****

**学年学期： 2021 - 2022 学年 2 学期**

**课程名称： 商品管理系统**

**课程学分： 2 考核形式： 大作业**

**任课教师： 徐贤胜**

**题 目： 基于前后端分离开发模式的商品管理系统**

**学 院： 数据学院 专业班级：信计（专升本）211**

**学 号： 3210443005 学生姓名： 仰菁斌**

**提交日期： 2022年7月21日**

|  |  |
| --- | --- |
| **评分标准：**   1. **在规定日期前提交作业 [10分]** 2. **论文格式规范、美观，图文编排合理，结构合理 [10分]** 3. **论文内容 [80分]** 4. 论文选题与内容符合，思路清晰 **[30分]** 5. 软件操作熟练，求解过程准确 **[20分]** 6. 实验结果合理准确，可重复 **[20分]** 7. 答辩能准确回答问题，对课程内容熟悉 [**10**分**]** | |
| **教师考核评语：** | |
| **考核得分** |  |
| **任课教师/评阅人签名** | **徐贤胜** |
| **评阅日期** | **2022年6月23日** |

（注：此表每名学生一份，随学生大作业或论文等形式的考核材料一并存档）

# 摘 要

商品管理系统。

系统整体开发采用前后端分离的设计模式，前端使用VUE + element ui 设计，后端使用Spring Boot框架。数据库使用MySQL。

系统实现了登陆注册、商品的增、删、查、改等功能模块。编码规范，注释充分，界面设计友好。

**关键词：**前后端分离；VUE；element ui；Spring Boot；商品管理，logback,slf4j；

目录

[摘 要 3](#_Toc109242134)

[第1章 绪论 6](#_Toc109242135)

[1.1 开发背景 6](#_Toc109242136)

[1.2 开发模式及技术 6](#_Toc109242137)

[第2章 项目开发方法 7](#_Toc109242138)

[2.1相关技术框架介绍 7](#_Toc109242139)

[2.1.1 Spring Boot框架 7](#_Toc109242140)

[2.1.2 Spring Date 8](#_Toc109242141)

[2.1.3 MySQL 9](#_Toc109242142)

[2.1.4 VUE 9](#_Toc109242143)

[2.1.4 Slf4j 9](#_Toc109242144)

[第3章 功能介绍 10](#_Toc109242145)

[3.1 登录 10](#_Toc109242146)

[3.1.1 登录界面 10](#_Toc109242147)

[3.1.2 登录界面实现脚本代码 10](#_Toc109242148)

[3.2 商品管理 12](#_Toc109242149)

[3.2.1 商品管理主界面 12](#_Toc109242150)

[3.2.2 主界面实现脚本代码 12](#_Toc109242151)

[3.3 添加商品信息 15](#_Toc109242152)

[3.3.1 添加商品信息页面 15](#_Toc109242153)

[3.3.1 添加商品信息脚本代码 16](#_Toc109242154)

[3.4 查询商品信息 19](#_Toc109242155)

[3.4.1 查询商品信息页面 19](#_Toc109242156)

[3.4.2 查询商品信息脚本代码 19](#_Toc109242157)

[3.5 修改商品信息 20](#_Toc109242158)

[3.5.1 ARIMA(1,1,0)模型 20](#_Toc109242159)

[3.5.2 修改商品信息脚本代码 20](#_Toc109242160)

[第4章 数据库 23](#_Toc109242161)

# 绪论

## 开发背景

本系统为商品管理系统。

随着我国经济的飞速发展，各种类型规模的公司企业迅速崛起，许多从事生产和经营管理的企业都有自己生产和销售的产品，而这些产品都需要储存在仓库中，对于每个企业来说，随着企业规模的不断扩大，产品数量的急剧增加，所生产产品的种类也会不断地更新与发展，有关产品的各种信息量也会成倍增长。面对庞大的产品信息量，如何有效地管理库存产品，对这些企业来说是非常重要的，库存管理的重点是销售信息能否及时反馈，从而确保企业运行效益。而库存管理又涉及入库、出库的产品、经办人员及客户等方方面面的因素，如何管理这些信息数据，是一项复杂的系统工程，充分考验着仓库管理员的工作能力，工作量的繁重是可想而知的，所以这就需要由商品管理系统来提高库存管理工作的效率，这对信息的规范管理、科学统计和快速查询，减少管理方面的工作量，同时对于调动广大员工的工作积极性，提高企业的生产效率，都具有十分重要的实现意义。

## 开发模式及技术

本系统基于前后端分离的开发模式设计，前端使用VUE + element ui 设计，后端使用Spring Boot框架。数据库使用MySQL。

系统实现了登陆注册、商品的增、删、查、改等功能模块。编码规范，注释充分，界面设计友好。

# 项目开发方法

## 2.1相关技术框架介绍

### 2.1.1 Spring Boot框架

Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者。

Spring框架是Java平台上的一种开源应用框架，提供具有控制反转特性的容器。尽管Spring框架自身对编程模型没有限制，但其在Java应用中的频繁使用让它备受青睐，以至于后来让它作为EJB（EnterpriseJavaBeans）模型的补充，甚至是替补。Spring框架为开发提供了一系列的解决方案，比如利用控制反转的核心特性，并通过依赖注入实现控制反转来实现管理对象生命周期容器化，利用面向切面编程进行声明式的事务管理，整合多种持久化技术管理数据访问，提供大量优秀的Web框架方便开发等等。Spring框架具有控制反转（IOC）特性，IOC旨在方便项目维护和测试，它提供了一种通过Java的反射机制对Java对象进行统一的配置和管理的方法。Spring框架利用容器管理对象的生命周期，容器可以通过扫描XML文件或类上特定Java注解来配置对象，开发者可以通过依赖查找或依赖注入来获得对象。Spring框架具有面向切面编程（AOP）框架，SpringAOP框架基于代理模式，同时运行时可配置；AOP框架主要针对模块之间的交叉关注点进行模块化。Spring框架的AOP框架仅提供基本的AOP特性，虽无法与AspectJ框架相比，但通过与AspectJ的集成，也可以满足基本需求。Spring框架下的事务管理、远程访问等功能均可以通过使用SpringAOP技术实现。Spring的事务管理框架为Java平台带来了一种抽象机制，使本地和全局事务以及嵌套事务能够与保存点一起工作，并且几乎可以在Java平台的任何环境中工作。Spring集成多种事务模板，系统可以通过事务模板、XML或Java注解进行事务配置，并且事务框架集成了消息传递和缓存等功能。Spring的数据访问框架解决了开发人员在应用程序中使用数据库时遇到的常见困难。它不仅对Java:JDBC、iBATS/MyBATIs、Hibernate、Java数据对象（JDO）、ApacheOJB和ApacheCayne等所有流行的数据访问框架中提供支持，同时还可以与Spring的事务管理一起使用，为数据访问提供了灵活的抽象。Spring框架最初是没有打算构建一个自己的WebMVC框架，其开发人员在开发过程中认为现有的StrutsWeb框架的呈现层和请求处理层之间以及请求处理层和模型之间的分离不够，于是创建了SpringMVC。

