```

Câu lệnh Artisan sẽ tạo ra file controller tại app/Http/Controllers/SPController.php. Controller sẽ bao gồm method cho các hoạt động của tài nguyên sẵn có bao gồm: index, show, update, destroy.

Tiếp theo, đăng ký một định tuyến đa tài nguyên cho controller:

```
Route::resource("/san-pham", "SPController");
```

Laravel sẽ tạo cho ta một danh sách các url:

GET /san-pham

GET /san-pham/:id

POST /san-pham/:id

PUT /san-pham/:id

DELETE /san-pham/:id

2.3.8. Middleware

HTTP middleware cung cấp một giải pháp tiện ích cho việc lock các HTTP request vào ứng dụng. Ví dụ, Laravel có chứa một middleware xác thực người dùng đăng nhập vào hệ thống. Nếu user chưa đăng nhập, middleware sẽ chuyển hướng user tới màn hình login. Còn nếu user đã đăng nhập rồi, thì middleware sẽ cho phép request được thực hiện tiếp tiến trình xử lý.

Tất nhiên là có thể viết thêm middleware để thực hiện nhiều tác vụ nữa ngoài việc kiểm tra đăng nhập vào hệ thống. Middleware CORS chịu trách nhiệm cho việc thêm các header hợp lý vào trong tất cả các response gửi ra ngoài. Middleware log có thể thực hiện ghi log cho tất cả các request tới chương trình. Vài middleware đã có sẵn trong Laravel framework, bao gồm middleware cho bảo trì, xác thực, phòng chống CSRF và còn nữa. Tất cả những middleware này nằm trong thư mục `app/Http/Middleware`.

2.3.9. Requests

Để lấy đối tượng của HTTP request hiện tại thông qua dependency injection, ta phải type- hint `Illuminate\Http\Request` vào trong hàm khởi tạo của controller hay phương thức trong controller. Đối tượng của request hiện tại sẽ được tự động inject vào bởi service container:

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;

class HomeController extends Controller
{
    public function store(Request $request)
    {
        $name = $request->get("name");
    }
}
```



```
| }
```

Một số method cơ bản của đối tượng Request:

- all: lấy hết những params được truyền lên.
- get: lấy giá trị dựa vào key truyền vào.
- input: cũng là lấy giá trị dựa vào key truyền vào.
- has: Kiểm tra xem key có được truyền lên hay không.
- only: Chỉ lấy những giá trị mà key được truyền vào.

2.3.10. Response

Tất cả các route và controller luôn nên trả về giá trị cho phía người dùng. Laravel cung cấp nhiều cách khác nhau để phản hồi lại phía người dùng. Cơ bản nhất sẽ là trả về dữ liệu text như ví dụ ở phía dưới:

```
| Route::get('/', function () {  
|     return 'Hello World';  
| })
```

Ngoài ra, Laravel còn hỗ trợ rất nhiều cách trả về dữ liệu như array, string, json,...

Khi phản hồi cho phía người dùng thì ta hoàn toàn có thể:

- Đính kèm header trả về cho người dùng.
- Đính kèm cookie để lưu lại thông tin đăng nhập.
- Mã hóa dữ liệu.

2.4. VueJS

2.4.1. VueJS là gì ?

VueJS là một Frontend Framework được viết bằng ngôn ngữ Javascript, cho phép tạo ra trang giao diện người dùng Single Page Application. VueJS là một trong những framework nổi tiếng được sử dụng để tạo ra ứng dụng web. Đối với VueJS các thành phần được viết thành các component sau đó ghép chúng lại để tạo ra một trang hoàn chỉnh. Với cách viết này, VueJS giúp các nhà phát triển có thể tái sử dụng lại code một cách dễ dàng.

Một file vue thông thường sẽ có đuôi .vue và có định dạng như sau:

```
<template>
//
</template>
<script>
//
</script>
<style>
//
</style>
```

2.4.2. Data:

Trong phần script của VueJS sẽ chứa một object được sử dụng làm data để render trong template:

```
<script>
export default {
  data() {
    return {
      Name: 'Hello world'
    };
  }
}
</script>
```

2.4.3. Directives

VueJS sử dụng các thuộc tính đặc biệt được gọi là directives để "điều khiển" giao diện dựa trên phần dữ liệu trong script.

Một số directive cơ bản:

- v-if, v-show: Thường được sử dụng để ẩn hiện UI
- v-for : Thường dùng để duyệt qua dữ liệu để tạo ra các component con

- v-bind: Dùng để gắn dữ liệu vào trong component theo cú pháp: v-bind:[attribute]: [value]
- v-html: Hiển thị các text có input là dạng HTML

2.4.4. Methods

Trong mỗi file .vue ta có thể định nghĩa các hàm thực hiện một chức năng bất kì để có thể sử dụng lại nó khi cần. Các hàm này có thể được gọi cả trong phần script lẫn phần template:

```
<script>
export default {
  methods: {
    async getData() {
      const response = await fetchDataHere('https://jsonplaceholder.com/posts');
      console.log(response)
    }
  }
}
</script>
```

Với ví dụ trên method `getData` sẽ tiến hành fetch dữ liệu từ trang <https://jsonplaceholder.com/posts> sau đó sẽ tiến hành in ra dữ liệu vừa fetch được

2.4.5. Computed

Computed Properties có thể hiểu là nó khai báo giống như 1 methods nhưng lại được truy cập giống như cách chúng ta truy cập data của component:

```
<script>
export default {
  data() {
    return {
      numbers: [5, 8, 3]
    }
  },
}
```

```

    computed: {
      numberTotal() {
        Return this.numbers.reduce((sum, val) => sum + val);
      }
    }
  }
</script>

```

2.4.6. Watchers

Watchers cho phép theo dõi sự thay đổi thuộc tính trong data hoặc trong computed property.

```

<script>
  export default {
    data() {
      return {
        count: 0,
      },
    },
    watch: {
      count(newValue, oldValue) {
        console.log("Bien count da thay doi");
      }
    }
  }
</script>

```

2.4.7. Filters

Filters đóng vai trò giống một bộ format cho string hoặc số.

```

<template>
  <div id="app">

```

```

        <p>{{ name| tranform }}</p>
    </div>
</template>
<script>
    export default {
        data() {
            return {
                name: "foo",
            }
        }, filters: {
            tranform (value) {
                if (!value) return ""
                value = value.toString()
                return value.toLowerCase();
            }
        }
    }
</script>

```

2.4.8. Props

Ứng dụng VueJS được xây dựng dựa trên các components khác nhau và hơn nữa các components còn có thể truyền data của nó xuống component con thông qua props.

```
<component :data="{ text: 'hello world' }"></component>
```

2.4.9. Events

Để có thể thực hiện các thao tác như click trong ứng dụng ta sẽ sử dụng thêm một directives khác đó là v-on. Cú pháp sẽ là v-on:[event]="[handler]"

2.4.10. Slot

Slot là cách mà ta có thể sử dụng để tạo ra các component khung với nội dung có thể thay đổi tùy theo vị trí mà ta muốn sử dụng nó.

Ví dụ:

```
// modal.vue
<template>
  <div class="modal">
    <div class="modal-header">
      <slot name="header" />
    </div>
    <div class="modal-content">
      <slot />
    </div>
    <div class="modal-footer">
      <slot name="footer" />
    </div>
  </div>
</template>

// login
<template>
  <modal>
    <template slot="header">Header</template>
    <div>Content</div>
    <template slot="footer">Footer</template>
  </modal>
</template>
```

2.5. Supervisor

2.5.1. Supervisor là gì

Supervisor là công cụ giúp quản lý giám sát các tiến trình đang chạy trên Linux. Supervisor sẽ giúp một tiến trình ở trong hệ thống có thể chạy liên tục, tạo ra các log

giúp người vận hành giám sát được các công việc đã và đang thực hiện. Khi có các lỗi xảy ra thì supervisor sẽ tiến hành thử lại, các thông số này có thể được cấu hình.

2.5.2. Cài đặt supervisor

Để cài Supervisor, ta chạy lệnh sau:

```
| sudo apt install supervisor
```

2.5.3. Cấu hình supervisor để giám sát laravel queue

Mỗi tiến trình do Supervisor giám sát sẽ được cấu hình ở file dạng .conf nằm trong thư mục /etc/supervisor/conf.d

Cấu trúc 1 file config cơ bản sẽ như sau:

```
| [program:laravel-worker]
| process_name=%(program_name)s_%(process_num)02d
| directory=/home/luccui/admin
| command=/home/luccui/admin/php artisan queue:work --tries=3
| autostart=true
| autorestart=true
| user=root
| numprocs=4
| redirect_stderr=true
| stdout_logfile=/home/luccui/admin/storage/logs/worker.log
```

Sau khi cấu hình xong ta chạy lệnh sau:

```
| systemctl restart supervisord
```

Kiểm tra supervisor đã hoạt động chưa ta chạy lệnh sau:

```
| systemctl status supervisord
```

Thông tin sẽ hiển thị như sau:

```
● supervisor.service - Supervisor process control system for UNIX
   Loaded: loaded (/lib/systemd/system/supervisor.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-02-13 09:03:18 +07; 4 days ago
     Docs: http://supervisord.org
    Main PID: 1541 (supervisord)
      Tasks: 1 (limit: 4915)
   CGroup: /system.slice/supervisor.service
           └─1541 /usr/bin/python /usr/bin/supervisord -n -c /etc/supervisor/supervisord.conf
```

CHƯƠNG 3: ĐỐI TƯỢNG VÀ PHƯƠNG PHÁP NGHIÊN CỨU

3.1. Đối tượng nghiên cứu

Tìm hiểu các khái niệm cơ bản liên quan mô hình hướng sự kiện và mô hình hàng đợi tin nhắn RabbitMQ.

Tìm hiểu cơ chế hoạt động của Laravel, Vue để xây dựng ứng dụng website bán hàng.

Nghiên cứu RestfulAPI với Laravel và ứng dụng hàng đợi tin nhắn trong dự án mô hình hướng sự kiện.

Nghiên cứu cài đặt phát triển website bán hàng.

3.2. Phương pháp nghiên cứu

Tìm hiểu và sử dụng mô hình hàng đợi tin nhắn RabbitMQ để xây dựng website bán hàng trực tuyến, quản lý nghiệp vụ cửa hàng cho phép người dùng có thể mua bán sản phẩm trực tuyến.

Tìm hiểu nguyên lý xây dựng website bán hàng trực tuyến và tích hợp hệ thống thanh toán trực tuyến.

3.3. Đạo đức nghiên cứu

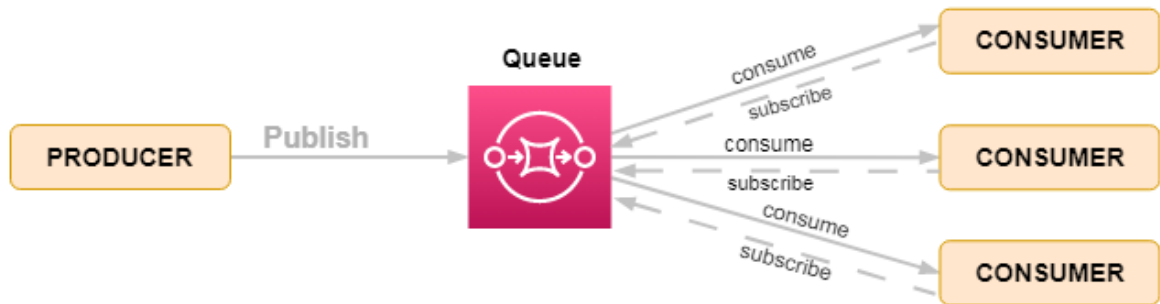
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU VÀ THẢO LUẬN

4.1. Tổng quan về mô hình hàng đợi tin nhắn RabbitMQ

4.1.1. AMQP là gì ?

RabbitMQ là một message broker (message-oriented middleware) sử dụng giao thức AMQP – Advanced Message Queue Protocol. Nó được viết bằng ngôn ngữ lập trình Erlang. RabbitMQ chạy được trên đa hệ điều hành, nó cung cấp các công cụ và trình quản lý giao diện cực kỳ trực quan cho người lập trình để giám sát và một phương tiện trung gian để giao tiếp giữa nhiều thành phần trong một hệ thống lớn. RabbitMQ là trung gian nhận các tin nhắn từ nhiều thành phần của hệ thống sau đó sẽ đưa đến các thành phần chịu trách nhiệm xử lý để xử lý. Trong một hệ thống phân tán, việc giao tiếp giữa các thành phần với nhau gần như là bắt buộc, nhưng nếu có quá nhiều giao tiếp mà các liên kết này quá chặt chẽ sẽ làm cho ứng dụng trở nên phức tạp, việc bảo trì ứng dụng trở thành một việc đặt biệt khó khăn. Khi các thành phần trong ứng dụng giao tiếp và liên kết với nhau quá nhiều thì sẽ rất khó khăn trong việc triển khai code. Một khi một thành phần bị lỗi thì nó sẽ ảnh hưởng đến tất cả các thành phần mà lệ thuộc vào nó. Giải pháp ở đây là thay thế các liên kết trực tiếp, để các thành phần này được giao tiếp với nhau thì phải sử dụng một giao thức trung gian qua một message broker. Với sự tham gia của một message broker các producer có nhiệm vụ chính là đưa tin nhắn vào trong message queue, từ message queue này message broker có nhiệm vụ phân phát đến các consumer chịu trách nhiệm xử lý message đó. Vì là producer giao tiếp các consumer trung gian thông qua giao tiếp trung là là message broker nên ở hai thành phần này chúng ta hoàn toàn có thể viết được bằng các ngôn ngữ khác nhau, do đó thì sẽ tận dụng được ưu thế của từng ngôn ngữ lập trình để phát triển hệ thống phù hợp với từng yêu cầu. Một đặc tính quan trọng của RabbitMQ là các tác vụ này sẽ xảy ra bất đồng bộ. Các producer không thể biết được consumer nào sẽ được giao để xử lý tác vụ này cũng như không thể nào biết được khi nào tác vụ này được hoàn thành. Đối với producer, nhiệm vụ chính chỉ là đưa tin nhắn vào trong hàng đợi còn việc phân phát tin nhắn đến đâu thì việc này do RabbitMQ chịu trách nhiệm. Một RabbitMQ được xem như là một trung gian hoạt động

của dịch vụ của hệ thống, nó có thể giảm được độ trễ và tốc độ tải của một ứng dụng bằng cách ủy thác các tác vụ cho các hệ thống bên thứ ba.



