JIANGXI NORMAL UNIVERSITY

江西师范大学

本科生毕业设计(论文)

中文题目 基于 Web 的进销存仓库管理系统

外文题目 Warehouse Managment System Based on Web

学　　号 201726702112

姓　　名 励轩

学　　院 软件学院

专　　业 软件工程

指导教师 邓有莲

完成时间 2021 年 5 月

江西师范大学教务处制

声　　明

本人郑重声明：

所呈交的毕业设计（论文）是本人在指导教师指导下进行的研究工作及取得的研究成果。其中除加以标注和致谢的地方，以及法律规定允许的之外，不包含其他人已经发表或撰写完成并以某种方式公开过的研究成果，也不包含为获得其他教育机构的学位或证书而作的材料。其他同志对本研究所做的任何贡献均已在文中作了明确的说明并表示谢意。

本毕业设计（论文）成果是本人在江西师范大学读书期间在指导教师指导下取得的，成果归江西师范大学所有。

特此声明。

声明人（毕业设计（论文）作者）学号：201726702112

声明人（毕业设计（论文）作者）签名：

签名日期：　2021 年 5 月 18 日

摘 要

小到超市大到企业都需要进行仓库管理。经营者通过使用仓库管理系统，可以更充分地了解库存波动，从而采取更有效的管理手段来提高经营效率。

本文实现的系统采用基于 Web 的分层架构模式，应用 SpringBoot、Shiro、MybatisPlus 等技术进行开发，实现了进货、出货、退货、存货、数据维护以及权限管理 6 个功能点，并能在库存量达到预警线时进行及时预警，基本满足了仓储管理人员的功能需求。

在系统设计中参考了软件生命周期原理，将设计过程划分为:计划和需求分析、项目架构设计、开发和编程、测试 4 个阶段。论文开篇介绍了系统开发背景以及目的；其次围绕着数据、功能、非功能需求，论述了开发目标；随后结合 UML 图对业务流程进行细化；紧接着阐述了系统架构的设计思路和关键功能点的实现步骤；最后对本次设计分析了不足之处，并进行总结和展望。

关键词：仓库管理；权限管理；库存预警；SpringBoot；Shiro

Abstract

From supermarket to enterprise, warehouse management is required. By using the warehouse management system, operators can better understand inventory fluctuations and adopt more effective methods to improve efficiency.

The system implemented in this paper adopts a web-based hierarchical architecture model, and applies SpringBoot, Shiro, MybatisPlus and other technologies for development, and realizes six functional points of purchase, sale, return goods, inventory, data maintenance and authority management. On this basis, the system can provide timely warning when the inventory reaches the warning line. It can basically meet the functional requirements of managers.

According to the Software Life Cycle, the system is divided into four stages: planning and demand analysis, project architecture, programming and testing. At the beginning of the paper, it introduces the background and purpose of system development. Secondly, it analyzes the goals of system development around the data, function, and non-functional requirements. Thirdly, it combines the UML diagram to refine the process of business and describe the main process. Fourthly, it introduces the architecture of the system and the design ideas of key function points. At last, the deficiencies of this system design are analyzed and prospected.

Key words： warehouse managment; authority management; Stock warning; SpringBoot; Shiro

目 录

第 1 章 绪论

1.1 开发背景

仓库管理对保障企业发展、降低经营成本、加快物资流转有着重要影响，其核心在于站在仓储经营的角度将物品信息进行统一管理，并集中维护准确可靠的仓储信息源。

仓库管理系统相较于传统的库存管理方式，能够在以下方面大显身手：数据持久化便于数据备份和恢复；支持筛选条件检索，提高查找效率；更紧密地将进销存业务结合，协调各部门之间的工作。

目前主流的仓库管理系统大致可以分为 4 种类型：第一类针对零售配送业务设计，如天猫的同城半日达，通过提高出入库效率来加快配送速度。第二类针对电商设计，如亚马逊，产生订单后将出库指令发送给仓库管理系统，在配货拣货后商品出库。第三类针对顺丰等物流设计，将各地的派送物品运送至中转仓，通过仓库管理系统规划物品的派送计划。第四类针对各类制造产业设计，将材料自动分配至流水线用于生产。

虽然仓库管理系统具有不同种类，但企业管理体制却大同小异：采购人员根据库存报表生成请购单，制定了采购计划；财务部门审核请购单，批准采购部门进行物资采购；物资经过库存部门验收入库后，将定期进行库存盘点和损耗统计；统计部门按月进行统计分析并生成相应报表，抄送至采购部门。

由此可见，完整的库存管理流程包含了入库、出库、库存、分析报表、财务等模块，其中入库、出库、库存是重中之重，也是系统的核心所在。

1.2 开发目标、意义

本系统旨在迎合中小型仓储企业的需求，实现基础的进销存三项业务以及日常的数据维护和权限管理功能，能够解决库存信息不及时、不准确等问题。

系统将基于 Web 进行开发设计，能够多方面地辅助仓储工作：以系统代替人工，减轻工作量并减少人力成本；提供高效管理，加快物资周转速度；物品信息持久化，实现信息的备份与恢复；依托数据库来统一管理数据，简化查询分析的步骤，并保证数据准确性；对用户操作进行限制，避免了非法用户对数据的恶意操作。

1.3 论文主要工作

通过研究仓储企业的经营管理并分析其系统需求，设计包括进货、出货、退货、存货、数据维护以及权限管理功能的仓库管理系统，实现库存数据可视化。

本文完成的工作如下：运用软件工程原理，将系统设计划分为计划和需求分析、项目架构设计、开发和编程、测试 4 个阶段。第一阶段：通过研究仓库管理系统的现状与业务内容，总结出系统的三种角色，进而根据角色的业务需求划分功能模块，并使用 UML 建模语言对功能点进行细化。第二阶段：确定了系统基于 Web 的主架构，并根据功能的特殊需求进行技术选型；第三阶段：结合选择的技术对系统的各个功能模块进行实现；第四阶段：运用多种测试手段测试开发完成的关键功能点。

本系统具有界面友好、功能完善、操作方便、数据安全的特色，企业员工只需经过简单的培训，就能够非常熟练地使用本软件。将 B/S 三层架构技术和进销存业务良好地结合在一起，基本满足了仓储管理人员在功能上需求，提高了企业仓储管理人员的办公效率。在系统的架构设计上通过分层提高了系统的灵活性，能够更轻易地适应需求变更。而权限管理也使得员工具有不同的权限。仓库管理人员通过使用本系统可以轻松地管理和控制库存物资的出入库及存储现状，从而实时掌握库存情况，降低管理仓库的人工成本。

1.4 论文结构

论文的组织结构如下：第一章绪论，介绍了系统开发背景、目标和意义。第二章需求分析，介绍了系统在功能、非功能、数据三方面的需求。第三章系统设计，结合 UML 统一建模语言介绍了系统的功能模块和数据库的设计。第四章模块实现，介绍了系统的架构以及功能模块的实现步骤。第五章总结和展望，总结并分析本次设计内容，并提出未来的计划。

第 2 章 需求分析

2.1 功能需求

2.1.1 角色分析

包括：出入库管理员、库存管理员和系统管理员 3 种。

表 2.1 角色分析

出入库管理员	管理出入库及退货信息
库存管理员	库存损耗，库存统计，维护业务数据
系统管理员	维护系统数据，用户权限管理

2.1.2 总用例分析

出库、入库，数据维护、库存损耗、库存统计、用户权限管理 6 个用例。

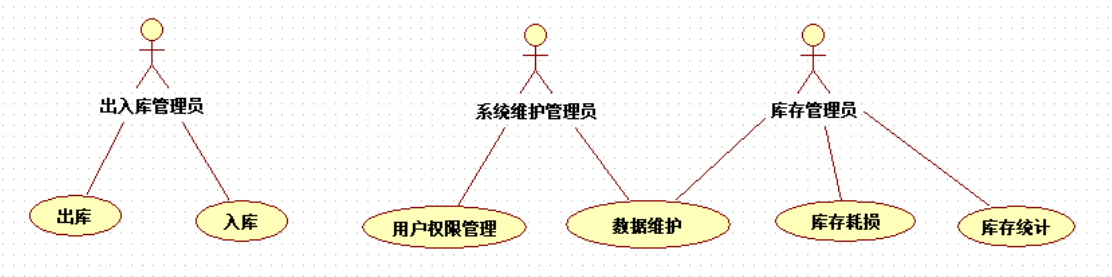

图 2.1 总用例图

2.1.3 数据维护用例分析

在数据维护用例的参与者中，库存和系统管理员分别负责维护业务和系统数据。将数据维护用例根据用户行为，细分业务和系统数据维护 2 项子用例。

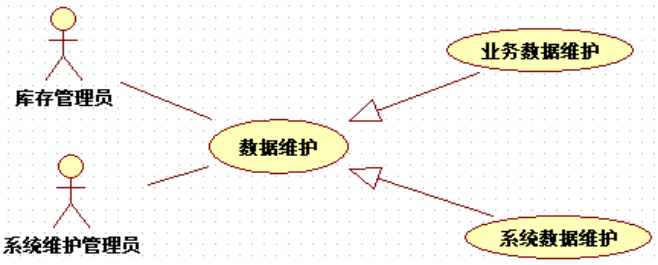

图 2.2 数据维护用例图

其中业务数据维护用例与商品、供应商、客户管理之间存在泛化关系。

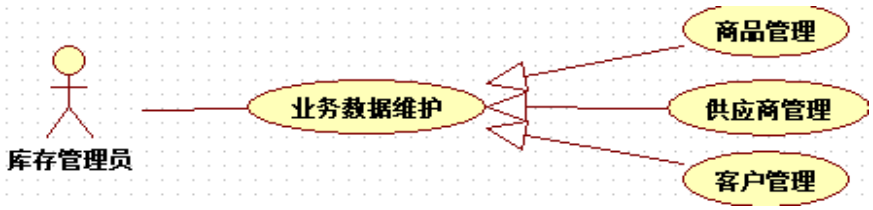

图 2.3 业务数据维护用例图

商品管理用例：商品信息录入，图片上传；供应商管理用例：供应商，地址等信息的录入；客户管理用例：客户，联系方式等信息的录入。

表 2.1 商品管理用例规约

参与者		库存管理员
前置条件		成功登陆本系统，被授予商品管理的权限
	动作	系统响应
基本事件流	上传图片	对合法图片进行压缩上传，并刷新页面中的缩略图
	添加商品信息	录入信息至商品表，刷新商品列表网页，提示添加成功
异常事件流	商品价格非法	提示价格格式非法
	未输入必填项	提示必填项未输入不能提交表单
	上传错误图片	拒绝上传并弹框提示图片格式非法

系统数据维护用例与日志查询、公告管理之间存在泛化关系。

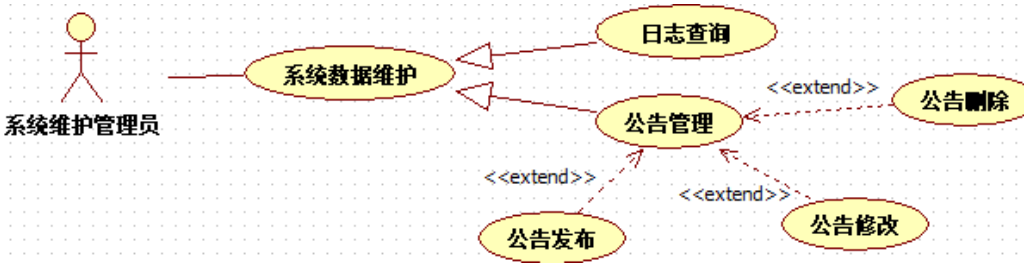

图 2.4 系统数据维护用例图

日志查询用例：日志数据的搜索；公告管理用例：公告展示在系统主页面上，公告的发布、修改、删除与公告管理用例间存在拓展关系。

表 2.2 发布公告用例规约

参与者		系统管理员
前置条件		成功登陆本系统，被授予发布公告的权限
	动作	系统响应

基本事件流	发布公告	提示发布成功，录入公告信息，主页面显示公告内容
	修改公告	提示修改成功，更新公告记录，显示修改后的公告内容
	删除公告	提示删除成功，主页面不显示该公告

