

## TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT KHOA CÔNG NGHỆ THÔNG TIN

***Học*** ***phần:***

## QUY TRÌNH VÀ CÔNG CỤ PHÁT TRIỂN PHẦN MỀM

***Đề* *tài:***

## **QUẢN LÝ WEBSITE BÁN CÁ**

### **Giảng viên hướng dẫn: Nguyễn Thị Nhung** Lớp: 73DCTT11

**Thành** **viên** **nhóm** **gồm:** Hoàng Lâm Tùng

Nguyễn Ngọc Anh

Phùng Quang Huy

Phạm Thế Mạnh

MỤC LỤC

[LỜI MỞ ĐẦU 4](#_Toc24948)

[CHƯƠNG 1. GIỚI THIỆU 5](#_Toc2415)

[1. Mục đích 5](#_Toc9774)

[2. Phạm vi 5](#_Toc2253)

[CHƯƠNG 2. GIT VÀ SVN 6](#_Toc4491)

[2.1. Git 6](#_Toc5110)

[Tại sao cần dùng Git? 6](#_Toc27509)

[2.1.1. Giới thiệu 6](#_Toc27304)

[2.1.2. Cách thức hoạt động 6](#_Toc13868)

[2.1.3. Một số thuật ngữ cơ bản 7](#_Toc10678)

[2.1.4. Các lệnh cơ bản 9](#_Toc30851)

[2.1.5. Quy trình làm việc với Git hiệu quả 11](#_Toc19662)

[2. SVN 12](#_Toc5899)

[2.2.1. Giới thiệu về SVN a, SVN là gì ? 12](#_Toc1897)

[2.2.2. Cách thức hoạt động 14](#_Toc7105)

[2.2.3. Một số thuật ngữ cơ bản 15](#_Toc30294)

[2.2.4. Các lệnh cơ bản 16](#_Toc2034)

[CHƯƠNG 3. SSH, FTP, SFTP 18](#_Toc30332)

[3.1. SSH 18](#_Toc27249)

[3.1.1. Định nghĩa 18](#_Toc4807)

[3.1.2. Mục đích sử dụng 18](#_Toc1111)

[3.1.3. Điều kiện và cách sử dụng 18](#_Toc19035)

[3.1.4. Cách thức xác thực và bảo mật trong SSH 19](#_Toc22756)

[3.1.5. Các lệnh cơ bản 20](#_Toc14720)

[3.2. FTP 21](#_Toc10548)

[3.2.1. Khái niệm FTP 21](#_Toc17134)

[3.2.2. Mô hình hoạt động 21](#_Toc8941)

[3.2.3 Các lệnh cơ bản 22](#_Toc275)

[3.3. SFTP 22](#_Toc30819)

[3.3.1. Khái niệm SFTP 22](#_Toc6295)

[3.3.2. Mô hình hoạt động 23](#_Toc31105)

[3.3.3. Các lệnh cơ bản 23](#_Toc22120)

[3.4. So Sánh FTP và SFTP 24](#_Toc13359)

[CHƯƠNG 4. TRIỂN KHAI DỰ ÁN 26](#_Toc8979)

[4.1. Giới thiệu 26](#_Toc19634)

[4.2. Cách triển khai 26](#_Toc36)

[CHƯƠNG 4. GIAO DIỆN 39](#_Toc23375)

[1. Giao diện trang chủ 39](#_Toc5620)

[2. Giao diện trang phụ kiện 39](#_Toc15236)

[3. Giao diện trang cửa hàng 39](#_Toc23687)

[4. Giao diện trang giới thiệu 40](#_Toc19924)

[5. Giao diện trang liên hệ 40](#_Toc8351)

[6. Giao diện trang đăng kí 41](#_Toc2333)

[7. Giao diện trang đăng nhập 41](#_Toc28214)

[8. Giao diện chi tiết sản phẩm 42](#_Toc24478)

[9. Giao diện giỏ hàng 42](#_Toc29692)

[10. Giao diện trang thanh toán 43](#_Toc23820)

[11. Giao diện đăng nhập ADMIN 43](#_Toc20365)

[12. Giao diện trang chủ ADMIN 44](#_Toc20072)

[13. Giao diện quản lý sản phẩm 44](#_Toc28786)

[14. Giao diện quản lý đơn hàng 44](#_Toc2421)

[15. Giao diện quản lý tài khoản 45](#_Toc28726)

# 

# LỜI MỞ ĐẦU

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, việc kinh doanh trực tuyến đang trở thành xu hướng không thể thiếu đối với các doanh nghiệp thuộc mọi lĩnh vực. Đặc biệt, ngành kinh doanh đồ thủy sinh và cá cảnh – một lĩnh vực đòi hỏi sự kết hợp giữa đam mê nghệ thuật và sự hiểu biết kỹ thuật – cũng không nằm ngoài xu hướng này.

Website bán đồ thủy sinh và cá cảnh không chỉ là nơi trưng bày sản phẩm mà còn là cầu nối quan trọng giữa doanh nghiệp và khách hàng. Một website chuyên nghiệp, thân thiện với người dùng sẽ giúp doanh nghiệp gia tăng khả năng tiếp cận thị trường, quảng bá thương hiệu và nâng cao trải nghiệm mua sắm của khách hàng.

Dưới đây là bản báo cáo của nhóm chúng em về đề tài này. Do kinh nghiệm còn ít và thời gian có hạn nên nếu có gì sai sót mong thầy cô bộ môn góp ý thêm. Chúng em xin chân thành cảm ơn.

1

# CHƯƠNG 1. GIỚI THIỆU

### Mục đích

Báo cáo này nhằm tìm hiểu và trình bày về sử dụng Git, SVN, SSH, FTP và SFTP trong quy trình phát triển phần mềm cho đề tài quản lý khách sạn. Mục tiêu cuối cùng của bài báo cáo là triển khai thành công dự án lên một máy ảo chạy hệ điều hành Linux.

### Phạm vi

Tập trung vào việc giới thiệu và mô tả các công cụ và giao thức quan trọng sau đây:

− **Git** **và** **SVN**: Hai hệ thống quản lý mã nguồn phiên bản phổ biến được sử dụng để theo dõi và quản lý mã nguồn phần mềm.

− **SSH** **(Secure** **Shell)**: Một giao thức mạng được sử dụng để thiết lập kết nối an toàn giữa các máy tính và máy chủ từ xa.

− **FTP** **(File** **Transfer** **Protocol)**: Một giao thức truyền tải tệp tin và thư mục qua mạng.

− **SFTP** **(SSH** **File** **Transfer** **Protocol)**: Một giao thức an toàn để truyền

tải tệp tin và thư mục thông qua kết nối SSH.

Chúng ta sẽ giải thích cách mỗi công cụ hoạt động, cách chúng được sử dụng trong môn học Công cụ và phát triển phần mềm và cách chúng sẽ được kết hợp để đạt được mục tiêu triển khai dự án trên máy ảo Linux.

[](https://www.studocu.com/vn?utm_campaign=shared-document%26utm_source=studocu-document%26utm_medium=social_sharing%26utm_content=bao-cao-quy-trinh-va-cong-cu-phat-trien-phan-mem)2

# CHƯƠNG 2. GIT VÀ SVN

### Git

**Tại** **sao** **cần** **dùng** **Git?**

Việc sử dụng Git để lưu trữ, kiểm soát phiên bản là cực kỳ cần thiết. Sử dụng Git sẽ không tạo ra xung đột giữa các lập trình viên khi thực hiện code hệ thống. Ngoài ra, với việc các dự án thường xuyên có thay đổi, sử dụng Git sẽ giúp nhà phát triển có thể dễ dàng tìm lại, quay lại các phiên bản cũ hơn. Việc kiểm soát sẽ cực kỳ đơn giản và hiệu quả.

Git cũng có các lợi ích tuyệt vời đối với các dự án đang chạy song song. Đặc biệt là các dự án có cùng một cơ sở code. Việc sử dụng Git sẽ giúp phân nhánh tốt hơn. Qua đó giúp việc thực hiện các dự án dễ dàng và hiệu quả hơn:

* Thao tác thực hiện nhanh, gọn lẹ hơn, việc sử dụng an toàn hơn.
* Người dùng có thể dễ dàng kết hợp các nhánh với nhau, qua đó giúp đơn giản hóa quy trình code theo nhóm.

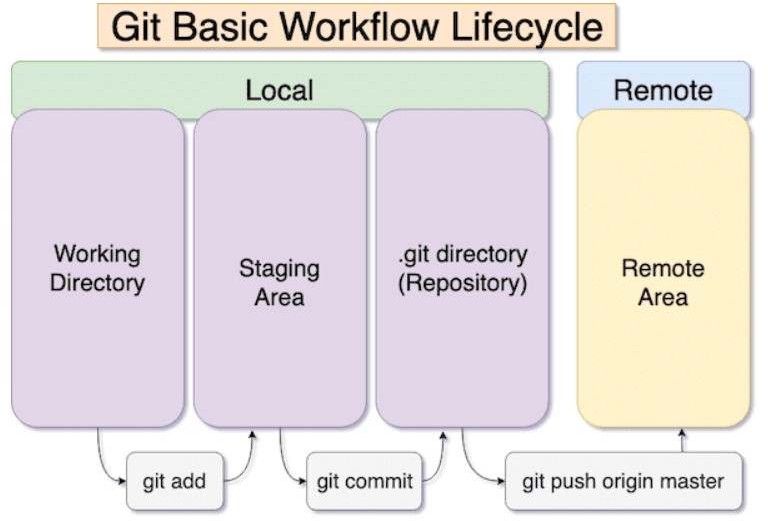
Khi sử dụng Git, người dùng chỉ cần sử dụng clone mã nguồn từ kho chứa/phiên bản nào đó có sẵn trong kho lưu trữ/ một nhánh trong kho lưu trữ. Hệ thống lưu trữ chính xác sẽ cho phép bạn có thể làm việc bất cứ lúc nào, ở bất cứ đâu. Giờ đây, việc lập trình các dự án song song sẽ cực kỳ dễ dàng, hiệu quả và chính xác.

### Giới thiệu

Git là một hệ thống quản lý phiên bản phân tán **(Distributed** **Version** **Control** **System** ***–*** ***DVCS*)**, nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. **Git** cung cấp cho mỗi lập trình viên kho lưu trữ (**repository**) riêng chứa toàn bộ lịch sử thay đổi.

### Cách thức hoạt động

3



**Working** **Directory**: đây là bản sao mã cục bộ, nơi có thể thêm sửa, xóa đổi mã.

**Staging** **area**: là khu vực trung gian giữa thư mục làm việc (working directory) và kho lưu trữ (repository). Khi bạn thay đổi các tệp trong thư mục làm việc, trước khi đưa những thay đổi đó vào kho lưu trữ, bạn cần commit chúng bằng cách thêm vào staging area. Điều này cho phép bạn quản lý rõ ràng việc nào sẽ được bao gồm trong phiên bản tiếp theo của dự án. Sau khi đã đưa thay đổi vào staging area, bạn có thể thực hiện commit để lưu lại các thay đổi này vào kho lưu trữ.

**Repository** (.git folder) là nơi bạn đặt các tập tin sau khi đã sắp xếp chúng, và khi bạn muốn lưu lại sự thay đổi của chúng, bạn chạy lệnh git commit để đưa những tập tin này vào trong kho lưu trữ. Điều này giúp bạn theo dõi lịch sử và quản lý phiên bản của dự án một cách cụ thể và có tổ chức.

**Remote** (Actual git repository): Đó là đích đến cuối cùng của các tệp, khi chúng nằm trong cơ sở mã git của bạn. Nơi bất kỳ ai có quyền truy cập vào mã của bạn đều có thể lấy từ đó.

* + 1. **Một** **số** **thuật** **ngữ** **cơ** **bản**

**Repository:** Repository là nơi sẽ ghi lại trạng thái của thư mục và file. Bằng việc đặt thư mục muốn quản lý lịch sử thay đổi dưới sự quản lý của

4

repository, có thể ghi chép lại lịch sử thay đổi của thư mục và file trong thư mục đó. Repository của Git được phân thành 2 loại là remote repository và local repository.

− **Remote** **repository:** Là repository để chia sẻ giữa nhiều người và bố trí trên server chuyên dụng.

− **Local** **repository:** Là repository bố trí trên máy của bản thân mình, dành

cho một người dùng sử dụng.

**Index:** là một khu vực trung gian nằm giữa thư mục làm việc (working directory) và kho lưu trữ (repository). Nó có vai trò là nơi bạn chuẩn bị các thay đổi để sau đó thêm chúng vào kho lưu trữ thông qua một commit.

**Branch:** Các **Branch** (nhánh) đại diện cho các **phiên** **bản** **cụ** **thể** của một kho lưu trữ tách ra từ project chính của bạn.Branch cho phép bạn theo dõi các thay đổi thử nghiệm bạn thực hiện đối với kho lưu trữ và có thể hoàn nguyên về các phiên bản cũ hơn.

**Commit:** Để ghi lại việc thêm/ thay đổi file hay thư mục vào repository thì sẽ thực hiện thao tác gọi là Commit. Khi thực hiện commit, trong repository sẽ tạo ra commit (hoặc revision) đã ghi lại sự khác biệt từ trạng thái đã commit lần trước đến trạng thái hiện tại nối tiếp nhau theo thứ tự thời gian.

**Push**: Được sử dụng để đưa nội dung từ kho lưu trữ cục bộ lên server.

**Pull**: Lấy thay đổi từ remote repository về.

**Head**: Các commit ở đầu của một branch được gọi là head. Nó đại diện cho commit mới nhất của repository mà bạn hiện đang làm việc.

**Master**: Nhánh chính của tất cả các repository của bạn. Nó bao gồm những thay đổi và commit gần đây nhất.

**Origin**: Đây là phiên bản mặc định của hệ thống lưu trữ. Origin sẽ đóng vai trò liên lạc với nhánh chính.

**Fetch**: Sử dụng lệnh Fetch sẽ cho phép bạn nạp và tải các bản sao trên hệ thống lưu trữ và tải các tệp nhánh về máy tính cá nhân. Bạn có thể sử dụng lệnh

5

Fetch để lưu các thay đổi mới nhất vào kho lưu trữ. Fetch cho phép bạn có thể lưu trữ nhiều nhánh trong cùng một thời điểm.

**Fork**: là thuật ngữ thể hiện các bản sao của một kho lưu trữ. Tiện ích fork cho phép người dùng - các lập trình viên có thể thử nghiệm các thay đổi. Nhờ việc sử dụng trên bản sao nên người dùng có thể thoải mái thay đổi mà không lo ảnh hưởng tới các dự án chính.

**Tags**: được sử dụng để theo dõi commit quan trọng trong hệ thống. Người dùng có thể sử dụng các tags để chú thích commit, giúp việc tra cứu đối tượng nhanh chóng và đầy đủ hơn.

