**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN ĐIỆN TỬ**

---------------o0o---------------

****

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**TIÊU ĐỀ LUẬN VĂN**

**GVHD:**

**SVTH:**

**MSSV:**

**TP. HỒ CHÍ MINH, THÁNG NĂM 20**

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

TRƯỜNG ĐẠI HỌC BÁCH KHOA Độc lập – Tự do – Hạnh phúc.

-----✩----- -----✩-----

Số: \_\_\_\_\_\_ /BKĐT

Khoa: **Điện – Điện tử**

Bộ Môn: **Điện Tử**

N**HIỆM VỤ LUẬN VĂN TỐT NGHIỆP**

1. HỌ VÀ TÊN : MSSV:

1. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG** LỚP :
2. Đề tài:
3. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

1. Ngày giao nhiệm vụ luận văn: ...............................
2. Ngày hoàn thành nhiệm vụ: ...................................
3. Họ và tên người hướng dẫn: Phần hướng dẫn

................................................................. .....................................

................................................................. .....................................

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

*Tp.HCM, ngày…... tháng….. năm 20*

**CHỦ NHIỆM BỘ MÔN NGƯỜI HƯỚNG DẪN CHÍNH**

**PHẦN DÀNH CHO KHOA, BỘ MÔN:**

Người duyệt (chấm sơ bộ):.......................

Đơn vị:......................................................

Ngày bảo vệ : ...........................................

Điểm tổng kết: .........................................

Nơi lưu trữ luận văn: ...............................

***LỜI CẢM ƠN***

*Tp. Hồ Chí Minh, ngày tháng năm .*

**Sinh viên**

**TÓM TẮT LUẬN VĂN**

Luận văn này trình bày về …

**MỤC LỤC**

[1. GIỚI THIỆU 1](#_Toc310380267)

[1.1 Tổng quan 1](#_Toc310380268)

[1.2 Tình hình nghiên cứu trong và ngoài nước 1](#_Toc310380269)

[1.3 Nhiệm vụ luận văn 1](#_Toc310380270)

[2. LÝ THUYẾT 1](#_Toc310380271)

[3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG 1](#_Toc310380272)

[4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM (NẾU CÓ) 2](#_Toc310380273)

[5. KẾT QUẢ THỰC HIỆN 2](#_Toc310380274)

[6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 4](#_Toc310380275)

[6.1 Kết luận 4](#_Toc310380276)

[6.2 Hướng phát triển 4](#_Toc310380277)

[7. TÀI LIỆU THAM KHẢO 4](#_Toc310380278)

[8. PHỤ LỤC 4](#_Toc310380279)

DANH SÁCH HÌNH MINH HỌA

[Hình 5‑1 Kết quả thi công 3](#_Toc310380287)

[Hình 5‑2 Kết quả mô phỏng 3](#_Toc310380288)

**DANH SÁCH BẢNG SỐ LIỆU**

[Bảng 1 Thông số hệ thống 3](#_Toc310380293)

# GIỚI THIỆU

## Tổng quan

Mô tả tổng quan về lĩnh vực liên quan đến đề tài và những mục tiêu cần nghiên cứu. Từ đó giới thiệu nhiệm vụ cần đặt ra cho đề tài luận văn.

## Tình hình nghiên cứu trong và ngoài nước

Xem xét nghiên cứu nhiều nguồn tài liệu

Nhận xét cách làm và kết quả của các nghiên cứu đã có, những tài liệu luận văn mà sinh viên đã tham khảo.

## Nhiệm vụ luận văn

Mô tả các nhiệm vụ của đề tài bao gồm yêu cầu, kết quả cần đạt và giới hạn đề tài. Trong từng nội dung sinh viên cũng cần trình bày thêm cách tiếp cận cũng như ý tưởng thực hiện.

(Ví dụ)

Nội dung 1: Tìm hiểu lý thuyết về …

Nội dung 2: Tìm hiểu về Kit …

Nội dung 3: Thiết kế bộ điều khiển … sử dụng giải thuật …

# LÝ THUYẾT

(Tùy theo nội dung nghiên cứu mà sinh viên chọn tiêu đề cho phù hợp)

Đối với nội dung tìm hiểu lý thuyết, sinh viên cần trình bày:

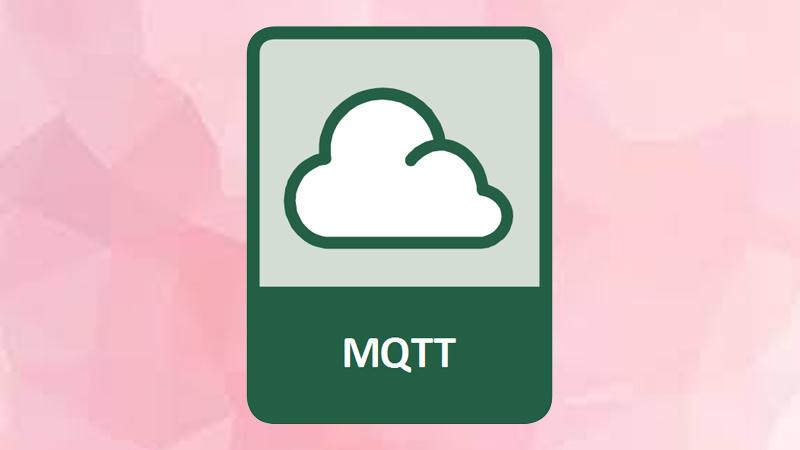
* Ngắn gọn và liên quan trực tiếp đến đề tài
* Mỗi chương liên quan đến một vấn đề
* Với phần lý thuyết không quan trọng, sinh viên có thể đưa vào mục tài liệu tham khảo.

**2.1. Giao thức kết nối.**

**Giao thức MQTT**

**Định nghĩa**

MQTT là từ viết tắt của cụm**Message Queuing Telemetry Transport**(tạm dịch:*giao thức truyền thông điệp*. Đây là một trong những giải pháp tiêu chuẩn của [IoT](https://tino.org/iot-platform-la-gi/) (**Internet of Things**) vì quá trình truyền tải của MQTT rất nhẹ, độ chính xác cao và khả năng kết nối băng thông hiệu quả. MQTT còn được hiểu là một giao thức nhắn tin thông minh và đơn giản, được tạo ra nhằm phục vụ cho các thiết bị hạn chế về băng thông.



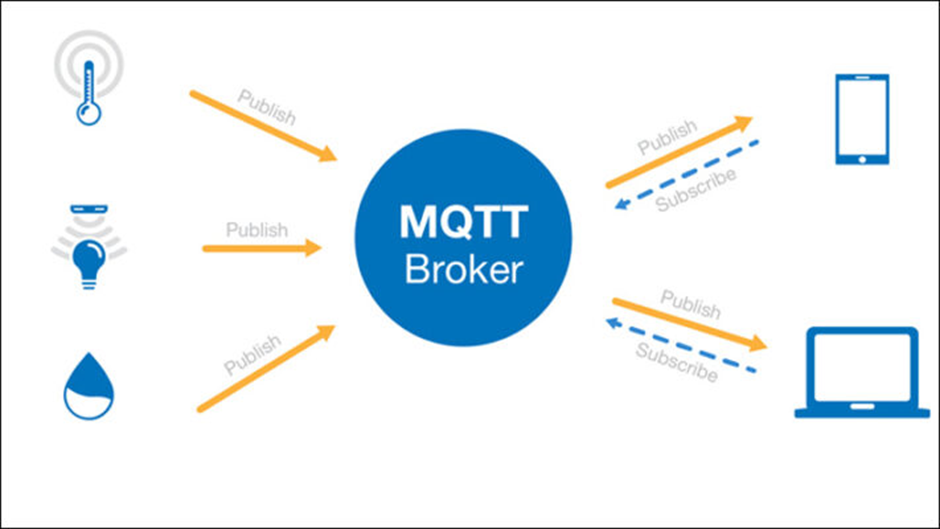
Nhiệm vụ của giao thức MQTT là hỗ trợ người dùng đọc, xuất bản dữ liệu. Bên cạnh đó, giao thức này còn giúp bạn gửi lệnh để điều khiển từ xa. Những hoạt động này đều thông qua nút cảm biến và một số tính năng khác. Nhiều người cho rằng MQTT là phương tiện để các thiết bị giao tiếp với nhau dễ dàng hơn.

**Lịch sử hình thành**

MQTT chính thức ra đời vào cuối những năm 1990. Giao thức này được phát minh bởi hai kỹ sư tài năng – Andy Stanford-Clark và Arlen Nipper. Nhiệm vụ của MQTT là “phương tiện” giao tiếp giữa hai thiết bị. Điển hình là đường ống dầu khí và [SCADA](https://tino.org/scada-la-gi/) (hệ thống giám sát và thu thập dữ liệu).

