|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI



phạm văn trường

**XÂY DỰNG WEBSITE OHOTOUR VÀ QUẢN LÝ CẤU HÌNH TRIỂN KHAI TRÊN DOCKER**

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2021

|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI

PHẠM VĂN TRƯỜNG

**XÂY DỰNG WEBSITE OHOTOUR VÀ QUẢN LÝ CẤU HÌNH TRIỂN KHAI TRÊN DOCKER**

|  |  |
| --- | --- |
| Ngành : | Công nghệ thông tin |
| Mã ngành: | 7480201 |

|  |  |
| --- | --- |
| GIÁO VIÊN HƯỚNG DẪN: | THS. NGUYỄN VĂN NAM |
|  |  |

HÀ NỘI, NĂM 2021

|  |  |
| --- | --- |
| Logo-WRU | ***CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM***  **Độc lập - Tự do - Hạnh phúc**  ----------★----------  **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP** |

Họ tên sinh viên: Phạm Văn Trường Hệ đào tạo : Đại học chính quy

Lớp: 58TH4 Ngành: Công nghệ thông tin

Khoa: Công nghệ thông tin

1. TÊN ĐỀ TÀI:

Xây dựng website ohotour và quản lý cấu hình triển khai trên Docker

2- CÁC TÀI LIỆU CƠ BẢN:

|  |  |
| --- | --- |
| [1] | V. Hoang, "Tổng quan về ảo hóa VMWare," [Online]. Available: https://www.thegioimaychu.vn/blog/ao-hoa/huong-dan-tong-quan-ve-ao-hoa-vmware-p3588/. |
| [2] | AnhQT, "TỔNG QUAN VỀ CÔNG NGHỆ ẢO HÓA VMware – Part2," [Online]. Available: https://itforvn.com/tong-quan-ve-cong-nghe-ao-hoa-vmware-part2/. |
| [3] | C. Tín, "Giới thiệu tổng quan về kiến trúc của Docker," [Online]. Available: https://tel4vn.edu.vn/blog/gioi-thieu-tong-quan-ve-kien-truc-cua-docker/. |
| [4] | C. Tín, "SO SÁNH SỰ KHÁC NHAU GIỮA MÁY ẢO VÀ DOCKER CONTAINER," [Online]. Available: https://tel4vn.edu.vn/blog/so-sanh-su-khac-nhau-giua-may-ao-va-docker-container/. |

3 - NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN: Tỷ lệ %

|  |  |
| --- | --- |
| Phần | Mức độ hoàn thành |
| Chương 1: Tổng quan về cơ sở lý thuyết | 30% |
| Chương 2: Phân tích thiết kế hệ thống | 35% |
| Chương 3: Kết quả cài đặt | 35% |

4. GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN

|  |  |
| --- | --- |
| Phần | Họ tên giáo viên hướng dẫn |
| Chương 1: Tổng quan về cơ sở lý thuyết | ThS. Nguyễn Văn Nam |
| Chương 2: Phân tích thiết kế hệ thống | ThS. Nguyễn Văn Nam |
| Chương 3: Kết quả cài đặt | ThS. Nguyễn Văn Nam |

5. NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Ngày 15 tháng 03 năm 2021

|  |  |
| --- | --- |
| **Trưởng Bộ môn**  *(Ký và ghi rõ Họ tên)* | **Giáo viên hướng dẫn chính**    **ThS. Nguyễn Văn Nam** |

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua

Ngày…. tháng…. năm 20…… **Chủ tịch Hội đồng**

*(Ký và ghi rõ Họ tên)*

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày 03 tháng 12 năm 2021

**Sinh viên làm Đồ án tốt nghiệp**

*(Ký và ghi rõ Họ tên)*

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

TÊN ĐỀ TÀI: XÂY DỰNG WEBSITE OHOTOUR VÀ QUẢN LÝ CẤU HÌNH TRIỂN KHAI TRÊN DOCKER

*Sinh viên thực hiện*: *Phạm Văn Trường*

*Lớp*: *58TH4*

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

**TÓM TẮT ĐỀ TÀI**

Thói quen du lịch và nghỉ mát trong thời đại kinh tế phát triển như hiện nay là rất cao. Mọi người không chỉ cần tìm đến những nơi nổi tiếng, danh lam thắng cảnh mà còn xuất hiện nhiều hình thức du lịch, nghỉ mát dành cho gia đình hay các tổ chức cung cấp những dịch vụ vui chơi, hoạt động ngoài trời. Thị trường home-sharing ra đời giúp các nhà cung cấp dịch vụ lớn nhỏ có thể kết nối với nhau hoặc với khách hàng và họ chia sẻ nhà giúp các chủ nhà mang lại thu nhập và tận dụng được nguồn cung bất động sản.

Website OhoTour được phát triển dựa vào mô hình home-sharing giúp khách hàng và các chủ cung cấp dịch vụ liên hệ cùng nhau một các trực quan hơn vừa để khách hàng có thể chọn dịch vụ và nhà cung cấp phù hợp vừa để các nhà cung cấp quản lý được các dịch vụ và các chuyến du lịch của khách hàng.

Website sẽ sử dụng ReactJS là một thư viện của JavaScript để xây dựng giao diện và xử lý logic dữ liệu. Kết hợp với xây dựng website, Docker sẽ được sử dụng để triển khai nó giúp cho việc quản lý cấu hình một cách tự động và nhanh chóng hơn.

**CÁC MỤC TIÊU CHÍNH**

* Mục tiêu 1: tìm hiểu và thực hiện Đồ án tốt nghiệp bằng ReactJS, Docker.
* Mục tiêu 2: ứng dụng Docker trong triển khai website OhoTour.

**KẾT QUẢ DỰ KIẾN**

* Hoàn thành website Ohotour giúp đặt chỗ ở, homestay cho chuyến đi nghỉ.
* Triển khai được phần mềm sử dụng Docker.
* Tổng hợp thông tin, viết báo cáo đồ án tốt nghiệp.

**LỜI CAM ĐOAN**

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

|  |  |
| --- | --- |
|  | **Tác giả ĐATN**  *Trường*  **Phạm Văn Trường** |

**LỜI CÁM ƠN**

Được sự hướng dẫn của ThS Nguyễn Văn Nam và sự đồng ý của Khoa Công nghệ thông tin trường Đại Học Thủy Lợi, em đã được nhận đề tài *“***Xây dựng website ohotour và quản lý cấu hình triển khai trên docker**.*”* cho đồ án tốt nghiệp kết thúc 4,5 năm quá trình học tại trường mình.

Để có được điều này, trước hết cho phép em được gửi lời cảm ơn chân thành đến toàn thể các thầy cô giáo trong trường Đại học Thủy Lợi nói chung và đặc biệt là các thầy cô giáo trong khoa Công nghệ thông tin nói riêng, những người đã tận tình dạy dỗ, truyền đạt cho em những kiến thức, kĩ năng học tập trong suốt thời gian học tập tại trường và tạo mọi điều kiện tốt nhất để em hoàn thành các môn học tại trường cũng như là đồ án tốt ngiệp này. Đặc biệt hơn hết, em xin gửi lời cảm ơn chân thành nhất tới thầy giáo – Ths Nguyễn Văn Nam - Giảng viên khoa Công nghệ thông tin, người đã nhiệt tình chỉ bảo giúp đỡ em rất nhiều trong suốt quá trình hoàn thành đồ án từ lúc phân tích tới thiết kế và hoàn thành đồ án.

Mặc dù nhiều cố gắng để đạt được kết quả tốt cho đề tài, nhằm phản ánh thực lực trong quá trình học tập và rèn luyện tại Trường Đại học Thủy Lợi, song do những hạn chế về kiến thức và kinh nghiệm nên không thể tránh khỏi những thiếu sót. Em rất mong sự góp ý của các Giảng viên để em ngày một cải thiện khả năng hơn trong học tập cũng như trong công việc sau này.

Em xin chân thành cảm ơn!

**MỤC LỤC**

[MỞ ĐẦU 2](#_Toc76417925)

[CHƯƠNG 1: TỔNG QUAN VỀ CƠ SỞ LÝ THUYẾT 3](#_Toc76417926)

[1.1 Ảo hóa 3](#_Toc76417927)

[1.1.1 Ảo hóa là gì? 3](#_Toc76417928)

[1.1.2 Các thành phần của một hệ thống ảo hóa 4](#_Toc76417929)

[1.1.3 Mục tiêu của ảo hóa 5](#_Toc76417930)

[1.1.4 Ảo hoá ứng dụng 5](#_Toc76417931)

[1.1.5 Ưu điểm và nhược điểm của ảo hoá? 6](#_Toc76417932)

[1.2 Công nghệ ảo hóa Docker 8](#_Toc76417933)

[1.2.1 Khái niệm về công nghệ ảo hóa docker 8](#_Toc76417934)

[1.2.2 So sánh sự khác nhau giữa máy ảo(VM) và Docker container 9](#_Toc76417935)

[1.2.3 Kiến trúc của Docker và các thành phần trong nó 13](#_Toc76417936)

[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG WEBSITE QUẢN LÍ HÀNG HOÁ 24](#_Toc76417937)

[2.1 Yêu cầu và đối tượng sử dụng 24](#_Toc76417938)

[2.1.1 Quản lý yêu cầu 24](#_Toc76417939)

[2.1.2 Đối tượng sử dụng 26](#_Toc76417940)

[2.2 Biểu đồ tuần tự 26](#_Toc76417941)

[2.2.1 Usecase đăng nhập 26](#_Toc76417942)

[2.2.2 Usecase đăng xuất 27](#_Toc76417943)

[2.2.3 UseCase Quản lí danh mục sản phẩm 28](#_Toc76417944)

[2.2.4 UseCase Quản lí sản phẩm 32](#_Toc76417945)

[2.2.5 UseCase Quản lí kho 37](#_Toc76417946)

[2.2.6 UseCase Quản lí khách hàng 42](#_Toc76417947)

[2.2.7 UseCase Quản lí nhà cung cấp 46](#_Toc76417948)

[2.2.8 UseCase Giao dịch 50](#_Toc76417949)

[CHƯƠNG 3: KẾT QUẢ CÀI ĐẶT 54](#_Toc76417950)

[3.1 Hệ thống trang web quản lí hàng hóa 54](#_Toc76417951)

[3.2 Deploy trên các môi trường bằng Docker 60](#_Toc76417952)

[KẾT LUẬN 67](#_Toc76417953)

[TÀI LIỆU THAM KHẢO 68](#_Toc76417954)

**DANH MỤC HÌNH ẢNH**

[Hình 1.1 2 loại phần mềm ảo hoá chính 4](#_Toc76330515)

[Hình 1.2 Ứng dụng của docker 8](#_Toc76330516)

[Hình 1.3 So sánh sự khác nhau giữa máy ảo (VM) và Docker container 9](#_Toc76330517)

[Hình 1.4 Kiến trúc của service trên máy ảo 10](#_Toc76330518)

[Hình 1.5 Kiến trúc của service trên Docker container 12](#_Toc76330519)

[Hình 1.6 Các thành phần của docker engine 14](#_Toc76330520)

[Hình 1.7 Kiến trúc của Docker 15](#_Toc76330521)

[Hình 2. 1 Biểu đồ tuần tự luồng đăng nhập 28](#_Toc76330541)

[Hình 2. 2 Biểu đồ tuần tự luồng đăng xuất 29](#_Toc76330542)

[Hình 2. 3 Biểu đồ Usecase quản lý danh mục sản phẩm 30](#_Toc76330543)

[Hình 2. 4 Biểu đồ tuần tự thêm danh mục sản phẩm 31](#_Toc76330544)

[Hình 2. 5 Biểu đồ tuần tự sửa danh mục sản phẩm 32](#_Toc76330545)

[Hình 2. 6 Biểu đồ tuần tự xóa danh mục sản phẩm 33](#_Toc76330546)

[Hình 2. 7 Biểu đồ Usecase quản lý sản phẩm 34](#_Toc76330547)

[Hình 2. 8 Biểu đồ tuần tự thêm sản phẩm 35](#_Toc76330548)

[Hình 2. 9 Biểu đồ tuần tự xem danh sách sản phẩm 36](#_Toc76330549)

[Hình 2. 10 Biểu đồ tuần tự sửa sản phẩm 37](#_Toc76330550)

[Hình 2. 11 Biểu đồ tuần tự xóa sản phẩm 38](#_Toc76330551)

[Hình 2. 12 Biểu đồ Usecase quản lý kho hàng 39](#_Toc76330552)

[Hình 2. 13 Biểu đồ tuần tự thêm kho hàng 40](#_Toc76330553)

[Hình 2. 14 Biểu đồ tuần tự sửa kho hàng 41](#_Toc76330554)

[Hình 2. 15 Biểu đồ tuần tự xóa kho hàng 42](#_Toc76330555)

[Hình 2. 16 Biểu đồ tuần tự xem danh sách sản phẩm tồn kho 43](#_Toc76330556)

[Hình 2. 17 Biểu đồ Usecase quản lý khách hàng 44](#_Toc76330557)

[Hình 2. 18 Biểu đồ tuần tự thêm khách hàng 45](#_Toc76330558)

[Hình 2. 19 Biểu đồ tuần tự sửa khách hàng 46](#_Toc76330559)

[Hình 2. 20 Biểu đồ tuần tự xóa khách hàng 47](#_Toc76330560)

[Hình 2. 21 Biểu đồ Usecase quản lý nhà cung cấp 48](#_Toc76330561)

[Hình 2. 22 Biểu đồ tuần tự thêm nhà cung cấp 49](#_Toc76330562)

[Hình 2. 23 Biểu đồ tuần tự sửa nhà cung cấp 50](#_Toc76330563)

[Hình 2. 24 Biểu đồ tuần tự xóa nhà cung cấp 51](#_Toc76330564)

[Hình 2. 25 Biểu đồ Usecase quản lý giao dịch 52](#_Toc76330565)

[Hình 2. 26 Biểu đồ tuần tự thêm phiếu nhập hàng 53](#_Toc76330566)

[Hình 2. 27 Biểu đồ tuần tự thêm phiếu bán hàng 54](#_Toc76330567)

[Hình 3. 1 Màn hình danh sách sản phẩm và sản phẩm 55](#_Toc76334444)

[Hình 3. 2 Màn hình tạo sản phẩm 56](#_Toc76334445)

[Hình 3. 3 Màn hình quản lí kho hàng 56](#_Toc76334446)

[Hình 3. 4 Màn hình danh sách phiếu nhập hàng 57](#_Toc76334447)

[Hình 3. 5 Màn hình tạo phiếu nhập hàng 57](#_Toc76334448)

[Hình 3. 6 Màn hình Danh sách sản phẩm tồn kho 58](#_Toc76334449)

[Hình 3. 7 Màn hình danh sách phiếu bán hàng 59](#_Toc76334450)

[Hình 3. 8 Màn hình tạo phiếu bán hàng 59](#_Toc76334451)

[Hình 3. 9 Màn hình danh sách khách hàng 60](#_Toc76334452)

[Hình 3. 10 Màn hình thêm khách hàng 60](#_Toc76334453)

[Hình 3. 11 Màn hình danh sách nhà cung cấp 61](#_Toc76334454)

[Hình 3. 12 Màn hình thêm nhà cung cấp 61](#_Toc76334455)

[Hình 3. 13 Danh sách container để chạy dự án 62](#_Toc76334456)

[Hình 3. 14 Dockerfile database 63](#_Toc76334457)

[Hình 3. 15 Dockerfile api 63](#_Toc76334458)

[Hình 3. 16 Dockerfile web\_build 64](#_Toc76334459)

[Hình 3. 17 File Docker compose 66](#_Toc76334460)

# MỞ ĐẦU

Ở buổi ban sơ của kỉ nguyên máy tính, quy trình phát triển phần mềm không hề có sự phân tách rạch ròi giữa hai giai đoạn phát triển (development) và vận hành (operations). Người kĩ sư đảm nhiệm việc lập trình, đồng thời cũng kiêm luôn việc kiểm thử, triển khai sản phẩm (và điều này cho đến nay vẫn đúng đối với các sản phẩm vừa và nhỏ).

Sau đó, sự bùng nổ về quy mô của các công ty và sản phẩm công nghệ kéo theo quy mô hệ thống phình ra theo cấp số nhân. Từ một vài server, hệ thống có thể phát triển lên đến hàng chục, hàng trăm, hàng nghìn, hoặc thậm chí hàng triệu server (ví dụ như trường hợp của Google, Facebook). Trước nhu cầu phát triển và cải tiến sản phẩm liên tục để đáp ứng thị trường đó, sự chia tách này lại bộc lộ những nhược điểm rõ rệt cần tối ưu. Từ đó Docker ra đời và là một giải pháp cho sự tối ưu hóa chu trình phát triển phần mềm, giúp sản phẩm IT được release nhanh và thường xuyên hơn.Docker là platform đứng thứ nhất trong danh sách bảng xếp hạng các platform mà Developer muốn sử dụng nhiều nhất khi phát triển ứng dụng, theo khảo sát của Stack Overflow 2019. Docker là một platform ở tầng OS (operating system) có thể tinh chỉnh được, phục vụ cho việc chạy ảo hóa các dịch vụ/ ứng dụng một cách nhanh chóng. Ngày trước, mọi người hay dùng máy ảo (hay còn gọi tắt là VMs – Virtual Machines) để triển khai ứng dụng, việc này mất rất nhiều thời gian. Khi chuyển sang sử dụng Docker thì thời gian triển khai một ứng dụng nằm trong Docker container được rút ngắn đi rất nhiều.

Và trong bản đồ án tốt nghiệp này, em sẽ tiến hành nghiên cứu về Docker thông qua đề tài: “Tối ưu hóa chu trình phát triển phần mềm bằng công cụ Docker”.

Nội dung chính của báo cáo được trình bày trong 3 chương:

Chương 1. Tổng quan về cơ sở lý thuyết

Chương 2. Phân tích thiết kế hệ thống website quản lý hang hoá

Chương 3. Kết quả cài đặt

# CHƯƠNG 1: TỔNG QUAN VỀ CƠ SỞ LÝ THUYẾT

Trước khi có công nghệ ảo hóa, hệ điều hành và các ứng dụng chạy trên một máy tính vật lý. Một mối quan hệ 1:1 tồn tại giữa một máy tính vật lý và hệ điều hành. Mối quan hệ này sử dụng ít công suất, chỉ khoảng 5 – 10% công suất của máy chủ vật lý. Khi muốn triển khai nhiều hệ điều hành thì phải có nhiều máy chủ vật lý. Mỗi lần nâng cấp phần cứng thì cần rất nhiều thời gian để mua, lắp ráp và cài đặt. Máy chủ dự phòng vật lý là một quá trình tốn nhiều thời gian. Trong các môi trường không ảo hóa thì thời gian được dành để mua phần cứng, lắp ráp, cài đặt hệ điều hành, bản cập nhật hệ điều hành, cài đặt và cấu hình các ứng dụng cần thiết có thể mất nhiều thời gian.

Mô hình này khá không linh hoạt và có nhiều chi phí phát sinh như chi phí đầu tư, không gian, điện năng tiêu thụ, hệ thống làm mát, chi phí bảo trì… Từ đó khái niệm ảo hoá ra đời.

## Ảo hóa

* + 1. **Ảo hóa là gì?**

Công nghệ ảo hóa là một công nghệ được ra đời nhằm khai thác triệt để khả năng làm việc của một máy chủ vật lý. Ảo hóa cho phép vận hành nhiều máy ảo trên cùng một máy chủ vật lý, dùng chung các tài nguyên của một máy chủ vật lý như CPU, Ram, ổ cứng, … và các tài nguyên khác. Các máy ảo khác nhau có thể vận hành hệ điều hành và ứng dụng trên cùng một máy chủ vật lý.

Công nghệ ảo hoá cho phép hợp nhất và chạy nhiều khối lượng công việc như các máy ảo trên một máy vi tính duy nhất. Một máy ảo là một máy tính được tạo ra bởi phần mềm, giống như một máy tính vật lý, chạy một hệ điều hành và các ứng dụng. Mỗi máy ảo có phần cứng ảo riêng của nó, bao gồm một CPU, bộ nhớ, đĩa cứng, và card mạng ảo, giống như phần cứng cho hệ điều hành và ứng dụng.

