



天津中德应用技术大学
Tianjin Sino-German University of Applied Sciences

本科生毕业设计

基于微服务架构的校园线上二手回收与寄修系统设计与实现

Design and implementation of campus online Second-hand Recycling
and mailing system based on micro-service architecture

姓 名 曹兆祺
学 院 软件与通信学院
专 业 通信工程
指导教师 胡晓光
职 称 副教授
完成时间 2022年6月

天津中德应用技术大学

本科毕业生毕业设计（论文）的声明

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，进行研究工作所取得的成果。除文中已经注明引用的内容外，本毕业设计（论文）的研究成果不包含任何他人创作的、已公开发表或没有公开发表的作品内容。对本设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本毕业设计（论文）原创性声明的法律责任由本人承担。

毕业设计（论文）作者签名：

年 月 日

本人声明：该毕业设计（论文）是本人指导学生完成的研究成果，已经审阅过设计（论文）的全部内容，并能够保证题目、关键词、摘要部分中英文内容的一致性和准确性。

毕业设计（论文）指导教师签名：

年 月 日

摘要

在目前二手市场中，线上交易占据绝大部分。在二手物品线上交易平台中：闲鱼、转转等已经占有足量的市场份额；上述二者通过先发优势，迅速抢占市场，因此获得了飞速发展。但在高速发展背后，仍然存在许多问题：1.线上交易的便利性取决于快递的时效性。而目前快递时效为3-5天；在面对用户的紧急需求时，则较为棘手；2.在另一方面，线上交易暴露出一些其他弊端：实物与展示品差异不定、售后服务繁琐等问题。

基于上述问题，市场急需一种新型的二手交易平台解决传统单纯线上交易的问题。基于解决问题的思路，本文提出的方法是通过采用线上+线下的方式进行二手商品的回收与售卖、物品的寄修。通过此方法可实现：在既保证线上线下服务质量的同时又促进了线上和线下的交易的同时发展。通过在线上开发出一种系统实现线上交易，在线下设立实体门店提供服务。以此方案解决传统线上存在的弊端。

通过上述分析，笔者决定开发这一系统。本系统主要包括以下平台：移动端、门户端、管理端、后端服务。在移动设备端，通过采用基于uni-app框架的uView UI多平台混合开发框架来实现；在门户端采用基于Vue.js的Nuxt.js框架开发，使用Element UI和LayUI绘制界面；在本系统的管理端主要采用基于Vue.js的Element UI框架进行界面绘制；在系统后台服务的开发中，主要通过基于微服务架构进行设计和开发。移动设备端与门户端主要功能有：物品维修预约、物品捐赠、物品回收与售卖、积分兑换物品、个人信息管理、全文检索等功能构成；管理端主要包括以下功能：用户数据管理、角色信息管理、系统角色的权限管理、菜单管理、回收与维修数据管理、订单信息管理、活动发布、积分管理等功能构成；在系统后台设计并实现了：用户鉴权、用户数据管理、数据管理、积分管理、订单管理、搜索服务、文件系统管理等微服务。在最终系统上线部署阶段，移动端采用uView UI多平台框架进行发布；门户端与后台管理端采用NPM打包方式或在打包后通过npm run dev命令部署至云端服务器，通过Nginx服务器进行反向代理；后台服务部署采用基于Docker的虚拟化容器技术进行部署。

通过上述方式实现了基于微服务架构的校园线上二手回收与寄修系统的设计与实现。在工作中尾声阶段，进行了系统测试与上线部署。系统开发达到预期效果。在后续中系统运维中，可以对系统进行持续性演进和维护。

关键词：二手回收；寄修；微服务架构

ABSTRACT

In the current second-hand market, online transactions occupy the majority. In the second-hand goods online trading platform: Xianyu, Zhuan and so on have occupied a sufficient amount of market share; Both of these have achieved rapid growth by rapidly seizing the market through first-mover advantage. However, behind the rapid development, there are still many problems: 1. The convenience of online trading depends on the timeliness of express delivery. The current delivery time limit is 3-5 days; In the face of users' urgent needs, it is more difficult; 2. On the other hand, online trading exposes some other disadvantages, such as the difference between objects and exhibits, and the tedious after-sale service.

Based on the above problems, the market is in urgent need of a new type of second-hand trading platform to solve the problem of traditional pure online trading. Based on the idea of solving the problem, the method proposed in this paper is to recycle and sell second-hand goods and send and repair second-hand goods through online + offline mode. Through this method, it can be realized that both online and offline service quality can be guaranteed while simultaneously promoting the development of online and offline transactions. Through the online development of a system to achieve online transactions in the offline establishment of physical stores to provide services. This solution solves the disadvantages existing in traditional online.

Through the above analysis, the author decided to develop this system. This system mainly includes the following platforms: mobile terminal, portal terminal, management terminal, back-end service. In the mobile device side, the uView UI multi-platform hybrid development framework based on uni-app is adopted to achieve this. In the portal, nuxt.js framework based on vue.js is used to develop the interface, and Element UI and LayUI are used to draw the interface. In the management end of this system, the Element UI framework based on vue.js is used to draw the interface. In the development of system background service, mainly through the design and development based on micro-service architecture. The main functions of mobile device terminal and portal terminal include: item maintenance reservation, item donation, item recovery and sale, point exchange, personal information management, full-text retrieval and other functions; The management terminal mainly includes the following functions: user data management, role information management, system role permission

management, menu management, recovery and maintenance data management, order information management, event release, integral management and other functions; In the system background design and implementation: user authentication, user data management, data management, integral management, order management, search services, file system management and other micro services. In the final system online deployment stage, the mobile terminal uses uView UI multi-platform framework to publish; The portal end and the background management end are packaged in NPM mode or deployed to the cloud server through the NPM run dev command, and the reverse proxy is implemented through the Nginx server. Background service deployment adopts docker-based virtualization container technology to deploy.

Through the above methods, the design and implementation of campus online second-hand recycling and mailing system based on micro-service architecture are realized. At the end of the work, the system was tested and deployed online. The system development achieves the expected results. In subsequent operations in the system, the system can be persistent evolution and maintenance.

Key words: Second-hand recycling; Send the repaired; Microservices Architecture

目 录

第一章 绪论.....	1
1.1 研究背景及意义.....	1
1.2 国内外研究现状.....	1
1.3 研究内容.....	2
1.4 论文的组织安排.....	3
第二章 技术栈和研究方法	5
2.1 移动端技术栈.....	5
2.2 前端技术栈.....	6
2.3 服务端技术栈.....	7
2.4 研究方法.....	11
2.5 本章小结.....	12
第三章 需求分析.....	13
3.1 功能需求分析.....	13
3.2 性能需求分析.....	16
3.3 安全需求分析.....	17
3.4 软硬件需求分析.....	17
3.5 本章小结.....	17
第四章 系统设计.....	19
4.1 系统概要设计.....	19
4.2 系统详细设计.....	33
4.3 本章小结.....	48
第五章 系统实现.....	50
5.1 开发环境搭建.....	50
5.2 模块实现.....	51
5.3 关键功能代码分析.....	66
5.4 本章小结.....	70
第六章 系统测试.....	71
6.1 单元测试.....	71
6.2 集成测试.....	84

6.3 测试结论.....	85
6.4 本章总结.....	86
第七章 总结与展望	87
参考文献.....	88
致谢.....	90

第一章 绪论

1.1 研究背景及意义

目前，单纯线上交易存在诸多问题：以著名二手回收平台，闲鱼、转转为例。虽然二者在二手回收市场占有较大份额，但存在诸多问题：快递时效性无法保证、服务质量层次不齐、交易纠纷易发等。因此线上线下交易结合势在必行。

当前学生的网购物品数量较大，种类繁多，部分物品回收价值较高，具有一定回收价值；多数高校位于城市郊区位置，交通较为不便，学生所购物品损坏后维修难度大，可能导致直接丢弃后购买新品，不利于环境保护并实现碳达峰、碳中和的目标。

近一段时间来，微服务架构发展迅猛。其中尤以 Spring Cloud 框架应用广泛。在系统中通过采用 Spring Cloud 框架可以实现系统的快速迭代。本系统后台采用 Spring Cloud 框架开发。在移动端，混合开发模式异军突起，尤其以 uni-app 框架为代表。在 Web 前端发展出许多优秀 UI 框架，诸如：Element UI、uView UI、LayUI 等。本系统移动端、管理端、门户端采用上述方案。通过研究以上问题可以较为方便的开发出本系统，从而解决上述问题。

1.2 国内外研究现状

1.2.1 国外研究现状

在国外，欧美等发达国家具有完整的二手平台，其对市场有完善的知识产权保护，完整的物流体系^[1]。

二手市场在美国存在相当长一段时间，具体形式主要为：线下（旧书摊、跳蚤市场、募捐活动）、线上（ebay、amazon）等。二手市场发展在一定情况下刺激了线下市场。在全球校园市场：以麻省理工为例，每年 4 月 10 日会举办跳蚤市场，在日韩等国家，购买二手商品的多为走读生。在西欧国家英国，跳蚤市场已经走出校园，极大地刺激了经济发展，由此产生了比较著名的跳蚤市场。

在维修市场中，国外缺乏专门的手机维修门店与平台，主要维修由第三方负责。因此维修服务质量良莠不齐，维修效果无法确保。在线上：例如美国的 eRecyclingCorp 平台主要提供回收服务，缺乏专业维修服务。

1.2.2 国内研究现状

在国内：伴随着改革开放，我国经济飞速发展，人民生活水平的不断提高^[2]。共享经济愈加火热，它的领域从最初的共享单车已经进入出行、交通等领域，二手交易平台正是共享经济的产物。

分析国内的二手交易平台可以发现：他们都有着比较成熟的 C2C 模式，抑或是 C2C-C2B2 模式。国内主要二手回收市场由闲鱼、转转、爱回收所占据，闲鱼在行业内占据回收的龙头位置。它们有比较明显的优点：比如：成熟的平台、交易体系成熟、用户基数众多等，而相反的是也存在比较显著的缺点：物流时效性无法保证、服务质量层次不齐、退换货成本高等。诸多问题影响用户的体验。

分析当下国内维修平台可以发现：他们大多存在于线下，且由于人群密集、服务质量无法满足客户需求、服务人数存在限制等问题。在转换思路后，我们发现可以线上预约、线下维修并领取这种模式处理这些问题，这样可以显著减少人群的聚集，也可提高服务人数、服务质量。

1.3 研究内容

本课题主要研究内容为当下校园二手回收与线上寄修过程中存在的诸多问题。为解决存在的问题，本文中提出了一种解决方法：即通过线上和线下同时搭建服务体系：线上通过设计并实现相应的智能终端 App 与门户网站实现线上物品的回收预约、回收品售卖、物品捐赠、积分兑换等功能；通过设计并实现对应的管理后台数据管理、物品管理、交易订单管理、活动发布等功能；线下通过在校园设立相应门店提供服务，在人流量较高处进行二手商品的回收与销售。基于此，笔者希望可以解决上述存在的问题。

本系统中，主要由移动设备端 App、门户网站、管理端、服务端构成。

在移动设备端，本系统采用基于混合开发 uni-app 框架的 uView UI 框架，主要包括：基本登录注册功能、个人信息管理、二手商品交易、物品捐赠、物品维修、积分兑换商品等功能。

在门户网站端，本系统采用基于 Vue.js 的 Nuxt.js 框架开发，采用 SSR (Server Side Render，服务端渲染) 模式。通过服务端渲染后，浏览器显示服务端生成 HTML 页面。采用此种模式可以极大利于 SEO (Search Engine Optimization，搜索引擎优化)，实现的功能类似于移动端（增加购物车功能）。

在管理端，本系统采用基于 Vue.js 的 Element UI 框架绘制界面。主要实现了：个人基础信息管理、二手物品管理、寄修物品信息管理、订单管理、积分管理、活动发布等功能。

在服务端，采用基于微服务架构的设计思想，采用 Spring Cloud 框架设计了如下微服务^[3]：

- 1.rs_api_servive：接口微服务，定义本系统使用的接口；
- 2.rs_auth_service：用户鉴权微服务，主要用于用户的权限验证、登录注册；
- 3.rs_center_service：注册中心微服务，主要基于 Spring Cloud 框架的 Spring Cloud Eureka 组件设计，在系统中用于微服务的注册与发现；
- 4.rs_filesystem_service：文件系统微服务，主要用于系统内文件的上传存储；
- 5.rs_gateway_service：网关微服务，主要用于转发前端经 Nginx 转发的 API 请求后传送至各个微服务；
- 6.rs_manage_service：数据管理微服务，主要用于处理管理端、移动端的接口请求并处理数据后返回处理结果；
- 7.rs_order_servie：订单微服务，主要处理与订单有关的请求并返回数据；
- 8.rs_points_service：积分微服务，主要用于处理与用户积分相关的请求并返回数据；
- 9.rs_search_service：搜索微服务，主要用于处理移动端的全局搜索并返回相应数据；
- 10.rs_user_service：用户微服务，主要处理用户管理的请求并返回相应数据。

基于以上设计内容，我们通过采用前后端分离的思想，选取对应的开发框架进行开发。在后台服务端，通过设计服务端的架构，将其进行分层，由上至下主要为：展现层、网络层、业务层、数据层、基础设施层^[4]，在每一层对功能进行详细划分，由此实现各个服务间的高耦合、低内聚，在项目测试完成后的部署阶段，采用以虚拟化技术为代表的 Docker 技术进行部署。在系统部署结束后：基于 DevOps 思想，利用 Jenkins 工具对系统进行持续集成(CI)/持续交付(CD)不断运维，希冀于通过此方法实现系统的较好状态运行。

1.4 论文的组织安排

第一章：主要介绍本篇论文的绪论内容，在本章节主要介绍了：本文介绍系统的研究背景、研究意义、国内外研究现状。在章最后介绍了本论文的主要研究内容；

第二章：技术栈和研究方法，主要分为两部分，第一部分通过介绍移动端、前端、服务端的技术以便于读者了解系统使用的技术；第二部分介绍了本文的研究方法，这些方法被用于项目开发过程、撰写论文过程中；

第三章：需求分析，在本章节分析了本系统的主要需求。包括：功能需求、性能需求、安全需求和软硬件需求。通过详细的需求分析，使得系统的设计更加

明确；

第四章：系统设计，介绍本系统的基本构成，基本介绍不同平台的功能的设计流程；

第五章：系统实现，本章节介绍了系统各个用户端的具体实现流程与方法；

第六章：总结与展望，对于本文工作进行简单总结，对未来发展作以展望。

第二章 技术栈和研究方法

2.1 移动端技术栈

在移动 App 开发中主要存在四种主流开发模式：即：原生开发(Native App)、网页开发(Web App)、混合开发(Native App)和 React Native App 开发^[5]。

2.1.1 原生开发 (Native App)

在原生开发模式中，主要使用对应平台支持的 IDE 和相对应的编程语言进行开发。例如：使用 Android Studio v3.5 IDE 亦或 Eclipse for Android developer IDE 等开发环境，运用 Java 编程语言开发 Android App。他们比较显著的特点在于和操作系统进行深度绑定，使用专有的 UI 库进行开发，可以方便调用系统的底层接口。但与此同时，也有比较明显的缺点：开发后的 App 跨平台性差、开发成本高、维护成本高等。

2.1.2 网页开发 (Web App)

网页开发模式中，一般使用基于 Vue.js 框架、H5、JS、CSS 等编程语言进行开发。在设计和开发过程中基于前端开发思想，可以使得 App 具有：简单、便于维护、部署简单等特点。在部署阶段，采用 NPM 打包进行部署、通过 Nginx 反向代理处理来自 App 的数据请求接口。

2.1.3 混合开发 (Hybrid App)

混合开发是目前使用较多的开发模式，它具有如下特点：开发维护系统成本较低、可实现功能丰富、系统迭代快速等。在混合开发中，主要分为三种模式：

H5 加壳模式：主要以 Ionic 框架和 uni-app 框架为代表；

JS Run 原生模式：以 RN 和 Wexx 为代表的，主要使用 JS 进行编写，在 App 运行时映射为原生控件运行^[6]；

自成模式：以 Flutter 为代表，使用自有渲染引擎、自有布局进行开发。

在上述三种模式中，以 H5 加壳模式使用较多，本系统移动端使用基于 uni-app 的 uView UI 框架进行开发。

2. 1. 4React Native App

此种开发模式主要由 Meta(Face Book)发起。不同于 H5 开发方式和原生开发方式，此种开发方式需要投入巨大物力、人力资源。开发 App 性能接近于原生开发性能，开发者主要使用 JS 编程语言和 React 框架进行开发。具有跨平台、性能卓越、扩展性好等特点。

2. 2 前端技术栈

在本系统的后台管理端，使用了较多的技术点：包括基础性的 Java Script、CSS、HTML 技术；其中 Java Script 负责与用户的交互、HTML 负责绘制页面、CSS 负责页面的美化。在基础技术基础上演进出了许多优秀的前端基础框架。比较主流框架有 Vue.js、React.js、Angular.js 等。在这些基础框架之上演进出了纷繁复杂的 UI 框架，比如：Element UI、LayUI 等。这些框架极大简化了开发的流程、简化了开发难度，使得产品更快速迭代。

在本系统管理端中，采用基于 Vue.js 的 Element UI 框架；在本系统门户网站采用 LayUI 框架。

2. 2. 1Vue. js

Vue.js 通常被用于前端开发，是一种渐进式的 JavaScript 框架。称其为渐进式框架的原因在于在开发时，不需要掌握全部技能。跟随项目进度可以自由增加功能。Vue.js 在最初被设计为可以自底向上逐层应用^[7]，这一特点使得开发前端应用更为方便。Vue.js 具有双向数据绑定、页面组件化等优点，它将数据与组件进行深度绑定，使得开发的单页应用功能更为丰富。

2. 2. 2Element UI

Element UI 是一款基于 Vue.js 的前端 UI 框架，此框架具有丰富的组件库。一致、反馈、效率、可控原则贯穿了使用 Element UI 框架开发软件的全过程。在使用方面可采用 NPM 包管理器安装，也可通过 CDN 方式进行引入。本系统后台管理端使用此 UI 框架进行开发。

2. 2. 3Nuxt. js

Nuxt.js 是一个基于 Vue.js 的应用框架，主要用于开发 SSR 应用^[8]。其优点在于可将配置文件整合至 nuxt.config.js 中，实现了良好的扩展性。SSR 应用主要分为客户端、服务端，服务端渲染 html 页面至客户端显示，有利于网站的 SEO。在本项目中，门户网站主要使用该项技术进行开发。

2. 2. 4LayUI

LayUI 为一款开源的 WebUI 框架，使用 HTML、CSS、JS 方式进行开发。LayUI 主要面向后端开发者，具有：易上手、简约、组件丰富等特点。在本项目门户网站页面构建中使用了 LayUI。

2. 3 服务端技术栈

服务端程序的开发是在前端的基础之上并注重发生在前端背后的逻辑。在最初，前后端处于未分离时代。所有设计思想基于 MVC 框架，整体开发集中于后端。开发效率缓慢、维护困难。比如：JSP、PHP 等技术。

经过一段时间发展后，前后端逐渐处于半分离状态。主要开发思想是基于 MVC，比如：Spring MVC 等技术。

在经过长时间发展后，编程整体形成了前后端分离的大趋势。在前后端分离的整体设计中，前端主要负责 View 层和 Controller 层；后端负责 Model 层和服务层^[9]。之后 Node.js 的出现提供了和后端交互的 API。经过单体式服务的不断演进，后台服务设计并发展出了基于分布式的微服务架构设计。主要发展出 Spring Cloud、Dubbo、istio 框架等。其中以 Spring Cloud 使用最为广泛。

本系统中采用前后端分离的设计思想，在系统服务端使用 Spring Cloud 框架进行程序设计与开发。

2. 3. 1Spring Cloud

微服务的引入，为我们开发系统带来了许多好处：独立部署、启动迅速、适用于敏捷开发、便于复用、职责单一、动态扩容等特点。

Spring Cloud 框架基于 HTTP 的 RESTs 服务构建服务体系，是一系列框架的集合。主要包括 Spring Cloud Eureka（服务注册中心）、Spring Cloud Ribbon（负载均衡组件）、Spring Cloud Feign（外部接口调用）、Spring Cloud Zuul