Trước khi MQTT xuất hiện, các thiết bị này không thể giao tiếp với nhau. Bởi vì chúng chỉ sử dụng các giao thức độc quyền, riêng biệt. Nhờ MQTT, vấn đề này đã được khắc phục nhanh chóng. Ngoài ra, điểm mạnh của giao thức này là: nhẹ, băng thông tối thiểu, dễ triển khai, cung cấp dữ liệu chất lượng,… Do đó, MQTT đã được ứng dụng mạnh mẽ hơn.

**Thành phần và cách thức hoạt động của MQTT**

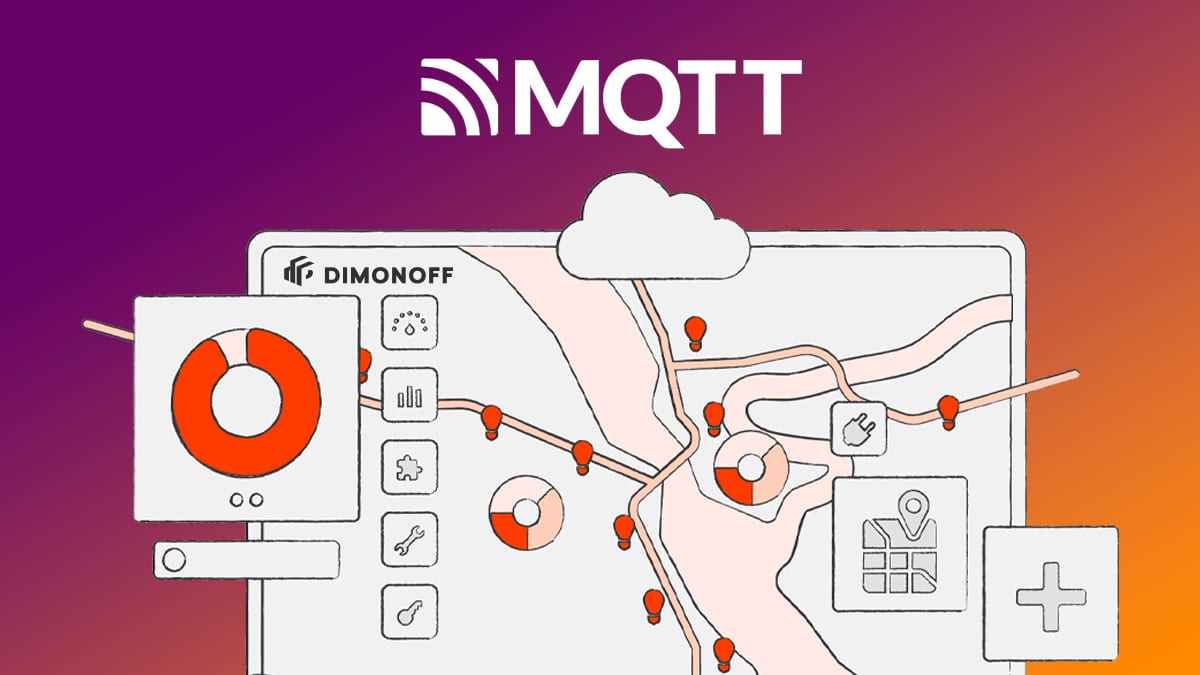
****

**Thành phần**

* + **MQTT broker:** Là thành phần được tạo ra dưới dạng mã nguồn mở. Một số MQTT broker có hỗ trợ dịch vụ điện toán đám mây. Chức năng của thành phần này là sàng lọc tin nhắn theo kênh. Kế đó, chúng đưa các tin nhắn này đến thiết bị hoặc ứng dụng đã đăng ký kênh ấy.
  + **MQTT client:** Thành phần này được nối kết với broker để truyền và nhận dữ liệu. Trong đó, client gửi dữ liệu được gọi là publisher, client đăng ký nhận dữ liệu là subscriber
  + **Topic:**Là chủ đề hay kênh được quản lý bởi broker và được trao đổi bởi các client với nhau

**Cách thức hoạt động**

* + **Broker:** Được xem là tâm điểm giữa của mọi kết nối đến tất cả client. Broker đảm nhiệm vai trò nhận tin nhắn, thông điệp được truyền tải từ publisher. Sau đó, thành phần này sẽ sắp xếp lại chúng và chuyển đến các subscriber nhất định.
  + **Client:** Công việc của client là xuất bản hoặc đăng ký nhiều kênh khác nhau
  + **Publisher:** Có chức năng truyền tải thông điệp, tin nhắn đến bất kỳ kênh nào
  + **Subscriber:** Có chức năng nhận thông điệp, tin nhắn từ những kênh đã đăng ký. Tuy nhiên, thành phần này chỉ có thể nhận dữ liệu từ publisher khi đã đăng ký chính xác kênh tương ứng.



**Một số khái niệm liên quan đến MQTT**

**MQTT – Publish/Subscribe (Xuất bản/Đăng ký)**

Khi giao thức MQTT hoạt động, những MQTT client (hay còn được gọi là node trạm) sẽ kết nối với một broker (**MQTT server**). Các node trạm sẽ đăng ký với một vài kênh (**Topic**). Hoạt động này được gọi là Subscribe (*Đăng ký*). Dữ liệu và kênh đã Subscribe sẽ được gửi đến mỗi client thông qua các note trạm. Quá trình client gửi dữ liệu đến một kênh gọi là Publish (*Xuất bản*).

**MQTT – Message (Tin nhắn)**

Tin nhắn trong giao thức MQTT được định dạng theo kiểu plain-text. Tuy nhiên, người dùng có thể tùy chỉnh tin nhắn thành các định dạng khác. Message là những thông tin trao đổi giữa các thiết bị của bạn. Chúng có thể là lệnh hoặc dữ liệu.

**MQTT – Topic (Kênh)**

Chủ đề là một trong những khái niệm quan trọng trong giao thức MQTT. Cũng được xem là “sợi dây nối kết” giữa Publish (*xuất bản*) và Subscribe (*đăng ký*). Nếu một tin nhắn được xuất bản trong một kênh, những người đăng ký kênh ấy sẽ nhận được tin nhắn này.

**MQTT – QoS**

QoS là từ viết của cụm **Qualities of Service** (tạm dịch: *chất lượng dịch vụ*). Người dùng có 3 lựa chọn khi Publish và Subscribe:

* + QoS0: Broker và client sẽ gửi dữ liệu duy nhất 1 lần dựa trên giao thức TCP/IP.
  + QoS1: Broker và client được phép gửi dữ liệu tối thiểu 1 lần và nhận xác nhận từ đầu kia
  + QoS2: Broker và client gửi dữ liệu và đầu bên kia chỉ nhận đúng 1 lần. Hoạt động này thông qua 4 bước bắt tay.

**MQTT – Retain**

Trong giao thức MQTT, retain đóng vai trò là flag (*lá cờ*) gắn cho một tin nhắn. Bên cạnh đó, retain chỉ nhận những giá trị là 0 hoặc 1 (tương tự như giá trị false hoặc true). Trong đó, nếu retain là 1, broker sẽ giữ lại tin nhắn cuối cùng của 1 kênh kèm theo QoS tương ứng. Client sẽ nhận được tin nhắn ấy khi đăng ký vào kênh chứa tin nhắn được lưu lại.

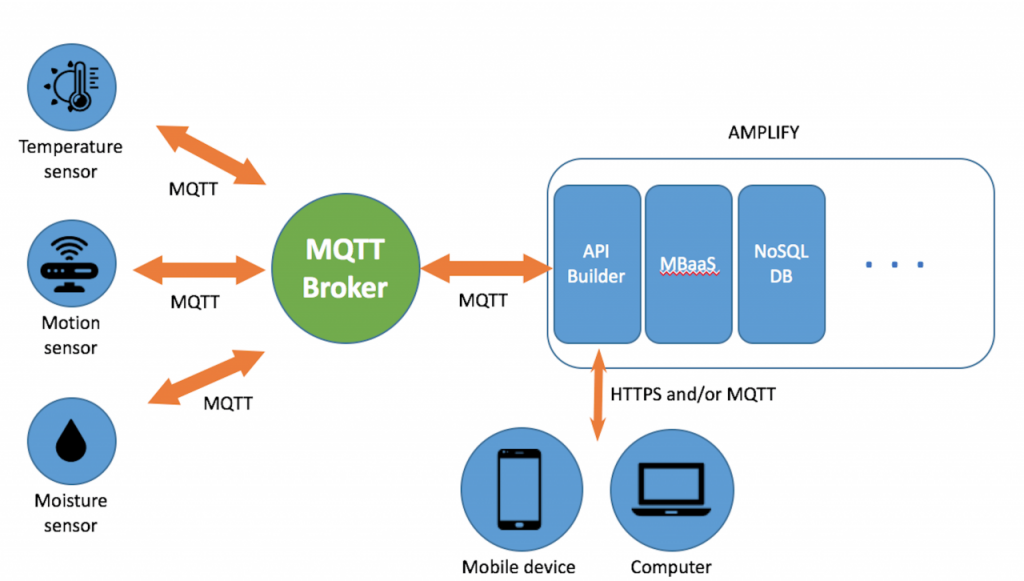


**MQTT – Bridge**

Đây là một tính năng quan trọng của MQTT broker. Chúng có vai trò giúp MQTT broker kết nối và trao đổi dữ liệu với nhau. Tính năng này chỉ được thực hiện khi có ít nhất 2 broker.

