BAN CƠ YẾU CHÍNH PHỦ

# HỌC VIỆN KỸ THUẬT MẬT MÃ

****

BÁO CÁO MÔN HỌC

# KỸ THUẬT LẬP TRÌNH

**ĐỀ TÀI: NGHIÊN CỨU VÀ XÂY DỰNG HTTP PROGRAMMING**

Nhóm sinh viên thực hiện:

# Bùi Trung Kiên - AT180228

# Nguyễn Đức Anh - AT180602

# Nguyễn Thái Dương - AT180612

# Nguyễn Tuấn Minh - AT180633

Lớp: **A18-01**

***Hà Nội - 2024***

MỤC LỤC

[DANH MỤC HÌNH ẢNH 1](#_Toc162376678)

[DANH MỤC TỪ VIẾT TẮT 1](#_Toc162376679)

[LỜI NÓI ĐẦU 1](#_Toc162376736)

[CHƯƠNG 1: TỔNG QUAN VỀ GIAO THỨC HTTP 2](#_Toc162376737)

[1.1 Giới Thiệu Về Giao Thức HTTP 2](#_Toc162376738)

[1.1.1 Khái Niệm Cơ Bản Về Giao Thức HTTP 2](#_Toc162376739)

[1.1.2 Các Phiên Bản Của Giao Thức HTTP 2](#_Toc162376740)

[1.1.3 Các Đặc Điểm Của Giao Thức HTTP 3](#_Toc162376741)

[1.2 Lịch Sự Và Phát Triển Về Giao Thức HTTP 4](#_Toc162376742)

[1.2.1 Xuất Hiện Và Phát Triển Của HTTP 4](#_Toc162376743)

[1.2.2 Những Cải Tiến Của Giao Thức HTTP Qua Các Phiên Bản 5](#_Toc162376744)

[1.2.3 Tầm Quan Trọng Của Giao Thức HTTP Đối Với Việc Truyền Tải Dữ Liệu Trên Mạng 6](#_Toc162376745)

[1.3 Mục Đích Và Cơ Chế Hoạt Động Của Giao Thức HTTP 6](#_Toc162376746)

[1.3.1 Mục Đích Sử Dụng Của Giao Thức HTTP 6](#_Toc162376747)

[1.3.2 Cơ Chế Hoạt Động Của Giao Thức HTTP 7](#_Toc162376748)

[1.3.3 Khởi Tạo Kết Nối Giữa Client Và Server 8](#_Toc162376749)

[CHƯƠNG 2: HTTP PROGRAMMING 10](#_Toc162376750)

[2.1 Các Thành Phần Cơ Bản 10](#_Toc162376751)

[2.1.1 Phương Thức HTTP 10](#_Toc162376752)

[2.1.2 HTTP Request 13](#_Toc162376753)

[2.1.3 HTTP Response 14](#_Toc162376754)

[2.2 Cách hoạt động cơ bản của HTTP Programming 14](#_Toc162376755)

[2.2.1 Mô hình Request/Response trong HTTP Programming 14](#_Toc162376756)

[2.2.2 Cú Pháp Của Request/Response 15](#_Toc162376757)

[2.3 Công cụ và thư viện hỗ trợ lập trình . 18](#_Toc162376758)

[2.3.1 Ngôn ngữ lập trình phổ biến trong HTTP programming. 18](#_Toc162376759)

[2.3.2 Thư viện và công cụ hỗ trợ. 20](#_Toc162376760)

[2.4 Ứng Dụng Thực Tế Của HTTP Programming 34](#_Toc162376761)

[2.4.1 RESTful API và JSON 34](#_Toc162376762)

[2.4.2 Nghiên Cứu Trường Hợp 39](#_Toc162376763)

[CHƯƠNG 3: TRIỂN KHAI VÀ THỰC NGHIỆM 45](#_Toc162376764)

[3.1 Kịch Bản 1 45](#_Toc162376765)

[3.1.1 Mô Hình Thực Nghiệm 45](#_Toc162376766)

[3.1.2 Mô Tả Kịch Bản 45](#_Toc162376767)

[3.1.3 Thực Hiện 46](#_Toc162376768)

[3.1.4 Kết luận 53](#_Toc162376769)

[3.2 Kịch Bản 2 53](#_Toc162376770)

[3.2.1 Mô Hình Thực Nghiệm 53](#_Toc162376771)

[3.2.2 Mô Tả Kịch Bản 54](#_Toc162376772)

[3.2.3 Thực Hiện 54](#_Toc162376773)

[3.2.4 Kết Luận 55](#_Toc162376774)

[KẾT LUẬN 56](#_Toc162376775)

[TÀI LIỆU THAM KHẢO 57](#_Toc162376776)

[PHỤ LỤC](#_Toc162376777)

DANH MỤC HÌNH ẢNH

[Hình 2.1: HTTP Request các thông tin được gửi từ client tới server 14](#_Toc162378727)

[Hình 2.2: Sự chuyển đổi thông tin giữa client và server 15](#_Toc162378729)

[Hình 2.3: Cấu trúc gói tin HTTP Request 15](#_Toc162378730)

[Hình 2.4: Cấu trúc gói tin HTTP Response 17](#_Toc162378731)

[Hình 2.5: Mô hình MVT của Django 25](#_Toc162378732)

[Hình 2.6: Quy trình hoạt động của HTTP authentication: 29](#_Toc162378733)

[Hình 2.7: Mô hình web clients và server 39](#_Toc162378734)

[Hình 2.8: Trang web ghép nhiều yếu tố yêu cầu các giao dịch HTTP riêng biệt cho từng nguồn nhúng 40](#_Toc162378735)

[Hình 3.1: Mô hình thực nghiệm 1 45](#_Toc162378736)

[Hình 3.2: Sử dụng phương thức POST trong hàm xử lý(Python) 46](#_Toc162378737)

[Hình 3.3: Sử dụng phương thức POST gửi Request từ Client 46](#_Toc162378738)

[Hình 3.4: Thông báo của server 46](#_Toc162378739)

[Hình 3.5: Server gửi phản hồi cho máy client 47](#_Toc162378740)

[Hình 3.6: Sử dụng phương thức GET trong hàm xử lý(Python) 47](#_Toc162378741)

[Hình 3.7: Sử dụng phương thức GET gửi Request từ client 47](#_Toc162378742)

[Hình 3.8: Thông báo của server 48](#_Toc162378743)

[Hình 3.9: Server gửi phản hồi cho máy client 48](#_Toc162378745)

[Hình 3.10: Sử dụng phương thức PUT trong hàm xử lý(Python) 48](#_Toc162378747)

[Hình 3.11: Sử dụng phương thức PUT gửi Request cập nhật thông tin user 49](#_Toc162378748)

[Hình 3.12: Thông báo của server 49](#_Toc162378749)

[Hình 3.13: Server phản hồi cho client đã cập nhật thành công 50](#_Toc162378750)

[Hình 3.14: Thông báo user trước khi cập nhật 50](#_Toc162378751)

[Hình 3.15: Thông tin user sau khi cập nhật bằng phương thức PUT 51](#_Toc162378752)

[Hình 3.16: Sử dụng phương thức DELETE trong hàm xử lý(Python) 51](#_Toc162378753)

[Hình 3.17: Sử dụng phương thức DELETE gửi Request từ client 52](#_Toc162378754)

[Hình 3.18: Thông báo của server 52](#_Toc162378755)

[Hình 3.19: Server phản hồi cho client đã xóa thành công 52](#_Toc162378756)

[Hình 3.20: Danh sách user sau khi thực hiện phương thức DELETE 52](#_Toc162378757)

[Hình 3.21: Mô hình thực nghiệm 2 53](#_Toc162378758)

[Hình 3.22: Cửa sổ pop-up 54](#_Toc162378759)

[Hình 3.23: Đoạn mã JavaScript trong URL 54](#_Toc162378760)

[Hình 3.24: Thông báo sau khi chèn mã JavaScript 55](#_Toc162378761)

DANH MỤC TỪ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Từ viết tắt | Tên đầy đủ | Giải thích |
| 1 | HTTP | HyperText Transfer Protocol | Giao thức truyền tải siêu văn bản, là một giao thức thuộc tầng 7 - ứng dụng cho các hệ thống thông tin phân tán, công tác. |
| 2 | TCP | Transmission Control Protocol | Giao thức điều khiển truyền vận, là một trong các giao thức cốt lõi của bộ giao thức TCP/IP, thuộc tầng 4 - vận chuyển |
| 3 | IP | Internet Protocol | Giao thức trên Internet |
| 4 | QUIC | Quick UDP Internet Connections | Là một giao thức truyền tải mới của Google, được thiết kế để thay thế TCP và TLS, với mục đích cải thiện tốc độ và độ tin cậy của việc truyền tải dữ liệu trên Internet. QUIC sử dụng UDP thay vì TCP để truyền tải dữ liệu, hỗ trợ multiplexing và khôi phục kết nối nhanh chóng trong trường hợp xảy ra sự cố |
| 5 | UDP | User Datagram Protocol | Là một giao thức truyền thông không đảm bảo đáng tin cậy trong lớp giao vận của mô hình TCP/IP |
| 6 | SCTP | Stream Control Transmission Protocol | Giao thức truyền vận điều khiển dòng |
| 7 | SLL/TLS | Secure Sockets Layer/Transport Layer Security | Là một phương thức mã hóa dữ liệu được sử dụng để bảo vệ sự riêng tư và an toàn khi truyền tải dữ liệu trên Internet. SSL/TLS sử dụng các phương thức mã hóa khác nhau để mã hóa dữ liệu trước khi truyền tải qua mạng, đảm bảo rằng chỉ những người có quyền truy cập mới có thể đọc được dữ liệu này |
| 8 | URL | Uniform Resource Locator | Vị trí của một trang web trên Internet. |
| 9 | API | Application Programming Interface | Là một giao diện lập trình ứng dụng, cung cấp các phương thức và công cụ cho phép các ứng dụng khác truy cập và tương tác với các dịch vụ hoặc ứng dụng khác. |
| 10 | NTLM | New Technology LAN Manager | Là giao thức xác thực được sử dụng trên các mạng bao gồm các hệ thống chạy hệ điều hành Windows và chạy trên các hệ thống độc lập. |
| 11 | URI | Uniform Resource Identifier | Là một chuỗi ký tự được sử dụng để nhận diện các tài nguyên, chuỗi ký tự này có thể là một định vị tài nguyên hoặc một định danh tài nguyên hoặc cả hai. |
| 12 | SQL | Structured Query Language | Là một ngôn ngữ truy vấn được sử dụng để truy xuất và quản lý cơ sở dữ liệu. |
| 13 | XSS | Cross-site Scripting | Là một lỗ hổng bảo mật trong các ứng dụng web, cho phép kẻ tấn công chèn mã độc vào trang web và lấy được thông tin của người dùng. |
| 14 | CRUD | Create, Read, Update, Delete | Là một kỹ thuật lập trình cơ bản được sử dụng trong quản lý cơ sở dữ liệu, gồm các hoạt động cơ bản là tạo mới, đọc, cập nhật và xóa dữ liệu trong cơ sở dữ liệu. |
| 15 | PHP | Personal Home Page | Là đoạn mã lệnh hoặc một chuỗi ngôn ngữ kịch bản được dùng để phát triển các ứng dụng web chạy trên máy chủ |
| 16 | DVWA | Damn Vulnerable Web Application | Là một ứng dụng web được phát triển để học và thực hành các kỹ thuật kiểm thử bảo mật |

LỜI NÓI ĐẦU

1. LÝ DO CHỌN ĐỀ TÀI

Trong thời đại mạng số ngày nay, Internet đã trở thành một phần không thể tách rời trong cuộc sống hàng ngày của con người. Cùng với sự phát triển của công nghệ thông tin, việc trao đổi thông tin và dữ liệu qua mạng trở nên phổ biến hơn bao giờ hết. Trong môi trường này, HTTP (Hypertext Transfer Protocol) là giao thức chính được sử dụng để truyền tải thông tin giữa các máy tính trên Internet. Lý do chọn đề tài về HTTP programming là để tập trung vào nghiên cứu sâu hơn về cách thức hoạt động của giao thức này và áp dụng kiến thức đó vào phát triển các ứng dụng web hiện đại.

Một trong những lý do quan trọng hơn khiến việc nghiên cứu về HTTP trở nên cần thiết là vì vai trò quan trọng của nó trong việc xây dựng các ứng dụng web. Hiểu rõ về cách thức hoạt động của HTTP không chỉ giúp cho nhà phát triển hiểu rõ hơn về cách làm việc của mạng Internet mà còn giúp tối ưu hóa hiệu suất và bảo mật của ứng dụng web. Đặc biệt, trong bối cảnh ngày nay khi các ứng dụng web ngày càng phức tạp và đa dạng, việc nắm vững kiến thức về HTTP là không thể thiếu để đảm bảo sự ổn định và an toàn của hệ thống.

1. MỤC TIÊU CỦA ĐỀ TÀI

Mục tiêu của đề tài là tập trung vào việc nghiên cứu sâu về HTTP programming để hiểu rõ hơn về cách thức hoạt động của giao thức HTTP, các phương thức và tiêu chuẩn liên quan. Đồng thời, mục tiêu của đề tài là áp dụng những kiến thức thu được vào thực hành phát triển ứng dụng web, từ các ứng dụng cơ bản đến các ứng dụng phức tạp và đa nền tảng. Bằng cách này, đề tài mong muốn đóng góp vào việc nâng cao kiến thức và kỹ năng của các nhà phát triển trong lĩnh vực công nghệ thông tin, đặc biệt là trong việc phát triển ứng dụng web an toàn và hiệu quả.

# TỔNG QUAN VỀ GIAO THỨC HTTP

## [Giới Thiệu Về Giao Thức HTTP](bookmark://_bookmark4)

### Khái Niệm Cơ Bản Về Giao Thức HTTP

**HTTP (Hyper Text Transfer Protocol**) là một giao thức nằm ở tầng ứng dụng (Application layer) của tập giao thức TCP/IP, sử dụng để truyền nhận dữ liệu giữa các hệ thống phân tán thông qua internet, cụ thể giao thức hoạt động theo mô hình Client-Server bằng cách thực hiện các quá trình request-response giữa các hệ thống máy tính khác nhau. Giao thức HTTP quy định cấu trúc của các gói tin và cách thức truyền nhận dữ liệu giữa client và server thông qua môi trường internet. Với khả năng truyền dẫn siêu văn bản (text, hình ảnh, âm thanh, video,…), HTTP hiện là nền tảng truyền dẫn dữ liệu của ứng dụng duyệt web hiện nay và được ứng dụng rất nhiều trong các hệ thống Internet of Things.

### Các Phiên Bản Của Giao Thức HTTP

Giao thức HTTP đã trải qua nhiều phiên bản khác nhau, mỗi phiên bản đều mang lại những cải tiến và điều chỉnh để đáp ứng yêu cầu ngày càng phức tạp của môi trường web. Dưới đây là một số phiên bản quan trọng của giao thức HTTP:

* HTTP/0.9: Phiên bản ban đầu của HTTP, được giới thiệu vào năm 1991. Nó chỉ hỗ trợ phương thức GET và không có khả năng chứng thực hay tiêu đề.
* HTTP/1.0: Ra đời vào năm 1996, HTTP/1.0 đã cung cấp nhiều cải tiến so với phiên bản trước. Nó hỗ trợ nhiều phương thức như GET, POST, HEAD, và các tiêu đề mở rộng. Tính năng kết nối giữa các yêu cầu và phản hồi đã được thêm vào, nhưng vẫn giữ nguyên mô hình yêu cầu/phản hồi.
* HTTP/1.1: Được công bố vào năm 1997, HTTP/1.1 là một phiên bản quan trọng, nhanh chóng trở thành tiêu chuẩn cho web. Nó giới thiệu nhiều cải tiến như kết nối kiểu keep-alive (duy trì kết nối), pipelining (gửi nhiều yêu cầu mà không cần đợi phản hồi trước), tiêu đề Host (cho phép máy chủ phục vụ nhiều trang web trên cùng một IP), và nhiều phương thức khác.
* HTTP/2: Được công bố vào năm 2015, HTTP/2 tập trung vào việc tối ưu hóa hiệu suất truyền tải dữ liệu. Nó sử dụng kỹ thuật multiplexing để cho phép nhiều luồng dữ liệu truyền tải trên cùng một kết nối, giúp giảm độ trễ và tăng tốc độ tải trang web. HTTP/2 cũng hỗ trợ nén tiêu đề và ưu tiên các yêu cầu.
* HTTP/3: Là phiên bản mới nhất, HTTP/3 được phát triển với mục tiêu tăng cường hiệu suất và bảo mật. Nó sử dụng giao thức truyền dữ liệu QUIC thay thế cho TCP, giúp giảm độ trễ và tối ưu hóa việc truyền tải dữ liệu trên mạng không ổn định.

Mỗi phiên bản HTTP tiếp theo đều cố gắng cải thiện hiệu suất, bảo mật và khả năng mở rộng của giao thức để đáp ứng những thách thức ngày càng lớn từ sự phát triển nhanh chóng của Internet và ứng dụng web.

### Các Đặc Điểm Của Giao Thức HTTP

Các đặc điểm chính:

* Stateless (Không lưu trạng thái): HTTP là một giao thức không giữ trạng thái giữa các yêu cầu và phản hồi. Mỗi yêu cầu từ client đều độc lập và không phụ thuộc vào các yêu cầu trước đó.
* Text-based (Dựa trên văn bản): Dữ liệu truyền qua HTTP được trình bày dưới dạng văn bản, giúp dễ dàng đọc và hiểu, nhưng cũng tăng kích thước dữ liệu so với các giao thức truyền dữ liệu nhị phân.
* Request-Response (Yêu cầu - Phản hồi): Giao thức hoạt động dựa trên mô hình yêu cầu từ phía client và phản hồi từ phía server.
* Connectionless (Không giữ kết nối): Mỗi yêu cầu và phản hồi đều được xử lý như là một giao tiếp độc lập, và kết nối được đóng sau mỗi yêu cầu.
* Có thể sử dụng trên nhiều phương tiện truyền thông (transport mediums): HTTP có thể hoạt động trên TCP/IP, nhưng cũng có thể sử dụng trên các phương tiện truyền thông khác như UDP hay SCTP.

Ưu điểm của HTTP:

* Dễ hiểu và triển khai: HTTP là giao thức văn bản, giúp việc hiểu và triển khai nhanh chóng và dễ dàng.
* Khả năng tương thích: HTTP được hỗ trợ rộng rãi trên nhiều nền tảng và thiết bị, làm cho nó trở thành một lựa chọn phổ biến cho việc truyền tải dữ liệu trên Internet.
* Stateless giúp giảm áp lực cho server: Không giữ trạng thái giữa các yêu cầu giúp giảm áp lực cho server và dễ dàng mở rộng hệ thống.
* Linh hoạt: Có thể sử dụng trên nhiều ứng dụng, từ trang web đơn giản đến ứng dụng web phức tạp.
* Hỗ trợ proxy và caching: HTTP có hỗ trợ cho proxy, giúp tăng tốc độ truy cập và giảm tải cho server thông qua caching.
* Tuy nhiên, đối với các ứng dụng yêu cầu tính bảo mật cao hoặc cần truyền tải dữ liệu lớn, HTTP có thể không đáp ứng đầy đủ nhu cầu và có thể yêu cầu sự hỗ trợ của các phiên bản cải tiến như HTTPS hay HTTP/2.