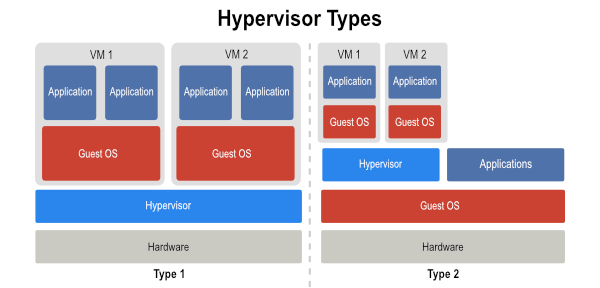
Máy chủ trong các hệ thống CNTT ngày nay thường được thiết kế để chạy một hệ điều hành và một ứng dụng. Điều này không khai thác triệt để hiệu năng của hầu hết các máy chủ rất lớn. Ảo hóa cho phép ta vận hành nhiều máy chủ ảo trên cùng một máy chủ vật lý, dùng chung các tài nguyên của một máy chủ vật lý qua nhiều môi trường khác nhau. Các máy chủ ảo khác nhau có thể vận hành nhiều hệ điều hành và ứng dụng khác nhau trên cùng một máy chủ vật lý. [1]

### Các thành phần của một hệ thống ảo hóa

Tài nguyên vật lý chính (Host machine / Host hardwave): [Máy chủ vật lý](https://www.thegioimaychu.vn/blog/thuat-ngu/bare-metal-server/), CPU, RAM, ổ đĩa cứng, card mạng… Nhiệm vụ là chia tài nguyên cấp cho các máy ảo.

* Phần mềm ảo hóa ([Hypervisor](https://www.thegioimaychu.vn/blog/thuat-ngu/hypervisor/)): cung cấp truy cập cho mỗi máy chủ ảo đến tài nguyên của [máy chủ vật lý](https://www.thegioimaychu.vn/blog/thuat-ngu/bare-metal-server/), lập kế hoạch và phân chia tài nguyên vật lý cho các máy chủ ảo, cung cấp giao diện quản lý cho các máy chủ ảo
* Hệ điều hành khách (Guest Operating System): được cài đặt trên một máy chủ ảo, thao tác như ở trên hệ điều hành thông thường.
* Mảy ảo (Virtual Machine): nó hoạt động như một máy chủ vật lý thông thường với tài nguyên riêng, giao diện riêng, hệ điều hành riêng.

Một hệ thống ảo hóa bắt buộc phải có đầy đủ các thành phần: tài nguyên vật lý, phần mềm ảo hóa, máy chủ ảo và hệ điều hành khách. Khi có đầy đủ 4 thành phần của hệ thống ảo hóa, người dùng có thể dễ dàng xây dựng cho mình một hệ thống ứng dụng ảo hóa hoàn chỉnh.



Hình 1.1 2 loại phần mềm ảo hoá chính

### Mục tiêu của ảo hóa

Ảo hóa hướng tới việc đảm bảo 4 đặc tính quan trọng gồm: Sẵn sàng (Availability), Ổn định (Scalability), Tối ưu (Optimization), Quản trị (Management).

* Sẵn sàng (Availability): giúp các ứng dụng hoạt động liên tục bằng cách giảm thiểu (bỏ qua) thời gian chết (downtime) khi phần cứng gặp sự cố, khi nâng cấp hoặc di chuyển.
* Ổn định (Scalability):  khả năng tùy biến, thu hẹp hay mở rộng mô hình server dễ dàng mà không làm gián đoạn ứng dụng.
* Tối ưu (Optimization): sử dụng triệt để nguồn tài nguyên phần cứng và tránh lãng phí bằng cách giảm số lượng thiết bị vật lý cần thiết (giảm số lượng server, switch, cáp, ...)
* Quản trị (Management): khả năng quản lý tập trung, giúp việc quản lý trở nên dễ dàng hơn bao giờ hết.

### Ảo hoá ứng dụng

Ảo hóa có 4 mức độ ảo hóa: Ảo hoá toàn phần, ảo hoá song song, ảo hoá hệ điều hành, ảo hoá ứng dụng. Ảo hóa ứng dụng là một dạng của công nghệ ảo hóa cho phép chúng ta tách rời mối liên kết giữa ứng dụng và hệ điều hành và cho phép phân phối lại ứng dụng phù hợp với nhu cầu user. Môt ứng dụng được ảo hóa sẽ không được cài đặt lên máy tính một cách thông thường, mặc dù ở góc độ người sử dụng, ứng dụng vẫn hoạt động một cách bình thường. Việc quản lý việc cập nhật phần mềm trở nên dễ dàng hơn, giải quyết sự đụng độ giữa các ứng dụng và việc thử nghiệm sự tương thích của chúng cũng trở nên dễ dàng hơn. Hiện nay đã có khá nhiều chương trình ảo hóa ứng dụng như VirtualBox, Vagrant, Docker, Citrix XenApp, Microsoft Application Virtualization, Vmware ThinApp… [2].

Ảo hóa ứng dụng là giải pháp tiến đến công nghệ "điện toán đám mây" cho phép ta sử dụng phần mềm của công ty mà không cần phải cài phần mềm này vào bất cứ máy tính con nào.

Giải pháp Ảo hóa ứng dụng cho ta những lợi ích nổi trội sau:

* + - Tất cả các máy tính đều có thể sử dụng phần mềm ảo như đang cài trên máy tính của mình mà không phải lo về cấu hình (ví dụ chạy Photoshop trên máy P4 chỉ có 512 MB RAM). Tốc độ phần mềm luôn ổn định và ko phụ thuộc vào cấu hình từng máy.
    - Các máy tính con luôn ở trong tình trạng sạch và chạy nhanh hơn. Lọai bỏ hòan tòan việc phải sửa lỗi phần mềm do virus, spyware hoặc do người dùng sơ ý.
    - Cho phép sử dụng phần mềm mà không phải quan tâm đến hệ điều hành ta đang sử dụng (ví dụ: ta có thể dùng Microsoft Office 2007 ngay trong Linux, Windows 98 hoặc MAC-OS).
    - Chúng ta có thể phân phối phần mềm 1 cách linh động đến 1 số cá nhân hoặc nhóm có nhu cầu sử dụng thay vì cài vào tất cả mọi máy như cách phổ thông. Việc phân phối hoặc gỡ bỏ phần mềm ra các máy tính có thể diễn ra chỉ trong vòng chỉ vài giây thay vì hàng tuần nếu như công ty có hàng chục máy tính.
    - Thông tin luôn luôn được lưu trữ an toàn ở server trung tâm thay vì có thể phân tán ra từng máy con. Cho dù chúng ta ở bất cứ nơi nào (tại 1 máy tính khác, tại nhà hay thậm chí ở internet cafe), việc truy nhập và sử dụng phần mềm của doanh nghiệp trở nên dễ dàng qua 1 hệ thống bảo mật hiện đại nhất.

Ảo hóa ứng dụng là giải pháp cho phép sử dụng và quản lý phần mềm doanh nghiệp 1 cách hiệu quả có hệ thống. Tiết kiệm tối đa chi phí bảo trì, hỗ trợ kỹ thuật và quản lý từng máy tính.

Trong ảo hóa ứng dụng thì công nghệ ảo hóa Docker đang được đánh giá là tương lai của công nghệ ảo hoá (future of virtualization), công nghệ này là sản phẩn của một dự án phần mềm tự do nguồn mở phát hành theo giấy phép Apache. Khác biệt lớn của Docker và các công nghệ ảo hóa khác đó là tiết kiệm đáng kể nguồn lực sử dụng. Với docker có thể chạy 20 container (tương tự như một hệ điều hành nhỏ) trên cùng một máy host (host machine), mà nếu sử dụng công nghệ ảo hóa khác như Vagrant sẽ cần một máy chủ với cấu hình rất lớn. Docker làm được điều này là bởi vì khác với Virtual Machine ở chỗ thay vì tách biệt giữa hai môi trường guest và host, thì các container của Docker chia sẻ các resource với host machine

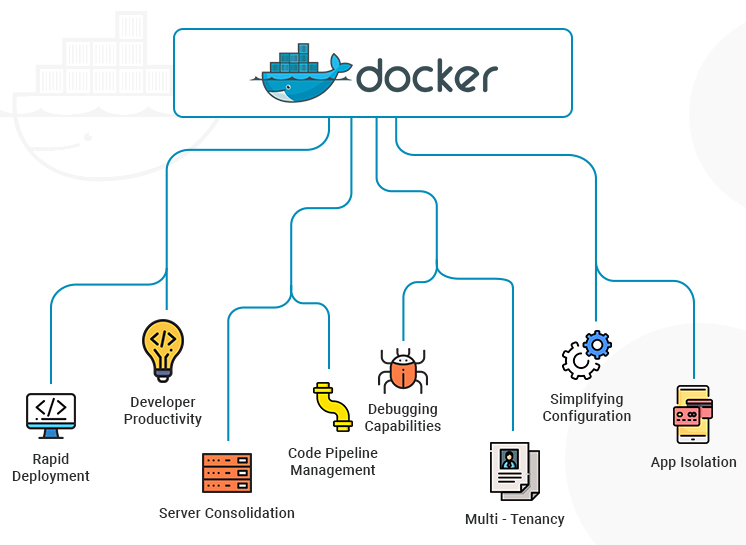
* + 1. **Ưu điểm và nhược điểm của ảo hoá?**
* Ưu điểm của ảo hóa: Ngày nay xu hướng ảo hóa máy chủ đã trở thành xu hướng chung của hầu hết các doanh nghiệp trên toàn thế giới. Những khó khăn trong thời kỳ khủng hoảng khiến cho các doanh nghiệp phải tìm mọi cách để giảm thiểu chi phí. Ảo hóa được coi là một công nghệ giúp các doanh nghiệp cắt giảm chi tiêu hiệu quả với khả năng tận dụng tối đa năng suất của các thiết bị phần cứng. Việc áp dụng công nghệ ảo hóa máy chủ đem có những ưu điểm như:
  + Tiết kiệm năng lượng tiêu thụ, giảm chi phí duy trì server (tiền điện để chạy và làm mát server)
  + Giảm số lượng thiết bị vật lý cần thiết (giảm số lượng server, switch, cáp, phí gia công)
  + Tận dụng tối đa nguồn tài nguyên, tránh lãng phí.
  + Quản lý tập trung, liên tục, nâng cao hiệu quả làm việc của quản trị viên.
  + Khả năng mở rộng dể dàng
* Nhược điểm.
  + Thông thường, mỗi máy ảo chỉ sử dụng một file VMDK (file này có thể được chia nhỏ tùy theo cách cài đặt) để lưu lại toàn bộ dữ liệu trong máy ảo và một số file nhỏ khác để lưu cấu hình máy ảo. Do đó, nếu một trong số những tệp tin bị lỗi hoặc bị mất mà chưa được backup thì có thể xem như máy ảo đã bị hư hoàn toàn và không thể phục hồi.
  + Ngoài ra nếu máy chủ có cấu hình phần cứng thấp nhưng lại có một máy ảo sử dụng quá nhiều tài nguyên hoặc chạy quá nhiều máy ảo sẽ làm chậm toàn bộ hệ thống bao gồm các máy ảo và các ứng dụng chạy trên máy ảo. Đồng thời do một hoặc vài máy chủ phải đảm nhận nhiều máy ảo chạy trên nó nên máy chủ gặp trục trặc, sự cố thì các máy ảo cũng sẽ bị ảnh hưởng theo.
  + Còn ở góc độ bảo mật, nếu hacker nắm quyền điều khiển một máy chủ vật lý chứa các máy ảo thì hacker có thể kiểm soát được tất cả các máy ảo trong nó. [1]

## Công nghệ ảo hóa Docker

### Khái niệm về công nghệ ảo hóa docker

Docker là một trong những phát triển tuyệt vời trong những năm gần đây trong lĩnh vực công nghệ thông tin. Docker đã mở ra một hướng mới cho các nhà phát triển phần mềm về xây dựng giải pháp và các kỹ thuật ứng dụng cho dự án mới. Docker giúp cho việc tổ chức, quản lý các môi trường lập trình được gọn gàng hơn, dễ dàng hơn.

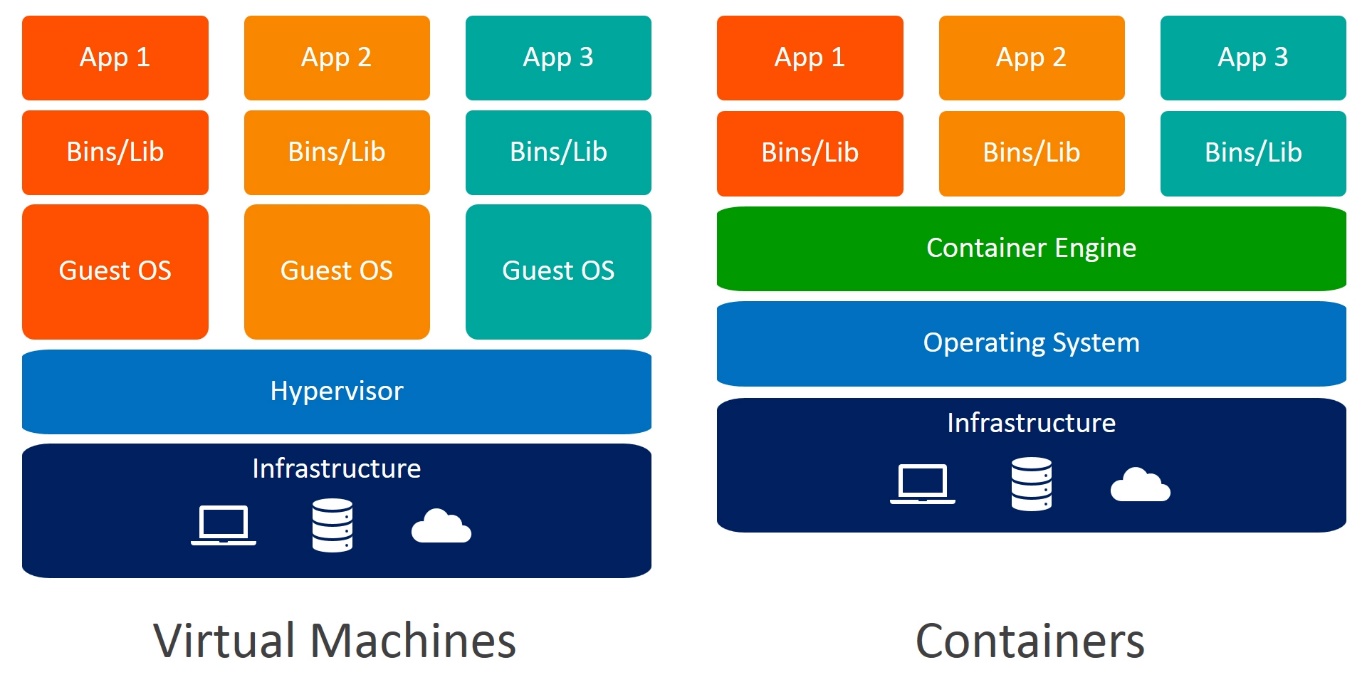
 Docker là một nền tảng miễn phí (open platform) dành cho việc phát triển, triển khai ứng dụng, cho phép người dùng có thể phát triển và triển khai phần mềm một cách nhanh chóng mà không phải quan tâm nhiều đến hạ tầng bên dưới.



Hình 1.2 Ứng dụng của docker

Docker cung cấp khả năng đóng gói và triển khai ứng dụng trong một môi trường cô lập được gọi là container. Với tính cô lập và bảo mật, Docker cho phép người dùng có thể triển khai nhiều container đồng thời trên một server (máy chủ) nhất định. Các container sử dụng rất ít dung lượng bộ nhớ vì chúng không cần sử dụng hypervisor, thay vào đó các container sẽ tương tác trực tiếp với nhân (kernel) của server. Người dùng có thể khởi chạy nhiều container trên một tổ hợp phần cứng nhất định so với việc sử dụng máy ảo, và thậm chí có thể chạy các Docker container trên các máy ảo. [3]

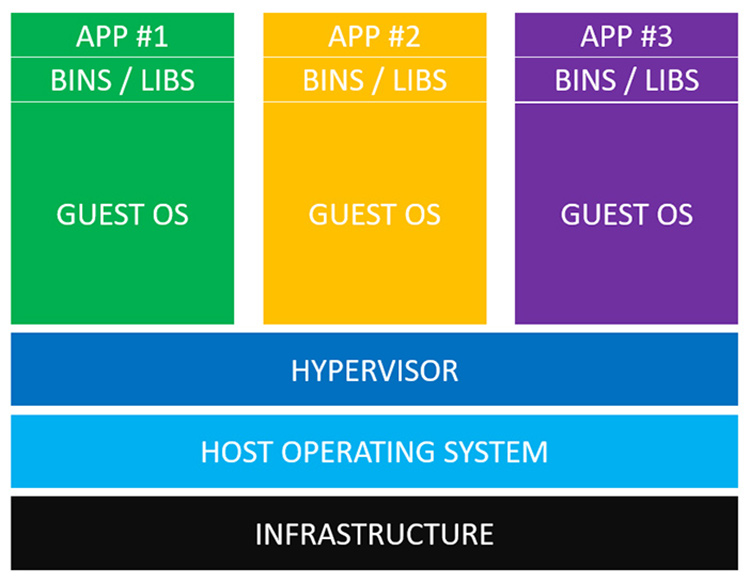
### So sánh sự khác nhau giữa máy ảo(VM) và Docker container



Hình 1.3 So sánh sự khác nhau giữa máy ảo (VM) và Docker container

**Kiến trúc của service hoạt động trên máy ảo**

Máy ảo (Virtual Machine – VM) là một mô phỏng của hệ thống máy tính. Nói một cách đơn giản, công nghệ này giúp người dùng có thể tạo ra nhiều “máy tính logic” trên một “máy tính vật lý”, dễ dàng quản lý vấn đề bảo mật. Kiến trúc của service khi hoạt động trên nền tảng máy ảo thông qua ảnh minh họa bên dưới:



Hình 1.4 Kiến trúc của service trên máy ảo

**Infrastructure (hạ tầng):**có thể là laptop, một server (máy chủ) chuyên dụng (dedicated server) hoat động trong một trung tâm dữ liệu (Data Center) hoặc một server riêng ảo (Virtual Private Server – VPS) trên cloud như DigitalOcean, Amazon, GCP, …

**Host Operating System (hệ điều hành của server):**hệ điều hành đang sử dụng cho server đang sử dụng, có thể là macOS, Windows hoặc Linux.

**Hypervisor** (phần mềm giám sát máy ảo): có thể coi máy ảo như một máy tính độc lập được đóng gói vào một tập tin duy nhất, nhưng cần phải có một phần mềm để có thể chạy tập tin đó. Các hypervisor được sử dụng để tạo, startup, dừng và reset lại các máy ảo, cho phép mỗi máy ảo hoặc “guest” truy cập vào lớp tài nguyên phần cứng vật lý bên dưới, chẳng hạn như CPU, RAM và ổ cứng. Phần mềm này cũng có thể giới hạn số lượng tài nguyên hệ thống mà mỗi máy ảo có thể sử dụng để đảm bảo cho nhiều máy ảo cùng hoạt động đồng thời trên một hệ thống. Có thể kể đến các hypervisor phổ biến trên thị trường hiện nay như HyperKit cho macOS, Hyper-V cho Windows và KVM cho Linux. Ngoài ra, 2 hypervisor phổ biến khác là VirtualBox và VMWare.

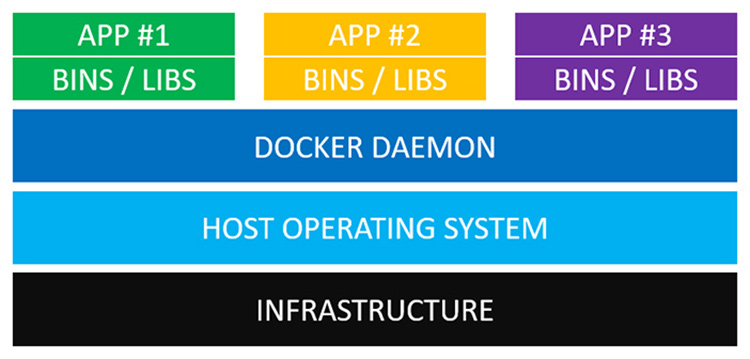
**Guest OS (hệ điều hành của máy ảo):**với mỗi máy ảo được tạo ra, người quản trị cần phải cài đặt một hệ điều hành đi kèm cho máy ảo đó để cài đặt và triển khai các service cần thiết. Ví dụ bạn cần triển khai 3 service trên 3 máy ảo khác nhau và mỗi “Guest OS” sẽ tiêu tốn ít nhất 700 MB dung lượng ổ cứng, vậy sẽ tiêu tốn khoảng 2.1 GB dung lượng ổ cứng của server để tạo 3 máy ảo để triển khai 3 service khác nhau. Chưa kể để các “Guest OS” này sẽ sử dụng một lượng tài nguyên khác của hệ thống như CPU, RAM.