2.1.4 权限管理分析

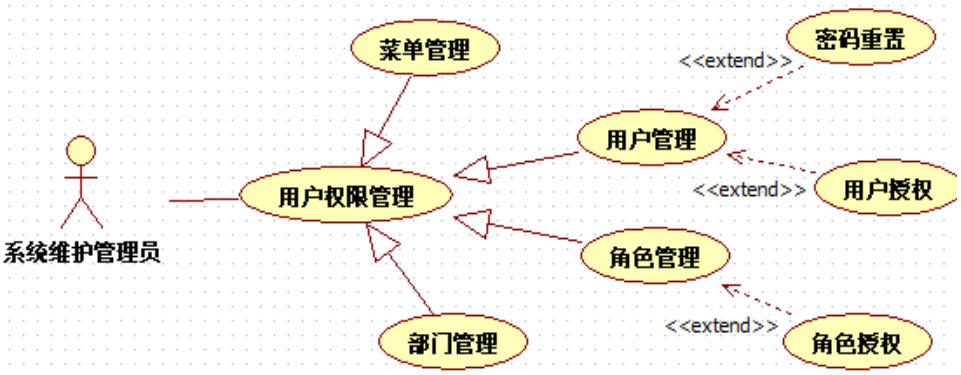

图 2.5 权限管理用例图

权限管理用例与菜单、部门、用户、角色管理之间存在泛化关系；密码重置和用户授权与权限管理用例间存在拓展关系；角色授权与角色管理间存在拓展关系。

表 2.3 权限管理用例规约

参与者		系统管理员
前置条件		成功登陆本系统，被授予权限管理的权限
	动作	系统响应
基本事件流	修改用户头像	对合法图片进行压缩上传，并刷新页面中的缩略图
	修改用户信息	修改用户信息至用户表，刷新用户列表网页，提示用户修改成功
可选事件流	密码重置	重置用户密码为系统预设值的默认密码 提示用户密码重置成功
	用户授权	弹出授权弹窗，展示该用户下拥有的所有角色信息

2.1.5 入库用例分析

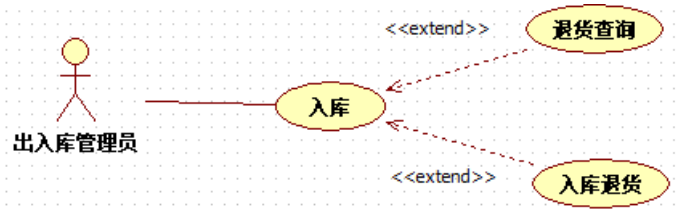

图 2.6 入库用例图

入库与退货查询，入库退货用例之间存在拓展关系。

表 2.4 入库用例规约

参与者		出入库管理员
前置条件		成功登陆本系统，被授予出入库管理的权限
	动作	系统响应
基本事件流	入库	入库信息校验，数量不能为负数，价格为正两位小数 对于非法入库信息，对用户进行错误弹窗 信息无误则生成入库单并录入数据库
可选事件流	入库退货	退货数量不能大于入库数量，且不能为负数
	退货查询	查询入库单下商品的退货信息，包含退货数量，退货日期等

2.2 数据需求

2.2.1 顶层数据流图

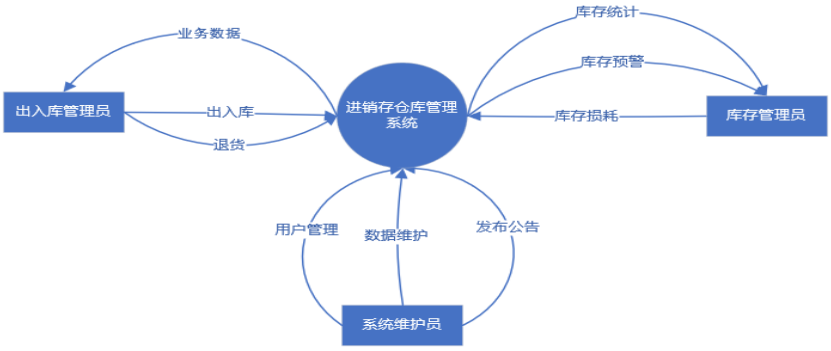

图 2.7 系统顶层数据流图

顶层数据流图用"进销存仓库管理系统"来代表所有加工。然后通过箭头表示输入输出数据流，表明了数据流向的范围。最后用矩形代表出入库管理员，系统维护员和库存管理员 3 个实体。

2.2.2 底层数据流图

将"进销存仓库管理系统"细分为多个加工。在绘制过程中遵循父子图数据流平衡原则，同时保持子图内部的数据流平衡：输入输出流不能单条地出现在加工上，并且加工前后的输入输出也要匹配。

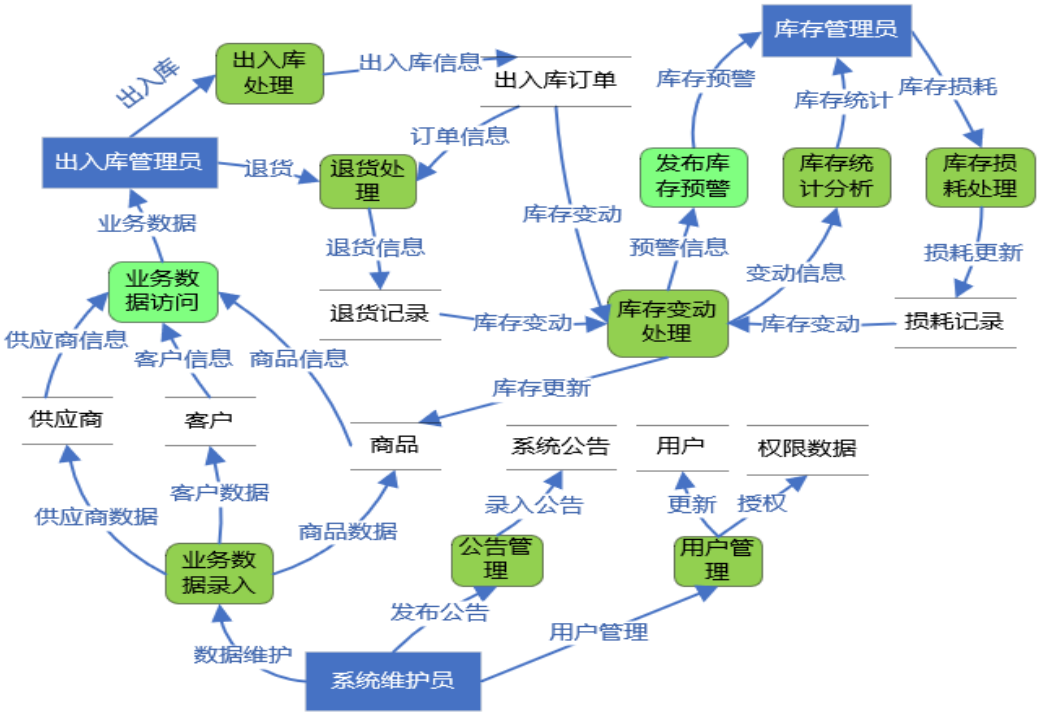

图 2.8 系统底层数据流图

以下挑选几条重要的加工进行描述：

退货处理：根据退货请求和出入库订单整理为退货信息并录入退货记录。

库存变动处理：处理出入库订单表，退货记录表，损耗记录表中的库存变动记录，若库存超出库存预警值上下限的范围，则将预警信息输出至"发布库存预警"。最后在更新商品库存的同时，将库存变动信息输出至"库存统计分析"。

发布库存预警：对预警信息进行加工处理，向库存管理员进行库存预警。

库存统计分析：库存变动信息处理为库存统计报单，展示给库存管理员。

库存损耗处理：对损耗记录进行加工处理，更新损耗记录表。

2.2.3 概念结构设计

将系统的需求抽象为数据库设计的概念模型，并使用 ER 图描述实体和实体间的联系。根据底层数据流图可知仓库管理系统包括供应商、商品、客户、用户、部门、公告等实体。

其中，供应商和用户，客户和用户皆为一对一关系：用户一次进货可以从一个供应商进货多个商品，也可以向一个客户销售多个商品；用户和商品、部门、公告、角色、权限都为一对多关系。

