老师好： 以下是我们的小组及仓库信息：

javaee小组成员： 20301155邹佳琪 20301177吴芳玲 20301002党田乐 （每人平分贡献占比，均为33.3%）

github仓库地址： <https://github.com/xuelingzou/javaee_group>

四次作业分别在四个分支中，已用分支名字区分。

这是javaee四次作业完成情况说明对照文档

Assignment 1, A Shipping and Transportation Web application Development with Spring MVC and More

In this assignment, you are asked to develop functions for a shipping and transportation web application.

***Requirements:***

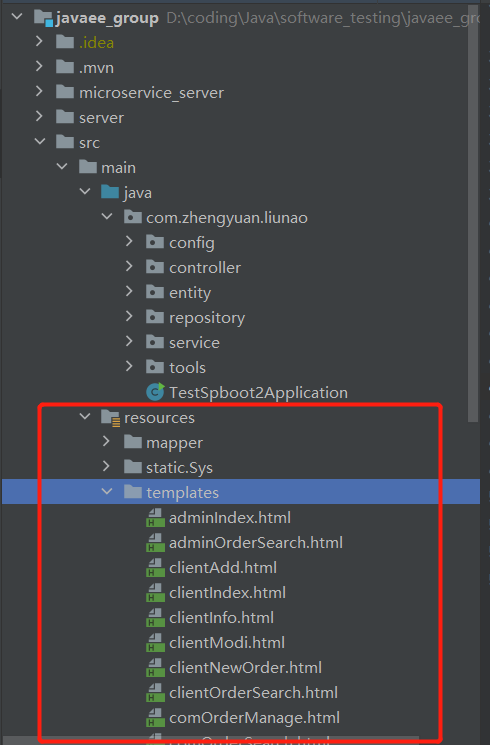
1, Using Spring MVC + Spring Data JPA/Mybatis + Thymeleaf for Web Application Development.

我们使用Spring MVC + Spring Data JPA/Mybatis + Thymeleaf采用前后端不分离形式实现网络货运物流平台系统，代码请见first\_homework分支

后端代码位置：



前端代码位置：

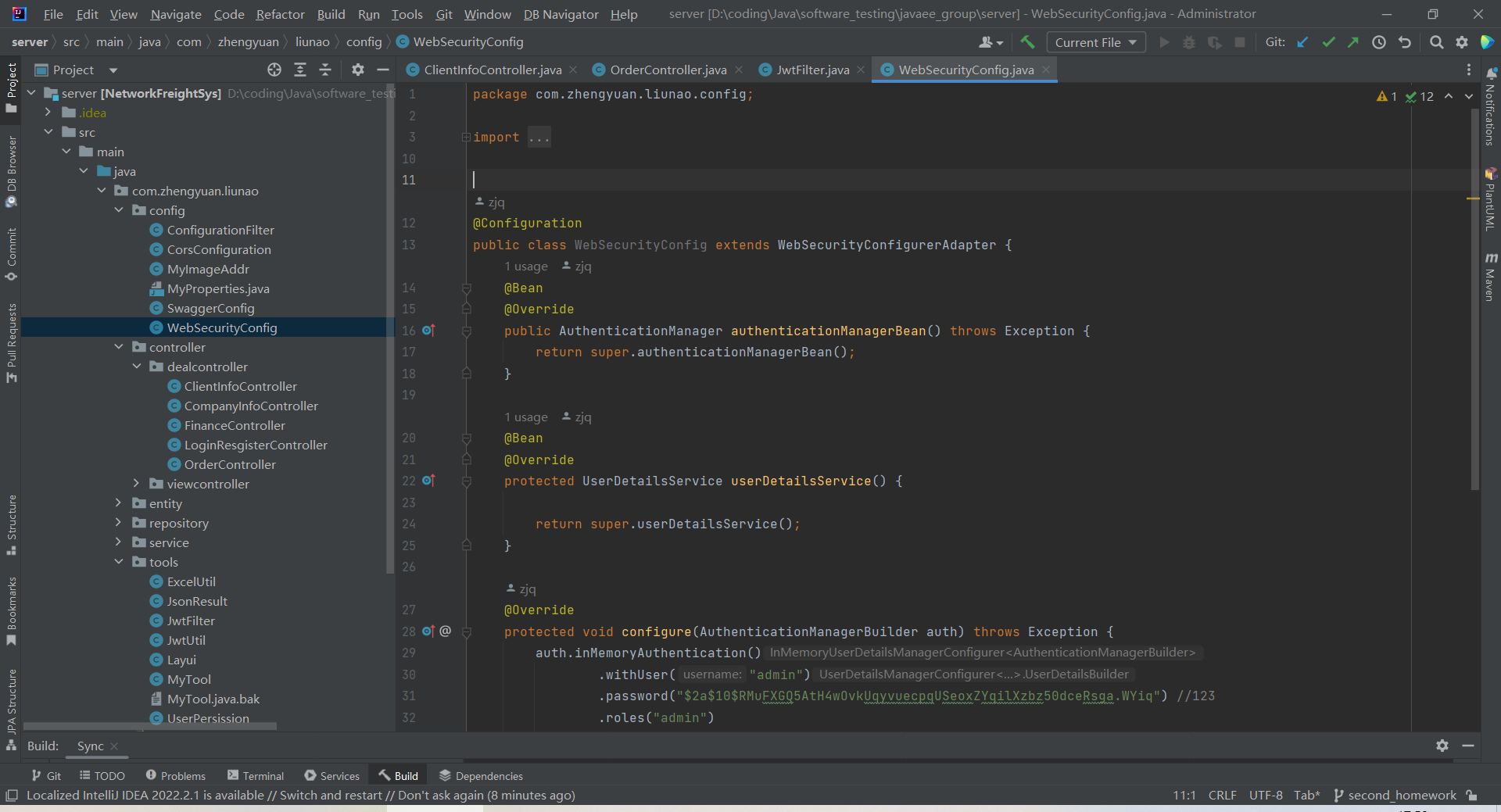


具体功能点请看第一次作业分支中的系统设计文档

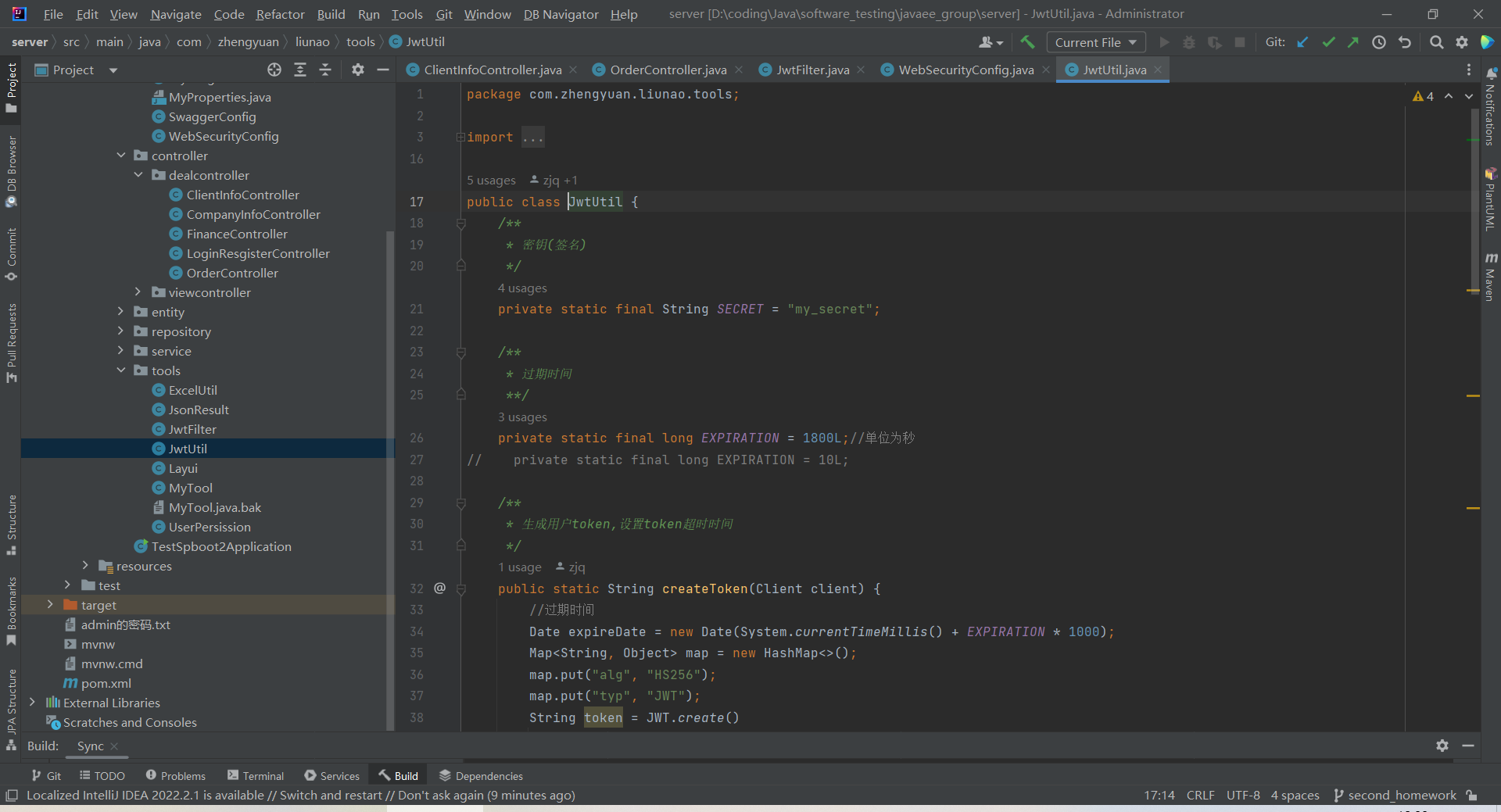
2, Authentication and authorization for web access is necessary.

具体代码位置如下：

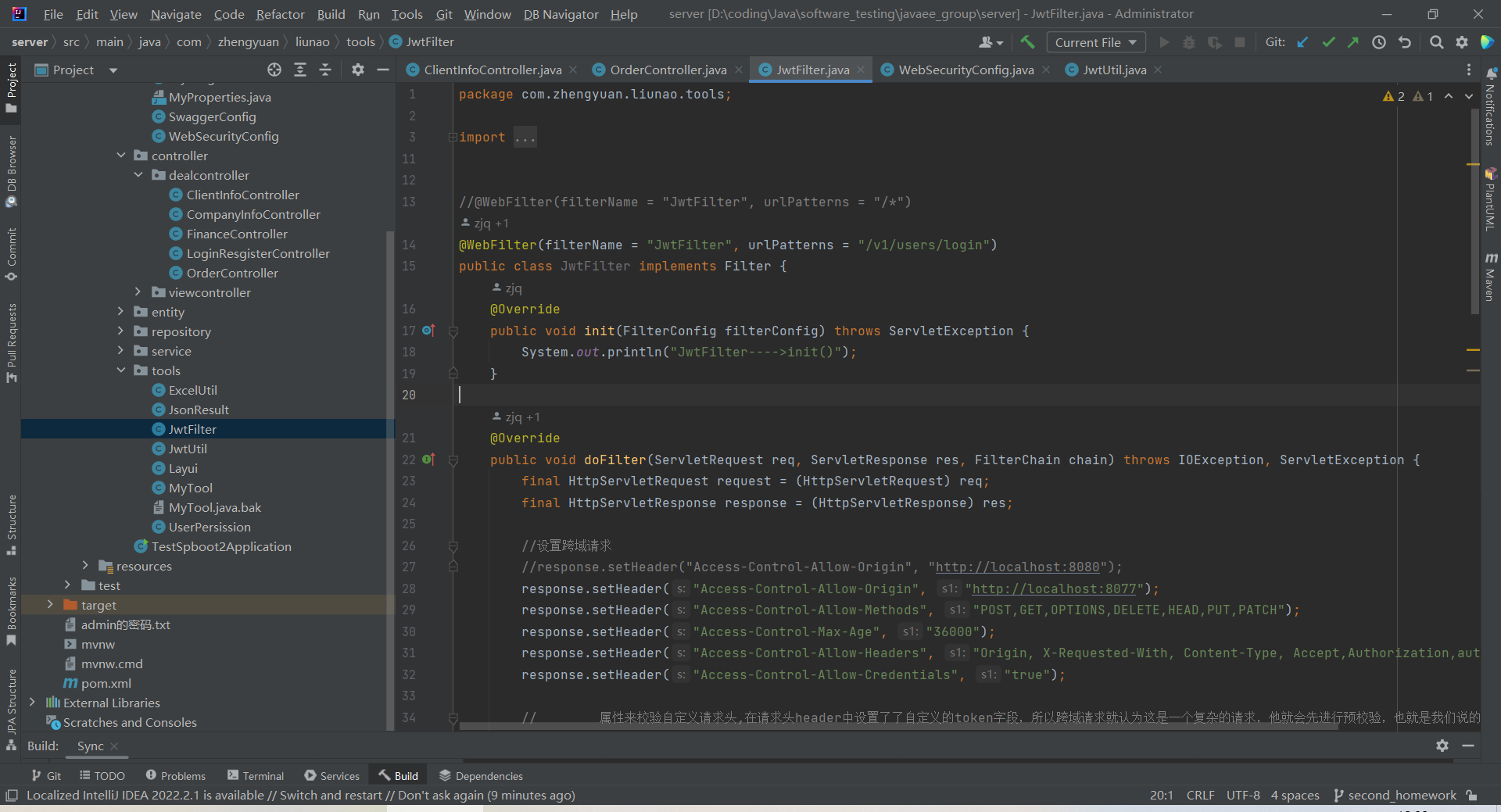
Config包中的WebSecurityConfig.java



tools包中的JwtUtil.java



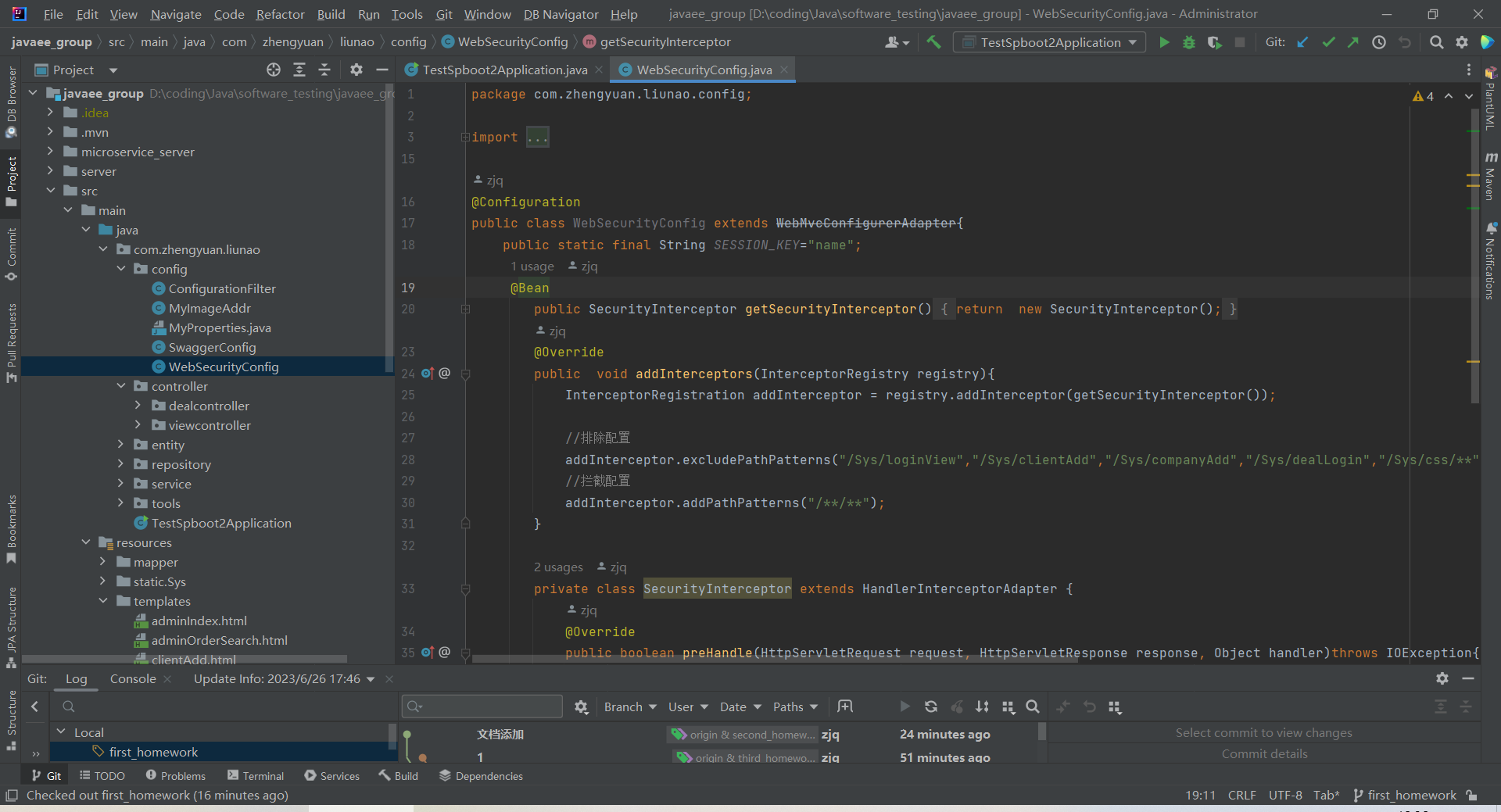
tools包中的JwtFilter.java



3, Unit testing for repository level and integration testing controllers are required.

