CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN

* 1. Lí do chọn đề tài
  2. Mục tiêu và nhiệm vụ của đề tài
     1. Mục tiêu
     2. Nhiệm vụ
  3. Phạm vi nghiên cứu
  4. Kết quả dự kiến

CHƯƠNG 2: KIẾN THỨC NỀN TẢNG

1. Cơ sở lí thuyết
2. Web server

**Web server**

Một web server là một phần mềm hoặc phần cứng được sử dụng để lưu trữ, xử lý và cung cấp các trang web và tài nguyên web khác cho các trình duyệt web của người dùng. Nó hoạt động như một trung gian giữa máy tính của bạn và các trang web mà bạn muốn truy cập.

Khi một trình duyệt web (như Chrome hoặc Firefox) gửi yêu cầu đến một trang web, nó sẽ gửi một HTTP request tới địa chỉ IP của máy chủ chứa trang web đó. Máy chủ sau đó sẽ xử lý yêu cầu và gửi trang web tương ứng (HTTP response) trở lại trình duyệt.

**Express, NodeJS**

Node.js là môi trường thực thi mã JavaScript phía máy chủ, cho phép bạn chạy mã JavaScript trên máy chủ.

Express.js (hoặc gọi tắt là Express) là một framework ứng dụng web phía máy chủ (back-end) xây dựng trên ngôn ngữ JavaScript và chạy trên môi trường Node.js. Express cung cấp các công cụ và tính năng để xây dựng và quản lý web server, cho phép xử lý yêu cầu HTTP từ trình duyệt và cung cấp các tệp tin, dữ liệu hoặc tương tác với cơ sở dữ liệu.

Express không phải là một web server độc lập mà nó chạy trên môi trường Node.js. Express sử dụng Node.js để chạy máy chủ, và từ đó xử lý các yêu cầu và phản hồi lại cho trình duyệt web.

1. Mô hình RESTful API

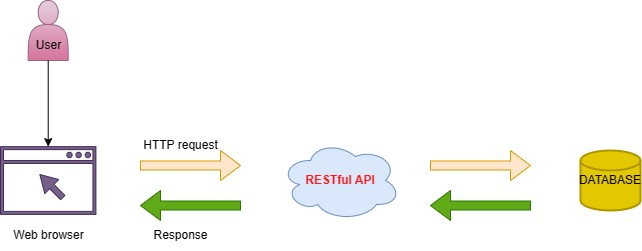
**Khái niệm**

RESTful API (Representational State Transferful Application Programming Interface) là một kiểu thiết kế cho các giao diện lập trình ứng dụng (API) được thiết kế để sử dụng các tiêu chuẩn và các nguyên tắc đơn giản để tạo ra các giao tiếp hiệu quả giữa các phần mềm khác nhau. Kiểu thiết kế này tập trung vào tài nguyên (resource) và các thao tác mà bạn có thể thực hiện trên chúng.

Một RESTful API cung cấp một cách tiếp cận chuẩn hóa và trực quan cho các ứng dụng và dịch vụ để truy cập và quản lý dữ liệu. Các ứng dụng có thể sử dụng RESTful API để gửi các yêu cầu và nhận phản hồi từ các máy chủ hoặc dịch vụ web.

**Cách thức hoạt động**

RESTful API hoạt động như một giao diện giữa khách hàng và máy chủ, cho phép khách hàng yêu cầu và tương tác với các tài nguyên thông qua các phương thức HTTP chuẩn. Các yêu cầu được xử lý bởi API, và sau đó phản hồi được trả về cho khách hàng. API có thể tương tác với cơ sở dữ liệu để lấy hoặc cập nhật thông tin.



1. Ngôn ngữ lập trình
2. **Ngôn ngữ lập trình**

**Khái niệm**

JavaScript là một ngôn ngữ lập trình đa năng được sử dụng chủ yếu trong phát triển ứng dụng web. Nó có khả năng thực hiện các tác vụ phía máy khách (client-side) trên trình duyệt web, như thêm, sửa đổi và tương tác với nội dung trang web.