(API 网关) 等模块^[10], 如图 2-1 所示为现有后台服务架构。

本系统后台服务采用基于微服务思想设计的 Spring Cloud 框架。

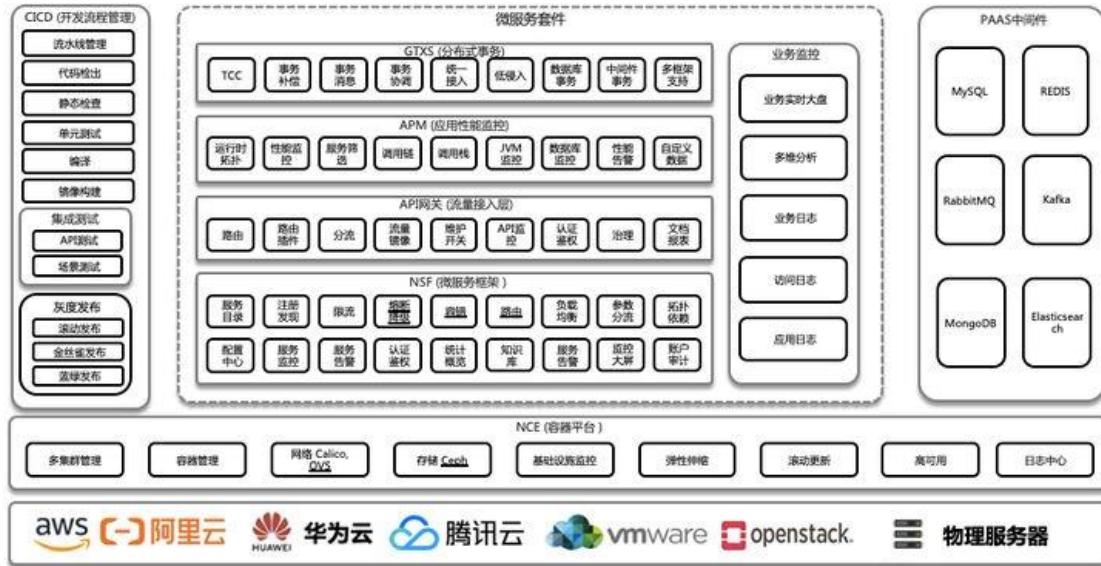


图 2-1 后台服务框架

2. 3. 2 Docker

在虚拟化技术中, Docker 使用最为广泛, 它是一个开源、轻量级的容器引擎。Docker 主要用于 Windows、Linux 操作系统之上创建并管理容器。Docker 区别于 VM Ware、Virtual Box 具有轻量化、开销小、高性能等特点。

本系统的后台服务均部署在 Docker 容器中, 包括本系统中的 MySQL、Redis、Jenkins、MongoDB、Fast DFS 服务均安装在 Docker 容器中。

如图 2-2 所示, 为 Docker 体系的工作原理。本项目通过 Docker 借助其他后端服务部署于云端服务, 以此实现本系统后台服务的开发与后期维护。

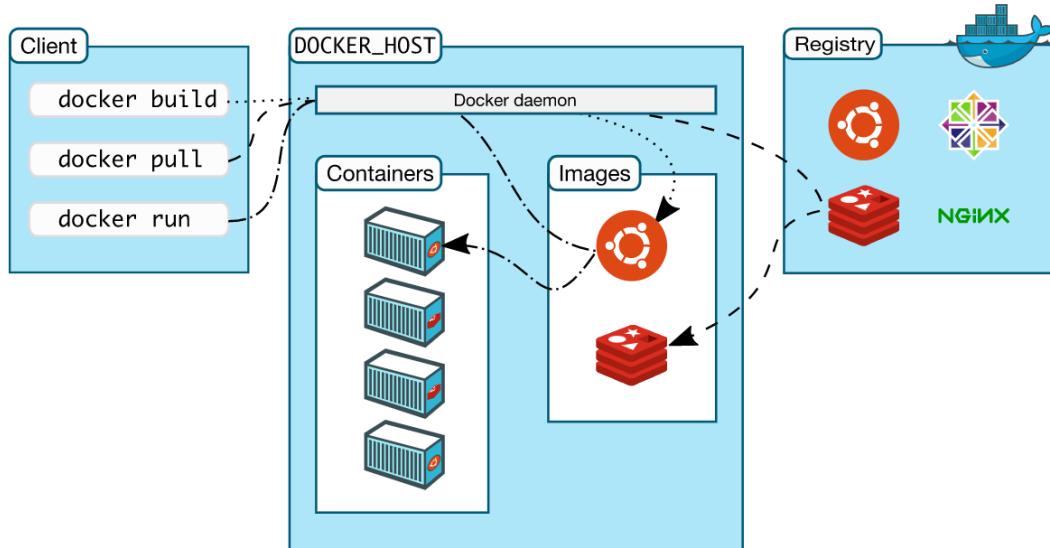


图 2-2 Docker 工作流程

2. 3. 3ELK

ELK 是一种解决方案，它是三个软件套件的首字母（Elasticsearch、Logstash、Kibana）的组合。通过三个软件的配合使用，我们可以实现数据、日志的实时分析。

Elasticsearch 是一个实时的分布式搜索和分析引擎，主要用于数据的全文搜索与分析；Logstash 是一个具有实时的数据搜索引擎；Kibana 负责监控二者运行状态；Elasticsearch 中进行数据的索引查找、数据交互并生成图表^[11]。

在本系统中主要通过 Logstash 操作 Elasticsearch 进行数据分析，IK 分词器分词加工，最终输出至后台服务并返回至前台显示。如图 2-3 为 ELK 架构图。

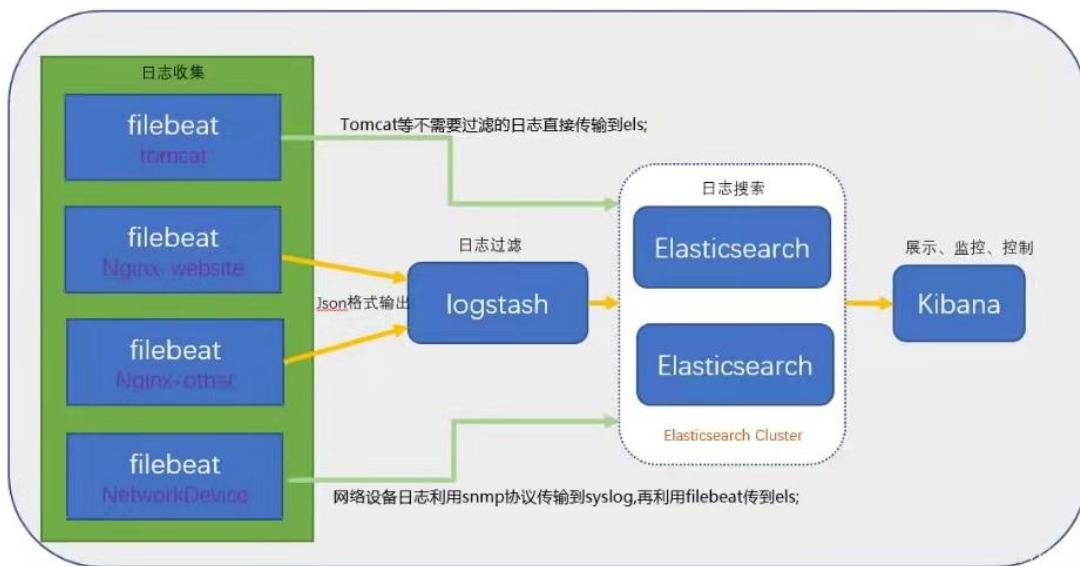


图 2-3 ELK 架构图

2. 3. 4MySQL

数据库被定义为一种按照数据结构组织、存储、管理数据的仓库，主要分为关系型数据库和非关系型数据库。使用较多数据库有 MySQL、SQL Server 等数据库，MySQL 为瑞典 MySQLAB 公司开发，将数据存储于不同表中，具有存储速度快速、灵活性高特点^[12]。

本系统采用 MySQL 存储各个平台数据，使用 Navicat for MySQL 设计并管理 MySQL 数据库。

2.3.5 MongoDB

除关系型数据库外，非关系型数据库在本系统中也有使用。著名的非关系型数据库有：MongoDB、HBase 等，其中 MongoDB 使用较多；MongoDB 是一个基于分布式的、非关系型数据库。它具有许多显著特点：高性能、易部署、使用方便等特点^[13]。本系统使用 MongoDB 存储各个平台的上传的文件索引数据。一般将其部署于云端 Docker 容器中，在本地使用可视化工具 Studio 3T 进行数据管理。

2.3.6 Redis

Redis(全称：Remote Dictionary Server)是一个键-值存储系统。Redis 作为非关系型数据库存储形式多样。Redis 可以采用分布式方式实现主从结构，可以实现数据的快速同步。在本系统后台服务使用 Redis 存储用户的认证信息。

2.3.7 FastDFS

FastDFS 是一个基于 C 语言开发的开源分布式文件系统，主要用于中小型文件存储，可通过 API 接口进行访问。FastDFS 由跟踪器(Tracker Server)、存储点(Storage Server)组成。跟踪器负责资源调度，访问控制、负载均衡等功能；存储节点存储用户上传的具体文件。本系统使用 FastDFS 存储用户上传的数据，如图 2-4 为具体文件存储流程。

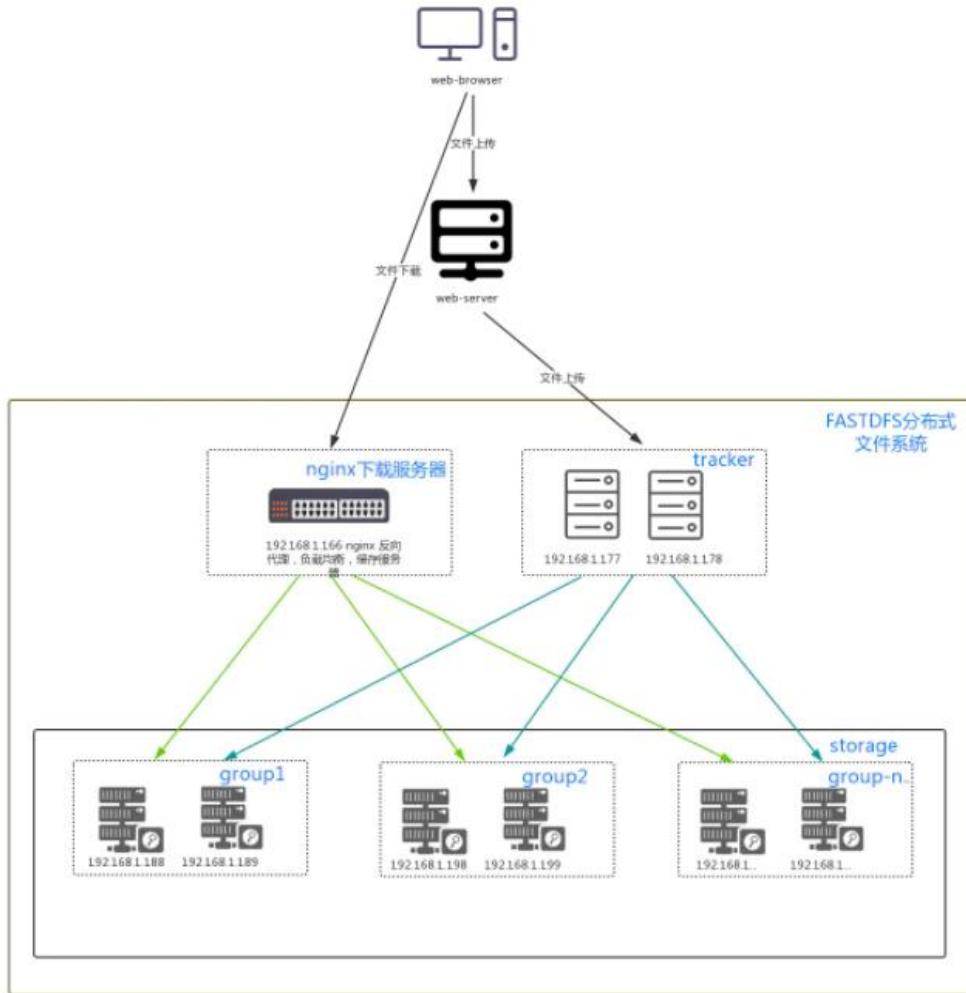


图 2-4 FastDFS 文件存储流程

2. 3. 8Nginx

Nginx 是一个轻量级、高性能的反向代理服务器,可处理静态文件、可进行反向代理加速、可进行负载均衡；具有跨平台、启动迅速、占用内存小等特点^[14]。本系统使用 Nginx 进行服务的反向代理和负载均衡，使用 Nginx 进行后台管理端的接口请求转发。

2. 4 研究方法

2. 4. 1 文献分析法

文献分析法旨在通过研究已有的文献资料，熟悉软件设计的基本方法，更加

合理的设计软件。通过上述方法使得软件之间分工更趋于明晰、具体设计更加合理、模块之间调用更加便利。经过文献研究法的进一步分析，我们可以更加便捷快速并合理的设计系统。

2.4.2 模型法

通过对用户需求的调研分析，我们可以建立一个较好的需求模型。经过对于需求模型的进一步分析，设计人员可以很快的设计出系统原型。在为用户展示系统原型后，我们可以很快速的明确用户具体需求以及潜在需求。经过使用模型法的分析后，我们设计的系统原型更加符合用户需求。

2.5 本章小结

本章主要介绍开发本系统所需的技术：分为移动端技术栈、前端技术栈、服务端技术栈。对于这些技术栈进行了简要介绍，明确了每种技术在项目中的使用情况。在章节第二部分对本项目的研究方法做了概括性的描述。

第三章 需求分析

3.1 功能需求分析

功能需求分析主要旨在通过对系统中每个模块进行功能性需求分析，明确用户的功能需求^[15]。

3.1.1 移动端功能需求分析

移动端功能需求详情如表 3-1 所示。

表 3-1 功能需求详情分析表

需求编号	模块	功能	需求描述
A1	主页	轮播图	展示相关活动简介
A2		活动列表	展示现有活动
A3		活动详情	展示具体活动详情
A4		商品列表	浏览已发布商品
A5		商品详情	查看商品详情信息
A6		物品评论	查看物品评价
A7			发起物品评价
A8		商品购买	购买已发布物品
B1	捐赠	信息填写	捐赠前信息填写
B2		发布捐赠	发布已填写的捐赠
B3		评价服务	评价捐赠服务
C1	回收	开始寄件	输入物品信息，开始回收
C2		信息确认	确认用户输入信息
C3		确认回收	确认回收物品信息
C4		文件上传	上传物品介绍图
D1	维修	信息输入	输入物品维修信息
D2		开始维修	确认信息，开始维修
D3		进度查询	查询已维修进度
D4		在线评价	评价已完成维修服务质量
E1	登录	密码登录	通过密码登录 App

F1	退出		退出账户
G1	我的	个人信息	查阅个人信息
G2		积分商城	查询已有积分信息
G3			积分兑换物品
G4			输入收货信息
G5		订单	查询已有回收订单
G6			查询已购买订单
G7			查询已发货订单
G8		地址管理	查询个人地址信息
G9			添加个人地址信息
G10			修改、删除个人地址信息
G11			设置默认个人地址
G12		捐献管理	已捐赠物品管理
G13		维修商品	查询已维修物品信息
G14			修改待维修物品信息
H1	搜索	全文检索	搜索商品信息

3.1.2 门户端功能需求分析

门户功能需求详情如表 3-2 所示。

表 3-2 功能需求详情分析表

需求编号	模块	功能	需求描述
A1	主页	轮播图	展示相关活动简介
A2		活动列表	展示现有活动
A3		活动详情	展示具体活动详情
A4		商品列表	浏览已发布商品
A5		商品详情	查看商品详情信息
A6		商品购买	购买已发布物品
A7		添加购物车	添加商品至购物车
A8		浏览捐赠物品	浏览所有捐赠物品数量
A9		发布评论	发布物品评论

A10		查看评论	查看用户对物品的评价
B1	捐赠	信息填写	捐赠前信息填写
B2		发布捐赠	发布已填写的捐赠
C1	回收	开始寄件	输入物品信息, 开始回收
C2		信息确认	确认用户输入信息
C3		确认回收	确认回收物品信息
C4		文件上传	上传物品介绍图
D1	维修	信息输入	输入物品维修信息
D2		开始维修	确认信息, 开始维修
D3		进度查询	查询已维修进度
E1	登录	密码登录	通过密码登录
F1	退出		退出账户
G1	我的	个人信息	查阅个人信息
G2		积分	查询已有积分信息
G3			积分兑换物品
G4			输入收货信息
G5		订单	查询已有回收订单
G6			查询已购买订单
G7			查询已发货订单
G8		购物车	查询购物车信息
G9			删除已添加物品
G10		维修商品	查询已维修物品信息
H1	搜索	全文检索	搜索商品信息

3.1.3 后台管理端功能需求分析

后台管理端功能需求分析如表 3-2 所示。

表 3-2 后台管理端功能需求分析表

需求编号	模块	功能	需求描述
J1	DashBoard	数据概览	查询当前系统订单、物品、交易数据
J2	用户管理	用户与角色管理	管理已有用户与角色数据
J3		物流人员管理	管理已注册物流人员信息

J4		维修人员管理	管理维修人员信息
J5		用户地址管理	管理用户已有地址信息
J6		用户积分管理	管理已有用户积分信息
J7		积分等级管理	管理用户等级信息
J8		评论管理	管理当前评论信息
Q1	系统管理	角色与权限管理	管理角色与角色的权限
Q2		菜单管理	管理系统菜单
K1	数据管理	轮播图列表	发布轮播图数据
K2			删改轮播图数据
K3		回收物品分类	回收物品分类管理
K4		捐赠物品管理	用户已捐赠物品管理
K5		维修物品管理	管理用户发布的维修物品信息
L1	订单	订单管理	管理订单信息
M1	活动	活动管理	发布活动信息
M2			上传活动介绍图
N1	积分商城	积分物品管理	积分商城物品发布与管理
N2		物品积分设置	设置发布物品兑换所需积分
O1	登录	密码登录	用户登录与鉴权
P1	退出		用户退出, 清除 cookie

3.2 性能需求分析

本系统对性能的需求主要涵盖如下方面:

- 数据精确度: 在用户对数据进行操作时, 确保不出现误删、重复插入等数据冗余情况;
- 时间特性: 系统执行用户请求时, 响应时延保持在 2 秒内;
- 兼容特性: 移动端 App 应具有良好的系统兼容性, 应具备良好的跨平台特性。

3.3 安全需求分析

安全需求主要包括以下内容：

- 身份与权限验证：在用户登录时对用户身份信息进行检验，严格控制用户访问权限（采用动态菜单进行权限管理）；
- 数据加密：针对系统产生数据进行严格加密存储；
- 数据完整性：针对系统用户上传下载文件进行完整性校验，确保数据准确无误；
- 敏感数据访问控制：对于用户隐私信息，进行严格访问控制，严防用户数据泄露。

3.4 软硬件需求分析

1、服务器软硬件要求：如表 3-3 硬件要求表所示。

表 3-3 服务器软硬件要求表

操作系统	CentOS 8.2 及以上；
内存	大于 16GB；
运行环境要求	安装 JDK\JRE、node.js、maven 等基础环境；
CPU 主频	基准主频高于 3.5GHz；
硬盘容量	剩余空间大于 256GB 及以上；

2、移动端软硬件要求：如表 3-4 所示。

表 3-4 移动端软硬件要求

操作系统	Android 9.0 及以上或 IOS 14 及以上
CPU	主频 2.0GHz
内存	大于 8GB
浏览器	Chrome、Edge 等
外部存储	剩余空间大于 2GB

3、管理端与门户端软硬件要求：PC 端使用一般个人电脑即可，浏览器推荐 Chrome 浏览器，以此获得最佳体验。

3.5 本章小结

本章主要介绍了本系统的各个平台需求分析，分别从以下四个方面入手：功

能性、性能、安全性、软硬件进行了分析。通过前文分析进一步明确了系统的需实现的功能。

第四章 系统设计

4.1 系统概要设计

4.1.1 总体设计

(一)、系统总体结构设计：

本系统总体如图 4-1 所示，本项目主要分为三大平台，主要分为十二个模块，其中：移动端与门户端功能类似（门户端增加购物车模块），故在图中用移动端代替二者。

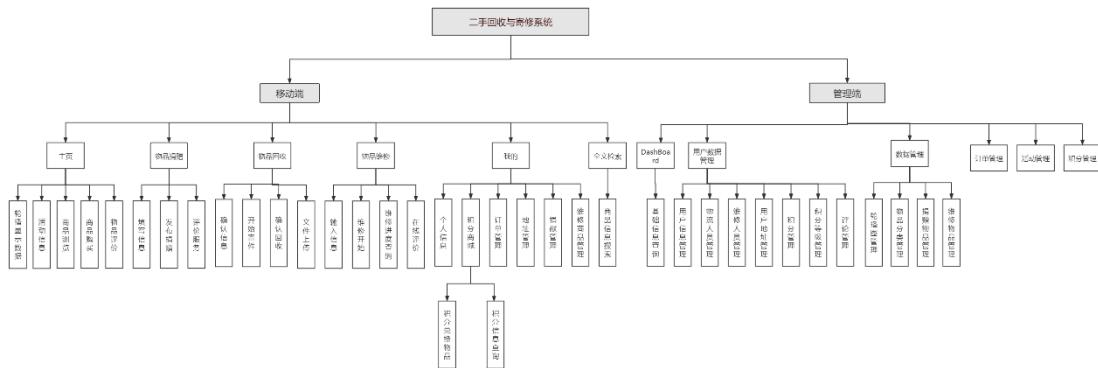


图 4-1 系统结构图

(二)、系统总体架构设计：

如图 4-2 展示了本系统的架构，图中将系统分为技术层、业务层、运行支持层、服务应用层、数据层、运行环境、服务层七层结构。在技术层中主要展示了系统开发过程中所使用的技术，在业务层阐述了本系统实现的大体功能，在运行层中展示了 Nginx 和 CDN 技术等，主要被用于做接口转发。服务应用层中阐述了本系统应用的基础工具，数据层和数据库层阐述了本系统数据存储的方式、数据存储的工具等。在运行环境层和服务层阐述了本系统部署和开发时的环境。

