System Design Report

姓　　名：李可佳

学　　号：**20301105**

姓　　名：王姝昕

学　　号：**20301118**

姓　　名：刘畅

学　　号：**20301106**

目录

项目概述：

Transport 项目是一个使用前后端分离框架的物流控制系统的网页应用，前端使用 Vue 框架；后端使用 Spring Boot 框架。

本报告将详细介绍基于前后端分离框架的 Spring Boot、Thymeleaf、MyBatis、Bootstrap、JWT、Java security、Vue、Element plus、Restful API 技术实现的系统设计。

技术栈：

Spring Boot

Spring Boot 是基于 Spring 框架的快速开发框架，可以用来创建独立的、生产级别的 Spring 应用程序。它提供了开箱即用的配置，让开发人员可以快速构建和部署应用程序。它的优点包括：

1. 简单易用：Spring Boot 提供了默认的配置和依赖，可以轻松地创建和部署应用程序。
2. 快速开发：Spring Boot 提供了很多开箱即用的功能，包括自动配置、内嵌服务器、开发工具等，可以让开发人员更快地开发应用程序。
3. 易于测试：Spring Boot 支持单元测试和集成测试，可以帮助开发人员更好地测试应用程序。
4. 生产级别：Spring Boot 提供了健康检查、指标监控等功能，可以让应用程序更易于管理和维护。

Thymeleaf

Thymeleaf 是一个基于 Java 的模板引擎，可以用于 Web 和非 Web 环境中的模板渲染。它的优点包括：

1. 简单易学：Thymeleaf 的语法类似于 HTML，易于理解和学习。
2. 可以处理 HTML 和 XML：Thymeleaf 可以处理 HTML 和 XML 等多种模板格式，非常灵活。
3. 可以与 Spring 框架集成：Thymeleaf 可以与 Spring 框架集成，可以方便地在 Spring 应用程序中使用。
4. 安全可靠：Thymeleaf 可以防止跨站点脚本攻击（XSS），可以保障应用程序的安全。

MyBatis

MyBatis 是一个持久层框架，它可以将 Java 对象映射到数据库表中，实现 Java 对象与数据库之间的交互。

它的优点包括：

1. 灵活性高：MyBatis 可以根据开发人员的需求进行配置，可以实现灵活的数据库操作。
2. 易于学习：MyBatis 的语法简单明了，易于学习和使用。
3. 可以与 Spring 框架集成：MyBatis 可以与 Spring 框架集成，可以方便地在 Spring 应用程序中使用。
4. 性能优秀：MyBatis 可以实现高效的数据库操作，可以提高应用程序的性能。

MySQL

MySQL 是一个开源的关系型数据库管理系统，支持多种操作系统。

它的优点包括：

1. 简单易用：MySQL 的安装和配置非常简单，易于使用。
2. 可靠性高：MySQL 具有高可靠性和稳定性，可以保证数据的安全。
3. 性能优秀：MySQL 可以实现高效的数据库操作，可以提高应用程序的性能。
4. 易于扩展：MySQL 可以轻松地扩展到多台服务器上，支持高可用性和负载均衡。

Bootstrap

Bootstrap 是一个流行的前端框架，用于快速构建响应式和移动优先的 Web 页面。

它的优点包括：

1. 快速开发：Bootstrap 提供了大量的预定义组件和样式，可以快速构建 Web 页面。
2. 响应式设计：Bootstrap 可以自动适应不同的屏幕尺寸和设备类型，可以为用户提供更好的体验。
3. 浏览器兼容性：Bootstrap 兼容大多数现代浏览器，可以为用户提供一致的体验。
4. 可定制性：Bootstrap 可以根据开发人员的需求进行定制，可以实现个性化的 Web 设计。

JWT

JWT（JSON Web Token）是一种用于身份验证和授权的开放标准。它是一种基于 JSON 的安全令牌，用于在不同系统之间安全地传递声明。JWT 的优点包括：

1. 简单性：JWT 使用 JSON 格式，易于理解和使用。
2. 自包含性：JWT 令牌包含所有必要的信息，如用户身份信息、权限和过期时间等。这意味着不需要在服务器端存储会话信息，减轻了服务器负担。
3. 跨平台和跨语言支持：JWT 是基于标准的开放协议，可以在不同的平台和语言之间进行交互。
4. 可扩展性：JWT 允许自定义声明，可以根据需要添加额外的信息。

Java security

Java Security（Java 安全）是 Java 平台提供的一组安全功能和 API。它包含各种机制和工具，用于保护 Java 应用程序和数据的安全性。Java Security 的优点包括：

1. 强大的安全性功能：Java Security 提供了一系列功能，如身份验证、访问控制、加密、

数字签名和安全通信等，可以帮助开发人员构建安全可靠的应用程序。

2. 平台无关性：Java Security 是 Java 平台的一部分，因此可以在不同的操作系统和硬件平台上使用。
3. 开发人员友好性：Java Security 提供了简单易用的 API 和工具，使开发人员能够轻松地集成和使用安全功能

Vue

Vue 是一种流行的 JavaScript 前端框架，用于构建用户界面。它具有以下优点：
简单易学：Vue 采用了直观的语法和简洁的 API，使得学习和使用变得容易，即使对于初学者也是如此。

1. 响应式设计：Vue 使用响应式数据绑定的概念，使得数据和 UI 保持同步。当数据发生变化时，Vue 会自动更新相关的 UI 组件，减少了手动操作的复杂性。
2. 组件化开发：Vue 鼓励将应用程序拆分为可复用的组件，每个组件都有自己的模板、逻辑和样式。这种组件化开发的方式使得代码更加清晰、可维护性更强。
3. 生态系统丰富：Vue 拥有庞大的生态系统，提供了各种工具和插件来满足不同项目的需求，如路由管理、状态管理和 UI 组件库等。

Element plus

Element Plus 是一个基于 Vue 的开源 UI 组件库，提供了丰富的可复用组件，用于快速构建现代化的 Web 界面。Element Plus 的优点包括：

1. 丰富的组件库：Element Plus 提供了大量的高质量组件，涵盖了常见的 UI 元素和交互模式，如按钮、表格、表单、对话框等。这些组件可以帮助开发人员快速构建美观、功能强大的界面。
2. 定制化和可扩展性：Element Plus 允许开发人员根据自己的需求进行样式定制和功能扩展。组件的样式和行为都可以通过简单的配置进行自定义。
3. 文档和社区支持：Element Plus 提供了详细的文档和示例，帮助开发人员快速上手和解决问题。同时，它也有活跃的社区支持，可以分享经验和获取帮助。
4. 高质量和可靠性：Element Plus 是一个经过广泛测试和验证的组件库，具有良好的质量和稳定性，可用于各种生产环境。

Restful API

RESTful API（Representational State Transfer API）是一种基于 HTTP 协议的架构风格，用于设计和开发网络应用程序的 API。它具有以下特点：

1. 资源和标识：RESTful API 将应用程序中的数据和功能抽象为资源，并通过唯一的标识符（URI）进行访问。每个资源都有自己的 URI，可以通过 HTTP 方法对其进行操作。
2. 状态转移：RESTful API 使用 HTTP 方法（GET、POST、PUT、DELETE 等）来表示对资源的不同操作，如获取资源、创建资源、更新资源和删除资源。这些操作导致资源的状态转移。
3. 无状态性：RESTful API 是无状态的，即每个请求都是独立的，服务器不会存储客户端

的状态信息。客户端每次请求都需要提供所有必要的信息，服务器仅根据请求进行处理。

4. 统一接口：RESTful API 使用统一的接口约束，包括使用 HTTP 方法进行操作、使用 URI 标识资源、使用 HTTP 状态码表示请求结果，以及使用 MIME 类型进行数据交换。
5. 可扩展性：RESTful API 支持按需扩展和定制。开发人员可以根据应用程序的需求添加新的资源和操作，并通过合适的 URI 进行访问。
6. 可浏览性：RESTful API 可以提供可浏览的 API 文档，使开发人员能够通过浏览器直接了解和测试 API 的功能。

通过使用 RESTful API，开发人员可以构建出具有良好可读性、可扩展性和松耦合性的 Web 服务，实现不同系统之间的数据交换和功能集成。

后端架构设计：

entity 层

实体层。
主要用于**定义与数据库对象应的属性，提供 get/set 方法**,tostring 方法,有参无参构造函数。

dao 层

持久层，主要与数据库交互。
DAO 层**可以**进行数据业务的处理，Dao 层的数据源和数据库连接的参数都是在配置文件中进行配置的。

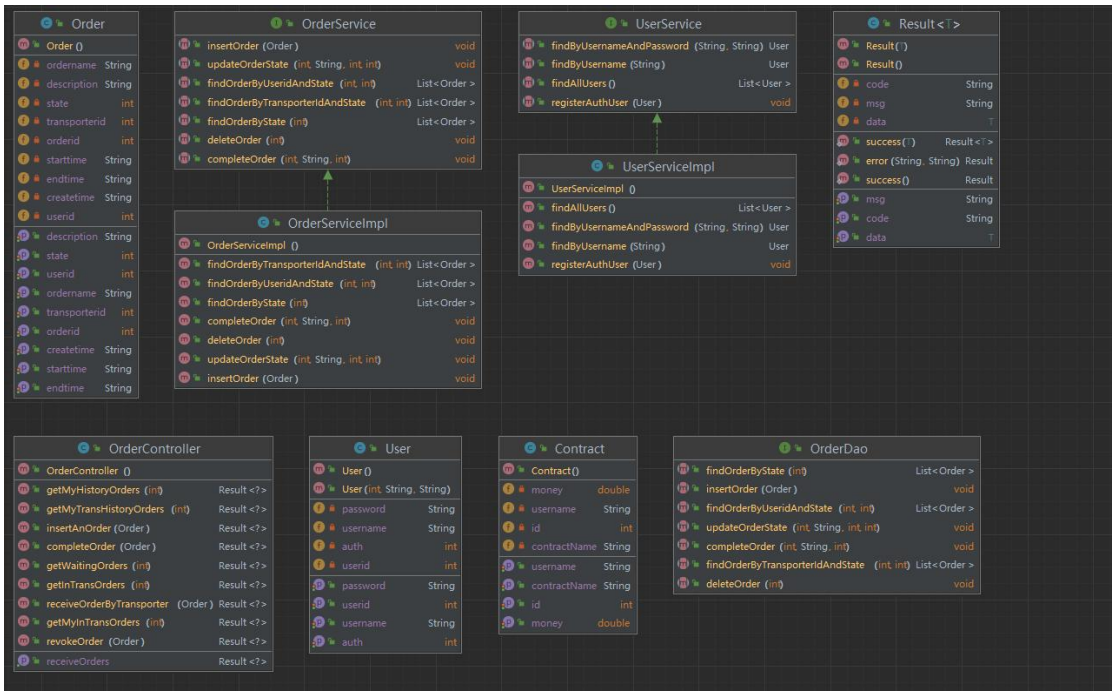
service 层

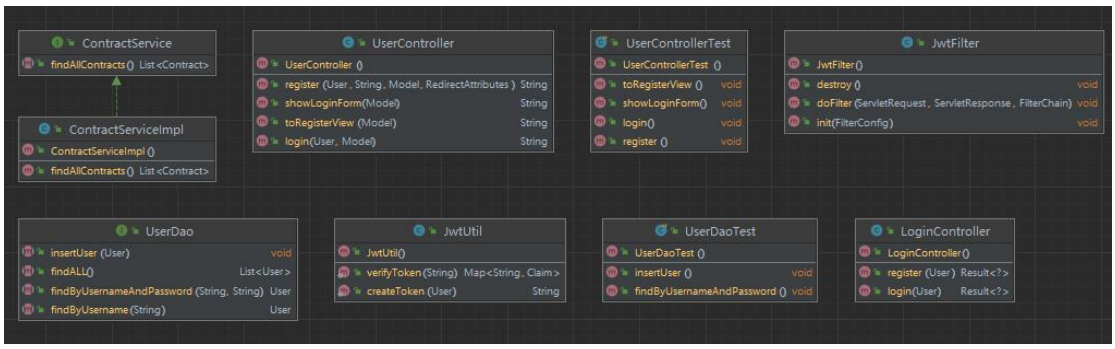
业务层，控制业务。
业务模块的逻辑应用设计，**先设计接口，再创建要实现的类，然后在配置文件中进行配置其实现的关联**。接下来就可以在 service 层调用接口进行业务逻辑应用的处理。

controller 层

控制层 控制业务逻辑。
具体的业务模块流程的控制，controller 层**主要调用 Service 层里面的接口控制具体的业务流程，控制的配置也要在配置文件中进行**。

项目类图：

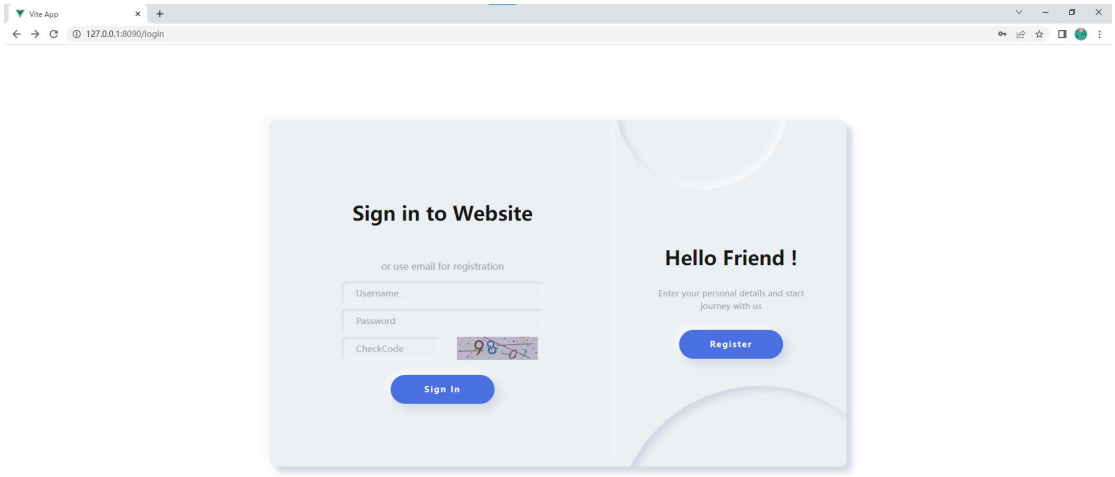


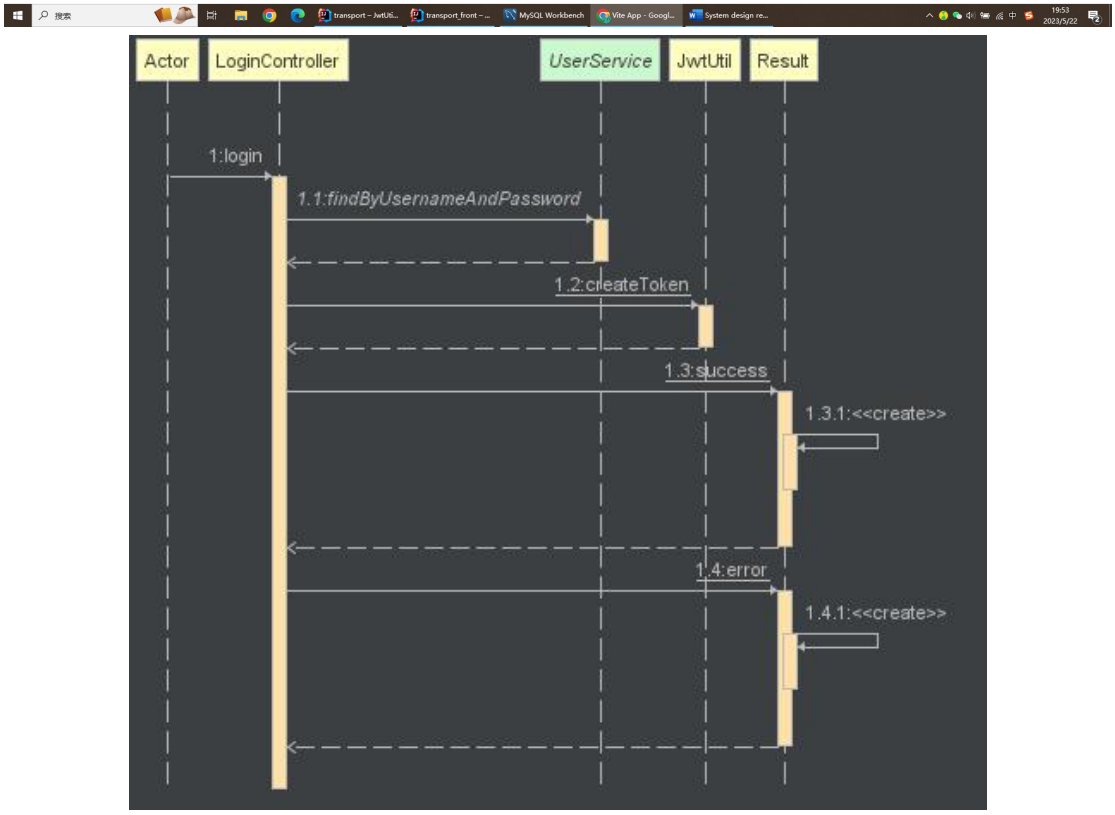


功能模块：

登录

用户登录成功后服务端返回 JWT 格式的 Token





JWT

Token 生成规则:

header{

alg: HS256

typ: JWT

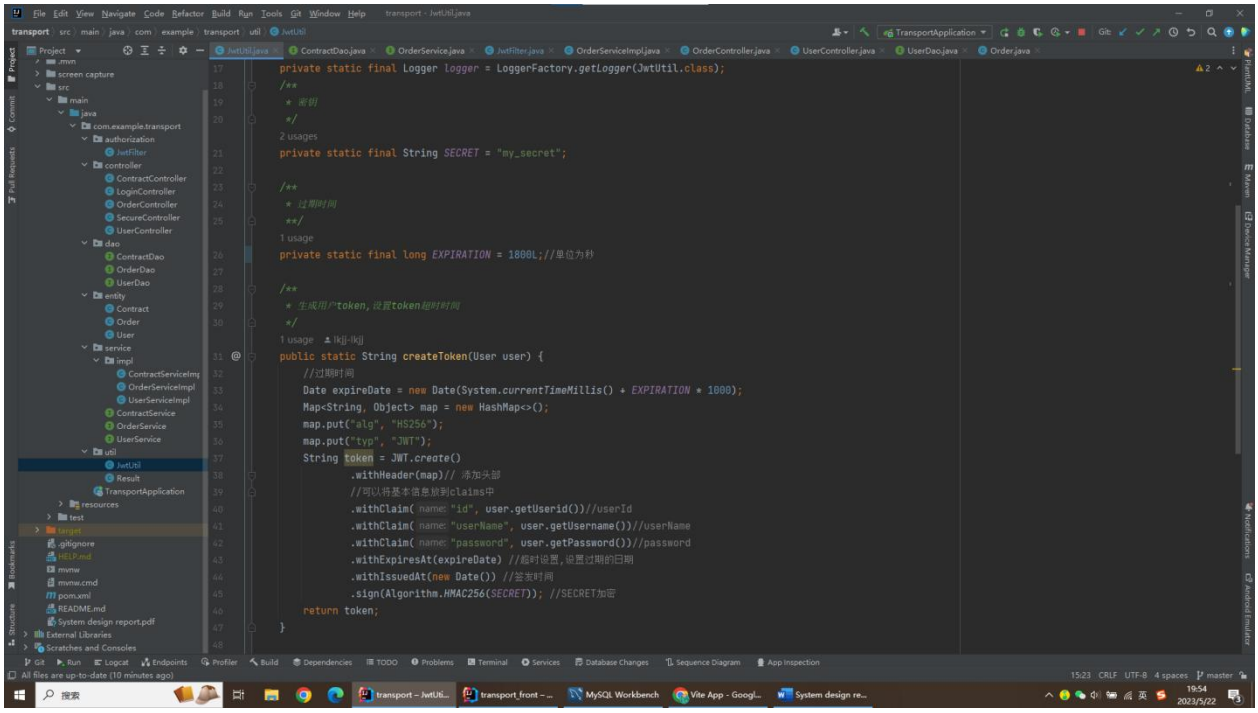
}

body{

id

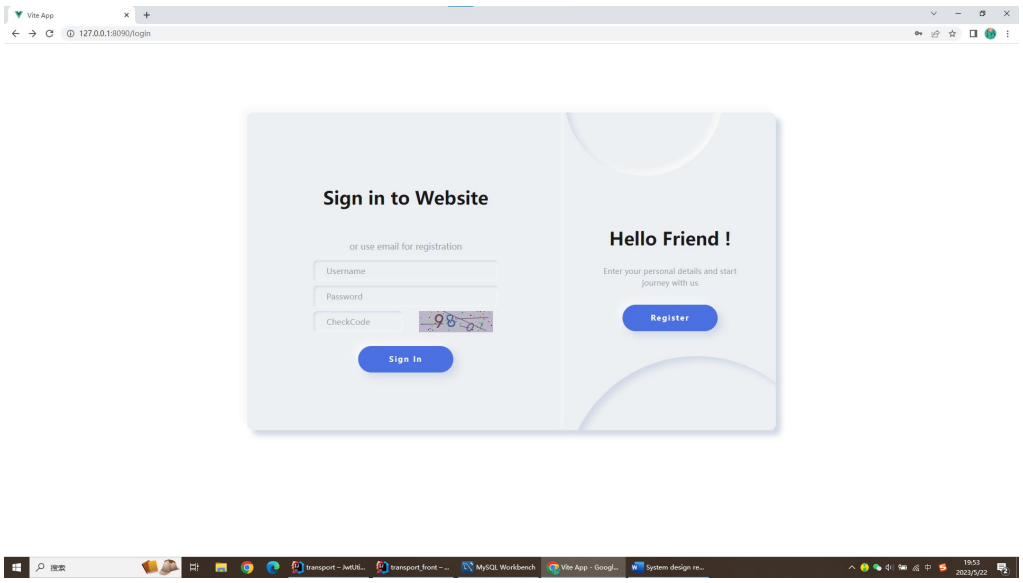
username
password
logintime
}
秘钥为“my_secret”

设置过期时间为 1800 秒。
若 token 不合法或者 token 为空则返回登录页面，无法请求信息。

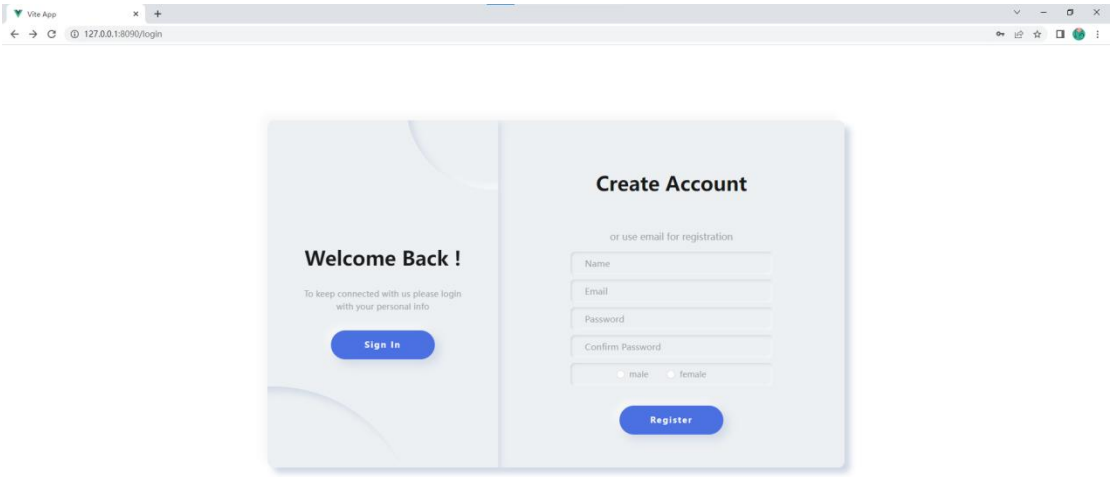


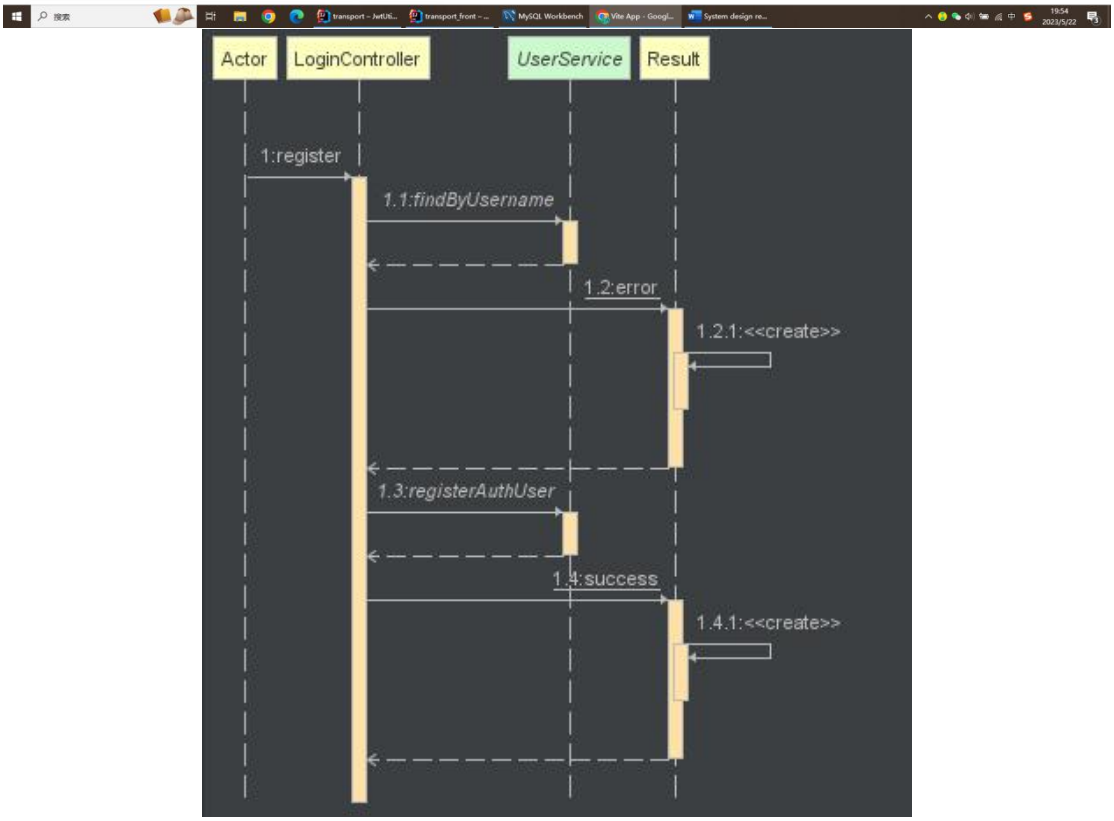
图片验证码

点击可切换，
进行真人验证。

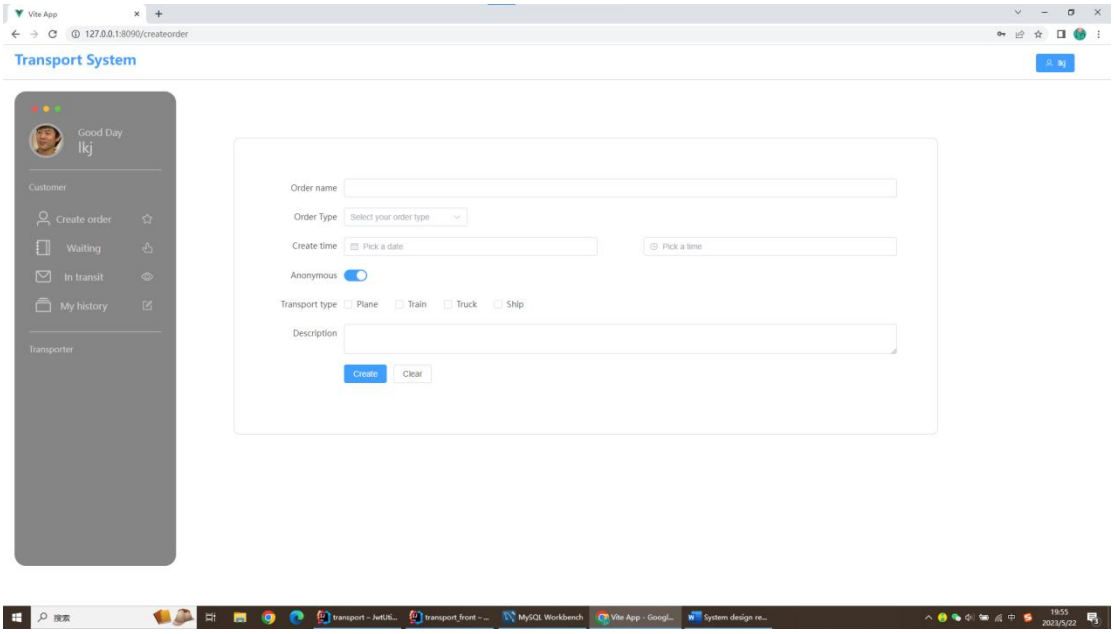


注册

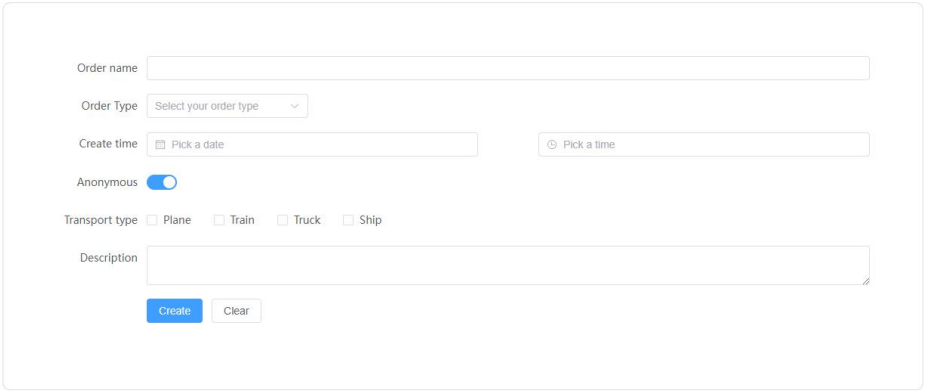

Vite App
127.0.0.1:8090/login
Welcome Back !
To keep connected with us please login with your personal info
Sign In
Create Account
or use email for registration
Name
Email
Password
Confirm Password
male
female
Register

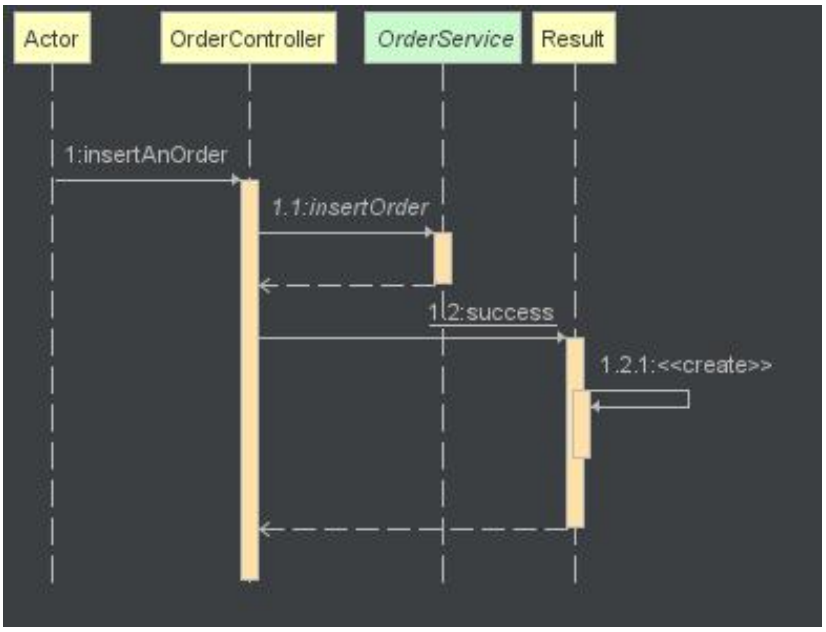

Actor
LoginController
UserService
Result
1:register
1.1:findByUsername
1.2:error
1.2.1:<<create>>
1.3:registerAuthUser
1.4:success
1.4.1:<<create>>

客户界面


Transport System
Good Day
lkj
Customer
Create order
Waiting
In transit
My history
Transporter
Order name
Order Type
Select your order type
Create time
Pick a date
Pick a time
Anonymous
Transport type
Plane
Train
Truck
Ship
Description
Create
Clear

