|  |
| --- |
| **TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH**  **KHOA ĐIỆN – ĐIỆN TỬ VIỄN THÔNG** |



**LUẬN VĂN TỐT NGHIỆP**

**NGHIÊN CỨU SỬ DỤNG API TRONG QUẢN TRỊ MẠNG VÀ CẤU HÌNH THIẾT BỊ**

Ngành: **ĐIỆN-ĐIỆN TỬ VIỄN THÔNG**

Chuyên ngành: **ĐIỆN TỬ VIỄN THÔNG**

Giảng viên hướng dẫn: Phạm Thúy Oanh

Sinh viên thực hiện: Nguyễn Duy Mẫn

MSSV: 1651040025 Lớp: DV16

TP. Hồ Chí Minh, 2020

**LỜI NÓI ĐẦU**

Việc quản trị hệ thống đã bắt đầu cách đây 25 năm, các hệ thống mạng được triển khai và cấu hình dùng giao diện dòng lệnh. Phần lớn hiện tại chúng ta vẫn sử dụng CLI hay SNMP. SNMP vẫn là giao thức chủ yếu để giám sát trạng thái của thiết bị và CLI vẫn là phương pháp chủ chốt để quản trị. Tuy nhiên CLI được xây dựng cho con người, không được thiết kế từ đầu cho quản trị tự động dùng máy. Cách dùng Telnet, SSH, và CLI đòi hỏi tương tác giữa người quản trị và hệ thống. Cách cấu hình này đơn lẻ, không có tính lặp lại và không hiệu quả. Tất cả các định dạng được dùng trong CLI là dạng văn bản, không có mô hình dữ liệu chung giữa các nền tảng khác nhau. Các hạn chế này của CLI là nguyên nhân để API hiện thực và phát triển cho quá trình quản trị tự động.

API đã được sử dụng trong lập trình website và thiết kế các phần mềm ứng dụng di động. Khái niệm API sử dụng cho hạ tầng mạng vẫn còn rất mới mẻ nên trong đề tài này em xin trình bày một ứng dụng sử dụng API để quản trị và cấu hình thiết bị mạng

Để hoàn thành được đề tài này, em xin gửi lời cảm ơn đến cô Oanh - giảng viên khoa Điện – Điện Tử Viễn Thông, cô đã chỉ bảo và hướng dẫn để em có hướng đi đúng đắn khi bắt tay vào nghiên cứu đề tài này. Ngoài ra, em cũng xin cảm ơn một số giảng viên trong khoa đã luôn tận tình chỉ bảo và góp ý một cách gián tiếp để em có thêm động lực hoàn thành quyển luận văn tốt nghiệp này.

Với kinh nghiệm còn hạn chế nên khi làm luận văn này, em không thể tránh khỏi những thiếu sót, rất mong nhận được sự đóng góp và chỉ bảo đến từ phía giảng viên để chúng em có thể nâng cao chất lượng đề tài này.

Em xin chân thành cảm ơn!

Hồ Chí Minh, ngày 10 tháng 1 năm 2021

Sinh viên

Nguyễn Duy Mẫn

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

**LỜI MỞ ĐẦU**

1. **Tính cấp thiết của đề tài.**

**Đặt vấn đề**

Thế giới đang thay đổi từng ngày và hạ tầng mạng cũng vậy. Việc doanh nghiệp đang chuyển dần mạng truyền thống sang các mạng thông minh hơn, có tính linh hoạt và bảo mật là nhu cầu tất yếu để phát triển. Việc quản lí một mạng lưới các thiết bị mạng mới và tiên tiến là một bài toán dành cho cả nhà sản xuất thiết bị và doanh nghiệp sử dụng. Nhà sản xuất cần đưa ra một công nghệ quản lí tốt hơn để giải quyết vấn đề khi sử dụng công nghệ cũ SNMP. Vì thế mà API đang được nghiên cứu để thay thế dần các công nghệ cũ.

**Ý nghĩa đề tài**

Giúp công nghệ quản trị mạng bằng API đến gần với thực tế hơn

**Lý do chọn đề tài**

Với mong muốn tiếp cận và nghiên cứu những công nghệ mới sẽ phát triển trong tương lai. Em mong muốn API sẽ được ứng dụng thực tế tại Việt Nam trong tương lai gần.

1. **Mục đích nghiên cứu.**

Tìm hiểu và nghiên cứu kiến thức về lập trình phát triển ứng dụng web. Tìm hiểu và học hỏi các kiến thức mới về quản trị mạng.

1. **Nhiệm vụ nghiên cứu.**

Nghiên cứu và tìm tòi các công nghệ quản lí thiết bị mạng mới sắp được ứng dụng ở tương lai.

1. **Phương pháp nghiên cứu.**

Trao đổi với các thành viên trên cộng đồng mạng để trau dồi kiến thúc về lập trình web và sử dụng API.

Học hỏi cách thức sử dụng và quản lý dữ liệu, cách tạo một ứng dụng web.

1. **Kết quả đạt được.**

Xây dựng ứng dụng web quản lí thiết bị theo gian thực.

Website có một số ứng dụng như xem thông tin thiết bị, xem cấu hình, gửi mail thông báo khi thiết bị khởi động lại, …

Mục lục

[**CHƯƠNG 1. TỔNG QUAN VÀ LÝ DO LỰA CHỌN ĐỀ TÀI** 8](#_Toc61464202)

[**1.1.** **Tổng quan và lý do lựa chọn đề tài** 8](#_Toc61464203)

[**1.2.** **Những khó khăn gặp phải khi thực hiện đề tài** 9](#_Toc61464204)

[**CHƯƠNG 2. TỔNG QUAN ĐỀ TÀI** 10](#_Toc61464205)

[**2.1.** **Tổng quan về API** 10](#_Toc61464206)

[**2.1.1.** **Khái niệm API** 10](#_Toc61464207)

[**2.1.2.** **REST API** 12](#_Toc61464208)

[**2.2.** **Tổng quan về lập trình WEB** 17](#_Toc61464209)

[**2.3.** **Giới thiệu mô hình hệ thống** 18](#_Toc61464210)

[**2.4.** **Tài nguyên Phòng thí nghiệm:** 20](#_Toc61464211)

[**2.5.** **Một số ngôn ngữ và thư viện sử dụng trong đề tài** 24](#_Toc61464212)

[**CHƯƠNG 3. THỰC HIỆN ĐỀ TÀI** 32](#_Toc61464213)

[**3.1.** **Gọi API** 32](#_Toc61464214)

[**3.2.1.** **Xác thực API** 32](#_Toc61464215)

[**3.2.2.** **Sử dụng API** 34](#_Toc61464216)

[**3.2.** **Xây dựng Website** 36](#_Toc61464217)

[**3.2.1.** **Quy trình thực hiện** 36](#_Toc61464218)

[**3.2.2.** **Tạo Route** 39](#_Toc61464219)

[**3.2.3.** **Tạo Controller** 40](#_Toc61464220)

[**3.2.4.** **Tạo Views** 41](#_Toc61464221)

[**Sử lí data** 42](#_Toc61464222)

[**Sử lí data=ajax** 42](#_Toc61464223)

[**3.3.** **Kết quả và định hướng phát triển** 44](#_Toc61464224)

[Code: 48](#_Toc61464225)

[Tài liệu tham khảo 48](#_Toc61464226)

MỤC LỤC HÌNH

DANH MỤC TỪ VIẾT TẮT

|  |
| --- |
| AJAX |
| API |
| CLI |
| CSS |
| HTML |
| HTTP |
| HTTPS |
| IoT |
| JS |
| MPLS |
| MVC |
| PHP |
| REST |
| RESTful |
| SD-WAN |
| SNMP |
| SSH |
| SSL |
| TCP/IP |
| TLS |
| URL |
| XAMPP |
| XML |

# **CHƯƠNG 1. TỔNG QUAN VÀ LÝ DO LỰA CHỌN ĐỀ TÀI**

* 1. **Tổng quan và lý do lựa chọn đề tài**

Những năm gần đây, các doanh nghiệp đang thay đổi. Họ xây dựng các ứng dụng để phục vụ nhu cầu kinh doanh thay đổi quá nhanh hàng ngày. Các ứng dụng muốn nói chuyện trực tiếp với hạ tầng mạng. Trong các trung tâm dữ liệu, số lượng thiết bị cần quản trị càng ngày càng lớn. Một người kỹ sư Amazon giờ đây phải quản trị 250 ngàn node, so với yêu cầu cách đây 15 năm anh ta chỉ cần quản lý 250 node. Trong thế giới của IoT, làm thế nào chúng ta có thể truy cập và cấu hình xe hơi, đèn, cửa, tủ lạnh, đồ chơi…. Rõ ràng, chúng ta cần giao tiếp giữa máy với máy, chúng ta cần Network Automation. Do đó, lĩnh vực Networking đầy bảo thủ cũng đã bắt đầu cố gắng cải tiến đi theo hướng tự động hóa để đáp ứng nhu cầu thay đổi. Có hai hướng tiếp cận:

* Hướng đầu tiên, các người kỹ sư mạng cố gắng tự động hóa các tác vụ quản trị, cấu hình bằng phần mềm nhưng vẫn tận dụng CLI. Họ dùng các Python scripts để cấu hình các thiết bị thông qua CLI trong một phiên Telnet/SSH. Ví dụ, một script có thể có dạng tương tác với thiết bị như sau khi cố gắng truy cập vào thiết bị. Script sau đó sẽ cố gắng gửi một tập hợp các lệnh đến CLI sau khi login thành công. Cách dùng script dạng này thì cũng hữu dụng. Tuy nhiên nó khó viết, cần người viết có kiến thức chuyên sâu về lập trình, sửa lỗi và kiến thức Netwrok. Nếu thiết bị mạng của một hãng cho ra kết quả trả về của một lệnh theo định dạng kiểu khác, người kỹ sư mạng sẽ phải tìm ra sự thay đổi này và sửa đổi các phần script tương ứng. Ngoài ra, nếu trong môi trường hạ tầng gồm nhiều thiết bị của các hãng khác nhau, số script cần viết tăng nhanh tương ứng.
* Ở hướng nghiên cứu thứ hai, các kỹ sư mạng tập trung vào việc cải tiến phần giao tiếp giữa các công cụ phần mềm với các thiết bị phần cứng. Phần giao tiếp này có tên gọi là API (Application Programming Interfaces – API). API là một cơ chế giúp phần mềm trên một thiết bị này nói chuyện với phần mềm trên một thiết bị khác. API được dùng rất nhiều trên Internet và trong thế giới của các nhà phát triển ứng dụng, trong thế giới của phần mềm. API hiện diện ở khắp nơi, ngoại trừ trong các thiết bị mạng. Tuy nhiên những năm gần đây, các hãng sản xuất thiết bị mạng viễn thông đều đã chú ý hỗ trợ APIs trong các sản phẩm của mình. Một thiết bị có hỗ trợ giao tiếp APIs sẽ giúp các công cụ phần mềm quản lý truy xuất trực tiếp các thông tin cấu hình, các trạng thái hoạt động và các dòng lưu lượng trên thiết bị*.* API là giao tiếp giữa phần mềm với phần mềm. Các thiết bị mạng hiện đại hầu hết đều hỗ trợ APIs.

API sẽ sớm trở thành một công cụ chủ chốt để cấu hình và quản trị các thiết bị mạng trong một tương lai rất gần. Do đó việc nghiên cứu đề tài: “NGHIÊN CỨU SỬ DỤNG API TRONG QUẢN TRỊ MẠNG VÀ CẤU HÌNH THIẾT BỊ” là rất cần thiết.

* 1. **Những khó khăn gặp phải khi thực hiện đề tài**

Từ 25 năm nay, để quản trị và cấu hình hạ tầng mạng, người kỹ sư mạng phải dùng các giao tiếp dòng lệnh chậm chạp gọi là ‘giao diện dòng lệnh’(Command Line Interface – CLI). CLI, ngay ngày đầu, được thiết kế cho giao tiếp giữa người và máy. Khi người kỹ sưtelnet/SSH/console đến router, thứ mà họ thường được thấy trước tiên chính là CLI.Bên cạnh telnet/SSH, các phần mềm quản trị mạng hiện tại cũng dùng giao thức SNMP để thu thập thông tin về các thành phần của hệ thống mạng. SNMP thật ra cũng được dùng để cấu hình các thiết bị mạng, tuy nhiên tính năng này của SNMP ít được khai thác do khả năng hỗ trợ thay đổi các thông số hoạt động của thiết bị rất hạn chế. CLI và SNMP đã sử dụng lâu dài nên việc API thay thế cả hai trong việc quản lí và cấu hình là không dễ dàng. Bên cạnh đó, hạ tầng thiết bị lỗi thời và không hỗ trợ API cũng là khó khăn trong việc thực hiện đề tài.

