TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN GIỮA KỲ

MÔN LẬP TRÌNH WEB NÂNG CAO

TÌM HIỂU VỀ GIAO THỨC WEB SOCKET VÀ XÂY DỰNG TRANG WEB CHAT TRỰC TUYẾN

Người hướng dẫn: **GV. NGUYỄN ĐÌNH TRUNG**

Người thực hiện: **TĂNG KIẾN TRUNG – 51900718**

ĐẶNG ĐĂNG TRÍ – 51900715

TRƯƠNG TUẤN THỊNH – 51900712

Lớp : **19050201**

Khoá : **23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN GIỮA KỲ

MÔN LẬP TRÌNH WEB NÂNG CAO

TÌM HIỂU VỀ GIAO THỨC WEB SOCKET VÀ XÂY DỰNG TRANG WEB CHAT TRỰC TUYẾN

Người hướng dẫn: **GV. NGUYỄN ĐÌNH TRUNG**

Người thực hiện: **TĂNG KIẾN TRUNG – 51900718**

ĐẶNG ĐĂNG TRÍ – 51900715

TRƯƠNG TUẤN THỊNH – 51900712

Lớp : **19050201**

Khoá : **23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Lời nói đầu tiên nhóm em xin gửi lời cảm ơn chân thành đến Thầy Nguyễn Đình Trung Giảng viên phụ trách giảng dạy thực hành môn học "Lập trình web nâng cao" của nhóm chúng em trong học kỳ II năm học 2021 này. Cảm ơn thầy đã giảng dạy và phụ trách chúng em trong môn học này, nhờ thầy mà chúng em được tiếp cận một môn học vô cùng thú vị, nhờ thầy mà chúng em tiếp thu được những kiến thức vô cùng quan trọng về kiến thức web nâng cao mà sau này trong quá trình học tập các môn tiếp theo lẫn quá trình đi làm là vô cùng cần thiết. Cảm ơn thầy đã tận tụy giảng dạy dù trong điều kiện học tập trực tuyến vì tình hình dịch bệnh đã có phần khó khăn vì không thể trực tiếp đứng lớp giảng dạy nhưng chúng em vẫn có thể tiếp thu bài giảng một cách đầy đủ và tốt nhất.

Sau cùng thì nhóm hy vọng sẽ nhận được những sự góp ý và đóng góp chân thành từ thầy, do nhóm còn nhiều hạn chế và giới hạn về mặt kiến thức, trình độ cho nên không thể tránh khỏi những sai sót. Chúng em sẽ tiếp thu những đóng góp, nhận xét từ thầy và tiếp tục hoàn thiện bản thân, vận dụng những nhận xét đó vào trong những môn học tiếp theo. Chúng em xin chúc thầy người luôn tận tụy dành thời gian cho sinh viên của mình có những ngày làm việc thật là hiệu quả và tràn đầy sức khỏe.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Nhóm xin cam đoan đây là bài báo cáo tiểu luận giữa kỳ của riêng chúng em và được sự hướng dẫn của GV. Nguyễn Đình Trung trong môn học "Lập trình web nâng cao". Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung tiểu luận giữa kỳ của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm 2022

Tác giả

(ký tên và ghi rõ họ tên)

Tăng Kiến Trung

Đặng Đăng Trí

Trương Tuấn Thịnh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm 2021
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm 2021
(kí và ghi họ tên)

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

CHƯƠNG 1 – PHẦN CHUNG

1. Socket?

1.1 Socket là gì?

Socket là gì? – Socket là một điểm cuối (**end-point**) của một liên kết giao tiếp hai chiều (two-way communication) giữa hai chương trình chạy trên mạng. Nghĩa là một socket được sử dụng để cho phép 1 process nói chuyện với 1 process khác.

Các lớp Socket được sử dụng để tiến hành kết nối giữa client và server. Nó được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.

1.2 Socket sử dụng như thế nào?

Sau khái niệm socket là gì? Thì chúng ta sẽ đến chức năng của socket, nó được áp dụng trong quá trình làm việc, nếu có thể chạy nhiều socket cùng một lúc nên công việc của chúng ta sẽ nhanh hơn, nâng cao hiệu suất làm việc.

"Socket được hỗ trợ trong hầu hết các hệ điều hành như Windows, Linux. Và được sử dụng trong nhiều ngôn ngữ lập trình khác nhau."

Socket hoạt động như thế nào?

Socket giúp lập trình viên kết nối các ứng dụng để truyền và nhận dữ liệu trong môi trường có kết nối Internet bằng cách sử dụng phương thức TCP IP và UDP.

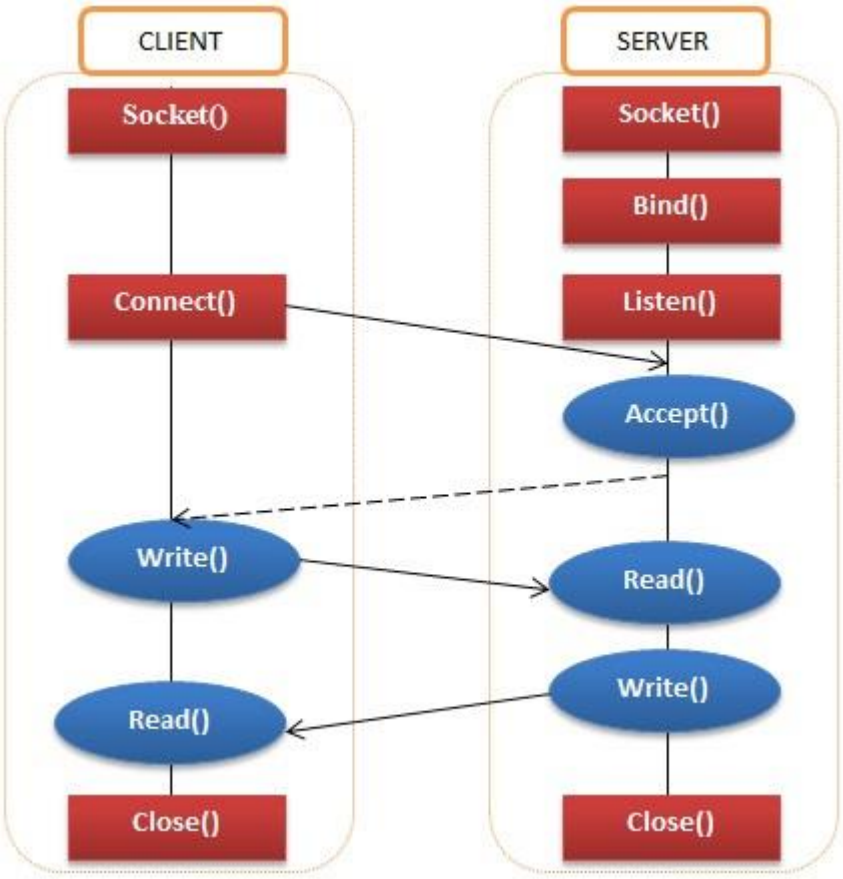
Khi cần trao đổi dữ liệu cho nhau thì 2 ứng dụng cần phải biết thông tin IP và port bao nhiêu của ứng dụng kia.

Có rất nhiều dạng socket khác nhau phụ thuộc vào sự khác biệt giữa cách truyền dữ liệu (protocol). Dạng phổ biến nhất là TCP và UDP.

1.3 Phân loại Socket

1.3.1 Stream Socket là gì?

Dựa trên giao thức TCP(Tranmission Control Protocol), stream socket thiết lập giao tiếp 2 chiều theo mô hình client và server. Được gọi là socket hướng kết nối.



Hình 1. Cách hoạt động của Stream Socket

Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng tuần tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

Chúng cung cấp luồng dữ liệu không trùng lặp và có cơ chế được thiết lập tốt để tạo và hủy kết nối và phát hiện lỗi. Nếu việc gửi các gói dữ liệu gặp gián đoạn hoặc không thể gửi được thì người gửi sẽ nhận được một gói tin báo lỗi.

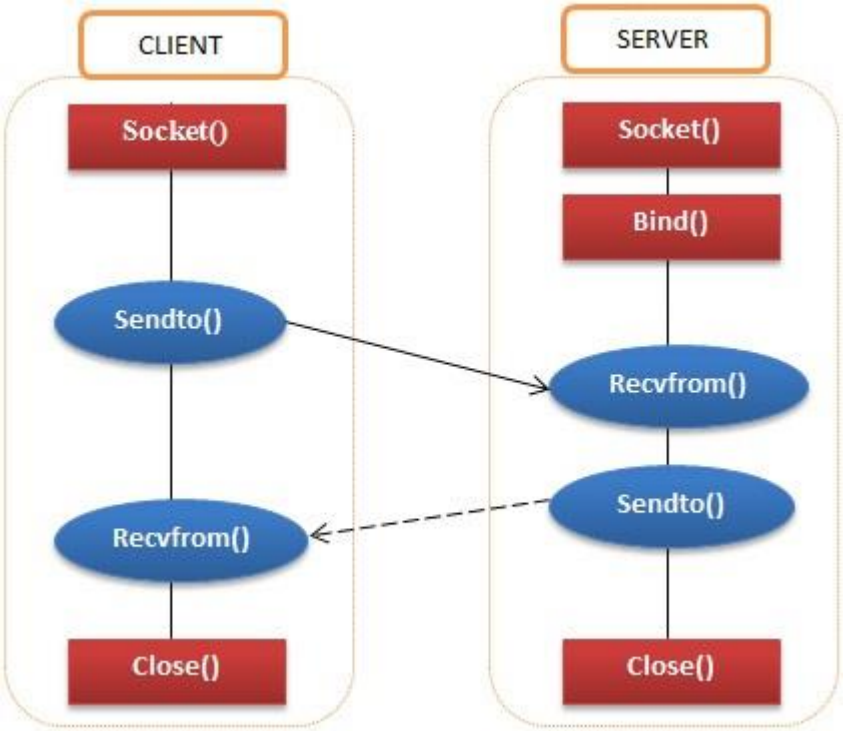
Tóm tắt đặc điểm:

- Có một đường kết nối (địa chỉ IP) giữa 2 tiến trình.
- Một trong hai tiến trình kia phải đợi tiến trình này yêu cầu kết nối.
- Mô hình client /sever thì sever lắng nghe và chấp nhận từ client.

- Mỗi thông điệp gửi phải có xác nhận trả về.
- Các gói tin chuyển đi tuần tự.

1.3.2 Datagram Socket

Dựa trên giao thức UDP(User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 process. Tức là nó cung cấp connection-less point cho việc gửi và nhận packets. Gọi là socket không hướng kết nối



Hình 2. Cách hoạt động của Datagram Socket

Do không yêu cầu thiết lập kết nối, không phải có những cơ chế phức tạp. Nên tốc độ giao thức khá nhanh, thuận tiện cho các ứng dụng truyền dữ liệu nhanh như chat, game online…

Tóm tắt đặc điểm:

- Hai tiến trình liên lạc với nhau không kết nối trực tiếp
- Thông điệp gửi đi phải kèm theo thông điệp người nhận

- Thông điệp có thể gửi nhiều lần
- Người gửi không chắc chắn thông điệp đến tay người nhận.
- Thông điệp gửi sau có thể đến trước và ngược lại.
- Để có thể thực hiện các cuộc giao tiếp, một trong 2 quá trình phải công bố port của socket mà mình đang sử dụng.

2. Websocket là gì?

HTTP là gì?

Trước khi trả lời câu hỏi Websocket thì ta hãy nói sơ qua về HTTP. Đây sẽ là cơ sở để ta so sánh, đánh giá và hiểu hơn vì sao Websocket lại vượt trội hơn HTTP trong việc xây dựng các ứng dụng real-time.

HTTP là giao thức truyền tải một chiều. Trong giao thức này, máy khách sẽ gửi yêu cầu và máy chủ gửi phản hồi ngược lại.

Ví dụ, khi người dùng gửi 1 yêu cầu đến máy chủ theo dạng HTTP hoặc HTTPS (HTTP có mức độ bảo mật cao hơn). Sau khi nhận được yêu cầu, máy chủ sẽ gửi phản hồi ngược lại cho máy khách. Khi thực hiện phản hồi xong, kết nối sẽ bị đóng lại.

Vì vậy, mỗi lần gửi yêu cầu HTTP sẽ tạo một kết nối mới giữa máy chủ và máy khách.

Websocket là giao thức, công nghệ hỗ trợ giao tiếp hai chiều giữa client và server để tạo một kết nối trao đổi dữ liệu. Giao thức này không sử dụng HTTP mà thực hiện nó qua TCP socket để tạo một kết nối hiệu quả và ít tốn kém.. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.

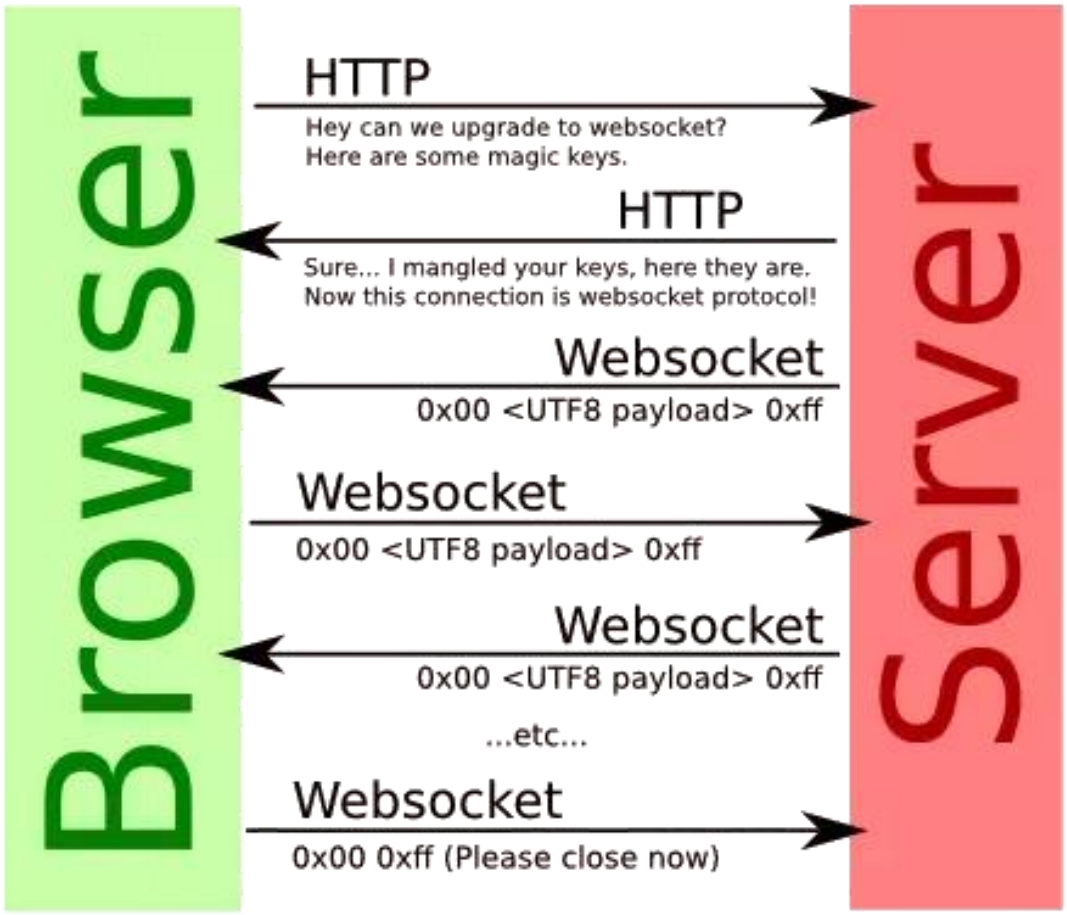
WebSocket là một giao thức giao tiếp máy tính, hỗ trợ các channel giao tiếp full-duplex (song công toàn phần) qua một kết nối TCP. Giao thức WebSocket được IETF chuẩn hóa RFC 6455 vào năm 2011. Hiện nay, API WebSocket trong Web IDL cũng đang được chuẩn hóa bởi W3C.

