**Web back-end (1) - Một số khái niệm**

1. **Web & Website**:

* Web là hệ thống thông tin hoạt động trên Internet.
* Website là tập hợp các trang web có liên quan, được tổ chức dưới một tên miền chung.

1. **Ứng dụng Web**:

* Là chương trình chạy trên trình duyệt, hoạt động theo mô hình client-server.
* Ví dụ: Facebook, Tiki.

1. **Kiến trúc Client-Server**:

* **Client (máy khách)**: Trình duyệt web gửi yêu cầu (request).
* **Server (máy chủ)**: Xử lý yêu cầu và gửi phản hồi (response).
* Quá trình duyệt web: Nhập URL → Gửi yêu cầu → Nhận phản hồi → Hiển thị trang web

Quá trình này diễn ra như sau:​

* Người dùng nhập địa chỉ URL vào thanh địa chỉ của trình duyệt.​
* Trình duyệt gửi yêu cầu đến máy chủ web tương ứng.​
* Máy chủ xử lý yêu cầu và gửi lại nội dung trang web.​
* Trình duyệt nhận nội dung và hiển thị trang web cho người dùng.​

**Web back-end (2) - Tổng quan về phát triển ứng dụng web**

**1. Phát triển ứng dụng web là gì?**

Phát triển ứng dụng web là quá trình tạo ra các ứng dụng chạy trên nền tảng web. Quá trình này bao gồm nhiều giai đoạn như:​

* Lập dự án​
* Xác định yêu cầu​
* Phân tích​
* Thiết kế​
* Lập trình​
* Triển khai
* Bảo trì và cập nhật



2. Các phương pháp tạo ứng dụng web

 **Sử dụng hệ quản trị nội dung (CMS):** CMS là phần mềm cho phép tạo, quản lý và chỉnh sửa nội dung trên trang web mà không yêu cầu kiến thức lập trình chuyên sâu. Các CMS phổ biến bao gồm WordPress, Joomla và Drupal. Ưu điểm của CMS là dễ sử dụng, tiết kiệm thời gian và linh hoạt trong việc tùy chỉnh giao diện và chức năng.

 **Sử dụng framework và thư viện hỗ trợ:** Phương pháp này dựa vào các framework và thư viện có sẵn để phát triển ứng dụng web, giúp tăng tốc độ phát triển và đảm bảo tính nhất quán. Ví dụ về các framework phổ biến là Django (Python), Ruby on Rails (Ruby) và Laravel (PHP).​

 **Lập trình thuần túy (không sử dụng framework):** Phương pháp này yêu cầu viết mã từ đầu mà không dựa vào các công cụ hỗ trợ, đòi hỏi kiến thức sâu về ngôn ngữ lập trình và các công nghệ web. Ưu điểm là kiểm soát hoàn toàn mã nguồn và tối ưu hóa hiệu suất, nhưng mất nhiều thời gian và công sức hơn.

**Web back-end (3) - Web server**

**1. Web server là gì?**

Web server có thể được hiểu theo hai nghĩa:​

* **Phần mềm:** Các chương trình như Apache, Nginx, IIS, Node.js được sử dụng để lưu trữ và cung cấp nội dung cho ứng dụng web.​
* **Phần cứng:** Máy chủ vật lý được cài đặt các phần mềm web server để phục vụ nội dung web.

**2. Chức năng chính của web server:**

* **Lưu trữ tập tin:** Lưu trữ tất cả các tập tin và dữ liệu của ứng dụng web, bao gồm HTML, hình ảnh, video.​
* **Nhận yêu cầu (request):** Tiếp nhận yêu cầu từ trình duyệt web của người dùng khi họ truy cập ứng dụng web.​
* **Xử lý yêu cầu:** Đọc nội dung tập tin, thực thi mã nguồn, kết nối và làm việc với cơ sở dữ liệu.​
* **Gửi phản hồi (response):** Trả về kết quả xử lý cho trình duyệt để hiển thị nội dung cho người dùng.

A diagram of a web server

AI-generated content may be incorrect.