# **CHƯƠNG 2. TỔNG QUAN ĐỀ TÀI**

* 1. **Tổng quan về API**

1. **Khái niệm API**

Để giao tiếp và định cấu hình mạng, các nhà phát triển phần mềm đã nghiên cứu sử dụng ứng dụng các giao diện lập trình (API- Application Programming Interfaces). API là cơ chế được sử dụng để giao tiếp với các ứng dụng và phần mềm. Chúng cũng được sử dụng để giao tiếp với các thành phần khác nhau của mạng thông qua phần mềm. Mặc dù được dịnh nghĩa là giao diện nhưng API thực tế chỉ là các hàm hay thủ tục thông thường. Các hàm này được viết trên nhiều ngôn ngữ lập trình khác nhau như Python, .NET, PHP, … Người sử dụng gửi yêu cầu truy cập API request đến máy chủ để yêu cầu thông tin và nhận được trả lời API response từ máy chủ cùng với thông tin được yêu cầu.

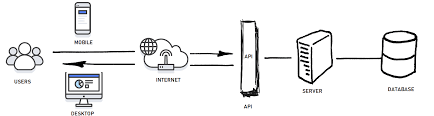


Hình 2. 1 API

Các loại API:

* **Web API:** Là hệ thống API được sử dụng trong các hệ thống website, chẳng hạn: Google, Facebook… Hầu hết các website đều cung cấp hệ thống API cho phép bạn kết nối, lấy dữ liệu hoặc cập nhật cơ sở dữ liệu. Phần lớn các trang web API được thiết kế theo tiêu chuẩn [RESTful](https://thietbitudong.com.vn/restful-api-la-gi/). Và Cisco cũng đã đưa ra hệ thống API của riêng mình được sử dụng trên nhiều thiết bị mạng mới sau này của họ. Ví dụ như API Meraki dùng trong các thiết bị bao gồm không dây, chuyển mạch, bảo mật, quản lý di động doanh nghiệp và camera an ninh, API Webex dùng trong hội nghị truyền và hội nghị truyền hình web, API cho SD-WAN, …
* **API trên hệ điều hành**: Khái niệm này có trước cả web API, Microsoft cung cấp các hệ điều hành Windows cùng các tài liệu API là đặc tả các hàm, phương thức, lời gọi hàm cũng như các giao thức kết nối cho người dùng, giúp người dùng API có thể tạo ra các phần mềm ứng dụng có thể tương tác trực tiếp với hệ điều hành Windows. Ngoài ra Windows, API còn có trên cả hệ điều hành MAC OS và LINUX.
* **API của thư viện phần mềm (framework):** API mô tả và quy định các hành động mong muốn mà các thư viện cung cấp. Một API có thể có nhiều cách triển khai khác nhau, giúp cho một chương trình viết bằng ngôn ngữ này có thể sử dụng được thư viện viết bằng ngôn ngữ khác.

**Khi nhắc tới API thì ta thường nghĩ đến Web API, bởi vì Web API được sử dụng rộng rãi và thông dụng. Web API** là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp, trao đổi dữ liệu qua lại. Dữ liệu được Web API trả lại thường ở dạng [JSON](https://topdev.vn/blog/json-la-gi/) hoặc XML thông qua giao thức HTTP hoặc HTTPS.



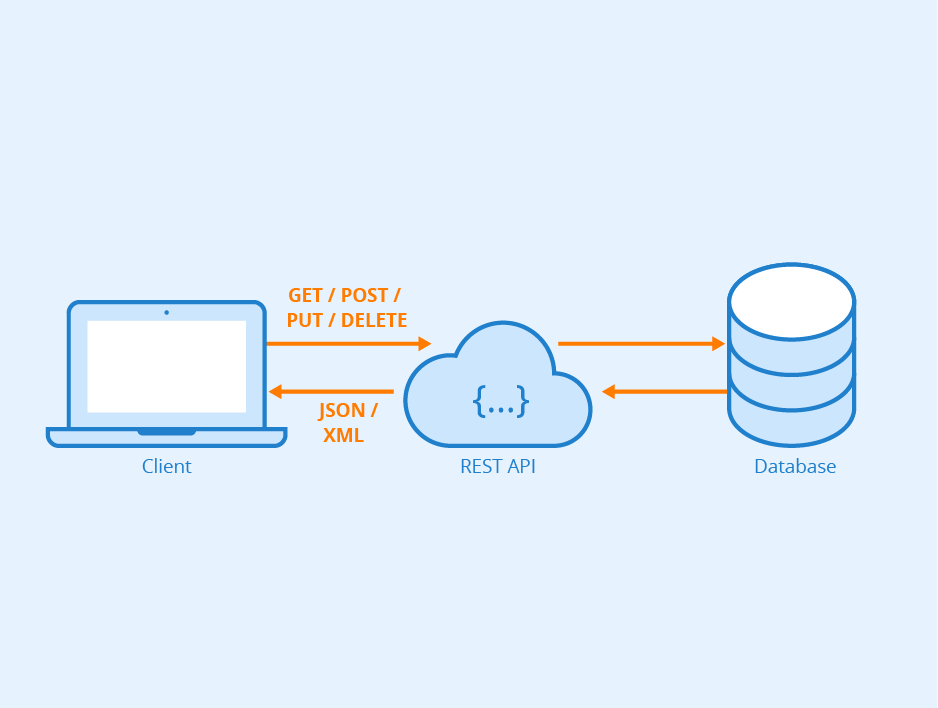
Hình 2. 2 Mô hình Web API

**Cách Web API hoạt động:**

* Đầu tiên là xây dựng URL API để bên thứ ba có thể gửi yêu cầu dữ liệu đến máy chủ cung cấp nội dung, dịch vụ thông qua giao thức HTTP hoặc HTTPS.
* Tại web server cung cấp nội dung, các ứng dụng nguồn sẽ thực hiện kiểm tra xác thực nếu có và tìm đến tài nguyên thích hợp để tạo nội dung trên cơ sở dữ liệu và trả về kết quả.
* Máy chủ trả về kết quả theo định dạng JSON hoặc XML tùy theo yêu cầu của người dùng thông qua giao thức HTTP/HTTPS.
* Tại nơi yêu cầu ban đầu là ứng dụng web hoặc ứng dụng di động, dữ liệu JSON/XML sẽ được phân tích để lấy dữ liệu. Sau khi có được dữ liệu thì thực hiện tiếp các hoạt động như lưu dữ liệu xuống cơ sở dữ liệu, hiển thị dữ liệu…

1. **REST API**

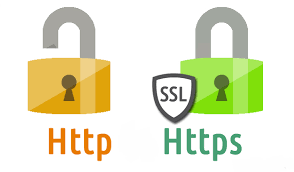
REST là viết tắt của REpresentational State Transfer. Tiến sĩ Roy Fielding (đồng sáng lập giao thức HTTP) đã đặt ra thuật ngữ này vào đầu thế kỷ này, ông đã phác thảo REST trong luận văn tiến sĩ của mình. Nó là một dạng chuyển đổi cấu trúc dữ liệu, là một kiểu kiến ​​trúc dành cho việc thiết kế các ứng dụng có kết nối. Có những loại API khác như SOAP, nhưng REST dễ dàng tích hợp hơn. Nó sử dụng giao thức HTTP để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, … đến một URL để [xử lý dữ liệu](https://movan.vn/quan-ly-co-du-lieu-cho-pheu-ban-hang/).



Hình 2. 3 REST API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế dịch vụ Web) để tiện cho việc quản lý các tài nguyên. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, …), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP. **REST** có chức năng quan trọng nhất là quy định cách sử dụng các phương thức HTTP (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các tài nguyên. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một **RESTful API**. REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

HTTP - HyperText Transfer Protocol (giao thức truyền tải siêu văn bản). Đây là một giao thức ứng dụng được sử dụng thường xuyên nhất trong bộ các giao thức TCP/IP (gồm một nhóm các giao thức nền tảng cho internet).



Hình 2. 4 HTTP và HTTTPS

HTTP hoạt động dựa trên **mô hình Client (máy khách) - Server (máy chủ)**. Máy khách (Client) là các máy tính của người dùng. Sau một thao tác nào đó của người dùng như nhấn, nhập dữ liệu,… các máy khách(**Client)** sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này. Nếu máy chủ không trả lời hoặc báo lỗi thì máy khách sẽ bỏ qua và tiếp tục những hoạt động tiếp theo.

**HTTPS**là phần mở rộng **bảo mật**của HTTP. Website được cài đặt chứng chỉ SSL/TLS có thể dùng giao thức HTTPS để thiết lập kênh kết nối an toàn tới máy chủ. **SSL** là chữ viết tắt của **Secure Sockets Layer (Lớp socket bảo mật).** Một loại bảo mật giúp mã hóa liên lạc giữa website và trình duyệt. Công nghệ này đang **lỗi thời**và được thay thế hoàn toàn bởi **TLS. Còn TLS**là chữ viết tắt của **Transport Layer Security,**nó cũng giúp bảo mật thông tin truyền giống như SSL.

Một **URL** (Uniform Resource Locator) dịch ra là định vị tài nguyên thống nhất, được gọi một cách thông thường là một địa chỉ web, là một tham chiếu đến tài nguyên web chỉ định vị trí của nó trên một mạng máy tính và cơ chế để truy xuất nó. Một URL có cấu trúc như sau:

protocol://domain:port/path?**query string#fragment**

Trong một **URL** có nhiều thành phần:

* **Protocol**: giao thức tầng ứng dụng được sử dụng bởi client và server. Thường sử dụng HTTP hoặc HTTPS
* **Domain**: tên miền hoặc một địa chỉ IP.
* **Port (cổng kết nối):** Khi một máy chủ chạy nhiều dịch vụ khác nhau thì cần phải biết chính xác port thì mới kết nối được đến dịch vụ tương ứng trên máy chủ đó. Ví dụ dịch vụ web thường có port là 80.
* **Path (đường dẫn tuyệt đối**): Là địa chỉ chính xác chứa dữ liệu, thường được tổ chức theo cấp bậc, xuất hiện dưới dạng một chuỗi các phân đoạn được cách nhau bởi dấu gạch chéo
* **Query string (truy vấn):** Có thể có hoặc không. Một truy vấn được tách ra từ phần trước bằng một dấu chấm hỏi (?), chứa một chuỗi truy vấn của dữ liệu không theo cấp bậc. Cú pháp của nó không được định nghĩa rõ ràng nhưng theo quy ước thì nó thường là một dãy các cặp giá trị thuộc tính được phân cách bởi dấu chấm phẩy (;) hoặc ký tự &.
* **Fragment** (chỉ định mục con): Có thể có hoặc không. Được tách ra từ phần trước bởi một dấu thăng (#). Trong trang web thì chỉ định mục con thường là một thuộc tính ID của một phần tử cụ thể giúp cho người dùng dễ dàng di chuyển đến vị trí của phần tử đó.
* HTTP – Requests

**HTTP Request Method**: Là phương thức để chỉ ra hành động mong muốn được thực hiện trên tài nguyên đã xác định.

Cấu trúc của một HTTP Request:

* Một **Request-line** = **Phương thức** + **URI–Request** + **Phiên bản HTTP**. Giao thức HTTP định nghĩa một tập các giao thức GET, POST, PUT ... máy khách có thể sử dụng một trong các phương thức đó để gửi yêu cầu lên máy chủ.
* Có thể có hoặc không các trường **header.**
* Một dòng trống để đánh dấu sự kết thúc của các trường **Header**.

