

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



**BÀI TẬP LỚN GIỮA KỲ MÔN CHUYÊN ĐỀ CÔNG  
NGHỆ PHẦN MỀM**

**Tìm hiểu về Web API và Demo**

*Người hướng dẫn:* **Thầy Lục Minh Tuấn**

*Người thực hiện:* **Nguyễn Văn Huy– 51800783**

**Khoá : 22**

**Lớp : 18050203**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

## **LỜI CẢM ƠN**

Em xin chân thành cảm ơn thầy đã hướng dẫn em trong quá trình học thực hành và dòi hạn nộp bài.

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

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Thầy Trần Thanh Phước, Thầy 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 16 tháng 12 năm 2021*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Văn Huy*

## **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)

# MỤC LỤC

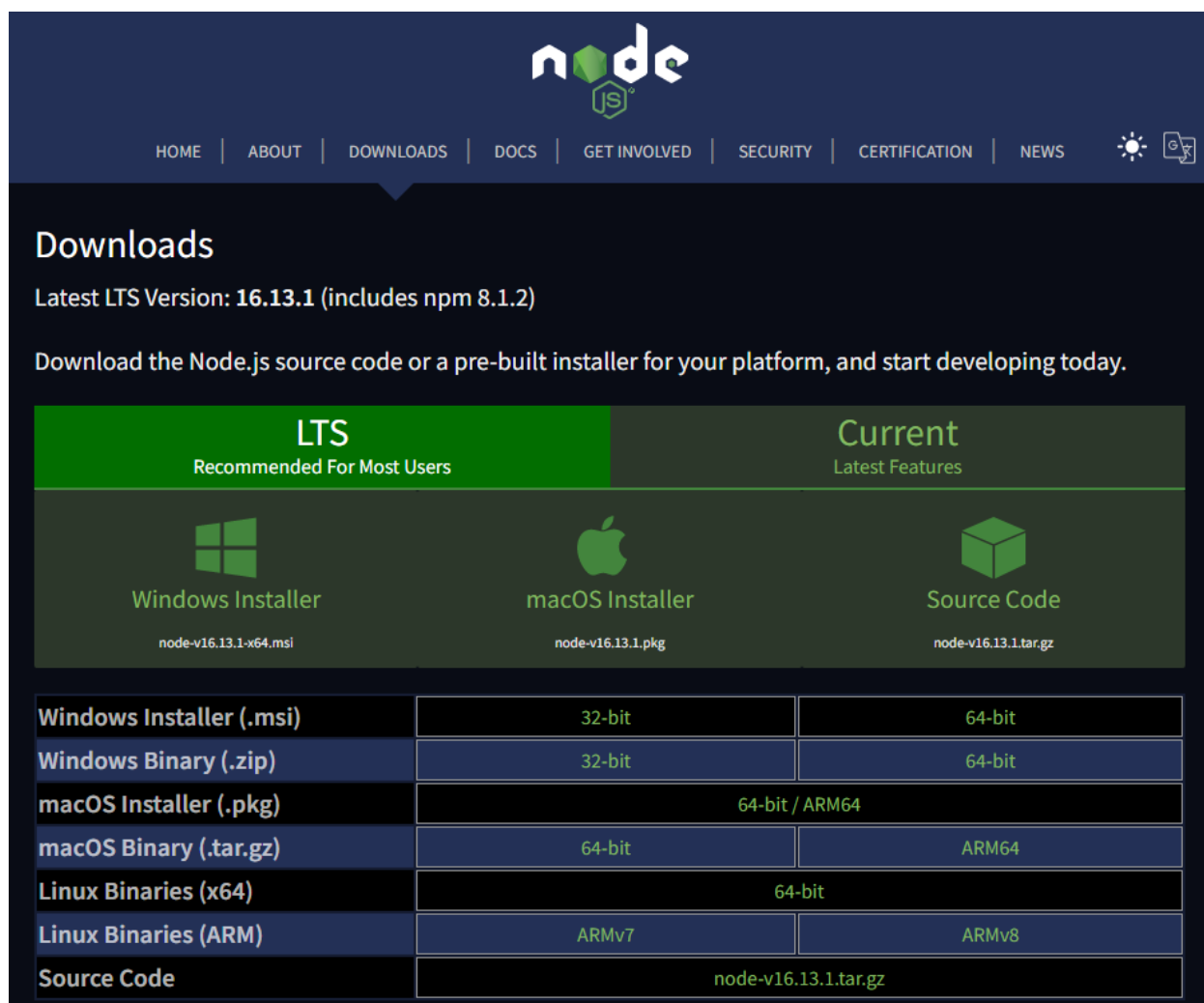
CHƯƠNG 1 : HƯỚNG DẪN CÀI ĐẶT VÀ THIẾT LẬP MÔI TRƯỜNG NODE JS, NPM VÀ MONGO DATABASE .....	7
1. Cài đặt Nodejs .....	7
2. Cài đặt MongoDB .....	9
3. Cài đặt và khởi chạy project API .....	12
CHƯƠNG 2: WEB API VÀ DEMO:API CHO ỨNG DỤNG SHOPPING CART(BÁN QUẦN ÁO).....	13
1. Giới thiệu .....	13
2. Authentication .....	13
2.1 Register .....	13
2.2 Login.....	14
2.3 Refresh Token.....	16
2.4 Log out.....	17
3 CRUD Tag.....	18
3.1 Tag list .....	18
3.2 Add tag.....	18
3.3 Delete tag .....	19
4 CRUD Category .....	20
4.1 Category list.....	20
4.2 Add Category .....	21

4.3	Delete Category .....	21
4.4	Update Category .....	22
5	CRUD Product .....	23
5.1	Upload product picture .....	23
5.2	Add product .....	24
5.3	Get Product with ID.....	25
5.4	Update Product .....	26
5.5	Update Product .....	27
5.6	Phân trang Product.....	28
6	Cart API.....	29
6.1	Get Cart Infor.....	29
6.2	Add product to cart .....	30
6.3	Subtract product from cart.....	30
6.4	Remove product from cart.....	32
7	Order API .....	32
7.1	Create Order.....	32
7.2	Get Order list .....	33
7.3	Get Order with id .....	34
TÀI LIỆU THAM KHẢO .....		35

# CHƯƠNG 1 : HƯỚNG DẪN CÀI ĐẶT VÀ THIẾT LẬP MÔI TRƯỜNG NODE JS, NPM VÀ MONGO DATABASE

## 1. Cài đặt Nodejs

Để cài đặt môi trường NodeJS trước hết cần vào trang <https://nodejs.org/en/download/> để cài đặt phiên bản thích hợp:

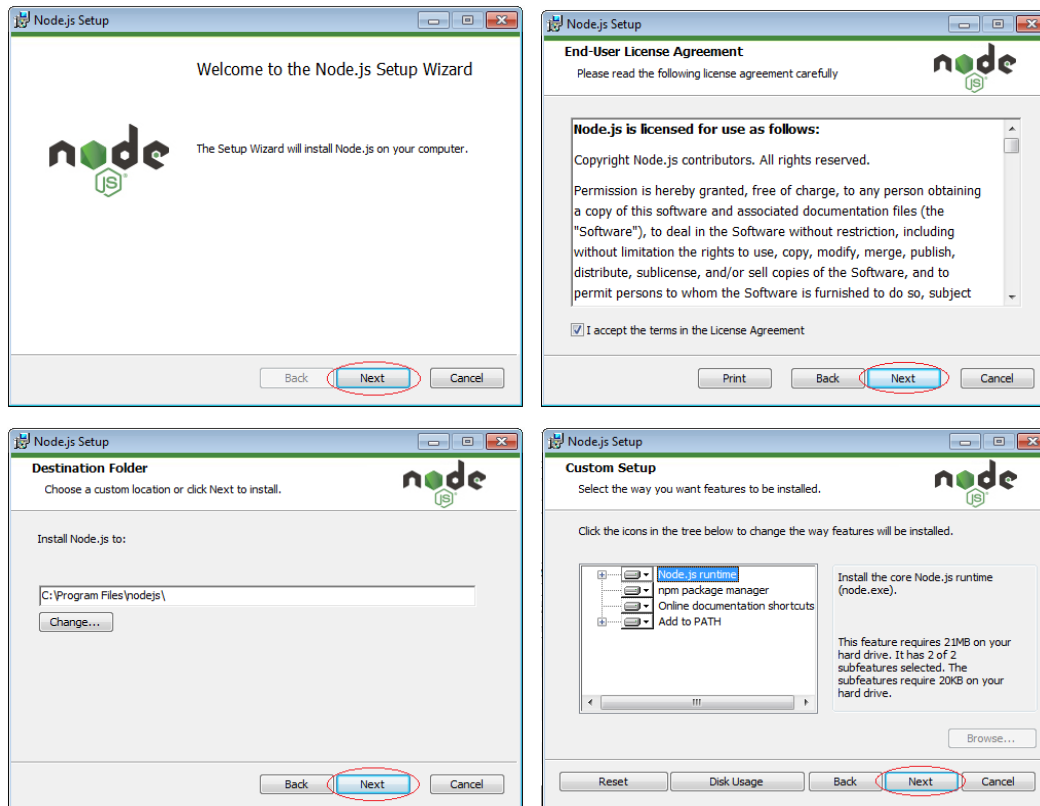


	LTS Recommended For Most Users	Current Latest Features
Windows Installer	node-v16.13.1-x64.msi	
macOS Installer	node-v16.13.1.pkg	
Source Code	node-v16.13.1.tar.gz	