### Các lệnh cơ bản

|  |  |
| --- | --- |
| **Câu** **lệnh** **Git** | **Mô** **tả** |
| *git* *init* | Khởi tạo một kho lưu trữ Git trống. |
| *git* *clone* *<URL>* | Clone dự án từ kho lưu trữ xa. |
| *git* *add* *<tên\_tệp>* | Thêm tệp/tập tin vào Git (staging). |
| *git* *add* *.* | Thêm tất cả tệp/tập tin thay đổi vào Git (staging). |
| *git* *commit* *-m* *"Thông* *điệp* *commit"* | Lưu các thay đổi vào Git. |
| *git* *status* | Xem trạng thái sự thay đổi trong thư mục làm việc. |
| *git* *diff* | Xem sự thay đổi trong các tệp/tập tin. |
| *git* *push* *origin* *<tên\_nhánh>* | Gửi các thay đổi lên kho lưu trữ xa. |
| *git* *pull* *origin* *<tên\_nhánh>* | Lấy sự thay đổi từ kho lưu trữ xa. |
| *git* *checkout* *-b*  *<tên\_nhánh\_mới>* | Tạo một nhánh mới và chuyển đến nhánh đó. |
| *git* *checkout* *<tên\_nhánh>* | Chuyển đổi giữa các nhánh. |
| *git* *checkout* *<ID\_commit>* | Tạo và chuyển sang một nhánh từ một commit cụ thể (detached HEAD). |
| *git* *branch* *-a* | Xem danh sách tất cả các nhánh cục bộ và nhánh từ xa. |

6

|  |  |
| --- | --- |
| *git* *branch* *-d* *<tên\_nhánh>* | Xóa một nhánh cục bộ đã được hợp nhất |
| *git* *branch* *-D* *<tên\_nhánh>* | Xóa một nhánh cục bộ chưa được hợp nhất |
| *git* *branch* *<tên\_nhánh\_mới>* | Tạo một nhánh từ một nhánh khác (không chuyển đến nhánh mới). |
| *git* *branch* | Xem danh sách các nhánh cục bộ. |
| *git* *branch* *-m* *<tên\_nhánh\_cũ>*  *<tên\_nhánh\_mới>* | Đổi tên một nhánh cục bộ. |
| *git* *log* | Xem lịch sử commit. |
| *git* *log* *<tên\_nhánh>* | Xem lịch sử commit trên một nhánh cụ thể. |
| *git* *log* *--oneline* | Xem lịch sử commit dưới dạng một dòng cho mỗi commit. |
| *git* *rm* *<tên\_tệp>* | Xóa các tệp/tập tin khỏi Git (đã bỏ tracking). |
| *git* *rm* *--cached* *<tên\_tệp>* | Xóa tệp/tập tin khỏi Git (vẫn còn tracking). |
| *git* *reset* *<tên\_tệp>* | Bỏ tệp đã thêm ra khỏi staging arae |
| *git* *stash* | Tạo và áp dụng một stash (lưu trữ tạm thời các thay đổi chưa commit). |
| *git* *stash* *apply* | Ap dụng lại stash. |
| *git* *stash* *list* | Xem danh sách các stash đã tạo. |
| *git* *stash* *drop* *<stash\_index>* | Xóa một stash cụ thể. |
| *git* *stash* *save* *--keep-index* | Tạo và áp dụng một ghim (chứa thay đổi tạm thời nhưng không xóa khỏi stash). |
| *git* *stash* *branch*  *<tên\_nhánh\_mới>* | Tạo một nhánh mới từ stash và áp dụng stash vào nhánh mới. |
| *git* *stash* *clear* | Xóa tất cả stash. |
| *git* *show* *<ID\_commit>* | Xem thông tin về một commit cụ thể. |
| *git* *show* *<tên\_nhãn>* | Xem thông tin về một nhãn cụ thể. |

7

|  |  |
| --- | --- |
| *git* *tag* *-a* *<tên\_nhãn>* *-m* *"Thông* *điệp* *tag"* *<ID\_commit>* | Tạo và quản lý nhãn (tag) cho các commit cụ thể. |
| *git* *tag* | Xem danh sách các nhãn đã tạo. |
| *git* *tag* *-d* *<tên\_nhãn>* | Xóa một nhãn cục bộ hoặc từ xa. |
| *git* *tag* *-a* *<tên\_nhãn>*  *<ID\_commit>* | Tạo một nhãn (tag) cho một commit cụ thể. |
| *git* *merge* *<tên\_nhánh>* | Merge các thay đổi từ một nhánh khác vào nhánh hiện tại. |
| *git* *fetch* *--* *all* | Lấy tất cả các nhánh trên reponsitory về local |

### Quy trình làm việc với Git hiệu quả

Để sử dụng GIT hiệu quả trong các dự án, sẽ có những quy tắc và quy trình làm việc giúp cho quá trình phát triển mã nguồn được đồng bộ, giảm thiểu xảy ra sai sót, xung đột. Các thành viên trong nhóm cần tuân thủ các quy tắc của GIT working flow như sau:

− Repository của dự án bắt buộc phải có hai nhánh là master và develop.

Trong đó nhánh master là nhánh stable chứa code hoàn chỉnh được dùng cho production. Nhánh develop là nhánh có mã nguồn mới nhất về các tính năng được phát triển và đang được deploy lên server để testing.

− Người phát triển phần mềm cần fork repository gốc của dự án về thành

một repository của mình, khi phát triển về một tính năng mới hay sửa lỗi, người lập trình cần tạo ra một branch mới từ repository của mình rồi mới thực hiện công việc trên branch đó. Khi hoàn thành công việc thì commit code của mình vào branch sau đó tạo merge request để merge code của mình vào repository gốc của dự án.

− Cần pull code từ repository gốc khi phát triển một tính năng mới để đảm

bảo code trên máy mình là code mới nhất từ dự án, tránh xảy ra xung đột (conflict code) sau khi tạo merge request.

8

− Đối với người có quyền merge code từ các branch khác vào branch

develop cần có review kĩ càng, khi xảy ra conflict phải gọi những người tham gia vào việc phát triển tính năng ấy để có thể giải quyết một cách tốt nhất, tránh trường hợp sau khi merge code không chạy được hoặc chạy sai hoàn toàn so với mục đích của người lập trình.

## SVN

* + 1. **Giới** **thiệu** **về** **SVN** **a,** **SVN** **là** **gì** **?**

SVN (Subversion) là một phần mềm mã nguồn mở có khả năng kiểm soát và quản lý các phiên bản mã nguồn khác trong quá trình phát triển phần mềm. Có thể hiểu rằng, nó giúp người dùng có thể thực hiện các thao tác chỉnh sửa tệp và thư mục như những công cụ khác. Ngoài ra, bạn có thể xem lại những thay đổi và truy xuất các phiên bản trước của thông tin dữ liệu.

### b, Quá trình phát triển của SVN:

**Năm** **2000:** CollabNet, Inc. bắt đầu chiêu mộ lập trình viên để xây dựng một hệ thống kiểm soát phiên bản thay thế cho CVS. Họ tạo CollabNet Enterprise Edition (CEE) với một thành phần là kiểm soát phiên bản, nhưng hạn chế của CVS trở nên rõ ràng.

**Tháng** **2** **năm** **2000:** CollabNet liên hệ với Karl Fogel, tác giả cuốn "Phát Triển Mã Nguồn Mở với CVS", để tham gia vào dự án. Karl Fogel đồng ý tham gia và bắt đầu thiết kế cơ bản của hệ thống mới. Karl Fogel và Jim Blandy (người thảo luận về thiết kế một hệ thống kiểm soát phiên bản mới từ trước) bắt đầu xây dựng hệ thống kiểm soát phiên bản mới, gọi là Subversion.

**Tháng** **3** **năm** **2000:** CollabNet thuê Karl Fogel và Ben Collins-Susman để thiết kế chi tiết của Subversion.

**Năm** **2001:** Nhờ sự giúp đỡ từ các chuyên gia khác như Brian Behlendorf, Jason Robbins (CollabNet), và Greg Stein (chuyên về WebDAV/DeltaV), Subversion nhanh chóng thu hút cộng đồng phát triển tích cực.

9

**Ngày** **31** **tháng** **8** **năm** **2001:** Subversion bắt đầu "tự-lưu-trữ", chuyển từ việc sử dụng CVS quản lý mã nguồn riêng của nó sang việc sử dụng Subversion.

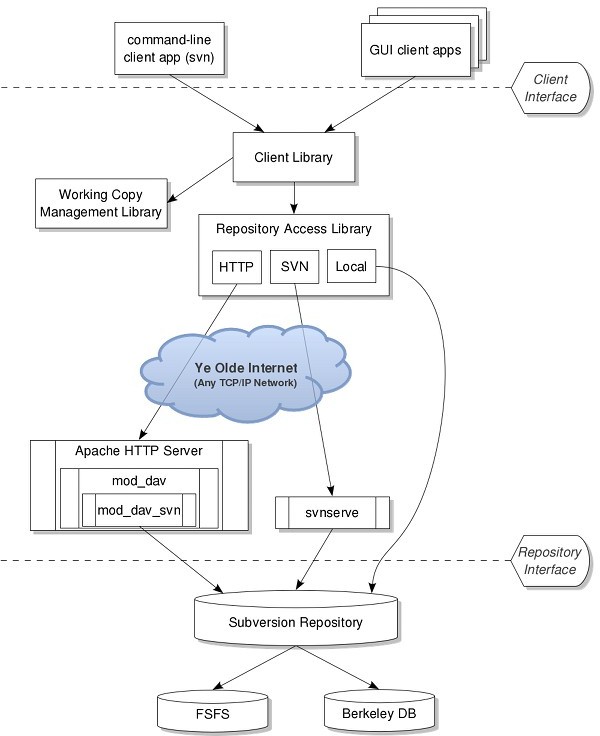
**Năm** **2009:** CollabNet hợp tác với nhà phát triển Subversion để tích hợp dự án vào Apache Software Foundation (ASF), một tổ chức nổi tiếng về mã nguồn mở.

**Năm** **2010:** Subversion chính thức được tiếp nhận vào danh sách dự án quan trọng của ASF. Dự án đổi tên thành "Apache Subversion" và có địa chỉ trang web mới là [http://subversion.apache.org.](http://subversion.apache.org/)

Từ những bước trên, Subversion đã phát triển từ một ý tưởng để cải thiện hệ thống kiểm soát phiên bản hiện có thành một dự án mã nguồn mở quan trọng dưới sự hỗ trợ của CollabNet, Karl Fogel, và cộng đồng phát triển rộng lớn.

10

### Cách thức hoạt động

****

Subversion được chia thành ba phần chính, mỗi phần có các thành phần khác nhau và được ngăn cách bởi giao diện (interface) của người dùng và kho lưu trữ:

**Giao** **diện** **Người** **dùng** **(User** **Interface):** Gồm GUI client apps (ứng dụng giao diện đồ họa) và command-line client app (ứng dụng dòng lệnh) như

`svn`.

11

**Thư** **viện** **Khách** **hàng** **(Client** **Library):** Client library làm cầu nối giữa giao diện người dùng và các thành phần khác của Subversion. Trỏ vào Working Copy Management Library và Repository Access Library.

**Working** **Copy** **Management** **Library:** Điều khiển quản lý bản sao làm việc (working copy) trên máy người dùng.

**Repository** **Access** **Library:** Cho phép tương tác với kho lưu trữ thông qua các giao thức như HTTP và SVN (Subversion protocol). Các thành phần HTTP và SVN trỏ vào lớp dưới đây thông qua mạng TCP/IP.

**Apache** **HTTP** **(mod\_dav,** **mod\_dav\_svn)** **và** **svnserve:** Là các máy chủ để xử lý yêu cầu từ Repository Access Library. Dùng để cung cấp các giao thức truy cập và quản lý kho lưu trữ.

**Subversion** **Repository:** Đây là lớp cuối cùng chứa dữ liệu và lịch sử thay đổi của dự án. Gồm hai loại lưu trữ: FSFS (File System) và Berkeley DB.

Tổng cộng, kiến trúc của Subversion được xây dựng theo mô hình client- server, với các thành phần khác nhau cùng làm việc để quản lý sự phát triển và kiểm soát phiên bản của dự án.

* + 1. **Một** **số** **thuật** **ngữ** **cơ** **bản**

− **Repository**: nơi lưu trữ tất cả các phiên bản của mã nguồn, cùng với lịch

sử thay đổi. Có thể hiểu repository như một cơ sở dữ liệu chứa toàn bộ thông tin về mã nguồn và các thay đổi liên quan.

− **Working** **copy:** bản sao công việc là không gian làm việc cục bộ và riêng

tư mà bạn sử dụng để tương tác với kho lưu trữ Subversion trung tâm. Bạn sử dụng bản sao làm việc để sửa đổi nội dung dự án của mình và tìm nạp các thay đổi do người khác cam kết.

− **Revision**: Mỗi lần commit, một phiên bản mới sẽ được tạo ra trong

repository. Phiên bản này được đánh số để thể hiện sự thứ tự của các thay đổi.

12

− **Branch**: Nhánh là quá trình tạo ra một bản sao của mã nguồn từ nhánh

chính hoặc một nhánh hiện tại. Những thay đổi trong nhánh mới này có thể phát triển độc lập mà không ảnh hưởng đến nhánh gốc.

− **Trunk**: Đây là một thư mục tại đó tất cả các phát triển chính xảy ra và thường được kiểm tra bởi các lập trình viên làm việc trên dự án.

− **Tags**: Thư mục các thẻ được sử dụng để giữ các ảnh chụp nhanh

(snapshot) được đặt tên của dự án. Hoạt động thẻ cho phép đặt các tên để mô tả và dễ ghi nhớ cho các phiên bản cụ thể trong repository.

− **Merge**: Là quá trình kết hợp các thay đổi từ một nhánh vào nhánh khác.

Khi bạn hoàn thành công việc trên một nhánh (ví dụ: tính năng hoặc sửa lỗi), bạn có thể merge những thay đổi đó vào nhánh chính để tích hợp vào phiên bản chính.

### Các lệnh cơ bản

|  |  |
| --- | --- |
| **Lệnh** **SVN** | **Mô** **tả** |
| *svn* *checkout* *<URL>* | sao chép dự án từ kho lưu trữ về máy cục bộ. |
| *svn* *update* | Cập nhật dự án với phiên bản mới nhất từ kho lưu trữ. |
| *svn* *add* *<tên\_tệp>* | Thêm tệp/tập tin mới vào phiên bản kiểm soát. |
| *svn* *commit* *-m* *"Thông* *điệp* *commit"* | Lưu các thay đổi vào kho lưu trữ với thông điệp. |
| *svn* *status* | Kiểm tra trạng thái của các tệp/tập tin trong thư mục làm việc. |
| *svn* *diff* | Xem sự thay đổi trong các tệp/tập tin so với phiên bản gốc. |
| *svn* *log* | Xem lịch sử commit của dự án. |
| *svn* *move* *<tên\_cũ>* *<tên\_mới>* | Di chuyển hoặc đổi tên tệp/tập tin. |

13

|  |  |
| --- | --- |
| *svn* *copy* *<nguồn>* *<đích>* | Sao chép tệp/tập tin hoặc thư mục. |
| *svn* *delete* *<tên\_tệp\_được\_xóa>* | Xóa tệp/tập tin hoặc thư mục khỏi phiên bản kiểm soát. |
| *svn* *revert* *<tên\_tệp\_hoàn\_tác>* | Hoàn tác (revert) thay đổi không mong muốn. |
| *svn* *copy* *<nguồn>* *<đích>* | Tạo nhánh mới từ nhánh hiện tại hoặc từ một phiên bản cụ thể. |

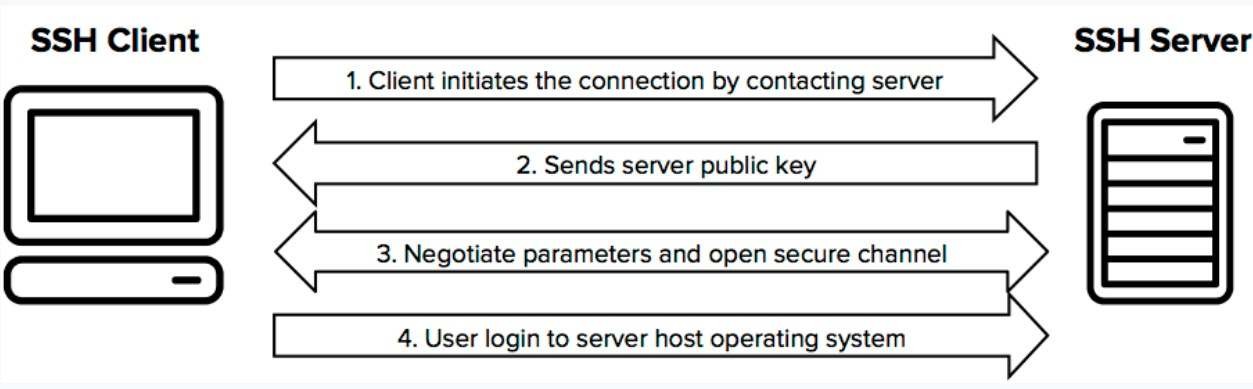
14

# CHƯƠNG 3. SSH, FTP, SFTP

## SSH

### Định nghĩa

**SSH** (Secure Shell) là giao thức mạng được mã hoá bảo mật, nó tạo ra kết nối bảo mật (tạo ra một đường dẫn tách biệt với cách dịch vụ khác để duy trì kết nối riêng tư giữa máy trạm và máy chủ) trên một mạng lưới không bảo mật. Nó cũng được hiểu như ứng dụng dùng để kết nối đăng nhập vào 1 hệ thống máy tính bởi người dùng.