**Request Header Fields**: Các trường header cho phép client truyền thông tin bổ sung về yêu cầu, và về chính máy khách, đến máy chủ. Một số trường: Accept-Charset, Accept-Encoding, Accept-Language, Authorization, Expect, From, Host, …

* Tùy chọn một thông điệp

Khi yêu cầu đến máy chủ, máy chủ thực hiện một trong 3 hành động sau:

Máy chủ phân tích yêu cầu nhận được, maps yêu cầu với tập tin trong tập tài liệu của máy chủ, và trả lại tập tin yêu cầu cho máy khách.

Máy chủ phân tích yêu cầu nhận được, maps yêu cầu vào một chương trình trên máy chủ, thực thi chương trình và trả lại kết quả của chương trình đó.

Yêu cầu từ máy khách không thể đáp ứng, máy chủ trả lại thông báo lỗi.

Tiêu chuẩn HTTP method trong REST

* GET (SELECT): Truy xuất hoặc đọc thông tin.
* POST (CREATE): Tạo mới một đối tượng.
* PUT (UPDATE): Cập nhật toàn phần.
* PATCH (UPDATE: Cập nhật một phần.
* DELETE (DELETE): Xoá một đối tượng.
* HTTP - Responses

Cấu trúc của một HTTP response:

* Một **Status-line** = **Phiên bản HTTP** + **Mã trạng thái** + **Trạng thái**
* Có thể có hoặc không có các trường header
* Một dòng trống để đánh dấu sự kết thúc của các trường header
* Tùy chọn một thông điệp

**Mã trạng thái**: Thông báo về kết quả khi nhận được yêu cầu và xử lí bên máy chủ cho máy khách.

* Tiêu chuẩn mã trạng thái trong REST

Khi chúng ta yêu cầu một API nào đó thường thì sẽ có vài status code để nhận biết sau

* Status code 2xx (thành công)

Khi yêu cầu thành công mà không gặp bất kì lỗi nào.

* 200 – OK: Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.
* 201 – Created: Tạo mới một dữ liệu
* 204 - No Content: Không có dữ liệu nào được trả về
* Status code 3xx (điều hướng)

Máy chủ phải thực hiện hành động bổ sung để hoàn thành yêu cầu.

* 301 - Permanent Redirect: chuyển hướng hoàn toàn
* 302 - Temporarily Redirect: chuyển hướng tạm thời
* 304 - Not Modified: Trả về từ cache
* Status code 4xx (lỗi người dùng)

Lỗi này thường về phía người dùng (người sử dụng trình duyệt web).

* 400 - Bad Request: Yêu cầu không hợp lệ.
* 401 – Unauthorized: Yêu cầu không có bảo mật.
* 403 – Forbidden: Không được phép truy cập.
* 404 - Not Found: Không tìm thấy bất kì thông tin nào
* 405 - Method Not Allowed: Phương thức (GET, POST, PUT, PATCH, DELETE) không cho phép.
* 415 - Unsupported Media Type: Không hỗ trợ kiểu tài nguyên này.
* 422 - Unprocessable Entity: Dữ liệu không được xác thực.
* 429 - Too Many Requests: Yêu cầu bị từ chối do bị giới hạn.
* Status code 5xx (lỗi máy chủ)

Lỗi này thuộc về phía máy chủ, có thể đến từ code, tràn bộ nhớ, hết dung lượng, …

* 500 – Internal Server Error: Máy chủ bị lỗi.
* 501 – Not implement: Chưa thực hiện được.
* 502 – Bad Gateway: đa số là do lỗi máy chủ (Nginx, Apache, Lite Speed, …)
* 503 – Service Unavailable: Máy chủ tạm thời không xử lý được.

1. **Ưu, nhược điểm của API khi cấu hình:**
   * **Ưu điểm: API sử dụng được với các hệ thống lớn như các công ty mạng viễn thông hay công ty đa quốc gia. Việc giám sát hệ thống sẽ dễ dàng hơn khi sử dụng API. Việc giám sát một cách chi tiết và theo thời gian thực giúp cho người kĩ sư mạng có thể nắm bắt các tình huống không lường trước. Do đó việc khắc phục sự cố trong hệ thống sẽ dễ dàng hơn.**
   * **Nhược điểm: khi sử dụng API thì cần người kĩ sư hiểu biết rõ về nó, đồng thời còn cần thêm các kĩ năng khác như lập trình, tư duy logic, phán đoán tình huống, … Đặc biệt, API là công nghệ mới nên cần thời gian để tiếp tục phát triển và ứng dụng rộng rãi. Việc thay đổi hệ thống mạng mới để sử dụng được API thì nhu cầu về tiền bạc và kĩ sư có trình độ cao rất lớn. API có chức năng chính là thu thập dữ liệu và giám sát hệ thống từ xa nên việc cấu hình thiết bị phụ thuộc nhiều vào các công cụ hỗ trợ cấu hình như Ansible.**
   1. **Tổng quan về lập trình WEB**

Website còn gọi là trang web, hoặc trang mạng, khác với thuật ngữ tiếng anh “web page”, là một tập hợp trang web, thường chỉ nằm trong một tên miền hoặc tên miền phụ lưu trữ trên các máy chủ chạy online, trên đường truyền World Wide Web của Internet.

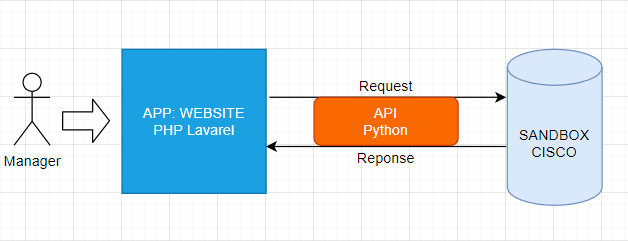
Lập trình web là công việc có nhiệm vụ nhận toàn bộ dữ liệu từ bộ phận thiết kế web để chuyển thành một hệ thống website hoàn chỉnh có tương tác với cơ sở dữ liệu và tương tác với người dùng dựa trên ngôn ngữ máy tính. Việc hiển thị của trang web (hình ảnh, màu sắc, font chữ, …) sao cho thân thiện với trải nghiệm của người dùng được gọi là **Front-end.**

**Việc xử lí các logic nghiệp vụ phức tạp đảm bảo website hoạt động bình thường được gọi là Back-end. Để xây dựng một khung website hoàn chỉnh có đầy đủ chức năng từ Front-end đến Back-end thì em sử dụng PHP làm ngôn ngữ chính để tạo web và sử dụng Lavarel. Laravel**là một PHP Framework mã nguồn mở miễn phí, được phát triển bởi Taylor Otwell với phiên bản đầu tiên được ra mắt vào tháng 6 năm 2011. **Laravel** ra đời nhằm mục đích hỗ trợ phát triển các ứng dụng web, dựa trên mô hình MVC (Model – View – Controller).

**Để xây dựng giao diện dùng người dùng trong đề tài thì em sử dụng** 3 ngôn ngữ chủ đạo đó là: HTML, CSS và Javascript. Ngoài ra em còn sử dụng Bootstrap để giúp việc thiết kế website đơn giản và nhanh hơn. Bootstrap bao gồm các mẫu thiết kế dựa trên HTML và CSS cho kiểu chữ, biểu mẫu, nút, bảng, điều hướng và nhiều thứ khác.

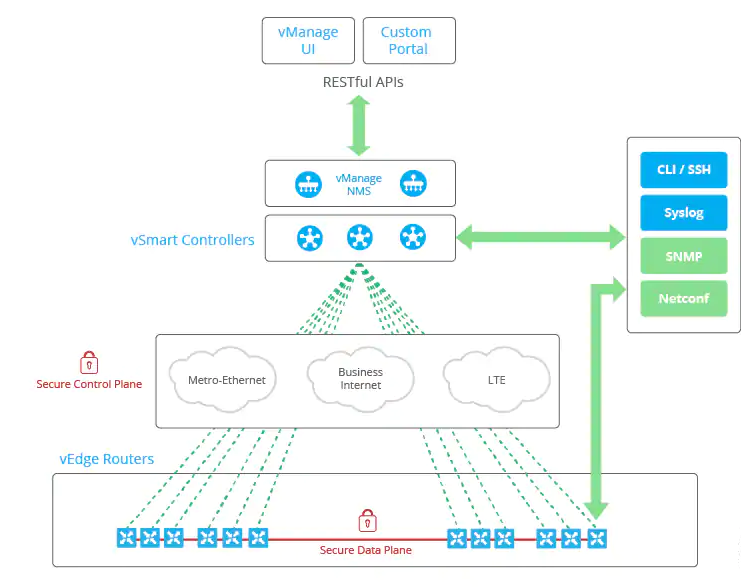
* 1. **Giới thiệu mô hình hệ thống**
  2. **Mô hình của đề tài**

**Mô hình của đề tài**

****

**Người quản lí sẽ sử dụng website riêng để quản lí và có thể cấu hình một thiết bị mạng. Trang web này được thiết kế dựa vào Lavarel framework của ngôn ngữ PHP. PHP là một ngôn ngữ mạnh để lập trình web. Trong khi việc sử dụng API dựa vào Python, Python là ngôn ngữ lập trình bậc cao, được nhà cung cấp Cisco giới thiệu để sử dụng API một cách nhanh chóng và tiện lợi.**

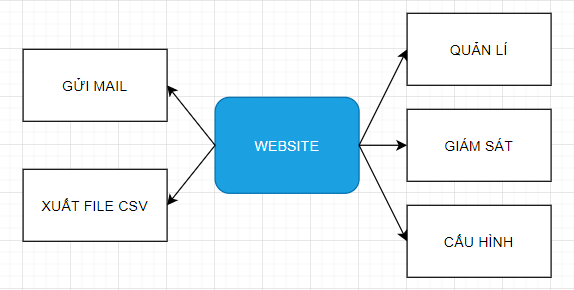
* + 1. **Mô hình mạng SD-WAN**

****

Cisco SD-WAN bao gồm hệ thống quản lý mạng vManage (Management Plane), bộ điều khiển vSmart (Control Plane), bộ điều phối vBond (Orchestration Plane) và bộ định tuyến vEdge (Data Plane). VManage NMS là một hệ thống quản lý mạng tập trung cung cấp giao diện để dễ dàng theo dõi, cấu hình và duy trì tất cả các thiết bị. Phần mềm VManage NMS hỗ trợ RESTful API. Bộ điều khiển tập trung, được gọi là Cisco vSmart Controllers, giám sát mặt phẳng điều khiển của cấu trúc Cisco SD-WAN, quản lý hiệu quả việc cung cấp, bảo trì và bảo mật cho toàn bộ mạng lớp phủ. Một thiết bị khác, được gọi là Cisco vBond Orchestrator, tự động xác thực tất cả các thiết bị Cisco vEdge khác khi chúng tham gia vào mạng. vEdge router là các thiết bị định tuyến Router vật lý hoặc các thiết bị Router ảo, được đặt tại các chi nhánh, văn phòng hoặc trên đám mây và cung cấp các kết nối an toàn giữa các site thông qua một hoặc nhiều đường WAN. Nó chịu trách nhiệm chuyển tiếp lưu lượng, bảo mật, mã hóa, Chất lượng dịch vụ (QoS), các giao thức định tuyến như Border Gateway Protocol (BGP) và Open Shortest Path First (OSPF), ...

* + 1. **Tính năng ứng dụng**

**Tính năng của trang web**

****

**Trang web có đầy đủ tính năng của một ứng dụng quản lí từ xa :**

* **Quản lí thời gian thực**
* **Giám sát trạng thái, hoạt động**
* **Cấu hình đơn giản**
* **Gửi mail thông báo**
* **Xuất thông tin thiết bị ra file CSV**
  1. **Tài nguyên**

**Phòng thí nghiệm**

**Giới thiệu:**

DevNet Sandbox cung cấp các phòng thí nghiệm đóng gói miễn phí và chứa các công nghệ mới của Cisco. Sandbox dịch nghĩa là hộp cát, sau mỗi lần thực hành thử nghiệm, các phòng thí nghiệm được trả lại thiết kế ban đầu của nó như lúc bắt đầu. Trên DevNet Sand box ta có thể gọi API, thử nghiệm cấu hình Ansible, cấu hình các thiết bị mới, …. Để truy cập Sandbox ta truy cập vào: <https://developer.cisco.com/site/sandbox/>

Phân loại

Có 2 loại hộp cát:

* Always-On Sandboxes
* Reservation Sandboxes



Always-On Sandboxes (hộp cát luôn bật): luôn sẵn sàng hoạt động khi ta cần. Nhưng môi trường này được **chia sẻ** giữa tất cả người dùng nên quyền truy cập quản trị bị hạn chế. Loại hộp cát này bị giới hạn về tài nguyên chia sẻ nên ít được sử dụng.