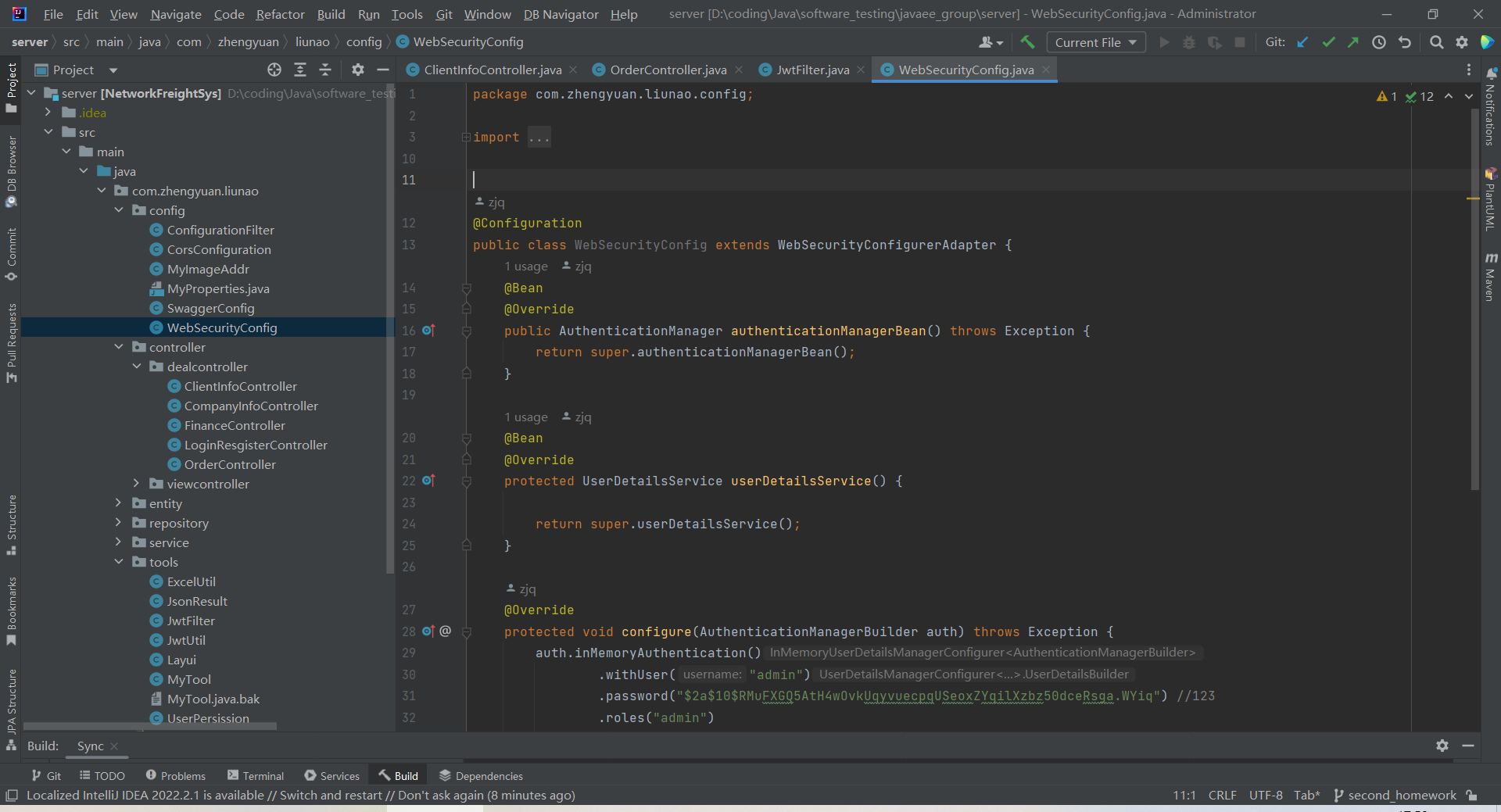
请参照第一次作业分支下的用户手册文档，其中包含所有单元测试和集成测试截图。

4, You are encouraged to apply spring security, cookies, session management, interceptors /filters to improve the system functions.

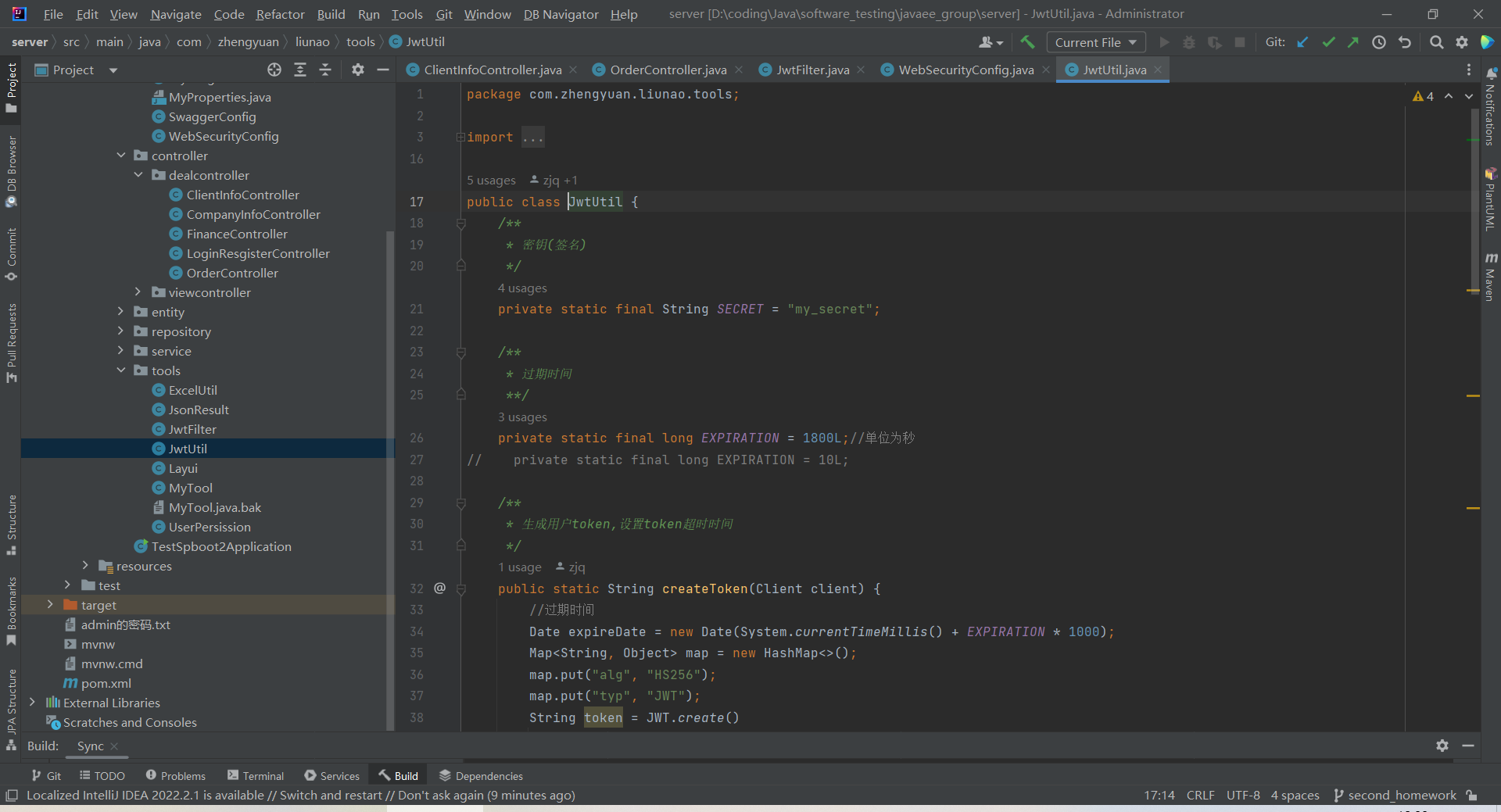


具体代码位置如下：

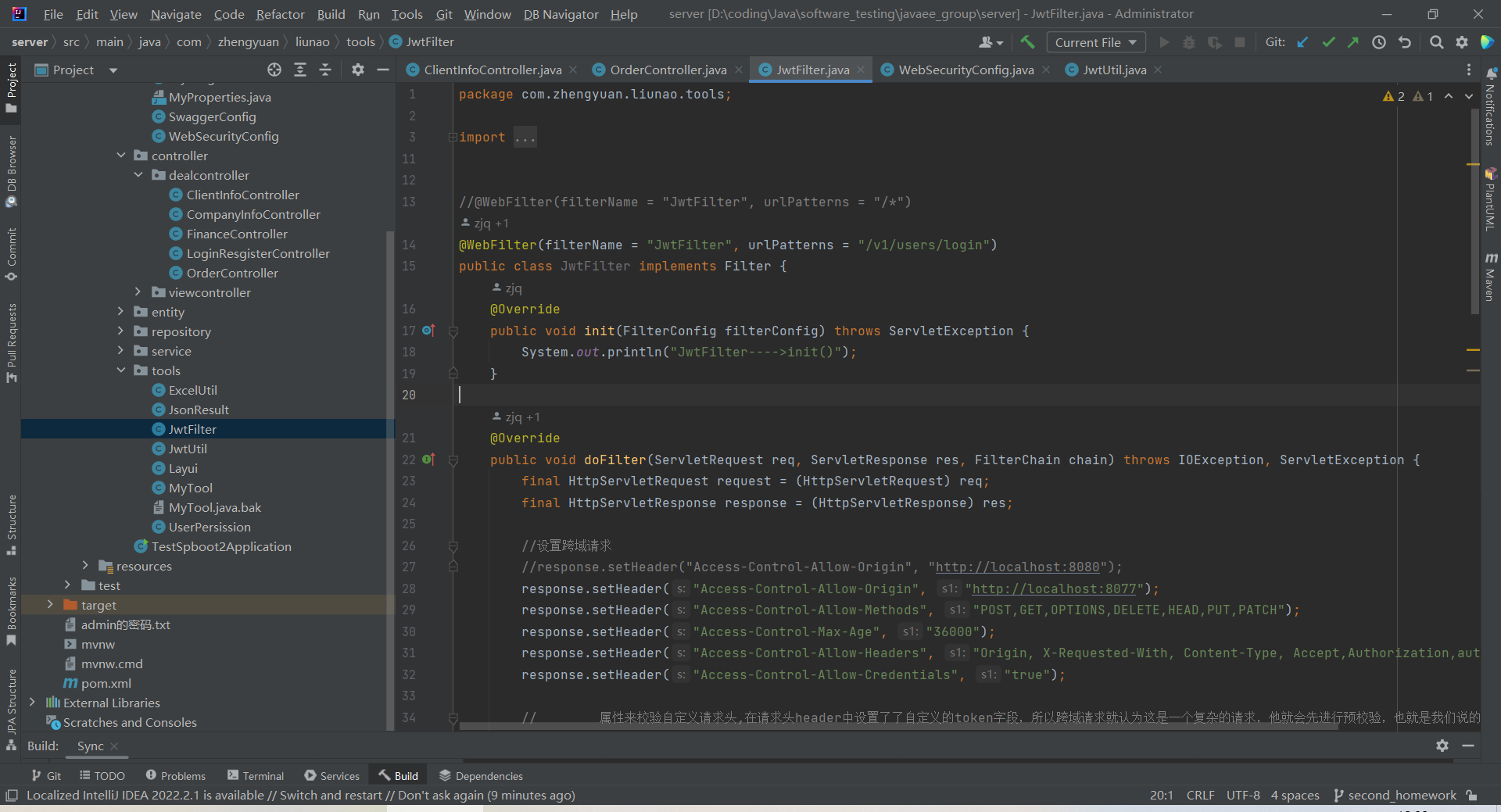
Config包中的WebSecurityConfig.java



tools包中的JwtUtil.java



tools包中的JwtFilter.java



***Delivery:***

1, You need to archive the project source code and running and testing screen capture to a github repository.

2, Providing a system design report is necessary.

3, Hand in due to 2023-4-30

Reference:

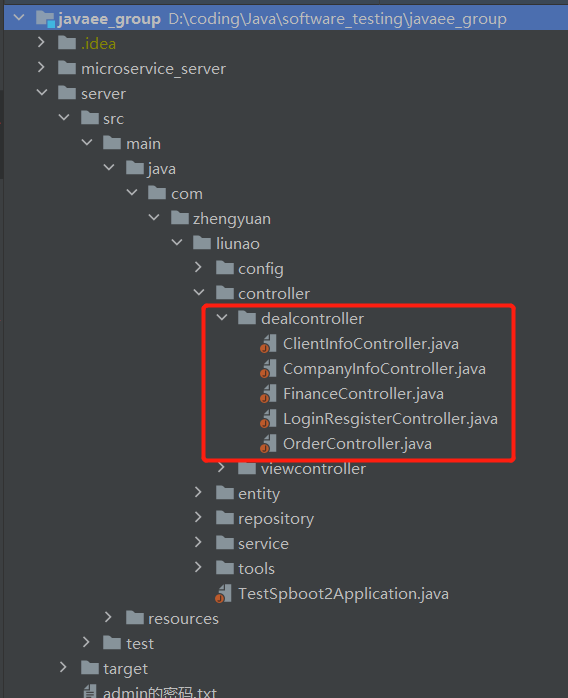
Assignment 2, Re-design of assignment1 with REST API and More

In this assignment, you are asked to re-develop the shipping and transportation management web application using REST api and Ajax.

***Requirements:***

1, Design and implement the shipping and transportation services with Restful API.

