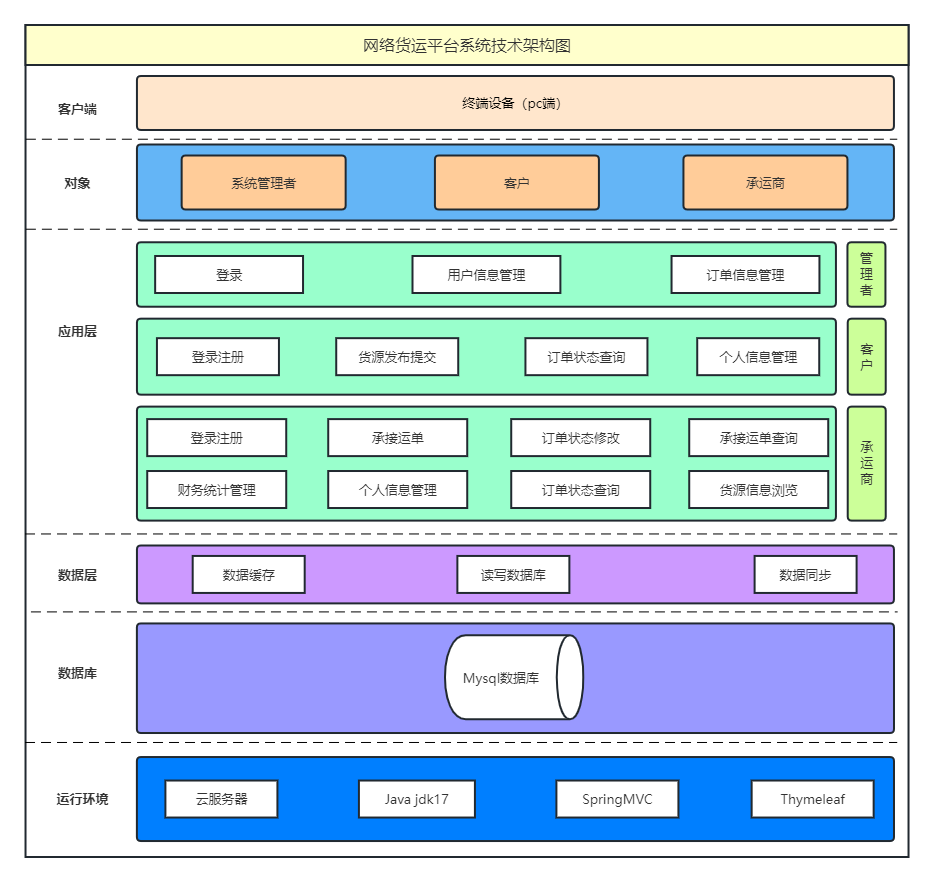
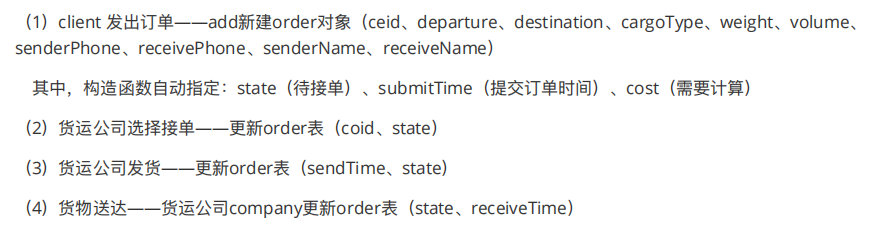
