ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ────────

**BÀI TẬP LỚN**

IOT VÀ ỨNG DỤNG

**HỆ THỐNG GIÁM SÁT NHÀ TỪ XA**

|  |  |
| --- | --- |
| **Sinh viên thực hiện** | **Mã sinh viên** |
| Tạ Hữu Bình | 20190094 |
| Nguyễn Quốc Hào | 20194043 |
| Hà Đức Tuấn | 20190072 |
| Mã học phần | : IT4735 |
| Mã lớp | : 136050 |
| Giảng viên hướng dẫn | : TS. Nguyễn Đình Thuận |

Hà Nội, tháng 3 năm 2023

Mục lục

[Chương 1. Giới thiệu đề tài 4](#_Toc129733949)

[Chương 2. Thành viên nhóm và phân chia công việc 5](#_Toc129733950)

[I. Thông tin thành viên nhóm 5](#_Toc129733951)

[II. Phân chia công việc 5](#_Toc129733952)

[Chương 3. Thiết kế hệ thống 6](#_Toc129733953)

[I. Sơ đồ thiết kế tổng quan 6](#_Toc129733954)

[II. Phân tích chi tiết các thành phần hệ thống 6](#_Toc129733955)

[**1.** **Giao thức MQTT và xây dựng MQTT Broker** 6](#_Toc129733956)

[**2.** **Cảm biến DHT11** 7](#_Toc129733957)

[**3.** **Vi xử lý ESP32** 7](#_Toc129733958)

[**4.** **Server** 8](#_Toc129733959)

[***a.*** ***NodeJS*** 8](#_Toc129733960)

[***b.*** ***ExpressJS*** 9](#_Toc129733961)

[***c.*** ***Mongoose*** 9](#_Toc129733962)

[***d.*** ***MQTT.js*** 9](#_Toc129733963)

[**5.** **Database** 9](#_Toc129733964)

[**6.** **Web client** 10](#_Toc129733965)

[***a.*** ***HTML + CSS*** 10](#_Toc129733966)

[***b.*** ***JavaScript + Jquery*** 10](#_Toc129733967)

[III. Các công nghệ sử dụng 10](#_Toc129733968)

[1. Server 10](#_Toc129733969)

[2. Web frontend 10](#_Toc129733970)

[3. MQTT Broker 10](#_Toc129733971)

[4. Hardware 10](#_Toc129733972)

[5. Giao thức sử dụng 11](#_Toc129733973)

[Chương 4. Xây dựng và triển khai 12](#_Toc129733974)

[I. Các thành phần của hệ thống 12](#_Toc129733975)

[1. DHT11 12](#_Toc129733976)

[2. MQTT 12](#_Toc129733977)

[3. Webclient 13](#_Toc129733978)

[4. Xây dựng MQTT Broker 15](#_Toc129733979)

[5. Server 16](#_Toc129733980)

[6. Database 18](#_Toc129733981)

[Chương 5: Kết luận 20](#_Toc129733982)

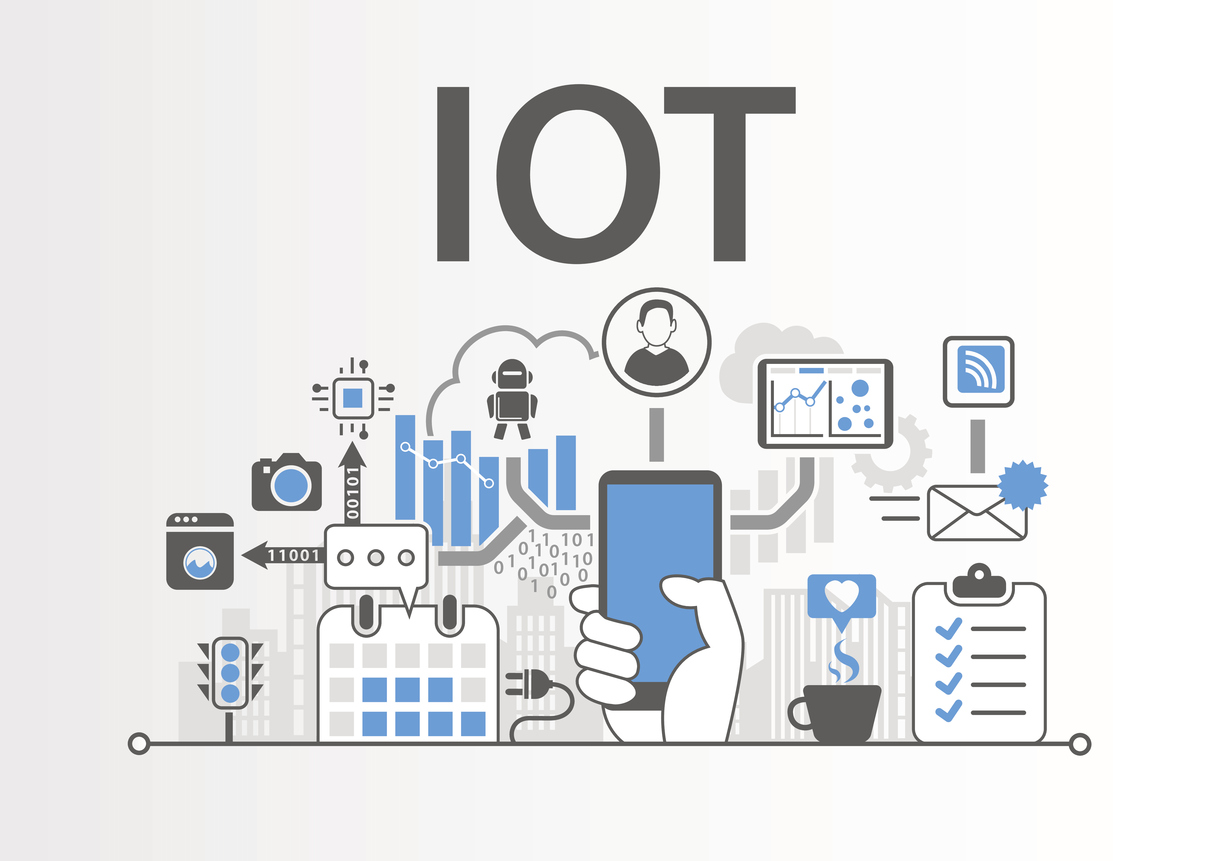
[I. Khó khăn trong quá trình hoàn thiện 20](#_Toc129733983)