**Đặc điểm nổi bật**

* Hướng sự kiện: JavaScript cho phép bạn thêm các bộ xử lý sự kiện (event handler) để phản ứng với các sự kiện như nhấp chuột, gõ phím, v.v.
* Hỗ trợ nhiều môi trường
* Hỗ trợ các thư viện và framework phong phú
* Hỗ trợ đa nền tảng

1. **MEVN stack**

**MongoDB**

MongoDB là một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL) mã nguồn mở được thiết kế để lưu trữ và quản lý dữ liệu. Đây là một cơ sở dữ liệu không yêu cầu cấu trúc dữ liệu cụ thể và không sử dụng ngôn ngữ truy vấn SQL.

* **Cơ sở dữ liệu dựa trên tài liệu (Document-Oriented):**

MongoDB lưu trữ dữ liệu dưới dạng tài liệu (documents) có định dạng JSON-like, cho phép lưu trữ thông tin liên quan vào cùng một bản ghi.

* **Không yêu cầu cấu trúc dữ liệu cố định:**

MongoDB không đòi hỏi cấu trúc dữ liệu cố định cho các tài liệu. Điều này cho phép bạn lưu trữ các tài liệu với các trường khác nhau mà không cần phải định nghĩa trước cấu trúc của chúng.

* **Hỗ trợ các tương tác phức tạp:**

MongoDB hỗ trợ truy vấn phức tạp và các phép toán trong dữ liệu để thực hiện các thao tác phức tạp như thao tác tổ chức, thống kê và nhiều tính toán khác.

* **Hỗ trợ truy vấn sắp xếp và tìm kiếm:**

MongoDB cung cấp các tùy chọn để tìm kiếm, sắp xếp và lọc dữ liệu một cách hiệu quả.

* **Hỗ trợ chia nhỏ dữ liệu (Sharding):**

MongoDB cho phép chia nhỏ cơ sở dữ liệu và phân phối dữ liệu trên nhiều nút, giúp cải thiện hiệu suất và khả năng mở rộng.

* **Hỗ trợ đa nền tảng và môi trường nhúng:**

MongoDB có thể chạy trên nhiều nền tảng khác nhau và có khả năng nhúng vào các ứng dụng.

**VueJS**

Vue.js là một thư viện JavaScript mã nguồn mở được sử dụng để xây dựng giao diện người dùng đơn trang (Single Page Applications - SPAs) và các ứng dụng web động. Nó tập trung vào việc xây dựng giao diện người dùng hiệu quả và dễ sử dụng.

**Đặc điểm nổi bật:**

* **Reactive Data Binding:**

Vue.js sử dụng một hệ thống liên kết dữ liệu reactivity mạnh mẽ. Điều này có nghĩa là khi dữ liệu thay đổi, giao diện người dùng sẽ tự động cập nhật để phản ánh các thay đổi đó.

* **Components-Based Architecture:**

Vue.js sử dụng kiến trúc dựa trên các thành phần (components), cho phép bạn tái sử dụng và tổ chức mã nguồn của bạn một cách dễ dàng.

* **Virtual DOM:**

Vue.js sử dụng một Virtual DOM để tối ưu hóa việc cập nhật DOM thực tế. Điều này giúp tăng tốc quá trình render và cập nhật giao diện người dùng.

* **Hỗ trợ mở rộng:**

Vue.js có một cộng đồng phát triển lớn và nhiều plugin, middleware, và các thư viện hữu ích khác để hỗ trợ việc phát triển ứng dụng.

* **Progressive Framework:**

Vue.js là một framework linh động, có thể được triển khai từng phần trong các dự án hiện có, điều này cho phép bạn tích hợp Vue.js vào các ứng dụng web đã tồn tại.

* **Hỗ trợ SSR (Server-Side Rendering):**

Vue.js hỗ trợ Server-Side Rendering, giúp tối ưu hóa trải nghiệm người dùng và SEO.

**ExpressJS**

Express.js (đôi khi gọi tắt là Express) là một framework ứng dụng web phía máy chủ (backend) được xây dựng trên ngôn ngữ JavaScript và chạy trên môi trường Node.js. Express cung cấp các công cụ và tính năng để xây dựng và quản lý web server, cho phép xử lý yêu cầu HTTP từ trình duyệt và cung cấp các tệp tin, dữ liệu hoặc tương tác với cơ sở dữ liệu.