WebSockets mới xuất hiện trong HTML5, là một kỹ thuật Reverse Ajax. WebSockets cho phép các kênh giao tiếp song song hai chiều và hiện đã được hỗ trợ trong nhiều trình duyệt (Firefox, Google Chrome và Safari). Kết nối được mở thông qua một HTTP request (yêu cầu HTTP), được gọi là liên kết WebSockets với những header đặc biệt. Kết nối được duy trì để bạn có thể viết và nhận dữ liệu bằng JavaScript như khi bạn đang sử dụng một TCP socket đơn thuần.

Về bản chất, WebSocket khác với HTTP, mặc dù cả giao thức đều ở trên layer 7 của mô hình OSI, và cùng phụ thuộc vào TCP ở layer 4. Tuy nhiên, RFC 6455 cho biết WebSocket "được thiết kế để hoạt động trên các cổng HTTP 443 và 80, cũng như để hỗ trợ proxy HTTP và làm trung gian". Do đó, WebSocket hoàn toàn có khả năng tương thích với giao thức HTTP. Để có được sự tương thích này, WebSocket handshake sẽ sử dụng một header HTTP Upgrade để thay đổi giao thức HTTP thành WebSocket.

Giao thức WebSocket hỗ trợ tương tác giữa một trình duyệt web (hay ứng dụng client) với một web server, nhưng chi phí sẽ thấp hơn so với các lựa chọn half-duplex khác như HTTP polling. Do đó, việc truyền dữ liệu theo thời gian thực với server sẽ trở nên hiệu quả hơn.

Vậy cách hỗ trợ tương tác trên của WebSocket là gì? WebSocket sẽ cung cấp một cách được tiêu chuẩn để server có thể gửi content đến client mà không cần được request bởi client. Đồng thời cho phép các thông báo được truyền qua lại trong khi vẫn giữ cho kết nối được mở. Từ đó tạo nên một giao tiếp hai chiều giữa client và server.



Hình 3. Minh họa qui trình hoạt động của Websocket

Các giao tiếp này thường được xử lý qua port TCP 443 (hoặc port 80 với những kết nối không được bảo mật). Do đó, các môi trường có thể dễ dàng chặn những kết nối non-web bằng firewall. Bên cạnh đó, các giao tiếp browser-server không tiêu chuẩn cũng có thể được thiết lập bằng các công nghệ stopgap như Comet.

WebSockets hỗ trợ phương thức giao tiếp 2 chiều giữa client và server thông qua TCP (port 80 và 443). Theo phân tích từ http://websocket.org/quantum.html, WebSockets có thể giảm kích thước của HTTP header lên đến 500 – 1000 lần, giảm độ trễ của network lên đến 3 lần. Do đó, hỗ trợ tốt hơn đối với các ứng dụng web apps real – time.

- Dữ liệu truyền tải thông qua giao thức HTTP (thường dùng với kĩ thuật AJAX) chứa nhiều dữ liệu không cần thiết trong phần header. Một header request/response của HTTP có kích thước khoảng 871 byte, trong khi với WebSocket, kích thước này chỉ là 2 byte (sau khi đã kết nối). Vậy giả sử bạn làm một ứng dụng game có thể tới 10,000 người chơi đăng nhập cùng lúc, và mỗi giây họ sẽ gửi/nhận dữ liệu từ server. Hãy so sánh lượng dữ liệu header mà giao thức HTTP và WebSocket trong mỗi giây:
- HTTP: 871 x 10,000 = 8,710,000 bytes = 69,680,000 bits per second (66 Mbps)
- WebSocket: 2 x 10,000 = 20,000 bytes = 160,000 bits per second (0.153 Kbps) Như chúng ta thấy chỉ riêng phần header thôi cũng đã chiếm một phần lưu lượng đáng kể với giao thức HTTP truyền thống.

WebSocket có công dụng gì?

WebSocket là một phương thức giúp máy trạm và máy chủ có thể giao tiếp thời gian thực 2 chiều với nhau.

Khả năng giảm độ trễ của mạng đến mức tối đa và vượt trội hơn HTTP. Nếu bạn phát triển các ứng dụng thời gian thực như: sàn tiền ảo, Game, chat hay video call,… sử dụng WebSocket sẽ mang lại trải nghiệm tối ưu nhất cho người dùng.

Khả năng giảm kích thước header HTTP lên đến 1000 lần! Nếu một request HTTP có header nặng 871 byte, khi sử dụng WebSocket, khối lượng header của request chỉ còn 2 byte.

2.1 Giao thức bắt tay của WebSocket

Giao thức Handshake

Để thiết lập một giao thức WebSocket, trước tiên client sẽ gửi một WebSocket handshake request. Sau đó server sẽ trả về một WebSocket handshake response như ví dụ ở dưới.

Client request (giống với HTTP, mỗi dòng sẽ kết thúc bằng \r\n và thêm một khoảng trắng bắt buộc ở cuối:

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Server response:(Server Architecture)

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
```

Giao thức handshake sẽ bắt đầu bằng một HTTP request/response, cho phép server xử lý các kết nối HTTP cũng như kết nối WebSocket ở trên cùng một cổng. Khi kết nối đã được thiết lập, giao tiếp sẽ chuyển sang giao thức nhị phân hai chiều, không phù hợp với HTTP.

Bên cạnh các header Upgrade, client cũng sẽ gửi một header Sec-WebSocket-Key chứa các byte ngẫu nhiên đã được mã hóa base64, và server sẽ phản hồi bằng một hash của key trong header Sec-WebSocket-Accept. Việc này nhằm ngăn chặn caching proxy

gửi lại giao tiếp WebSocket trước đó. Đồng thời không cung cấp bất kỳ xác thực, quyền riêng tư hay tính toàn vẹn nào. Hàm hash sẽ append chuỗi cố định (một UUID) 258EAFA5-E914-47DA-95CA-C5AB0DC85B11 vào giá trị từ header Sec-WebSocket-Key (không được giải mã từ base64). Cùng với đó là áp dụng hàm băm (hashing fucntion) SHA-1 và giải mã kết quả bằng base64.

Để xác nhận việc kết nối, client sẽ gửi một giá trị Sec-WebSocket-Key được mã hóa bằng Based64 đến server. Sau đó bên server sẽ thực hiện:

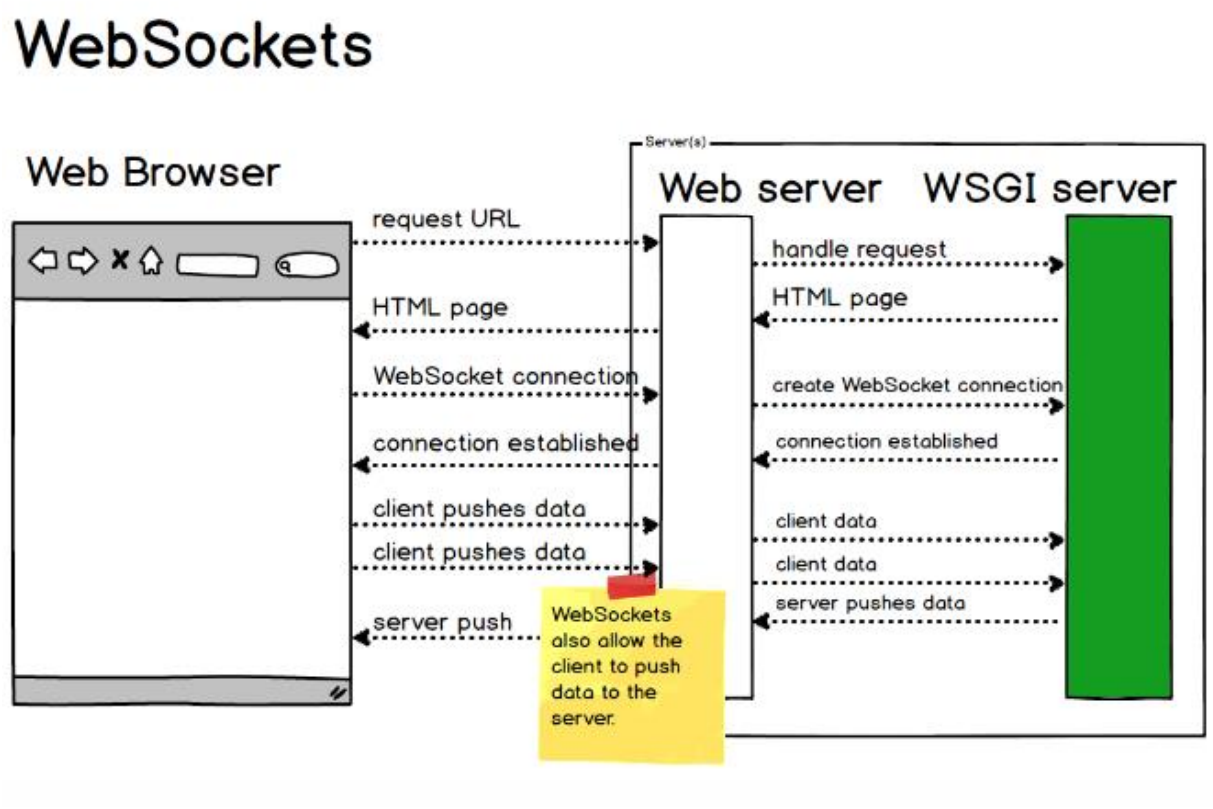
- Nối thêm chuỗi cố định là "258EAFA5-E914-47DA-95CA-C5AB0DC85B11″ vào Sec-WebSocket-Key để được chuỗi mới là "x3JJHMbDL1EzLkh9GBhXDw==258EAFA5-E914-47DA-95CA-C5AB0DC85B11″.
- Thực hiện mã hóa SHA-1 chuỗi trên để được "1d29ab734b0c9585240069a6e4e3e91b61da1969″.
- Mã hóa kết quả vừa nhận được bằng Base64 để được "HSmrc0sMlYUkAGmm5OPpG2HaGWk=".
- Gửi response lại client kèm với giá trị Sec-WebSocket-Accept chính là chuỗi kết quả vừa tạo ra.

Sau khi kết nối được thiết lập thành công, client và server có thể gửi dữ liệu WebSocket hoặc text frame qua lại trong chế độ full-duplex. Dữ liệu được frame tối thiểu, với một header nhỏ ở trước payload. Việc truyền WebSocket được mô tả như các "messages". Ở đó, mỗi message có thể được split trên nhiều data frame. Việc này sẽ cho phép gửi các message khi có sẵn dữ liệu ban đầu, nhưng độ dài của message chưa được xác định (nó sẽ gửi từng data frame cho đến khi kết thúc với bit FIN).

Ta cũng có thể sử dụng các phần mở rộng của giao thức này để ghép (multiplex) nhiều stream đồng thời. Chẳng hạn như để tránh việc sử dụng duy nhất một socket cho payload lớn.

Client sẽ kiểm tra status code (phải bằng 101) và Sec-WebSocket-Accept xem có đúng với kết quả mong đợi không và thực hiện kết nối.

2.2 Cấu trúc WebSocket



Hình 4. Minh họa quá trình hoạt động WebSocket

Giao thức chuẩn thông thường của WebSocket là ws:// , giao thức secure là wss:// . Chuẩn giao tiếp là String và hỗ trợ buffered arrays và blobs.

2.2.1 Các thuộc tính của WebSocket

THUỘC TÍNH	**MÔ TẢ**
readyState	Diễn tả trạng thái kết nối. Nó có các giá trị sau:

	• Giá trị 0: kết nối vẫn chưa được thiết lập (WebSocket.CONNECTING) • Giá trị 1: kết nối đã thiết lập và có thể giao tiếp (WebSocket.OPEN) • Giá trị 2: kết nối đang qua handshake đóng (WebSocket.CLOSING) • Giá trị 3: kết nối đã được đóng (WebSocket.CLOSED)
bufferedAmount	Biểu diễn số byte của UTF-8 mà đã được xếp hàng bởi sử dụng phương thức send()

2.2.2 Các sự kiện WebSocket

SỰ KIỆN	**EVENT HANDLER**	**MÔ TẢ**
open	onopen	Khi một WebSocket chuyển sang trạng thái mở, "onopen" sẽ được gọi.
message	onmessage	Khi WebSocket nhận dữ liệu từ Server.
error	onerror	Có bất kỳ lỗi nào trong giao tiếp.
close	onclose	Kết nối được đóng. Những sự kiện được truyền cho "onclose" có ba tham số là "code", "reason", và "wasClean".

2.2.3 Các phương thức của WebSocket

PHƯƠNG THỨC	MÔ TẢ
send()	send(data) gửi dữ liệu tới server. Message data là string, ArrayBuffer, blob.
close()	Đóng kết nối đang tồn tại.

2.3 Tại sao lại cần WebSocket

Hiện tại, có 2 phương thức để client – server của web apps có thể giao tiếp được với nhau, đó là:

2.3.1 AJAX POLLING

Phương thức này có nghĩa là client gửi request liên tục lên server để lấy data thông qua AJAX sau những khoảng thời gian nhất định. Điều này bắt buộc server phải trả data về dữ liệu, cho dù có dữ liệu mới hay không.

Ví dụ cho các trường hợp này chính là tracking của google, thông thường sẽ được update sau 15'…

2.3.2 AJAX LONG POLLING

Phương thức này cũng tương tự như phương thức trên, nhưng thay vì server trả về reponse cho client sau khi nhận request thì ở phương thức này, server chỉ trả về response khi có dữ liệu mới.

Cả 2 phương thức này đều thông qua giao thức HTTP do đó sẽ bao gồm rất nhiều thông tin header (đồng nghĩa với chậm). Ngoài ra cả phương thức trên đều không hỗ trợ đồng song song giữa client server. Do đó, WebSocket ra đời để mang đến phương thức giao tiếp real time tốt hơn dành cho client – server, hỗ trợ song song – cả client và server đều có thể gửi và nhận request cùng một thời điểm.

Đặc điểm

Cấu trúc: hỗ trợ chuẩn giao thức mới: ws:// cho chuẩn thông thường và wss:// cho chuẩn secure (tương tự http:// và https://)

Message data types: chuẩn giao tiếp là String, hiện tại đã hỗ trợ buffered arrays và blobs. Vẫn chưa hỗ trợ JSON (tuy nhiên có thể serialize thành String)

2.4 Ưu nhược điểm Websocket

Ưu điểm

- WebSockets cung cấp khả năng giao tiếp hai chiều mạnh mẽ, có độ trễ thấp và dễ xử lý lỗi. Không cần phải có nhiều kết nối như phương pháp Comet long-polling và cũng không có những nhược điểm như Comet streaming.
- WebSocket cung cấp giao thức giao tiếp hai chiều mạnh mẽ. Nó có độ trễ thấp và dễ xử lý lỗi. Websocket thường được sử dụng cho những trường hợp yêu cầu real time như chat, hiển thị biểu đồ hay thông tin chứng khoán.
- API cũng rất dễ sử dụng trực tiếp mà không cần bất kỳ các tầng bổ sung nào, so với Comet, thường đòi hỏi một thư viện tốt để xử lý kết nối lại, thời gian chờ timeout, các AJAX request (yêu cầu AJAX), các tin báo nhận và các dạng truyền tải tùy chọn khác nhau (AJAX long-polling và jsonp polling).
- Các gói tin (packets) của Websocket nhẹ hơn HTTP rất nhiều. Nó giúp giảm độ trễ của network nhiều lần.
- Đây là một công nghệ được phát triển nhằm mục đích khắc phục độ trễ của HTTP. Vì vậy, WebSocket có độ trễ thấp.
- Hỗ trợ giảm header xuống mức tối đa.
- Truyền và nhận dữ liệu trực tiếp, không cần thông qua các tầng bổ sung.
- Hầu hết các trình duyệt hiện nay đều hỗ trợ giao thức WebSocket. Trong đó có cả Google Chrome, Firefox, Microsoft Edge, Internet Explorer, Safari và Opera.

Nhược điểm:

- WebSocket còn mới và là tính năng mới của HTML5, chưa thể tương thích với toàn bộ các trình duyệt web, chưa được tất cả các trình duyệt hỗ trợ..
- Không có phạm vi yêu cầu nào. Do WebSocket là một TCP socket chứ không phải là HTTP request, nên không dễ sử dụng các dịch vụ có phạm vi-yêu cầu, như SessionInViewFilter của Hibernate. Hibernate là một framework kinh điển cung cấp một bộ lọc xung quanh một HTTP request. Khi bắt đầu một request, nó sẽ thiết lập một contest (chứa các transaction và liên kết JDBC) được ràng buộc với luồng request. Khi request đó kết thúc, bộ lọc hủy bỏ contest này.
- Truyền tải dữ liệu bằng WebSocket sẽ có thể phát sinh một số lỗi.

2.5 WebSockets API

2.5.1 Giới thiệu WebSockets API (dựa trên JavaScripts API và HTML5).

Kiểm tra trình duyệt có hỗ trợ WebSockets hay không

Việc đầu tiên cần làm khi làm việc với WebSockets trên client là kiểm tra WebSockets có được trình duyệt hỗ trợ hay không:

```
if ('WebSocket' in window){
  /* WebSocket is supported. You can proceed with your code*/
} else {
  /*WebSocket
```

Đóng/mở WebSockets

Giả sử trình duyệt hỗ trợ WebSockets, chúng ta bắt đầu khởi tạo và mở socket.

Khởi tạo WebSockets

var connection = new WebSocket('ws://example.org:12345/myapp');

hoặc với WebSockets secure

var connection = new WebSocket('wss://example.org:12345/myapp');

Mở connection đến server

```
connection.onopen = function(){
  /*Send a small message to the console once the connection is established */
  console.log('Connection open!');
}
```

Đóng WebSockets

```
connection.onclose = function(){
  console.log('Connection closed');
}
```

hoặc

```
connection.close();
```

Ngoài ra, chúng ta cũng có thể kiểm tra trường hợp có lỗi xảy ra

```
connection.onerror = function(error){
  console.log('Error detected: ' + error);
}
```

2.5.2 Gửi – nhận message

Gửi message đến server

```
var message = {
'name': 'bill murray',
'comment': 'No one will ever believe you'
};
connection.send(JSON.stringify(message));
```

Nhận message từ server

```
connection.onmessage = function(e){
  var server_message = e.data;
  console.log(server_message);
}
```

2.5.3 Thiết lập kết nối WebSocket

Vậy cách để thiết lập kết nối WebSocket là gì? Trước hết, WebSocket Open Handshake không sử dụng lược đồ http:// hay https://, vì chúng không tuân theo giao thức HTTP. Thay vào đó, URI WebSocket sẽ sử dụng giao thức ws: (hay wss: với WebSocket bảo mật). Phần còn lại của URI sẽ tương tự với HTTP URI: gồm host, port, path và bất kỳ tham số truy vấn nào.

"ws:" "//" host [":" port] path ["?" query]

"wss:" "//" host [":" port] path ["?" query]

Các kết nối WebSocket chỉ có thể được thiết lập cho những URI tuân theo lược đồ này. Do đó, khi thấy một URI với lược đồ ws:// (hoặc wss://), cả client lẫn server đều phải tuân theo giao thức WebSocket.

Kết nối WebSocket được thiết lập bằng cách upgrading một cặp HTTP request/repsonse. Nếu một client có hỗ trợ WebSocket muốn thiết lập một kết nối, client nãy sẽ gửi một HTTP request có bao gồm các header bắt buộc sau:

Connection: Upgrade

Header Connection thường kiểm soát việc mở kết nối mạng sau khi transaction kết thúc. Giá trị phổ biến của header này là keep-alive, giúp đảm bảo kết nối được liên tục cho những request sau đến cùng server. Trong WebSocket opening handshake, ta đặt header thành Upgrade, cho biết rằng ta muốn giữ cho kết nối luôn mở, và dùng cho những request non-HTTP.

Upgrade: websocket

Header Upgrade được client sử dụng để yêu cầu server chuyển sang các giao thức khác trong danh sách, sắp xếp theo thứ tự ưu tiên giảm dần. Trong ví dụ này là websocket, cho biết rằng client muốn thiết lập một kết nối WebSocket.

Sec-WebSocket-Key: q4xkcO32u266gldTuKaSOw==

Sec-WebSocket-Key là một giá trị ngẫu nhiên, dùng một lần (nonce), được tạo ra bởi client. Giá trị của header là một giá trị 16 byte được mã hóa base64, được tạo ngẫu nhiên

Sec-WebSocket-Version: 13

Cho biết phiên bản WebSocket duy nhất được chấp nhận là 13. Bất kỳ phiên bản nào khác được liệt kê trong header này đều không hợp lệ.

Các header trên kết hợp với nhau sẽ tạo ra một request HTTP GET từ client đến một URI ws:// như sau:

GET ws://example.com:8181/ HTTP/1.1

Host: localhost:8181

Connection: Upgrade

Pragma: no-cache

Cache-Control: no-cache

Upgrade: websocket

Sec-WebSocket-Version: 13

Sec-WebSocket-Key: q4xkcO32u266gldTuKaSOw==

Sau khi client gửi request ban đầu để mở một kết nối WebSocket, nó sẽ đợi server reply. Reply từ server phải chứa một response code HTTP 101 Switching Protocols. Response này cho biết server đang chuyển sang giao thức mà client đã yêu cầu ở trong request header Upgrade. Bên cạnh đó, server phải bao gồm cả các header HTTP xác thực các kết nối được upgrade thành công:

HTTP/1.1 101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: fA9dggdnMPU79lJgAE3W4TRnyDM=

Connection: Upgrade:

Đảm bảo rằng kết nối đã được upgrade.

Upgrade: websocket

Đảm bảo rằng kết nối đã được upgrade.

Sec-WebSocket-Accept: fA9dggdnMPU79lJgAE3W4TRnyDM=`

Sec-WebSocket-Accept là một giá trị được mã hóa base64 và SHA-1 hashed. Ta có thể tạo giá trị này bằng cách nối các client Sec-WebSocket-Key nonce và giá trị tĩnh 258EAFA5-E914-47DA-95CA-C5AB0DC85B11 được định nghĩa ở trong RFC 6455. Mặc dù Sec-WebSocket-Key và Sec-WebSocket-Accept có vẻ hơi phức tạp, nhưng chúng phải tồn tại để client và server có thể biết được đối tác có hỗ trợ WebSocket.

Tiếp đến, sau khi client nhận được server response, kết nối WebSocket sẽ được mở để bắt đầu việc truyền dữ liệu.

2.5.4 Cân nhắc về bảo mật

Khác với các cross-domain HTTP request, WebSocket request không bị giới hạn bởi chính sách Same-Origin. Do đó, các server WebSocket phải xác thực header "Origin" so với origin dự kiến trong quá trình thiết lập kết nối, nhằm tránh tấn công Cross-Site WebSocket Hijacking (một loại tấn công tương tự như Cross-Site Request Forgery – CRF). Loại tấn công này có thể xảy ra khi kết nối được xác thực bằng cookies hay HTTP. Do đó, tốt nhất là hãy sử dụng token hay những cơ chế bảo mật tương tự để xác thực kết nối WebSocket khi chuyển các dữ liệu riêng tư qua WebSocket.

2.5.5 Proxy traversal

Client triển khai giao thức WebSocket muốn kiểm tra xem user agent có được cấu hình để sử dụng proxy khi kết nối đến host đích và host hay không. Nếu có thì sẽ sử dụng phương thức HTTP CONNECT để thiết lập tunnel.

Mặc dù bản thân giao thức WebSocket không hề biết về các proxy server hay firewall, nó có giao thức handshake tương thích với HTTP. Do đó sẽ cho phép các HTTP server chia sẻ các port HTTP và HTTPS của họ (80 và 443) bằng một WebSocket gateway hay server. Giao thức WebSocket xác định một tiền tố ws:// và wss:// để xác định kết nối WebSocket và WebSocket Secure. Cả hai lược đồ đều sử dụng cơ chế HTTP uprade để upgrade lên giao thức WebSocket.

Một số proxy server có thể hoạt động tốt với WebSocket, tuy nhiên một số khác cũng có thể ngăn WebSocket hoạt động bình thường. Từ đó gây ảnh hưởng đến các kết

nối. Đôi khi còn có thể yêu cầu thêm một số proxy server bổ sung, hoặc yêu cầu upgrade để có thể hỗ trợ WebSocket.

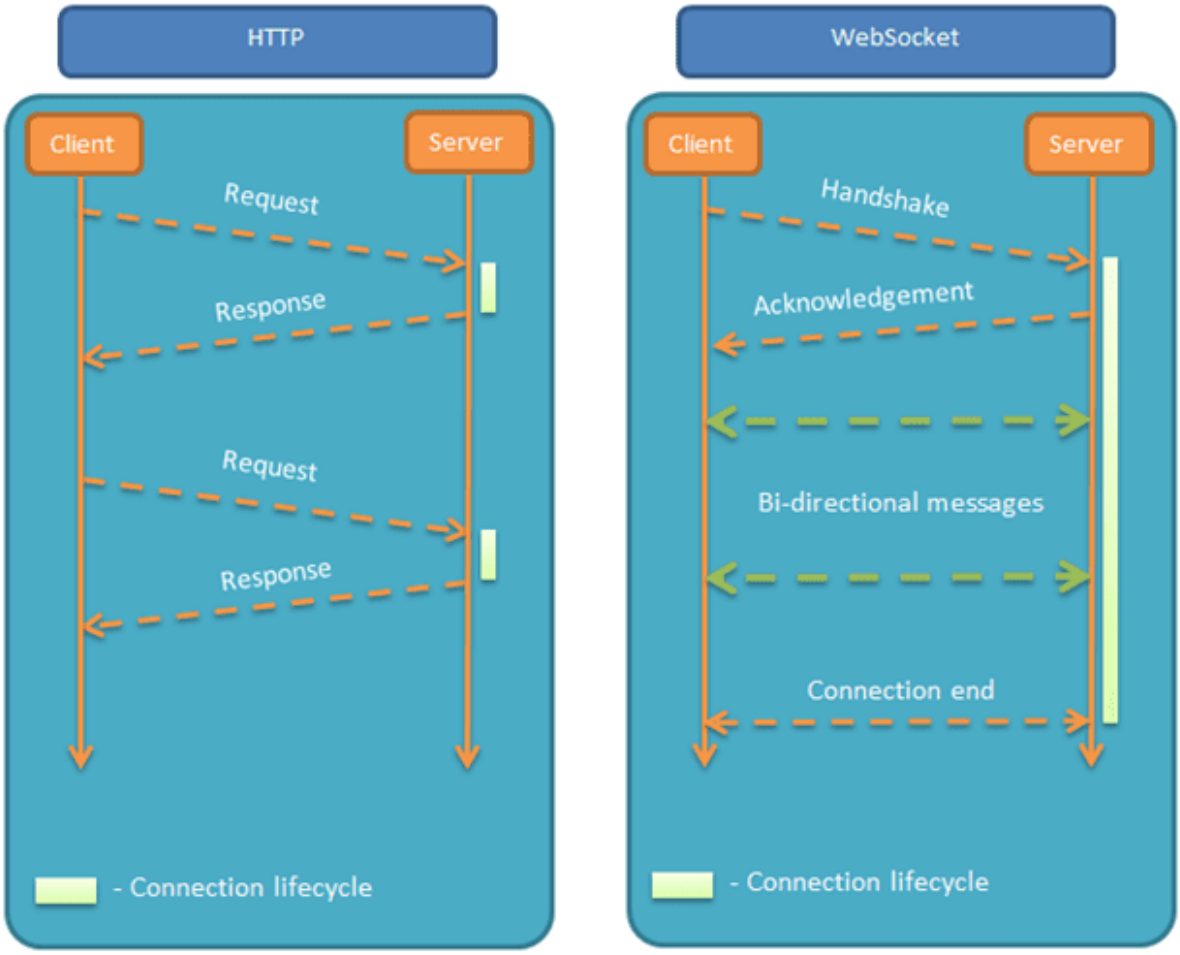
Nếu lưu lượng không được mã hóa đi qua một transparent proxy không hỗ trợ WebSocket, nhiều khả năng kết nối sẽ không thể thực hiện.

Mặt khác, với những lưu lượng WebSocket được mã hóa, thì việc sử dụng TLS giúp đảm bảo lệnh HTTP CONNECT được gửi khi trình duyệt được cấu hình để sử dụng proxy server. Việc này sẽ thiết lập một tunnel, cung cấp giao tiếp TCP low-level, end-to-end thông qua HTTP proxy, ở giữa WebSocket Secure client và WebSocket server.

Đối với transparent proxy, trình duyệt sẽ không biết về proxy server, nên HTTP CONNECT sẽ không được gửi đi. Tuy nhiên, vì traffic được mã hóa, nên các transparent proxy trung gian có thể cho phép lưu lượng được mã hóa đi qua. Do đó, nếu sử dụng WebSocket Secure thì khả năng thiết lập kết nối WebSocket sẽ cao hơn. Việc sử dụng mã hóa dù tốn chi phí tài nguyên, nhưng sẽ mang lại tỷ lệ thành công cao hơn vì nó đi qua tunnel an toàn hơn.

2.6 So sánh WebSocket và HTTP

Nếu vẫn đang phân vân giữa WebSocket và HTTP liệu có gì giống nhau và khác nhau nữa hay không, chúng ta sẽ tiếp tục tìm hiểu!



Hình 5. Sự khác nhau giữa Websocket và HTTP

WebSocket và HTTP giống nhau ở điểm nào?

Điểm giống nhau là cả WebSocket và HTTP đều là những giao thức hỗ trợ việc truyền tải thông tin giữa máy chủ và máy trạm.

Mục đích của các nhà phát triển khi tạo ra WebSocket là nhằm để khắc phục được nhược điểm độ trễ cao của HTTP.

WebSocket và HTTP khác nhau ở điểm nào?

Điểm khác biệt lớn nhất được liên tục được nhắc trong bài đó chính là WebSocket có tốc độ truyền tải nhanh và độ trễ thấp hơn nhiều so với HTTP.

Quá trình vận hành

Như bên trên ảnh, bạn sẽ thấy ngay được điểm khác biệt giữa WebSocket và HTTP chính là sự giao tiếp giữa máy chủ và máy khách.

HTTP là giao thức 1 chiều dựa theo giao thức TCP, bạn có thể tạo ra các kết nối dựa vào request HTTP, sau khi kết nối được thực hiện xong và được phản hồi lại, quá trình sẽ kết thúc và đóng lại.

Trong khi đó, **WebSocket** là một **giao thức truyền tải 2 chiều** giữa máy chủ và máy khách. Dữ liệu có thể truyền 2 chiều giữa máy khách – máy chủ hoặc máy chủ – máy khách dựa trên những kết nối đã được thiết lập.

Ứng dụng vào thực tiễn

Hầu hết các ứng dụng theo thời gian thực – real-time đều đang ứng dụng **WebSocket** để truyền và nhận dữ liệu trên 1 kênh liên lạc duy nhất.

Trong khi đó, **HTTP** được ứng dụng và sử dụng vào các dịch vụ ứng dụng RESTful đơn giản, chủ yếu là nhận thông tin 1 chiều về server để xử lý => phản hồi => đóng kết nối.

Ứng dụng dựa vào mức độ ưu tiên

Đối với các ứng dụng cần phải thường xuyên hoặc liên tục cập nhật thông tin, WebSocket sẽ là ưu tiên hàng đầu vì WebSocket có kết nối nhanh hơn, độ trễ thấp hơn so với HTTP.

Khi bạn muốn giữ lại một kết nối trong khoảng thời gian cụ thể hoặc bạn chỉ có nhu cầu sử dụng kết nối để truyền dữ liệu, HTTP sẽ chính là lựa chọn ưu tiên hơn dành cho bạn.

Nên sử dụng WebSocket trong trường hợp nào?

Có thể nói rằng, WebSocket là một công nghệ "xịn" có độ trễ rất thấp và phù hợp cho những dự án, những tác vụ cần phải nhanh ví dụ như:

- Ứng dụng theo thời gian thực
- Game Online
- Ứng dụng chat

Ứng dụng theo thời gian thực

Tất nhiên, ứng dụng được ưu tiên hàng đầu để sử dụng các công nghệ tân tiến, có mức độ cập nhật dữ liệu nhanh chóng từ máy trạm đến máy khách và ngược lại. Các ứng dụng thời gian thực sẽ sử dụng WebSocket để hiện thị ở máy khách một cách liên tục nhờ vào các máy chủ phu hỗ trợ. WebSocket sẽ liên tục truyền/ đẩy dữ liệu đi trong cùng 1 kết nối đã mở. Vì thế, tốc độ của ứng dụng được cải thiện hơn rất nhiều.

Một số ví dụ điển hình như: Sàn giao dịch Bitcoin, chứng khoán,… Những trang web, ứng dụng về các ngành này sẽ cần phải cập nhật thông với tốc độ tối đa nhất nhằm hỗ trợ người dùng có thể nhanh chóng bán ra hoặc mua vào.

Hình 6. Minh họa ứng dụng của Websocket trong xây dựng ứng dụng thời gian thực

Game Online

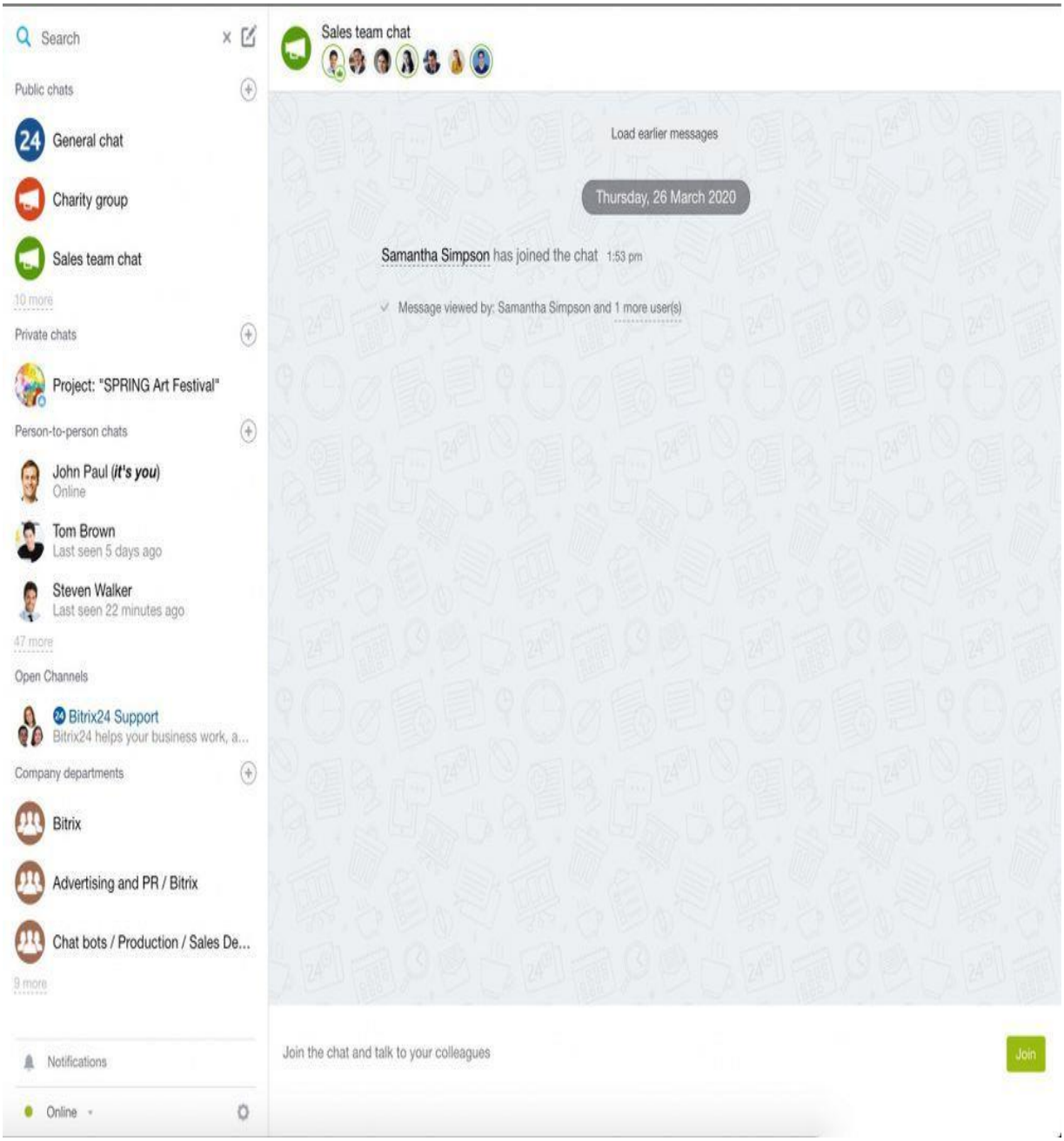
Trong các trò chơi điện tử, đặc biệt là các game online, việc kết xuất hình ảnh và cập nhật dữ liệu trong thời gian thực là một trong những yếu tố “sống còn”. Ví dụ, bạn đang chơi một trò chơi bắn súng sinh tồn, bạn thấy một khẩu súng “xịn” trước mắt, bạn

nhặt khẩu súng lên để hạ những người chơi khác. Nhưng độ trễ không được tối ưu sẽ khiến bạn bị người chơi khác bắn hạ khi bạn còn chưa kịp làm gì.

Ứng dụng chat

Với các ứng dụng chat, nhắn tin, độ trễ sẽ không cần thiết phải tối ưu hoá tốt nhất trong thời gian thực. Tuy nhiên, những ứng dụng nhắn tin, chat luôn có một độ trễ nhất định có thể chấp nhận được.

Nếu bạn đang phát triển các ứng dụng nhắn tin hay làm việc nội bộ như Bitrix24, bạn nên tham khảo công nghệ WebSocket để ứng dụng vào phần mềm của mình nhé!



Hình 7. Minh họa ứng dụng của Websocket trong xây dựng ứng dụng chat

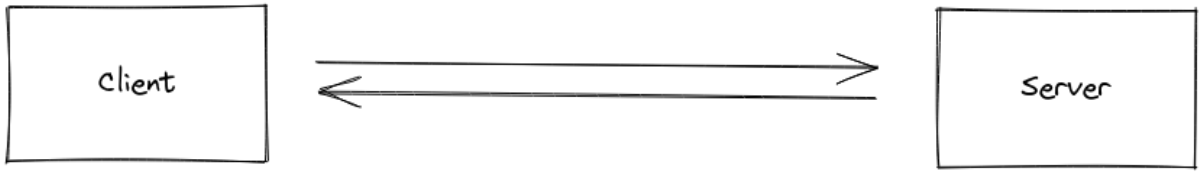
Trong trường hợp, bạn chỉ cần một công nghệ nhằm để chuyển dữ liệu 1 lần để xử lý hoặc truy vấn có tính đơn giản, HTTP sẽ phù hợp và tiết kiệm được nhiều tài nguyên nhân lực, thời gian phát triển của bạn hơn.

Đến đây, Tino Group đã giúp bạn tìm hiểu rất nhiều thông tin về WebSocket, giúp bạn hiểu được WebSocket là gì cũng như điểm khác biệt giữa WebSocket và HTTP. Mong rằng những thông tin này sẽ có thể giúp bạn xây dựng, phát triển ứng dụng của mình một cách tối ưu nhất!

3. Socket.IO:

3.1 Socket.IO là gì?:

Socket.io là một thư viện giúp real-time giúp chúng ta có thể giao tiếp 2 chiều giữa client và server. Nó được xây dựng dựa trên giao thức WebSocket và hỗ trợ thêm các tính năng như HTTP long-polling, tự động re-connect.



Hình 8. Minh họa ứng dụng của Socket.IO

3.2 Tính năng của Socket.IO:

3.2.1 Giao tiếp giữa Client và Server:

Socket.io sử dụng 2 khái niệm “emit” để phát sự kiện đi và “on” để lắng nghe sự kiện từ server.

Đặc biệt giao tiếp này là cả 2 chiều tức client vừa có thể phát và lắng nghe sự kiện từ phía server gửi về và ngược lại.

Ví dụ cho sử dụng Socket.io:

Server:

```
import { Server } from "socket.io";

const io = new Server(3000);

io.on("connection", (socket) => {
  // send a message to the client
  socket.emit("hello from server", 1, "2", { 3: Buffer.from([4]) });

  // receive a message from the client
  socket.on("hello from client", (...args) => {
    // ...
  });
});
```

Hình 9. Minh họa ứng dụng của Socket.IO

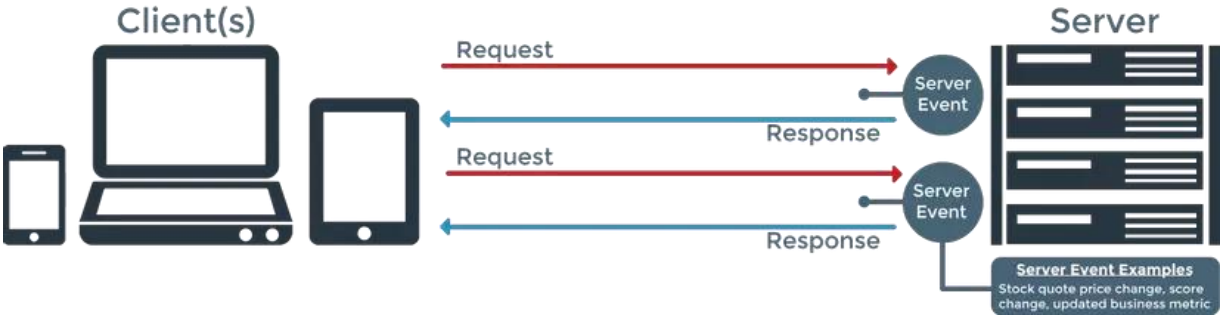
Client:

```
import { io } from "socket.io-client";

const socket = io("ws://localhost:3000");

// send a message to the server
socket.emit("hello from client", 5, "6", { 7: Uint8Array.from([8]) });

