Báo cáo định kì:

Mục lục

[1 GIỚI THIỆU TỔNG QUAN VỀ NODEJS 2](#_Toc110435194)

[1.1 Khái niệm NodeJS 2](#_Toc110435195)

[1.2 Đặc điểm của NodeJS 3](#_Toc110435196)

[1.3 NodeJS có thể làm được gì 3](#_Toc110435197)

[2 Giới thiệu Reactjs 3](#_Toc110435198)

[2.1 1.1 Reactjs là gì 3](#_Toc110435199)

[2.2 Virtual DOM 4](#_Toc110435200)

[2.2.1 Mô hình Đối tượng Tài liệu (DOM) là gì? 4](#_Toc110435201)

[2.2.2 Virtual DOM là gì 4](#_Toc110435202)

[2.2.3 Cách tương tác với nhau Virtual DOM và ReactJS DOM 4](#_Toc110435203)

[2.3 Performance 4](#_Toc110435204)

[**2.4** **Component** 5](#_Toc110435205)

[2.5 Điểm mạnh 5](#_Toc110435206)

[3 Tìm hiểu về socketIo 5](#_Toc110435207)

[**3.1** **Cùng nhìn lại về websocket** 5](#_Toc110435208)

[3.2 Socket io là gì? 5](#_Toc110435209)

[3.3 Socket io có những ưu điểm 6](#_Toc110435210)

[3.3.1 Bảo mật cao 6](#_Toc110435211)

[3.3.2 Kết nối tự động tới server 6](#_Toc110435212)

[3.3.3 Mã hóa nhị phân 6](#_Toc110435213)

[3.3.4 Cho phép tạo kênh và phòng 7](#_Toc110435214)

[4 Giới thiệu về My SQL 7](#_Toc110435215)

[4.1 **SQL là gì??** 7](#_Toc110435216)

[4.2 Hệ quản trị cơ sở dữ liệu: 7](#_Toc110435217)

[4.3 **Giới thiệu về hệ quản trị dữ liệu MySQL:** 7](#_Toc110435218)

[5 Tìm hiểu về oath 2.0: 7](#_Toc110435219)

[5.1 OAuth2 là gì? 7](#_Toc110435220)

[5.2 Một số khái niệm cơ bản 7](#_Toc110435221)

[5.3 OAuth2 hoạt động 8](#_Toc110435222)

[5.4 Sơ đồ luồng hoạt động của OAuth2 8](#_Toc110435223)

[5.5 Ưu điểm của OAuth2 9](#_Toc110435224)

[6 Báo cáo chương trình 10](#_Toc110435225)

[6.1 Sơ đồ cơ sở dữ liệu: 10](#_Toc110435226)

[6.2 Các Model dữ liệu trong code: 11](#_Toc110435227)

[6.3 API: 25](#_Toc110435228)

[6.4 Frontend 26](#_Toc110435229)

Danh mục hình:

[Hình 1 Sơ đồ hoạt động của OAth2 8](#_Toc110435124)

[Hình 2 Mô hình Diagram 10](#_Toc110435125)

[Hình 3 Model của user 12](#_Toc110435126)

[Hình 4 Các mối liên kết của user 14](#_Toc110435127)

[Hình 5 Model của Post 16](#_Toc110435128)

[Hình 6 Model của Comment 18](#_Toc110435129)

[Hình 7 Mô hình của Like Post 18](#_Toc110435130)

[Hình 8 Model của conversation 19](#_Toc110435131)

[Hình 9 Model của participant 20](#_Toc110435132)

[Hình 10 Model của message 21](#_Toc110435133)

[Hình 11 Model của attachment 22](#_Toc110435134)

[Hình 12 Model của OTP 23](#_Toc110435135)

[Hình 13 Model của follower 24](#_Toc110435136)

[Hình 14 Các api của xác thực và user 25](#_Toc110435137)

[Hình 15 Các api của post và comment 25](#_Toc110435138)

[Hình 16 Các api của conversation và các model 26](#_Toc110435139)

[Hình 17 Các model còn lại 26](#_Toc110435140)

# GIỚI THIỆU TỔNG QUAN VỀ NODEJS

## Khái niệm NodeJS

* NodeJS là một nền tảng (platform) phía Server side được xây dựng, vận hành trên V8 JavaScript runtime của Chrome giúp xây dựng và phát triển các ứng dụng mạng một cách nhanh chóng và có khả năng mở rộng dễ dàng.
* Node.js ra mắt vào năm 2009, bởi Ryan Dahl và nhận sự bảo trợ từ công ty Joyent (Hoa Kỳ).
* Phần core bên dưới của Node.js hầu hết được viết bằng C++. Chính vì thế mà tốc độ xử lý và hiệu năng khá cao.
* Bên cạnh đó, Node.js còn tạo ra ứng dụng có tốc độ xử lý nhanh với thời gian thực (real-time).
* Node.js cũng được áp dụng để tạo các sản phẩm có lượng truy cập lớn, mở rộng nhanh và đổi mới công nghệ… hoặc tạo dự án startup tiết kiệm thời gian nhất.