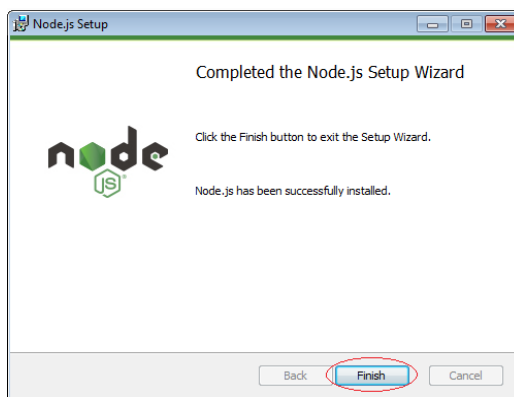
	32-bit	64-bit
Windows Installer (.msi)		
Windows Binary (.zip)		
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v16.13.1.tar.gz	

Ở đây do máy tính của em xài hệ điều hành window 64bit nên em sẽ chọn phiên bản node-v16.13.1-x64.msi.

Sau khi tải về, double click vào file để tiến hành cài đặt.



Theo mặc định, phần mềm NPM cũng được cài đặt vào hệ thống. Đây là một phần mềm quản lý các thư viện Javascript.

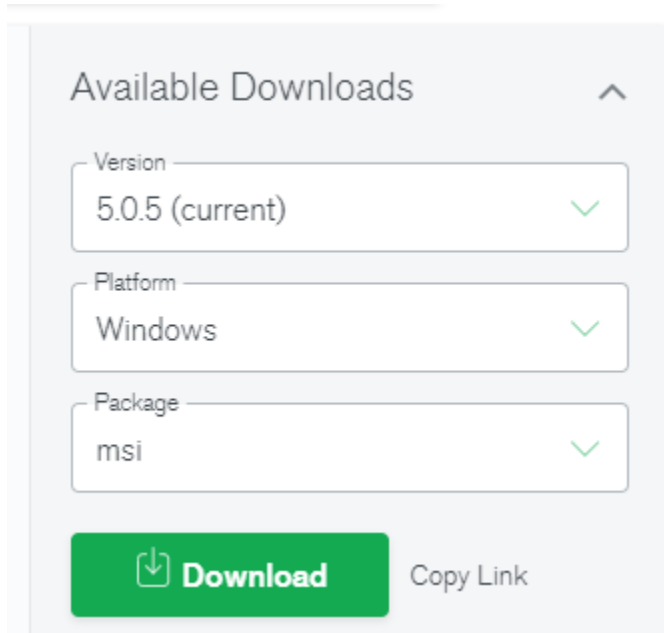


Sau khi cài đặt thành công, thầy cần mở cửa sổ cmd lên để kiểm tra phiên bản nodejs và npm bằng cách gõ các dòng lệnh `node -v` và `npm -v`.

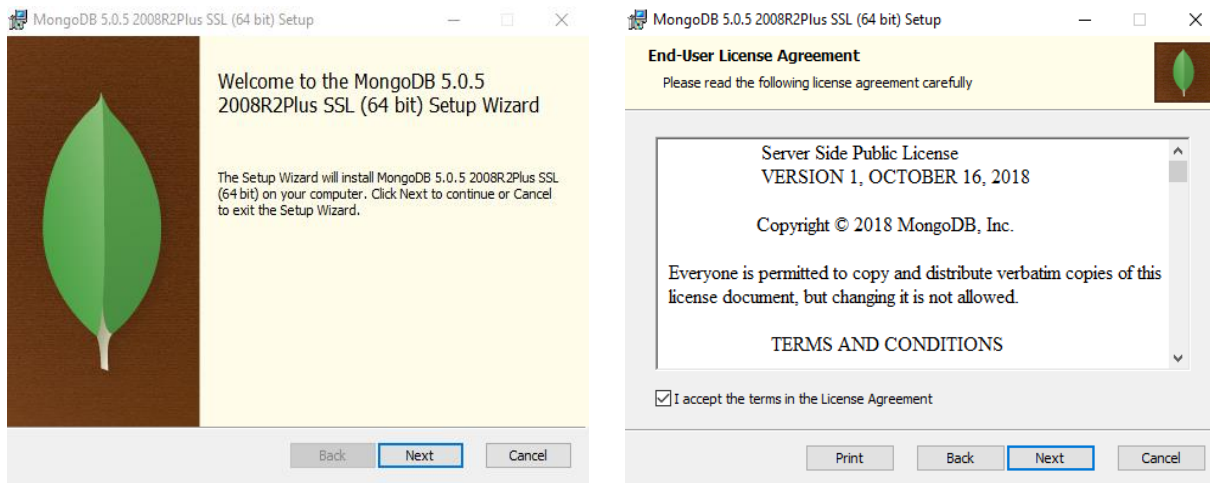


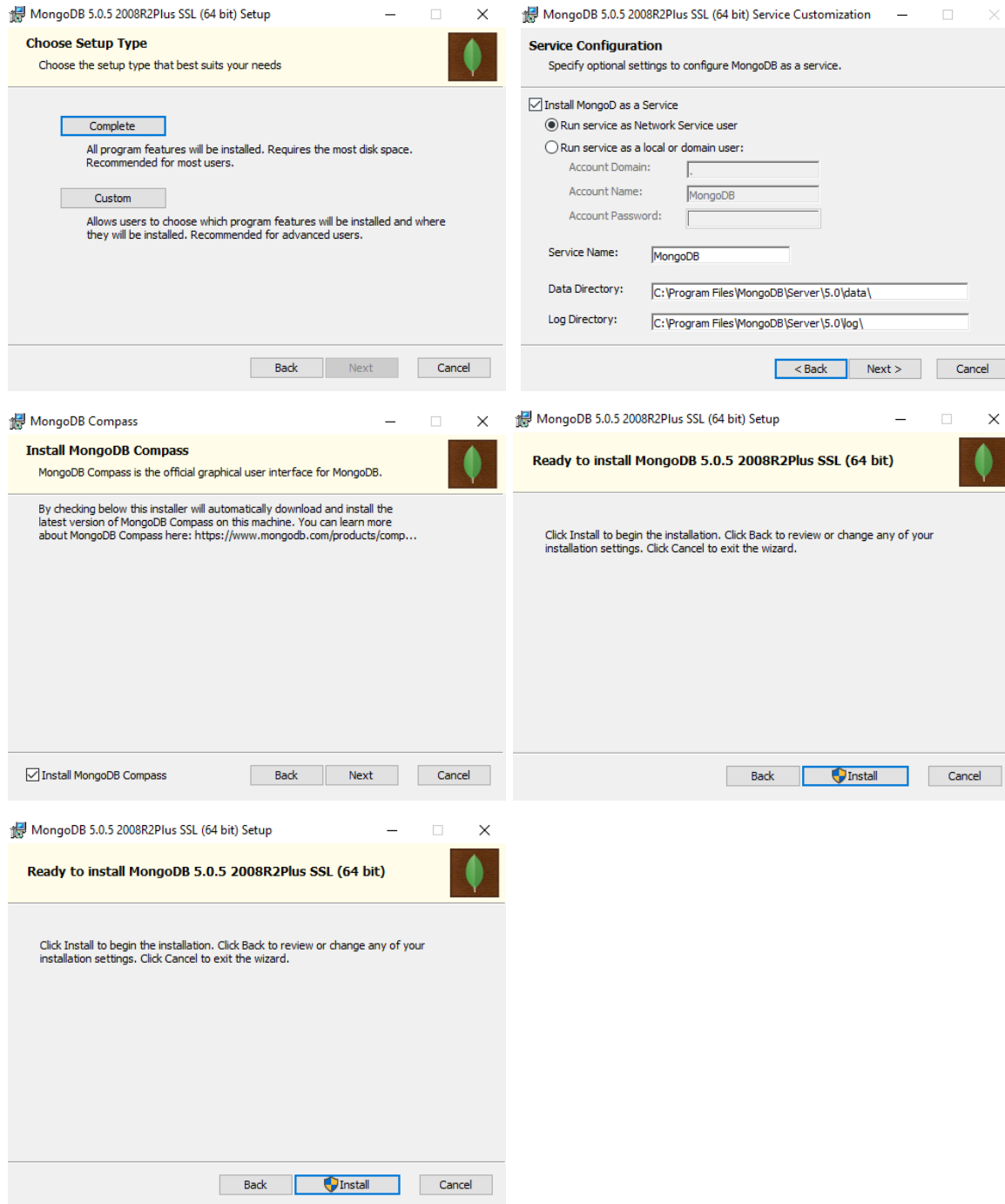
## 2. Cài đặt MongoDB

Truy cập trang <https://www.mongodb.com/try/download/community> để tải file cài đặt MongoDB. Chọn Platform và version thích hợp để cài đặt. Hình dưới là version mà em sử dụng.

