HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

PHẠM XUÂN MẠNH

XÂY DỰNG HỆ THỐNG IOT GIÁM SÁT CÁC TRẠM PHÁT THANH CẤP XÃ TRONG HỆ THỐNG TRUYỀN THANH KHÔNG DÂY

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

PHẠM XUÂN MẠNH

XÂY DỰNG HỆ THỐNG IOT GIÁM SÁT CÁC TRẠM PHÁT THANH CẤP XÃ TRONG HỆ THỐNG TRUYỀN THANH KHÔNG DÂY

Chuyên ngành: Kỹ thuật viễn thông

Mã số: 8.52.02.08

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN QUỐC UY

HÀ NỘI – 2020

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi. Nội dung của luận văn có tham khảo và sử dụng các tài liệu, thông tin được đăng tải trên những tạp chí và các trang web theo danh mục tài liệu tham khảo. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Tác giả luận văn

PHẠM XUÂN MẠNH

LỜI CẢM ƠN

Để có thể hoàn thành tốt luận văn của mình, đầu tiên em xin gửi lời cảm ơn chân thành tới thầy TS. Nguyễn Quốc Uy, người đã đồng hành cùng em trong suốt chặng đường vừa qua và cũng là người luôn tận tình hướng dẫn em trong suốt quá trình thực hiện luận văn này.

Em xin chân thành cảm ơn Ban giám đốc Học viện Công nghệ Bưu chính Viễn thông, quý thầy cô trong Học viện đã tận tâm giảng dạy và truyền đạt những kiến thức cũng như những kinh nghiệm quý báu trong suốt quá trình học tập của em tại Học viện. Vốn kiến thức được tiếp thu trong quá trình học tập không chỉ là nền tảng cho quá trình thực hiện luận văn tốt nghiệp mà còn là hành trang quý báu cho sự nghiệp của em sau này.

Em cũng xin cảm ơn sự ủng hộ và giúp đỡ nhiệt tình của gia đình, bạn bè, những người thân đã động viên, giúp đỡ em trong suốt quá trình học tập và thực hiện luận văn tốt nghiệp này.

Mặc dù đã cố gắng hết sức, song chắc chắn luận văn không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và góp ý của quý thầy cô để em có thể rút kinh nghiệm và hoàn thành tốt hơn luận văn tốt nghiệp này.

Cuối cùng em xin kính chúc quý thầy cô, gia đình và bạn bè dồi dào sức khỏe, thành công trong sự nghiệp.

Em xin chân thành cảm ơn!

Hà Nội, ngày 10 tháng 5 năm 2020

Học viên

PHẠM XUÂN MẠNH

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

API: Application Programming Interface	Giao diện lập trình ứng dụng
CSS: Cascading Style Sheets	Ngôn ngữ tạo phong cách cho web
DOM: Document Object Model	Mô hình đối tượng trong tài liệu
HTML: HyperText Markup Language	Ngôn ngữ đánh dấu siêu văn bản
HTTP: HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
IoT: Internet of Things	Internet vạn vật
JSON: Javascript Object Notation	Kiểu dữ liệu mở rộng trong Javascript
JSX: Javascript XML	Cú pháp mở rộng cho Javascript
RAM: Random Access Memory	Bộ nhớ khả biến
RDBMS: Relational Database Management System	Hệ thống quản lý cơ sở dữ liệu quan hệ
REST: Representational State Transfer	Một dạng chuyển đổi cấu trúc dữ liệu
UI: User Interface	Giao diện người dùng
URL: Uniform Resource Locator	Đường dẫn
XML: Extensible Markup Language	Ngôn ngữ đánh dấu mở rộng

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ

LỜI MỞ ĐẦU

1. Lí do chọn đề tài

Trên thế giới, các nghiên cứu về công nghệ Radio số (Internet radio) đã được quan tâm từ khá lâu với nhiều kết quả công bố, đặc biệt các hệ thống truyền thanh qua Internet, tuy nhiên việc ứng dụng công nghệ này tại Việt Nam vẫn chưa được thực sự quan tâm và mới chỉ bắt đầu triển khai mấy năm trở lại đây. Hướng ứng dụng chính trên thế giới hiện nay được tập trung nghiên cứu và triển khai là xây dựng các hệ thống máy chủ nội dung số và máy chủ phát thanh để truyền phát nội dung số qua Internet đến các thiết bị cuối thu tín hiệu qua Internet. Những thiết bị này có thể tích hợp các nút bẩm cảnh báo thông qua Internet để gửi tín hiệu lên máy chủ và hệ thống phần mềm quản lý.

Ở Việt Nam, dù đã thu hút được khá nhiều sự quan tâm của các nhóm nghiên cứu, đặc biệt là nhóm nghiên cứu của Học viện Công nghệ Bưu chính Viễn thông, do thầy TS. Nguyễn Quốc Uy chủ trì về mảng Radio số, truyền thanh qua Internet. Nhóm nghiên cứu tập trung vào phát triển ứng dụng quan trọng của hệ thống truyền thanh qua Internet kết hợp truyền thanh qua sóng FM tại Việt Nam. Các ưu điểm của hệ thống truyền thanh qua Internet như sau:

- Tại những khu vực thành thị và tại những vùng sâu, xa, dân tộc thiểu số, hệ thống đều có thể truyền dữ liệu thông qua Internet và sóng FM. Phần mềm trên hệ thống máy chủ phát sóng có thể giám sát trạng thái và điều khiển các máy thu Internet Radio ứng dụng công nghệ IoT.

- Trong các trường hợp thiên tai, cứu nạn, hệ thống ngoài chức năng thu phát, truyền thanh, còn có thể nhận phản hồi trong những tình huống khẩn cấp.

Hệ thống nếu được triển khai trên các tỉnh, để phục vụ phủ sóng cho các vùng sâu, vùng xa, trong những tình huống lưu động khẩn cấp là một giải pháp mang lại lợi ích to lớn cho phát triển kinh tế, ý nghĩa chính trị và đảm bảo an ninh quốc phòng và chủ quyền quốc gia. Chính vì vậy, việc xây dựng hệ thống truyền thông radio số đồng nhất 03 cấp (tỉnh, huyện, xã) phục vụ phổ biến kiến thức thông

tin kinh tế, văn hóa, xã hội, an ninh, quốc phòng và các thông tin khẩn cấp của tỉnh Đắk Lắk và khu vực Tây Nguyên, đảm bảo an toàn thông tin và tính dự phòng hệ thống trong trường hợp khẩn cấp có ý nghĩa ứng dụng rất lớn.

2. Tổng quan vấn đề cần nghiên cứu

IOT hiện nay đang là một xu hướng mạnh mẽ trên toàn thế giới, mở ra những cơ hội chưa từng có cho các nền kinh tế, doanh nghiệp, tổ chức và các cá nhân để cạnh tranh trong môi trường mới. Phạm vi ứng dụng công nghệ IoT thực sự rộng lớn và đa dạng, từ quản lý giao thông, quản lý đô thị, quản lý môi trường, ứng phó khẩn cấp đến các dịch vụ y tế chăm sóc sức khỏe, nhà thông minh, hướng tới nữa là thành phố thông minh và tất nhiên là cả hệ thống truyền thanh không dây qua Internet.

Trong quá trình xây dựng hệ thống truyền thanh không dây đồng nhất 03 cấp cho một tỉnh, việc kiểm soát tình trạng hoạt động của hàng trăm xã, hàng trăm điểm thu phát sóng qua Internet Radio là vô cùng cần thiết. Việc này chỉ được thực hiện nếu áp dụng công nghệ IoT vào trong việc xây dựng hệ thống, kết hợp với các quy trình hoạt động của hệ thống truyền thanh qua Internet.

3. Mục đích nghiên cứu của luận văn

Trong khuôn khổ luận văn, tác giả sẽ xây dựng một hệ thống IoT để theo dõi thông số từ các hệ thống Internet Radio phát thanh từ các xã. Dữ liệu sẽ được gửi về hệ thống phần mềm trên server cấp tỉnh. Hệ thống gồm demo phần cứng và phần mềm đáp ứng được yêu cầu đặt ra.

4. **Đối tượng và phạm vi nghiên cứu**

Về cơ bản, IoT platform là một hệ thống theo dõi và quản lý các thông số của máy thu Internet Radio từ xa mà không cần phải đến trực tiếp máy thu Internet Radio. Hệ thống tự động cập nhật các thông số cần thiết cho người quản lý, giúp cho người quản lý dễ dàng theo dõi cũng như quản lý nhiều máy thu Internet Radio một lúc. Việc theo dõi và quản lý thông qua IoT platform này vừa giúp tiết kiệm nhân lực, dễ dàng và đặc biệt là rất nhanh chóng, đáp ứng yêu cầu độ tin cậy cao của hệ thống khi hoạt động trong thực tế.

5. **Phương pháp nghiên cứu**:

Nhờ sự hướng dẫn cũng như định hướng của thầy hướng dẫn, học viên thực hiện tìm kiếm và thu thập tài liệu, bài báo đã được công bố để tìm hiểu lý thuyết cơ bản về IoT platform, từ đó tìm hiểu và phân tích các kết quả đã có được và xây dựng một IoT platform nhỏ chạy thực tế để khảo sát đưa ra kết quả và định hướng nghiên cứu tiếp sau này.

Nội dung của luận văn gồm 4 phần chính:

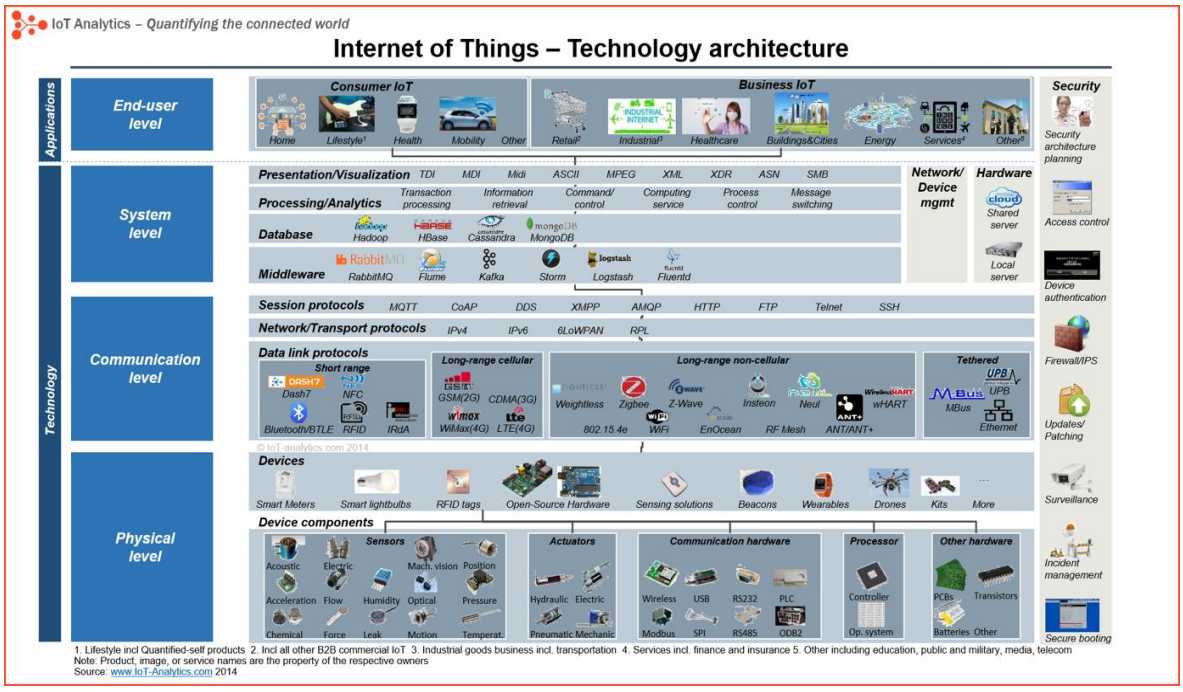
- *Chương 1: Nghiên cứu tổng quan, thuyết minh ý tưởng*
- *Chương 2: Mô hình hệ thống truyền thanh không dây và ứng dụng công nghệ iot*
- *Chương 3: Xây dựng hệ thống iot quản lý hoạt động của các trạm phát thanh*
- *Chương 4: Kiểm thử IOT platform với phần cứng mô phỏng máy thu Internet Radio*

CHƯƠNG 1: NGHIÊN CỨU LÝ THUYẾT TỔNG QUAN VỀ INTERNET OF THINGS

1.1. Tổng quan về IoT và IoT platform

Internet of Things, hay IoT là khái niệm kết nối các thiết bị với nhau và với Internet. IoT là một mạng lưới khổng lồ gồm các vật (things) và con người được kết nối - tất cả đều thu thập và chia sẻ dữ liệu với nhau. Việc kết nối có thể thực hiện qua Internet, 3G, Wifi, ZigBee, Bluetooth… Các thiết bị được kết nối với nhau và cùng kết nối trong cùng một mạng. Con người có thể giám sát, điều khiển thiết bị, thu thập dữ liệu ở bất cứ nơi nào và bất cứ thời điểm nào thông qua IoT Platform.

Hệ thống IoT được ứng dụng rất rộng rãi trong thực tế. Các lĩnh vực ứng dụng IoT và kiến trúc tổng quát hệ thống IoT có thể được tìm kiếm trong hình vẽ dưới đây.



Hình 1.1: Các thành phần cơ bản của IoT system

IoT platform là trung tâm của việc triển khai IoT, là một phần mềm để khai báo, định nghĩa thiết bị phần cứng, các giao thức kết nối và các ứng dụng phần mềm khác. Nó cung cấp một giải pháp hiệu quả cho việc quản lý và cấu hình thiết bị, thu thập và phân tích dữ liệu, có khả năng kết nối với các dịch vụ đám mây và

tích hợp với điện thoại thông minh và các thiết bị khác của người sử dụng. Có rất nhiều các IoT platform khác nhau, tuy nhiên hầu hết tất cả đều có các thành phần cơ bản chung giống nhau:

- Thiết bị kết nối: Chúng là các loại máy móc, cảm biến hay các thiết bị kết nối khác thực hiện một hành động cụ thể: thu thập dữ liệu, kết nối với nhau, truyền và nhận dữ liệu, ...
- Phương thức kết nối: Dựa trên mạng viễn thông mà các thiết bị có thể kết nối, giao tiếp được với nhau và với server/cloud. Điều này phụ thuộc vào yêu cầu của dự án IoT từ đó chọn ra phương thức kết nối hiệu quả nhất.
- Xử lý dữ liệu: Được xử lý ở trên server/cloud. Nhận dữ liệu từ các thiết bị, từ đó phân tích và đưa ra hành động sẽ được thực hiện trong IoT platform.
- Giao diện: Cung cấp cho người dùng một giao diện trực quan để có thể tương tác và nhìn thấy được hoạt động của toàn bộ hệ thống.

1.2. Các yêu cầu và đặc điểm của IoT platform

Các IoT platform đảm bảo việc tích hợp liền mạch các phần cứng khác nhau bằng cách sử dụng một loạt các giao thức giao tiếp phổ biến (như MQTT, HTTP, CoAP, …). Sử dụng các API do IoT platform cung cấp, ta có thể tải dữ liệu IoT thu thập được vào các hệ thống phân tích, lưu trữ hoặc xử lý dữ liệu tới các thiết bị được kết nối hoặc truyền dữ liệu giữa chúng bằng việc sử dụng các loại ứng dụng người dùng khác nhau. Để đánh giá xem liệu một IoT platform có thật sự tốt hay không, cần dựa vào các tiêu chí sau đây:

- Tính khả mở: Cho phép chạy trên các thiết bị có nền tảng hệ điều hành khác nhau.
- Dễ sử dụng: Cung cấp một giao diện dễ nhìn, thân thiện, cung cấp các API đa dạng để người dùng có thể tùy chỉnh hệ thống theo cách riêng.
- Tương tác và thích hợp: Cung cấp khả năng xử lý nhiều loại thiết bị phần cứng thông qua nhiều loại giao thức kết nối để truyền dữ liệu cho server/cloud.

- Tính bảo mật: Mã hóa thông tin truyền giữa các thiết bị với server/cloud, kiểm soát quyền truy cập vào hệ thống, bảo mật dữ liệu lưu trữ, ...

Để đạt được giá trị từ Internet of Things (IoT), việc cần phải có là một nền tảng để tạo và quản lý ứng dụng, chạy các phân tích, lưu trữ và bảo mật dữ liệu. Giống như một hệ điều hành dành cho máy tính, một nền tảng làm rất nhiều thứ đằng sau đó, tạo tra môi trường cho các nhà phát triển, giúp nhà quản lý và người dùng sử dụng dễ dàng hơn và tiết kiệm chi phí hơn. Do đó, các IoT platform góp phần xây dựng những ứng dụng to lớn và được sử dụng rộng rãi trong nhiều lĩnh vực như: điện lực, giao thông, xây dựng nhà thông minh, thành phố thông minh …

1.3. Kết luận về nhu cầu thực tế và khả năng áp dụng của đề tài

Ở Việt Nam hiện nay, số lượng các máy thu Internet Radio là rất lớn do đó để quản lý và theo dõi các thông số của máy thu Internet Radio cần một số lượng lớn nhân lực. Điều này dẫn đến việc lãng phí nguồn nhân lực, trong khi nguồn nhân lực đó cần thiết cho những công việc khác quan trọng hơn. Hơn nữa, để truyền tải công suất điện lớn từ nơi sản xuất đến nơi tiêu thụ, thì giải pháp tăng điện áp để hạn chế tổn thất công suất và giảm giá thành đầu tư đường dây là một lựa chọn tối ưu. Để lượng công suất tải truyền đi càng lớn thì điện áp càng cao. Vì thế mà việc quản lý và theo dõi các thông số trực tiếp tại trạm biếm áp là rất khó khăn và khá nguy hiểm cho người tham gia thực hiện. Để giải quyết được bài toán thực tế này thì việc xây dựng một IoT platform để quản lý và theo dõi thông số từ máy thu Internet Radio là rất cần thiết. Chính vì thế, trong khuôn khổ đề tài em mong muốn xây dựng một IoT platform dùng để quản lý các máy thu Internet Radio và theo dõi các thông số của các máy thu Internet Radio đó.

