**TÀI LIỆU LẬP TRÌNH WED 1**

Mục lục :

1. Các khái niệm cơ bản về Internet: Trình duyệt, Domain, Hosting, DNS SERVER...,

2. Cách hoạt động của một trang Web và mối quan hệ giữa các thành phần + ví dụ

3. Khái niệm backend, frontend + ngôn ngữ + framework

4. Quy trình phát triển ứng dụng Web

5. Giao thức HTTP và HTTPS, các bản tin Http Request và Http Response

6. Các mã trạng thái phản hồi của HTTP Response

7. Phân biệt các phương thức (method) của HTTP Request

8. Công cụ GIT + cách sử dụng GITHUB

Nội dung :

**I. Các khái niệm cơ bản về Internet: Trình duyệt, Domain, Hosting, DNS SERVER...,**

### 1. ****Trình duyệt (Web Browser)****

Trình duyệt là phần mềm giúp người dùng truy cập và hiển thị các trang web trên Internet. Nó hoạt động bằng cách gửi yêu cầu đến máy chủ web thông qua giao thức HTTP hoặc HTTPS, sau đó nhận và hiển thị nội dung của trang web.

**Ví dụ:** Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.

### 2. ****Tên miền (Domain Name)****

Tên miền là địa chỉ dễ nhớ thay thế cho địa chỉ IP của một trang web, giúp người dùng truy cập dễ dàng hơn.

**Ví dụ:**

google.com là tên miền của Google.

facebook.com là tên miền của Facebook.

Tên miền có nhiều phần mở rộng như:

.com (thương mại)

.edu (giáo dục)

.gov (chính phủ)

.vn (Việt Nam)

### 3. ****Hosting (Web Hosting)****

Hosting là dịch vụ lưu trữ dữ liệu và chạy website trên máy chủ. Khi một website được thiết kế, nó cần có nơi lưu trữ các tệp tin (HTML, CSS, hình ảnh, video, cơ sở dữ liệu...). Hosting giúp trang web có thể hoạt động 24/7 trên Internet.

**Các loại hosting phổ biến:**

**Shared Hosting:** Nhiều website dùng chung một máy chủ.

**VPS (Virtual Private Server):** Máy chủ ảo riêng biệt, có tài nguyên riêng.

**Dedicated Server:** Máy chủ vật lý riêng, mạnh mẽ hơn VPS.

**Cloud Hosting:** Lưu trữ trên nhiều máy chủ, đảm bảo ổn định cao.

### 4. ****DNS Server (Domain Name System Server)****

DNS Server là hệ thống phân giải tên miền thành địa chỉ IP tương ứng. Khi bạn nhập một địa chỉ web vào trình duyệt, DNS sẽ tìm địa chỉ IP của website đó để kết nối.

**Ví dụ:** Khi bạn nhập google.com, DNS sẽ dịch nó thành địa chỉ IP (ví dụ: 142.250.182.14) để trình duyệt truy cập vào máy chủ của Google.

Một số DNS phổ biến:

**Google DNS:** 8.8.8.8 và 8.8.4.4

**Cloudflare DNS:** 1.1.1.1

**OpenDNS:** 208.67.222.222

### 5. ****Giao thức HTTP và HTTPS****

**HTTP (HyperText Transfer Protocol):** Giao thức truyền tải dữ liệu giữa máy chủ và trình duyệt, nhưng không mã hóa thông tin, dễ bị tấn công.

**HTTPS (HyperText Transfer Protocol Secure):** Phiên bản bảo mật của HTTP, sử dụng chứng chỉ SSL/TLS để mã hóa dữ liệu, giúp bảo vệ thông tin của người dùng.

**Ví dụ:**

http://example.com → Không bảo mật.

https://example.com → Được mã hóa an toàn.

### 6. ****IP Address (Địa chỉ IP)****

Địa chỉ IP là số nhận dạng duy nhất của một thiết bị trên mạng. Có hai loại IP:

**IPv4:** Định dạng 32-bit, ví dụ: 192.168.1.1

**IPv6:** Định dạng 128-bit, ví dụ: 2001:db8::ff00:42:8329

### 7. ****Máy chủ Web (Web Server)****

Web Server là máy chủ chịu trách nhiệm lưu trữ và cung cấp nội dung web. Khi bạn truy cập một website, trình duyệt gửi yêu cầu đến máy chủ web để lấy dữ liệu và hiển thị.

**Một số Web Server phổ biến:**

**Apache**

**Nginx**

**Microsoft IIS**

### 8. ****ISP (Internet Service Provider) – Nhà cung cấp dịch vụ Internet****

ISP là công ty cung cấp kết nối Internet cho người dùng.

**Ví dụ các ISP tại Việt Nam:**

Viettel

VNPT

FPT Telecom

**II. Cách hoạt động của một trang Web và mối quan hệ giữa các thành phần + ví dụ**

### ****1. Quá Trình Hoạt Động Của Một Trang Web****

Khi người dùng nhập một địa chỉ web vào trình duyệt (ví dụ: www.example.com), các bước sau diễn ra:

#### ****Bước 1: Người dùng gửi yêu cầu từ trình duyệt****

Bạn nhập www.example.com vào trình duyệt (Chrome, Firefox, Edge...). Trình duyệt gửi yêu cầu đến **DNS Server** để lấy địa chỉ IP của tên miền.

