****

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

VÀ TRUYỀN THÔNG VIỆT - HÀN

**KHOA KHOA HỌC MÁY TÍNH**

**BÁO CÁO LẬP TRÌNH MẠNG**

**ĐỀ TÀI: TÌM HIỂU VÀ XÂY DỰNG**

**WEB SERVER**

**Sinh viên thực hiện : Nguyễn Trần Tiến 20SE5**

**: Nguyễn Thị Hiền 20SE5**

**Giảng viên hướng dẫn : ThS. Nguyễn Thanh Cẩm**

***Đà Nẵng, tháng 01 năm 2023***

**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &**

**TRUYỀN THÔNG VIỆT –HÀN**

**KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO LẬP TRÌNH MẠNG**

**ĐỀ TÀI: TÌM HIỂU VÀ XÂY DỰNG**

**WEB SERVER**

***Đà Nẵng, tháng 01 năm 2023***

# LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt quãng thời gian năm năm từ khi bắt đầu học tập tại giảng đường trường Đại học Công nghệ thông tin và Truyền Thông Việt Hàn – Đại học Đà Nẵng, nhóm em đã nhận được rất nhiều sự quan tâm, giúp đỡ của thầy cô và các bạn.Với lòng biết ơn sâu sắc nhất, nhóm em xin gửi lời cảm ơn chân thành tới toàn thể các thầy cô giáo trong trường Đại học Công nghệ thông tin và Truyền Thông Việt Hàn – Đại học Đà Nẵng, những người đã dạy dỗ và truyền đạt vốn kiến thức vô cùng quý báu của mình cho chúng em trong suốt quãng thời gian học tại đây. Những tình cảm và kiến thức mà thầy cô đã gửi trao cho chúng em sẽ là hành trang cho chúng em để bước đi trên những đoạn đường tương lai.

Nhóm em cũng xin gửi lời cảm ơn sâu sắc tới Thầy giáo – Thạc sĩ Nguyễn Thanh Cẩm, người đã tận tình hỗ trợ và dẫn dắt chúng nhóm em trong suốt quá trình thực hiện đồ án để có được kết quả một cách tốt nhất. Một lần nữa nhóm em xin được chân thành cảm ơn. Em xin trân trọng cảm ơn!!

***Sinh viên***

Nguyễn Trần Tiến-20SE5

Nguyễn Thị Hiền-20SE5

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

Đà Nẵng , ngày….tháng….năm 2022

Giảng viên hướng dẫn

MỤC LỤC

[PHẦN MỞ ĐẦU 1](#_Toc81944192)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 1](#_Toc81944193)

1.[1 Mô tả tổng quan đề tài 1](#_Toc81944194)

1.[2 Các chức năng của đề tài 1](#_Toc81944195)

1.[3 Tìm hiểu chung về Web Server 1](#_Toc81944196)

1.3.1 [Web Server là gì ? 1](#_Toc81944206)

1.[3.2 Chức năng của Web Server 1](#_Toc81944206)

1.[3.3 Web Server hoạt động như thế nào ? 2](#_Toc81944206)

1.[3.4 Một số Web Server phổ biến hiện nay 2](#_Toc81944206)

1.[4. Tìm hiểu giao thức HTTP, HTTPS và các phương thức của nó 5](#_Toc81944197)

1.4.1 [Giao thức HTTP là gì ? 5](#_Toc81944206)

1.4.2 [Cách hoạt động giao thức HTTP 5](#_Toc81944206)

1.4.3 [HTTP - Request 6](#_Toc81944206)

1.4.4 [HTTP - Reponses 8](#_Toc81944206)

1.4.5 [Giao thức HTTPS là gì ? 10](#_Toc81944206)

1.[5. Tồng quan về ngôn ngữ và công cụ sử dụng 5](#_Toc81944197)

[CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 12](#_Toc81944199)

[2.1 Phân tích bài toán 12](#_Toc81944205)

[2.1.1 Mô hình sử dụng 13](#_Toc81944207)

[2.1.2 Giao thức sử dụng 13](#_Toc81944208)

[2.1.3 Socket là gì ? 13](#_Toc81944206)

[2.1.4 Tổng quan về ngôn ngữ và công cụ sử dụng 15](#_Toc81944206)

[2.2 Thiết kế hệ thống 21](#_Toc81944205)

[2.2.1 Danh sách các actor và chức năng 21](#_Toc81944206)

[2.2.2 Biểu đồ Usecase 21](#_Toc81944206)

[2.2.3 Biểu đồ lớp 22](#_Toc81944207)

[CHƯƠNG 3: CHƯƠNG TRÌNH DEMO 24](#_Toc81944211)

[3.1 Lý do chọn đề tài 24](#_Toc81944212)

[3.2 Giới thiệu sản phẩm 24](#_Toc81944212)

[KIẾN NGHỊ VÀ KẾT LUẬN 28](#_Toc81944223)

[1. Kết quả đạt được 28](#_Toc81944224)

[2. Hạn chế 28](#_Toc81944225)

[3. Kết luận 28](#_Toc81944226)

[4. Hướng phát triển 28](#_Toc81944226)

[TÀI LIỆU THAM KHẢO 29](#_Toc81944227)

MỤC LỤC HÌNH ẢNH

[Hình 1 Mô hình hoạt động của Web Server 3](#_Toc81944238)

[Hình 2 Apache 4](#_Toc81944239)

[Hình 3 Nginx 5](#_Toc81944240)

[Hình 4 Mô hình TCP/IP và mô hình OSI 6](#_Toc81944241)

[Hình 5 Mô hình hoạt động của giao thức HTTP 7](#_Toc81944242)

[Hình 6 Định dạng của một HTTP request 7](#_Toc81944243)

[Hình 7 Ví dụ một định dạng HTTP request 7](#_Toc81944243)

[Hình 8 HTTP Methods CRUD 8](#_Toc81944243)

[Hình 9 Định dạng gói tin HTTP response 10](#_Toc81944243)

[Hình 10 Chứng chỉ SSL 12](#_Toc81944243)

[Hình 11 Mô hình client/server 14](#_Toc81944243)

[Hình 12 Socket 15](#_Toc81944243)

[Hình 13 Socket qua ví dụ 15](#_Toc81944243)

[Hình 14 TCP Socket 16](#_Toc81944243)

[Hình 15 Java 18](#_Toc81944243)

[Hình 16 HTML 19](#_Toc81944243)

[Hình 17 CSS 20](#_Toc81944243)

[Hình 18 Javascript 21](#_Toc81944243)

[Hình 19 Usecase chính 22](#_Toc81944243)

[Hình 20 Biểu đồ lớp 23](#_Toc81944243)

[Hình 21 Mô hình hoạt động Web Server 23](#_Toc81944243)

[Hình 22 Client gửi request lên Server và được trả Reponse 24](#_Toc81944243)

[Hình 23 Trang điều khiển của Server 25](#_Toc81944243)

[Hình 24 Tiến hành bật và chạy Server 25](#_Toc81944243)

[Hình 25 Cổng port đang đươc sử dụng 25](#_Toc81944243)

[Hình 26 Server được chạy và nhận được request của người dùng 26](#_Toc81944243)

**PHẦN MỞ ĐẦU**

1. **Lý do chọn đề tài**

* Lập trình mạng là một môn học bổ ích giúp em có thể hiểu biết thêm các mô hình mạng, các giao thức ở trong mạng, các giao thức. Lập trình mạng giúp em biết đến Web Server, các công dụng của nó.
* Từ những kiến thức mà chúng em đã học được trong thời gian qua. Cũng như niềm mong muốn tìm hiểu thử cách thức của một Web Server hoạt động như thế nào từ đó em quyết định thực hiện đề tài Tìm hiểu và xây dựng Web Server

1. **Mục tiêu thực hiện**

* Hoàn thành được sản phẩm và báo cáo đồ án môn học.
* Tìm hiểu được các Web Server hiện nay và xem cách thức hoạt động của nó.
* Thực hiện được một Web Server đơn giản và xem cách.

1. **Phương pháp nghiên cứu**

* Tìm hiểu (tham khảo ý kiến mọi người, thực hiện việc tìm kiếm tài liệu trên mạng)
* Vận dụng kiến thức phân tích bài toán
* Tìm ra phương pháp tối ưu giải quyết bài toán
* Báo cáo và nhận phản hồi từ người hướng dẫn
* Hoàn thiện công việc được giao.

1. **Nội dung nghiên cứu**

* Tìm hiểu về các Web Server hiện nay, tìm hiểu mô hình hoạt động của chúng.
* Tìm hiểu các mô hình mạng, socket các giao thức sử dụng trong Web Server
* Xác định và phân tích các yêu cầu cho Web Server.
* Triển khai demo.
* Kiểm thử ứng dụng.

1. **Bố cục**

Sau phần *Mở đầu*, báo cáo được trình bày trong ba chương, cụ thể như sau:

Chương 1. *Cơ sở lý thuyết*

Chương 2. *Phân tích và thiết kế hệ thống*

Chương 3. *Chương trình demo* Chương này trình bày các kết quả, chức năng cài đặt được của hệ thống: về phía người dùng & về phía người quản trị, giao diện…

Cuối cùng là *Kết luận*, *Tài liệu tham khảo* và *Phụ lục* liên quan đến đề tài

# CƠ SỞ LÝ THUYẾT

## Tổng quan đề tài

* Tên đề tài: **Tìm hiểu và xây dựng Web Server**
* Tổng quan: Với mục đích tìm hiểu cách thức hoạt động của Web Server như thế nào cũng như các giao thức, các mô hình mạng được sử dụng để xậy dựng lên một Web Server. Đề tài bao gồm phần server và client. Phía server sẽ hổ trợ các chức năng như GET, POST, … giúp người quản trị có thể thay đổi được các thông số như port, chạy và dừng server cũng xem các thông số như tổng số client kết nối với server, địa chỉ IP. Client là người dùng khi truy cập một website và gửi yêu cầu mà client mong muốn nhận lại kết quả, khi đó phía server sẽ phân tích và trả dữ liệu về lại cho người dùng.