CHƯƠNG 2: MÔ HÌNH HỆ THỐNG TRUYỀN THANH KHÔNG DÂY VÀ ỨNG DỤNG CÔNG NGHỆ IOT

2.1. Lý thuyết về hệ thống truyền thanh không dây

Truyền thanh không dây hiện nay không chỉ còn được hiểu là truyền thanh qua sóng FM mà nó đã được nâng lên một tầng cao hơn – truyền thanh qua Internet. Truy cập Internet rất quan trọng, bởi vì nó đòi hỏi một đài phát thanh có khả năng giao tiếp với một mạng lưới toàn cầu. Hiện nay, các thiết bị sử dụng phổ biến nhất chủ yếu là máy tính xách tay, để bàn và điện thoại di động. Chúng ta không thể quên về điện thoại di động, đặc biệt là các thiết bị nghe nhạc bỏ túi như iPod của Apple. Trong năm 2010 và 2011 Internet Radio đã trở thành xu hướng mới trong việc phát triển khả năng kết nối Internet. Mục đích chính là cung cấp cho người dùng truy cập vào một loạt các nội dung thông tin, đặc biệt là thông qua các mạng truyền thông toàn cầu. Kết nối Internet tương tự có thể được thực hiện cả trong cách truyền thống, ví dụ - thông qua cáp kết nối được đến thiết bị - thông qua việc sử dụng công nghệ không dây. Ví dụ về các công nghệ như là sóng radio FM và tín hiệu vệ tinh hoặc cơ sở hạ tầng điện thoại di động, cho phép chuyển hàng Megabit mỗi giây, đảm bảo nội dung có chất lượng rất cao. Sự phát triển của Internet là cần thiết nhưng chưa đủ cho Internet Radio. Sự xuất hiện của Internet radio sẽ không thể thực hiện được mà không có phát triển đáng kể của công nghệ truyền âm thanh theo thời gian thực trên phạm vi toàn cầu. Với mục đích này cần phải có kết nối Internet với băng thông lớn.

Vai trò của Internet Radio rất quan trọng, đặc biệt trong trường hợp truyền tải đa phương tiện, đòi hỏi kết nối Internet tốc độ cao, đi kèm chi phí đang tăng lên tương ứng với số lượng người dùng. Công nghệ tiến bộ kỹ thuật trong việc truyền tải hiệu quả của âm thanh cũng là một yếu tố quan trọng, được thực hiện bởi các phương pháp mới và hiệu quả hơn trong việc nén dữ liệu. Với việc nén dữ liệu, chúng ta chỉ cần ít băng thông để gửi cùng một lượng dữ liệu. Từ đó, sự ra đời công nghệ streaming, công nghệ mà cho phép bạn truyền âm thanh và video trong nhiều luồng. Công nghệ streaming hiện nay có thể được thực hiện bởi các phần mềm độc

quyền hoặc mã nguồn mở để truyền và nhận sóng vô tuyến hoặc các tin tức đa phương tiện. Chương trình đầu tiên đã được thực hiện vào tháng 04/1996 khi công ty Real Networks, tác giả của phần mềm Real Player nổi tiếng và coder đặc trưng bởi tỷ lệ nén tốt. Kết quả là các định dạng phương tiện truyền thông trực tuyến đã được giới thiệu bởi những người khổng lồ CNTT như Microsoft với định dạng file WMA (Windows Media Audio) và Windows Media Player hay như Apple với công nghệ Quick Time.

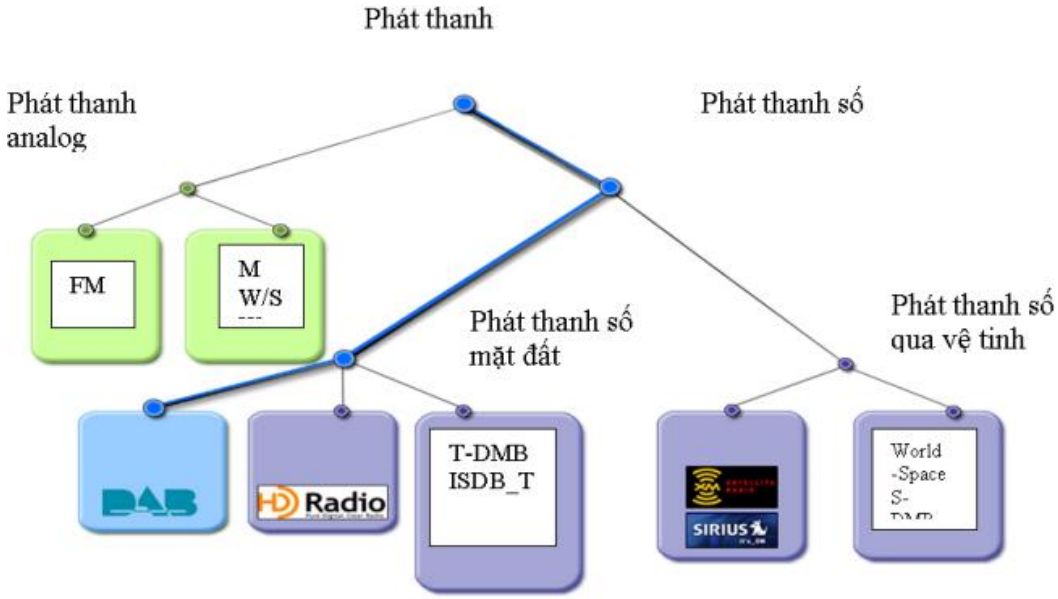
Công nghệ Shoutcast được phát triển như một phương pháp cho phép truyền phát các file nhạc ở định dạng MP3, đi kèm với công nghệ Podcasting cho phép tự động lựa chọn và tải các chương trình này, có thể được chạy lại khi nào muốn. Sự kết hợp Shoutcast và Podcast mang đến một giải pháp Internet Radio khá hoàn thiện. Điều này dẫn đến việc các thiết bị nghe nhìn không cần thiết phải là các đài có sóng FM, không cần thiết phải là các máy tính để bàn, laptop mà chỉ cần đơn giản là thiết bị nhỏ gọn mà kết nối được Internet, qua 3G, wifi là có thể đài phát thanh Internet.

Sự khác biệt giữa các đài phát thanh cổ điển và đài phát thanh Internet không chỉ nằm ở các giải pháp công nghệ mà còn nằm ở khía cạnh pháp luật. Trước hết, phát thanh Radio Internet không cần phải xin giấy phép như trong trường hợp của đài phát thanh truyền thống. Lí do là không có nhu cầu về phân bổ tần số. Internet là một phương tiện để chung và tất cả mọi người trên thế giới có thể sử dụng nó. Họ có quyền lựa chọn được nghe gì, xem gì. Internet Radio sẽ là giải pháp tốt nếu nội dung phong phú, dẫn đến số người nghe có thể lên đến hàng triệu.

Waldemar Dubaniowski tại hội thảo "Dịch vụ nghe nhìn trên Internet và bản quyền bảo vệ trong môi trường kỹ thuật số" đã lưu ý rằng sự phát triển nhanh chóng của phương tiện truyền thông số như phát thanh truyền hình đã chọn một hướng đi khác nhau. Đài phát thanh Internet đã có một sự phát triển cực kỳ mạnh mẽ bởi chi phí phát triển và vận hành Radio Internet là rất thấp so với hệ thống Radio cổ điển.

Radio Internet phát triển mạnh mẽ ở mảng cá nhân, khi họ khởi đầu từ những công việc yêu thích như hát, kể chuyện, làm Vlog… rồi sau đó khi lượng người

dùng quan tâm, số lượng người theo dõi tăng lên, họ phát triển kênh Internet Radio của họ trở thành một kênh truyền thông rất mạnh, nội dung phong phú, đáp ứng yêu cầu của thính giả. Vì tính cá nhân, phát thanh truyền hình qua Internet có tập khách hàng rất hẹp, ví dụ, sinh viên của nhạc phim, các bạn trẻ, yêu thích bóng đá… Bằng cách này, ngày càng có nhiều đài phát thanh Internet được lấp đầy khoảng trống đã tồn tại nhiều năm trong phát thanh truyền hình truyền thống. Chúng tương ứng với các yêu cầu và nhu cầu của khán giả mà các đài phát thanh truyền thống, vì nhiều lý do, sẽ không bao giờ có thể cung cấp cụ thể.



Hình 2.1: Tiến trình phát triển của phát thanh trên thế giới

Tóm lại, các Radio truyền thống có một phạm vi rộng lớn, chất lượng âm thanh tốt, nhưng thiếu một con đường thuận tiện cho việc áp dụng công nghệ mới. Việc phải được cấp giấy phép hoạt động bởi cơ quan có thẩm quyền làm hạn chế số lượng đài phát thanh truyền thống. Mặt khác, phát thanh truyền hình qua Internet có thể được mô tả như một hình thức giao tiếp với phạm vi không giới hạn (vì nó có thể được sử dụng từ bất cứ nơi nào trên thế giới), và nó có một điểm mạnh vô cùng đó là khả năng tương tác với người nghe, mặc dù số lượng không nhiều. Chi phí phát triển một kênh Internet Radio cũng thấp hơn rất nhiều so với truyền thống là một ưu điểm nổi trội dẫn đến việc lép vế của Radio truyền thống.

2.2. Mô hình hệ thống truyền thanh không dây đồng nhất 3 cấp

Trong mục này, sau khi nghiên cứu về công nghệ truyền thông qua Internet, nhóm tác giả đi vào thiết kế mô hình hệ thống truyền thông không dây đồng nhất 03 cấp, áp dụng cho hệ thống truyền thông tại Đắk Lắk. Trước tiên nhóm thực hiện đề tài sẽ làm rõ khái niệm Radio số đồng nhất 03 cấp:

Truyền thanh Radio số đồng nhất ba cấp, có nghĩa là hệ thống sử dụng công nghệ truyền thanh qua Internet. Nội dung được truyền từ máy chủ cấp Tỉnh, thông qua Internet (cáp quang, wifi, 3G) để truyền sóng đến các đài truyền thanh cấp Huyện, cấp Xã. Tại các điểm cấp Huyện, Xã có trang bị máy thu Internet Radio. Do đó các đài truyền thanh tại Tỉnh, Huyện, Xã đều có thể đồng loạt phát đi cùng một nội dung (đồng nhất về phát nội dung). Ngoài ra nội dung có thể được gửi lên máy chủ từ cấp Xã, cấp Huyện và cấp Tỉnh, và nội dung này có thể được lấy từ xã này, huyện này rồi phát sang xã khác, huyện khác (đồng nhất về tạo nội dung).

Ngoài tính đồng nhất, thì hệ thống phải đảm bảo hoạt động tốt khi Internet có vấn đề, do đó nhóm đề xuất sử dụng mô hình lai ghép, tức là mặc định truyền thanh qua Internet, nhưng khi Internet có vấn đề thì sẽ truyền thanh qua sóng FM. Ngoài tính đồng nhất, tính bền vững chính là điểm nổi bật của hệ thống mới so với hệ thống cũ.

Để đảm bảo các yêu cầu trên, hệ thống phải đảm bảo được sự đồng nhất từ phần cứng đến phần mềm. Do đó nhóm thực hiện đề tài phải nghiên cứu, thiết kế, xây dựng mô hình phần cứng, phần mềm, hệ thống lưu trữ và quản lí cơ sở dữ liệu cần thiết và quy trình hoạt động để đảm bảo cho toàn bộ hệ thống có thể hoạt động trơn tru, linh hoạt.

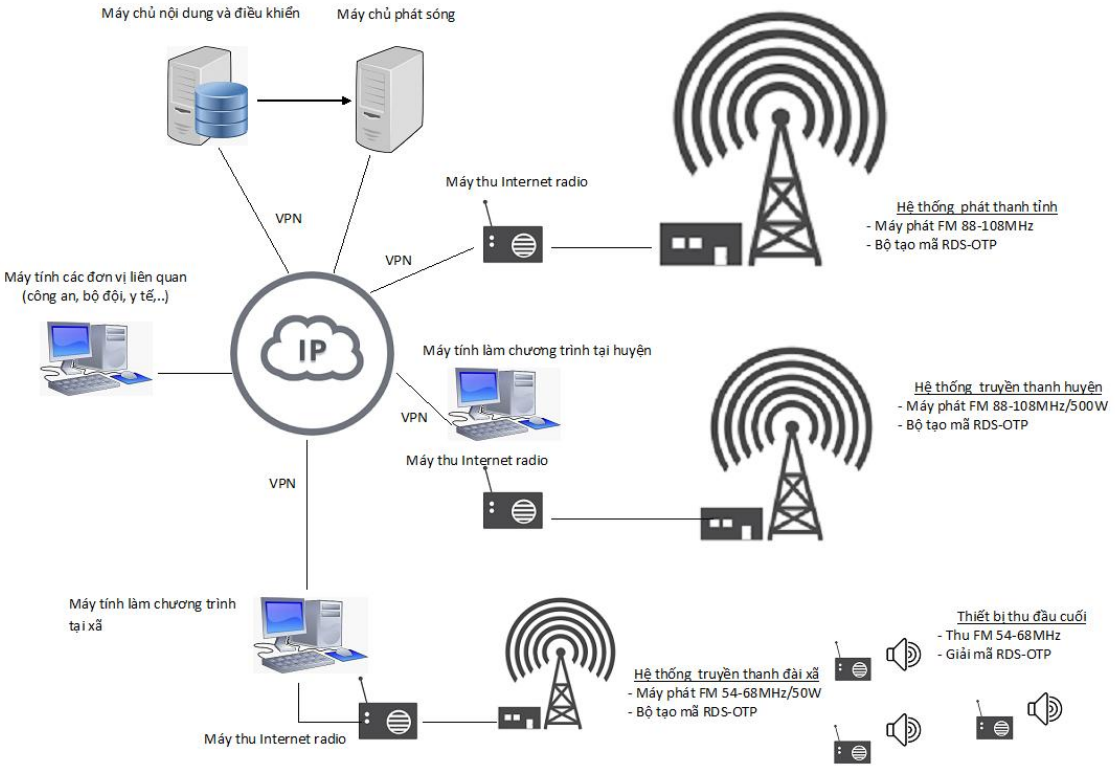
Hướng giải quyết mới của đề tài là nghiên cứu xây dựng hệ thống truyền thông radio số đồng nhất 3 cấp (tỉnh, huyện, xã) dựa trên công nghệ truyền thanh qua Internet. Mục tiêu đạt được là nâng cao chất lượng dịch vụ, hệ thống phải bảo mật, có thể phát thanh đồng nhất 3 cấp, qua Internet và không dây (truyền thanh qua sóng FM). Ngoài ra hệ thống có thể giám sát và điều khiển từ xa các thiết bị tại nơi

thu sóng Internet và phát sóng FM. Để cụ thể hóa mục tiêu trên, những nội dung cần nghiên cứu của đề tài như sau:

- Nghiên cứu lý thuyết, thiết kế, xây dựng mô hình hệ thống truyền thông radio số đồng nhất 3 cấp (tỉnh, huyện, xã), bao gồm nghiên cứu lí thuyết chính về xây dựng hệ thống Radio số, truyền thanh qua Internet, thiết kế mô hình phần cứng truyền phát thông tin, thiết kế mô hình cơ sở dữ liệu, thiết kế chức năng phần mềm lưu trữ dữ liệu và hiết kế phần mềm quản lý cho hệ thống máy chủ phát thanh.

- Nghiên cứu xây dựng hệ thống máy chủ sản xuất nội dung số và thống máy chủ phát sóng: từ thiết kế, xây dựng và tích hợp, chế tạo phần cứng đến thiết kế xây dựng và phát triển phần mềm.

- Nghiên cứu, thiết kế, chế tạo máy thu Internet Radio (cạnh máy phát FM), kết hợp nghiên cứu, thiết kế và chế tạo máy phát mã RDS-OTP để truyền tín hiệu số cùng với sóng FM và máy thu FM tích hợp bộ giải mã RDS-OTP. Đây là bộ giải pháp tổng thể đảm bảo rằng hệ thống phát thanh được bảo mật, tin cậy, đồng nhất 3 cấp.



Hình 2.2: Mô hình hệ thống truyền thông không dây đồng nhất 3 cấp

Qua khảo sát và nghiên cứu sơ bộ tại địa bàn tỉnh Đắk Lắk, em đề xuất mô hình hệ thống phần cứng, được mô tả như trong hình 2.2, bao gồm:

- Hệ thống máy chủ nội dung phát sóng;
- Hệ thống máy chủ phát sóng qua internet (streaming server);
- Hệ thống máy trạm làm chương trình tại tỉnh, huyện, xã;
- Hệ thống máy thu Internet Radio tại các huyện, xã; Máy tạo mã RDS-OTP; Hệ thống máy phát FM cấp huyện (88-108 MHz/500W) và cấp xã (54-68 MHz/50W) có tích hợp máy phát mã RDS-OTP;
- Hệ thống thiết bị đầu cuối có tích hợp module giải mã RDS-OTP, nhận tín hiệu và phát trực tiếp ra loa. Hệ thống sử dụng kết hợp (hybrid) giữa truyền phát qua FM và qua Internet.

Trong tình huống bình thường, hệ thống hoạt động qua đường truyền Internet từ cấp tỉnh đến cấp huyện, xã. Các máy thu Internet radio tại xã, huyện sẽ thu sóng trực tiếp từ hệ thống máy chủ phát sóng (streaming server), sau đó phát sóng đến các điểm thu FM tại các cụm dân cư. Trong các tình huống Internet bị đứt cáp (thiên tai, gãy đứt), hệ thống có thể tự động chuyển qua phát thanh qua FM từ đài phát thanh cấp Tỉnh đến các đài cấp huyện, xã. Ngoài ra khi Internet bị đứt cáp do thiên tai, các đài truyền thanh xã có thể sử dụng module sim 3G để máy thu Internet radio cấp xã có thể lấy được dữ liệu từ máy chủ phát sóng và sau đó phát đến các loa đặt tại cụm dân cư.

Sau khi xây dựng, chế tạo hệ thống, bước tiếp theo là thử nghiệm, đánh giá các tham số kỹ thuật, chức năng của hệ thống trong phòng thí nghiệm, và triển khai thực địa. Dựa trên các kết quả thử nghiệm, việc tối ưu các tham số của từng module phần cứng và phần mềm sẽ được tiến hành nhằm đạt được mục tiêu đề ra. Các tham số chính làm cơ sở cho việc tối ưu là tính bảo mật của hệ thống, sự ổn định khi vận hành trong cả những tình huống khẩn cấp.

