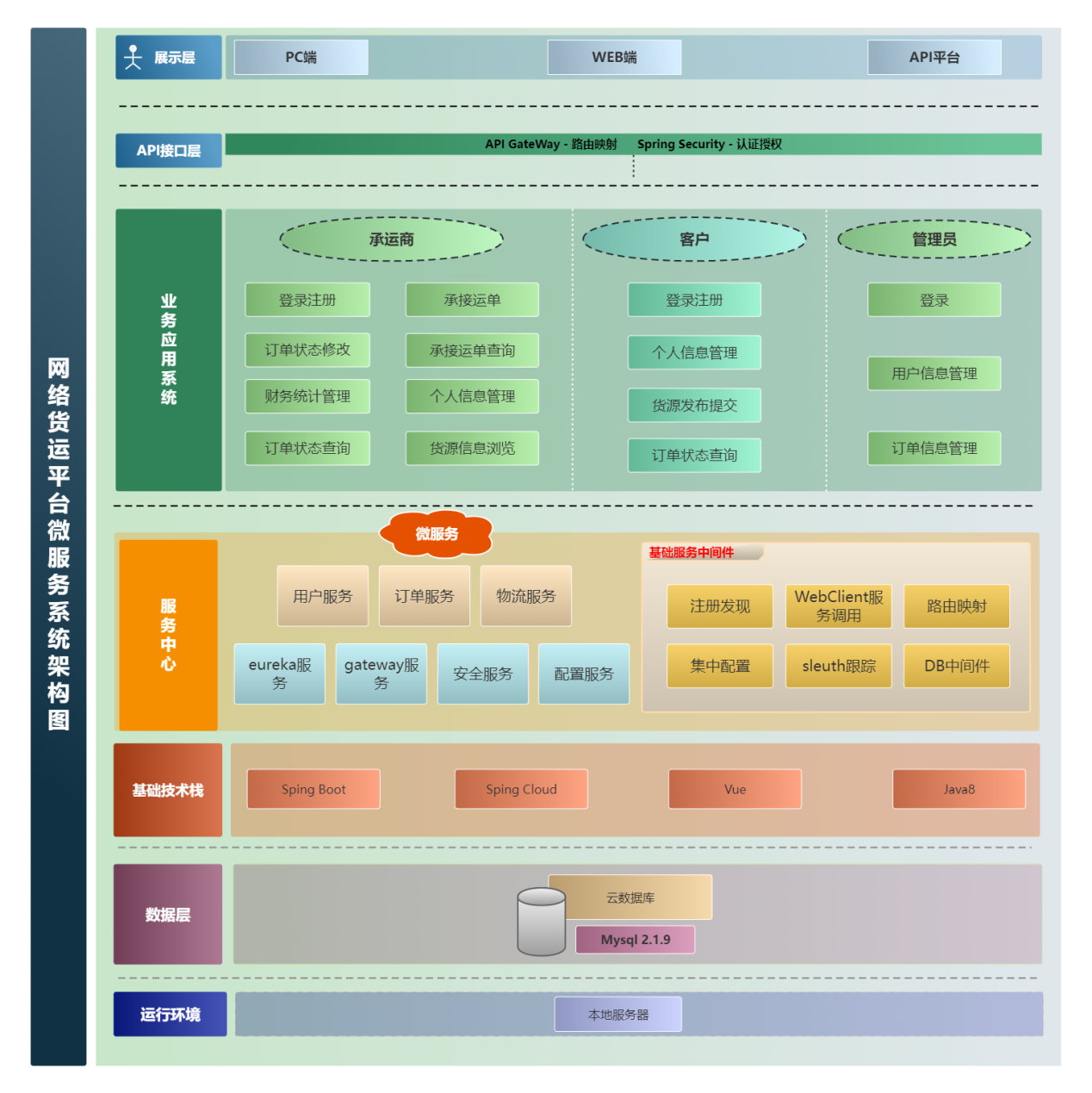
**Spring Cloud微服务版系统设计报告（作业3、4）**

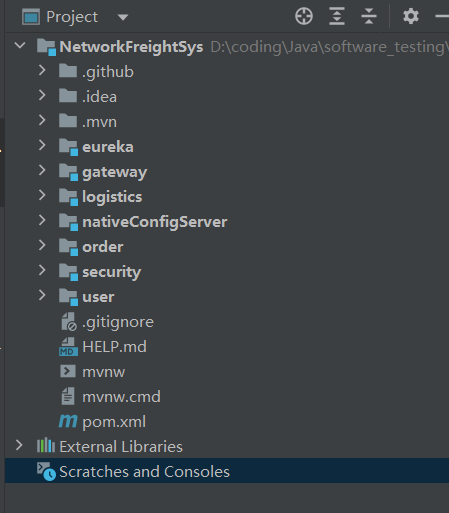
1. 技术架构图

****

1. 微服务架构说明

后端微服务共分为七大部分，如下图所示，现做如下阐述：

1. 用户user服务：包含登录注册、客户信息管理、承运商信息管理功能；
2. 订单order服务：包含对订单的修改、查询、订单财务管理等功能；
3. 物流logistics服务：包含物流录入、查询等功能；
4. eureka服务：实现服务的注册与发现；
5. gateway服务：实现路由映射，外部用户可通过gateway通过服务调用到内部接口；
6. security服务：授权与认证等安全功能；
7. nativeConfigServer配置服务：实现集中配置。