**1. Người dùng gửi yêu cầu từ trình duyệt**

* Khi người dùng nhập địa chỉ URL vào thanh địa chỉ hoặc nhấn vào một liên kết, trình duyệt sẽ gửi một **HTTP request** đến web server.
* Yêu cầu này có thể là:
  + **GET request** (lấy dữ liệu, ví dụ: truy cập trang web)
  + **POST request** (gửi dữ liệu, ví dụ: đăng nhập, gửi form)

**2. Web server nhận yêu cầu và xử lý**

* Máy chủ web (ví dụ: Apache, Nginx) nhận yêu cầu và kiểm tra loại tài nguyên được yêu cầu:
  + Nếu tài nguyên là một tệp tĩnh (HTML, CSS, hình ảnh, video...), máy chủ sẽ gửi trực tiếp tệp đó về cho trình duyệt.
  + Nếu tài nguyên là một tệp động (PHP, Python, Node.js...), máy chủ sẽ gửi yêu cầu xử lý đến một chương trình hoặc ứng dụng phía server.

**3. Gửi yêu cầu đến ứng dụng back-end (nếu cần)**

* Nếu yêu cầu cần xử lý dữ liệu, web server sẽ chuyển tiếp yêu cầu đến ứng dụng web (backend), chẳng hạn như một **PHP script, Java servlet, hoặc API backend**.
* Ứng dụng back-end có thể:
  + Xử lý logic nghiệp vụ.
  + Truy vấn cơ sở dữ liệu.
  + Xử lý xác thực, đăng nhập, thanh toán...

**4. Ứng dụng back-end xử lý dữ liệu và gửi phản hồi**

* Sau khi xử lý yêu cầu, ứng dụng backend sẽ gửi kết quả về web server.
* Kết quả có thể là:
  + **Một trang HTML được tạo động** (ví dụ: danh sách sản phẩm, bài viết blog...)
  + **Dữ liệu JSON hoặc XML** (nếu server hoạt động như API)
  + **Mã trạng thái HTTP** (ví dụ: lỗi 404 nếu không tìm thấy tài nguyên)

**5. Web server gửi phản hồi về trình duyệt**

* Web server nhận kết quả từ ứng dụng backend và gửi phản hồi HTTP về cho trình duyệt.
* Phản hồi này có thể chứa:
  + **Trang web hoàn chỉnh** để hiển thị cho người dùng.
  + **Dữ liệu cần xử lý bằng JavaScript** nếu là ứng dụng web động.

**6. Trình duyệt hiển thị nội dung cho người dùng**

* Trình duyệt nhận nội dung HTML, CSS, JavaScript và hiển thị giao diện trang web.
* Nếu trang web có JavaScript, nó có thể tiếp tục gửi các yêu cầu AJAX để lấy thêm dữ liệu mà không cần tải lại trang.

**3. Một số phần mềm web server phổ biến:**

* Apache​
* Nginx​
* Node.js​
* IIS​
* Tomcat​
* Lighttpd​

**4. Cài đặt và sử dụng Nginx:**

Bài viết hướng dẫn cách tải, cài đặt và chạy Nginx trên máy tính:​

* **Tải Nginx:** Truy cập trang chủ [nginx.org](https://nginx.org/), chọn mục Download và tải phiên bản phù hợp với hệ điều hành.​
* **Cài đặt Nginx:** Giải nén thư mục nginx đã tải về vào ổ đĩa (ví dụ: C:\n​ginx-1.24.0).​
* **Chạy Nginx:** Nhấp đúp chuột vào tập tin nginx.exe để khởi động chương trình. Kiểm tra trong Task Manager để xác nhận tiến trình đang chạy.​
* A screenshot of a computer

  AI-generated content may be incorrect.

**Web back-end (4) - Ứng dụng web bán hàng**

vòng đời của một dự án gồm các bước:

- Lập dự án

- Xác định yêu cầu

- Phân tích

- Thiết kế

- Lập trình

- Triển khai

- Bảo trì và cập nhật

**Môi trường và công cụ sử dụng:**

* Ngôn ngữ lập trình phía client:​
  + HTML: Tạo giao diện.
  + CSS: Định dạng và trang trí giao diện.
  + JavaScript: Xử lý hiệu ứng và giao tiếp giữa client và server.
* Ngôn ngữ lập trình phía server:​
  + Ngôn ngữ: JavaScript.
  + Framework: Express.
* Web server:​
  + Node.js.
* Hệ quản trị cơ sở dữ liệu:​
  + PostgreSQL.
* Công cụ làm việc với PostgreSQL:​
  + DBeaver.
* Quản lý mã nguồn:​
  + Git, GitHub.