我们对后端controller中的所有API接口都进行了重构，修改为Restful API设计规范形式。



我们严格遵守规范，以下是 RESTful API 的设计规范：

* 每个资源使用唯一的 URL (URI) 作为其标识符。
* 将 HTTP 方法（GET、POST、PUT、DELETE 和 PATCH 等）用于标识要对资源执行的操作。
* URL 组成：/resource/collection/id。
* /resource：该 URL 指定了资源名称。
* /collection：该 URL 表示该资源所属的集合名称。
* /id：该 URL 表示某个特定资源的唯一标识符。
* 只使用名词来描述资源。
* 不使用动词来描述资源。HTTP 方法描述了要执行的操作类型。
* 对资源进行分类/过滤可以在查询参数中使用。
* 在请求正文中，使用 MIME 类型来描述数据格式。
* 禁止使用常量 URL；
* 最好不要暴露数据库或任何底层实现细节，所有资源都应该对外部客户端隐藏它们的实现。
* 保持接口清晰。可以设置版本号来维护API的兼容性。

代码中restful API接口示例：（其余接口请老师直接看第二次作业分支中server后端代码）

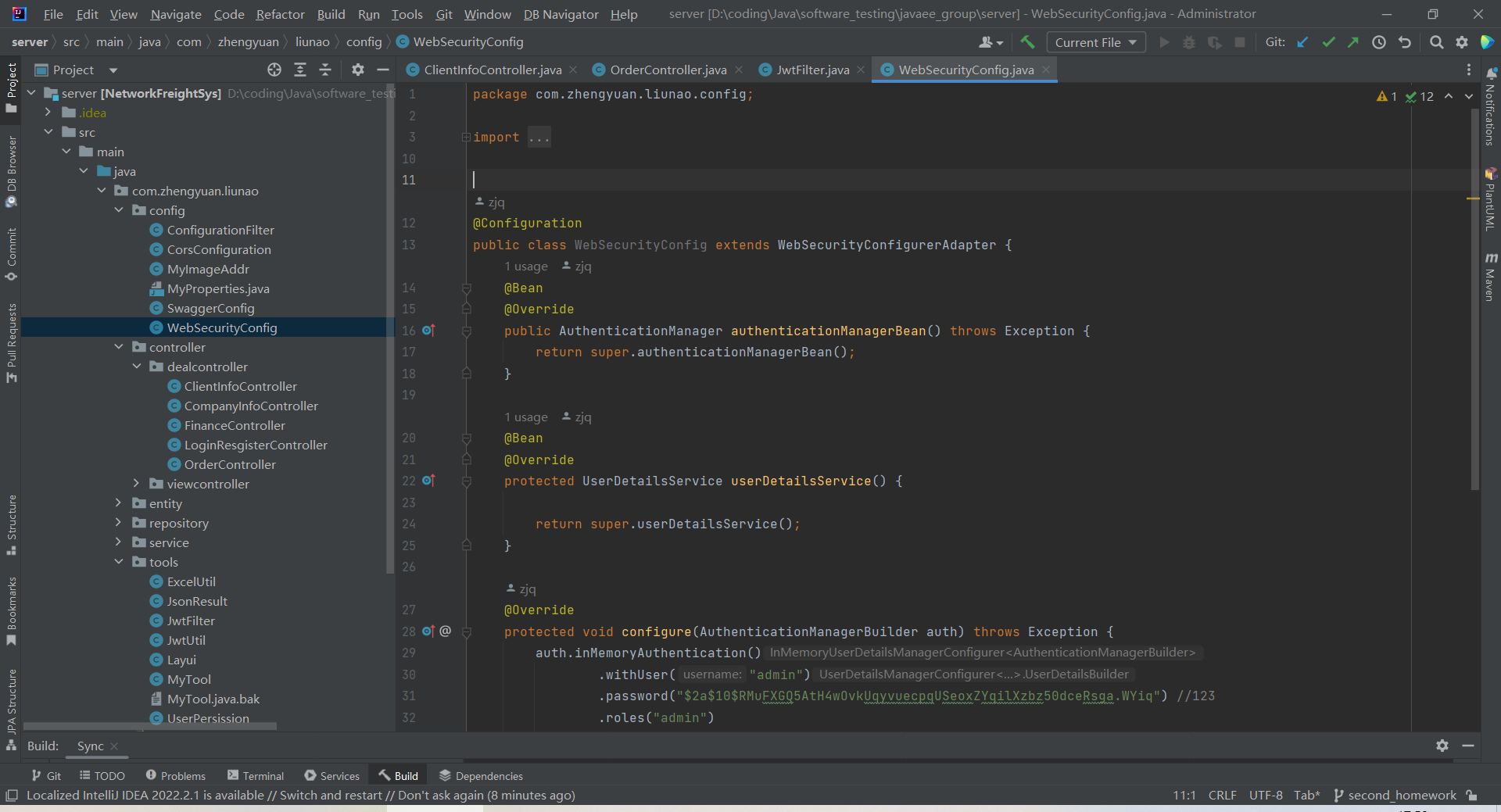




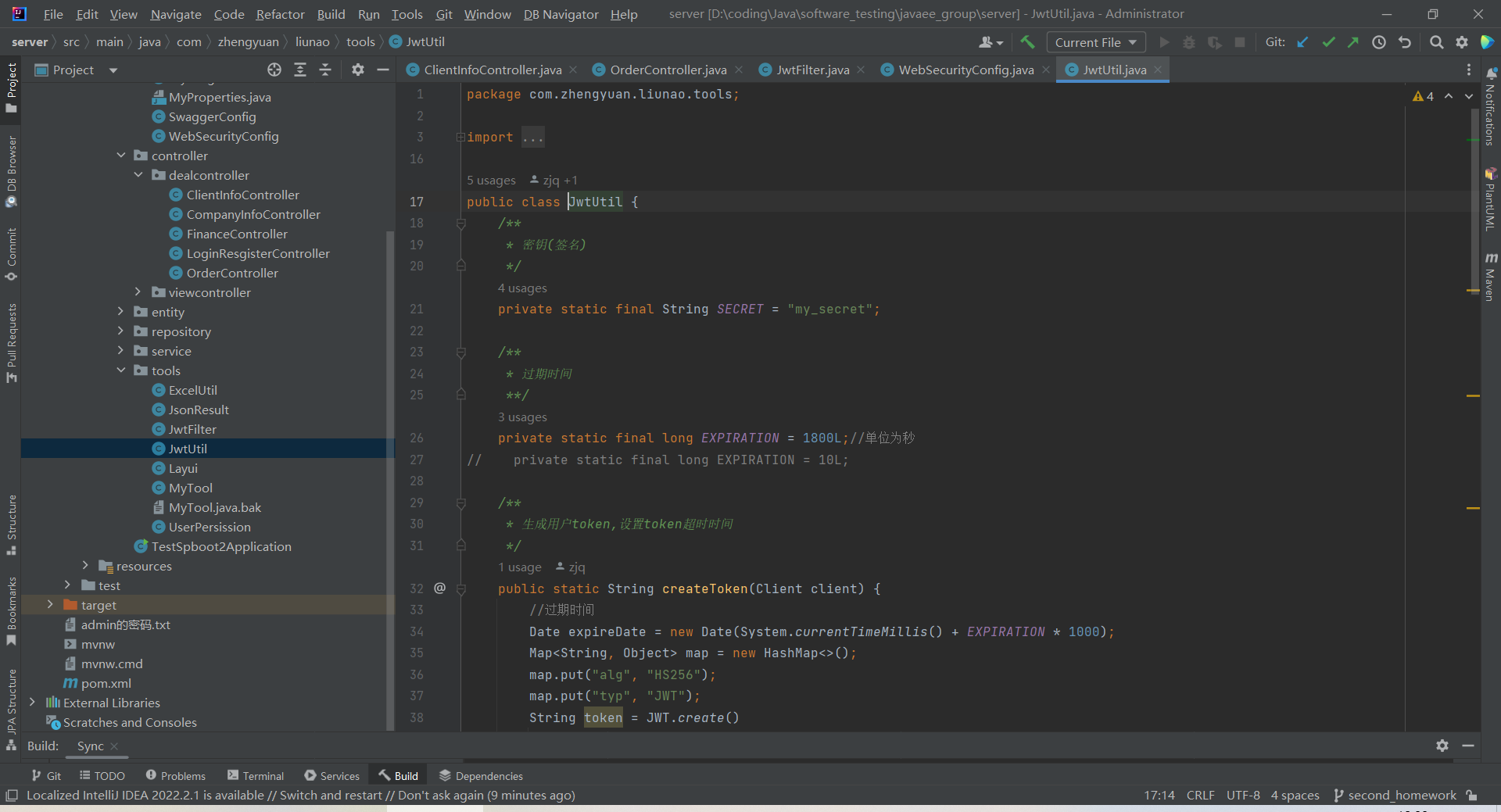
2, API Authentication and authorization using spring security and JWT is necessary.

具体代码位置如下：

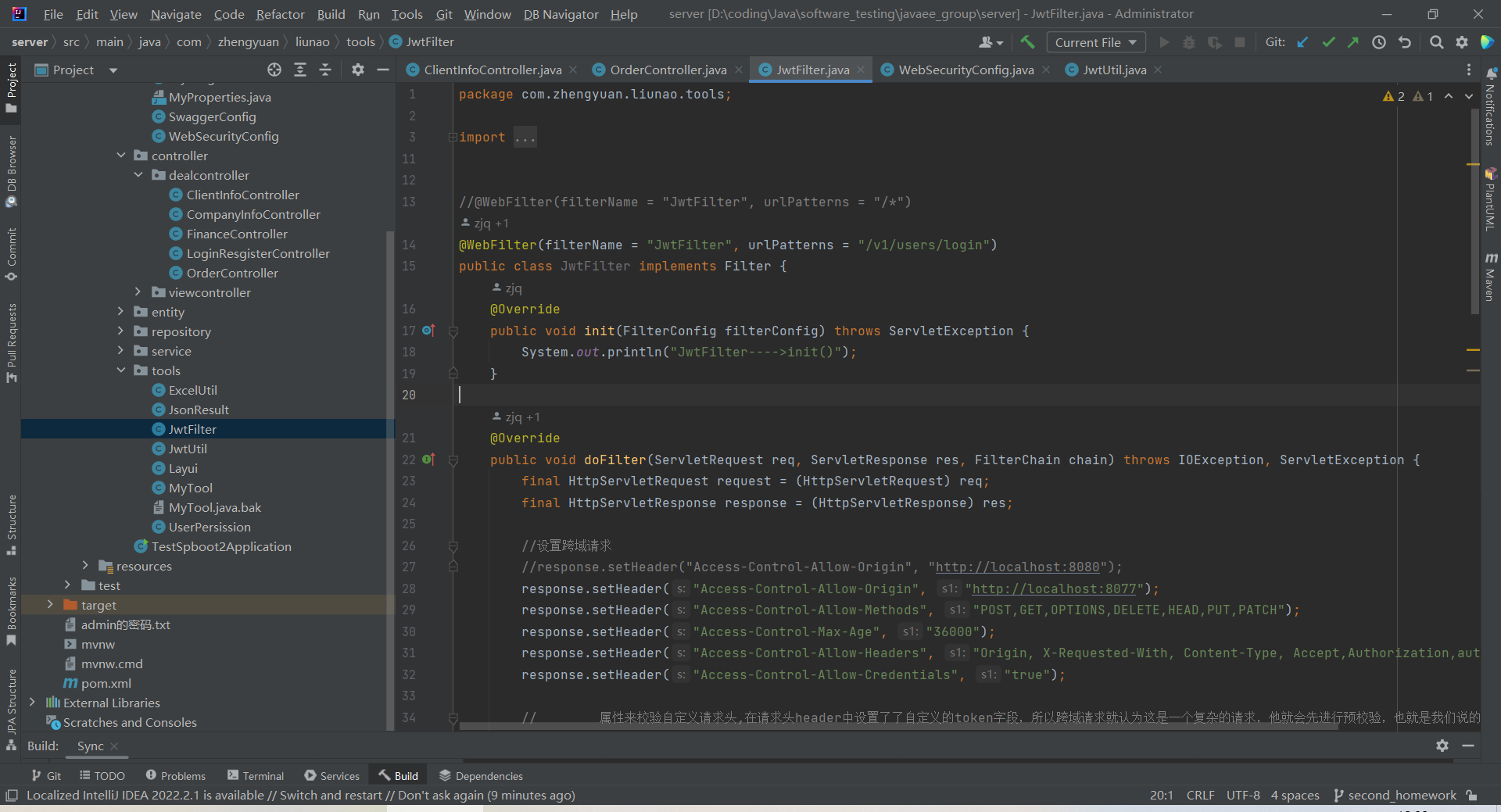
Config包中的WebSecurityConfig.java



tools包中的JwtUtil.java



tools包中的JwtFilter.java



3, Continuous Unit testing for new added functions and components are required.

请参照第二次作业分支下的用户手册文档，其中包含所有单元测试和集成测试截图。

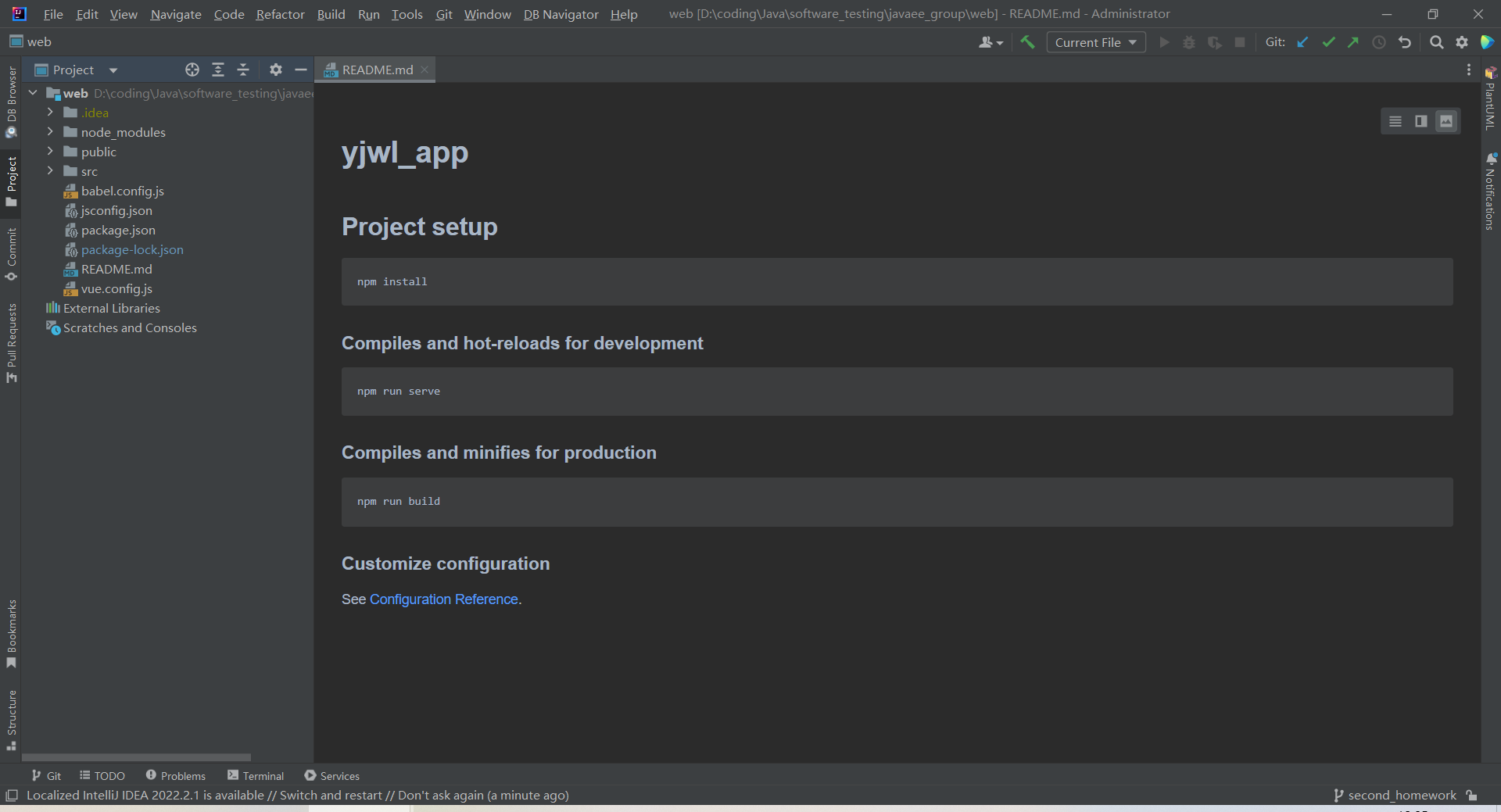
4, You are encouraged to apply openapi document, rate limiting and etc to improve your rest api.

我们使用swagger自动生成API文档，openapi文档。



5, Web-End may need redesign with ajax and vue.js replacing thymeleaf.

详见第二次作业分支中的web文件夹中代码，前端已重新设计，用ajax和vue.js取代thymelaf



***Delivery:***

1, You need to archive the project source code and running and testing screen capture to a github repository.

2, Provide a system design report.