[II. Hướng phát triển 20](#_Toc129733984)

[Tài liệu tham khảo 21](#_Toc129733985)

# Chương 1. Giới thiệu đề tài

Với sự phát triển của Internet, smartphone và đặc biệt là các thiết bị cảm biến thì Internet of Things (IoT) đã trở thành xu hướng mới của thế giới. Các ứng dụng công nghệ đến từ Internet of Things đã mang đến những giá trị to lớn cho con người như những lĩnh vực như y học, sinh học, giao thông, …



Trong những năm ngần đây một trong những ứng dụng nổi bật của IoT đang được quan tâm và ứng dụng nhiều hiện nay là Smart Home. Đó cũng là lý do mà nhóm chọn đề tài này cho bài tập lớn của môn học.

# Chương 2. Thành viên nhóm và phân chia công việc

## Thông tin thành viên nhóm

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ và tên** | **Email** |
| 20190094 | Tạ Hữu Bình | binh.th190094@sis.hust.edu.vn |
| 20194043 | Nguyễn Quốc Hào | hao.nq194043@sis.hust.edu.vn |
| 20190072 | Hà Đức Tuấn | tuan.hd190072@sis.hust.edu.vn |

## Phân chia công việc

|  |  |
| --- | --- |
| **Họ tên** | **Công việc** |
| Tạ Hữu Bình | Xây dựng web client và tương tác với MQTT |
| Nguyễn Quốc Hào | Lắp đặt phần cứng, lập trình truyền và nhận dữ liệu với ESP 32 |
| Hà Đức Tuấn | Thiết kế cơ sở dữ liệu với MongoDB và xây dựng server |

# Chương 3. Thiết kế hệ thống

## Sơ đồ thiết kế tổng quan

Diagram

Description automatically generated

Hệ thống gồm 6 thành phần chính:

* Server
* Database
* HiveMQ Cloud Broker
* Web
* ESP32
* Sensor

## Phân tích chi tiết các thành phần hệ thống

1. **Giao thức MQTT và xây dựng MQTT Broker**

MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp theo mô hình publish/subscribe, được sử dụng nhiều cho các hệ thống IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định.

HiveMQ Cloud Broker là một dịch vụ môi trường trung gian IoT được cung cấp bởi HiveMQ, cung cấp giải pháp trung gian truyền thông dễ dàng cho các ứng dụng IoT. Nó được thiết kế để giảm thiểu thời gian và chi phí triển khai, vận hành và mở rộng các ứng dụng IoT. HiveMQ Cloud Broker cung cấp cho người dùng một cách dễ dàng để kết nối các thiết bị IoT, giám sát và điều khiển các dữ liệu IoT và tăng cường độ tin cậy và hiệu suất của hệ thống IoT. Nó cung cấp các tính năng an ninh, phân tích và đám mây để giúp người dùng quản lý các ứng dụng IoT một cách dễ dàng và hiệu quả. Do đó trong project này, nhóm sử dụng HiveMQ Cloud Broker.