**2. Cài đặt các công cụ cần thiết**

* **VS Code:** Trình soạn thảo mã nguồn được sử dụng để viết và quản lý mã nguồn cho dự án. Hướng dẫn tải và cài đặt tại <https://code.visualstudio.com/>.​
* **Git:** Hệ thống quản lý phiên bản phân tán, hỗ trợ theo dõi và quản lý các thay đổi trong mã nguồn. Hướng dẫn tải và cài đặt tại <https://git-scm.com/>.

**Git thực hành (1) - Hệ thống quản lý phiên bản**

1. Phiên bản là gì?

Các bản khác nhau của tập tin, thư mục hoặc toàn bộ mã nguồn dự án. Trong quá trình phát triển, mã nguồn liên tục thay đổi, do đó cần có khả năng xem lại hoặc khôi phục trạng thái của dự án tại các thời điểm trước đó.

**2. Hệ thống quản lý phiên bản (Version Control System - VCS)**

VCS là phần mềm giúp lưu lại từng thay đổi của mã nguồn dự án và hỗ trợ:​

 Khôi phục trạng thái dự án ở các thời điểm khác nhau.​

 Xác định ai đã thực hiện các thay đổi và nội dung thay đổi.

 Khôi phục nội dung mã nguồn bị xóa

 So sánh những thay đổi của dự án theo thời gian.​

**Hệ thống quản lý phiên bản cục bộ:** Lưu trữ các phiên bản tại máy tính cá nhân, thường sử dụng công cụ như RCS. Phương pháp này gặp khó khăn khi làm việc nhóm do thiếu khả năng chia sẻ và cộng tác.

**Hệ thống quản lý phiên bản tập trung (Centralized Version Control Systems – CVCS):** Sử dụng một máy chủ trung tâm chứa các phiên bản của dự án, các máy khách lấy và cập nhật dữ liệu từ máy chủ này.

**Hệ thống quản lý phiên bản phân tán (Distributed Version Control Systems - DVCS):** Mỗi máy khách lưu trữ một bản sao đầy đủ của kho chứa, bao gồm toàn bộ lịch sử phiên bản. Điều này cho phép khôi phục dữ liệu từ bất kỳ máy khách nào nếu máy chủ gặp sự cố. Ví dụ: Git, Mercurial, Darcs.

**Git thực hành (2) - Tổng quan về Git**

**1. Git là gì?**

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System - DVCS), cho phép theo dõi và quản lý các thay đổi trong mã nguồn của dự án một cách hiệu quả. Một số tính năng nổi bật của Git bao gồm:​

Theo dõi lịch sử thay đổi

Cộng tác hiệu quả

Phân nhánh và hợp nhất

Bảo mật

Phân tán

**2. Tải và cài đặt Git**

Để sử dụng Git, cần tải và cài đặt phần mềm này trên máy tính:​

* Truy cập trang chủ <https://git-scm.com/>, chọn phiên bản phù hợp với hệ điều hành đang sử dụng và tải về.​
* Sau khi tải về, thực hiện cài đặt như một phần mềm thông thường.​
* Để kiểm tra Git đã được cài đặt thành công hay chưa:​
  + Mở cửa sổ dòng lệnh (Command Prompt).​
  + Nhập lệnh git --version hoặc git -v và nhấn Enter.​
  + Nếu xuất hiện thông tin về phiên bản của Git, có nghĩa là Git đã được cài đặt thành công trên máy tính.



**Web back-end (5) - Nodejs và NPM**

**1. Node.js là gì?**

Node.js là một môi trường chạy JavaScript mã nguồn mở, đa nền tảng, cho phép phát triển các ứng dụng như máy chủ web, ứng dụng web, công cụ dòng lệnh và các tập lệnh. Node.js không phải là một framework hay ngôn ngữ lập trình, mà là một nền tảng bao gồm nhiều thành phần như:​

* **V8 Engine:** Trình biên dịch và thực thi mã JavaScript.​
* **HTTP Module:** Module cho phép tạo máy chủ HTTP.​
* **NPM (Node Package Manager):** Công cụ quản lý các gói thư viện và module.