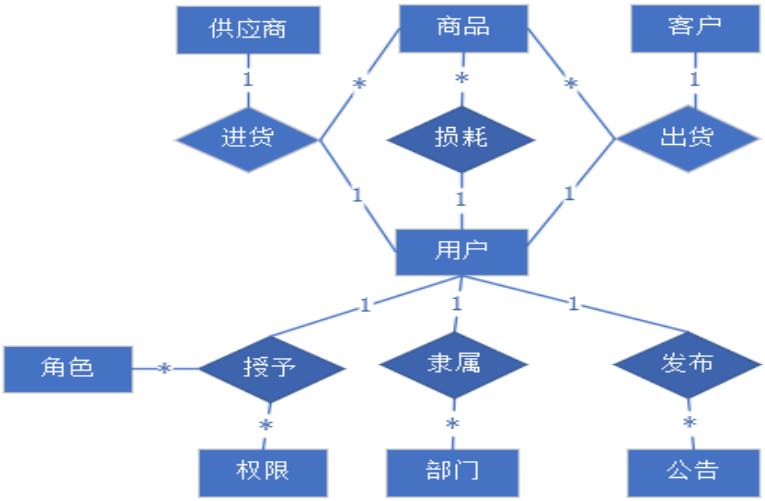

图 2.9 仓库管理系统 ER 图

2.3 非功能性需求

易用性：页面简洁明了，易上手。

安全性：针对不同职位的系统使用人员设置不同访问权限。

可扩充性：能在少量改动的情况下提高系统负载能力。

界面需求：系统的登录、主界面风格保持一致，以绿白两种颜色作为主色调。

可维护性：采用面向对象的分析来对系统功能进行建模，注重可维护性。

数据完整性：保证数据安全性的同时，验证数据的有效性。

第 3 章 系统设计

3.1 功能模块设计

根据功能需求分析可知，仓库管理系统可以划分为出入库、库存、权限管理、数据维护 4 个功能模块。

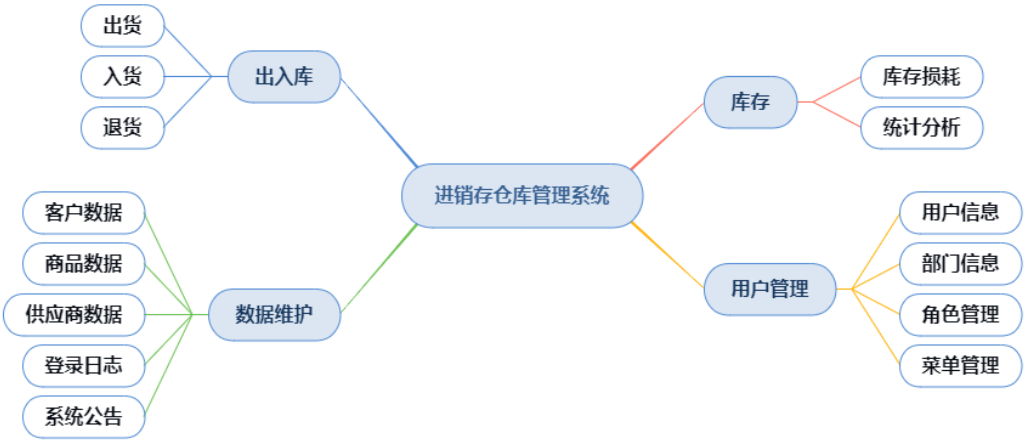

图 3.1 功能模块图

3.1.1 出入库管理设计

1）功能结构设计

“出入库”为出入库管理员提供物品入库和出库功能；此外还负责对物品进行退货操作。主要涉及出货，入货，退货 3 个子模块。

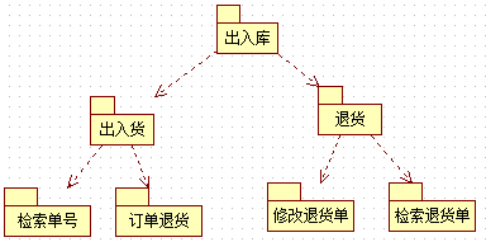

图 3.2 出入库包图

2）类图设计

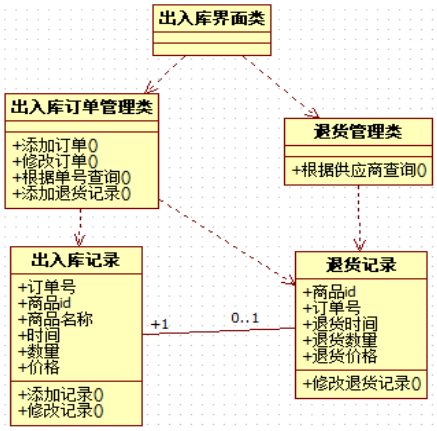

图 3.3 出入库类图

出入库类图中，一共有 5 个类，其中：出入库界面类负责响应页面的各类请求，它的执行依赖于出入库订单管理类和退货管理类；出入库订单管理类作为主功能类；退货管理类负责根据供应商查询退货记录；出入库记录类主要负责根据单号记录出入库信息，依赖于出入库订单管理类；退货记录类主要负责根据出入库单号记录退货信息，依赖于出入库订单管理类。

3）顺序图设计

出入库添加搜索功能的顺序图如图 3.4 所示。

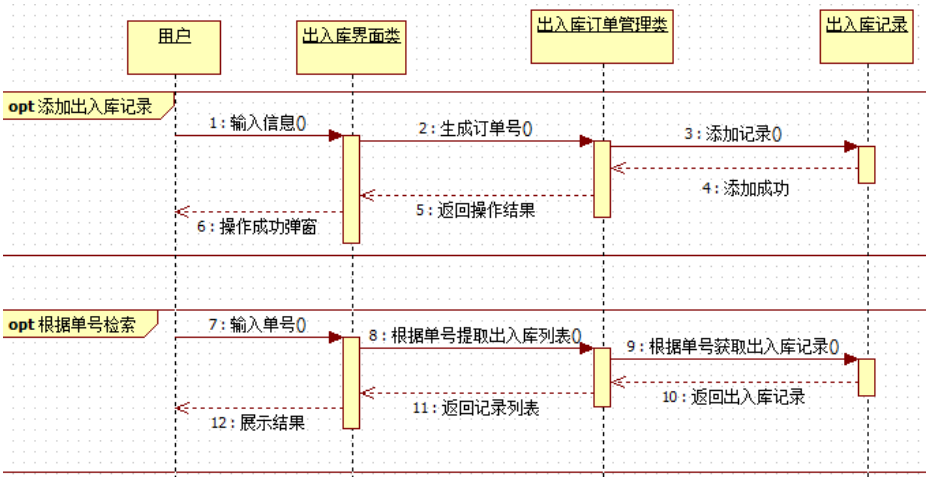

图 3.4 出入库管理顺序图

添加退货记录功能的顺序图如图 3.5 所示。

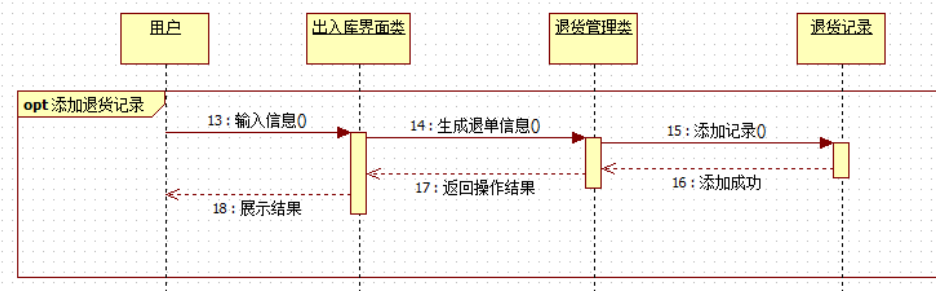

图 3.5 添加退货记录顺序图

4）核心处理流程设计

针对上述顺序图，图 3.4 中“生成订单号”还可以进一步细化。

由出入库管理员对入库单进行添加或修改操作。添加时系统将生成全局唯一的订单号，一个订单号可以对应多条商品入库记录。

用户的出入库操作依托订单号进行唯一标识。采用雪花算法实现订单号的生成。此算法生成出的订单号共计 64 位，首位作为符号位，41 位代表时间戳，最后的 22 为由 10 位工作机器 ID 和 12 位序列号共同组成。相较于传统的自增主键更不容易被爬取，有着更好的安全性。

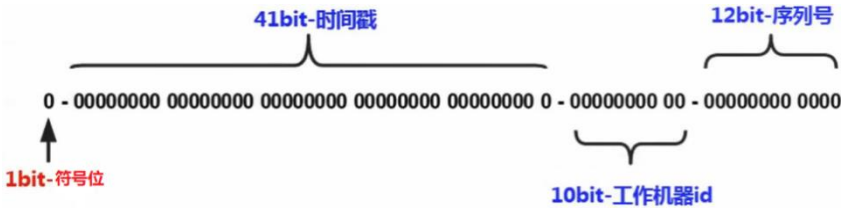

图 3.6 雪花算法图

此算法预留的 10 位的工作机器 id，利用了移位运算将 10 位 bit 由数据中心 id 和工作中心 id 共同占用，并使用额外的变量区分出数据中心和工作中心占用的位数，为后续扩容保留余地。若拓展了工作中心的数量时，只需要改变工作机器 id 和占用的位数，就能保证生成的订单号在分布式的环境下全局唯一。

由于一台机器上只需要一个订单号生成器的实例，以避免生成重复订单号，可以使用懒汉单例模式来对此算法进行封装。又因为 Jvm 的指令优化，在多线程环境下引发指令重排(初始化对象和分配内存地址的执行顺序颠倒)而导致对象重复创建。

另一种思路是采用双重检查锁，即重复检查实例是否被创建，只在创建实例代码块进行同步，避免了同步带来的性能问题，如图 3.7 所示。

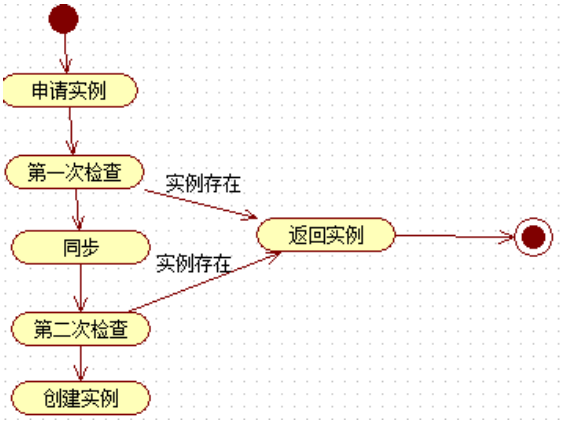

图 3.7 双重检查锁流程图

3.1.2 库存管理设计

1）功能结构设计

为库存管理员提供录入库存损耗和生成统计报表。

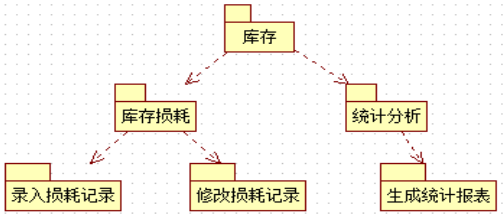

图 3.8 库存包图

2）类图设计

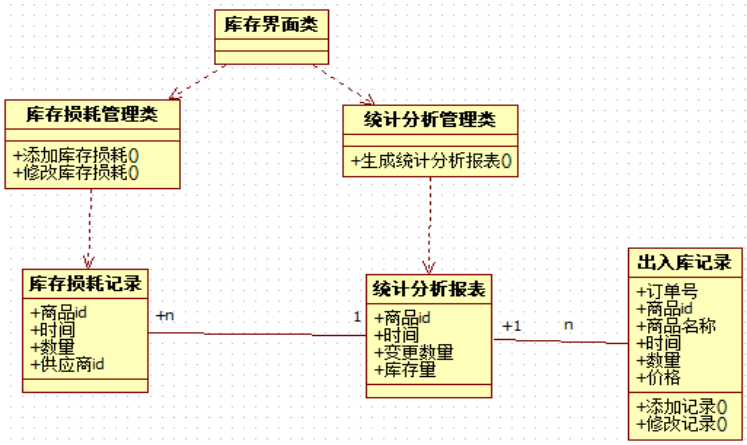

图 3.9 库存类图

库存类图中一共有 6 个类，其中：库存界面类主要负责响应页面的各类请求，它的执行依赖于库存损耗管理类和统计分析管理类；库存损耗管理类作为主功能类；统计分析管理类根据库存损耗和出入库记录生成统计报表。

