Họ tên: Trần Vân Anh
MSV: B20DCCN075

I. Câu 1-5

Functionalities

Actor	Functionalities	Detailed Description
Customer	Tìm Kiếm (Search)	- **By Browser**: Người sử dụng có thể duyệt trên web và chọn mặt hàng mình muốn để thêm vào giỏ hàng. Người sử dụng có thể xem chi tiết mặt hàng. - **By Keyword**: Tìm kiếm sản phẩm thông qua từ khóa. - **By Image**: Tìm kiếm sản phẩm bằng cách tải hình ảnh của sản phẩm. - **By Voice**: Tìm kiếm sản phẩm bằng cách nói tên hoặc mã của sản phẩm.
	Thêm Vào Giỏ Hàng (Add to Cart)	Thêm sản phẩm vào giỏ hàng.
	Thanh Toán (Pay)	Thực hiện thanh toán sản phẩm
	Đặt Hàng (Order)	Đặt hàng và theo dõi trạng thái đơn hàng.
	Vận Chuyển (Shipping)	Xem chi tiết và tùy chọn giao hàng.
	Xem sách (View Book)	Xem các sách đang có sẵn
	Đăng Ký (Customer Registration)	Đăng ký tài khoản trên website
	Đăng Nhập (Customer Login)	Đăng nhập vào hệ thống.
	Thay Đổi Mật Khẩu (Change Password)	Thay đổi mật khẩu tài khoản.
Admin	Quản lý sách (Manage Book)	Có thể thêm, chỉnh sửa, xem và xóa sách.

	Quản lý đơn hàng (Manage Order)	Có thể thêm đơn hàng mới, xem danh sách đơn hàng, chỉnh sửa và xóa đơn hàng.
	Quản lý khách hàng (Manage Customer)	Có thể thêm bản ghi khách hàng mới, xem danh sách khách hàng, chỉnh sửa và xóa bản ghi khách hàng.
	Quản lý kho (Manage Stock)	Quản lý toàn bộ hàng tồn kho, bao gồm việc thêm mới, cập nhật thông tin, và theo dõi số lượng hàng hóa.

1. Mô tả bằng ngôn ngữ tự nhiên

Module Sách:

Admin có thể quản lý sách

Admin có thể chỉnh sửa/xóa sách

Admin có thể xem danh sách tất cả sách

Khách hàng có thể xem sách của mình

Module Đơn hàng:

Admin có thể thêm đơn hàng mới

Admin có thể xem danh sách chi tiết đơn hàng

Chỉ có admin mới có thể chỉnh sửa và cập nhật bản ghi của đơn hàng

Admin sẽ có thể xóa các bản ghi của đơn hàng

Module Khách hàng:

Admin có thể thêm bản ghi khách hàng mới

Admin có thể xem danh sách chi tiết khách hàng

Chỉ có admin mới có thể chỉnh sửa và cập nhật bản ghi của khách hàng

Admin sẽ có thể xóa các bản ghi của khách hàng

Module Kho:

Admin có thể quản lý kho

Admin có thể chỉnh sửa/xóa kho

Admin có thể xem danh sách tất cả các kho

Khách hàng có thể xem kho của mình

Các chức năng được thực hiện bởi người dùng Admin:

Đăng nhập cho Admin

Quên mật khẩu cho Admin

Chỉnh sửa Hồ sơ cho Admin

Thay đổi Mật khẩu cho Admin

Chức năng Đăng xuất

Bảng điều khiển cho Người dùng Admin

Quản lý Sách

Thêm Sách Mới

Chỉnh sửa Sách Hiện tại

Xem chi tiết về Sách

Liệt kê tất cả các Sách

Quản lý Đơn hàng

Thêm Đơn hàng Mới

Chỉnh sửa Đơn hàng Hiện tại

Xem chi tiết về Đơn hàng

Liệt kê tất cả các Đơn hàng

Quản lý Khách hàng

Thêm Khách hàng Mới

Chỉnh sửa Khách hàng Hiện tại

Xem chi tiết về Khách hàng

Liệt kê tất cả các Khách hàng

Quản lý Kho Sách

Thêm Kho Sách Mới

Chỉnh sửa Kho Sách Hiện tại

Xem chi tiết về Kho Sách

Liệt kê tất cả các Kho Sách

Báo cáo của dự án Cửa hàng Sách Trực tuyến

Báo cáo về tất cả các Sách

Báo cáo về tất cả các Đơn hàng

Báo cáo về tất cả các Khách hàng

Báo cáo về tất cả các Kho Sách

Các chức năng được thực hiện bởi người dùng Khách hàng:

Đăng ký Khách hàng

Đăng nhập Khách hàng

Tất cả các Trang Sách

Chi tiết Sách

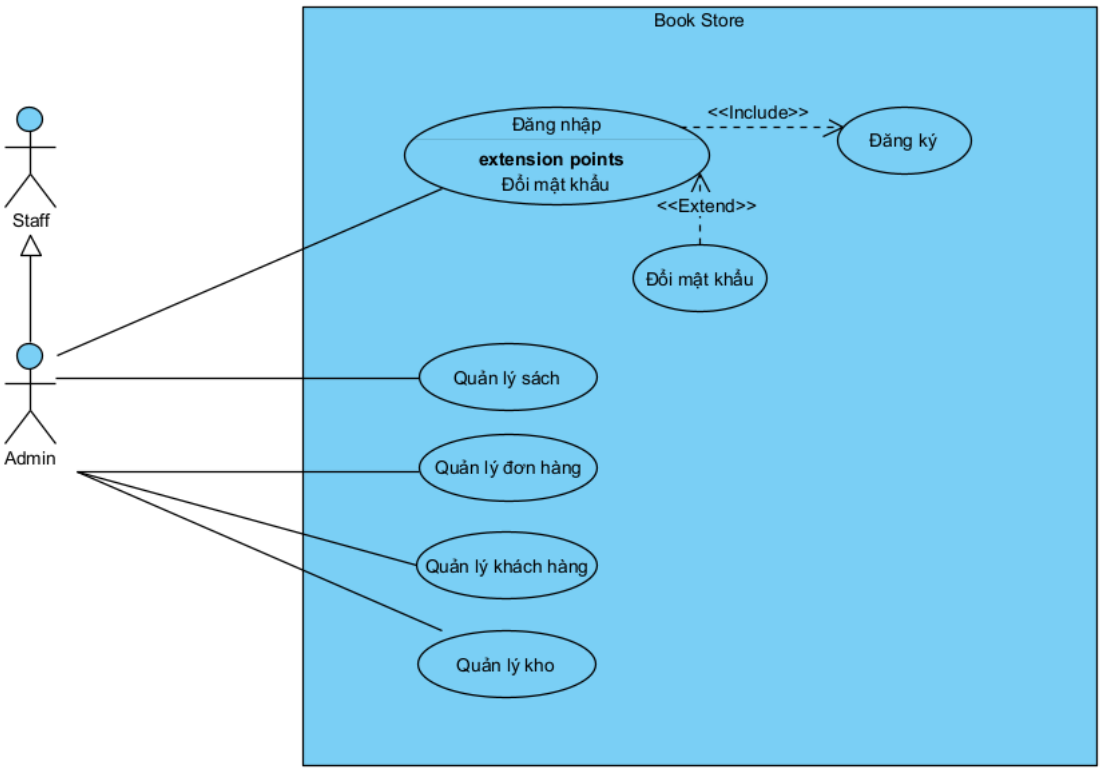
Giỏ hàng

Vận chuyển

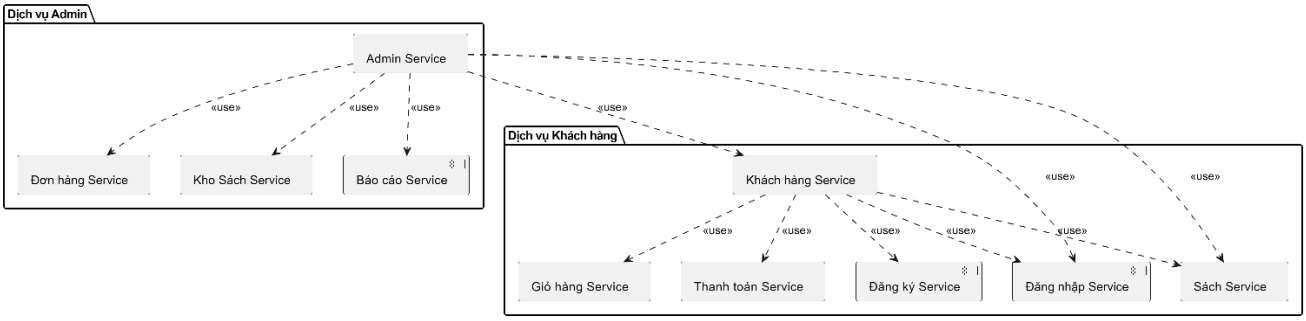
Thanh toán

Đơn hàng của Khách hàng

Thay đổi Mật khẩu



2. Biểu đồ phân rã Hệ ecomSys



3.