Reservation Sandboxes (hộp cát đặt trước): Hộp cát đặt trước là môi trường truy cập riêng tư và yêu cầu đặt trước. Các hộp cát đóng gói sẵn này thường bao gồm thiết bị hỗ trợ cho công nghệ được đánh dấu! Các hộp cát này chứa mạng, miền, độ phân giải tên, máy chủ phát triển, ... Hộp cát đặt chỗ được truy cập bằng cách kết nối với VPN và toàn quyền truy cập quản trị được cấp cho tất cả thiết bị trong hộp cát.

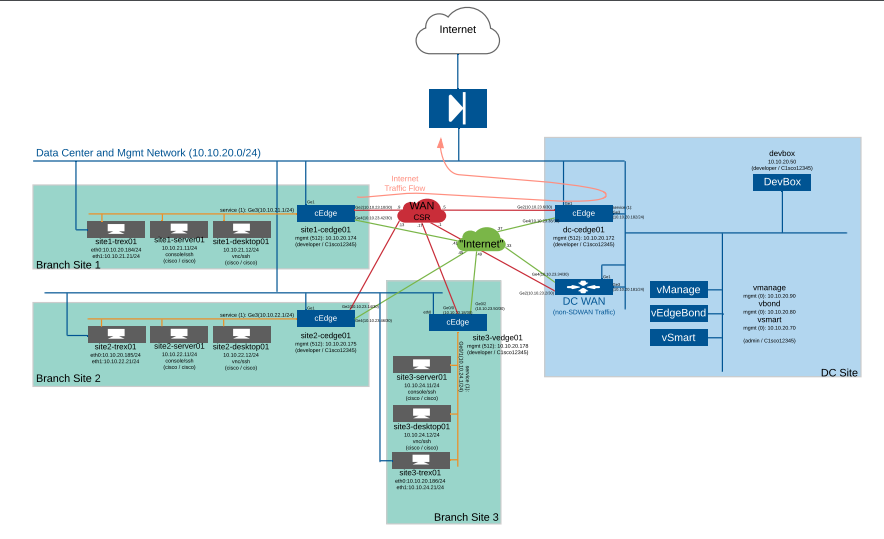
Ứng dụng

Trong đề tài này em chọn hộp cát đặt trước: Cisco SD-WAN Sandbox để phục vụ cho đề tài này. Cisco SD-WAN Sandbox bao gồm một môi trường SD-WAN ảo hoàn chỉnh và tất cả các thành phần của nó mà các nhà phát triển cam kết sử dụng để phát triển, gỡ lỗi và kiểm tra các mẫu SD-WAN ứng dụng của Cisco. Trong hộp cát này, có thể:

* Phát triển / thử nghiệm các ứng dụng loại SD-WAN
* Tương tác với các lệnh gọi API SD-WAN

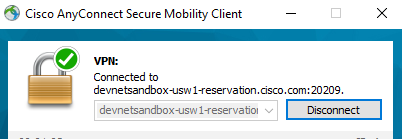
Thành phần

Môi trường hộp cát SD-WAN được tạo sẵn với cấu trúc: 1 bộ định tuyến DC / Trung tâm khu vực IOS-XE SD-WAN (CSR1000v), 2 bộ định tuyến nhánh IOS-XE SD-WAN (CSR1000v) với 1 bộ định tuyến tại mỗi nhánh và 1 bộ định tuyến nhánh vEdge Cloud. Tất cả các bộ định tuyến này đều có cấu hình khuôn mẫu để tận dụng 2 mạch truyền tải MPLS và Internet công cộng.



Cách kết nối:

Để kết nối VPN vào phòng thí nghiệm, ta cần phải cài đặt phần mềm VPN ứng với Cisco Sandbox. Đó là phần mềm Cisco AnyConnect Secure Mobility Client

.

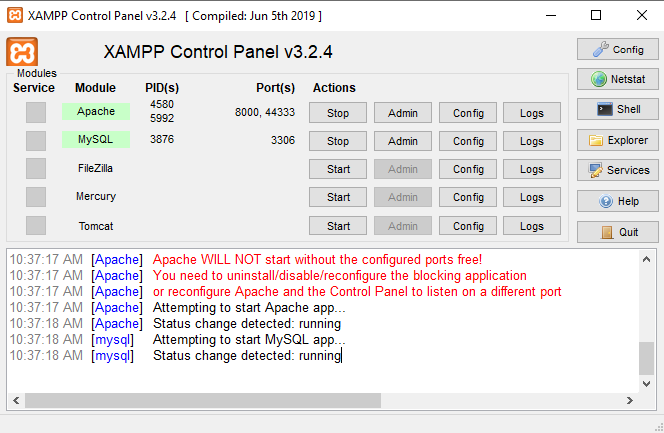
Sau khi cài đặt thành công và kết nối với phòng thí nghiệm, ta có thể tương tác với hệ thống và các thiết bị trong Lab này

Web Server:

XAMPP:

Giới thiệu:

XAMPP là 5 chữ cái đầu của các phần mềm được tích hợp trong chương trình. Đó là Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) Perl (P). Đây là chương trình giúp tạo web server được dùng trên nhiều hệ điều hành như Windows, Linux, MacOS, Cross-platform và Solaris. XAMPP có lợi thế hoàn toàn miễn phí và được sử dụng mã nguồn mở. Ngoài ra, cấu hình của web server tương đối gọn nhẹ, đơn giản nên chúng ngày càng được sử dụng phổ biến. Tiếp đến, XAMPP có giao diện quản lý tiện lợi, dễ dàng sử dụng, cho phép người dùng chủ động bật, tắt hay tái khởi động các dịch vụ máy chủ theo nhu cầu sử dụng.



Ứng dụng của XAMPP

* Phần mềm XAMPP được ứng dụng để xây dựng và phát triển các website theo ngôn ngữ PHP.
* Sử dụng để nghiên cứu, phát triển website thông qua Localhost của máy tính cá nhân.
* Được các lập trình viên sử dụng trong nhiều tình huống như học tập, thử nghiệm lập trình website, nghiên cứu nâng cấp website.

Ưu điểm và nhược điểm khi sử dụng XAMPP

Ưu điểm

* XAMPP chạy được trên nhiều hệ điều hành: Window, MacOS, Linux, Cross-platform, ...
* Măc dù XAMPP có cấu hình đơn giản và dễ sử dụng nhưng lại mang đến nhiều tính năng hữu ích như: lập [Server](https://hostingviet.vn/server-la-gi-tai-sao-phai-su-dung-server) giả định, lập Mail Server giả định và hỗ trợ SSL trên Localhost.
* XAMPP tích hợp nhiều thành phần quan trọng: PHP, Apache, MySql
* Mã nguồn mở: giao diện quản lý khá tiện lợi, cho phép chủ động bật, tắt hay khởi động lại bất kỳ dịch vụ máy chủ.

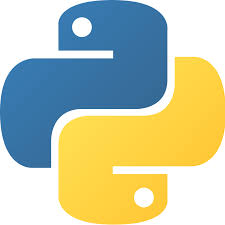
Vì thế, người dùng không cần cài đặt từng phần trên riêng lẻ mà chỉ cần cài XAMPP là có một web server hoàn chỉnh.

Nhược điểm

* Do cấu hình đơn giản nên XAMPP gây ra một số bất tiện như: không hỗ trợ cấu hình Module, không có Version MySQL.
  1. **Một số ngôn ngữ và thư viện sử dụng trong đề tài**

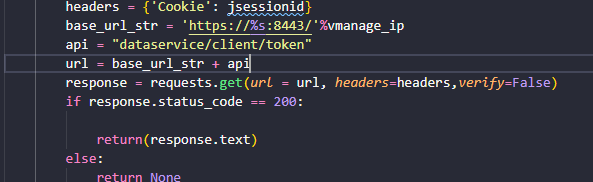
**Python:**

Python là một ngôn ngữ lập trình hướng đối tượng rất thông dụng dùng để viết các tiện ích hệ thống và các đoạn mã trên Internet. Nó cũng được sử dụng như ngôn ngữ kết dính đóng vai trò tích hợp C và C++. Được tạo ra bởi Guido Van Rossum tại Amsterdam năm 1990. Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động. Python được phát triển trong một dự án mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý. Phiên bản mới nhất của Python đến thời điểm nghiên cứu hiện tại là 3.8.4.



Python là ngôn ngữ có hình thức khá đơn giản và rõ ràng, do đó tạo nên sự dễ dàng tiếp cận cho những lập trình viên mới bắt đầu. Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, nó đã chuyển sang các hệ điều hành khác từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

Trong đề tài, em sử dụng Python và thư viện requests, json để thực hiện để gọi API. Thư viện **Requests** giúp em có thể thực hiện các tác vụ như gửi yêu cầu tới máy chủ cũng như xử lý phản hồi một cách đơn giản.



Trong đề tài sử dụng dữ liệu dạng JSON là định dạng dữ liệu phổ biến được sử dụng để truyền và nhận dữ liệu giữa ứng dụng web và web server nên ta cần thư viện JSON để xử lí dữ liệu.

PHP-Lavarel:

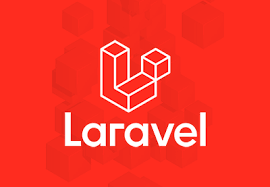
Hypertext Preprocessor, thường được viết tắt thành PHP là một ngôn ngữ lập trình kịch bản hay một loại mã lệnh chủ yếu được dùng để phát triển các ứng dụng viết cho máy chủ, mã nguồn mở, dùng cho mục đích tổng quát. PHP rất thích hợp với web và có thể dễ dàng nhúng vào trang HTML. PHP đã nhanh chóng trở thành một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) web phổ biến nhất thế giới vì các lí do như: tối ưu hóa cho các [ứng dụng web](https://vi.wikipedia.org/wiki/%E1%BB%A8ng_d%E1%BB%A5ng_web), tốc độ nhanh, nhỏ gọn, cú pháp giống [C](https://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)) và [Java](https://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), dễ học và thời gian xây dựng sản phẩm tương đối ngắn hơn so với các ngôn ngữ khác. PHP được tạo ra bởi [Rasmus Lerdorf](https://en.wikipedia.org/wiki/Rasmus_Lerdorf) và nó xuất hiện lần đầu tiên vào năm 1994. Sự phát triển của PHP có liên quan đến sự đóng góp rất lớn của [Zend](https://vi.wikipedia.org/w/index.php?title=Zend_Inc.&action=edit&redlink=1) Inc., công ty do các nhà phát triển cốt lõi của PHP lập nên nhằm tạo ra một môi trường chuyên nghiệp để đưa PHP phát triển ở quy mô [doanh nghiệp](https://vi.wikipedia.org/wiki/Doanh_nghi%E1%BB%87p). Ngoài ra công ty này còn góp phần xây dựng với cộng đồng tạo ra ngôn ngữ, các thư viện, tài liệu gốc của PHP.



Một trong web có thể được viết ra bằng nhiều [**các ngôn ngữ lập trình**](https://mindx.edu.vn/blog/post/ngon-ngu-lap-trinh-nao-de-hoc-nhat) khác nhau. Tuy nhiên, em đã sử dụng ngôn ngữ lập trình PHP để thiết kế web. Có rất nhiều lý do khiến em học và sử dụng PHP để lập trình web. Tiêu biểu như:

* Ngôn ngữ PHP có thể sử dụng trên mọi hệ điều hành, dễ dàng kết hợp với các cơ sở dữ liệu và được sử dụng rộng rãi.
* Tài liệu học PHP rất nhiều, cộng đồng PHP lớn nên khi gặp bất kỳ khó khăn nào cần giải đáp sẽ dễ dàng được hỗ trợ.
* Cơ sở dữ liệu của PHP rất lớn nên khi thiết kế website có thể sử dụng nhiều hệ cơ sở dữ liệu khác nhau.