1. eureka服务注册与发现+运行步骤与环境

1.项目运行环境为java jdk11

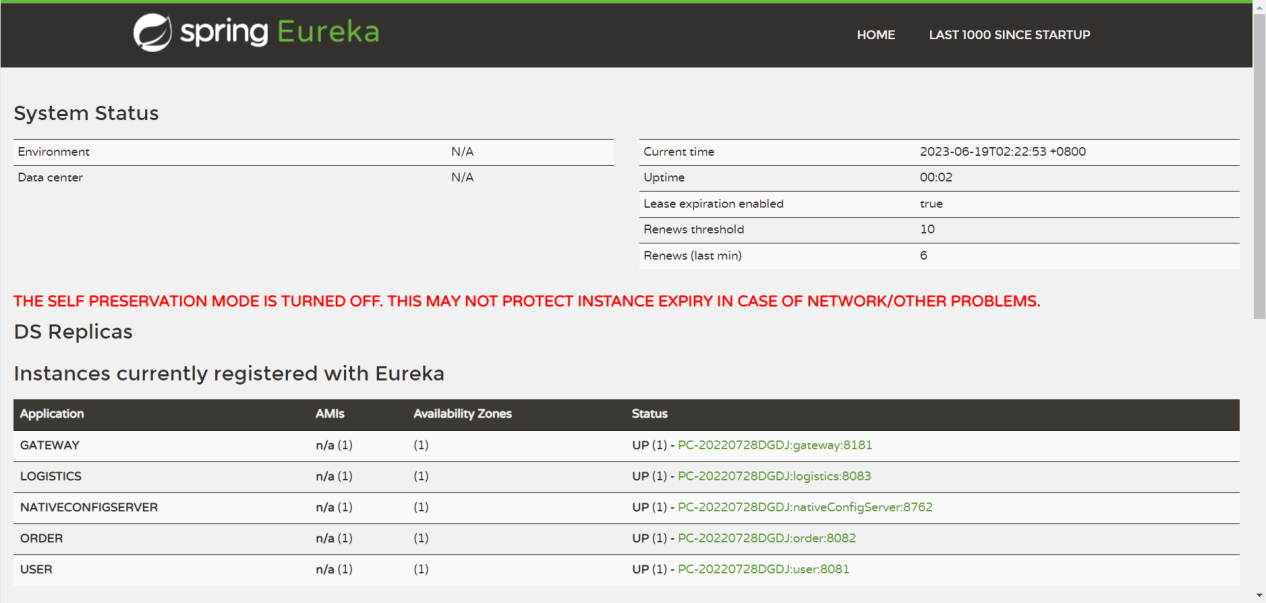
2.后端微服务项目的运行步骤为：

（1）**依次**运行eureka、nativeConfigServer、gateway

（2）再运行user、logistics、order

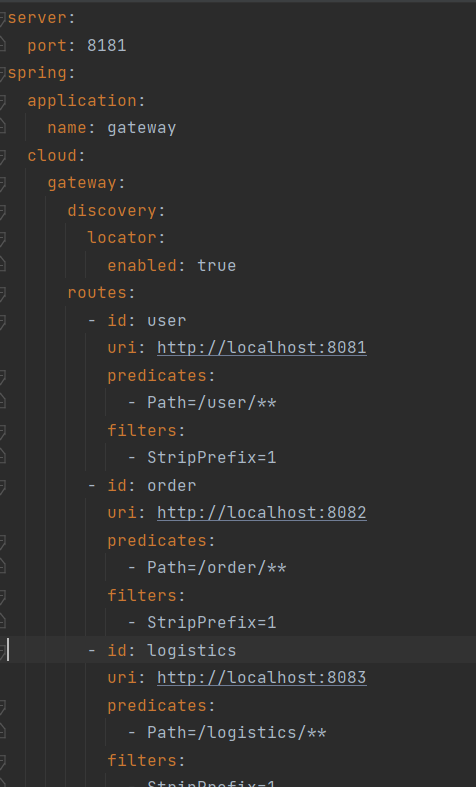
3.登录网址：<http://localhost:8761/>

可以看到eureka中被注册的服务：