## Đặc điểm của NodeJS

* **Bất đồng bộ và phát sinh sự kiện** (Non-blocking and Event Driven**):** Tất các các APIs của thư viện Node.js đều bất đồng bộ (non-blocking), NodeJS không cần đợi một API trả về dữ liệu. Server chuyển sang một API khác sau khi gọi nó và có cơ chế riêng để gửi thông báo và nhận phản hồi về các hoạt động của Node.js và API đã gọi.
* **Tốc độ nhanh:**  Phần core phía dưới được viết gần như toàn bộ bằng C++ kết hợp Chrome V8 Engine nên tốc độ xử lý công việc của Node.js cực nhanh, nhưng vẫn đảm bảo được tính chuẩn xác.
* **Đơn giản – Hiệu năng cao:** Node.js sử dụng một mô hình luồng đơn luồng (single thread) và các sự kiện lặp (event-loop). Cơ chế sự kiện cho phép phía Server trả về phản hồi theo non-blocking, đồng thời tăng hiệu quả sử dụng. Các luồng đơn cung cấp dịch vụ cho nhiều request hơn hẳn Server truyền thống.
* **Không lưu bộ nhớ đệm** (non buffer): Nền tảng Node.js không có vùng nhớ đệm, tức không cung cấp khả năng lưu trữ dữ liệu buffer.

## NodeJS có thể làm được gì

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

# Giới thiệu Reactjs

## 1.1 Reactjs là gì

ReactJS là một thư viện JavaScript được tạo ra để xây dựng giao diện người dùng có khả năng tương tác tốt và nhanh chóng cho các ứng dụng web và di động. Nó là một thư viện mã nguồn mở, xây dựng dựa trên các component, giao diện người dùng chỉ chịu trách nhiệm cho tầng view của ứng dụng.

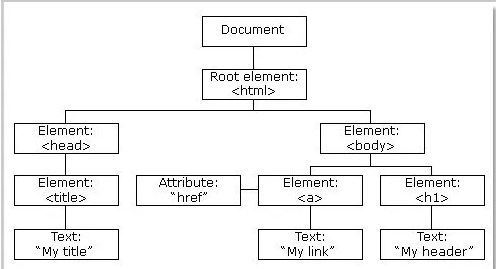
Nó được sử dụng bởi các công ty lớn, đã thành lập và các công ty mới thành lập như: Netflix, Airbnb, Instagram và New York Times..v..v. ReactJS mang lại nhiều lợi thế cho các lập trình viên, khiến nó trở thành một lựa chọn tốt hơn so với các Framework khác như Angular.

Trong kiến ​​trúc Model View Controller (MVC), tầng view chịu trách nhiệm về giao diện của ứng dụng. ReactJS được tạo ra bởi Jordan Walke, một kỹ sư phần mềm tại Facebook.

## **Virtual DOM**

### Mô hình Đối tượng Tài liệu (DOM) là gì?

DOM (Mô hình Đối tượng Tài liệu) coi một file XML hoặc HTML như một cấu trúc cây, trong đó mỗi nút là một đối tượng đại diện cho một phần của tài liệu.



### Virtual DOM là gì

Virtual DOM là một định dạng dữ liệu của JavaScript, với khối lượng nhẹ và được dùng để thể hiện nội dung của DOM – Document Object Model – Mô hình Đối tượng Tài liệu tại một thời điểm nhất định nào đó. Khi trạng thái của một đối tượng thay đổi, VDOM chỉ thay đổi đối tượng đó trong DOM thực thay vì cập nhật tất cả các đối tượng.

### Cách tương tác với nhau Virtual DOM và ReactJS DOM

Khi trạng thái của một đối tượng thay đổi trong ứng dụng ReactJS, VDOM sẽ được cập nhật. Sau đó, nó so sánh trạng thái trước đó của nó và sau đó chỉ cập nhật các đối tượng đó trong DOM thực thay vì cập nhật tất cả các đối tượng.  
Điều này làm cho mọi thứ hoạt động nhanh chóng.

## Performance

ReactJS sử dụng VDOM, giúp các ứng dụng web chạy nhanh hơn nhiều so với những ứng dụng được phát triển với các Framework front-end khác.

ReactJS chia giao diện người dùng phức tạp thành các thành phần riêng lẻ, cho phép nhiều người dùng làm việc trên từng thành phần đồng thời, do đó đẩy nhanh thời gian phát triển cũng như cải thiện hiệu năng của ứng dụng.

## **Component**

ReactJS được xây dựng xoay quanh các component, trong khi đó, các Framework khác dùng template. Để tạo ra một component có đầy đủ những đặc tính, bạn chỉ cần sử dụng phương thức createClass dùng để nhận một tham số mô tả đặc tính.

## Điểm mạnh

* **Dễ dàng tạo các ứng dụng động**
* **Cải thiện hiệu năng** **với Virtual DOM**
* **Các component có thể tái sử dụng**
* **Dễ dàng viết component dễ dàng hơn**

# Tìm hiểu về socketIo

## **Cùng nhìn lại về websocket**

* Trong trình duyệt mà được hỗ trợ WebSockets protocol, một kết nối giữa server và browser được tạo ra quanh HTTP và được gọi là một “HTTP hanshake” (cái bắt tay). Một kết nối được tạo ra trình duyệt và server mở ra một cổng giao tiếp liên tục thông qua một TCP socket. Nó sẽ hỗ trợ cho cả việc gửi và truy vấn message trên một cổng kết nối. Điều này giúp server load ít hơn, giảm số message bị trễ, “and unify PUSH communication using standalone connection”.
* Tuy nhiên WebSocket mắc phải vấn đề đó là HTTP proxies, firewall và hosting provider. Khi Websocket sử dụng một phương thức giao tiếp ngoài HTTP, một phần nhiều trong số đó chưa được hỗ trợ và block bất cứ kết nối socket nào. Vấn đề này chỉ được giải quyết khi sử dụng thư viện trừu tượng mà có thể dễ dàng thay đổi giữa các giao thức dựa trên resoures có sẵn.