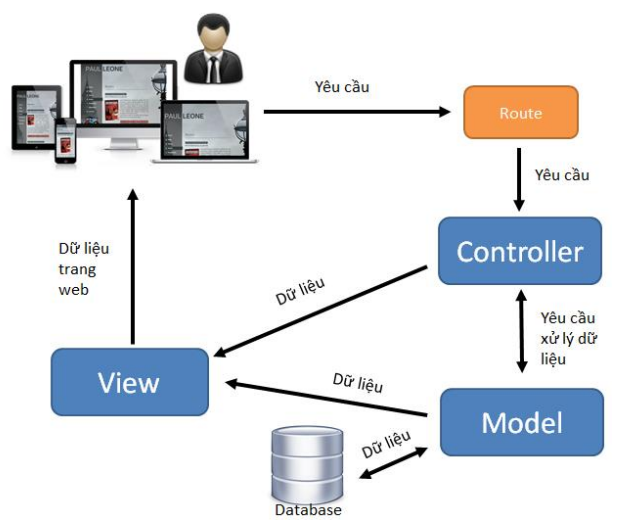
Laravel là một PHP Framework mã nguồn mở và miễn phí, được phát triển bởi Taylor Otwell và nhắm mục tiêu hỗ trợ phát triển các ứng dụng web theo cấu trúc model- view- controller (MVC). Những tính năng nổi bật của Laravel bao gồm cú pháp dễ hiểu và rõ ràng, một hệ thống đóng gói Modum và quản lý gói phụ thuộc, nhiều cách khác nhau để truy cập vào các cơ sở dữ liệu, nhiều tiện ích khác nhau hỗ trợ việc triển khai vào bảo trì ứng dụng.



MVC - viết tắt của Model View Controller. Đây là một mô hình, cho phép chúng ta tách biệt các thành phần xử lý trong hệ thống. Để từ đó chúng ta có thể giải quyết các công việc một các nhanh gọn và dễ dàng hơn. Model : sẽ đảm nhiệm các công việc trao đổi dữ liệu với database.

View : là thành phần giao diện, hiển thị dữ liệu cho người dùng.

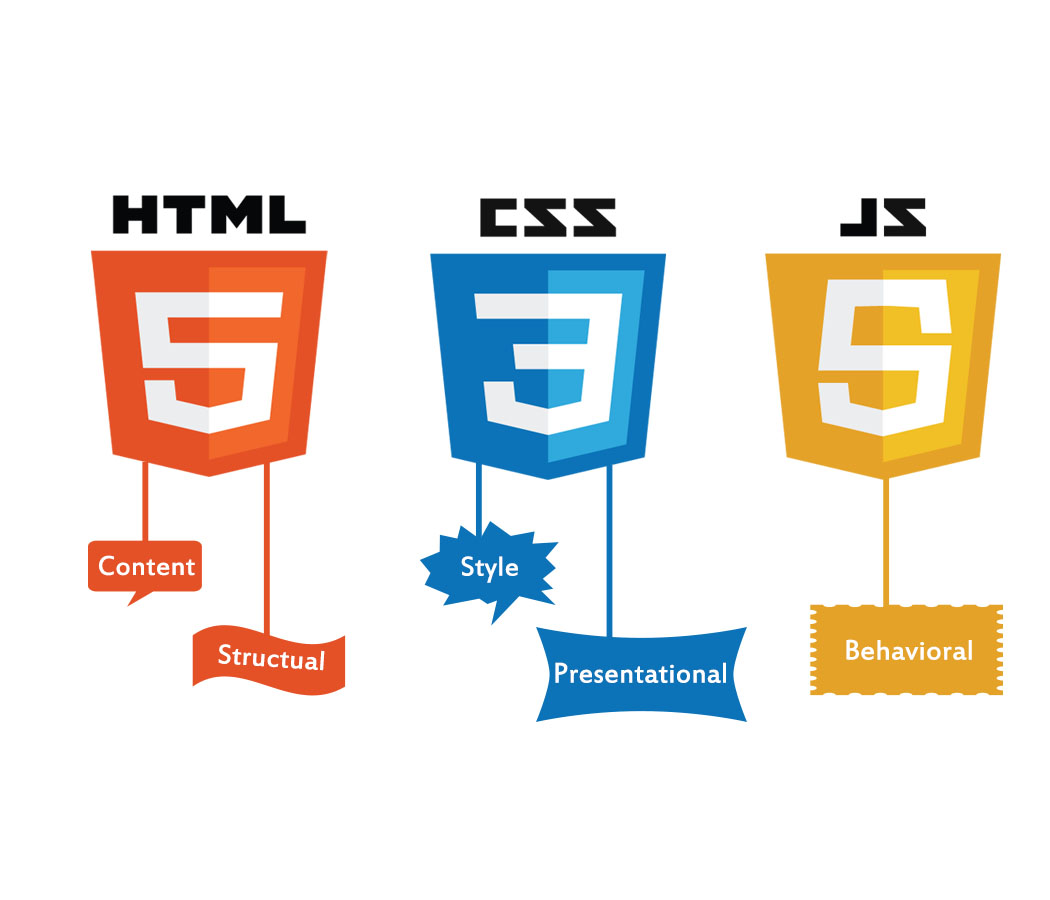
Controller : có chức năng điều khiển, sắp xếp và xử lý các yêu cầu của người dùng.



Khi người dùng gửi một yêu cầu lên hệ thống, hệ thống sẽ gửi về cho phía Controller qua Route. Route thực hiện chức năng định tuyến đến những controller cụ thể dựa trên yêu cầu từ người dùng và quản lí sơ đồ đường dẫn của trang web xử lý các yêu cầu của người dùng. Trong quá trình làm việc, Controller sẽ phải thông qua lớp Model để làm việc với CSDL. Sau khi xử lý xong công việc, Controller sẽ đưa sang Views để hiển thị cho người dùng.

HTML-CSS-JS-Bootstrap:

HTML - Hypertext Markup Language: Là ngôn ngữ đánh dấu văn bản. Nó như một khuôn mẫu quy chuẩn cho các thành phần sẽ hiển thị trên trang web. Coi một website là cơ thể sống thì HTML sẽ có vai trò như một khung xương định hình.



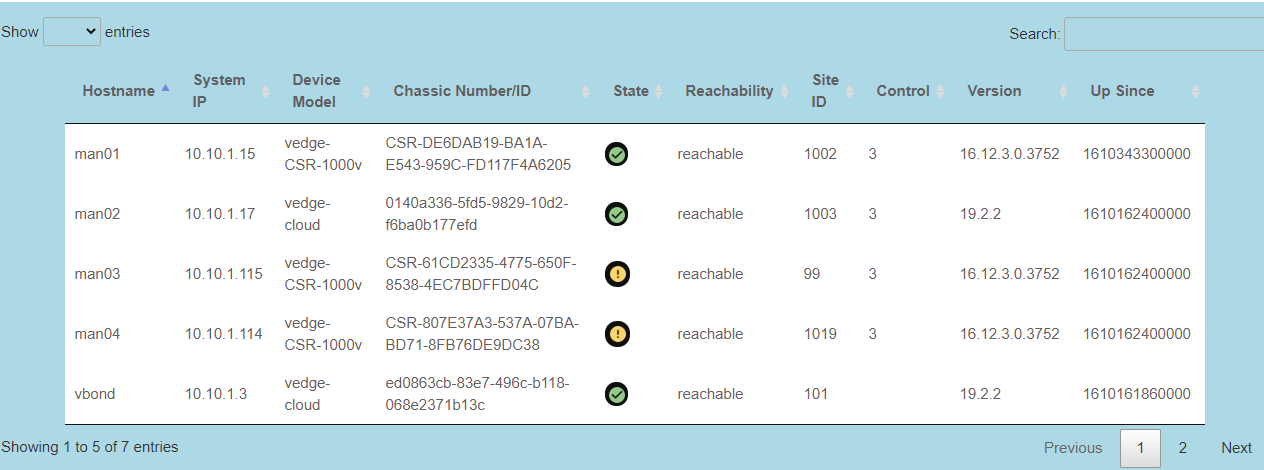
CSS - Cascading Style Sheets: **CSS** là ngôn ngữ tạo phong cách cho trang web. Nó dùng để tạo phong cách và định kiểu cho những yếu tố được viết dưới dạng ngôn ngữ đánh dấu, như là [HTML](https://www.hostinger.vn/huong-dan/html-la-gi/). Nó có thể điều khiển định dạng của nhiều trang web cùng lúc để tiết kiệm công sức cho người viết web. Nó phân biệt cách hiển thị của trang web với nội dung chính của trang bằng cách điều khiển bố cục, màu sắc, và font chữ.

Javascript: Javascript đảm nhận những hoạt động mà người dùng thao tác với trang web. Ví dụ khi người dùng ấn nút mua hàng trên một website bán hàng trực tuyến, javascript sẽ đảm nhiệm chức năng gửi thông tin đơn hàng về máy chủ và thông báo cho người dùng đã mua hàng thành công.

**Bootstrap:** là một bộ sưu tập miễn phí của các **mã nguồn mở** và công cụ dùng để tạo ra một mẫu webiste hoàn chỉnh. Với các thuộc tính về giao diện được quy định sẵn như kích thước, màu sắc, độ cao, độ rộng…, sử dụng Bootstrap giúp em tiết kiệm thời gian khi làm việc với **framework** này trong quá trình **thiết kế giao diện website**.

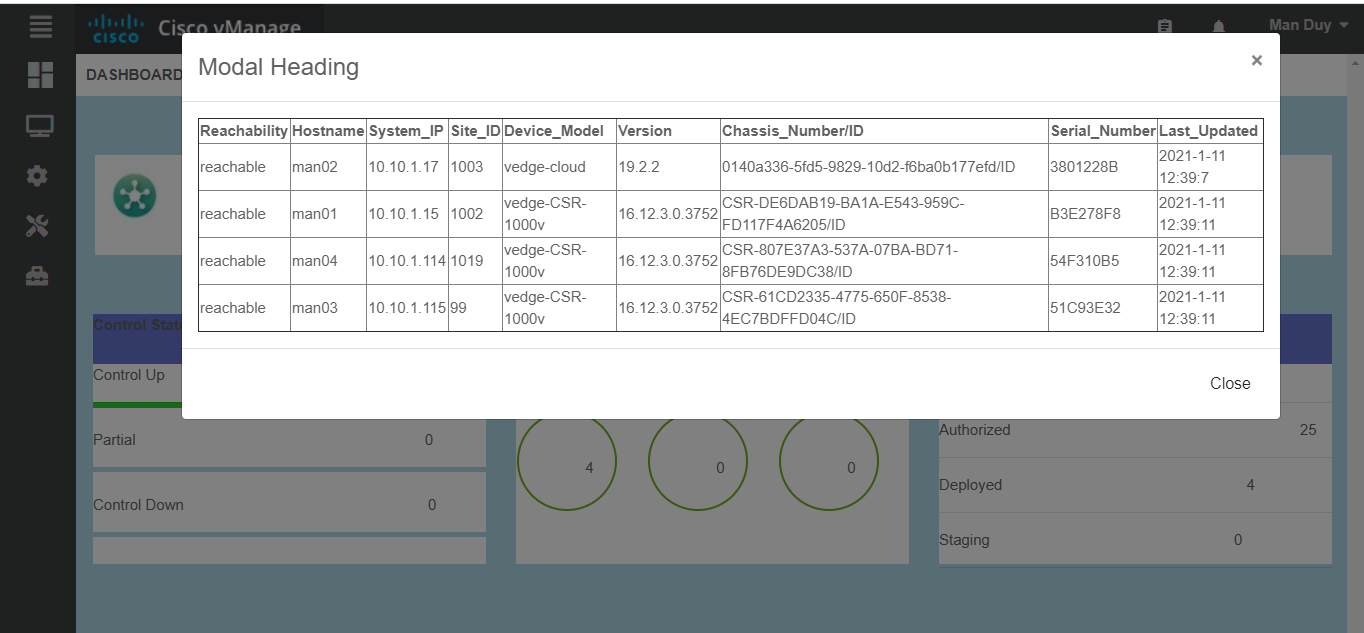


jQuery là thư viện được viết từ JavaScript, jQuery giúp xây dựng các chức năng bằng Javascript dễ dàng, nhanh và giàu tính năng hơn. Em sử dụng jQuery DataTables để hiển thị thông tin trong các bảng và thêm các tương tác vào chúng. Nó cung cấp tìm kiếm, sắp xếp và phân trang mà không cần bất kỳ cấu hình nào.