### 2.1.2 Spring Date Jpa

Jpa是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 避免了几乎所有的 Jpa代码和手动设置参数以及获取结果集。Jpa可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Ordinary Java Object,普通的 Java对象)映射成数据库中的记录。

特点：

简单易学：本身就很小且简单。没有任何第三方依赖，最简单安装只要两个jar文件+配置几个sql映射文件。易于学习，易于使用。通过文档和源代码，可以比较完全的掌握它的设计思路和实现。

灵活：Jpa不会对应用程序或者数据库的现有设计强加任何影响。 sql写在xml里，便于统一管理和优化。通过sql语句可以满足操作数据库的所有需求。

解除sql与程序代码的耦合：通过提供DAO层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试。sql和代码的分离，提高了可维护性。

提供映射标签，支持对象与数据库的orm字段关系映射。

提供对象关系映射标签，支持对象关系组建维护。

提供xml标签，支持编写动态sql

### 2.1.3 MySQL

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。MySQL是一种关系型数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体低，尤其是开放源码这一特点，一般中小型和大型网站的开发都选择 MySQL 作为网站数据库。

### 2.1.4 VUE

VUE是一套用于构建用户界面的渐进式框架。Vue采用自底向上增量开发的设计，提供了MVVM数据绑定和可组合的组件系统，具有简单、灵活的API，通过简单的API可实现响应式的数据绑定和可组合的视图组件。

### 2.1.4 Slf4j

　slf4j，simple logging facade for java的缩写，翻译为java的简单日志外观。slf4j是一个开源项目，它提供我们一个一致的API来使用不同的日志框架，比如： java.util.logging，logback，log4j等。slf4j使用户可以在运行时嵌入他们想使用的日志框架。从名字中可以看出，它其实使用的是facade设计模式来实现的。

　　使用slf4j，只有一个强制性的依赖，就是slf4j-api-x.x.x.jar，我们在编写代码的时候，只会使用这个jar包里的API，应用程序在运行时去类路径下查找绑定的具体日志框架，并使用该绑定的日志框架进行实际的日志操作，如果在应用程序的类路径下面没有找到合适的绑定的话，slf4j默认使用一个没有任何操作的实现