## Các chức năng của đề tài

* Phía Server:
  + Hiển thị thông tin của phía Client yêu cầu: Địa chỉ IP, HostName, Thông tin của Request
  + Nhận được request từ Client sau đó thực hiện việc phân tích và trả dữ liệu về phía Client
  + Tính tổng số port, chạy và dừng Server
  + Hỗ trợ các phương thức như GET, POST
* Phía Client:
  + Gửi Request tới phía Server và yêu cầu Server trả về thông tin phía Client mong muốn. Thao tác trên Web Server

## Tìm hiểu chung về Web Server

### Web Server là gì:

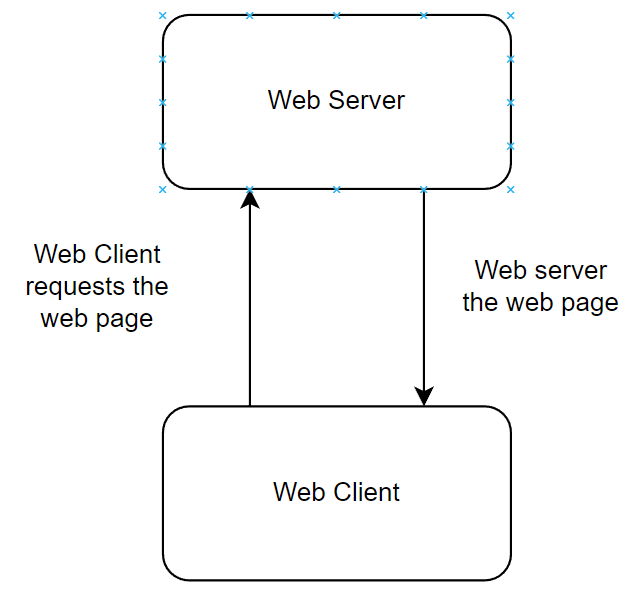
* WebServer được biết đến là máy chủ web được dùng để xử lý cái request từ trình duyệt web máy khách và gửi thông tin đến client đã gửi yêu cầu thông qua giao thức HTTP hoặc các giao thức khác.

### Chức năng của Web Server

* Chức năng cơ bản của web server không thể thiếu là lưu trữ, xử lý và phân phối nội dung website đến với khách hàng.
* Mỗi máy chủ là một IP khác nhau và có thể đọc các ngôn ngữ như file \*.htm và \*.html,… Máy chủ này cũng cho phép một trang web để được xem bằng HTTP, HTTP (HyperText Transfer Protocol) là giao thức chính cho việc chuyển dữ liệu trên web.
* Web server phải có dung lượng lớn thì mới có thể lưu trữ vận hành tốt một kho dữ liệu trên internet. Nhờ đó, web server có khả năng vận hành trơn tru một hệ thống máy tính hoạt động, đảm bảo các hoạt động này liên tục để duy trì cung cấp dữ liệu cho mạng lưới máy tính của mình

### Web Server hoạt động như thế nào ?

* Bất cứ khi nào bạn xem một trang web trên internet, có nghĩa là bạn đang yêu cầu trang đó từ một web server. Khi bạn nhập URL trên trình duyệt của mình (ví dụ: https://google.com) nó sẽ tiến hành các bước sau để gửi lại phản hồi cho bạn.

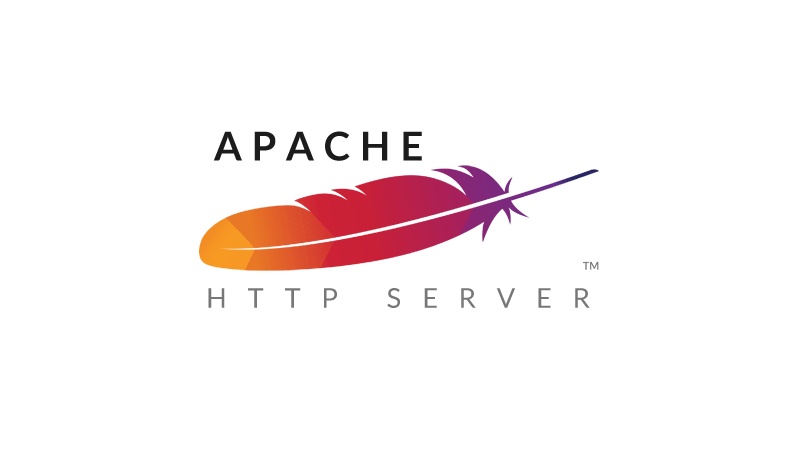


Hình 1: Mô hình hoạt động của Web Server

* Trình duyệt web của bạn trước tiên cần phải xác định địa chỉ IP nào mà tên miền google.com trỏ về. Trình duyệt sẽ yêu cầu thông tin từ một hoặc nhiều máy chủ DNS (thông qua internet). Máy chủ DNS sẽ cho trình duyệt biết địa chỉ IP nào tên miền sẽ trỏ đến cũng là nơi đặt trang web. Lúc này trình duyệt web đã biết địa chỉ IP của trang web, nó có thể yêu cầu URL đầy đủ từ webserver.
* Web server phản hồi bằng cách gửi lại những thông tin client yêu cầu… Nếu trang không tồn tại hoặc có lỗi khác xảy ra, nó sẽ gửi lại thông báo lỗi thích hợp.
* Trình duyệt web của bạn nhận lại được các tập tin html css (nhiều file khác)… và render hiển thị trang theo yêu cầu.

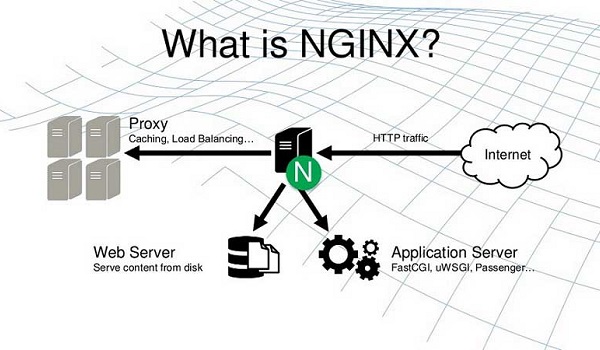
### Một số Web Server phổ biến hiện nay

* Apache HTTP server
  + Apache là web server được sử dụng rộng rãi nhất thế giới. Apache được phát triển và duy trì bởi một cộng đồng mã nguồn mở dưới sự bảo trợ của Apache Software Foundation. Apache được phát hành với giấy phép Apache License là được sử dụng tự do, miễn phí.
  + Apache hay là chương trình máy chủ HTTP là một chương trình dành cho máy chủ đối thoại qua giao thức HTTP. Apache chạy trên các hệ điều hành tương tự như Unix, Microsoft Windows, Novell Netware và các hệ điều hành khác. Apache đóng một vai trò quan trọng trong quá trình phát triển của mạng web thế giới
  + Nó giúp chủ website đưa nội dung lên web – vì vậy có tên gọi là “web server”. Apache là một trong số những web server lâu đời và đáng tin cậy nhất, phiên bản đầu tiên đã được ra mắt từ hơn 20 năm trước.
  + Khi một người truy cập vào website của bạn, họ sẽ nhập tên miền vào thanh địa chỉ. Sau đó, web server sẽ chuyển những files được yêu cầu xuống máy tính của người truy cập như là một nhân viên vân chuyển ảo.



Hình 2: Apache

* Mặc dù chúng ta gọi Apache là web server, nhưng nó lại không phải là server vật lý, nó là một phần mềm chạy trên server đó. Công việc của nó là thiết lập kết nối giữa server và trình duyệt người dùng (Firefox, Google Chrome, Safari, vâng vâng.) rồi chuyển file tới và lui giữa chúng (cấu trúc 2 chiều dạng client-server). Apache là một phần mềm đa nền tảng, nó hoạt động tốt với cả server Unix và Windows.
* Khi một khách truy cập tải một trang web trên website của bạn, ví dụ, trang chủ “Giới Thiệu”,trình duyệt người dùng sẽ gửi yêu cầu tải trang web đó lên server và Apache sẽ trả kết quả với tất cả đầy đủ các file cấu thành nên trang Giới Thiệu (hình ảnh, chữ…). Server và client giao tiếp với nhau qua giao thức HTTP và Apache chịu trách nhiệm cho việc đảm bảo tiến trình này diễn ra mượt mà và bảo mật giữa 2 máy.
* Apache là một nền tảng module có độ tùy biến rất cao. Modules cho phép quản trị server tắt hoặc thêm chức năng. Apache có modules cho bảo mật caching, URL rewriting, chứng thực mật khẩu, vâng vâng. bạn cũng có thể thiết lập cấu hình server riêng qua file gọi là .htaccess, vốn là file cấu hình Apache nó được hỗ trợ hoàn toàn bởi mọi gói.
* Nginx:
  + NGINX là một phần mềm web server mã nguồn mở, sử dụng kiến trúc hướng sự kiện (event-driven) không đồng bộ (asynchronous). Mục tiêu ban đầu để phục vụ HTTP cache nhưng sau được áp dụng vào reverse proxy, HTTP load balancer và các giao thức truyền mail như IMAP4, POP3, và SMTP.
  + NGINX chính thức ra đời vào tháng 10/2014. Đây là phần mềm giúp server có tốc độ và khả năng mở rộng lớn nhất, đồng thời, xử lý và thao tác trên hàng nghìn kết nối cùng lúc. Do đó, rất nhiều “ông lớn” công nghệ hiện nay đều lựa chọn NGINX như Google, Adobe, Netflix, WordPress…
  + Về cơ bản, NGINX cũng hoạt động tương tự như các web server khác. Khi bạn mở một trang web, trình duyệt của bạn sẽ liên hệ với server chứa website đó. Server sẽ tìm kiếm đúng file yêu cầu của website và gửi về cho bạn. Đây là một trình tự xử lý dữ liệu single – thread, nghĩa là các bước được thực hiện theo một trình tự duy nhất. Mỗi yêu cầu sẽ được tạo một thread riêng.
  + Tuy nhiên, NGINX hoạt động theo kiến trúc bất đồng bộ (asynchronous) hướng sự kiện (event driven). Nó cho phép các threads tương đồng được quản lý trong một tiến process. Mỗi process hoạt động sẽ bao gồm các thực thể nhỏ hơn, gọi là worker connections dùng để xử lý tất cả threads.
  + Worker connections sẽ gửi các yêu cầu cho worker process, worker process sẽ gửi nó tới master process, và master process sẽ trả lời các yêu cầu đó. Đó là lý do vì sao một worker connection có thể xử lý đến 1024 yêu cầu tương tự nhau. Nhờ vậy, NGINX có thể xử lý hàng ngàn yêu cầu khác nhau cùng một lúc.



