

1. Создали два проекта

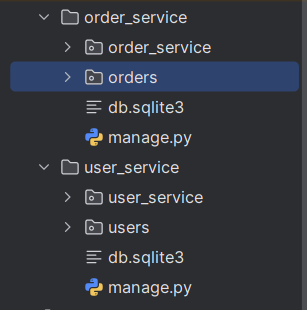
django-admin startproject user\_service

django-admin startproject order\_service

В каждом проекте создали приложение:

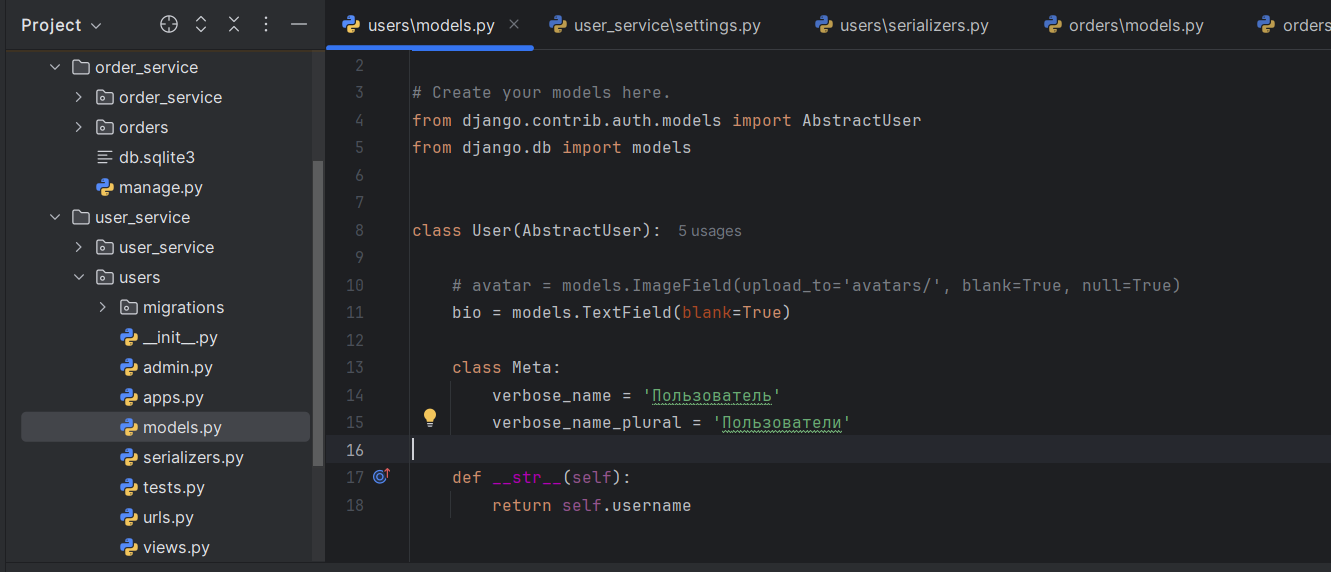
cd user\_service && python manage.py startapp users

cd order\_service && python manage.py startapp orders

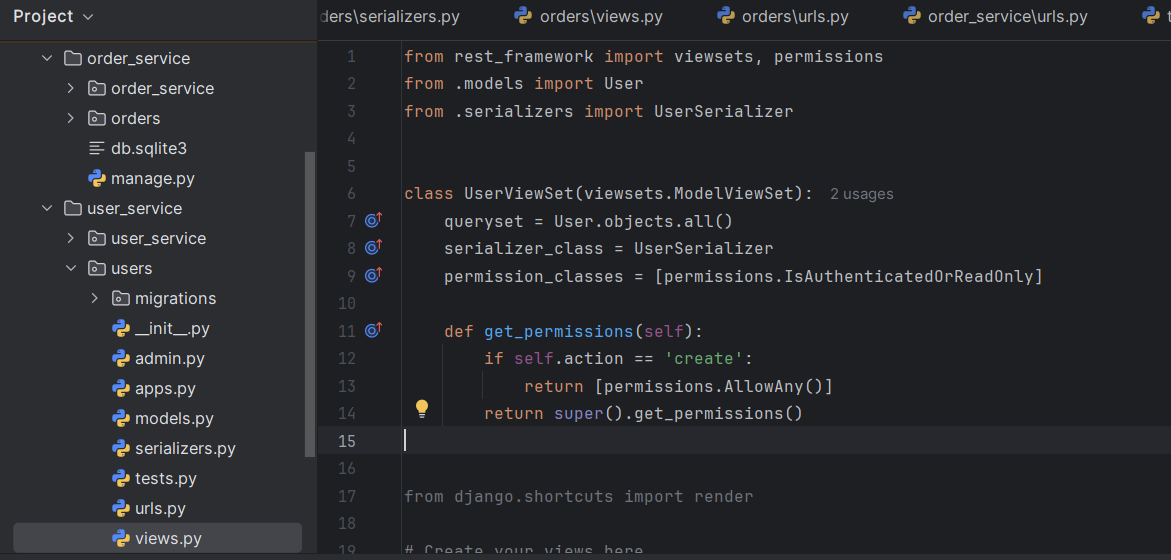


2. Настройка User Service

1) В users/models.py определите модель пользователя:



2) Настройте DRF (Django REST Framework) и создайте API для CRUD операций.



3. Настройка Order Service

1) В orders/models.py определите модель заказа:

from django.db import models  
from django.conf import settings  
  
class Order(models.Model):  
 STATUS\_CHOICES = [  
 ('PENDING', 'Pending'),  
 ('PROCESSING', 'Processing'),  
 ('COMPLETED', 'Completed'),  
 ('CANCELLED', 'Cancelled'),  
 ]  
  
 user\_id = models.PositiveIntegerField() # ID пользователя из User Service  
 product\_name = models.CharField(max\_length=255)  
 quantity = models.PositiveIntegerField()  
 total\_price = models.DecimalField(max\_digits=10, decimal\_places=2)  
 status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='PENDING')  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
 updated\_at = models.DateTimeField(auto\_now=True)  
  
 class Meta:  
 ordering = ['-created\_at']  
  
 def \_\_str\_\_(self):  
 return f"Order #{self.id} - {self.product\_name}"  
  
# Create your models here.

2) Реализуйте API для создания заказа с проверкой пользователя в User Service:

import requests  
from rest\_framework import viewsets, status  
from rest\_framework.response import Response  
from rest\_framework.permissions import IsAuthenticated, AllowAny  
from .models import Order  
from .serializers import OrderSerializer  
  
USER\_SERVICE\_URL = 'http://localhost:8000/api/users/'  
  
  
class OrderViewSet(viewsets.ModelViewSet):  
 queryset = Order.objects.all()  
 serializer\_class = OrderSerializer  
 permission\_classes = [AllowAny]  
  
 def create(self, request, \*args, \*\*kwargs):  
 # Проверяем существование пользователя в User Service  
 user\_id = request.data.get('user\_id')  
 try:  
 response = requests.get(f"{USER\_SERVICE\_URL}{user\_id}/", timeout=5)  
 if response.status\_code != 200:  
 return Response(  
 {'error': 'User not found'},  
 status=status.HTTP\_400\_BAD\_REQUEST  
 )  
 except requests.exceptions.RequestException:  
 return Response(  
 {'error': 'User service unavailable'},  
 status=status.HTTP\_503\_SERVICE\_UNAVAILABLE  
 )  
  
 return super().create(request, \*args, \*\*kwargs)  
  
 def get\_queryset(self):  
 *"""Фильтрация заказов по текущему пользователю"""* return self.queryset.filter(user\_id=self.request.user.id)

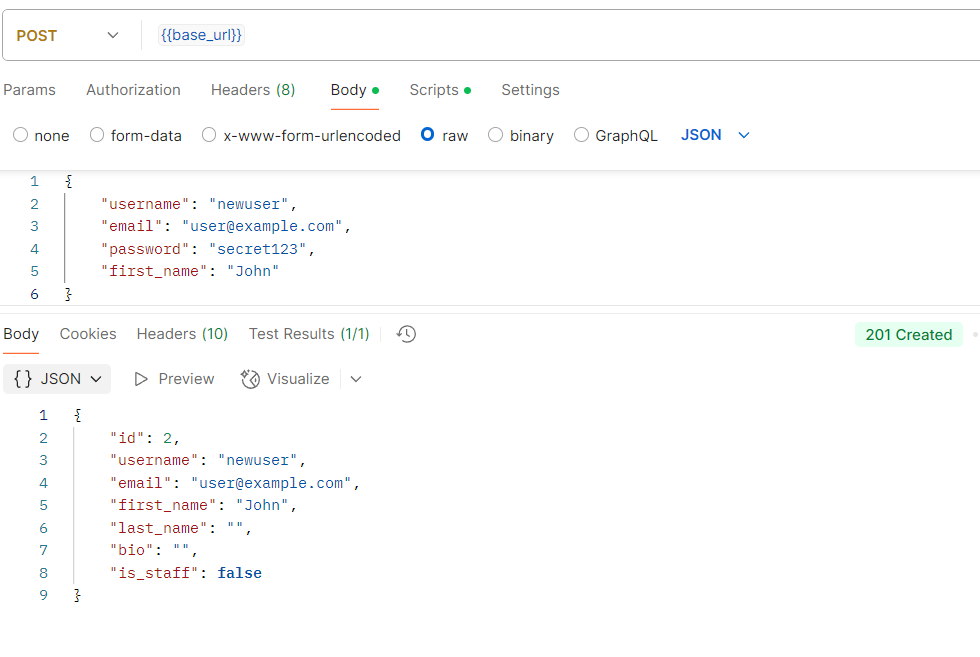
4. Настройка CORS

Установите django-cors-headers и разрешите запросы между сервисами:

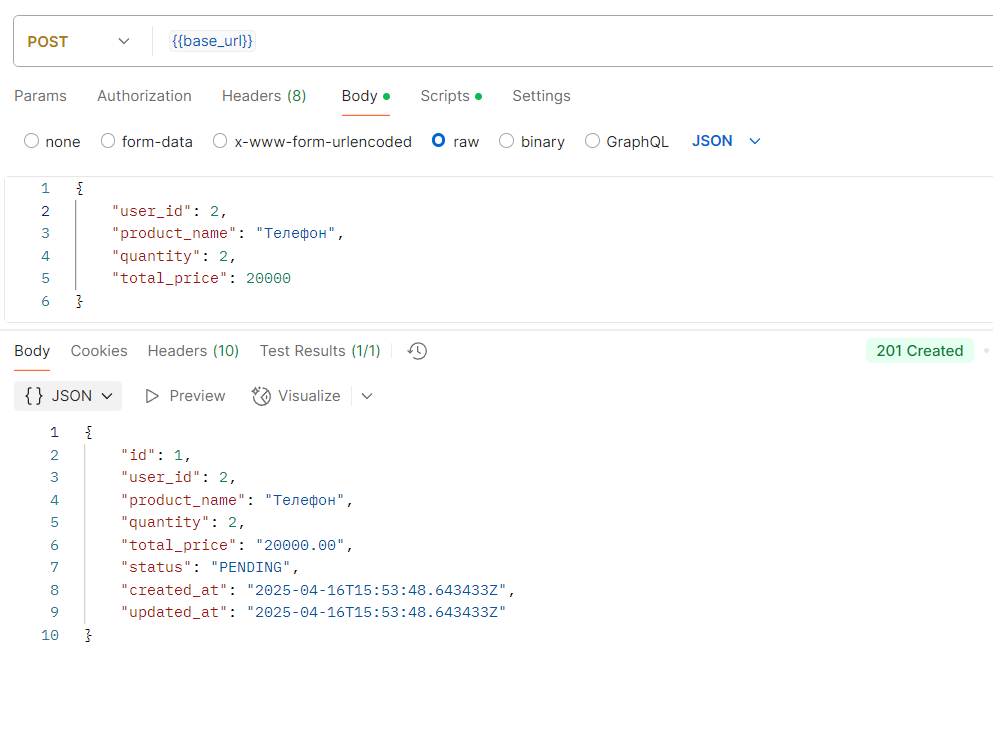
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
  
 'corsheaders.middleware.CorsMiddleware',#cors  
 'django.middleware.common.CommonMiddleware',#cors  
  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
CORS\_ALLOW\_ALL\_ORIGINS = True

5. Тестирование

1 Создали пользователя



2 Создали заказ



3) Сделали интеграционные тесты с мокированием другого сервиса для теста

# orders/tests.py  
from django.urls import reverse  
from rest\_framework.test import APITestCase  
from unittest.mock import patch  
import requests  
  
class OrderAPITestCase(APITestCase):  
 @patch('requests.get')  
 def test\_create\_order\_with\_mocked\_user(self, mock\_get):  
 # Мокируем ответ User Service  
 mock\_get.return\_value.status\_code = 200  
 mock\_get.return\_value.json.return\_value = {  
 "id": 2,  
 "username": "newuser"  
 }  
  
 # Данные для заказа  
 data = {  
 "user\_id": 2,  
 "product\_name": "Ноутбук",  
 "quantity": 1,  
 "total\_price": 50000  
 }  
  
 # Отправляем запрос  
 url = reverse('order-list')  
 response = self.client.post(url, data, format='json')  
  
 # Проверяем результат  
 self.assertEqual(response.status\_code, 201)  
 self.assertEqual(response.data['product\_name'], "Ноутбук")  
  
 @patch('requests.get')  
 def test\_create\_order\_with\_invalid\_user(self, mock\_get):  
 # Мокируем ошибку 404 от User Service  
 mock\_get.return\_value.status\_code = 404  
  
 response = self.client.post(  
 reverse('order-list'),  
 {"user\_id": 999, "product\_name": "Несуществующий пользователь"},  
 format='json'  
 )  
 self.assertEqual(response.status\_code, 400)  
 self.assertIn("User not found", str(response.data))  
# Create your tests here.