2.3. Ưu nhược điểm hệ thống truyền thanh hiện nay và nhu cầu xây dựng hệ thống truyền thanh không dây mới

a) *Ưu điểm* của hệ thống truyền thanh truyền thống

Hạ tầng truyền dẫn phát sóng được đầu tư công nghệ mới, hiện đại, hiệu quả. Hạ tầng truyền dẫn phát sóng được chuyển đổi dần từ công nghệ tương tự sang công nghệ số nhằm nâng cao chất lượng dịch vụ và tiết kiệm băng tần, cụ thể là:

- Việc ứng dụng công nghệ số vào sản xuất chương trình phát thanh cho phép thực hiện các chương trình trực tiếp thuận tiện và dễ dàng hơn theo hướng mở, tương tác với thính giả, đáp ứng nhu cầu thông tin đa chiều.
- Việc thực hiện số hóa hệ thống sản xuất chương trình phát thanh đã góp phần quan trọng trong việc cung cấp tới công chúng những sản phẩm báo chí với chất lượng âm thanh và hình ảnh cao
- Việc phát triển phát thanh số, đưa các chương trình phát thanh tích hợp trên các thiết bị thông minh như điện thoại di động sẽ là xu thế phát triển của phát thanh hiện đại.

Nhược điểm

Hệ thống truyền thanh hiện nay ở Việt Nam cũng còn tồn tại các nhược điểm như:

- Một số các trang thiết bị trong dây truyền phát thanh được đầu tư từ rất lâu, quản lý, khai thác khó khăn phức tạp, đặc biệt khó khăn trong công tác sửa chữa. Giá thành thiết bị thay mới lại quá cao.
- Hệ thống truyền thanh được đầu tư tại nhiều tỉnh, thành phố hiện tại đã cũ. Nhiều hệ thống loa truyền thanh chất lượng chưa tốt gây phản ứng tiêu cực cho người nghe. Điển hình có thể kể tới hiện tượng "thu được sóng phát thanh trên một số chuyến bay" mà một số cơ quan báo chí phản ánh.
- Khả năng đảm bảo an toàn dữ liệu thấp, hệ thống truyền thanh không dây FM dễ bị hack, phát những bản tin tuyên truyền tiếng nước ngoài, hay chống phá nhà nước vẫn còn xảy ra. Hiện có rất nhiều nơi trên lãnh thổ Việt Nam có thể bắt được các đài của Trung Quốc. Giải pháp hiện

giờ vẫn là dùng các đài phát có công suất phát lớn hơn phát sóng cùng tần số, "đè" lên sóng phát thanh Trung Quốc.

b) Nhu cầu của các tỉnh xây dựng hệ thống truyền thanh không dây. Từ những số liệu thu thập được, chúng ta rút ra một số kết luận sau:

- Nhu cầu của Phát thanh tại các địa phương nói chung hiện nay là rất lớn, nội dung chương trình yêu cầu phải đa dạng, hấp dẫn.
- Nhiều khu vực mật độ dân cư thưa thớt, trình độ dân trí thấp, hệ thống thông tin liên lạc, phát thanh được đầu tư hạn chế, chưa đáp ứng được nhu cầu. chuyển tải thông tin của chính quyền.
- Hệ thống truyền thanh cơ sở hiện nay sử dụng chủ yếu là truyền thanh qua sóng FM, không đảm bảo chất lượng tín hiệu khi truyền tới các khu vực vùng sâu, vùng xa, vùng trũng phát sóng.

Do đó cần xây dựng hệ thống truyền thông radio số đồng nhất 03 cấp (tỉnh, huyện, xã) phục vụ phổ biến kiến thức thông tin kinh tế, văn hóa, xã hội, an ninh, quốc phòng và các thông tin khẩn cấp, đảm bảo an toàn thông tin và tính dự phòng hệ thống trong trường hợp khẩn cấp.

2.4. Vai trò của hệ thống IoT trong việc quản lí hệ thống truyền thanh không dây

Trong quá trình xây dựng hệ thống truyền thanh không dấy đồng nhất 03 cấp, việc quan trọng nhất là kiểm soát quá trình hoạt động của các trạm thu phát sóng cấp dưới. Do đó việc xây dựng một hệ thống phần mềm có thể theo dõi hoạt động của hệ thống các trạm thu phát Radio Internet theo thời gian thực là vô cùng quan trọng, và ở đây công nghệ IoT sẽ được sử dụng.

Công nghệ IoT cho phép máy chủ qua Internet có thể thu thập dữ liệu được gửi lên từ các máy Internet Radio từ xa. Với mô hình hệ thống truyền thanh không dây đồng nhất 03 cấp như trong hình 2.2, các máy Internet Radio, ngoài việc thu sóng từ server phát sóng, có thể gửi dữ liệu ngược lại từ cấp cơ sở lên. Dữ liệu đó có thể là dữ liệu các cảm biến (nhiệt độ, độ ẩm trong phòng), hay tình trạng hoạt động của máy (đang hoạt động, tắt hay tạm nghỉ,…). Ngoài ra còn có thể tích hợp

thêm các nút như cảnh báo nguy hiểm, báo cháy hay thiên tai lũ lụt. Với các chức năng cảnh báo, thay vì người dùng phải gọi điện thì chỉ cần nhấn nút là tín hiệu cảnh báo sẽ được gửi thẳng lên hệ thống.

Hệ thống cảnh báo khẩn cấp trong hệ thống truyền thanh không dây đồng nhất 03 cấp được sử dụng trong 02 trường hợp chính:

- Khi cấp cao nhất muốn đưa các thông báo khẩn cấp xuống các cấp thấp hơn (Tỉnh đưa thông báo đến huyện, xã), khi phát sinh tình huống khẩn cấp về an ninh quốc phòng, thiên tai, bão lũ. Hệ thống cảnh báo này hoạt động qua loa hoặc qua các thiết bị chuyên dụng tại trạm phát thanh để cảnh báo nhân viên quản lý trạm phát thanh. Cơ chế phát tin bài khẩn cấp ở đây vẫn là phát tin bài như bình thường nhưng có quyền ghi đè lên tin bài đang được phát hiện nay. Trong trường hợp vấn đề khẩn cấp không được thông báo qua tin bài, hệ thống có thể bật các đèn cảnh báo, tín hiệu cảnh báo tự động. Quá trình này được thực hiện tự động nhờ áp dụng công nghệ IoT.
- Tin khẩn cấp được cảnh báo từ cấp cơ sở lên trên hệ thống. Trong các tình huống khẩn cấp cần thông báo như lũ quét, trộm cướp… mà phía cơ sở cần cảnh báo lên cấp cao hơn, có thể dùng điện thoại di động như một lựa chọn. Trong tình huống không có sóng hoặc không thể liên lạc qua di động, việc có một hệ thống cảnh báo khác tích hợp vào hệ thống truyền thanh không dây là một giải pháp tốt. Ở đây, một số nút nhấn và mic có thể được tích hợp để gửi tín hiệu cảnh báo kèm với ghi âm nếu cần.

Phát thông điệp cảnh báo khẩn cấp là một dịch vụ quan trọng mà các đài truyền hình cung cấp cho cộng đồng mà họ phục vụ. Bằng cách cung cấp thông tin kịp thời và chính xác liên quan đến các sự kiện khẩn cấp như cháy rừng, lũ lụt, lốc xoáy, đóng cửa trường học và cảnh báo Amber, các đài truyền hình đóng một vai trò quan trọng trong việc giúp ngăn ngừa thương tích, tử vong và thiệt hại tài sản.

2.5. Kết chương

Trong chương này, em đã nghiên cứu tìm hiểu về hệ thống truyền thanh không dây, từ đó xây dựng được mô hình hệ thống truyền thanh không dây, đồng

nhất 03 cấp. Tiếp theo đó, với kiến thức về hệ thống IoT, em đề xuất xây dựng hệ thống IoT cho phép kết nối giữa máy chủ điều khiển hệ thống truyền thanh không dây, với các máy Internet Radio theo mô hình hệ thống IoT – tức là trao đổi dữ liệu 2 chiều, phục vụ mục đích theo dõi, kiểm tra tình trạng hoạt động của các trạm phát thanh cấp cơ sở và cảnh báo trong các tình huống khẩn cấp.

Trong chương tiếp theo, em sẽ mô tả quá trình xây dựng hệ thống IoT, tập trung vào hệ thống phần mềm, có mô phỏng trên phần cứng.

CHƯƠNG 3: XÂY DỰNG HỆ THỐNG IOT QUẢN LÝ HOẠT ĐỘNG CỦA CÁC TRẠM PHÁT THANH

Một trong những thứ quan trọng nhất trong IoT system là IoT platform – nền tảng phần mềm theo dõi, quản lý và điều khiển hệ thống.

Hệ thống phần mềm hoạt động trên một máy chủ dựa trên các công nghệ để điều khiển để xử lý dữ liệu thiết bị, theo dõi, giám sát, điều khiển từ xa. Máy chủ có thể được cài đặt trên máy cục bộ hoặc chạy trên đám mây và được xây dựng trên các công nghệ được thiết kế để mở rộng tới hàng tỷ sự kiện thiết bị được xử lý mỗi ngày. Hệ thống đảm bảo sự liên tục của dữ liệu được gửi từ thiết bị, đảm bảo lưu trữ dữ liệu an toàn và dữ liệu không bao giờ bị xóa, bất kể dung lượng hay thời gian có lớn đi nữa. Hệ thống cung cấp giao diện dịch vụ cho phép các bên thứ ba mở rộng và tùy chỉnh hệ thống để làm việc với các công nghệ mới và cung cấp hệ thống truyền thông giữa các thiết bị tiên tiến cho phép kiểm soát toàn bộ vòng đời của các thiết bị trong toàn bộ hệ thống IoT. Hệ thống cho phép sử dụng các giao thức truyền thông mới và cung cấp hệ quản trị HTML5 cho phép tất cả dữ liệu hệ thống được xem và thao tác theo cách đơn giản nhất. Trong các mục sau, chúng ta sẽ nghiên cứu và xây dựng hoàn thiện hệ thống IoT quản lý các trạm phát thanh.

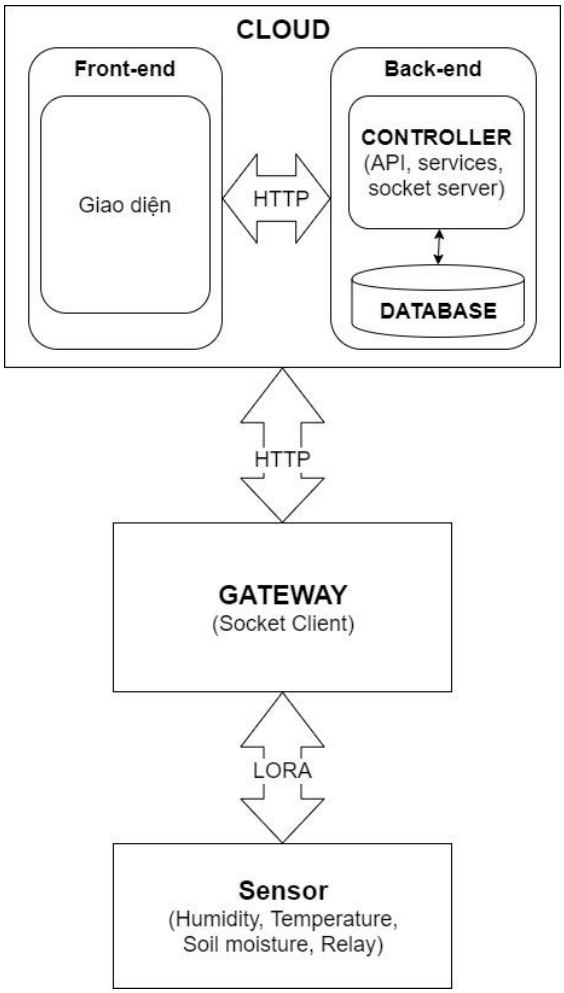
3.1. Mô hình hệ thống và phương thức trao đổi dữ liệu trong hệ thống IoT

Trong mô hình hệ thống IoT đơn giản, việc phân cấp thiết bị gồm 2 lớp: Lớp máy chủ và Lớp IoT gateway, phù hợp cho mô hình các trạm phát thanh, khi dữ liệu trao đổi trực tiếp từ máy thu Internet Radio lên máy chủ quản lý. IoT Gateway chính là Internet Radio, được tích hợp các module phần mềm đọc dữ liệu từ các cảm biến (dữ liệu đây có thể là nhiệt độ, độ ẩm, tình trạng hoạt động của máy) hay đọc trạng thái các nút nhấn cảnh báo khẩn cấp. Mô hình đơn giản của hệ thống sẽ được thấy như trong hình 3.1 dưới đây.

Khối IoT Gateway: đọc dữ liệu từ sensors nhiệt độ độ ẩm, trạng thái hoạt động… rồi chuyển dữ liệu lên khối Cloud Backend trên server thông qua mạng Internet hoặc mạng WAN nội bộ.

Khối Cloud Backend: thực hiện lưu trữ giá trị sensor nhận được từ khối IoT Gateway và chuyển tiếp dữ liệu lên Cloud Frontend.

Khối Cloud Frontend: Thực hiện hiển thị giá trị các sensor và điều khiển nếu cần. Giao diện phải thiết kế đẹp, đáp ứng được yêu cầu.



Hình 3.1: Mô hình hệ thống IoT

3.1.2. Trao đổi dữ liệu với Socketio

Socketio là một phương thức truyền dữ liệu giúp xây dựng một ứng dụng realtime. Socketio sẽ giúp các bên ở những địa điểm khác nhau kết nối với nhau, truyền dữ liệu ngay lập tức thông qua server trung gian. Socketio có thể được sử

dụng trong nhiều ứng dụng như chat, game online, cập nhật kết quả của một trận đấu đang xảy ra...

Cấu trúc một ứng dụng realtime sử dụng Socketio bao gồm 2 phần: phía server, phía client.

- Phía server (máy chủ quản lý hệ thống phát thanh): Đây là nơi sẽ cài đặt Socketio. Ngôn ngữ để dựng server là NodeJS
- Phía client: Ở phía client – máy thu Internet radio sẽ xây dựng giao diện người dùng. Ngôn ngữ để dựng server là ReacJS

Cơ chế hoạt động của Socketio

Khai báo sử dụng Socketio:

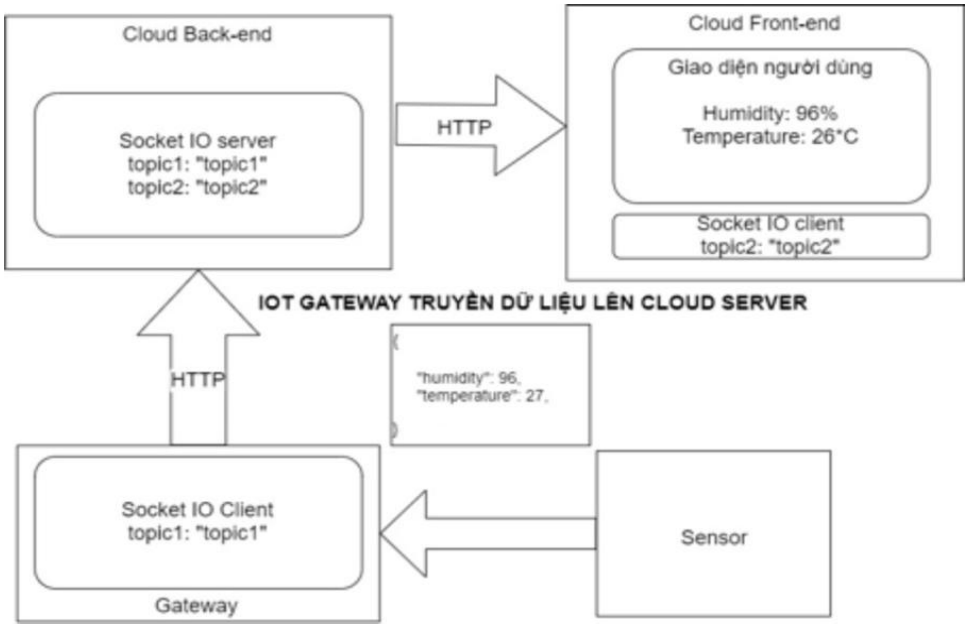
Cơ chế hoạt động của một ứng dụng realtime đó là thông qua server để lắng nghe (listen) data và truyền data về các máy client. Vì vậy cần cài khai báo sử dụng socketio ở cả phía server và client.

Cơ chế lắng nghe, truyền dữ liệu của Socketio:

Để lắng nghe data, ta sử dụng câu lệnh socket.on(), để phát dữ liệu thì sử dụng lệnh socket.emit().

Ví dụ, client gửi 1 đoạn chat đi, thì khi đó ở phía server cần viết code để nhận dữ liệu đoạn code đó và truyền dữ liệu chat đó đi đến các server khác. Đồng thời ở ở phía client cũng cần viết code để gửi và nhận dữ liệu từ server.

Quy trình truyền dữ liệu có thể được thấy trong hình 3.2 dưới đây:



Hình 3.2: Mô hình truyền tải dữ liệu trong hệ thống truyền thanh không dây

Dữ liệu sẽ được truyền vào Internet Radio (IoT Gateway), có thể thông qua một topic nào đó, sau đó dữ liệu được truyền lên Cloud Backend cũng thông qua topic của SocketIO. Cuối cùng dữ liệu sẽ được hiển thị trên Frontend cho người giám sát. Tương tự như vậy cho chiều dữ liệu ngược lại từ server xuống máy thu Internet Radio.

3.2. Nghiên cứu, xây dựng giao diện phần mềm quản lý, giám sát và cảnh báo

3.2.1. Lựa chọn ngôn ngữ xây dựng frontend với ReactJS

React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code javasscript trong code html thông qua các attribute như ng-model, ng-repeat...thì với react là một library cho phép nhúng code html trong code javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong JS.Tích hợp giữa javascript và HTML vào trong JSX làm cho các component dễ hiểu.

DOM (Document Object Model) là một thư viện giao diện người dùng giống như các phần tử, thuộc tính. DOM sẽ được tạo ra khi trang web vừa tải xong DOM và tồn tại dưới dạng tree nodes được dùng để quản lí, truy xuất, chỉnh sửa tới bất kì phần tử nào thông qua đối tượng gốc là document. Bất cứ khi nào phương thức setState () được gọi, ReactJS reset DOM ảo từ đầu. Việc tạo lại rất nhanh nên nó không ảnh hưởng đến hiệu suất. Tại bất kỳ thời điểm nào, ReactJS duy trì hai DOM ảo, một với DOM được cập nhật trạng thái ảo và một với DOM DOM trạng thái trước đó gần đây nhấ để so sánh và cập nhật phần tử thay đổi.