## [Lịch Sự Và Phát Triển Về Giao Thức HTTP](bookmark://_bookmark5)

### Xuất Hiện Và Phát Triển Của HTTP

Ngày Xưa:

Giao thức HTTP xuất hiện ban đầu vào những năm 1990 khi World Wide Web (WWW) mới bắt đầu phát triển. Ngày xưa, web chỉ chứa những trang văn bản đơn giản và hình ảnh. Điều này thể hiện trong phiên bản đầu tiên của HTTP, HTTP/0.9, chỉ hỗ trợ phương thức GET để lấy thông tin văn bản từ máy chủ.

HTTP/1.0, ra đời vào năm 1996, đã mở rộng khả năng của HTTP bằng cách hỗ trợ truyền tải đa phương tiện, mã hóa và phương thức POST. Tuy nhiên, các trang web vẫn giữ được sự đơn giản và hầu hết là tĩnh, không có nhiều nội dung động hoặc tương tác.

Hiện Đại:

Ngày nay, Internet và World Wide Web đã phát triển mạnh mẽ. Các trang web đầy đủ chức năng với nhiều nội dung động, ứng dụng web phức tạp, và trải nghiệm người dùng tương tác cao. Trong bối cảnh này, HTTP/1.1 đã trở nên hạn chế do độ trễ và tải trang lớn.

Đáp ứng đòi hỏi ngày càng cao về hiệu suất và trải nghiệm người dùng, HTTP/2 được giới thiệu vào năm 2015. Nó sử dụng kỹ thuật multiplexing để tăng tốc độ truyền tải, giảm độ trễ, và cải thiện hiệu suất trên các kết nối không ổn định.

HTTP/3 đang trong quá trình phát triển và dự kiến sử dụng giao thức QUIC để cải thiện thêm về hiệu suất và bảo mật. QUIC sử dụng UDP thay vì TCP để giảm độ trễ và cải thiện tốc độ truyền tải.

Tổng cộng, sự phát triển của giao thức HTTP phản ánh xu hướng ngày càng phức tạp và đòi hỏi cao trong việc truyền tải thông tin trên Internet.

### Những Cải Tiến Của Giao Thức HTTP Qua Các Phiên Bản

HTTP là giao thức truyền tải dữ liệu trên Internet, và nó đã trải qua nhiều phiên bản khác nhau với các cải tiến đáng kể. Dưới đây là một số cải tiến quan trọng của HTTP qua các phiên bản:

* HTTP/0.9: Phiên bản đầu tiên của HTTP được phát hành vào năm 1991, chỉ hỗ trợ truyền tải các tài liệu HTML đơn giản.
* HTTP/1.0: Phiên bản này được phát hành vào năm 1996, cho phép truyền tải các tài liệu phức tạp hơn, hỗ trợ các thông tin định danh và các loại tài liệu khác nhau.
* HTTP/1.1: Phiên bản này được phát hành vào năm 1999, giới thiệu nhiều cải tiến như đường truyền kéo dài (persistent connection), tạo yêu cầu mới trên cùng một đường truyền, tải nhiều tài liệu trên cùng một kết nối và nén thông tin.
* HTTP/2: Phiên bản này được phát hành vào năm 2015, giới thiệu nhiều cải tiến đáng kể như đa luồng (multiplexing), nén header, ưu tiên hóa yêu cầu (priority), phân tầng và đẩy thông tin từ máy chủ tới trình duyệt.
* HTTP/3: Phiên bản này đang được phát triển và dự kiến sẽ thay thế , HTTP/2. Nó sử dụng giao thức truyền tải QUIC để tăng tốc độ truyền tải, đặc biệt là trong môi trường mạng không ổn định.

Các cải tiến này đã giúp tăng tốc độ truyền tải, cải thiện hiệu suất và đáp ứng nhu cầu truyền tải dữ liệu ngày càng lớn trên Internet.

### Tầm Quan Trọng Của Giao Thức HTTP Đối Với Việc Truyền Tải Dữ Liệu Trên Mạng

HTTP là giao thức quan trọng nhất trong việc truyền tải dữ liệu trên mạng. Nó là một phần không thể thiếu của hệ thống World Wide Web (WWW), cho phép truy cập vào các trang web và tải xuống các tài liệu từ Internet.

HTTP cho phép truyền tải các tài liệu web từ máy chủ đến trình duyệt của người dùng thông qua các giao thức mạng như TCP/IP. Giao thức này hỗ trợ việc truyền tải các tài liệu khác nhau, bao gồm văn bản, hình ảnh, video, âm thanh và các ứng dụng web.

Điều quan trọng là HTTP đảm bảo tính an toàn và bảo mật trong quá trình truyền tải dữ liệu. Các phiên bản mới của HTTP cải tiến việc bảo mật dữ liệu bằng các tính năng như mã hóa SSL/TLS, nén thông tin và ưu tiên hóa yêu cầu.

Ngoài ra, HTTP cũng cung cấp các tính năng tiên tiến để tăng tốc độ truyền tải và cải thiện hiệu suất. Điều này rất quan trọng trong bối cảnh ngày nay, khi người dùng Internet đang yêu cầu tốc độ truyền tải cao hơn và nhu cầu truyền tải dữ liệu ngày càng lớn hơn.

Tóm lại, giao thức HTTP đóng một vai trò rất quan trọng trong việc truyền tải dữ liệu trên mạng, đảm bảo tính an toàn và bảo mật, tăng tốc độ truyền tải và cải thiện hiệu suất.

## Mục Đích Và Cơ Chế Hoạt Động Của Giao Thức HTTP

### Mục Đích Sử Dụng Của Giao Thức HTTP

Giao thức HTTP (Hypertext Transfer Protocol) được sử dụng để truyền tải các tài liệu web từ máy chủ đến trình duyệt của người dùng thông qua mạng Internet. Các mục đích sử dụng chính của HTTP bao gồm:

* Truy cập trang web: HTTP là giao thức cơ bản để truy cập vào các trang web trên Internet. Người dùng sử dụng trình duyệt để gửi yêu cầu đến máy chủ thông qua HTTP và nhận lại các tài liệu web như trang HTML, hình ảnh, video và âm thanh.
* Tải xuống tài liệu: HTTP cho phép người dùng tải xuống các tài liệu từ Internet, bao gồm các tài liệu trang web, ứng dụng và file đính kèm.
* Tương tác với các ứng dụng web: HTTP cho phép các ứng dụng web tương tác với nhau thông qua việc truyền tải các yêu cầu và phản hồi HTTP.
* Tạo dịch vụ web: HTTP là một phần quan trọng của các dịch vụ web, cho phép các ứng dụng khác nhau truyền tải dữ liệu với nhau thông qua mạng Internet.
* Bảo mật dữ liệu: HTTP được sử dụng để bảo mật dữ liệu trên mạng bằng cách sử dụng các tính năng như SSL/TLS để mã hóa dữ liệu và bảo vệ khỏi các cuộc tấn công mạng.

Tóm lại, giao thức HTTP được sử dụng để truyền tải các tài liệu Web và các ứng dụng web, tải xuống các tải liệu từ Internet, bảo mật dữ liệu trên mạng và tạo dịch vụ Web

### Cơ Chế Hoạt Động Của Giao Thức HTTP

Giao thức HTTP (Hypertext Transfer Protocol) hoạt động dựa trên mô hình yêu cầu/ phản hồi (request/response). Đó là, trình duyệt gửi yêu cầu HTTP đến máy chủ, máy chủ xử lý yêu cầu và gửi phản hồi HTTP trở lại cho trình duyệt.

Cơ chế hoạt động cụ thể của giao thức HTTP như sau:

* Người dùng gửi yêu cầu HTTP từ trình duyệt đến máy chủ. Yêu cầu HTTP bao gồm các thông tin như địa chỉ URL (Uniform Resource Locator) của trang web và loại yêu cầu (GET, POST, PUT, DELETE, vv).
* Máy chủ nhận yêu cầu HTTP từ trình duyệt và xử lý yêu cầu đó. Quá trình xử lý có thể bao gồm việc truy vấn cơ sở dữ liệu, xử lý mã nguồn, tạo ra tài liệu web và trả về phản hồi HTTP.
* Máy chủ gửi phản hồi HTTP trở lại cho trình duyệt của người dùng. Phản hồi HTTP bao gồm các thông tin như mã trạng thái HTTP (200 OK, 404 Not Found, vv), các tiêu đề (header) và nội dung của tài liệu web (HTML, hình ảnh, video, vv).
* Trình duyệt của người dùng nhận phản hồi HTTP từ máy chủ và hiển thị tài liệu web cho người dùng.

Ngoài ra, HTTP cũng có các tính năng như phiên làm việc (session) để theo dõi các yêu cầu và phản hồi của người dùng, và cookie để lưu trữ thông tin người dùng.

Tóm lại, cơ chế hoạt động của giao thức HTTP dựa trên mô hình yêu cầu/ phản hồi, trong đó người dùng gửi yêu cầu HTTP đến máy chủ, máy chủ xử lý yêu cầu và gửi phản hồi HTTP trở lại cho trình duyệt của người dùng.

### Khởi Tạo Kết Nối Giữa Client Và Server

Kết nối giữa Client và Server trong mô hình truyền thông web thông thường được thực hiện qua giao thức TCP/IP. Dưới đây là mô tả cách kết nối được khởi tạo:

* 3 bước cổ tay (Three-Way Handshake):
* Bước 1 (SYN): Client gửi một gói tin có cờ SYN (Synchronize) đến Server để mở một kết nối. Gói tin này chứa một số ngẫu nhiên (ISN - Initial Sequence Number) để xác định dữ liệu truyền tải.
* Bước 2 (SYN-ACK): Server nhận được gói tin SYN, và nó sẽ gửi lại một gói tin có cờ SYN và ACK (Acknowledgment) để đồng ý mở kết nối. Gói tin này cũng bao gồm một số ngẫu nhiên và xác định ACK của Client.
* Bước 3 (ACK): Sau khi nhận được gói tin SYN-ACK, Client sẽ gửi lại một gói tin có cờ ACK để xác nhận rằng nó đã nhận được gói tin của Server. Khi Server nhận được gói tin ACK này, quá trình ba bước đã hoàn tất và kết nối được thiết lập.
* Truyền thông dữ liệu:

Sau khi quá trình cổ tay hoàn tất, Client và Server có thể truyền thông dữ liệu qua kết nối đã thiết lập. Dữ liệu được chia thành các gói tin và gửi qua mạng. Mỗi gói tin chứa các thông tin như số thứ tự, số ACK, các cờ kiểm soát, và dữ liệu thực sự.

* Đóng kết nối:
* Khi một trong hai bên muốn đóng kết nối, họ gửi một gói tin có cờ FIN (Finish) để bắt đầu quá trình đóng kết nối. Đối phương sẽ xác nhận với một gói tin ACK, và sau đó gửi một gói tin FIN của riêng mình.
* Khi một bên nhận được gói tin FIN từ đối tác, nó sẽ gửi một gói tin ACK để xác nhận việc đóng kết nối. Quá trình này có thể xảy ra từ cả hai bên.
* Sau khi mỗi bên xác nhận việc đóng kết nối của đối tác, kết nối sẽ được đóng hoàn toàn.

Quy trình trên được gọi là TCP Three-Way Handshake và cung cấp một cách đáng tin cậy để khởi tạo và đóng kết nối giữa Client và Server

# HTTP PROGRAMMING

## Các Thành Phần Cơ Bản

### Phương Thức HTTP

#### Phương Thức GET

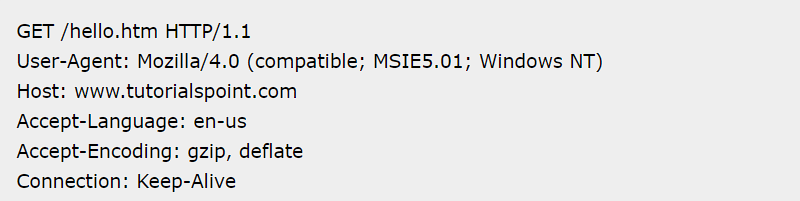
Phương thức GET là một trong các phương thức HTTP cơ bản, được sử dụng để yêu cầu truy cập tài nguyên từ server thông qua URL. Khi client sử dụng phương thức GET, nó sẽ gửi yêu cầu đến server để lấy thông tin từ tài nguyên được chỉ định trong URL.

Các đặc điểm của phương thức GET:

* GET được sử dụng để yêu cầu dữ liệu từ server, chứ không phải để gửi dữ liệu lên server.
* GET gửi các thông tin yêu cầu dưới dạng các tham số query (query parameters) nằm trong URL. Các tham số này có thể được sử dụng để truyền dữ liệu hoặc chỉ định yêu cầu cụ thể.
* Phương thức GET không gây ảnh hưởng đến dữ liệu trên server.
* Phương thức GET có thể được lưu trữ trong bộ nhớ đệm của client và có thể được bookmarked hoặc chia sẻ với người dùng khác.

Tuy nhiên, vì các thông tin yêu cầu được gửi dưới dạng các tham số nằm trong URL, nên phương thức GET có thể bị hạn chế bởi giới hạn độ dài URL của trình duyệt hoặc server. Ngoài ra, các thông tin yêu cầu cũng có thể bị lộ nếu URL được lưu trữ trong lịch sử trình duyệt hoặc được chia sẻ với người khác. Do đó, nếu cần truyền dữ liệu nhạy cảm, phương thức POST hoặc các phương thức khác nên được sử dụng thay cho phương thức GET.

*Ví dụ sau sử dụng phương thức GET để tìm nạp hello.htm:*



#### Phương Thức POST

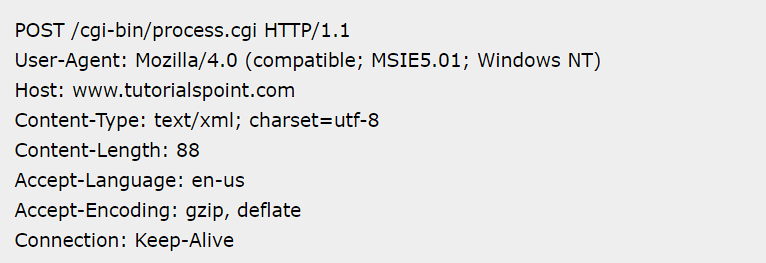
Phương thức POST là một trong các phương thức HTTP cơ bản, được sử dụng để gửi dữ liệu từ client lên server. Khi client sử dụng phương thức POST, nó sẽ gửi yêu cầu đến server kèm theo dữ liệu được đóng gói trong phần thân của yêu cầu.

Các đặc điểm của phương thức POST:

* POST được sử dụng để gửi dữ liệu lên server, chẳng hạn như thông tin đăng nhập, biểu mẫu hoặc bất kỳ dữ liệu nào mà client muốn lưu trữ trên server.
* Dữ liệu được gửi trong phần thân của yêu cầu, không hiển thị trực tiếp trong URL như phương thức GET.
* Phương thức POST có thể gửi dữ liệu lớn hơn và không bị giới hạn bởi giới hạn độ dài URL của trình duyệt hoặc server.
* Phương thức POST có thể được sử dụng để tạo, cập nhật hoặc xóa tài nguyên trên server.

Tuy nhiên, phương thức POST cũng có một số hạn chế. Vì dữ liệu được gửi trong phần thân của yêu cầu, nên phương thức POST có thể tốn nhiều băng thông hơn và gây trễ hơn so với phương thức GET. Ngoài ra, phương thức POST cũng không thể được lưu trữ trong bộ nhớ đệm của trình duyệt hoặc bookmarked, do đó không thể được chia sẻ với người dùng khác.

*Ví dụ sau sử dụng phương thức POST để gửi dữ liệu biểu mẫu đến máy chủ, dữ liệu này sẽ được xử lý bởi một process.cgi:*



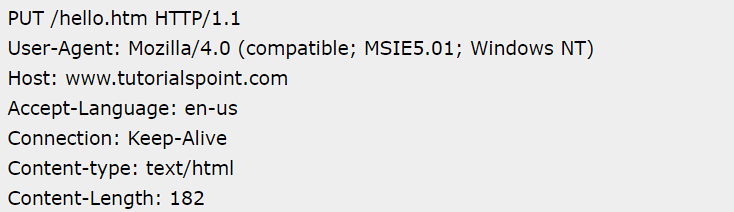
#### Phương Thức PUT

Phương thức PUT là một trong các phương thức HTTP cơ bản, được sử dụng để tạo hoặc cập nhật một tài nguyên trên server. Khi client sử dụng phương thức PUT, nó sẽ gửi yêu cầu đến server để tạo hoặc cập nhật tài nguyên tại đường dẫn được chỉ định.

Các đặc điểm của phương thức PUT:

* Phương thức PUT được sử dụng để tạo hoặc cập nhật một tài nguyên trên server.
* Đường dẫn đích để lưu trữ tài nguyên được chỉ định bởi URL của yêu cầu.
* Dữ liệu được gửi trong phần thân của yêu cầu, và thường là một đối tượng được mã hóa theo một định dạng nào đó (ví dụ: JSON, XML).
* Phương thức PUT thường được sử dụng để tạo mới tài nguyên hoặc cập nhật toàn bộ nội dung của tài nguyên đã tồn tại.

Tuy nhiên, phương thức PUT cũng có một số hạn chế. Vì phương thức PUT được sử dụng để tạo hoặc cập nhật toàn bộ nội dung của tài nguyên, do đó khi client sử dụng phương thức PUT để cập nhật một tài nguyên, nó phải cung cấp toàn bộ nội dung của tài nguyên đó. Nếu client không cung cấp đủ thông tin, server có thể sẽ bị xảy ra lỗi. Ngoài ra, phương thức PUT cũng không được hỗ trợ bởi tất cả các server và firewall, do đó có thể gây khó khăn cho việc triển khai.

*Ví dụ sau yêu cầu máy chủ lưu nội dung thực thể đã cho trong hello.htm ở thư mục gốc của máy chủ:* 

#### Phương Thức DELETE

Phương thức DELETE là một trong các phương thức HTTP cơ bản, được sử dụng để xóa một tài nguyên trên server. Khi client sử dụng phương thức DELETE, nó sẽ gửi yêu cầu đến server để xóa tài nguyên tại đường dẫn được chỉ định.

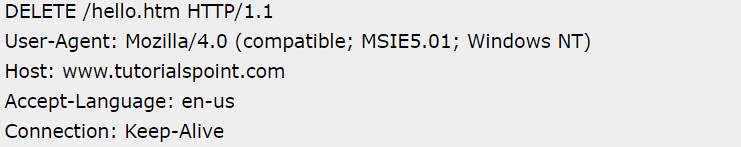
Các đặc điểm của phương thức DELETE:

* Phương thức DELETE được sử dụng để xóa một tài nguyên trên server.
* Đường dẫn đích để xóa tài nguyên được chỉ định bởi URL của yêu cầu.
* Phương thức DELETE không có thân yêu cầu, do đó client không cần gửi bất kỳ dữ liệu nào kèm theo yêu cầu DELETE.

Phương thức DELETE được sử dụng để xóa tài nguyên đã tồn tại trên server. Khi server nhận được yêu cầu DELETE, nó sẽ xóa tài nguyên tương ứng và trả về phản hồi HTTP với mã trạng thái thành công hoặc lỗi tùy thuộc vào kết quả của hoạt động xóa.