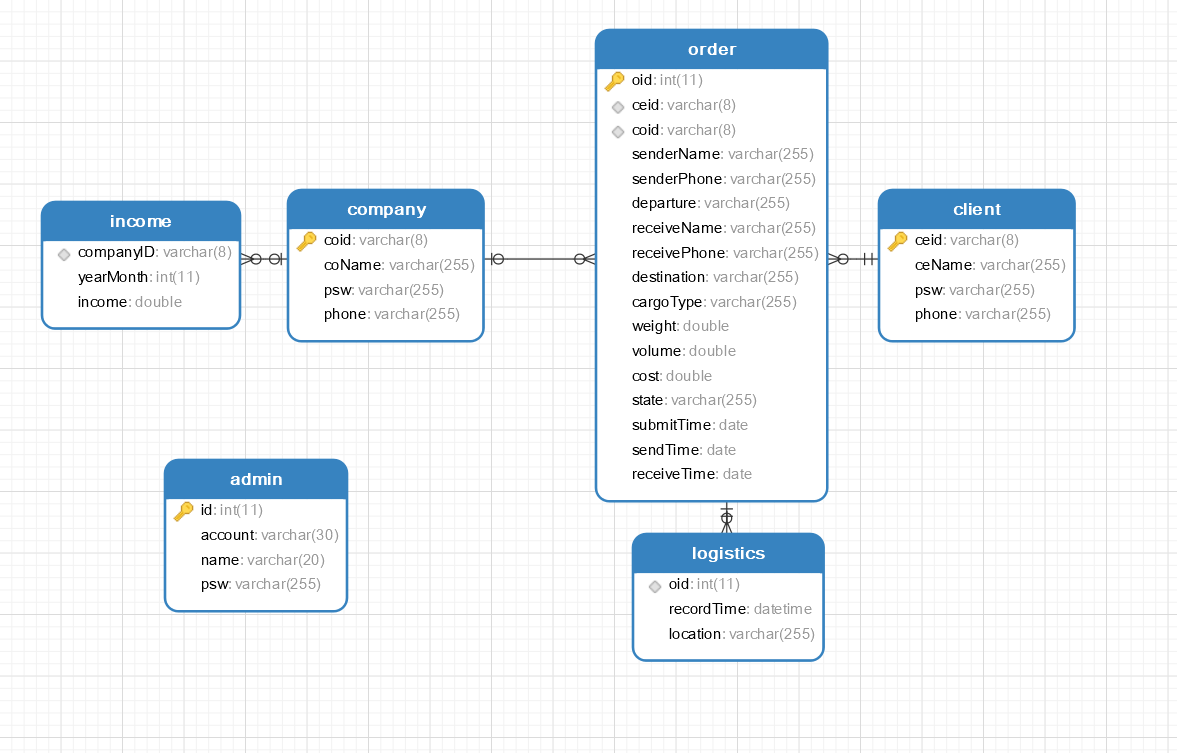
1. 技术架构图