#### ****Bước 2: DNS phân giải tên miền thành địa chỉ IP****

DNS Server tìm địa chỉ IP tương ứng với www.example.com (ví dụ: 203.0.113.45). Trình duyệt sử dụng địa chỉ IP này để gửi yêu cầu đến máy chủ web của trang web.

#### ****Bước 3: Máy chủ web nhận yêu cầu****

Máy chủ web (Web Server) xử lý yêu cầu và lấy dữ liệu từ **Hosting** hoặc **Cơ sở dữ liệu**. Nếu trang web có dữ liệu động (như trang đăng nhập), máy chủ web sẽ chạy mã nguồn (PHP, Python, Node.js...) để lấy thông tin từ **Cơ sở dữ liệu**.

#### ****Bước 4: Máy chủ gửi phản hồi****

Máy chủ web gửi dữ liệu trang web (HTML, CSS, JavaScript, hình ảnh, video...) về trình duyệt.

#### ****Bước 5: Trình duyệt hiển thị trang web****

Trình duyệt nhận dữ liệu, giải mã và hiển thị nội dung trang web cho người dùng.

### ****2. Mối Quan Hệ Giữa Các Thành Phần****

| **Thành Phần** | **Vai Trò** |
| --- | --- |
| **Trình duyệt** | Gửi yêu cầu truy cập trang web và hiển thị nội dung. |
| **Tên miền (Domain)** | Địa chỉ dễ nhớ để truy cập website (vd: google.com). |
| **DNS Server** | Dịch tên miền thành địa chỉ IP của máy chủ web. |
| **Hosting** | Nơi lưu trữ mã nguồn, hình ảnh, cơ sở dữ liệu của trang web. |
| **Máy chủ Web (Web Server)** | Xử lý yêu cầu từ trình duyệt, cung cấp dữ liệu trang web. |
| **Cơ sở dữ liệu (Database Server)** | Lưu trữ thông tin người dùng, bài viết, sản phẩm... |
| **Giao thức HTTP/HTTPS** | Truyền dữ liệu giữa trình duyệt và máy chủ web. |

Ví dụ về mối quan hệ:

Khi bạn truy cập facebook.com, trình duyệt gửi yêu cầu đến DNS để lấy địa chỉ IP.

Sau đó, máy chủ web của Facebook nhận yêu cầu, lấy dữ liệu từ cơ sở dữ liệu (ví dụ: thông tin bài đăng, danh sách bạn bè) rồi gửi về trình duyệt.

Trình duyệt hiển thị giao diện Facebook mà bạn thấy.

### ****3. Ví Dụ Minh Họa Cụ Thể****

Ví dụ về quá trình tải một trang web (Google Search)

1. Bạn nhập www.google.com vào trình duyệt.
2. Trình duyệt hỏi **DNS Server** để lấy địa chỉ IP của Google.
3. **DNS Server** trả về IP 142.250.182.14 (ví dụ).
4. Trình duyệt gửi yêu cầu đến **Máy chủ web của Google**.
5. Máy chủ web của Google phản hồi bằng trang tìm kiếm Google.
6. Trình duyệt hiển thị giao diện Google Search cho bạn.

Ví dụ về trang web động (Facebook)

1. Bạn nhập www.facebook.com vào trình duyệt.
2. Máy chủ web của Facebook kiểm tra bạn có đăng nhập hay chưa.
3. Nếu chưa đăng nhập, nó hiển thị trang đăng nhập. Nếu đã đăng nhập, nó 4. truy vấn **Cơ sở dữ liệu** để lấy danh sách bạn bè, bài đăng... 5.Máy chủ gửi dữ liệu về trình duyệt để hiển thị timeline Facebook.

### ****4. Kết Luận****

Một trang web hoạt động thông qua sự phối hợp giữa **trình duyệt, DNS, hosting, máy chủ web, cơ sở dữ liệu** và **giao thức truyền tải dữ liệu**. Mỗi thành phần có vai trò riêng, đảm bảo quá trình truy cập web diễn ra nhanh chóng và ổn định. Trang web có thể là **tĩnh** (chỉ hiển thị nội dung cố định) hoặc **động** (có cơ sở dữ liệu và tương tác với người dùng).

**III.Khái niệm backend, frontend + ngôn ngữ + framework**

### ****1. Frontend Là Gì?****

Frontend (giao diện người dùng) là phần hiển thị của một trang web mà người dùng có thể thấy và tương tác trực tiếp, bao gồm các thành phần như bố cục, hình ảnh, biểu mẫu và nút bấm.