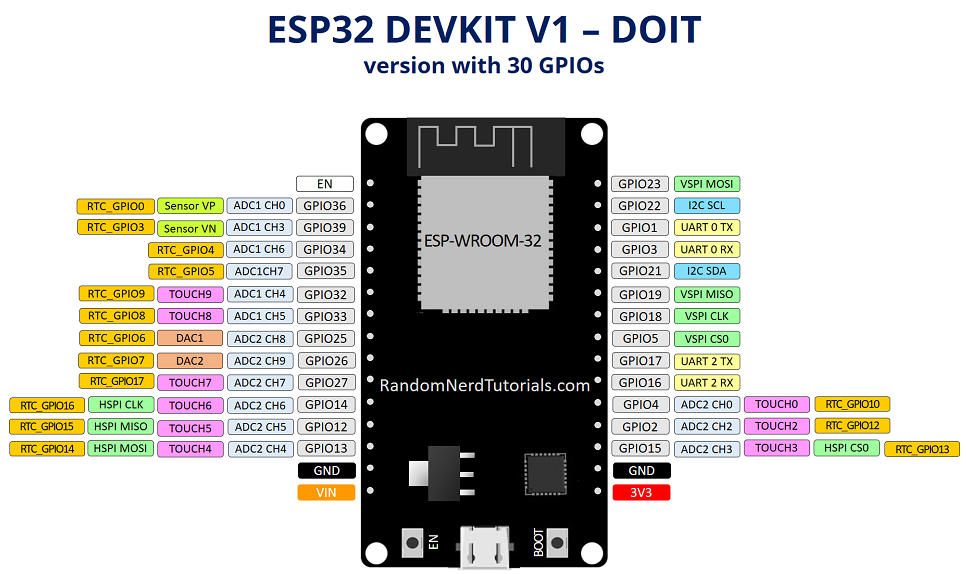
1. **Cảm biến DHT11**

DHT11 là một cảm biến kỹ thuật số giá rẻ để đo nhiệt độ, độ ẩm môi trường.

Cảm biến DHT11 bao gồm một phần tử cảm biến độ ẩm điện dung và một điện trở nhiệt để cảm nhận nhiệt độ. Tụ điện cảm biến độ ẩm có hai điện cực với chất nền giữ ẩm làm chất điện môi giữa chúng. Thay đổi giá trị điện dung xảy ra với sự thay đổi của các mức độ ẩm. IC đo, xử lý các giá trị điện trở đã thay đổi này và chuyển chúng thành dạng kỹ thuật số. Để đo nhiệt độ, cảm biến này sử dụng một nhiệt điện trở có hệ số nhiệt độ âm, làm giảm giá trị điện trở của nó khi nhiệt độ tăng. Để có được giá trị điện trở lớn hơn ngay cả đối với sự thay đổi nhỏ nhất của nhiệt độ, cảm biến này thường được làm bằng gốm bán dẫn hoặc polymer.

1. **Vi xử lý ESP32**

ESP32 là một hệ thống vi điều khiển trên chip (SoC) giá rẻ của Espressif Systems, nhà phát triển của ESP8266 SoC. Nó là sự kế thừa của SoC ESP8266 và có cả hai biến thể lõi đơn và lõi kép của bộ vi xử lý 32-bit Xtensa LX6 của Tensilica với Wi-Fi và Bluetooth tích hợp.



Thông số kỹ thuật ESP32 DEVKIT DOIT:

* + Số lượng lõi: 2 (dual core)
  + Wifi: 2.4GHz
  + Kiến trúc: 32 bit
  + Clock frequency: 240 MHz
  + RAM: 512KB

1. **Server**

Server sử dụng framework ExpressJS, Mongoose và MQTT.js trên runtime NodeJS.

1. ***NodeJS***

Node.js là một nền tảng phát triển được xây dựng trên engine JavaScript V8 của Google, cho phép lập trình viên sử dụng JavaScript để phát triển các ứng dụng server-side. Node.js cho phép xây dựng các ứng dụng web động, ứng dụng desktop, các dịch vụ back-end, các ứng dụng mạng và các ứng dụng Internet of Things (IoT). Node.js được thiết kế để xử lý hàng nghìn kết nối đồng thời, cho phép xử lý nhiều yêu cầu cùng một lúc mà không làm giảm tốc độ xử lý. Node.js còn được phát triển với cộng đồng lớn, có nhiều thư viện và framework hỗ trợ giúp cho việc phát triển ứng dụng trở nên dễ dàng hơn.

1. ***ExpressJS***

Express.js là một framework phổ biến và mạnh mẽ cho Node.js, được sử dụng để phát triển các ứng dụng web động. Nó cung cấp cho lập trình viên các công cụ và tính năng để xây dựng các ứng dụng web động dễ dàng hơn.

Express.js cho phép xử lý các yêu cầu HTTP, quản lý các middleware, tạo các route, thực hiện xác thực và nhiều hơn nữa. Nó là một framework rất linh hoạt và cho phép bạn tùy chỉnh để phù hợp với nhu cầu phát triển của mình. Với cộng đồng lớn và đầy năng lượng, Express.js là một trong những framework được sử dụng rộng rãi nhất cho phát triển ứng dụng web động trên Node.js.