3, Hand in due to 2023-5-22

Reference:

Assignment 3, A shipping and transportation services development with Micro-services Architecture and Spring-Cloud

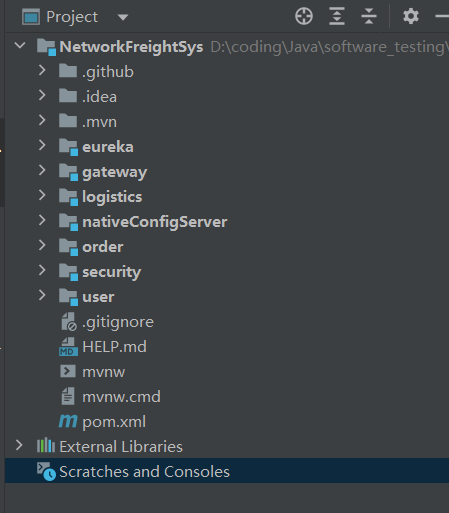
In this assignment, you are asked to refactor the shipping and transportation management server-end using spring cloud.

***Requirements:***

1, Re-structuring your shipping and transportation services as micro-services.

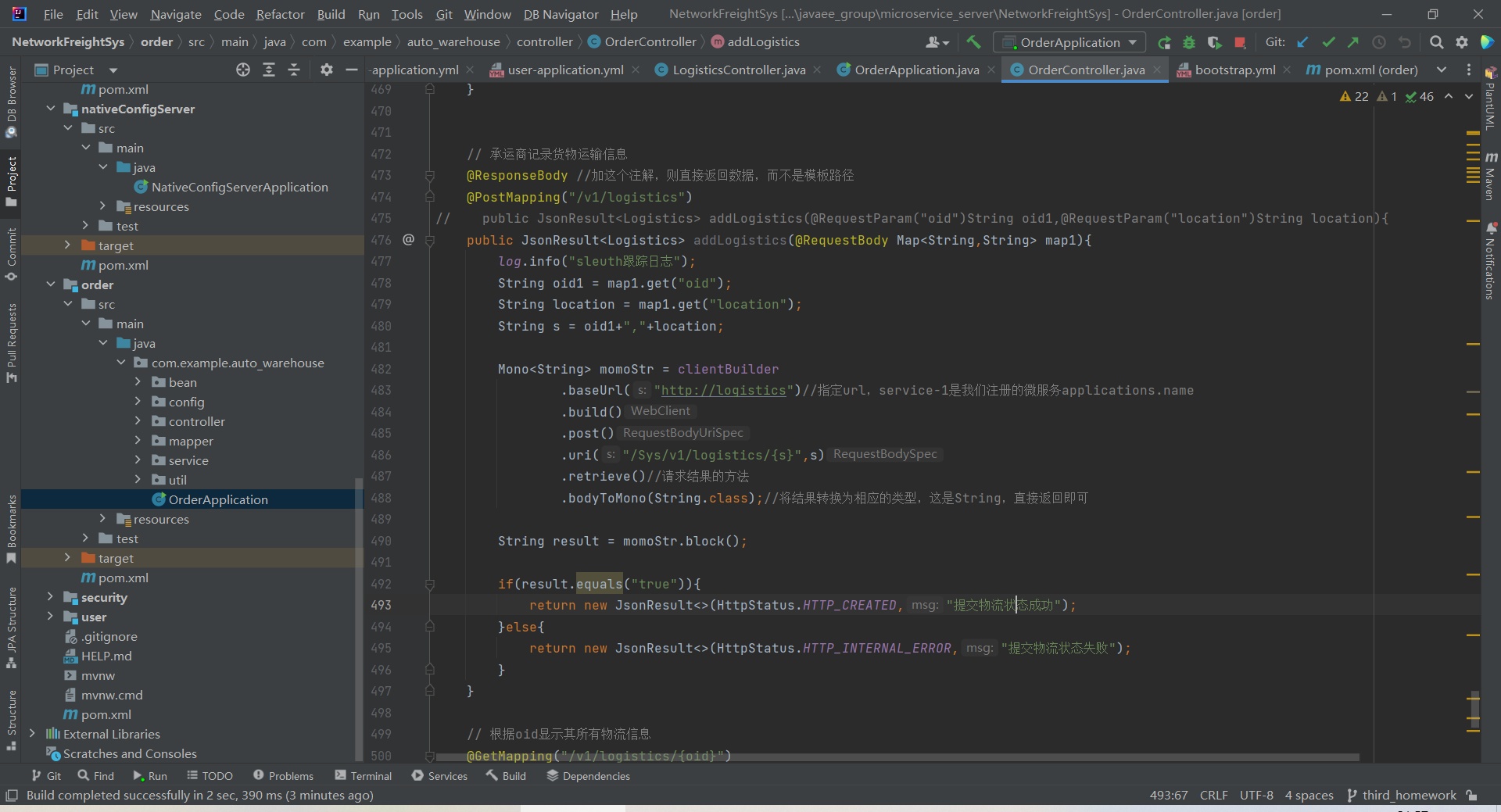
后端微服务共分为七大部分，如下图所示，现做如下阐述：

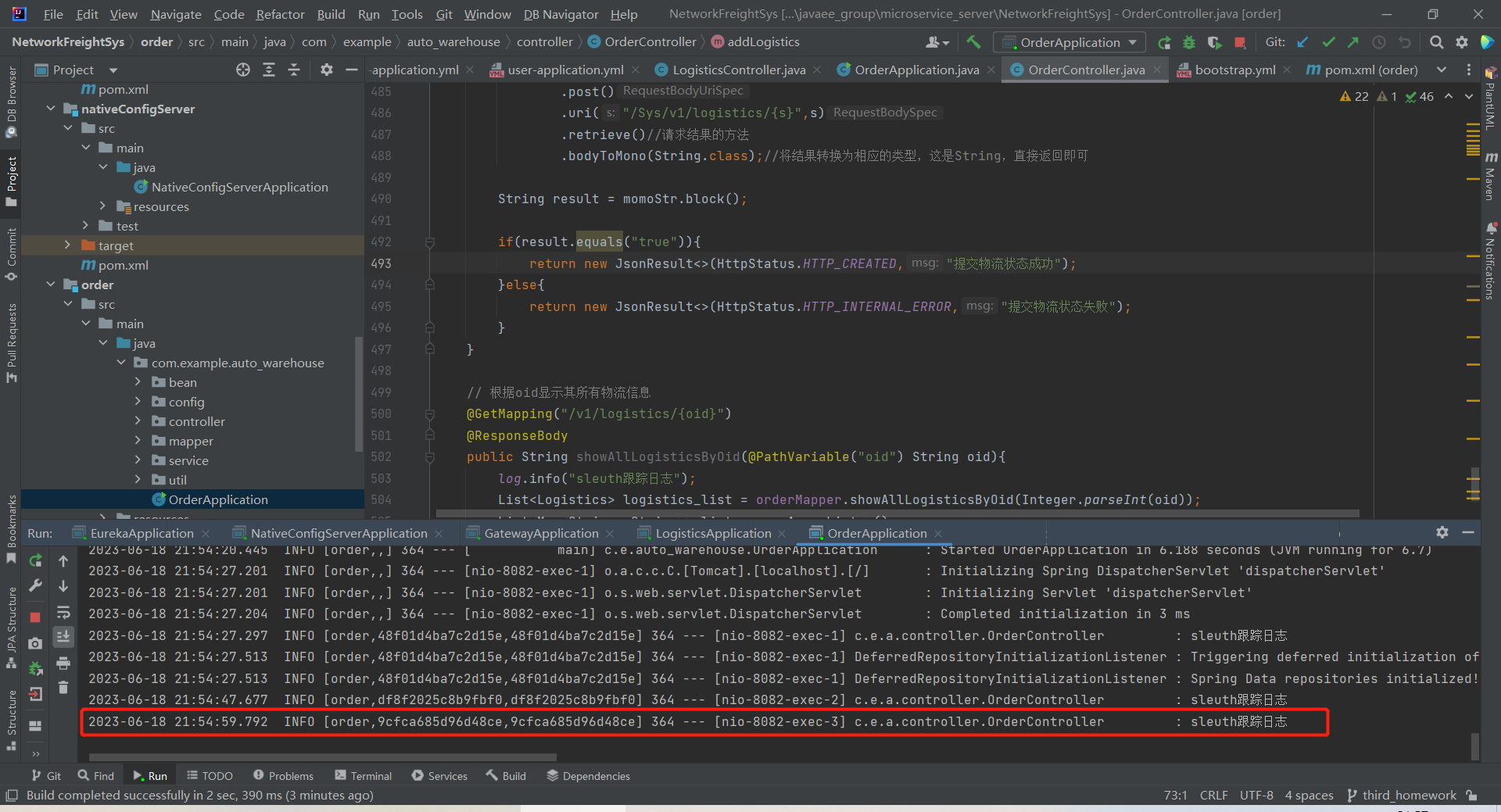
1. 用户user服务：包含登录注册、客户信息管理、承运商信息管理功能；
2. 订单order服务：包含对订单的修改、查询、订单财务管理等功能；
3. 物流logistics服务：包含物流录入、查询等功能；
4. eureka服务：实现服务的注册与发现；
5. gateway服务：实现路由映射，外部用户可通过gateway通过服务调用到内部接口；
6. oauth2Server服务：oauth2授权与认证等安全功能；
7. nativeConfigServer配置服务：实现集中配置。



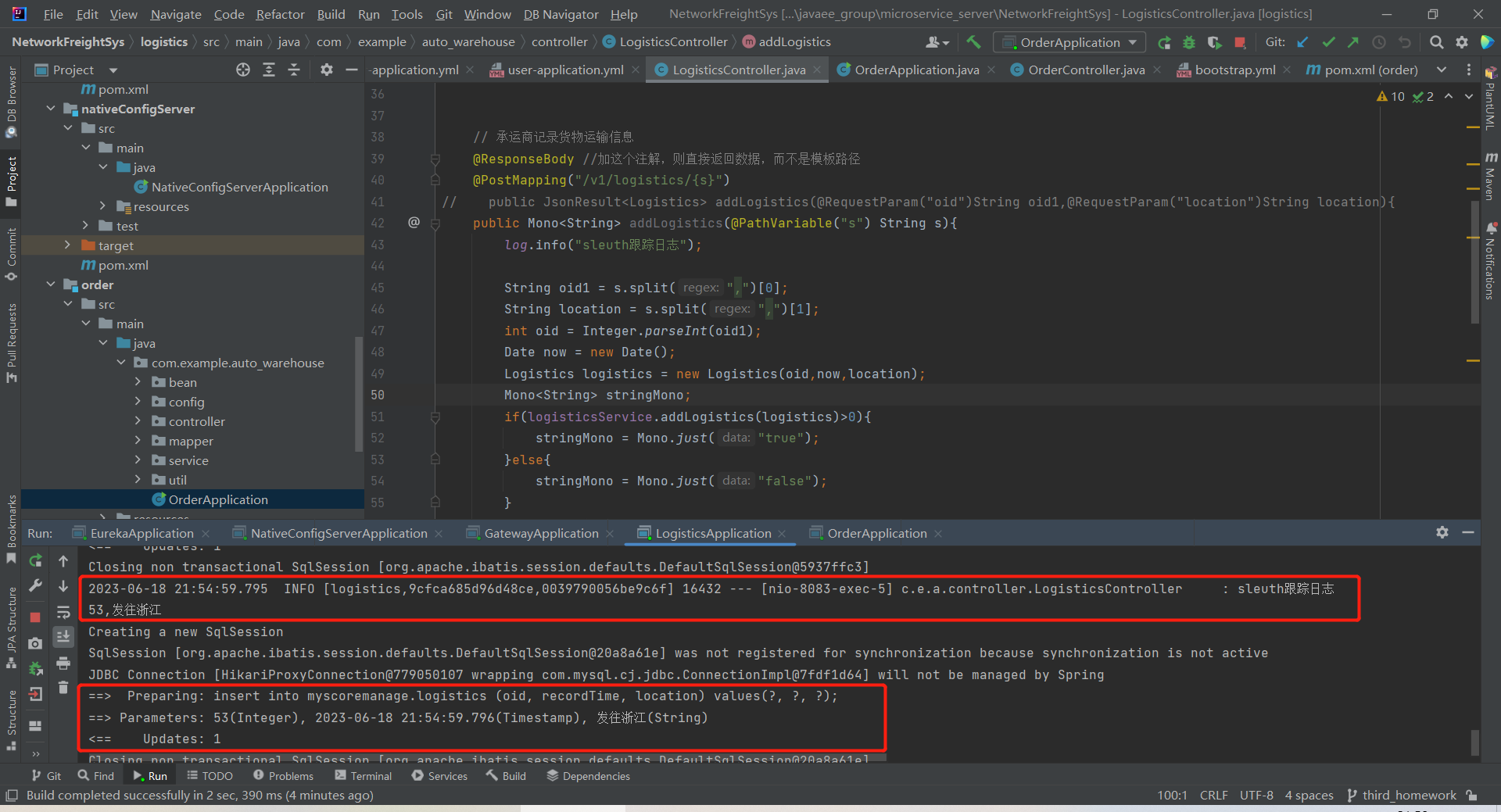
采用WebClient进行服务间调用（javaee课上曾老师推荐）。比如，若承运商要进行物流信息更新或查询，会先访问order服务中的http://localhost:8181/order/Sys/v1/logistics的post接口和http://localhost:8181/order/Sys/v1/logistics/{oid}的get接口。接下来order服务会采用WebClient调用logistics服务，图示如下：

Order服务：

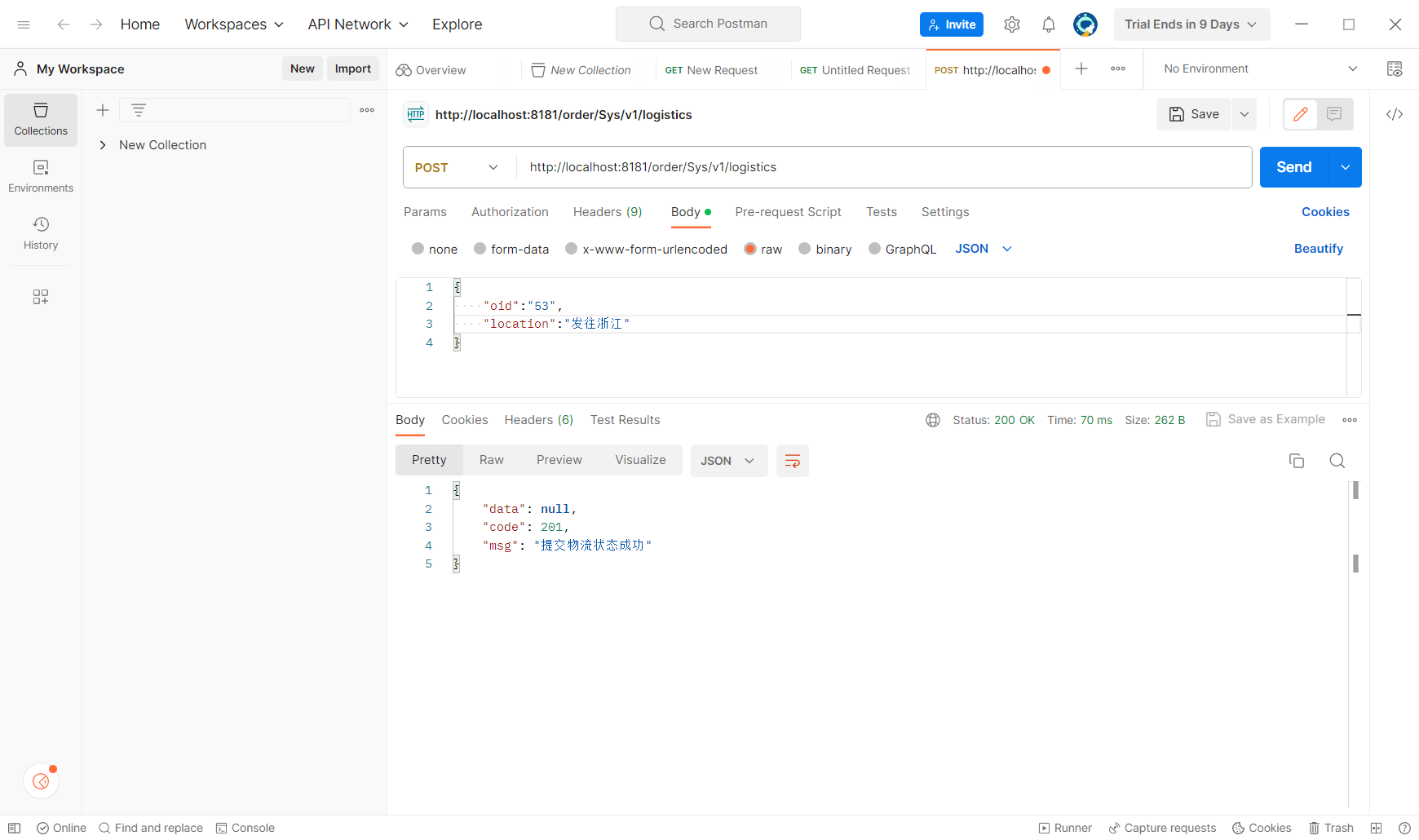




Logistics服务：对logistics表的实际操作是在logistics服务中的，可以看到有数据库的更新信息。Order服务只负责调用传递参数。



Postman测试结果：



2, Service discovery with Eureka is necessary.