**Bins / Libs (các tập tin / thư viện cần thiết):**các service / application sẽ cần phải có các gói tập tin, thư viện đi kèm để có thể hoạt động.

**App (application – service):** mã nguồn của các ứng dụng, phần mềm.

**Kiến trúc của service hoạt động trên Docker container**

Nếu so sánh với kiến trúc của service khi hoạt động trên nền tảng máy ảo, service hoạt động trên Docker container sẽ loại bỏ lớp “Guest OS”, với tốc độ khởi tạo service nhanh hơn hẳn so với việc sử dụng máy ảo. Docker Container sẽ giảm thiểu và đơn giản hóa các bản cập nhật bảo mật. Kiến trúc của service khi hoạt động trên nền tảng Docker container thông qua ảnh minh họa bên dưới:



Hình 1.5 Kiến trúc của service trên Docker container

**Infrastructure (hạ tầng):**có thể là laptop, một server chuyên dụng (dedicated server) hoat động trong một trung tâm dữ liệu (Data Center) hoặc một server riêng ảo (Virtual Private Server – VPS) trên cloud như DigitalOcean, Amazon, GCP, …

**Host Operating System (hệ điều hành của server):**hệ điều hành đang sử dụng cho server đang sử dụng, có thể là macOS, Windows hoặc Linux.

**Docker daemon (còn gọi là Docker Engine):**đây là service hoạt động trên server, được dùng để quản lý các thành phần cần thiết để khởi tạo và tương tác với Docker container.

**Bins / Libs (các tập tin / thư viện cần thiết):**các gói tập tin, thư viện đi kèm của service được thêm vào Docker image.

**App (application – service):** mã nguồn của các ứng dụng, phần mềm được thêm vào Docker container.

**Sự khác nhau giữa máy ảo và Docker Container**

Docker daemon có thể giao tiếp trực tiếp với hệ điều hành của server và phân bổ tài nguyên cho các Docker container đang chạy, đảm bảo mỗi container hoạt động độc lập với các container khác và hệ điều hành của server. Thay vì phải đợi một phút để máy ảo khởi động, người dùng có thể khởi động Docker container chỉ trong vài mili giây và tiết kiệm được rất nhiều dung lượng ổ đĩa và các tài nguyên hệ thống khác do không cần phải sử dụng “guest OS” cồng kềnh cho mỗi ứng dụng. Người dùng sẽ không cần ảo hóa vì Docker chạy trực tiếp trên hệ điều hành của server.

|  |  |
| --- | --- |
| **Máy ảo** | **Docker container** |
| Kích thước (dung lượng) lớn. | Kích thước (dung lượng) nhỏ. |
| Hiệu suất hạn chế. | Hiệu suất gốc (native). |
| Mỗi máy ảo sẽ có một hệ điều hành riêng. | Container sẽ sử dụng hệ điều hành của host. |
| Ảo hóa về mặt phần cứng | Ảo hóa về mặt hệ điều hành |
| Thời gian khởi động tính theo phút | Thời gian khởi động tính theo mili giây |
| Phân bổ bộ nhớ theo nhu cầu cần thiết | Yêu cầu ít dung lượng bộ nhớ hơn |
| Hoàn toàn bị cô lập và an toàn hơn | Cô lập ở mức tiến trình, có thể kém an toàn hơn |

Việc sử dụng máy ảo hay Docker container sẽ phụ thuộc vào nhu cầu sử dụng của người dùng. Máy ảo rất phù hợp trong việc cách ly tài nguyên hệ thống và toàn bộ môi trường làm việc. Đây sẽ là lựa chọn tốt hơn để chạy các ứng dụng yêu cầu tất cả tài nguyên và chức năng của hệ điều hành khi bạn cần chạy nhiều ứng dụng trên server hoặc có nhiều hệ điều hành khác nhau để quản lý. Ví dụ: công ty của bạn cung cấp dịch vụ web hosting, bạn có thể sẽ sử dụng máy ảo để phân phối tài nguyên của server công ty cho từng khách hàng.

Mặt khác, triết lý của Docker là cô lập các ứng dụng riêng lẻ, không phải toàn bộ hệ thống. Một ví dụ hoàn hảo về điều này sẽ là chia nhỏ một loạt các dịch vụ ứng dụng web thành các Docker image của riêng chúng và triển khai chúng bằng Docker Container. Docker Container là lựa chọn tốt hơn khi ưu tiên lớn nhất của bạn là tối đa hóa số lượng ứng dụng đang chạy trên một số lượng server tối thiểu. [4]

### Kiến trúc của Docker và các thành phần trong nó

#### Docker engine

Docker Engine là phần cốt lõi của Docker, được xây dựng theo kiểu kiến trúc client-server và được cài đặt trên máy Host. Docker Engine có 3 thành phần:

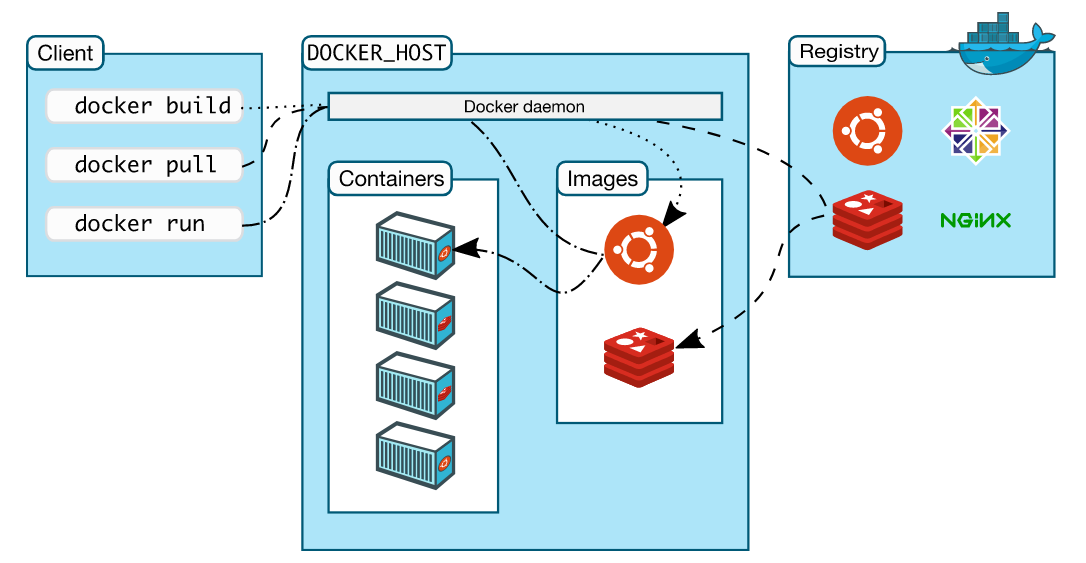
* + - Một tiến trình Docker daemon (dockerd) có nhiệm vụ khởi tạo và quản lý các container, image, volume, network.
    - REST API là nơi nhận các yêu cầu từ phía Docker client và giao tiếp với Docker daemon để yêu cầu Docker daemon thực thi các yêu cầu từ phía người dùng.
    - Docker CLI có vai trò là client, cung cấp giao diện tương tác với người dùng (command line) và gửi các request tương ứng đến Docker Daemon thông qua REST API.



Hình 1.6 Các thành phần của docker engine

#### 1.2.3.2 Kiến trúc của Docker

Docker sử dụng kiến trúc client – server. Docker client sử dụng một REST API (thông qua UNIX socket, hoặc cổng mạng) để có thể giao tiếp với Docker daemon, tiến trình này thực hiện công việc tạo, chạy và phân phối các Docker container. Docker client và daemon có thể chạy trên cùng một hệ thống hoặc có thể kết nối Docker client với Docker daemon từ xa. Kiến trúc của Docker sẽ bao gồm:



Hình 1.7 Kiến trúc của Docker

***Docker daemon***

Daemon Docker (dockerd) lắng nghe các yêu cầu của người dùng thông qua Docker API và quản lý các đối tượng Docker như image, container, network và volume. Một daemon cũng có thể giao tiếp với các daemon khác để quản lý các Docker service.

***Docker client***

Docker client (docker) là cách thức mà nhiều người dùng tương tác với Docker. Khi bạn sử dụng các câu lệnh như docker run, client thông qua Docker API sẽ gửi các lệnh này đến dockerd, nơi thực hiện chúng. Docker client có thể giao tiếp với nhiều hơn một Docker daemon.

***Docker registry***

Docker registry sẽ là nơi lưu trữ các Docker image. Docker Hub là nơi lưu trữ Docker image công khai (public registry) mà bất kỳ ai cũng có thể sử dụng và Docker được định cấu hình mặc định để tìm image trên Docker Hub. Ngoài ra người dùng có thể cấu hình các registry riêng tư khác để lưu trữ Docker image.

Khi người dùng sử dụng câu lệnh “docker pull” hoặc “docker run”, các image chỉ định sẽ được tải về dựa trên registry đã được cấu hình trước đó. Khi người dùng sử dụng câu lệnh “docker push”, image cũng sẽ được tải lên registry mà người dùng đã cấu hình từ trước đó.

***Các đối tượng khác***

Khi sử dụng Docker, bạn sẽ khởi tạo và sử dụng image, container, network, volume, plugin và các đối tượng khác. Sau đây sẽ là tổng quan về các đối tượng này.

**Docker image**

Docker Image là template read-only (chỉ cho phép đọc) với các hướng dẫn để tạo Docker container. Image sẽ được sử dụng để đóng gói các ứng dụng và các thành phần đi kèm của ứng dụng, được lưu trữ ở server hoặc trên registry. Ví dụ bạn có thể sử dụng Dockerfile để tạo ra một Docker image sử dụng hệ điều hành Ubuntu và cài đặt Apache server với những cài đặt, cấu hình tùy chỉnh của riêng mình.

**Docker container**

Container là 1 “runable instance” của image. Bạn có thể khởi tạo, dừng hoặc xóa container bằng cách sử dụng Docker API hoặc CLI. Bạn có thể kết nối container đến 1 hoặc nhiều network, thư mục lưu trữ, hoặc thậm chí tạo ra 1 image mới dựa trên tình trạng hiện tại của container. Mặc định 1 container được “cách ly” với các container và server nếu người dùng không có các cài đặt gì thêm.

**Docker volume**

Volume được thiết kể để làm nơi lưu trữ các dữ liệu độc lập với vòng đời của container.

**Docker network**

Cung cấp một private network mà chỉ tồn tại giữa container và server, giúp các container có thể giao tiếp được với nhau một cách dễ dàng.

**Docker service**

Service cho phép bạn mở rộng các container thông qua nhiều Docker daemon, chúng giao tiếp với nhau thông qua swarm cluster bao gồm nhiều manager và worker. Mỗi một node của swarm là 1 Docker daemon giao tiếp với nhau bằng cách sử dụng Docker API. Theo mặc định thì service được cân bằng tải trên các node.

#### 1.2.3.3 Cài đặt Docker

Hiện giờ Docker có đầy đủ bản cài đặt trên Windows, MacOS, Linux. Tùy thuộc việc chúng ta muốn chạy trên môi trường nào thì tải về bản cài đặt tương ứng để cài đặt.

**Cài Docker trên macOS**

Tải bộ cài tại <https://hub.docker.com/editions/community/docker-ce-desktop-mac>, cài đặt đơn giản như các công cụ thông thường.

**Cài Docker trên Windows 10**

Đảm bảo hệ thống là Windows 10 Pro

Tải bộ cài tại <https://hub.docker.com/editions/community/docker-ce-desktop-windows>, tiến hành cài đặt. Đối với Windows phải kích hoạt chế độ Hyper-V virtualization.

Chạy lệnh PowerShell sau để kích hoạt:

Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All

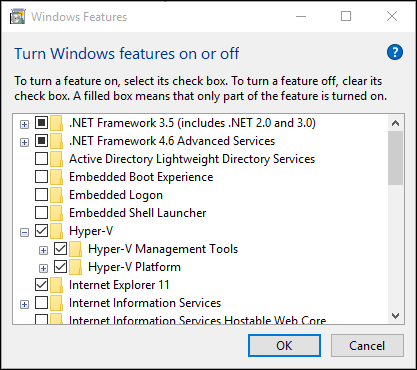
**Kích hoạt thông qua thiết lập Windows**

Nhấn phải chuột vào biểu tượng cửa sổ, chọn **Apps and Features**

Chọn **Programs and Features**

Chọn **Turn Windows Features on or off**

Đánh dấu vào **Hyper-V** như hình dưới



**Cài Docker trên Ubuntu**

Chạy các lệnh để cài đặt:

sudo apt update

sudo apt install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

sudo apt update

apt-cache policy docker-ce

sudo apt install docker-ce

sudo systemctl status docker

Sau khi cài đặt, chúng ta có thể cho user hiện tại thuộc group docker, để khi gõ lệnh không cần xin quyền sudo

sudo usermod -aG docker $USER

Logout sau đó login lại để có hiệu lực.

Ngoài ra khi sử dụng đến thành phần docker-compose thì cài thêm sudo apt install docker-compose

**Cài Docker trên CentOS7/RHEL7**

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

sudo yum install docker-ce

sudo usermod -aG docker $(whoami)

sudo systemctl enable docker.service

sudo systemctl start docker.service

#Cài thêm Docker Compose

sudo yum install epel-release

sudo yum install -y python-pip

sudo pip install docker-compose

sudo yum upgrade python\*

docker-compose version

**Cài Docker trên CentOS8/RHEL8**

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

sudo yum install docker-ce

# Nếu lỗi, chạy lệnh sau

sudo dnf install --nobest docker-ce -y

sudo usermod -aG docker $(whoami)

sudo systemctl enable docker.service

sudo systemctl start docker.service

#Cài thêm Docker Compose

sudo yum install epel-release

sudo yum -y install python2-pip

sudo pip2 install docker-compose

sudo yum upgrade python\*

docker-compose version

Khi đã có **Docker** trên máy, làm việc với Docker để quan lý các thành phần của nó ... hầu hết làm việc qua giao diện dòng lệnh CLI của hệ thống: trên macOS/Linux mở termial để gõ lệnh, trên Windows thì dùng PS (PowerShell) hoặc cmd để chạy các lệnh Docker.

**Làm quen ban đầu Docker**

Giờ hãy mở giao diện lệnh termial (macOS, Linux) hay PowerShell của Windows nên và gõ:

**Kiểm tra phiên bản Docker**

docker --version

Hoặc lệnh thông tin chi tiết hơn:

docker info



**Image và container trong Docker**

Image là một gói phần mềm trong đó chứa những thứ cần như thư viện, các file cấu hình, biến môi trường để chạy mội ứng dụng nào đó. Nó giống như cái USB chứa bộ cài đặt hệ điều hành Windows!

Khi một phiên bản của image chạy, phiên bản chạy đó gọi là container - (vậy muốn có container phải có image). Bất ký lúc nào chúng ta cũng có thể kiểm tra xem có bao nhiêu container đang chạy và nó sinh ra từ image nào.

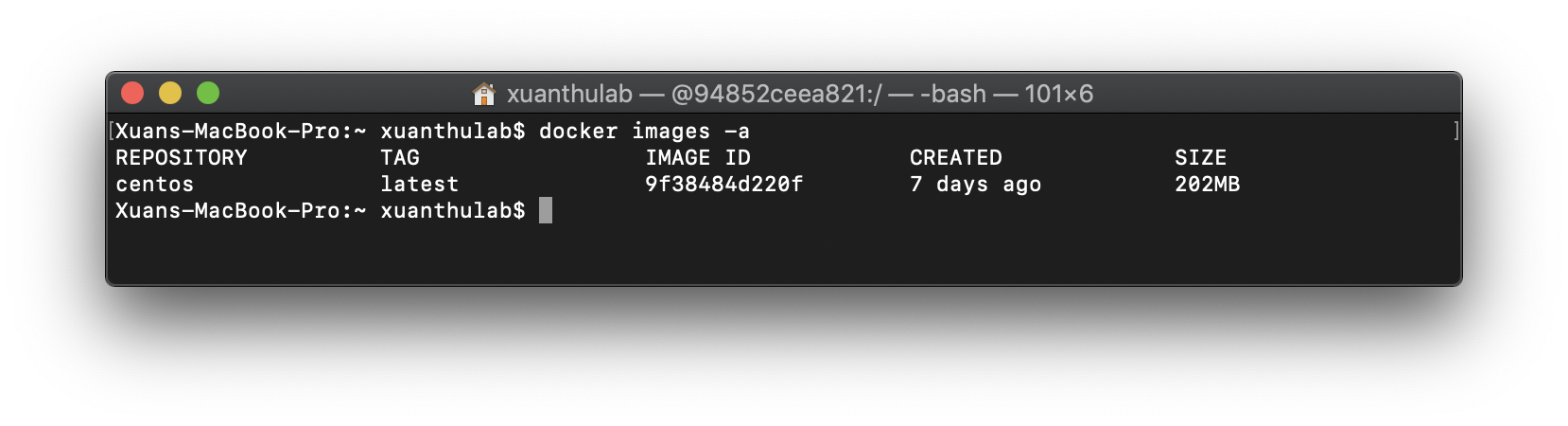
Bước đầu, để có image nào đó chúng ta tải về từ https://hub.docker.com/search?q=&type=image, tại đó có đủ các loại phù hợp với công việc!

**Tải về một image**

Kiểm tra các image đang có

docker images -a

Nó sẽ liệt kê ra các image đang có, với dạng:



Repository tên image trên kho chứa, ví dụ hệ điều hành CentOS có tên trên hub.docker.com là centos (xem [centos](https://hub.docker.com/_/centos))

TAG là phiên bản image, với giá trị latest có nghĩa là bản cuối. Muốn tải về bản khác latest vào mục TAGS trên hub.docker.com tìm bản phù hợp.

IMAGE ID một chuỗi định danh duy nhất (tên) của image trên hệ thống.

Để tải về một image nào đó, dùng lệnh

docker pull nameimage:tag

#Hoặc tải về bản cuối

docker pull nameimage

Bây giờ hệ thống không có hệ điều hành Ubuntu, muốn tải nó về - vì trên hub.docker.com nó có tên ubuntu, ta sẽ tải về bản mới nhất bằng lệnh:

docker pull ubuntu



Giờ thì hệ thống sẽ có thêm image với tên ubutu, có thể góc lại lệnh kiểm tra docker images -a ở trên để xem.

**Tạo và chạy container**

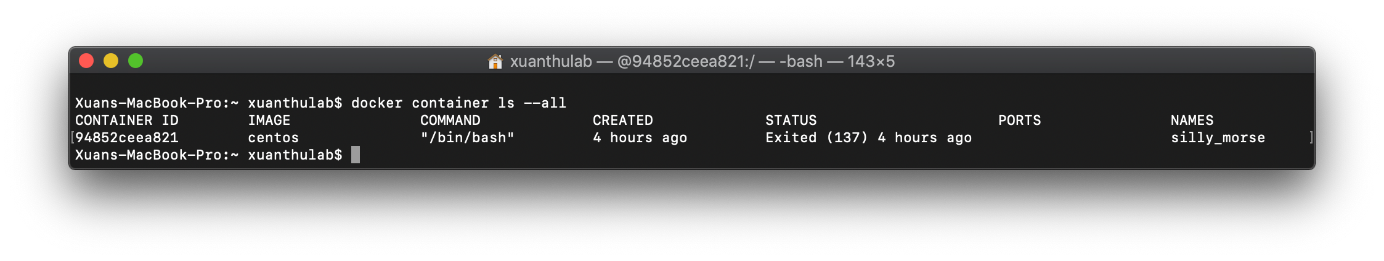
Như trên, container là một phiên bản chạy của image, trước tiên.

Kiểm tra có các container nào đang chạy:

docker ps

Liệt kê tất cả các container:

docker container ls --all



Các container đã tạo, liệt kê ra hãy chú ý mấy thứ

CONTAINER ID một con số (mã hash) gán cho container, dùng mã này để quản lý container này, như xóa bỏ, khởi động, dừng lại ...

IMAGE cho biết container sinh ra từ image nào.