## Socket io là gì?

* Là một module của NodeJs
* Được xây dựng nhằm mục đích tạo ra real time NodeJS application. [Socket.io](http://socket.io/) cung cấp cho lập trình viên các đặc trưng như event, room và tự động phục hồi lại kết nối.
* Khi chúng ta include [Socket.io](http://socket.io/) module vào trong ứng dụng của mình nó sẽ cung cấp cho chúng ta hai object đó là: socket server quản lý functionality phía server và socket client điều khiển funtionality phía client.
* Khi client muốn kết nối tới [Socket.io](http://socket.io/) server, nó sẽ gửi cho server một “handshake HTTP request”. Server sẽ phân tích request đó với những thông tin cần thiết trong suốt quá trình kết nối. Nó sẽ tìm cấu hình của middleware mà đã được đăng ký với server và thực thi chúng trước khi đưa ra sự kiện kết nối. Khi kết nối thành công thì connection event listener được thực thi, tạo ra một instance mới của socket có thể coi như định danh của client mà mỗi một client kết nối tới sẽ có 1 định danh. Các bạn có thể thấy rõ khi xem hình dưới đây
* Một module khác của Node.js là LightStreamer-adapter cũng có tạo các kết nối từ client tới server nhưng không trực tiếp mà thông qua LightStreamer Server, đó là các máy chủ theo thời gian thực và nằm ngoài tiến trình của Node.js Server

## Socket io có những ưu điểm

### Bảo mật cao

Socket io được xây dựng dựa trên Engine.IO. Nó sẽ khởi chạy phương thức long-polling trước nhất để kết nối. Sau đó nó mới sử dụng các phương thức giao tiếp tốt hơn như là Websocket chẳng hạn. Vì được thiết lập chặt chẽ như vậy nên khi socket io xuất hiện nó sẽ tự động tạo những kết nối bảo mật như là: proxy và cân bằng tải hoặc là tường lửa cá nhân và phần mềm chống vi rút.

### Kết nối tự động tới server

Đặc điểm tự động kết nối đến server của socket io là gì? Giả sử trong quá trình khởi chạy bị mất kết nối giữa client và server thì socket io sẽ tự động gắn kết nối mãi mãi cho đến khi nào server phản hồi lại. Và đây là tính năng có thể tùy chỉnh được nên bạn có quyền chọn không kết nối tự động đến bất kỳ server nào mà mình muốn.

### Mã hóa nhị phân

Socket io có thể hỗ trợ mã hóa nhị phân như ArrayBuffer và Blob trên trình duyệt hoặc là ArrayBuffer và Buffer trong Node.js.

### Cho phép tạo kênh và phòng

Đặc điểm cho phép tạo kênh và phòng của Socket.io là gì? Có thể nói đây là 1 tính năng khá nổi bật khi mà socket io có thể tạo ra mối quan hệ giữa các phần hoặc các module riêng lẻ bằng cách tạo ra những kênh riêng biệt khác nhau. Ngoài việc tạo kênh, nó còn hỗ trợ tạo phòng cho các clients tham gia với mục đích gửi thông báo đến 1 nhóm người dùng được kết nối với 1 số thiết bị nào đó chẳng hạn. Trong đó có thể kể đến 1 số các API đơn giản như hình dưới đây.

# **Giới thiệu về My SQL**

## **SQL là gì??**

**SQL** (**Structured Query Languge)** hay ngôn ngữ truy vấn có cấu trúc) là một loại ngôn ngữ máy tính phổ biến để tạo, sửa, và lấy dữ liệu từ một hệ quản trị [cơ sở dữ liệu quan hệ](https://vi.wikipedia.org/wiki/C%C6%A1_s%E1%BB%9F_d%E1%BB%AF_li%E1%BB%87u_quan_h%E1%BB%87). Ngôn ngữ này phát triển vượt xa so với mục đích ban đầu là để phục vụ các hệ quản trị cơ sở dữ liệu đối tượng-quan hệ. Nó là một tiêu chuẩn **ANSI/ISO**.