1. ***Mongoose***

Mongoose là một thư viện MongoDB mạnh mẽ và linh hoạt cho Node.js. Nó cho phép lập trình viên tương tác với cơ sở dữ liệu MongoDB bằng cách định nghĩa các mô hình đối tượng (object models) dựa trên schema (bảng mẫu) để truy xuất dữ liệu.

Mongoose cung cấp nhiều tính năng hữu ích như xác thực, tìm kiếm, tạo, cập nhật và xóa dữ liệu MongoDB một cách dễ dàng. Với Mongoose, lập trình viên có thể đơn giản hóa việc phát triển ứng dụng web Node.js với cơ sở dữ liệu MongoDB một cách nhanh chóng và hiệu quả.

1. ***MQTT.js***

Thư viện MQTT.js được server sử dụng để kết nối tới MQTT Broker. Với sự hỗ trợ của thư viện cộng với sự linh hoạt của ngôn ngữ Javascript, server có thể subscribe và nhận dữ liệu từ MQTT Broker chỉ với vài dòng code.

1. **Database**

MongoDB là hệ quản trị cơ sở dữ liệu dạng NoSQL được sử dụng trong project này. Khác với các hệ quản trị dạng SQL, MongoDB không sử dụng các bảng dữ liệu và cũng không có những yêu cầu chặt chẽ, mang tính bắt buộc. Cấu trúc dữ liệu trong MongoDB cũng có thể thay đổi dễ dàng, giúp làm tăng tốc độ phát triển ứng dụng và giảm sự phức tạp khi triển khai.

Để khai thác khả năng của MongoDB, nhóm sử dụng dịch vụ MongoDB Atlas – một dịch vụ quản lý dữ liệu đám mây (cloud) do đội ngũ của MongoDB cung cấp. Server sẽ kết nối trực tiếp đến cơ sở dữ liệu thông qua dịch vụ này trên Internet. Vì vậy, nhóm không cần phải tự cài đặt và cấu hình MongoDB trên máy tính của mình. Có thể thấy, dịch vụ này rất hữu ích đối với những project nhỏ, cần thời gian phát triển nhanh và mang tính prototype.

1. **Web client**
   1. ***HTML + CSS***

Nhóm sử dụng HTML và CSS để tạo giao diện. Đây là những ngôn ngữ dễ học trong thời gian ngắn, không đòi hỏi quá nhiều kiến thức sâu để có thể làm ra một trang web đơn giản.

* 1. ***JavaScript + Jquery***

Để tương tác với các thành phần trong web, nhóm cũng sử dụng JavaScript và Jquery. Đây cũng là những ngôn ngữ tương đối dễ học và phát triển.

## Các công nghệ sử dụng

### Server

* NodeJS
* ExpressJS
* Mongoose
* MQTT.js

### Web frontend

* HTML
* CSS
* JavaScript
* Jquery
* mqtt-ws.js (Paho JavaScript client)

### MQTT Broker

* HiveMQ

### Hardware

* ESP32
* DHT11
* Led
* Button

### Giao thức sử dụng

* Web – Server: HTTPS
* Server/Hardware – MQTT Broker: MQTT

# Chương 4. Xây dựng và triển khai

## Các thành phần của hệ thống

### DHT11

*Sử dụng thư viện DHT.h của adafruit sensor*

Khởi tạo DHT:

DHT dht(DHTPIN, DHTTYPE);

Với DHTPIN = 15, DHTTYPE = DHT11

Đọc dữ liệu từ sensor:

float humi = dht.readHumidity();

  // read temperature in Celsius

  float tempC = dht.readTemperature();

  // read temperature in Fahrenheit

  float tempF = dht.readTemperature(true);

Kết quả chạy:

Text

Description automatically generated

### MQTT

*Kết nối với MQTT Broker bằng thư viện PubSubClient.h*

Khởi tạo đối tượng client kết nối với Broker:

WiFiClientSecure wifiClient;

PubSubClient mqttClient(wifiClient);

Set broker server và hàm callback (Hàm được gọi khi có dữ liệu gửi tới topic đã subcribe):

mqttClient.setServer(mqttServer, mqttPort);

mqttClient.setCallback(callback);

Kết nối tới broker:

    while (!mqttClient.connected()) {

      Serial.println("Reconnecting to MQTT Broker..");

      String clientId = "ESP32Client-";

      clientId += String(random(0xffff), HEX);

      if (mqttClient.connect(clientId.c\_str(), mqtt\_username, mqtt\_password)) {

         Serial.println("Connected to MQTT BRoker!.");

         // subscribe to topic

         mqttClient.subscribe(topicSubcribe);