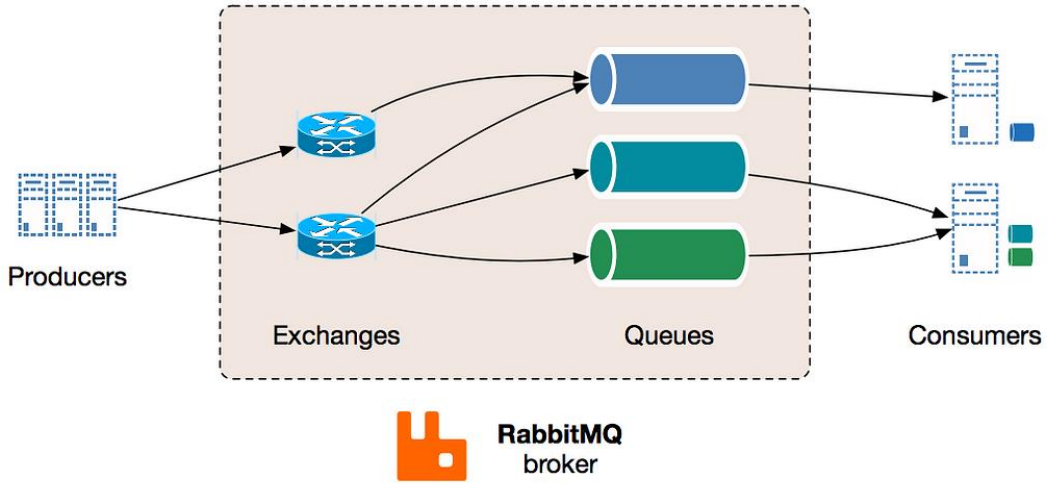
Giao tiếp đồng bộ:

```
import requests


def synchronous_communication():
    # Gửi yêu cầu đồng bộ
    response = requests.get("https://api.service1.com/data")

    # Kiểm tra mã trạng thái
    if response.status_code == 200:
        data = response.json()
        # Xử lý dữ liệu

synchronous_communication()
```

Giao tiếp bất đồng bộ:

```
import requests
import asyncio


async def asynchronous_communication():
    loop = asyncio.get_event_loop()
    # Gửi yêu cầu bất đồng bộ
    response = await loop.run_in_executor(None, requests.get,
    "https://api.service2.com/data")


    # Kiểm tra mã trạng thái
    if response.status_code == 200:
        data = response.json()
        # Xử lý dữ liệu


asyncio.run(asynchronous_communication())
```

Hàng đợi tin nhắn



a. Ưu điểm:

- Giao tiếp không đồng bộ làm giảm khớp nối giữa các dịch vụ

- Có thể xử lý khối lượng lớn tin nhắn và hỗ trợ cân bằng tải

- Hỗ trợ các mẫu nhắn tin khác nhau như xuất bản-đăng ký, điểm-điểm và yêu cầu-trả lời

b. Nhược điểm:

- Giới thiệu sự phức tạp do các thành phần bổ sung (môi giới tin nhắn)