创建订单

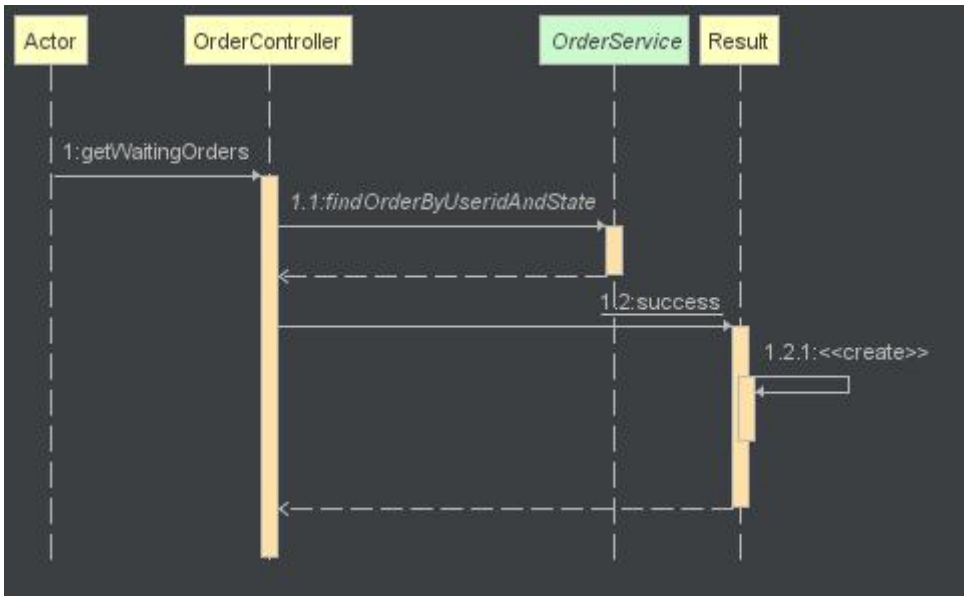

Order name
Order Type
Select your order type
Create time
Pick a date
Pick a time
Anonymous
Transport type
Plane
Train
Truck
Ship
Description
Create
Clear

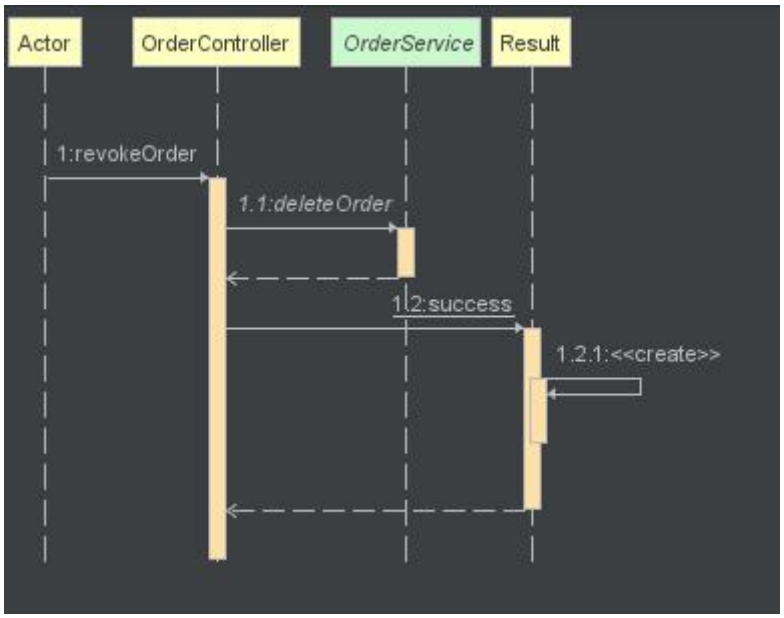


查看等待被承接的订单

点击 revoke 可撤回未被承接的订单

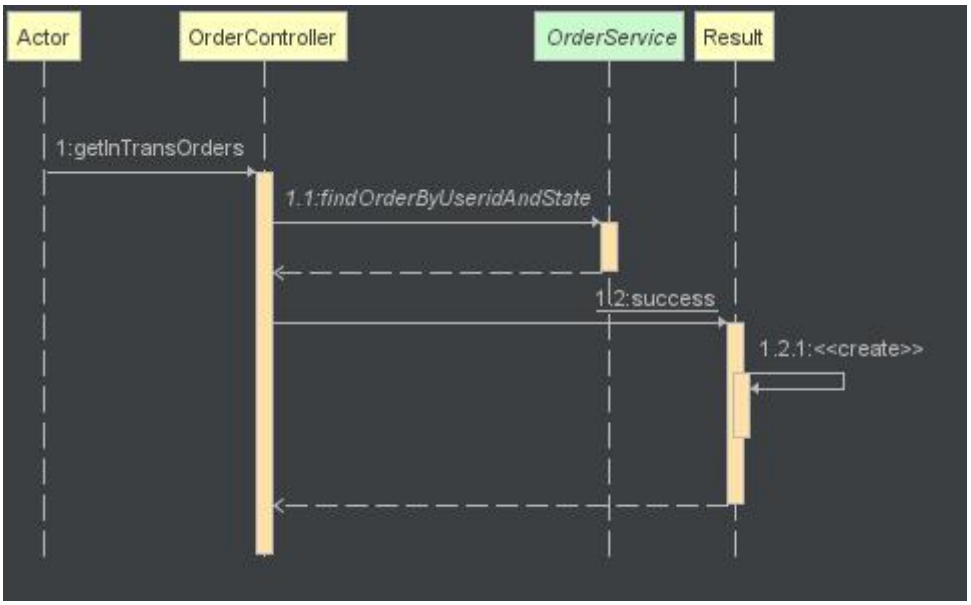
Name	Description	Create Time	User Id	State	Operations
222	dasdasd	2023-05-Mo 19:55:52	1	Waiting be transported	Revoke
444	Ship project	2023-05-Mo 19:56:10	1	Waiting be transported	Revoke





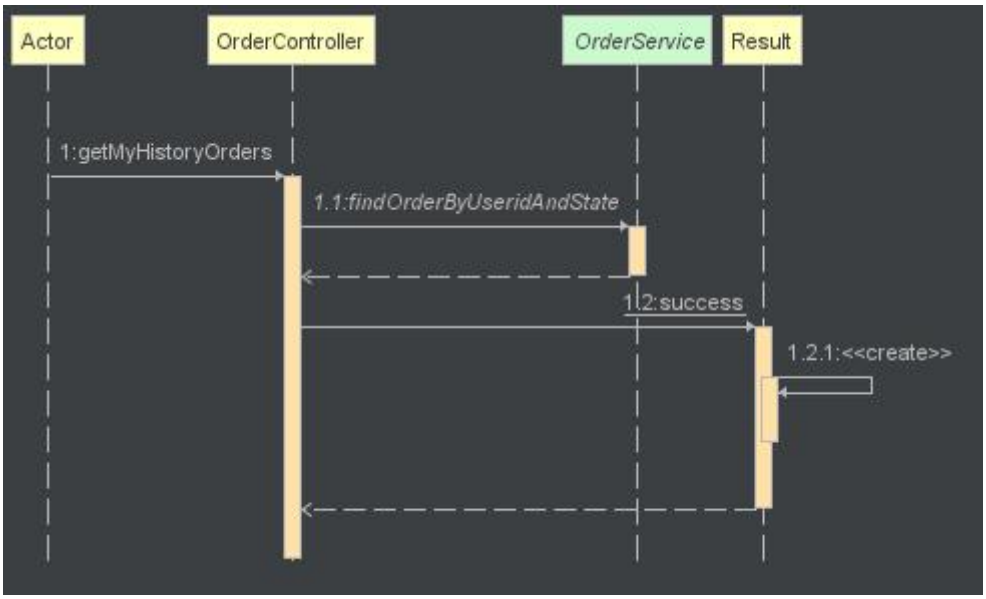
查看运输中的订单

Name	Description	Create time	Userid	TransporterId	Take order time	State
second	nothing	2023-05-Mo 12:42:42	1	2	2023-05-22 14:51:12	In transit

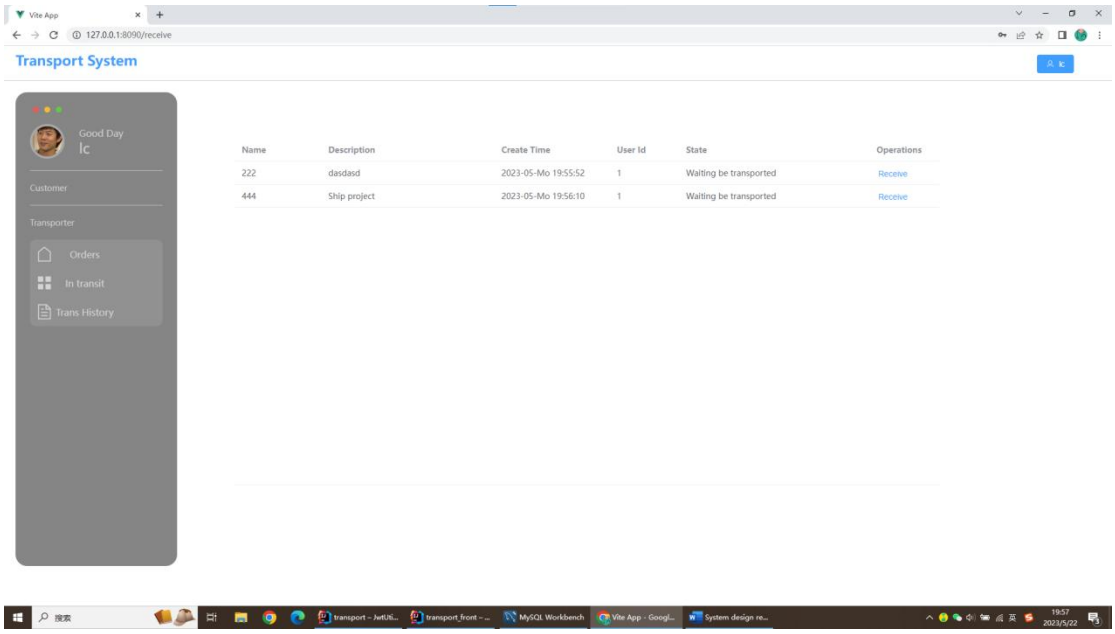


查看历史订单

Name	Description	Create time	Userid	TransporterId	Take order time	Completed time	State
firstorder	nothing	2023-05-Mo 12:42:42	1	2	2023-05-Mo 12:42:42	2023-05-22 15:01:44	Completed
third	nothing	2023-05-Mo 12:42:42	1	2	2023-05-Mo 12:42:42	2023-05-Mo 12:42:42	Completed
four	nothing	2023-05-Mo 12:42:42	1	2	2023-05-Mo 12:42:42	2023-05-Mo 12:42:42	Completed

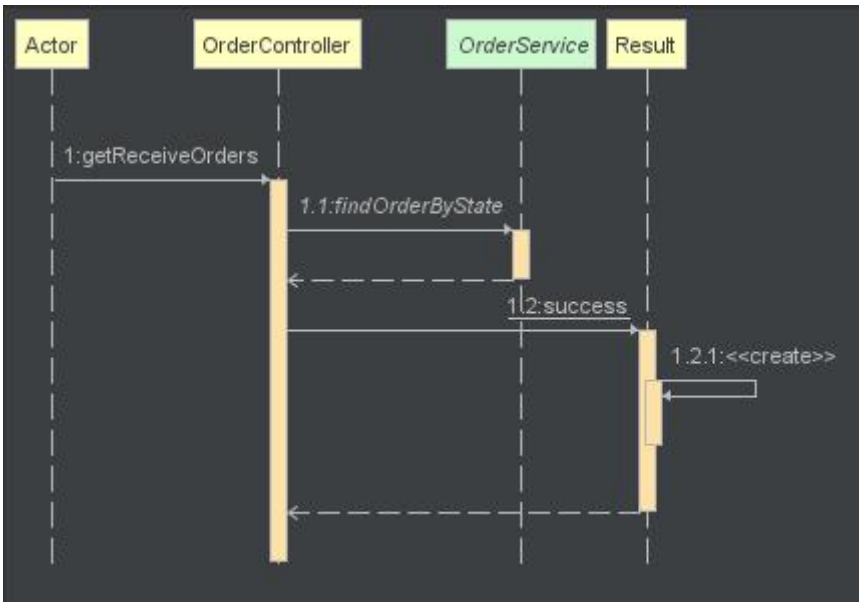


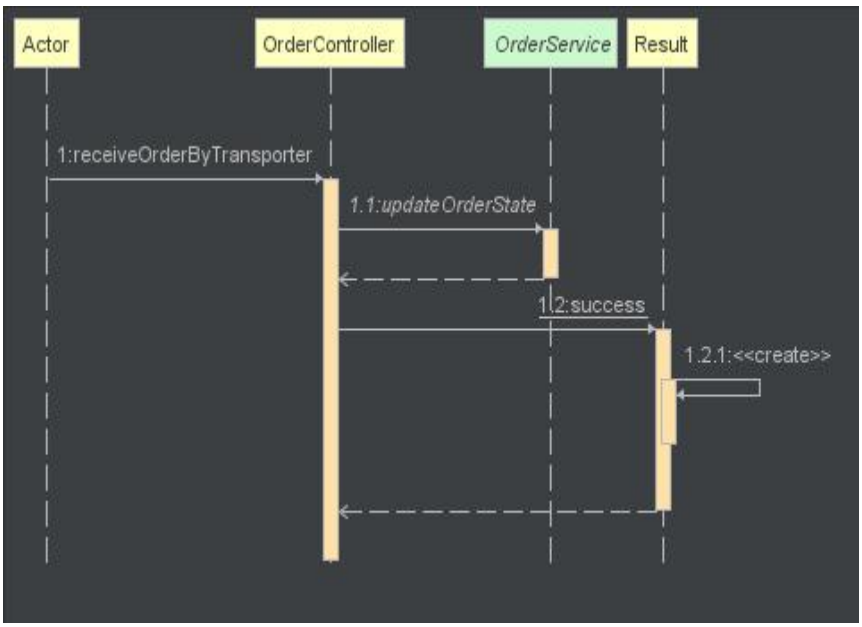
运输者界面



承接订单

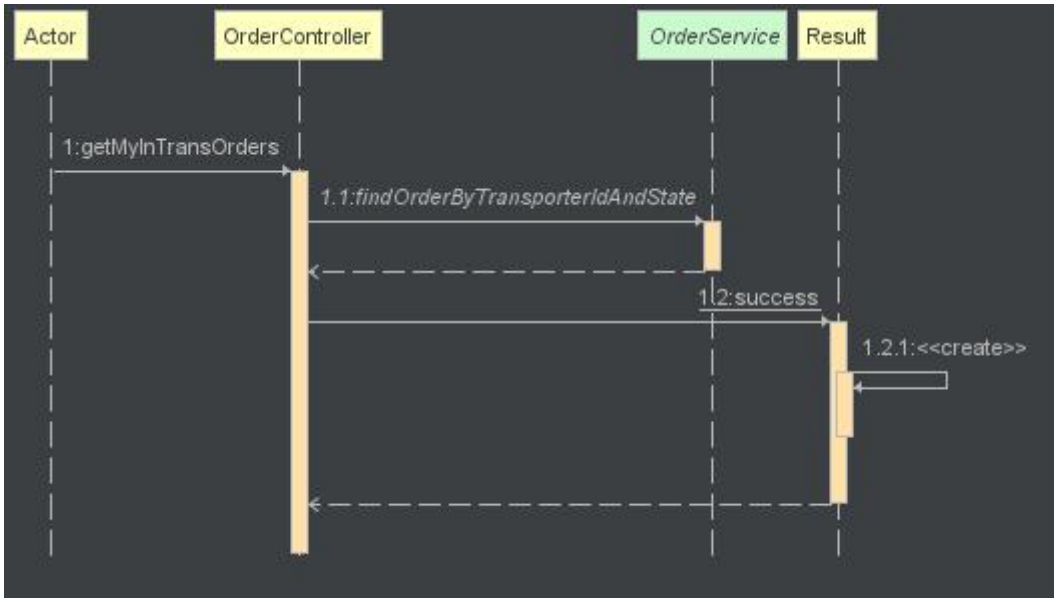
Name	Description	Create Time	User Id	State	Operations
222	dasdasd	2023-05-Mo 19:55:52	1	Waiting be transported	Receive
444	Ship project	2023-05-Mo 19:56:10	1	Waiting be transported	Receive

