

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

NGUYỄN VĂN HỘI

LÊ MAI VĂN KHÁNH

KHÓA LUẬN TỐT NGHIỆP

**TÌM HIỂU KUBERNETES VÀ XÂY DỰNG ỨNG DỤNG
TỰ ĐỘNG CHỐT ĐƠN LIVESTREAM FACEBOOK**
Using Kubernetes to build auto-deal application for Facebook
livestream

KỸ SƯ NGÀNH CÔNG NGHỆ PHẦN MỀM

TP. HỒ CHÍ MINH, 2021

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

NGUYỄN VĂN HỘI – 16520456

LÊ MAI VĂN KHÁNH – 16520580

KHÓA LUẬN TỐT NGHIỆP

**TÌM HIỂU KUBERNETES VÀ XÂY DỰNG ỨNG DỤNG
TỰ ĐỘNG CHỐT ĐƠN LIVESTREAM FACEBOOK**
**Using Kubernetes to build auto-deal application for Facebook
livestream**

KỸ SƯ NGÀNH CÔNG NGHỆ PHẦN MỀM

GIẢNG VIÊN HƯỚNG DẪN
THS. HOÀNG VĂN HÀ

TP. HỒ CHÍ MINH, 2021

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số
ngày của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. – Chủ tịch.
2. – Thư ký.
3. – Ủy viên.
4. – Ủy viên.

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

TRƯỜNG ĐẠI HỌC

Độc Lập - Tự Do - Hạnh Phúc

CÔNG NGHỆ THÔNG TIN

TP. HCM, ngày 15 tháng 01 năm 2021

NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(CỦA CÁN BỘ HƯỚNG DẪN)

Tên khóa luận:

Tìm hiểu kubernetes và xây dựng ứng dụng tự động chốt đơn livestream facebook.

Nhóm sinh viên thực hiện:

NGUYỄN VĂN HỘI 16520456

LÊ MAI VĂN KHÁNH 16520580

Cán bộ hướng dẫn:

ThS. HOÀNG VĂN HÀ

Đánh giá Khóa luận:

1. Về cuốn báo cáo:

Số trang

Số chương

Số bảng số liệu

Số hình vẽ

Số tài liệu tham khảo

Sản phẩm

Một số nhận xét về hình thức cuốn báo cáo:

•

2. Về nội dung nghiên cứu:

•

•

3. Về chương trình ứng dụng:

•

4. Về thái độ làm việc của sinh viên:

•

Đánh giá chung:

Điểm từng sinh viên:

NGUYỄN VĂN HỘI: /10

LÊ MAI VĂN KHÁNH: /10

Người nhận xét

(Ký và ghi rõ họ tên)

Hoàng Văn Hà

TP. HCM, ngày 15 tháng 01 năm 2021

NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(CỦA CÁN BỘ PHẢN BIỆN)

Tên khóa luận:

Tìm hiểu kubernetes và xây dựng ứng dụng tự động chốt đơn livestream facebook.

Nhóm sinh viên thực hiện:

NGUYỄN VĂN HỘI 16520456

LÊ MAI VĂN KHÁNH 16520580

Cán bộ phản biện:

Đánh giá Khóa luận:

1. Về cuốn báo cáo:

Số trang

Số chương

Số bảng số liệu

Số hình vẽ

Số tài liệu tham khảo

Sản phẩm

Một số nhận xét về hình thức cuốn báo cáo:

2. Về nội dung nghiên cứu:

3. Về chương trình ứng dụng:

4. Về thái độ làm việc của sinh viên:

Đánh giá chung:

Điểm từng sinh viên:

NGUYỄN VĂN HỘI: /10

LÊ MAI VĂN KHÁNH: /10

Người nhận xét

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện tại khoa Công nghệ phần mềm trường Đại học Công nghệ Thông tin – ĐHQG TP.HCM chúng em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành Khóa luận tốt nghiệp của mình.

Để hoàn thành khóa luận này, chúng em xin gửi lời cảm ơn chân thành đến:

Ban Giám hiệu trường Đại học Công nghệ Thông tin – ĐHQG TP.HCM vì đã tạo điều kiện về cơ sở vật chất với hệ thống thư viện hiện đại, đa dạng các loại sách, tài liệu thuận lợi cho việc tìm kiếm, nghiên cứu thông tin.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Hoàng Văn Hà và thầy Nguyễn Công Hoan đã tận tình giúp đỡ, định hướng cách tư duy và cách làm việc khoa học. Đó là những góp ý hết sức quý báu không chỉ trong quá trình thực hiện luận văn này mà còn là hành trang tiếp bước cho chúng em trong quá trình học tập và lập nghiệp sau này.

Và cuối cùng, chúng em xin gửi lời cảm ơn đến gia đình, tất cả thầy cô trong khoa, bạn bè, tập thể lớp KTPM2016 là những người luôn sẵn sàng sẻ chia và giúp đỡ trong học tập và cuộc sống. Mong rằng, chúng ta sẽ mãi mãi gắn bó với nhau.

Trong quá trình làm khóa luận này chúng em không tránh khỏi được những sai sót, chúng em kính mong nhận được sự chỉ dẫn và góp ý của quý thầy cô để khóa luận được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn. Xin chúc những điều tốt đẹp nhất sẽ luôn đồng hành cùng mọi người.

Thành phố Hồ Chí Minh, 15 tháng 01 năm 2021

Sinh viên

NGUYỄN VĂN HỘI

LÊ MAI VĂN KHÁNH

MỤC LỤC

• TÓM TẮT KHÓA LUẬN	21
Chương 1. GIỚI THIỆU CHUNG	1
1.1. Lý do chọn đề tài	1
1.2. Tính năng mới/ khác biệt về chức năng của đề tài so với một số ứng dụng khác trên thị trường	2
1.3. Đối tượng nghiên cứu	3
1.4. Phạm vi nghiên cứu	3
1.5. Phương pháp nghiên cứu	4
Chương 2. KIẾN THỨC NỀN TẢNG	5
2.1. Tổng quan về kiến trúc Microservice	5
2.1.1. Khái niệm	5
2.1.2. Đặc điểm của Microservice	5
2.1.3. Ưu điểm và nhược điểm của Microservice	6
2.1.4. So sánh Microservice mà Monolithic	6
2.2. Tổng quan về Javascript	8
2.2.1. Khái niệm	8
2.2.2. Tại sao chọn Javascript để phát triển	8
2.3. Hệ quản trị cơ sở dữ liệu Postgresql	9
2.4. Tổng quan Nodejs và thư viện Koa	9
2.4.1. Giới thiệu Nodejs	9
2.4.2. Giới thiệu thư viện Koa	10
2.5. Tổng quan về Vuejs	10

2.5.1.	Giới thiệu về Vuejs	10
2.5.2.	Một số khái niệm và thành phần chính	11
2.6.	Giới thiệu về AWS S3	12
2.6.1.	AWS S3 là gì?	12
2.6.2.	Đặc điểm nổi bật của AWS S3	12
2.6.3.	Chi phí đăng ký sử dụng	13
2.7.	Tổng quan về Facebook API	14
2.7.1.	Một số khái niệm chính	14
2.7.1.1.	Định nghĩa về API	14
2.7.1.2.	Định nghĩa về Facebook API	14
2.7.1.3.	Định nghĩa về Facebook Graph API	15
2.7.2.	Cấu trúc Facebook Graph API	16
2.7.2.1.	HTTP	17
2.7.2.2.	Access Token	17
2.7.2.3.	URL lưu trữ	18
2.7.2.4.	Objects ID	18
2.7.3.	Facebook Pages API	18
2.8.	Tổng quan về Docker	19
2.8.1.	Docker là gì?	19
2.8.2.	Sự khác nhau giữa docker và virtual machine	19
2.8.3.	Tại sao nên dùng docker	19
2.8.4.	Kiến trúc của docker	20
2.8.5.	Các thành phần cơ bản của docker	21
2.9.	Tổng quan về Kubernetes	23

2.9.1.	Kubernetes là gì?	23
2.9.2.	Tại sao nên dùng kubernetes	23
2.9.3.	Sự khác nhau giữa Kubernetes và Docker Swarm	24
2.9.4.	Kiến trúc của kubernetes	28
2.9.5.	Các thành phần cơ bản của Kubernetes	30
2.9.6.	Cách cài đặt và hệ thống hỗ trợ	34
Chương 3.	XÂY DỰNG HỆ THỐNG	36
3.1.	Xây dựng kiến trúc hệ thống	36
3.1.1.	Xác định yêu cầu hệ thống	36
3.1.2.	Phân tích yêu cầu hệ thống	38
3.1.2.1.	Người dùng (cửa hàng)	38
3.1.2.2.	Quản trị viên:	39
3.2.	Phân tích thiết kế hệ thống	40
3.2.1.	Sơ đồ use case	40
3.2.1.1.	Sơ đồ use case	40
3.2.1.2.	Danh sách các actors	53
3.2.1.3.	Danh sách các use cases	54
3.2.2.	Sơ đồ lớp	56
3.2.3.	Phân tích và thiết kế CSDL	58
3.2.3.1.	Bảng Categories	60
3.2.3.2.	Bảng Products	61
3.2.3.3.	Bảng Variants	62
3.2.3.4.	Bảng Đơn vị	62
3.2.3.5.	Bảng Users	63

3.2.3.6. Bảng Orders	63
3.2.3.7. Bảng Shipping Information	64
3.2.3.8. Bảng Coupons	65
3.2.3.9. Bảng Stores	65
3.2.3.10. Bảng Notes	66
3.2.3.11. Bảng Syntaxes	66
3.2.3.12. Bảng Product Groups	67
3.2.3.13. Bảng Import Receipts	67
3.2.3.14. Bảng Import receipt details	68
3.2.3.15. Bảng Export receipts	68
3.2.3.16. Bảng Export receipt details	69
3.2.3.17. Bảng Customers	69
3.2.3.18. Bảng Customer Groups	70
3.2.3.19. Bảng Customer Blocks	70
3.2.3.20. Bảng Livestreams	71
3.2.3.21. Bảng Comment samples	71
3.2.3.22. Bảng Message samples	72
3.2.3.23. Bảng Logs	72
3.2.3.24. Bảng Settings	73
3.2.3.25. Bảng Order details	73
3.3 Thiết kế giao diện	74
3.1.1. Giao diện Client	74
3.1.2. Giao diện Admin	97
Chương 4. ÁP DỤNG TRIỀN KHAI ỨNG DỤNG VỚI KUBERNETES TRÊN DIGITALOCEAN	99

4.1.	Giới thiệu DigitalOcean	99
4.2.	Đăng ký tài khoản và tạo mới một kubernetes cluster	99
4.3.	Triển khai ứng dụng lên cụm cluster	102
Chương 5. KẾT LUẬN, HƯỚNG PHÁT TRIỂN		119
5.1.	Ưu điểm	119
5.2.	Nhược điểm	119
5.3.	Hướng phát triển	120

DANH MỤC HÌNH VẼ

Hình 2-1: Kiến trúc microservices.....	5
Hình 2-2: So sánh Microservice và Monolithic	7
Hình 2-3: Vòng đời Vuejs.....	11
Hình 2-4: Ảnh minh họa Facebook Graph API	15
Hình 2-5: Graph API Explorer.....	16
Hình 2-6: Kiến trúc của docker.....	20
Hình 2-7: Thành phần cơ bản của docker	22
Hình 2-8: Kiến trúc của Docker Swarm	25
Hình 2-9: So sánh Kubernetes với Docker Swarm	26
Hình 2-10: Kiến trúc của Kubernetes	28
Hình 2-11: Ảnh minh họa Pods	30
Hình 2-12: Ảnh minh họa ReplicaSet.....	31
Hình 2-13: Ảnh minh họa Service	32
Hình 2-14: Ảnh minh họa PV, PVC	33
Hình 2-15: Ảnh minh họa Ingress.....	34
Hình 3-1: Quy trình xử lý hàng hoá của FLAD	36
Hình 3-2: Biểu đồ use case mức tổng quát	40
Hình 3-3: Biểu đồ use case Đăng nhập.....	41
Hình 3-4: Biểu đồ use case Quản lý người dùng	41
Hình 3-5: Biểu đồ use case Quản lý danh mục sản phẩm.....	42
Hình 3-6: Biểu đồ use case Quản lý sản phẩm	42
Hình 3-7: Biểu đồ use case Quản lý đơn vị	43
Hình 3-8: Biểu đồ use case Nhập kho.....	43
Hình 3-9: Biểu đồ use case Xuất kho	44
Hình 3-10: Biểu đồ use case Quản lý livestream.....	44
Hình 3-11: Biểu đồ use case Quản lý cú pháp.....	45
Hình 3-12: Biểu đồ use case Quản lý nhóm hàng hoá.....	45
Hình 3-13: Biểu đồ use case Quản lý fanpage.....	46

Hình 3-14: Biểu đồ use case Thiết lập tự động.....	47
Hình 3-15: Biểu đồ use case Quản lý tin nhắn mẫu.....	47
Hình 3-16: Biểu đồ use case Quản lý comment mẫu.....	48
Hình 3-17: Biểu đồ use case Quản lý tài khoản.....	48
Hình 3-18: Biểu đồ use case Tổng hợp tồn kho.....	49
Hình 3-19: Biểu đồ use case Quản lý khách hàng	49
Hình 3-20: Biểu đồ use case Quản lý số điện	50
Hình 3-21: Biểu đồ use case Quản lý nhóm khách hàng	50
Hình 3-22: Biểu đồ use case Quản lý đơn hàng.....	51
Hình 3-23: Biểu đồ use case Quản lý bài đăng.....	51
Hình 3-24: Biểu đồ use case Bán hàng	52
Hình 3-25: Biểu đồ use case Thống kê	52
Hình 3-26: Biểu đồ use case Xem nhật ký truy cập.....	53
Hình 3-27: Biểu đồ lớp	56
Hình 3-28: Giao diện đăng nhập.....	74
Hình 3-29: Giao diện đăng ký.....	75
Hình 3-30: Giao diện quên mật khẩu.....	75
Hình 3-31: Giao diện kết nối với facebook	76
Hình 3-32: Giao diện lựa chọn fanpage kết nối	76
Hình 3-33: Giao diện trang chủ	77
Hình 3-34: Giao diện nhóm hàng hóa.....	78
Hình 3-35: Giao diện tạo mới nhóm hàng hóa	78
Hình 3-36: Giao diện cập nhật nhóm hàng hóa	79
Hình 3-37: Giao diện hàng hóa	79
Hình 3-38: Giao diện tạo mới hàng hóa.....	80
Hình 3-39: Giao diện cập nhật hàng hóa	80
Hình 3-40: Giao diện đơn vị tính	81
Hình 3-41: Giao diện đối tác giao hàng	81
Hình 3-42: Giao diện nhóm khách hàng	82

Hình 3-43: Giao diện khách hàng	82
Hình 3-44: Giao diện nhập kho.....	83
Hình 3-45: Giao diện xuất kho	83
Hình 3-46: Giao diện tồn kho	84
Hình 3-47: Giao diện bán hàng.....	84
Hình 3-48: Giao diện lựa chọn sản phẩm bán.....	85
Hình 3-49: Giao diện điền thông tin bán hàng.....	85
Hình 3-50: Giao diện đơn hàng	86
Hình 3-51: Giao diện khách hàng bị chặn	87
Hình 3-52: Giao diện quản lý bài đăng trên facebook.....	88
Hình 3-53: Giao diện marketing sản phẩm.....	89
Hình 3-54: Giao diện danh sách chatbot.....	90
Hình 3-55: Giao diện tạo mới chatbot	90
Hình 3-56: Giao diện thiết lập	91
Hình 3-57: Giao diện nhật ký truy cập	92
Hình 3-58: Giao diện báo cáo theo đơn hàng	92
Hình 3-59: Giao diện báo cáo theo tin nhắn	93
Hình 3-60: Giao diện báo cáo theo fanpage	93
Hình 3-61: Giao diện báo cáo theo sản phẩm	94
Hình 3-62: Giao diện thiết lập cú pháp	94
Hình 3-63: Giao diện thiết lập nhóm sản phẩm	95
Hình 3-64: Giao diện danh sách video livestream	95
Hình 3-65: Giao diện chi tiết livestream video.....	96
Hình 3-66: Giao diện chi tiết đơn hàng	97
Hình 3-67: Giao diện trang chủ	97
Hình 3-68: Giao diện quản lý tài khoản người dùng	98
Hình 3-69: Giao diện báo cáo số lượng người dùng.....	98
Hình 4-1: Giao diện trang chủ DigitalOcean	100
Hình 4-2: Tạo mới Kubernetes – Chọn khu vực.....	100

Hình 4-3: Tạo mới Kubernetes – Chọn dung lượng cụm	101
Hình 4-4: Giao diện một kubernetes cluster	102
Hình 4-5: Kubernetes dashboard	114
Hình 4-6: Danh sách Node trong cụm cluster.....	115
Hình 4-7: Danh sách Deployment trong cụm cluster	115
Hình 4-8: Danh sách Service trong cụm cluster	116
Hình 4-9: Tạo mới Pods trong cụm cluster.....	116
Hình 4-10: API đăng nhập	117
Hình 4-11: API xem danh sách loại hàng hoá	118

DANH MỤC BẢNG

Bảng 3-1: Danh sách actors	53
Bảng 3-2: Danh sách use case.....	55
Bảng 3-3: Các lớp của sơ đồ lớp.....	57
Bảng 3-4: Các đối tượng và thuộc tính của đối tượng.....	60
Bảng 3-5: Bảng Loại sản phẩm.....	61
Bảng 3-6: Bảng Sản phẩm	61
Bảng 3-7: Bảng Variants.....	62
Bảng 3-8: Bảng Units	63
Bảng 3-9: Bảng Users	63
Bảng 3-10: Bảng Orders	64
Bảng 3-11: Bảng Shipping Information.....	65
Bảng 3-12: Bảng Coupons	65
Bảng 3-13: Bảng Stores	66
Bảng 3-14: Bảng Notes	66
Bảng 3-15: Bảng Syntaxes.....	67
Bảng 3-16: Bảng Product Groups	67
Bảng 3-17: Bảng Import receipts	68
Bảng 3-18: Bảng Import receipt details	68
Bảng 3-19: Export receipt details	69
Bảng 3-20: Bảng Export receipt details	69
Bảng 3-21: Bảng customers	70
Bảng 3-22: Bảng customer groups.....	70
Bảng 3-23: Bảng Customer Blocks	71
Bảng 3-24: Bảng Livestream	71
Bảng 3-25: Bảng Comment samples	72
Bảng 3-26: Bảng message samples.....	72
Bảng 3-27: Bảng Chatlist.....	73
Bảng 3-28: Bảng Settings	73

Bảng 3-29: Bảng Order details 74

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Từ đầy đủ	Giải thích
CSDL	Cơ sở dữ liệu	Cơ sở dữ liệu cho ứng dụng
HTTP	HyperText Transfer Protocol	Giao thức truyền siêu văn bản
HTTPS	HyperText Transfer Protocol Secure	Giao thức truyền siêu văn bản với bảo mật
Amazon S3	Amazon Simple Storage Service	Dịch vụ lưu trữ đơn giản của Amazon
k8s	Kubernetes	Viết tắt của kubernetes
API	Application Programming Interface	Giao diện lập trình ứng dụng
SDK	Software Development Kit	Bộ công cụ và phần mềm phục vụ cho việc phát triển phần mềm

• TÓM TẮT KHÓA LUẬN

Khóa luận “TÌM HIỂU KUBERNETES VÀ XÂY DỰNG ỨNG DỤNG TỰ ĐỘNG CHỐT ĐƠN LIVESTREAM FACEBOOK” gồm 05 chương:

Chương 1: Giới thiệu về đề tài, đưa ra các điểm nổi bật so với các ứng dụng trước.

Tiếp đến là đề xuất các giải pháp để giải quyết các vấn đề đã đặt ra. Ngoài ra, chương 1 cũng đề cập đến đối tượng nghiên cứu, phạm vi đề tài, phương pháp nghiên cứu.

Chương 2: Trình bày các kiến thức, khái niệm nền tảng về kubernetes, cùng các công nghệ đi kèm để xây dựng ứng dụng chốt đơn livestream trên Facebook.

Chương 3: Trình bày chi tiết quy trình xây dựng ứng dụng, từ lựa chọn công nghệ, xác định và phân tích yêu cầu bài toán cho đến xây dựng CSDL và cuối cùng là xây dựng giao diện cho ứng dụng.

Chương 4: Tập trung trình bày cách áp dụng kubernetes vào việc triển khai ứng dụng. Các kiến thức cần nắm, các tool hỗ trợ, cách đăng ký, sử dụng và triển khai thực tế.

Chương 5: Kết luận, rút ra được các ưu nhược điểm của ứng dụng và hướng phát triển trong tương lai.

Chương 1. GIỚI THIỆU CHUNG

1.1. Lý do chọn đề tài

Công nghệ thông tin và Live Commerce (bán hàng online), đang dần được ứng dụng rộng rãi và xã hội nói chung và những cá nhân bán lẻ và doanh nghiệp lớn nói riêng.

Trong bối cảnh covid vẫn đang rình rập quay trở lại, đe doạ đẩy nền kinh tế vào chu kỳ suy giảm mới trong năm 2021. Do vậy, chuyển đổi số các hoạt động kinh doanh lên mạng đã trở thành lựa chọn tất yếu cho các doanh nghiệp và người tiêu dùng.

Tuy nhiên “nền kinh tế không chạm” với hình thức đăng tin bán với nội dung đơn thuần là chữ, ảnh và video truyền thống còn đơn điệu và nhảm chán, không có sự tương tác tức thời với khách hàng nên còn thiếu tính “người”, chưa thể sánh được với các hình thức mua bán truyền thống có tiếp xúc tại cửa hàng.

Để khắc phục được những nhược điểm đó, bán hàng thông qua truyền hình trực tiếp(Livestream, hay còn gọi là Live Commerce) đã bùng nổ và trở thành nền tảng bán hàng chính của nhiều cá nhân và doanh nghiệp.

Vấn đề đặt ra khi khi số lượng bán hàng của mỗi lượt livestream tăng cao thì người bán hàng sẽ khó khăn trong việc chốt đơn hàng, cũng như quản lý thông tin của khách hàng . Vì vậy nhóm muốn tạo ra một ứng dụng có thể quản lý và tự động chốt đơn hàng livestream trên facebook, giúp người bán dễ và thuận tiện hơn trong việc quản lý bán hàng khi livestream. Đặc biệt với nhóm sẽ nghiên cứu và sử dụng kubernetes, một trong những nền tảng mã nguồn mở giúp tự động hóa rất nhiều các quy trình thủ công liên quan đến việc triển khai, quản lý, mở rộng, ... để có thể triển khai và xây dựng ứng dụng này.

1.2. Tính năng mới/ khác biệt về chức năng của đè tài so với một số ứng dụng khác trên thị trường

Các ứng dụng chốt đơn livestream hiện nay trên thị trường còn khá ít, 2 cái tên nổi bật hiện là OCM của Misa và ứng dụng Tpos, nhìn chung các ứng dụng không có gì khác biệt về ý tưởng cũng như cách sử dụng.

Sau quá trình khảo sát, dùng thử cả hai ứng dụng trên, thì nhóm đã chọn lọc ra những chức năng thuận tiện nhất để giúp đỡ cho người bán hàng có thể bán hàng dễ dàng và mang lại hài lòng cho người mua hàng.

Khảo sát qua các hai ứng dụng nói trên thì nhóm thấy có một số hạn chế như:

- Khó sử dụng: người dùng sẽ rất khó để làm quen với ứng dụng lúc đầu, cần phải có support bên ứng dụng hỗ trợ mới có thể biết cách sử dụng.
- Giao diện không thân thiện: giao diện của các ứng dụng hơi rối, bố trí chưa hợp lý không đem lại cảm giác thân thiện.

⇒ Điều này đã làm dẫn đến sự hạn chế thoái mái trong việc livestream bán hàng.

- ✓ **Ứng dụng Flad** sẽ chú trọng vào việc tạo ra giao diện đơn giản để cho người bán hàng có thể dễ dàng và nhanh chóng làm quen với ứng dụng. Cũng như tạo ra nghiệp vụ đơn giản trong việc chốt đơn và chăm sóc khách hàng đặt đơn trong lúc livestream.

OCM, Tpos không hiển thị trực tiếp bình luận của khách hàng khi người bán hàng đang livestream.

⇒ Làm cho người bán hàng mất thời gian và khó kiểm soát được những bình luận của khách khi mua hàng. Cũng như theo dõi được đâu là bình luận hợp lệ để mua hàng.

- ✓ **Ứng dụng Flad** lấy trực tiếp comment từ phiên livestream, mỗi bình luận mua hàng sẽ được phần mềm kiểm tra và thông báo tính hợp lệ, không hợp lệ của từng bình luận.

1.3. Đối tượng nghiên cứu

Khoá luận này hướng đến nghiên cứu các đối tượng sau:

- Các công nghệ:
 - Docker, Docker Compose
 - Kubernetes
 - Postgres
 - Nodejs, Koa Framework
 - AWS S3
 - Facebook API
 - Vuejs
- Đối tượng trong phạm vi đề tài hướng đến:
 - Các doanh nghiệp, các cửa hàng hoặc các cá nhân nhỏ lẻ, sử dụng nền tảng livestream trên facebook để bán hàng.
 - Người mua hàng trong nước.

1.4. Phạm vi nghiên cứu

Ứng dụng Flad được nhóm xây dựng bằng ngôn ngữ Javascript trên môi trường Web application và docker, Service back-end được xây dựng bằng nodejs postgres, còn front-end sử dụng vuejs để xây dựng.

Tìm hiểu về Kubernetes dựa trên một số khía cạnh:

- Khái niệm cơ bản.
- Thành phần cơ bản.
- Ưu nhược điểm.
- Cách hoạt động.
- Cách triển khai: local, server.
- Cách quản lý, scale, khắc phục sự cố

Tìm hiểu và khai thác hết những thứ có thể áp dụng vào ứng dụng với API do facebook cung cấp.

1.5. Phương pháp nghiên cứu

Cách tiếp cận: ứng dụng được xây dựng dựa theo kiến trúc Microservice trên nền tảng Web application.

Nhóm đã sử dụng các phương pháp nghiên cứu:

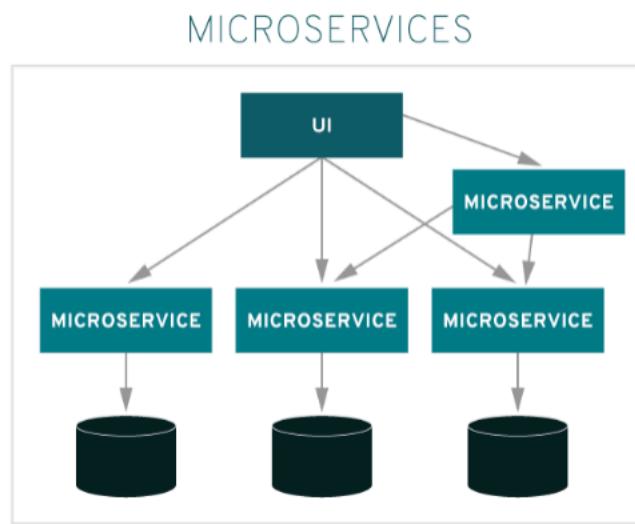
- Phương pháp đọc tài liệu
- Phương pháp phân tích các ứng dụng hiện có trên thị trường
- Phương pháp thực nghiệm

Chương 2. KIẾN THỨC NỀN TẢNG

2.1. Tổng quan về kiến trúc Microservice

2.1.1. Khái niệm

Microservice là một kiểu kiến trúc phần mềm. Các module trong phần mềm này được chia ra nhiều service rất nhỏ. Mỗi service này đều có một logic riêng, một trách nhiệm riêng và có thể deploy riêng biệt.



Hình 2-1: Kiến trúc microservices

2.1.2. Đặc điểm của Microservice

Tập hợp một nhóm nhỏ các service: mức độ chi tiết của một service là nhỏ và mỗi service này sẽ chịu một trách nhiệm cụ thể (single responsibility) và chỉ tập trung vào nhiệm vụ đó.

Việc phát triển và mở rộng một service là hoàn toàn độc lập. Điều này mang lại tính linh hoạt cho hệ thống.

Giảm tải được các mối quan ngại về công nghệ sử dụng. Chọn một công nghệ phù hợp với vấn đề của doanh nghiệp có thể được giải quyết dễ dàng. Các service giao tiếp với nhau thông qua API, do vậy mỗi service có thể dùng một ngôn ngữ riêng biệt.

2.1.3. Ưu điểm và nhược điểm của Microservice

– **Ưu điểm:**

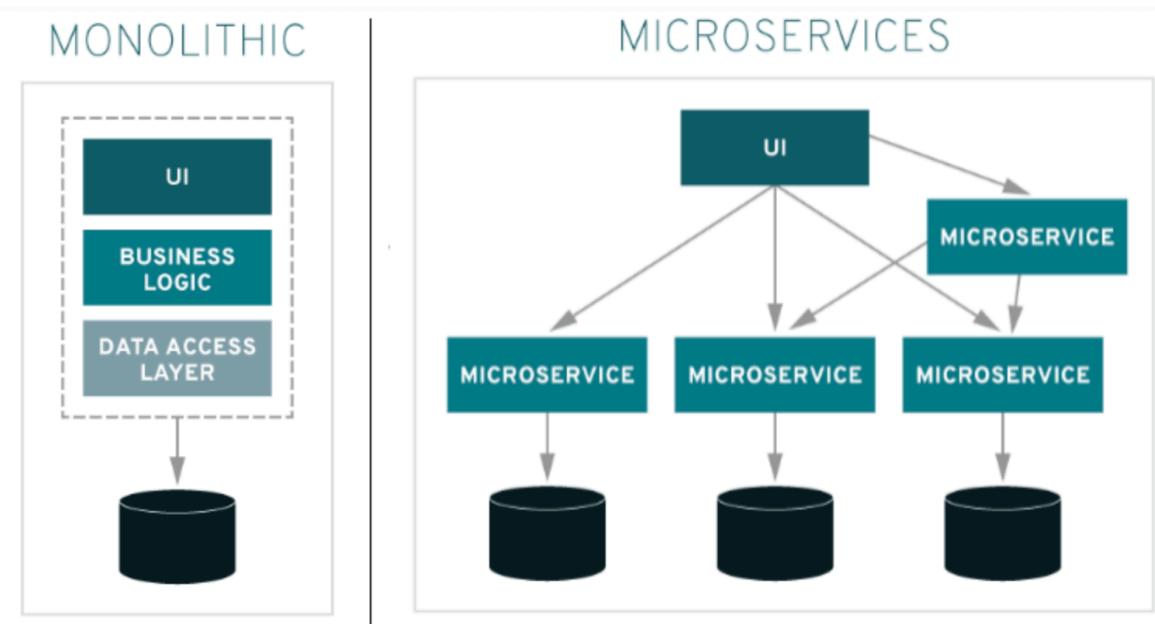
- Dễ nâng cấp và scale, đây là điều quan trọng nhất.
- Do tách biệt nên nếu một service bị lỗi, toàn bộ hệ thống vẫn hoạt động bình thường.
- Các service nằm tách biệt nhau, bạn có thể thoải mái sử dụng ngôn ngữ lập trình riêng, database riêng.
- Khả năng testing dễ dàng hơn - các services nhỏ hơn và nhanh hơn để test
- Cải thiện khả năng bảo trì - mỗi service tương đối nhỏ do đó dễ hiểu và thay đổi hơn
- Dễ dàng hơn trong việc tích hợp 3rd-party

– **Nhược điểm:**

- Dịch vụ sẽ có nhiều thành phần phải quản lý.
- Các yêu cầu về hạ tầng để đáp ứng cho các thành phần của dịch vụ cũng sẽ phức tạp hơn.
- Dịch vụ sẽ được phát triển bởi nhiều công nghệ, ngôn ngữ lập trình.
- Việc kiểm thử toàn bộ hệ thống sẽ phức tạp hơn.

2.1.4. So sánh Microservice mà Monolithic

Monolithic là kiến trúc mà chúng ta thường dùng để xây dựng phần mềm. Toàn bộ các module (view, business, database, report) đều được gom chung vào một project lớn. Khi deploy, sẽ đẩy khối code này lên server và config để nó chạy.



Hình 2-2: So sánh Microservice và Monolithic

Monolithic:

- Mỗi dịch vụ sẽ hoạt động độc lập và đầy đủ các tính năng như: xác thực, định danh người dùng, cho đến xử lý nghiệp vụ.
- Mỗi dịch vụ được mở rộng bằng cách tạo thêm một node dịch vụ mới và phân tải request vào các node dịch vụ.
- Phát triển dịch vụ quy mô nhỏ nhanh chóng và đơn giản, thuận tiện cho việc kiểm thử cho mỗi dịch vụ.

Microservice:

- Mỗi dịch vụ sẽ được chia nhỏ thành nhiều thành phần khác nhau, mỗi thành phần sẽ hoạt động độc lập, được phát triển độc lập và chỉ xử lý nghiệp vụ cho các chức năng đó.
- Mỗi thành phần không lệ thuộc vào công nghệ phát triển với các thành phần khác.
- Kiến trúc phù hợp với các dịch vụ lớn, thuận lợi trong việc phát triển lâu dài cho ứng dụng.

2.2. Tổng quan về Javascript

2.2.1. Khái niệm

JavaScript là một ngôn ngữ lập trình đa nền tảng (**cross-platform**), ngôn ngữ lập trình kịch bản, hướng đối tượng. JavaScript là một ngôn ngữ nhỏ và nhẹ (small and lightweight). Khi nằm bên trong một môi trường (host environment), JavaScript có thể kết nối tới các object của môi trường đó và cung cấp các cách quản lý chúng (object).

2.2.2. Tại sao chọn Javascript để phát triển

- Thân thiện với người mới bắt đầu:
 - Javascript là ngôn ngữ có cấu trúc đơn giản, dễ học, dễ làm, chạy trên mọi thiết bị.
- Cộng đồng lớn:
 - Vì là ngôn ngữ mã nguồn mở nên Javascript có một cộng đồng khá lớn và chất lượng.
 - Cộng đồng hỗ trợ và chia sẻ kinh nghiệm của Javascript rất dồi dào.
- Ngôn ngữ đa năng:
 - Javascript là ngôn ngữ lập trình đa năng, chạy đa nền tảng, từ web app cho đến mobile app.
 - Hầu hết các ứng dụng hiện tại đều sử dụng Javascript, ở máy chủ (serve) hoặc máy khách (client).
- Cơ hội việc làm:
 - Nhu cầu và cơ hội việc làm của javascript rất đa dạng với mức lương cao.