3）顺序图设计

库存损耗记录添加功能的顺序图如图 3.10 所示。

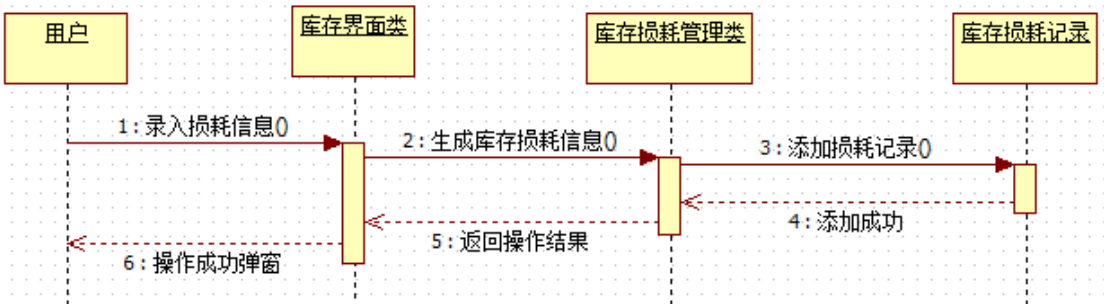

图 3.10 库存损耗记录添加顺序图

生成统计报表功能的顺序图如图 3.11 所示。

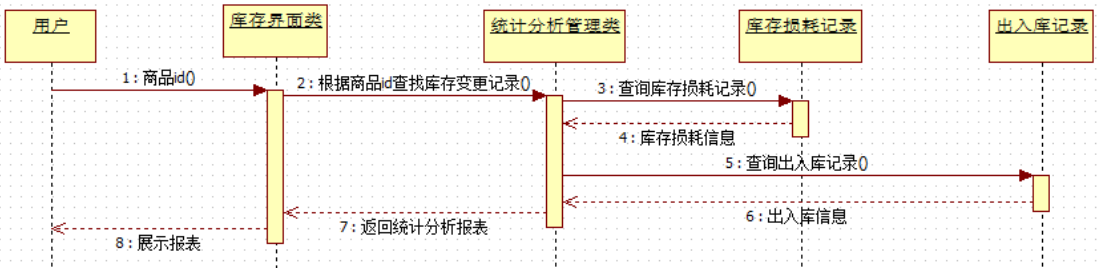

图 3.11 生成统计报表顺序图

4）核心处理流程设计

针对上述顺序图，图 3.10 中“添加损耗记录”方法还可以进一步细化，如图 3.12 所示。

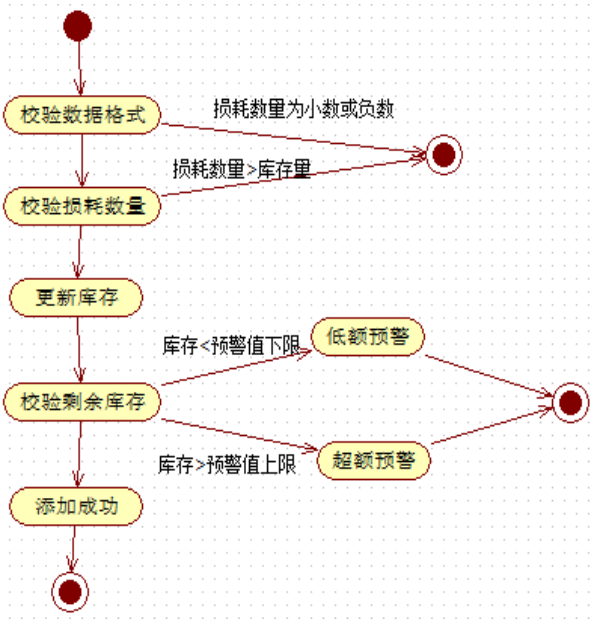

图 3.12 添加损耗记录活动图

在持有库存管理权限的情况下，打开库存管理页面，点击添加按钮，弹出库存损耗添加弹窗。

库存损耗数量只能输入正整数，在数字格式非法时，将弹框对用户进行提示。当数据格式正确时，校验当前库存量是否小于损耗量，否则提示用户”损耗数量不能大于库存量”，在校验成功后更新库存。更新后的库存量在高于预警值上限或低于预警值下限的情况下，将触发库存预警，否则提示添加成功。

3.1.3 权限管理设计

1）功能结构设计

为系统管理员提供系统数据维护和用户权限管理功能。

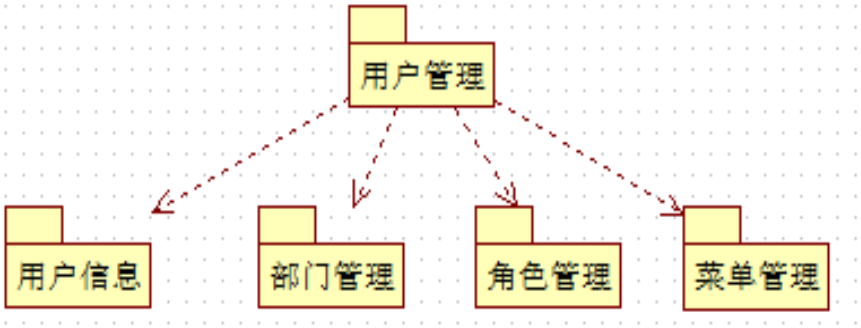

图 3.13 权限管理包图

2）类图设计

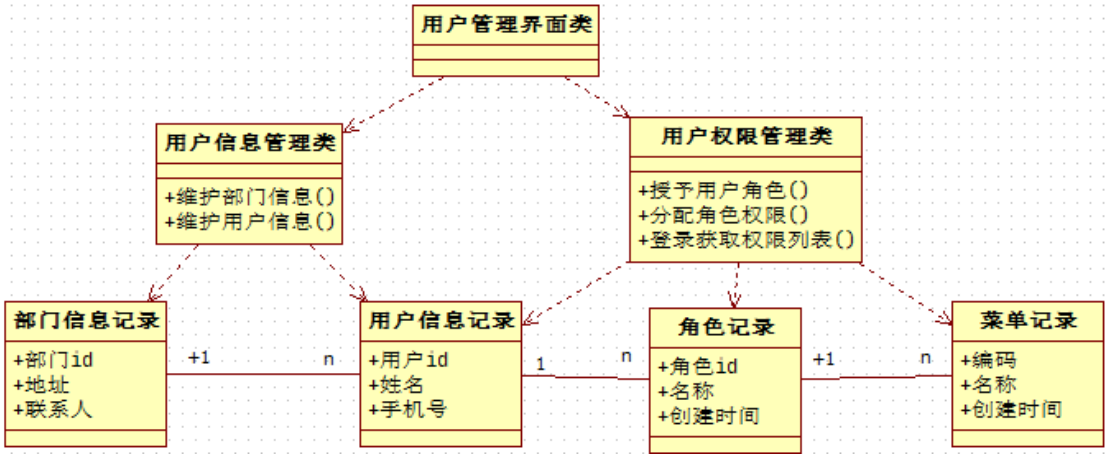

图 3.14 权限管理类图

在权限管理类图中：权限管理界面类主要负责响应页面的各类请求，它的执行依赖于用户信息和用户权限管理类；用户信息管理类负责维护用户和部门信息；用户权限管理类负责授予角色和分配权限；部门和用户记录，用户和角色记录，菜单和角色记录都存在一对多的关系。

3）顺序图设计

授予用户角色，分配角色权限，维护菜单记录功能的顺序图如图 3.15 所示。

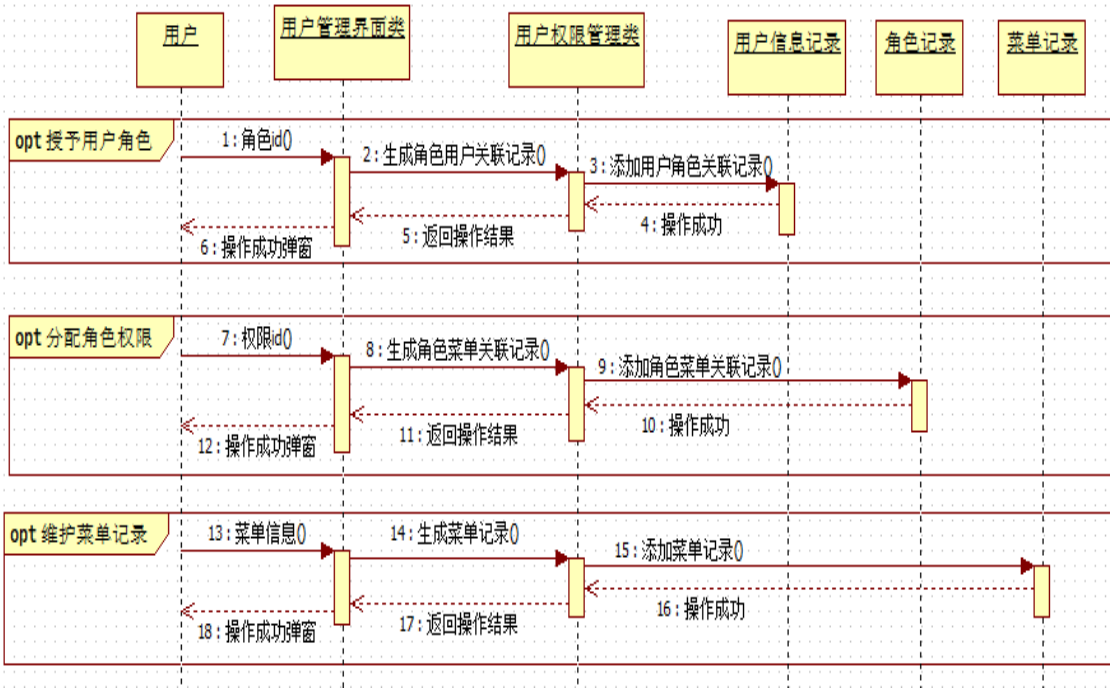

图 3.15 权限管理顺序图

登录获取权限列表的顺序图如图 3.16 所示。

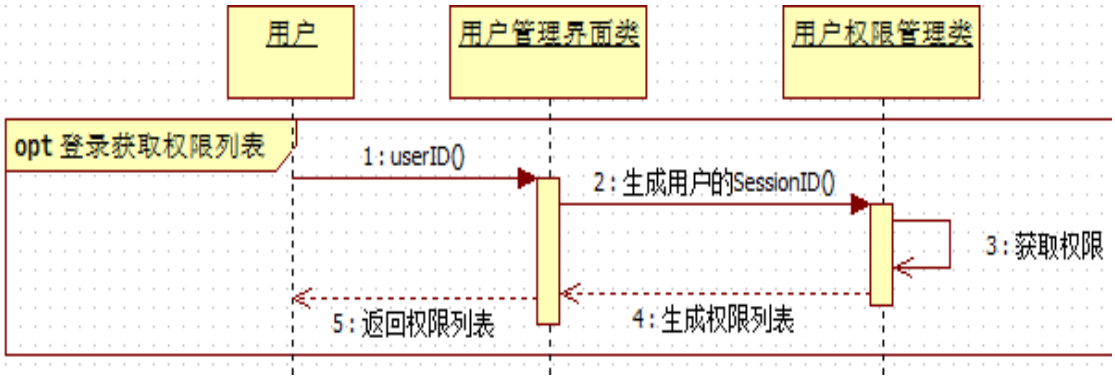

图 3.16 登录获取权限列表顺序图

4）核心处理流程设计

针对上述顺序图，图 3.17 中“获取权限”方法还可以进一步细化：在获取用户拥有的权限过程中，首先根据用户 ID 查询出用户被授予的所有角色；再检索出角色下的所有权限；然后将权限列表与用户的 SessionID 一同保存至 Session，用于后续的权限验证。

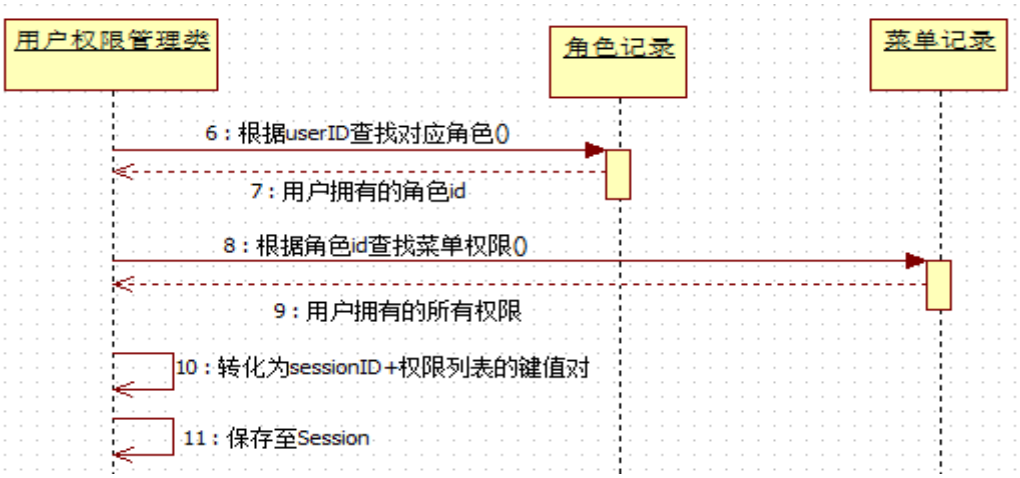

图 3.17 获取权限活动图

3.1.4 数据维护设计

1）功能结构设计

数据维护功能主要是为系统的正常运行提供数据支持。

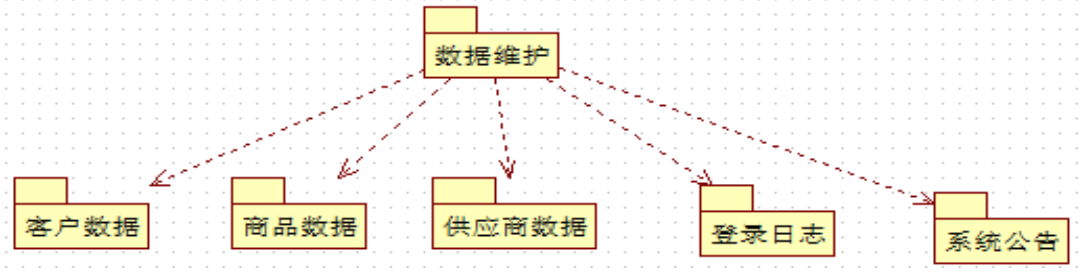

图 3.18 数据维护包图

2）类图设计

在数据维护类图中：数据维护界面类负责响应页面的各类请求，它的执行依赖于业务数据管理类和系统数据管理类；业务数据管理类负责维护系统的业务数据，包括客户，商品和供应商数据；系统数据管理类负责维护系统登录日志和公告数据。