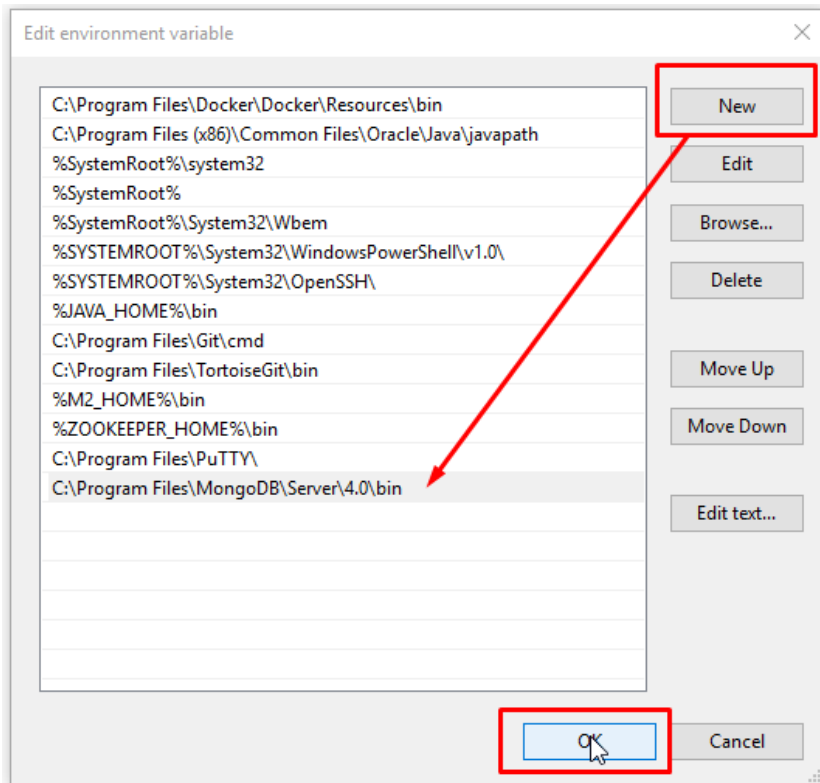
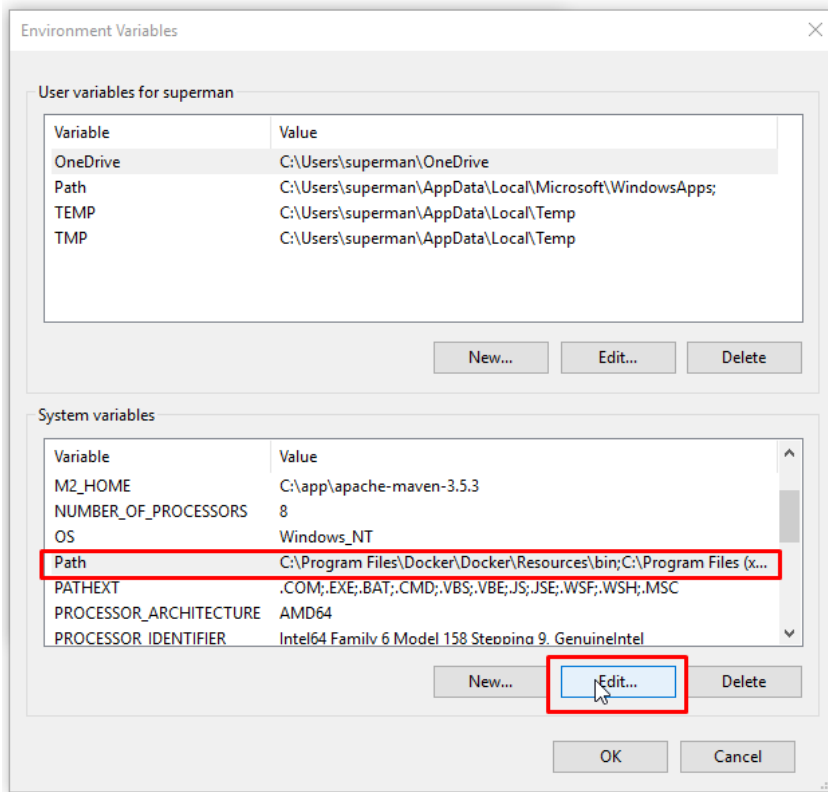


Các bước tiếp theo:





Tiếp theo là thêm biến môi trường cho MogoDB.



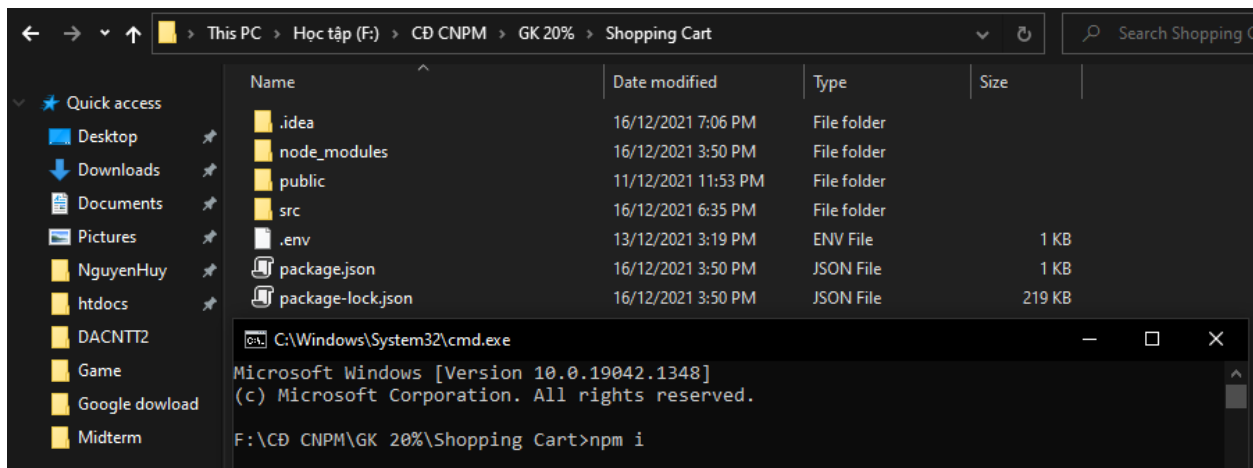
Chạy thử MongoDB bằng cách mở màn hình cmd hoặc powerShell và chạy lệnh mongo hoặc mở ứng dụng MongoDB Compass.

Ngoài ra thầy có thể xem hướng dẫn cài đặt từ:

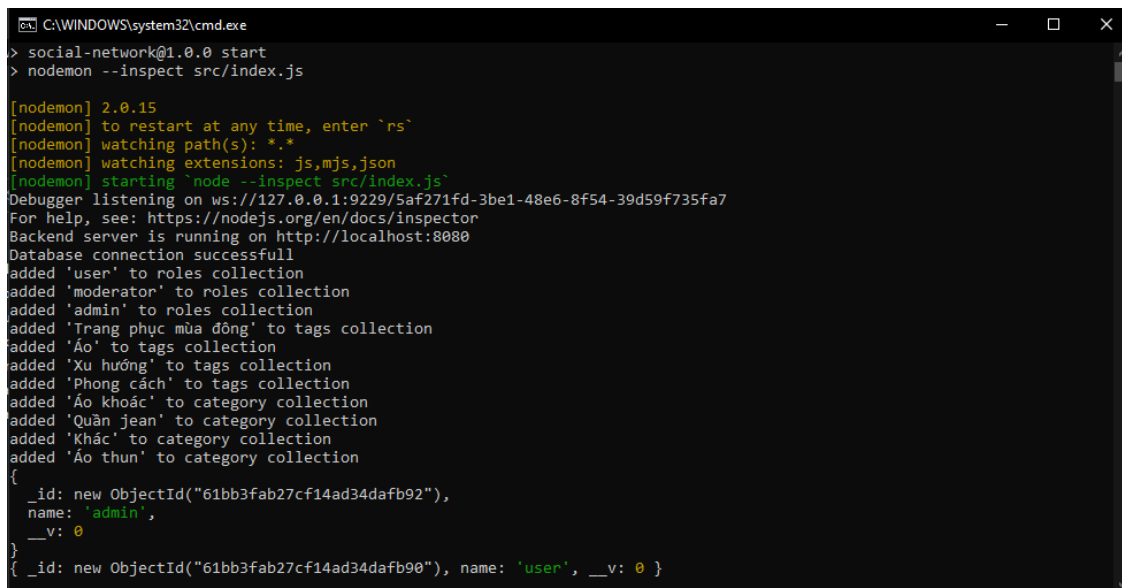
<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

### 3. Cài đặt và khởi chạy project API

Tại thư mục chứa source code của project, mở cmd và chạy lệnh npm install để tải về node modules thư viện cho project.