**NPM (Node Package Manager)**

NPM là công cụ đi kèm với Node.js, dùng để quản lý các thư viện và module trong ứng dụng web. NPM giúp cài đặt, cập nhật và quản lý các gói phụ thuộc một cách hiệu quả.​

Để kiểm tra NPM đã được cài đặt:​

* **Kiểm tra phiên bản:** Mở cửa sổ dòng lệnh và nhập npm -v. Nếu hiển thị phiên bản của NPM, nghĩa là cài đặt thành công.​

A screenshot of a computer

AI-generated content may be incorrect.

NPM bao gồm ba thành phần chính:​

* **Website:** Trang web <https://www.npmjs.com/> để tìm kiếm và khám phá các gói thư viện.​
* **Công cụ dòng lệnh (CLI):** Cho phép cài đặt và quản lý các gói trực tiếp từ dòng lệnh.​
* **Registry:** Kho lưu trữ các gói thư viện.​

Để sử dụng một gói thư viện, ví dụ như Express, bạn có thể:​[Langbiang](https://legiacong.blogspot.com/2021/10/web-back-end-5-lap-trinh-phia-server.html?utm_source=chatgpt.com)

* **Tìm kiếm trên website NPM:** Truy cập <https://www.npmjs.com/> và tìm kiếm "express".​
* **Cài đặt bằng CLI:** Mở cửa sổ dòng lệnh trong thư mục dự án và nhập npm install express để cài đặt Express vào dự án.

**Web back-end (6) - Cấu hình Web server trong Nodejs**

* 1. Tạo tập tin package.json bằng NPM

**Mở Terminal hoặc Command Prompt:** Sử dụng công cụ dòng lệnh như CMD, PowerShell hoặc Terminal trong VS Code.

 **Di chuyển đến thư mục dự án:** Sử dụng lệnh cd để chuyển đến thư mục chứa dự án, ví dụ: cd E:\TeoShop.​

 **Khởi tạo package.json:** Chạy lệnh npm init -y để tạo tập tin package.json với các giá trị mặc định. Nếu thành công, nội dung tập tin sẽ bao gồm các thông tin như tên dự án, phiên bản, điểm khởi đầu (main), các lệnh (scripts), từ khóa, tác giả, giấy phép và mô tả.​

**2. Cài đặt và sử dụng Express để tạo Web server**

Express là một framework phổ biến cho Node.js, giúp đơn giản hóa việc xây dựng các ứng dụng web. Để cài đặt và sử dụng Express:​

* **Cài đặt Express:** Trong thư mục dự án, chạy lệnh npm install express để thêm Express vào dự án và cập nhật package.json với thông tin về thư viện này.​
* **Tạo tập tin index.js:** Tạo một tập tin index.js trong thư mục dự án với nội dung sau:

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Xin chào, đây là trang chủ!');

});

app.listen(port, () => {

console.log(`Ứng dụng đang chạy tại http://localhost:${port}`);

});

**Web back-end (7) - Local và Global, Dependencies và devDependencies**

**1. Khái niệm Local và Global trong NPM**

* **Cài đặt cục bộ (Local):** Gói được cài đặt chỉ áp dụng trong phạm vi dự án hiện tại. Thông tin về gói này được lưu trong tập tin package.json dưới mục dependencies, và mã nguồn của gói được đặt trong thư mục node\_modules của dự án.​
* **Cài đặt toàn cục (Global):** Gói được cài đặt có thể sử dụng trên toàn hệ thống, không phụ thuộc vào dự án cụ thể nào. Khi cài đặt theo kiểu này, thông tin về gói không được ghi vào package.json của bất kỳ dự án nào

**2. Cài đặt gói theo kiểu cục bộ**

* **Cú pháp lệnh:** npm install <package\_name>

**3. Cài đặt gói theo kiểu toàn cục**

* **Cú pháp lệnh:** npm install <package\_name> -g

**4. Phân biệt Dependencies và devDependencies**

* **Dependencies:** Là các gói cần thiết cho ứng dụng hoạt động bình thường. Chúng được liệt kê trong mục dependencies của package.json.​
* **devDependencies:** Là các gói chỉ cần thiết trong quá trình phát triển, như các công cụ kiểm thử hoặc biên dịch. Chúng được liệt kê trong mục devDependencies của package.json.

