诚信声明

本人郑重声明：本人所呈交的毕业论文 (设计)，是在导师的指导下独立进行研究所取得的成果。毕业论文 (设计) 中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人或集体已经发表或在网上发表的论文。

特此声明。

论文作者签名：孟争斌

日 期：2023年 2月18日

# 摘 要

随着移动互联时代的飞速发展，微信小程序在线上和线下的融合越来越紧密，其已然成为一个互联网的大平台。小程序作为一种新型的移动互联工具，凭借其天然优势，也成为了众多创业者关注的对象。在当下的移动互联网时代，电子商务与移动社交网络也在不断地融合，微信、微博、抖音等应用也成为新时期的宠儿，人们对于移动社交网络平台和软件的需求也日益剧增。

同时，随着“互联网+”的深入发展，人们对于社区生活有着越来越多的需求。作为一种新型的社区生活方式，用户需求多元化带来了新的O2O模式，社区O2O是社区用户进行线上线下服务场景化需求，通过平台提供丰富多样的生活方式实现社区生活场景化、生活化。而在微信小程序这个强大的 APP平台上可以实现上述的一切功能。基于微信小程序所具有的巨大优势，让其成为了新时代互联网营销平台中最具潜力和前景的一个产品，其在微信小程序上有很好的用户体验及用户粘性优势。

社区O2O系统主要功能是帮助商户实现在线下服务的延伸，将居民与商户连接起来，形成一种新型的生活方式。O2O是线上线下相结合，通过互联网手段将线上订单和客户带到线下。O2O社区系统是将线下服务转移到线上来，并实现线上和线下的融合。

本文将以微信小程序为平台，对当前市场上存在的O2O社区系统进行研究与分析，提出一种基于微信小程序平台为O2O社区系统提供服务平台的解决方案。首先从该系统运行特点出发，分析其实现方式；然后通过对其设计、开发和等进行了研究；最后将该系统运用于一个真实项目中去测试与实际应用。本系统在设计时主要以“生活服务”为主进行开发，并能够根据用户需求灵活提供线上线下联动和一站式便捷社区服务体验。 具体内容包括：线上线下场景化联动，如友邻集市、服务热线、投诉建议等；线下服务入口，如帮送快递、帮丢垃圾、帮送外卖、帮购商品等。本文通过阐述微信小程序在社区中的应用价值、发挥的作用以及在实际应用中所遇到的问题，并针对该问题提出相应的解决方案。

**关键词：O2O社区系统；微信小程序；智慧物业社区**

# Abstracts

The integration of WeChat applets,both online and offline,is becoming increasingly more likely due to the swift advancement of mobile Internet,

thus making it a major part of the Internet. As a new type of mobile Internet tool, small programs have also become the object of attention for many entrepreneurs by virtue of their natural advantages. In the current mobile Internet era, e-commerce and mobile social networks are also continuously integrating, and applications such as WeChat, Weibo and Jitterbug have become the darlings of the new era, and people's demand for mobile social network platforms and software is also increasing dramatically.

 At the same time, with the in-depth development of "Internet+", people have more and more demands for community life. As a new type of community lifestyle, the diversification of user needs brings a new O2O model, community O2O is the community users to carry out online and offline service scenario needs, through the platform to provide a rich and diverse lifestyle to achieve community life scenario, life. And in WeChat applet, a powerful APP platform, all the above functions can be realized[16]. Based on the huge advantages that WeChat applet has, it has become one of the most potential and promising products in the new era of Internet marketing platform, which has good user experience and user stickiness advantages on WeChat applet.

The main function of community O2O system is to help merchants realize the extension of offline services and connect residents with merchants to form a new lifestyle.O2O is the combination of online and offline, bringing online orders and customers offline through Internet means.The O2O Community System is the transfer of the offline services to the Internet and the integration of the Internet and the offline.

This paper will use WeChat applet as a platform to study and analyze the O2O community system existing in the current market, and propose a solution based on WeChat applet platform to provide service platform for O2O community system. Firstly, we analyze the implementation of the system from its operation characteristics; then we study its design, development and so on; finally, we apply the system to a real project for testing and practical application. The system is designed with "life services" as the main focus of development, and can flexibly provide online and offline linkage and one-stop convenient community service experience according to user needs. The specific contents include: online and offline scenario-based linkage, such as friends and neighbors market, service hotline, complaints and suggestions, etc.; offline service portal, such as help to deliver, help to throw away garbage, help to deliver take-away, help to buy goods, etc. This paper elaborates the application value of WeChat applets in the community, the role they play and the problems encountered in practical application, and proposes corresponding solutions to the problem.

**Keywords: O2O community system; WeChat mini program; Smart community**

# **目 录**

[摘 要 3](#_Toc7992)

[Abstracts 4](#_Toc23966)

[目 录 6](#_Toc18940)

[第一章 绪论 8](#_Toc8671)

[1.1研究背景 8](#_Toc18480)

[1.2研究目的和意义 8](#_Toc31870)

[1.3国内外社区系统应用现状 8](#_Toc5699)

[1.4论文的主要内容和结构 9](#_Toc24472)

[第二章 相关技术 10](#_Toc20381)

[2.1小程序架构设计 10](#_Toc11180)

[2.2开发语言与开发工具 10](#_Toc8856)

[2.2.1开发技术 10](#_Toc11833)

[2.2.2开发工具 12](#_Toc17686)

[第三章 O2O系统分析 14](#_Toc29301)

[3.1可行性分析 14](#_Toc30438)

[3.1.1 技术可行性 14](#_Toc23703)

[3.1.2 经济可行性 14](#_Toc18533)

[3.1.3 法律可行性 15](#_Toc3206)

[3.1.4 操作可行性 15](#_Toc20211)

[3.2 系统设计原则 15](#_Toc10989)

[3.3 性能分析 16](#_Toc32304)

[3.4 业务流程分析 17](#_Toc7170)

[3.4.1登录流程 17](#_Toc21546)

[3.4.2新增信息流程 18](#_Toc11657)

[3.4.3删除信息流程 19](#_Toc14297)

[第四章 系统总体设计 21](#_Toc16850)

[4.1 系统架构设计 21](#_Toc9373)

[4.2系统的总体结构 21](#_Toc18373)

[4.4 数据库设计 23](#_Toc21154)

[4.4.1 数据对象ER图 23](#_Toc4157)

[4.4.2 数据表 25](#_Toc19274)

[第五章 系统开发与实现 30](#_Toc21305)

[5.1用户端 30](#_Toc4629)

[5.1.1用户登录 30](#_Toc3000)

[5.1.2 发布功能 33](#_Toc3485)

[5.2管理端 35](#_Toc18223)

[5.2.1 管理端登录 35](#_Toc31139)

[5.2.2 菜单管理 41](#_Toc11775)

[5.2.3 通知公告 43](#_Toc12855)

[5.2.4 审计日志 45](#_Toc29783)

[5.2.5 服务菜单 48](#_Toc27019)

[5.2.6 社区管理 50](#_Toc8036)

[5.2.7 业主管理 53](#_Toc9454)

[第六章 系统测试 56](#_Toc10429)

[6.1 系统测试的目的 56](#_Toc765)

[6.2 测试方案 56](#_Toc6759)

[6.2.1 测试策略 56](#_Toc7189)

[6.2.2 功能测试 57](#_Toc20471)

[6.2.3 性能测试 57](#_Toc6015)

[6.3 测试特性及分析 57](#_Toc11656)

[6.4 测试结果 58](#_Toc8957)

[结 论 59](#_Toc25507)

[参考文献 60](#_Toc10129)

[致 谢 61](#_Toc20313)

# 第一章 绪论

## 1.1研究背景

随着科技的发展，微信、QQ、新浪微博、抖音等一系列具有强大功能的移动端软件纷纷出现，而微信小程序更是在当今的移动互联网领域取得了巨大的成就，成为当今移动互联网行业的一个里程碑。微信作为一个巨大的应用平台，拥有着广阔的发展前景。通过对这些应用平台进行研究和分析，可以发现基于微信构建的小程序有很大潜力和价值[8]。基于新时期的背景下，O2O系统成为了市场发展最具前景的一个方向，而将这一功能运用到生活中去也让其具有更大价值。本文将以微信小程序为平台，进行社区O2O系统开发应用，并通过实施案例分析该方案目前所存在问题及解决对策等相关内容。

## 1.2研究目的和意义

随着人们对于移动社交网络的需求与日俱增，构建以互联网为基础的新型社区生活系统也是十分的有必要，而微信小程序则成为了新时期移动互联的一个重要工具，其具有的便捷性、开放性、快捷性以及高效率等特点让其成为了社区O2O系统建设中最具潜力和前景的一个产品。在微信小程序平台上构建“O2O社区系统”将线上服务向线下延伸，实现真正社区生活服务一站式体验，同时又能将用户与商户连接起来形成一种新型的生活方式[3]。微信小程序可以有效地解决用户在线消费、线下服务和社区管理方面缺乏支持的问题，从而提升用户体验。在本系统构建中，利用微信小程序平台提供的强大功能可以更好地实现线上线下一体化整合应用。