# 功能介绍

## 3.1 登录

### 3.1.1 登录前端界面

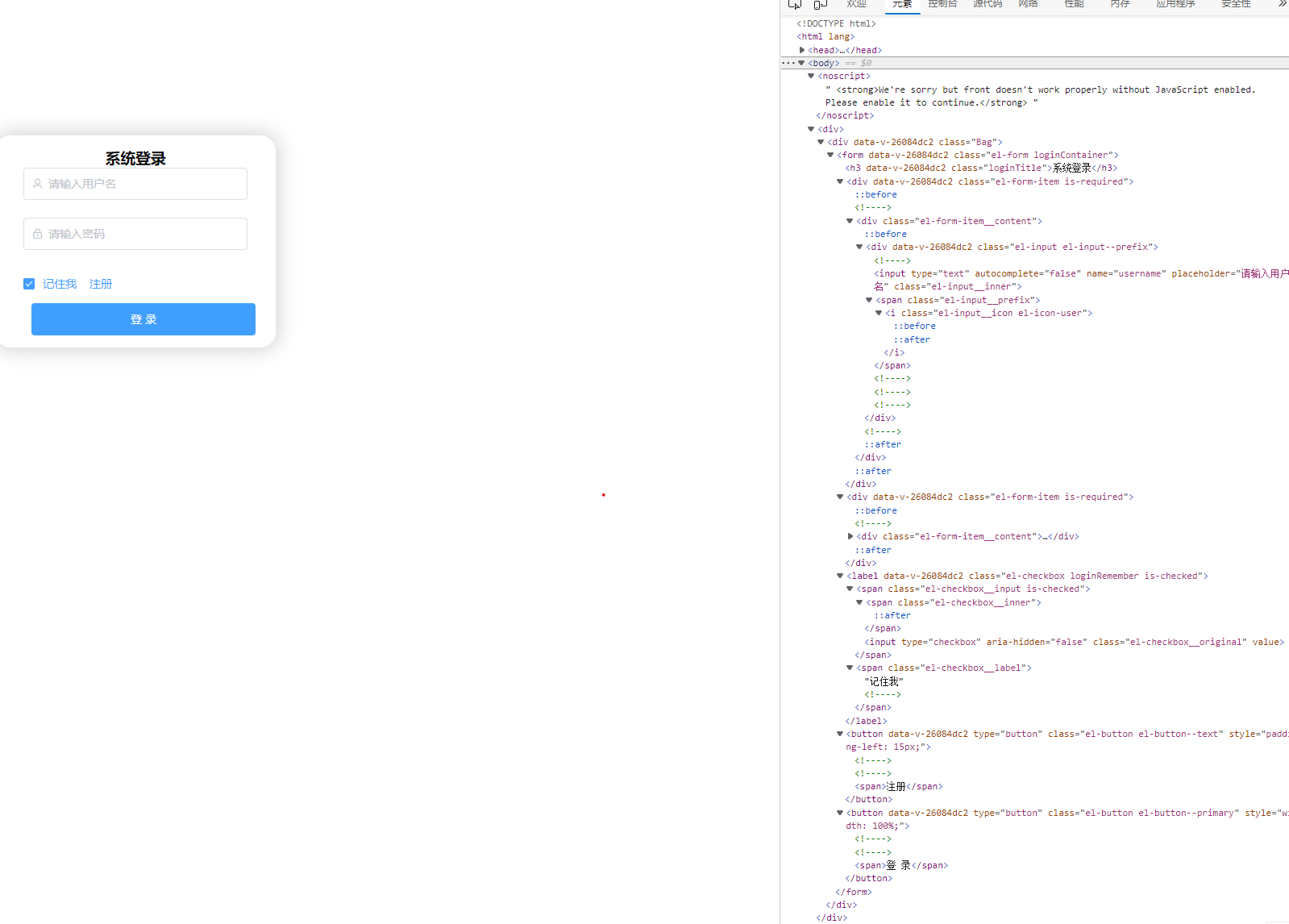
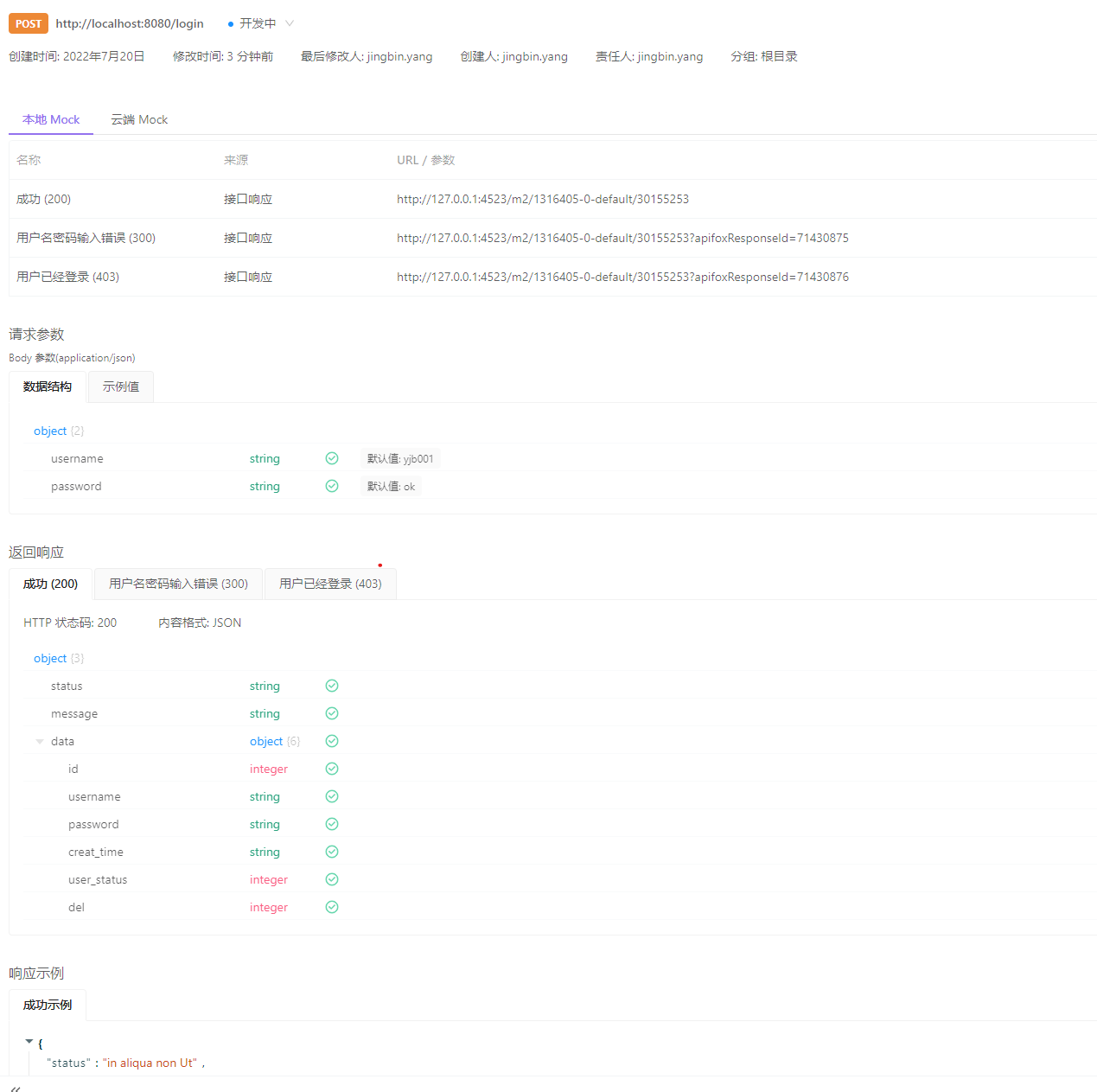


图3.1登录

### 

### 3.1.2 登录接口测试

链接: [https://www.apifox.cn/apidoc/shared-ddc1f729-dbc3-46dd-b2d9-959fcc679511](https://www.apifox.cn/apidoc/shared-ddc1f729-dbc3-46dd-b2d9-959fcc679511%20%20) 访问密码 : pF4D63WI



3.1.3登录接口实现

接口地址

@CrossOrigin

@PostMapping("login")

public Result login(@RequestBody User userl){}

1,首次登录情况

//获取当前时间

String currentTime = DateUtil.*getCurrentTime*();

//判断缓存中是否存在用户

User u = (User)redisTemplate.opsForValue().get(username);

//如果为首次登录

//存入缓存有效期为5分钟

redisTemplate.opsForValue().setIfAbsent(username,userByPasswordAndUser,3000, TimeUnit.*SECONDS*);

Result<User> userResult = Result.*bulidResult*(Status.*OK*, userByPasswordAndUser);

1. 密码输入错误情况

User user = new User(username, password);

Result<Object> errorResult = Result.*bulidResult*(Status.*PWD\_ERROR*,user);