// receive a message from the server
socket.on("hello from server", (...args) => {
  // ...
});
```

Hình 10. Minh họa ứng dụng của Socket.IO

3.2.2 HTTP long-polling fallback:

Trong trường hợp Websocket không thể thiết lập kết nối giữa client và server thì kết nối sẽ trở lại sử dụng HTTP long-polling.



Hình 11. Minh họa hoạt động của Socket.IO

3.2.3 Automatic reconnection:

Trong một số điều kiện cụ thể, kết nối WebSocket giữa máy chủ và máy khách có thể bị gián đoạn mà cả hai bên đều không biết về trạng thái bị hỏng của liên kết.

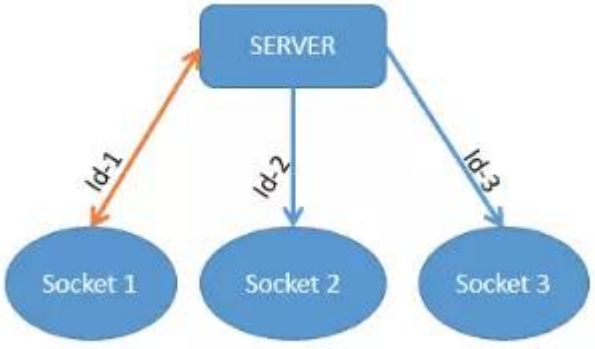
Socket.io sử dụng cơ chế "Heartbeat" (nhịp tim) để kiểm tra định kỳ trạng thái của kết nối. Trong trường hợp Client bị ngắt kết nối thì hệ thống sẽ tự động kết nối lại với độ trễ dự phòng theo cấp số nhân, tránh làm quá tải Server.

3.2.4 Packet buffering:

Các gói tin, dữ liệu mà Client gửi đi sẽ được lưu vào bộ nhớ đệm trong trường hợp chưa kết nối hay đứt kết nối đến Server. Và các gói tin đó sẽ được gửi lại khi kết nối lại thành công.

3.2.5 Broadcasting:

Server có thể gửi gói tin đến toàn bộ các client đã kết nối đến server hoặc một nhóm client kết nối trong subnet.



Hình 12. Minh họa ứng dụng của Socket.IO

Ví dụ:

```
// to all connected clients
io.emit("hello");

// to all connected clients in the "news" room
io.to("news").emit("hello");
```

Hình 13. Minh họa ứng dụng của Socket.IO

3.2.6 Multiplexing:

Tính năng cho phép phân chia các kênh "giao tiếp" riêng biệt cho một kết nối được chia sẻ.

Ví dụ:

```
io.on("connection", (socket) => {
  // classic users
});

