TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN LẬP TRÌNH WEB NÂNG CAO**

**XÂY DỰNG HỆ THỐNG THÔNG BÁO**

*Người hướng dẫn*: **GV ĐẶNG MINH THẮNG**

*Người thực hiện*: **NGUYỄN HỮU TẤN ĐẠT – 51702075**

**NGÔ MINH HIẾU – 51702017**

Lớp **: 17050201**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN LẬP TRÌNH WEB NÂNG CAO**

**XÂY DỰNG HỆ THỐNG THÔNG BÁO**

Người hướng dẫn: **GV ĐẶNG MINH THẮNG**

Người thực hiện: **NGUYỄN HỮU TẤN ĐẠT**

**NGÔ MINH HIẾU**

Lớp **: 17050201**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

LỜI CẢM ƠN

Trước tiên em xin gửi lời cảm ơn chân thành nhất tới quý thầy khoa Công Nghệ Thông Tin Trường Đại học Tôn Đức Thắng và các thầy giảng dạy bộ môn lập trình web nâng cao đã chỉ dạy tận tình và truyền đạt nhiều kiến thức hơn về lập trình web. Cho chúng em cơ hội áp dụng những kiến thức đã học và tự học để xây dựng 1 hệ thống bao gồm thông báo, quản lí bài viết.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của chúng tôi và được sự hướng dẫn của GV Đặng Minh Thắng;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

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

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

*TP. Hồ Chí Minh, ngày 1 tháng 5 năm 2021*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*DAT*

*Nguyễn Hữu Tấn Đạt*

*HIEU*

*Ngô Minh Hiếu*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

* Giới thiệu về đề tài.
* Giới thiệu 1 số thư viện trong js và các công cụ được sử dụng để xây dụng hệ thống.
* Miêu tả về back-end và các api để gọi đến back-end.
* Miêu tả về front-end và giao diện thực nghiệm.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc71882958)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc71882959)

[TÓM TẮT iv](#_Toc71882960)

[MỤC LỤC 1](#_Toc71882961)

[MỤC LỤC HÌNH ẢNH 2](#_Toc71882962)

[CHƯƠNG 1 – GIỚI THIỆU 4](#_Toc71882963)

[1. GIỚI THIỆU VỀ ĐỀ BÀI 4](#_Toc71882964)

[2. GIỚI THIỆU VỀ THƯ VIỆN SỬ DỤNG 4](#_Toc71882965)

[a) Back-end 4](#_Toc71882966)

[b) Front-end 6](#_Toc71882967)

[CHƯƠNG 2 – CÁCH XÂY DỰNG HỆ THỐNG 7](#_Toc71882968)

[1. XÂY DỰNG BACK-END 7](#_Toc71882969)

[1) User (Người dùng ) 7](#_Toc71882970)

[2) Post (Bài viết ) 20](#_Toc71882971)

[3) Comment (Bình Luận ) 28](#_Toc71882972)

[4) Notify (Thông Báo ) 31](#_Toc71882973)

[2. XÂY DỰNG FRONT-END 36](#_Toc71882974)

MỤC LỤC HÌNH ẢNH

[Hình 1 : API để đăng nhập với google mail 7](#_Toc71883060)

[Hình 2 : Thông tin được server trả về 8](#_Toc71883061)

[Hình 3 : API Đăng ký người dùng là khoa/ phòng ban 9](#_Toc71883062)

[Hình 4 : Kết quả trả về sau khi đăng kí 10](#_Toc71883063)

[Hình 5 : API đăng nhập 11](#_Toc71883064)

[Hình 6 : Sau khi đăng nhập thành công thì server trả về thông tin của tài khoản 12](#_Toc71883065)

[Hình 7 : API Thay đổi mật khẩu đối với khoa 13](#_Toc71883066)

[Hình 8 : Accesstoken và phương thức PUT để đổi mật khẩu 14](#_Toc71883067)

[Hình 9 : Sau khi cập nhật mật khẩu 15](#_Toc71883068)

[Hình 10 : API Đăng nhập với AccessToken 16](#_Toc71883069)

[Hình 11 : Server trả về kết quả sau khi đăng nhập bằng Access Token 17](#_Toc71883070)

[Hình 12 : API cập nhật lại thông tin của người dùng đăng nhập bằng google gmail 18](#_Toc71883071)

[Hình 13 : Thông tin từ API trả về từ Postman 19](#_Toc71883072)

[Hình 14 : API gửi đi để thêm bài viết cho user 20](#_Toc71883073)

[Hình 15 : Đính kèm Access Token vào API thêm bài viết 20](#_Toc71883074)

[Hình 16 : Kết quả trả về sau khi thêm bài viết 21](#_Toc71883075)

[Hình 17 : API Cập nhật lại bài viết 22](#_Toc71883076)

[Hình 18 : Nếu id bài viết không có hoặc access token không đúng 22](#_Toc71883077)

[Hình 19 : Kết quả trả về nếu đúng Id bài viết hoặc đúng Access Token 23](#_Toc71883078)

[Hình 20 : API xóa bài viết 24](#_Toc71883079)

[Hình 21 : API Xóa bài nếu không đúng người viết bài hoặc không tồn tại bài viết 24](#_Toc71883080)

[Hình 22 : API lấy thông tin bài viết với idpost 25](#_Toc71883081)

[Hình 23 : API lấy bài post theo id của user 26](#_Toc71883082)

[Hình 24 : API để lấy tất cả các bài viết trong hệ thống 27](#_Toc71883083)

[Hình 25 : API thêm comment vào bài viết 28](#_Toc71883084)

[Hình 26 : Kết quả thêm comment thành công 28](#_Toc71883085)

[Hình 27 : API dùng để sửa comment của bài viết 29](#_Toc71883086)

[Hình 28 : Kết quả sau khi thực hiện phương thức 30](#_Toc71883087)

[Hình 29 : API được sử dụng để xóa comment 30](#_Toc71883088)

[Hình 30 : API được sử dụng cho phòng/khoa đăng thông báo 31](#_Toc71883089)

[Hình 31 : API để cập nhật lại Thông báo 32](#_Toc71883090)

[Hình 32 : Kết quả trả về sau khi cập nhật 32](#_Toc71883091)

[Hình 33 : API xóa thông báo 33](#_Toc71883092)

[Hình 34 : API để lấy tất cả thông báo của khoa theo id khoa 34](#_Toc71883093)

[Hình 35 : API lấy tất cả thông báo 35](#_Toc71883094)

[Hình 36 : API để lấy thông tin của 1 thông báo theo idNotify 36](#_Toc71883095)

CHƯƠNG 1 – GIỚI THIỆU

1. GIỚI THIỆU VỀ ĐỀ BÀI

* Với tình hình COVID diễn biến ngày càng phức tạp, Khoa Công nghệ thông tin (CNTT) muốn tăng cường sự tương tác giữa các sinh  viên trên nền tảng trực tuyến. Do đó Khoa CNTT muốn cải tiến hệ thống thông báo (Notification) hiện có trong hệ thống thông tin sinh  viên (Student Portal), bổ sung thêm các tính năng giúp sinh viên có thể trao đổi, tương tác trực tuyến và nhận được sự hỗ trợ từ nhà  trường tốt hơn.

Hệ thống này có thể coi như là một mạng xã hội thu nhỏ nơi nhà trường có thể đăng các thông báo, và sinh viên có thể đăng bài lên để trao đổi và chia sẻ

1. GIỚI THIỆU VỀ THƯ VIỆN SỬ DỤNG
2. Back-end

* **Express JS**: là một Framework nhỏ, nhưng linh hoạt được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile
  + Về các package hỗ trợ: Expressjs có vô số các package hỗ trợ nên các bạn không phải lo lắng khi làm việc với Framework này.
  + Về performance: Express cung cấp thêm về các tính năng (feature) để dev lập trình tốt hơn. Chứ không làm giảm tốc độ của NodeJS.
* **Body Parser**: là một module hoạt động như 1 middleware giúp chuyển các dữ liệu từ front-end sẽ được gắn vào body của request mà front-end gửi xuống back-end
* **CORS** : Cross-origin resource sharing là một cơ chế cho phép các tài nguyên của server được sử dụng bởi các tên miền khác với tên miền của trang đó nếu không có module này thì sẽ dễ dẫn đến lỗi thường gặp là :
  + “**no ‘access-control-allow-origin’ header is present on the requested resource**”
  + Đây chính là lỗi CORS policy mà bất cứ lập trình viên nào cũng sẽ gặp phải. Khi call API tới server mà không có header Access-Control-Allow-Origin
* **Moongose** : là một Object Document Mapper (ODM). Điều này có nghĩa là Mongoose cho phép bạn định nghĩa các object (đối tượng) với một schema được định nghĩa rõ ràng, được ánh xạ tới một MongoDB document. Mongoose cung cấp một số lượng đáng kinh ngạc các chức năng cho việc tạo ra và làm việc với các schema. Mongoose hiện có 8 SchemaTypes. Đó là:String, Number, Date, Buffer, Boolean, Mixed, ObjectId, Array.
* **Google-auth-library**: được dùng để lấy thông tin sau khi sử dụng tài khoản gmail để đăng nhập
* **Dotenv**: được sử dụng khởi tạo biến môi trường vi có một số biến có thể thay đổi khi đưa lên các server online hoặc muốn thay đổi chỉ cần vào file .env thay đổi giá trị của biến môi trường đó
* **Socketio**: là một module trong Node.js được phát triển vào năm 2010. Nó được phát triển để sử dụng các kết nối mở để tạo điều kiện giao tiếp thời gian thực, trả về giá trị thực ở tại thời điểm đó. Socket.io cho phép giao tiếp hai chiều giữa máy khách và máy chủ. Giao tiếp hai chiều được bật khi máy khách có Socket.io trong trình duyệt và máy chủ cũng đã tích hợp gói Socket.io.
* **Express router**: được sử dụng để chia server thành các router để giúp cho người lập trình viên dễ quản lí cũng như dễ bảo trì nếu xảy ra lỗi.
* **Jwt (jsonwebtoken)**: được dùng để khỏi tạo ra token id ở trong hệ thống để không dễ bị lộ thông tin trong hệ thống cũng như dễ quản lý bằng cách tạo chữ kí riêng biệt giữa 1 serect key được cài đặt sẵn trong biến môi trường, và 1 chuỗi các objectId và nó là duy nhất trong moongose DB aslat.
* **Axios**: được dùng để gửi 1 request khác đến 1 sever khác có sẵn trên mạng dùng để lưu trữ giá trị ảnh của hệ thống
* **FormData**: được sử dụng để tạo request dưới dạng form data vì server online đấy yêu cầu như thế
* **Express-validator** : là module được sử dụng vào mục đích kiểm tra giá trị mà người dùng gửi requrest có hợp lệ hay không trước khi đưa vào xử lý.
* **Bcrypt** : BCrypt là một thuật toán mã hóa mật khẩu được thiết kế bởi Niels Provos and David Mazières. BCrypt được đánh giá là bảo mật và an toàn hơn so với MD5 và SHA bởi mỗi lần thực hiện băm nó lại cho một giá trị khác nhau, việc này khiến cho việc dò tìm mật khẩu trở nên khó hơn. Được sử dụng để mã hóa mật khẩu của người dùng nhập vào và mỗi lần mã hóa sẽ ra một đoạn mã mới nên cùng 1 mật khẩu có thể bâm ra 2 đoạn mã khác nhau nên trong Bcrypt sẽ có hàm compare để so sánh mật khẩu với đoạn mã đã mã hóa

1. Front-end

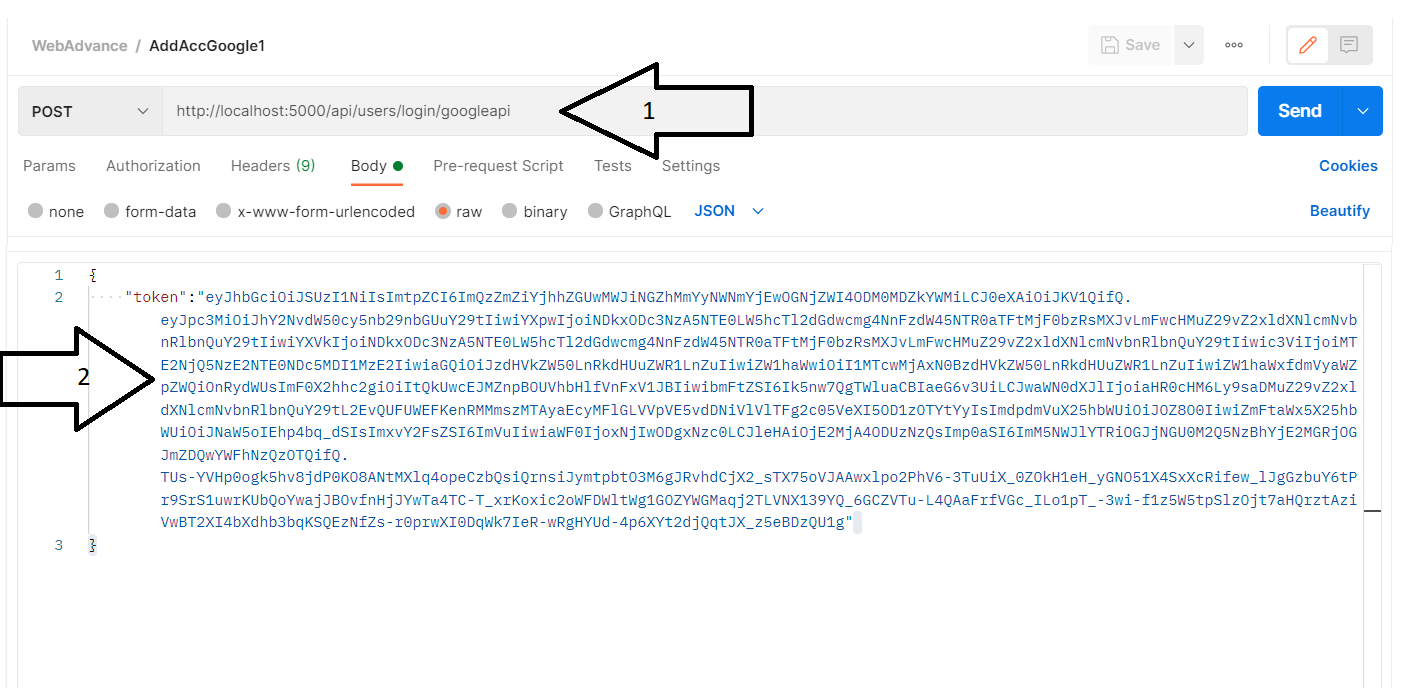
CHƯƠNG 2 – CÁCH XÂY DỰNG HỆ THỐNG

1. XÂY DỰNG BACK-END

* Về phía back-end chúng em xây dựng dựa trên thư viện Express và chia ra các router và dùng postman để gọi đến api mà cách em xây dụng