#### ****Các Ngôn Ngữ Lập Trình Frontend****

**HTML (HyperText Markup Language):** Định dạng và tổ chức nội dung trên trang web.

**CSS (Cascading Style Sheets):** Thiết kế giao diện, màu sắc, bố cục.

**JavaScript:** Tạo các hiệu ứng động, xử lý tương tác người dùng.

#### ****Các Framework và Thư Viện Frontend Phổ Biến****

**React.js:** Thư viện JavaScript do Facebook phát triển, giúp xây dựng giao diện động.

**Vue.js:** Framework nhẹ, dễ học, phù hợp cho các dự án nhỏ và vừa

**Angular:** Framework do Google phát triển, phù hợp với các ứng dụng lớn.

**Bootstrap:** Framework CSS giúp thiết kế giao diện đẹp và thân thiện với thiết bị di động.

### ****2. Backend Là Gì?****

Backend (phần xử lý phía máy chủ) là nơi lưu trữ và xử lý dữ liệu, điều khiển logic của trang web. Người dùng không thể thấy trực tiếp phần này nhưng nó đóng vai trò quan trọng trong việc đảm bảo trang web hoạt động đúng.

#### ****Các Ngôn Ngữ Lập Trình Backend****

**PHP:** Thường được dùng trong phát triển web, phổ biến với WordPress, Laravel.

**Python:** Được sử dụng với Django, Flask để xây dựng hệ thống web.

**Node.js:** Chạy JavaScript trên server, hỗ trợ các ứng dụng thời gian thực.

**Java:** Phổ biến trong các ứng dụng doanh nghiệp với Spring Boot.

**C# (.NET):** Được dùng nhiều trong các ứng dụng Windows và web với ASP.NET.

#### ****Các Framework và Công Nghệ Backend Phổ Biến****

**Laravel (PHP):** Framework PHP mạnh mẽ, hỗ trợ phát triển nhanh.

**Django (Python):** Framework bảo mật cao, phù hợp với ứng dụng lớn.

**Express.js (Node.js):** Nhẹ, linh hoạt, thường dùng để xây dựng API.

**Spring Boot (Java):** Hỗ trợ phát triển các ứng dụng doanh nghiệp lớn.

**ASP.NET (C#):** Framework của Microsoft, mạnh mẽ trong hệ thống doanh nghiệp.

### ****3. Sự Khác Biệt Giữa Frontend Và Backend****

| **Tiêu chí** | **Frontend** | **Backend** |
| --- | --- | --- |
| Vai trò | Hiển thị giao diện, tương tác với người dùng | Xử lý dữ liệu, điều khiển logic hệ thống |
| Ngôn ngữ | HTML, CSS, JavaScript | PHP, Python, Java, C#, Node.js |
| Framework | React.js, Angular, Vue.js | Laravel, Django, Express.js, Spring Boot |
| Vị trí | Chạy trên trình duyệt của người dùng | Chạy trên máy chủ |

### ****4. Full-Stack Developer Là Gì?****

Full-stack developer là lập trình viên có thể làm cả frontend và backend. Họ hiểu cách hoạt động của giao diện người dùng, cách xử lý dữ liệu trên máy chủ, cũng như cách kết nối hai phần này với nhau.

#### ****Các Công Nghệ Full-Stack Phổ Biến:****

**MEAN Stack (MongoDB, Express.js, Angular, Node.js)**

**MERN Stack (MongoDB, Express.js, React.js, Node.js)**

**LAMP Stack (Linux, Apache, MySQL, PHP)**

### ****5. Kết Luận****

**Frontend** tập trung vào trải nghiệm người dùng, sử dụng HTML, CSS, JavaScript và các framework như React.js, Vue.js, Angular.

**Backend** xử lý dữ liệu, sử dụng PHP, Python, Java, Node.js cùng các framework như Laravel, Django, Express.js.

**Full-stack developer** có thể làm cả frontend và backend, sử dụng nhiều công nghệ khác nhau.

**IV. Quy trình phát triển ứng dụng Web**

### ****1. Xác Định Yêu Cầu (Requirement Analysis)****

Mục tiêu: Hiểu rõ nhu cầu của khách hàng và xác định chức năng của ứng dụng.

Xác định đối tượng sử dụng và mục tiêu của ứng dụng.

Liệt kê các tính năng cần có (ví dụ: đăng nhập, quản lý sản phẩm, thanh toán online).

Lập tài liệu mô tả yêu cầu (SRS – Software Requirement Specification).

**Ví dụ:**  
Nếu phát triển một trang thương mại điện tử, cần có tính năng đăng ký, giỏ hàng, thanh toán, theo dõi đơn hàng.

### ****2. Thiết Kế (Design)****

Mục tiêu: Xây dựng giao diện và kiến trúc hệ thống.

**Thiết kế UI/UX:** Dùng các công cụ như Figma, Adobe XD để vẽ wireframe, mockup.

**Thiết kế hệ thống:** Xác định database, API, giao tiếp giữa frontend và backend.

**Lựa chọn công nghệ:** Ngôn ngữ lập trình (JavaScript, PHP, Python…), framework (React.js, Laravel, Django…).

**Ví dụ:**

UI/UX designer thiết kế giao diện trang chủ, trang sản phẩm.

Backend developer thiết kế database lưu thông tin khách hàng, đơn hàng.

### ****3. Phát Triển (Development)****

Mục tiêu: Viết mã nguồn và xây dựng các tính năng của ứng dụng.