//密码错误，打印日志

*log*.info("Incorrect account name or password input {}. Time {}.",user,currentTime);

return errorResult;

1. 重复登录情况
2. 拦截5s内重复提交/login

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

//组合判断条件，这里只用了url

String requestURL = request.getRequestURL().toString();

//拿到用户信息,暂时不用token

String username = request.getParameter("username");

//判断redeis是否已经存在数据，

Boolean ifAbsent = stringRedisTemplate.opsForValue().setIfAbsent(requestURL, "", Objects.*nonNull*(repeatSubmitByMethod) ? repeatSubmitByMethod.seconds() : repeatSubmitByCls.seconds(), TimeUnit.*SECONDS*);

//如果setIfAbsent为false，表示已经发送请求，直接抛出异常

if (ifAbsent != null && !ifAbsent) throw new RepeatSubmitException(username);

}

1. 验证缓存是否已经登录，打印并存储日志

Result<Object> exitResult = Result.*bulidResult*(Status.*EXIT*,u);

//用户已经登陆日志

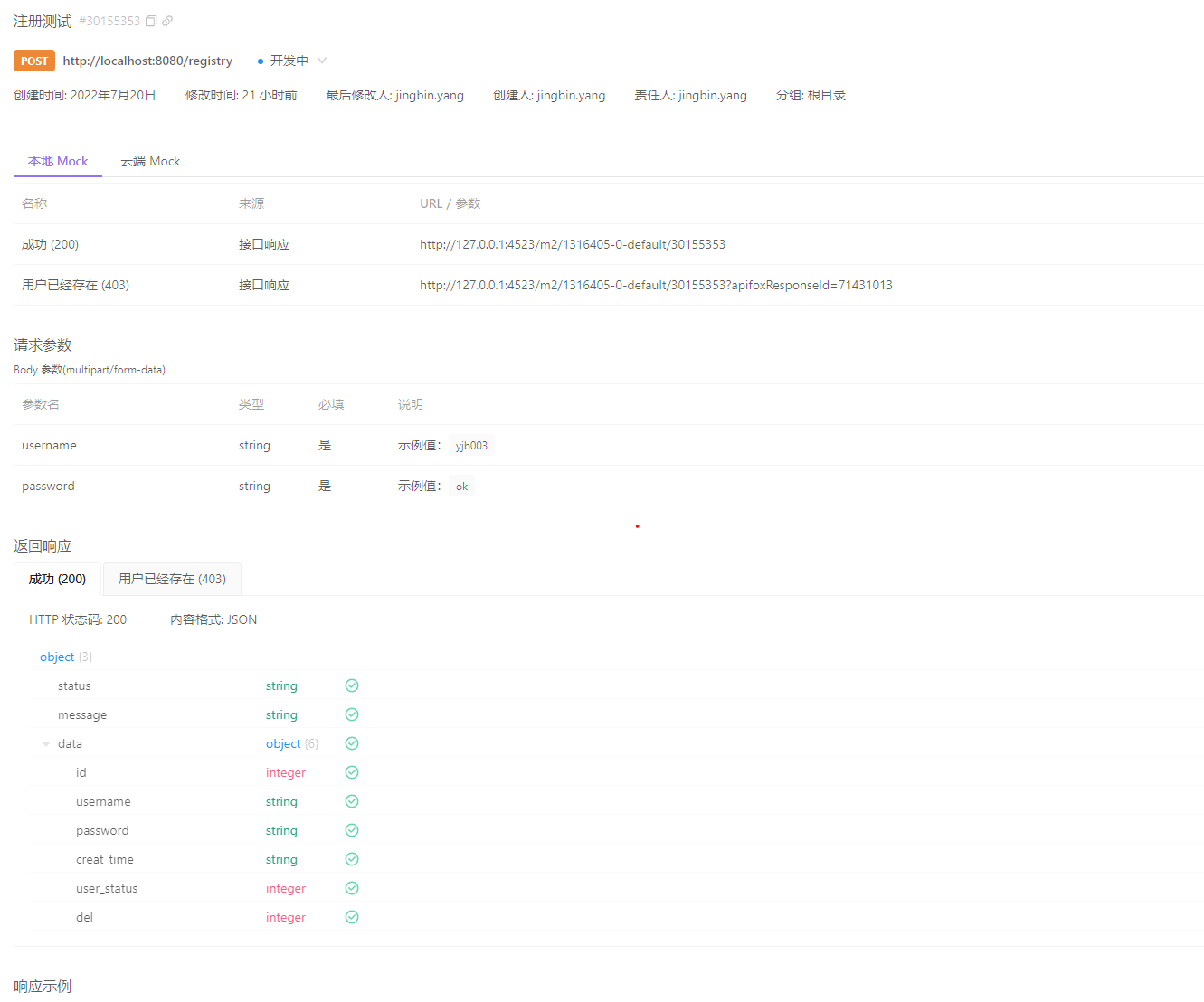
*log*.info("The user {}. has logged in the log Time {}.",u,currentTime);

3.1.4登录接口日志

//登陆成功，打印日志

*log*.info("The user logining successful {}. Time{}.",userByPasswordAndUser,currentTime);

