Họ tên: Trần Vân Anh MSV: B20DCCN075

I. Bài tập 4

1. 10 apps + design with MongoDB

```
🕏 settings.py 🗙
ecomSys_cnpm2_02_vanh > ecomSys_cnpm2_02_vanh > 📌 settings.py > ...
       # Application definition
 31
 32
       INSTALLED APPS = [
            "django.contrib.admin",
            "django.contrib.auth",
            "django.contrib.contenttypes",
            "django.contrib.sessions",
            "django.contrib.messages",
            "django.contrib.staticfiles",
            "rest_framework",
            "category",
 41
            "author",
 42
            "publisher",
            "book",
           "search",
 45
            "cart",
            "order",
 47
            'clothes',
            'mobile',
            'payment',
            'shipping'
 52
```

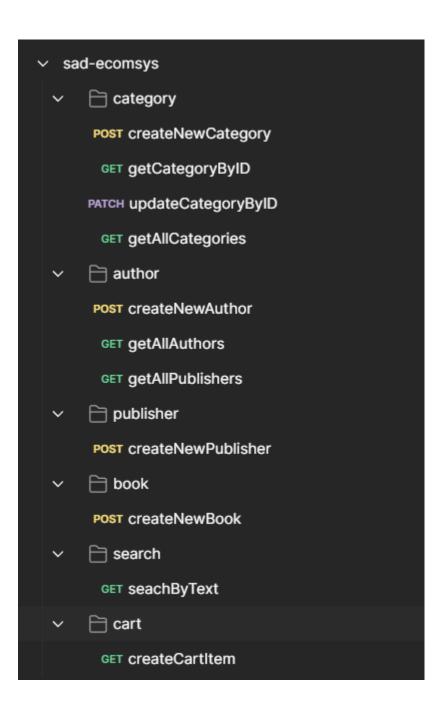
```
ecomSys_cnpm2_02_vanh > ecomSys_cnpm2_02_vanh > esettings.py > ...

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 }
```

2. Rest API



```
ecomSys_cnpm2_02_vanh > ecomSys_cnpm2_02_vanh > 🕏 urls.py > ...
       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')),
 11
           path('api/', include('order.urls')),
 12
 13
 14
```

3. Apps

3.1. Category

```
ecomSys_cnpm2_02_vanh > category >  serializers.py > ...

1    from rest_framework import serializers
2    from .models import Category
3
4    class CategorySerializer(serializers.ModelSerializer):
5         class Meta:
6         model = Category
7         fields = '__all__'
8
```

```
ecomSys_cnpm2_02_vanh > category >  views.py > ...
    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

serializer_class = CategorySerializer

12
```

```
ecomSys_cnpm2_02_vanh > category > → urls.py > ...

1 from django.urls import path
2 from .views import CategoryListAPIView, CategoryDetailAPIView

3 urlpatterns = [
5 path('categories/', CategoryListAPIView.as_view(), name='category-list'),
6 path('categories/<int:pk>/', CategoryDetailAPIView.as_view(), name='category-detail'),
7
8
```

3.2. Author

```
ecomSys_cnpm2_02_vanh > author > views.py > ...

1    from rest_framework import generics
2    from .models import Author
3    from .serializers import AuthorSerializer

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

8    class AuthorDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
10         queryset = Author.objects.all()
11         serializer_class = AuthorSerializer
12
```

```
ecomSys_cnpm2_02_vanh > author >  urls.py > ...

1   from django.urls import path
2   from .views import AuthorListAPIView, AuthorDetailAPIView

3   urlpatterns = [
5     path('authors/', AuthorListAPIView.as_view(), name='author-list'),
6     path('authors/<int:pk>/', AuthorDetailAPIView.as_view(), name='author-detail'),
7   ]
8
```