Kế đó gõ lệnh npm start để chạy server cho api.



Trong trường hợp thầy muốn cài đặt lại các Port hoặc thay đổi link Database, thầy có thể chỉnh sửa từ file .env

## CHƯƠNG 2: WEB API VÀ DEMO:API CHO ỨNG DỤNG SHOPPING CART(BÁN QUẦN ÁO)

### 1. Giới thiệu

Ứng dụng web API shopping cart được viết bằng ngôn ngữ js dựa trên môi trường nodejs với đường dẫn mặc định là <http://localhost:8080/api>

Web API được viết với các chức năng như:

- Authentication
- CRUD sản phẩm
- Tương tác với giỏ hàng
- Đặt hàng
- Một số chức năng khác

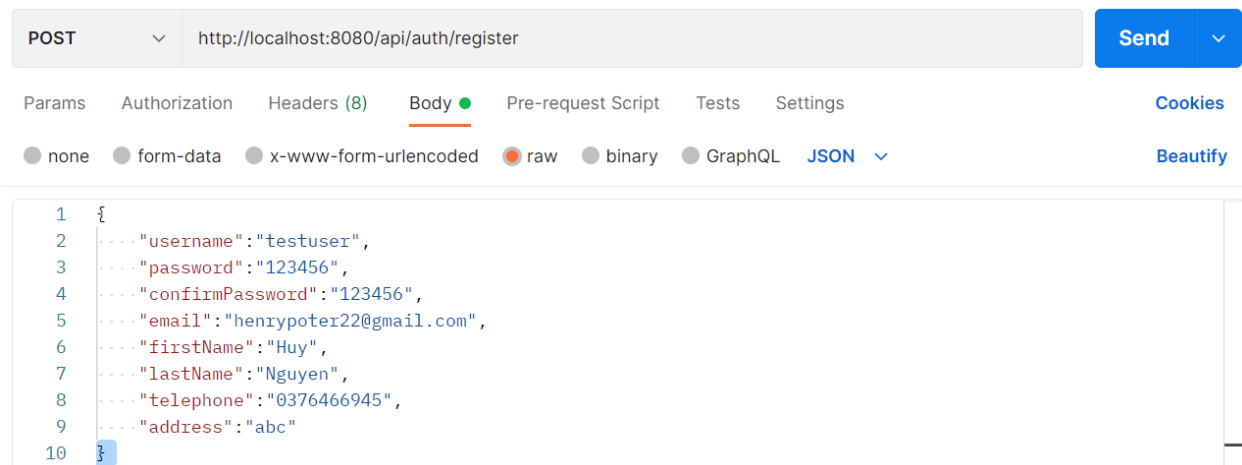
### 2. Authentication

#### 2.1 *Register*

Url: <http://localhost:8080/api/auth/register>

Phương thức: Post

Dữ liệu truyền vào:



Kết quả:



Bên cạnh việc truyền dữ liệu qua Json thì sử dụng form-data cũng cho kết quả tương tự. Ngoài ra api đăng ký cũng sử dụng validate cho các trường hợp không hợp lệ như mật khẩu dưới 6 ký tự, số điện thoại không hợp lệ, sai email, mật khẩu không khớp v.v..

## 2.2 Login

Url: <http://localhost:8080/api/auth/login>

Phương thức: Post

Dữ liệu truyền vào:

### Kết quả:

Dữ liệu được trả về bao gồm thông tin người dùng, access token dùng để xác thực khi sử dụng các chức năng khác, refresh token để xin cấp lại accesstoken trong trường hợp accesstoken hết hạn mà không cần phải đăng nhập lại. Accesstoken có tuổi thọ là 15 phút trong khi refreshtoken là 30 ngày. Do đó bảo mật refresh token là rất quan trọng vì khi kẻ tấn công có được refresh token và accesstoken thì có thể sử dụng được các chức năng của tài khoản trong vòng 30 ngày.

Bên cạnh việc truyền dữ liệu qua Json thì sử dụng form-data cũng cho kết quả tương tự. Ngoài ra api đăng ký cũng sử dụng validate cho các trường hợp không hợp lệ như mật khẩu dưới 6 ký tự v.v..

### 2.3 Refresh Token

Url: <http://localhost:8080/api/auth/refresh>

Phương thức: Post

Do lúc này người dùng sử dụng các chức năng cần authorized do đó phải truyền accesstoken lấy ra từ login thành công thì mới có thể sử dụng được. Theo như kết quả trả về từ mục 2.2 thì ta có accesstoken là:

`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwZXI6bnVzZXJuYW11IjoiSHV5ZGVwemFpIn0sImhhdCI6MTYzOTU5MTQyMCwiZXhwIjoxNjM5Njc3ODIwfQ.T-V9ZNnRXu_1Wan-syaCYIAxSG5ii2QyxliUmREXCSA`

Để sử dụng được API này, vào tab Authorization trong phần sử dụng API, chọn Type là API Key. Kế đó, nhập Key là x\_authorization với value chính là accesstoken đã lấy ra được từ phần login. Các API cần authorized khác cách làm cũng tương tự.

POST <http://localhost:8080/api/auth/refresh> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Type API Key

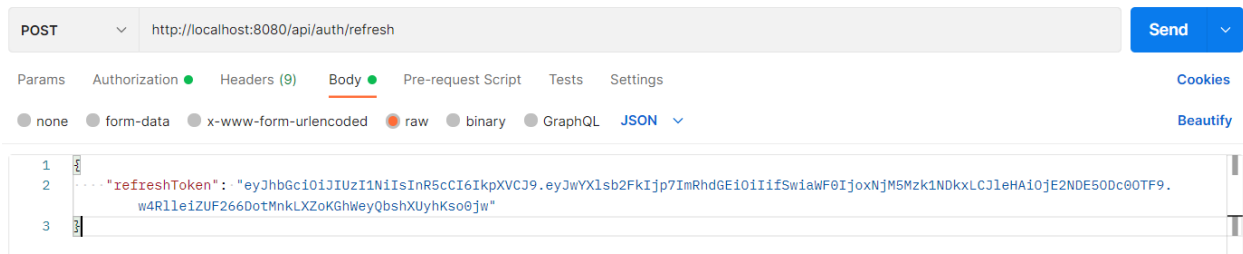
The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

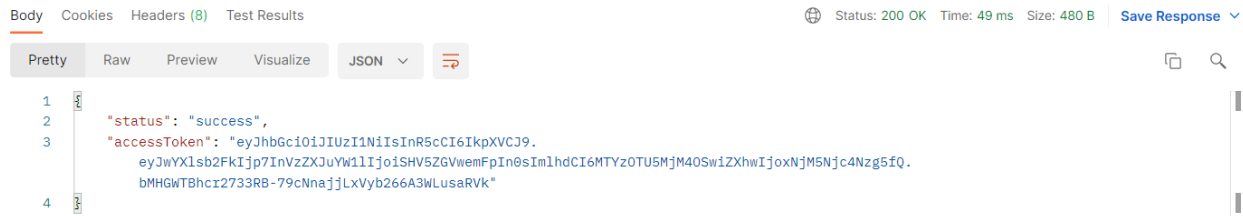
Key	x_authorization
Value	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwZXI6bnVzZXJuYW11IjoiSHV5ZGVwemFpIn0sImhhdCI6MTYzOTU5MTQyMCwiZXhwIjoxNjM5Njc3ODIwfQ.T-V9ZNnRXu_1Wan-syaCYIAxSG5ii2QyxliUmREXCSA
Add to	Header

Dữ liệu truyền vào chính là refreshtoken đã nhận được ở API login.





Kết quả:



Dữ liệu được trả về chính là access token dùng để xác thực khi sử dụng các chức năng khác mà không bị invalid. Bên cạnh việc truyền dữ liệu qua Json thì sử dụng form-data cũng cho kết quả tương tự.