1.项目运行环境为java jdk11

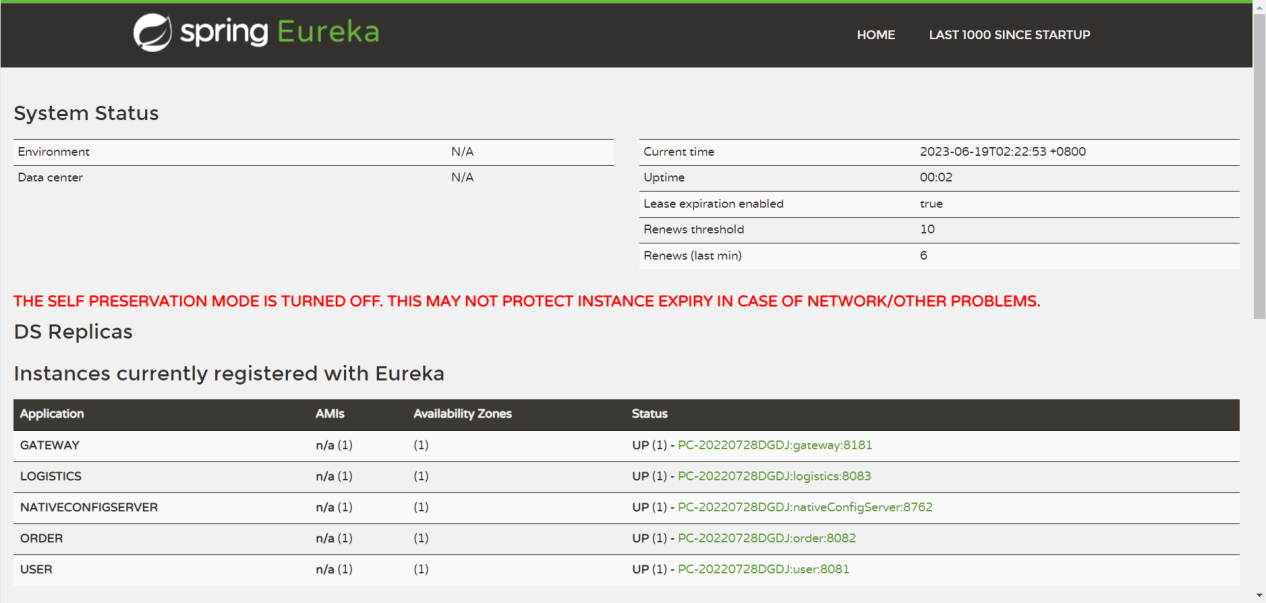
2.后端微服务项目的运行步骤为：

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

（2）再运行user、logistics、order

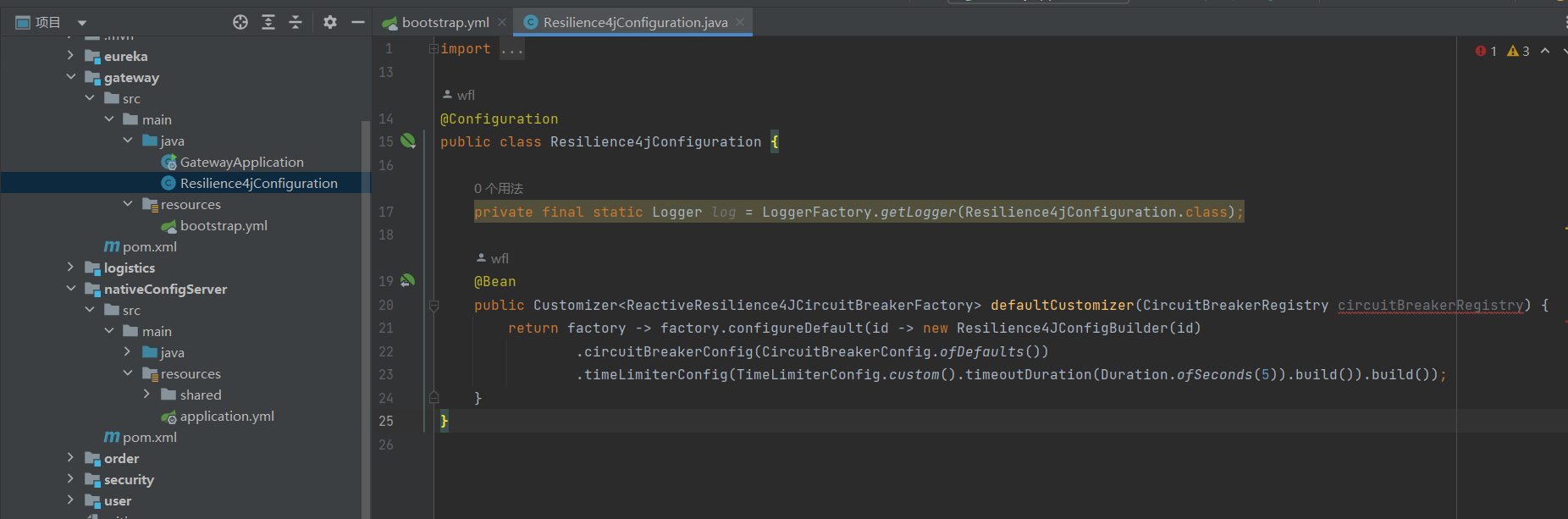
3.登录网址：<http://localhost:8761/>

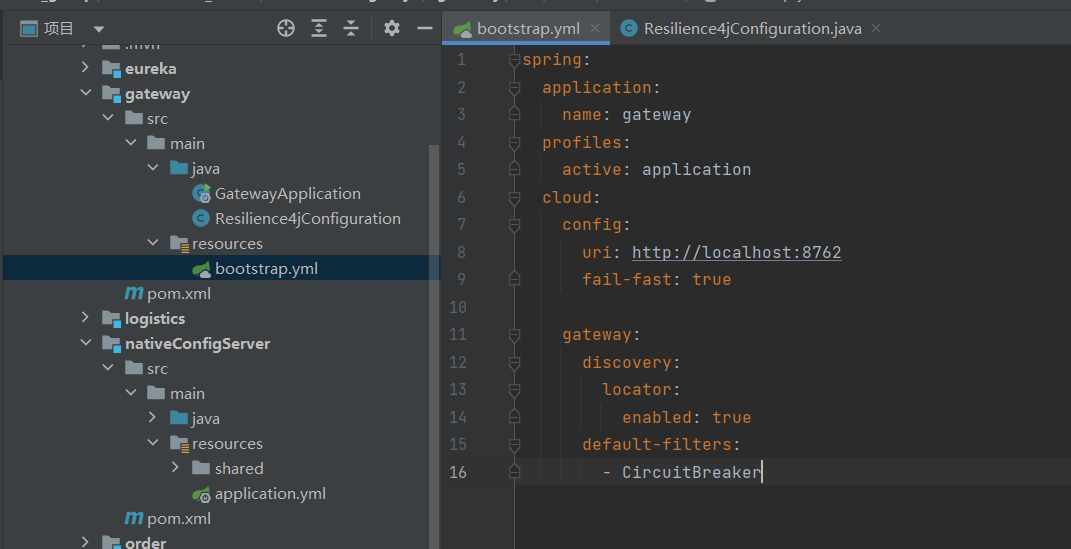
可以看到eureka中被注册的服务：



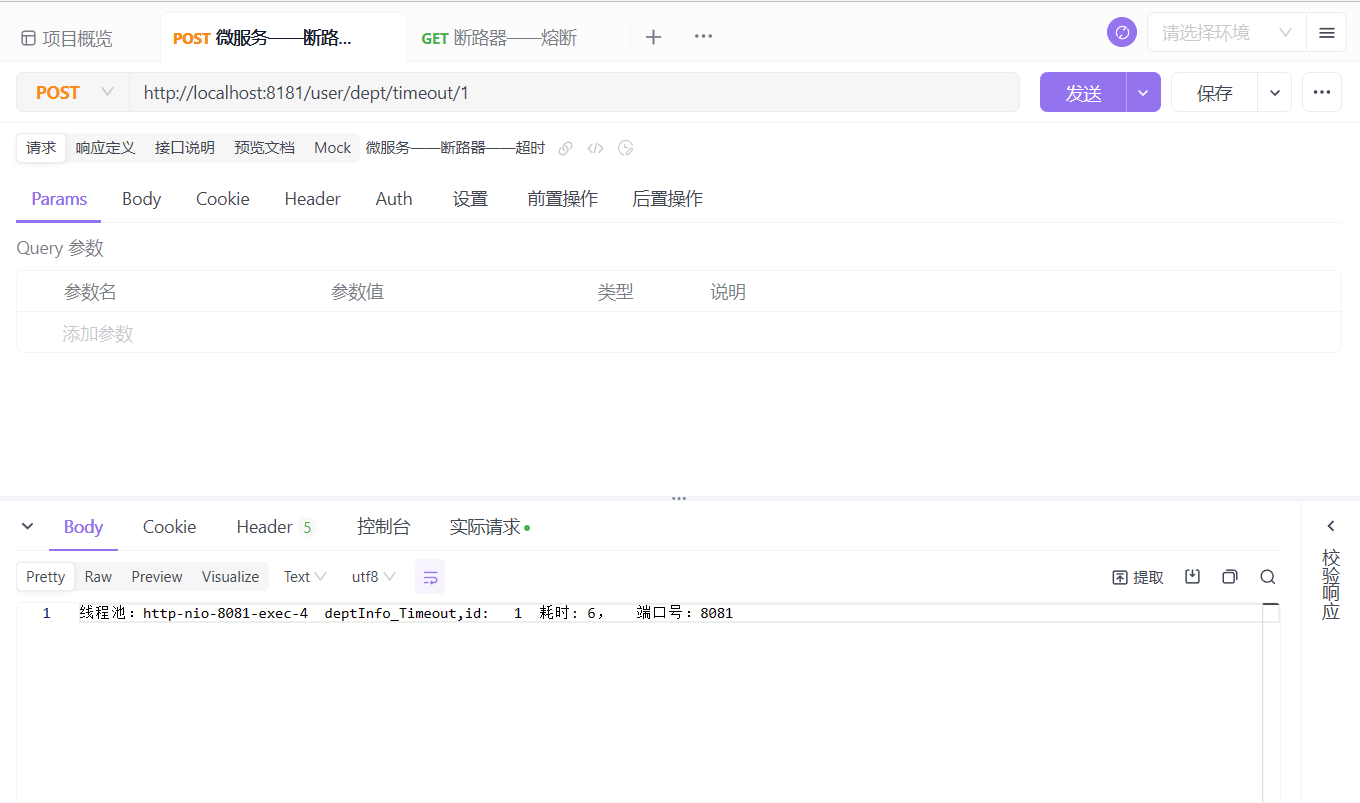
3, Circuit breaker implementation with Resilience4j or Hystrix.

采用Resilience4j进行服务熔断功能的实现。本项目在gateway微服务上配置Resilience4j熔断器，使用CircuitBreaker默认配置，实现粗粒度的全局控制。具体代码实现如下：





在user微服务上增加测试接口，设置6秒的线程延时，没加熔断器之前的结果：



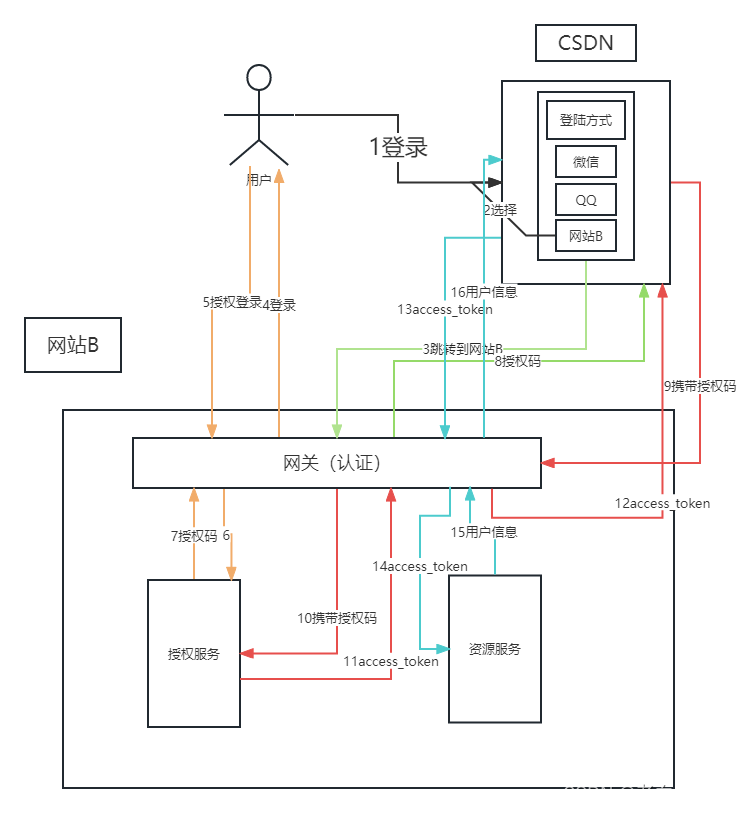
加了熔断器之后的结果：



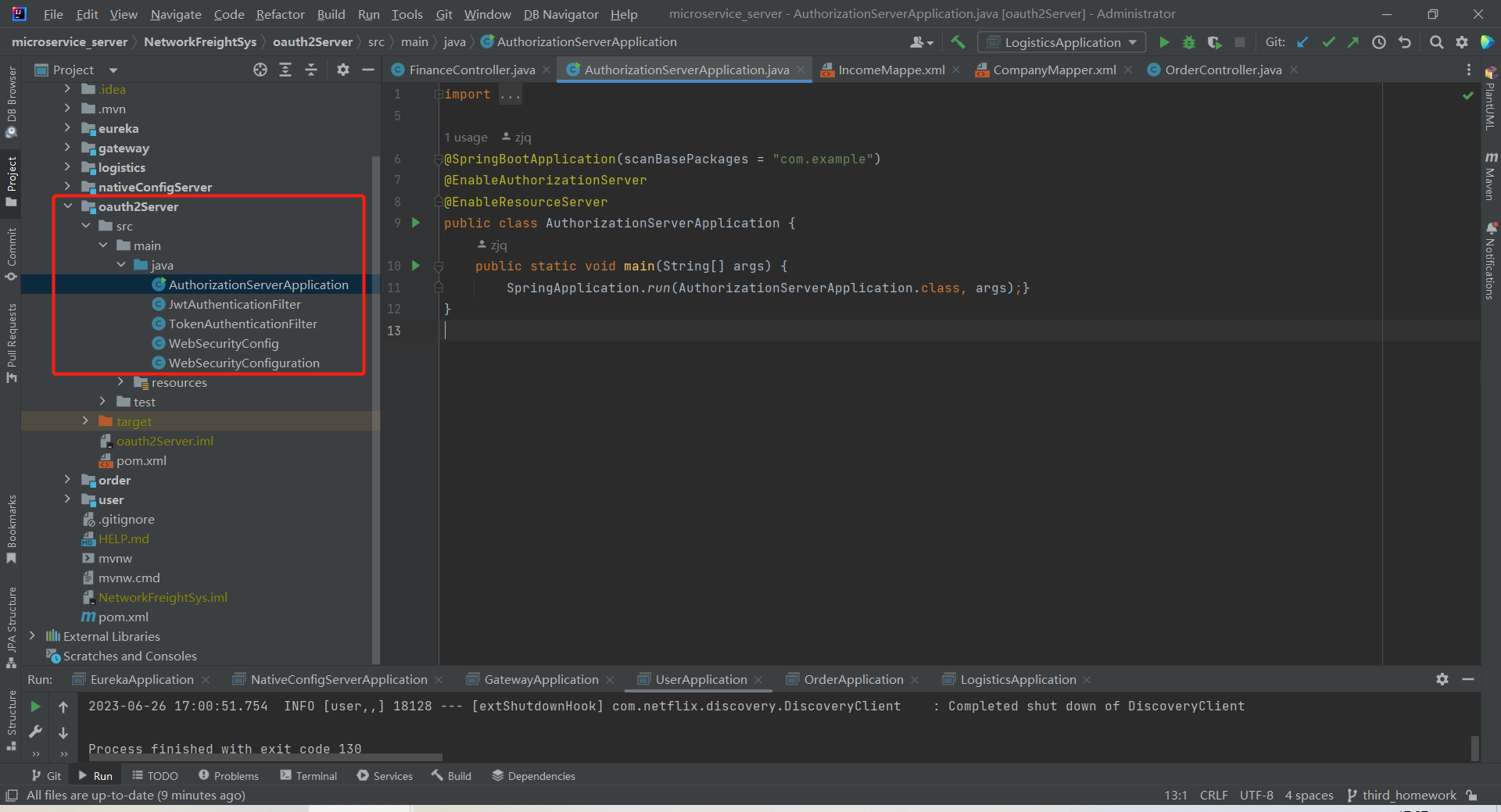
说明网关上的熔断器起了作用，运行良好。

4, Oauth2 authorization server integrated.

