

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**



NGUYỄN BÁ ĐẠT

**NGHIÊN CỨU XÂY DỰNG API DỰA TRÊN
DJANGO REST FRAMEWORK**

Mã số sinh viên: 1751010022

**ĐỒ ÁN NGÀNH
NGÀNH KHOA HỌC MÁY TÍNH**

Giảng viên hướng dẫn: TS. Trương Hoàng Vinh

TP. HỒ CHÍ MINH, 2020

LỜI CẢM ƠN

Em xin chân thành cảm ơn trường Đại học Mở TP.HCM cùng khoa công nghệ thông tin đã tạo điều kiện cho em được học tập, rèn luyện, phát triển bản thân giúp em có được kiến thức, các kỹ năng cần thiết và nền tảng để phát triển sự nghiệp của em sau trong con đường tiếp theo.

Em cũng xin chân thành cảm ơn các quý thầy cô, đội ngũ giảng viên và công nhân viên của trường đã hỗ trợ, giúp đỡ, dạy dỗ em trong suốt quá trình học tập tại trường.

Em xin gửi lời cảm ơn chân thành đến Thầy TS. Trương Hoàng Vinh đã giúp đỡ, hướng dẫn em tận tình trong quá trình nghiên cứu thực hiện đồ án và bài báo cáo này.

Với kinh nghiệm và trình độ còn hạn chế nên trong quá trình thực hiện đồ án vẫn còn nhiều thiếu sót, rất mong nhận được sự góp ý, chỉ dẫn của quý thầy cô.

Em xin kính chúc quý thầy cô thật nhiều sức khỏe và đạt được nhiều thành công, em xin chân thành cảm ơn.

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

MỤC LỤC

| | | |
|-------------------------|--|----|
| Chương 1. | TỔNG QUAN | 7 |
| 1.1. | Đặt vấn đề và lý do chọn đề tài | 7 |
| 1.2. | Phạm vi đề tài và đối tượng nghiên cứu | 7 |
| 1.3. | Bố cục báo cáo | 8 |
| Chương 2. | CƠ SỞ LÝ THUYẾT | 9 |
| 2.1. | Django Rest Framework..... | 9 |
| 2.2. | MySQL..... | 10 |
| 2.3. | Entity-Attribute-Value Model | 10 |
| 2.4. | API | 13 |
| 2.5. | REST API..... | 15 |
| 2.6. | OAuth 2 | 17 |
| 2.7. | OTP and Mail | 19 |
| Chương 3. | XÂY DỰNG API BÁN HÀNG TRỰC TUYẾN..... | 20 |
| 3.1. | Xác định yêu cầu hệ thống | 20 |
| 3.2. | Thiết kế cơ sở dữ liệu | 22 |
| 3.3. | Xây dựng trang quản trị (Admin)..... | 26 |
| 3.3.1. | Giai đoạn 1: Tạo cơ sở dữ liệu | 26 |
| 3.3.2. | Giai đoạn 2: Tạo trang Admin..... | 27 |
| 3.4. | Xây dựng các API theo yêu cầu hệ thống | 33 |
| 3.4.1. | API Product | 38 |
| 3.4.2. | API Customer | 45 |
| 3.4.3. | API Order | 62 |
| Chương 4. | KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN | 80 |
| 4.1. | Kết quả đạt được | 80 |
| 4.2. | Hạn chế và hướng phát triển | 80 |
| TÀI LIỆU THAM KHẢO..... | | 82 |

DANH MỤC HÌNH

| | |
|---|----|
| Hình 2.1. Mô hình dữ liệu EAV (Nguồn: Viblo) | 11 |
| Hình 2.2. Sơ đồ VD mô hình EAV | 11 |
| Hình 2.3. Sơ đồ mô tả API (Nguồn: Medium) | 13 |
| Hình 2.4. Ứng dụng của API (Nguồn: itviec.com)..... | 14 |
| Hình 2.5. Sơ đồ minh họa OAuth (Nguồn: digitalocean.com)..... | 18 |
| Hình 2.6. Ví dụ mail OTP code | 19 |
| Hình 3.1. Sơ đồ usecase..... | 20 |
| Hình 3.2. Phân quyền nhân viên..... | 21 |
| Hình 3.3. Sơ đồ class | 22 |
| Hình 3.4 Sơ đồ dữ liệu tổng quát (EER). | 22 |
| Hình 3.5. Cơ sở dữ liệu Product | 23 |
| Hình 3.6. Cơ sở dữ liệu Order | 24 |
| Hình 3.7. Các trạng thái của đơn hàng | 24 |
| Hình 3.8. Cơ sở dữ liệu Customer | 25 |
| Hình 3.9. Cấu trúc dự án..... | 26 |
| Hình 3.10. Danh sách API Product..... | 33 |
| Hình 3.11. Danh sách API Order..... | 33 |
| Hình 3.12. Danh sách API Customer | 34 |
| Hình 3.13. Ví dụ mẫu gọi API..... | 37 |
| Hình 3.14. API Get Category GET: Ok | 38 |
| Hình 3.15. API Get Manufacturer GET: Ok | 39 |
| Hình 3.16. API Get Discount GET: Ok..... | 40 |
| Hình 3.17. API Get Product GET: Ok..... | 42 |
| Hình 3.18. API Get Product Attribute: Ok | 43 |
| Hình 3.19. API Get Product Attribute Value: Ok | 44 |
| Hình 3.20. API Sign Up: Ok | 45 |
| Hình 3.21. API Sign Up: Tên tài khoản đã tồn tại | 45 |
| Hình 3.22. API Sign Up: Sai thông tin ngày sinh | 46 |
| Hình 3.23. API Sign Up: Sai thông tin SĐT | 46 |
| Hình 3.24. API Sign In: Ok..... | 47 |
| Hình 3.25. API Sign In: Sai thông tin đăng nhập..... | 48 |
| Hình 3.26. API Sign In: Thiếu thông tin đăng nhập..... | 48 |
| Hình 3.27. Thêm token xác thực vào headers của API | 49 |
| Hình 3.28. Xác thực token, token không tồn tại..... | 49 |
| Hình 3.29. Xác thực token, token hết hạn | 49 |
| Hình 3.30. API Sign Out: Ok | 51 |
| Hình 3.31. API Sign Out: Token quá hạn..... | 51 |
| Hình 3.32. API Get Customer Info: Ok..... | 52 |
| Hình 3.33. API Add Tel Number: Ok | 52 |
| Hình 3.34. API Edit Tel Number: Ok..... | 53 |
| Hình 3.35. API Edit Tel Number: Sai mã SĐT | 53 |
| Hình 3.36. API Delete Tel Number: Ok..... | 53 |
| Hình 3.37. API Add Ship Address: Ok | 54 |
| Hình 3.38. API Delete Ship Address: Ok..... | 54 |

| | |
|---|----|
| Hình 3.39. API Edit Ship Address: Ok..... | 55 |
| Hình 3.40. API Edit Ship Address: Sai mã địa chỉ..... | 55 |
| Hình 3.41. API Edit Customer..... | 56 |
| Hình 3.42. API Edit Customer: Sai mật khẩu..... | 56 |
| Hình 3.43. API Edit Customer: Sai dữ liệu ngày sinh..... | 57 |
| Hình 3.44. API Edit Customer: Gửi thiếu thông tin..... | 57 |
| Hình 3.45. API Change Password: Ok | 58 |
| Hình 3.46. API Change Password: Mật khẩu không khớp..... | 58 |
| Hình 3.47. API Change Password: Sai mật khẩu cũ | 58 |
| Hình 3.48. API Request Restore Password: Ok | 59 |
| Hình 3.49. Mail mã xác thực khôi phục mật khẩu | 59 |
| Hình 3.50. API Request Restore Password: Email không tồn tại..... | 59 |
| Hình 3.51. API Change Password OTP: Ok..... | 61 |
| Hình 3.52. API Get Cart: Ok | 62 |
| Hình 3.53. API Add Item to Cart: Ok | 62 |
| Hình 3.54. API Add Item to Cart: Sai kiểu dữ liệu | 63 |
| Hình 3.55. API Add Item to Cart: Mã sản phẩm không tồn tại..... | 63 |
| Hình 3.56. API Add Item to Cart: Số lượng không hợp lệ..... | 63 |
| Hình 3.57. API Add Item to Cart: Số lượng tăng tự động..... | 64 |
| Hình 3.58. API Del Item form Cart: Ok | 64 |
| Hình 3.59. API Del Item form Cart: Sản phẩm không có trong giỏ..... | 64 |
| Hình 3.60. API Customer Add Order: Ok | 65 |
| Hình 3.61. Mail thông báo khách hàng đặt hàng thành công | 65 |
| Hình 3.62. API Get Customer Order (GET): Ok..... | 69 |
| Hình 3.63. API Get Customer Order (POST): DS đơn hàng có trạng thái PROCESSING | 69 |
| Hình 3.64. API Get Customer Order Detail: Ok | 70 |
| Hình 3.65. API Get Customer Order Detail: Mã đơn hàng không tồn tại..... | 70 |
| Hình 3.66. API Get Customer Order Detail: Mã đơn hàng không phải số..... | 70 |
| Hình 3.67. API Customer Cancel Order: Ok..... | 71 |
| Hình 3.68. Mail thông báo KH hủy đơn thành công | 71 |
| Hình 3.69. API Customer Return Order: Ok | 73 |
| Hình 3.70. Mail thông báo KH trả hàng thành công | 73 |
| Hình 3.72. API Admin Ship Order: Ok | 76 |
| Hình 3.73. Mail thông báo KH đơn hàng đang được vận chuyển đi..... | 76 |
| Hình 3.74. API Admin Done Order: Ok..... | 78 |
| Hình 3.75. Mail thông báo KH đơn hàng đã hoàn tất..... | 78 |

Chương 1. TỔNG QUAN

Giới thiệu tổng quan về đề tài nghiên cứu và phần báo cáo. Bao gồm lý do lựa chọn đề tài, phạm vi đề tài và đối tượng nghiên cứu.

1.1. Đặt vấn đề và lý do chọn đề tài

Với tốc độ phát triển của ngành công nghệ thông tin nói chung và thương mại điện tử nói riêng thì nhu cầu của người tiêu dùng ngày càng cao, việc mua sắm không chỉ đơn thuần là tại các địa điểm bán hàng nữa mà còn trên các trang web mua sắm, ứng dụng mua hàng online.

Việc phát triển một hệ thống web bán hàng cũng vô cùng phức tạp, mất nhiều thời gian, chi phí và không chỉ trên web mà còn các ứng dụng di động khác nữa. Vậy nếu mỗi lần muốn triển khai bán hàng trên một ứng dụng hay phương tiện khác, ta lại phải thiết kế và xây dựng lại hệ thống trên phương tiện đó?

Với công nghệ hiện nay, chúng ta đã có giải pháp để giải quyết cho vấn đề đó là chỉ cần xây dựng một hệ thống API cho cửa hàng và khi triển khai bán hàng trên các phương tiện khác, ta chỉ cần gọi lại các API để sử dụng trên các phương tiện đó, như vậy ta có thể tiết kiệm được phần lớn thời gian và chi phí để phát triển và mở rộng hệ thống bán hàng của mình. Về đề tài đồ án của em cũng chính là giải pháp cho vấn đề trên, nghiên cứu xây dựng API dựa trên Django Rest Framework.

1.2. Phạm vi đề tài và đối tượng nghiên cứu

Trong đồ án này, em sẽ thiết kế một hệ thống API hỗ trợ cho việc bán hàng online. Mặt hàng đầu tiên em hướng đến để thiết kế cơ sở dữ liệu cho hệ thống là các đồ dùng điện tử. Hệ thống sẽ hỗ trợ cho khách hàng việc xem chọn lựa hàng hóa cần mua và bỏ vào giỏ hàng của mình sau đó tiến hành đặt hàng khi đã tạo tài khoản trong hệ thống. Khách hàng cũng có thể quản lý các thông tin tài khoản và theo dõi các đơn hàng của mình. Ngoài ra em cũng sẽ xây dựng một web quản trị cùng các chức năng cơ bản để giúp chủ cửa hàng và nhân viên có thể quản lý các thông tin hàng hóa và đơn hàng của cửa hàng.

Hướng phát triển của đề tài là xây dựng một hệ thống API, một cơ sở dữ liệu có thể mở rộng được để bán tất cả mặt hàng có thể bán được và tích hợp được thêm các dịch vụ hỗ trợ thanh toán và giao hàng từ bên thứ ba. Để có thể phát triển hệ thống nhanh

chống em sử dụng một framework thuộc ngôn ngữ Python là Django cùng với hệ quản trị cơ sở dữ liệu MySQL cho đề tài của mình.

1.3. Bố cục báo cáo

Phần báo cáo của em bao gồm 4 chương chính, về nội dung của các chương còn lại bao gồm: Chương 2 em sẽ giới thiệu về các cơ sở lý thuyết của những công nghệ em sử dụng để phát triển đề tài. Chương 3 là gồm các nội dung, giai đoạn, quá trình em thực hiện đồ án này và kết quả đạt được của từng giai đoạn đó. Chương 4 em sẽ tổng kết lại đề tài cũng như kết quả và những thứ em đã học, rút ra được sau quá trình làm đồ án này của mình.

Chương 2. CƠ SỞ LÝ THUYẾT

Trong phần báo cáo này bao gồm các công nghệ, cơ sở lý thuyết, khái niệm của các công nghệ được em tìm hiểu và ứng dụng vào đề tài nghiên cứu của mình.

2.1. Django Rest Framework

Django là một framework web bậc cao của ngôn ngữ Python, giúp việc phát triển xây dựng ứng dụng web nhanh chóng, gọn gàng, và có tính ứng dụng thực tế. Django được xây dựng bởi các nhà phát triển dày dặn kinh nghiệm, nó giúp xử lý được phần lớn sự phức tạp của việc phát triển web, vì vậy chúng ta có thể dành tập trung vào việc phát triển ứng dụng web của mình mà không phải xây dựng lại nền tảng đã có của Django. Dưới đây bao gồm các ưu điểm và cũng chính là lý do em lựa chọn Django:

- *Phát triển nhanh chóng:* Django được thiết kế để giúp các nhà phát triển đưa ứng dụng web của mình từ ý tưởng đến hoàn thiện nhanh nhất có thể.
- *Hỗ trợ đầy đủ tác vụ:* Django được bao gồm rất nhiều tính năng mà bạn có thể sử dụng để xử lý các tác vụ phát triển web thông thường chẳng hạn như xác thực người dùng (User Authentication), Administration, Site map và nhiều tác vụ khác.
- *Đảm bảo tính bảo mật:* Django rất coi trọng việc bảo mật và giúp các nhà phát triển tránh được nhiều lỗi bảo mật phổ biến, chẳng hạn như SQL Injection, giả mạo yêu cầu cross-site. Hệ thống User Authentication của Django cung cấp cách bảo mật tốt nhất để quản lý tài khoản người dùng và mật khẩu của họ.
- *Khả năng mở rộng vượt trội:* Một số trang web doanh nghiệp lớn trên thế giới sử dụng Django để đáp ứng nhu cầu lưu lượng truy cập lớn vì khả năng mở rộng quy mô nhanh chóng và linh hoạt của nó.
- *Tính linh hoạt cao:* Các công ty, tổ chức và kể cả chính phủ đã sử dụng Django để xây dựng mọi thứ từ hệ thống quản lý nội dung đến mạng xã hội và cả nền tảng khoa học máy tính.

Django REST Framework (DRF) là một bộ công cụ linh hoạt và mạnh mẽ dùng để xây dựng các API web. Các tính năng và cũng là lý do nên sử dụng REST framework của Django:

- Các API có thể duyệt được là một thắng lợi lớn về khả năng sử dụng cho các nhà phát triển.
- Có những chính sách xác thực bao gồm các thư viện như OAuth1a và OAuth2.

- Serialization (Sự tuần tự hóa, chuỗi hóa) hỗ trợ cho cả nguồn dữ liệu ORM và non-ORM.
- Có thể tùy chỉnh mọi thứ (Cứ sử dụng các hàm cơ bản bình thường nếu bạn không cần thêm các tính năng tối ưu hơn).
- Có nhiều tài liệu tham khảo phong phú và một cộng đồng hỗ trợ lớn.
- Được sử dụng và tin tưởng bởi những công ty quốc tế uy tín như là Mozilla, Red Hat và Heroku.

Sự khác biệt giữa Django và DRF là cách mà chúng được sử dụng. Django thì được dùng để xây dựng, tạo một ứng dụng web còn DRF thì dùng để tạo các API.

2.2. MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở. MySQL là một ngôn ngữ truy vấn có cấu trúc, là cơ sở dữ liệu quan hệ tổ chức dữ liệu vào một hay nhiều bảng dữ liệu mà các kiểu dữ liệu có thể sẽ liên kết với nhau và các liên kết này giúp tạo cấu trúc cho dữ liệu. SQL là một ngôn ngữ mà lập trình viên sử dụng để tạo, chỉnh sửa, trích xuất dữ liệu từ dữ liệu quan hệ, cũng như quản lý việc truy cập vào cơ sở dữ liệu. MySQL cũng được các web lớn tin dùng như Facebook, Youtube, Twitter, Google, ...

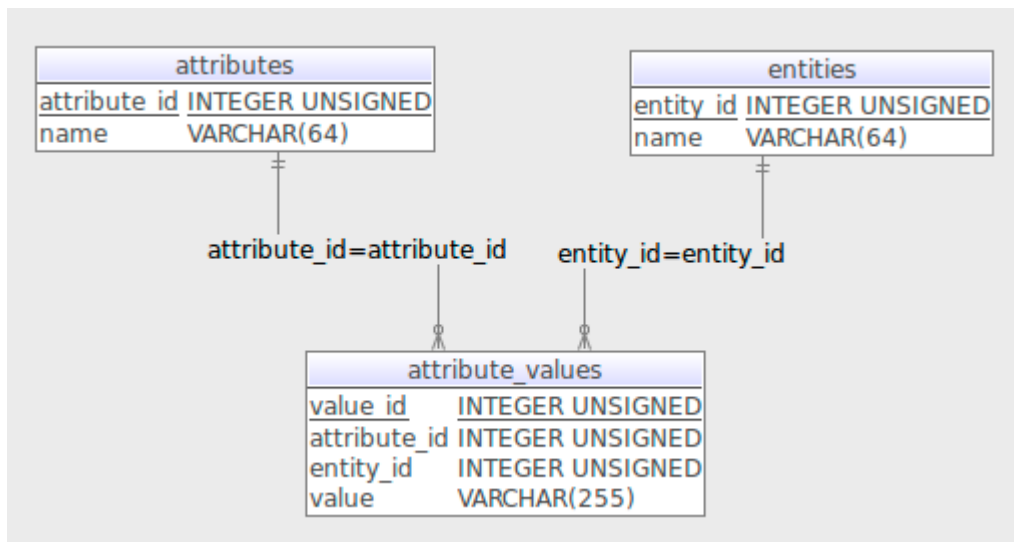
Sau đây là các ưu điểm và cũng là lý do em sử dụng MySQL làm hệ quản trị dữ liệu cho hệ thống và đề tài nghiên cứu của mình:

- Tính linh hoạt và dễ sử dụng: Do là mã nguồn mở, nên ta hoàn toàn có thể thay đổi source code theo nhu cầu phát triển mà không mất bất kì khoản chi phí nào. Và việc cài đặt MySQL cũng vô cùng đơn giản và nhanh chóng.
- Có tiêu chuẩn chung: Do MySQL đã được sử dụng trong ngành công nghệ và dữ liệu trong nhiều năm qua, nên nó là một kỹ năng cơ bản nhất của một chuyên gia lập trình.
- An toàn: Khi chọn một phần mềm quản trị hệ cơ sở dữ liệu thì tính an toàn luôn là vấn đề được coi trọng nhất, vì vậy MySQL luôn đặt vấn đề bảo mật ở tiêu chuẩn rất cao.

2.3. Entity-Attribute-Value Model

Entity-Attribute-Value Pattern (dịch: mô hình thực thể - thuộc tính – giá trị) (EAV Pattern) là một mô hình dữ liệu có thể dễ dàng mở rộng số lượng thuộc tính của các thực thể.

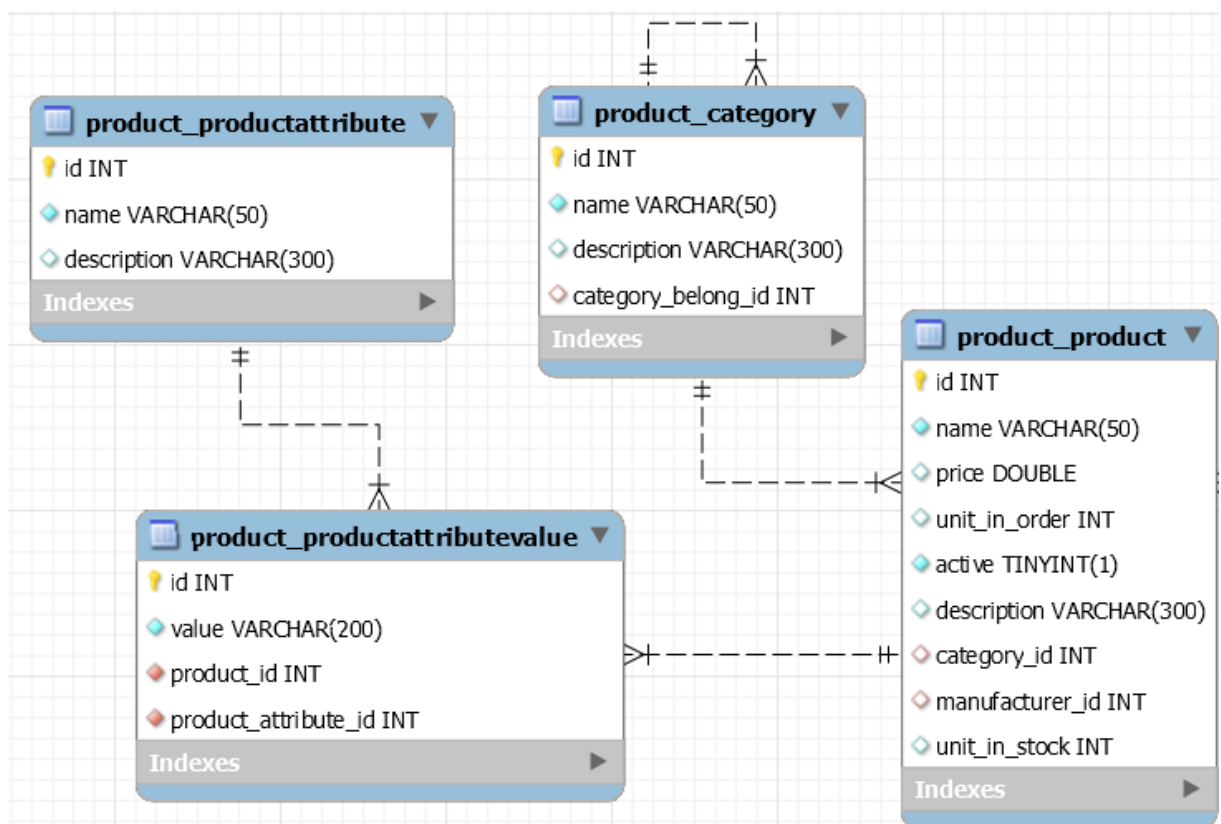
EAV là một cấu trúc thiết kế cơ sở dữ liệu trong Magento.



Hình 2.1. Mô hình dữ liệu EAV (Nguồn: [Viblo](#))

- Entity: Bảng chứa thông tin cơ bản của thực thể.
- Attribute: Bảng chứa các thuộc tính ta có thể thêm mới vào thực thể về sau.
- Value: Bảng chứa giá trị với 2 khóa ngoại là thuộc tính là thực thể.

VD: Với cấu trúc bảng dữ liệu sau



Hình 2.2. Sơ đồ VD mô hình EAV

Ta có:

- Entity là bảng product_product.
- Attribute là bảng product_productattribute.
- Value là bảng product_productattributevalue.

Với cách thiết kế trên, hệ thống có thể dễ dàng mở rộng ra sau này, bán nhiều loại sản phẩm hơn (điện thoại, đồ gia dụng, quần áo, ...). Giả sử công ty ban đầu chỉ bán điện thoại với 1 category là điện thoại với các product là các điện thoại khác nhau. Sau này mở rộng, công ty cần mở rộng hệ thống để bán thêm nhiều mặt hàng điện tử khác, category điện thoại khi này sẽ có category_belong là category đồ điện tử và trong category đồ điện tử sẽ bao gồm nhiều các category khác như laptop, tivi, tủ lạnh, Và các sản phẩm khi này không chỉ còn là điện thoại mà gồm nhiều mặt hàng khác với nhiều thuộc tính khác nhau thì thiết kế trên chỉ cần thêm 1 attribute mới vào và tạo value của attribute cho product đó.

Product:

| id | name | price | unit_in_stock | unit_in_order | active |
|----|------------------------------------|----------|---------------|---------------|--------|
| 1 | Asus ROG Zephyrus M GU502GU AZ090T | 32990000 | 20 | 0 | 1 |

ProductAttribute:

| id | name |
|----|-------------------|
| 1 | Thông số màn hình |
| 2 | CPU |
| 3 | Ram |
| 4 | Ổ cứng |
| 5 | Pin |
| 6 | Ảnh |

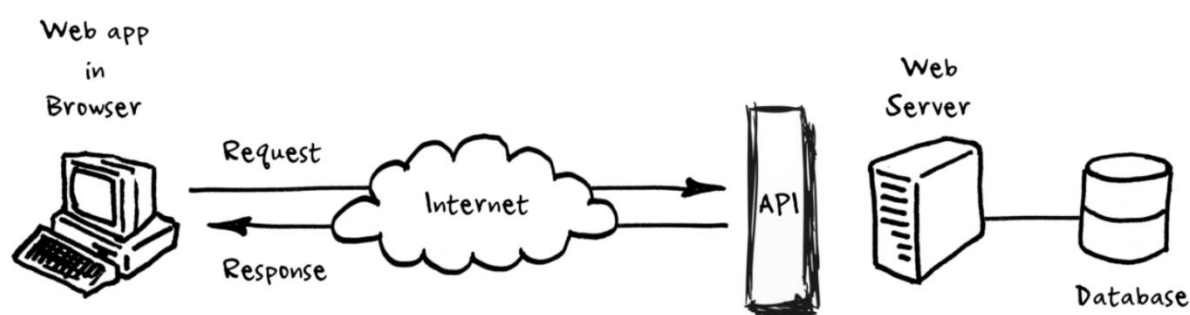
ProductAttributeValue:

| id | value | product_id | product_attribute_id |
|----|---|------------|----------------------|
| 1 | 15.6" FHD (1920 x 1080) IPS, 100% sRGB, 240... | 1 | 1 |
| 2 | Intel Core i7-9750H 2.6GHz up to 4.5GHz 12MB | 1 | 2 |
| 3 | 16GB DDR4 2666MHz Onboard (1x SO-DIMM so... | 1 | 3 |
| 4 | 512GB SSD PCIE G3X4 (Support RAID 0) (2 slots) | 1 | 4 |
| 5 | 4 Cell 76WHr | 1 | 5 |
| 6 | https://product.hstatic.net/1000026716/produ... | 1 | 6 |

Vì những đặc điểm trên nên em chọn mô hình EAV để thiết kế dữ liệu sản phẩm cho hệ thống của mình để có thể dễ dàng mở rộng, thêm mới các thuộc tính cho sản phẩm về sau.

2.4. API

Application Programming Interface (API) là một hàm dùng để truyền dữ liệu (giao tiếp) giữa một phần mềm này với phần mềm khác và trong đó, nó cũng chứa các điều khoản nhất định cho việc trao đổi đó.



Hình 2.3. Sơ đồ mô tả API (Nguồn: [Medium](#))

API không phải là cơ sở dữ liệu hay máy chủ, nó là một hàm hoặc đoạn mã dùng để quản lý các điểm truy cập cho máy chủ. API có thể nói đơn giản nó là một cách thức giao tiếp riêng giữa các ứng dụng trên nhiều nền tảng khác nhau từ đó giúp cho việc phát triển hệ thống thuận tiện hơn. API cũng là một công cụ có mã nguồn mở, vì vậy có thể dễ dàng kết nối và sử dụng khi có internet. Cũng vì là mã nguồn mở, các ứng dụng khác có thể truy cập dễ dàng khi có internet nên API thường gặp các vấn đề về bảo mật.

API cũng được chia làm 3 loại chính theo chính sách sử dụng của chúng:

- Private API: Các API này được thiết kế để cải thiện các giải pháp và dịch vụ trong một tổ chức. Các nhà phát triển hoặc nhà thầu nội bộ có thể sử dụng các API này để tích hợp các hệ thống hoặc ứng dụng CNTT của công ty, xây dựng hệ thống mới hoặc ứng dụng hướng tới khách hàng tận dụng các hệ thống hiện có. Ngay cả khi các ứng dụng có sẵn công khai, bản thân giao diện vẫn chỉ có sẵn cho những người làm việc trực tiếp với nhà xuất bản API. Chiến lược riêng tư cho phép một công ty kiểm soát hoàn toàn việc sử dụng API.

- **Partner API:** Các API này được quảng bá, giới thiệu công khai nhưng chỉ được chia sẻ với các doanh nghiệp, đối tác kinh doanh đã ký thỏa thuận với nhà xuất bản. Trường hợp sử dụng phổ biến của các API này là tích hợp phần mềm giữa hai bên. Một công ty cấp cho các đối tác quyền truy cập vào dữ liệu hoặc năng lực được hưởng lợi từ các luồng doanh thu bổ sung. Đồng thời, nó có thể giám sát cách sử dụng các tài sản kỹ thuật số bị lộ, đảm bảo liệu các giải pháp của bên thứ ba sử dụng API của họ có cung cấp trải nghiệm người dùng tốt hay không và duy trì danh tính công ty trong ứng dụng của họ.
- **Public API:** Các nhà phát triển bên thứ ba có thể sử dụng các API dễ dàng. Mục đích của các API này nhằm nâng cao nhận thức, tăng giá trị thương hiệu cho chủ sở hữu, phát triển API đó.



