

**TRƯỜNG ĐẠI HỌC SÀI GÒN**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI**

**TÊN ĐỀ TÀI: TÌM HIỂU VỀ NODEJS**

**GIẢNG VIÊN HƯỚNG DẪN: THS. PHẠM THI VƯƠNG**

**Thành viên nhóm:**

Nguyễn Ngọc Minh – 3118410269

Trần Thanh Phong – 3118410332

Đặng Thị Kiều Oanh – 3118410320

Nguyễn Minh Tin – 3118410434

Trương Trọng Quyền – 3118410361

**TP. HCM tháng .../2022**

## LỜI CẢM ƠN

Nhóm chúng em xin được bày tỏ sự trân trọng và lòng biết ơn đối với thầy Ths. Phạm Thi Vương, giảng viên khoa Công nghệ thông tin – Trường Đại học Sài Gòn. Trong suốt thời gian học và làm đồ án, thầy đã dành rất nhiều thời gian quý báu để chỉ bảo, hướng dẫn, định hướng cho chúng em trong việc tự học, tìm hiểu và thực hiện đồ án. Nhóm chúng em xin được cảm ơn thầy đã giảng dạy chúng em trong quá trình học tập, thực hành, làm bài tập, đọc và nhận xét đồ án của cả nhóm, giúp cả nhóm hiểu bài hơn, chỉ cho nhóm những hạn chế mà cả nhóm cần khắc phục trong học tập cũng như mai sau đi làm. Chúng em xin chân thành cảm ơn thầy!

[illegible]

Giảng viên hướng dẫn

3

# LỜI MỞ ĐẦU

Hiện nay, công nghệ thông tin xuất hiện ở mọi nơi, sự phát triển nhanh chóng của nó kéo theo việc ngành lập trình trở thành một lựa chọn lý tưởng cho các bạn trẻ. Và khi nhắc tới lập trình, ta có thể nói tới Javascript. Chính vì thế nên Javascript ngày càng trở nên phổ biến hơn với nhiều tính năng và các thư viện được hỗ trợ cho developer, điều đó khiến cho các giao diện web càng trở nên sinh động hơn. Mọi thứ mà chúng ta có thể làm được trên web ngày nay là Javascript có thể chạy được trên server, cũng như chạy được trên browser, điều này là khó tưởng tượng trong những năm trở lại đây, hoặc nó chỉ đóng gói trong môi trường sandboxed như Flash hoặc JavaApplets. Thật vậy, rõ ràng rằng từ xưa đến nay chúng ta vẫn quan niệm rằng lập trình bên phía server chỉ dùng được những ngôn ngữ như php, ruby, ... nhưng từ khi NodeJS ra đời nó mang đến tư tưởng mới cho việc lập trình cả bên phía server cũng như bên phía client. Và đối với NodeJS thì chúng ta biến những điều mà trước kia chỉ có thể thao tác được bên phía client thì nay cũng thao tác và xử lý được trên server và ngược lại.

NodeJS được viết bằng ngôn ngữ javascript, nó là một trình biên đóng gói của Google's V8 JavaScript engine, libuv platform abstraction layer và một thư viện lõi được viết bằng Javascript. Mục tiêu của Node js là làm cho web có khả năng push như trong một số ứng dụng gmail. NodeJS cung cấp công cụ giúp lập trình viên có thể làm việc trong non-blocking, mô hình I/O. Sau hơn 20 năm nghiên cứu, xây dựng và phát triển, nhóm kỹ sư đã cho ra đời sản phẩm ứng dụng web node js chạy thời gian thực và kết nối 2 chiều client và server, cho phép trao đổi dữ liệu một cách tự do. NodeJS được InfoWorld bình chọn là "Công nghệ của năm" năm 2012.

# MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU CHUNG .....	7
1.1 Tổng quan về NodeJS .....	7
1.2 Lịch sử hình thành và phát triển.....	7
1.3 Đặc điểm của NodeJS .....	8
1.4 Tại sao nên chọn NodeJS? .....	9
1.4.1 Tất cả đều viết bằng Javascripts .....	9
1.4.2 Khả năng mở rộng .....	9
1.4.3 Sẵn sàng cho thể hệ web thời gian thực.....	10
1.4.4 Hiệu năng.....	10
1.4.5 Tận dụng tối đa phần cứng .....	10
1.4.6 Cộng đồng đông đảo.....	10
1.4.7 Tìm kiếm ứng viên phát triển .....	10
1.4.8 Có nhiều sự lựa chọn cho máy chủ .....	10
1.5 So sánh NodeJS so với các Framework khác. ....	11
1.5.1 NodeJS với Laravel .....	11
1.5.2 NodeJS với Django.....	14
1.6 Các công ty đang tuyển và phát triển về NodeJS.....	16
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	20
2.1 Kiến trúc NodeJS và cách cài đặt NodeJS.....	20
2.4.1 Kiến trúc NodeJS.....	20
2.4.2 Cách cài đặt NodeJS .....	25
2.2 Trình quản lý Node (NPM).....	26
2.3 Module HTTP .....	27
2.4 Framework .....	27
2.4.1 Express.....	27
2.4.2 SocketIO .....	28
2.5 Ứng dụng đầu tiên.....	28
2.6 Route trong NodeJS .....	31
2.6.1 Basic Route .....	32
2.6.2 Route Parameters .....	33
2.6.3 Tách route ra file riêng .....	33
2.7 Kết nối cơ sở dữ liệu trong NodeJS.....	34
2.7.1 MySQL .....	35

2.7.2	Mongodb.....	36
2.7.3	SQL Server .....	38
2.8	Query Builder.....	39
2.8.1	Join bảng .....	39
2.8.2	Insert dữ liệu .....	40
2.8.3	Update dữ liệu.....	41
2.8.4	Delete dữ liệu.....	41
2.9	Kiến trúc REST.....	42
2.9.1	Phương thức HTTP được sử dụng trong REST.....	42
2.9.2	RESTful Web Service.....	45
2.9.3	Tạo RESTful cho một thư viện.....	46
2.9.4	Liệt kê các User .....	46
2.9.5	Thêm User mới .....	47
2.9.6	Hiển thị thông tin User .....	48
2.9.7	Xóa User .....	49
CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG VỚI NODEJS.....		51
3.1	Giới thiệu chung.....	51
3.2	Cấu trúc.....	51
3.1	Front-end .....	51
3.2	Back-end.....	52
3.3	Giao diện website.....	55
3.1	Trang chính.....	55
3.2	Trang đăng nhập .....	55
3.3	Trang đăng kí.....	56
3.4	Trang tìm kiếm .....	56
3.5	Trang thêm bài viết.....	56
3.6	Trang chỉnh sửa .....	57
3.7	Xem những bài đăng của mình hoặc người khác .....	58
CHƯƠNG 4: TỔNG KẾT.....		59
4.1	Đánh giá.....	59
4.1.1	Ưu điểm .....	59
4.1.2	Nhược điểm .....	59
4.2	Kết luận.....	60
TÀI LIỆU THAM KHẢO .....		61

# CHƯƠNG 1: GIỚI THIỆU CHUNG

## 1.1 Tổng quan về NodeJS

NodeJS là một mã nguồn mở, đa nền tảng, chạy trên môi trường JavaScript, được xây dựng trên V8 JavaScript engine của Chrome - V8 thực thi mã JavaScript bên ngoài trình duyệt. Nó được tạo ra vào năm 2009 đi kèm với một lợi thế chính - NodeJS cho phép thực hiện lập trình bất đồng bộ. Node.js bao gồm có V8 JavaScript engine của Google, libUV, và vài thư viện khác.

Ở chế độ đồng bộ thực thi từng dòng và tiến hành thực thi dòng tiếp theo khi dòng hiện tại đã thực thi xong.

Khi bất đồng bộ thực thi tất cả dòng code cùng một lúc.

NodeJS là một nền tảng được xây dựng trên JavaScript runtime của Chrome với mục đích xây dựng các ứng dụng mạng nhanh chóng và có thể mở rộng được một cách dễ dàng hơn. NodeJS sử dụng mô hình I/O lập trình theo sự kiện, non-blocking, do đó NodeJS khá gọn nhẹ và hiệu quả - công cụ hoàn hảo cho các ứng dụng chuyên sâu về dữ liệu theo thời gian thực chạy trên các thiết bị phân tán.

NodeJS là môi trường runtime mã nguồn mở đa nền tảng, được sử dụng để phát triển các ứng dụng mạng và ứng dụng server-side. Các ứng dụng NodeJS được viết bằng JavaScript và có thể chạy trong NodeJS runtime trên OS X, Microsoft Windows và Linux.

NodeJS cũng cung cấp một thư viện bao gồm rất nhiều các module JavaScript khác nhau nhằm đơn giản hóa việc phát triển các ứng dụng web, qua đó giảm thiểu tình trạng sử dụng quá nhiều NodeJS.

## 1.2 Lịch sử hình thành và phát triển

NodeJS ra đời vào năm 2009 bởi Ryan Dahl và những lập trình viên khác làm việc tại Joyent. NodeJS được tạo lần đầu cho hệ điều hành Linux sử dụng. Nó được phát triển vào bảo trì bởi Ryan Dahl và được tài trợ bởi Joyent.

Trong năm 2011, một bộ phận package manager đã giới thiệu bộ thư viện cho NodeJS gọi là npm. Tháng 6 năm 2011, Microsoft hợp tác với Joyent để tạo ra bản cho Windows.

Tháng 12, do xung đột nội bộ nên NodeJS bị chia rẽ, IO.js được hình thành Node.js được *InfoWorld* bình chọn là “Công nghệ của năm” năm 2012.

Để bắt đầu dùng NodeJS, bạn phải hiểu sự khác nhau giữa NodeJS với các môi trường truyền thống chạy trên server (server side) phổ biến như PHP, Python, Ruby, etc...

### 1.3 Đặc điểm của NodeJS

NodeJS có nhiều đặc điểm nổi bật, vượt trội. Nắm được các đặc điểm này sẽ giúp bạn hiểu sâu hơn về NodeJS, đồng thời đưa ra quyết định sử dụng nền tảng này chính xác.

*Bất đồng bộ và phát sinh sự kiện (Non-blocking and Event Driven):* Tất cả các APIs của thư viện NodeJS đều bất đồng bộ (non-blocking), NodeJS không cần đợi một API trả về dữ liệu. Server chuyển sang một API khác sau khi gọi nó và có cơ chế riêng để gửi thông báo và nhận phản hồi về các hoạt động của NodeJS và API đã gọi.

*Tốc độ nhanh:* Phần core phía dưới được viết gần như toàn bộ bằng C++ kết hợp Chrome V8 Engine nên tốc độ xử lý công việc của NodeJS cực nhanh, nhưng vẫn đảm bảo được tính chuẩn xác.

*Đơn giản – Hiệu năng cao:* NodeJS sử dụng một mô hình luồng đơn luồng (single thread) và các sự kiện lặp (event-loop). Cơ chế sự kiện cho phép phía Server trả về phản hồi theo non-blocking, đồng thời tăng hiệu quả sử dụng. Các luồng đơn cung cấp dịch vụ cho nhiều request hơn hẳn Server truyền thống.

*Không lưu bộ nhớ đệm (non buffer):* Nền tảng NodeJS không có vùng nhớ đệm, tức không cung cấp khả năng lưu trữ dữ liệu buffer.



NodeJS nền tảng phát triển ứng dụng mạnh mẽ, NodeJS có thể đáp ứng mọi nhu cầu lập trình, phát triển ứng dụng thông qua javascript, là một ngôn ngữ thông dụng hiện nay. Các chuyên gia trong lĩnh vực lập trình, công nghệ khuyên dùng NodeJS khi phát triển các ứng dụng như Websocket server, Fast File Upload Client, Ad Server, Cloud Services, Microservice, RESTful API, Any Real-time Data Application, ...

## **1.4 Tại sao nên chọn NodeJS?**

NodeJS được viết bằng JavaScript với cộng đồng người dùng lớn mạnh. Nếu bạn cần hỗ trợ gì về NodeJS, sẽ nhanh chóng có người hỗ trợ bạn. Theo tác giả của ngôn ngữ Javascript, Ryan Dahl: “Javascript có những đặc tính mà làm cho nó rất khác biệt so với các ngôn ngữ lập trình động còn lại, cụ thể là nó không có khái niệm về đa luồng, tất cả là đơn luồng và hướng sự kiện.”

Có thể chạy ứng dụng Nodejs ở bất kỳ đâu trên máy Mac – Window – Linux, hơn nữa cộng đồng Nodejs rất lớn và hoàn toàn miễn phí.

### **1.4.1 Tất cả đều viết bằng Javascripts**

Ngôn ngữ Javascript không còn lạ lẫm gì đến với dân lập trình, đặc biệt là lập trình web. Javascript được tạo ra để lập trình client-side. Giờ Javascript có thể viết cho Server thì còn gì bằng nữa. Xử lý front-end, backend đều sử dụng một ngôn ngữ.

### **1.4.2 Khả năng mở rộng**

Việc thiết kế một ứng dụng có khả năng mở rộng là một thách thức lớn đối với các công nghệ lập trình.

Tốc độ xây dựng và mở rộng một ứng dụng Node.JS gây ấn tượng mạnh. Một trong những bí mật để Node.JS có thể mở rộng được đó là Event Loop. Kiến trúc này của Node.JS khiến nó đáp ứng các yêu cầu từ client hoàn toàn khác so với các công nghệ web server khác. Do đó bộ nhớ node.js sử dụng cho từng request cũng

nhỏ hơn nhiều và nó có thể đáp ứng được số lượng client nhiều hơn hàng trăm lần so với ngôn ngữ khác.

### **1.4.3 Sẵn sàng cho thế hệ web thời gian thực**

Công nghệ đang phát triển chóng mặt. Các công nghệ theo thời gian thực đang dần phát triển. Các công nghệ như WebSocket đã làm cho ứng dụng web gần như không có thời gian trễ. NodeJS có thể xử lý một lượng kết nối khổng lồ.

### **1.4.4 Hiệu năng**

NodeJS sử dụng engine V8 của Google, cùng với non-blocking IO thì việc tạo một trang web chậm chạp như bây giờ gần như không thể.

### **1.4.5 Tận dụng tối đa phần cứng**

Là một lập trình web, chẳng ai không biết sự tồn kém chính là host (phần cứng). Nhưng giờ NodeJS đã cải thiện được điều đó. Hơn thế nữa là việc NodeJS có thể phát triển theo chiều ngang chỉ cần cho một Load Balancer đứng phía trước là được rồi.

### **1.4.6 Cộng đồng đông đảo**

Cộng đồng NodeJS được phát triển rất nhanh, có nhiều webmaster, diễn đàn hình thành để hỗ trợ. Chỉ cần lập trình viên biết NodeJS và biết tìm kiếm là được

### **1.4.7 Tìm kiếm ứng viên phát triển**

Bạn là một doanh nghiệp cần tìm lập trình viên NodeJS thì đơn giản chỉ cần tìm người biết Javascript và biết tìm kiếm là được.

### **1.4.8 Có nhiều sự lựa chọn cho máy chủ**

Vì ít tốn tài nguyên, cho nên việc chọn máy chủ phù hợp là cực dễ. Miễn máy chủ nào hỗ trợ Apache hoặc IIS là được. Mà những loại máy chủ này thì giờ không thiếu.