2.3. Hệ quản trị cơ sở dữ liệu Postgresql

PostgreSQL là hệ quản trị cơ sở dữ liệu được viết theo hướng mã nguồn mở và rất mạnh mẽ.

Postgresql là một hệ thống quản trị cơ sở dữ liệu quan hệ-đối tượng (object-relational database management system) có mục đích chung.

Nó khá phổ biến với giới lập trình viên bởi:

- PostgreSQL được thiết kế để chạy trên các nền tảng tương tự UNIX.
- PostgreSQL là một phần mềm mã nguồn mở miễn phí.
- PostgreSQL không yêu cầu quá nhiều công tác bảo trì bởi có tính ổn định cao.

2.4. Tổng quan Nodejs và thư viện Koa

2.4.1. Giới thiệu Nodejs

Nodejs là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

Nodejs tạo ra được các ứng dụng có tốc độ xử lý nhanh, realtime thời gian thực.

Nodejs áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án Startup nhanh nhất có thể.

Những ứng dụng nên viết bằng Nodejs:

- Websocket server: Các máy chủ web socket như là Online Chat, Game Server...
- Fast File Upload Client: là các chương trình upload file tốc độ cao.
- Ad Server: Các máy chủ quảng cáo.
- Cloud Services: Các dịch vụ đám mây.
- RESTful API: đây là những ứng dụng mà được sử dụng cho các ứng dụng khác thông qua API.

- Any Real-time Data Application: bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực.

2.4.2. Giới thiệu thư viện Koa

Koa là một Web framework Nodejs, được viết bởi team của Expressjs. Koa sử dụng nhiều chức năng mới của ES6, được xem là chưa ổn định với các bản Nodejs dưới 4.0, nhưng mới vừa rồi Node v6 vừa ra mắt, đánh dấu Koajs sẽ là 1 framework mạnh và ổn định hơn.

Một số đặc điểm nổi bật của Koa:

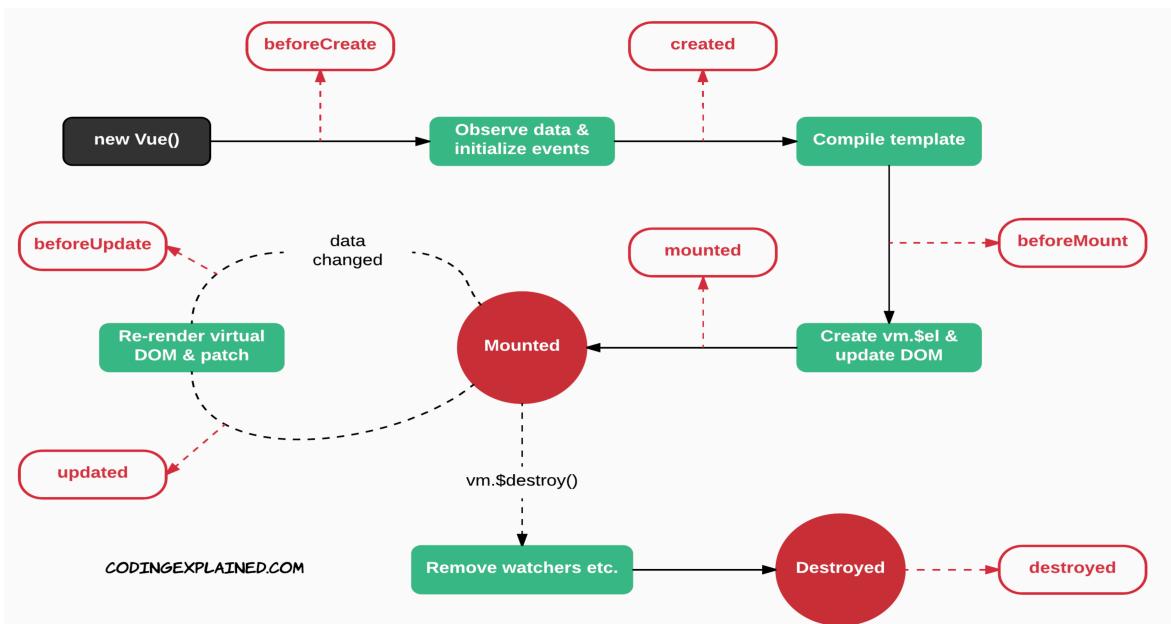
- Chủ yếu tận dụng chức năng generators của Javascript ES6
- Có phong cách code rất lite, giúp app nhỏ gọn và dễ kiểm soát.
- Được viết dựa trên co - giúp xử lý generators, thay vì sử dụng callback, Koa nhờ đó có cú pháp đơn giản và sạch sẽ hơn.
- Khác với Express với hàng tá các middleware được cài đặt sẵn. Định hướng là 1 framework core tinh gọn, Koa không có bất kì middleware được tích hợp sẵn cả.

2.5. Tổng quan về Vuejs

2.5.1. Giới thiệu về Vuejs

Vue.js là một framework linh động dùng để xây dựng giao diện người dùng. Khác với các framework nguyên khối (monolithic), Vue được thiết kế từ đầu theo hướng cho phép và khuyến khích việc phát triển ứng dụng theo từng bước. Khi phát triển lớp giao diện (view layer), người dùng chỉ cần dùng thư viện lõi (core library) của Vue, vốn rất dễ học và tích hợp với các thư viện hoặc dự án có sẵn. Cùng lúc đó, nếu kết hợp với những kỹ thuật hiện đại như SFC (single file components) và các thư viện hỗ trợ, Vue cũng đáp ứng được dễ dàng nhu cầu xây dựng những ứng dụng một trang (SPA - Single-Page Applications) với độ phức tạp cao hơn nhiều.

2.5.2. Một số khái niệm và thành phần chính



Hình 2-3: Vòng đời Vuejs

Vuejs gồm các thành phần sau:

- **Vue Component:** là một trong những tính năng quan trọng nhất trong Vue. Nó giúp cho việc kế thừa các thành phần HTML cơ bản, dễ dàng đóng gói và tái sử dụng code.
- **Vue Directive:** là một đoạn lệnh nhỏ mà ta có thể chèn chúng vào các phần tử DOM. Tên của các đoạn lệnh đó được bắt đầu bằng tiền tố `v-` dùng để Vue có thể biết được rằng bạn đang sử dụng một chút đánh dấu đặc biệt và để cho cú pháp được nhất quán. Một số directive thường sử dụng trong vue như: `v-if`, `v-for`, `v-model`, và ta có thể custom thêm một số directive khác tùy và mục đích sử dụng của mỗi dự án.
- **Vue Router:** là bộ định tuyến chính thức cho Vue.js. Nó tích hợp sâu với lõi Vue.js để làm cho việc xây dựng các Single Page Applications với Vue.js trở nên dễ dàng. Đồng thời nó giúp cho việc điều hướng của ứng dụng trở nên dễ dàng hơn.

- **Vuex:** Khi project của bạn sử dụng Vuejs mà có chứa nhiều components sẽ gặp phải vấn đề là việc chia sẻ dữ liệu giữa các component và việc dùng chung dữ liệu của các component phải được cập nhật khi có dữ liệu thay đổi. Vuex đóng vai trò là một centralized state management. Vuex quản lý các state thông qua việc implement:
 - State: một object chứa dữ liệu
 - Getter: giống như một bộ lọc cho phép truy cập và sàng lọc dữ liệu
 - Mutation: nơi chúng ta thực hiện các thay đổi state
 - Action: dùng để fire mutation, commit các thay đổi.

2.6. Giới thiệu về AWS S3

2.6.1. AWS S3 là gì?

Amazon Simple Storage Service (Amazon S3) là kho lưu trữ dành cho Internet. Bạn có thể sử dụng Amazon S3 để lưu trữ và truy xuất dữ liệu với khối lượng bất kỳ, vào bất cứ thời điểm nào, từ bất cứ nơi nào trên web. Bạn có thể thực hiện được các tác vụ đó bằng Bảng điều khiển quản lý AWS, một giao diện web đơn giản và trực quan.

2.6.2. Đặc điểm nổi bật của AWS S3

S3 an toàn vì AWS cung cấp:

- Mã hóa dữ liệu mà bạn lưu trữ. Nó có thể xảy ra theo 2 cách:
 - Mã hóa ở client side
 - Mã hóa ở server side
- Nhiều bản copy được duy trì để cho phép phục hồi dữ liệu trong trường hợp dữ liệu bị hỏng.

S3 bền vững:

- Nó thường xuyên xác minh tính toàn vẹn của dữ liệu bằng cách sử dụng checksum.

- Ngay cả trong khi dữ liệu lưu trữ hoặc truy xuất dữ liệu, nó sẽ kiểm tra lưu lượng mạng đến cho bất kỳ gói dữ liệu bị hỏng nào.

S3 có khả năng mở rộng cao vì nó tự động tăng dung lượng lưu trữ của bạn theo yêu cầu và bạn chỉ trả tiền cho bộ nhớ bạn sử dụng

Loại dữ liệu được lưu trữ trên AWS S3:

- Có thể lưu trữ bất kỳ loại data với bất kỳ loại format nào.
- Trên S3 khi chúng ta nói về dung lượng, số lượng của đối tượng mà chúng ta có thể lưu trữ trên S3 là không giới hạn.
- Khi ta nói về dữ liệu, có 2 loại chính:
 - Dữ liệu truy cập thường xuyên.
 - Dữ liệu truy cập không thường xuyên.

2.6.3. Chi phí đăng ký sử dụng

Amazon đưa ra 3 lớp lưu trữ cung cấp cho khách hàng trải nghiệm tốt nhất với chi phí hợp lý

- **Amazon S3 Standard để truy cập dữ liệu thường xuyên**
 - Phù hợp với các trường hợp sử dụng nhạy cảm với hiệu suất, nơi độ trễ phải được giữ ở mức thấp.
- **Amazon S3 Standard để truy cập dữ liệu không thường xuyên**
 - Phù hợp với trường hợp sử dụng nơi dữ liệu được lưu trữ lâu dài và ít được truy cập thường xuyên, tức là vẫn lưu trữ dữ liệu nhưng vẫn mong đợi hiệu suất cao.
- **Amazon Glacier**
 - Phù hợp với các trường hợp sử dụng nơi dữ liệu được lưu trữ hiệu suất cao là điều không cần thiết, nó có chi phí thấp hơn so với 2 dịch vụ trên.

2.7. Tổng quan về Facebook API

2.7.1. Một số khái niệm chính

2.7.1.1. Định nghĩa về API

API (Application Programming Interface), tạm dịch là giao diện chương trình ứng dụng. API là một phương tiện để giao tiếp giữa các chương trình, là xu hướng trong thế giới lập trình. Không chỉ chỉ có Facebook API mà Google, Yahoo, Amazon và các công ty lớn khác cũng cung cấp các API riêng, với API này bạn có thể tạo ra các ứng dụng bằng cách sử dụng tính năng hoặc dữ liệu hiện có trên máy chủ của họ.

2.7.1.2. Định nghĩa về Facebook API

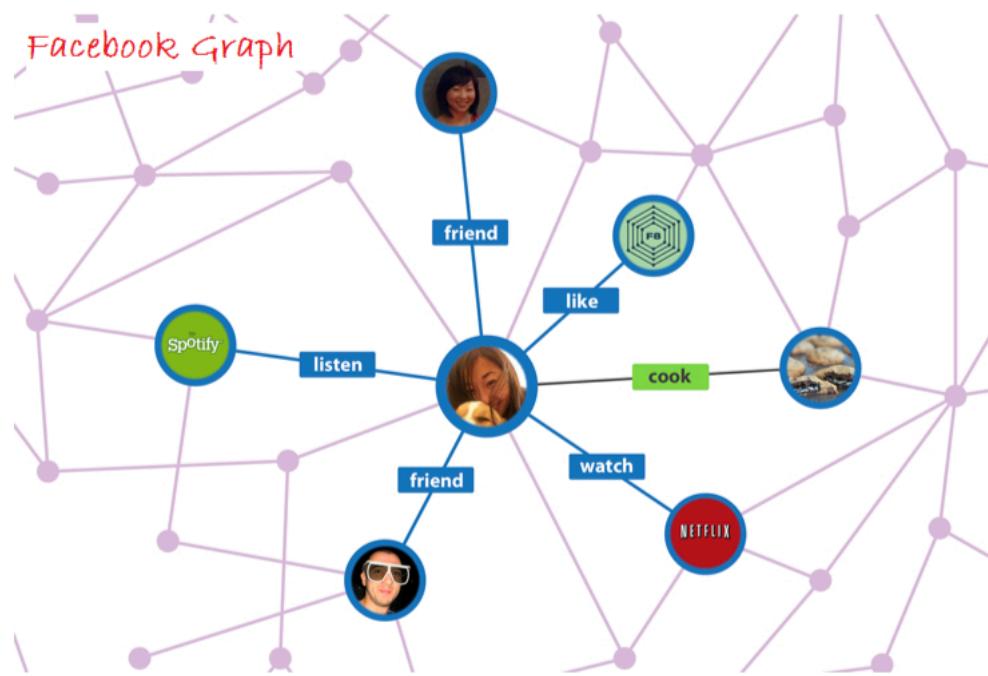
Facebook API là nền tảng do Facebook cung cấp cho người viết ứng dụng để dễ dàng trong việc tạo ứng dụng và đảm bảo người viết ứng dụng không can thiệp quá sâu vào hệ thống của Facebook.

Qua Facebook API, ta có thể lấy được thông tin về người dùng như là danh sách bạn bè, thông tin cá nhân, ảnh profile.... nếu như họ cấp quyền cho ta truy cập trang cá nhân của họ.

Facebook sẽ gửi một phương thức POST đến máy chủ Facebook API. Nó bao gồm một số các thông số yêu cầu như api_key của ứng dụng, session_key của người dùng đưa ra yêu cầu.

Bên cạnh đó Facebook còn thêm vào tham số fb_sig để thông báo ứng dụng đưa ra yêu cầu. Bằng cách này tất cả các lời gọi API sẽ được đảm bảo, Facebook có thể xác minh các yêu cầu được gửi từ một ứng dụng đã được chấp thuận.

2.7.1.3. Định nghĩa về Facebook Graph API



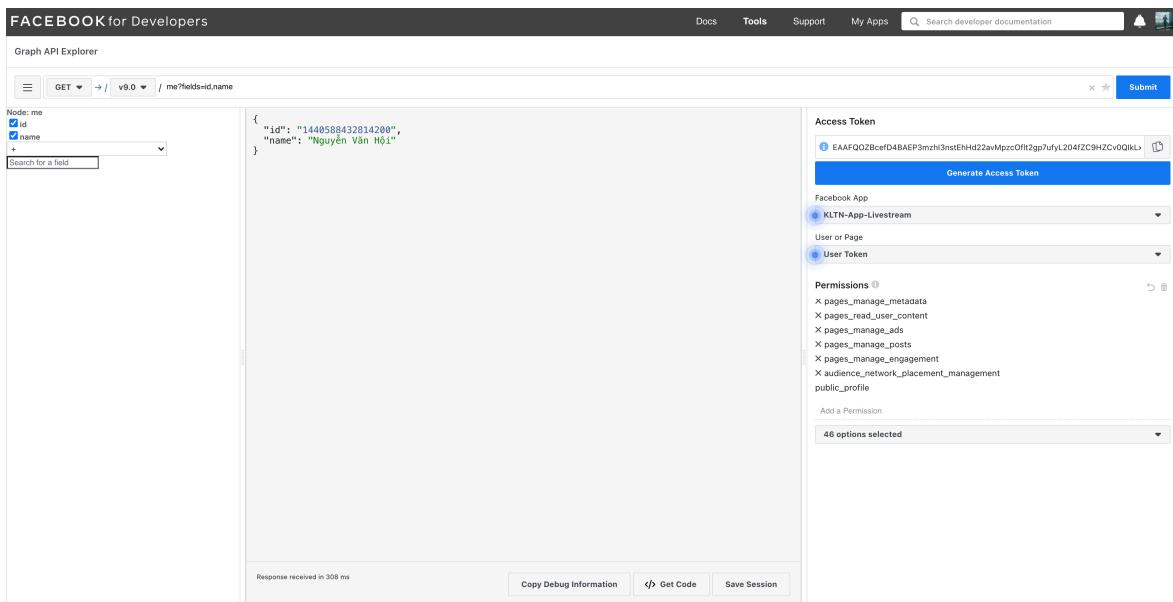
Hình 2-4: Ảnh minh họa Facebook Graph API

Graph ở đây chính là đồ thị. Graph sinh ra để miêu tả quan hệ giữa các thực thể.

Facebook coi các mối quan hệ giữa các thực thể như là một "Đồ thị xã hội" (Social Graph).

Facebook Graph API là cách chủ yếu để tải dữ liệu vào và lấy dữ liệu ra từ đồ thị xã hội của Facebook. Đó là một HTTP API cấp thấp mà bạn có thể sử dụng để truy vấn dữ liệu, post status, tải lên hình ảnh và một loạt các nhiệm vụ khác.

2.7.2. Cấu trúc Facebook Graph API



Hình 2-5: Graph API Explorer

Graph API được đặt tên theo ý tưởng "đồ thị xã hội" - đại diện cho các thông tin trên Facebook. Nó bao gồm:

- nodes (nút): là các đối tượng riêng như là người dùng, ảnh, trang cá nhân, bình luận...
- edges (cạnh): là các kết nối giữa những đối tượng riêng ở trên, ví dụ như kết nối hình ảnh và trang chứa hình ảnh đó, bình luận và bức ảnh được bình luận...
- fields (trường): dữ liệu của đối tượng riêng ở trên, ví dụ như tên, ngày sinh của người dùng, tên trang...

Vì vậy mà chúng ta sử dụng Graph API là để:

- Sử dụng nodes để lấy dữ liệu về đối tượng cụ thể.
- Sử dụng edges để lấy tập hợp các đối tượng khác được kết nối với node.
- Sử dụng fields để chỉ định dữ liệu mà bạn muốn có trong phản hồi từ Facebook.

2.7.2.1. HTTP

Graph API dựa trên HTTP nên API này hoạt động với bất cứ ngôn ngữ nào có thư viện HTTP, chẳng hạn như cURL và urllib. Bạn cũng có thể sử dụng Graph API ngay trong trình duyệt của mình.

2.7.2.2. Access Token

Khi ai đó kết nối với một ứng dụng bằng cách Đăng nhập vào Facebook và đồng ý cấp quyền truy cập cho ứng dụng thì lúc đó ứng dụng sẽ lấy được mã truy cập tạm thời của người dùng đó.

Access Token là một chuỗi xác định người dùng, ứng dụng hoặc trang. Ứng dụng có thể dùng mã đó để thực hiện lệnh gọi Graph API.

Có thể lấy mã truy cập bằng nhiều phương thức. Mã bao gồm thông tin về thời gian mã sẽ hết hạn và ứng dụng đã tạo mã đó. Vì kiểm tra quyền riêng tư, phần lớn các lệnh gọi API trên Facebook đều cần có mã truy cập. Mã truy cập có các loại khác nhau để hỗ trợ các trường hợp sử dụng khác nhau.

Có 3 loại mã truy cập là:

- Mã truy cập người dùng: dùng để thay mặt một người sửa đổi hoặc ghi dữ liệu Facebook của người đó
- Mã truy cập ứng dụng: dùng để đăng hành động trong Open Graph
- Mã truy cập trang: dùng để sửa đổi dữ liệu thuộc về 1 trang Facebook.

Sở dĩ gọi là mã truy cập tạm thời vì mã truy cập có 2 loại:

- Mã ngắn hạn: Thường có thời hạn khoảng 1 đến 2 giờ.
- Mã dài hạn: Thường có thời hạn khoảng 60 ngày.

Các thời hạn này sẽ không giữ nguyên. Mã truy cập được tạo bằng cách đăng nhập web thường là mã ngắn hạn nhưng ta cũng có thể chuyển thành mã dài hạn bằng cách thực hiện lệnh gọi API phía máy chủ cùng với secret key của ứng dụng.

Sau khi có mã truy cập, ta có thể sử dụng mã này để thực hiện gọi lệnh từ ứng dụng di động, trình duyệt web hoặc từ máy chủ của ta đến máy chủ của Facebook.

Hiện tại đều phải sử dụng HTTPs để lấy được access_token.

2.7.2.3. URL lưu trữ

Hầu như tất cả các yêu cầu đều được chuyển đến URL lưu trữ graph.facebook.com.

Chỉ có video tải lên sử dụng graph-video.facebook.com.

2.7.2.4. Objects ID

Mỗi node có một ID duy nhất để truy cập thông qua Graph API. Để có thông tin về node, ta phải truy vấn trực tiếp đến ID của node đó.

2.7.3. Facebook Pages API

Pages API cho phép các ứng dụng truy cập, cập nhật và cài đặt nội dung của fanpage trên facebook. Có thể tạo, tải bài đăng, nhận xét về nội dung do trang sở hữu, nhận thông tin chi tiết về trang, hoặc cập nhật các hành động mà người dùng thực hiện trên trang,...

Pages API là tập hợp các endpoints của facebook Graph API mà các ứng dụng có thể sử dụng để tạo và quản lý cài đặt nội dung của trang.

Một số thành phần chính của Pages API:

- Access Tokens:
 - Xác thực khi gọi một API của facebook đều thông qua access tokens.
 - Hầu hết các endpoints đều yêu cầu Page access tokens và là duy nhất trên mỗi trang.
- Permissions:
 - Hầu hết các endpoints yêu cầu một hoặc nhiều quyền mà người dùng ứng dụng cung cấp thông qua việc đăng nhập facebook.
 - Tất cả quyền yêu cầu trang đều yêu cầu xem xét ứng dụng trước khi ứng dụng có thể sử dụng chúng để truy cập.
 - Một số quyền cần có để truy cập vào sử dụng api của pages:
 - manage_pages
 - page_manage_ads

- page_manage_metadata
 - page_read_engagement
 - page_read_user_content
- Page-SScoped User IDs:
- Người dùng tương tác với page được xác định bằng Page-SScoped User IDs (PSID). PSID là ID duy nhất cho mỗi cặp người dùng – trang. Endpoints của page và API messenger dựa trên PSID, vì vậy có thể sử dụng PSID để xác định tương tác của người dùng với page, cũng như các đoạn hội thoại của page với người dùng đó.
- Rate limits:
- Tất cả các endpoints của page đều phải tuân theo giới hạn được định mức sẵn. Có thể xem mức hạn chế này trong trang tổng quan của ứng dụng.

2.8. Tổng quan về Docker

2.8.1. Docker là gì?

Docker là nền tảng cung cấp cho các công cụ, service để các developers, admins systems có thể phát triển, thực thi, chạy các ứng dụng với containers. Nó là một nền tảng để cung cấp cách để building, deploy và run các ứng dụng một cách dễ dàng trên nền tảng ảo hóa - "Build once, run anywhere".

2.8.2. Sự khác nhau giữa docker và virtual machine

Docker: Dùng chung kernel, chạy độc lập trên Host Operating System và có thể chạy trên bất kỳ hệ điều hành nào cũng như cloud. Khởi động và làm cho ứng dụng sẵn sàng chạy trong 500ms, mang lại tính khả thi cao cho những dự án cần sự mở rộng nhanh.

Virtual Machine: Cần thêm một Guest OS cho nên sẽ tốn tài nguyên hơn và làm chậm máy thật khi sử dụng. Thời gian khởi động trung bình là 20s có thể lên đến hàng phút, thay đổi phụ thuộc vào tốc độ của ổ đĩa

2.8.3. Tại sao nên dùng docker

Tiện lợi: với docker mọi thứ trở nên đơn giản, chỉ với vài dòng lệnh, ta có thể nhanh chóng tạo được môi trường ảo hóa chứa đầy đủ những cài đặt cần thiết cho project.

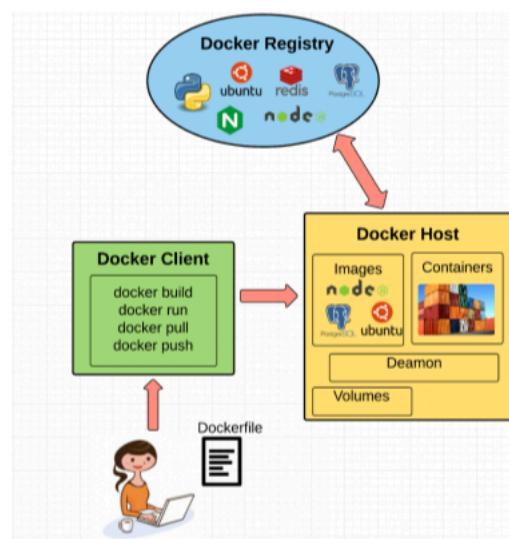
Dễ dàng sử dụng: Docker rất dễ cho mọi người sử dụng từ developers, systems admins, architect, ... nó tận dụng lợi thế của container để build, test nhanh chóng.

Tốc độ: Docker container rất nhẹ và nhanh, ta có thể tạo và chạy docker container trong vài giây so sánh với VMs thì mỗi lần chạy VMs cần rất nhiều thời gian khởi động.

Khả năng di động: môi trường develop được dựng lên bằng docker có thể chuyển từ người này sang người khác mà không làm thay đổi cấu hình ở trong.

Chia sẻ: DockerHub là một “app store for docker images”. Trên DockerHub có hàng ngàn public images được tạo bởi cộng đồng. Dễ dàng tìm thấy những image cần và chỉ cần pull về và sử dụng với một số sửa đổi nhỏ.

2.8.4. Kiến trúc của docker



Hình 2-6: Kiến trúc của docker

Docker Daemon: lắng nghe các yêu cầu từ Docker Client để quản lý các đối tượng như Container, Image, Network và Volumes. Các Docker Daemon cũng giao tiếp với nhau để quản lý các Docker Service.

Docker Client: là một công cụ giúp người dùng giao tiếp với Docker host.

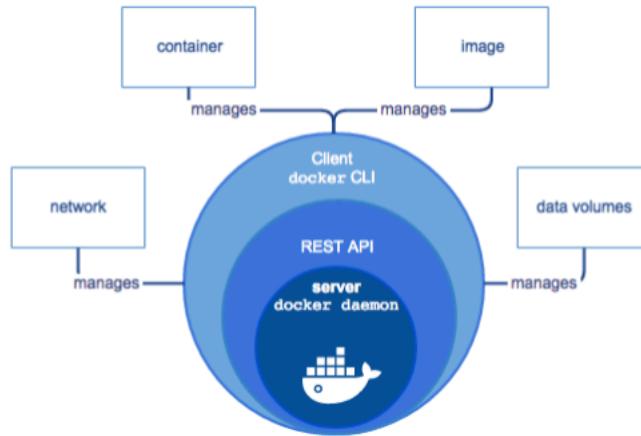
Docker Registry (Docker Hub): là một kho chứa các image được publish bởi cộng đồng Docker. Nó giống như GitHub và bạn có thể tìm những image cần thiết và pull về sử dụng.

Docker object: chính là các đối tượng mà ta thường xuyên gặp khi sử dụng Docker. Gồm có Images và Containers.

- **Images:** là một file bất biến, không thay đổi, chứa các source code, libraries, dependencies, tools và các files khác cần thiết cho một ứng dụng để chạy. Ta có thể tự build một image riêng cho mình hoặc sử dụng những image được publish từ cộng đồng Docker Hub. Một image sẽ được build dựa trên những chỉ dẫn của Dockerfile.
- **Containers:** là một run-time environment mà ở đó người dùng có thể chạy một ứng dụng độc lập. Những container này rất gọn nhẹ và cho phép bạn chạy ứng dụng trong đó rất nhanh chóng và dễ dàng.

2.8.5. Các thành phần cơ bản của docker

Docker Engine: là một ứng dụng client-server.



Hình 2-7: Thành phần cơ bản của docker

- Các thành phần:
 - o Server hay còn được gọi là docker daemon: chịu trách nhiệm tạo, quản lý các Docker objects như images, containers, networks, volume.
 - o REST API: docker daemon cung cấp các api cho Client sử dụng để thao tác với Docker.
 - o Client là thành phần đầu cuối cung cấp một tập hợp các câu lệnh sử dụng api để người dùng thao tác với Docker.

Distributaion Tools: là các công cụ phân tán

- Giúp chúng ta lưu trữ và quản lý các Docker Images như: Docker Registry, Docker Trusted Registry, Docker Hub.
- Docker Hub là một công cụ phần mềm như một dịch vụ cho phép người dùng public hay private các images của chúng ta.

Orchestration Tools:

- Docker Machine: Machine tạo Docker Engine trên laptop của bạn hoặc trên bất cứ dịch vụ cloud phổ biến nào như AWS, Azure, Google Cloud, ...
- Docker Compose: là công cụ giúp định nghĩa và khởi chạy multi-container Docker applications

- Docker Swarm: là một công cụ giúp chúng ta tạo ra một clustering Docker. Nó giúp chúng ta gom nhiều Docker Engine lại với nhau và ta có thể "nhìn" nó như duy nhất một virtual Docker Engine

Dockerfile: như một script dùng để build các image trong container. Dockerfile bao gồm các câu lệnh liên tiếp nhau được thực hiện tự động trên một image gốc để tạo ra một image mới. Dockerfile giúp đơn giản hóa tiến trình từ lúc bắt đầu đến khi kết thúc.

2.9. Tổng quan về Kubernetes

2.9.1. Kubernetes là gì?

Kubernetes là một nền tảng nguồn mở, khả chuyên, có thể mở rộng để quản lý các ứng dụng được đóng gói và các service, giúp thuận lợi trong việc cấu hình và tự động hoá việc triển khai ứng dụng. Kubernetes là một hệ sinh thái lớn và phát triển nhanh chóng. Các dịch vụ, sự hỗ trợ và công cụ có sẵn rộng rãi.

Tên gọi Kubernetes có nguồn gốc từ tiếng Hy Lạp, có ý nghĩa là người lái tàu hoặc hoa tiêu. Google mở mã nguồn Kubernetes từ năm 2014. Kubernetes xây dựng dựa trên một thập kỷ rưỡi kinh nghiệm mà google có được với việc vận hành một khối lượng lớn workload trong thực tế, kết hợp với các ý tưởng và thực tiễn tốt nhất từ cộng đồng.

2.9.2. Tại sao nên dùng kubernetes

Kubernetes quản lý và vận hành ứng dụng trên các container. Và để các ứng dụng được chạy trên môi trường production không bị downtime và nếu một container bị tắt đi, một container khác cần phải khởi động lên.

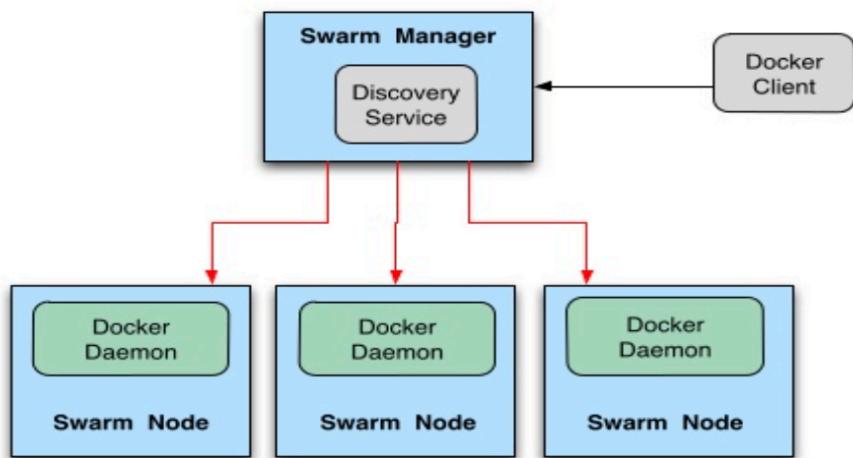
Trong docker thì ta có thể sử dụng docker swarm cho trường hợp này. Và Kubernetes cũng giải quyết bài toán như vậy.

Kubernetes cung cấp một framework để chạy các hệ phân tán một cách mạnh mẽ. Nó đảm nhiệm việc nhân rộng và chuyển đổi dự phòng cho ứng dụng của bạn, cung cấp các mẫu deployment và hơn thế nữa.

Kubernetes cung cấp cho chúng ta rất nhiều tính năng tiêu biểu như:

- **Service discovery và cân bằng tải:** Kubernetes có thể expose một container sử dụng DNS hoặc địa chỉ IP của riêng nó. Nếu lượng traffic truy cập đến một container cao, Kubernetes có thể cân bằng tải và phân phối lưu lượng mạng (network traffic) để việc triển khai được ổn định.
- **Điều phối bộ nhớ:** Kubernetes cho phép bạn tự động mount một hệ thống lưu trữ mà bạn chọn, như local storages, public cloud providers, ...
- **Tự động rollouts và rollbacks:** bạn có thể mô tả trạng thái mong muốn cho các container được triển khai dùng Kubernetes và nó có thể thay đổi trạng thái thực tế sang trạng thái mong muốn với tần suất được kiểm soát. Ví dụ, bạn có thể tự động hóa Kubernetes để tạo mới các container cho việc triển khai của bạn, xoá các container hiện có và áp dụng tất cả các resource của chúng vào container mới.
- **Đóng gói tự động:** bạn cung cấp cho Kubernetes một cluster gồm các node mà nó có thể sử dụng để chạy các tác vụ được đóng gói (containerized task). Bạn cho Kubernetes biết mỗi container cần bao nhiêu CPU và bộ nhớ (RAM). Kubernetes có thể điều phối các container đến các node để tận dụng tốt nhất các resource của bạn.
- **Tự phục hồi:** Kubernetes khởi động lại các containers bị lỗi, thay thế các container, xoá các container không phản hồi lại cấu hình health check do người dùng xác định và không cho các client biết đến chúng cho đến khi chúng sẵn sàng hoạt động.
- **Quản lý cấu hình và bảo mật:** Kubernetes cho phép bạn lưu trữ và quản lý các thông tin nhạy cảm như: password, OAuth token và SSH key. Bạn có thể triển khai và cập nhật lại secret và cấu hình ứng dụng mà không cần build lại các container image và không để lộ secret trong cấu hình stack của bạn.