**Frontend Development:** Xây dựng giao diện người dùng bằng HTML, CSS, JavaScript, React.js, Vue.js…

**Backend Development:** Xây dựng logic xử lý dữ liệu bằng PHP, Python, Node.js…

**Database Development:** Thiết kế và triển khai cơ sở dữ liệu bằng MySQL, MongoDB…

**API Development:** Xây dựng API để kết nối frontend và backend.

**Ví dụ:**

Lập trình viên frontend xây dựng trang sản phẩm có hiển thị danh sách hàng hóa.

Backend xử lý đăng nhập, xác thực người dùng, lưu thông tin vào cơ sở dữ liệu.

### ****4. Kiểm Thử (Testing)****

Mục tiêu: Phát hiện và sửa lỗi trước khi triển khai.

**Unit Testing:** Kiểm tra từng phần nhỏ của ứng dụng.

**Integration Testing:** Kiểm tra sự kết nối giữa các module.

**User Acceptance Testing (UAT):** Kiểm tra thực tế với người dùng.

**Performance Testing:** Đánh giá tốc độ và hiệu suất ứng dụng.

**Ví dụ:**

Kiểm tra đăng nhập có hoạt động đúng không.

Kiểm tra giỏ hàng có cập nhật chính xác khi thêm sản phẩm.

### ****5. Triển Khai (Deployment)****

Mục tiêu: Đưa ứng dụng lên môi trường thực tế để người dùng sử dụng.

Mua tên miền và hosting/server.

Cài đặt và cấu hình server (Apache, Nginx, Docker…).

Đưa mã nguồn lên server (Dùng Git, FTP, CI/CD…).

Kiểm tra hoạt động của ứng dụng sau khi triển khai.

**Ví dụ:**

Deploy website lên server AWS, DigitalOcean hoặc Heroku.

Cấu hình HTTPS để bảo mật dữ liệu.

### ****6. Bảo Trì Và Cập Nhật (Maintenance & Updates)****

Mục tiêu: Cải thiện hiệu suất, sửa lỗi và cập nhật tính năng mới.

Giám sát lỗi và sửa chữa khi cần thiết.

Nâng cấp bảo mật để tránh hacker tấn công.

Cập nhật tính năng mới dựa trên phản hồi người dùng.

**Ví dụ:**

Cập nhật giao diện website để tăng trải nghiệm người dùng.

Tối ưu tốc độ tải trang để cải thiện hiệu suất.

### ****Kết Luận****

Phát triển ứng dụng web gồm các bước: **Xác định yêu cầu → Thiết kế → Phát triển → Kiểm thử → Triển khai → Bảo trì.**

Mỗi giai đoạn giúp đảm bảo ứng dụng hoạt động ổn định, bảo mật và đáp ứng nhu cầu người dùng.

Quy trình này có thể áp dụng cho các ứng dụng nhỏ (blog cá nhân) đến hệ thống lớn (thương mại điện tử, mạng xã hội).

**V. Giao thức HTTP và HTTPS, các bản tin Http Request và Http Response**

### ****1. HTTP Là Gì?****

HTTP (**HyperText Transfer Protocol**) là giao thức truyền tải siêu văn bản được sử dụng để trao đổi dữ liệu giữa trình duyệt và máy chủ web.

HTTP hoạt động theo mô hình **client-server**, trong đó trình duyệt (client) gửi yêu cầu và máy chủ web (server) phản hồi.

Dữ liệu truyền tải qua HTTP không được mã hóa, dễ bị tấn công và đánh cắp thông tin.

**Ví dụ:**  
Khi bạn nhập http://www.example.com vào trình duyệt, HTTP sẽ gửi yêu cầu đến máy chủ web để lấy dữ liệu trang web.

### ****2. HTTPS Là Gì?****

HTTPS (**HyperText Transfer Protocol Secure**) là phiên bản bảo mật của HTTP, sử dụng **SSL/TLS** để mã hóa dữ liệu trước khi truyền tải.

HTTPS giúp bảo vệ thông tin người dùng, đặc biệt quan trọng trong các giao dịch tài chính, đăng nhập, gửi dữ liệu nhạy cảm.

Các trang web có HTTPS thường hiển thị biểu tượng **ổ khóa** trên thanh địa chỉ trình duyệt.

**Ví dụ:**

Khi bạn truy cập https://www.bank.com, dữ liệu đăng nhập của bạn được mã hóa để tránh bị đánh cắp.

Các trang web như Facebook, Google, ngân hàng đều sử dụng HTTPS để bảo mật thông tin.

### ****3. Sự Khác Biệt Giữa HTTP Và HTTPS****

| **Tiêu chí** | **HTTP** | **HTTPS** |
| --- | --- | --- |
| Bảo mật | Không mã hóa dữ liệu | Mã hóa dữ liệu bằng SSL/TLS |
| Tốc độ | Nhanh hơn một chút | Chậm hơn do mã hóa nhưng an toàn hơn |
| Cổng sử dụng | 80 | 443 |
| Ứng dụng | Trang web công khai, không yêu cầu bảo mật cao | Trang web thương mại điện tử, ngân hàng, mạng xã hội |

### ****4. Bản Tin HTTP Request Và HTTP Response****

#### ****a. HTTP Request (Yêu Cầu HTTP)****

Là yêu cầu từ trình duyệt (client) gửi đến máy chủ (server).