Trong đó, có một broker sẽ biến đổi thành bridge với những thông số sau:

* + **Address**: Tên địa chỉ broker cần kết nối
  + **Bridge\_protocol\_version**: Phiên bản mới của MQTT được sử dụng cho 2 broker
  + **Topic**: Bao gồm: tên topic được trao đổi giữa 2 broker, chiều trao đổi và topic mapping giữa 2 broker



**Giao thức MQTT phù hợp với môi trường:**

MQTT là giao thức thông minh, sở hữu nhiều tính năng thiết thực. Do đó đây chính là sự lựa chọn hoàn hảo cho:

- Quá trình giao tiếp giữa máy và máy (M2M).

- Những thiết bị nhúng bị giới hạn về bộ nhớ và tốc độ.

- Nơi có nhà mạng viễn thông đắt, thiếu uy tín và băng thông thấp.

**Ưu điểm của giao thức MQTT:**

- Có tính bảo mật, an toàn thông tin cao.

- Nhẹ và tối ưu.

- Tin nhắn được gửi đi đáng tin cậy.

- Có khả năng truyền thông theo hai hướng.

- Hỗ trợ đa dạng các thiết bị kết nối.

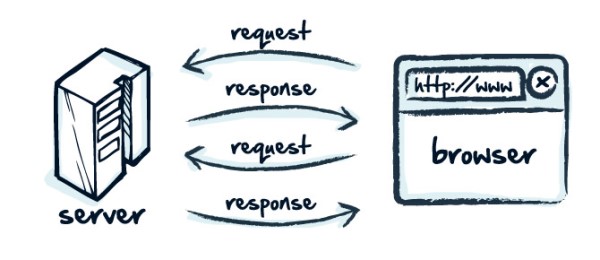
- Khắc phục hạn chế về mạng viễn thông kém chất lượng.

**Hạn chế của MQTT:**

- Giao thức MQTT chỉ sử dụng TLS/SSL để bảo mật, chúng không được mã hóa.

- Không khả thi cho việc mở rộng mạng MQTT trên toàn cầu.

**HTTP**



HTTP (Hypertext Transfer Protocol) là giao thức giao tiếp giữa web server và client. Giao thức này sử dụng mô hình client - server, trong đó client sẽ thực hiện request HTTP đến server web và server sẽ xử lý và trả về response với dữ liệu (image, css, js, html, xml, json ...) tương ứng với request của client.

Mỗi tương tác HTTP bao gồm một request và response. Theo bản chất của nó HTTP là Stateless.

Stateless có nghĩa là sau khi client gửi dữ liệu lên server, server thực thi xong, trả kết quả thì “quan hệ” giữa client và server bị “cắt đứt”. Tất cả các request đều tách biệt với nhau vì vậy mọi request phải tự chứa đủ thông tin để thực hiện request. Điều đó có nghĩa là mỗi transaction của mô hình dựa trên thông điệp của HTTP được xử lý riêng biệt với nhau.

**URL**

URL (Uniform Resource Locator) có lẽ là khái niệm được biết đến nhiều nhất của Web, nó cũng là một trong những khái niệm quan trọng và hữu ích nhất.

URL là một địa chỉ web được sử dụng để xác định tài nguyên trên Web.

Ý tưởng về web được cấu trúc xung quanh các tài nguyên, từ khi bắt đầu, Web là nền tảng để chia sẻ tệp văn bản, HTML, tài liệu, hình ảnh, v.v. và như vậy nó có thể được coi là một bộ sưu tập tài nguyên.

**Protocol**

Thông thường chúng là HTTP (hoặc HTTPS cho phiên bản HTTP an toàn).

Ngoài ra còn có các giao thức thường dùng khác:

* + File Transfer Protocol (FTP) - là giao thức chuẩn được sử dụng để truyền tệp giữa máy khách và máy chủ qua mạng.
  + Simple Mail Transfer Protocol (SMTP) là một tiêu chuẩn để truyền email...

**Domain**

Tên được sử dụng để xác định một hoặc nhiều địa chỉ IP nơi tài nguyên được đặt.

**Path**

Chỉ định vị trí tài nguyên trên máy chủ. Nó sử dụng logic giống như vị trí tài nguyên được sử dụng trên thiết bị bạn đang đọc bài viết này (VD: /search/cars/mazda.pdf hoặc /home/linhdn1198/my\_folder/my\_file).

**Parameters**

Dữ liệu bổ sung được sử dụng để xác định hoặc tìm kiểm tài nguyên trên máy chủ.

Khi tìm kiếm các bài viết và biết thêm thông tin về HTTP, bạn có thể gặp cụm từ URI (or uniform resource identifier). URI đôi khi được sử dụng thay vì URL nhưng chủ yếu là trong các thông số kỹ thuật chính thức.

**HTTP Requests**

Trong HTTP, mọi request phải có địa chỉ URL. Ngoài ra, request cần một phương thức. Bốn phương thức HTTP chính là: GET, PUT, POST, DELETE.

* + Các phương thức này tương ứng trực tiếp với các hành động: read, update, create, delete.
  + Tất cả các thông điệp HTTP có một hoặc nhiều header, theo sau là body, optional. Phần body chứa dữ liệu sẽ được gửi cùng với request hoặc dữ liệu nhận được cùng với phản hồi.
  + Phần đầu tiên của mỗi request HTTP chứa ba mục:

1. Phần đầu tiên là một phương thức cho biết phương thức HTTP nào được sử dụng. VD: GET, POST ...
2. Phần thứ hai là một URL được request.
3. Phần thứ ba là một phiên bản HTTP đang được sử dụng. Phiên bản 1.1. là phiên bản phổ biến nhất cho hầu hết các trình duyệt, ngoài ra còn có phiên bản 2.0.
   * Ngoài ra còn có một số điều khác trong request HTTP:

+ Referer header: Cho biết URL mà từ đó request bắt nguồn.

+ User-Agent header: Thông tin bổ sung về trình duyệt đang được sử dụng để tạo request.

+ Host header: xác định duy nhất một tên máy chủ, cần thiết khi nhiều trang web được lưu trữ trên cùng một máy chủ.

+ Cookie header: gửi thông số bổ sung cho khách hàng.

**HTTP Responses**

Giống như trong các request HTTP, response HTTP cũng bao gồm ba mục:

VD: HTTP/1.1 200 OK

1. Phần đầu tiên là phiên bản HTTP đang được sử dụng.
2. Phần thứ hai là mã số của kết quả cho request.
3. Phần thứ ba là một mô tả văn bản của phần thứ hai.

Có một số điều khác trong response HTTP:

- Server header: Thông tin mà phần mềm máy chủ web đang được sử dụng.

- Set-Cookie header: Thông tin mà phần mềm máy chủ web đang được sử dụng.

- Message body: Thông thường, một response HTTP có thể chứa nội dung thư.

- Content-Length header: cho biết kích thước của cơ thể thông điệp theo byte.

**HTTP Methods**

Các phương thức phổ biến nhất là GET và POST, tuy nhiên còn có một số phương thức khác.

* + GET: Được sử dụng để request dữ liệu từ một tài nguyên được chỉ định trong đó dữ liệu không được sửa đổi theo bất kỳ cách nào vì các request GET không thay đổi trạng thái của tài nguyên.
  + HEAD: Phương thức này có chức năng tương tự như phương thức GET nhưng với một điểm khác biệt là sự trở lại của phương thức HEAD không được chứa body trong response. Tuy nhiên, response trả về sẽ chứa các header giống như khi GET được sử dụng. Phương thức HEAD được sử dụng để kiểm tra xem tài nguyên có mặt trước khi thực hiện request GET.
  + POST: Được sử dụng để gửi dữ liệu đến một máy chủ để tạo tài nguyên.
  + PUT: Phương pháp để cập nhật tài nguyên hiện có trên máy chủ bằng cách sử dụng nội dung trong phần thân của request.
  + PATCH: Phương thức PATCH được sử dụng để áp dụng sửa đổi một phần cho tài nguyên.
  + DELETE: Phương thức DELETE xóa tài nguyên đã chỉ định.
  + TRACE: Phương pháp được thiết kế cho mục đích chẩn đoán. Response sẽ chứa trong cơ thể nội dung chính xác của thông báo request.
  + OPTIONS: Phương pháp này được sử dụng để mô tả các tùy chọn giao tiếp (phương thức HTTP) có sẵn cho tài nguyên đích.