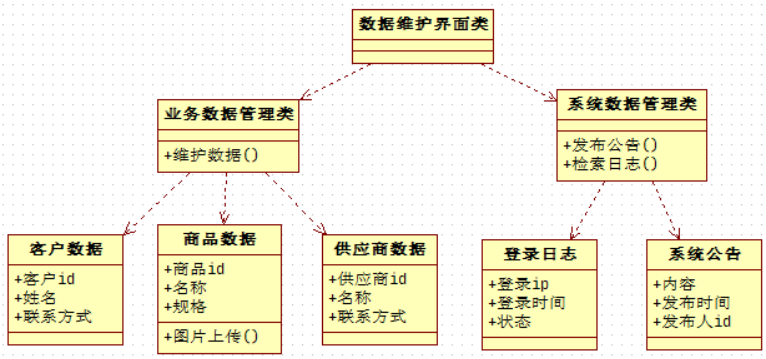

图 3.20 数据维护类图

3）顺序图设计

商品数据维护的顺序图如图 3.21 所示。

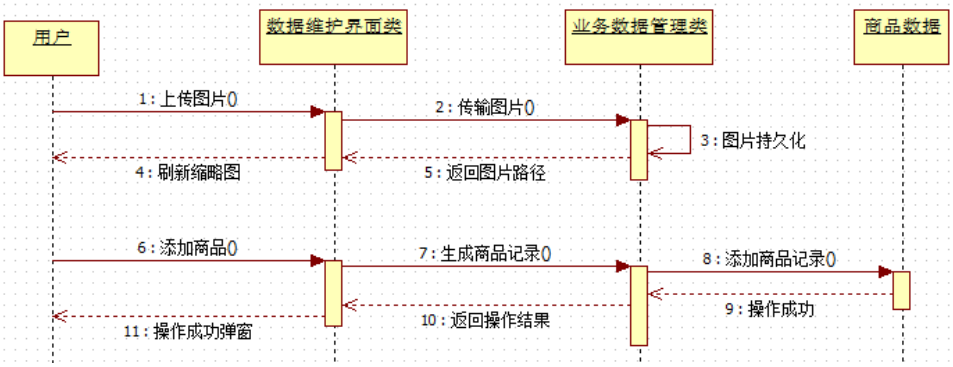

图 3.21 商品维护顺序图

4）核心处理流程设计

针对上述顺序图，图 3.22 中“添加商品”还可以细化，如图 3.22 所示。

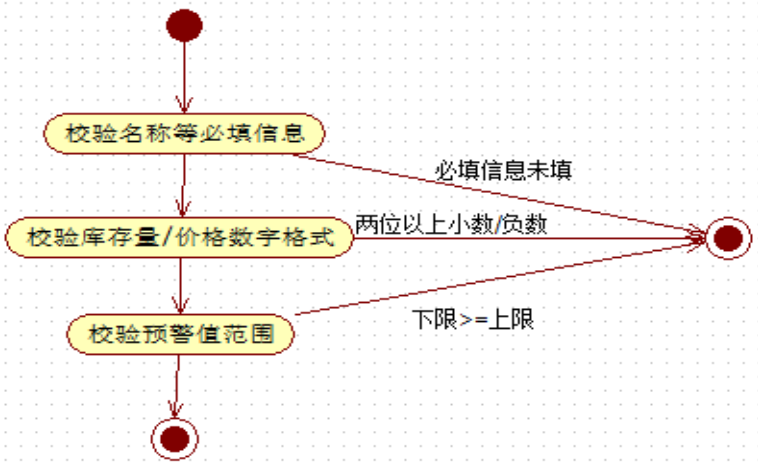

图 3.22 添加商品活动图

商品的名称/供应商等信息为必填项，任意一项未填写将无法完成添加操作。库存量，库存预警值限制为正整数，商品价格只能输入正数且最大 2 位小数，在

数字格式非法时，将弹框对用户进行提示。商品预警值分为上限和下限，其中上限值必须大于上限。

商品添加成功后，录入商品记录至数据库，关闭弹窗并刷新商品列表。

3.2 数据库设计

3.2.1 逻辑结构设计

将 ER 图中数据概念模型转化为逻辑结构，首先将实体直接映射为逻辑表。

商品实体转化为商品关系(商品编号，库存数量)；客户实体转化为客户关系(客户编号，联系方式)；公告实体和发布联系转化为公告关系(公告编号，发布人用户编号)；供应商实体转化为供应商关系(供应商编号，联系方式)；权限实体转化为权限关系(权限编号，父级权限编号)；角色实体转化为角色关系(角色编号，角色名称)；用户实体和隶属联系转化为用户部门关系(用户编号，部门编号)。部门实体转化为部门关系(部门编号，父级部门编号)；

随后将实体间的联系转化为表之间的逻辑联系。

进货联系转化为进货关系(进货编号，商品编号，操作人编号)和进货退单关系(进货编号，商品编号，操作人编号，退货数量)

损耗联系转化为损耗关系(损耗编号，商品编号，操作人编号)

出货联系转化为出货关系(出货编号，商品编号，操作人编号)和出货退单关系(出货编号，商品编号，退货数量)

授予联系转化为角色与权限关系(角色编号，权限编号)和角色与用户关系(用户编号，角色编号)

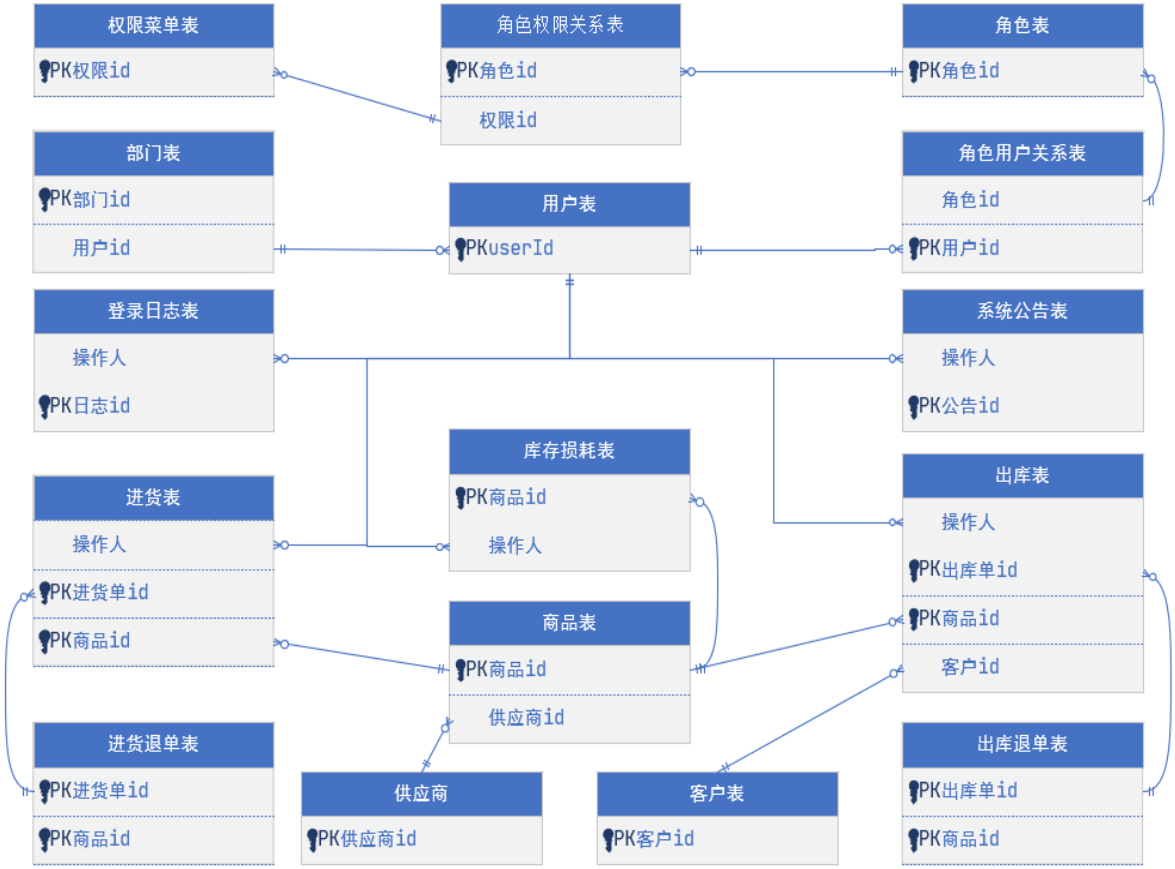

图 3.23 表逻辑结构图

3.2.2 表结构设计

表 3.1 部门表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	部门编号
pid	Int(11)	否		父级部门
name	Char(6)			名称

表 3.2 用户表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	
name	varchar(11)	否		用户名
sex	Varchar(255)			性别
deptid	Int(11)	否		所属部门

表 3.3 权限菜单表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	编号
pid	Int(11)	否		父级编号
type	varchar(6)	否		类型
typecode	Varchar(255)	否		权限编码

表 3.4 角色表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	编号
name	Varchar(255)	否		名称

表 3.5 角色权限关系表

字段	类型	允许空	主/外键	描述
rid	Int(11)	否	主键	角色编号
pid	Int(11)	否	主键	权限 id

表 3-6 角色用户关系表

字段	类型	允许空	主/外键	描述
rid	Int(11)	否	主键	角色编号
uid	Int(11)	否	主键	用户 id

表 3.7 登陆日志表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	编号
loginname	Varchar(32)	否		登陆信息
loginip	Varchar(32)	否		登陆 IP
logintime	datetime	否		登陆时间

表 3.8 系统公告表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	

title	Varchar(255)	否		标题
content	Text(255)			内容

表 3.9 客户表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	
customername	Varchar(255)	否		名称
address	datetime	否		地址
connectionperson	Int(11)	否		联系人

表 3.10 供应商表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	
providername	Varchar(255)	否		名称
connectionperson	Int(11)	否		联系人
phone	Varchar(255)			电话

表 3.11 商品表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	
goodsname	Varchar(255)	否		商品名称
providerid	Varchar(255)	否	外键	供应商编号
number	Int(11)	否		库存数量
goodsimg	Varchar(255)			商品图片
upperlimit	Int(11)	否		预警上限
lowerlimit	Int(11)	否		预警下限

表 3.12 进货表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	进货单
providerid	Varchar(255)	否		供应商
number	Int(11)	否		数量

inportprice	Varchar(255)			价格
goodsid	Int(11)	否	主键	商品编号

表 3.13 进货退单表

字段	类型	允许空	主/外键	描述
id	Int(11)	否	主键	进货单
providerid	Varchar(255)	否		供应商
number	Int(11)	否		数量
goodsid	Int(11)	否	主键	商品编号

表 3.14 出库表

字段	类型	允许空	主/外键	描述
id	Bigint(20)	否	主键	出库单号
customerid	Int(10)	否	外键	客户编号
number	Int(11)	否		数量
saleprice	double	否		销售价格
goodsid	Int(11)		主键	商品编号