1. 业务流程



1. 货物运输过程期间——货运公司可以录入物流信息新建logistics对象
2. 数据库设计



（加粗代表主键）

客户client表

**ceid**：客户id

ceName：客户姓名

psw：客户登录密码

phone：客户联系⽅式

货运公司company表

**coid**：货运公司id

coName：货运公司姓名

psw：货运公司登录密码

phone：货运公司联系⽅式

订单order表：

**oid**：订单id（⾃增1）

ceid：提交订单的⽤户id

coid：接单的货运公司（初期为null）

senderName：发件⼈名

senderPhone：发件⼈电话

departure：发件地址

receiveName：收件⼈名

receivePhone：收件⼈电话

destination：收件地址

cargoType：货物种类weight：货物重量

volume：货物体积

cost：运费

state：货物状态（待接单/已接单/已到达）

submitTime：提交订单时间

sendTime：发货时间

receiveTime：到达时间

收入income表：

companyID：货运公司id（外键）

yearMonth：年月

income：某年某月的收入

物流logistics表：

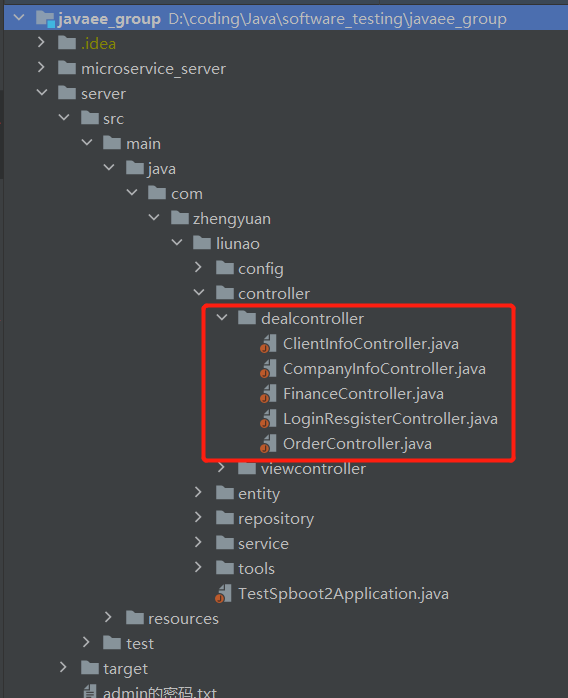