### 3.1.3注册接口测试



## 3.2 商品管理

### 3.2.1 商品管理主界面

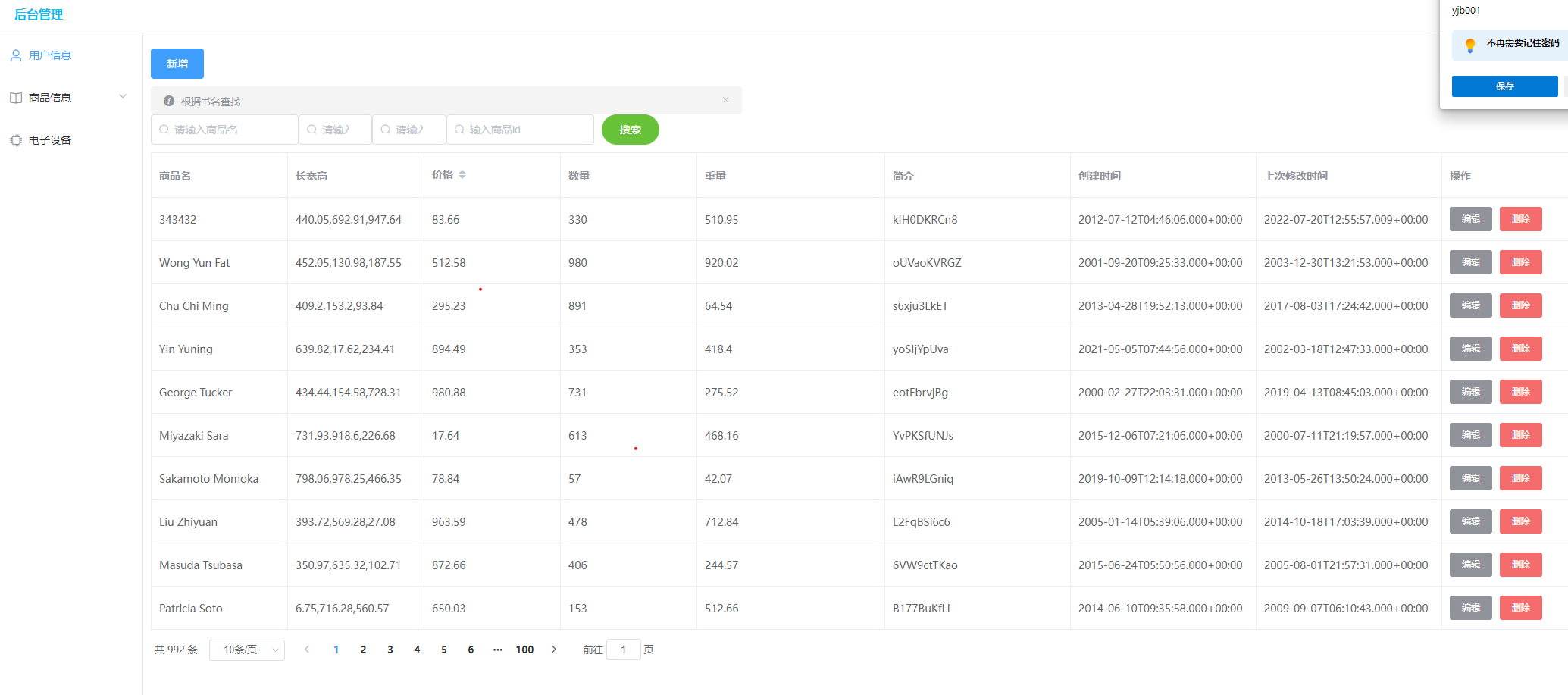


图3.2 商品管理主界面

### 3.2.2 主界面接口实现

接口一：普通分页请求

@GetMapping("page/{pageNo}/size/{size}")

@CrossOrigin

public Page getPage1(@PathVariable("pageNo") Integer pageNo,@PathVariable("size") Integer size){

Page<Good> goodNoCriteria = goodService.findGoodNoCriteria(pageNo,size);

redisTemplate.opsForValue().set(pageNo + "页", goodNoCriteria,5000, TimeUnit.*SECONDS*);

return goodNoCriteria;

}

Service层：findGoodNoCriteria( pageNo,size )，传入当前页码和每页的行数

Model层：数据模型Good，详见Swagger文档

接口二：带条件的分页请求

@GetMapping("page2/{pageNo}/size/{size}")

@CrossOrigin

@RepeatSubmit

public Page getPage2(@PathVariable("pageNo") Integer pageNo,@PathVariable("size") Integer size, GoodQuery goodQuery){

redisTemplate.setKeySerializer(new StringRedisSerializer());

//如果用户所有的输入都为空，查询全部数据

if (goodQuery.getId() == null && goodQuery.getName() == ""

&& goodQuery.getLessPrice() =="" && goodQuery.getMorePrice() == ""){

//如果缓存中存在应该从缓存中取出

Page ma = (Page)redisTemplate.opsForValue().get(pageNo + "页");

if (ma == null || ma.isEmpty()){

Page<Good> goodNoCriteria = goodService.findGoodNoCriteria(pageNo, size);

redisTemplate.opsForValue().set(pageNo + "页", goodNoCriteria,2000, TimeUnit.*SECONDS*);

return goodNoCriteria;

}

return ma;

}

//从mysql中拉取数据，此接口限制重复提交

Page<Good> goodCriteria = goodService.findGoodCriteria(pageNo,size,goodQuery);

return goodCriteria;

Service层：findGoodCriteria( pageNo, size， goodQuery )，传入当前页码和每页的行数，goodQuery 为数据查询模型

Model层：数据模型Good，详见Swagger文档

注：该接口无日志

### 3.2.3 主界面接口测试



## 3.3 添加或修改商品信息

### 3.3.1 添加或修改商品信息页面

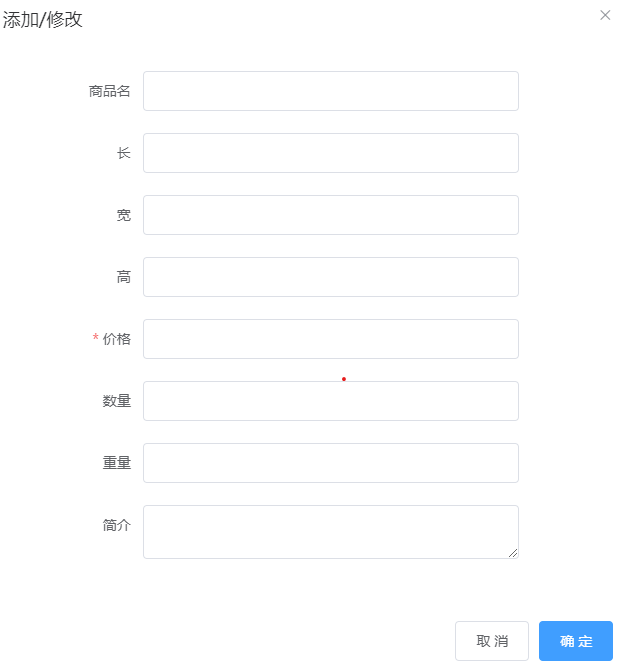


图3.4添加商品

### 3.3.2 添加或修改商品接口实现

1.1.接口地址

@PutMapping("save")