**Web back-end (8) - Một số chủ đề JavaScript (1)**

**1. Lập trình đồng bộ trong JavaScript**

Lập trình đồng bộ (synchronous programming) trong JavaScript là kỹ thuật mà các lệnh được thực thi tuần tự, từ trên xuống dưới. Mỗi lệnh phải hoàn thành trước khi lệnh tiếp theo được thực thi. Điều này có nghĩa là nếu một tác vụ mất nhiều thời gian (như đọc tập tin, gọi API), chương trình sẽ bị "chặn" cho đến khi tác vụ đó hoàn tất.​

Ví dụ về lập trình đồng bộ:

function cong(a, b) {

return a + b;

}

console.log("Bắt đầu");

let kq = cong(6, 8); // thực thi đồng bộ

console.log("Kết quả:", kq);

console.log("Kết thúc");

Kết quả khi chạy:

Bắt đầu

Kết quả: 14

Kết thúc

**2. Hàm ẩn danh trong JavaScript**

Hàm ẩn danh (anonymous function) là hàm không có tên khi được định nghĩa. Thay vì khai báo với từ khóa function đi kèm tên, hàm ẩn danh thường được gán trực tiếp vào biến, truyền làm tham số, hoặc thực thi ngay lập tức mà không cần gọi hàm.

VD:

// Gán vào biến

const cong = function(a, b) {

return a + b;

};

// Truyền làm tham số

setTimeout(function() {

console.log("Đã hết 3 giây!");

}, 3000);

// Thực thi ngay lập tức

(function() {

console.log("Hàm chạy ngay sau khi định nghĩa!");

})();

**Web back-end (9) - Một số chủ đề JavaScript (2)**

**1. Hàm mũi tên (Arrow Function)**

Hàm mũi tên, được giới thiệu trong ES6 (ECMAScript 2015), cung cấp một cách viết hàm ngắn gọn và hiện đại trong JavaScript. Chúng thường được sử dụng trong lập trình bất đồng bộ và các tình huống khác. Hàm mũi tên sử dụng ký hiệu => để kết nối tham số và thân hàm, mang lại sự ngắn gọn và một số đặc tính độc đáo về ngữ nghĩa.​

*Cú pháp của hàm mũi tên:*

* **Không tham số:** () => { ... }
* **Một tham số:** x => { ... } (không cần dấu ngoặc cho một tham số)​
* **Nhiều tham số:** (x, y) => { ... }

Hàm không có tham số:

const thongBao = () => {

console.log("Chưa có thông báo, khi nào có thông báo sẽ thông báo!");

}

thongBao();

Hàm có một tham số:

const binhPhuong = x => x \* x;

console.log(binhPhuong(3)); // 9

Hàm có nhiều tham số:

const cong = (x, y) => x + y;

console.log(cong(2, 4)); // 6

**2. Lập trình bất đồng bộ trong JavaScript**

Lập trình bất đồng bộ (asynchronous programming) cho phép JavaScript thực thi các tác vụ mà không chặn luồng chính, giúp cải thiện hiệu suất và trải nghiệm người dùng. Trong lập trình đồng bộ, các lệnh được thực thi tuần tự, và một lệnh phải hoàn thành trước khi lệnh tiếp theo được thực thi. Điều này có thể gây ra tình trạng "chặn" khi gặp các tác vụ mất nhiều thời gian như đọc tập tin, gọi API, hoặc giao tiếp client-server.

**Web back-end (10) - Hàm callback**

**1. Hàm callback là gì?**

Trong JavaScript, hàm callback là một hàm được truyền vào một hàm khác như một tham số và sẽ được thực thi sau khi hàm "cha" hoàn thành một tác vụ nào đó. Nói cách khác, callback cho phép một đoạn mã được thực thi sau khi một đoạn mã khác hoàn thành. Đây là một trong ba kỹ thuật được sử dụng để lập trình bất đồng bộ trong JavaScript.​

**2. Đặc điểm của hàm callback:**

* **Truyền như tham số:** Callback là một hàm được truyền vào hàm khác để thực thi sau khi một điều kiện hoặc tác vụ hoàn tất
* **Thực thi bất đồng bộ:** Thường dùng để xử lý các tác vụ không chặn luồng chính (non-blocking), như truy cập thiết bị (đọc/ghi tập tin), gọi API, hoặc trong các hàm hẹn giờ (timer).​
* **Tính linh hoạt:** Có thể là hàm ẩn danh (anonymous function), hàm mũi tên (arrow function), hoặc hàm đã được định nghĩa trước (function expression)
* **"Callback Hell":** Nếu lồng quá nhiều callback, mã nguồn có thể trở nên khó đọc và khó bảo trì. Callback hell thường được giải quyết bằng Promise hoặc async/await.