## 1.3国内外社区系统应用现状

随着互联网技术的不断发展，人们对于生活的需求变得越来越多元化，社区O2O系统的出现正是为了满足广大居民的多样化需求，并取得了迅猛的发展。目前来看，国内外都有大量的社区O2O系统，国内也有很多优秀的平台[9]。

在国外方面， Google Earth、 Amazon Fargate等都推出了各自的“O2O社区系统”，这类系统通过将用户与商户连接起来，从而实现用户在线上和线下的联动[13]。另外，国外一些城市为了方便本地居民和商户，提供了相应的社区O2O系统的使用方法。

在国内方面，2015年12月17日微信小程序正式上线公测，作为互联网时代一款备受关注的社交软件，微信小程序一直备受创业者的青睐。截至目前为止腾讯官方已经推出了6个版本的“O2O社区系统”[4]。微信小程序具有显著的优势，使其成为移动互联网时代的一个重要的基础设施和平台。

## 1.4论文的主要内容和结构

本O2O智慧社区系统被划分为六个章节，以便对系统的实现进行详细阐述。

第一章绪论，本章以智慧社区系统开发的背景为基础，对系统实施后给社区居民带来便利性进行了系统性的叙述和概括[15]。

第二章相关技术，本章主要针对智慧社区系统开发用到的各项技术，包括了Java 开发语言、MySQL数据库以及微信开发者工具、VS Code、IntelliJ IDEA等进行了介绍。

第三章对智慧社区系统进行了可行性分析，并为系统的有效开发提供了设计原则、性能分析、系统运行等方面的数据支持。

第四章系统设计，对智慧社区系统的每个功能的模块设计、主要的实施过程、数据库之间的关系和结构设计进行书面说明。

第五章系统实现，本章将对智慧社区系统的各个用户模块的实现内容、实施步骤、界面截图以及说明进行详尽的阐释。

第六章对智慧社区系统的测试方法、功能测试的内容和测试结果进行了完整的描述，以及对该系统的性能进行评估。

# 第二章 相关技术

## 2.1小程序架构设计

小程序是一种具有便捷性、高效性、开放性、封闭性以及共享性等特点的手机端应用程序，对于用户而言，小程序的使用非常便捷，只需要轻轻一点就可以进入。根据小程序的特性、开发流程以及相关技术的要求，确定了其架构和功能。在小程序的整体架构设计中，主要包括前端展示页面以及后台服务管理后台等两部分。管理后台通过服务器进行部署，而前端部分的代码可以在用户的移动端上实现直接运行。

小程序平台通过前端和后端的架构设计将整个应用的操作流程和用户使用习惯更加合理地进行安排，而在具体的开发过程中则可以根据业务需求对系统进行功能设计和调整[2]。对于微信小程序的架构设计，必须具备对业务逻辑的清晰理解，并能够通过编写代码实现。

## 2.2开发语言与开发工具

由于本系统以O2O社区服务为主，因此对于开发语言的选择需要慎重。在选择开发工具时，应当综合考虑小程序的设计与实施情况、项目开发成本、风险因素、业务逻辑、开发难度以及可能存在的风险等因素，以确保小程序的有效开发[5]。

通过以下这些开发技术和工具的支撑，可以使智慧社区系统的开发工作更加顺利，同时不仅可以有效降低开发成本，而且还能够有效避免一些常见的开发问题，更能够根据客户的需求，提供全面、完善的解决方案。

### 2.2.1开发技术

本O2O社区系统具有较强的跨平台性，所以本系统前端选用 NodeJS作为开发平台，以Vue为主要的开发语言，uni-app作为用户端的开发框架。对于服务端采用 Java语言、框架为Spring Boot、数据库为MySQL。

1. Node JS：是一款基于谷歌V8引擎的开源框架，它通过事件驱动和NIO模型，使JavaScript 可以在服务器端实现高效、可靠的开发。它具有学习成本、开发成本相对较低的优势，很受大家的青睐。
2. Vue框架：是一个基于HTML、CSS 和 JavaScript实现的一套声明式的、组件化的MVVM前端框架，是一个简单易用、高的构建前端界面的开发利器。它是一个框架，也是一个生态。其功能覆盖了大部分前端开发常见的需求。但 Web 世界是十分多样化的，不同的开发者在 Web 上的构建形式和规模有很大差异。因此，Vue的设计非常注重灵活性和“可以被逐步集成”这个特点。根据你的需求场景，可以用不同的方式使用 Vue：
3. 无需构建步骤，渐进式增强静态的 HTML
4. 在任何页面中作为 Web Components 嵌入
5. 单页应用 （SPA）
6. 全栈 / 服务端渲染 （SSR）
7. Jamstack / 静态站点生成 （SSG）
8. 开发出各种不同的界面，包括桌面、移动、WebGL和命令行终端。
9. Java语言：其拥有多种独特的功能，其中包括：支持多种平台、面向对象、执行效率高、多线程等。在软件开发中以Java面向对象的特性，可以为软件开发提供了易于理解和简化开发，且能解决许多实际的开发问题。 Java语言的一个重要优势就是它能够跨越多种不同的操作系统，这使得它的可靠性和可扩展性得到了显著的提高。
10. Spring Boot：其不仅拥有Spring框架IOC、APO等特点，此外，由于Spring简化了配置，使得Java软件的开发变得更加高效、便捷。
11. MySQL：它是一个非常出色的RDBMS，其也是WEB应用技术的重要组成部分。同时，MySQL具有极高的数据存储的灵活性，可以将数据根据业务需要存储在不同的表里，从而大大提升了系统的效率。MySQL在不同操作系统上进行数据的CURD表现都非常出色，为数据存储和数据管理提供了安全可靠的支持。

### 2.2.2开发工具

小程序的开发工具是决定整个小程序性能和质量的重要因素之一。前后端使用不同的开发工具，前端开发工具为微信开发者工具和VS Code。服务端以IDEA为开发工具。

1. Visual Studio Code：Microsoft公司开发的一款功能强大、插件众多的代码编写软件，它可以用于编写各种Web应用，且可以在桌面上运行。这款软件拥有强大的TypeScript、JavaScript以及NodeJs内置功能，同时还支持Java、Python、Go等多种语言。
2. 微信开发者工具：由Tencent的开发团队精心设计和开发的，旨在满足开发者对于微信小程序开发的不同需求。其在开发过程中被广泛的应用，并且持续的迭代更新和改进为开发者带来更优质的用户体验。通过扫码可以进入开发测试环境进行真机测试，从而可以进一步完善更多的功能需求。它有以下特点：
3. 根据不同手机的屏幕尺寸大小，可以对小程序进行精确的切割，从而可以适配不同型号的手机屏幕。
4. 在完成视图布局之后，点击编译按钮，可以快速地更新视图界面。
5. 通过使用控制台，您可以轻松地进行调试和打印输出相关内容。
6. 可以输入版本号和相应的注释将代码上传至腾讯服务器，为确保代码的准确性，需要经过微信团队严格的审查流程后方可完成代码上传。
7. 通过分析资源文件，我们可以对相关项目的文件进行精确的断点调试，以达到最佳的效果。
8. 使用远程调试工具，无论是在手机端还是PC端，都能为用户提供实用的帮助。
9. 采用子父层级的结构来展示组件，可以显著提升视图调试的效率和便捷性。
10. 微信的代码大小必须小于2M； 在开发过程中，必须对域名进行合法性审查；在使用小程序时，必须为其配置一个安全的服务器域名以确保安全性。

在本系统实施过程中，为了减少对系统功能设计的不利影响，将对前端页面做了如下优化：

1. 减少前端页面尺寸，优化整体布局。
2. 优化小程序界面 UI，使界面更加美观。
3. 在小程序中采用微信自带的控件实现与服务端数据的存储和交互。
4. 利用微信小程序自身的功能与设计规范，将系统界面统一成符合微信小程序风格的布局与颜色来提升用户体验。
5. 将页面数据逻辑、数据同步与服务器端数据库连接同步。

# 第三章 O2O系统分析

本章将对该系统在多方面进行相关分析，由此可以帮助在开发智慧社区小程序系统时了解自己需要做哪些事情，从而提高开发设计的效率和更好地进行相关开发工作的安排。应当进行全面的市场调研以确保目标的明确性，并对需求进行精细的分析，以便为整个系统的开发提供准确的指导。

## 3.1可行性分析

可行性分析主要是评估系统需求、技术能力和成本效益，以确定其可行性。完成设计的智慧社区管理小程序可能会受到时间和空间的限制[14]。为了确保项目的成功，我们必须对其可行性进行细致的评估，并预防风险和控制成本的投入。经过对其全面的评估，可以确保在项目开始实施之前达到预期的目标和要求。