@CrossOrigin

public Result addOrUpdate(HttpServletRequest request, @RequestBody Good good){}

1.2.接口实现

String currentTime = DateUtil.*getCurrentTime*();

String username = request.getParameter("username");

User u = (User)redisTemplate.opsForValue().get(username);

//获取当前登陆用户的id

String uid = u.getId().toString();

1. 用户为新增则id为null

if (good.getId() == null){

//新增商品

good.setUser\_id(uid);

goodService.addOrSaveGood(good);

Result<Good> goodResult = Result.*bulidResult*(Status.*OK*, good);

*log*.info("Product {}. added successfully Time {}.",good,currentTime);

return goodResult;

}

1. 用户为修改id不为null

Good g = goodService.findById(good.getId());

g.setDescrip(good.getDescrip());

g.setHeigth(good.getHeigth());

g.setLength(good.getLength());

g.setName(good.getName());

g.setPrice(good.getPrice());

g.setDel(good.getDel());

g.setWidth(good.getWidth());

g.setImg(good.getImg());

g.setWeigth(good.getWeigth());

g.setModify\_time(DateUtil.*getCurrentTimeStamp*());

g.setUser\_id(uid);

Good good1 = goodService.addOrSaveGood(g);

Result<Good> updateResult = Result.*bulidResult*(Status.*OK*, good1);

*log*.info("Product {}. modification successfully Time {}.",good1,currentTime);

return updateResult;

注意：Jpa框架save( )方法可根据模型主键id是否为空自动实现新增或者修改

### 3.3.3 添加或修改商品接口测试

## 

## 3.4 删除商品

### 3.4.1 查询商品信息页面

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

### 3.4.2 删除商品接口实现

1.接口地址

@DeleteMapping("delete/{gid}")

@CrossOrigin

public Result deleteGood(@PathVariable("gid") Integer gid){}

1. 接口实现及日志

String currentTime = DateUtil.*getCurrentTime*();

goodService.deleteGood(new Good(gid));

Result<Integer> gidResult = Result.*bulidResult*(Status.*OK*, gid);

*log*.info("Product {}. deleted successfully Time {}.",gid,currentTime);

return gidResult;

### 3.4.3 删除商品接口测试



总结：该项目后端主要使用了SpringBoot框架，利用maven导入了启动器

spring-boot-starter-web,spring-boot-starter-data-jpa,spring-boot-starter-cache

spring-boot-starter-data-redis,spring-boot-starter-test，启动器会自动会运行配置类，注册关键的Bean到容器中，比如说用的的日志类log，这样我们就使用这些启动器的强大功能。