3.3. Publisher

```
ecomSys_cnpm2_02_vanh > publisher >  serializers.py > ...

1    from rest_framework import serializers

2    from .models import Publisher

3    class PublisherSerializer(serializers.ModelSerializer):

5    class Meta:

6    model = Publisher

7    fields = '__all__'

8
```

```
ecomSys_cnpm2_02_vanh > publisher >  views.py > ...

1    from rest_framework import generics
2    from .models import Publisher
3    from .serializers import PublisherSerializer
4
5    class PublisherListAPIView(generics.ListCreateAPIView):
6         queryset = Publisher.objects.all()
7         serializer_class = PublisherSerializer
8
9    class PublisherDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
10         queryset = Publisher.objects.all()
11         serializer_class = PublisherSerializer
12
```

```
ecomSys_cnpm2_02_vanh > publisher >  urls.py > ...

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'),

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

path('publishers/<int:pk>/', PublisherDetailAPIView.as_view(), name='publisher-detail'),
```

3.4. Book

```
ecomSys_cnpm2_02_vanh > book > views.py > ...

1    from rest_framework import generics
2    from .models import Book
3    from .serializers import BookSerializer
4
5    class BookListAPIView(generics.ListCreateAPIView):
6         queryset = Book.objects.all()
7         serializer_class = BookSerializer
8
9    class BookDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
10    queryset = Book.objects.all()
11    serializer_class = BookSerializer
12
```

```
ecomSys_cnpm2_02_vanh > book >  urls.py > ...

1   from django.urls import path
2   from .views import BookListAPIView, BookDetailAPIView

3   urlpatterns = [
5     path('books/', BookListAPIView.as_view(), name='book-list'),
6     path('books/<int:pk>/', BookDetailAPIView.as_view(), name='book-detail')
7   ]
8
```

3.5. Search

```
ecomSys_cnpm2_02_vanh > search > views.py > ...

1     from rest_framework.views import APIView
2     from rest_framework.response import Response
3     from book.models import Book
4     from book.serializers import BookSerializer
5
6     # http://localhost:8000/api/search/?query=harry
7
8     class BookSearchAPIView(APIView):
9     def get(self, request, format=None):
10         query = request.query_params.get("query", "")
11          books = Book.objects.filter(title__icontains=query)
12          serializer = BookSerializer(books, many=True)
13          return Response(serializer.data)
14
```

3.6. Cart

```
ecomSys_cnpm2_02_vanh > cart > 🕏 serializers.py > ...
       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"]
 21
```

```
ecomSys_cnpm2_02_vanh > cart > ♥ views.py > ...

1     from rest_framework import generics

2     from .models import Cart

3     from .serializers import CartSerializer

4     class CartListCreateAPIView(generics.ListCreateAPIView):

6     queryset = Cart.objects.all()

7     serializer_class = CartSerializer

8     class CartRetrieveUpdateDestroyAPIView(generics.RetrieveUpdateDestroyAPIView):

10     queryset = Cart.objects.all()

11     serializer_class = CartSerializer

12
```

```
ecomSys_cnpm2_02_vanh > cart >  urls.py > ...

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'),

path('carts/<int:pk>/', CartRetrieveUpdateDestroyAPIView.as_view(), name='cart-detail'),

}
```

3.7. Order

```
ecomSys_cnpm2_02_vanh > order > 🕏 serializers.py > ...
      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)
              for cart item data in cart items data:
                   order.carts.add(cart item data.id)
              return order
 24
```

```
ecomSys_cnpm2_02_vanh > order > views.py > ...

1     from rest_framework import generics
2     from .models import Order
3     from .serializers import OrderSerializer
4
5     class OrderListCreateAPIView(generics.ListCreateAPIView):
6         queryset = Order.objects.all()
7         serializer_class = OrderSerializer

9     class OrderRetrieveUpdateDestroyAPIView(generics.RetrieveUpdateDestroyAPIView):
10         queryset = Order.objects.all()
11         serializer_class = OrderSerializer
12
```

II. Câu 1-5

Functionalities

	T . 1*.*	D. T.ID. C.
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
	Thâm Vào Ciả	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	Đăng nhập vào hệ thống.
	(Customer	
	Login)	
	Thay Đổi Mật	Thay đổi mật khẩu tài khoản.
	Khẩu (Change	
	Password)	
Admin	Quản lý sách	Có thể thêm, chỉnh sửa, xem và xóa sách.
	(Manage Book)	
	Quản lý đơn	Có thể thêm đơn hàng mới, xem danh sách đơn
	hàng (Manage	hàng, chỉnh sửa và xóa đơn hàng.
	Order)	
	Quản lý khách	Có thể thêm bản ghi khách hàng mới, xem danh
	hàng (Manage	sách khách hàng, chỉnh sửa và xóa bản ghi khách
	Customer)	hàng.
	Quản lý kho	Quản lý toàn bộ hàng tồn kho, bao gồm việc
	(Manage Stock)	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