## 2.4 Log out

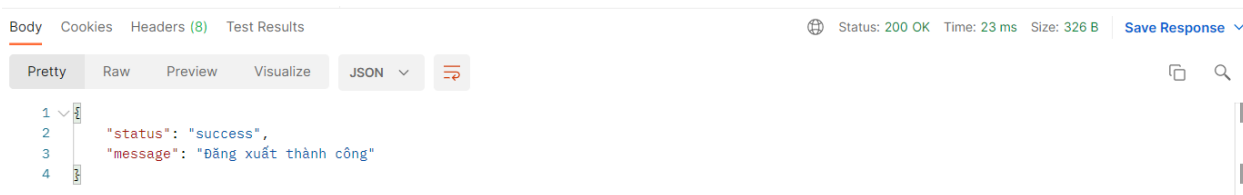
Url: <http://localhost:8080/api/auth/logout>

Phương thức: Post

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào không cần do các thông tin cần thiết cho việc authorized đều nằm ở header

Kết quả:



Dữ liệu được trả về bao gồm thông báo thành công để client biết mà xóa hết các thông tin xác thực trong bộ nhớ cũng như đăng xuất ra khỏi ứng dụng.

### 3 CRUD Tag

Đối với quần áo, bên cạnh category dùng để phân loại thì cũng cần tag để giúp người dùng dễ tìm kiếm hơn. Thay vì mỗi loại quần áo chỉ được có 1 category thì nó có thể sở hữu nhiều tag, ví dụ như vào mùa đông ta có thể thêm các tag vào cho một chiếc áo khoác như: trang phục mùa đông, đồ len ấm, v.v

#### 3.1 Tag list

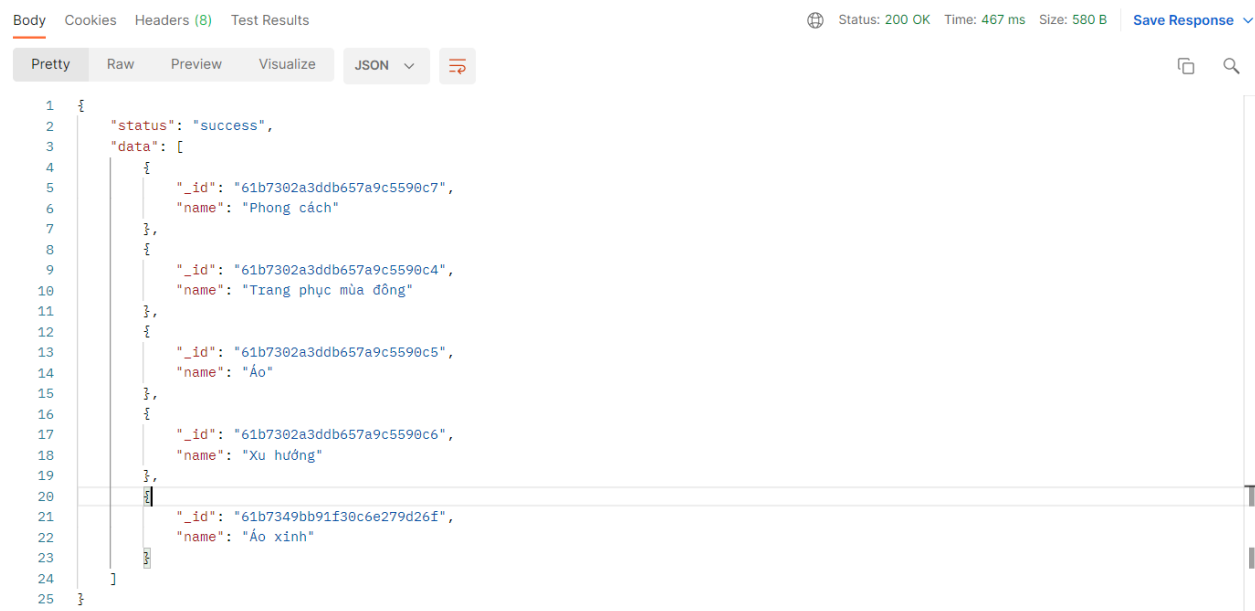
Url: <http://localhost:8080/api/tag>

Phương thức: Get

Authorization: Không cần thiết

Dữ liệu truyền vào: Không

Kết quả:



```
1 {
2   "status": "success",
3   "data": [
4     {
5       "_id": "61b7302a3ddb657a9c5590c7",
6       "name": "Phong cách"
7     },
8     {
9       "_id": "61b7302a3ddb657a9c5590c4",
10      "name": "Trang phục mùa đông"
11    },
12    {
13      "_id": "61b7302a3ddb657a9c5590c5",
14      "name": "Áo"
15    },
16    {
17      "_id": "61b7302a3ddb657a9c5590c6",
18      "name": "Xu hướng"
19    },
20    {
21      "_id": "61b7349bb91f30c6e279d26f",
22      "name": "Áo xinh"
23    }
24  ]
25 }
```

Dữ liệu được trả chính là list các tag trong database.

#### 3.2 Add tag

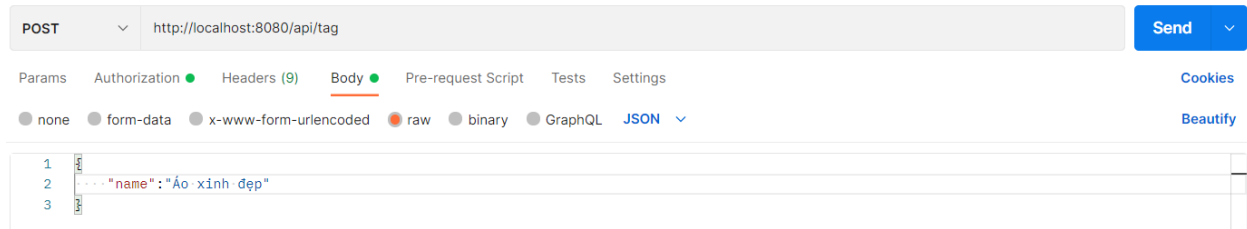
Url: <http://localhost:8080/api/tag>

Phương thức: Post

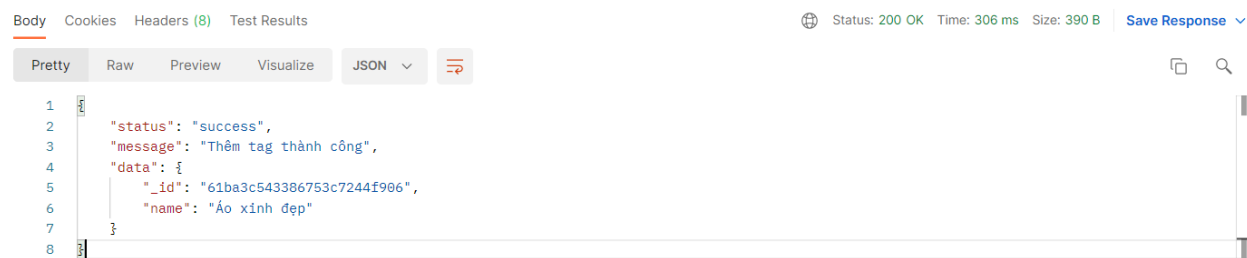
Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:



Kết quả:



Bên cạnh việc truyền dữ liệu qua Json thì sử dụng form-data cũng cho kết quả tương tự.

### 3.3 Delete tag

Url: <http://localhost:8080/api/tag/{tagID}>

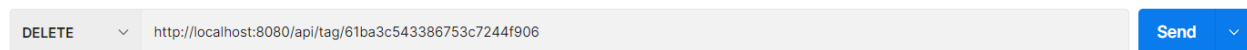
Phương thức: Delete

Tham số: tagID(Id của tag)

Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không có



Kết quả:

```
Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON
1
2  "status": "success",
3  "message": "Xóa tag thành công"
4
```

## 4 CRUD Category

Category là một phần quan trọng trong shopping cart, dùng để phân loại các sản phẩm cũng như dùng để lọc dữ liệu.

### 4.1 Category list

Url: <http://localhost:8080/api/category/all>

Phương thức: Get

Authorization: Không cần thiết

Dữ liệu truyền vào: Không

```
GET http://localhost:8080/api/category/all ... Send
```

Kết quả:

```
Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON
1
2  "status": "success",
3  "data": [
4    {
5      "_id": "61b7302a3ddb657a9c5590bc",
6      "name": "Khắc",
7      "active": true
8    },
9    {
10     "_id": "61b7302a3ddb657a9c5590be",
11     "name": "Áo khoác",
12     "active": true
13   },
14   {
15     "_id": "61b7302a3ddb657a9c5590bd",
16     "name": "Áo thun",
17     "active": true
18   },
19   {
20     "_id": "61b7302a3ddb657a9c5590bf",
21     "name": "Quần jean",
22     "active": true
23   },
24 ]
```

Dữ liệu được trả chính là list các category trong database.