oid：订单id（外键）

recordTime：记录物流信息的时间

location：货物的途径地信息

1. 重构后端API为Restful API设计规范

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）

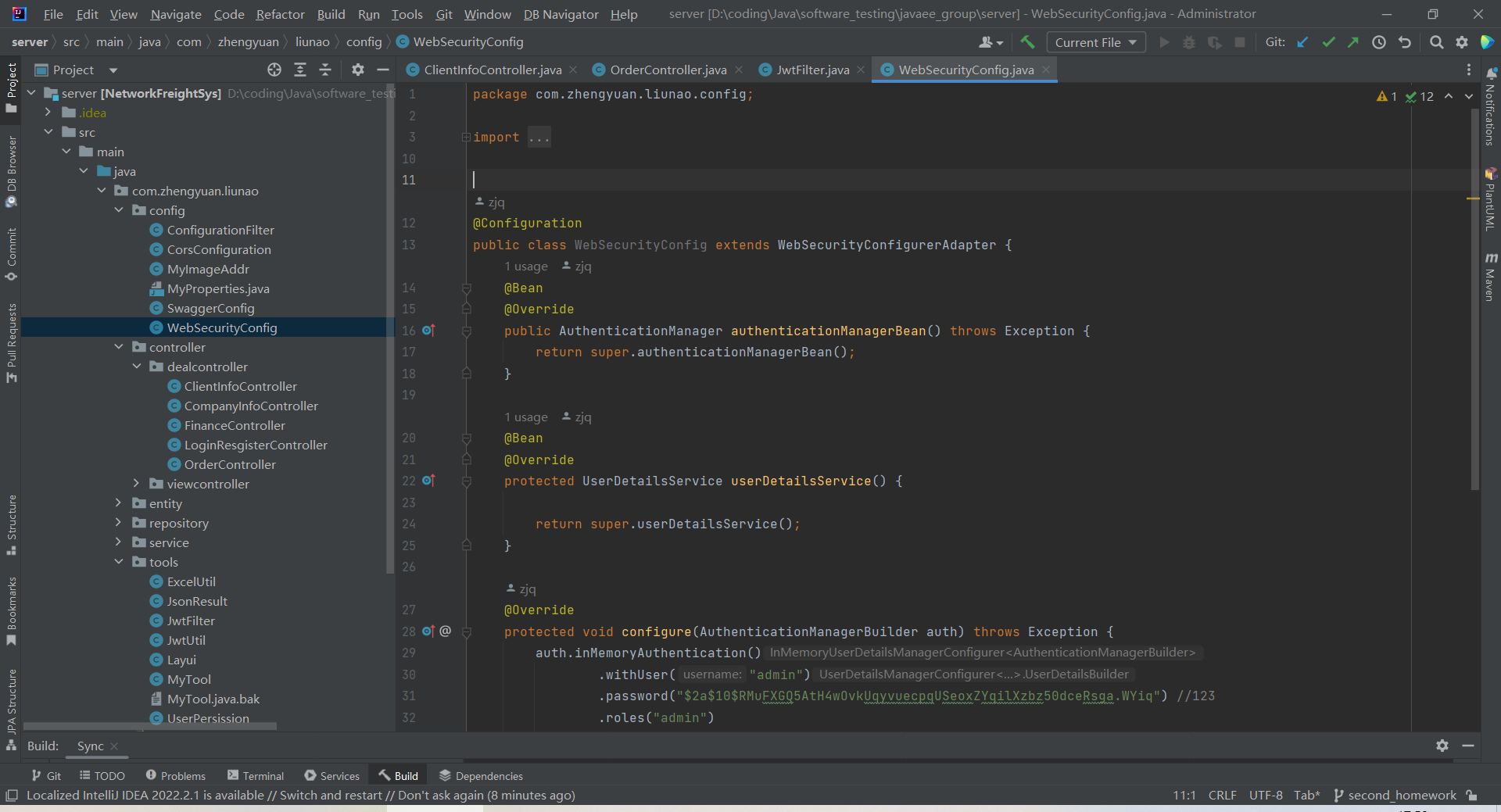




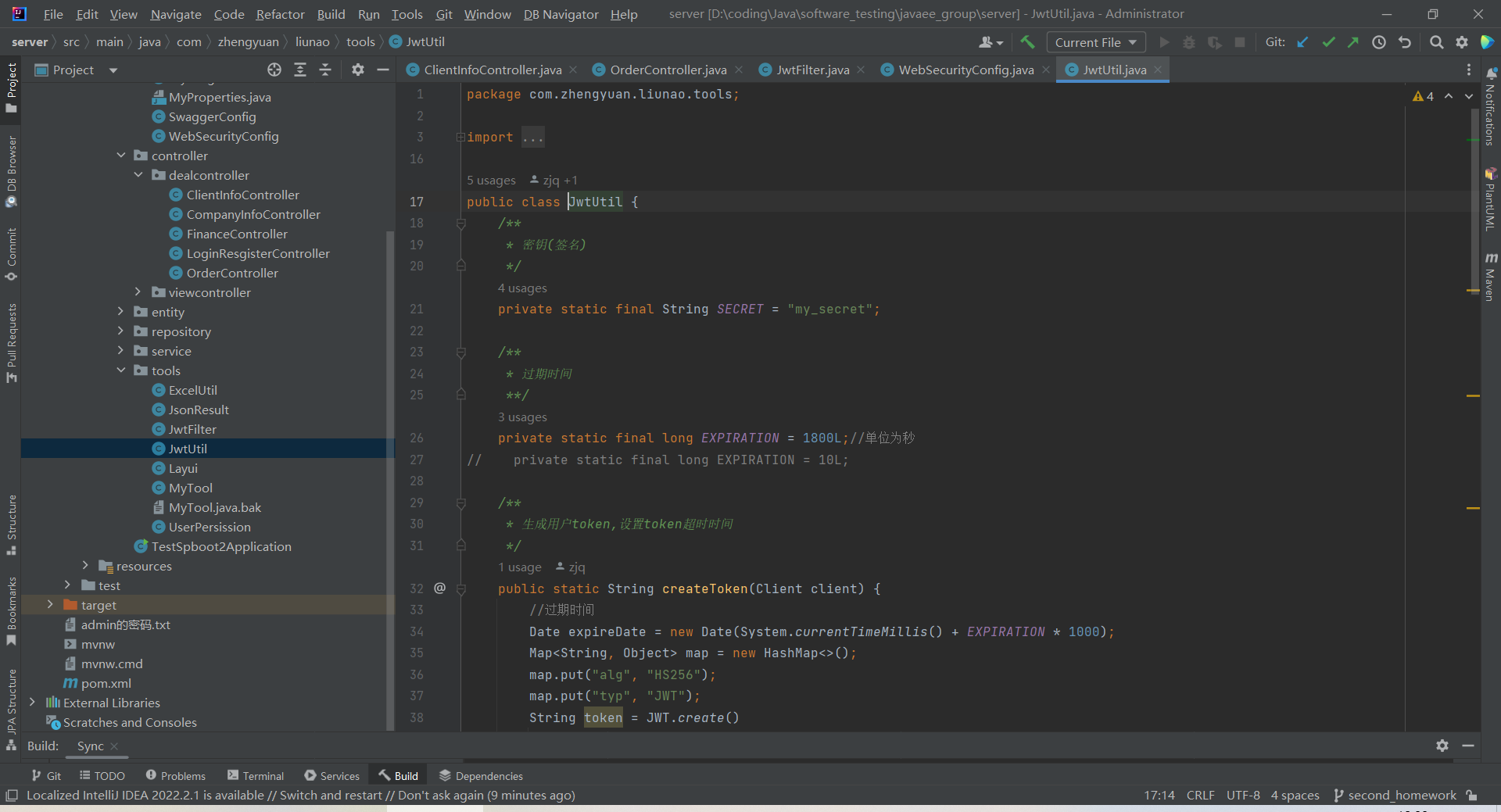
1. 使用Spring Security和JWT进行身份认证和授权

具体代码位置如下：

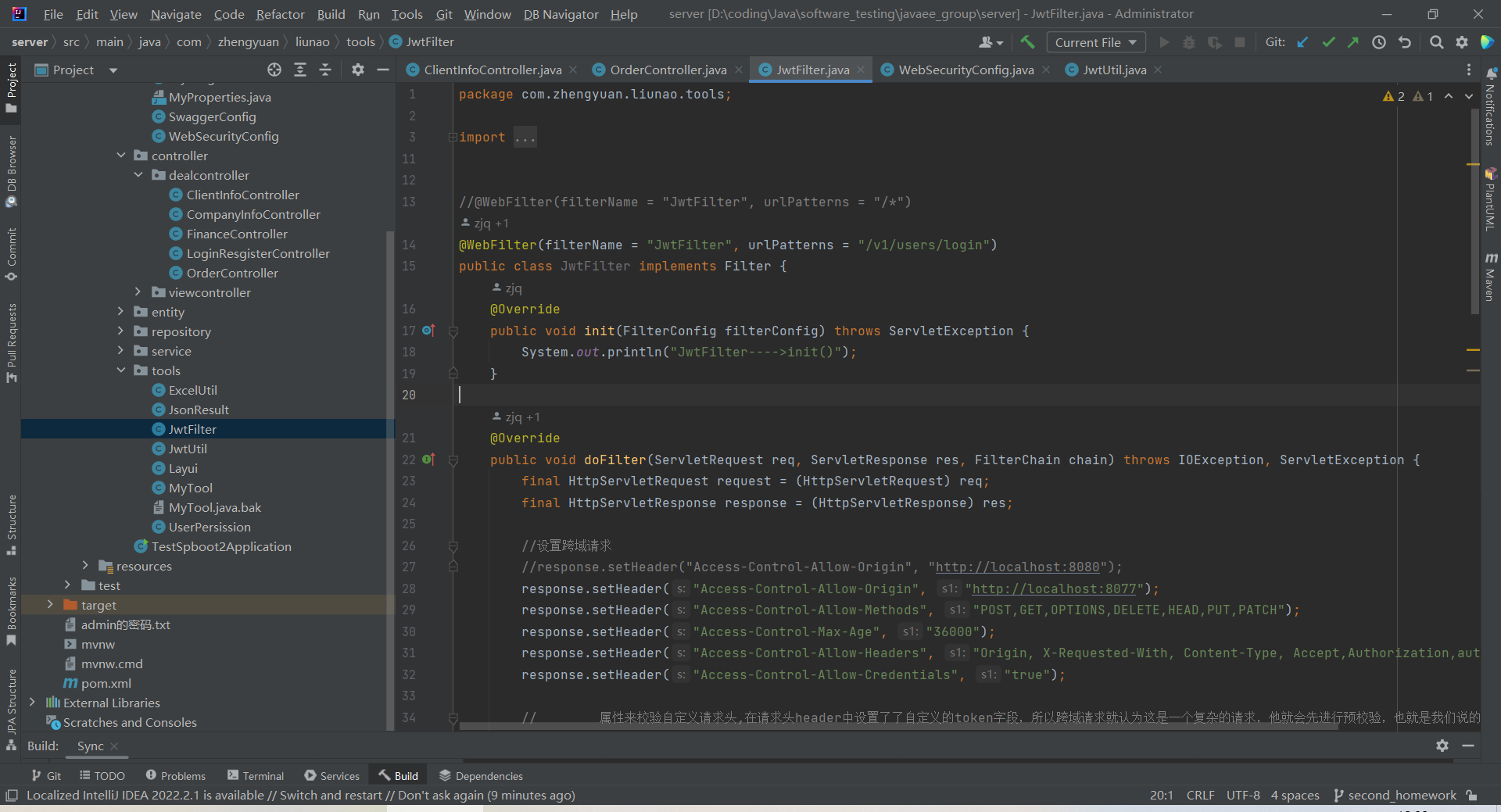