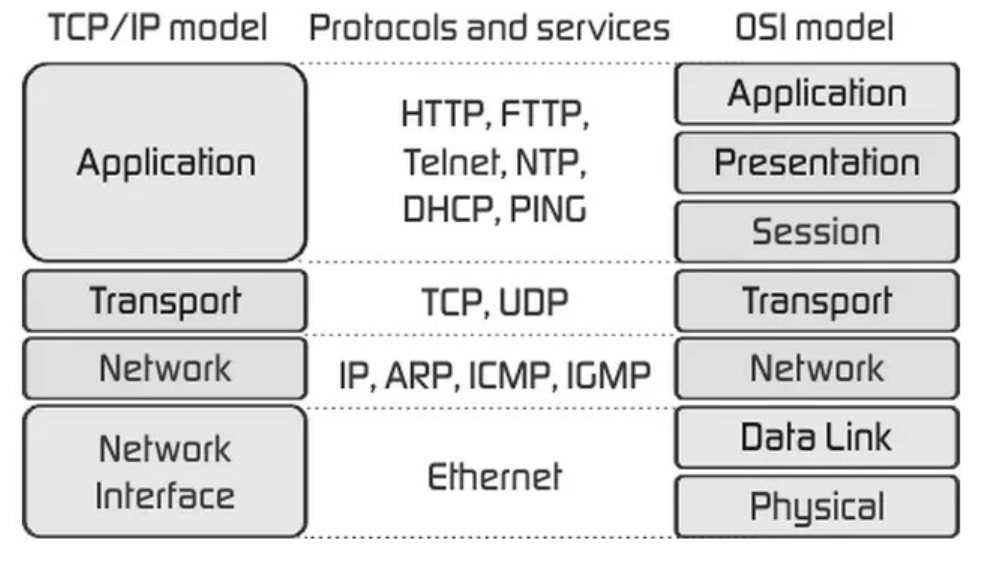
Hình 3: Nginx

## Tìm hiểu giao thức HTTP, HTTPs và các phương thức của nó

### Giao thức: Protocol (giao thức mạng) là tập hợp các quy tắc được thiết lập nhằm xác định cách để định dạng, truyền và nhận dữ liệu sao cho các thiết bị mạng máy tính - từ server và router tới endpoint - có thể giao tiếp với nhau, bất kể sự khác biệt về cơ sở hạ tầng, thiết kế hay các tiêu chuẩn cơ bản giữa chúng.

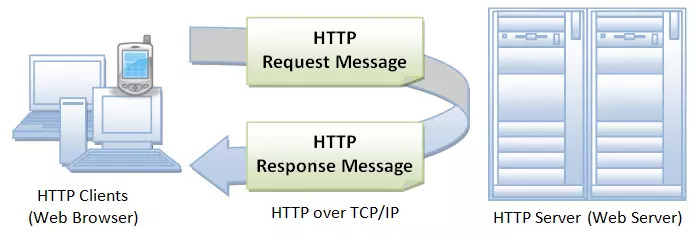
### Giao thức HTTP là gì?

* HTTP (HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản) là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW
* Giao thức HTTP là một giao thức tầng ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet).

Hình 4: Mô hình TCP/IP và mô hình OSI

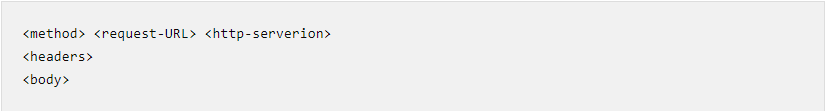
### Cách hoạt động giao thức HTTP

* Giao thức HTTP hoạt động dựa trên mô hình Client – Server. Thông thường khi các bạn lướt web, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này.
* Ngoài ra, khi các hệ thống trao đổi dữ liệu với nhau, chúng cũng sử dụng giao thức này nhưng 2 bên đều là server.

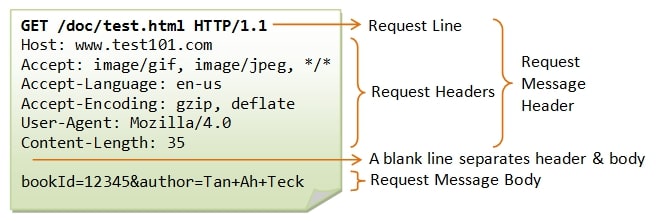
Hình 5: Mô hình hoạt động của giao thức HTTP

### HTTP – Request

* Một HTTP client (máy khách) gửi một HTTP request (yêu cầu) lên server (máy chủ) nhờ một thông điệp có định dạng như sau:

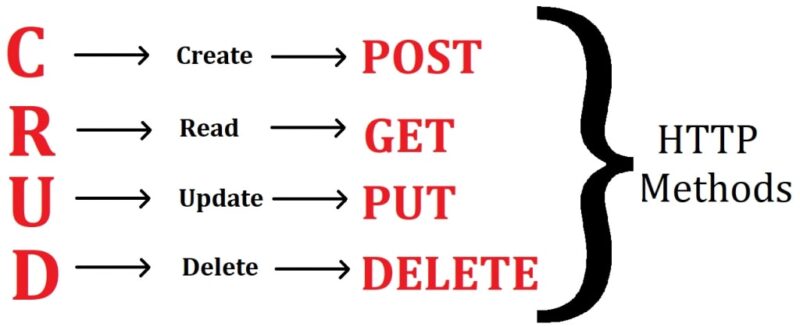


Hình 6: Định dạng của một HTTP request



Hình 7: Ví dụ một định dạng HTTP request

* Request Line: Bắt đầu của HTTP Request sẽ là dòng Request-Line bao gồm 3 thông tin:
  + Method: Báo cho Server rằng hành động sẽ phải sử lý với thông tin được gửi từ client lên.
  + Phương thức GET
    - Câu truy vấn sẽ được đính kèm vào đường dẫn HTTP request. Ví dụ: /?username=”tinohost”&pass=”tenmien”
    - GET request có thể được cached, bookmark và lưu trong lịch sử của trình duyệt mà bị giới hạn về chiều dài (chiều dài của URL là có hạn).
    - Lưu ý: Bạn không nên dùng GET request với dữ liệu quan trọng mà chỉ dùng để nhận dữ liệu, không có tính bảo mật.
* Phương thức POST
* Câu truy vấn sẽ được gửi trong phần message body của HTTP request.
* POST không thể, cached, bookmark hay lưu trong lịch sử trình duyệt và cũng không bị giới hạn về độ dài.
* Các phương thức khác:
  + HEAD: tương tự như GET nhưng chỉ gửi về HTTP header.
  + PUT: tải lên một mô tả về URL định trước.
  + DELETE: xóa một tài nguyên định trước.
  + OPTIONS: trả về phương thức HTTP mà server hỗ trợ.
  + CONNECT: chuyển kết nối của HTTP request thành một kết nối HTTP tunnel.

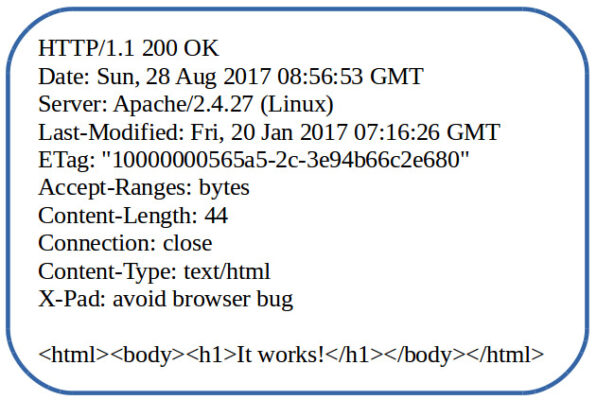


Hình 8: HTTP Methods CRUD

* + Request URL:
    - Một **URL** (Uniform Resource Locator) được sử dụng để xác định duy nhất một tài nguyên trên Web. Một URL có cấu trúc như sau:
    - Trong một **URL** có 4 thành phần:
      * Protocol: giao thức tầng ứng dụng được sử dụng bởi client và server.
      * Hostname: tên DNS domain.
      * Port: Cổng TCP để server lắng nghe request từ client.
      * Path-and-file-name: Tên và vị trí của tài nguyên yêu cầu.
  + HTTP version:
    - HTTP version là Phiên bản giao thức HTTP đang sử dụng.
* Request header:
  + Tiếp theo dòng Request-Line là các trường Request-header, cho phép client gửi thêm các thông tin bổ sung về thông điệp HTTP request và về chính client. Một số trường thông dụng như:
    - Accept loại nội dung có thể nhận được từ thông điệp response. Ví dụ: text/plain, text/html…
    - Accept-Encoding: các kiểu nén được chấp nhận. Ví dụ: gzip, deflate, xz, exi…
    - Connection: tùy chọn điều khiển cho kết nối hiện thời. Ví dụ: keepalive,Upgrade…
    - Cookie: thông tin HTTP Cookie từ server
    - User-Agent: thông tin về user agent của người dùng.
* Request Body:
  + Body là dữ liệu mà Client sẽ gửi lên Server, Một request body có thể là một đoạn text thuần túy, HTML, XML, JSON, Javascript, hoặc một tập các cặp key-value dạng form-data.
  + Khi bạn mở một web, trình duyệt sẽ nhận payload dạng HTML, nó chính là giao diện mà chúng ta quan sát được trên trình duyệt đó.
  + Thông thường khi làm việc với các HTTP APIs chúng ta sẽ gửi và nhận các payload dạng JSON hoặc XML.
  + Không phải tất cả các message HTTP đều phải có payload: POST và PUT có thể có, còn với GET và DELETE thì có thể không có payload.
  + Sau khi request server sẽ xử lý và phản hổi (response) tới client theo 1 trong 3 phương pháp:
    - Server phân tích request nhận được, maps yêu cầu với tập tin trong tập tài liệu của server, và trả lại tập tin yêu cầu cho client.
    - Server phân tích request nhận được, maps yêu cầu vào một chương trình trên server, thực thi chương trình và trả lại kết quả của chương trình đó.
    - Request từ client không thể đáp ứng, server trả lại thông báo lỗi.

### HTTP – Responses

* HTTP response là bản tin trả về từ server sang client, trong đó sẽ có các trường thông tin mà request yêu câu.
* Định dạng gói tin HTTP response cũng gồm 3 phần chính là: Status line, Header và Body.