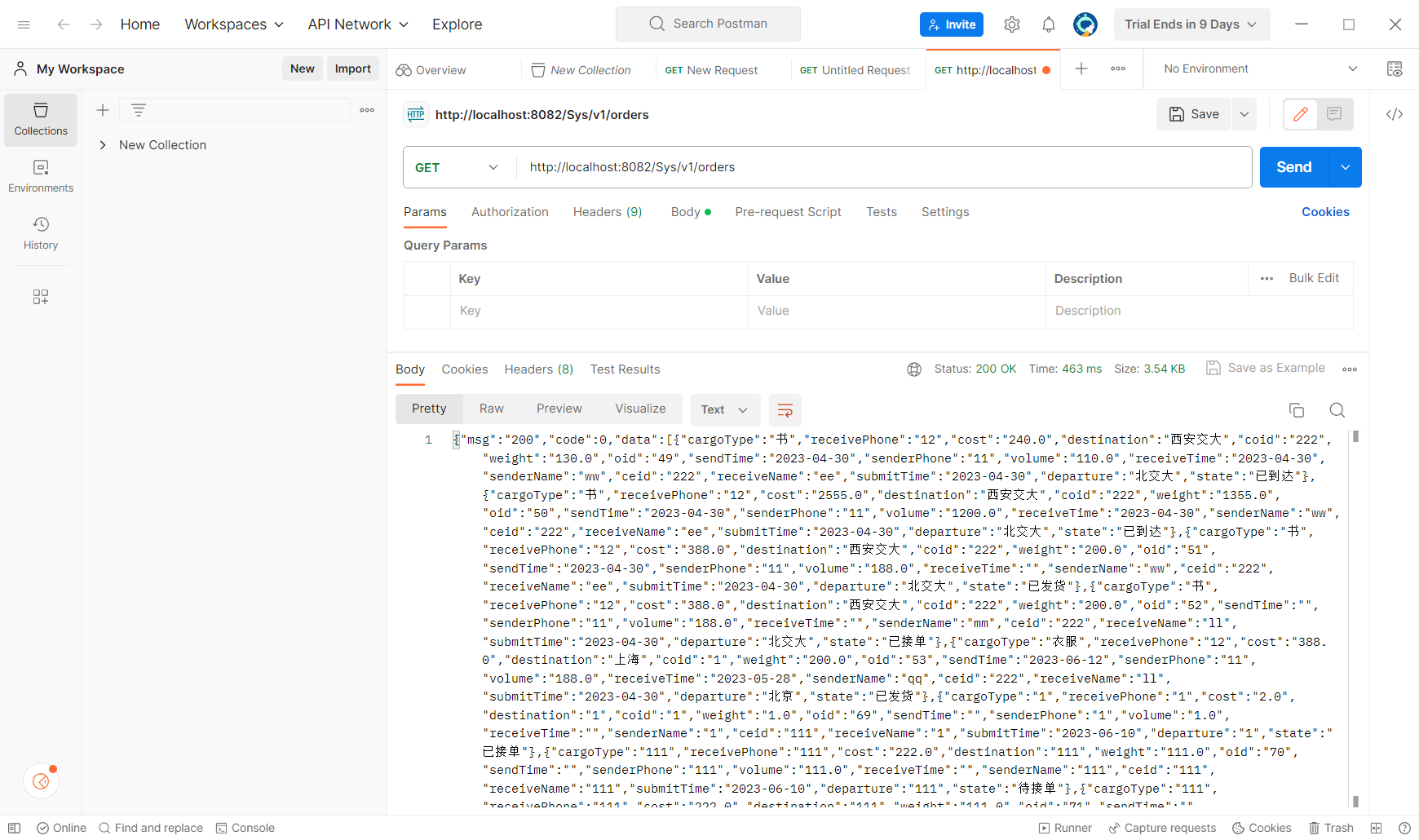


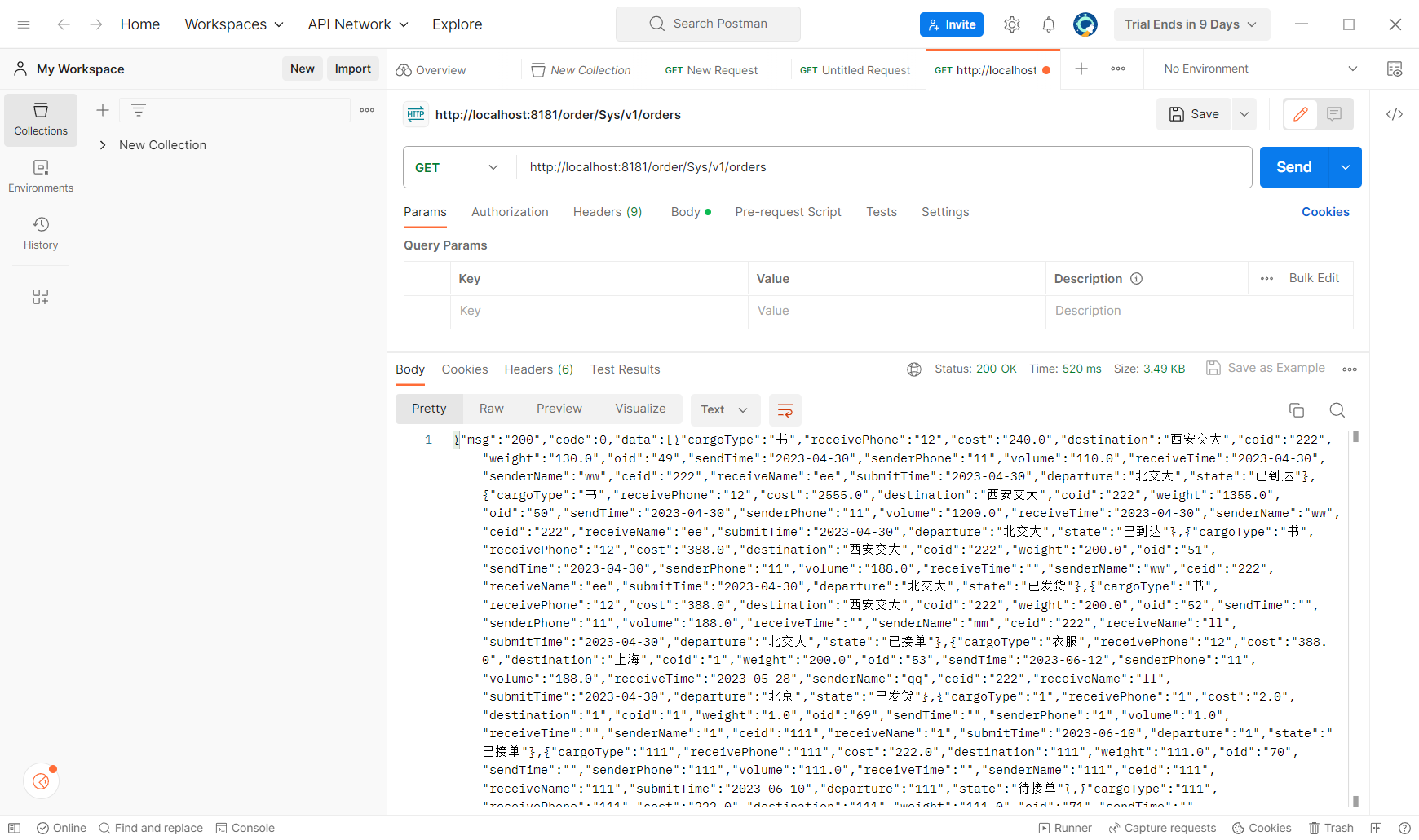
1. gateway

gateway实现路由映射，外部用户可通过服务调用到内部接口。通过服务调用到内部接口可以减少客户端记录不同端口号的数量，直接通过服务对接口进行访问。



比如order服务中的API接口URL原本为http://localhost:8082/Sys/v1/orders ，现可通过http://localhost:8181/order/Sys/v1/orders直接访问：