io.of("/admin").on("connection", (socket) => {
  // admin users
});
```

Hình 14. Minh họa ứng dụng của Socket.IO

CHƯƠNG 2 – PHẦN RIÊNG

1. 51900718 – Tăng Kiến Trung:

1.1 Front-end, Back-end là gì? Ví dụ:

1.1.1 Front-end:

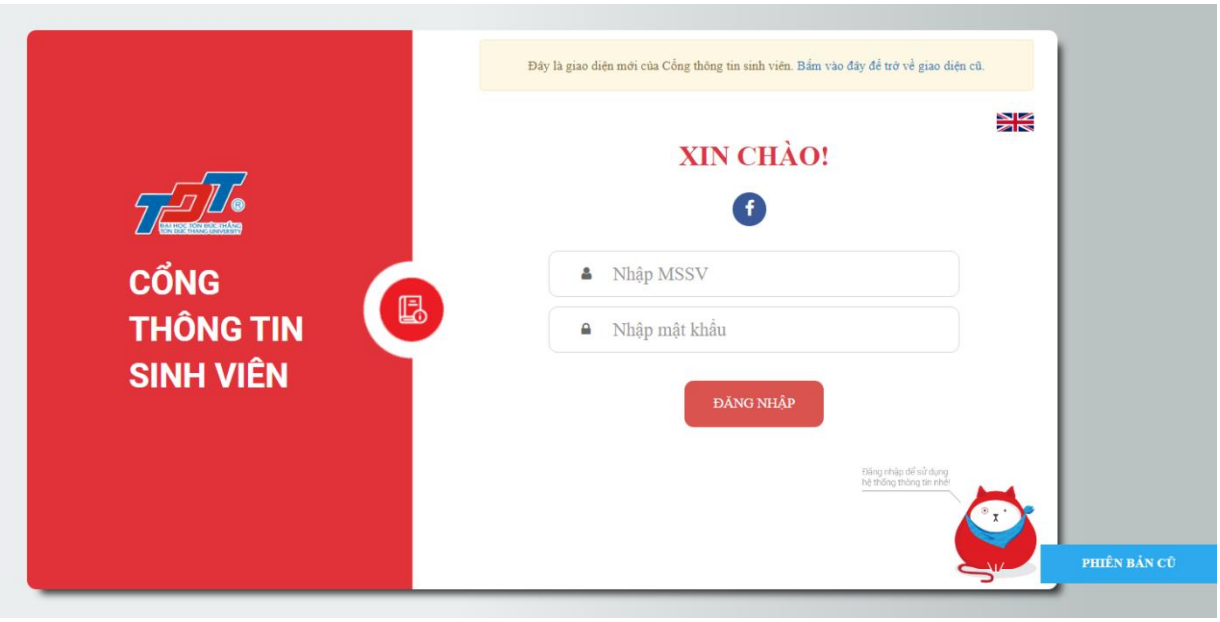
Front End (còn được biết đến như client-side) là tất cả những gì liên quan đến điều mà người dùng nhìn thấy mỗi khi truy cập vào một trang web, một ứng dụng web, hoặc ứng dụng mobile.

Front End trong lập trình web là giao diện của một trang web, là phần người dùng tương tác trực tiếp trên website. Nó bao gồm tất cả những thứ chúng ta có thể trải nghiệm trên một ứng dụng, website như: Màu sắc, kiểu văn bản, hình ảnh, đồ thị và bảng, menu điều hướng, font chữ, cho tới các menu xổ xuống và các thanh trượt, là một sự kết hợp của HTML, CSS, và JavaScript được đảm nhiệm bởi trình duyệt máy tính.…

Người dùng tương tác trực tiếp với nhiều khía cạnh thuộc front-end như: nhận biết vị trí đặt để của logo, màu sắc chủ đạo, tìm kiếm và đọc thông tin, sử dụng các button và tính năng trên web… Mục đích cuối cùng của Front End là nhằm mang lại một giao diện bắt mắt, giúp người dùng dễ dàng thao tác và sử dụng các chức năng của ứng dụng. Tính responsive và performance là hai yếu tố quan trọng nhất trong lập trình front-end. Lập trình viên front-end cần phải đảm bảo website của mình có khả năng responsive với nhiều loại kích thước màn hình khác nhau để người dùng có được trải nghiệm tốt nhất.
Ví dụ:

Tất cả mọi thứ chúng ta đang nhìn thấy trên trang web, ứng dụng mobile… mỗi khi truy cập đều có thể được thực hiện bởi một lập trình viên front-end và đó là những gì thuộc về front-end. Một designer sẽ có trách nhiệm thiết kế và tạo ra logo, đồ họa, một photographer sẽ cung cấp hình ảnh, một copywriter sẽ viết nội dung. Và cuối cùng lập trình viên front-end sẽ tổng hợp những yếu tố đó lại và tạo nên một giao diện front-end cho người dùng trải nghiệm.

Lấy ví dụ khi chúng ta truy cập vào hệ thống thông tin sinh viên của trường ĐH Tôn Đức Thắng, ta sẽ thấy được giao diện như sau:



Những gì ta thấy được trên màn hình thì đó là front-end.

1.1.2 Back-end:

Back-end (còn được biết đến là server-side của một website). Back-end có nhiệm vụ lưu trữ, quản lý, sắp xếp dữ liệu, ngoài ra nó còn có trách nhiệm đảm bảo các chức năng ở client-side có thể hoạt động ổn định. Trái ngược với front-end chúng ta có thể nhìn bằng mắt thường mỗi khi truy cập vào một ứng dụng web hay mobile, thì back-end sẽ là phần mà chúng ta không thể nhìn thấy hay tương tác với nó và back-end sẽ luôn chạy nền để cung cấp các chức và đảm bảo cho front-end chạy ổn định. Back-end là phần mà không được tương tác trực tiếp bởi người dùng mà được tương tác gián tiếp thông qua front-end (ví dụ: submit form, nhấn button yêu cầu lấy dữ liệu từ cơ sở dữ liệu…).

Phần back end của một trang web bao gồm một máy chủ, một ứng dụng, và một cơ sở dữ liệu. Một lập trình viên back-end xây dựng và duy trì công nghệ mà sức mạnh của những thành phần đó, cho phép phần giao diện người dùng của trang web có thể tồn

tại được. Có thể cho rằng Back-end giống như bộ não của con người. Nó xử lý những yêu cầu, câu lệnh và lựa chọn thông tin chính xác để hiển thị lên màn hình.

Back End của một trang web bao gồm: Máy chủ, ứng dụng và cơ sở dữ liệu. Để kết nối các phần này với nhau, các Developer sẽ sử dụng ngôn ngữ lập trình như Ruby, Python, PHP, Java… và công cụ như Oracle, SQL Server, MySQL…

Ví dụ:

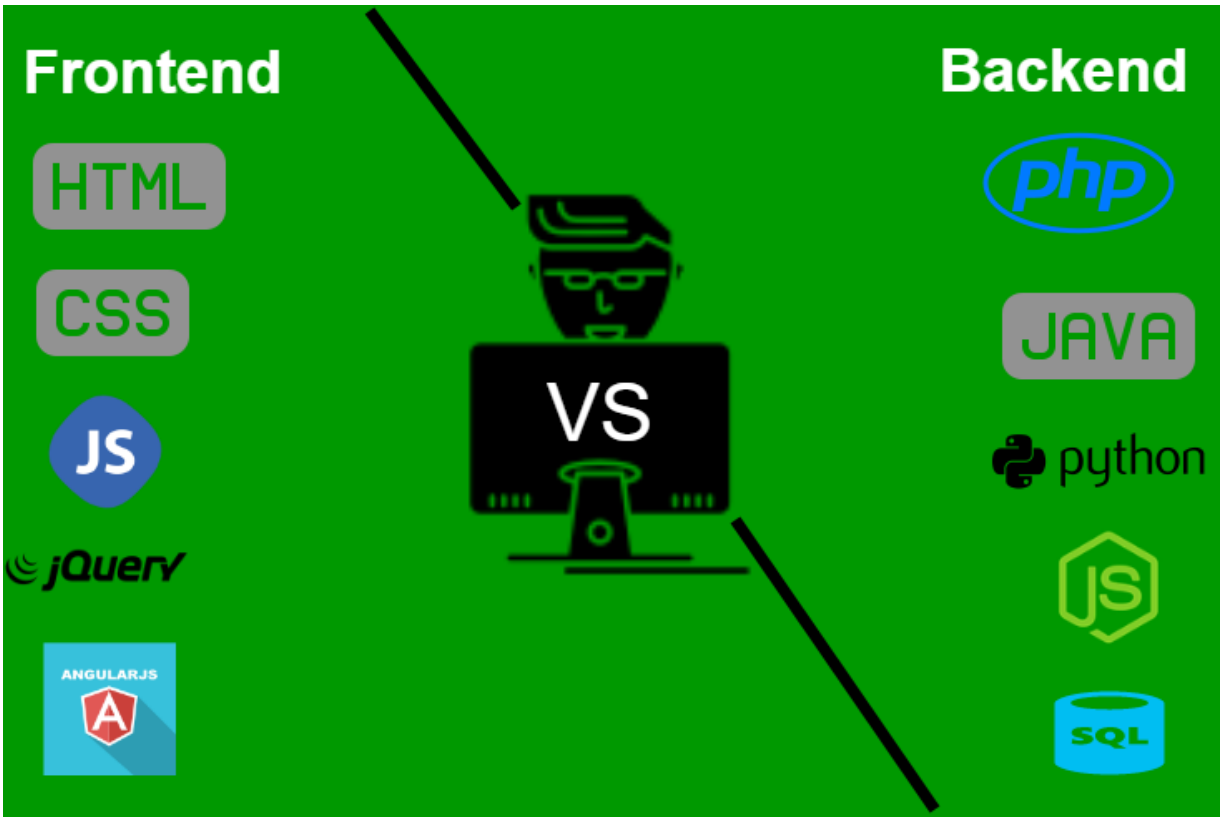
Khi chúng ta điều hướng tới trang thông tin sinh viên của trường ĐH Tôn Đức Thắng, thì máy chú phía back-end sẽ có nhiệm vụ trả về các tài nguyên như HTML, CSS, JS để trình duyệt render những tài nguyên này thành trang web mà người dùng có thể hiểu và tương tác được.

Tiếp theo khi chúng ta nhập tên tài khoản và mật khẩu rồi nhấn "ĐĂNG NHẬP" thì những dữ liệu như tên tài khoản, mật khẩu sẽ được gửi đến server của nhà trường. Sau đó những dữ liệu này sẽ được kiểm tra và đối chiếu với tài khoản đã được lưu trong cơ sở dữ liệu của trường. Nếu khớp ta sẽ được cấp quyền và truy cập thành công. Ngược lại thì server sẽ trả về thông báo lỗi và front-end sẽ hiển thị thông báo đó lên giao diện để người dùng đọc.

So sánh Back-end và Front-end

Back-end và Front-end hoạt động song song với nhau để đảm bảo một ứng dụng hoặc website hoạt động bình thường. Sự khác biệt giữa Front-end và Back-end liên quan đến người dùng. Trong khi Front-end là những gì người dùng nhìn thấy được bằng mắt thường, Back-end là thứ giúp các chức năng được xây dựng ở phần Front-end có thể sử dụng được.

Điểm khác biệt giữa dễ thấy nhất Front-end và Back-end là những gì mà người dùng nhìn thấy được bằng mắt thường và không nhìn thấy được của một website, một ứng dụng mobile...



Ngôn ngữ Front End thông dụng:

- HTML: HTML là viết tắt của Hypertext Markup Language. Nó là ngôn ngữ đánh dấu được sử dụng để thiết kế phần giao diện người dùng.
- CSS: Là ngôn ngữ đi kèm với HTML, quyết định các yếu tố về bố cục, màu sắc, phông chữ của một website.
- JavaScript: Được sử dụng để cải thiện và nâng cao chức năng của một trang web.

Ngôn ngữ lập trình Back End thông dụng:

- Java: Là ngôn ngữ lập trình được sử dụng phổ biến nhất cho các trang web và ứng dụng như Netflix, Tinder, Google Earth và Uber.
- Ruby on Rails (RoR): Đây là ngôn ngữ được ưa chuộng bởi các Developer, nó giúp việc lập trình Back-end trở nên dễ dàng hơn.

- Python: Python là một trong những ngôn ngữ lập trình được sử dụng phổ biến nhất trên thế giới. Một số trang web và ứng dụng sử dụng ngôn ngữ Python: Spotify, Google, Instagram, Reddit, Dropbox.
- PHP: Ngôn ngữ lập trình này khá dễ học. Đây cũng là ngôn ngữ lập trình được sử dụng cho các website như Facebook, Wikipedia, Tumblr, MailChimp và Flickr.

1.2 Node.js để xây dựng Front-end hay Back-end? Vì sao?

Trước khi trả lời câu hỏi Node.js để xây dựng Front-end hay Back-end thì ta hãy tìm hiểu Node.js là gì?

Node.js là gì?

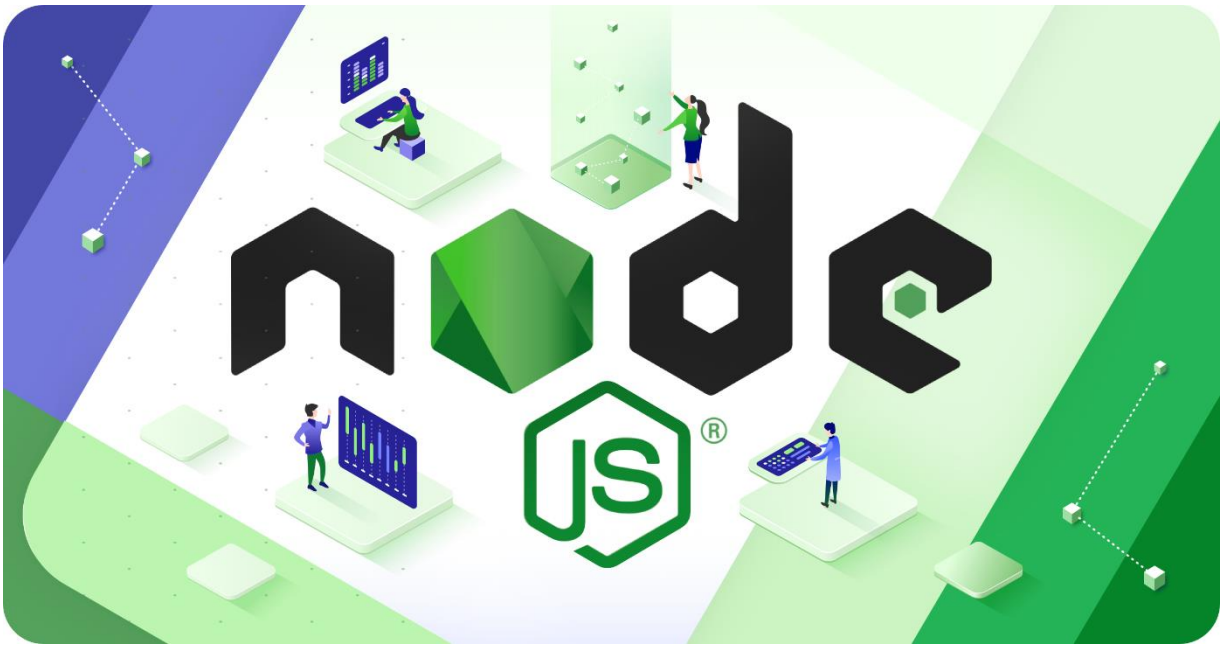
Đây là định nghĩa chính thức trên trang web chính của Node.js:

"Node.js là một JavaScript runtime được build dựa trên engine JavaScript V8 của Chrome. Node.js sử dụng kiến trúc hướng sự kiện event-driven, mô hình non-blocking I/O làm cho nó nhẹ và hiệu quả hơn. Hệ thống nén của Node.js, npm, là hệ thống thư viện nguồn mở lớn nhất thế giới."

Node.js là một nền tảng (Platform) mã nguồn mở chạy trên môi trường V8 JavaScript runtime (một trình thông dịch JavaScript chạy cực nhanh trên trình duyệt Chrome). Node.js giúp các nhà phát triển xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng.

Node.js 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,..

Cả trình duyệt JavaScript và Node.js đều chạy trên JavaScript runtime V8 engine. Công cụ này lấy code JavaScript của chúng ta và convert nó sang mã máy (bytecode) cho việc thực thi nhanh hơn. Mã máy là loại code thấp cấp hơn để máy tính có thể chạy mà không cần biên dịch nó.



Các tính năng chính của Node.js

- Single-Threaded: tuân theo mô hình một luồng với vòng lặp sự kiện
- Khả năng mở rộng: Máy chủ phản hồi theo cách non-blocking (không chặn) nên có khả năng mở rộng cực cao.
- Cộng đồng: hỗ trợ các bộ mã nguồn mở. Có rất nhiều mô-đun hữu ích được tích hợp vào ứng dụng Node.js theo thời gian.
- Mã nguồn mở: là một nền tảng thời gian chạy mã nguồn mở và đa nền tảng được sử dụng để tạo tất cả các loại ứng dụng bằng JavaScript.
- Zero Buffering: Ứng dụng không lưu dữ liệu đệm trong Node.js vì chúng chỉ xuất ra các số liệu thống kê trong các phần lớn.

Tại sao nên sử dụng Node.js?

- Do sử dụng các mô hình I/O *hướng sự kiện* và non-blocking (không chặn) nên nó có trọng lượng nhẹ và hiệu quả.
- Thời gian chạy JavaScript được xây dựng trên công cụ JavaScript V8 của Chrome. Node.js có tốc độ cao vì nó là một công cụ mã nguồn mở lấy mã JS và biên dịch nhanh chóng.

Chúng ta thường hiểu lầm rằng Node.js chỉ có thể sử dụng để xây dựng back-end nhưng câu trả lời là **Node.js có thể được dùng để xây dựng cho cả Front-end lẫn Back-end.**

Lý do là vì:

Node.js là một môi trường biên dịch dựa trên *engine JavaScript V8 của Chrome, điều này cho phép chúng ta lựa chọn giữa việc sử dụng Javasciprt như thế nào? Khi nào sử dụng, sử dụng ở front-end hay back-end.*