Hình 9: Định dạng gói tin HTTP response

* Response Status:
  + Gồm 3 trường là phiên bản giao thức (HTTP version), mã trạng thái (Status code) và mô tả  trạng thái (Status text):
    - Phiên bản giao thức (HTTP version): phiên bản của giao thức HTTP mà server hỗ trợ, thường là HTTP/1.0 hoặc HTTP/1.1
    - Mã trạng thái (Status code): mô tả trạng thái kết nối dưới dạng số, mỗi trạng thái sẽ được biểu thị bởi một số nguyên. Ví dụ: 200, 404, 302,…
    - Mô tả trạng thái (Status text): mô tả trạng thái kết nối dưới dạng văn bản một cách ngắn gọn, giúp người dùng dễ hiểu hơn so với mã trạng thái. Ví du: 200 OK, 404 Not Found, 403 Forbiden,…
  + Một số loại thông dụng mà server trả về cho client như sau:
    - 1xx: information Message:
      * Các status code này chỉ có tính chất tạm thời, client có thể không quan tâm.
* 2xx Successful: Khi đã xử lý thành công request của client, server trả về status dạng này:
  + 200 OK: request thành công.
  + 202 Accepted: request đã được nhận, nhưng không có kết quả nào trả về, thông báo cho client tiếp tục chờ đợi.
  + 204 No Content: request đã được xử lý nhưng không có thành phần nào được trả về.
  + 205 Reset: giống như 204 nhưng mã này còn yêu câu client reset lại document view.
  + 206 Partial Content: server chỉ gửi về một phần dữ liệu, phụ thuộc vào giá trị range header của client đã gửi.
* 3xx Redirection: Server thông báo cho client phải thực hiện thêm thao tác để hoàn tất request:
  + 301 Moved Permanently: tài nguyên đã được chuyển hoàn toàn tới địa chỉ Location trong HTTP response.
  + 303 See other: tài nguyên đã được chuyển tạm thời tới địa chỉ Location trong HTTP response.
  + 304 Not Modified: tài nguyên không thay đổi từ lần cuối client request, nên client có thể sử dụng đã lưu trong cache.
* 4xx Client error: Lỗi của client:
  + 400 Bad Request: request không đúng dạng, cú pháp.
  + 401 Unauthorized: client chưa xác thực.
  + 403 Forbidden: client không có quyền truy cập.
  + 404 Not Found: không tìm thấy tài nguyên.
  + 405 Method Not Allowed: phương thức không được server hỗ trợ.
* 5xx Server Error: Lỗi của server:
  + 500 Internal Server Error: có lỗi trong quá trình xử lý của server.
  + 501 Not Implemented: server không hỗ trợ chức năng client yêu cầu.
  + 503 Service Unavailable: Server bị quá tải, hoặc bị lỗi xử lý.
* Response Header:
  + Header của gói tin response có chức năng tương tự như gói tin request, giúp server có thể truyền thêm các thông tin bổ sung đến client dưới dạng các cặp “Name:Value”.
* Response Body:
  + Là nơi đóng gói dữ liệu để trả về cho client, thông thường trong duyệt web thì dữ liệu trả về sẽ ở dưới dạng một trang HTML để trình duyệt có thể thông dịch được và hiển thị ra cho người dùng.
  + Hoặc trả về dạng JSON, XML khi giao tiếp bằng API.

### Giao thức HTTPs là gì?

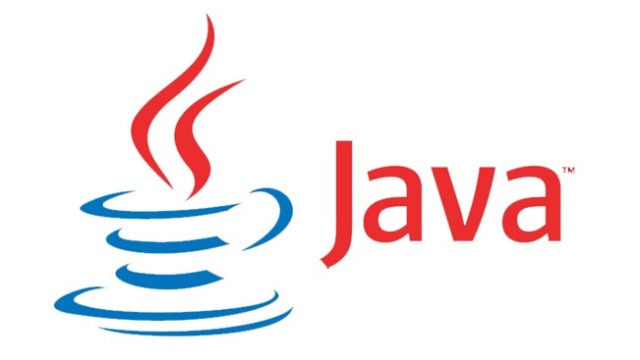
* HTTPS viết tắt của Hyper Text Transfer Protocol Secure (giao thức truyền tải siêu văn bản bảo mật) là phiên bản an toàn của HTTP, giao thức mà nhờ đó dữ liệu được gửi giữa trình duyệt và trang web bạn đang kết nối. Chữ ‘S’ ở cuối HTTPS là viết tắt của “Secure” (Bảo mật)
* HTTPS cũng có những hoạt động giống như HTTP. Tuy nhiên, chúng được bổ sung SSL và TSL. Giao thức này đảm bảo không có bên thứ 3 đăng nhập được và đánh cắp thông tin. Dù cho bạn đang tìm kiếm thông tin bằng máy tính công cộng hay máy cá nhân thì HTTPS cũng sẽ đảm bảo thông tin liên lạc của bạn với máy chủ được bảo mật.
* Bảo mật PKI – Public Key Infrastructure
  + Cả hai giao thức TLS và SSL đều sử dụng hệ thống PKI (Public Key Infrastructure, hạ tầng khóa công khai) không đối xứng. Một hệ thống không đối xứng sử dụng hai “khóa” để mã hóa thông tin liên lạc, khóa “công khai” và khóa “riêng”. Bất cứ thứ gì được mã hóa bằng khoá công khai (public key) chỉ có thể được giải mã bởi khóa riêng (private key) và ngược lại.
  + Một hệ thống PKI hoàn chỉnh bao gồm các thành phần sau:
    - Khóa công khai – Public Key
    - Ủy quyền thẻ chứng thực – Certification Authority (CA)
    - Ủy quyền thẻ đăng ký – Registration Authority (RA)
    - Ủy quyền xác nhận hợp lệ – Validation Authority (VA)
    - Thẻ chứng thực số – Digital certificates (DC)
    - Thẻ chứng thực bị CA thu hồi – Certificate revocation list (CRL)
    - Hệ thống phân phối thẻ – Certificate Distribution System (CDS)
* Chứng chỉ SSL là gì ?
  + Khi yêu cầu kết nối giao thức HTTPS với trang web, đầu tiên trang web sẽ gửi chứng chỉ SSL tới trình duyệt của bạn. Chứng chỉ này chứa khóa công khai cần thiết để bắt đầu phiên bảo mật. Dựa trên trao đổi ban đầu này, trình duyệt và trang web sẽ bắt đầu giao thức SSL handshake (giao thức bắt tay). Giao thức SSL handshake liên quan đến việc tạo bí mật chia sẻ để thiết lập kết nối an toàn duy nhất giữa bạn và trang web.
  + Khi sử dụng chứng chỉ SSL đáng tin cậy trong quá trình kết nối HTTPS, người dùng sẽ thấy biểu tượng ổ khóa trong thanh địa chỉ của trình duyệt. Khi một chứng chỉ Extended Validation Certificate được cài đặt trên một trang web, thanh địa chỉ sẽ chuyển sang màu xanh lá cây.



Hình 10: Chứng chỉ SSL

## Tổng quan về ngôn ngữ và công cụ sử dụng

* Java :
  + Java là một một ngôn ngữ lập trình hiện đại, bậc cao, hướng đối tượng, bảo mật và mạnh mẽ. và là một Platform.
  + Platform: Bất cứ môi trường phần cứng hoặc phần mềm nào mà trong đó có một chương trình chạy, thì được hiểu như là một Platform. Với môi trường runtime riêng cho mình (JRE) và API, Java được gọi là Platform.
  + Ngôn ngữ lập trình Java ban đầu được phát triển bởi Sun Microsystems do James Gosling khởi xướng và phát hành vào năm 1995. Phiên bản mới nhất của Java Standard Edition là Java SE 8. Với sự tiến bộ của Java và sự phổ biến rộng rãi của nó, nhiều cấu hình đã được xây dựng để phù hợp với nhiều loại nền tảng khác nhau. Ví dụ: J2EE cho các ứng dụng doanh nghiệp, J2ME cho các ứng dụng di động.
* Các phiên bản J2 mới đã được đổi tên thành Java SE, Java EE và Java ME. Phương châm của java là "Write Once, Run Anywhere" - viết một lần chạy nhiều nơi, nghĩa là bạn chỉ cần viết một lần trên window chẳng hạn, sau đó vẫn chương trình đó bạn có thể chạy trên Linux, Android, các thiết bị J2ME…
* Các tính năng của Java
  + Hướng đối tượng: Trong Java, tất cả đều là một Object. Java có thể mở rộng và bảo trì dễ dàng bởi nó được xây dựng dựa trên mô hình Object.
  + Nền tảng độc lập: Khi được biên dịch, Java không được biên dịch thành ngôn ngữ máy trên nền tảng cụ thể mà thay vào mã byte – một nền tảng độc lập. Mã byte này được thông dịch từ máy ảo (JVM) trên một nền tảng nào đó mà nó đang chạy.
  + Đơn giản: Java được thiết kế đơn giản, dễ học. Chỉ hiểu khái niệm cơ bản về OOP Java, để trở thành master về java rất dễ.
  + Bảo mật: Tính năng an toàn của Java cho phép phát triển các hệ thống không virus, không giả mạo, việc xác thực dựa trên mã hoá khóa công khai.
  + Kiến trúc – trung lập: Trình biên dịch của Java tạo ra các định dạng tệp đối tượng kiến trúc trung lập, khiến mã biên dịch được thực thi trên nhiều bộ vi xử thông qua hệ điều hành Java.
  + Portable: Một loại kiến trúc trung lập, phụ thuộc vào việc thực hiện, là những đặc điểm chính khi nói về Portable của Java. Các trình biên dịch tại Java được viết bằng ANSI C với ranh giới Portable gọn gàng, gọi là Subset POSIX, bạn có thể mang Byte Code của Java lên bất cứ một nền tảng nào.
  + Mạnh mẽ: Java luôn nỗ lực loại trừ các tình huống dễ bị lỗi thông qua việc kiểm tra lỗi tại thời điểm biên dịch và kiểm tra lỗi tại runtime.
  + Đa luồng: Giúp tạo ra các chương trình thực hiện cùng lúc nhiều tác vụ, cho phép các nhà phát triển xây dựng ứng dụng tương tác chạy trơn tru hơn.
  + Thông dịch: Các mã byte Java được dịch trực tiếp tới các máy tính gốc và không được lưu trữ ở bất cứ đâu.
  + Hiệu năng cao: Khi sử dụng trình biên dịch Just-In-Time, Java cho phép thực hiện hiệu năng cao.
  + Phân tán: Java được sử dụng thiết kế cho môi trường phân tán của Internet.
  + Năng động: Java năng động hơn C hoặc C++ do được thiết kế với mục đích thích ứng môi trường đang phát triển. Các chương trình Java sở hữu lượng lớn thông tin tại runtime có thể được sử dụng để xác minh hay giải quyết các truy cập vào các đối tượng tại runtime.
* Được biết Java rất phổ biến và đã thống trị lĩnh vực này từ đầu những năm 2000, đến nay Java đã được sử dụng trong đa dạng các lĩnh vực khác nhau. Cụ thể:
  + Desktop App như media player, acrobat reader, antivirus,…
  + Web App như javatpoint.com, irctc.co.in,…
  + Enterprise App như một vài ứng dụng về xử lý các nghiệp vụ ngân hàng,…
  + Thiết bị Mobile như các ứng dụng IOS hay Android.
  + Hệ thống nhúng.
  + Smart Card.
  + Robot.
  + Game App.
  + Socket trong Java là một trong những nút của liên kết giao tiếp hai chiều giữa các chương trình máy khách và máy chủ đang chạy trên mạng. Điểm cuối hoặc nút là sự kết hợp của địa chỉ IP và số cổng. Có một số cổng cho mỗi ổ cắm để lớp TCP có thể xác định ứng dụng nơi gửi dữ liệu.