      } else {

          Serial.print("failed with state ");

          Serial.print(mqttClient.state());

          delay(2000);

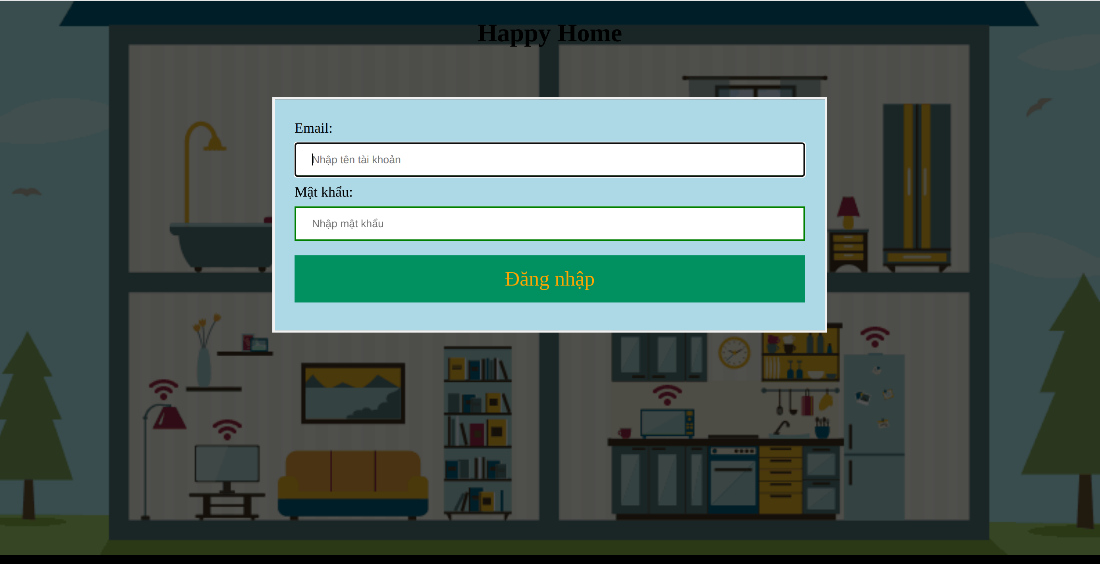
      }

     }

### Webclient

* + - * 1. ***Đăng nhập, đăng xuất***

Người dùng đăng nhập bằng tài khoản đã được cấp trước đó để truy cập vào hệ thống website.



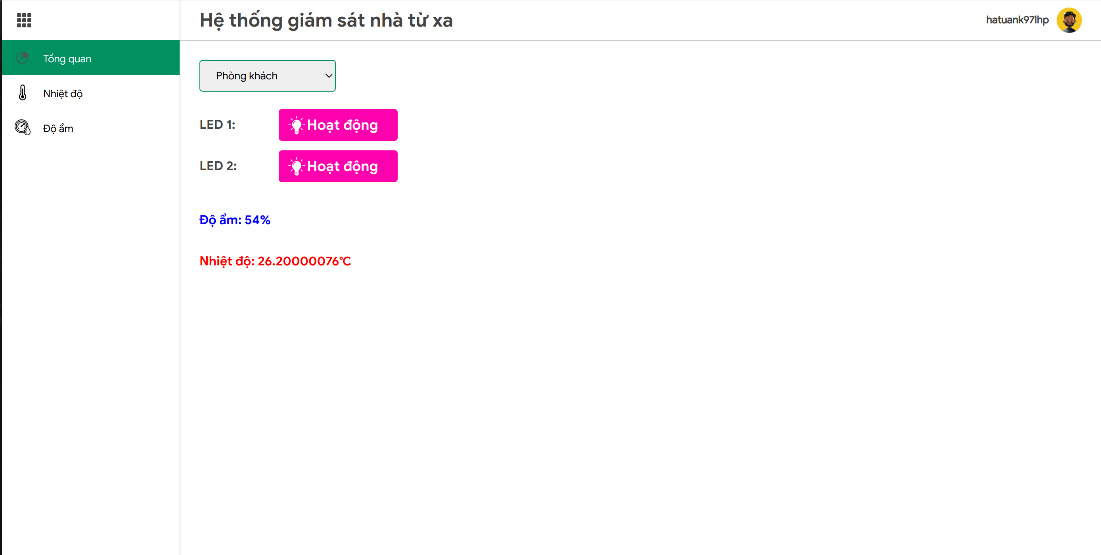
Sau khi đăng nhập người dùng sẽ truy cập được vào website.

Graphical user interface, application, Teams

Description automatically generated

* 1. ***Xem thông tin các phòng, các thiết bị***
* *Xem thông tin các phòng*

Khi người dùng bấm vào chọn phòng để hiển ra các phòng trong nhà. Sau khi chọn phòng, các thiết bị có trong phòng sẽ được hiện ra, hoặc nếu vào các tab Nhiệt độ hay Độ ẩm, thông tin thống kê sẽ được hiển thị.