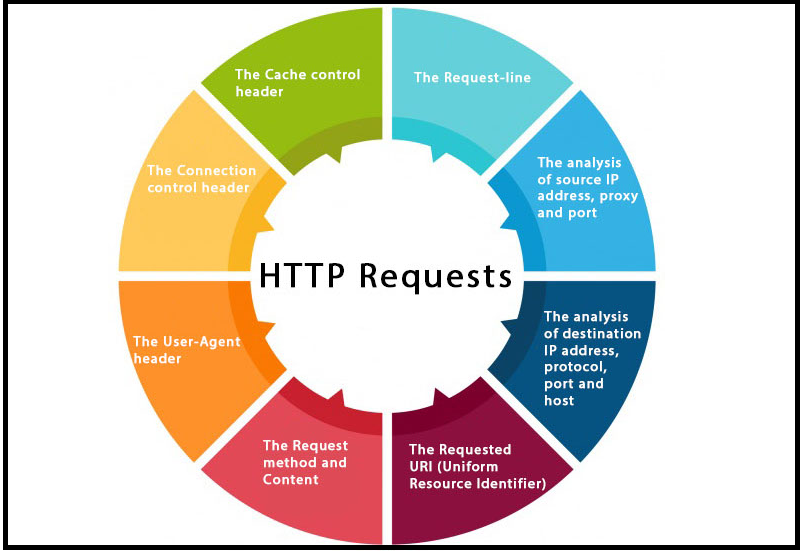
Tuy nhiên, phương thức DELETE cũng có một số hạn chế. Vì phương thức DELETE xóa hoàn toàn tài nguyên được chỉ định, do đó client không thể lấy lại dữ liệu đã xóa sau khi hoàn tất yêu cầu DELETE. Ngoài ra, phương thức DELETE cũng không được hỗ trợ bởi tất cả các server và firewall, do đó có thể gây khó khăn cho việc triển khai.

*Ví dụ sau yêu cầu máy chủ xóa tệp hello.htm đã cho ở thư mục gốc của máy chủ:*



### HTTP Request

Đây chính là tập hợp thông tin được gửi từ các máy khách (client) đến máy chủ (server). Nó là những yêu cầu cần máy chủ tìm kiếm hoặc xử lý và phản hồi kết quả lại client.

Các yêu cầu HTTP được gửi đến có thể là các file dưới dạng XML hoặc Json. Và đó cũng là những định dạng mà cả hai phía client – server đều có thể hiểu được.

HTTP Request các thông tin được gửi từ client tới server

### HTTP Response

HTTP Response (Phản hồi HTTP) là một phần quan trọng trong quá trình giao tiếp giữa máy khách và máy chủ trên Internet. Khi máy chủ nhận được một yêu cầu từ máy khách, nó sẽ xử lý yêu cầu đó và gửi lại một phản hồi. Phản hồi này chứa thông tin và dữ liệu mà máy khách cần để hiển thị trang web hoặc thực hiện các tác vụ khác.

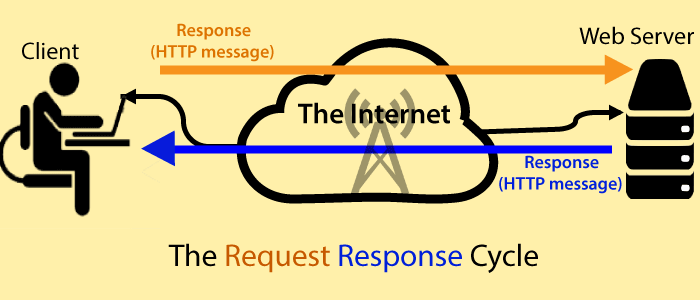
## Cách hoạt động cơ bản của HTTP Programming

### Mô hình Request/Response trong HTTP Programming

#### HTTP Request

Khi bạn lên trình duyệt browser gõ một địa chỉ, trình duyệt sẽ dựa vào tên domain để gửi yêu cầu truy cập đến địa chỉ IP mà domain này trỏ tới.  Lúc này, phía server sẽ phân tích yêu cầu và sẽ gửi luồng xử lý tới vị trí vị trí lưu trữ của mã nguồn PHP (hoặc mã nguồn bất kì). Nhiệm vụ của các mã nguồn là tiếp nhận yêu cầu, phân tích request đó và trả kết quả lại cho client.

#### HTTP Response



Sự chuyển đổi thông tin giữa client và server

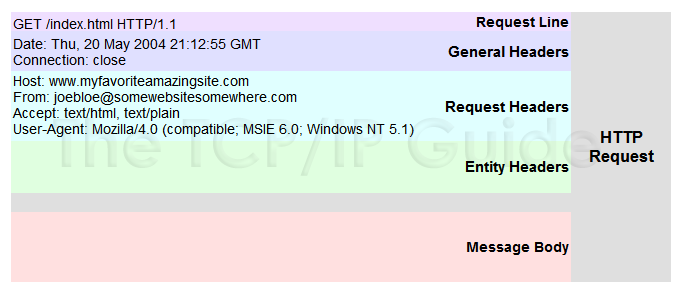
Khi bạn nhập vào một địa chỉ, ví dụ bkns.com, kết quả trả về (response) chính là giao diện của website và các thông tin của header. Như vậy dữ liệu mà server trả về là những đoạn mã HTML kèm theo các thông tin của header.

Browser sẽ dựa vào các thông tin này để hiển thị trạng thái kết quả của request. Mã HTML dùng để hiển thị giao diện của website. Nếu bạn nhập vào một URL không tồn tại thì kết quả trả về không có gì.

### Cú Pháp Của Request/Response

#### HTTP Request

Gồm 3 phần chính là: Request line, Header và Body. Dữ liệu được chia thành các dòng và định dạng kết thúc dòng là <CR><LF> (tương ứng với các ký tự 0x0A,0x0D trong bảng mã ASCII). Riêng dòng cuối cùng của gói tin sẽ được báo hiệu bằng 2 lần định dạng kết thúc: <CR><LF><CR><LF>.

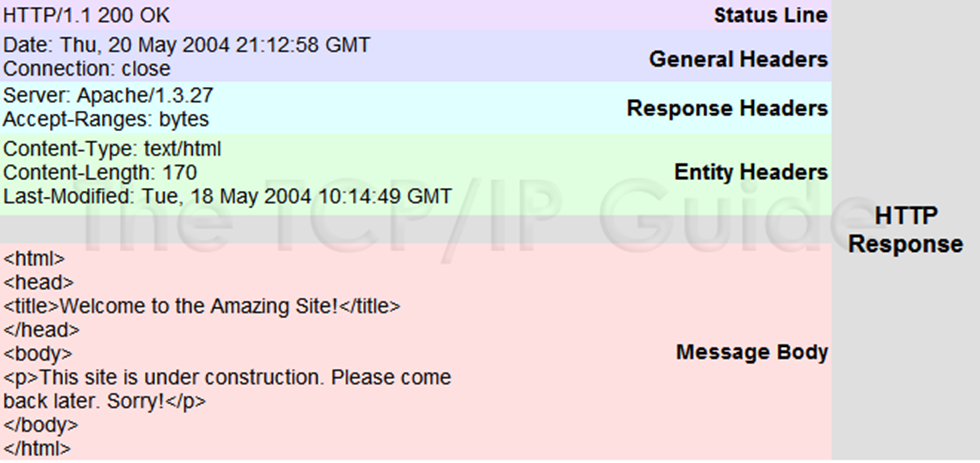


Cấu trúc gói tin HTTP Request

* Request line là dòng đầu tiên của gói HTTP Request, bao gồm 3 trường: phương thức (method), đường dẫn (path – có nhiều bài viết gọi là URL hoặc URI cho trường này) và phiên phản giao thức (HTTP version).
  + Phương thức (method) có thể là: GET, POST, HEAD, PUT và DELETE. Hai phương thức phổ biến nhất là GET và POST, trong ví dụ trên là phương thức GET thường dùng để yêu cầu tài nguyên cung cấp trong trường URL.
  + Đường dẫn (path) dùng để định danh nguồn tài nguyên mà client yêu cầu, bắt buộc phải có ít nhất là dấu “/”.
  + Phiên bản giao thức (HTTP version): là phiên bản HTTP client đang sử dụng (thường là HTTP/1.0 hoặc HTTP/1.1)
* Tiếp theo là các dòng Header, các dòng này là không bắt buộc, viết ở định dạng “Name:Value” cho phép client gửi thêm các thông tin bổ sung về thông điệp HTTP request và thông tin về chính client. Một số header 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: Keep-Alive, Close…
  + Cookie: thông tin HTTP Cookie từ server
* Cuối cùng là phần Body, là dữ liệu gửi từ client đến server trong gói tin HTTP request. Đa số các gói tin gửi theo phương thức GET sẽ có Body trống, các phương thức như POST hay PUT thường dùng để gửi dữ liệu nên sẽ có bao gồm dữ liệu trong trường Body.

#### HTTP Response

Định dạng gói tin HTTP response cũng gồm 3 phần chính là: Status line, Header và Body.

**

Cấu trúc gói tin HTTP Response

* Dòng Status line 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,…
* Các dòng Header line của gói tin response có chức năng tương tựn như gói tin request, giúp server có thể truyền thêm các thông tin bổ sung đến client dưới dạng các cặp “Name:Value”.
* Phần 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.

## Công cụ và thư viện hỗ trợ lập trình .

### Ngôn ngữ lập trình phổ biến trong HTTP programming.

HTTP (Hypertext Transfer Protocol) là giao thức quan trọng trong lập trình web, cho phép trình duyệt gửi yêu cầu đến máy chủ và nhận phản hồi. Nó làm cho việc truyền tải dữ liệu giữa máy khách và máy chủ trở nên có thể, đồng thời cung cấp cơ chế quản lý phiên làm việc, cookie và xác thực. Nhờ HTTP, người dùng có thể trải nghiệm ứng dụng web một cách hiệu quả và tương tác trực tuyến.Một số ngôn ngữ lập trình phía máy chủ:

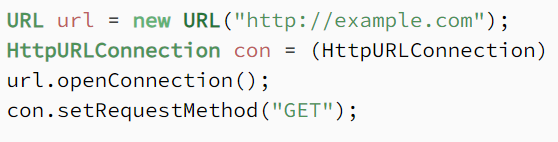
* Java:

Java là ngôn ngữ lập trình phổ biến trong việc phát triển ứng dụng web. Được biết đến với tính đa nền tảng, hướng đối tượng và bảo mật, Java được sử dụng rộng rãi trong hơn 2 thập kỷ. Nó linh hoạt và có thể áp dụng cho mọi loại ứng dụng từ di động đến máy chủ, đồng thời cung cấp sự đáng tin cậy và hiệu suất cao.

Trong ngôn ngữ lập trình java, Lớp HttpUrlConnection cho phép thực hiện các yêu cầu HTTP cơ bản mà không cần sử dụng bất kỳ thư viện bổ sung nào. Tất cả các lớp cần sử dụng đều là một phần của gói java.net. Nhược điểm của việc sử dụng phương pháp này là mã có thể phức tạp hơn so với các thư viện HTTP khác.

Chúng ta có thể tạo một thể hiện của HttpUrlConnection bằng cách sử dụng phương thức openConnection() của lớp URL. Lưu ý rằng phương thức này chỉ tạo ra một đối tượng kết nối nhưng chưa thiết lập kết nối.

Lớp HttpUrlConnection được sử dụng cho tất cả các loại yêu cầu bằng cách đặt thuộc tính requestMethod thành một trong các giá trị: GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE.

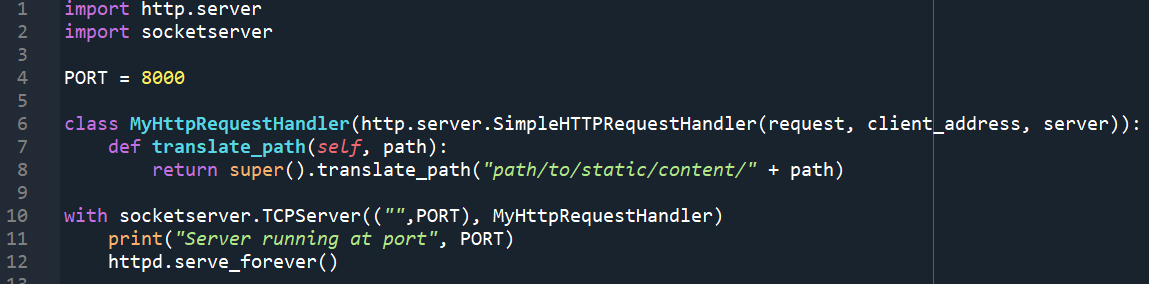
*Ví dụ tạo một kết nối đến một URL cụ thể bằng phương thức GET:* 

* Python:

Python, với cú pháp đơn giản và dễ đọc, là ngôn ngữ lập trình mạnh mẽ và linh hoạt, phổ biến trong lập trình web. Các framework như Django, Flask và FastAPI cung cấp công cụ và thư viện để xây dựng từ dự án nhỏ đến lớn. Python chạy trên nhiều nền tảng và tích hợp tốt với công nghệ web hiện đại, giúp phát triển web trở nên dễ dàng và hiệu quả.

Module http.server trong Python cho phép tạo máy chủ HTTP nhanh chóng để chia sẻ tệp tin, lưu trữ tài nguyên tĩnh, thử nghiệm giao thức HTTP, và thực hiện các đoạn mã Python từ xa chỉ với một lệnh đơn..

*Dưới đây là một ví dụ đơn giản về cách sử dụng module http.server trong Python để khởi chạy một máy chủ HTTP cục bộ và phục vụ nội dung tĩnh từ thư mục hiện tại:*



Đoạn code trên sử dụng module http.server trong Python để tạo ra một máy chủ HTTP cục bộ và phục vụ nội dung tĩnh từ thư mục hiện tại của bạn. Khi chạy đoạn mã này, máy chủ HTTP cục bộ sẽ được khởi chạy và bắt đầu lắng nghe trên cổng 8000. Bạn có thể truy cập vào http://localhost:8000 trên trình duyệt của bạn để xem nội dung tĩnh từ thư mục đã được chỉ định.

Ngôn ngữ lập trình với máy khách:

* JavaScript:

JavaScript là một ngôn ngữ lập trình trong web, cho phép thêm tính năng tương tác và động cho trang web. Nó điều khiển phần tử HTML, tương tác với CSS để thay đổi giao diện và giao tiếp với máy chủ để lấy/gửi dữ liệu mà không cần tải lại trang.

Ta có thể tạo HTTP request đến server trong JavaScript qua XMLHttpRequest, một API trong JavaScript được sử dụng để tạo và gửi các yêu cầu HTTP đến máy chủ và xử lý các phản hồi từ máy chủ mà không cần tải lại trang web.

*Ví dụ sau gửi phương thức request dạng GET đến https://httpbin.org/get*:



Đoạn mã JavaScript trên sử dụng XMLHttpRequest để gửi một yêu cầu GET đồng bộ đến "https://httpbin.org/get" và sau đó in ra trạng thái của yêu cầu (status) và nội dung phản hồi (responseText).

### Thư viện và công cụ hỗ trợ.

#### Postman và các tiện ích kiểm thử

API testing là quá trình xác nhận rằng API hoạt động đúng như mong muốn. API testing là một cách để test tích hợp hệ thống giúp chúng ta kiểm chứng được khả năng tích hợp của một thành phần với những thành phần khác trong hệ thống.

Postman là một ứng dụng mã nguồn mở giúp phát triển và kiểm thử các API. Nó cung cấp môi trường cho các nhà phát triển tạo, chia sẻ, kiểm thử và quản lý API. Postman cho phép tạo yêu cầu HTTP, kiểm thử tự động, quản lý biến môi trường và biến toàn cục, giúp tiết kiệm thời gian trong quá trình phát triển và kiểm thử API. Đặc biệt, nó cung cấp chế độ tương tác thông minh với API, giúp người dùng hiểu rõ hơn về hoạt động và phản hồi của chúng.

* Một số lý do khiến cho Postman được sử dụng rộng rãi:
* Test tự động: Postman cung cấp công cụ cho phép viết các script test và thực thi chúng một cách tự động theo thời gian đặt trước.
* Đa dạng loại test: Postman hỗ trợ nhiều loại test như integration tests, regression tests, functional tests, mock tests, end-to-end tests.
* Ứng dụng đa nền tảng: Postman online để có thể dùng trên bất cứ trình duyệt nào và cả ứng dụng cài đặt trên máy cho Windows, MacOS, Linux.
* Báo cáo: Postman tích hợp sẵn các loại báo cáo về các API đang sử dụng như: Thống kê số lượng API, số lượng workspace, đo lường thời gian, số lượng test thành công, thất bại theo thời gian ...
* Phù hợp với tester ở mọi level.
* Các chức năng chính của Postman:
* API repository: Lưu trữ thông tin về các API, bao gồm phân loại, đặc tả, tài liệu hướng dẫn, luồng thực thi, test case và kết quả test, cho phép cộng tác trên nền tảng duy nhất.
* API tools: Cung cấp các công cụ từ thiết kế đến kiểm thử, viết tài liệu, giả lập dữ liệu và khám phá API trong kho lưu trữ.
* API collaboration: Cho phép tạo và quản lý các workspace, phân quyền cho người dùng từ cá nhân đến toàn bộ thế giới truy cập vào API.
* API governance: Cung cấp tính năng quản lý API dựa trên quy tắc và báo cáo tích hợp, giúp kiểm soát việc tạo và chạy API.
* API Integrations: Tích hợp với các hệ thống phát triển phần mềm như CircleCI, GitLab, Jenkins để tối ưu quá trình phát triển.
* Các thành phần chính của Postman:
  + - 1. Settings

Trong Postman, "Settings" cung cấp các tùy chọn cài đặt chung để tinh chỉnh trải nghiệm sử dụng và hiệu suất của ứng dụng. Một số thông tin cụ thể về danh mục "cài đặt chung" bao gồm: Cài đặt tài khoản, Giao diện người dùng, Cài đặt bảo mật, Cài đặt proxy, Cài đặt kết nối mạng, Cập nhật ứng dụng.