Hình 15: Java

* HTML:
  + HTML được tạo ra bởi Tim Berners-Lee, một nhà vật lý học của trung tâm nghiên cứu CERN ở Thụy Sĩ. Hiện nay, HTML đã trở thành một chuẩn Internet được tổ chức W3C (World Wide Web Consortium) vận hành và phát triển. Bạn có thể tự tìm kiếm tình trạng mới nhất của HTML tại bất kỳ thời điểm nào trên Website của W3C. Phiên bản đầu tiên của HTML xuất hiện năm 1991, gồm 18 tag HTML. Phiên bản HTML 4.01 được xuất bản năm 1999. Sau đó, các nhà phát triển đã thay thế HTML bằng XHTML vào năm 2000. Đến năm 2014, HTML được nâng cấp lên chuẩn HTML5 với nhiều tag được thêm vào markup, mục đích là để xác định rõ nội dung thuộc loại là gì (ví dụ như: <article>, <header>, <footer>,…).Theo Mozilla Developer Network thì HTML Element Reference hiện nay có khoảng hơn 140 tag. Tuy nhiên một vài tag trong số đó đã bị tạm ngưng (do không được hỗ trợ bởi các trình duyệt hiện hành).
  + HTML (viết tắt của từ HyperText Markup Language, hay là "Ngôn ngữ Đánh dấu Siêu văn bản") là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web trên World Wide Web. Nó có thể được trợ giúp bởi các công nghệ như CSS và các ngôn ngữ kịch bản giống như JavaScript. Các trình duyệt web nhận tài liệu HTML từ một web server hoặc một kho lưu trữ cục bộ và render tài liệu đó thành các trang web đa phương tiện. HTML mô tả cấu trúc của một trang web về mặt ngữ nghĩa và các dấu hiệu ban đầu được bao gồm cho sự xuất hiện của tài liệu.
  + Các phần tử HTML là các khối xây dựng của các trang HTML. Với cấu trúc HTML, hình ảnh và các đối tượng khác như biểu mẫu tương tác có thể được nhúng vào trang được hiển thị. HTML cung cấp một phương tiện để tạo tài liệu có cấu trúc bằng cách biểu thị ngữ nghĩa cấu trúc cho văn bản như headings, paragraphs, lists, links, quotes và các mục khác. Các phần tử HTML được phân định bằng các tags, được viết bằng dấu ngoặc nhọn. Các tags như <img /> và <input /> giới thiệu trực tiếp nội dung vào trang. Các tags khác như <p> bao quanh và cung cấp thông tin về văn bản tài liệu và có thể bao gồm các thẻ khác làm phần tử phụ. Các trình duyệt không hiển thị các thẻ HTML, nhưng sử dụng chúng để diễn giải nội dung của trang.
  + HTML có thể nhúng các chương trình được viết bằng scripting như JavaScript, điều này ảnh hưởng đến hành vi và nội dung của các trang web. Việc bao gồm CSS xác định giao diện và bố cục của nội dung. World Wide Web Consortium (W3C), trước đây là đơn vị bảo trì HTML và là người duy trì hiện tại của các tiêu chuẩn CSS, đã khuyến khích việc sử dụng CSS trên HTML trình bày rõ ràng kể từ năm 1997



Hình 16: HTML

* CSS:
* CSS là chữ viết tắt của Cascading Style Sheets, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (HTML). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web. Bạn có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng,…thì CSS sẽ giúp chúng ta có thể thêm style vào các phần tử HTML đó như đổi bố cục, màu sắc trang, đổi màu chữ, font chữ, thay đổi cấu trúc…
* CSS được phát triển bởi W3C (World Wide Web Consortium) vào năm 1996, vì HTML không được thiết kế để gắn tag để giúp định dạng trang web.
* Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.



Hình 17: CSS

* Javascript:
* JavaScript là ngôn ngữ lập trình website phổ biến hiện nay, nó được tích hợp và nhúng vào HTML giúp website trở nên sống động hơn. JavaScript đóng vai trò như là một phần của trang web, thực thi cho phép Client-side script từ phía người dùng cũng như phía máy chủ (Nodejs) tạo ra các trang web động.
* JavaScript được phát tiển bởi Brendan Eich tại hãng truyền thông Netscape với tên đầu tiên là Mocha. Sau đó, đổi tên thành LiveScript và cuối cùng là JavaSript được sử dụng phổ biến tới thời điểm bây giờ. Phiên bản mới nhất của JavaScript là ECMAScript (là phiên bản chuẩn hóa của JavaScript). Với ECMAScript 2 phát hành năm 1998 và ECMAScript 3 được ra mắt năm 1999 và hoạt động mạnh mẽ trên mọi trình duyệt và các thiết bị khác nhau.
* Năm 2016, JavaScript đạt kỷ lục đến 92% website sử dụng và được đánh giá từ một ngôn ngữ lập trình riêng trở thành công cụ quan trọng nhất trên các bộ công cụ lập trình web của các lập trình viên. Nếu bạn sử dụng internet khi truy cập vào các website, có thể nhìn thấy các hiệu ứng slide ảnh chuyển động, menu sổ xuống,…đều được tạo nên từ JavaScript.
* Những website có sử dụng JS thì chúng sẽ giúp cho trang web đó có sự tương tác cũng như tăng thêm nhiều trải nghiệm mới cho người dùng. Người dùng cũng có thể tận dụng JS với mục đích là để kiểm tra những input thay vì cách kiểm tra thủ công thông qua hoạt động truy xuất database. Giao diện của ứng dụng phong phú với nhiều thành phần như Drag and Drop, Slider để cung cấp đến cho người dùng một Rich Interface (giao diện giàu tính năng). Giúp thao tác với người dùng phía Client và tách biệt giữa các Client với nhau.



Hình 18: Javascript

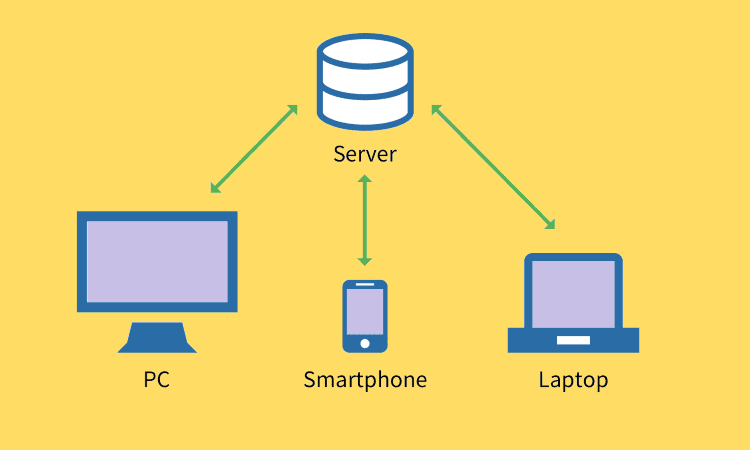
* Vmware:
* VMware là phần mềm máy chủ ảo, đây là một môi trường ảo có chức năng như một hệ thống máy tính ảo với CPU, bộ nhớ, mạng và bộ lưu trữ riêng. Chúng được tạo trên một hệ thống phần cứng vật lý. Phần mềm được gọi là hypervisor chia các tài nguyên của máy khỏi phần cứng và cung cấp một cách thích hợp để VM có thể sử dụng.
* Các máy chủ vật lý, được trang bị hypervisor như KVM được gọi là host. Nhiều máy ảo sử dụng tài nguyên của nó gọi là guest. Hypervisor xử lý các tài nguyên CPU, RAM và Disk như một nhóm tài nguyên. Có thể dễ dàng được di chuyển giữa các máy ảo hiện có hoặc sang các máy ảo mới.
* Các máy ảo được cách ly với phần còn lại của hệ thống và nhiều máy ảo có thể tồn tại trên một phần cứng, như một máy chủ. Chúng có thể được di chuyển giữa các máy chủ lưu trữ. Tùy theo nhu cầu hoặc để sử dụng tài nguyên hiệu quả hơn.
* VMware server được cài đặt và hoạt động như một ứng dụng on top của hệ điều hành Windows hay Linux. Virtualization layer sẽ phân cùng server vật lý sao cho có nhiều máy ảo có thể chạy cùng lúc trên cùng một máy chủ đơn.
* Tài nguyên của máy chủ vật lý trên máy tính được coi là một nhóm tài nguyên thống nhất và được phân bổ cho virtual machines và được kiểm soát.
* VMware Server sẽ tách riêng từng máy chủ ảo khỏi máy chủ của máy ảo VMware với các máy chủ khác, để máy này sẽ không bị ảnh hưởng nếu có máy khác bị treo. Dữ liệu sẽ không bị rò rỉ trên server và ứng dụng chỉ có thể giao tiếp thông qua Internet được cấu hình. VMware Server đóng gói môi trường máy ảo như là các tệp được tập hợp lại, giúp sao lưu các tệp dễ dàng di chuyển và copy.