**Đặc điểm nổi bật:**

* **Không yêu cầu cấu hình phức tạp:**

Express được thiết kế để đơn giản và dễ sử dụng. Nó không yêu cầu nhiều cấu hình phức tạp, giúp bạn nhanh chóng bắt đầu phát triển ứng dụng.

* **Middleware-based:**

Express sử dụng kiến trúc dựa trên middleware, cho phép bạn xây dựng các ứng dụng bằng cách kết hợp các middleware khác nhau để xử lý yêu cầu HTTP.

* **Hỗ trợ đa nền tảng:**

Express có khả năng hoạt động trên nhiều nền tảng, đảm bảo tính tương thích trên nhiều môi trường.

* **Hỗ trợ đa dạng về định dạng phản hồi:**

Express có khả năng trả về dữ liệu ở nhiều định dạng khác nhau, bao gồm HTML, JSON, XML, v.v.

* **Hỗ trợ đa kiểu route:**

Express hỗ trợ định nghĩa các tuyến đường (routes) để xử lý các yêu cầu từ trình duyệt và cung cấp phản hồi tương ứng.

* **Hỗ trợ template engine:**

Express hỗ trợ sử dụng các template engine như EJS, Pug, Handlebars để tạo ra các trang HTML động.

* **Môi trường phát triển mạnh mẽ:**

Express có một cộng đồng phát triển lớn, đi kèm với nhiều middleware và thư viện hữu ích.

* **Hỗ trợ các tính năng mở rộng như WebSocket:**

Mặc dù Express tập trung chủ yếu vào xử lý yêu cầu HTTP, nhưng nó có thể được mở rộng để hỗ trợ các tính năng mở rộng khác như WebSocket thông qua các middleware.

**NodeJS**

Node.js là một môi trường chạy mã JavaScript phía máy chủ (backend) được xây dựng trên JavaScript Engine V8 của Google Chrome. Nó cho phép bạn thực thi mã JavaScript ngoài môi trường trình duyệt, cho phép phát triển ứng dụng web và các ứng dụng máy chủ đa nền tảng.

**Đặc điểm nổi bật:**

* **JavaScript Everywhere:**

Node.js cho phép sử dụng JavaScript để xây dựng cả phía máy chủ (backend) và phía máy khách (frontend), giúp duy trì sự nhất quán trong mã nguồn.

* **Mô hình Non-blocking I/O:**

Node.js sử dụng mô hình không chặn I/O (non-blocking I/O) cho phép xử lý đa nhiệm hiệu quả, đồng thời xử lý hàng loạt yêu cầu mà không cần tạo ra các luồng riêng biệt.

* **Có tính sẵn có (Lightweight):**

Node.js có quy mô nhỏ gọn và tài nguyên tiêu tốn thấp, giúp tối ưu hóa hiệu suất và giảm độ trễ.

* **Hỗ trợ cộng đồng lớn:**

Node.js có một cộng đồng phát triển lớn, đi kèm với nhiều thư viện, module, và tài liệu phong phú.

* **Hỗ trợ package manager npm:**

Node.js đi kèm với npm (Node Package Manager), giúp quản lý các thư viện và module của bên thứ ba một cách dễ dàng.

* **Hỗ trợ nhiều môi trường và nền tảng:**

Node.js có thể chạy trên nhiều nền tảng khác nhau bao gồm Windows, macOS, Linux, và nhiều nền tảng khác.

* **Phù hợp với ứng dụng thời gian thực (Real-time Applications):**

Node.js thích hợp cho việc phát triển các ứng dụng thời gian thực như ứng dụng trò chơi, ứng dụng chat, ứng dụng trực tuyến, v.v.

* **Hỗ trợ các framework và thư viện phong phú:**

Node.js có nhiều framework và thư viện mạnh mẽ như Express.js, Socket.io, Mongoose, và nhiều công cụ hữu ích khác.