Hình 2.4. Ứng dụng của API (Nguồn: itviec.com)

Ngoài cách phân loại trên, API còn được phân loại theo hệ thống mà chúng được thiết kế:

- **Database API:** Các API này cho phép giao tiếp giữa các ứng dụng và các hệ thống quản lý cơ sở dữ liệu. Các nhà phát triển làm việc với cơ sở dữ liệu bằng việc viết các truy vấn để truy cập dữ liệu, thay đổi bảng, v.v...

- Operating Systems API: Các API dùng để xác định cách mà các ứng dụng sử dụng dịch vụ và tài nguyên của hệ điều hành. Mọi hệ điều hành đều có bộ API của nó, chẳng hạn như API Windows hoặc API Linux.
- Remote API: Các API từ xa xác định các tiêu chuẩn tương tác cho các ứng dụng chạy trên các máy khác nhau. Nói cách khác, một sản phẩm phần mềm truy cập các tài nguyên nằm bên ngoài thiết bị yêu cầu chúng, điều này giải thích tên. Vì hai ứng dụng được định vị từ xa được kết nối qua mạng truyền thông, đặc biệt là internet, nên hầu hết các API từ xa đều được viết dựa trên các tiêu chuẩn web.
- Web API: Đây là nhóm API được sử dụng nhiều nhất. Các API này dựa trên web trả về (response) dữ liệu theo yêu cầu (request) từ phía máy khách hàng (client), nơi gọi API. Các API này cho phép lấy dữ liệu từ các nguồn bên ngoài. Ta có thể gửi cho API một yêu cầu chi tiết về thông tin chúng ta muốn lấy. API cũng cho phép các trang web của ta thay đổi dữ liệu trên các ứng dụng khác.

Dựa vào các lý thuyết trên, em sử dụng web API dưới dạng public API cho hệ thống của mình.

2.5. REST API

REST là từ viết tắt của REpresentational State Transfer. Đây là phong cách thiết kế cho các hệ thống siêu đa phương tiện phân tán và được Roy Fielding trình bày lần đầu tiên vào năm 2000 trong luận văn nổi tiếng của ông.

Cũng như các phong cách thiết kế khác, REST cũng có 6 nguyên tắc ràng buộc riêng mà một API phải tuân thủ theo để được gọi là RESTful. 6 nguyên tắc đó bao gồm:

- Client-server: Bằng cách tách các mối quan tâm về giao diện người dùng khỏi các mối quan tâm về lưu trữ dữ liệu, REST cải thiện tính di động của giao diện người dùng trên nhiều nền tảng và cải thiện khả năng mở rộng bằng cách đơn giản hóa các thành phần máy chủ.
- Stateless: Mỗi yêu cầu từ máy khách đến máy chủ phải chứa tất cả thông tin cần thiết để hiểu yêu cầu và không thể tận dụng bất kỳ ngữ cảnh được lưu trữ nào trên máy chủ. Do đó, trạng thái phiên được giữ hoàn toàn trên máy khách.
- Cacheable: Ràng buộc bộ nhớ cache yêu cầu dữ liệu trong một phản hồi cho một yêu cầu phải được gắn nhãn ngầm định hoặc rõ ràng là có thể lưu vào bộ nhớ cache hoặc không thể lưu vào bộ nhớ cache. Nếu một phản hồi có thể lưu vào bộ nhớ

cache, thì bộ đệm ẩn của máy khách được cấp quyền sử dụng lại dữ liệu phản hồi đó cho các yêu cầu tương đương sau này.

- Uniform interface: Bằng cách áp dụng nguyên tắc chung của kỹ thuật phần mềm cho giao diện thành phần, kiến trúc hệ thống tổng thể được đơn giản hóa và khả năng hiển thị của các tương tác được cải thiện. Để có được một giao diện thống nhất, cần có nhiều ràng buộc kiến trúc để hướng dẫn hành vi của các thành phần. REST được định nghĩa bởi bốn ràng buộc giao diện: xác định tài nguyên; thao tác các nguồn lực thông qua các đại diện; thông điệp tự mô tả; và, siêu phương tiện như động cơ của trạng thái ứng dụng.
- Layered System: Kiểu hệ thống phân lớp cho phép một kiến trúc bao gồm các lớp phân cấp bằng cách hạn chế hành vi của thành phần sao cho mỗi thành phần không thể “nhìn thấy” ngoài lớp trực tiếp mà chúng đang tương tác.
- Code on demand (Không bắt buộc): REST cho phép mở rộng chức năng của ứng dụng khách bằng cách tải xuống và thực thi mã dưới dạng các applet hoặc script. Điều này đơn giản hóa khách hàng bằng cách giảm số lượng các tính năng bắt buộc phải triển khai trước.

Đó là những lý thuyết em tham khảo được từ bài luận văn của Roy Fielding [9]. Em sẽ áp dụng các tính chất sau vào hệ thống của mình bao gồm:

- Self-documenting, API được thiết kế khi nhìn vào có thể đoán được, hiểu được API đó sẽ làm những gì.
- Tính mở rộng và tùy biến cho các API.
- API method sẽ gắn liền với ý nghĩa, chức năng của API đó. VD:

| | | |
|--------|-------------------------------|---|
| GET | GET Get Cart | Dùng để lấy thông tin, truy xuất dữ liệu. |
| POST | POST Add Item to Cart | Thêm, tạo mới dữ liệu. |
| DELETE | DEL Del Item form Cart | Xóa dữ liệu. |
| PUT | PUT Edit Customer | Cập nhật, thay đổi dữ liệu. |

- API sẽ phản hồi lại thông báo và status code rõ ràng sau khi gọi API. Các status code thông dụng thường được sử dụng và cũng là các status em sẽ dùng cho các API của hệ thống:
 - 200 OK - Đây là status code thường được sử dụng nhất để cho biết rằng hoạt động của API đã được thực hiện thành công.
 - 201 CREATED – Status code này có thể được sử dụng khi phương thức POST tạo mới thành công một tài nguyên.
 - 202 ĐƯỢC CHẤP NHẬN – Status code này có thể được sử dụng để xác nhận yêu cầu được gửi đến máy chủ.
 - 400 YÊU CẦU XẤU – Status code dùng để báo xác thực đầu vào phía máy khách không thành công.
 - 401 UNAUTHORIZED / 403 FORBIDDEN – Status code này có thể được sử dụng nếu người dùng hoặc hệ thống không được phép (ủy quyền) thực hiện một số hoạt động nhất định.
 - 404 NOT FOUND – Status code này có thể được sử dụng nếu bạn đang tìm kiếm một số tài nguyên nhất định và nó không có sẵn trong hệ thống.
 - 500 LỖI MÁY CHỦ NỘI BỘ - Status code này không bao giờ được đưa ra một cách rõ ràng nhưng có thể xảy ra nếu hệ thống bị lỗi.
 - 502 BAD GATEWAY – Status code này có thể được sử dụng nếu máy chủ nhận được phản hồi không hợp lệ từ máy chủ ngược dòng.

2.6. OAuth 2

OAuth 2.0 là giao thức tiêu chuẩn dùng để ủy quyền (xác thực). Nó là một phương thức, giao thức giúp xác thực giữa các ứng dụng để chia sẻ dữ liệu cho nhau mà không cần cung cấp thông tin cá nhân (username, password).

OAuth xác định 3 vai trò:

- Resource Owner (User): Là người dùng cho phép ứng dụng truy cập vào tài khoản của họ. Quyền truy cập của ứng dụng vào tài khoản của người dùng bị giới hạn trong "phạm vi" của ủy quyền được cấp (ví dụ: quyền đọc hoặc ghi).
- Resource / Authorization Server (API): Máy chủ tài nguyên lưu trữ các tài khoản người dùng được bảo vệ và máy chủ ủy quyền xác minh danh tính của người dùng

sau đó cấp mã thông báo truy cập cho ứng dụng. Theo quan điểm của nhà phát triển ứng dụng, API của dịch vụ hoàn thành cả vai trò tài nguyên và máy chủ ủy quyền.

- Client (Application): Là ứng dụng muốn truy cập vào tài khoản của người dùng. Trước khi có thể làm như vậy, nó phải được người dùng ủy quyền và ủy quyền phải được xác thực bởi API.



Hình 2.5. Sơ đồ minh họa OAuth (Nguồn: digitalocean.com)

Giải thích chi tiết cho luồng hoạt động OAuth trên:

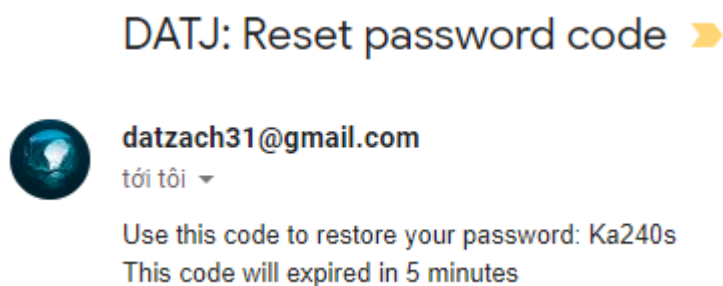
- 1: Ứng dụng yêu cầu ủy quyền để truy cập tài nguyên dịch vụ từ người dùng.
- 2: Nếu người dùng cho phép yêu cầu, ứng dụng sẽ nhận được một khoản cấp phép.
- 3: Ứng dụng yêu cầu mã thông báo truy cập từ máy chủ ủy quyền (API) bằng cách trình bày xác thực danh tính của chính nó và cấp ủy quyền.
- 4: Nếu danh tính ứng dụng được xác thực và việc cấp phép hợp lệ, máy chủ ủy quyền (API) sẽ cấp mã thông báo truy cập cho ứng dụng. Việc ủy quyền đã hoàn tất.
- 5: Ứng dụng yêu cầu tài nguyên từ máy chủ tài nguyên (API) và xuất trình mã thông báo truy cập để xác thực.
- 6: Nếu mã thông báo truy cập hợp lệ, máy chủ tài nguyên (API) sẽ cung cấp tài nguyên cho ứng dụng.

Đây chỉ là ý tưởng chung cho luồng hoạt động, xử lý của OAuth, luồng hoạt động thực tế của quá trình này sẽ khác. Nó phụ thuộc vào loại cấp phép đang sử dụng và cơ chế hoạt động, quản lý của hệ thống.

Trong đề tài này, em dùng token để xác thực cho các API của hệ thống. Khi người dùng đăng nhập vào, hệ thống sẽ tạo ra một token liên kết với người dùng đó cùng với key (khóa) là đoạn mã đã được băm từ chuỗi tên đăng nhập của người dùng và ngày giờ họ đăng nhập. Sau khi tạo xong, token đó sẽ được lưu vào cơ sở dữ liệu và sẽ được dùng để xác thực cho các hành động thực thi của API gửi yêu cầu lên hệ thống. Sau khi tạo và lưu token hoàn tất, hệ thống sẽ trả về key của token đó, người dùng (client) có thể sử dụng token để thực hiện các giao dịch có tính cá nhân và cần được bảo mật như tạo đơn hàng, sửa thông tin tài khoản, v.v... Token sẽ hết hạn sử dụng sau 1 tiếng kể từ khi token được tạo ra khi người dùng đăng nhập vào hệ thống.

2.7. OTP and Mail

Phương thức Email OTP cho phép bạn xác thực bằng mật khẩu dùng một lần (OTP) được gửi đến địa chỉ email đã đăng ký. Khi bạn cố gắng xác thực trên bất kỳ thiết bị nào, máy chủ sẽ gửi một email đến địa chỉ email đã đăng ký với OTP. Bạn có thể sử dụng OTP này để xác thực một lần trong một khoảng thời gian ngắn.



Hình 2.6. Ví dụ mail OTP code

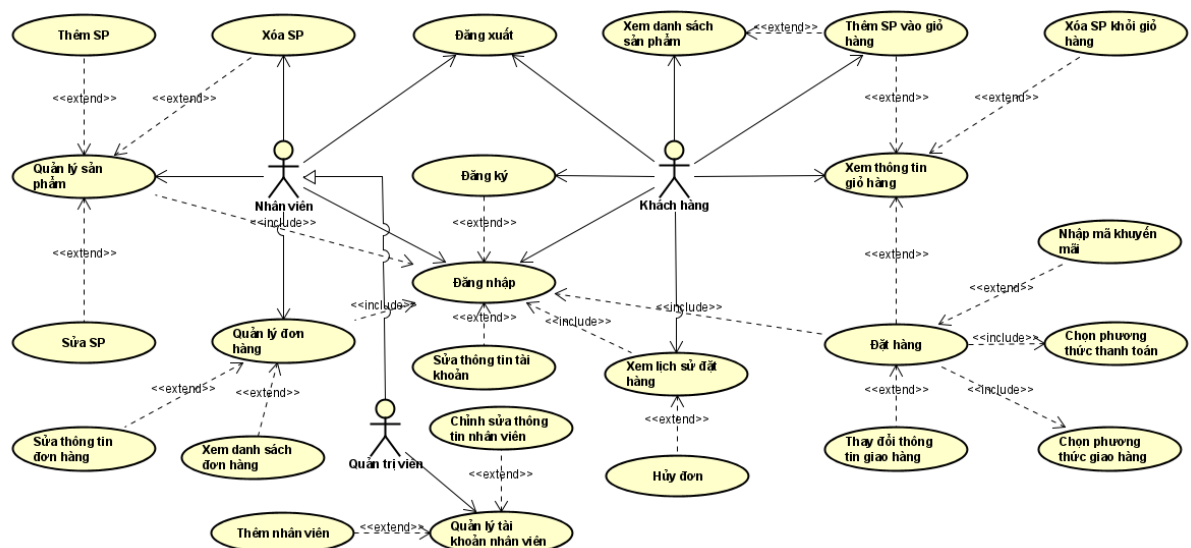
Để hỗ trợ cho việc người dùng bị quên mật khẩu, em sử dụng lý thuyết trên để thực hiện việc lấy mật khẩu cho người dùng. Với cách thực hiện là khi người dùng yêu cầu khôi phục mật khẩu hệ thống sẽ tạo 1 mã OTP (Sau khi đã xác thực email hợp lệ) liên kết với tài khoản vừa yêu cầu và gửi đến email mà người dùng đã cung cấp. Người dùng sẽ dùng mã này để tạo lại mật khẩu mới. Hệ thống cũng sẽ gửi mail thông báo về các thông tin tình trạng đơn hàng cho khách hàng.

Chương 3. XÂY DỰNG API BÁN HÀNG TRỰC TUYẾN

Trong phần báo cáo này gồm quá trình em thực hiện triển khai đề tài nghiên cứu của mình qua từng giai đoạn. Bước đầu em xác định các chức năng cần thiết của hệ thống cho các đối tượng sử dụng và thiết kế cơ sở dữ liệu để đáp ứng được các chức năng đó. Sau đó xây dựng các API dựa trên Django Rest Framework.

3.1. Xác định yêu cầu hệ thống

Hệ thống website bán hàng online gồm 2 đối tượng sử dụng chính là khách hàng và nhân viên. Từ đó xây dựng các chức năng chính và cơ bản để phục vụ cho hệ thống. Dưới đây là sơ đồ usecase của hệ thống:



Hình 3.1. Sơ đồ usecase

Theo sơ đồ trên, đối tượng quản lý của hệ thống gồm có quản trị viên và nhân viên. Quản trị viên là người có toàn quyền để quản lý các chức năng điều hành, quản lý. Nhân viên được chia ra nhiều loại nhân viên để quản lý từng thành phần của hệ thống như sau:



Hình 3.2. Phân quyền nhân viên

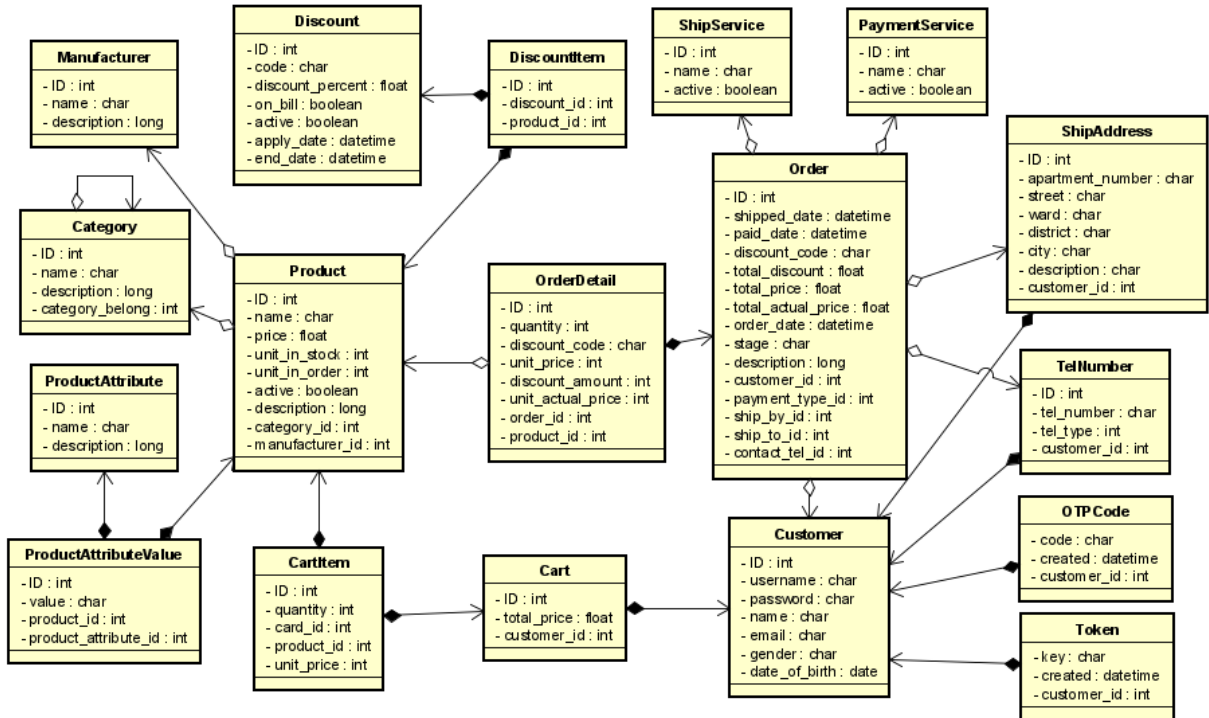
- Nhân viên dịch vụ: Quản lý các dịch vụ bên thứ 3 cung cấp cho hệ thống như các dịch vụ thanh toán, các dịch vụ giao hàng.
- Nhân viên kho: Quản lý các sản phẩm, nhà cung cấp, thương hiệu, loại sản phẩm, số lượng sản phẩm tồn kho và đang được đặt hàng.
- Nhân viên nhân sự: Quản lý nhân viên và phân quyền cho các nhân viên.
- Nhân viên thống kê đơn: Quản lý các đơn hàng, theo dõi đơn hàng, cập nhật trạng thái của các đơn hàng.

Khách hàng là đối tượng sử dụng hệ thống gồm các thao tác chính sau:

- Đăng ký, đăng nhập, đăng xuất, chỉnh sửa thông tin tài khoản.
- Xem danh sách các sản phẩm.
- Thêm sản phẩm vào giỏ hàng hoặc xóa sản phẩm khỏi giỏ hàng.
- Tiến hành đặt hàng tạo đơn hàng, trong lúc tạo đơn hàng phải thực hiện các công việc bao gồm chọn phương thức thanh toán và cung cấp thông tin giao hàng. Có thể nhập mã khuyến mãi nếu có.
- Xem lịch sử các đơn hàng đã đặt, theo dõi tình trạng các đơn hàng và có thể hủy đơn đối với các đơn hàng chưa thanh toán và còn trong tình trạng xử lý (PROCESSING).

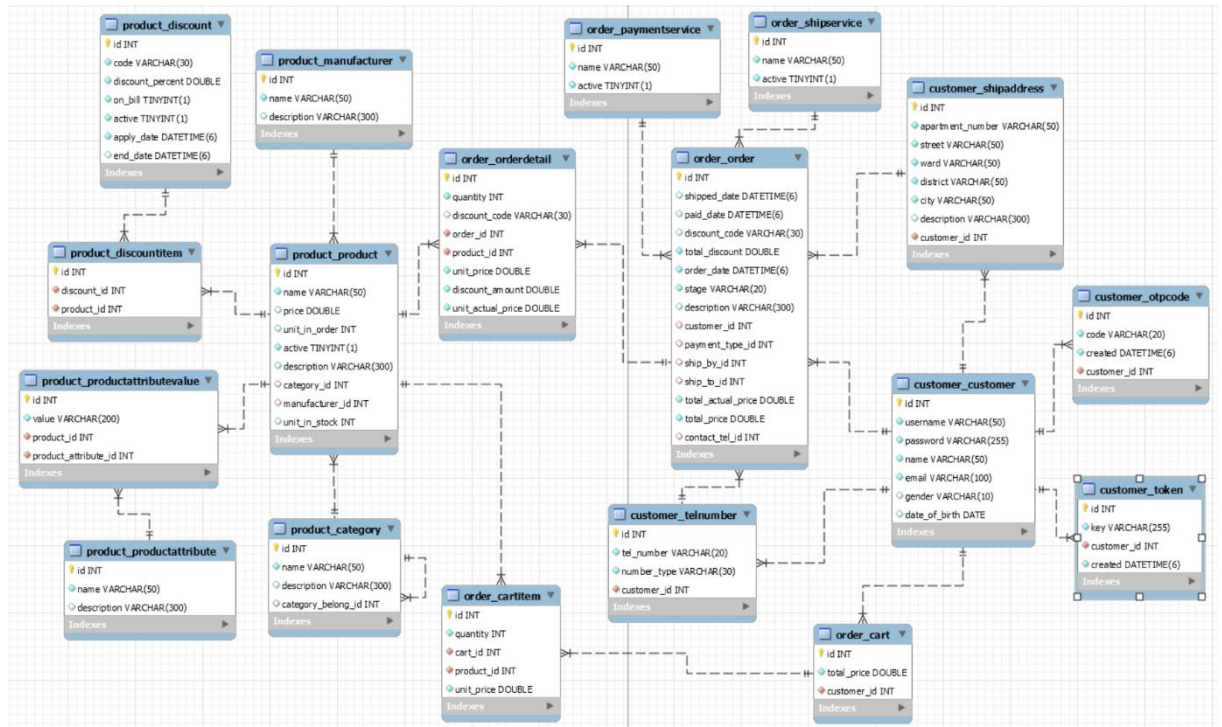
3.2. Thiết kế cơ sở dữ liệu

Từ các yêu cầu đã được xác định ở phần trên, em thiết kế được sơ đồ class như sau:



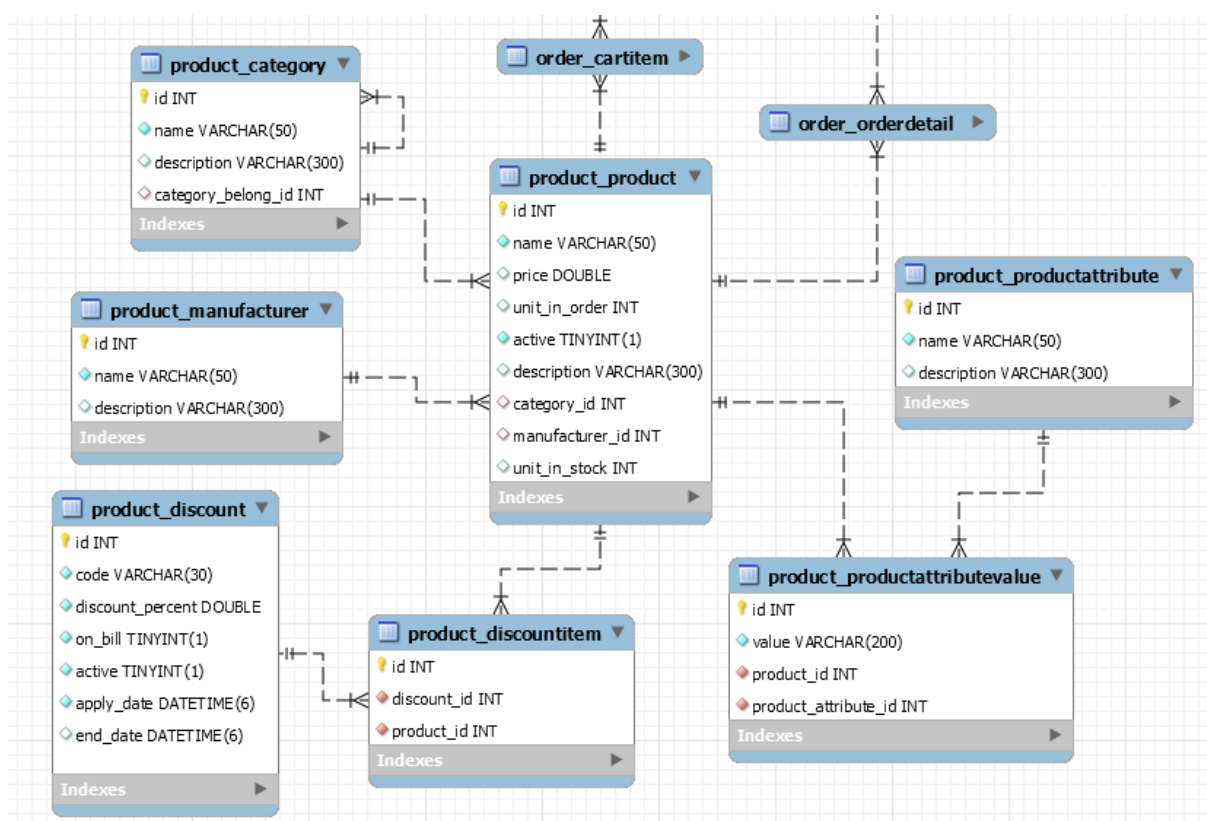
Hình 3.3. Sơ đồ class

Từ sơ đồ class trên em có được sơ đồ thiết kế dữ liệu sau:



Hình 3.4 Sơ đồ dữ liệu tổng quát (EER).

Em phân dữ liệu trên thành 3 nhóm dữ liệu chính gồm: Product, Order, Customer.



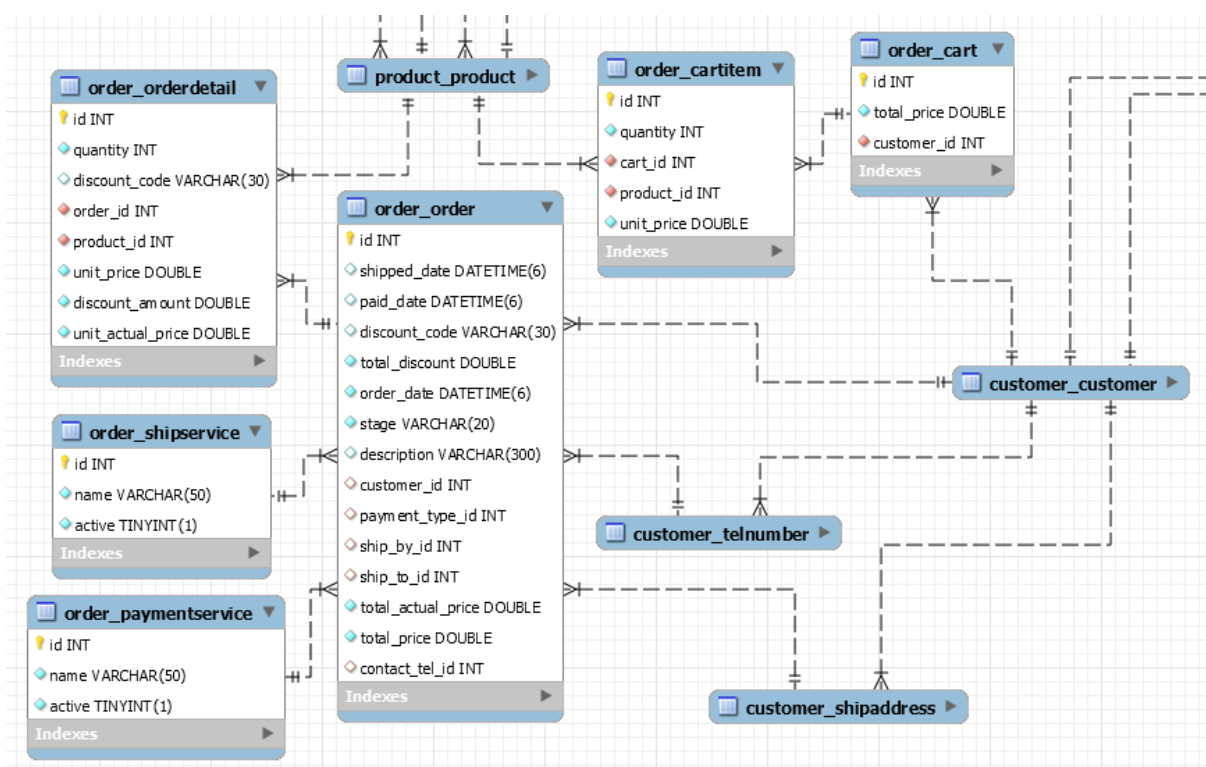
Hình 3.5. Cơ sở dữ liệu Product

Bảng Category và Manufacturer được dùng để phân loại cho sản phẩm, và bảng Category cũng được dùng để phân loại và gom nhóm chính nó (VD: danh mục linh kiện gồm nhiều danh mục khác như ram, ổ cứng, v.v...).

Discount được dùng để xác định khuyến mãi được áp dụng trên các sản phẩm nào hoặc không áp dụng cho sản phẩm mà áp dụng cho tổng bill bằng thuộc tính on_bill (Nếu áp dụng on_bill sẽ không áp dụng cho sản phẩm và ngược lại).

Bảng ProductAttribute dùng để định nghĩa các thuộc tính của 1 sản phẩm. ProductAttributeValue dùng để xác định giá trị của thuộc tính và sản phẩm. Cách thiết kế này được dựa theo mô hình Entity-Attribute-Value (EAV) [1] giúp cho dữ liệu sản phẩm có thể dễ dàng thêm mới các thuộc tính về sau.

Bảng Product được dùng làm khóa ngoại cho các bảng OrderDetail và CartItem của dữ liệu Order trong phần tiếp theo.