Mô hình hoạt động của SSH

### Mục đích sử dụng

Giao thức SSH được dùng ở nhiều nền tảng từ Linux, MacOS, Window với các mục đích như:

* + - * Thiết lập kết nối an toàn giữa hai máy tính, cho phép người dùng truy cập và quản lý máy chủ từ xa một cách bảo mật.
      * Mục đích chính là bảo vệ thông tin truyền qua mạng khỏi bị đánh cắp hoặc thay đổi với bất kỳ ai có ý định xấu.

### Điều kiện và cách sử dụng

Để sử dụng kết nối SSH, bạn phải được cung cấp một dịch vụ SSH từ máy Remote (từ xa, server), sau đó ở máy Client (local) sử dụng các chương trình SSH Client để kết nối đến máy remote và thực hiện các tác vụ tùy nhu cầu.

* + - * **Máy** **chủ** **dịch** **vụ** **Remote**: là máy tại đó chạy một SSH Server, nó cung cấp dịch vụ kết nối SSH đến nó, thông qua xác thực tài khoản user/pass hoặc xác thực bằng public/private key

15

* + - * **SSH** **Client**: là các chương trình chạy ở máy trạm (local) có chức năng kết nối đến SSH Server. Bản thân OpenSSH cũng là Client để kết nối - nên ở máy trạm ta sẽ tập trung vào sử dụng OpenSSH Client để thực hành, ngoài ra còn nhiều SSH Client khác, như Putty chạy trên Windows. Khi kết nối có thể tùy cấu hình từ OpenSSH Server có thể xác thực bằng cách nhập username/password hoặc bằng cặp file public/private key.
    1. **Cách** **thức** **xác** **thực** **và** **bảo** **mật** **trong** **SSH**

### Cách xác thực trong SSH

SSH có nhiều các xác thực một người dùng, nhưng 2 cách thông dụng nhất là xác thực bằng mật khẩu và xác thực bằng public key.

* **Xác** **thực** **bằng** **mật** **khẩu:** Xác thực dựa trên mật khẩu đơn giản là bạn chỉ việc sử dụng mật khẩu của user bạn tạo để truy cập, server sẽ lưu chúng, và đối chiếu với mật khẩu của bạn khi đăng nhập. Cách này thì không đủ an toàn do bạn có khả năng bị đánh cắp mật khẩu.
* **Xác** **thực** **bằng** **public-key:** Cách này sử dụng một cặp khóa – public-key và private-key – được tạo ra dựa trên thuật toán mã hóa public-key. Cặp khóa sau khi được tạo ra từ một máy tính, ta sẽ lấy public-key lưu vào server, khi truy cập ta sẽ dựa vào private-key lưu trên máy local và đặc tính liên quan mật thiết tới nhau của chúng để thiết lập kết nối. Kiểu xác thực này còn cho cho phép chúng ta thiết lập một kết nối an toàn một cách tự động hóa (automation).

### Bảo mật trong SSH

**Mã** **Hóa:** SSH hỗ trợ nhiều loại thuật toán mã hóa public-key

* + **rsa**: thuật toán được sử dụng nhiều nhất, ra đời từ năm 1977 dựa trên sự phức tạp của việc phân tích thừa số nguyên tố. Khi sử dụng nên kèm theo kích thước của khóa ít nhất là 2048 bits, tốt nhất nên là 4096 bits.
  + **dsa**: thuật toán dựa trên tính phức tạp của việc tính toán logarit rời rạc. Đã bị loại bỏ ở OpenSSH version 7 vì lý do bảo mật.
  + **ecdsa**: thuật toán dựa trên toạ độ của các điểm dựa trên đường cong

Elliptic. Có thể thay thế cho RSA bởi mức an toàn và tốc độ xử lý cao

16

hơn, kèm theo đó là việc sử dụng khoá có độ dài nhỏ hơn so với RSA. Từ đó làm tăng tốc độ xử lý một cách đáng kể. Chỉ hỗ trợ với 3 loại kích thước khóa: 256, 384, and 521 bits. Để an toàn nhất thì nên sử dụng 521 bits !

* + **ed25519**: một thuật toán được thêm vào OpenSSH từ version 6.5. Là bản cải tiến của ECDSA, cung cấp bảo khả năng mật tốt hơn với hiệu suất nhanh hơn so với DSA hoặc ECDSA. Chưa thực sự phổ cập trên toàn thế giới.

**Bảo** **Mật** **Cổng:** SSH thường chạy trên cổng mạng 22. Điều này đảm bảo rằng máy chủ SSH chỉ nghe trên cổng này và giúp ngăn chặn các cuộc tấn công từ các cổng khác

### Các lệnh cơ bản

Thực hiện kết nối:

*ssh* *username@hostname*

Trong đó:

* *‘username’* là tên người dùng của bạn trên máy chủ FTP.
* *‘hostname’* là tên hoặc địa chỉ IP của máy chủ FTP. Một số lệnh:

|  |  |
| --- | --- |
| **Lệnh** | **Mô** **tả** |
| *ls* | Liệt kê các tệp và thư mục trên máy từ xa. |
| *cd* | Di chuyển đến thư mục trên máy từ xa. |
| *pwd* | Hiển thị đường dẫn hiện tại trên máy từ xa. |
| *mkdir* | Tạo một thư mục trên máy từ xa. |
| *touch* | Tạo một tệp trên máy từ xa. |
| *cp* | Sao chép file/folder |
| *mv* | di chuyển hoặc đổi tên file/folder |
| *rm* *[remote* *file]* | Xóa tệp hoặc thư mục trên máy từ xa. |

17

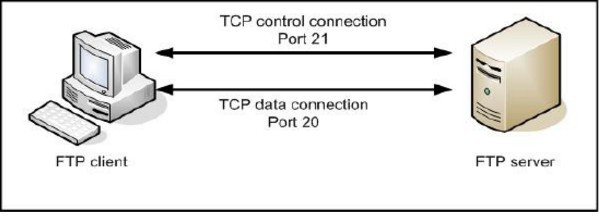
|  |  |
| --- | --- |
| *nano* | Biên tập file (text editor) |
| *cat* | Xem nội dung file |
| *find* | Tìm file và thư mục |
| *logout* *hoặc* *exit* | Thoát khỏi phiên SFTP. |

## FTP

### Khái niệm FTP

**FTP** - File Transfer Protocol (Giao thức truyền tải tập tin) được dùng trong việc trao đổi dữ liệu trong mạng thông qua giao thức TCP/IP, thường hoạt động trên 2 cổng là 20 và 21. Với Giao thức này, các máy client trong mạng có thể truy cập đến máy chủ FTP để gửi hoặc lấy dữ liệu. Người dùng có thể truy cập vào máy chủ FTP để truyền và nhận dữ liệu dù ở xa

### Mô hình hoạt động

****

*Mô* *hình* *hoạt* *động* *của* *FTP*

Hoạt động dựa trên mô hình cơ bản của việc truyền và nhận dữ liệu từ máy Client đến máy Server. Quá trình truyền và nhận dữ liệu được tạo nên từ 2 tiến trình TCP logic là Control Connection và Data Connection.

* + - * **Control** **Connection**: Thực hiện trên cổng 21. Đây là phiên làm việc TCP logic đầu tiên được tạo khi quá trình truyền dữ liệu bắt đầu. Tuy nhiên, tiến trình này chỉ kiểm soát các thông tin điều khiển đi qua nó, ví dụ các tập lệnh. Quá trình này được duy trì trong suốt quá trình làm việc diễn ra.

18

* + - * **Data** **Connection**: Thực hiện trên cổng 20. Được tạo ra với mục đích chuyên biệt là truyền tải dữ liệu giữa máy Client và máy Server. Kết nối sẽ tự động ngắt khi quá trình truyền tải dữ liệu hoàn tất.

### Các lệnh cơ bản

Thực hiện kết nối:

*ftp* *hostname* hoặc *ftp* *-u* *username* *-p* *password* *hostname*

Trong đó:

* *‘username’* là tên người dùng của bạn trên máy chủ FTP.
* *‘password’* là mật khẩu của bạn trên máy chủ FTP.
* *‘hostname’* là tên hoặc địa chỉ IP của máy chủ FTP. Một số lệnh:

|  |  |
| --- | --- |
| **Lệnh** | **Mô** **tả** |
| *ls* | Liệt kê các tệp và thư mục trên máy từ xa. |
| *cd* | Di chuyển đến thư mục trên máy từ xa. |
| *df* *[path]* | Hiển thị thông tin về không gian đĩa trên máy từ xa. |
| *pwd* | Hiển thị đường dẫn hiện tại trên máy từ xa. |
| *mkdir* | Tạo một thư mục trên máy từ xa. |
| *rename* *[oldpath]* *[newpath]* | Đổi tên hoặc di chuyển tệp/thư mục trên máy từ xa. |
| *delete* *[remote* *file]* | Xóa tệp hoặc thư mục trên máy từ xa. |
| *get* *[remote]* *[local]* | Tải tệp từ máy từ xa về máy cục bộ. |
| *put* *[local]* *[remote]* | Tải tệp từ máy cục bộ lên máy từ xa. |
| *quit* *hoặc* *bye* | Thoát khỏi phiên SFTP. |

## SFTP

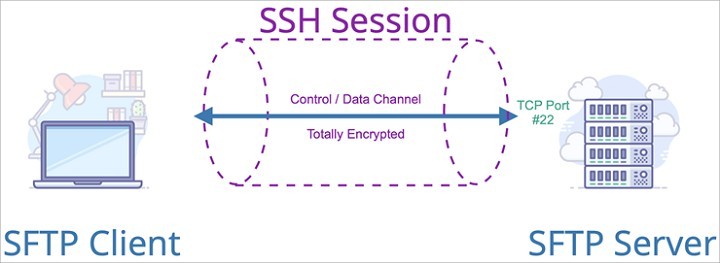
### Khái niệm SFTP

**SFTP** (Secure File Transfer Protocol, hoặc là SSH File Transfer Protocol) là một giao thức mạng giúp bạn có thể upload hoặc download dữ liệu

19

trên máy chủ. Đồng thời, có thể sử dụng giao thức này để sửa chữa, tạo hoặc xóa các tập tin, thư mục trên Linux.

### Mô hình hoạt động

****

*Mô* *hình* *hoạt* *động* *của* *SFTP*

Kết nối giữa máy chủ và máy khách được hình thành thông qua giao thức SSH, giao thức này an toàn. Trước khi truyền, giao thức truyền tệp bảo mật thường mã hóa thông tin và dữ liệu. Thường sử dụng cổng 22 để thực hiện các chức năng của nó.

Khái niệm về SFTP dựa trên “phương pháp tạo đường hầm”, đặc biệt hữu ích khi truyền tệp qua các trang web. Trong trường hợp của SFTP, cả quá trình tải lên và tải xuống các tệp đều được duy trì bằng cách sử dụng khóa SSH theo cách rất an toàn.

### Các lệnh cơ bản

Thực hiện kết nối:

*sftp* *username@hostname*

Trong đó:

* *‘username’* là tên người dùng của bạn trên máy chủ FTP.
* *‘hostname’* là tên hoặc địa chỉ IP của máy chủ SFTP. Một số lệnh:

**Mô** **tả**

**Lệnh**

20

|  |  |
| --- | --- |
| *ls* | Liệt kê các tệp và thư mục trên máy từ xa. |
| *cd* | Di chuyển đến thư mục trên máy từ xa. |
| *df* *[path]* | Hiển thị thông tin về không gian đĩa trên máy từ xa. |
| *pwd* | Hiển thị đường dẫn hiện tại trên máy từ xa. |
| *mkdir* | Tạo một thư mục trên máy từ xa. |
| *rename* *[oldpath]* *[newpath]* | Đổi tên hoặc di chuyển tệp/thư mục trên máy từ xa. |
| *rm* *[remote* *file]* | Xóa tệp hoặc thư mục trên máy từ xa. |
| *get* *[remote]* *[local]* | Tải tệp từ máy từ xa về máy cục bộ. |
| *put* *[local]* *[remote]* | Tải tệp từ máy cục bộ lên máy từ xa. |
| *exit* *hoặc* *quit* | Thoát khỏi phiên SFTP. |

* 1. **So** **Sánh** **FTP** **và** **SFTP**

Để truyền tệp thành công, giao thức truyền tệp hoặc FTP không thể cung cấp kênh an toàn giữa các máy chủ. Mặt khác, để truyền tệp thành công giữa các máy chủ, SFTP có khả năng cung cấp một kênh bảo mật cao.

FTP là một cổng không an toàn liên quan đến giao thức IP hoặc TCP. Mặt khác, SFTP là một cổng bảo mật liên quan đến giao thức SSH, một chương trình ứng dụng có đăng nhập từ xa.

Trong trường hợp FTP, kiểm soát cao hơn được thiết lập trên kết nối cổng TCP 21. Mặt khác, trong trường hợp SFTP, kết nối được thiết lập giữa máy chủ và máy khách được thực hiện thông qua giao thức SSH một cách an toàn.

Giao thức truyền tệp không an toàn chỉ sở hữu mật khẩu FTP, nghĩa là nó không hỗ trợ mã hóa dữ liệu trước khi gửi. Tuy nhiên, ở đây thông tin và dữ liệu được gửi dưới dạng văn bản thuần túy. Mặt khác, giao thức truyền tệp được bảo mật thường mã hóa thông tin và dữ liệu trước quá trình gửi.

21

Giao thức truyền tệp hoặc FTP thường sử dụng cổng số 21 để thực hiện các hoạt động của nó. Mặt khác, trong trường hợp giao thức truyền tệp an toàn hoặc SFTP, nó thường sử dụng cổng số 22 để thực hiện các hoạt động của mình.

Khái niệm về FTP dựa trên phương pháp trực tiếp khi quá trình truyền tệp có liên quan. Mặt khác, khái niệm về SFTP dựa trên “phương pháp tạo đường hầm”, đặc biệt khi quá trình truyền tệp có liên quan giữa các máy chủ khác nhau.

Trong trường hợp của FTP, cả hai quá trình tải xuống tệp cùng với tải tệp lên đều diễn ra mà không có sự tham gia của bất kỳ biện pháp bảo mật nào. Mặt khác, trong trường hợp SFTP, cả quá trình tải lên và tải xuống tệp đều được duy trì với sự trợ giúp của khóa SSH, một cách rất an toàn. Dưới đây là bảng so sánh vắn tắt giữa FTP và SFTP:

|  |  |  |
| --- | --- | --- |
| **Thông** **số** **so** **sánh** | **FTP** | **SFTP** |
| Kiểu | Không được bảo mật | Bảo mật |
| Cổng được sử dụng | Cổng 20 và 21 | Cổng 22 |
| Giao thức theo sau | IP hoặc TCP | SSH |
| Loại phương pháp | Phương pháp trực tiếp | Phương pháp đào hầm |
| Mã hoá | Không được mã hoá | Được mã hoá |

22

# CHƯƠNG 4. TRIỂN KHAI DỰ ÁN

### Giới thiệu

Ở *chương* *4* này sẽ tập trung vào quá trình và công cụ phát triển phần mềm, đặc biệt là việc đưa project *“website* *quản* *lý* *và* *đặt* *phòng* *khách* *sạn”* lên máy ảo và quản lý nó trên một máy chủ. Trong quá trình này, chúng tôi đã học cách sử dụng một loạt công cụ quan trọng như Git, SVN, và SFTP để giúp các thành viên trong dự án có thể truy cập và lập trình trên máy chủ chung.