**Cấu trúc HTTP Request:**

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0

Accept: text/html

**Các thành phần:**

**Request Method:** Phương thức yêu cầu (GET, POST, PUT, DELETE…).

**URL:** Đường dẫn của tài nguyên cần truy cập.

**HTTP Version:** Phiên bản HTTP (HTTP/1.1, HTTP/2).

**Headers:** Thông tin bổ sung như loại trình duyệt, loại nội dung chấp nhận.

**Các Phương Thức HTTP Chính:**

| **Phương Thức** | **Mô Tả** |
| --- | --- |
| **GET** | Lấy dữ liệu từ máy chủ (ví dụ: truy cập trang web) |
| **POST** | Gửi dữ liệu lên máy chủ (ví dụ: gửi form đăng ký) |
| **PUT** | Cập nhật dữ liệu trên máy chủ |
| **DELETE** | Xóa dữ liệu trên máy chủ |

#### ****b. HTTP Response (Phản Hồi HTTP)****

Là phản hồi từ máy chủ gửi về trình duyệt sau khi nhận HTTP Request.

**Cấu trúc HTTP Response:**

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 342

<html>

<body>

<h1>Chào mừng đến với Example.com</h1>

</body>

</html>

**Các thành phần:**

**Status Code:** Mã trạng thái HTTP (200, 404, 500…).

**Headers:** Thông tin bổ sung như loại nội dung, độ dài nội dung.

**Body:** Nội dung phản hồi (HTML, JSON, XML…).

**Các Mã Trạng Thái HTTP Thường Gặp:**

| **Mã Trạng Thái** | **Ý Nghĩa** |
| --- | --- |
| **200 OK** | Thành công |
| **301 Moved Permanently** | Chuyển hướng vĩnh viễn |
| **400 Bad Request** | Yêu cầu không hợp lệ |
| **401 Unauthorized** | Chưa xác thực |
| **403 Forbidden** | Bị từ chối truy cập |
| **404 Not Found** | Không tìm thấy tài nguyên |
| **500 Internal Server Error** | Lỗi máy chủ |

### ****5. Kết Luận****

**HTTP** là giao thức truyền tải dữ liệu không mã hóa, trong khi **HTTPS** bảo mật dữ liệu bằng SSL/TLS.

**HTTP Request** là yêu cầu từ trình duyệt đến máy chủ, gồm phương thức (GET, POST…), URL, headers.

**HTTP Response** là phản hồi từ máy chủ, gồm mã trạng thái (200, 404…), headers và nội dung trang web.

**HTTPS quan trọng** đối với các trang web có giao dịch tài chính, đăng nhập, và bảo vệ dữ liệu cá nhân.

**VI. Các mã trạng thái phản hồi của HTTP Response**

### ****1. Nhóm 1xx - Thông Tin (Informational Responses)****

Dùng để báo hiệu rằng máy chủ đã nhận được yêu cầu và đang xử lý tiếp.

| **Mã** | **Ý Nghĩa** | **Mô Tả** |
| --- | --- | --- |
| **100 Continue** | Tiếp tục | Máy chủ nhận được một phần yêu cầu và yêu cầu tiếp tục gửi phần còn lại. |
| **101 Switching Protocols** | Chuyển đổi giao thức | Máy chủ đồng ý thay đổi giao thức theo yêu cầu của client. |
| **103 Early Hints** | Gợi ý sớm | Gửi một số header trước khi phản hồi đầy đủ. |

### ****2. Nhóm 2xx - Thành Công (Success Responses)****

Báo hiệu rằng yêu cầu của client đã được xử lý thành công.

| **Mã** | **Ý Nghĩa** | **Mô Tả** |
| --- | --- | --- |
| **200 OK** | Thành công | Máy chủ xử lý yêu cầu thành công. |
| **201 Created** | Đã tạo | Tài nguyên mới đã được tạo thành công (thường gặp với phương thức POST). |
| **202 Accepted** | Đã chấp nhận | Yêu cầu được chấp nhận nhưng chưa xử lý xong. |
| **204 No Content** | Không có nội dung | Máy chủ đã xử lý yêu cầu nhưng không có nội dung trả về. |

### ****3. Nhóm 3xx - Chuyển Hướng (Redirection Responses)****

Báo hiệu rằng client cần thực hiện thêm bước chuyển hướng.