Giới thiệu về Components:

- Là một đoạn mã ngắn có ý nghĩa thể hiện một chức năng hay đối tượng nhất định, có thể tái sử dụng
- Chúng ta có thể sử dụng nhiều mã JSX trong một component
- Một ứng dụng React có thể có hàng chục, hoặc hàng trăm các component liên kết, tương tác với nhau
- Ví dụ về component khi xây dựng website " quản lí, theo dõi, giám sát nông trại thông minh": trang Dashboard bao gồm biểu đồ, bảng, bản đồ,... Mỗi phần của trang Dashboard là một component con, là Dashboard là một component cha.

Vậy để gọi các component con vào component cha hay nói cách khác là để hiển thị lên giao diện của 1 trang Dashboard chỉ cần import component con vào component cha. Một ứng dụng React có thể có hàng chục, hoặc hàng trăm các component liên kết, tương tác với nhau. Ví dụ về component khi xây dựng website "Quản lí, theo dõi các thông số của Internet Radio": trang Dashboard bao gồm biểu đồ, bảng, bản đồ,... Mỗi phần của trang Dashboard là một component con và Dashboard là một component cha. Vậy để gọi các component con vào component cha hay nói cách khác là để hiển thị lên giao diện của 1 trang Dashboard chỉ cần import component con vào component cha.

```
// Component Chart
import React from “react”;
export class Chart extends React.Component {
      render() {
            return(
                  {Code for Chart}
            )
      }
}
```

- Sau đó ta chỉ cần import component Chart này vào component Dashboard :

```
import React from ‘react’;
import ReactDOM from ‘react-dom’;
import { Chart} from ‘./Chart.js’;
class Dashboard extends React.Component {
       render() {
            return (
                  <div><Chart/></div>
           );
      }
}
```

Hai dạng data cơ bản của ReactJS:

ReactJS có 2 dạng data cơ bản là: props và state.

Props (Viết tắt của properties): Props là cách mà component có thể nhận được các giá trị của thuộc tính truyền từ bên ngoài vào. Các component có thể trao đổi dữ liệu với nhau thông qua props. Props sẽ được truyền từ component cha xuống các component con. Props có thể là 1 object, function, string, number … .

Khi một props được truyền vào component thì nó là bất biến (nó không thay đổi giá trị trừ khi dữ liệu gốc của nó thay đổi). Props có thể sử dụng trong function component và cả class component.

Ví dụ: Sử dụng props trong function component: Trong một function component các props có sẵn bằng cách thêm tham số props làm tham số của function.

```
function DemoProps(props) {
  return (
    <div>
     <h1>{props.title}</h1>
     <p>{props.description}</p>
    </div>
  )
}
export default DemoProps
```

Sử dụng props trong class component: Trong một component class có thể nhận được props bằng this.props trong component.

```
import React, { Component } from 'react';
class DemoProps extends Component {
   render() {
      return (
         <div>
            <h1>{this.props.title}</h1>
            <p>{this.props.description}</p>
         </div>
      );
   }
}
export default DemoProps;
```

* Truyền props cho component: Để truyền props cho component thì khi khởi tạo một component, đặt các props theo cách tương tự như đặt các thuộc tính trong HTML. Ở ví dụ này, title được truyền vào dưới dạng một chuỗi đơn giản và description được truyền vào dưới dạng một biến.

```
<DemoProps title="Đây là tiêu đề  bài viết" description={this.state.desc}>
```

Trong props có một thuộc tính đặc biệt là children. Nó chứa giá trị của bất cứ thứ gì được truyền vào trong phần nội dung của component, this.props.children sẽ trả về mọi thứ nằm giữa các thẻ JSX mở và đóng của component.

State là một đối tượng Javascript lưu trữ dữ liệu động của một component.

State là dữ liệu động, nó cho phép một component theo dõi thông tin thay đổi ở giữa các kết xuất (render) và làm cho nó có thể tương tác. State chỉ có thể được sử dụng ở trong một component sinh ra nó, các component khác không thể dử dụng cũng như thay đổi được. Nếu dự đoán được một component sẽ cần quản lý state, thì nó nên được tạo ở trong class component chứ không phải trong functions component.

Để khai báo một state, trong hàm khởi tạo constructor() ta sử dụng cú pháp:

```
this.state = {name_of_object: value}
```

Để lấy ra dữ liệu mà chúng ta khai báo trong state sử dụng cú pháp:

```
this.state.name_of_object
```

State được bắt đầu bằng cách sử dụng this.state, tuy nhiên tất cả các thay đổi tiếp theo đối với state được thực hiện bằng cách sử dụng this.setState. Việc sử dụng this.setState đảm bảo rằng các component bị ảnh hưởng bởi state được hiện thị lại trong trình duyệt.

Lí do lựa chọn ReactJS:

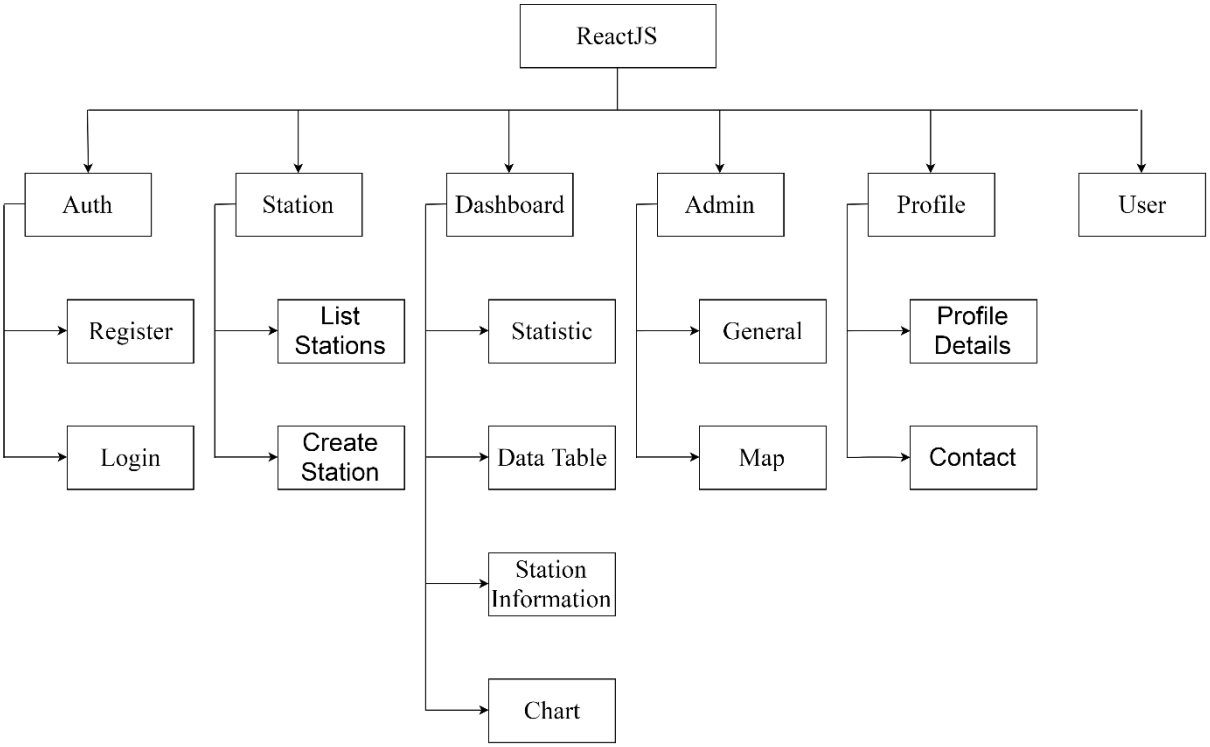
- Trong lĩnh vực phát triển công nghệ, các chủ doanh nghiệp và developer luôn tìm kiếm những phương pháp tốt nhất để giúp doanh nghiệp của họ có những

lợi thế cạnh tranh tốt hơn. Và một trong những công nghệ tốt nhất có thể giúp những doanh nghiệp vượt mặt đối thủ trong việc tạo ra những ứng dụng web chính là ReactJS.

- ReactJS cho phép các doanh nghiệp tạo ra những ứng dụng web với UI tốt hơn để nâng cao trải nghiệm người dùng. Đây cũng chính là công nghệ mà các doanh nghiệp cần để có được lượng tương tác của người dùng, tỉ lệ click cũng như chuyển đổi cao hơn. Hơn thế, các doanh nghiệp sử dụng ReactJS được đảm bảo có giao diện tốt hơn so với những doanh nghiệp sử dụng các framework khác bởi ReactJS giúp ngăn chặn việc cập nhật của DOM giúp ứng dụng nhanh hơn và truyền tải tốt hơn UX.

3.2.2. Xây dựng giao diện phần mềm quản lý các máy phát sóng

Sơ đồ frontend của IoT platform



Hình 3.3: Sơ đồ frontend của IoT platform quản lý Internet Radio

Frontend của IoT platform quản lý Internet Radio được xây dựng bằng ReactJS và chia thành các trang chính:

Trang Auth: Thực hiện chức năng đăng ký, đăng nhập tài khoản, cho phép người dùng tạo một tài khoản mới và đăng nhập vào trang quản lý.

Trang Station: Liệt kê danh sách các Internet Radio mà người dùng tham gia theo dõi, quản lý. Cho phép admin tạo ra các Internet Radio mới.

Trang Dashboard: Thống kê các thông số của Internet Radio theo thời gian thực; tổng hợp các thông số dưới dạng bảng theo thời gian; đưa ra các thông tin của Internet Radio cũng như bản đồ và giá trị cảnh báo; dựa vào số liệu vẽ đồ thị. Đặc biệt, cho phép xuất ra file để dễ dàng quản lý.

Trang Admin: Trang dành riêng cho admin, cho phép admin chỉnh sửa các thông tin của Internet Radio hoặc xóa một trạm bất kỳ.

Trang Profile: Cung cấp các thông tin chi tiết của người dùng, cho phép người dùng chỉnh sửa các thông tin của mình. Đồng thời cung cấp các thông tin liên lạc của các user khác đang tham gia quản lý Internet Radio.

Trang Users: Liệt kê các users đã đăng ký tài khoản. Admin được quyền thêm user vào quản lý các Internet Radio hoặc xóa user đó ra khỏi hệ thống.

Sử dụng ReactJS xây dựng giao diện, chi tiết như dưới đây:

Trang Auth: Tạo giao diện đăng nhập cho người dùng. Khi người dùng submit sẽ gửi thông tin về cho backend để backend xử lý.

```
class SignIn extends React.Component {
  render() {
    return (
      <React.Fragment>
        <Card>
          <CardBody>
            <h1 className="text-center">ELECTRIC</h1>
            <Form onSubmit={this.handleSubmit}>
              <FormGroup>
                <Label className="text-primary">Email</Label>
                <Input />
```

```
                    </FormGroup>
                    <FormGroup>
                        <Label className="text-primary ">Password</Label>
                        <Input />
                    </FormGroup>
                </Form>
                <div className="text-center mt-2">
                    <NavLink to=”/auth/sign-up”>Do not have an account. Signup?</NavLink>
                </div>
            </CardBody>
        </Card>
    </React.Fragment>
  ); }
}
```

Chi tiết xem tại phụ lục 1:

- Funtion SignUp: Tạo giao diện đăng ký cho người dùng, khi người dùng submit sẽ gửi thông tin đăng ký về cho backend để xử lý.

```
<Form onSubmit={this.handleSubmit}>
    <FormGroup>
        <Label  className="text-primary">Email</Label>
        <Input />
    </FormGroup>
    <FormGroup>
        <Label  className="text-primary">Full Name</Label>
        <Input />
    </FormGroup>
    <FormGroup>
        <Label  className="text-primary ">Password</Label>
```

```
            <Input />
        </FormGroup>
        <FormGroup>
            <Label className="text-primary">Confirm Password</Label>
            <Input />
        </FormGroup>
        <div className="text-center mt-3">
            <Button>
                Sign up
            </Button>
        </div>
</Form>
```

Chi tiết xem tại Phụ lục 1.

Trang Station

- Function Stations: Liệt kê tất cả các Internet Radio mà user đang quản lý, khi chọn một trạm thì chuyển đến trang quản lý dữ liệu của trạm đó.

```
{this.state.data.map(({ id, manager, machine, photo, power, sub_id, address, name, phone_number }, index) => {
    if (ValidInput.isEmpty(this.state.keyWord)) {
        return (
            <TableStation
                key={index}
                id={id}
                index={index + 1}
                is_admin={isAdmin}
                manager={manager}
                machine={machine}
                sub_id={sub_id}
                photo={photo}
```

```
                        power={power}
                        address={address}
                        name={name}
                        phone_number={phone_number}
                    />
                );
            } else {
                if (manager.indexOf(this.state.keyWord) !== -1) {
                    return (
                        <TableStation
                            key={index}
                            id={id}
                            index={index + 1}
                            is_admin={isAdmin}
                            manager={manager}
                            machine={machine}
                            sub_id={sub_id}
                            photo={photo}
                            power={power}
                            address={address}
                        />
                    );
                }
            }
})
```

Chi tiết xem tại phụ lục 1.

Trang Dashboard

- Function Dashboard: Tạo giao diện quản lý các thông số của Internet Radio, cung cấp các thông tin cần thiết của trạm, vẽ đồ thị và cho phép xuất ra file. Dưới đây là code hiển thị dữ liệu của điện áp các pha.

```
<Col sm="3">
  <Card className="flex-fill">
    <CardHeader>
      <span className="badge badge-primary float-right">V</span>
      <h5 className="card-title mb-0">Voltage</h5>
    </CardHeader>
    <CardBody>
      <Media>
        <div className="d-inline-block mr-3">
          <h3 className="font-weight-light ">
          <Zap className="feather-md text-primary mb-1 mr-2 " color="lightblue" />
            UA
          </h3>
        </div>
        <Media body>
          <h3 className={"mb-1 text-center " + convertToClassName(data.UA, "UA")}>{data.UA}</h3>
      </Media>
      </Media>
      <Media>
        <div className="d-inline-block mr-3">
          <h3 className="font-weight-light ">
          <Zap className="feather-md text-warning mb-1 mr-2" color="lightblue" />
            UB
```

```
            </h3>
          </div>
          <Media body>
              <h3 className={"mb-1 text-center text-center " +
convertToClassName(data.UB, "UB")}>{data.UB}</h3>
          </Media>
        </Media>
        <Media>
          <div className="d-inline-block mr-3">
            <h3 className="font-weight-light ">
            <Zap className="feather-md text-danger mb-1 mr-2"
color="lightblue" />
              UC
            </h3>
          </div>
          <Media body>
            <h3 className={"mb-1 text-center " + convertToClassName(data.UC,
"UC")}>{data.UC}</h3>
           </Media>
        </Media>
      </CardBody>
   </Card>
</Col>
```

Chi tiết xem tại phụ lục 1.

Trang Admin

- Function Admin: Tạo giao diện trang quản lý Internet Radio cho admin. Dưới đây là đoạn code hiển thị nhiệt độ môi trường và nhiệt độ dầu trong máy

```
<Row>
  <Col>
    <FormGroup xs="6">
      <Label for="name_of_temp_high">Temp High</Label>
      <Input
        type="text"
          name="temp_high"
          placeholder="°C"
          value={this.state.data.temp_high}
          onChange={this.handleChange}
           autoComplete="off"
         />
      </FormGroup>
  </Col>
  <Col>
      <FormGroup>
            <Label for="name_of_oil_temp_low">Oil Temp High</Label>
                  <Input
                        type="text"
                        name="oil_temp_high"
                        placeholder="°C"
                        value={this.state.data.oil_temp_high}
                        onChange={this.handleChange}
                        autoComplete="off"
                  />
            </FormGroup>
      </Col>
</Row>
```

3.3. Nghiên cứu, xây dựng phần mềm backend trên máy chủ

Sau khi xây dựng giao diện phần mềm (frontend) dựa trên ReactJS như đã mô tả ở mục trên, phần tiếp theo chúng ta sẽ xây dựng phần mềm backend cho máy chủ. Để phần frontend của một website có thể hoạt động, thì backend phải được xây dựng đủ tốt. Phần backend bao gồm một máy chủ, một ứng dụng và một cơ sở dữ liệu. Backend 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. Để khiến cho máy chủ, ứng dụng, và cơ sở dữ liệu có thể giao tiếp được với nhau, các lập trình viên backend sử dụng các ngôn ngữ server-side như PHP, Python, Java, NodeJS… để xây dựng một ứng dụng, và các công cụ như MySQL, Oracle, và SQL Server để tìm kiếm, lưu trữ, hoặc thay đổi dữ liệu và phục vụ trở lại tới người dùng trong phần frontend. Như vậy backend dùng để xử lý mọi logic nghiệp vụ phức tạp ẩn ở phía sau, giúp cho hệ thống hoạt động hiệu quả hơn.

3.3.1. RESTful API

API là viết tắt của Application Programming Interface (giao diện lập trình ứng dụng) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. Mỗi bộ API dành cho từng đối tượng đều có sự khác biệt nhất định. Thực tế, khi sử dụng bất kì phần mềm hay ứng dụng nào, những trải nghiệm mà nó mang đến không đơn thuần chỉ dựa vào một phần mềm hay ứng dụng độc lập. Tối thiểu, nó cần đến sự kết hợp giữa phần mềm, ứng dụng với hệ điều hành đề có thể hiển thị và người dùng có thể tương tác. Một phần mềm hay ứng dụng riêng biệt được thiết kế về giao diện, cơ chế hoạt động … nhưng không thể bao gồm tất cả các công việc còn lại như một hệ điều hành. Chính bởi vậy, giống như một sự phân công lao động, từng phần mềm sẽ có trách nhiệm riêng, được thiết kế để đảm bảo khả năng thực hiện công việc đó và API là cầu nối để những phần mềm này kết nối, tận dụng những điều kiện sẵn có của nhau.