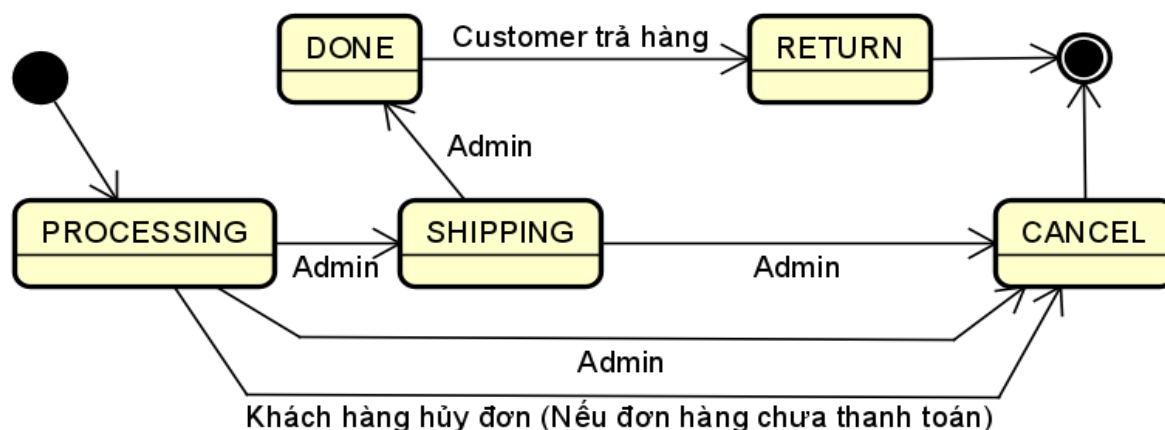


Hình 3.6. Cơ sở dữ liệu Order

Bảng ShipService và PaymentService được dùng để lưu trữ các thông tin dịch vụ cung cấp bởi bên thứ 3 cho hệ thống.

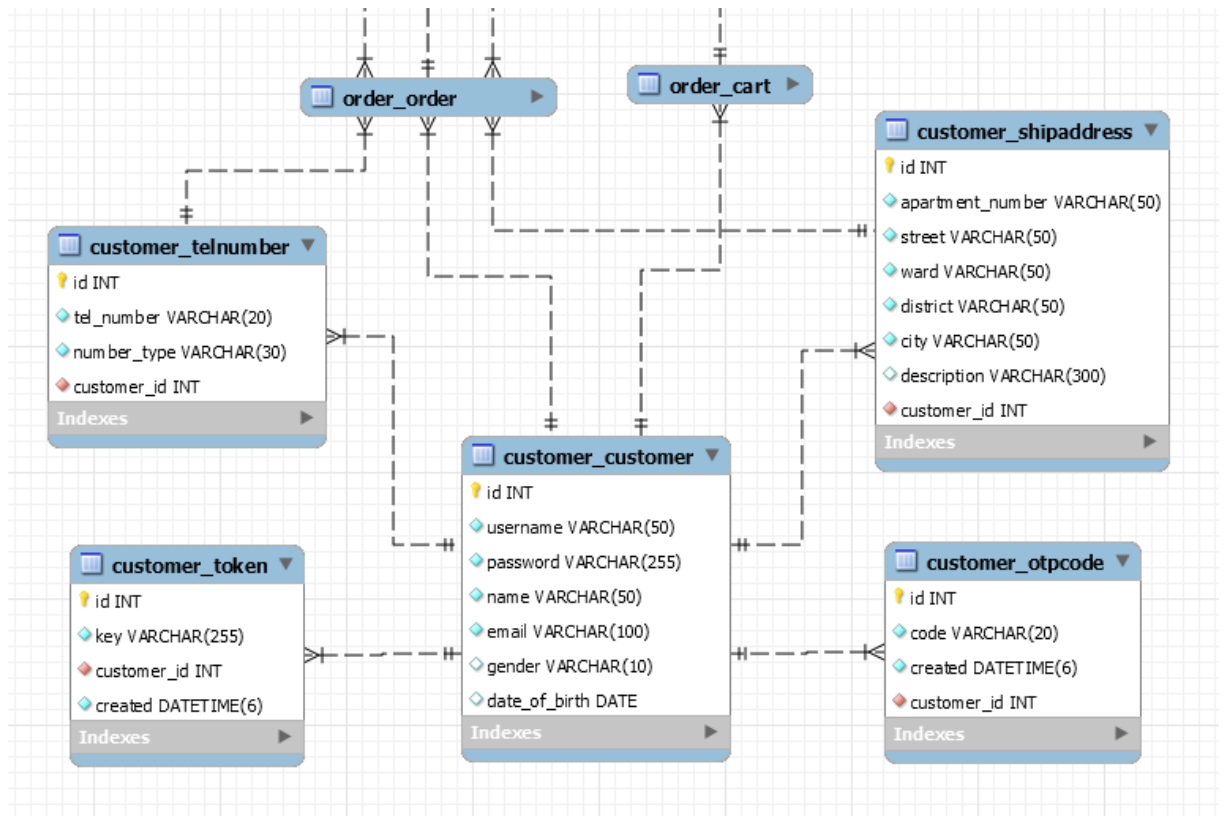
Bảng Cart được dùng để lưu các sản phẩm đã được thêm vào giỏ và hiện có trong giỏ hàng của khách hàng. Có khóa ngoại tham chiếu đến bảng Customer, xác định giỏ hàng đó của khách hàng nào.

Bảng Order dùng để lưu trữ thông tin đơn đặt hàng của khách hàng, các thông tin giao hàng (ship_to, contact_tel, customer_id) là khóa ngoại tham chiếu đến các bảng TelNumber, ShipAddress, Customer của dữ liệu Customer. Thông Stage của bảng Order gồm 5 trạng thái:



Hình 3.7. Các trạng thái của đơn hàng

Đơn hàng được khởi tạo bởi khách hàng và có trạng thái là PROCESSING. Nhân viên hệ thống sẽ theo dõi, cập nhật các trạng thái cho đơn hàng. Sau khi khách hàng nhận được hàng, đơn hàng sẽ có trạng thái DONE, lúc này khách hàng có thể trả lại hàng nếu không ưng ý. Khách hàng cũng có thể hủy đơn đặt hàng chỉ trong trường hợp đơn hàng đó chưa được thanh toán.



Hình 3.8. Cơ sở dữ liệu Customer

Bảng TelNumber và ShipAddress được dùng để giúp khách hàng lưu trữ nhiều thông tin để đặt hàng.

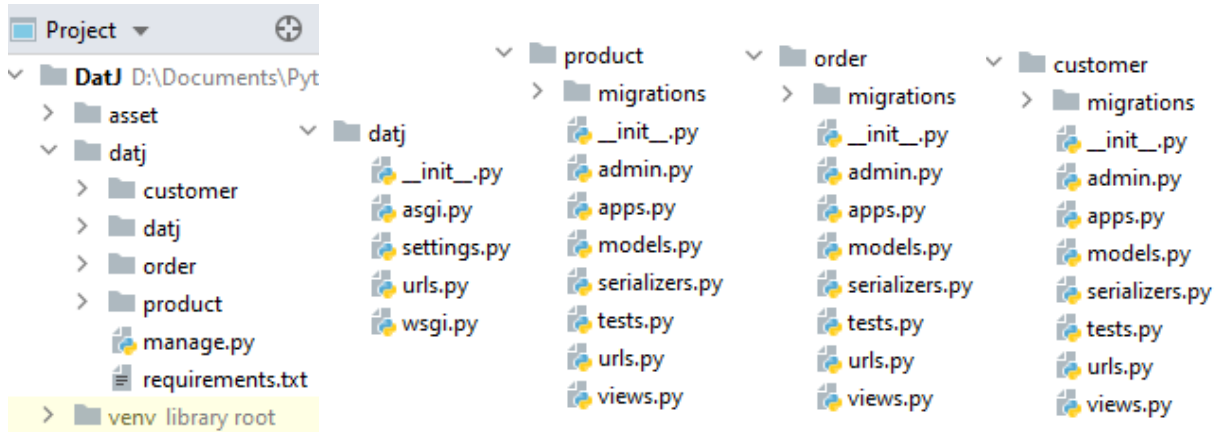
Bảng OTPCode được dùng để lưu mã xác nhận 1 lần, phục vụ cho việc khôi phục tài khoản nếu khách hàng bị quên mật khẩu, và thẻ dùng để xác thực cho các chức năng cần chứng thực.

Token được dùng để lưu trữ token xác thực để thực hiện các giao dịch, thêm sản phẩm vào giỏ, đặt hàng, thay đổi thông tin tài khoản.

3.3. Xây dựng trang quản trị (Admin)

3.3.1. Giai đoạn 1: Tạo cơ sở dữ liệu

Khởi tạo dự án với cấu trúc:



Hình 3.9. Cấu trúc dự án

Sử dụng phương pháp code-first [2] để tạo dữ liệu cho dự án. Đầu tiên tạo class (Là 1 bảng dữ liệu) cùng các thuộc tính đã được thiết kế ở các phần trên trong models

```
class Discount(models.Model):
    code = models.CharField(max_length=30, unique=True)
    discount_percent = models.FloatField(default=0)
    on_bill = models.BooleanField(default=False)
    active = models.BooleanField(default=True)
    apply_date = models.DateTimeField(default=datetime.now())
    end_date = models.DateTimeField(blank=True, null=True)

    def __str__(self):
        return self.code
```

Sau khi đã tạo các class theo thiết kế trước đó, nhờ vào framework Django em chạy các lệnh sau để tạo cơ sở dữ liệu vào hệ quản trị cơ sở dữ liệu MySQL

```
py manage.py makemigrations
py manage.py migrate
```

3.3.2. Giai đoạn 2: Tạo trang Admin

Tiếp đến, em sử dụng thư viện admin [3] của Django để tạo web với các chức năng cơ bản cho nhân viên.

Tại file admin em tạo 1 class ModelAdmin như sau:

```
class ProductAdmin(admin.ModelAdmin):
    inlines = [ProductAttributeValueInline]
    list_per_page = 12
    search_fields = ('name',)
    list_filter = ('category', 'manufacturer', 'active',)
```

Sau đó đăng ký model để chạy lên trang admin:

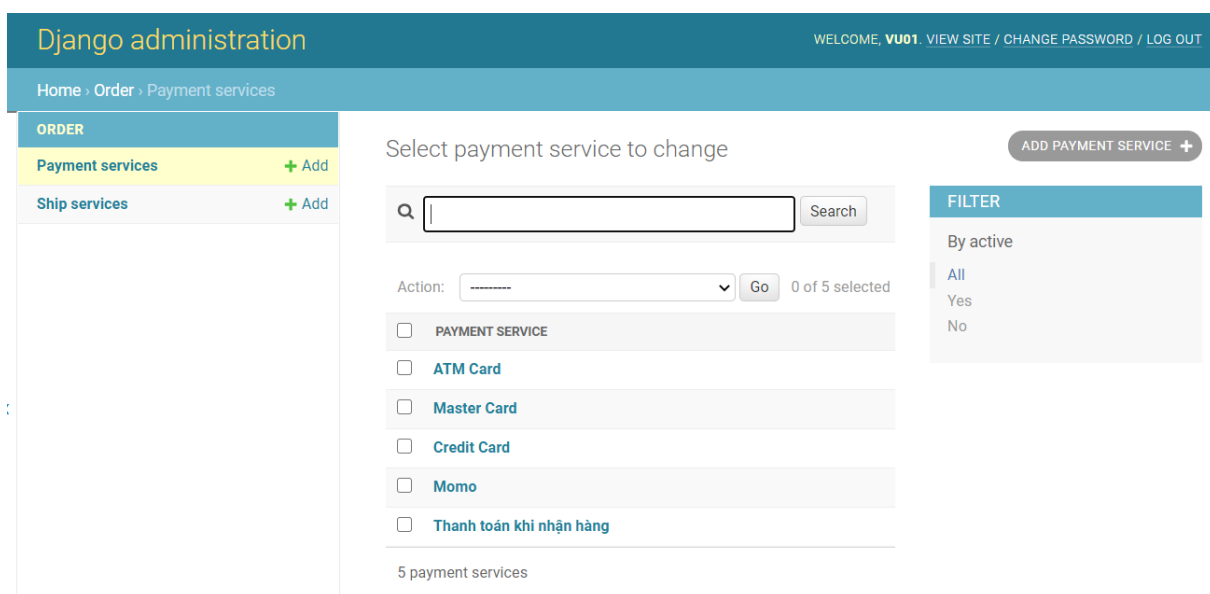
```
admin.site.register(Product, ProductAdmin)
```

Thao tác tương tự với các model khác và chỉnh sửa ModelAdmin theo yêu cầu để phù hợp với mô tả trước đó.

Sau khi hoàn tất, em dùng lệnh `py manage.py runserver` để chạy project và có được trang admin cơ bản như sau:

3.3.2.1. Quản lý các dịch vụ bên thứ ba:

Xem danh sách các dịch vụ



Thêm mới dịch vụ

Django administration

WELCOME, **VU01**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Order > Payment services > Add payment service

ORDER

Payment services [+ Add](#)

Ship services [+ Add](#)

Add payment service

Name:

Zalo Pay

☒ Active

Save and add another

Save and continue editing

SAVE

Chỉnh sửa thông tin hoặc xóa dịch vụ

Django administration

WELCOME, **VU01**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Order > Payment services > Momo

ORDER

Payment services [+ Add](#)

Ship services [+ Add](#)

Change payment service

HISTORY

Name:

Momo

☒ Active

Delete

Save and add another

Save and continue editing

SAVE

3.3.2.2. Quản lý kiểm kê đơn hàng:

Xem danh sách đơn hàng

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Order > Orders

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

ORDER

Order details

Orders

Payment services [+ Add](#)

Ship services [+ Add](#)

PRODUCT

Categories [+ Add](#)

Discount items [+ Add](#)

Discounts [+ Add](#)

Manufacturers [+ Add](#)

Product attribute values [+ Add](#)

Product attributes [+ Add](#)

Select order to view

| ORDER | STAGE | PAID DATE |
|---------------------------------------|------------|--------------------------|
| Order-61-nbdat22-11/04/2020, 09:29:29 | Processing | Nov. 4, 2020, 4:29 p.m. |
| Order-60-nbdat22-11/04/2020, 07:09:32 | Cancel | - |
| Order-59-nbdat22-11/02/2020, 10:47:53 | Cancel | - |
| Order-58-nbdat22-11/02/2020, 09:51:46 | Return | - |
| Order-57-nbdat22-11/02/2020, 09:51:27 | Done | Nov. 4, 2020, 5:23 p.m. |
| Order-56-nbdat22-11/02/2020, 09:45:35 | Done | Nov. 2, 2020, 5:24 p.m. |
| Order-55-nbdat22-11/02/2020, 09:45:07 | Done | Nov. 2, 2020, 5:24 p.m. |
| Order-54-nbdat22-11/02/2020, 09:42:11 | Cancel | - |
| Order-53-nbdat22-11/02/2020, 09:38:50 | Cancel | - |
| Order-52-nbdat22-11/02/2020, 09:37:04 | Return | - |
| Order-51-nbdat22-10/30/2020, 11:07:38 | Cancel | - |
| Order-50-nbdat22-10/30/2020, 11:05:09 | Done | Oct. 30, 2020, 6:13 p.m. |

1

2

19 orders

Show all

FILTER

By stage

All

Processing

Shipping

Done

Cancel

Return

By order date

Any date

Today

Past 7 days

This month

This year

By shipped date

Any date

Today

Past 7 days

3.3.2.3. Quản lý sản phẩm và thông tin khuyến mãi:

Xem danh sách sản phẩm (Tương tự khuyến mãi và các bảng khác)

Django administration

WELCOME, KH001. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Product > Products

PRODUCT

Categorys + Add

Discount items + Add

Discounts + Add

Manufacturers + Add

Product attribute values + Add

Product attributes + Add

Products + Add

Select product to change

Q

Search

Action: ----- Go 0 of 12 selected

☐ PRODUCT

☐ Chuột máy tính Asus TUF Gaming M3

☐ Chuột gaming Logitech G403 Hero

☐ Chuột máy tính không dây Logitech Mx Anywhere 2S

☐ Chuột máy tính không dây Logitech M337

☐ Chuột gaming không dây CORSAIR Corsair Dark Core R

☐ Chuột máy tính Corsair Harpoon PRO RGB

☐ Chuột gaming CorSAIR Ironclaw

☐ Bàn phím Logitech K120

☐ Bàn phím cơ Gaming Logitech G Pro X

FILTER

By category

All

Laptop

Màn hình

Bàn phím

Chuột

Thiết bị ngoại vi

-

By manufacturer

All

Asus

Dell

HP

CORSAIR

LOGITECH

ADD PRODUCT +

Thêm sản phẩm mới (Tương tự khuyến mãi và các bảng khác)

Add product

Category: Laptop

Manufacturer: CORSAIR

Name:

Price: 0

Unit in stock: 0

Unit in order: 0

☒ Active

Description:

PRODUCT ATTRIBUTE VALUES

| PRODUCT ATTRIBUTE | VALUE | DELETE? |
|-------------------|-------|--------------------------|
| ----- | | <input type="checkbox"/> |
| ----- | | <input type="checkbox"/> |

+ Add another Product attribute value

Save and add another




Save and continue editing

SAVE

Xóa hoặc chỉnh sửa thông tin sản phẩm (Tương tự khuyến mãi và các bảng khác)

Change product

[HISTORY](#)

Category: Laptop   

Manufacturer: Dell   

Name: Dell G3 Inspiron 3590 N5I5517W

Price: 22990000.0






















Unit in stock: 29

Unit in order: 0

☒ Active

Description:

PRODUCT ATTRIBUTE VALUES

| PRODUCT ATTRIBUTE | VALUE | DELETE? |
|--|--|-------------------------------------|
| Intel Core i5-9300H 2.4GHz up to 4.1GHz 8MB | | |
| CPU    | Intel Core i5-9300H 2.4GHz up to 4.1GHz 8MB | <input type="checkbox"/> |
| 2 x 4GB DDR4 2666MHz | | |
| Ram    | 2 x 4GB DDR4 2666MHz | <input type="checkbox"/> |
| 256GB SSD M.2 PCIE | | |
| Ổ cứng    | 256GB SSD M.2 PCIE | <input type="checkbox"/> |
| 15.6" FHD (1920 x 1080) IPS, Anti-Glarec | | |
| Thông số màn hình    | 15.6" FHD (1920 x 1080) IPS, Anti-Glarec | <input type="checkbox"/> |
| 3 Cell 56Whr | | |
| Pin    | 3 Cell 56Whr | <input type="checkbox"/> |
| https://product.hstatic.net/1000026716/product/ll-g3-inspiron-3590-n5i5517w_1_0c916cb24b5e4f328e249eb350768f22_master_3aa6f65982af408b913ab1f85f0f3b40.jpg | | |
| Ảnh    | https://product.hstatic.net/1000026716/prod | <input type="checkbox"/> |
| -----    | | <input checked="" type="checkbox"/> |

[+ Add another Product attribute value](#)

[Delete](#)[Save and add another](#)[Save and continue editing](#)[SAVE](#)

3.3.2.4. Quản lý nhân sự:

Xem danh sách nhân viên

Django administration

WELCOME, **su01** [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) > [Authentication and Authorization](#) > [Users](#)

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Users

+ Add

Select user to change

Q

Search

Action:

Go

 0 of 5 selected

| <input type="checkbox"/> | USERNAME | EMAIL ADDRESS | FIRST NAME | LAST NAME | STAFF STATUS |
|--------------------------|----------|---------------------|------------|-----------|--------------|
| <input type="checkbox"/> | admin | datzach31@gmail.com | | | ✓ |
| <input type="checkbox"/> | don01 | | | | ✓ |
| <input type="checkbox"/> | kho01 | | | | ✓ |
| <input type="checkbox"/> | su01 | | | | ✓ |
| <input type="checkbox"/> | vu01 | | | | ✓ |

5 users

ADD USER

+

FILTER

By staff status

All

Yes

No

By superuser status

All

Yes

No

By active

All

Yes

No

Thêm nhân viên mới

Add user

First, enter a username and password. Then, you'll be able to edit more user options.

Username:

nhanviemmoi

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:

•

Password confirmation:

•

Enter the same password as before, for verification.

Save and add another

Save and continue editing

SAVE

Thêm, chỉnh sửa các thông tin của nhân viên

Change user

[HISTORY](#)

| | |
|---|---|
| Username: | <input type="text" value="nhanviemmoi"/> |
| Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. | |
| Password: | algorithm: pbkdf2_sha256 iterations: 216000 salt: 8bUri7***** hash: MykXUO***** |
| Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form . | |

Personal info

| | |
|----------------|--|
| First name: | <input type="text" value="Văn A"/> |
| Last name: | <input type="text" value="Nguyễn"/> |
| Email address: | <input type="text" value="test123@gmail.com"/> |

Permissions

- ☒ Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
- ☒ Staff status
Designates whether the user can log into this admin site.

Phân quyền cho nhân viên vừa thêm

Groups:

| | |
|--|--|
| <div>Available groups ?</div> <div><input type="text" value="Filter"/></div> <div>Nhân viên dịch vụ Nhân viên kho Nhân viên nhân sự Nhân viên thống kê đơn</div> | <div>Chosen groups ?</div> <div></div> |
| Choose all | Remove all |

The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

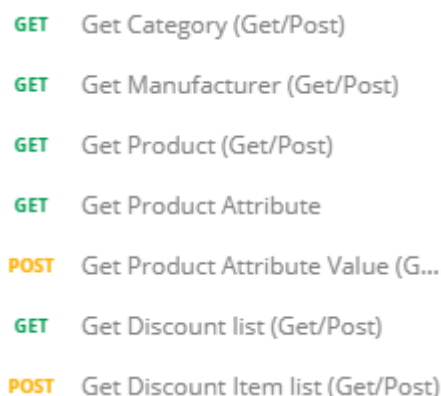
User permissions:

| | |
|---|--|
| <div>Available user permissions ?</div> <div><input type="text" value="Filter"/></div> <div>admin log entry Can add log entry admin log entry Can change log entry admin log entry Can delete log entry admin log entry Can view log entry auth group Can add group auth group Can change group auth group Can delete group</div> | <div>Chosen user permissions ?</div> <div></div> |
|---|--|

3.4. Xây dựng các API theo yêu cầu hệ thống

Dựa theo sơ đồ usecase và class ở trên, em phân làm 3 nhóm API chính là:

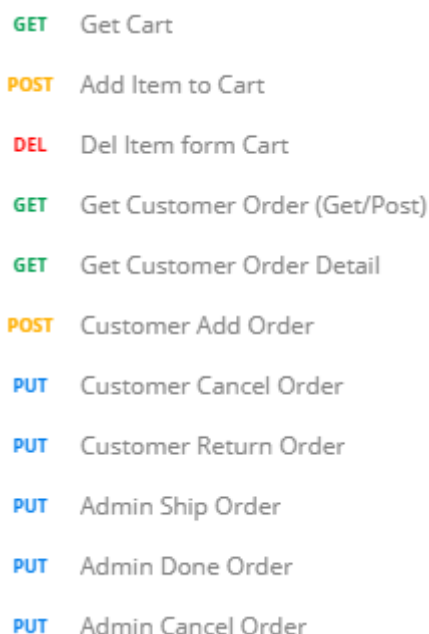
- API Product: Các API dùng để tải dữ liệu, thông tin về sản phẩm và các thông tin liên quan, xoay quanh đến sản phẩm đó (Như là Category, Manufacturer, Discount). Các API của Product sẽ bao gồm như sau:

A screenshot of a list of API endpoints for the Product module. The endpoints are listed with their HTTP methods in colored text: GET (green), POST (orange), and GET (green). The endpoints are: Get Category (Get/Post), Get Manufacturer (Get/Post), Get Product (Get/Post), Get Product Attribute, Get Product Attribute Value (G...), Get Discount list (Get/Post), and Get Discount Item list (Get/Post).

GET Get Category (Get/Post)
GET Get Manufacturer (Get/Post)
GET Get Product (Get/Post)
GET Get Product Attribute
POST Get Product Attribute Value (G...
GET Get Discount list (Get/Post)
POST Get Discount Item list (Get/Post)

Hình 3.10. Danh sách API Product

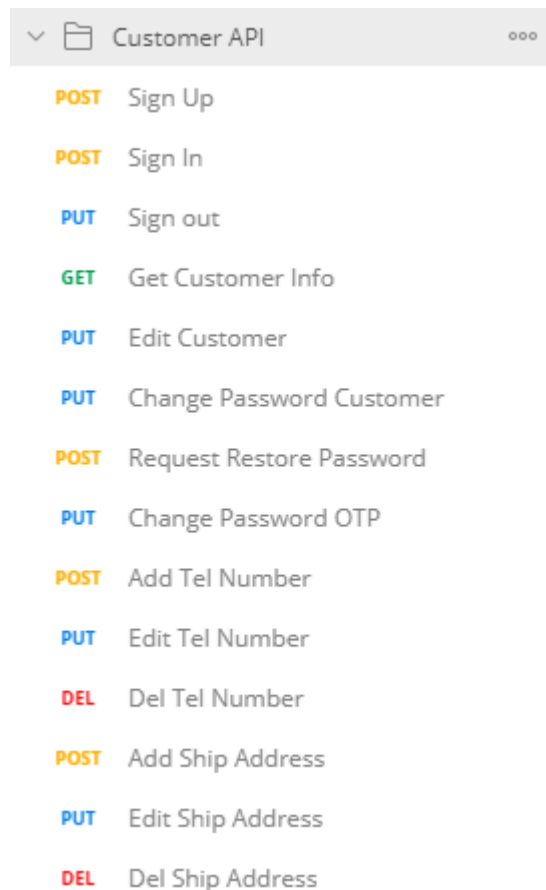
- API Order: Các API này được dùng để phục vụ khách hàng trong việc thêm, xóa sản phẩm từ giỏ hàng, đặt hàng, xem lịch sử thông tin đơn hàng, hủy đơn và trả hàng. Các API này sẽ được chứng thực bằng token [5]. Ngoài ra còn có các API dành cho nhân viên để thực hiện các thao tác cập nhật trạng thái của các đơn hàng. Các API này cũng sẽ được chứng thực bằng token [5] của nhân viên. Các API của Order sẽ bao gồm như sau:

A screenshot of a list of API endpoints for the Order module. The endpoints are listed with their HTTP methods in colored text: GET (green), POST (orange), DEL (red), GET (green), GET (green), POST (orange), PUT (blue), PUT (blue), PUT (blue), PUT (blue), and PUT (blue). The endpoints are: Get Cart, Add Item to Cart, Del Item form Cart, Get Customer Order (Get/Post), Get Customer Order Detail, Customer Add Order, Customer Cancel Order, Customer Return Order, Admin Ship Order, Admin Done Order, and Admin Cancel Order.

GET Get Cart
POST Add Item to Cart
DEL Del Item form Cart
GET Get Customer Order (Get/Post)
GET Get Customer Order Detail
POST Customer Add Order
PUT Customer Cancel Order
PUT Customer Return Order
PUT Admin Ship Order
PUT Admin Done Order
PUT Admin Cancel Order

Hình 3.11. Danh sách API Order

- API Customer: Các API này giúp khách hàng tương tác với hệ thống, giúp khách hàng đăng ký, đăng nhập, đăng xuất và sửa thông tin tài khoản, thêm mới hoặc sửa xóa các thông tin đặt hàng. Các API này sẽ được chứng thực bằng token [5]. API sẽ dùng thư viện mail [6] của Django để gửi mã xác thực 1 lần về mail của khách hàng khi có nhu cầu lấy lại mật khẩu. Các API của Customer sẽ bao gồm như sau:



| Method | Endpoint |
|--------|--------------------------|
| POST | Sign Up |
| POST | Sign In |
| PUT | Sign out |
| GET | Get Customer Info |
| PUT | Edit Customer |
| PUT | Change Password Customer |
| POST | Request Restore Password |
| PUT | Change Password OTP |
| POST | Add Tel Number |
| PUT | Edit Tel Number |
| DEL | Del Tel Number |
| POST | Add Ship Address |
| PUT | Edit Ship Address |
| DEL | Del Ship Address |

Hình 3.12. Danh sách API Customer

Em dùng thư viện `rest_framework` [4] của Django để xây dựng, viết các API và các API này sẽ được viết theo chuẩn REST [7].

API được viết gồm 3 công đoạn:

- **Serializer:**

Tạo các serializer để đọc dữ liệu từ request khi gửi API lên server và trả dữ liệu về khi server response các API đó.

Code để tạo 1 serializer dùng để response từ server khi gọi 1 API:

```
class GetProAttributeValueSerializer(serializers.ModelSerializer):
    class Meta:
        model = ProductAttributeValue
        fields = ('pk', 'product', 'product_attribute', 'value',)
```

Code để tạo 1 serializer dùng để đọc dữ liệu từ request khi gửi API lên server:

```
class KeywordAttributeValueSerializer(serializers.Serializer):
    pk = serializers.IntegerField(default=0)
    product = serializers.IntegerField(default=0)
    product_attribute = serializers.IntegerField(default=0)
    value = serializers.CharField(max_length=100,
    default="null")
```

- **Views:**

Tại đây gồm nhiều class để xử lý cho các API khi gửi lên server với method khác nhau (GET/ POST/ PUT/ DELETE/ ...).