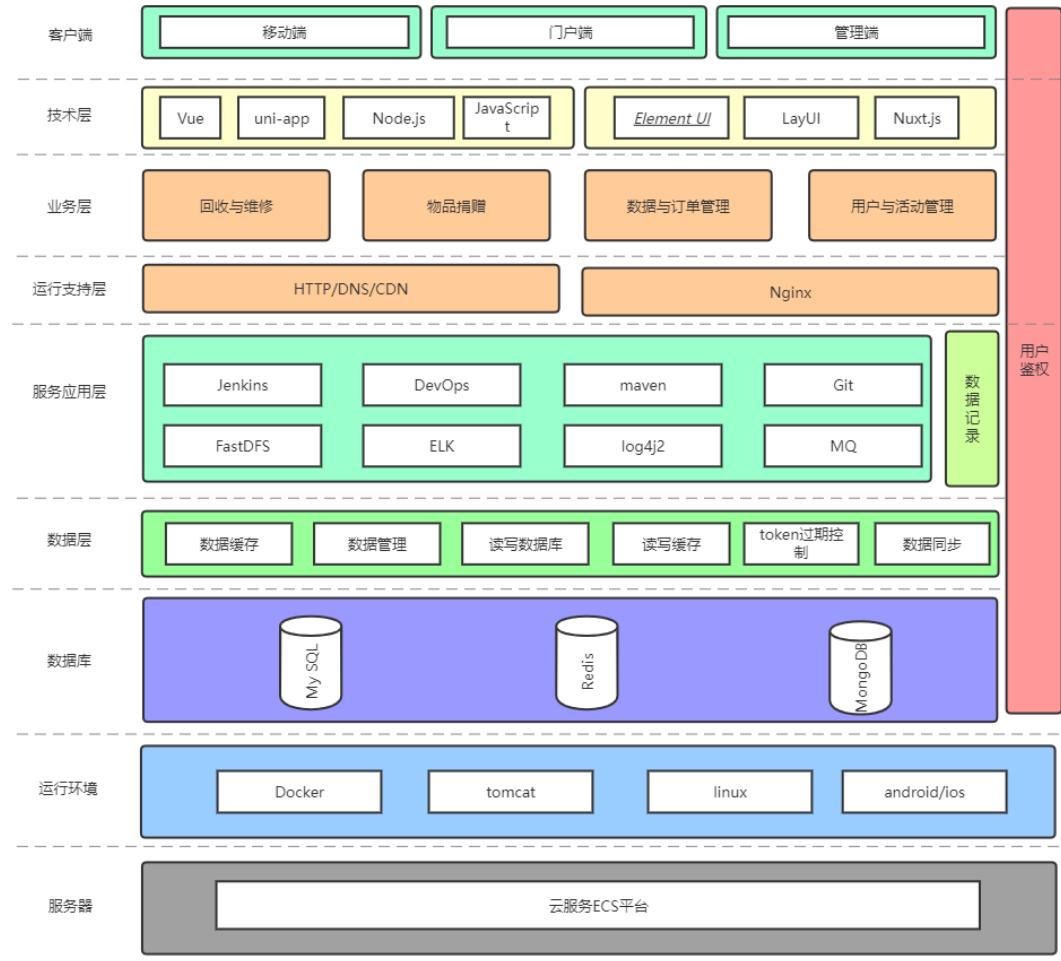


图 4-2 系统架构图

4.1.2 设计流程

(一)、移动端与门户端设计流程

在移动端与门户端，主要有分为以下模块：主页、捐赠、回收、维修、登录注册、我的、搜索、购物车（门户端独有）、物品浏览与购买。具体模块功能设计如下：

1、主页设计

主页面主要包括以下功能：轮播信息展示、活动浏览、活动查看详情等功能。

2、捐赠功能设计

此功能主要设计为：信息填写、捐赠物品的发布、评价服务三个流程。通过填写欲捐赠物品信息、完成捐赠物品后，可以获得对应的积分奖励。获得积分后，用户可通过积分兑换功能相应奖励物品。

3、回收功能设计

回收功能被设计为四个小的模块，分别由信息确认、寄件、确认信息、文件上传组成。通过输入物品信息、上传商品介绍图、填写物流信息、确认所填信息完成二手物品的发布。

4、维修功能设计

维修功能主要设计为待维修产品的预约信息的填写、开始维修、维修进度查询、在线评价功能。通过此项功能的设计，可以使得用户更加方便的进行产品维修，提神用户体验。

5、登录注册功能设计

登录注册功能为密码登录、手机号登录，第三方授权登录三种方式。通过设计丰富的登录方式极大地提升了用户的使用体验。

6、我的模块功能设计

此模块主要设计以下功能：个人信息查询；积分商城查询积分信息、通过已有积分兑换商品；查看我的订单信息、查看订单详情；查看我的地址、新增地址，修改或删除已有地址；查看已捐赠的物品信息；查看我的维修订单信息、修改维修订单信息。

7、搜索功能设计

此模块主要被设计为可搜索全文的商品信息，在搜索出结果后点击即可进入商品详情页。

8、购物车功能设计

此模块主要可添加中意二手商品，也可进入购物车页面删除物品。

9、商品浏览与购买功能设计

用户可在门户或 app 主页点击浏览商品，可选择购买物品或添加至购物车。

(二)、管理端功能设计

在管理端，主要包括以下功能模块：DashBoard、用户管理、系统管理、数据管理、订单管理、活动管理、积分商城、登录注册与退出。具体功能设计如下：

1、DashBoard 设计：

此功能被设计为多种信息的查询，包括：订单成交量查询，已发布物品总数查询、交易数量查询等。

2、用户管理功能设计：

用户管理主要设计以下功能：用户信息的管理、物流人员信息管理、维修人员信息管理、用户地址信息管理，用户积分管理、积分等级管理、用户评论管理。上述管理功能主要包括增删改查四个方面。

3、系统管理功能设计：

系统管理主要管理管理端角色、菜单、权限信息，主要设计以下功能：角色与角色具有的权限管理、系统菜单管理。上述管理功能主要包括增删改查四个方面。

面。

4、数据管理功能设计：

数据管理功能主要由以下部分组成：轮播图的发布与修改、删除；回收物品分类的管理、用户已捐赠物品的管理、用户已发布维修物品的管理。轮播图管理被设计为可以管理移动端显示的轮播图数据的功能。其他功能分别分别设计为增删改查数据四个方面。

5、订单管理设计

在商城管理系统中、订单管理尤为重要。本系统将订单管理单独列出，包含有订单状态管理、查看订单信息、取消订单等功能。

6、活动管理设计

活动管理中主要设计为促销活动的发布，已发布活动的编辑，活动的取消，为活动添加宣传图片等功能。

7、积分商城设计

积分商城功能设计为主要包括：已发布可积分兑换物品的管理，修改已发布物品兑换所需积分、查看积分可兑换物品信息等功能。

8、登录注册、登出设计

本系统登录设计为单点登录方案，采用 Spring Security Oauth2.0 认证授权方案，通过 Redis 存储用户认证信息。

（三）、后台服务设计

后台服务主要采用微服务的设计思想，将整个服务切割为：`rs_center_service`、`rs_user_service`、`rs_auth_service`、`rs_gateway_service`、`rs_search_service`、`rs_manage_service`、`rs_points_service`、`rs_order_service`、`rs_filesystem_service` 微服务构成。其中 `rs_center_service` 为注册中心，负责处理存储微服务的状态信息；`rs_gateway_service` 被称为网关，负责转发前端 API 请求，起到过滤、转发、负载均衡效果。下文中将对前文中未提到的微服务做简单介绍。

1、`rs_search_service`

搜索微服务：主要被设计为处理来自各平台的搜索请求，并从后台获取搜索结果返回至前台显示。

2、`rs_auth_service`

鉴权微服务：主要设计为处理用户登录、注册、登出请求处理的微服务，负责用户权限校验、信息存储、令牌管理等操作的处理。

3、`rs_suer_service`

用户微服务：主要设计为负责处理用户基础信息的管理：包括个人信息、地址信息、评论信息等，同时也包括物流人员、维修人员个人信息管理。

4、`rs_manage_service`

管理微服务：此微服务设计为管理用户发布的捐赠、维修、二手商品信息、还包括活动的发布与管理、系统数据字典的管理等功能，主要与 MySQL 数据库进行交互。

5、rs_points_service

积分微服务：主要设计为管理用户的积分信息、管理已有的积分等级信息，负责处理积分商城物品数据的管理。

6、rs_order_service

订单微服务：主要设计为管理移动端用户产生的订单数据，对于用户交易数据进行处理，系统实时性要求较高。

7、rs_filesystem_service

文件系统微服务：主要设计为处理上传文件的存储管理服务。将接收的文件信息进行记录存入 MongoDB 后，最后将文件上传至 FastDFS 中予以保存，返回微服务存储路径，并更新至 MongoDB 数据库。

4. 1. 3 接口设计

(一)、外部接口

1、用户界面设计

本系统用户界面设计主要分为移动端界面设计、管理端和门户端界面设计。

在移动端原型设计阶段采用以 Axure 为代表的原型设计工具进行设计，在开发中采用基于 Vue.js 的 uView UI 框架进行页面组件化设计；

在管理端设计过程中采用基于 Vue.js 的 Element UI 框架进行组件化设计；

在门户端设计过程中采用基于 Vue.js 的 Element UI 和 LayUI 框架进行组件化设计。

2、软件接口设计

软件接口包括服务端与前台之间的通信、服务器与数据库、中间件之间通信的接口。在服务端与前台通信中通常采用 API(Application Programmer Interface) 进行通信。服务器与数据库、服务器与中间件之间使用 TCP/IP 协议进行通信。

(二)、内部接口

内部接口通常指暴露系统于微服务之间，微服务之间通过远程过程调用(RPC) 调用接口。一般在系统之间通常采用中间件进行通信。例如常用中间件有：RabbitMQ、Kafka 等。

4. 1. 4 数据结构设计

(一)、关系型数据库(MySQL)表设计

在本系统中，数据库主要分为交易数据数据库与用户数据数据库。两者数据相互隔离，分布式存储确保数据的安全性，下文中将对数据库表作详细设计。

1、交易数据数据库(recycle_shop)

本数据库主要负责课程数据的管理。具体表设计如下：

表 4-1 活动表(activity)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	活动 id	varchar	32	0	✓	✓
submit_id	发布者 id	varchar	32	0	✓	✗
activity_name	活动名	varchar	32	0	✗	✗
time	活动时间范围	varchar	32	0	✗	✗
content	活动内容	varchar	255	0	✗	✗
status	活动状态	varchar	10	0	✗	✗
remark	活动备注	varchar	255	0	✗	✗
create_time	创建时间	datetime	0	0	✗	✗
update_time	修改时间	datetime	0	0	✗	✗

表 4-2 轮播图数据表(adsense)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
src	图片路径	varchar	255	0	✓	✗
url	URL	varchar	255	0	✗	✗
type	图片类型	varchar	32	0	✓	✗
create_time	创建时间	datetime	0	0	✓	✗

表 4-3 数据字典表(dict)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
code	字典代码	varchar	255	0	✓	✗
content	字典内容	varchar	255	0	✗	✗
type	类型	varchar	255	0	✗	✗
status	状态	varchar	255	0	✓	✗
create_time	创建时间	datetime	0	0	✓	✗

update_time	修改时间	datetime	0	0	×	×
-------------	------	----------	---	---	---	---

表 4-4 捐赠物品信息表(donate_goods)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
goods_category_id	物品分类 id	varchar	32	0	√	×
user_id	用户 id	varchar	0	0	×	×
name	物品名	varchar	32	0	√	×
count	物品数	int	32	0	√	×
status	物品状态	varchar	32	0	√	×
create_time	创建时间	datetime	0	0	√	×
update_time	更新时间	datetime	0	0	×	×

表 4-5 物流信息表(express)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
order_id	订单 id	varchar	32	0	√	×
user_id	用户 id	varchar	255	0	×	×
goods_id	物品 id	varchar	32	0	√	×
deliver_user_id	物流人员 id	varchar	32	0	√	×
delivey_time	预约时间	varchar	32	0	√	×
is_express	物流方式	varchar	32	0	√	×
express_no	物流单号	varchar	32	0	×	×
create_time	创建时间	datetime	0	0	×	×
status	状态	varchar	32	0	×	×
type	物流类型	varchar	255	0	×	×

表 4-6 维修物品表(fix_goods)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
name	物品名	varchar	64	0	√	×
submit_id	发布者 id	varchar	32	0	√	×

goods_order_id	物品订单 id	varchar	32	0	✓	✗
comment_id	评价 id	char	32	0	✗	✗
fix_function	维修方式	varchar	255	0	✗	✗
address	地址信息	varchar	32	0	✗	✗
content	物品属性描述	varchar	32	0	✗	✗
phone	联系方式	varchar	32	0	✗	✗
deliver_time	邮寄时间	datetime	0	0	✓	✗
update_time	更新时间	datetime	0	0	✗	✗

表 4-7 商品评价表(goods_comment)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
goods_id	物品 id	varchar	32	0	✓	✗
parent_id	回复 id	varchar	128	0	✓	✗
name	评价用户	varchar	255	0	✓	✗
content_text	评价内容	varchar	255	0	✓	✗
all_reply	回复数	int	0	0	✓	✗
content	评价详情	varchar	255	0	✗	✗
like_num	点赞数目	int	0	0	✗	✗
date	评价日期	datetime	0	0	✗	✗
url	商品 URL	varchar	255	0	✗	✗

表 4-8 商品图片表(goods_pic)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
user_id	用户 id	varchar	32	0	✓	✗
goods_id	物品 id	varchar	32	0	✗	✗
src	路径	varchar	8	2	✗	✗
status	状态	varchar	32	0	✗	✗
type	图片类型	varchar	32	0	✗	✗
upload_time	上传时间	datetime	0	0	✗	✗

表 4-9 积分等级表(grade)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
grade	等级	int	0	0	×	×
max_value	最大值	int	0	0	×	×
min_value	最小值	int	0	0	×	×
status	状态	int	0	0	×	×
create_time	创建时间	datetime	0	0	×	×
update_time	更新时间	datetime	0	0	×	×

表 4-10 积分物品表(grade_goods)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
goods_category_id	物品分类 id	varchar	32	0	×	×
grade_goods_name	物品名	varchar	32	0	×	×
grade_count	兑换所需积分	int	0	0	×	×
good_count	物品总数	int	0	0	×	×
status	状态	varchar	32	0	×	×
create_time	创建时间	datetime	0	0	√	×
update_time	更新时间	datetime	0	0	√	×

表 4-11 订单表(order_list)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
user_id	用户 id	varchar	32	0	√	×
goods_id	商品 id	varchar	32	0	√	×
address_id	地址 id	varchar	32	0	√	×
express_name	物流属性名称	varchar	32	0	√	×
order_status	订单状态	varchar	32	0	√	×
product_count	订单数目	double	11	0	×	×
goods_count_price	物品价格	double	255	0	×	×
order_amount_total	订单总额	double	255	0	×	×

pay_channel	支付渠道	varchar	255	0	×	×
content	内容	varchar	255	0	×	×
order_settlement_status	订单状态	varchar	255	0	×	×
order_settle_time	订单成交时间	datetime	0	0	×	×
create_time	创建时间	datetime	0	0	×	×
update_time	更新时间	datetime	0	0	×	×
delivey_time	交付时间	datetime	0	0	√	×
send_time	发货时间	datetime	0	0	√	×

表 4-12 积分表(points)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
userid	用户 id	varchar	32	0	√	×
name	积分名	varchar	32	0	×	×
grade	积分等级	int	0	0	×	×
count	积分数目	int	0	0	×	×
status	交易状态	varchar	32	0	√	×
create_time	创建时间	varchar	255	0	×	×
update_time	更新时间	varchar	255	0	×	×

表 4-13 回收物品表(recycle_goods)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
name	物品名	varchar	64	0	×	×
submit_id	卖家 id	varchar	128	0	√	×
category_id	分类 id	varchar	11	0	√	×
weight	重量	double	128	0	√	×
old_price	原价	double	128	2	×	×
single_price	现价	double	128	2	×	×
count	总数	int	0	0	×	×
count_price	总价	double	32	0	×	×
status	企业状态	varchar	32	0	×	×

create_date	创建时间	datetime	0	0	×	×
update_date	更新时间	datetime	0	0	×	×
postage_price	邮费	double	0	0	×	×

表 4-14 回收物品分类表(recycle_goods_category)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
name	物品名	varchar	32	0	✓	✗
points	积分数目	int	0	0	✓	✗
status	状态	int	0	0	✗	✗
create_date	创建时间	datetime	0	0	✗	✗
update_date	更新时间	datetime	0	0	✗	✗

表 4-15 购物车表(shop_cart)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
name	物品名	varchar	32	0	✓	✗
img	物品图片	varchar	255	0	✓	✗
title	标题	varchar	255	0	✗	✗
price	价格	double	255	2	✗	✗
amount	数量	int	255	0	✗	✗
type	类型	varchar	255	0	✗	✗
user_id	用户 id	varchar	255	0	✗	✗
create_time	创建时间	datetime	0	0	✗	✗
update_time	更新时间	datetime	0	0	✗	✗
create_order_time	更新时间	datetime	0	0	✗	✗

2、用户信息数据库 (rs_user)

本数据库存储用户的基础信息，具体表设计如下：

表 4-16 评论信息(comment)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	评论 id	varchar	32	0	✓	✓

天津中德应用技术大学 2022 届本科生毕业设计

user_id	用户 id	varchar	32	0	✓	✗
rating	评价等级信息	varchar	32	0	✓	✗
content	评价内容	varchar	255	0	✓	✗
status	评论状态	varchar	32	0	✗	✗
create_time	评论时间	datetime	0	0	✗	✗
update_time	修改时间	datetime	0	0	✗	✗

表 4-17 物流人员信息表(deliver_user)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
rg_id	物品 id	varchar	32	0	✓	✗
order_id	订单 id	varchar	500	0	✗	✗
user_id	用户 id	varchar	32	0	✓	✗
name	物流人员姓名	varchar	32	0	✓	✗
tel	手机号	varchar	32	0	✓	✗
status	账户状态	varchar	32	0	✓	✗
register_time	注册时间	datetime	0	0	✗	✗
update_time	更新时间	datetime	0	0	✓	✗

表 4-18 维修人员表(fix_user)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
fix_goods_id	维修物品 id	varchar	32	0	✓	✗
name	维修物品名	varchar	128	0	✓	✗
phone	手机号	varchar	128	0	✓	✗
sex	性别	varchar	128	0	✓	✗
age	年龄	varchar	32	0	✗	✗
status	账户状态	varchar	32	0	✗	✗
register_time	注册时间	datetime	0	0	✗	✗
update_time	更新时间	datetime	0	0	✗	✗

表 4-19 用户表(rs_user)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
username	用户 id	varchar	45	0	√	×
password	姓名	varchar	96	0	√	×
salt	加盐	varchar	45	0	×	×
name	个人简历	varchar	45	0	√	×
userpic	照片	varchar	255	0	×	×
utype	账户类型	varchar	32	0	√	×
birthday	生日	datetime	0	0	×	×
sex	性别	char	1	0	×	×
email	邮件	varchar	45	0	×	×
phone	手机号	varchar	45	0	×	×
qq	QQ	varchar	32	0	×	×
status	用户状态	varchar	32	0	√	×
create_time	创建时间	datetime	0	0	√	×
update_time	更新时间	datetime	0	0	×	×

表 4-20 用户权限表(rs_permission)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
role_id	权限 id	varchar	32	0	√	×
menu_id	菜单 id	varchar	255	0	√	×
create_time	创建时间	datetime	0	0	×	×

表 4-21 用户角色表(rs_role)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	√	√
role_name	权限名称	varchar	255	0	×	×
role_code	菜单代码	varchar	255	0	×	×
description	属性描述	varchar	255	0	×	×
create_time	创建时间	datetime	0	0	×	×

update_time	创建时间	datetime	0	0	×	×
status	角色状态	char	1	0	✓	✗

表 4-22 用户角色表(rs_user_role)

名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
user_id	用户 id	varchar	32	0	✗	✗
role_id	权限 id	varchar	32	0	✗	✗
create_time	创建时间	datetime	0	0	✗	✗
creator	操作者	varchar	255	0	✗	✗

表 4-23 系统菜单表(rs_menu)

属性名称	属性描述	属性类型	属性长度	小数点	是否非空	是否主键
id	id	varchar	32	0	✓	✓
code	代码	varchar	32	0	✗	✗
p_id	父 id	varchar	32	0	✗	✗
menu_name	菜单名	varchar	32	0	✗	✗
url	URL	varchar	255	0	✗	✗
is_menu	是否菜单	varchar	255	0	✗	✗
level	等级	int	0	0	✗	✗
sort	排序	int	0	0	✗	✗
status	状态	char	1	0	✗	✗
icon	图标	varchar	255	0	✗	✗
create_time	创建时间	datetime	0	0	✗	✗
update_time	更新时间	datetime	0	0	✗	✗