## **Hệ quản trị cơ sở dữ liệu:**

Một hệ quản trị cơ sở dữ liệu (tiếng Anh: Database Management System, viết tắt DBMS) là một chương trình máy tính (một bộ các chương trình) được thiết kế để quản lý một cơ sở dữ liệu, một tập hợp dữ liệu lớn có cấu trúc, phục vụ cho các yêu cầu về dữ liệu của một số lượng lớn người sử dụng.  
Ví dụ điển hình của hệ quản trị cơ sở dữ liệu bao gồm kế toán, nguồn nhân lực và hệ thống hỗ trợ khách hàng. Đầu tiên, hệ quản trị cơ sở dữ liệu chỉ có ở các công ty lớn với đầy đủ phần cứng cần thiết hỗ trợ cho một tập hợp dữ liệu lớn. Hệ quản trị cơ sở dữ liệu. Gần đây, nó đã trở thành một phần tiêu chuẩn của bất kỳ công ty nào.  
Có rất nhiều hệ quản trị CSDL như (SQL Server của Microsoft, MySQL của Oracle, ... ), nhưng trong bài viết này, chúng ta cùng tìm hiểu MySQL

## **Giới thiệu về hệ quản trị dữ liệu MySQL:**

MySQL là hệ quản trị cơ sở dữ liệu tự do nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng. Vì MySQL là hệ quản trị cơ sở dữ liệu tốc độ cao, ổn định và dễ sử dụng, có tính khả chuyển, hoạt động trên nhiều hệ điều hành cung cấp một hệ thống lớn các hàm tiện ích rất mạnh. Với tốc độ và tính bảo mật cao, MySQL rất thích hợp cho các ứng dụng có truy cập CSDL trên internet. Người dùng có thể tải về MySQL miễn phí từ trang chủ. MySQL có nhiều phiên bản cho các hệ điều hành khác nhau: phiên bản Win32 cho các hệ điều hành dòng Windows, Linux, Mac OS X, Unix, FreeBSD, NetBSD, Novell NetWare, SGI Irix, Solaris, SunOS,..  
MySQL là một trong những ví dụ rất cơ bản về Hệ Quản trị Cơ sở dữ liệu quan hệ sử dụng Ngôn ngữ truy vấn có cấu trúc (SQL).  
MySQL được sử dụng cho việc bổ trợ NodeJs, PHP, Perl, và nhiều ngôn ngữ khác, làm nơi lưu trữ những thông tin trên các trang web viết bằng NodeJs, PHP hay Perl,...

# Tìm hiểu về oath 2.0:

## OAuth2 là gì?

* Đầu tiên chúng ta cùng tìm hiểu OAuth là gì ? OAuth là viết tắt của Open Authentication hoặc Open Authorization, là một phương thức xác thực các ứng dụng có thể chia sẻ tài nguyên với nhau mà không cần chia sẻ thông tin về username và password như những cách truyền thống.
* Và như ở trên mình đã nói OAuth là viết tắt của Open Authentication hoặc Open Authorization. Với nghĩa Open Authentication nghĩa là xác thực người dùng thông qua việc đăng nhập. Còn với nghĩa là Open Authorization nghĩa là cấp quyền để có thể truy cập vào các resource do đó nếu chúng ta có 1 tài khoản Facebook thì có thể truy cập vào được nhiều ứng dụng khác nhau mà không cần đăng ký một tài khoản mới của ứng dụng đó.

## Một số khái niệm cơ bản

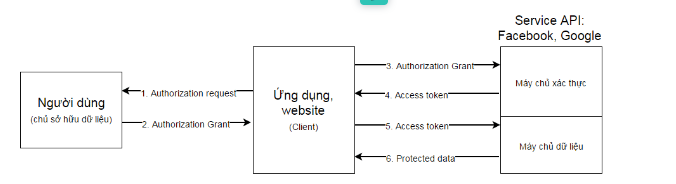
* Resoure Owner: là chủ sở hữu của dữ liệu mà ta muốn chia sẻ. Ví dụ bạn muốn chia sẻ các thông tin như email, ngày sinh, giới tính, địa chỉ cho một trang web nào đó thì chúng ta có thể đăng nhập trang web đó bằng Facebook thì những thông tin như email, ngày sinh, giới tính và địa chỉ là những tài nguyên cần chia sẻ và Resource Owner lúc này chính là bạn
* Resource Server: là server lưu trữ những thông tin mà ta chia sẻ. Server này phải có khả năng nhận và trả lời các yêu cầu (request) truy xuất dữ liệu. Như ở ví dụ trên của chúng ta thì resource server chính là Facebook.
* Client: Là những ứng dụng muốn sử dụng những tài nguyên mà ta chia sẻ. Và chúng ta có thể thấy là ở ví dụ trên Client sẽ là trang web muốn các thông tin cá nhân của ta.
* Authorization Server: Là đối tượng quyết định việc cấp quyền truy cập vào dữ liệu cho client. Như trong ví dụ trên, đây chính là authorization server của Facebook. Đôi khi resource server và authorization server có thể là một , nhưng về mặt chức năng mà nói, đây là 2 chức năng hoàn toàn riêng biệt.

## OAuth2 hoạt động

* Khi bạn đăng nhập với Facebook (Google) thì trang web sẽ dẫn bạn tới trang của Facebook hoặc Google và liệt kê một số các quyền trang web đó muốn bạn chia sẻ. Như ví dụ ở trên thì website sẽ yêu cầu bạn cung cấp các thông tin như email, ngày sinh, giới tính, địa chỉ. Nếu bạn đồng ý cung cấp những thông tin đó thì mới có thể đăng nhập để sử dụng các dịch vụ của website đó.
* Khi bạn đồng ý cung cấp các quyền đó, Facebook sẽ sinh ra một AccessToken cho các quyền đó và website đấy có thể sử dụng được. Do đó nếu có kẻ xấu muốn đánh cắp tài khoản của bạn từ website kia thì cũng chỉ lấy được những thông tin đó mà không lấy đưoc tài khoản của bạn, không ảnh hưởng tới các ứng dụng khác bạn đang sử dụng.