APIView dùng để xử lý method POST để lấy và trả về dữ liệu giá trị thuộc tính của 1 sản phẩm:

Tạo class cho API `GetProAttributeValueAPIView`

```
class GetProAttributeValueAPIView(APIView):
```

Tạo hàm để xử lý method POST cho API trên

```
def post(self, request):
    mydata = KeywordAttributeValueSerializer(data=request.data)
    if not mydata.is_valid():
        return Response('Something wrong! Check your data',
            status=status.HTTP_400_BAD_REQUEST)

    pk = mydata.data['pk']
    product = mydata.data['product']
    product_attribute = mydata.data['product_attribute']
    value = mydata.data['value']
    list_product_attribute_value =
ProductAttributeValue.objects.all()

    if pk != 0:
        list_product_attribute_value =
list_product_attribute_value.filter(pk=pk)
    else:
        if product != 0:
            list_product_attribute_value =
list_product_attribute_value.filter(product=product)
        if product_attribute != 0:
            list_product_attribute_value =
list_product_attribute_value.filter(product_attribute=product_attrib
ute)
        if value != "null":
            list_product_attribute_value =
list_product_attribute_value.filter(value__icontains=value)

    mydata =
GetProAttributeValueSerializer(list_product_attribute_value,
many=True)
    return Response(data=mydata.data, status=status.HTTP_200_OK)
```

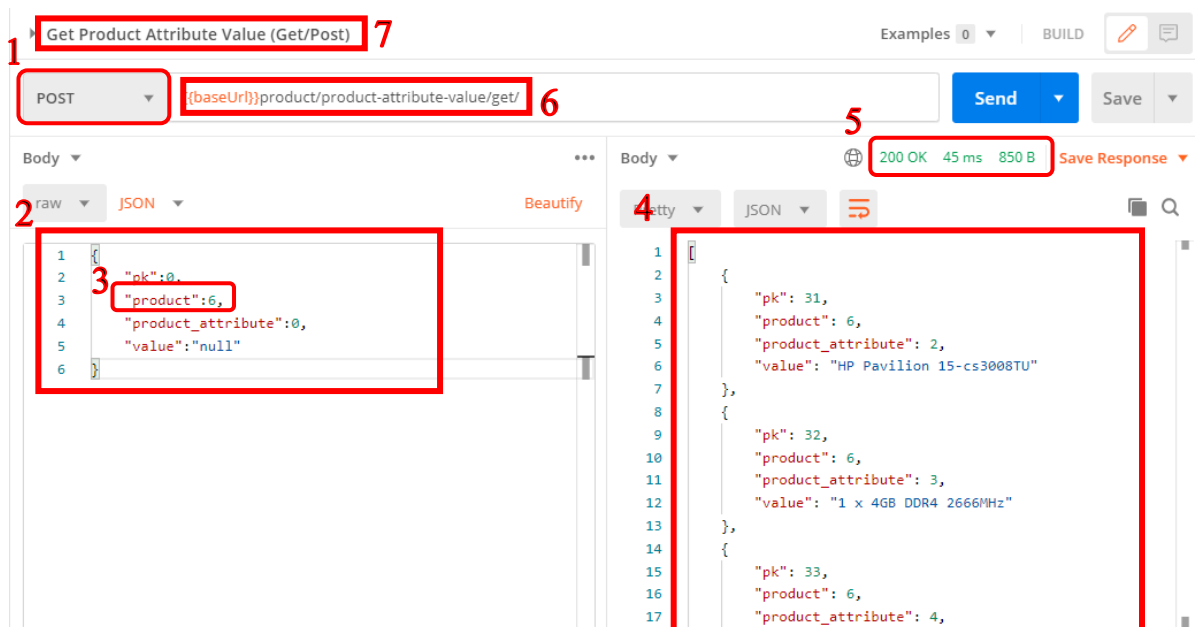
Sau các bước đó em đã có được 1 API xử lý lấy giá trị thuộc tính của sản phẩm theo các từ khóa được gửi lên server.

- Urls:

Đây là bước cuối cùng giúp cho ta có thể gọi API để sử dụng. Vào file urls của dự án, định nghĩa đường dẫn và chỉ định đường dẫn sẽ gọi đến API nào để xử lý bằng cách thêm đoạn code sau vào `urlpatterns = []`:

```
path('product/product-attribute-value/get/',  
GetProAttributeValueAPIView.as_view()),
```

Kết quả khi chạy API đã tạo trên:



Hình 3.13. Ví dụ mẫu gọi API

- 1: Gọi API với method POST
- 2: Danh sách dữ liệu gửi lên request
- 3: Tìm giá trị thuộc tính của sản phẩm có mã là 6
- 4: Kết quả trả về từ API
- 5: Status code trả về từ API
- 6: Đường dẫn (URL) của API (`{{baseUrl}}`): tên biến của môi trường trong Postman với giá trị là 'http://127.0.0.1:8000/'
- 7: Tên API

3.4.1. API Product

Cách làm tương tự với các bước trên, em có được các API sau:

GET Get Category (Get/Post)

API dùng để lấy danh sách các loại sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

The screenshot shows a REST client interface for the endpoint `{{baseUrl}}product/category/get/` with a GET method. The response status is 200 OK, with a response time of 36 ms and a body size of 644 B. The response body is displayed in a pretty-printed JSON format:

```
[
  {
    "pk": 1,
    "name": "Laptop",
    "description": null,
    "category_belong": null
  },
  {
    "pk": 2,
    "name": "Màn hình",
    "description": null,
    "category_belong": null
  },
  {
    "pk": 3,
    "name": "Bàn phím",
    "description": null
  }
]
```

Hình 3.14. API Get Category GET: Ok

Tìm loại sản phẩm với tên có chứa chuỗi “lap”.

The screenshot shows a REST client interface for the endpoint `{{baseUrl}}product/category/get/` with a POST method. The request body is a JSON object: `{ "pk": 0, "name": "lap", "category_belong": 0 }`. The response status is 200 OK, with a response time of 35 ms and a body size of 355 B. The response body is displayed in a pretty-printed JSON format:

```
[
  {
    "pk": 1,
    "name": "Laptop",
    "description": null,
    "category_belong": null
  }
]
```

GET Get Manufacturer (Get/Post)

API dùng để lấy danh sách các nhà sản xuất (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

GET `{{baseUrl}}product/manufacturer/get/` **Send** **Save**

Params **Query Params**

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body **200 OK 33 ms 504 B** **Save Response**

Pretty JSON

```
1 {
2   {
3     "pk": 1,
4     "name": "Asus",
5     "description": null
6   },
7   {
8     "pk": 2,
9     "name": "Dell",
10    "description": null
11  },
12  {
13    "pk": 3,
14    "name": "HP",
15    "description": null
16  },
17  }
```

Hình 3.15. API Get Manufacturer GET: Ok

Tìm nhà sản xuất với tên có chứa chuỗi “sus”.

POST `{{baseUrl}}product/manufacturer/get/` **Send** **Save**

Body **raw JSON Beautify**

```
1 {
2   "pk": 0,
3   "name": "sus"
4 }
```

Body **200 OK 30 ms 330 B** **Save Response**

Pretty JSON

```
1 {
2   {
3     "pk": 1,
4     "name": "Asus",
5     "description": null
6   }
7 }
```


GET Get Discount list (Get/Post)

API dùng để lấy danh sách các thông tin khuyến mãi (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

The screenshot shows a REST client interface for the endpoint `Get Discount list (Get/Post)`. The method is set to **GET** and the URL is `{{baseUrl}}product/discount/get/`. The response status is **200 OK** with a response time of **171 ms** and a body size of **992 B**. The response body is displayed in JSON format, showing a list of two discount items:

```
[{"pk": 1, "code": "TEST01", "discount_percent": 0.1, "on_bill": true, "active": true, "apply_date": "2020-10-25T20:28:39+07:00", "end_date": "2020-10-27T20:32:01+07:00"}, {"pk": 2, "code": "Asus11", "discount_percent": 0.15, "on_bill": false, "active": true, "apply_date": "2020-10-28T17:07:41+07:00", "end_date": null}]
```

Hình 3.16. API Get Discount GET: Ok

Lấy khuyến mãi có mã 4 và điều kiện là giảm giá trên toàn bill.

The screenshot shows a REST client interface for the endpoint `Get Discount list (Get/Post)`. The method is set to **POST** and the URL is `{{baseUrl}}product/discount/get/`. The response status is **200 OK** with a response time of **34 ms** and a body size of **426 B**. The response body is displayed in JSON format, showing a single discount item that matches the filters:

```
[{"pk": 4, "code": "Mouse101", "discount_percent": 0.2, "on_bill": false, "active": true, "apply_date": "2020-11-01T22:36:43+07:00", "end_date": null}]
```

POST Get Discount Item list (Get/Post)

API dùng để lấy danh sách khuyến mãi của các sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

Get Discount Item list (Get/Post) Examples 0 BUILD

GET http://127.0.0.1:8000/product/discount-item/get/ Send Save

Params Query Params

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Pretty JSON

```
1 [
2   {
3     "pk": 5,
4     "discount": 2,
5     "product": 1
6   },
7   {
8     "pk": 6,
9     "discount": 2,
10    "product": 2
11  },
12  {
13    "pk": 7,
14    "discount": 2,
15    "product": 7
16  },
17  {
```

200 OK 40 ms 963 B Save Response

Tìm khuyến mãi của sản phẩm mã 22 với mã khuyến 4.

Get Discount Item list (Get/Post) Examples 0 BUILD

POST http://127.0.0.1:8000/product/discount-item/get/ Send Save

Body raw JSON Beautify

```
1 {
2   "pk": 0,
3   "discount": 4,
4   "product": 22
5 }
```

Body Pretty JSON

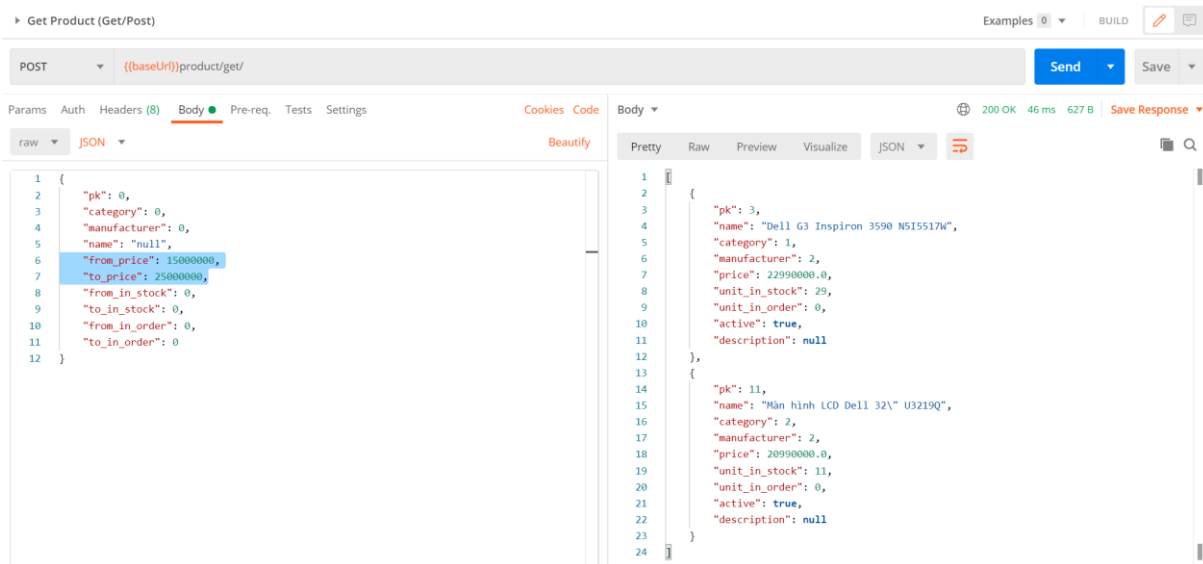
```
1 [
2   {
3     "pk": 13,
4     "discount": 4,
5     "product": 22
6   }
7 ]
```

200 OK 26 ms 324 B Save Response

GET Get Product (Get/Post)

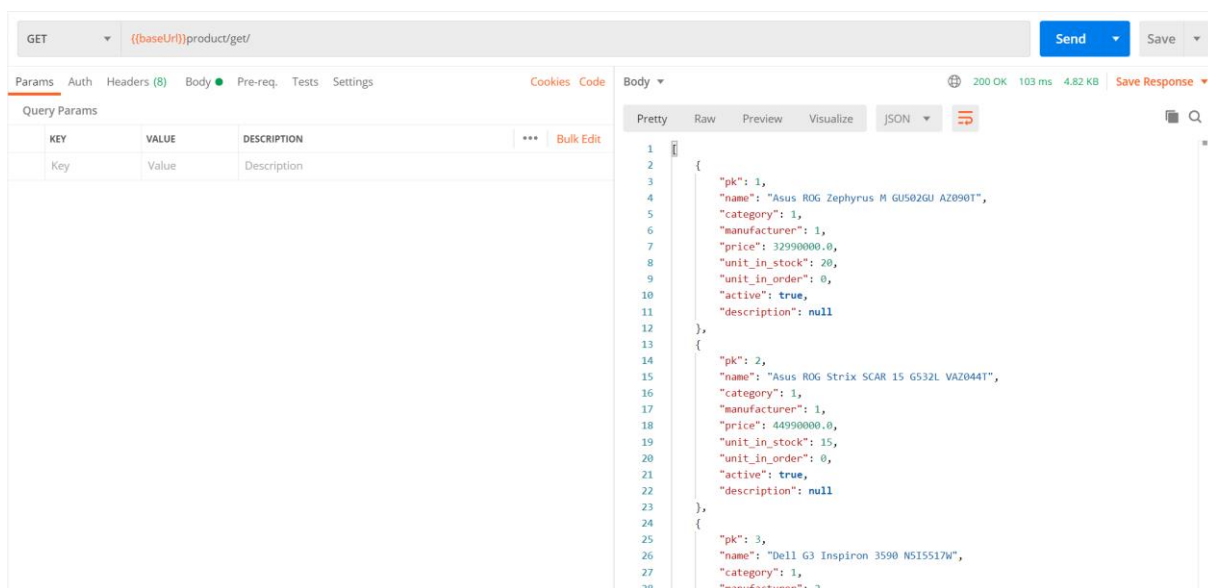
API dùng để lấy danh sách tất cả sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:



Hình 3.17. API Get Product GET: Ok

Lấy các sản phẩm có giá từ 15000000 đến 25000000.



GET Get Product Attribute

API dùng để lấy danh sách tất cả thuộc tính của tất cả sản phẩm (GET).

Kết quả:

The screenshot displays a REST client interface for the endpoint `GET {{baseUrl}}product/product-attribute/get/`. The response is a 200 OK status with a 22 ms response time and a 1.37 KB body. The response body is shown in JSON format, containing an array of six product attributes.

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

```
1 {
2   {
3     "pk": 1,
4     "name": "Thông số màn hình",
5     "description": null
6   },
7   {
8     "pk": 2,
9     "name": "CPU",
10    "description": null
11  },
12  {
13    "pk": 3,
14    "name": "Ram",
15    "description": null
16  },
17  {
18    "pk": 4,
19    "name": "Ổ cứng",
20    "description": null
21  },
22  {
23    "pk": 5,
24    "name": "Pin",
25    "description": null
26  },
27  {
28    "pk": 6,
29    "name": "Ảnh",
30    "description": null
31  }
32 }
```

Hình 3.18. API Get Product Attribute: Ok

POST Get Product Attribute Value (Get/Post)

API dùng để lấy danh sách tất cả giá trị thuộc tính của các sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

The screenshot shows a REST client interface with a GET request to the endpoint `{{baseUrl}}product/product-attribute-value/get/`. The response is a JSON array of product attributes, including product ID, product name, and product description.

```
1 {
2   {
3     "pk": 1,
4     "product": 1,
5     "product_attribute": 1,
6     "value": "15.6\" FHD (1920 x 1080) IPS, 100% sRGB, 240Hz, 3ms, 300nits,
7       Pantone® Validated, NanoEdge"
8   },
9   {
10    "pk": 2,
11    "product": 1,
12    "product_attribute": 2,
13    "value": "Intel Core i7-9750H 2.6GHz up to 4.5GHz 12MB"
14  },
15  {
16    "pk": 3,
17    "product": 1,
18    "product_attribute": 3,
19    "value": "16GB DDR4 2666MHz Onboard (1x SO-DIMM socket, up to 32GB SDRAM)"
20  },
21  {
22    "pk": 4,
23    "product": 1,
24    "product_attribute": 4,
25    "value": "512GB SSD PCIE G3X4 (Support RAID 0) (2 slots)"
26  },
27 }
```

Hình 3.19. API Get Product Attribute Value: Ok

Lấy danh sách giá trị thuộc tính của sản phẩm có mã là 5.

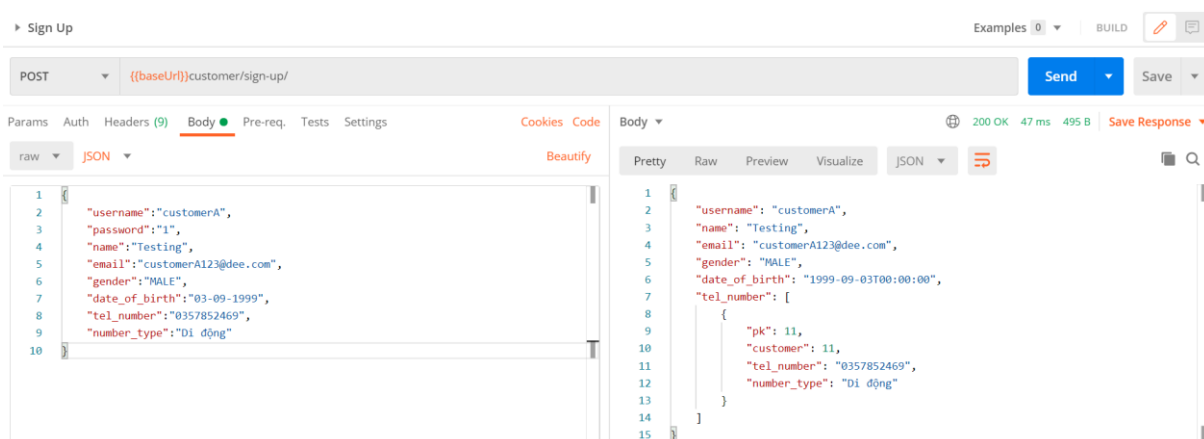
The screenshot shows a REST client interface with a POST request to the endpoint `{{baseUrl}}product/product-attribute-value/get/`. The request body is a JSON object with `"product": 5` and `"product_attribute": 0`. The response is a JSON array of product attributes, including product ID, product name, and product description.

```
1 {
2   {
3     "pk": 25,
4     "product": 5,
5     "product_attribute": 2,
6     "value": "HP ENVY 13-aq1057TX"
7   },
8   {
9     "pk": 26,
10    "product": 5,
11    "product_attribute": 3,
12    "value": "1 x 8GB Onboard DDR4 2400MHz"
13  },
14  {
15    "pk": 27,
16    "product": 5,
17    "product_attribute": 4,
18    "value": "512GB SSD M.2 NVMe"
19  },
20  {
21    "pk": 28,
22    "product": 5,
23    "product_attribute": 1,
24    "value": "13.3\" IPS (1920 x 1080)"
25  },
26 }
```

3.4.2. API Customer

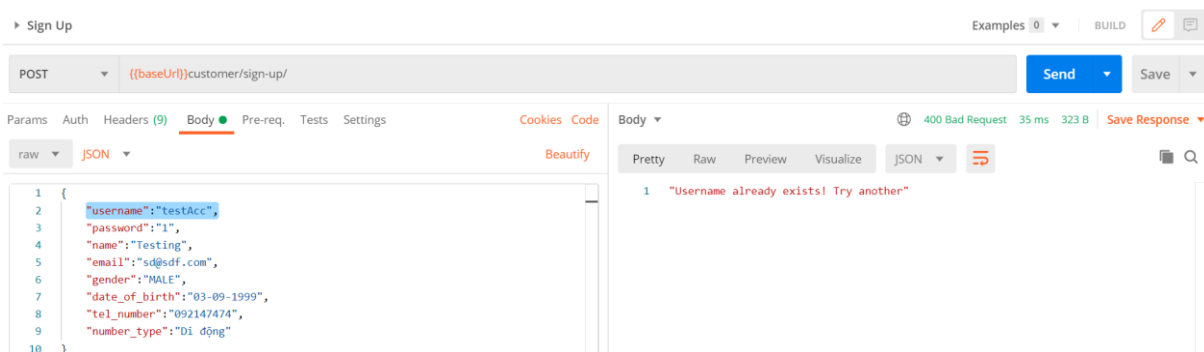
POST Sign Up

Đây là API với method POST giúp khách hàng tạo tài khoản mới vào hệ thống bằng việc cung cấp các thông tin cần thiết.

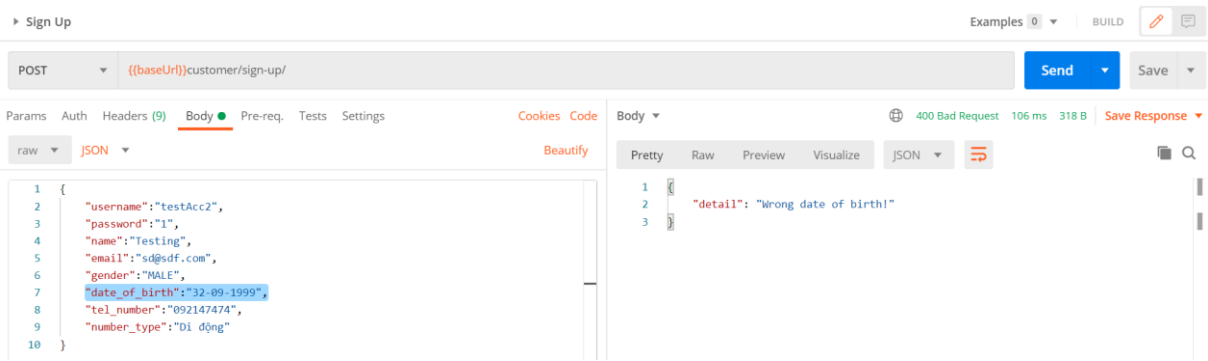


Hình 3.20. API Sign Up: Ok

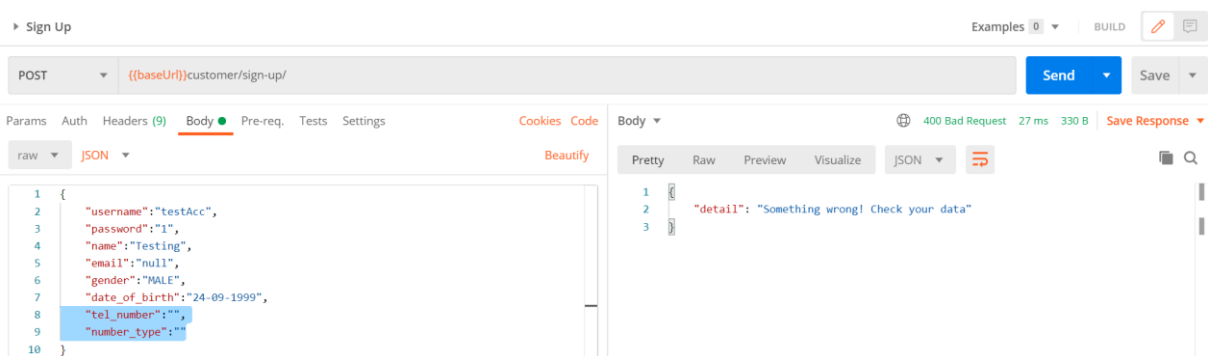
API sẽ báo lỗi nếu trùng tên tài khoản, email, thiếu hoặc sai các thông tin cần thiết cho việc tạo tài khoản.



Hình 3.21. API Sign Up: Tên tài khoản đã tồn tại



Hình 3.22. API Sign Up: Sai thông tin ngày sinh



Hình 3.23. API Sign Up: Sai thông tin SĐT

Trước khi lưu thông tin tài khoản khách hàng vào cơ sở dữ liệu, mật khẩu của khách hàng sẽ được mã hóa bằng hàm băm sha256 cung cấp bởi thư viện hashlib trong python để tăng tính bảo mật.

Đoạn code em dùng để băm mật khẩu

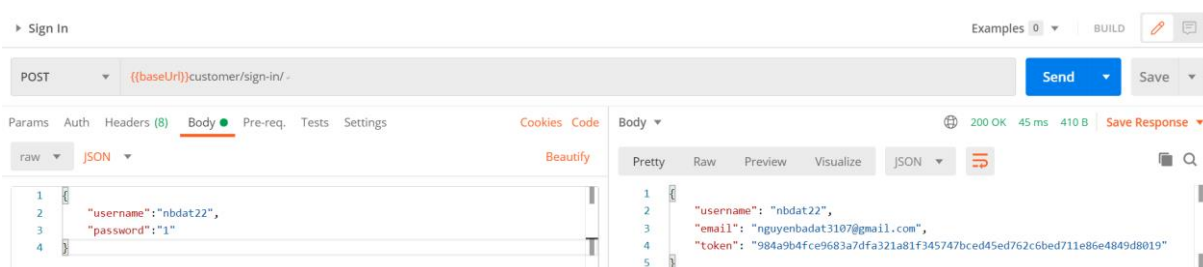
```
password = hashlib.sha256(password.strip().encode("utf-8")).hexdigest()
```

Kết quả sau khi tạo tài khoản khách hàng bằng API Sign Up:

| | | | | | | |
|----|-----------|---|---------|----------------------|------|------------|
| 11 | customerA | 6b86b273ff34fce19d6b804eff5a3f5747ada4ea... | Testing | customerA123@dee.com | MALE | 1999-09-03 |
|----|-----------|---|---------|----------------------|------|------------|

POST Sign In

API phục vụ cho việc khách hàng đăng nhập vào hệ thống. Sau khi cung cấp thông tin đăng nhập hợp lệ, API sẽ trả về 1 token xác thực, khách hàng (web hoặc các ứng dụng khác) sẽ sử dụng token này để thực hiện các chức năng khác có yêu cầu xác thực bằng token.



Hình 3.24. API Sign In: Ok

Token được tạo ra bằng cách băm chuỗi ghép bởi tên đăng nhập của khách hàng và ngày giờ gửi API Sign In:

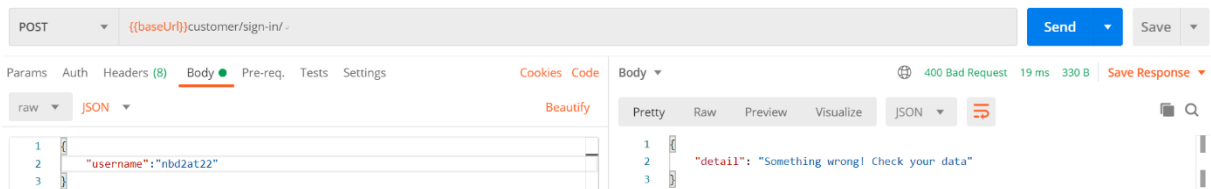
```
key = hashlib.sha256((customer.username +  
str(datetime.now()))).encode("utf-8")).hexdigest()
```

Token với khóa mới được tạo sẽ được lưu xuống cơ sở dữ liệu cùng với ngày giờ tạo khóa đó.

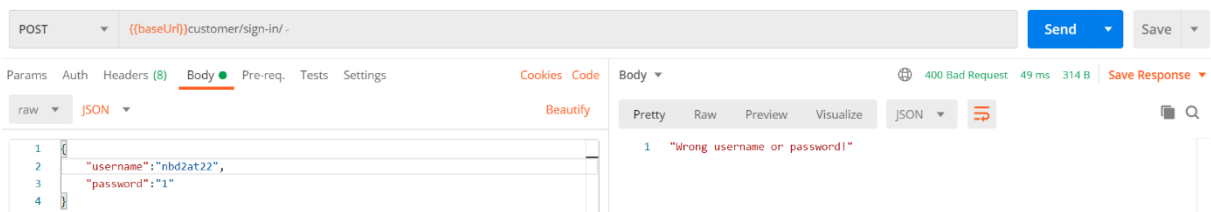
| id | key | customer_id | created |
|----|--|-------------|----------------------------|
| 1 | 984a9b4fce9683a7dfa321a81f345747bced45e... | 3 | 2020-11-03 15:24:55.409049 |

Mỗi khách hàng chỉ có một mã token duy nhất, sẽ được tạo mới mỗi lần đăng nhập vào hệ thống và các token đó sẽ tự hết hạn sau một tiếng đồng hồ.

API sẽ báo lỗi khi khách hàng nhập sai tên đăng nhập, mật khẩu hoặc thiếu một trong hai thông tin đó.



Hình 3.25. API Sign In: Sai thông tin đăng nhập

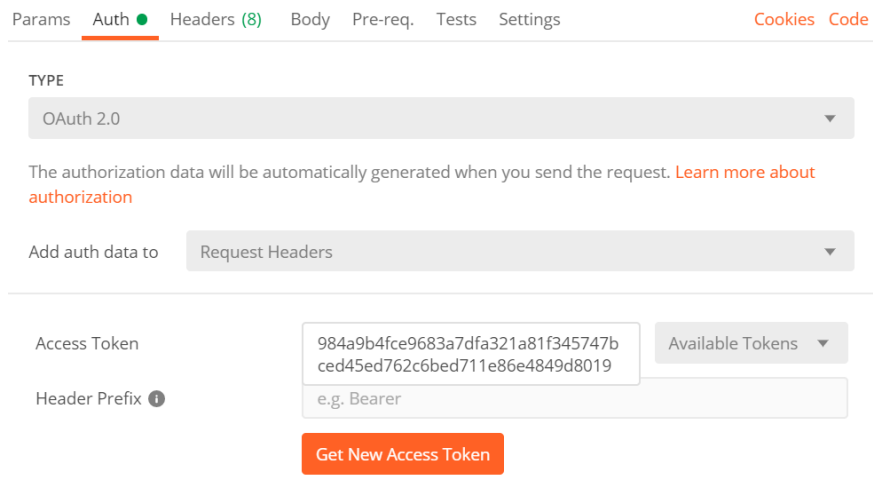


Hình 3.26. API Sign In: Thiếu thông tin đăng nhập

Xác thực token:

Sau chức năng Sign In, hệ thống đã có được token để xác thực cho các API tiếp theo.