Hiện nay, API có các tính năng như:

- API tuân thủ các tiêu chuẩn (phổ biến nhất là HTTP và REST), dễ sử dụng, thuận tiện trong truy cập và dễ hiểu.
- API được thiết kế hướng đến từng đối tượng cụ thể và được xử lý giống như các sản phẩm thay vì những mã code.
- Để đảm bảo tính bảo mật và giúp việc quản lý nghiêm ngặt, thuận tiện trên phương diện theo dõi hiệu quả, quy mô, API hiện nay được chuẩn hóa nhiều hơn.
- API có vòng đời phát triển như mọi loại sản phẩm phần mềm từ thiết kế, thử nghiệm, quản lý, các phiên bản nâng cấp …

API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML và status code. Mẫu dữ liệu JSON:

```
{
   "data": {
      "id": "1",
      "name": "Nguyen Tieu Chau"
   }
}
```

Status code:

- 1xx: Thông tin
- 2xx: Thành công
- 3xx: Chuyển hướng
- 4xx: Lỗi client
- 5xx: Lỗi server

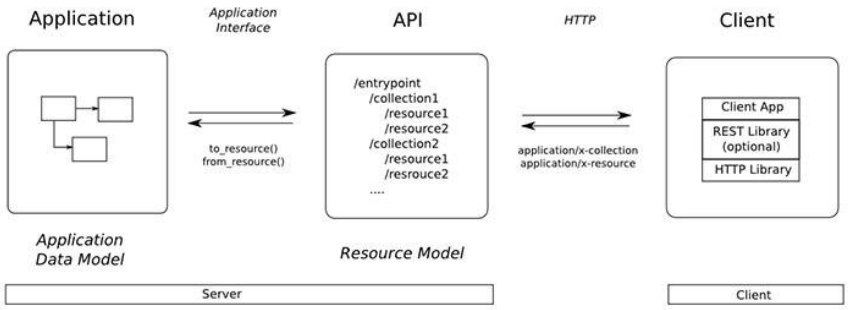
Một số status code thường gặp:

- 200 OK – Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.
- 400 Bad Request – Request không hợp lệ
- 404 Not Found – Không tìm thấy resource từ URL…

REST (Representational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, … đến một URL để xử lý dữ liệu.

RESTful API là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

* Cách thức hoạt động RESTful API:



Hình 3.4: Sơ đồ truyền nhận dữ liệu qua API

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

- GET (SELECT): Trả về một Resource hoặc một danh sách Resource. GET là một hành động an toàn, chỉ đọc và sẽ không làm thay đổi trạng thái của server. Mỗi lần bạn nhập một URL trong trình duyệt của mình, chẳng hạn như http://google.com, bạn đang gửi một yêu cầu GET đến server của Google.
- POST (CREATE): Tạo mới một Resource. POST được sử dụng để tạo một tài nguyên mới. Một ví dụ quen thuộc về POST là đăng ký người dùng trên trang web hoặc ứng dụng. Sau khi gửi form, một yêu cầu

POST với dữ liệu người dùng có thể được gửi đến server, sau đó sẽ ghi thông tin đó vào cơ sở dữ liệu.

- PUT/PATCH (UPDATE): Cập nhật thông tin cho Resource. PUT cập nhật tài nguyên hiện có, có thể được sử dụng để chỉnh sửa các cài đặt của người dùng có sẵn. Không giống như POST, PUT là bình thường, có nghĩa là cùng một call có thể được thực hiện nhiều lần mà không tạo ra kết quả khác. Ví dụ, nếu bạn gửi cùng một yêu cầu POST để tạo người dùng mới trong cơ sở dữ liệu nhiều lần, nó sẽ tạo một người dùng mới có cùng dữ liệu cho mỗi yêu cầu bạn thực hiện. Tuy nhiên, sử dụng cùng một yêu cầu PUT trên cùng một người dùng sẽ liên tục tạo ra kết quả giống nhau.
- DELETE (DELETE): Xoá một Resource, sẽ xóa một tài nguyên hiện có.

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.

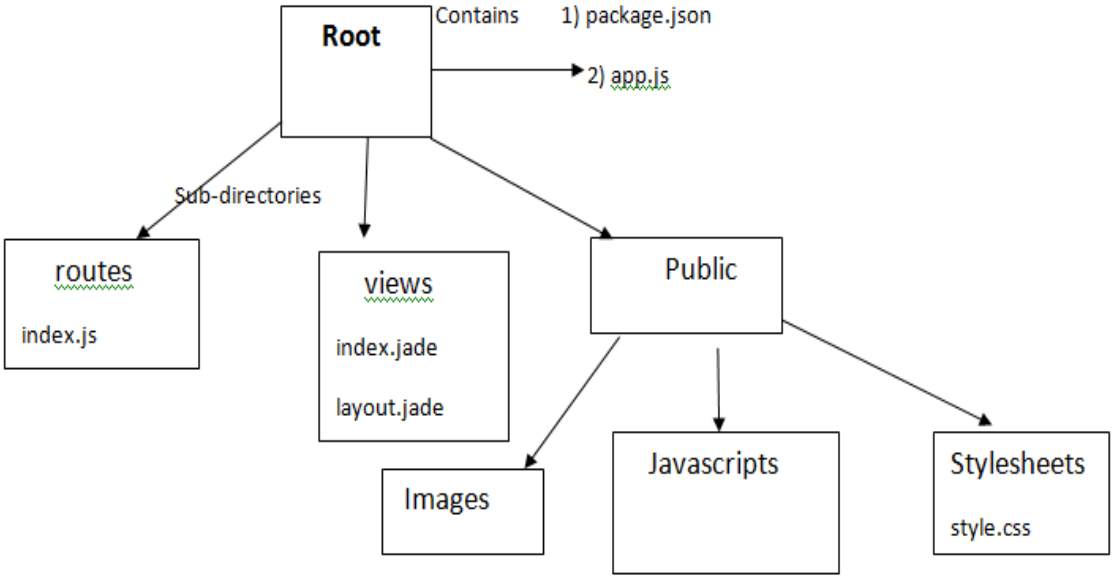
Hiện tại đa số lập trình viên viết RESTful API giờ đây đều chọn JSON làm format chính thức nhưng cũng có nhiều người chọn XML làm format, nói chung dùng thế nào cũng được miễn tiện và nhanh.

3.3.2. ExpressJS

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 middleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.

ExpressJS có một số chức năng chính như sau:

- Thiết lập các lớp trung gian để trả về các HTTP request.
- Define router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL.
- Cho phép trả về các trang HTML dựa vào các tham số.

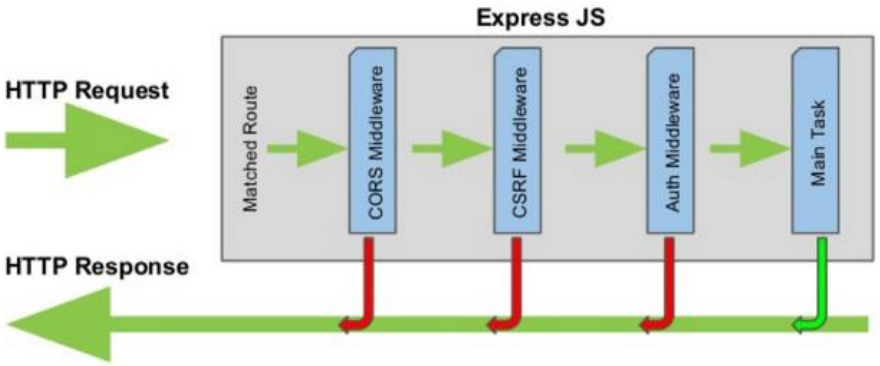


Hình 3.5: Sơ đồ cây thư mục trong ExpressJS

Khi nhận được 1 HTTP request:

- CORS middleware: bảo mật được cài đặt vào toàn bộ các trình duyệt hiện nay. Chính sách này ngăn chặn việc truy cập tài nguyên của các domain khác một cách vô tội vạ.
- CSRF middleware: xác thực CSRF của request, chống fake request.
- Error-handling middleware: phục vụ cho việc xử lý lỗi.
- Auth middleware: Xử lý token từ HTTP request để xác nhận người dùng.
- Validation middleware: Kiểm tra dữ liệu đầu vào (kiểm tra những trường cần thiết và kiểu dữ liệu).
- Permission middleware: Kiểm tra quyền của người dùng có được thực hiện nhiệm vụ này không.
- Main task: Thực hiện xử lý request đó liên quan đến database và xử lý dữ liệu đó rồi response lại cho người dùng.

* Quá trình xử lý một api:



Hình 3.6: Quá trình xử lý một api của backend

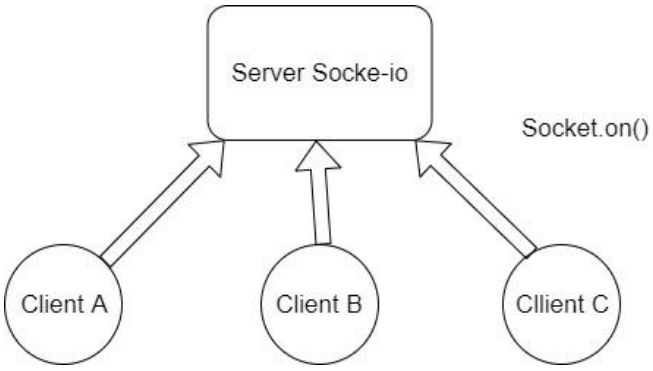
3.3.3. Socket-io

Để xây dựng một ứng dụng realtime cần sử dụng socket-io. Socket-io sẽ giúp các bên ở những địa điểm khác nhau kết nối với nhau, truyền dữ liệu ngay lập tức thông qua server trung gian. Socket-io có thể được sử dụng trong nhiều ứng dụng như chat, game online, cập nhật kết quả của một trận đấu đang xảy ra, ...

Socket-io không phải là một ngôn ngữ, mà chỉ là 1 công cụ giúp thực hiện những ứng dụng realtime. Vì thế, không thể sử dụng socket-io để thay thế hoàn toàn cho một ngôn ngữ, mà phải sử dụng kết hợp với một ngôn ngữ khác. Ngôn ngữ đó có thể là php, asp.net, nodejs, ...

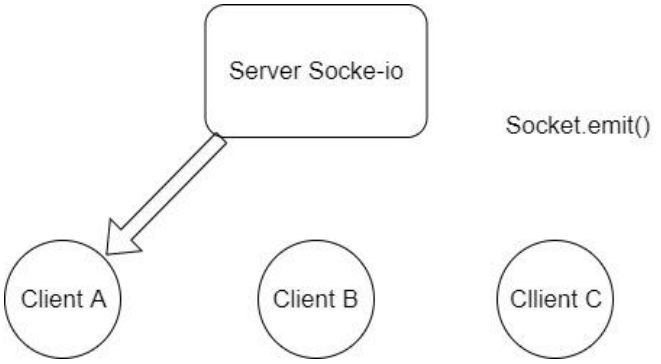
Phương thức hoạt động:

- Cơ chế lắng nghe: sử dụng câu lệnh socket.on()

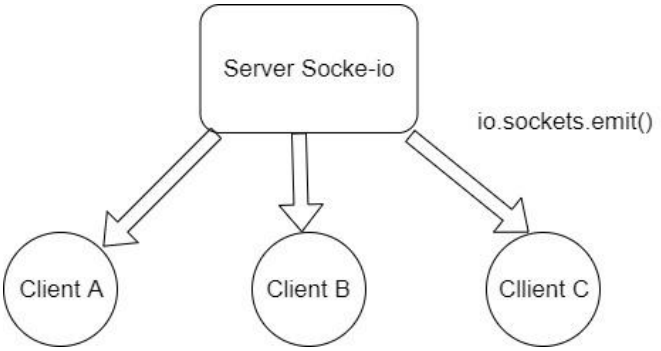


Hình 3.7: Sơ đồ nhận dữ liệu của socket-io server

- Cơ chế truyền data: sử dụng câu lệnh socket.emit() hoặc io.sockets.emit() cho phép truyền dữ liệu đến 1 client xác định hay đến tất cả các client như hình 3.8, 3.9 dưới đây.



Hình 3.8: Sơ đồ truyền dữ liệu từ socket-io server tới một client xác định



Hình 3.9: Sơ đồ truyền dữ liệu từ socket-io server tới tất cả client

3.3.4. MongoDB

MongoDB là một cơ sở dữ liệu mã nguồn mở và là cơ sở dữ liệu NoSQL hàng đầu, được hàng triệu người sử dụng. MongoDB được viết bằng C++, chính vì thế nên nó có khả năng tính toán với tốc độ cao chứ không giống như các hệ quản trị CSDL hiện nay. Ngoài ra, MongoDB là một cơ sở dữ liệu đa nền tảng, hoạt động trên các khái niệm Collection và Document, nó cung cấp hiệu suất cao, tính khả dụng cao và khả năng mở rộng dễ dàng. Cụ thể hơn như sau:

- Database là một vùng chứa vật lý cho Collections. Mỗi Database đều có tập các tệp riêng trên hệ thống tệp. Một máy chủ MongoDB thường có nhiều Database.
- Collection là một nhóm tài liệu MongoDB. Nó tương đương với một mảng RDBMS. Một Collection tồn tại trong một Database duy nhất. Collection không thực thi lược đồ.

- Document là tập hợp các cặp khóa-giá trị. Document có schema động. Schema động có nghĩa là các document trong cùng một collection không cần phải có cùng một tập hợp các trường hoặc cấu trúc và các trường phổ biến trong tài liệu của bộ sưu tập có thể chứa các loại dữ liệu khác nhau.

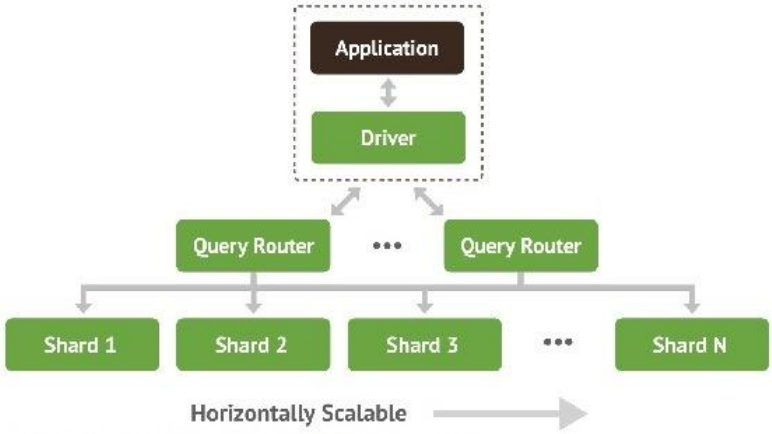
Bảng sau đây cho thấy mối quan hệ của thuật ngữ RDBMS với MongoDB:

Bảng 3.1: Mối quan hệ của thuật ngữ RDBMS với MongoDB

RDBMS	**MongoDB**
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
Mysql/sqlplus	mongo

MongoDB hoạt động dưới một tiến trình ngầm service, luôn mở một cổng (Cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.

❖ Cách thức hoạt động của MongoDB



Hình 3.10: Cấu trúc của cơ sở dữ liệu MongoDB

Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

Một số câu lệnh cơ bản của MongoDB được mô tả trong bảng 3.2 dưới đây:

Bảng 3.2: Một số câu lệnh cơ bản của MongoDB

Câu lệnh	**MongoDB**
Tạo Database	use DATABASE_NAME
Tạo Collection	db.createCollection(name, options)
Insert Document	db.COLLECTION_NAME.insert(document)
Truy vấn Document	db.COLLECTION_NAME.find()
Cập nhật Document	db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
Xóa Database	db.dropDatabase()
Xóa Collection	db.COLLECTION_NAME.drop()
Xóa Document	db.COLLECTION_NAME.remove(DELETION_CRITERIA)

MongoDB có rất nhiều những ưu điểm như:

- Do MongoDB sử dụng 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, nên bạn muốn gì thì cứ insert vào thoải mái.

- Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau, không có join như trong RDBMS nên khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.
- MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster

Tuy nhiên, mongoDB cũng có một số các nhược điểm:

- Một ưu điểm của MongoDB cũng chính là nhược điểm của nó. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.
- Tốn bộ nhớ do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên dễ bị dư thừa dữ liệu.
- Khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng điều này sẽ là nhược điểm vì sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện...

3.3.5. NodeJS và lý do lựa chọn NodeJS

NodeJS là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên JavaScript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng. Phần Core bên dưới của NodeJS được viết hầu hết bằng C++ nên cho tốc độ xử lý và hiệu năng khá cao. NodeJS sử dụng kiến trúc hướng sự kiện event-driven, mô hình non-blocking I/O làm cho nó nhẹ và hiệu quả hơn. NodeJS tạo ra được các ứng dụng có tốc độ xử lý nhanh, realtime thời gian thực. Hệ thống nén của NodeJS, npm, là hệ thống thư viện nguồn mở lớn nhất thế giới. NodejJS áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án Startup nhanh nhất có thể.