- Có thể là thách thức để theo dõi luồng tin nhắn trong một hệ thống phân tán

- Độ trễ bổ sung so với giao tiếp trực tiếp

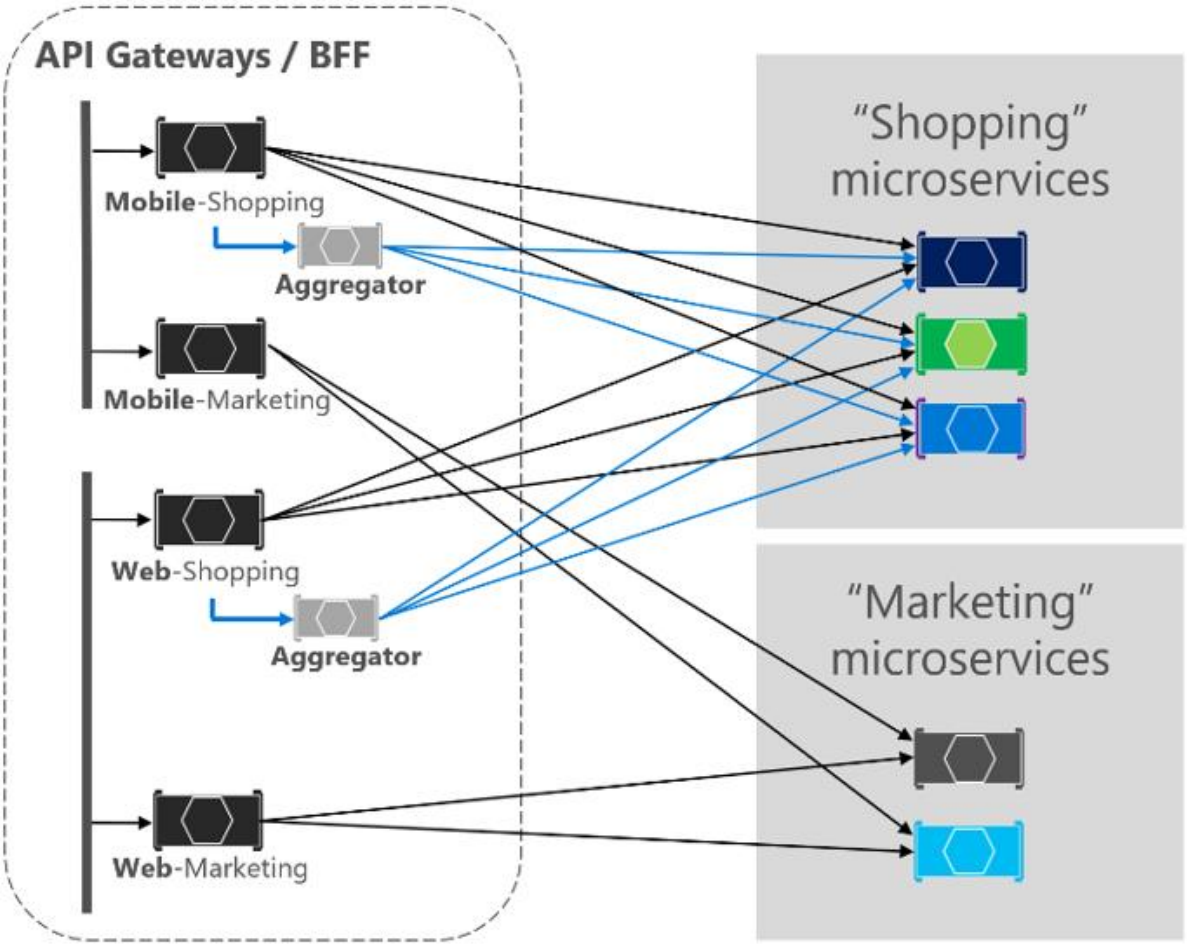
Ví dụ về công nghệ hàng đợi tin nhắn:

- #rabbitmq

- #apachekafka

- #amazon SQS

gRPC



a. Ưu điểm :

- Hiệu suất cao do tuần tự hóa nhị phân

- Hỗ trợ nhiều #programminglanguages

- Truyền phát hai chiều cho phép giao tiếp thời gian thực

- Tự động tạo mã máy khách và máy chủ

b. Nhược điểm :

- Phức tạp hơn để thiết lập và cấu hình so với các API RESTful

- Không được hỗ trợ rộng rãi như RESTful API

- Giới hạn ở giao thức HTTP / 2

5, Trình bày sử dụng các dạng communication giữa các service với code cho hệ ecomSys