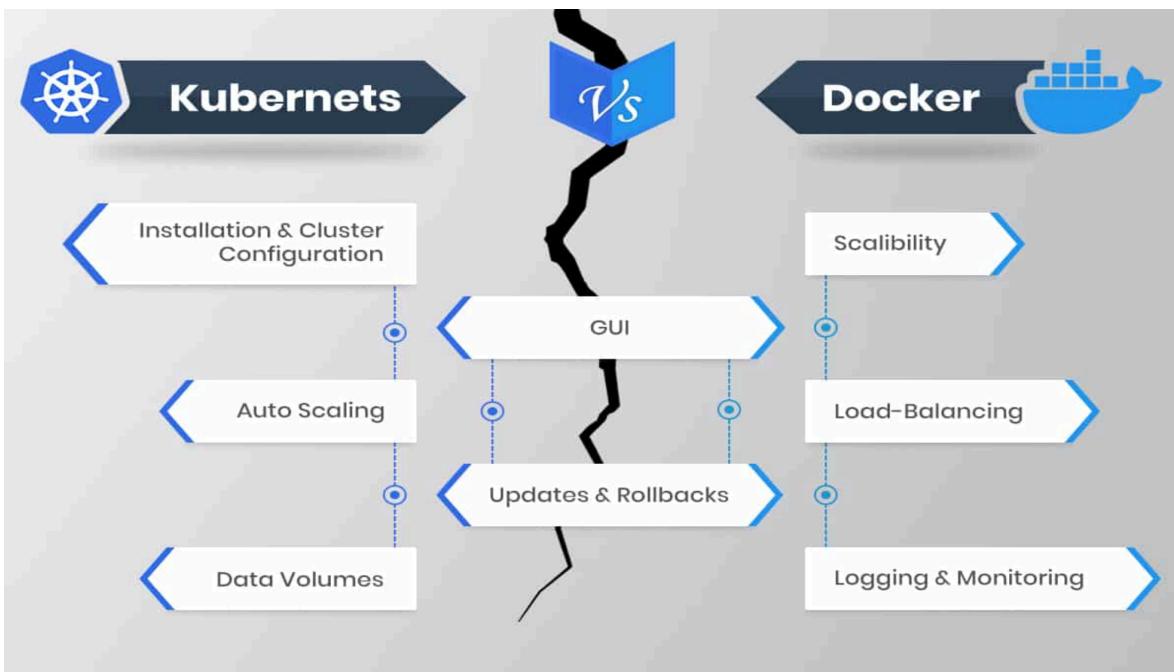
2.9.3. Sự khác nhau giữa Kubernetes và Docker Swarm



Hình 2-8: Kiến trúc của Docker Swarm

Docker Swarm: là một nhóm các máy chạy Docker và tập hợp lại với nhau thành một cluster. Không như docker engine, sau khi các máy này tập hợp vào Swarm, mọi câu lệnh Docker sẽ được thực thi trên Swarm manager.

- Các máy tham gia vào swarm được gọi là worker node. Các node này chỉ có khả năng cung cấp khả năng hoạt động chứ không có quyền quản lý các node khác.
- Có khả năng khởi chạy các container trên nhiều máy (cluster - *máy ảo hoặc máy vật lý*) hoặc trên một máy duy nhất (*standalone*).



Hình 2-9: So sánh Kubernetes với Docker Swarm

Cả Kubernetes và Docker Swarm đều là hai trong số các nền tảng mã nguồn mở được sử dụng nhiều nhất, cung cấp hầu hết các tính năng tương tự. Tuy nhiên, khi xem xét kỹ hơn, có thể nhận thấy một số khác biệt cơ bản giữa hai chức năng này.

Dưới đây là những điểm chú ý:

- Định nghĩa ứng dụng

- Trong Kubernetes, một ứng dụng có thể được deploy bằng cách sử dụng kết hợp các pod, deployments và services (hoặc micro-services).
- Trong khi đối với Docker Swarm, các ứng dụng có thể được deploy như một service (hoặc micro-service) trong một Swarm cluster. File YAML có thể được dùng để cụ thể hoá multi-container. Hơn nữa, Docker Compose có thể deploy ứng dụng.

- Install và set-up

- Trong Kubernetes, bước install được thực hiện thủ công và cần có kế hoạch cụ thể để Kubernetes hoạt động trơn tru. Hướng dẫn cài đặt thường được không thống nhất giữa các nhà cung cấp. Ngoài ra, cần

năm được cấu hình cluster như địa chỉ IP của một node hoặc nhiệm vụ của mỗi node.

- Trái với Kubernetes, việc cài đặt Docker Swarm rất đơn giản. Với Docker, chỉ cần một bộ công cụ tùy chọn để build theo môi trường và cấu hình. Docker Swarm cũng cung cấp tính linh hoạt bằng cách cho phép tất cả các node mới tham gia vào một cluster hiện có với tư cách là manager hoặc worker.

- **Yêu cầu**

- Kubernetes yêu cầu kiến thức về CLI (Command Line Interface) để chạy trên Docker. Cần hiểu về Docker CLI để điều hướng bên trong một cấu trúc, sau đó bổ sung infrastructure ngôn ngữ chung Kubernetes để chạy các program đó.
- Với lợi thế là một công cụ của Docker, Docker Swarm sử dụng ngôn ngữ chung để điều hướng trong một cấu trúc. Điều này cung cấp tính biến thiên và tốc độ cho công cụ này.

- **Khả năng thay đổi quy mô**

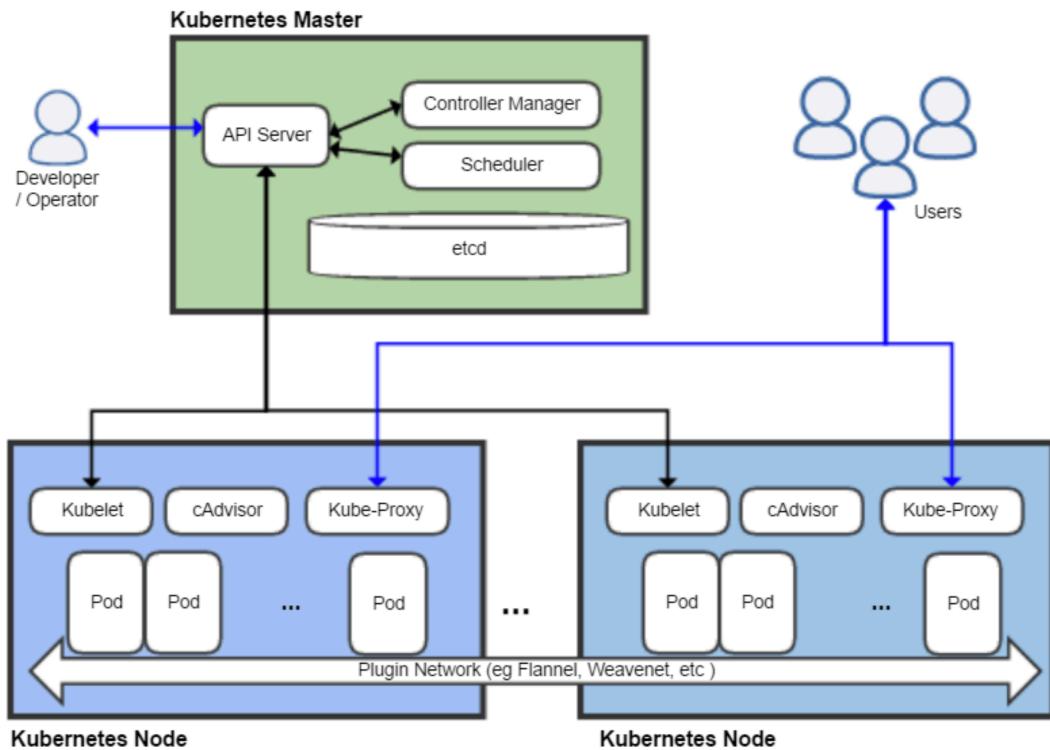
- Kubernetes là một all-in-one framework cho các hệ thống phân tán. Đây là một hệ thống phức tạp vì nó cung cấp một bộ API thống nhất và đảm bảo mạnh mẽ về trạng thái cluster, làm chậm deployment và thay đổi quy mô container.
- So với Kubernetes, Docker Swarm có thể deploy các container nhanh hơn; điều này cho phép thời gian phản ứng nhanh để thay đổi quy mô theo yêu cầu.

- **Tính sẵn sàng cao**

- Trong Kubernetes, tất cả các pod được phân phối giữa các node và điều này cung cấp tính sẵn sàng cao bằng cách chấp nhận lỗi ứng dụng. Hơn nữa, các load-balancing của Kubernetes phát hiện các pod không lành mạnh và loại bỏ chúng, điều này hỗ trợ tính sẵn sàng cao.

- Docker Swarm cũng cung cấp tính sẵn sàng cao vì các service có thể được nhân bản trong các node Swarm. Trong Docker Swarm, các node quản lý Swarm chịu trách nhiệm cho toàn bộ cluster và quản lý tài nguyên của các node worker.

2.9.4. Kiến trúc của kubernetes



Hình 2-10: Kiến trúc của Kubernetes

Kiến trúc Kubernetes có các thành phần chính sau:

- Master Node
- Worker / Slave Node
- Key-value Store (v.v.)

Master Node: (hay node chính) là nơi đảm nhiệm tất cả các tác vụ quản trị chịu trách nhiệm quản lý cụm Kubernetes. Có thể có nhiều hơn một node chính trong cụm để tăng khả năng chịu lỗi. Việc có nhiều hơn một node master giúp cụm

kubernetes có tính sẵn sàng cao, trong đó một trong số chúng sẽ là node chính mà chúng ta thực hiện tất cả các tác vụ.

Để quản lý trạng thái cụm, nó sử dụng **etcd** trong đó tất cả các node master kết nối với nó.

Có thể thấy trong sơ đồ, nó bao gồm 4 thành phần:

- **etcd cluster:** là thành phần cơ bản cần thiết cho Kubernetes, nó lưu trữ các cấu hình chung cho cả cụm máy, etcd chạy tại máy master. etcd là một dự án nguồn mở, nó cung cấp dịch vụ lưu trữ dữ liệu theo cặp key/value
- **kube-apiserver:** kubernetes API là thực thể quản lý trung tâm nhận tất cả các yêu cầu REST để sửa đổi (đối với pod, service, bộ sao chép / bộ điều khiển và những thứ khác), đóng vai trò là giao diện người dùng cho cụm. Ngoài ra, đây là thành phần duy nhất giao tiếp với cụm etcd, đảm bảo dữ liệu được lưu trữ trong etcd và phù hợp với chi tiết dịch vụ của các pod được triển khai.
- **kube-controller-manager:** là tiến trình chạy các controller để xử lý các background task của Cluster, giúp căn chỉnh cluster vào đúng với trạng thái đã khai báo (declare) ở Resource.
- **cloud-controller-manager:** chịu trách nhiệm quản lý các tiến trình của bộ điều khiển với sự phụ thuộc vào nhà cung cấp cloud (nếu có).
- **kube-scheduler:** Giúp lập lịch các pod trên các node khác nhau dựa trên việc sử dụng tài nguyên. Nó đọc các yêu cầu hoạt động của dịch vụ và lên lịch trên node phù hợp nhất.

Worker Node: là một máy chủ hay có thể nói là một máy ảo chạy các ứng dụng sử dụng các Pod được điều khiển bởi node Master.

Trên node worker, các pod được lập lịch. Để truy cập các ứng dụng từ thế giới bên ngoài, chúng ta kết nối với chúng qua các node.

Các thành phần của worker node:

- **kubelet:** là service chính trên mỗi node, thường xuyên nhận các thông số của pod mới hoặc được sửa đổi (chủ yếu thông qua kube-apiserver) và đảm bảo

rằng các pod và container của chúng không có vấn đề gì và chạy ở trạng thái mong muốn. Thành phần này cũng báo cáo cho master về tình trạng của node nơi mà nó đang chạy.

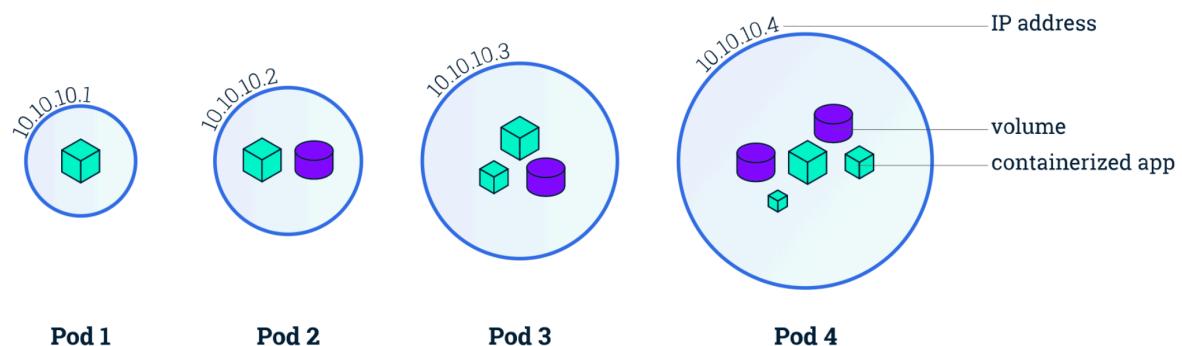
- **kube-proxy:** một dịch vụ proxy chạy trên mỗi worker node để xử lý vấn đề về mạng trên mỗi worker node và expose các port của service với bên ngoài internet. Nó thực hiện chuyển tiếp yêu cầu đến các pod/container chính xác trên các network bị cô lập khác nhau trong một cụm.

2.9.5. Các thành phần cơ bản của Kubernetes

Một số thành phần chính và quan trọng của k8s bao gồm:

Node: là đơn vị nhỏ nhất xét về phần cứng. Nó là một máy vật lý hay máy ảo (VPS) trong cụm máy (cluster). Đây là nơi container thực sự được triển khai để chạy.

Pods: k8s không chạy các container một cách trực tiếp, thay vào đó nó bọc một hoặc một vài container vào với nhau trong một cấu trúc gọi là POD. Các container trong thì sẽ chia sẻ với nhau tài nguyên và mạng cục bộ của pod.



Hình 2-11: Ảnh minh họa Pods

- Pod là thành phần đơn vị nhỏ nhất để k8s thực hiện việc nhân bản (replication), khi cần thiết thì k8s có thể cấu hình để triển khai nhân bản ra nhiều pod có chức năng giống nhau để tránh quá tải, hoặc có thể tạo ra nhiều bản copy của pod khi không quá tải nhằm phòng lỗi.
- Pod có thể có nhiều container mà pod là đơn vị scale nên nếu có thể thì cấu hình ứng dụng sao cho một pod có ít container nhất càng tốt.

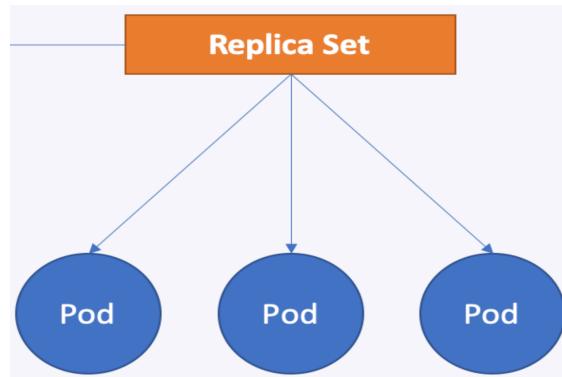
kubectl: là cú pháp lệnh, dùng để tương tác với Cluster, cú pháp chính:

```
kubectl [command] [TYPE] [NAME] [flags]
```

- Trong đó:

- [command] là lệnh, hành động như apply, get, delete, describe, ...
- [TYPE] kiểu tài nguyên như ns, no, po, svc, ...
- [NAME] tên đối tượng lệnh tác động
- [flags] các thiết lập, tùy thuộc loại lệnh

ReplicaSet: là một điều khiển controller, nó đảm bảo ổn định các nhân bản (số lượng, tình trạng của POD, replica) khi đang chạy.

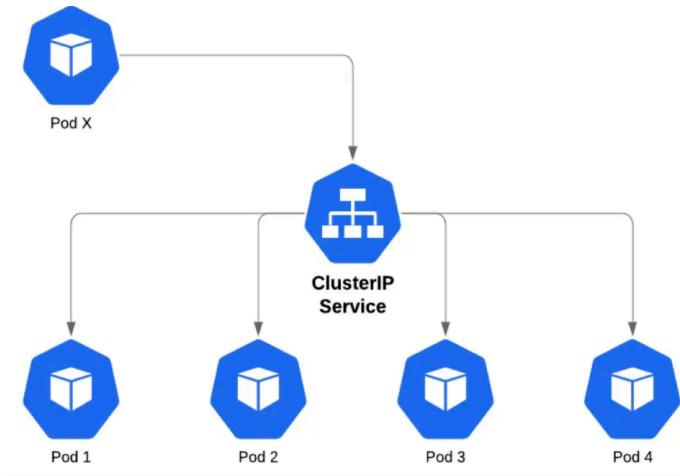


Hình 2-12: Ảnh minh họa ReplicaSet

HPA: (Horizontal Pod Autoscaler) là chế độ tự động scale (nhân bản pod) dựa vào mức độ hoạt động của CPU đối với POD, nếu một pod quá tải thì nó có thể nhân bản thêm pod khác và ngược lại. Số nhân bản dao động trong khoảng min, max cấu hình.

Deployment: dùng để quản lý một nhóm các pod – các pod được nhân bản, nó tự động thay thế các pod bị lỗi, không phản hồi bằng các pod mới nó tạo ra. Đồng thời đảm bảo ứng dụng có một hoặc nhiều Pod để phục vụ các yêu cầu.

Service:

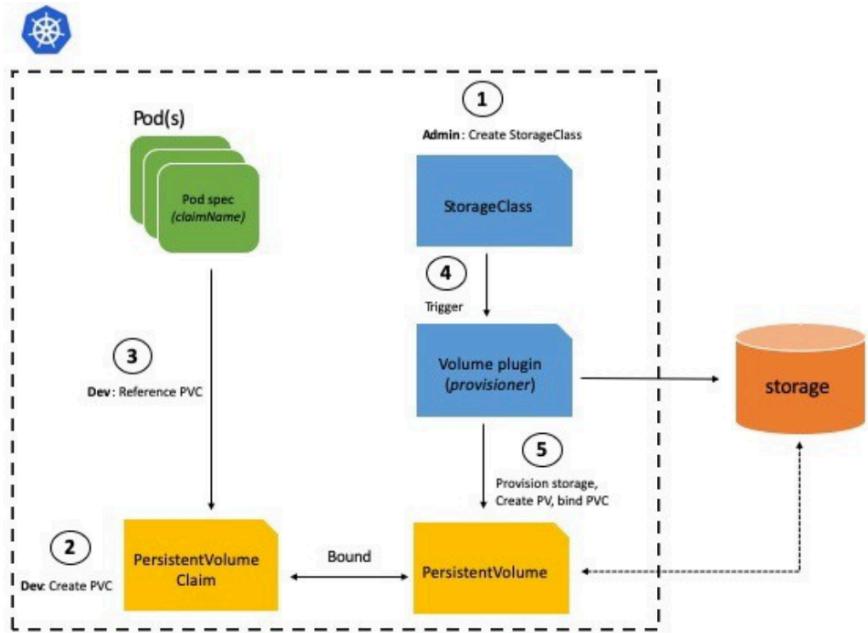


Hình 2-13: Ảnh minh họa Service

- Các POD được quản lý trong k8s, trong vòng đời chỉ diễn ra theo hướng: tạo ra, chạy, khi kết thúc thì bị xoá và khởi tạo lại POD mới thay thế. Không thể tạm dừng POD đang chạy.
- Mỗi POD khi tạo ra có một IP để liên lạc, tuy vậy vẫn đề là mỗi khi POD thay thế thì sẽ có một IP khác, nên các dịch vụ truy cập không biết IP mới nếu ta cấu hình nó truy cập POD nào đó cố định.
=> Service sinh ra để giải quyết những vấn đề trên.
- Service là một đối tượng trừu tượng nó xác định ra một nhóm các POD và chính sách để truy cập đến POD đó. Nhóm các POD mà service xác định thường dùng kỹ thuật selector (chọn các POD thuộc về service theo label của POD)
- Service cũng là một dịch vụ mạng, tạo ra cơ chế cân bằng tải truy cập đến các điểm cuối mà service đó phục vụ.

DaemonSet: đảm bảo chạy trên mỗi NODE một bản copy của POD. Triển khai DaemonSet khi cần ở mỗi máy (Node) một POD, thường dùng cho các ứng dụng thu thập log, tạo ổ đĩa trên mỗi Node.

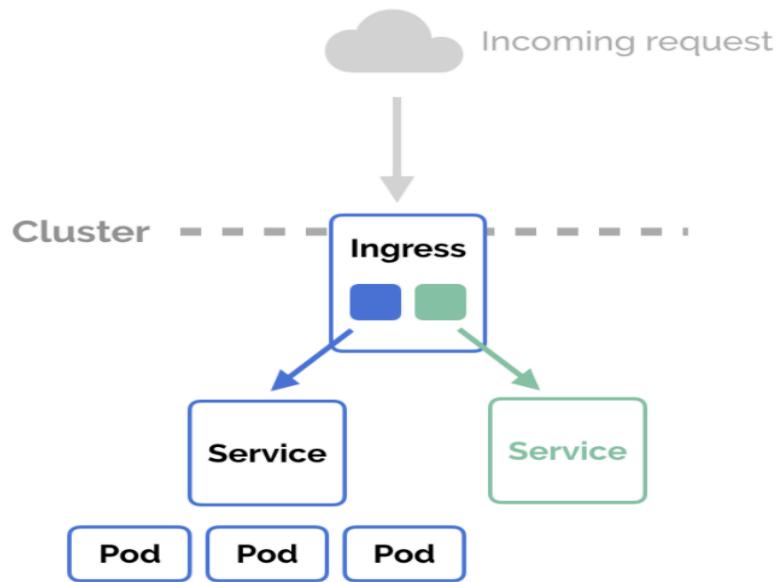
PV, PVC:



Hình 2-14: Ảnh minh họa PV, PVC

- PersistentVolume (pv): là một thành phần không gian lưu trữ dữ liệu trong cluster, các PersistentVolume giống với Volume bình thường, tuy nhiên nó tồn tại độc lập với các POD (pod bị xoá PV vẫn tồn tại), có nhiều loại PV có thể triển khai như: NFS, Clusterfs, ...
- PersistentVolumeClaim (pvc): là yêu cầu sử dụng không gian lưu trữ (sử dụng PV). PVC hoạt động và sử dụng tài nguyên của PV.

Ingress:



Hình 2-15: Ảnh minh họa Ingress

- là thành phần được dùng để điều hướng các yêu cầu traffic giao thức HTTP, HTTPS từ bên ngoài vào các dịch vụ bên trong Cluster.
- Ingress chỉ để phục vụ các cổng, yêu cầu HTTP, HTTPS còn các loại cổng khác.

2.9.6. Cách cài đặt và hệ thống hỗ trợ

K8s là một hệ thống gồm nhiều thành phần tương tác với nhau. Có rất nhiều cách để cài đặt k8s, đáng kể nhất:

- Minikube: để cài 1 cluster test, hoặc sử dụng trên localhost.
- kubedm: đang trong giai đoạn phát triển, để cài trên hệ thống máy vật lý/máy ảo dùng ubuntu hay centos.
- Kargo: phần mềm dựa trên Ansible để cài trên rất nhiều nơi bao gồm cả máy ảo và vật lý, AWS, GKE, ...
- Cài bằng tay trên CoreOS container Linux.
- Một số hệ thống lớn cũng cung cấp kubernetes cài đặt sẵn:
 - Google Kubernetes Engine (GKE): miễn phí 300\$ cho tài khoản mới, có thể đăng ký và thử nghiệm

- DigitalOcean Kubernetes: đăng ký sử dụng với chỉ 10\$/1 tháng/ 1 cụm máy.
- Amazon EKS
- Azure Kubernetes Service (AKS)

Chương 3. XÂY DỰNG HỆ THỐNG

3.1. Xây dựng kiến trúc hệ thống

3.1.1. Xác định yêu cầu hệ thống

Qua khảo sát tìm hiểu, tham khảo các ứng dụng TMĐT hiện nay, sau khi phân tích nhóm quyết định mô hình xử lý hàng hoá như sau:



Hình 3-1: Quy trình xử lý hàng hoá của FLAD

Nhập kho:

- Nhà cung cấp chờ hàng đến
- Nhân viên kiểm tra hàng sau đó phân loại, sắp xếp
- Nhân viên cập nhật hàng lên ứng dụng FLAD thông qua ứng dụng FLAD

Bán hàng:

- Khách hàng comment theo cú pháp khi livestream đang phát
- Hệ thống tự động nhận diện comment đúng cú pháp và thêm vào đơn hàng
- Nhân viên chốt đơn hàng và tiến hành các thủ tục giao hàng đến khách hàng

Đóng gói:

- Nhân viên lấy hàng trong kho theo đơn hàng và đóng gói

Vận chuyển:

- Nhân viên giao hàng chuyển hàng theo địa chỉ của khách hàng
- Nhân viên giao hàng mang biên lai về cửa hàng để báo cáo

Hậu mãi:

- Nhân viên giải đáp thắc mắc của khách hàng
- Xử lý hàng hóa không có người nhận
- Xử lý hàng hóa bị trả lại

Từ dữ liệu thu thập được trong quá trình khảo sát các ứng dụng tương tự hiện nay như T-POS, Misa, ứng dụng sẽ được chia thành 2 phần và gồm những chức năng chính sau đây:

Phần người dùng (chủ cửa hàng):

- + Quản lý đơn hàng
- + Quản lý sản phẩm
- + Quản lý livestreams
- + Quản lý khuyến mãi
- + Phản hồi tin nhắn khách hàng
- + Thống kê doanh thu

Phần quản trị viên:

- + Quản lý người dùng

- + Các chỉ số thống kê (người dùng,)

3.1.2. Phân tích yêu cầu hệ thống

3.1.2.1. Người dùng (cửa hàng)

– Quản lý sản phẩm:

- + **Thêm:** Thêm các sản phẩm mới vào các danh mục mặt hàng tương ứng, gồm các thông tin cơ bản như: tên sản phẩm, ảnh, thông tin mô tả, thông số kỹ thuật, **các loại mặt hàng:** màu sắc, kích thước, số lượng, giá.
- + **Cập nhật:** Người bán hàng sửa lại các thông tin của sản phẩm. cũng như thêm sản phẩm.
- + **Xóa:** Xóa các sản phẩm không còn kinh doanh.

– Quản lý đơn hàng:

- + **Đã tạo đơn :** Là trạng thái mặc định khi có đơn hàng mới.
- + **Đã chốt đơn:** Sau khi xác nhận đơn hàng với khách hàng.
- + **Đã xác nhận thông tin:** Xác nhận thông tin giao hàng.
- + **Đang giao:** Nhân viên giao hàng đang trong quá trình đi giao hàng cho khách hàng đó.
- + **Đã giao:** Sản phẩm đã được giao thành công cho khách hàng.
- + **Đã hủy:** Hủy các đơn hàng không hợp lệ (thông tin không hợp lệ, huỷ theo yêu cầu của khách hàng,...).
- + **Trả hàng:** Khi sản phẩm đến tay người dùng nhưng bị lỗi, sản phẩm hư hỏng trong quá trình giao hoặc do một số nguyên nhân khách quan thì người dùng sẽ liên hệ đến chủ shop (người bán) để tiến hành trả hàng. Hàng sẽ được nhân viên đến lấy và trả về kho.
- + **Bom hàng:** người dùng không nhận hàng dù đơn hàng đúng với yêu cầu của khách hàng

– Thông kê doanh thu:

- + Trong chức năng thống kê doanh thu, người bán hàng sẽ thống kê theo khoảng thời gian mong muốn. Chức năng thống kê sẽ hiển thị tổng tiền, mã đơn hàng, ngày giao hàng, tên khách hàng.
- **Quản lý khuyến mãi:**
 - + Người dùng có thể thêm các mã khuyến mãi.
- **Quản lý danh mục:**
 - + **Thêm danh mục:** Cho phép thêm các danh mục sản phẩm
 - + **Chỉnh sửa danh mục:** Chức năng cho phép chỉnh sửa các thuộc tính danh mục.
 - + **Xóa danh mục:** Chức năng cho phép xóa danh mục.

3.1.2.2. Quản trị viên:

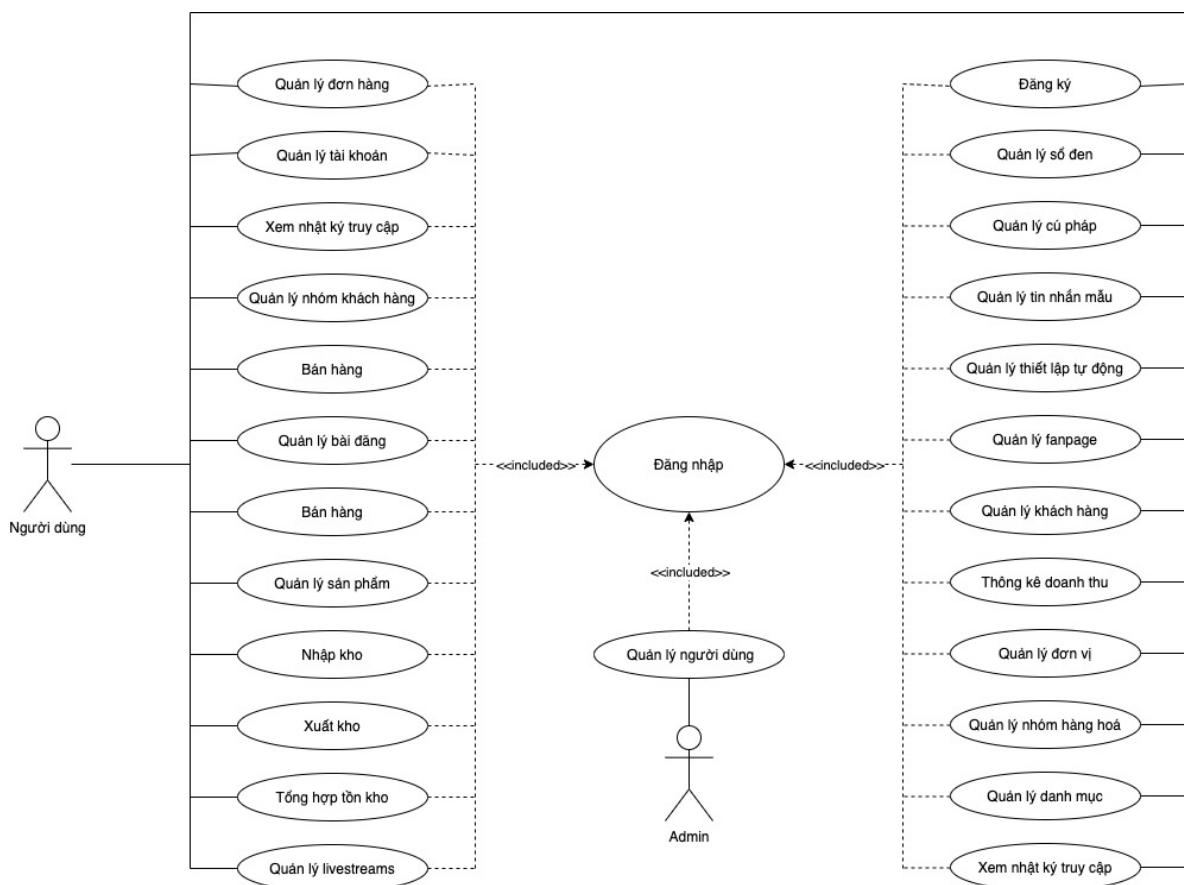
- **Quản lý người dùng.**
 - + **Phân quyền người dùng:** Người dùng có thể được thăng cấp thành quản trị viên để cùng admin quản lý ứng dụng.
 - + **Xem thông tin của tất cả người dùng:** Người quản trị có thể khóa và mở khóa người dùng. Người dùng bị khóa sẽ không thể đăng nhập vào ứng dụng FLAD.

3.2. Phân tích thiết kế hệ thống

3.2.1. Sơ đồ use case

3.2.1.1. Sơ đồ use case

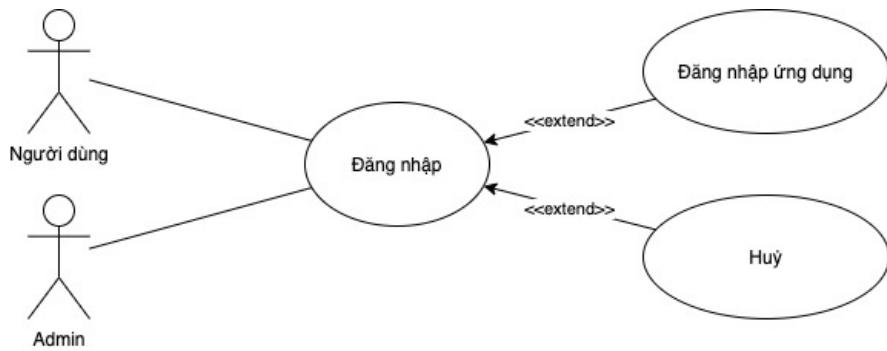
Use case mức tổng quát:



Hình 3-2: Biểu đồ use case mức tổng quát

Use case Đăng nhập:

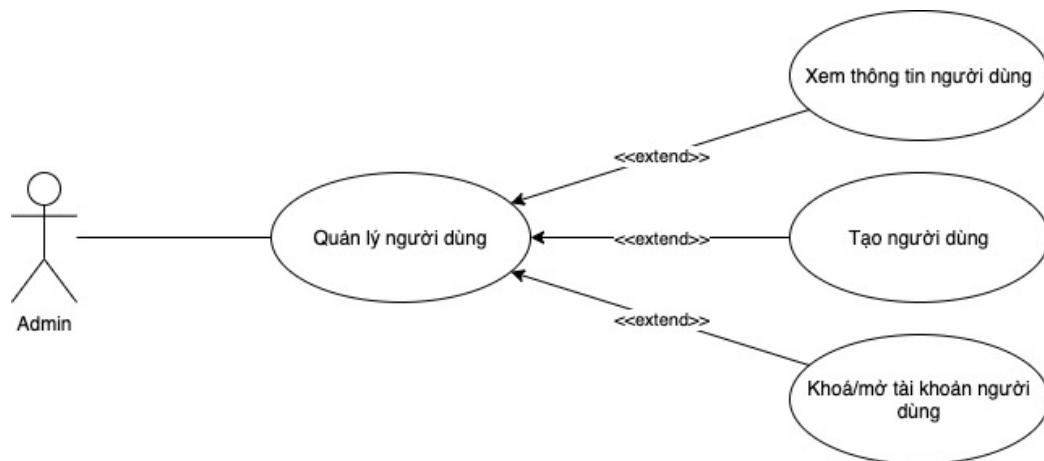
- Mục đích: đảm bảo xác thực thông tin người dùng và an toàn bảo mật hệ thống.
- Tác nhân: người dùng và Admin.
- Mô tả: người dùng đăng nhập vào ứng dụng để sử dụng các chức năng bán hàng, admin đăng nhập vào phần quản trị để quản lý người dùng.