## 1.5 So sánh NodeJS so với các Framework khác.

### 1.5.1 NodeJS với Laravel

Laravel	NodeJS
Là một PHP MVC Framework	Đây là thời gian chạy JavaScript dựa trên Công cụ chạy JavaScript của Google Chrome.
Hơi tụt hậu trong các phân khúc thị phần so với NodeJS.	Dẫn đầu thị trường về công nghệ so với Laravel. Nó nằm trong số 10000 trang web hàng đầu.
Không có mô hình IO vì nó chỉ là một khuôn khổ PHP.	Sử dụng mô hình IO hướng sự kiện, không chặn, giúp nó hoạt động hiệu quả và dễ dàng hơn.
Đang bùng nổ trong các lĩnh vực kinh doanh của sự nghiệp và giáo dục, mua sắm, kinh doanh và 95 ngành công nghiệp khác.	Chủ yếu được sử dụng trong các ngành như nghệ thuật và giải trí, internet và viễn thông, máy tính và điện tử, và công nghiệp ô tô.
Nó tích hợp tốt với thư viện giao diện người dùng.	Không có tính năng tích hợp như vậy trong NodeJS.
Lý tưởng cho các ứng dụng MySQL và Maria DB.	Được tạo ra để hoạt động với Express JS và lý tưởng cho MongoDB / MongooseJS.
Các mẫu của khung Laravel được thiết kế sáng tạo và hiệu quả để xây dựng và cung cấp bộ cục cơ bản với các phần cụ thể.	Không có mẫu tích hợp nào cho NodeJS, tuy nhiên, có thể đạt được nó thông qua một số công cụ của bên thứ ba được phát triển để tích hợp vào NodeJS.

Ở một số quốc gia, bao gồm Ba Lan, Indonesia, Hà Lan, Thổ Nhĩ Kỳ và 118 quốc gia khác, Laravel có lợi thế.	NodeJS dẫn đầu tại Hoa Kỳ, Nga, Brazil, Trung Quốc và 89 quốc gia khác.
Nếu bạn cần một hệ thống quy mô đầy đủ để xử lý một trang web lớn dựa trên CMS, hãy sử dụng Laravel.	Nếu bạn cần một kiến trúc dựa trên dịch vụ nhỏ gọn, hãy sử dụng NodeJS.
Nền tảng Laravel chứa Eloquent ORM, yêu cầu triển khai cơ bản PHP Active Record. Nó cho phép các nhà phát triển ứng dụng web đưa ra các truy vấn cơ sở dữ liệu cú pháp PHP thay vì viết mã SQL.	Có một khả năng trong NodeJS để chia sẻ mã giữa các ứng dụng giao diện người dùng và phụ trợ, hoặc thậm chí để kết xuất thiết bị đồng hình.
Laravel chịu trách nhiệm bảo vệ ứng dụng Web trong hệ thống của riêng mình. Nó yêu cầu mật khẩu được băm và ướp muối, có nghĩa là mật khẩu sẽ không bao giờ lưu trong cơ sở dữ liệu dưới dạng văn bản thuần túy. Điều này cũng sử dụng “Thuật toán băm Bcrypt” để tạo biểu diễn mật khẩu được mã hóa.	Vì nút JS có nguồn mở nên cộng đồng npm cung cấp nhiều gói tuyệt vời.

Cả hai khuôn khổ đều được coi là duy nhất và có những lợi ích và hạn chế riêng của chúng.

Tất cả phụ thuộc vào loại sản phẩm bạn muốn tạo, điều đó quyết định nền tảng nào phù hợp hơn với độ phức tạp của sản phẩm.

Chúng tôi đã nhìn thấy cả hai riêng lẻ và cùng nhau. Vì vậy, bây giờ bạn có thể thuê nhà phát triển node.js hoặc thuê nhà phát triển Laravel tùy thuộc vào yêu cầu dự án của bạn.

*Nói tóm lại:*

- **Chọn Laravel, nếu:**

- + Bạn muốn bố cục trang web tương tác được xây dựng với nội dung hấp dẫn.
- + Tầm nhìn của bạn là tạo ra một ứng dụng nâng cao mà không cần thêm bất kỳ thành phần nào hoặc chi phí bổ sung bằng cách tận dụng Blade Template Engine của Laravel.
- + Bạn muốn tận dụng các tiện ích con khác nhau có khả năng tương thích với JS và CSS để tạo các ứng dụng web tùy chỉnh.
- + Bạn yêu cầu một khuôn khổ được tài liệu hóa xuất sắc với sự hỗ trợ liên tục từ một cộng đồng lớn.
- + Yêu cầu dự án của bạn là xây dựng một ứng dụng web ngày càng nhanh hơn với các thư viện hướng đối tượng.
- + Mục tiêu của bạn là tạo một ứng dụng web thân thiện với SEO và đáng tin cậy với tính bảo mật cao.

- **Chọn Node.js nếu:**

- + Bạn muốn xây dựng một ứng dụng web để truyền phát nội dung.
- + Bạn muốn xây dựng một ứng dụng trang đơn hiệu quả.
- + Bạn muốn xây dựng các ứng dụng web phong phú với khả năng xử lý dữ liệu hiệu quả.

- + Bạn muốn tạo một ứng dụng web nhiều người dùng trong thời gian thực.
- + Bạn muốn tạo các ứng dụng trò chơi dựa trên trình duyệt.

### 1.5.2 NodeJS với Django

	Django	Node.JS
<b>Phát hành lần đầu</b>	2005	2009
<b>Phát triển trên</b>	Python	JavaScript, C, and C++
<b>kiến trúc</b>	Web development framework	JS runtime environment
<b>Khía cạnh quan trọng</b>	MTV model	Event-driven model
<b>Bảo vệ</b>	Django an toàn hơn NodeJS; vì nó có một hệ thống tích hợp, bảo vệ khỏi bất kỳ lỗi bảo mật nào.	NodeJS không được bảo mật như Django, vì nó cần thao tác thủ công trong hệ thống để quản lý các lỗi bảo mật.
<b>Hiệu suất</b>	Django cho hiệu suất tốt hơn, nhờ vào hệ thống mẫu nội bộ được tích hợp sẵn để thúc đẩy việc thực hiện nhiệm vụ quan trọng kịp thời.	Hiệu suất của NodeJS cũng tốt, vì nó cho phép các chuyên gia web tự do hơn khi triển khai. Nhưng điều này làm tăng thời gian phát triển tổng thể cần thiết để tạo ra một ứng dụng.
<b>Sự phức tạp</b>	Nó phức tạp hơn NodeJS. Ở đây, một nhà phát triển phải tuân theo một lộ trình cụ thể để giải quyết bất kỳ vấn đề nào.	Nó ít phức tạp hơn Django. Ở đây, một nhà phát triển được phép giải quyết bất kỳ vấn đề nào theo ý muốn của họ.
<b>Hiệu quả về chi</b>	Nó năng động hơn và cho	Nó dễ học hơn nhiều nhưng

<b>phí</b>	tốc độ nhanh, giúp tiết kiệm chi phí hơn.	hấp thụ nhiều thời gian hoạt động hơn, làm cho nó trở thành một giải pháp thay thế ít hiệu quả hơn về chi phí.
<b>Danh tiếng</b>	Nó có một danh tiếng đáng kể.	Với sự tăng trưởng ổn định về mức độ phổ biến, NodeJS có thể sớm trở thành một nền tảng được ưa thích.
<b>Cộng đồng</b>	Django có một cộng đồng nhỏ.	NodeJS có một cộng đồng tích cực với những người dùng đủ điều kiện để hỗ trợ bạn về các bản cập nhật và sửa đổi mới nhất.
<b>Cơ hội phát triển Full-stack</b>	Nó không cung cấp phát triển toàn bộ ngăn xếp.	Do khả năng tạo cả hai phần, giao diện người dùng và phần phụ trợ của một ứng dụng, bằng một ngôn ngữ lập trình - JavaScript, NodeJS là một trong những công nghệ được lựa chọn nhiều nhất để phát triển ứng dụng web.
<b>Tính linh hoạt</b>	Nó cung cấp một tính linh hoạt nhất định và có các tính năng phát triển nghiêm trọng.	Với thư viện JavaScript, nhiều công cụ và tính năng khác nhau có sẵn trong NodeJS. Bạn thậm chí có thể tạo một ứng dụng dựa trên JS từ đầu.

Sự thành công của bất kỳ sự phát triển ứng dụng web nào phụ thuộc rất nhiều vào các phương pháp và công cụ mà nhóm của bạn sẽ sử dụng. Cả hai, Node.js và Django web framework, đều là

những công nghệ tiên tiến sẽ mang lại lợi ích cho gần xếp của bạn.

## 1.6 Các công ty đang tuyển và phát triển về NodeJS

### ❖ Hamsa Corporation

700 – 1500 USD

CIC Tower, No 2 Nguyen Thi Due street, Cau Giay, Ha Noi

#### Yêu cầu tối thiểu:

- Từ 6 tháng kinh nghiệm (Junior) sử dụng NodeJS hoặc ReactJS (chỉ cần biết 1 trong 2, biết cả 2 càng tốt)
- Tiếng Anh ở mức độ đọc hiểu cơ bản.
- Ứng viên chưa có nhiều kinh nghiệm sẽ được đào tạo

#### Yêu cầu mong muốn:

- Có kiến thức về JavaScript, OOP, API
- Có kinh nghiệm về Typescript + ReactJs + NextJs là một điểm cộng.
- Có GitHub profile

### ❖ V3Tech

2,060 - 2,460 USD

#28E0, Street (St 292), Phnom Penh, Cambodia.

Working in Cambodia, District 1, Ho Chi Minh

#### Yêu cầu:

- Tiếng Anh trung cấp.
- Tốt nghiệp đại học chuyên ngành CNTT, Khoa học máy tính hoặc các chuyên ngành khác có liên quan.
- 1 năm kinh nghiệm (mới hơn cũng được chấp nhận) trong phát triển ứng dụng Web Front-end và Back-end.
- Phải tinh thông toán học.
- Thông minh với tư duy logic tốt.



- Có kinh nghiệm vững vàng về ReactJS, NodeJS, AngularJS hoặc các ngôn ngữ khác.
- Kinh nghiệm mạnh mẽ về cơ sở dữ liệu (MySQL, Mongo DB)
- Có kinh nghiệm phát triển với API.
- Có kinh nghiệm tốt về công cụ GitLab.
- Kiến thức về React Redux là một điểm cộng.
- Tư duy rất mạnh mẽ trong cấu trúc dữ liệu và thuật toán.
- Có khả năng tham gia đồng thời nhiều công việc và hoàn thành chúng với chất lượng cao.
- Tiếp thu nhanh và ham học hỏi.
- Có thể làm việc dưới áp lực.

❖ **ASIM TELECOM**

500 – 2000 USD

18 Nguyen Van Mai Street, Tan Binh, Ho Chi Minh

5 Dien Bien Phu Street, Ba Dinh, Ha Noi

Yêu cầu:

- Đã tốt nghiệp trường đại học, cao đẳng,... chuyên ngành CNTT hoặc các chuyên ngành liên quan.
- Có kinh nghiệm từ 1 năm trở lên về lập trình NodeJS.
- Có kinh nghiệm làm việc với MySQL, Redis, MongoDB là ưu thế.
- Có khả năng tìm hiểu nghiệp vụ, quy trình; phân tích, thiết kế dữ liệu.
- Có tư duy lập trình tốt, có khả năng làm việc độc lập, sáng tạo trong công việc.
- Ưu tiên ứng viên đã có kinh nghiệm làm việc với Angular.

❖ **Cohost**

900 - 2000 USD

17 Hang Chuoi Street, Tang Bach Ho Ward, Hai Ba Trung, Ha Noi

Yêu cầu bắt buộc:

- Có kinh nghiệm phát triển ứng dụng thời gian thực
- Thành thạo ít nhất một ngôn ngữ Python, NodeJS
- Có hiểu biết về kiến trúc Micro-services
- Nắm vững về RESTful APIs và giao tiếp giữa các API
- Có kinh nghiệm làm việc với Redis, Elasticsearch, Mysql, Kafka

*Yêu cầu ưu tiên:*

- Nắm vững về GraphQL APIs và giao tiếp giữa các API
- Đã làm việc AWS hoặc Google Cloud là một lợi thế
- Đã từng làm việc với các hệ thống chat là một lợi thế
- Có hiểu biết về AI, Machine Learning
- Có thể lên kế hoạch, thiết kế hệ thống
- Hoà đồng có khả năng làm việc team-work tốt
- Có khả năng tự học và nghiên cứu công nghệ mới

❖ **Oode.com**

2,300 - 2,500 USD

Work from home, District 1, Ho Chi Minh

work from home, Hai Chau, Da Nang

*Yêu cầu:*

- Ít nhất phải có bằng đại học
- Cần có kỹ năng giao tiếp tốt để làm việc với khách hàng và đối tác
- Kiến thức sâu rộng về JavaScript / Nodejs, ngăn xếp web, thư viện và khung công tác
- Kiến thức sâu rộng về các hệ quản trị cơ sở dữ liệu như PostgreSQL, MongoDB
- Hiểu biết thành thạo về GIT
- Kỹ năng phân tích mạnh mẽ và năng khiếu giải quyết vấn đề

– Tiếng anh giao tiếp

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1 Kiến trúc và cách cài đặt NodeJS

#### 2.4.1 Kiến trúc NodeJS

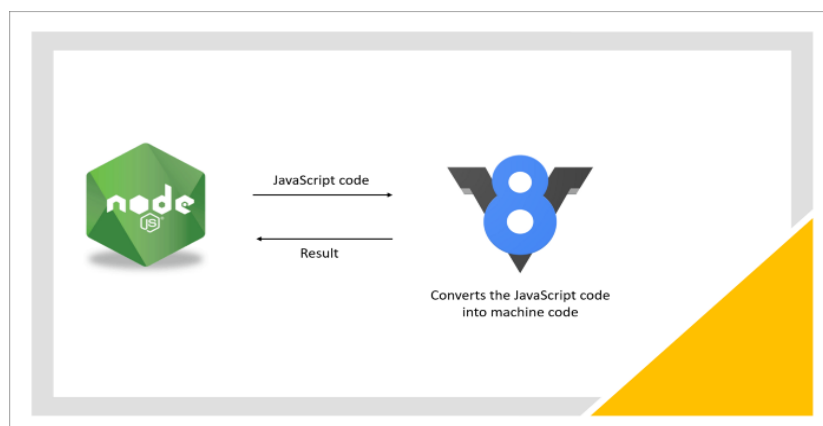
Node.js không chỉ được viết bằng Javascript mà được viết bằng C++ và cả Javascript để có thể hoạt động chính xác. Có rất nhiều thư viện mà Node phụ thuộc vào. Tuy nhiên, V8 và LIBUV là hai dependency quan trọng nhất xử lý phần lớn các hoạt động của node.js.

Hai dependency này cung cấp một lớp trừu tượng và cho phép chúng ta viết mã JavaScript thuần chạy trong Node.js mà vẫn cho phép truy cập vào các tính năng đọc file được triển khai trong LIBUV và các thư viện C++ khác.