Không phải bất kỳ ứng dụng nào cũng nên sử dụng NodeJS, ví dụ như một ứng dụng cần tính ổn định cao hay có logic phức tạp. Tuy nhiên, lại có khá nhiều các ứng dụng cần đến NodeJS bởi các ưu điểm của nó như:

- Fast File Upload Client: Là các chương trình upload file tốc độ cao.
- RESTful API: Node.js tập trung vào nhu cầu web và trên các thiết bị di động. Nó được dùng phổ biến nhất trong lĩnh vực phát triển web. NodeJS cho phép người dùng có khả năng sử dụng công nghệ push thông qua websockets. Điều này là đáng chú ý bởi vì nó cho phép kết nối hai chiều trong thời gian thực. Có nghĩa rằng cả client và server có thể khởi tạo giao tiếp và cho phép tự do trao đổi dữ liệu.
- Real-time Data Application: Bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực. Điều này là rất cần thiết cho hệ thống theo dõi các thông số.
- Micro Services: Ý tưởng của micro services là chia nhỏ một ứng dụng lớn thành các dịch vụ nhỏ và kết nối chúng lại với nhau và Nodejs có thể làm tốt điều này.
- "Javascript có những đặc tính mà làm cho nó rất khác biệt so với các ngôn ngữ lập trình động còn lại, cụ thể là nó không có khái niệm về đa luồng, tất cả là đơn luồng và hướng sự kiện."
- Nodejs chạy đa nền tảng phía Server, sử dụng kiến trúc hướng sự kiện Event-driven, cơ chế non-blocking I/O làm cho nó nhẹ và hiệu quả.
- Các ứng dụng NodeJS đáp ứng có tốc độ xử lý nhanh, hiệu năng khá cao, chạy thời gian thực và chạy đa nền tảng, đa thiết bị.
- Nodejs áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án Startup nhanh nhất có thể

3.3.6. Nghiên cứu xây dựng module backend thu thập, lưu trữ, trao đổi dữ liệu

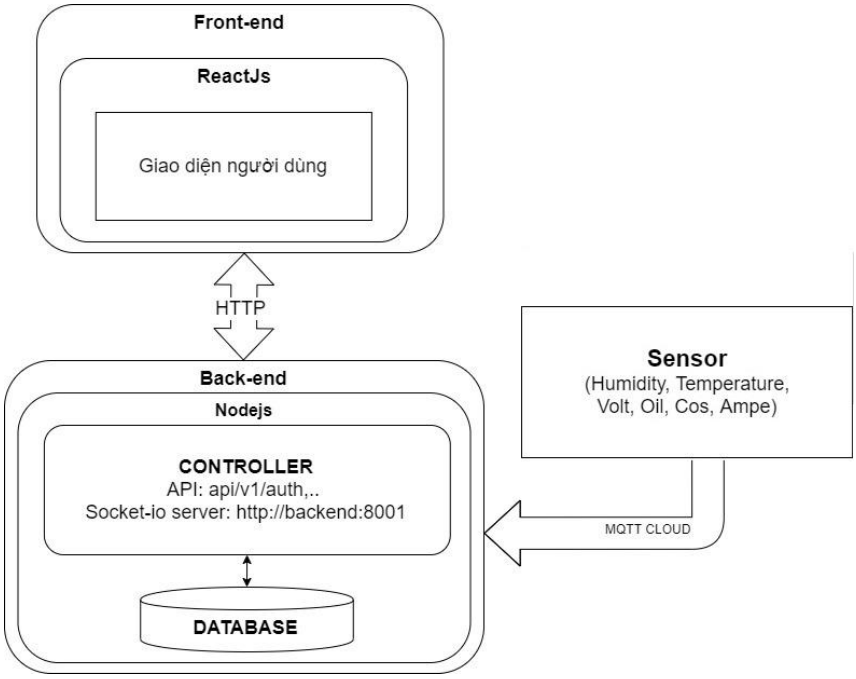
Phần mềm backend của hệ thống máy chủ bao gồm các module Controller và Database, chi tiết được thể hiện trong hình 3.11. Khối Backend được xây dựng bằng NodeJS, thực hiện một số chức năng như:

- Thực hiện kết nối với Sensor box thông qua giao thức MQTT. Sau đó, phân tích dữ liệu nhận được rồi lưu vào database.
- Tạo các API để xử lý các yêu cầu truy cập dữ liệu từ database của user. Các API này sẽ được kiểm tra qua các middleware nhằm xác thực, kiểm tra quyền của user.
- Xây dựng server socket-io.

Khởi tạo server backend

Sử dụng Nodejs và module Express xây dựng server. Cài đặt server với các lệnh như sau:

```
let express = require('express');
let app = express();
let port = normalizePort(process.env.PORT || 8001);
app.set('port', port);
let server = http.createServer(app);
server.listen(port); // Server chạy trên port 8001
server.on('listening', onListening);
function onListening() {
  let addr = server.address();
  let bind = typeof addr === 'string' ? 'pipe ' + addr : 'port ' + addr.port;
  logColor(`Listening on: color:yellow${bind}`);
}
```



Hình 3.11: Sơ đồ khối hệ thống IoT platform

Tạo kết nối server với database MongoDB

```
const mongoose = require('mongoose');
const options = {
   useUnifiedTopology: true,
   useNewUrlParser: true,
   useCreateIndex: true,
   useFindAndModify: false
}
Mongoose.connect("mongodb://localhost/substation", options).then(
   () => {console.log("Successfully connected to mongodb://localhost/substation");},
   err => {console.error( Error("Unable to connect to database") );}
)
```

Tạo file database, thể hiện các trường trong một table. Dưới đây là file database cho table substation_data:

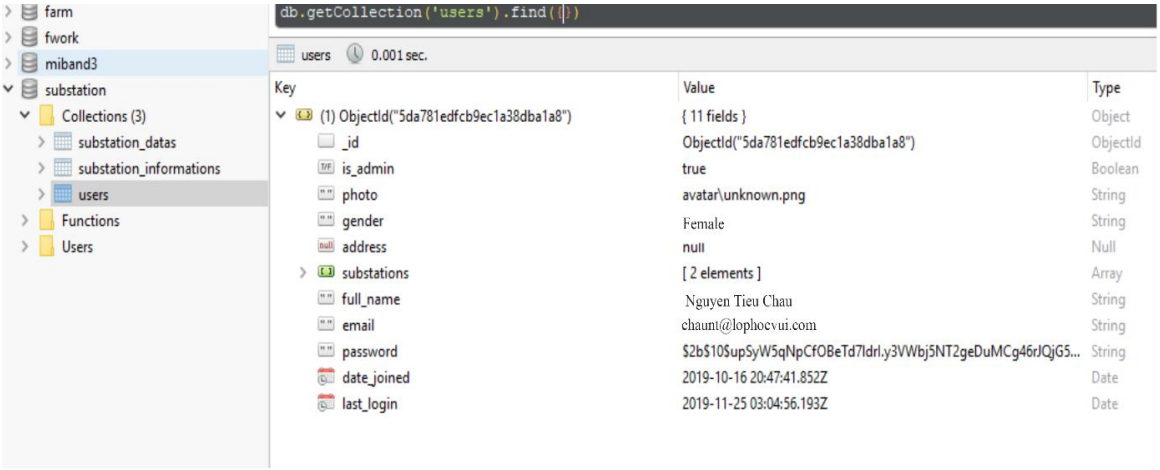
```
const mongoose = require('mongoose');
const SubstationSchema = new mongoose.Schema({
```

```
    sub_id: String,
    Temperature: Number,
    Humidity: Number,
    Light: Number,
    Status: Text,
    time: Date,
    created_date: {
        type: Date,
        default: Date.now()
    }
},{ versionKey: false });
const Data = mongoose.model("substation_data", SubstationSchema);
module.exports = Data;
```

Tiếp theo là bảng dữ liệu quản lý users với các trường thông tin như sau:

- _id: ObjectId
- is_admin: User có là admin hay không
- photo: Ảnh của user
- address: Địa chỉ của user
- substations: Các máy thu Internet Radio
- full_name: Họ và tên của user
- email: email của user
- password: Mật khẩu của user
- date_joined: Ngày user đăng ký tài khoản
- last_login: Ngày đăng nhập gần nhất

Bảng dữ liệu users được mô tả như trong hình 3.12 dưới đây:



Hình 3.12: Bảng dữ liệu của người dùng

Danh sách các APIs được tạo ra như trong bảng 3.3 dưới đây:

Bảng 3.3: Danh sách các APIs của backend hệ thống

Đường dẫn: api/v1/API	**Phương thức**	**Chức năng**
auth	POST	Đăng nhập
auth/register	POST	Đăng ký tài khoản
information	GET	Lấy thông tin của tất cả các máy thu Internet Radio
information/:stationId	GET	Lấy thông tin của một máy thu Internet Radio xác định
information	POST	Tạo một máy thu Internet Radio mới (Quyền admin)
information/:stationId	PATCH	Chỉnh sửa thông tin của một máy thu Internet Radio xác định (Quyền admin)
information/add_substation	POST	Thêm quyền xem cho các user không phải admin (Quyền admin)
information/stationId	DELETE	Xóa một máy thu Internet Radio xác định (Quyền admin)
users	GET	Lấy thông tin của tất cả các users mà

		user được theo dõi
users/me	GET	Lấy thông tin của chính user đang đăng nhập
users/:userId	GET	Lấy thông tin của một user xác định
users/me	PATCH	Chỉnh sửa thông tin của chính user đang đăng nhập
users/:userId	PATCH	Chỉnh sửa thông tin của một user xác định
users/change_avatar	POST	Thay đổi avatar của user
users/change_password	POST	Thay đổi mật khẩu tài khoản của user
data/:substationId	GET	Lấy dữ liệu mới nhất của trạm
notification	GET	Liệt kê các thông báo gửi đến cho user
notification/:notificationId/set-as-read	POST	Set một thông báo ở trạng thái đã đọc
notification/set-as-read	POST	Set tất cả các thông báo ở trạng thái đã đọc

Bằng việc xây dựng hoàn thiện backend và frontend cho toàn bộ hệ thống, IoT platform được tạo ra sẽ có thể đáp ứng được hết các yêu cầu như thu thập dữ liệu, lưu trữ và hiển thị thông tin. Ngoài ra còn có thể điều khiển thiết bị bằng việc thay đổi giá trị các topic của SocketIO.

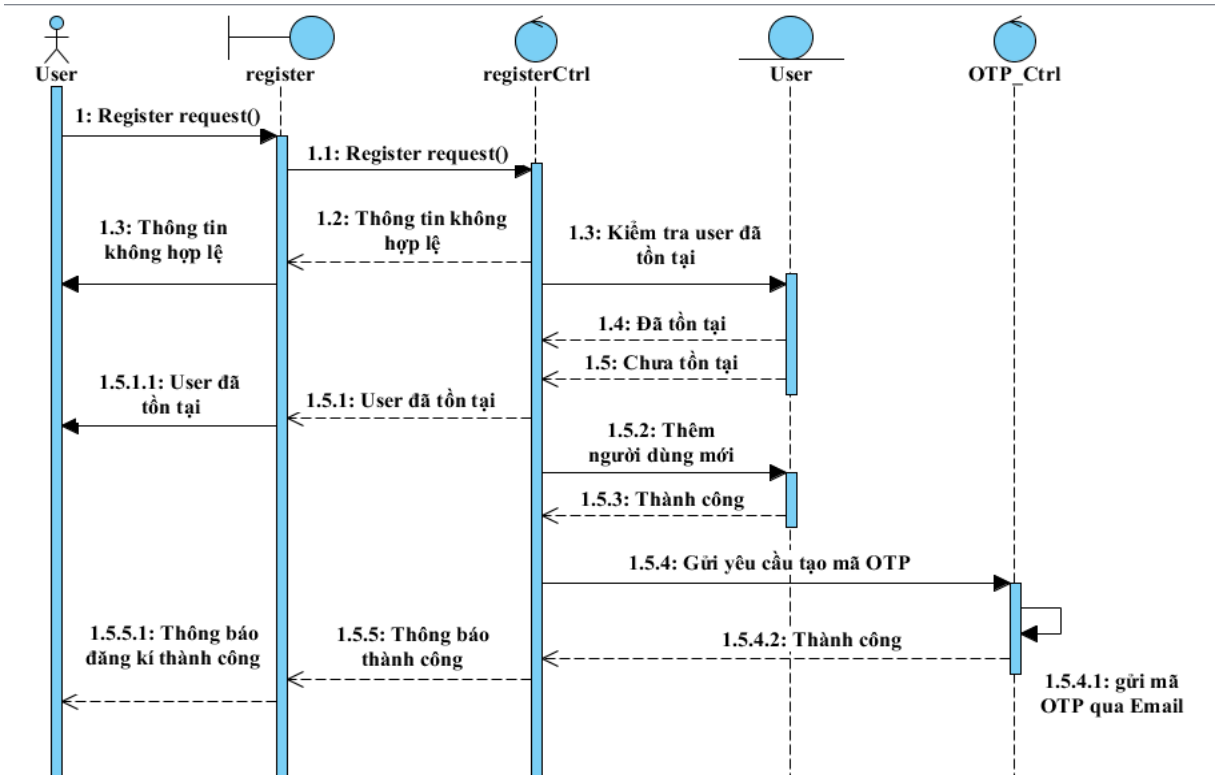
3.4. Nghiên cứu xây dựng module phần mềm xác thực người dùng trong hệ thống

Để các API hoạt động thì một phần quan trọng tạo nên nó đó là các controller. Các controller này nhằm phục vụ, thực hiện các quá trình lưu trữ, lấy dữ liệu từ database và kiểm tra quyền của mỗi user và các trường trả về trong dữ liệu.

- ***API Register: api/v1/auth/register với method POST:***

Đầu tiên user yêu cầu đăng ký tài khoản trong trang đăng ký. Sau đó yêu cầu đăng ký được đưa đến registerCtrl, registerCtrl kiểm tra thông tin đăng ký tài khoản nếu thông tin đăng ký không hợp lệ nó sẽ báo về cho user là thông tin không hợp lệ.

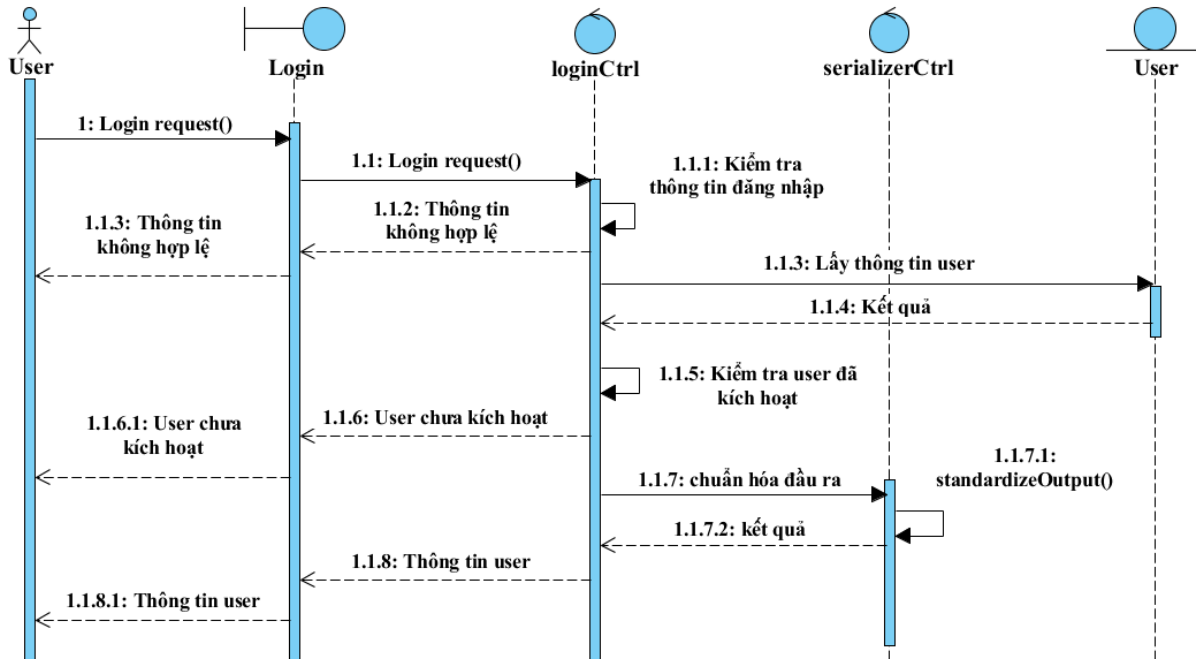
Sau đó, tiếp tục kiểm tra thông tin user đã tồn tại hay chưa, nếu user đã tồn tại trong cơ sở dữ liệu sẽ báo về là user đã tồn tại. Nếu thông tin đăng ký hợp lệ và user chưa tồn tại trong cơ sở dữ liệu nó sẽ thêm người dùng mới vào cơ sở dữ liệu và báo thành công.



Hình 3.13: Quy trình tuần tự phần đăng ký tài khoản

Sau registerCtrl thêm người dùng mới vào cơ sở dữ liệu và báo về thành công nó sẽ gửi yêu cầu tạo mã OTP đã tới OTP_Ctrl, sau đó OTP_Ctrl gửi mã OTP qua email.

- API Login: api/v1/auth với method POST:



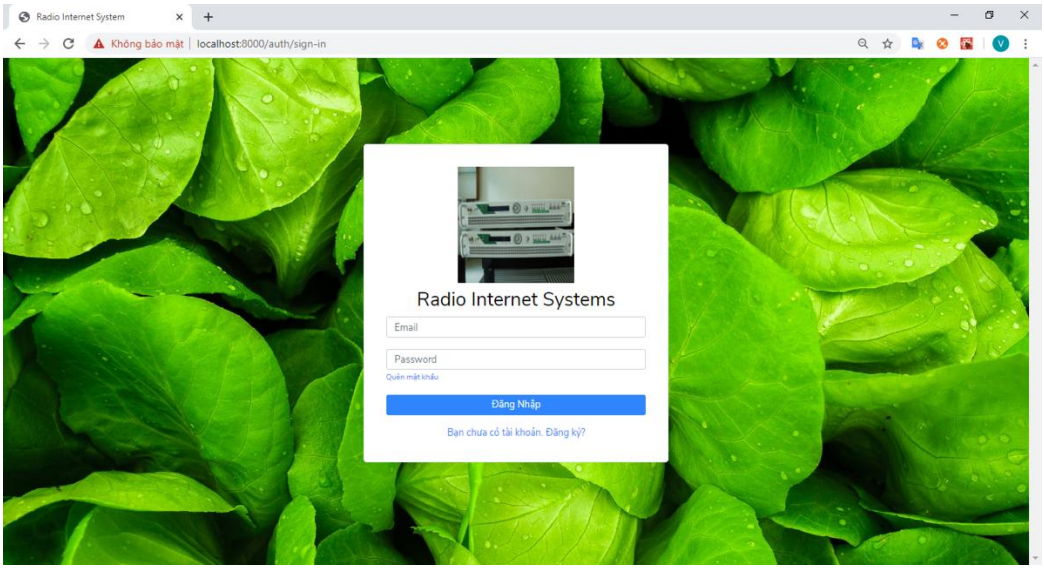
Hình 3.14: Quy trình tuần tự phần đăng nhập tài khoản

Đầu tiên user đăng nhập tài khoản trong trang Login. Sau đó, yêu cầu đăng nhập được gửi đến loginCtrl, loginCtrl kiểm tra thông tin đăng nhập nếu thông tin đăng nhập không hợp lệ nó sẽ báo về cho user là thông tin không hợp lệ. Nếu thông tin hợp lệ thì tiếp tục kiểm tra xem user đã được kích hoạt hay chưa, nếu chưa kích hoạt thì báo về là user chưa kích hoạt. Nếu user đã được kích hoạt thì thông tin được chuẩn hóa và cho phép user đăng nhập.