表 3.15 出库退单表

字段	类型	允许空	主/外键	描述
id	Bigint(20)	否	主键	出库单号
customerid	Int(10)	否	外键	客户编号
number	Int(11)	否		数量
goodsid	Int(10)	否	主键	商品编号

表 3.16 库存损耗表

字段	类型	允许空	主/外键	描述
id	Bigint(20)	否	主键	商品 id
losstime	Int(10)	否	外键	时间
number	Double	否		数量
providerid	Varchar(255)	否	外键	供应商

第 4 章 模块实现

4.1 系统软件架构

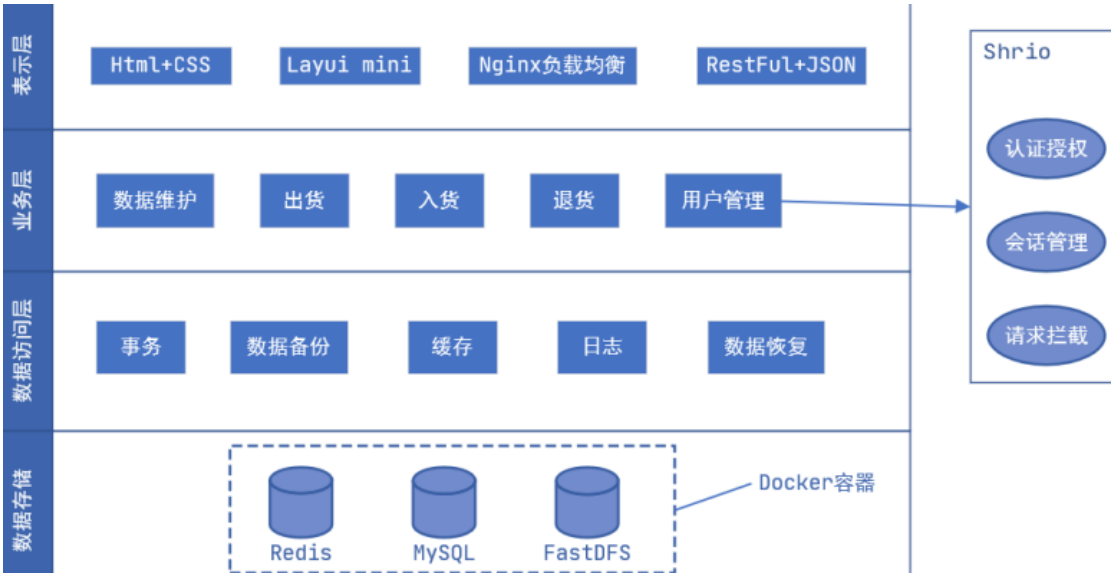

图 4.1 架构设计图

本系统基于 Web 的分层设计思想进行实施开发，可以自顶向下地将系统架构拆分为：

1)表示层:负责与用户交互，界面渲染和后台数据的展示。

2)业务层:与仓储业务密切相关，所有数据在此进行加工处理，并封装处理结果返回给调用方。

3)数据访问层:向上提供访问数据库的接口，支持业务层的数据处理。

4)数据存储层:针对数据类型区分出持久化需求，在磁盘或内存中选择数据的存储位置。

本系统的分层架构为系统的维护带来了便利：首先，层与层之间的交互和影响仅限于相邻层，符合迪米特法则。其次，相邻层通过接口进行交互，接口可以被不同形式地实现，具有开放性。最后，设计者可以将复杂系统依据设计需求，将步骤逐层分解为若干简单的步骤，从而降低系统开发的难度。

4.2 权限管理(RBAC)实现

4.2.1 设计思想

Role-Based Access Control 是目前主流的权限管理设计理念，Role-Based 表示了它以”角色”作为中心。具体地来说：权限、角色、用户三者之间存在层级的依赖关系，先将权限赋予给角色，再把角色分配给用户，简化授权流程。

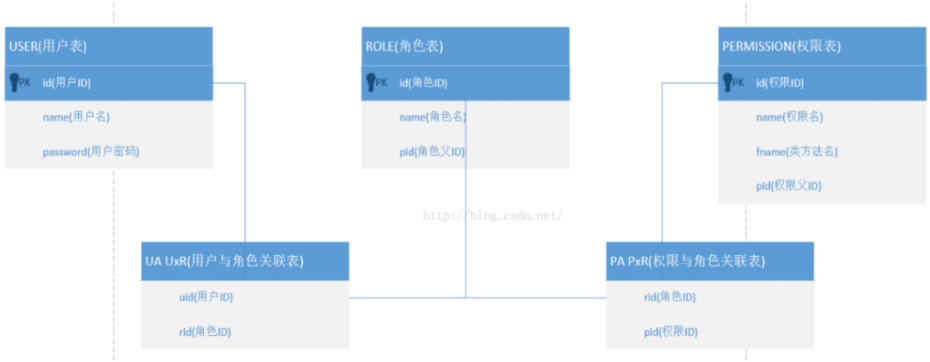

图 4.2 RBAC 设计思想

首先，用户的所有操作都需要先与 Shiro 的 Subject(代理主体)进行交互，由 Subject 委托给内部的 SecurityManager(安全管理器)进行授权操作。

其次，SecurityManager 将访问 Realm(领域)，以获取用户的权限数据。Realm 被链式地存储在 Realm Chain 中，SecurityManager 对 Realm 的轮循访问将采取短路机制，一旦有 Realm 接受处理并返回授权结果，就不再访问剩余的 Realm。

随后，SecurityManager 接收并处理授权结果，决定用户是否有权限进行此操作，并将结果返回给 Subject。

最后，Subject 接收 SecurityManager 返回的处理结果，若允许则用户操作继续执行，若拒绝则用户操作结束，提示用户权限不足。

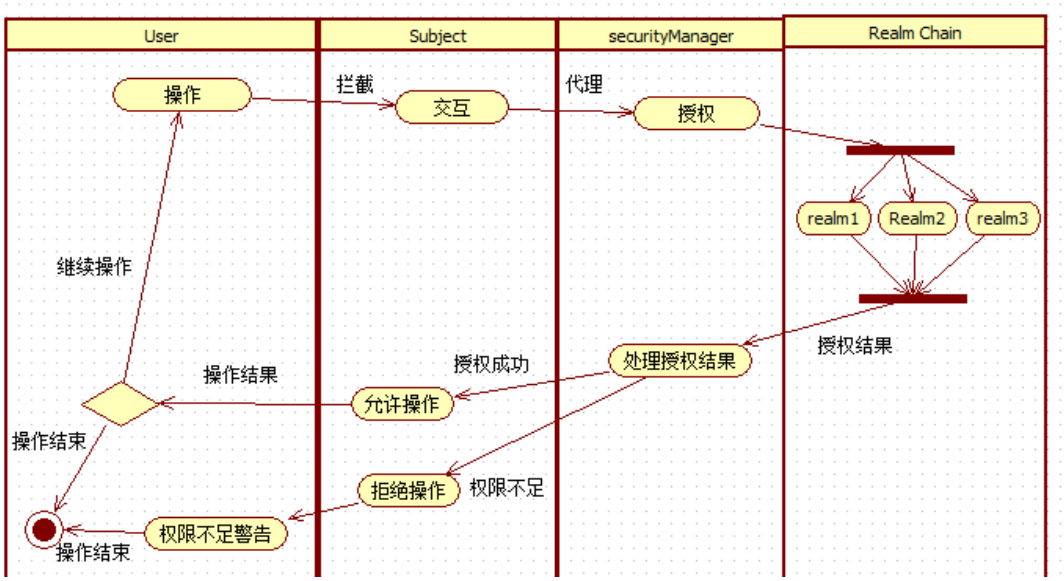

图 4.3 限制用户权限操作流程图

4.2.2 操作界面

图 4.4 用户分配角色界面

图 4.2.3 用户分配角色的界面由可复选的列表网格和确认/关闭按钮组成，列表中包含勾选框以及可选的各个角色.选择角色点击确认按钮后完成分配。

图 4.5 分配权限界面

分配权限界面由可复选的权限树形网格以及确认/关闭按钮组成。点击左侧的”+-”按钮可以收起或展开树形权限.选择权限并点击确认按钮后完成分配。

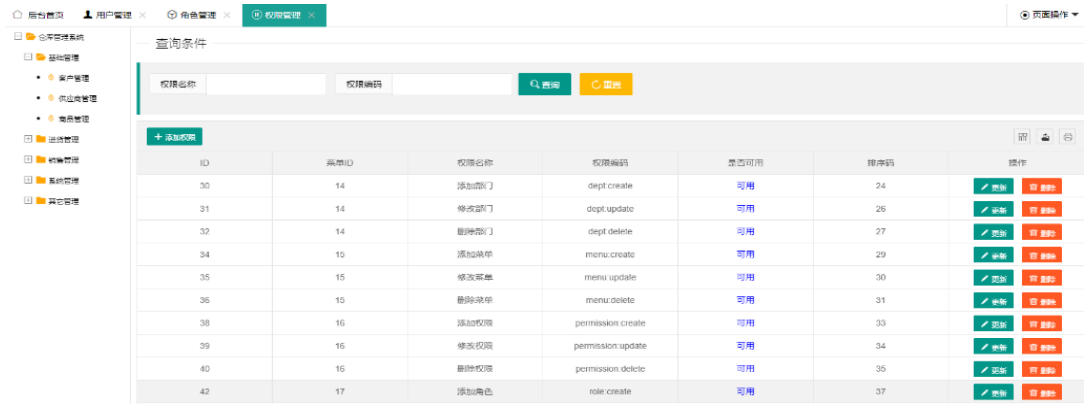

图 4.6 权限管理页面

权限管理页面由左侧的树形菜单导航栏，右侧顶部的搜索区以及右侧下方

的权限网格组成。点击左侧的导航栏可以定位到该菜单下的所有权限。

4.2.3 核心代码

1)Shrio 配置。包含了加密算法，放行路径和拦截路径。

```
hash-algorithm-name：MD5 //指定 MD5 的加密算法
hash-iterations：2              //散列 2 次
anon-urls： -/index.html*     //放行路径
authc-ulrs：-/**              //拦截路径
```

2)Realm 授权主要代码：

```
authorizationInfo.addStringPermissions(permissions)；
```

3)Service 层主要代码：

```
@Autowired     //使用注解方式注入 permissionMapper
private PermissionMapper permissionMapper；
public List<Permission> findList(Map<String，Object> queryMap){
    return permissionMapper.findList(queryMap)；
}
```

4)View 层（以 permissionManager.html 为例）核心代码：

```
<button shiro：hasPermission="update">
    <span>更新</span>
</button>
```

4.2.4 测试

以权限搜素功能为例，采用黑盒测试中的等价类划分方法，黑盒测试的优点在于容易实施，不需要关注内部的实施。是一种更接近于用户使用角度的测试方式。

表 4.1 权限模块之权限搜索测试用例

模块	权限搜索
前置条件	被授予权限管理权限
测试方法	黑盒测试：等价类划分
测试目的	验证权限搜索的结果

编号	步骤	参数值	期望结果
Per-1	手动输入待查找的检索条件，然后点击搜索按钮	检索条件=“添加”	1.搜索结果根据 ID 降序显示 2.显示“添加”相关信息 3.右下角显示搜索结果的总条数 4.超出指定条数的信息分页显示
Per-2	检索条件为空，直接点击搜索按钮	检索条件=“”	1.搜索结果根据 ID 降序显示 3.右下角显示搜索结果的总条数 4.超出指定条数的信息分页显示
Per—3	手动输入待查找的检索条件，然后点击”搜索”按钮	检索条件=“asfsegdzbtd”	1.结果为空，页面不显示数据 2.右下角显示搜索结果条数为 0
Per—4	搜索结果的第一页点击”上一页”按钮	/	1.”上一页”按钮无法点击
Per—5	在搜索结果的第二页点击”上一页”按钮	/	1.下方页数跳转到第 1 页 2.数据表格显示为第 1 页信息