COMMAND cho biết lệnh, ứng dụng chạy khi container chạy (/bin/bash là terminate)

STATUS cho biết trạng thái, (exit - đang dừng)

Để tạo và chạy một container theo cú pháp:

docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Các tham số đó là:

[OPTIONS] các thiết lập khi tạo container, có rất nhiều thiết lập tùy mục đích tạo container, sẽ tìm hiểu các thiết lập qua từng trường hợp cụ thể.

IMAGE tên image hoặc ID của image từ nó sinh ra container.

[COMMAND] [ARG...] lệnh và tham số khi container chạy.

Trở lại phần trên, đang có image hệ điều hành Ubuntu, giờ muốn chạy nó (tạo một container từ nó), hãy gõ lệnh sau:

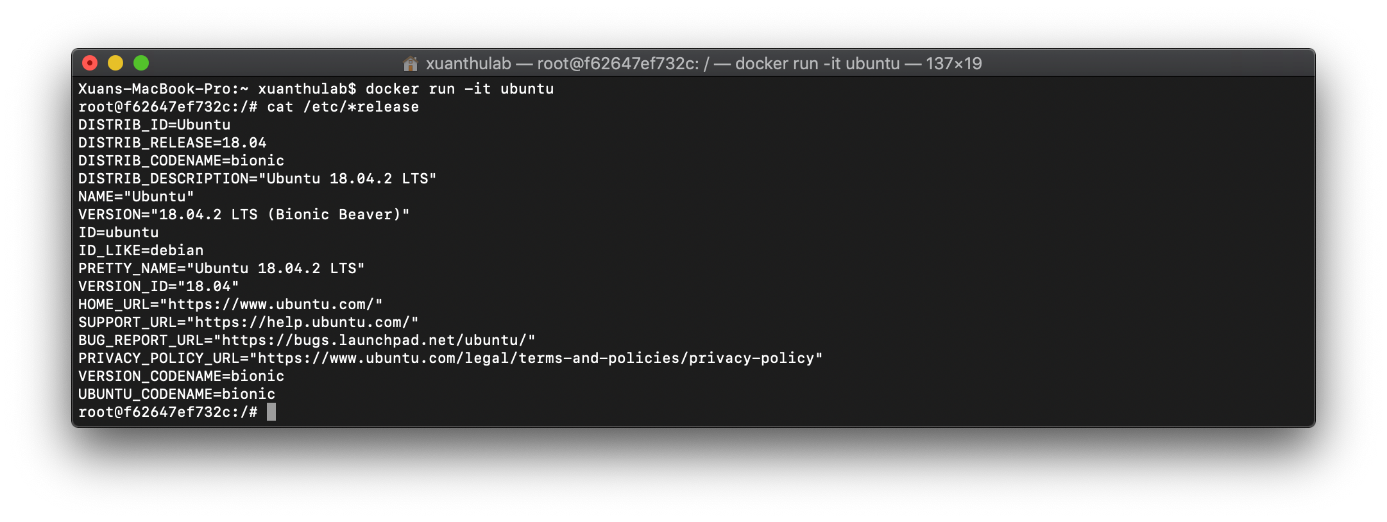
docker run -it ubuntu

Chú ý có tham số -it để khi container chạy, chúng ta có terminate làm việc ngay với Ubuntu. Tham số này có nghĩa là

-t nó có nghĩa là console, cho phép kết nối với terminal để tương tác

-i có nghĩa duy trì mở stdin để nhập lệnh.

Sau lệnh này, nếu đang chạy Ubuntu, có thể gõ vài lệnh Ubuntu kiểm tra xem. Ví dụ kiểm tra phiên bản Ubuntu, xem hình dưới với lệnh cat /ect/\*release. Như vậy chúng ta đã tạo một container



Nếu đang ở terminal của Ubuntu, có thể kết thúc bằng lệnh exit, hoặc nhấn nhanh CTRL+P,CTRL+Q để thoat termial nhưng container vẫn đang chạy.

**Một vài tham số khác:**

Tạo và chạy container, container tự xóa khi kết thúc thì thêm vào tham số --rm

docker run -it --rm ubuntu

Tạo và chạy container - khi container chạy thi hành ngay một lệnh nào đó, ví dụ ls -la

docker run -it --rm debian ls -la

Tạo và chạy container - ánh xạ một thự mục máy host vào một thư mục container, chia sẻ dữ liệu

docker run -it --rm -v path-in-host:path-in-container debian

**Vào container đang chạy**

Kiểm tra xem bằng lệnh docker ps, nếu có một container với ID là containerid đang chạy, để quay trở lại terminal của nó dùng lệnh:

docker container attach containerid

Chạy một container đang dừng

docker container start -i containerid

Nếu cần xóa bỏ hẳn một container thì dùng lệnh

docker container rm containerid.

# CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG WEBSITE OHOTOUR

## Yêu cầu và đối tượng sử dụng

### Quản lý yêu cầu

#### 2.1.1.1 Yêu cầu tiền đề (yêu cầu kĩ thuật với người phát triển hệ thống, ngôn ngữ phát triển)

* Kỹ thuật lập trình tốt, cách lập trình dễ đọc phục vụ việc nâng cấp trong tương lai.
* Môi trường phát triển: ngôn ngữ Javascript (framework React JS và Node JS), Ant design, cơ sở dữ liệu mySQL Phát triển trên nền WEB.
* Triển khai bằng Docker với mô hình MicroService.

#### 2.1.1.2 Yêu cầu giao diện

* Sử dụng màu sắc hài hòa, chuyên nghiệp, màu chủ đạo là màu đen và màu trắng.
* Giao diện thân thiện với người dùng.
* Giao diện một ngôn ngữ (tiếng Việt).

#### 2.1.1.3 Yêu cầu chức năng

Hệ thống được xây dựng là một trang web quản lí bán hàng thức ăn chăn nuôi, với các chức năng cơ bản:

* Chức năng đăng nhập:

Chức năng này dành cho quản trị viên sử dụng để đăng nhập vào tài khoản trên web. Người dùng sẽ được cung cấp một tài khoản đăng nhập vào hệ thống.

* Chức năng đăng xuất:

Chức năng này dành cho quản trị viên sử dụng để đăng xuất khỏi hệ thống. Người dùng sẽ được cung cấp một tài khoản admin và đăng nhập vào hệ thống quản trị, sau đó người dùng có thể đăng xuất khỏi tài khoản quản trị này.

* Chức năng quản lí danh mục sản phẩm:

Chức năng này dành cho quản trị viên sử dụng cho phép xem danh sách và tìm kiếm danh mục sản phẩm, thêm các danh mục mới, cập nhật lại thông tin các danh mục đã có và xoá danh mục sản phẩm trên web.

* Chức năng quản lí sản phẩm:

Chức năng này dành cho quản trị viên sử dụng cho phép xem danh sách và tìm kiếm sản phẩm, thêm các sản phẩm mới, cập nhật lại thông tin các sản phẩm đã có và xoá sản phẩm trên web.

* Chức năng quản lí kho hàng:

Chức năng này dành cho quản trị viên sử dụng cho phép xem danh sách và tìm kiếm các kho hàng, thêm các kho mới, cập nhật lại thông tin các kho đã có và xoá kho hàng trên web.

* Chức năng quản lí khách hàng:

Chức năng này dành cho quản trị viên sử dụng cho phép xem danh sách và tìm kiếm các khách hàng, thêm khách hàng mới, cập nhật lại thông tin khách hàng đã có và xoá khách hàng trên web.

* Chức năng quản lí nhà cung cấp:

Chức năng này dành cho quản trị viên sử dụng cho phép xem danh sách và tìm kiếm các nhà cung cấp, thêm nhà cung cấp mới, cập nhật lại thông tin nhà cung cấp đã có và xoá nhà cung cấp trên web.

* Chức năng quản lí nhập hàng:

Chức năng này dành cho quản trị viên sử dụng cho phép xem lại danh sách và tìm kiếm các phiếu nhập, tạo phiếu nhập mới, cập nhật lại thông tin và xoá phiếu nhập hàng trên web.

* Chức năng quản lí bán hàng:

Chức năng này dành cho quản trị viên sử dụng cho phép xem lại danh sách và tìm kiếm các phiếu bán hàng, tạo phiếu bán hàng mới, cập nhật lại thông tin và xoá phiếu bán hàng trên web.

#### 2.1.1.4 Yêu cầu phi chức năng

* Tính bảo mật, độ tin cậy, khả năng chịu lỗi, hiệu năng.
* Dữ liệu được sao lưu thường xuyên.
* Có khả năng nâng cấp hệ thống.

### Đối tượng sử dụng

* Hệ thống chỉ gồm một tác nhân là người quản trị hệ thống (admin) quản lí tất cả các phần: quản lý danh mục sản phẩm, sản phẩm, kho hàng, xem lại danh sách phiếu nhập phiếu bán và thêm sửa xóa các phiếu, quản lí khách hàng và nhà cung cấp.

## Biểu đồ tuần tự

### Usecase đăng nhập

* Đặc tả UseCase:

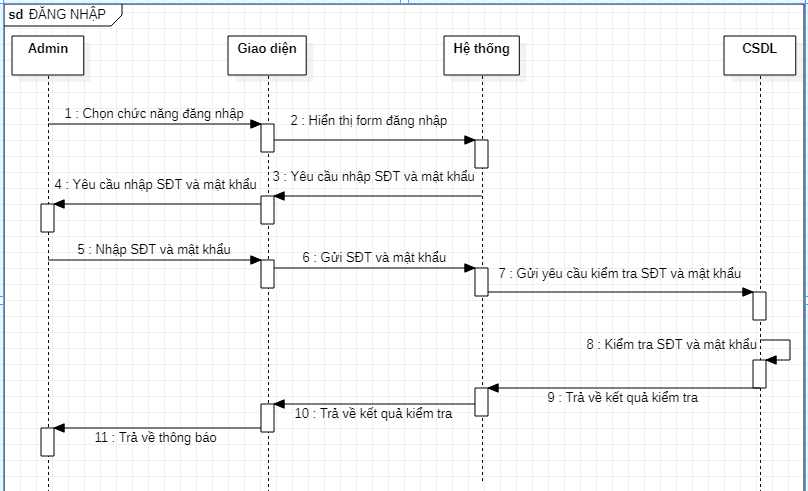
**-** Mô tả: Cho phép người dùng có thể truy cập vào website của hệ thống thông qua username và password.

**-** Actor: Admin.

**-** Điều kiện trước: Truy cập vào website dành cho quản trị và thiết bị sử dụng có kết nối internet với server.

**-** Luồng chính: Hiển thị trang tổng quan website với quyền người dùng admin.

* Biểu đồ tuần tự:



Hình 2. 1 Biểu đồ tuần tự luồng đăng nhập

### Usecase đăng xuất

#### Đặc tả UseCase:

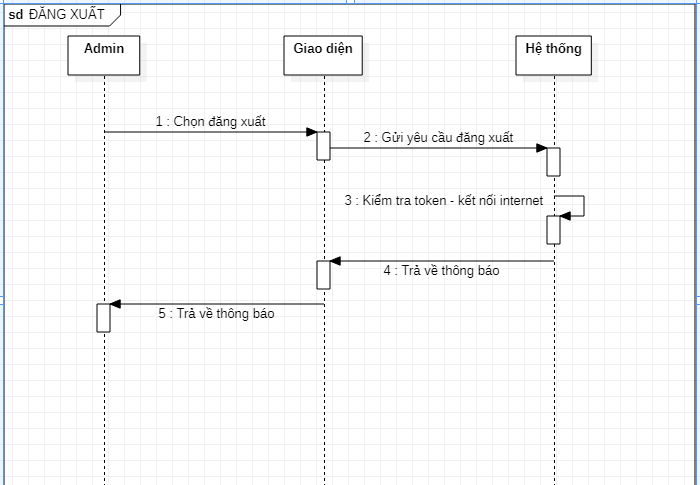
**-** Mô tả: Cho phép người dùng có thể rời khỏi website của hệ thống.

**-** Actor: Admin.

**-** Điều kiện trước: Truy cập vào website dành cho quản trị và thiết bị sử dụng có kết nối internet với server. Người dùng đã đăng nhập vào hệ thống.

**-** Luồng chính: Hiển thị thông báo đăng xuất.

* Biểu đồ tuần tự:



Hình 2. 2 Biểu đồ tuần tự luồng đăng xuất

### UseCase Quản lí danh mục sản phẩm

#### Đặc tả UseCase:

* Mô tả: Cho phép người dùng thực hiện thêm, sửa, xóa, xem danh sách danh mục sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.

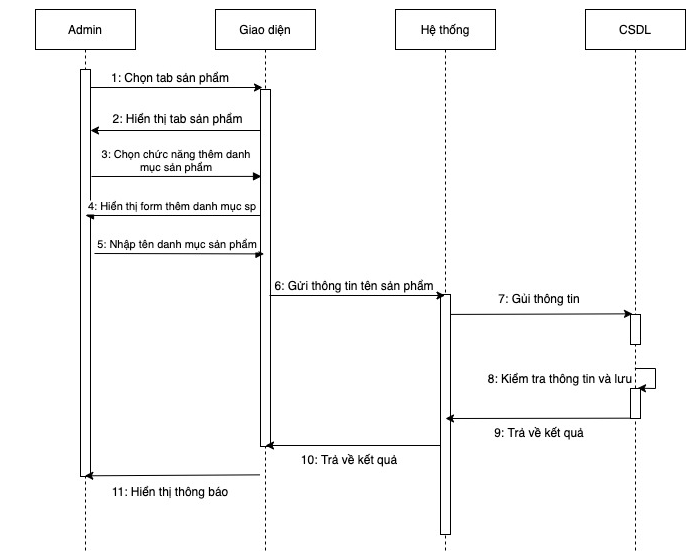
#### Biểu đồ UseCase:



Hình 2. 3 Biểu đồ Usecase quản lý danh mục sản phẩm

#### 2.3.3.1 UseCase thêm danh mục sản phẩm

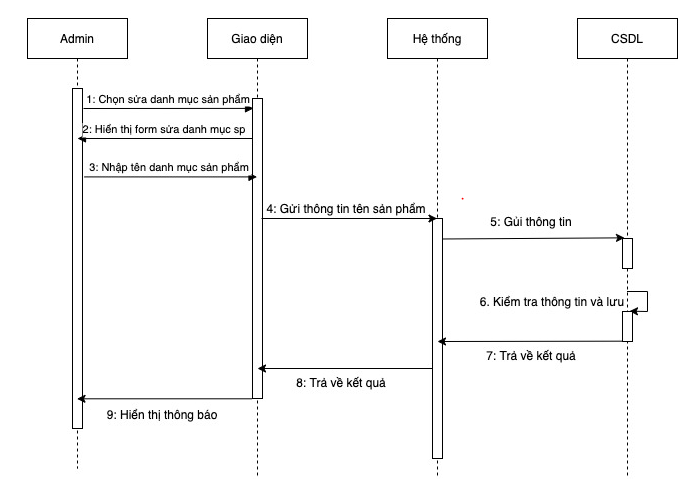
* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể thêm danh mục sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form thêm mới danh mục sản phẩm (tên danh mục sản phẩm).
* Biểu đồ tuần tự:



Hình 2. 4 Biểu đồ tuần tự thêm danh mục sản phẩm

#### 2.3.3.2 Usecase sửa danh mục sản phẩm

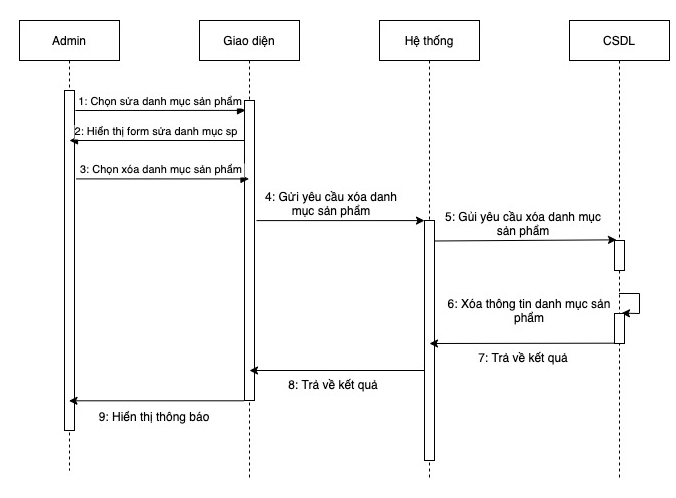
* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể sửa các danh mục sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form sửa danh mục sản phẩm
* Biểu đồ tuần tự:



Hình 2. 5 Biểu đồ tuần tự sửa danh mục sản phẩm

#### 2.3.3.3 UseCase Xóa danh mục sản phẩm

* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể xóa các danh mục sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Biểu đồ tuần tự:



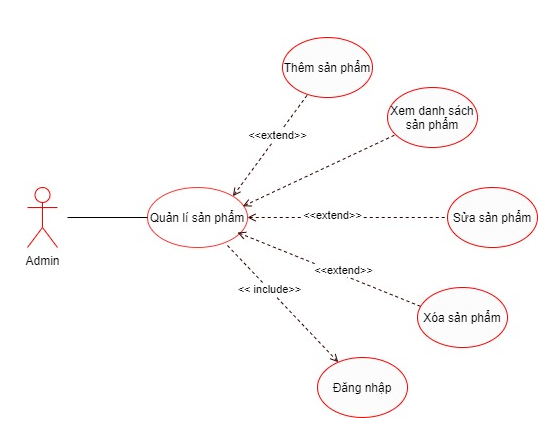
Hình 2. 6 Biểu đồ tuần tự xóa danh mục sản phẩm

### UseCase Quản lí sản phẩm

#### Đặc tả UseCase:

* Mô tả: Cho phép người dùng thực hiện thêm, sửa, xóa, xem danh sách các sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Ngoại lệ: Không có phản hồi về từ server: hiển thị ra thông báo lỗi không kết nối.