Điều đặc biệt ở đây là Node.js liên kết tất cả các thư viện này với nhau, bất kể chúng được viết bằng C++ hay JavaScript, và sau đó cho phép chúng ta truy cập vào các chức năng của chúng bằng JavaScript thuần. Vì vậy, chúng ta tập trung vào code ứng dụng dễ dàng hơn nhiều mà không cần phải xử lý mã C++.

#### ❖ V8

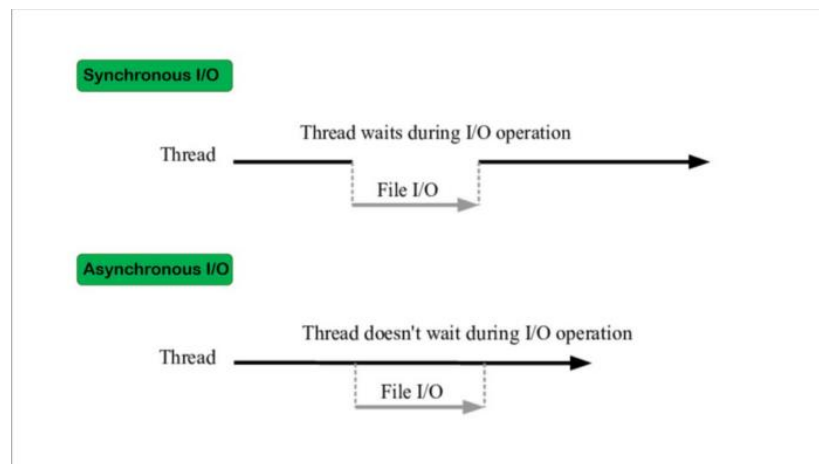
Node.js được xây dựng trên V8 engine của Google. Nó là công cụ javascript nhanh nhất. V8 engine chuyển đổi code javascript thành mã máy mà máy tính thực sự hiểu được. Kết quả sau đó được tạo và trả về node.js. Node.js không thể hiểu mã javascript mà chúng ta viết mà không có V8. Bên cạnh javascript, V8 cũng sử dụng C++.



Nói tóm lại V8 Engine đảm nhận vai trò quản lý và cấp phát Memory cho các ứng dụng NodeJS. Vì vậy bất cứ vấn đề gì liên quan đến Memory, thì gốc vẫn xuất phát từ V8.

#### ❖ Đầu vào-đầu ra không đồng bộ

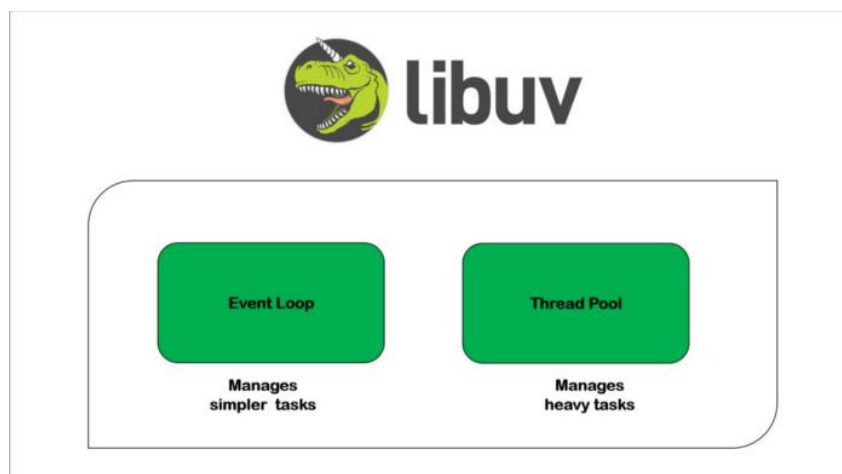
I/O không đồng bộ cho phép các ứng dụng xử lý chồng chéo với các hoạt động I/O. Nói cách đơn giản, mục tiêu là chương trình không bao giờ bị nghẽn lại



Trong I/O đồng bộ, thread - Luồng (một luồng chỉ là một chuỗi các lệnh) sẽ đợi cho đến khi toàn bộ hoạt động kết thúc. Mặt khác trong I/O không đồng bộ, luồng không đợi trong các hoạt động. Thao tác I/O sẽ chạy ở chế độ nền và nó sẽ được gọi khi kết thúc. Tính năng I/O không đồng bộ cho phép ứng dụng có thêm thời gian để thực hiện các xử lý khác trong khi I/O đang diễn ra.

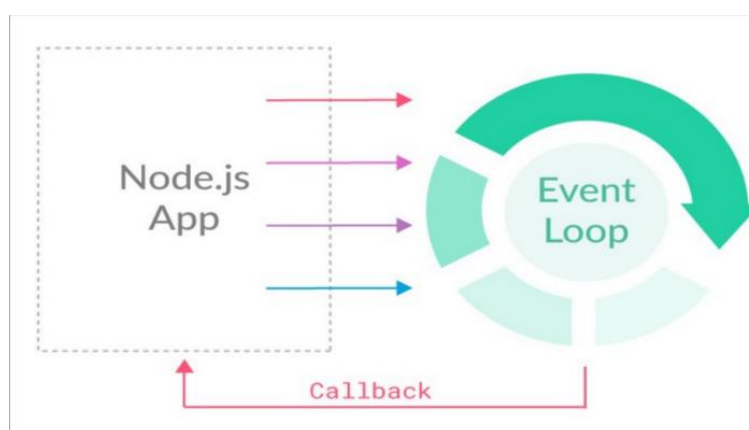
#### ❖ LIBUV

LIBUV là một dependency node.js. Nó cung cấp cho node.js quyền truy cập vào hệ điều hành máy, mạng, hệ thống tệp. LIBUV là một thư viện mã nguồn mở tập trung mạnh vào I/O không đồng bộ (Input-output). LIBUV được viết bằng C++. Ngoài việc tập trung vào I/O không đồng bộ LIBUV còn triển khai hai tính năng quan trọng là event loop và thread pool.



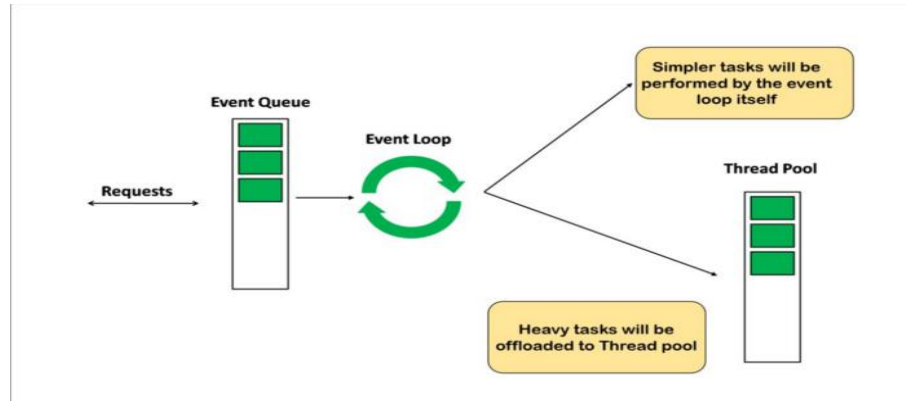
### ❖ Event Loop

Khi chúng ta sử dụng Node.js trên máy tính, có nghĩa là có một tiến trình Node đang chạy trên máy tính đó. Quá trình này chỉ là một chương trình đang được thực thi. Bây giờ trong quá trình đó, Node.js chạy trong một thread (luồng) duy nhất. Một thread về cơ bản chỉ là một chuỗi các hướng dẫn và không cần thiết phải hiểu sâu về thread hoặc quy trình là gì. Chỉ cần tưởng tượng thread là một hộp nơi mã của chúng ta được thực thi trong bộ xử lý của máy tính. Bây giờ, điều cần thiết để hiểu ở đây là thực tế là Node chạy trong single thread, ví dụ, nếu chúng ta có 4 nhiệm vụ khác nhau thì tất cả bốn tác vụ này sẽ xảy ra trong một thread duy nhất.



Bạn có 1 người dùng hay 100 người dùng hoặc có thể 100 triệu người dùng truy cập ứng dụng của bạn cùng một lúc thì vẫn vậy. Event loop được gọi là trung tâm của node.js. Nó thực thi tất cả các hàm gọi

lại (các hàm được gọi ngay khi một số công việc kết thúc) trong một luồng duy nhất và nó cũng giảm tải các tác vụ nặng hoặc tốn kém như nén tệp vào một thread pool. Eventloop giúp lập trình không đồng bộ có thể thực hiện được trong Node.js.



Nó xử lý tất cả các sự kiện đến và thực hiện phân cân bằng bằng cách giảm tải các tác vụ nặng hơn vào Thread Pool và tự thực hiện các tác vụ đơn giản hơn. Event Loop là trái tim của Node js, nó làm cho Node hoàn toàn khác biệt với các ngôn ngữ Back end khác.

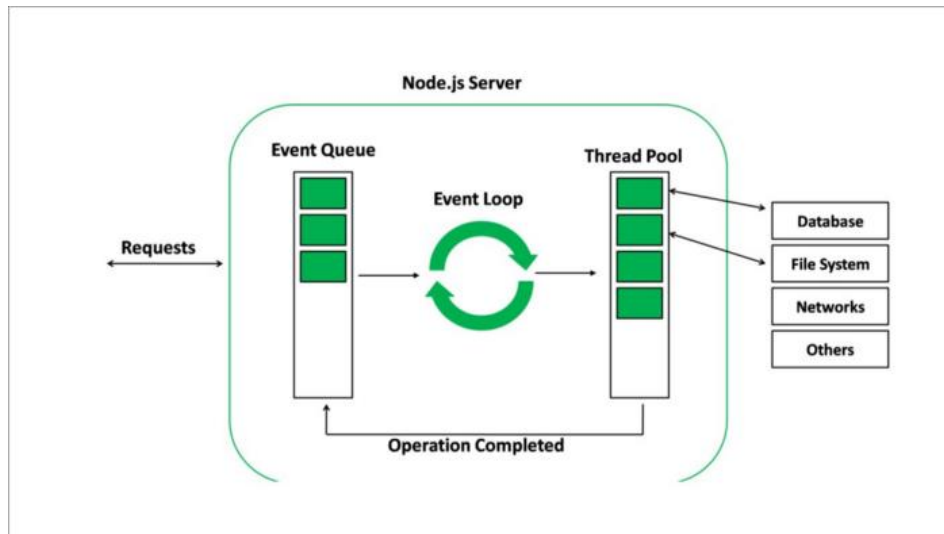
Event Loop chịu trách nhiệm xử lý các tác vụ đơn giản hơn và Thread pool quản lý các tác vụ nặng.

### ❖ Thread pool

Thread pool cung cấp cho chúng ta 4 thread bổ sung hoàn toàn tách biệt với single thread trong Event Loop. Chúng ta có thể cấu hình nó lên đến 128 thread, nhưng thông thường, 4 thread này là đủ. Vì vậy, 4 thread này cùng nhau tạo thành một thread pool. Event loop sẽ tải bớt các tác vụ nặng vào Thread pool và điều này diễn ra tự động.

Các DEV không phải là người quyết định cái gì đi vào Thread pool và cái gì không. Một số tác vụ tốn kém được giảm tải là : tất cả các hoạt động xử lý tệp, mọi thứ liên quan đến mật mã, như mật khẩu lưu vào bộ nhớ đệm, sau đó là tất cả các hoạt động liên quan đến nén, tra cứu DNS (khớp các miền web với địa chỉ IP thực tương ứng của

chúng). Đó là những thứ dễ làm nghẽn luồng chính nhất. Vì vậy, Node sẽ xử lý nó bằng cách tự động tải chúng vào Thread pool.



Ví dụ: Ở đây khi yêu cầu truy cập vào máy chủ Node, nó sẽ chuyển đến Event loop thông qua hàng đợi. Bây giờ Event loop sẽ kiểm tra xem tác vụ có phải là tác vụ nặng như các thao tác xử lý tệp hay các thao tác liên quan đến mạng hay không. Nếu nhiệm vụ nặng, nó sẽ tải nó xuống Thread pool, các thread trong này sẽ thực thi tác vụ riêng biệt. nó sẽ không chặn Event loop của chúng tôi và để Event loop có thể xử lý các tác vụ đơn giản hơn.

#### ❖ Các thư viện quan trọng khác

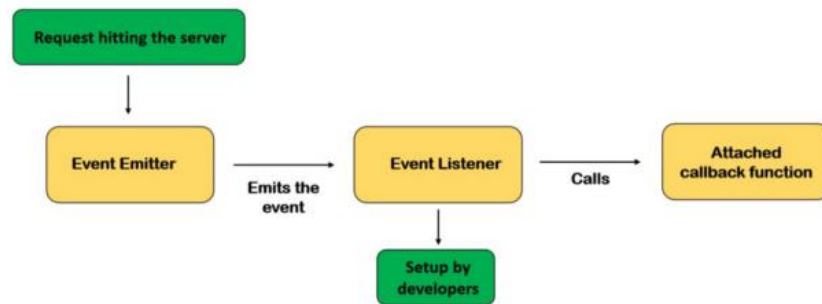
Các thư viện quan trọng nhất cho Node.js là LIBUV và V8. Tuy nhiên, Node không chỉ hoạt động dựa trên V8 và LIBUV mà còn dựa trên một số thư viện khác như HTTP parser for parsing HTTP, C-ARES for DNS queries, OpenSSL for cryptography, and Zlib for file compression. Khi tất cả các thành phần này kết hợp hoàn hảo với nhau, Node.JS sẵn sàng được sử dụng ở phía máy chủ cho tất cả các ứng dụng.

#### ❖ Kiến trúc hướng sự kiện - Event-Driven Architecture

Hầu hết các mô-đun cốt lõi của Node, như HTTP, Hệ thống tệp được xây dựng dựa trên kiến trúc hướng sự kiện. Khái niệm này thực



sự khá đơn giản. Trong Node, có một số đối tượng nhất định được gọi là bộ phát sự kiện phát ra các sự kiện được đặt tên ngay khi có điều gì đó quan trọng xảy ra trong ứng dụng, chẳng hạn như yêu cầu truy cập máy chủ hoặc tệp hoàn tất để đọc. Sau đó, các sự kiện này được chọn bởi các trình nghe sự kiện đã thiết lập, điều này sẽ kích hoạt các chức năng (hàm gọi lại) được gắn với mỗi trình nghe.



Một mặt, chúng ta có các bộ phát sự kiện sẽ phát ra các sự kiện được đặt tên, và mặt kia, chúng ta có các bộ xử lý sự kiện phản ứng với các sự kiện được phát ra bằng cách gọi các hàm gọi lại.

Kiến trúc hướng sự kiện giúp cho việc phản ứng nhiều lần với cùng một sự kiện trở nên dễ dàng hơn. Tất cả những gì chúng ta phải làm là thiết lập nhiều người nghe.

## 2.4.2 Cách cài đặt NodeJS

### ❖ Windows

Đối với windows, bạn chỉ cần vào trang chủ [Nodejs.org](https://nodejs.org) để download và cài đặt Nodejs. Để kiểm tra đã cài đặt được nodejs hay chưa, bạn cần mở chương trình "Node.js Command Prompt" lên bằng cách vào Start gõ search từ "prompt" rồi gõ **node -v**.

```
Command Prompt
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>node -v
v16.14.0

C:\Users\Admin>
```

Nếu xuất ra version của Nodejs tức là bạn đã cài đặt thành công.