Sử dụng API RESTful

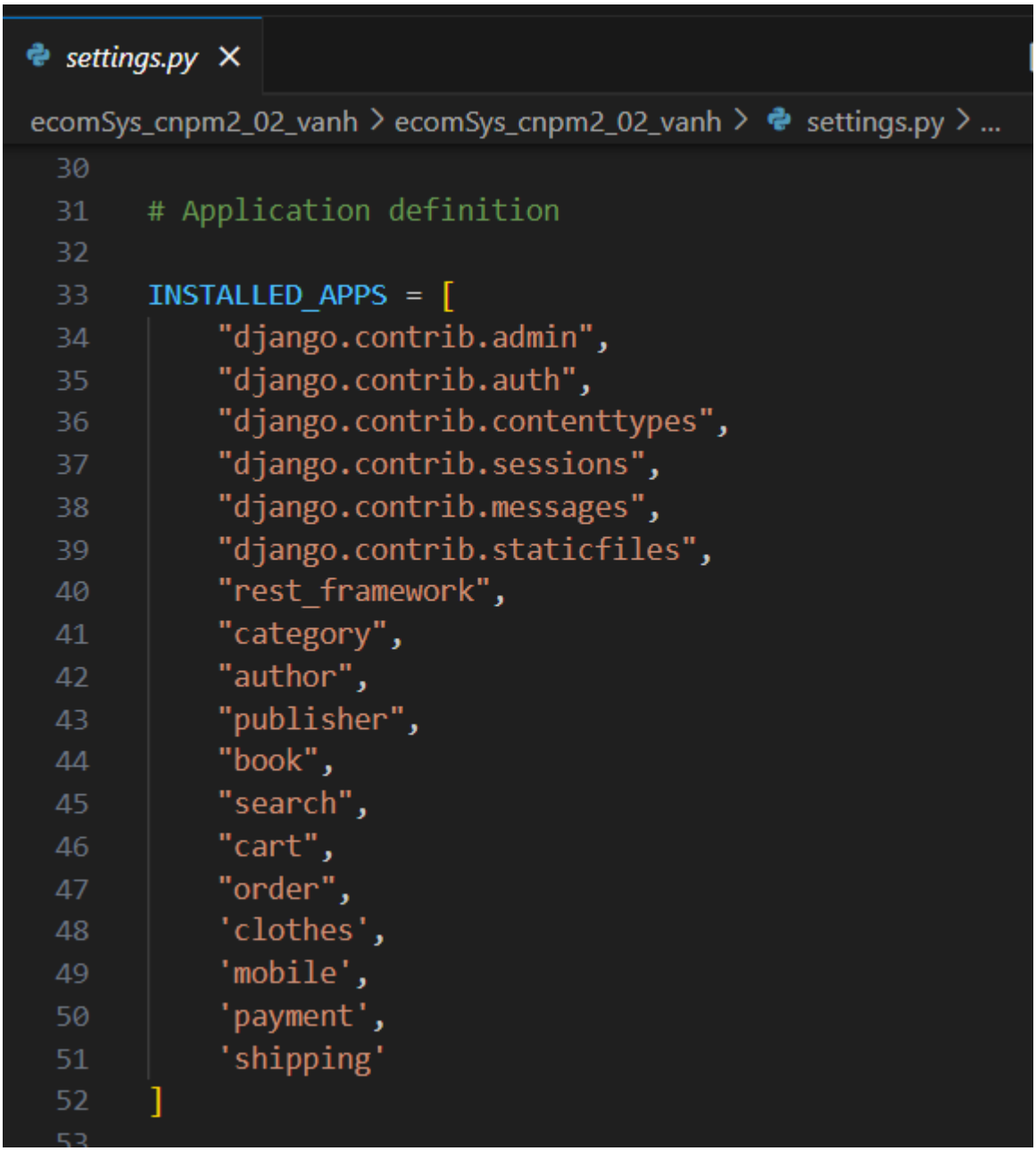
Ví dụ

```python
class CartViewSet(viewsets.ViewSet):

    def retrieve(self, request, pk=None):
        cart = get_object_or_404(Cart.objects.all(), pk=pk)
        cart_items = get_list_or_404(CartItem.objects.all(), cart=cart)
        cart_total = Decimal("0.00")
        for cart_item in cart_items:
            print(cart_item.product_id)
            if int(cart_item.category_id) == 1:
                response = requests.get(
                    f"http://localhost:8001/api/v1/books/{cart_item.product_id}"
                ).json()

                print(response)
                cart_total += cart_item.quantity * Decimal(response.get("price"))
            elif int(cart_item.category_id) == 4:
                response = requests.get(
                    f"http://localhost:8002/api/v1/clothes/{cart_item.product_id}"
                ).json()

                print(response)
                cart_total += cart_item.quantity * Decimal(response.get("price"))
            elif int(cart_item.category_id) == 5:
                response = requests.get(
                    f"http://localhost:8003/api/v1/mobiles/{cart_item.product_id}"
                ).json()

                print(response)
                cart_total += cart_item.quantity * Decimal(response.get("price"))

        response = {
            "cart": CartSerializer(cart).data,
            "items": CartItemSerializer(cart_items, many=True).data,
            "total": cart_total,
        }
```

```python
return Response(
    data=response,
    status=status.HTTP_200_OK,
    content_type="application/json",
)
```