除上述所述数据库表外，系统中还有其他生成的数据库表：
oauth_accesstoken、oauth_approvals 等；一般用于鉴权微服务的用户鉴权的使用，在系统中使用较少故不再阐述。

(二)、非关系型数据库设计

在本系统中，将文件的索引信息使用非关系型数据库 MongoDB 存储，将用户令牌信息存储在非关系型数据库 Redis 中。在 MongoDB 中主要分为以下集合进行存储：

- filesystem 集合：存储文件索引信息；
- fs.chunks 集合：文件分块标识信息；
- fs.files 集合：分块文件内容；

4.2 系统详细设计

4.2.1 总体设计

(一)、系统开发环境

1、软件环境

表 4-24 软件环境

分类	名称	版本号
操作系统环境	Microsoft Windows 11	22000.120
关系型数据库客户端	Navicat™ for MySQL	11.1.13(64bit)企业版
非关系型数据库客户端	Studio 3T for MongoDB	2021.4.0
虚拟容器	Docker	20.10.7
数据库	Redis	6.0.6
数据库	MySQL	5.6
数据库	MongoDB	4.2
文件存储	FastDFS	3.04
搜索	ELK	7.10.1
服务器操作系统	CentOS	8.2
管理端与门户端开发环境	JetBrains WebStorm	2019.1x64
移动端开发环境	Hbuild X	3.2.16.20211122
后端开发环境	IntelliJ IDEA	2020.1x64

2、硬件环境

表 4-25 硬件环境

分类	名称	配置
开发系统	CPU	Intel Core i5-7300HQ CPU@2.50GHz
	内存	16.0GB
	硬盘	128GB+1TB
服务器	CPU	Intel Xeon 8core
	内存	16GB
	硬盘	80GB

(二)、数据库结构设计

在上述分析中,我们详细的讨论了本系统的数据库表结构。在接下来论述中,我们将分析 rs 和 rs_user 数据库的 E-R 模型。

1、E-R 图

(1). rs 数据库 E-模型

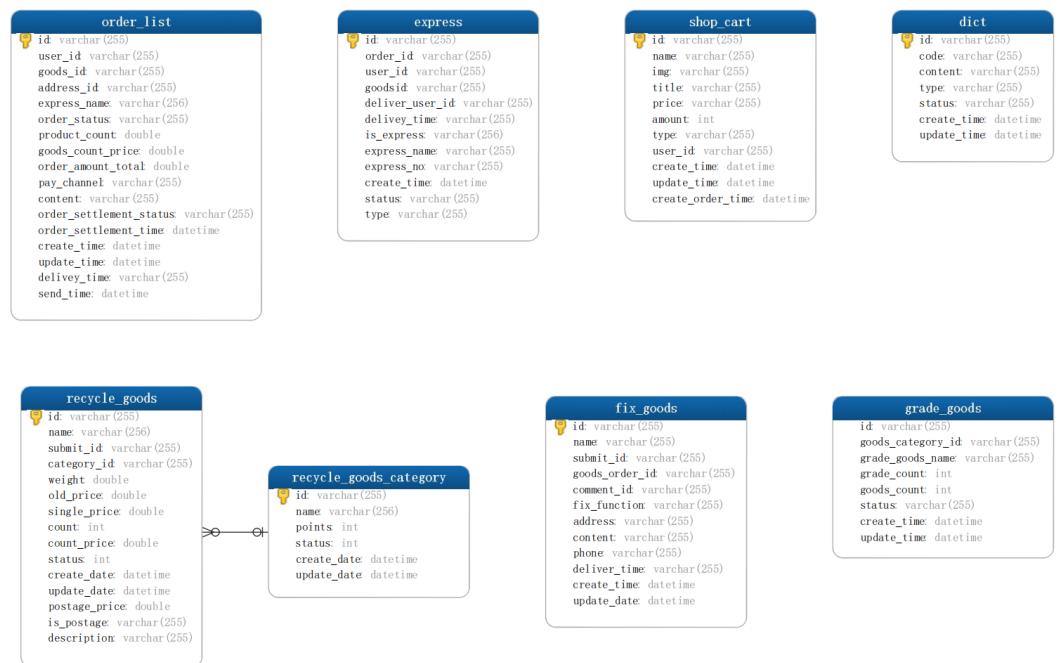


图 4-3 rs 数据库 E-R 图

(2). rs_user 数据库 E-R 模型

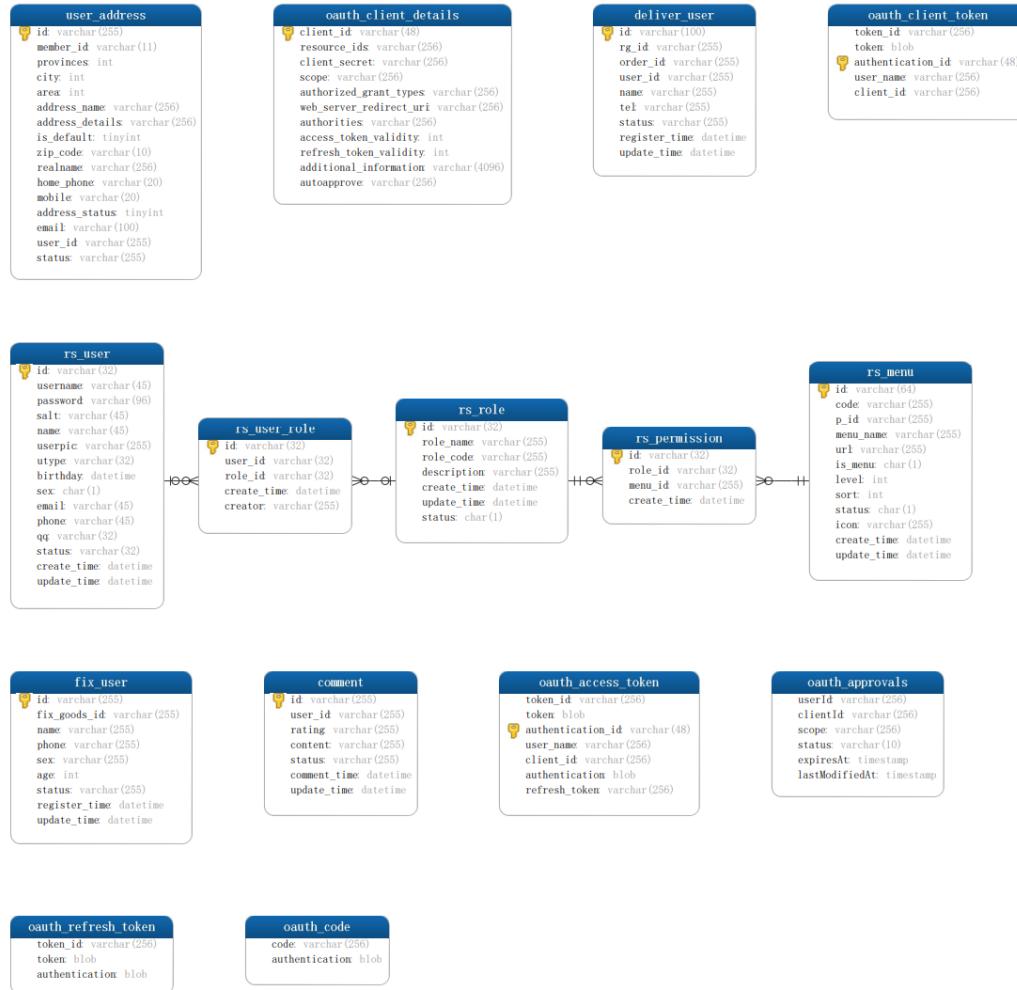


图 4-4 rs_user 数据库 E-R 模型

(三)、系统项目端口分配设计

本项目端口分配如表 4-26 所示。

表 4-26 系统端口分配

项目	端口号	项目	端口号
FastDFS	22122	rs_center_service	50101
MongoDB	27017	rs_gateway_service	50201
Nginx	80	rs_search_service	46000
ELK	9200	rs_auth_service	45000
MySQL	3306	rs_user_service	42000
图片服务器	8888	rs_manage_service	41000
Redis	6379	rs_order_service	43000
移动端	8080	rs_filesystem_service	22100

管理端	8080	rs_points_service	44000
门户端	8080		

4.2.2 模块设计

(一) 移动端与门户端功能设计流程

1、物品捐赠流程

如图 4-5 为物品捐赠流程，用户可点击开始捐赠进行捐赠。首先判断用户登录状态，如果用户未登录，则进入登录界面。输入捐赠物品信息后，确认捐赠的信息。在捐赠物品结束后，对服务进行评价。至此，捐赠物品流程结束。

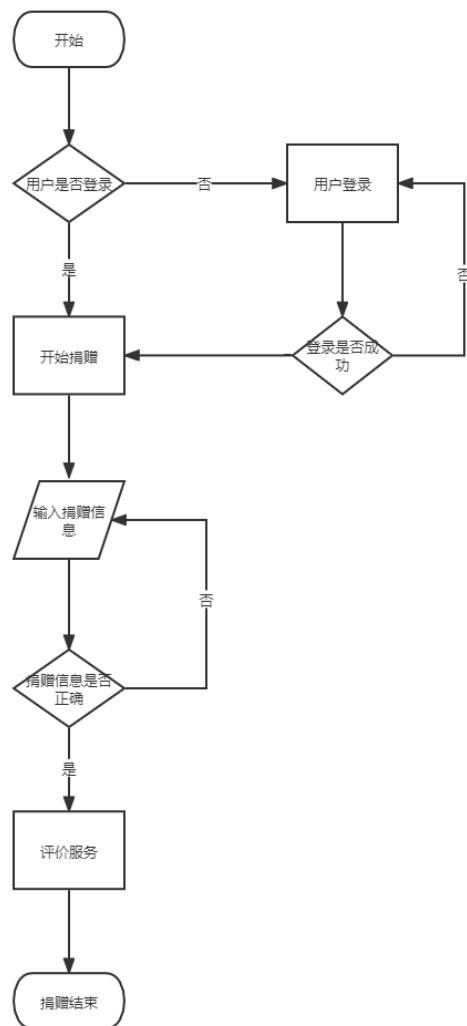


图 4-5 物品捐赠流程

2、物品回收流程

物品回收作为本系统的一大功能，如图 4-6 为具体流程：在判断用户是否登录成功后，开始输入物品信，上传商品介绍图；物品信息确认成功后，选择所需

物流方式，确认物流信息后，对服务进行评价。物品回收流程至此结束。

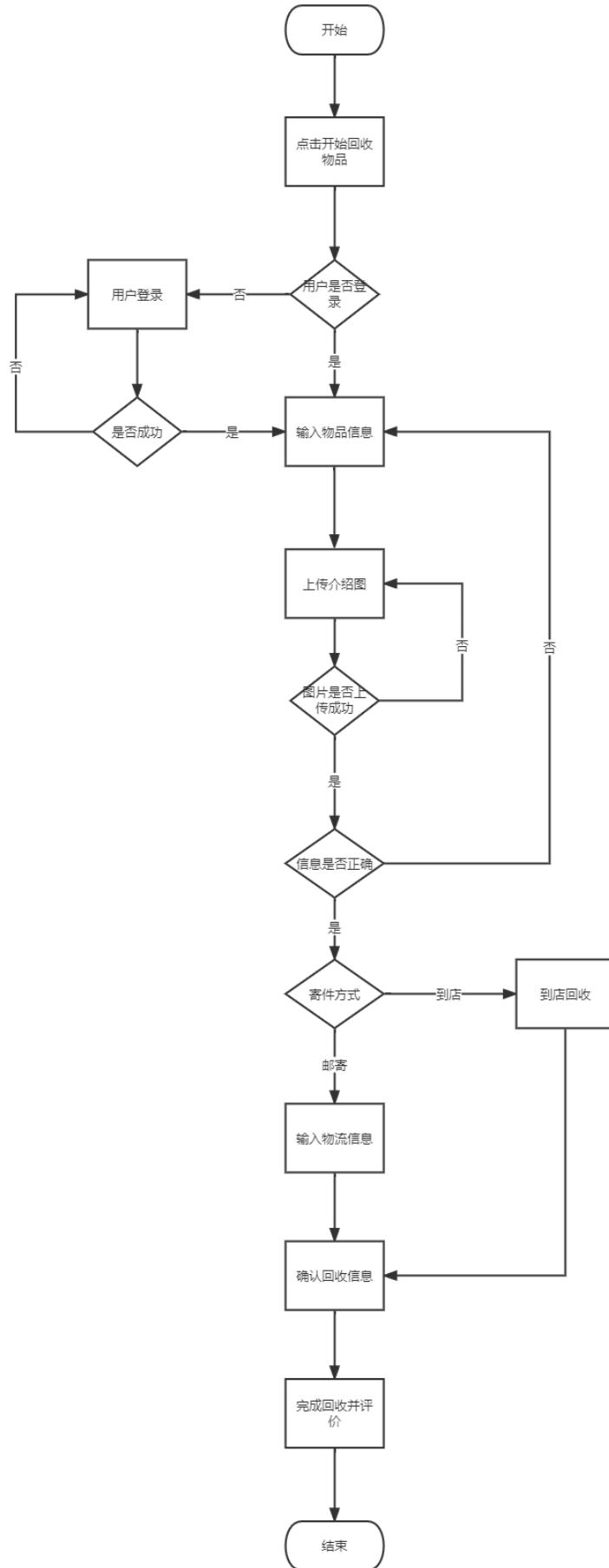


图 4-6 物品回收流程

3、维修物品发布流程

维修物品为移动端的重要功能。如图 4-7 所示，在判断结束用户登录状态后，首先用户输入待维修物品信息，确认维修价格，支付维修费用后可查询维修进度。在维修结束后，可对维修服务进行评价。

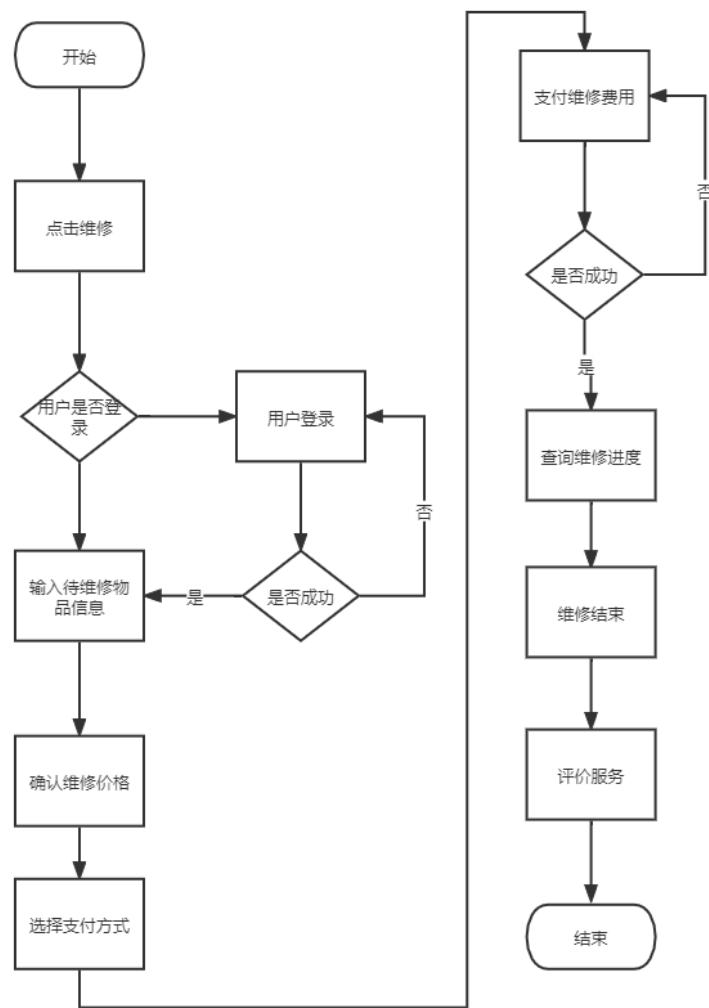


图 4-7 物品维修发布流程

4、积分兑换物品流程

在用户捐赠完成后，可以获取积分，通过获得积分兑换相应的奖励物品。如图 4-8 兑换具体流程如下：当用户进入积分商城，首先判断用户是否登录，用户登录后获取用户已有积分；当用户点击兑换物品按钮后，判断用户剩余积分是否满足兑换当前物品所需积分；满足所需积分后，提示用户填写对应物流信息，点击确认后，兑换物品流程结束。

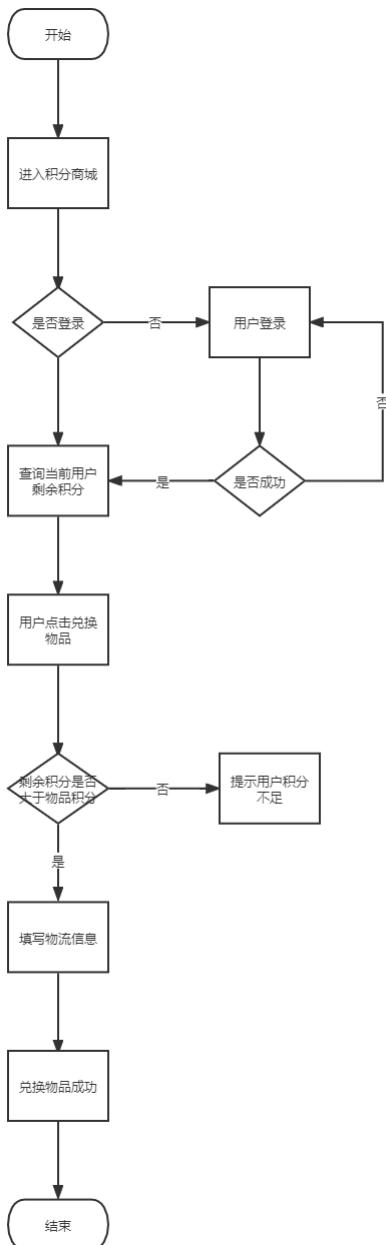


图 4-8 积分兑换商品流程

5、二手物品购买流程

二手交易的主要目的在于物品的交易。如图 4-9 为二手物品交易流程。当用户浏览商品后具备购买意向时，点击购买商品。系统首先判断用户是否登录。在确认用户登录后，提示用户选择个人收货信息、选择支付方式。根据选择支付方式完成支付后，卖家发货。至此二手物品购买流程结束。

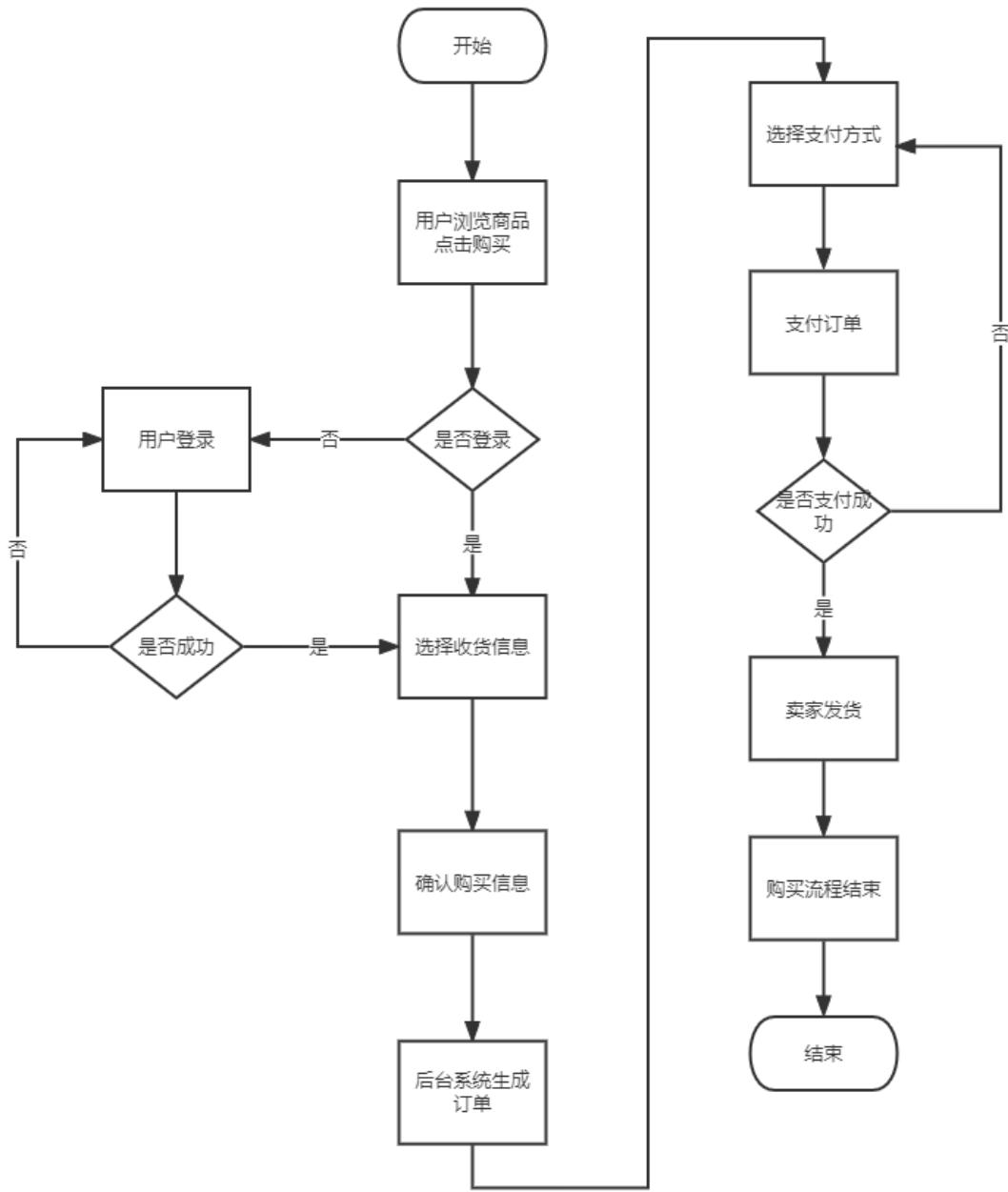


图 4-9 二手物品购买流程

(二)、管理端设计流程

在上述移动端设计流程之外，在后台管理端主要流程如下：

1、用户数据管理流程

在管理端，使用较多的功能为数据增删改查。数据管理大致流程如图 4-10 所示：通过接口请求后台服务，后台服务通过对数据库的操作，完成前端对应请求。最后将执行结果通过接口返回至前端。

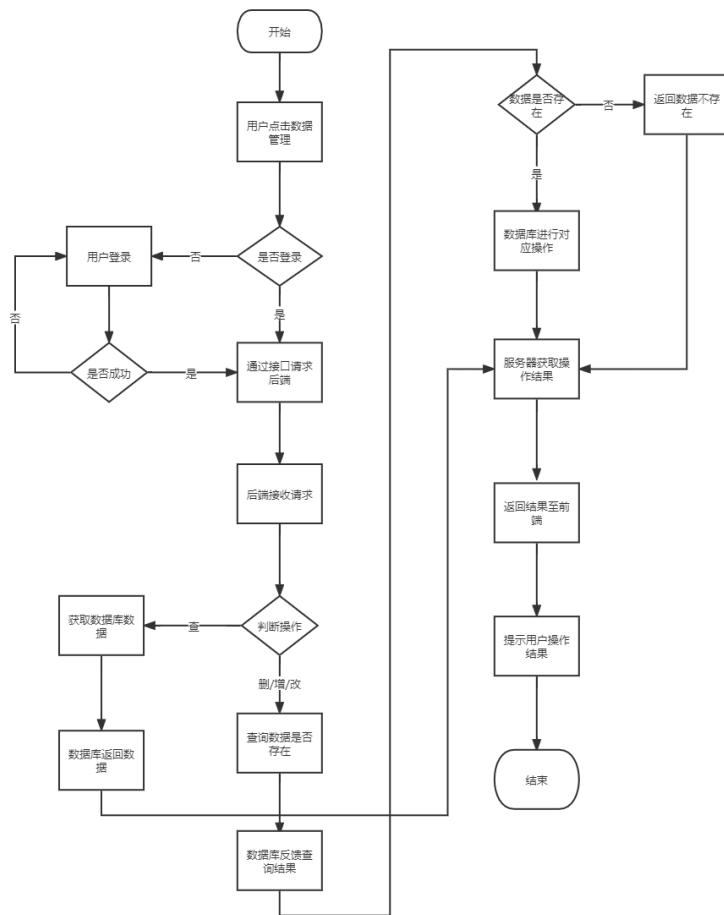


图 4-10 数据管理流程

2、活动发布流程

在管理端的基本数据管理流程介绍结束后，后文将对活动发布的具体流程作以介绍。如图 4-11 在活动发布前，首先获取并判断用户权限，符合操作权限后，点击发布活动。在输入活动信息、上传活动介绍图、确定活动起始时间后点击发布，后台服务处理请求并存储数据至数据库。最终返回操作结果至前端界面提示用户。

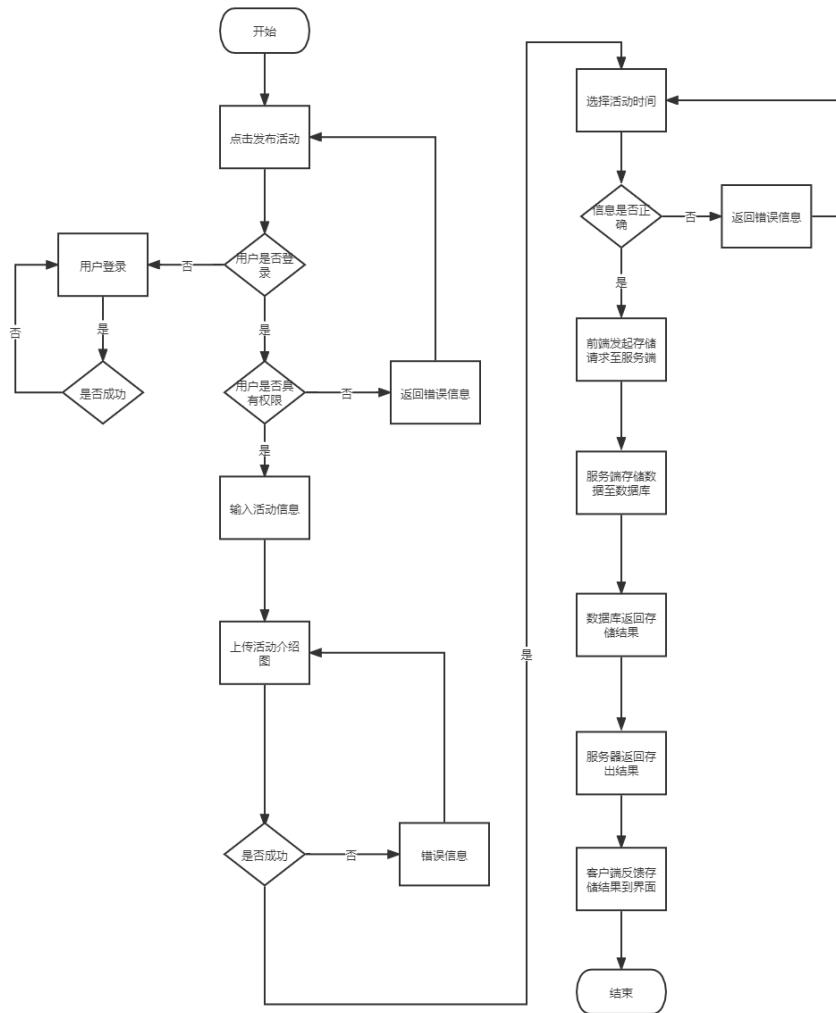


图 4-11 活动发布流程

(三)、服务后台设计流程

后台服务主要用于处理来自前端的接口请求，包括数据的基本操作、文件上传、用户登录请求的处理等流程。下文将对上述流程进行详细描述。

1、文件上传流程

如图 4-12 来自前端文件上传请求经过服务端处理后，服务端将文件信息记录至 MongoDB 数据库；在此之后服务端将文件存储上传至 FastDFS，服务端收到 FastDFS 的存储信息后存储至 MySQL 数据库。在存储成功后，服务端返回存储的文件信息至前端，最终提示用户上传是否成功。

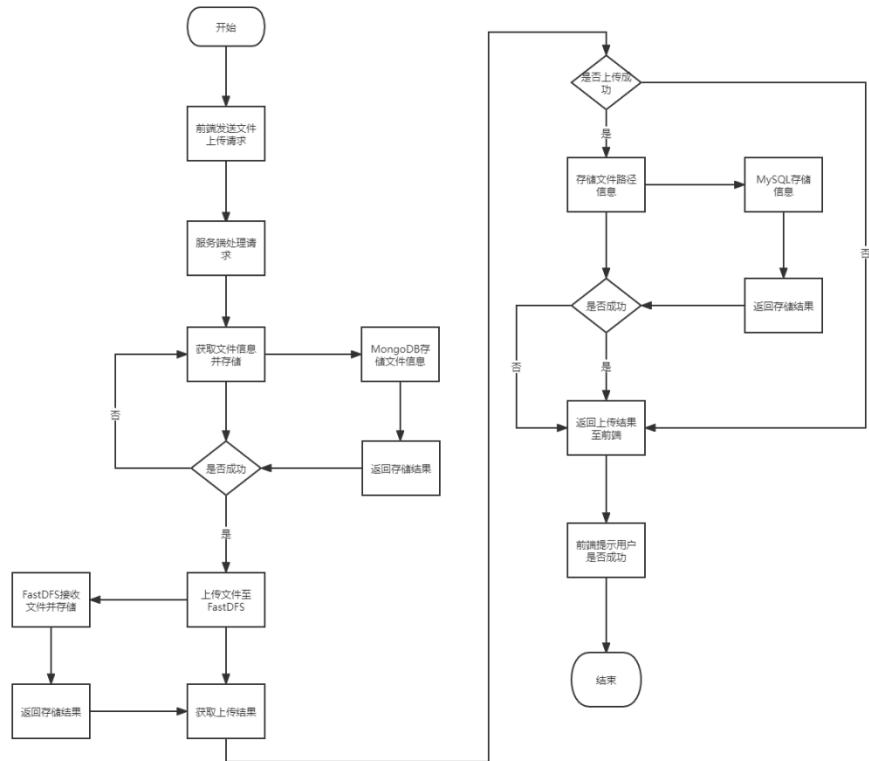


图 4-12 文件上传流程

2、基础数据管理流程

在前后端交互中，有许多请求都是用于数据的基本操作（增、删、改、查）。上述请求流程整体相似，在本部分将加以说明。如图 4-13 所示，当前端数据管理请求发送经接口发送至后端时，网关微服务解析接口请求并转发至对应微服务处理。微服务收到对应请求后进行处理并提取请求对应参数。服务端获取参数后，从数据库查询对应数据进行操作，数据库反馈操作结果至服务端。最终，服务端将数据处理结果反馈至前端并提示用户操作是否成功。

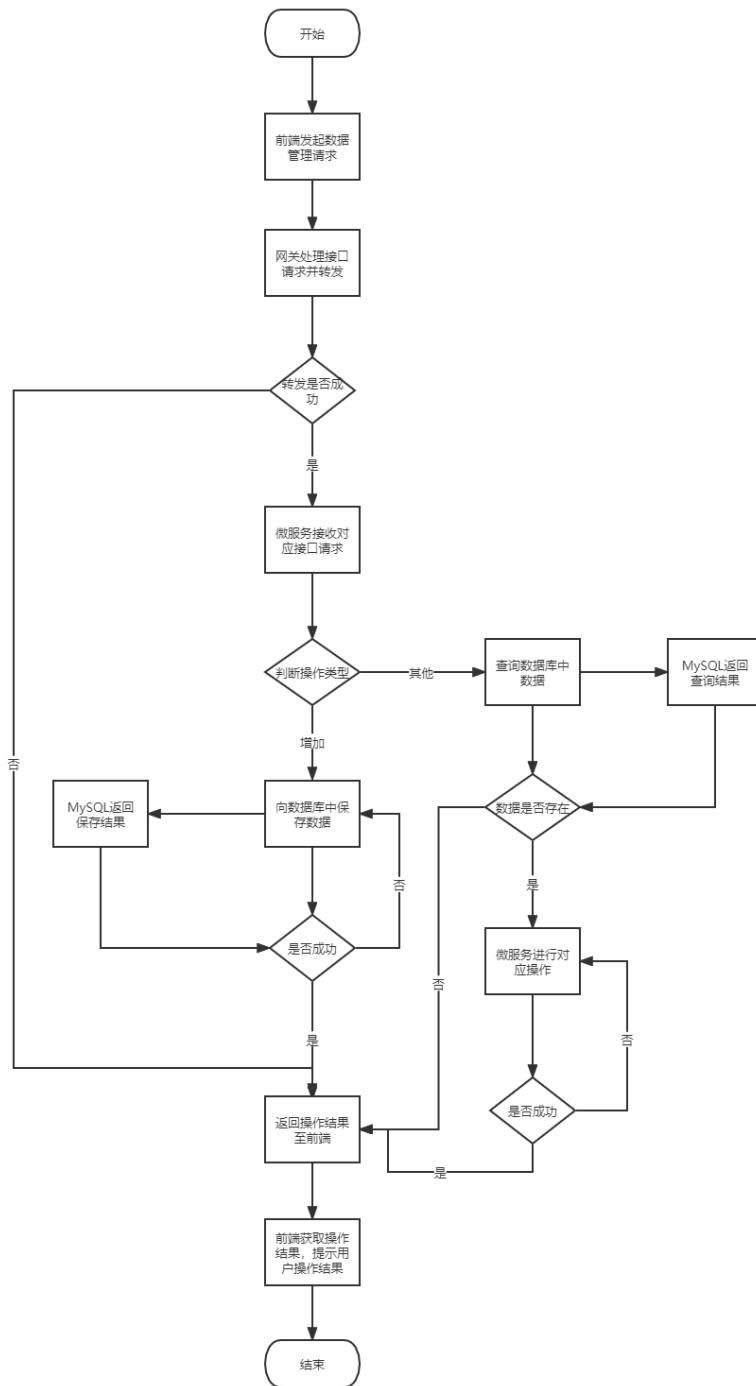


图 4-13 基础数据管理流程

3、登录流程

在当前软件开发中, 使用较多技术为单点登录技术。本系统移动端、管理端、门户端采用单点登录技术。如图 4-14 所示, 当用户点击输入个人信息点击登录后, 前端校验数据是否合法。当数据校验合法后, 前端发起登录请求。登录接口请求经网关微服务转发至鉴权微服务后, 鉴权微服务首先判断登录请求中 cookie 携带 token 参数是否合法。在参数合法后, 鉴权微服务发起远程调用请求 (RPC)

自身服务实例获取用户令牌。令牌获取成功后，解析对应的 token。将获取的 token 保存至远程 Redis 数据库中，保存成功后，获取令牌过期时间。令牌过期时间合法后，将令牌信息存入浏览器 cookie 中返回前端存储。

前端首先判断 token 存在后，请求获取服务端 jwt 令牌信息。后台微服务收到请求后，由 cookie 获取对应令牌信息。鉴权微服务根据对应令牌信息查询 Redis 中 token 是否存在并返回对应查询结果至前端。前端获取用户令牌信息后，解析对应数据。当数据解析完成后，提示用户登录结果。

登出流程类似于上文所述，当用户点击登出后，前端删除对应 cookie。之后前端发送请求通知微服务端删除 Redis 中的令牌信息。

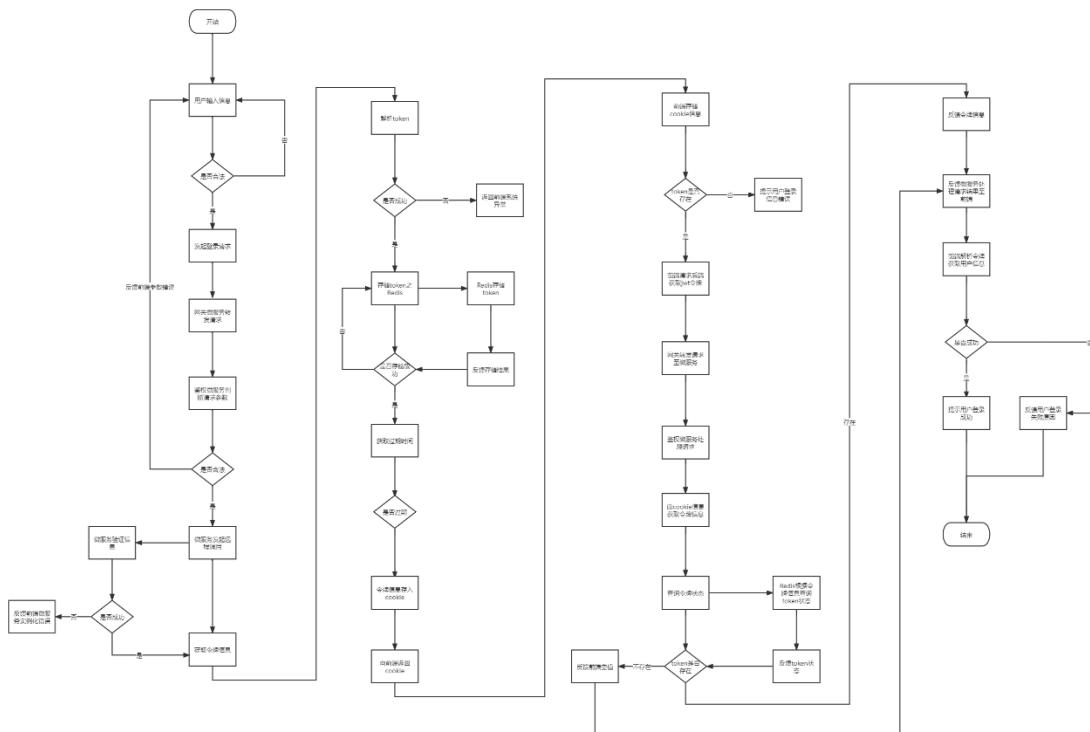


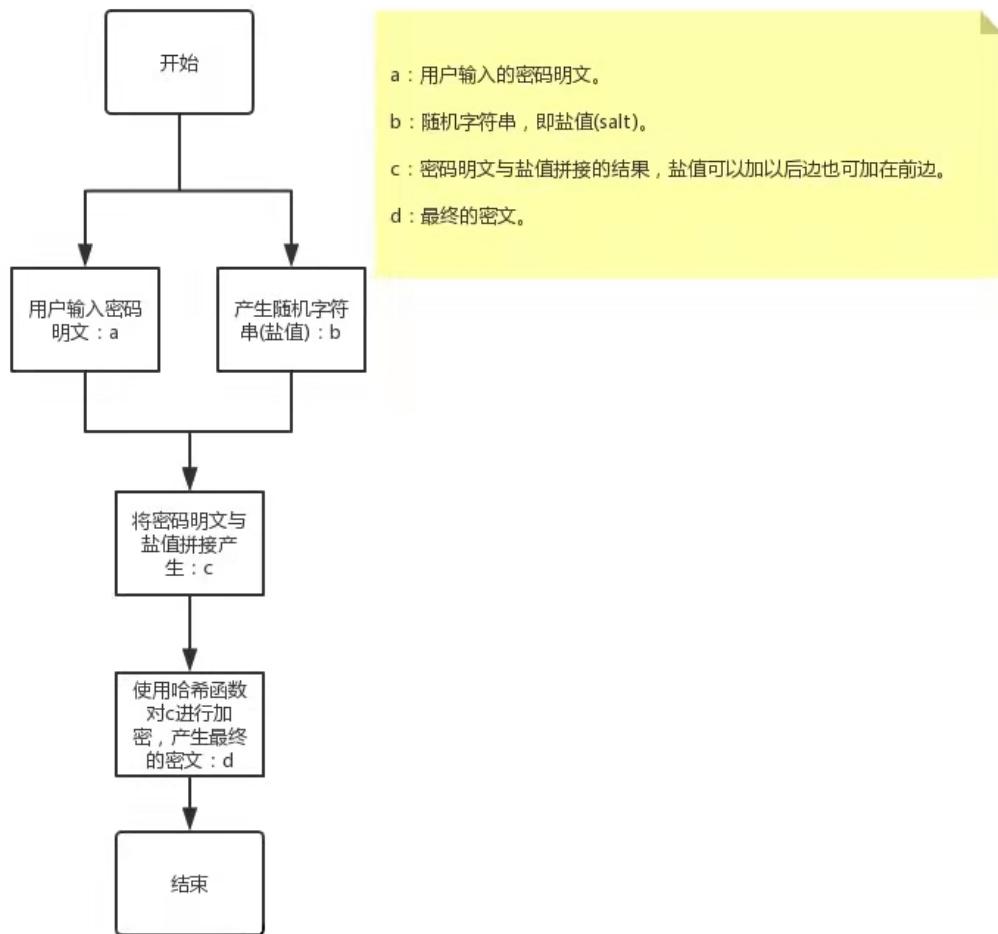
图 4-14 系统单点登录流程图

4.2.3 算法设计

在本系统中，算法设计主要集中于用户数据的加密存储方面。在密码存储方式的发展上，主要经过了三代的发展。第一代密码存储方式采用明文存储，存在诸多缺点：保密性差、存在数据泄露风险；第二代存储采取对称加密、非对称加密方式，主要采用 MD5、RSA 等加密方式。该种方式存在的主要缺点在于相同密码在数据库中存储相同，存在被撞库风险；第三代密码存储方式主要采用哈希算法加盐存储。此种加密方式有点在于：根据所加盐不同，用户相同密码存储内容不同，极大地降低了用户密码被撞库的风险。在此之外还可采取 BCrypt 或者 PBKDF2 增加入侵者破解密码的难度。通过第三代方式加密的数据，极大地增加了用户数据的安全性，在目前被广泛使用。

如下为加盐算法加密解密的主要流程。

如图 4-15 为加盐存储加密过程。当用户注册时，获取用户输入明文与随机产生的字符串进行拼接形成拼接字符串。拼接后的字符串最终通过 Hash 算法进行加密，由此密码加密完成。加密数据存入数据库。



如图 4-16 为用户密码解密过程。首先获取用户输入的明文，从数据库查询对应盐值。将用户密码明文与盐值进行拼接。最终经过哈希算法加密拼接的字符串生成密文，从数据库中查询对应用户密文和生成密文进行比较。如果相同则认证成功，反之认证失败。

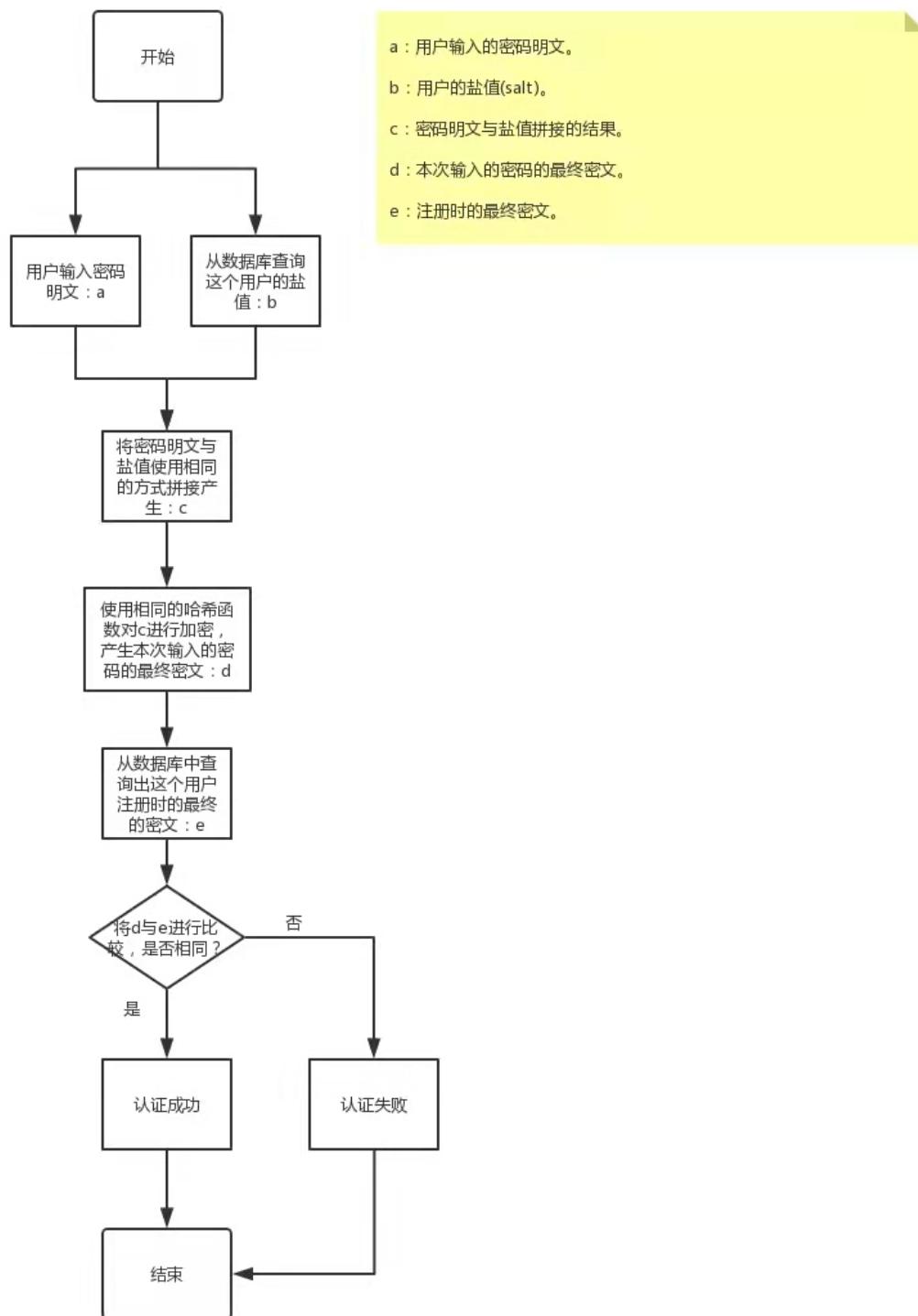


图 4-16 加盐解密流程

4.3 本章小结

本章主要介绍本系统的总体设计，包括概要设计和详细设计。在概要设计阶

段，首先进行了系统的总体设计介绍。后文介绍系统的设计流程，明确了每个模块的具体设计流程。在接口设计阶段，对于本系统的接口进行简要设计。在本章最后分析了本系统的数据结构。在系统详细设计中分析了数据库表关系，阐述了每个模块的实现流程。最后对于本系统加密存储使用到的算法做以简要的说明。

第五章 系统实现

5.1 开发环境搭建

在实现本系统的基本功能前，我们需要安装相应的 IDE。在服务端的搭建之前，需购买最低配置以上配置的服务器。通过较好的配置，才能实现系统的完整功能。如表 4-24 所示，展示的为本系统开发所需的软件工具和系统环境。本系统开发的硬件环境配置如表 4-25 所示。在接下来章节将详细阐述系统前后端开发环境的搭建。在此之前需要安装 JDK、JRE、Maven、MySQL、Studio 3T、Node.js 等基础开发环境。

5.1.2 后端环境搭建

在开发后端服务前，需先安装基础环境：JDK、JRE 等。在基础环境之上我们需要安装版本控制工具 Git v2.28.0、构建工具 Maven 3.6.1，服务启动工具 Tomcat v8.5.643 等在部署完上述工具包后，我们安装集成开发环境：IDEA，对应的开发环境版本号为 IntelliJ IDEA 2020.1 (Ultimate Edition)。在 IDEA 安装结束后，我们构建项目基础工程，选择 Maven 方式构建服务，逐步搭建系统所需微服务。微服务配置所需对应端口分配如表 4-26 所示，以此确保端口号不重复。

5.1.1 前端环境搭建

在前端环境搭建之前，需要完成以下工作：准备软件开发基础环境如表 4-24 所示。在安装 IDE 之前，需安装前端运行库：Node.js。软件版本为：Node.js v12.6.1。安装完成后，为进一步提高前端下载第三方库下载速度，需要使用 NPM 设置国内镜像源。在设置镜像源后运行 cnpm install 脚本命令安装前端所需第三方库库。

(一)、Web 端开发环境搭建

在准备工作结束后，可以从网络下载本系统如表 4-24 所示的本系统开发所需的前端开发环境。在管理端我们使用 WebStorm 进行开发，IDE 具体版本为 JetBrains WebStorm 2019.1x64。在安装结束后，为了更好地开发项目，我们需要构建基于 Vue.js 的脚手架工程。使用 vue create vue-hello 命令构建脚手架工程。构建结束后测试基础工程开始编程。在测试阶段使用 Google Chrome 浏览器进行基本测试。

(二)、移动端开发环境搭建

在系统移动端采用基于 Vue.js 的 uni-app 框架开发。移动端的界面绘制采用 uView 框架绘制。移动端采用 HBuilder X 工具进行开发，使用的 IDE 版本为：HBuilder X v3.2.16.20211122。在系统最终测试阶段采用两个平台进行测试：微信小程序端采用微信小程序开发工具开发，测试版本为 v1.05.2111300 Stable；在 Android 端采用 Android Studio 测试对应版本号为 Android Studio Arctic Fox | 2020.3.1 Patch 3。

5.1.3 云端服务环境搭建

服务端需要部署多个环境在此之前需要部署基本环境，服务器硬件要求如表 4-25 所示。

(一)、基础服务构建环境

操作系统环境：CentOS v8.2；

基础运行工具：JDK：1.8 及以上；

部署工具：Docker：v 20.10.7；

版本控制工具：Git v2.28.0。

数据库、持续集成工具、中间件等工具具体版本信息如表 4-24 所示。

(二)、前后端部署环境

管理端项目部署采用 NPM 打包方式部署进行。在打包结束后通过 FileZilla 上传对应打包文件。在服务端安装 Nginx，通过配置 Nginx.conf 文件请求后端接口。在服务器安装 Docker 容器，构建基础镜像。在 PC 客户端使用 Maven 工具进行打包，通过 Dockerfile 文件构建微服务镜像。将最终镜像通过 FileZilla 工具上传至服务器中，通过 docker 命令将镜像构建为容器。在启动容器时根据表 4-25 分配指定端口。在部署最后阶段启动 Nginx 服务，通过启动相应微服务容器，在客户端通过 postman（apifox）进行接口测试。至此，系统前后端测试环境部署完成。

5.2 模块实现

在基础环境和开发环境部署完成后，在下文中我们将对移动端与门户端、管理端、后台服务分别予以实现。通过展示各个平台功能的具体实现流程，以此展示本系统实现的主要功能。

在本部分主要包括代码解析部分和实现效果展示两部分。

5.2.1 移动端与门户端基本功能实现

(一)、物品捐赠功能实现

1、移动端与门户端功能实现

当用户点击开始捐赠后，首先判断用户是否登录。确认用户登录成功后进入信息输入页面。移动端与门户端判断用户信息合法后，用户确认捐赠信息并发布捐赠，在最后对捐赠服务进行评价。如图 5-1 为提交捐赠信息的代码示例。

```
submitdata() {
    this.$u.api.addDonateGoods(this.form).then(res => {
        res = res.data;
        if (res.code == '10000') {
            this.showToast(res.message, 'success');
            this.goods_id = res.donateGoods.id;
            this.change();
        } else {
            this.showToast(res.message, 'error');
        }
    })
}
```

图 5-1 移动端数据提交代码示例

2、服务端数据处理功能实现

当服务端接收到前端接口请求时，首先判断请求参数是否合法。在参数合法后，设置物品状态、存储时间，获取捐赠物品对应分类信息。当判断移动端传送分类信息无误后，保存捐赠物品信息，最后向前端返回捐赠成功信息。当捐赠成功时，前端请求接口更新用户积分，否则返回错误提示。示例代码如图 5-2 所示。

```
public DonateGoodsResult addDonateGoods(DonateGoods donateGoods) {  
    if (donateGoods == null){  
        return new DonateGoodsResult(CommonCode.FAIL,null);  
    }  
    donateGoods.setStatus("0");//状态未发货  
    donateGoods.setCreate_time(new Date());  
    String category_name = donateGoods.getGoods_category_id();  
    Optional<RecycleGoodsCategory> recycleGoodsCategoryOptional =  
        Optional.ofNullable(recycleGooodsCategoryRepository  
            .findRecycleGoodsCategoryByName(category_name));  
    if (recycleGoodsCategoryOptional.isPresent()) {  
        //总价  
        donateGoods.setGoods_category_id(recycleGoodsCategoryOptional.get().getId());  
        donateGooodsRepository.save(donateGoods);  
        return new DonateGoodsResult(CommonCode.SUCCESS,donateGoods);  
    }  
    //后台管理端无需查询分类id 直接保存  
    donateGoooodsRepository.save(donateGoods);  
    return new DonateGoodsResult(CommonCode.SUCCESS,donateGoods);  
}
```

图 5-2 服务端处理前端捐赠物品示例代码

3、效果图

移动端与门户端捐赠物品信息录入界面效果如图 5-3、5-4 所示。



图 5-3 移动端捐赠物品信息录入界面

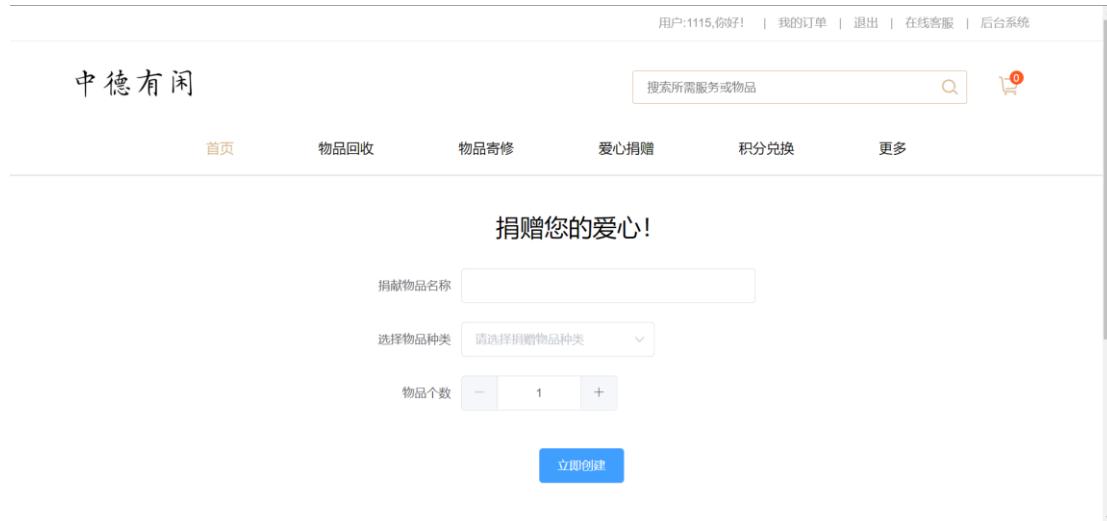


图 5-4 门户端捐赠物品信息录入界面

(二)、二手物品发布功能实现

1、移动端与门户端实现

在本部分，App 或门户端需要获取用户发布二手物品的基本信息。移动端通过采用 uView 框架采取组件化方式绘制页面，门户端采用 LayUI 绘制界面，通过文件上传组件上传文件。最后，通过 Vue.js 的数据双向绑定方式获取用户输入的物品信息，之后进行数据校验。在校验通过后，请求后端接口保存数据。在确认物品基本信息后，提示用户进行下一步操作录入物流信息。在完成上述步骤后，确认发布物品信息并确认发布。如图 5-5 为绘制发布页面部分代码示例。

```
<u-form-item label="物品重量" label-width="100px" prop="weight">
    <u-input v-model="form.weight" placeholder="请输入回收物品重量" type="number" />
</u-form-item>
<u-form-item label="原价格" label-width="100px" prop="single_price">
    <u-input v-model="form.old_price" placeholder="请输入原始价格" type="number" />
</u-form-item>
<u-form-item label="期望价格" label-width="100px" prop="single_price">
    <u-input v-model="form.single_price" placeholder="请输入期望价格" type="number" />
</u-form-item>
<u-form-item label="物品个数" label-width="100px" prop="count">
    <u-input v-model="form.count" placeholder="请输入回收物品个数(整数)" type="number" />
</u-form-item>
<u-form-item label="邮费" label-width="100px" prop="single_price">
    <u-input v-model="form.postage_price" placeholder="请输入邮费" type="number" />
</u-form-item>
```

图 5-5 发布页面绘制部分代码

2、后台服务数据保存实现

如图 5-6 所示，在接收到前端二手物品保存请求后，服务端首先判断对应参数是否合法。当移动端与门户端传递参数合法后，后台服务计算二手物品总价、

设置物品状态。在判断物品分类信息合法后，将二手物品信息存入数据库。最后将数据处理结果返回至前台。

```

public RecycleGoodsResult addRecycleGoods(RecycleGoods recycleGoods) {
    if (recycleGoods == null){
        return new RecycleGoodsResult(CommonCode.FAIL,null);
    }
    recycleGoods.setStatus(0); //状态未发货
    recycleGoods.setCreate_date(new Date());
    recycleGoods.setCount_price(recycleGoods.getSingle_price()*recycleGoods.getCount());
    //判断分类是否存在
    String category_name = recycleGoods.getCategory_id();
    Optional<RecycleGoodsCategory> recycleGoodsCategoryOptional =
        Optional.ofNullable(recycleGooodsCategoryRepository
            .findRecycleGoodsCategoryByName(category_name));
    if (recycleGoodsCategoryOptional.isPresent()){
        //总价
        recycleGoods.setCategory_id(recycleGoodsCategoryOptional.get().getId());
        RecycleGoods optionalRecycleGoods = recycleGooodsRepository.save(recycleGoods);
        return new RecycleGoodsResult(CommonCode.SUCCESS,optionalRecycleGoods);
    } else {
        //直接保存
        RecycleGoods optionalRecycleGoods = recycleGooodsRepository.save(recycleGoods);
        return new RecycleGoodsResult(CommonCode.SUCCESS,optionalRecycleGoods);
    }
}

```

图 5-6 添加二手物品后端示例代码

3、最终效果展示

如图 5-7、5-8 为移动端、门户端发布二手物品页面展示。



图 5-7 二手物品发布移动端效果

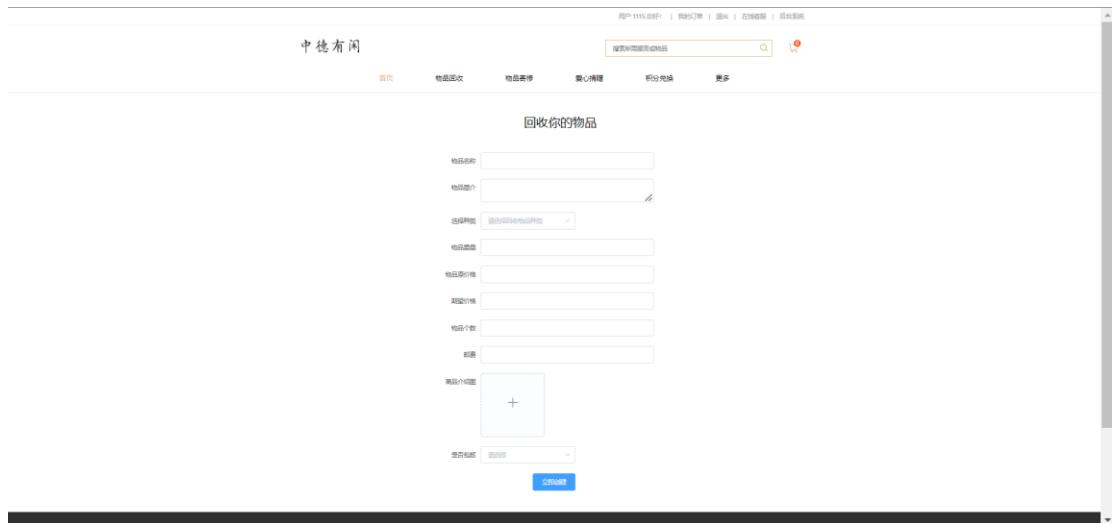


图 5-8 二手物品发布门户端效果

(三)、维修功能实现

1、移动端、门户端功能实现

在移动端与门户端，提示用户输入对应维修物品信息。在信息输入完成后，预约物品维修时间，选择个人寄货地址信息。在用户确认物品维修信息后，系统生成订单并返回需支付价格。用户支付完成后等待维修人员维修完成，在维修完成后评价服务。至此，移动端维修信息确认成功，请求后端保存维修信息。如图 5-9 为维修物品提交至服务端代码示例。

```

this.$u.api.addOrder(this.form).then(res => {
    res = res.data;
    // console.log(res)
    if (res.code == '10000') {
        this.showToast(res.message, 'success')

        this.order_id = res.orderList.id;
        // 获取id
        this.$u.route({
            url: '/pages/example/fixing_finished?id=' + this.order_id +
                '&goods_id=' +
                this
                .form.goods_id + '&fix_user_id=' + this.id.id
        })
    }

    // console.log(res)
    // this.$ref.resetFields()
    // 清除数据
    this.form.express_no = null;
    this.form.address = null;
    this.form.express_name = null;
    this.form.is_express = null;
    this.form.delivey_time = null;
    this.form.deliver_user_id = null;
} else {
    this.showToast(res.message, 'error')
}
})

```

图 5-9 移动端维修物品信息提交示例代码

2、后台服务实现

在前端发送保存信息请求后台服务接收后，首先判断参数是否合法，在参数合法后存入维修物品信息。最后返回前端存储数据是否成功。如图 5-10 为添加维修物品示例代码。

```

public FixGoodsResult addFixGoods(FixGoods fixGoods) {
    if (fixGoods == null){
        return new FixGoodsResult(CommonCode.FAIL,null);
    }
    fixGoods.setCreate_time(new Date());
    // 总价
    fixGoodsRepository.save(fixGoods);
    return new FixGoodsResult(CommonCode.SUCCESS,fixGoods);
}

```

图 5-10 添加维修物品信息示例代码

3、移动端与门户端维修页面效果图

如图 5-11、5-12 为移动端、门户端维修页面效果图。



图 5-11 移动端维修页面效果图

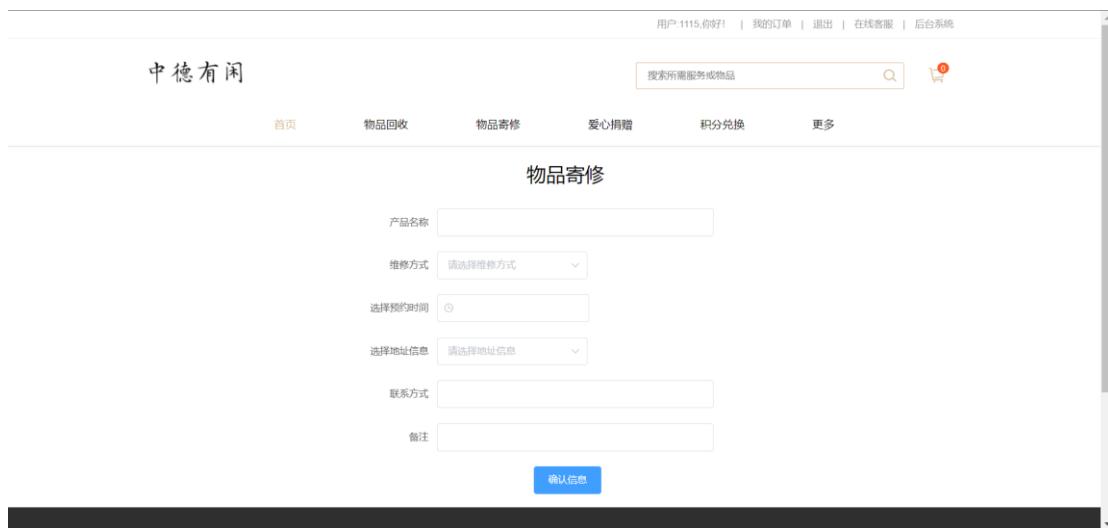


图 5-12 门户端维修页面效果图

(四)、积分兑换物品功能实现

1、在用户捐赠完成后，可获得对应积分。在获取足够积分后，用户可前往 App 与门户端积分商城版块凭积分兑换相应的积分物品。在用户选择欲兑换商品

后，移动端判断用户积分剩余是否满足要求。积分满足后，移动端或门户端请求服务端更新积分物品数量信息，更新用户积分信息。如图 5-13 为更新积分兑换物品数量信息的示例代码。

```
this.form.count = this.current_count;
this.form.count = this.form.count - item.grade_count;
| console.log(this.form.count < item.grade_count)
if (this.form.count < item.grade_count) {
    this.showToast('您的积分不足', 'error')
} else {
    //调用接口
    this.$u.put('/api/points/updatepoints/' + this.id.id, this.form).then(res => {
        res = res.data;
        if (res.code == '10000') {
            // console.log(res);
            this.updategoods(item);
        } else {
            this.showToast('兑换失败，物品已兑换完！', 'error')
        }
    })
}
```

图 5-13 积分兑换物品示例代码

2、后端处理实现流程

当用户兑换完成后，移动端与门户端请求服务端更新用户积分信息。服务端首先更新用户积分信息，之后判断用户积分与等级对应关系。在积分处于等级最大最小值中间时，确定新的积分等级并存入数据库。服务端将更新的用户积分返回至前台。示例代码如图 5-14 所示。

```

public ResponseResult updatePoints(String id, Points points) {
    if (id == null && points == null){
        return new ResponseResult(CommonCode.INVALID_PARAM);
    }
    Optional<Points> expressOptional = pointsRepository.findById(id);
    if (expressOptional.isPresent()){
        Points points1 = expressOptional.get();
        points1.setUpdate_time(new Date());
        points1.setCount(points.getCount());
        points1.setGrade(points.getGrade());
        points1.setName(points.getName());
        points1.setUserid(points.getUserid());
        points1.setStatus(points.getStatus());
        pointsRepository.save(points1);
        return new ResponseResult(CommonCode.SUCCESS);
    }else if(points.getUserid() != null){
        //更新用户积分信息
        String user_id = points.getUserid();
        Points points1 = pointsRepository.findByUserid(user_id);
        if (points1 != null){
            //查询积分信息不为空
            points1.setUpdate_time(new Date());
            //获取当前积分信息
            int count = points.getCount();
            //更新等级信息
            //查询当前等级最小值信息
            int grade = points1.getGrade();
            Optional<Grade> grade1 = gradeRepository.findByGrade(grade);
            if(count < grade1.get().getMin_value()){
                //当前积分值值小于当前积分等级最小值
                //等级减1
                do {
                    grade = grade - 1;
                    grade1= gradeRepository.findByGrade(grade);
                    //使用更新的等级继续查询
                }while (count < grade1.get().getMin_value());
            }
            //保存更新的等级 积分信息
            points1.setCount(count);
            points1.setGrade(grade);
            pointsRepository.save(points1);
            //返回更新成功信息
            return new ResponseResult(CommonCode.SUCCESS);
        }
    }
    return new ResponseResult(CommonCode.INVALID_PARAM);
}

```

图 5-14 服务端积分更新示例代码

3、移动端与门户端兑换效果

如图 5-15、5-16 为移动端、门户端兑换效果图。



图 5-15 移动端积分兑换物品效果图



图 5-16 门户端积分兑换物品效果图

(五)、购买二手物品功能实现

1、移动端与门户端实现

当用户点击立即购买按钮并选择收货信息后，移动端与门户端发起订单生成请求至服务器，服务器处理请求后返回结果至前端。用户付款结束后，移动端同步数据至服务器同时更新订单状态。如图 5-17 为购买二手物品实现示例代码。

```
this.$u.api.getRecycleGoods(this.id).then(res => {
    this.goods = res.data.recycleGoodsList;
    this.goods.goodsPic.src = imgUrl.imgUrl + this.goods.goodsPic.src
    for (i = 0; i < this.list.length; i++) {
        this.list[i].image = this.goods.goodsPic.src;
        // console.log(this.list.image)
    }
    // console.log(this.goods)
})
```

图 5-17 购买二手物品示例代码

2、服务端实现

在服务端接收到前台订单生成请求后，生成对应订单信息并设置订单状态保存到数据库。保存成功后，返回处理结果。如图 5-18 为服务端生成订单示例代码。

```
public OrderListResult addOrder(OrderList orderList) {
    if (orderList == null) {
        return new OrderListResult(CommonCode.FAIL,null);
    }
    //保存物流对应的订单号
    orderList.setCreate_time(new Date());

    if(orderList.getOrder_status() == null || orderList.getOrder_settlement_status() == null) {
        orderList.setOrder_status("10005");//已发布物品
        orderList.setOrder_settlement_status("10005");
    }
    OrderList orderList1 = orderListRepository.save(orderList);
    String goods_id = orderList1.getGoods_id();
    Express express = expressRepository.findByGoodsid(goods_id);
    express.setOrder_id(orderList1.getId());
    expressRepository.save(express);
    return new OrderListResult(CommonCode.SUCCESS,orderList1);
}
```

图 5-18 服务端生成订单示例代码

3、实现效果图

如图 5-19、5-20 为移动端、门户端购买物品界面效果图。

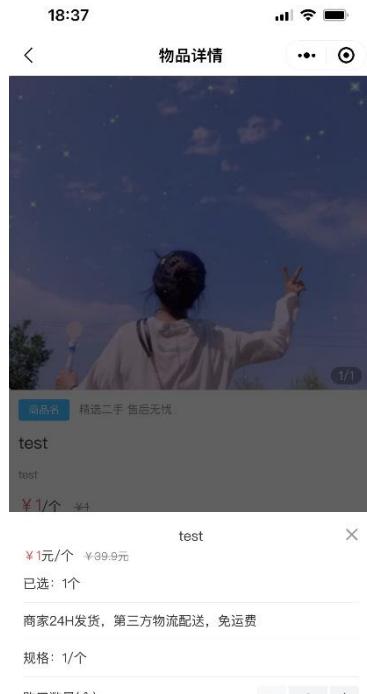


图 5-19 移动端购买二手物品效果图



图 5-20 门户端购买二手物品效果图

5.2.2 管理端基本功能实现

(一)、用户数据管理功能实现

1、管理端实现

在管理端数据管理是使用较多功能之一。在用户点击管理数据按钮后，管理端验证数据合法性，在数据合法后发送请求至服务后台。如图 5-21 为发送请求

示例代码。

```
saveAdd() {
    this.addform.goods_category_id = this.addform.goods_category_id[0];
    this.$api.userList.addFixGoods(this.addform).then(res => {
        if (res.code !== '10000') {
            this.$Message.error("错误！")
        }

        this.$Message.success('保存成功');
        this.delVisible = false;
        this.getData()
    });
    // console.log(this.addform)
    this.addVisible = false;
}
```

图 5-21 发送请求示例代码

2、服务端处理请求实现

服务端接收后端请求后，提取参数并检验是否合法。controller 类接收参数后调用相应服务对应方法进行数据处理。如图 5-22 为 controller 调用示例代码。

```
@Override
@GetMapping("/getorder")
public OrderListResult getOrder(@RequestParam("id") String id) {
    return orderListService.getOrder(id);
}

@Override
@PostMapping("/addorder")
public OrderListResult addOrder(OrderList orderList) {
    return orderListService.addOrder(orderList);
}

@Override
@DeleteMapping("/delorder")
public ResponseResult deleteOrder(@RequestParam("id") String id) {
    return orderListService.deleteOrder(id);
}

@Override
@PutMapping("/updateorder/{id}")
public ResponseResult updateOrder(@PathVariable("id") String id, OrderList orderList) {
    return orderListService.updateOrder(id,orderList);
}

@Override
@GetMapping("/getorder/list/{page}/{size}")
public QueryResponseResult findOrderList(@PathVariable("page") int page,
                                         @PathVariable("size") int size,
                                         QueryOrderListRequest queryOrderListRequest) {
    return orderService.findOrder(page,size,queryOrderListRequest);
}
```

图 5-22 前端接口处理

3.管理端效果展示

如图 5-23 所示为数据管理效果图。

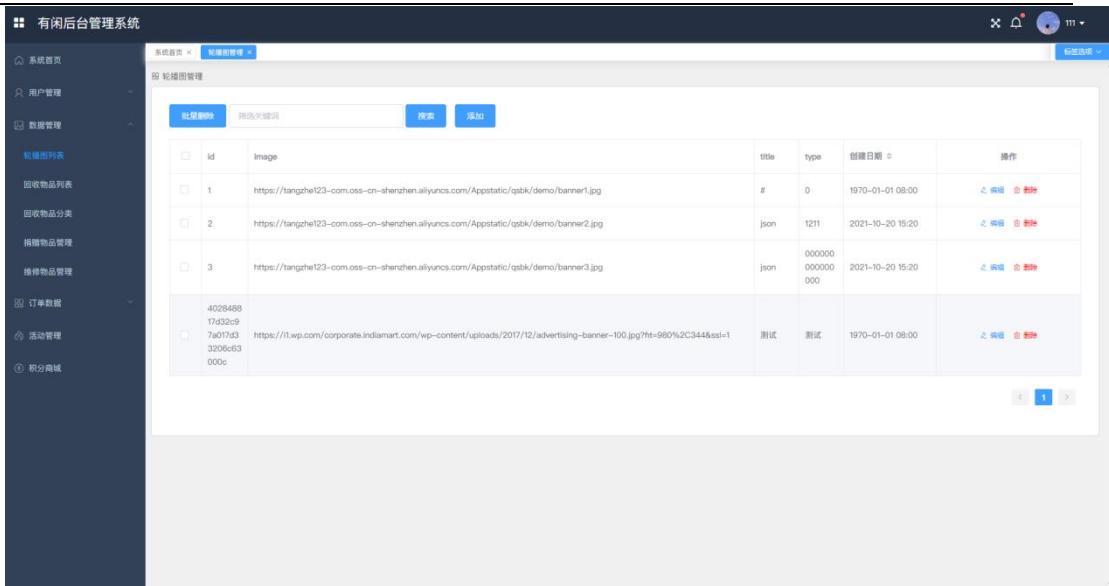


图 5-23 数据管理效果

(二)、活动发布功能实现

1、管理端实现

当用户点击输入活动信息，选择所发起活动起始时间。上传活动介绍图后点击添加按钮，前端校验通过后请求后端服务。如图 5-24 所示为管理端活动发布实现示例代码。

```
saveAdd() {
    this.addform.submit_id = this.user.username;
    this.addform.time = this.addform.time[0] + '_' + this.addform.time[1];
    // console.log(this.addform)
    this.$api.userList.addActivity(this.addform).then(res => {
        if (res.code !== '10000') {
            this.$Message.error("错误!")
        }

        this.$Message.success('保存成功');
        this.addform.activity_name = null;
        this.addform.content = null;
        this.addform.time = null;
        this.addform.remark = null;
        this.addform.status = null;
        this.delVisible = false;
        this.getData()
    });
    // console.log(this.addform)
    this.addVisible = false;
}
```

图 5-24 管理端发送保存数据请求示例代码

2、服务端请求处理实现

服务端接收前端请求后，检查参数是否合法。在参数合法后，请求数库

保存数据，示例代码 5-25 所示。

```
/*
 * description: 添加活动
 * @param activity description
 * @return
 * @throws
 * @author czq
 * @date ${date} ${time}
 */
public ActivityResult addActivity(Activity activity) {
    if (activity == null){
        return new ActivityResult(CommonCode.FAIL,null);
    }
    activity.setCreate_time(new Date());
    //总价
    activityRepository.save(activity);
    return new ActivityResult(CommonCode.SUCCESS,activity);
}
```

图 5-25 添加活动服务端实现示例代码

3、管理端效果展示

如图 5-26 为管理端添加活动页面效果图。

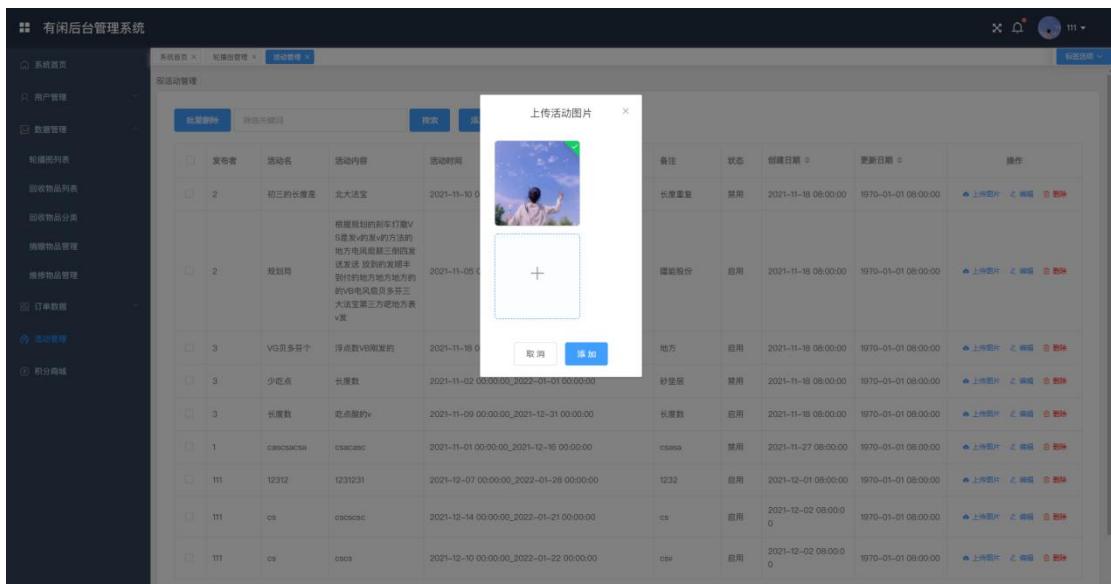


图 5-26 管理端添加页面

5.3 关键功能代码分析

在上述 5.2 部分分别对本系统 App 端、管理端的实现过程进行了详细的论

述。在阐述基本实现代码后，展示了实现效果图。在本部分将对系统重点功能进行介绍。

5.3.1 登录流程实现代码分析

(一)、前端请求实现

1、当用户输入账户信息后，前端校验账户信息是否合法并发送校验结果和数据至前端。如图 5-27 为请求示例代码。前端请求成功后获取 token 令牌后存储至浏览器 cookie 中。

```
let para = Object.assign({}, this.ruleForm);
// console.log(para)
loginApi.login(para).then(res => {
    // console.log(res)
    if (res.success) {
        this.$router.push('/');
        this.$Message.success("成功！");
        // console.log(res)
        if (this.returnUrl !== 'undefined' && this.returnUrl !== '') {
            //跳转到首页
            window.location.href = this.returnUrl;
        } else {
            window.location.href = 'http://www.zdyouxian.com/'
        }
    } else {
        this.$Message.error("错误！")
    }
})
```

图 5-27 前端请求后端获取 token

2、如图 5-28 为主程序获取 token 并判断是否存在，存在之后携带 cookie 中 token 信息请求服务端获取 jwt 令牌。在 jwt 令牌接收后，解析用户信息并提示用户登录是否成功。

```

} else if (uid) { //已登录 获取jwt令牌
    //请求获取jwt
    systemApi.getjwt().then((res) => {
        if (res.success) {
            let jwt = res.jwt;

            utilApi.jwtparse(jwt);
            let activeUser = utilApi.getActiveUser();
            // console.log(activeUser);
            if (activeUser) {
                utilApi.setUserSession("activeUser", JSON.stringify(activeUser))
                // localStorage.setItem('ms_username', this.ruleForm.phone);
                // localStorage.setItem('userpic', res.userMessage.userpic);
                // localStorage.setItem('user_id', res.userMessage.id);

            }
            next();
        } else { //不成功重新登陆
            //跳转到统一登陆
            // console.log("window.location")
            window.location = "http://www.zdyouxian.com/#/login";//?returnUrl=" + i
            next();
        }
    })
}

```

图 5-28 前端获取后端 jwt 令牌

(二)、服务端请求实现

1、如图 5-29 微服务端处理前端获取 token 信息的示例代码。当服务端接收前端请求后，提取用户账户信息判断信息是否合法。在信息合法后，请求服务获取 token。当用户信息正确时，token 获取成功。在获取成功 token 后服务端将 token 信息保存在 Redis 中。在判断 token 时间已过期后，返回允许用户登录信息。

```

public AuthToken login(String username, String password, String clientId, String clientSecret)
    申请令牌
    AuthToken authToken = applyToken(username,password,clientId,clientSecret);
    if (authToken == null ){
        ExceptionCast.cast(AuthCode.AUTH_LOGIN_APPLYTOKEN_FAIL);
    }
    save token to redis
    String access_token = authToken.getAccess_token();
    String content = (String) JSON.toJSONString(authToken);
    boolean saveTokenResult = saveToken(access_token,content,tokenValiditySeconds);
    if (!saveTokenResult){
        ExceptionCast.cast(AuthCode.AUTH_LOGIN_TOKEN_SAVEFAIL);
    }
    return authToken;
}

```

图 5-29 前端获取 jwt 令牌

2、在前端携带 cookie 请求后端服务获取令牌，服务端接收请求获取 cookie 中 token 信息并查询 Redis 获取 jwt 令牌。当服务端成功获取 jwt 令牌后，返回至前端解析。如图 5-30 为示例代码。

```

    /
    @Override
    @GetMapping("/userjwt")
    public JwtResult userjwt() {
        // 获取cookie中的令牌
        String access_token = getTokenFromCookie();
        // 从redis中查询jwt
        AuthToken authToken = authService.getUserToken(access_token);
        if (authToken == null){
            return new JwtResult(CommonCode.FAIL,null);
        }
        return new JwtResult(CommonCode.SUCCESS,authToken.getJwt_token());
    }
}

```

图 5-30 获取 jwt 令牌服务端示例代码

5.3.2 文件上传流程实现代码分析

(一)、前端请求实现

当用户选择文件点击上传后，前端携带文件参数和文件内容请求后台文件上传 API，如图 5-31 所示。

```

<el-upload
    :before-remove="handleRemove"
    :before-upload="setupUploadData"
    :data="uploadVal"
    :file-list="fileList"
    :limit="picMax"
    :on-exceed="rejectUpload"
    :on-success="handleSuccess"
    action="/api/filesystem/upload"
    list-type="picture-card"
    name="multipartFile">
    <i class="el-icon-plus"></i>
</el-upload>

```

图 5-31 前端文件上传示例代码

(二)、服务端请求实现

如图 5-32 为服务端文件上传接口处理示例代码。在接收到前端文件上传请求后解析文件数据。当接口参数合法后，通过服务端 FastDFS 插件上传文件至 FastDFS 云端服务。在上传结束后，云端返回文件 id。在服务器确认 id 合法后，存储文件数据至 MongoDB 数据库。在此之外，如果前端传递 meta 信息则解析对应数据并保存。保存所有数据至数据库后，服务器返回文件信息至前端提示用户是否上传成功。

```

public UploadFileResult uploadFileResult (MultipartFile multipartFile,
String fileTag, String businessKey, String meta){
    if (multipartFile == null) {
        ExceptionCast.cast(FileSystemCode.FS_UPLOADFILE_FILEISNULL);
    } // 上传的文件 得到id
    String fid = fdfs_upload(multipartFile);
    if (StringUtils.isEmpty(fid)) {
        ExceptionCast.cast(FileSystemCode.FS_UPLOADFILE_FILEISNULL);
    } // 信息存储进入MongoDB
    FileSystem fileSystem = new FileSystem();
    fileSystem.setFileId(fid);
    fileSystem.setFilePath(fid);
    fileSystem.setFiletag(fileTag);
    fileSystem.setBusinesskey(businessKey);
    fileSystem.setFileName(multipartFile.getOriginalFilename());
    fileSystem.setFileType(multipartFile.getContentType());
    if (StringUtils.isEmpty(meta)){
        try {
            Map map = JSON.parseObject(meta, Map.class);
            fileSystem.setMetadata(map);
        }catch (Exception e){
            e.printStackTrace();
        }
    }
    fileSystemRepository.save(fileSystem);
    return new UploadFileResult(CommonCode.SUCCESS,fileSystem);
}