## Sơ đồ luồng hoạt động của OAuth2

Từ ví dụ trên chúng ta có thể hình dung ra OAuth2 sẽ được hoạt động theo luồng như sơ đồ dưới đây:



Hình 1: Sơ đồ hoạt động của OAth2

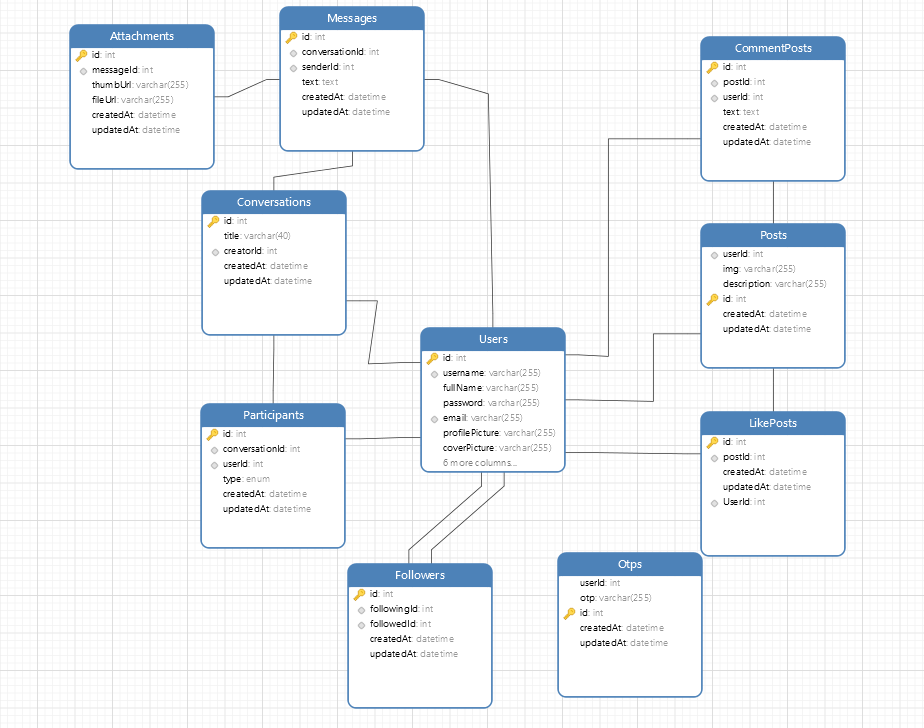
* Ứng dụng yêu cầu ủy quyền để truy cập vào Resource Server (Gmail,Facebook, Twitter hay Github…) thông qua User
* Nếu User ủy quyền cho yêu cầu trên, Ứng dụng sẽ nhận được ủy quyền từ phía User (dưới dạng một token string)
* Ứng dụng gửi thông tin định danh (ID) của mình kèm theo ủy quyền của User tới Authorization Server
* Nếu thông tin định danh được xác thực và ủy quyền hợp lệ, Authorization Server sẽ trả về cho Ứng dụng access\_token. Đến đây quá trình ủy quyền hoàn tất.
* Để truy cập vào tài nguyên (resource) từ Resource Server và lấy thông tin, Ứng dụng sẽ phải đưa ra access\_token để xác thực.
* Nếu access\_token hợp lệ, Resource Server sẽ trả về dữ liệu của tài nguyên đã được yêu cầu cho ứng dụng đó.

## Ưu điểm của OAuth2

* OAuth 2.0 là một giao thức rất linh hoạt dựa trên SSL (Secure Sockets Layer đảm bảo dữ liệu giữa máy chủ web và trình duyệt vẫn giữ được tính riêng tư) để lưu token truy cập của người dùng.
* Cho phép truy cập hạn chế vào dữ liệu của người dùng và cho phép truy cập khi authorization token hết hạn.
* Có khả năng chia sẻ dữ liệu cho người dùng mà không phải tiết lộ thông tin cá nhân.
* Dễ dàng hơn để thực hiện và cung cấp xác thực mạnh mẽ hơn.

# Báo cáo chương trình

## Sơ đồ cơ sở dữ liệu:



Hình 2: Mô hình Diagram

## Các Model dữ liệu trong code:



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| email | String | X | Email của user khi đăng ký tài khoản và dùng để đăng nhập vào website |
| password | String | X | Mật khẩu |
| name | String | X | Họ tên người dùng |
| username | String | x | Tên người dùng của user |
| id | Int | X | Khoá chính |
| fullName | String | x | Họ tên của người dùng |
| profilePicture | String |  | Ảnh đại diện của người dùng |
| coverPicture | String |  | Ảnh bìa của người dùng |
| isAdmin | Tinyint(1) | x | Role của người dùng |
| description | String |  | Mô tả của người dùng |
| city | String |  | Địa chỉ của người dùng |
| country | String |  | Quốc giá của người dùng |