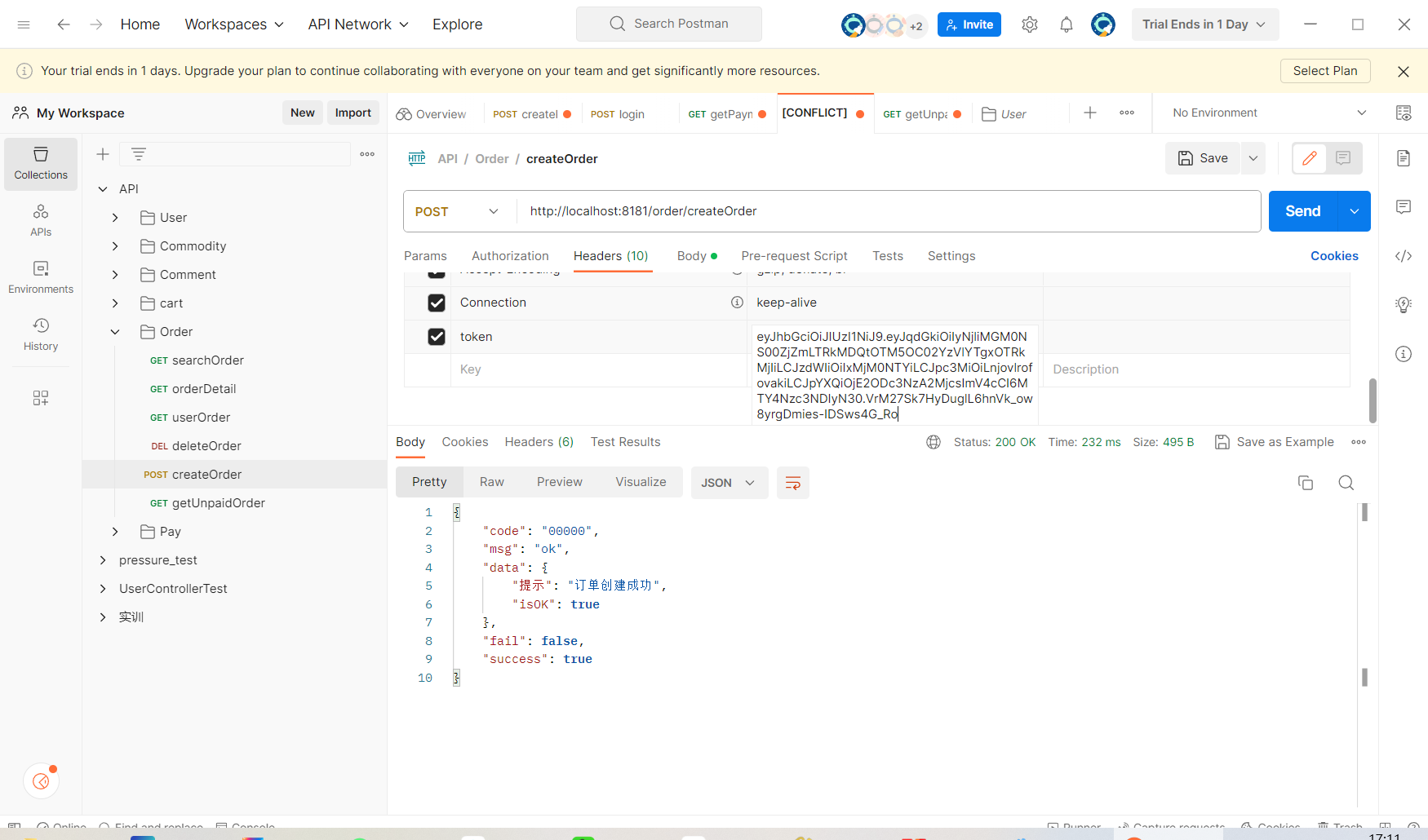
原理：



在oauth2Server中实现oauth2集成进spring cloud实现认证授权：

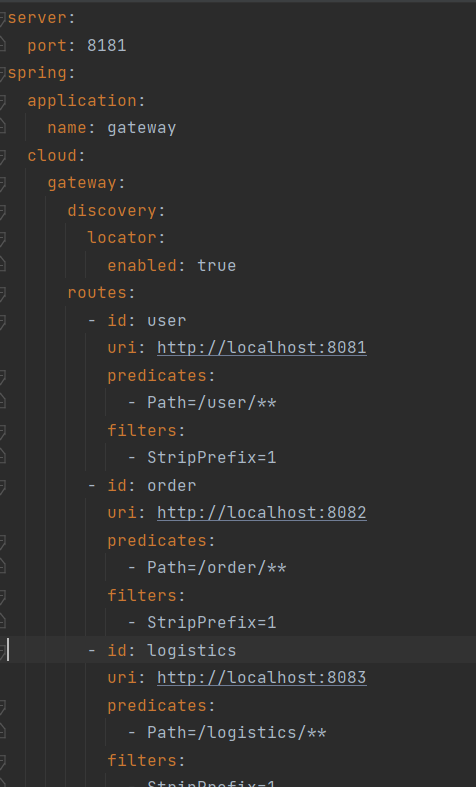


下面是一个示例截图，网关认证token是否有效，进行订单创建成功：

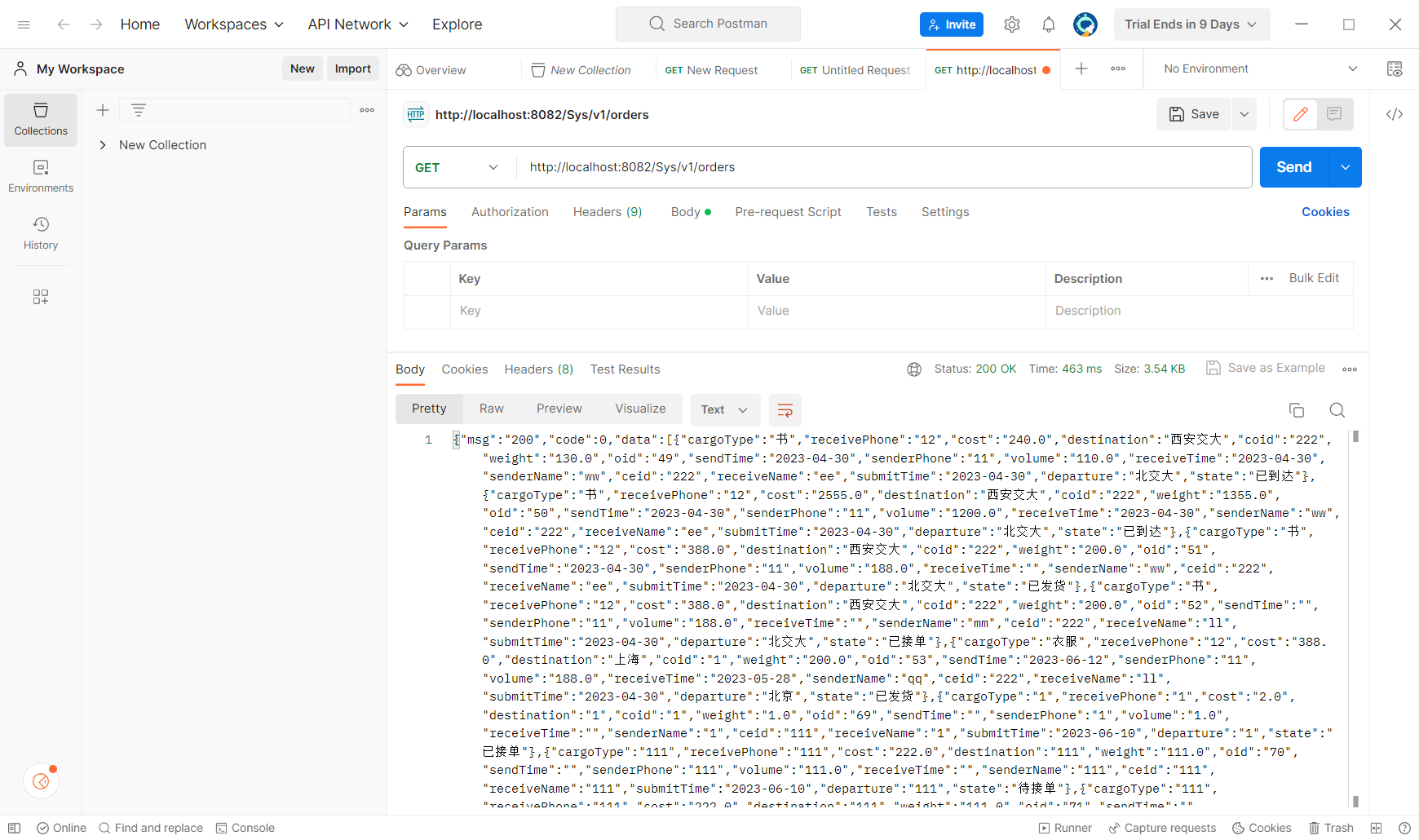


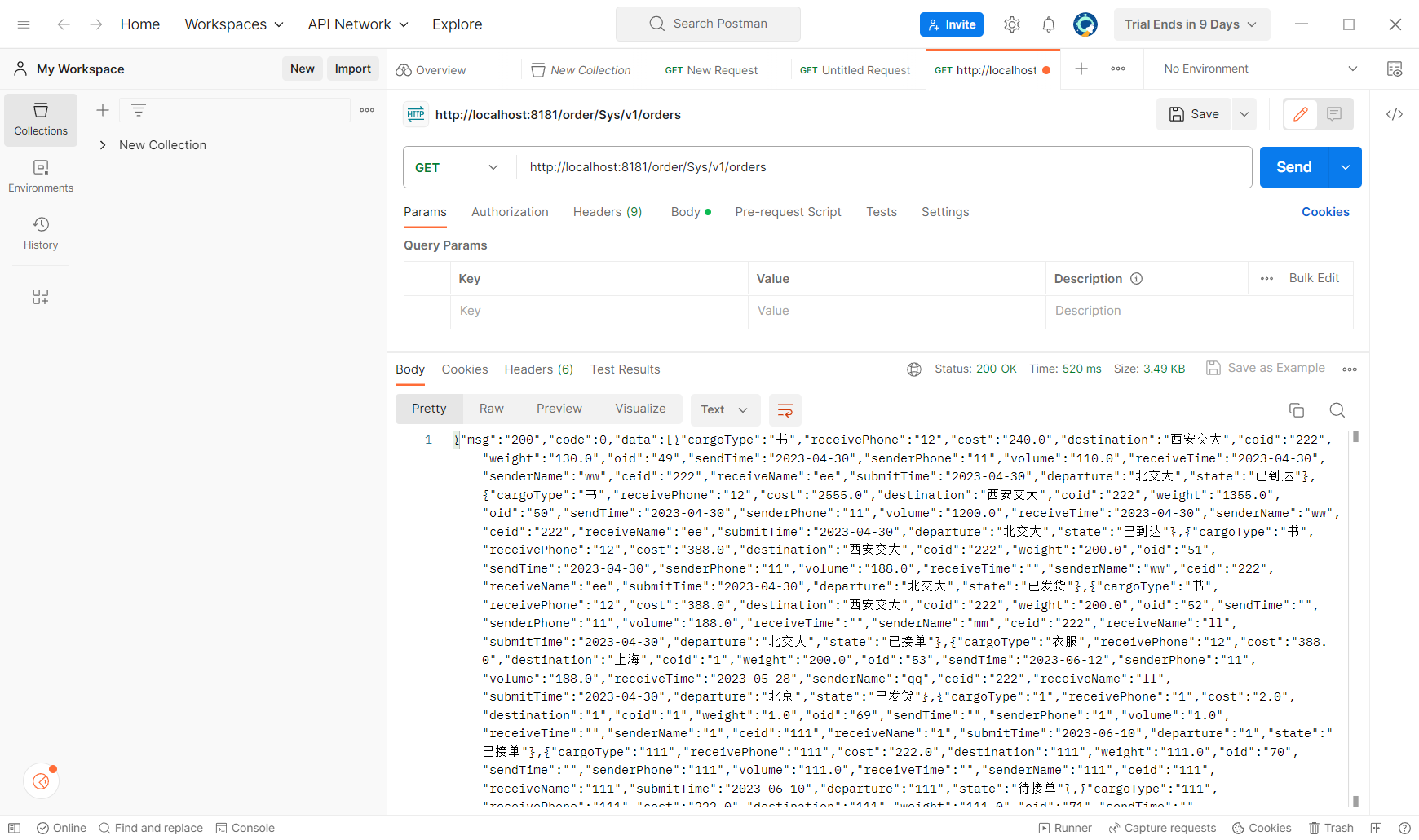
5, Expose API to external users with Gateway

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：

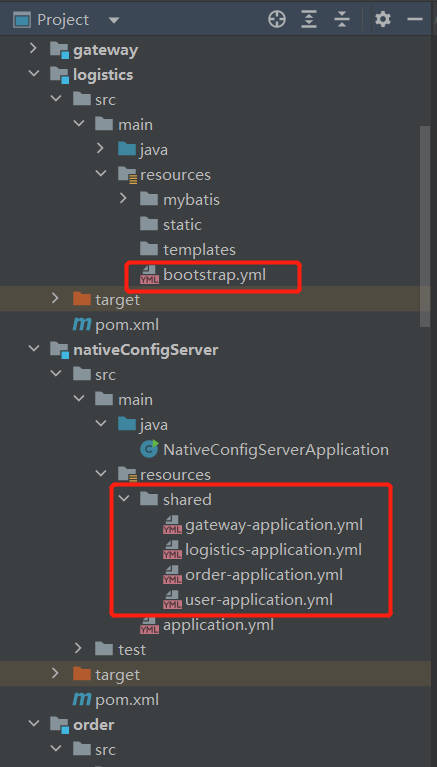


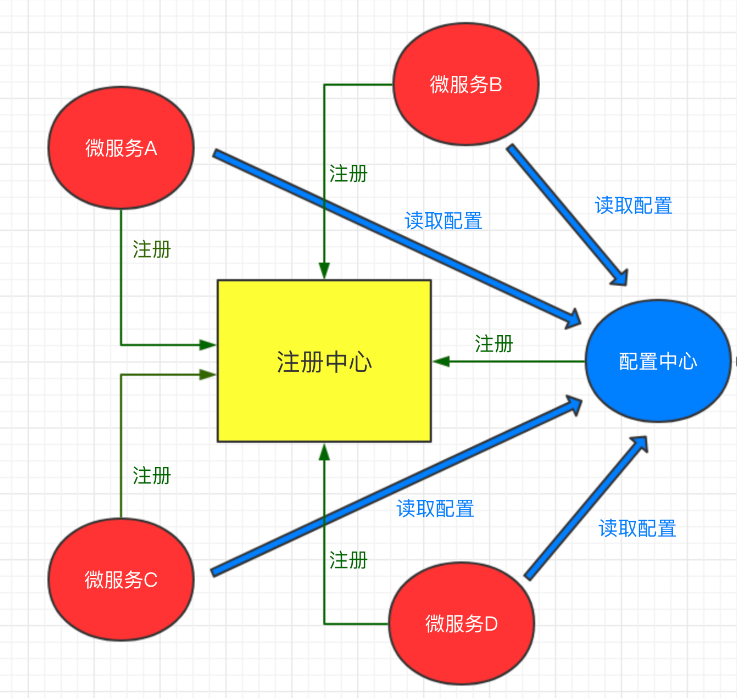


6, Centralized configuration and tracking with Spring cloud config server and sleuth.