### 3.1.1 技术可行性

技术可行性是评估软件开发质量的关键因素，它能够帮助我们更好地利用先进的技术来提升项目的整体表现，从而达到更高的成功率。本系统技术方面主要使用uni-app、Vue、Springboot、MySQL等成熟的开源技术。其中uni-app具有多端打包部署的优势，为后续系统多端扩展和部署提供了有利支撑。Springboot已经是在Java开发技术领域经过多年的开发验证的Java服务端框架，其可以提高系统的稳定性、安全性和提高开发效率，并达到预期的开发目标。MySQL作为主流的RDBMS，也是WEB应用的重要组成部分，其数据操作简单、管理高效、安全可靠、多平台兼容、完整的网络连接以及稳定的性能，为系统的数据存储提供了有力支撑[1]。

经过对相关技术栈的可行性分析，我们确认本系统采用的技术均能够满足技术要求。

### 3.1.2 经济可行性

任何软件都需要考虑在开发、设计、后续运营方面投入的成本以及评估该软件能否创造真正的经济效益。

本智慧社区系统小程序完全由开发者自行完成，无需购买任何软件或端口，在开发之前，开发者可以免费进行市场调研和搭建开发环境，这些都是免费的，完全取决于开发者的技术能力和经验。如果遇到自己无法解决的问题，可以通过互联网相关技术论坛求助或者寻求相关行业比较资深的同学朋友给予一定的帮助[7]。也可以通过与同学和指导老师的协作，我们可以更有效地获取相关信息，从而实现智慧社区系统的开发，不仅可以获得显著的经济效益，而且还能够有效降低财务开支。本系统采用的技术栈均为成熟的开源技术，可有效减少资金投入和降低开发成本。

### 3.1.3 法律可行性

系统能否有效地部署运行，首先必须在设计和开发系统平台时，确保其符合国家法律法规，所有相关执行落地时也需要以我们国家的法律法规为前提。由于我们在开发过程中采用的均为开源技术,不涉及任何侵权行为,且符合相关开源法规,因此本系统在法律上是可行的。

### 3.1.4 操作可行性

任何软件系统在设计的时候都需要考虑该系统的用户使用体验。操作的可行性是指系统操作和操作过程不能过于繁琐，应在系统的功能使用基础上提高用户的操作体验[6]，智慧社区系统以 SSM框架为基础，其具有简单的管理功能和友好且易上手的界面设计。用户只需通过互联网进入管理系统就可以进行相关数据的维护。本系统具有较好的使用体验，它完善且灵活的功能既实现了实际的需求，又提高了工作效率。其次就是系统的功能方面了，只要会一些基本电脑操作的使用者就可以轻松完成相关功能的操作。

## 3.2 系统设计原则

1. 关于智慧社区管理小程序的基本要求
2. 环境要求：能够适应多种不同的系统，包括Windows、Linux等。
3. 功能要求：实现首页管理、用户管理、社区管理、服务热线、投诉建议、友邻市集、帮送帮取、快递取送、外卖送达、垃圾处理、商品购买等功能模块的集成，以满足用户的多样化需求。
4. 性能：能够保证在不同操作系统下的整体运行速度及其稳定性符合预期指标。
5. 数据安全要求：用户账号须验证通过才能访问该系统，并且其授权须根据其角色权限的限制以确保数据的安全。
6. 开发目标
7. 实现社区信息化为社区用户提供全面性、精细化和便捷性的生活需求。
8. 通过提升用户体验，实现运维人员对信息的有效管理。
9. 为用户提供简单友好的界面操作；
10. 经济效益

设计系统的目的在于实现人力成本和各种开支的有效降低，因此必须考虑系统开发的成本和利益。

## 3.3 性能分析

本O2O社区小程序系统采用分布式集群架构，以确保系统具有较高的并发性和性能运行效率。通过负载均衡策略来优化服务器端性能，以提升小程序运行时的性能，同时确保系统的稳定性。其包括：

（1）服务端：小程序服务器，为用户提供数据处理和计算服务。

（2）客户端：小程序的用户操作应用，为用户提供简单友好的界面操作。

（3）数据库：使用RDBMS和Redis缓存相结合。为系统数据处理提高性能。

（4）负载均衡控制服务器端性能优化。本小程序采用多数据源配置，大大缩短了数据处理时间，进而提升了系统运行效率。同时也使得整个小程序系统的运维更加便捷、高效[10]。

## 3.4 业务流程分析

### 3.4.1登录流程

为了确保系统安全，用户应在登录窗口中输入完整且准确的个人信息，并点击登录按钮以确保安全性。只有在后台程序经过验证后，用户才能够进行相关的功能操作；若登录校验失败，系统将会显示错误提示，要求用户重新登录，如图3.4.1所示。

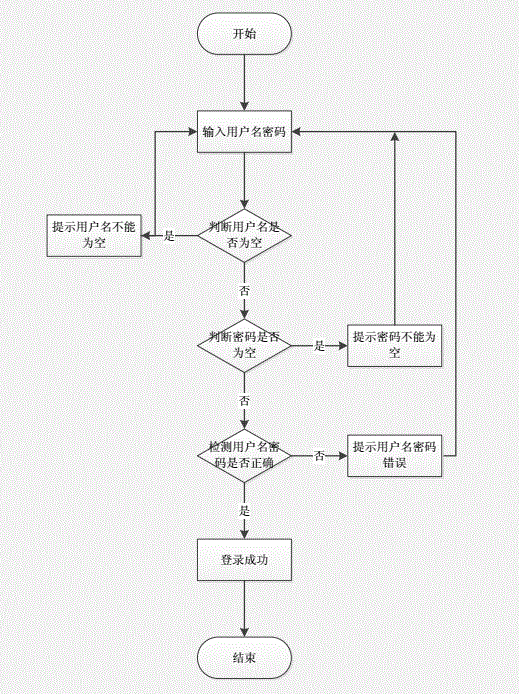
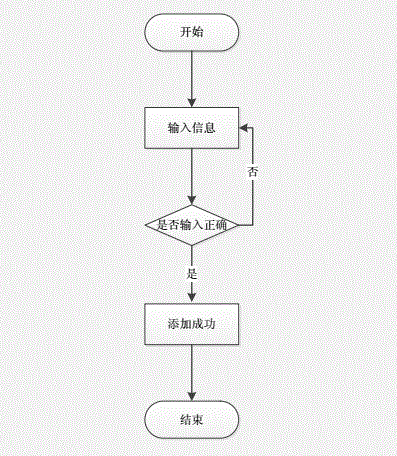


图3.4.1 登录流程图

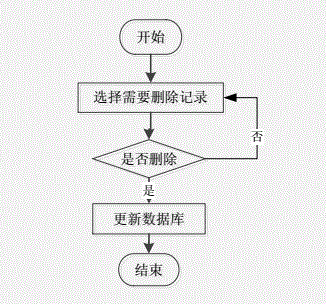
### 3.4.2新增信息流程

为了使智慧社区小程序更有效地运行，用户可以在添加界面中按照指定的格式和内容输入自己想要的数据，若填写都符合要求，则可确认添加成功；若填写不符合要求，则需重新输入，以确保最终结果符合预期的标准。如图3.4.2所示。

图3.4.2数据新增流程图

### 3.4.3删除信息流程

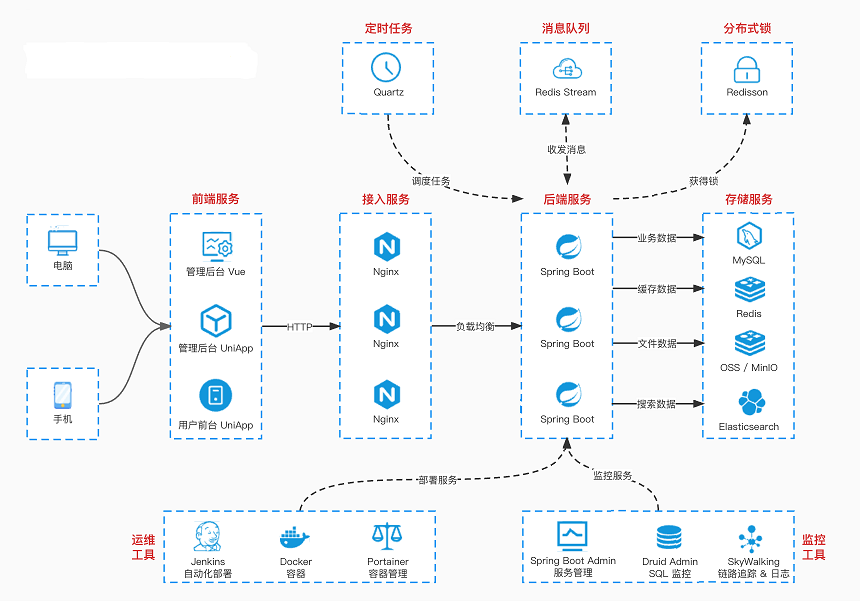
管理用户成功登录后，可根据已授权的功能进行相关操作。当其尝试删除数据时,必须进行相关校验和确认,以确保操作的正确性。若校验通过，方可执行删除操作,否则不进行任何执行。如图3.4.3 所示。