Mặt khác, nó cũng có các tính năng để tự động hóa các tác vụ thông thường bao gồm thực hiện kiểm tra mã, bộ công cụ, v.v. cho nên có thể sử dụng nó như một môi trường front-end. Dưới đây là một số ứng dụng của Node.js trong việc xây dựng front-end lẫn back-end.

Năng suất và hiệu quả: Nhờ việc giảm chuyển đổi ngữ cảnh giữa nhiều ngôn ngữ khác nhau, chúng ta có thể tiết kiệm được rất nhiều thời gian. Sử dụng JavaScript cho cả phần back-end và front-end giúp tăng hiệu quả, vì có nhiều công cụ chung cho cả hai. Việc sử dụng JavaScript cho cả hai phần cũng khiến cho công việc của developer trở nên dễ dàng hơn do chỉ cần tập trung vào một ngôn ngữ.

Front-end:

Code Processors: Trong các tập tin HTML, CSS, JS chúng ta nhận được mỗi khi request đến server, thường sẽ chứa rất nhiều ký tự khoảng trắng dư thừa, cũng như tên hàm, tên biến đặt theo quy ước để lập trình viên dễ hiểu. Tuy nhiên thì với trình duyệt những ký tự hay quy ước trên là dư thừa và làm tăng kích thước của những tập tin này. Do đó những tập tin HTML, CSS, JS thường sẽ được loại bỏ hết các ký tự thừa để tập trung vào render và truyền tải tập tin nhanh hơn do kích thước đã được giảm đi nhiều.

Code Linters: Linter là một chương trình giúp chúng ta xác định và sửa các lỗi trong code của mình. Các lỗi liên quan đến cú pháp, một vài lỗi tự định nghĩa tiêu chuẩn được định ra bởi nhóm phát triển và các lỗi lập trình có thể được xác định bằng cách sử

dụng liners. Custom một linters có thể nâng cao mức độ hiệu quả của toàn nhóm lập trình viên trong một công ty.

Module Bundlers: Module bundlers là chương trình gôm các file code lại và gộp chúng lại thành một file duy nhất. Các chương trình này thường được sử dụng để đính kém và Web Frameworks như là React. Và trong Node.js cũng có một module bundlers tên là Webpack, module này được xây dựng ở tầng trên cùng của Node.js.

Styling: Việc thiết kế, trang trí một trang web thường sẽ được thực hiện bằng CSS. Tuy nhiên có một số packages như là "styled-components" đã làm cho việc thiết kế trở nên rất dễ dàng. Styled-components là một thư viện được viết cho ReactJS. Thư viện này cho phép chúng ta dễ dàng thiết kế, trang trí bằng ngôn ngữ JS điều này giúp tăng tính hiệu quả hơn.

Packages: npm cung cấp rất nhiều packages giúp cho quá trình lập trình, phát triển trở nên dễ dàng hơn bao giờ hết. Ví dụ: chúng ta cần truy cập vào các thành phần như "trình chỉnh sửa văn bản", "trình chọn màu", "thành phần xác thực".. chúng đều có thể được cài đặt từ npm. Việc xây dựng front-end đã trở nên dễ dàng, lúc này công việc của chúng ta chỉ là kết hợp các thành phần tách biệt nhau trở thành một giao diện UI đẹp đẽ.

Back-end:

Bởi vì Node.js sử dụng cơ chế non-blocking, cho nên đã có thể giảm được đáng kể số lượng threads cần thiết để sử dụng các yêu cầu UI. Điều này đã làm cho Node.js trở thành một môi trường gọn – nhẹ. Từ đó mà một số ứng dụng có thể được phát triển nhờ ưu điểm này:

Network và API : Node.js có rất nhiều thư viện đa dạng cho phép chúng ta xây dựng API lẫn gọi API. Chúng ta cần phải hiểu rõ một request HTTP hoạt động ra sao, chứa những gì… Node.js giúp cho chúng ta dễ dàng quản lý các request HTTP. Ta có thể chỉnh sửa header, cài đặt proxies, và nhận kết quả trả về phù hợp với loại ứng dụng của chúng ta.

Database Integration: Node.js có nhiều thư viện và interfaces cho phép chúng ta tương tác và làm việc với nhiều hệ cơ sở dữ liệu khác nhau. Từ cơ sở dữ liệu quan hệ SQL, MySQL cho đến cơ sở dữ liệu không quan hệ như MongoDB… Các thao tác CRUD (Thêm, Xóa, Sửa, Xem) có thể được thực thi thông qua code Javascript.

Operating System-Level Control over Application: Node.js còn có cung cấp một số tính năng giúp ta có thể tương tác hoặc lập trình ở mức độ Hệ điều hành.

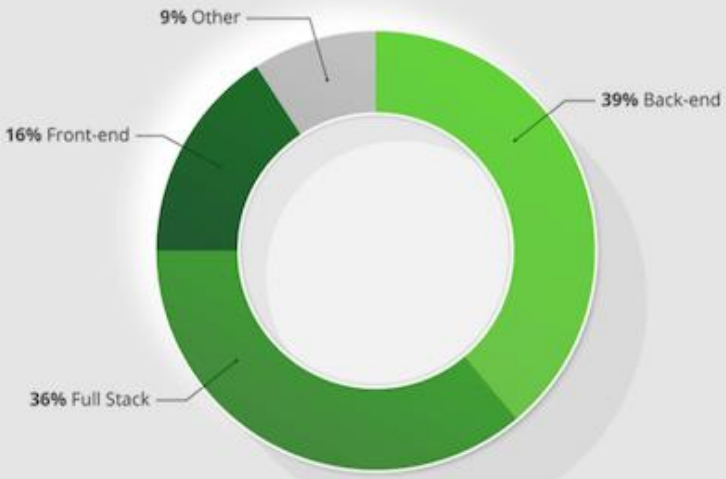
Ứng dụng thời gian thực: Các ứng dụng như live stream, chat trực tuyến, chat video, Iot…

Dưới đây là thống kê về cách mà Node.js được sử dụng trong các tổ chức, công ty. Chúng ta có thể thấy Node.js đã được sử dụng trong cả front-end lẫn back-end. Qua thống kê trên cũng đã góp phần chứng minh cho câu trả lời Node.js có thể được sử dụng trong việc xây dựng cả front-end lẫn back-end.

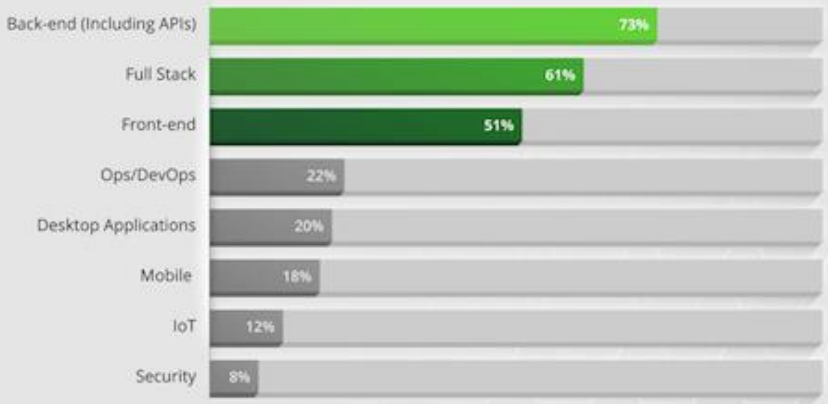
USAGE APPLICATION

How organizations use Node.js

DEVELOPMENT PRIMARY FOCUS



DEVELOPMENT NET FOCUS



Kết luận:

Chúng ta có thể sử dụng Node.js để xây dựng cả front-end lẫn back-end. Node.js cung cấp cho ta những tính năng giúp cá nhân hóa, linh động, và một cộng đồng người dùng lẫn thư viện vô cùng lớn và đa dạng. Giúp cho việc phát triển một ứng dụng full-stack trở nên dễ dàng.

1.3 Express.js để xây dựng Front-end hay Back-end? Vì sao?

Em xin trả lời rằng Express.js là để xây dựng Back-end. Để lý giải vì sao thì chúng ta hãy cùng đi qua các khái niệm, cũng như cơ sở lý thuyết về Express.js.

1.3.1 Express.js là gì?

Định nghĩa Express.js:

Express.js là một framework ứng dụng web có mã nguồn mở và miễn phí được xây dựng trên nền tảng Node.js. Express.js được sử dụng để thiết kế và phát triển các ứng dụng web một cách nhanh chóng. Để hiểu Express.js, người dùng chỉ cần phải biết JavaScript, do đó nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn đối với các lập trình viên và nhà phát triển đã thành thạo JavaScript trước đó.

Vì Express.js là một framework của Node.js nên hầu hết các mã đã được viết sẵn cho các lập trình viên làm việc. Chúng ta có thể tạo các ứng dụng web cho một trang, nhiều trang hoặc kết hợp lại bằng cách sử dụng Express.js. Framework này khá nhẹ, giúp tổ chức các ứng dụng web ở phía máy chủ thành một kiến trúc MVC hoàn hảo hơn.

Express.js được xây dựng sẵn giúp lập trình viên tạo ứng dụng web phía máy chủ. Đó là một cách tiếp cận cho công việc của lập trình viên trở nên nhanh hơn và dễ dàng hơn. Các đặc điểm của Express.js là tính linh hoạt, tính tối giản và khả năng mở rộng khiến nó trở nên lý tưởng hơn so với các lựa chọn khác.

Express.js hỗ trợ nâng cao các chức năng của Node.js. Nếu bạn không sử dụng Express.js, bạn phải thực hiện rất nhiều lập trình phức tạp để xây dựng một API hiệu quả. Express.js đã giúp cho việc lập trình trong Node.js trở nên dễ dàng hơn rất nhiều.

Express js là một Framework nhỏ, nhưng linh hoạt được xây dựng trên nền tảng của Node.js. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile.

Vì Express js chỉ yêu cầu ngôn ngữ lập trình Javascript nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn với các lập trình viên và nhà phát triển. Express.js cũng là một khuôn khổ của Node.js do đó hầu hết các mã code đã được viết sẵn cho các lập trình viên có thể làm việc.

Trên thực tế, nếu không sử dụng Express.js, bạn sẽ phải thực hiện rất nhiều bước lập trình phức tạp để xây dựng nên một API hoặc một ứng dụng web hiệu quả. Express js đã giúp cho việc lập trình trong Node.js trở nên dễ dàng hơn và có nhiều tính năng mới bổ sung.

Express.js được sử dụng để làm gì?

Express.js sẽ giúp chúng ta tổ chức kiến trúc back-end của mình. Các nhà phát triển web thường sử dụng Express.js để triển khai kiến trúc MVC, điều này cho phép họ viết một codebase back-end bảo trì tương đối dễ dàng.

Bởi vì Express.js hoạt động trên back-end, nên bạn có thể coi công nghệ này như một "bộ não đằng sau một trang web". Ví dụ: Express.js có thể xác định cách những

trang được định tuyến trên một trang web. Hơn nữa, một nhà phát triển có thể sử dụng Express.js để quản lý xác thực trên một trang web.

1.3.2 Các tính năng và lợi ích nổi bật của Express.js:

Các tính năng của Express.js:

Phát triển máy chủ nhanh hơn

Express.js cung cấp cho bạn nhiều tính năng phổ biến của Node.js dưới dạng hàm có thể dễ dàng sử dụng ở bất kỳ đâu trong chương trình. Điều này sẽ giúp rút ngắn thời gian để viết code.

Phần mềm trung gian - Middleware

Phần mềm trung gian là một phần trong chương trình cho phép truy cập vào cơ sở dữ liệu, xem xét yêu cầu của khách hàng và các phần mềm trung gian khác. Tính năng này chịu trách nhiệm chính cho việc tổ chức chức năng khác nhau của Express.js.

Định tuyến - Routing

Express.js cung cấp một cơ chế định tuyến nâng cao giúp duy trì trạng thái của trang web với sự trợ giúp của URL và các HTTP Method.