1. User (Người dùng )

* Sinh viên : Dùng Google Mail
* Khoa , Phòng : Dùng tài khoản được đăng kí bới Admin



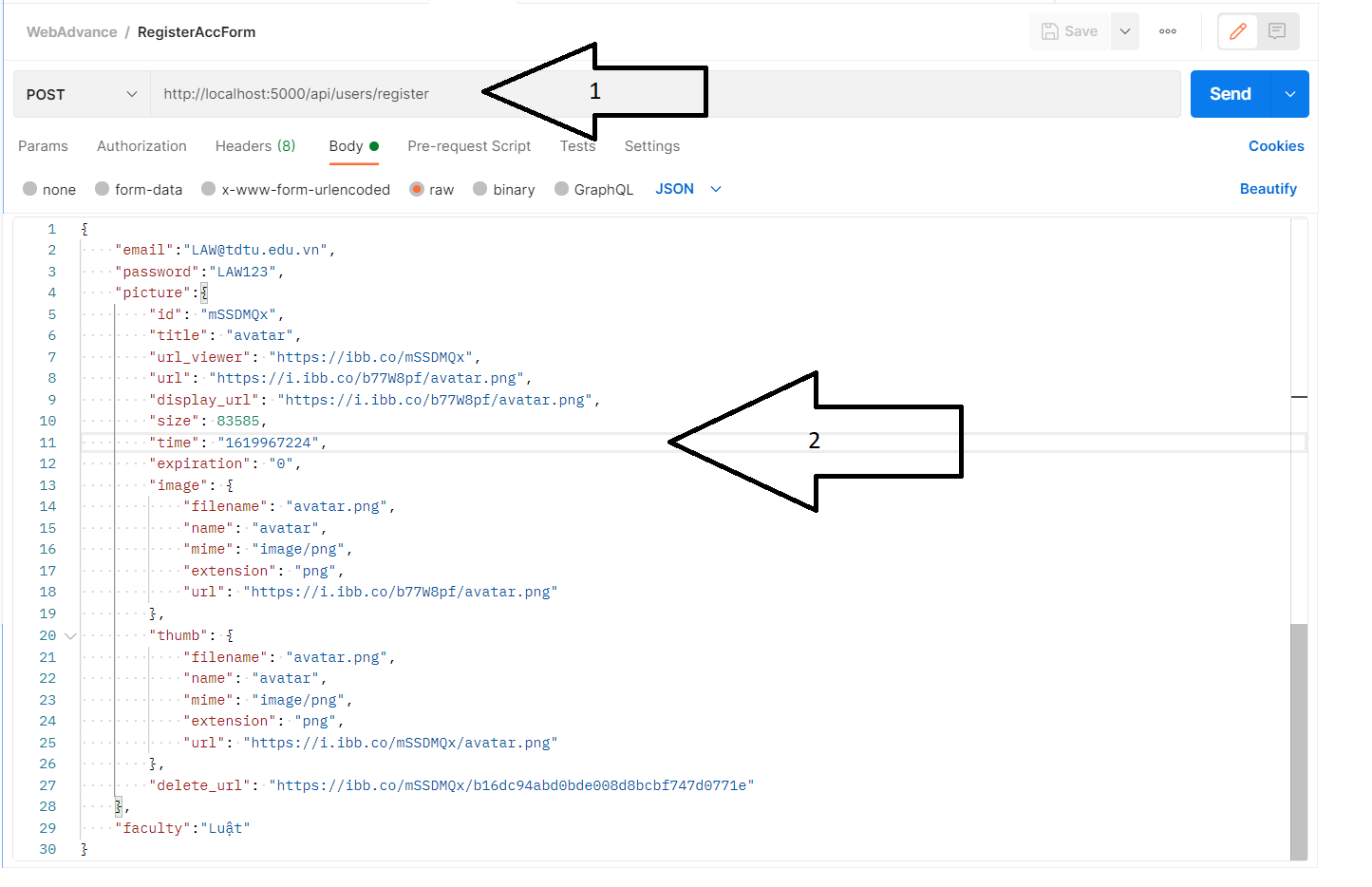
Hình 1 : API để đăng nhập với google mail

* 1: Là đường dẫn đến API : <http://localhost:5000/api/users/login/googleapi>
* Sử dụng phương thức POST
* 2: token là tham số truyền vào cùng với đường dẫn thì để có được token này sẽ sử dụng 1 thư viện trên front-end để lấy tokenId khi đăng nhập bằng account google mail và từ tokenId này ở server có thể lọc được các thông tin cần thiết



Hình 2 : Thông tin được server trả về

* Thông tin trả về bao gồm tên , mail , khoa và lớp (Nếu đăng nhập lần đầu khoa sẽ được để trống), avatar(sẽ được lưu bởi 1 api khác trên mạng )
* Access Token này sẽ được khởi tạo bởi thư viện jsonwebToken được hình thành bằng secret key cùng với \_id của người dùng trong database moongose
* Ngày đầu tiền mà tài khoản lần đầu đăng nhập (ngày khởi tạo tài khoản )



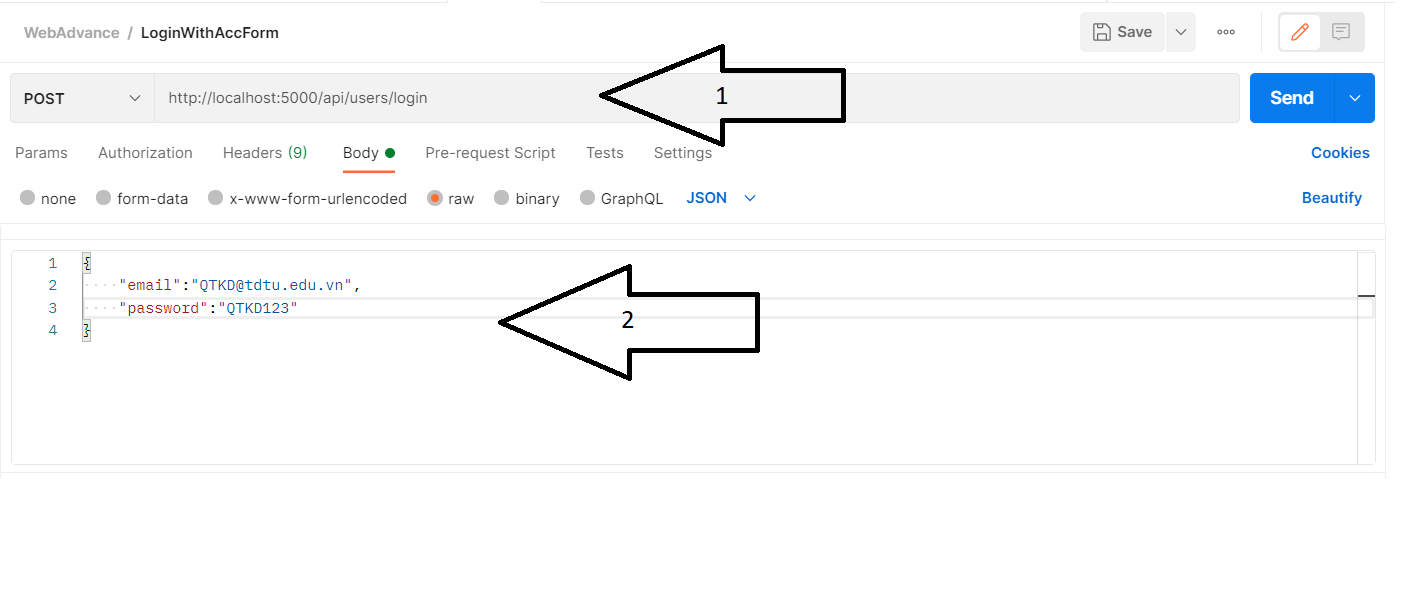
Hình 3 : API Đăng ký người dùng là khoa/ phòng ban

* 1: Là đường dẫn đến API : <http://localhost:5000/api/users/register>
* Sử dụng Phương Thức POST
* 2: Thông tin đăng ký bao gồm email , password , avatar , faculty



Hình 4 : Kết quả trả về sau khi đăng kí

* Thông tin trả về là thông tin được nhập vào bằng form trên front-end và hiện ra thông báo là thêm người dùng thành công
* Kèm theo đó là access token được đính kèm thông tin trả về



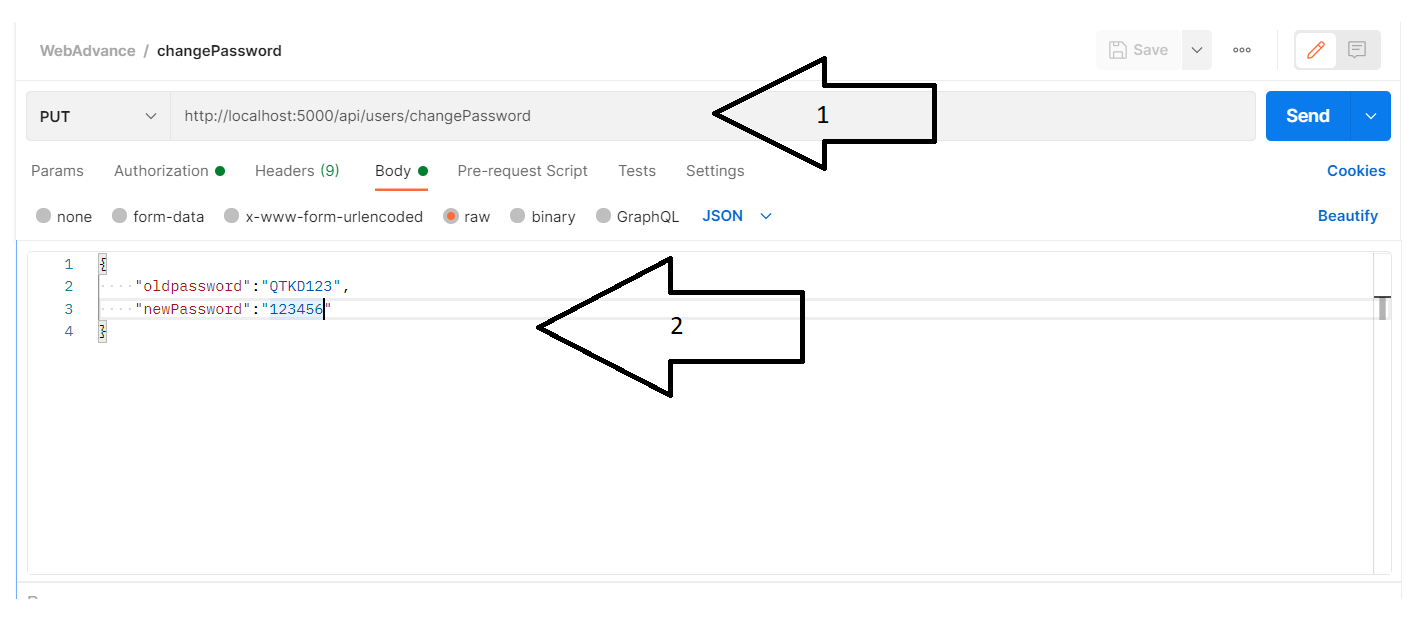
Hình 5 : API đăng nhập

* 1: Là đường dẫn đến API : <http://localhost:5000/api/users/login>
* Sử Dụng Phương thức POST
* 2: Thông tin truyền theo đường dẫn bao gồm email , password ở phần đăng kí



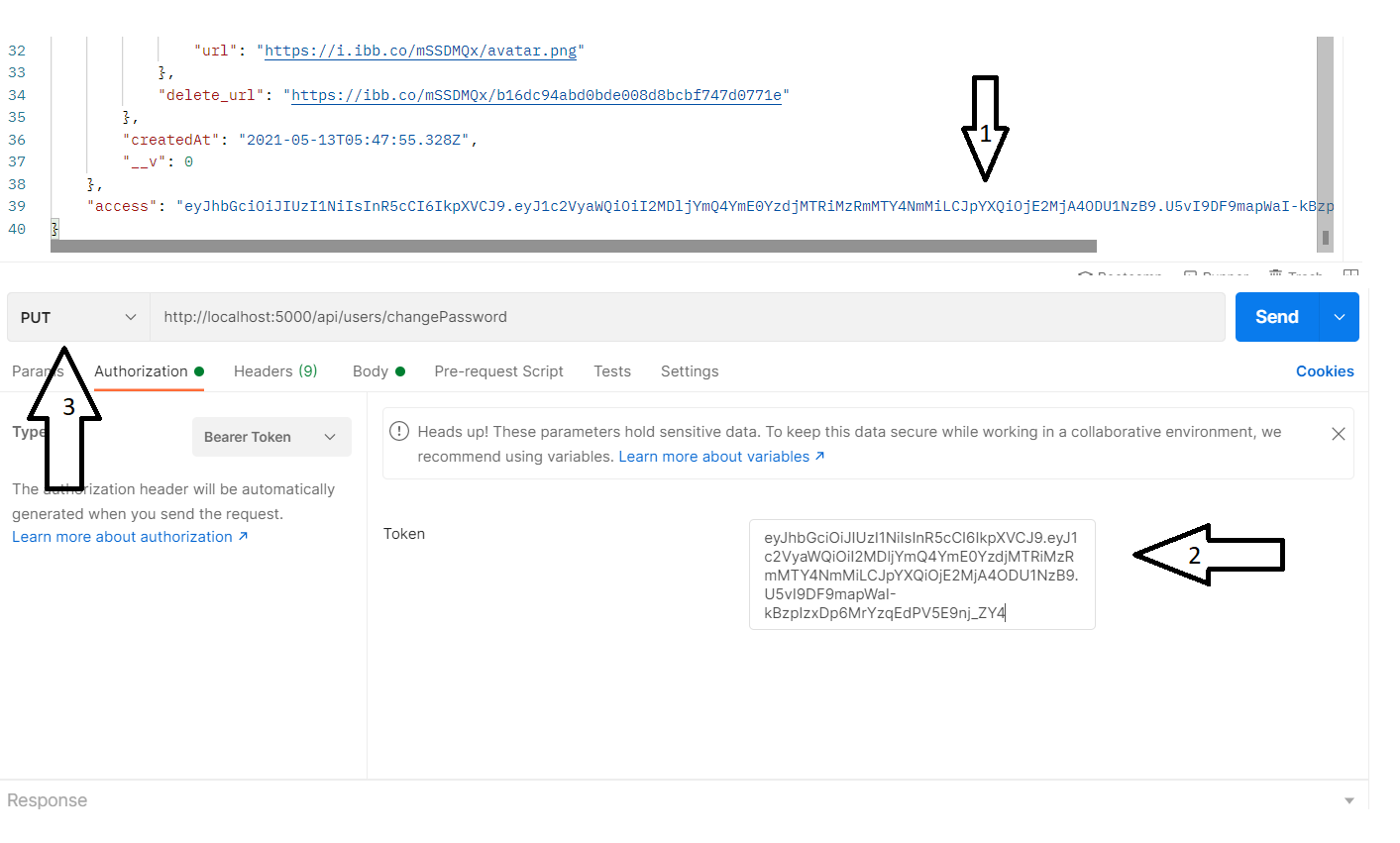
Hình 6 : Sau khi đăng nhập thành công thì server trả về thông tin của tài khoản