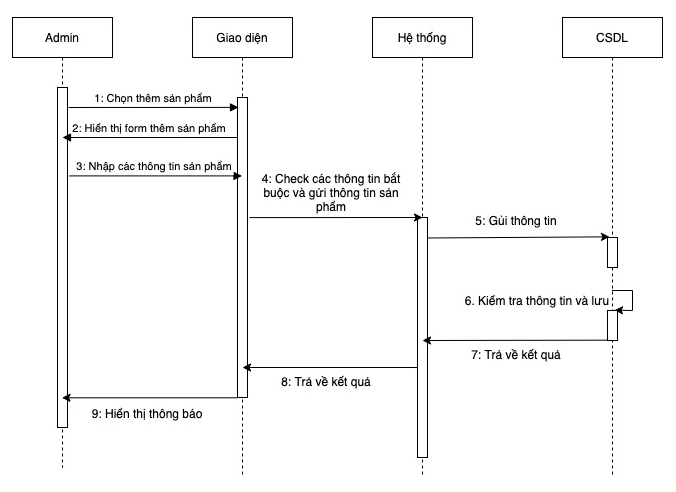
#### Biểu đồ UseCase:



Hình 2. 7 Biểu đồ Usecase quản lý sản phẩm

#### 2.3.4.1 UseCase thêm sản phẩm

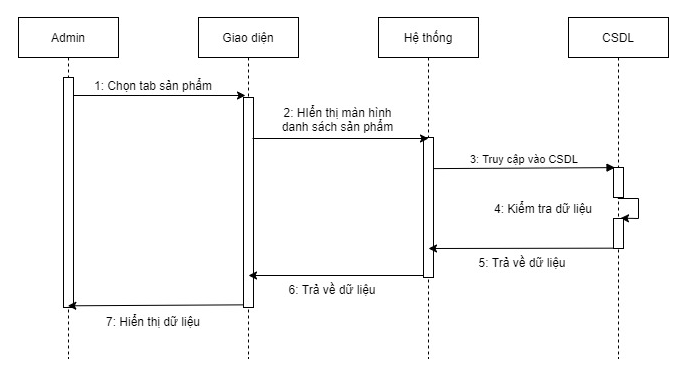
* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể thêm sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form thêm mới sản phẩm (mã sản phẩm, tên sản phẩm, đơn vị tính, nhóm mặt hàng, giá bán lẻ, giá sỉ, giá nhập, ảnh sản phẩm).
* Biểu đồ tuần tự:



Hình 2. 8 Biểu đồ tuần tự thêm sản phẩm

#### 2.3.4.2 UseCase xem danh sách sản phẩm

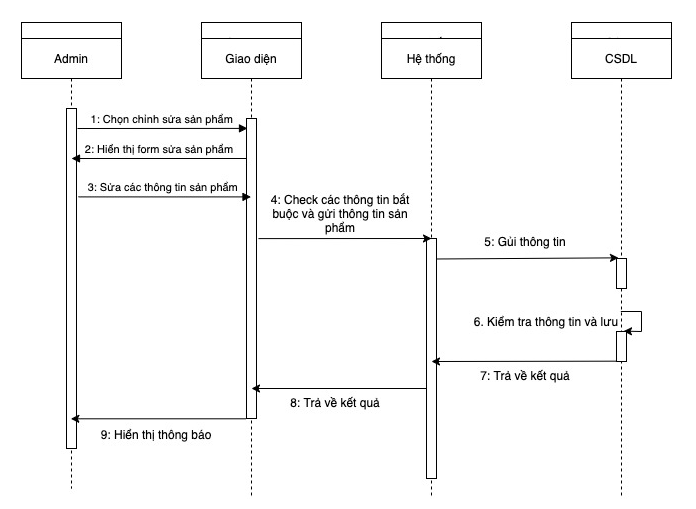
* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể xem danh sách sản phẩm của hệ thống.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Biểu đồ tuần tự:



Hình 2. 9 Biểu đồ tuần tự xem danh sách sản phẩm

#### 2.3.4.3 Usecase sửa sản phẩm

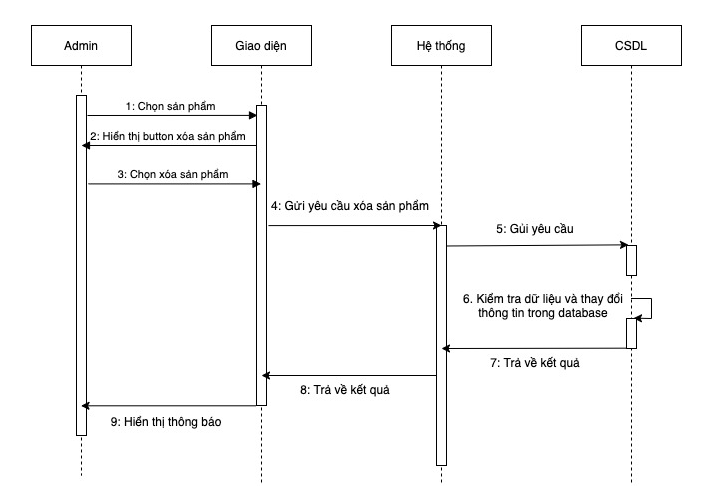
* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể sửa thông tin sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form sửa sản phẩm
* Biểu đồ tuần tự:



Hình 2. 10 Biểu đồ tuần tự sửa sản phẩm

#### 2.3.4.4 UseCase Xóa sản phẩm

* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể xóa các sản phẩm.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Biểu đồ tuần tự:



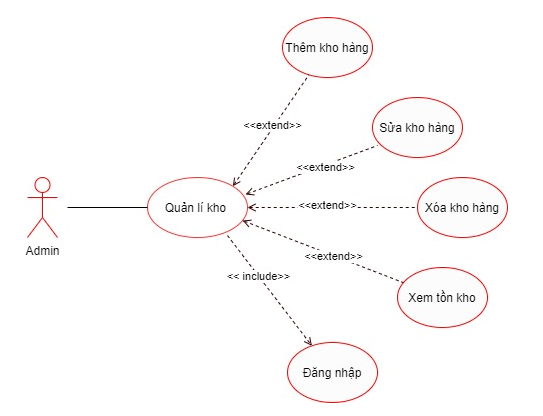
Hình 2. 11 Biểu đồ tuần tự xóa sản phẩm

### UseCase Quản lí kho

#### Đặc tả UseCase:

* Mô tả: Cho phép người dùng thực hiện thêm, sửa, xóa, xem danh sách các kho hàng.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.

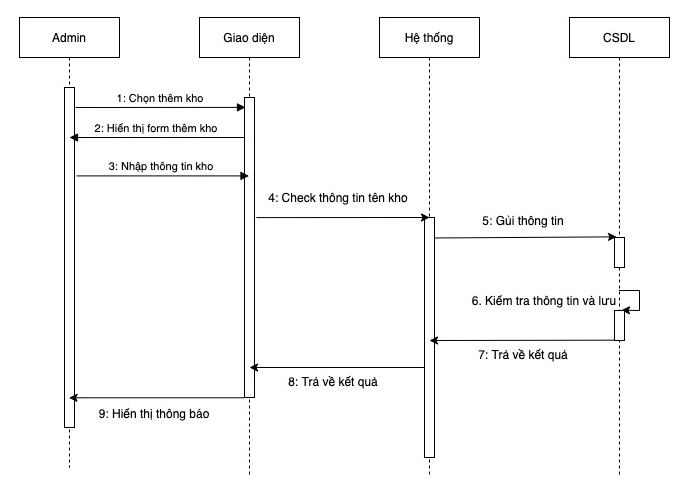
#### Biểu đồ UseCase:



Hình 2. 12 Biểu đồ Usecase quản lý kho hàng

#### 2.3.5.1 UseCase thêm kho hàng

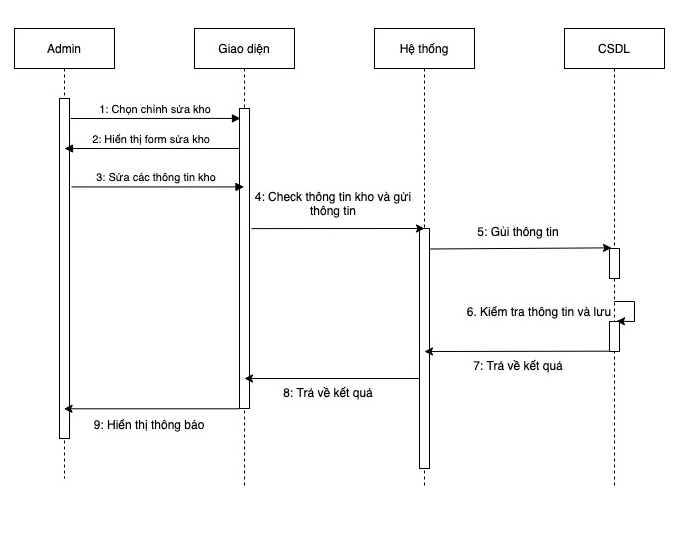
* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể thêm kho vào trong hệ thống.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form thêm mới kho (tên kho hàng).
* Biểu đồ tuần tự:



Hình 2. 13 Biểu đồ tuần tự thêm kho hàng

#### 2.3.5.2 Usecase sửa kho hàng

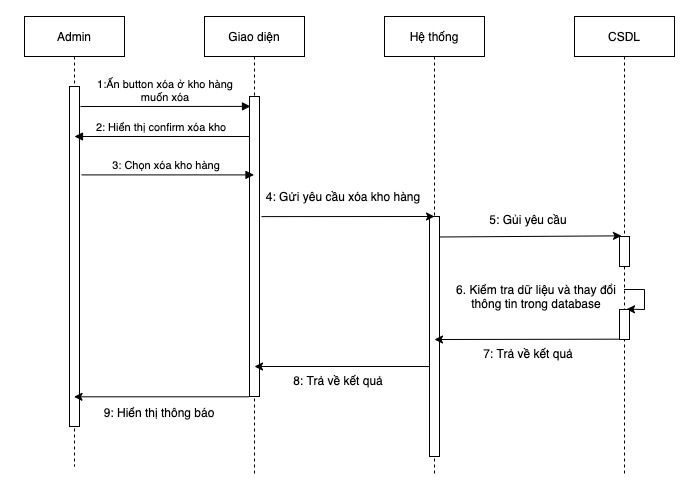
* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể sửa thông tin kho hàng.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form sửa kho hàng
* Biểu đồ tuần tự:



Hình 2. 14 Biểu đồ tuần tự sửa kho hàng

#### 2.3.5.3 UseCase Xóa kho hàng

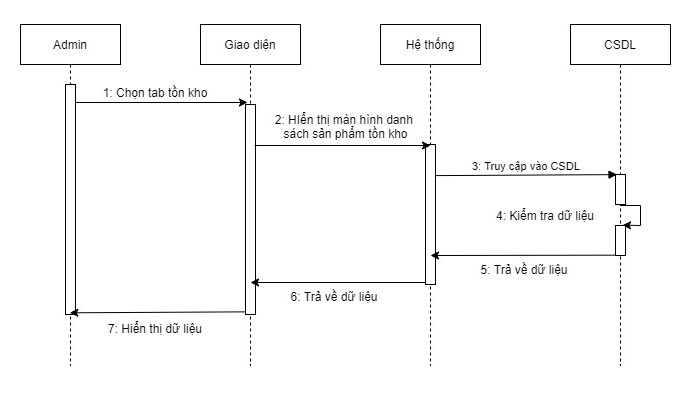
* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể xóa các kho hàng.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Biểu đồ tuần tự:



Hình 2. 15 Biểu đồ tuần tự xóa kho hàng

#### 2.3.5.4 UseCase xem danh sách tồn kho

* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể xem danh sách số lượng sản phẩm tồn kho của hệ thống.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Biểu đồ tuần tự:



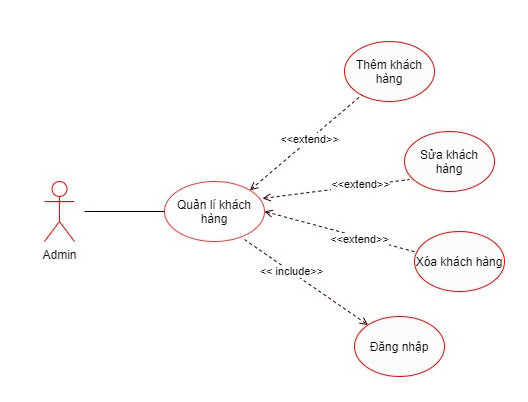
Hình 2. 16 Biểu đồ tuần tự xem danh sách sản phẩm tồn kho

### UseCase Quản lí khách hàng

#### Đặc tả UseCase:

* Mô tả: Cho phép người dùng thực hiện thêm, sửa, xóa và xem danh sách các khách hàng.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.

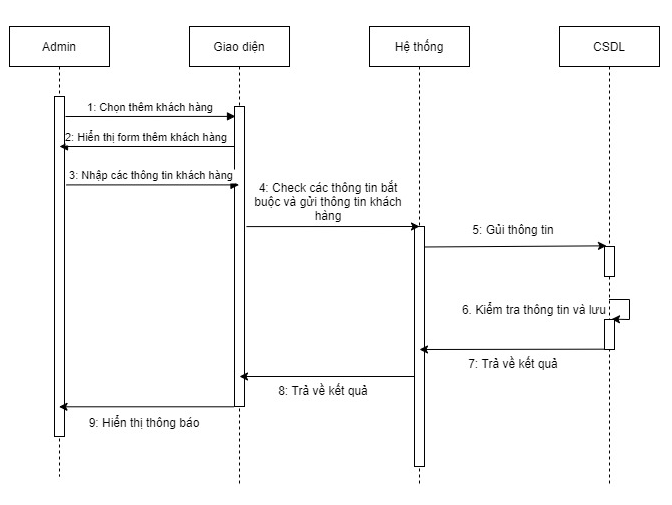
#### Biểu đồ UseCase:



Hình 2. 17 Biểu đồ Usecase quản lý khách hàng

#### 2.3.6.1 UseCase thêm khách hàng

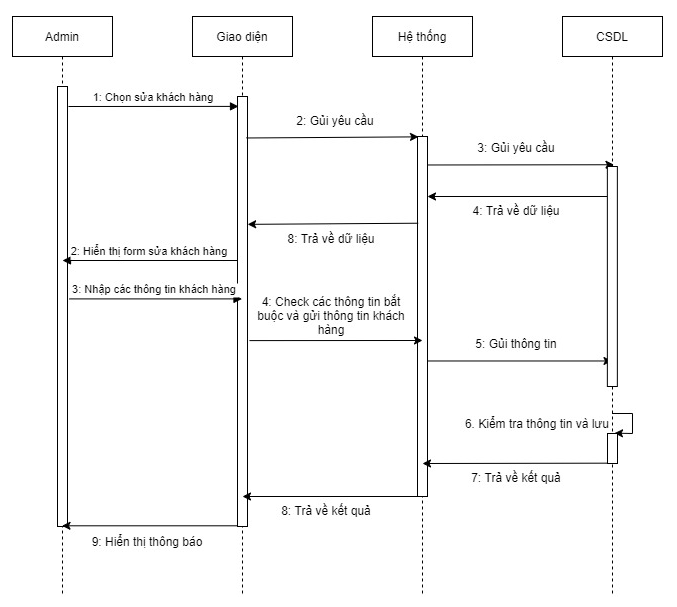
* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể thêm khách hàng vào trong hệ thống.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form thêm mới khách hàng (tên khách hàng, số điện thoại, tỉnh thành, email, ngày sinh, giới tính, địa chỉ, mã số thuế, người đại diện, chức vụ).
* Biểu đồ tuần tự:



Hình 2. 18 Biểu đồ tuần tự thêm khách hàng

#### 2.3.6.2 Usecase sửa khách hàng

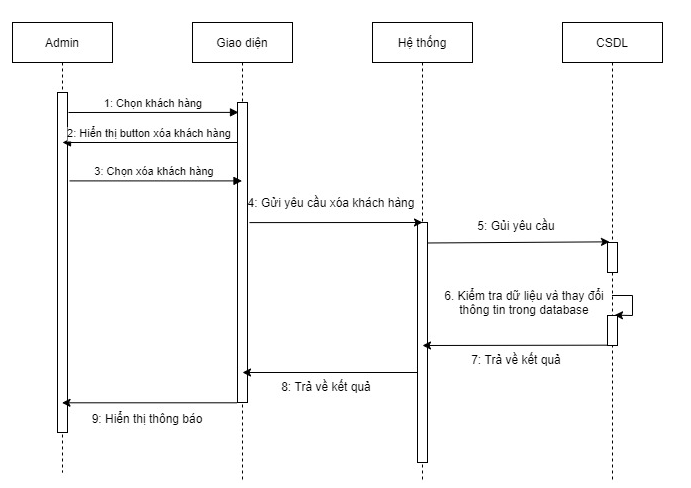
* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể sửa thông tin khách hàng.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form sửa thông tin khách hàng
* Biểu đồ tuần tự:



Hình 2. 19 Biểu đồ tuần tự sửa khách hàng

#### 2.3.6.3 UseCase Xóa khách hàng

* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể xóa các khách hàng.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Biểu đồ tuần tự:



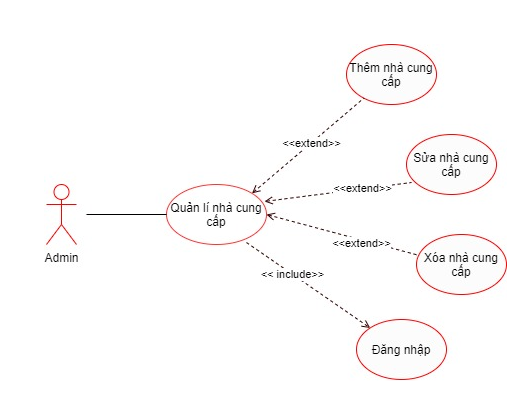
Hình 2. 20 Biểu đồ tuần tự xóa khách hàng

### UseCase Quản lí nhà cung cấp

#### Đặc tả UseCase:

* Mô tả: Cho phép người dùng thực hiện thêm, sửa, xóa và xem danh sách các nhà cung cấp.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.

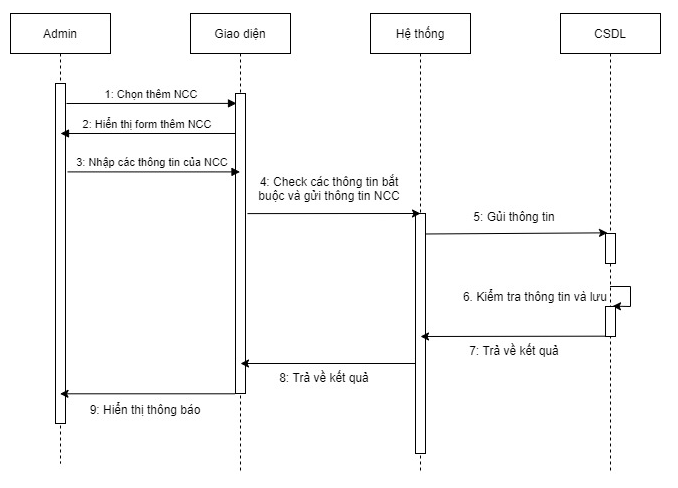
#### Biểu đồ UseCase:



Hình 2. 21 Biểu đồ Usecase quản lý nhà cung cấp

#### 2.3.7.1 UseCase thêm nhà cung cấp

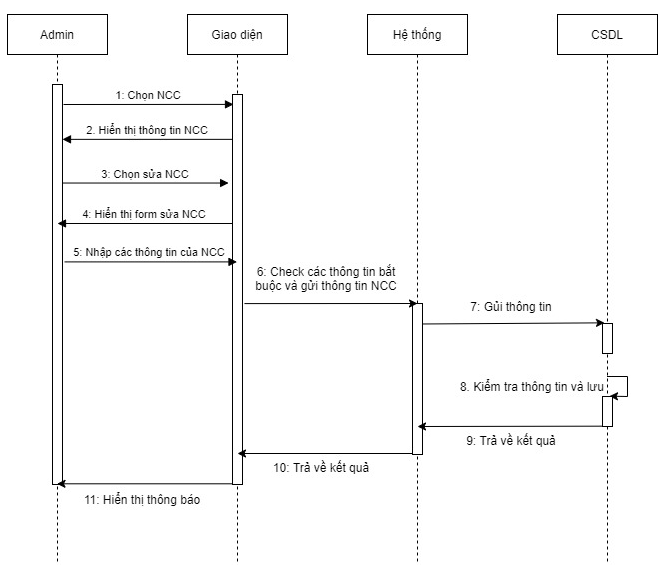
* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể thêm nhà cung cấp vào trong hệ thống.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form thêm mới nhà cung cấp (tên nhà cung cấp, số điện thoại, tỉnh thành, mã nhà cung cấp, email, địa chỉ, mã số thuế).
* Biểu đồ tuần tự:



Hình 2. 22 Biểu đồ tuần tự thêm nhà cung cấp

#### 2.3.7.2 Usecase sửa nhà cung cấp

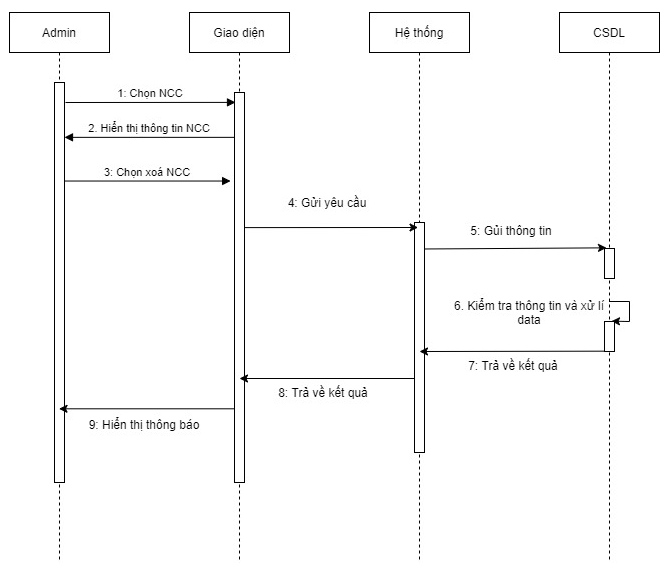
* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể sửa thông tin nhà cung cấp.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form sửa thông tin nhà cung cấp
* Biểu đồ tuần tự:



Hình 2. 23 Biểu đồ tuần tự sửa nhà cung cấp

#### 2.3.7.3 UseCase Xóa nhà cung cấp

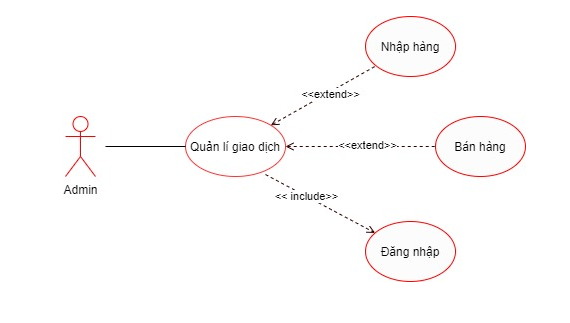
* Đặc tả UseCase:
* Mô tả: Cho phép quản trị viên có thể xóa các nhà cung cấp.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Biểu đồ tuần tự:



Hình 2. 24 Biểu đồ tuần tự xóa nhà cung cấp

### UseCase Giao dịch

#### Biểu đồ UseCase:



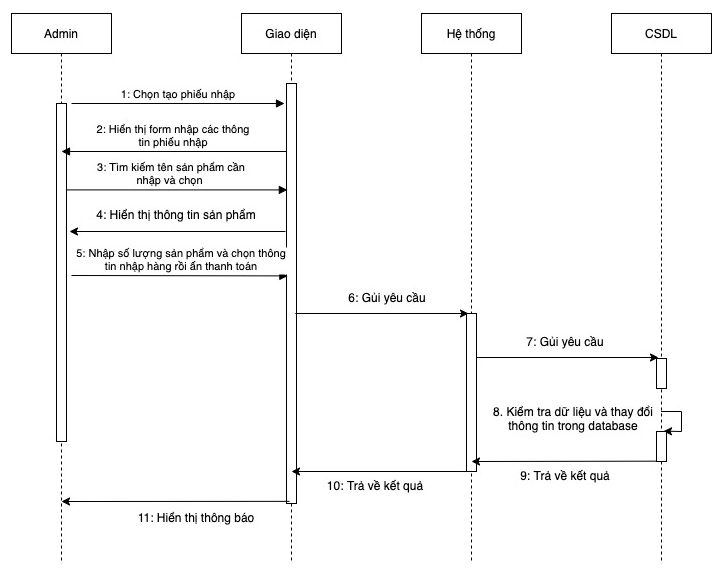
Hình 2. 25 Biểu đồ Usecase quản lý giao dịch

#### Đặc tả UseCase:

* Mô tả: Cho phép người dùng thực hiện các giao dịch nhập hàng và bán hàng vào hệ thống
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.