Tạo mẫu – Templating

Express.js cung cấp các công cụ tạo khuôn mẫu cho phép các nhà phát triển tạo nội dung động trên các trang web bằng việc xây dựng các mẫu HTML ở phía máy chủ.

Nội dung động: Sử dụng các mẫu HTML, công cụ tạo khuôn mẫu của Express.js đang cung cấp nội dung động trên trang web.

Gỡ lỗi - Debugging

Gỡ lỗi là yếu tố quan trọng để phát triển các ứng dụng web. Express.js giúp gỡ lỗi dễ dàng hơn bằng cách cung cấp một cơ chế có khả năng xác định chính xác phần ứng dụng web có lỗi.

Những lợi ích của Express.js

- Rất dễ học, chỉ cần bạn biết JavaScript, bạn sẽ không cần phải học một ngôn ngữ mới để học Express.js
- Giúp cho việc phát triển back-end dễ dàng hơn nhiều khi sử dụng Express.js
- Mã JavaScript được diễn giải thông qua Google V8 JavaScript Engine của Node.js. Do đó, mã sẽ được thực hiện một cách nhanh chóng và dễ dàng.
- Express.js rất đơn giản để tùy chỉnh và sử dụng theo nhu cầu.
- Cung cấp một module phần mềm trung gian linh hoạt và rất hữu ích để thực hiện các tác vụ bổ sung theo phản hồi và yêu cầu.
- Hỗ trợ phát triển ứng dụng theo mô hình MVC, đây là mô hình phổ biến cho việc lập trình web hiện nay.

Kết luận:

Express.js dùng để xây dựng Back-end.

1.4 Express.js là thư viện mã nguồn mở hay Framework hay là ngôn ngữ lập trình web hay công cụ xây dựng web? Vì sao?

Như những thông tin mà ta đã biết về Express.js đã được đưa ra ở phần 1.3 thì Express.js là một Framework của NodeJS. Nó không phải là thư viện mã nguồn mở. Lại càng không phải là ngôn ngữ lập trình Web mà nó chỉ là một Framework sử dụng ngôn ngữ Javascript để lập trình. Và đây cũng không phải là công cụ xây dựng web. Để bổ sung cho câu trả lời thì chúng ta hãy cùng đi qua khái niệm về thư viện, framework là gì?

1.4.1 Library là gì?

Là một tập hợp các chức năng (functions), các lớp (class) được viết sẵn để có thể tái sử dụng. Mỗi function hoặc class phục vụ cho một công việc cụ thể nào đó.

Ví dụ:

- JQuery là một library, nó cung cấp các chức năng giúp chúng ta thao tác với DOM.
- LinqJS là một library, nó cung cấp các chức năng giúp chúng ta truy vấn (query) dữ liệu dễ dàng, đơn giản và nhanh hơn.

1.4.2 Framework là gì?

Là một tập hợp các Library đã được đóng gói để hỗ trợ phát triển ứng dụng dựa trên framework đó.

Đồng thời, Framework cung cấp các nguyên tắc, cấu trúc của ứng dụng mà chúng ta phải tuân thủ theo nó.

Ví dụ 1: Angular là một framework. Mục đích Angular framework là giúp cho người dùng xây dựng được các ứng dụng website dạng single page một cách dễ dàng và nhanh chóng. Nó tập trung vào việc phát triển font-end cho ứng dụng web. Angular cung cấp sẵn cho bạn các directives, services, data-biding, filters,… Để sử dụng Angular, chúng ta phải tuân thủ theo mô hình và cách hoạt động của nó. Chẳng hạn, một page sẽ có phần html gọi là template, phần xử lý gọi là controller, các quy định về việc sử dụng $scope, isolate-scope, cách để trao đổi dữ liệu giữa các page như thế nào. Nghĩa là Angular team đã viết sẵn các thư viện (Libraries) để bạn sử dụng lại, cùng với một khuôn mẫu (design parttern) mà bạn phải tuân theo nó để có thể xây dựng được ứng dụng.

1.4.3 Những điểm khác nhau giữa Framework và Library là gì?

- Framework và Library đều cung cấp các tính năng (functions) được viết sẵn để chúng ta có thể tái sử dụng.
- Framework lớn hơn và phức tạp hơn Library.
- Sử dụng Framework bạn phải thay đổi cấu trúc code của dự án (project’s structure) theo các quy tắc của framework đó để có thể sử dụng được các functions mà framework đó cung cấp.

- Chúng ta có thể sử dụng các functions của Library một cách trực tiếp mà không cần thay đổi cấu trúc code của dự án.
- Framework có thể hiểu là một khung chương trình, người dùng bổ sung code và tuân theo quy tắc để tạo ra ứng dụng. Còn Library chỉ cung cấp các chức năng tiện ích hay các class để sử dụng trong quá trình xây dựng ứng dụng.
- Framework hoạt động chủ động. Nghĩa là nó có thể đưa ra các quyết định gọi hoặc bị gọi bởi các Library hay ứng dụng nào đó.
- Library hoạt động bị động. Nghĩa là nó chỉ được gọi khi nào chúng ta cần dùng nó.

Phân biệt Express.js và Node.js

Để phân biệt, trước hết chúng ta đi đến yếu tố phân biệt cơ bản nhất dưới đây.

Express.js là một framework dựa trên Node.js để triển khai ứng dụng web với việc sử dụng các nguyên tắc của Node.js. Nó giúp phát triển web dễ dàng

Node.js là một nền tảng sử dụng JavaScript để xây dựng các ứng dụng I/O hướng sự kiện phía máy chủ.

Bảng so sánh Express.js và Node.js

Express.js	Node.js
Là framework của Node.js, sử dụng để xây dựng phần back-end của ứng dụng web.	Được sử dụng để xây dựng cả front-end và back-end của ứng dụng web.
Được viết bằng một ngôn ngữ lập trình duy nhất là Javascript.,...	Được viết bằng nhiều ngôn ngữ lập trình khác nhau như C/C++,
Là một framework	Không phải là framework

Để sử dụng Express.js các lập trình viên cần cài đặt Express.js cùng Node.js.	Các lập trình viên chỉ cần cài đặt Node.js trên thiết bị của mình để có thể sử dụng
Được sử dụng để xây dựng nên các ứng dụng phía máy chủ trên Node.js.	Được sử dụng để phát triển lên các ứng dụng mạng và phía máy chủ.
Express.js phù hợp với các dự án có quy mô nhỏ.	Node.js được sử dụng cho các dự án có quy mô lớn.
Express.js chỉ được sử dụng ở phía máy chủ.	Node.js có thể sử dụng được cả phía máy chủ và máy khách.
Express.js tương thích với tất cả các hệ điều hành tương thích với Node.js.	Node.js tương thích với tất cả các hệ điều hành chính.
Cung cấp các thành phần định tuyến và có phần mềm trung gian để hỗ trợ giúp phát triển ứng dụng web một cách dễ dàng hơn.	Cung cấp nhiều tính năng cho các nhà phát triển trong xây dựng một ứng dụng web.
Express.js chỉ hỗ trợ ngôn ngữ JavaScript.	Node.js hỗ trợ nhiều ngôn ngữ khác nhau như: TypeScript, CoffeeScript và Ruby.
Nó được sử dụng bởi IBM, PayPal, Fox Sports,…	Nó được sử dụng bởi LinkedIn, PayPal, Walmart, Uber,…

Kết luận:

Express.js là một Framework không phải thư viện cũng không phải ngôn ngữ lập trình và công cụ xây dựng web.

2. 51900715 – Đặng Đăng Trí:

2.1.1 Front-end:

Front-end là tất cả những gì mà người dung có thể nhìn thấy và tương tác trực tiếp như giao diện. Tất cả mọi thứ bạn nhìn thấy khi điều hướng trên Internet, từ các font chữ, màu sắc cho tới các menu xổ xuống và các thanh trượt đều là Front-end. Front-end

thường sẽ làm nhiệm vụ chính là tương tác với người dung và gửi các yêu cầu tương ứng dựa trên hành vi của người dung xuống cho Back-end để tiến hành xử lý yêu cầu.

Front-end được cấu thành chính từ 3 ngôn ngữ là HTML, CSS, JavaScript. Hiện nay, có rất nhiều framework để hỗ trợ cho việc thiết kế Front-end như Angular, React, …

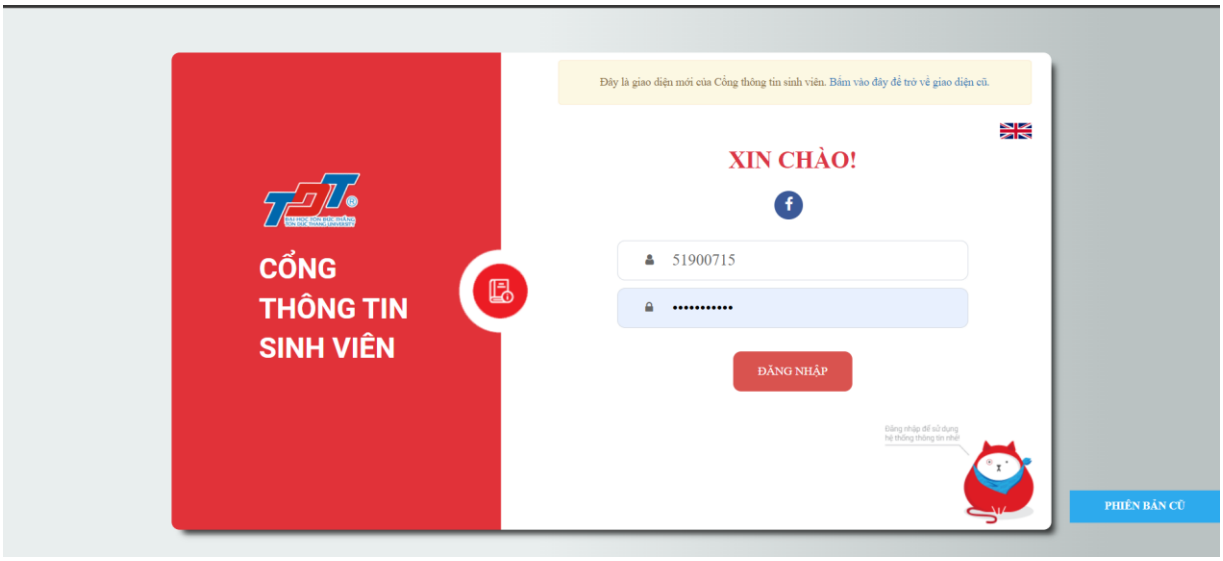
2.1.2 Back-end:

Back-end, trái ngược với Front-end, Back-end là tất cả những phần hỗ trợ cho hoạt động của ứng dụng mà người dung không thể nhìn thấy được. Back-end xử lý thông tin dựa trên những yêu cầu từ phía người dùng và đưa ra kết quả tương ứng với yêu cầu. Các yêu cầu này có thể đến từ Front-end hoặc cũng có thể đến từ các công cụ lập trình. Kết quả trả về của Back-end thường chỉ là thông tin cần tìm hoặc kết quả thành công hay thất bại của một yêu cầu.

Back-end được cấu thành chính từ 3 thành phần: máy chủ, ứng dụng, cơ sở dữ liệu. Hiện nay có rất nhiều loại cơ sở dữ liệu khác nhau nhằm đáp ứng các nhu cầu đa dạng của người dùng: mongoDB, Postgres, MySql, SqlServer, H2, … Và cũng có rất nhiều các công cụ, framework Back-end như Spring, SpringBoot, Express, Fastify, Electron, Activej, …

2.1.3 Ví dụ:

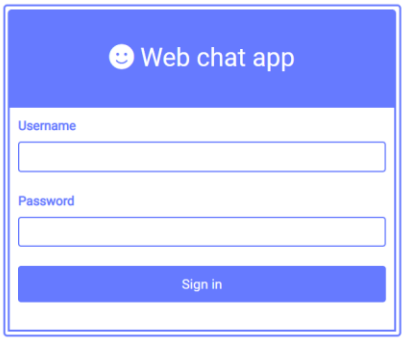
Một vài ví dụ về Front-end:



Giao diện trang web của trường đại học Tôn Đức Thắng

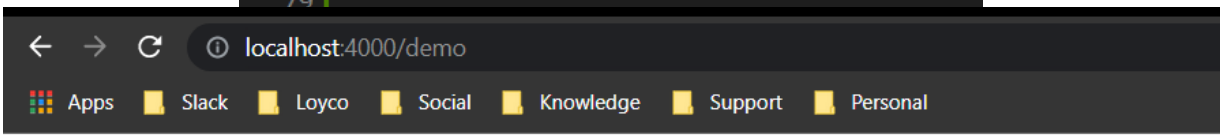
```
<body>
    <div class="join-container">
        <header class="join-header">
            <h1><i class="fas fa-smile"></i> Web chat app</h1>
        </header>
        <form id="login_form" class="login-form">
            <div class="alert alert-danger text-center" id="alert_danger" style="display: none;"></div>
            <div class="form-outline mb-4">
                <label class="form-label">Username</label>
                <input type="text" id="username" name="username" class="form-control" />
            </div>