**HTTP Headers**

Có ba thành phần chính tạo nên cấu trúc request/response bao gồm:

* + First line
  + Headers
  + Body/Content

Các HTTP Header được thêm vào sau dòng đầu tiên và được định nghĩa là các cặp tên: giá trị được phân tách bằng dấu hai chấm. HTTP header được sử dụng để gửi các tham số bổ sung cùng với request hoặc response.

Phần body của request bao gồm dữ liệu được gửi cùng với request hoặc dữ liệu nhận được cùng với response.

Có nhiều loại header khác nhau chia làm 4 loại chính:

* + General header: Header có thể được sử dụng trong cả yêu cầu và thông điệp response và độc lập với dữ liệu được trao đổi.
  + Request header: Các header này xác định tham số cho dữ liệu được yêu cầu hoặc tham số cung cấp thông tin quan trọng về máy khách thực hiện request.
  + Response header: Những header này chứa thông tin về response.
  + Entity header: Các entity header mô tả nội dung tạo nên phần thân của thông điệp.

**HTTP status codes**

Mỗi thông báo response HTTP phải chứa status code HTTP trong dòng đầu tiên, cho chúng ta biết kết quả của request.

Có 5 nhóm status code được nhóm theo chữ số đầu tiên:

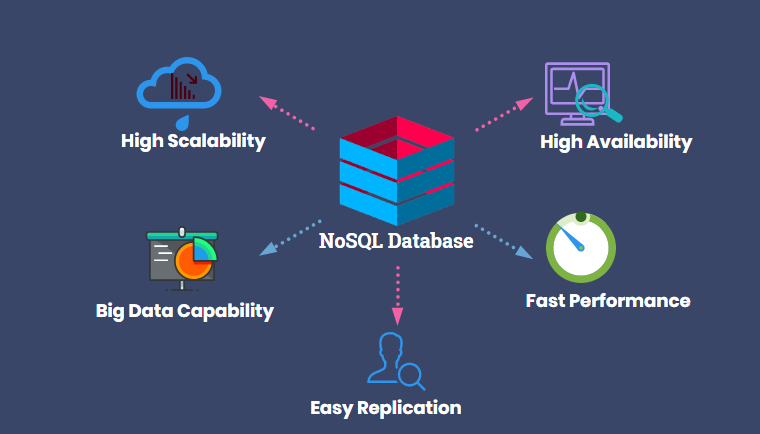
* + - 1xx: Thông tin response.
    - 2xx: Request đã thành công.
    - 3xx: Máy khách được chuyển hướng đến một tài nguyên khác.
    - 4xx: Yêu cầu có chứa một số lỗi.
    - 5xx: Máy chủ gặp lỗi khi thực hiện yêu cầu.
  1. **MERN stack & Socket.io**

Để xây dựng một website full-stack nhóm đã sử dụng Javascript bao gồm: MongoDB, ExpressJs, React, Node.js và Socket.io

**MongoDB**

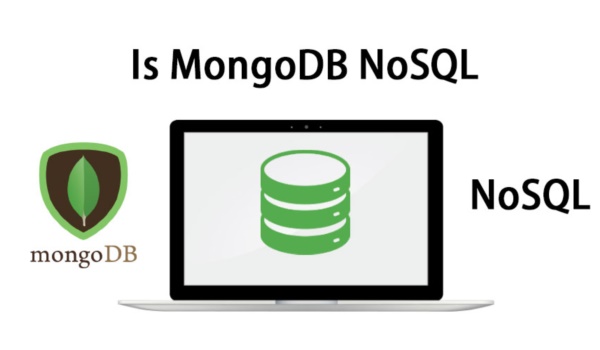
**Khái niệm về NoSQL**

Cơ sở dữ liệu NoSQL là viết tắt của Not Only SQL hoặc Not SQL. Trong đó thì cơ sở dữ liệu quan hệ RDBMS sẽ dùng cú pháp SQL khi cần lưu trữ, cũng như trích xuất dữ liệu. Còn đối với hệ thống cơ sở dữ liệu NoSQL sẽ bao gồm hàng loạt những công nghệ cơ sở dữ liệu với khả năng lưu trữ ở dạng cấu trúc, bán cấu trúc hoặc phi cấu trúc, hay dạng đa hình.  
Sử dụng cơ sở dữ liệu NoSQL là lựa chọn lý tưởng, hợp lý trong nhiều ứng dụng hiện đại hiện nay. Việc sử dụng NoSQL đem lại nhiều giá trị, nhiều lợi ích nên được tin tưởng ngày càng nhiều. Trong đó những lý do nên chọn NoSQL phải kể tới chính là:

****

* + - *Tính linh hoạt cao*: sơ đồ linh hoạt giúp quá trình phát triển nhanh chóng, cũng đảm bảo ở khả năng lặp lại cao hơn rất nhiều. Chính mô hình dữ liệu có khả năng linh hoạt cao sẽ biến cơ sở dữ liệu NoSQL trở thành sự lựa chọn hoàn hảo cho những dữ liệu ở dạng không được tổ chức thành cấu trúc, hay có cấu trúc chưa thực sự hoàn chỉnh.
    - *Có khả năng thay đổi ở quy mô*: đặc trưng của NoSQL chính là việc được thiết kế với mục đích giúp tăng quy mô bằng cách dùng những cụm phần cứng đã được phân phối thay vì tăng ở quy mô thông qua việc bổ sung máy chủ mạnh mẽ, tốn kém nhiều chi phí.
    - *Sở hữu hiệu năng cao*: việc được tối ưu hóa theo các mô hình dữ liệu cụ thể, cũng như các mẫu truy cập giúp NoSQL có thể tăng hiệu năng cao hơn so với việc chúng ta cố gắng đạt được mức độ chức năng tương tự, nhờ vào cơ sở dữ liệu quan hệ.
    - *Thực sự thiết thực*: với NoSQL cung cấp những API và các kiểu dữ liệu thực sự thiết thực, được tiến hành xây dựng riêng cho từng mô hình dữ liệu tương ứng. Nó đem lại tính ứng dụng cao hơn rất nhiều trong quá trình sử dụng.

**Khái niệm MongoDB**



MongoDB là một cơ sở dữ liệu NoSQL, một dạng database hướng tài liệu. Chúng thường được sử dụng để lưu trữ dữ liệu khối lượng lớn. MongoDB không sử dụng cấu trúc dạng bảng như relational database. Thay vào đó, MongoDB sẽ lưu trữ dữ liệu dưới dạng Document JSON. Vì vậy, mỗi một collection sẽ các các kích cỡ và các document khác nhau. Bên cạnh đó, việc các dữ liệu được lưu trữ trong document kiểu JSON dẫn đến chúng được truy vấn rất nhanh.

MongoDB được ra đời bởi MongoDB Inc. (một công ty phần mềm của Mỹ). Công ty được thành lập lần đầu tiên vào năm 2007 với tên gọi 10gen. Nó cũng là một PaaS (Platform as a Service) tương tự như Windows Azure và Google App Engine, sau đó đã được chuyển thành nguồn mở từ năm 2009.

MongoDB đã trở thành một trong những NoSQL database nổi trội nhất lúc bấy giờ, được dùng làm backend cho rất nhiều website như eBay, SourceForge và The New York Times.

**Các feature của MongoDB**

**Hướng tài liệu**

Vì MongoDB là cơ sở dữ liệu kiểu NoSQL. Nó lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu, thích ứng với tình hình và yêu cầu thực tế của doanh nghiệp.

**Các ad hoc query**

MongoDB hỗ trợ tìm kiếm bằng field, range query, regular expression và các phép tìm kiếm thông thường.

**Indexing**

Các index có thể được tạo để cải thiện hiệu suất của các tìm kiếm trong MongoDB. Bất kỳ field nào trong MongoDB document đều có thể được index.

**Replication**

“Nhân bản” là có một phiên bản giống hệt phiên bản đang tồn tại, đang sử dụng. Với cơ sở dữ liệu, nhu cầu lưu trữ lớn, đòi hỏi cơ sở dữ liệu toàn vẹn, không bị mất mát trước những sự cố ngoài ý muốn. Vì vậy, người ta nghĩ ra khái niệm “nhân bản”, nghĩa là tạo một phiên bản cơ sở dữ liệu giống hệt cơ sở dữ liệu đang tồn tại, và lưu trữ chúng ở một nơi khác, đề phòng có sự cố.

**Aggregation**

Các Aggregation operation xử lý các bản ghi dữ liệu và trả về kết quả đã được tính toán. Các phép toán tập hợp nhóm các giá trị từ nhiều Document lại với nhau, và có thể thực hiện nhiều phép toán đa dạng trên dữ liệu đã được nhóm đó để trả về một kết quả duy nhất.

**Lưu trữ file**

MongoDB được dùng như một hệ thống file tận dụng những function trên và hoạt động như một cách phân phối qua Sharding.