#### 2.3.8.1 UseCase nhập hàng

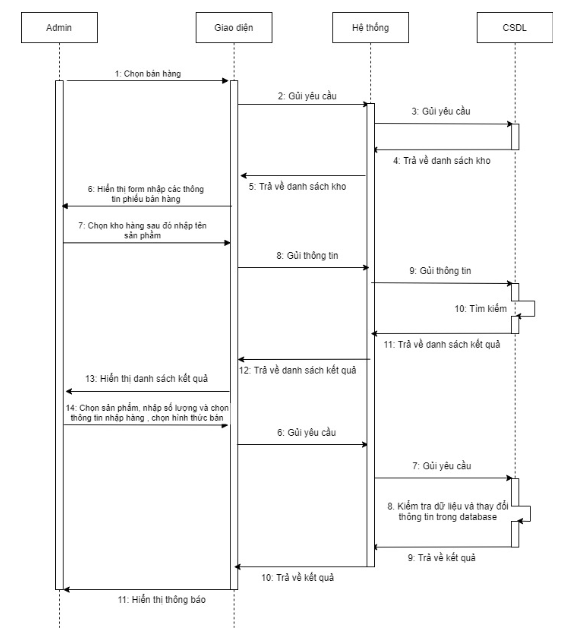
* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể thêm phiếu nhập hàng từ nhà cung cấp vào trong hệ thống.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form thêm phiếu nhập hàng (sản phẩm và số lượng, giá nhập, hạn sử dụng, kho hàng, nhà cung cấp, tiền hàng đã trả).
* Biểu đồ tuần tự:



Hình 2. 26 Biểu đồ tuần tự thêm phiếu nhập hàng

#### 2.3.8.2 UseCase bán hàng

* Đặc tả UseCase:
* Mô tả: Cho phép người dùng có thể thêm phiếu bán hàng cho khách hàng vào trong hệ thống.
* Actor: Admin.
* Điều kiện trước: Truy cập vào website dành cho quản trị viên và thiết bị sử dụng có kết nối internet với server.
* Luồng chính: Hiển thị form thêm phiếu bán hàng (kho hàng, sản phẩm và số lượng, đơn giá, hình thức bán hàng, khách hàng, chiết khấu, tiền thanh toán).
* Biểu đồ tuần tự:



Hình 2. 27 Biểu đồ tuần tự thêm phiếu bán hàng

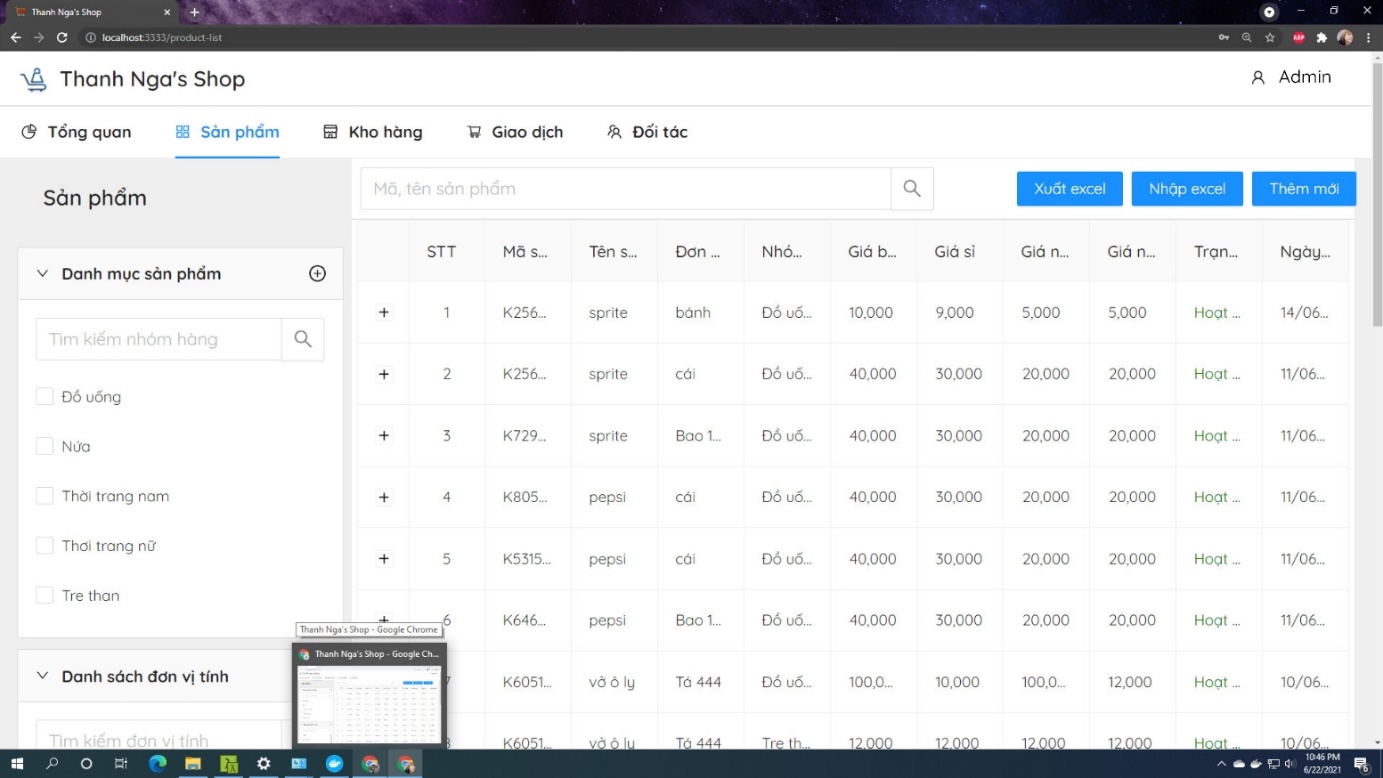
# CHƯƠNG 3: KẾT QUẢ CÀI ĐẶT

Với mục đích chủ yếu là sử dụng Docker để tối ưu hóa chu trình phát triển phần mềm nên website quản lý hàng hoá được xây dựng trong thời gian làm đồ án tốt nghiệp chỉ mang tính chất mô phỏng mà chưa được hoàn thiện như một website quản lý thông thường. Cụ thể các kết quả đạt được trong quá trình xây dựng website dùng công cụ Docker bao gồm các chức năng như dưới đây.

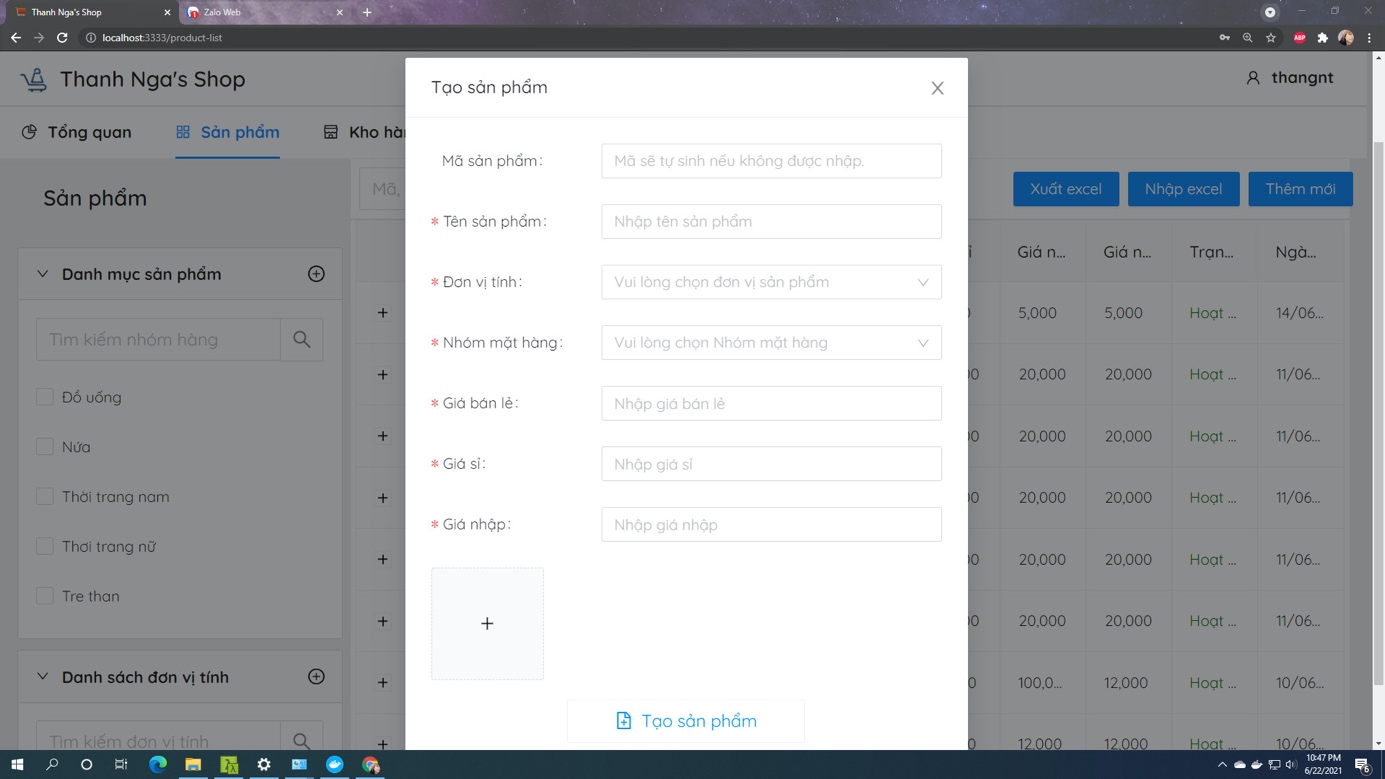
## 3.1 Hệ thống trang web quản lí hàng hóa

***Tab quản lí sản phẩm và danh mục sản phẩm***

Màn hình quản lí sản phẩm sẽ hiển thị danh sách các sản phẩm và danh mục sản phẩm của hệ thống. Người dùng có thể thêm, sửa, xoá danh mục sản phẩm ở bên trái màn hình. Với danh sách sản phẩm, người dùng có thể thêm sản phẩm và ấn vào chi tiết sản phẩm để sửa và xoá sản phẩm.

**

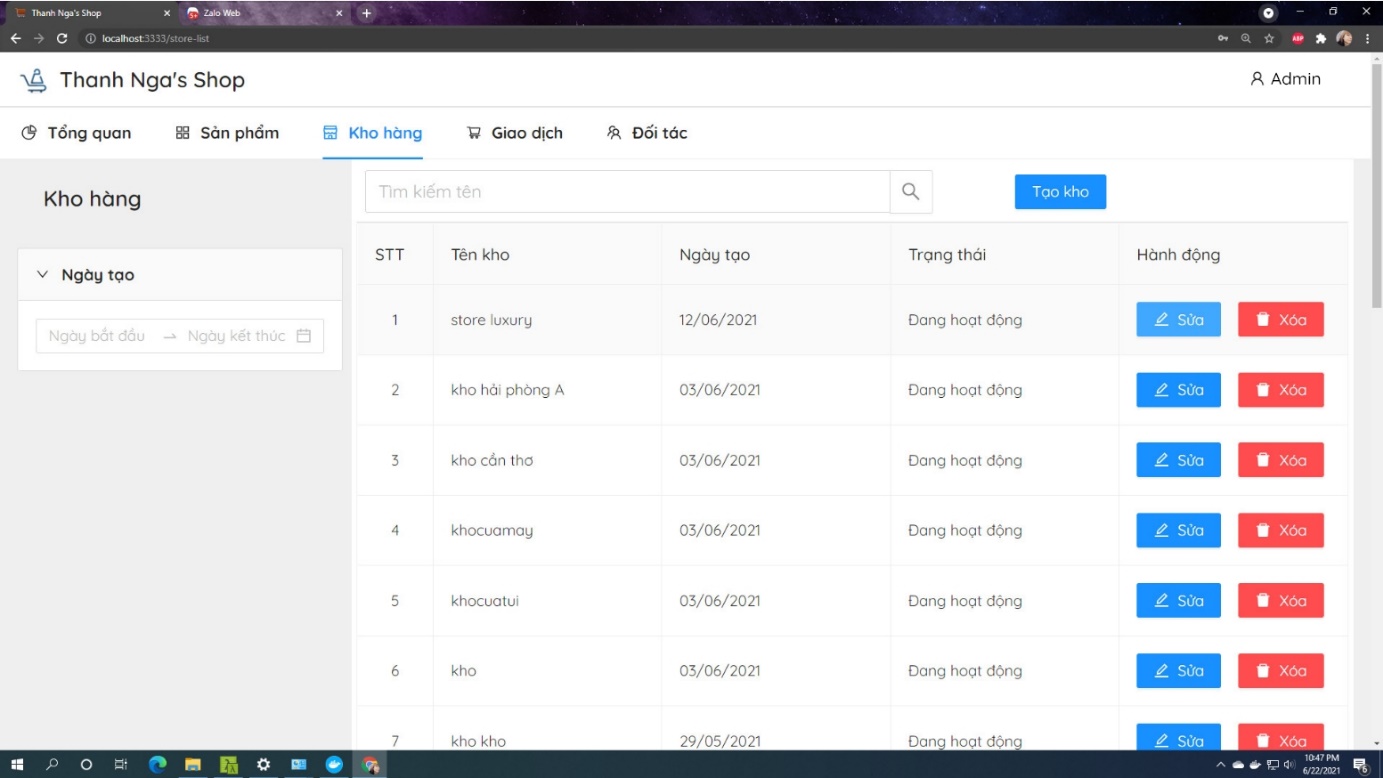
Hình 3. 1 Màn hình danh sách sản phẩm và sản phẩm



Hình 3. 2 Màn hình tạo sản phẩm

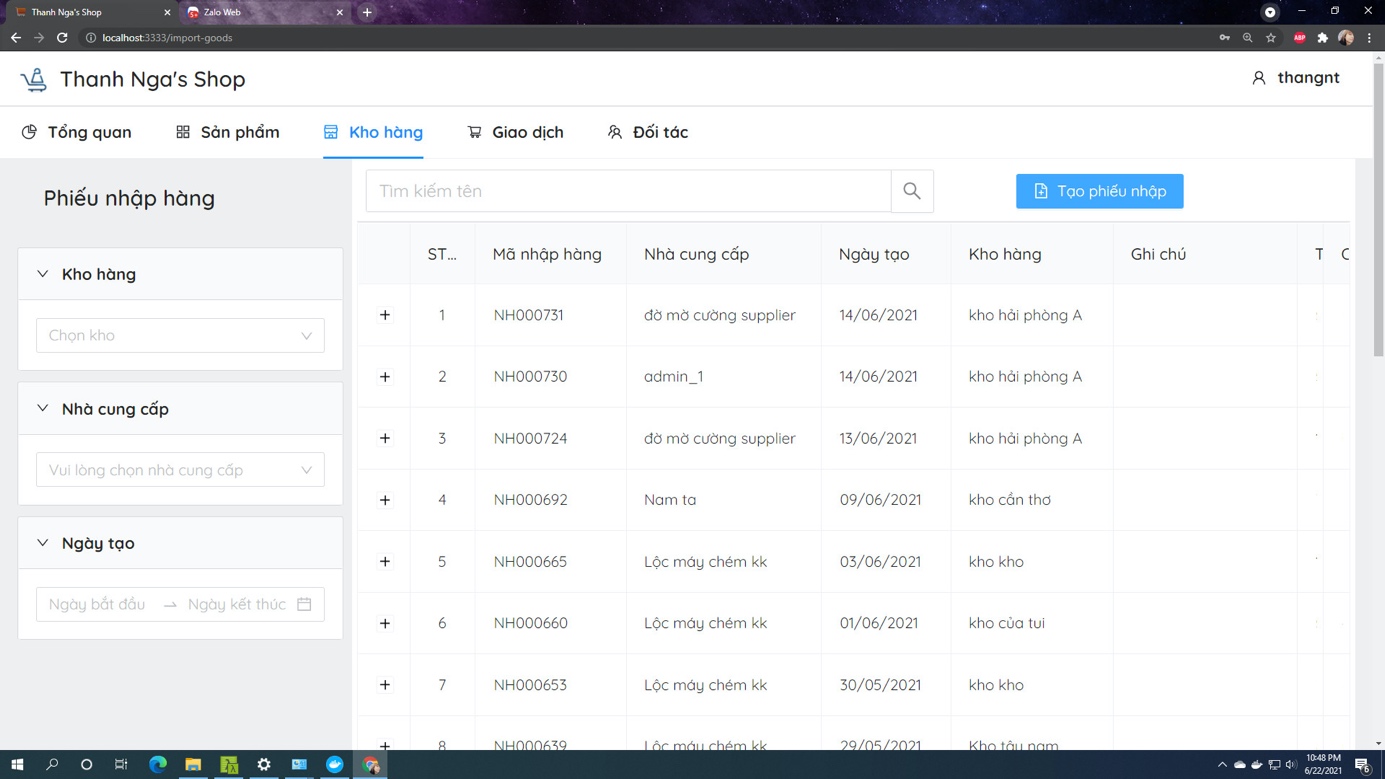
***Tab quản lí kho hàng***

Màn hình quản lí kho hàng sẽ hiển thị danh sách các kho hàng của hệ thống. Người dùng có thể thêm kho hàng và ấn vào một kho hàng để sửa và xoá kho hàng đó.