| **Mã** | **Ý Nghĩa** | **Mô Tả** |
| --- | --- | --- |
| **301 Moved Permanently** | Chuyển hướng vĩnh viễn | URL đã thay đổi vĩnh viễn, client nên sử dụng URL mới. |
| **302 Found (Moved Temporarily)** | Chuyển hướng tạm thời | URL tạm thời thay đổi, có thể quay lại URL cũ sau này. |
| **304 Not Modified** | Không thay đổi | Client có thể sử dụng phiên bản cũ được lưu trong bộ nhớ cache. |
| **307 Temporary Redirect** | Chuyển hướng tạm thời | Tương tự 302 nhưng yêu cầu dùng cùng phương thức HTTP như ban đầu. |
| **308 Permanent Redirect** | Chuyển hướng vĩnh viễn | Tương tự 301 nhưng đảm bảo sử dụng phương thức ban đầu. |

### ****4. Nhóm 4xx - Lỗi Từ Client (Client Errors)****

Báo hiệu rằng client gửi yêu cầu không hợp lệ.

| **Mã** | **Ý Nghĩa** | **Mô Tả** |
| --- | --- | --- |
| **400 Bad Request** | Yêu cầu không hợp lệ | Yêu cầu sai cú pháp hoặc lỗi dữ liệu. |
| **401 Unauthorized** | Chưa xác thực | Yêu cầu cần xác thực nhưng chưa có thông tin đăng nhập. |
| **403 Forbidden** | Bị cấm | Client không có quyền truy cập tài nguyên. |
| **404 Not Found** | Không tìm thấy | Tài nguyên không tồn tại hoặc URL sai. |
| **405 Method Not Allowed** | Phương thức không được phép | Phương thức HTTP (GET, POST...) không được hỗ trợ cho tài nguyên này. |
| **408 Request Timeout** | Hết thời gian chờ | Máy chủ không nhận được yêu cầu trong thời gian quy định. |
| **429 Too Many Requests** | Quá nhiều yêu cầu | Client gửi quá nhiều yêu cầu trong thời gian ngắn. |

### ****5. Nhóm 5xx - Lỗi Từ Máy Chủ (Server Errors)****

Báo hiệu rằng máy chủ gặp sự cố khi xử lý yêu cầu.

| **Mã** | **Ý Nghĩa** | **Mô Tả** |
| --- | --- | --- |
| **500 Internal Server Error** | Lỗi máy chủ | Máy chủ gặp lỗi không xác định. |
| **501 Not Implemented** | Chưa hỗ trợ | Máy chủ không hỗ trợ phương thức được yêu cầu. |
| **502 Bad Gateway** | Cổng trung gian lỗi | Máy chủ trung gian (proxy, gateway) nhận phản hồi không hợp lệ từ máy chủ gốc. |
| **503 Service Unavailable** | Dịch vụ không khả dụng | Máy chủ tạm thời quá tải hoặc đang bảo trì. |
| **504 Gateway Timeout** | Hết thời gian phản hồi | Máy chủ trung gian không nhận được phản hồi kịp thời từ máy chủ gốc. |
| **505 HTTP Version Not Supported** | Phiên bản HTTP không hỗ trợ | Máy chủ không hỗ trợ phiên bản HTTP được sử dụng. |

### ****6. Kết Luận****

**1xx**: Thông tin – Cho biết máy chủ đang xử lý yêu cầu.

**2xx**: Thành công – Yêu cầu được xử lý thành công.

**3xx**: Chuyển hướng – Client cần thực hiện bước tiếp theo để hoàn tất yêu cầu.

**4xx**: Lỗi từ client – Lỗi do yêu cầu không hợp lệ hoặc không có quyền truy cập.

**5xx**: Lỗi từ máy chủ – Máy chủ gặp lỗi khi xử lý yêu cầu.

Các mã trạng thái này giúp lập trình viên **debug lỗi, tối ưu website** và cải thiện trải nghiệm người dùng.

**VII. Phân biệt các phương thức (method) của HTTP Request**

### ****1. GET – Lấy Dữ Liệu****

**Chức năng**: Dùng để **lấy dữ liệu** từ server.

**Đặc điểm**:

Dữ liệu yêu cầu được đính kèm vào **URL** dưới dạng query string (?key=value).

**Không** thay đổi dữ liệu trên server.

Dễ bị lưu cache và có thể đánh dấu trên lịch sử trình duyệt.

Có giới hạn độ dài URL (tùy trình duyệt, thường khoảng 2048 ký tự).

**Ví dụ**:

GET /products?id=123 HTTP/1.1

Host: example.com

Truy vấn thông tin sản phẩm có id=123.

### ****2. POST – Gửi Dữ Liệu****

**Chức năng**: Dùng để **gửi dữ liệu** lên server (ví dụ: gửi form đăng ký, đăng nhập).

**Đặc điểm**:

Dữ liệu được gửi trong **body** của request, không hiển thị trên URL.

**Có thể thay đổi dữ liệu trên server** (tạo bản ghi mới, cập nhật dữ liệu).

Không bị giới hạn độ dài dữ liệu như GET.

**Không bị lưu cache.**

**Ví dụ**:

POST /register HTTP/1.1

Host: example.com

Content-Type: application/json

{

"username": "john\_doe",

"password": "secure123"

}

Gửi thông tin đăng ký tài khoản lên server.

### ****3. PUT – Cập Nhật Dữ Liệu****

**Chức năng**: Dùng để **cập nhật toàn bộ** tài nguyên trên server.

**Đặc điểm**:

Nếu tài nguyên tồn tại, nó sẽ bị **thay thế hoàn toàn**.