* Thông tin trả về gồm email , mật khẩu đã hash sau khi dùng bcrypt, avatar, ngày khởi tạo , Access Token



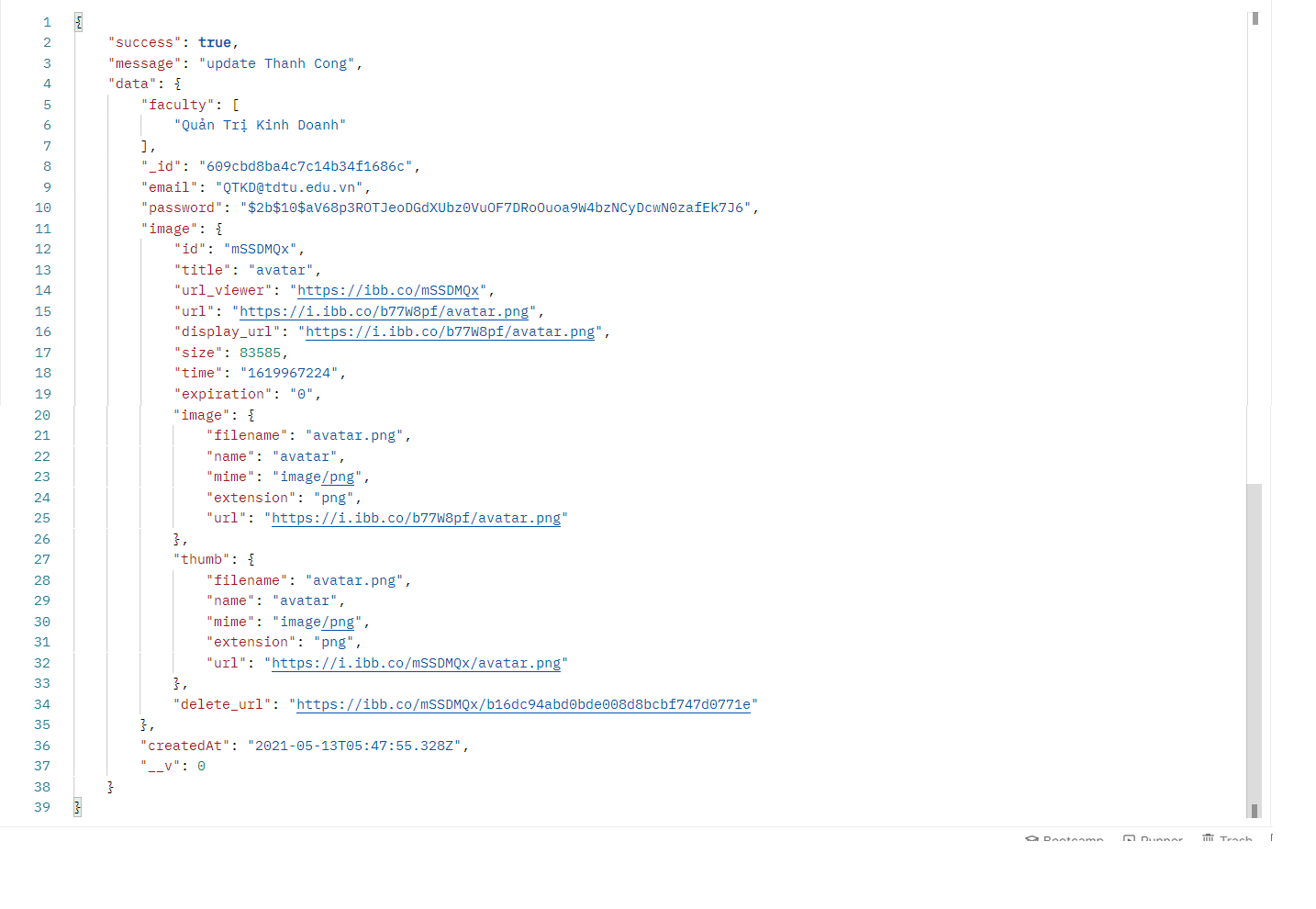
Hình 7 : API Thay đổi mật khẩu đối với khoa

* 1: Là đường dẫn đến API : <http://localhost:5000/api/users/changePassword>.
* Sử dụng phương thức PUT
* 2: Thông tin được đưa đến đường dẫn API bao gồm oldPassword, newPassword kèm theo form là barer token và ta đính token sau khi đăng nhập của user vào trong form và đẩy lên server

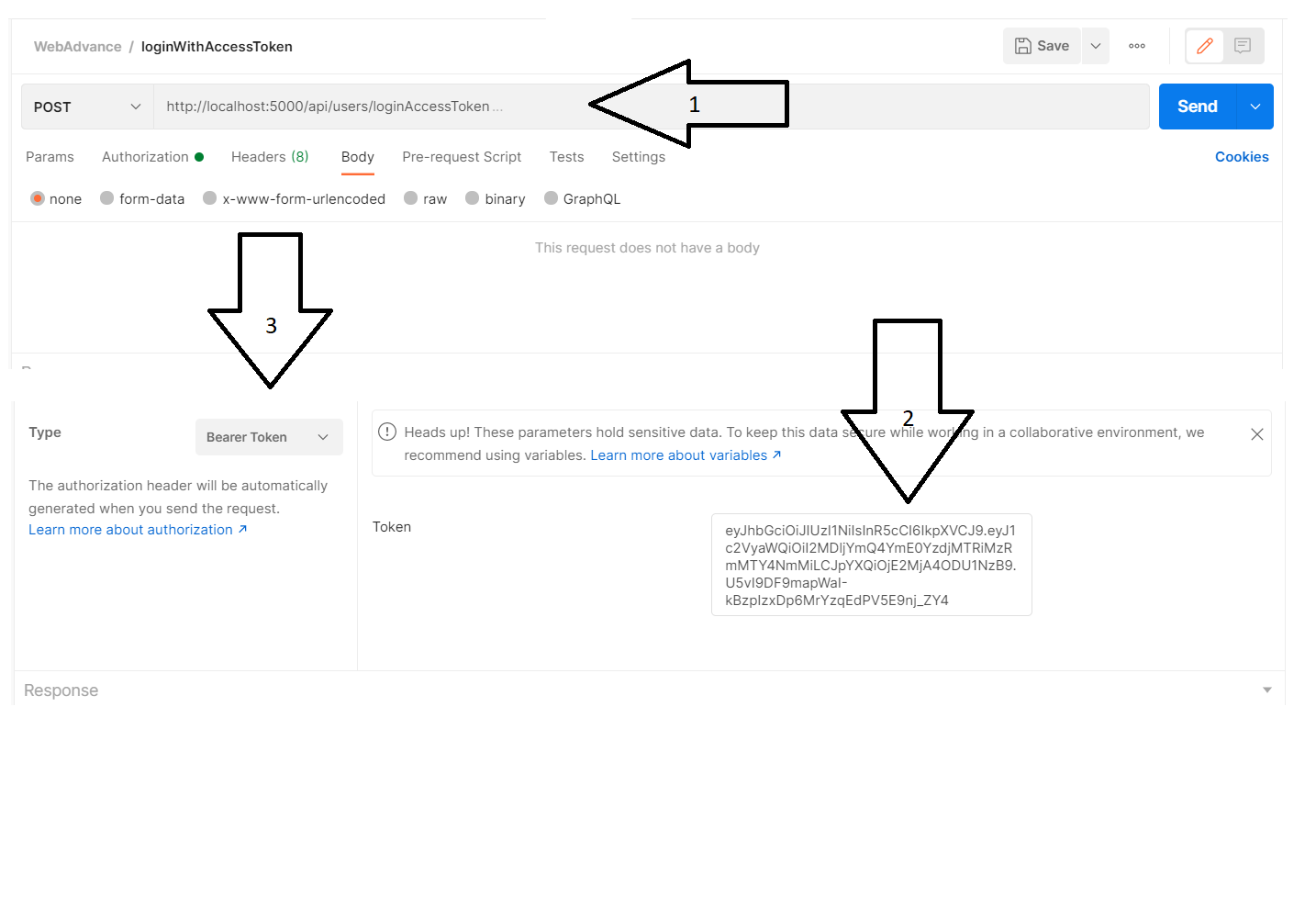


Hình 8 : Accesstoken và phương thức PUT để đổi mật khẩu

* 1: Access Token được đính gửi cùng với thông tin sau khi đăng nhập
* 2: Dán Access Token ở 1 lên qua phần Athorization barer token để gửi
* 3: Dùng phương thức PUT để thay đổi



Hình 9 : Sau khi cập nhật mật khẩu

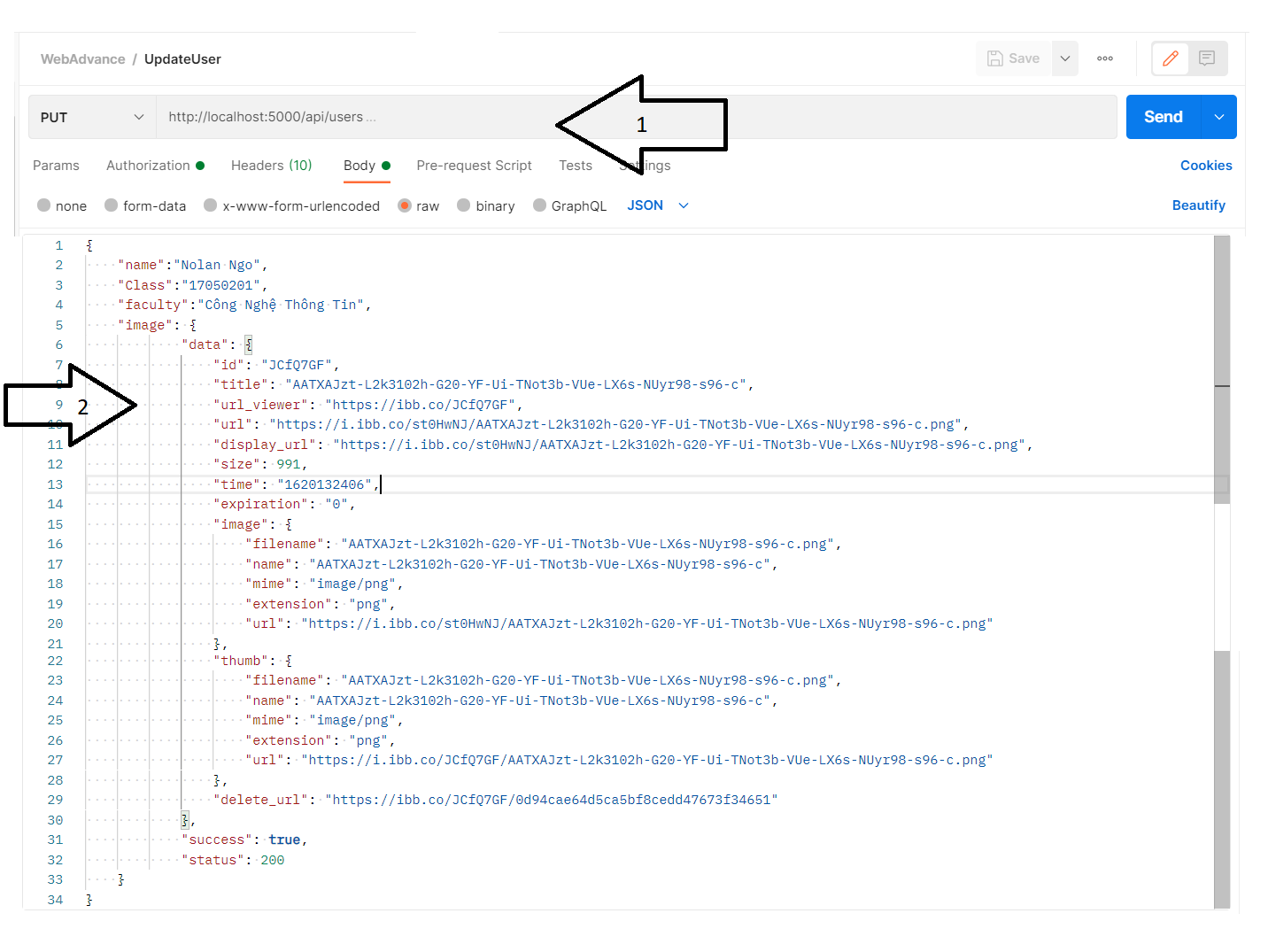


Hình 10 : API Đăng nhập với AccessToken

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/users/loginAccessToken>
* Sử dụng phương thức POST
* 2,3 : Truyền vào barer Token cái token mà khi đăng nhập server đã gửi để lần sau đăng nhập người dùng chỉ cần truyền AccessToken vào là sẽ đăng nhập được