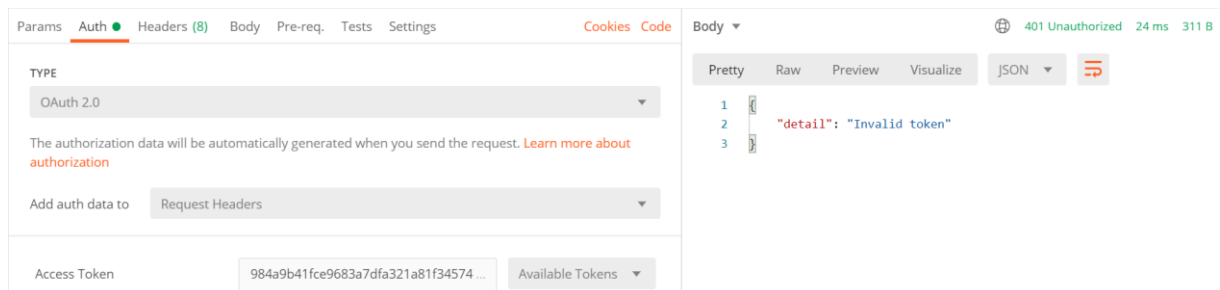
Các API có yêu cầu về xác thực sẽ phải thêm thông Authorization vào headers trước khi gọi API:



Hình 3.27. Thêm token xác thực vào headers của API

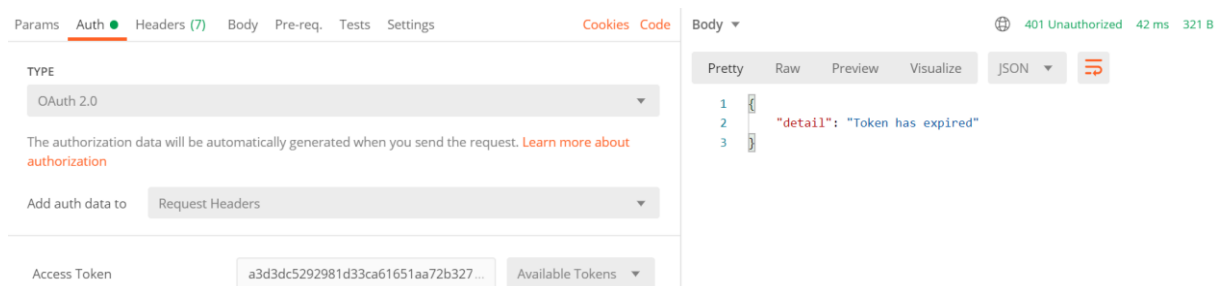
Khi gửi API đi, server sẽ kiểm tra token đó có hợp lệ hay không với các điều kiện:

- Token phải tồn tại trong kho dữ liệu.



Hình 3.28. Xác thực token, token không tồn tại

- Token vẫn còn hạn sử dụng (1 tiếng từ khi khách hàng đăng nhập tạo mới token).



Hình 3.29. Xác thực token, token hết hạn

Code dùng để xác thực token:

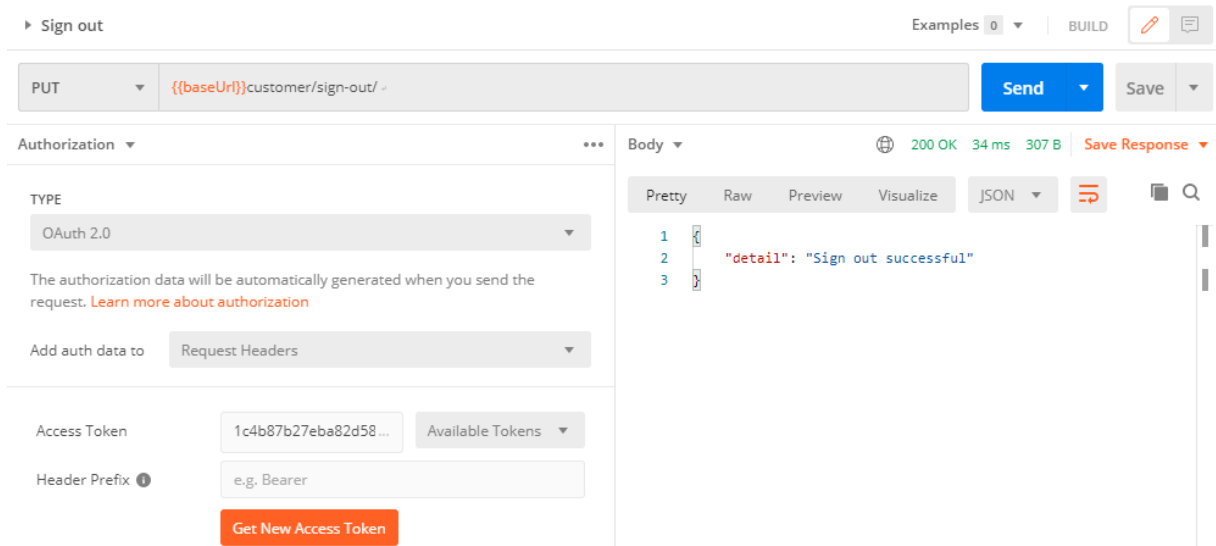
```
try:
    token = request.headers['Authorization']
except:
    return Response({
        "detail": "Token not found"
    },
    status=status.HTTP_203_NON_AUTHORITATIVE_INFORMATION)
token = Token.objects.filter(key=token).first()
if token is None:
    return Response({
        "detail": "Invalid token"
    }, status=status.HTTP_401_UNAUTHORIZED)
if (token.created + timedelta(hours=1)) <
datetime.now(timezone.utc):
    return Response({
        "detail": "Token has expired"
    }, status=status.HTTP_401_UNAUTHORIZED)
```

Sau khi xác thực token thành công, API được phép thực hiện chức năng của nó và API đó có thể lấy các thông tin tài khoản khách hàng từ token đó bằng đoạn code sau:

```
customer =
Customer.objects.filter(pk=token.customer.pk).first()
```

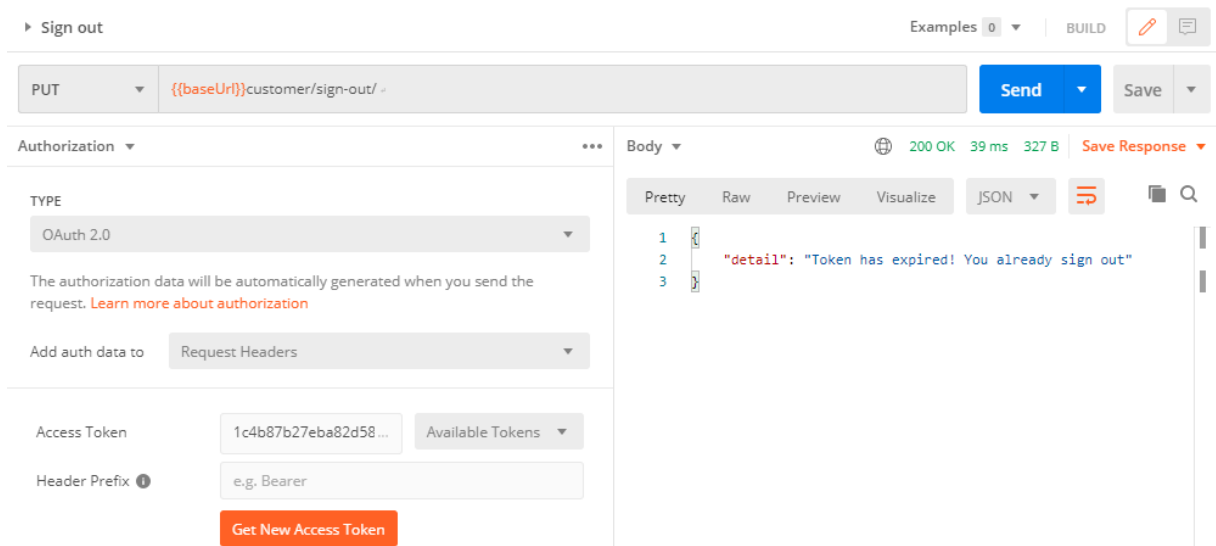
PUT Sign out [Xác thực token]

API giúp khách hàng đăng xuất bằng cách vô hiệu hóa token của khách hàng đó.



Hình 3.30. API Sign Out: Ok

API sẽ thông báo cho khách hàng khi token đó đã quá hạn (đã được sign out).

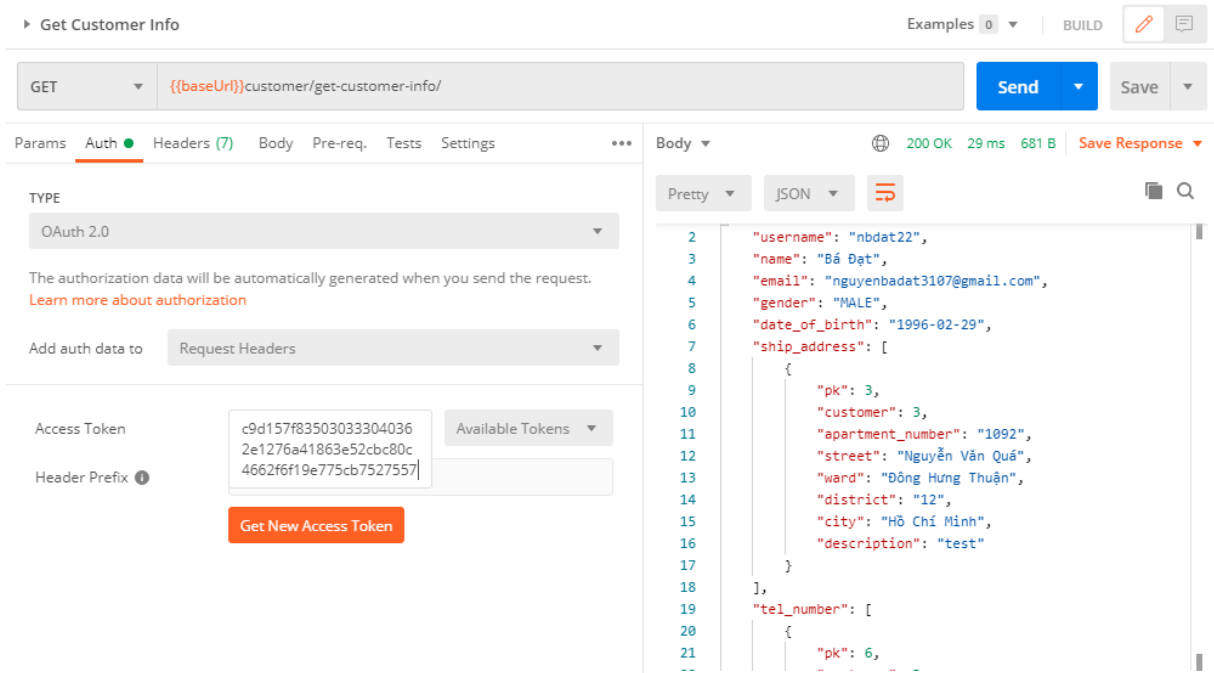


Hình 3.31. API Sign Out: Token quá hạn

GET Get Customer Info

[Xác thực token]

API dùng để lấy các thông tin tài khoản của khách hàng.

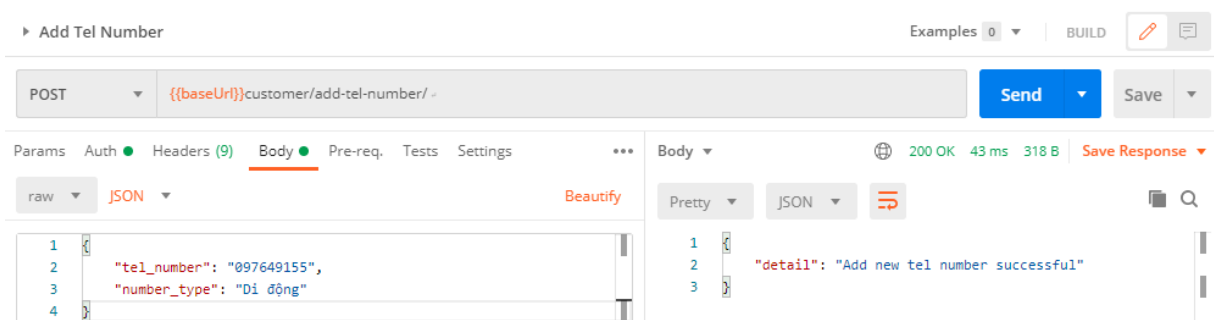


Hình 3.32. API Get Customer Info: Ok

POST Add Tel Number

[Xác thực token]

API thêm số điện thoại mới cho khách hàng. (Báo lỗi nếu không đủ thông tin)

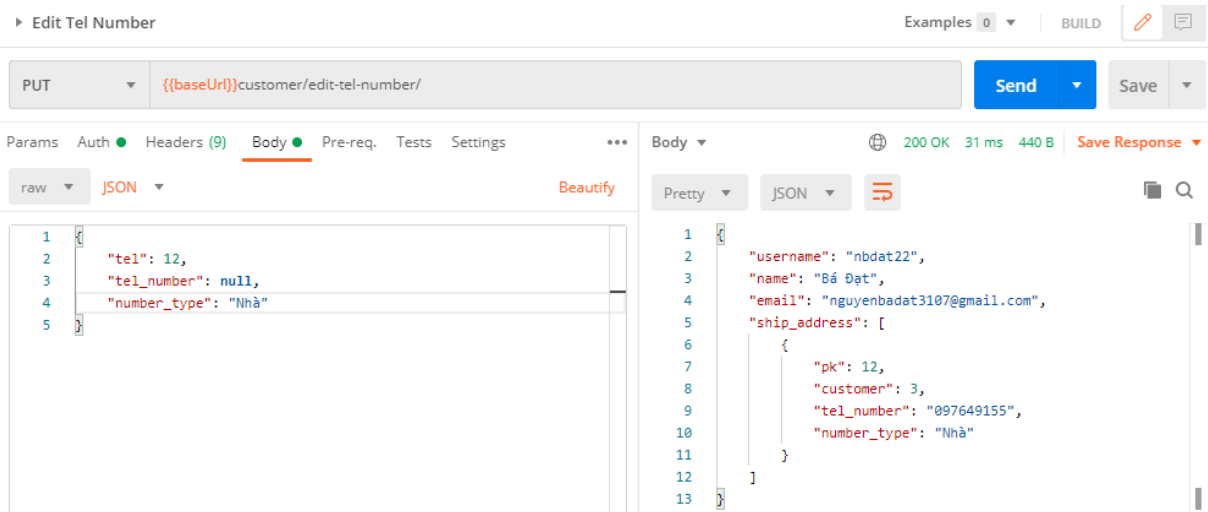


Hình 3.33. API Add Tel Number: Ok

PUT Edit Tel Number

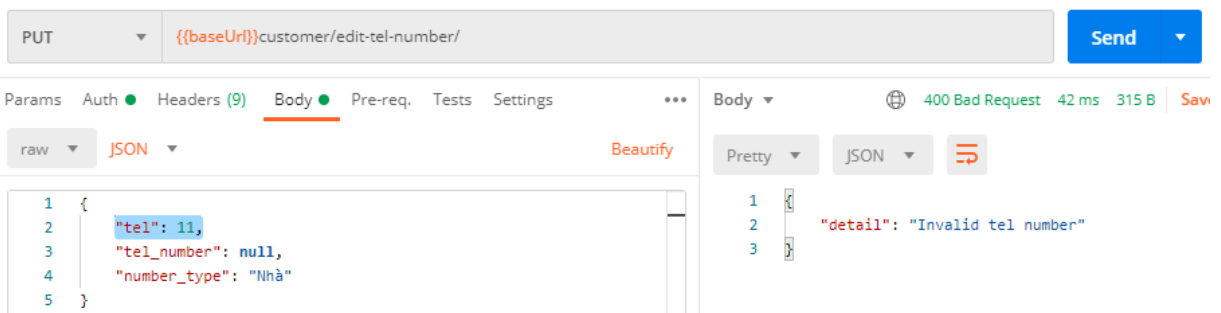
[Xác thực token]

API sửa thông tin số điện thoại của khách hàng.



Hình 3.34. API Edit Tel Number: Ok

API này cũng sẽ kiểm tra mã SĐT vừa nhận có phải thuộc sở hữu của khách hàng gửi API đó đi hay không qua token khách hàng vừa gửi đi. Nếu không khớp sẽ báo lỗi.

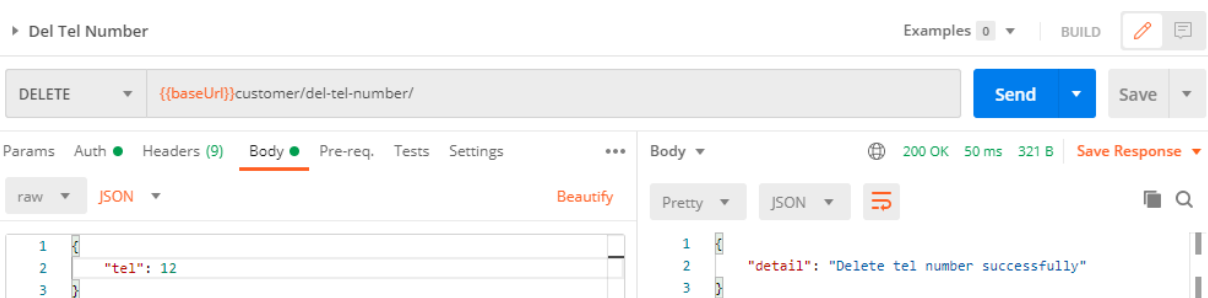


Hình 3.35. API Edit Tel Number: Sai mã SĐT

DEL Del Tel Number

[Xác thực token]

API xóa số điện thoại của khách hàng.

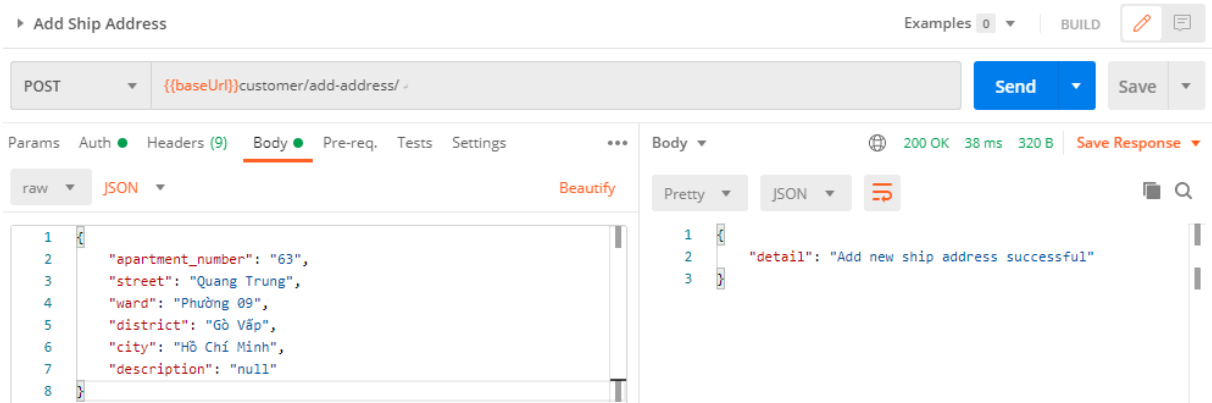


Hình 3.36. API Delete Tel Number: Ok

POST Add Ship Address

[Xác thực token]

API thêm thông tin địa chỉ giao hàng mới cho khách hàng. (Báo lỗi nếu không đủ thông tin)

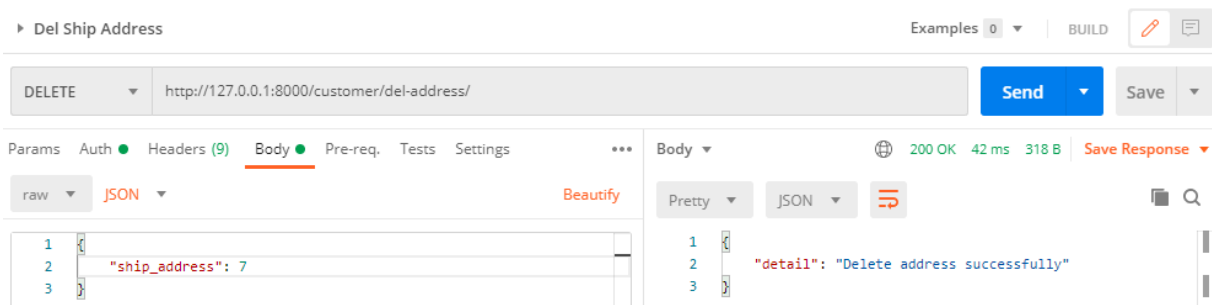


Hình 3.37. API Add Ship Address: Ok

DEL Del Ship Address

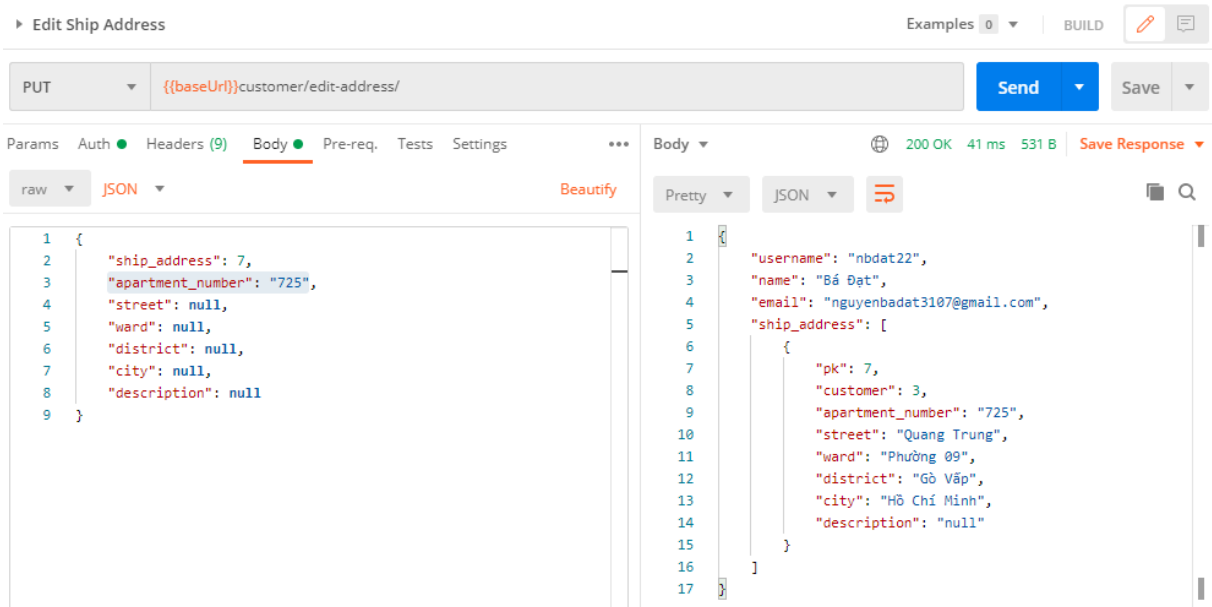
[Xác thực token]

API xóa địa chỉ giao hàng.



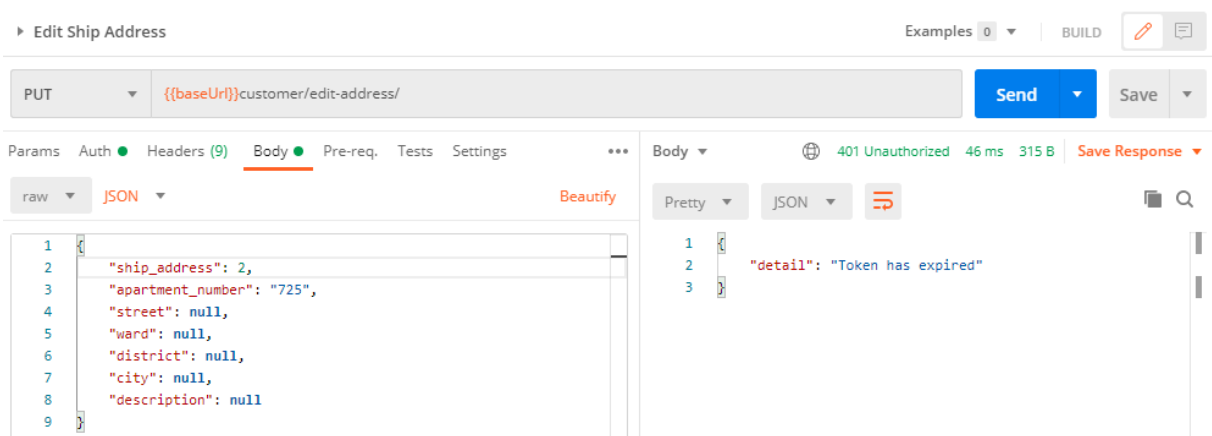
Hình 3.38. API Delete Ship Address: Ok

API dùng để sửa thông tin địa chỉ giao hàng. API chỉ sửa những thông tin gửi lên có giá trị khác null.



Hình 3.39. API Edit Ship Address: Ok

API này cũng sẽ kiểm tra mã địa chỉ vừa nhận có phải thuộc sở hữu của khách hàng gửi API đó đi hay không qua token khách hàng vừa gửi đi. Nếu không khớp sẽ báo lỗi.

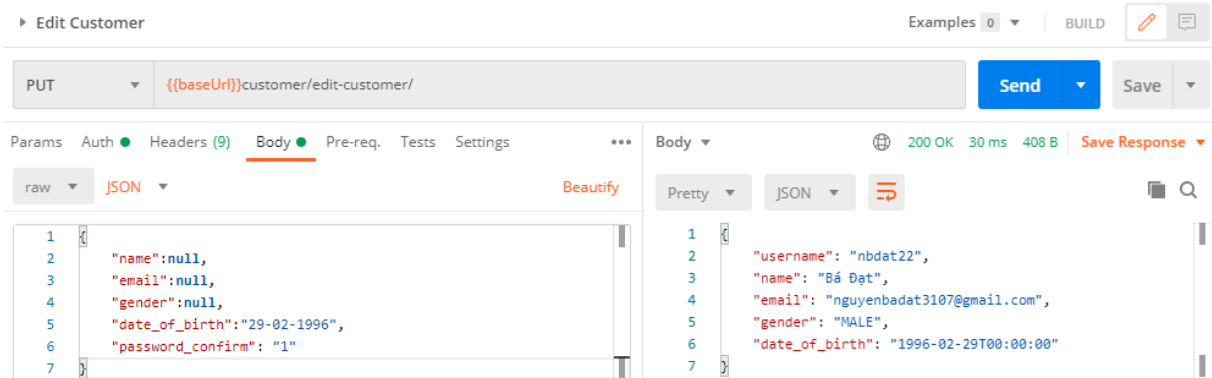


Hình 3.40. API Edit Ship Address: Sai mã địa chỉ

PUT Edit Customer

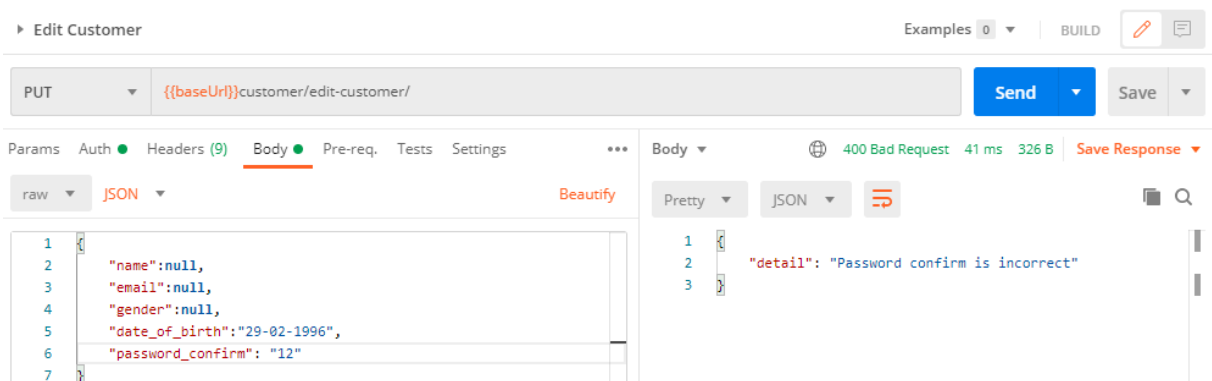
[Xác thực token]

API sửa thông tin tài khoản của khách hàng. API chỉ sửa những thông tin gửi đi có giá trị khác null.



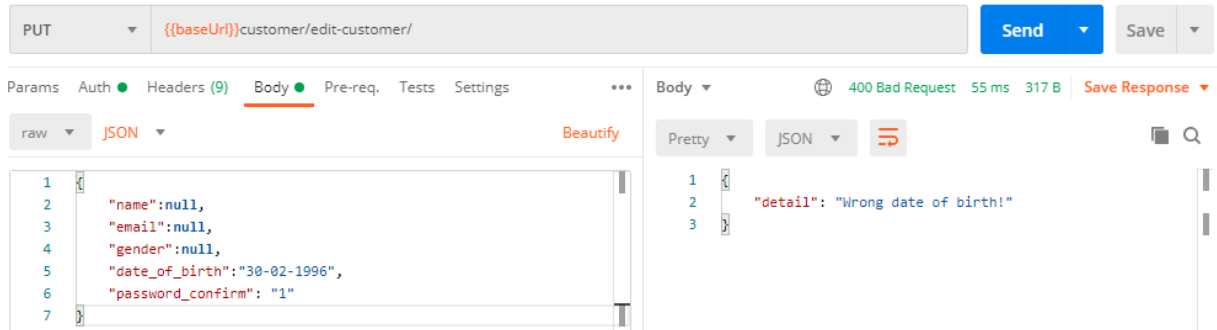
Hình 3.41. API Edit Customer

API báo lỗi nếu mật khẩu xác thực không đúng.

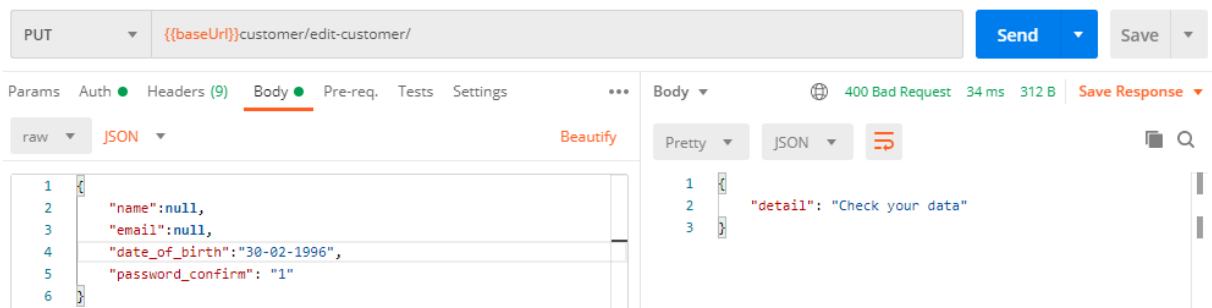


Hình 3.42. API Edit Customer: Sai mật khẩu

API báo lỗi nếu thiếu thông tin gửi đi hoặc sai dữ liệu ngày tháng năm sinh.