Hình 4-1. Cách hoạt động của producer và consumer

4.1.2. Sự tách biệt thực sự giữa bên public message và bên nhận thông tin

Hệ thống sử dụng kiến trúc hướng sự kiện tách các thành phần trong hệ thống, phân tách quyền sở hữu dữ liệu theo miền:

- Bên publish message không cần quan tâm đến việc các sản phẩm mà họ sản xuất sẽ được tiêu thụ như thế nào.
- Bên nhận thông tin không cần quan tâm đến việc chúng được sản xuất như thế nào. Do mô hình sử dụng cách thức trao đổi này, nên thường các hệ thống microservices có thể được thực hiện bằng các ngôn ngữ khác nhau hoặc sử dụng các công nghệ khác nhau và phù hợp với các công việc cụ thể. Do đó, việc mã hóa dữ liệu sự kiện không quan trọng – nó có thể là JSON, XML, Avro, v.v.

Việc tách các thành phần của một ứng dụng cũng cho phép chúng được chia tỷ lệ dễ dàng và độc lập với nhau trên toàn bộ dự án. Các nhà phát triển có thể sửa đổi hệ thống nhanh chóng bằng cách thêm hoặc xóa động các producer và bên nhận thông tin mà không cần thay đổi bất kỳ logic nào trong bất kỳ dịch vụ nào.

4.1.3. Các loại exchange trong RabbitMQ

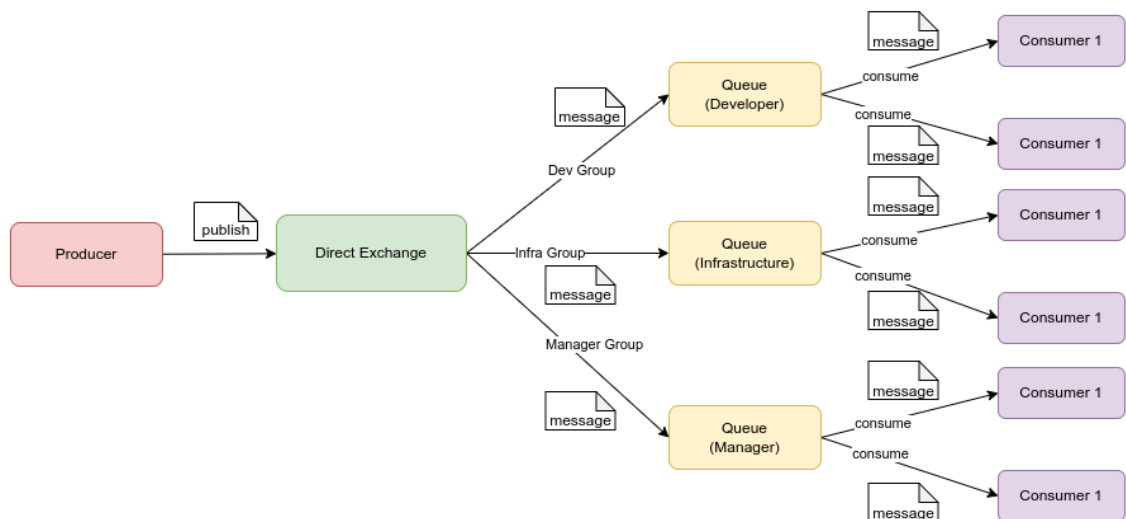
Direct Exchange

Trong mô hình hàng đợi hàng đợi tin nhắn, direct exchange được dùng để vận chuyển các tin nhắn ở trong hàng đợi sau đây sẽ đưa đến các consumer bằng cách dựa vào các

routing key. Loại exchange này thường thấy nhất sẽ sử dụng ở trong các trường hợp định tuyến tin nhắn đơn hướng mặc dù nó có thể sử dụng cho định tuyến đa hướng.

Luồng đi của một Direct exchange sẽ như sau:

- Một producer sau khi tạo ra một message thì sẽ đẩy message này tới exchange.
- Một hàng đợi sẽ tiến hành binding tới exchange này bằng cách sử dụng routing key. Ta có thể hoàn toàn tạo ra nhiều queue và binding chúng tới exchange và cũng có thể sử dụng chung routing key hoặc là khác routing key.
- Một tin nhắn khi được đẩy vào trong exchange có gắn kèm routing key, dựa theo thông tin routing key này thì tin nhắn sẽ được đưa đến một hoặc là nhiều hàng đợi đã đăng ký trước đó



Hình 4-2. Cách hoạt động của Direct Exchange

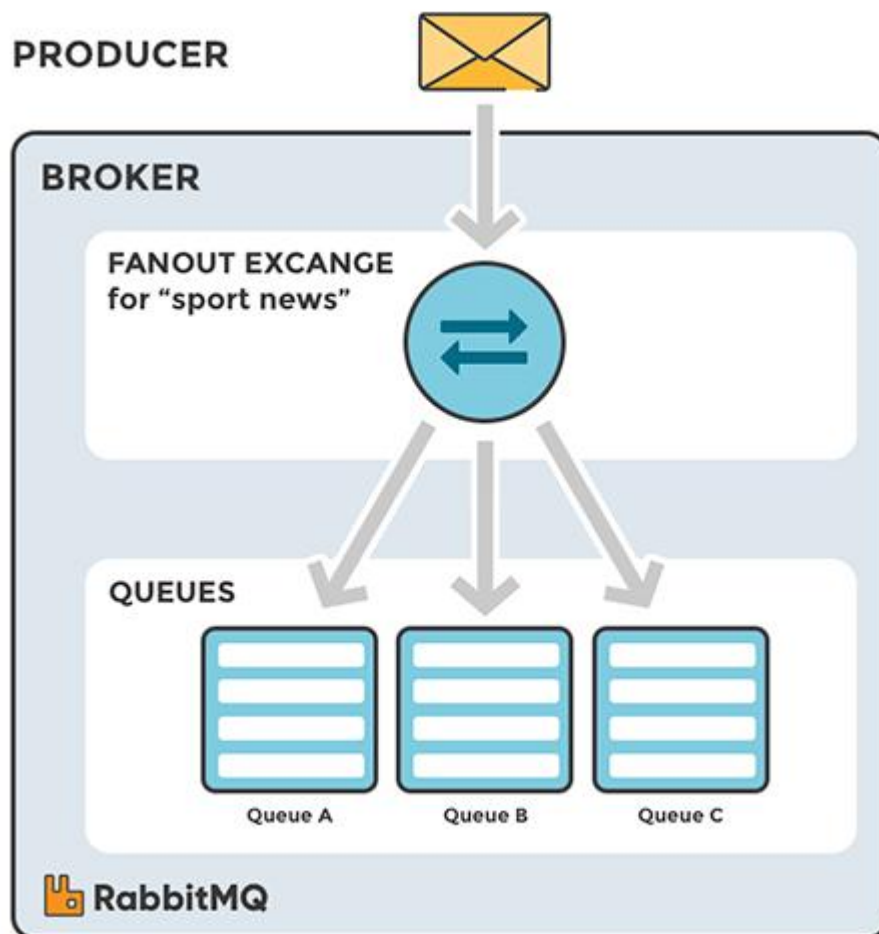
Default Exchange

Trong rabbitMQ thì mỗi exchange sẽ không được phép đặt tên trùng nhau, default exchange thực chất cũng là một direct exchange nhưng không có tên. Nếu mà lựa chọn default exchange thì các tin nhắn sẽ được đẩy đến các hàng đợi cùng tên với routing key. Khi mà routing key có tên giống với tên của hàng đợi thì mỗi khi có tin nhắn mới thì ngay lập tức sẽ được liên kết và trao đổi.

Fanout Exchange

Fanout exchange sẽ vận chuyển các tin nhắn đến tất cả các queue mà ở xung quanh nó. Các tin nhắn sẽ được phân tán ra khắp các queue ngoại trừ routing key hoàn toàn sẽ bị bỏ qua.

Fanout exchange hữu ích khi mà một tin nhắn hay thông điệp cần chuyển đến một hay là nhiều queue và mỗi message sẽ có cách xử lý tin nhắn khác nhau. Giống như có một việc gì đó xảy ra chẳng hạn như một dự báo thời tiết hay là một sự kiện gì đó thì tất cả các thiết bị di động được kết nối sẽ được thông báo.



Hình 4-3. Cách hoạt động của Fanout Exchange

Topic exchange

Topic exchange định tuyến tới một hoặc nhiều hàng đợi dựa trên sự trùng khớp giữa routing key và pattern. Topic exchange thường được sử dụng để thực hiện định tuyến tin nhắn phát đa hướng. Một số ví dụ về các trường hợp sử dụng:

- Phân phối dữ liệu liên quan đến vị trí địa lý cụ thể.
- Xử lý tác vụ nền được thực hiện bởi nhiều workers, mỗi công việc có khả năng xử lý các nhóm tác vụ cụ thể.
- Cập nhật tin tức liên quan đến phân loại hoặc gắn thẻ (ví dụ: chỉ dành cho một môn thể thao hoặc đội cụ thể).
- Điều phối các dịch vụ của các loại khác nhau trong cloud

Header exchange

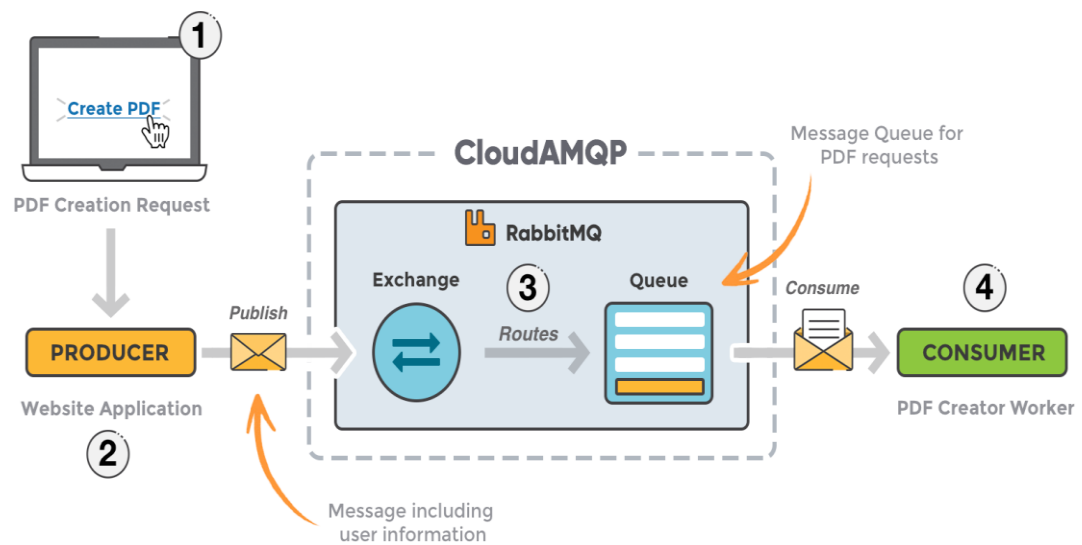
Là một hệ thống định tuyến sử dụng các đối số của header và các giá trị tùy chọn để định tuyến tin nhắn đến các hàng đợi liên quan. Header exchange cũng tương như topic exchange ngoại trừ thay vì sử dụng các routing key đối với header exchange thì sẽ sử dụng giá trị từ header để định tuyến. Nếu mà giá trị của nguồn khớp với giá trị của của header thì tin nhắn sẽ được liên kết và trao đổi với nhau.

Trong liên kết giữa exchange và hàng đợi, một đối số cụ thể được gọi là x-match sẽ cho biết tất cả các header phải khớp hay là chỉ duy nhất một header được phép. Thuộc tính x-match sẽ có giá trị có thể là any hoặc là all, all sẽ là giá trị mặc định. Giá trị của “all” cho biết rằng tất cả các cặp tiêu đề (key, value) phải khớp, trong khi “any” cho biết rằng ít nhất một cặp phải khớp. Thay vì một chuỗi, các tiêu đề có thể được tạo với nhiều loại dữ liệu hơn, chẳng hạn như số nguyên hoặc giá trị băm. Loại trao đổi tiêu đề (khi được sử dụng với tùy chọn liên kết “all”) hữu ích cho việc điều hướng các thông báo có chứa một tập hợp con các tiêu chí đã biết (không có thứ tự).

4.1.4. Khi nào vào tại sao nên sử dụng RabbitMQ

Hàng đợi tin nhắn cho phép các máy chủ web phản hồi các yêu cầu một cách nhanh chóng thay vì bị buộc phải thực hiện các quy trình tốn nhiều tài nguyên ngay tại chỗ có thể làm chậm thời gian phản hồi. Hàng đợi tin nhắn cũng hữu ích khi muốn phân phối một tin nhắn cho nhiều consumer hoặc để cân bằng tải giữa các worker .

Ví dụ: Các consumer lấy một tin nhắn ra khỏi hàng đợi và bắt đầu xử lý tệp PDF. Đồng thời, producer đang xếp hàng các tin nhắn mới. Yêu cầu có thể được tạo bằng một ngôn ngữ lập trình và được xử lý bằng ngôn ngữ lập trình khác.



Hình 4-4. Ví dụ về RabbitMQ xử lý file PDF

1. Người dùng gửi yêu cầu tạo PDF tới ứng dụng web.
2. Ứng dụng web (producer) gửi một thông báo tới RabbitMQ bao gồm dữ liệu từ yêu cầu như tên và email.
3. Một exchange chấp nhận các tin nhắn từ producer và định tuyến chúng đến đúng hàng đợi tin nhắn để tạo file PDF.
4. Các worker xử lý PDF (consumer) nhận được thông báo tác vụ và bắt đầu xử lý PDF.