## 4.2 Add Category

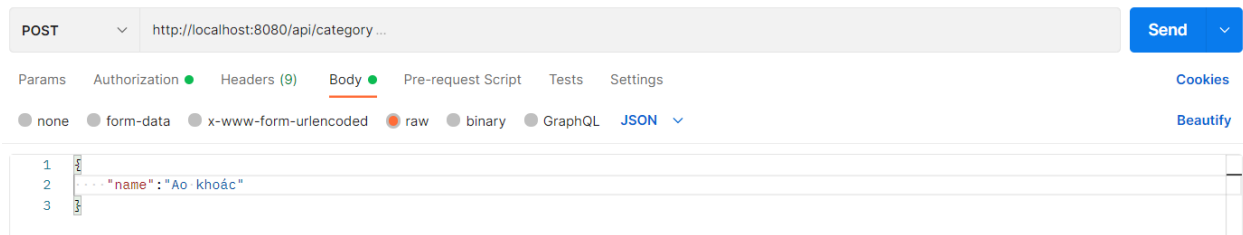
Url: <http://localhost:8080/api/category>

Phương thức: Post

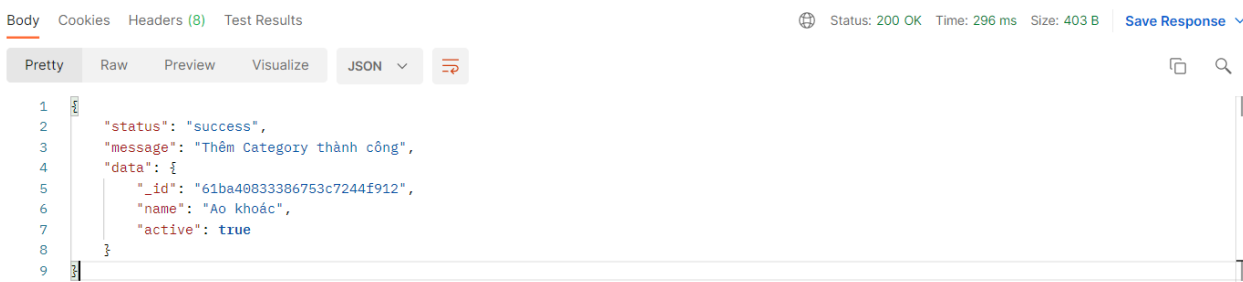
Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:



Kết quả:



Bên cạnh việc truyền dữ liệu qua Json thì sử dụng form-data cũng cho kết quả tương tự. API cũng áp dụng validate các thông tin truyền vào.

## 4.3 Delete Category

Url: <http://localhost:8080/api/category/{categoryID}>

Phương thức: Delete

Tham số: categoryID (Id của category)

Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không có

DELETE  Send

Kết quả:

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "message": "Xóa Category thành công"
4 }
```

Status: 200 OK Time: 82 ms Size: 326 B Save Response

#### 4.4 Update Category

Url: <http://localhost:8080/api/category/{categoryID}>

Phương thức: Put

Tham số: categoryID (Id của category)

Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:

PUT  Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "name": "Nón quai thao"
3 }
```

Hoặc thêm trường active:

PUT  Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "name": "Nón quai thao",
3   "active": true
4 }
```

Kết quả:

```
Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON
1
2 {"status": "success",
3  "message": "Cập nhật Category thành công",
4  "data": {
5    "_id": "61b7302a3ddb657a9c5590bf",
6    "name": "Nón quai thao",
7    "active": true
8  }
9 }
```

Bên cạnh việc truyền dữ liệu qua Json thì sử dụng form-data cũng cho kết quả tương tự. API cũng áp dụng validate các thông tin truyền vào.

## 5 CRUD Product

Product là một phần quan trọng trong shopping cart, dùng để lưu thông tin cũng như quản lý các sản phẩm.

### 5.1 Upload product picture

Việc tạo và cập nhật các product mà không có ảnh sản phẩm thì đúng là một thiếu sót. Do đó em đã thêm api upload vào. Việc tách upload ra khỏi form tạo sản phẩm sẽ giúp cho việc tạo sản phẩm linh hoạt hơn với json thay vì form data

Url: <http://localhost:8080/api/products/upload>

Phương thức: Post

Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:

POST http://localhost:8080/api/products/upload Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE	CONTENT TYPE	DESCRIPTION	Bulk Edit
<input checked="" type="checkbox"/> productPicture	17932425.png	Auto		

Kết quả:

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 475 ms Size: 385 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
{"status": "success",
"message": "Upload ảnh thành công",
"data": "/public/images/products/163959697124717932425.png"}
```

http://localhost:8080/public/images/products/163959697124717932425.png

Save

GET http://localhost:8080/public/images/products/163959697124717932425.png Send


Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies Headers (10) Test Results

Status: 200 OK Time: 303 ms Size: 154.25 KB Save Response



Dữ liệu được trả chính là url của bức ảnh để sử dụng cho việc thêm sản phẩm và cập nhật sản phẩm. Điểm hay của việc tách api upload ra là để dễ dàng trong việc tái sử dụng tài nguyên.

## 5.2 Add product

Url: <http://localhost:8080/api/products>

Phương thức: Post

Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:



POST <http://localhost:8080/api/products> Send

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	CONTENT TYPE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Quần hoa	Auto			
<input checked="" type="checkbox"/>	desc	Đây là áo phông 2d	Auto			
<input checked="" type="checkbox"/>	productPicture	/public/images/products/1639334910168...	Auto			
<input checked="" type="checkbox"/>	categoryID	61b62a02c7a28727ec32734a	Auto			
<input checked="" type="checkbox"/>	price	1000	Auto			

## Kết quả:

Body Cookies Headers (8) Test Results Status: 200 OK Time: 40 ms Size: 645 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "status": "success",
3    "message": "Thêm sản phẩm thành công",
4    "data": {
5      "name": "Quần hoa màu đỏ",
6      "desc": "Đây là áo phông 2d",
7      "price": 1000,
8      "productPicture": "/public/images/products/163933491016817932425.png",
9      "tags": [],
10     "active": true,
11     "_id": "61ba45333386753c7244f93c",
12     "createdAt": "2021-12-15T19:42:43.287Z",
13     "category": {
14       "_id": "61b7302a3ddb657a9c5590bc",
15       "name": "Khác"
16     }
17   }
18 }

```

Bên cạnh việc truyền dữ liệu qua form data thì sử dụng json cũng cho kết quả tương tự. Đối với trường hợp truyền sai categoryID, category được tạo cho product sẽ chuyển về dạng mặc định là “Khác”.

API cũng áp dụng validate các thông tin truyền vào.

### 5.3 Get Product with ID

Url: <http://localhost:8080/api/products/{ProductID}>

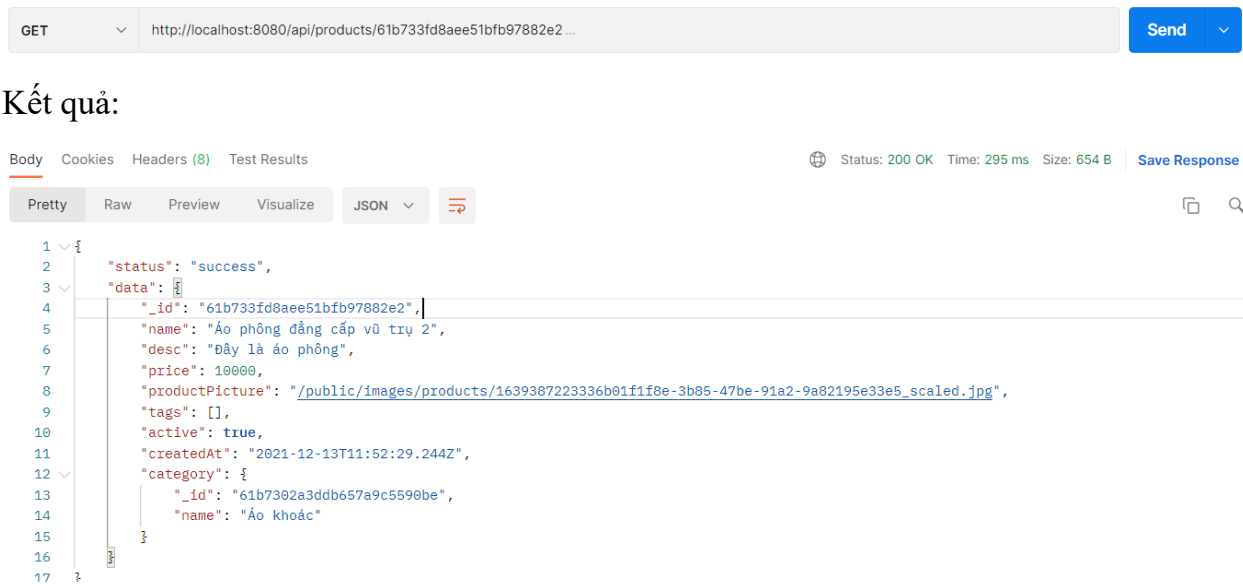
Phương thức: Get

Tham số: ProductID (Id của product)

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Không cần thiết

Dữ liệu truyền vào: Không có



Kết quả trả về là thông tin của product, đối với productID không hợp lệ, thông tin trả về sẽ là:

```
1 {
2   "status": "error",
3   "message": "Sản phẩm không tồn tại! Vui lòng thử lại với sản phẩm khác"
4 }
```

## 5.4 Update Product

Url: <http://localhost:8080/api/products/{ProductID}>

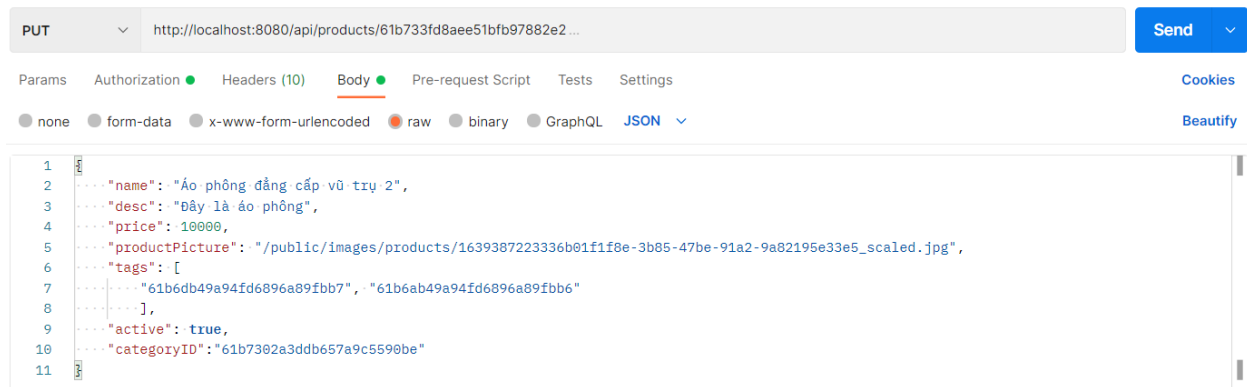
Phương thức: Put

Tham số: ProductID (Id của product)

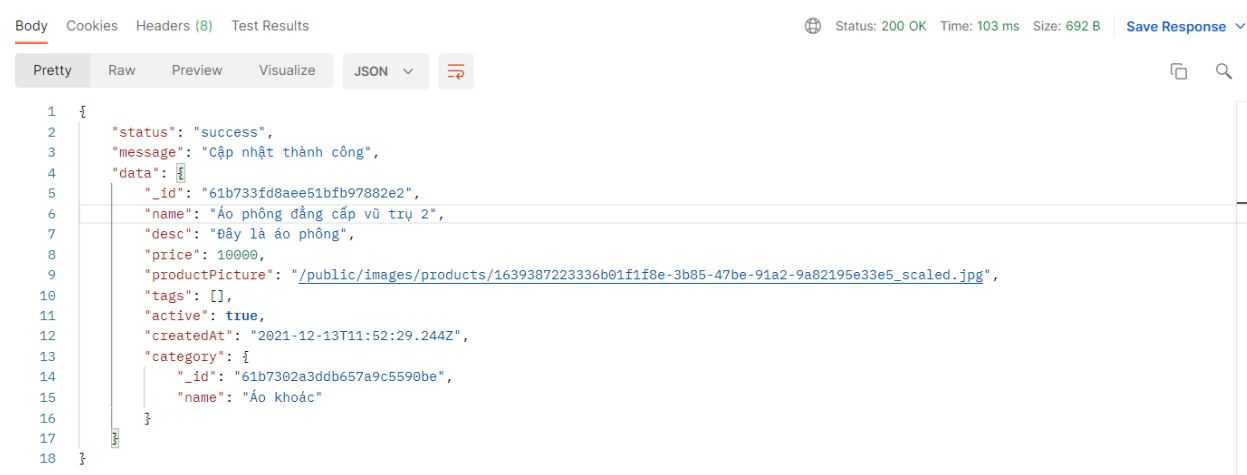
Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:



Kết quả:



Bên cạnh việc truyền dữ liệu qua form data thì sử dụng json cũng cho kết quả tương tự. Đối với trường hợp truyền sai categoryID, category được tạo cho product sẽ chuyển về dạng mặc định là “Khác”. Tương tự với tag thì sẽ chỉ thêm vào khi tag là hợp lệ. API cũng áp dụng validate các thông tin truyền vào.

### 5.5 Update Product

Url: <http://localhost:8080/api/products/{ProductID}>

Phương thức: Delete

Tham số: ProductID (Id của product)

Loại tài khoản có thể sử dụng được chức năng này: Admin(Admin được thêm vào từ database).

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không

DELETE ▼ http://localhost:8080/api/product/61b7326ed86140c2f4fea242 ... Send ▼

Kết quả:

Body Cookies Headers (8) Test Results Status: 200 OK Time: 58 ms Size: 330 B Save Response

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "status": "success",
3   "message": "Xóa sản phẩm thành công"
4 }
```

## 5.6 Phân trang Product

Xem sản phẩm dưới sạng trang

Url: <http://localhost:8080/api/products/pages/{page}>

Phương thức: Get

Tham số: page

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Không cần thiết

Dữ liệu truyền vào: Không có

GET ▼ http://localhost:8080/api/products/pages/2 Send ▼

Kết quả:

Body Cookies Headers (8) Test Results Status: 200 OK Time: 91 ms Size: 7.97 KB Save Response

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "totalPage": 5,
3   "currentPage": 2,
4   "productList": [
5     {
6       "_id": "61bb248d511425173af8ce15",
7       "tags": [],
8       "active": true,
9       "name": "Ergonomic Fresh Shoes",
10      "desc": "The Apollotech B340 is an affordable wireless mouse with reliable connectivity, 12 months battery life and modern design",
11      "price": 41,
12      "productPicture": "/public/images/products/product.png",
13      "createdAt": "2021-12-16T11:35:41.241Z",
14      "category": {
15        "_id": "61bb248b511425173af8cde9",
16        "name": "Khác"
17      }
18    },
19    {
20      "_id": "61bb248d511425173af8ce16",
21      "tags": [],
22      "active": true,
23      "name": "Intelligent Metal Ball",
24      "desc": "Andv shoes are designed to keeping in mind durability as well as trends. the most stvlsh range of shoes & sandals".
```

Kết quả trả về là danh sách sản phẩm với trang tương ứng.

## 6 Cart API

Giỏ hàng và đặt hàng là 2 chức năng quan trọng nhất của shopping cart.

### 6.1 Get Cart Infor

Lấy ra các sản phẩm trong giỏ hàng là một trong những yêu cầu cần thiết trong thương mại điện tử.

Url: <http://localhost:8080/api/cart>

Phương thức: Get

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không có

GET ▼ http://localhost:8080/api/cart... Send ▼

Kết quả:

```
Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON ▼ 🔍
1 {
2   "status": "success",
3   "message": "Lấy thông tin giỏ hàng thành công",
4   "cart": {
5     "userID": "61b7308aad5a7cd9dddb338",
6     "items": [
7       {
8         "productID": "61b733fd8aee51bf97882e2",
9         "quantity": 12,
10        "_id": "61ba44ae3386753c7244f933",
11        "name": "Áo phông đăng cấp vũ trụ 2",
12        "productPicture": "/public/images/products/1639387223336b01f1f0e-3b85-47be-91a2-9a82195e33e5_scaled.jpg",
13        "productPrice": 10000
14      },
15    ],
16    "total": 120000
17  }
18 }
```

Dữ liệu được trả về bao gồm thông tin giỏ hàng và tổng tiền.

## 6.2 Add product to cart

Url: <http://localhost:8080/api/cart/add>

Phương thức: Post

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:

KEY	VALUE	CONTENT TYPE	DESCRIPTION
productID	61b733fd8aee51bfb97882e2	Auto	
quantity	6	Auto	

Kết quả:

```
1 {
2   "status": "success",
3   "message": "Thêm sản phẩm vào giỏ hàng thành công",
4   "item": {
5     "productID": "61b733fd8aee51bfb97882e2",
6     "quantity": 18,
7     "name": "Áo phông đẳng cấp vũ trụ 2",
8     "productPicture": "/public/images/products/1639387223336b01f1f8e-3b85-47be-91a2-9a82195e33e5_scaled.jpg",
9     "productPrice": 10000
10  }
11 }
```

Dữ liệu được trả về chính là thông tin sản phẩm được thêm vào. Trong trường hợp mã sản phẩm sai thì sẽ có tin nhắn trả về:

```
{
  "status": "error",
  "message": "Thông tin sản phẩm không hợp lệ"
}
```

API cũng hoạt động với dữ liệu json.

## 6.3 Subtract product from cart

Url: <http://localhost:8080/api/cart/subtract>

Phương thức: Post

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào:

POST ⌵ http://localhost:8080/api/cart/subtract ... Send ⌵

Params Authorization ● Headers (9) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	KEY	VALUE	CONTENT TYPE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	productID	61b733fd8aee51bfb97882e2	Auto			
<input checked="" type="checkbox"/>	quantity	3	Auto			

Kết quả:

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK Time: 19 ms Size: 599 B Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ 🔍

```
1  {
2    "status": "success",
3    "message": "Giảm số lượng sản phẩm trong giỏ hàng thành công",
4    "item": {
5      "productID": "61b733fd8aee51bfb97882e2",
6      "quantity": 12,
7      "name": "Áo phông đẳng cấp vũ trụ 2",
8      "productPicture": "/public/images/products/1639387223336b01f1f8e-3b85-47be-91a2-9a82195e33e5_scaled.jpg",
9      "productPrice": 10000
10   }
11 }
```

Dữ liệu được trả về chính là thông tin sản phẩm được giảm bớt vào. Trong trường hợp mã sản phẩm sai thì sẽ có tin nhắn trả về:

```
1  {
2    "status": "error",
3    "message": "Thông tin sản phẩm không hợp lệ"
4  }
```

Nếu như số lượng sản phẩm về 0 thì sẽ xóa sản phẩm đó ra khỏi giỏ hàng.