1. Công cụ sử dụng
2. Công cụ vẽ sơ đồ trực tuyến draw.io

**Khái niệm**

Draw.io là một ứng dụng web vẽ sơ đồ trực tuyến (diagramming tool) giúp người dùng tạo ra các biểu đồ, sơ đồ, và các hình vẽ khác một cách dễ dàng và linh hoạt. Đây là một công cụ hữu ích trong việc trình bày thông tin, thiết kế đồ họa, và làm việc với các tài liệu liên quan đến hình ảnh.

Draw.io hỗ trợ nhiều loại biểu đồ UML bao gồm biểu đồ lớp, biểu đồ tuần tự, biểu đồ trạng thái, và nhiều loại khác giúp việc phân tích nghiệp vụ cho website một cách hiệu quả hơn.

**Các loại biểu đồ phổ biến**

**Biểu đồ tuần tự**

Biểu đồ tuần tự (Sequence Diagram) là một loại biểu đồ trong Unified Modeling Language (UML) được sử dụng để mô tả các tương tác giữa các đối tượng trong một hệ thống theo thời gian. Biểu đồ tuần tự thể hiện các sự kiện và thông điệp được gửi giữa các đối tượng theo thứ tự thời gian.

**Các thành phần chính của một biểu đồ tuần tự bao gồm:**

* Các Đối Tượng (Objects):

Được biểu thị dưới dạng các hộp hoặc phân đoạn. Mỗi đối tượng tham gia vào tương tác sẽ có một hộp hoặc phân đoạn riêng.

* Thời Gian (Lifeline):

Được biểu thị bằng các đường dọc dọc, thể hiện thời gian tồn tại của đối tượng trong biểu đồ.

* Tin Nhắn (Message):

Được biểu thị bằng các mũi tên nối giữa các đối tượng, thể hiện việc gửi thông điệp hoặc sự kiện từ một đối tượng đến một đối tượng khác.

* Sự Kích Hoạt (Activation Bar):

Được biểu thị bằng các vùng nằm ngang trên các đối tượng hoặc thời gian, thể hiện thời gian mà đối tượng hoạt động.

* Cuộc Gọi Phương Thức (Method Call):

Biểu thị việc gọi một phương thức từ một đối tượng đến một đối tượng khác.

* Sự Kết Thúc (Return Message):

Biểu thị sự trả lại kết quả từ một cuộc gọi phương thức.

Biểu đồ tuần tự giúp hiểu rõ hơn về cách các đối tượng tương tác với nhau và thứ tự xảy ra của các sự kiện trong hệ thống. Đây là một công cụ hữu ích trong phân tích và thiết kế hệ thống phần mềm.

**Biểu đồ hoạt động**

Biểu đồ hoạt động (Activity Diagram) là một loại biểu đồ trong Unified Modeling Language (UML) được sử dụng để mô tả các quy trình, hoạt động, và tương tác giữa các thành phần trong hệ thống.

**Các thành phần chính của biểu đồ hoạt động:**

* Hoạt Động (Activity):

Được biểu thị bằng các hình oval và thể hiện một công việc hoặc tác vụ cần thực hiện.

* Kết Nối (Control Flow):

Được biểu thị bằng các mũi tên nối giữa các hoạt động, thể hiện thứ tự thực hiện các công việc.

* Trạng Thái Bắt Đầu (Initial State):

Được biểu thị bằng một hình tròn không viền, thể hiện trạng thái ban đầu của quy trình.

* Trạng Thái Kết Thúc (Final State):

Được biểu thị bằng một hình tròn có nền đậm hoặc có biểu tượng kết thúc, thể hiện trạng thái kết thúc của quy trình.

* Ngắt Tác Động (Fork and Join):

Được sử dụng để chia ra hoặc kết hợp các luồng điều khiển.

* Hình Bình Thường (Object Node):

Được sử dụng để biểu thị đối tượng hoặc dữ liệu tham gia vào hoạt động.

* Bản Cục (Partition):

Được sử dụng để nhóm các hoạt động và đối tượng vào các phần khác nhau.

Biểu đồ hoạt động giúp hiểu rõ hơn về quy trình và tương tác giữa các thành phần trong hệ thống. Đây là một công cụ hữu ích trong phân tích và thiết kế hệ thống phần mềm.