Tiếp theo là kiểm tra NPM - Công cụ quản lý package của NodeJS.

***npm -v***

Tương tự nếu xuất ra version của NPM bạn đã cài đặt thành công.

```
Command Prompt
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>node -v
v16.14.0

C:\Users\Admin>npm -v
8.3.1

C:\Users\Admin>
```

## ❖ Linux

Đầu tiên, bạn nên update hệ thống linux đang sử dụng.

***sudo apt-get update***

Cài NodeJS:

***sudo apt-get install nodejs***

Cài đặt NPM:

***sudo apt-get install npm***

Để kiểm tra NPM và NodeJS đã cài đặt được chưa:

***nodejs -v***

***npm -v***

## 2.2 Trình quản lý Node (NPM)

- **NPM** là trình quản lý gói cho các Node.js hoặc các module nếu muốn.
- Chương trình NPM được cài trên máy tính khi cài đặt Node.js

- Có thể sử dụng NPM để cài các gói CDM: `npm install upper-case`.

## 2.3 Module HTTP

HTTP là một module được tích hợp sẵn vào trong Node.js (nên sẽ không cần phải download), module này có nhiệm vụ khởi tạo một cổng kết nối HTTP server trả về client.

Để sử dụng được bất kỳ module nào trong Node.js thì chúng ta cần phải require module đó. Và để require một module trong Node.js chúng ta sử dụng từ khóa **require**.

```
var http = require('http');
```

Tiếp đó để khởi tạo server trong HTTP module chúng ta sử dụng 2 phương thức `createServer().listen()`.

**VD:** khởi tạo một con server chạy port 8000.

```
var http = require('http');  
http.createServer().listen(8000);
```

Bên trong phương thức `createServer` chứa một phương thức ẩn danh có 2 tham số truyền vào.

- + Tham số thứ nhất: là biến chứa các thông số liên quan đến **request** mà người dùng gửi lên.

- + Tham số thứ hai là biến chứa các thông số liên quan đến **response** mà chúng ta muốn trả về client.

## 2.4 Framework

### 2.4.1 Express

ExpressJS là một trong những framework phổ biến dùng để xây dựng API và Website phổ biến nhất của NodeJS. Nó được sử dụng rộng rãi đến mức hầu như mọi dự án Web nào đều bắt đầu bằng việc tích hợp Express. Có rất nhiều lý do để chọn ExpressJS:

- + Có nhiều tính năng hỗ trợ tất cả những gì bạn cần trong việc xây dựng Web và API
- + Quản lý các route dễ dàng

- + Cung cấp một nền tảng phát triển cho các API
- + Hỗ trợ nhiều thư viện và plugin
- + Bảo mật và an toàn hơn so với việc code thuần
- + Hỗ trợ cộng đồng tuyệt vời

### 2.4.2 SocketIO

SocketIO là một web-socket framework có sẵn cho nhiều ngôn ngữ lập trình.

Trong NodeJS, SocketIO cho phép xây dựng một các ứng dụng realtime như chatbot, tickers, dashboard APIs, và nhiều ứng dụng khác. SocketIO có lợi ích hơn so với NodeJS thông thường.

- + Hỗ trợ route URL tùy chỉnh cho web socket
- + Tích hợp dễ dàng hơn với Express JS
- + Hỗ trợ clustering với Redis

## 2.5 Ứng dụng đầu tiên

Dưới đây mô tả cách cài đặt và chạy một project “HelloWorld”.

Để cài đặt Nodejs trên ubuntu rất đơn giản bạn chỉ cần mở terminal lên và chạy command:

```
sudo apt install nodejs
```

Tiếp theo để kiểm tra xem đã cài đặt thành công Nodejs hay chưa bạn chỉ cần chạy command

```
node -v hoặc node -version
```

```
h2s@h2s-VirtualBox:~$ node --version
v10.15.2
h2s@h2s-VirtualBox:~$ npm -v
5.8.0
h2s@h2s-VirtualBox:~$ █

C:\Users\Tarnots>node -v
v16.14.2
C:\Users\Tarnots>
```

Nếu nó hiện ra trên thì có nghĩa bạn đã cài đặt thành công

- Npm (Node package manager) là một công cụ tạo và quản lý các thư viện lập trình Javascript cho Node.js nó được tích hợp sẵn vào trong node.js. Nên khi các bạn cài đặt node.js thì đã có npm luôn rồi.

Bây giờ chúng ta sẽ tạo một project Nodejs với npm bạn chỉ cần chạy command:

```
npm init --yes hoặc npm init
```

Tiếp theo mình sẽ cd đến thư mục vừa cài project nodejs để cài thêm framework `express` của Nodejs, khi đã có npm thì việc cài đặt rất đơn giản vì `express` nó đã là một module trong npm bạn chỉ cần chạy command sau để cài:

```
npm install express --save
```

Thêm `--save` để có thể lưu phiên bản của module vào phần `dependencies` trong file `package.json`.

### ❖ *Code Demo*

Khi Nodejs chạy nó sẽ chạy vào file được chỉ định trong key `main` ở file `packager.json` và ở đây mình đang để mặc định là

file `index.js` vì thế mình sẽ tạo một file mới có tên là `index.js` để đảm bảo viết code vào đó thì chắc chắn code được chạy . Và đây là file `index.js` của mình

```
const express = require('express')
const router = express.Router()

router.get('/', (req, res, next) =>{
  res.send("helloworld")
})
```

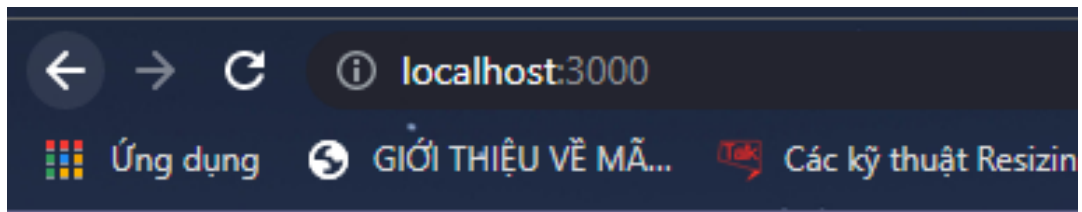
Sau đây mình sẽ giải thích qua đoạn code trên :

`const express = require('express');` Trong Nodejs bạn muốn sử dụng module nào thì bạn cần phải require module đó vào, ở đây mình muốn sử dụng `express` nên mình phải require nó vào thôi .

`const app = express();` Tạo một `app` sử dụng module `express` mình vừa require ở trên

```
app.get('/', function(req, res){
  res.send("Hello World");
})
```

Hàm `get()` sẽ có 2 tham số, tham số đầu tiên là địa chỉ mà server sẽ nhận request từ client để thực thi function là tham số thứ 2. Ở đây khi truy cập vào đường dẫn `http://localhost:3000/` thì bạn sẽ nhận được kết quả



```
app.listen(port, function(error){  
  
    if (error) {  
  
        console.log("Something went wrong");  
  
    }  
  
    console.log("server is running port: " + port);  
  
})
```

Hàm `listen()` sẽ khởi động server. Hàm này có 2 tham số, tham số đầu tiên là port mà ứng dụng NodeJS của bạn sẽ chạy, tham số thứ 2 là một callback function sẽ được gọi khi server khởi động. Function này lại chứa một tham số `error` để bắt lỗi khi mà server không thể khởi động vì một lý do nào đó. Ở đây mình `console.log()` để biết là server có khởi động thành công hay gặp lỗi, còn nếu các bạn không thích thì có thể viết lại như thế này.

```
app.listen(port)
```

## 2.6 Route trong NodeJS

Route là một thành phần cực kỳ quan trọng của một website, nó giúp website biết được người dùng truy cập đến nơi nào của trang web, từ đó

phản hồi lại một cách thích hợp. Trong ExpressJs, route được tích hợp sẵn và dễ dàng sử dụng.

### 2.6.1 Basic Route

Để sử dụng các phương thức GET, POST, PUT, DELETE trong route, bạn chỉ cần khai báo như sau:

```
// respond with "Hello World!" on the homepage

app.get('/', function (req, res) {

  res.send('Hello World!');

});

// accept POST request on the homepage

app.post('/', function (req, res) {

  res.send('Got a POST request');

});

// accept PUT request at /user

app.put('/user', function (req, res) {

  res.send('Got a PUT request at /user');

});

// accept DELETE request at /user

app.delete('/user', function (req, res) {

  res.send('Got a DELETE request at /user');

});
```



## 2.6.2 Route Parameters

Ví dụ ta có đoạn code bên dưới:

```
// Get user's info by id

app.get('/user/:id', function(req, res) {

    var id = req.params['id'];

    res.send('The id is ' + id);

});
```

Với khai báo bên trên khi bạn truy cập vào địa chỉ localhost:3000/user/6969 thì server sẽ trả lại bạn đoạn text: The id is 6969.

## 2.6.3 Tách route ra file riêng

Thông thường chúng ta thường tách các đoạn code route ra thành những file route riêng biệt để tiện quản lý, VD: userRouter.

- Tạo file userRouter.js với code như sau:

```
module.exports = function(app) {

    app.route('/user')

        .get(function (req, res) {

            res.send('Hello World!');

        })

        .post(function (req, res) {

            res.send('Got a POST request');
```

```

    })

    .put(function (req, res) {

        res.send('Got a PUT request at /user');

    })

    .delete(function (req, res) {

        res.send('Got a DELETE request at /user');

    });

}

```

- Chỉnh file index.js lại như sau:

```

var express = require('express');

var app = express();

var userRouter = require('userRouter');

userRouter(app);

var server = app.listen(3000, function () {

    var host = server.address().address

    var port = server.address().port

    console.log("Ung dung Node.js đang hoạt động tại địa chỉ:
http://%s:%s", host, port)

});

```

## 2.7 Kết nối cơ sở dữ liệu trong NodeJS

## 2.7.1 MySQL

Để kết nối node.js đến mysql chúng ta cần require module mysql mà chúng ta đã cài đặt (ở trên) theo cú pháp:

```
npm install mysql
```

Tiếp đó chúng ta cần phải sử dụng phương thức **createConnection** trong module mysql để cấu hình kết nối đến mysql theo cú pháp:

```
var conn = mysql.createConnection({
  host      : hostName,
  user      : userName,
  password  : password,
  database  : databaseName,
  charset   : charsetType
});
```

**Trong đó:**

- **hostName** - là host database
- **userName** - là user để đăng nhập vào mysql (nếu dùng xampp mặc định sẽ là admin).
- **password** - là mật khẩu của user để đăng nhập vào mysql (nếu dùng xampp mặc định sẽ bỏ trống).
- **database** - là database các bạn cần kết nối (nếu không muốn cấu hình có thể bỏ trống).
- **charsetType** - là kiểu charset. Mặc định thì **charset = utf8\_general\_ci**.

**Trong đó:** Các tham số truyền vào có giá trị tương ứng như ở phần cấu hình object. và các param truyền theo sẽ được cấu hình cách nhau bởi dấu & ([GET Method](#)).

Sau khi đã cấu hình xong thông tin config, bây giờ chúng ta sẽ sử dụng phương thức **connect** để tiến hành kết nối đến database theo cú pháp:

```
conn.connect();  
//hoặc  
conn.connect(function (err) {  
    //code  
});
```

Trong đó:

- `conn` - là object mysql đã chứa thông tin kết nối database.
- `err` - là biến chứa thông tin lỗi nếu có.

Để đóng kết nối Node.js với MySQL thì mọi người sử dụng phương thức **end** với cú pháp như sau:

```
conn.end();  
//hoặc  
conn.end(function (err) {  
    //  
});
```

Trong đó:

- `conn` - là object mysql đã chứa thông tin kết nối database.
- `err` - là biến chứa thông tin lỗi nếu có.

## 2.7.2 Mongodb

Để cài đặt mongoose bạn tiến hành mở terminal lên và gõ dòng lệnh:

```
npm i mongoose
```

**Mongoose** cung cấp một giải pháp dựa trên lược đồ đơn giản để mô hình hóa dữ liệu ứng dụng của bạn. Nó bao gồm các khuôn mẫu tích hợp, xác thực, xây dựng truy vấn, logic dữ liệu.

Để kết nối với MongoDB từ Node.js bằng module `mongoose`, hãy gọi phương thức `connect()`, trên biến tham chiếu đến `mongoose`, với URI cơ sở dữ liệu MongoDB được truyền làm đối số cho hàm.

```
var mongoose = require('mongoose');

mongoose.connect(uriConnect, [, options]);
```

Ở đây chúng ta sẽ có 2 tham số có thể truyền vào:

- `urlConnect`: url kết nối với MongoDB.
- `options`: một object chứa các tính chỉnh tùy chọn.

Ví dụ trong file `app.js` chúng ta kết nối với MongoDB ta thực hiện như sau:

```
var mongoose = require('mongoose')
mongoose.connect('mongodb://localhost:27017/nodejs',
{useNewUrlParser: true, useNewUrlParser: true, useUnifiedTopology:
true});
```

Trong một số trường hợp nào đó mà việc kết nối giữa server NodeJS và MongooseDB gặp vấn đề thì bạn có thể lỗi bằng cách bắt sự kiện `error` trong phương thức `mongoose.connection()`:

Khi kết nối được thực hiện mà không có bất kỳ lỗi nào và thành công, callback function được gọi trong `db.open ('open', [callback_function])` sẽ được thực thi.

Ở đây mình có file `app.js` có nội dung như sau:

```
var mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/freetuts',
{useNewUrlParser: true, useNewUrlParser: true, useUnifiedTopology:
true});
```

```

var db = mongoose.connection;

//Bắt sự kiện error

db.on('error', function(err) {

  if (err) console.log(err)

});

//Bắt sự kiện open

db.once('open', function() {

  console.log("Kết nối thành công !");

});

```

Đây là kết quả khi kết nối thành công.

```

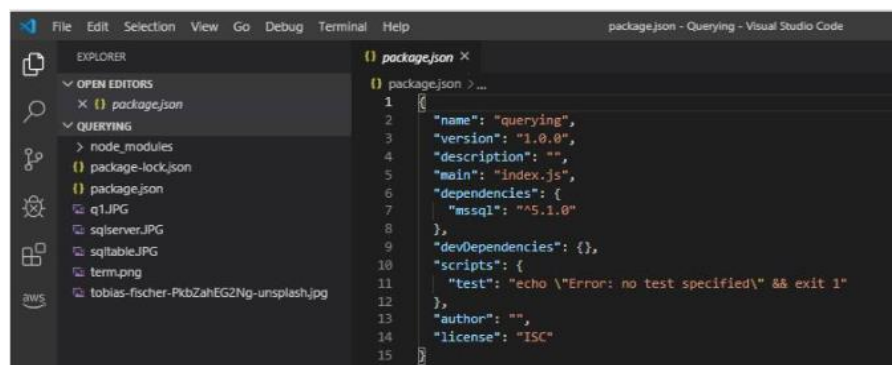
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
tri@tri:~/Desktop/freetuts.net/mysql$ node app
Kết nối thành công !