II. Bài tập 4

1. **10 apps + design with MongoDB**

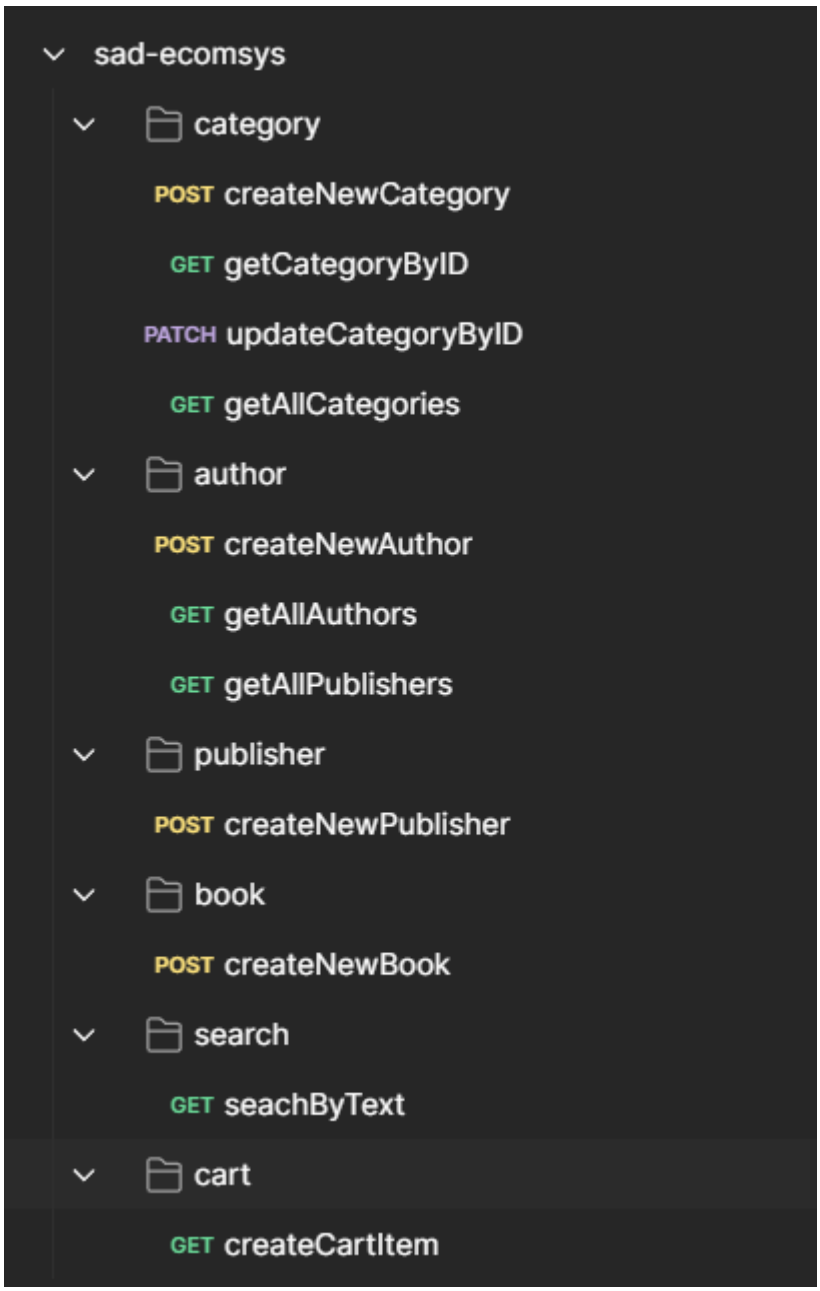


settings.py

ecomSys_cnpm2_02_vanh > ecomSys_cnpm2_02_vanh > settings.py > ...

```
85  # Database
86  # https://docs.djangoproject.com/en/4.1/ref/settings/#databases
87
88  DATABASES = {
89      # "default": {
90      #     "ENGINE": "django.db.backends.sqlite3",
91      #     "NAME": BASE_DIR / "db.sqlite3",
92      # }
93      "default": {
94          "ENGINE": "djongo",
95          "NAME": "ecomSys_cnpm2_02_vanh",
96          "ENFORCE_SCHEMA": False,
97          "CLIENT": {
98              "host": "mongodb://localhost:27017/",
99          },
100     }
101 }
102
```

2. Rest API


sad-ecomsys
category
POST createNewCategory
GET getCategoryByID
PATCH updateCategoryByID
GET getAllCategories
author
POST createNewAuthor
GET getAllAuthors
GET getAllPublishers
publisher
POST createNewPublisher
book
POST createNewBook
search
GET seachByText
cart
GET createCartItem

ecomSys_cnpm2_02_vanh > ecomSys_cnpm2_02_vanh > urls.py > ...