Hình 11 : Server trả về kết quả sau khi đăng nhập bằng Access Token



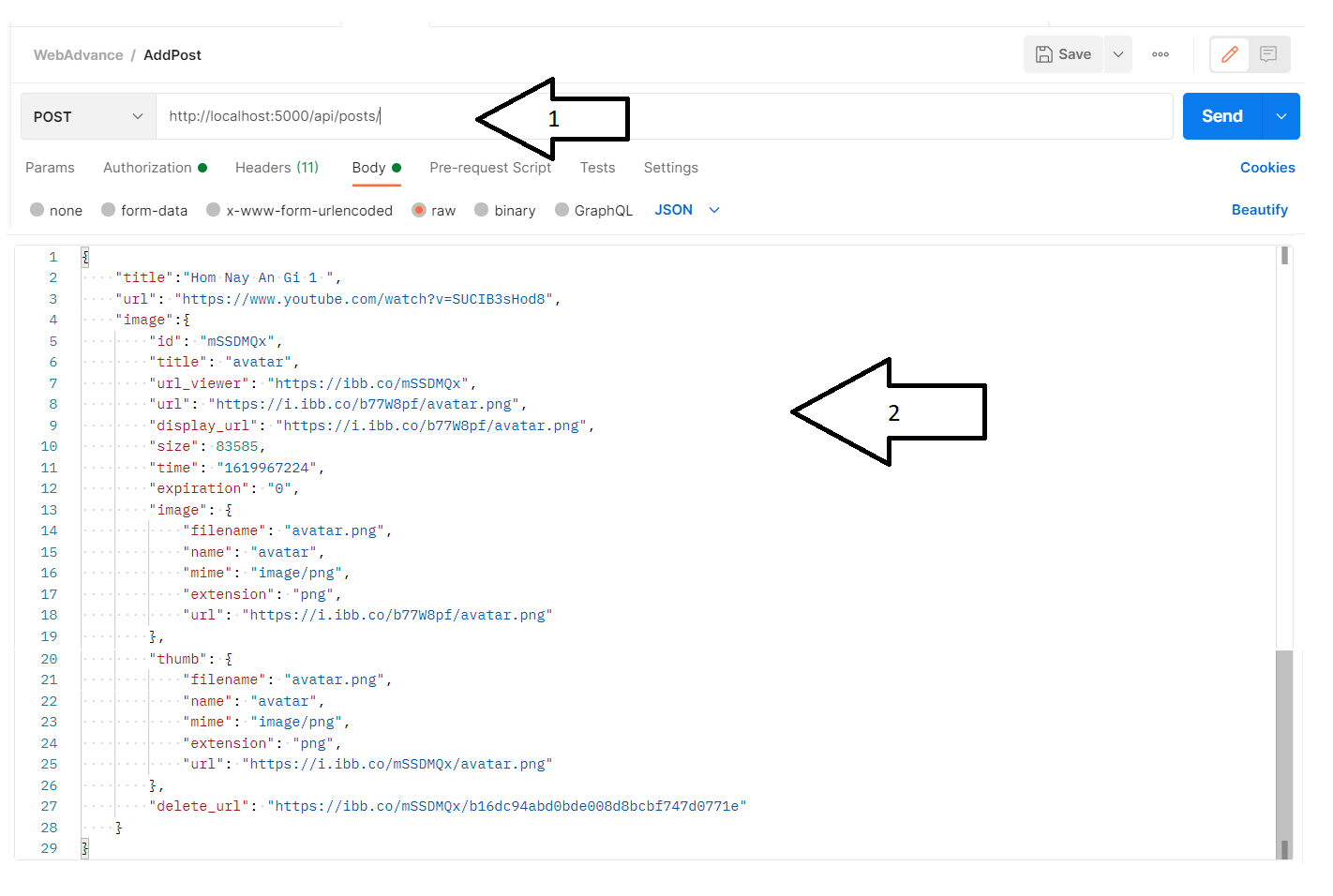
Hình 12 : API cập nhật lại thông tin của người dùng đăng nhập bằng google gmail

* 1 : Là đường dẫn đến API : [http://localhost:5000/api/users/](http://localhost:5000/api/users/n)
* Phương thức là phương thức PUT
* 2 : Thông tin truyền theo form bao gồm name, class , faculty , image(avatar), class , faculty sau khi đăng nhập bằng google gmail thì 2 thông tin này được khởi tạo với không có giá trị nên là sau khi đăng nhập người dùng (Sinh viên) phải cập nhật lại tên lớp , và khoa của mình

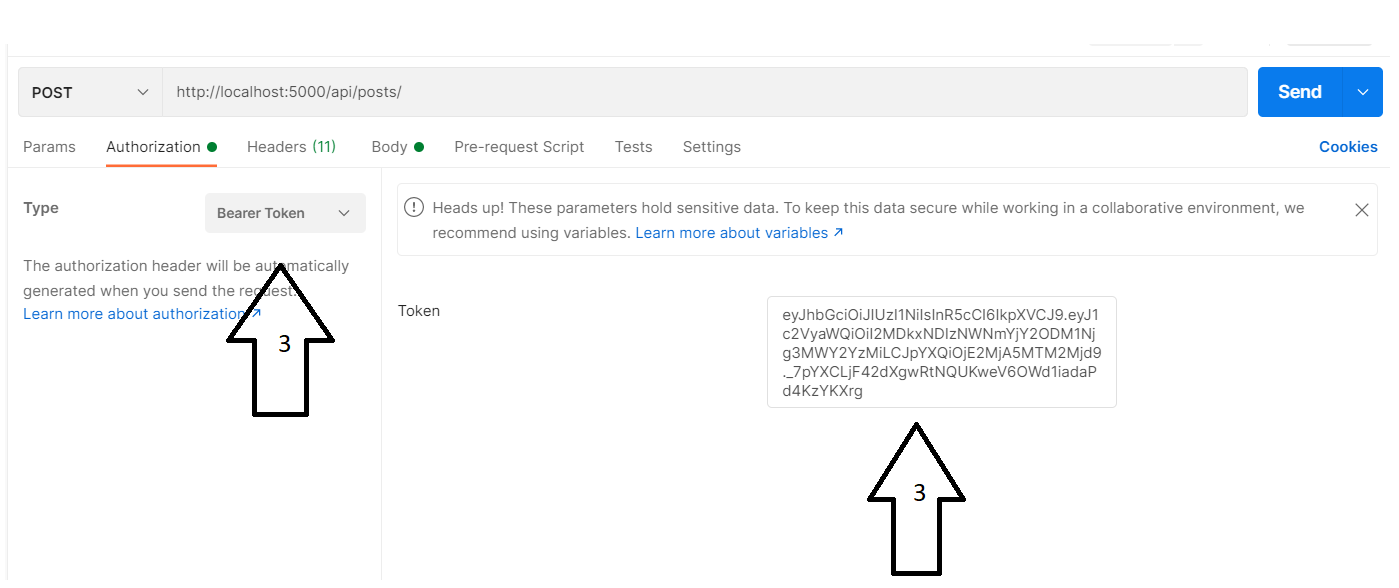


Hình 13 : Thông tin từ API trả về từ Postman

1. Post (Bài viết )



Hình 14 : API gửi đi để thêm bài viết cho user

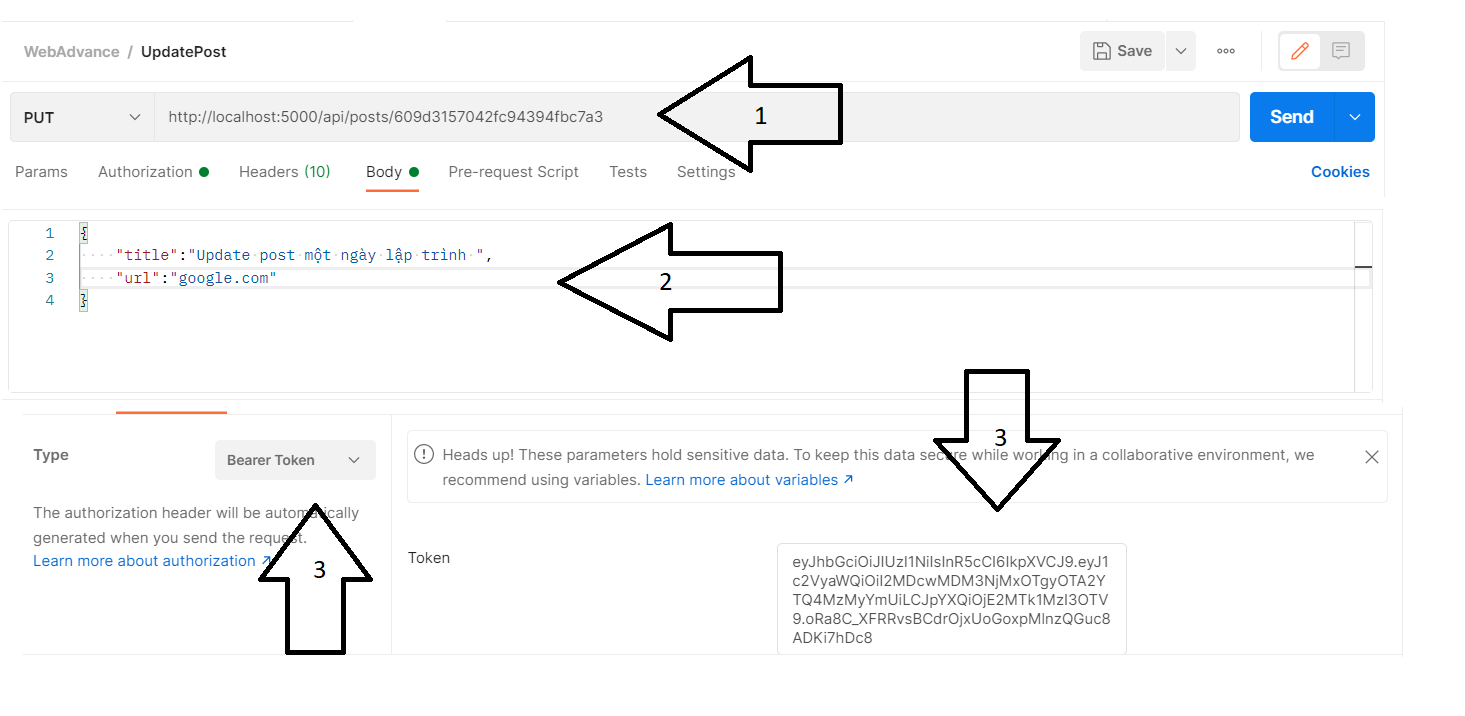


Hình 15 : Đính kèm Access Token vào API thêm bài viết

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/posts/>
* Phương thức để thêm bài viết là phương thức POST
* 2 : Thông tin cần thêm vào để tạo thành công 1 bài viết gồm: title(Nội dung của bài viết ) , url (là đường dẫn từ youtube), image (hình ảnh có thêm vào 1 lần nhiều tấm ảnh ) url hoặc image có thể bỏ trống 1 trong 2 hoặc cả 2 cũng được
* 3 : Đính kèm Access Token sau khi đăng nhập vào barer token và gửi kèm theo request

