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



**LUẬN VĂN TỐT NGHIỆP**

**……(TÊN LUẬN VĂN TỐT NGHIỆP)……**

Ngành: **ĐIỆN-ĐIỆN TỬ VIỄN THÔNG**

Chuyên ngành: **ĐIỆN TỬ VIỄN THÔNG**

Giảng viên hướng dẫn:

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!

**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

**Ý nghĩa đề tài**

Hệ thống điểm danh đóng một vài trò quan trọng nhằm góp phần hạn chế những việc không hay như trốn học, bỏ học giữa tiết trong môi trường nhà trường. Hệ thống điểm danh dựa trên nhận diện khuôn mặt bằng camera còn góp một phần vào việc học của sinh viên, học sinh, giúp cho họ biết tích cực tham gia đến lớp học để trau dồi kiến thức mỗi ngày.

**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

1. **Mục đích nghiên cứu.**

Tìm hiểu và nghiên cứu kiến thức về lập trình hướng đối tượng. Tìm hiểu và học hỏi các kiến thức cơ bản của xử lý ảnh và nhận dạng đối tượng.

1. **Nhiệm vụ nghiên cứu.**

Xử lý hình ảnh đầu vào và trích xuất những đặc trưng để nhận diện khuôn mặt.

Xây dựng hệ thống điểm danh tự động và lưu trữ dữ liệu trên SQL Server.

1. **Phương pháp nghiên cứu.**

Trao đổi với các thành viên trên cộng đồng mạng để trau dồi kiến thúc về lập trình hướng đối tượng với ngôn ngữ lập trình Python.

Học hỏi cách thức sử dụng và quản lý SQL cơ bản.

1. **Kết quả đạt được.**

Trích xuất các đặc trưng của một khuôn mặt và lưu vào một thư mục để nhận diện.

Xây dựng được giao diện điểm danh tự động.

Lưu và trả kết quả một cách nhanh chóng.

MỤC LỤC

MỤC LỤC HÌNH

DANH MỤC TỪ VIẾT TẮT

API

HTTP

HTTPS

REST

RESTFUL

CLI

SNMP

IoT

SSH

# **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 Apps để phục vụ nhu cầu kinh doanh thay đổi quá nhanh hàng ngày. Các Apps 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 quản lí và cấu hình thiết bị mạng bằng API ” 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 VỀ API**

* 1. **Tổng quan về API**

1. **Khái niệm API**

Để giao tiếp và định cấu hình mạng, các nhà phát triển phần mềm thường sử dụng ứng dụng các giao diện lập trình (API). 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. API chỉ là các hàm hay thủ tục thông thường. Các hàm này được viết trên nhiều ngôn ngữ lập trình khác nhau. 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 server cùng với thông tin được yêu cầu.

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. Đa số Web API được thiết kế theo tiêu chuẩn [RESTful](https://thietbitudong.com.vn/restful-api-la-gi/).
* **API trên hệ điều hành**: Windows hay Linux có rất nhiều API. Họ cung cấp các tài liệu API là đặc tả các hàm, phương thức cũng như các giao thức kết nối. Nó giúp [lập trình viên](https://itviec.com/viec-lam-it/developer?utm_source=blogpost&utm_medium=center_post&utm_campaign=linktosite) có thể tạo ra các phần mềm ứng dụng có thể tương tác trực tiếp với hệ điều hành.
* **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.

**Cách Web API hoạt động**

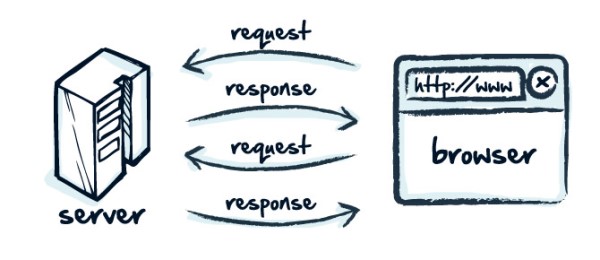
* Đầu tiên là xây dựng URL API để bên thứ ba có thể gửi request dữ liệu đến máy chủ cung cấp nội dung, dịch vụ thông qua giao thức HTTP hoặc HTTPS.
* Tại web server cung cấp nội dung, các ứng dụng nguồn sẽ thực hiện kiểm tra xác thực nếu có và tìm đến tài nguyên thích hợp để tạo nội dung trả về kết quả.
* Server trả về kết quả theo định dạng JSON hoặc XML 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 parse để lấy data. Sau khi có được data thì thực hiện tiếp các hoạt động như lưu dữ liệu xuống Cơ sở dữ liệu, hiển thị dữ liệu…

