**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

---🙡 🕮 🙣---



**NGUYỄN BÁ ĐẠT**

**HỆ THỐNG WEBSITE BÁN HÀNG ONLINE**

**Mã số sinh viên: 1751010022**

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Giảng viên hướng dẫn: DƯƠNG HỮU THÀNH**

**TP.HỒ CHÍ MINH, 2020**

# LỜI CẢM ƠN

# NHẬN XÉT GIÁO VIÊN HƯỚNG DẪN

# MỤC LỤC

[**LỜI CẢM ƠN 1**](#_Toc55302830)

[**NHẬN XÉT GIÁO VIÊN HƯỚNG DẪN 2**](#_Toc55302831)

[**MỞ ĐẦU 4**](#_Toc55302832)

[**Chương 1. Tổng quan 5**](#_Toc55302833)

[**Chương 2. Cơ sở lý thuyết 5**](#_Toc55302834)

[**2.1. Framework Django 5**](#_Toc55302835)

[**2.2. MySQL Database 6**](#_Toc55302836)

[**Chương 3. Nội dung thực hiện 6**](#_Toc55302837)

[**3.1. Xác định yêu cầu hệ thống 6**](#_Toc55302838)

[**3.2. Thiết kế cơ sở dữ liệu 8**](#_Toc55302839)

[**3.3. Xây dựng trang quản trị (Admin) 12**](#_Toc55302840)

[**3.1.1. Giai đoạn 1: 12**](#_Toc55302841)

[**3.4. Tạo các API theo yêu cầu hệ thống 13**](#_Toc55302842)

[**Chương 4. Kết luận 13**](#_Toc55302843)

[**TÀI LIỆU THAM KHẢO 14**](#_Toc55302844)

# MỞ ĐẦU

# Chương 1. Tổng quan

Hệ thống website bán các thiết bị phần cứng, linh kiện cho máy tính và các đồ dùng công nghệ online và có thể mở rộng để bán được nhiều mặt hàng khác.

Website sử dụng framework Django của ngôn ngữ Python để xây dựng cấu trúc của web, tạo các API và trang admin cơ bản. Để lưu trữ dữ liệu, hệ thống sử dụng hệ quản trị cơ sở dữ liệu MySQL. Website chủ yếu phục vụ cho 3 đối tượng chính là khách hàng, nhân viên và quản trị viên.

# Chương 2. Cơ sở lý thuyết

## 2.1. Framework Django

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 thế. 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 dễ dàng tập trung vào việc phát triển ứng dụng web của mình mà không phải xây dựng lại nền tảng đã có của 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.

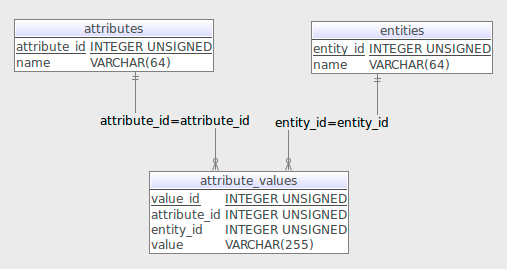
* *Đảm bảo tính bảo mật:* Django rất coi trọng việc bảo mật và giúp các nhà phát triển tránh được nhiều lỗi bảo mật phổ biến, chẳng hạn như SQL Injection, giả mạo yêu cầu cross-site. Hệ thống User Authentication của Django cung cấp cách bảo mật tốt nhất để quản lý tài khoản người dùng và mật khẩu của 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 dụng đến mạng xã hội và cả nền tảng khoa học máy tính.

## 2.2. MySQL Database

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 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.

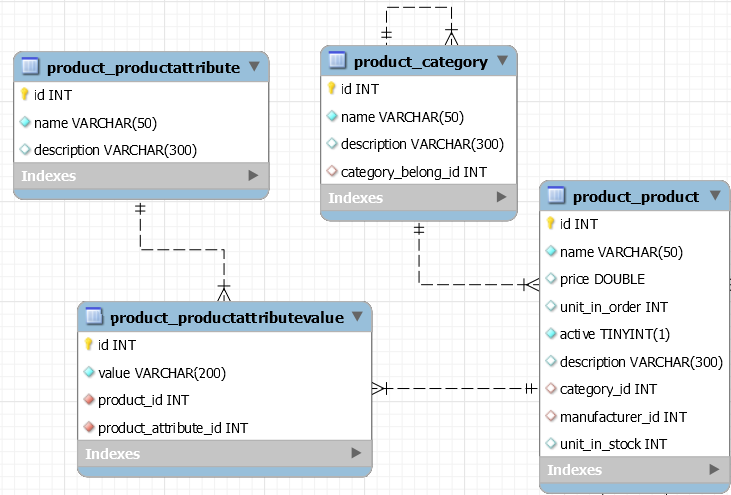
## 2.3. EAV Pattern

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 cở sở dữ liệu trong Magento.

* 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

Ta có:

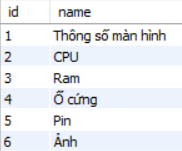
* Entity là bảng product\_product.
* Attribute là bảng product\_productattribute.
* Value là bảng product\_productattributevalue.

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.

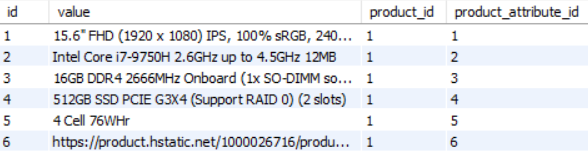
Product:



ProductAttribute:



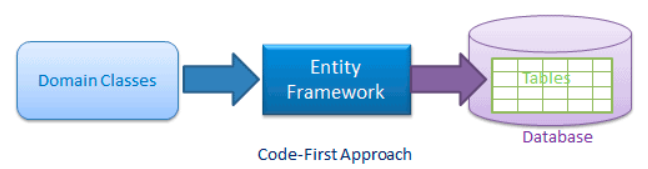
ProductAttributeValue:



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. Code-first

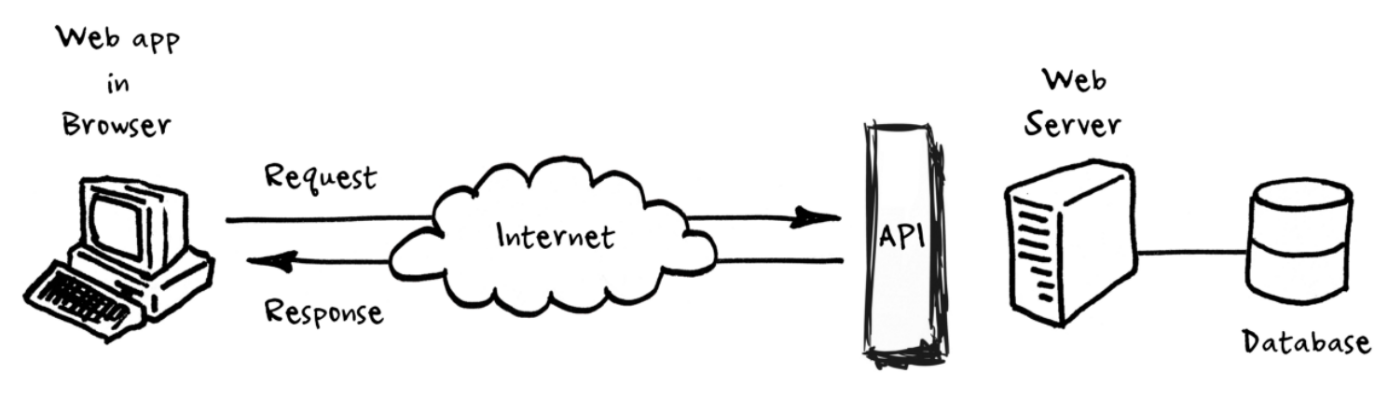
Code-first là một cách tạo dữ liệu bằng việc tập trung vào code, tạo các class của các thực thể cho hệ thống trước rồi từ đó entity framework sẽ tạo ra cơ sở dữ liệu từ code đó.



Với cách tiếp cận đó, ta có thể dễ dàng thay đổi thiết kế cơ sở dữ liệu và phát triển hệ thống một cách nhanh chóng.

## 2.5. 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.



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.

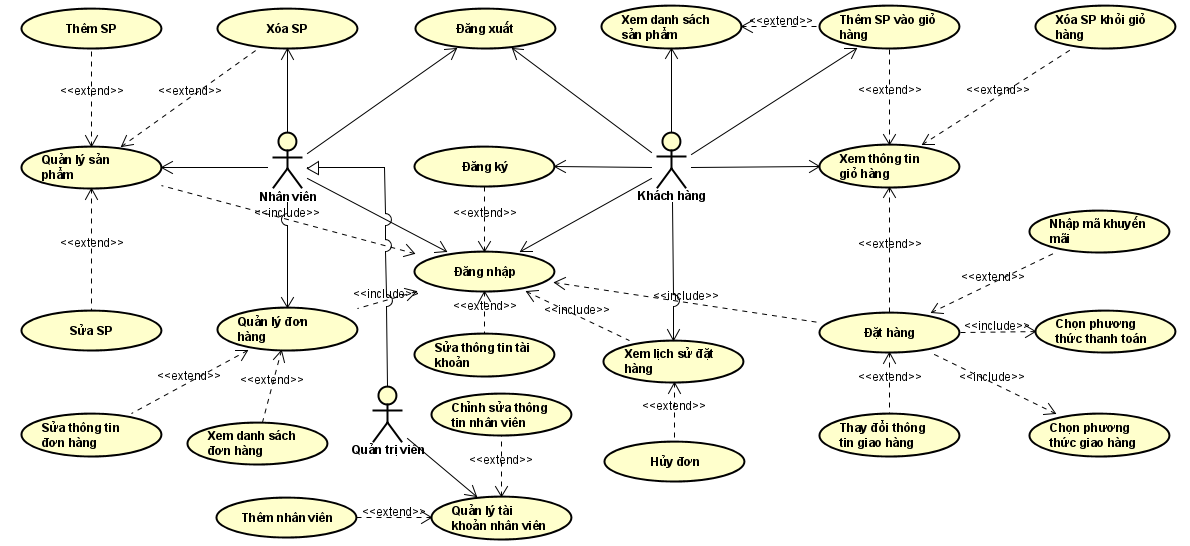
## 2.6. RESTful API

## 2.7. OAuth 2

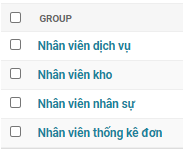
## 2.8. OTP and Mail

# Chương 3. Nội dung thực hiện và kết quả

## 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 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 chưc năng điều hành, quản lý. Nhân viên được chia ra nhiều loại nhân viên để quản lý từng thành phần của hệ thống như:

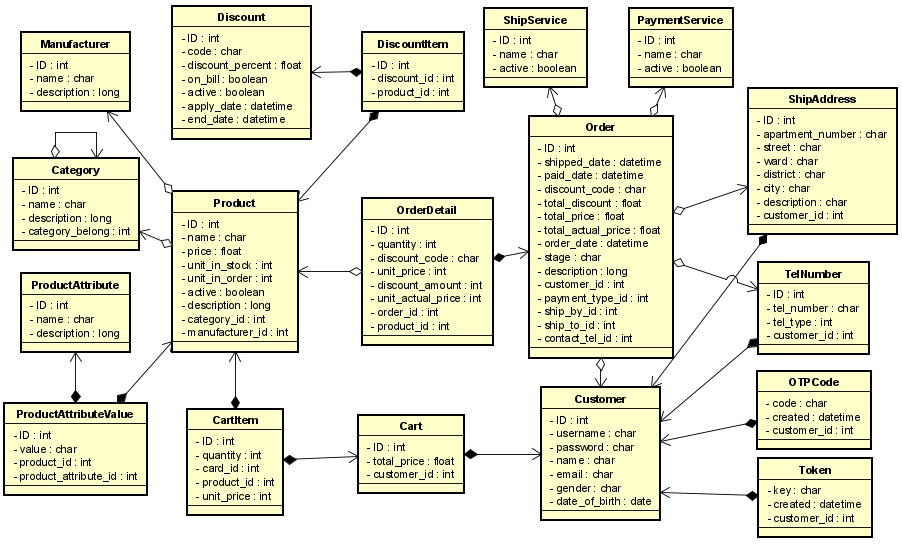
Hình 2. Phân quyền nhân viên

* 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:

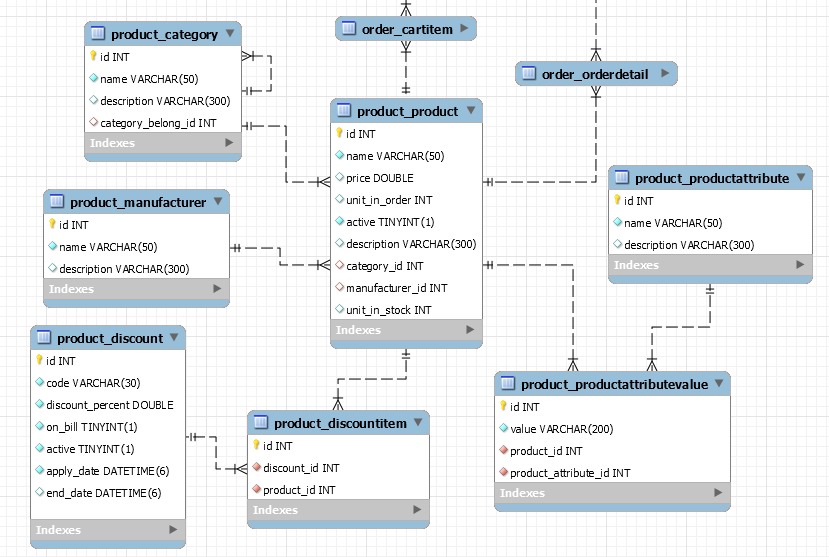
* Đăng ký, đăng nhập, đăng xuất, chỉnh sửa thông tin tài khoản.
* Xem danh sách các sản phẩm.
* Thêm sản phẩm vào giỏ hàng hoặc xóa sản phẩm khỏi giỏ hàng.
* Tiến hành đặt hành tạo đơn hàng, trong lúc tạo đơn hàng phải thực hiện các công việc bao gồm chọn phương thức thanh toán và cung cấp thông tin 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. Sơ đồ class

Từ sơ đồ class trên em phân dữ liệu thành 3 nhóm chính gồm: Product, Order, Customer.

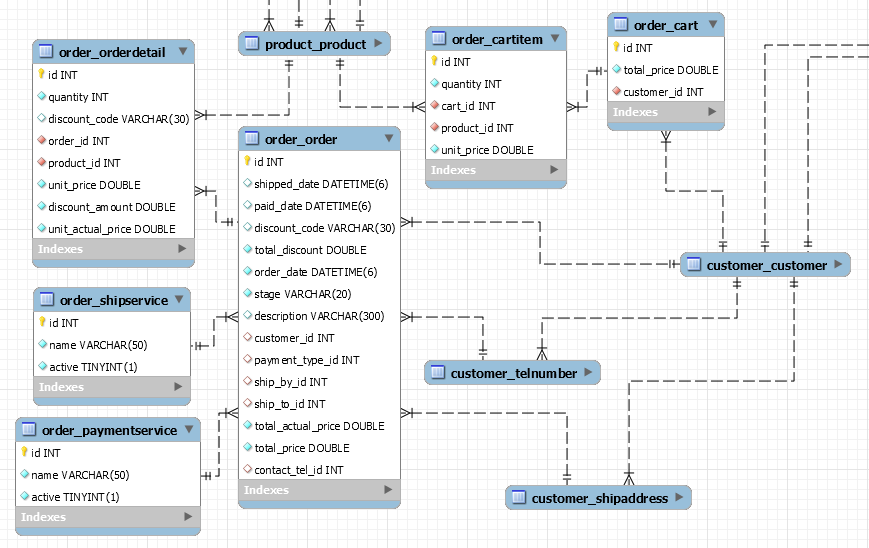
Hình 4. Dữ liệu Product

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,…).

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).

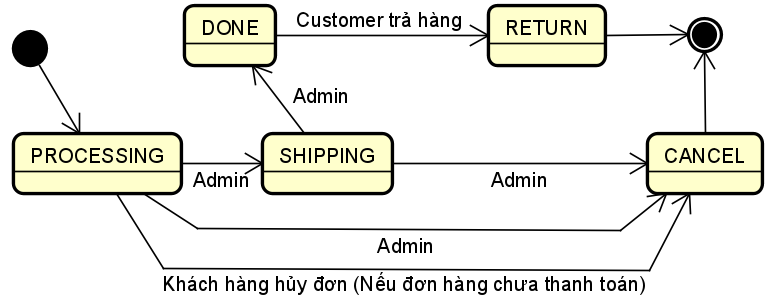
Bảng ProductAttribute dùng để định nghĩa các thuộc tính của 1 sản phẩm. ProductAttributeValue dùng để xác định giá trị của thuộc tính và sản phẩm. Cách thiết kế này được dựa theo mô hình Entity-Attribute-Value (EAV) [1] giúp cho dữ liệu sản phẩm có thể dễ dàng thêm mới các thuộc tính về sau.

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

Hình 5. Dữ liệu Order

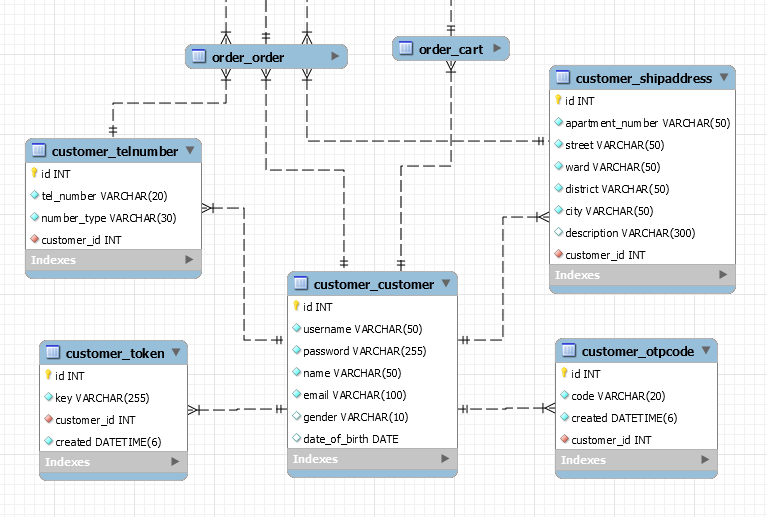
Bảng ShipService và PaymentService được dùng để lưu trữ các thông tin dịch cung cấp bởi bên thứ 3 cho hệ thống.

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 6. 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 7. Dữ liệu Customer

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

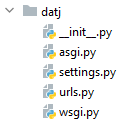
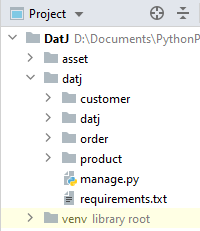
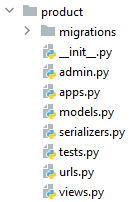
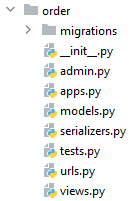
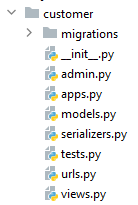
Bảng OTPCode được dùng để lưu mã xác nhận 1 lần, phục vụ cho việc khôi phục tài khoản nếu khách hàng bị quên mật khẩu, và thể dùng để xác thực cho các chức năng cần chứng thực.

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 8. 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:

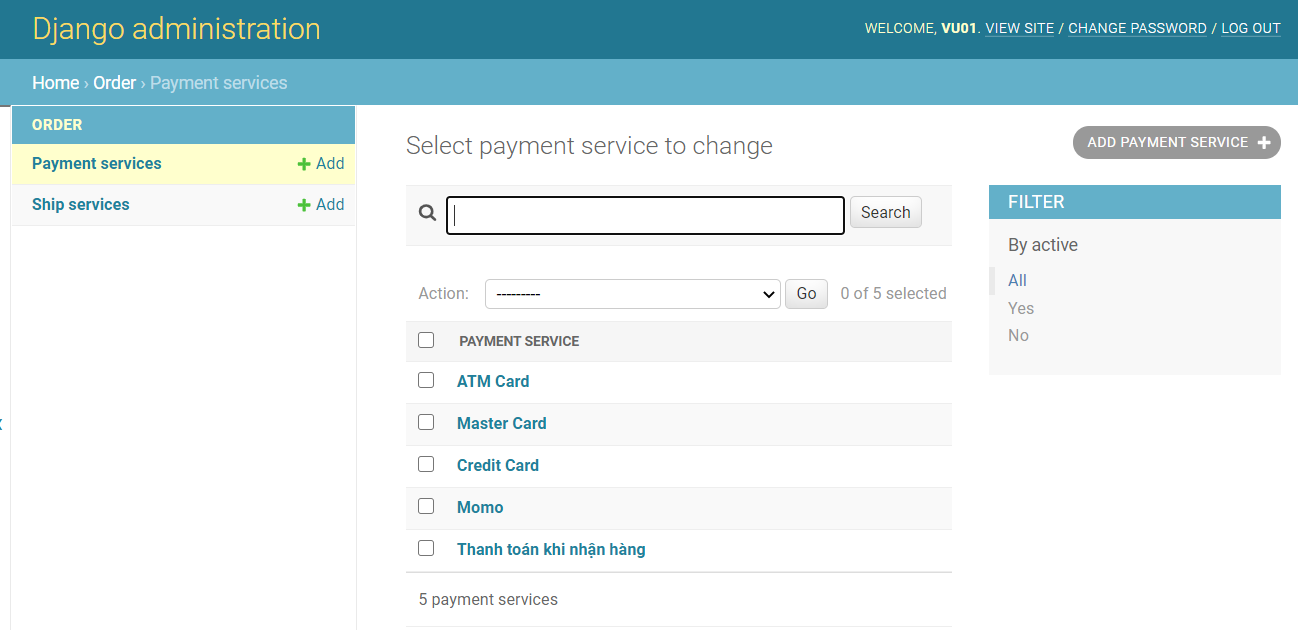
admin.site.register(Product, ProductAdmin)

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

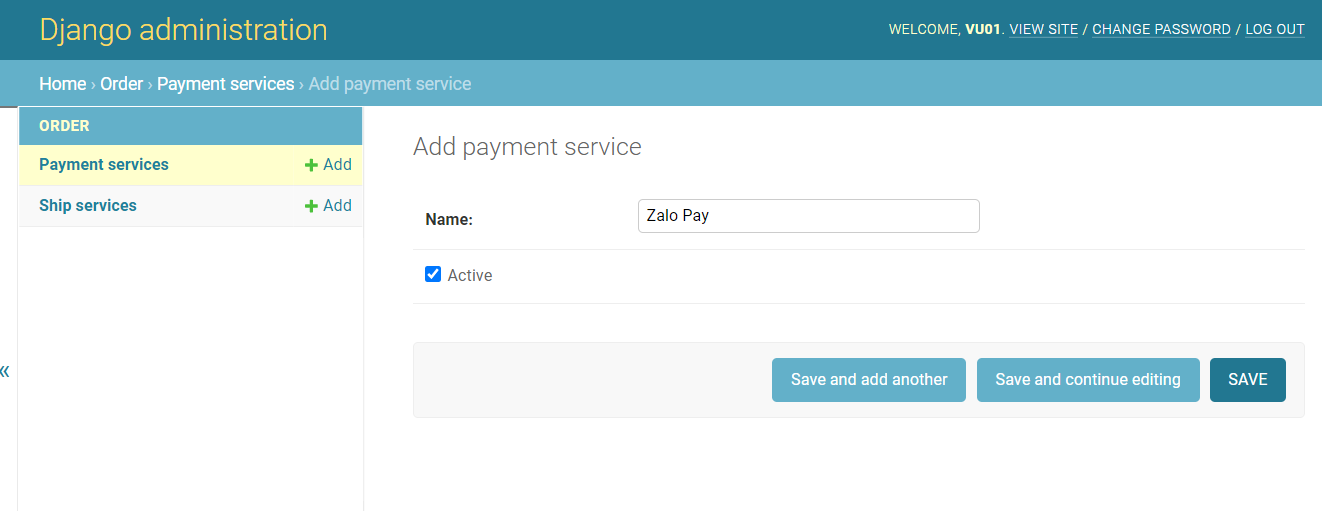
Sau khi hoàn tất, em dùng lệnh py manage.py runserver để chạy project và có được trang admin cơ bản như sau:

- Quản lý các dịch vụ bên thứ 3:

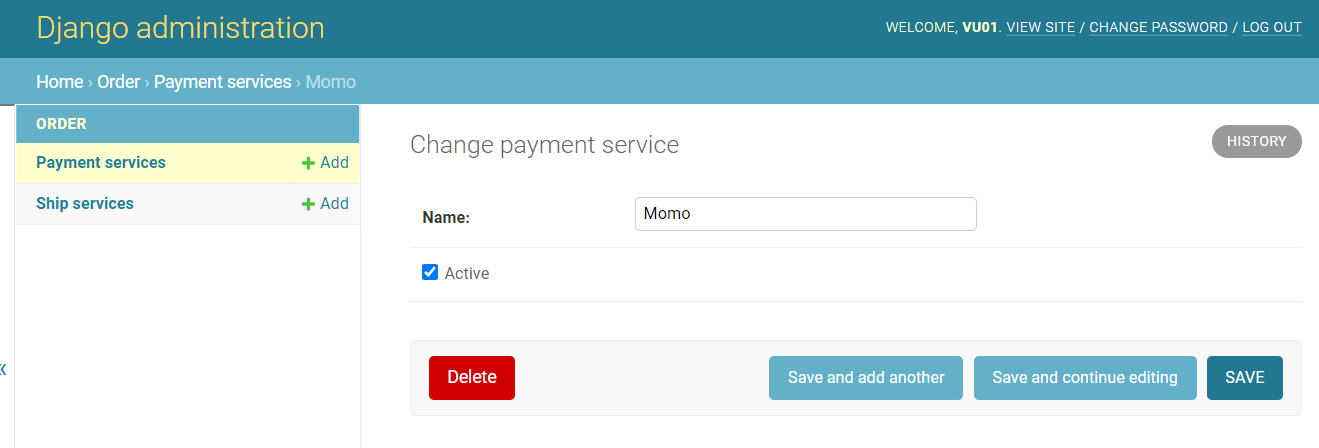
Xem danh sách các dịch vụ



Thêm mới dịch vụ

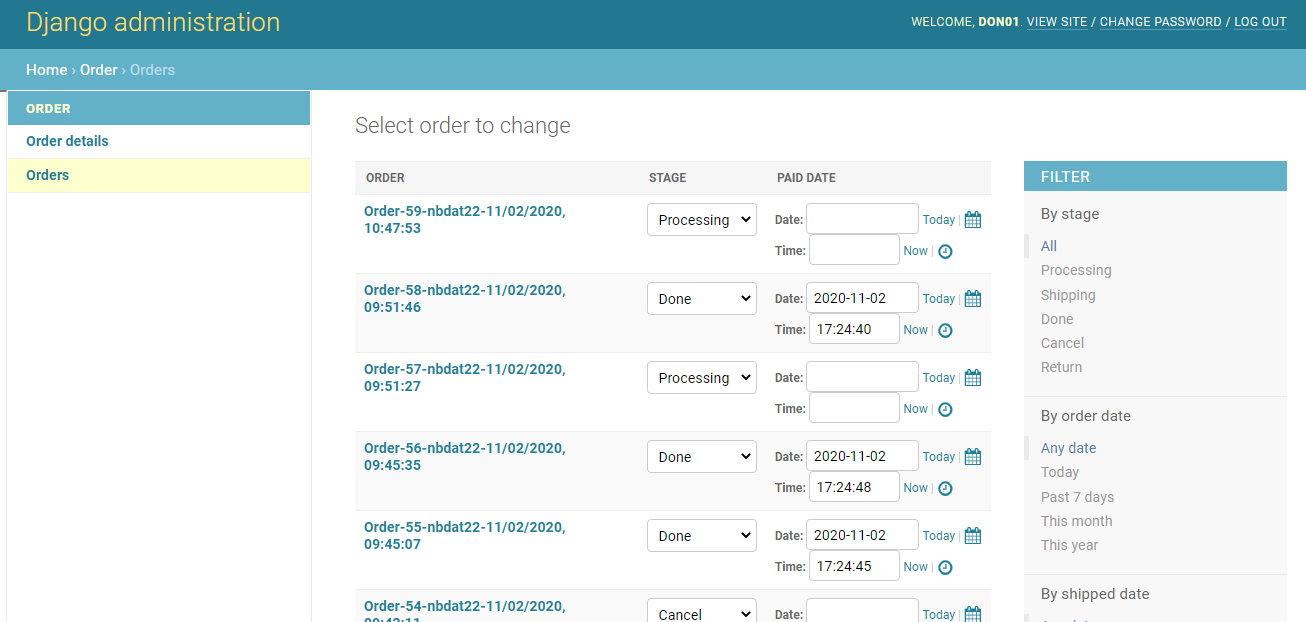


Chỉnh sửa thông tin hoặc xóa dịch vụ



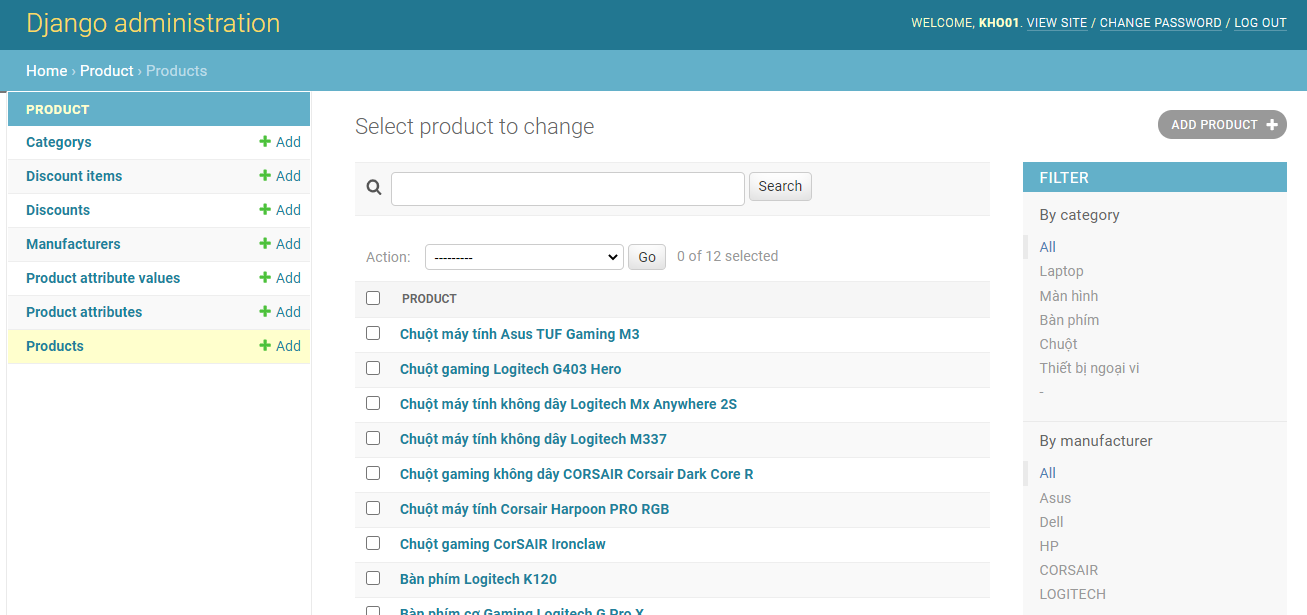
- Quản lý kiểm kê đơn hàng:

Xem danh sách đơn hàng, cập nhật trạng thái đơn hàng

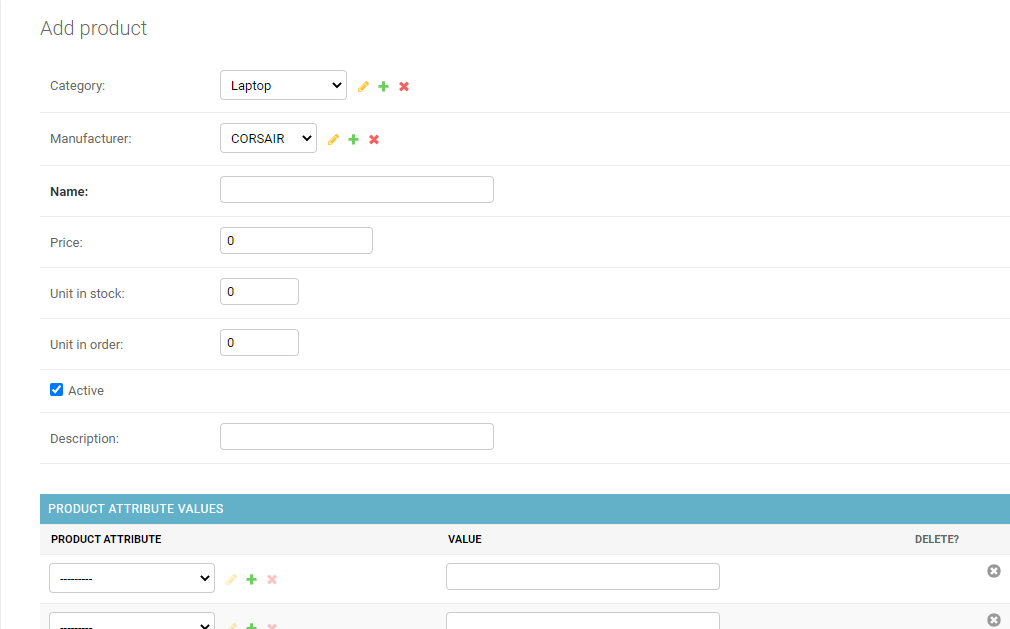
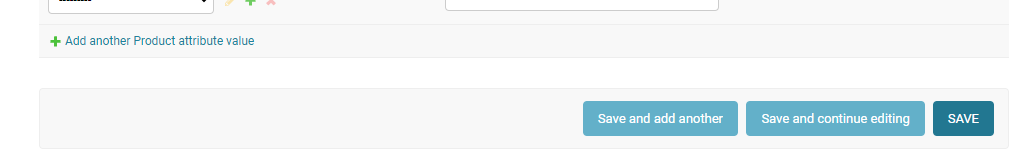


- Quản lý sản phẩm và thông tin khuyến mãi:

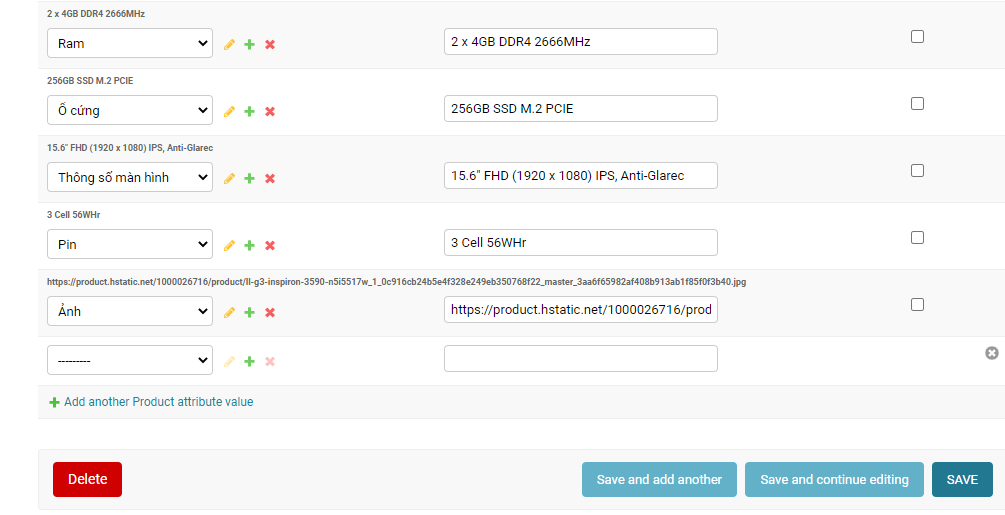
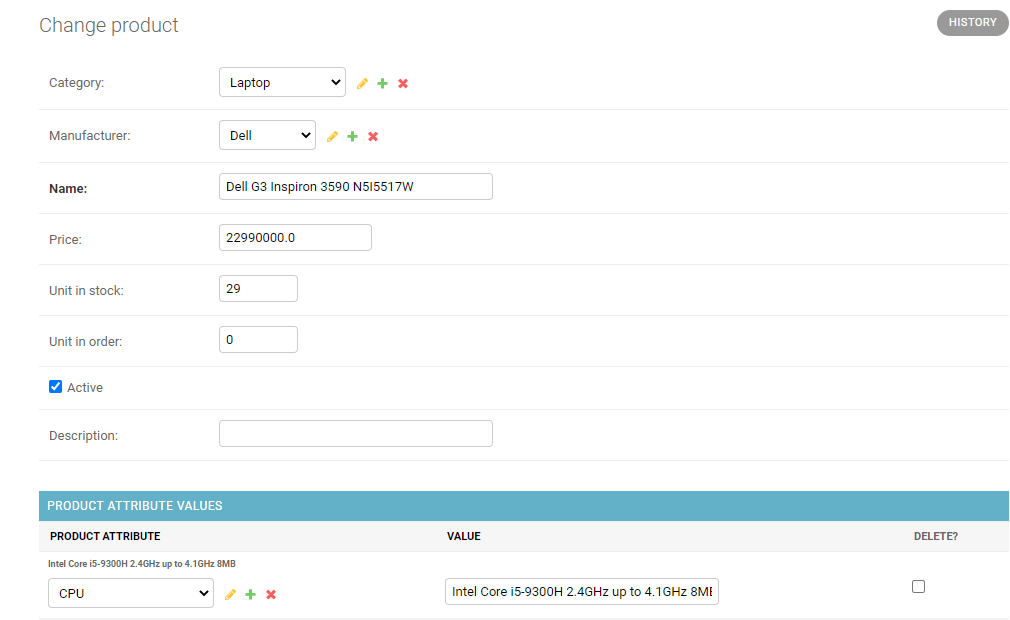
Xem danh sách sản phẩm (Tương tự khyến mãi và các bảng khác)



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

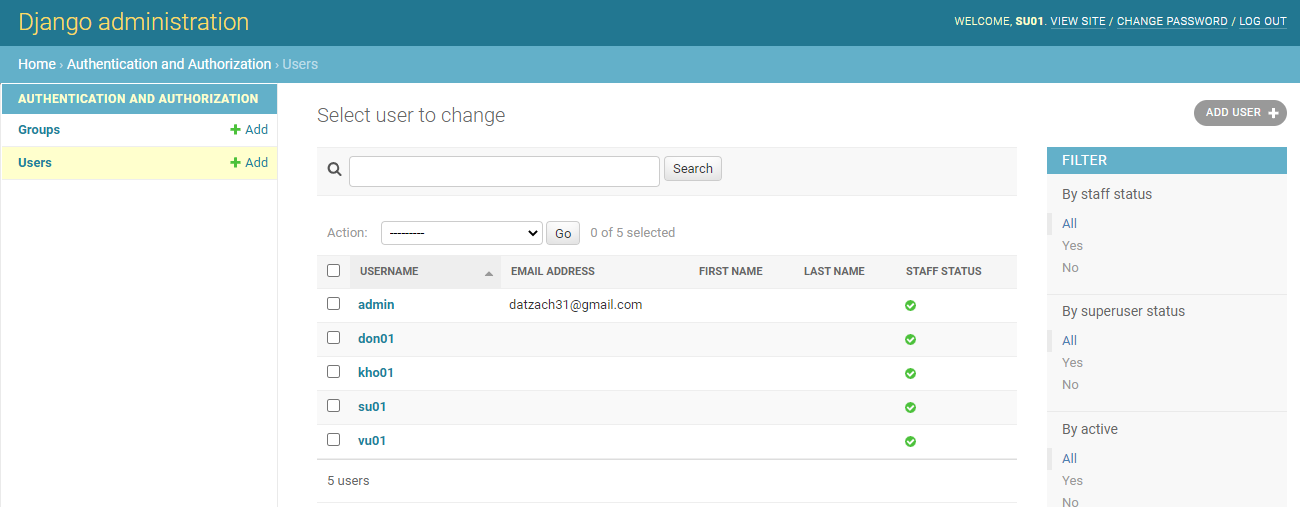


Xóa hoặc chỉnh sửa thông tin sản phẩm (Tương tự khyến mãi và các bảng khác)

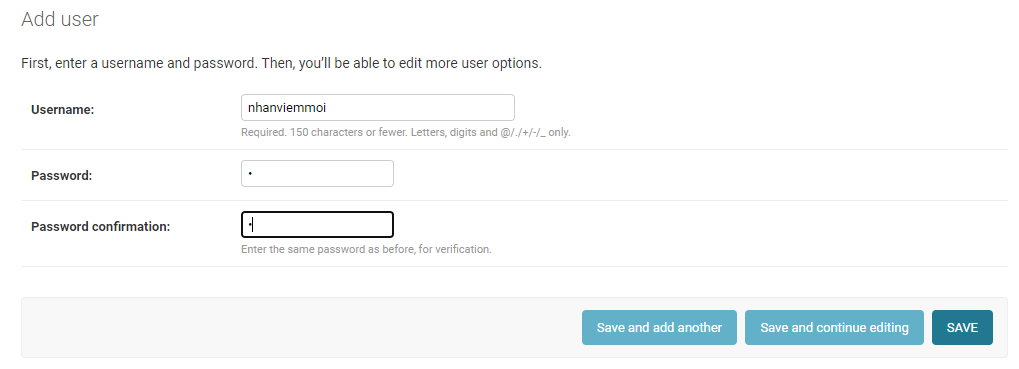


- Quản lý nhân sự:

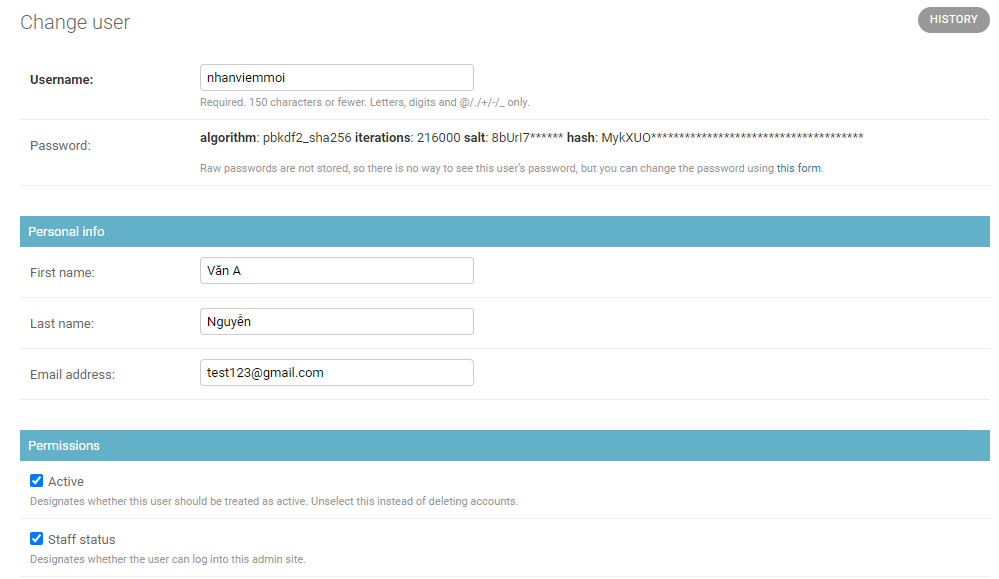
Xem danh sách nhân viên



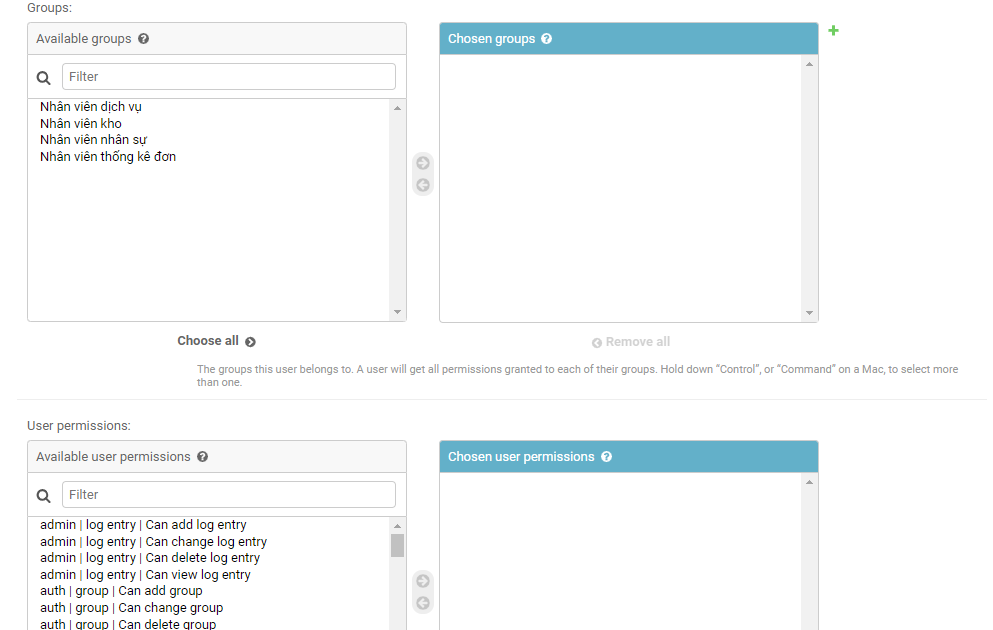
Thêm nhân viên mới



Thêm, chỉnh sửa các thông tin của nhân viên

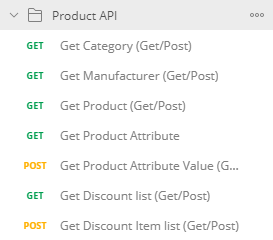


Phân quyền cho nhân viên vừa thêm

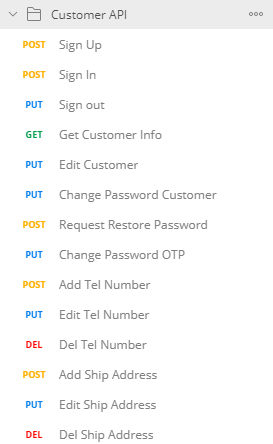


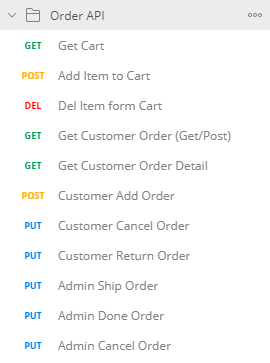
## 3.4. Xây dựng các API theo yêu cầu hệ thống