3. Ứng dụng của hàm callback:

Xử lý sự kiện (event handling)

Tác vụ bất đồng bộ

Xử lý mảng

Hoàn thành tác vụ

Tùy chỉnh logic

**Git thực hành (3) - Cấu hình định danh người dùng**

 **System (hệ thống):** Áp dụng cho tất cả người dùng và kho lưu trữ trên hệ thống. Tập tin cấu hình nằm tại C:\Program Files\Git\etc\gitconfig trên Windows. Cần quyền quản trị để thiết lập.​

 **Global (người dùng):** Áp dụng cho tất cả kho lưu trữ của người dùng hiện tại. Tập tin cấu hình nằm tại C:\Users\<tên\_người\_dùng>\.gitconfig trên Windows. Đây là phạm vi được sử dụng phổ biến nhất.​

 **Local (kho lưu trữ):** Áp dụng cho một kho lưu trữ cụ thể. Tập tin cấu hình nằm tại .git\config trong thư mục gốc của kho lưu trữ.

Để thiết lập tên và email cho tất cả các kho lưu trữ của người dùng hiện tại, sử dụng lệnh:​

git config --global user.name "Tên của bạn"

git config --global user.email [email@example.com](mailto:email@example.com)

Để kiểm tra các cấu hình đã thiết lập, dùng lệnh:

git config –list

**Git thực hành (4) - Các khu vực làm việc của Git**

 **Thư mục làm việc (Working Directory):** Đây là nơi bạn trực tiếp chỉnh sửa, thêm hoặc xóa các tập tin trên máy tính của mình. Nó chứa bản sao hiện tại của dự án, bao gồm cả các thay đổi chưa được theo dõi bởi Git. Ví dụ, khi bạn tạo một tập tin mới như index.js trong dự án, tập tin này sẽ xuất hiện trong thư mục làm việc.​

 **Khu vực tổ chức tạm (Staging Area):** Còn gọi là khu tạm, đây là vùng trung gian lưu trữ các thay đổi mà bạn đã chọn để chuẩn bị cho việc commit. Các thay đổi được thêm vào khu tạm bằng lệnh git add. Ví dụ, sau khi chỉnh sửa index.js, bạn sử dụng git add index.js để đưa thay đổi này vào khu tạm.​

 **Kho chứa (Repository):** Đây là nơi lưu trữ tất cả các tập tin và lịch sử thay đổi của dự án. Thư mục .git trong dự án chính là kho chứa. Khi bạn thực hiện lệnh git commit, các thay đổi từ khu tạm sẽ được ghi vào kho chứa, tạo thành một điểm lưu trữ trong lịch sử phiên bản của dự án.

**Git thực hành (5) - Các khu vực làm việc của Git**

**1. Khu tạm (Staging Area):**

Khu tạm, còn được gọi là index, là một khu vực trung gian giữa Thư mục làm việc (Working Directory) và Kho chứa (Repository). Nó cho phép bạn chọn lọc và tổ chức các thay đổi mà bạn muốn đưa vào commit tiếp theo.​

* **Gỡ bỏ tập tin khỏi Khu tạm:** Nếu bạn đã thêm một tập tin vào Khu tạm bằng lệnh git add nhưng sau đó quyết định không muốn commit tập tin đó nữa, bạn có thể sử dụng lệnh git rm --cached <tên\_tập\_tin> để gỡ bỏ tập tin khỏi Khu tạm mà không xóa nó khỏi Thư mục làm việc. Ví dụ:

git rm --cached index.js

**2. Tập tin .gitignore:**

Tập tin .gitignore là một cơ chế trong Git cho phép bạn chỉ định các tập tin hoặc thư mục mà Git sẽ bỏ qua khi thực hiện các lệnh như git add hoặc git status. Điều này giúp loại bỏ các tập tin không cần thiết khỏi lịch sử phiên bản, như tập tin tạm, tập tin log, hoặc thư viện phụ thuộc.​