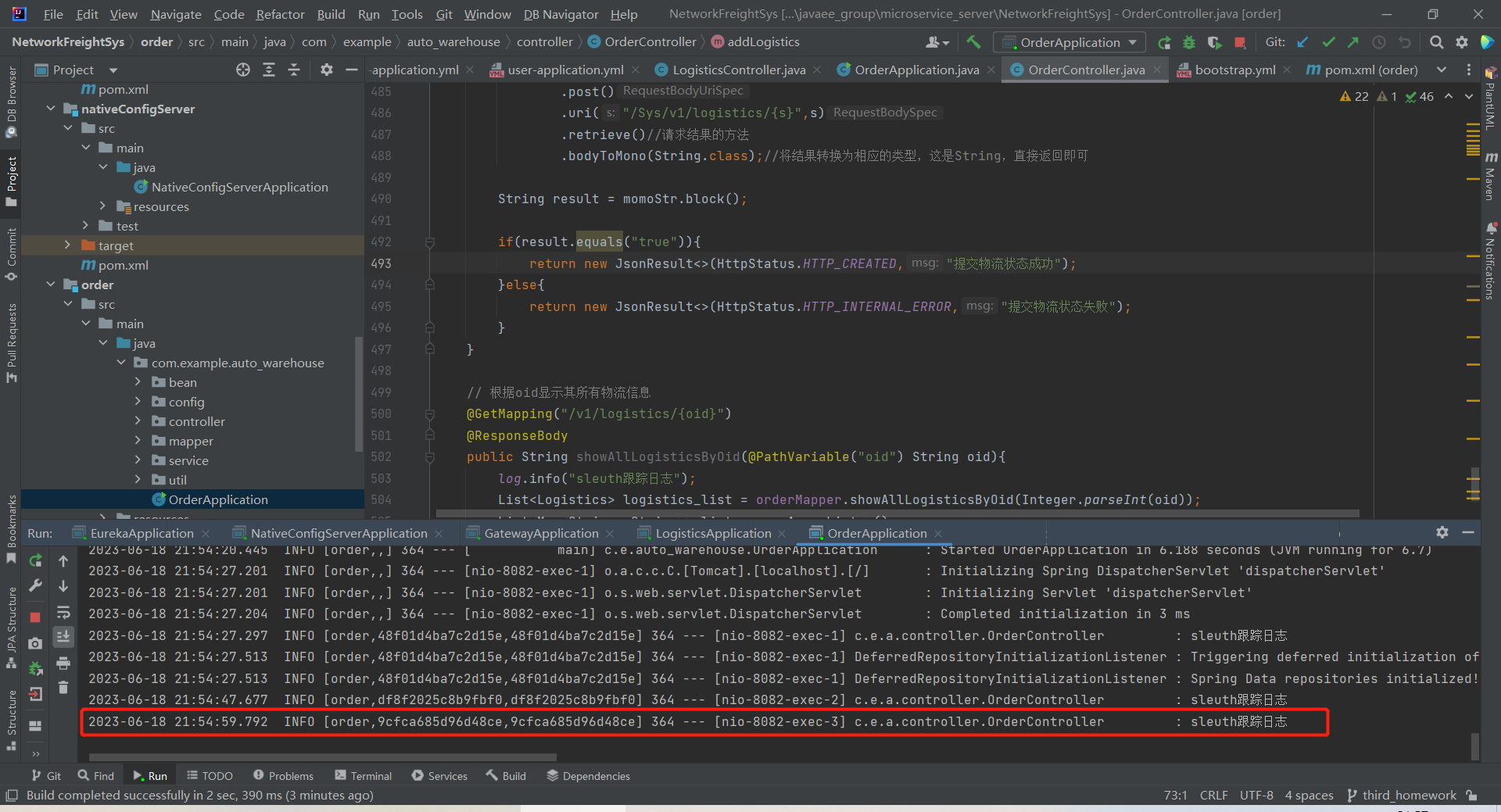
集中配置：

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：





Sleuth跟踪：





***Delivery:***

1, You need to archive the project source code and running and testing screen capture to a github repository.

2, Provide a design report.

3, Hand in due to 2023-6-xx

Reference:

Assignment 4, Event-notification for microservices

In this assignment, you are asked to further improve your assignment 3 project by adding the following features:

*1, Integration of microservices with event-driven brokers.*

*2, Distributed configuration with spring cloud config server.(optional)*

*3, Distributed log tracing with spring cloud sleuth or Zipkin (optional)*

***Requirements:***

1, You need to rebuild or develop microservices which are integrated with (kafka) message-queue, to form an event-driven architecture.

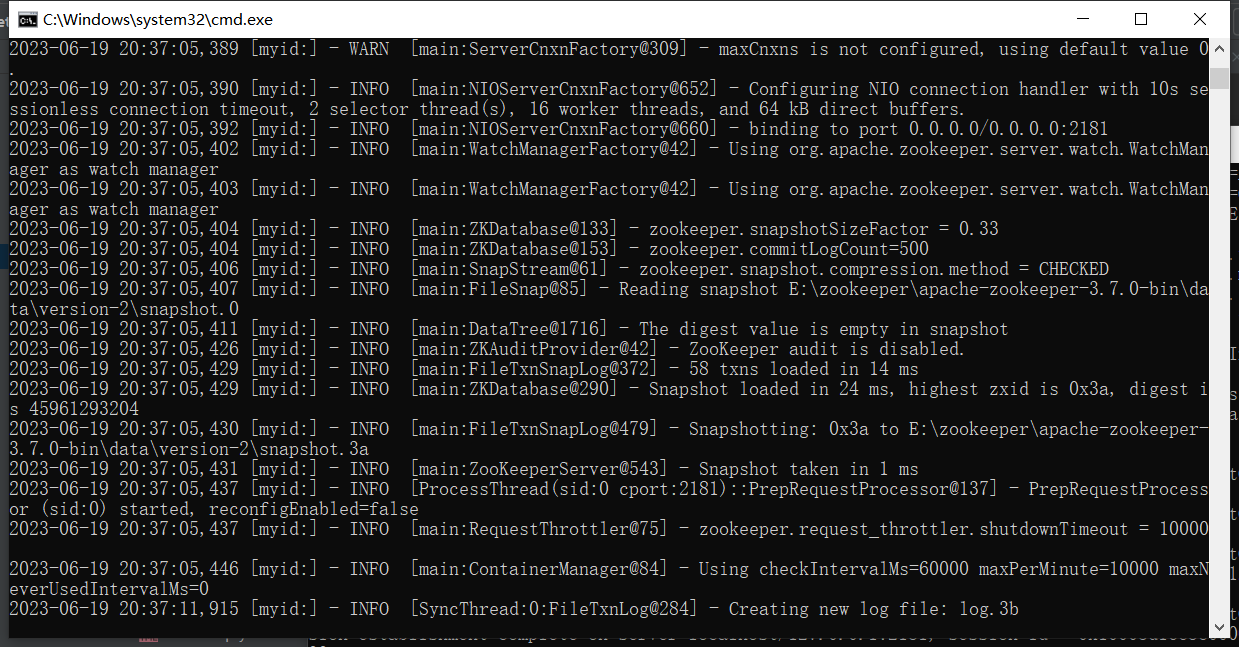
1.运行步骤：

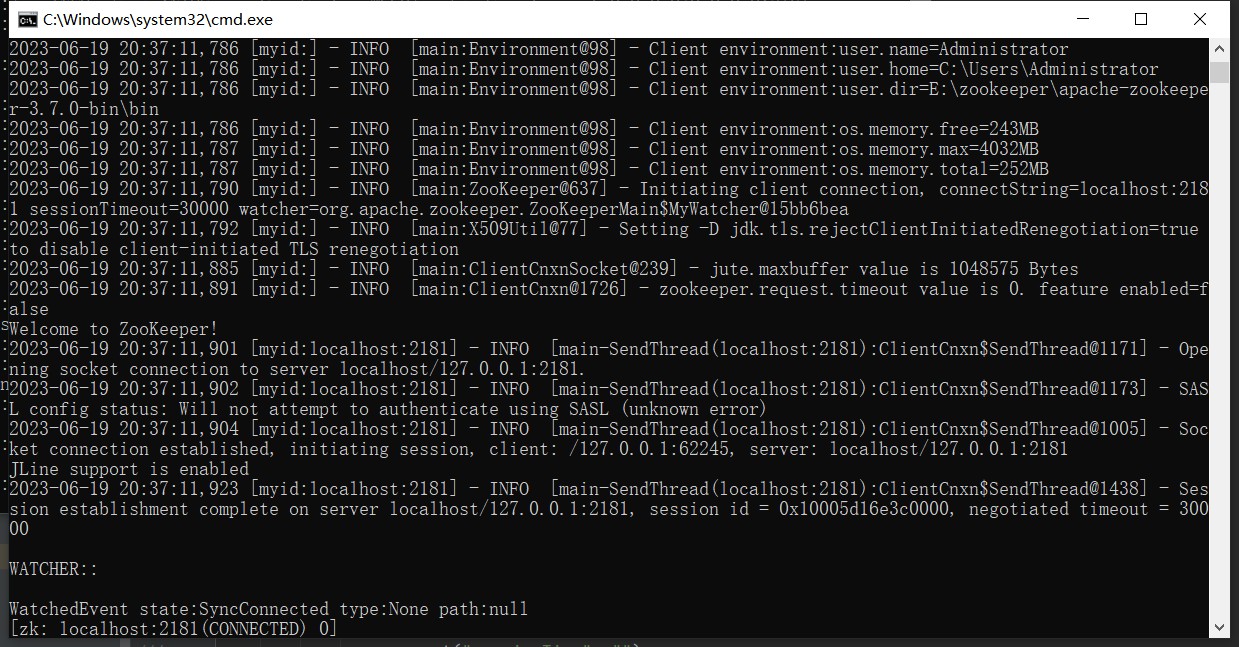
（1）先启动zookeeper和kafka服务器

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

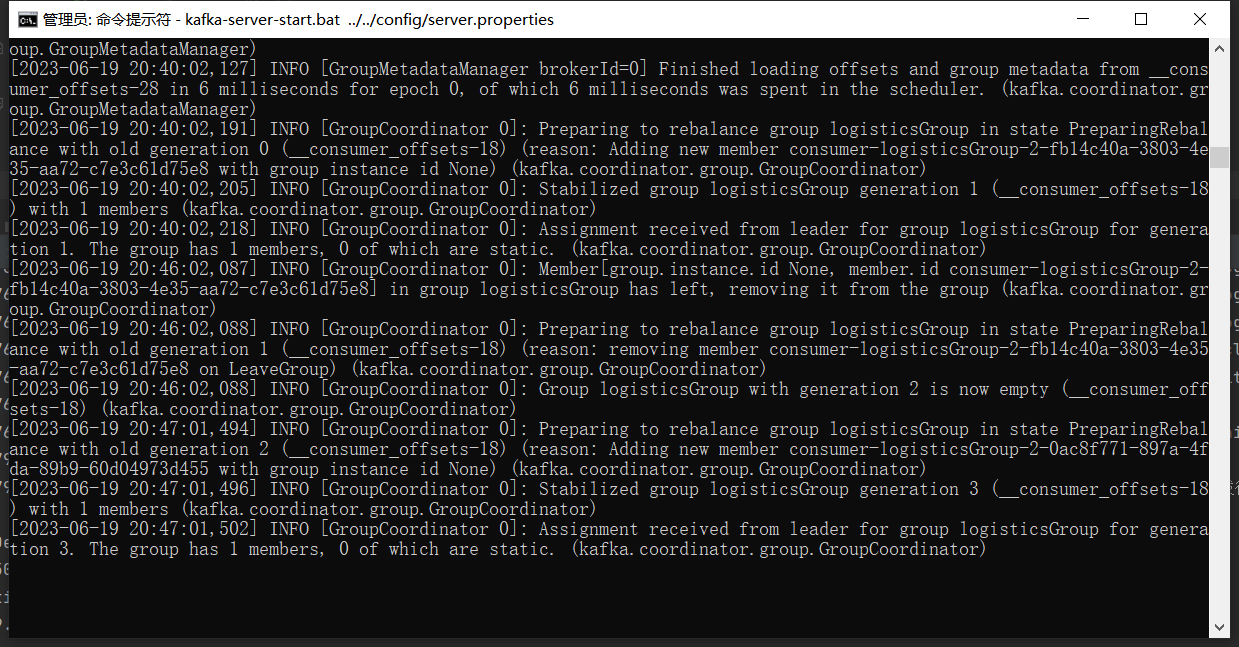
（3）启动order服务和logistics服务（需要保证order在logistics服务之前启动）