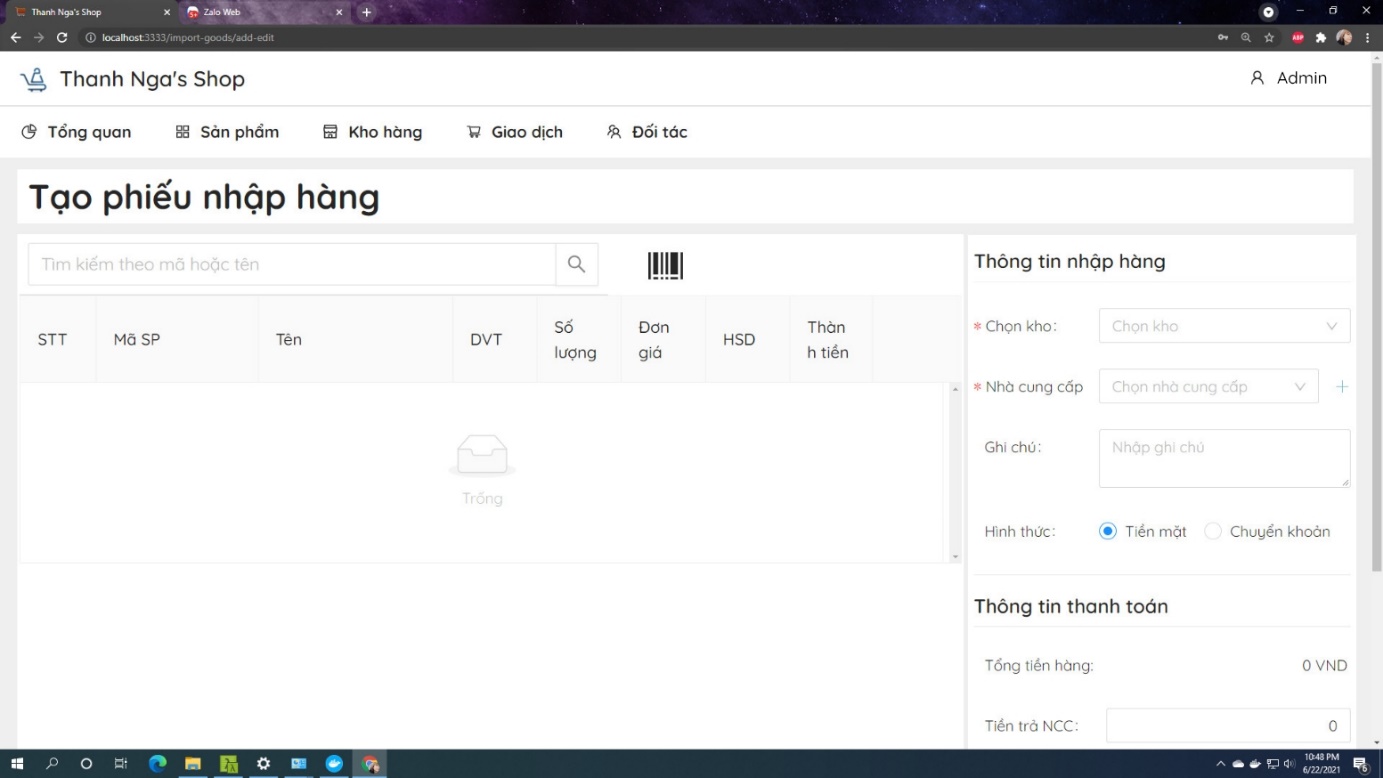


Hình 3. 3 Màn hình quản lí kho hàng

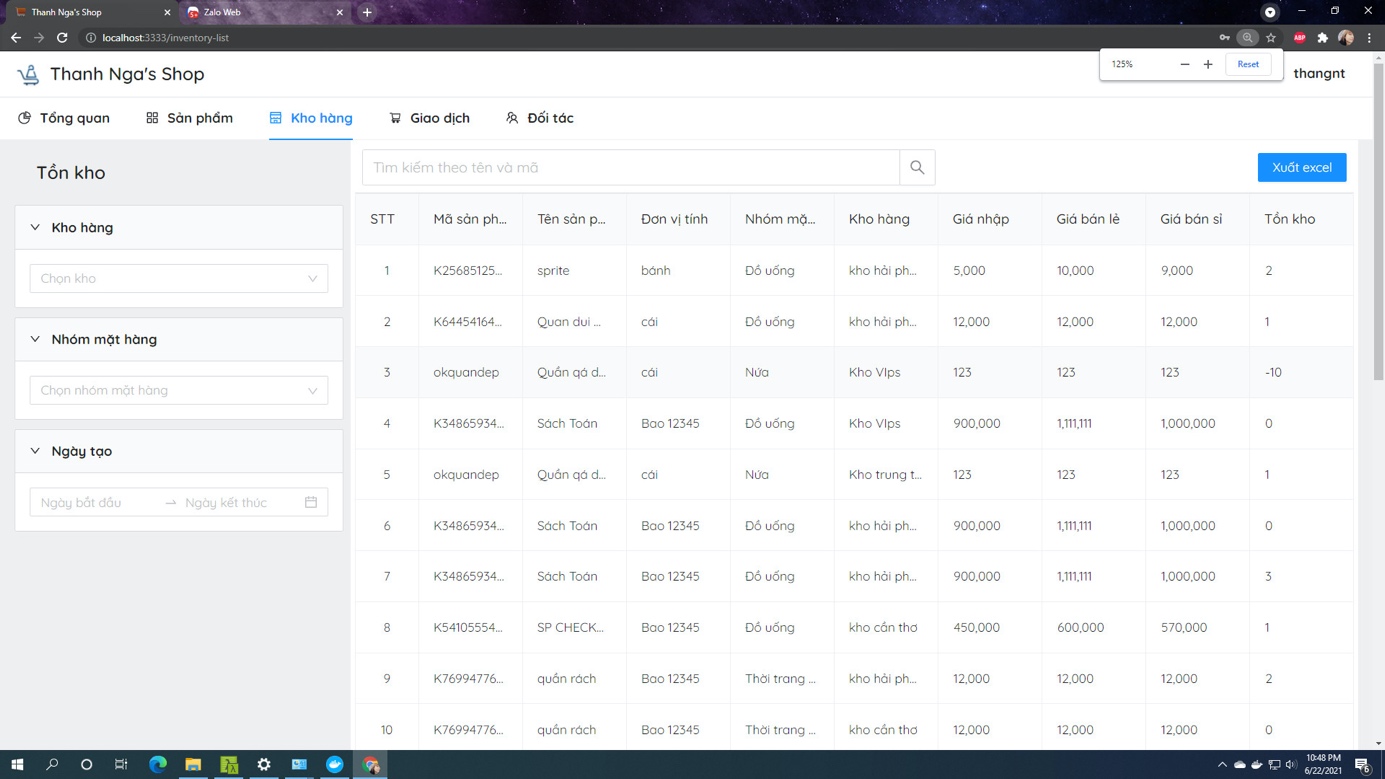
Bên cạnh đó người dùng còn có thể xem danh sách phiếu nhập hàng, tạo phiếu nhập hàng và xem danh sách sản phẩm tồn kho của hệ thống



Hình 3. 4 Màn hình danh sách phiếu nhập hàng



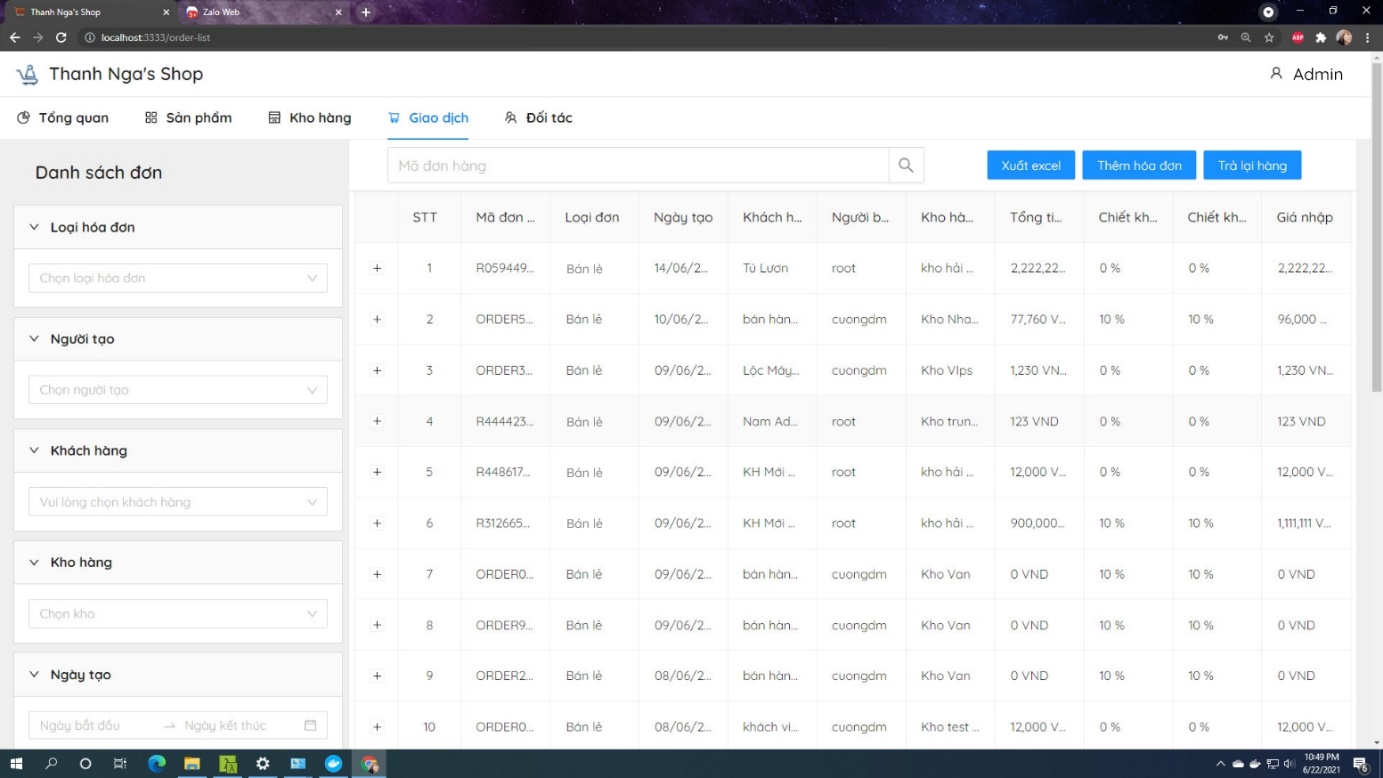
Hình 3. 5 Màn hình tạo phiếu nhập hàng



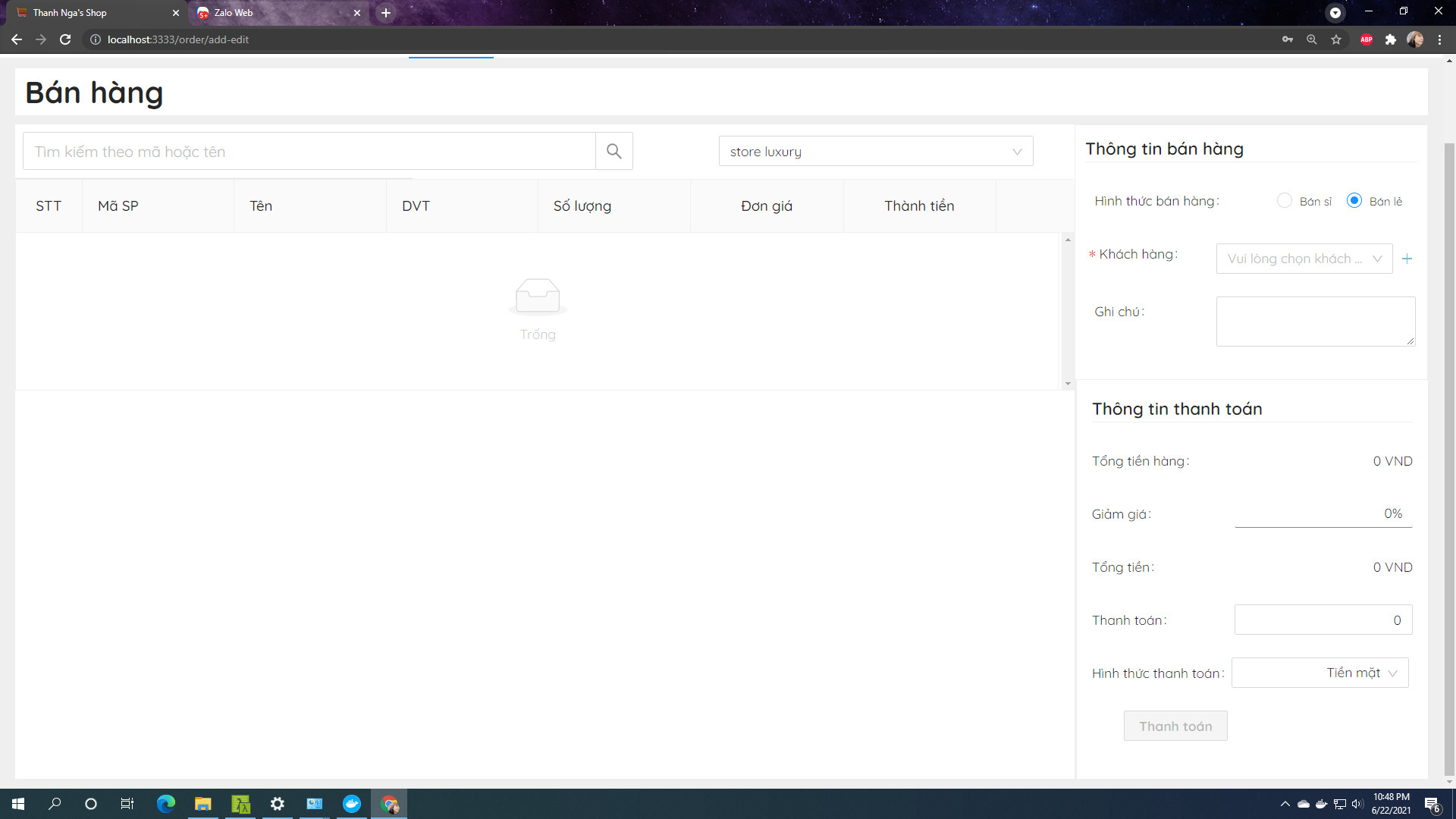
Hình 3. 6 Màn hình Danh sách sản phẩm tồn kho

***Tab giao dịch***

Màn hình quản lí giao dịch sẽ hiển thị danh sách các giao dịch bán hàng của hệ thống. Người dùng có thể thêm phiếu nhập hàng và ấn vào một đơn hàng để sửa và xoá đơn hàng đó.



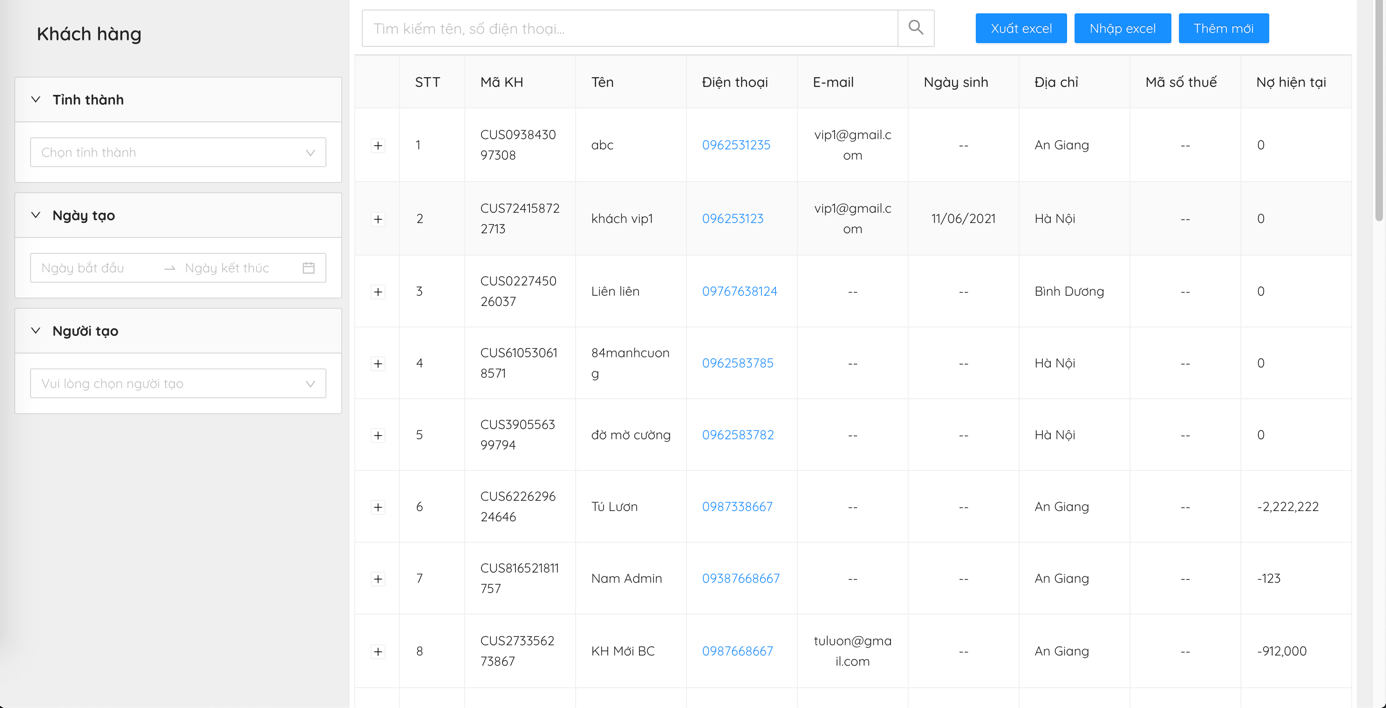
Hình 3. 7 Màn hình danh sách phiếu bán hàng



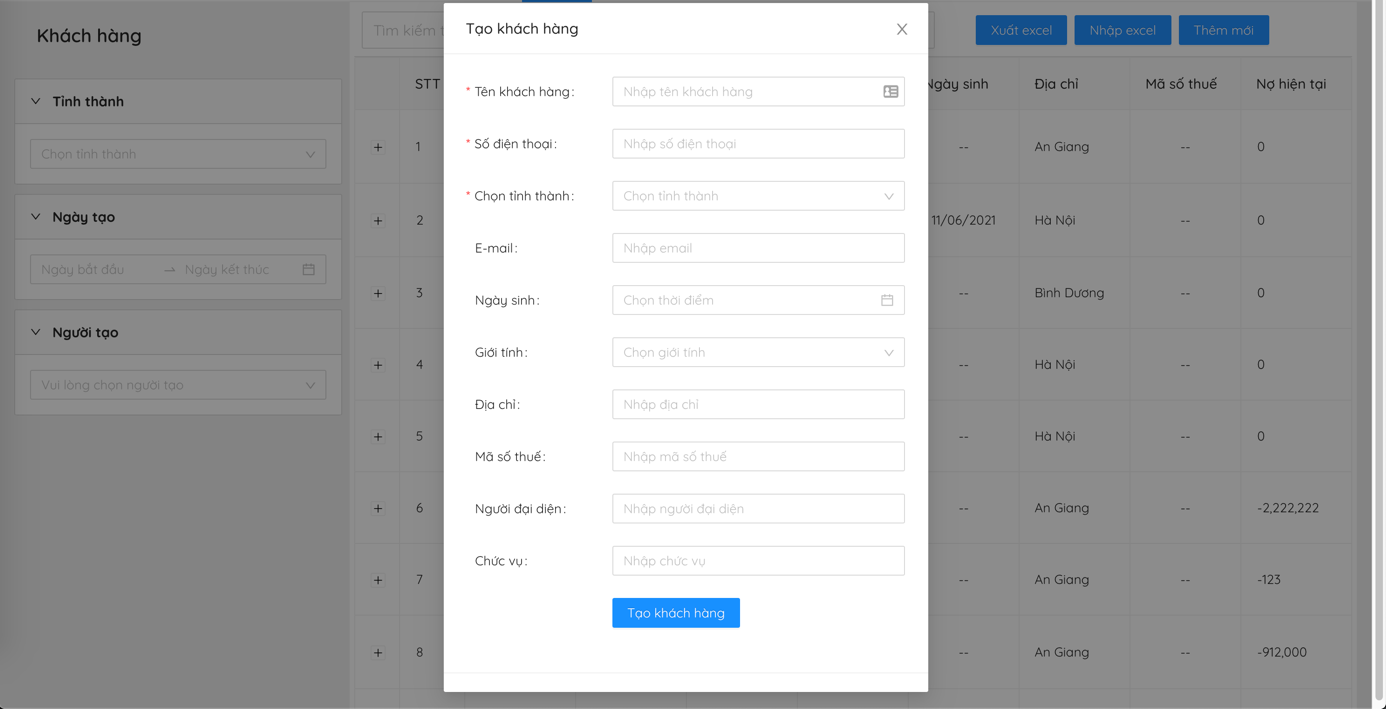
Hình 3. 8 Màn hình tạo phiếu bán hàng

***Tab quản lí khách hàng***

Màn hình quản lí khách hàng sẽ hiển thị danh sách các khách hàng của hệ thống. Người dùng có thể thêm khách hàng và ấn vào một khách hàng để sửa và xoá khách hàng đó.