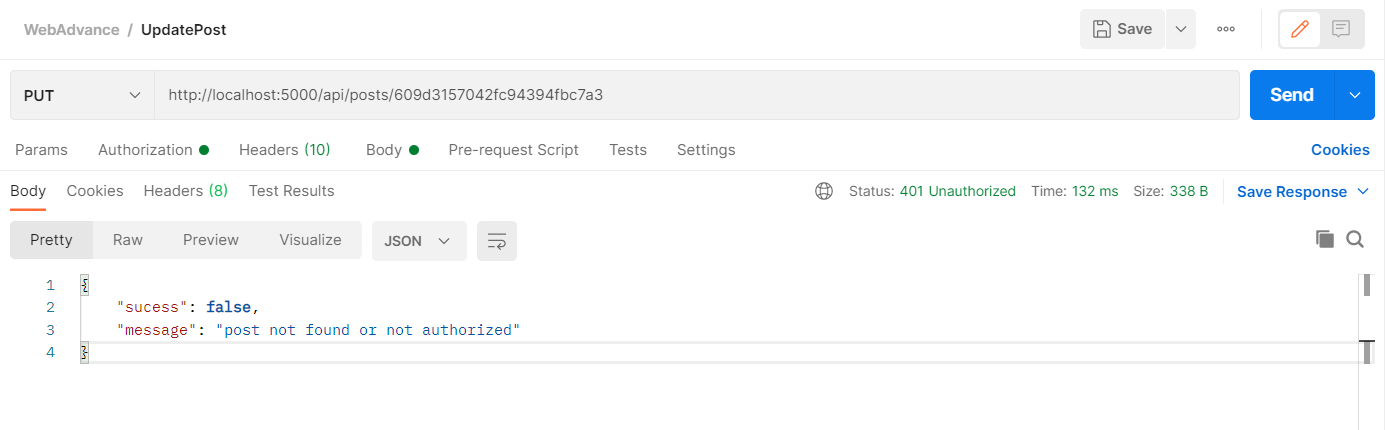


Hình 16 : Kết quả trả về sau khi thêm bài viết



Hình 17 : API Cập nhật lại bài viết

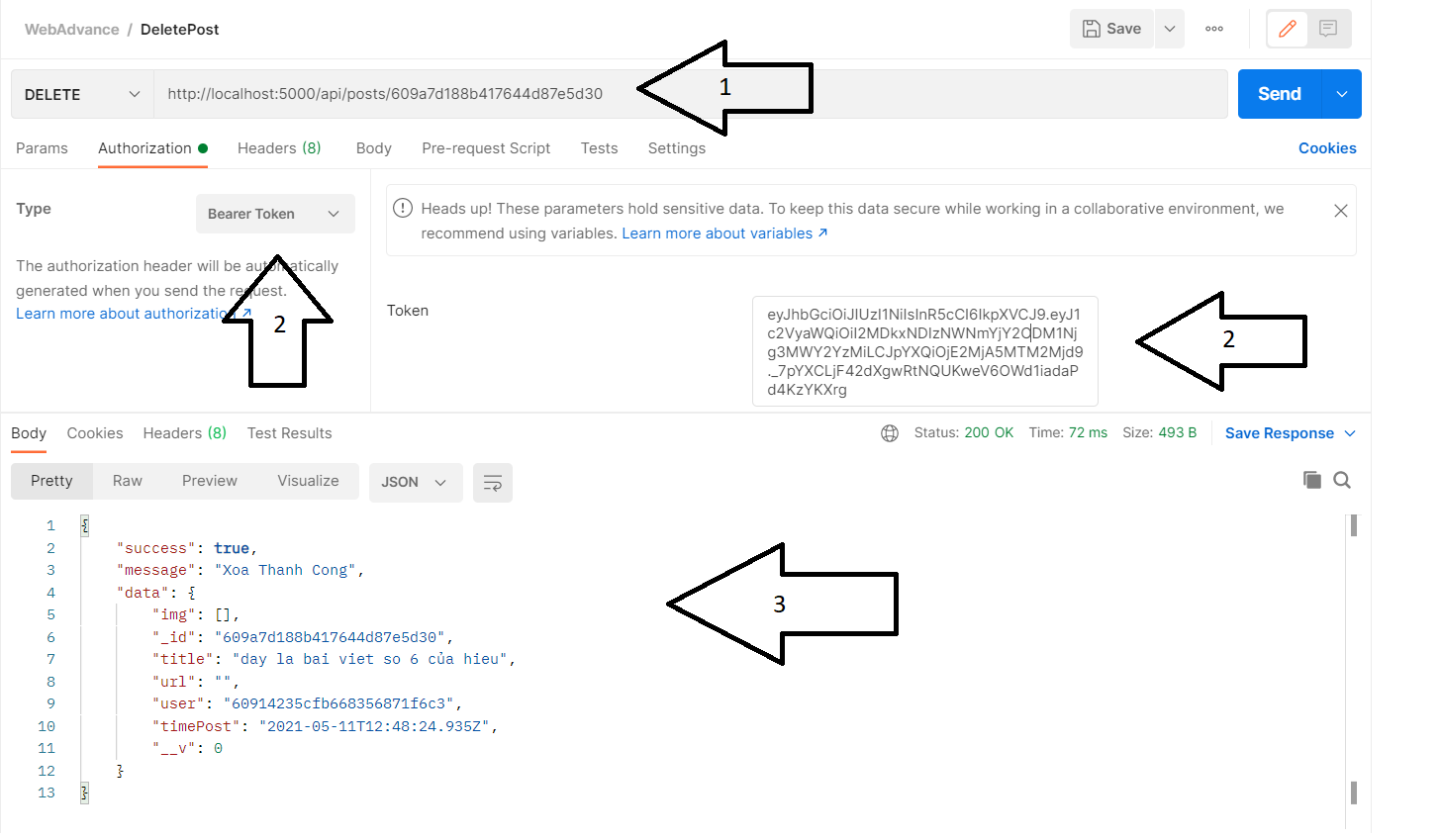
* 1 : Là đường dẫn đến API : [http://localhost:5000/api/posts/idbaiviet](http://localhost:5000/api/posts/idbaiviet/)
* Phương thức dùng để cập nhật là phương thức PUT
* 2 : Thông tin cập nhật
* 3 : Đính Access token vào barer token để xác nhận đúng phải bài của người đó đăng hay không để cho phép cập nhật



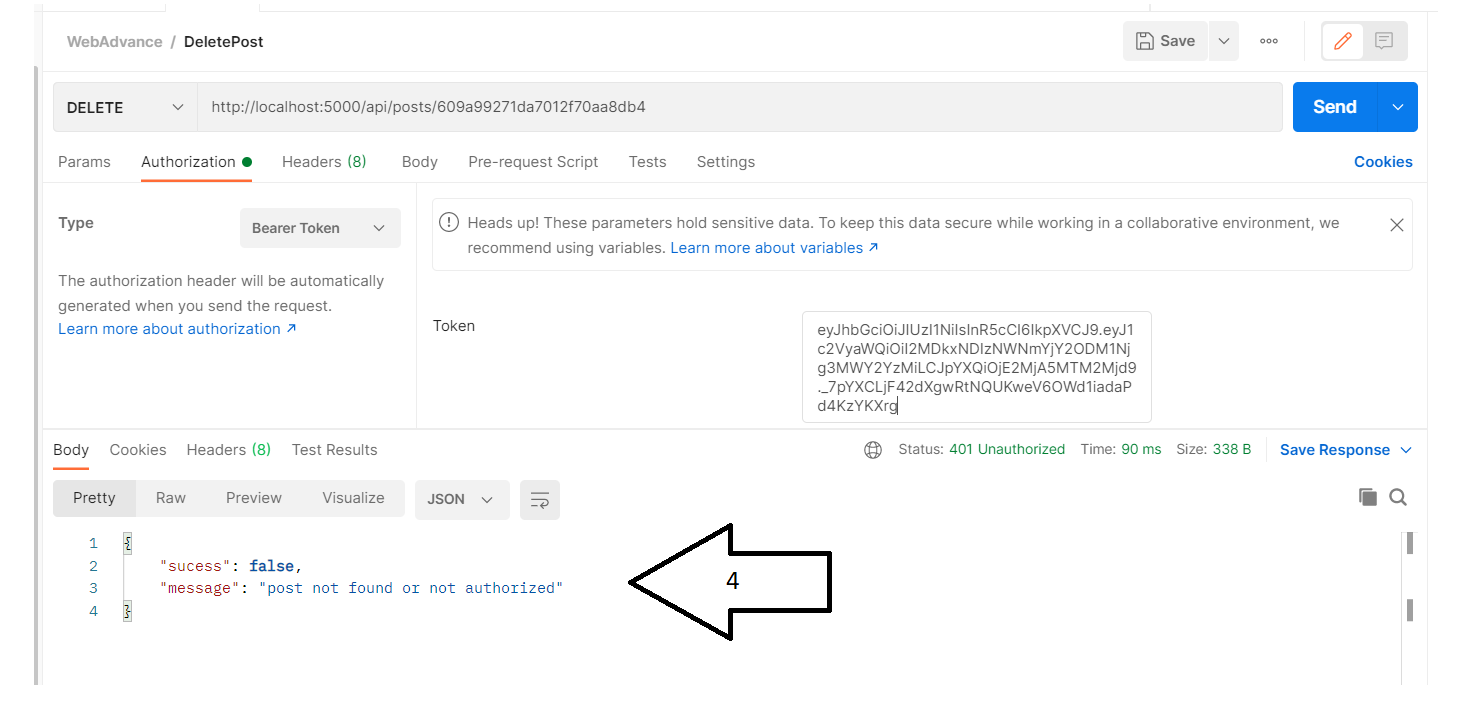
Hình 18 : Nếu id bài viết không có hoặc access token không đúng



Hình 19 : Kết quả trả về nếu đúng Id bài viết hoặc đúng Access Token

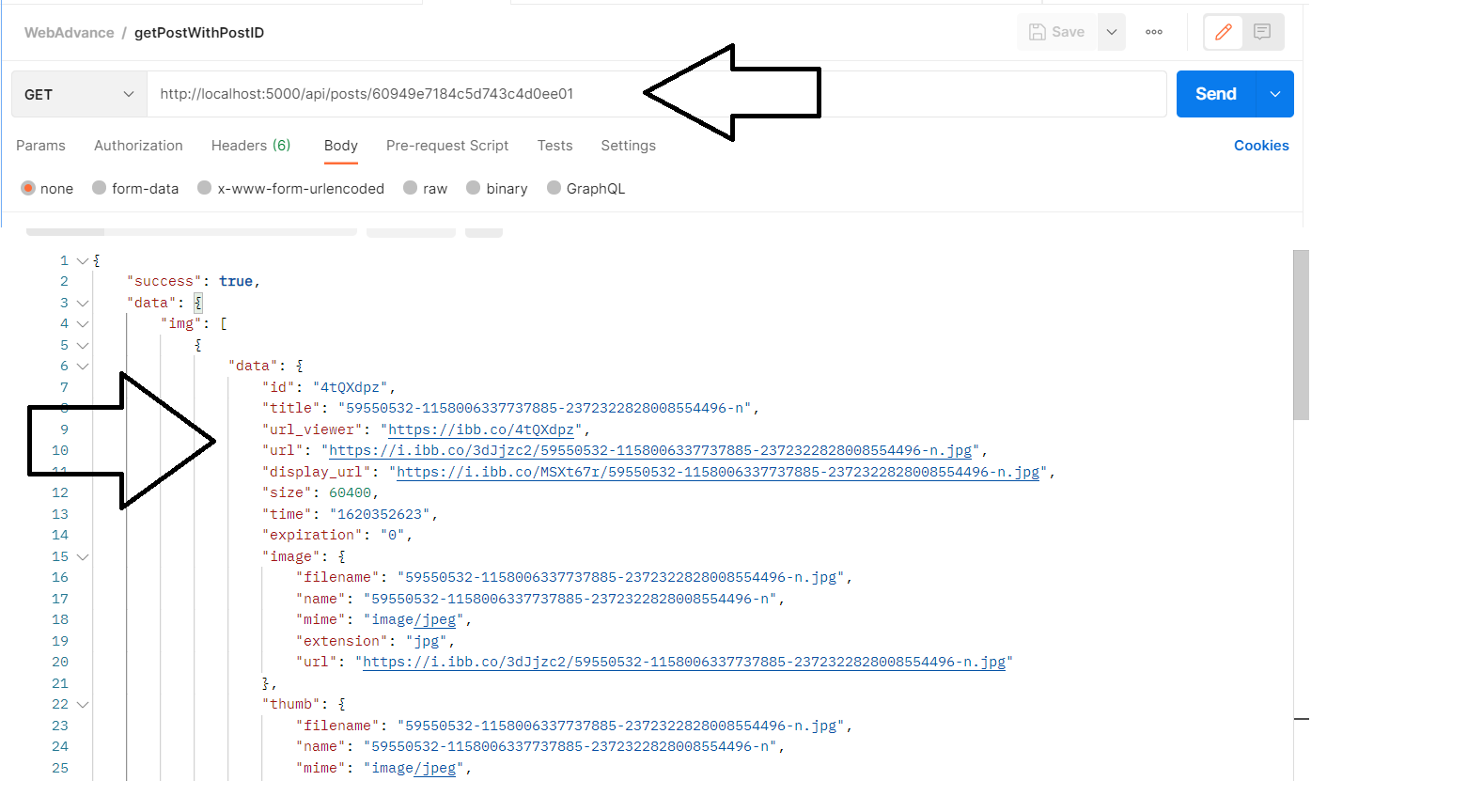


Hình 20 : API xóa bài viết



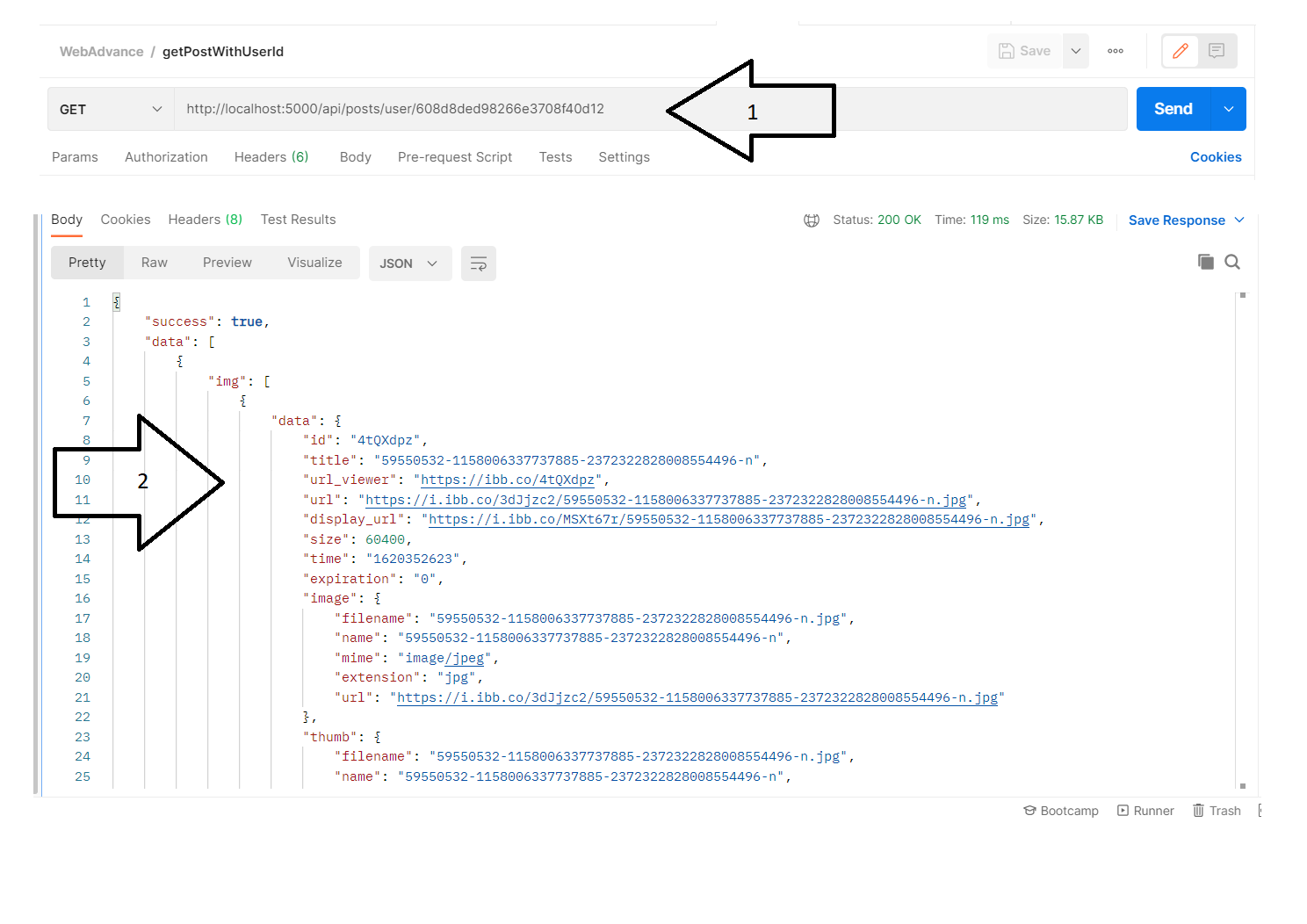
Hình 21 : API Xóa bài nếu không đúng người viết bài hoặc không tồn tại bài viết

* 1 : Là đường dẫn đến API : [http://localhost:5000/api/posts/idbaiviet](http://localhost:5000/api/posts/idbaiviet/)
* Phương thức để xóa bài viết bằng phương thức DELETE.
* 2 : Thêm Access Token cùng request để xem xét có đúng người post bài này không để có thể xóa được bài đăng này
* 3 : Kết quả trả về sau khi xóa thành công
* 4 : Kết quả kế sai Access Token và hoặc sai bài bài viết