1. **REST API**

REST (**RE**presentational **S**tate **T**ransfer) được đưa ra vào năm 2000, trong luận văn tiến sĩ của Roy Thomas Fielding (đồng sáng lập giao thức HTTP). Nó là một dạng chuyển đổi cấu trúc dữ liệu, là một phong cách kiến ​​trúc cho việc thiết kế các ứng dụng có kết nối. Nó sử dụng HTTP đơn giản 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/). RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các tài nguyên. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP. Chức năng quan trọng nhất của **REST**là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. 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 là chữ viết tắt của 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).

Http hoạt động dựa trên **mô hình Client (máy khách) – Server (máy chủ)**. 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.



**HTTPS là phiên bản an toàn của http**(viết tắt của Hyper Text Transfer Protocol Secure - giao thức truyền tải siêu văn bản bảo mật), giao thức mà qua đó dữ liệu được gửi giữa trình duyệt và trang web bạn đang kết nối.

Uniform Resource Locator (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:

**protocol://hostname:port/path-and-file-name**

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 - 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 ... Client có thể sử dụng một trong các phương thức đó để gửi request lên server.
* 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 client, đến server. Một số trường: Accept-Charset, Accept-Encoding, Accept-Language, Authorization, Expect, From, Host, …

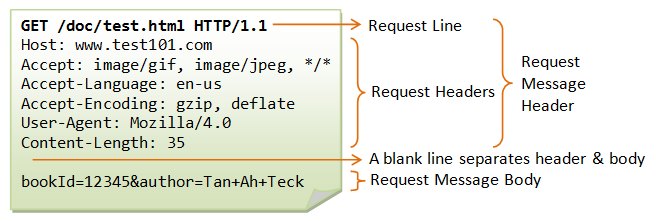
* Tùy chọn một thông điệp

Khi request đến server, server thực hiện một trong 3 hành động sau:

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.



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 server cho client.

* Tiêu chuẩn Status code 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 request thành công mà không gặp mất 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)

Server phải thực hiện hành động bổ sung để hoàn thành request.

* 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 browser hay khách viếng thăm website).

* 400 - Bad Request: Request không hợp lệ
* 401 – Unauthorized: Request 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 Resource này.
* 422 - Unprocessable Entity: Dữ liệu không được xác thực
* 429 - Too Many Requests: Request bị từ chối do bị giới hạn
* Status code 5xx (lỗi server)

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

* 500 – Internal Server Error: Hư
* 501 – Not implement: Chưa code
* 502 – Bad Gateway: đa số là do lỗi server (Nginx, Apache, Lite Speed, …)
* 503 – Service Unavailable: Server tạm thời không xử lý được/ kịp

1. **Ưu, nhược điểm của API khi cấu hình:**

**A**

* 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 giao 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.

Ngonp ngữ Back-end

* 1. **Tài nguyên**

**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/>

Có 2 loại hộp cát:

* Always-On Sandboxes
* Reservation Sandboxes



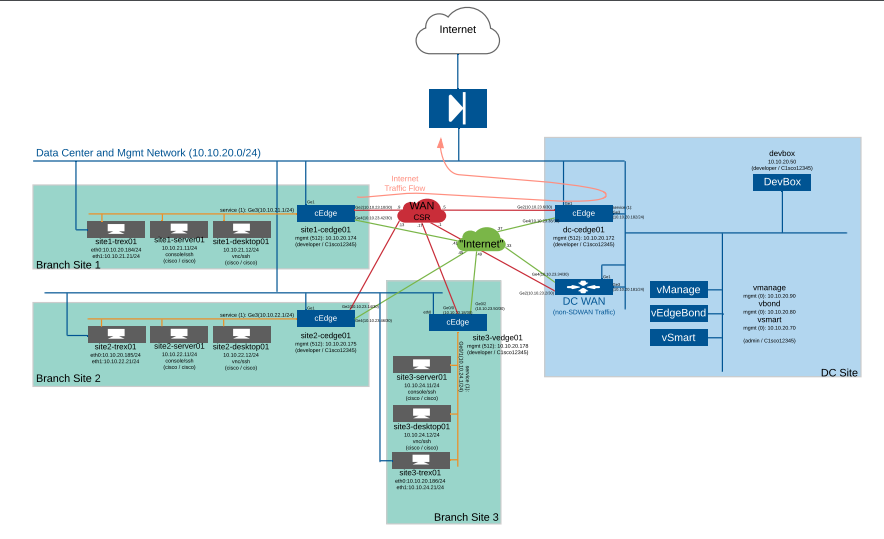
Always-On Sandboxes (hộp cát luôn bật): luôn sẵn sàng hoạt động khi ta cần. Nhưng môi trường này được **chia sẻ** giữa tất cả người dùng nên quyền truy cập quản trị bị hạn chế. Loại hộp cát này bị giới hạn về tài nguyên chia sẻ nên ít được sử dụng.