2. Collections trong Postman là một phần quan trọng để tổ chức và quản lý các yêu cầu API. Thành phần cung cấp cách thức lưu trữ, sắp xếp và chia sẻ các yêu cầu API vô cùng dễ dàng. Dưới đây là một số thông tin chi tiết về thành phần Collections trong Postman:

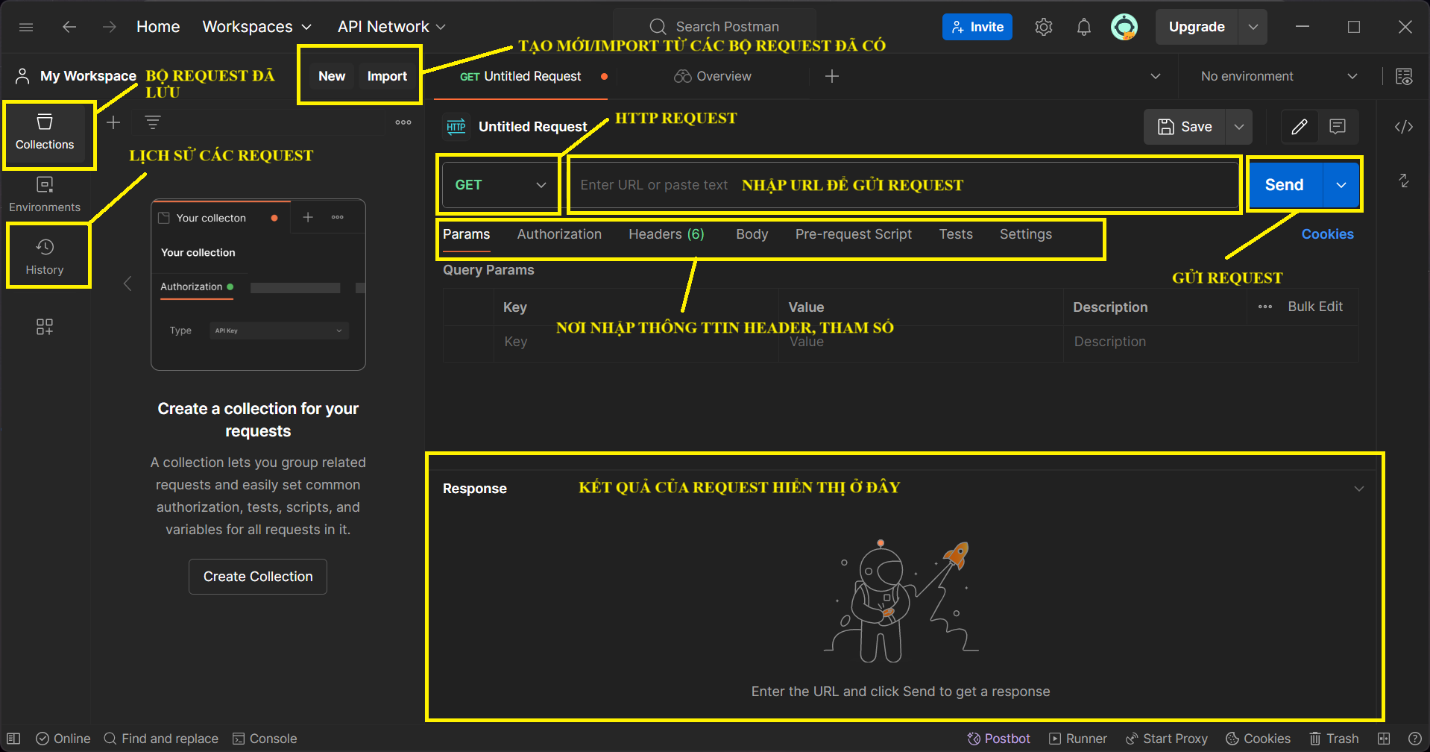
* Lưu trữ yêu cầu: Collections cho phép bạn tạo và lưu trữ các yêu cầu HTTP, bao gồm các phương thức như GET, POST, PUT, DELETE và các yêu cầu khác.
* Sắp xếp và quản lý: Bằng cách sử dụng collections, bạn có thể tổ chức các yêu cầu vào các nhóm cụ thể, giúp dễ dàng quản lý, tìm kiếm và thực hiện các thao tác nhóm trên chúng.
* Biến môi trường: Collections cũng hỗ trợ việc sử dụng các biến môi trường, cho phép bạn tương tác và chia sẻ thông tin giữa các yêu cầu trong collection một cách hiệu quả.
* Chia sẻ và sử dụng chung: Bạn có thể chia sẻ collection với đồng nghiệp hoặc cộng đồng thông qua các liên kết chia sẻ, hoặc sử dụng các collection có sẵn từ cộng đồng Postman.
* Kiểm thử tự động: Collections cũng hỗ trợ việc tạo và quản lý các bộ kiểm thử tự động, cho phép kiểm tra tính đúng đắn và hiệu suất của API một cách tự động.
  + - 1. API Content

Là phần quan trọng nhất, thường đề cập đến dữ liệu cụ thể mà bạn gửi hoặc nhận từ các APIs đang tương tác. Người dùng cần lưu ý các chức năng trong thành phần này như sau:

* Headers (Đầu mục): Thông tin tiêu đề gửi đi hoặc nhận về, bao gồm thông tin như kiểu ngôn ngữ (content-type), định dạng (JSON, XML) và các thông tin xác thực.
* Body (Nội dung): Dữ liệu cụ thể được gửi đi hoặc nhận về từ API, có thể chứa thông tin tương tác như thông tin đăng ký, thông tin sản phẩm hoặc bất kỳ dữ liệu nào khác mà API yêu cầu hoặc gửi lại.
* Response (Phản hồi): Thông tin phản hồi từ API sau khi bạn gửi yêu cầu, bao gồm trạng thái HTTP (200, 404, v.v.), dữ liệu phản hồi và thông tin khác như thời gian phản hồi.

Trong Postman, bạn có thể tương tác với API content thông qua việc tùy chỉnh các yêu cầu HTTP để điều chỉnh headers, body và xử lý các phản hồi từ API. Điều này cho phép người dùng thử nghiệm và xây dựng các yêu cầu API cụ thể dễ dàng.

Các cơ sở chức năng của Postman:



Ưu điểm của Postman:

* Test API dễ dàng: Postman cho phép người dùng tạo request cho API trong các Collection, bao gồm cả việc thử nghiệm. Điều này giúp thực hiện các hoạt động thử nghiệm một cách dễ dàng và hiệu quả hơn.
* Linh động: Postman cho phép import dự án từ nền tảng khác thông qua nhiều loại file và link khác nhau, không yêu cầu phải tạo mới Collection hoặc environment.
* Bộ gỡ lỗi: Postman cung cấp bộ gỡ lỗi giúp kiểm tra dữ liệu xuất, đảm bảo tính vẹn toàn của kết quả sau khi sử dụng.
* Phù hợp: Postman hỗ trợ cả giao diện người dùng (UI) và không giao diện người dùng (non-UI), cung cấp code tự động cho ngôn ngữ lập trình Java, tạo điều kiện thuận lợi cho mọi người sử dụng.

Nhược điểm của Postman:

* Giới hạn của phiên bản miễn phí: Phiên bản miễn phí có hạn chế về số lượt gửi yêu cầu và tính năng cao cấp.
* Hạn chế khi tương tác với các API phức tạp: Postman gặp hạn chế khi làm việc với các API phức tạp.
* Quản lý và theo dõi bộ kiểm thử không hiệu quả: Trong các kiểm thử lớn, việc quản lý và theo dõi bộ kiểm thử trở nên không hiệu quả.

#### Các Framework phổ biến

##### Django

Django là một Web Framework bậc cao miễn phí, sử dụng mã nguồn mở được lập trình bằng Ngôn ngữ Python. Django cho phép người dùng kiểm soát chặt chẽ quá trình phát triển website thông qua một nền tảng duy nhất, có khả năng xử lý các thao tác phức tạp khi lập trình web. Django hiện nay sở hữu một cộng đồng đông đảo người sử dụng và có nhiều tài liệu hỗ trợ cho các developer mới tìm hiểu.

Ưu điểm của Django:

* Đơn giản, tiết kiệm thời gian: Django viết bằng Python, dễ học và sử dụng. Nó cung cấp mã ngắn và đơn giản hơn, với thư viện dữ liệu phong phú và tự động loại bỏ mã trùng lặp.
* Độ bảo mật cao: Django được đánh giá cao về bảo mật với các tính năng như quản lý tài khoản an toàn, mã hóa mật khẩu và xử lý lỗ hổng bảo mật một cách tích cực.
* Có khả năng mở rộng: Django có thể đáp ứng nhu cầu truy cập lớn, phù hợp với các ứng dụng web. Nó cho phép mở rộng dự án bằng cách chạy các máy chủ riêng biệt trên cùng một cơ sở dữ liệu.
* Đa nền tảng: Người dùng có thể thao tác với dự án ở bất kỳ quy mô nào, từ hệ điều hành Mac, Linux cho đến Windows. Django cũng cho phép sử dụng nhiều cơ sở dữ liệu cùng một lúc, đem đến sự tiện lợi cao.
* Cộng đồng người dùng lớn mạnh với hơn 2000 người sử dụng.

Nhược điểm của Django:

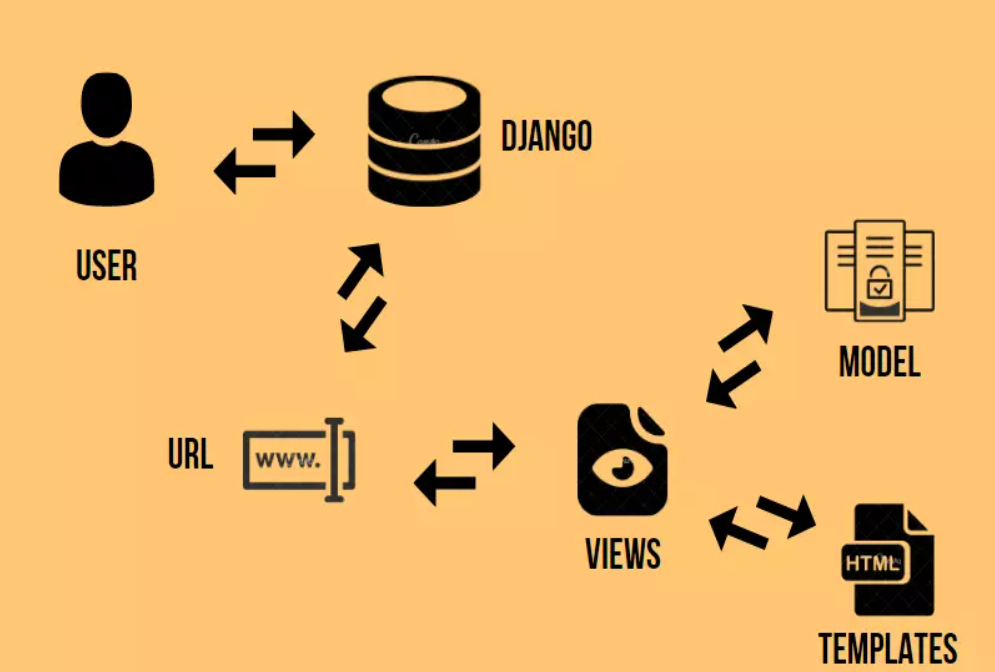
* Có thể tồn tại một vài vấn đề khi phát triển các dự án nhỏ.
* Không được thông báo khi xuất hiện lỗi. Người dùng có thể mất nhiều thời gian để phát hiện vấn đề.

Django quản lý mã cho ứng dụng web bằng cách sử dụng mô hình Mô hình-Khung nhìn-Mẫu (MVT). Khi người dùng truy cập trang web, máy khách gửi yêu cầu đến máy chủ, máy chủ xử lý yêu cầu bằng cách sử dụng cơ sở dữ liệu và gửi thông tin phản hồi về máy khách. Mã máy khách hiển thị thông tin cho người dùng dưới dạng trang web.

Model (M) là lớp có chức năng truy cập và lưu trữ dữ liệu: Từ cách thức truy cập, phương thức dữ liệu, cho đến mối quan hệ các dữ liệu.

View (V) là lớp chứa các logic, giúp truy cập dữ liệu qua Model và truyền đến Template tương ứng.

Template (T) là lớp hiển thị. Nó lưu trữ những gì liên quan đến việc hiển thị dữ liệu trên web hoặc các nền tảng khác.



Mô hình MVT của Django

##### Flask

Python Flask là một framework web dựa trên Python, được thiết kế đơn giản, linh hoạt và dễ sử dụng. Nó cung cấp các tính năng cơ bản như định tuyến URL, xử lý yêu cầu và phản hồi HTTP, và hỗ trợ tính năng mở rộng thông qua các Flask extensions. Flask được ưa chuộng trong cộng đồng Python với tài liệu dễ tiếp cận và phù hợp cho cả người mới học lập trình và lập trình viên có kinh nghiệm.

Những đặc điểm chính của Python Flask:

* Thuộc tính

Người lập trình cần chú ý đến các thuộc tính cốt lõi quan trọng mà Python Flask cung cấp để xây dựng ứng dụng web, chẳng hạn như:

* Routing: Flask cung cấp hệ thống định tuyến linh hoạt cho phép xác định URL và liên kết chúng với mã xử lý trong ứng dụng.
* Templates: Hỗ trợ sử dụng các template engines như Jinja2 để tạo giao diện người dùng linh hoạt và dễ dàng tái sử dụng.
* Request and Response Objects: Hỗ trợ sử dụng các template engines như Jinja2 để tạo giao diện người dùng linh hoạt và dễ dàng tái sử dụng.
* Extensions: Có hệ sinh thái extension đa dạng, bao gồm xác thực người dùng, kết nối cơ sở dữ liệu và quản lý phiên.
* Lightweight and Modular: Thiết kế nhẹ nhàng và dễ tích hợp với các công nghệ và thư viện khác.
* Built-in Development Server: Cung cấp máy chủ phát triển tích hợp sẵn cho việc kiểm thử và phát triển ứng dụng.
* Unit Testing Support: Cung cấp công cụ hỗ trợ kiểm thử đơn vị
* Công cụ tạo mẫu

Trong Python Flask, Jinja2 là công cụ tạo mẫu phổ biến nhất, cho phép tạo ra các template linh hoạt với cú pháp gần gũi với Python. Jinja2 hỗ trợ kế thừa, biến, điều kiện, vòng lặp và các filter để xử lý dữ liệu, giúp phát triển giao diện người dùng linh hoạt và ứng dụng web phong phú với dữ liệu động một cách hiệu quả.

Một số tính năng quan trọng để xây dựng hiệu ứng web:

* Nhẹ và dễ sử dụng: Cấu trúc nhẹ và mã nguồn dễ đọc, giúp người phát triển tiếp cận và tùy chỉnh dễ dàng.
* Định tuyến linh hoạt: Flask cung cấp cơ chế định tuyến linh hoạt, giúp quản lý và xử lý yêu cầu HTTP hiệu quả.
* Công cụ mẫu: Flask tích hợp Jinja2, đây là một loại trình biên dịch mẫu mạnh mẽ cho phép tạo ra các giao diện người dùng.
* Được mở rộng rộng rãi: Mặc dù nhỏ gọn, Flask có khả năng mở rộng mạnh mẽ thông qua các tiện ích và thư viện của cộng đồng.
* Máy chủ phát triển tích hợp: Flask cung cấp máy chủ phát triển tích hợp, giúp kiểm tra và phát triển ứng dụng mà không cần cấu hình bổ sung.
* Gửi yêu cầu RESTful: Flask hỗ trợ xây dựng API và các ứng dụng RESTful một cách hiệu quả.
* Cộng đồng lớn và tích cực..

Một số so sánh giữa Flask và Django:

* Độ phức tạp: Flask được thiết kế đơn giản và linh hoạt, trong khi Django cung cấp giải pháp toàn diện và phức tạp cho việc phát triển ứng dụng web.
* Cấu trúc: Flask không yêu cầu cấu trúc dự án cụ thể, trong khi Django đưa ra cấu trúc tiêu chuẩn dựa trên mô hình "Batteries included".
* Tính linh hoạt: Flask cho phép sử dụng các thư viện và công nghệ bên ngoài một cách tự nhiên, trong khi Django tích hợp nhiều tính năng sẵn có.
* Kích thước: Flask nhẹ nhàng và tập trung vào các công cụ cơ bản, trong khi Django có nhiều tính năng tích hợp sẵn có thể làm tăng kích thước của ứng dụng.
* Học và sử dụng: Flask có cú pháp đơn giản, dễ hiểu và dễ học sử dụng, trong khi Django có khung làm việc lớn hơn và ít tập trung vào tính đơn giản.

Ưu điểm của Flask:

* Linh hoạt: Flask được thiết kế để linh hoạt và dễ dàng tùy chỉnh theo nhu cầu cụ thể của từng dự án.
* Dễ học và sử dụng: Flask có mã cấu trúc đơn giản, dễ hiểu và tài liệu hướng dẫn phong phú, giúp người dùng dễ dàng tiếp cận và sử dụng.
* Đa chức năng: Flask cho phép tích hợp nhiều công nghệ và thư viện bổ sung khác nhau, từ các công cụ front-end cho cơ sở dữ liệu và mạng dịch vụ.
* Mạnh mẽ cho ứng dụng nhỏ đến trung bình: Flask thích hợp để xây dựng các trang web ứng dụng với quy mô từ nhỏ đến trung bình với hiệu suất tốt.

Nhược điểm của Flask:

* Khả năng mở rộng: Flask không cung cấp sẵn các tính năng lớn hơn như các khung, điều này có thể làm tăng mức độ phức tạp khi cần tích hợp nhiều tính năng mở rộng.
* Không phù hợp cho ứng dụng lớn: Bình không phù hợp cho các ứng dụng lớn với quy mô phức tạp và yêu cầu hiệu suất cao.

An toàn và bảo mật trong HTTP Programming.

#### Xác thực và quyền truy cập.

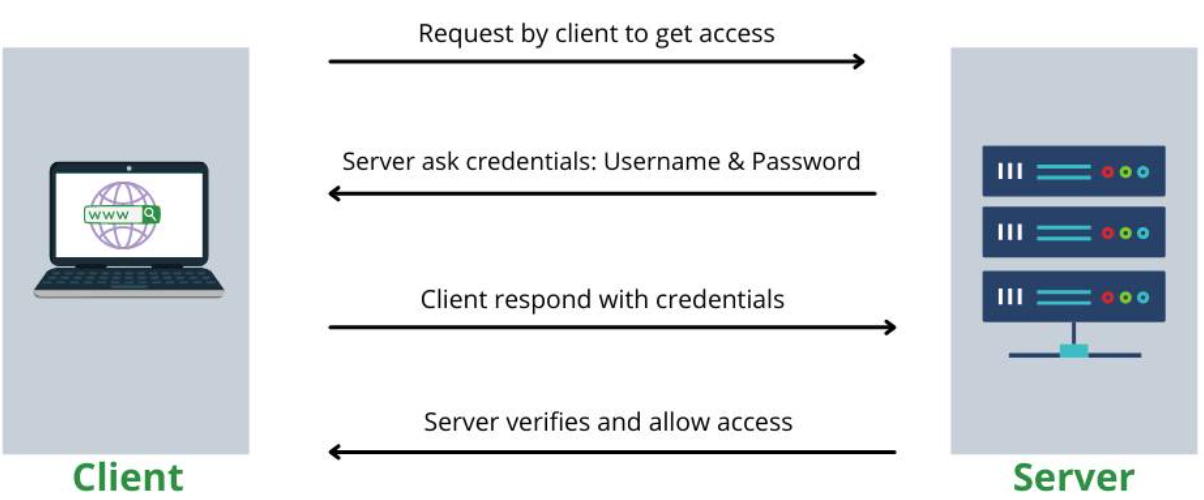
HTTP authentication – còn được gọi là cơ chế xác thực HTTP giúp người dùng xác minh danh tính. Nó là một cơ chế bảo mật kiểm tra xem người dùng có đủ điều kiện truy cập web hay không. Khi máy khách và máy chủ giao tiếp với nhau, dựa vào thành phần HTTP, máy chủ sẽ yêu cầu thông tin đăng nhập để xác thực.

* Các sơ đồ HTTP Authentication phổ biến

Khi truy cập bằng chế độ bảo mật, HTTP thông qua cơ chế xác thực đảm bảo sự an toàn. Có rất nhiều sơ đồ xác thực được sử dụng ở đây, các cái tên phổ biến như:

* HTTP Authentication Schemes: Đây là các phương thức xác thực được máy chủ cung cấp cho người dùng lựa chọn, đảm bảo tính an toàn khi truy cập web.
* Basic authentication (xác thực cơ bản): Yêu cầu người dùng cung cấp tên người dùng và mật khẩu trong định dạng văn bản rõ ràng, dựa trên mô hình thử thách - phản hồi.
* Digest authentication (xác thực thông báo): An toàn hơn xác thực cơ bản, sử dụng giá trị nonce và mã hóa MD5 để tăng cường bảo mật.
* Bearer authentication (xác thực đa yếu tố): Dựa trên mã thông báo và cung cấp lớp bổ sung và bảo mật cấp với mã thông báo để xác thực thông tin từ người dùng.
* NTLM (New Technology LAN Manager – quản lý mạng LAN theo công nghệ mới). Giao thức bảo mật của Windows xác thực danh tính người dùng mà không cần thông tin đăng nhập và truy cập vào tài nguyên.
* Negotiate authentication (đàm phán xác thực): Phiên bản nâng cấp của NTLM, sử dụng giao thức Kerberos để cung cấp xác thực nhanh chóng và an toàn.
* Hoạt động của HTTP Authentication.