            <div class="form-outline mb-4">
                <label class="form-label">Password</label>
                <input type="password" id="password" name="password" class="form-control" />
            </div>
            <button type="submit" class="btn btn-primary btn-block mb-4 btn-custom">Sign in</button>
        </form>
    </div>
</body>
```

```css
.chat-header {
  background: var(--dark-color-a);
  color: #fff;
  border-top-left-radius: 5px;
  border-top-right-radius: 5px;
  padding: 15px;
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.chat-main {
  display: grid;
  grid-template-columns: 1fr 3fr;
}
      You, 19 hours ago • Updating ...
.chat-sidebar {
  background: var(--dark-color-b);
  color: #fff;
  padding: 20px 20px 60px;
  overflow-y: scroll;
}
```



Một đoạn code HTML, CSS và kết quả của nó

```
app.get("/demo", (req, res) => {
  res.json(usersCredentail);
  return;
})
```




```
{
    "user1": "123456",
    "user2": "123456",
    "user3": "123456"
}
```

Một đoạn code Back-end và kết quả trả về của nó

2.2 Node.js để xây dựng Front-end hay Back-end? Vì sao?

Trên trang web chính của Nodejs đã định nghĩa: *"Node.js là một JavaScript runtime được build dựa trên engine JavaScript V8 của Chrome. Node.js sử dụng kiến trúc hướng sự kiện event-driven, mô hình non-blocking I/O làm cho nó nhẹ và hiệu quả hơn. Hệ thống nén của Node.js, npm, là hệ thống thư viện nguồn mở lớn nhất thế giới."*

Việc có mô hình non-blocking I/O (non-blocking Input/Output) đồng nghĩa với việc Nodejs có thể xử lý đồng thời cùng lúc nhiều request mà không cần phải chờ request trước hoàn thành mới xử lý tới request tiếp theo đã giúp giải quyết vấn đề nhức nhối ở các server cũ là nhu cầu về đa luồng

Cũng vì Nodejs sử dụng kiến trúc hướng sự kiện event-driven nên việc xây dựng Frontend bằng Nodejs cũng phát triển rất mạnh mẽ, có thể làm mới giao diện người dùng với các thông tin mới mà không cần tốn thời gian tải lại toàn bộ trang web. Điều này được thấy rất nhiều ở các framework Frontend hiện nay như angular và react.

Tóm lại, Nodejs hoàn toàn có thể sử dụng để xây dựng cả Frontend và Backend đều được.

2.3 Express.js để xây dựng Front-end hay Back-end? Vì sao?

Express.js là một framework của Nodejs có mã nguồn mở và miễn phí được sử dụng để thiết kế và phát triển các ứng dụng web. Điểm mạnh của express chính là khả năng tổ chức kiến trúc code rất tốt, chia rõ nhiệm vụ của từng kiến trúc, dễ dàng triển khai kiến trúc MVC. Và cũng vì lý do đó, lập trình viên thường sử dụng express để xây dựng Back-end vì nó sẽ tạo nên một codebase (không gian code) dễ đọc, dễ hiểu và dễ bảo trì. Bên cạnh đó express cũng có các tính năng nổi bật như router (định tuyến), middleware (trung gian), view template (khuôn mẫu).

Trên thực tế, express cũng có thể sử dụng để phát triển Front-end nhưng chỉ dừng lại ở mức html thuần và có hỗ trợ một ít trong việc hiển thị data từ server giống như các tập tin jsp. Có rất nhiều framework mạnh mẽ và tiện lợi để phát triển front-end như Angular và React. Và với xu hướng hiện tại của các phần mềm là microservice (một dạng thiết kế hệ thống sẽ chia nhỏ các kiến trúc riêng biệt của hệ thống để mỗi phần chỉ

đảm nhận đúng chức năng của nó như vậy sẽ giúp dễ xây dựng cũng như bảo trì), thì việc chia rạch ròi 2 phần là Front-end và Back-end rất quan trọng nên thường trong thực tế lập trình viên sẽ sử dụng express để phát triển Back-end và một công nghệ khác để phát triển Fron-end

Tóm lại, trong thực tế express thường được dung để phát triển Back-end.

2.4 Express.js là thư viện mã nguồn mở hay Framework hay là ngôn ngữ lập trình web hay công cụ xây dựng web? Vì sao?

Như đã trình bày ở phần trên Express.js là một framework của NodeJs. Ở đây chúng ta khẳng định Express không phải là một ngôn ngữ lập trình vì nó được xây dựng dựa trên JavaScript và vẫn được phát triển với cú pháp của JavaScript chứ không phải được định nghĩa như môn ngôn ngữ riêng biệt. Express lại càng không phải là một công cụ để xây dựng web vì nó không có giao diện người dùng. Tuy nhiên express cũng không phải là một thư viện vì thư viện sẽ là một tập hợp các chức năng, các lớp được viết sẵn và sẽ hoạt động một cách bị động, chỉ được gọi khi nào chúng ta cần dung, nhưng ở đây express lại hoạt động như một khung chương trình và công việc của lập trình viên là bổ sung thêm code để hoàn thiện chương trình của mình.

3. 51900712 – Trương Tuấn Thịnh:

3.1 Front-end, Back-end là gì? Ví dụ:

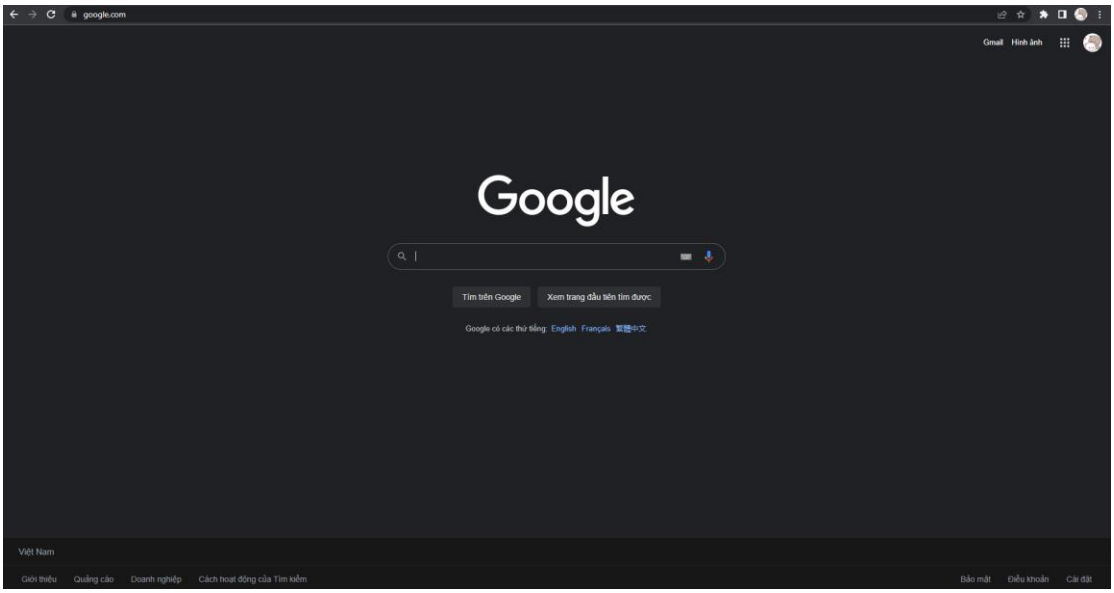
3.1.1 Front-end:

Front-end là phần nhìn thấy được của một trang web hay ứng dụng, phần mềm. Đó là phần tương tác với người dùng hay phía Client.

Nói chung, tất cả mọi thứ từ font chữ, màu sắc, hình ảnh cho tới các menu, scrollbar,... nhìn thấy được, tương tác được với người dùng là Front-end của một trang Web/ ứng dụng.

Front-end thường được xây dựng bằng 3 ngôn ngữ chính gồm HTML, CSS và JavaScript.

Ví dụ:



Trên là giao diện trang chủ của Google.com, Front-end của trang web là toàn bộ những gì trên hình như cái button, textbox, font chữ, logo google,...

3.1.2 Back-end:

Back-end là phần ngược lại với Front-end, nếu Front-end là tất cả những gì nhìn thấy được bằng mắt thường, thì Back-end chính là phần chìm của hệ thống một trang web hay ứng dụng.

Back-end bao gồm các logic thuật toán xử lý, truy xuất dữ liệu, nhờ đó mà trang web có thể hiển thị dữ liệu lưu trên hệ thống cũng như ghi dữ liệu vào máy chủ hệ thống đó.

Ví dụ:

Để nội dung gợi ý được hiển thị lên giao diện cho người dùng thì đó là kết hợp giữa front-end và back-end. Front-end là thứ quyết định vẻ đẹp bên ngoài, tính tiện dụng thì Back-end chính là thứ quyết định nội dung hiển thị.

3.2 Node.js để xây dựng Front-end hay Back-end? Vì sao?

NodeJS không phải là một ngôn ngữ, thư viện hay một framework, Nodejs là môi trường thực thi (Runtime Environment) mã nguồn mở, đa nền tảng, chạy trên JavaScript và được xây dựng trên V8 JavaScript engine của Chrome - V8 thực thi mã JavaScript bên ngoài trình duyệt.

Vì Nodejs là môi trường thực thi nên nó có thể tự chạy server-side trên chính môi trường thực thi của chính nó.

Ngoài ra, nodejs còn có npm (Node Package Manager) với một cộng đồng lập trình viên cũng như người dùng mạnh mẽ, đa dạng tài nguyên và hỗ trợ tốt cho lập trình server-side cũng như client-side.

Tóm lại, Nodejs có thể sử dụng để xây dựng Back-end lẫn Front-end.

3.3 ExpressJS để xây dựng Front-end hay Back-end? Vì sao?

ExpressJS là một framework được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. Expressjs hỗ trợ các method HTTP và midleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.

Một số tính năng chính của ExpressJS như:

Phát triển máy chủ nhanh chóng: ExpressJS cung cấp nhiều tính năng dưới dạng các hàm để dễ dàng sử dụng ở bất kỳ đâu trong chương trình. Điều này đã loại bỏ nhu cầu viết mã từ đó tiết kiệm được thời gian.

Phần mềm trung gian Middleware: Đây là phần mềm trung gian có quyền truy cập vào cơ sở dữ liệu, yêu cầu của khách hàng và những phần mềm trung gian khác. Phần mềm Middleware này chịu trách nhiệm chính cho việc tổ chức có hệ thống các chức năng của Express.js.

Định tuyến - Routing: Express js cung cấp cơ chế định tuyến giúp duy trì trạng thái của website với sự trợ giúp của URL.

Tạo mẫu - Templating: Các công cụ tạo khuôn mẫu được Express.js cung cấp cho phép các nhà xây dựng nội dung động trên các website bằng cách tạo dựng các mẫu HTML ở phía máy chủ.

Gỡ lỗi - Debugging: Để phát triển thành công các ứng dụng web không thể thiết đi việc gỡ lỗi. Giờ đây với Expressjs việc gỡ lỗi đã trở nên dễ dàng hơn nhờ khả năng xác định chính xác các phần ứng dụng web có lỗi.

Vì thế, ExpressJS chỉ được sử dụng để hỗ trợ phát triển Server-side của một trang web hay Back-end và chỉ hỗ trợ ngôn ngữ JavaScript.

3.4 ExpressJS là thư viện mã nguồn mở hay Framework hay là ngôn ngữ lập trình web hay công cụ xây dựng web? Vì sao?

Đầu tiên, ExpressJS không thể là một ngôn ngữ lập trình bởi vì nó sử dụng ngôn ngữ JavaScript (một ngôn ngữ lập trình đã tồn tại).

ExpressJS càng không thể là một công cụ xây dựng trang web, bởi nó không hề có giao diện sử dụng, hay chức năng, tính năng để người dùng xây dựng trang web.

Thư viện (Library) là một tập hợp chức năng (function), lớp (class)... được cung cấp sẵn và có thể tái sử dụng. Mỗi function hay class đều được phục vụ cho một công việc cụ thể nào đó. Thư viện chỉ được sử dụng khi ta gọi đến hàm, lớp nào đó trong nó.

Framework được xem như là bộ khung sẵn có của chương trình, chứa tập hợp thư viện phần mềm, API... hỗ trợ cho việc phát triển sao cho tiết kiệm thời gian và hợp lý nhất.

Theo những thông tin như tính năng về ExpressJS ở mục 3.3 thì ExpressJS cũng không phải là một thư viện, bởi nó bao gồm các quy tắc phải tuân theo, và dựa trên những quy tắc đó giúp người dùng dễ dàng hơn, tiết kiệm hơn thời gian xây dựng trang web.

Vì thế, ExpressJS là một framework cho Nodejs, hỗ trợ trong việc phát triển web phía Server.

TÀI LIỆU THAM KHẢO

1. https://expressjs.com
2. https://itnavi.com.vn/blog/expressjs-la-gi/?amp
3. https://socket.io
4. https://topdev.vn/blog/socket-la-gi-websocket-la-gi/
5. https://viblo.asia/p/websocket-la-gi-Ljy5VxkbZra
6. https://topdev.vn/blog/socket-la-gi-websocket-la-gi/
7. https://bizflycloud.vn/tin-tuc/websocket-la-gi-uu-nhuoc-diem-cua-websocket-khi-su-dung-lam-phuong-thuc-giao-tiep-trong-moi-truong-internet-20210122155209401.htm
8. https://wiki.tino.org/websocket-la-gi/
9. https://nhanhoa.com/tin-tuc/websocket-la-gi.html
10. https://vietnix.vn/websocket-la-gi/
11. https://vietnix.vn/websocket-la-gi/
12. https://kb.pavietnam.vn/websocket-la-gi-hieu-ro-ve-websocket.html