* **Công dụng của .gitignore:**
  + **Tránh commit "rác":** Ngăn chặn việc thêm các tập tin không liên quan vào kho chứa.​
  + **Giữ lịch sử sạch sẽ:** Tập trung vào mã nguồn chính, giúp lịch sử commit rõ ràng và dễ theo dõi.​
  + **Tăng hiệu quả:** Giảm kích thước kho chứa và tránh xung đột không cần thiết.​
* **Cách sử dụng .gitignore:**
  + **Tạo tập tin .gitignore** trong thư mục gốc của dự án.​
  + **Thêm các mẫu (pattern)** của tập tin hoặc thư mục muốn bỏ qua vào tập tin .gitignore. Ví dụ

node\_modules/

dist/

.env

\*.log

\*.tmp

**Web back-end (11) - Lập trình giao diện**

**1. Commit đầu tiên với Git**

* **Mục tiêu:** Khởi tạo dự án web (ví dụ: "TeoShop") và thực hiện commit đầu tiên để quản lý mã nguồn.​[Langbiang](https://legiacong.blogspot.com/2025/03/web-back-end-11-lap-trinh-giao-dien.html?m=1&utm_source=chatgpt.com)
* **Các bước thực hiện:**
  + Tạo tập tin .gitignore để loại trừ các tập tin không cần thiết khỏi kho lưu trữ.
  + Sử dụng lệnh git add . để thêm tất cả các tập tin vào khu vực staging.
  + Thực hiện commit với thông điệp mô tả, ví dụ: git commit -m "Initialize TeoShop project with Express and Nodemon".
  + Kiểm tra lịch sử commit bằng lệnh git log.

**2. Sử dụng Free HTML Template**

**Web back-end (12) - Ứng dụng client-server**

**1. Ứng dụng cục bộ (Local Application)**

Ứng dụng cục bộ là những phần mềm chạy trực tiếp trên thiết bị của người dùng mà không cần kết nối mạng. Chúng thường được phát triển bằng các ngôn ngữ như C, C++, Java, hoặc Python và sử dụng tài nguyên hệ thống như tập tin, cơ sở dữ liệu cục bộ. Ví dụ: Microsoft Word, Excel, Photoshop.​

**2. Ứng dụng client-server**

Đây là mô hình phổ biến trong phát triển web hiện đại, bao gồm hai thành phần chính:​

* Client (máy khách): Gửi yêu cầu (request) để truy cập dịch vụ hoặc tài nguyên. Client thường là giao diện người dùng trên trình duyệt web hoặc ứng dụng di động.​
* Server (máy chủ): Nhận yêu cầu, xử lý và gửi phản hồi (response) về client. Server lưu trữ dữ liệu, thực hiện xử lý logic và cung cấp dịch vụ

Giao tiếp giữa client và server thường diễn ra qua mạng sử dụng các giao thức như HTTP hoặc TCP/IP. Trong giai đoạn phát triển, client và server có thể chạy trên cùng một máy tính để thuận tiện cho việc kiểm thử

**3. Ứng dụng web dưới góc nhìn lập trình viên**

Ứng dụng web là một dạng cụ thể của mô hình client-server, trong đó:​

* Client: Trình duyệt web hiển thị giao diện người dùng.​
* Server: Web server xử lý các yêu cầu từ client, thực hiện logic nghiệp vụ và truy cập cơ sở dữ liệu.​

Lập trình viên thường chia thành:​

* Front-end Developer: Phát triển giao diện người dùng bằng HTML, CSS, JavaScript và các framework liên quan.​
* Back-end Developer: Xử lý logic phía server sử dụng các ngôn ngữ như Java, Python, PHP, hoặc JavaScript (Node.js), cùng với cơ sở dữ liệu và các framework hỗ trợ.​
* Full-stack Developer: Thành thạo cả front-end và back-end, có khả năng phát triển toàn bộ ứng dụng web.

**Web back-end (13) - Lập trình client-server**

**1. Quy trình xử lý yêu cầu trong mô hình client-server**

1. **Client gửi yêu cầu (request)**: Người dùng sử dụng trình duyệt web để gửi yêu cầu đến server.​
2. **Server nhận yêu cầu**: Web server luôn ở trạng thái hoạt động, sẵn sàng nhận các yêu cầu từ client.​
3. **Phân tích và xử lý yêu cầu**: Server phân tích yêu cầu và xử lý tương ứng.​
4. **Truy xuất dữ liệu**: Nếu cần, server truy xuất dữ liệu từ cơ sở dữ liệu hoặc các nguồn khác.​
5. **Tạo phản hồi (response)**: Server tạo phản hồi, thường là một trang HTML đã được chèn dữ liệu.​
6. **Gửi phản hồi về client**: Server gửi phản hồi về cho client.​
7. **Client hiển thị nội dung**: Trình duyệt nhận phản hồi và hiển thị nội dung cho người dùng.