Chúng tôi sẽ trình bày cụ thể về quá trình triển khai và quản lý dự án trên máy ảo, bao gồm cách chúng tôi thiết lập môi trường phát triển, sao lưu và khôi phục dữ liệu, quản lý phiên bản, và cách tương tác với các thành viên trong nhóm. Chúng tôi cũng sẽ chia sẻ những kiến thức và kinh nghiệm thu được từ việc này, đặc biệt là về cách làm việc trong môi trường phát triển phần mềm thực tế tại các công ty.

### Cách triển khai

1. **CÀI** **ĐẶT** **MÁY** **ẢO** **VỚI** **HỆ** **ĐIỀU** **HÀNH** **LINUX**

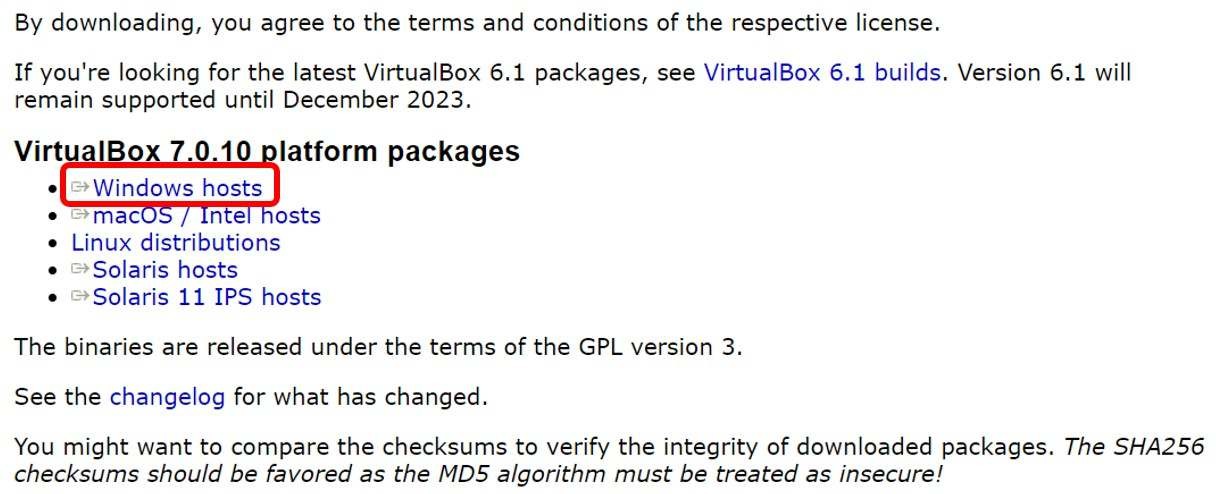
### Tải VirtualBox

**Bước** **1:** Truy cập *https://[www.virtualbox.org/wiki/Downloads](http://www.virtualbox.org/wiki/Downloads)* bằng trình duyệt Internet trên máy tính của bạn. Đây là trang web mà bạn sẽ tải xuống tệp cài đặt VirtualBox.



23

**Bước** **2:** Ở bên dưới tiêu đề "**VirtualBox** **7.0.10** **platform** **packages**", chọn **Windows** **hosts**. Tệp **VirtualBox** **EXE** sẽ bắt đầu tải xuống máy tính của bạn.



**Bước** **3:** Mở tệp **VirtualBox** **EXE**. Đi tới vị trí chứa tệp EXE đã tải xuống và bấm đúp vào tệp. Làm như vậy sẽ mở cửa sổ cài đặt VirtualBox.



**Bước** **4:** Điều hướng qua các lời nhắc cài đặt. Thực hiện như sau:

* Nhấp vào **Next** trên ba trang đầu tiên.
* Bấm vào **Yes** khi được nhắc.
* Nhấp vào **Install**
* Bấm vào **Yes** khi được nhắc.

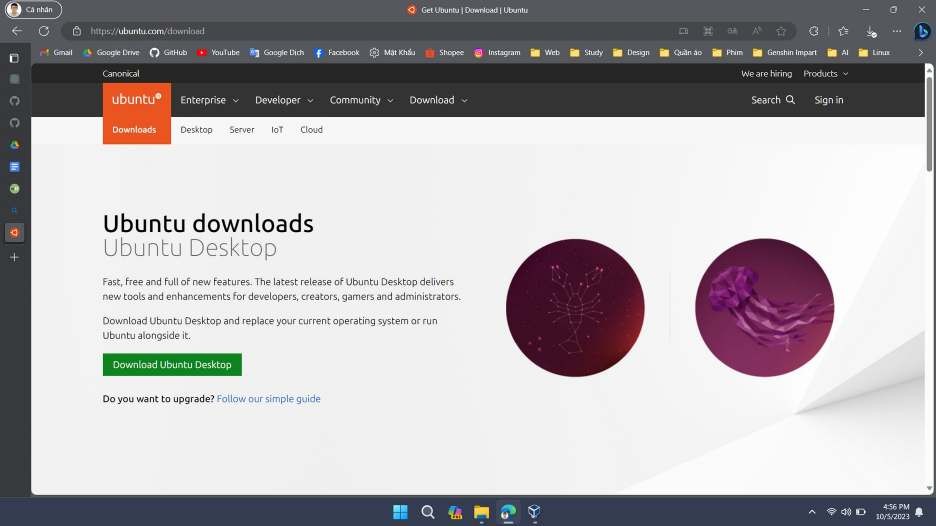
24



**Bước** **5:** Nhấp vào **Finish** khi được nhắc. Nó ở phía dưới bên phải của cửa sổ. Làm như vậy sẽ đóng cửa sổ cài đặt và mở VirtualBox. Bây giờ bạn đã cài đặt và mở VirtualBox, bạn có thể tạo một máy ảo để chạy bất kỳ hệ điều hành nào trên PC của mình. Đảm bảo rằng bạn không bỏ chọn button *“Start”* trước khi thực hiện việc này.

* 1. **Cài** **đặt** **máy** **ảo** **với** **hệ** **điều** **hành** **Ubuntu**

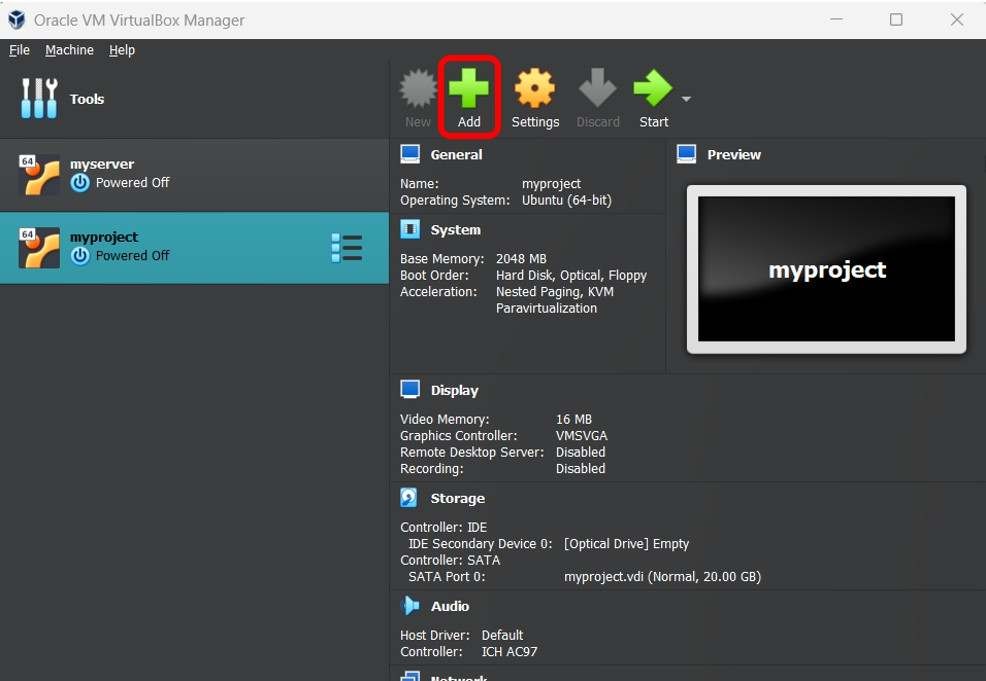
**Bước** **1:** Truy cập trang web với đường dẫn *https://ubuntu.com/download*, sau đó lựa chọn phiên bản ubuntu muốn cài.



**Bước** **2:** Sau khi đã lựa chọn phiên bản và tải xuống ubuntu, mở ứng dụng

**VirtualBox.exe**, tại giao diện trang chủ chọn **New** để cài đặt máy ảo.

25



**Bước** **3:** Khi hiển thị cửa sổ **Create** **Virtual** **Machine**, nhập Name và nhập địa chỉ lưu trữ file đã tải xuống ở bước 1. Sau đó chọn **Next** để tiếp tục.

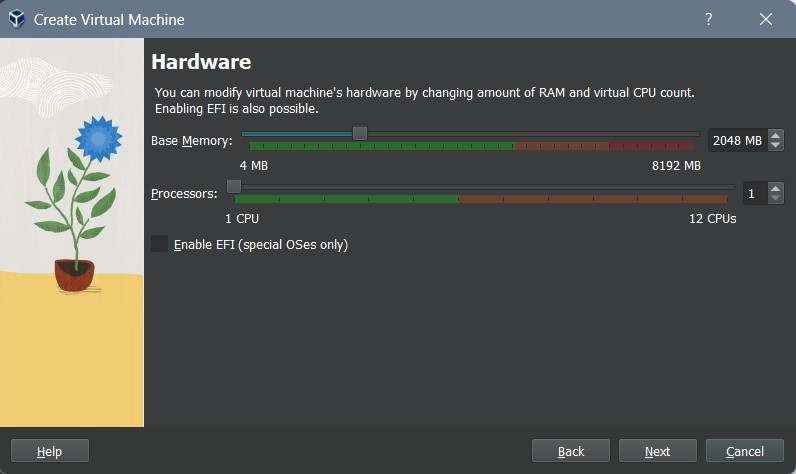


**Bước** **4:** Tại cửa sổ **Unattended** **Guest** **OS** **Install** **Setup,** nhập Username và Password để thiết lập User. Sau đó, chọn **Next** để tiếp tục.

26



**Bước** **5:** Tại cửa sổ **Hardware**, bạn có thể sửa đổi phần cứng của máy ảo bằng cách thay đổi dung lượng RAM và số lượng CPU ảo.

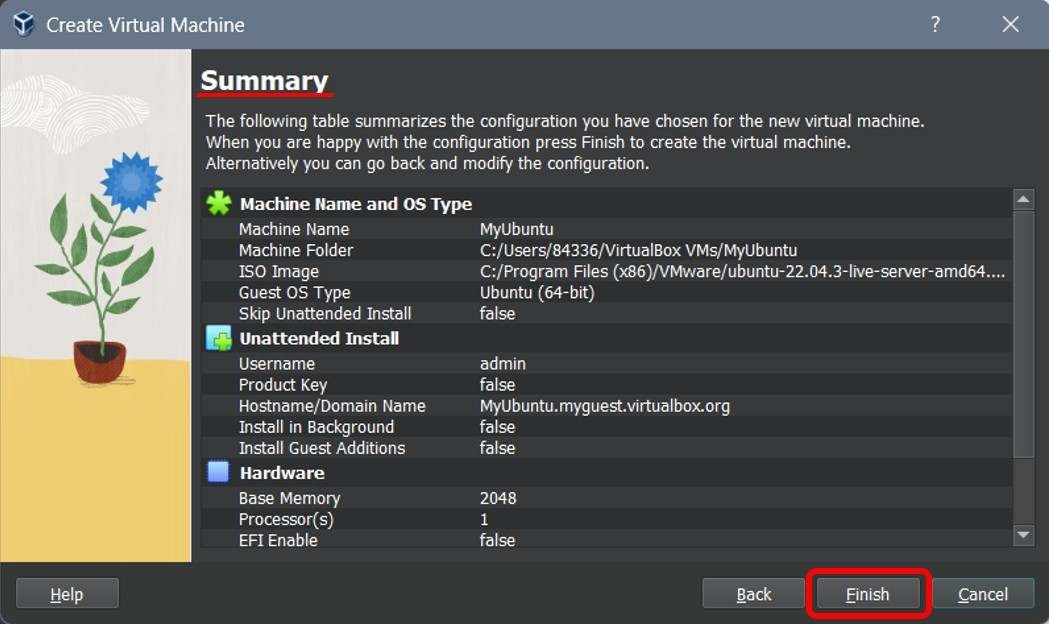


**Bước** **6:** Tại cửa số **Virtual** **Hard** **disk**, bạn có thể thêm đĩa cứng ảo vào máy mới. Bạn có thể tạo một tệp đĩa cứng mới hoặc chọn một tệp hiện có. Ngoài ra, bạn có thể tạo một máy ảo mà không cần ổ cứng ảo.

27



**Bước** **7:** Tại cửa sổ **Summary**, chọn Finish để kết thúc quá trình tạo máy ảo với hệ điều hành Ubuntu.



28

## CÀI ĐẶT MÔI TRƯỜNG

* 1. **Cài** **đặt** **và** **cấu** **hình** **SSH**

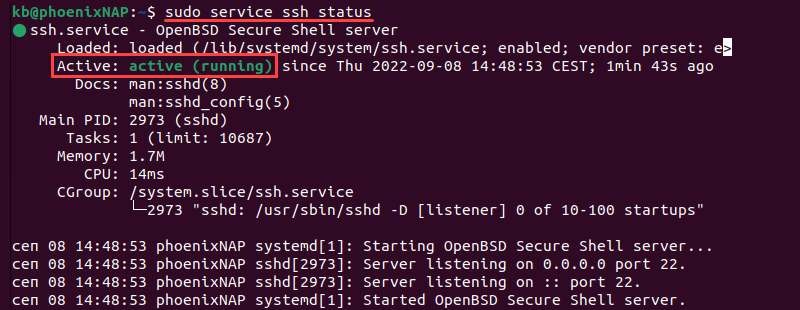
### Cài đặt

Thực hiện lần lượt các lệnh sau trên terminal của máy ảo:

* + - * *`sudo* *apt-get* *update`* (cập nhật danh sách gói phần mềm)
      * *`sudo* *apt-get* *install* *openssh-server* *-y`* (cài đặt dịch vụ SSH)
      * `*sudo* *systemctl* *start* *sshd`* (bắt đầu dịch vụ ssh trên hệ thống sử dụng hệ điều hành systemd)
      * *`sudo* *systemctl* *enable* *ssh`* (tự động khởi động SSH cùng với hệ thống khi được khởi động lại)

Để kiểm tra đã cài đặt thành công và SSH đang chạy

* + - * *`sudo* *service* *ssh* *status`*

**

### Cấu hình SSH

Mở file cấu hình: *`sudo* *nano* */etc/ssh/sshd\_config`*

Sau đó cấu hình lại như sau: Port 22

ListenAddress 0.0.0.0

HostKey /etc/ssh/ssh\_host\_rsa\_key HostKey /etc/ssh/ssh\_host\_ecdsa\_key HostKey /etc/ssh/ssh\_host\_ed25519\_key

29

Restart lại dịch vụ SSH để xác nhận sửa đổi:

`*sudo* *systemctl* *restart* *sshd.service`*

### Cấu hình tường lửa

Cấu hình chính sách tường lửa trên hệ thống Linux bằng `ufw` (Uncomplicated Firewall) để cho phép kết nối SSH đến máy chủ:

*`sudo* *ufw* *allow* *from* *any* *to* *any* *port* *22* *proto* *tcp`*

* 1. **Cài** **đặt** **và** **thiết** **lập** **Git**

### Cài đặt Git

Thực hiện lần lượt các lệnh sau trên terminal của máy ảo:

* + - *`sudo* *apt-get* *update`* (cập nhật danh sách gói phần mềm trên hệ thống)
    - *`sudo* *apt-get* *install* *git`* (cài đặt Git trên hệ thống)
    - *`git* *--version`* (Kiểm tra phiên bản Git hiện tại và xem nó đã được cài đặt thành công hay chưa)

### Thiết lập Git

Sử dụng các lệnh sau để thiết lập tài khoản Git trên máy ảo

* + - *`git* *config* *--global* *user.name* *"Tên\_của\_bạn"`* (thiết lập username mặc định của Git)
    - *`git* *config* *--global* *user.email* *"Địa\_chỉ\_email\_của\_bạn"`* (thiết lập email mặc định của Git)

Để kiểm tra username đang được sử dụng ta thực hiện:

*`git* *config* *user.name`*

Để kiểm tra email đang được sử dụng ta thực hiện:

*`git* *config* *user.email`*

Hoặc sử dụng lệnh `*git* *config* *–list`* để hiển thị cả 2 nội dung trên.