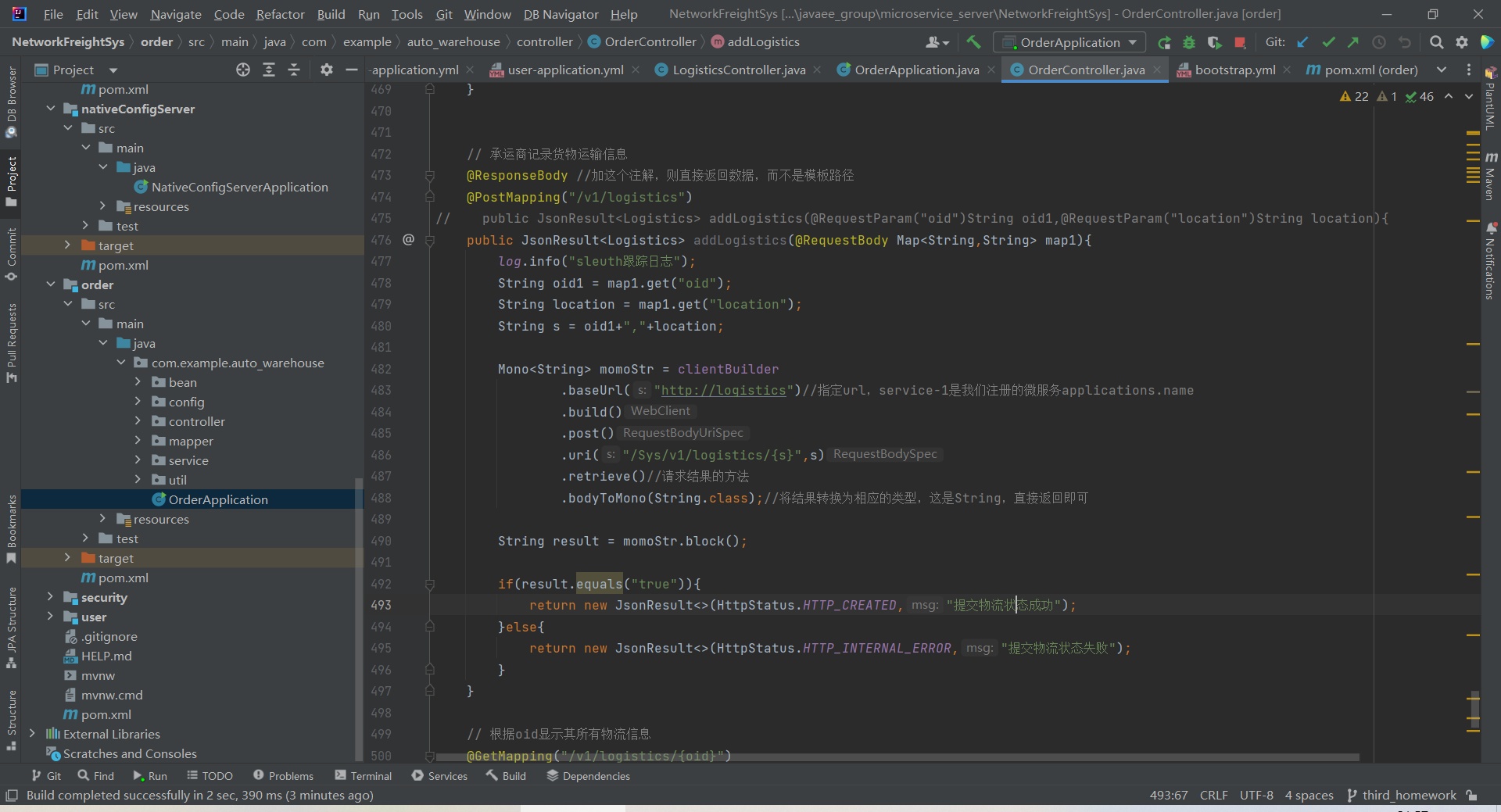


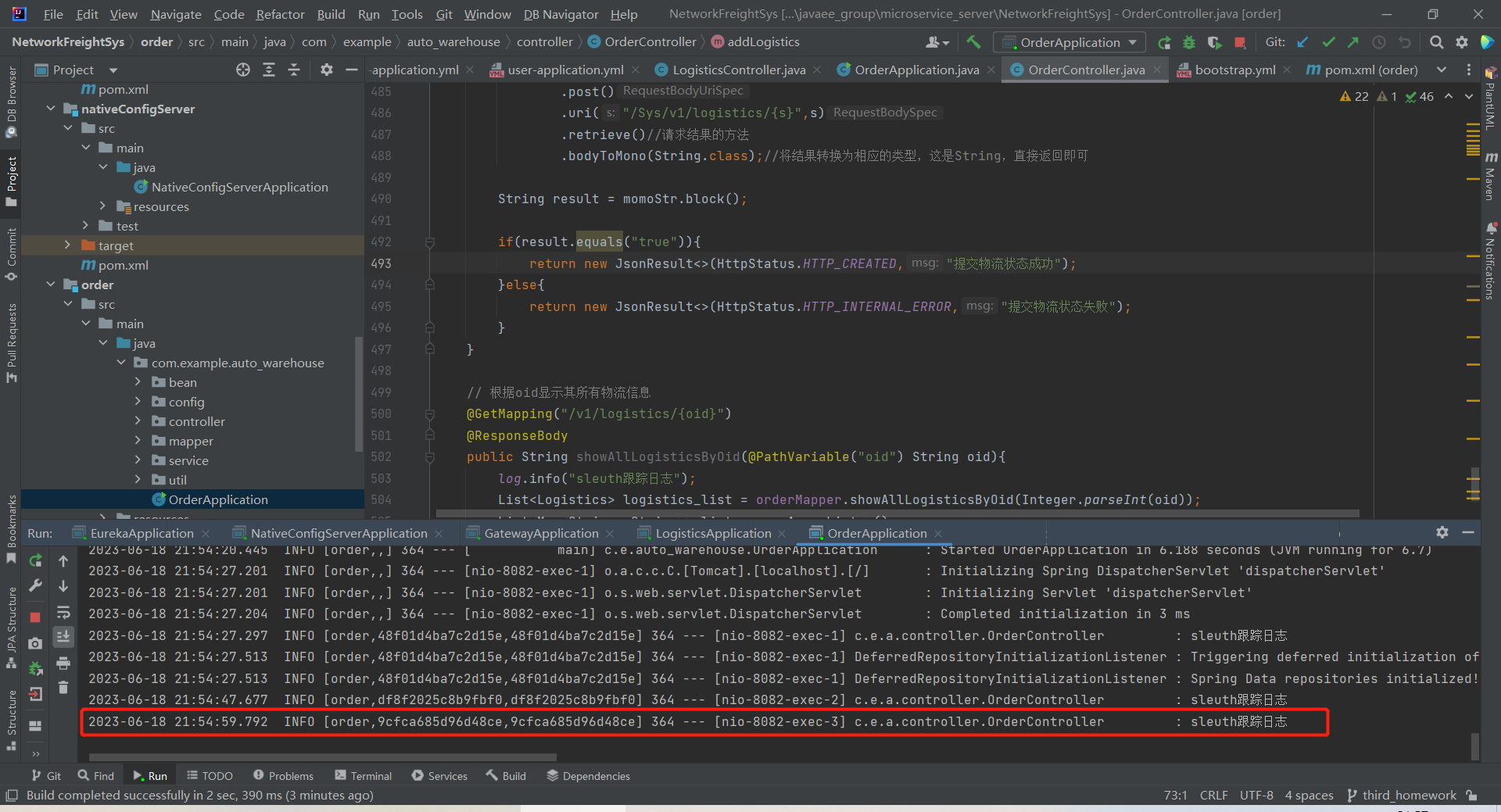


1. 服务调用

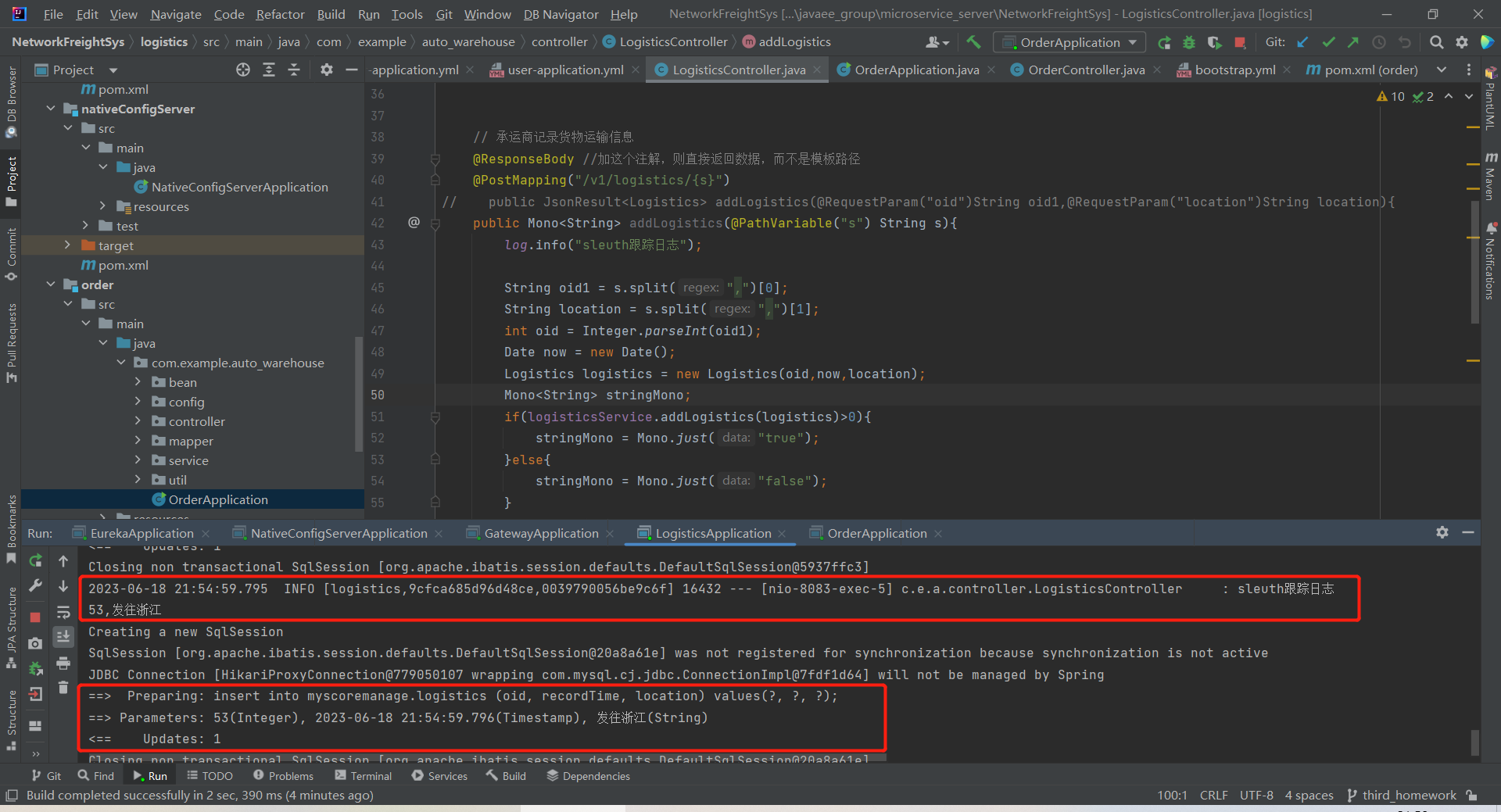
采用WebClient进行服务间调用（javaee课上曾老师推荐）。比如，若承运商要进行物流信息更新或查询，会先访问order服务中的http://localhost:8181/order/Sys/v1/logistics的post接口和http://localhost:8181/order/Sys/v1/logistics/{oid}的get接口。接下来order服务会采用WebClient调用logistics服务，图示如下：