```

### 2.7.3 SQL Server

Để cài đặt sql server vào node.js ta cần câu lệnh: `npm install mssql`

Và khi cài xong chúng ta có thể kiểm tra nó ở file package.json



Và để có thể kết nối được sql server với node.js ta cần tạo một file config.js để thực hiện kết nối.

```

35 index.js > sql.connect() callback > sqlRequest.query() callback
1
2 // Access the drivers
3 var sql = require("mysql");
4
5 // config for your database
6 const config = {
7   user: 'nusoftware',
8   password: 'nusoftware',
9   server: 'LENOVO LAPTOP H10\\MSSQLSERVER01',
10  database: 'Company',
11  port: 1433 };
12
13 // connect to your database
14 sql.connect(config, function (err) {
15
16   if (err) console.log(err);
17
18   // create a new Request object
19   let sqlRequest = new sql.Request();
20
21   // query to the database and get the records/fields in the data Object
22   let sqlQuery='Select EmpId, FirstName, LastName, Title From Employees Where PerformanceRating=4';
23   sqlRequest.query(sqlQuery, function (err, data) {
24
25     if (err) console.log(err)
26
27     // Display the data in the console
28     console.table(data.recordset);
29
30     // Close the Connection
31     sql.close();
32
33   });
34 });
35

```

ở trên ta dùng câu lệnh config để kết nối với SQL Server .

đầu tiên ta cần truy cập trình điều khiển cho máy chủ SQL. Tiếp theo thì ta cần thiết lập cấu hình để cho code biết cách tìm phiên bản cơ sở dữ liệu, thông tin đăng nhập cần thiết để truy cập vào cơ sở dữ liệu mà chúng tôi muốn truy cập. kế tiếp ta thực hiện kết nối và kiểm tra nếu có lỗi thì thông báo và thực hiện câu lệnh truy vấn và dưới đây là kết quả hiện ở màn hình console.

35

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

C:\Program Files\nodejs\node.exe index.js

query run

(index)	EmpId	FirstName	LastName	Title
0	'000001'	'Fred'	'Jones'	'Lawyer'
1	'000002'	'Jane'	'Jones'	'Lawyer'
2	'000004'	'Frank'	'Brown'	'Clerk'
3	'000005'	'Jillian'	'Hill'	'Manager'

## 2.8 Query Builder

### 2.8.1 Join bảng

Ta thực hiện GET dữ liệu Post (là một bài đăng giống như cách mình đăng trạng thái trên Facebook)

Ta thực hiện join bảng bằng gọi tất cả các post trong bảng dữ liệu.

```
3   import express from 'express';
4   import mongoose from 'mongoose';
5
6   import PostMessage from '../models/postMessage.js';
7
8   const router = express.Router();
9
10  export const getPosts = async (req, res) => {
11    const { page } = req.query;
12
13    try {
14      const LIMIT = 8;
15      const startIndex = (Number(page) - 1) * LIMIT; // get
the starting index of every page
16
17      const total = await PostMessage.countDocuments({});
18      const posts = await PostMessage.find().sort({ _id: -1
}).limit(LIMIT).skip(startIndex);
19
20      res.json({ data: posts, currentPage: Number(page),
numberOfPages: Math.ceil(total / LIMIT)});
21    } catch (error) {
22      res.status(404).json({ message: error.message });
23    }
24  }
```

Join bảng bằng cách “*search theo tên*” hoặc “*search theo #tag*”

```
export const getPostsBySearch = async (req, res) => {
  const { searchQuery, tags } = req.query;

  try {
    const title = new RegExp(searchQuery, "i");

    const posts = await PostMessage.find({ $or: [ { title }, { tags:
{ $in: tags.split(',') } } ] });

    res.json({ data: posts });
  } catch (error) {
    res.status(404).json({ message: error.message });
  }
}
```

### 2.8.2 Insert dữ liệu

Ta insert dữ liệu :



```

export const createPost = async (req, res) => {
  const post = req.body;

  const newPostMessage = new PostMessage({ ...post, creator:
req.userId, createdAt: new Date().toISOString() })

  try {
    await newPostMessage.save();

    res.status(201).json(newPostMessage);
  } catch (error) {
    res.status(409).json({ message: error.message });
  }
}

```

### 2.8.3 Update dữ liệu

Ta Update dữ liệu:

```

export const updatePost = async (req, res) => {
  const { id } = req.params;
  const { title, message, creator, selectedFile, tags } = req.body;

  if (!mongoose.Types.ObjectId.isValid(id)) return
res.status(404).send(`No post with id: ${id}`);

  const updatedPost = { creator, title, message, tags, selectedFile,
_id: id };

  await PostMessage.findByIdAndUpdate(id, updatedPost, { new:
true });

  res.json(updatedPost);}

```

### 2.8.4 Delete dữ liệu

Ta Delete dữ liệu:

```

export const deletePost = async (req, res) => {
  const { id } = req.params;

  if (!mongoose.Types.ObjectId.isValid(id)) return
res.status(404).send(`No post with id: ${id}`);

  await PostMessage.findByIdAndRemove(id);

```

```
res.json({ message: "Post deleted successfully." });
}
```

## 2.9 Kiến trúc REST

REST là viết tắt của Representational State Transfer. REST là một chuẩn web dựa vào các kiến trúc cơ bản sử dụng giao thức HTTP. Nó xử lý tài nguyên, nơi mà mỗi thành phần là một tài nguyên và nguồn tài nguyên này có thể được truy cập qua các giao diện chung bởi sử dụng các phương thức HTTP chuẩn. REST lần đầu tiên được giới thiệu bởi Roy Fielding năm 2000.

Về cơ bản, một REST Server cung cấp các chế độ truy cập đến nguồn tài nguyên và REST Client truy cập và sửa đổi các nguồn tài nguyên này bởi sử dụng phương thức HTTP. Ở đây mỗi nguồn tài nguyên được xác định bởi một URI. REST sử dụng các cách biểu diễn khác nhau để biểu diễn các nguồn tài nguyên như text, JSON, XML nhưng phổ biến nhất vẫn là JSON.

### 2.9.1 Phương thức HTTP được sử dụng trong REST

Dưới đây là các phương thức HTTP được sử dụng rộng rãi trong kiến trúc REST:

- + **GET**: Được sử dụng chỉ để đọc các nguồn tài nguyên.

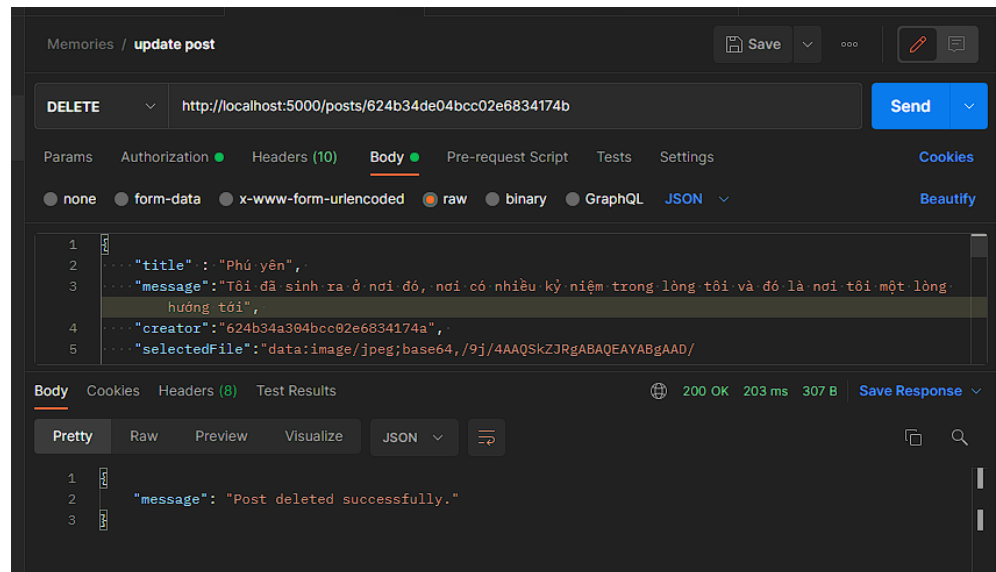
Ta thực hiện GET api trong Postman với id:

GET api tất cả post:



+ **DELETE**: Được sử dụng để xóa các nguồn tài nguyên.

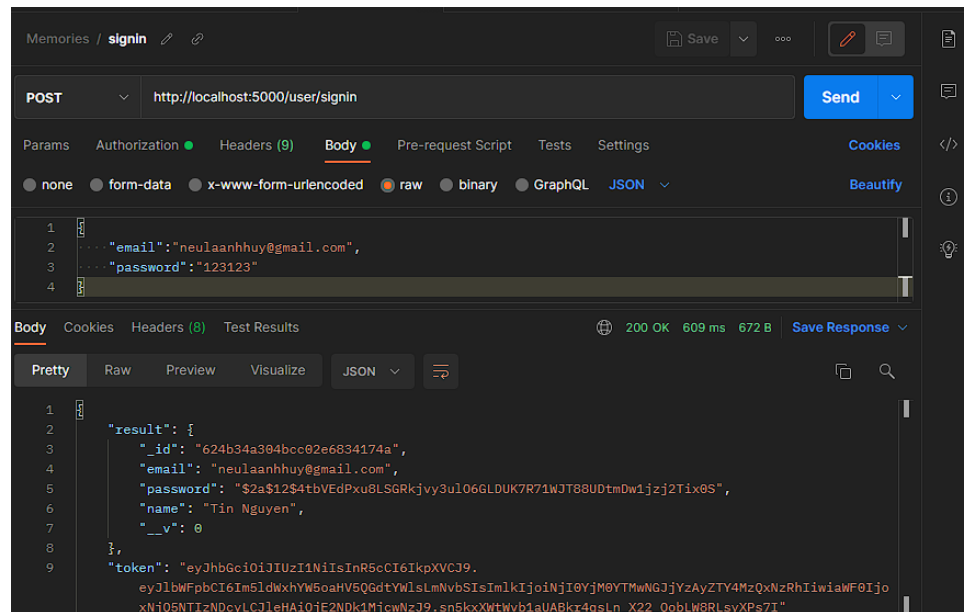
Khi muốn xóa một đối tượng ta lấy id của đối tượng đó và thực hiện lệnh DELETE để xóa đối tượng được chọn. Việc xóa này tùy thuộc vào dự án của bạn có cần xác thực quyền như trên PATCH hay không? Nếu có ta vẫn phải đăng nhập để lấy token để thực hiện được việc xóa Post.



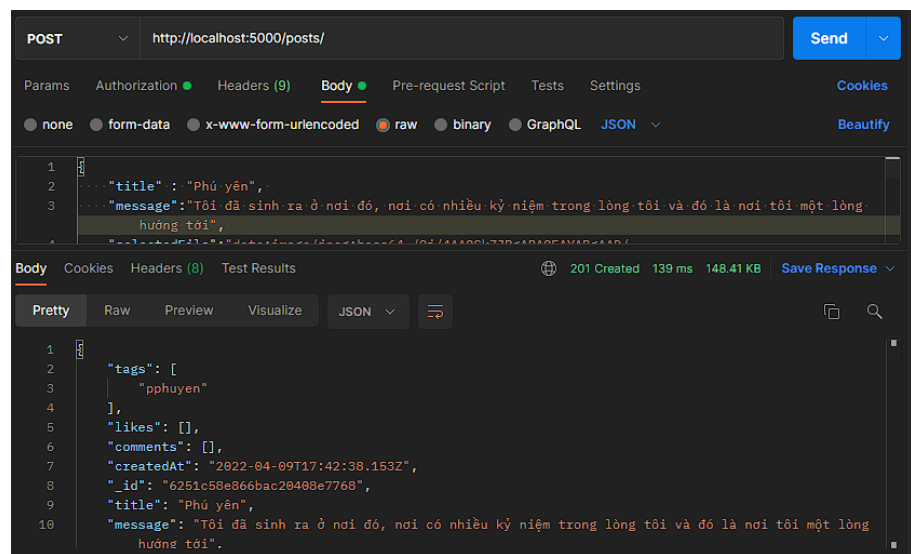
+ **POST**: Được sử dụng để cập nhật các bản ghi hiện tại và tạo mới nguồn tài nguyên.

Lệnh Post ta có thể dùng để đăng nhập hoặc tạo một đối tượng.

Và hình dưới cho ta thấy về việc đăng nhập một tài khoản và ta cũng thấy có token của tài khoản được đăng nhập và token dùng để thực hiện các lệnh Patch, Delete, Created thuộc quyền quản lý của tài khoản



Và phần này ta dùng lệnh Post để tạo ra một khoảng khắc với token của một tài khoản



### 2.9.2 RESTful Web Service

Một web service là một tập hợp các giao thức và chuẩn được sử dụng cho mục đích trao đổi giữa ứng dụng và hệ thống. Các ứng dụng phần mềm được viết bởi các ngôn ngữ khác nhau và chạy bởi các nền tảng khác nhau có thể sử dụng web service để trao đổi dữ liệu qua mạng máy tính như internet theo các cách tương tự như trao đổi trên một máy tính.

Web service dựa trên các kiến trúc REST được biết như RESTful webservice. Những webservice này sử dụng phương thức HTTP để triển khai các định nghĩa kiến trúc REST. Một RESTful web service thường được định nghĩa một URI (kiểu như đường dẫn), Uniform Resource Identifier như một service (dịch vụ).