Nếu tài nguyên chưa tồn tại, server có thể tạo mới.

**Idempotent** (gửi nhiều lần kết quả vẫn như nhau).

**Ví dụ**:

PUT /users/123 HTTP/1.1

Host: example.com

Content-Type: application/json

{

"username": "john\_updated",

"email": "john@example.com"

}

Cập nhật toàn bộ thông tin của user có id=123.

### ****4. PATCH – Cập Nhật Một Phần Dữ Liệu****

**Chức năng**: Dùng để **cập nhật một phần** của tài nguyên.

**Đặc điểm**:

Chỉ thay đổi các thuộc tính được chỉ định, không thay thế toàn bộ dữ liệu như PUT.

**Không idempotent** nếu thay đổi có tính toán (ví dụ: tăng số lượng hàng).

**Ví dụ**:

PATCH /users/123 HTTP/1.1

Host: example.com

Content-Type: application/json

{

"email": "newemail@example.com"

}

👉 Chỉ cập nhật email cho user id=123, các thông tin khác giữ nguyên.

### ****5. DELETE – Xóa Dữ Liệu****

**Chức năng**: Dùng để **xóa tài nguyên** trên server.

**Đặc điểm:**

Yêu cầu server xóa tài nguyên được chỉ định.

**Có thể idempotent**, nhưng có trường hợp chỉ đánh dấu là đã xóa thay vì xóa thực sự.

**Ví dụ**:

DELETE /users/123 HTTP/1.1

Host: example.com

Xóa người dùng có id=123.

### ****6. HEAD – Lấy Header Của Response****

**Chức năng**: Tương tự như **GET**, nhưng **chỉ trả về phần header**, không có body.

**Ứng dụng**:

Kiểm tra tài nguyên có tồn tại không mà không cần tải nội dung.

Kiểm tra kích thước hoặc loại dữ liệu trước khi tải về.

**Ví dụ**:

HEAD /products/123 HTTP/1.1

Host: example.com

Kiểm tra xem sản phẩm có tồn tại mà không tải nội dung chi tiết.

### ****7. OPTIONS – Kiểm Tra Phương Thức Hỗ Trợ****

**Chức năng**: Dùng để kiểm tra **server hỗ trợ những phương thức nào** cho một tài nguyên.

**Ứng dụng**:

Dùng trong **CORS (Cross-Origin Resource Sharing)** để kiểm tra quyền truy cập.

**Ví dụ**:

OPTIONS /users HTTP/1.1

Host: example.com

Kiểm tra các phương thức HTTP được hỗ trợ trên /users.

**Phản hồi từ server**:

HTTP/1.1 200 OK

Allow: GET, POST, PUT, DELETE

Server hỗ trợ GET, POST, PUT, DELETE trên /users.

## ****So Sánh Các Phương Thức HTTP****

| **Phương Thức** | **Chức Năng** | **Idempotent?** | **Có Dữ Liệu Trong Body?** | **Dữ Liệu Hiển Thị Trên URL?** |
| --- | --- | --- | --- | --- |
| **GET** | Lấy dữ liệu | ✅ Có | ❌ Không | ✅ Có |
| **POST** | Gửi dữ liệu (tạo mới) | ❌ Không | ✅ Có | ❌ Không |
| **PUT** | Cập nhật toàn bộ dữ liệu | ✅ Có | ✅ Có | ❌ Không |
| **PATCH** | Cập nhật một phần dữ liệu | ❌ Không | ✅ Có | ❌ Không |
| **DELETE** | Xóa tài nguyên | ✅ Có | ❌ Không | ✅ Có hoặc ❌ Không |
| **HEAD** | Lấy header của response | ✅ Có | ❌ Không | ✅ Có |
| **OPTIONS** | Kiểm tra phương thức hỗ trợ | ✅ Có | ❌ Không | ✅ Có |

## ****Kết Luận****

**GET**: Lấy dữ liệu, không thay đổi dữ liệu server

**POST**: Gửi dữ liệu mới lên server, không idempotent.

**PUT**: Cập nhật toàn bộ dữ liệu, idempotent.

**PATCH**: Cập nhật một phần dữ liệu, không idempotent.

**DELETE**: Xóa dữ liệu, có thể idempotent.

**HEAD**: Lấy header, không có body.

**OPTIONS**: Kiểm tra phương thức được hỗ trợ.

Hiểu rõ từng phương thức giúp xây dựng API hiệu quả, tránh lỗi bảo mật và tối ưu hiệu suất.

**VIII. Công cụ GIT + cách sử dụng GITHUB**

## ****1. GIT Là Gì?****

Git là một **hệ thống quản lý phiên bản phân tán (Distributed Version Control System - DVCS)**, giúp theo dõi và quản lý các thay đổi trong mã nguồn của dự án.

**Lợi ích của Git:**  
✅ Theo dõi lịch sử thay đổi của mã nguồn.  
✅ Hỗ trợ làm việc nhóm, cho phép nhiều người cùng chỉnh sửa một dự án.  
✅ Dễ dàng quay lại phiên bản trước nếu có lỗi.  
✅ Hoạt động nhanh và hiệu quả trên cả dự án lớn.