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:

- 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:

- 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:



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\_attribute)  
 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)

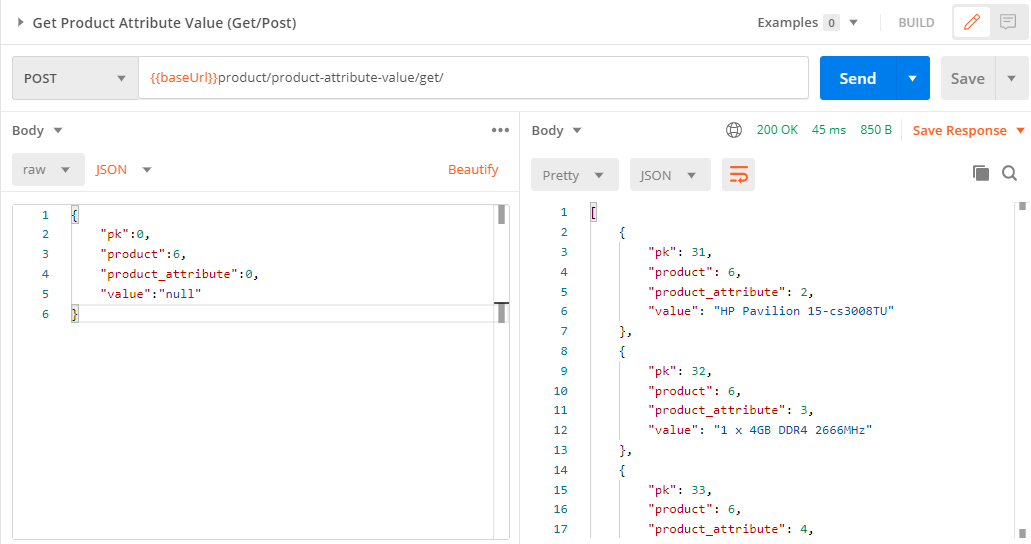
Sau các bước đó em đã có được 1 API xử lý lấy giá trị thuộc tính của sản phẩm theo các từ khóa được gửi lên server.

* **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:



1

3

2

5

4

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

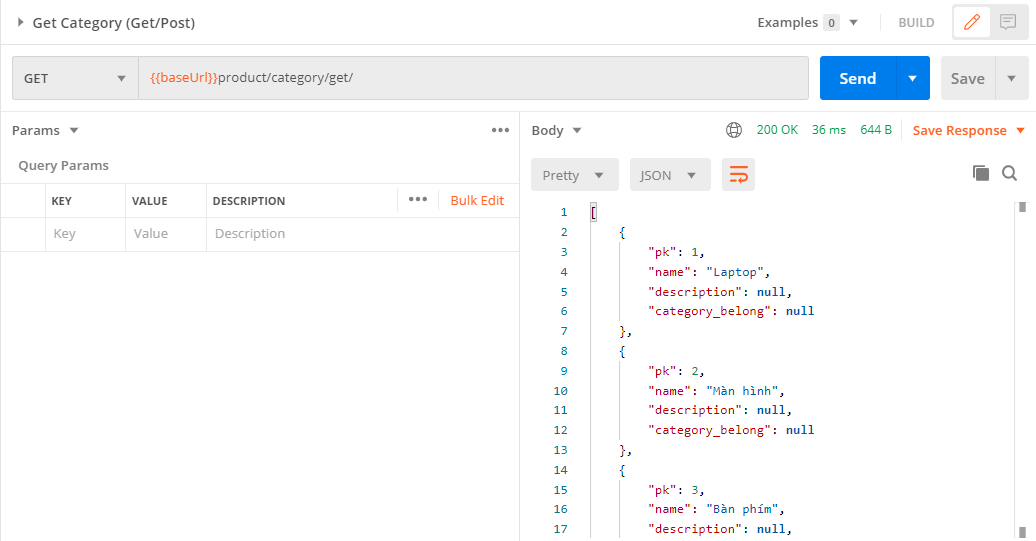
5: Status code trả về từ API

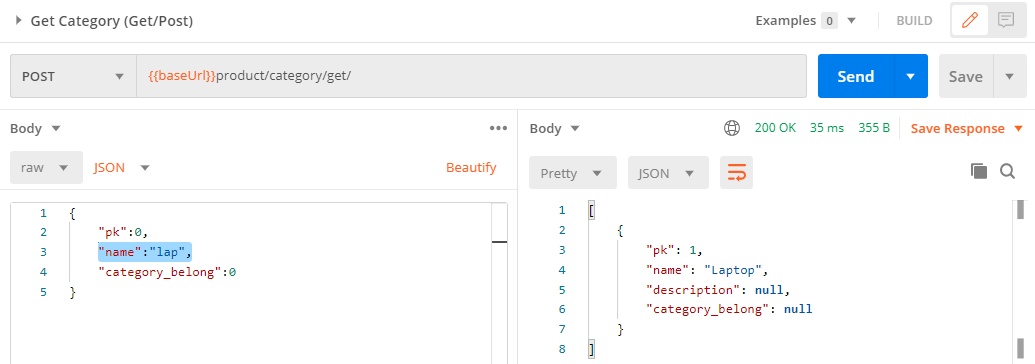
### 3.4.1. API Product

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



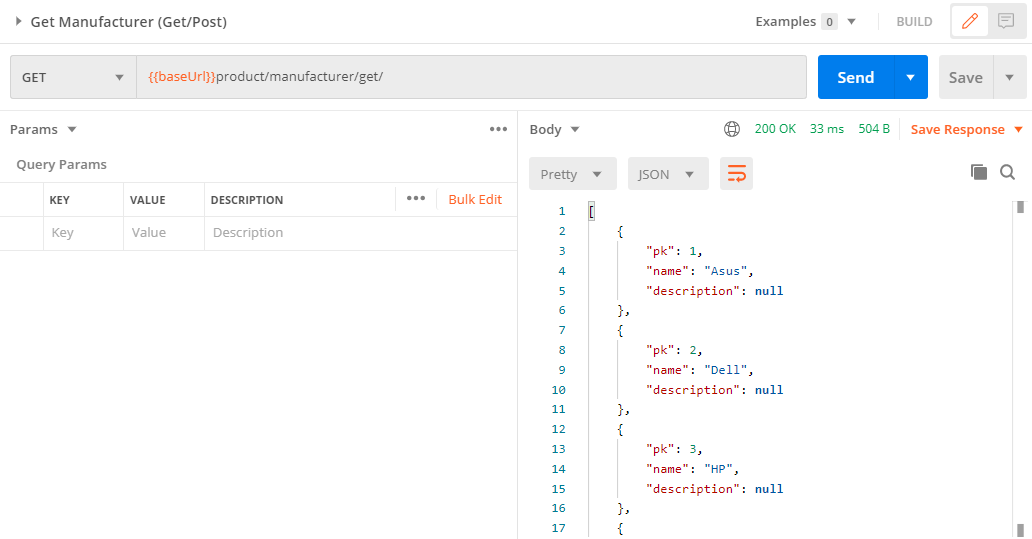
API dùng để lấy danh sách các loại sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

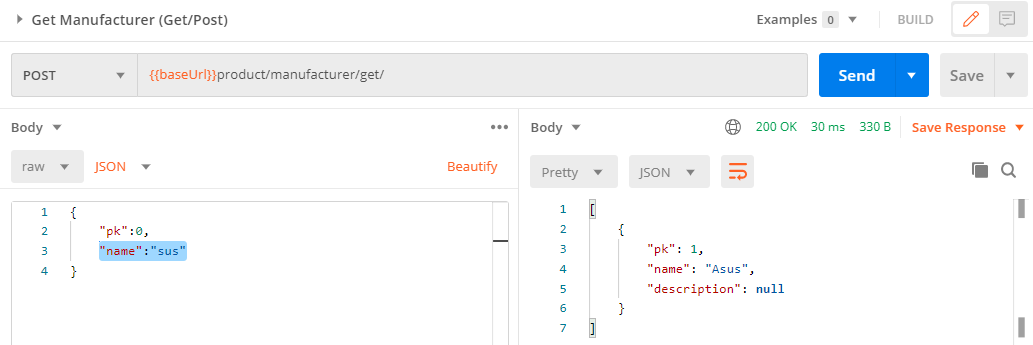
Kết quả:

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



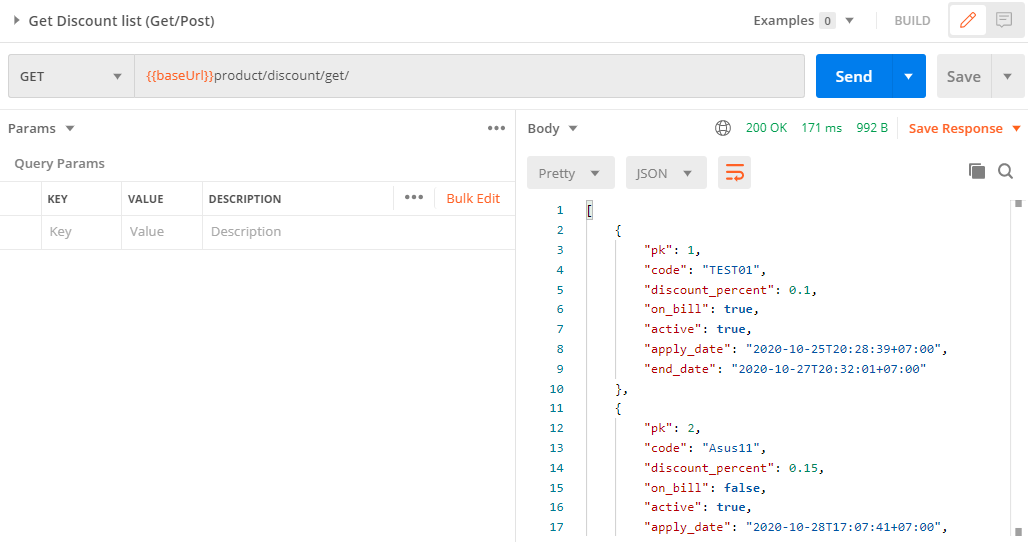
API dùng để lấy danh sách các nhà sản xuất (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

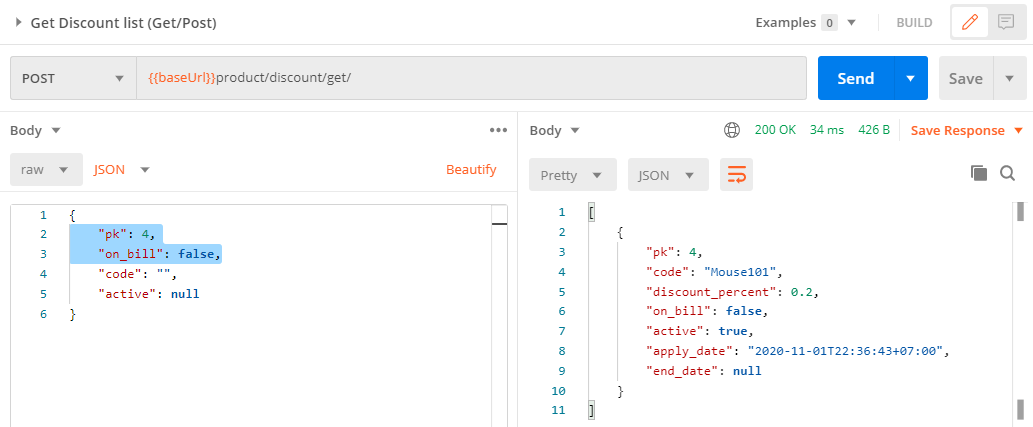
Kết quả:

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



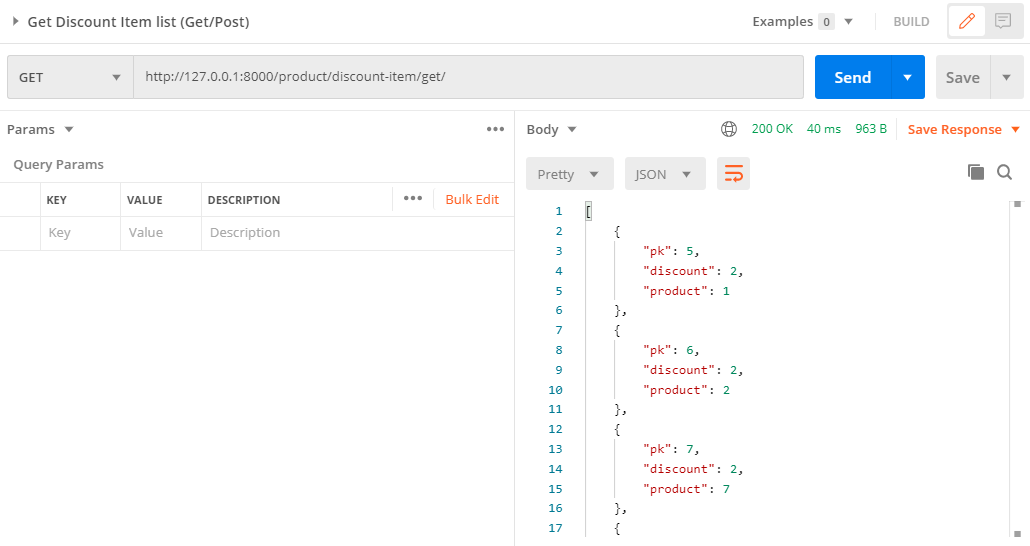
API dùng để lấy danh sách các thông tin khuyến mãi (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

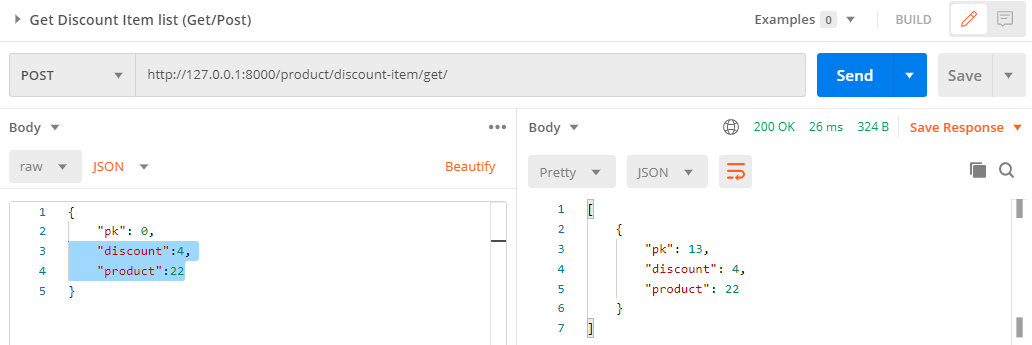
Lấy khuyến mãi có mã 4 và điều kiện là giảm giá trên toàn bill.



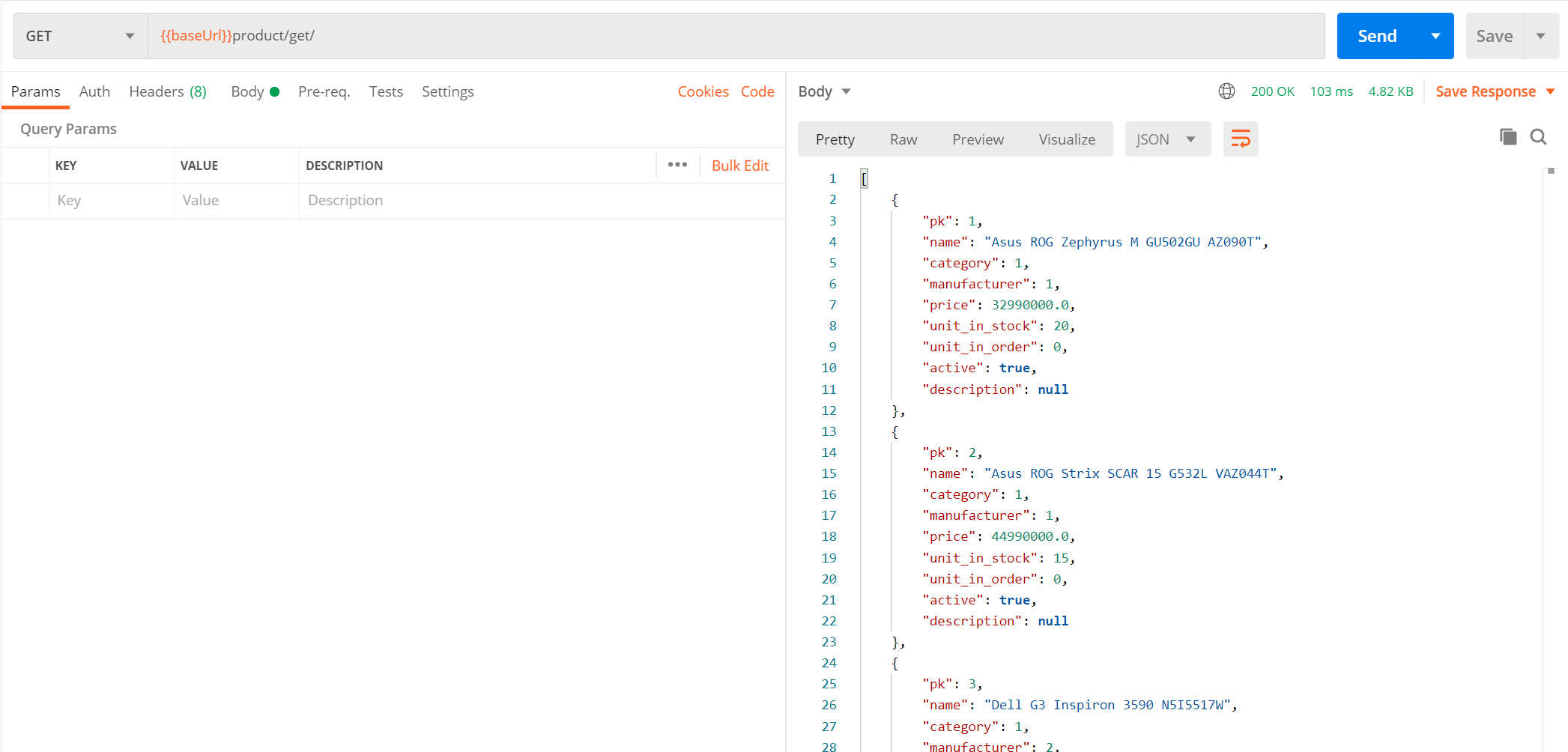
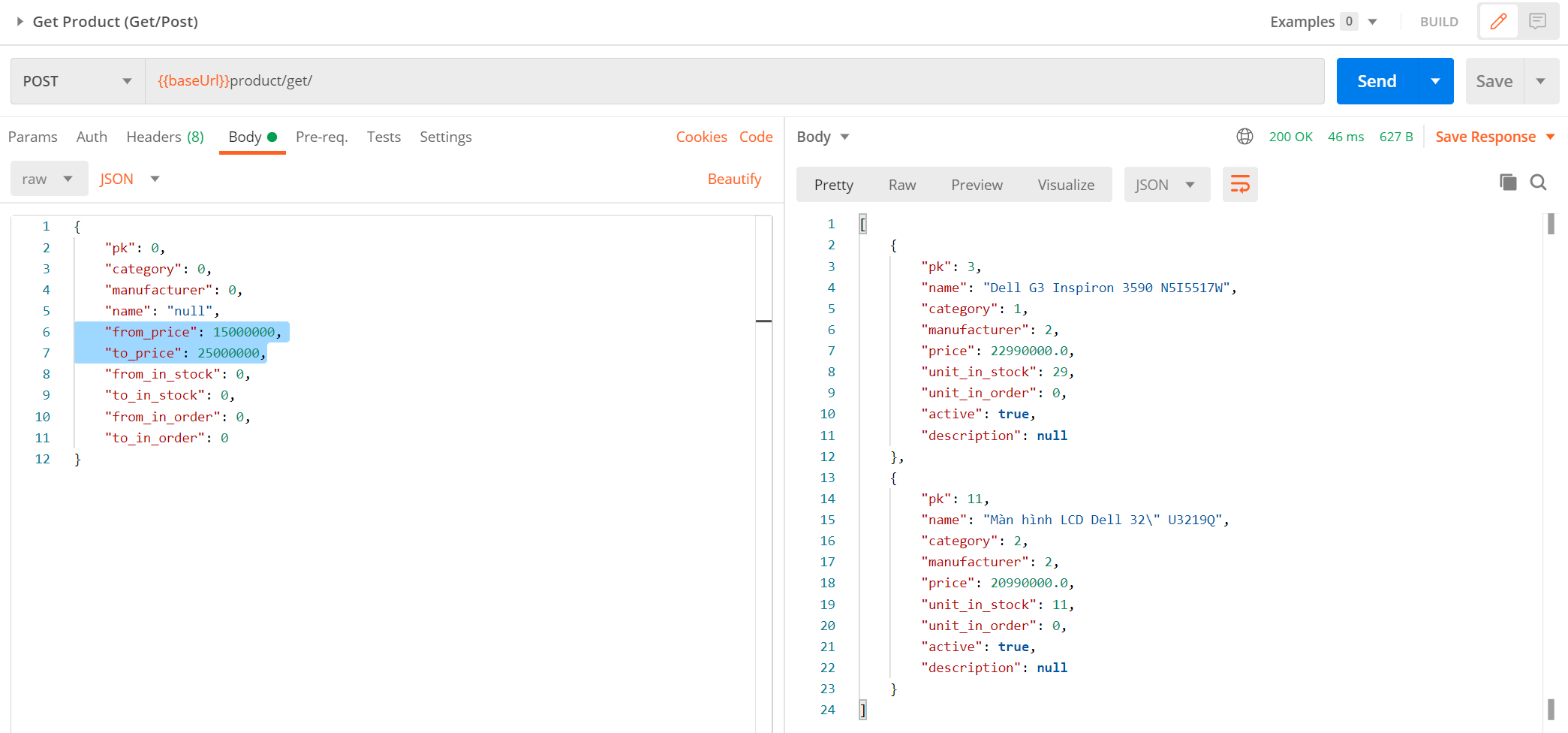
API dùng để lấy danh sách khuyến mãi của các sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