4.1.5. Hướng dẫn cài đặt và sử dụng RabbitMQ

Trong bài hướng dẫn này sẽ sử dụng docker để cài đặt RabbitMQ, một trong lợi ích cực kỳ lớn của docker là

Trang Docker Hub của RabbitMQ cung cấp cho các Docker Image bao gồm cả Management plugin sử dụng Alpine Linux (1 Linux version được tối ưu để có dung lượng rất nhỏ). Chúng ta có thể sử dụng những tag có tên kết thúc với management-alpine để cài đặt RabbitMQ sử dụng Docker Compose.

Tạo file cấu hình docker-compose.yml với nội dung sau:

```
version: '3.9'

services:

  rabbitmq:

    image: rabbitmq:3-management-alpine

    container_name: 'rabbitmq'

    ports:

      - 5672:5672

      - 15672:15672

    volumes:

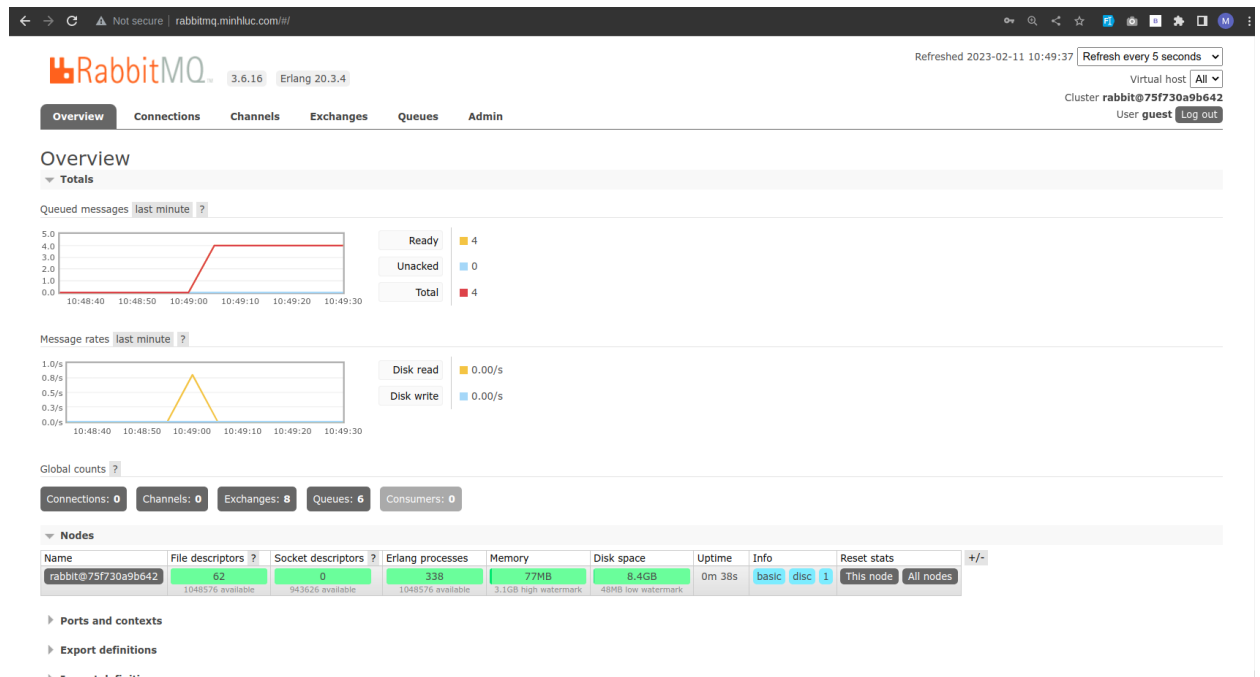
      - /home/luccui/workspace/laravel/docker/data:/var/lib/rabbitmq

      - /home/luccui/workspace/laravel/docker/log:/var/log/rabbitmq
```

Ở đây, đang cài đặt latest version của RabbitMQ 3.9.15. Chúng ta cần expose 2 port 5672 và 15672 của RabbitMQ server và RabbitMQ Management UI để cho bên ngoài sử dụng. 2 thư mục bên trong container của RabbitMQ cũng được mount ra các thư mục

bên ngoài để có thể dễ dàng làm việc với RabbitMQ. Kết quả khi chạy command “docker-compose up” trong thư mục chứa tập tin docker-compose.yaml với nội dung trên như sau:

Như vậy là đã start được một RabbitMQ server cùng với management UI. Bây giờ thì ta đã có thể access tới RabbitMQ Management UI sử dụng địa chỉ <http://localhost:15672/>. Đăng nhập với username và password là guest, ta sẽ thấy kết quả như sau:



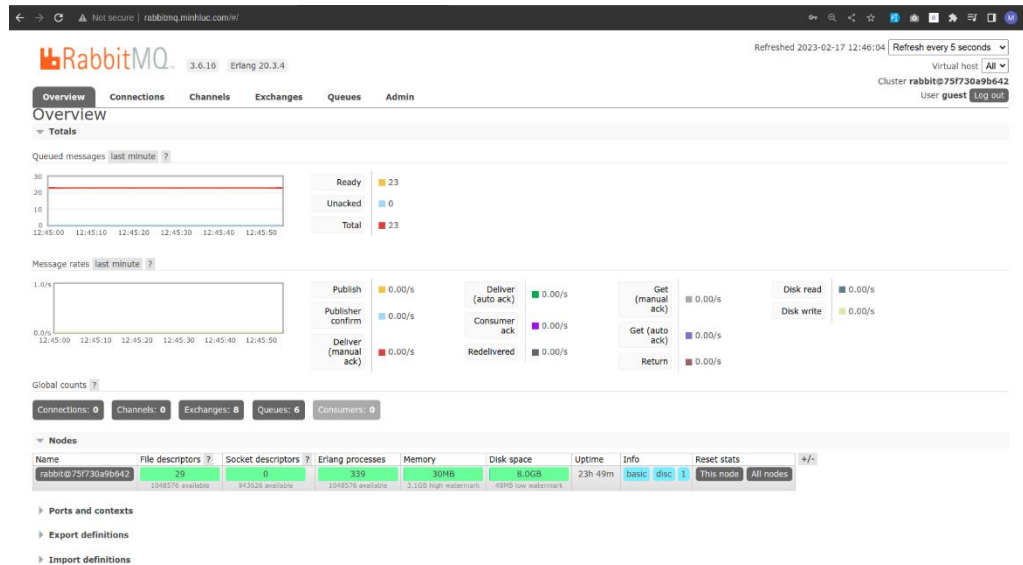
Hình 4-5. Giao diện tổng quan RabbitMQ

4.2. Giới thiệu các thành phần trong RabbitMQ Management Interface

RabbitMQ cung cấp cho các nhà phát triển giao diện giám sát các tin nhắn hoạt động trong hệ thống được luân chuyển như thế nào.

4.2.1. Overview

Hiện thị hai biểu đồ chính, biểu đồ thứ nhất thể hiện các tin nhắn được đẩy vào trong hàng đợi và biểu đồ thứ hai cho biết tốc độ xử lý của tin nhắn đó.



Hình 4-1. Giao diện overview RabbitMQ

Overview

Đối với queue messages:

- **Ready:** Cho biết trạng thái các tin nhắn đang sẵn sàng để đưa đến các consumer
- **Unacked:** Cho biết số lượng tin nhắn đang chờ hàng đợi xác nhận
- **Total:** Cho biết tổng số tin nhắn trong hàng đợi

Đối với Message rates:

- **Publish:** cho biết thông tin tốc độ tin nhắn được gửi lên hệ thống từ ứng dụng.
- **Publisher confirm:** Thông tin tốc độ mà hệ thống phản hồi
- **Deliver:** Hiện thị tốc độ tin nhắn gửi theo phương thức ACK

Nodes

Ta hoàn toàn có thể phân tán các hàng đợi tin nhắn ra thành các cluster với một node master quản lý hết tất cả các cluster đó. Việc ứng dụng phân tán sẽ giúp ích và giảm tải rất nhiều cho hàng đợi.

Churn statistics

Hiện thị các trạng thái tạo mới cập nhật message trong các connection, queue.

Ports & context

Thông tin cổng lắng nghe của hệ thống đang sử dụng.

Import & export definition:

Cho phép tải về và thêm vào cấu hình cho RabbitMQ.

4.2.2. Connections

Hiển thị connection giữa chương trình ứng dụng và RabbitMQ. Ở trang này sẽ hiển thị một số thông tin cơ bản sau:

- Địa chỉ IP
- Tên người dùng truy cập
- Trạng thái
- Giao thức SSL/TLS
- Channels

4.2.3. Channels

Hiển thị tất các channel hiện có trong hệ thống RabbitMQ. Ở trang này sẽ hiển thị bảng với các thông số cơ bản:

- Địa chỉ IP
- Tên người dùng
- Trạng thái
- Message rates,...

4.2.4. Exchanges

Khi tin nhắn được đẩy vào trong hàng đợi RabbitMQ thì exchange sẽ có nhiệm vụ định tuyến dựa theo cấu hình mà đã được thiết lập trước đó để phân phát tin nhắn đến từng message queue. Ở trang này sẽ hiển thị bảng với các thông số cơ bản:

- Tên exchange
- Loại exchange
- Tỷ lệ xử lý tin nhắn

3.6.16 Erlang 20.3.4
Refreshed 2023-02-17 17:32:49
Refresh every 5 seconds

Overview
Connections
Channels
Exchanges
Queues
Admin

Cluster **rabbit@75f730a9b642**
User **guest** [Log out](#)

Exchanges

▼ All exchanges (8)

Page 1 of 1 - Filter: ☐ Regex ?

Displaying 8 items , page size up to: 100

Name	Type	Features	Message rate in	Message rate out	+/-
(AMQP default)	direct	D	0.00/s	0.00/s	
amq.direct	direct	D			
amq.fanout	fanout	D			
amq.headers	headers	D			
amq.match	headers	D			
amq.rabbitmq.log	topic	D I			
amq.rabbitmq.trace	topic	D I			
amq.topic	topic	D			

► Add a new exchange

Hình 4-6. Giao diện Exchange RabbitMQ

4.2.5. Queue

Hiện thị danh sách queue của hệ thống. Ở trang này sẽ hiển thị một số thông số quan trọng người giám sát hệ thống cần phải biết:

- Tên queue
- Các tham số đầu vào của queue
- Số lượng tin nhắn đang chờ xác nhận
- Tổng số tin nhắn đã và đang xử lý
- Tỷ lệ xử lý tin nhắn

3.6.16 Erlang 20.3.4
Refreshed 2023-02-17 17:32:40
Refresh every 5 seconds

Overview
Connections
Channels
Exchanges
Queues
Admin

Cluster **rabbit@75f730a9b642**
User **guest** [Log out](#)

Queues

▼ All queues (6)

Page 1 of 1 - Filter: ☐ Regex ?

Displaying 6 items , page size up to: 100

Overview			Messages			Message rates			+/-
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
admin_queue	D	idle	16	0	16	0.00/s	0.00/s	0.00/s	
mail_queue	D	idle	3	0	3	0.00/s			
order	D	idle	0	0	0				
order_queue	D	idle	4	0	4				
product	D	idle	0	0	0				
product_queue	D	idle	0	0	0				

► Add a new queue

Hình 4-7. Giao diện Queue RabbitMQ

4.2.6. Admin

Tại trang admin ta có thể thay đổi thông số về policy, permission queue.

4.3. Sử dụng RabbitMQ trên ứng dụng Laravel

Trước hết ta cần phải cài đặt thư viện bằng dòng lệnh sau:

```
composer require vladimir-yuldashev/laravel-queue-rabbitmq
```

Package sẽ tự động được đăng ký trong ứng dụng Laravel

Thêm connection vào trong file config/queue.php

```
'connections' => [  
  
    // ...  
  
    'rabbitmq' => [  
  
        'driver' => 'rabbitmq',  
  
        'queue' => env('RABBITMQ_QUEUE', 'default'),  
  
        'connection' => PhpAmqpLib\Connection\AMQPLazyConnection::class,  
  
        'hosts' => [  
  
            [  
  
                'host' => env('RABBITMQ_HOST', '127.0.0.1'),
```

```

        'port' => env('RABBITMQ_PORT', 5672),

        'user' => env('RABBITMQ_USER', 'guest'),

        'password' => env('RABBITMQ_PASSWORD', 'guest'),

        'vhost' => env('RABBITMQ_VHOST', '/'),

    ],

],

'options' => [

    'ssl_options' => [

        'cafile' => env('RABBITMQ_SSL_CAFILE', null),

        'local_cert' => env('RABBITMQ_SSL_LOCALCERT', null),

        'local_key' => env('RABBITMQ_SSL_LOCALKEY', null),

        'verify_peer' => env('RABBITMQ_SSL_VERIFY_PEER', true),

        'passphrase' => env('RABBITMQ_SSL_PASSPHRASE', null),

    ],

    'queue' => [

        'job'
        VladimirYuldashev\LaravelQueueRabbitMQ\Queue\Jobs\RabbitMQJob::class,

    ],

```

=>

```
],

/*

* Set to "horizon" if you wish to use Laravel Horizon.

*/

'worker' => env('RABBITMQ_WORKER', 'default'),

'after_commit' => false,

],

// ...

]
```

Ta hoàn toàn có thể tùy chỉnh một số cấu hình của RabbitMQ như sau:

- Khi mà ta muốn trì hoãn các message được xử lý ta có thể thêm vào trong cấu hình như sau:

```
'options' => [

    'queue' => [

        // ...

        'prioritize_delayed' => false,

        'queue_max_priority' => 10,
```

```
    ],  
    ],  
]
```

- Khi muốn publish message dưới dạng exchange routing key thì có thể sử dụng cấu hình tham khảo dưới đây:

```
'options' => [  
    'queue' => [  
        // ...  
        'exchange' => 'application-x',  
        'exchange_type' => 'topic',  
        'exchange_routing_key' => '',  
    ]  
]
```

- Trong Laravel, các công việc thất bại được lưu trữ trong CSDL. Nhưng ta có thể hoàn toàn chỉ định một số quy trình khác cũng làm điều gì đó với thông báo. Khi muốn hướng dẫn RabbitMQ định tuyến lại các message thất bại đến một exchange hoặc một queue cụ thể, thì điều này có thể thực hiện được bằng cách thêm các tùy chọn bổ sung.

```
'options' => [  

```

```
'queue' => [  
  
    'reroute_failed' => true,  
  
    'failed_exchange' => 'failed-exchange',  
  
    'failed_routing_key' => 'application-x.%s',  
  
    ],  
  
]
```

4.4. Mô tả ứng dụng

Một cửa hàng thời trang đang phát triển và có dấu hiệu người dùng tăng lên đột hàng ngày có thể xử lý được khoảng chục nghìn đơn hàng cùng một lúc muốn xây dựng một website bán hàng thời trang online với mục tiêu sau:

Cửa hàng sẽ bán nhiều loại mặt hàng thời trang khác nhau có thể là quần áo, giày dép, các phụ kiện thời trang,.... Để quản lý được các mặt hàng thuộc nhóm nào thì mỗi mặt hàng được gom vào trong một loại danh mục khác nhau. Thông tin của danh mục gồm mã và tên danh mục.