Hình 3 model của user



Hình 4 các mối liên kết của user



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| userId | Int | X | Id của người dùng viết bài |
| id | Int | X | Khoá chính của bài viết |
| img | String |  | Ảnh của bài viết |
| description | String |  | Mô tả của bài viết |

Hình 5 Model của Post



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| userId | Int | X | Id của người dùng viết bình luận |
| id | Int | X | Khoá chính của bình luận bài viết |
| postId | Int | x | ID của bài viết |
| text | String | x | Nội dung của bình luận |

Hình 6 Model của Comment



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| UserId | Int | X | Id của người dùng thích bài viết |
| id | Int | X | Khoá chính của lượt like |
| postId | String |  | ID của bài viết |

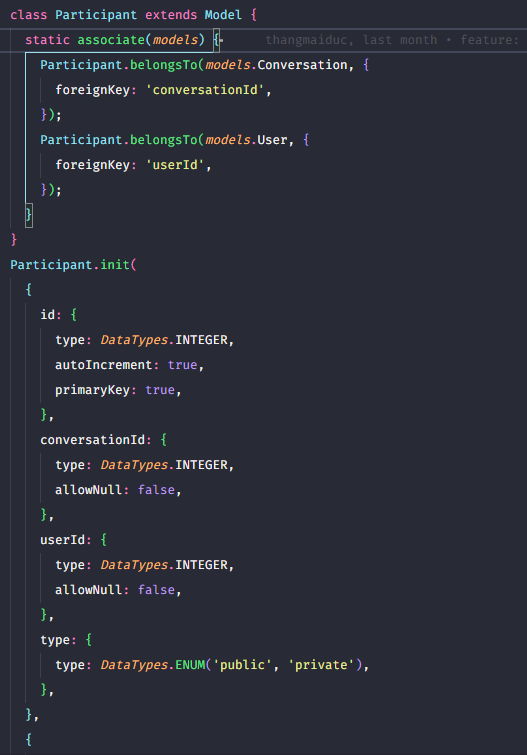
Hình 7 Mô hình của Like Post



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| creatorId | Int | X | Id của người tạo cuộc trò chuyện |
| id | Int | X | Khoá chính của cuộc trò chuyện |
| title | String |  | Tên cuộc trò chuyện |

Hình 8 Model của conversation



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| userId | Int | X | Id của người tham gia cuộc trò chuyện |
| id | Int | X | Khoá chính |
| conversationId | Int | x | ID của cuộc trò chuyện mà họ tham gia |
| type | String | x | Kiểu người tham gia |

Hình 9 Model của participant



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| senderId | Int | X | Id của người dùng gửi tin nhắn |
| id | Int | X | Khoá chính của tin nhắn |
| conversationId | Int | x | Id cuộc trò chuyện mà tin nhắn gửi |
| text | String |  | Nội dung tin nhắn |

Hình 10 Model của message



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| messageId | Int | X | Id của tin nhắn |
| id | Int | X | Khoá chính tệp đính kèm |
| thumbUrl | String |  | Đường dẫn của thumb |
| fileUrl | String |  | Đường dẫn của file |

Hình 11 Model của attachment



Diễn giải

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| userId | Int | X | Id của người dùng nhận OTP |
| id | Int | X | Khoá chính |
| otp | String | X | Nội dung OTP |