图3.4.3 数据删除流程

# 系统总体设计

## 4.1 系统架构设计

本系统采用B/S架构，由客户端、管理端、服务器端和数据中心组成。客户端基于uni-app、vue2开发，管理端基于vue3、typescript4等技术栈开发，服务端基于springboot、MyBatis框架的结合，数据中心主要由MySQL、Redis、MinIO来存储相关数据[11]。整个系统为前后端分离部署，采用Nginx反向代理技术，将前端请求转发至后端服务器，以实现数据处理和分析。如图4.1所示。

4.1 技术架构图

## 4.2系统的总体结构

本O2O智慧社区系统功能主要由用户端包括：首页、社区、需求发布等功能和管理端包括：用户管理、社区信息、业主信息、菜单管理、日志管理等两部分组成。整体使用模块化的架构，实现了各个部分的完全独立，并且彼此之间没有任何干扰，从而大大简化了系统的设计和开发。社区用户端如4.2-1所示，而管理端则如4.2-2所示。

|  |  |
| --- | --- |
| a4b77cd3ca6e904fc39b00144bed0f0 | 68d22657935719cc34ee634763271a7 |

4.2-1用户端图

|  |
| --- |
| 图片1 |

4.2-2管理后台图

## 4.4 数据库设计

完善的数据库架构是实现信息系统成功的基础关键。数据库能够为程序提供准确的数据信息，包括数据的存储大小、数据类型和结构，从而为开发提供更加高效、准确的数据存储服务。通过对比，选择使用MySQL既能保证数据的安全，又能保证数据的稳定且其具有出色的自我维护功能，以E-R图为例，可以直观展示出系统内部各个数据库表和表字段之间的关系。

### 4.4.1 数据对象ER图

通过构建概念模型,我们可以将实际世界的对象信息抽象化,从而更加清晰地表达和描述它们。在构建数据库时,我们需要首先确定每个对象实体的特征及其之间的关联性,以便建立一个数据表来描述相互关联的信息。通过E-R图可以构建出一个更加逼真的数据库概念模型，从而更加准确地反映出实际世界中的事物对象。

1. 用户管理图，如图4.4.1-1所示，可以清晰地展示出用户对象的相关信息。

|  |  |
| --- | --- |
| D:\Files Data\WeChat\WeChat Files\menglove520815\FileStorage\Temp\1675592804920.jpg |  |

图4.4.1-1用户管理ER图

1. 服务菜单E-R图，如图4.4.2-2 图所示：

|  |  |
| --- | --- |
|  |  |

图4.4.1-2 服务菜单管理ER图

1. 社区信息E-R图，如图4.4.1-3所示：

|  |  |
| --- | --- |
|  |  |

图4.4.1-3社区信息ER图

### 4.4.2 数据表

经过精心的数据结构分析，我们可以清楚地辨别出各种表格与结构之间的联系，进而有效地检测、优化和改进数据库，使其具备更好的性能，满足用户的多样需求。根据所采用的DBMS的性能，我们重新定义并重新组织了数据库的概念框架。然而，该系统不仅可以完美地实现智慧社区的小程序功能，还具有明确的结构。

|  |
| --- |
| 1. 服务列表菜单（community\_menu） |

|  |
| --- |
| 1. 业主信息（community\_owner） |

|  |
| --- |
| 1. 小区信息（community\_community） |
| 1. 用户收件地址（community\_address） |

|  |
| --- |
| 1. 菜单权限表（system\_menu） |
| 1. 通知公告表（system\_notice） |
| 1. 操作日志记录（system\_operate\_log） |

# 第五章 系统开发与实现

本章旨在阐述系统的整体架构及其实现方法。主要介绍系统的各个功能，并展示具体实现的关键逻辑代码；其次，主要为实现本系统所需解决方案，以及该设计方案在实际使用中出现的问题。最终，我们展示了一个实际操作界面，详细描述了小程序中各功能模块的开发过程。

## 5.1用户端

### 5.1.1用户登录

本系统登录模块实现社区用户移动端的注册/登录，并为其提供安全的身份认证。首先，用户需要登录社区系统，在用户界面中点击“我的-登录/注册”选项，以完成注册、登录和信息维护。根据系统要求设置手机号、密码等，然后将需要提交的信息填入表单中，并保存到数据库中。该功能实现了用户的账户注册和登录认证，使登录系统更安全，更可靠。

1. 注册/登录，如图5.1.1所示：

|  |  |  |
| --- | --- | --- |
| C:\Users\Administrator\AppData\Local\Temp\WeChat Files\dbfe738cee43879246712342b18566a.png | C:\Users\Administrator\AppData\Local\Temp\WeChat Files\3fab8d2933ce5c51e7046d5e49df5a1.png | C:\Users\Administrator\AppData\Local\Temp\WeChat Files\4297739abc703bb05cb039bbc59bccf.png |

图5.1.1 用户APP端登录图

1. 用户APP端登录的核心逻辑代码如下：

@Api（tags = ”用户端 - 认证”）

@RestController

@RequestMapping(”/community/mini/auth”)

@Validated

@Slf4j

public class AppAuthController {

@Resource

private SecurityProperties securityProperties;

@Resource

private CommunityAuthService authService;

@PostMapping(”/login”)

@ApiOperation（”使用手机 + 密码登录”）

public CommonResult<AppAuthLoginRespVO> login(@RequestBody @Valid AppAuthLoginReqVO reqVO) {

return *success*(authService.login(reqVO));

}

}

*/\*\**

*\* 会员的认证 Service 接口*

*\*/*

@Service

@Slf4j

public class CommunityAuthServiceImpl implements CommunityAuthService {

@Resource

private CommunityUserService userService;

@Resource

private SmsCodeApi smsCodeApi;

@Resource

private LoginLogApi loginLogApi;

@Resource

private SocialUserApi socialUserApi;

@Resource

private OAuth2TokenApi oauth2TokenApi;

@Resource

private WxMaService wxMaService;

@Resource

private PasswordEncoder passwordEncoder;

@Resource

private CommunityUserMapper userMapper;

@Override

public AppAuthLoginRespVO login(AppAuthLoginReqVO reqVO) {

/ /通过手机号码和密码进行登录。

CommunityUserDO user = login0(reqVO.getMobile(), reqVO.getPassword());

// 如果 socialType 非空，说明需要绑定社交用户

if (reqVO.getSocialType() != null) {

socialUserApi.bindSocialUser(new SocialUserBindReqDTO(user.getId(), getUserType().getValue(),

reqVO.getSocialType(), reqVO.getSocialCode(), reqVO.getSocialState()));

}

// 创建 Token 令牌，记录登录日志

return createTokenAfterLoginSuccess(user, reqVO.getMobile(), LoginLogTypeEnum.*LOGIN\_MOBILE*);

}

private AppAuthLoginRespVO createTokenAfterLoginSuccess(CommunityUserDO user, String mobile, LoginLogTypeEnum logType) {

// 插入登陆日志

createLoginLog(user.getId(), mobile, logType, LoginResultEnum.*SUCCESS*);

// 创建 Token 令牌

OAuth2AccessTokenRespDTO accessTokenRespDTO = oauth2TokenApi.createAccessToken(new OAuth2AccessTokenCreateReqDTO()

.setUserId(user.getId()).setUserType(getUserType().getValue())

.setClientId(OAuth2ClientConstants.*CLIENT\_ID\_DEFAULT*));

// 构建返回结果

return AuthConvert.*INSTANCE*.convert(accessTokenRespDTO);

}

}

### 5.1.2 发布功能

智慧社区系统小程序发布的功能包括：帮取快递、帮送外卖、帮丢垃圾、帮购商品等。如图5.1.2-1 、5.1.2-2、5.1.2-3、5.1.2-4所示。

|  |  |
| --- | --- |
| C:\Users\Administrator\AppData\Local\Temp\WeChat Files\68d22657935719cc34ee634763271a7.jpg图5.1.2-1 帮取快递发布界面 | C:\Users\Administrator\AppData\Local\Temp\WeChat Files\36460ba2c91b6099b673ceacff4dd25.jpg图5.1.2-2 帮送外卖发布界面 |
| 图5.1.2-3 购丢垃圾发布界面C:\Users\Administrator\AppData\Local\Temp\WeChat Files\a9afcd694c9f44bd36154948d689022.jpg | C:\Users\Administrator\AppData\Local\Temp\WeChat Files\80016750799a8c7e3c3825c89c76f82.jpg图5.1.2-4 购买商品发布界面 |

## 5.2管理端

### 5.2.1 管理端登录

在智能社区的管理端，用户登录的主要功能包括：

1、权限控制：根据不同角色加载相应功能和数据。

2、系统登录：通过输入账户、密码、滑动验证码等信息进行登录。

3、用户信息：例如：修改密码、修改账号状态、重新登录等。

1. 管理员登录，如图5.2.1所示：

|  |
| --- |
| 图5.2.1 管理登录图 |

1. 管理端登录的关键逻辑代码如下：

@Api(tags = ”管理后台 - 认证”)

@RestController

@RequestMapping("/system/auth")

@Validated

@Slf4j

public class AuthController {

@Resource

private AdminAuthService authService;

@Resource

private AdminUserService userService;

@Resource

private RoleService roleService;

@Resource

private PermissionService permissionService;