主要使用redis做了商品缓存，减少了部分查询数据库的开销。

# 数据库

本系统使用MYSQL,Redis数据库。

根据模型,共需要两张表，分别为用户表和商品表。

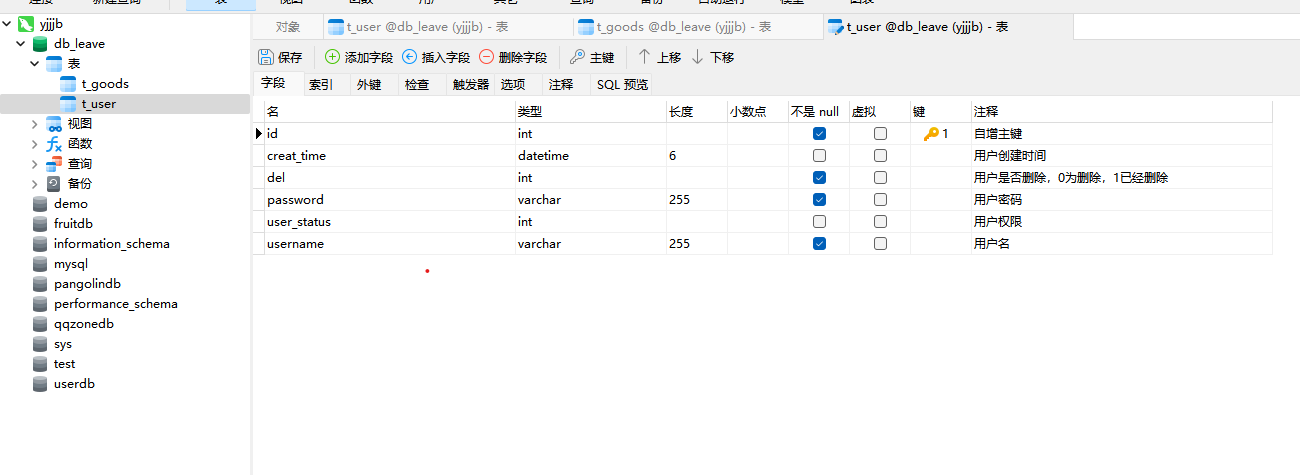


图4.1用户信息表

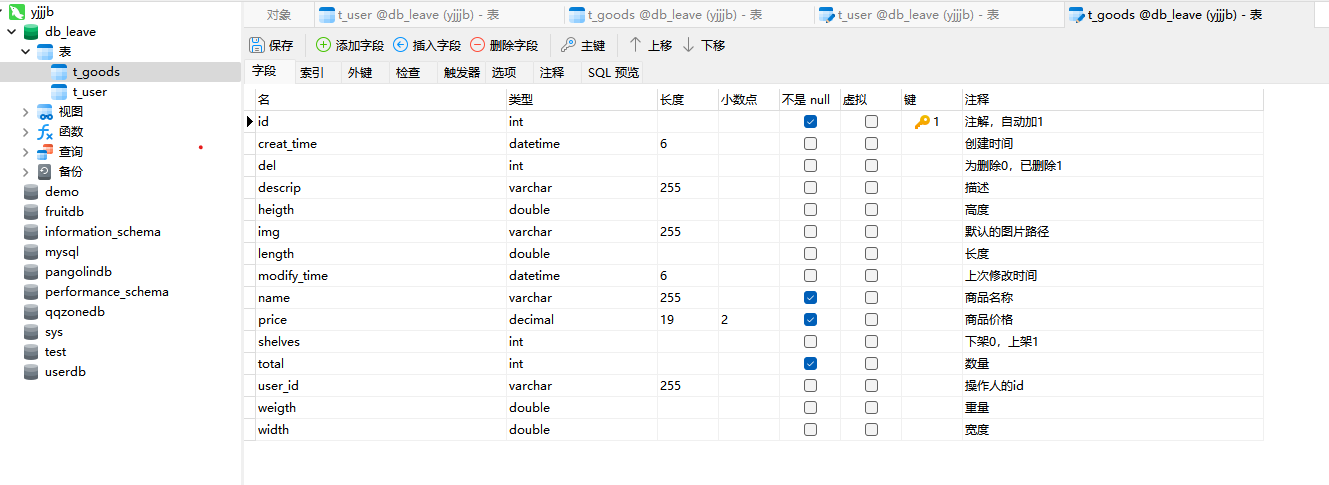


图4.2商品信息表

图4.2商品信息表

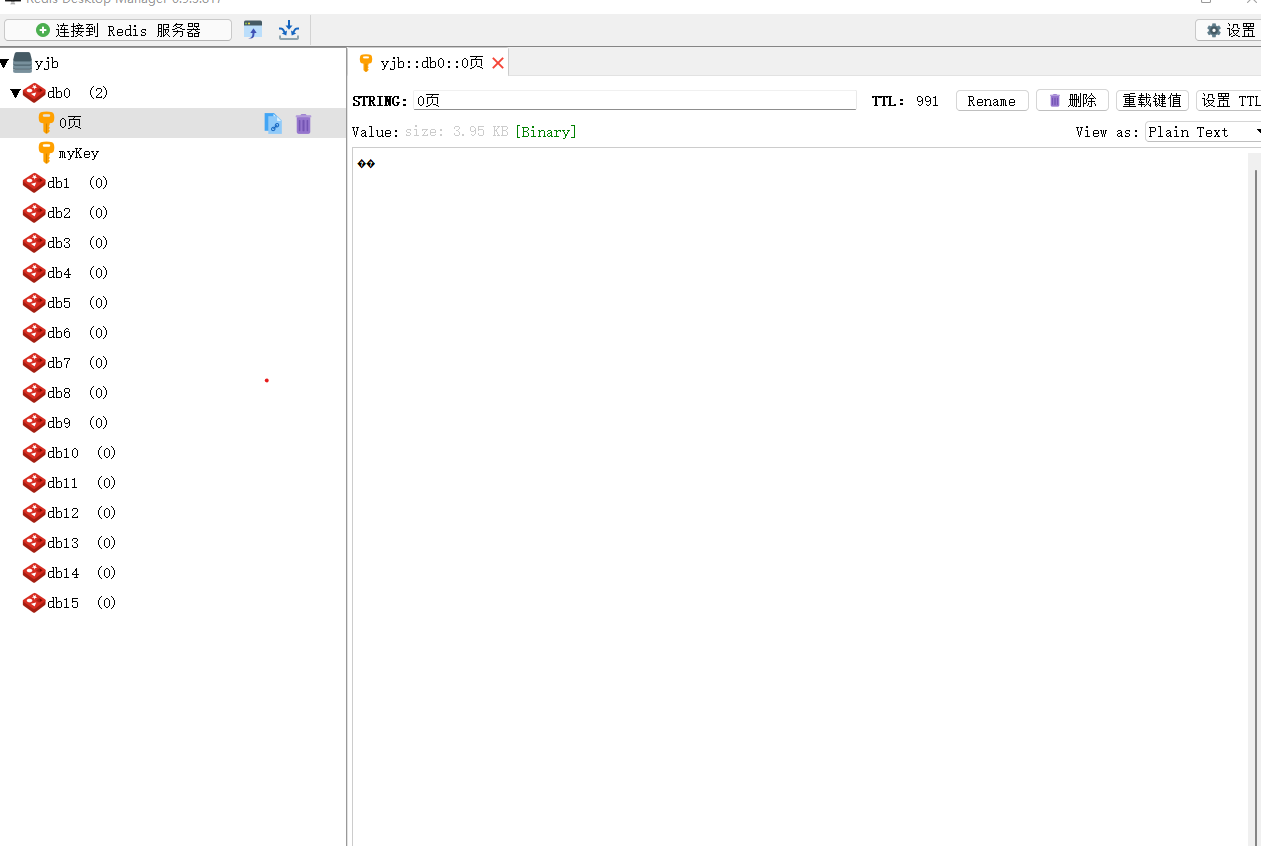


图4.3 redis缓存商品和用户登录信息

# Docker镜像配置prometheus并使用Grafana监控面板

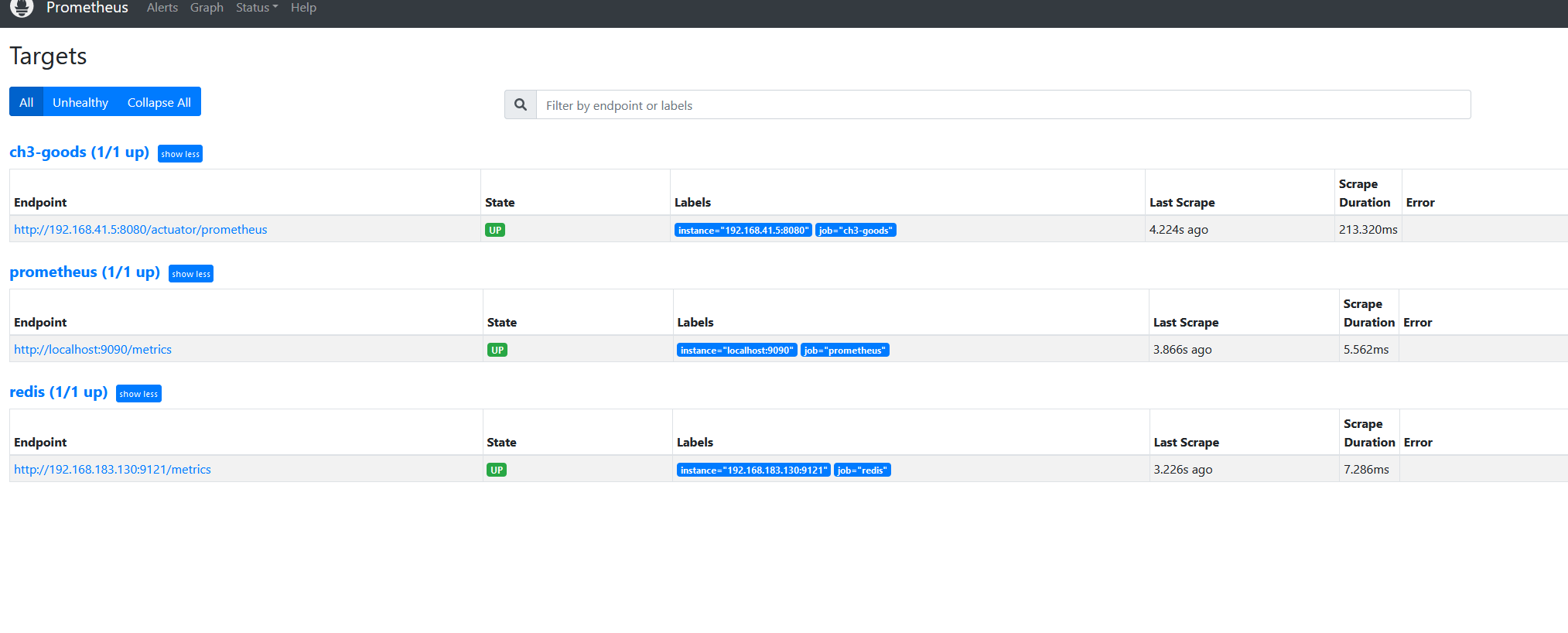