**Cân bằng tải**

Bằng cách sử dụng Sharding, MongoDB chia tỷ lệ theo chiều ngang để người dùng chọn một Shard key, xác định các dữ liệu được phân phối trong collection. Nhìn chung, dữ liệu được chia thành các phạm vi và được phân phối đồng đều dựa trên các Shard key. Nó chạy trên nhiều máy chủ, cân bằng tải hoặc sao chép dữ liệu để giữ hệ thống luôn hoạt động trong trường hợp có lỗi về phần cứng.

**3. Một số thuật ngữ phổ biến được sử dụng trong MongoDB**

**1. \_id**

Đây là trường bắt buộc trong mọi document của MongoDB. Trường \_id đại diện cho một giá trị duy nhất trong document MongoDB. Trường \_id giống như khóa chính của document. Nếu bạn tạo một document mới mà không có trường \_id, MongoDB sẽ tự động tạo trường \_id để đại diện cho document đó và nó là duy nhất trong cơ sở dữ liệu.

**2**. **Collection**

Đây là một nhóm gồm nhiều document trong MongoDB. Tập hợp tương đương với một bảng được tạo trong bất kỳ RDMS nào khác như Oracle hoặc MS SQL. Collection chứa trong một cơ sở dữ liệu duy nhất.

**3. Cursor**

Đây là một con trỏ đến tập kết quả của một truy vấn. Khách hàng có thể lặp lại qua một con trỏ để truy xuất kết quả.

**4. Database**

Đây là vùng chứa cho các collection như RDMS, trong đó nó là vùng chứa cho các bảng. Mỗi database có một bộ tệp riêng trên hệ thống tệp. Một máy chủ MongoDB có thể lưu trữ nhiều database.

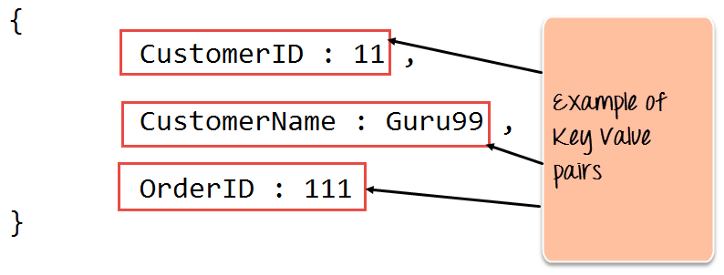
**5. Document**

Bản ghi trong bộ sưu tập MongoDB về cơ bản được gọi là document. Document sẽ bao gồm tên trường và các giá trị.

**6. Field**

Là một cặp tên-giá trị trong tài liệu. Một tài liệu có thể có không hoặc nhiều trường. Các trường tương tự như các cột trong cơ sở dữ liệu quan hệ.

Sơ đồ sau đây cho thấy một ví dụ về Field với các cặp giá trị Khóa. Ví dụ bên dưới, CustomerID và 11 là một trong những cặp giá trị khóa được xác định trong tài liệu.



Ví dụ về Trường với các cặp giá trị Khóa

**4. Ưu điểm của MongoDB**

- Đầu tiên có thể nhắc đến là tính linh hoạt lưu trữ dữ liệu theo các kích cỡ khác nhau. Do dữ liệu dưới dạng hướng tài liệu JSON nên bạn có thể chèn vào thoải mái bất cứ thông tin gì bạn muốn.

- Khác với RDBMS, dữ liệu ở đây không bị ràng buộc và không phải tuân theo khuôn khổ nhất định. Điều này giúp bạn tiết kiệm thời gian cho việc kiểm tra sự thỏa mãn về cấu trúc nếu muốn chèn, xóa, cập nhật hay thay đổi các dữ liệu trong bảng.

- MongoDB dễ dàng mở rộng hệ thống bằng cách thêm node vào cluster – cụm các node chứa dữ liệu giao tiếp với nhau.

- Tốc độ truy vấn nhanh hơn nhiều so với hệ quản trị cơ sở dữ liệu quan hệ RDBMS do dữ liệu truy vấn được cached lên bộ nhớ RAM để lượt truy vấn sau diễn ra nhanh hơn mà không cần đọc từ ổ cứng.

- Cuối cùng cũng là một ưu điểm về hiệu suất truy vấn của MongoDB. Trường dữ liệu “\_id” luôn được tự động đánh chỉ mục để đạt hiệu suất cao nhất.

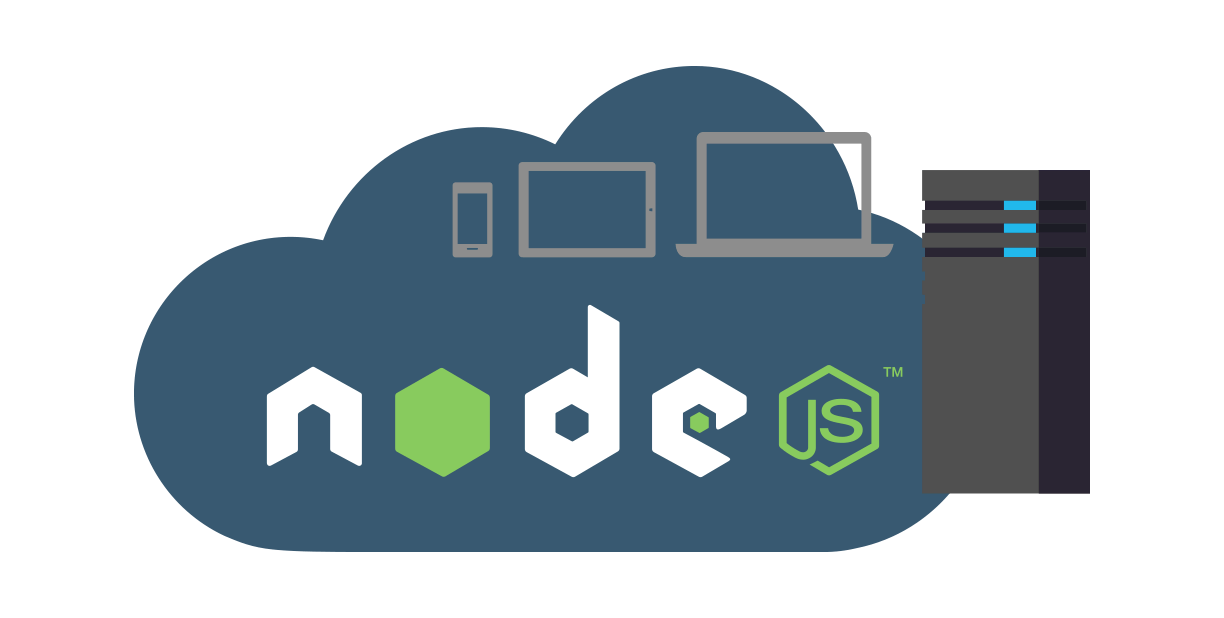
**4. Nhược điểm của MongoDB**

- Dữ liệu trong MongoDB không bị ràng buộc như RDBMS. Nhưng người sử dụng lưu ý cẩn thận mọi thao tác để không xảy ra các kết quả ngoài ý muốn gây ảnh hưởng đến dữ liệu.

- Một nhược điểm lo ngại là bộ nhớ của thiết bị. Chương trình này thường tốn bộ nhớ do dữ liệu được lưu dưới dạng key-value. Trong khi các collection chỉ khác về value nên sẽ lặp lại key dẫn đến thừa dữ liệu.

- Thông thường, dữ liệu thay đổi từ RAM xuống ổ cứng phải qua 60 giây thì chương trình mới thực hiện hoàn tất. Đây là nguy cơ bị mất dữ liệu nếu bất ngờ xảy ra mất điện trong vòng 60 giây đó.

**Node.js**



**Định nghĩa NodeJS**

NodeJS là mã nguồn mở chạy trên môi trường V8 JavaScript runtime (một trình thông dịch [JavaScript](https://wiki.tino.org/html-css-javascript-la-gi/)chạy cực nhanh trên trình duyệt Chrome). NodeJS giúp các nhà phát triển xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng.

NodeJS có thể được dùng để xây dựng các loại ứng dụng khác nhau như các ứng dụng dòng lệnh, ứng dụng web, ứng dụng trò chuyện theo thời gian thực, máy chủ [REST API](https://wiki.tino.org/restful-api-la-gi/),.. Tuy nhiên, NodeJS thường được dùng chủ yếu để xây dựng các chương trình mạng như máy chủ web, tương tự như PHP, Java hoặc ASP.NET.

Vì đây là mã nguồn mở nên cho phép ta có thể sử dụng miễn phí và đồng thời được liên tục được chỉnh sửa, cải tiến bởi cộng đồng các nhà phát triển toàn cầu.