AJAX:

Là viết tắt của cụm từ **Asynchronous Javascript and XML**. AJAX không phải dùng một công nghệ duy nhất, cũng không phải ngôn ngữ lập trình, AJAX là một bộ kỹ thuật phát triển web. Là phương thức trao đổi dữ liệu với máy chủ và cập nhật một hay nhiều phần của trang web giúp chúng ta tạo ra sự sinh động cho Website của mình mà không tải lại trang. Ajax là một thuật ngữ chung cho các kỹ thuật Javascript khác nhau được sử dụng để kết nối với máy chủ web tự động mà không nhất thiết phải tải lại trang. Về mặt kỹ thuật, nó đề cập đến việc sử dụng các đối tượng [XmlHttpRequest](https://www.w3schools.com/xml/xml_http.asp) để tương tác với một máy chủ web động thông qua Javascript.



# **CHƯƠNG 3. THỰC HIỆN ĐỀ TÀI**

* 1. **Gọi API**

1. **Xác thực API**

 Cisco vManage hỗ trợ RESTful API, cung cấp các cuộc gọi để lấy thời gian thực và thông tin các thiết bị trong mạng và cho các mẫu cấu hình thiết bị tải lên và thông tin cấu hình liên quan khác.

Máy chủ kiểm soát truy cập API dựa trên các phiên. Tất cả người dùng sẽ có thể nhận được một phiên sau khi đăng nhập thành công. Mỗi phiên API sử dụng một mã thông báo duy nhất có giá trị trong suốt phiên. Nếu một yêu cầu API không bao gồm mã thông báo này, vManage sẽ từ chối yêu cầu, trừ khi điểm cuối được đưa vào danh sách được phép. Các bước để người dùng sử dụng API:

* Đăng nhập bằng tên người dùng và mật khẩu để thiết lập phiên :
* URL để gửi yêu cầu tới: Sử dụng https: //{vmanage-ip-address}/ j\_security\_check, thực hiện thao tác đăng nhập và kiểm tra bảo mật trên máy chủ web vManage tại địa chỉ IP được chỉ định.
* Phương thức yêu cầu: POST.
* Đầu vào lệnh gọi API: đầu vào là một ứng dụng, vì vậy đối với loại nội dung, sử dụng: ‘application/x-www-form-urlencoded’.
* Dữ liệu truyền vào được định dạng ở dạng JSON: {‘j\_username’=‘admin’,

‘j\_password’= ‘C1sco12345’}

Mã thông báo phiên nằm trong HTTP phản hồi. Ta có thể đăng nhập thông qua một yêu cầu API và tiếp tục sử dụng cùng một phiên cho đến khi hết hạn. Một phiên sẽ hết hạn sau 30 phút không hoạt động hoặc sau 24 giờ, là tổng thời gian hoạt động của một phiên. Trong thông báo phiên trả về, mã thông báo phiên có dạng:

‘JSESSIONID=9ua2RLnShMgQ59cKgsLWp-\_SGcRdJoasVPONXVhZ.81ac6722-a226-4411-9d5d-45c0ca7d567b’

* Lấy thông tin Token để được phép truy cập vào tài nguyên yêu cầu:

Một khi người dùng đã đăng nhập vào hệ thống thì những yêu cầu tiếp theo từ phía người dùng sẽ chứa thêm mã Token, cho phép người dùng quyền truy cập vào các đường dẫn, dịch vụ, và tài nguyên mà cần phải có sự cho phép nếu cần mã Token đó. Ngoài ra Token được dùng để ngăn chặn yêu cầu giả mạo trên nhiều trang web API. Nhận mã thông báo ngăn chặn giả mạo yêu cầu trên nhiều trang web, được yêu cầu cho hầu hết các hoạt động POST.

URL để gửi yêu cầu tới: https://{vmanage-ip-address}/dataservice/client/token

Phương thức: GET

Đầu vào lệnh gọi API: {'Cookie': jsessionid}

Với jsessionid là phiên cookie được yêu cầu xác thực trước đó  
Mã thông báo XSRF nằm trong nội dung phản hồi. Sử dụng mã thông báo X-XSRF-TOKEN cùng với cookie cho các yêu cầu API tiếp theo.

X-XSRF-TOKEN nhận được có dạng:

‘1C981A10B2338E5899F865A8C81D55B7CE9C0EE5938C88692A7CF26105263676EB52E35E7491E2ADEC29392013E70D740675’

1. **Sử dụng API**

Ta có thể sử dụng API để thực hiện những việc sau :

* GET — Truy xuất hoặc đọc thông tin.
* PUT — Cập nhật một đối tượng.
* POST — Tạo một đối tượng.
* DELETE — Xóa một đối tượng.

Bảng dưới đây mô tả các trường cần thiết cho tất cả các phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | GET | POST | PUT/PATCH | DELETE |
| **URL** | Bắt buộc | Bắt buộc | Bắt buộc | Bắt buộc |
| **Type** | Bắt buộc | Bắt buộc | Bắt buộc | Bắt buộc |
| **Cookie** | Bắt buộc | Bắt buộc | Bắt buộc | Bắt buộc |
| **Token** | Không bắt buộc | Bắt buộc | Bắt buộc | Bắt buộc |
| **Payload** |  | Bắt buộc | Bắt buộc | Bắt buộc |

Nguyên tắc gọi API :

Tất cả lệnh gọi đến https:// {vmanage-ip-address}/dataservice đều phải dùng "application/json" làm tiêu đề Content-Type.

Đối với yêu cầu GET, ta cần thêm cookie vào và có thể bắt buộc có mã thông báo token với một số yêu cầu nhất định.

Đối với các yêu cầu POST, PUT, DELETE thì ta cần truyền thêm mã thông báo token vào header.

Ví dụ: headers = {'Cookie': jsessionid, 'X-XSRF-TOKEN':token, 'Content-Type': 'application/json'}

Khi sử dụng phương thức POST, cần phải có nội dung để xác định đối tượng và nội dung tùy theo yêu cầu của API và nội dung truyền vào phải có dạng JSON.

Ví dụ:

Payload = {

"query": {

"condition": "AND",

"rules": [

{

"value": [

"24"

],

"field": "entry\_time",

"type": "date",

"operator": "last\_n\_hours"

},

{

"field": "system\_ip",

"operator": "in",

"type": "string",

"value": [

"10.10.1.113"

]

}

]

},

"aggregation": {

"field": [

{

"property": "severity\_level",

"order": "asc",

"sequence": 1

}

],

"histogram": {

"property": "entry\_time",

"interval": 1,

"type": "hour",

"order": "asc"

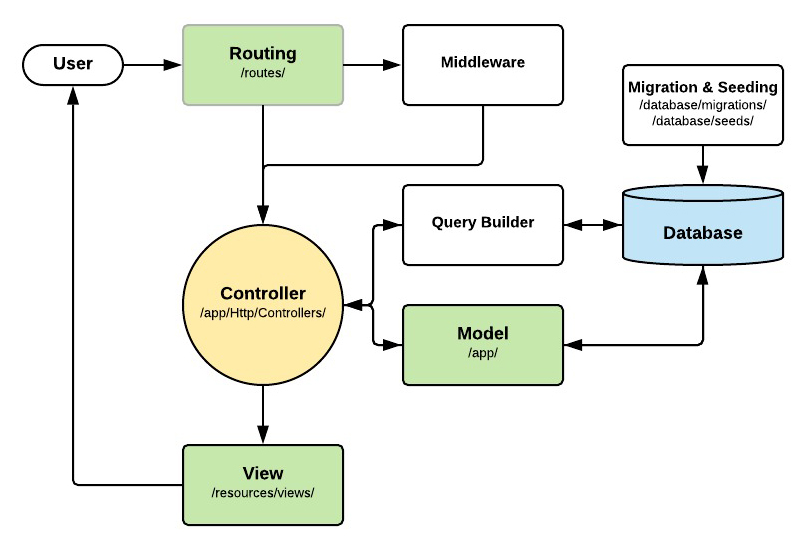
}

}

}

* 1. **Xây dựng Website**
     1. **Quy trình thực hiện**

Trước khi xây dựng trang web bằng Lavarel ta cần phải hiểu cách làm việc của Lavarel. Hình dưới đây sẽ cho chúng ta hiểu khái quát cách hoạt động của Laravel, vì đây là cấu trúc cơ bản nên tất nhiên chưa đầy đủ.

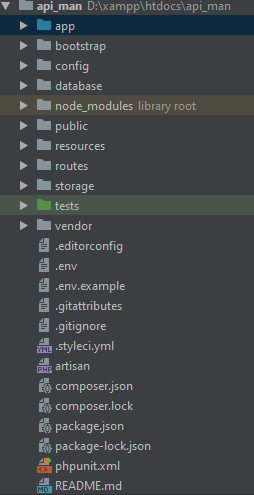


|  |  |
| --- | --- |
| **Folder / File** | **Mô tả** |
| User | User gửi yêu cầu. |
| Routing | Yêu cầu từ User sẽ được Routing điều hướng:   * Tới Controller để xử lý yêu cầu. * Tới Middleware để kiểm tra điều hướng cần thiết. * Tới thẳng View nếu không cần xử lý. |
| Middleware | Middleware có thể xem như bộ lọc HTTP từ request, VD như dùng để kiểm tra, xác thực người dùng đăng nhập vào hệ thống. Sau khi kiểm tra, xác thực xong sẽ trả về Controller để tiếp tục xử lý. |
| Controller | Controller được xem như trung tâm điều khiển của hệ thống, tất cả thao tác xử lý nên được thực hiện ở đây.   * Controller kết nối với Database thông qua điều khiển từ Model. Hoặc có thể thao tác trực tiếp tới Database thông qua các Query Builder * Kết quả xử lý sẽ được trả về view. |
| Model | Khi có yêu cầu từ Controller, Model sẽ tương tác với Database và trả kết quả về Controller, một số trường hợp cần thiết thì Model cũng có thể trả thẳng kết quả về View. |
| Database | Lưu trữ và chứa dữ liệu. |
| Migration & Seeding | * Migration, được dùng để tạo Database. * Seeding, được dùng để tạo dữ liệu ảo cho Database. |
| Query Builder | Câu truy vấn Database. |
| View | View nhận dữ liệu xử lý từ Controller (hoặc Model, Routing), hiển thị kết quả cho người dùng |

Để tạo một dự án mới bằng Lavarel ta cần một phần mềm tạo máy chủ web, trong đề tài này em sử dụng XAMPP để làm máy chủ web ảo. Mặc định của Xampp để chạy được chương trình PHP là cần đặt trong thư mục D:\xampp\htdocs, do đó cần chuyển tới thư mục htdocs trong XAMPP để tạo một dự án mới. gõ câu lệnh sau để tạo dự án:

composer create-project laravel/laravel api\_man

Sau khi gõ câu lệnh xong thì ta có cấu trúc một dự án Lavarel như hình dưới :

****

**Trong đó :**

|  |  |
| --- | --- |
| **Thư mục** | **Ý nghĩa** |

|  |  |
| --- | --- |
| App | Chứa toàn bộ code của các ứng dụng như model, controller,… |
| Bootstrap | Chứa 1 số file lúc khởi động framework và cấu hình autoloading. |
| Config | Chứa file cấu hình của ứng dụng |
| Database | Chứa 2 thư mục migration (tạo và thao tác database) và seeds (tạo dữ liệu mẫu), tiện lợi để lưu trữ dữ liệu sau này. |
| Public | Chứa các file đối tượng public như asset (ảnh, CSS, javascript, …) |
| Resources | Chứa những file view và raw, các file biên soạn như LESS, SASS, hoặc JavaScript. |
| Routes | Chứa tất cả các điều khiển route (đường dẫn) trong dự án. Chứa các file route sẵn có: web.php, channels.php, api.php, và console.php. |
| Storage | Chứa các file biên soạn blade templates, file based sessions, file caches, và những file sinh ra từ dự án.   * Thư mục app, dùng để chứa những file sinh ra từ dự án. * Thư mục framework, chứa những file sinh ra từ framework và caches. * Thư mục logs, chứa những file logs. * Thư mục /storage/app/public, lưu những file người dùng tạo ra như hình ảnh. |
| Tests | Chứa các file test |
| Vendor | Nơi chứa các thư viện của Composer |

* + 1. **Tạo Route**

Laravel không cho phép chạy file PHP bất kỳ nào một cách tùy tiện bằng việc gõ tên file lên thanh địa chỉ trình duyệt, mà tất cả phải được điều khiển thông qua Routing. Routing chứa sơ đồ đường dẫn của cả ứng dụng, có tác dụng giống như một tấm bản đồ chỉ dẫn yêu cầu đến đúng đích.