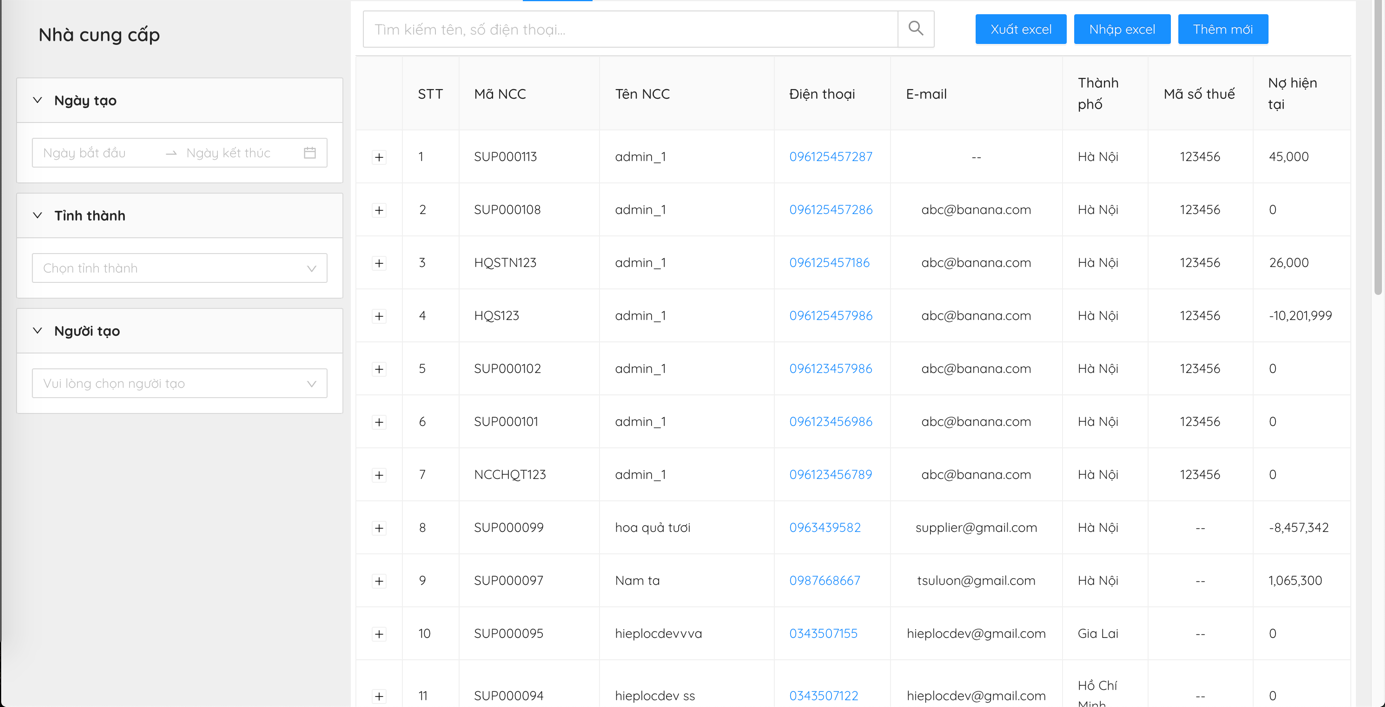
Hình 3. 9 Màn hình danh sách khách hàng



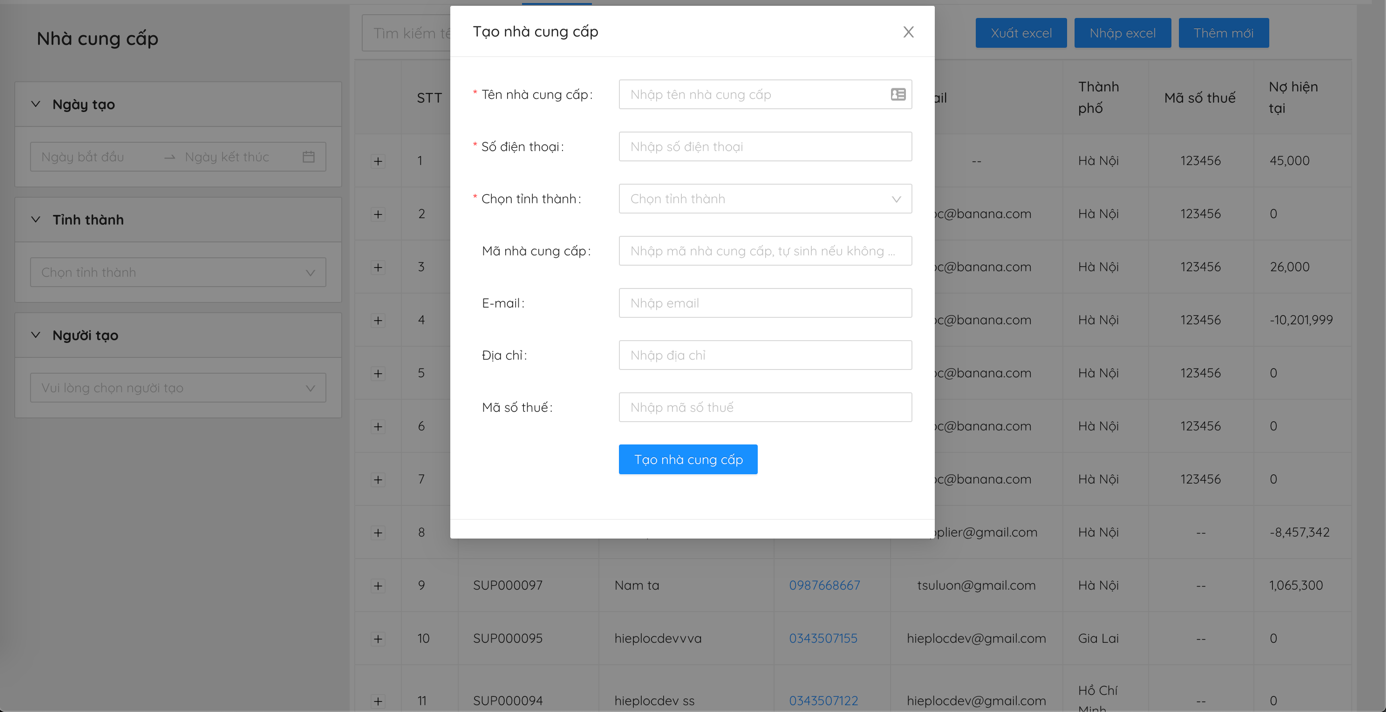
Hình 3. 10 Màn hình thêm khách hàng

***Tab quản lí nhà cung cấp***

Màn hình quản lí nhà cung cấp sẽ hiển thị danh sách các nhà cung cấp của hệ thống. Người dùng có thể thêm nhà cung cấp và ấn vào một nhà cung cấp để sửa và xoá nhà cung cấp đó.



Hình 3. 11 Màn hình danh sách nhà cung cấp

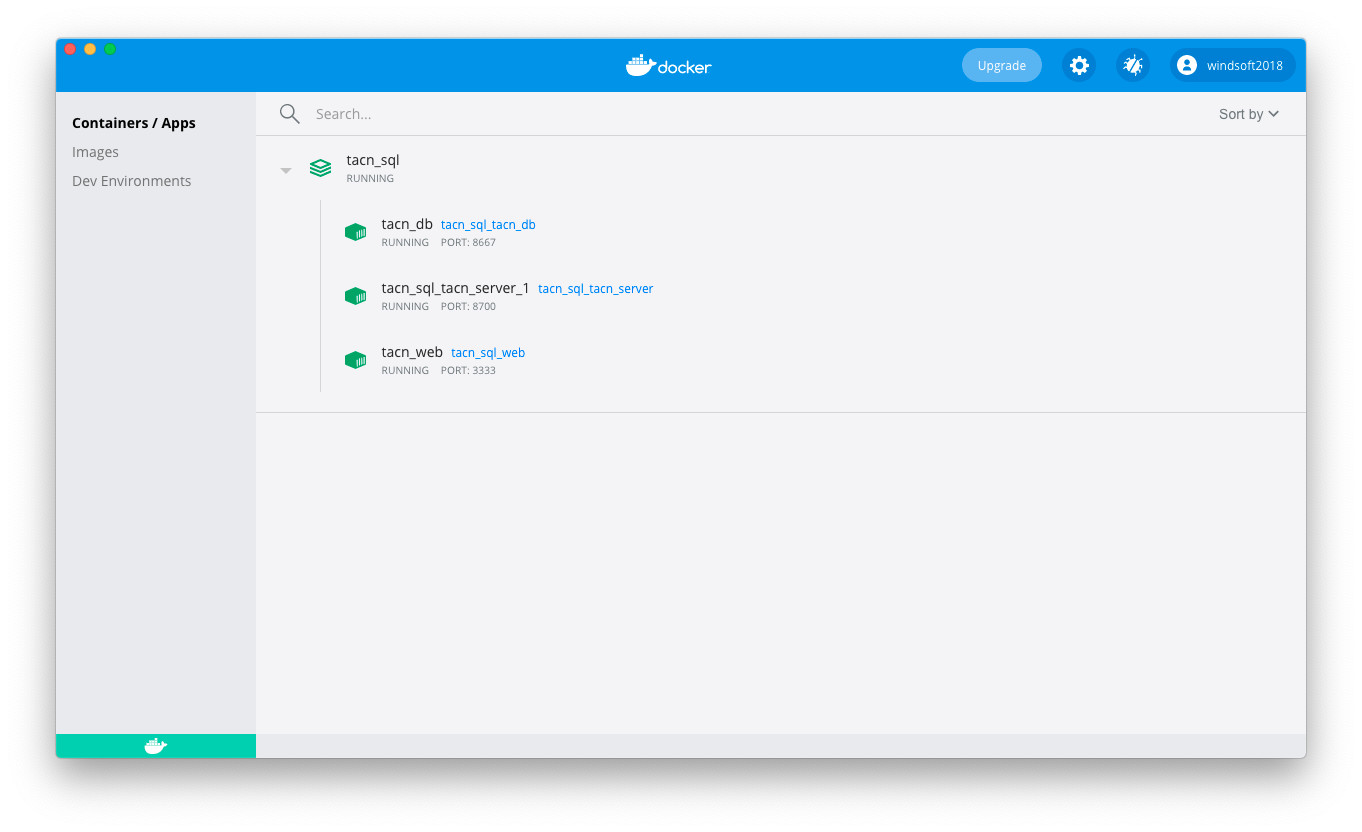


Hình 3. 12 Màn hình thêm nhà cung cấp

## 3.2 Deploy trên các môi trường bằng Docker

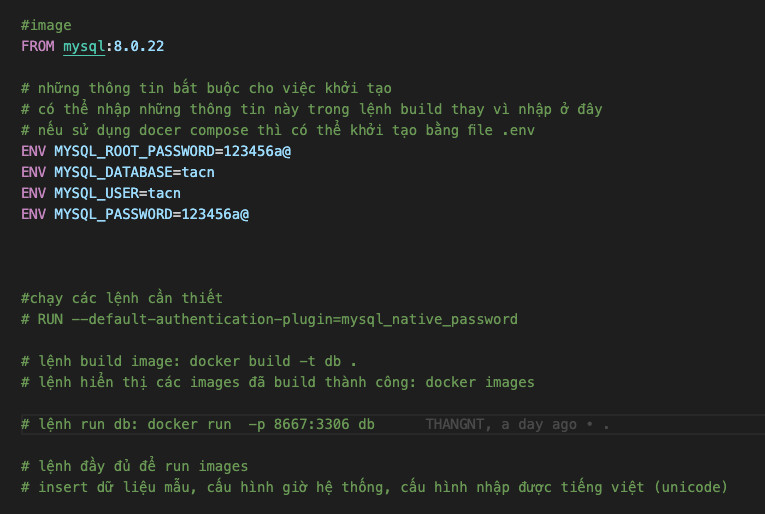
Sau khi đã xây dựng được hệ thống, việc setup và deploy application lên một hoặc nhiều server rất vất vả từ việc phải cài đặt các công cụ, môi trường cần cho application đến việc chạy được ứng dụng chưa kể việc không đồng nhất giữa các môi trường trên nhiều server khác nhau dẫn đến việc conflic. Đó là lúc Docker pháy huy tác dụng, các phần mềm sẽ được Container Engine cô lập bằng cách đóng gói chúng thành các container.

Các container được sử dụng để tạo môi trường triển khai trên các nền tảng khác nhau.



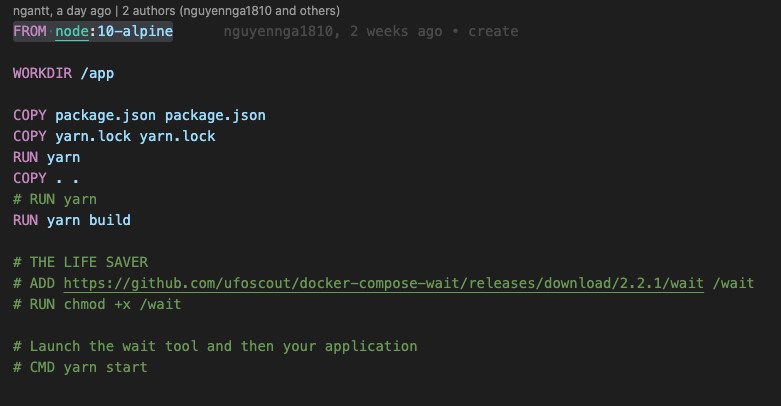
Hình 3. 13 Danh sách container để chạy dự án

Service chạy mysql quản lý dữ liệu cho hệ thống



Hình 3. 14 Dockerfile database

Service api của toàn bộ hệ thống



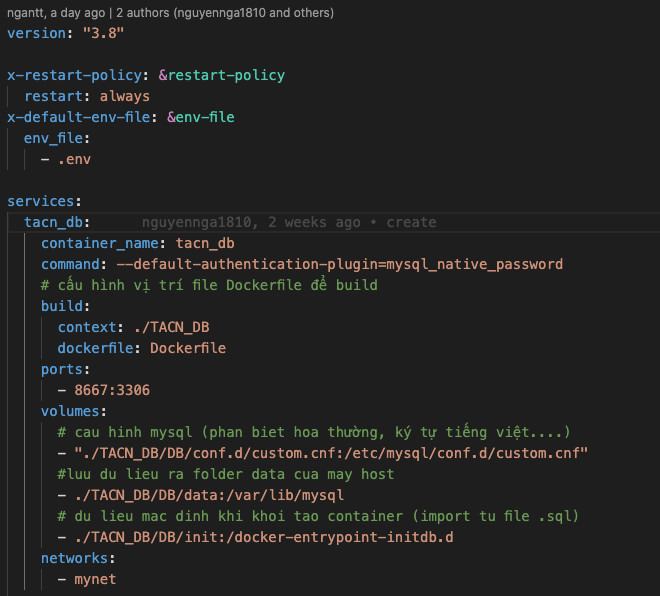
Hình 3. 15 Dockerfile api

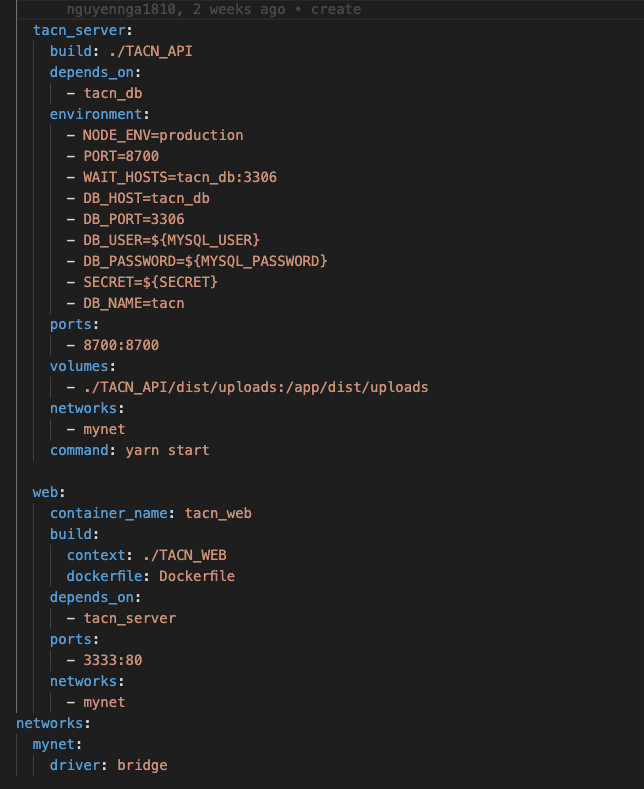
Service nodejs cho việc build website



Hình 3. 16 Dockerfile web\_build

File docker compose hỗ trợ triển khai đơn giản chỉ bằng một vài lệnh





Hình 3. 17 File Docker compose

Lợi ích sau khi sử dụng Docker để setup và deploy application:

* Không cần tốn nhiều thời gian vào việc cài đặt các ứng dụng môi trường, Docker start và stop chỉ trong vài giây.
* Chúng ta có thể khởi chạy container trên mỗi hệ thống mà chúng ta muốn.
* Container có thể build và loại bỏ nhanh hơn máy ảo.
* Dễ dàng thiết lập môi trường làm việc. Chỉ cần config 1 lần duy nhất và không bao giờ phải cài đặt lại các dependencies. Nếu thay đổi máy hoặc có người mới tham gia vào project thì chỉ cần lấy config đó và đưa cho họ.
* Nó giữ cho word-space của bạn sạch sẽ hơn khi bạn xóa môi trường mà ảnh hưởng đến các phần khác.

# KẾT LUẬN

Trong thời gian gần 3 tháng thực hiện học phần tốt nghiệp em đã đạt được những kết quả như sau:

* Tìm hiểu về khái niệm ảo hoá và các thành phần, mục tiêu, ưu nhược điểm của ảo hoá
* Tìm hiểu về mức độ ảo hoá ứng dụng.
* Nghiên cứu, tìm hiểu về công nghệ ảo hoá Docker và kiến trúc của Docker.
* Sử dụng thư viện Ant design để hỗ trợ cắt giao diện trang web.
* Nghiên cứu để triển khai trang web bằng công cụ Docker với mô hình MicroService.
* Nắm được các bước thu thập, phân tích, thiết kế một hệ thống trên nền tảng ngôn ngữ Javascript (framework React JS và Node JS), cơ sở dữ liệu MySQL để lập trình ra một website.
* Bước đầu xây dựng hệ thống quản lí bán hàng trên nền tảng website, hoàn thành một số chức năng chính giúp quản lí bán hàng….

Bên cạnh đó trong quá trình thực hiện vẫn còn một số hạn chế và khó khăn như:

* Giao diện còn chưa đẹp và hài hòa.
* Hệ thống chưa thực sự hoàn thiện.
* Yêu cầu chức năng phần mềm là rất nhiều, vì kinh nghiệm làm việc trong thực tế còn hạn chế nên đề tài của em chưa thể đưa ra một hệ thống hoàn chỉnh, việc quản trị cơ sở dữ liệu còn gặp nhiều khó khăn.

Hướng phát triển:

Trong tương lai em sẽ hoàn thiện website bổ sung các chức năng và tiện ích khác hỗ trợ quản lí bán hàng hiệu quả hơn, nghiên cứu sâu để nâng cấp sử dụng tiến trình CI/CD làm cho quá trình phát triển phần mềm nhanh hơn, thời gian release (ra đời) một sản phẩm được rút ngắn rất nhiều, nâng cao hiệu năng làm việc của lập trình viên.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | V. Hoang, "Tổng quan về ảo hóa VMWare," [Online]. Available: https://www.thegioimaychu.vn/blog/ao-hoa/huong-dan-tong-quan-ve-ao-hoa-vmware-p3588/. |
| [2] | AnhQT, "TỔNG QUAN VỀ CÔNG NGHỆ ẢO HÓA VMware – Part2," [Online]. Available: https://itforvn.com/tong-quan-ve-cong-nghe-ao-hoa-vmware-part2/. |
| [3] | C. Tín, "Giới thiệu tổng quan về kiến trúc của Docker," [Online]. Available: https://tel4vn.edu.vn/blog/gioi-thieu-tong-quan-ve-kien-truc-cua-docker/. |
| [4] | C. Tín, "SO SÁNH SỰ KHÁC NHAU GIỮA MÁY ẢO VÀ DOCKER CONTAINER," [Online]. Available: https://tel4vn.edu.vn/blog/so-sanh-su-khac-nhau-giua-may-ao-va-docker-container/. |