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path("admin/", admin.site.urls),
    path("api/", include("category.urls")),
    path("api/", include("author.urls")),
    path("api/", include("publisher.urls")),
    path("api/", include("book.urls")),
    path("api/", include("search.urls")),
    path('api/', include('cart.urls')),
    path('api/', include('order.urls')),
]

```

3. Apps

3.1. Category

ecomSys_cnpm2_02_vanh > category > models.py > ...

```python
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField(blank=True, null=True)

    def __str__(self):
        return self.name

```

ecomSys_cnpm2_02_vanh > category > serializers.py > ...

```python
from rest_framework import serializers
from .models import Category

class CategorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Category
        fields = '__all__'
```

ecomSys_cnpm2_02_vanh > category > views.py > ...

```python
from rest_framework import generics
from .models import Category
from .serializers import CategorySerializer

class CategoryListAPIView(generics.ListCreateAPIView):
    queryset = Category.objects.all()
    serializer_class = CategorySerializer

class CategoryDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Category.objects.all()
    serializer_class = CategorySerializer
```

ecomSys_cnpm2_02_vanh > category > urls.py > ...

```python
from django.urls import path
from .views import CategoryListAPIView, CategoryDetailAPIView

urlpatterns = [
    path('categories/', CategoryListAPIView.as_view(), name='category-list'),
    path('categories/<int:pk>/', CategoryDetailAPIView.as_view(), name='category-detail'),
]
```

3.2. Author

ecomSys_cnpm2_02_vanh > author > models.py > ...

```python
from django.db import models

class Author(models.Model):
    name = models.CharField(max_length=100)
    bio = models.TextField(blank=True, null=True)
    birth_date = models.DateField(blank=True, null=True)

    def __str__(self):
        return self.name

```

ecomSys_cnpm2_02_vanh > author > serializers.py > ...

```python
from rest_framework import serializers
from .models import Author

class AuthorSerializer(serializers.ModelSerializer):
    class Meta:
        model = Author
        fields = '__all__'

```

ecomSys_cnpm2_02_vanh > author > views.py > ...

```python
from rest_framework import generics
from .models import Author
from .serializers import AuthorSerializer

class AuthorListAPIView(generics.ListCreateAPIView):
    queryset = Author.objects.all()
    serializer_class = AuthorSerializer

class AuthorDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Author.objects.all()
    serializer_class = AuthorSerializer

```

ecomSys_cnpm2_02_vanh > author > urls.py > ...

```python
from django.urls import path
from .views import AuthorListAPIView, AuthorDetailAPIView

urlpatterns = [
    path('authors/', AuthorListAPIView.as_view(), name='author-list'),
    path('authors/<int:pk>/', AuthorDetailAPIView.as_view(), name='author-detail'),
]

```

3.3. Publisher

ecomSys_cnpm2_02_vanh > publisher > models.py > ...

```python
from django.db import models

class Publisher(models.Model):
    name = models.CharField(max_length=100)
    website = models.URLField(blank=True)

    def __str__(self):
        return self.name

```

ecomSys_cnpm2_02_vanh > publisher > serializers.py > ...

```python
from rest_framework import serializers
from .models import Publisher

class PublisherSerializer(serializers.ModelSerializer):
    class Meta:
        model = Publisher
        fields = '__all__'

```

ecomSys_cnpm2_02_vanh > publisher > views.py > ...

```python
from rest_framework import generics
from .models import Publisher
from .serializers import PublisherSerializer

class PublisherListAPIView(generics.ListCreateAPIView):
    queryset = Publisher.objects.all()
    serializer_class = PublisherSerializer

class PublisherDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Publisher.objects.all()
    serializer_class = PublisherSerializer

```

ecomSys_cnpm2_02_vanh > publisher > urls.py > ...

```python
from django.urls import path
from .views import PublisherListAPIView, PublisherDetailAPIView

urlpatterns = [
    path('publishers/', PublisherListAPIView.as_view(), name='publisher-list'),
    path('publishers/<int:pk>/', PublisherDetailAPIView.as_view(), name='publisher-detail'),
]

```

3.4. Book

ecomSys_cnpm2_02_vanh > book > models.py > ...

```python
from django.db import models
from category.models import Category
from author.models import Author
from publisher.models import Publisher

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    publisher = models.ForeignKey(Publisher, on_delete=models.CASCADE)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    description = models.TextField(blank=True, null=True)

    def __str__(self):
        return self.title

```

ecomSys_cnpm2_02_vanh > book > serializers.py > ...

```python
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
    class Meta:
        model = Book
        fields = '__all__'

```

ecomSys_cnpm2_02_vanh > book > views.py > ...