### Cài đặt MySQL

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở được sử dụng vô cùng rộng rãi hiện nay, tuân theo mô hình quan hệ và sử dụng SQL để quản lý dữ liệu. Dưới đây là các bước để cài đặt MySQL:

Cài đặt gói MySQL:

30

*`sudo* *apt* *install* *mysql-server* *-y`*

Kích hoạt và bắt đầu chạy MySQL trên hệ thống:

*`sudo* *systemctl* *start* *mysql.service`*

Kiểm tra xem dịch vụ MySQL hoạt động hay chưa:

*`sudo* *systemctl* *is-active* *mysql`*

Cài đặt user và passwork cho MySQL:

*`sudo* *mysql`*

*`ALTER* *USER* *root@localhost* *IDENTIFIED* *WITH* *mysql\_native\_password* *BY* *'admin';`*

### Cài đặt Apache2

Apache2 là một máy chủ web mã nguồn mở mạnh mẽ và phổ biến được sử dụng rộng rãi trong các dự án phát triển ứng dụng web. Với khả năng xử lý các yêu cầu HTTP, quản lý bảo mật, phân quyền, và hỗ trợ cho nhiều ngôn ngữ lập trình, Apache2 đóng vai trò quan trọng trong việc phục vụ và quản lý các trang web và ứng dụng web. Dưới đây là các bước cài đặt Apache2 để bắt đầu sử dụng nó trong dự án:

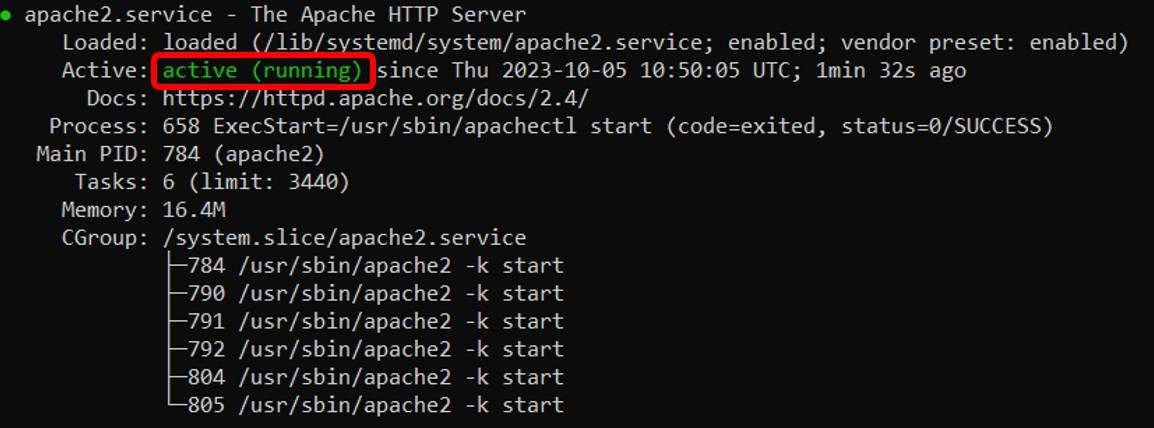
* Sử dụng lệnh sau để cài đặt apache2 trên máy ảo:

*`sudo* *apt* *update`*

*`sudo* *apt* *install* *apache2`*

* Kiểm tra trạng thái của Apache2:

*`sudo* *systemctl* *status* *apache2`*

**

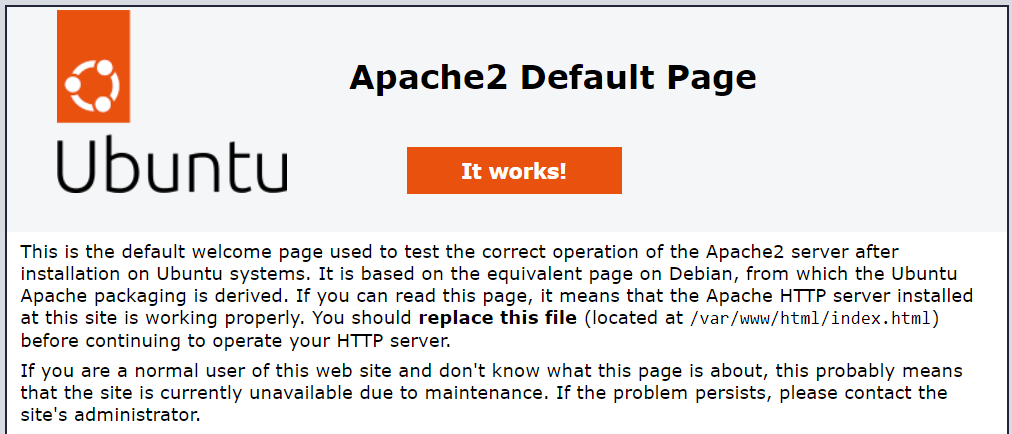
* Mở cổng port truy cập cho Apache2:

31

*`sudo* *ufw* *allow* *80/tcp`*

* Kiểm tra Apache2 bằng trình duyệt web, hãy thay đổi `*your\_serve\_ip`* bằng địa chỉ IP của máy ảo.

*`http://your\_server\_ip/`*

**

### Cài đặt PHP

- Cài đặt PHP và các gói liên quan:

*`sudo* *apt* *update`*

*`sudo* *apt* *install* *php* *libapache2-mod-php* *php-mysql`*

Trong đó:

+ `php`: Cài đặt gói chính của PHP.

+ `libapache2-mod-php`: Gói này liên kết PHP với Apache.

+ `php-mysql`: Gói này cho phép PHP làm việc với CSDL MySQL.

* Kiểm tra cài đặt PHP:

*`php* *-v`*

* Kiểm tra tệp cấu hình:

*`ls* */etc/apache2/mods-enabled/php\*.conf`*

* Khởi động lại Apache2:

*`sudo* *systemctl* *restart* *apache2`*

32

## ĐƯA PROJECT TRÊN MÁY ẢO

* 1. **Đưa** **file** **lên** **máy** **ảo** **và** **quản** **lý** **source** **bằng** **Git**

Chuyển vào đường dẫn /var/www để clone project vào:

*`cd* */var/www/`*

*`sudo* *git* *clone* *<URL\_project\_cua\_ban>`*

### Export file .sql

Thực hiện các lệnh sau

*`mysql* *-u* *root* *-p`* (truy cập vào mysql trên máy ảo)

*`CREATE* *DATABASE* *burninghotel;`*

*`USE* *burninghotel;`*

*`source* */duong/dan/toi/file.sql;`*

### Cấu hình Virtual Host

Phân quyền cho tệp và thư mục:

*`sudo* *chown* *-R* *www-data:www-data* */var/www/BurningHotel`*

Khởi động lại Apache:

*`sudo* *systemctl* *restart* *apache2*

Tạo một tệp cấu hình Virtual Host

*`sudo* *cp*

*/etc/apache2/sites-available/000-default.conf/etc/apache2/sites-* *available/burninghotel.conf`*

*`sudo* *nano* */etc/apache2/sites-available/burninghotel.conf`*

Tại đây, cấu hình Virtual Host như sau:

<VirtualHost \*:80>

ServerAdmin webmaster@localhost ServerName burninghotel.com DocumentRoot /var/www/BurningHotel

ErrorLog ${APACHE\_LOG\_DIR}/error.log CustomLog ${APAC HE\_LOG\_DIR}/access.log combined

33

</VirtualHost>

Kích hoạt Virtual Host và khởi động lại Apache:

*`sudo* *a2ensite* *burninghotel.conf`*

*`sudo* *systemctl* *restart* *apache2`*

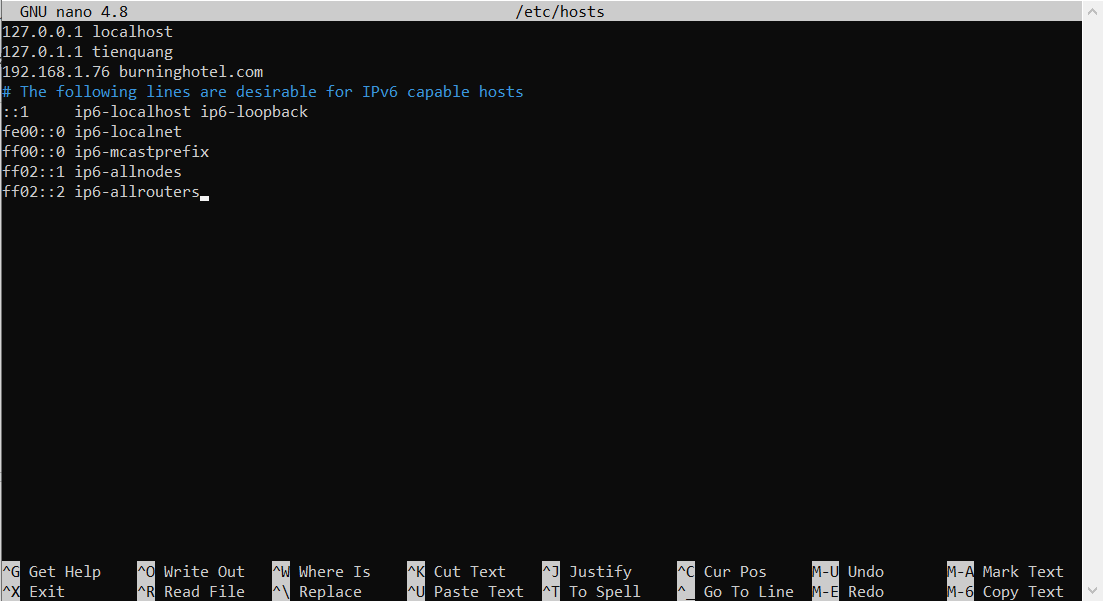
### Ánh xạ tên miền

Mở tệp /etc/hosts để chỉnh sửa:

*`sudo* *nano* */etc/hosts`*

Thêm các cặp tên miền và địa chỉ IP:

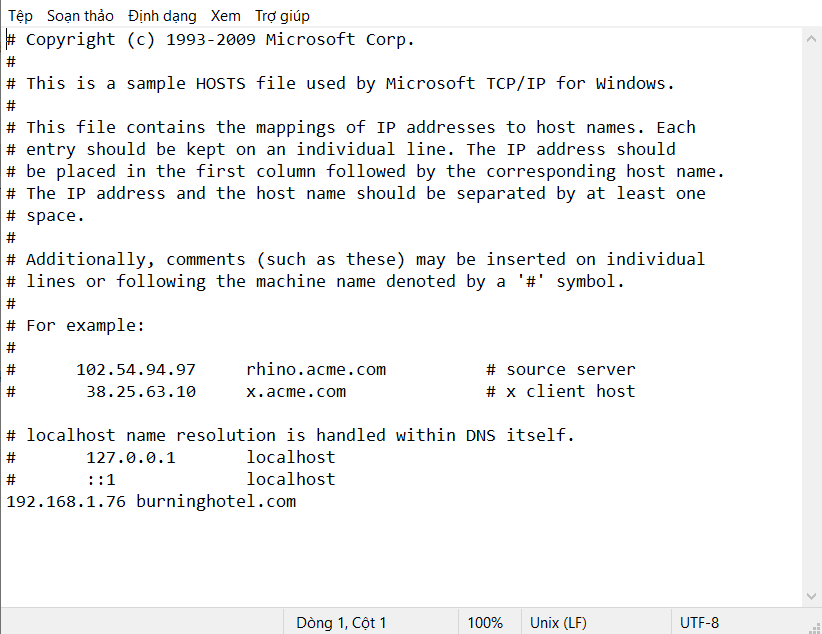
*`ip\_address* *burninghotel.com`*

**

### Cấu hình file hosts trên window

Tại đường dẫn *`C:\Windows\System32\drivers\etc`* thêm ip\_address và domain\_name vào file hosts.