HTTP có một framework chung để kiểm soát kiềm truy cập của người dùng vào tài nguyên web. Tất nhiên framework sẽ dựa vào tiêu đề (header) xác thực để sử dụng. Các header hỗ trợ người dùng cách cung cấp thông tin đăng nhập của họ và hồ sơ. Có hai loại header chủ yếu là: WWW-Authenticate header và Proxy Authentication header.



Quy trình hoạt động của HTTP authentication:

Quy trình hoạt động của HTTP authentication:

* Giai đoạn Request (Yêu cầu)

Máy khách đưa ra yêu cầu truy cập dưới dạng ẩn danh. Lúc đó máy chủ không có bất cứ thông tin nào về máy khách yêu cầu.

* Giai đoạn Challenge (Thách thức, yêu cầu xác minh)

Sau khi thấy yêu cầu, máy chủ sẽ phản hồi trạng thái 401 (không được phép). Tiếp đó nó hướng dẫn người dùng cần xác minh danh tính ở phần header.

* Giai đoạn Response (Phản hồi)

Máy khách phản hồi yêu cầu xác thực của máy chủ bằng tên người dùng, mật khẩu để truy cập vào tài nguyên.

* Giai đoạn Proxy Authentication (Xác thực Proxy)

Máy chủ Proxy sẽ gửi trạng thái 407 yêu cầu xác thực. Nó sẽ đứng ra cung cấp xác thực cho máy khách để truy cập tài nguyên. Trường hợp này sẽ chỉ xảy ra khi bạn sử dụng máy chủ Proxy để xác thực.

* Giai đoạn xác minh

Sau khi đã nhận được thông tin đăng nhập, máy chủ sẽ xác minh chúng. Nếu thông tin không hợp lệ, nó trả về trạng thái 403 (không hợp lệ). Nếu trong trường hợp thông tin đăng nhập đúng nó sẽ ra thông báo chào mừng.

Thực tế, quá trình xác thực HTTP Authentication diễn ra khá phức tạp. Các bước trong quy trình trên chỉ là những bước chính, đặc biệt ở những hệ thống lớn. Các lớp bảo mật bổ sung nên được thêm vào để bảo vệ người dùng khỏi tin tặc. Có thể từ xác thực một yếu tố, hai yếu tố hoặc đa yếu tố sẽ đảm bảo sự an toàn người dùng khi giao tiếp trên mạng.

* Quyền truy cập

Trong giao thức HTTP, quyền truy cập là cách mà máy chủ web quản lý và kiểm soát việc truy cập vào các tài nguyên trên máy chủ. Cụ thể, quyền truy cập quy định ai được phép truy cập vào các tài nguyên nào, và trong trường hợp nào. Dưới đây là một số cách thức thông thường để quản lý quyền truy cập trong HTTP:

* Phân quyền dựa trên URI (Uniform Resource Identifier): Máy chủ thiết lập quy tắc cho phép, từ chối truy cập vào các tài nguyên cụ thể dựa trên URI của chúng.

Ví dụ: Một trang web có thể chỉ cho phép người dùng đã đăng nhập truy cập vào các URI bắt đầu bằng "/profile", trong khi từ chối truy cập cho các URI khác.

* Phân quyền dựa trên phương thức HTTP: Máy chủ có thể áp dụng quy tắc phân quyền dựa trên phương thức HTTP được sử dụng.

Ví dụ, một máy chủ có thể chỉ cho phép phương thức GET cho một tài nguyên nhất định, trong khi từ chối các phương thức khác như POST hoặc DELETE.

* Phân quyền dựa trên thông tin xác thực của người dùng: Máy chủ có thể kiểm tra thông tin xác thực của người dùng để quyết định quyền truy cập của họ.

Ví dụ, một máy chủ có thể cho phép truy cập đầy đủ vào một tài nguyên cho người dùng đã đăng nhập, nhưng từ chối truy cập cho những người dùng chưa xác thực.

* Phân quyền dựa trên vai trò (Role-based Access Control): Máy chủ có thể sử dụng cơ chế phân quyền dựa trên vai trò để xác định quyền truy cập của người dùng. Mỗi người dùng có thể được gán một hoặc nhiều vai trò, và các tài nguyên sẽ được cấp quyền truy cập dựa trên vai trò của người dùng đó.

Ví dụ: Một hệ thống quản lý dự án có vai trò như Quản trị viên, Quản lý dự án và Thành viên dự án. Quản trị viên có thể truy cập vào tất cả các tài nguyên, trong khi Thành viên dự án chỉ có thể truy cập vào dự án mà họ được gán vào.

* Phân quyền dựa trên địa chỉ IP: Máy chủ có thể thiết lập quy tắc cho phép hoặc từ chối truy cập từ các địa chỉ IP cụ thể. Điều này cho phép máy chủ hạn chế truy cập từ các vị trí cụ thể hoặc cho phép truy cập chỉ từ một số địa chỉ đáng tin cậy.

Ví dụ: Một dịch vụ có thể chỉ cho phép truy cập từ một danh sách các địa chỉ IP được xác định trước, từ chối tất cả các yêu cầu từ các địa chỉ IP không được phép.

Quyền truy cập trong HTTP đóng vai trò quan trọng trong việc bảo vệ thông tin và tài nguyên của máy chủ, và các cơ chế quyền truy cập cung cấp các phương tiện để đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập vào các tài nguyên cụ thể.

#### Phòng chống các tấn công phổ biến.

##### Các Loại Tấn Công Phổ Biến:

* SQL Injection (SQLi): Tấn công này là một trong những nguy cơ lớn nhất trong lập trình web. Kẻ tấn công có thể chèn mã SQL độc hại vào các truy vấn SQL của ứng dụng web để truy cập hoặc thay đổi cơ sở dữ liệu.
* Cross-Site Scripting (XSS): XSS là một loại tấn công mà kẻ tấn công chèn mã JavaScript độc hại vào các trang web hoặc ứng dụng web, thường thông qua các điểm đầu vào không được kiểm tra kỹ, để đánh cắp thông tin người dùng hoặc kiểm soát trình duyệt của họ.
* HTTP Request Smuggling: Tấn công này tận dụng các lỗ hổng trong cách máy chủ web và proxy xử lý các yêu cầu HTTP, dẫn đến việc kiểm soát hoặc thậm chí lộ thông tin nhạy cảm.
* Session hijacking: là một kỹ thuật tấn công mạng trong đó kẻ tấn công chiếm quyền kiểm soát của một phiên làm việc (session) giữa hai thực thể (thường là một máy tính và một máy chủ), nhằm mục đích thực hiện các hành động không được phép hoặc đánh cắp thông tin nhạy cảm.

Kỹ thuật này thường được thực hiện bằng cách đánh cắp hoặc giả mạo các thông tin xác thực của người dùng, chẳng hạn như cookie, token hoặc session ID, để lấy được quyền truy cập vào phiên làm việc của họ. Khi kẻ tấn công có được thông tin này, họ có thể giả mạo danh tính của người dùng và tiếp tục phiên làm việc mà không cần phải đăng nhập lại. Điều này cho phép kẻ tấn công thực hiện các hành động không mong muốn, như thay đổi thông tin cá nhân, thực hiện giao dịch tài chính, hoặc truy cập vào dữ liệu nhạy cảm.

Có nhiều phương pháp mà kẻ tấn công có thể thực hiện session hijacking, bao gồm:

* Packet Sniffing: Kẻ tấn công có thể sử dụng các công cụ như sniffer để bắt và phân tích các gói tin truyền qua mạng để lấy thông tin đăng nhập, như username và password.
* Session Fixation: Kẻ tấn công có thể tạo ra một phiên làm việc và cố ý truyền ID phiên này đến người dùng, sau đó chờ họ đăng nhập. Khi người dùng đăng nhập với ID phiên đã được cung cấp, kẻ tấn công sẽ sử dụng ID phiên này để chiếm quyền kiểm soát phiên của họ.
* Cross-site Scripting (XSS): XSS cho phép kẻ tấn công chèn mã JavaScript độc hại vào một trang web, khiến người dùng truy cập vào trang web đó sẽ thực hiện các hành động không mong muốn mà không biết. Trong trường hợp này, mã JavaScript có thể lấy session ID của người dùng và gửi nó đến kẻ tấn công.
* Man-in-the-Middle (MitM) Attack: Trong cuộc tấn công này, kẻ tấn công chèn mình vào giữa người dùng và máy chủ để theo dõi và thậm chí thay đổi thông tin truyền qua. Khi người dùng đăng nhập, kẻ tấn công có thể lấy thông tin đăng nhập của họ và tiếp tục phiên làm việc..
* HTTP Header Injection: Kẻ tấn công chèn các thông tin độc hại vào các tiêu đề HTTP để thực hiện các cuộc tấn công phá vỡ hoặc lừa đảo.

##### **Biện Pháp Phòng Chống**

Để bảo vệ ứng dụng web khỏi các cuộc tấn công liên quan đến HTTP, dưới đây là các biện pháp phòng chống cụ thể:

* Kiểm Tra Đầu Vào (Input Validation)

Mô Tả: Hãy luôn kiểm tra và xác thực các đầu vào từ người dùng trước khi xử lý chúng. Điều này giúp ngăn chặn các cuộc tấn công như SQL Injection (SQLi) và Cross-Site Scripting (XSS).

Ví Dụ: Trong một biểu mẫu đăng nhập, kiểm tra các trường như "Tên người dùng" và "Mật khẩu" để đảm bảo chúng không chứa các ký tự đặc biệt hay các câu lệnh SQL.

Tấn công liên quan: SQL Injection (SQLi), Cross-Site Scripting (XSS), HTTP Header Injection, HTTP Parameter Pollution (HPP).

* Sử Dụng HTTPS

Mô Tả: Sử dụng HTTPS để mã hóa dữ liệu trên đường truyền, ngăn chặn các cuộc tấn công giữa đường truyền và bảo vệ thông tin người dùng.

Ví Dụ: Triển khai chứng chỉ SSL/TLS và sử dụng giao thức HTTPS cho các trang web, đặc biệt là những trang yêu cầu nhập liệu nhạy cảm như trang đăng nhập hoặc thanh toán.

Tấn công liên quan: Man-in-the-Middle (MitM), sniffing attacks.

* Thiết Lập Xác Thực và Ủy Quyền Mạnh Mẽ

Mô Tả: Xác thực người dùng và quản lý phiên làm việc một cách an toàn để ngăn chặn tấn công Session Hijacking và các cuộc tấn công khác.

Ví Dụ: Sử dụng các cơ chế xác thực mạnh mẽ như JWT (JSON Web Tokens) hoặc OAuth để quản lý phiên làm việc và quyền truy cập.

Tấn công liên quan: Session Hijacking, Password Attacks.

* Sử Dụng Các Tiêu Đề Bảo Mật HTTP (HTTP Security Headers)

Mô Tả: Thiết lập các tiêu đề bảo mật HTTP như Content Security Policy (CSP) và X-Frame-Options để giảm thiểu rủi ro từ các cuộc tấn công như XSS và clickjacking.

Ví Dụ: Thiết lập tiêu đề CSP để chỉ cho phép tải tài nguyên từ các nguồn được tin cậy và chặn việc thực thi các mã script không an toàn.

Tấn công liên quan: Cross-Site Scripting (XSS), Clickjacking.

* Giám Sát và Ghi Log

Mô Tả: Thực hiện giám sát liên tục và ghi log để theo dõi hoạt động của ứng dụng web, từ đó phát hiện và ứng phó với các hoạt động bất thường hoặc tấn công.

Ví Dụ: Thiết lập hệ thống giám sát và ghi log để lưu trữ thông tin về các yêu cầu HTTP đến và từ máy chủ, bao gồm cả các chi tiết về người gửi yêu cầu và nội dung của yêu cầu.

Tấn công liên quan: Các cuộc tấn công mà không thể phát hiện ngay lập tức như Advanced Persistent Threats (APT).

## Ứng Dụng Thực Tế Của HTTP Programming

### RESTful API và JSON

#### Giới thiệu về RESTful API

**API (Giao diện lập trình ứng dụng)** là một giao diện xác định các quy tắc bạn phải tuân theo để giao tiếp với các hệ thống phần mềm khác nhau. Các nhà phát triển tiết lộ hoặc tạo ra API để các ứng dụng khác có thể giao tiếp với ứng dụng của họ theo cách tự động. Ví dụ, ứng dụng theo dõi thời gian làm việc nội bộ tiết lộ một API yêu cầu tên đầy đủ của nhân viên và một khoảng thời gian cụ thể. Khi nhận được thông tin này, nó xử lý thời gian làm việc của nhân viên và trả về số giờ làm việc trong khoảng thời gian đó.

**REST (Representational State Transfer)** là một kiến trúc phần mềm đặt ra điều kiện cho cách một API nên hoạt động. REST ban đầu được tạo ra như một hướng dẫn để quản lý việc giao tiếp trên một mạng phức tạp như internet. Bạn có thể sử dụng kiến trúc dựa trên REST để hỗ trợ giao tiếp hiệu suất và đáng tin cậy ở quy mô lớn. Bạn có thể triển khai và điều chỉnh nó một cách dễ dàng, mang lại khả năng qu ản lý và di động trên nhiều nền tảng cho bất kỳ hệ thống API nào.

**RESTful API (Giao diện lập trình ứng dụng hợp nhất trạng thái biểu diễn**) là một giao diện mà hai hệ thống máy tính sử dụng để trao đổi thông tin một cách an toàn qua internet. Hầu hết các ứng dụng kinh doanh phải giao tiếp với các ứng dụng nội bộ và bên thứ ba khác nhau để thực hiện các nhiệm vụ khác nhau. RESTful API hỗ trợ việc trao đổi thông tin này vì chúng tuân theo các tiêu chuẩn giao tiếp phần mềm an toàn, đáng tin cậy và hiệu quả.

##### Các nguyên tắc cơ bản của kiến trúc REST:

* **Giao diện thống nhất**: Giao diện thống nhất là quan trọng đối với thiết kế bất kỳ dịch vụ web RESTful nào. Nó chỉ ra rằng máy chủ chuyển thông tin theo một định dạng tiêu chuẩn. Định dạng nguồn được gọi là biểu diễn trong REST. Biểu diễn này có thể khác biệt so với biểu diễn nội bộ của nguồn trên ứng dụng máy chủ. Ví dụ, máy chủ có thể lưu trữ dữ liệu dưới dạng văn bản nhưng gửi nó dưới dạng biểu diễn HTML.
* **Stateless**: Trong kiến trúc REST, tính statelessness (trạng thái không lưu giữ) đề cập đến một phương pháp giao tiếp trong đó máy chủ hoàn thành mọi yêu cầu của máy khách độc lập với tất cả các yêu cầu trước đó. Máy khách có thể yêu cầu nguồn theo bất kỳ thứ tự nào và mỗi yêu cầu là stateless hoặc độc lập với các yêu cầu khác. Rất quan trọng, mô hình này đảm bảo rằng máy chủ có thể hiểu và hoàn tất yêu cầu mỗi lần.
* **Hệ thống lớp**: Trong kiến trúc hệ thống lớp, máy khách có thể kết nối với các trung gian được ủy quyền giữa máy khách và máy chủ, và nó vẫn nhận được phản hồi từ máy chủ. Máy chủ cũng có thể chuyển tiếp yêu cầu đến các máy chủ khác. Bạn có thể thiết kế dịch vụ web RESTful của mình để chạy trên nhiều máy chủ với nhiều lớp như bảo mật, ứng dụng và logic kinh doanh, làm việc cùng nhau để đáp ứng yêu cầu của máy khách. Những lớp này không thể nhìn thấy từ phía máy khách.
* **Có thể lưu cache**: Dịch vụ web RESTful hỗ trợ việc lưu cache, là quá trình lưu trữ một số phản hồi trên máy khách hoặc trung gian để cải thiện thời gian phản hồi của máy chủ. Ví dụ, nếu bạn truy cập một trang web có các hình ảnh tiêu đề và chân trang chung trên mọi trang, máy chủ phải gửi lại cùng các hình ảnh này mỗi khi bạn chuyển đến một trang mới. Để tránh điều này, máy khách có thể lưu trữ (hoặc cache) những hình ảnh này sau phản hồi đầu tiên và sau đó sử dụng chúng trực tiếp từ cache. Dịch vụ web RESTful kiểm soát việc lưu cache bằng cách sử dụng các phản hồi API mô tả chúng là có thể lưu cache hoặc không thể lưu cache.
* **Mã trên yêu cầu**: Trong kiến trúc REST, máy chủ có thể tạm thời mở rộng hoặc tùy chỉnh chức năng máy khách bằng cách chuyển mã lập trình phần mềm cho máy khách. Ví dụ, khi bạn điền một biểu mẫu đăng ký trên bất kỳ trang web nào, trình duyệt của bạn ngay lập tức làm nổi bật bất kỳ sai sót nào bạn gặp, chẳng hạn như số điện thoại không đúng. Nó có thể làm điều này nhờ vào mã được gửi từ máy chủ.

##### Các lợi ích của RESTful API bao gồm:

* **Khả năng mở rộng**: Hệ thống triển khai REST API có thể mở rộng hiệu quả vì REST tối ưu hóa giao tiếp máy khách-máy chủ. Stateless giảm tải máy chủ vì máy chủ không phải giữ lại thông tin yêu cầu máy khách trước đó. Lưu trữ cache được quản lý tốt giúp loại bỏ hoặc giảm bớt một số giao tiếp máy khách-máy chủ. Tất cả những tính năng này hỗ trợ khả năng mở rộng mà không gây ra hạn chế hiệu suất.
* **Linh động**: RESTful web service hỗ trợ sự tách biệt toàn bộ giữa máy khách và máy chủ. Nó đơn giản và giảm kết nối các thành phần máy chủ khác nhau để mỗi phần có thể phát triển độc lập. Thay đổi công nghệ hoặc nền tảng tại máy chủ ứng dụng không ảnh hưởng đến máy khách ứng dụng. Khả năng lớp ứng dụng tăng cường linh động hơn nữa. Ví dụ, nhà phát triển có thể thay đổi lớp cơ sở dữ liệu mà không cần phải viết lại logic ứng dụng.
* **Độc lập**: REST API độc lập với công nghệ được sử dụng. Bạn có thể viết cả máy khách và máy chủ ứng dụng bằng nhiều ngôn ngữ lập trình mà không ảnh hưởng đến thiết kế API. Bạn cũng có thể thay đổi công nghệ dưới tầng máy chủ hoặc máy khách mà không ảnh hưởng đến giao tiếp.

##### Cách RESTful API hoạt động:

Dưới đây là các bước chung cho RESTful API :

* Chọn phương thức HTTP phù hợp với yêu cầu máy khách muốn thực hiện. Ví dụ, sử dụng GET để truy cập dữ liệu, POST để tạo mới, PUT để cập nhật, DELETE để xóa,…
* Máy khách gửi yêu cầu đến máy chủ. Máy khách tuân theo tài liệu API để định dạng yêu cầu một cách máy chủ hiểu được.
* Máy chủ xác thực máy khách và xác nhận rằng máy khách có quyền thực hiện yêu cầu đó.
* Máy chủ nhận và xử lý yêu cầu nội bộ.
* Máy chủ trả về một phản hồi cho máy khách. Phản hồi chứa thông tin cho biết liệu yêu cầu có thành công hay không. Phản hồi cũng bao gồm bất kỳ thông tin nào mà máy khách đã yêu cầu.
* Máy khách xử lý dữ liệu trả về theo cách phù hợp với ứng dụng của nó. Thông thường, dữ liệu được trả về dưới dạng JSON hoặc XML và được chuyển đổi thành các đối tượng hoặc cấu trúc dữ liệu phù hợp với ngôn ngữ lập trình của máy khách.Nếu có lỗi trong quá trình gửi và xử lý yêu cầu thì mã trạng thái trong phản hồi sẽ cung cấp thông tin về lỗi đó.