@Resource

private SocialUserService socialUserService;

@Resource

private SecurityProperties securityProperties;

@PermitAll

@OperateLog(enable = false) // 避免 Post 请求被记录操作日志

@PostMapping("/login")

@ApiOperation(”使用账号密码登录”)

public CommonResult<AuthLoginRespVO> login(@RequestBody @Valid AuthLoginReqVO reqVO) {

return *success*(authService.login(reqVO));

}

@PostMapping("/logout")

@PermitAll

@ApiOperation(”登出系统”)

@OperateLog(enable = false) // 避免 Post 请求被记录操作日志

public CommonResult<Boolean> logout(HttpServletRequest request) {

String token = *obtainAuthorization*(request, securityProperties.getTokenHeader());

if (StrUtil.*isNotBlank*(token)) {

authService.logout(token, LoginLogTypeEnum.*LOGOUT\_SELF*.getType());

}

return *success*(true);

}

@PostMapping("/refresh-token")

@PermitAll

@ApiOperation(”刷新令牌”)

@ApiImplicitParam(name = ”refreshToken”， value = ”刷新令牌”， required = true， dataTypeClass = String.class)

@OperateLog(enable = false) // 避免 Post 请求被记录操作日志

public CommonResult<AuthLoginRespVO> refreshToken(@RequestParam("refreshToken") String refreshToken) {

return *success*(authService.refreshToken(refreshToken));

}

@GetMapping("/get-permission-info")

@ApiOperation(”获取登录用户的权限信息”)

public CommonResult<AuthPermissionInfoRespVO> getPermissionInfo() {

// 获得用户信息

AdminUserDO user = userService.getUser(*getLoginUserId*());

if (user == null) {

return null;

}

// 获得角色列表

Set<Long> roleIds = permissionService.getUserRoleIdsFromCache(*getLoginUserId*(), *singleton*(CommonStatusEnum.*ENABLE*.getStatus()));

List<RoleDO> roleList = roleService.getRolesFromCache(roleIds);

// 获得菜单列表

List<MenuDO> menuList = permissionService.getRoleMenuListFromCache(roleIds,

SetUtils.*asSet*(MenuTypeEnum.*DIR*.getType(), MenuTypeEnum.*MENU*.getType(), MenuTypeEnum.*BUTTON*.getType()),

*singleton*(CommonStatusEnum.*ENABLE*.getStatus()))； // 只要开启的

// 拼接结果返回

return *success*(AuthConvert.*INSTANCE*.convert(user, roleList, menuList));

}

}

*/\*\**

*\* Auth Service 实现类*

*\*/*

@Service

@Slf4j

public class AdminAuthServiceImpl implements AdminAuthService {

@Resource

private AdminUserService userService;

@Resource

private LoginLogService loginLogService;

@Resource

private OAuth2TokenService oauth2TokenService;

@Resource

private SocialUserService socialUserService;

@Resource

private MemberService memberService;

@Resource

private Validator validator;

@Resource

private CaptchaService captchaService;

@Resource

private SmsCodeApi smsCodeApi;

*/\*\**

*\* 验证码的开关，默认为 true*

*\*/*

@Value("${captcha.enable:true}")

private Boolean captchaEnable;

@Override

public AdminUserDO authenticate(String username, String password) {

final LoginLogTypeEnum logTypeEnum = LoginLogTypeEnum.*LOGIN\_USERNAME*;

/ /校验账号是否存在。

AdminUserDO user = userService.getUserByUsername(username);

if (user == null) {

createLoginLog(null, username, logTypeEnum, LoginResultEnum.*BAD\_CREDENTIALS*);

throw *exception*(*AUTH\_LOGIN\_BAD\_CREDENTIALS*);

}

if (!userService.isPasswordMatch(password, user.getPassword())) {

createLoginLog(user.getId(), username, logTypeEnum, LoginResultEnum.*BAD\_CREDENTIALS*);

throw *exception*(*AUTH\_LOGIN\_BAD\_CREDENTIALS*);

}

// 校验是否禁用

if (ObjectUtil.*notEqual*(user.getStatus(), CommonStatusEnum.*ENABLE*.getStatus())) {

createLoginLog(user.getId(), username, logTypeEnum, LoginResultEnum.*USER\_DISABLED*);

throw *exception*(*AUTH\_LOGIN\_USER\_DISABLED*);

}

return user;

}

@Override

public AuthLoginRespVO login(AuthLoginReqVO reqVO) {

// 校验验证码

verifyCaptcha(reqVO);

// 使用账号密码，进行登录

AdminUserDO user = authenticate(reqVO.getUsername(), reqVO.getPassword());

// 如果 socialType 非空，说明需要绑定社交用户

if (reqVO.getSocialType() != null) {

socialUserService.bindSocialUser(new SocialUserBindReqDTO(user.getId(), getUserType().getValue(),

reqVO.getSocialType(), reqVO.getSocialCode(), reqVO.getSocialState()));

}

// 创建 Token 令牌，记录登录日志

return createTokenAfterLoginSuccess(user.getId(), reqVO.getUsername(), LoginLogTypeEnum.*LOGIN\_USERNAME*);

}

private void createLoginLog(Long userId, String username,

LoginLogTypeEnum logTypeEnum, LoginResultEnum loginResult) {

// 插入登录日志

LoginLogCreateReqDTO reqDTO = new LoginLogCreateReqDTO();

reqDTO.setLogType(logTypeEnum.getType());

reqDTO.setTraceId(TracerUtils.*getTraceId*());

reqDTO.setUserId(userId);

reqDTO.setUserType(getUserType().getValue());

reqDTO.setUsername(username);

reqDTO.setUserAgent(ServletUtils.*getUserAgent*());

reqDTO.setUserIp(ServletUtils.*getClientIP*());

reqDTO.setResult(loginResult.getResult());

loginLogService.createLoginLog(reqDTO);

/ /更新最后登录时间。

if (userId != null && Objects.*equals*(LoginResultEnum.*SUCCESS*.getResult(), loginResult.getResult())) {

userService.updateUserLogin(userId, ServletUtils.*getClientIP*());

}

}

@Override

public AuthLoginRespVO refreshToken(String refreshToken) {

OAuth2AccessTokenDO accessTokenDO = oauth2TokenService.refreshAccessToken(refreshToken, OAuth2ClientConstants.*CLIENT\_ID\_DEFAULT*);

return AuthConvert.*INSTANCE*.convert(accessTokenDO);

}

@VisibleForTesting

void verifyCaptcha(AuthLoginReqVO reqVO) {

// 如果验证码关闭，则不进行校验

if (!captchaEnable) {

return;

}

// 校验验证码

ValidationUtils.*validate*(validator, reqVO, AuthLoginReqVO.CodeEnableGroup.class);

CaptchaVO captchaVO = new CaptchaVO();

captchaVO.setCaptchaVerification(reqVO.getCaptchaVerification());

ResponseModel response = captchaService.verification(captchaVO);

// 验证不通过

if (!response.isSuccess()) {

// 创建登录失败日志（验证码不正确)

createLoginLog(null, reqVO.getUsername(), LoginLogTypeEnum.*LOGIN\_USERNAME*, LoginResultEnum.*CAPTCHA\_CODE\_ERROR*);

throw *exception*(*AUTH\_LOGIN\_CAPTCHA\_CODE\_ERROR*, response.getRepMsg());

}

}

private AuthLoginRespVO createTokenAfterLoginSuccess(Long userId, String username, LoginLogTypeEnum logType) {

// 插入登陆日志

createLoginLog(userId, username, logType, LoginResultEnum.*SUCCESS*);

// 创建访问令牌

OAuth2AccessTokenDO accessTokenDO = oauth2TokenService.createAccessToken(userId, getUserType().getValue(),

OAuth2ClientConstants.*CLIENT\_ID\_DEFAULT*, null);

// 构建返回结果

return AuthConvert.*INSTANCE*.convert(accessTokenDO);

}

@Override

public void logout(String token, Integer logType) {

// 删除访问令牌

OAuth2AccessTokenDO accessTokenDO = oauth2TokenService.removeAccessToken(token);

if (accessTokenDO == null) {

return;

}

// 删除成功，则记录登出日志

createLogoutLog(accessTokenDO.getUserId(), accessTokenDO.getUserType(), logType);

}

}

### 5.2.2 菜单管理

通过菜单管理，用户可以轻松实现系统菜单的更新、删除、展示和关闭，并且可以实时监测菜单的变化，从而有效提升用户体验。

1. 菜单管理，如图5.2.2所示：

|  |
| --- |
| 图5.2.2 菜单管理 |

1. 菜单管理的关键逻辑代码如下：

@Api（tags = ”管理后台 - 菜单”）

@RestController

RequestMapping(”/system/menu”)

@Validated

public class MenuController {

@Resource

private MenuService menuService;

@PostMapping(”/create”)

@ApiOperation（”创建菜单”）

@PreAuthorize(”@ss.hasPermission(’system:menu:create’)”)

public CommonResult<Long> createMenu(@Valid @RequestBody MenuCreateReqVO reqVO) {