Hình 3-3: Biểu đồ use case Đăng nhập

Use case Quản lý người dùng:

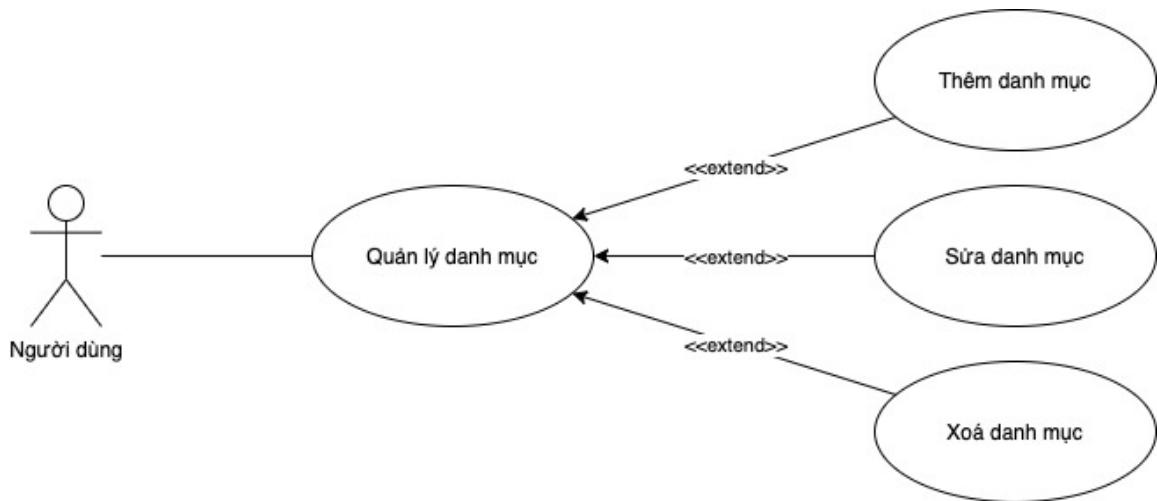
- Mục đích: quản lý người dùng cho ứng dụng.
- Tác nhân: Admin.
- Mô tả: sau khi đăng nhập vào phần quản trị, admin có thể xem thông tin người dùng, tạo người dùng, khoá/mở tài khoản người dùng



Hình 3-4: Biểu đồ use case Quản lý người dùng

Use case Quản lý danh mục sản phẩm:

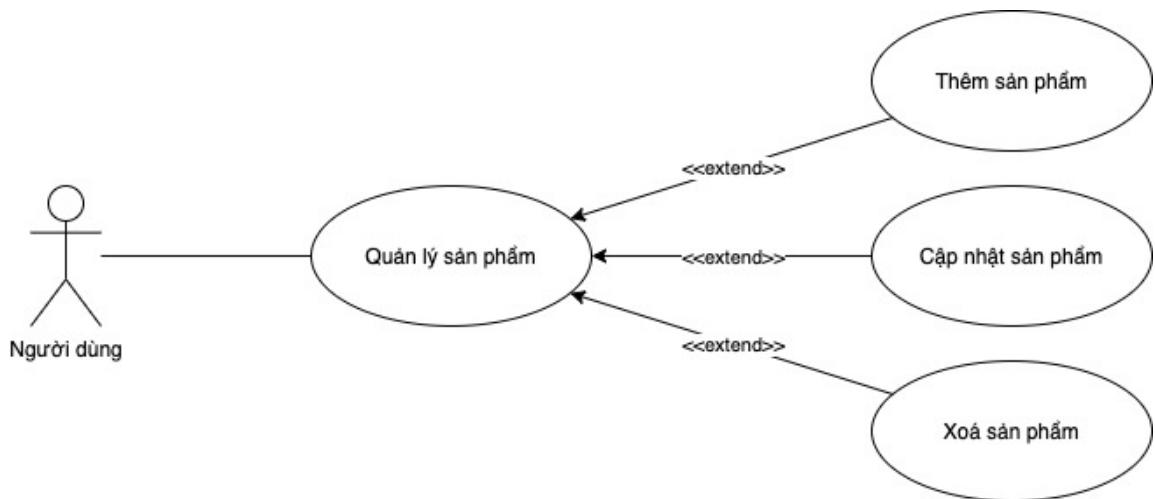
- Mục đích: quản lý danh mục sản phẩm cho ứng dụng.
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập người dùng tiến hành thêm, sửa, xoá danh mục sản phẩm



Hình 3-5: Biểu đồ use case Quản lý danh mục sản phẩm

Use case Quản lý sản phẩm:

- Mục đích: quản lý sản phẩm cho ứng dụng.
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa sản phẩm

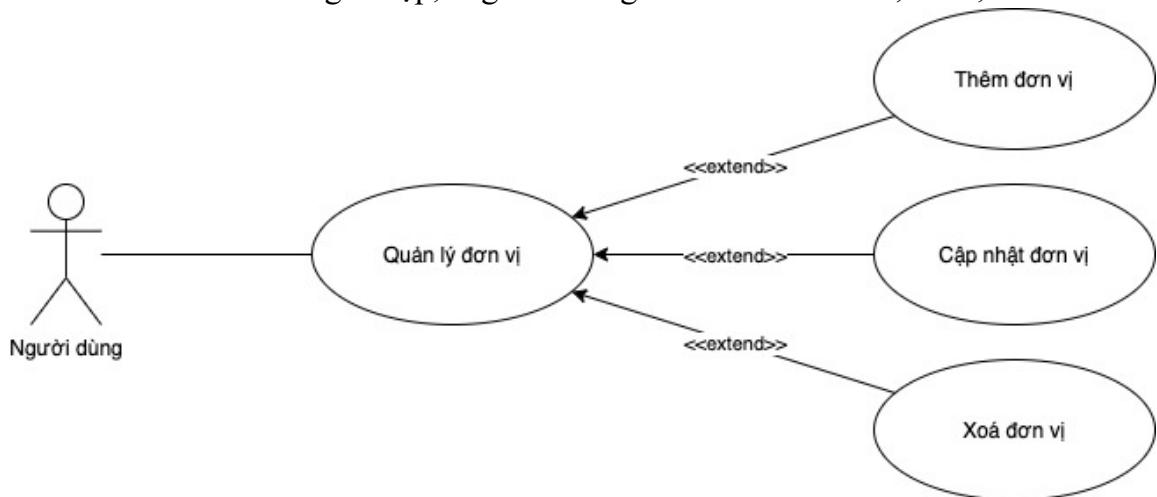


Hình 3-6: Biểu đồ use case Quản lý sản phẩm

Use case Quản lý đơn vị:

- Mục đích: quản lý đơn vị cho ứng dụng.
- Tác nhân: người dùng.

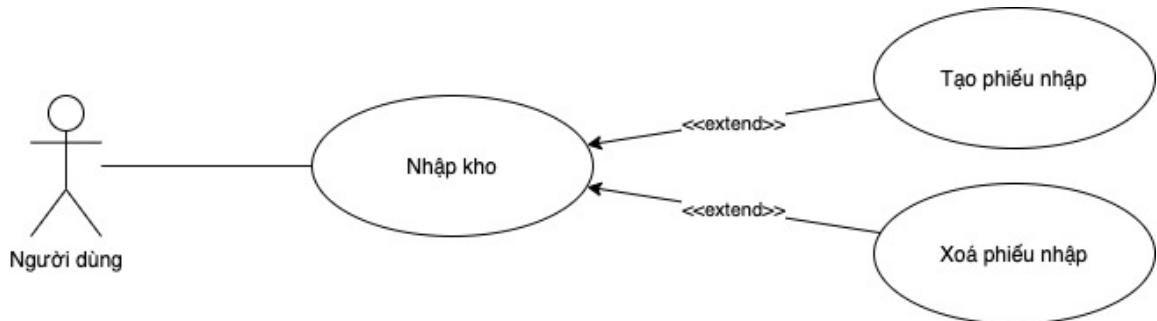
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa đơn vị



Hình 3-7: Biểu đồ use case Quản lý đơn vị

Use case Nhập kho:

- Mục đích: quản lý việc nhập kho
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa phiếu nhập kho

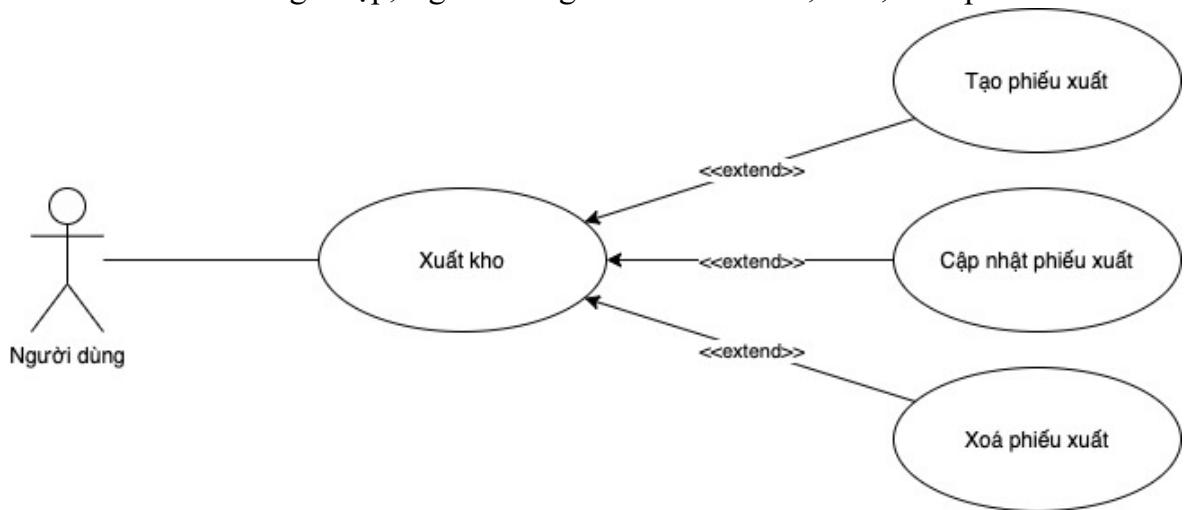


Hình 3-8: Biểu đồ use case Nhập kho

Use case Xuất kho:

- Mục đích: quản lý việc xuất kho
- Tác nhân: người dùng.

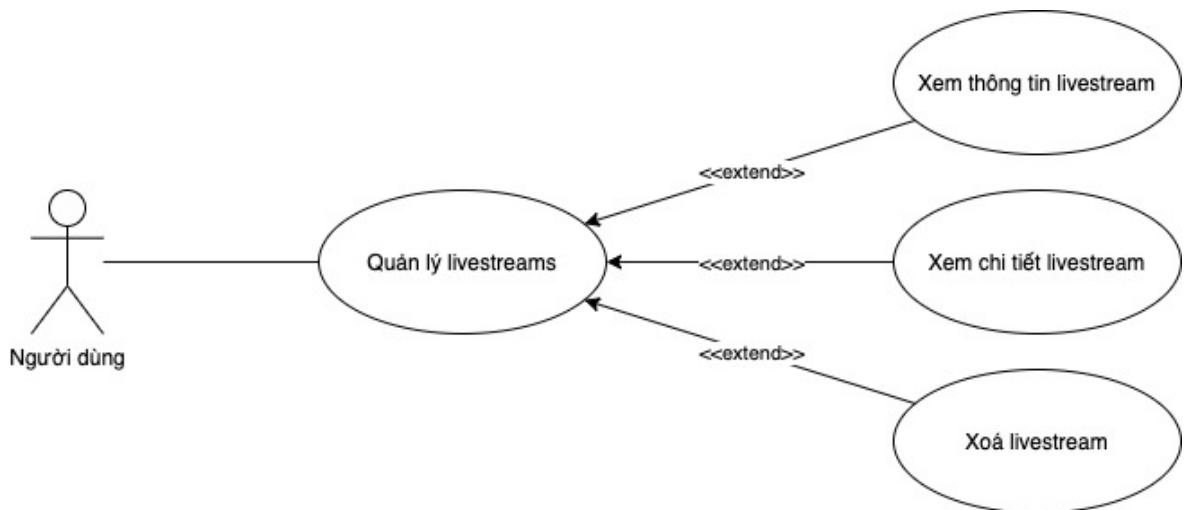
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa phiếu xuất kho



Hình 3-9: Biểu đồ use case Xuất kho

Use case Quản lý livestream:

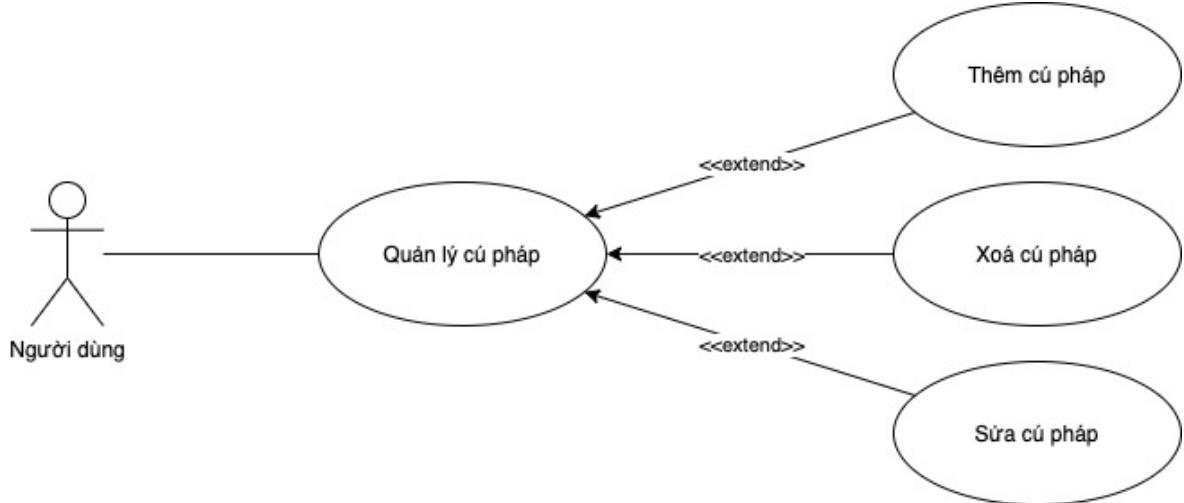
- Mục đích: quản lý các livestream trên page đã kết nối.
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành xem thông tin chi tiết các livestream



Hình 3-10: Biểu đồ use case Quản lý livestream

Use case Quản lý cú pháp:

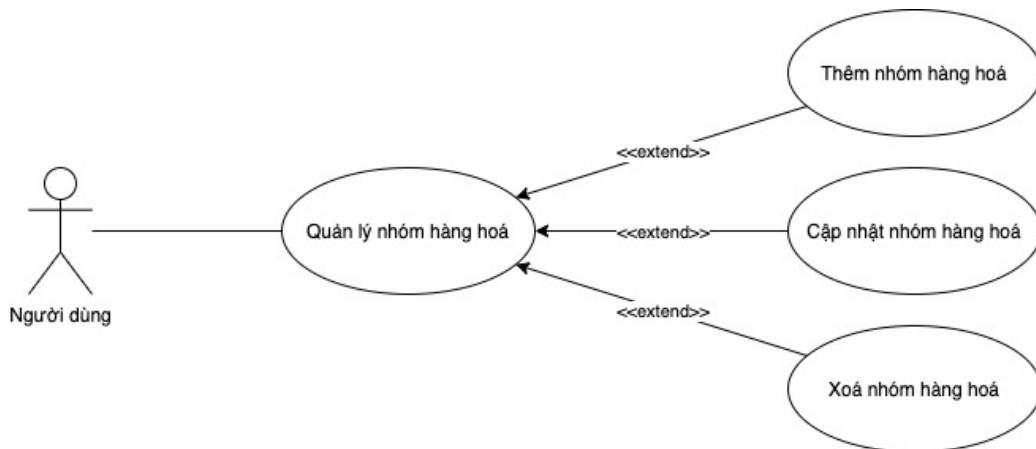
- Mục đích: quản lý cú pháp
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa nhóm hàng hoá



Hình 3-11: Biểu đồ use case Quản lý cú pháp

Use case Quản lý nhóm hàng hoá:

- Mục đích: quản lý nhóm hàng hoá
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa nhóm hàng hoá

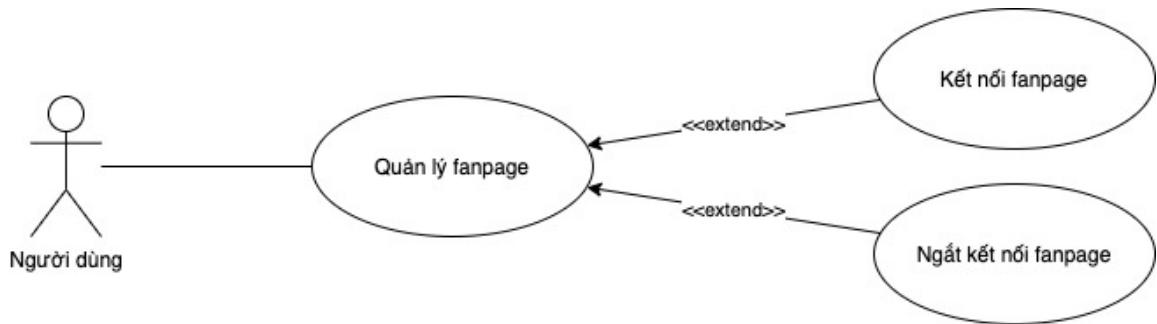


Hình 3-12: Biểu đồ use case Quản lý nhóm hàng hoá

Use case Quản lý fanpage:

- Mục đích: quản lý fanpage

- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành kết nối / ngắt kết nối fanpage

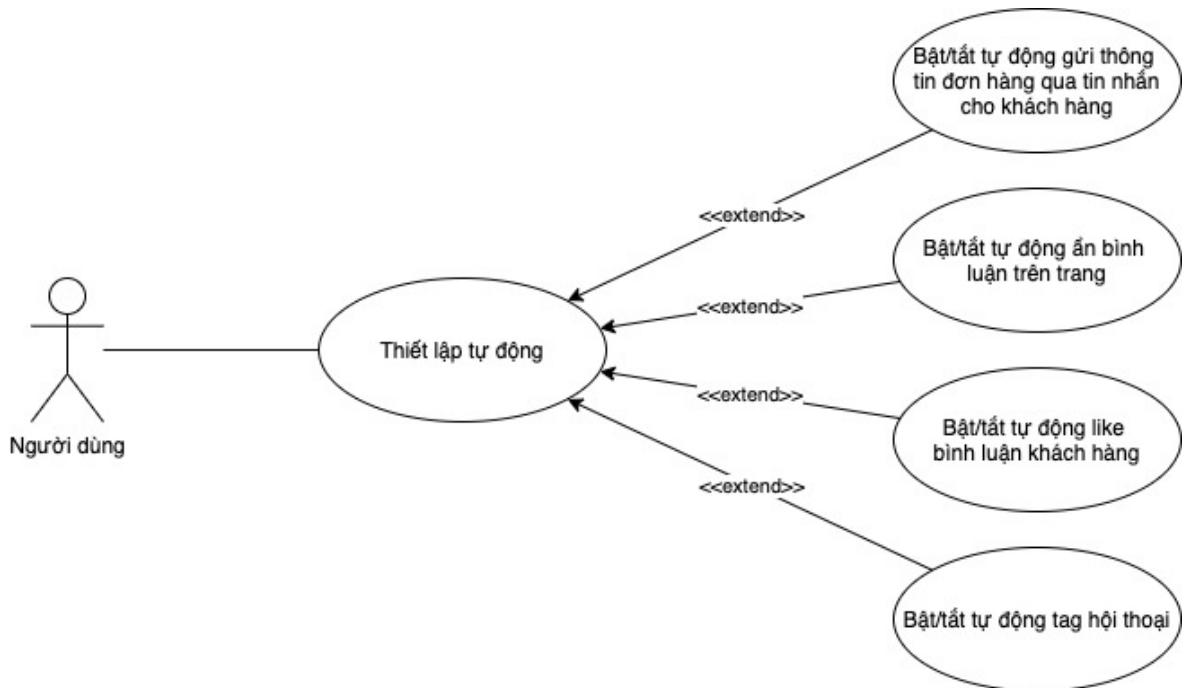


Hình 3-13: Biểu đồ use case Quản lý fanpage

Use case Thiết lập tự động:

- Mục đích: thiết lập tự động trong quá trình livestream
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành bật, tắt các tính năng: tự động gửi thông tin đươn hàng qua tin nhắn cho khách hàng, tự động ẩn bình luận trên trang, tự

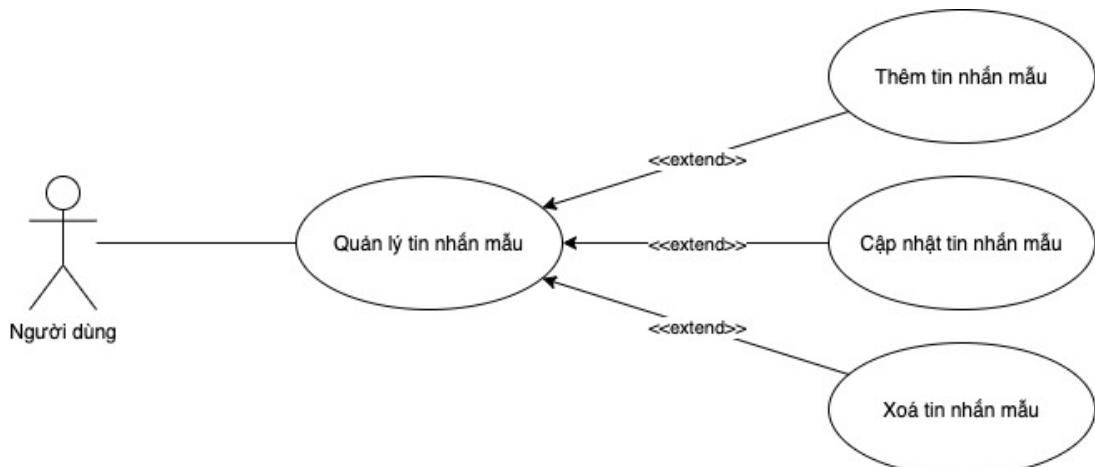
động like bình luận trên trang, tự động like bình luận khách hàng, tự động tag hội thoại



Hình 3-14: Biểu đồ use case Thiết lập tự động

Use case Quản lý tin nhắn mẫu:

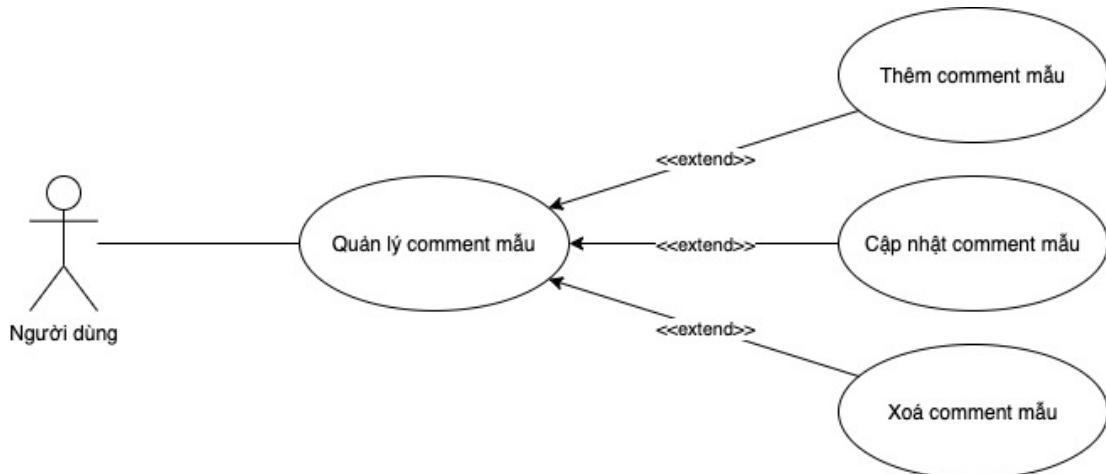
- Mục đích: quản lý tin nhắn mẫu
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa tin nhắn mẫu



Hình 3-15: Biểu đồ use case Quản lý tin nhắn mẫu

Use case Quản lý comment mẫu:

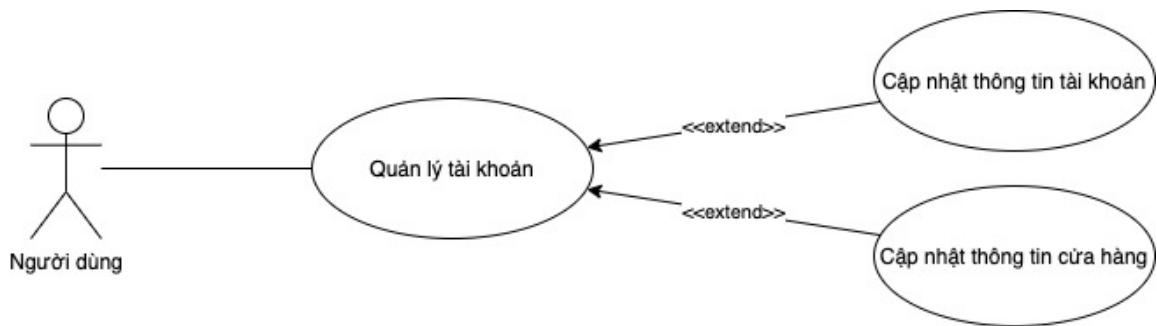
- Mục đích: quản lý comment mẫu
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa comment mẫu



Hình 3-16: Biểu đồ use case Quản lý comment mẫu

Use case Quản lý tài khoản:

- Mục đích: quản lý tài khoản
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành cập nhật thông tin tài khoản, cửa hàng

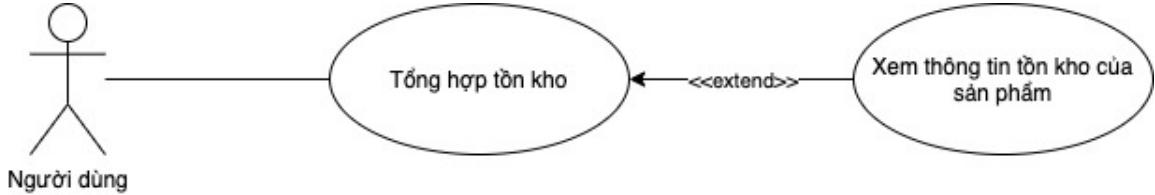


Hình 3-17: Biểu đồ use case Quản lý tài khoản

Use case tổng hợp tồn kho:

- Mục đích: tổng hợp tồn kho

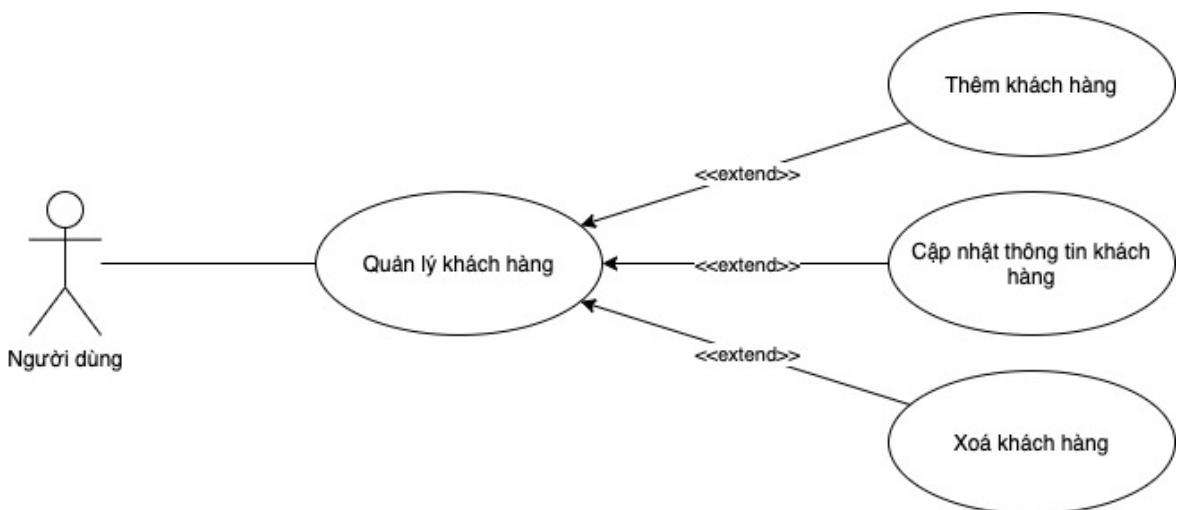
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành xem thông tin tồn kho của cửa hàng



Hình 3-18: Biểu đồ use case Tổng hợp tồn kho

Use case Quản lý khách hàng:

- Mục đích: quản lý khách hàng
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa khách hàng

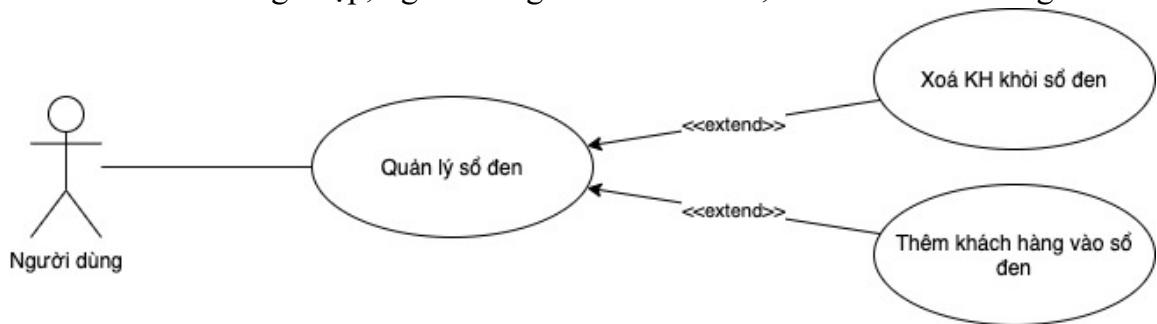


Hình 3-19: Biểu đồ use case Quản lý khách hàng

Use case Quản lý số đên:

- Mục đích: quản lý các khách hàng bị tố cáo vì lí do xấu
- Tác nhân: người dùng.

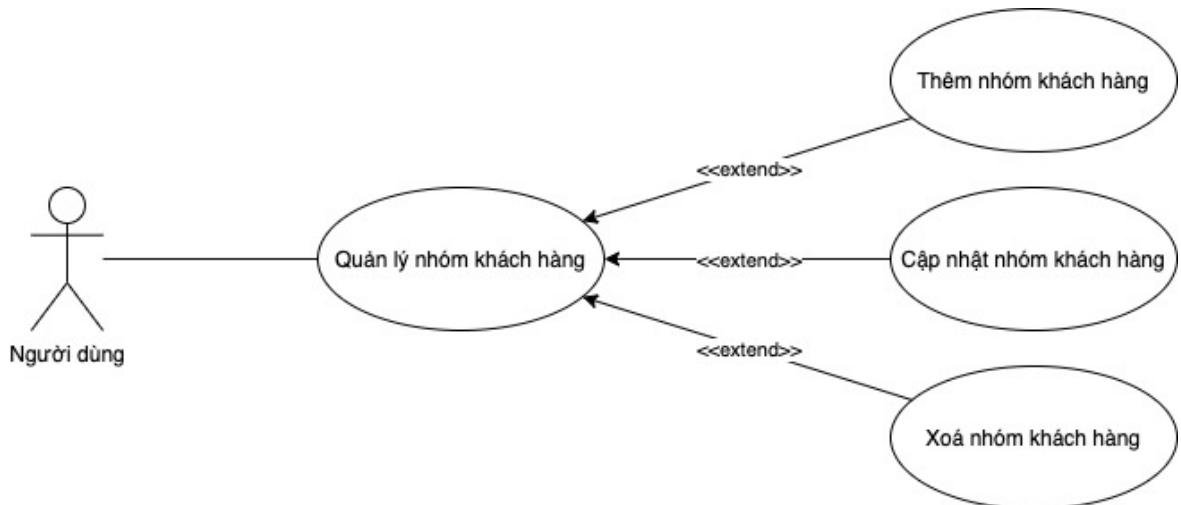
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá các khách hàng ở sổ đen



Hình 3-20: Biểu đồ use case Quản lý sổ đen

Use case Quản lý nhóm khách hàng:

- Mục đích: quản lý nhóm khách hàng
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thêm, xoá, sửa nhóm khách hàng

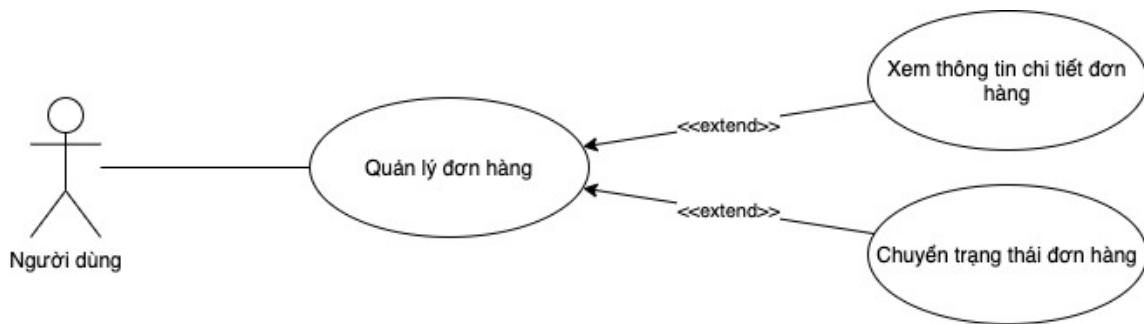


Hình 3-21: Biểu đồ use case Quản lý nhóm khách hàng

Use case Quản lý đơn hàng:

- Mục đích: quản lý các đơn hàng.
- Tác nhân: người dùng.