### 2.9.3 Tạo RESTful cho một thư viện

Giả sử chúng ta có một cơ sở dữ liệu dựa trên JSON chứa thông tin về User, tên file là **users.json**:

```
1 {
2   "user1" : {
3     "name" : "phi",
4     "password" : "password1",
5     "profession" : "sinhvien",
6     "id": 1
7   },
8   "user2" : {
9     "name" : "manh",
10    "password" : "password2",
11    "profession" : "giangvien",
12    "id": 2
13  },
14  "user3" : {
15    "name" : "thang",
16    "password" : "password3",
17    "profession" : "laptrinhvien",
18    "id": 3
19  }
20 }
```

Dựa vào các thông tin cơ bản này, chúng ta sẽ cung cấp các RESTful API sau đây:

STT	URI	Phương thức HTTP	POST body	Kết quả
1	listUsers	GET	empty	Hiển thị danh sách user
2	addUser	POST	JSON String	Thêm một user mới
3	deleteUser	DELETE	JSON String	Xóa một user hiện tại.
4	:id	GET	empty	Hiển thị chi tiết một user

### 2.9.4 Liệt kê các User

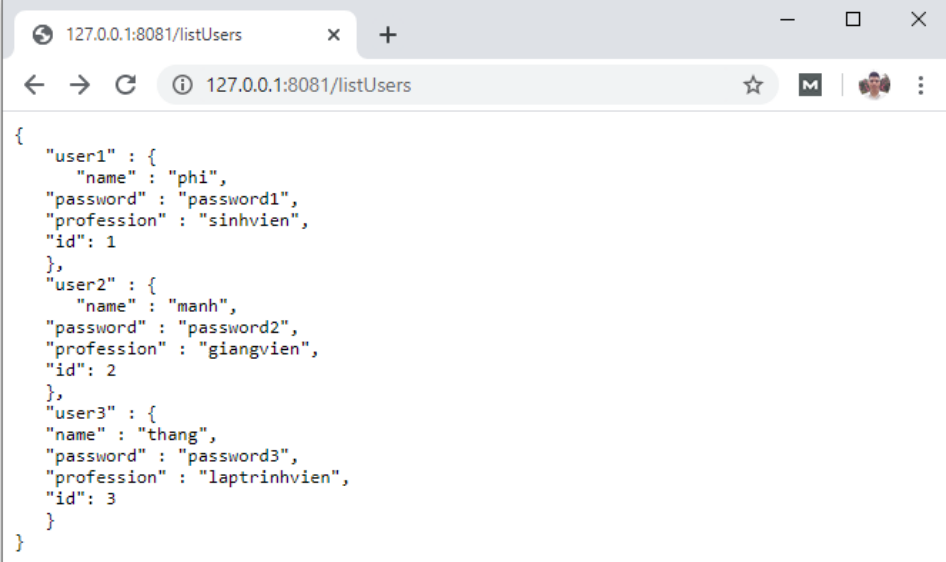
Chúng ta cùng triển khai RESTful API đầu tiên có tên listUsers bởi sử dụng đoạn code sau đây:

```

1  var express = require('express');
2  var app = express();
3  var fs = require("fs");
4
5  app.get('/listUsers', function (req, res) {
6    fs.readFile(__dirname + "/" + "users.json", 'utf8', function (err, data) {
7      console.log( data );
8      res.end( data );
9    });
10 })
11
12 var server = app.listen(8081, function () {
13   var host = server.address().address
14   var port = server.address().port
15   console.log("Ung dung dang lang nghe tai: http://%s:%s", host, port)
16 })

```

Bây giờ thử truy cập API đã được định nghĩa trên bởi sử dụng `http://127.0.0.1:8081/listUsers` trên máy tính local. Nó sẽ cho ra kết quả sau đây:



```

{
  "user1" : {
    "name" : "phi",
    "password" : "password1",
    "profession" : "sinhvien",
    "id": 1
  },
  "user2" : {
    "name" : "manh",
    "password" : "password2",
    "profession" : "giangvien",
    "id": 2
  },
  "user3" : {
    "name" : "thang",
    "password" : "password3",
    "profession" : "laptrinhvien",
    "id": 3
  }
}

```

## 2.9.5 Thêm User mới

API sau chỉ ra cách thêm một User mới vào danh sách. Dưới đây là thông tin của User mới:

```

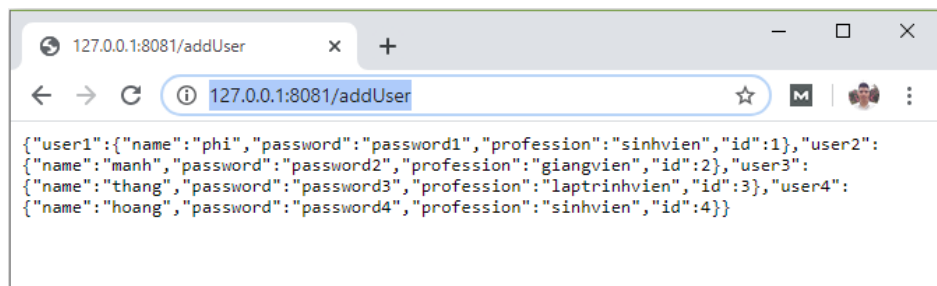
1  user = {
2    "user4" : {
3      "name" : "vinh",
4      "password" : "password4",
5      "profession" : "sinhvien",
6      "id": 4
7    }
8  }

```

Bạn có thể sử dụng Ajax để thực hiện việc này, nhưng để đơn giản chúng ta sẽ hard code ở đây. Dưới đây là phương thức addUser API để thêm một user mới trong cơ sở dữ liệu.

```
1  var express = require('express');
2  var app = express();
3  var fs = require("fs");
4
5  var user = {
6    "user4" : {
7      "name" : "hoang",
8      "password" : "password4",
9      "profession" : "sinhvien",
10     "id": 4
11   }
12 }
13
14 app.get('/addUser', function (req, res) {
15   // Đầu tiên, đọc tất cả các User đang tồn tại.
16   fs.readFile(__dirname + "/" + "users.json", 'utf8',
17     function (err, data) {
18       data = JSON.parse( data );
19       data["user4"] = user["user4"];
20       console.log( data );
21       res.end( JSON.stringify(data));
22     });
23 })
24
25 var server = app.listen(8081, function () {
26   var host = server.address().address
27   var port = server.address().port
28   console.log("Ứng dụng đang lắng nghe tại: http://%s:%s", host, port)
29 })
```

Bây giờ thử truy cập API trên bởi sử dụng `http://127.0.0.1:8081/addUsers` trên máy tính local. Kết quả sẽ được hiện ra như sau:



## 2.9.6 Hiện thị thông tin User

Bây giờ cùng triển khai một API mà gọi đến userID để hiện thị chi tiết thông tin User tương ứng.

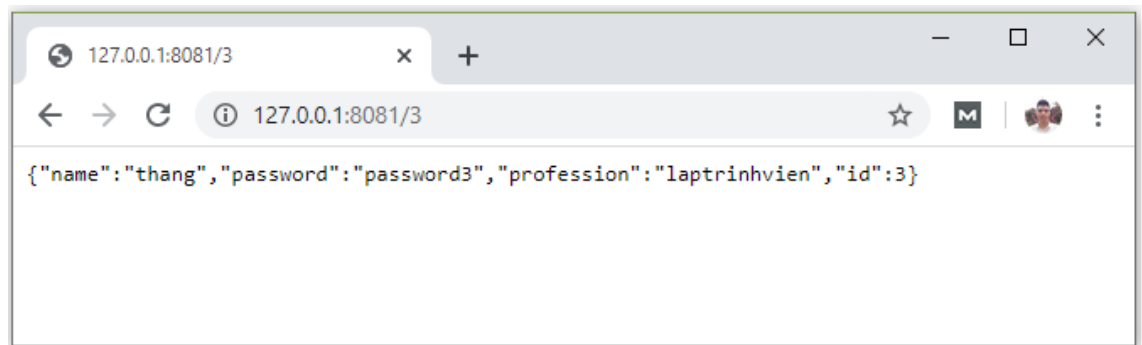


```

1  var express = require('express');
2  var app = express();
3  var fs = require("fs");
4
5  app.get('/:id', function (req, res) {
6    // Dau tien, doc tat ca cac User dang ton tai.
7    fs.readFile(__dirname + "/" + "users.json", 'utf8',
8      function (err, data) {
9        users = JSON.parse( data );
10       var user = users["user" + req.params.id]
11       console.log( user );
12       res.end( JSON.stringify(user));
13     });
14  })
15
16  var server = app.listen(8081, function () {
17    var host = server.address().address
18    var port = server.address().port
19    console.log("Ung dung dang lang nghe tai: http://%s:%s", host, port)
20  })

```

Tiếp đó, bạn gọi service trên bởi sử dụng địa chỉ  
<http://127.0.0.1:8081/3> trên máy tính local. Kết quả sẽ như sau:



### 2.9.7 Xóa User

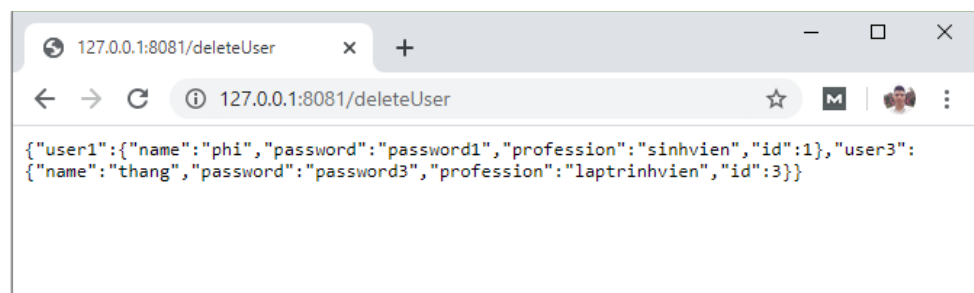
API này tương tự như addUser API, tại đây bạn có thể nhận một dữ liệu đầu vào thông qua req.body và sau đó dựa vào userID để xóa User đó khỏi Database. Để đơn giản, giả sử chúng ta xóa user có ID là 2.

```

1  var express = require('express');
2  var app = express();
3  var fs = require("fs");
4
5  var id = 2;
6  app.get('/deleteUser', function (req, res) {
7    // Dau tien, doc tat ca cac User dang ton tai.
8    fs.readFile( __dirname + "/" + "users.json", 'utf8',
9      function (err, data) {
10       data = JSON.parse( data );
11       delete data["user" + 2];
12
13       console.log( data );
14       res.end( JSON.stringify(data));
15     });
16  })
17
18  var server = app.listen(8081, function () {
19    var host = server.address().address
20    var port = server.address().port
21    console.log("Ung dung dang lang nghe tai: http://%s:%s", host, port)
22  })

```

Gọi service trên bởi sử dụng `http://127.0.0.1:8081/deleteUser` trên máy local. Nó sẽ cho ra kết quả sau đây:



## CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG VỚI NODEJS

### 3.1 Giới thiệu chung

Trong cuộc sống ta có những khoảng khắc muốn được nhiều người biết đến và muốn họ cùng tận hưởng với mình. Ta cũng có thể tương tác với họ qua bài viết mình đăng hoặc mình thấy trên bài đăng, vì thế ứng dụng memories-app được biết đến như là một cầu nối của ta với mọi người trên app. Web được thiết kế bởi React, Node.js và mongoDB một công nghệ không ít người biết đến và cấu trúc xây dựng lên trang web ta sẽ đi chi tiết

### 3.2 Cấu trúc

#### 3.1 Front-end

Web được thiết kế bởi framework React...và ta có những plugin hỗ trợ như material-ui, react-redux, react-router-dom....

```
client > package.json > {} dependencies > axios
dependencies: {
  "@material-ui/core": "^4.9.10",
  "@material-ui/icons": "^4.9.1",
  "@material-ui/lab": "^4.0.0-alpha.58",
  "@testing-library/jest-dom": "^4.2.4",
  "@testing-library/react": "^9.3.2",
  "@testing-library/user-event": "^7.1.2",
  "axios": "^0.19.2",
  "jwt-decode": "^3.1.2",
  "material-ui-chip-input": "^1.1.0",
  "moment": "^2.27.0",
  "react": "^16.12.0",
  "react-dom": "^16.12.0",
  "react-file-base64": "^1.0.3",
  "react-google-login": "^5.1.25",
  "react-redux": "^7.1.3",
  "react-router-dom": "^5.2.0",
  "react-scripts": "3.4.1",
  "redux": "^4.0.5",
  "redux-thunk": "^2.3.0"
}
```

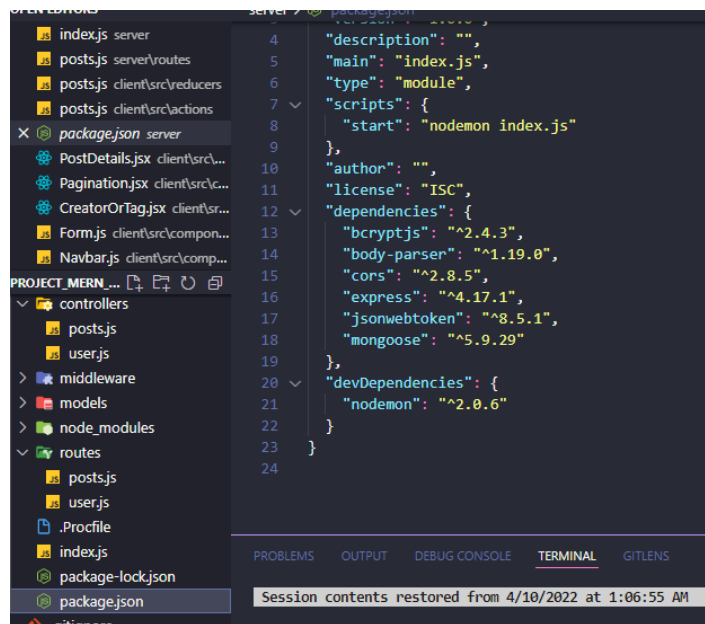
Ta dùng axios để tạo api gọi đường dẫn giao tiếp với server.

```
client > src > api > index.js > ...
1 import axios from 'axios';
2
3 const API = axios.create({ baseURL: 'http://localhost:5000' });
4
5 API.interceptors.request.use((req) => {
6   if (localStorage.getItem('profile')) {
7     req.headers.Authorization = `Bearer ${JSON.parse(localStorage.getItem('profile')).token}`;
8   }
9   return req;
10 });
11
12
13 export const fetchPost = (id) => API.get(`/posts/${id}`);
14 export const fetchPosts = (page) => API.get(`/posts?page=${page}`);
15 export const fetchPostsByCreator = (name) => API.get(`/posts/creator?name=${name}`);
16 export const fetchPostsBySearch = (searchQuery) => API.get(`/posts/search?searchQuery=${searchQuery}`);
17 export const createPost = (newPost) => API.post('/posts', newPost);
18 export const likePost = (id) => API.patch(`/posts/${id}/likePost`);
19 export const comment = (value, id) => API.post(`/posts/${id}/commentPost`, { value });
20 export const updatePost = (id, updatedPost) => API.patch(`/posts/${id}`, updatedPost);
21 export const deletePost = (id) => API.delete(`/posts/${id}`);
22
23 export const signIn = (formData) => API.post('/user/signin', formData);
24 export const signUp = (formData) => API.post('/user/signup', formData);
```

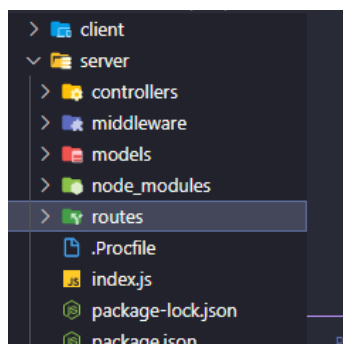
Và tạo những lệnh giao tiếp với server-client để thực hiện việc truy xuất dữ liệu và đưa dữ liệu lên giao diện.

### 3.2 Back-end

Phần Back-end ta dùng Node.js và MongoDB dùng để lưu trữ và truy xuất dữ liệu. Ngoài ra ta còn dùng framework là Express để dễ dàng cho việc tạo và sử dụng api hơn. Cũng không thể thiếu JsonWebToken (JWT) cho việc xác thực người dùng trong việc truy xuất dữ liệu.



Ở Server ta có các thư mục chính như: controller, models, middleware, routes.



Và phần quan trọng trong việc kết nối giữa database và server nằm ở file Index.js.

Ở file này ta dùng Connection\_url để giữ link liên kết với mongoose với port là 5000. Sau đó ta dùng lệnh mongoose.connect() để kết nối với mongoDB và thực hiện kiểm tra kết nối của nó.

```
import mongoose from 'mongoose';
import cors from 'cors';

import postRoutes from './routes/posts.js';
import userRouter from './routes/user.js';

const app = express();

app.use(express.json({ limit: '30mb', extended: true }));
app.use(express.urlencoded({ limit: '30mb', extended: true }));
app.use(cors());

app.use('/posts', postRoutes);
app.use("/user", userRouter);

const CONNECTION_URL = 'mongodb+srv://nin:123@cluster0.heggn.mongodb.net/memories-mern-app?retryWrites=true&w=majority';
const PORT = process.env.PORT || 5000;

mongoose.connect(CONNECTION_URL, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => app.listen(PORT, () => console.log(`Server Running on Port: http://localhost:${PORT}`)))
  .catch((error) => console.log(`${error} did not connect`));

mongoose.set('useFindAndModify', false);
```

Và ta cũng có app.use() để gọi tới routes nơi ta chứa các api của 2 đối tượng đó là user và post.