Thông thường các loại sản phẩm thời trang bán ra sẽ có hai loại chính là sản phẩm không có biến thể tức là sản phẩm này sẽ không có các tùy chọn về kích cỡ hoặc là màu sắc của sản phẩm. Thông tin chi tiết của sản phẩm gồm mã sản phẩm, tên sản phẩm, mô tả ngắn về sản phẩm, nội dung mô tả chi tiết về sản phẩm này, hình ảnh minh họa sản phẩm, sản phẩm này có thuộc loại hot, mới nhất hay không. Đối với sản phẩm có nhiều biến thể thì sẽ có biến thể về màu sắc và kích cỡ. Một sản phẩm này sẽ có một hoặc nhiều màu sắc và một hoặc nhiều kích cỡ. Các màu sắc và kích cỡ được gom lại thành một biến thể của sản phẩm và từng biến thể này sẽ có một mức giá cũng như một mã nhất định. Khi mà sản phẩm không còn hàng hoặc số lượng đặt hàng quá nhiều so với cửa

hàng đang có thì website vẫn hiển thị nhưng hệ thống sẽ đưa ra gợi ý cho khách hàng thông tin liên hệ mặt hàng này. Thông tin liên hệ mặt hàng sẽ gồm mã sản phẩm, thông tin liên hệ như tên, email, số điện thoại, ngày liên hệ. Các thông tin này sau khi được khách hàng liên hệ thì bên cửa hàng sẽ có trách nhiệm gọi đến để tư vấn về sản phẩm. Khách hàng khi mua sắm tại website thì sẽ tiến hành đăng ký tài khoản bằng số điện thoại sau khi đã đăng ký thành công thì hệ thống sẽ tiến hành gửi mã xác thực số điện thoại có hợp lệ hay không sau khi đã xác nhận hợp lệ thì hệ thống sẽ thêm thông tin khách hàng vào trong CSDL thông tin này sẽ bao gồm: mã khách hàng, họ tên khách hàng, email, số điện thoại. Khi khách hàng muốn mua sắm thì sẽ tiến hành đăng nhập và tiến hành thêm các sản phẩm vào trong giỏ hàng. Mỗi khách hàng có thể thêm được nhiều loại sản phẩm. Sau khi đã lựa chọn được hết tất cả các sản phẩm muốn mua thì sẽ tiến hành thanh toán cho đơn hàng mà mình đã đặt.

Mỗi khách hàng sẽ được phân vào trong từng nhóm khách hàng. Từng nhóm khách hàng này sẽ được sắp xếp theo từng mức độ mua hàng của từng khách hàng, thông tin về nhóm khách hàng này sẽ bao gồm, tên nhóm khách hàng, và nhóm khách hàng này có thể thuộc về một nhóm khách hàng nào đó. Các nhóm khách hàng này sẽ được ưu đãi một số giảm giá về sản phẩm. Khi có một sự kiện nào đó để kích thích sự mua sắm của khách hàng thì hệ thống sẽ tạo ra một chương trình giảm giá có thể sẽ là giảm giá theo phần trăm của đơn hàng hoặc là giảm giá theo giá trị đơn hàng. Sau khi đã tạo ra được chương trình giảm giá thì hệ thống sẽ tiến hành gửi email ưu đãi đến từng khách hàng nhận được ưu đãi. Thông tin về giảm giá này sẽ gồm: mã giảm giá, loại giảm giá, giá trị giảm giá, ngày sẽ bắt đầu áp dụng giảm giá này. Đối với những khách hàng không thuộc bất kì ưu đãi này cũng có thể nhận được mã giảm giá từ các chương trình mà hệ thống tạo ra. Thông tin về mã giảm giá sẽ gồm: mã mã giảm giá, tên chương trình giảm giá, mã giảm giá, ngày bắt đầu, ngày kết thúc, loại giảm giá, giá trị giảm.

Để tăng lượng tương tác website thì bên cửa hàng cũng sẽ tạo ra những bài đánh giá về sản phẩm hay là một số bài viết giới thiệu về sản phẩm. Thông tin về bài viết bao gồm mã bài viết, tên bài viết, nội dung bài viết để tăng chất lượng hiển thị hệ thống

cũng có một số tiêu chí về seo giúp hiển thị được top tìm kiếm trên google. Các bài viết này có thể thuộc về nhiều loại danh mục và sẽ có nhiều loại hashtag khác nhau.

4.5. Yêu cầu hệ thống

4.5.1. Yêu cầu chức năng

- Xem danh sách sản phẩm bán.
- Tìm kiếm sản phẩm.
- Phân loại sản phẩm theo danh mục, nhà cung cấp, quy cách.
- Thêm sản phẩm vào trong giỏ hàng.
- Xóa sản phẩm trong giỏ hàng.
- Xem thông tin giỏ hàng.
- Đặt hàng.
- Đăng ký, đăng nhập tài khoản khách hàng.
- Theo dõi đơn hàng cho khách không đăng ký tài khoản.
- Thanh toán đơn hàng.
- Quản trị hệ thống.
- Duyệt đơn hàng.
- Chức năng thống kê sản phẩm mua nhiều nhất, xem nhiều nhất.
- Chức năng báo cáo doanh thu.

4.5.2. Yêu cầu phi chức năng

- Giao diện thân thiện, dễ sử dụng.
- Tốc độ truy xuất nhanh.
- Hạn chế đến mức thấp nhất các sai sót có thể xảy ra trong quá trình sử dụng và có khả năng mở rộng và tích hợp các tính năng mới.
- CSDL kết nối chính xác và toàn vẹn dữ liệu.

4.6. Mô hình CSDL

Mô hình dữ liệu

Mô tả chi tiết các bảng

Mô tả bảng customer_categories(loại khách hàng)

Bảng 4-1. Chi tiết bảng loại khách hàng

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	Id	bigint	Khóa chính	Mã nhóm khách hàng
2	name	varchar(100)		Tên nhóm khách hàng
3	parent_id	bigint	Khóa ngoại	Tên nhóm cha
4	created_at	timestamp		Ngày tạo
5	updated_at	timestamp		Ngày cập nhật

Mô tả bảng customers (khách hàng)

Bảng 4-2. Chi tiết bảng khách hàng

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã khách hàng
2	first_name	varchar(50)		Họ
3	last_name	varchar(50)		Tên
4	phone	varchar(11)		Số điện thoại
5	email	varchar(100)		Email
6	password	varchar(100)		Mật khẩu
7	verify_token	varchar(10)		Mã xác thực

8	is_verified	int		Kiểm tra xác thực
9	province_id	int		Mã tỉnh
10	district_id	int		Mã huyện
11	ward_id	int		Mã xã
12	ship_address	varchar(255)		Địa chỉ
13	created_at	timestamp		Ngày tạo
14	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **brands** (thương hiệu)

Bảng 4-3. Chi tiết bản thương hiệu

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã thương hiệu
2	name	varchar(50)		Tên thương hiệu
3	created_at	timestamp		Ngày tạo
4	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **categories** (danh mục)

Bảng 4-4. Chi tiết bảng danh mục

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã danh mục
2	name	varchar(50)		Tên danh mục

3	created_at	timestamp		Ngày tạo
4	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **colors** (màu sắc)

Bảng 4-5. Chi tiết bảng màu sắc

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	Id	bigint	Khóa chính	Mã màu sắc
2	Name	varchar(50)		Tên màu sắc
3	created_at	timestamp		Ngày tạo
4	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **sizes** (kích cỡ)

Bảng 4-6. Chi tiết bảng kích cỡ

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã kích cỡ
2	name	varchar(50)		Tên kích cỡ
3	created_at	timestamp		Ngày tạo
4	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **coupons** (mã giảm giá)

Bảng 4-7. Chi tiết bảng mã giảm giá

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã giảm giá
2	name	varchar(50)		Tên giảm giá
3	code	varchar(30)		Mã
4	from	datetime		Ngày bắt đầu
5	to	datetime		Ngày kết thúc
6	desc_by	varchar(20)		Giảm giá theo
7	value	double		Giá trị giảm
8	created_at	timestamp		Ngày tạo
9	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **discounts** (giảm giá)

Bảng 4-8. Chi tiết bảng giảm giá

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã giảm giá
2	customer_category_id	bigint	Khóa ngoại	Mã loại khách hàng
3	type	varchar(20)		Loại giảm giá
4	value	double		Giá trị giảm

5	created_at	timestamp		Ngày tạo
6	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **products** (sản phẩm)

Bảng 4-9. Chi tiết bảng sản phẩm

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã sản phẩm
2	name	varchar(255)		Tên sản phẩm
3	description	varchar(255)		Mô tả
4	long_description	text		Mô tả chi tiết
5	image	varchar(255)		Hình ảnh
6	thumb	varchar(255)		Thumbnail
7	stock	int		Số lượng sản phẩm
8	is_hot	int		Là sản phẩm hot
9	is_new	int		Là sản phẩm mới
10	is_unlimited	int		Không giới hạn số lượng
11	has_variant	int		Có biến thể
12	price	int		Giá
13	sell_price	int		Giá bán

14	category_id	bigint		Mã danh mục
15	supplier_id	bigint		Mã nhà cung cấp
16	brand_id	bigint		Mã thương hiệu
17	created_at	timestamp		Ngày tạo
18	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **product_colors** (màu sắc sản phẩm)

Bảng 4-10. Chi tiết bảng màu sắc sản phẩm

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	product_id	bigint		Mã sản phẩm
2	color_id	bigint		Mã màu sắc

Mô tả bảng: **product_sizes** (kích cỡ sản phẩm)

Bảng 4-11. Mô tả bảng kích cỡ sản phẩm

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	product_id	Bigint		Mã sản phẩm
2	size_id	bigint		Mã kích cỡ

Mô tả bảng: **skus** (biến thể sản phẩm)

Bảng 4-12. Mô tả bảng biến thể sản phẩm

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint		Mã sản phẩm

2	sku	varchar(30)		Mã biến thể
3	product_id	bigint		Mã sản phẩm
4	color_id	bigint		Mã màu sắc
5	size_id	bigint		Mã kích cỡ
6	price	int		Giá
7	created_at	timestamp		Ngày tạo
8	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **tags**

Bảng 4-13. Chi tiết bảng tags

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã hashtag
2	name	varchar(50)		Tên hashtag
3	created_at	timestamp		Ngày tạo
4	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **product_contacts** (liên hệ sản phẩm)

Bảng 4-14. Chi tiết bảng liên hệ sản phẩm

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã liên hệ sản phẩm
2	product_id	bigint	Khóa ngoại	Mã sản phẩm

3	name	varchar(100)		Tên liên hệ
4	phone	varchar(20)		Số điện thoại
5	email	varchar(100)		Email
6	message	varchar(255)		Nội dung
7	created_at	timestamp		Ngày tạo
8	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **product_tags** (tag sản phẩm)

Bảng 4-15. Chi tiết bảng tag sản phẩm

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	product_id	bigint	Khóa chính, Khóa ngoại	Mã sản phẩm
2	tag_id	bigint	Khóa chính, Khóa ngoại	Mã hashtag

Mô tả bảng: **payment_types** (phương thức thanh toán)

Bảng 4-16. Chi tiết bảng phương thức thanh toán

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã loại thanh toán
2	name	varchar(100)		Tên loại thanh toán
3	created_at	timestamp		Ngày tạo

4	updated_at	timestamp		Ngày cập nhật
---	------------	-----------	--	---------------

Mô tả bảng: **order_trackings** (theo dõi đơn hàng)

Bảng 4-17. Chi tiết bảng theo dõi đơn hàng

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã xem thông tin đơn hàng
2	phone	varchar(20)		Số điện thoại
3	expired_at	datetime		Thời gian hết hạn
4	is_verified	int		Mã đã xác nhận hay chưa
5	created_at	timestamp		Ngày tạo
6	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **orders** (đơn hàng)

Bảng 4-18. Chi tiết bảng đơn hàng

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã đơn hàng
2	order_number	varchar(20)		Mã đơn hàng
3	order_date	datetime		Ngày đặt
4	ship_date	datetime		Ngày vận chuyển

5	sub_total	int		Thành tiền
6	discount	int		Tổng giảm
7	fee	int		Phí vận chuyển
8	total	int		Tổng tiền
10	customer_id	bigint	Khóa ngoại	Mã khách hàng
11	coupon_id	bigint	Khóa ngoại	Mã giảm giá
12	payment_type_id	bigint	Khóa ngoại	Mã loại thanh toán
13	province_id	int		Mã tỉnh
14	district_id	int		Mã huyện
15	ward_id	int		Mã xã
16	address	varchar(255)		
16	note	varchar(500)		Ghi chú
17	created_at	timestamp		Ngày tạo
18	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **detail_orders** (chi tiết đơn hàng)

Bảng 4-19. Chi tiết bảng chi tiết đơn hàng

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã chi tiết đơn hàng

2	order_id	bigint		Mã đơn hàng
3	product_id	bigint		Mã sản phẩm
4	sku_id	bigint		Mã biến thể
5	name	varchar(255)		Tên sản phẩm
6	price	int		Giá sản phẩm
7	quantity	int		Số lượng
8	total	int		Thành tiền
9	created_at	timestamp		Ngày tạo
10	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **post_categories** (danh mục bài viết)

Bảng 4-20. Chi tiết bảng danh mục bài viết

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
1	id	bigint	Khóa chính	Mã loại bài viết
2	name	varchar(100)		Tên loại bài viết
3	created_at	timestamp		Ngày tạo
4	updated_at	timestamp		Ngày cập nhật

Mô tả bảng: **posts** (bài viết)

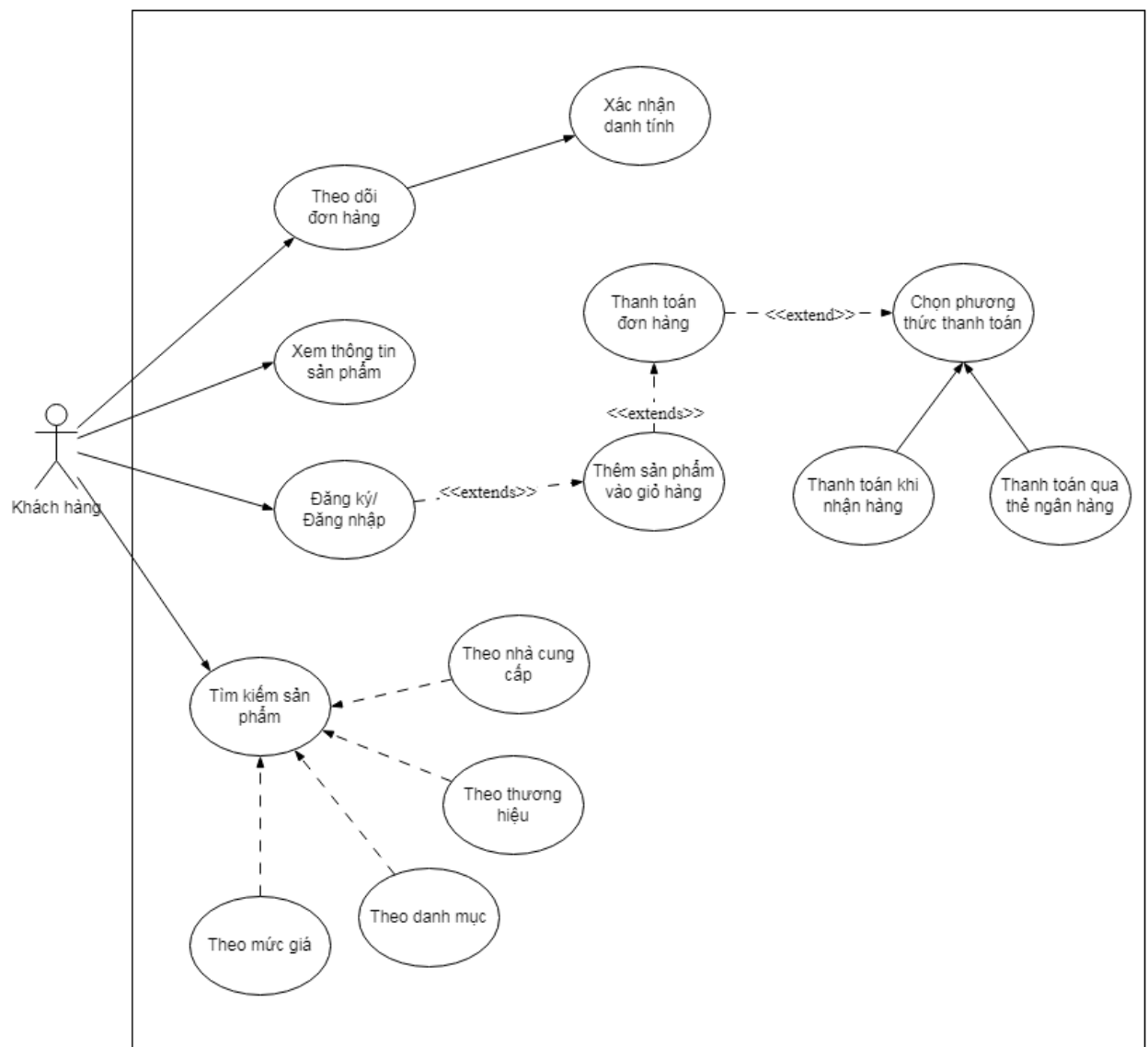
Bảng 4-21. Chi tiết bảng bài viết

STT	Thuộc tính	Kiểu dữ liệu	Miền giá trị	Diễn giải
-----	------------	--------------	--------------	-----------

1	id	bigint	Khóa chính	Mã bài viết
2	post_category_id	bigint	Khóa ngoại	Mã loại danh mục bài viết
3	title	bigint		Mã đơn hàng
4	thumbnail	varchar(255)		Ảnh thumbnail
5	preview	varchar(100)		Nội dung xem trước
6	content	text		Nội dung chi tiết
7	seo_title	varchar(255)		Tiêu đề SEO
8	meta_keyword	varchar(255)		Meta keyword
9	meta_description	text		Mô tả cho thẻ meta
10	view	int		Số lượng xem
11	created_at	timestamp		Ngày tạo
12	updated_at	timestamp		Ngày cập nhật

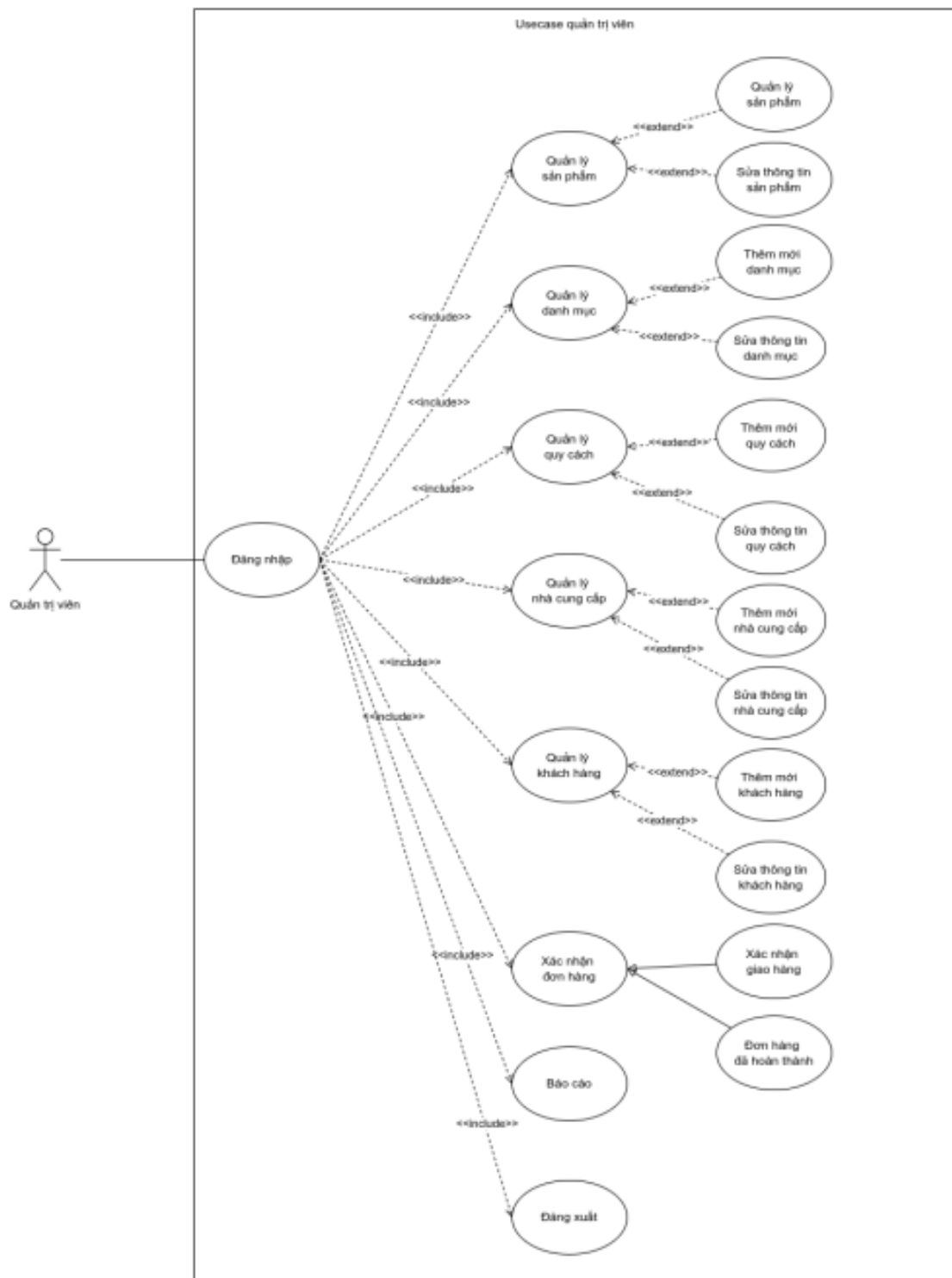
4.7. Mô hình xử lý

4.7.1. Usecase khách hàng



Hình 4-2. Usecase khách hàng

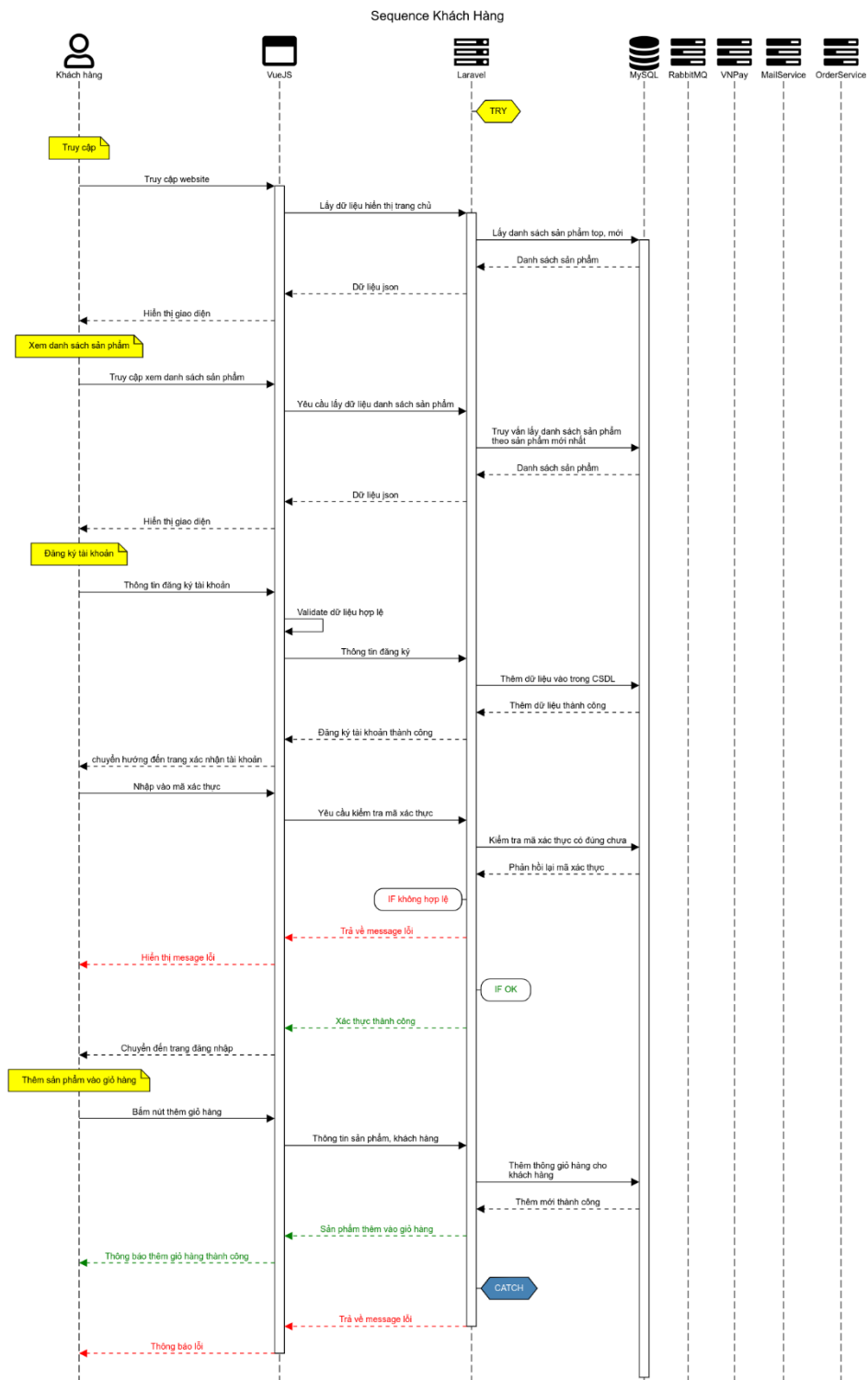
4.7.2. Usecase quản trị viên



Hình 4-3. Usecase quản trị viên

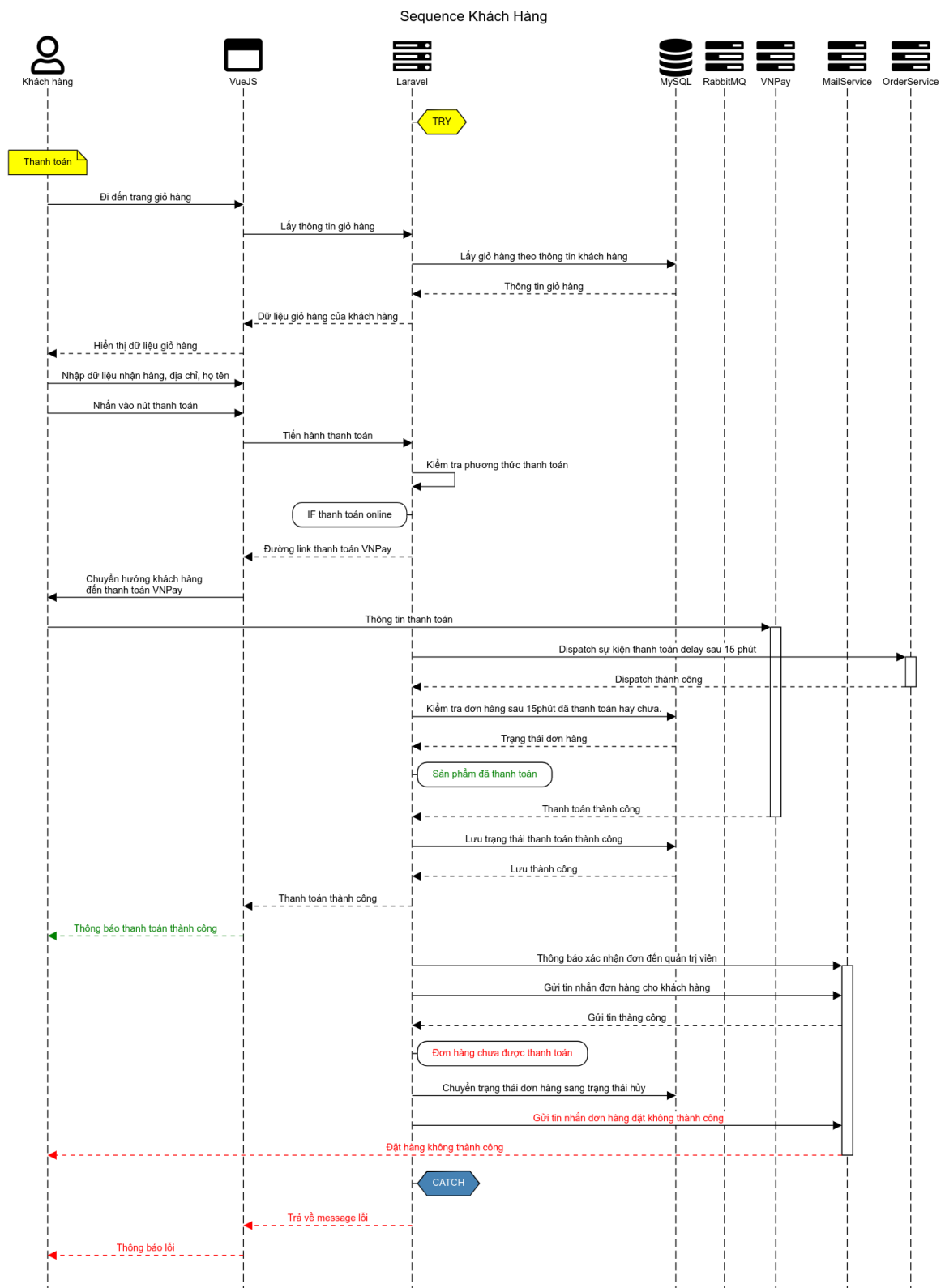
4.8. Mô hình tuần tự

Mô hình tuần tự hoạt động khách hàng



Hình 4-4. Mô hình tuần tự hoạt động mua hàng của khách hàng

Mô hình tuần tự thanh toán



Hình 4-5. Mô hình tuần tự hoạt động thanh toán đơn hàng

4.9. Mô tả hoạt động

4.9.1. Mô tả hoạt động của khách hàng

Khách hàng khi truy cập vào website mua hàng thì đầu tiên sẽ đi đến trang chủ cửa hàng. Tại màn hình trang chủ sẽ hiển thị thông tin slide quảng cáo của cửa hàng, phía bên dưới slide sẽ có danh sách sản phẩm bán chạy nhiều nhất và sản phẩm mới nhất tại cửa hàng. Khi mà khách hàng kéo đến gần cuối cùng ở trang chủ sẽ có một nút xem tất cả. Khi mà khách hàng bấm vào xem tất cả thì hệ thống sẽ lập tức chuyển hướng khách hàng đến màn hình xem danh sách sản phẩm.

Tại màn hình danh sách sản phẩm, hệ thống sẽ hiển thị tất cả sản phẩm mà cửa hàng đang bán do sản phẩm bán ra số lượng rất nhiều nên để tăng trải nghiệm chất lượng cho người dùng thì sản phẩm hiển ra sẽ được phân trang. Tại màn hình danh sách sản phẩm này khách hàng có thể lọc sản phẩm dựa theo nhiều tiêu chí như là sản phẩm bán nhiều nhất, theo danh mục, thương hiệu,... Khi khách hàng muốn xem chi tiết một đơn hàng nào đó thì có thể click trực tiếp vào hình ảnh sản phẩm. Hệ thống sẽ chuyển hướng trang đến trang chi tiết sản phẩm đó.

Tại màn hình chi tiết sản phẩm hệ thống sẽ hiển ra tất cả các thông tin cần thiết để khách hàng có quyết định mua hàng hay không. Phía dưới màn hình chi tiết sản phẩm sẽ là một số sản phẩm được hệ thống gợi ý để khách hàng có thể lựa chọn. Khi mà khách hàng lựa chọn được sản phẩm rồi thì tại màn hình chi tiết sản phẩm này sẽ có ô nhập số lượng sản phẩm. Khách hàng có thể tăng giảm số lượng sản phẩm bằng cách nhấn vào 2 nút kế bên ô nhập số lượng sản phẩm. Sau khi đã thêm được số lượng sản phẩm muốn mua rồi nếu khách không muốn mua tiếp sản phẩm khác thì có thể nhấn vào nút mua ngay, nếu mà khách chưa đăng nhập thì hệ thống sẽ chuyển hướng trực tiếp trên trang đăng nhập và sau khi đăng nhập hoàn tất sẽ được chuyển hướng trực tiếp đến trang giỏ hàng. Còn nếu khách hàng bấm vào thêm giỏ hàng thì vẫn sẽ ở lại trang hiển tại. Khi mà hoàn tất quá trình chọn hàng rồi, khách hàng tiếp tục vào trong mục giỏ hàng bằng cách nhấn vào nút có biểu tượng giỏ hàng trên thanh menu bên phải thì sẽ được chuyển hướng đến trang giỏ hàng.

Tại trang giỏ hàng khách hàng có thêm tăng giảm số lượng sản phẩm mà đã được thêm vào trong giỏ hàng trước đó. Khách hàng cũng có thể xóa sản phẩm mà mình không muốn mua nữa. Ở phía bên dưới trang giỏ hàng sẽ có ô nhập thông tin mã giảm giá, khi khách hàng nhận được mã giảm giá từ chương trình của cửa hàng thì có thể nhập vào để thêm giảm giá cho đơn hàng đó, hệ thống sẽ kiểm tra mã đơn hàng đó còn thời hạn hợp lệ hay không, nếu đã hợp lệ thì hệ thống sẽ trừ tiền trực tiếp vào trong đơn hàng đó. Sau khi đã hoàn tất quá trình thay đổi trong giỏ hàng. Thì khách hàng ấn vào nút thanh toán để chuyển đến trang thanh toán.

Tại trang thanh toán nếu khách hàng chưa cập nhật địa chỉ thì hệ thống sẽ yêu cầu khách hàng nhập địa chỉ để giao hàng. Có hai phương thức thanh toán để khách hàng có thể thanh toán đó là thanh toán khi nhận hàng và thanh toán online. Khi mà chọn thanh toán khi nhận hàng thì khách hàng cần phải đợi sự xác nhận từ phía bên cửa hàng. Còn thanh toán online thì hệ thống sẽ mặc định đơn hàng đó sẽ là đơn hàng đã được xác nhận từ phía cửa hàng. Khi mà dùng phương thức thanh toán online thì khi mà khách hàng nhấn vào nút thanh toán hệ thống sẽ chuyển hướng người dùng đến trang thanh toán. Sau khi đã hoàn tất quá trình thanh toán thì hệ thống sẽ chuyển người dùng đến trang giao hàng thành công và email về đơn hàng sẽ được gửi đến mail đã đăng ký trước đó của khách hàng.

4.9.2. Mô tả hoạt động của quản trị viên

Lần đầu vào trang quản trị thì hệ thống sẽ yêu cầu quản trị viên đăng nhập vào hệ thống, sau khi đã kiểm tra hợp lệ thì hệ thống sẽ chuyển hướng đầu tiên đến trang quản trị sản phẩm.

Tại trang quản trị sản phẩm sẽ hiển thị một số thông tin cơ bản như, số lượng khách hàng trong hệ thống, số lượng đơn hàng hoàn thành và tổng doanh thu trong tháng. Phía dưới sẽ là biểu đồ về đơn đặt hàng của hệ thống. Tại đây hệ thống sẽ tự động cập nhật 2 giây 1 lần bao gồm tổng số đơn, số đơn đã hoàn thành, số đơn thất bại và tỉ lệ sản phẩm thất bại, hoàn thành. Ngoài ra trang quản trị cũng hiển thị 10 đơn hàng gần nhất để người quản trị viên có thể nhanh chóng biết được các đơn hàng để đi đến xác nhận.

Tại trang quản trị hệ thống cũng cho phép quản trị viên có thể thêm thông tin về sản phẩm, nhà cung cấp, danh mục sản phẩm,...

Hệ thống sẽ phân loại khách hàng dựa theo tổng số tiền mua hàng hoặc khách hàng cũng có thể được thiết lập và nhóm khách hàng một cách thủ công. Mỗi nhóm khách hàng sẽ có ưu đãi riêng, ưu đãi này có thể là giảm giá dựa trên phần trăm đơn hàng hoặc giảm giá theo một giá trị cố định. Hệ thống cũng thể tạo mã giảm giá để gửi đến khách hàng, khách hàng nhận được mã giảm giá này có thể giảm giá trực tiếp vào trong đơn hàng.

Ngoài ra hệ thống cũng có cho phép người quản trị quản lý các đơn hàng sản phẩm, in hóa đơn sản phẩm gửi đến cho khách hàng.

4.10. Cài đặt và thử nghiệm

4.10.1. Thiết kế giao diện

Giao diện chung

Header

Hiển thị các danh mục menu:

- Trang chủ: Thông tin quan trọng của cửa hàng muốn truyền tải đến khách hàng.
- Shop: Giới thiệu thông tin về cửa hàng.
- Sản phẩm: Liệt kê các sản phẩm đang được đăng bán tại cửa hàng.
- Giới thiệu: Các bài blog và bài giới thiệu sản phẩm của cửa hàng.
- Mua ngay: Di chuyển nhanh đến giỏ hàng thanh toán.

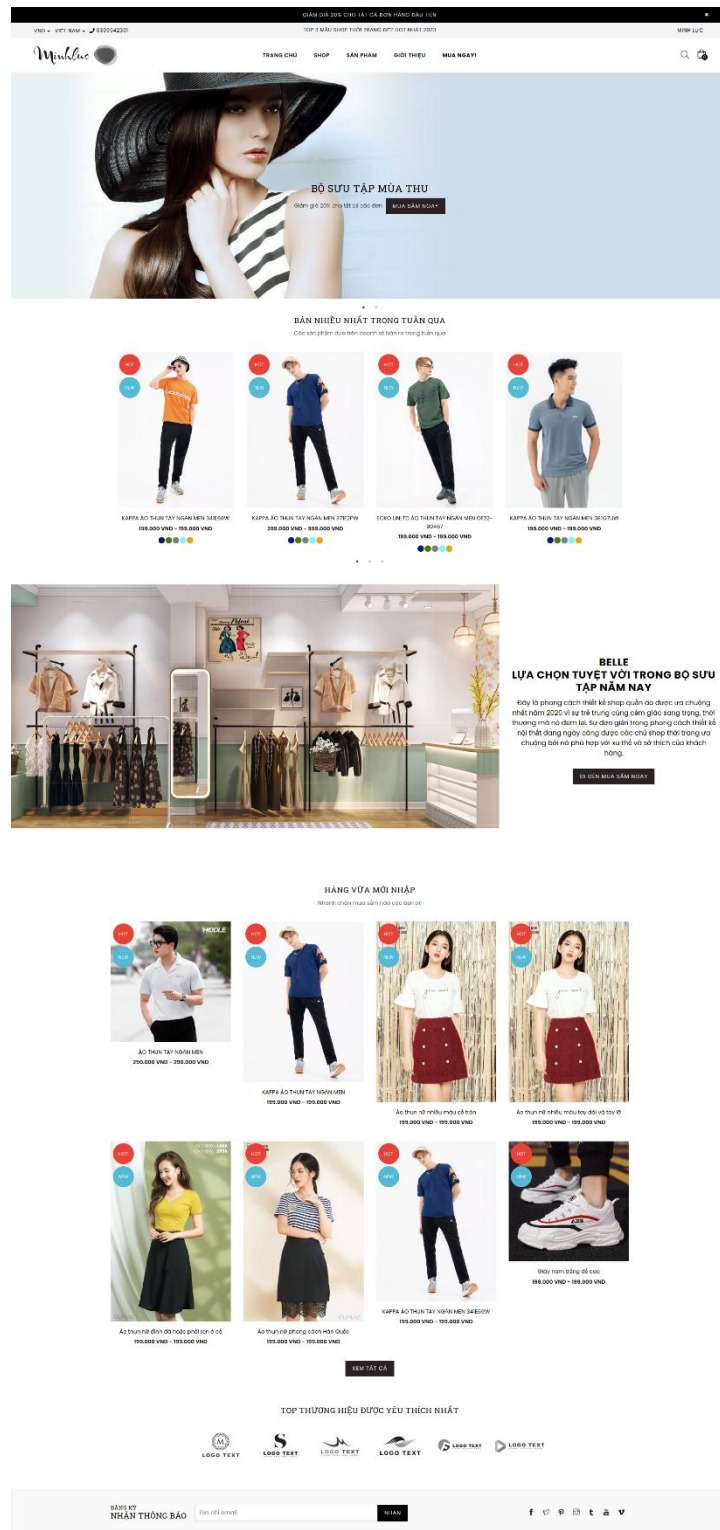
Footer:

- Hiển thị danh sách một số nhãn hàng đang có ở trong cửa hàng.
- Hiển thị các truy cập nhanh, thông tin của cửa hàng, dịch vụ cửa hàng.

Giao diện màn hình trang chủ

- Ở màn hình giao diện sẽ hiển thị danh sách banner về một số chương trình của cửa hàng.
- Hiển thị top 8 sản phẩm được bán chạy nhất trong cửa hàng dựa theo số lượng sản phẩm được bán ra có thể trượt qua lại bằng thanh slider.

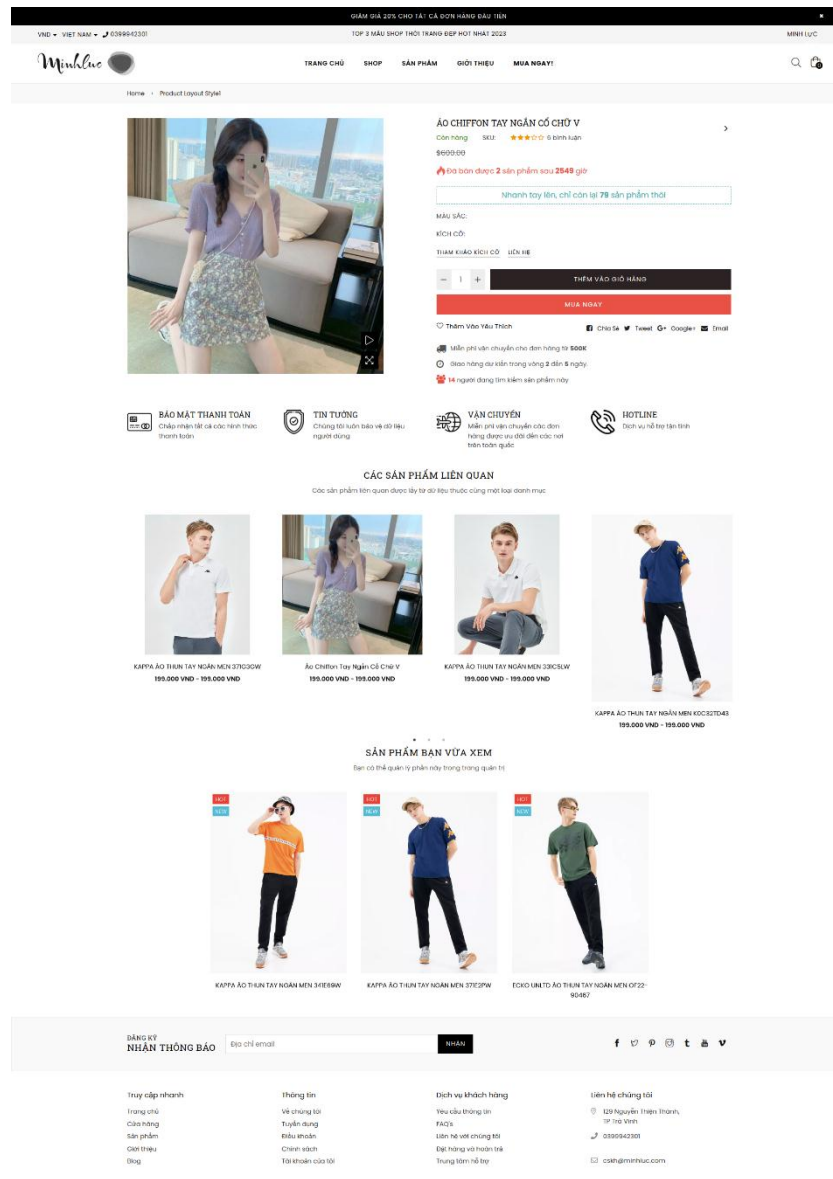
- Hiện thị 8 sản phẩm mới nhất được bán ở trong cửa hàng khi bấm vào nút xem tất cả thì sẽ di chuyển đến màn hình danh sách sản phẩm.
- Khi người dùng nhấn vào ảnh của sản phẩm thì sẽ đi đến chi tiết thông tin về sản phẩm.



Hình 4-8. Giao diện trang chủ

Giao diện chi tiết sản phẩm

- Bên trái sẽ hiển thị hình ảnh sản phẩm.
- Bên phải sẽ hiển thị thông tin về sản phẩm như mức giá, màu sắc, kích cỡ, giá cả sản phẩm.
- Có thể thay đổi số lượng sản phẩm để thêm vào trong giỏ hàng.
- Khi ấn vào nút thêm vào giỏ hàng thì hệ thống sẽ lấy thông tin số lượng ở trong ô số lượng sản phẩm để thêm vào trong giỏ hàng.
- Còn khi ấn vào nút mua ngay thì hệ thống cũng sẽ lấy thông tin số lượng và sau đó thêm sản phẩm vào trong giỏ hàng và tiến hành di chuyển đến màn hình giỏ hàng.

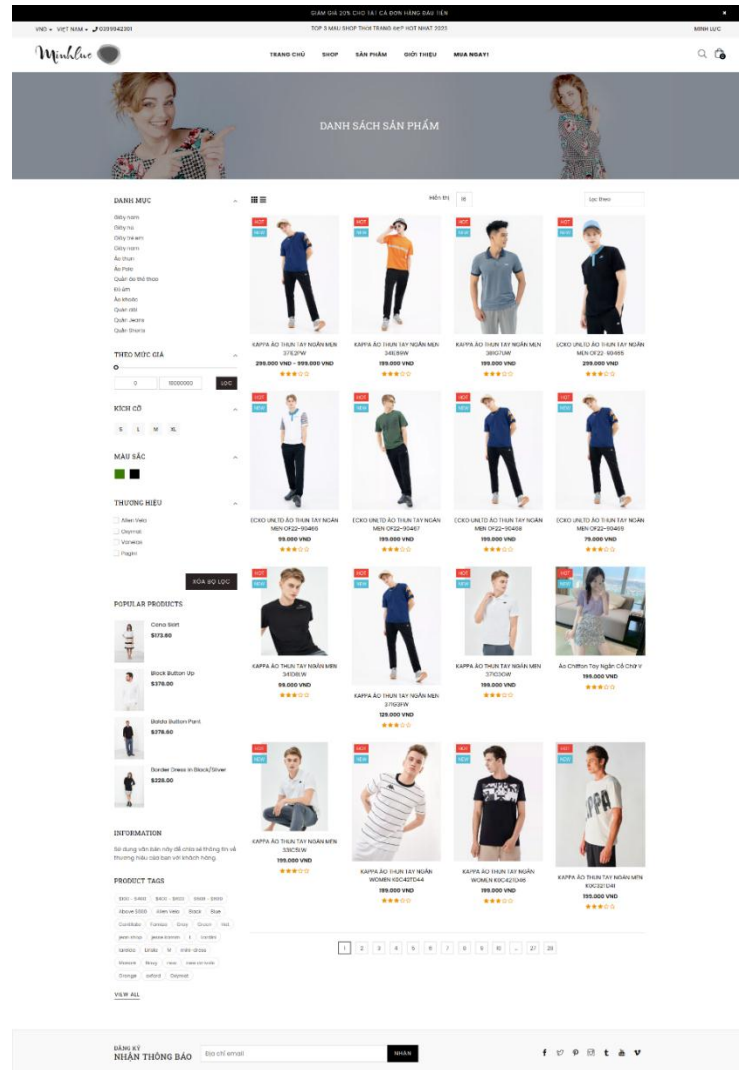


Hình 4-9. Giao diện chi tiết sản phẩm.

Giao diện danh sách sản phẩm

- Thanh bên trái hiển thị các bộ lọc để người dùng có thể lọc , bao gồm lọc theo danh mục, màu sắc, kích cỡ, thương hiệu,...
- Bên phải sẽ hiển thị danh sách sản phẩm và phân trang sản phẩm
- Đối với mỗi sản phẩm khi mà rê chuột vào trong từng sản phẩm sẽ hiển thị một số nút như thêm vào giỏ hàng. Khi mà bấm vào thêm vào giỏ hàng thì hệ thống sẽ thêm một sản phẩm được chọn vào trong giỏ hàng. Ngoài ra còn có một số nút

như là xem nhanh sản phẩm, khi mà click vào thì sẽ hiển thị một popup hiển thị thông tin nhanh về sản phẩm.

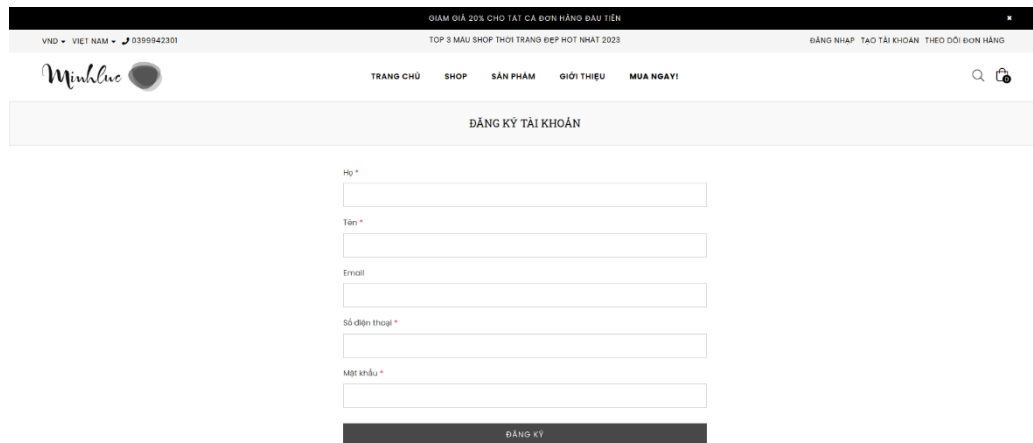


Hình 4-10. Giao diện danh sách sản phẩm

Giao diện đăng ký tài khoản

- Hiển thị form đăng ký tài khoản bao gồm một số trường: họ, tên, số điện thoại, email, mật khẩu.
- Sau khi đăng ký tài khoản hoàn tất thì có một mã tin nhắn sẽ được gửi đến khách hàng để mà tiến hành xác thực tài khoản. Khách hàng dùng mã nhận được điền

vào trong form xác thực tài khoản. Nếu thành công thì sẽ di chuyển đến màn hình quản lý tài khoản. Còn thất bại sẽ thông báo lỗi.

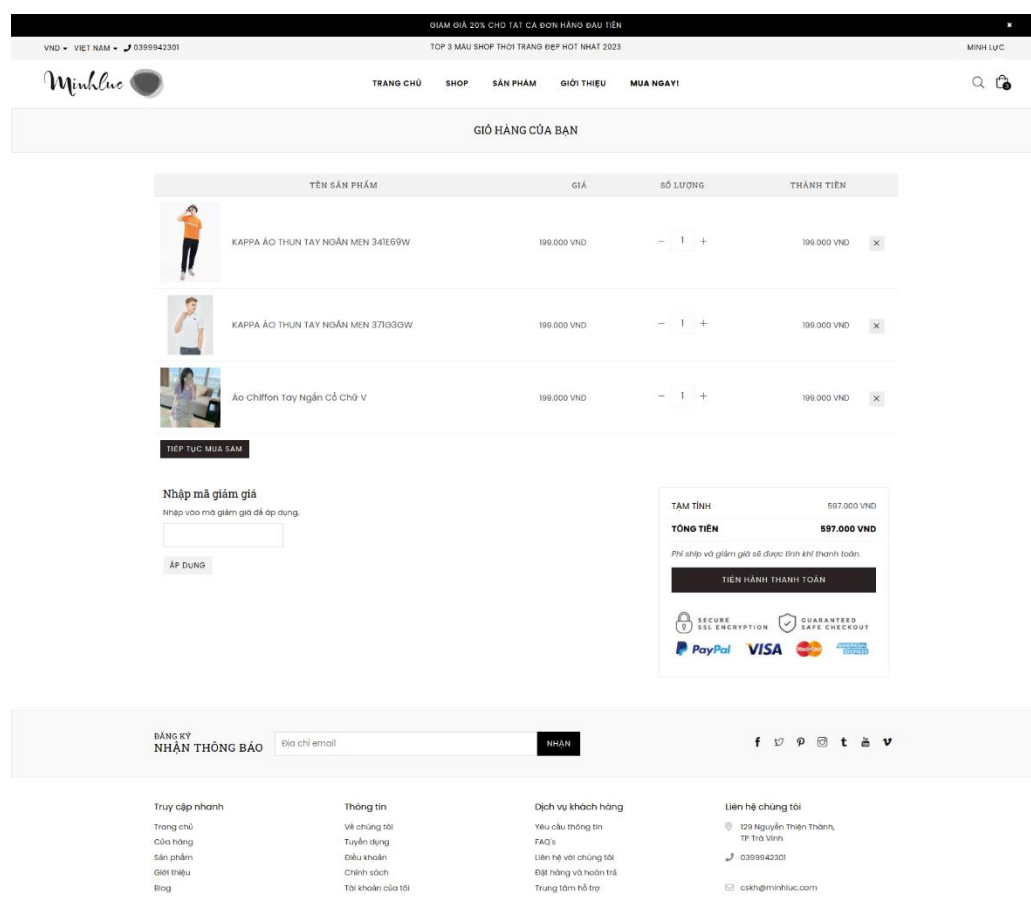


The screenshot shows the registration page of the Minh Luc website. At the top, there is a black header with white text: "GIẢM GIÁ 20% CHO TẤT CẢ ĐƠN HÀNG ĐẦU TIÊN". Below this is a light gray navigation bar with the following elements: "VND • VIET NAM • 0398942301" on the left, "TOP 3 MẪU SHOP THỜI TRANG ĐẸP HÓT NHẤT 2023" in the center, and "ĐĂNG NHẬP TẠO TÀI KHOẢN THEO DÕI ĐƠN HÀNG" on the right. The Minh Luc logo is on the left, and navigation links "TRANG CHỦ", "SHOP", "SẢN PHẨM", "GIỚI THIỆU", and "MUA NGAY!" are in the center. A search icon is on the right. Below the navigation bar is a light gray banner with the text "ĐĂNG KÝ TÀI KHOẢN". The registration form consists of several input fields with red asterisks indicating required fields: "Họ *", "Tên *", "Email", "Số điện thoại *", and "Mật khẩu *". A dark gray button labeled "ĐĂNG KÝ" is at the bottom of the form.