下面是zookeeper启动图：



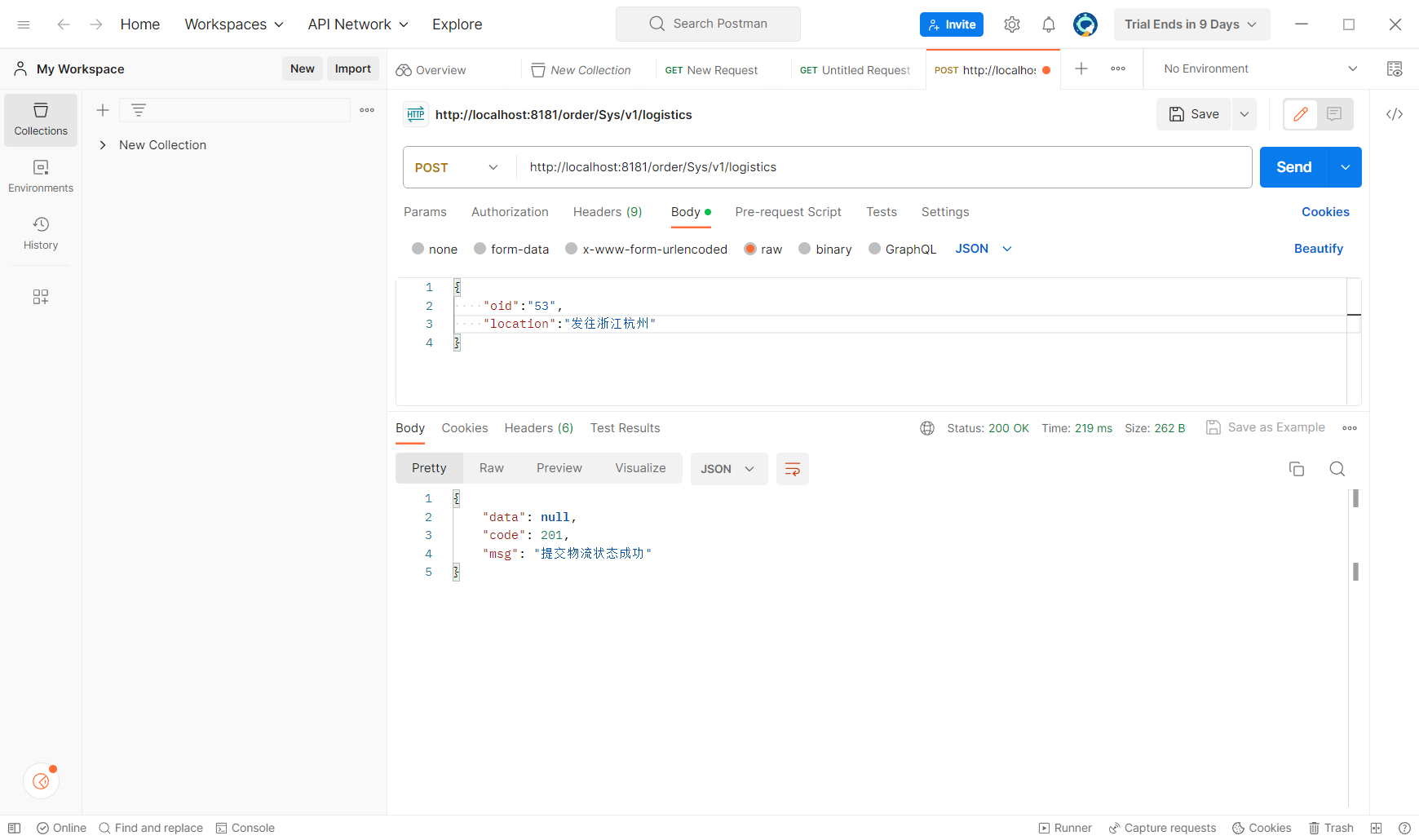


下面是kafka服务器运行图：

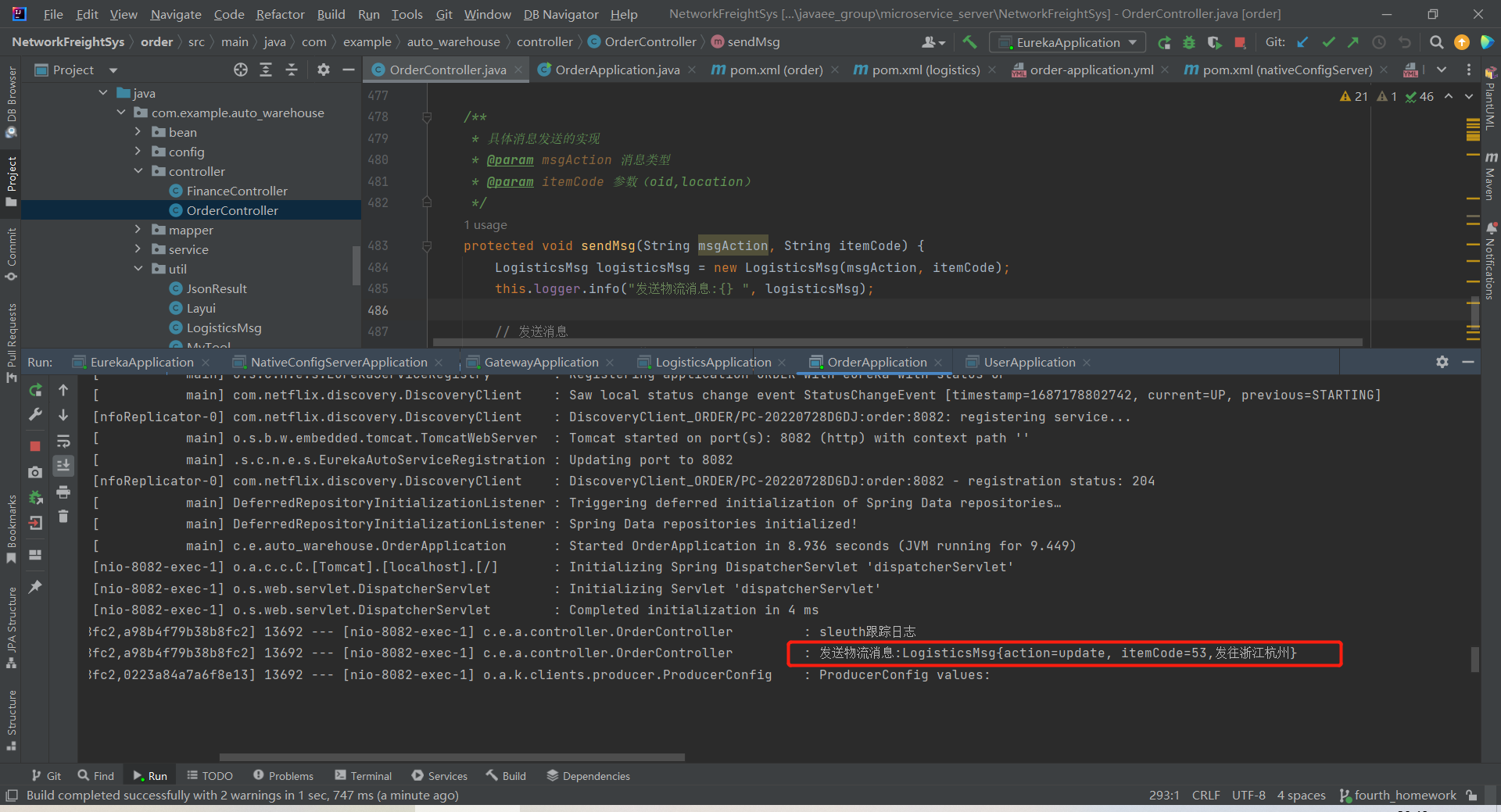


2.说明：

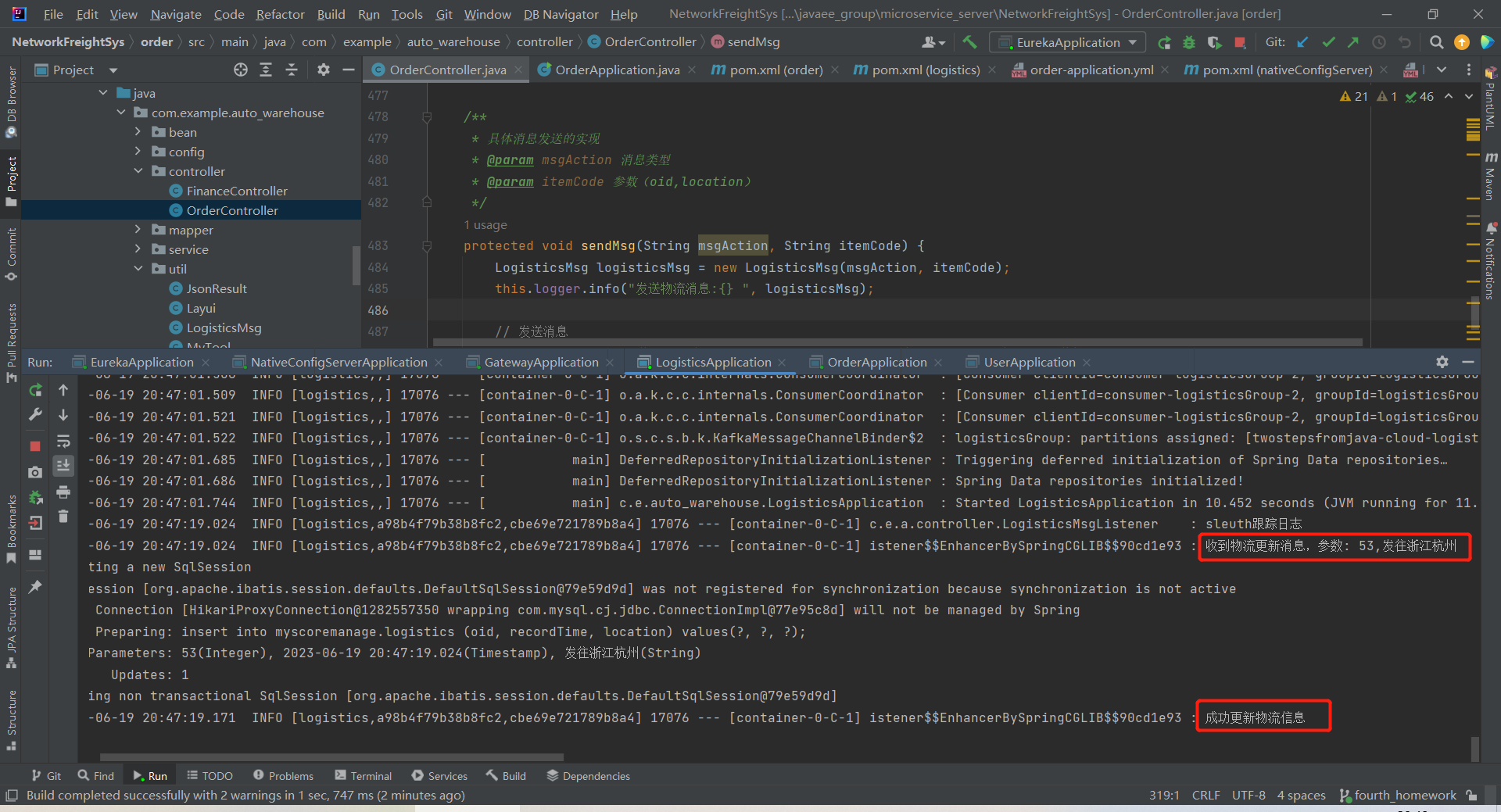
在order和logistics服务中运用kafka消息队列实现事件驱动，比如，调用order服务中的http://localhost:8181/order/Sys/v1/logistics接口，同时传递oid和location参数：



Order服务被调用接口后，向logistics服务发送kafka消息：

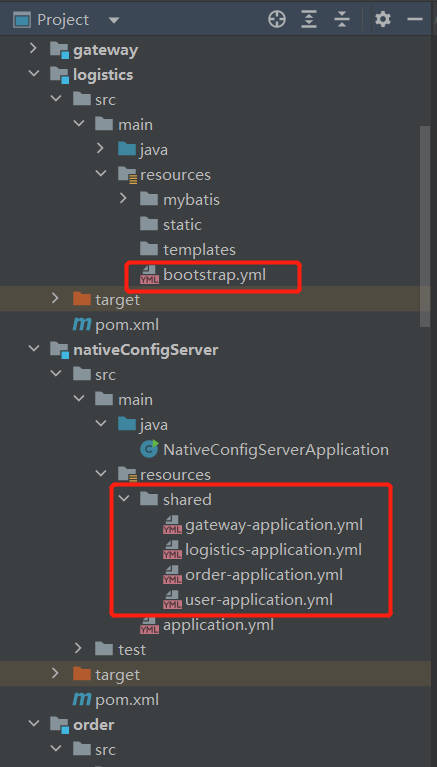


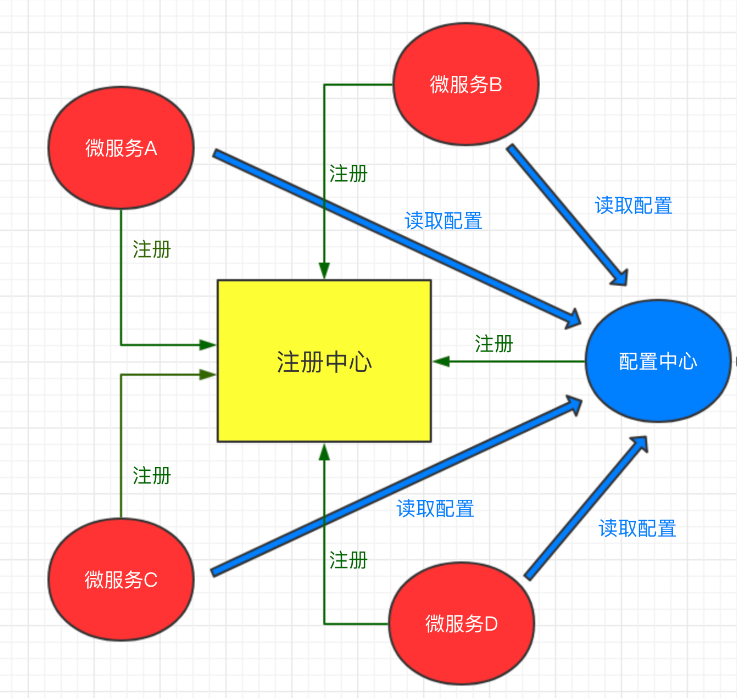
Logistics服务的消息监听器LogisticsMsgListener从消息队列中收到消息，进行物流更新信息处理，并打印出更新成功的信息：



2, Optional, integrating spring cloud config server for centralized configuration.

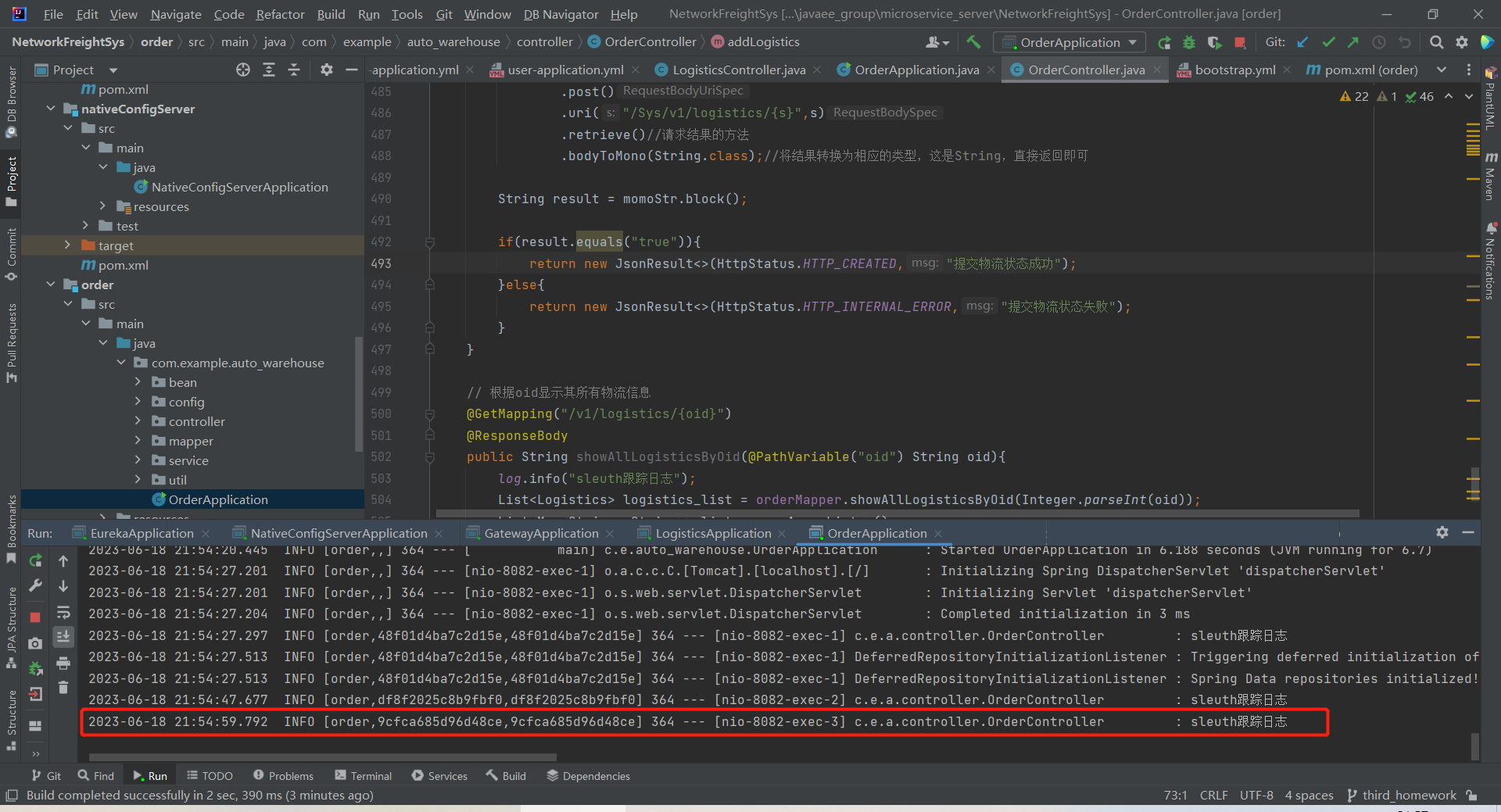
采用Spring Cloud Config server进行服务的集中配置，各个服务通过bootstrap.yml获取到自己的application.yml和nativeConfigServer服务的信息，各服务的application.yml统一放置于nativeConfigServer的shared文件夹下：





3, Optional, integrating Sleuth for centralized log tracing.

Sleuth跟踪：





***Delivery:***

1, You need to archive the project source code and running and testing screen capture to a github repository.

2, Provide a design report.

3, Hand in due to 2023-6-xx

Reference: