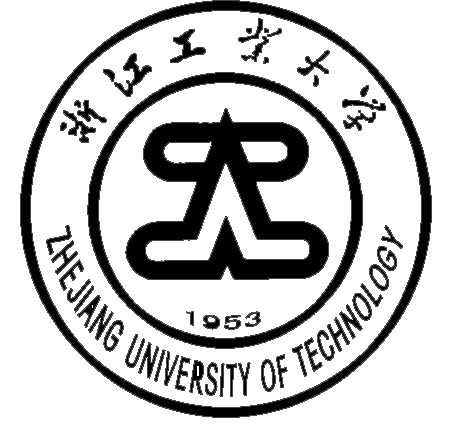


软件工程课程设计

－易仓管理系统项目技术文档

**2023-2024(1）**



项目名称 易仓管理系统——小微型企业与个体商户的智能仓储物流解决方案

组长姓名 乔哲虎

小组其他成员姓名 申屠名晓 王培宇

指导教师 董天阳

所在学院 计算机科学与技术学院

提交日期 2024年1月29日

《软件工程课程设计自评表》

|  |  |  |
| --- | --- | --- |
| 成员姓名 | 分工和任务情况 | 项目自评 |
| 乔哲虎 | 1. 前后端代码编写 2. 规范化 |  |
| 申屠名晓 | 1. 货物识别 2. 订单识别 |  |
| 王培宇 | 1. 短信提醒 2. 订单二维码 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 成员签名 |  |  |  |

目录

[第1章 绪论 4](#_Toc157428095)

[1.1 项目概述 4](#_Toc157428096)

[1.2 组织结构与团队分工 4](#_Toc157428097)

[第2章 项目管理计划说明书 6](#_Toc157428098)

[2.简介 6](#_Toc157428099)

[2.1.2 项目概述 6](#_Toc157428100)

[2.2参考材料 7](#_Toc157428101)

[2.3项目组织 7](#_Toc157428102)

[2.3.1外部接口 7](#_Toc157428103)

[2.3.2内部接口 8](#_Toc157428104)

[2.3.3角色和责任 8](#_Toc157428105)

[2.4管理过程计划 8](#_Toc157428106)

[2.4.1启动计划 8](#_Toc157428107)

[2.4.2工作计划 10](#_Toc157428108)

[2.4.3控制计划 14](#_Toc157428109)

[2.4.4风险管理计划 15](#_Toc157428110)

[第3章 需求说明书 17](#_Toc157428111)

[3.1 系统功能需求： 17](#_Toc157428112)

[3.2用户界面要求： 18](#_Toc157428113)

[3.3性能需求： 19](#_Toc157428114)

[3.4安全性与隐私保护需求 19](#_Toc157428115)

[3.5法规遵从性或行业标准要求 19](#_Toc157428116)

[第4章 传统的分析说明书 19](#_Toc157428117)

[4.1可行性分析： 19](#_Toc157428118)

[4.2需求分析结果： 20](#_Toc157428119)

[第5章 面向对象的分析说明书 25](#_Toc157428120)

[5.1对象模型： 25](#_Toc157428121)

[5.2动态模型： 28](#_Toc157428122)

[5.3属性与方法： 52](#_Toc157428123)

[第6章 面向对象的设计说明书 61](#_Toc157428124)

[6.1设计原则： 61](#_Toc157428125)

[6.2类设计： 61](#_Toc157428126)

[6.3架构设计： 65](#_Toc157428127)

[6.4数据库设计： 65](#_Toc157428128)

[关系模式设计： 65](#_Toc157428129)

[数据类型定义 66](#_Toc157428130)

[ER图 69](#_Toc157428131)

[数据库操作的设计： 69](#_Toc157428132)

[第7章 测试技术说明书 75](#_Toc157428133)

[7.1测试策略： 75](#_Toc157428134)

[7.2测试用例设计： 76](#_Toc157428135)

[7.3测试环境搭建： 77](#_Toc157428136)

[7.4测试工具选择： 77](#_Toc157428137)

[7.5测试进度安排和质量指标： 77](#_Toc157428138)

[第8章 用户使用说明书 78](#_Toc157428139)

[系统简介： 78](#_Toc157428140)

[安装与启动： 78](#_Toc157428141)

[功能模块介绍： 79](#_Toc157428142)

[界面与导航： 79](#_Toc157428143)

[常见问题与解决方案： 79](#_Toc157428144)

[系统维护与技术支持： 79](#_Toc157428145)

[附录： 79](#_Toc157428146)

[第9章 做小组项目的体会与心得 80](#_Toc157428147)

[9.1 80](#_Toc157428148)

[参考文献 81](#_Toc157428149)

[附录 83](#_Toc157428150)

[附录A X小组XXX项目的源代码.rar 83](#_Toc157428151)

[附录B X小组XXX项目的安装说明书.doc 83](#_Toc157428152)

[附录C X小组XXX项目的原图.rar 83](#_Toc157428153)

# 绪论

## 1.1 项目概述

1. 项目背景

随着市场竞争的日益激烈，客户对企业的物流服务质量和水平有了更高标准的要求，从而推动当下许多企业存在较多仓储管理问题亟需解决。小微型企业和个体商户面临了新的仓储物流挑战，如以下几个问题

1. 空间利用率与货物定位困难

由于规模较小，仓库设施和管理系统可能较为简陋，缺乏明确的区域划分和智能的库存管理系统，导致货物堆放混乱，无法有效利用有限的空间资源。此外，由于缺乏精准的定位追踪技术，找货效率低下，影响订单处理速度

1. 库存管理不精确

小微企业往往没有完善的库存管理体系，账目不清、条目逻辑混乱，库存数据更新不及时，易出现物资储备过多或过少的情况，不仅增加了存储成本，还可能导致缺货断供的风险

1. 物流信息透明度低

缺乏实时的货物跟踪功能，使得客户对订单状态无法进行准确掌握，降低了服务体验，同时也不利于企业内部进行有效的物流调度和决策。

1. 仓库账目管理混乱，没有条目逻辑；
2. 仓库管理体系不明确，导致管理不规范；
3. 仓库运行效率低，找货时间长，库存耗时且不准确。

针对上述问题，我们拟设计一个“易仓管理系统”尝试解决这些问题

1. 项目目标和意义

我们设计的“易仓仓储管理系统”通过提供仓库管理、货物管理、订单管理以及货物识别和跟踪等功能，旨在尝试解决这些问题。系统将实现智能化分区管理、精细化库存控制、规范化业务流程，并通过货物识别技术和物流跟踪功能，提高仓储物流的整体效率和服务质量，从而满足小微型企业和个体商户对于提升仓储管理水平和优化物流服务的需求。

## 1.2 组织结构与团队分工

**要注明每个人的分工以及实际做了哪部分。**

小组成员角色和职责分配

1. 组织结构

该小组采取了扁平化且职能分明的组织结构，由以下三个核心角色构成：

1. **组长**：作为团队的最高决策者和协调者，直接对整个项目负责。在技术层面，组长不仅承担着前后端开发的整体架构设计与实现工作，同时也在非技术层面扮演项目经理的角色，把控项目的整体进度、任务分配和团队协作。
2. **数据库及接口开发者**：作为小组中的关键技术人员之一，专注于系统后台支持部分，具体职责包括构建高效稳定的数据存储方案以及设计并实施货物识别与订单识别的核心接口功能，确保数据准确无误地流转和处理。
3. **需求分析及辅助功能开发者**：在团队中主要承担前期的需求理解和后期的用户体验优化工作。具体任务涵盖了从需求分析图表绘制以直观展示业务流程，到实际操作层面上的订单二维码生成和货物状态短信通知等功能的设计与实现，这些都极大地提升了系统的实用性和用户友好性。

通过这样的组织结构设置，小组内部形成了良好的互补和协同效应，确保了项目的有序进行和目标达成。

1. 团队分工

组长角色与职责： 组长作为整个项目的领导者和核心开发者，主要负责项目整体的统筹规划、进度控制以及关键部分的技术实现。具体分工包括：

* 项目管理：制定项目计划、分配工作任务、跟踪项目进度、协调团队内部资源。
* 前后端代码编写：承担系统的主要开发工作，涵盖前端用户界面设计与交互逻辑编程，以及后端服务器架构搭建、业务逻辑处理和接口设计。

成员一角色与职责： 该成员专注于数据库管理和特定功能模块的实现，其具体分工为：

* 数据库设计：负责构建符合项目需求的关系型或非关系型数据库模型，确保数据存储的安全性、有效性和高效访问性能。
* 货物识别与订单识别接口实现：设计并开发用于自动识别货物信息和解析订单数据的相关接口，确保数据准确无误地从外部系统输入到内部系统中。

成员二角色与职责： 另一位成员则侧重于分析文档制作和相关辅助功能的开发，其具体任务包括：

* 需求分析图表绘制：根据项目需求进行详细的需求分析，并通过可视化工具（如九数云）绘制人员结构分析图表或其他类型的需求分析图来优化项目流程和组织结构。
* 订单二维码生成：设计和开发生成订单二维码的功能模块，以便在实际操作中快速扫描获取订单详情。
* 货物短信提醒：负责集成短信服务，实现在特定业务场景下向客户发送关于货物状态变化的实时短信提醒。

# 项目管理计划说明书

## 2.简介

### 2.1.2 项目概述

#### 1. 意图、范围和目标

* 意图与范围：

本课程设计项目将在有限的时间内，即从12月14日至1月26日，专注于开发一款简化版的易仓仓储管理系统原型。系统将针对核心的仓储管理功能进行设计与实现，包括基础的仓库区域划分、货物入库出库记录、订单处理以及初步的货物识别、订单识别、二维码生成和物流状态跟踪等。

* 目标：

在短短一个半月的时间里，团队成员将密切协作，运用Vue.js前端框架、Spring Boot后端技术以及MySQL数据库等构建具有实用价值的软件应用，并通过模拟数据验证其基本功能，为后续深入开发提供坚实的基础。

#### 2. 设想和限制

* 设想：

鉴于时间紧迫，项目初期将集中力量完成关键功能模块的开发，如仓库管理和订单管理系统的搭建，同时利用现成的API服务（例如百度OCR）实现简单版本的货物识别、订单识别等功能。此外，可能会采用二维码生成和扫描作为临时替代方案以模拟物流状态追踪。

* 限制：

由于课程设计周期较短，复杂的实时物流追踪和AI图像识别等功能可能仅能实现概念性演示，而非全面集成。同时，团队也将侧重于代码质量控制及文档编写，确保项目的可维护性和扩展性。

#### 3.可交付项目

* 核心功能模块

包含简易版的仓库管理模块（区域划分与库存更新）、订单管理模块（创建、查询与状态变更）以及基于API服务的货物识别功能演示。

* 配套文档

项目管理计划说明书，需求说明书，传统的分析说明书，面向对象的分析说明书 ，面向对象的设计说明书，测试技术说明书，用户使用说明书等详尽的技术文档。

#### 4. 时间表概述

* 在12月14日至12月31日期间，主要完成需求分析与系统设计工作；
* 1月1日至1月21日，集中精力进行编码开发与单元测试；
* 1月21日至1月26日，进行集成测试、系统优化以及撰写最终报告和用户使用指南，准备项目验收。

## 2.2参考材料

* 《1058.1-1987 - IEEE Standard for Software Project Management Plans》
* 《1058-1998 - IEEE Standard for Software Project Management Plans》
* 《16326-2009 - ISO/IEC/IEEE International Standard - Systems and Software Engineering--Life Cycle Processes--Project Management》

## 2.3项目组织

### 2.3.1外部接口

1.百度API集成：

项目通过调用百度的智能服务接口，实现了订单货物识别和仓库定位路线规划功能。具体而言，系统与百度OCR API进行交互以识别订单上的货物信息，并利用地图与路径规划API来对仓库调拨作业进行定位于路径规划。

2.阿里云API调用：

系统连接阿里云提供的短信发送服务API，实现对货物状态变化、重要订单提醒等事件的通知功能。当货物出入库或订单状态变更时，系统自动触发短信发送至相关负责人或客户，确保信息实时传递。

### 2.3.2内部接口

#### 1.技术栈选用

易仓仓储管理系统采用前后端分离架构，前端使用Vue.js框架构建用户界面，后端基于Spring Boot搭建业务逻辑层，并结合MySQL数据库存储核心数据。

#### 2.功能模块划分

系统内部分为仓库管理模块、货物管理模块、订单管理模块、物流运输模块以及图像识别模块，各模块之间通过API接口相互协同工作，保证了系统的高效性和数据一致性。

### 2.3.3角色和责任

* 项目经理/组长：

负责整个项目的统筹规划、协调资源以及关键的前后端代码编写工作，确保系统基本功能顺利实现并按期交付。

* 数据库工程师与接口开发人员：

负责数据库的设计与优化，包括表结构定义、索引建立以及SQL性能调优；同时承担对接百度API的任务，实现在货物识别与订单识别功能中的准确应用。

* 分析师与UI/UX设计师：

该成员主要负责需求分析工作，包括绘制图表梳理业务流程，以及设计并实现订单二维码生成机制，确保订单流通过程中信息的安全传输与便捷读取。此外，还承担起配合阿里云短信服务接口，设置和维护货物状态变动时的短信提醒功能，确保系统在货物跟踪环节具备及时有效的通知能力。

## 2.4管理过程计划

### 2.4.1启动计划

#### 1.估算计划

* 工作分解结构（WBS）：

通过将整个项目拆分为可管理的组件（如前端开发、后端开发、数据库设计、API接口集成等），为各个任务分配工作量和时间。

* 三点估算：

针对每个子任务，基于历史数据、专家判断及风险分析，分别进行最乐观（O）、最可能（M）、最悲观（P）的时间和成本估算，计算出期望工期（E = (O + 4M + P) / 6）和相应的成本。

* 敏捷估算法：

对于迭代开发过程中的需求变更或技术实现不确定性，采用敏捷方式定期重新评估剩余工作量，并据此调整进度计划。

#### 2.人员安置计划

根据项目需求和技术栈，团队配置如下：

* 组长/项目经理：1人，全程参与项目，负责整体协调与主要代码编写。
* 前后端开发工程师：组长兼任此角色，主要负责vue前端框架以及springboot后端框架的开发。
* 数据库设计师及接口开发人员：1人，负责数据库设计并对接百度API实现货物与订单识别功能。
* 需求分析师兼UI/UX设计师：1人，负责需求分析、图表绘制、订单二维码生成和阿里云短信服务接口的集成。

以上人员在整个项目的生命周期内全职投入，具体时间周期与项目总体周期同步。

#### 3.资源获取计划

* 硬件资源：

所有开发人员拥有足够的高性能笔记本电脑，以及稳定的校园网网络环境支持远程协作。

* 软件资源：
  + 开发工具：IntelliJ IDEA、Visual Studio Code等开发环境。
  + 技术平台：购买并注册使用Vue.js、Spring Boot相关的开发许可证（如有必要）。
  + 第三方服务：申请调用百度API和阿里云短信服务的相关开发者账号，获得相应接口调用权限及测试额度。
  + 版本控制：利用免费的GitHub远程仓库进行版本控制和协作开发。
* 服务资源：

采购和维护服务器资源，用于部署系统应用，并确保满足性能要求。同时，需要考虑对MySQL数据库服务器的租赁或自建（暂时止于本地部署测试阶段，未采购相应服务器）。

#### 4.项目人员培训计划

* 技术培训：鉴于项目采用的技术栈，安排专项培训课程，包括但不限于：
  + Vue.js基础到高级特性
  + Element UI组件库的使用
  + Spring Boot框架开发及微服务架构实践
  + 百度API和阿里云API的调用文档学习及实战演练
  + MySQL数据库设计与优化
* 协作工具培训：指导团队成员熟悉和掌握GitHub的使用，包括代码提交、分支管理、Pull Request流程等。
* 持续学习：鼓励团队成员在项目执行过程中不断学习新知识，适应技术更新，提升个人能力，并定期组织内部分享会交流心得经验。

### 2.4.2工作计划

#### 1.工作活动

* 需求分析与系统设计（12月14日至12月31日）
  + 需求收集和整理：调研用户需求，明确易仓管理系统的核心功能点。
  + 需求分析文档编写：梳理详细需求清单，并绘制流程图、数据流图等图表。
  + 系统架构设计：确定前后端分离技术栈，设计数据库模型，规划模块划分及接口定义。
  + 技术选型确认：确定使用百度API进行货物识别、订单识别和路线规划，阿里云API进行短信发送。
* 编码开发与单元测试（1月1日至1月21日）
  + 前端开发任务：
    - 仓库管理界面开发（vue+elementUI）
    - 订单管理模块实现
    - 货物管理模块实现
    - 路线规划功能集成
  + 后端开发任务：
    - springboot后端服务搭建
    - 数据库表结构创建与初始化
    - 百度云订单识别接口实现
    - 百度云货物识别接口实现
    - 百度云路线规划实现
    - 阿里云短信发送接口对接
    - 订单生成二维码功能实现
  + 单元测试计划：

针对每个模块编写并执行单元测试用例，确保代码质量。

* 集成测试与优化阶段（1月21日至1月26日）
  + 系统集成：将各模块代码整合为完整系统，解决可能存在的接口兼容问题。
  + 功能测试：全面验证系统的各项功能是否满足设计要求，包括外部API的集成效果。
  + 性能优化：根据测试结果对系统进行性能调优，如数据库查询优化、前端加载速度提升等。
  + 文档编写：完成最终报告撰写，包含项目概述、关键技术、实现细节、测试结果等内容。
  + 用户指南制作：编制简洁明了的用户操作手册，便于实际用户快速上手系统。

#### 2.时间表分配

* 需求分析与设计工作依赖于前期的信息收集和团队内部讨论，是后续编码的基础，需先于开发进行。
* 编码开发完成后，才能进入集成测试环节，以验证各个独立开发的部分能否协同工作。
* 最终验收前的工作（集成测试、优化、文档撰写和用户指南）相互关联，形成一个闭环，确保项目的完整性。

基于以上三点，以及之前4.2.1中对工作活动的划分，绘制了以下WBS工作划分结构（如图4.1）以及时间表（如图4.2，使用Monday.com的服务进行项目管理划分）

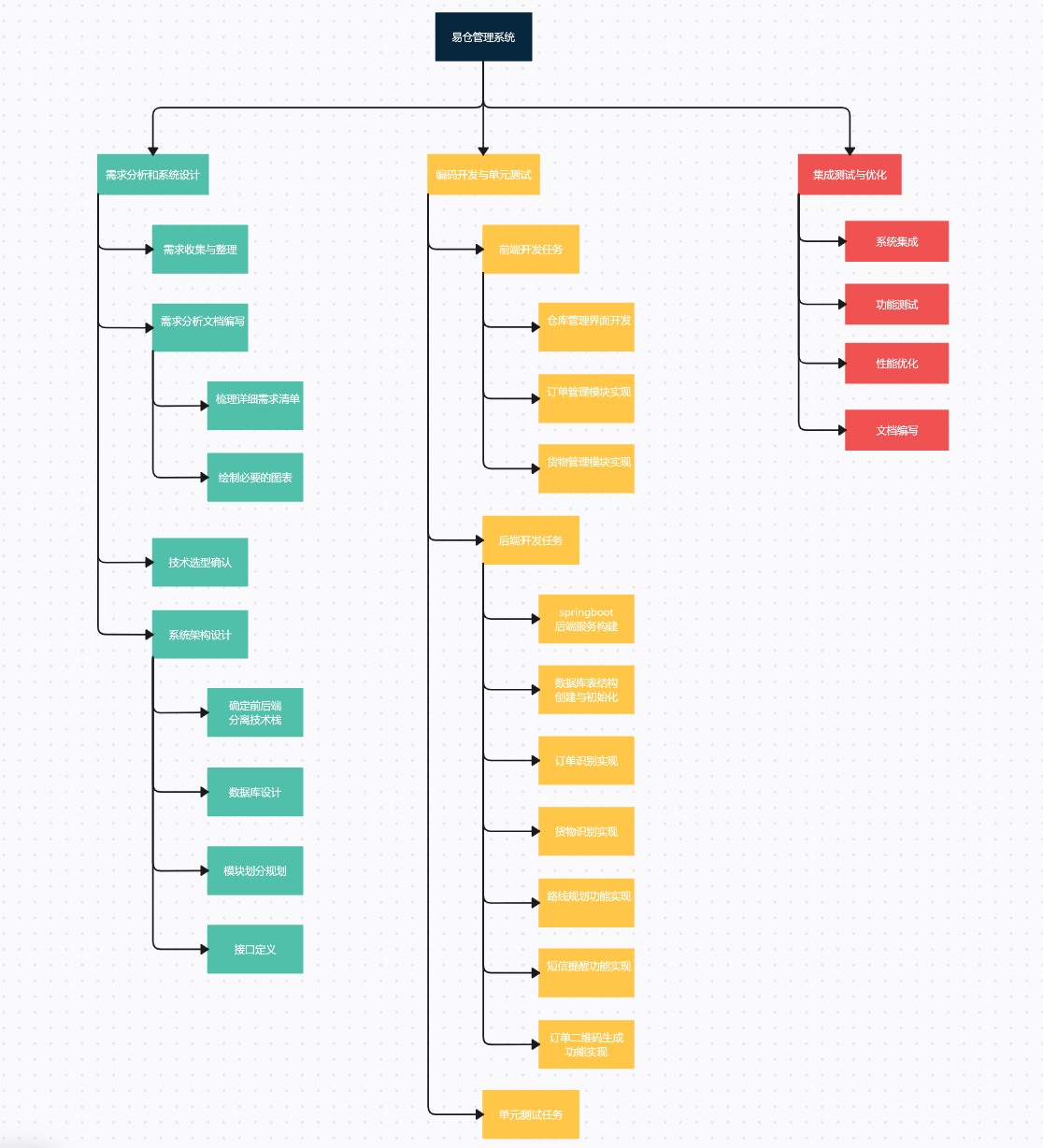


图2.1：WBS工作划分结构

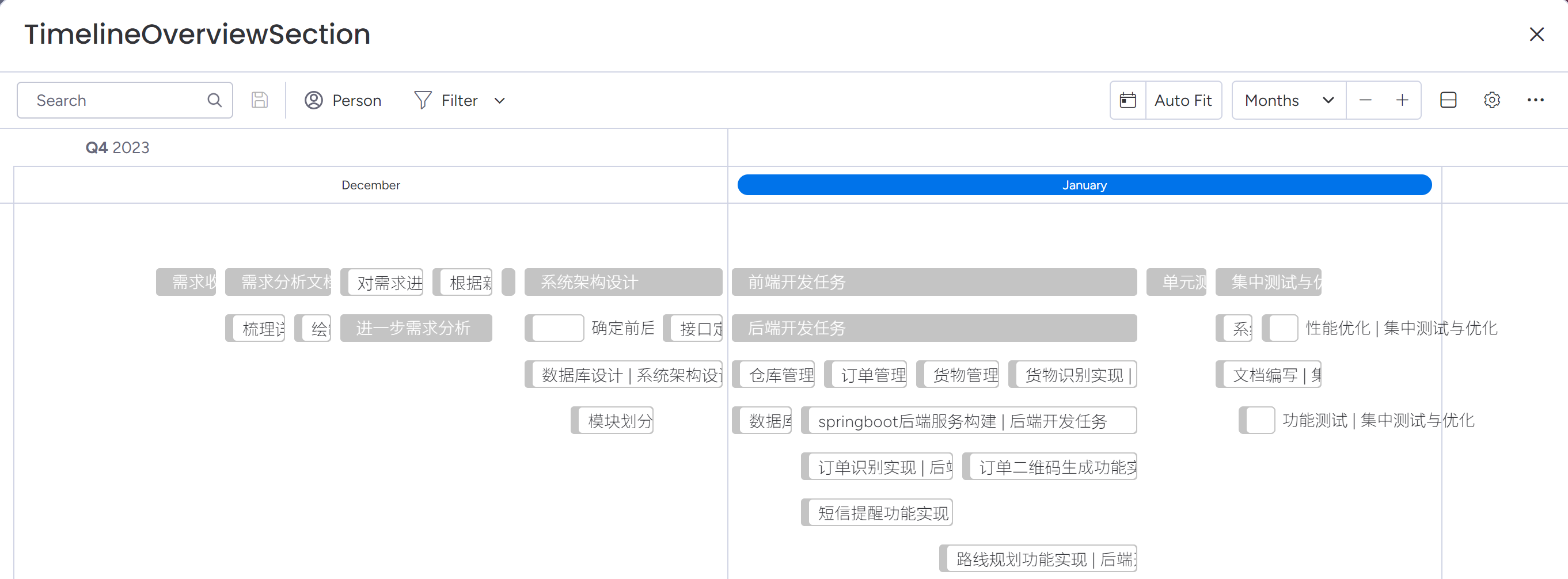


图2.2：工作划分时间线视图

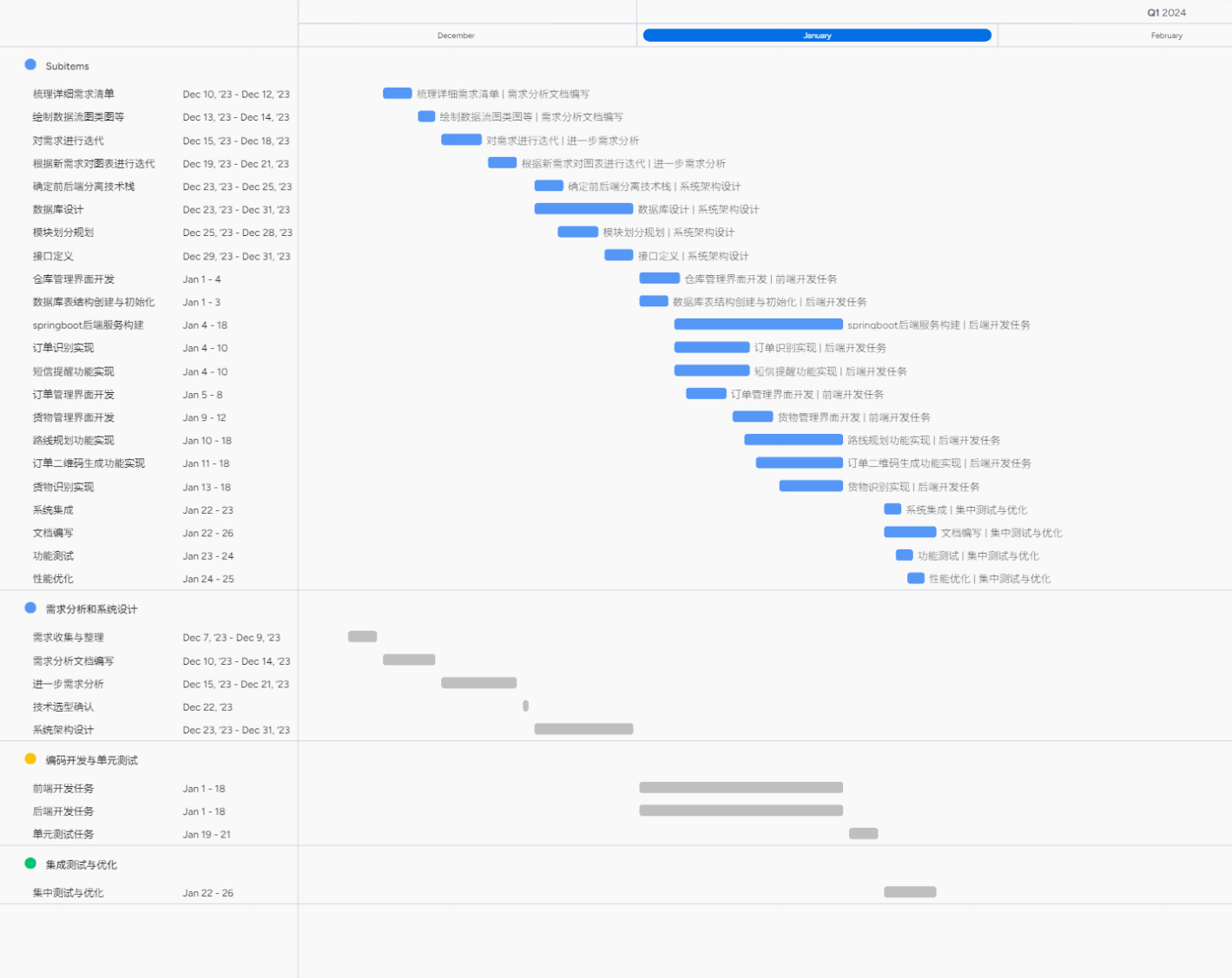


图2.3工作管理Gantt图

#### 3.资源分配

* 组长/项目经理：

全程参与所有阶段，负责统筹协调以及关键模块的开发。

* 数据库设计师及接口开发者：

在需求分析阶段配合组长完成数据库设计，在编码阶段主要负责数据库开发与API接口的实现。

* 需求分析师兼UI/UX设计师：

主要负责需求分析、图表绘制和相关功能模块的前端实现，同时负责二维码生成和短信提醒功能的开发。

#### 4.预算分配

* 开发工具与软件许可费用：购买或续费Vue.js、Spring Boot相关的开发环境和许可证（如有必要）。
* 第三方服务费用：
  + 百度API调用产生的费用预算，按预计调用量预估成本。
  + 阿里云短信服务费用预算，根据预期发送量计算。
* 硬件资源租赁或购置费用：服务器、网络设备等基础设施投入预算。(开发阶段仅使用自有笔记本作为服务器)
* 人力资源成本：根据成员工作时长、薪酬标准分配人员工资预算。但本次开发作为课程设计，并无该方面预算。
* 测试与优化成本：可能涉及模拟数据生成、负载压力测试等第三方服务成本。
* 培训成本：使用网络上免费的公开学习资源进行学习，仅需要时间成本
* 运维成本：考虑系统上线后的运维支持、更新维护等长期支出。

### 2.4.3控制计划

#### 1.需求控制计划

* 需求变更管理：由于项目需求相对稳定，小组将采用敏捷开发中的轻量级需求管理方法。在需求分析阶段明确并记录原始需求，并将其存档于GitHub仓库的文档区域，作为基线。
* 监控机制：任何新需求或现有需求的变更必须通过微信群或Github提交，经过全体组员讨论和评估其对现有进度的影响以及可行性。
* 控制流程：对于影响系统核心功能的需求变动，需经过组长审批；非关键性需求调整可以由相关模块负责人决定是否采纳。所有需求变更应更新至需求规格说明书，并确保与项目进度、代码实现同步。

#### 2.时间表控制计划

* 进展测量：以周为单位设定开发里程碑，每个成员每周汇报各自完成的工作内容及进度，同时使用Monday的项目管理计划查看工作状态。
* 落后应对措施：如果实际进展落后于计划，首先分析滞后原因，如技术难题、资源分配不均等。根据问题性质采取相应措施，可能包括但不限于优化工作流程、增加协作时间、调整分工或者必要时加班赶工。
* 项目进度监控工具：利用Monday项目的任务分配、截止日期提醒等功能，确保每个团队成员清晰了解自己的任务完成期限。

#### 3.预算控制计划

* 由于本次课程设计项目几乎不需要考虑直接成本，主要投入在于人力和API调用费用。
* API成本控制：百度和阿里云API调用均有免费额度，在开发过程中合理规划API调用，避免超出免费范围导致额外支出。若出现超出预期的调用量，可重新评估是否有必要升级服务套餐或优化功能以减少调用量。
* 时间成本：虽然小组未设置严格的预算约束，但依然需要关注隐性的时间成本，保证高效利用有限的开发周期。

#### 4.质量控制计划

* 编码质量：

实施代码审查制度，确保SpringBoot和Vue.js编码规范一致且高质量。利用IDE内置的静态代码检查工具辅助提升代码质量。

* 功能测试：

针对易仓管理系统各个功能模块进行单元测试和集成测试，确保货物识别、路线规划、短信发送等功能正确无误。编写自动化测试脚本，以便在代码迭代中快速验证功能稳定性。

* 用户体验：

在系统设计阶段充分考虑用户界面友好性和操作便利性，结合验收标准进行系统优化。

* 文档质量：

撰写详细的设计文档、测试报告以及最终用户手册，作为质量控制的重要组成部分。在项目后期尤其注重系统优化和用户体验改进。

### 2.4.4风险管理计划

在本次课程设计项目中，考虑到团队规模较小（仅包含三个成员），且开发工作在学校环境中使用个人笔记本进行，我们将重点针对以下风险制定管理计划及应对策略：

#### 1.人力资源风险

* 风险描述：由于团队成员数量有限，可能会因学业压力、健康状况或其他个人事务导致人力资源紧张。
* 应对策略：
  + 制定灵活的工作分配表，确保任务量均匀分布并预留一定的缓冲时间。
  + 定期组织小组会议，及时了解每位成员的进度和困难，并提供必要的支持。
  + 建立备份机制，每个关键环节至少有两人熟悉，以备不时之需。

#### 2.技术风险

* 风险描述：成员的技术能力差异可能导致项目开发效率降低或出现难以解决的技术难题。
* 应对策略：
  + 提前开展技术培训和学习分享会，提升团队整体技术水平。
  + 遇到技术难题时，利用在线资源、教师指导以及同学间协作解决问题。
  + 对于复杂模块，采用分阶段开发和迭代测试的方式，逐步攻克难点。

#### 3.硬件设备风险

* 风险描述：使用个人笔记本开发可能遇到设备故障、性能不足或软件环境不稳定等问题。
* 应对策略：
  + 要求团队成员提前检查并优化自身电脑配置，保证开发环境稳定可靠。
  + 采用云端开发环境如GitHub Codespaces等远程开发工具，实现异地协同开发和数据备份。
  + 若设备出现问题，立即启动应急预案，利用学校计算机实验室或借用其他可用资源。

#### 4.沟通协作风险

* 风险描述：由于团队小且分散，沟通不畅可能影响项目进度和质量。
* 应对策略：
  + 使用高效的团队协作工具，如微信群保持日常沟通。
  + 设定固定的团队会议时间(每周四或周五)，讨论问题、同步进展并确定下一步工作计划。
  + 在github远程仓库每次提交或修改代码、文件时都对改变做出解释，保证所有重要决策、设计方案和代码变更都记录在案，便于团队内部查阅和跟进。

#### 5.时间管理风险

* 风险描述：课程设计期限临近，可能出现赶工现象，影响项目最终成果的质量。
* 应对策略：
  + 制定详细的项目时间线和里程碑计划，合理安排各个阶段的任务完成时间。
  + 设置中期检查点，评估项目进度，对滞后部分及时调整资源投入。
  + 在项目初期就养成良好的时间管理习惯，避免后期赶工。

通过以上风险管理措施，本团队旨在最大程度地减少潜在问题的影响，确保项目的顺利进行和高质量交付。

# 需求说明书

## 3.1 系统功能需求：

1. **仓库管理模块**

* 库区与库位管理：系统应支持多层级的仓库增减启用，仓库内库区划分、库位定义，实现精准的空间资源管理。
* 库存管理：提供实时库存查询、库存盘点、库存上下限预警等功能，确保库存数据准确无误。
* 入库管理：包括采购入库、调拨入库、订单入库等操作流程，对接货物识别功能快速完成实物与系统信息匹配，同时记录详细的入库单据和货物信息。
* 出库管理：涵盖销售出库、调拨出库等多种场景，根据订单需求进行拣选配货，支持动态更新库存状态。

1. **货物管理模块**

* 商品管理：支持多维度的商品分类、编码体系，录入详尽的商品属性信息（如重量、体积、单价），并可根据订单追踪单品或批次的历史流转记录。
* 二维码管理：原定自动生成或对接外部打码系统为每件货物赋予唯一标识符，支持条形码和二维码扫描以快速获取货物详情。由于技术以及时间的限制，暂时为为每一个订单生成一个二维码储存订单信息。

1. **订单管理模块**

* 订单创建与审核：接收来自各种渠道的订单信息，支持一键导入、手动录入或API接口识别接入，具备完善的订单审核机制。
* 订单批量录入：针对批量订单处理时，系统能够灵活地进行批量录入，满足不同发货要求。
* 订单识别：利用OCR技术自动识别并解析上传的订单文件，减少人工干预，降低错误率。

1. **物流管理模块**

* 物流路线规划：集成百度地图API，根据发货地址、收货地址及实时交通情况智能规划最优配送路径，提升物流时效性。

1. **自动化提醒与交互模块**

* 短信提醒服务：通过阿里云短信服务API，在关键业务节点发送自动通知，例如订单确认、发货通知、异常状况报告等。
* 二维码生成与应用：每个订单或单个货物生成专属二维码，用于内部操作过程中的快速识别与定位，同时也可用于客户自助查询与签收确认。

## 3.2用户界面要求：

1. **登录与人员管理界面**

* 登录界面：简洁明了的登录界面，支持用户名/密码验证，同时提供联系功能。
* 人员管理界面：管理员界面中有任务管理界面，显示各个仓库管理人员列表，展示管理人员的信息，并且可以在修改面板对管理人员的信息（联系方式，姓名，性别，密码等）进行修改。

1. **仓库管理界面**

* 仓库管理视图：提供仓库列表，包括仓库相关的详细信息，标记库存状态，可以方便地查看仓库的储存情况，以及仓库的负责人信息，也可以设定仓库是否启用。
* 库存查询与编辑页面：提供详细的商品库存查询表单，支持按商品编码、名称等多种维度检索。

1. **货物管理界面**

* 货物信息维护页：包含货物列表展示，支持新增、编辑和删除货物，详尽的货物属性字段。
* 二维码生成与显示：能够生成符合规范二维码标签，并快捷地展示和传输，方便货物商品订单的相关信息。
* 订单管理界面
* 订单列表界面：展示入库记录，出库记录，调拨记录等多重订单形式，支持对订单信息进行一定程度的更新，可以通过订单号或其他订单信息对订单进行查询。
* 订单录入：方便快捷的订单录入界面，自动检查订单完整性；支持订单多货物快速录入功能，点击按钮可以马上录入下一条订单。
* 订单二维码查询：每一条订单都有其包含订单信息的二维码，可以通过查看二维码按钮弹出订单二维码

1. **物流路线规划界面**

* 仓库定位模块：提供友好易用的地图定位工具，在左上角显示已录入的仓库，点击仓库名能在地图上自动定位显示仓库位置。
* 路线规划展示模块：基于地图服务API，自动生成最优配送路径，并以可视化的方式呈现给用户。

1. **系统首页**

* 仓储状态栏：显示过去一段时间仓库总的出库量、入库量、仓库金额、销售金额等等。
* 仓库金额变化曲线图：基于过去一段时间仓库金额和销售金额的变化情况绘制曲线图，更直观地展示过去一段时间中仓库的运转状况。

## 3.3性能需求：

* 响应时间：系统页面加载、数据请求响应应在合理时间内完成，保证用户交互流畅，尤其对于高频操作如库存查询、订单处理等应达到毫秒级响应。
* 并发处理能力：系统应能有效应对高峰时段大量用户同时在线操作，保障高并发场景下稳定运行，例如多用户同时进行入库出库操作不受影响。
* 数据准确性：系统需要具备严格的事务控制机制，保证数据的一致性和完整性，所有交易记录和统计报表必须精确无误。
* 如响应时间、系统并发处理能力、数据准确性等非功能性需求

## 3.4安全性与隐私保护需求

* 用户数据加密存储：敏感信息如账号密码、客户联系方式、订单详情等必须经过加密处理存储在数据库中。
* 访问控制：基于角色权限的访问控制机制，确保只有授权用户可以访问相应的数据资源和服务。
* 安全传输：所有网络通信均采用HTTPS协议，保障数据在传输过程中的安全

## 3.5法规遵从性或行业标准要求

* 遵守相关法律法规，尤其是关于个人信息保护和数据安全的法律规范，如《中华人民共和国网络安全法》等。

# 传统的分析说明书

## 4.1可行性分析：

4.1.1技术可行性

技术架构：

* 系统类型： 选择本地部署，确保仓库操作系统可以在内部网络中运行，提高数据安全性。
* 技术栈： 用Spring Boot作为后端框架，Vue作为前端框架，MySQL作为数据库，确保这些技术能够满足系统的性能和功能需求。

集成性：

* 与其他系统的集成：使用Spring Boot的集成能力，确保系统能够与企业内现有的系统进行无缝集成。
* API设计：调用百度api的OCR识别，阿里云的短信服务api等等

安全性：

* 数据安全：使用MySQL的安全特性和加密机制，确保数据在传输和存储中的安全性。
* 用户身份验证和授权：设计健全的身份验证和授权系统，以确保仅有权用户能够访问特定的功能和数据。

可扩展性：

* 系统扩展性：采用模块化结构，确保系统能够轻松扩展以适应未来的业务增长。

4.1.2经济可行性

开发成本：

* 人力资源：由于开发团队只是几个大学生，成本相对较低。
* 技术工具和许可证：选择免费或开源的开发工具，以降低技术工具的成本。

运营成本：

* 维护成本：系统日常运维和维护的成本，例如一些api服务费。
* 硬件成本：如果需要购买新硬件来支持本地部署，考虑硬件成本。

4.1.3操作可行性

用户培训：

* 培训计划：制定用户培训计划，确保用户能够顺利使用新统。
* 文档：创建易于理解的用户文档，包括系统功能和常见问题解答。

变更管理：

* 用户反馈：设计一个反馈机制，以便用户能够报告问题和提供改进建议。

操作流程改进：

* 流程优化：结合系统的特性，优化仓库操作流程，确保最大限度地发挥系统的效益。

## 4.2需求分析结果：

**数据流图：**

* **顶层数据流图**

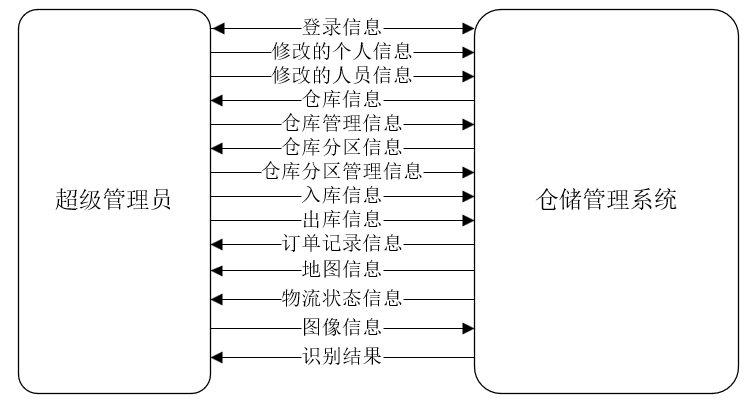


图4-1 顶层数据流图

* **第0层数据流图**

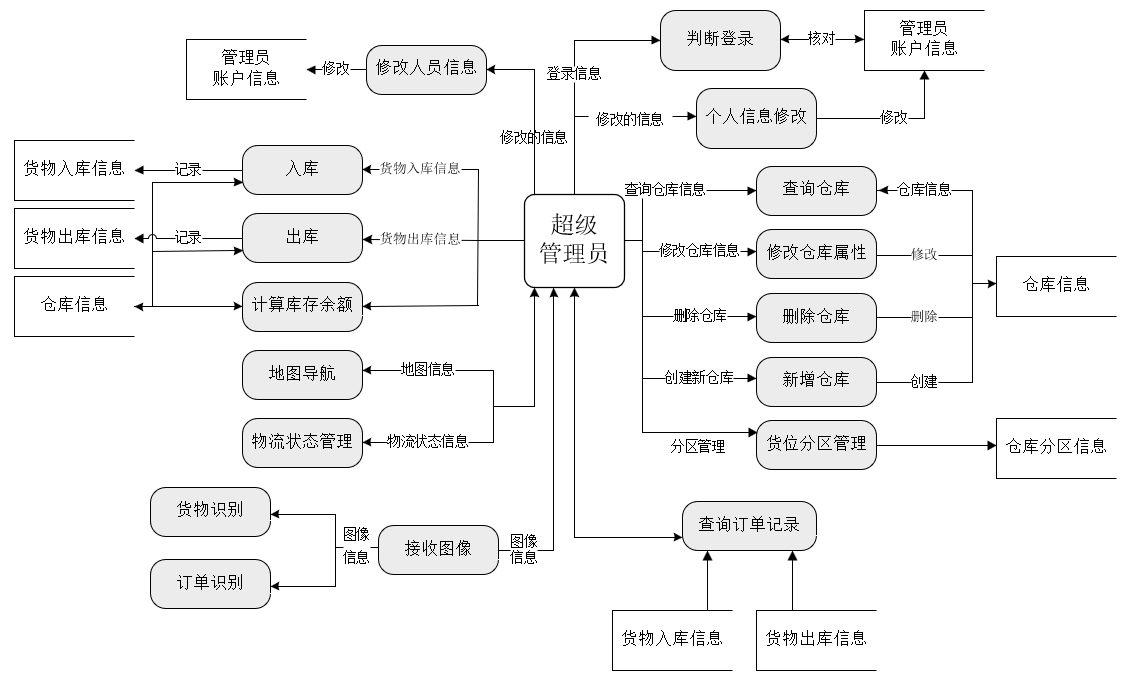


图4-2 第0层数据流图

* **第1层数据流图**
  + 账号登录数据流图

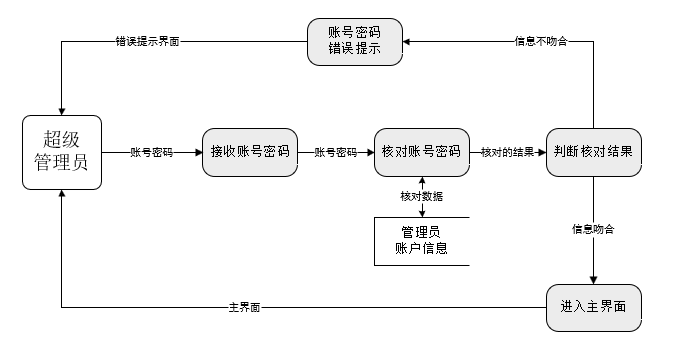


图4-3 账户登录数据流图

* + 人员管理数据流图

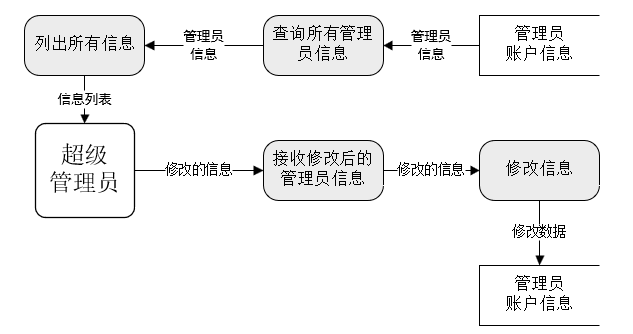


图4-4 人员管理数据流图

* + 仓库管理数据流图



图4-5 仓库管理数据流图

* + 货位分区管理数据流图

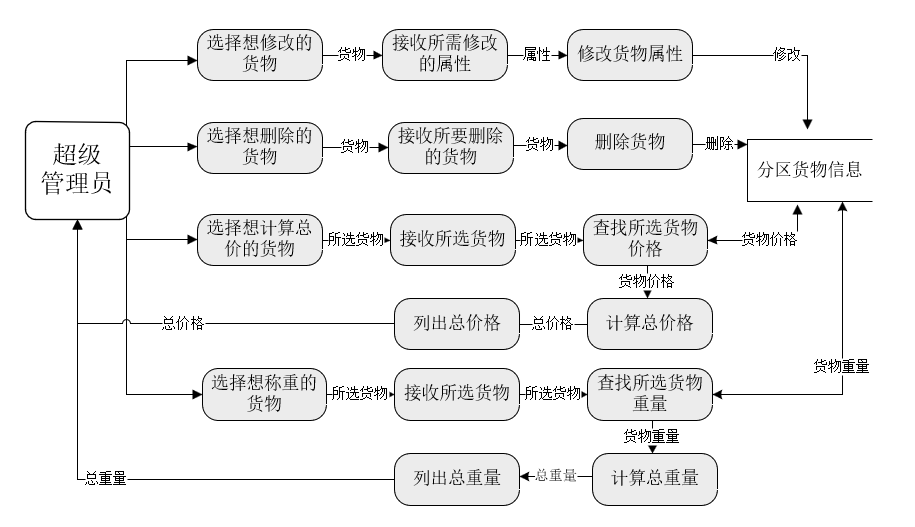


图4-6 货位分区管理数据流图

* + 货物管理数据流图

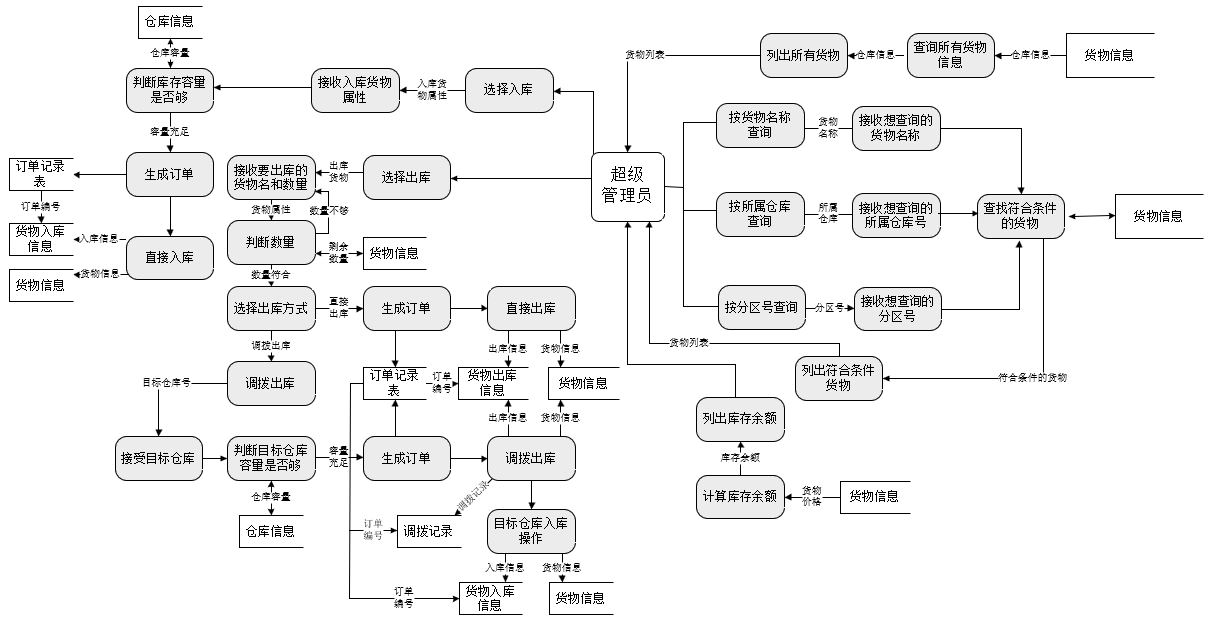


图4-7 货物管理数据流图

* + 订单记录管理数据流图

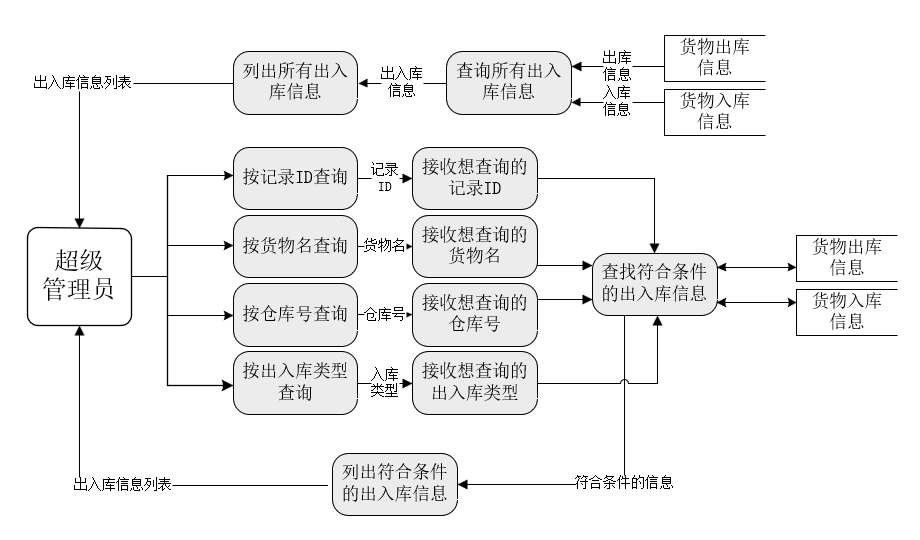


图4-8 订单记录管理数据流图

* + 物流运输管理数据流图

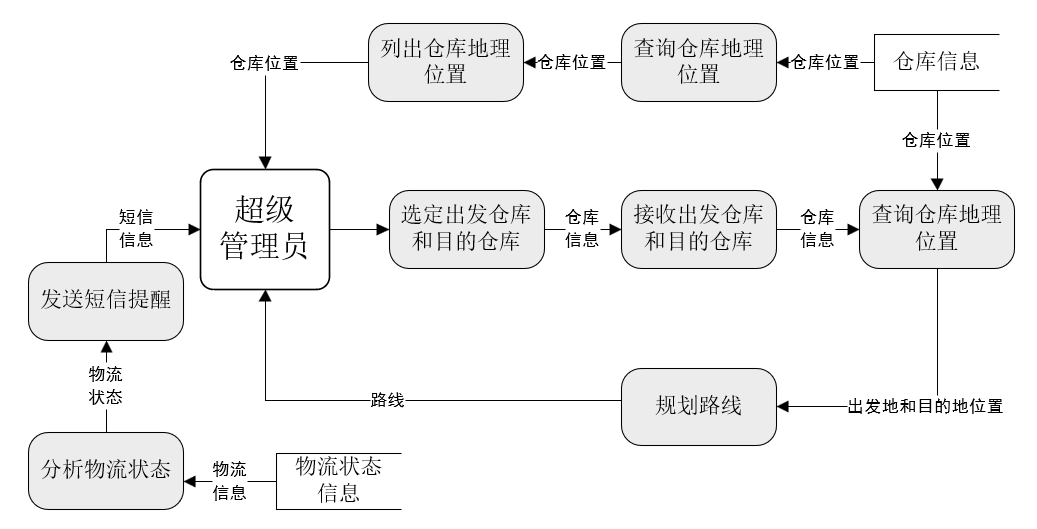


图4-9 物流运输数据流图

* + 图像识别管理

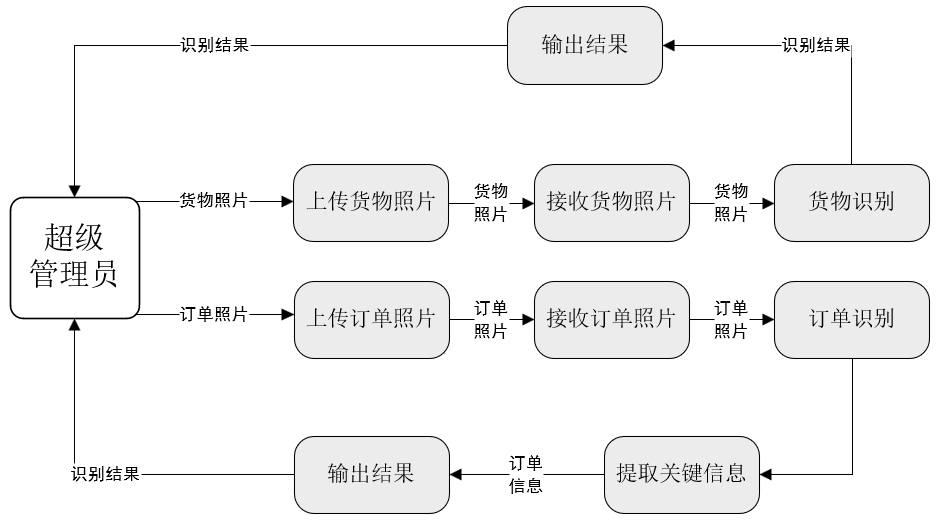


图4-10 图像识别数据流图

* + 个人信息修改数据流图

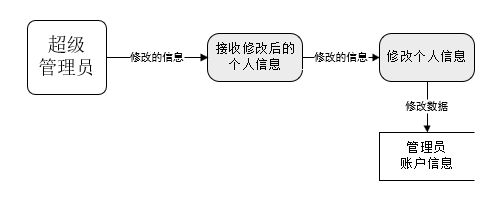


图4-11 个人信息修改数据流图

# 面向对象的分析说明书

**类图**

**每个用例的实现：用例类图、顺序图或协作图**

## 5.1对象模型：

**定义系统的类及它们之间的关系，包括实体类（如仓库、货物、订单等）和控制类（业务逻辑处理）**

**5.1.1仓库类图**

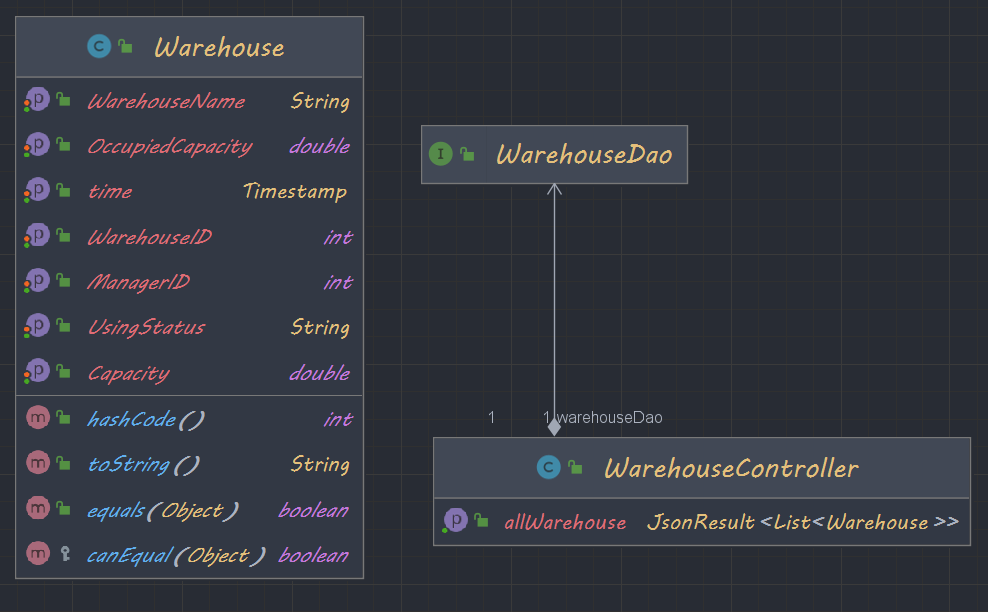


图5-1仓库类图

**5.1.2货物类图**

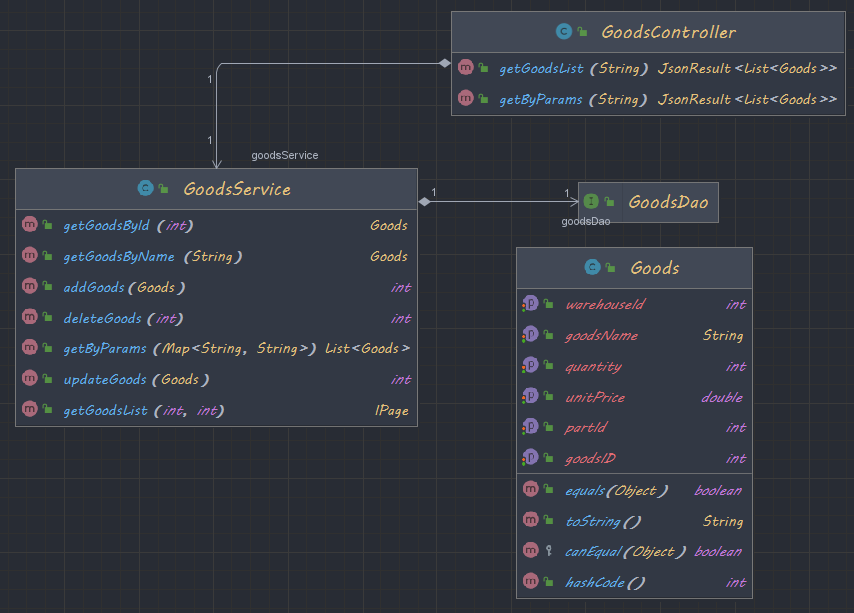


图5-2货物类图

**5.1.3订单类图**

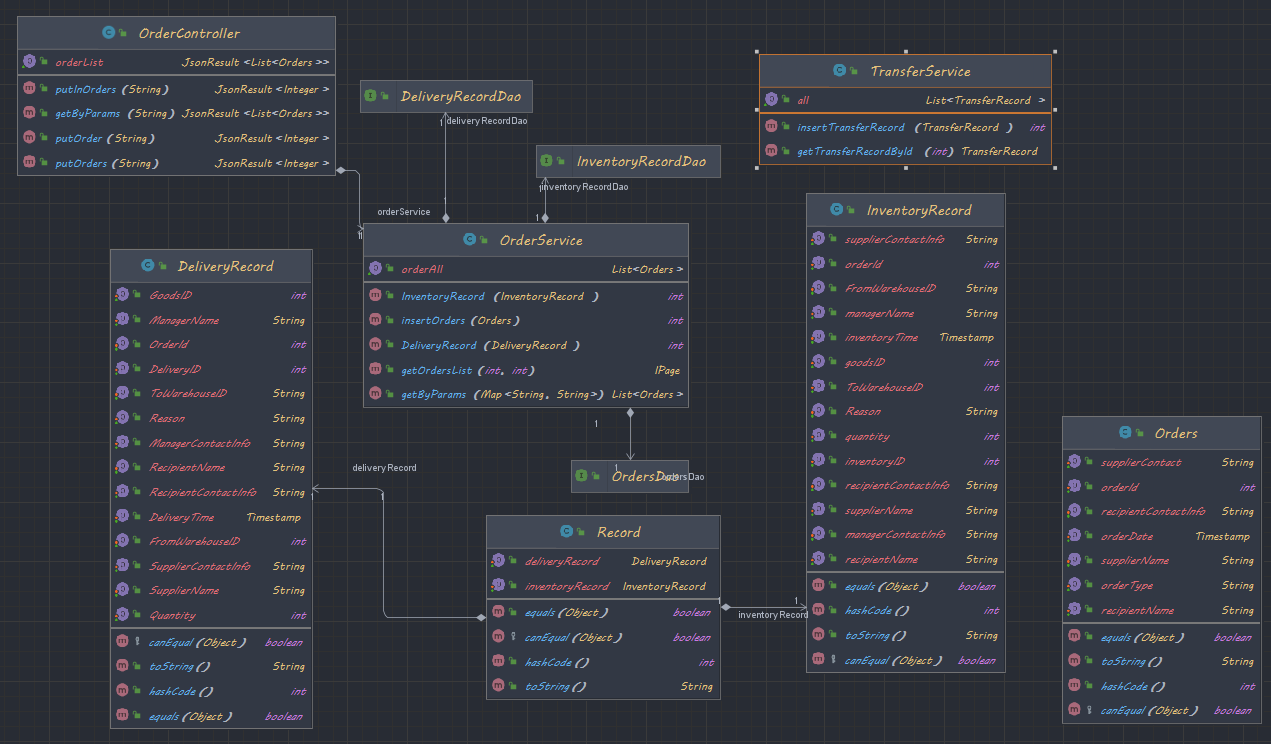


图5-3订单类图

**5.1.3人员类图**

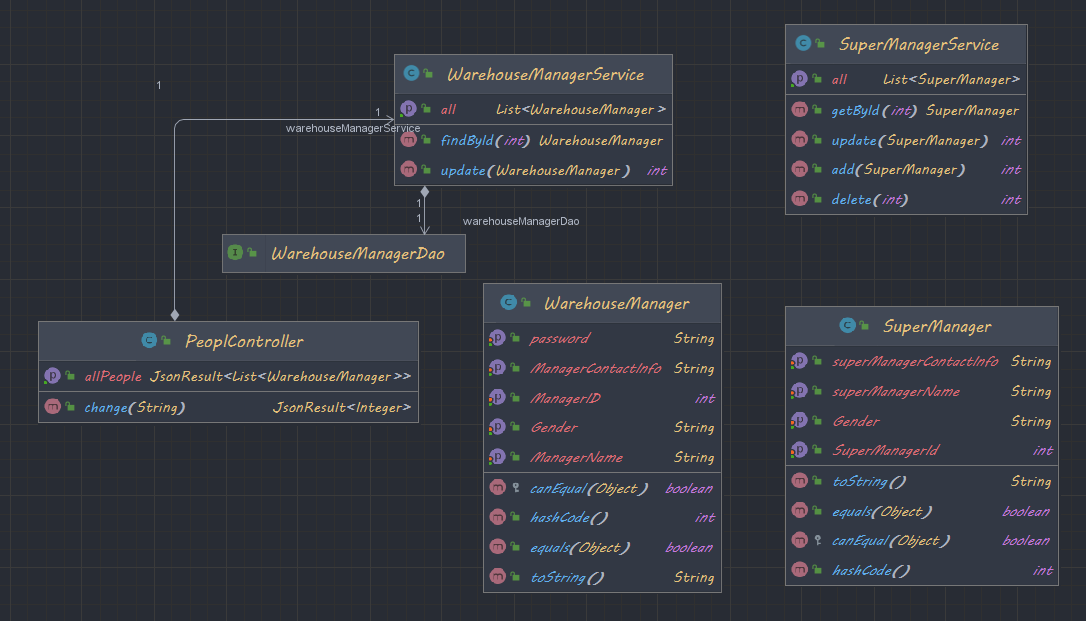


图5-4人员类图

**5.1.3智能识别类图**

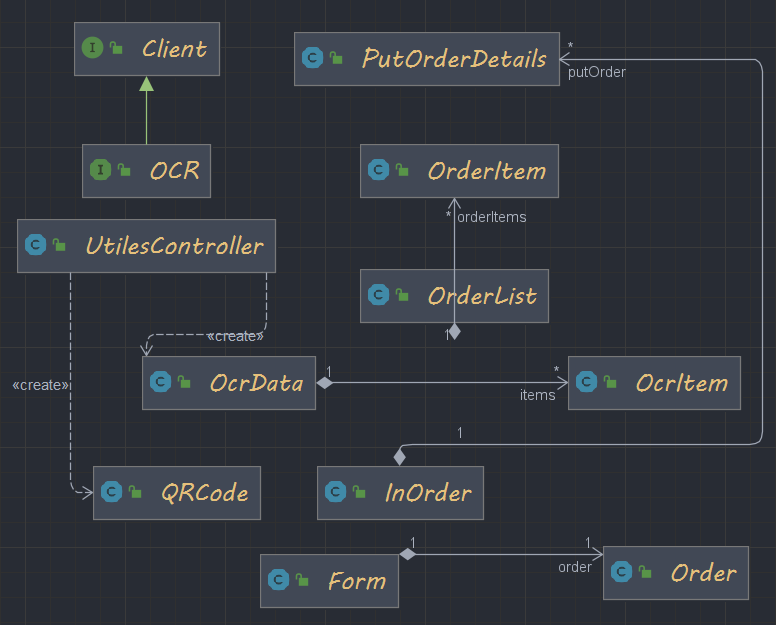


图5-5人员类图

## 5.2动态模型：

通过**用例图**描述系统外部行为，并结合**顺序图、协作图**等表示各对象间的交互过程

5.2.1 用例图

**易仓管理系统用例图：**

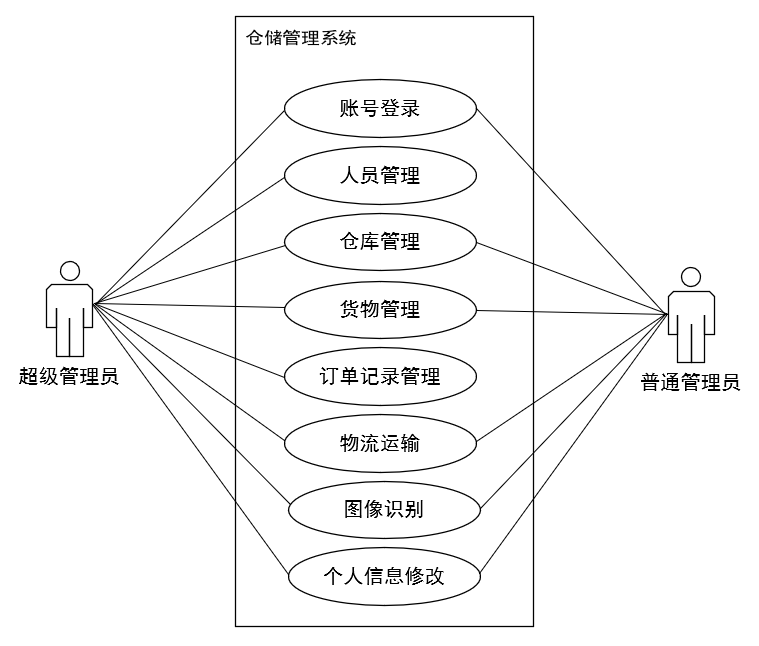


图5-？ 易仓管理系统用例图

**用例描述 ：**

表5-1账号登录用例描述

|  |  |
| --- | --- |
| 用例名称 | 账号登录 |
| 用例描述 | 使用者使用账号密码登录，系统将输入内容与数据库中账号密码进行比对，账号密码正确后成功进入系统 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.使用者输入账号密码  2.系统进行账号密码与数据库中内容比对  3.比对正确，使用者成功进入系统主界面  4.账号密码错误，提示错误信息，重新输入 |

表5-2人员管理用例描述

|  |  |
| --- | --- |
| 用例名称 | 人员管理 |
| 用例描述 | 超级管理员对仓库普通管理人员进行信息修改，包括修改性别、名字、电话、密码 |
| 参与者 | 超级管理员 |
| 基本路径 | 1. 查看各管理员基本信息 2. 对管理员信息进行修改，包括性别、名字、电话、密码 |

表5-3仓库管理用例描述

|  |  |
| --- | --- |
| 用例名称 | 仓库管理 |
| 用例描述 | 对仓库进行管理，包括查询仓库、修改仓库属性、新增仓库、货位分区管理等功能 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.按仓库号、仓库名、负责人查询仓库  2.修改仓库属性，包括仓库号、仓库名、容量、负责人，仓库启用状态  3.删除所选仓库  4.创建新的仓库  5.将仓库根据货物类型分成多种区域，点击货位进入对应分区货物管理界面，可以对货物进行称重和计算总价 |

表5-4货物管理用例描述

|  |  |
| --- | --- |
| 用例名称 | 货物管理 |
| 用例描述 | 对物品进行管理，包括查询所有货物、入库、出库、计算库存余额功能 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 查询所有货物，包括按名称查询、仓库号查询、分区号查询 2. 对货物进行入库处理，包括直接入库和调拨入库 3. 对货物进行出库处理，包括直接出库和调拨出库 4. 计算仓库的库存余额 |

表5-5订单记录管理用例描述

|  |  |
| --- | --- |
| 用例名称 | 订单记录管理 |
| 用例描述 | 对出入库订单记录进行查询管理 |
| 参与者 | 超级管理员 |
| 基本路径 | 1.查看所有订单记录  2.按记录ID、货物名、仓库ID、出入库类型进行订单查询 |

表5-6物流运输用例描述

|  |  |
| --- | --- |
| 用例名称 | 物流运输 |
| 用例描述 | 显示各个仓库的地理位置，并提供地图导航；物流状态管理，发短信给接收者物流状态 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 显示各个仓库的具体位置 2. 选定出发地和目的地提供导航服务 3. 自动发短信给接收者目前的物流状态 |

表5-7图像识别用例描述

|  |  |
| --- | --- |
| 用例名称 | 图像识别 |
| 用例描述 | 对输入的图像进行智能识别，包含图像识别和订单识别 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 输入一个货物照片，进行智能识别，输出货物名称 2. 输入一个订单照片，进行智能识别，提取订单关键信息后输出 |

表5-8个人信息修改用例描述

|  |  |
| --- | --- |
| 用例名称 | 个人信息修改 |
| 用例描述 | 对个人信息进行修改 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.对个人信息进行修改，包括姓名、性别、联系方式等等 |

**人员管理用例图：**

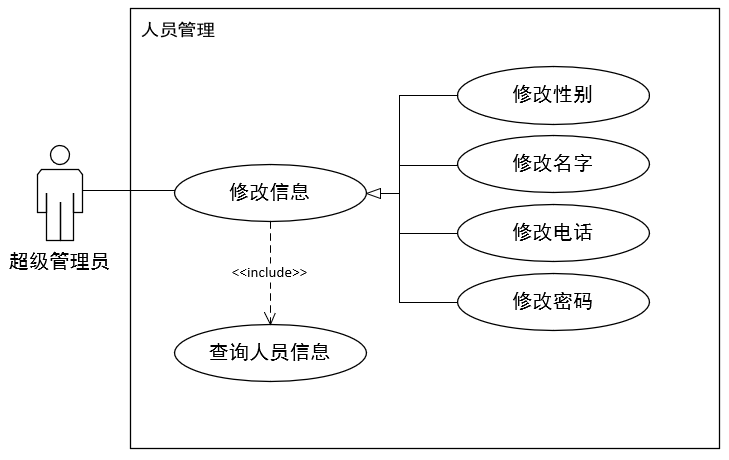


图5- 人员管理用例图

**用例描述 ：**

表5-9修改信息用例描述

|  |  |
| --- | --- |
| 用例名称 | 修改信息 |
| 用例描述 | 超级管理员对仓库普通管理人员进行信息修改，包括修改性别、名字、电话、密码 |
| 参与者 | 超级管理员 |
| 基本路径 | 1.查看各管理员基本信息，并列出所有信息  2.对管理员信息进行修改，包括性别、名字、电话、密码 |

**仓库管理用例图：**

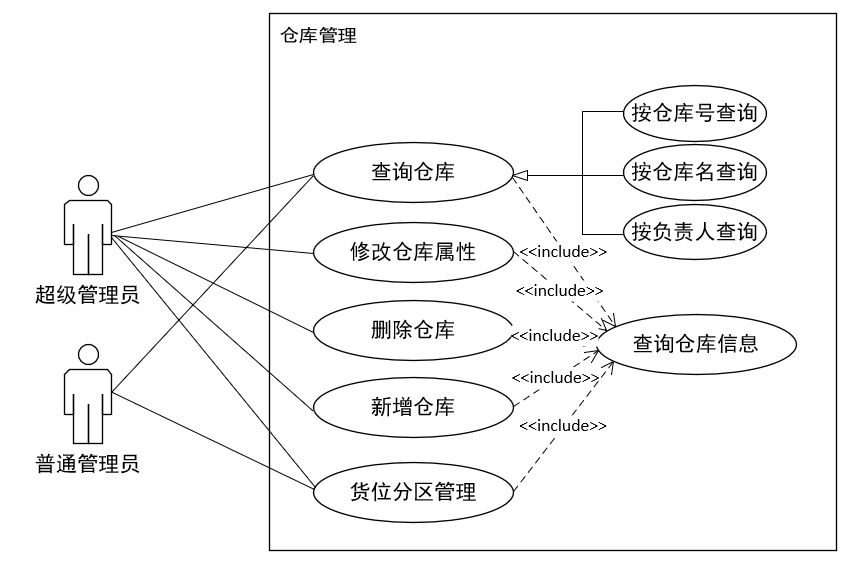


图5- 仓库管理用例图

**用例描述 ：**

表5-10查询仓库用例描述

|  |  |
| --- | --- |
| 用例名称 | 查询仓库 |
| 用例描述 | 查询所存在的仓库，可以按照仓库号、仓库名、负责人进行快速查询。 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 查询仓库信息并列出所有所存在的仓库供使用者查询 2. 按照仓库号进行快速查询 3. 按照仓库名进行快速查询 4. 按照负责人进行快速查询 |

表5-11修改仓库属性用例描述

|  |  |
| --- | --- |
| 用例名称 | 修改仓库属性 |
| 用例描述 | 使用者可以修改仓库的属性，包括：仓库号、仓库名、仓库容量、负责人、状态 |
| 参与者 | 超级管理员 |
| 基本路径 | 1. 查询仓库信息，列出所有存在的仓库供使用者修改 2. 修改仓库的属性，包括：仓库号、仓库名、仓库容量、负责人、状态 |

表5-12删除仓库用例描述

|  |  |
| --- | --- |
| 用例名称 | 删除仓库 |
| 用例描述 | 删除所选择的仓库 |
| 参与者 | 超级管理员 |
| 基本路径 | 1. 查询仓库信息，列出所有存在的仓库供使用者删除 2. 删除所选仓库 |

表5-13新增仓库用例描述

|  |  |
| --- | --- |
| 用例名称 | 新增仓库 |
| 用例描述 | 创建新的仓库，创建其属性，包括：仓库号、仓库名、仓库容量、负责人、状态 |
| 参与者 | 超级管理员 |
| 基本路径 | 1. 新增仓库，输入其属性，包括：仓库号、仓库名、仓库容量、负责人、状态 2. 查询仓库信息，查询有无所存在的仓库与新增仓库仓库号或仓库名重复 3. 无重复后创建成功 |

表5-14货位分区管理用例描述

|  |  |
| --- | --- |
| 用例名称 | 货位分区管理 |
| 用例描述 | 每个仓库拥有不同的货位分区，每个分区储存不同分类的货物，使用者对其进行管理 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 修改货物信息，包括：货物名称、货物货位、状态、数量、单价、重量 2. 删除所选货物 3. 计算所选择货物的总价 4. 计算所选择货物的总重 |

**货位分区管理用例图：**

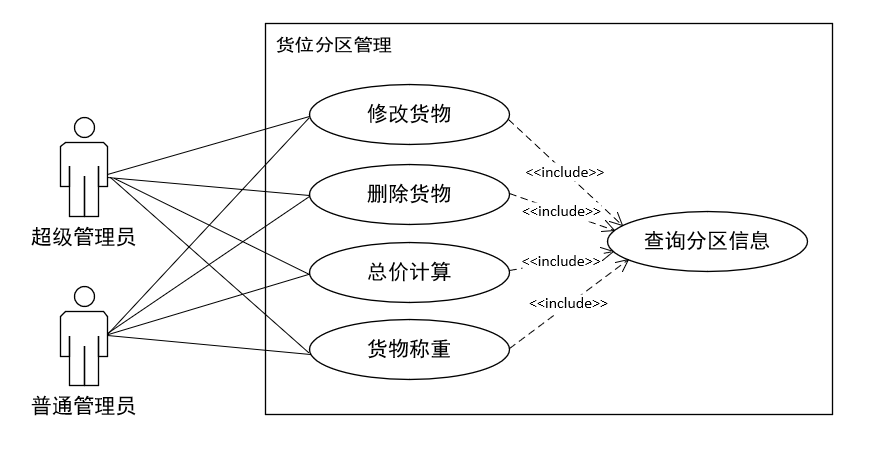


图5- 货位分区管理用例图

**用例描述 ：**

表5-15修改货物用例描述

|  |  |
| --- | --- |
| 用例名称 | 修改货物 |
| 用例描述 | 在货位分区管理界面，对所选货物的属性进行修改，包括：货物名称、货物货位、状态、数量、单价、重量 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.查询分区内货物信息  2.列出所有货物信息  3.修改所选货物属性，包括：货物名称、货物货位、状态、数量、单价、重量 |

表5-16删除货物用例描述

|  |  |
| --- | --- |
| 用例名称 | 删除货物 |
| 用例描述 | 在货位分区管理界面，对所选货物进行删除 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.查询分区内货物信息  2.列出所有货物信息  3.删除所选货物 |

表5-17总价计算用例描述

|  |  |
| --- | --- |
| 用例名称 | 总价计算 |
| 用例描述 | 选中所想计算价格的货物，进行货物总价计算 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.查询分区内货物信息  2.列出所有货物信息  3.选中所想计算价格的货物，进行货物总价计算 |

表5-18货物称重用例描述

|  |  |
| --- | --- |
| 用例名称 | 货物称重 |
| 用例描述 | 选中所想称重的货物，进行货物总重量计算 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.查询分区内货物信息  2.列出所有货物信息  3.选中所想称重的货物，进行货物总重量计算 |

**货物管理用例图：**

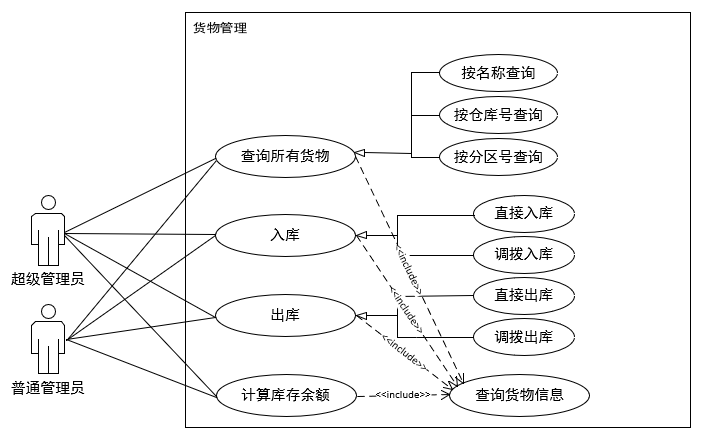


图5- 货物管理用例图

**用例描述 ：**

表5-19查询所有货物用例描述

|  |  |
| --- | --- |
| 用例名称 | 查询所有货物 |
| 用例描述 | 查询所有库存货物，超级管理员可以查询所有仓库货物，普通管理员查询自己所属仓库货物。可以按货物名称、仓库号、分区号查询。 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 查询仓库货物信息 2. 列出所有货物信息 3. 按货物名称进行快速查询 4. 按货物号进行快速查询 5. 按分区号进行快速查询 |

表5-20入库用例描述

|  |  |
| --- | --- |
| 用例名称 | 入库 |
| 用例描述 | 输入入库货物的各项属性进行入库，入库方式包括直接入库、调拨入库 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 输入入库货物的各项属性进行入库 2. 判断库存容量是否充足 3. 库存充足后，成功入库并生成订单 |

表5-21出库用例描述

|  |  |
| --- | --- |
| 用例名称 | 出库 |
| 用例描述 | 列出仓库所有货物信息，选择货物进行出库，出库方式包括直接出库、调拨出库。如遇调拨出库，对方库存允许情况下执行双方仓库出入库操作 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 查询仓库信息 2. 列出仓库所有货物信息 3. 选择货物进行出库，并选择出库方式 4. 如选择直接出库，成功出库并生成订单 5. 如选择调拨出库，判断对方仓库库存是否充足 6. 若库存充足，成功出库，对方仓库执行入库操作，并生成订单 |

表5-22计算库存余额用例描述

|  |  |
| --- | --- |
| 用例名称 | 计算库存余额 |
| 用例描述 | 计算仓库库存余额 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1.计算仓库库存余额 |

**订单记录管理用例图：**

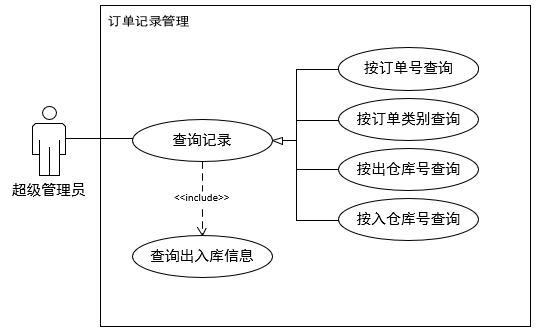


图5- 订单记录管理用例图

**用例描述 ：**

表5-23查询记录用例描述

|  |  |
| --- | --- |
| 用例名称 | 查询记录 |
| 用例描述 | 查询仓库中所有出入库信息记录，并可以按订单号、订单类别、出仓库号、入仓库号进行快速查询 |
| 参与者 | 超级管理员 |
| 基本路径 | 1. 查询出入库信息 2. 列出所有出入库信息 3. 按订单号、订单类别、出仓库号、入仓库号进行快速查询 |

**物流运输用例图：**

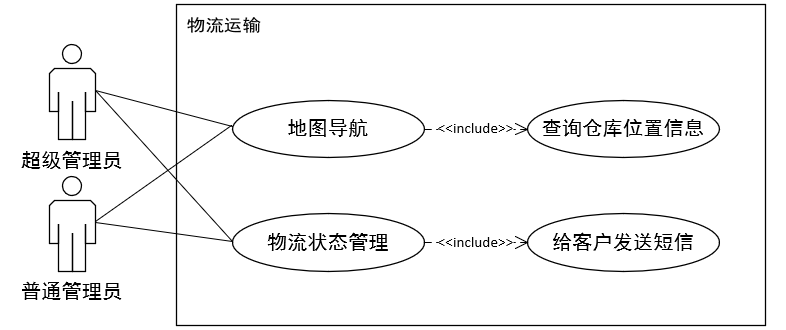


图5- 物流运输用例图

**用例描述 ：**

表5-24地图导航用例描述

|  |  |
| --- | --- |
| 用例名称 | 地图导航 |
| 用例描述 | 查询各个仓库位置信息并显示在地图上，选定出发地和目的地提供路径规划导航服务 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 查询各个仓库位置信息 2. 显示各个仓库在地图上的位置 3. 选定出发地和目的地 4. 提供路线规划导航服务 |

表5-25物流状态管理用例描述

|  |  |
| --- | --- |
| 用例名称 | 物流状态管理 |
| 用例描述 | 货物发货、已到货、签收时各发送短信给接收商 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 货物发货时发送短信提醒 2. 货物到货时发送短信提醒 3. 货物签收时发送短信提醒 |

**图像识别用例图：**

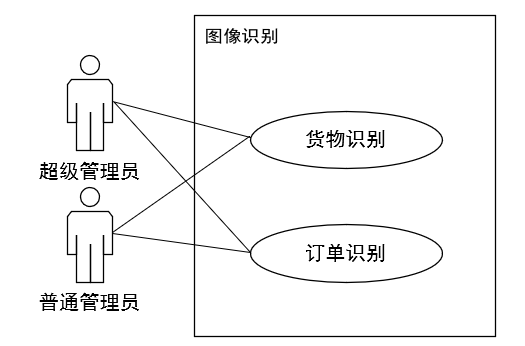


图5- 图像识别用例图

**用例描述 ：**

表5-26货物识别用例描述

|  |  |
| --- | --- |
| 用例名称 | 货物识别 |
| 用例描述 | 输入货物照片，调用百度api进行智能识别，输出货物名称信息 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 输入货物照片 2. 调用百度api进行识别 3. 识别货物结果信息进行输出 |

表5-27订单识别用例描述

|  |  |
| --- | --- |
| 用例名称 | 订单识别 |
| 用例描述 | 输入订单照片，调用百度api进行智能识别，提取订单关键信息后输出结果 |
| 参与者 | 超级管理员、普通管理员 |
| 基本路径 | 1. 输入订单照片 2. 调用百度api进行识别 3. 提取订单关键信息后进行输出 |

5.2.2 顺序图与协作图

采用顺序图和协作图，描述按照时间的先后顺序对象之间交互动作过程，展示了对象之间的消息传递和协作方式。

* **账号登录**
  + 顺序图

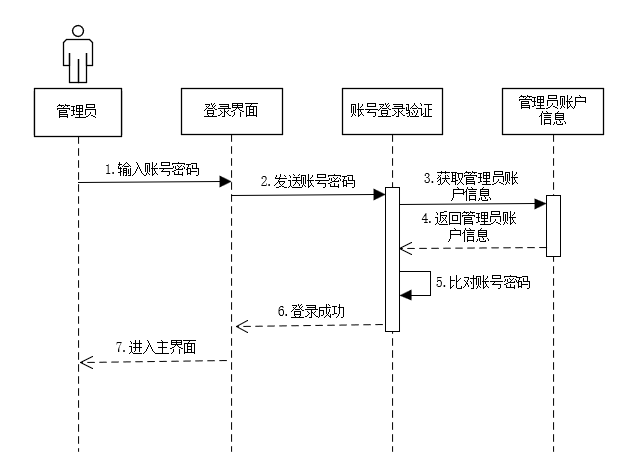


图5- 账号登录顺序图

* + 协作图

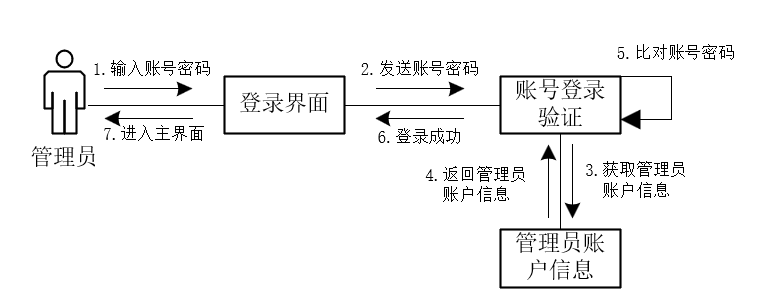


图5- 账号登录协作图

* **人员管理**
  + 顺序图

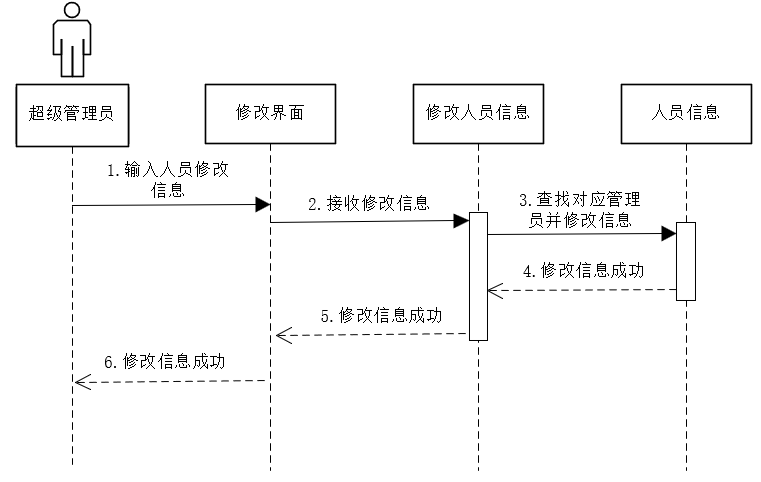


图5- 人员管理顺序图

* + 协作图

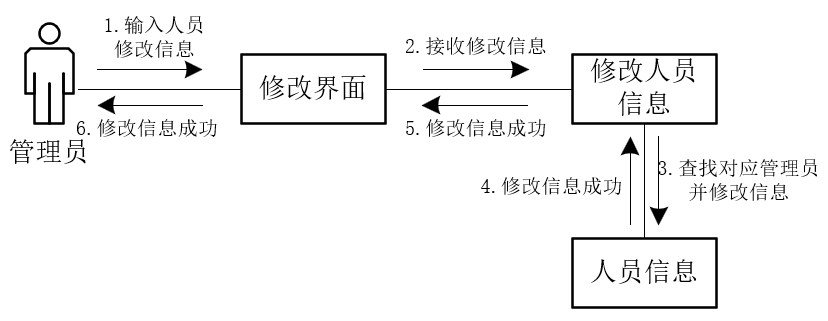


图5- 人员管理协作图

* **仓库管理-查询仓库**
  + 顺序图

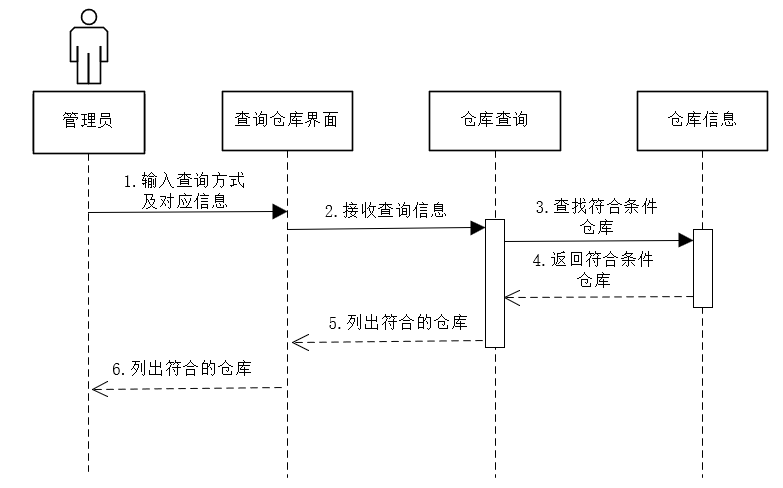


图5- 仓库管理-查询仓库顺序图

* + 协作图

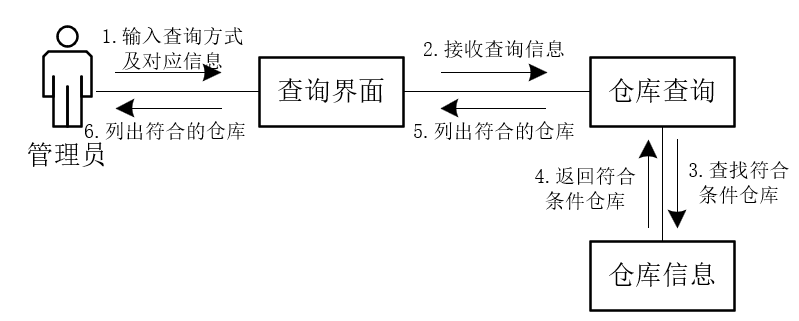


图5- 仓库管理-查询仓库协作图

* **仓库管理-修改仓库属性**
  + 顺序图

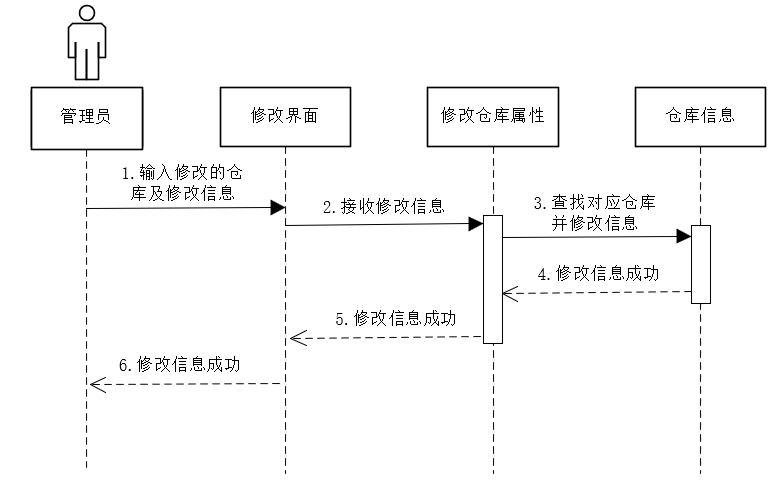


图5- 仓库管理-修改仓库属性顺序图

* + 协作图

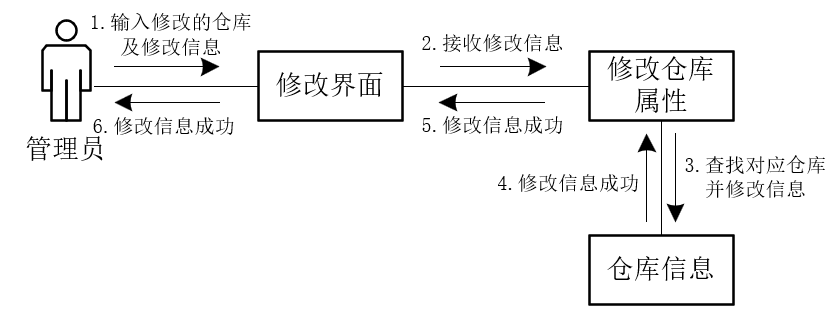


图5- 仓库管理-修改仓库属性协作图

* **仓库管理-删除仓库**
  + 顺序图

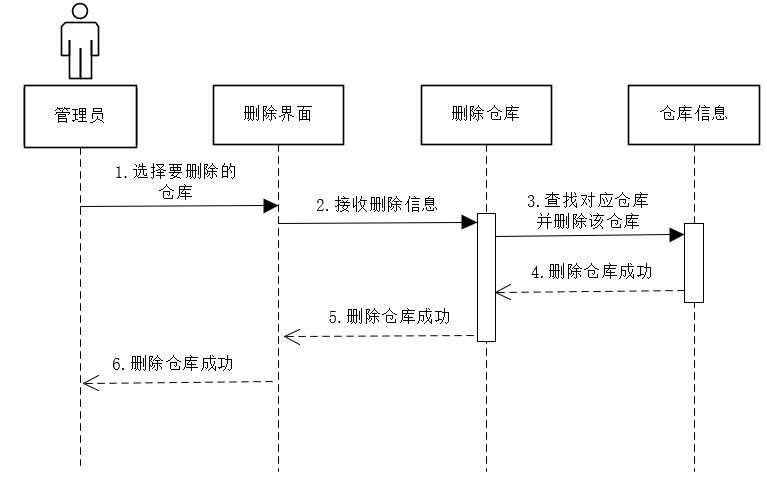


图5- 仓库管理-删除仓库顺序图

* + 协作图

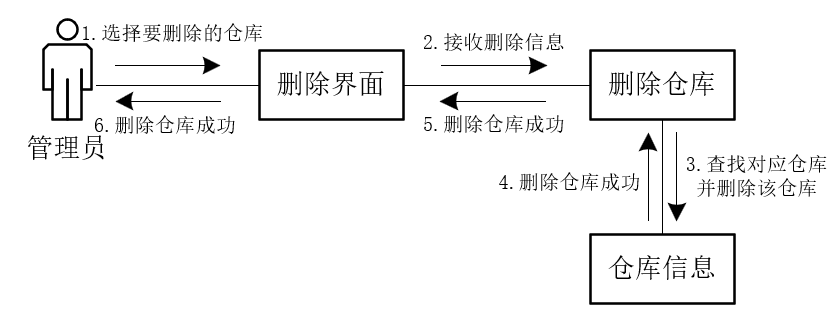


图5- 仓库管理-删除仓库协作图

* **仓库管理-新增仓库**
  + 顺序图

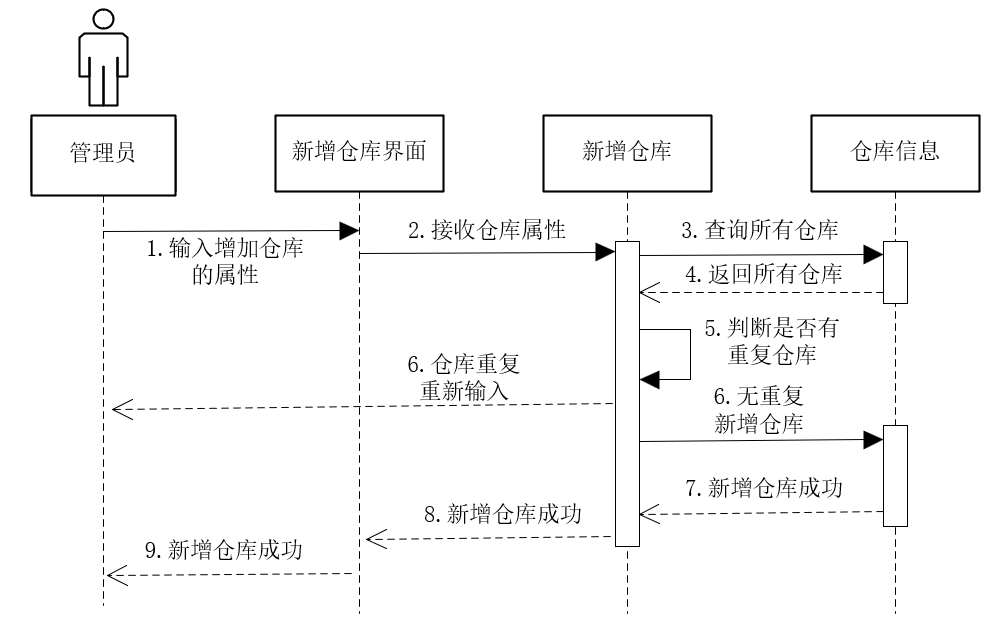


图5- 仓库管理-新增仓库顺序图

* + 协作图

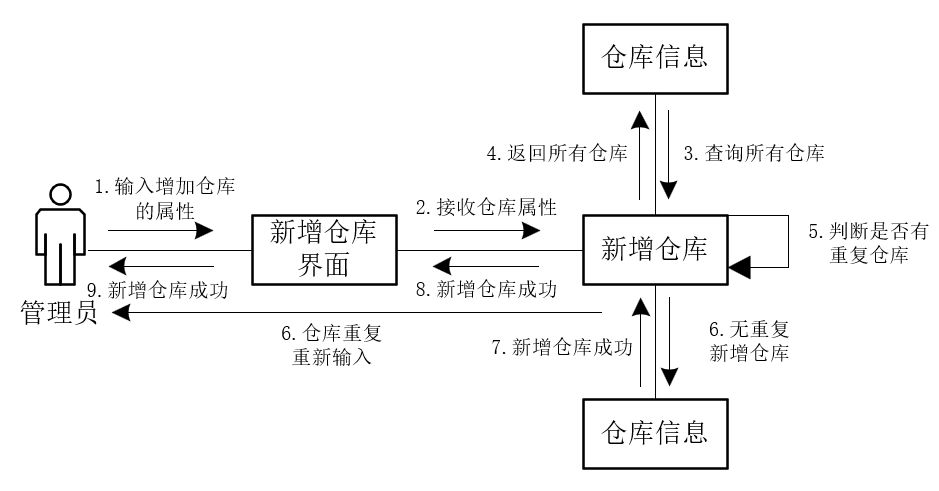


图5- 仓库管理-新增仓库协作图

* **仓库管理-货位分区管理-修改货物**
  + 顺序图

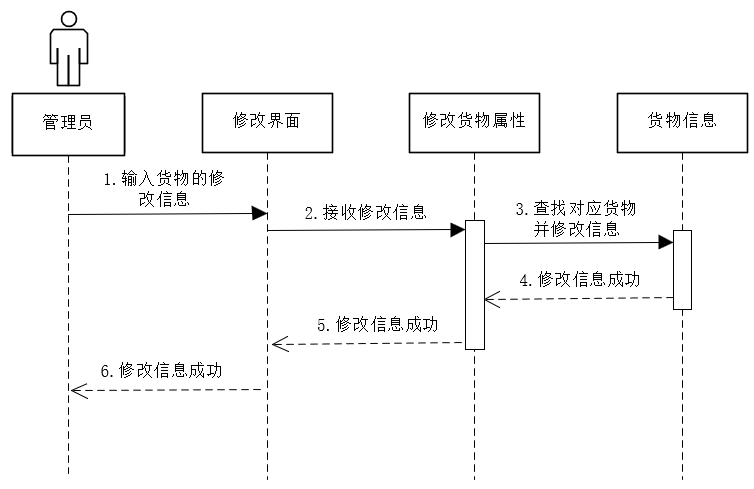


图5- 仓库管理-货位分区管理-修改货物顺序图

* + 协作图

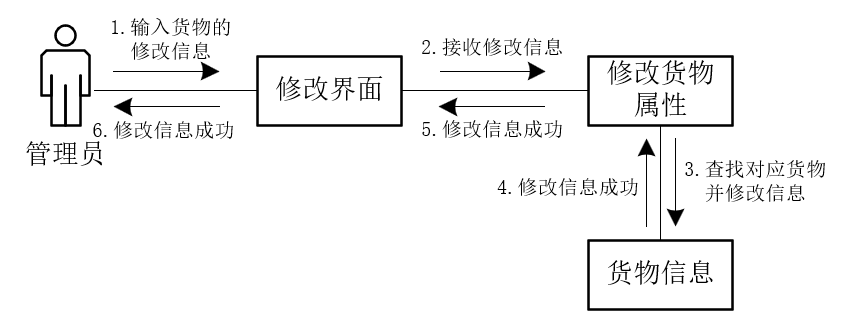


图5- 仓库管理-货位分区管理-修改货物协作图

* **仓库管理-货位分区管理-删除货物**
  + 顺序图

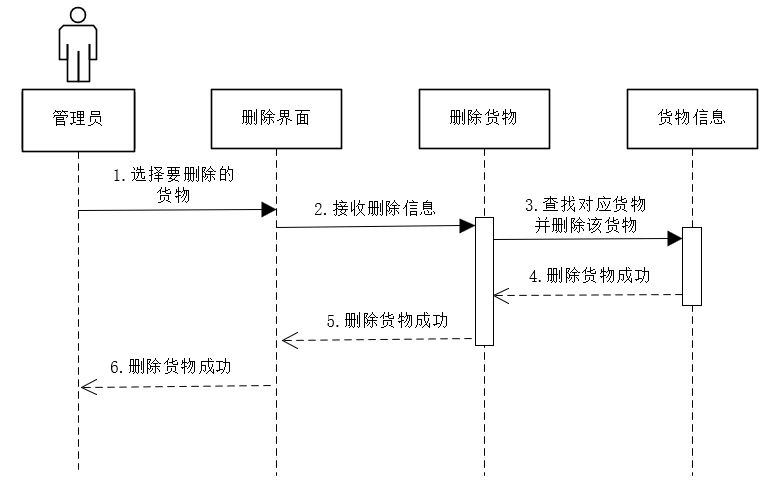


图5- 仓库管理-货位分区管理-删除货物顺序图

* + 协作图

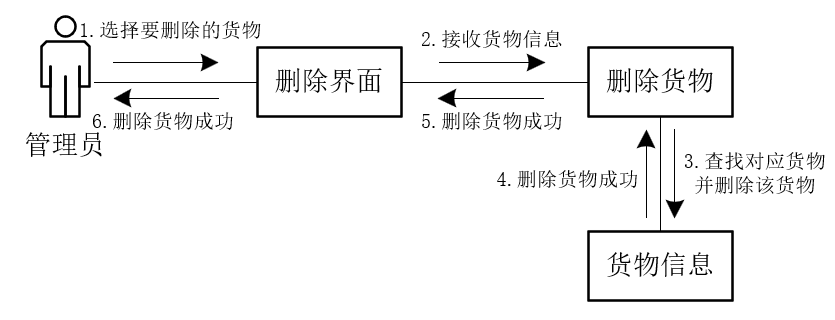


图5- 仓库管理-货位分区管理-删除货物协作图

* **仓库管理-货位分区管理-总价计算**
  + 顺序图

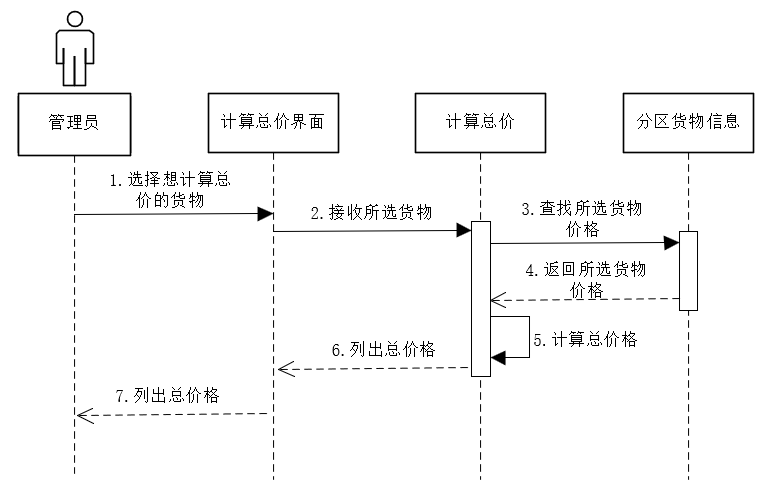


图5- 仓库管理-货位分区管理-总价计算顺序图

* + 协作图

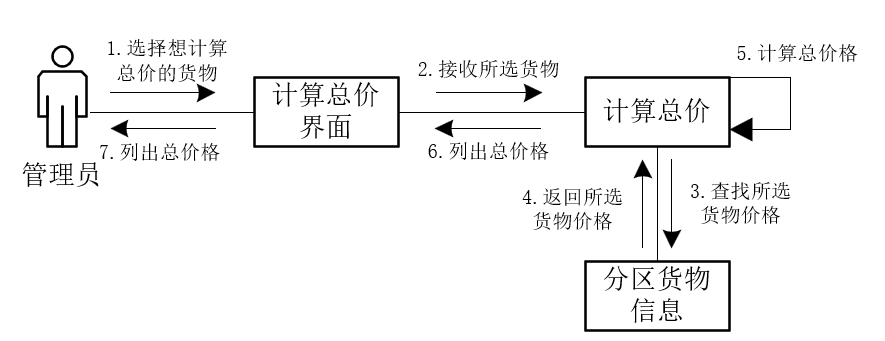


图5- 仓库管理-货位分区管理-总价计算协作图

* **仓库管理-货位分区管理-货物称重**
  + 顺序图

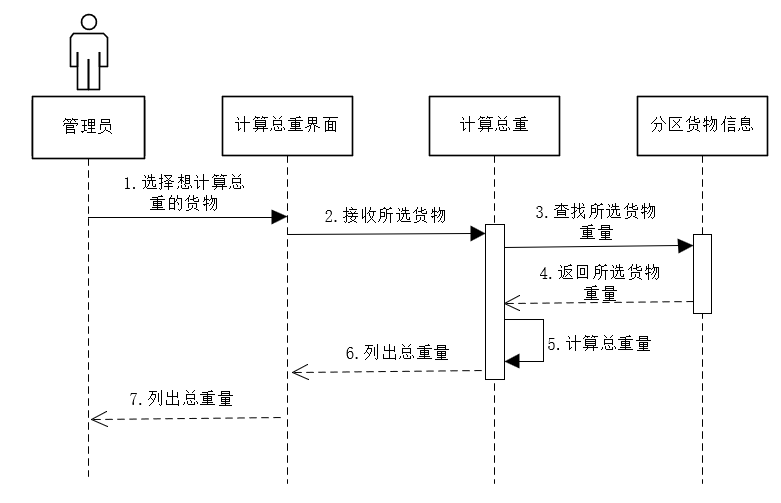


图5- 仓库管理-货位分区管理-货物称重顺序图

* + 协作图

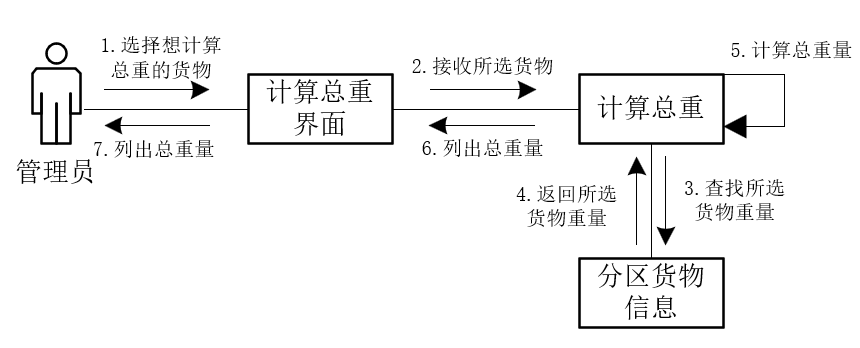


图5- 仓库管理-货位分区管理-货物称重协作图

* **货物管理-查询所有货物**
  + 顺序图

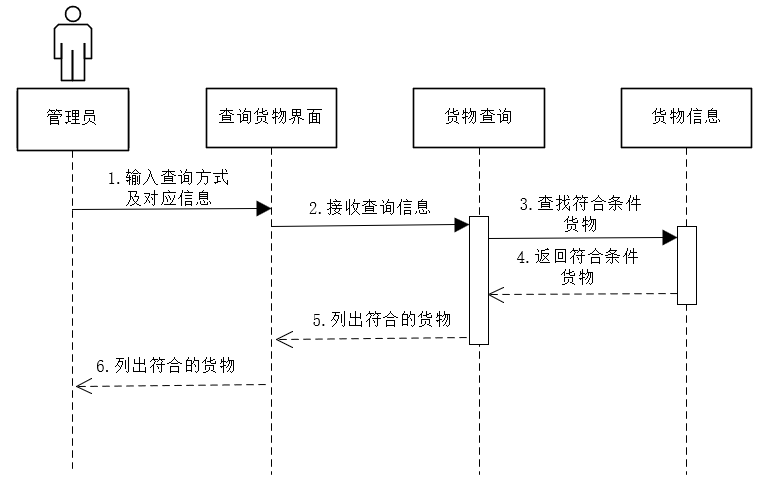


图5- 货物管理-查询所有货物顺序图

* + 协作图

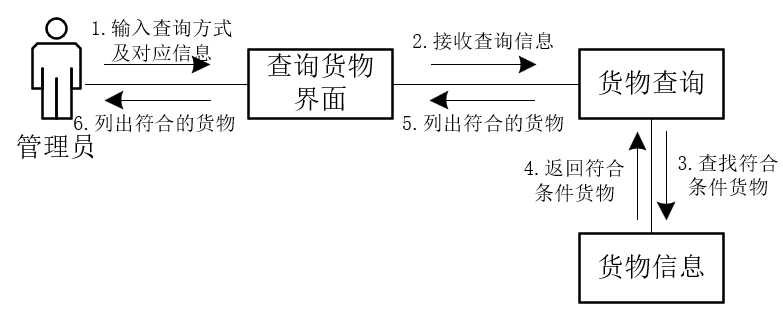


图5- 货物管理-查询所有货物协作图

* **货物管理-入库**
  + 顺序图

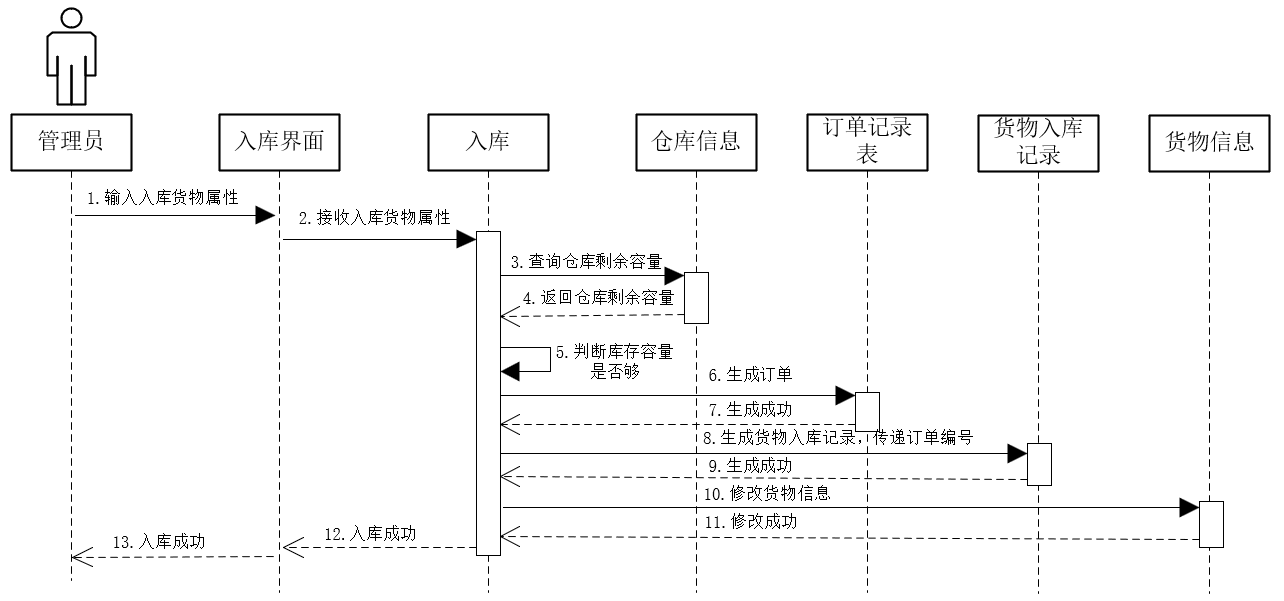


图5- 货物管理-入库顺序图

* + 协作图

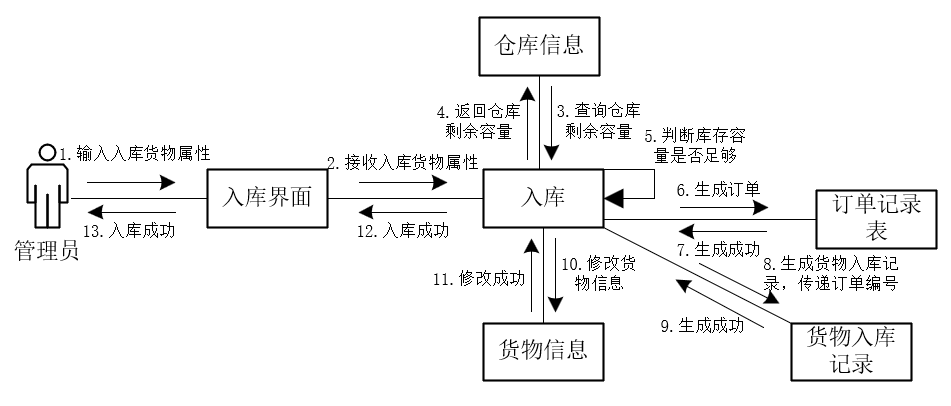


图5- 货物管理-入库协作图

* **货物管理-出库-直接出库**
  + 顺序图

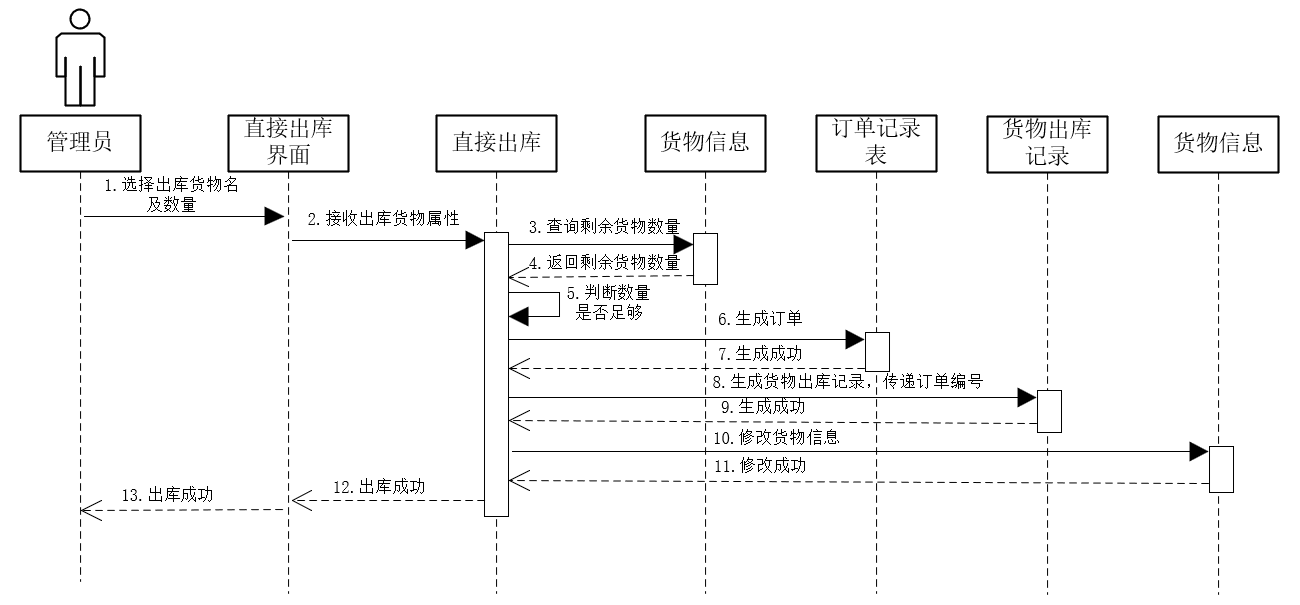


图5- 货物管理-出库-直接出库顺序图

* + 协作图

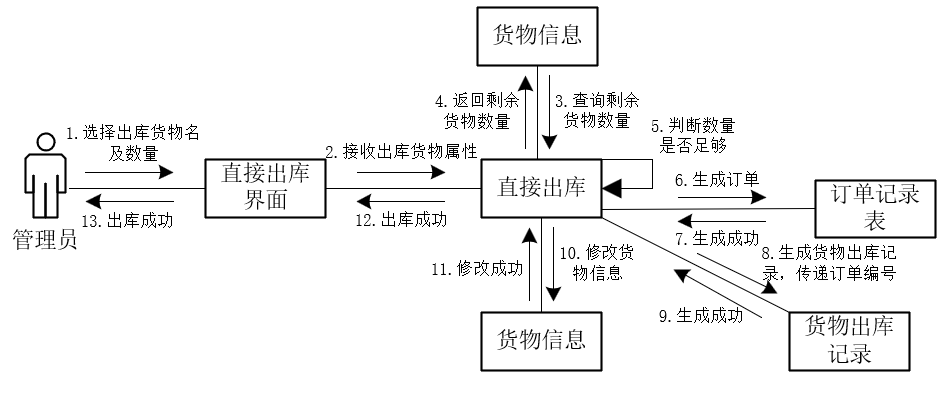


图5- 货物管理-出库-直接出库协作图

* **货物管理-出库-调拨出库**
  + 顺序图



图5- 货物管理-出库-调拨出库顺序图

* + 协作图

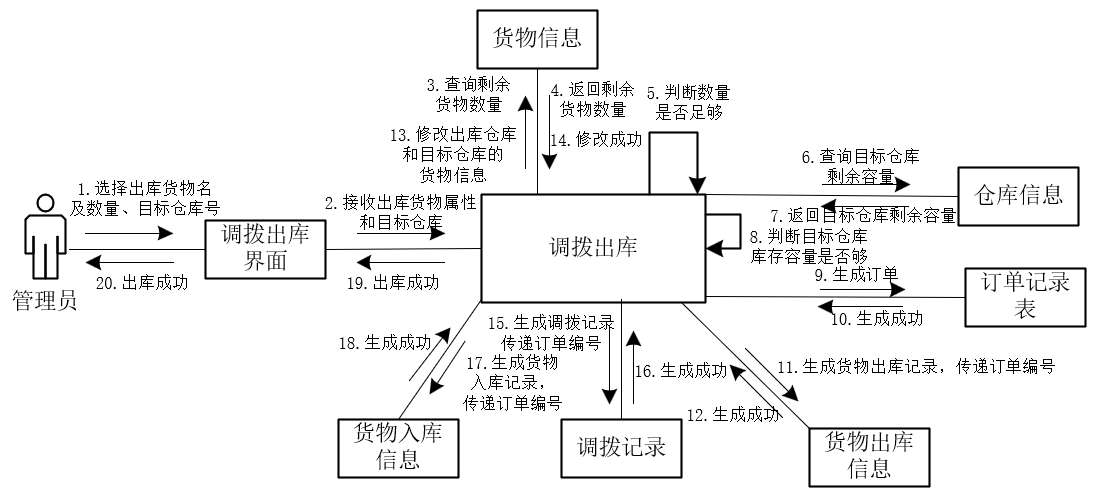


图5- 货物管理-出库-调拨出库协作图

* **货物管理-计算库存余额**
  + 顺序图

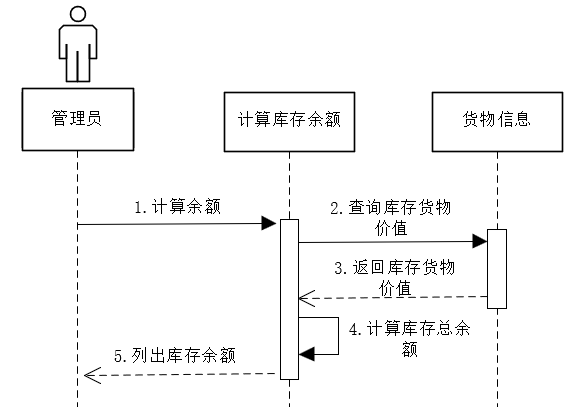


图5- 货物管理-计算库存余额顺序图

* + 协作图

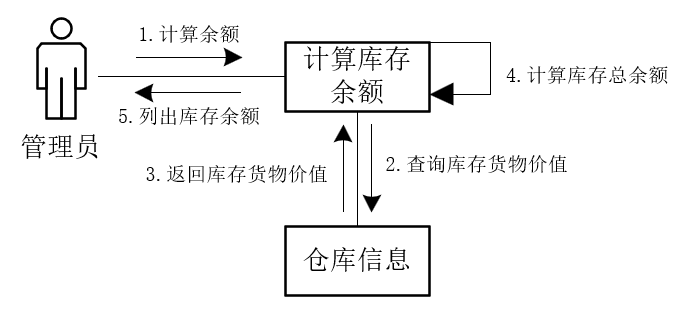


图5- 货物管理-计算库存余额协作图

* **订单记录管理**
  + 顺序图

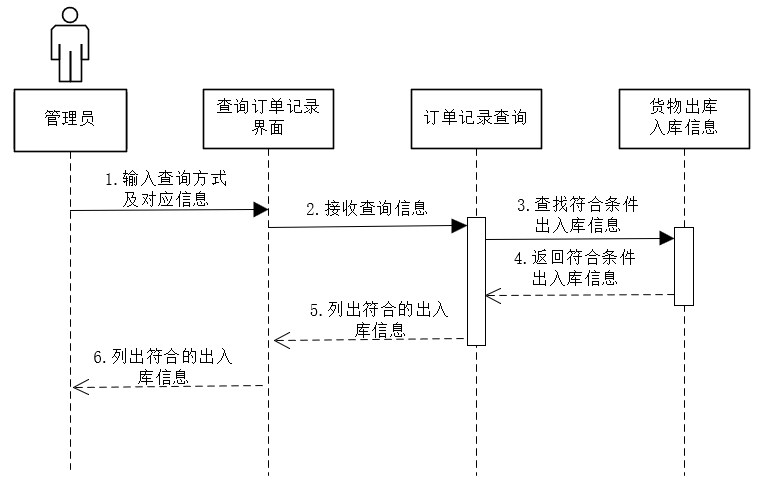


图5- 订单记录管理顺序图

* + 协作图

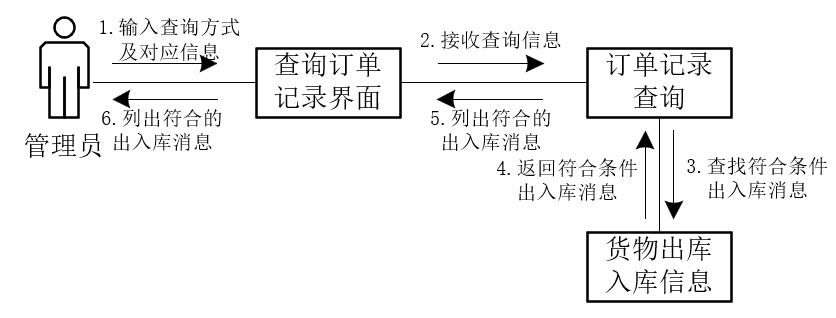


图5- 订单记录管理协作图

* **物流运输-地图导航**
  + 顺序图

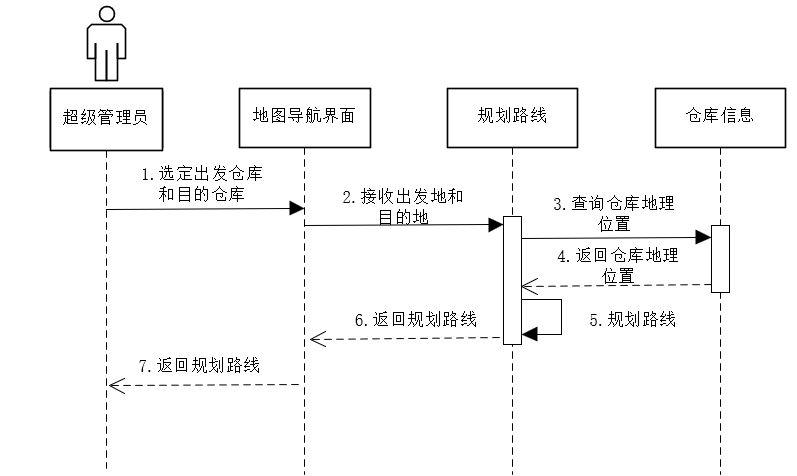


图5- 物流运输-地图导航顺序图

* + 协作图

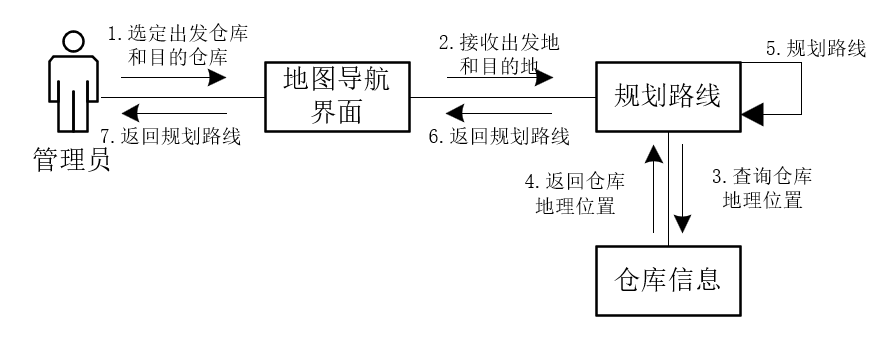


图5- 物流运输-地图导航协作图

* **物流运输-物流状态管理**
  + 顺序图

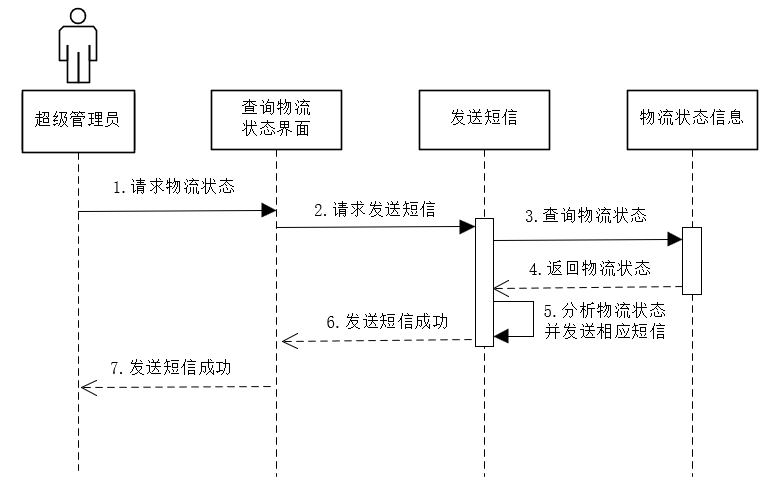


图5- 物流运输-物流状态管理顺序图

* + 协作图

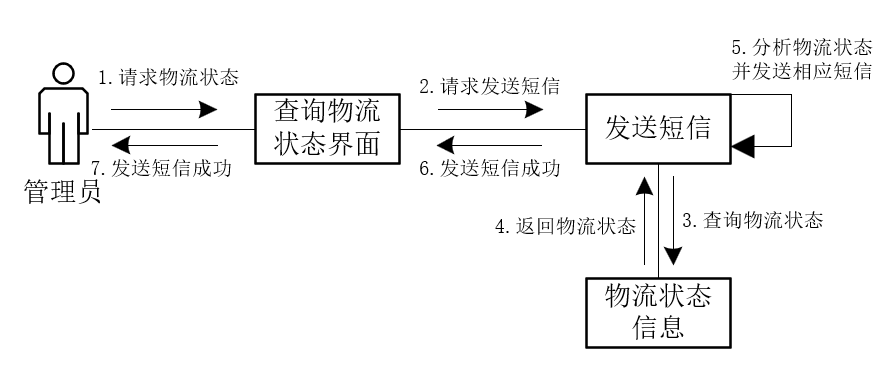


图5- 物流运输-物流状态管理协作图

* **图像识别-货物识别**
  + 顺序图

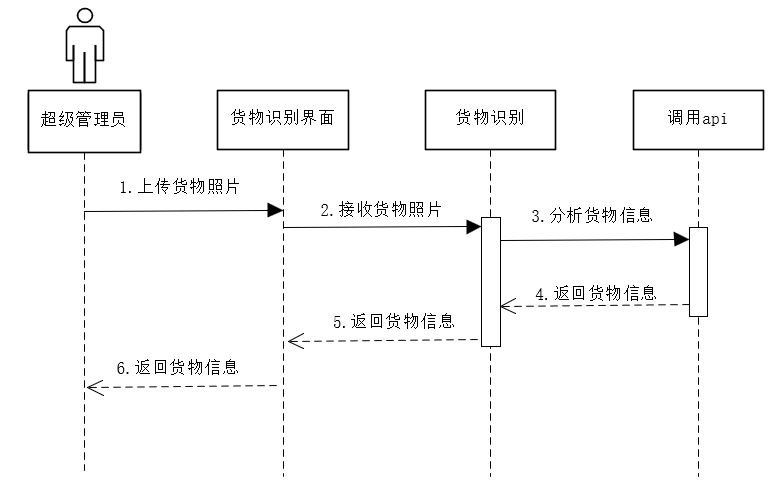


图5- 图像识别-货物识别顺序图

* + 协作图

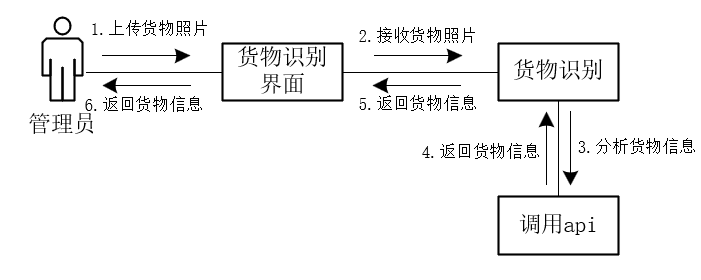


图5- 图像识别-货物识别协作图

* **图像识别-订单识别**
  + 顺序图

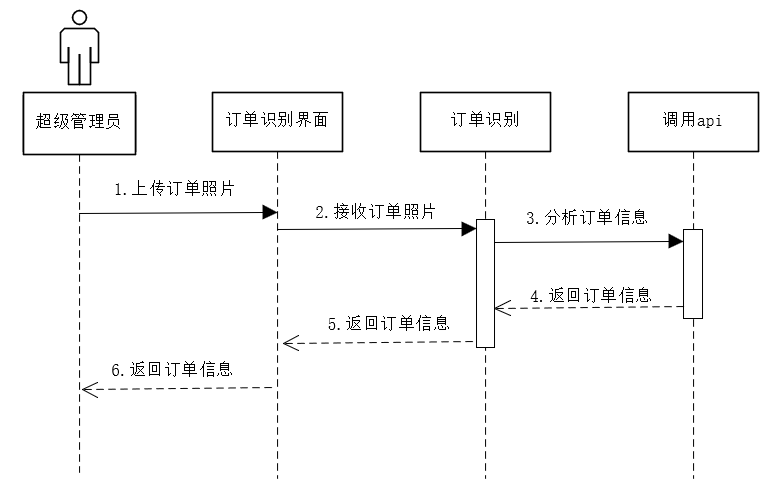


图5- 图像识别-订单识别顺序图

* + 协作图

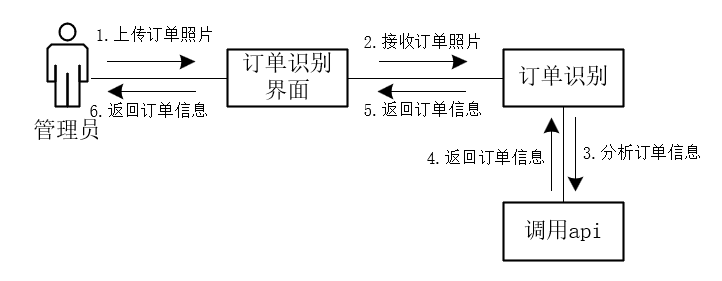


图5- 图像识别-订单识别协作图

## 5.3属性与方法：

详细列出每个类的属性（数据成员）和方法（操作）



图5-53 入库订单

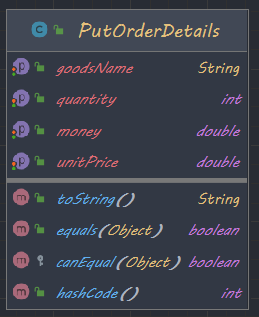


图5-54 入库订单细节

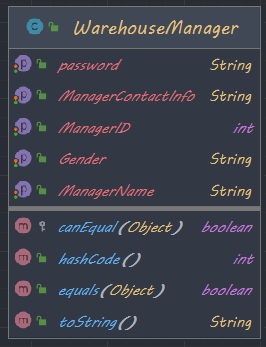


图5-55 仓库管理员

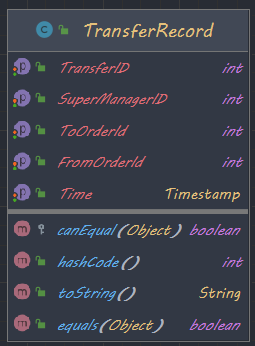


图5-56 调拨记录



图5-57超级管理员



图5-58订单细节

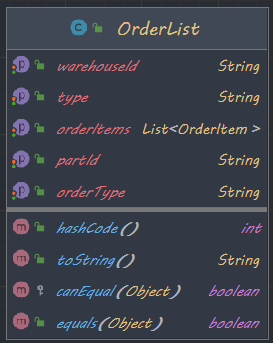


图5-59订单列表

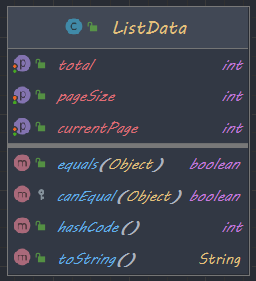


图5-60分页数据

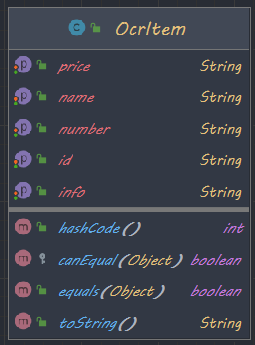


图5-61识别细节

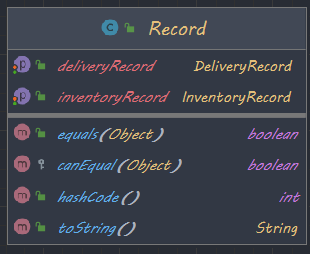


图5-62订单记录

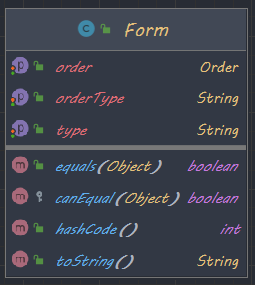


图5-63表单数据

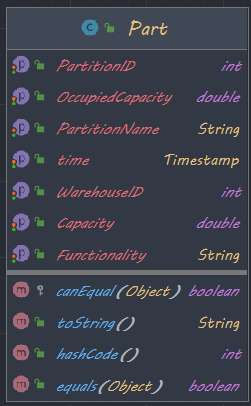


图5-64分区数据



图5-65订单数据

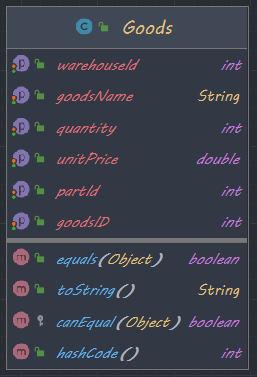


图5-66货物数据



图5-67订单列表

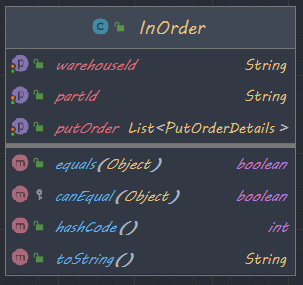


图5-68入库订单

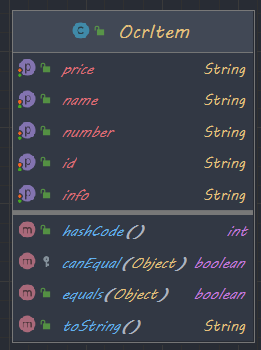


图5-69识别数据细节



图5-70识别数据

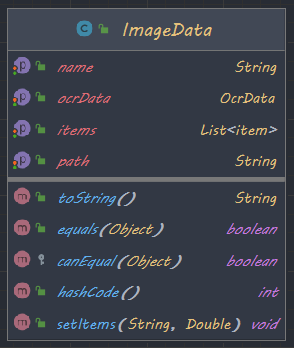


图5-71图像数据



图5-72出库记录



图5-73调拨数据

# 面向对象的设计说明书

## 6.1设计原则：

遵循的设计模式、原则和指导方针，如单一职责原则、开闭原则等

单一职责原则（Single Responsibility Principle, SRP）：一个类应该只有一个引起它变化的原因。换句话说，一个类应该只负责一项职责。

开闭原则（Open-Closed Principle, OCP）：软件实体（类、模块、函数等）应该对扩展开放，对修改关闭。这意味着软件实体的行为可以扩展，但是在修改已有的代码时应该尽量避免。

里氏替换原则（Liskov Substitution Principle, LSP）：子类型必须能够替换它们的基类型。这意味着如果一个程序使用基类的地方都可以使用其子类，那么该程序可以接受子类替换基类而不会产生任何错误或异常。

接口隔离原则（Interface Segregation Principle, ISP）：客户端不应该依赖它不需要的接口。一个类对另一个类的依赖应该建立在最小的接口上。

依赖倒置原则（Dependency Inversion Principle, DIP）：高层模块不应该依赖低层模块，两者都应该依赖抽象。抽象不应该依赖细节，细节应该依赖抽象。

## 6.2类设计：

详细介绍每个类的设计细节，包括继承层次、封装接口、实现方式

在本次的课程设计，我们采用了SpingBoot作为我们的后端框架，因此遵循了domain-dao-service-controller的层次架构。

Domain Layer：业务逻辑层，负责处理业务逻辑。这个层应该包含数据实体（Domain Objects）和业务逻辑方法（Domain Logic）。

Data Access Layer：数据访问层，负责与数据库进行交互。这个层应该包含数据访问对象（Data Access Objects, DAOs）和它们所使用的数据库操作（Repository）。

Service Layer：服务层，负责处理服务层。这个层应该包含服务接口（Service Interfaces）和实现这些接口的服务类（Service Implementations）。

Controller Layer：控制器层，负责处理HTTP请求。这个层应该包含处理HTTP请求的控制器（HTTP Controller）和它们所使用的服务（Services）。

其中Domain中的每一个类对应数据库中的一张表，这也和我们所学的面向对象设计所相符合；Dao中的每一个Dao和Domain中的每一个类相对应，作为接口提供对应的数据库操作方法，这里使用了MybaitsPlus作为本次课程设计的数据库框架；Service则提供对数据的进一步封装操作，更复杂；而Controller则和前端直接交互，进行数据的传输和处理。

**Domain：**

│─DeliveryRecord.java

│─Goods.java

│─ImageData.java

│─InventoryRecord.java

│─JsonResult.java

│─ListData.java

│─Orders.java

│─Part.java

│─Record.java

│─SuperManager.java

│─TransferRecord.java

│─Warehouse.java

│─WarehouseManager.java

│

└─dataHandle

Form.java

InOrder.java

OcrData.java

OcrItem.java

Order.java

OrderItem.java

OrderList.java

PutOrderDetails.java

QRCode.java

SMS.java

**Dao：**

│─DeliveryRecordDao.java

│─GoodsDao.java

│─InventoryRecordDao.java

│─OrdersDao.java

│─SuperManagerDao.java

│─TransferRecordDao.java

│─WarehouseDao.java

└─WarehouseManagerDao.java

Service：

│─DataHandleService.java

│─GoodsService.java

│─OrderService.java

│─SuperManagerService.java

│─TransferService.java

│─WarehouseManagerService.java

│

└─ocr

│─Client.java

│─Image.java

│─OCR.java

│

└─impl

│─Imageimpl.java

└─OCRimpl.java

**Controller：**

│─GoodsController.java

│─OrderController.java

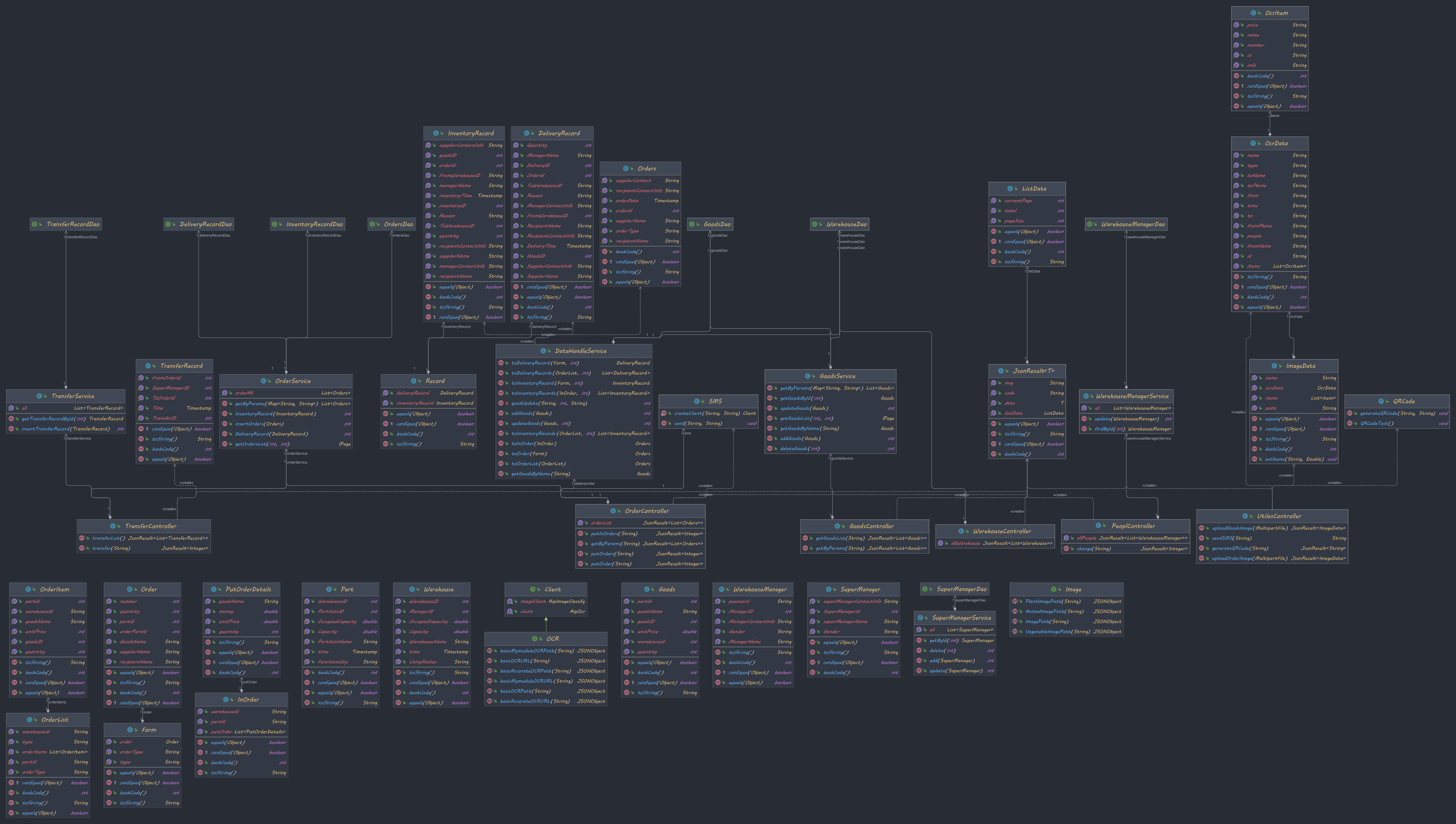
│─PeoplController.java

│─TransferController.java

│─UtilesController.java

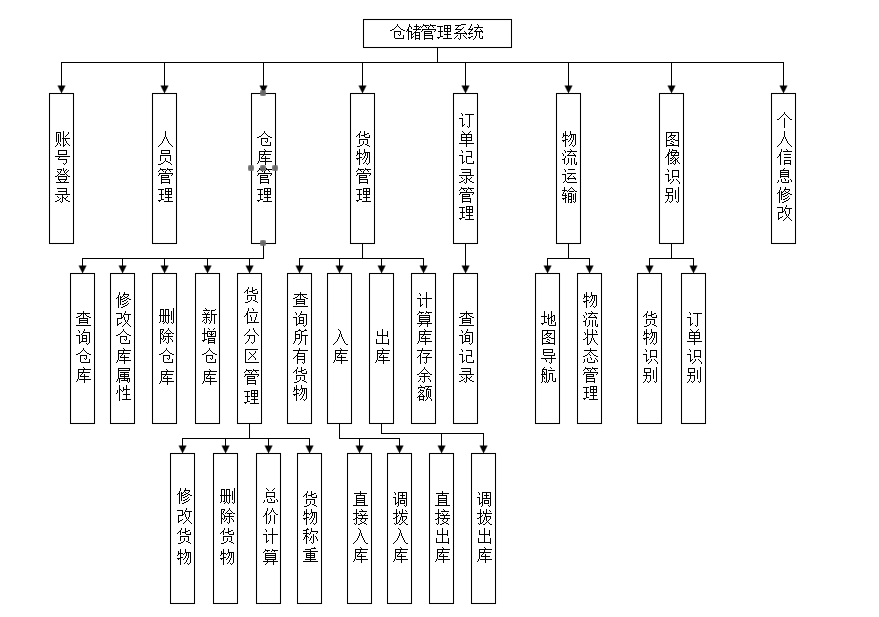
└─WarehouseController.java

其中各个类的详细设计和继承层次、封装接口、实现方式等见附件-系统类图，如下所示：



## 6.3架构设计：

整体系统架构图如下：



## 6.4数据库设计：

表结构、ER图以及数据库操作的设计

1. 关系模式设计：

SuperManager（SuperManagerID , CreateTime , UpdateTime , Gender , superManagerName , superManagerContactInfo , PWD）

WarehouseManager（ManagerID, CreateTime, UpdateTime, Gender, ManagerName, ManagerContactInfo, PWD）

Warehouse（WarehouseID, ManagerID, WarehouseName, Capacity, OccupiedCapacity, CreateTime, UpdateTime, UsingStatus）

Partition（PartitionID, WarehouseID, PartitionName, Functionality, Capacity, OccupiedCapacity, CreateTime, UpdateTime）

Goods（GoodsID, WarehouseID, PartitionID, GoodsName, Quantity, UnitPrice, CreateTime, UpdateTime）

Order（OrderNumber, RecipientName, RecipientContactInfo, SupplierName, SupplierContact, OrderType, OrderDate）

ManagerWarehouse（ManagerID, WarehouseID）

Inventory（InventoryID, OrderNumber, Quantity, InventoryTime, ToWarehouseID, Reason）

Delivery（DeliveryID, OrderNumber, GoodsID, Quantity, DeliveryTime, FromWarehouseID, Reason）

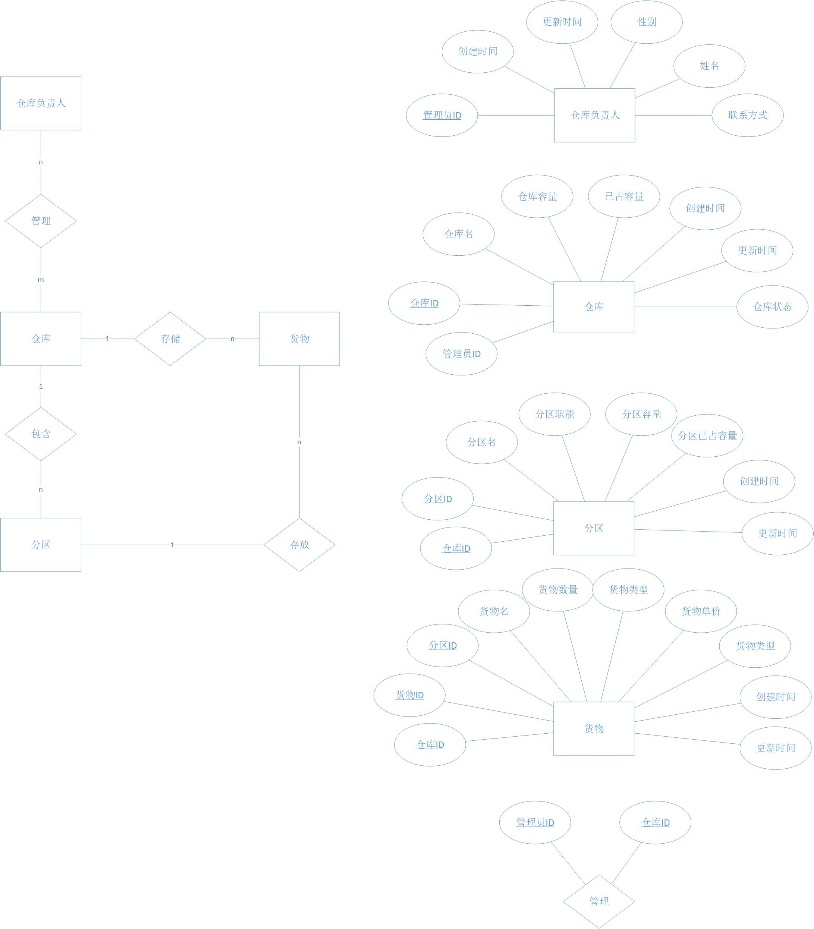
Transfer（TransferID, FromOrderNumber, ToOrderNumber, SuperManagerID, CreateTime, UpdateTime）

1. 数据类型定义

* SuperManager（超级管理员）
  + SuperManagerID（主键，INT类型）
  + CreateTime（创建时间，DATE类型，NOT NULL）
  + UpdateTime（更新时间，DATE类型，NOT NULL）
  + Gender（性别，VARCHAR(10)类型）
  + superManagerName（姓名，VARCHAR(50)类型）
  + superManagerContactInfo（联系方式，VARCHAR(20)类型）
  + PWD（密码，longtext类型）
* WarehouseManager（仓库负责人）
  + ManagerID（主键，INT类型）
  + CreateTime（创建时间，DATE类型，NOT NULL）
  + UpdateTime（更新时间，DATE类型，NOT NULL）
  + Gender（性别，VARCHAR(10)类型）
  + ManagerName（姓名，VARCHAR(50)类型）
  + ManagerContactInfo（联系方式，VARCHAR(20)类型）
  + PWD（密码，longtext类型）
* Warehouse（仓库）
  + WarehouseID（主键，INT类型）
  + ManagerID（外键，INT类型，关联WarehouseManager表的ManagerID字段）
  + WarehouseName（仓库名称，VARCHAR(50)类型）
  + Capacity（最大容量，DECIMAL(10,2)类型）
  + OccupiedCapacity（已占用容量，DECIMAL(10,2)类型）
  + CreateTime（创建时间，DATE类型，NOT NULL）
  + UpdateTime（更新时间，DATE类型，NOT NULL）
  + UsingStatus（使用状态，ENUM('启用', '停用')类型）
* Partition（分区）
  + PartitionID（主键，INT类型）
  + WarehouseID（外键，INT类型，关联Warehouse表的WarehouseID字段）
  + PartitionName（分区名称，VARCHAR(50)类型）
  + Functionality（功能或用途，VARCHAR(50)类型）
  + Capacity（最大容量，DECIMAL(10,2)类型）
  + OccupiedCapacity（已占用容量，DECIMAL(10,2)类型）
  + CreateTime（创建时间，DATE类型，NOT NULL）
  + UpdateTime（更新时间，DATE类型，NOT NULL）
* Goods（货物）
  + GoodsID（主键，INT类型）
  + WarehouseID（外键，INT类型，关联Warehouse表的WarehouseID字段）
  + PartitionID（外键，INT类型，关联Part表的PartitionID字段）
  + GoodsName（货物名称，VARCHAR(50)类型）
  + Quantity（数量，INT类型）
  + UnitPrice（单价，DECIMAL(10,2)类型）
  + CreateTime（创建时间，DATE类型，NOT NULL）
  + UpdateTime（更新时间，DATE类型，NOT NULL）
* Order（订单）
  + OrderNumber（主键，INT类型，AUTO\_INCREMENT）
  + RecipientName（接受方姓名，VARCHAR(50)类型）
  + RecipientContactInfo（接受方联系方式，VARCHAR(20)类型）
  + SupplierName（供应商名称，VARCHAR(255)类型，NOT NULL）
  + SupplierContact（供应商联系方式，VARCHAR(255)类型，NOT NULL）
  + OrderType（订单类型信息，longtext类型）
  + OrderDate（订单创建日期，DATE类型）
* ManagerWarehouse（负责人仓库）
  + ManagerID（外键，INT类型，关联WarehouseManager表的ManagerID字段）
  + WarehouseID（外键，INT类型，关联Warehouse表的WarehouseID字段）
  + 主键约束（Composite Primary Key）：(ManagerID, WarehouseID)
* 入库记录（Inventory）
  + InventoryID（主键，INT类型，AUTO\_INCREMENT）
  + OrderNumber（外键，INT类型，关联Orders表的OrderNumber字段）
  + Quantity（数量，INT类型）
  + InventoryTime（入库时间，DATE类型）
  + ToWarehouseID（外键，INT类型，关联Warehouse表的WarehouseID字段）
  + Reason（入库原因或描述，VARCHAR(100)类型）
* 出库记录（Delivery）
  + DeliveryID（主键，INT类型，AUTO\_INCREMENT）
  + OrderNumber（外键，INT类型，关联Orders表的OrderNumber字段）
  + GoodsID（外键，INT类型，关联Goods表的GoodsID字段）
  + Quantity（数量，INT类型）
  + DeliveryTime（出库时间，DATE类型）
  + FromWarehouseID（外键，INT类型，关联Warehouse表的WarehouseID字段）
  + Reason（出库原因或描述，VARCHAR(100)类型）
* 调拨记录（Transfer）
  + TransferID（主键，INT类型，AUTO\_INCREMENT）
  + FromOrderNumber（外键，INT类型，关联Orders表的OrderNumber字段，反映调出情况）
  + ToOrderNumber（外键，INT类型，关联Orders表的OrderNumber字段，反映调入情况）
  + SuperManagerID（外键，INT类型，关联SuperManager表的SuperManagerID字段）
  + CreateTime（创建时间，DATE类型，NOT NULL）
  + UpdateTime（更新时间，DATE类型，NOT NULL）

实际进行代码编写时，进行了一定的简化，比如并不是所有的关系中都有创建时间或者更新时间，对非必要的的属性进行了删减。

### ER图



### 数据库操作的设计：

Service层：

*@Service  
public class* DataHandleService {  
 *@Autowired* GoodsDao goodsDao;  
 *@Autowired* WarehouseDao warehouseDao;  
  
 *public* Goods getGoodsByName(String name) {  
  
 QueryWrapper qw = *new* QueryWrapper();  
 qw.eq("goodsname",name);  
 *return* goodsDao.selectOne(qw);  
 }  
  
 *public int* updateGoods(Goods goods,*int* quantity){  
 Warehouse warehouse = warehouseDao.selectById(goods.getWarehouseId());  
 System.out.println(warehouse.getCapacity());  
 System.out.println(goods.getQuantity());  
 *if*(warehouse.getCapacity() >= quantity){  
 warehouse.setCapacity(warehouse.getCapacity() - quantity);  
 warehouseDao.updateById(warehouse);  
  
 *return* goodsDao.updateById(goods);  
 }*else return* 0;  
 }  
 *private int* goodUpdate(String name,*int* quantity,String type){  
 *//出入库修改货物表  
// goodsService = new GoodsService();* Goods goods = getGoodsByName(name);  
*// Goods goods1=new Goods();  
  
// goods1.setGoodsID(goods.getGoodsID());  
  
 if*(type.equals("入库")){  
 goods.setQuantity(goods.getQuantity()+quantity);  
 }*else* {  
 goods.setQuantity(goods.getQuantity()-quantity);  
 }  
 System.out.println("goodUpdate:"+goods);  
 *return* updateGoods(goods,quantity);  
  
 }  
  
 *public int* addGoods(Goods goods){  
 Warehouse warehouse = warehouseDao.selectById(goods.getWarehouseId());  
 *if*(warehouse.getCapacity() <= goods.getQuantity()){  
 warehouse.setCapacity(warehouse.getCapacity() - goods.getQuantity());  
 warehouseDao.updateById(warehouse);  
 *return* goodsDao.insert(goods);  
 }*else return* 0;  
  
 }  
 *public* InventoryRecord toInventoryRecord(Form data, *int* orderId){*//入库订单  
// goodsService = new GoodsService();* InventoryRecord inventoryRecord=*new* InventoryRecord();  
 Goods goods = getGoodsByName(data.getOrder().getGoodsName());  
 *int* res = goodUpdate(data.getOrder().getGoodsName(), data.getOrder().getNumber(), "入库");  
  
 System.out.println("res:"+res);  
 inventoryRecord.setGoodsID(goods.getGoodsID());  
 inventoryRecord.setQuantity(data.getOrder().getNumber());  
 inventoryRecord.setOrderId(orderId);  
 inventoryRecord.setRecipientName(data.getOrder().getRecipientName());  
 inventoryRecord.setSupplierName(data.getOrder().getSupplierName());  
  
 *if*(res == 0) *return null*;  
 *return* inventoryRecord;  
 }  
  
 *public* DeliveryRecord toDeliveryRecord(Form data,*int* orderId){*//出库订单  
// goodsService = new GoodsService();* DeliveryRecord deliveryRecord=*new* DeliveryRecord();  
 Goods goods = getGoodsByName(data.getOrder().getGoodsName());  
 System.out.println(goods);  
 *int* res = goodUpdate(data.getOrder().getGoodsName(), data.getOrder().getNumber(), "出库");  
  
 deliveryRecord.setGoodsID(goods.getGoodsID());  
 deliveryRecord.setQuantity(data.getOrder().getNumber());  
 deliveryRecord.setOrderId(orderId);  
 deliveryRecord.setRecipientName(data.getOrder().getSupplierName());  
 deliveryRecord.setSupplierName(data.getOrder().getRecipientName());  
  
 *if*(res == 0) *return null*;  
 *return* deliveryRecord;  
 }

*@Service  
public class* GoodsService {  
 *@Autowired* GoodsDao goodsDao;  
 *@Autowired* WarehouseDao warehouseDao;  
  
 *public* Goods getGoodsById(*int* id) {  
  
 *return* goodsDao.selectById(id);  
 }  
  
 *public* Goods getGoodsByName(String name) {  
  
 QueryWrapper qw = *new* QueryWrapper();  
 qw.eq("goodsname",name);  
 *return* goodsDao.selectOne(qw);  
 }  
  
 *public* IPage getGoodsList(*int* currentPage, *int* pageSize) {  
 IPage page = *new* Page(currentPage,pageSize);  
 goodsDao.selectPage(page,*null*);  
 *return* page;  
 }  
  
 *public int* addGoods(Goods goods){  
 Warehouse warehouse = warehouseDao.selectById(goods.getWarehouseId());  
 *if*(warehouse.getCapacity() <= goods.getQuantity()){  
 warehouse.setCapacity(warehouse.getCapacity() - goods.getQuantity());  
 warehouseDao.updateById(warehouse);  
 *return* goodsDao.insert(goods);  
 }*else return* 0;  
  
 }  
  
 *public int* updateGoods(Goods goods){  
 Warehouse warehouse = warehouseDao.selectById(goods.getWarehouseId());  
 *if*(warehouse.getCapacity() <= goods.getQuantity()){  
 warehouse.setCapacity(warehouse.getCapacity() - goods.getQuantity());  
 warehouseDao.updateById(warehouse);  
 *return* goodsDao.updateById(goods);  
 }*else return* 0;  
 }  
  
 *public int* deleteGoods(*int* id) {  
 *return* goodsDao.deleteById(id);  
 }  
  
 *public* List<Goods> getByParams(Map<String, String> params) {  
 QueryWrapper qw = *new* QueryWrapper();  
 qw.eq(!"0".equals(params.get("warehouseId")),"warehouseId",Integer.parseInt(params.get("warehouseId")));  
 qw.eq(!"0".equals(params.get("partId")),"partId",Integer.parseInt(params.get("partId")));  
 qw.like("" != params.get("goodsName"),"goodsname",params.get("goodsName"));  
 *return* goodsDao.selectList(qw);  
 }  
}

*@Service  
public class* OrderService {  
 *@Autowired  
 private* DeliveryRecordDao deliveryRecordDao;  
 *@Autowired  
 private* InventoryRecordDao inventoryRecordDao;  
 *@Autowired  
 private* OrdersDao ordersDao;  
  
 *// 添加记录  
 public int* DeliveryRecord(DeliveryRecord deliveryRecord) {  
 System.out.println(deliveryRecord);  
 deliveryRecordDao.insert(deliveryRecord);  
 *return* deliveryRecord.getDeliveryID();  
 }  
  
 *public int* InventoryRecord(InventoryRecord inventoryRecord) {  
 System.out.println("InventoryRecord:"+inventoryRecord);  
 *// 添加记录* inventoryRecordDao.insert(inventoryRecord);  
 *return* inventoryRecord.getInventoryID();  
 }  
  
 *public* IPage getOrdersList(*int* currentPage, *int* pageSize) {  
 IPage page = *new* Page(currentPage,pageSize);  
 ordersDao.selectPage(page,*null*);  
 *return* page;  
 }  
  
 *public* List<Orders> getOrderAll() {  
 *return* ordersDao.selectList(*null*);  
 }  
  
 *public int* insertOrders(Orders orders) {  
 ordersDao.insert(orders);  
 *return* orders.getOrderId();  
 }  
  
 *public* List<Orders> getByParams(Map<String, String> params) {  
 QueryWrapper qw = *new* QueryWrapper();  
 qw.eq(!"0".equals(params.get("warehouseId")),"warehouseId",Integer.parseInt(params.get("warehouseId")));  
 qw.eq(!"0".equals(params.get("partId")),"partId",Integer.parseInt(params.get("partId")));  
 qw.like("" != params.get("goodsName"),"goodsname",params.get("goodsName"));  
 *return* ordersDao.selectList(qw);  
 }  
}

*@Service  
public class* SuperManagerService {  
 *//增删改查  
 @Autowired* SuperManagerDao superManagerDao;  
  
 *public* SuperManager getById(*int* id){  
 *return* superManagerDao.selectById(id);  
 }  
  
 *public* List<SuperManager> getAll(){  
 *return* superManagerDao.selectList(*null*);  
 }  
  
 *public int* add(SuperManager superManager){  
 *return* superManagerDao.insert(superManager);  
 }  
  
 *public int* update(SuperManager superManager){  
 *return* superManagerDao.updateById(superManager);  
 }  
  
 *public int* delete(*int* id){  
 *return* superManagerDao.deleteById(id);  
 }  
  
}

*@Service  
public class* TransferService {  
 *@Autowired  
 private* TransferRecordDao transferRecordDao;  
  
 *//插入  
 public int* insertTransferRecord(TransferRecord transferRecord) {  
 *return* transferRecordDao.insert(transferRecord);  
 }  
  
 *//查询  
 public* TransferRecord getTransferRecordById(*int* id) {  
 *return* transferRecordDao.selectById(id);  
 }  
  
 *//查询所有  
 public* List<TransferRecord> getAll() {  
 *return* transferRecordDao.selectList(*null*);  
 }  
}

# 测试技术说明书

## 7.1测试策略：

单元测试、集成测试、系统测试和验收测试的方法论

单元测试（Unit Testing）：这是对软件中的最小可测试单元进行检查和验证。对于面向对象编程，这一般指的是类或方法。单元测试主要用于验证单个组件的正确性。通常由开发人员编写，并使用如JUnit、NUnit等框架进行。

集成测试（Integration Testing）：这是在所有模块单独测试后，将它们合并并作为组进行测试。它的目的是发现模块接口间的问题。集成测试可以采用自顶向下、自底向上或沙盒等方法进行。

系统测试（System Testing）：这是对整个系统或应用的测试。系统测试将软件系统视为一个整体，目标是验证系统满足所有指定的需求。

验收测试（Acceptance Testing）：这通常是软件测试的最后一步，目的是验证软件系统是否满足用户的业务需求。验收测试可以由客户、最终用户、或者由第三方代表这些利益相关者进行。

## 7.2测试用例设计：

列举代表性测试用例，覆盖所有关键场景和边界条件

1.人员管理：

* 添加新员工时，所有必填字段都已填写。
* 添加新员工时，输入无效的数据（例如，年龄为负数）。
* 删除存在的员工。
* 删除不存在的员工。

2.仓库管理：

* 添加新仓库时，所有必填字段都已填写。
* 添加新仓库时，输入无效的数据（例如，仓库面积为负数）。
* 删除存在的仓库。
* 删除不存在的仓库。

3.货物管理：

* 添加新货物时，所有必填字段都已填写。
* 添加新货物时，输入无效的数据（例如，货物数量为负数）。
* 删除存在的货物。
* 删除不存在的货物。

4.订单管理：

* 创建新订单时，所有必填字段都已填写。
* 创建新订单时，输入无效的数据（例如，订单数量为负数）。
* 删除存在的订单。
* 删除不存在的订单。

5.订单识别：

* 识别有效的订单编号。
* 识别无效的订单编号。
* 识别非订单，如动植物等

6.货物识别：

* 识别有效的货物编号。
* 识别无效的货物编号。
* 识别非货物，如文本等

## 7.3测试环境搭建：

所需的软硬件配置和网络环境设定

由于编写代码和进行测试的硬件环境差异性较大，因此很难提供一个确切的软硬件配置和网络环境设定的详细列表。以下是一些通用的建议：

1. 操作系统：编写代码和进行测试的环境是Windows10

2. 编程语言：Java+vue

3. 编程编辑器：可以使用跨平台的编辑器，如Visual Studio Code+IDEA

4. 数据库：MySQL

1. 测试框架：使用JUnit等测试框架

8. 依赖管理：使用Maven依赖管理工具，确保项目依赖清晰且可用。

9. 代码版本控制：使用Git或其他代码版本控制工具，确保代码在版本控制系统中归档。

## 7.4测试工具选择：

选用何种自动化测试工具和技术进行辅助测试

## 7.5测试进度安排和质量指标：

如何跟踪测试进度并确保达到预定的质量标准

测试进度安排：

1. 首先，根据项目的需求和规模，确定测试的总体目标和优先级。

2. 然后，将测试任务分解为多个阶段，每个阶段对应一个特定的测试目标或场景。

3. 在每个阶段中，确定需要测试的用例集，并将其分配给相应的测试人员。

4. 最后，确保所有测试阶段都按照计划和预期进行，并在必要时进行调整。

测试质量指标：

1. 测试覆盖率：这是评估测试进度的重要指标，表示代码中被测试的部分占整体的百分比。理想情况下，覆盖率应该达到100%。

2. 测试成功率：表示测试中所有用例都成功的比例。这个指标可以反映出测试的整体质量。

3. 测试稳定性：表示测试结果在多次运行时的一致性。稳定性是确保测试有效性的基本要求。

4. 测试effort（效率）：表示完成测试所消耗的时间与预期时间的比例。这个指标可以帮助项目团队了解测试工作的effort。

5. 测试可维护性：表示测试用例的可读性和可维护性。良好的可维护性有助于长期维护测试用例集。

6. 自动化测试率：表示自动化测试用例占总数的比例。这个指标可以帮助项目团队了解自动化测试工作的Progress。

7. 测试反馈 loops（反馈循环）：表示从发现bug到fix的周期时间。较短的反馈周期可以提高项目的bug-fix效率。

8. 外部测试资源：表示项目使用的外部测试资源（如测试服务、测试工具等）与项目规模的比例。这个指标可以帮助项目团队了解外部测试资源的使用情况。

# 用户使用说明书

## 系统简介：

系统概述：简述易仓仓储管理系统的核心功能与优势。

系统环境要求：包括硬件配置、软件平台、网络条件等。

## 安装与启动：

安装步骤：详细描述如何下载、安装及注册激活系统。

启动与登录：指导用户如何打开和登录到易仓仓储管理系统。

## 功能模块介绍：

仓库管理：说明仓库新增、修改、删除以及区域划分的具体操作流程。

货物管理：详细介绍货物入库、出库、调拨、查询以及盘点的操作方法。

订单管理：解释如何创建订单、跟踪订单状态、处理发货提醒与签收通知。

物流状态实时追踪：阐述二维码扫描识别与物流路径优化的使用方法。

短信提醒设置：指导用户如何设置和定制各类短信提醒模板。

## 界面与导航：

系统界面布局：对系统主界面及其各个菜单项进行图示说明。

导航与操作指引：详述各功能模块之间的跳转逻辑及具体操作步骤。

## 常见问题与解决方案：

列举用户在使用过程中可能遇到的常见问题，提供详细的解决办法或故障排查步骤。

## 系统维护与技术支持：

提供系统更新、备份恢复、权限管理等相关维护指南。

明确联系方式和技术支持渠道，以便用户在遇到问题时获取及时的帮助。

## 附录：

可能包括快捷键列表、词汇表、版本更新日志等补充信息。

撰写用户使用说明书时需确保语言通俗易懂，图文并茂，让不同层次的用户都能轻松上手，高效地利用易仓仓储管理系统进行日常管理工作。

# 做小组项目的体会与心得

## 9.1

# 参考文献

【1】

【2】

# 附录

## 附录A X小组XXX项目的源代码.rar

## 附录B X小组XXX项目的安装说明书.doc

## 附录C X小组XXX项目的原图.rar