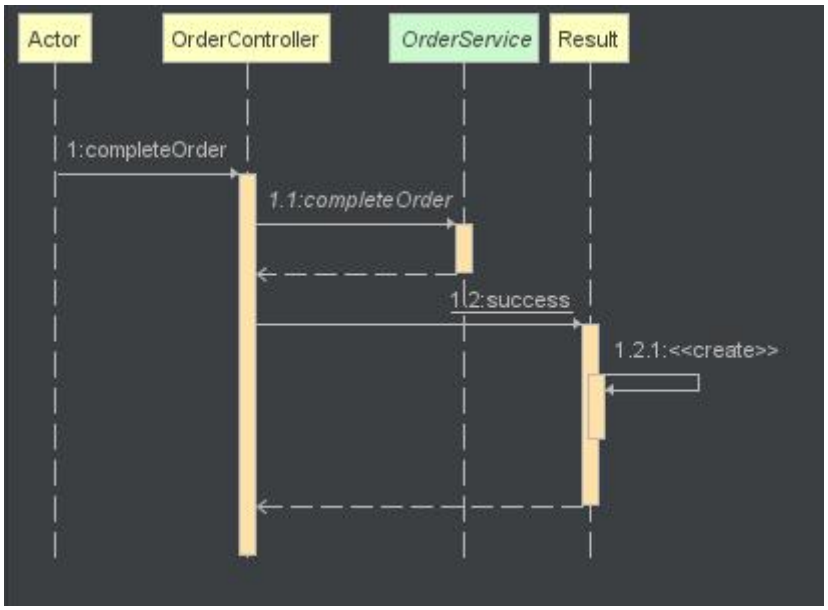




查看运输中的订单

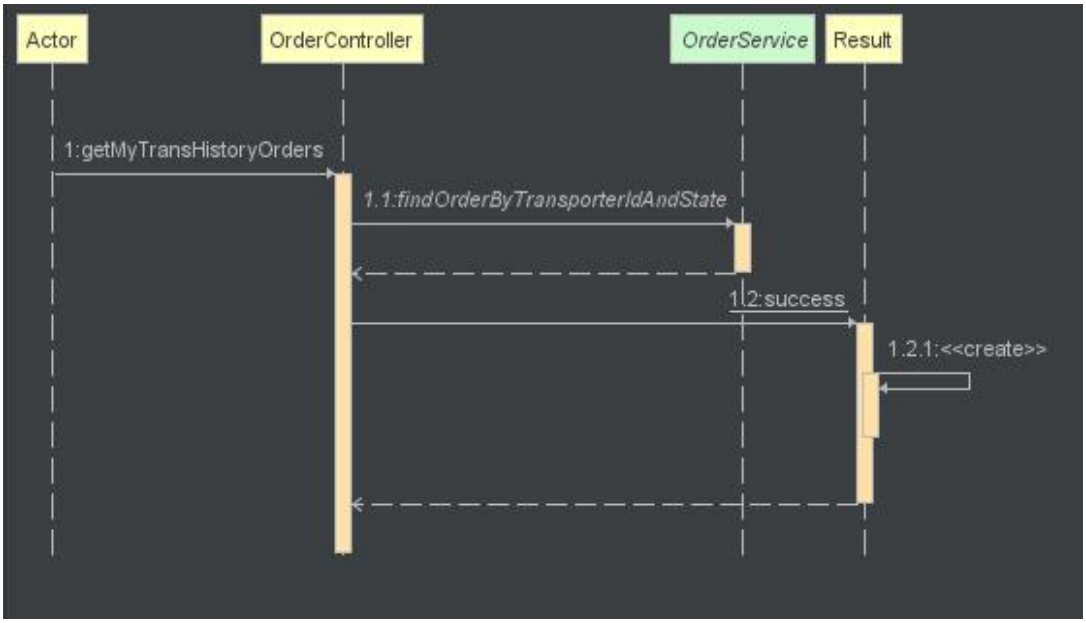
Name	Description	Create time	Userid	TransporterId	Take order time	State	Operations
second	nothing	2023-05-Mo 12:42:42	1	2	2023-05-22 14:51:12	In transit	Complete





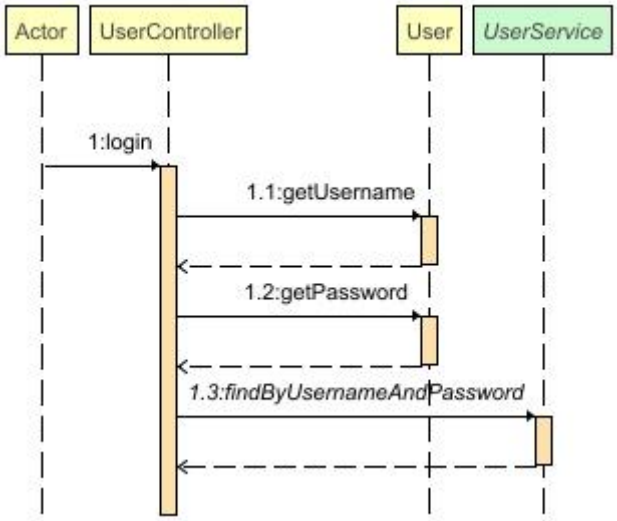
承运的历史订单

Name	Description	Create time	Userid	TransporterId	Take order time	Completed time	State
firstorder	nothing	2023-05-Mo 12:42:42	1	2	2023-05-Mo 12:42:42	2023-05-22 15:01:44	Completed
third	nothing	2023-05-Mo 12:42:42	1	2	2023-05-Mo 12:42:42	2023-05-Mo 12:42:42	Completed
four	nothing	2023-05-Mo 12:42:42	1	2	2023-05-Mo 12:42:42	2023-05-Mo 12:42:42	Completed



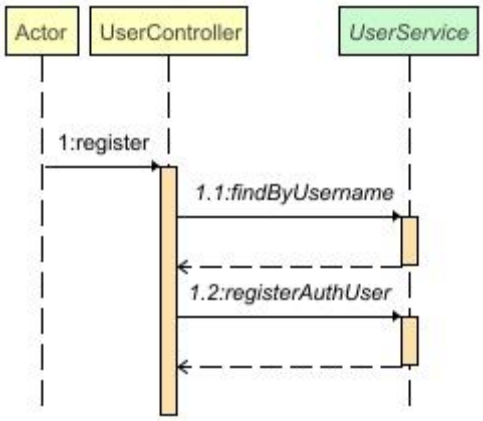
登录（thymeleaf）

用户输入用户名和密码，进入 UserController 层验，UserController 层再调用底层接口查询数据并验证身份。用户登录功能的时序图如图所示：



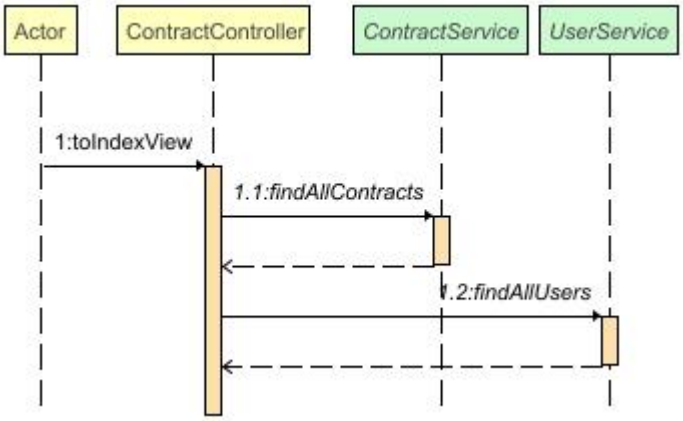
注册（thymeleaf）

用户输入用户名和密码，以及确认密码后，向 UserController 发送请求，UserController 确认用户名不存在后即向数据库中插入用户信息，并返回登录页面，注册成功。若两次密码不一致或用户已存在则返回注册页面，显示相关错误信息。时序图如图所示：



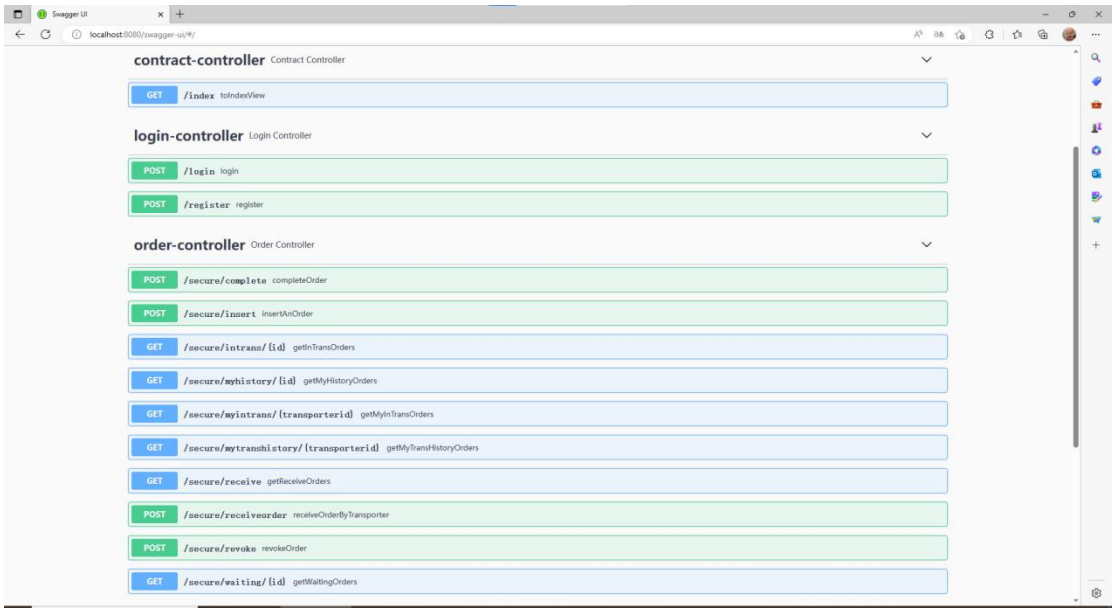
查询所有用户和查询所有合同（thymeleaf）

分别向 UserService 和 ContractService 查询用户和合同的所有记录，并返回到主页面展示给用户，时序图如图所示：



Restful API 设计

使用 swagger2 进行前后端联合开发，api 设计符合 restful api 风格。

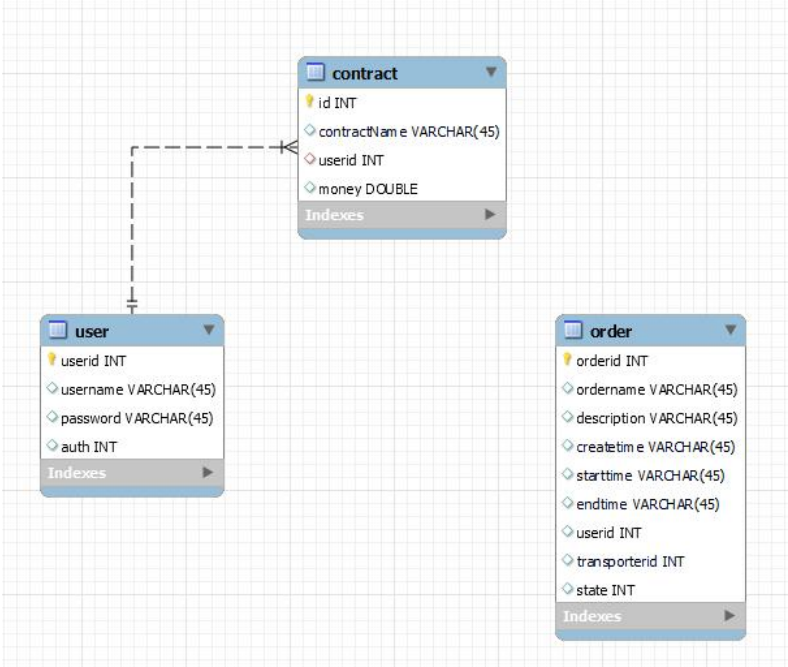


数据库设计：

简介：

数据库目前有 user 和 contract 两张表，其中 contract 表的 userid 属性是外键，关联 user 表中的 userid 主键。ER 图如图所示：

ER 图：

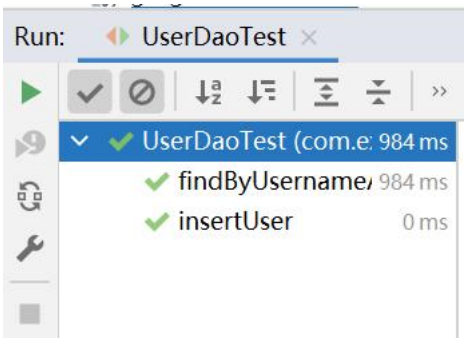


测试方案：

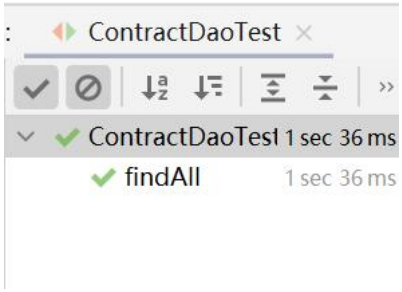
本次测试主要针对 dao 层的各方法进行单元测试和 controller 层的各接口进行集成测试。

单元测试主要采用 JUnit4 测试框架，添加@Test 注解，在测试方法中使用传统的 System.out.println 对方法结果进行输出查看。

对 UserDao 层的 findByUsernameAndPassword，insertUser 方法进行测试，测试均通过。

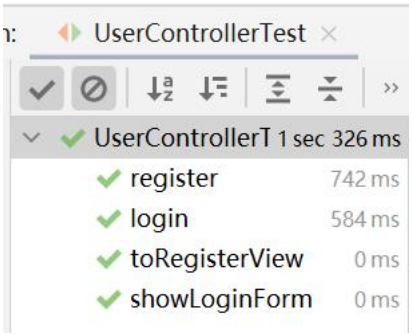


对 ContractDao 层的 findAll 方法进行测试，测试通过。



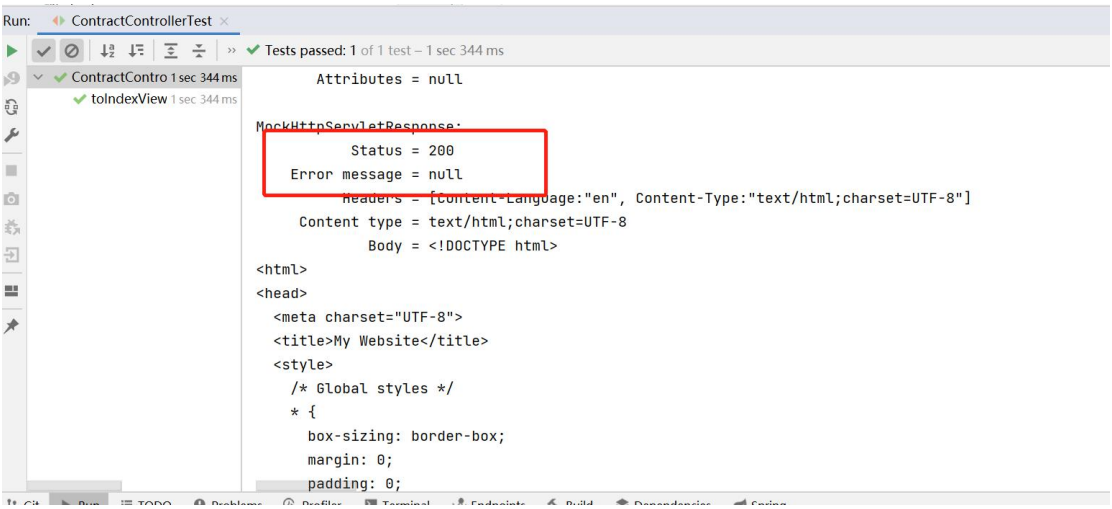
集成测试主要采用 MockMvc 工具，它由 spring-test 包提供，实现了对 Http 请求的模拟，能够直接使用网络的形式，转换到 Controller 的调用，不依赖网络环境。