```python
from rest_framework import generics
from .models import Book
from .serializers import BookSerializer

class BookListAPIView(generics.ListCreateAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer

class BookDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer

```

ecomSys_cnpm2_02_vanh > book > urls.py > ...

```python
from django.urls import path
from .views import BookListAPIView, BookDetailAPIView

urlpatterns = [
    path('books/', BookListAPIView.as_view(), name='book-list'),
    path('books/<int:pk>/', BookDetailAPIView.as_view(), name='book-detail')
]

```

3.5. Search

ecomSys_cnpm2_02_vanh > search > views.py > ...

```python
from rest_framework.views import APIView
from rest_framework.response import Response
from book.models import Book
from book.serializers import BookSerializer

# http://localhost:8000/api/search/?query=harry

class BookSearchAPIView(APIView):
    def get(self, request, format=None):
        query = request.query_params.get("query", "")
        books = Book.objects.filter(title__icontains=query)
        serializer = BookSerializer(books, many=True)
        return Response(serializer.data)

```

ecomSys_cnpm2_02_vanh > search > urls.py > ...

```python
from django.urls import path
from .views import BookSearchAPIView

urlpatterns = [
    path('search/', BookSearchAPIView.as_view(), name='book-search'),
]

```

3.6. Cart

ecomSys_cnpm2_02_vanh > cart > models.py > ...

```python
from django.db import models
from book.models import Book

class Cart(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)

    def __str__(self):
        return f"{self.quantity} x {self.book.title}"

```

ecomSys_cnpm2_02_vanh > cart > serializers.py > ...

```python
from rest_framework import serializers
from .models import Cart
from book.models import Book

class CartSerializer(serializers.ModelSerializer):
    book_id = serializers.PrimaryKeyRelatedField(
        queryset=Book.objects.all(), source="book", write_only=True
    )
    bookID = serializers.PrimaryKeyRelatedField(source="book.id", read_only=True)

    def validate_book_id(self, value):
        try:
            Book.objects.get(pk=value)
        except Book.DoesNotExist:
            raise serializers.ValidationError("Book does not exist.")
        return value

    class Meta:
        model = Cart
        fields = ["id", "book_id", "bookID", "quantity"]

```

ecomSys_cnpm2_02_vanh > cart > views.py > ...

```python
from rest_framework import generics
from .models import Cart
from .serializers import CartSerializer

class CartListCreateAPIView(generics.ListCreateAPIView):
    queryset = Cart.objects.all()
    serializer_class = CartSerializer

class CartRetrieveUpdateDestroyAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Cart.objects.all()
    serializer_class = CartSerializer

```

ecomSys_cnpm2_02_vanh > cart > urls.py > ...

```python
from django.urls import path
from .views import CartListCreateAPIView, CartRetrieveUpdateDestroyAPIView

urlpatterns = [
    path('carts/', CartListCreateAPIView.as_view(), name='cart-list-create'),
    path('carts/<int:pk>/', CartRetrieveUpdateDestroyAPIView.as_view(), name='cart-detail'),
]

```

3.7. Order

ecomSys_cnpm2_02_vanh > order > models.py > ...

```python
from django.db import models
from cart.models import Cart

class Order(models.Model):
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    carts = models.ManyToManyField(Cart)

    def __str__(self):
        return f"Order #{self.pk}"

```

ecomSys_cnpm2_02_vanh > order > serializers.py > ...

```python
from rest_framework import serializers
from .models import Order
from cart.models import Cart


class OrderSerializer(serializers.ModelSerializer):
    cart_items = serializers.PrimaryKeyRelatedField(
        queryset=Cart.objects.all(), many=True, source="carts", write_only=True
    )

    class Meta:
        model = Order
        fields = ["id", "created_at", "updated_at", "cart_items"]

    def create(self, validated_data):
        cart_items_data = validated_data.pop("carts", [])
        order = Order.objects.create(**validated_data)

        # Duyệt qua từng Cart và lưu thông tin vào Order
        for cart_item_data in cart_items_data:
            order.carts.add(cart_item_data.id)

        return order

```

ecomSys_cnpm2_02_vanh > order > views.py > ...

```python
from rest_framework import generics
from .models import Order
from .serializers import OrderSerializer

class OrderListCreateAPIView(generics.ListCreateAPIView):
    queryset = Order.objects.all()
    serializer_class = OrderSerializer

class OrderRetrieveUpdateDestroyAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Order.objects.all()
    serializer_class = OrderSerializer

```