```

图 5-32 文件上传请求接口服务端处理示例代码

5.4 本章小结

本章主要介绍了本系统软件环境的搭建：包括开发环境的搭建、基本部署环境搭建。在本章第二部分介绍了本系统关键功能的实现，以及部分代码的解析。在最后简要介绍了本系统比较重要功能的实现方法并对实现代码作以简要介绍。

第六章 系统测试

在系统测试阶段，我们需要逐步对系统进行科学、合理的测试。测试方法采用软件工程中的白盒测试和黑盒测试。通过测试用例进行逐项严格测试，使得我们在系统上线前以最小代价发现潜藏的问题；测试应具有严格的过程，严格按照测试计划执行^[16]。在测试用例执行前合理的制定测试计划。在测试结束后，需要对测试结果进行记录并编写测试报告。

6.1 单元测试

在本系统中单元测试模块采取以平台为界进行分块，主要分为移动端、管理端、服务端测试。

6.1.1 移动端与门户端测试

(一)、测试范围与目的

移动端与门户端测试范围如表 6-1 所示。移动端与门户端测试目的在于：通过严格执行测试计划，检验系统是否达到预期效果；以最小的代价，发现潜藏于系统的问题，使得系统可靠性进一步增加。

表 6-1 移动端与门户端测试范围

项目	细分功能	测试项目
主页	轮播图	查询
	活动详情	查询
	活动	查询
	商品列表	查询
	商品详情	查询
	评论	查询
		增加
	商品购买	交易
	捐赠物品浏览	查询
捐赠	信息填写	增加
	发布捐赠	增加
	评价服务	增加
回收	信息确认	增加
	开始寄件	增加

	确认回收	增加
	文件上传	增加
维修	信息输入	增加
	开始维修	增加
	进度查询	增加
	在线评价	增加
	密码登录	登录
注册	用户注册	注册
退出	登出	账户状态退出
我的	个人信息	查询
	积分商城	信息查询
		兑换物品
		信息确认
	订单	基础信息查询
		详细查询
		更新订单信息
	地址管理	查询
		增加
		修改
		删除
	捐献管理	查询
	维修商品	列表查询
		评价
	购物车	添加与删除物品
搜索		查询商品信息

(二)、测试环境及工具

移动端与门户端测试环境采取表 4-24 中软件开发环境，测试工具采用 Google Chrome 浏览器。

(三)、测试结果

根据表 6-1 测试用例，使用测试工具在测试环境进行严格测试。测试结果如表 6-2 所示。

表 6-2 移动端测试用例

项目	细分功能	测试项目	端	测试次数	成功次数	端口号	存在问题	备注	所属平台

主页	轮播图	查询	移动、门户	5	5	8800、8080			h5、web
	活动详情	查询	移动、门户	5	5	8800、8080			h5、web
	活动	查询	移动、门户	5	5	8800、8080			h5、web
	商品列表	查询	移动、门户	5	5	8800、8080			h5、web
	商品详情	查询	移动、门户	5	5	8800、8080			h5、web
	评论	查询	移动、门户	5	5	8800、8080			h5、web
		增加	移动、门户	5	5	8800、8080			h5、web
	商品购买	交易	移动、门户	5		8800、8080		未完善	h5、web
	捐赠物品浏览	查询	门户	5	5	8080			Web
捐赠	信息填写	增加	移动、门户	5	5	8800、8080			h5、web
	发布捐赠	增加	移动、门户	5	5	8800、8080			h5、web
	评价服务	增加	移动、门户	5	5	8800、8080			h5、web
回收	信息确认	增加	移动、门户	5	5	8800、8080			h5、web
	开始寄件	增加	移动、门户	5	5	8800、8080			h5、web
	确认回收	增加	移动、门户	5	5	8800、8080			h5、web
	文件上传	增加	移动、门户	5	5	8800、8080			h5、web
维修	信息输入	增加	移动、门户	5	5	8800、8080			h5、web

	开始维修	增加	移动、门户	5	5	8800、8080			h5、web
	进度查询	增加	移动、门户	5	5	8800、8080			h5、web
	在线评价	增加	移动、门户	5	5	8800、8080			h5、web
登	密码登录		移动、门户	5	5	8800、8080			h5、web
退	退出		移动、门户	5	5	8800			h5、web
我的	个人信息	查询	移动、门户	5	5	8800、8080			h5、web
	积分商城	信息查询	移动、门户	5	5	8800、8080			h5、web
		兑换物品	移动、门户	5	5	8800、8080			h5、web
		信息确认	移动、门户	5	5	8800、8080			h5、web
	订单	基础信息查询	移动、门户	5	5	8800、8080			h5、web
		详细查询	移动、门户	5	5	8800、8080	信息显示错误	数据异常	h5、web
		更新订单信息	移动、门户	5	0	8800、8080	接口错误	数据异常	h5、web
	地址管理	查询	移动、门户	5	5	8800、8080			h5、web
		增加	移动、门户	5	5	8800、8080			h5、web
		修改	移动、门户	5	5	8800、8080			h5、web
		删除	移动、门户	5	5	8800、8080			h5、web
	捐献管理	查询	移动、门户	5	5	8800、8080			h5、web

	维修商品	列表查询	移动、门户	5	5	8800、8080			h5、web
		评价	移动、门户	5	5	8800、8080			h5、web
购物车	购物车	添加、删除	门户	5	5	8080			web
搜索			移动、门户	5	5	8800、8080			h5、web

6.1.2 管理端测试

(一)、测试范围与目的

管理测试范围如表 6-3 所示。管理端测试目的在于：通过合理的设计测试用例，严格执行测试计划，发现潜藏于管理端的问题；争取以最小的代价发现潜在的问题，增强系统的可靠性^[17]。

表 6-3 管理端测试范围

功能	细分功能	测试项目
dashboard	用户与角色管理	查询
		增
		改
		删
		查询
	物流人员管理	增
		改
		删
		查询
		增
用户管理	维修人员管理	改
		删
		查询
		增
		改
	用户地址管理	删
		查询
		增
		改
		删
		查询

系统管理	用户积分管理	增
		改
		删
		查询
	积分等级管理	增
		改
		删
		查询
	评论管理	增
		改
		删
		查询
	角色与角色权限管理	增
		删
		改
		查
	菜单管理	增
		删
		改
		查
数据管理	轮播图列表	增
		改
		删
		查询
	回收物品分类	增
		改
		删
		查询
	捐赠物品管理	增
		改
		删
		查询
	维修物品管理	增
		改

		删
		查询
		增
		改
		删
		查询
		增
		改
		删
		查询
		增
		改
		删
		查询
订单管理	订单管理	
活动管理	活动管理	
积分商城	积分商城	
登录		用户登录
退出		用户退出

(二)、测试环境及工具

管理端测试环境如表 4-24 所示，测试工具采用 Google Chrome 浏览器。

(三)、测试结果

根据表 6-3 严格执行测试计划，测试结果如表 6-4 所示。

表 6-4 管理端测试结果

功能	细分功能	测试项目	端	测试次数	成功次数	端口号	存在问题	备注
dashbo ard	数据查询	查询	管 理	5	5	8080		
用户管 理	用户列表	增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
	物流人员 管理	增	管 理	5	5	8080		

	维修人员 管理	改	管 理	5	5	8080		
		删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
		增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
	用户地址 管理	增	管 理	5	2	8080	参数非法	间接性问题
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
	用户积分 管理	增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
	积分等级 管理	增	管 理	5	5	8080		
		改	管 理	5	5	8080		

	评论管理	删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
		增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
	系统管 理	查询	管 理	5	5	8080		
		增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
	菜单管理	查询	管 理	5	5	8080		
		增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
	数据管 理	查询	管 理	5	5	8080		
		增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		

		查询	管理	5	5	8080		
回收物品 分类		增	管理	5	5	8080		
		改	管理	5	5	8080		
		删	管理	5	5	8080		
		查询	管理	5	5	8080		
捐赠物品 管理		增	管理	5	5	8080		
		改	管理	5	5	8080		
		删	管理	5	5	8080		
		查询	管理	5	2	8080	分类名显 示问题	
维修物品 管理		增	管理	5	5	8080		
		改	管理	5	5	8080		
		删	管理	5	5	8080		
		查询	管理	5	5	8080		
订单管 理	订单管理	增	管理	5	5	8080		
		改	管理	5	5	8080		
		删	管理	5	5	8080		
		查询	管理	5	5	8080		

活动管理	活动管理	增	管 理	5	5	8080		
		改	管 理	5	5	8080		
		删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
积分商城	积分商城	增	管 理	5	5	8080		
		改	管 理	5	5	8080		编辑分类名无法传递
		删	管 理	5	5	8080		
		查询	管 理	5	5	8080		
登录			管 理	5	5	8080		
退出			管 理	5	5	8080		

6.1.3 服务端测试

服务端测试主要集中于数据操作、用户鉴权等功能测试。服务端测试旨在验证服务端功能符合预期，增强服务端功能的可靠性。

(一)、测试范围与目的

服务端测试范围为编写的所有测试接口。测试接口以检验接口功能的正确性。如图 6-1 为测试部分接口举例。

activity-controller : 活动管理接口		Show/Hide List Operations Expand Operations
POST	/manage/activity/addactivity	添加维修物品信息
DELETE	/manage/activity/delactivity	删除活动信息
GET	/manage/activity/getactivity/list/{page}/{size}	分页查询活动信息
GET	/manage/activity/getactivitybyapp/list/{page}/{size}	分页查询app活动信息
GET	/manage/activity/getactivitybyid	单个活动详情查询
PUT	/manage/activity/updateactivity/{id}	更新活动信息
admin-controller : 后台管理数据查询接口		Show/Hide List Operations Expand Operations
dict-controller : 字典管理接口		Show/Hide List Operations Expand Operations
donate-goods-controller : 捐赠管理接口		Show/Hide List Operations Expand Operations
goods-pic-controller : 图片管理接口		Show/Hide List Operations Expand Operations
manage-controller : 管理接口		Show/Hide List Operations Expand Operations

图 6-1 部分接口展示

(二)、测试环境及工具

测试环境如表 4-24 服务端开发环境所示。测试工具采用 swagger-ui.html 和 postman 结合测试。在使用 postman 测试前，需先启动 Nginx。具体版本要求详见表 4-24。

(三)、测试结果

如图 6-2 为 swagger-ui.html 测试结果。

GET /manage/activity/getactivity/list/{page}/{size} 分页查询活动信息

Response Class (Status 200)
OK

Model Example Value

```
{
  "code": 0,
  "message": "string",
  "queryResult": {
    "list": [
      {}
    ],
    "total": 0
  },
  "success": true
}
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
page	1	page	path	integer
size	10	size	path	integer
activity_name		query		string
status		query		string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:41000/manage/activity/getactivity/list/1/10'
```

Request URL

```
http://localhost:41000/manage/activity/getactivity/list/1/10
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

```
{
  "success": true,
  "code": 10000,
  "message": "操作成功!",
  "queryResult": {
    "list": [
      {
        "id": "402848817d32c97a017d32ee3a710008",
        "submit_id": "2",
        "activity_name": "初三的长度是",
        "time": "2021-11-10 00:00:00_2021-12-09 00:00:00",
        "content": "北大法宝",
        "status": "1",
        "remark": "长度重复",
        "update_time": null,
        "create_time": "2021-11-18T00:00:00.000+0000"
      },
      {
        "id": "402848817d32c97a017d32eecc6a10009"
      }
    ],
    "total": 2
  }
}
```

Response Code

```
200
```

Response Headers

```
{
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 08 Dec 2021 01:41:41 GMT",
  "transfer-encoding": "chunked"
}
```

图 6-2 swagger-ui.html 测试结果

如图 6-3 为 postman 测试结果。

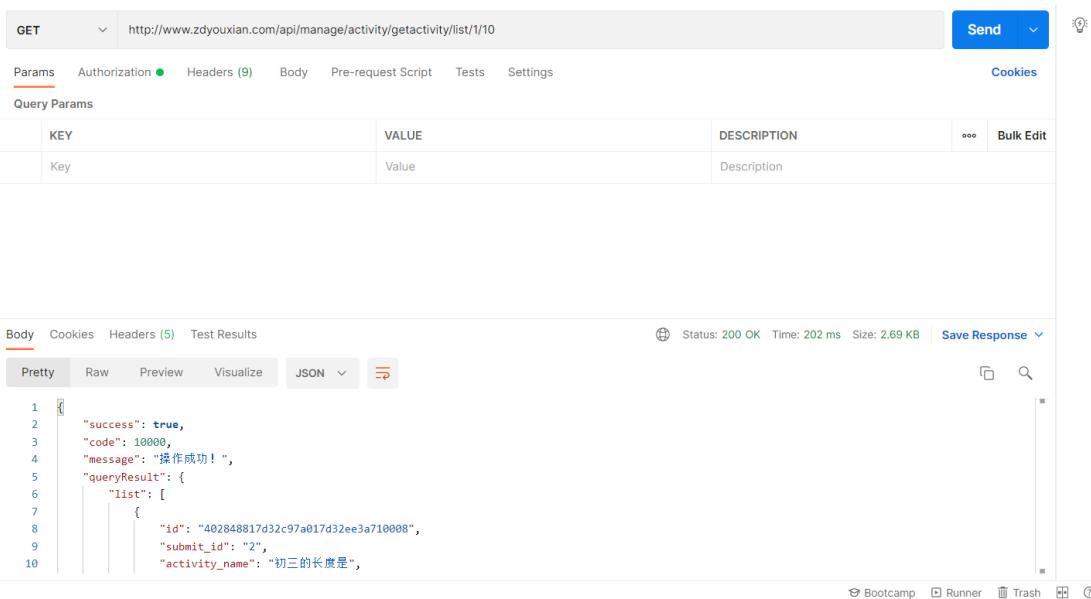


图 6-3 postman 测试结果

6.2 集成测试

在系统集成测试阶段，我们需要对整个系统的整体功能进行全面测试。通过前台与后台的集成测试接口发现潜藏于系统之中的 bug。

(一)、测试范围与目的

本系统集成测试范围为后端所有接口与前端功能的集成测试。由于接口较多，故只列出测试中有问题的项目。

(二)、测试环境与工具

本系统测试前端环境与后端环境参考表 4-24 所示安装，测试工具采取手工方式。

(三)、测试结果

测试结果如表 6-5 所示。

表 6-5 测试结果表

微服务	测试项目	端	测试次数	成功次数	端口号	存在问题	备注
rs_center_service	所有接口		10	10	50101		
rs_gateway_service	所有接口		10	10	50201		
rs_search_service	所有接口		10	10	46000		

rs_auth_service	所有接口	管理	5	3	45000	退出后页面跳转错误	
rs_user_service	所有接口		10	10	42000		
rs_manage_service	所有接口	管理	5	3	41000	分类显示错误	
	所有接口	管理	5	0		添加错误	
rs_points_service	所有接口		10	10	44000		
order	所有接口		10	10	43000		
filesystem	文件上传	管理	3	0	22100	上传文件为空	
	文件上传	管理	3	0		接口错误	
其他			10	8	8080	setUserSession	
移动端	所有	移动	10	10	8080		
门户端	所有	门户	10	10	8080		
dashboard	所有	管理	5	4	8800	无限弹窗	

6.3 测试结论

经过上述测试后，我们可以从以下几个方面总结测试：

(一)、功能性

通过表 6-1、表 6-3、系统集成测试结果我们可以发现：

- 1、系统的基本功能已正常运行；
- 2、系统功能具备可扩展性；
- 3、系统经过测试已具备上线条件。

(二)、易用性

在系统前端设计中，我们必须考虑系统的易用性。经过上述测试我们发现：

- 1、移动端基于 Vue.js 采用 uView UI 框架绘制界面。经过测试：具有美观、简约、便于操作、符合直觉等特点；
- 2、管理端采用基于 Vue.js 的 Element UI 框架绘制界面。经过上述测试，可以发现管理端具有易用、简约等特点。
- 3、门户端采用基于 Vue.js 的 Element UI 框架和 LayUI 绘制界面，使用 Nuxt.js 框以利于 SEO，经过上述测试，可以发现门户网站具有稳定、简单易用等特点。

（三）、可靠与安全性

安全性：在本系统数据存储中采用云端加盐加密存储，在用户账户信息的加密存储中采取加盐存储方式存储，具有一定安全性；

可靠性：本系统后台基于微服务架构采用分布式体系部署微服务。在出现设备宕机时，可以启用备份机，迅速恢复系统功能。通过上述方式，此系统具备一定可靠性。

（四）、移动端兼容性

本系统移动端采用混合开发框架 uni-app，使用 uView UI 框架绘制界面。可一次开发兼容多个平台，具备多个系统多个设备的兼容性。

6.4 本章总结

在本章系统测试中，我们对系统的整体功能进行了流程式的分析。采取了白盒测试、黑盒测试方法，通过单元测试、集成测试步骤进行了细致的测试^[18]。在检测出问题后，修改 bug 后对系统进行回归测试确保功能正常运行。在最后，对测试结果做以总结。

第七章 总结与展望

在当下，二手物品市场交易主要以线上为主。一方面，由于闲鱼和转转已经在二手市场占据了大部分的市场份额^[19]，故新生交易平台生存存在困境。另一方面，线上交易虽然省力，但是时效性和保障性较差。快递时效平均为 3 天，限制了用户的紧急需求。与此同时线上商品展示与实物相差较多，容易产生交易纠纷，同时售后响应慢。

因此在人流量较大的区域如学校、小区等，亟需一种新型的二手交易平台以克服单纯线上交易的缺点。

本项目紧抓线上交易的痛点，在人流量大的区域采用线上+线下的方式进行闲置物品的回收再利用，同时设立维修预约平台、物品捐赠平台等。基于学校人流量大致固定的特点，可以长期有效的形成物品循环链条，保证物品流通量。因此，实现此系统具备一定必要性。

在本文中通过对基于微服务架构的校园线上二手回收与寄修系统设计与实现的详细介绍，介绍了开发二手回收系统的基本流程、介绍了相关技术，通过解析实现功能的代码进一步让读者了解本系统的实现过程与方法。最终通过测试验证系统功能的稳健性，使得读者对于本系统开发了解更加深入。

在系统上线后，系统运维正式展开。本系统具备一定可扩展性，可在已有功能基础上开发新的功能。在后续运维中，我们可以持续的、集成性的扩展、运维此系统。

参考文献

- [1] 王轶多,王培宇.大学生闲置物品研究报告及建立转卖平台的思考[J].智库时代,2020(15):255-256.
- [2] 陆飞.统筹城乡医疗保险制度研究[D].南京工业大学,2014.
- [3] 徐冉.基于微服务架构的秒杀系统服务端设计与实现[D].东南大学,2020.DOI:10.27014/d.cnki.gdnau.2020.002753.
- [4] 王振宇.基于大数据时代的企业云会计财务共享中心资产管理研究[J].环渤海经济瞭望,2019(07):17.DOI:10.16457/j.cnki.hbhjjlw.2019.07.009.
- [5] 吴志林.基于数据驱动的实时路况估计方法研究及系统开发[D].武汉理工大学,2018.
- [6] 胡娟.基于移动混合开发技术的高校学工管理平台设计[J].信息技术与信息化,2021(12):213-216.
- [7] 陈亚帅.基于内容审核的数据泄露防护系统的设计与实现[D].山东大学,2020.DOI:10.27272/d.cnki.gshdu.2020.002896.
- [8] 导入“Nuxt.js”的要点和优点[J].个人电脑;工作站软件开发技术信息杂志,2020(Sep. TN. 425):56-60
- [9] 巢晟盛.基于SpringBoot微服务架构下前后端分离的MVVM模型浅析[J].电脑知识与技术,2021,17(23):128-129,141.
- [10] 邱健.基于微服务的社交电商系统的设计与实现[D].北京交通大学,2021.DOI:10.26944/d.cnki.gbfju.2021.001102.
- [11] 刘聪.基于隐私保护的房产数据管理平台的设计与实现[D].山东师范大学,2017.
- [12] 赵安江.闪烁晶体测试系统的研究与设计[D].华中科技大学,2020.DOI:10.27157/d.cnki.ghzku.2020.002659.
- [13] 王颖.Android移动终端远程侦控系统的设计与实现[D].西安科技大学,2017.
- [14] 廖娟娟.轨道交通行业物资采购管理系统的应用与实践[J].科学技术创新,2020(34):99-100.DOI:10.3969/j.issn.1673-1328.2020.34.044.
- [15] 黄柏霖.中外经典诗歌信息系统设计与实现[D].华中师范大学,2017.
- [16] 吉立建.公交一卡通运营管理系统的应用与实践[D].华南理工大学,2016.
- [17] 邹建平.软件测试基础及测试过程和策略探索[J].价值工程,2015,34(9):319-322.
- [18] Ronald J. Leach. Introduction to Software Engineering, Second Edition[M]. Taylor and Francis;CRC Press:2016-02-16.
- [19] Mak Martijn,Heijungs Reinout. Environmental Externalities of Secondhand Markets—Based on a Dutch Auctioning Company[J]. Sustainability,2022,14(3).

致谢

感谢胡晓光老师。

感谢各位评委百忙之中审阅我的论文。