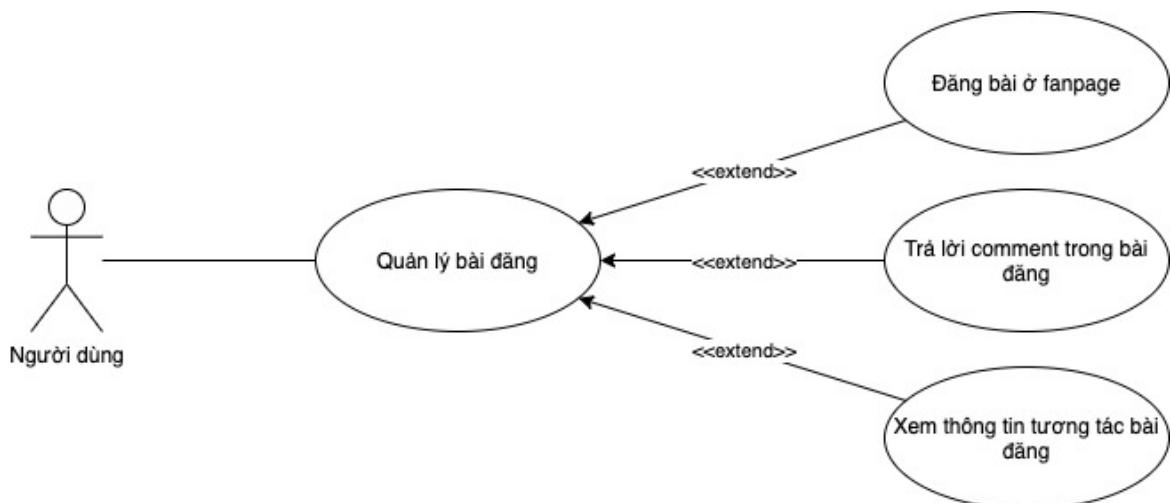
- Mô tả: sau khi đăng nhập vào ứng dụng, người dùng tiến hành cập nhật trạng thái đơn hàng



Hình 3-22: Biểu đồ use case Quản lý đơn hàng

Use case Quản lý bài đăng:

- Mục đích: quản lý bài đăng
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành đăng bài, trả lời comment (thủ công hoặc hàng loạt), xem thông tin tương tác của bài đã đăng

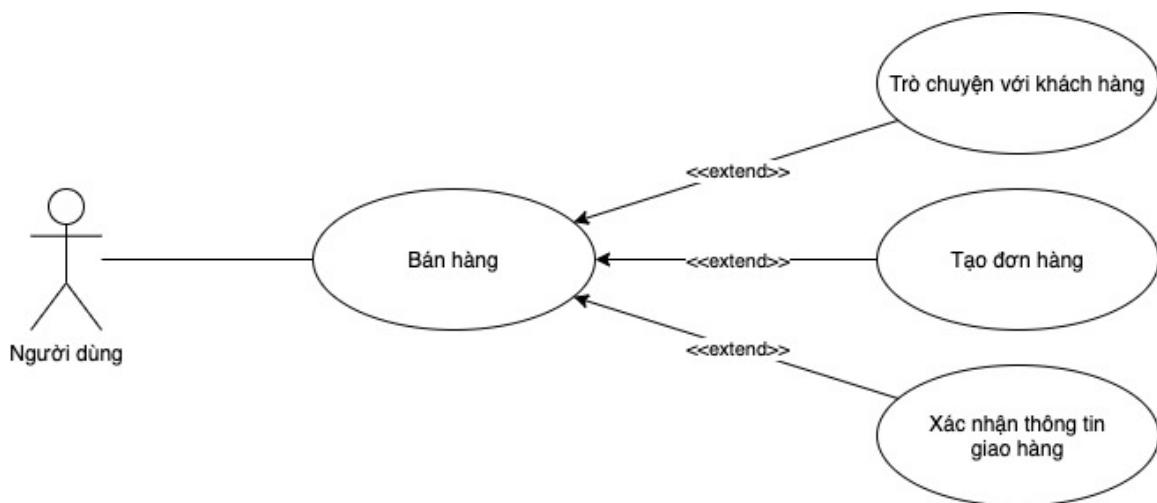


Hình 3-23: Biểu đồ use case Quản lý bài đăng

Use case Bán hàng:

- Mục đích: Người dùng bán hàng thông qua các chức năng của FLAD
- Tác nhân: người dùng.

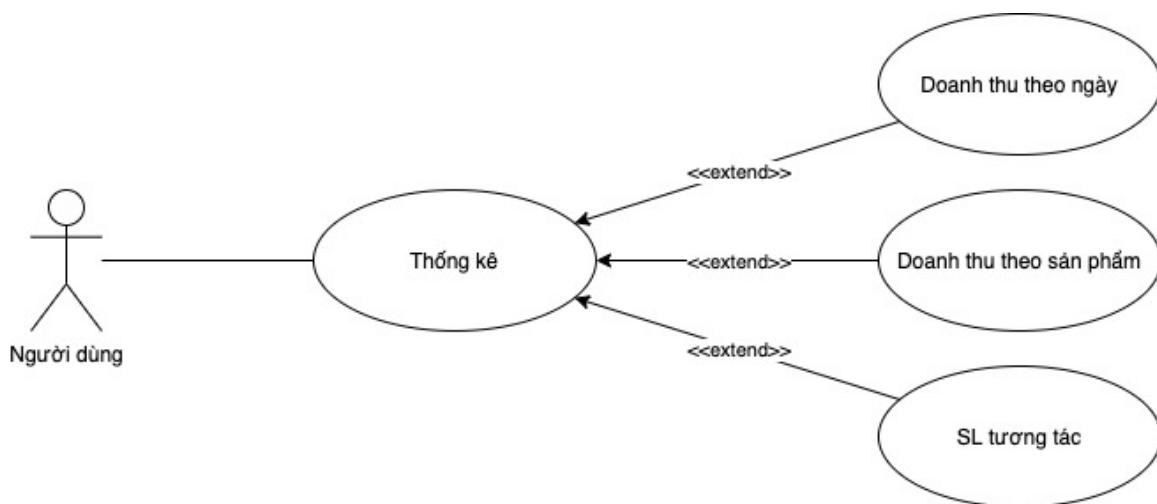
- Mô tả: sau khi đăng nhập, người dùng tiến hành bán hàng thông qua các chức năng: trò chuyện với khách hàng, tạo đơn hàng, xác nhận thông tin giao hàng



Hình 3-24: Biểu đồ use case Bán hàng

Use case Thống kê:

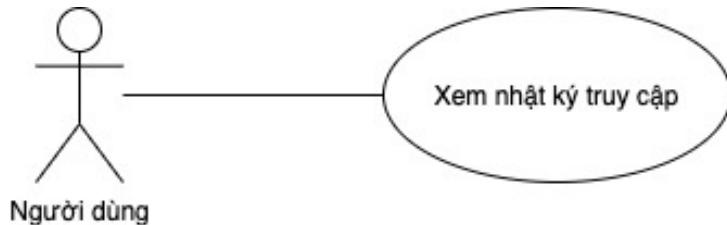
- Mục đích: thống kê các thông số
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành thống kê doanh thu theo ngày, sản phẩm, SL tương tác



Hình 3-25: Biểu đồ use case Thống kê

Use case Xem nhật ký truy cập:

- Mục đích: xem nhật ký truy cập
- Tác nhân: người dùng.
- Mô tả: sau khi đăng nhập, người dùng tiến hành xem nhật ký truy cập của hệ thống



Hình 3-26: Biểu đồ use case Xem nhật ký truy cập

3.2.1.2. Danh sách các actors

STT	Tên actor	Ý nghĩa
1	Quản trị viên (Admin)	<p>Admin có một tài khoản riêng để đăng nhập vào phần quản trị quản trị.</p> <p>Admin có thể cấp quyền cho người dùng khác trở thành admin.</p> <p>Ngoài ra, Admin có đầy đủ các quyền như người dùng khách hàng trên ứng dụng.</p>
2	Người dùng	<p>Người dùng là người đã có tài khoản và đăng nhập vào ứng dụng.</p> <p>Người dùng có thể bán hàng và sử dụng các chức năng khác để phục vụ cho việc bán hàng</p>

Bảng 3-1: Danh sách actors

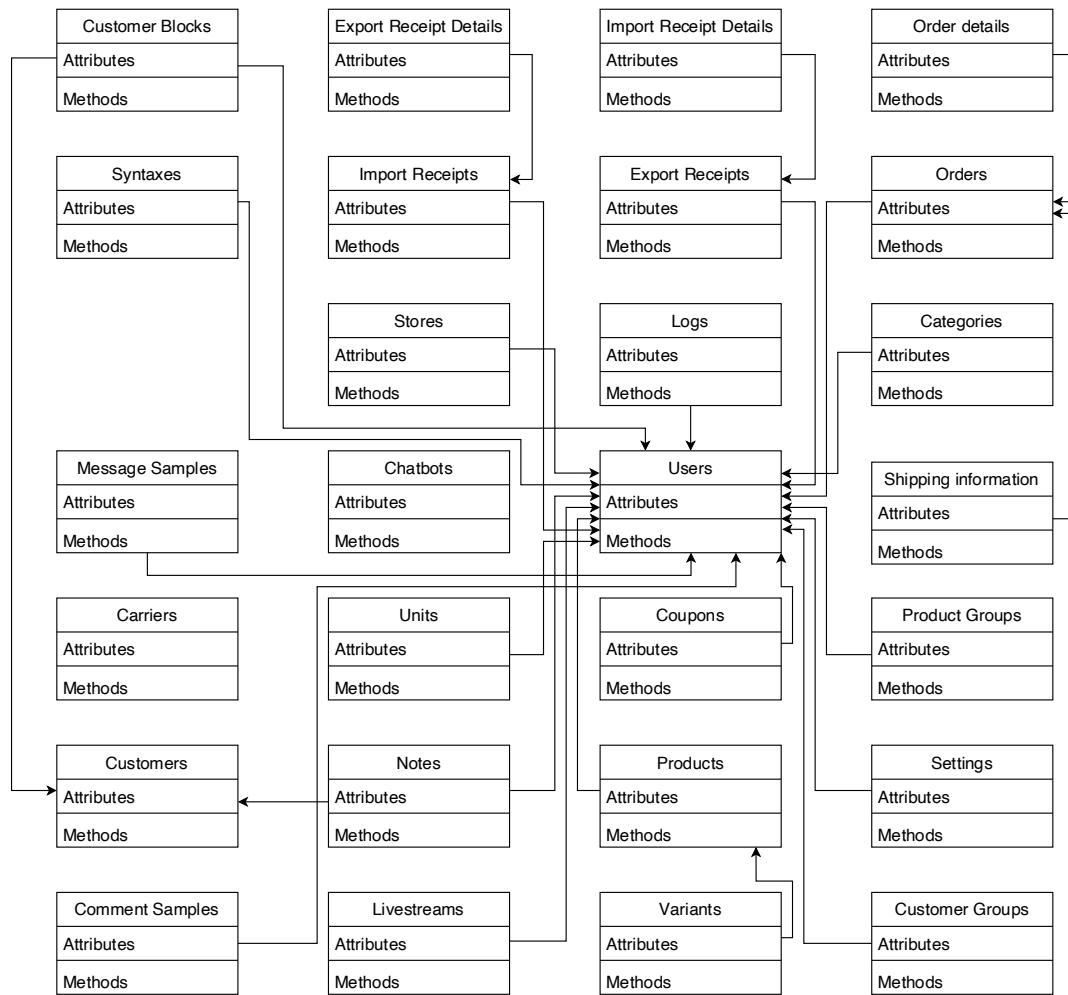
3.2.1.3. Danh sách các use cases

STT	Tên use case	Ý nghĩa
1	Use case Mức tổng quát	Đưa ra các actor có trong hệ thống quản lý, và chức năng chính của mỗi actor.
2	Use case Đăng nhập	Yêu cầu đăng nhập để xác thực trong ứng dụng
3	Use case Quản lý người dùng	Quản lý người dùng để ứng dụng hoạt động tốt hơn (đảm bảo tính công bằng)
4	Use case quản lý đơn vị	Quản lý đơn vị được dùng cho sản phẩm
5	Use case Quản lý danh mục sản phẩm	Quản lý các danh mục sản phẩm hàng hóa (thêm, sửa, xóa danh mục)
6	Use case Quản lý sản phẩm	Quản lý sản phẩm (thêm, sửa, xóa sản phẩm)
7	Use case Nhập kho	Quản lý việc Nhập kho của cửa hàng
8	Use case Xuất kho	Quản lý việc Xuất kho của cửa hàng
9	Use case Quản lý video livestream	Quản lý các livestream đã phát
10	Use case Quản lý cú pháp	Quản lý các cú pháp để hệ thống nhận diện comment của khách hàng
11	Use case Quản lý nhóm hàng hoá	Quản lý các nhóm hàng hoá để livestream
12	Use case Quản lý fanpage	Quản lý fanpage kết nối với FLAD
13	Use case Quản lý thiết lập tự động	Quản lý các thiết lập cho việc tự động hoá

14	Use case Quản lý tin nhắn mẫu	Quản lý tin nhắn mẫu
15	Use case Quản lý bình luận mẫu	Quản lý bình luận mẫu
16	Use case Quản lý tài khoản	Quản lý thông tin tài khoản FLAD
17	Use case Tổng hợp tồn kho	Xem tồn kho các mặt hàng của cửa hàng
18	Use case quản lý khách hàng	Quản lý khách hàng đã có đơn hàng ở cửa hàng
19	Use case Quản lý số đen	Quản lý khách hàng bị tố cáo
20	Use case quản lý nhóm khách hàng	Quản lý các nhóm khách hàng để phục vụ cho các chiến dịch quảng bá
21	Use case Quản lý đơn hàng	Quản lý đơn hàng (sửa đơn hàng, hủy, duyệt đơn hàng)
22	Quản lý bài đăng ở facebook	Quản lý và theo dõi các bài đăng của page ở facebook
23	Use case Chat	Nhắn tin facebook ngay trên giao diện FLAD
24	Use case Thống kê doanh thu	Xem thống kê doanh thu trong khoảng thời gian
25	Use case Xem nhật ký truy cập	Xem nhật ký truy cập FLAD

Bảng 3-2: Danh sách use case

3.2.2. Sơ đồ lớp



Hình 3-27: Biểu đồ lớp

Mô tả sơ đồ lớp

STT	Tên lớp	Mô tả
1	CATEGORIES	Lớp loại sản phẩm
2	UNITS	Lớp đơn vị tính
3	PRODUCTS	Lớp sản phẩm
4	VARIANTS	Lớp thuộc tính của sản phẩm
5	CARRIERS	Lớp đơn vị vận chuyển

6	USERS	Lớp người dùng
7	STORES	Lớp cửa hàng
8	SYNTAXES	Lớp cú pháp
9	CUSTOMERS	Lớp khách hàng
10	CUSTOMER_GROUPS	Lớp khách hàng
11	CUSTOMER_BLOCKS	Lớp khách hàng bị block
12	COUPONS	Lớp mã khuyến mãi
13	ORDERS	Lớp đơn hàng
14	ORDER_DETAILS	Lớp chi tiết đơn hàng
15	SHIPPING_INFORMATION	Lớp thông tin giao hàng
16	PRODUCT_GROUPS	Lớp nhóm sản phẩm
17	LIVESTREAMS	Lớp livestreams
18	NOTES	Lớp ghi chú
19	IMPORT_RECEIPTS	Lớp nhập kho
20	EXPORT_RECEIPTS	Lớp xuất kho
21	IMPORT_RECEIPT_DETAILS	Lớp chi tiết nhập kho
22	EXPORT_RECEIPT_DETAILS	Lớp chi tiết xuất kho
23	COMMENT_SAMPLES	Lớp comment mẫu
24	MESSAGE_SAMPLES	Lớp tin nhắn mẫu
25	LOGS	Lớp nhật ký hoạt động
26	SETTINGS	Lớp thiết lập
27	CHATBOTS	Lớp chatbots

Bảng 3-3: Các lớp của sơ đồ lớp

3.2.3. Phân tích và thiết kế CSDL

Dựa trên việc xác định yêu cầu và phân tích yêu cầu, ứng dụng FLAD bao gồm các đối tượng sau đây:

STT	Tên lớp	Thuộc tính
1	CATEGORIES	id, parent_id, name, sku, description, user_id
2	UNITS	id, name, description, user_id, is_default
3	PRODUCTS	id, unit_id, category_id, user_id, name, sku, image, description, stock, buy_price, sell_price, size, weight
4	VARIANTS	id, product_id, name, size, color, sku, image, stock, buy_price, sell_price
5	CARRIERS	id, name, code, phone, email, address, service, type, user_id, is_default, stop_track
6	USERS	id, name, description, user_id, is_default
7	STORES	id, name, city, country, address, phone, email, user_id, is_locked
8	SYNTAXES	id, user_id, details, is_default
9	CUSTOMERS	id, name, code, birthday, phone, email, address, gender, description, facebook_user_id, user_id,

		customer_group_id, is_blocked
10	CUSTOMER_GROUPS	id, name, code, description, user_id
11	CUSTOMER_BLOCKS	id, reason, count, customer_id, user_id
12	COUPONS	id, name, code, reason, type, value, user_id
13	ORDERS	id, customer_id, user_id, coupon_id, code, facebook_page_id, facebook_user_id, facebook_livestream_id, weight, discount, total, subtotal
14	ORDER_DETAILS	id, variant_id, order_id, price, quantity, code, source, facebook_livestream_id, total, subtotal
15	SHIPPING_INFORMATION	id, order_id, carrier_id, cod_price, price, address, receiver_name, receiver_phone, size, weight, delivery_date, receipt_expected_date, receipt_date
16	PRODUCT_GROUPS	id, name, ids, user_id
17	LIVESTREAMS	id, syntax_id, product_group_id, facebook_livestream_id, facebook_page_id, user_id
18	NOTES	id, details, user_id, customer_id

19	IMPORT_RECEIPTS	id, name, instant, code, delivery, description, user_id, time
20	EXPORT_RECEIPTS	id, name, instant, code, delivery, description, user_id, time
21	IMPORT_RECEIPT_DETAILS	id, quantity, variant_id, unit_id, import_receipt_id
22	EXPORT_RECEIPT_DETAILS	id, quantity, variant_id, unit_id, export_receipt_id
23	COMMENT_SAMPLES	id, user_id, title, content
24	MESSAGE_SAMPLES	id, user_id, title, content
25	LOGS	id, user_id, action, user_action, description, reference
26	SETTINGS	id, config, user_id
27	CHATBOTS	id, name, content, facebook_page_id, facebook_user_id, user_id

Bảng 3-4: Các đối tượng và thuộc tính của đối tượng

3.2.3.1. Bảng Categories

Bảng categories lưu trữ thông tin các loại sản phẩm, nhằm giúp người dùng dễ dàng phân loại sản phẩm cần bán.

CATEGORIES			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã danh mục
NAME	VARCHAR(255)	Not null	Tên danh mục
SKU	VARCHAR(255)	Not null	Mã SKU

DESCRIPTION	VARCHAR(255)	Not null	Mô tả
USER_ID	INT	Foreign key	User sở hữu
PARENT_ID	INT	Foreign key	Danh mục cha

Bảng 3-5: Bảng Loại sản phẩm

3.2.3.2. Bảng Products

Bản products lưu các thông tin chi tiết của sản phẩm nhằm giúp người dùng dễ dàng tra cứu, sản phẩm.

PRODUCTS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã sản phẩm
NAME	VARCHAR(255)	Not null	Tên sản phẩm
SKU	VARCHAR(255)	Not null	Mã SKU
IMAGE	VARCHAR(255)	Not null	URL hình ảnh
DESCRIPTION	VARCHAR(255)	Not null	Mô tả
STOCK	INT	Not null	Tồn kho
BUY_PRICE	INT	Not null	Giá mua
SELL_PRICE	INT	Not null	Giá bán
SIZE	VARCHAR(255)	Not null	Kích thước
WEIGHT	INT	Not null	Cân nặng
UNIT_ID	INT	Foreign key	Mã đơn vị
CATEGORIY_ID	INT	Foreign key	Mã danh mục
USER_ID	INT	Foreign key	Mã người dùng

Bảng 3-6: Bảng Sản phẩm

3.2.3.3. Bảng Variants

Bảng variants lưu các loại sản phẩm, có các thông số khác nhau như: giá, màu, size, tồn kho, ...

VARIANTS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã variant
PRODUCT_ID	INT	Foreign key	Mã sản phẩm
NAME	VARCHAR(255)	Not null	Tên
SIZE	VARCHAR(255)	Not null	Kích thước
COLOR	VARCHAR(255)	Not null	Màu
SKU	VARCHAR(255)	Not null	Mã
IMAGE	VARCHAR(255)	Not null	URL ảnh
STOCK	INT	Not null	Tồn kho
SELL_PRICE	INT	Not null	Giá bán
BUY_PRICE	INT	Not null	Giá mua

Bảng 3-7: Bảng Variants

3.2.3.4. Bảng Đơn vị

Bảng đơn vị lưu thông tin các đơn vị của sản phẩm

UNITS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã đơn vị
NAME	INT	Primary key	Tên đơn vị
DESCRIPTION	TEXT	Not null	Mô tả

USER_ID	INT	Foreign key	Mã người dùng
IS_DEFAULT	INT	Foreign key	Mặc định

Bảng 3-8: Bảng Units

3.2.3.5. Bảng Users

Bảng Users lưu thông tin người dùng sử dụng hệ thống

USERS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã người dùng
NAME	INT	Not null	Tên người dùng
PHONE	VARCHAR(255)	Not null	SĐT
EMAIL	VARCHAR(255)	Not null	Email
ADDRESS	VARCHAR(255)	Not null	Địa chỉ
PASSWORD	VARCHAR(255)	Not null	Mật khẩu
ROLE	VARCHAR(255)	Not null	Vai trò

Bảng 3-9: Bảng Users

3.2.3.6. Bảng Orders

Bảng Orders lưu thông tin các đơn hàng của khách hàng

ORDERS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã đơn hàng
CUSTOMER_ID	INT	Foreign key	Mã khách hàng
USER_ID	INT	Foreign key	Mã người dùng

COUPON_ID	INT	Foreign key	Mã khuyến mãi
FACEBOOK_PAGE_ID	VARCHAR (255)	Not null	Facebook Page ID
FACEBOOK_USER_ID	VARCHAR (255)	Not null	FB User ID
FACEBOOK_LIVESTREAM_ID	VARCHAR (255)		Facebook Livestream ID
WEIGHT	INT	Not null	Cân nặng gói hàng
DISCOUNT	INT	Not null	Giảm giá
SUBTOTAL	INT	Not null	Tổng tiền
TOTAL	INT	Not null	Tổng thanh toán

Bảng 3-10: Bảng Orders

3.2.3.7. Bảng Shipping Information

Bảng shipping information lưu thông tin giao hàng của đơn hàng

SHIPPING INFORMATION			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã vận đơn
ORDER_ID	INT	Foreign key	Mã đơn hàng
CARRIER_ID	INT	Foreign key	Mã ĐVVC
COD_PRICE	INT	Not null	Tiền thu hộ
PRICE	INT	Not null	Phí giao hàng

ADDRESS	VARCHAR(255)	Not null	Địa chỉ giao hàng
RECEIVER_NAME	VARCHAR(255)	Not null	Tên người nhận
RECEIVER_PHONE	VARCHAR(255)	Not null	SĐT người nhận
SIZE	VARCHAR(255)	Not null	Kích thước
WEIGHT	INT	Not null	Cân nặng gói hàng
EXPECTED_DATE	DATETIME	Not null	Ngày nhận dự kiến
DELIVERY_DATE	DATETIME	Not null	Ngày giao
RECEIPT_DATE	DATETIME	Not null	Ngày giao

Bảng 3-11: Bảng Shipping Information

3.2.3.8. Bảng Coupons

Bảng Coupons lưu thông tin mã khuyến mãi

Coupons			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã coupons
CODE	VARCHAR(255)	Not null	Mã khuyến mãi
TYPE	VARCHAR(255)	Not null	Loại giảm giá
VALUE	VARCHAR(255)	Not null	Giá trị giảm
USER_ID	INT	Foreign key	Mã người dùng

Bảng 3-12: Bảng Coupons

3.2.3.9. Bảng Stores

Bảng stores lưu thông tin cửa hàng

STORES

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã cửa hàng
NAME	VARCHAR(255)	Not null	Tên cửa hàng
CITY	VARCHAR(255)	Not null	Thành phố
COUNTRY	VARCHAR(255)	Not null	Quốc gia
ADDRESS	VARCHAR(255)	Not null	Địa chỉ
PHONE	VARCHAR(255)	Not null	SĐT
EMAIL	VARCHAR(255)	Not null	Email
USER_ID	INT	Foreign key	Mã người dùng

Bảng 3-13: Bảng Stores

3.2.3.10. Bảng Notes

Bảng notes lưu thông tin các ghi chú của người dùng với khách hàng

NOTES			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã ghi chú
DETAILS	VARCHAR(255)	Not null	Nội dung
USER_ID	INT	Foreign key	Mã người dùng
CUSTOMER_ID	INT	Foreign key	Mã khách hàng

Bảng 3-14: Bảng Notes

3.2.3.11. Bảng Syntaxes

Bảng syntaxes lưu các cú pháp tự động nhận diện comment của khách hàng

SYNTAXES

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã cú pháp
USER_ID	VARCHAR(20)	Foreign key	Mã người dùng
DETAILS	JSONB	Not null	Chi tiết cú pháp

Bảng 3-15: Bảng Syntaxes

3.2.3.12. Bảng Product Groups

Bảng Product Groups lưu thông tin các nhóm hàng hoá khi livestream

PRODUCT GROUPS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã nhóm
NAME	VARCHAR(255)	Not null	Tên nhóm
IDS	JSONB	Not null	Danh sách SP
USER_ID	INT	Foreign key	Mã người dùng

Bảng 3-16: Bảng Product Groups

3.2.3.13. Bảng Import Receipts

Bảng import receipts lưu thông tin các phiếu nhập kho

IMPORT RECEIPTS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã phiếu
NAME	VARCHAR(255)	Not null	Tên hiển thị
INSTANT	JSONB	Not null	Người phụ trách
CODE	VARCHAR(255)	Not null	Mã phiếu

DELIVERY	VARCHAR(255)	Not null	Giao cho
DESCRIPTION	VARCHAR(255)	Not null	Mô tả
USER_ID	INT	Foreign key	Mã nhân viên

Bảng 3-17: Bảng Import receipts

3.2.3.14. Bảng Import receipt details

Bảng import receipt details lưu thông tin chi tiết phiếu nhập kho

IMPORT RECEIPT DETAILS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã chi tiết
QUANTITY	INT	Not null	Số lượng nhập
VARIANT_ID	INT	Foreign key	Mã SP
UNIT_ID	INT	Foreign key	Mã đơn vị
IMPORT_RECEIPT_ID	INT	Foreign key	Mã phiếu nhập

Bảng 3-18: Bảng Import receipt details

3.2.3.15. Bảng Export receipts

Bảng export receipt lưu thông tin phiếu xuất kho

IMPORT RECEIPTS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã phiếu
NAME	VARCHAR(255)	Not null	Tên hiển thị
INSTANT	JSONB	Not null	Người phụ trách

CODE	VARCHAR(255)	Not null	Mã phiếu
DELIVERY	VARCHAR(255)	Not null	Giao cho
DESCRIPTION	VARCHAR(255)	Not null	Mô tả
USER_ID	INT	Foreign key	Mã nhân viên

Bảng 3-19: Export receipt details

3.2.3.16. Bảng Export receipt details

Bảng export receipt details lưu thông tin chi tiết phiếu xuất kho

IMPORT RECEIPT DETAILS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã chi tiết
QUANTITY	INT	Not null	Số lượng xuất
VARIANT_ID	INT	Foreign key	Mã SP
UNIT_ID	INT	Foreign key	Mã đơn vị
EXPORT_RECEIPT_ID	INT	Foreign key	Mã phiếu xuất

Bảng 3-20: Bảng Export receipt details

3.2.3.17. Bảng Customers

Bảng customers lưu thông tin của khách hàng

CUSTOMERS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã KH
NAME	INT	Not null	Tên KH
BIRTHDAY	DATETIME	Not null	Sinh nhật

PHONE	VARCHAR(255)	Not null	SĐT
EMAIL	VARCHAR(255)	Not null	Email
ADDRESS	VARCHAR(255)	Not null	Địa chỉ
GENDER	VARCHAR(255)	Not null	Giới tính
DESCRIPTION	VARCHAR(255)	Not null	Mô tả
FACEBOOK_USER_ID	VARCHAR(255)	Not null	Facebook ID
USER_ID	INT	Foreign key	Mã người dùng
CUSTOMER_GROUP_ID	INT	Foreign key	Mã nhóm KH

Bảng 3-21: Bảng customers

3.2.3.18. Bảng Customer Groups

Bảng customer groups lưu thông tin các nhóm khách hàng

CUSTOMER GROUPS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã tìm kiếm phô biến
NAME	VARCHAR(255)	Not null	Từ khóa tìm kiếm
CODE	VARCHAR(255)	Not null	Lượt tìm kiếm
DESCRIPTION	VARCHAR(255)	Not null	
USER_ID	INT		

Bảng 3-22: Bảng customer groups

3.2.3.19. Bảng Customer Blocks

Bảng Customer Reports lưu thông tin tố cáo khách hàng

CUSTOMER BLOCKS

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã tố cáo
CUSTOMER_ID	INT	Foreign key	Mã KH
USER_ID	INT	Foreign key	Mã người dùng
COUNT	INT	Not null	Số lần
REASON	DESCRIPTION	Not null	Lí do báo cáo

Bảng 3-23: Bảng Customer Blocks

3.2.3.20. Bảng Livestreams

Bảng livestreams lưu thông tin của livestream

LIVESTREAMS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã livestream
SYNTAX_ID	INT	Foreign key	Mã cú pháp
PRODUCT_GROUP_ID	INT	Not null	Mã nhóm SP
FACEBOOK_LIVESTREAM_ID	VARCHAR (255)	Not null	Mã livestream
FACEBOOK_PAGE_ID	VARCHAR (255)	Not null	Mã Page FB
USER_ID	INT	Not null	Mã người dùng

Bảng 3-24: Bảng Livestream

3.2.3.21. Bảng Comment samples

Bảng comment samples lưu các comment mẫu

COMMENT SAMPLES			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã mẫu
USER_ID	INT	Foreign key	Mã người dùng
TITLE	VARCHAR(255)	Not null	Tiêu đề
CONTENT	VARCHAR(255)	Not null	Nội dung

Bảng 3-25: Bảng Comment samples

3.2.3.22. Bảng Message samples

Bảng message samples lưu các tin nhắn mẫu

MESSAGE SAMPLES			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	Mã mẫu
USER_ID	INT	Foreign key	Mã người dùng
TITLE	VARCHAR(255)	Not null	Tiêu đề
CONTENT	VARCHAR(255)	Not null	Nội dung

Bảng 3-26: Bảng message samples

3.2.3.23. Bảng Logs

Bảng Log lưu nhật ký hoạt động của người dùng

LOGS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	ID nhật ký
USER_ID	INT	Foreign key	ID người dùng

ACTION	VARCHAR(255)	Not null	Hành động
USER_ACTION	VARCHAR(255)	Not null	Hành động
DESCRIPTION	VARCHAR(255)	Not null	Mô tả
REFERENCE	VARCHAR(255)	Not null	Mục tiêu

Bảng 3-27: Bảng Chatlist

3.2.3.24. Bảng Settings

Bảng settings lưu thông tin thiết lập của người dùng

SETTINGS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	ID thiết lập
CONFIG	JSONB	Not null	Thiết lập
USER_ID	INT	Foreign key	Mã người dùng

Bảng 3-28: Bảng Settings

3.2.3.25. Bảng Order details

Bảng Order details lưu thông tin chi tiết đơn hàng

ORDER DETAILS			
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	INT	Primary key	ID chi tiết
VARIANT_ID	INT	Foreign key	Mã sản phẩm
ORDER_ID	INT	Foreign key	Mã đơn hàng
PRICE	INT	Not null	Đơn giá
QUANTITY	INT	Not null	Số lượng

CODE	VARCHAR(255)	Not null	
SOURCE	VARCHAR(255)	Not null	Nguồn
FACEBOOK_LIVE STREAM_ID	VARCHAR(255)	Not null	Livestream ID
TOTAL	INT	Not null	Tổng thanh toán
SUBTOTAL	INT	Not null	Tổng tiền

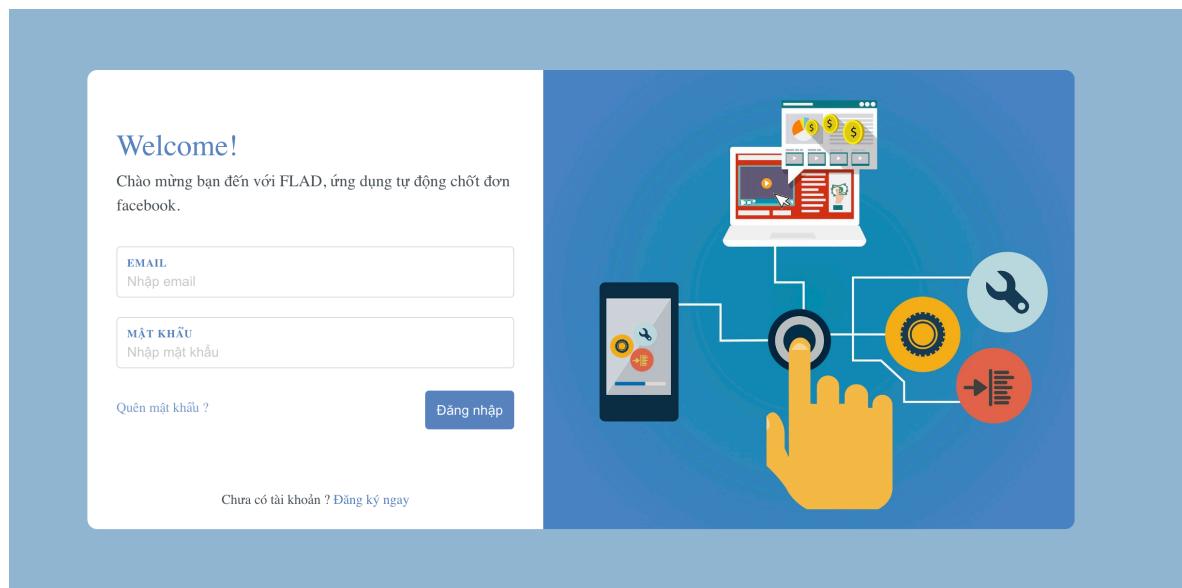
Bảng 3-29: Bảng Order details

3.3 Thiết kế giao diện

Ứng dụng Flad bao gồm 2 phía: người dùng ứng dụng (client) và người quản lý ứng dụng (admin).

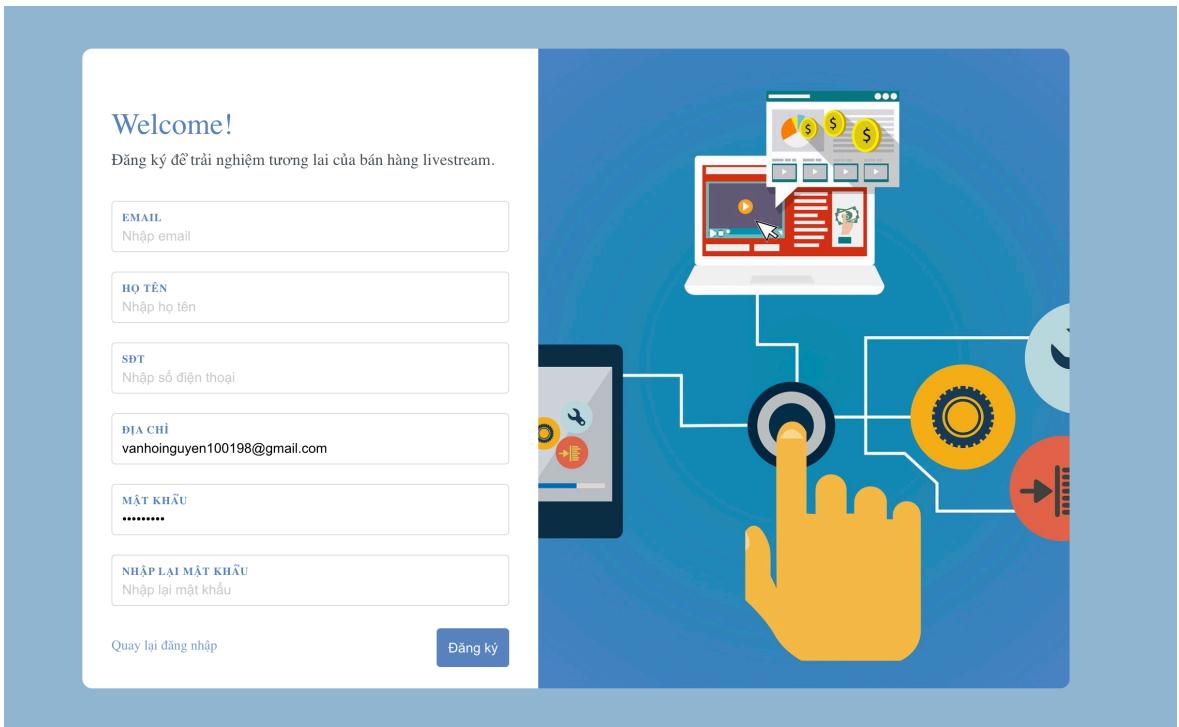
3.1.1. Giao diện Client

- Giao diện đăng nhập



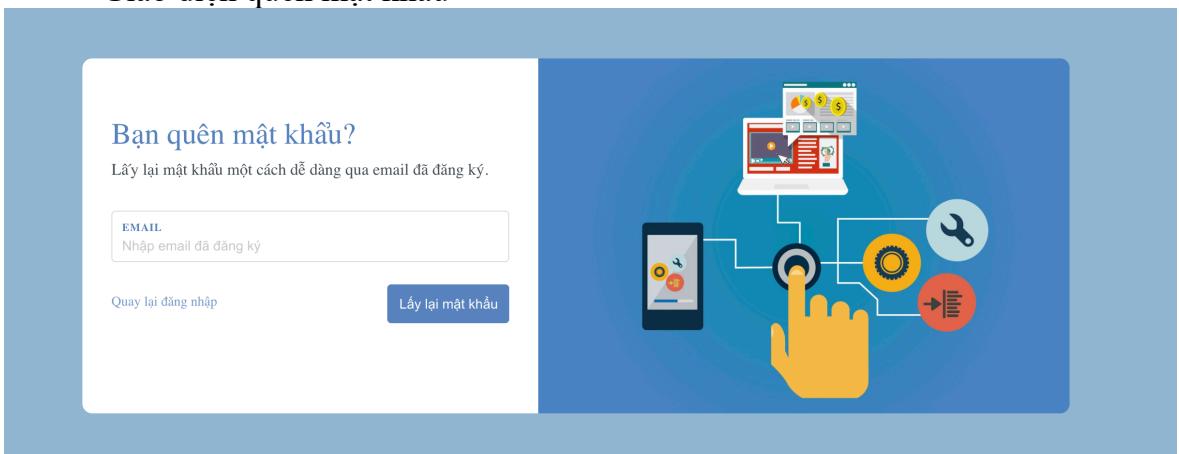
Hình 3-28: Giao diện đăng nhập

- Giao diện đăng ký



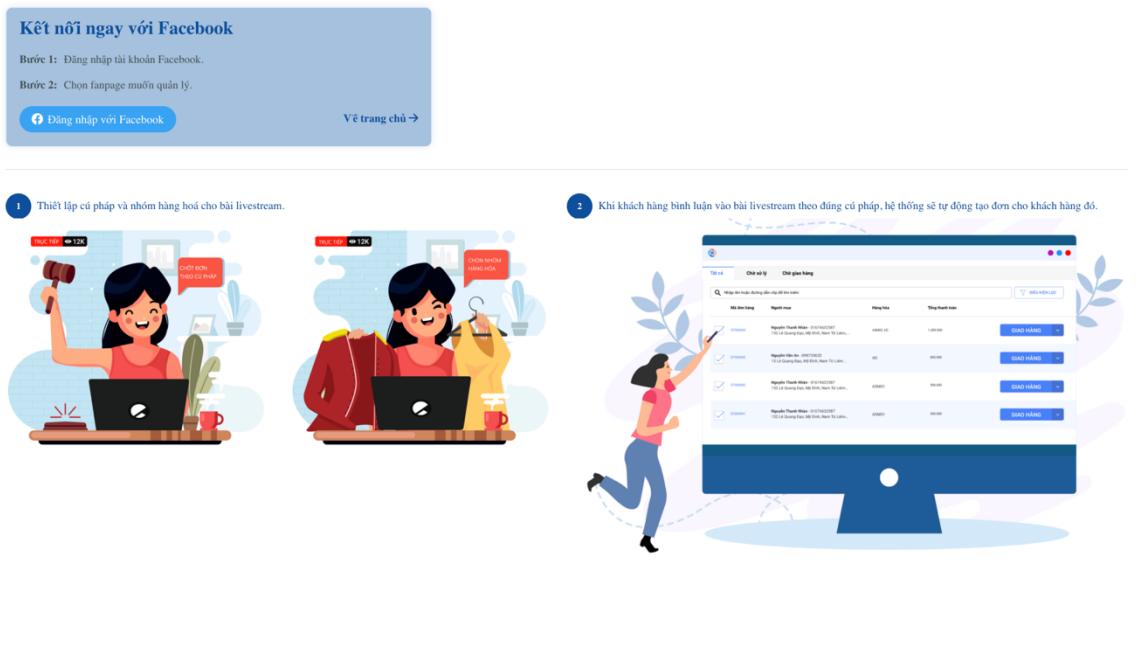
Hình 3-29: Giao diện đăng ký

- Giao diện quên mật khẩu

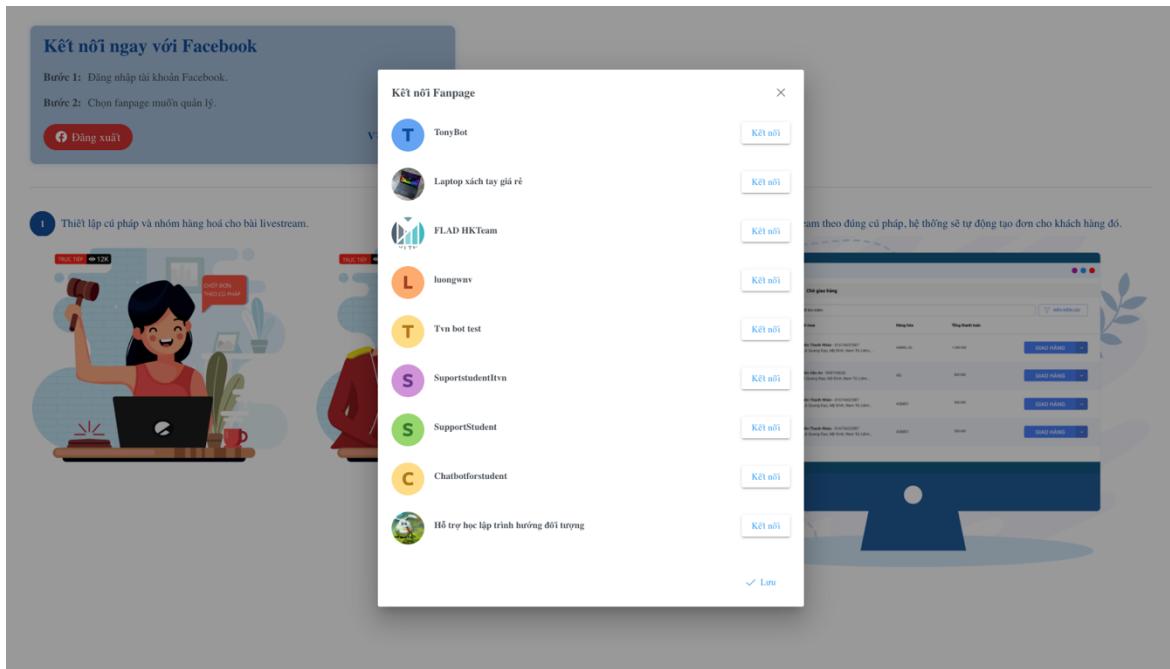


Hình 3-30: Giao diện quên mật khẩu

- Giao diện kết nối với facebook



Hình 3-31: Giao diện kết nối với facebook

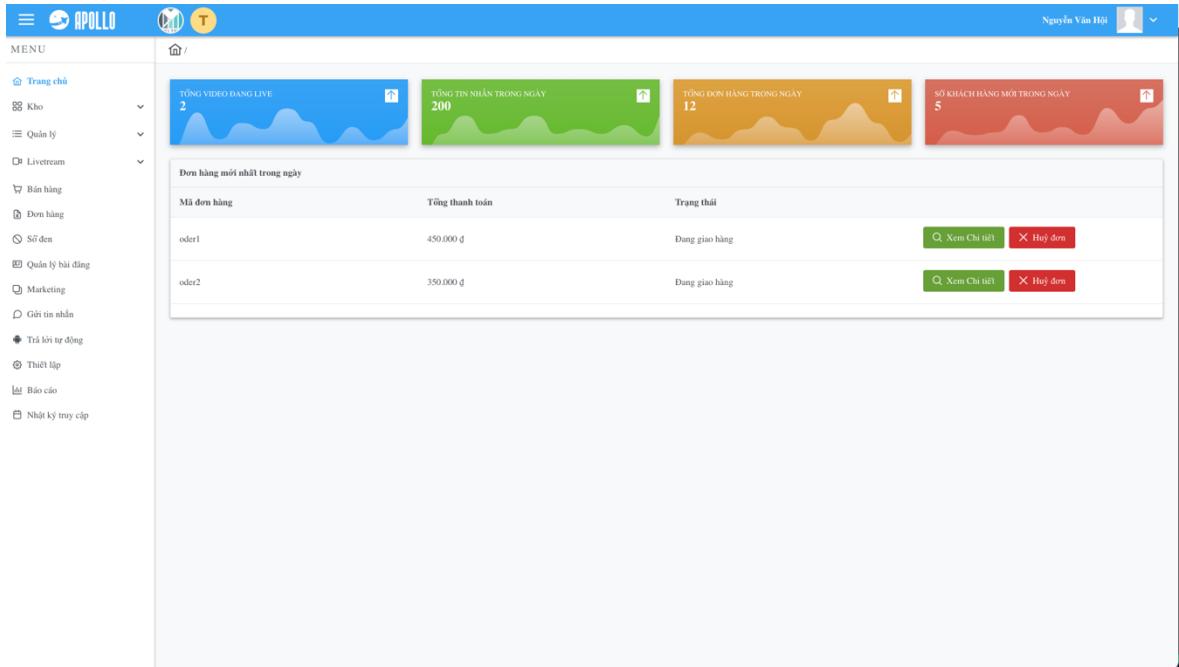


Hình 3-32: Giao diện lựa chọn fanpage kết nối

Trước khi truy cập vào bên trong ứng dụng, ta có thể đăng nhập với tài khoản facebook trước, sau đó sẽ hiển thị danh sách những fanpage mà tài khoản fb của bạn

sở hữu. Việc của bạn là lựa chọn những pages để bán hàng. Có thể bỏ qua bước kết nối trên bằng cách nhấn quay về trang chủ.

- Giao diện trang chủ



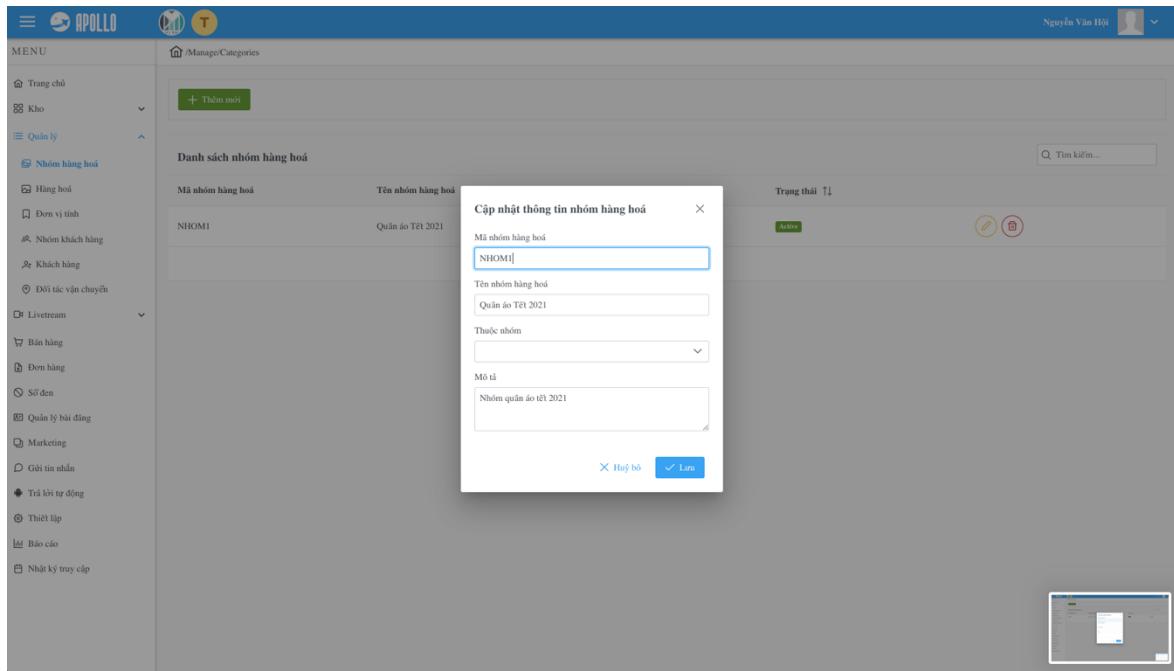
Hình 3-33: Giao diện trang chủ

- Giao diện danh sách nhóm hàng hoá

Mã nhóm hàng hoá	Tên nhóm hàng hoá	Mô tả	Trạng thái
NHOM1	Quần áo Tết 2021	Nhóm quần áo tết 2021	Active

Hình 3-34: Giao diện nhóm hàng hoá

Hình 3-35: Giao diện tạo mới nhóm hàng hoá

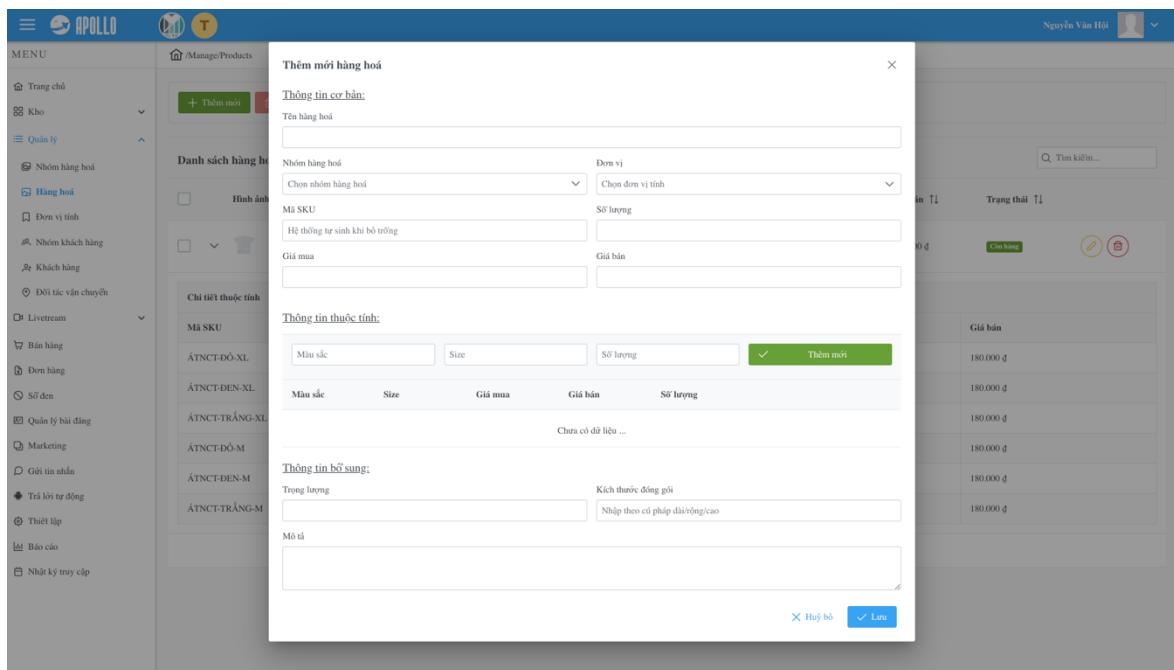


Hình 3-36: Giao diện cập nhật nhóm hàng hoá

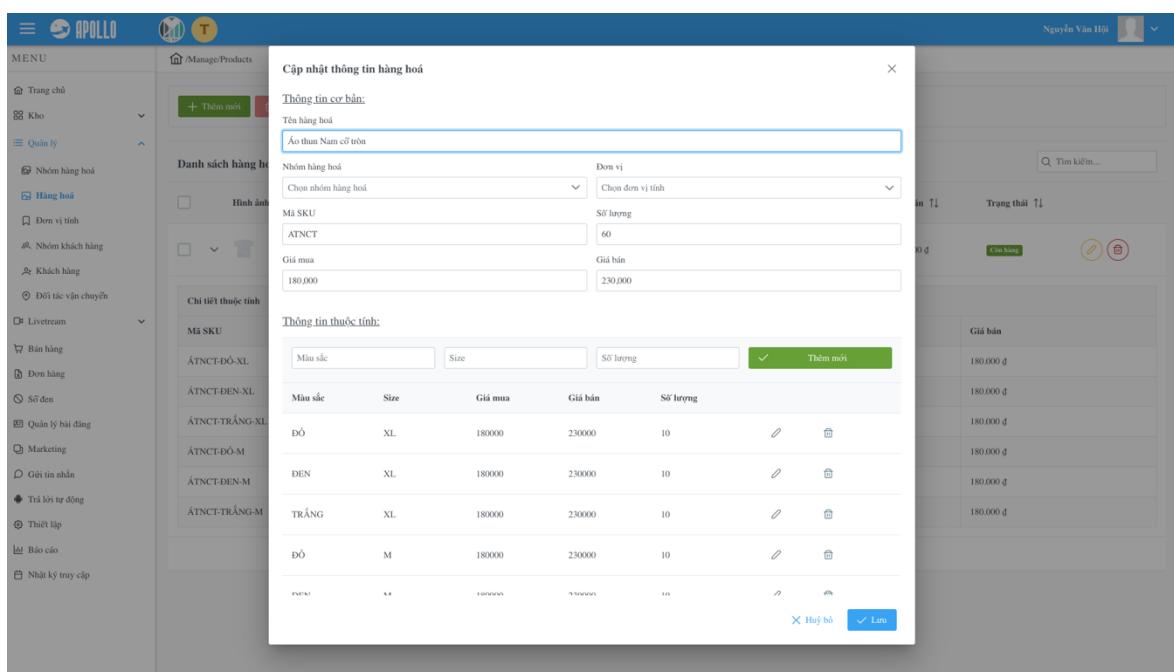
- Giao diện danh sách hàng hoá

Danh sách hàng hoá									
	Hình ảnh	Mã SKU ↑↓	Tên hàng hoá ↑↓	Nhóm hàng hoá ↑↓	Đơn vị tính ↑↓	Số lượng ↑↓	Giá mua ↑↓	Giá bán ↑↓	Trạng thái ↑↓
		ATNCT	Áo thun Nam cổ tròn	Quần áo Tết 2021	Chiếc	60	180.000 ₫	230.000 ₫	Còn hàng
Chi tiết thuộc tính									
Mã SKU	Tên hàng hoá			Số lượng	Giá mua		Giá bán		
ÁTNCT-DÓ-XL	Áo thun Nam cổ tròn (DÓ/XL)			10	230.000 ₫		180.000 ₫		
ÁTNCT-DEN-XL	Áo thun Nam cổ tròn (DEN/XL)			10	230.000 ₫		180.000 ₫		
ÁTNCT-TRẮNG-XL	Áo thun Nam cổ tròn (TRẮNG/XL)			10	230.000 ₫		180.000 ₫		
ÁTNCT-DÓ-M	Áo thun Nam cổ tròn (DÓ/M)			10	230.000 ₫		180.000 ₫		
ÁTNCT-DEN-M	Áo thun Nam cổ tròn (DEN/M)			10	230.000 ₫		180.000 ₫		
ÁTNCT-TRẮNG-M	Áo thun Nam cổ tròn (TRẮNG/M)			10	230.000 ₫		180.000 ₫		

Hình 3-37: Giao diện hàng hoá



Hình 3-38: Giao diện tạo mới hàng hoá



Hình 3-39: Giao diện cập nhật hàng hoá

- Giao diện danh sách đơn vị tính

Hình 3-40: Giao diện đơn vị tính

- Giao diện danh sách đối tác giao hàng

Hình 3-41: Giao diện đối tác giao hàng

- Giao diện danh sách nhóm khách hàng

Mã nhóm khách hàng ↑↓	Tên nhóm khách hàng ↑↓	Mô tả ↑↓	Trạng thái ↑↓
NHOMKH1	Khách hàng quen	Khách hàng quen, mua nhiều hàng ở shop.	Active
NHOMKH2	Khách Vãng Lai	Khách hàng lão lão mua hàng ở shop.	Active

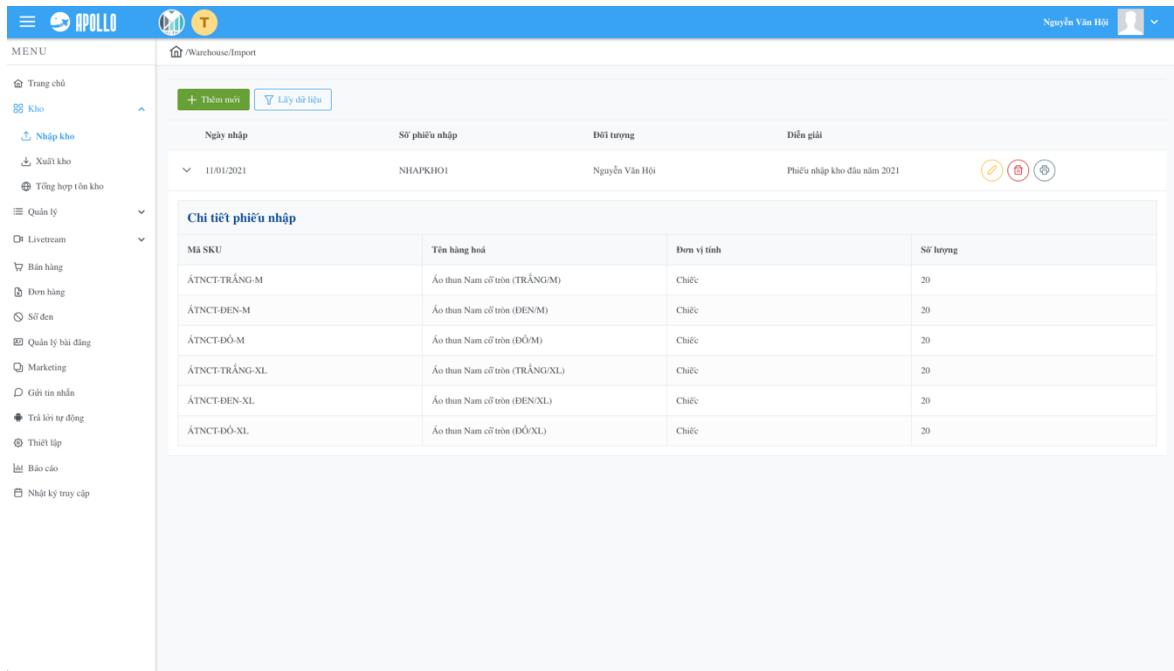
Hình 3-42: Giao diện nhóm khách hàng

- Giao diện danh sách khách hàng

Mã khách hàng ↑↓	Tên khách hàng ↑↓	Nhóm khách hàng ↑↓	Điện thoại ↑↓	Email ↑↓	Địa chỉ ↑↓	Ngày sinh ↑↓
KHI	Trần Ngọc Hưng	Khách hàng quen	0379926712	hangtran98@gmail.com	KTX Khu B, Di An, Bình Dương.	09/10/1998

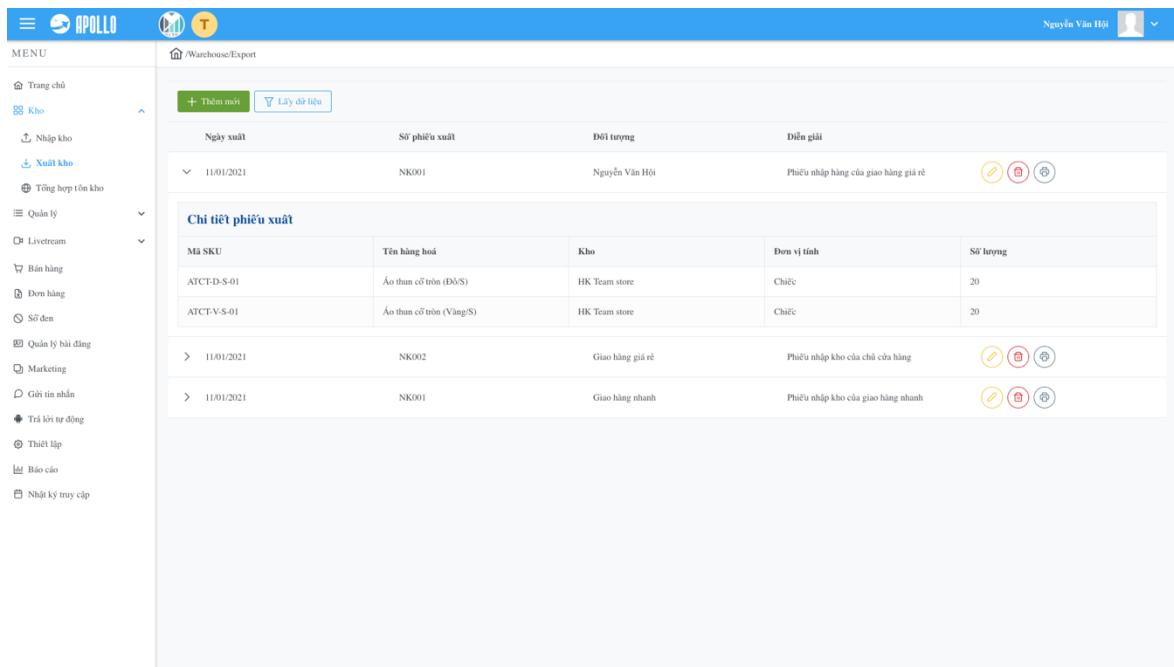
Hình 3-43: Giao diện khách hàng

- Giao diện danh sách Phiếu nhập kho



Hình 3-44: Giao diện nhập kho

- Giao diện danh sách phiếu xuất kho



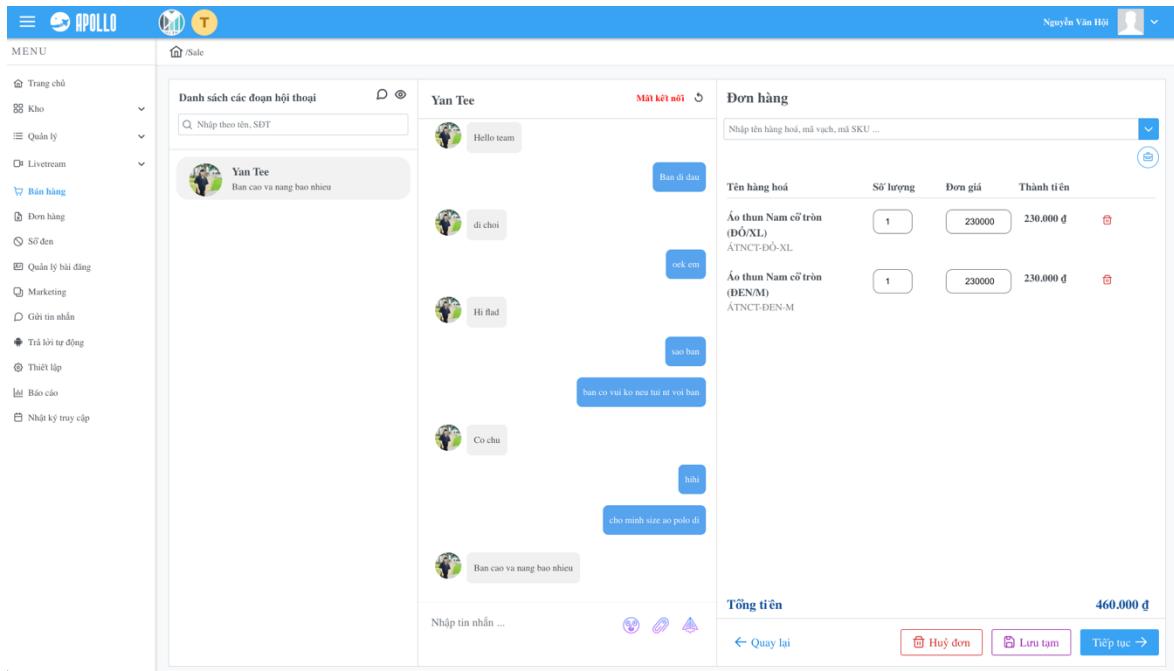
Hình 3-45: Giao diện xuất kho

- Giao diện danh sách tồn kho

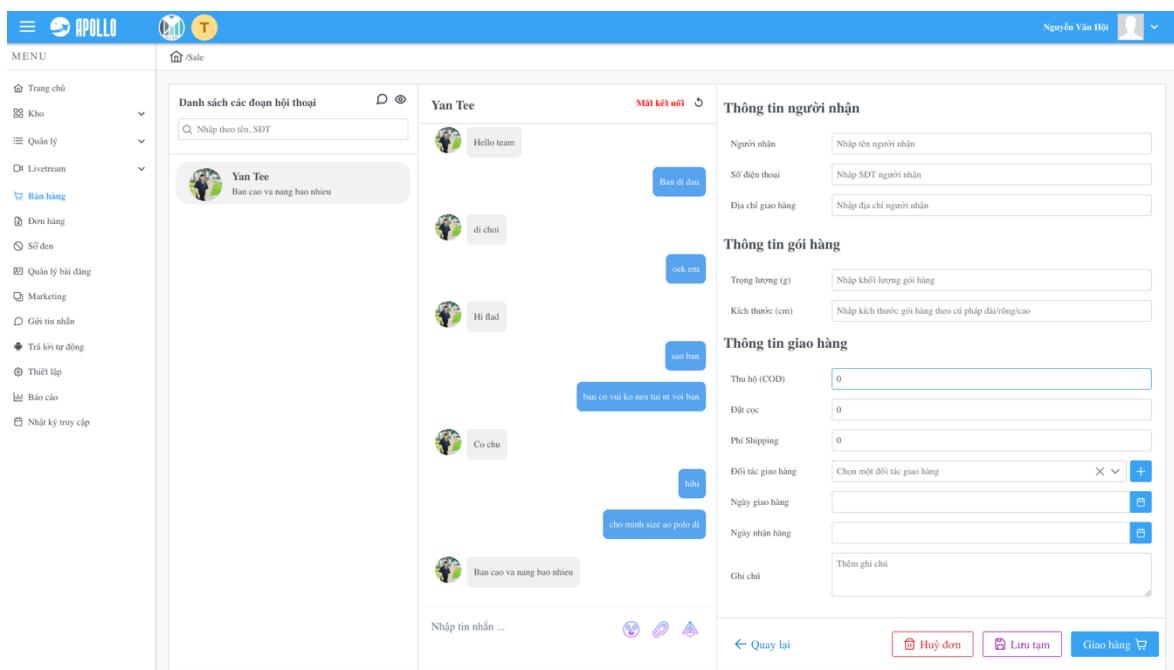
Hình 3-46: Giao diện tồn kho

- Giao diện bán hàng

Hình 3-47: Giao diện bán hàng



Hình 3-48: Giao diện lựa chọn sản phẩm bán



Hình 3-49: Giao diện điền thông tin bán hàng

Tại đây người dùng có thể trực tiếp nhắn tin với những khách hàng mua hàng thông qua hình thức nhắn tin. Người bán hàng có thể sửa đổi thông tin khách hàng đã

nhắn tin với mình. Có thể chọn lựa sản phẩm và tạo đơn hàng trực tiếp tại đây. Ngoài ra còn có thể tạo ra những ghi chú cho đơn hàng.