* *Xem thông tin các thiết bị*

Với bóng đèn, người dùng có thể ấn bật tắt để có thể bật và tắt các bóng đèn có trong phòng. Nếu trong phòng trang bị các thiết bị đo độ ẩm và nhiệt độ, người dùng cũng có thể nhìn thấy thông tin ở trên màn hình.

* 1. ***Thống kê số liệu***

Hệ thống còn có chức năng thống kê số liệu nhiệt độ và độ ẩm theo thời gian thực.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

### Xây dựng MQTT Broker

Các MQTT Broker sẽ nắm vai trò trung gian, tiếp nhận dữ liệu từ các thiết bị và đẩy dữ liệu về cho server và web. Phần demo chỉ sử dụng một broker.

Nhóm sử dụng HiveMQ Cloud để triển khai broker, với bản free thì có giới hạn 100 MQTT Client Sessions và 10 GB data traffic.

Graphical user interface, application

Description automatically generated

Nhóm cũng tạo MQTT credentials để xác thực cho thiết bị IoT muốn kết nối đến HiveMQ Cloud cluster.

Các thiết bị muốn truyền nhận dữ liệu thì phải publish/subscribe 2 topic:

* IoT\_hust/command: web sẽ gửi dữ liệu về topic này, esp32 sẽ đọc dữ liệu từ topic và sẽ tiến hành đặt trạng thái cho đèn led
* IoT\_hust/data: esp32 sẽ gửi dữ liệu trạng thái đèn led1, led2 và dữ liệu thông tin về độ ẩm, nhiệt độ lên topic này, web và server sẽ đọc dữ liệu từ topic này

Dưới đây là ví dụ cho việc esp32 gửi dữ liệu lên broker:

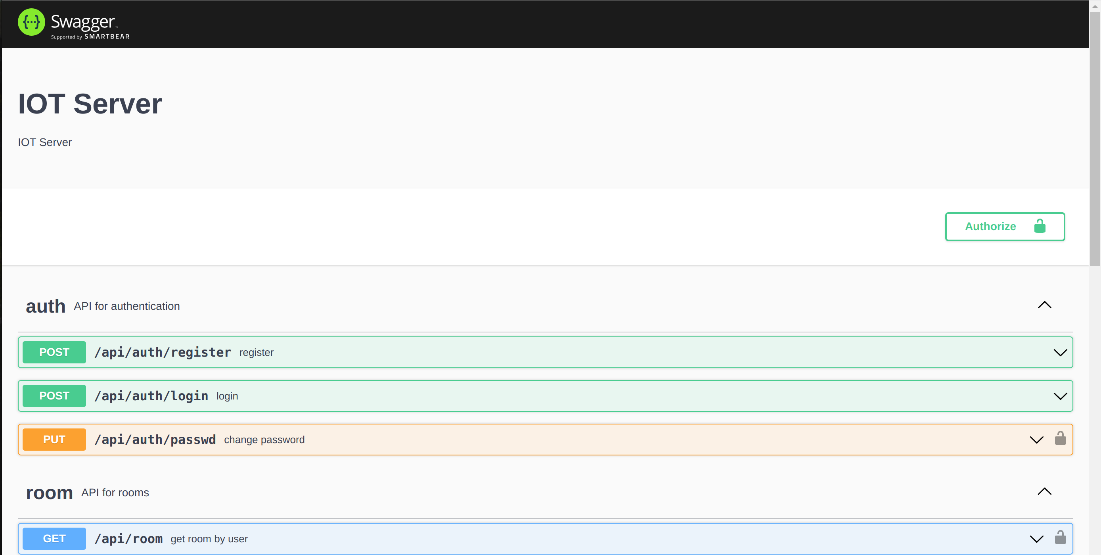
Graphical user interface, text, application, chat or text message