Đây là routes của user.

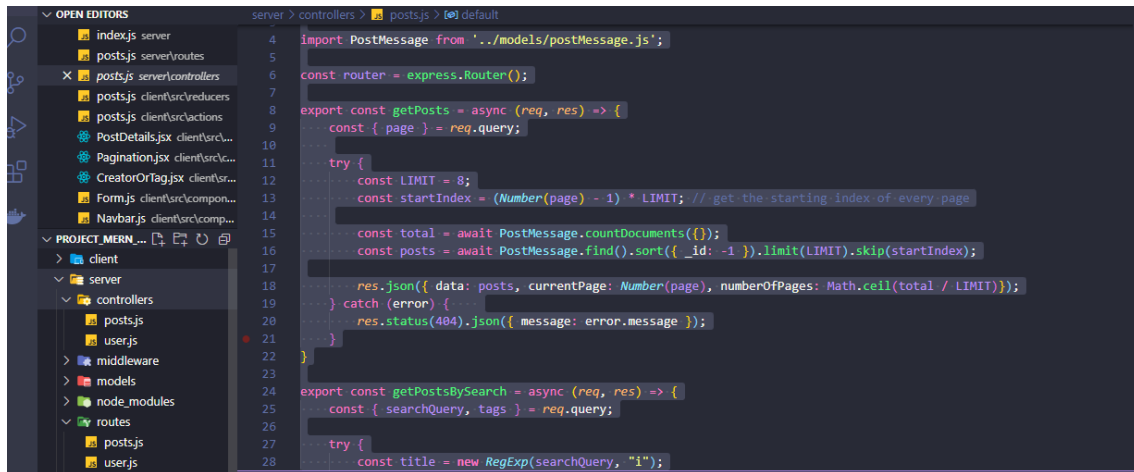
```
server > routes > user.js > ...
1 import express from "express";
2 const router = express.Router();
3
4 import { signin, signup } from "../controllers/user.js";
5
6 router.post("/signin", signin);
7 router.post("/signup", signup);
8
9 export default router;
```

Đây là routes của post

```
server > routes > posts.js > ...
1 import express from 'express';
2
3 import { getPosts, getPostsBySearch, getPostsByCreator, getPost, createPost, updatePost, likePost, commentPost, deletePost } from '../controllers/posts.js';
4
5 const router = express.Router();
6 import auth from "../middleware/auth.js";
7
8 router.get('/creator', getPostsByCreator);
9 router.get('/search', getPostsBySearch);
10 router.get('/', getPosts);
11 router.get('/:id', getPost);
12
13 router.post('/', auth, createPost);
14 router.patch('/:id', auth, updatePost);
15 router.delete('/:id', auth, deletePost);
16 router.patch('/:id/likePost', auth, likePost);
17 router.post('/:id/commentPost', commentPost);
18
19 export default router;
```

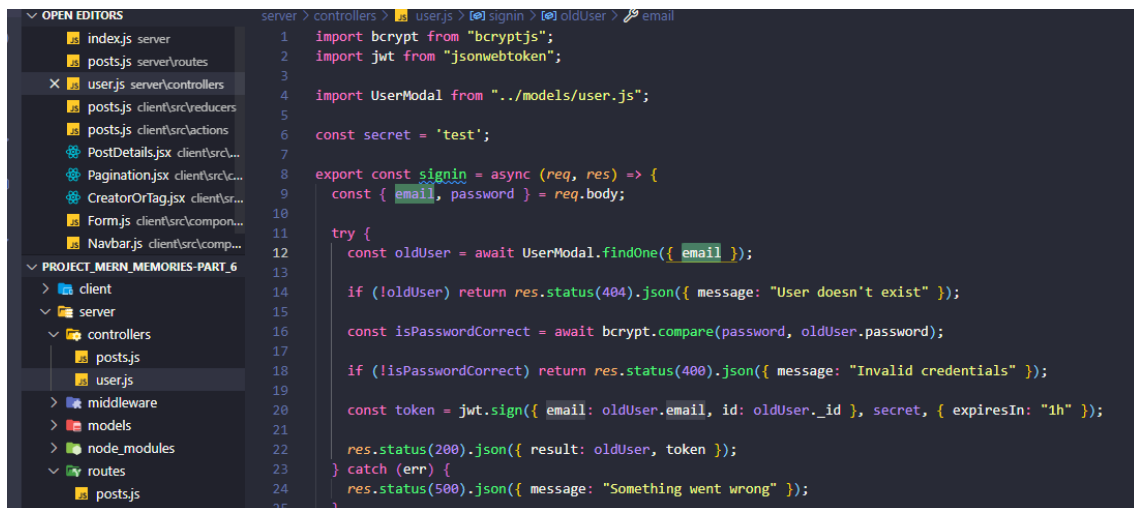
Khi sử dụng api ở routes ta có các chức năng tương ứng của 2 đối tượng được lưu ở controller. Bao gồm các chức năng như thêm, xóa, sửa, search của post và đăng nhập, đăng ký của user.

## Controller của post:



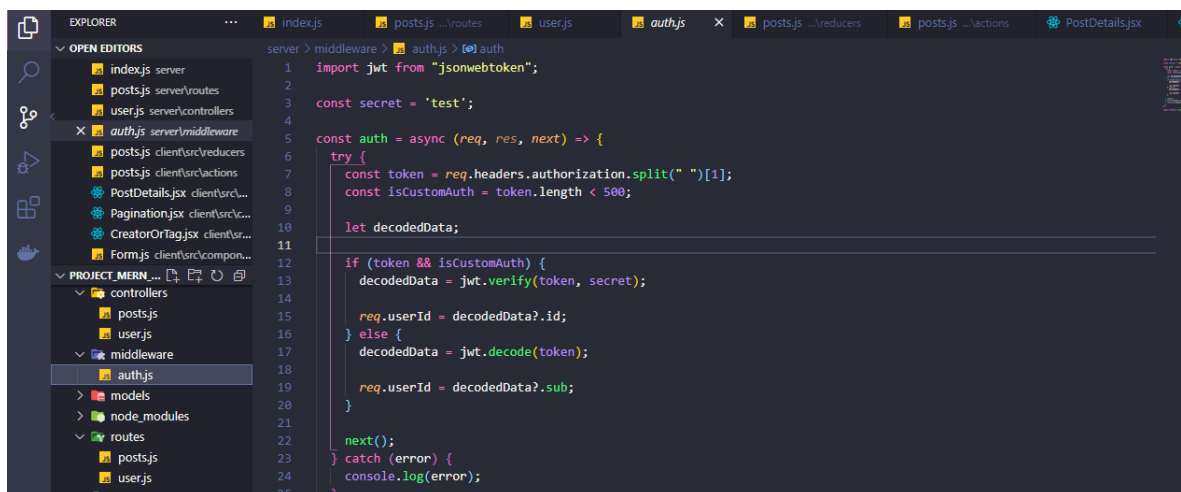
```
server > controllers > posts.js > default
4 import PostMessage from '../models/postMessage.js';
5
6 const router = express.Router();
7
8 export const getPosts = async (req, res) => {
9   const { page } = req.query;
10
11   try {
12     const LIMIT = 8;
13     const startIndex = (Number(page) - 1) * LIMIT; // get the starting index of every page
14
15     const total = await PostMessage.countDocuments();
16     const posts = await PostMessage.find().sort({ _id: -1 }).limit(LIMIT).skip(startIndex);
17
18     res.json({ data: posts, currentPage: Number(page), numberOfPages: Math.ceil(total / LIMIT) });
19   } catch (error) {
20     res.status(404).json({ message: error.message });
21   }
22 }
23
24 export const getPostsBySearch = async (req, res) => {
25   const { searchQuery, tags } = req.query;
26
27   try {
28     const title = new RegExp(searchQuery, "i");
```

## Và controller của user.



```
server > controllers > users.js > signin > email
1 import bcrypt from "bcryptjs";
2 import jwt from "jsonwebtoken";
3
4 import UserModel from '../models/user.js';
5
6 const secret = 'test';
7
8 export const signin = async (req, res) => {
9   const { email, password } = req.body;
10
11   try {
12     const oldUser = await UserModel.findOne({ email });
13
14     if (!oldUser) return res.status(404).json({ message: "User doesn't exist" });
15
16     const isPasswordCorrect = await bcrypt.compare(password, oldUser.password);
17
18     if (!isPasswordCorrect) return res.status(400).json({ message: "Invalid credentials" });
19
20     const token = jwt.sign({ email: oldUser.email, id: oldUser._id }, secret, { expiresIn: "1h" });
21
22     res.status(200).json({ result: oldUser, token });
23   } catch (err) {
24     res.status(500).json({ message: "Something went wrong" });
25   }
26 }
```

Thư mục middleware ta có file auth.js dùng để kiểm tra token nhằm xác định quyền của người dùng khi thực hiện việc truy xuất dữ liệu.



```
server > middleware > auth.js > auth
1 import jwt from "jsonwebtoken";
2
3 const secret = 'test';
4
5 const auth = async (req, res, next) => {
6   try {
7     const token = req.headers.authorization.split(" ")[1];
8     const isCustomAuth = token.length < 500;
9
10    let decodedData;
11
12    if (token && isCustomAuth) {
13      decodedData = jwt.verify(token, secret);
14      req.userId = decodedData?.id;
15    } else {
16      decodedData = jwt.decode(token);
17      req.userId = decodedData?.sub;
18    }
19
20    next();
21   } catch (error) {
22     console.log(error);
23   }
24 }
```

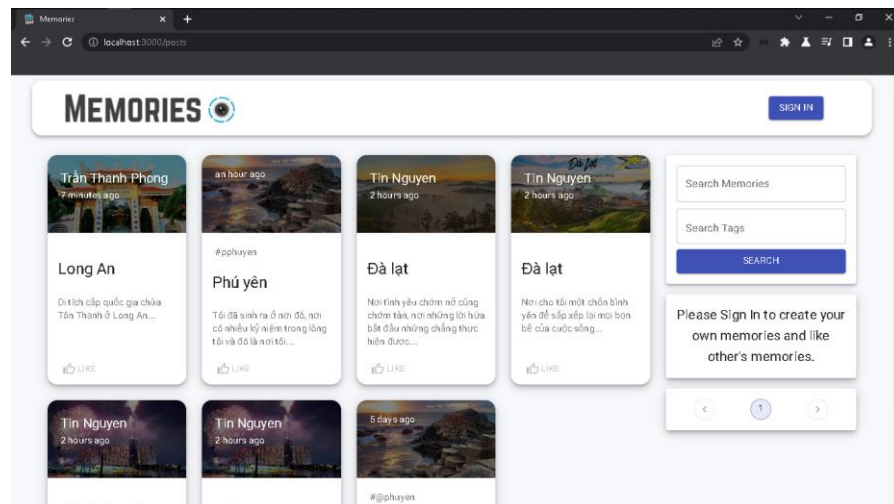
Việc thực hiện kiểm tra quyền xác thực được thực hiện ở các chức năng như thêm, xóa, sửa của bản thân và likepost của người khác.

```
12
13 router.post('/', auth, createPost);
14 router.patch('/:id', auth, updatePost);
15 router.delete('/:id', auth, deletePost);
16 router.patch('/:id/likePost', auth, likePost);
17 router.post('/:id/commentPost', commentPost);
18
19 export default router;
```

### 3.3 Giao diện website

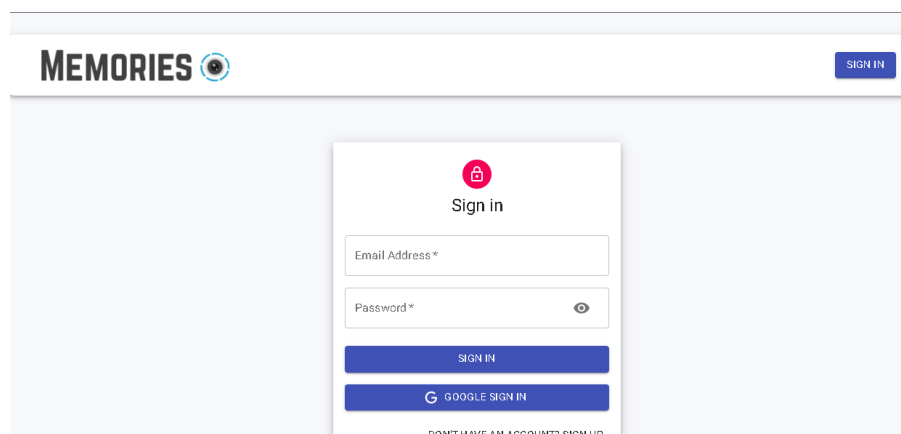
#### 3.1 Trang chính

Trang chính sẽ thấy những bài post mà người khác đã chia sẻ và mỗi bài đều được đính với tên người đã đăng bài đó.



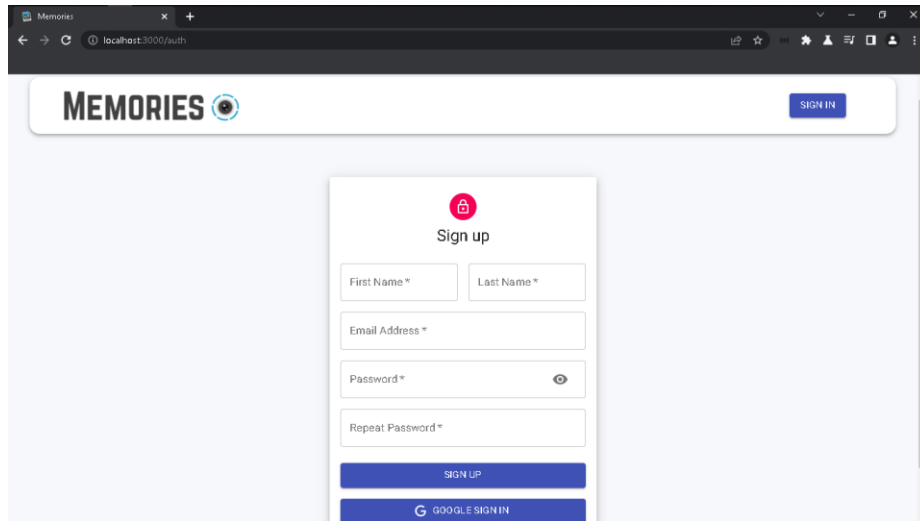
#### 3.2 Trang đăng nhập

Trang đăng nhập ngoài việc ta có thể đăng nhập bởi việc đăng ký tài khoản ta còn có thể đăng nhập bởi tài khoản google.