对 UserController 层的 showLoginForm，toRegisterView，login，register 接口进行测试，测试均通过。

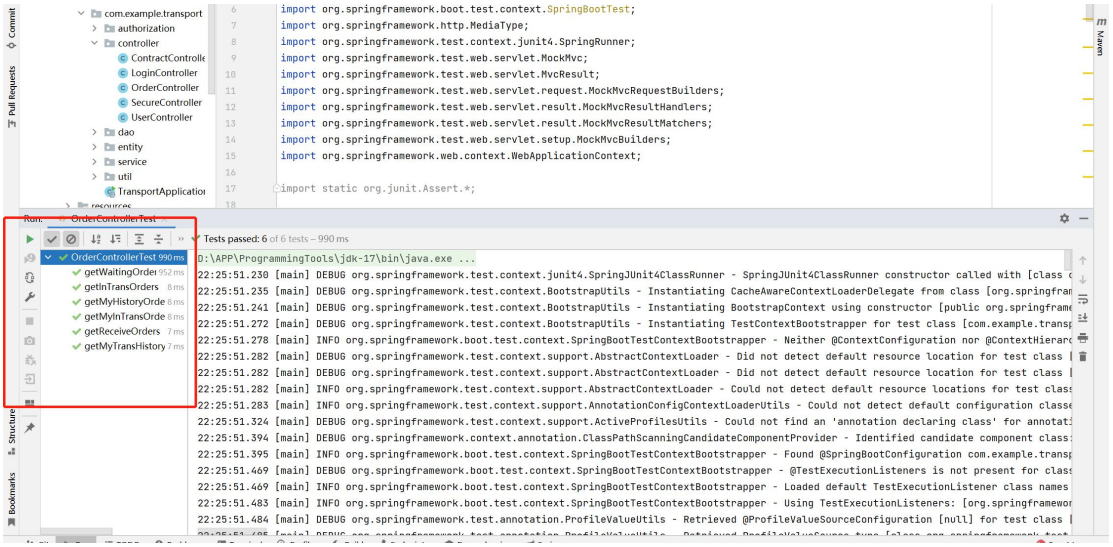


例如 showLoginForm 测试结果如下，返回状态码为 200，请求成功：

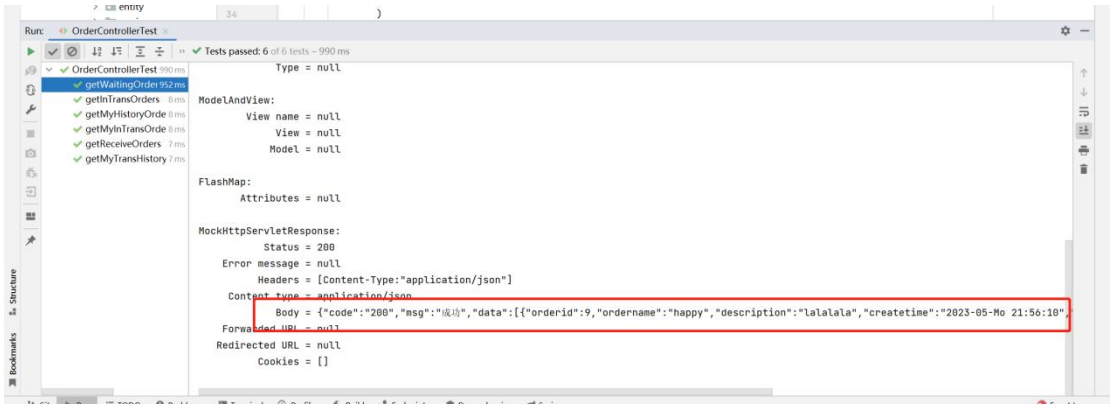
对 ContractController 层的 toIndexView 接口进行测试，结果如下，返回状态码为 200，请求成功：



OrderControllerTest：

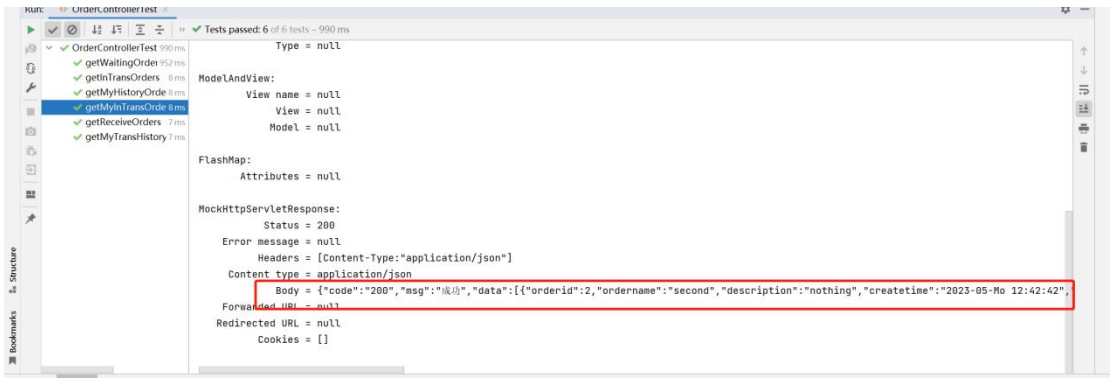


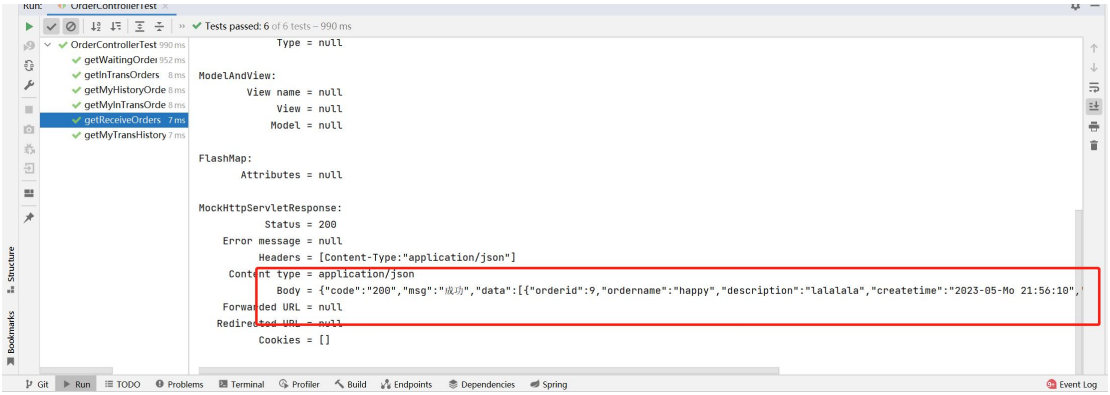
具体结果：

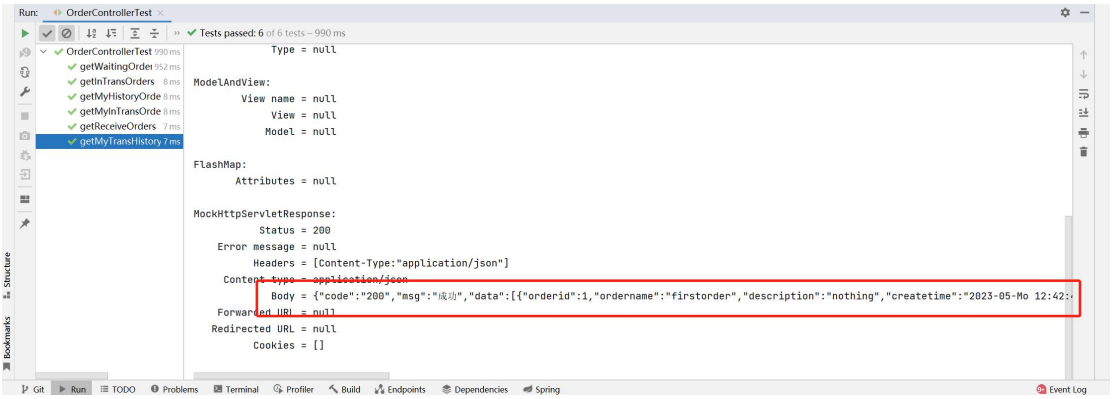

Run: OrderControllerTest
Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[{"orderid":9,"ordername":"happy","description":"lalalala","createtime":"2023-05-Mo 21:56:10",
Forwarded URL = null
Redirected URL = null
Cookies = []


Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[]}
Forwarded URL = null
Redirected URL = null
Cookies = []
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log

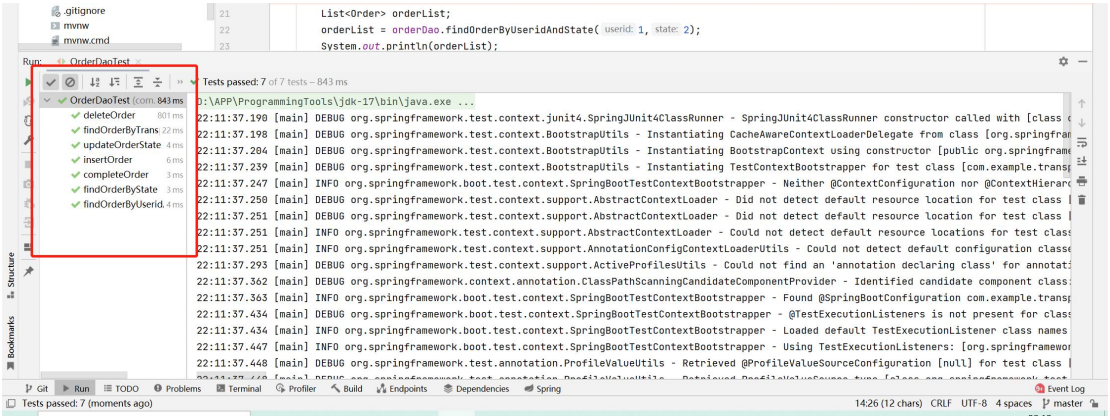

Run: OrderControllerTest
Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[{"orderid":1,"ordername":"firstorder","description":"nothing","createtime":"2023-05-Mo 12:42:
Forwarded URL = null
Redirected URL = null
Cookies = []
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log


Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[{"orderid":2,"ordername":"second","description":"nothing","createtime":"2023-05-Mo 12:42:42",
Forwarded URL = null
Redirected URL = null
Cookies = []

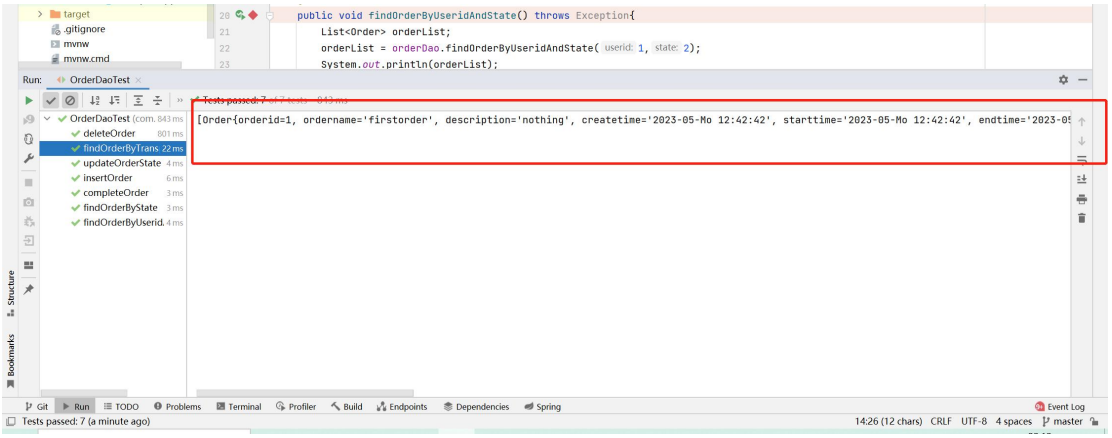


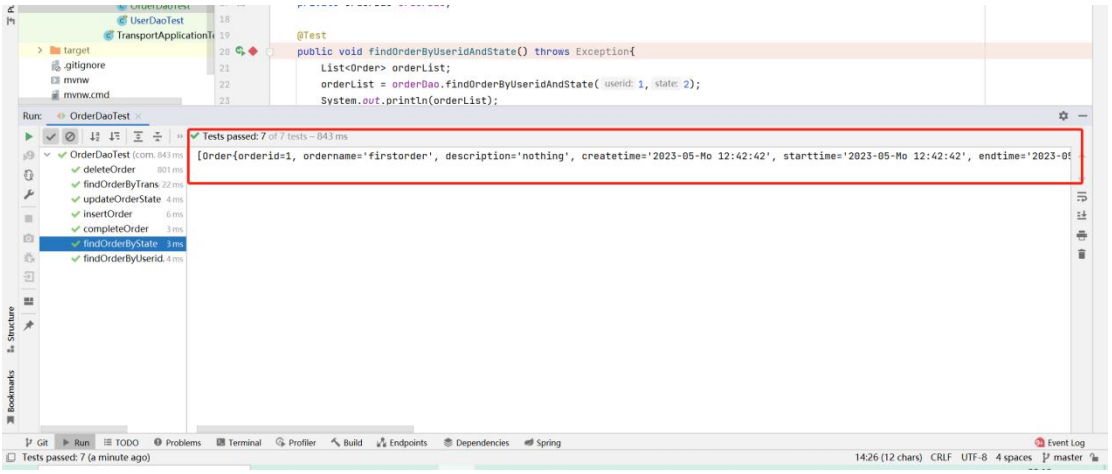


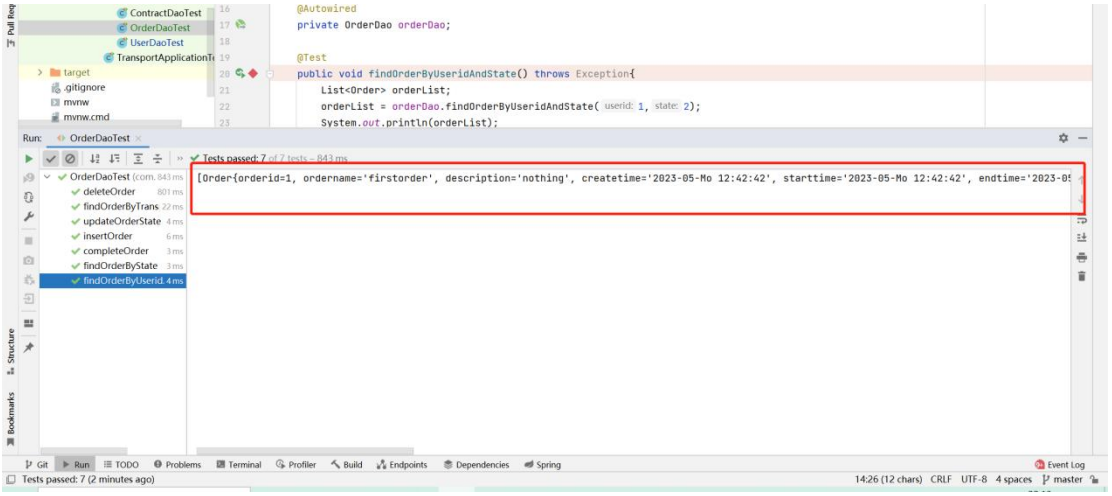
OrderDaoTest:

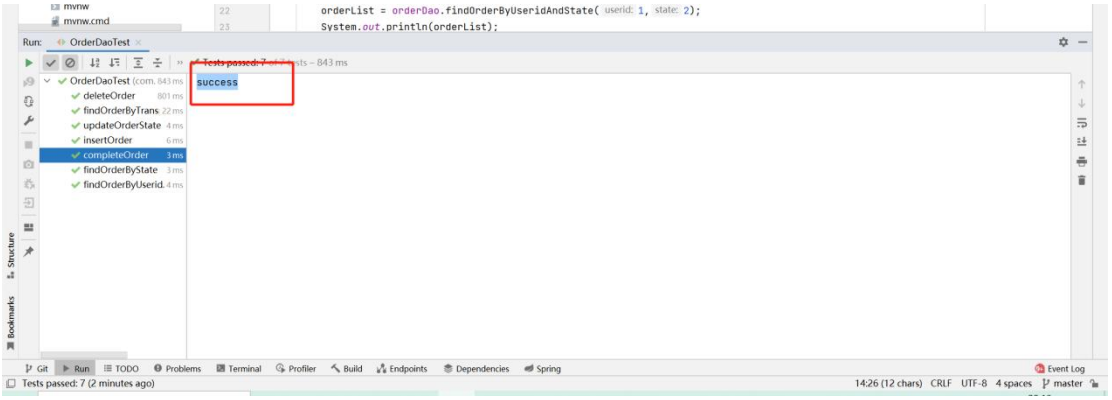


具体结果:

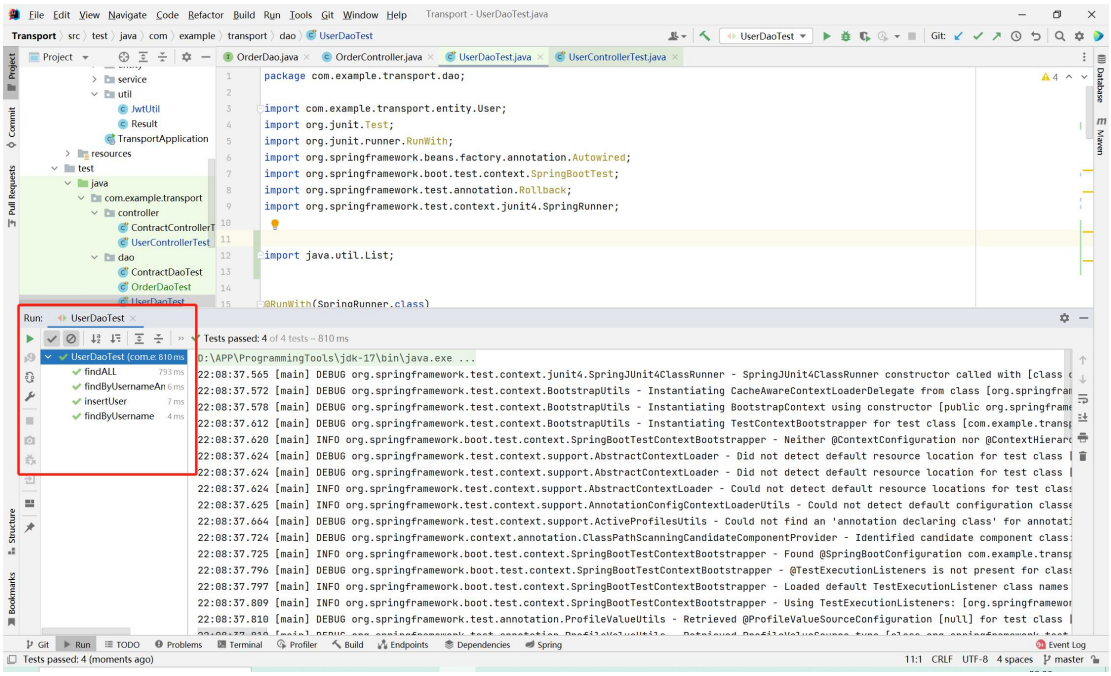



UserDaoTest
TransportApplicationTests
target
.gitignore
mvnw
mvnw.cmd
@Test
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
Tests passed: 7 of 7 tests – 843 ms
[Order{orderid=1, ordername='firstorder', description='nothing', createtime='2023-05-Mo 12:42:42', starttime='2023-05-Mo 12:42:42', endtime='2023-05
OrderDaoTest (com. 843 ms
deleteOrder 801 ms
findOrderByTrans 22 ms
updateOrderState 4 ms
insertOrder 6 ms
completeOrder 3 ms
findOrderByState 3 ms
findOrderByUserid. 4 ms
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log
Tests passed: 7 (a minute ago)
14:26 (12 chars)
CRLF
UTF-8
4 spaces
master

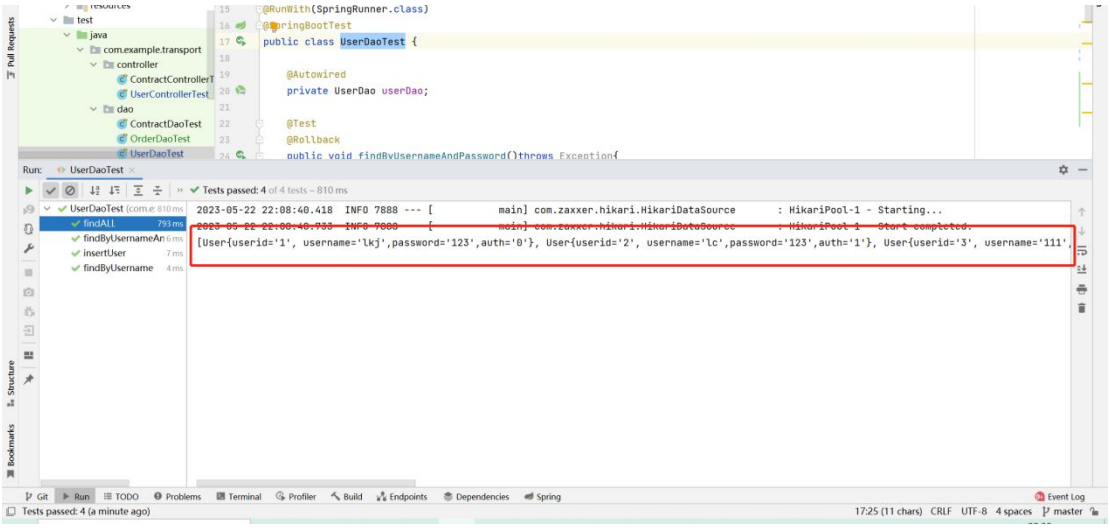

ContractDaoTest
OrderDaoTest
UserDaoTest
TransportApplicationTests
target
.gitignore
mvnw
mvnw.cmd
@Autowired
private OrderDao orderDao;
@Test
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
Tests passed: 7 of 7 tests – 843 ms
[Order{orderid=1, ordername='firstorder', description='nothing', createtime='2023-05-Mo 12:42:42', starttime='2023-05-Mo 12:42:42', endtime='2023-05
OrderDaoTest (com. 843 ms
deleteOrder 801 ms
findOrderByTrans 22 ms
updateOrderState 4 ms
insertOrder 6 ms
completeOrder 3 ms
findOrderByState 3 ms
findOrderByUserid. 4 ms
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log
Tests passed: 7 (2 minutes ago)
14:26 (12 chars)
CRLF
UTF-8
4 spaces
master


mvnw
mvnw.cmd
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
Tests passed: 7 of 7 tests – 843 ms
success
OrderDaoTest (com. 843 ms
deleteOrder 801 ms
findOrderByTrans 22 ms
updateOrderState 4 ms
insertOrder 6 ms
completeOrder 3 ms
findOrderByState 3 ms
findOrderByUserid. 4 ms
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log
Tests passed: 7 (2 minutes ago)
14:26 (12 chars)
CRLF
UTF-8
4 spaces
master

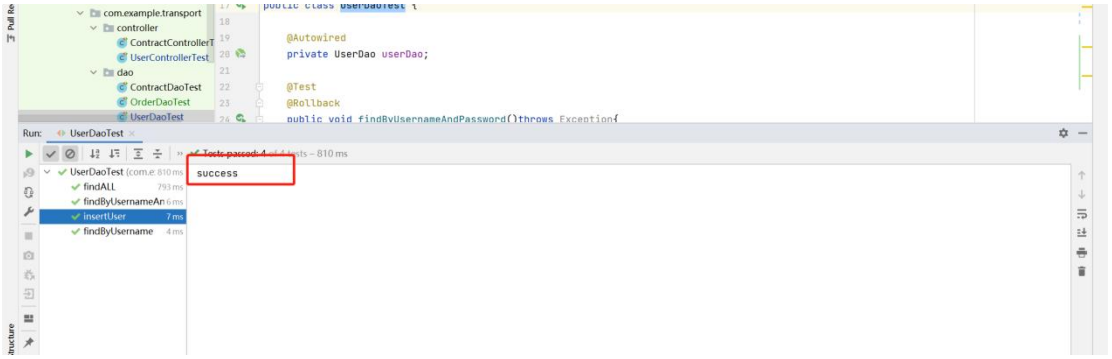
UserDaoTest



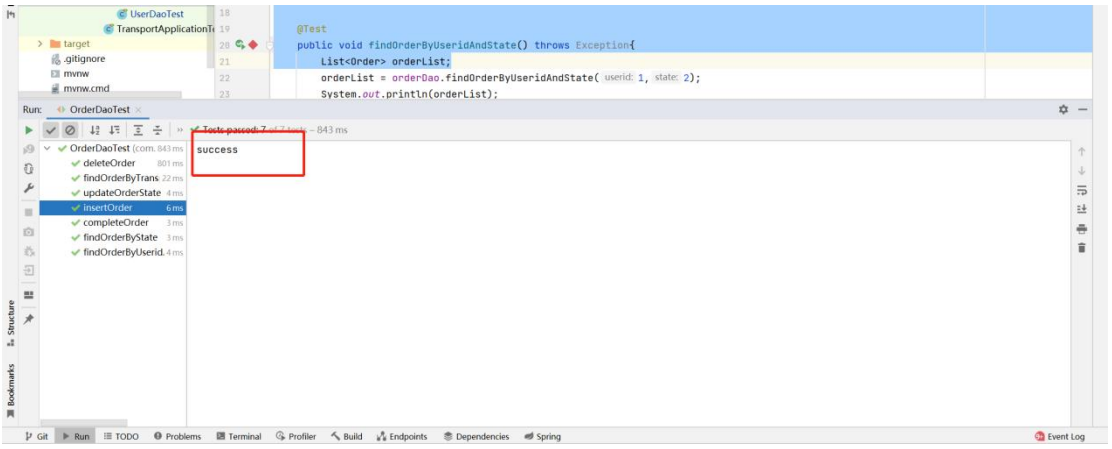
具体结果:

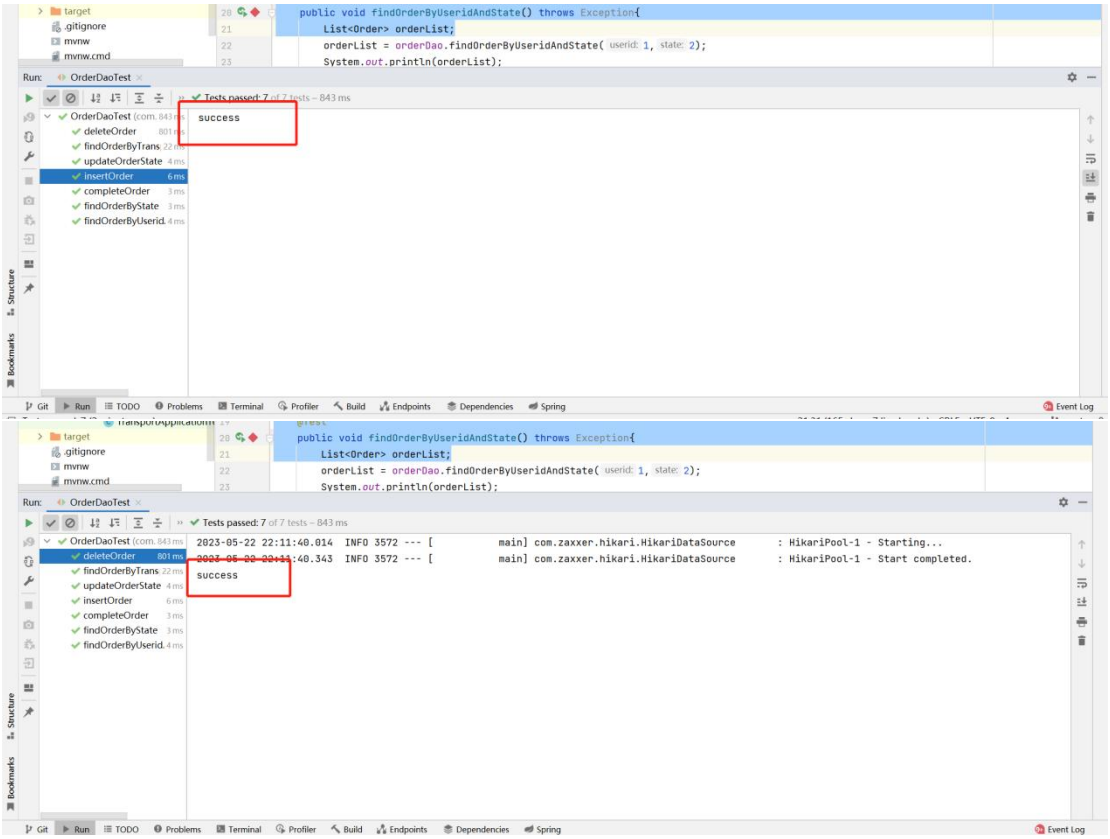



@Rollback
public void findByUsernameAndPassword()throws Exception{
Tests passed: 4 of 4 tests – 810 ms
User{userid='1', username='lkj',password='123',auth='0'}
findALL 793 ms
findByUsernameAn 6 ms
insertUser 7 ms
findByUsername 4 ms

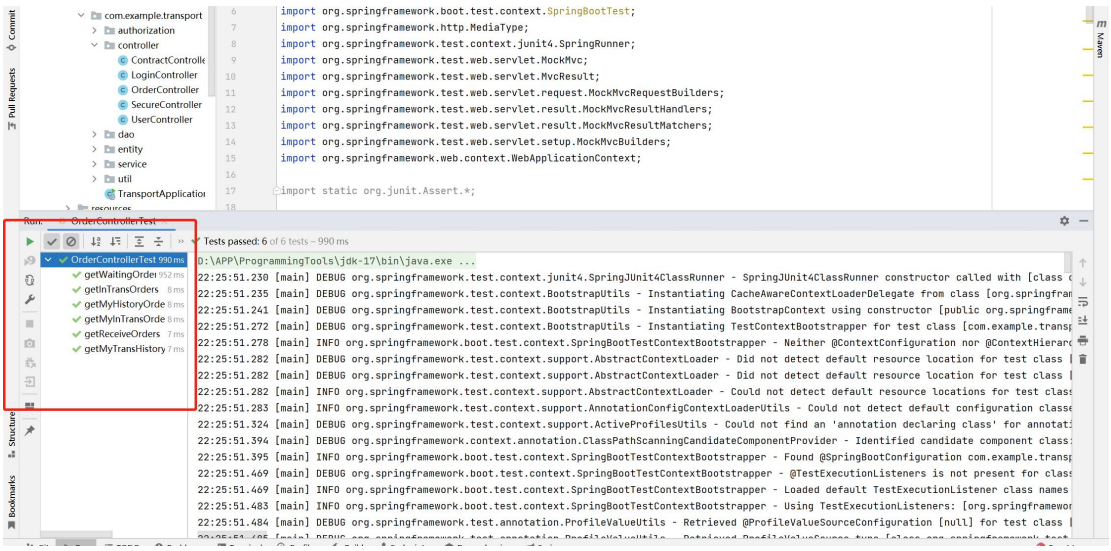

com.example.transport
controller
ContractControllerT
UserControllerTest
dao
ContractDaoTest
OrderDaoTest
UserDaoTest
@Autowired
private UserDao userDao;
@Test
@Rollback
public void findByUsernameAndPassword()throws Exception{
Tests passed: 4 of 4 tests – 810 ms
success
findALL 793 ms
findByUsernameAn 6 ms
insertUser 7 ms
findByUsername 4 ms


com.example.transport
controller
ContractControllerT
UserControllerTest
dao
ContractDaoTest
OrderDaoTest
UserDaoTest
@Autowired
private UserDao userDao;
@Test
@Rollback
public void findByUsernameAndPassword()throws Exception{
Tests passed: 4 of 4 tests – 810 ms
User{userid='1', username='lkj',password='123',auth='0'}
findALL 793 ms
findByUsernameAn 6 ms
insertUser 7 ms
findByUsername 4 ms

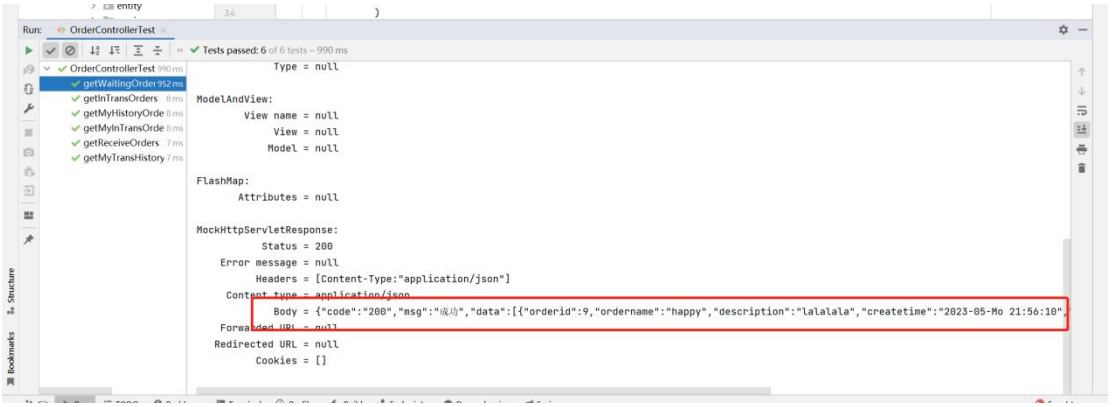

UserDaoTest
TransportApplicationT
target
.gitignore
mvnw
mvnw.cmd
@Test
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Tests passed: 7 of 7 tests – 843 ms
success
OrderDaoTest (com. 843 ms
deleteOrder 801 ms
findOrderByTrans 22 ms
updateOrderState 4 ms
insertOrder 6 ms
completeOrder 3 ms
findOrderByState 3 ms
findOrderByUserid 4 ms