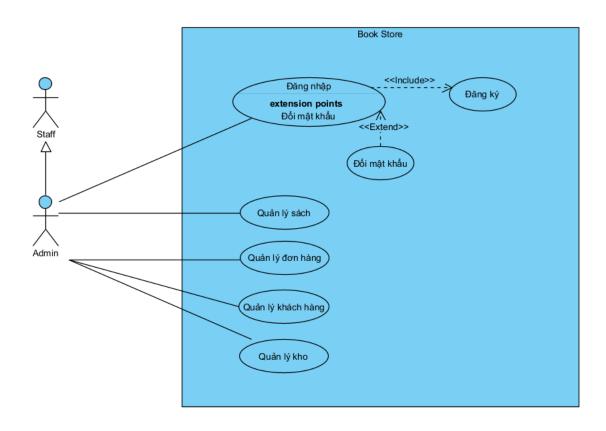
Giỏ hàng

Vận chuyển

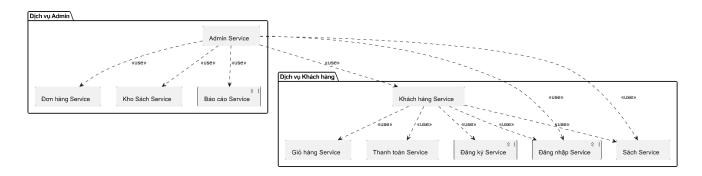
Thanh toán

Đơn hàng của Khách hàng

Thay đổi Mật khẩu



2. Biểu đồ phân rã Hệ ecomSys



3.

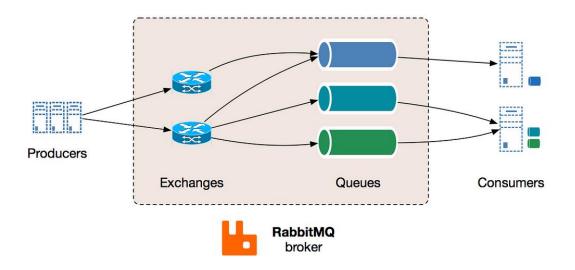
Giao tiếp đồng bộ:

Xử lý dữ liệu

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()
```

```
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:

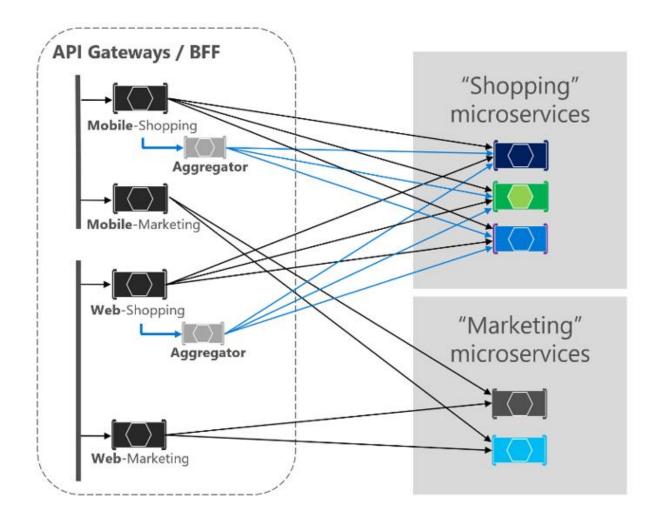
- Giao tiếp không đồng bộ làm giảm khớp nối giữa các dịch vụ
- Có thể xử lý khối lượng lớn tin nhắn và hỗ trợ cân bằng tải
- Hỗ trợ các mẫu nhắn tin khác nhau như xuất bản-đăng ký, điểm-điểm và yêu cầu-trả lời

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



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 / $2\,$