### 3.3 Trang đăng kí

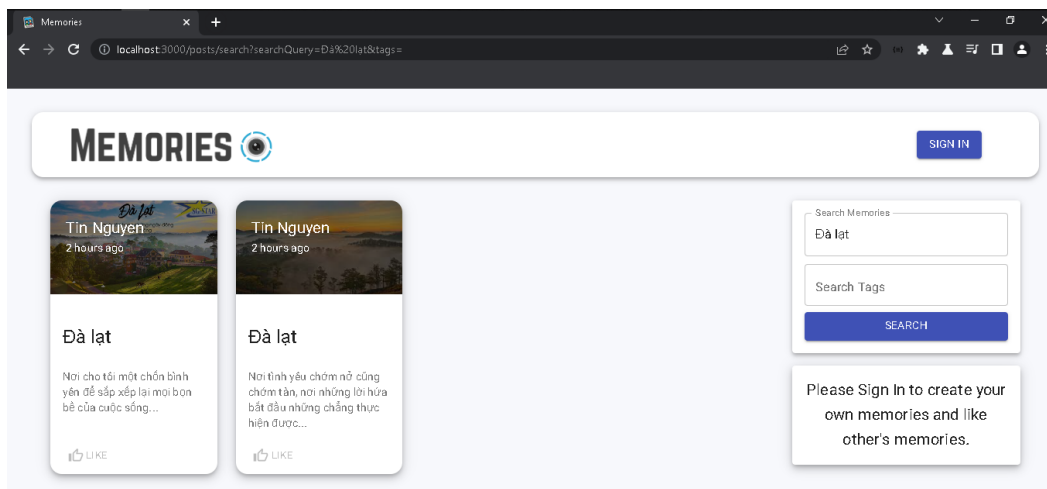
Hệ thống phải đăng ký bởi gmail của Google.



The screenshot shows a web browser window with the URL `localhost:3000/auth`. The page features the 'MEMORIES' logo and a 'SIGN IN' button in the top right. The main content is a 'Sign up' form with a red lock icon and the text 'Sign up'. The form includes input fields for 'First Name \*', 'Last Name \*', 'Email Address \*', 'Password \*' (with an eye icon for toggling visibility), and 'Repeat Password \*'. At the bottom of the form are two buttons: 'SIGN UP' and 'GOOGLE SIGN IN'.

### 3.4 Trang tìm kiếm

Chức năng tìm kiếm trong memories ta có thể tìm kiếm bởi title của bài post mà còn có thể tìm theo từ khóa được tag trong bài viết.

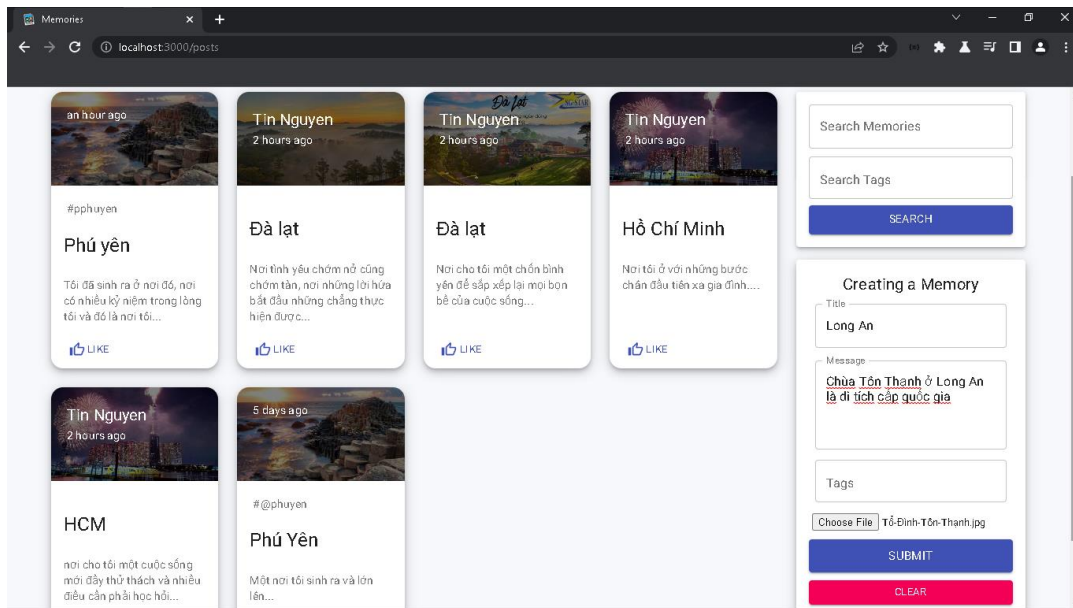


The screenshot shows a web browser window with the URL `localhost:3000/posts/search?searchQuery=Dà Lạt&tags=`. The page features the 'MEMORIES' logo and a 'SIGN IN' button in the top right. The main content area displays two search results for 'Dà Lạt' by 'Tin Nguyen', each with a '2 hours ago' timestamp and a 'LIKE' button. To the right, there is a search sidebar with 'Search Memories' and 'Search Tags' input fields, a 'SEARCH' button, and a message: 'Please Sign In to create your own memories and like other's memories.'

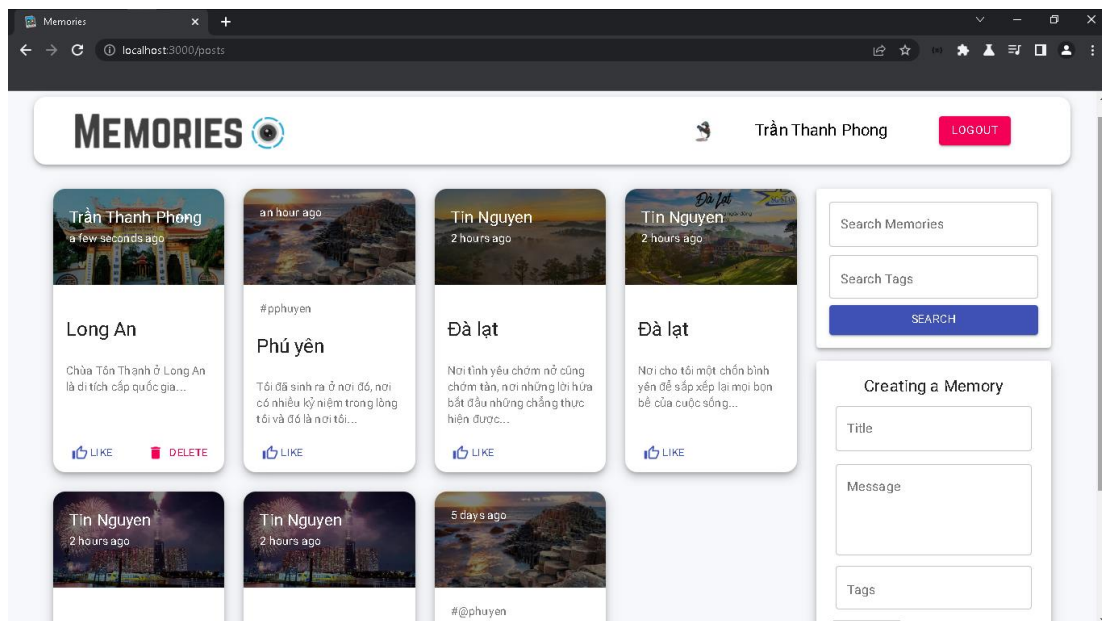
### 3.5 Trang thêm bài viết

Ở nơi thêm bài viết ta có các trường như: title, message, tag, tải hình ảnh...Sau khi ta nhập đủ các thông tin mình muốn thì nhấn button submit để hệ thống lưu bài viết về database và đưa lên danh sách Posts.



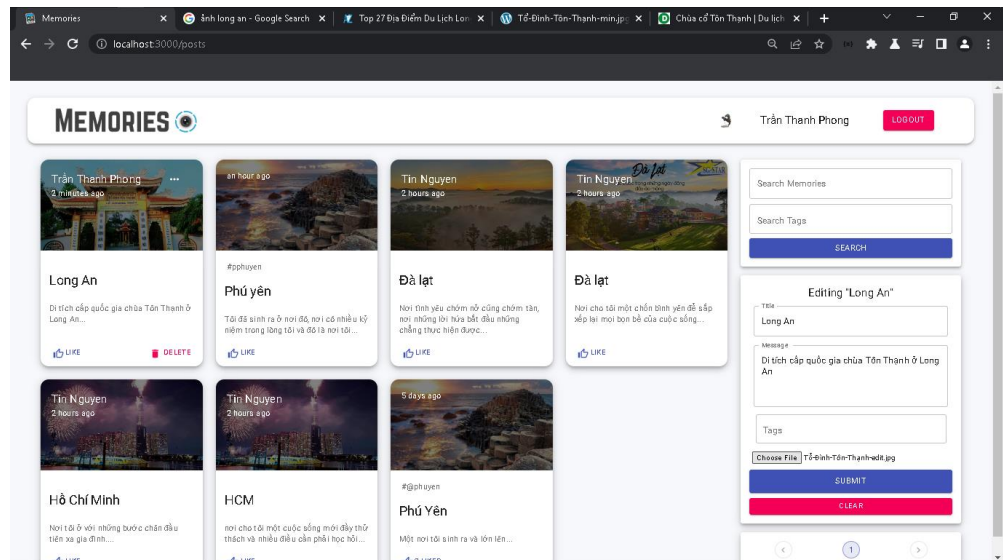


Và đây là kết quả sau khi ta nhập. Mỗi bài đều được đính với tên người đã đăng lên trên danh sách.



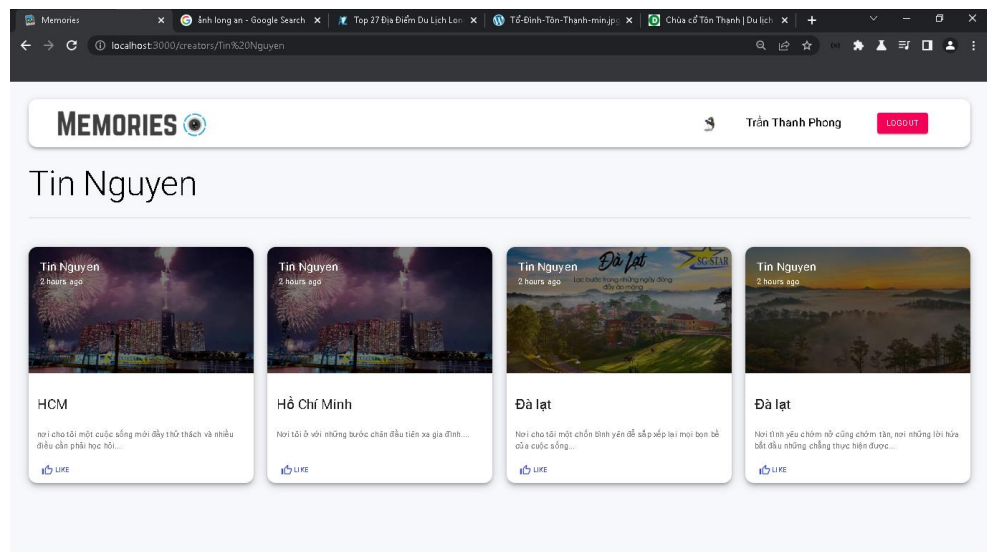
### 3.6 Trang chỉnh sửa

Ở những bài viết của chủ sở hữu muốn chỉnh sửa ta có dấu (“...”) ở phía trên góc phải khi bấm vào thì dữ liệu của bài viết được đẩy vào table “Edit” và việc chỉnh sửa được làm ở trong bản đó.



### 3.7 Xem những bài đăng của mình hoặc người khác

Người dùng có thể xem lại những bài đăng, những khoảnh khắc của mình hoặc người khác



## CHƯƠNG 4: TỔNG KẾT

### 4.1 Đánh giá

#### 4.1.1 Ưu điểm

+ *Tốc độ thực thi và khả năng mở rộng:*

Node.js có tốc độ rất nhanh. Đó là một yêu cầu khá quan trọng khi bạn là một startup đang cố gắng tạo ra một sản phẩm lớn và muốn đảm bảo có thể mở rộng nhanh chóng, đáp ứng được một lượng lớn người dùng khi trang web của bạn phát triển lên.

Node.js có thể xử lý hàng ngàn kết nối đồng thời trong khi PHP sẽ chỉ có nước súp đổ.

+ *JavaScript*

Do được viết bằng Javascript Node.js đang ngày càng trở nên lớn mạnh hơn vậy nên đây sẽ lợi thế cho việc tiếp cận dễ dàng hơn. Bất cứ khi nào bạn cần sự hỗ trợ về Node.js là gì hoặc những điều liên quan khác, sẽ có người hỗ trợ bạn vô cùng nhanh chóng.

+ *Dễ dàng cài đặt trên máy tính*

Chúng ta rất dễ cài đặt Node.js chạy cục bộ trên máy tính của bạn sử dụng các hệ điều hành như Windows, Mac hoặc Linux và bắt đầu phát triển ứng dụng ngay lập tức - chỉ việc tải phiên bản Node.js tương ứng tại đây.

#### 4.1.2 Nhược điểm

+ *Việc triển khai Node.js trên host không phải là điều dễ dàng*

Nếu bạn có một web hosting xài chung, bạn không thể đơn giản tải lên một ứng dụng Node.js và mong chờ nó hoạt động tốt.

VPS và dedicated server là một sự lựa chọn tốt hơn - bạn có thể cài đặt Node.js trên chúng. Thậm chí dễ hơn là sử dụng một dịch vụ có khả năng mở rộng như là Heroku, và bạn có thể hoàn toàn an

tâm để phát triển trang web của mình trên đó - bạn chỉ cần trả tiền khi cần thêm nhiều tài nguyên hơn.

+ *Còn đang phát triển*

Một nhược điểm lớn khác của Node.js đó là nó vẫn đang trong giai đoạn phát triển ban đầu, điều này có nghĩa là một số đặc trưng sẽ thay đổi trong quá trình phát triển tiếp theo.

Trong thực tế, nếu bạn đọc các tài liệu đi kèm, thì nó bao gồm một chỉ số ổn định (stability index), chỉ số này cho thấy mức độ rủi ro khi bạn sử dụng các đặc trưng hiện có.

## **4.2 Kết luận**

Node.js đòi hỏi phải làm thêm nhiều việc, nhưng phần thưởng của một ứng dụng nhanh chóng và mạnh mẽ là giá trị của nó. Node.js là một công nghệ đầy hứa hẹn và là sự lựa chọn tuyệt vời cho một ứng dụng cần hiệu năng cao, nhiều kết nối. Nó đã được chứng minh bởi các công ty như Microsoft, eBay...

## TÀI LIỆU THAM KHẢO

<https://viblo.asia/p/nodejs-tu-con-so-0-module-http-module-file-system-module-url-module-phan-1-gDVK2Mmv5Lj> (07/04/2022)

<https://nodejs.dev/learn/the-nodejs-http-module> (07/04/2022)

<https://niithanoi.edu.vn/nodejs-la-gi-tong-hop-day-du-ve-nodejs-ban-can-biet.html#framework-nodejs-pho-bien> (07/04/2022)

<https://viblo.asia/p/tong-quan-ve-node-js-AeJ1vOdQRkby> (07/04/2022)

<https://nguyenphudung.com/nodejs-la-gi-tim-hieu-tong-quan-ve-nodejs#toc--u-v-nh-c-i-m-c-a-nodejs> (07/04/2022)