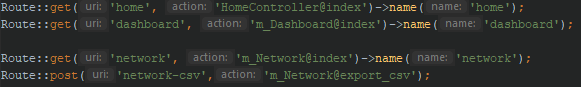
Routing được chứa trong thư mục /routes/

* /routes/api.php: điều khiển các route của ứng dụng, như route của ứng dụng User (đăng ký, đăng nhập, ...).
* /routes/web.php: điều khiển các route của view

Cấu trúc của một Routing

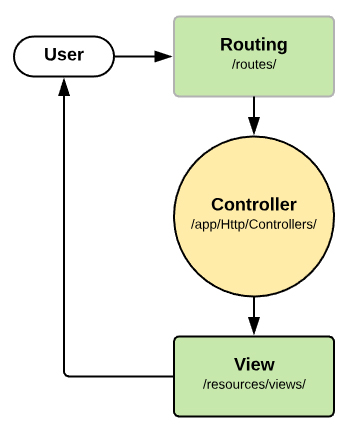
Route::METHOD('URL', ACTION);

Ví dụ:



* + 1. **Tạo Controller**
* Controller chứa các xử lý logic (tính toán) giúp xử lí các yêu cầu từ Route.
* Controller sinh ra là để trở thành trung tâm xử lý logic.
* Controller được đặt bên trong thư mục /app/Http/Controllers/.

Để hiểu một cách cơ bản nhất cách hoạt động của một Controller ta xem cấu trúc đơn giản sau:



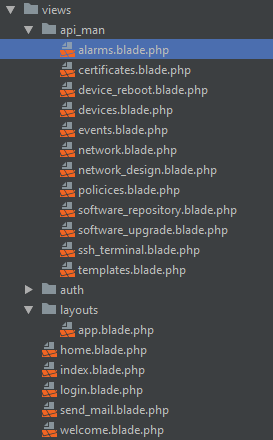
|  |  |
| --- | --- |
| **Folder / File** | **Mô tả** |
| User | User gửi yêu cầu. |
| Routing | Yêu cầu gửi từ User sẽ được Routing điều hướng:   * Tới Controller để xử lý yêu cầu. * Tới thẳng View nếu không cần xử lý. |
| Controller | Controller được xem như trung tâm điều khiển của hệ thống, tất cả thao tác xử lý nên được thực hiện ở đây.   * Kết quả xử lý sẽ được trả về view. |
| View | View nhận dữ liệu xử lý từ Controller (hoặc Routing), hiển thị kết quả cho người dùng. |

* + 1. **Tạo Views**

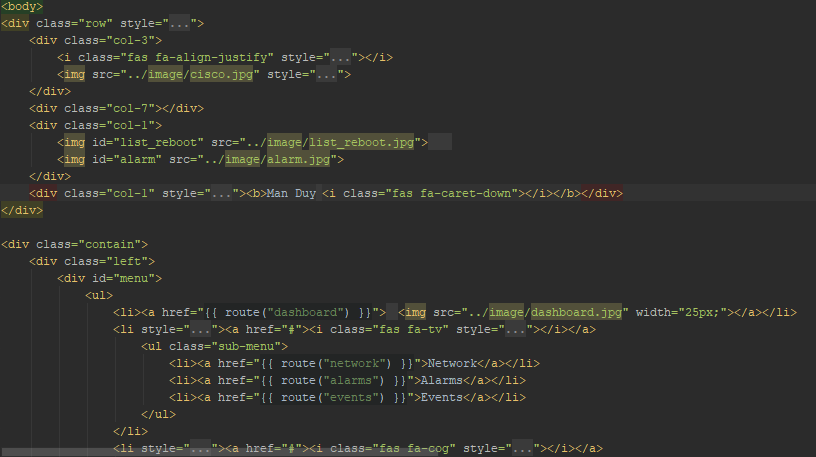
Views là nơi giúp hiển thị những gì chúng ta thấy trên trình duyệt, phần lớn là cấu trúc HTML.Trong view thì ta có thể sử dụng tất cả các ngôn ngữ trong file PHP hỗ trợ như: HTML, CSS, JS, …

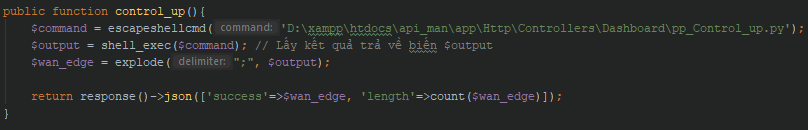
Views chứa trong thư mục /resources/views/

Views sử dụng cấu trúc Blade Templates, nên tên file của views sẽ là: name.blade.php, trong đó name là tên views bất kỳ.

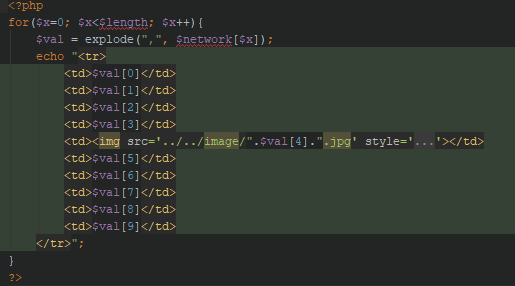
****

Để hiển thị Views ra màn hình, ta cần chỉ đường đi đến nó. Views có nhiệm vụ nhận dữ liệu từ bộ xử lí Controller (hoặc Route) để hiển thị kết quả ra màn hình.