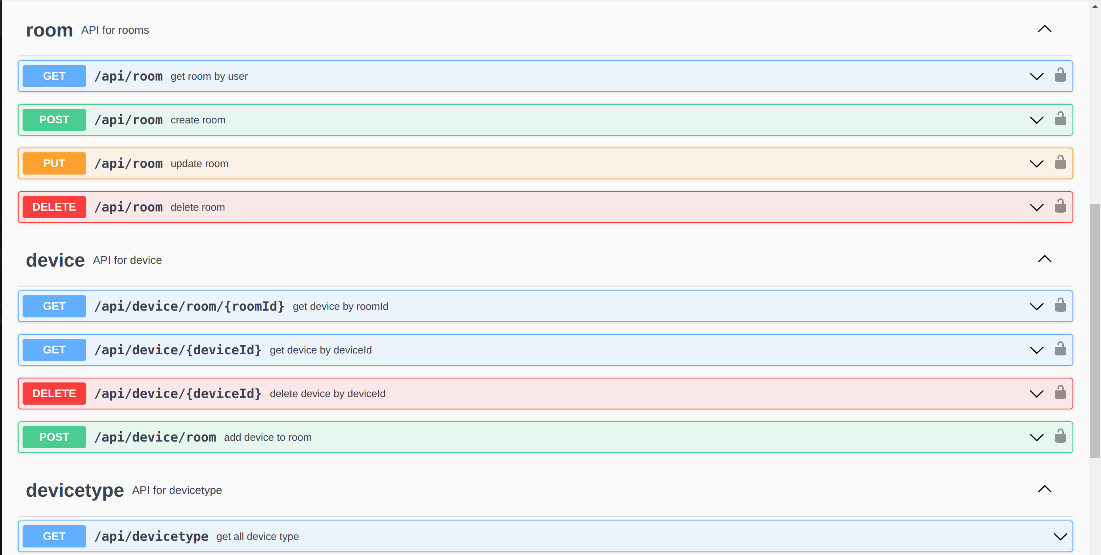
Description automatically generated

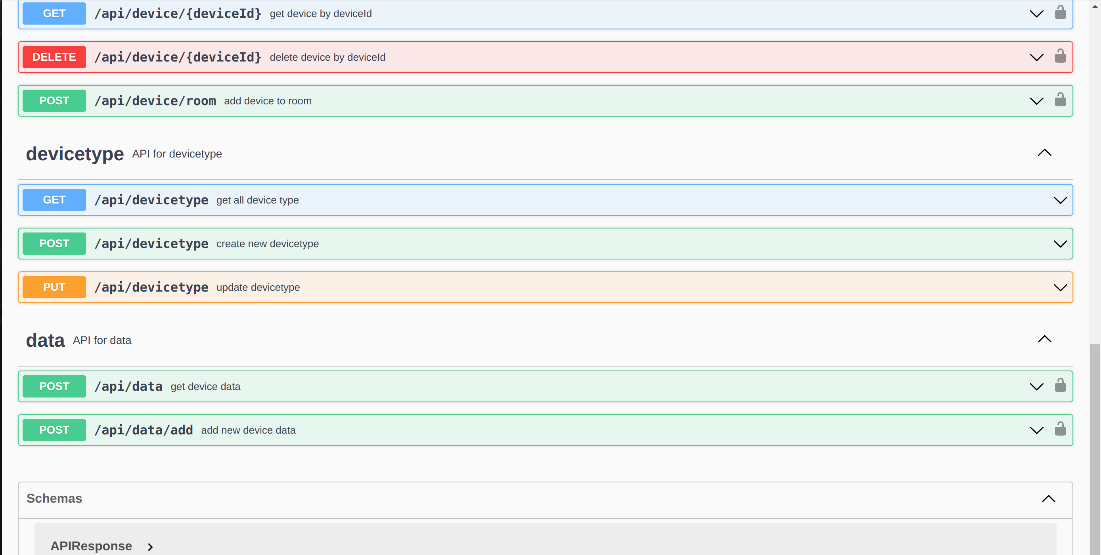
### Server

Server sẽ bao gồm các hoạt động sau:

* Kết nối và subscribe các topic của các MQTT Broker để liên tục nhận dữ liệu do các thiết bị trong nhà thông mình đẩy lên. Dữ liệu sau khi nhận sẽ được lưu trữ tại cơ sở dữ liệu MongoDB.
* Giao tiếp với cơ sở dữ liệu MongoDB để đẩy cũng như lấy dữ liệu trả về cho API.
* Cung cấp các API để mobile app và web sử dụng. Các API chính bao gồm:
  + Đăng nhập, đổi mật khẩu.
  + Xem danh sách các phòng trong nhà, tạo phòng mới.
  + Xem, thêm và cập nhật các thiết bị vào một căn phòng.
  + Xem lịch sử dữ liệu của các thiết bị.
  + Xem và thêm các loại thiết bị khác nhau.