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Phân tích bài toán

### Mô hình client/server

* Khái niệm:
  + Gồm 2 phần gồm Server và Client. Một chương trình server có thể phục vụ nhiều client đồng thời
  + Về phía Client**:** Nó sẽ đưa ra những yêu cầu cho phía server và nhận đáp ứng từ phía server. Với vai trò là máy khách chúng sẽ không cung cấp tài nguyên đến các máy tính khác mà chỉ sử dụng tài nguyên được cung cấp từ máy chủ. Một client của mô hình này cũng có thể là server cho mô hình khác
  + Về phía Server: Server có thời gian chạy vô tận và chỉ dừng chạy bởi người dùng hoặc tắt máy tính. Sau khi khởi tạo nó sẽ thực hiện việc mở và lắng nghe yêu cầu từ phía Client. Nếu có yêu cầu nó sẽ nhận thực hiện yêu cầu đó và trả về lại cho phía Client
* Nguyên tắc hoạt động:
  + Trong mô hình Client Server, server chấp nhận tất cả các yêu cầu hợp lệ từ mọi nơi khác nhau trên Internet, sau đó trả kết quả về máy tính đã gửi yêu cầu đó.
  + Để máy khách và máy chủ có thể giao tiếp được với nhau thì giữa chúng phải có một giao thức.
  + Một số giao thức hiện tại như HTTP, HTTPs, TCP/IP, FTP…
* Ưu điểm:
  + Client server đảm bảo toàn vẹn dữ liệu khi có sự cố xảy ra.
  + Dễ dàng mở rộng, xây dựng hệ thống mạng.
  + Người dùng có thể lấy dữ liệu từ xa, thao tác như gửi nhận file hay tìm kiếm thông tin.
  + Có khả năng chống quá tải mạng.
  + Giúp hỗ trợ người dùng nhiều dịch vụ đa dạng và sự tiện dụng bởi khả năng truy cập từ xa mà các mô hình cũ không có.
  + Chỉ cần chung định dạng giao tiếp mà không cần chung nền tảng là có thể hoạt động được.
* Nhược điểm:
  + Do phải trao đổi dữ liệu giữa 2 máy tính khác nhau và ở 2 khu vực địa lý cách xa nhau. Vì vậy, khả năng bảo mật thông tin mạng là một hạn chế nữa của Client server. Tuy nhiên vấn đề này đã có một số giao thức đã hỗ trợ bảo mật dữ liệu khi truyền tải. Giao thức được sử dụng phổ biến như HTTPS.
  + Cần bảo trì, bảo dưỡng server thường xuyên.



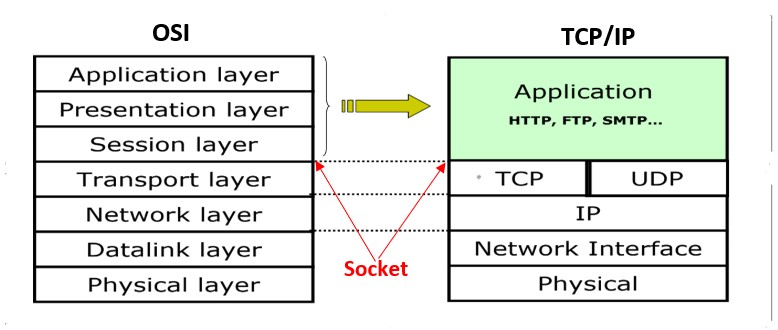
Hình 11: Mô hình client/server

### Giao thức sử dụng

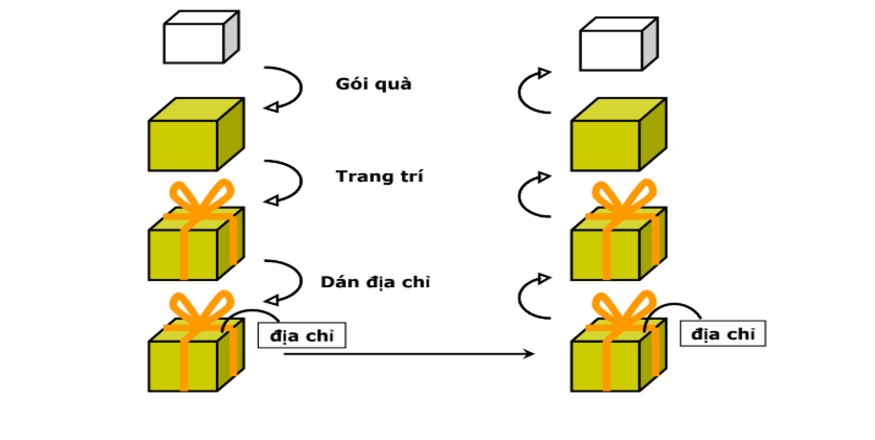
* Giao thức truyền thông gồm 2 loại là giao thức hướng kết nối và giao thức không kết nối
* Giao thức sử dụng đề tài là giao thức hướng kết nối:
  + Truyền thông theo kiểu điểm điểm.
  + Dữ liệu được truyền qua mạng là một dòng byte đi liên tục từ nơi gửi tới nơi nhận.
  + Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự.
  + Quá trình truyền có 3 giai đoạn:
    - 1.Thiết lập kết nối
    - 2.Truyền dữ liệu kèm theo cơ chế kiểm soát chặt chẽ
    - 3.Hủy bỏ kết nối
* Giao thức tiêu biểu là TCP
* Giao thức HTTP: HTTP (HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản) là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW
* Giao thức HTTP là một giao thức tầng ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet).

### Socket là gì ?

* Khái niệm:
  + Socket là giao diện lập trình ứng dụng mạng dùng để truyền và nhận dữ liệu.
  + Socket chính là điểm cuối của một liên kết truyền thông 2 chiều giữa 2 chương trình chạy trên môi trường Internet.
  + Socket dùng để thể hiện kết nối giữa server và client và bị ràng buộc nhau bởi một địa chỉ socket. Một địa chỉ socket bao gồm địa chỉ IP và cổng port
  + Ở đây, Socket chính là cửa giao tiếp giữa tầng ứng dụng và tầng giao vận (Transport layer). Nói cách khác, Socket là giao diện do ứng dụng tạo ra trên máy trạm, quản lí bởi hệ điều hành qua đó các ứng dụng có thể gửi/nhận thông điệp đến/từ các ứng  dụng khác

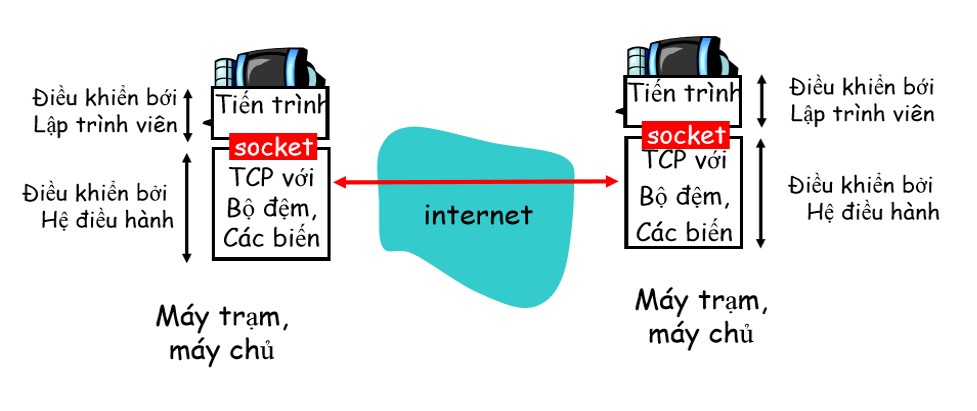


Hình 12: Socket



Hình 13: Socket qua ví dụ

* Stream socket: Còn có tên gọi là socket TCP. Nó hoạt động dựa trên giao thức hướng kết nối mà ở đây là TCP. Tức là chỉ sử dụng được khi mà máy chủ và máy khách có sự liên kết với nhau
  + Ưu điểm:
    - 1. Đảm bảm tính toàn vẹn dữ liệu.
    - 2. Khi thông điệm gửi đi đều phải có thông báo trả về mới tính là hoàn thành.
    - 3.Đảm bảo truyền dữ liệu đúng đối tượng, đúng thử tự.
  + Nhược điểm:
    - 1. Giữa máy chủ và máy nhận chỉ có 1 IP nên khi kết nối 1 máy phải chờ và máy còn lại chấp nhận
* Hoạt động của TCP Socket
  + Đúng như tính chất của TCP chúng ta cần có liên kết 2 chiều trước khi server và client có thể trao đổi thông điệp với nhau.
  + Ban đầu, phía server tạo Socket được ràng buộc với một cổng (port number) để chờ nhận yêu cầu từ phía client.
  + Tiếp đến phía client yêu cầu server bằng cách tạo một Socket TCP trên máy kèm với địa chỉ IP và port number của tiến tình tương ứng trên máy server. Khi client tạo Socket, client TCP tạo liên kết với server TCP và chờ chấp nhận kết nối từ server.
  + TCP cung cấp dịch vụ truyền dòng tin cậy và có thứ tự giữa client và server, giữa máy chủ và máy nhận chỉ có 1 địa chỉ IP duy nhất. Thêm vào đó, mỗi thông điệp truyền đi đều có xác nhận trả về.