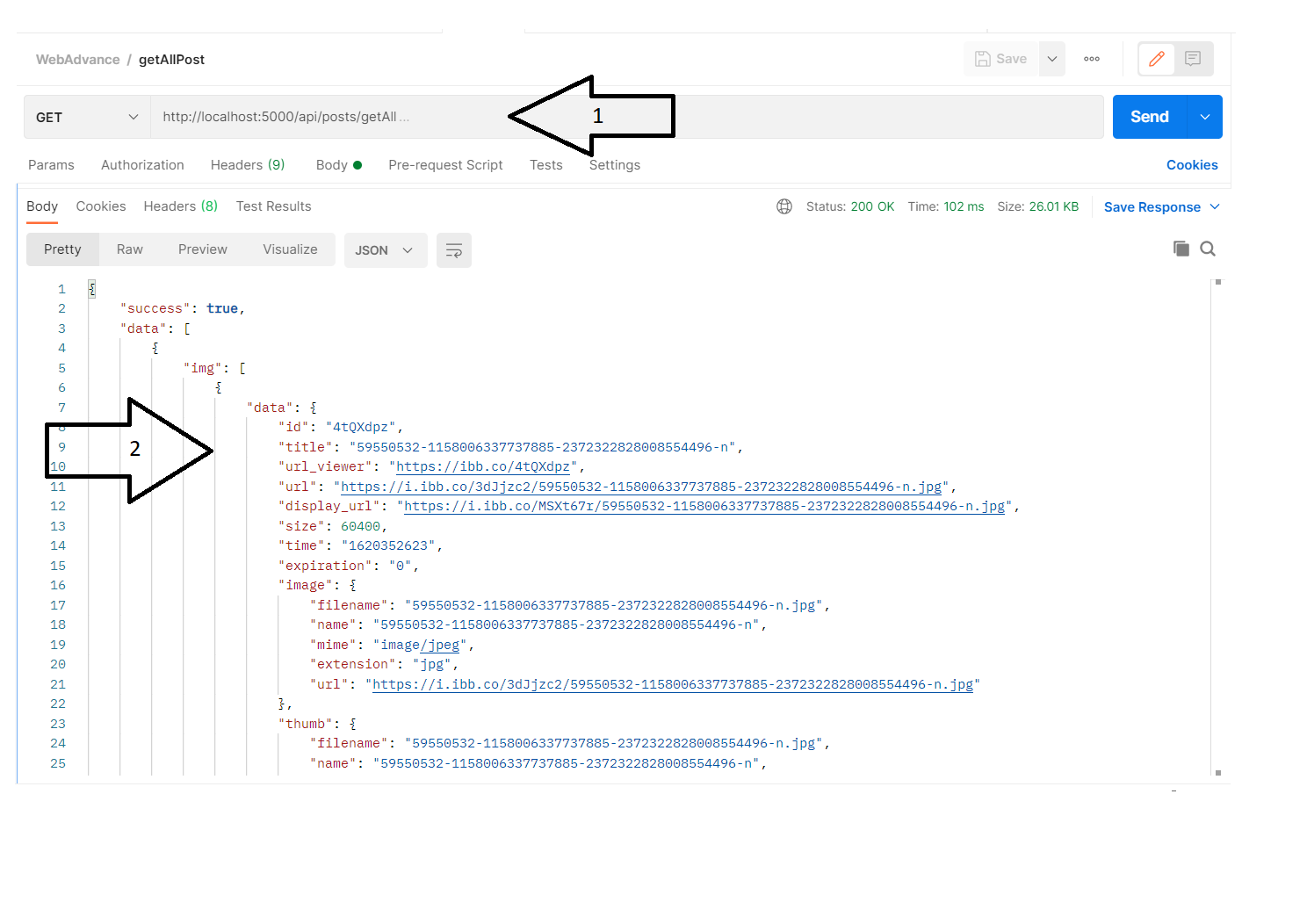
Hình 22 : API lấy thông tin bài viết với idpost

* 1 : Là đường dẫn đến API : [http://localhost:5000/api/posts/idbaiviet](http://localhost:5000/api/posts/idbaiviet/)
* Phương thức để lấy bài viết là GET



Hình 23 : API lấy bài post theo id của user

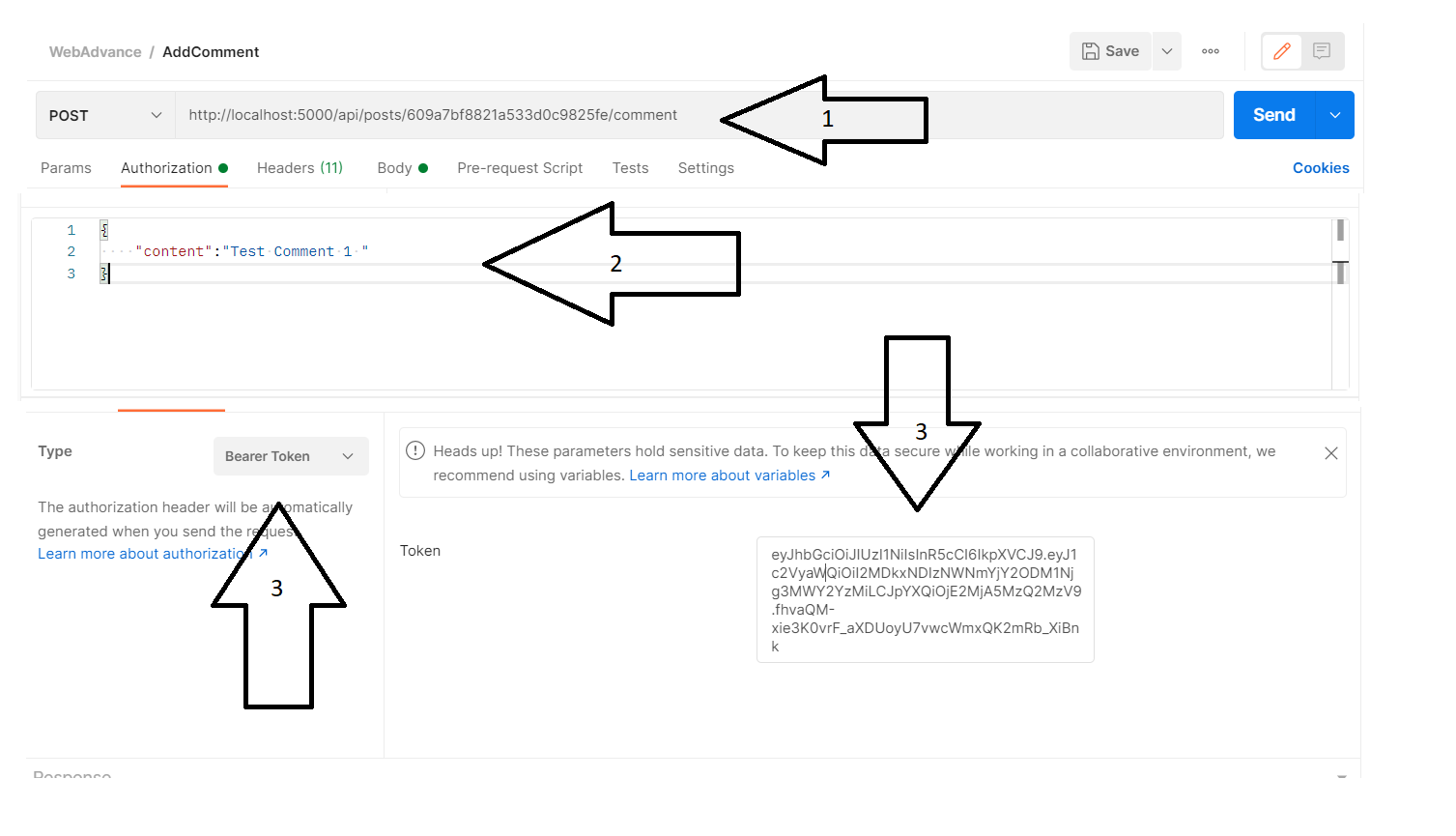
* 1 : Là đường dẫn đến API : [http://localhost:5000/api/posts/user/idUser](http://localhost:5000/api/posts/user/idUsert)
* Phương thức được sử dụng là GET
* 2 : Thông tin trả về tất cả các bài post mà user đó post lên



Hình 24 : API để lấy tất cả các bài viết trong hệ thống

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/posts/getAll>
* Phương thức được sử dụng là GET
* 2 : Thông tin tất cả các bài viết có trong hệ thống

1. Comment (Bình Luận )

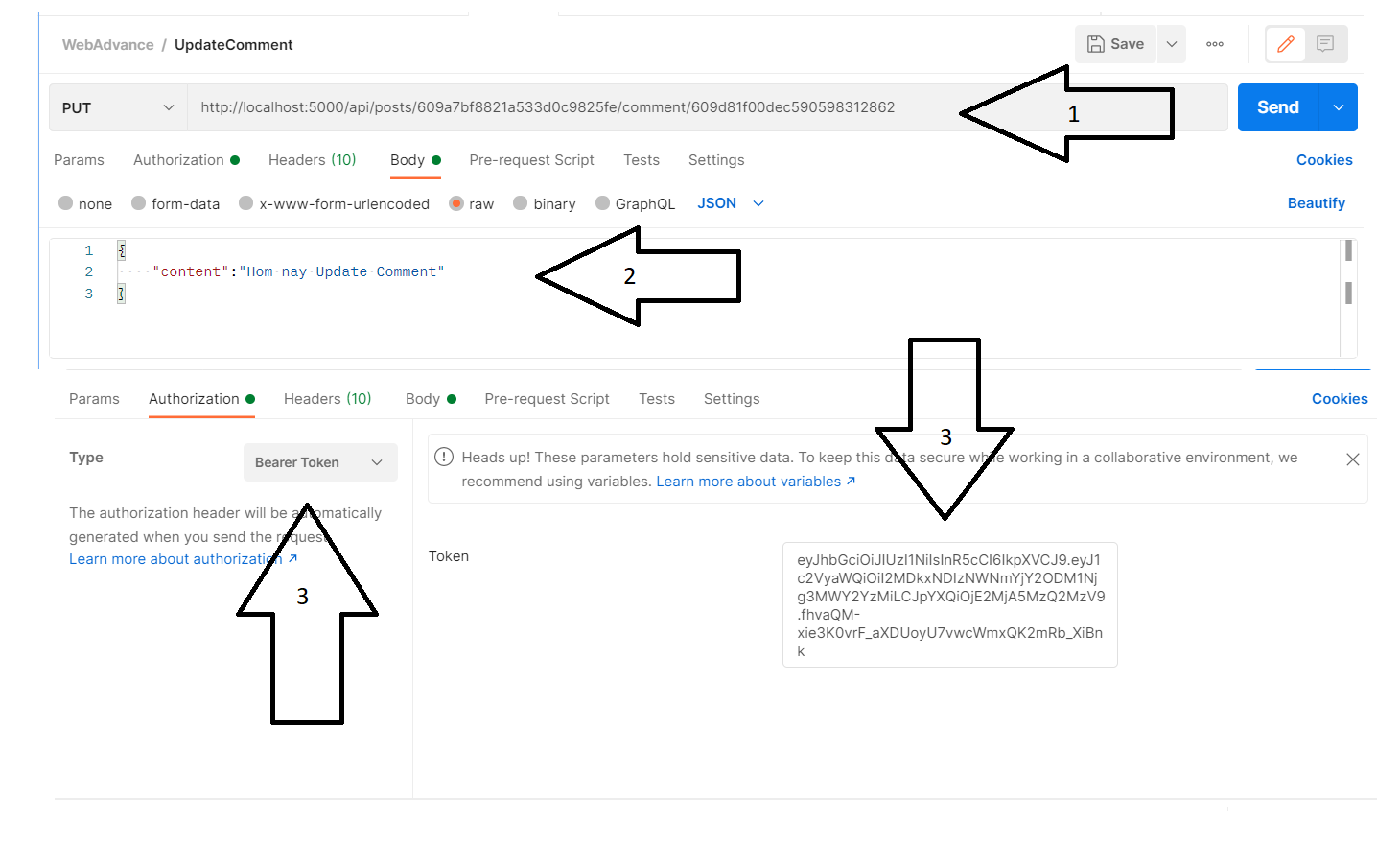


Hình 25 : API thêm comment vào bài viết

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/posts/idbaiviet/comment>
* Phương thức được sử dụng để gửi comment POST
* 2 : Thông tin thêm được kèm theo request gửi lên server
* 3 : Đính kèm Access token cùng với request để biết ai là người comment đó