Long menuId = menuService.createMenu(reqVO);

return *success*(menuId);

}

@PutMapping(”/update”)

@ApiOperation（”修改菜单”）

@PreAuthorize(”@ss.hasPermission(’system:menu:update’)”)

public CommonResult<Boolean> updateMenu(@Valid @RequestBody MenuUpdateReqVO reqVO) {

menuService.updateMenu(reqVO);

return *success*(true);

}

@DeleteMapping(”/delete”)

@ApiOperation（”删除菜单”）

@ApiImplicitParam（name = ”id”， value = ”角色编号”， required= true， example = ”1024”）

@PreAuthorize(”@ss.hasPermission(’system:menu:delete’)”)

public CommonResult<Boolean> deleteMenu(@RequestParam(”id”) Long id) {

menuService.deleteMenu(id);

return *success*(true);

}

@GetMapping(”/list”)

@ApiOperation（value = ”获取菜单列表”， notes = ”用于【菜单管理】界面”）

@PreAuthorize(”@ss.hasPermission(’system:menu:query’)”)

public CommonResult<List<MenuRespVO>> getMenus(MenuListReqVO reqVO) {

List<MenuDO> list = menuService.getMenus(reqVO);

list.sort(Comparator.*comparing*(MenuDO::getSort));

return *success*(MenuConvert.*INSTANCE*.convertList(list));

}

}

### 5.2.3 通知公告

系统管理-通知公告拥有多种功能，包括新增、更改、删除、详细说明，它不仅可以保证APP和管理后台的通知信息的完整性，而且还可以提供加便捷的服务。

1. 通知公告，如图5.2.3所示：

|  |
| --- |
| 图5.2.3通知公告界面图 |

1. 通知公告的关键逻辑代码如下：

@Api(tags = ”管理后台 - 通知公告”)

@RestController

@RequestMapping("/system/notice")

@Validated

public class NoticeController {

@Resource

private NoticeService noticeService;

@PostMapping("/create")

@ApiOperation(”创建通知公告”)

@PreAuthorize("@ss.hasPermission('system:notice:create')")

public CommonResult<Long> createNotice(@Valid @RequestBody NoticeCreateReqVO reqVO) {

Long noticeId = noticeService.createNotice(reqVO);

return *success*(noticeId);

}

@PutMapping("/update")

@ApiOperation(”修改通知公告”)

@PreAuthorize("@ss.hasPermission('system:notice:update')")

public CommonResult<Boolean> updateNotice(@Valid @RequestBody NoticeUpdateReqVO reqVO) {

noticeService.updateNotice(reqVO);

return *success*(true);

}

@DeleteMapping("/delete")

@ApiOperation(”删除通知公告”)

@ApiImplicitParam(name = ”id”， value = ”编号”， required = true， example = ”1024”)

@PreAuthorize("@ss.hasPermission('system:notice:delete')")

public CommonResult<Boolean> deleteNotice(@RequestParam("id") Long id) {

noticeService.deleteNotice(id);

return *success*(true);

}

@GetMapping("/page")

@ApiOperation(”获取通知公告列表”)

@PreAuthorize("@ss.hasPermission('system:notice:query')")

public CommonResult<PageResult<NoticeRespVO>> pageNotices(@Validated NoticePageReqVO reqVO) {

return *success*(NoticeConvert.*INSTANCE*.convertPage(noticeService.pageNotices(reqVO)));

}

}

### 5.2.4 审计日志

系统管理-审计日志主要包含登录、操作两个日志模块，其中登录日志记录了用户端APP户的登录和管理端用户的登录时的一些重要信息，操作日志记录了对系统功能的增删改的操作信息。

1. 如图5.2.4-1、图5.2.4-2所示：

|  |
| --- |
| 图5.2.4-1登录日志图 |
| 图5.2.4-2操作日志图 |

1. 审计日志核心逻辑代码

@Api(tags = ”管理后台 - 操作日志”)

@RestController

@RequestMapping("/system/operate-log")

@Validated

public class OperateLogController {

@Resource

private OperateLogService operateLogService;

@Resource

private AdminUserService userService;

@GetMapping("/page")

@ApiOperation(”查看操作日志分页列表”)

@PreAuthorize("@ss.hasPermission('system:operate-log:query')")

public CommonResult<PageResult<OperateLogRespVO>> pageOperateLog(@Valid OperateLogPageReqVO reqVO) {

PageResult<OperateLogDO> pageResult = operateLogService.getOperateLogPage(reqVO);

// 获得拼接需要的数据

Collection<Long> userIds = CollectionUtils.*convertList*(pageResult.getList(), OperateLogDO::getUserId);

Map<Long, AdminUserDO> userMap = userService.getUserMap(userIds);

// 拼接数据

List<OperateLogRespVO> list = new ArrayList<>(pageResult.getList().size());

pageResult.getList().forEach(operateLog -> {

OperateLogRespVO respVO = OperateLogConvert.*INSTANCE*.convert(operateLog);

list.add(respVO);

// 拼接用户信息

MapUtils.*findAndThen*(userMap, operateLog.getUserId(), user -> respVO.setUserNickname(user.getNickname()));

});

return *success*(new PageResult<>(list, pageResult.getTotal()));

}

@ApiOperation(”导出操作日志”)

@GetMapping("/export")

@PreAuthorize("@ss.hasPermission('system:operate-log:export')")

@OperateLog(type = *EXPORT*)

public void exportOperateLog(HttpServletResponse response, @Valid OperateLogExportReqVO reqVO) throws IOException {

List<OperateLogDO> list = operateLogService.getOperateLogs(reqVO);

// 获得拼接需要的数据

Collection<Long> userIds = CollectionUtils.*convertList*(list, OperateLogDO::getUserId);

Map<Long, AdminUserDO> userMap = userService.getUserMap(userIds);

// 拼接数据

List<OperateLogExcelVO> excelDataList = OperateLogConvert.*INSTANCE*.convertList(list, userMap);

// 输出

ExcelUtils.*write*(response， ”操作日志.xls”， ”数据列表”， OperateLogExcelVO.class， excelDataList)；

}

}

### 5.2.5 服务菜单

服务菜单有新增、修改、开启/关闭、导出等功能，主要负责用户端APP首页服务菜单的数据维护。

1. 服务菜单。如图5.2.5所示：

|  |
| --- |
| 图5.2.5服务菜单界面图 |

1. 智慧社区服务菜单核心逻辑代码

@Api(tags = ”管理后台 - 社区服务列表菜单”)

@RestController

@RequestMapping("/community/menu")

@Validated

public class CommunityMenuController {

@Resource

private CommunityMenuService menuService;

@PostMapping("/create")

@ApiOperation(”创建服务列表菜单”)

@PreAuthorize("@ss.hasPermission('community:menu:create')")

public CommonResult<Integer> createMenu(@Valid @RequestBody MenuCreateReqVO createReqVO) {

return *success*(menuService.createMenu(createReqVO));

}

@PutMapping("/update")

@ApiOperation(”更新服务列表菜单”)

@PreAuthorize("@ss.hasPermission('community:menu:update')")

public CommonResult<Boolean> updateMenu(@Valid @RequestBody MenuUpdateReqVO updateReqVO) {

menuService.updateMenu(updateReqVO);

return *success*(true);

}

@DeleteMapping("/delete")

@ApiOperation(”删除服务列表菜单”)

@ApiImplicitParam(name = ”id”， value = ”编号”， required = true)

@PreAuthorize("@ss.hasPermission('community:menu:delete')")

public CommonResult<Boolean> deleteMenu(@RequestParam("id") Integer id) {

menuService.deleteMenu(id);

return *success*(true);

}

@GetMapping("/get")

@ApiOperation(”获得服务列表菜单”)

@ApiImplicitParam(name = ”id”， value = ”编号”， required = true， example = ”1024”)

@PreAuthorize("@ss.hasPermission('community:menu:query')")

public CommonResult<MenuRespVO> getMenu(@RequestParam("id") Integer id) {

CommunityMenuDO menu = menuService.getMenu(id);

return *success*(MenuConvert.*INSTANCE*.convert(menu));

}

@GetMapping("/list")

@ApiOperation(”获得服务列表菜单列表”)

@ApiImplicitParam(name = ”ids”， value = ”编号列表”， required = true， example = ”1024，2048”)

@PreAuthorize("@ss.hasPermission('community:menu:query')")

public CommonResult<List<MenuRespVO>> getMenuList(@RequestParam("ids") Collection<Integer> ids) {

List<CommunityMenuDO> list = menuService.getMenuList(ids);

return *success*(MenuConvert.*INSTANCE*.convertList(list));

}

@GetMapping("/page")

@ApiOperation(”获得服务列表菜单分页”)

@PreAuthorize("@ss.hasPermission('community:menu:query')")

public CommonResult<PageResult<MenuRespVO>> getMenuPage(@Valid MenuPageReqVO pageVO) {

PageResult<CommunityMenuDO> pageResult = menuService.getMenuPage(pageVO);

return *success*(MenuConvert.*INSTANCE*.convertPage(pageResult));

}

@GetMapping("/export-excel")