Các công cụ và thư viện thường được sử dụng để làm việc với RESTful API bao gồm cả Postman, cURL, Fetch API trong JavaScript, Retrofit trong Android, và thư viện HTTP cho ngôn ngữ lập trình cụ thể như requests trong Python.

Mặc dù có nhiều kiến trúc API khác nhau, RESTful API vẫn được sử dụng rộng rãi trong ngành công nghiệp và là một trong những lựa chọn phổ biến cho việc tạo các giao diện lập trình ứng dụng linh hoạt và hiệu quả.

#### JSON Trong Truyền Thông HTTP

JSON (JavaScript Object Notation) là một định dạng tiêu chuẩn để biểu diễn dữ liệu có cấu trúc. Mặc dù JSON bắt nguồn từ ngôn ngữ lập trình JavaScript, nhưng hiện nay nó đã trở thành một phương tiện phổ biến để trao đổi dữ liệu giữa các hệ thống. Hầu hết các API hiện đại đều chấp nhận các yêu cầu JSON và phát ra các phản hồi dưới dạng JSON

##### Loại dữ liệu JSON

* Chuỗi trong dấu ngoặc kép: "Hello world"
* Chuỗi trong dấu ngoặc kép: "Hello world"
* Boolean true/false: true, false
* Giá trị Null: null

##### Cơ bản về cú pháp JSON:

* Đối tượng (Object) sử dụng dấu ngoặc nhọn ‘ {} ‘ và cặp khóa-giá trị
* Mảng (Array) sử dụng dấu ngoặc vuông ‘ [] ‘ cho các giá trị có thứ tự
* Khóa và giá trị (Key-value) được phân tách bằng dấu hai chấm ‘:’
* Dấu phẩy ‘ , ’ phân tách các phần tử
* Dấu ngoặc kép " được sử dụng cho giá trị chuỗi

##### Đặc Điểm của JSON

* Cấu trúc nhẹ, con người dễ đọc
* Dựa trên các cặp key-value (khóa-giá trị) và danh sách có thứ tự
* Sử dụng mảng, đối tượng, chuỗi, số, boolean, null
* Định dạng độc lập ngôn ngữ để biểu diễn và truyền dữ liệu
* Lựa chọn phổ biến thay thế cho XML cho các API web

##### Ưu điểm của JSON khi sử dụng trong REST:

* JSON rất thích hợp để biểu diễn và truyền dữ liệu qua HTTP
* Sự đơn giản của JSON tương thích tốt với việc mô hình hóa tài nguyên REST
* JSON có hiệu suất cao so với XML

Tóm lại, JSON nhẹ và hoàn toàn phù hợp với các nguyên tắc của REST để tiếp cận tài nguyên qua HTTP. Sự kết hợp mạnh mẽ này khiến chúng trở nên lý tưởng cho các dịch vụ web.

### Nghiên Cứu Trường Hợp

#### Xây Dựng Ứng Dụng Web Động

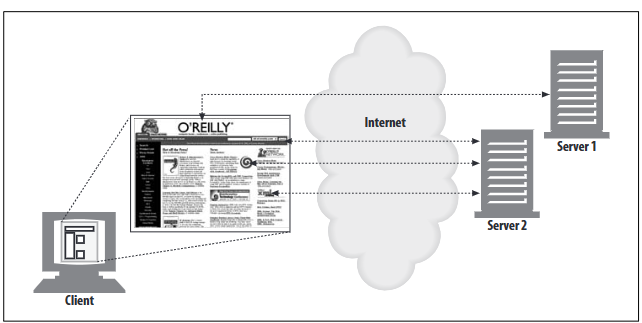
Mô hình web clients và server

Nội dung web động trên máy chủ web. Máy chủ web sử dụng giao thức HTTP, nên chúng thường được gọi là HTTP servers. Những HTTP servers này lưu trữ dữ liệu của Internet và cung cấp dữ liệu khi nó được yêu cầu bởi HTTP clients. Các khách hàng gửi các yêu cầu HTTP đến máy chủ, và máy chủ trả lại dữ liệu được yêu cầu trong các phản hồi HTTP, như được mô tả trong Hình 1-1. Cùng nhau, HTTP client và HTTP servers tạo thành các thành phần cơ bản của World Wide Web.

Web clients and servers

Bạn có thể sử dụng HTTP client hàng ngày. Những HTTP client phổ biến nhất là trình duyệt web, chẳng hạn như Microsoft Internet Explorer hoặc Netscape Navigator. Trình duyệt web yêu cầu các đối tượng HTTP từ máy chủ và hiển thị các đối tượng trên màn hình của bạn.

Khi bạn duyệt đến một trang web, chẳng hạn như

**“http://www.oreilly.com/index.html,”** trình duyệt của bạn gửi một yêu cầu HTTP đến máy chủ **www.oreilly.com** (Hình 2.7). Máy chủ cố gắng tìm đối tượng mong muốn (trong trường hợp này là **“/index.html”)** và nếu thành công, máy chủ sẽ gửi đối tượng đó đến khách hàng trong một phản hồi HTTP, cùng với loại đối tượng, độ dài của đối tượng và các thông tin khác.

Trang web ghép nhiều yếu tố yêu cầu các giao dịch HTTP riêng biệt cho từng nguồn nhúng

Một ứng dụng thường phát ra nhiều giao dịch HTTP để hoàn thành một nhiệm vụ. Chẳng hạn, một trình duyệt web phát ra một chuỗi giao dịch HTTP để tải và hiển thị một trang web đầy đủ đồ họa. Trình duyệt thực hiện một giao dịch để tải “bộ xương” HTML mô tả bố cục trang, sau đó phát ra các giao dịch HTTP bổ sung để mỗi hình ảnh nhúng, trình chơi đồ họa, ứng dụng Java, ... Các nguồn nhúng này có thể thậm chí nằm trên các máy chủ khác nhau, như minh họa trong hình 2.8. Do đó, một "trang web" thường là một bộ sưu tập các nguồn, không phải một nguồn duy nhất.

Trang web ghép nhiều yếu tố yêu cầu các giao dịch HTTP riêng biệt cho từng nguồn nhúng.

#### Tích Hợp Dịch Vụ Với HTTP Programming

* API:

Tích hợp API vào HTTP programming mang lại nhiều lợi ích quan trọng, đặc biệt là trong việc phát triển ứng dụng và hệ thống phân tán. Dưới đây là một số lợi ích chính:

* Tương Tác và Giao Tiếp:API cho phép các thành phần của ứng dụng tương tác và truyền thông dữ liệu qua mạng. Điều này tạo ra khả năng kết nối giữa các thành phần ứng dụng và giữa các ứng dụng khác nhau.
* Mô Hình Phân Tán:Sử dụng API giúp xây dựng mô hình phân tán cho ứng dụng. Các dịch vụ có thể hoạt động độc lập và kết nối với nhau thông qua HTTP, tạo ra kiến trúc linh hoạt và mở rộng.
* Khả Năng Mở Rộng và Linh Hoạt:API giúp ứng dụng dễ dàng mở rộng và tích hợp các tính năng mới mà không làm ảnh hưởng đến các phần khác của hệ thống. Điều này tăng tính linh hoạt và dễ bảo trì của ứng dụng.
* Tích Hợp Dịch vụ Bên Ngoài:Các dịch vụ của bên thứ ba có thể được tích hợp dễ dàng thông qua API. Điều này giúp mở rộng khả năng sử dụng các tài nguyên và dịch vụ từ các nguồn khác nhau.
* Tính Tương Thích Đa Nền Tảng:Với HTTP là giao thức chung, API tạo ra tính tương thích đa nền tảng. Ứng dụng có thể chạy trên nhiều loại thiết bị và hệ điều hành khác nhau mà không gặp vấn đề tương thích.
* Bảo Mật và Phân Quyền:API cung cấp cơ hội để triển khai các biện pháp bảo mật như xác thực và ủy quyền. Điều này đảm bảo rằng chỉ có những người dùng được phép có thể truy cập và sử dụng các dịch vụ.
* Tự Động Hóa:API hỗ trợ tự động hóa quy trình làm việc thông qua việc kích thích sự kiện và thông báo. Các webhooks cho phép tự động gửi thông báo khi có sự kiện quan trọng xảy ra.

Ví dụ về lợi ích của việc tích hợp API vào HTTP programming:

* Sử dụng API Google Maps để hiển thị bản đồ trong ứng dụng của bạn.
* Sử dụng API Facebook để đăng nhập người dùng vào ứng dụng của bạn.
* Sử dụng dịch vụ thanh toán trực tuyến để xử lý thanh toán trong ứng dụng của bạn.
* OAuth(OAuth 2.0) và token-based authentication:

Token-based authentication là một phương pháp phổ biến được sử dụng trong ứng dụng web để xác thực và phân quyền người dùng. Nó cung cấp một cách tiếp cận an toàn và không lưu trạng thái cho việc xác thực người dùng, trong đó khách hàng nhận và trình bày một token để truy cập tài nguyên được bảo vệ. Nó cho phép khách hàng xác thực bản thân bằng cách trình bày một token thay vì gửi thông tin xác thực với mỗi yêu cầu. Token thường được tạo và cấp phát bởi máy chủ xác thực sau khi đăng nhập thành công và được sử dụng để xác minh danh tính và tình trạng phân quyền của người dùng.

Cơ chế hoạt động như sau:

* Đăng nhập: Người dùng cung cấp thông tin xác thực của họ cho máy chủ, thường là thông qua một biểu mẫu đăng nhập trên giao diện người dùng. Thông tin này có thể bao gồm tên người dùng và mật khẩu.
* Xác thực: Máy chủ nhận thông tin đăng nhập từ người dùng và kiểm tra tính hợp lệ của chúng. Điều này có thể bao gồm việc so sánh tên người dùng và mật khẩu với cơ sở dữ liệu người dùng được lưu trữ trên máy chủ. Nếu thông tin xác thực đúng, máy chủ sẽ tiếp tục vào bước tiếp theo.
* Tạo Token: Sau khi xác thực thành công, máy chủ tạo ra một token duy nhất. Token này có thể được tạo bằng cách sử dụng một thuật toán mã hóa, thường dựa trên chuỗi ký tự ngẫu nhiên hoặc bằng cách mã hóa các thông tin như tên người dùng và thời gian hết hạn.
* Phản hồi: Máy chủ trả về token cho người dùng thông qua một phản hồi HTTP. Thông thường, token sẽ được gửi trong phần header hoặc body của phản hồi HTTP.
* Gửi Token: Từ giờ trở đi, người dùng sẽ gửi token này cùng với mỗi yêu cầu mà họ gửi đến máy chủ. Thông thường, token được gửi trong phần header của yêu cầu HTTP, thông qua một header như Authorization.
* Kiểm tra Token: Máy chủ nhận và kiểm tra token được gửi kèm theo mỗi yêu cầu. Nó sẽ kiểm tra tính hợp lệ của token bằng cách kiểm tra chữ ký số hoặc giải mã nó bằng cách sử dụng cùng thuật toán mà nó đã sử dụng để tạo ra token ban đầu. Nếu token hợp lệ, máy chủ sẽ tiếp tục xử lý yêu cầu của người dùng.

Quá trình này cho phép máy chủ xác thực người dùng một cách an toàn mà không cần lưu trữ mật khẩu của họ trực tiếp. Token được sử dụng để đại diện cho quyền truy cập được cấp cho người dùng và có thể được sử dụng trong một khoảng thời gian nhất định trước khi hết hạn.

OAuth và Oauth2.0 là các giao thức phổ biến trong ứng dụng web để thực hiện quyền ủy quyền và quyền truy cập một cách an toàn. Chúng cung cấp một phương pháp chuẩn để cấp quyền truy cập vào tài nguyên được bảo vệ thay mặt người dùng mà không cần chia sẻ thông tin xác thực nhạy cảm của họ.OAuth và OAuth 2.0 được thiết kế để giải quyết những thách thức của việc cho phép ứng dụng của bên thứ ba truy cập vào tài nguyên được bảo vệ thay mặt người dùng mà không cần chia sẻ thông tin xác thực nhạy cảm của họ. Những giao thức này cho phép người dùng cấp quyền truy cập giới hạn cho ứng dụng của bên thứ ba, nâng cao tính bảo mật và quyền riêng tư.

OAuth Oauth(Oauth 2.0) và token-based authentication là hai phương thức xác thực phổ biến được sử dụng trong HTTP programming để bảo mật API và các dịch vụ web. Việc tích hợp các phương thức này mang lại nhiều lợi ích cho các nhà phát triển và doanh nghiệp:

* Bảo Mật Tăng Cường:OAuth và token-based authentication cung cấp một cơ chế bảo mật mạnh mẽ hơn so với các phương thức xác thực truyền thống như Basic Authentication. Thông tin đăng nhập được giữ an toàn và không được truyền qua mạng dưới dạng văn bản thuần túy.
* Phân Quyền Linh Hoạt:Cơ chế xác thực token cung cấp phương thức phân quyền linh hoạt. Mỗi token thường được đặc tả với các quyền cụ thể, quyết định những tài nguyên nào có thể truy cập và những hành động nào có thể thực hiện.
* Khả Năng Tương Tác An Toàn:OAuth cho phép ứng dụng yêu cầu quyền truy cập vào tài nguyên từ phía người dùng mà không cần chia sẻ mật khẩu. Điều này giúp giảm rủi ro an ninh và tạo ra một quy trình xác thực an toàn.
* Quản Lý Phiên Bản:Token có thể được sử dụng để quản lý phiên bản và thời gian hiệu lực. Điều này giúp kiểm soát quyền truy cập và đảm bảo rằng thông tin xác thực không được sử dụng sau khi hết hạn.
* Ứng Dụng Thứ Ba:OAuth cho phép ứng dụng thứ ba yêu cầu quyền truy cập vào tài nguyên mà không cần chia sẻ thông tin đăng nhập. Điều này giúp người dùng giữ quyền kiểm soát đối với dữ liệu của họ và giảm rủi ro về an ninh.
* Tích Hợp Dịch vụ Bên Ngoài:Token-based authentication hỗ trợ tích hợp với các dịch vụ bên ngoài thông qua các APIs. Các ứng dụng có thể tương tác một cách an toàn và đáng tin cậy với các dịch vụ khác.
* Dễ Dàng Đăng Nhập:Đối với người dùng, quá trình đăng nhập và cấp quyền truy cập thường trở nên dễ dàng hơn khi sử dụng OAuth. Thông qua quy trình xác thực này, họ không cần phải nhập mật khẩu trực tiếp vào ứng dụng thứ ba.
* Thực Hiện Đơn Giản:Sử dụng OAuth và token-based authentication giúp giảm độ phức tạp của việc quản lý xác thực và phân quyền. Các thư viện và framework thường hỗ trợ việc thực hiện và duy trì quá trình này.

Ví dụ về lợi ích của việc tích hợp OAuth hoặc token-based authentication trong HTTP programming:

* Sử dụng OAuth để cho phép người dùng đăng nhập vào ứng dụng của bạn bằng tài khoản Google hoặc Facebook của họ.
* Sử dụng token-based authentication để truy cập vào API của một dịch vụ web bên thứ ba.

# TRIỂN KHAI VÀ THỰC NGHIỆM

## Kịch Bản 1

### Mô Hình Thực Nghiệm

Mô hình thực nghiệm 1

Phía Server (Django): Xây dựng các API bằng Django sử dụng cơ chế xử lý request/response của Django. Định nghĩa các đường dẫn URL trong Django URLconf để định tuyến các yêu cầu từ client đến các view xử lý.

Phía Client (Postman): Sử dụng Postman để tạo và gửi các yêu cầu HTTP đến các API của Django. Thực hiện các yêu cầu GET, POST, PUT, DELETE hoặc các loại yêu cầu HTTP khác để tương tác với dữ liệu trên máy chủ Django. Kiểm tra và xem phản hồi từ máy chủ Django để đảm bảo rằng các yêu cầu được xử lý đúng cách và dữ liệu trả về là chính xác.

### Mô Tả Kịch Bản

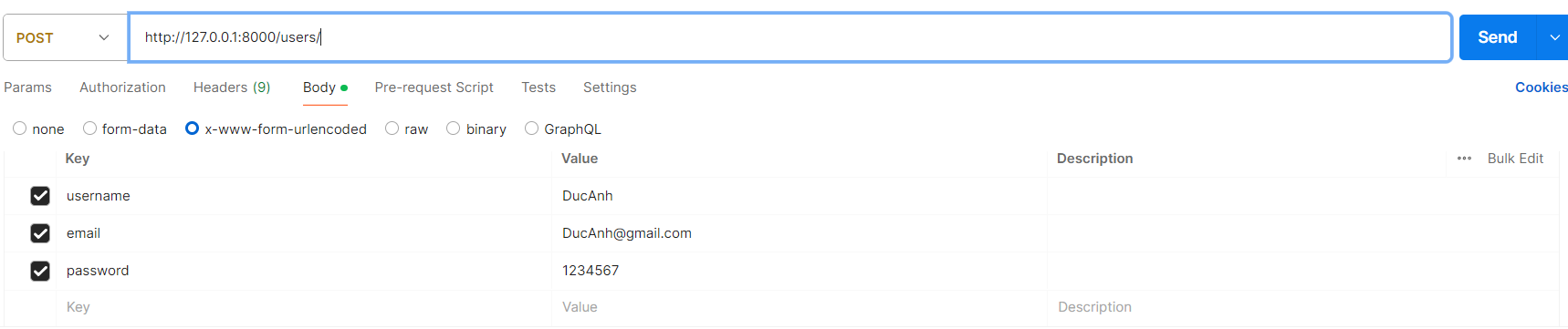
Sử dụng Postman để thực hiện các yêu cầu Create-Read-Update-Delete như:

* POST để tạo mới một user trên máy chủ Django.
* GET để lấy thông tin user từ máy chủ Django.
* PUT để cập nhật một thông tin user đã tồn tại trên máy chủ Django.
* DELETE để xóa một user trên máy chủ Django.

### Thực Hiện

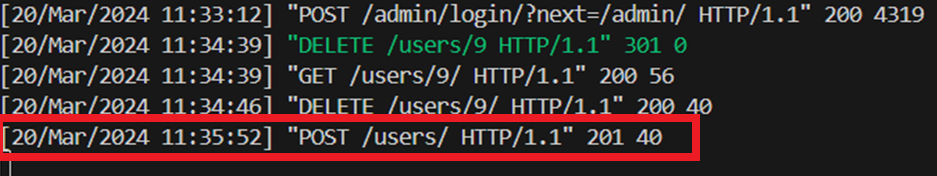
#### Sử dụng POST để tạo mới một user.

Sử dụng phương thức POST trong hàm xử lý(Python)

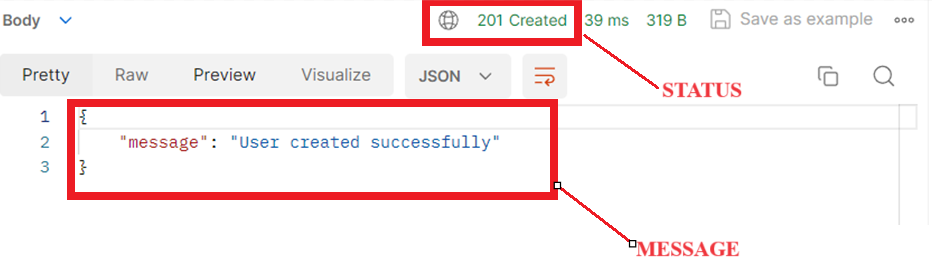
Đây là đoạn code xử lý phương thức POST khi một client gửi request đến server cho phép tạo mới một một người dùng trên máy chủ Django.