CHƯƠNG 4: KIỂM THỬ IOT PLATFORM VỚI PHẦN CỨNG MÔ PHỎNG MÁY THU INTERNET RADIO

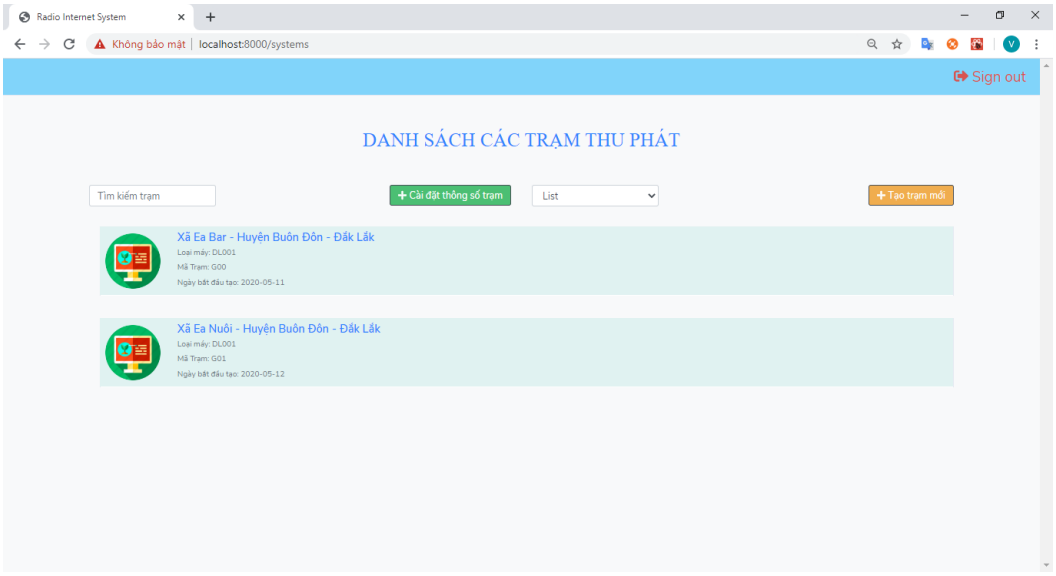
4.1. Kiểm thử giao diện phần mềm

Đầu tiên là đăng ký và đăng nhập vào hệ thống. Tính năng đăng ký đã bị disable, nên chỉ có thể tạo tài khoản bằng quyền Admin, với lí do là các tài khoản của nhân viên quản lý trạm phát thanh cấp xã chỉ được tạo ra bởi quản lý cấp cao hơn. Giao diện đăng nhập được thể hiện như trong hình dưới đây:

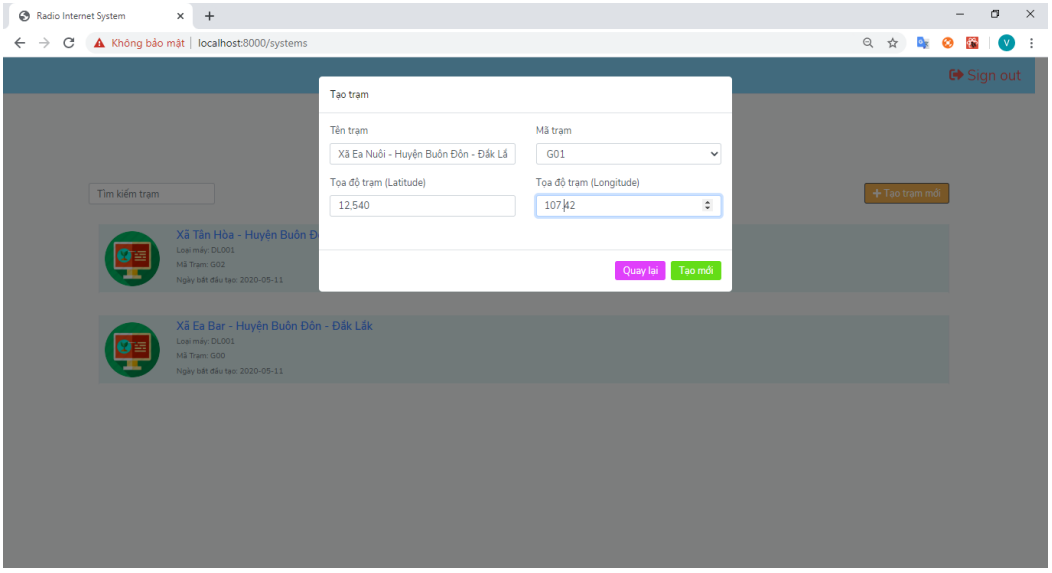


Hình 4.1: Giao diện đăng nhập tài khoản

Sau khi đăng nhập vào tài khoản, nhân viên có thể nhìn thấy tình trạng hoạt động của các trạm thu phát sóng do mình quản lý (được phân quyền quản lý). Giao diện quản lý các trạm thu phát như dưới đây.

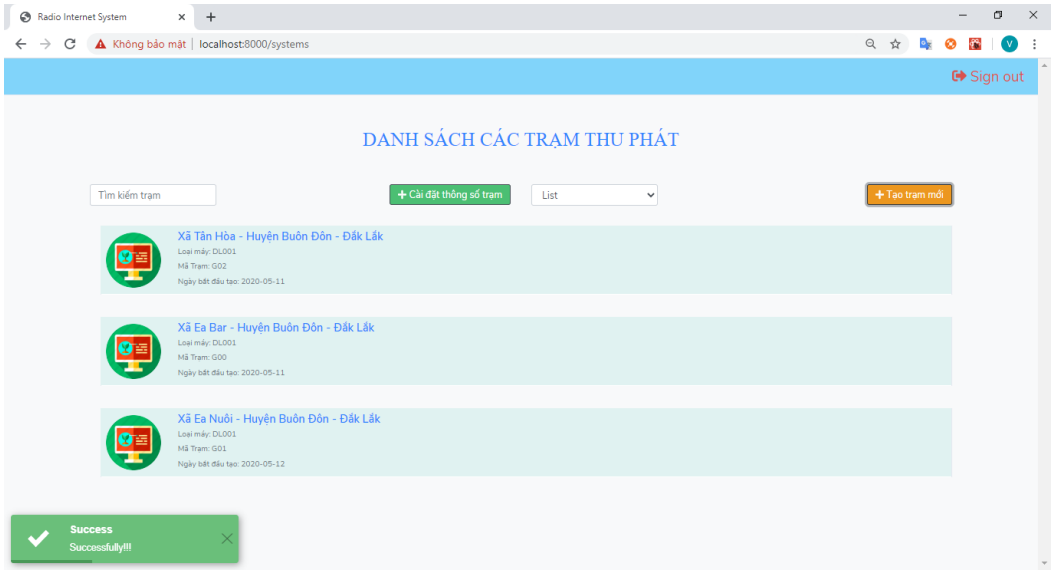


Hình 4.2: Giao diện quản lý các trạm thu phát sóng



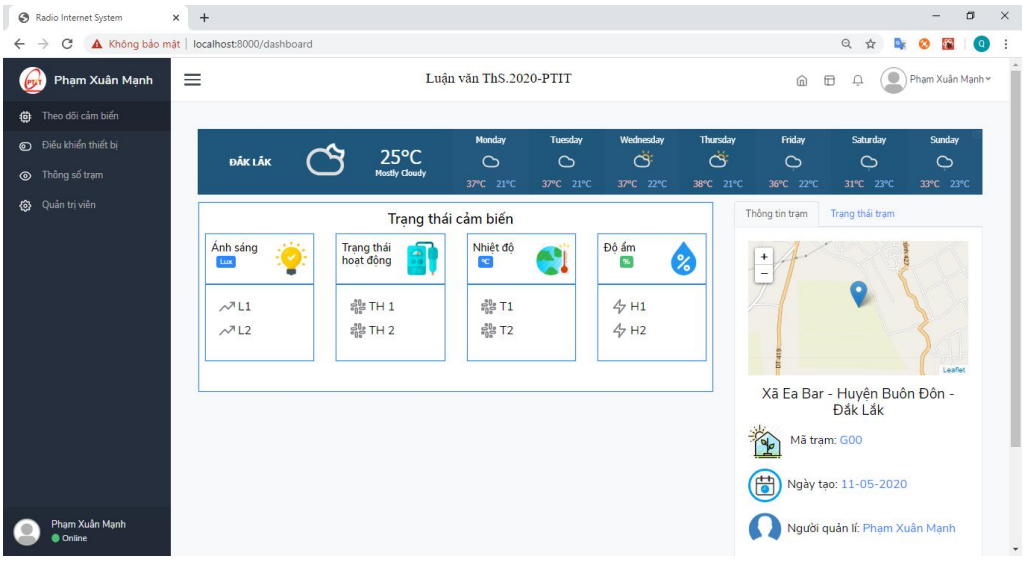
Hình 4.3: Giao diện quản lý các trạm thu phát sóng

Nhân viên quản lý có thể tạo ra một trạm phát sóng mới và phân quyền quản lý cho một nhân viên khác ở cấp cơ sở. Giao diện tạo trạm mới như trong hình 4.3 trên, với mã trạm, tên trạm và toạ độ trạm (để hiển thị trên bản đồ). Sau khi khởi tạo, có thể thấy trạm mới hiển thị trên giao diện quản lý như hình 4.4 dưới đây:



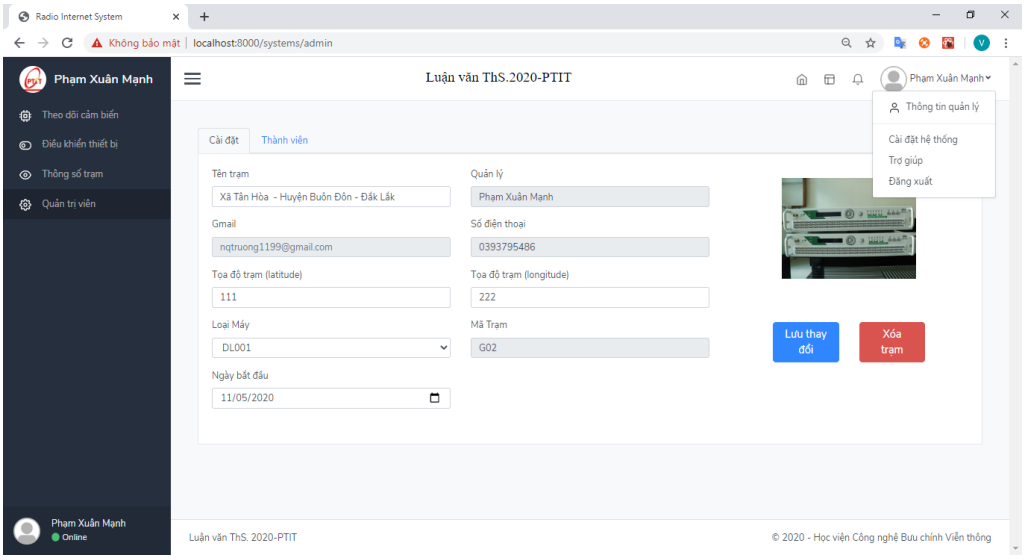
Hình 4.4: Giao diện quản lý các trạm thu phát sóng sau khi tạo mới

Sau đó, dữ liệu từ các trạm thu phát sóng sẽ được cập nhật thời gian thực trên phần mềm theo dõi, quản lý, giám sát các trạm thu phát sóng. Dữ liệu bao gồm: Nhiệt độ, độ ẩm, ánh sáng môi trường, tình trạng máy phát (on/off), được thể hiện như trong hình 4.5 dưới đây. Vị trí và các thông tin chi tiết của Internet Radio cũng được thể hiện ở sidebar bên phải.



Hình 4.5: Giao diện quản lý các trạm thu phát sóng (internet radio)

Trong hình 4.6 dưới đây là giao diện quản trị của nhân viên, có thể thay đổi thông tin trạm, thông tin người quản lý… cũng như là các nhân viên cấp cơ sở được phân quyền quản lý trạm phát sóng đó.

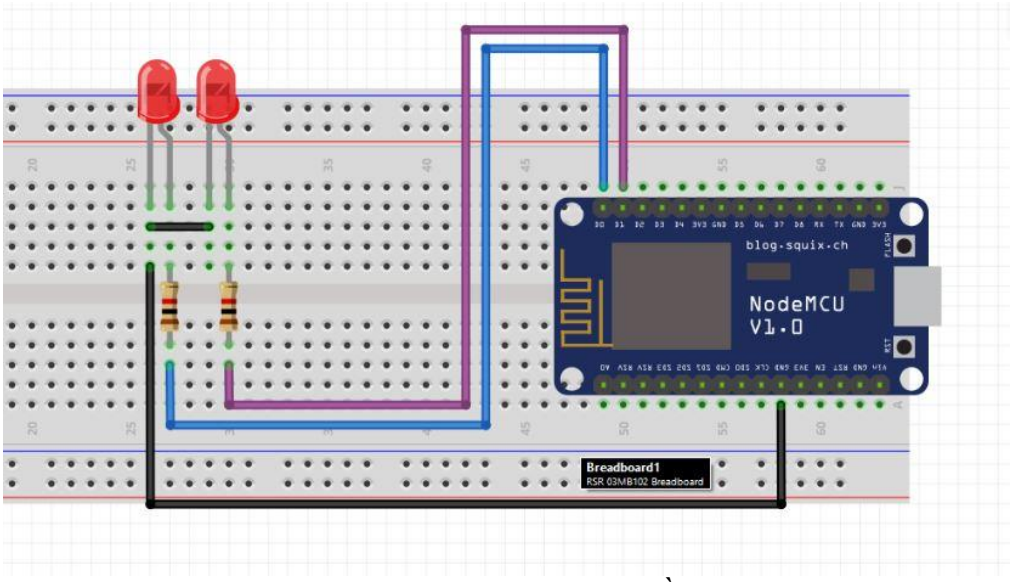


Hình 4.6: Giao diện quản lý thông tin trạm thu/phát sóng

Bằng việc sử dụng các thuật toán giả lập các dữ liệu được gửi về từ các máy thu Internet Radio, tác giả đã hoàn thiện giao diện phần mềm để đáp ứng các yêu cầu phân quyền nhân viên, theo dõi, giám sát và quản lý một trạm thu phát sóng. Trong mục dưới đây, tác giả xin trình bày tiếp về tính năng điều khiển thiết bị từ xa.

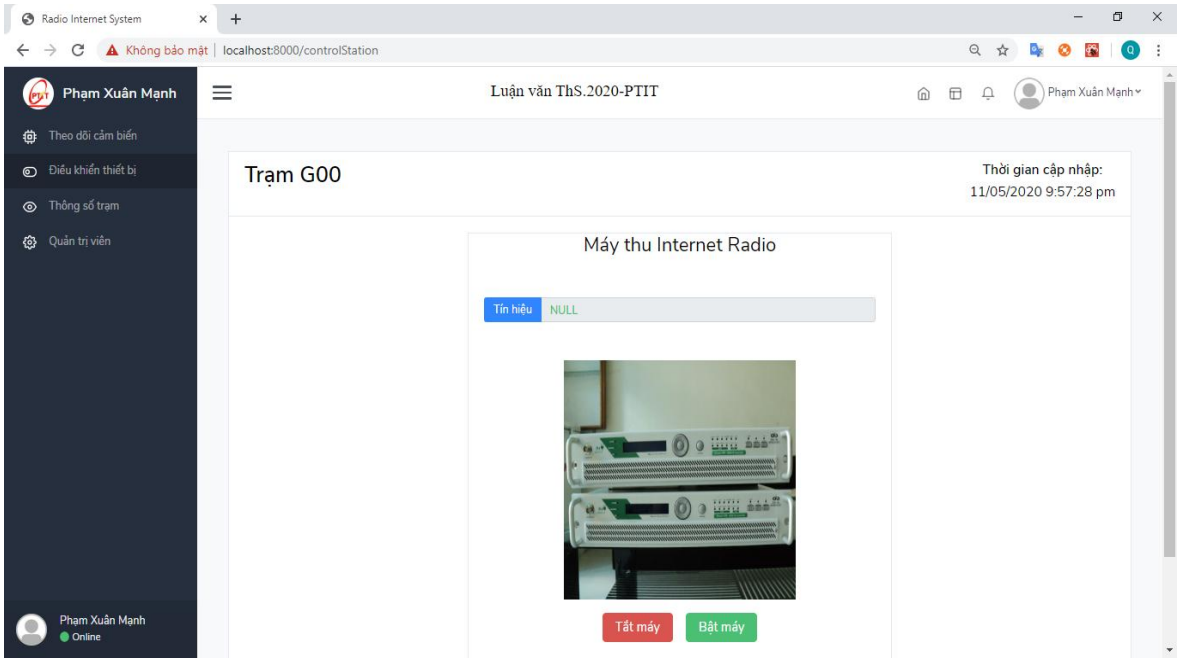
4.2. Kiểm thử tương tác giữa phần cứng mô phỏng Internet radio với phần mềm

Thiết bị mô phỏng là một con chip ESP8266 trong bộ kit Node MCU, có chức năng thu phát dữ liệu qua Wifi/Internet. Lí do lựa chọn module Node MCU bởi vì khả năng đọc dữ liệu cảm biến, điều khiển led, truyền nhận dữ liệu qua Internet, tương tự như module bên trong của Internet Radio. Dữ liệu theo chiều thuận, được gửi liên tục từ Node MCU lên IoT Platform và hiển thị thời gian thực đã được kiểm tra trong mục trên. Dưới đây là mô hình kết nối đèn Led để điều khiển từ xa qua giao diện web.



Hình 4.7: Mô phỏng Internet Radio bằng Node MCU

Trong hình 4.8 dưới đây là giao diện điều khiển thiết bị từ xa. Tình trạng tín hiệu cho biết tình trạng đang hoạt động hay không của máy thu. Việc điều khiển tắt/bật máy có thể được thực hiện từ xa, thông qua 02 button trên giao diện website.



Hình 4.8: Giao diện điều khiển Internet radio từ xa

KẾT LUẬN

Sau thời gian thực hiện luận văn với sự nỗ lực của bản thân cùng với các kiến thức và kinh nghiệm được truyền đạt từ các thầy cô trong khoa, đặc biệt là sự hướng dẫn tận tình của giảng viên hướng dẫn, thầy TS. Nguyễn Quốc Uy, em đã hoàn thành đồ án "*Xây dựng hệ thống iot giám sát các trạm phát thanh cấp xã trong hệ thống truyền thanh không dây"* với các kết quả đạt được như sau:

- Hiểu được kiến trúc tổng thể của IoT platform, tại sao cần thiết xây dựng IoT Platform để quản lý các máy thu Internet Radio trong hệ thống truyền thanh không dây.
- Xây dựng backtend cho hệ thống IoT platform sử dụng NodeJS.
- Xây dựng frontend cho hệ thống IoT platform sử dụng ReactJS.