Hình 3.43. API Edit Customer: Sai dữ liệu ngày sinh

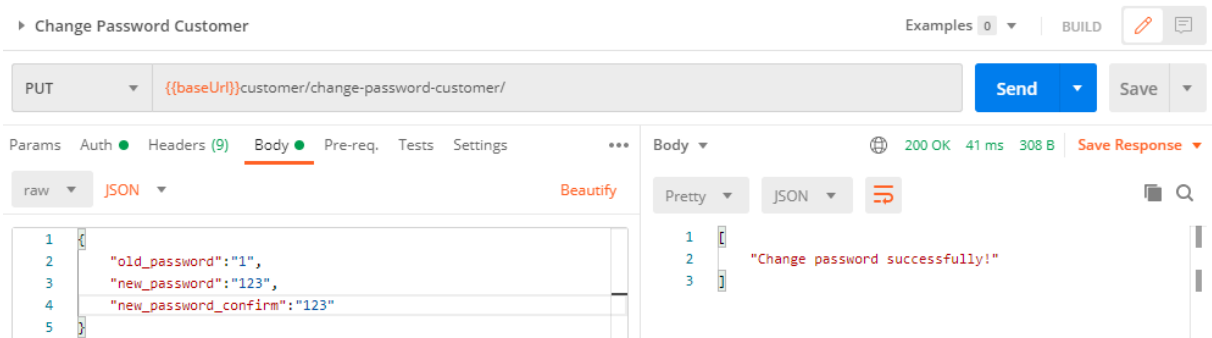


Hình 3.44. API Edit Customer: Gửi thiếu thông tin

PUT Change Password Customer

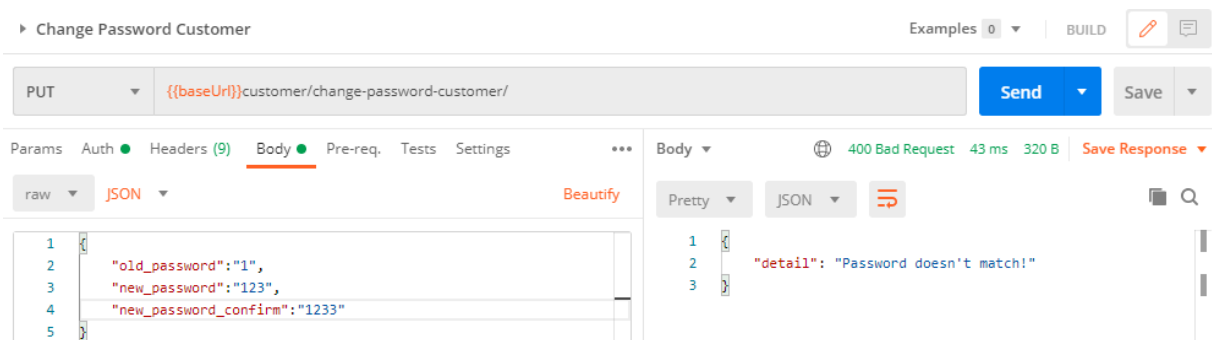
[Xác thực token]

API hỗ trợ khách hàng đổi mật khẩu tài khoản.



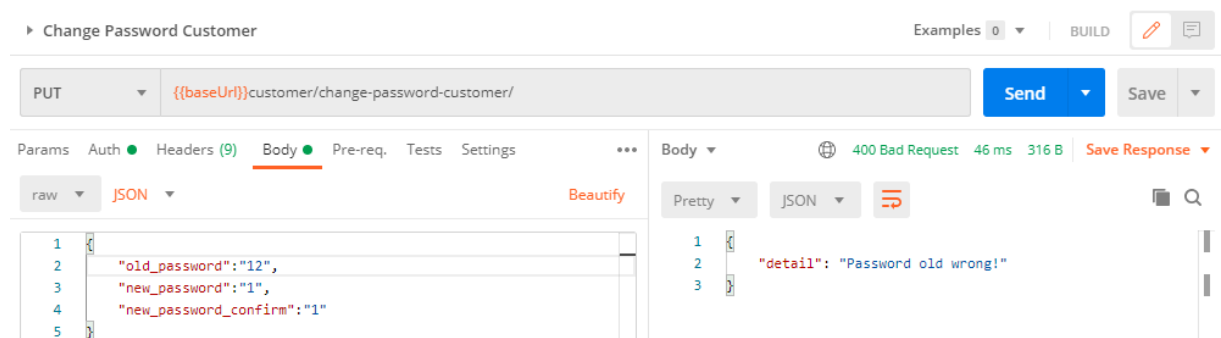
Hình 3.45. API Change Password: Ok

API báo lỗi nếu mật khẩu mới không trùng khớp.



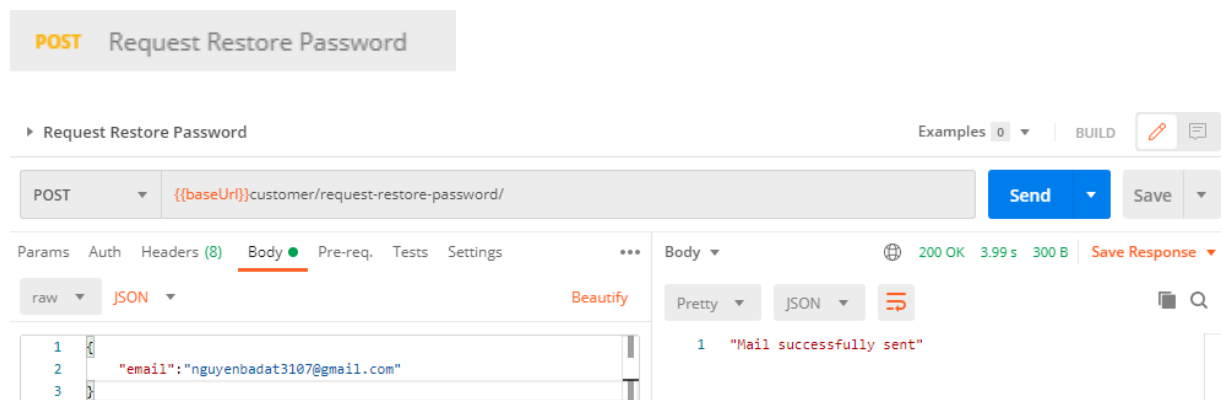
Hình 3.46. API Change Password: Mật khẩu không khớp

API báo lỗi nếu mật khẩu cũ không chính xác.

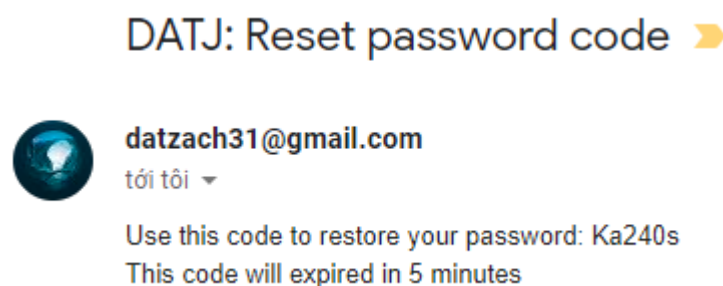


Hình 3.47. API Change Password: Sai mật khẩu cũ

Trong trường hợp khách hàng bị quên mật khẩu, họ có thể gọi API sau để gửi mã OTP qua email và dùng mã đó để khôi phục đặt lại mật khẩu mới.

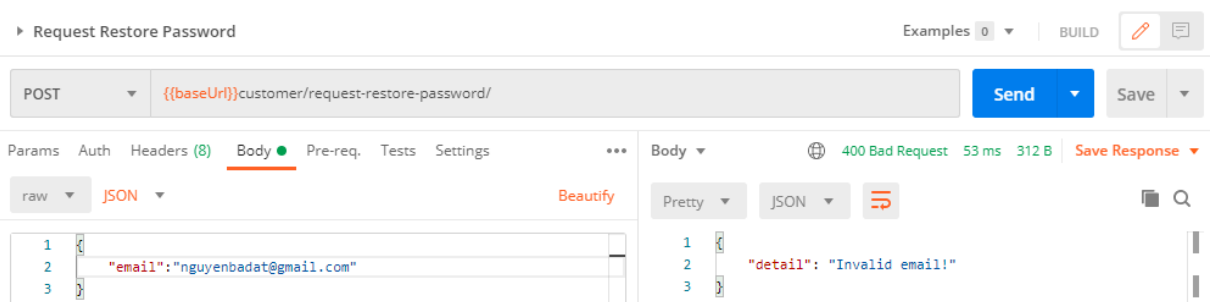


Hình 3.48. API Request Restore Password: Ok



Hình 3.49. Mail mã xác thực khôi phục mật khẩu

API sẽ kiểm tra email gửi đi có khớp với tài khoản khách hàng nào trong kho hay không, nếu email gửi đi đó không thuộc bất kì tài khoản nào hệ thống sẽ báo lỗi.



Hình 3.50. API Request Restore Password: Email không tồn tại

Nếu email hợp lệ, API sẽ tạo một đoạn mã xác thực liên kết đến tài khoản khách hàng có email vừa gửi kia và lưu đoạn mã đó xuống cơ sở dữ liệu. Sau khi tạo và lưu mã, đoạn mã đó sẽ được gửi cho khách hàng thông qua email vừa cung cấp, khách hàng sau đó có thể sử dụng đoạn mã đó để khôi phục đặt lại mật khẩu mới.

Code để tạo đoạn mã ngẫu nhiên gồm 6 ký tự:

```
letters_and_digits = string.ascii_letters + string.digits
code = ''.join((random.choice(letters_and_digits) for i in
range(6)))
```

Em dùng thư viện email [6] của Django để gửi thông tin khôi phục mật khẩu qua email của khách hàng bằng đoạn code sau:

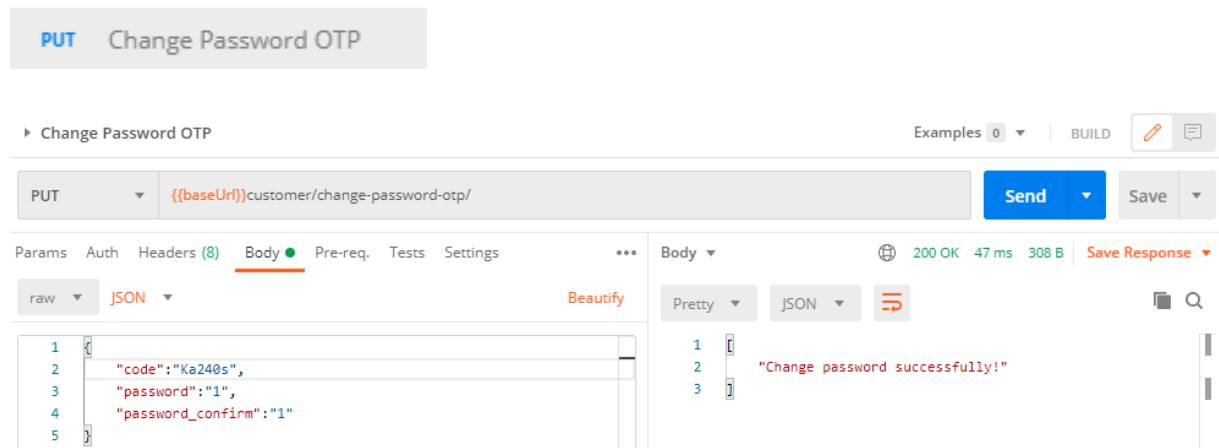
```
content = "Use this code to restore your password: " + code +
"\nThis code will expired in 5 minutes"
send_mail(
    'DATJ: Reset password code',
    content,
    settings.EMAIL_HOST_USER,
    [email],
    fail_silently=False,
)
```

`settings.EMAIL_HOST_USER` là email được dùng để gửi mail đi, được cấu hình tại file `settings` của dự án.