Sử dụng phương thức POST gửi Request từ Client

Tiến hành gửi request từ client sử dụng phương thức POST với các tham số username, password, email được truyền dưới dạng ‘x-www-form-urlencoded’ tới URL “<http://127.0.0.1:8000/users/>” thực hiện view “user\_list” để tạo mới một user mớitrên cơ sở dữ liệu của máy chủ Django.



Thông báo của server

Đây là thông báo của server khi nhận được và xử lý một yêu cầu từ client(postman). Sau khi xử lý yêu cầu, server sẽ trả về mã HTTP 201(Tức là yêu cầu đã được xử lý thành công )

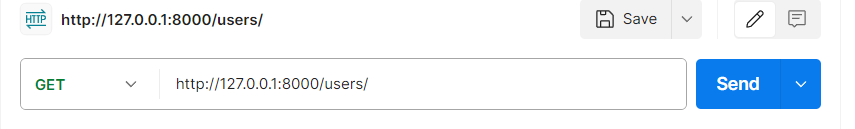
Server gửi phản hồi cho máy client

Và sau khi server xử lý xong và đã gửi lại phản hồi cho máy client. Phản hồi tới máy client bao gồm message, status. Trong trường hợp thành công , phản hồi của client POSTMAN sẽ hiển thị như Hình ….

#### Sử dụng GET để lấy thông tin user.

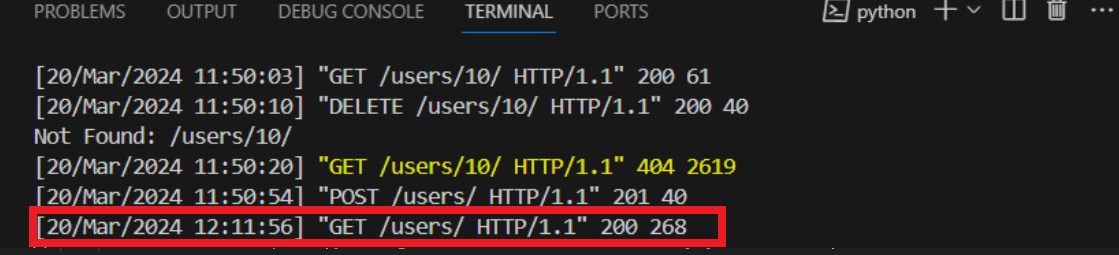
Sử dụng phương thức GET trong hàm xử lý(Python)

Hình trên là đoạn code xử lý phương thức GET khi một client gửi request đến server cho phép lấy ra tất cả các thông tin danh sách user trong cơ sở dữ liệu máy chủ Django

Thực hiện gửi request GET từ client sang server để lấy dữ liệu

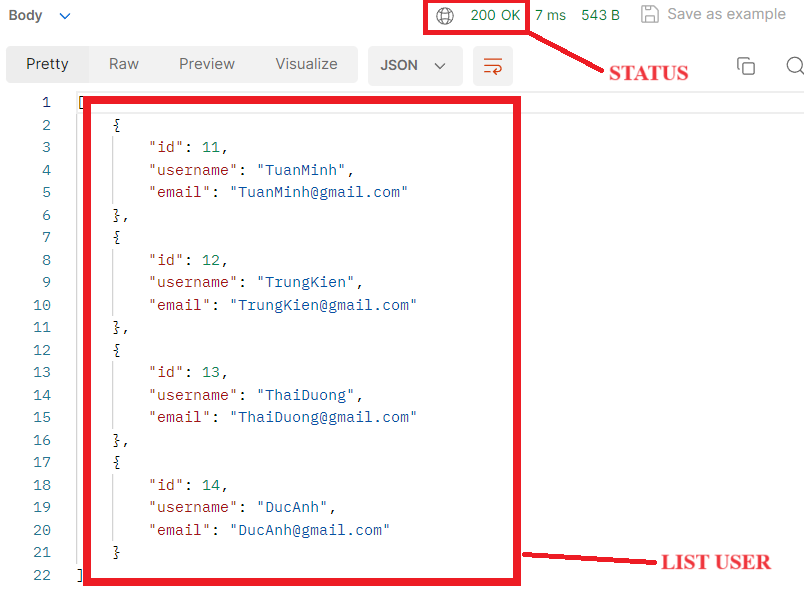
Sử dụng phương thức GET gửi Request từ client

Ở Client, ta gửi request sử dụng phương thức GET tới URLS “<http://127.0.0.1:8000/users/>”. Khi server nhận được nó sẽ thực hiện phần hàm xử lý GET trong view “user\_list” .Và server sẽ hiển thị thông báo như hình sau:



Thông báo của server

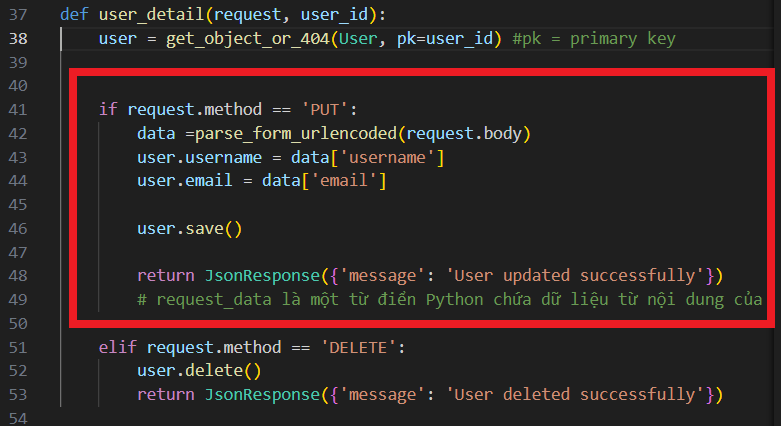
Thông báo này cho ta biết status 200 nghĩa là server đã xử lý thành công.Và tiếp tục server sẽ gửi phản hồi về cho client bao gồm status và message dưới dạng JSON



Server gửi phản hồi cho máy client

Ở client(POSTMAN) sẽ nhận reponse và sẽ hiển thị như hình trên: status 200 có nghĩa là thực hiện thành công , message là list danh sách thông tin user được thực hiện dưới dạng JSON.

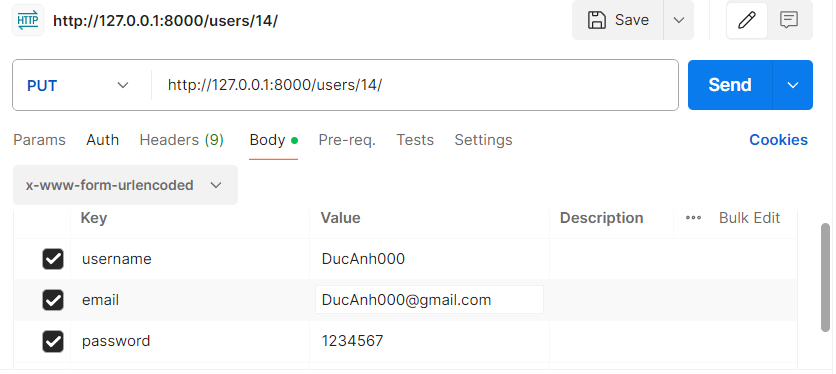
#### Sử dụng PUT để cập nhật một thông tin user đã tồn tại.



Sử dụng phương thức PUT trong hàm xử lý(Python)

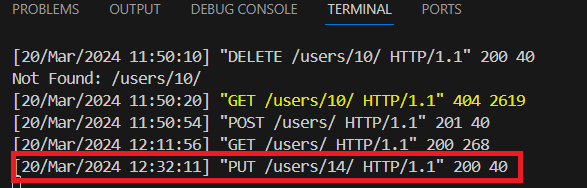
Đây là đoạn code xử lý phương thức POST khi một client gửi request đến server Django cho phép cập nhật một một người dùng trên máy chủ Django. Dòng code 38 là kiểm tra user có tồn tại trên cơ sở dữ liệu của server dựa trên id của user. Nếu user tồn tại thì server sẽ trả về đối tượng đó , còn nếu không tồn tại thì server sẽ trả về lỗi ‘404’. Dòng 42 là sử dụng hàm ‘prase\_form\_urlencoded’ để chuyển đổi dữ liệu trong body của request sang dạng dict. Từ dòng 43->46 ,những dòng code này sẽ thực hiện cập nhật thông tin mới vào user cụ thể trong cơ sở dữ liệu của server

Quá trình thực hiện cập nhật thông tin sử dụng phương thức PUT:



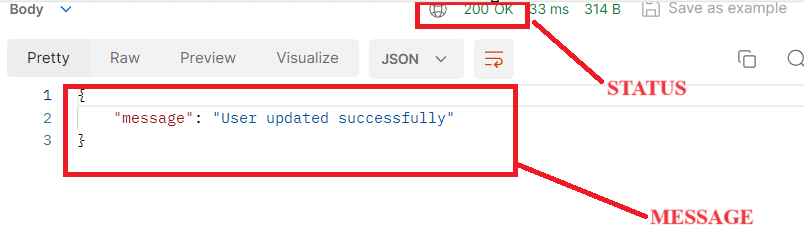
Sử dụng phương thức PUT gửi Request cập nhật thông tin user

Ở máy client gửi request có body chứa các thông tin muốn cập nhật đến URL “/users/id/” (với id ở đây là id của user muốn cập nhật, kiểu dữ liệu int).Trong trường hợp này, ở client POSTMAN thực hiện cập nhật thông tin user có id =14 .Thông tin cập nhật bao gồm : username : DucAnh000 và emai: DucAnh000@gmail.com.



Thông báo của server

Sau khi server nhận được và đã xử lý request từ POSTMAN , server sẽ thông báo dòng thông báo như hình trên .Thông báo cho thấy status là 200 có nghĩa là server đã xử lý thành công. Và tiếp theo server sẽ gửi phản hồi lại cho client

Ở máy Client sẽ nhận được phản hồi và hiển thị như hình dưới:

Server phản hồi cho client đã cập nhật thành công

Status 200 : Cho thấy server đã xử lý request thành công

Và message : Cho thấy user đã được cập nhật thành công

Tiếp theo , để kiểm tra kết quả chính xác có thể sử dụng GET để lấy ra danh sách user (như phần 3.3.2) và sẽ nhận được danh sách các user hiển thị như hình dưới:

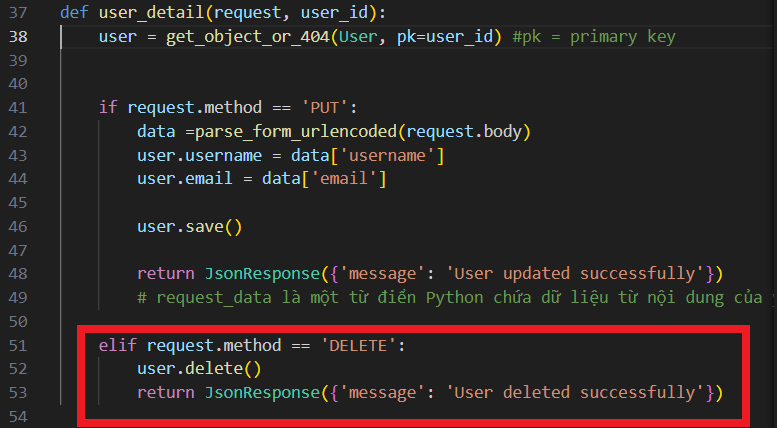
Thông báo user trước khi cập nhật



Thông tin user sau khi cập nhật bằng phương thức PUT

Có thể thấy, user có id ‘14’ đã được chuyển thành ‘DucAnh000’ và có email ‘DucAnh000@gmail.com’

#### Sử dụng DELETE để xóa một user.

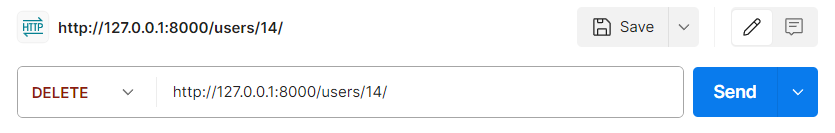


Sử dụng phương thức DELETE trong hàm xử lý(Python)

Đây là đoạn code xử lý phương thức DELETE khi một client gửi request đến server Django cho phép xóa một một người dùng tồn tại trên cơ sở dữ liệu của máy chủ Django.

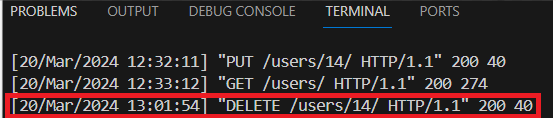
Quá trình xóa thông tin user được thực hiện như sau:

Đầu tiên, ở client(POSTMAN) gửi một yêu cầu sử dụng phương thức ‘DELETE’ tới URL “/users/id/” (với id ở đây là id của user muốn cập nhật, kiểu dữ liệu int) của server.Trong trường hợp này , client sẽ gửi request xóa thông tin user có id 14:



Sử dụng phương thức DELETE gửi Request từ client

Sau khi nhận được yêu cầu, server sẽ xử lý và hiển thị thông báo:



Thông báo của server

Thông báo này cho thấy server đã xử lý thành công yêu cầu từ client. Và server sẽ phản hồi cho client biết. PostMan sẽ nhận được phản hồi và hiển thị như hình dưới:



Server phản hồi cho client đã xóa thành công

Để kiểm tra kết quả chính xác có thể sử dụng GET để lấy ra danh sách user (như phần 3.3.2) và sẽ nhận được danh sách các user hiển thị như hình dưới:



Danh sách user sau khi thực hiện phương thức DELETE

Như vậy , có thể thấy thông tin của user có id là 14 đã bị xóa khỏi cơ sở dữ liệu của máy chủ.

### Kết luận

Mô hình CRUD (Create, Read, Update, Delete) là một mô hình quản lý dữ liệu cơ bản trong phát triển phần mềm, trong đó bao gồm các hoạt động cơ bản như tạo mới, đọc, cập nhật và xóa dữ liệu. Các phương thức HTTP như POST, GET, PUT và DELETE kết hợp trong mô hình CRUD cung cấp một cơ chế linh hoạt và mạnh mẽ để quản lý và tương tác với dữ liệu trong hệ thống. Sự kết hợp của chúng cho phép thực hiện các hoạt động CRUD một cách hiệu quả và linh hoạt, đáp ứng nhu cầu của ứng dụng phần mềm một cách đầy đủ và toàn diện.

## Kịch Bản 2

### Mô Hình Thực Nghiệm



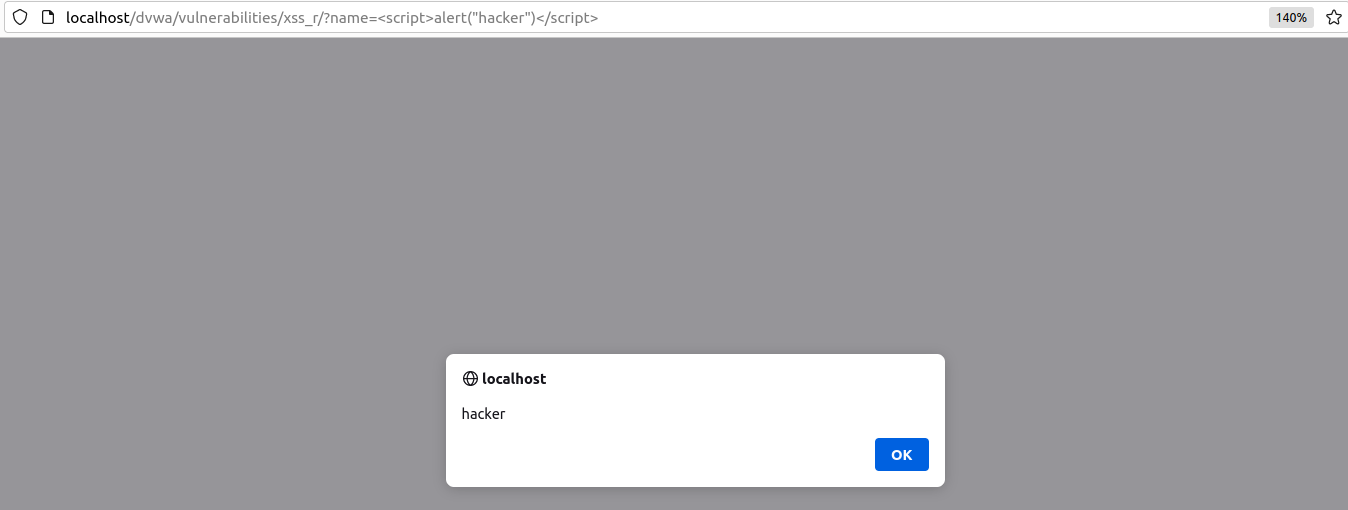
Mô hình thực nghiệm 2

Mô hình xây dựng trên Damn Vulnerable Web Application (DVWA) là một ứng dụng mã nguồn PHP/MySQL tập hợp sẵn các lỗi logic về bảo mật ứng dụng web trong mã nguồn PHP. Lỗi logic khi lập trình có thể áp dụng đối với các loại ngôn ngữ lập trình nhằm giảm thiểu khả năng tạo ra lổ hổng bảo mật từ tư duy lập trình chưa cẩn thận. Mục tiêu chính của DVWA đó là tạo ra một môi trường thực hành hợp pháp. Giúp cho các nhà phát triển ứng dụng web hiểu hơn về hoạt động lập trình an toàn và bảo mật hơn. Bên cạnh đó DVWA cũng cung cấp cho các pentester phương pháp học và thực hành tấn công khai thác lỗi bảo mật ứng dụng web ở mức cơ bản và nâng cao.

### Mô Tả Kịch Bản

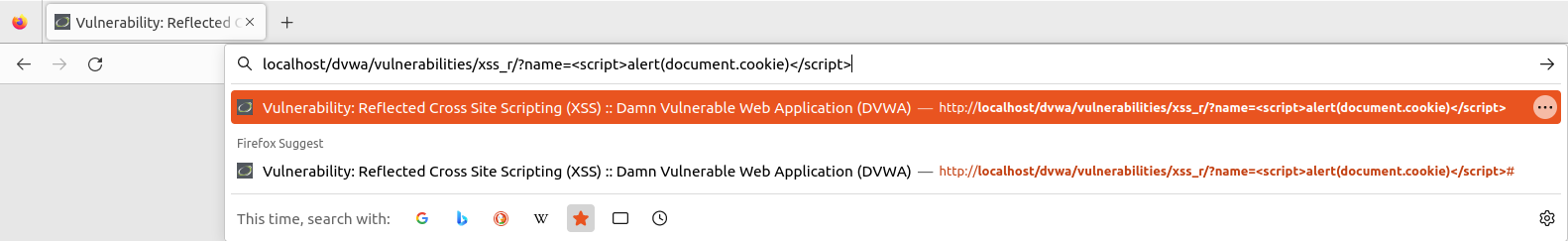
Chèn mã JavaScript vào các tham số URL để lấy cookie và chiếm phiên của người dùng.