Hình 26 : Kết quả thêm comment thành công

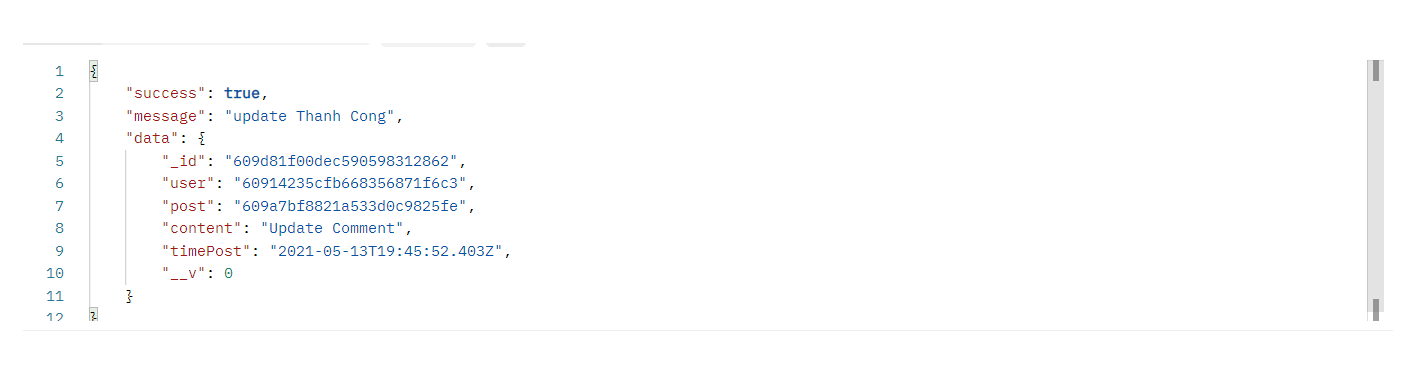


Hình 27 : API dùng để sửa comment của bài viết

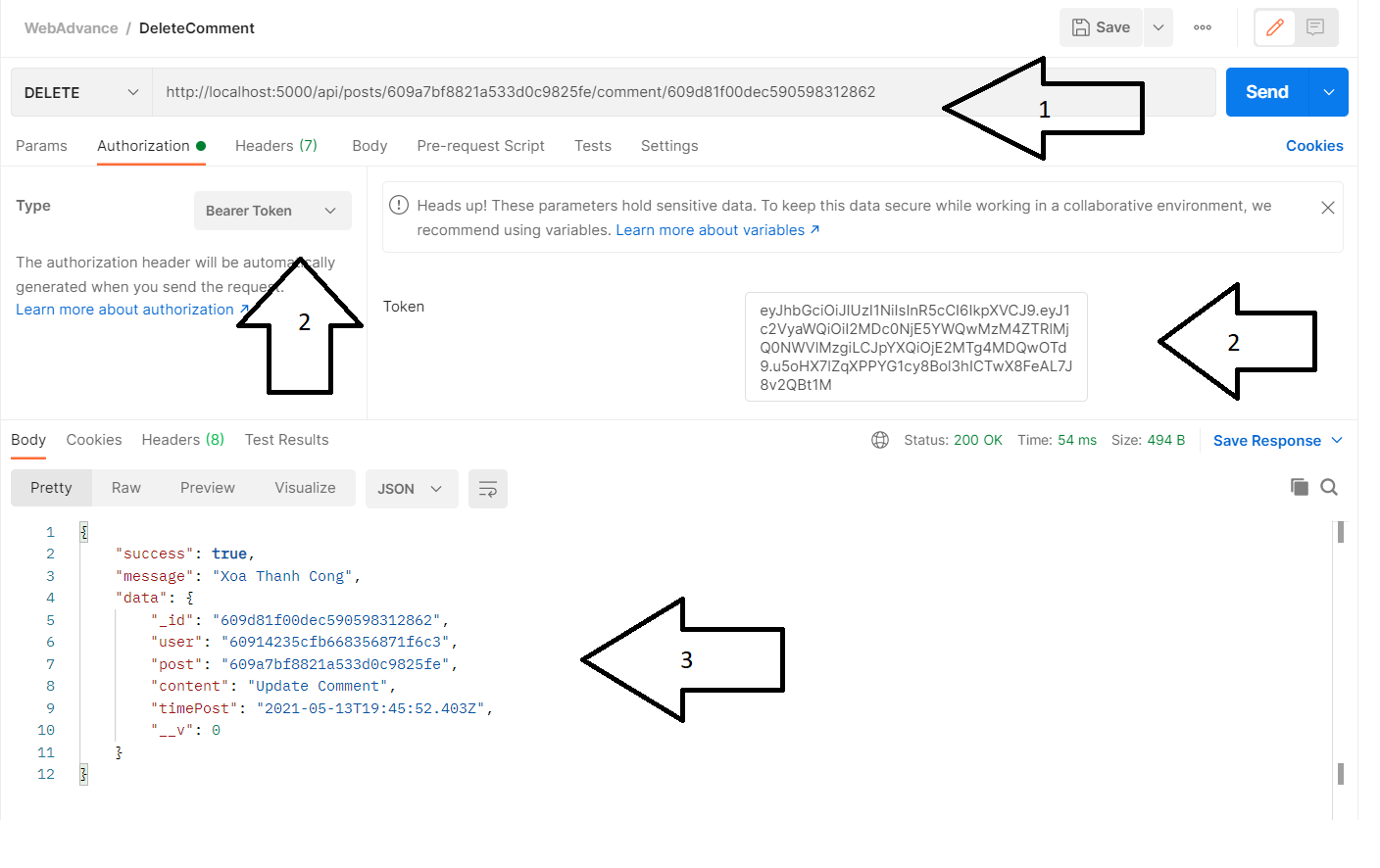
* 1 : Là đường dẫn đến API :

<http://localhost:5000/api/posts/idbaiviet/comment/idComment>

* Phương thức được thực hiện ở đây là PUT
* 2 : Thông tin thêm được kèm theo request gửi lên server là content là nội dung của comment
* 3 : Đính Access Token vào barer token của user để xem user có quyền thay đổi comment đó không



Hình 28 : Kết quả sau khi thực hiện phương thức



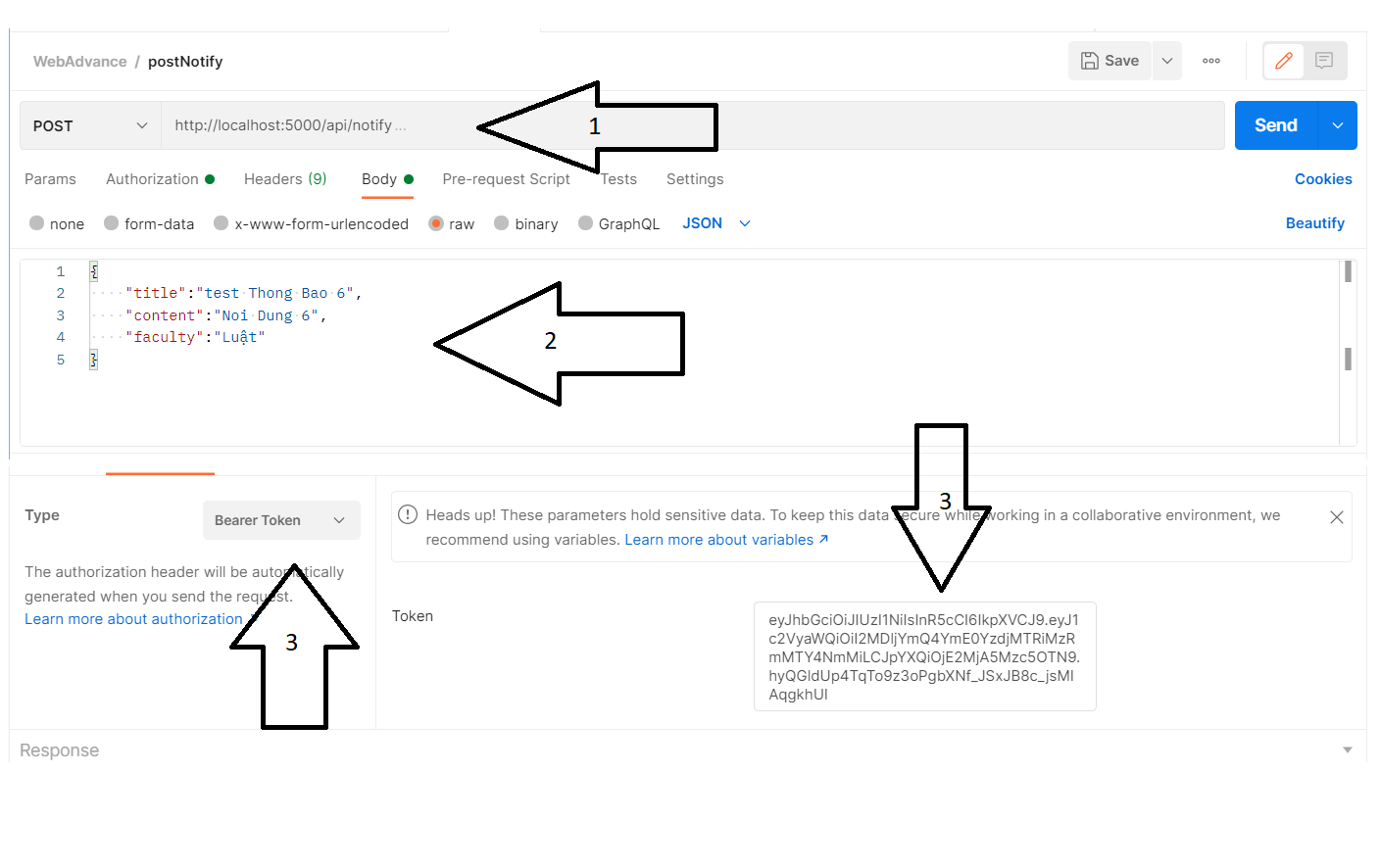
Hình 29 : API được sử dụng để xóa comment

* 1 : Là đường dẫn đến API :

<http://localhost:5000/api/posts/idbaiviet/comment/idComment>

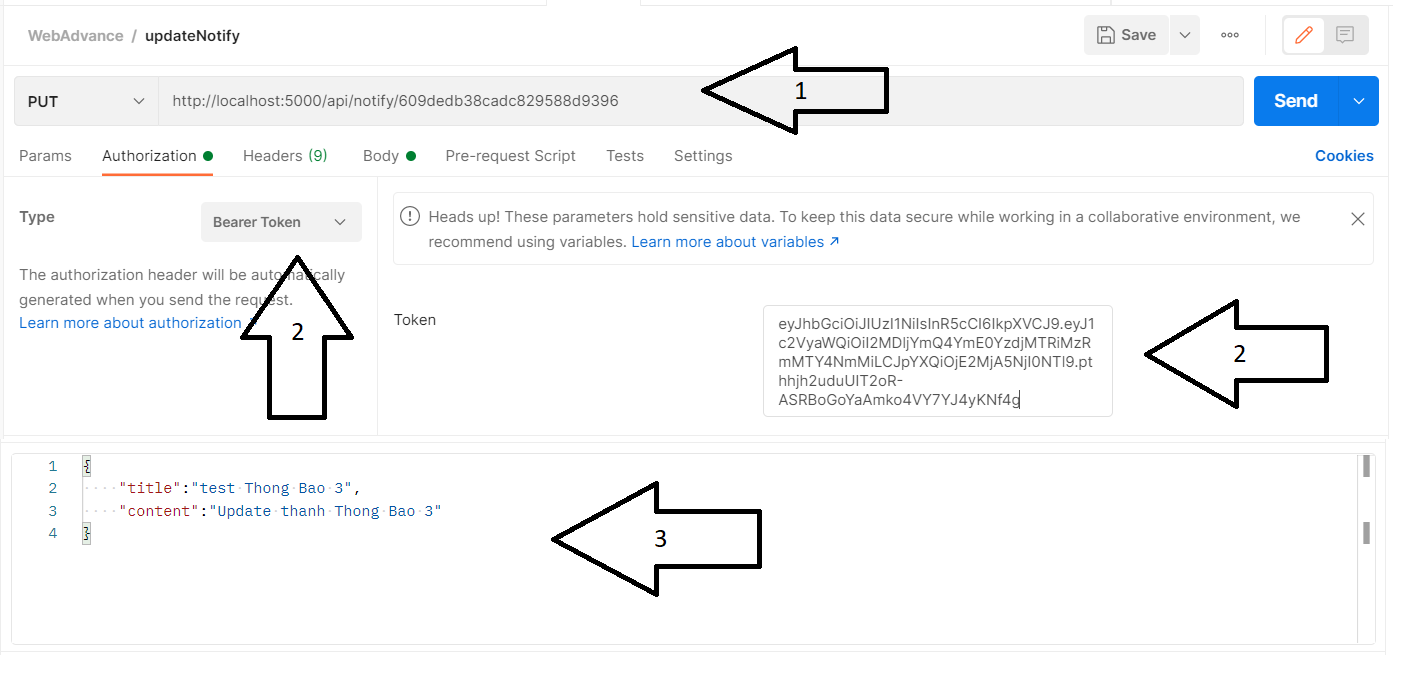
* Phương thức được thực hiện ở đây là PUT
* 2 : Đính Access Token vào barer token của user để xem user có quyền thay đổi comment đó không
* 3 : Thông tin trả về sau khi thực hiện việc xóa commnet

1. Notify (Thông Báo )



Hình 30 : API được sử dụng cho phòng/khoa đăng thông báo

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/notify>
* Phương thức được sử dụng để đăng thông báo là notify
* 2 : Thông tin được truyền theo request gửi lên API gồm title , content , faculty(Chọn khoa sẽ nhận được thông báo )
* 3 : đính kèm Access Token khi login tài khoản của phòng hoặc khoa

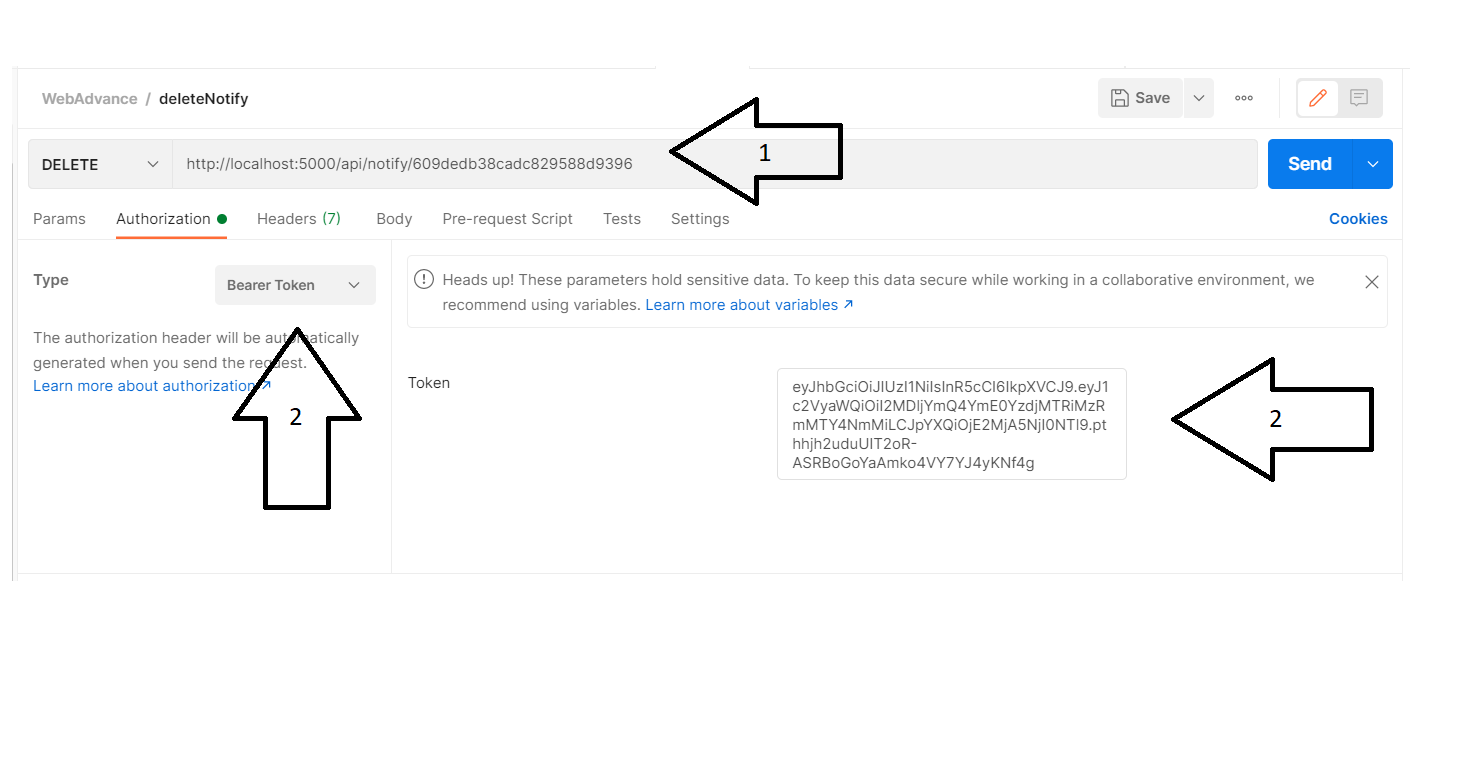


Hình 31 : API để cập nhật lại Thông báo

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/notify/idNotify>
* Phương Thức Được sử dụng để cập nhật là PUT
* 2 : Kèm theo Access Token của khoa sau khi đăng nhập vào barer token để xem có đúng khoa đã đăng thông báo đó không
* 3 : Thông tin chỉnh sửa có thể là title , content