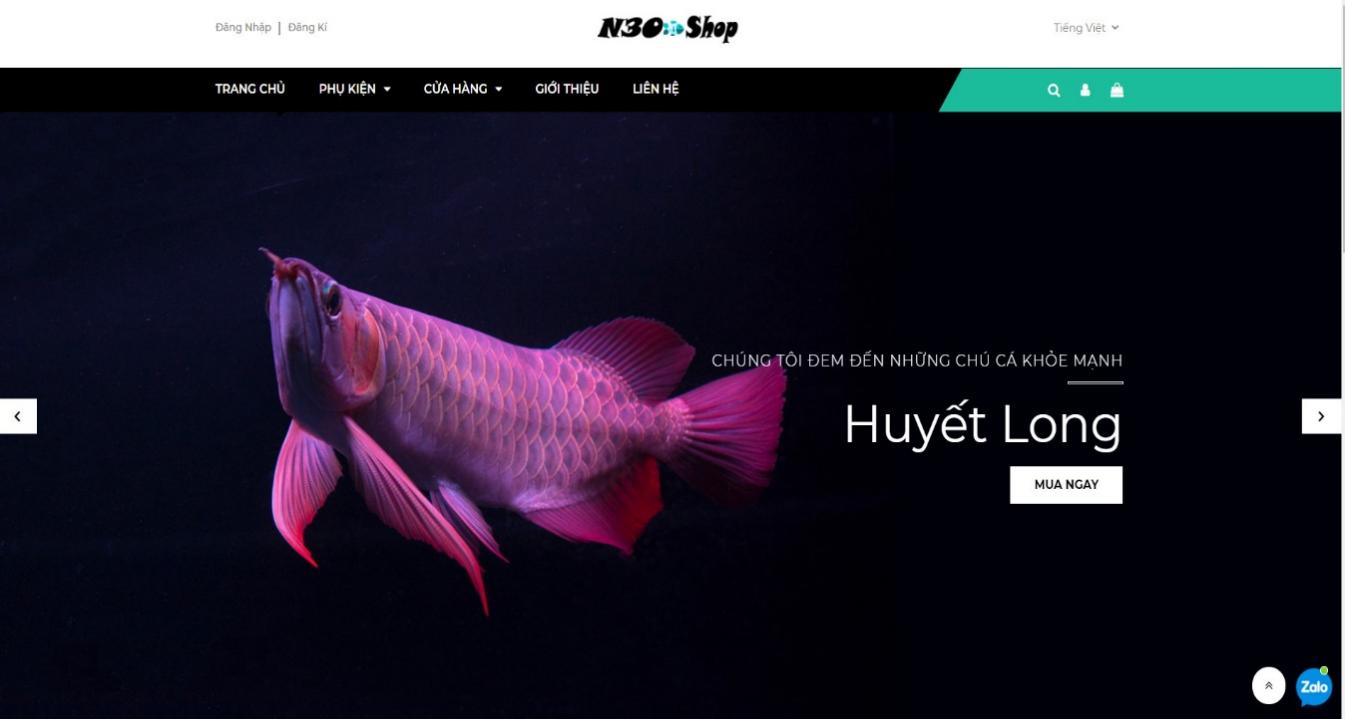
34



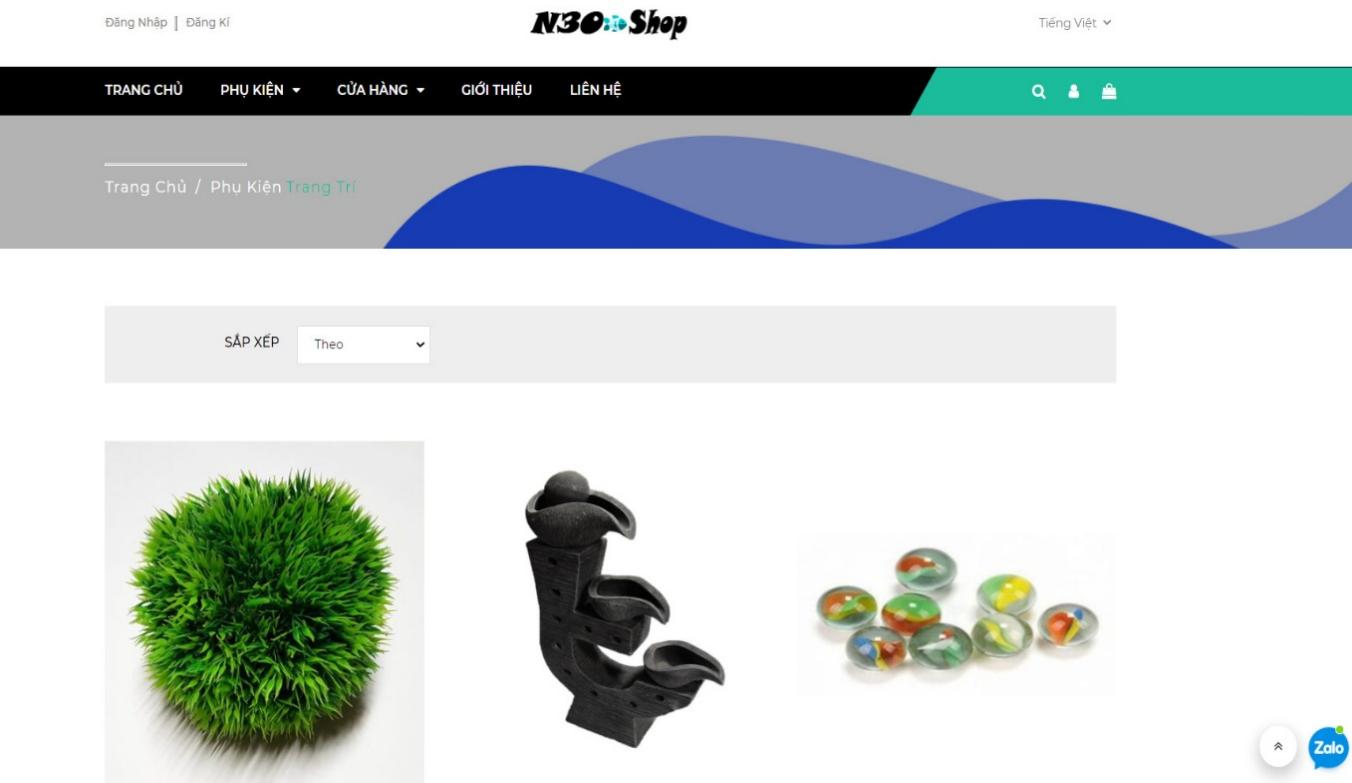
Từ đó, bạn có thể truy cập với tên miền của bạn trên trình duyệt Internet.