图5.1 prometheus的Targets列表

5.1 项目上线

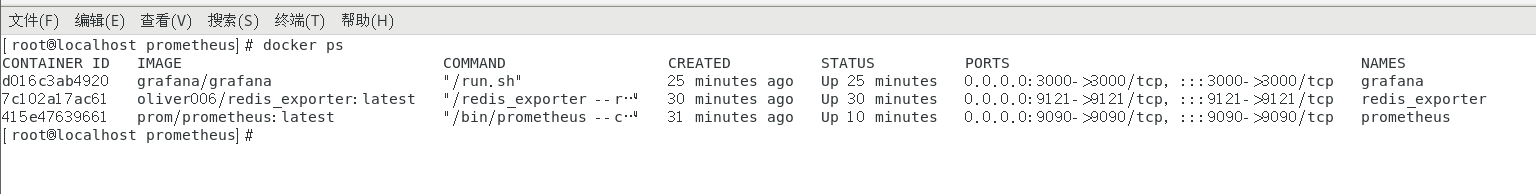
访问url: <http://192.168.183.130:9090/>

promtheus部署在远程linux服务器http://192.168.183.130:9090,

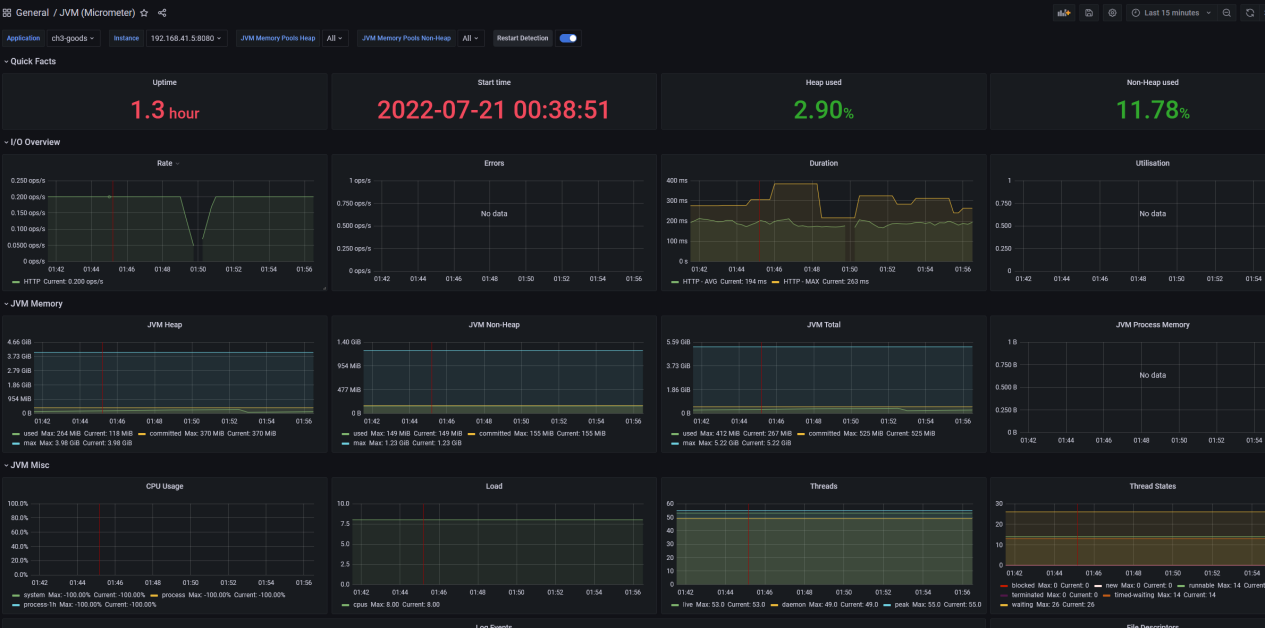
Redis数据库部署在本机,http://192.168.41.5:6379

SpringBoot 项目部署在本机http://192.168.41.5:8080

5.2 Docker 容器



5.3 JVM监控

5.4 Redis 监控