**Biểu đồ usecase**

Biểu đồ Use Case (Use Case Diagram) là một loại biểu đồ trong Unified Modeling Language (UML) được sử dụng để mô tả các tác vụ, chức năng và tương tác giữa các thành phần trong hệ thống từ góc độ của người sử dụng cuối.

**Biểu đồ Use Case bao gồm các phần chính sau:**

* Use Case:

Use case thể hiện một tác vụ hoặc chức năng mà hệ thống cung cấp cho người sử dụng. Nó được biểu thị bằng một hình oval và thường được ghi tên mô tả tác vụ.

* Actor:

Actor đại diện cho một người hoặc hệ thống bên ngoài tương tác với hệ thống. Actor được biểu thị bằng một hình người hoặc biểu tượng khác.

* Association:

Một mũi tên nối từ một actor đến một use case thể hiện mối quan hệ tương tác giữa actor và use case.

* Include:

Include được sử dụng để chỉ ra rằng một use case bao gồm các hoạt động của một use case khác.

* Extend:

Extend được sử dụng để chỉ ra rằng một use case có thể mở rộng bằng cách thêm các hoạt động bổ sung.

Biểu đồ Use Case giúp người phân tích và thiết kế hệ thống hiểu rõ hơn về các chức năng mà hệ thống cung cấp và tương tác giữa người sử dụng và hệ thống. Nó tập trung vào góc độ của người dùng cuối mà không đi sâu vào các chi tiết kỹ thuật.

**Biểu đồ lớp**

Biểu đồ lớp (Class Diagram) là một loại biểu đồ trong Unified Modeling Language (UML) được sử dụng để mô hình hóa cấu trúc của hệ thống bằng cách biểu thị các lớp, giao diện, các mối quan hệ giữa chúng, và các thuộc tính và phương thức của từng lớp.

**Các thành phần chính của biểu đồ lớp bao gồm:**

* Lớp (Class):

Biểu thị một mô hình, một đối tượng, hoặc một khái niệm trong hệ thống. Mỗi lớp được biểu thị bằng một hình chữ nhật chứa tên của lớp.

* Thuộc Tính (Attribute):

Biểu thị các đặc điểm hoặc thông tin của một lớp. Thuộc tính được liệt kê dưới dạng tên: kiểu dữ liệu.

* Phương Thức (Method):

Biểu thị các hành động hoặc chức năng mà một lớp có thể thực hiện. Phương thức được liệt kê dưới dạng tên(phần tham số): kiểu dữ liệu trả về.

* Mối Quan Hệ (Relationship):

Biểu thị các mối quan hệ giữa các lớp hoặc các thành phần khác trong hệ thống. Có nhiều loại mối quan hệ như kế thừa, gắn kết, gồm nhiều, chứa nhiều, và nhiều loại khác.

* Giao Diện (Interface):

Biểu thị một tập hợp các phương thức mà các lớp có thể triển khai.

* Abstract Class (Lớp Trừu Tượng):

Biểu thị một lớp mà không thể tạo một đối tượng trực tiếp từ lớp đó. Thường được sử dụng như một bản mẫu cho các lớp con.

* Gói (Package):

Biểu thị một nhóm các lớp hoặc giao diện có liên quan đến nhau.

Biểu đồ lớp giúp mô tả cấu trúc tĩnh của hệ thống, cho phép người thiết kế hiểu cách các phần tử trong hệ thống tương tác và hợp tác với nhau. Nó là một công cụ hữu ích trong quá trình phân tích và thiết kế hệ thống phần mềm.

1. Phần mềm quản lý, lưu trữ source code Github

**Khái niệm**

GitHub là một nền tảng phát triển phần mềm dựa trên nền tảng web, cung cấp dịch vụ lưu trữ mã nguồn và quản lý dự án cho các nhóm phát triển phần mềm

**Đặc điểm nổi bật của github:**

* Lưu Trữ Mã Nguồn:

GitHub cung cấp một nền tảng mạnh mẽ cho việc lưu trữ mã nguồn dự án phát triển phần mềm. Người dùng có thể tạo và quản lý kho lưu trữ (repositories) cho các dự án của mình.