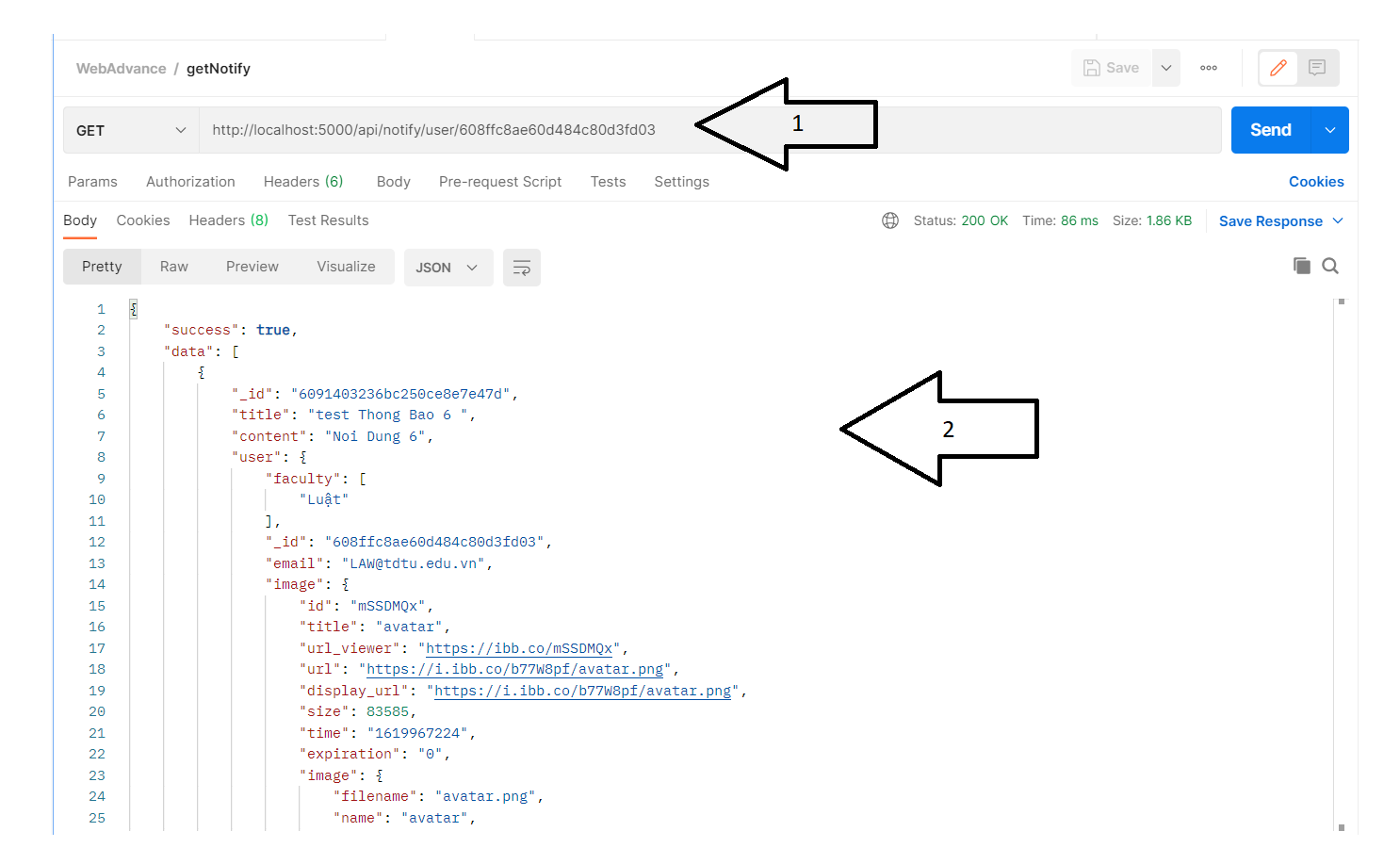


Hình 32 : Kết quả trả về sau khi cập nhật



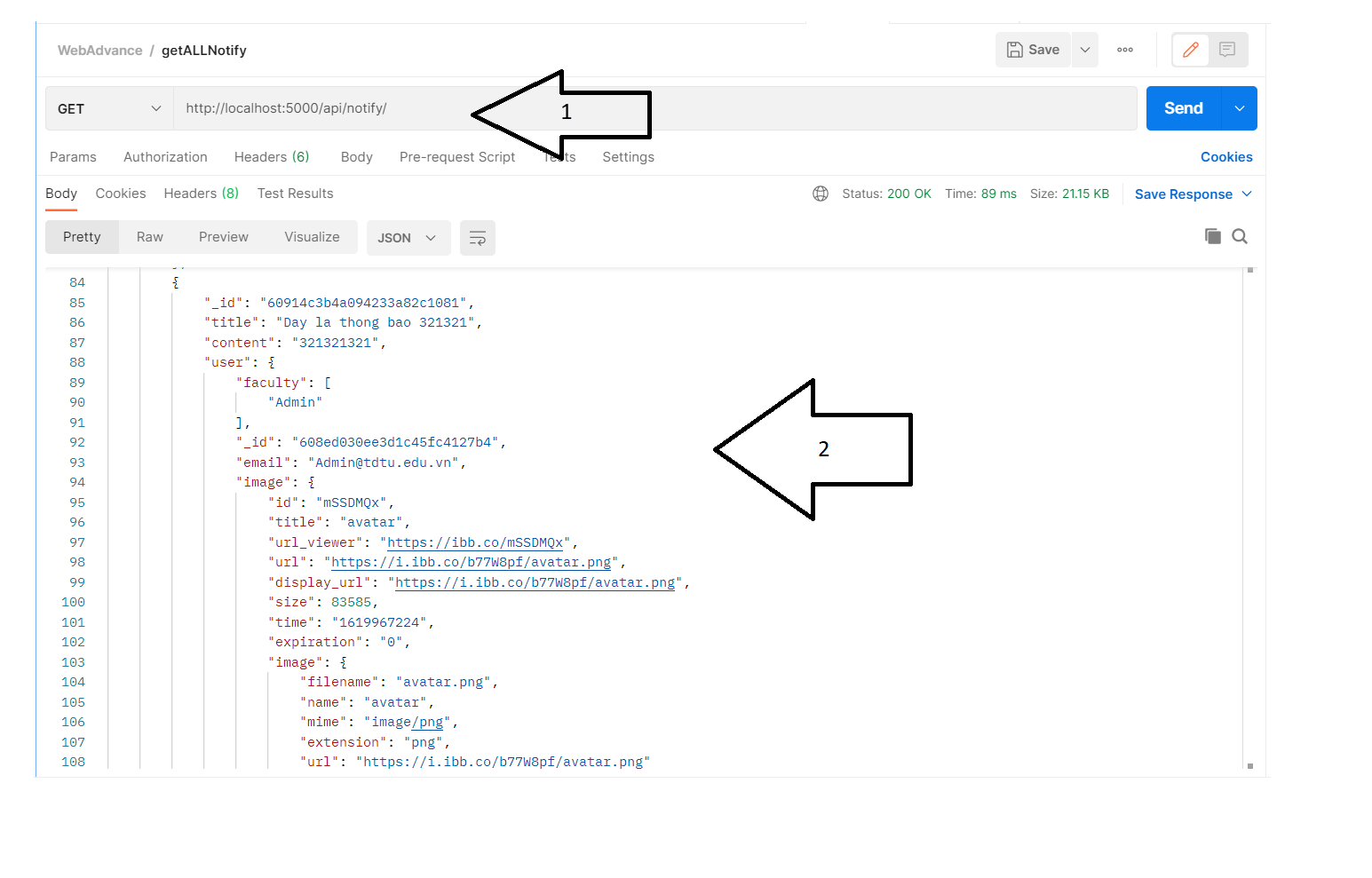
Hình 33 : API xóa thông báo

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/notify>/idNotify
* Phương thức sử dụng để xóa thông báo là DELETE
* 2 : Kèm theo Access Token của khoa sau khi đăng nhập vào barer token để xem có đúng khoa đã đăng thông báo đó không mới có thể xóa được



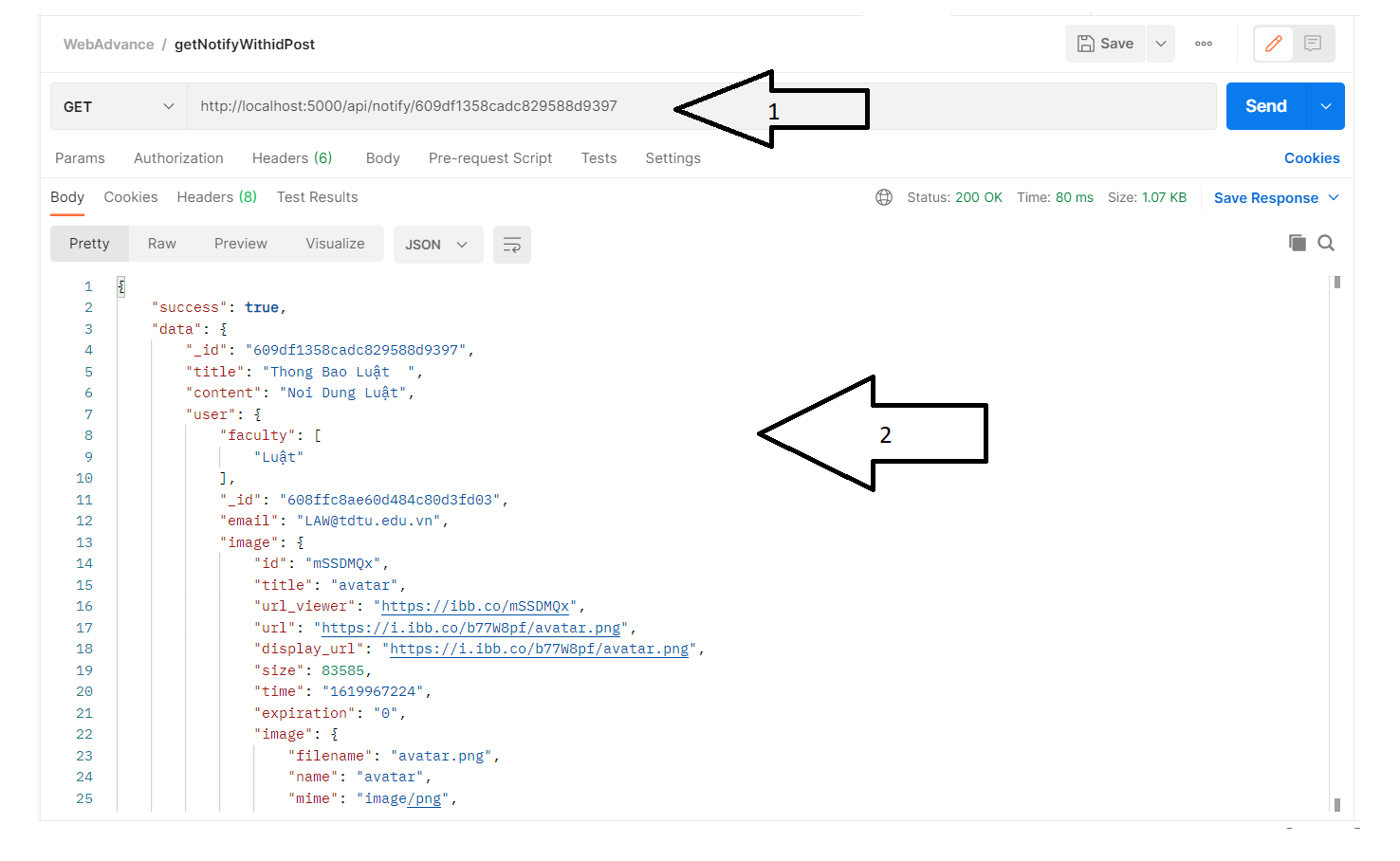
Hình 34 : API để lấy tất cả thông báo của khoa theo id khoa

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/notify>/user/idfaculty
* Phương thức để lấy tất cả thông báo theo khoa là GET
* 2 : Trả về tất cả thông báo mà khoa đó đã đăng lên



Hình 35 : API lấy tất cả thông báo

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/notify>
* Phương thức để lấy tất cả thông báo là GET
* 2 : Trả về tất cả thông báo có trong hệ thống



Hình 36 : API để lấy thông tin của 1 thông báo theo idNotify

* 1 : Là đường dẫn đến API : <http://localhost:5000/api/notify>/idNotify
* 2 : Trả về thông tin chi tiết của thông báo

1. XÂY DỰNG FRONT-END

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

1. [TÌM HIỂU VỀ MONGODB (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGlY3)
2. [Xây dựng stack M.E.R.N với app NodeJS - phần 1 - Hocweb.vn](https://hocweb.vn/xay-dung-mo-hinh-mern-voi-app-nodejs-phan-1/)

**Tiếng Anh**

1. [Node.js Tutorial (w3schools.com)](https://www.w3schools.com/nodejs/)
2. [GitHub - socketio/socket.io: Realtime application framework (Node.JS server)](https://github.com/socketio/socket.io)