@ApiOperation(”导出服务列表菜单 Excel”)

@PreAuthorize("@ss.hasPermission('community:menu:export')")

@OperateLog(type = *EXPORT*)

public void exportMenuExcel(@Valid MenuExportReqVO exportReqVO,

HttpServletResponse response) throws IOException {

List<CommunityMenuDO> list = menuService.getMenuList(exportReqVO);

// 导出 Excel

List<MenuExcelVO> datas = MenuConvert.*INSTANCE*.convertList02(list);

ExcelUtils.*write*(response， ”服务列表菜单.xls”， ”数据”， MenuExcelVO.class， datas)；

}

}

### 5.2.6 社区管理

社区管理有新增、修改、详情、导出等功能，主要负责用户端APP首页服务菜单的数维护。

1. 社区管界，如图5.2.6所示

|  |
| --- |
| 5.2.6社区管理界面图 |

1. 智慧社区社区管理核心逻辑代码

@Api(tags = ”管理后台 - 社区信息”)

@RestController

@RequestMapping("/community/community")

@Validated

public class CommunityController {

@Resource

private CommunityService communityService;

@PostMapping("/create")

@ApiOperation(”创建小区信息”)

@PreAuthorize("@ss.hasPermission('community:community:create')")

public CommonResult<Integer> create(@Valid @RequestBody CommunityCreateReqVO createReqVO) {

return *success*(communityService.create(createReqVO));

}

@PutMapping("/update")

@ApiOperation(”更新小区信息”)

@PreAuthorize("@ss.hasPermission('community:community:update')")

public CommonResult<Boolean> update(@Valid @RequestBody CommunityUpdateReqVO updateReqVO) {

communityService.update(updateReqVO);

return *success*(true);

}

@DeleteMapping("/delete")

@ApiOperation(”删除小区信息”)

@ApiImplicitParam(name = ”id”， value = ”编号”， required = true)

@PreAuthorize("@ss.hasPermission('community:community:delete')")

public CommonResult<Boolean> delete(@RequestParam("id") Integer id) {

communityService.delete(id);

return *success*(true);

}

@GetMapping("/get")

@ApiOperation(”获得小区信息”)

@ApiImplicitParam(name = ”id”， value = ”编号”， required = true， example = ”1024”)

@PreAuthorize("@ss.hasPermission('community:community:query')")

public CommonResult<CommunityRespVO> get(@RequestParam("id") Integer id) {

CommunityDO communityDO = communityService.get(id);

return *success*(CommunityConvert.*INSTANCE*.convert(communityDO));

}

@GetMapping("/list")

@ApiOperation(”获得小区信息列表”)

@ApiImplicitParam(name = ”ids”， value = ”编号列表”， required = true， example = ”1024，2048)

@PreAuthorize("@ss.hasPermission('community:community:query')")

public CommonResult<List<CommunityRespVO>> getList(@RequestParam("ids") Collection<Integer> ids) {

List<CommunityDO> list = communityService.getList(ids);

return *success*(CommunityConvert.*INSTANCE*.convertList(list));

}

@GetMapping("/page")

@ApiOperation(”获得小区信息分页”)

@PreAuthorize("@ss.hasPermission('community:community:query')")

public CommonResult<PageResult<CommunityRespVO>> getPage(@Valid CommunityPageReqVO pageVO) {

PageResult<CommunityDO> pageResult = communityService.getPage(pageVO);

return *success*(CommunityConvert.*INSTANCE*.convertPage(pageResult));

}

@GetMapping("/export-excel")

@ApiOperation(”导出小区信息 Excel”)

@PreAuthorize("@ss.hasPermission('community:community:export')")

@OperateLog(type = *EXPORT*)

public void exportExcel(@Valid CommunityExportReqVO exportReqVO,

HttpServletResponse response) throws IOException {

List<CommunityDO> list = communityService.getList(exportReqVO);

// 导出 Excel

List<CommunityExcelVO> datas = CommunityConvert.*INSTANCE*.convertList02(list);

ExcelUtils.*write*(response， ”小区信息.xls”， ”数据”， CommunityExcelVO.class， datas)；

}

}

### 5.2.7 业主管理

业主管理有新增、修改、详情、导出等功能，主要负责用户端APP首页服务菜单的数据维护。

1. 业主管理，如图5.2.7图

|  |
| --- |
| 5.2.7业主管理界面图 |

1. 业主管理核心逻辑代码

@Api(tags = ”管理后台 - 业主信息”)

@RestController

@RequestMapping("/community/owner")

@Validated

public class OwnerController {

@Resource

private OwnerService ownerService;

@PostMapping("/create")

@ApiOperation(”创建业主信息”)

@PreAuthorize("@ss.hasPermission('community:owner:create')")

public CommonResult<Long> createOwner(@Valid @RequestBody OwnerCreateReqVO createReqVO) {

return *success*(ownerService.createOwner(createReqVO));

}

@PutMapping("/update")

@ApiOperation(”更新业主信息”)

@PreAuthorize("@ss.hasPermission('community:owner:update')")

public CommonResult<Boolean> updateOwner(@Valid @RequestBody OwnerUpdateReqVO updateReqVO) {

ownerService.updateOwner(updateReqVO);

return *success*(true);

}

@DeleteMapping("/delete")

@ApiOperation(”删除业主信息”)

@ApiImplicitParam(name = ”id”， value = ”编号”， required = true)

@PreAuthorize("@ss.hasPermission('community:owner:delete')")

public CommonResult<Boolean> deleteOwner(@RequestParam("id") Long id) {

ownerService.deleteOwner(id);

return *success*(true);

}

@GetMapping("/get")

@ApiOperation(”获得业主信息”)

@ApiImplicitParam(name = ”id”， value = ”编号”， required = true， example = ”1024”)

@PreAuthorize("@ss.hasPermission('community:owner:query')")

public CommonResult<OwnerRespVO> getOwner(@RequestParam("id") Long id) {

OwnerDO owner = ownerService.getOwner(id);

return *success*(OwnerConvert.*INSTANCE*.convert(owner));

}

@GetMapping("/list")

@ApiOperation(”获得业主信息列表”)

@ApiImplicitParam(name = ”ids”， value = ”编号列表”， required = true， example = ”1024，2048”)

@PreAuthorize("@ss.hasPermission('community:owner:query')")

public CommonResult<List<OwnerRespVO>> getOwnerList(@RequestParam("ids") Collection<Long> ids) {

List<OwnerDO> list = ownerService.getOwnerList(ids);

return *success*(OwnerConvert.*INSTANCE*.convertList(list));

}

@GetMapping("/page")

@ApiOperation(”获得业主信息分页”)

@PreAuthorize("@ss.hasPermission('community:owner:query')")

public CommonResult<PageResult<OwnerRespVO>> getOwnerPage(@Valid OwnerPageReqVO pageVO) {

PageResult<OwnerDO> pageResult = ownerService.getOwnerPage(pageVO);

return *success*(OwnerConvert.*INSTANCE*.convertPage(pageResult));

}

@GetMapping("/export-excel")

@ApiOperation(”导出业主信息 Excel”)

@PreAuthorize("@ss.hasPermission('community:owner:export')")

@OperateLog(type = *EXPORT*)

public void exportOwnerExcel(@Valid OwnerExportReqVO exportReqVO,

HttpServletResponse response) throws IOException {

List<OwnerDO> list = ownerService.getOwnerList(exportReqVO);

// 导出 Excel

List<OwnerExcelVO> datas = OwnerConvert.*INSTANCE*.convertList02(list);

ExcelUtils.*write*(response， ”业主信息.xls”， ”数据”， OwnerExcelVO.class， datas)；

}

}

# 第六章 系统测试

对系统整体进行检测，以确定是否存在与系统的技术要求不一致的地方，并针对发现的问题在规定的时限内进行正确的修正，从而避免重大的经济损失就是系统测试。为了防止未经授权的攻击，我们必须对系统内部的安全机制进行有效的验证和审查以确保安全性。安全测试的目的是检验系统的稳定性，以及通过多种手段和策略，以期望能够打破现有的防护措施。因此，为了确保系统的安全，我们必须寻求一种能够有效抵御入侵的最新技术。

## 6.1 系统测试的目的

为了确保系统的长期运行和更加完善，必须对系统进行一个完整严格的系统测试，以便及早发现和解决出现的问题，从而有效地帮助我们实现设计的目标。经过测试,我们可以评估软件的性能、操作的可靠性和效率,并发现存在的缺陷,从而提出有效的改进措施。本章旨在发现系统中的问题并解决，尽管修改这些问题会耗费大量的成本，但对于系统的健壮性和稳定性来说是必须做的一件事。

为了确保智慧社区管理小程序能够稳定高效地运行，我们必须对其各个功能进行全面的系统测试和验证，以确保其准确性和可靠性，并及时发现和纠正任何潜在的问题，以防止出现任何形式的失误，从而为用户提供更优质的服务。