```
1  {
2    "status": "success",
3    "message": "Sản phẩm đã được xóa ra khỏi giỏ hàng",
4    "item": {}
5  }
```

API cũng hoạt động với dữ liệu json.

## 6.4 Remove product from cart

Url: <http://localhost:8080/api/cart/remove>

Phương thức: Post

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không

POST ▼ <http://localhost:8080/api/cart/remove/61b88eaf1cab5c5ba6e89bbc...> Send ▼

Kết quả:

Body Cookies Headers (8) Test Results ⌐ Status: 200 OK Time: 43 ms Size: 364 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ⌐

```
1 {
2   "status": "success",
3   "message": "Sản phẩm đã được xóa ra khỏi giỏ hàng",
4   "items": []
5 }
```

Dữ liệu được trả về chính là thông tin giỏ hàng sau khi sản phẩm trên được xóa ra khỏi giỏ hàng vào.

## 7 Order API

Giỏ hàng và đặt hàng là 2 chức năng quan trọng nhất của shopping cart.

### 7.1 Create Order

Tạo ra đơn đặt hàng cũng là một yêu cầu quan trọng trong shopping cart

Url: <http://localhost:8080/api/order/create>

Phương thức: Get

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

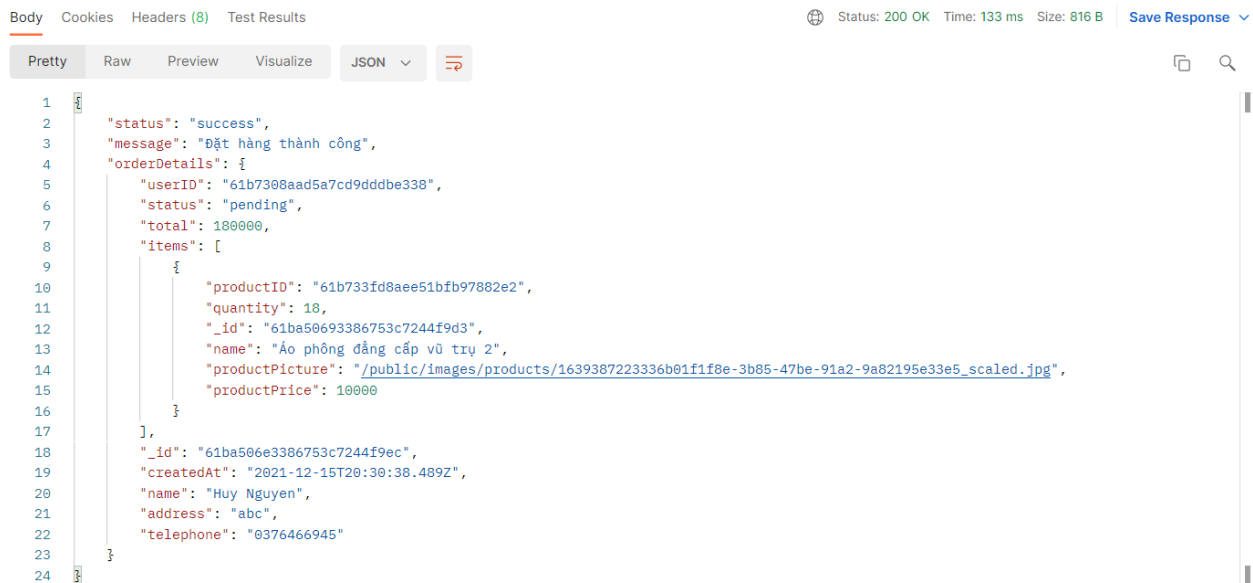
Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không có

POST ▼ <http://localhost:8080/api/order/create> Send ▼



Kết quả:



The screenshot shows a web browser's developer tools with the 'Test Results' tab selected. The response is a JSON object with the following structure:

```
1 {
2   "status": "success",
3   "message": "Đặt hàng thành công",
4   "orderDetails": {
5     "userID": "61b7308aad5a7cd9dddb338",
6     "status": "pending",
7     "total": 180000,
8     "items": [
9       {
10        "productID": "61b733fd8aee51bfb97882e2",
11        "quantity": 18,
12        "_id": "61ba50693386753c7244f9d3",
13        "name": "Áo phông đăng cấp vũ trụ 2",
14        "productPicture": "/public/images/products/1639387223336b01f1f0e-3b85-47be-91a2-9a82195e33e5_scaled.jpg",
15        "productPrice": 10000
16      }
17    ],
18    "_id": "61ba506e3386753c7244f9ec",
19    "createdAt": "2021-12-15T20:30:38.489Z",
20    "name": "Huy Nguyen",
21    "address": "abc",
22    "telephone": "0376466945"
23  }
24 }
```

Dữ liệu được trả về chính là thông tin đơn hàng, trong trường hợp giỏ hàng không có sản phẩm nào thì hóa đơn sẽ không được tạo.

## 7.2 Get Order list

Url: <http://localhost:8080/api/order>

Phương thức: Get

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không có

GET

<http://localhost:8080/api/order>

Send

Kết quả:

```
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 226 ms Size: 1.42 KB Save Response
Pretty Raw Preview Visualize JSON
1
2 "status": "success",
3 "orderList": [
4   {
5     "_id": "61b9b64d6bfbfaa514497419",
6     "userID": "61b7308aad5a7cd9dddb338",
7     "status": "pending",
8     "total": 0,
9     "items": [],
10    "createdAt": "2021-12-15T09:33:01.932Z",
11    "name": "Huy Nguyen",
12    "address": "abc",
13    "telephone": "0376466945"
14  },
15  {
16    "_id": "61b9b7494bcd4d333c6c444c",
17    "userID": "61b7308aad5a7cd9dddb338",
18    "status": "pending",
19    "total": 0,
20    "items": [
21      {
22        "productID": "61b733fd8aee51bfb97882e2",
23        "quantity": 6,
24        "_id": "61b9b6d86bfbfaa514497427",
25        "name": "Áo phông đăng cấp vũ trụ 2",
```

Dữ liệu được trả về chính là danh sách các hóa đơn của người dùng.

### 7.3 Get Order with id

Url: <http://localhost:8080/api/order/{orderID}>

Phương thức: Get

Tham số: orderID {id của order}

Loại tài khoản có thể sử dụng được chức năng này: Toàn bộ

Authorization: Sử dụng tương tự 2.3

Dữ liệu truyền vào: Không có

POST	<a href="http://localhost:8080/api/order/61b9b64d6bfbfaa514497419">http://localhost:8080/api/order/61b9b64d6bfbfaa514497419</a>	Send
------	---	------

Kết quả:

```
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 14 ms Size: 514 B Save Response
Pretty Raw Preview Visualize JSON
1 {
2   "status": "success",
3   "orderDetails": [
4     {
5       "_id": "61b9b64d6bfbaa514497419",
6       "userID": "61b7308aad5a7cd9dddb338",
7       "status": "pending",
8       "total": 0,
9       "items": [],
10      "createdAt": "2021-12-15T09:33:01.932Z",
11      "name": "Huy Nguyen",
12      "address": "abc",
13      "telephone": "0376466945"
14    }
15  ]
16 }
```

Dữ liệu được trả về chính là thông tin đơn hàng với id được chọn, trong trường hợp id sản phẩm là sai thì đơn hàng sẽ không được trả về.

## TÀI LIỆU THAM KHẢO

- [1]. <https://nvoulgaris.com/designing-a-restful-shopping-cart/>
- [2]. <https://resources.fabric.inc/blog/answers/shopping-cart-database-design>
- [3]. <https://viblo.asia/p/xay-dung-restful-api-don-gian-voi-nodejs-1Je5EdewlnL>