**Lịch sử ra đời của NodeJS**

NodeJS xuất hiện lần đầu vào năm 2009 bởi Ryan Dahl. Phiên bản đầu tiên của NodeJS chỉ hỗ trợ hệ điều hành Linux và MacOS X. Ban đầu, việc phát triển và bảo trì được thực hiện bởi Dahl. Sau đó được Joyent, một công ty phần mềm và dịch vụ tài trợ.

Vào tháng 1 năm 2010, một trình quản lý của NodeJS đã được giới thiệu để giúp các lập trình viên dễ dàng xuất bản và chia sẻ mã nguồn của các gói NodeJS, đồng thời đơn giản hóa việc cài đặt, gỡ cài đặt và cập nhật các phiên bản.

Vào năm 2011, Microsoft và Joyent đã cùng nhau phát triển phiên bản [Windows](https://wiki.tino.org/he-dieu-hanh-windows-la-gi/) của NodeJS, mở rộng số lượng hệ điều hành mà công cụ này có thể hỗ trợ và cung cấp thêm nhiều lựa chọn cho các nhà phát triển

Nền tảng NodeJS đã được hợp nhất với JS Foundation vào năm 2019 để tạo thành OpenJS Foundation gồm những nền tảng giúp quản lý dự án phát triển mã nguồn mở, phân tán của NodeJS.

**Đặc điểm và ứng dụng của NodeJS**

**Đặc điểm của NodeJS**

* + - NodeJS không cần đợi API trả dữ liệu về, do đó mọi APIs nằm trong thư viện NodeJS đều không được đồng bộ.
    - Đây là một Platform chứ không phải là một Framework. Do đó, NodeJS cho phép bạn có thể xây dựng các website một cách độc lập và nhanh chóng hơn.
    - NodeJS có thể chạy trên đa nền tảng gồm: Window, MacOS, Linux.
    - NodeJS được xem là một máy chủ đơn luồng và không thể hỗ trợ đa luồng.
    - NodeJS không được xem là một ngôn ngữ lập trình, nên những người mới phải nắm chắc kiến thức lập trình căn bản như: các giao thức, Javascript,… mới có thể sử dụng NodeJS. Tuy nhiên, cộng đồng NodeJS thường rất lớn, sẵn sàng support cho bạn mọi lúc mọi nơi.
    - Phần core của NodeJS thường được biết bằng ngôn ngữ C++ nên nó hiệu năng và tốc độ xử lý tương đối cao. Nhờ vậy, hầu hết các ứng dụng NodeJS đều có khả năng đáp ứng được thời gian thực chạy trên đa nền tảng, đa thiết bị,…

**Những ứng dụng của NodeJS trong lập trình web**

* + - NodeJS có thể tạo, mở, đọc, ghi, xóa và đóng các tệp ngay khi đang truy cập trên máy chủ.
    - Xây dựng nội dung cho các trang web động.
    - Thực hiện thu thập dữ liệu theo yêu cầu cụ thể.
    - Thực hiện truy vấn, sửa, xóa, thêm các dữ liệu trong các hệ quản trị cơ sở như: Microsoft SQL Server, [MySQL](https://wiki.tino.org/mysql-la-gi/), [MongoDB](https://wiki.tino.org/mongodb-la-gi/), PostgreSQL.

**Điểm mạnh và hạn chế của NodeJS**

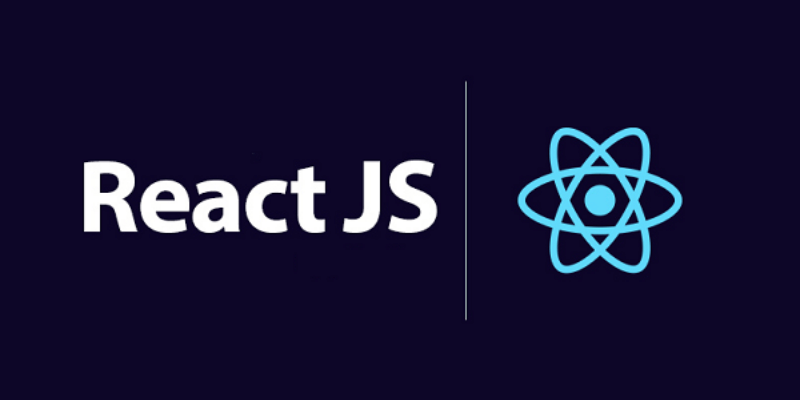
**Điểm mạnh**

* + - IO hướng sự kiện không đồng bộ giúp xử lý nhiều yêu cầu đồng thời.
    - Đáp ứng được những yêu cầu về thời gian thực.
    - Có tốc độ cực rất nhanh, đáp ứng được nhu cầu sử dụng của khách truy cập ‘khổng lồ’ trong thời gian ngắn.
    - Sử dụng JavaScript, một ngôn ngữ lập trình rất dễ học.
    - Chia sẻ cùng một đoạn mã với cả phía máy chủ và máy khách.
    - Npm và các module rất mạnh mẽ và vẫn đang tiếp tục phát triển.
    - Có một cộng đồng lớn mạnh, có nhiều mã được chia sẻ qua github
    - Tương thích với nhiều thiết bị, nhiều hệ điều hành như MacOS, Window, Linux,…

**Hạn chế**

* + - NodeJS không cung cấp khả năng mở rộng và không thể tận dụng lợi thế của nhiều lõi thường có trong phần cứng cấp máy chủ ngày nay.
    - Thao tác với cơ sở dữ liệu quan hệ là một khó khăn nếu bạn đang sử dụng NodeJS
    - Mỗi lần sử dụng lệnh gọi lại sẽ kết thúc với rất nhiều lệnh gọi lại lồng vào nhau.
    - Nếu không hiểu rõ về JavaScript, bạn sẽ gặp khó khăn với NodeJS
    - NodeJS không phù hợp với các tác vụ đòi hỏi nhiều CPU mà chỉ phù hợp với những I/O như máy chủ web.
    - Nếu bạn có một web hosting dùng chung, sẽ rất khó khăn nếu bạn tải lên một ứng dụng NodeJS. VPS và Dedicated server là một sự lựa chọn tốt hơn nhiều.

**ReactJS**



**ExpressJS**

**2.3. Công nghệ để host web server. ----------------------------------------------------**

**NGINX**



NGINX là một phần mềm web server mã nguồn mở, sử dụng kiến trúc hướng sự kiện (event-driven), bất đồng bộ (asynchronous). Mục tiêu ban đầu để phục vụ HTTP cache nhưng sau được áp dụng vào reverse proxy, HTTP load balancer và các giao thức truyền mail như IMAP4, POP3, và SMTP.

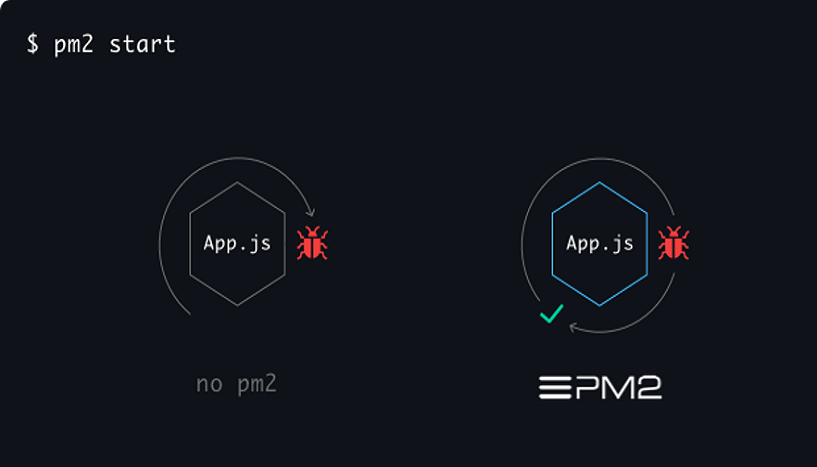
NGINX chính thức ra đời vào tháng 10/2014. Đây là phần mềm giúp server có tốc độ và khả năng mở rộng lớn nhất, đồng thời, xử lý và thao tác trên hàng nghìn kết nối cùng lúc. Do đó, rất nhiều “ông lớn” công nghệ hiện nay đều lựa chọn NGINX như Google, Adobe, Netflix, WordPress…

Về cơ bản, NGINX cũng hoạt động tương tự như các web server khác. Khi mở một trang web, trình duyệt sẽ liên hệ với server chứa website đó. Server sẽ tìm kiếm đúng file yêu cầu của website và gửi về cho bạn. Đây là một trình tự xử lý dữ liệu single – thread, nghĩa là các bước được thực hiện theo một trình tự duy nhất. Mỗi yêu cầu sẽ được tạo một thread riêng.

Tuy nhiên, NGINX hoạt động theo kiến trúc bất đồng bộ (asynchronous) hướng sự kiện (event driven). Nó cho phép các threads tương đồng được quản lý trong một tiến process. Mỗi process hoạt động sẽ bao gồm các thực thể nhỏ hơn, gọi là worker connections dùng để xử lý tất cả threads.