4.3 图片上传访问实现

4.3.1 设计思想

在持有商品管理权限的情况下，打开商品管理页面。商品图片的上传是可选项，存在默认图片；在上传时会进行格式校验，并在成功后刷新缩略图。

采用 fastdfs 分布式文件系统，由 Tracker(注册中心)和 Storage(存储中心)共同实现图片管理功能。

Storage 以”心跳”的形式向 Tracker 节点进行注册，Storage 定时向 Tracker 节点发送携带了状态，自身的 IP 等必要信息的报文，来证明自身的可用状态，进而被 Tracker 注册为活跃节点。

当 Client(客户端)进行上传前，先向请求 Tracker 可用的 Storage，Tracker 根据各个 Storage 提供服务的响应速度，网络拥塞情况等数据，择优返回给 Client 一个 IP 地址。

最后，Client 向 Storage 发起上传请求，Storage 将文件存入磁盘，并返回文件的访问路径。

而在 Client 根据访问路径向 Nginx 发起商品图片访问的请求过程中，由 Nginx 监听本次请求，首先访问本地缓存中是否存在此图片，若不存在，则通过 IP-Hash 算法，负载均衡地访问 Storage 集群中的其中一个节点，并将图片内容返回给 Client。

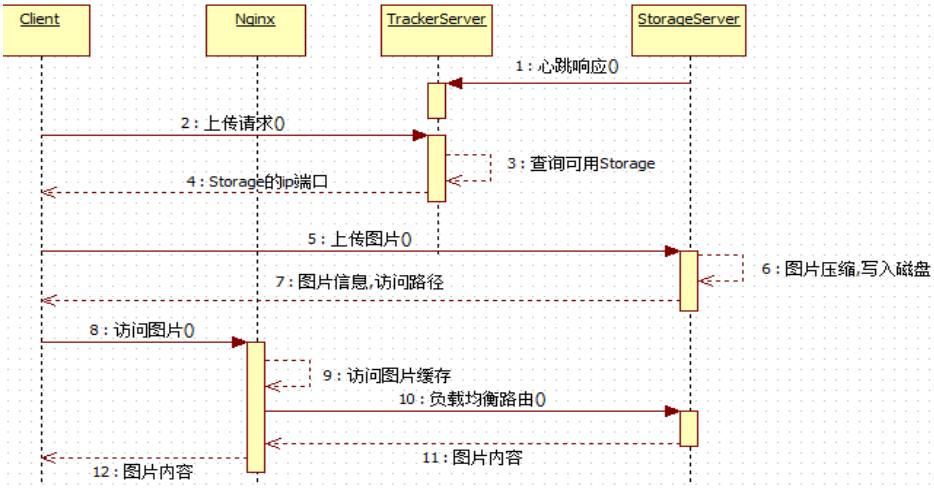

图 4.7 图片上传访问流程图

4.3.2 操作界面

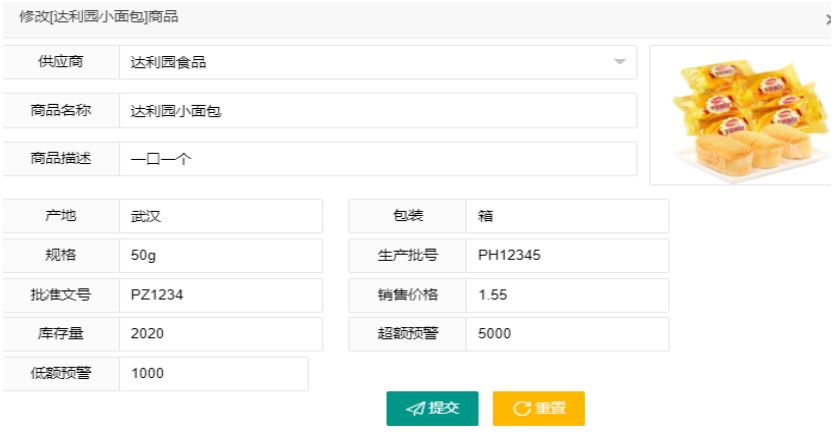

图 4.8 商品修改界面

商品信息修改界面由供应商，商品名称等输入框以及右上角的商品图片框组成。点击商品图片框跳出上传图片弹窗，可以更换商品的图片。

4.3.3 核心代码

1)FastDFS 配置：

so-timeout：2500　#读取时间

connect-timeout：600　#连接超时时间

tracker-list：-****　#tracker 服务配置地址列表

allow-types：-image/jpeg-image/png-image/gif　　//允许的文件类型

2)图片上传 Service 主要代码：

Stringexte=StringUtils.substringAfterLast(file.getOriginalFilename()，".")；

StorePath storePath = storageClient.uploadFile(file.getInputStream()，file.getSize()，ext，null)；

3)storage 节点配置

port=23000 #端口

store_path0=/data/fdfs/store_path #持久化路径配置

log_level=info　#输出日志级别

4)Nginx 配置的主要代码：

listen 80；#http 默认端口

server_name localhost；

root lx_storehouse；

……

location /group1/M00{

　　alias /root/fastdfs/storage/data；//文件持久化路径

4.3.4 测试

采用场景法，通过观察动作的发生与结果，从而发现需求中存在的问题。

表 4.2 图片上传测试用例

模块	图片上传		
前置条件	被授予添加商品权限		
测试方法	场景法		
测试目的	验证图片上传的路径，文件名，格式限制		
编号	步骤	参数值	期望结果
Goods—1	添加图片成功	上传 1.jpg 文件	1.弹框提示图片上传成功 2.缩略图刷新为上传的图片 3.图片存储地址与配置文件一致 4.图片被重命名为随机字符串+文件后缀

Goods—2	未上传商品图片且添加成功	/	1.商品图片为默认图片 2.未进行图片持久化操作
Goods—3	上传错误格式图片(除)	上传 1.zip 文件	1.图片上传失败 2.错误弹框”格式不支持”
Goods—4	打开上传至服务器后的图片	/	1.图片的分辨率与原图一致 2.上传的图片被重命名为随机字符串+文件后缀
Goods—5	上传正在打开的图片	/	1.图片上传成功
Goods—6	上传文件名带有中文的图片	文件名：图片.jpg	1.图片上传成功 2.上传的图片被重命名为随机字符串+文件后缀
Goods—7	在已上传成功一张图片的情况下，继续上传图片	文件名：图片.jpg	1.图片上传成功 2.原图片的缩略图被替换为新上传的图片 3.点击保存按钮时保存的是新上传的图片地址

4.4 入库实现

4.4.1 设计思想

首先在 Domain 层定义实体的各个属性，以及提供必要的 get/set 方法修改属性，在 Mapper 层中，MybatisPlus 提供了通用增删改查接口 BaseMapper，本系统自定义的数据库访问接口都继承了 BaseMapper，从而获得了父类的方法，如根据主键的查询等接口，并根据业务需求自定义了部分特殊接口.随后在 Service 层封装业务处理接口，紧接着用 Service 的实现类去实现接口，并在实现的方法中调用相应的 Dao 层持久化接口。在 Controller 层根据业务需求引用相应的 Service 层方法，来处理用户请求。

4.4.2 操作界面

图 4.9 入库记录列表

入库记录列表的左上角为添加按钮和供应商选择下拉框。选择供应商后方可添加入库记录。还能够在操作栏中对入库记录进行修改以及退货操作。

图 4.10 添加进货界面

添加进货界面由商品选择下拉框，商品数量，价格和备注输入框组成。

4.4.3 核心代码

1)入库 Service 主要代码：

```
inport.setId(SnowflakeIdWorker.getInstance().nextId())；//生成订单号
this.goodsService.updateGoods(goods)；//更新库存
```

2)单号生成主要代码：

```
private volatile static SnowflakeIdWorker snow = null；//分配静态区内存空间
//b.私有构造方法，禁止外部创建实例，参数为工作中心 id，数据中心 id
private SnowflakeIdWorker(long workerId，long datacenterId){
public static SnowflakeIdWorker getInstance(){//c.公开获取实例的方法
if(snow==null){//一次检查，实例被创建后不进入方法体
    synchronized(SnowflakeIdWorker.class){//e.同步
        if(snow==null){//二次检查
            snow=new SnowflakeIdWorker(workerId，datacenterId)；
}
```

3)执行 MybaitsPlus 的逆向工程插件：

根据 po 类对应的页面业务逻辑，自定义业务层数据传输的 vo 类。

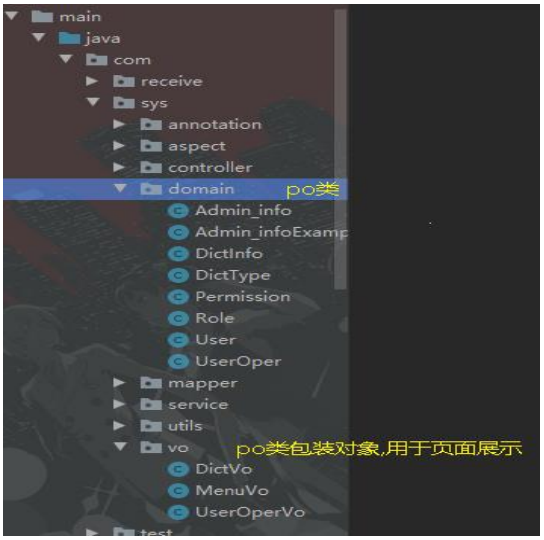

图 4.11 po 类的结构：

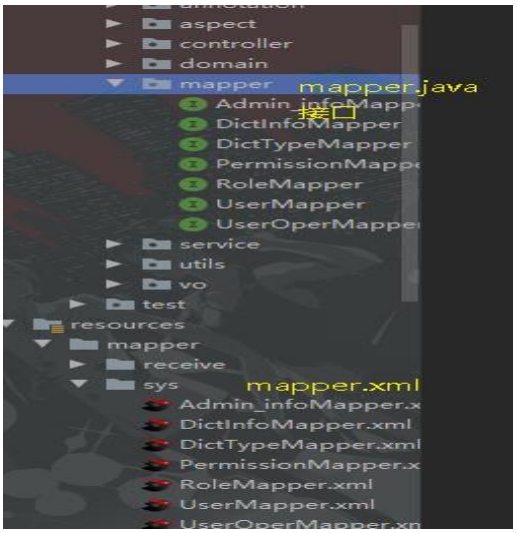

图 4.12 mapper 配置的结构：

4)定义添加入库的 mapper 接口：

int batchInsert(@Param("list") List<Inport> list)；

5)在 InportMapper.xml 中加入 sql：

SELECT providername FROM bus_provider AS provider

6)MybatisPlus 配置，指定扫描包位置

-classpath：mapper/*Mapper.xml

4.4.4 测试

采用黑盒测试中的边界值分析方法来测试添加入库，用 Junit 框架进行单元测试。边界值分析侧重于输入输出范围的边界，这往往是错误的发生点，通过边界值能够以最小成本生成测试数据，从而轻松地对接口进行自动化单元测试，这项技术目前被敏捷开发所广泛采用。

表 4.3 权限搜索测试用例

模块	入库
前置条件	被授予入库权限
测试方法	黑盒测试：边界值分析、Junit 单元测试
测试目的	验证入库时的数据校验，订单号随机生成

编号	步骤	参数值	期望结果
Inport—1	添加入库记录，不填写入库价格	入库价格=null	1.入库失败 2.弹框提示入库价格为必填项
Inport—2	添加入库记录，填入非法入库价格	入库价格=-100.01	1.非法输入被清空
Inport—3	添加入库记录，填入正确的入库价格	入库价格=100.01	1.入库成功 2.商品库存被更新
Inport—4	Junit 测试模拟生成百万入库订单场景	/	1.生成的订单号唯一 2.订单号不规律地递增