## 6.2 测试方案

### 6.2.1 测试策略

在日常的系统测试中，我们经常采用结、功能两种迥异的检验方法来进行验证。通过结果测试，可以深入洞察系统内部的工作流程，并依据需求规范确保系统内部操作能够正常运行；功能测试旨在评估系统的完整性，以确保每一项功能都能够正常运行。其是一种全面、可靠的评估手段，可以有效地帮助用户获取更多的信息。

### 6.2.2 功能测试

对系统各模块的功能进行全面的评估，以确保每个功能都能够正常运行，这就是功能测试，包括常见的数据增删改查等操作。在测试过程中，如果没有对新开发的软件的内部架构进行充分的理解，就可能会将其视为一个无关紧要的黑盒，而在不加思考地输入信息之后，就会得到系统提供的错误反馈。功能测试是一种特殊的检测技术，它可以检测输入错误信息后，系统是否会出现异常反应，以确保系统的正常运行。

### 6.2.3 性能测试

通常使用如Jmeter、LoadRunner、QTP等一系列自动化的测试工具，来评估软件程序的整体功能[12]，并且可以将负载测试和压力测试结合起来，以更好地评估软件程序的性能的测试就是性能测试。尽管压力测试能够揭示出系统能够提供的最佳性能，但负载测试则能够更好地评估系统对于不断增长的负载的反应能力。

## 6.3 测试特性及分析

经过充分的测试，智慧社区小程序系统的所有基础功能都很完善，可以有效地满足用户的需求。无论是在系统内部的功能性能还是在用户体验方面，它都具有卓越的价值。现从以下几点测试特性进行分析：

（1）复杂性：测试是一个复杂的过程，其复杂性可以从多个角度进行估计。

尽管系统已经通过了测试，但由于缺乏充分的测试用例设计，仍有可能出现不足的情况，因此，为了确保系统后续正常运行，必须定期进行检查和修正以确保系统的可靠性。

（2）经济效益：”选择性测试”是一种被广泛采用的有效方法。在进行测试时，应当遵循经济原则以确保测试的有效性和可行性。

通过对系统编码、运行流程以及历史需求的深入分析，可以有效地评估系统的性能特征。在进行软件测试时，应该遵守以下准则：

（1）为了确保软件的可靠性，我们必须将测试作为一项重要的工作，且在软件开发的需求分析、设计和开发整个阶段都需要参与，采取有效的措施来及早发现并有效地处理问题，从而进一步保障软件的可用性和稳定性。

（2）为了确保软件的正常运行，也必须对所有的测试计划和报告进行有效的整理和存储。重新设计这个系统旨在为日常维护提供便捷性。

（3）在软件测试的整个过程中，应当特别重视聚类分析问题。

（4）在软件测试中，最佳做法是避免使用自身的系统，而是采用他人开发的程序来实现测试。

（5）为了确保软件测试的准确性，我们将严格遵守相关的实施规范。

（6）全面评估测试结果，尽可能减少重复失误的发生。

## 6.4 测试结果

经过多次严格的测试，可以确保这款智慧社区系统能够为提高社区用户日常生活的便捷性也可以满足部分社区管理者的工作需求的同时系统也拥有完善的基础功能，操作简单易行，而且测试结果表明其性能卓越，因此，它非常适合普及和宣传。

# 结 论

本研究旨在深入探究微信小程序在O2O社区系统中的具体应用，以及在社区环境下的实践效果。总结出目前所存在的问题是线上线下连接能力不足、系统稳定性差、用户体验不佳等问题；提出改进建议是增加线上线下连接能力，丰富系统交互功能；增加消费者对系统的依赖性、提高用户体验度和提高用户满意度等。希望本论文针对以上提出的建议能够为下一步小程序在O2O社区系统中的实际运用提供一些参考和帮助。

尽管当前项目已基本完成，但为了更好地掌握系统设计开发的核心技术，仍需要花费更多的精力进行深入的研究、熟悉并加以巩固。通过学习，我发现了自己的一个缺点。在某些情况下，掌握必要的控制技巧并且能够理解，可以有效地利用常见的性能和控制方法，以提升整个过程的效率。经过数月的努力和分析，我们已经成功地完成了该系统的设计和调查，并且达到了预期的效果。然而，这个系统仍存在许多未完善的部分，在设计过程中也存在诸如技术漏洞等潜在问题。因此我通过专业网站和论坛的支持和协助，最终完成了智慧社区小程序的毕业设计。

通过采用SSM框架开发工具，并利用MySQL数据库来支持系统的数据交换，我们能够轻松地设计出一个高效、稳定且安全的系统，并且能够根据不同的技术要求，快速地调整数据库，从而达到最佳的开发效果。

我们在设计一个新的系统时面临着一个挑战：我们的英语水平需要进一步提升。许多涉及网络技术的信息和文献均以英语形式出现，因此我们可能无法准确掌握关键词和技术术语。通过使用翻译软件的即时翻译功能，我们可以勉强理解文本。显而易见，英语能力对于系统开发至关重要。

经过毕业设计的整个过程，我花费了巨大的精力和时间，从中也学习和巩固了很多软件开发的技术知识点，使得我收获颇多。尽管在系统开发过程中遭遇了多种技术挑战， 通过我不断的寻求解决方法最终这些问题都已得以解决。目前系统的实现仍存在许多不足之处，我将继续努力改进和完善系统，使得其可以更好的为用户提供服务。经过系统编程的学习和本次毕业设计对所学知识的使用，使我更加有信心，我坚信自己一定能在今后的工作中有所建树。

# 参考文献

1. 王晶晶,伍蕊.基于物联网的智慧社区综合管理平台设计[J].中国新技术新产品,2022(14):145-148.DOI:10.13612/j.cki.cntp.2022.14.038.
2. 孙雅婷.“互联网+”背景下的智慧社区建设研究[D].山西财经大学,2021.DOI:10.27283/d.cnki.gsxcc.2021.000217.
3. 邬思婷.“智慧式互联型”社区治理模式研究[D].上海师范大学,2022.DOI:10.27312/d.cnki.gshsu.2022.002099.
4. 朱霄逍.成都市成华区智慧社区中智能设施管理问题及对策研究[D].电子科技大学,2022.DOI:10.27005/d.cnki.gdzku.2022.000622.
5. 虞益飞.杭州高新区（滨江）智慧社区建设研究[D].西北大学,2018.
6. 秦晓鹏.基于SSH框架的互联网+社区服务系统的设计与实现[D].湖南大学,2018.
7. 朱雪峰.哈尔滨市政府推动智慧社区建设的问题研究[D].哈尔滨商业大学,2021.DOI:10.27787/d.cnki.ghrbs.2021.000369.
8. 徐丽.基于深度学习的智慧社区系统的设计与开发[D].浙江大学,2021.
9. 张巧岭.基于微信小程序的智慧社区系统的设计与实现[J].电子世界,2020(21):185-186.
10. 邓斌权,李剑波,瞿先超.基于云开发和微信小程序的智慧社区系统实现[J].电脑知识与技术,2020,16(07):82-84.
11. 周姝芸.“智慧社区”带来宜居新生活[N]. 人民日报海外版,2022-02-09(008).
12. 杨博. 基于WebGIS的智慧社区管理系统的设计与开发[D].重庆交通大学,2019.DOI:10.27671/d.cnki.gcjtc.2019.000786.
13. 刘音.智慧社区建设实现路径的影响因素分析及对策[J].居舍,2022(03):30-32.
14. 李登朝,李浩,孟俊俊.基于智慧社区的应急管理方法[J].中阿科技论坛(中英文),2022(01):76-80.
15. 郭贵芳. 基于Android的智慧社区系统的设计与实现[D].河北工业大学,2017.
16. Mathematics; Findings from Dalian University of Technology Yields New Findings on Mathematics (Real-time Fault Repair Scheme Based On Improved Genetic Algorithm)[J]. Journal of Engineering,2019.

# 致 谢

青春兵荒马乱，我们潦草收场。面对岁月摆下的宴席，我们曲终人散。初识在夏天，告别也在夏天。离別，是结束，亦是开始。

首先，我真诚地感谢我的导师在近两个月中给予我无微不至地提供了宝贵的指导与支持，他辛勤付出、无私奉献、勤勉尽职的工作精神令人感动，成为我终身学习的楷模，令我受益良多。

其次，父爱如山，母爱如海。父母是我们人生的领路人，也是我们一生中最重要的导师。父母之爱是最无私的爱，也是最伟大的爱，他们无条件地把最美好的东西都给予我们，让我们拥有最完美的未来。在这个世界上，没有任何一个人能像父母那样包容我们、教导我们。在此由衷的感谢他们让我学会了勇敢坚强、谦和善良，并给予我无尽的付出和支持。

终有一别，莫负相遇。感谢在西北大学所结识的每一位同学、老师，感谢你们对我的陪伴与照顾。若将人生比作一本书，那么大学时光就像一幅绚烂多姿的画卷，令人叹为观止; 若将人生比作一场演出，那么大学时光就像是其中最绚烂的一段。最美足相见，最难足重离，今朝此一别，何处还相遇。我们以梦为马，不辜负青春。梨花谢春红，岁月匆匆，突然成了过客，只愿总相逢。