Worker connections sẽ gửi các yêu cầu cho worker process, worker process sẽ gửi nó tới master process, và master process sẽ trả lời các yêu cầu đó. Đó là lý do vì sao một worker connection có thể xử lý đến 1024 yêu cầu tương tự nhau. Nhờ vậy, NGINX có thể xử lý hàng ngàn yêu cầu khác nhau cùng một lúc.

**PM2**



PM2 là một trình quản lý các process (tiến trình) dành cho các ứng dụng Nodejs. Nó được viết bằng chính Nodejs và Shell. PM2 cũng được tích hợp bộ cân bằng tải (load balancer).

**Tính năng chính của PM2:**

* + - Quản lý các process, bao gồm tự động restart app khi bị chết hoặc reboot hệ thống.
    - Giám sát ứng dụng.
    - Khai báo cấu hình qua JSON file.
    - Quản lý log.
    - Cluster mode.
    - Chạy các kịch bản lệnh cho hệ thống.
    - Seamless updates.
    - Cho phép tích hợp các module cho hệ thốngo các process của server còn sống và reload/restart với zero downtime.

**NAT port**

**NAT**

Như chúng ta biết, trong hệ thống mạng máy tính thì quá trình chuyển đổi địa chỉ mạng (Network Address Translation - NAT) là công việc liên quan tới việc ghi lại các địa chỉ nguồn gốc/điểm tới của các gói dữ liệu vận chuyển qua Router/Firewall ta gọi là NAT.

**Port forwarding**

Port forwarding là quá trình chuyển một port cụ thể trong hệ thống mạng cục bộ LAN từ một client/terminal/node qua một client/terminal/node của một network khác, điều này sẽ cho phép clients/terminals/nodes bên ngoài có thể truy cập vào clients/terminals/nodes trong mạng LAN bằng cách sử dụng port đó từ bên ngoài qua Router/Modem/Firewall đã mở chức năng NAT.

**Port**

Khi chúng ta kết nối vào Internet thì có rất nhiều đường vận chuyển dữ liệu khác nhau được sử dụng, máy vi tính sẽ âm thầm điều hành một cách chính xác các công việc vận chuyển đó thông qua rất nhiều port khác nhau. VD: Giao thức MSN hay dùng các port: 6891-6900, Giao thức HTTP thì hay dùng port 80...

**ADSL Router/DSL Router**

Chúng ta cần lưu ý là các DSL/ADSL Router sẽ phân máy tính chúng ta ra làm 2 vòng:

- Vòng 1 thì là Private LAN, các máy tính trong Private LAN này thì gọi là các Local IP, Private IP. Các Private IP này nếu được modem cấp phát tự động thông qua chức năng DHCP thì gọi là IP động, khi ta Forward port thì cần điền vào một IP cố định còn gọi là IP tĩnh.

- Vòng 2 là WAN (Wide Area Network), tại đây thì các IP được nhà cung cấp dịch vụ như FPT, Viettel, VDC, Netnam cung cấp cho ta để kết nối ta ra Internet. Tuỳ theo gói dịch vụ ta sẽ có IP tĩnh (cố định) hay IP động (thay đổi mỗi khi tắt modem).

Các router này sẽ chịu trách nhiệm trong việc điều khiển lưu lượng dữ liệu tới các máy tính khác nhau connect vào, nó cũng sử dụng chức năng Port forward để điều khiển các gói dữ liệu tới các máy tính khác nhau thông qua các port. Các router này thường tích hợp thêm Firewall để ngăn chặn các cuộc gọi (luồng truy cập) in/out ngoài luồng không khai báo vì vậy khi chúng ta làm công việc Port forward chính là ta khai báo luồng truy cập để các gói dữ liệu không bị chặn lại khi download/upload torrent.

# THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

* Yêu cầu thiết kế
  + Liệt kê các yêu cầu đặt ra
  + Ghi cụ thể (có tính định lượng) các yêu cầu, chi tiết kỹ thuật.
* **Phân tích**
  + Phân tích rõ cách thức dẫn đến phương pháp thiết kế từ yêu cầu đã đặt ra
  + Nêu rõ ưu điểm và khuyết điểm của từng phương pháp, từ đó lựa chọn phương pháp phù hợp
* Vẽ sơ đồ khối tổng quát và **giải thích** (nếu mạch đơn giản thì lược bỏ phần này)
  + Phải giải thích rõ nhiệm vụ, chức năng từng khối
* Vẽ sơ đồ khối chi tiết và **giải thích**
  + Phải giải thích rõ nhiệm vụ, chức năng từng khối
* Tính toán và vẽ sơ đồ mạch chi tiết
  + Thiết kế, vẽ sơ đồ mạch chi tiết và tính toán từng khối đã nêu trong phần trên

# THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM (NẾU CÓ)

* Yêu cầu đặt ra cho phần mềm
  + Liệt kê các yêu cầu đặt ra
  + Ghi cụ thể (có tính định lượng) các yêu cầu, chi tiết kỹ thuật.
* **Phân tích**
  + Phân tích các yêu cầu để đưa ra phương pháp thực hiện chương trình
* Vẽ lưu đồ giải thuật tổng quát và **giải thích** (nếu giải thuật đơn giản thì lược bỏ phần này)
  + Phải giải thích rõ nhiệm vụ, chức năng từng phần
* Vẽ lưu đồ giải thuật chi tiết và **giải thích**
  + Phải giải thích rõ nhiệm vụ, chức năng từng phần

Yêu cầu đặt ra:

- Tìm hiểu, xây dựng server để nhận dữ liệu từ thiết bị gửi lên.

- Thiết kế giao diện web, hiển thị thông tin thiết bị ứng với mỗi user khác nhau và cho phép giám sát, điều khiển thiết bị thông qua internet.

- Lưu trữ thông tin người dùng và dữ liệu tương ứng vào database để phục vụ phân tích và đánh giá.

- Tìm hiểu về một ứng dụng MERN gồm MongoDB, Express, React, NodeJs. Ứng dụng để viết một Web API cho phép người dùng và quản trị viên có thể biết được thông tin từ thiết bị.

- Tìm hiểu cách để gửi dữ liệu lấy được từ MQTT đưa lên MongoDB database.

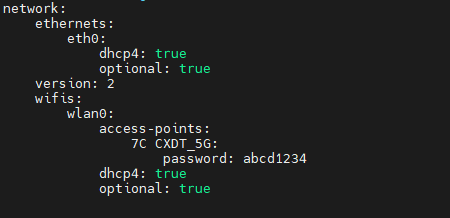
* Deploy Web Application lên Raspberry

Bước 1:

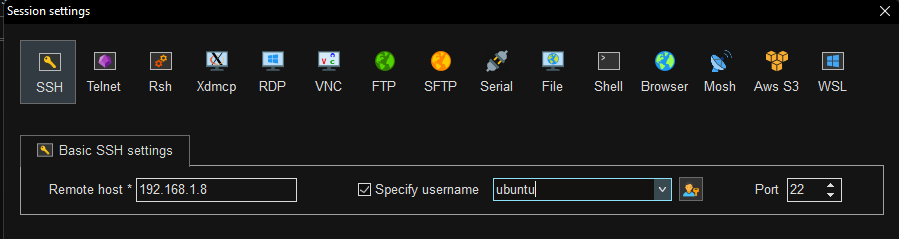
Thực hiện tải và cài đặt Ubuntu-server 20.04.4 LTS vào thẻ nhớ SD-card thông qua chương trình Raspberry Pi Imager do nhà sản xuất Raspberry Pi cung cấp.