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

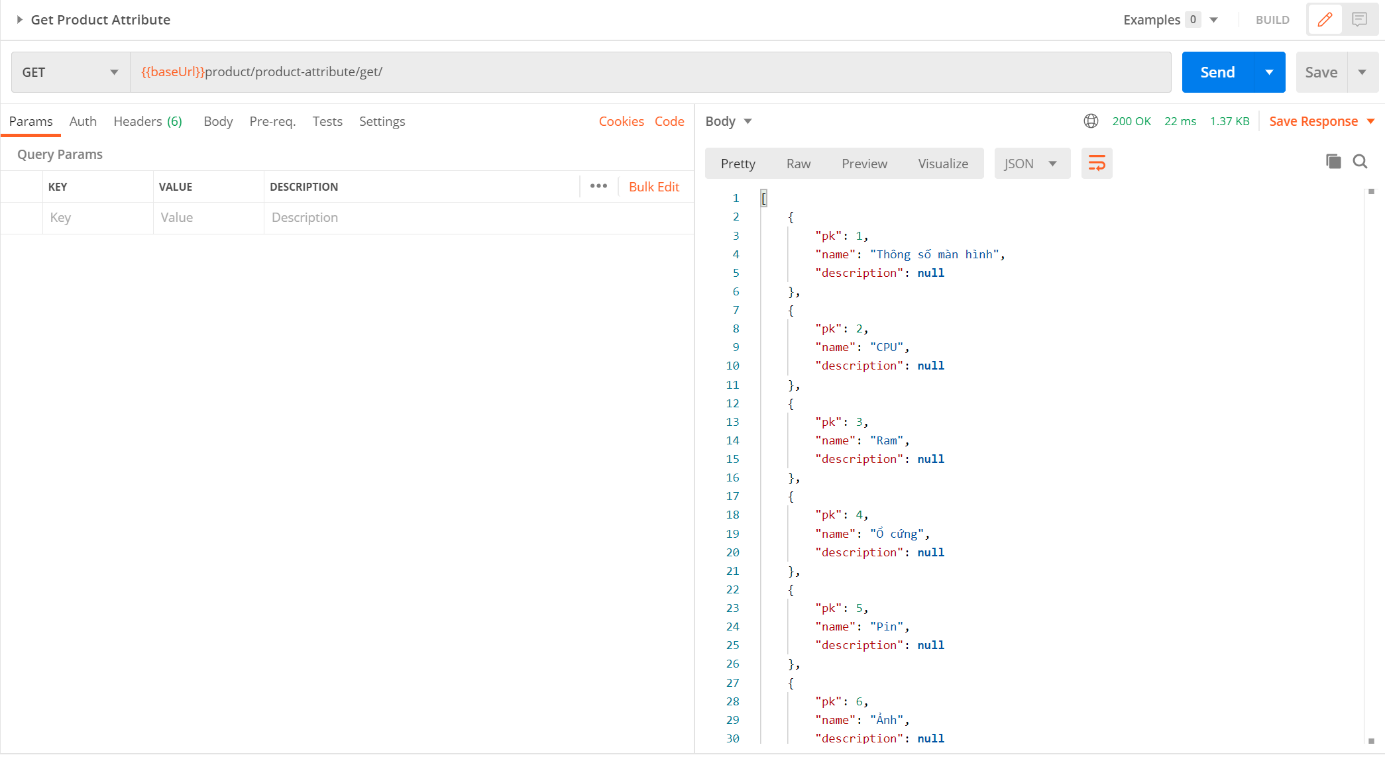


API dùng để lấy danh sách tất cả sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

Kết quả:

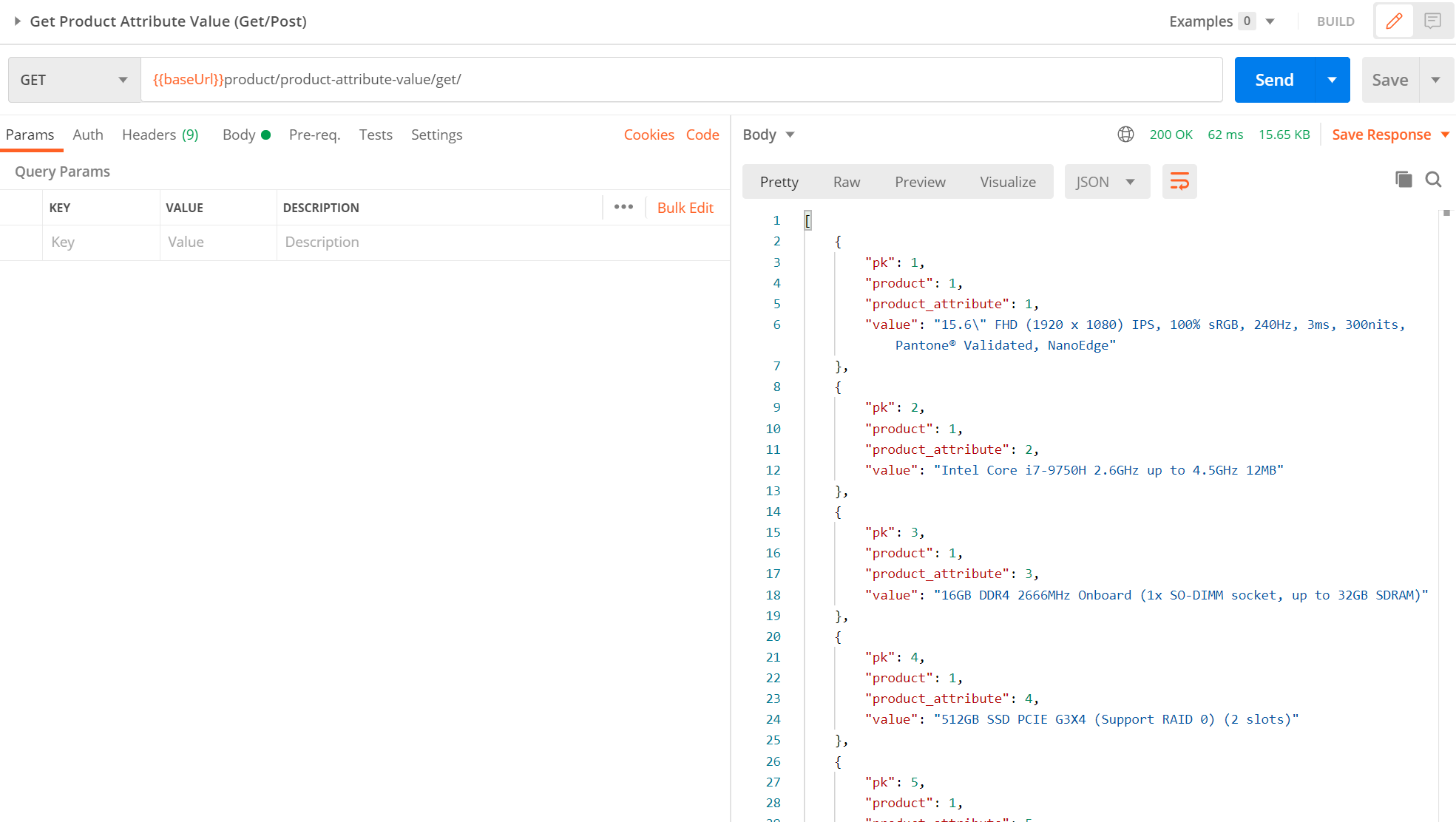
Lấy các sản phẩm có giá từ 15000000 đến 25000000.

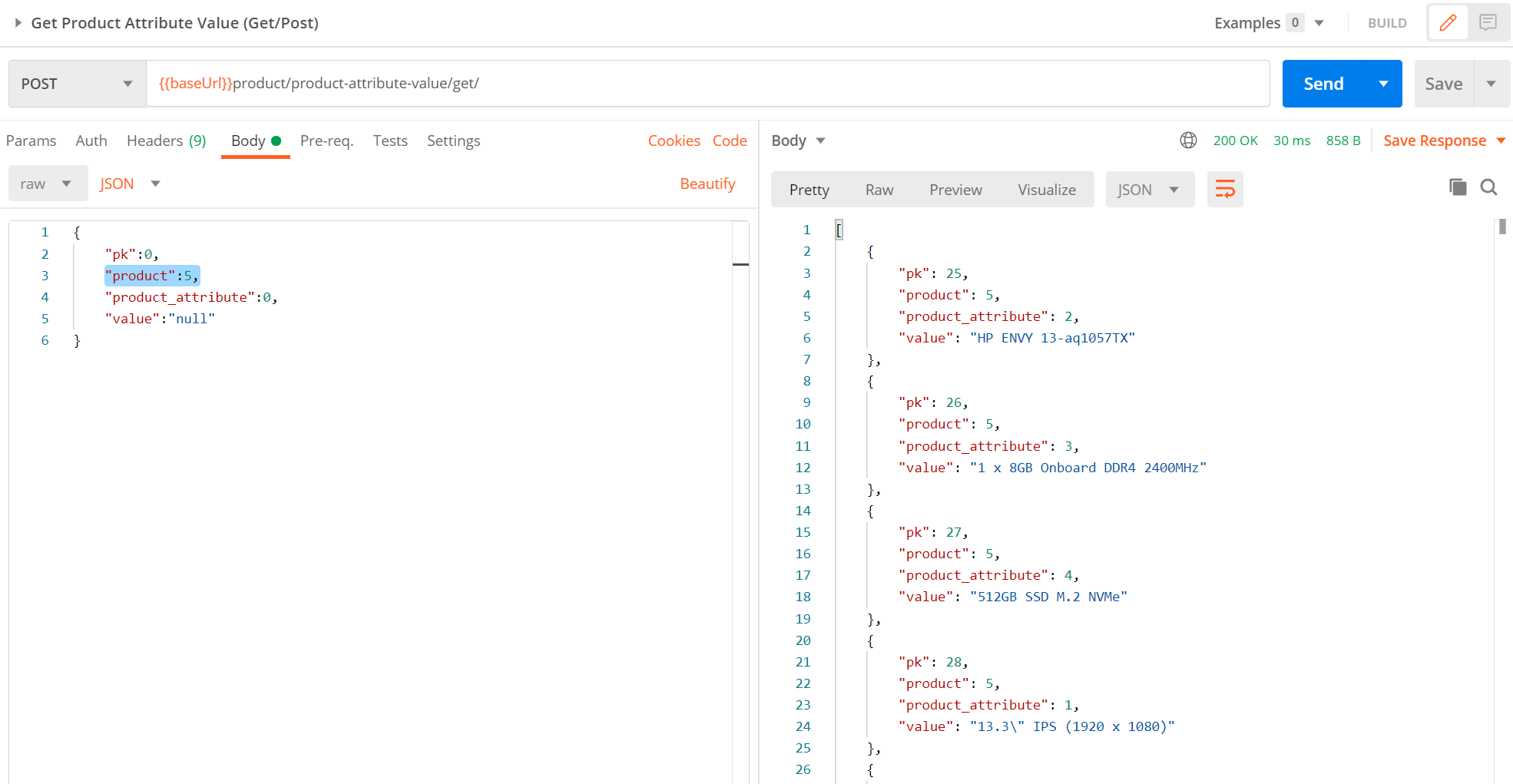
API dùng để lấy danh sách tất cả thuộc tính của tất cả sản phẩm (GET).

Kết quả:



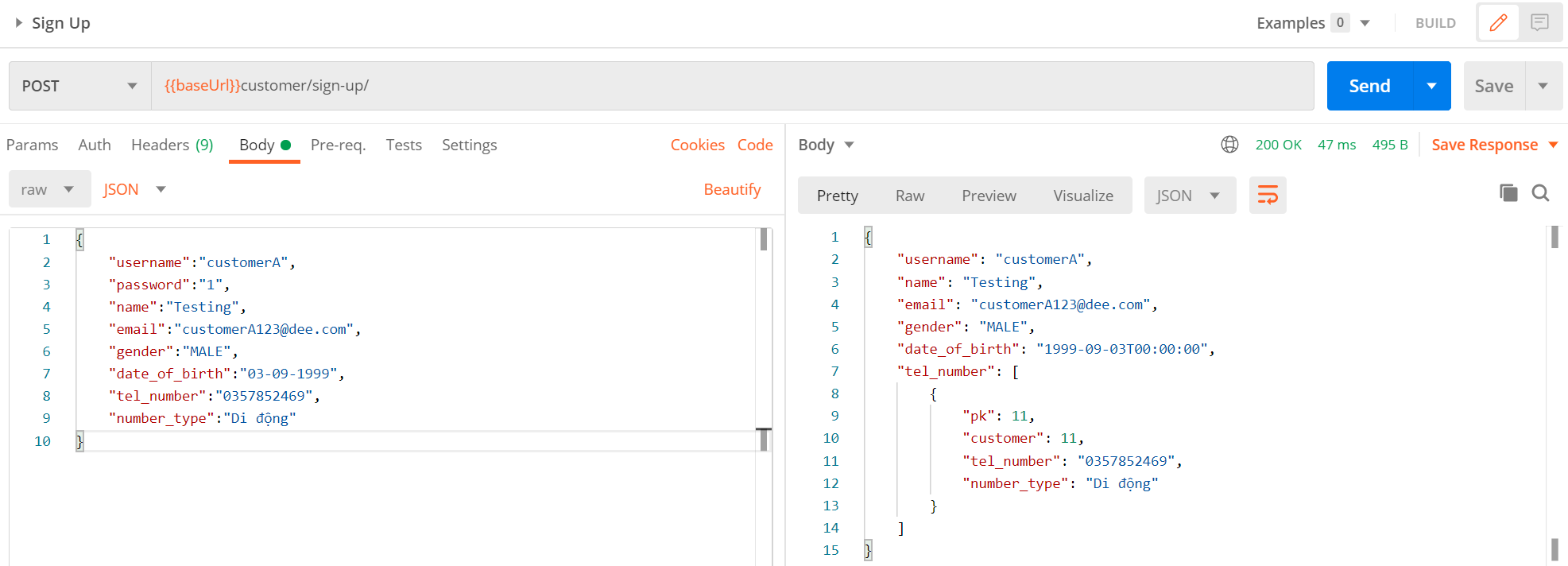
API dùng để lấy danh sách tất cả giá trị thuộc tính của các sản phẩm (GET), và lọc lấy theo điều kiện của dữ liệu gửi lên server (POST).