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.

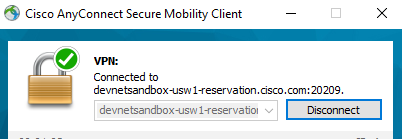
Trong đề tài này em chọn hộp cát đặt trước: Cisco SD-WAN Sandbox để phục vụ cho đề tài này. Cisco SD-WAN Sandbox bao gồm một môi trường SD-WAN ảo hoàn chỉnh và tất cả các thành phần của nó mà các nhà phát triển cam kết sử dụng để phát triển, gỡ lỗi và kiểm tra các mẫu SD-WAN ứng dụng của Cisco. Trong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

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

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.



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

* 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.

PHP:

PHP 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. Nó rất thích hợp với web và có thể dễ dàng nhúng vào trang HTML. Do được 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 nên 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. PHP được tạo ra bởi [Rasmus Lerdorf](https://en.wikipedia.org/wiki/Rasmus_Lerdorf) và nó xuất hiện lần đầu tiên vào năm 1995.



Ngôn ngữ, các thư viện, tài liệu gốc của PHP được xây dựng bởi cộng đồng và có sự đóng góp rất lớn của [Zend Inc.](https://vi.wikipedia.org/w/index.php?title=Zend_Inc.&action=edit&redlink=1), 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).

HTML - **H**ypertext **M**arkup **L**anguage: 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 - **C**ascading **S**tyle **S**heets: Là ngôn ngữ dùng để đặc tả các thành phần html. Html là khung xương thì css chính là những da thịt bao quanh bên ngoài. Css có tốt thì trang web mới đẹp.

Javascript: Là một cơ thể sống, điều còn thiếu chính là những hoạt động. Vì thế javascript sẽ đảm nhận những hoạt động mà người dùng thao tác với trang web. Ví dụ khi người dùng ấn nút mua hàng, javascript sẽ đảm nhiệm chức năng gửi thông tin đơn hàng về server và thông báo cho người dùng đã mua hàng thành công.

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- rõ ràng, một hệ thống đóng gói Modular và quản lý gói phụ thuộc, nhiều cách khác nhau để truy cập vào các cơ sở dữ liệu quan hệ, nhiều tiện ích khác nhau hỗ trợ việc triển khai vào bảo trì ứng dụng.

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.

AJAX Là viết tắt của cụm từ **Asynchronous Javascript and XML**. 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 reload 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.

**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…, các **nhà thiết kế website** có thể sáng tạo nhiều sản phẩm mới mẻ nhưng vẫn tiết kiệm thời gian khi làm việc với **framework** này trong quá trình **thiết kế giao diện website**.

# **CHƯƠNG 3. THỰC HIỆN ĐỀ TÀI**

* 1. **Xây dựng API**

1. **Xác thực API**

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 :

POST https://{vmanage-ip-address}/j\_security\_check

Content-Type: application/x-www-form-urlencoded

HTTP Body: "j\_username={admin}&j\_password={C1sco12345}"

Trong đó : tên người dùng và mật khẩu được mã hóa dưới dạng j\_username và j\_password. 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.

**Cookie:**

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 request 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 có 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:

GET https://{vmanage-ip-address}/dataservice/client/token

headers = {'Cookie': jsessionid}

Với header 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:**

1C981A10B2338E5899F865A8C81D55B7CE9C0EE5938C88692A7CF26105263676EB52E35E7491E2ADEC29392013E70D740675

1. 1. **Xây dựng Website**
   2. **Giới thiệu giao diện Website**