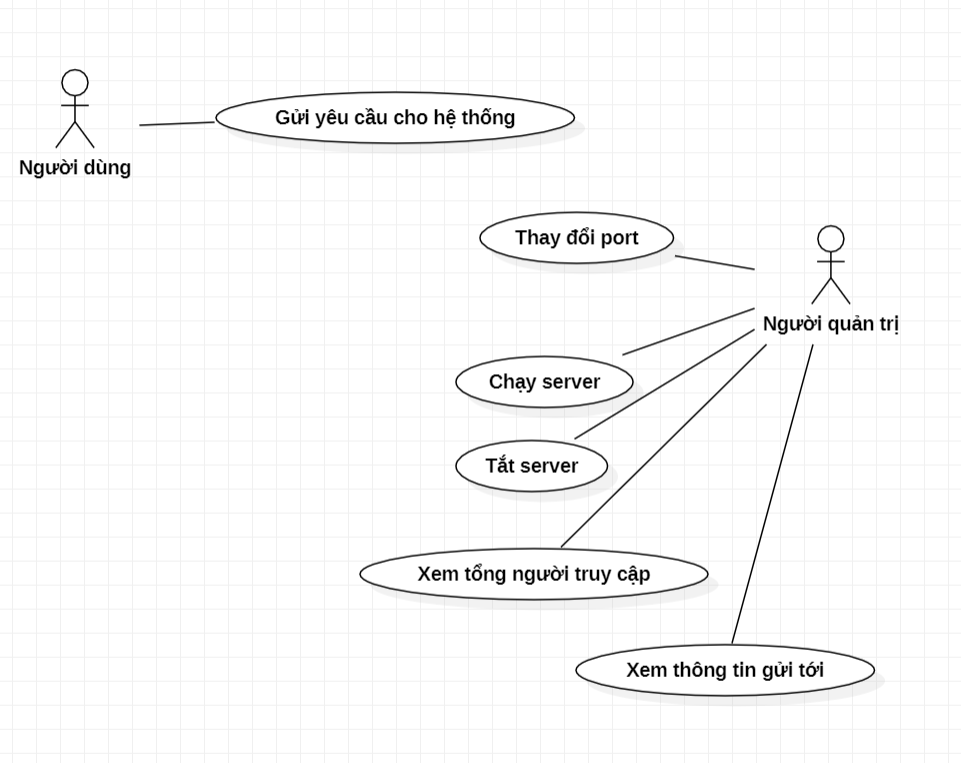
Hình 11: TCP Socket

## Thiết kế hệ thống

### Danh sách các actor và chứng năng

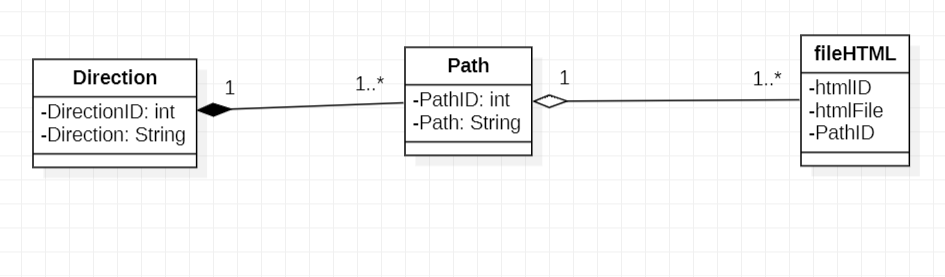
* Người dùng: Gửi yêu cầu lên hệ thống, Upload Website.
* Người quản trị: Bật tắt server, xem thông tin request gửi tới, xem tổng số người đang gửi request, thay đổi thông số port.

### Biểu dồ Usecase



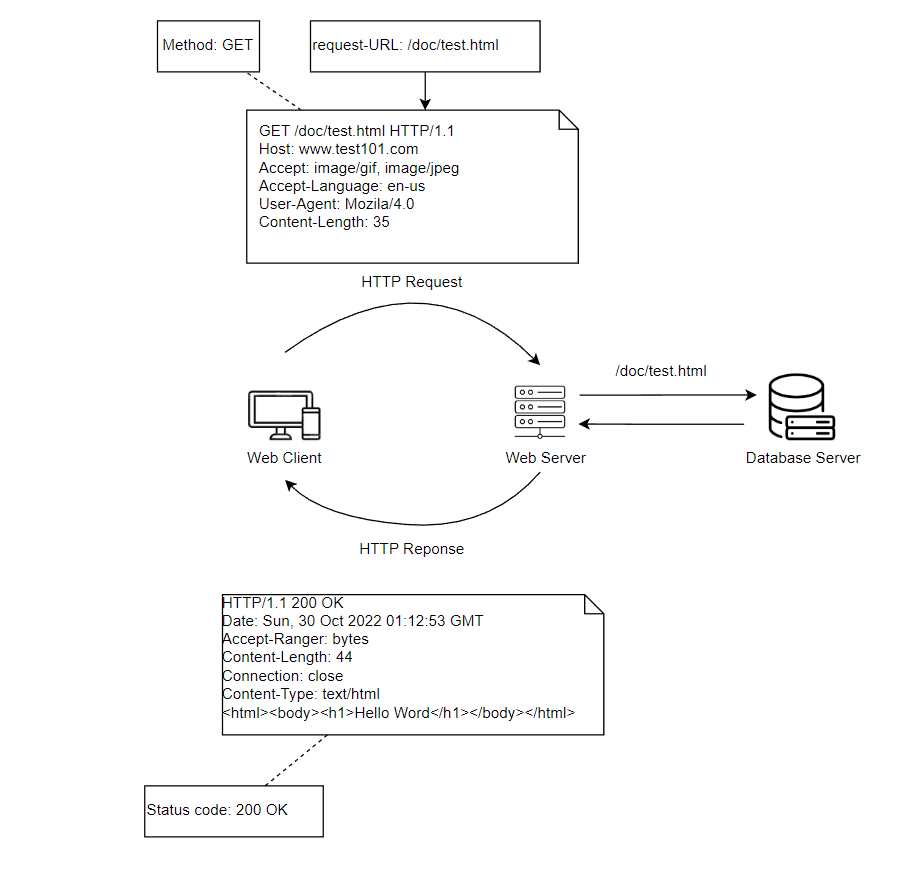
Hình 19: Usecase chính

### Biểu đồ lớp



Hình 20: Biểu đồ lớp

### Mô hình hoạt động



Hình 21: Mô hình hoạt động Web Server

Giải thích mô hình hoạt động Web Server:

* Khi Web Client gửi một HTTP Request cho bên phía Server. Trong HTTP Request đó sẽ chia thành 2 phần chính đó là Request Header và Request Body.
* Trong Request Header sẽ có 2 phần đó là Request Line và Request Header
* Request Line sẽ có Request Method (GET, POST, HEAD, DELETE, PUT), Request URL, HTTP Version
  + Request Method giúp báo cho Server rằng hành động sẽ phải xử lý với thông tin được gửi từ Client lên..
  + Request URL: được sử dụng để xác định duy nhất tài nguyên trên Web
  + HTTP Version là phiên bản giao thức HTTP đang sử dụng
* Request header:
  + Tiếp theo dòng Request-Line là các trường Request-header, cho phép client gửi thêm các thông tin bổ sung về thông điệp HTTP request và về chính client. Một số trường thông dụng như:
    - Accept loại nội dung có thể nhận được từ thông điệp response. Ví dụ: text/plain, text/html…
    - Accept-Encoding: các kiểu nén được chấp nhận. Ví dụ: gzip, deflate, xz, exi…
    - Connection: tùy chọn điều khiển cho kết nối hiện thời. Ví dụ: keepalive,Upgrade…
    - Cookie: thông tin HTTP Cookie từ server
    - User-Agent: thông tin về user agent của người dùng.
* Request Body:
  + Body là dữ liệu mà Client sẽ gửi lên Server, Một request body có thể là một đoạn text thuần túy, HTML, XML, JSON, Javascript, hoặc một tập các cặp key-value dạng form-data.
  + Khi bạn mở một web, trình duyệt sẽ nhận payload dạng HTML, nó chính là giao diện mà chúng ta quan sát được trên trình duyệt đó.
  + Thông thường khi làm việc với các HTTP APIs chúng ta sẽ gửi và nhận các payload dạng JSON hoặc XML.
  + Không phải tất cả các message HTTP đều phải có payload: POST và PUT có thể có, còn với GET và DELETE thì có thể không có payload.
  + Sau khi request server sẽ xử lý và phản hổi (response) tới client theo 1 trong 3 phương pháp:
    - Server phân tích request nhận được, maps yêu cầu với tập tin trong tập tài liệu của server, và trả lại tập tin yêu cầu cho client.
    - Server phân tích request nhận được, maps yêu cầu vào một chương trình trên server, thực thi chương trình và trả lại kết quả của chương trình đó.
    - Request từ client không thể đáp ứng, server trả lại thông báo lỗi.
* Sau khi nhận được yêu cầu và phân tích xong thì nó sẽ lấy Method tương ứng để thực hiện và các yêu cầu mà phía người dùng gửi tới. Từ đó nó sẽ truy xuất cơ sở dữ liệu tương ứng để lấy dữ liệu lên và trả về cho phía Client theo HTTP Reponse.