2. **Khởi tạo web server với Express**

'use strict';

const express = require('express');

const app = express();

const port = process.env.PORT || 9000;

// Khởi động web server

app.listen(port, () => {

console.log(`Server đang chạy trên cổng ${port}`);

});

**Web back-end (14) - Lập trình hướng sự kiện**

**1. Lập trình hướng sự kiện là gì?**

Lập trình hướng sự kiện là mô hình mà luồng thực thi của chương trình được điều khiển bởi các sự kiện (events) như:​

* Hành động của người dùng (nhấp chuột, gõ phím).​
* Thông điệp từ chương trình khác.​
* Sự kiện hệ thống (nhận request, đọc/ghi tập tin).​

Thay vì chạy tuần tự, chương trình sẽ chờ đợi các sự kiện xảy ra và thực thi các hàm xử lý sự kiện (event handler) tương ứng.​

**2. Lập trình hướng sự kiện trong Node.js**

Node.js được xây dựng dựa trên mô hình này, giúp xử lý hiệu quả các ứng dụng I/O-intensive như web server, API, ứng dụng mạng.​

**Các thành phần chính:**

* **Event:** Hành động hoặc thay đổi trạng thái trong ứng dụng (ví dụ: HTTP request, hoàn thành đọc tập tin).​
* **EventEmitter:** Bộ quản lý sự kiện, cho phép đăng ký và phát sự kiện.​
* **Event Loop:** Vòng lặp sự kiện, liên tục kiểm tra hàng đợi sự kiện và thực thi các hàm callback khi có sự kiện xảy ra.​
* **Event Handler:** Hàm xử lý sự kiện, được gọi khi sự kiện tương ứng xảy ra.​

**3. Thực hành với EventEmitter**

Bài viết hướng dẫn cách sử dụng module events của Node.js:​

1. Tạo một EventEmitter.​
2. Đăng ký một sự kiện với hàm xử lý bằng on().​
3. Phát sự kiện bằng emit().​

const EventEmitter = require('events');

const emitter = new EventEmitter();

emitter.on('greet', () => {

console.log('Hello, world!');

});

emitter.emit('greet');

**Web back-end (15) - Event-driven, Route handler và Middleware trong Express**

**1. Lập trình hướng sự kiện trong Express**

Express.js, xây dựng trên nền tảng Node.js, áp dụng mô hình lập trình hướng sự kiện để xử lý các yêu cầu HTTP. Trong mô hình này:​

* Mỗi yêu cầu HTTP (GET, POST, PUT, DELETE) được xem như một sự kiện.​
* Các module như http, fs, stream đóng vai trò là Event Emitter.​
* Express sử dụng Vòng lặp sự kiện (Event Loop) của Node.js để lắng nghe và xử lý các sự kiện một cách bất đồng bộ (asynchronous) và không chặn luồng (non-blocking).​
* Các Route handler và Middleware hoạt động như các hàm xử lý sự kiện (Event Handler), được gọi khi sự kiện tương ứng xảy ra.​

2. **Thực hành với GET request**

'use strict';

const express = require('express');

const app = express();

const port = process.env.PORT || 9000;

app.get('/', (req, res) => {

const name = req.query.name;

res.send(`Web server chào bạn ${name}`);

});

app.listen(port, () => {

console.log(`server đang chạy trên cổng ${port}`);

});

3. **Sử dụng Middleware để ghi log**

app.use((req, res, next) => {

console.log(`[${new Date().toISOString()}] nhận GET request tại ${req.url}`);

next(); // Chuyển tiếp yêu cầu đến Route handler tiếp theo

});

**4. Route handler và Middleware trong Express**

* **Route handler**: Là các hàm xử lý được gắn với một route cụ thể trong ứng dụng Express. Chúng được gọi khi một yêu cầu HTTP từ client khớp với route và method tương ứng. Route handler xử lý logic chính cho yêu cầu và trả về phản hồi.​
* **Middleware**: Là các hàm trung gian có thể xử lý yêu cầu trước khi nó đến Route handler. Middleware có thể thực hiện các tác vụ như xác thực, ghi log, xử lý lỗi, hoặc sửa đổi yêu cầu và phản hồi.