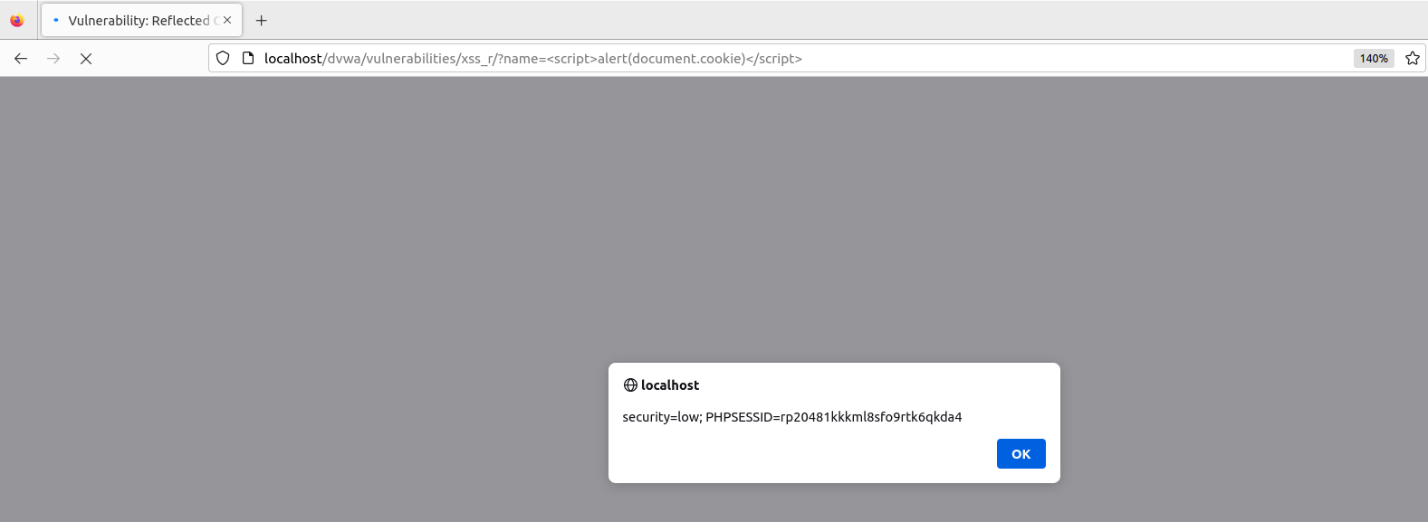
### Không có mô tả.Thực Hiện

Thực hiện chèn một đoạn mã JavaScript(như trên) vào thanh URL

Cửa sổ pop-up

Một cửa số pop-up chứa thông báo hacker đã hiện lên để thông báo rằng đoạn mã đã được thực thi trên trang web.

Đoạn mã JavaScript trong URL

Chèn một đoạn mã JavaScript vào thanh URL để lấy cookie từ người dùng. 

Thông báo sau khi chèn mã JavaScript

Sau khi chèn, cửa sổ pop-up đã hiện lên một thông tin PHPSESSID, cụ thể trong trường hợp này là chuỗi “rp020481kkkml8sf09rtk6qkda4”.Chuỗi này là một ID phiên duy nhất được tạo ra. Như vậy kẻ tấn công khi lấy được PHPSESSID của người dùng, họ có thể thực hiện hàng loạt các cuộc tấn công như đăng nhập giả mạo (session hijacking), xem và thay đổi thông tin phiên, tấn công XSS và CSRF.

### Kết Luận

Kịch bản này đã mô tả cuộc tấn công nhỏ sử dụng HTTP thông qua URL. Các cuộc tấn công sử dụng HTTP thông qua URL vẫn là một mối đe dọa đáng chú ý trong bối cảnh bảo mật hiện đại. Nó có thể gây ra rất nhiều nguy hiểm như đánh cắp thông tin cá nhân, chiếm quyền truy cập vào tải khoản, đánh cắp cookie phiên, tấn công phân phối mã độc,… Để tránh những cuộc tấn công cần áp dụng một số biện pháp như: Sử dụng HTTPs, kiểm tra và sàng lọc dữ liệu đầu vào, chặn các yêu cầu HTTP không hợp lệ,..

KẾT LUẬN

Tổng hợp lại, trong báo cáo đã tìm hiểu về HTTP Progamming, bao gồm thông tin về giao thức HTTP, các thành phần cơ bản của HTTP Progamming, các vấn đề bảo mật,cơ chế hoạt động ,và các ứng dụng của nó.Và đã xây dựng và triển khai mô hình CRUD sử dụng các phương thức HTTP nhằm giúp hiểu rõ hơn về cách hoạt động và sự lợi ích của nó trong vấn đề quản lý dữ liệu trong hệ thống. Việc hiểu về cách xử lý dữ liệu HTTP là rất quan trọng trong phát triển ứng dụng web và mobile hiện đại.

TÀI LIỆU THAM KHẢO

[1.]<https://www.datacamp.com/tutorial/makinghttprequestsinpython?fbclid=IwAR3yleU07XMDIWJD7E6nC8EhpWxmsi91QSl1Et3WpicOjLpquHjOVSZwc>

[2.]<https://cs.lmu.edu/~ray/notes/httpprogramming/?fbclid=IwAR1UgVZ4mAGpn--KaTlIIOWqCVRO6IQYiGKqfbTNVffVkCH7aNzEkwAUtzY>

[3.]<https://developer.mozilla.org/enUS/docs/Web/HTTP/Overview?fbclid=IwAR0lPPkc4L9EvH1GspSCCnDYtLZHaSBgOYD1RVmN3TZsYUboVsRXVwjQO0>

[4.] <https://devdocs.io/http-guides/>

[5.] "RESTful Web Services Cookbook" by Subbu Allamaraju

[6.] "Python Requests Essentials" by Rakesh Vidya Chandra

PHỤ LỤC

### Công cụ phát triển và Môi trường phát triển

#### A, Visual Studio Code

Visual Studio Code (hay còn gọi tắt là VS Code) là một trình biên tập mã nguồn mở và miễn phí, được phát triển bởi Microsoft. Nó là một ứng dụng đa nền tảng, có thể chạy trên Windows, macOS và Linux. VS Code được thiết kế để hỗ trợ nhiều ngôn ngữ lập trình khác nhau và có nhiều tính năng hữu ích như gợi ý mã, đánh dấu cú pháp, kiểm tra lỗi và hỗ trợ Git. VS Code là một trong những trình biên tập mã nguồn phổ biến nhất hiện nay và được sử dụng rộng rãi trong cộng đồng lập trình viên.

#### B, POSTMAN

Postman hiện là một trong những công cụ phổ biến nhất được sử dụng trong thử nghiệm các API. Với Postman, ta có thể gọi Rest API mà không cần viết dòng code nào. Postman còn có khả năng hỗ trợ tất cả các phương thức HTTP (GET,PUT,PATCH,DELETE). Từ đó có thể tạo và gửi các request HTTP như GET, POST, PUT, DELETE và nhận response từ máy chủ.

#### C, Django

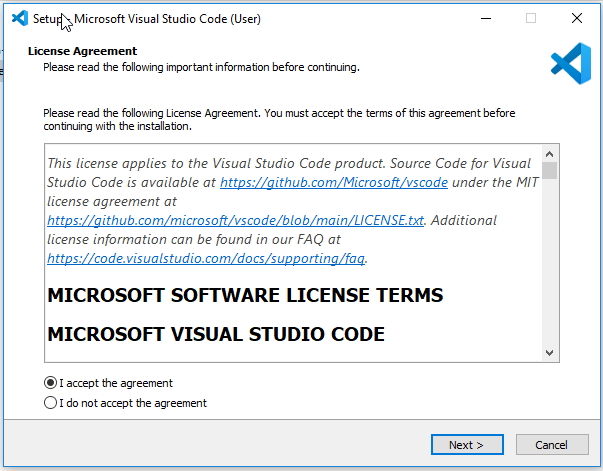
Django là một framework phát triển web mạnh mẽ của Python và nó cung cấp một cách tiện lợi để tạo ra các ứng dụng web và quản lý các request từ clients. Django có khả năng tạo và quản lý một máy chủ web. Với các tính năng và công cụ như Routing (Định tuyến), Views (Các chế độ hiển thị), Middleware (Phần mềm trung gian), Forms (Biểu mẫu), và Authentication (Xác thực), Django còn có khả năng cung cấp một cách tiện lợi để xử lý và đáp ứng các yêu cầu từ phía người dùng. Framework này cho phép phát triển ứng dụng web phức tạp mà không cần phải viết lại các phần cơ bản của hệ thống, giúp tăng tốc độ phát triển và đảm bảo tính bảo mật và hiệu suất của ứng dụng.

## Cấu Hình Môi Trường Phát Triển

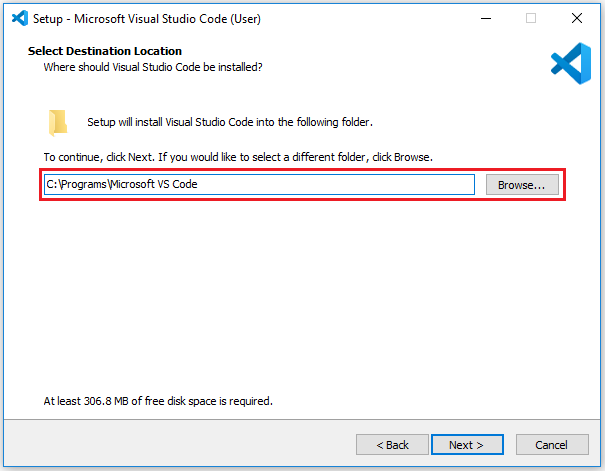
#### A, Visual Studio Code

Visual Studio Code có thể download ở website chính thức của Visual Studio Code với link bên dưới.

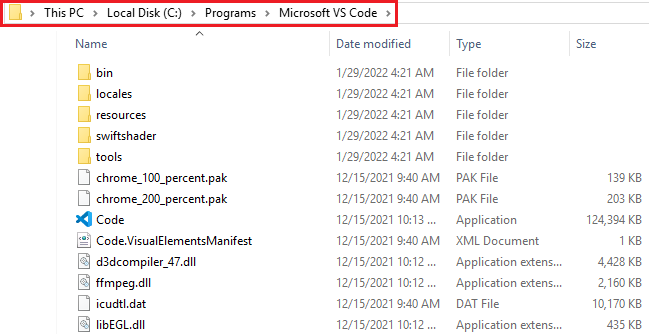
<https://code.visualstudio.com/download>

[](https://gochocit.com/wp-content/uploads/2022/01/install-visual-studio-code.png)

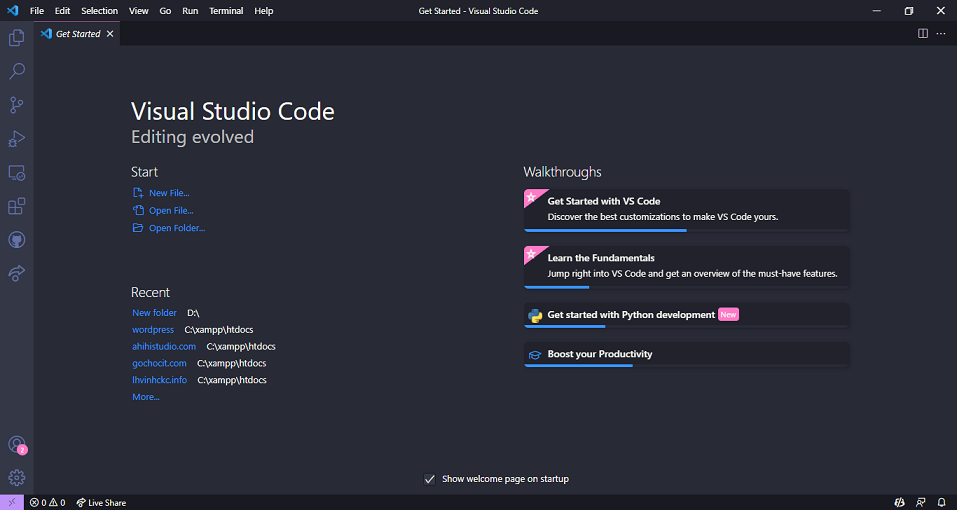
Nhấn Next rồi chọn đường dẫn cài đặt Visual Studio Code.

[](https://gochocit.com/wp-content/uploads/2022/01/custom-folder-install-visual-studio-code.png)

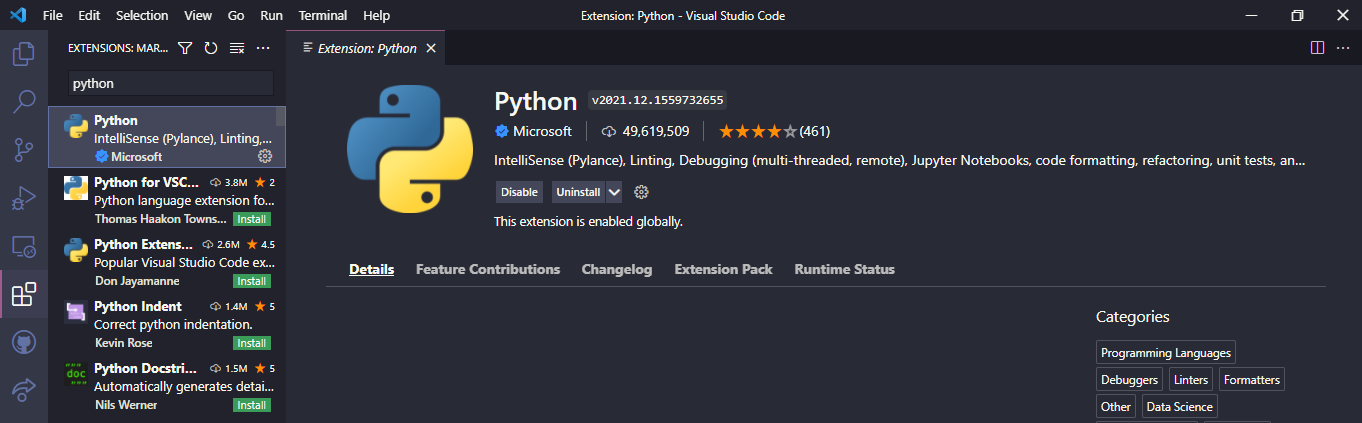
Sau khi cài đặt thành công, các bạn sẽ thấy folder cài đặt Visual Studio Code.

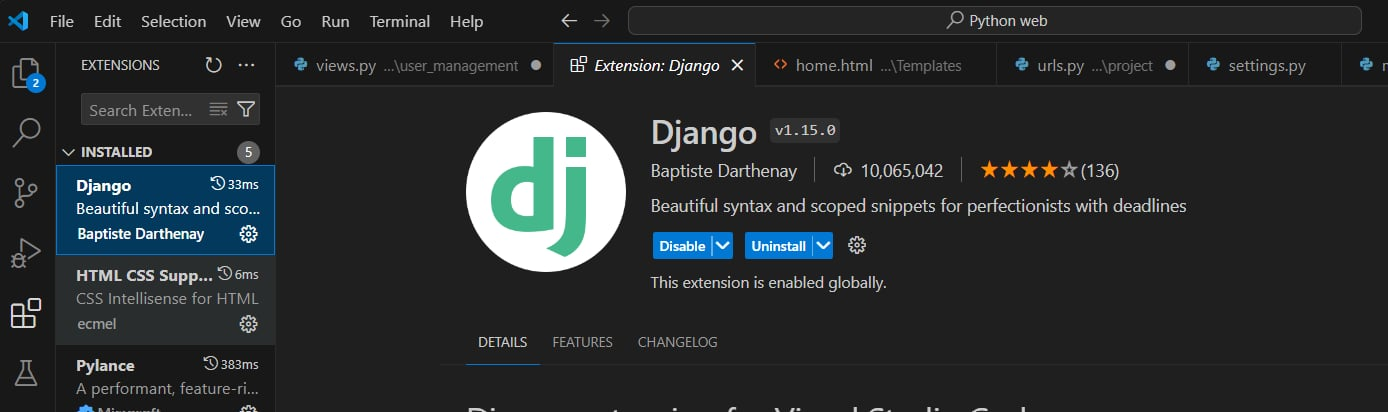
[](https://gochocit.com/wp-content/uploads/2022/01/folder-installed-visual-studio-code.png)

Bên dưới là giao diện khi mới bắt đầu “**Get Started**” của Visual Studio Code

[](https://gochocit.com/wp-content/uploads/2022/01/interface-visual-studio-code.png)

Cài đặt extension **Python** trong Visual Studio Code, các bạn tìm đến phần **Extensions** hoặc nhấn tổ hợp phím **Ctrl+Shift+X** rồi gõ **python** và tiến hành cài đặt extension **Python**.

[](https://gochocit.com/wp-content/uploads/2022/01/install-extension-python-visual-studio-code.png)

Cài đặt extension **Django** trong Visual Studio Code, các bạn tìm đến phần **Extensions** hoặc nhấn tổ hợp phím **Ctrl+Shift+X** rồi gõ **django** và tiến hành cài đặt extension **Django**.

#### POSTMAN

Postman hỗ trợ rất nhiều phiên bản hệ điều hành khác nhau như: trình duyệt, Mac OS, Windows, Linux.

Để cài đặt, bạn chỉ cần:

* **Bước 1:** truy cập vào https://www.postman.com/downloads/.
* **Bước 2:**chọn phiên bản phù hợp với thiết bị/ hệ điều hành của bạn phiên bản **32bit hoặc 64bit.**
* **Bước 3**: download về và cài đặt phần mềm Postman.

Nếu không muốn máy tính của mình quá năng, bạn có thể sử dụng phiên bản web của Postman. Bạn sẽ chỉ cần:

* Tạo download khoản và đăng nhập vào Postman
* Download và cài đặt Postman Desktop Agent

Postman sẽ tự động làm công việc hỗ cho bạn và gửi API lên phiên bản Postman Web.

Trong trường hợp bạn thích trải nghiệm tính năng mới của Postman, bạn có thể download phiên bản Postman Canary và sử dụng. Mặc dù bạn sẽ được trải nghiệm những tính năng mới nhất của phần mềm Postman.

C, Xây dựng server

* **Bước 1:** Truy cập đường dẫn [Download Python | Python.org](https://www.python.org/downloads/) để tiến hành tải môi trường.
* **Bước 2:** Sau khi tải về xong, tiến hành cài đặt môi trường.
* **Bước 3:** Tạo mới một folder chứa project, mở Terminal, nhập lệnh ví dụ dưới đây.

mkdir python

cd python

* **Bước 4:** Cài đặt Django.

pip install django

* **Bước 5:** Tạo mới một project django.

Django**-**admin startproject project

* **Bước 6:** Run server

cd myproject

python manage.py runserver

* Bước 7: Tạo một app django

python manage.py startapp user\_management

* **Bước 8:** Sử dụng Browser và truy cập vào địa chỉ.

Text

Description automatically generated

## Bảng phân công công việc

|  |  |
| --- | --- |
| **Họ và tên - MSV** | **Công việc** |
| Bùi Trung Kiên – AT180228 | 1.Tìm hiểu tổng quan giao thức HTTP  2.Triển khai xây dựng ứng dụng thực nghiệm  3.Xây dựng bản powerpoint |
| Nguyễn Thái Dương – AT180612 | 1.Tìm hiểu công cụ và thư viện hỗ trợ lập trình  2.An toàn và bảo mật trong HTTP Programming  3.Triển khai xây dựng ứng dụng thực nghiệm  4.Xây dựng bản words |
| Nguyễn Đức Anh – AT180602 | 1.Các thành phần cơ bản  2.Cách hoạt động cơ bản của HTTP Programming  3.Triển khai xây dựng ứng dụng thực nghiệm  4.Xây dựng bản powerpoint |
| Nguyễn Tuấn Minh – AT180633 | 1.Tìm hiểu về ứng dụng thực tế của HTTP Programming  2.Triển khai xây dựng ứng dụng thực nghiệm  3.Xây dựng bản words |