Sau đó em hoàn thiện IoT platform với các chức năng sau:

+ Có trang đăng ký, đăng nhập cho người dùng.
+ Trang chủ: Liệt kê các máy thu Internet Radio mà người dùng tham gia quản lý. Người dùng có thể nhìn thấy một cách tổng quát về các máy thu, cũng như có thể tìm kiếm một cách dễ dàng. Nếu là Admin thì có thêm quyền tạo ra các máy thu Internet Radio mới và phân công cho một user nào đó quản lý và chỉ có Admin mới có quyền làm điều này.
+ Lấy được dữ liệu từ sensor box gửi về backend lưu vào database. Hiển thị các dữ liệu cần theo dõi và quản lý của các máy thu/phát theo thời gian thực, có thể xuất dữ liệu qua file excel để dễ dàng quản lý.
+ Trang users: Liệt kê danh sách các người dùng đã đăng ký tài khoản. Trang users cung cấp một số thông tin về user như: ảnh đại diện, họ và tên, email, là member hay admin và đang tham gia quản lý các máy thu Internet Radio nào.
+ Trang profile: Cung cấp thông tin của user đang đăng nhập hoặc một user bất kỳ nào đó. Nếu là user đang đăng nhập thì của thể chỉnh sửa hoặc cập nhật thông tin người dùng. Ngoài ra, trang profile còn cung

cấp thông tin của những user khác cũng tham gia quản lý các máy thu Internet Radio khác để khi cần ta có thể liên lạc với họ.

+ Trang admin: Dành cho admin để quản lý các trạm, chỉnh sửa và cập nhật thông tin của máy thu Internet Radio hoặc cũng có thể xóa một máy thu Internet Radio bất kỳ.
+ Có thông báo cho người dùng trên web nếu có sự thay đổi hoặc được thêm vào quản lý một máy thu Internet Radio.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] J. P. Hoffbeck and M. M. Sugiyama, "Real-time FM radio for teaching DSP and communication systems," *2013 IEEE Frontiers in Education Conference (FIE)*, Oklahoma City, OK USA, 2014, pp. 1087-1090.

[2] L. Li, L. Sun, G. Xing, W. Huangfu, R. Zhou and H. Zhu, "ROCS: Exploiting FM Radio Data System for Clock Calibration in Sensor Networks," in *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2130-2144, 2015.

[3] A. F. Pratiwi, G. M. Aji, Purwiyanto, Chairunnisa and A. Munir, "Wireless electronic information board for tsunami early warning system based on FM radio," *2017 7th International Annual Engineering Seminar (InAES)*, Yogyakarta, 2017, pp. 1-4.

[4] H. Fuchs and N. Firber, "ISMA Interoperability and Conformance," in *IEEE MultiMedia*, vol. 12, no. , pp. 96-102, 2005.

[5] D. Radović, M. Čupić, S. Stefanović and D. Majstorović, "Internet radio player implementation using FFmpeg software support," *2017 International Conference on Smart Systems and Technologies (SST)*, Osijek, 2017, pp. 259-262.

[6] James A. Robertson (2016). *U.S. Patent No. US9230084B2*. Washington, DC: U.S. Patent and Trademark Office. Method and system for enabling secure one-time password authentication.

[7] Sung, Jong-Yeop; Lee, Sang-Duck; Ryu, Chang-Ju; Han, Seung-Jo, "Mutual Authentication Protocol using One Time Password for Mobile RFID System," Journal of the Korea Institute of Information and Communication Engineering, Volume 18, Issue 7, pp.1634-1642, 2014

[8] Gotimukul Venkatesh, Sunkara Venu Gopal, Mrudula Meduri, C. Sindhu, "Application of session login and one time password in fund transfer system using RSA algorithm," International conference of Electronics, Communication and Aerospace Technology (ICECA), 2017

[9] Icecast [Online]. Available: https://icecast.org

[10] Shoutcast [Online]. Available: https://www.shoutcast.com

[11] RVR eletronica http://www.rvr.it/en/

PHỤ LỤC 1: CODE CÁC FUNCTION CHÍNH CỦA FRONTEND CỦA IOT PLATFORM

Function SignIn:

```
class SignIn extends React.Component {
   constructor(props) {
      super(props);
      this.state = {
         email: "",
         password: "",
         submitted: false,
         loading: false,
         error: ""
      };
      this.handleChange = this.handleChange.bind(this);
      this.handleSubmit = this.handleSubmit.bind(this);
   }
   handleChange(event) {
      this.setState({
         [event.target.name]: event.target.value
      });
   }
   handleSubmit(event) {
      event.preventDefault();
      this.setState({ submitted: true });
      const { email, password } = this.state;

      // stop here if form is invalid
      if (!(email && password)) {
         return;
      }

      this.setState({ loading: true });
      setTimeout(() => {
         api.login(email, password, (err, result) => {
            if (err) {
               this.setState({ error: err.data === undefined ? err : err.data._error_message,
loading: false });
            } else {
               if (result._id !== undefined) {
                  localStorage.setItem(
                     "userInfo",
                     JSON.stringify({
                        id: result._id,
                        token: result.auth_token,
```

```
                        full_name: result.full_name,
                        photo: result.photo,
                        is_admin: result.is_admin
                    })
                );
            }
            window.location.replace("/stations");
        }
    });
  }, 500);
 }
 render() {
  const { email, password, submitted, loading, error } = this.state;
  return (
   <React.Fragment>
    {error && (
     <Alert color="danger" className="p-2">
      {error}
     </Alert>
    )}
    <Card>
     <CardBody>
      <div className="text-center">
       <CustomImg key={utils.randomString()} src={avatar} className="img-
-user--square-6x mb-2" />
      </div>
      <h1 className="text-center font-weight-bold SignIn-text font-size-
2r">ELECTRIC</h1>
      <Form onSubmit={this.handleSubmit}>
       <FormGroup>
        <Label className="text-primary">Email</Label>
        <Input
         bsSize="lg"
         type="email"
         name="email"
         value={this.state.email}
         onChange={this.handleChange}
         placeholder="Email"
         invalid={submitted && !email ? true : false}
        />
        <FormFeedback invalid>Email is a required field!</FormFeedback>
       </FormGroup>
       <FormGroup>
        <Label className="text-primary ">Password</Label>
        <Input
         bsSize="lg"
         type="password"
```

```
                        name="password"
                        value={this.state.password}
                        onChange={this.handleChange}
                        placeholder="Password"
                        invalid={submitted && !password ? true : false}
                     />
                     <FormFeedback invalid>Passwords is a required field!</FormFeedback>
                  </FormGroup>
                  <FormGroup>
                     <Link to="/auth/reset-password" className="d-inline float-right">
                        Fogot Password
                     </Link>
                     <CustomInput type="checkbox" id="rememberMe" label="Remember me" defaultChecked className="d-inline" />
                  </FormGroup>
                  <div className="text-center mt-4">
                     {loading === false ? (
                        <Button color="primary" size="lg" className="">
                           Sign In
                        </Button>
                     ) : (
                        <LoadingSprinner />
                     )}
                  </div>
               </Form>
               <div className="text-center mt-2">
                  <NavLink to="/auth/sign-up">Do not have an account. Signup?</NavLink>
               </div>
            </CardBody>
         </Card>
      </React.Fragment>
   );
  }
}
```

Function SignUp:

```
class SignUp extends React.Component {
  constructor(props) {
    super(props);
    this.term = React.createRef();
    this.textInput = React.createRef();
    this.state = {
      email: "",
      full_name: "",
```

```
            password: "",
            terms: false,
            submitted: false,
            loading: false,
            error: "",
            success: "",
            confirm_password: "",
            visible: false
        };
        this.handleChange = this.handleChange.bind(this);
        this.handleSubmit = this.handleSubmit.bind(this);
        this.statusTerms = this.statusTerms.bind(this);
    }
    handleSubmit(event) {
        event.preventDefault();
        let data = Object.assign(this.state);
        this.setState({ submitted: true });
        const { email, password, confirm_password } = this.state;

        // stop here if form is invalid
        if (!(email && password && confirm_password && password === confirm_password && !this.validatePassword(password))) {
            return;
        }
        this.setState({ loading: true });
        setTimeout(() => {
            api.register(data, (err, response) => {
                if (err) {
                    this.setState({ error: err.data === undefined ? err : err.data._error_message, loading: false });
                } else {
                    this.setState({ success: "Your account has been successfully created, Please check your email to activate your account", loading: false });
                }
            });
        }, 500);
    }
    handleChange(event) {
        this.setState({
            [event.target.name]: event.target.value
        });
    }
    statusTerms(event) {
        this.setState({
            [event.target.name]: event.target.checked
        });
    }
```

```
  validateEmail(value) {
    let error;
    if (/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/i.test(value)) {
      error = false;
    } else {
      error = true;
    }
    return error;
  }
  validatePassword(value) {
    let error;
    if (/(?=^.{8,}$).*$/i.test(value)) {
      error = false;
    } else {
      error = true;
    }
    return error;
  }
  render() {
    const { email, password, full_name, terms, confirm_password, submitted, loading,
error, success } = this.state;
    return (
      <React.Fragment>
        {success && (
          <Alert className="p-2" color="success">
            {success}
          </Alert>
        )}
        {error && (
          <Alert className="p-2" color="danger" isOpen={success ? false : true}>
            {error}
          </Alert>
        )}
        <Card>
          <CardTitle className="text-center mt-4">
            <h1>Register</h1>
          </CardTitle>
          <CardBody>
            <Form onSubmit={this.handleSubmit}>
              <FormGroup>
                <Label className="text-primary">Email</Label>
                <Input
                  bsSize="lg"
                  type="email"
                  name="email"
                  value={this.state.email}
                  onChange={this.handleChange}
```

```
                        placeholder="Email"
                        invalid={submitted && this.validateEmail(this.state.email) ? true :
false}
                    />
                    {!email && <FormFeedback invalid>Email s a required
field!</FormFeedback>}
                    {email && !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-
Z]{2,4}$/i.test(email) && <FormFeedback invalid>Email is not valid!</FormFeedback>}
                </FormGroup>
                <FormGroup>
                    <Label className="text-primary">Full Name</Label>
                    <Input
                        bsSize="lg"
                        type="text"
                        name="full_name"
                        value={this.state.full_name}
                        onChange={this.handleChange}
                        placeholder="Full Name"
                        invalid={submitted && !full_name ? true : false}
                    />
                    <FormFeedback invalid>Full name is a required
field!</FormFeedback>
                </FormGroup>
                <FormGroup>
                    <Label className="text-primary ">Password</Label>
                    <Input
                        bsSize="lg"
                        type="password"
                        name="password"
                        value={this.state.password}
                        onChange={this.handleChange}
                        placeholder="Password"
                        invalid={submitted && this.validatePassword(this.state.password) ?
true : false}
                    />
                    {!password && <FormFeedback invalid>Password is a required
field!</FormFeedback>}

                    {password && !/(?=^.{8,}$).*$/i.test(password) && (
                        <FormFeedback invalid>Your password must contain at least 8 or
more characters</FormFeedback>
                    )}
                </FormGroup>
                <FormGroup>
                    <Label className="text-primary">Confirm Password</Label>
                    <Input
                        bsSize="lg"
```

```
                        type="password"
                        name="confirm_password"
                        value={this.state.confirm_password}
                        onChange={this.handleChange}
                        placeholder="Confirm Password"
                        invalid={submitted && password !== confirm_password ? true : false}
                    />
                    <FormFeedback invalid>Confirm password incorrect. Please retype the password</FormFeedback>
                </FormGroup>
                <FormGroup>
                    <Label>
                        <CustomInput type="checkbox" name="terms" id="term" onChange={this.statusTerms} />
                    </Label>
                    <NavLink to="/term" target="_blank" className="mt-1">
                        You agree to our terms of service
                    </NavLink>
                </FormGroup>
                <div className="text-center mt-3">
                    {loading === false ? (
                        <Button color="primary" size="lg" disabled={!terms ? true : false}>
                            Sign up
                        </Button>
                    ) : (
                        <LoadingSprinner />
                    )}
                </div>
            </Form>
            <div className="text-center mt-2">
                <NavLink to="/auth/sign-in">Already have an account. Signin?</NavLink>
            </div>
        </CardBody>
    </Card>
</React.Fragment>
);
}
}
```

Function Stations:

```
<React.Fragment>
  <h1 className="text-center m-5">Stations</h1>
    <Container className="mt-2">
      <Row>
```

```
            <Col xs="3">
              <Input
                className="width-percent-40 ml-3"
                id="inputSearch"
                autoComplete="off"
                placeholder="Search Station"
                onKeyUp={this.handleSearch.bind(this)}
              />
            </Col>
            <Col xs="2"></Col>
            <Col xs="2" className="pr-4">
              <Input type="select" onChange={this.handleChangeType}
value={this.state.type}>
                <option value="list">List</option>
                <option value="map">Map</option>
              </Input>
            </Col>
          <Col xs="3"></Col>
          <Col xs="2" className="pr-4">
            <Button className="float-right mr-3 " onClick={this.handleShow.bind(this)}>
            <FontAwesomeIcon icon={faPlus} /> New Station
            </Button>
          </Col>
        </Row>
        <Row>
          <Col>
            {this.state.data.map(({ id, manager, machine, photo, power, sub_id, address,
name, phone_number }, index) => {
              if (ValidInput.isEmpty(this.state.keyWord)) {
                return (
                  <TableStation
                      key={index}
                      id={id}
                      index={index + 1}
                      is_admin={isAdmin}
                      manager={manager}
                      machine={machine}
                      sub_id={sub_id}
                      photo={photo}
                      power={power}
                      address={address}
                      name={name}
                      phone_number={phone_number}
                  />
                );
              } else {
                if (manager.indexOf(this.state.keyWord) !== -1) {
```

```
                    return (
                        <TableStation
                            key={index}
                            id={id}
                            index={index + 1}
                            is_admin={isAdmin}
                            manager={manager}
                            machine={machine}
                            sub_id={sub_id}
                            photo={photo}
                            power={power}
                            address={address}
                            />
                    ); }
                    }
                })
            </Col>
        </Row>
    </Container>
</React.Fragment>
```

Function Dashboard:

```
class Dashboard extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            data: [],
            data_tables: [],
            data_charts: [],
            dataFault: {
                Fault: "00000000"
            },
            status: true,
            info: [],
            isLoaded: false,
            isLoaderAPI_EvaluationList: false,
            type: null,
            response: false,
            socket: true,
            from_date: "",
            to_date: "",
            endpoint: config_socket.ip
        };
        this.handleChangeType = this.handleChangeType.bind(this);
        this.handleSearch = this.handleSearch.bind(this);
        this.handleChangeSocket = this.handleChangeSocket.bind(this);
```

```
    }
    handleChangeType(type) {
        this.setState({ type: type });
    }
    handleSearch(from, to) {
        this.setState({ from_date: from, to_date: to });
        const that = this;
        api.getDataReport(from, to, (err, result) => {
            if (err) {
                Notification("error", "Error", err.data === undefined ? err : err.data._error_message);
            } else {
                let element = [];
                let data = [...result];
                data.map((values, index) => {
                    let value = { ...values };
                    value.time = moment(value.time).format("DD/MM/YYYY h:mm:ss a");
                    element.push(value);
                });
                that.setState({ data_tables: element, data_charts: result, isLoaderAPI: true });
            }
        });
    }
    handleChangeSocket(socket) {
        if (socket === true) {
            this.setState({ socket: true });
        } else {
            this.setState({ socket: false });
        }
    }
    UNSAFE_componentWillMount() {
        const that = this;
        api.getData((err, result) => {
            if (err) {
                Notification("error", "Error", err.data === undefined ? err : err.data._error_message);
            } else {
                let element = [];
                let data = [...result];
                data.map((values, index) => {
                    let value = { ...values };
                    value.time = moment(value.time).format("DD/MM/YYYY h:mm:ss");
                    element.push(value);
                });
                if (data.length !== 0) that.setState({ data_tables: element, data: result[0], data_charts: result, isLoaderAPI: true, dataFault: result[0] });
            }
```

```
    });
  }
  componentDidMount() {
    const that = this;
    const { endpoint } = this.state;
    const sub_id = utils.getStationInfo().sub_id;
    const socket = socketIOClient(endpoint, {
      query: {
        token: utils.getAuthToken(),
        sub_id: sub_id
      }
    });
    socket.on("connect", function() {
      that.setState({ status: true });
    });
    socket.on("disconnect", function() {
      that.setState({ status: true });
    });
    socket.on("substation_" + sub_id, function(value) {
      that.setState({ dataFault: value, data: value });
      if (that.state.socket === true) {
        that.setState({ data_charts: [...that.state.data_charts, value] });
        var length = that.state.data_charts.length;
        if (length >= 20) {
          that.state.data_charts.shift();
        }
        var value_table = Object.assign({}, value);
        var date = moment(value_table.time).format("DD/MM/YYYY h:mm:ss");
        value_table["time"] = date;
        that.setState({ data_tables: [...that.state.data_tables, value_table] });
        var lengtht = that.state.data_tables.length;
        if (lengtht >= 20) {
          that.state.data_tables.shift();
        }
      }
    });
    socket.on("error", function(err) {
      console.log("Error: " + err.message);
    });
    this.setState({ info: utils.getStationInfo(), isLoaded: true });
  }
  render() {
    return !this.state.isLoaded ? (
      <p className="text-center">Loading...</p>
    ) : (
      <Container fluid className="p-0">
        <Row>
```

```
                <Col lg="8" md="12" className="d-flex">
                  <Statistics data={this.state.data} />
                </Col>
                <Col lg="4" md="2" className="d-md-block">
                  <StationInformation  data={this.state.info}  dataFault={this.state.dataFault}
status={this.state.status} />
                </Col>
              </Row>
              <Row>
                <Col lg="6" className="d-flex">
                  <Tables                                                    data={this.state.data_tables}
handleChangeType={this.handleChangeType} />
                </Col>
                <Col lg="6" className="d-flex">
                  <Chart
                    data={this.state.data_charts}
                    type={this.state.type}
                    handleSearch={this.handleSearch}
                    handleChangeSocket={this.handleChangeSocket}
                  />
                </Col>
              </Row>
            </Container>
        );
    }
}
```