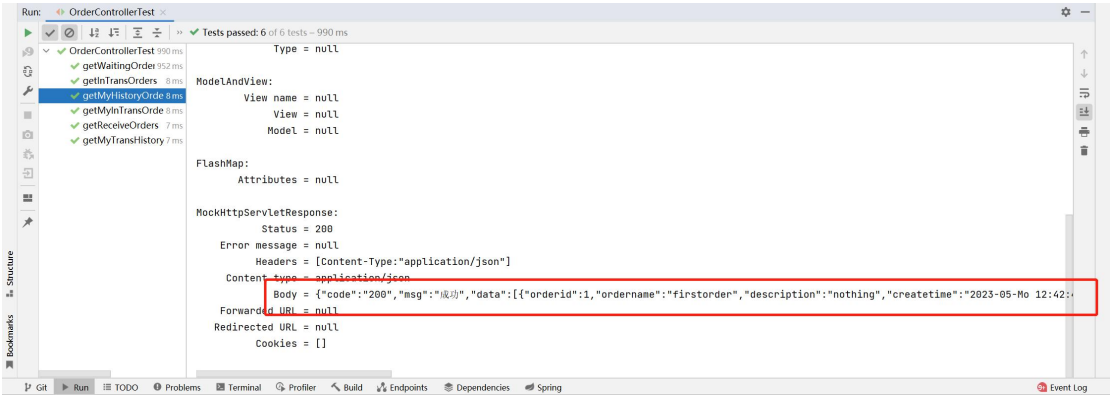
OrderControllerTest：



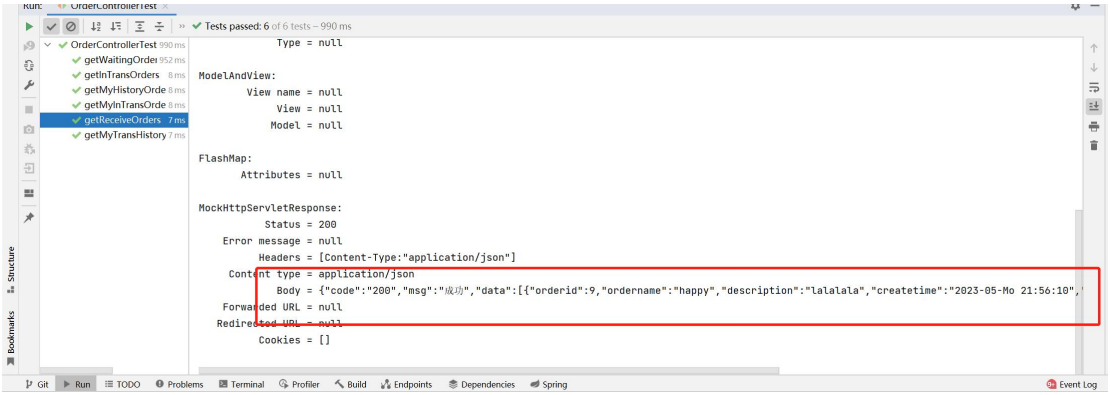
具体结果：

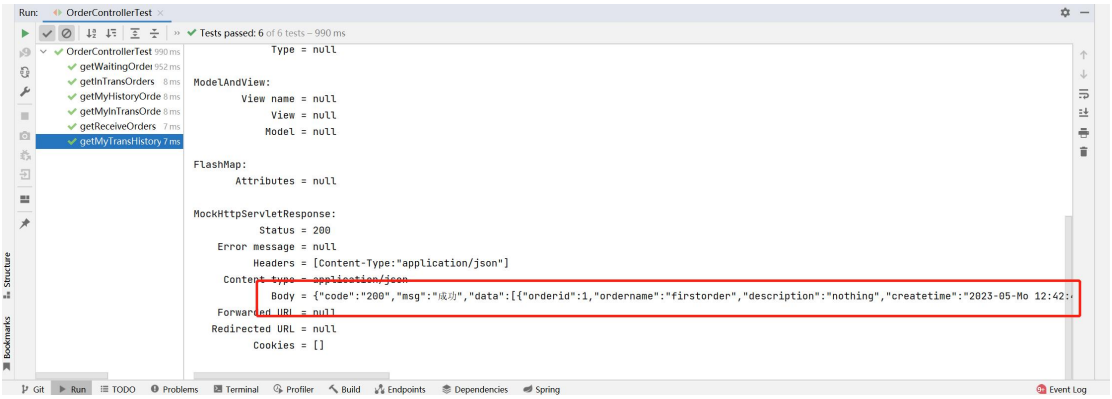

Run: OrderControllerTest
Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[{"orderid":9,"ordername":"happy","description":"lalalala","createtime":"2023-05-Mo 21:56:10",
Forwarded URL = null
Redirected URL = null
Cookies = []


Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[]}
Forwarded URL = null
Redirected URL = null
Cookies = []
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log

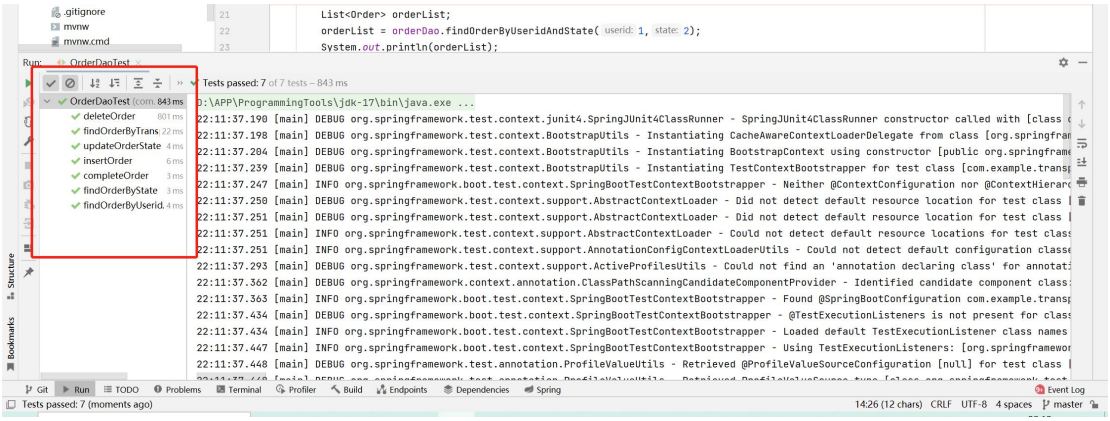

Run: OrderControllerTest
Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[{"orderid":1,"ordername":"firstorder","description":"nothing","createtime":"2023-05-Mo 12:42:
Forwarded URL = null
Redirected URL = null
Cookies = []
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log


Tests passed: 6 of 6 tests – 990 ms
OrderControllerTest 990 ms
getWaitingOrder 952 ms
getInTransOrders 8 ms
getMyHistoryOrde 8 ms
getMyInTransOrde 8 ms
getReceiveOrders 7 ms
getMyTransHistory 7 ms
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"code":"200","msg":"成功","data":[{"orderid":2,"ordername":"second","description":"nothing","createtime":"2023-05-Mo 12:42:42",
Forwarded URL = null
Redirected URL = null
Cookies = []

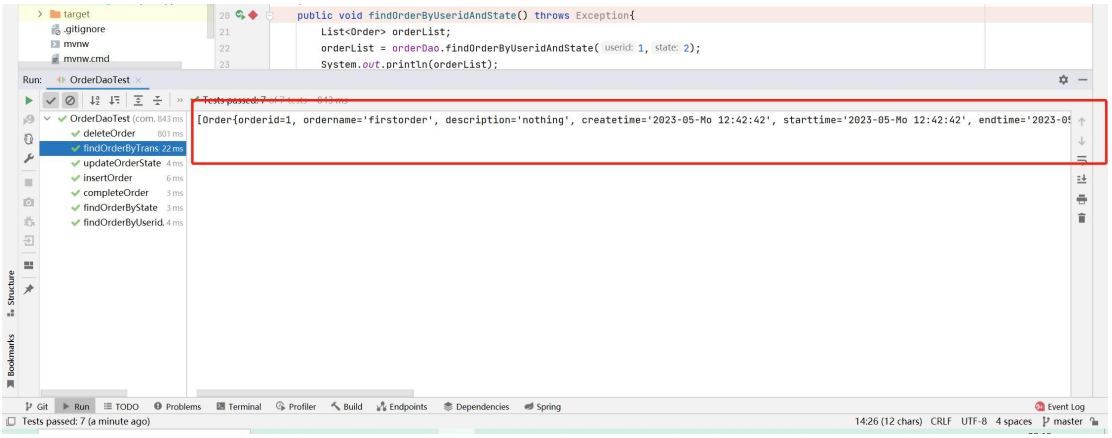


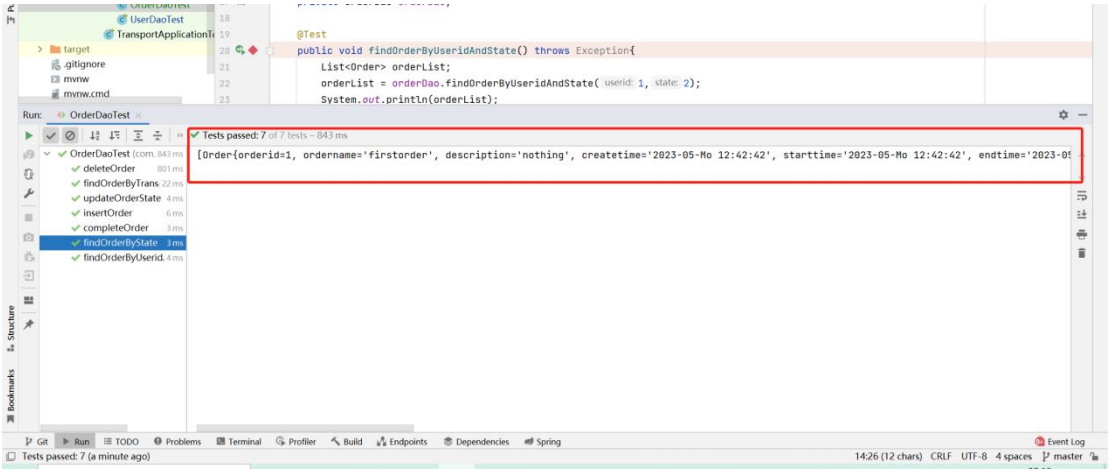


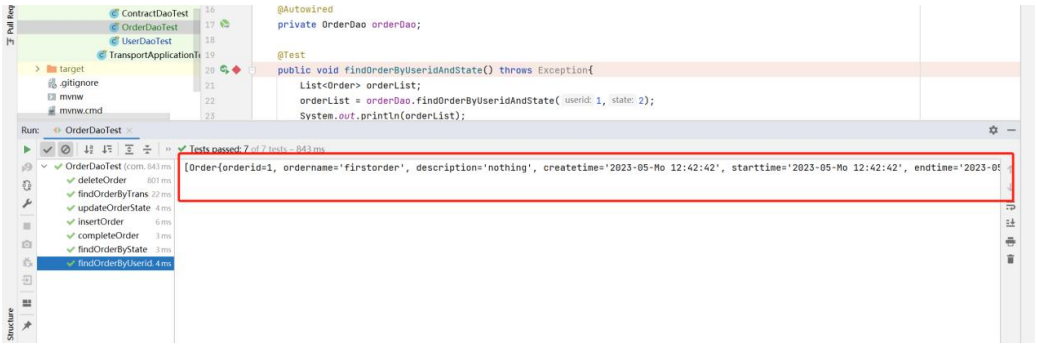
OrderDaoTest：

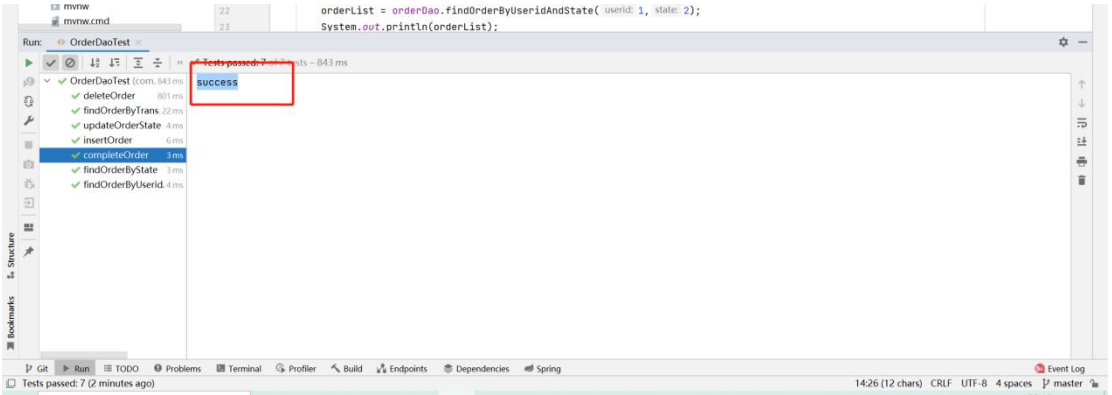


具体结果：


public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
[Order{orderid=1, ordername='firstorder', description='nothing', createtime='2023-05-Mo 12:42:42', starttime='2023-05-Mo 12:42:42', endtime='2023-05
Tests passed: 7 (a minute ago)

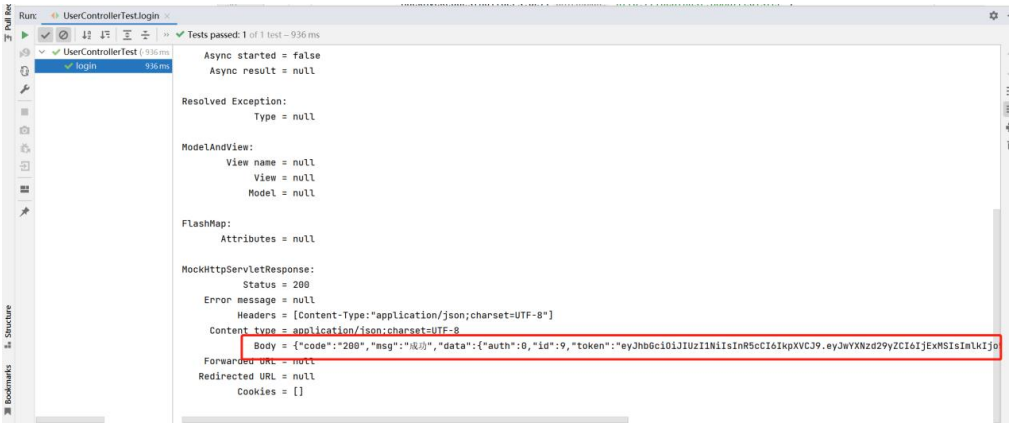

@Test
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
Tests passed: 7 of 7 tests – 843 ms
[Order{orderid=1, ordername='firstorder', description='nothing', createtime='2023-05-Mo 12:42:42', starttime='2023-05-Mo 12:42:42', endtime='2023-05
Tests passed: 7 (a minute ago)


@Autowired
private OrderDao orderDao;
@Test
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
[Order{orderid=1, ordername='firstorder', description='nothing', createtime='2023-05-Mo 12:42:42', starttime='2023-05-Mo 12:42:42', endtime='2023-05

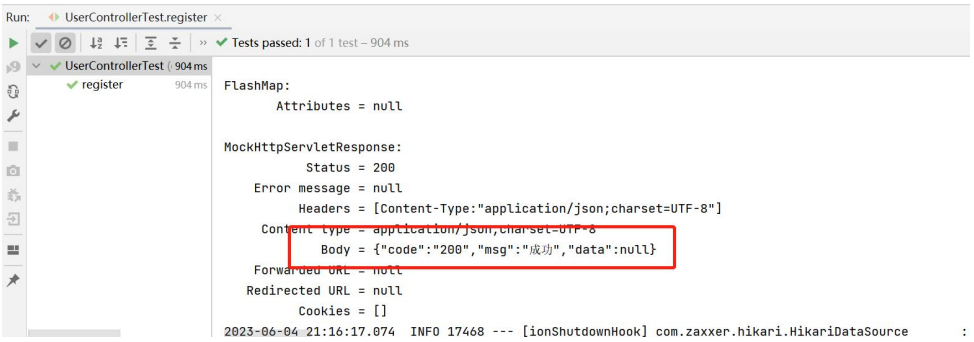

orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
success
Tests passed: 7 (2 minutes ago)

UserControllerTest:

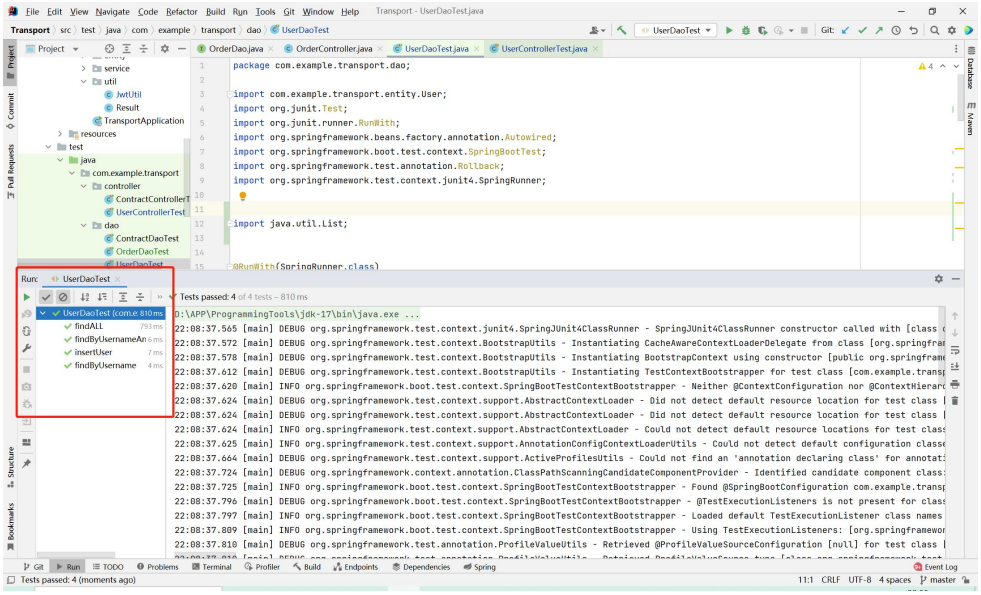
login 接口：



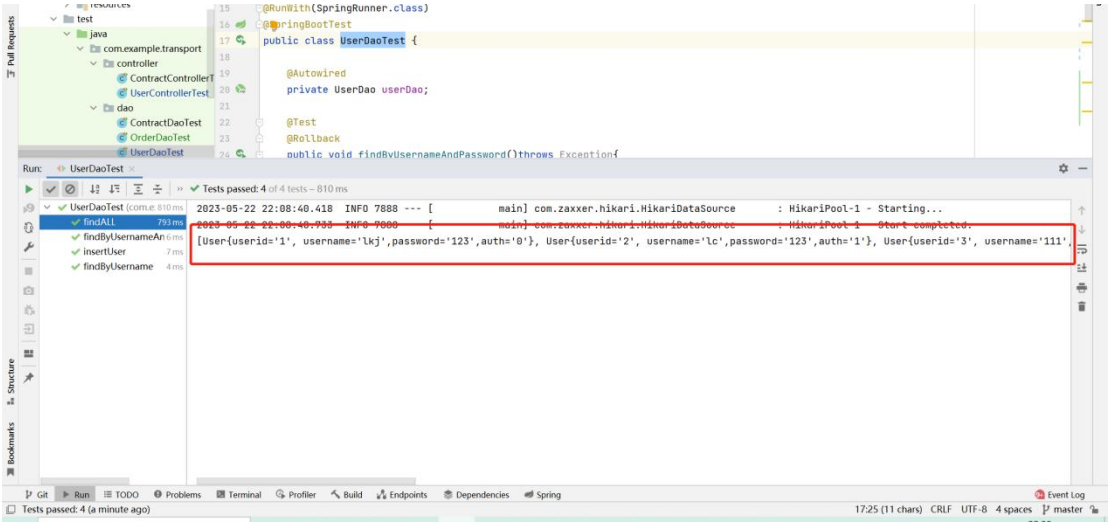
register 接口：

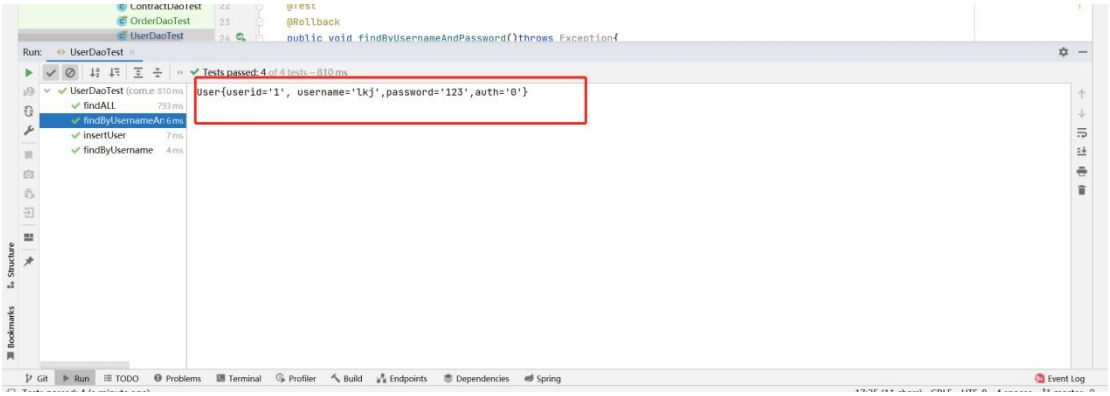


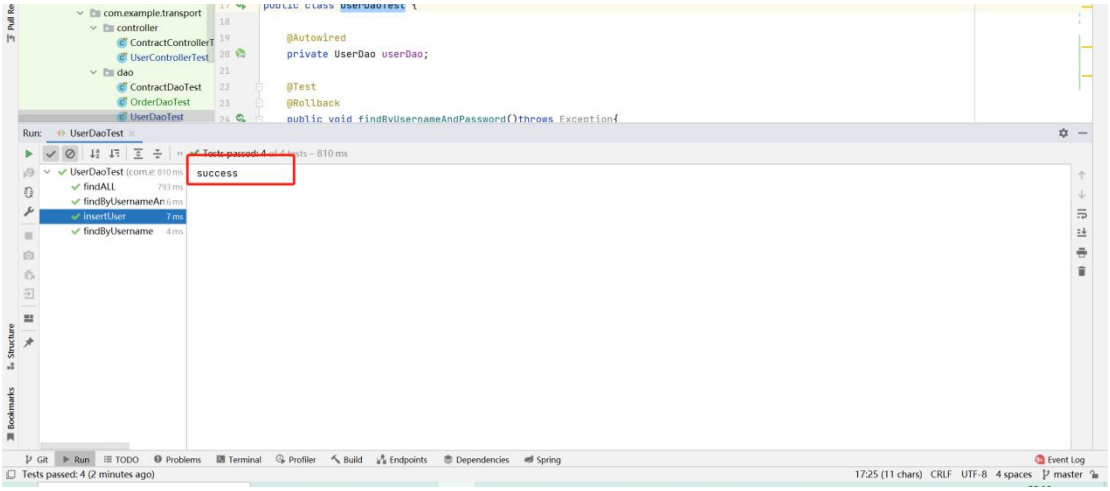
UserDaoTest：

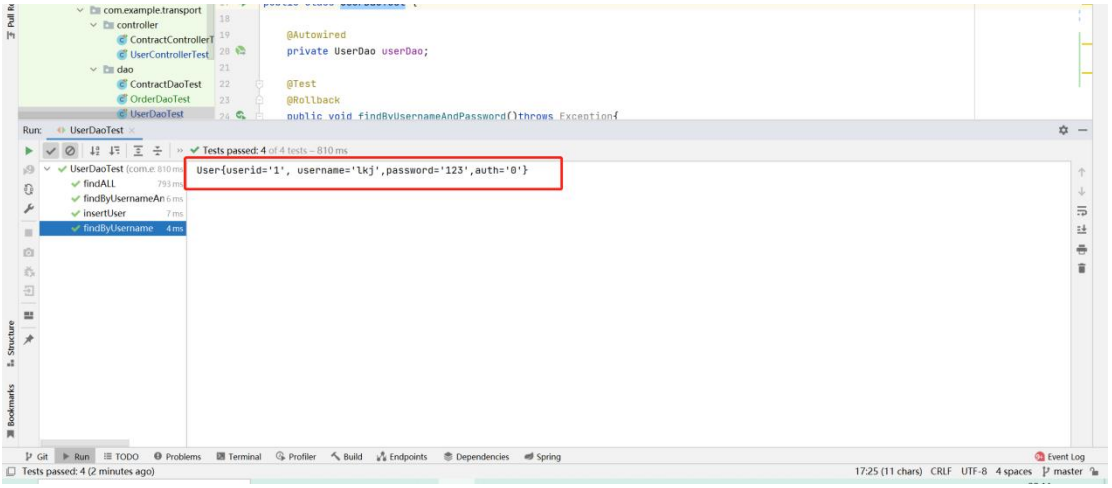


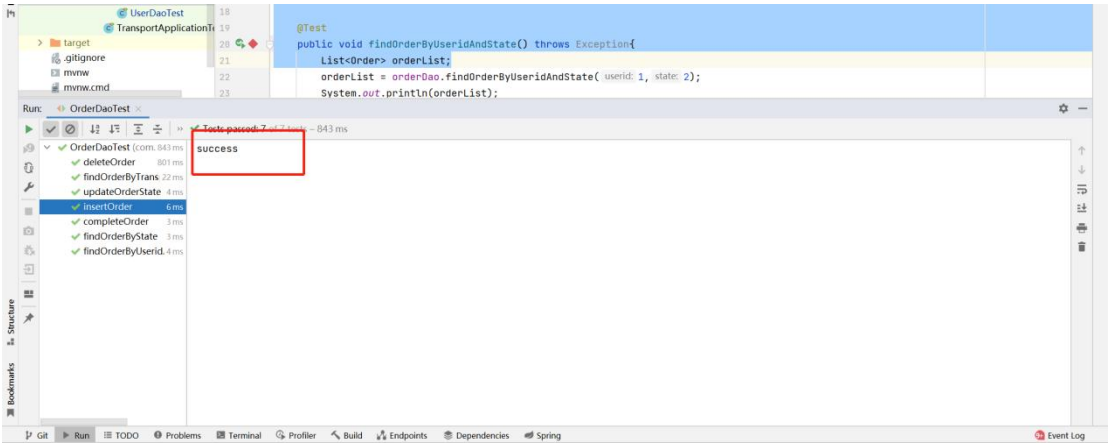
具体结果：

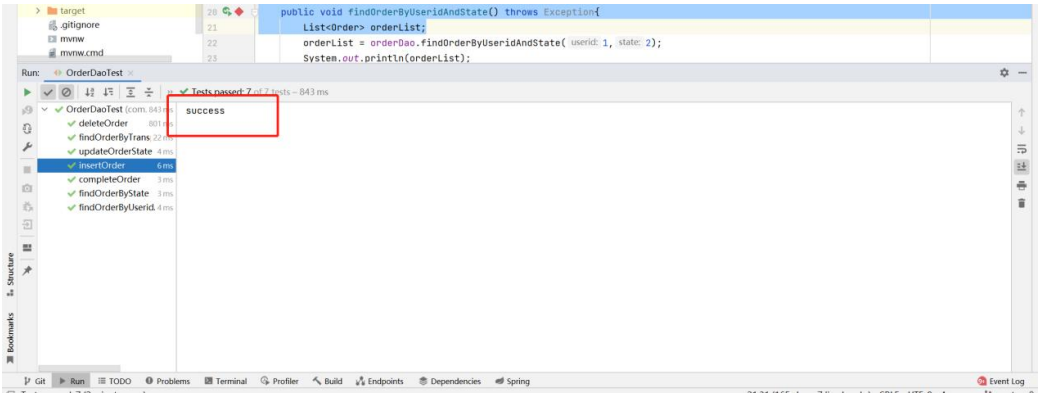

@RunWith(SpringRunner.class)
@SpringBootTest
public class UserDaoTest {
@Autowired
private UserDao userDao;
@Test
@Rollback
Run: UserDaoTest
Tests passed: 4 of 4 tests – 810 ms
2023-05-22 22:08:40.418 INFO 7888 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
[User{userid='1', username='lkj',password='123',auth='0'}, User{userid='2', username='lc',password='123',auth='1'}, User{userid='3', username='111',
Tests passed: 4 (a minute ago)

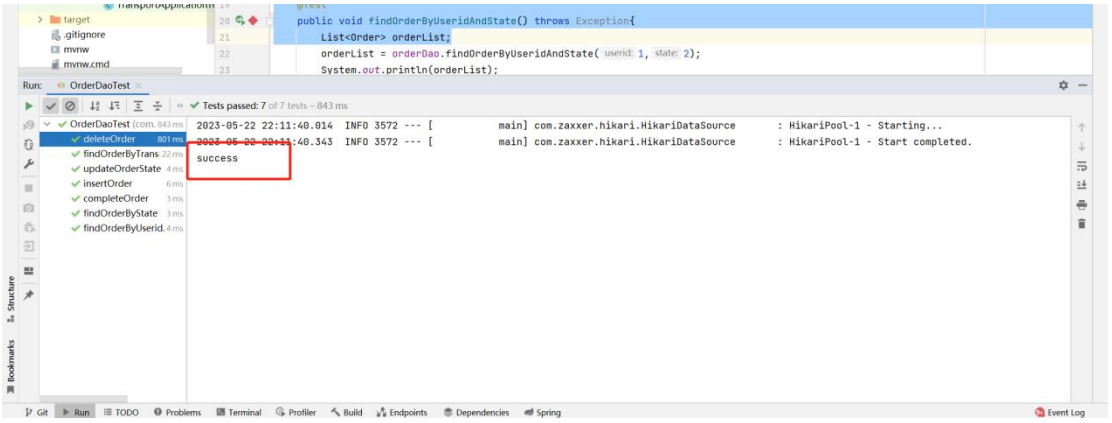

Run: UserDaoTest
Tests passed: 4 of 4 tests – 810 ms
User{userid='1', username='lkj',password='123',auth='0'}


Run: UserDaoTest
success


com.example.transport
controller
ContractControllerT
UserControllerTest
dao
ContractDaoTest
OrderDaoTest
UserDaoTest
@Autowired
private UserDao userDao;
@Test
@Rollback
public void findByUsernameAndPassword()throws Exception{
Run: UserDaoTest
Tests passed: 4 of 4 tests – 810 ms
User{userid='1', username='tkj',password='123',auth='0'}
findALL
findByUsernameAn
insertUser
findByUsername
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log
Tests passed: 4 (2 minutes ago)
17:25 (11 chars)
CRLF
UTF-8
4 spaces
master


UserDaoTest
TransportApplicationT
target
.gitignore
mvnw
mvnw.cmd
@Test
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
Tests passed: 7 of 7 tests – 843 ms
success
OrderDaoTest
deleteOrder
findOrderByTrans
updateOrderState
insertOrder
completeOrder
findOrderByState
findOrderByUserid
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log


target
.gitignore
mvnw
mvnw.cmd
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
Tests passed: 7 of 7 tests – 843 ms
success
OrderDaoTest
deleteOrder
findOrderByTrans
updateOrderState
insertOrder
completeOrder
findOrderByState
findOrderByUserid
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log


target
.gitignore
mvnw
mvnw.cmd
public void findOrderByUseridAndState() throws Exception{
List<Order> orderList;
orderList = orderDao.findOrderByUseridAndState(userid: 1, state: 2);
System.out.println(orderList);
Run: OrderDaoTest
Tests passed: 7 of 7 tests – 843 ms
2023-05-22 22:11:40.014 INFO 3572 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-05-22 22:11:40.343 INFO 3572 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
success
OrderDaoTest
deleteOrder
findOrderByTrans
updateOrderState
insertOrder
completeOrder
findOrderByState
findOrderByUserid
Git
Run
TODO
Problems
Terminal
Profiler
Build
Endpoints
Dependencies
Spring
Event Log

总结和展望：

本报告详细介绍了基于前后端分离框架的物流控制系统的系统设计，采用了 Spring Boot、Thymeleaf、MyBatis、Bootstrap、JWT、Java Security、Vue、Element Plus 和 Restful API 等技术。

其中：

1.前后端分离架构：系统采用前后端分离的架构模式，将前端和后端逻辑分离开发。前端使用 Vue 框架和 Element Plus 组件库进行开发，负责展示界面和与用户交互。后端使用 Spring Boot 框架搭建服务端，提供数据和业务处理的接口。

2.Spring Boot：作为后端开发框架，Spring Boot 提供了快速构建和部署 Java 应用程序的能力。它集成了许多常用的库和组件，简化了开发流程，提高了开发效率。

3.Thymeleaf 和 Bootstrap：Thymeleaf 是一种用于构建模板引擎的服务器端 Java 网页模板技术，与 Spring Boot 框架集成，可用于前端界面的渲染和数据绑定。Bootstrap 是一个用于构建响应式布局和样式的前端框架，提供了丰富的 UI 组件和样式。

4.MyBatis：作为持久层框架，MyBatis 简化了数据库操作和 SQL 语句的编写。它提供了对关系型数据库的访问和操作，可以与 Spring Boot 集成，实现数据持久化的功能。

5.JWT 和 Java Security：系统使用 JWT（JSON Web Token）进行用户身份验证和授权。JWT 是一种用于安全传输声明的开放标准，基于令牌的方式实现用户认证和授权。Java Security 提供了在后端验证和授权 JWT 令牌的功能。

6.Restful API：系统采用 Restful API 作为前后端交互的方式，通过 HTTP 协议传输数据。使用 REST 架构风格，定义了清晰的 API 接口，支持各种 HTTP 操作（GET、POST、PUT、DELETE），实现了数据的增删改查功能。

通过采用上述技术，该物流管理系统实现了良好的前端用户体验和后端数据处理能力，提供了灵活的、可扩展的控制系统。

目前该物流管理系统已经拥有了较为全面的订单系统以及模拟运输模块等，且系统已具有一定的安全性（Java security，JWT）。团队将会在后续的过程中设计微服务更加全面化系统功能。