以下为 Junit 单元测试代码

```
CountDownLatch cdl = new CountDownLatch(testNum)；
while (testNum-- > 0){//循环 100w 次
    pool.submit(()->{//订单号重复抛出断言异常
    synchronized(set){assert set.add(idWorker.nextId())；
    cdl.countDown()；//自减为 0 时唤醒主线程
})；
cdl.await()；//阻塞直至被唤醒
System.out.println(set.size())；//此处应为 100w
```

4.5 库存预警实现

4.5.1 设计思想

引起库存变动的用例有：添加商品、入库、入库退货、出库、出库退货、库存损耗等。当上述用例引发的库存变动导致库存量高于预警值上限或低于预警值下限时将触发库存预警。

可见库存预警是多个用例的公共行为，这将造成库存预警代码冗余。根据面向切面编程的思想，可以将库存预警视为一个“Aspect”，即切面。切面就是与业务无关，却被多个模块所公共调用的逻辑。而引起库存变动的用例可以视为“Pointcut”，即切入点。切入点可以理解为多个与切面交织的业务集合。

综上所述，将库存预警抽离为单独业务，封装成库存预警切面，织入会引发库存变动的切入点。使得每次的库存变动，都将自动地判断是否需要进行库存预

警，从而复用了库存预警模块，降低库存预警和库存更新业务的耦合性，也有利于后续的可维护性和拓展性。

4.5.2 操作界面

图 4.13 退货列表

退货列表由 ID，供应商，商品名称，规格，进货数量，进货价格，备注和操作栏 7 列组成，点击修改按钮修改退货信息。

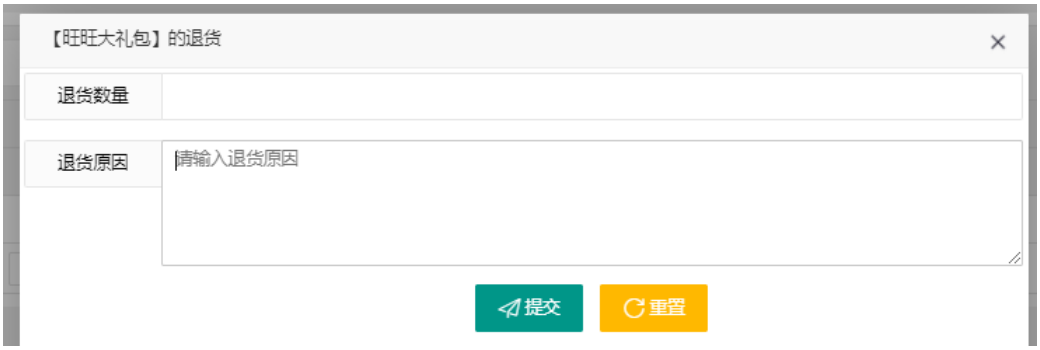

图 4.14 修改退货界面

修改退货界面由退货数量和退货原因输入框组成。这两个输入框都为必填项。点击提交后修改退货信息成功。

4.5.3 核心代码

1)Redis 配置

host：***.***.***.***//IP 地址

port：****//端口号

max-idle：20//最大空闲数

max-active：100//最大活跃连接数

2)正则表达式指定切入点：

```
String POINTCUT ="execution(*impl.*.updateGoods(..))";
```

3)Redis 序列化规则配置

```
redisTemplate.setKeySerializer(new StringRedisSerializer());
redisTemplate.setValueSerializer(new GenericJackson2JsonRedisSerializer());
```

4)切入方法

```
@Around(POINTCUT_GOODS_UPDATE)    //指定切入点
public Object cacheInportUpdate(ProceedingJoinPoint joinPoint){
……
}
```

4.5.4 测试

采用白盒测试中的路径覆盖法来测试库存预警，相较于黑盒测试，白盒测试能够更深入地发现代码中隐藏的问题。而路径覆盖是一种比较彻底的测试方式，可以对代码中所有可能的路径进行测试。

首先分析库存预警的代码(以伪代码形式展示)。

```
if(累计退货总额>进货数量)        //P1
    then 退货失败                //路径 A：
    else 插入退货记录            //路径 B
更新商品库存                     //路径 C
if(库存>预警上限)                //P2
    then 库存超额预警            //路径 D：
    elseif(库存<预警下限)        //P3：
        then 库存低额预警        //路径 E：
    else 取消库存预警            //路径 F：
```

通过控制流图来描述上述的程序控制流，如图 4.15 所示

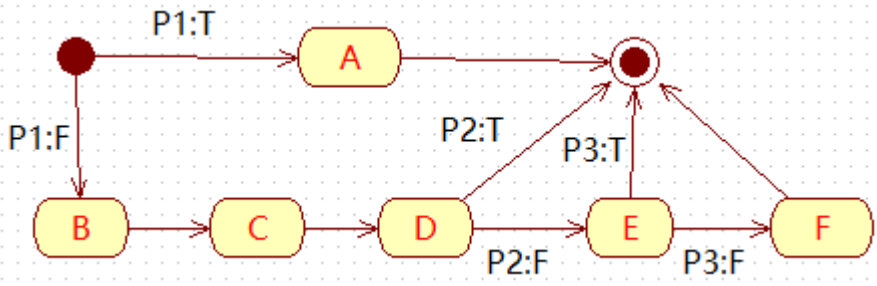

图 4.15 库存预警控制流图

图中 A、B、C、D、E、F 分别表示路径；条件判断按出现顺序定义为 P1 到 P3，一共有 3 个条件，记 P1：累计退货总额>进货数量、P2：库存>预警上限、P3：库存<预警下限。

圈复杂度能够度量程序逻辑复杂性，假定流图的圈复杂度为 V(G)，则 V(G)=E-N+2，其中 E 是流图中边的数量，N 是流图中结点的数量；得出

V(G)=10-8+2=4。

根据圈复杂度可得出 4 条独立路径，每条独立路径之间至少存在一条新的处理语句或新的条件判断。最后得到的路径为:路径 1：A；路径 2：BCD；路径 3：BCDE；路径 4：BCDEF

最后根据独立路径导出的相关测试用例如表 4.4 所示。

表 4.4 库存预警模块测试用例

模块	库存预警				
前置条件	无				
测试方法	白盒测试：路径覆盖				
测试目的	验证库存预警能否正确触发				
编号	参数值	P1	P2	P3	路径
Warning-1	当前库存 1000 退货数量=1001 预警值上限 1100 预警值下限 900	T	F	F	A
Warning—2	当前库存 1000 入货数量=101 预警值上限 1100 预警值下限 900	F	T	F	BCD
Warning—3	当前库存 1000 退货数量=101 预警值上限 1100 预警值下限 900	F	F	T	BCDE
Warning—4	当前库存 1000 退货数量=50 预警值上限 1100 预警值下限 900	F	F	F	BCDEF

第 5 章 总结和展望

5.1 总结

本次毕业主要完成了以下工作：在课题研究的基础上分析了系统需求，并划分功能模块进行详细设计。在编码环节使用了 SpringBoot，Shiro 等框架进行开发设计。最终开发出具有出库，入库，数据维护，库存损耗，库存统计，权限管理等功能的进销存仓库管理系统。在系统实现过程中遵循软件生命周期，通过 UML 的用例图，类图，流程图和时序图来说明模块间的交互方式，并详细设计出实现方案。

这次的毕业设计让我更深入地了解软件工程这门专业，得以完整地体验了软件从需求到设计最后实现的过程，学习了很多的新技术与新知识。

5.2 展望

本系统并不局限于简单的实现，更重要的是为中小型企业提供仓储的高效率管理方案，但目前本系统完成的只是对单一的仓库进行的进销存业务，还存在很大的提升空间。

首先，在新零售模式的影响下，库存的分布遍布全球，这要求仓库之间的高流通性。企业往往存在多个仓库间的协调调度物资，这需要考虑到调度所耗费的运输和劳务成本，来决定最优的调度方案。

其次，可以将系统进行模块化地拆分为多个子系统，以提高最大负载，系统的发布还可以与 Docker 结合，利用容器镜像的特性减少发布和运维的工作，进一步地缩短迭代所需的时间。

另外，本次设计中 AI 的特性很少，仍然属于传统的互联网技术，然而目前人工智能日新月异的发展使得 AI 的普及率越来越高，在后续的迭代工作中可以考虑追加 AI 的特性，例如：智能质检等。

最后，新技术的不断发展会在系统效率和用户体验上为我们带来进一步地提升，对此也需要时刻保持关注。

参考文献

[1]张忠，宋嘉诚，黄隽瑶.基于 JavaEE 物品仓储管理系统设计[J].电脑编程技巧与维护，2020：69-71.

[2]马梓昂，贾克斌.基于 Web 的高性能智能快递柜管理系统[J].计算机应用与软件，2020：1-5.

[3]黄启启，项前，程茂上，吕志军.基于微服务的仓储管理与控制系统[J].东华大学学报：自然科学版，2020：83-90.

[4]Josh Bond. warehouse managment System system featured in facility modernization[J]. Modern Materials Handling，2016，71(7).

[5]孟琪，韩晓晶.敏捷测试在软件项目中的应用研究与实践[J].科技资讯，2020，18(13)：24-25.DOI：10.16661/j.cnki.1672-3791.2020.13.024

[6] 姜文，刘立康.应用软件项目的迭代开发与测试[J].计算机技术与发展，2019，29(04)：7-12.

[7]李楚贞，曾琳，余育文.华润万家超市进销存管理系统的设计与实现[J].计算机产品与流通，2020，(11)：274.

[8]陈华恩.Scrum 敏捷方法在湖南翼方软件项目管理中的应用研究[D].湖南大学，2014.

[9]SAP 的仓库管理系统增强与实现[D].王晓.东南大学，2018

[10]周吉祥，商玉林，赵昊炜，刘涛，田野.仓储管理系统风险控制研究与设计[J].科学技术创新，2019，(09)：165-166.

[11]贺晓娇.基于 SAP 的仓库管理系统研究[J].财经界，2020(30)：27-28.

[12]肖自乾，王弗雄，陈经优.基本路径测试方法之圈复杂度计算[J].软件导刊，2010，9(01):10-12.

[13] D.Naga Malleswari，K. Bhaskar，A. Monica，B. Venkat vinay，U. Sai Anirud Varma. Analysis of Risk in Information System using Cyclomatic Complexity[J]. International Journal of Recent Technology and Engineering (IJRTE)，2019，8(1).

[14]冯欣. 代码质量控制与复杂度测量在大型软件项目中的研究及应用[D].东北师范大学，2006.

致谢

四年时光匆匆而去，大学过后又是一段新征程的开始。

本文是在我的导师邓有莲老师的耐心指导下完成的。邓有莲老师从开题报告到论文定稿，一直在给我提供宝贵的意见和建议，并指点我对论文内容进行必要的修改，倾注了老师很多的心血，我这才能够顺利地完成论文撰写。邓有莲老师不仅经验丰富，而且教导有方，对学生严格要求。勤奋刻苦、脚踏实地的学习态度是我从邓有莲老师身上所学习的最大财富。再次感谢邓有莲老师对我的指导，能成为您的学生，我感到很自豪。

感谢参考文献中各位学者的文献，他们的研究成果的启发了我的论文创作。感谢同学们分享了给我的各类素材，为本文的需求分析阶段提供了很大的帮忙。

最后还要感谢评审本论文的专家教授们，感谢你们提出的宝贵意见和建议。