- Giao diện danh sách đơn hàng

Mã đơn hàng	Mã hóa đơn	Mã vận đơn	Ngày tạo đơn	Ngày vận chuyển	Ngày nhận hàng dự kiến	Người tạo đơn	Người mua hàng	Tổng thanh toán	Trạng thái
Order1	Bill1	Shipping1	12/01/2021	12/01/2021	12/01/2021	Nguyễn Văn Hội	Lê Mai Văn Khanh	320.000đ	Chờ đóng

Hình 3-50: Giao diện đơn hàng

Hiển thị danh sách đơn hàng với nhiều trạng thái khác nhau, người bán hàng có thể cập nhật từng trạng thái của đơn hàng ở trang này.

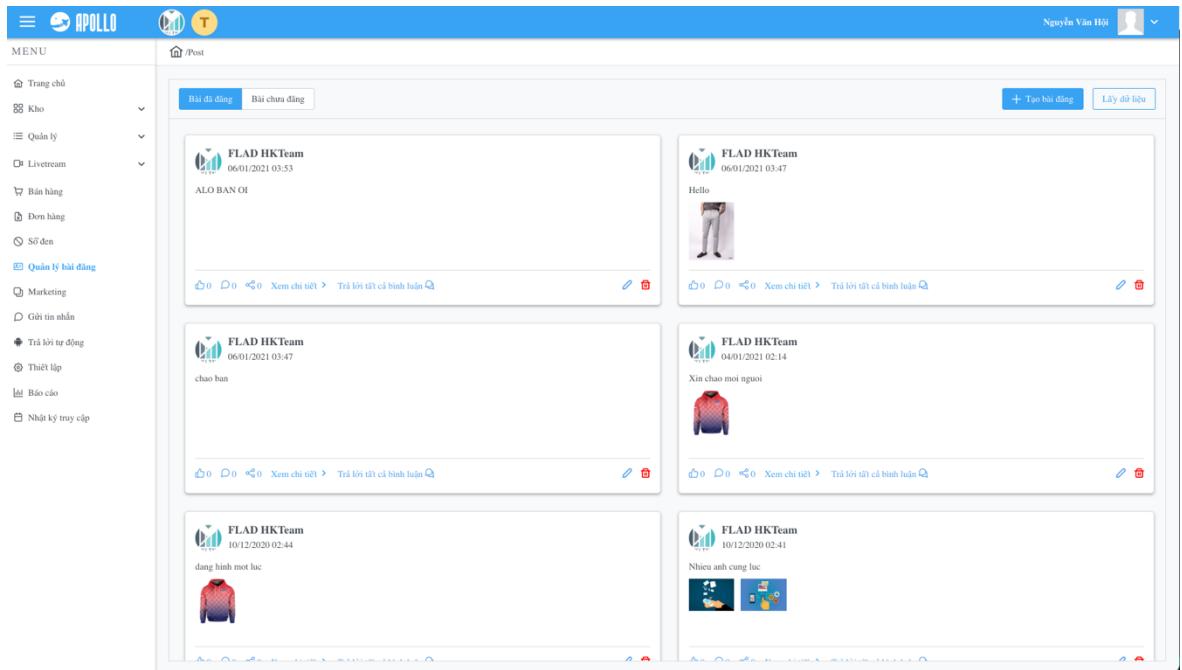
- Giao diện danh sách khách hàng bị chặn

The screenshot shows the APOLLO software interface. At the top, there is a blue header bar with the APOLLO logo and a user profile for 'Nguyễn Văn Hồi'. On the left, a vertical menu bar titled 'MENU' lists various features: Trang chủ, Kho, Quản lý, Livestream, Bán hàng, Đơn hàng, Sđt đen, Quản lý bài đăng, Marketing, Gửi tin nhắn, Trả lời tự động, Thiết lập, Báo cáo, and Nhật ký truy cập. The 'Sđt đen' option is currently selected. The main content area is titled 'Danh sách khách hàng bị báo cáo' (List of blocked customers). It includes a dropdown menu 'Chọn Fanpage' (Select Fanpage) and a table with columns: Tên Facebook, Họ tên, Số DT, Email, Số lần bị báo cáo, Lý do báo cáo, Người báo cáo, and Ngày cập nhật. A message at the bottom of the table says 'Chưa có dữ liệu ...' (No data available ...).

Hình 3-51: Giao diện khách hàng bị chặc

Hiển thị danh sách những khách hàng bị chặn, khách hàng bị chặn khi họ có những bình luận tiêu cực trên video livestream hoặc khi nhắn tin với page của người bán hàng. Người bán hàng có thể trực tiếp block khách hàng ngay lúc bán hàng hoặc khi đang livestream.

- Giao diện quản lý bài đăng trên facebook



Hình 3-52: Giao diện quản lý bài đăng trên facebook

Hiển thị danh sách những bài đăng trên facebook của page được kết nối. Tại đây người bán hàng có thể tạo, chỉnh sửa, hoặc xoá bài đăng của page mình trực tiếp mà không cần phải trực tiếp vào fanpage cá nhân của mình để thực hiện. Ngoài ra ở đây còn có chức năng giúp người bán hàng có thể trả lời tất cả những bình luận của khách hàng bình luận vào page của mình với một thao tác, không cần phải trả lời từng bình luận.

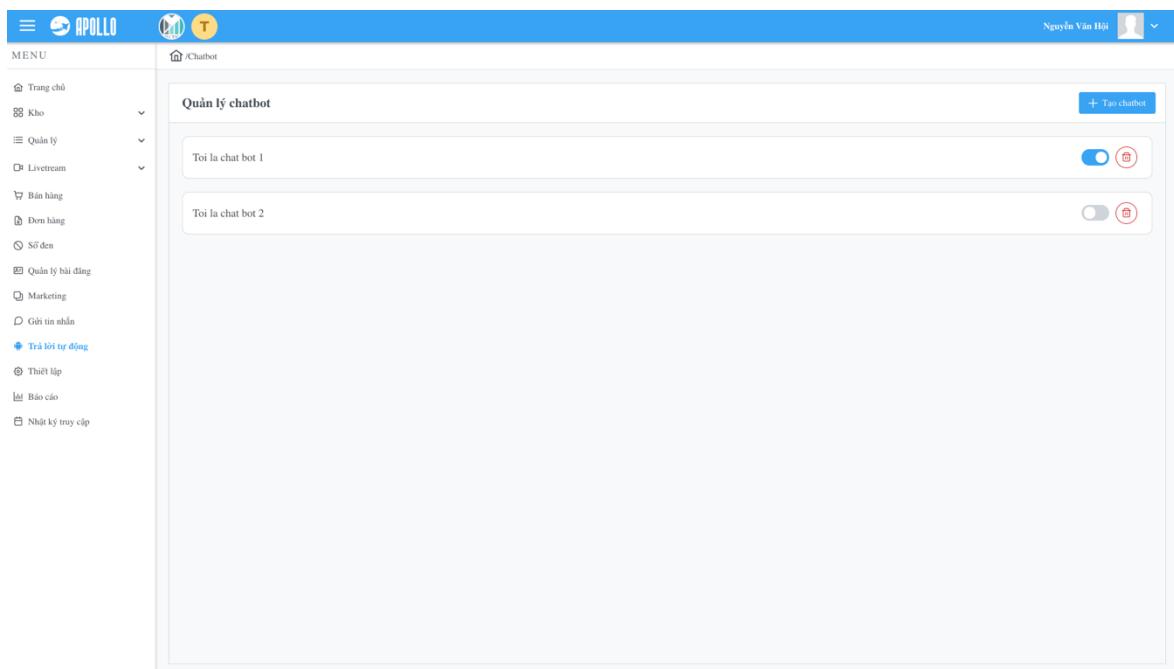
- Giao diện marketing sản phẩm

The screenshot displays the APOLLO software interface. At the top, there's a blue header bar with the APOLLO logo and a user profile for 'Nguyễn Văn Hết'. Below the header is a sidebar titled 'MENU' containing various management options: Trang chủ, Kho, Quản lý, Livetream, Bán hàng, Đơn hàng, Số liệu, Quản lý bài đăng, Marketing, Giới thiệu, Trả lời tự động, Thiết lập, Báo cáo, and Nhật ký truy cập. The 'Marketing' option is selected. The main content area is titled 'Marketing' and contains three buttons: 'Đồng hàng hóa' (in green), 'Gỡ hàng hóa' (in red), and 'Chia sẻ lên tường' (in blue). Below these buttons is a search bar with the placeholder 'Tim kiếm...'. A table titled 'Danh sách hàng hóa' lists products with columns for Hình ảnh, Tên hàng hóa, Mã SKU, Tồn kho, Giá bán, Nhóm hàng hóa, and Trạng thái. One item is listed: 'Áo thun Nam cổ tròn' (ATNCT) with a price of 230.000đ, belonging to 'Quần áo Tết 2021' and marked as 'Active'. Navigation arrows at the bottom of the table indicate more items.

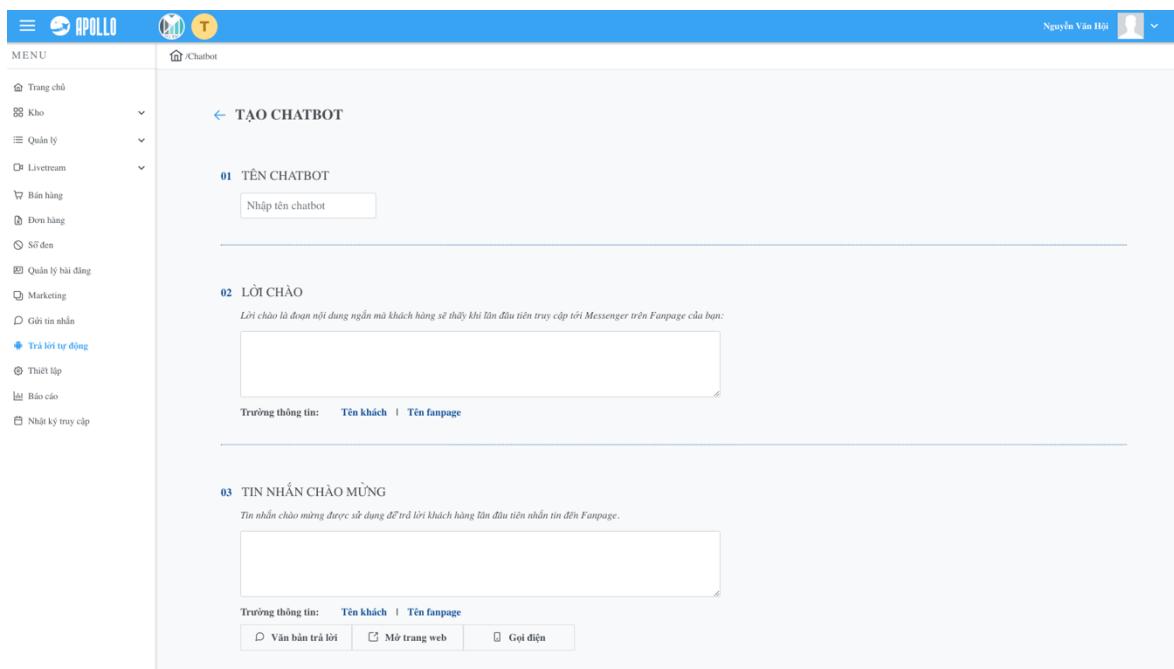
Hình 3-53: Giao diện marketing sản phẩm

Hiển thị danh sách sản phẩm được đồng bộ lên catalog của facebook, cho phép đồng bộ, gỡ hàng hoá trên catalog của facebook. Ngoài ra còn có thể đăng hết các hàng hoá trong kho lên facebook phục vụ mục đích quản cáo sản phẩm.

- Giao diện thiết lập chatbot



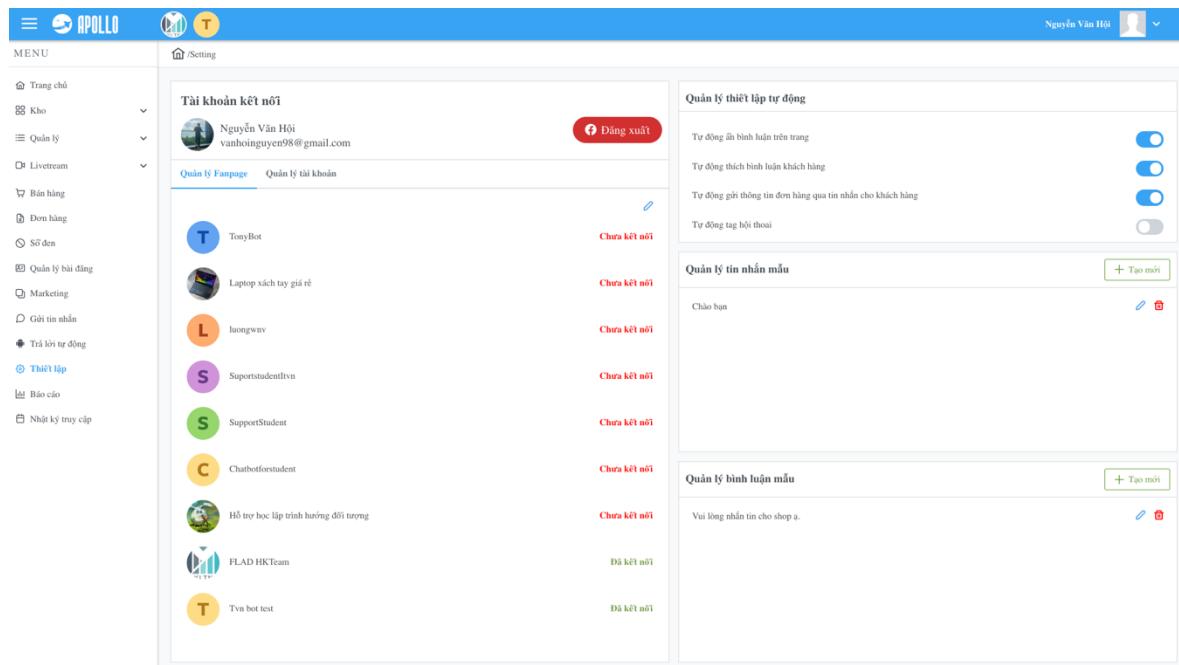
Hình 3-54: Giao diện danh sách chatbot



Hình 3-55: Giao diện tạo mới chatbot

Cho phép tạo ra kịch bản cho chatbot của page kết nối, với mục đích trả lời tự động khi khách hàng nhắn tin đến page. Người dùng được phép tạo ra nhiều chatbot với nhiều kịch bản khác nhau, giúp tự động hóa hơn việc bán hàng.

- Giao diện thiết lập



Hình 3-56: Giao diện thiết lập

Cho phép người dùng đăng nhập, đăng xuất tài khoản facebook đã kết nối. Lựa chọn kết nối với những page khác của tài khoản. Ngoài ra còn được phép thiết lập những tính năng cần thiết cho hệ thống. Cũng có thể tạo ra những tin nhắn, những bình luận mẫu phục vụ cho việc trả lời, bình luận tự động tại đây.

- Giao diện nhật ký truy cập

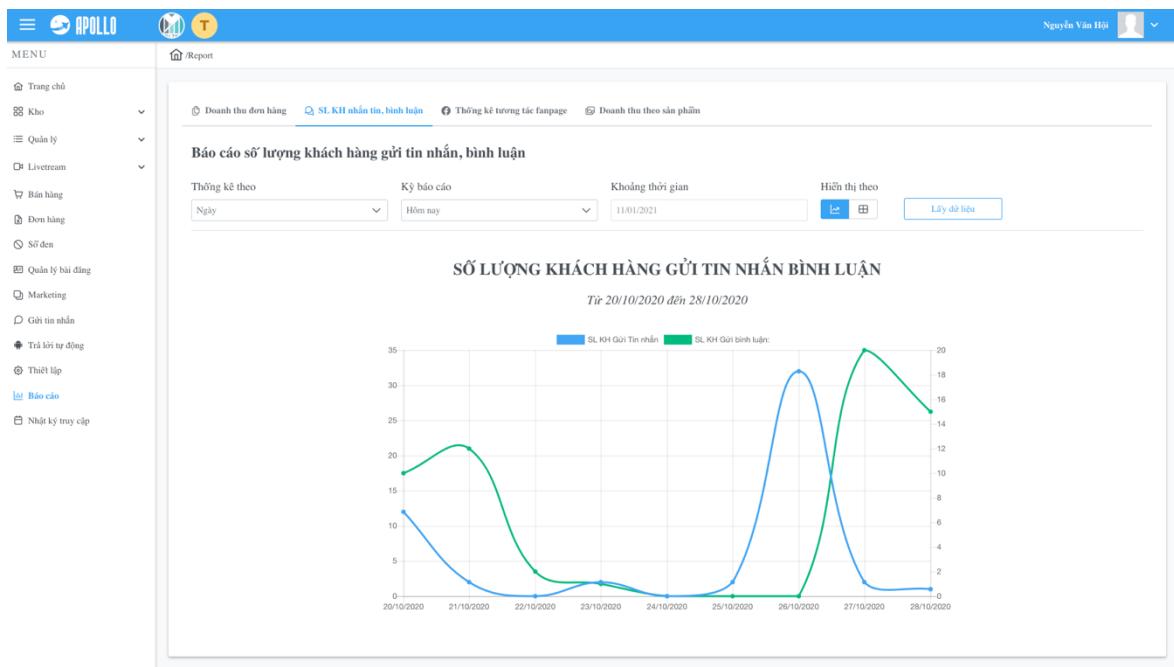
Thời gian	Người thực hiện	Hành động	Mô tả	Tham chiếu
11/01/2021 11:52	Nguyễn Văn Hội	Kết nối	Kết nối với page luongwv	luongwv
11/01/2021 11:52	Nguyễn Văn Hội	Huỷ kết nối	Huỷ kết nối với page Tvn bot test	Tvn bot test

Hình 3-57: Giao diện nhật ký truy cập

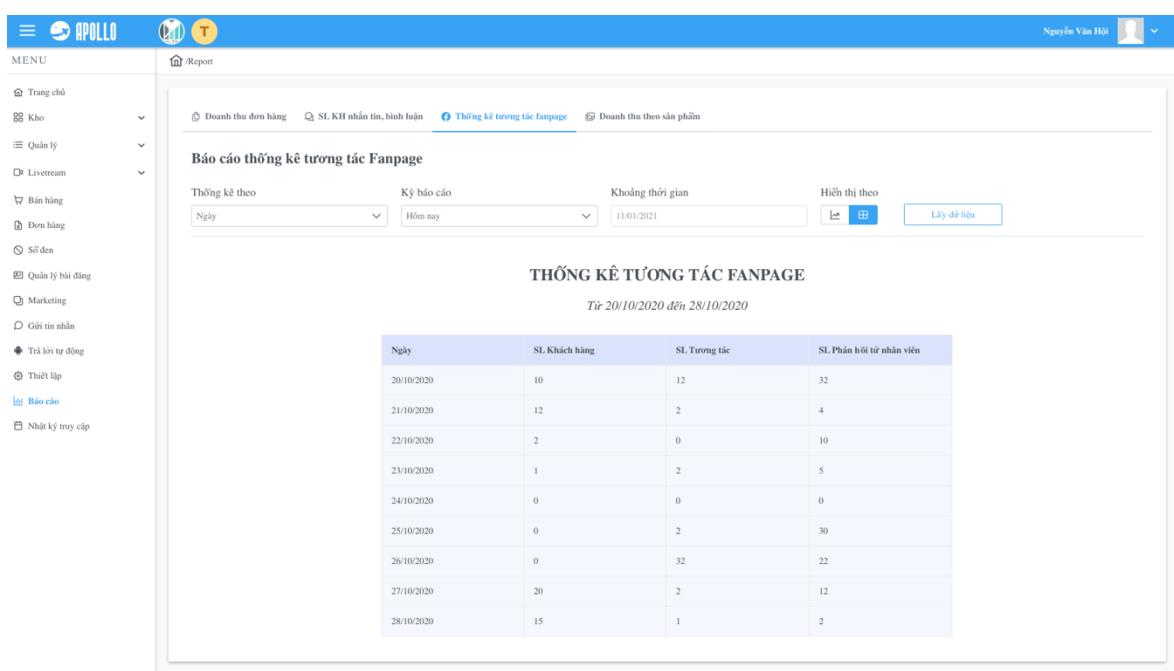
- Giao diện báo cáo

Từ 20/10/2020 đến 21/10/2020	Tổng đơn	Doanh thu
20/10/2020	15	15
21/10/2020	15	55

Hình 3-58: Giao diện báo cáo theo đơn hàng



Hình 3-59: Giao diện báo cáo theo tin nhắn

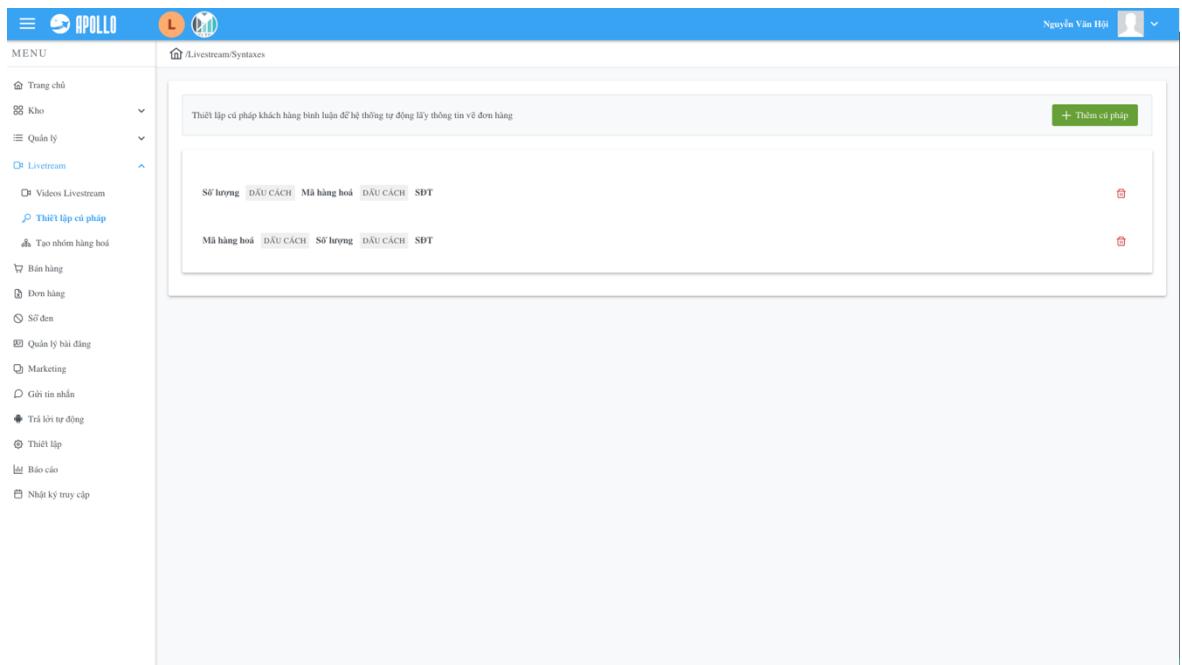


Hình 3-60: Giao diện báo cáo theo fanpage



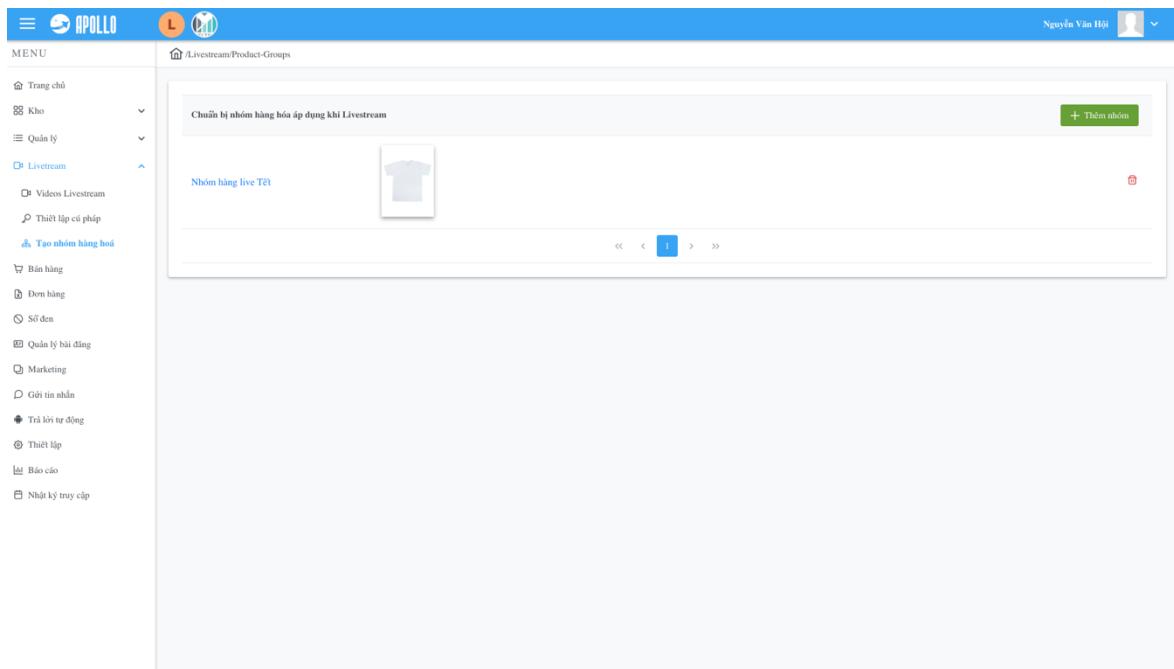
Hình 3-61: Giao diện báo cáo theo sản phẩm

- Giao diện thiết lập cú pháp



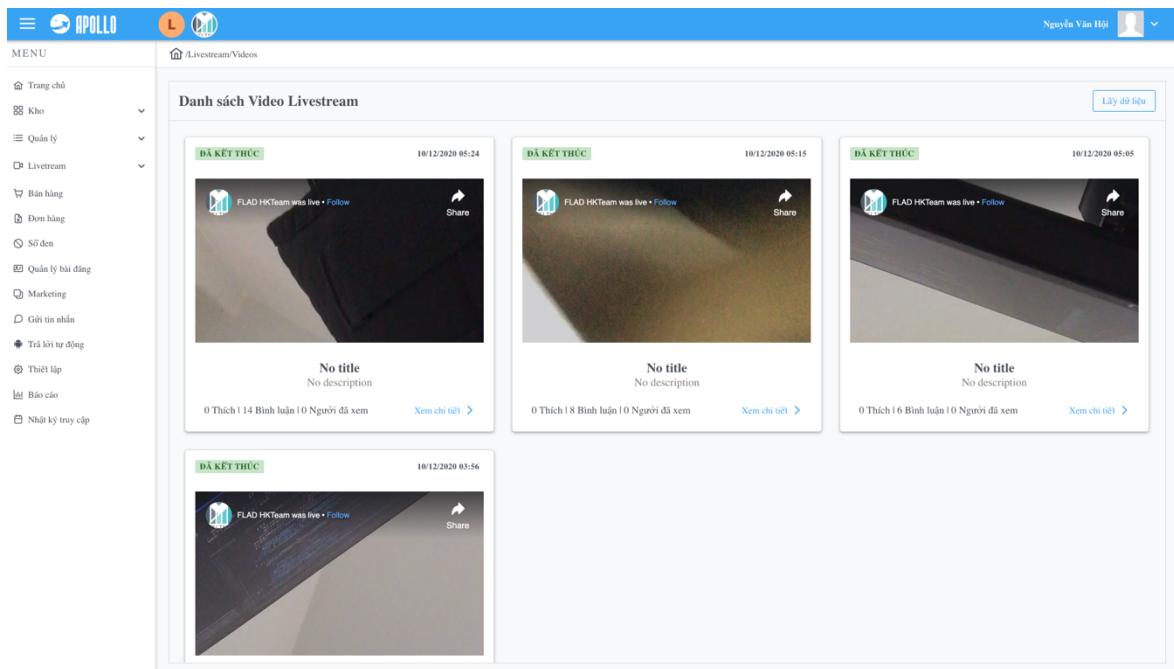
Hình 3-62: Giao diện thiết lập cú pháp

- Giao diện thiết lập nhóm sản phẩm



Hình 3-63: Giao diện thiết lập nhóm sản phẩm

- Giao diện danh sách video livestream



Hình 3-64: Giao diện danh sách video livestream

Hiển thị danh sách những video livestream của page được kết nối. Tại đây, sau khi người bán hàng tạo ra video livestream thông qua điện thoại hoặc những thiết bị khác thì sau đó người bán cần phải thiết lập danh sách cú pháp, danh sách những sản phẩm bán trong buổi livestream trước khi bước vào việc chốt đơn tự động.

- Giao diện chi tiết livestream video

The screenshot shows a Facebook live stream interface. On the left, there is a sidebar titled 'Bình luận từ live' (Comments from live) which lists several comments from users like FLAD HKTeam, Yan Tee, and Nguyễn Văn Hội. On the right, there is a main panel titled 'Đơn hàng' (Order) with a sub-section 'DANH SÁCH ĐƠN' (List of orders). It displays a single order entry for 'DANH SÁCH HÀNG HÓA' (List of goods) with a placeholder 'Nhập tên khách hàng, mã' (Enter customer name, code). Below this are tabs for 'Người mua' (Buyer), 'Hàng hoá' (Goods), and 'Thông tin GH' (Information GH). At the bottom right, there are buttons for 'Làm mới dữ liệu' (Refresh data), '+ Tạo đơn' (Create order), and 'Hoạt động gần đây' (Recent activity). The status bar at the top indicates 0 Người đã xem / 14 Bình luận / 0 Lượt thích / 0 Khách hàng / 0 Đơn hàng.

Hình 3-65: Giao diện chi tiết livestream video

Hiển thị chi tiết của video livestream trên facebook. Khi khách hàng xem livestream và bình luận mua hàng thì tại đây ứng dụng sẽ thực hiện việc bắt những bình luận mua hàng đó và kiểm tra tính hợp lệ của ứng dụng dựa trên cú pháp và danh sách nhóm sản phẩm đã thiết lập trước đó. Nếu bình luận hợp lệ, hệ thống sẽ tự tạo và hiển thị ra đơn hàng cho người bán hàng theo dõi. Còn nếu bình luận không hợp lệ, hệ thống cũng sẽ thông báo cho khách hàng những trường thông tin không hợp lệ, qua đó giúp người bán hàng linh động hơn trong việc kêu gọi khách hàng của mình bình luận đúng cú pháp.

- Giao diện chi tiết đơn hàng

Hình 3-66: Giao diện chi tiết đơn hàng

3.1.2. Giao diện Admin

- Giao diện trang chủ

Hình 3-67: Giao diện trang chủ

- Giao diện quản lý tài khoản người dùng

Hình 3-68: Giao diện quản lý tài khoản người dùng

- Giao diện báo cáo số lượng người dùng

Ngày	Số lượng
10/01/2021	10
11/01/2021	12

Hình 3-69: Giao diện báo cáo số lượng người dùng

Chương 4. ÁP DỤNG TRIỂN KHAI ỨNG DỤNG VỚI KUBERNETES TRÊN DIGITALOCEAN

4.1. Giới thiệu DigitalOcean

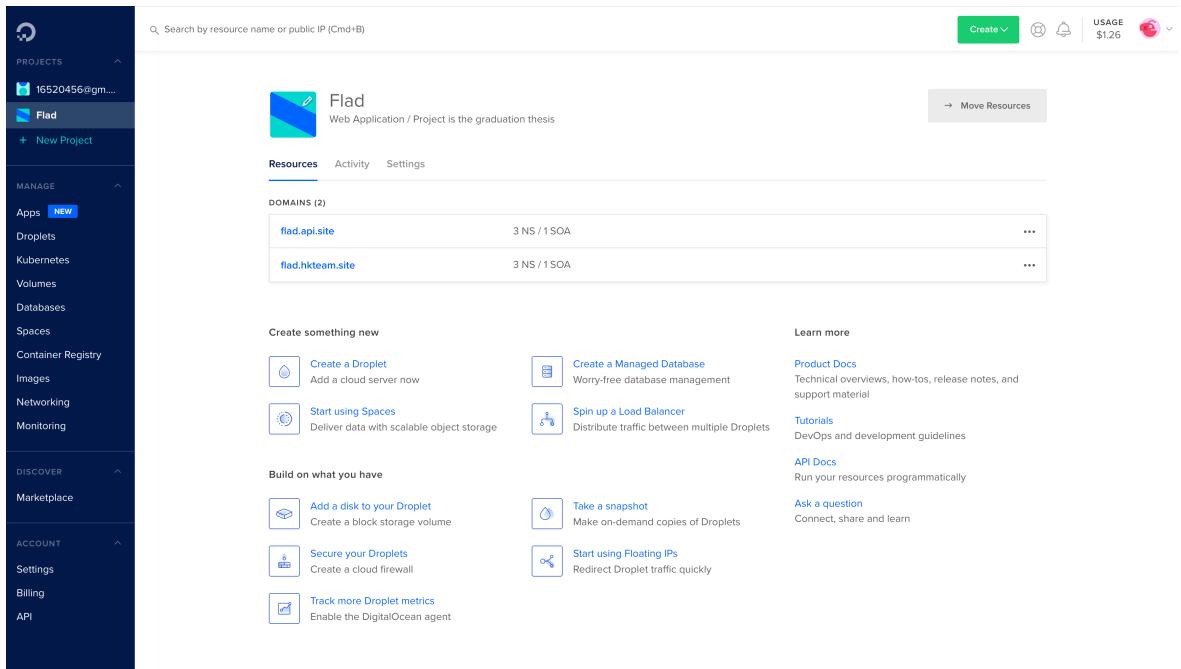
DigitalOcean là nhà cung cấp dịch vụ cloud server, các giải pháp về cơ sở hạ tầng, cho các nhà phát triển. Với DigitalOcean, bạn dễ dàng triển khai một máy chủ web nhanh chóng theo cam kết chỉ trong vòng 55 giây.

Tính năng nổi bật của DigitalOcean:

- Ổ cứng SSD: giúp nâng cao hiệu năng hoạt động của server.
- Control plane: đơn giản, dễ dàng thao tác, quản lý
- Cộng đồng hỗ trợ tích cực: có rất nhiều bài hướng dẫn cài đặt, sử dụng và quản trị server
- One-click Install Apps: thay vì phải cấu hình các ứng dụng phức tạp, chỉ với vài click, server của bạn sẽ được triển khai tự động.
- Với rất nhiều datacenter: trải dài trên khắp thế giới bạn sẽ dễ dàng lựa chọn location phù hợp.