Config包中的WebSecurityConfig.java



tools包中的JwtUtil.java



tools包中的JwtFilter.java

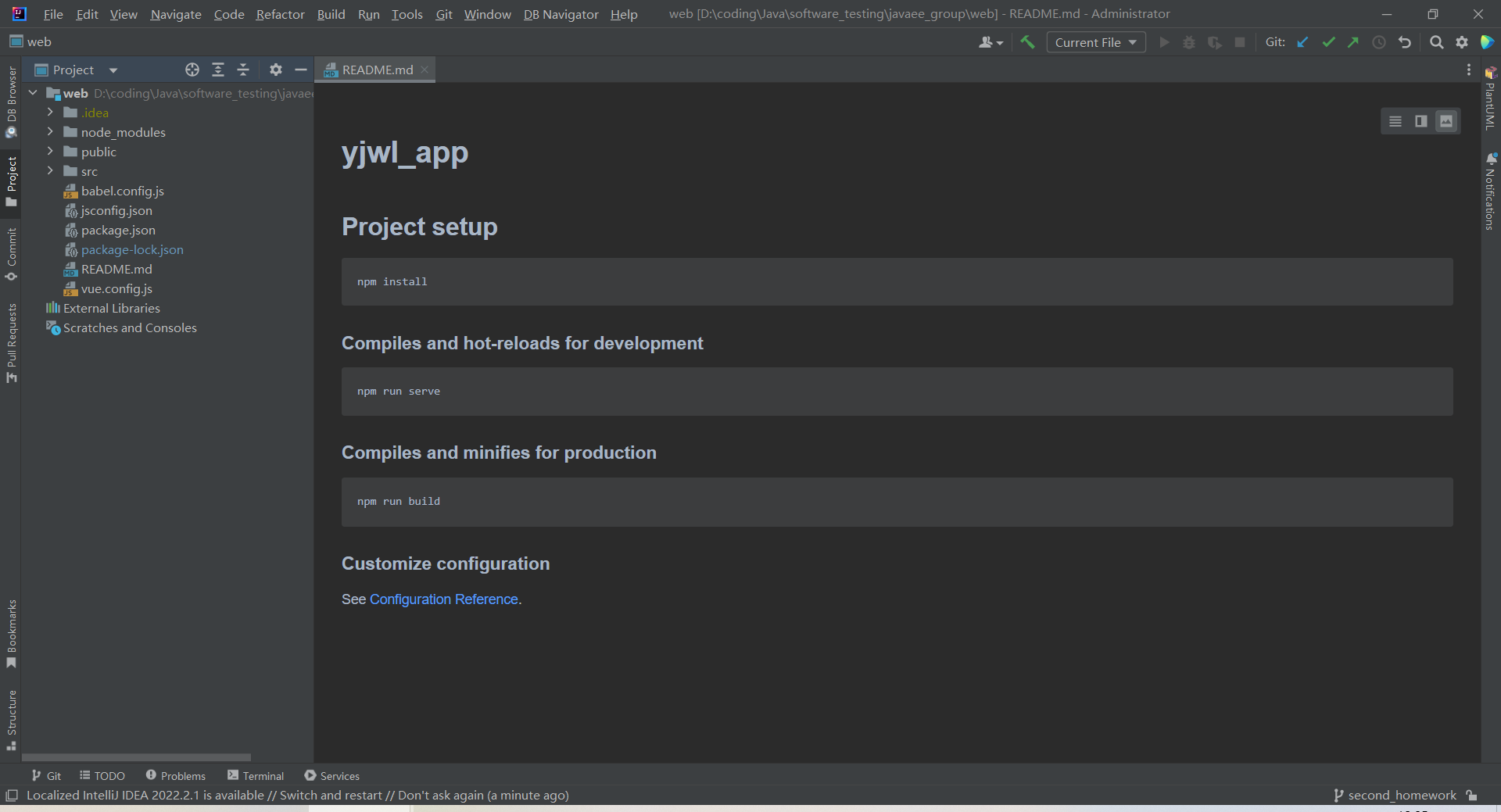


1. 单元测试和集成测试

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。

1. 前端重构为VUE框架，替换第一次作业的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



1. 使用swagger