Order服务：

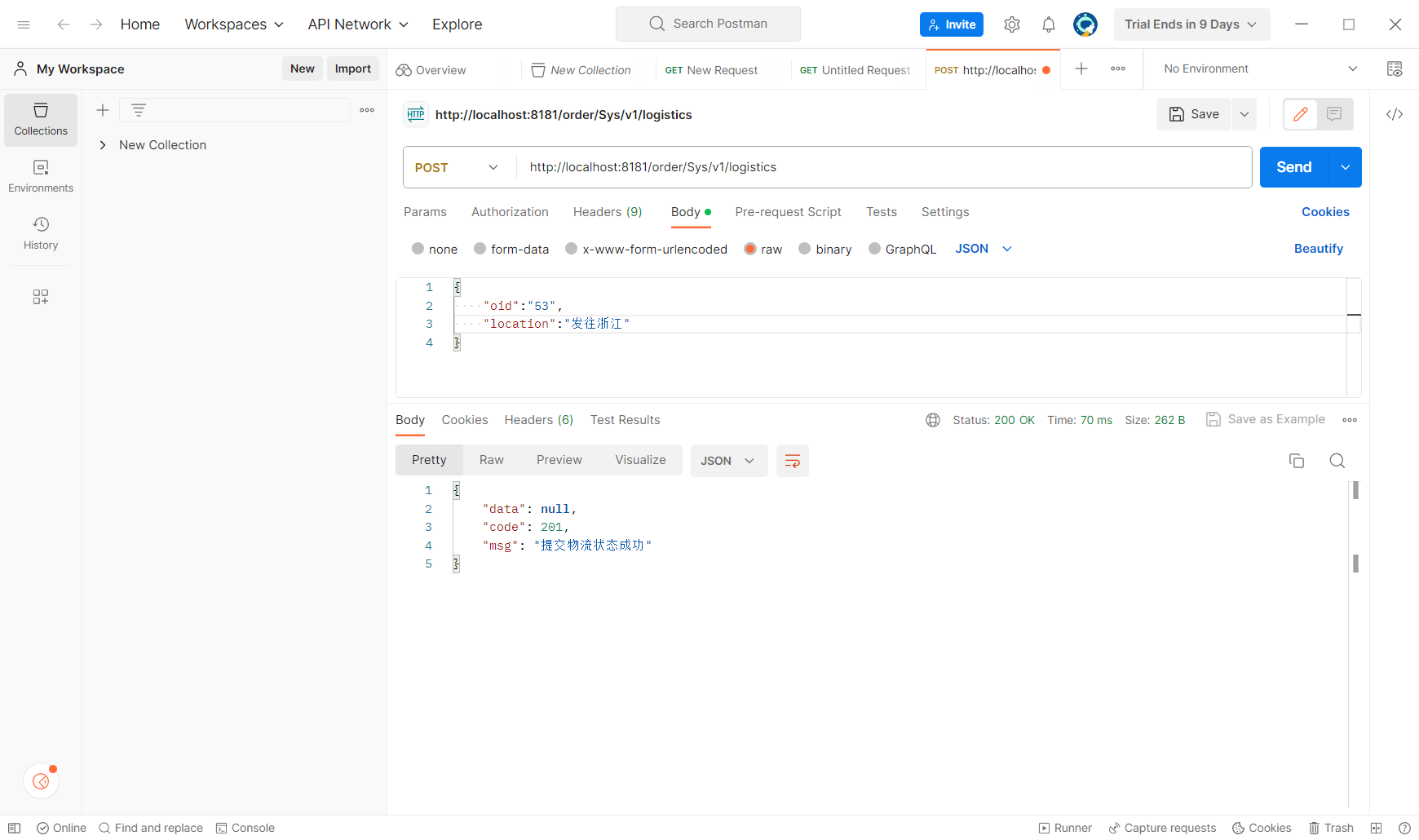




Logistics服务：对logistics表的实际操作是在logistics服务中的，可以看到有数据库的更新信息。Order服务只负责调用传递参数。

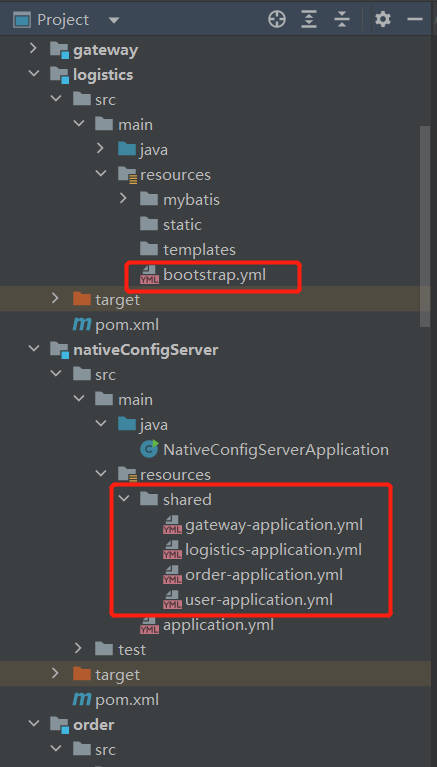


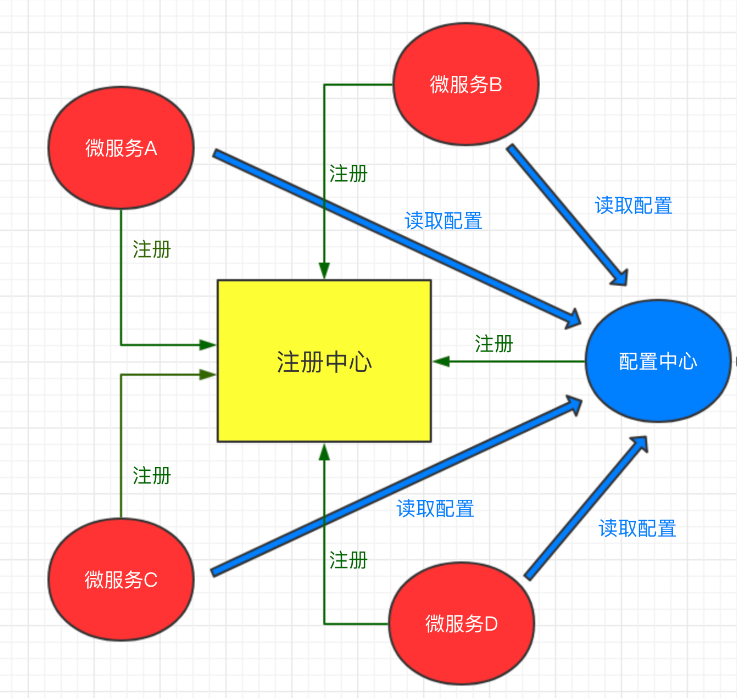
Postman测试结果：



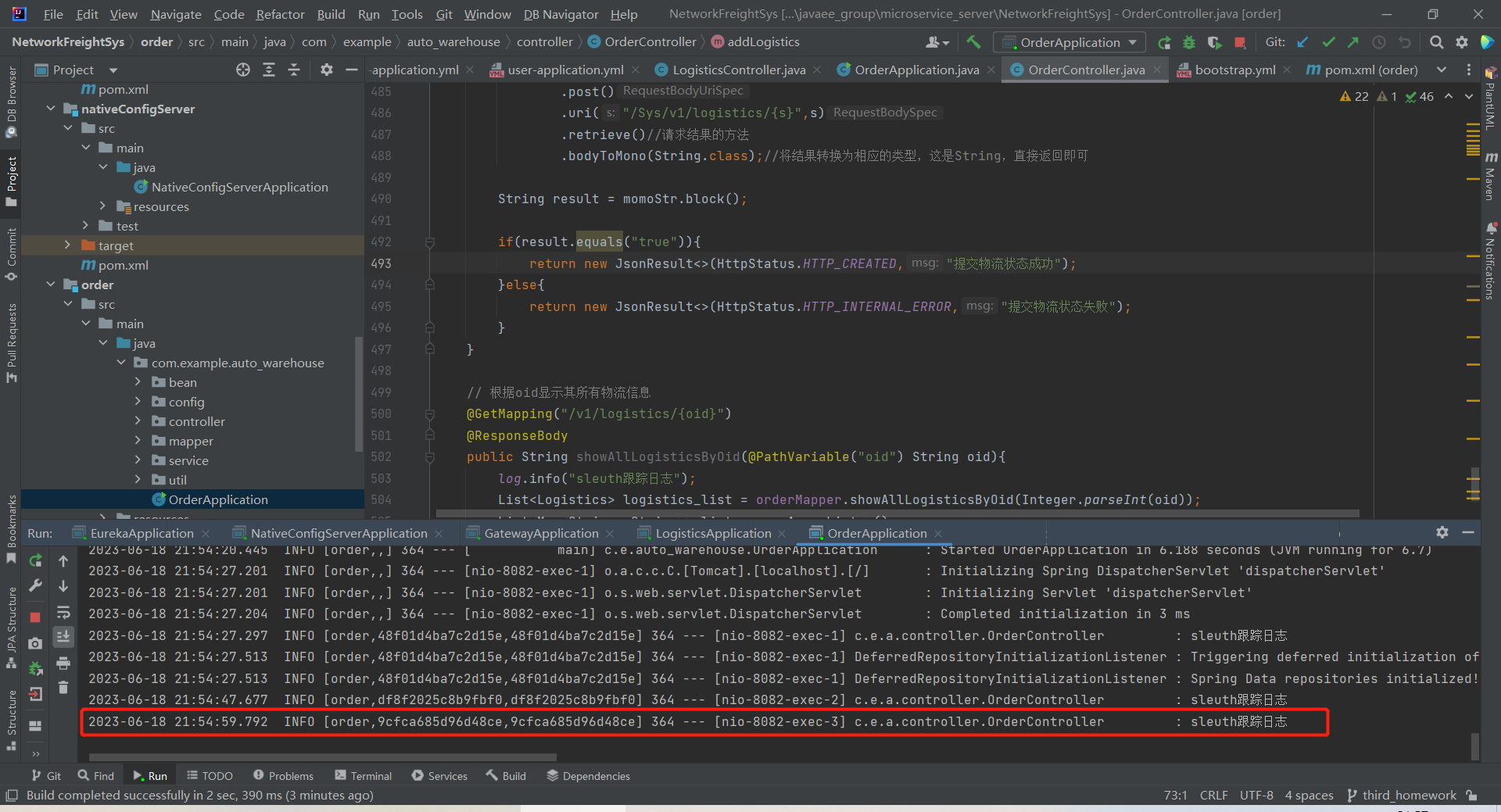
1. 集中配置

采用Spring Cloud Config server进行服务的集中配置，各个服务通过bootstrap.yml获取到自己的application.yml和nativeConfigServer服务的信息，各服务的application.yml统一放置于nativeConfigServer的shared文件夹下：