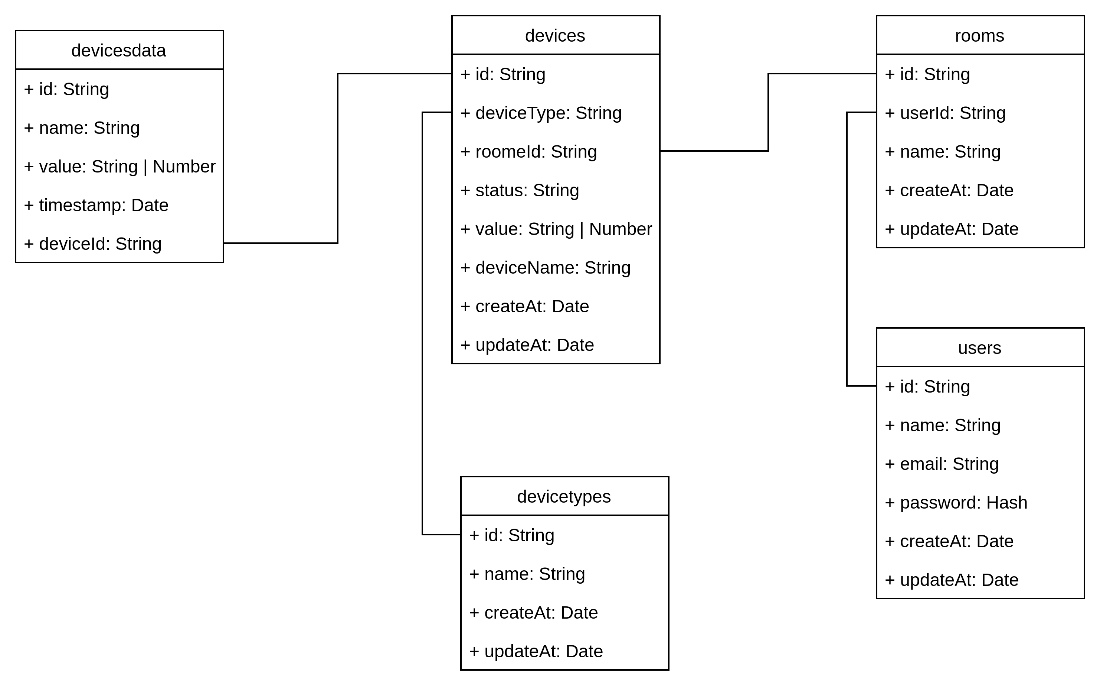






### Database

MongoDB không sử dụng mô hình bảng và khóa để lưu trữ dữ liệu. Tuy nhiên, để có thể xử lý và lưu trữ dữ liệu một cách nhất quán, việc xây dựng các bảng dữ liệu và mối liên kết giữa chúng vẫn là vô cùng cần thiết. Dưới đây là sơ đồ bảng được sử dụng trong project.



Sơ đồ gồm 5 bảng, hầu hết các bảng đều có hai trường mặc định là **createdAt** và **updatedAt** để lưu thông tin về thời điểm tạo và cập nhật của từng dữ liệu.

Bảng **Users** gồm các trường cơ bản để định danh người dùng như email, name. Password trong users là password của người dùng đã được hash lại.

Bảng **Rooms** dùng để quản lý danh sách phòng, bao gồm userId để biết được từng phòng là của ai.

Bảng **DeviceType** sẽ gồm danh sách các loại thiết bị khác nhau, như: cảm biến nhiệt, cảm biến độ ẩm, đèn led, …

Bảng **Devices** sẽ quản lý các thiết bị trong mỗi căn phòng, với các thông tin về loại thiết bị, định danh phòng, trạng thái của thiết bị, giá trị thông số hiện tại của thiết bị.

Bảng **DeviceDatas** lưu các dữ liệu mà các thiết bị trả về, gồm định danh của loại dữ liệu, giá trị của dữ liệu, timestamp (thời điểm dữ liệu được gửi đi) và một số thông tin thêm nếu có.

# Chương 5: Kết luận

## Khó khăn trong quá trình hoàn thiện

Do chưa có nhiều kinh nghiệm trong việc phát triển một hệ thống hoàn chỉnh nên còn nhiều khó khăn trong quá trình phát triển.

Cảm biến và ESP32 không quá khó để tìm hiểu nhưng trong quá trình lắp đặt và demo gặp nhiều các lỗi phát sinh. Nhóm đã phân chia công việc mỗi người làm một thành phần riêng biệt nhưng luôn có sự giúp đỡ và hỗ trợ lẫn nhau. Hệ thống gồm nhiều thành phần và các thành phần được phát triển riêng nên quá trình đồng bộ đòi hỏi nhóm phải có sự trao đổi với nhau rất nhiều để kiểm tra và khác phục lỗi.

* 1. Hướng phát triển

Đầu tiên do thời gian còn hạn chế nên còn nhiều chức năng chưa được hoàn thiện và các chức năng đến hiện tại chỉ là tối giản để đảo bảo đủ để minh họa những chức năng quan trọng nhất của hệ thống.

Nên hướng phát triển của nhóm sẽ cố gắng hoàn thiện các chức năng và triển khai lên các hệ thống thực tế mà không còn dưới dạng demo sản phẩm.

Toàn bộ mã nguồn được công khai tại: <https://github.com/Tahuubinh/IoT_project>

# Tài liệu tham khảo

[1] Bài giảng học phần IoT và Ứng dụng.

[2] Thư viện giao diện MUI:

<https://mui.com/>

[3] MQTT là gì? Vai của MQTT trong IoT:

[https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-IoT-V3m5WL3bKO7](https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7)