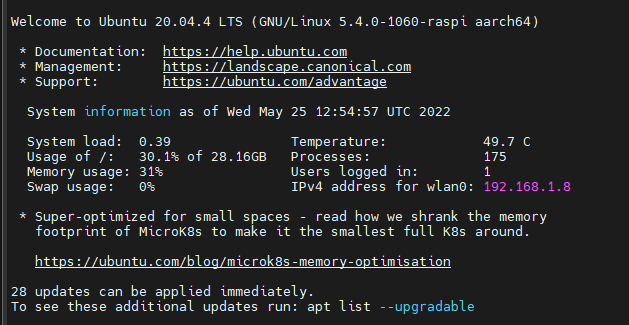


* + - Sau đó ta thực hiện đăng nhập wifi cho server bằng tên Wifi và mật khẩu qua tệp network-config



Bước 2: Sau khi khởi động server raspberry. Thực hiện truy cập vào server qua mạng nội bộ bằng giao thức SSH



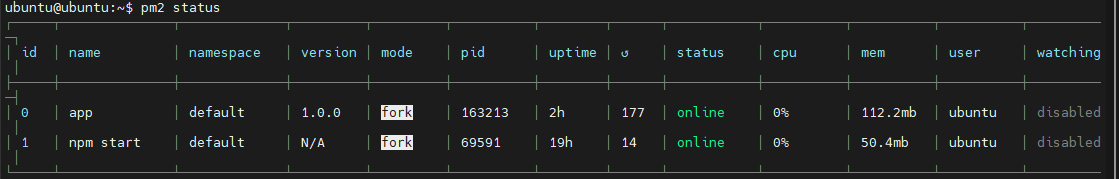


Bước 3: Thực hiện cài đặt phần mềm cần thiết hỗ trợ chạy web application như: MongoDB, NodeJs, Nginx, PM2,…

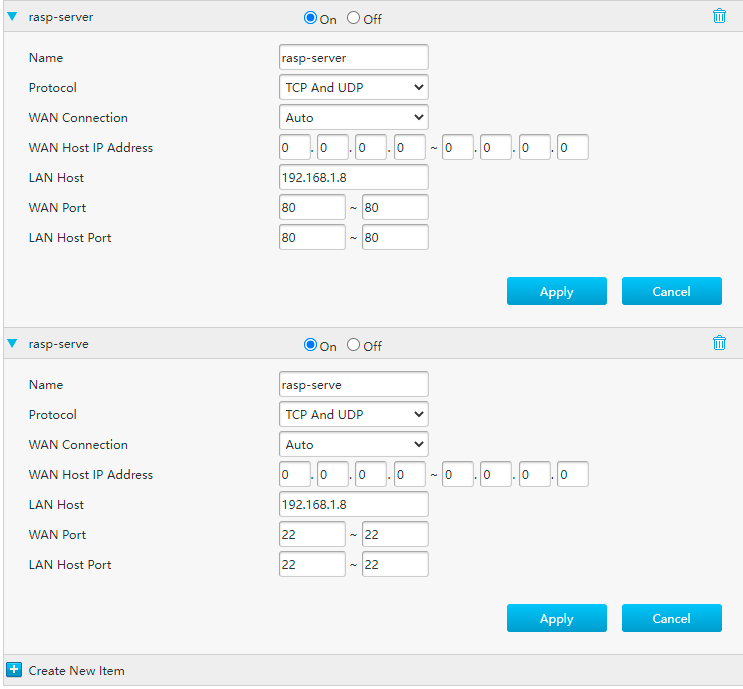
Bước 4: Thực hiện cài đặt MongoDB ở chế độ replica-set.

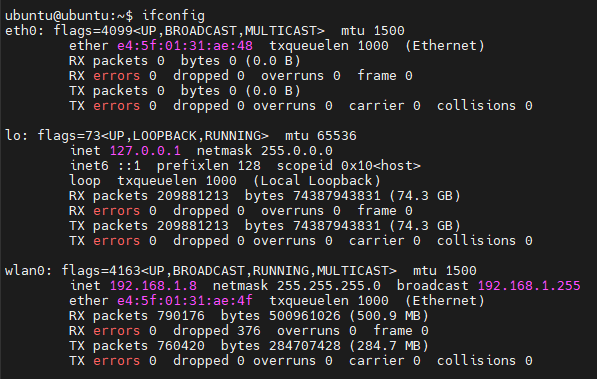
Bước 5: Copy folder chứa code của web application vào server. Cài đặt những framework bằng npm.

Bước 6: Chạy back-end và front-end của web bằng PM2.

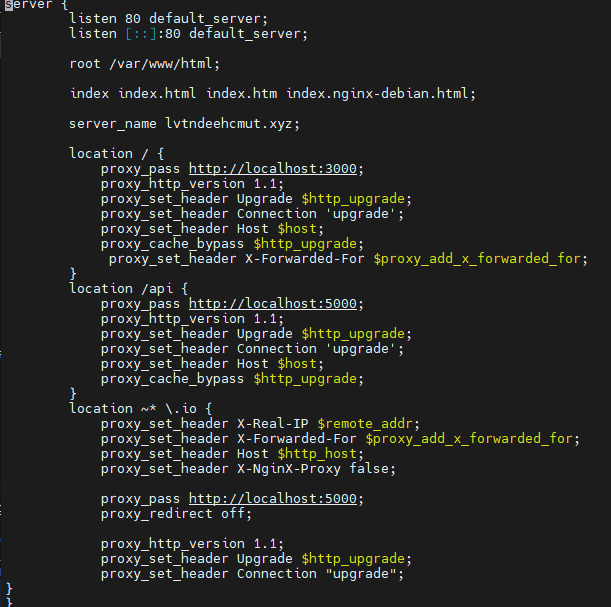


Bước 7: Thực hiện NAT port cho modem wifi.





Bước 8: Cấu hình Nginx tại file /etc/nginx/sites-available/default



Tại đây ta có thể thấy Nginx cho phép access từ port 80:

* + - Location “/” sẽ dẫn đến front-end của web tại địa chỉ <http://localhost:3000>
    - Location “/api” sẽ dẫn đến back-end của server tại địa chỉ <http://localhost:5000>
    - Location “~\* \.io” sẽ cho phép socket.io hoạt động thông qua và cùng port với back-end.

# KẾT QUẢ THỰC HIỆN

Trong phần này, sinh viên mô tả:

* Trình bày **cách thức đo đạc, thử nghiệm** 
  + Ghi rõ các thiết bị sử dụng và sơ đồ kết nối trong việc thử nghiệm
  + Ghi rõ các phần mềm sử dụng trong việc viết và thực thi chương trình
  + Ghi rõ cách bước tiến hành thử nghiệm (phần cứng và phần mềm)
* Trình bày số liệu đo đạc
  + Thực hiện thu thập số liệu trong nhiều trường hợp
  + Ghi rõ số liệu đo đạc thu được dưới hình thức bảng biểu, đồ thị …
* **Giải thích và phân tích về kết quả thu được**
  + Cần giải thích rõ ràng số liệu thu được trên các bảng biểu, đồ thị, dạng sóng …
  + Phân tích các số liệu để biết kết quả đã thực hiện là phù hợp, đạt yêu cầu

Nếu những bảng số liệu và kết quả mô phỏng quá nhiều, sinh viên có thể trình bày đưa vào phần Phụ Lục.

Ví dụ về hình minh họa: (dùng chức năng **Insert Caption** để tạo liên kết cho Danh sách hình minh họa)



Hình 5‑1 Kết quả thi công



Hình 5‑2 Kết quả mô phỏng

Ví dụ về Bảng số liệu

Bảng 1 Thông số hệ thống

|  |  |  |  |
| --- | --- | --- | --- |
| Thông số 1 | Thông số 2 | Thông số 3 | Thông số 4 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Sinh viên tóm tắt những điều rút ra được từ kết quả đề tài, những kinh nghiệm có được sau khi thực hiện đề tài. **Ưu và khuyết điểm** của kết quả nghiên cứu đề tài cũng được trình bày trong mục này. Sinh viên cần so sánh với mục tiêu đặt ra trong chương 1.

## Hướng phát triển

Sinh viên trình bày hướng phát triển và khả năng ứng dụng của đề tài

# TÀI LIỆU THAM KHẢO

Trong mục này, sinh viên liệt kê những tài liệu đã tham khảo khi thực hiện đề tài luận văn. Những nội dung trình bày ở mục trên có tham khảo tài liệu thì sinh viên cần ghi chú bằng chỉ số (ví dụ [1], [2]). Chỉ số này cần tương ứng danh mục tài liệu tham khảo. Sinh viên xem thêm hướng dẫn cách viết trích dẫn kiểu IEEE.

Ví dụ:

1. Tống Văn On, “Thiết kế mạch số với VHDL & Verilog”, Nhà xuất bản Lao động Xã Hội, 2007.
2. Altera Corp., “SDRAM Controller for Altera’s DE2/ DE1 boards”, [www.altera.com](http://www.altera.com)

<https://tino.org/vi/giao-thuc-mqtt-la-gi/>

<https://viblo.asia/p/nhung-dieu-can-biet-ve-http-ORNZq1wLZ0n>

<https://mona.media/nosql-la-gi/>

<https://vn.got-it.ai/blog/co-so-du-lieu-mongodb-la-gi-tong-quan-ve-mongodb>

<https://wiki.tino.org/nodejs-la-gi/>

<https://fptcloud.com/nginx/>

<https://viblo.asia/p/tong-quan-ve-pm2-3P0lPkkmZox>

<https://plpsoft.vn/30246-Tim-hieu-ve-Port-forwarding-Port-NAT-va-DMZ-tren-router>

# PHỤ LỤC

Trong phần này, sinh viên có thể trình bày:

* Những kết quả nghiên cứu bổ sung mà trong phần Kết quả luận văn chưa trình bày hết.
* Phần mã nguồn chương trình, sinh viên cũng có thể trình bày trong mục này. Để ngắn gọn, sinh viên chỉ đưa những mã nguồn chính vào phần Phụ lục.
* Sơ đồ toàn mạch chi tiết