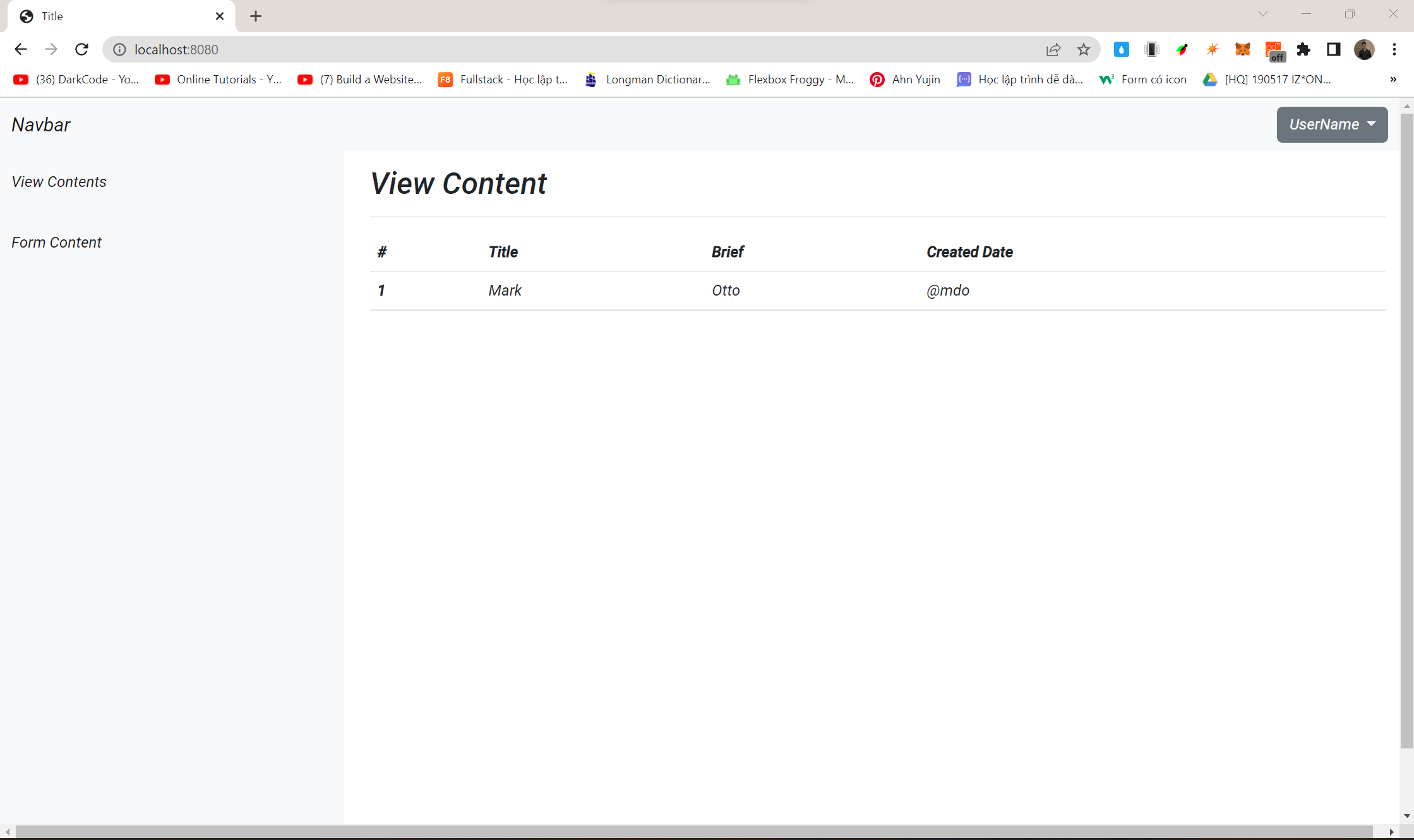
# CHƯƠNG TRÌNH DEMO

## Lý do chọn đề tài

* Đề tài xây dựng một WEB SERVER bao gồm phần server và client. Phía server sẽ hổ trợ các chức năng Authenticate, Cookies, Conditional GET, giúp người quản trị có thể thay đổi được các thông số như port, chạy và dừng server cũng xem các thông số như tổng số client kết nối với server, địa chỉ IP.

## Giới thiệu sản phẩm

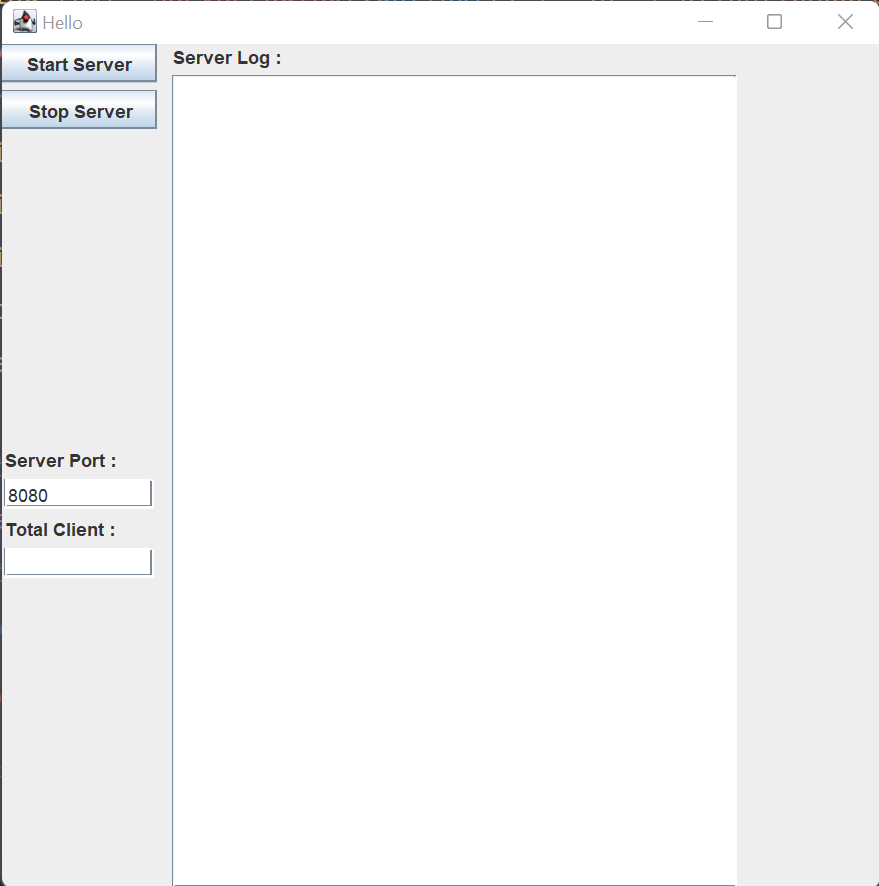
* Phía Client:



Hình 22: Client gửi request lên Server và được trả Reponse

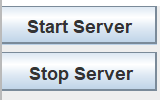
Giải thích: Sau khi người dùng là Client gửi một request lên cho phía Server theo method là GET/POST/PUT … Server sẽ nhận được yêu cầu của Client và từ đó nó truy xuất cơ sở dữ liệu lấy yêu cầu tương ứng của nó lên.

* Phía Server:



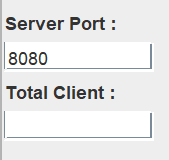
Hình 23: Trang điều khiển của Server

Giải thích: Trang điều khiển giúp cho người quản trị có thể thay đổi thông số Port, tắt mở Server, xem tổng số Client và nội dung Request ở phía người dùng

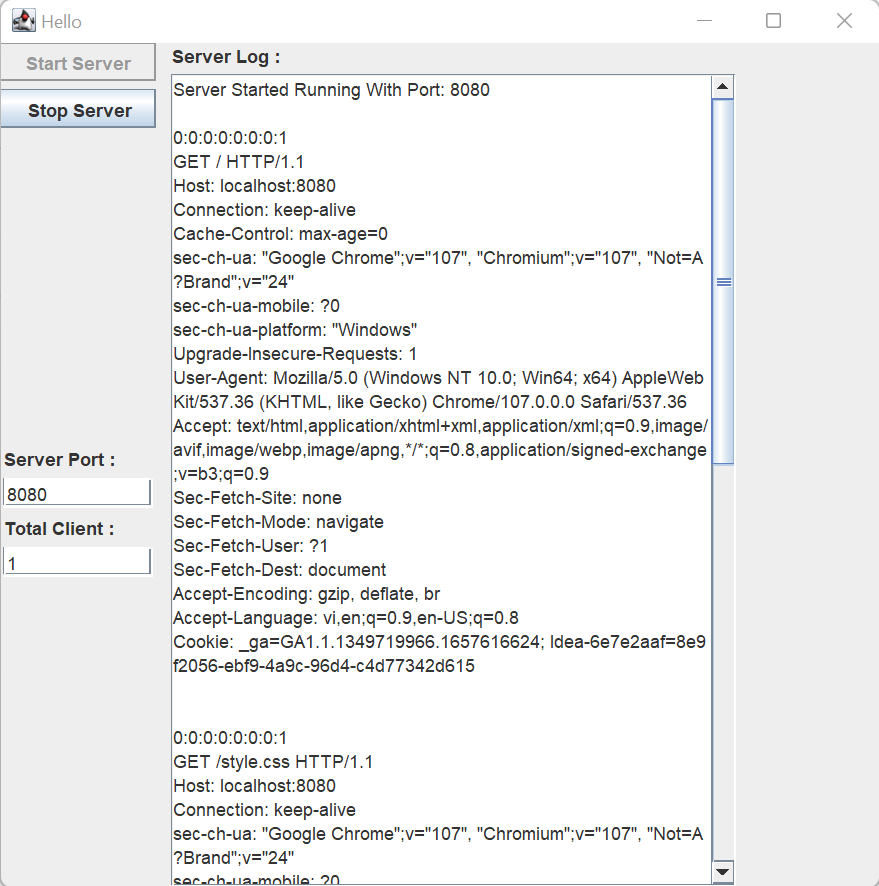


Hình 24: Tiến hành bật và chạy Server

Giải thích: Start Server giúp bật Server giúp người dùng có thể truy cập và gửi yêu cầu. Stop Server đóng Server và mọi hoạt động đều bị dừng lại người dùng không thể gửi yêu cầu.



Hình 25: Cổng port đang đươc sử dụng



Hình 26: Server được chạy và nhận được request của người dùng

Giải thích: Sau khi chạy Server người dùng có thể tiến hành gửi yêu cầu cho phía Server. Server sẽ xem được IP của Client và HTTP Request.

# KIẾN NGHỊ VÀ KẾT LUẬN

1. Kết quả đạt được

* Thực hiện việc tìm hiểu nắm rõ được quy trình hoạt động của Web Server. Từ đó đã xây dựng được Web Server riêng mình với các chức năng cần thiết.

2. Hạn chế

* Các chức năng của Web Server chưa được hoàn thiện một cách chỉnh chu nhất
* Code chưa tinh gọn

3. Kết luận

* Sau thời gian thực hiện việc tìm hiểu về Web Server và tiến hành xây dựng một Web Server cho mình. Nhóm đã nắm được cách thức hoạt đông của một Web Server. Cách hoạt động của các phương thức HTTP. Hiểu được HTTP Server là gì. Các thức gửi yêu cầu và nhận yêu cầu giữa client và phía server. Nắm được các kiến thức về lập trình mạng. Các giao tiếp giữa các mạng với nhau.

4. Hướng phát triển

* Sẽ tích hợp nhiều chức năng hơn nữa cho người dùng.

# TÀI LIỆU THAM KHẢO

1. <https://topdev.vn/blog/web-server/>
2. <https://topdev.vn/blog/nginx-la-gi/>
3. <https://www.hostinger.vn/huong-dan/apache-la-gi-giai-thich-cho-nguoi-moi-bat-dau-hieu-ve-apache-web-server#:~:text=gi%E1%BB%AFa%20server%2Dclient.-,Apache%20Web%20Server%20ho%E1%BA%A1t%20%C4%91%E1%BB%99ng%20nh%C6%B0%20th%E1%BA%BF%20n%C3%A0o%3F,%2C%20Safari%2C%20v%C3%A2ng%20v%C3%A2ng>
4. <https://codelearn.io/sharing/lap-trinh-socket-co-voi-tcpip-ava>