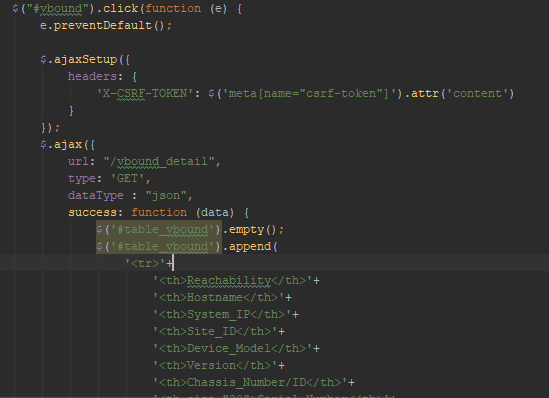
****

****

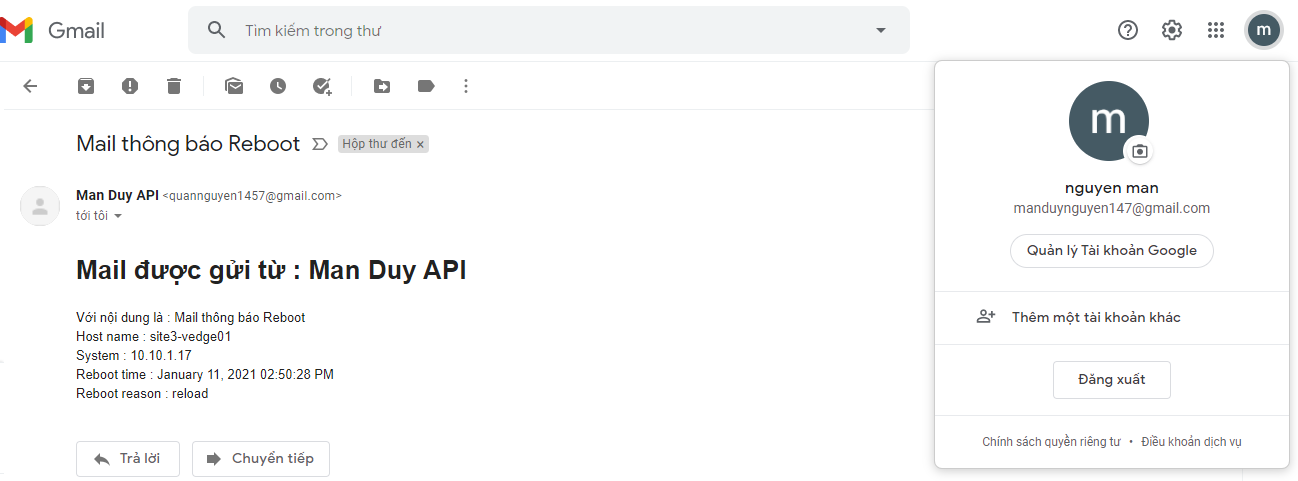
**Sử lí data**

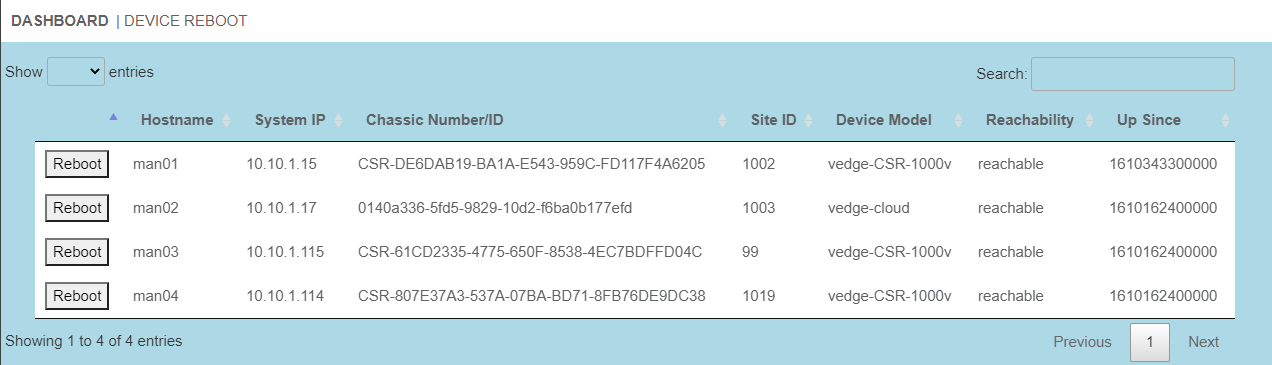
****

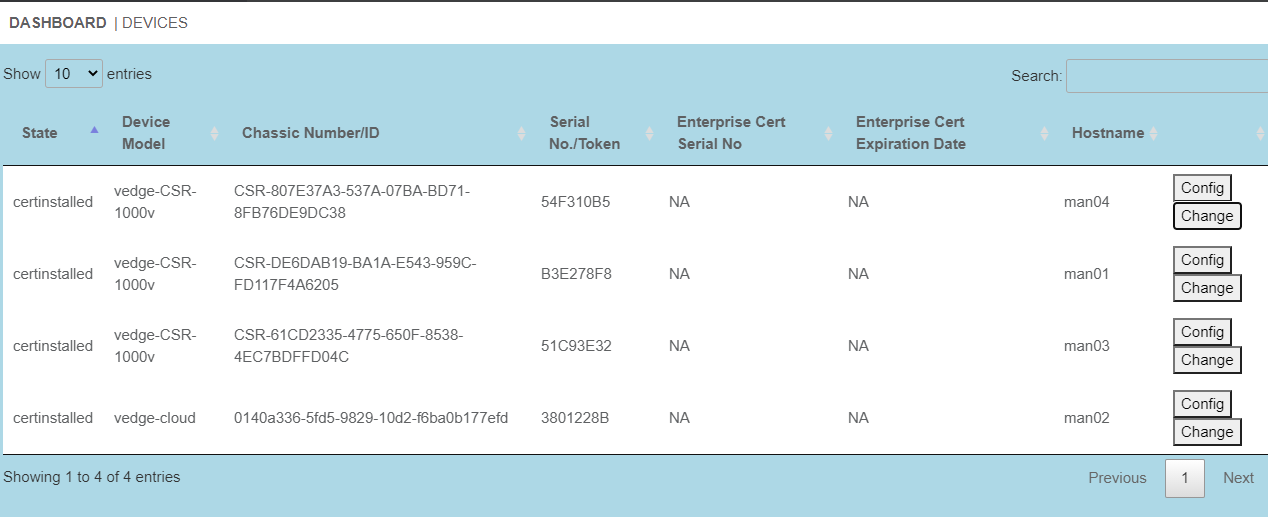
**Sử lí data=ajax**

****

**Dùng ajax**

****

****

****

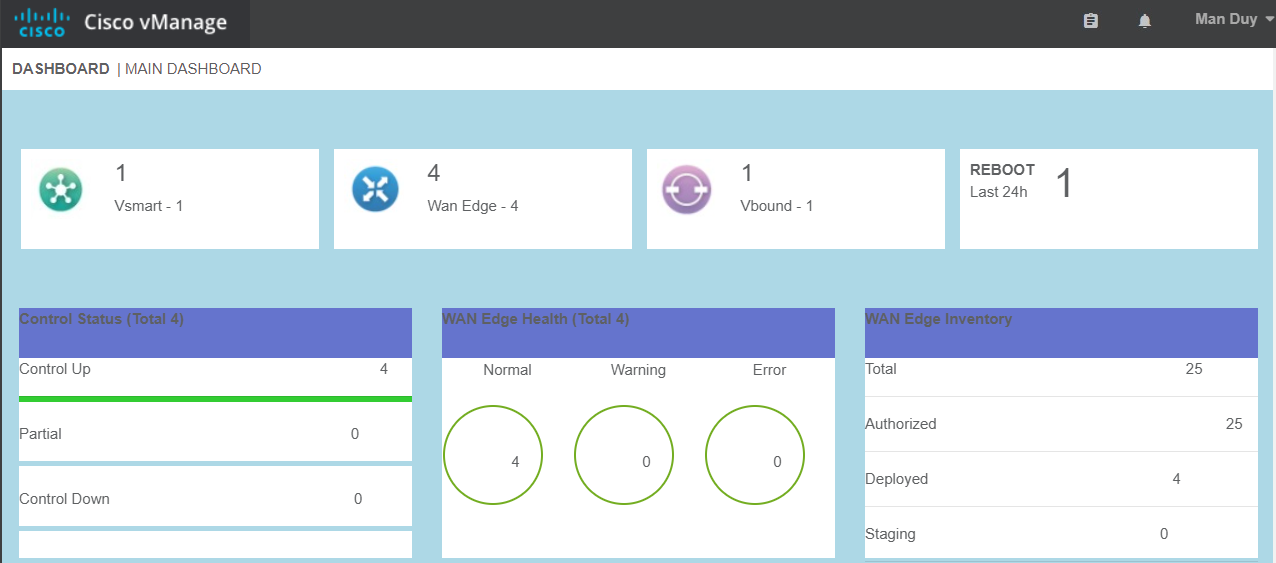
* 1. **Kết quả và định hướng phát triển**

1. **Kết quả**

**Giao diện đăng nhập :**



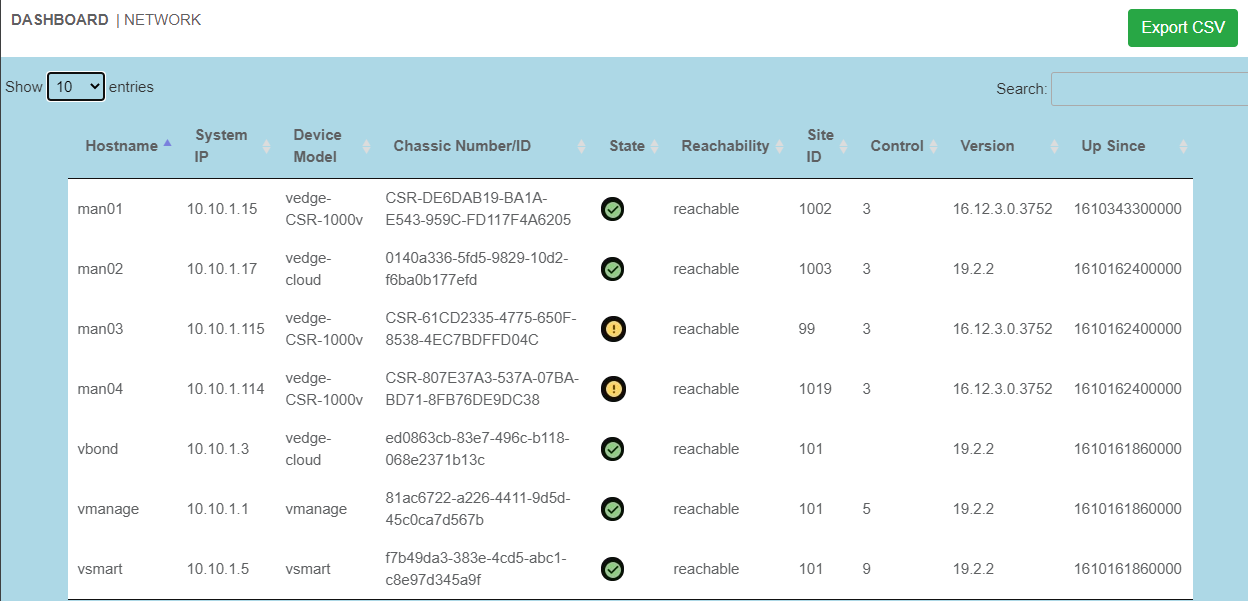
**Giao diện màn hình chính để quản lí tổng thể hệ thống :**

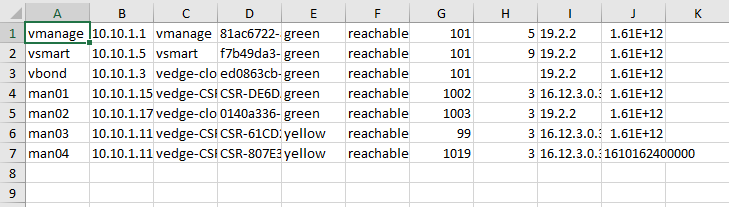
****

**Trong màn hình chính hiện thị các thông tin như :**

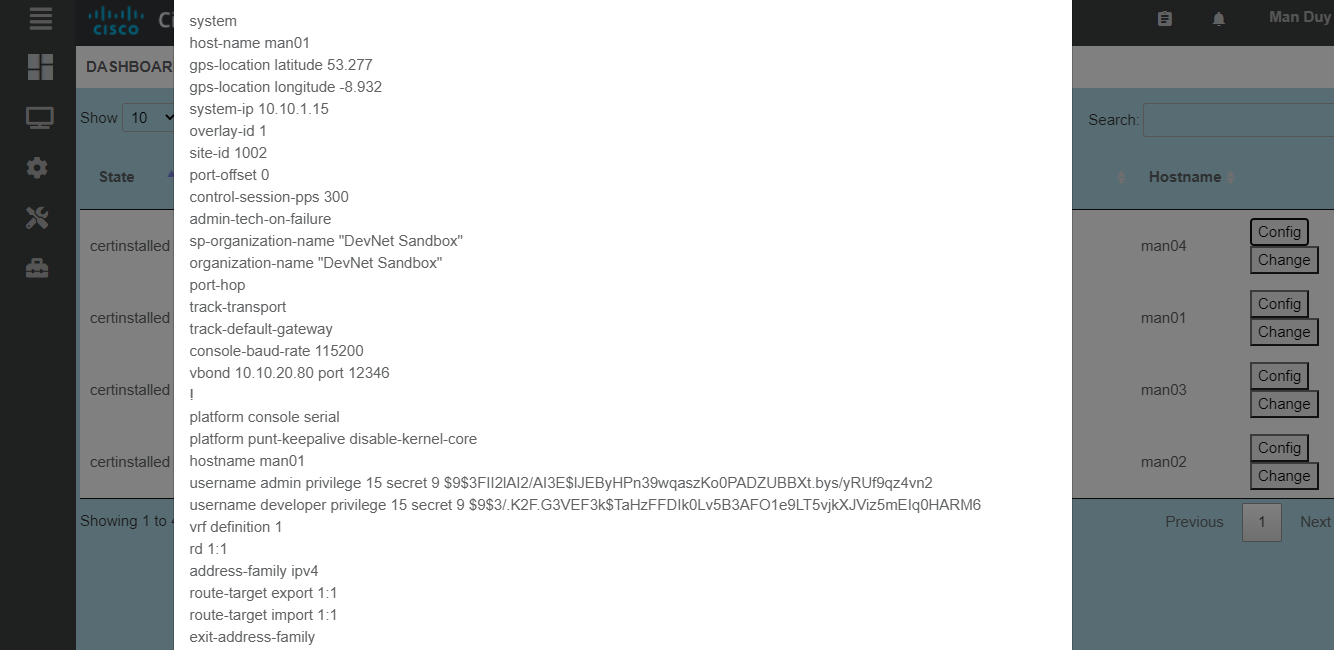
* **Số lượng thiết bị hoạt động**
* **Sức khỏe của các thiết bị**
* **Số lần khởi động lại của thiết bị trong 24h.**
* **Số lượng thiết bị trong kho lưu trữ**

**Xem thông tin thiết bị và xuất ra file CSV :**

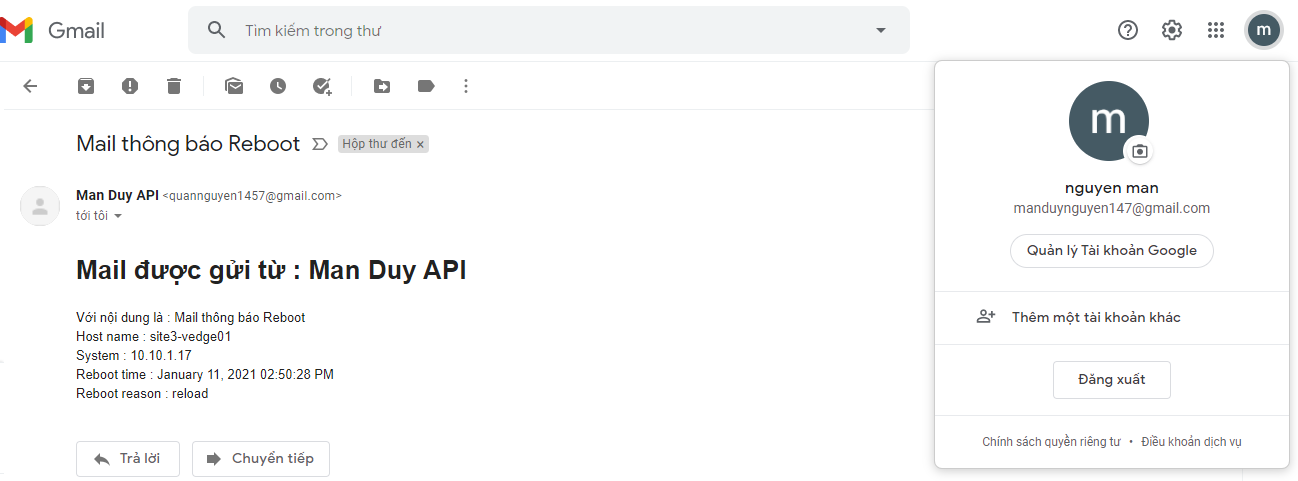
****

****

**Xem cấu hình thiết bị :**

****

**Gửi mail khi thiết bị khởi động lại :**

****

1. **Hướng phát triển :**

**Trang web sẽ sử dụng nhiều API hơn mà được Cisco cung cấp.**

**Sử dụng API để cấu hình sẽ được được đưa vào nhiều hơn so với hiện tại.**

Code:

https://github.com/victornguyen98/Lu-n-v-n.git

Tài liệu tham khảo

<https://sdwan-docs.cisco.com/Product_Documentation/Command_Reference/Command_Reference/vManage_REST_APIs>

<https://developer.cisco.com/docs/sdwan/>

<https://vi.wikipedia.org/wiki/Wikipedia>

<https://www.w3schools.com/>