1. Sleuth跟踪





1. kafka消息队列形成事件驱动

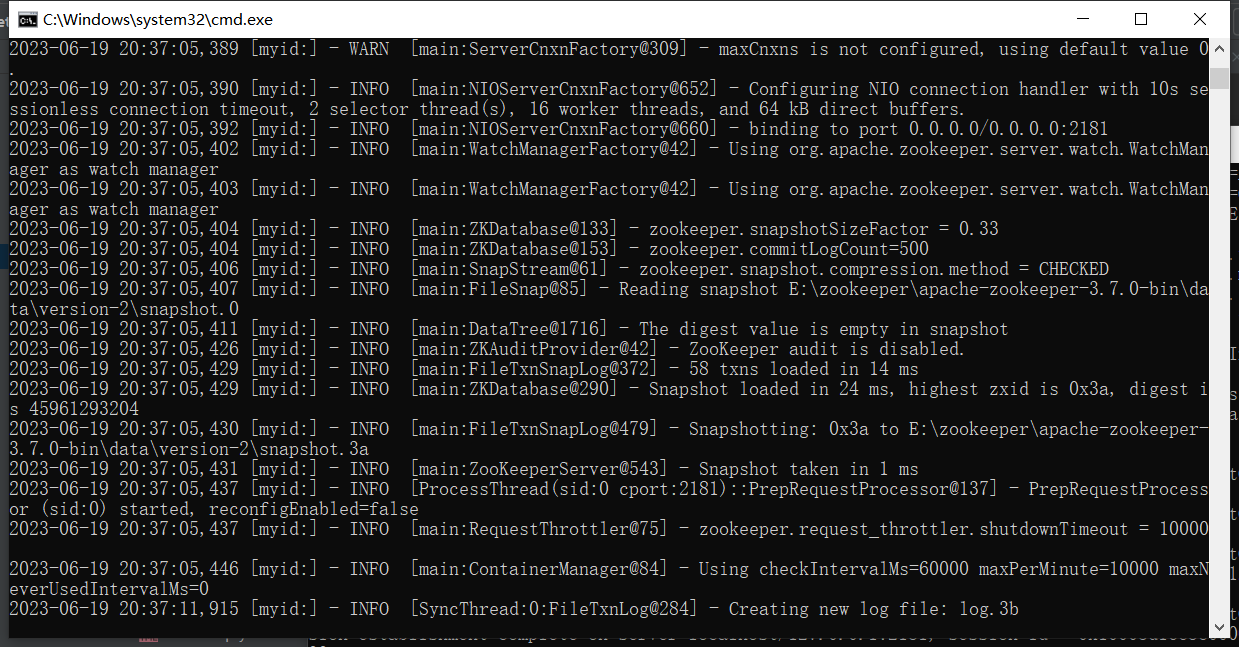
1.运行步骤：

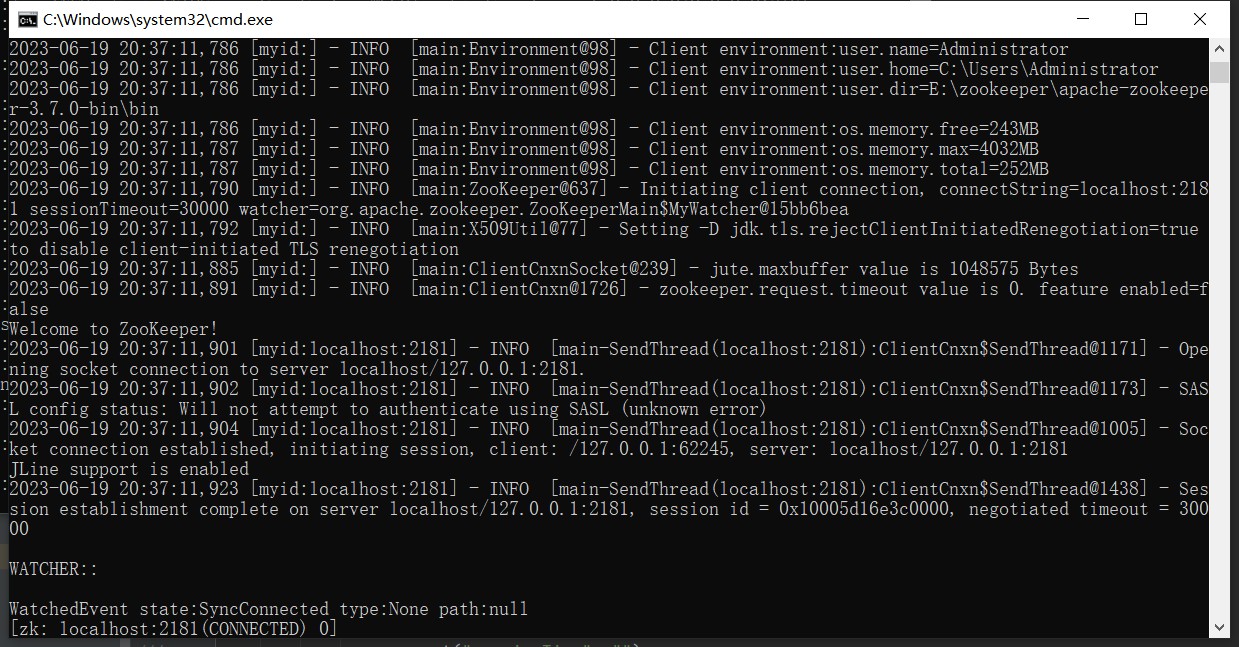
（1）先启动zookeeper和kafka服务器

（2）再依次启动eureka、nativeConfigServer、gateway的application

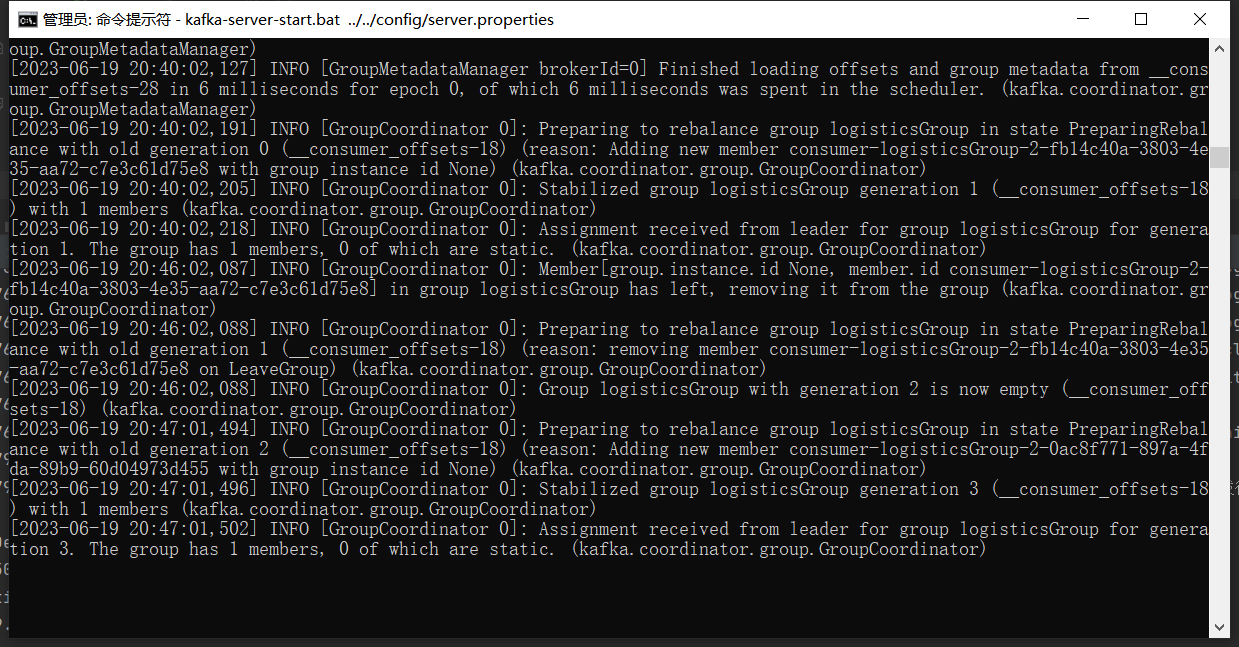
（3）启动order服务和logistics服务（需要保证order在logistics服务之前启动）

下面是zookeeper启动图：



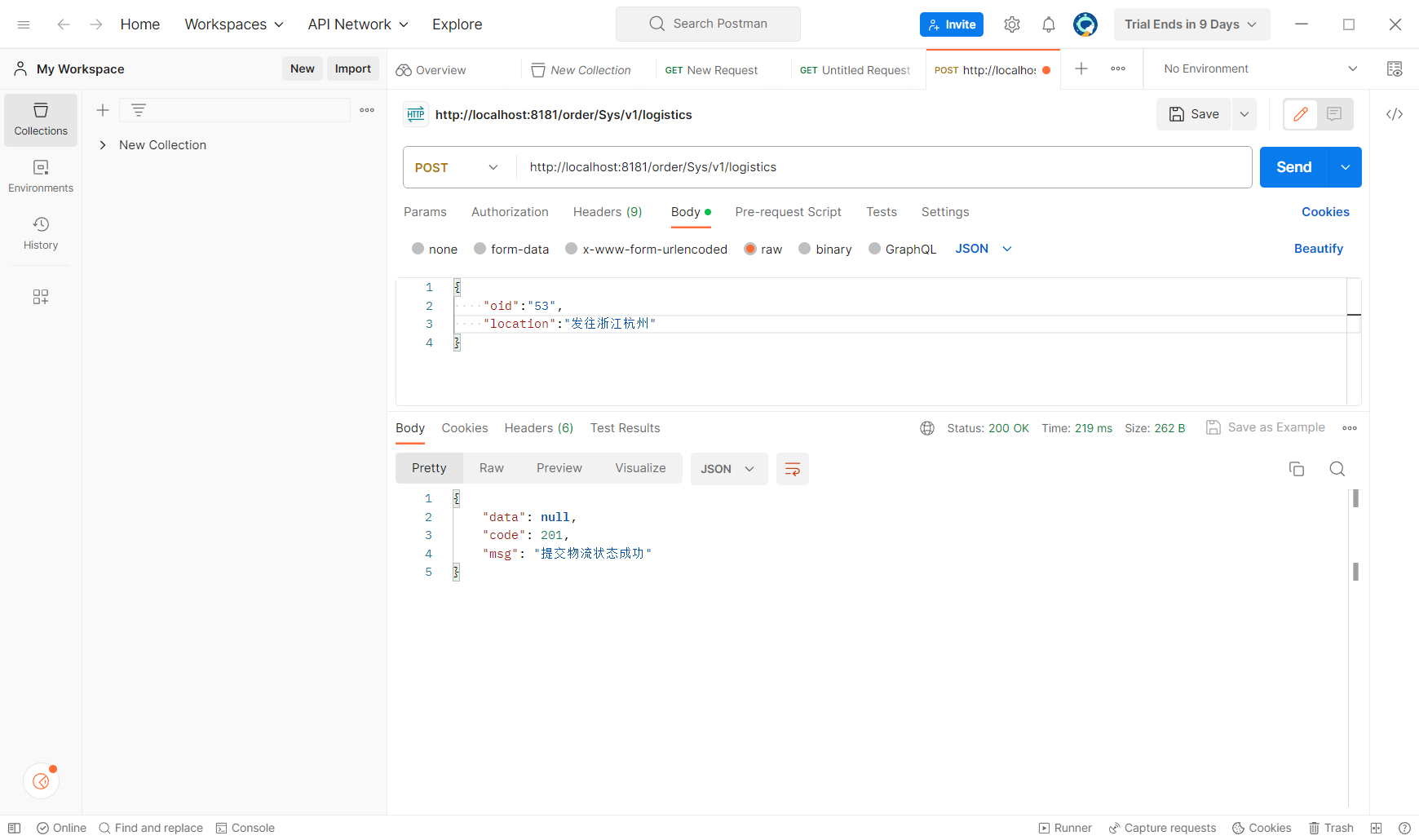


下面是kafka服务器运行图：

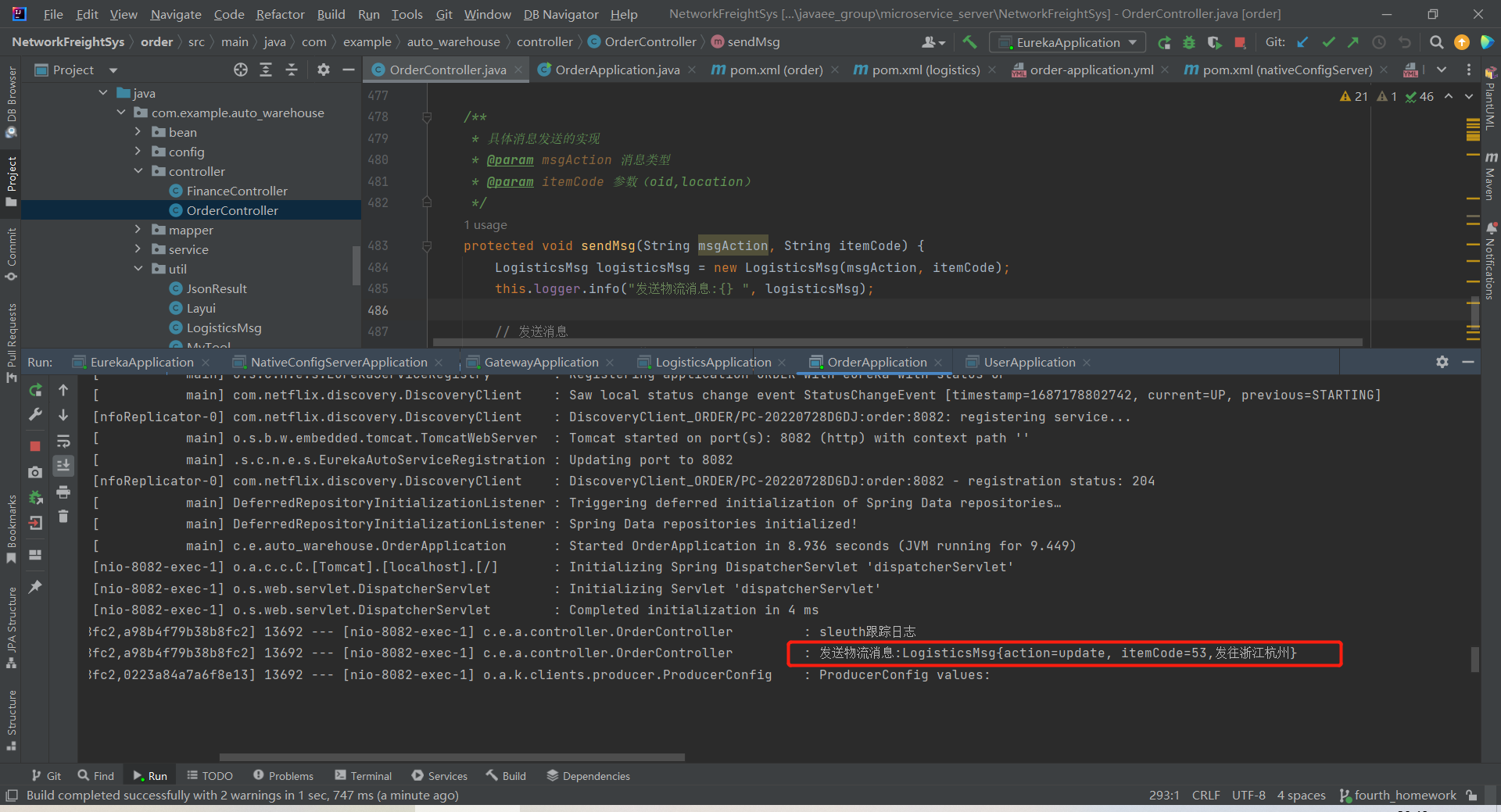


2.说明：

在order和logistics服务中运用kafka消息队列实现事件驱动，比如，调用order服务中的http://localhost:8181/order/Sys/v1/logistics接口，同时传递oid和location参数：



Order服务被调用接口后，向logistics服务发送kafka消息：



Logistics服务的消息监听器LogisticsMsgListener从消息队列中收到消息，进行物流更新信息处理，并打印出更新成功的信息：