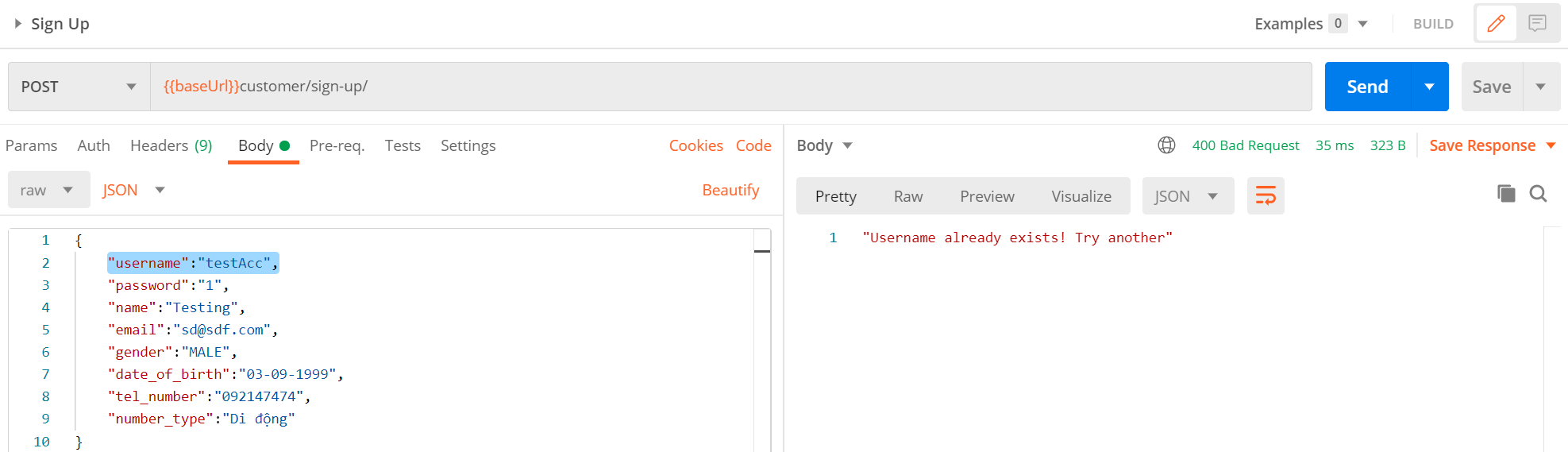
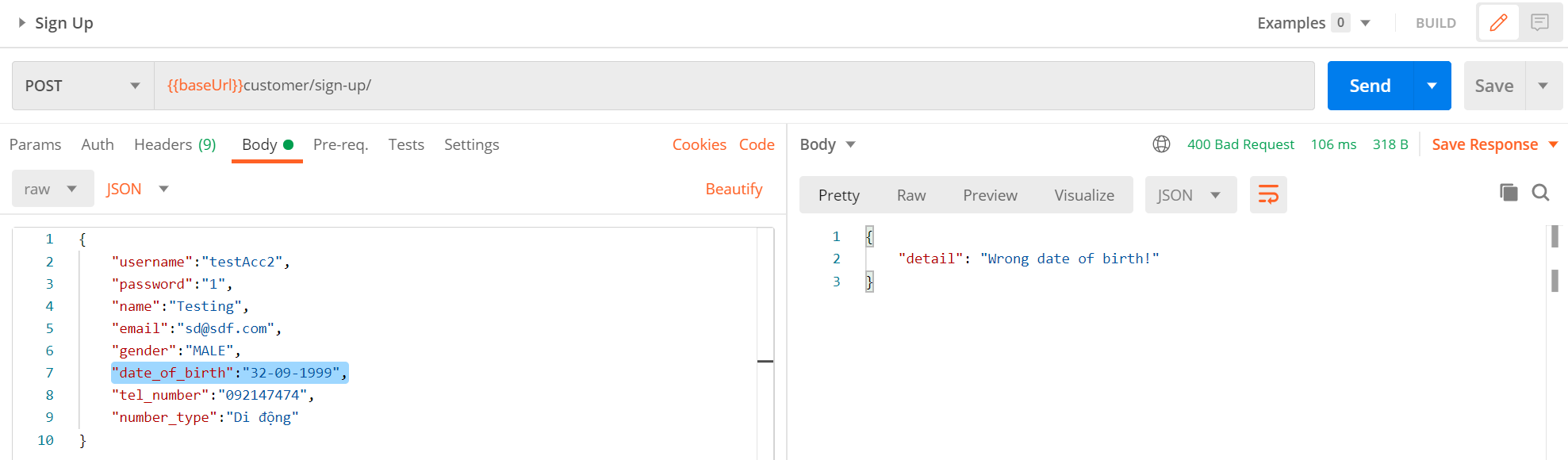
Kết quả:

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

### 3.4.2. API Customer

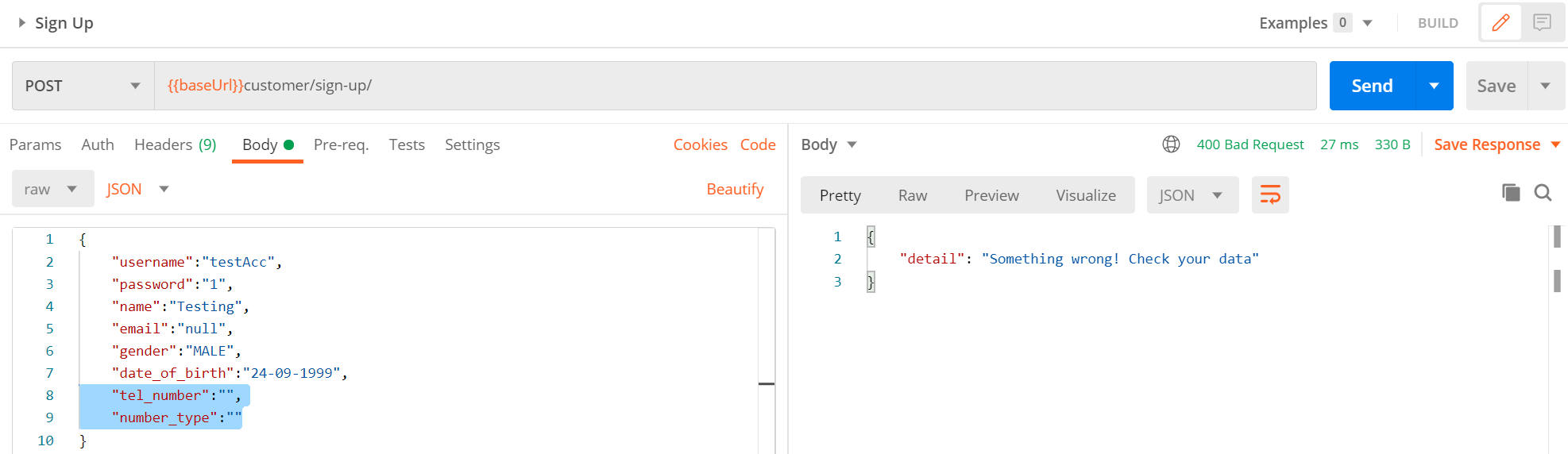


Đây là API với method POST giúp khách hàng tạo tài khoản mới vào hệ thống bằng việc cung cấp các thông tin cần thiết.

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

Hình 2. API Sign Up: Sai thông tin ngày sinh

Hình 1. API Sign Up: Tên tài khoản đã tồn tại



Hình 3. API Sign Up: Sai thông tin SĐT

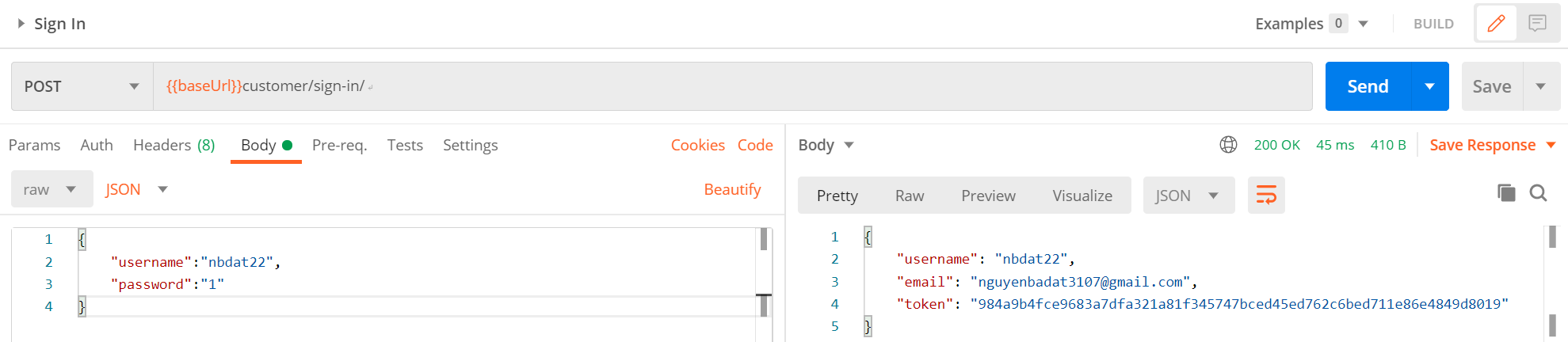
Trước khi lưu thông tin tài khoản khách hàng vào cơ sở dữ liệu, mật khẩu của khách hàng sẽ đăng mã hóa bằng hàm băm sha256 cung cấp bởi thư viện hashlib trong python để tăng tính bảo mật.

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:



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.

### 3.4.3. API Order

# Chương 4. Kết luận

# TÀI LIỆU THAM KHẢO

[1] <https://viblo.asia/p/tim-hieu-ve-entity-attribute-value-pattern-eav-structural-pattern-Eb85okV452G> - “”

[2] <https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx> - “”

[3] <https://docs.djangoproject.com/en/3.1/ref/contrib/admin/> - “”

[4] <https://www.django-rest-framework.org/tutorial/quickstart/> - “”

[5] <https://www.django-rest-framework.org/api-guide/authentication/> - “”

[6] <https://docs.djangoproject.com/en/3.1/topics/email/> - “”

[7] <https://medium.com/eway/nguy%C3%AAn-t%E1%BA%AFc-thi%E1%BA%BFt-k%E1%BA%BF-rest-api-23add16968d7> – “”

<https://docs.djangoproject.com/en/3.1/topics/email/> - “01/11/2020”

<https://blog.mailtrap.io/django-send-email/> - “01/11/2020”

https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/ - “”