## ****2. GitHub Là Gì?****

GitHub là một **dịch vụ lưu trữ mã nguồn** trực tuyến dựa trên Git, giúp lập trình viên **lưu trữ, chia sẻ và cộng tác** trên dự án của mình.

**Ngoài GitHub**, còn có các nền tảng khác như **GitLab, Bitbucket** hoạt động tương tự.

## ****3. Cài Đặt Git****

### ****Trên Windows:****

1.Tải Git từ trang chính thức: [https://git-scm.com/](https://git-scm.com/" \t "_new)

Cài đặt và chọn tùy chọn mặc định.

### ****Trên Linux (Ubuntu/Debian):****

sudo apt update

sudo apt install git

### ****Trên macOS:****

### brew install git

## ****4. Các Lệnh Cơ Bản Trong Git****

### ****4.1. Cấu Hình Git (Chỉ Cần Làm Một Lần)****

git config --global user.name "Tên Của Bạn"

git config --global user.email "email@example.com"

Định danh tác giả khi commit code.

### ****4.2. Khởi Tạo Git Trong Dự Án****

git init

Khởi tạo Git trong thư mục hiện tại.

### ****4.3. Kiểm Tra Trạng Thái Git****

git status

Kiểm tra những file nào đã thay đổi hoặc chưa được theo dõi.

### ****4.4. Thêm File Vào Git****

git add file1.txt # Thêm một file

git add . # Thêm tất cả file trong thư mục

Đánh dấu file để chuẩn bị commit.

### ****4.5. Commit Code (Lưu Trạng Thái Hiện Tại)****

git commit -m "Thêm file mới"

Lưu trạng thái hiện tại với mô tả thay đổi.

### ****4.6. Xem Lịch Sử Commit****

git log --oneline

Hiển thị danh sách commit theo thứ tự thời gian.

### ****4.7. Kiểm Tra Sự Khác Biệt Giữa Các File****

git diff

Hiển thị những thay đổi chưa commit.

### ****4.8.Hoàn Tác File Về Trạng Thái Ban Đầu****

git checkout -- file1.txt

Hủy bỏ các thay đổi chưa commit trong file.

### ****4.9. Xóa File Khỏi Git****

git rm file1.txt

git commit -m "Xóa file1.txt"

Xóa một file khỏi Git và commit thay đổi.

## ****5. Kết Nối Dự Án Với GitHub****

### ****5.1. Tạo Repository Trên GitHub****

1.Truy cập [GitHub](https://github.com/" \t "_new)

2. Đăng nhập và nhấn **New Repository**

3. Đặt tên repository và chọn **Public/Private**

4. Nhấn **Create Repository**

### ****5.2. Kết Nối Git Với GitHub****

### git remote add origin https://github.com/username/repository.git

git branch -M main

Liên kết Git với repository trên GitHub.

### ****5.3. Đẩy Code Lên GitHub****

git push -u origin main

Đẩy toàn bộ code lên GitHub.

### ****5.4. Clone Dự Án Về Máy****

git clone https://github.com/username/repository.git

Tải mã nguồn về máy từ GitHub.

### ****5.5. Kéo Code Mới Nhất Về Máy****

git pull origin main

Cập nhật code mới từ GitHub.

## ****6. Làm Việc Nhóm Với GitHub****

### ****6.1. Tạo Nhánh (Branch) Mới****

git branch feature-branch

git checkout feature-branch

Tạo một nhánh mới để làm việc riêng biệt.

### ****6.2. Gộp Nhánh Vào Main (Merge Branch)****

git checkout main

git merge feature-branch

Gộp code từ **feature-branch** vào nhánh **main**.

### ****6.3. Giải Quyết Xung Đột (Conflict)****

Khi có xung đột giữa các commit, Git sẽ yêu cầu bạn sửa file bị conflict.

Mở file bị lỗi, chỉnh sửa lại phần bị conflict.

Sau đó chạy:

git add .

git commit -m "Resolve conflict"

Xác nhận đã giải quyết xung đột.

## ****7. Các Tính Năng Khác Của GitHub****

### ****7.1. Fork Repository****

Cho phép sao chép một repository của người khác về tài khoản của bạn để chỉnh sửa mà không ảnh hưởng đến repo gốc.

### ****7.2. Pull Request (PR)****

Dùng để đề xuất thay đổi khi làm việc nhóm. Các bước:

1. Push code lên một nhánh mới trên GitHub.
2. Vào GitHub, chọn **Pull Request** → **New Pull Request**.
3. Mô tả thay đổi và gửi yêu cầu merge code.

## ****8. Kết Luận****

**Git** giúp quản lý phiên bản mã nguồn hiệu quả.

**GitHub** giúp chia sẻ code và làm việc nhóm.

Các lệnh cơ bản như **git init, git add, git commit, git push, git pull** giúp quản lý code dễ dàng.

Sử dụng **branch** và **merge** để làm việc nhóm hiệu quả.