**CHƯƠNG 4. GIAO DIỆN**

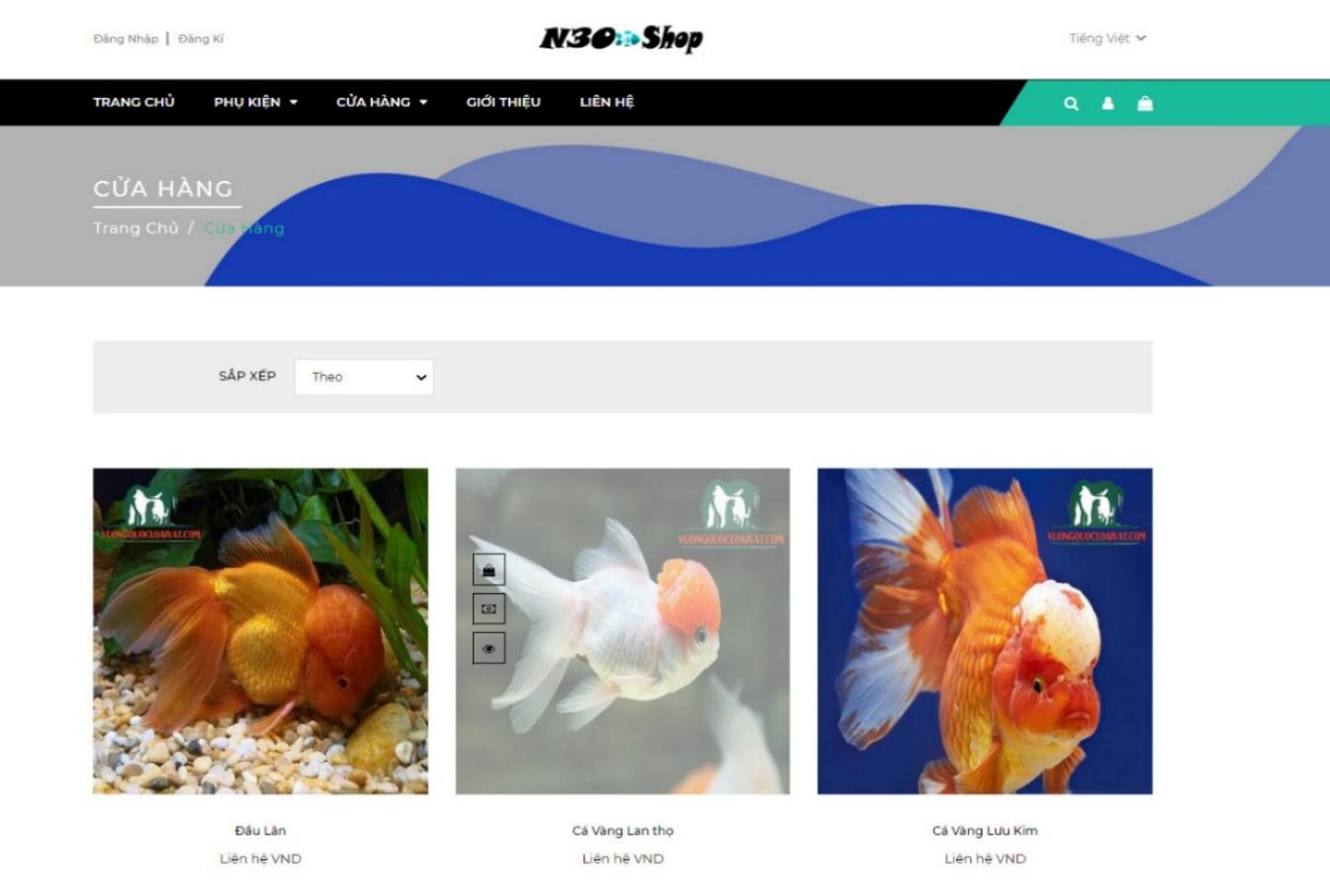
**1. Giao diện trang chủ**

****

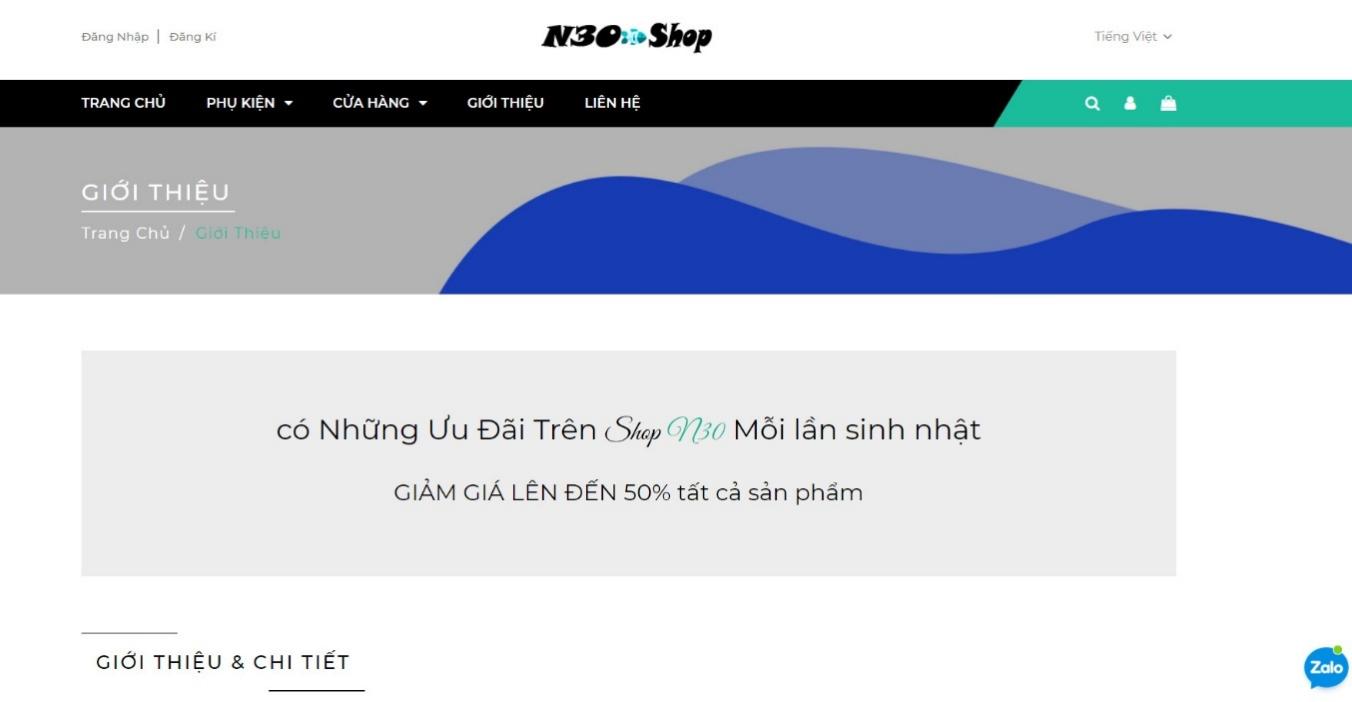
**2. Giao diện trang phụ kiện**

****

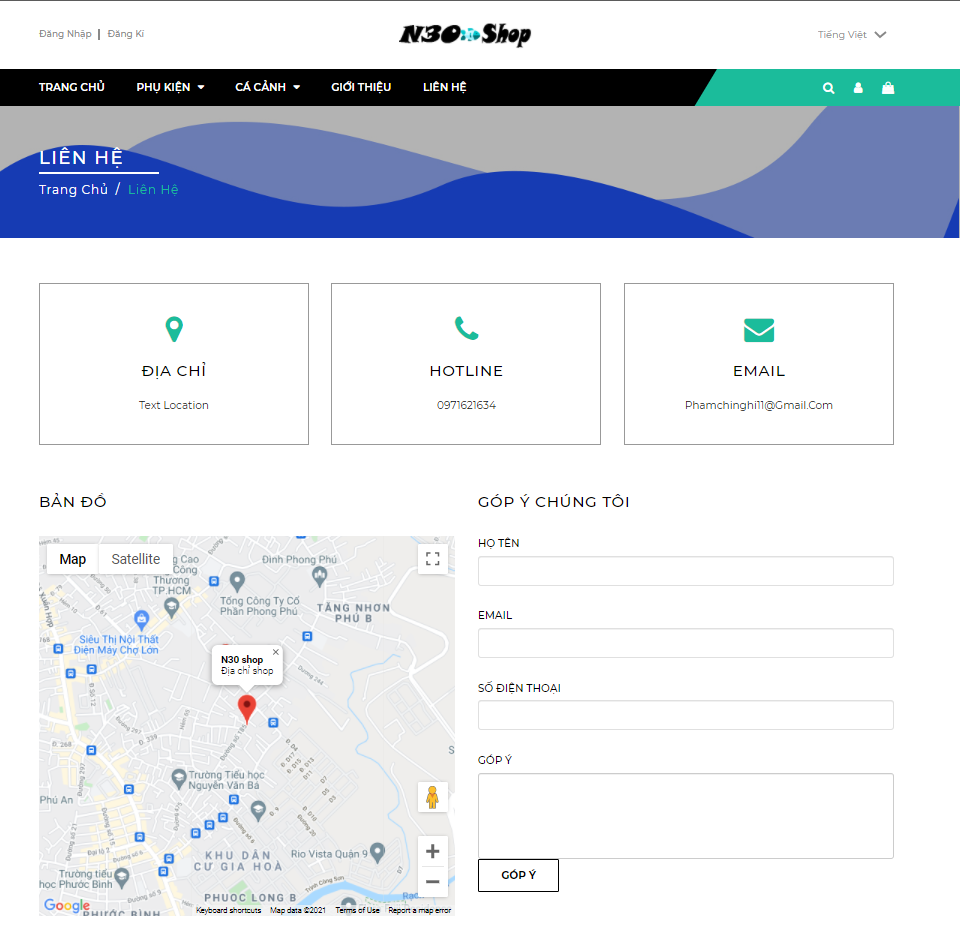
**3. Giao diện trang cửa hàng**

****

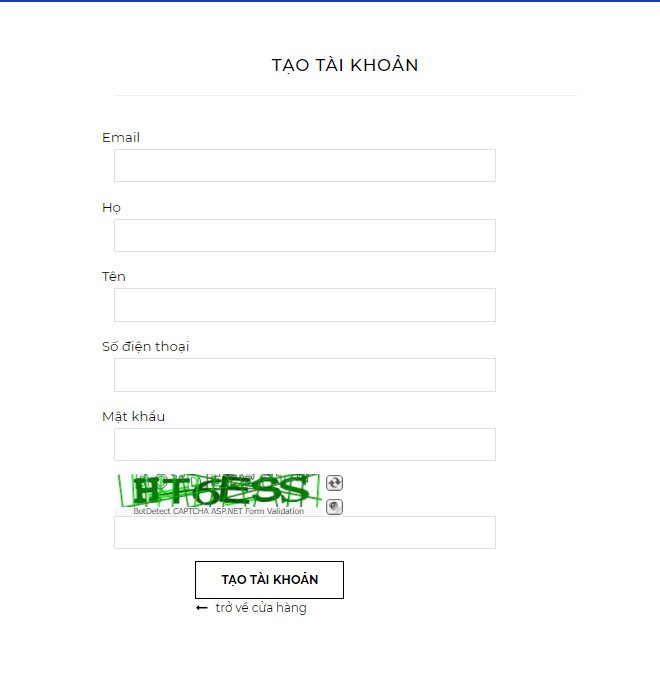
**4. Giao diện trang giới thiệu**

****

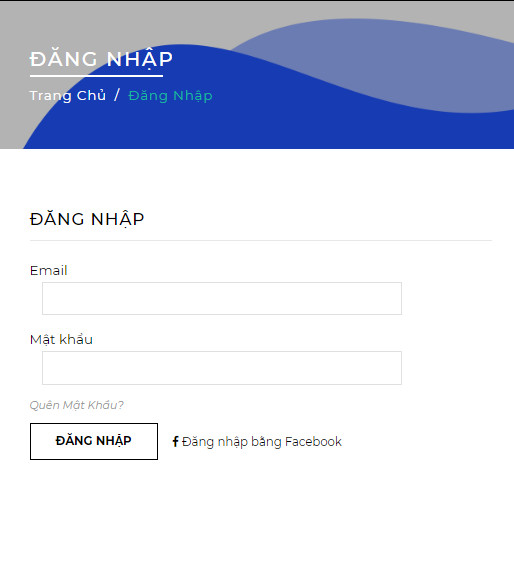
**5. Giao diện trang liên hệ**

****

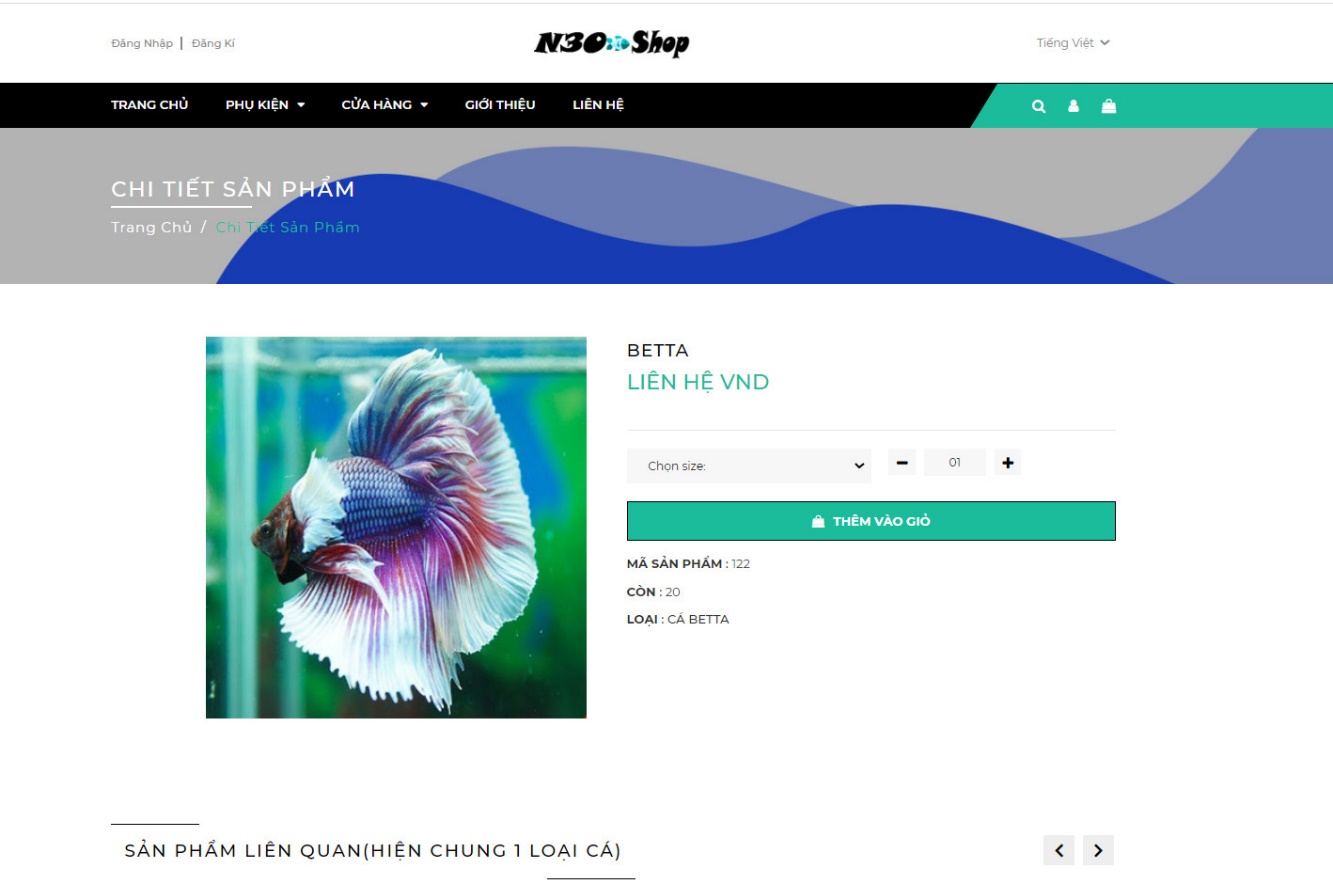
**6. Giao diện trang đăng kí**

****

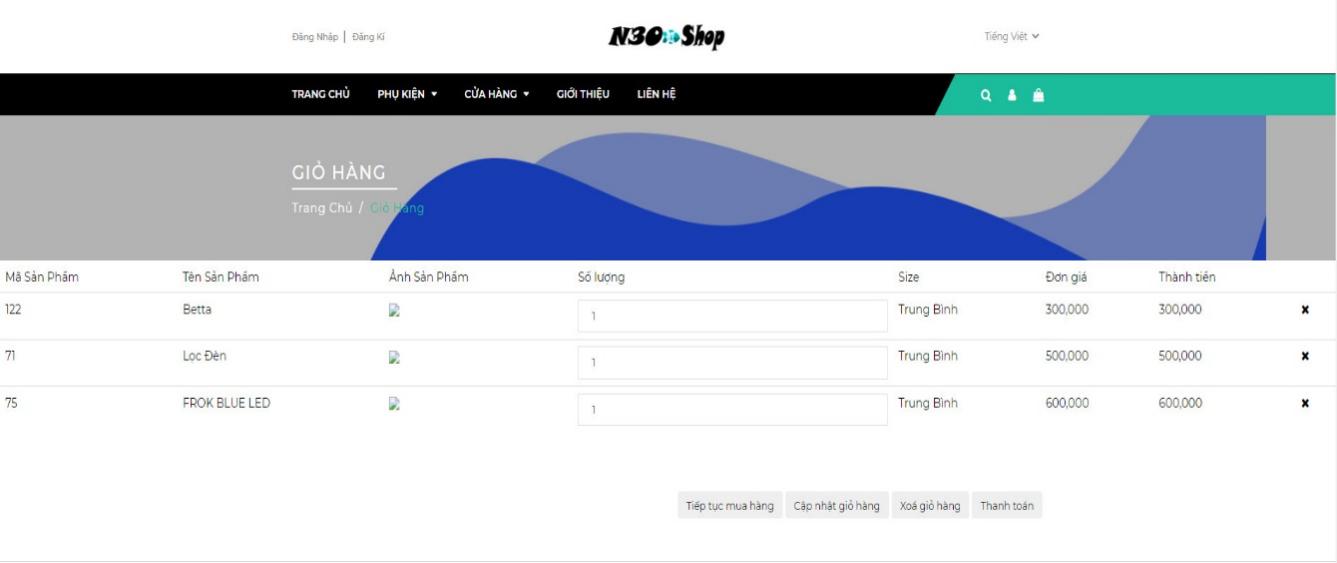
**7. Giao diện trang đăng nhập**

****

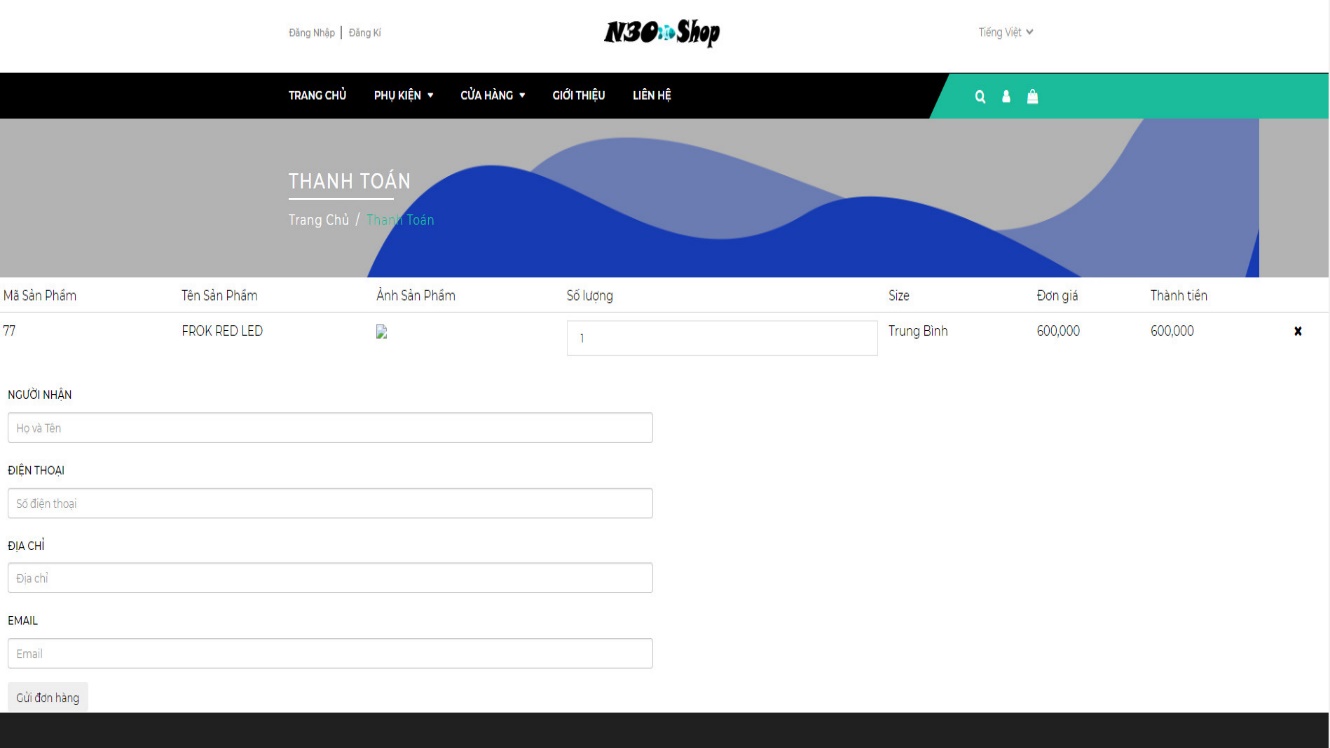