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_USE_TLS = True
EMAIL_PORT = 587
EMAIL_HOST_USER = '[user mail]'
EMAIL_HOST_PASSWORD = '[user mail password]'
```

`[email]` là địa chỉ dùng để gửi mail đến.

Sau khi khách hàng đã nhận được mail với đoạn mã khôi phục, họ có thể dùng API sau để đặt lại mật khẩu.

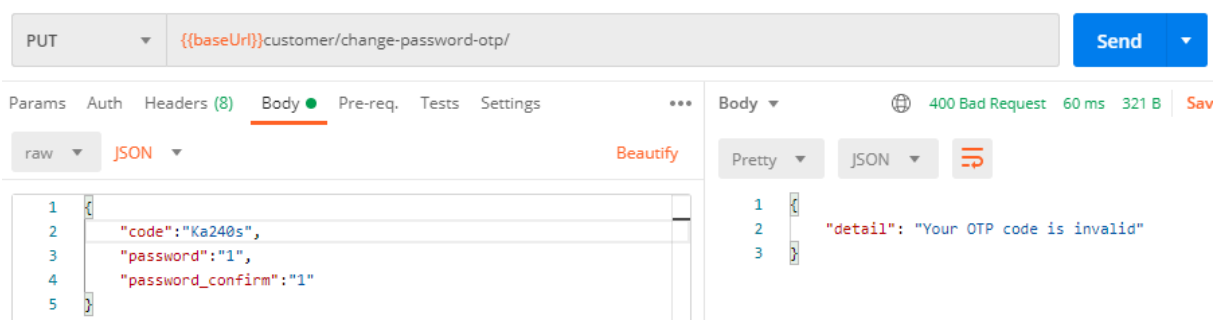


Hình 3.51. API Change Password OTP: Ok

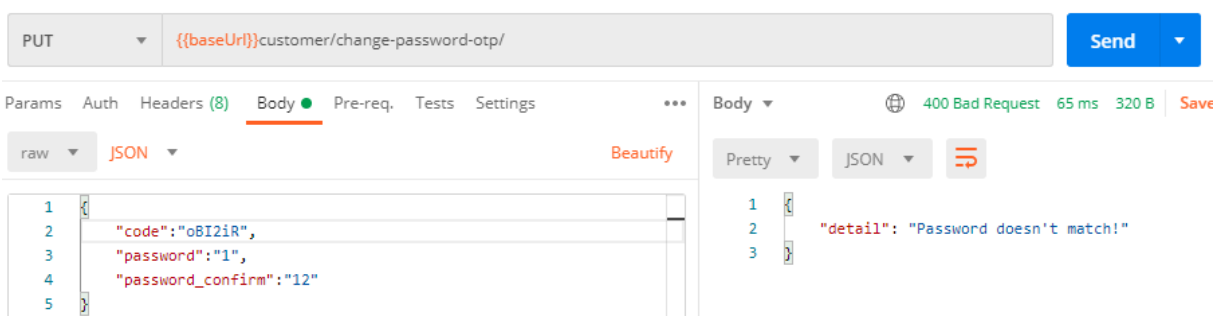
Do đoạn mã OTP đó đã được liên kết với tài khoản của khách hàng, nên API dễ dàng lấy thông tin khách hàng đó ra và thay đổi mật khẩu của họ sau khi API đã kiểm tra đoạn mã hợp lệ.

Sau khi đổi mật khẩu thành công, API sẽ vô hiệu hóa đoạn mã đó.

API sẽ báo lỗi khi mã xác thực không chính xác hoặc đã hết hạn (đã bị vô hiệu hóa).



API sẽ báo lỗi khi mật khẩu mới không khớp.

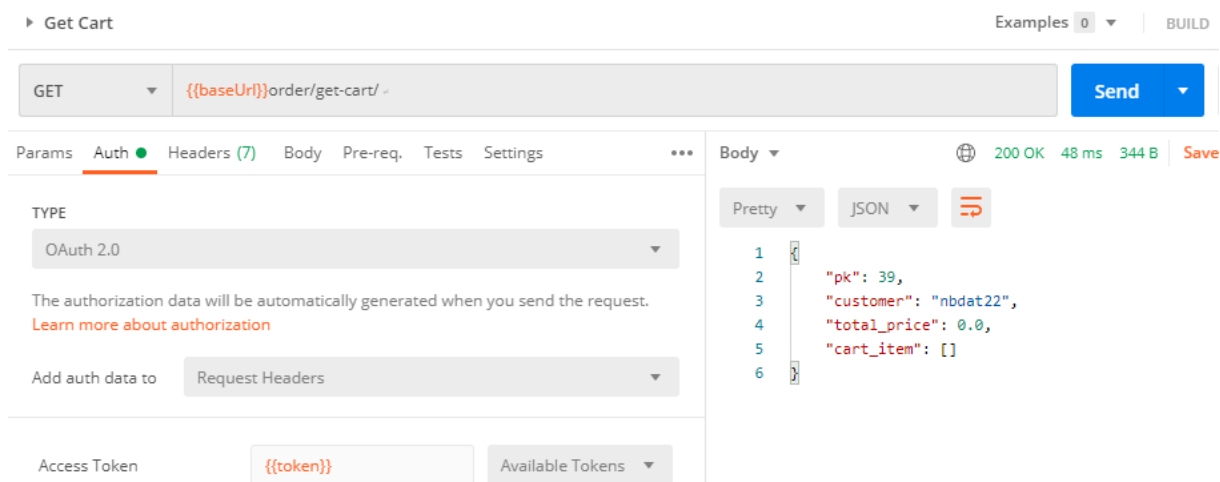


3.4.3. API Order

GET Get Cart

[Xác thực token]

API giúp khách hàng xem thông tin giỏ hàng hiện tại của mình.

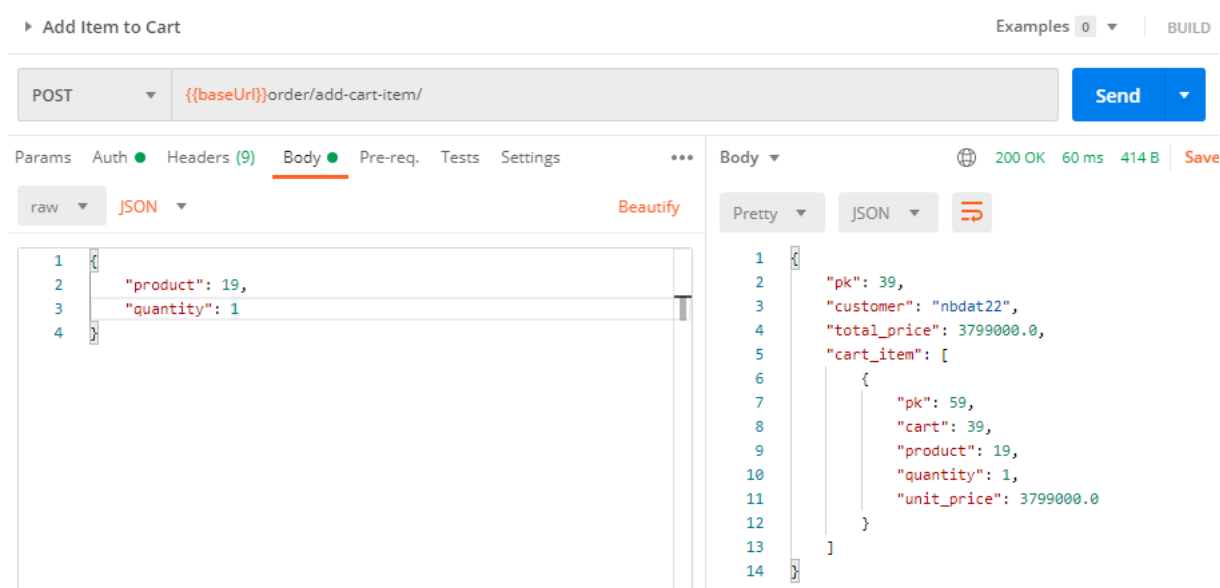


Hình 3.52. API Get Cart: Ok

POST Add Item to Cart

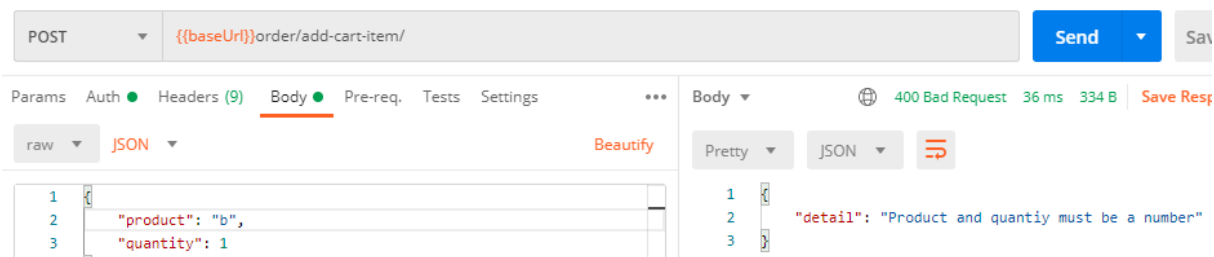
[Xác thực token]

API thêm sản phẩm vào giỏ hàng của khách hàng theo mã sản phẩm và số lượng cần thêm vào.

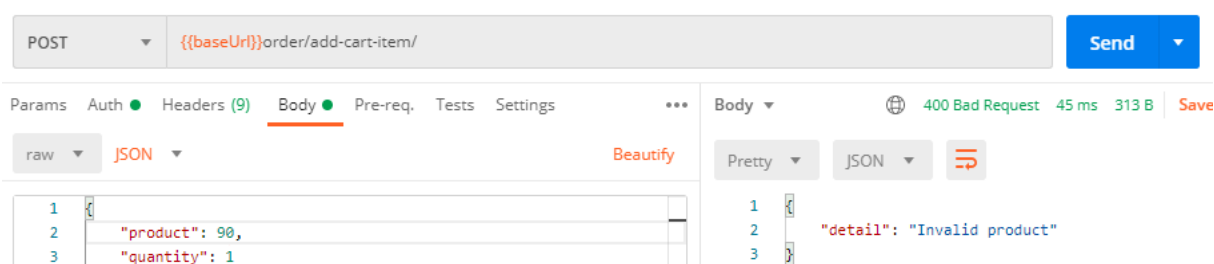


Hình 3.53. API Add Item to Cart: Ok

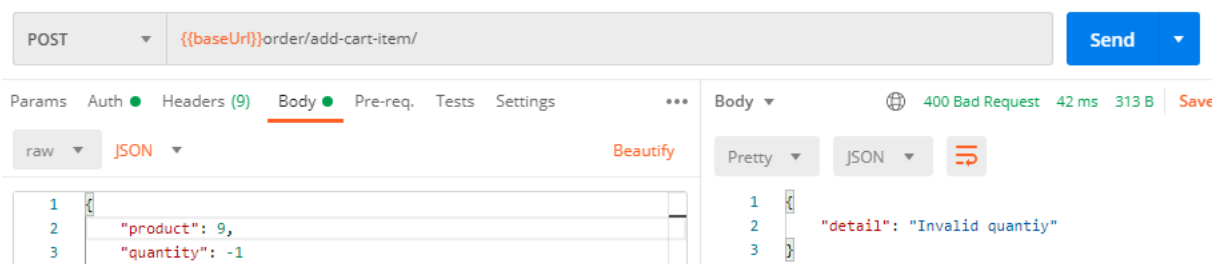
API sẽ báo lỗi khi dữ liệu gửi lên bị thiếu, mã sản phẩm không hợp lệ (Mã không tồn tại hoặc không phải số) hoặc số lượng không hợp lệ (Số lượng bé hơn 1 hoặc không phải là số).



Hình 3.54. API Add Item to Cart: Sai kiểu dữ liệu

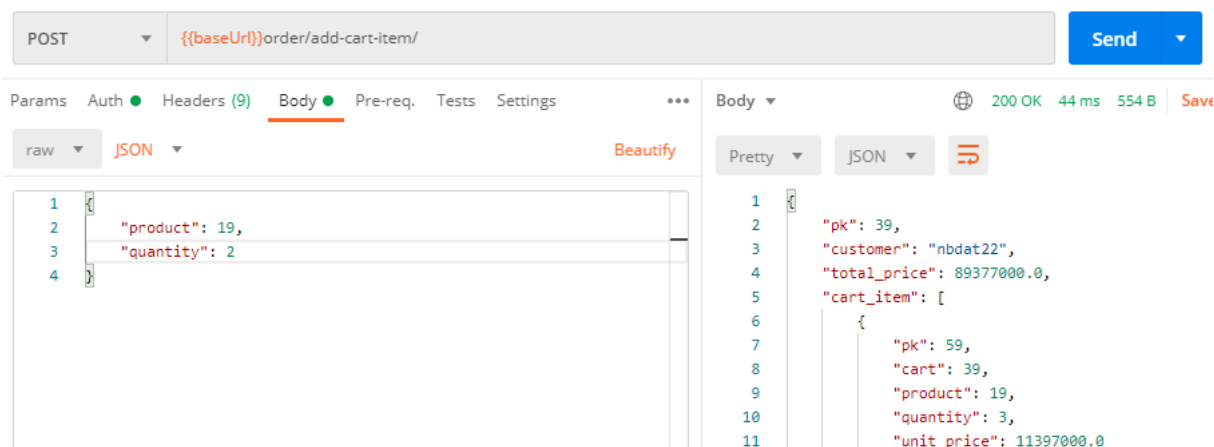


Hình 3.55. API Add Item to Cart: Mã sản phẩm không tồn tại

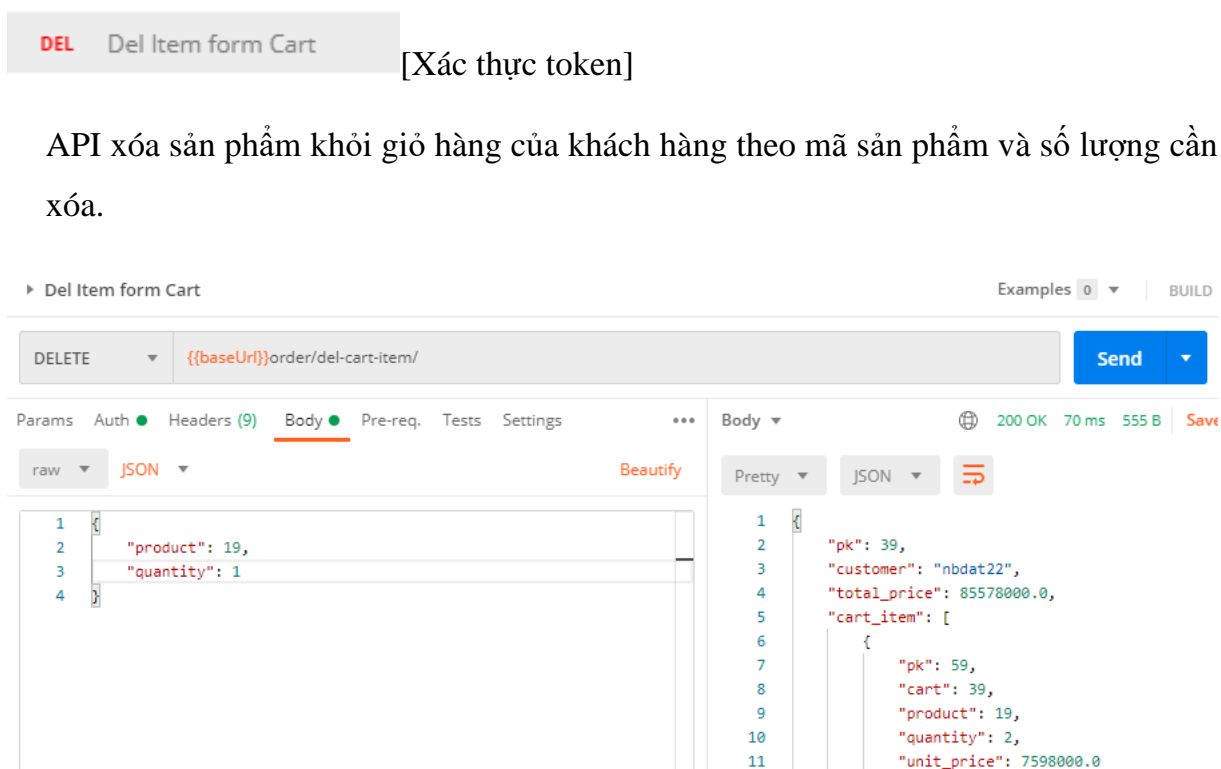


Hình 3.56. API Add Item to Cart: Số lượng không hợp lệ

API sẽ tự động tăng số lượng nếu sản phẩm vừa thêm vào đã tồn tại trong giỏ.



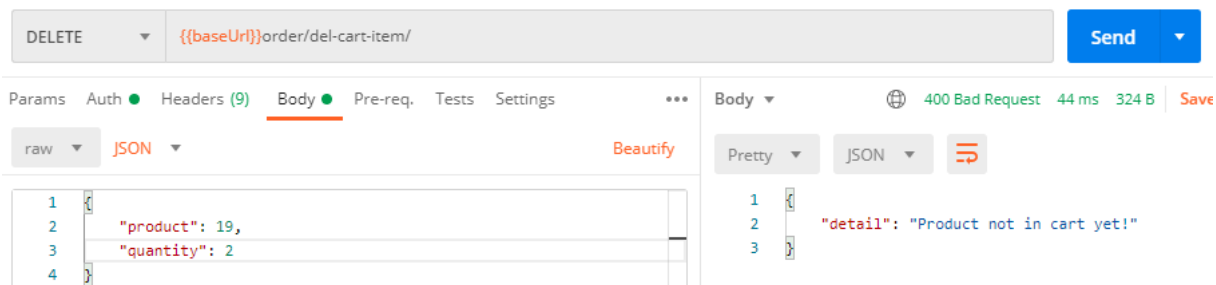
Hình 3.57. API Add Item to Cart: Số lượng tăng tự động



Hình 3.58. API Del Item form Cart: Ok

API sẽ báo lỗi khi dữ liệu gửi lên bị thiếu, mã sản phẩm không hợp lệ (Mã không tồn tại hoặc không phải số) hoặc số lượng không hợp lệ (Số lượng bé hơn 1 hoặc không phải là số). (Tương tự như API Add Item to Cart ở trên)

API sẽ báo lỗi nếu sản phẩm muốn xóa không có trong giỏ.

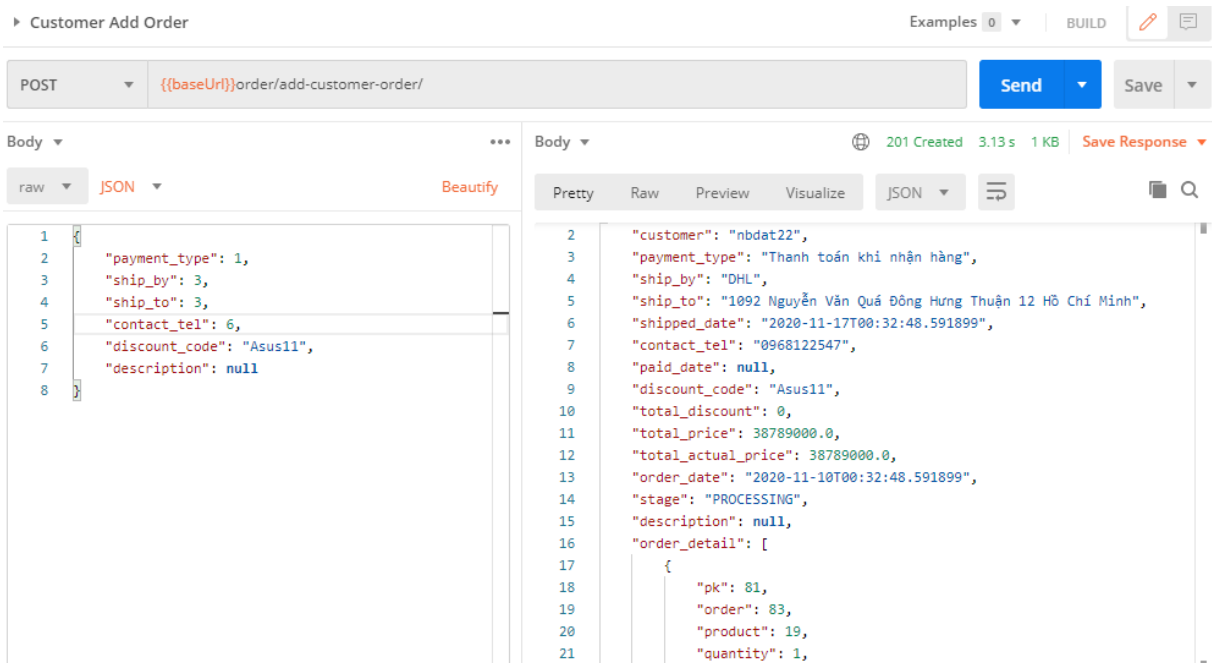


Hình 3.59. API Del Item form Cart: Sản phẩm không có trong giỏ

POST Customer Add Order

[Xác thực token]

API dùng để tạo đơn đặt hàng cho khách hàng và gửi mail thông báo đặt hàng thành công.



Hình 3.60. API Customer Add Order: Ok

DATJ: Order successful! Your ORDER ID: 83 ➡ [Hộp thư đến x](#)



datzach31@gmail.com

tới tôi ▾

Order information:

Customer: nbdat22
Payment type: Thanh toán khi nhận hàng
Ship service: DHL
Ship to: 1092 Nguyễn Văn Quá Đồng Hưng Thuận 12 Hồ Chí Minh
Contact tel: 0968122547
Paid date: None
Discount code: Asus11
Total discount: 0
Total price: 38789000.0
Total actual price: 38789000.0
Order date: 2020-11-10 00:32:48.591899
Stage: PROCESSING

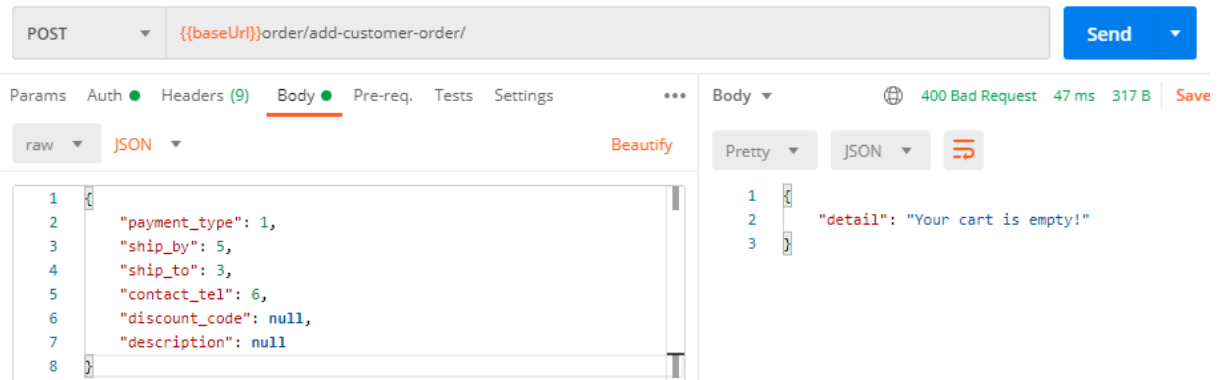
Order Detail:

Product: Bàn phím cơ Gaming Logitech G Pro X
Quantity: 1

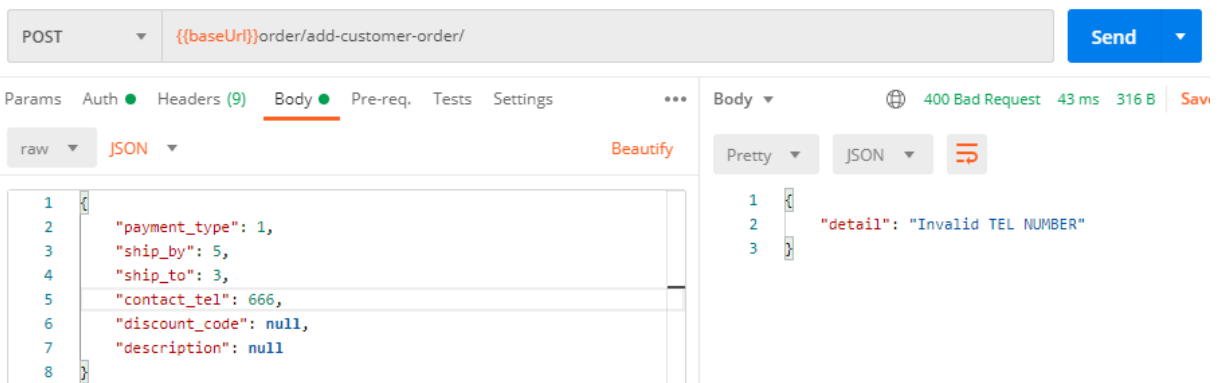
Hình 3.61. Mail thông báo khách hàng đặt hàng thành công

Khách hàng đặt hàng thành công khi:

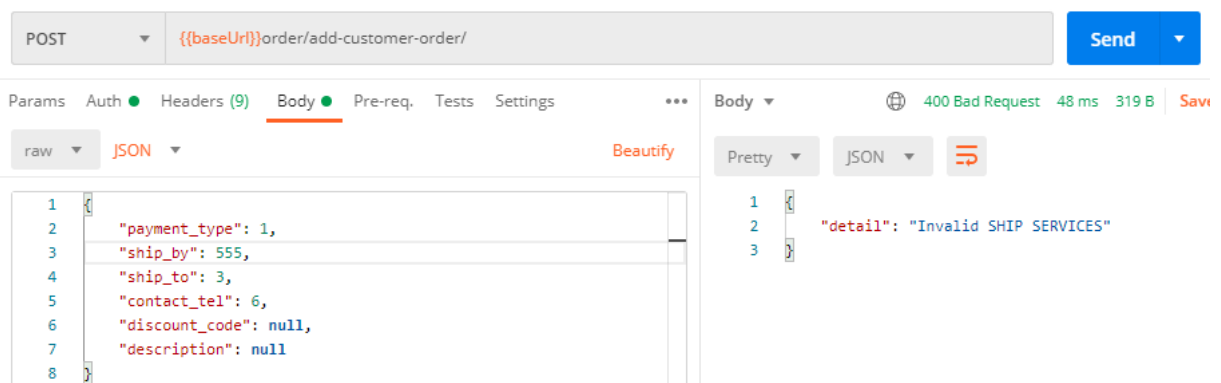
- Giỏ hàng không được phép rỗng.



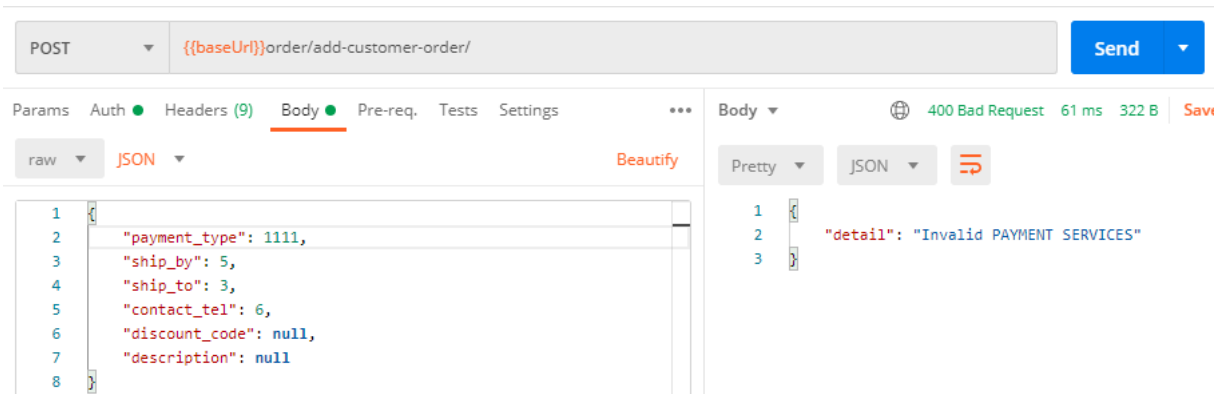
- Cung cấp đầy đủ thông tin để giao hàng (Địa chỉ giao hàng và điện thoại liên lạc) và thông tin giao hàng phải thuộc sở hữu của khách hàng đặt hàng.



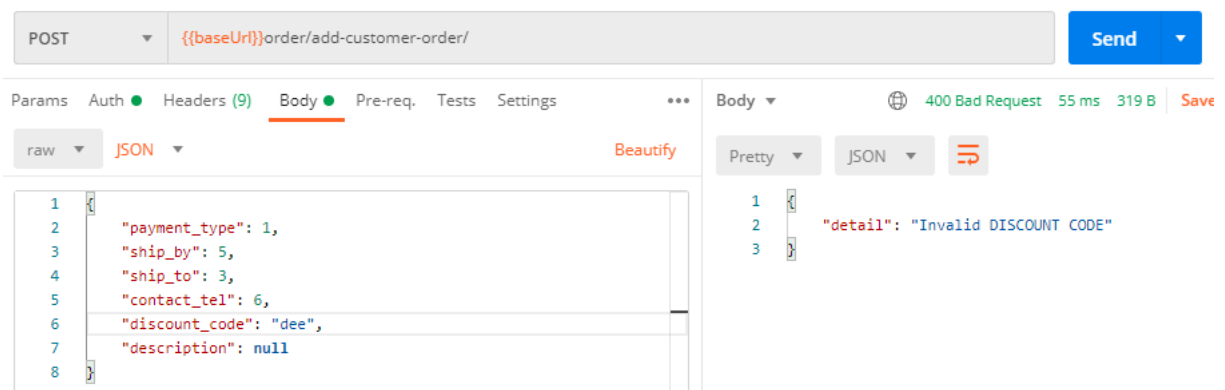
- Đã chọn được phương thức giao hàng mà hệ thống có hỗ trợ.



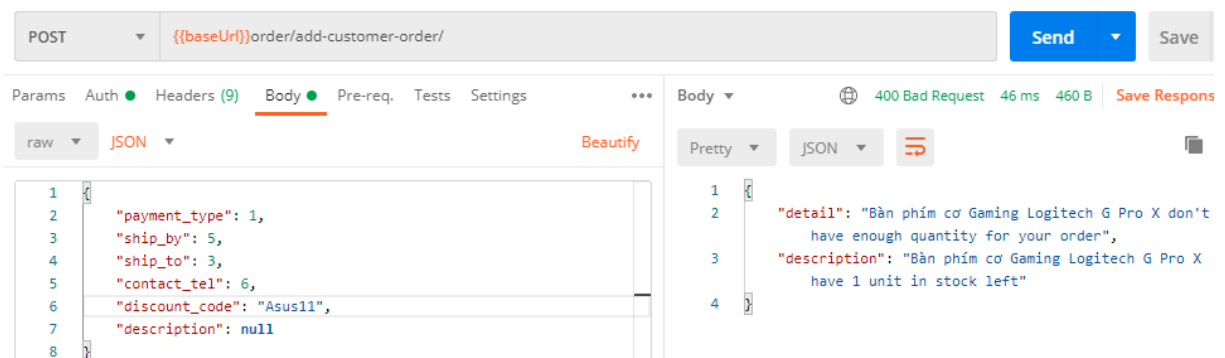
- Đã chọn được phương thức thanh toán mà hệ thống có hỗ trợ và phải hoàn tất thanh toán với các phương thức thanh toán trực tuyến.



- Nếu có mã giảm giá, phải nhập đúng mã giảm giá.



- Số lượng hàng tồn trong kho phải còn đủ số lượng để đáp yêu cầu đặt hàng.

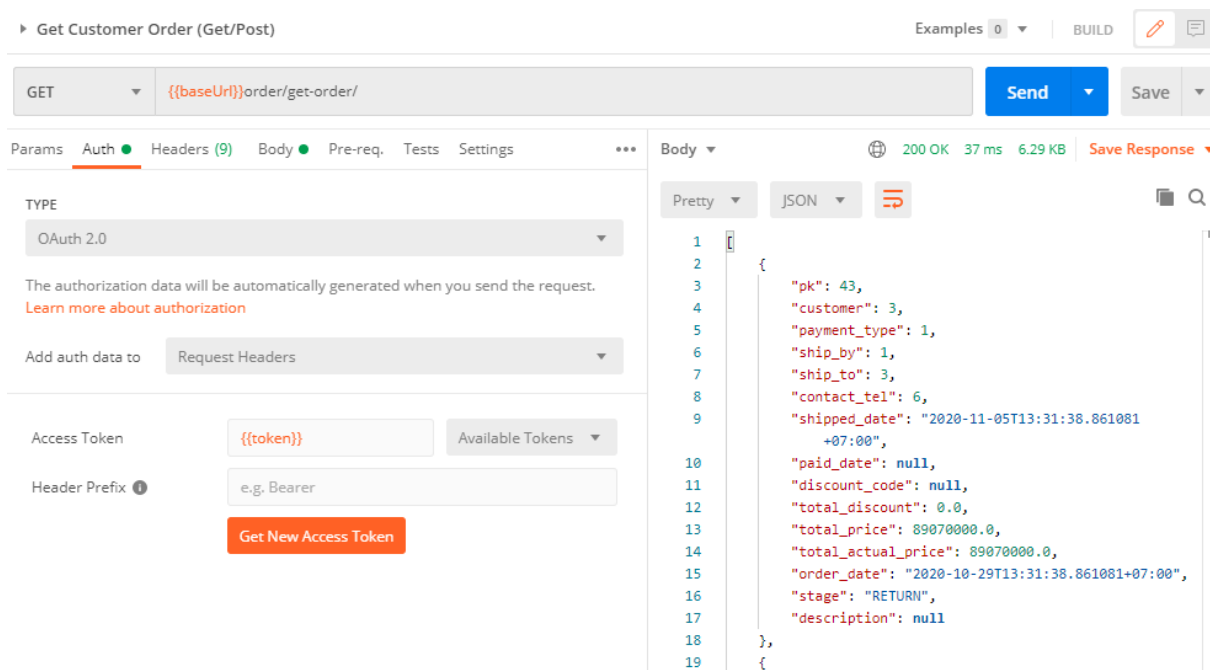


Sau khi kiểm tra các tính hợp lệ trên, API đã có thể tạo đơn hàng vào kho khi dữ liệu và các công việc để tạo đơn bao gồm:

- Tạo đơn hàng mới tại bảng Order với trạng thái PROCESSING cùng thông tin giảm giá nếu có.
- Đổ dữ liệu sản phẩm từ bảng CartItem sang bảng OrderDetail cùng với thông tin của mã giảm giá nếu có.
- Cập nhật số lượng unit_in_order tại bảng Product.
- Xóa dữ liệu tại bảng Cart và CartItem của khách hàng.
- Cập nhật và lưu lại tất cả thông tin.
- Gửi mail thông báo việc đặt hàng thành công cho khách hàng.

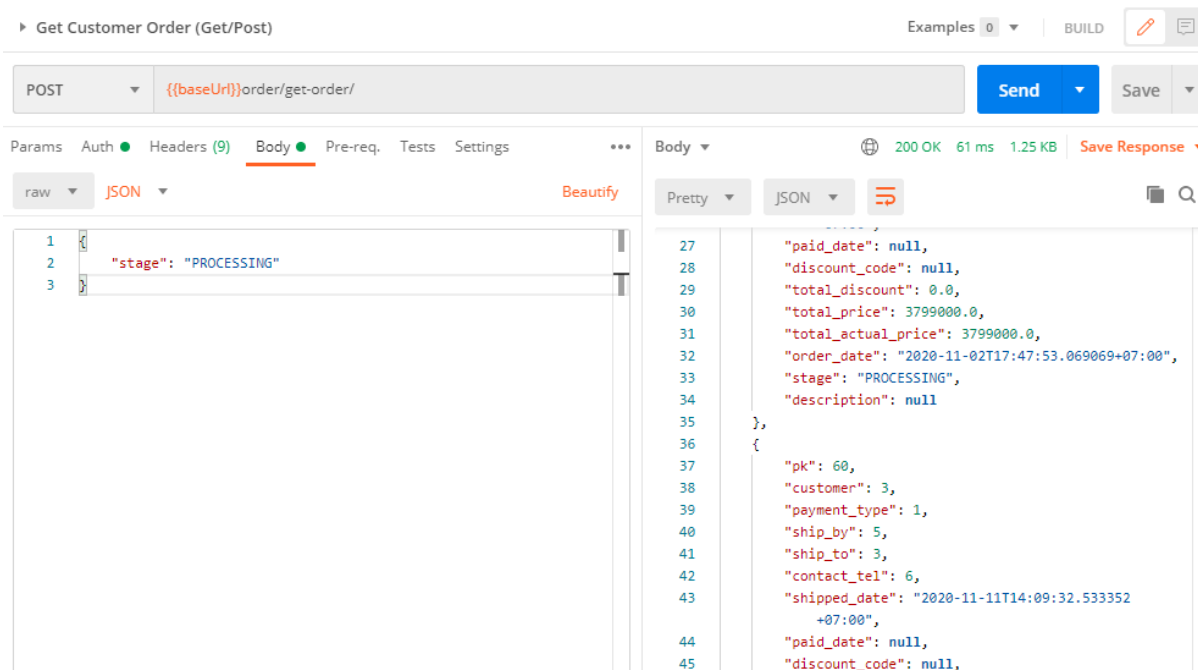
GET Get Customer Order (Get/Post) [Xác thực token]

API khi gọi bằng phương GET sẽ lấy danh sách tất cả đơn hàng của khách hàng.



Hình 3.62. API Get Customer Order (GET): Ok

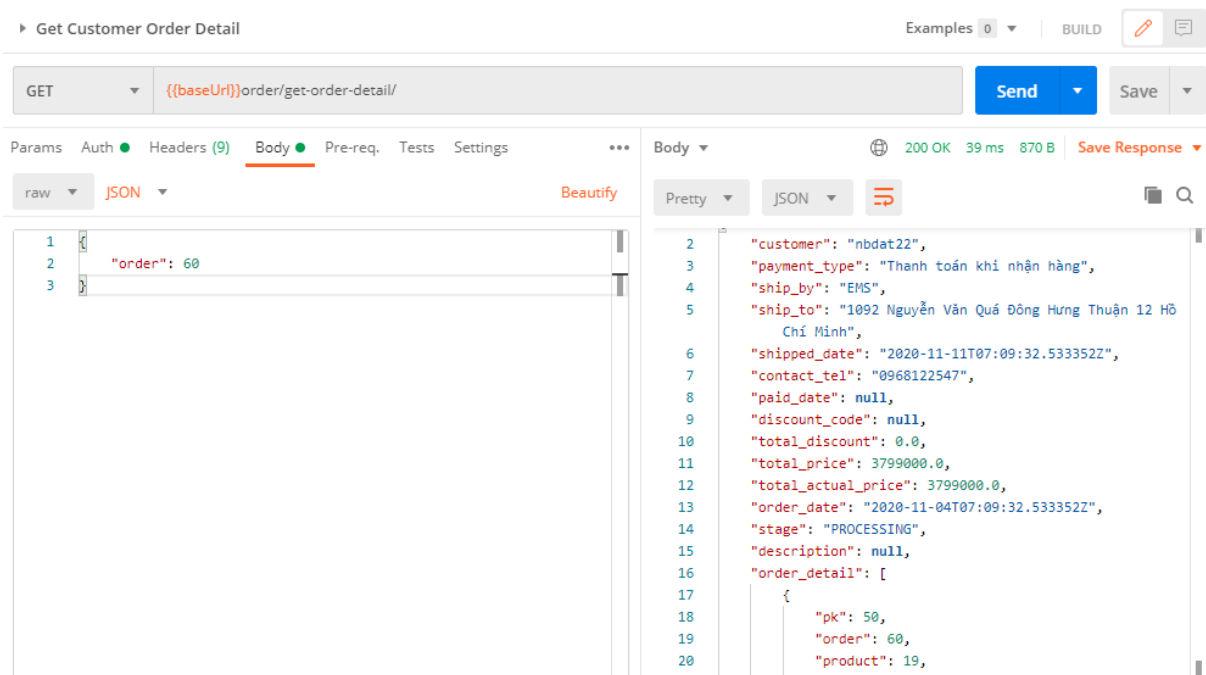
API khi gọi bằng phương thức POST sẽ lấy danh sách đơn hàng của khách hàng theo trạng thái được gửi đi.



Hình 3.63. API Get Customer Order (POST): DS đơn hàng có trạng thái PROCESSING

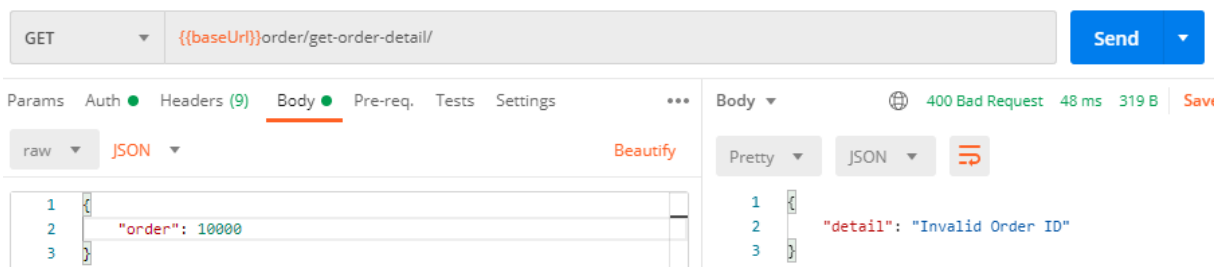
GET Get Customer Order Detail [Xác thực token]

API dùng để lấy thông tin chi tiết của đơn hàng theo mã đơn hàng.

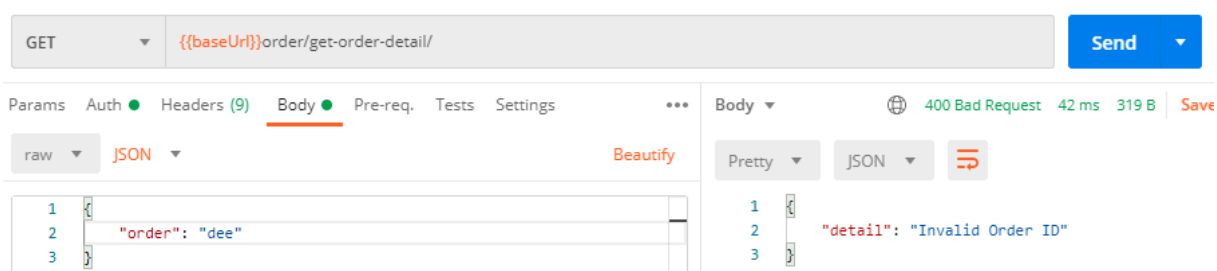


Hình 3.64. API Get Customer Order Detail: Ok

API sẽ báo lỗi khi nhận mã đơn hàng không hợp lệ (Mã đơn hàng không tồn hoặc mã không phải là số).



Hình 3.65. API Get Customer Order Detail: Mã đơn hàng không tồn tại

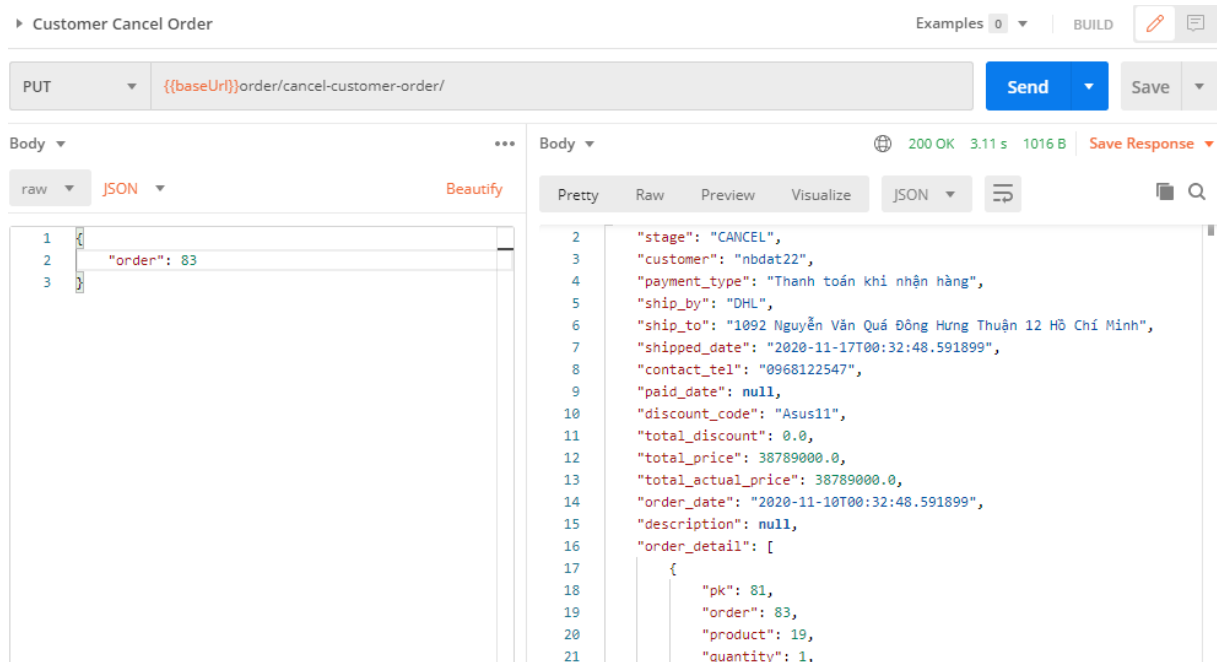


Hình 3.66. API Get Customer Order Detail: Mã đơn hàng không phải số


PUT Customer Cancel Order

[Xác thực token]

API hỗ trợ khách hàng hủy đơn đặt hàng.



Hình 3.67. API Customer Cancel Order: Ok

DATJ: Cancel ORDER successful! Your ORDER ID: 83  Hộp thư đến x



datzach31@gmail.com

tôi tôi ▾

 Tiếng Anh ▾ > Tiếng Việt ▾ Dịch thư

Order Detail:

Customer: nbdat22
Payment type: Thanh toán khi nhận hàng
Ship service: DHL
Ship to: 1092 Nguyễn Văn Quá Đồng Hưng Thuận 12 Hồ Chí Minh
Contact tel: 0968122547
Paid date: None
Discount code: Asus11
Total discount: 0.0
Total price: 38789000.0
Total actual price: 38789000.0
Order date: 2020-11-10 00:32:48.591899
Stage: CANCEL

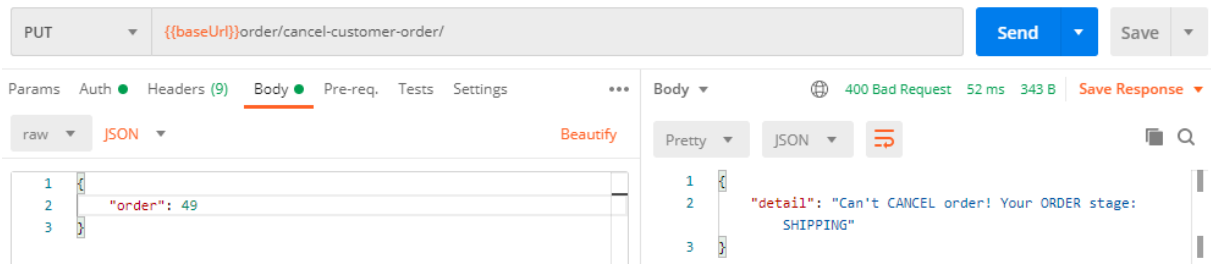
Order Detail:

Product: Bàn phím cơ Gaming Logitech G Pro X
Quantity: 1

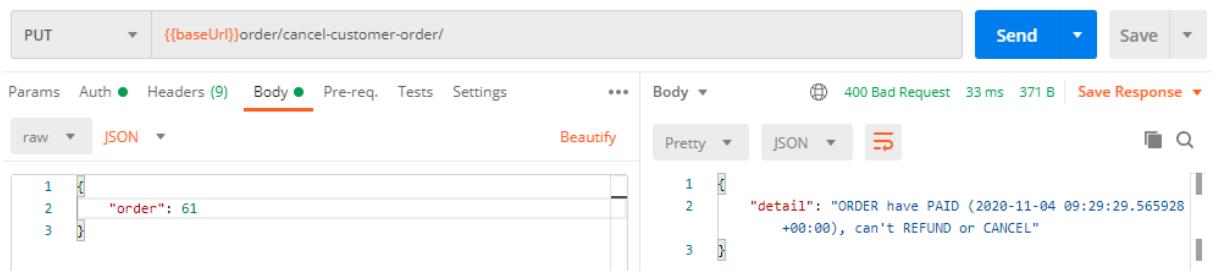
Hình 3.68. Mail thông báo KH hủy đơn thành công

Khách hàng chỉ được phép hủy đơn khi:

- Đơn hàng phải thuộc trạng thái PROCESSING (Đang xử lý).