Hình 12 model của OTP



Diễn giải

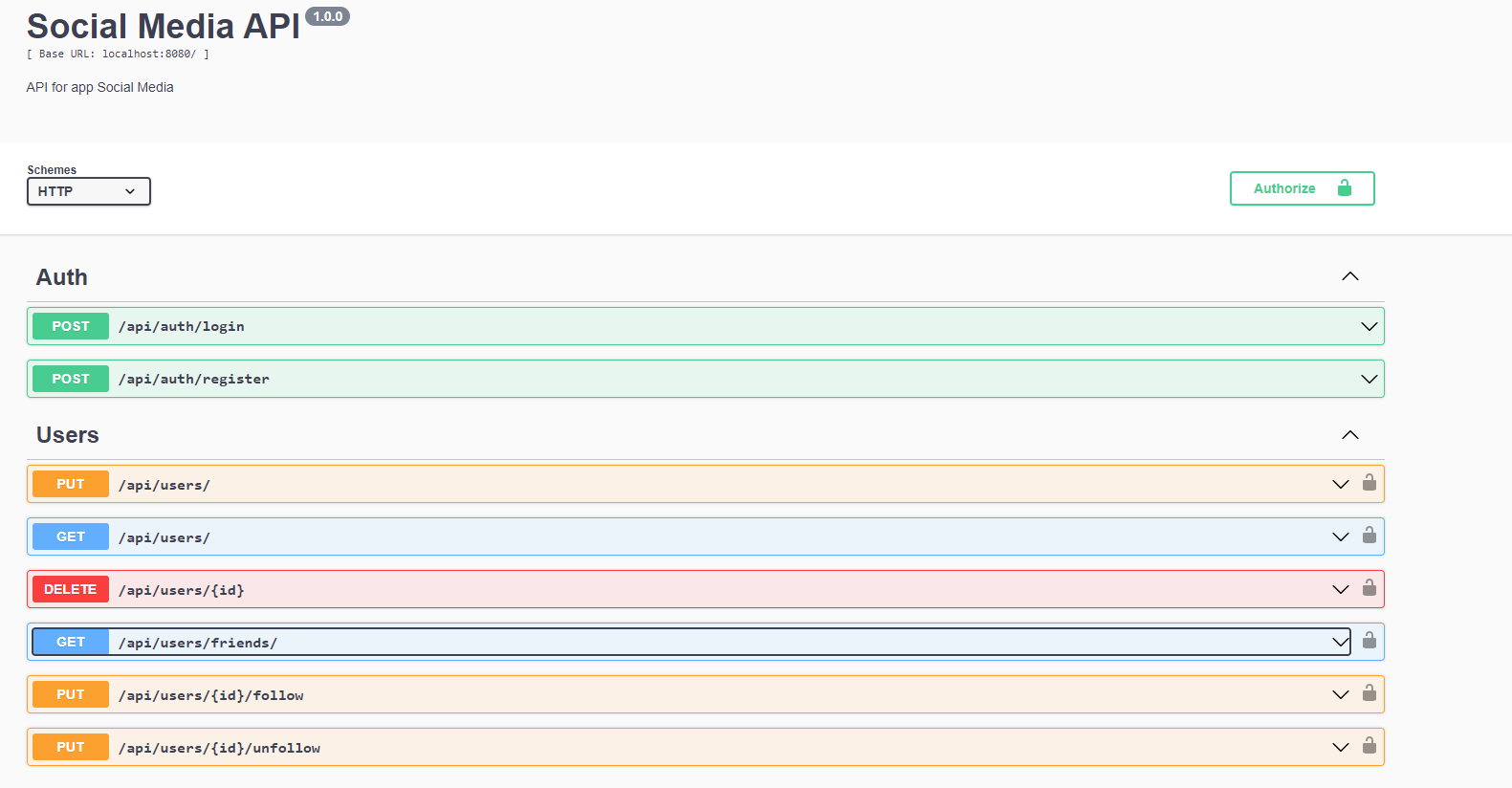
|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Bắt buộc** | **Ý nghĩa** |
| followingId | Int | X | Id của người dùng đang theo dõi |
| id | Int | X | Khoá chính |
| followedId | String | x | Id của người dùng được theo dõi |

Hình 13 Model của follower

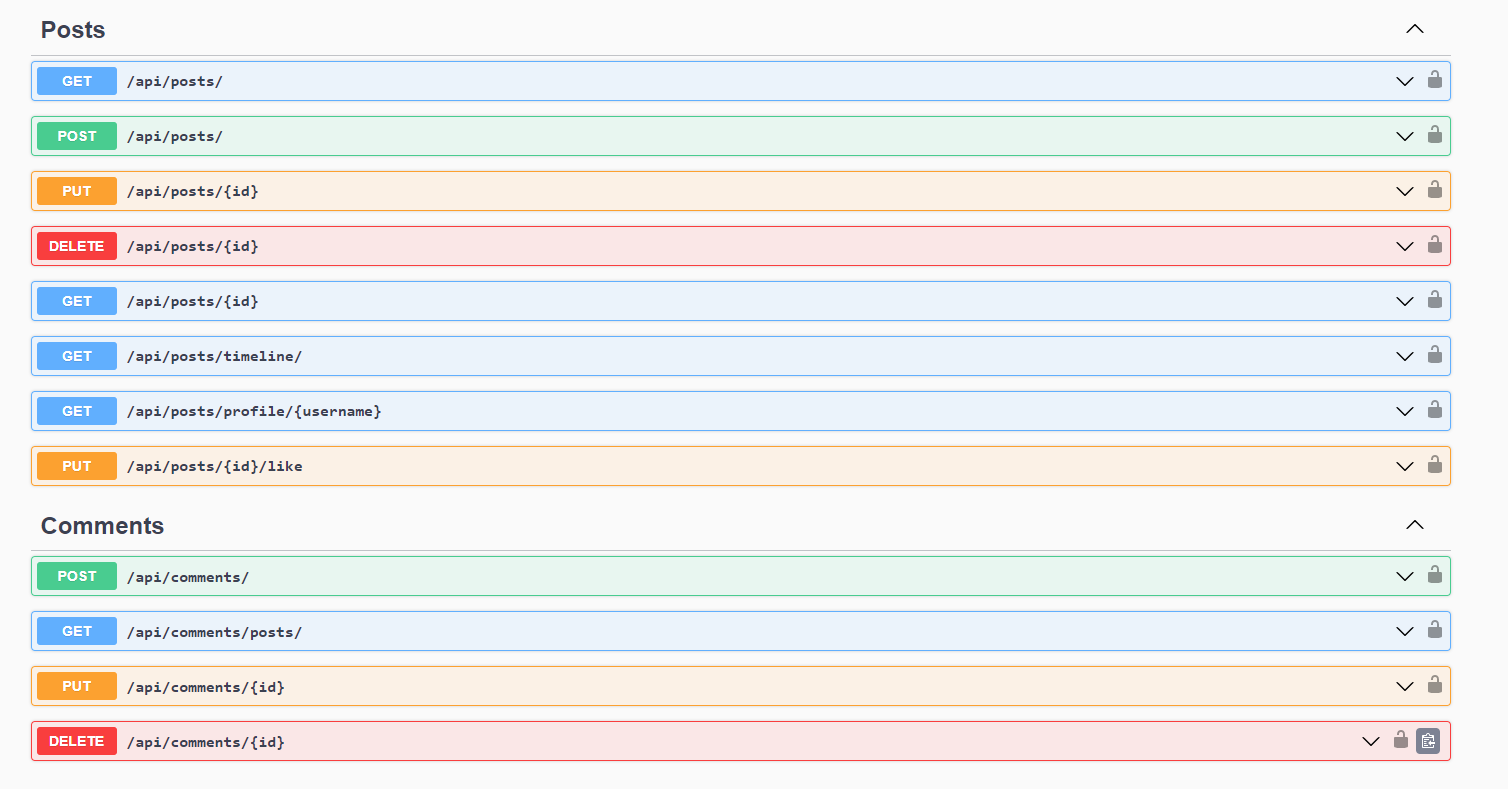
## API:

Em đã hoàn thành đa số các api, còn flow cho chat nhóm và chat cá nhân em đang suy nghĩ. Chưa hoàn thiện.

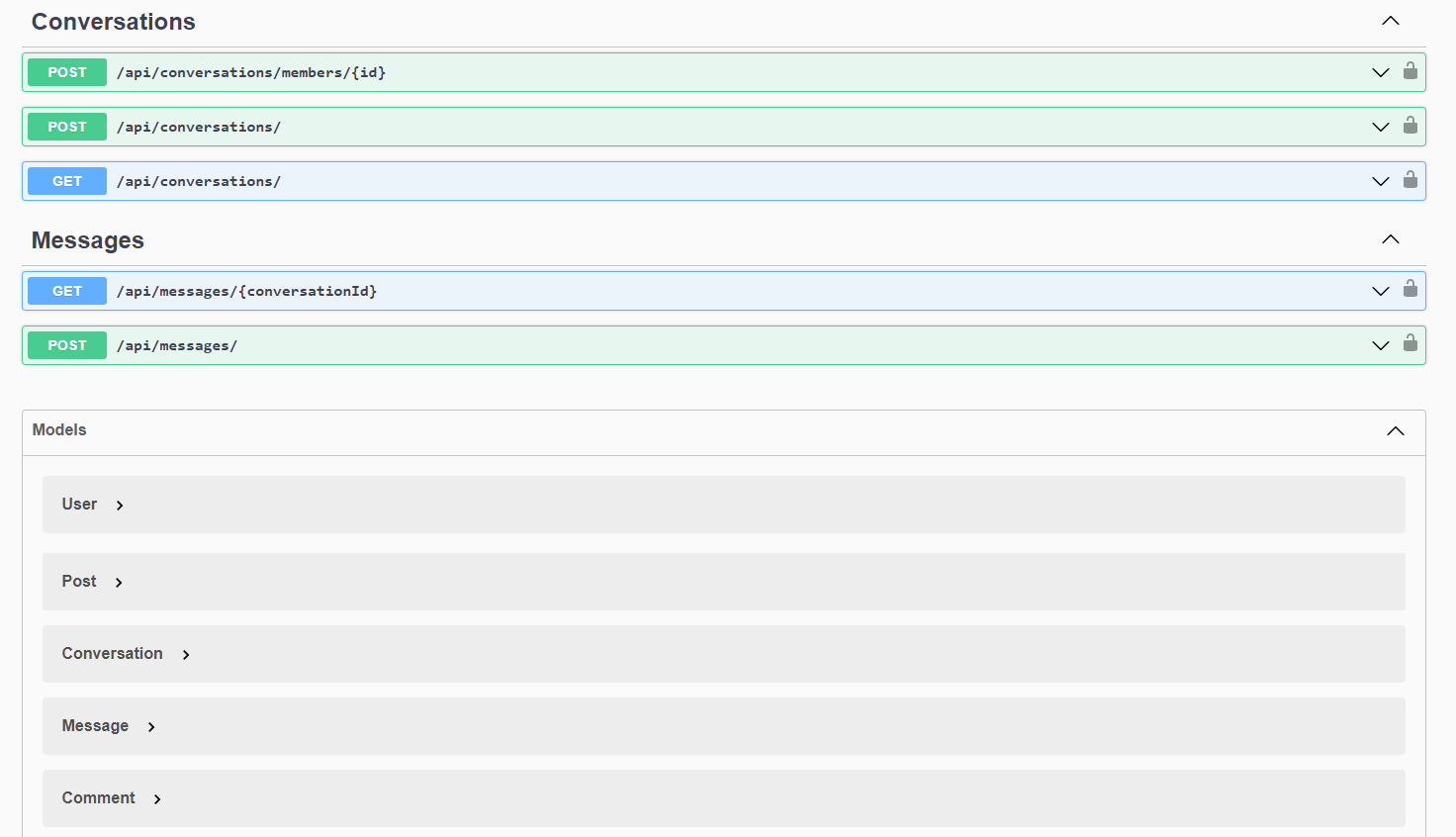
Phần admin chưa làm gì ạ.



Hình 14 Các api của xác thực và user



Hình 15 các api của post và comment



Hình 16 các api của conversation và các model



Hình 17 Các model còn lại

## Frontend

Đăng kí

Đăng nhập

Home

Cá nhân