4.2. Đăng ký tài khoản và tạo mới một kubernetes cluster

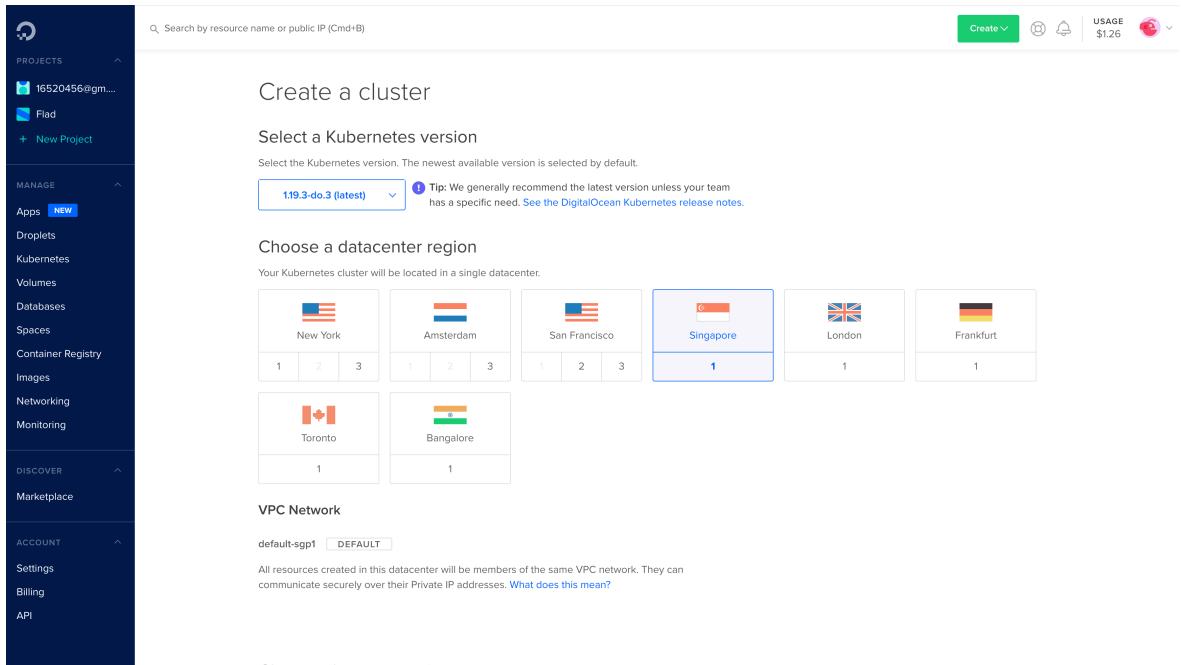
Để đăng ký được tài khoản DigitalOcean thì trước tiên ta cần phải có thẻ thanh toán quốc tế của bất kì ngân hàng nào cũng được. Sẽ không mất phí khi đăng ký, DigitalOcean chỉ trừ tiền khi bạn mua hoặc sử dụng dịch vụ của họ khi đã đăng ký xong tài khoản.



Hình 4-1: Giao diện trang chủ DigitalOcean

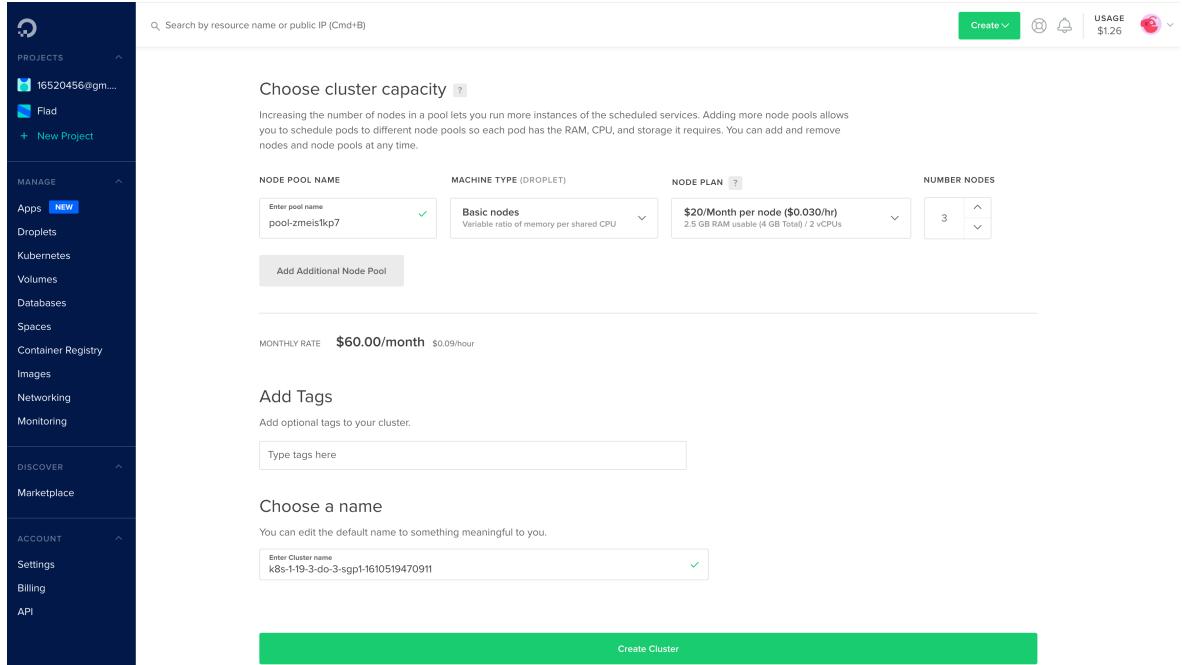
Sau khi đăng ký xong tài khoản, thì ta bắt đầu tạo một kubernetes cluster.

- Chọn khu vực đặt kubernetes



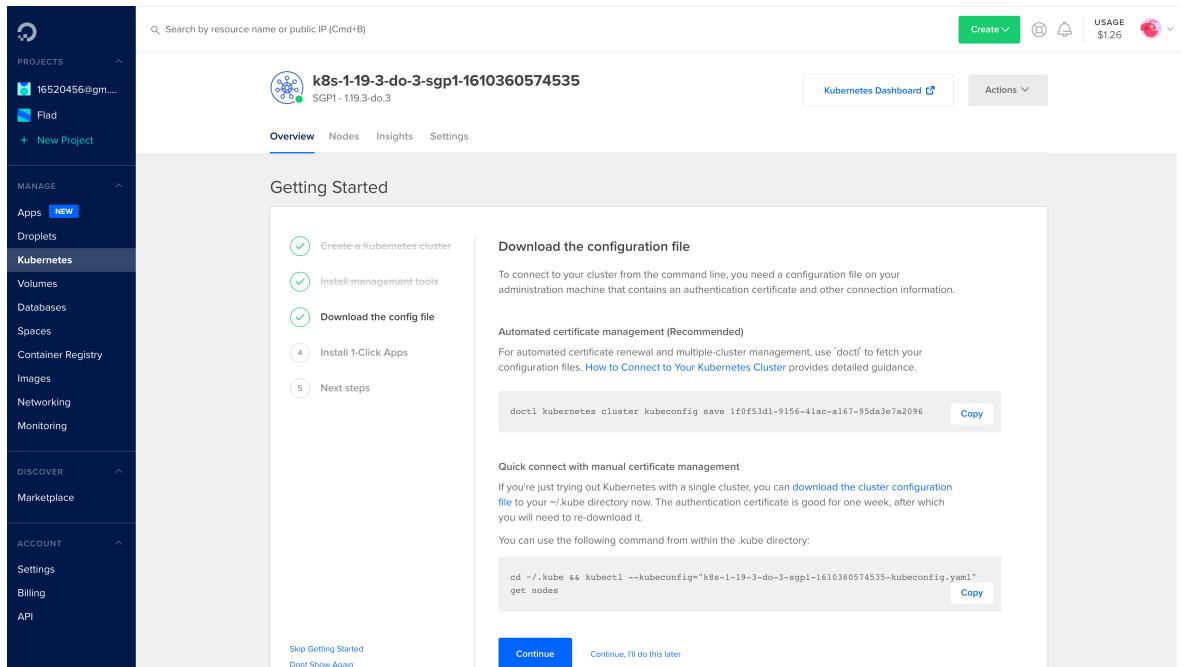
Hình 4-2: Tạo mới Kubernetes – Chọn khu vực

- Chọn dung lượng cho cụm cluster, có nhiều chọn lựa với các mức giá khác nhau. Hiện tại bạn có thể chọn dùng thử option 10\$/ 1 tháng/ 1 Node, 1GB Ram và 1 vCPU. Đây là gói dành cho development, thích hợp để test.



Hình 4-3: Tạo mới Kubernetes – Chọn dung lượng cụm

- Sau khi đã thiết lập hết tất cả các thông tin trên, nhấn Create Cluster, ta sẽ có 1 cluster kubernetes với 1 node.



Hình 4-4: Giao diện một kubernetes cluster

- Sau khi tạo xong 1 cluster, bạn có thể dựa vào hướng dẫn trên để cấu hình và triển khai ứng dụng của mình.
- Nhấn vào Kubernetes Dashboard để xem thông tin chi tiết những thành phần trong kubernetes như: pods, service, deployment,

4.3. Triển khai ứng dụng lên cụm cluster

Trước khi triển khai ứng dụng lên cụm cluster vừa tạo, ta cần phải cài một số tools cần thiết như:

- **kubectl**: công cụ dòng lệnh của kubernetes, giúp tương tác với cluster.
- **doctl**: là công cụ dòng lệnh của DigitalOcean, giúp tương tác với API của Digital Ocean

Để kết nối cụm cluster của bạn từ dòng lệnh, bạn cần tệp cấu hình trên máy của mình có chứa chứng chỉ xác thực và thông tin kết nối khác.

Xác thực với DigitalOcean

```
doctl auth init
```

Lưu kubeconfig và kết nối với cluster vừa tạo của DigitalOcean

```
doctl kubernetes cluster kubeconfig save 1f0f53d1-9156-41ac-a167-95da3e7a2096
```

Kiểm tra đã kết nối với cluster của DigitalOcean chưa, ta vào terminal và thực hiện dòng lệnh sau

```
kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
pool-sgpu0w5mt-31op3   Ready   <none>   44h   v1.19.3   10.104.0.3   128.199.186.119   Debian GNU/Linux 10 (buster)   4.19.0-11-amd64   docker://19.3.13
```

- Với pool-sgpu0w5mt-31op3 là tên của node trong cụm cluster vừa được tạo
- Node chạy trên hệ điều hành Debian GNU/Linux 10 (buster)

Để bắt đầu triển khai server back-end của ứng dụng lên cụm cluster vừa tạo ta cần thực hiện trước:

- Docker image chứa source code back-end. Có nhiều công cụ giúp ta lưu trữ và quản lý các docker image như: docker hub, docker registry, gitlab registry, ...
Ở đây dùng gitlab registry để build và lưu trữ image
- File cấu hình triển khai: ở đây ta cần triển khai app sử dụng rest Api và database sử dụng postgres, ta sử dụng file yaml để khai báo từng thành phần và đọc file để triển khai, tất cả bao gồm những file như sau:
 - app-configmap.yaml
 - app-deployment.yaml
 - app-service.yaml
 - postgres-config.yaml
 - postgres-storage.yaml
 - postgres-stafulset.yaml
 - postgres-service.yaml

Ta sẽ bắt đầu triển khai từng thành phần lên cluster của DigitalOcean với những file yaml ở trên

- **postgres-configmap.yaml**: file này sử dụng loại ConfigMap để triển khai những biến môi trường được sử dụng để kết nối tới database.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
  labels:
    app: postgres
data:
  POSTGRES_DB: fladdb
  POSTGRES_USER: flad
  POSTGRES_PASSWORD: flad
```

- Thực hiện lệnh sau để triển khai:

```
kubectl apply -f postgres-configmap.yaml
```

- Thực hiện lệnh sau để xoá:

```
kubectl delete configmap postgres-config
```

- **postgres-storage.yaml**: file này dùng hai loại Persistent Volume và Persistent Volume Claim dùng để lưu trữ data trên kubernetes

```
kind: PersistentVolume
```

```
apiVersion: v1

metadata:
  name: postgres-pv-volume
  labels:
    type: local
    app: postgres
spec:
  storageClassName: manual
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
  ---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: postgres-pv-claim
  labels:
    app: postgres
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

- Thực hiện lệnh sau để triển khai:

```
kubectl apply -f postgres-storage.yaml
```

- Thực hiện lệnh sau để xoá:

```
kubectl delete pv postgres-pv-column  
kubectl delete pvc postgres-pvc-claim
```

- **postgres-statefulset.yaml**: file này dùng StatefulSet để triển khai postgres container, sử dụng images từ debezium/postgres:12. Sử dụng cấu hình với config đã cấu hình sẵn ở ConfigMap và mount Volumn vào trong pv, pvc vừa tạo ở trên.

```
apiVersion: apps/v1  
kind: StatefulSet  
metadata:  
  name: postgres  
spec:  
  serviceName: "postgres"  
  selector:  
    matchLabels:  
      app: postgres  
  template:  
    metadata:  
      labels:  
        app: postgres
```

```
spec:
  containers:
    - name: postgres
      image: debezium/postgres:12
      imagePullPolicy: "IfNotPresent"
      ports:
        - containerPort: 5432
      volumeMounts:
        - name: postgredb
          mountPath: /var/lib/postgresql/data
      envFrom:
        - configMapRef:
            name: postgres-config
      volumes:
        - name: postgredb
      persistentVolumeClaim:
        claimName: postgres-pv-claim
```

- Thực hiện lệnh sau để triển khai:

```
kubectl apply -f postgres-statefulset.yaml
```

- Thực hiện lệnh sau để xoá:

```
kubectl delete statefulset postgres
```

- **postgres-service.yaml**: để truy cập vào StatefulSet vừa tạo, ta cần một postgres service để expose ra ip và cổng kết nối để truy cập vào.

```
apiVersion: v1
kind: Service
metadata:
  name: postgres
  labels:
    app: postgres
spec:
  type: NodePort
  ports:
  - port: 5432
    targetPort: 5432
    nodePort: 31001
  selector:
    app: postgres
```

- Thực hiện lệnh sau để triển khai:

```
kubectl apply -f postgres-service.yaml
```

- Thực hiện lệnh sau để xoá:

```
kubectl delete service postgres
```

- Ở đây sau khi triển khai ta thấy có hai địa chỉ IP: cluster-ip và external-ip và port 31001 sẽ được expose ra. Ta có thể sử dụng external-ip cùng với port 31001 để truy cập vào database.

- **app-configmap.yaml**: file này sử dụng loại ConfigMap để triển khai những biến môi trường được sử dụng trong app của ta.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  labels:
    app: app
data:
  ADMIN_EMAIL: admin@gmail.com
  GUEST_EMAIL: guest@gmail.com
  AWS_ACCESS_KEY_ID: AKIAJWI6SKSWNZSU5ZJQ
  AWS_SECRET_ACCESS_KEY: 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
  AWS_BUCKET_NAME: flad-hkteam
  AWS_BUCKET_REGION: ap-southeast-1
  AWS_S3_SIGNED_URL_EXPIRY: "3600"
  AWS_S3_PUBLIC_URL: https://flad-hkteam.s3-ap-southeast-1.amazonaws.com
  AWS_UPLOADS_BUCKET_NAME: flad-hkteam
  DB_CONN: postgres://flad:flad@postgres:5432/fladdb
```

- Thực hiện lệnh sau để triển khai:

```
Kubectl apply -f app-configmap.yaml
```

- Thực hiện lệnh sau để xoá:

```
kubectl delete configmap flad-api
```

- **app-deployment.yaml**: file này dùng Deployment để triển khai một nhóm các pod, và các nhóm được nhân bản. Ở đây khởi tạo thêm một container với image là prep, container được dùng để migration tạo mới những table ở database.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  # tên của deployment
  name: flad-api
  annotations:
    reloader.stakater.com/auto: "true"
spec:
  # Số Pod tạo ra
  replicas: 3
  # Thiết lập các POD do deploy quản lý, là POD có nhãn "app=flad-api"
  selector:
    matchLabels:
      app: flad-api
  # Định nghĩa mẫu POD, khi cần Deploy sử dụng mẫu này để tạo Pod
  template:
    metadata:
      labels:
        app: flad-api
    spec:
      # Khởi tạo container prep, dùng để migration database
      initContainers:
```

```
- name: prep

    image: registry.gitlab.com/kltn2/app-backend/prep:latest

    command:
      - ./bin/run

    args:
      - --migrate

    envFrom:
      - configMapRef:
          name: app-config

# Khởi tạo container của ứng dụng, với image được push lên gitlab
registry

  containers:
    - name: app

      image: registry.gitlab.com/kltn2/app-backend/flad:latest

      # Giới hạn tài nguyên sử dụng

      resources:
        limits:
          cpu: "0.5"
          memory: "600M"

        requests:
          cpu: "0.04"
          memory: "50M"

      ports:
        - containerPort: 1234

      envFrom:
        - configMapRef:
            name: app-config

# Kiểm tra xem container còn sống hay không
```

```
livenessProbe:  
  httpGet:  
    path: /  
    port: 1234  
  initialDelaySeconds: 30  
  periodSeconds: 3  
  timeoutSeconds: 5
```

- Thực hiện lệnh sau để triển khai:

```
kubectl apply -f app-deployment.yaml
```

- Thực hiện lệnh sau để xoá:

```
kubectl delete deployment flad-api
```

- **app-service.yaml**: Để truy cập bên ngoài vào deployment trên thì ta cần phải triển khai một service, kiểu service này dùng LoadBalancer để cân bằng tải các endpoints khi gọi tới địa chỉ được expose ra với port 1234.

```
apiVersion: v1  
kind: Service  
metadata:  
  name: flad-api  
spec:  
  selector:  
    app: flad-api  
  ports:
```

```
- name: http  
  
port: 1234  
  
targetPort: 1234  
  
protocol: TCP  
  
type: LoadBalancer
```

- Thực hiện lệnh sau để triển khai:

```
kubectl apply -f app-service.yaml
```

- Thực hiện lệnh sau để xoá:

```
kubectl delete service flad-api
```

- Ta có thể kiểm tra lại những triển khai đã tạo ở trên thông qua một số dòng lệnh cơ bản sau:

```
kubectl get nodes -o wide // Danh sách các node trong cluster
```

```
kubectl get pods -o wide // Danh sách các pods có trong node
```

```
kubectl get svc -o wide // Danh sách các service
```

```
kubectl get deployment -o wide // Danh sách các deployment
```

```
kubectl get pv -o wide // Danh sách các persistent volume
```

```
kubectl get pvc -o wide // Danh sách các persistent volume claim
```

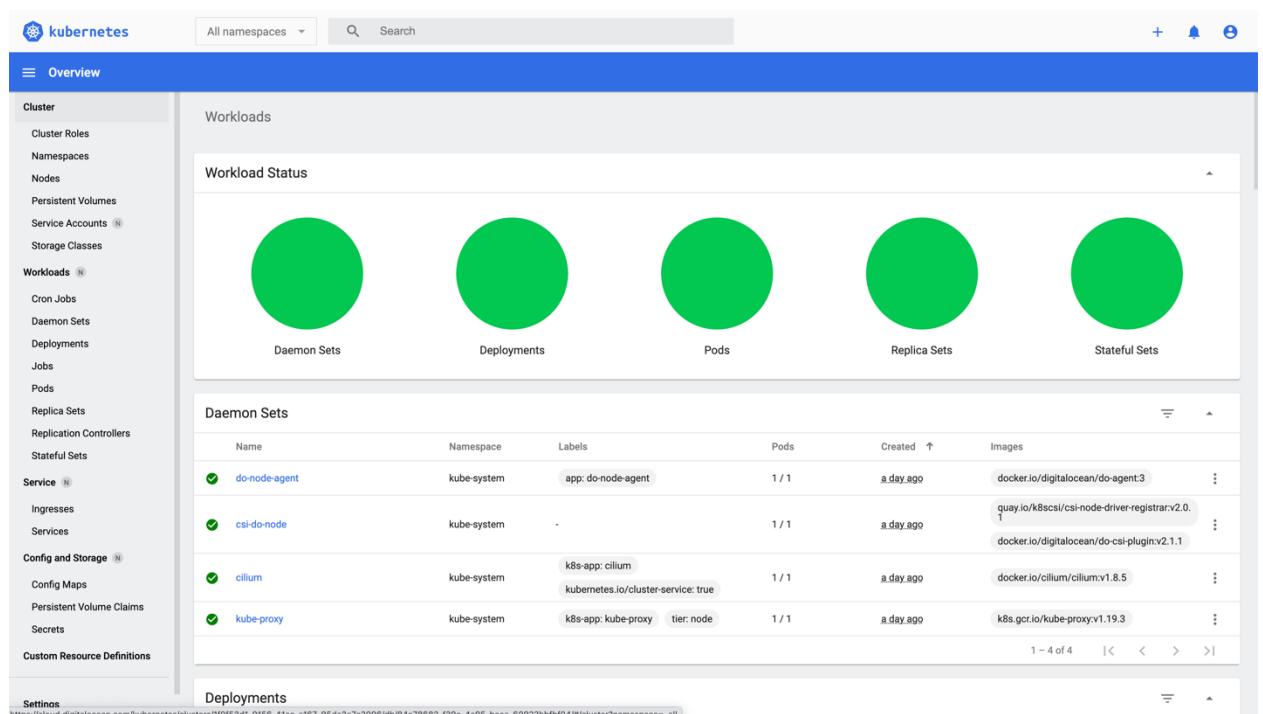
```
kubectl get statefulset -o wide // Danh sách các stateful set
```

```
kubectl get configmap -o wide // Danh sách các config map
```

- Ngoài ra ta có thể xem nhiều thông tin hơn thông tin hơn thông qua một số lệnh kubectl cung cấp, có thể tham khảo trực tiếp tại trang chủ của kubernetes.

Sau khi triển khai ứng dụng với các file yaml có sẵn trên ta có thể trực tiếp vào dashboard kubernetes mà DigitalOcean cung cấp để kiểm tra hoặc có thể trực tiếp thêm xoá, sửa những thành phần đã triển khai ở trên thông qua giao diện kubernetes dashboard.

Một số hình ảnh những thành phần triển khai thành công được hiển thị trên kubernetes dashboard của DigitalOcean.



Hình 4-5: Kubernetes dashboard

The screenshot shows the Kubernetes Nodes page. On the left, there's a sidebar with navigation links for Cluster, Workloads, Services, Config and Storage, and Settings. The main area is titled 'Nodes' and contains a table with one row. The table columns are: Name, Labels, Ready, CPU requests (cores), CPU limits (cores), Memory requests (bytes), Memory limits (bytes), and Created. The node 'pool-sgpu0w5mt-31op3' is listed with the following details:

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Created
pool-sgpu0w5mt-31op3	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/instance-type: s-1vc pu-2gb	True	702.00m (70.20%)	102.00m (10.20%)	762.68Mi (38.22%)	2.29Gi (117.6%)	a day ago

At the bottom of the table, there's a link 'Show all'.

Hình 4-6: Danh sách Node trong cụm cluster

The screenshot shows the Kubernetes Deployments page. On the left, there's a sidebar with navigation links for Cluster, Workloads, Services, Config and Storage, and Settings. The main area is titled 'Deployments' and contains a table with four rows. The table columns are: Name, Namespace, Labels, Pods, Created, and Images. The deployments listed are:

Name	Namespace	Labels	Pods	Created	Images
flad-api	default	-	1 / 1	6 hours ago	registry.gitlab.com/ktn2/app-backend/flad1@latest
doks-example	default	-	2 / 2	a day ago	digitalocean/doks-example
coredns	kube-system	k8s-app: kube-dns kubernetes.io/name: CoreDNS	2 / 2	a day ago	docker.io/coredns/coredns:1.7.1
cilium-operator	kube-system	io.cilium/app: operator name: cilium-operator	2 / 2	a day ago	docker.io/cilium/operator:v1.8.5

At the bottom of the table, there's a link '1 - 4 of 4'.

Hình 4-7: Danh sách Deployment trong cụm cluster

The screenshot shows the Kubernetes Services page. On the left, there's a sidebar with navigation links for Cluster, Workloads, Service, Ingresses, and Settings. The main area displays a table of services:

Services								
	Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Created	
	flad-api	default	-	10.245.211.168	flad-api:1234 TCP flad-api:30022 TCP	159.89.211.127:1234	6 hours ago	
	postgres	default	app: postgres	10.245.253.147	postgres:5432 TCP postgres:31001 TCP	-	a day ago	
	doks-example	default	-	10.245.137.157	doks-example:80 TCP doks-example:30200 TCP	104.248.98.95:80	a day ago	
kube-dns						kube-system	k8s-app: kube-dns kubernetes.io/cluster-service: true Show all	
						10.245.0.10	kube-dns:53 UDP kube-dns:53 TCP kube-dns:9153 TCP kube-dns:0 TCP	a day ago
	kubernetes	default	component: apiserver provider: kubernetes	10.245.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	a day ago	

At the bottom right, there are navigation links: 1 - 5 of 5, <, >, and >>.

Hình 4-8: Danh sách Service trong cụm cluster

The screenshot shows the Kubernetes Pods page. On the left, there's a sidebar with navigation links for Cluster, Workloads, Service, Ingresses, and Settings. The main area displays a table of pods:

Pods									
	Name	Namespace	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
	flad-api-77bf877bc4-kqv4l	default	app: flad-api pod-template-hash: 77bf877bc4	pool-sgpu0w5mt-31op3	Running	0	-	-	38 seconds ago
	postgres-0	default	app: postgres controller-revision-hash: postgres-6bbf66777	pool-sgpu0w5mt-31op3	Running	0	-	-	a day ago
	doks-example-8fcccb546-29gh	default	app: dok-example pod-template-hash: 8fcccb546	pool-sgpu0w5mt-31op3	Running	0	-	-	a day ago
	doks-example-8fcccb546-148e2	default	app: dok-example pod-template-hash: 8fcccb546	pool-sgpu0w5mt-31op3	Running	0	-	-	a day ago
	cilium-dmkjz	kube-system	controller-revision-hash: 7c7fd9644 k8s-app: cilium	pool-sgpu0w5mt-31op3	Running	0	-	-	a day ago
	kube-proxy-2nng8	kube-system	controller-revision-hash: 697dfb8c6 k8s-app: kube-proxy	pool-sgpu0w5mt-31op3	Running	0	-	-	a day ago
	csi-do-node-nnfrj	kube-system	controller-revision-hash: 5b87fb9db Show all	pool-sgpu0w5mt-31op3	Running	0	-	-	a day ago
	do-node-agent-g89lp	kube-system	controller-revision-hash: b8d9d74d9 Show all	pool-sgpu0w5mt-31op3	Running	0	-	-	a day ago
	cilium-operator-5f487bb876-tbh4t	kube-system	io.cilium/app: operator name: cilium-operator	pool-sgpu0w5mt-31op3	Running	2	-	-	a day ago

Hình 4-9: Tạo mới Pods trong cụm cluster

Sau khi ta triển khai thành công hệ thống sẽ tạo ra một service tên là flad-api với địa chỉ IP **159.89.211.127**, khi pod truy cập địa chỉ này với cổng **1234** thì nó truy cập đến các endpoint được định nghĩa trong service.

Dưới đây là một số hình ảnh truy cập thành công endpoint của service được test trên **insomnia** tool

The screenshot shows the Insomnia REST Client interface. The left sidebar lists various API endpoints under the 'FLAD - KLTN' project. The main area shows a POST request to `http://159.89.211.1234/auth/local`. The request body contains:

```

{
  "email": "guest@gmail.com",
  "password": "123456"
}

```

The response tab shows a **200 OK** status with a response time of 127 ms and a response size of 646 B. The response body is a JSON object:

```

{
  "id": 2,
  "email": "guest@gmail.com",
  "name": "guest",
  "role": "user",
  "provider": "local",
  "ok": true,
  "token": "eyJhbGciOiJIbGxlbWVudC1hbGciOJU1I1Uz1NLj9...eyJpZC10MjwibmFtZS16Ik1lZWNTIi1GhbmU0m5b9w5ImYtYt1I1I1c1Z3V1c3RAZ21hbmw0Y2811iw1yRkcm1zc16nVsbcic0f23dvcm1011MhEMETake0910mRn0l0l022XMaRnb1REZMyME00000vrc1JwCu0c1TA1VPh2d1E02Z6TUpzTT1L1JmW1Ym9va191i2VX21k1puwdwLc1Jy2N1l3Nfd69r2410hs16gw1Jv0g0101J1c2V1lwiw1Nt201z2R1Jc1b7afsc0s11AnHtX0c1y109w1v3Y1YR1Z79nCt61j1M01EMD1MD1UMt06ND0M1juw1k2W11sInw2G1F281Ht1Q1011m031fa1Ta1yv0E010r0111j1SM1o1f0_dqggpB54-7edvAbv1000RfBnp17514PTU1AHAT0"
}

```

Hình 4-10: API đăng nhập

The screenshot shows a browser window for the FLAD - KLTN (Development) API. The URL is `http://159.89.211.127:1234/categories`. The response status is **200 OK**, with a response time of **17.8 ms** and a size of **371 B**. The response was received **5 Days Ago**.

The left sidebar shows a tree view of the API structure. Under the **Categories** node, the **Get** method is selected. The right panel displays the API response:

```
1: {  
2:   "message": "Get Category successfully !",  
3:   "status": true,  
4:   "data": [  
5:     {  
6:       "id": 1,  
7:       "parentId": null,  
8:       "name": "Ao quan Han Quoc",  
9:       "sku": "AOHQ",  
10:      "stockOption": "Hang nhap tu han quoc",  
11:      "isDelete": false,  
12:      "status": 1,  
13:      "createdAt": "2021-01-08T01:25:21.254Z",  
14:      "updatedAt": "2021-01-08T01:25:21.254Z"  
15:    }  
16:  ]  
17: }
```

Hình 4-11: API xem danh sách loại hàng hoá

Chương 5. KẾT LUẬN, HƯỚNG PHÁT TRIỂN

5.1. Ưu điểm

Hoàn thành yêu cầu đặt ra ban đầu, cơ bản có đầy đủ tính năng cần thiết của một ứng dụng chốt đơn trên nền tảng livestream của facebook.

Giao diện đơn giản, bố cục hợp lý, người dùng dễ dàng theo tác trên ứng dụng.

Tốc độ phản hồi các thao tác nhanh, tối ưu hóa năng suất của người dùng.

Sinh viên thực hiện đã nắm được thêm về quy trình nghiệp vụ của ứng dụng bán hàng.

Tìm hiểu và ứng dụng tốt được một số công nghệ mới trong việc phát triển web hiện nay như: vuejs, koa, docker, Facebook API, ...

Hoàn thành việc tìm hiểu và ứng dụng Kubernetes vào việc xây dựng ứng dụng. Giúp cho ứng dụng hoạt động ổn định, dễ bảo trì và nâng cấp.

5.2. Nhược điểm

Giao diện chỉ hiển thị tốt trên desktop, hiển thị không tối ưu trên mobile.

Ứng dụng chỉ mang tính chất minh họa, chưa thể ứng dụng trực tiếp vào thực tế.

Facebook API sử dụng vẫn đang ở chế độ phát triển, chưa thể đưa vào thực tế, do một số nguyên nhân về vấn đề đánh giá của facebook đối với ứng dụng.

Một số tính năng vẫn còn hạn chế:

- Chưa phát triển tính năng xác thực tài khoản qua mail, số điện thoại cũng như chưa phát triển tính năng lấy lại mật khẩu.
- Tính năng đồng bộ sản phẩm lên facebook Catalog để phục vụ việc quảng cáo hàng hoá vẫn chưa hoàn thành do một số hạn chế từ facebook API.
- Một số tính năng sử dụng facebook API đôi khi vẫn bị lỗi.
- Chưa xử lý được những lỗi gặp phải đối với việc triển khai với kubernetes như khi server chết, quá tải, scale, nhân bản, ...

5.3. Hướng phát triển

Xây dựng các chửa năng còn thiếu, hoàn thiện các chức năng đã có.

Xây dựng cải tiến giao diện và xây dựng thêm giao diện mobile cho ứng dụng.

Tối ưu hóa code để tăng tốc độ cho ứng dụng.

Tối ưu hóa CSDL hợp lý hơn.

Xử lý những lỗi gặp phải khi sử dụng facebook API.

Xử lý những lỗi gặp phải khi triển khai ứng dụng với kubernetes.

Phát triển và đưa ứng dụng đi vào thực tế.

Học hỏi và tìm hiểu thêm về nhiều công nghệ liên quan đến việc phát triển ứng dụng đa nền tảng, đặc biệt là ứng dụng được kubernetes trong công việc sau này.

TÀI LIỆU THAM KHẢO

- [1] Document AWS S3

<https://docs.aws.amazon.com/AmazonS3/latest/API>Welcome.html>

- [2] Document docker

<https://docs.docker.com/get-started>

- [3] Document DigitalOcean

<https://www.digitalocean.com/docs/>

- [4] Document Nodejs

<https://nodejs.org/dist/lastest-v15.x/docs/api>

- [5] Document Koa

<http://koajs.com>

- [6] Document Kubernetes

<https://kubernetes.io/docs/home>

<https://v1-16.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.16/#-strong-api-overview-strong->

<https://minikube.sigs.k8s.io/docs/start/>

<https://xuanthulab.net/gioi-thieu-va-cai-dat-kubernetes-cluster.html>

<https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-architecture>

- [7] Document Facebook API

<https://developers.facebook.com/docs/graph-api>

<https://developers.facebook.com/docs/facebook-login>

<https://developers.facebook.com/docs/business-sdk>

<https://developers.facebook.com/docs/pages>

<https://developers.facebook.com/docs/messenger-platform>

<https://developers.facebook.com/docs/live-video-api>

[8] Document Postgresql

<https://www.postgresql.org/docs/13/index.html>

<http://knexjs.org>

[9] Document Vuejs

<https://vuejs.org/v2/guide>

[10] The new stack, The State of the Kubernetes ecosystem, Alex Williams,
Founder & Editor-in-Chief

[11] Primevue

<https://primefaces.org/primevue/showcase/#/setup>