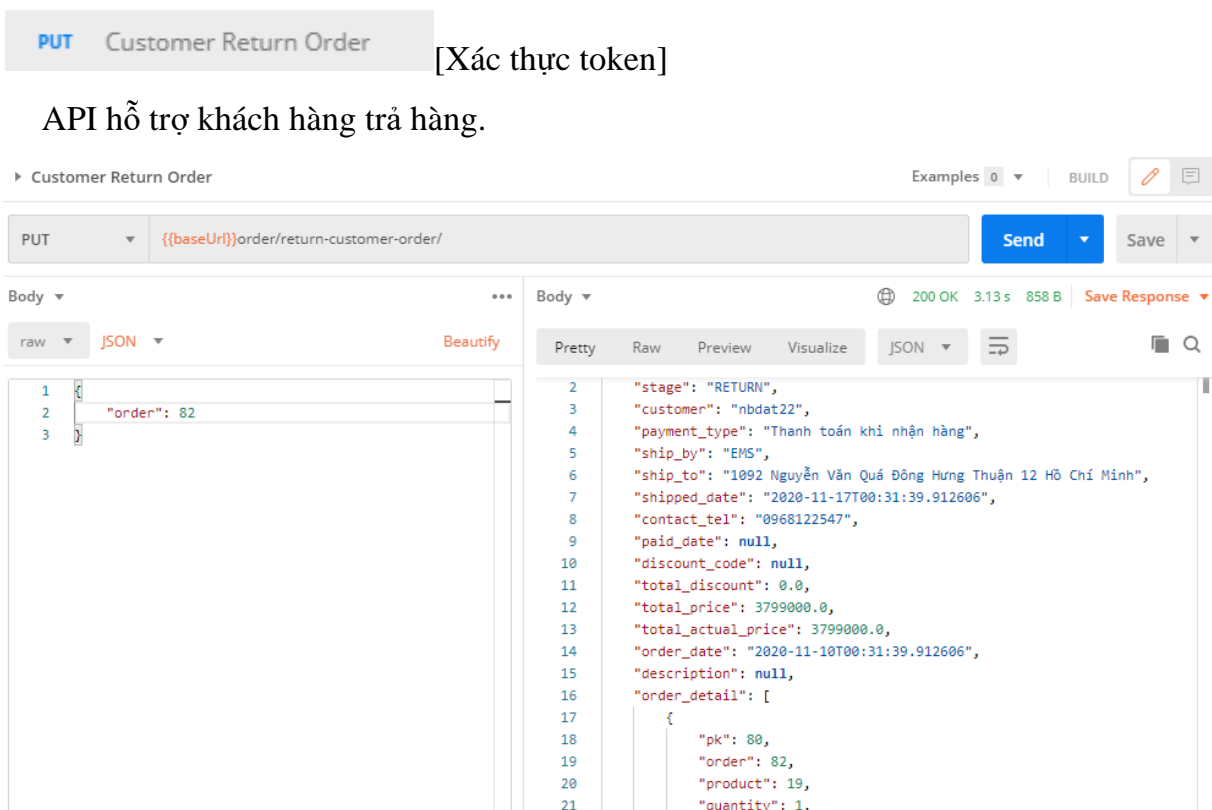


- Đơn hàng chưa được thanh toán.

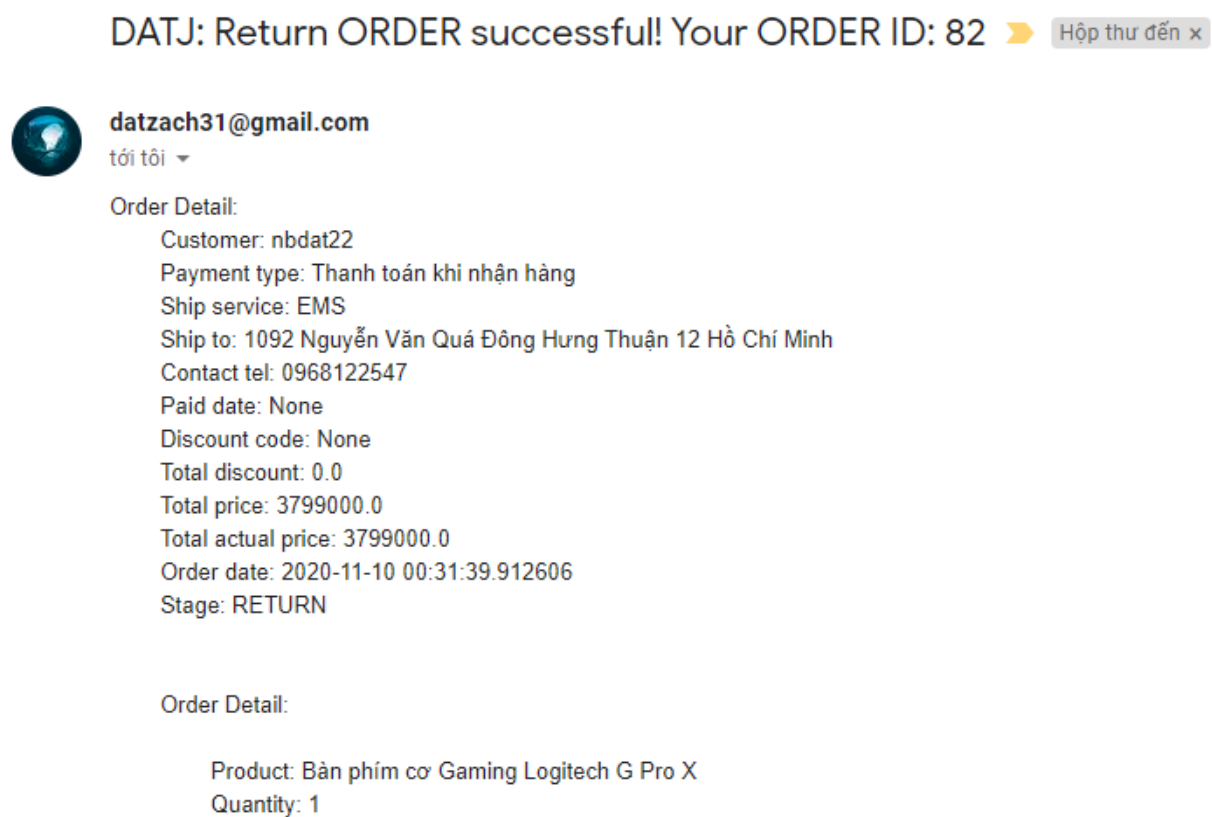


Sau khi kiểm tra đơn hàng được phép xóa, API sẽ thực thi tiếp các công việc sau để xóa đơn hàng:

- Đổi trạng thái đơn hàng từ PROCESSING sang CANCEL.
- Cập nhật lại số lượng unit_in_order tại bảng Product.
- Gửi mail thông báo việc hủy đơn thành công cho khách hàng.

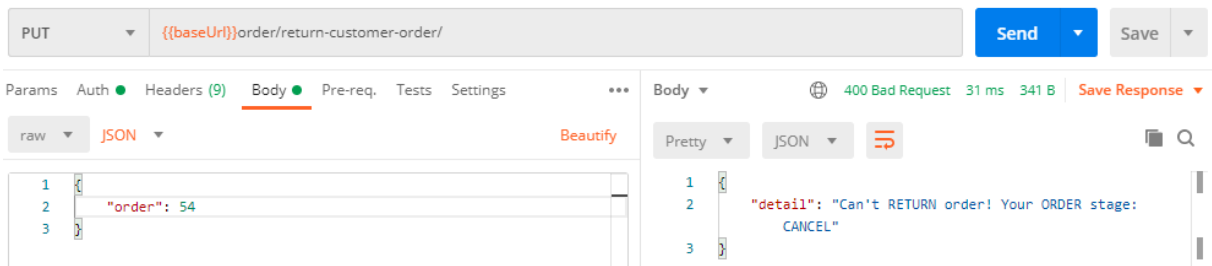


Hình 3.69. API Customer Return Order: Ok



Hình 3.70. Mail thông báo KH trả hàng thành công

Khách hàng chỉ được phép trả hàng khi đơn hàng ở trạng thái DONE (Đã hoàn tất).



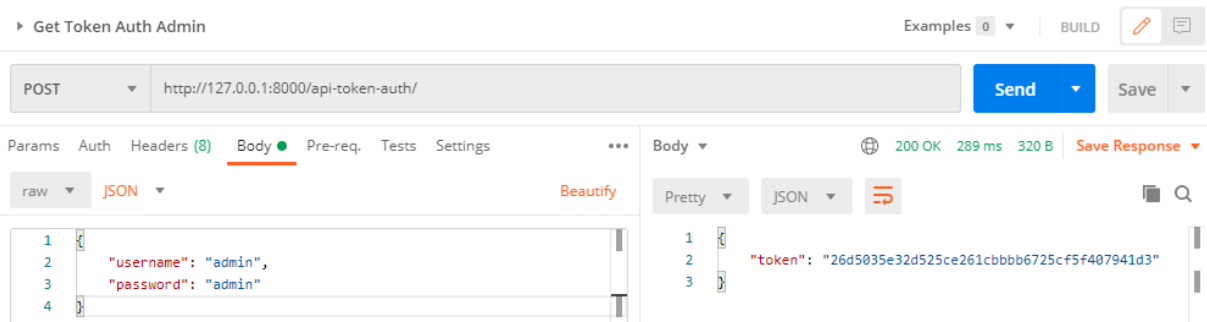
Sau khi API kiểm tra đơn hàng có thể trả về, API sẽ thực thi các công việc:

- Cập nhật trạng thái đơn hàng là RETURN và xóa thông tin ngày thanh toán.
- Cập nhật dữ liệu số lượng `unit_in_stock` tại bảng Product.
- Gửi mail thông báo việc trả hàng thành công cho khách hàng.

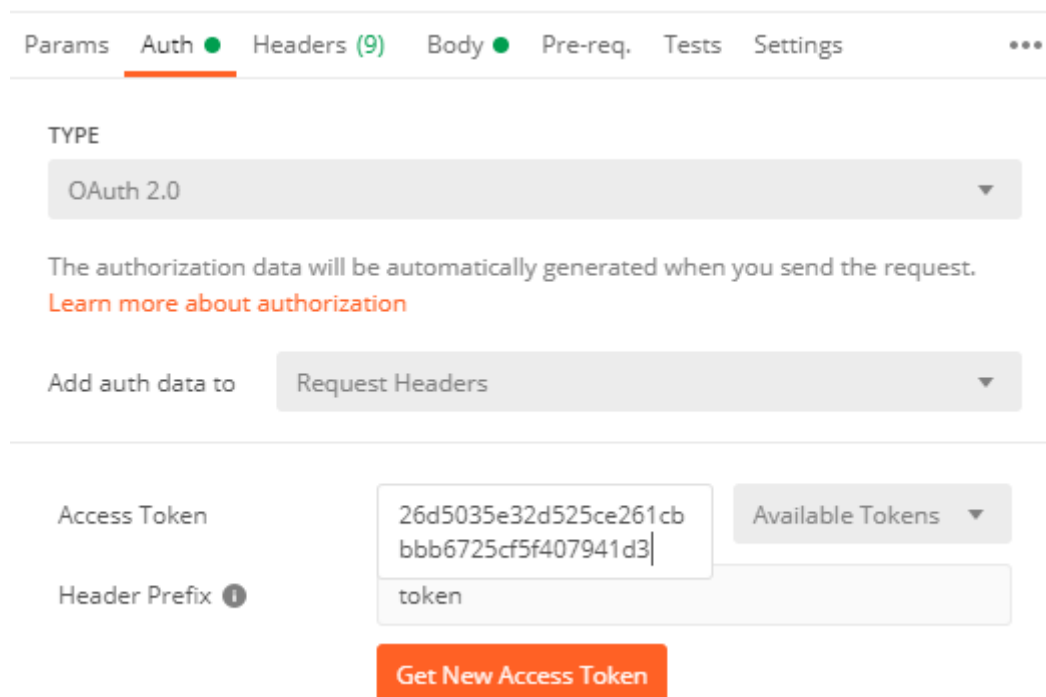
Các API tiếp theo dành cho nhân viên sử dụng hệ thống để cập nhật trạng thái của các đơn hàng và được xác thực token với thư viện

`rest_framework.authentication` của **Django**

Dùng API sau và đăng nhập với tài khoản nhân viên để tạo token:



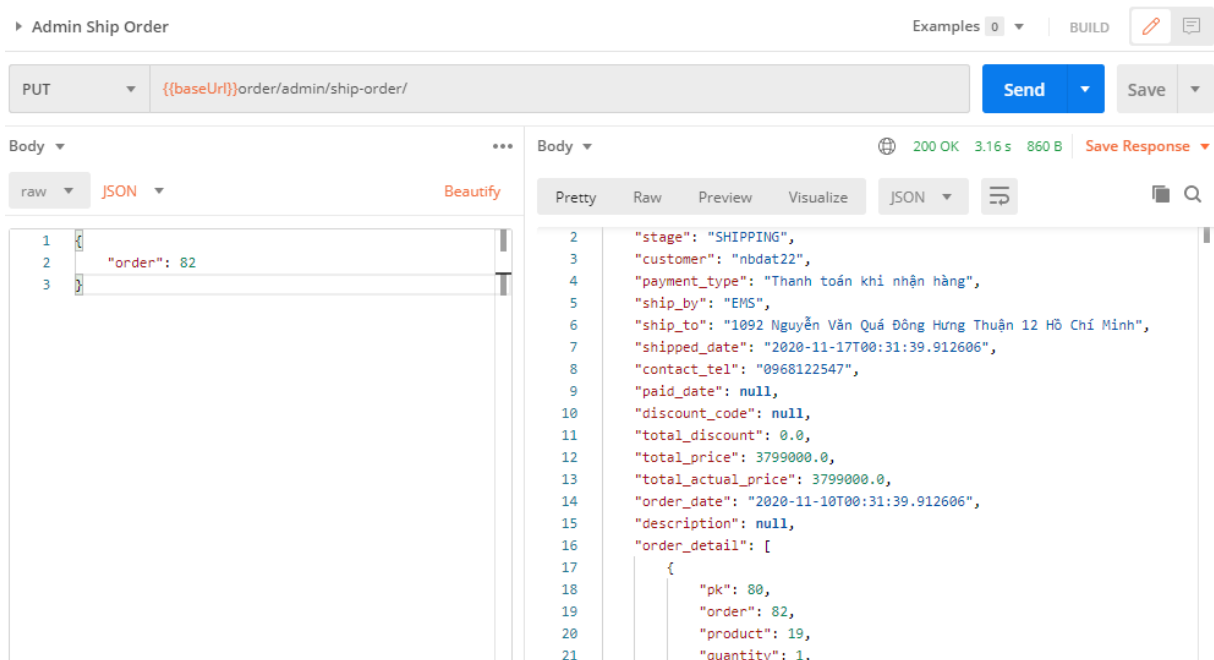
Dùng token đó để xác thực khi gọi API:



PUT Admin Ship Order

[Xác thực token nhân viên]

API giúp nhân viên cập nhật trạng thái đơn hàng thành SHIPPING.



Hình 3.71. API Admin Ship Order: Ok

DATJ: Your ORDER is SHIPPING! Your ORDER ID: 82

Hộp thư đến x



datzach31@gmail.com

tới tôi ▾

Order Detail:

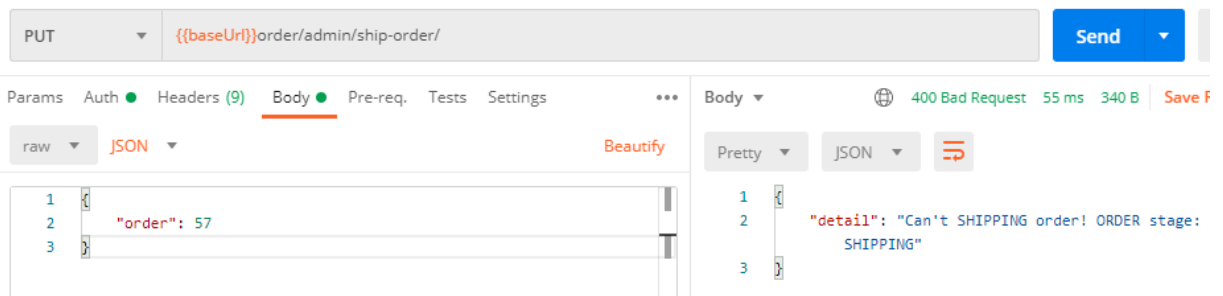
Customer: nbdat22
Payment type: Thanh toán khi nhận hàng
Ship service: EMS
Ship to: 1092 Nguyễn Văn Quá Đồng Hưng Thuận 12 Hồ Chí Minh
Contact tel: 0968122547
Paid date: None
Discount code: None
Total discount: 0.0
Total price: 3799000.0
Total actual price: 3799000.0
Order date: 2020-11-10 00:31:39.912606
Stage: SHIPPING

Order Detail:

Product: Bàn phím cơ Gaming Logitech G Pro X
Quantity: 1

Hình 3.72. Mail thông báo KH đơn hàng đang được vận chuyển đi

Đơn hàng chỉ được chuyển trạng thái giao hàng khi đơn hàng đang ở trạng thái PROCESSING.



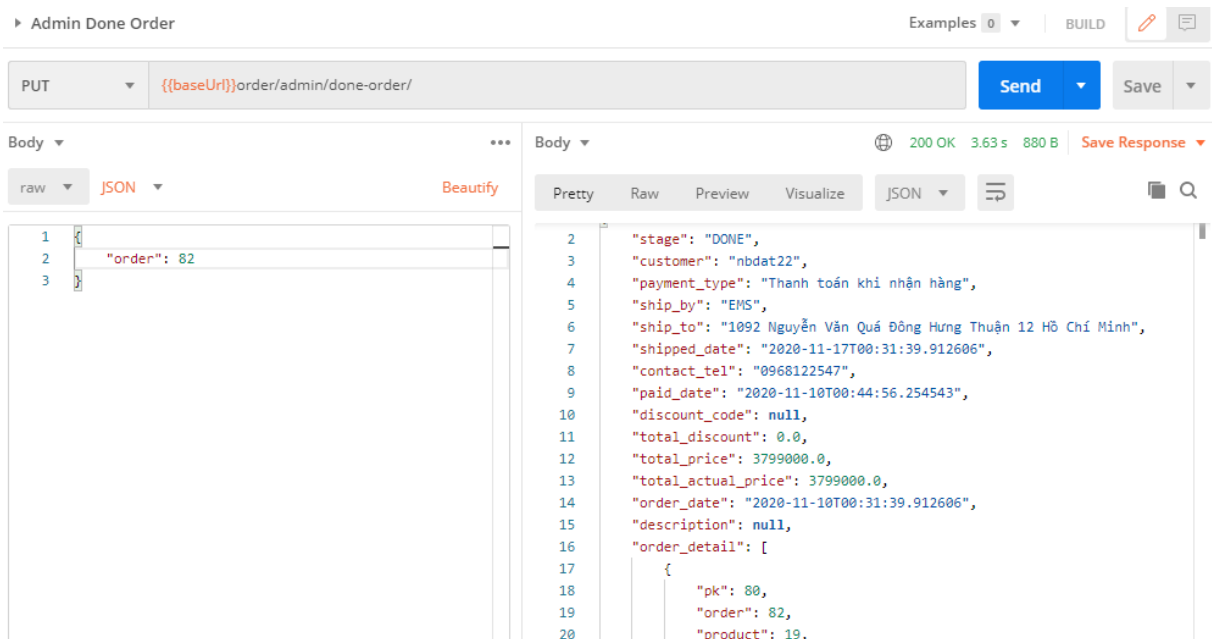
Sau khi API kiểm tra đơn hàng có thể giao, API sẽ thực thi các công việc:

- Cập nhật trạng thái đơn hàng từ PROCESSING sang SHIPPING.
- Cập nhật dữ liệu số lượng unit_in_order và unit_in_stock tại bảng Product.
- Gửi mail thông báo đơn hàng đang được giao cho khách hàng.

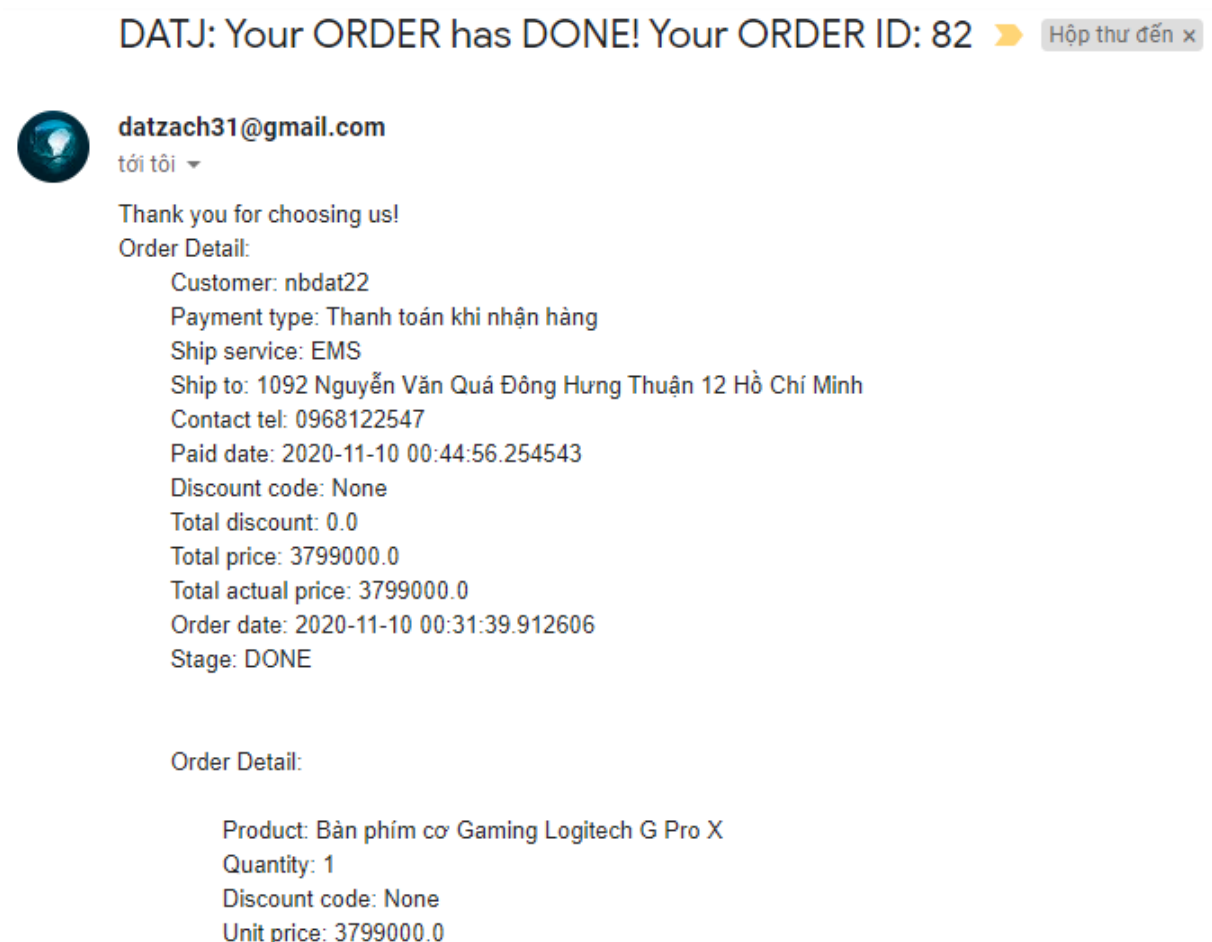
PUT Admin Done Order

[Xác thực token nhân viên]

API giúp nhân viên cập nhật trạng thái đơn hàng thành DONE.

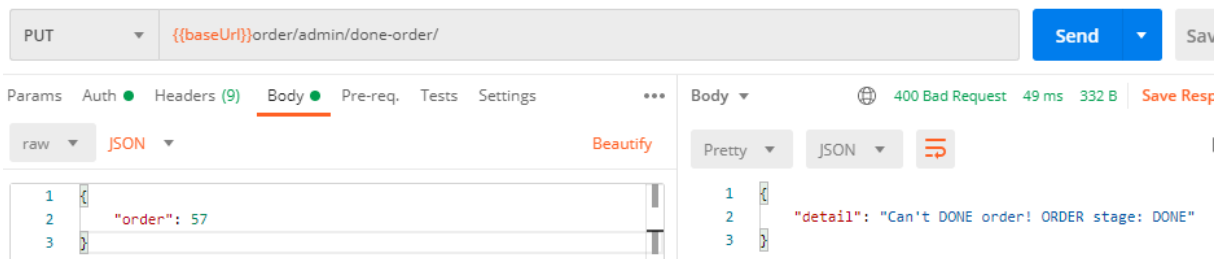


Hình 3.73. API Admin Done Order: Ok



Hình 3.74. Mail thông báo KH đơn hàng đã hoàn tất

Đơn hàng chỉ được chuyển trạng thái hoàn tất khi đơn hàng đang ở trạng thái SHIPPING.



Sau khi API kiểm tra đơn hàng có thể hoàn tất, API sẽ thực thi các công việc:

- Chuyển trạng thái đơn hàng từ SHIPPING sang DONE.
- Cập nhật ngày thanh toán hóa đơn nếu khách hàng thanh toán khi nhận hàng.
- Gửi mail thông báo đơn hàng hoàn tất cho khách hàng.

Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Kết quả đạt được

Sau khi hoàn thiện đồ án, em đã có được một hệ thống API được chia làm 3 nhóm API chính hỗ trợ khách hàng xem, chọn lựa hàng hóa (API Product), giúp khách hàng quản lý các thông tin tài khoản tại hệ thống (API Customer) và theo dõi các đơn hàng, đặt hàng, trả hàng và hủy đơn theo như cầu của họ. Cùng với đó là một web quản trị với các chức năng cơ bản dành cho chủ cửa hàng và các nhân viên với các thao tác quản lý hàng hóa, khuyến mãi, nhân sự và xem, kiểm kê các đơn hàng. Với các API cơ bản đó đã có thể ứng dụng vào một web bán hàng nhỏ, xử lý được các thao tác chọn mua hàng cho khách hàng. Và với cách thiết kế dữ liệu cho Product theo mô hình EAV em có thể dễ dàng mở rộng để bán được các mặt hàng khác về sau. Các API cũng được bảo mật bằng phương pháp OAuth 2, các API chỉ có thể gọi và thực thi được khi khách hàng đã đăng nhập hệ thống thành công và được cấp một token của riêng mình để thao tác với các API, chức năng khác của hệ thống. Cùng với tính năng gửi mail giúp khách hàng phục hồi mật khẩu dễ dàng và hệ thống thông báo kịp thời cho khách hàng về tình trạng các đơn hàng của mình.

Sau khi thực hiện đồ án và hoàn thiện phần báo cáo này, em đã có thêm được nhiều kiến thức chuyên môn mới cùng các kỹ năng tìm hiểu, nghiên cứu khi gặp các vấn đề mới, giải quyết vấn đề độc lập và quản lý phân chia thời gian, sắp xếp các công việc cần làm để thực hiện một dự án.

4.2. Hạn chế và hướng phát triển

Hệ thống của em còn nhiều mặt hạn chế bao gồm:

- Cách thiết kế dữ liệu của nhóm bảng Order (Order, OrderDetail, PaymentService, ShipService) làm cho việc trả hàng, giao các món hàng riêng lẻ trong đơn hàng chính khó thực thi được.
- Khó thực hiện truy vấn dữ liệu một cách nhanh chóng và tối ưu (VD: Truy vấn tất cả các Category con của một Category cha).
- Chưa có các chức năng thống kê đơn hàng, hàng hóa.
- Khó tích hợp được các dịch vụ bên thứ ba.

Và còn nhiều thiết sót khác em chưa thể nhận ra do trình độ còn hạn chế của bản thân. Hướng phát triển của hệ thống em sẽ khắc phục các hạn chế trên để hệ thống có thể dễ dàng mở rộng, tích hợp các tính năng mới và các cửa hàng loại hình mua bán online khác nhau có thể tích hợp, sử dụng hệ thống một cách nhanh chóng, dễ dàng.

TÀI LIỆU THAM KHẢO

- [1] Tìm hiểu về mô hình EAV: <https://viblo.asia/p/tim-hieu-ve-entity-attribute-value-pattern-eav-structural-pattern-Eb85okV452G> (22/10/2020)
- [2] Định nghĩa code-first: <https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx> (03/11/2020)
- [3] Django admin: <https://docs.djangoproject.com/en/3.1/ref/contrib/admin/> (21/10/2020)
- [4] Django Rest Framework: <https://www.django-rest-framework.org/tutorial/quickstart/> (29/10/2020)
- [5] Django Authentication: <https://www.django-rest-framework.org/api-guide/authentication/> (02/11/2020)
- [6] Django Mail: <https://docs.djangoproject.com/en/3.1/topics/email/> (03/11/2020)
- [7] Rest API: <https://medium.com/eway/nguy%C3%AAn-t%E1%BA%AFC-thi%E1%BA%BFt-k%E1%BA%BF-rest-api-23add16968d7> (04/11/2020)
- [8] Định nghĩa API: <https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/> (04/11/2020)
- [9] Luận văn của Roy Fielding: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (07/11/2020)

---HẾT---