**8. Giao diện chi tiết sản phẩm**

****

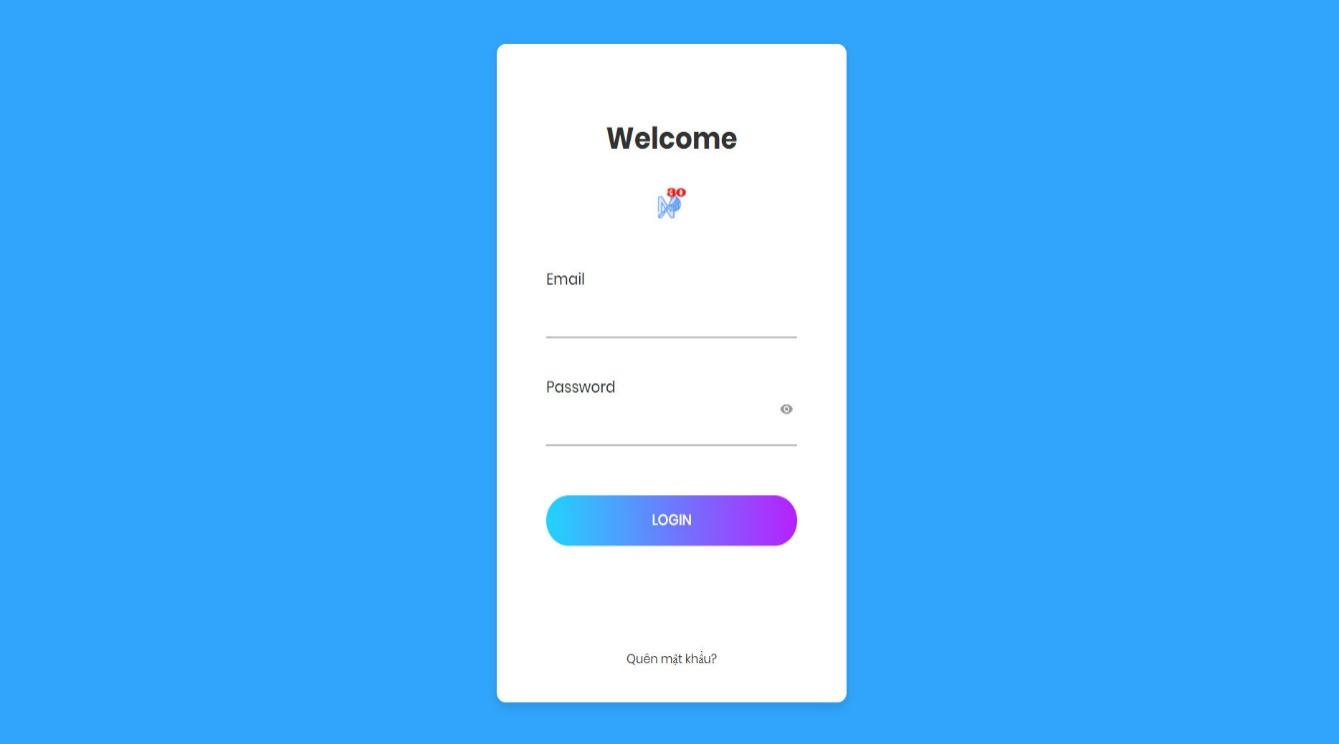
**9. Giao diện giỏ hàng**

****

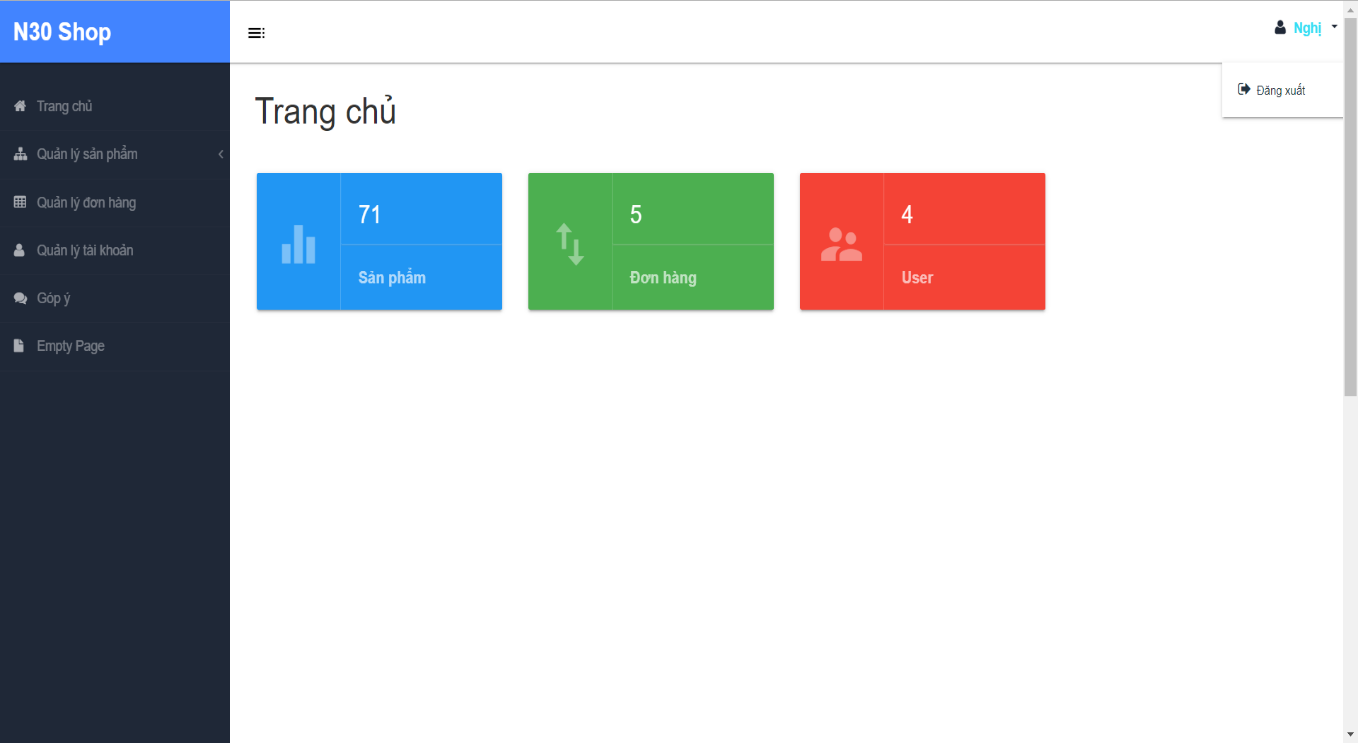
**10. Giao diện trang thanh toán**

****

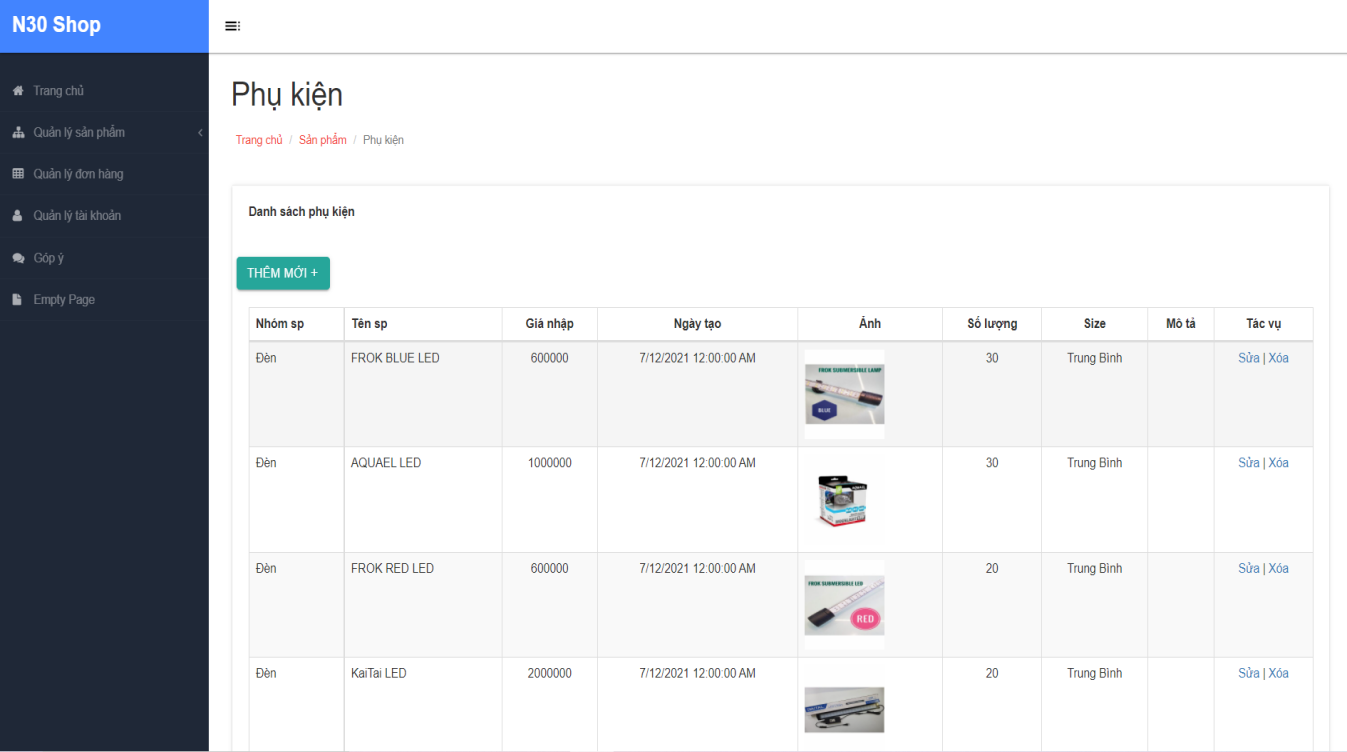
**11. Giao diện đăng nhập ADMIN**

****

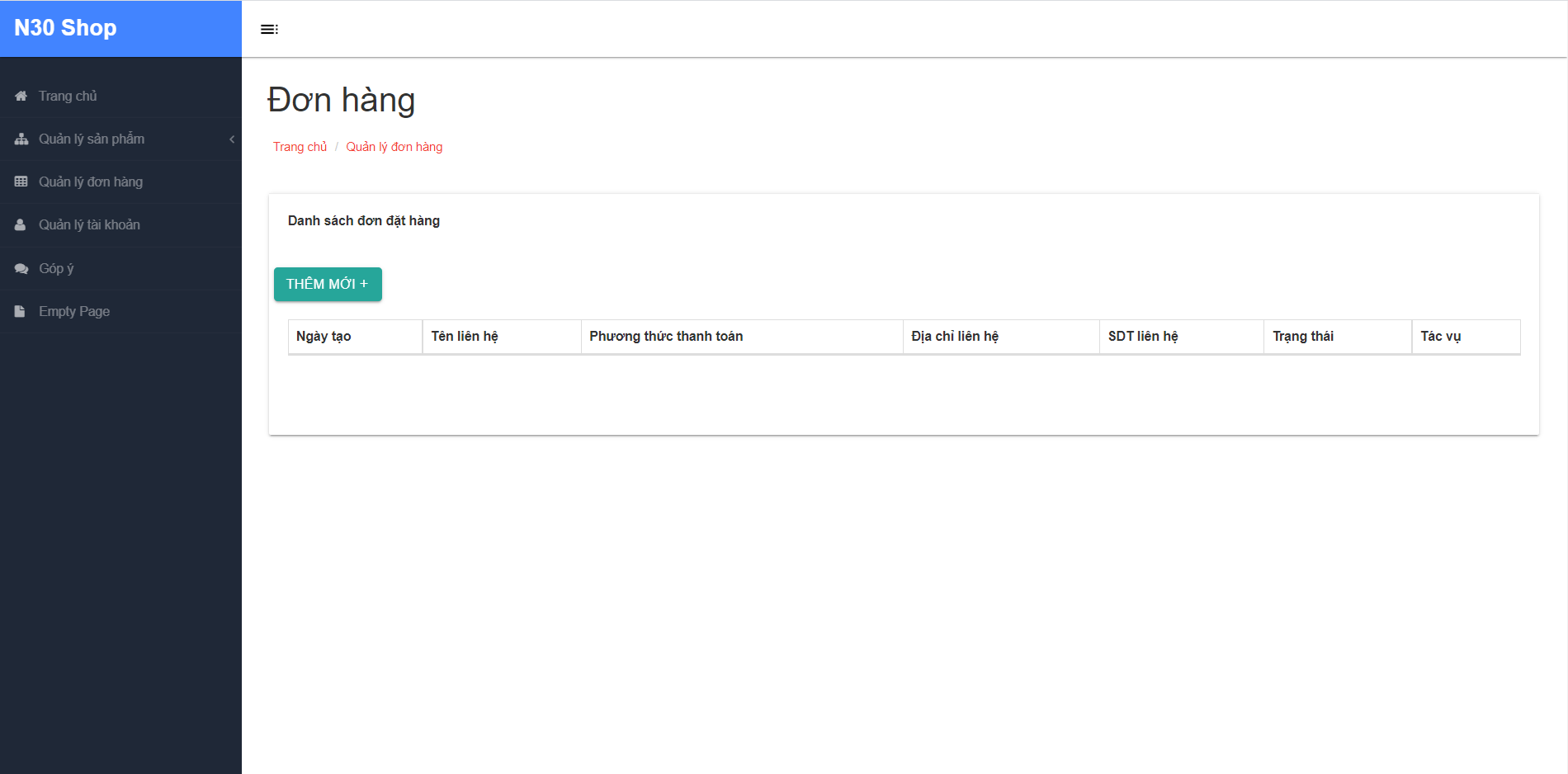
**12. Giao diện trang chủ ADMIN**

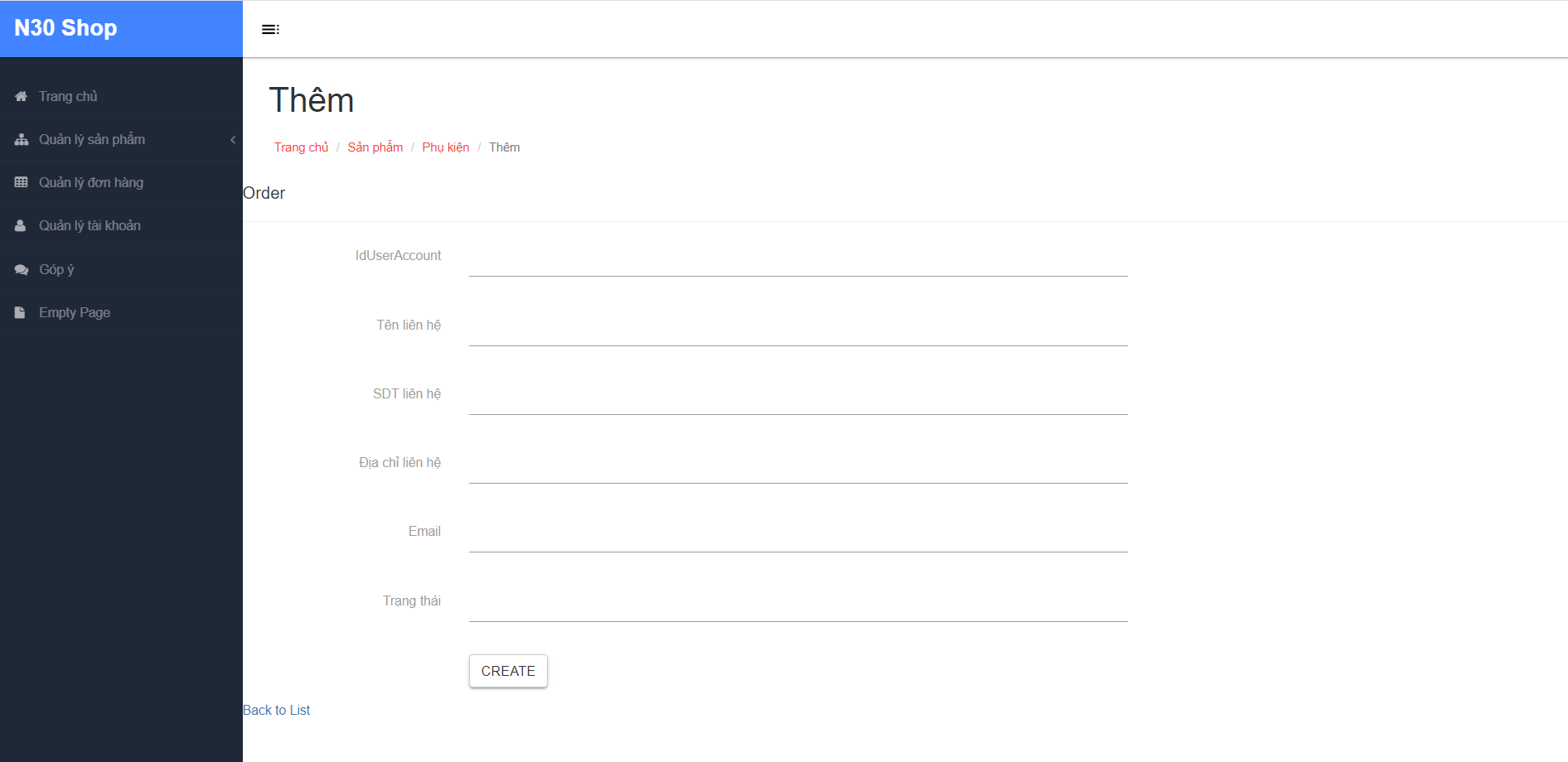
****

**13. Giao diện quản lý sản phẩm**

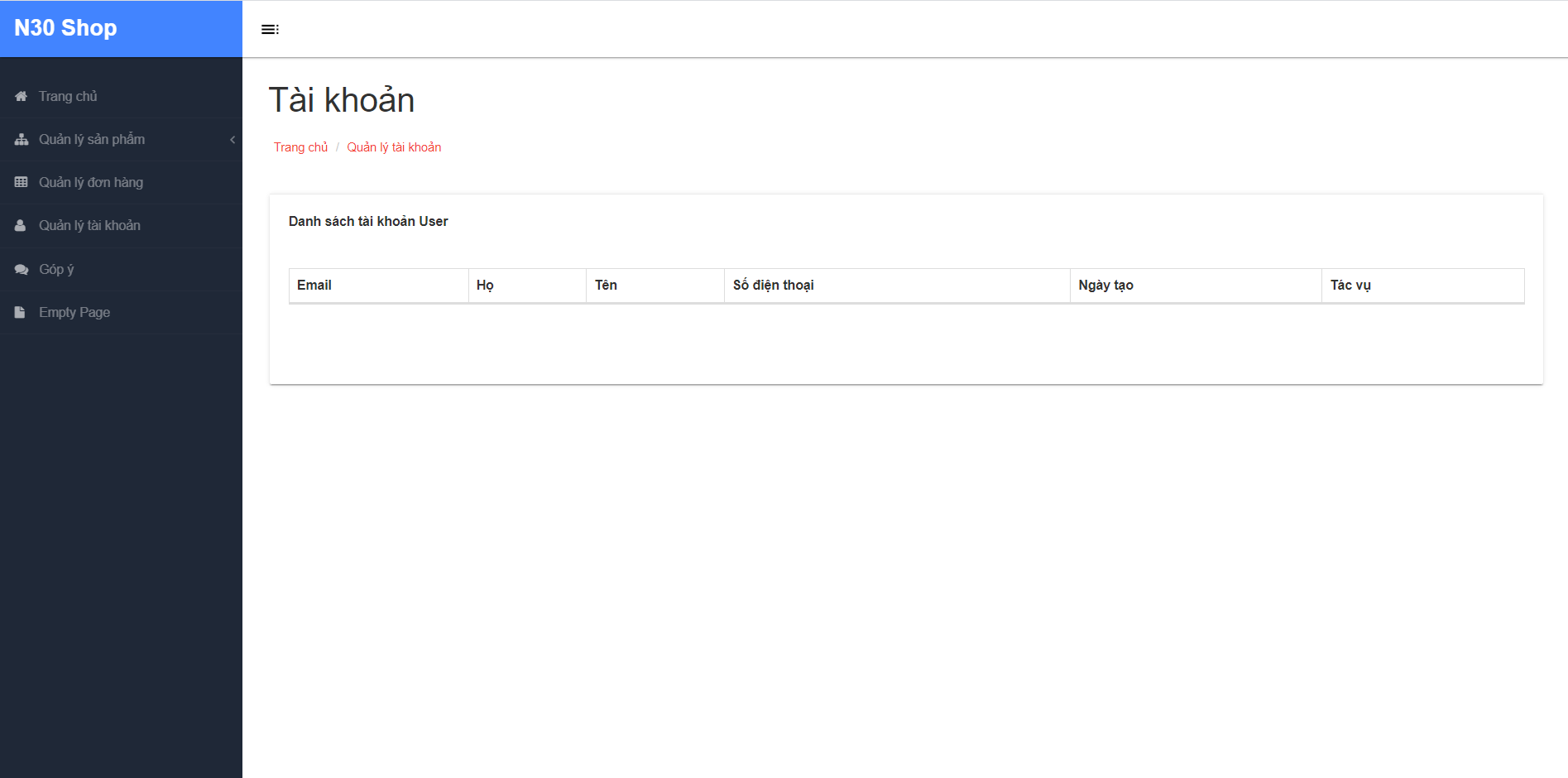
****

**14. Giao diện quản lý đơn hàng**

****

****

**15. Giao diện quản lý tài khoản**

****

37