* Hệ Thống Quản Lý Phiên Bản (Version Control):

GitHub hỗ trợ hệ thống quản lý phiên bản Git, giúp theo dõi sự thay đổi của mã nguồn, quản lý các nhánh (branches), và hợp nhất (merge) các thay đổi.

* Cộng Đồng Lớn:

GitHub có một cộng đồng lớn, thu hút hàng triệu nhà phát triển và người dùng trên toàn thế giới. Điều này tạo điều kiện thuận lợi cho việc chia sẻ, học hỏi và hợp tác trong cộng đồng.

* Các Công Cụ Hợp Tác:

GitHub cung cấp các tính năng hợp tác cho các dự án, bao gồm quản lý nhiều người dùng, theo dõi lỗi (issue tracking), bình luận, và các tính năng khác.

* Tích Hợp Công Cụ Phát Triển:

GitHub tích hợp tốt với nhiều công cụ phát triển phổ biến như Visual Studio Code, Git, Jenkins, và nhiều công cụ khác.

* Hosting Nền Tảng Web Tĩnh (Static Web Hosting):

GitHub cung cấp chức năng hosting nền tảng web tĩnh thông qua GitHub Pages, cho phép người dùng triển khai trang web từ kho lưu trữ của họ.

* Bảo Mật Và Quyền Truy Cập:

GitHub cung cấp các tính năng bảo mật như xác thực hai yếu tố (2FA), quyền truy cập cấp độ người dùng và kiểm soát truy cập.

* Phân Nhánh (Branching) Và Gộp Nhánh (Merging):

GitHub cho phép nhà phát triển tạo ra các nhánh để phát triển các tính năng riêng biệt và sau đó hợp nhất (merge) chúng trở lại nhánh chính.

1. Trình soạn thảo mã nguồn Visual studio code

**Khái niệm**

Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn mở và miễn phí phát triển bởi Microsoft. Nó là một ứng dụng mạnh mẽ được thiết kế để hỗ trợ việc phát triển ứng dụng và mã nguồn trên nhiều ngôn ngữ lập trình và nền tảng khác nhau.

**Đặc điểm nổi bật của VS Code:**

* Hỗ Trợ Nhiều Ngôn Ngữ Lập Trình:

VS Code hỗ trợ nhiều ngôn ngữ lập trình phổ biến như JavaScript, TypeScript, Python, C++, C#, Java, HTML, CSS, và nhiều ngôn ngữ khác.

* Mã Nguồn Mở và Miễn Phí:

VS Code là một ứng dụng mã nguồn mở, đi kèm với giấy phép MIT và hoàn toàn miễn phí sử dụng.

* Giao Diện Người Dùng Đơn Giản và Linh Hoạt:

Giao diện của VS Code được thiết kế đơn giản, dễ sử dụng và linh hoạt. Người dùng có thể tùy chỉnh giao diện và các thành phần theo sở thích cá nhân.

* Hỗ Trợ Debugging:

VS Code cung cấp các tính năng mạnh mẽ để hỗ trợ quá trình debugging, bao gồm chế độ Debug, breakpoint, theo dõi biến, và nhiều tính năng khác.

* Hỗ Trợ Git:

VS Code tích hợp sâu với hệ thống quản lý phiên bản Git, cho phép người dùng theo dõi sự thay đổi trong mã nguồn, thực hiện commit, pull, push và nhiều hoạt động khác.

* Cài Đặt Mở Rộng (Extensions):

VS Code hỗ trợ cài đặt các tiện ích mở rộng (extensions) để mở rộng chức năng và tùy chỉnh trải nghiệm phát triển.

* Tích Hợp Terminal:

VS Code có tích hợp cửa sổ terminal trong giao diện, giúp người dùng thực hiện các lệnh liên quan đến mã nguồn mà không cần chuyển đổi sang ứng dụng khác.

* Tích Hợp IntelliSense:

IntelliSense cung cấp gợi ý tự động, thông báo lỗi, và trợ giúp về cú pháp cho ngôn ngữ lập trình được sử dụng.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