Hình 4-11. Giao diện đăng ký khách hàng

Giao diện giỏ hàng

- Hiện thị các sản phẩm đã được khách hàng thêm vào trong giỏ hàng
- Khách hàng có thể thay đổi số lượng sản phẩm trong giỏ hàng và cũng có thể xóa sản phẩm .
- Khi thay đổi số lượng sản phẩm thì hệ thống sẽ tự động tính toán lại tổng tiền của sản phẩm.



Hình 4-12. Giao diện giỏ hàng

Giao diện thanh toán

Tại giao diện màn hình thanh toán bên trái sẽ hiển thị thông tin giao hàng: Tên, số điện thoại, email, địa chỉ giao hàng, thông tin ghi chú.

- Khi khách hàng chọn thông tin địa chỉ thì hệ thống sẽ tính toán phí vận chuyển đơn hàng dựa theo phí vận chuyển của giao hàng nhanh api.
- Đối với khách hàng thuộc loại khách hàng được giảm giá thì hệ thống sẽ áp dụng giảm giá đó trực tiếp cho khách hàng.
- Có hai phương thức để thanh toán tại màn hình thanh toán: thanh toán khi nhận hàng và thanh toán online. Đối với thanh toán online thì khi mà nhấn vào nút thanh toán hệ thống sẽ chuyển hướng đến trang thanh toán của VNPAY. Nếu thanh toán hoàn tất thì hệ thống sẽ di chuyển đến màn hình đặt hàng thành công.

VND • VIỆT NAM

+84399942301

GIẢM GIÁ 20% CHO TẤT CẢ ĐƠN HÀNG ĐẦU TIÊN

TOP 3 MÀU SHOP THỜI TRANG ĐẸP HOT NHẤT 2023

MÌNH LỰC

MinhLuc

TRANO CHÙSHOPSÁN PHẨMGIỚI THIỆUMUA NGAY!

🔍🛒

THANH TOÁN ĐƠN HÀNG

ĐĂNG NHẬP TÀI KHOẢN TẠI ĐÂY ? NHẤN VÀO ĐÂY

THÔNG TIN THANH TOÁN

Họ *

Mình

Tên *

Lực

Email *

customer@minhluc.com

Số điện thoại *

+84399942311

Tỉnh/Thành phố *

Trà Vinh

Quận/Huyện *

Huyện Duyên Hải

Phường/Xã *

Xã Ngũ Lạc

Ghi chú chủ đơn hàng *

LƯU LẠI

BẠN CÓ MÃ GIẢM GIÁ ? ÁN VÀO ĐÂY ĐỂ ÁP DỤNG MÃ GIẢM GIÁ

THÔNG TIN ĐƠN HÀNG

Tên sản phẩm	Giá	SL	Thành tiền
KAPPA Áo thun Tay Ngắn Men 341E59W	199.000 VND	1	199.000 VND
KAPPA Áo thun Tay Ngắn Men 370G3OW	199.000 VND	1	199.000 VND
Áo Chiffon Tay Ngắn Cổ Chữ V	199.000 VND	1	199.000 VND
Thành tiền			597.000 VND
Phi vận chuyển			
Giảm giá			
Tổng thanh toán			597.000 VND

PHƯƠNG THỨC THANH TOÁN

THANH TOÁN KHI NHẬN HÀNG

THANH TOÁN ONLINE

☒ Thanh toán khi nhận hàng

☐ Thanh toán online

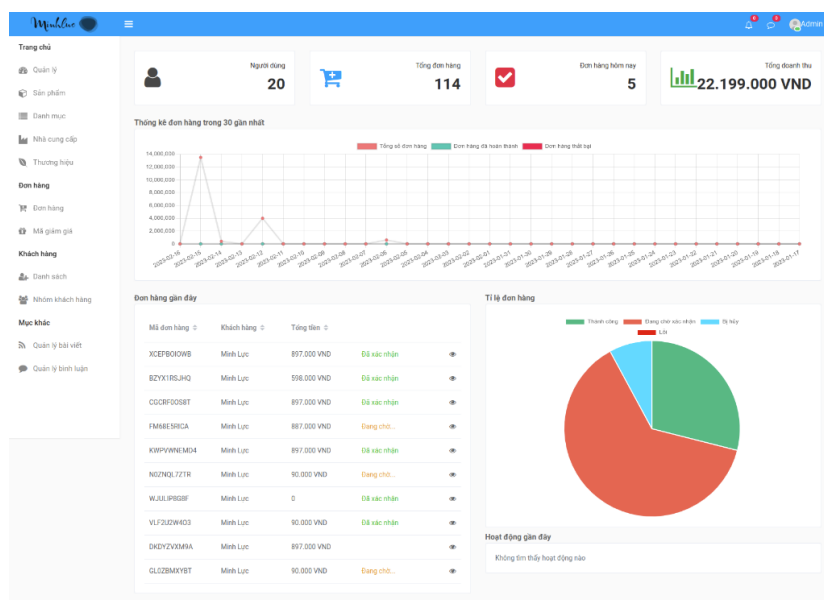
THANH TOÁN

BẠN CÒ MÃ GIẢM GIÁ ? AN VÀO ĐÂY ĐỂ ÁP DỤNG MÃ GIẢM GIÁ				
THÔNG TIN ĐƠN HÀNG				
Tên sản phẩm	Giá	SL	Thành tiền	
KAPPA Áo THUN TAY NGẮN MEN 34IE69W	199.000 VND	1	199.000 VND	
KAPPA Áo THUN TAY NGẮN MEN 37IG3OW	199.000 VND	1	199.000 VND	
Áo Chiffon Tray Ngắn Cổ Chữ V	199.000 VND	1	199.000 VND	
			Thành tiền	597.000 VND
			Phí vận chuyển	
			Giảm giá	
			Tổng thanh toán	597.000 VND

Hình 4-13. Giao diện thanh toán

Giao diện quản trị

- Hiện thị thông tin số lượng người dùng, tổng đơn hàng, tổng số đơn hàng của ngày hôm nay, doanh thu trong tháng này.
- Thống kê đơn hàng trong vòng 30 ngày gần nhất. Hệ thống sẽ tự động cập nhật lại thông tin trong vòng 2 giây.
- Hiện thị danh sách đơn hàng được thêm vào gần nhất và tỉ lệ đơn hàng bao gồm tỉ lệ đơn hàng hoàn thành, đơn hàng bị hủy và đơn hàng chưa được xác nhận.



Hình 4-14. Giao diện quản trị

Giao diện trang quản trị sản phẩm

- Hiện thị danh sách sản phẩm trong cửa hàng
- Thêm mới sản phẩm vào trong cửa hàng
- Đối với sản phẩm trong cửa hàng thì sẽ có hai loại sản phẩm là sản phẩm không có biến thể thì hệ thống sẽ yêu cầu quản trị viên của cửa hàng nhập giá sản phẩm và giá sản phẩm được ưu đãi bán ra.
- Đối với sản phẩm có biến thể thì hệ thống hiện tại có hai loại biến thể là biến thể về màu sắc và biến thể về hình ảnh. Ví dụ khi chọn biến thể màu sắc là màu xanh và màu vàng và biến thể về kích cỡ là S, M, L thì hệ thống sẽ tạo ra 6 loại biến thể cho sản phẩm là: Xanh/S, Xanh/M, Xanh/L, Vàng/S, Vàng/M, Vàng/L. Đối với từng loại sản phẩm quản trị viên phải nhập vào giá của từng loại biến thể.

Tên sản phẩm

ÁO THUN TAY NGẮN MEN

Mô tả

Polo Just Men họa tiết hoa lá hai màu Đen và trắng đẹp xuất sắc. Chất vải cao cấp cùng họa tiết in sắc nét, Form slim fit ôm vừa thoải mái

Áo thun Nike Oxymet

☐ Sản phẩm hot ☐ Sản phẩm mới ☐ Số lượng không giới hạn

Tags: thoitrang hienluc

Thêm mới

Biến thể sản phẩm

Màu sắc Kích cỡ

Màu sắc	Kích cỡ
Green	M L S
Black	M L S

Green/M	290000
Green/L	290000
Green/S	290000
Black/M	290000
Black/L	290000
Black/S	290000

Hình 4-15. Giao diện thêm sản phẩm

Giao diện danh sách loại khách hàng

- Hiện thị danh sách loại khách hàng
- Một loại khách hàng có thể thuộc vào trong một loại khách hàng lớn. Mã ưu đãi giảm giá sẽ được áp dụng với loại khách hàng và nếu có thêm mã giảm giá thì khách hàng cũng sẽ được tính cộng vào.

Tên	Loại ưu đãi	Giá trị	Ngày bắt đầu	
VIP	Phần trăm	1%	2023-02-07 22:52:42	
Hạng Đồng	Phần trăm	6%	2022-11-10 17:00:00	
Hạng Bạc	Phần trăm	7%	2022-11-10 17:00:00	
Hạng Vàng	Phần trăm	8%	2022-11-10 17:00:00	
Nhóm miền Nam	Phần trăm	3%	2022-11-10 17:00:00	
Khu vực Bình Dương	Giá trị	100000	2022-11-10 17:00:00	
Khu vực Tây Ninh	Phần trăm	5%	2022-11-10 17:00:00	
Nhóm VIP	Giá trị	10000	2022-11-10 17:00:00	
Gói thưởng	Giá trị	15000	2022-11-10 17:00:00	
Gói ưu đãi	Phần trăm	5%	2023-02-06 15:29:59	

Hình 4-16. Giao diện nhóm khách hàng

Giao diện tạo mã giảm giá

- Khi tạo mã giảm giá thì các quản trị viên phải thiết lập thông tin sản phẩm được áp dụng mã giảm giá.

Tên coupon	Mã coupon
Mã quà tặng tết	TET2023

Giảm giá theo	Giá trị	Trạng thái
Phần trăm	10	Bật

Thêm

Hình 4-6. Giao diện thêm mã giảm giá

4.10.2. Hướng dẫn cài đặt

Chuẩn bị:

- Cài đặt PHP 8.1 theo hướng dẫn trang chủ: <https://www.php.net/releases/8.1/en.php>
- Cài đặt composer theo hướng dẫn trên trang chủ: <https://getcomposer.org/download/>
- Cài đặt NodeJS theo hướng dẫn trên trang chủ: <https://nodejs.org/en/download/>
- Cài đặt docker theo hướng dẫn trên trang chủ: <https://www.docker.com>

Cài đặt:

Di chuyển đến trong thư mục dự án, trong thư mục dự án sẽ có 7 thư mục: rabbitmq, admin, frontend, mail, modules, coupon, order.

- Đối với thư mục rabbitmq thì sau khi đã cài đặt được docker và docker –compose thì ta sẽ tiến hành chạy lệnh “docker-compose up –d” ngay trong thư mục rabbitmq.
- Đối với thư mục admin, mail, coupon, order ta sẽ vào trong từng thư mục và tiến hành chạy lệnh “composer install” để cài đặt thư viện. Sau đó dùng lệnh php artisan serve để chạy ứng dụng laravel
- Đối với thư mục frontend sẽ có 2 thư mục con là cms và app tương ứng với mỗi thư mục ta sẽ dùng lệnh npm hoặc là yarn install để cài đặt thư viện. Và sau đó sẽ tiến hành chạy lệnh yarn serve để start ứng dụng lên.

Để có thể chạy queue cho từng service thì tương ứng với mỗi thư mục admin, mail, coupon, order ta sẽ tiến hành tạo supervisor.conf để tự động chạy lệnh php artisan queue:work mỗi khi khởi động.

CHƯƠNG 5: KẾT LUẬN

Kết quả đạt được

Về kiến thức:

- Thêm kiến thức mới về cách tiếp cận xây dựng hệ thống theo hướng sự kiện.
- Hiểu biết được cách hoạt động của hàng đợi tin nhắn RabbitMQ và ứng dụng để xây dựng hệ thống thiết kế theo hướng sự kiện.
- Học được cách xây dựng, thiết kế CSDL.

Về ứng dụng:

- Ứng dụng kiến thức hàng đợi tin nhắn để xây dựng website bán hàng với các chức năng cơ bản: mua hàng, thanh toán, gửi mail,... Do website được xây dựng bằng VueJS nên đảm bảo được tốc độ thực thi cao và mượt mà.

Về kỹ năng:

- Cải thiện được kỹ năng tư duy, phân tích thiết kế hệ thống, kỹ năng viết mã đạt tiêu chuẩn trong quá trình làm khóa luận.

Hạn chế

Do hạn chế về kiến thức cũng như công nghệ nên đề tài còn một số tính năng vẫn còn hoạt động chưa đúng theo ý muốn. Khả năng triển khai của hệ thống còn khá phức tạp nên khi deploy ứng dụng lên hosting còn gặp nhiều lỗi.

Hướng phát triển

Do kiến thức của bản thân có hạn trong khi kinh nghiệm phân tích thiết kế và vốn hiểu biết về lập trình web, lập trình với VueJS, Laravel framework còn hạn chế nên website còn nhiều điểm chưa phù hợp mong Thầy, cô đóng góp ý kiến để đề tài đề ngày một hoàn thiện hơn.

TÀI LIỆU THAM KHẢO