



# 本科毕业设计

题 目: 房屋租赁管理系统设计与实现

学 院: 智能科技学院

年级专业: 2021 级计算机科学与技术 (专升本) 3 班

学生姓名: 袁天罡

学 号: 211124010635

指导教师: 卓能文

2023 年 5 月

## 学术诚信声明

本人郑重声明：所呈交的毕业论文（设计），是本人在导师的指导下，独立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，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：\_\_\_\_\_

## 版权使用授权书

本人在导师指导下所完成的毕业设计（论文）及相关的资料（包括图纸、试验记录、原始数据、实物照片、图片、录音带、设计手稿等），知识产权归属吉利学院。本人完全了解吉利学院有关保存、使用毕业设计（论文）的规定。本人授权吉利学院可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业设计（论文）。如果发表相关成果，一定征得指导教师同意，且第一署名单位为吉利学院。本人离校后使用毕业设计（论文）或与该论文直接相关的学术论文或成果时，第一署名单位仍为吉利学院。

作者签名：\_\_\_\_\_ 指导教师签名：\_\_\_\_\_

## 摘 要

随着科学技术的迅速发展，导致农村劳动力过剩，大部分人流入城市寻求生计，最终成为城市人口。这就使得城市人口流动增加，房屋租赁也成为人们关心的重中之重。目前已有的房屋租赁方式有中介和小区物业进行代挂，但是这种传统的人为管理的方式存在很多弊端。比如说房源的真假难以分辨，看房过程繁琐，甚至还存在很多中介跑路的情况。当然市面上也有一部分看房软件，但经过调查出现很多监管不到位，房源虚假，中介费高和房源少等问题<sup>[1]</sup>。

所以，笔者做了一款房屋租赁系统来试图解决人们看房的困难。一款房屋租赁系统的存在可以带来很多好处。首先，它可以让房东和租户更容易地连接起来，节省彼此的时间和精力。其次，系统可以提供一些自动化功能，如在线预订、租金支付和合同签署等，使整个租赁过程更加快捷方便。此外，系统还可以提供租户信用评分、房源信息管理等功能，有助于提高租赁市场的透明度和规范性。最后，对于平台运营商而言，这类系统也是创造盈利模式的一个途径，因为他们可以通过收取服务费或广告费等方式获得收入<sup>[2]</sup>。

该系统采用前后端分离的设计理念，前端主要采用 Vue 框架。当前 Vue 是 Javascript 使用最常用的框架，因为 Vue 可用性高，并且用法多、范围广、对界面饱满有很大作用;后端部分使用 SpringBoot 框架，SpringBoot 框架更加高效安全可靠，解决了配置复杂冗余的问题，而且还具有很多非功能特性，是作为计算机本科生必须掌握的技术;后台数据使用 MySQL 进行管理。

**关键词：** Vue， SpringBoot 框架， MySQL， 交互

## ABSTRACT

With the rapid development of science and technology, there is a surplus of labor force in rural areas. Most of them flow into cities to seek livelihoods and eventually become urban population. This makes urban population mobility increase, housing rental has become the top priority of people's concern. At present, there are existing ways of housing rental agents and residential properties, but this traditional way of artificial management has many drawbacks. For example, the real estate is difficult to distinguish between the real estate and the real estate, and there are even many intermediaries running away. Of course, there are some house-viewing software on the market, but after investigation, there are many problems such as inadequate supervision, false housing, high agency fees and few housing resources<sup>[3]</sup>.

So, I built a rental system to try to solve the problem of people looking at houses. The existence of a rental system can bring many benefits. First, it allows landlords and tenants to connect more easily, saving each other time and effort. Secondly, the system can provide some automatic functions, such as online booking, rent payment and contract signing, to make the whole leasing process faster and more convenient. In addition, the system can also provide tenants with credit scores, housing information management and other functions, helping to improve the transparency and standardization of the rental market. Finally, for platform operators, such systems are also a way to create a revenue model, as they can earn revenue by charging for services or advertising.

The system adopts the design concept of separating the front and rear ends, and the front end mainly uses the Vue framework. At present, Vue is the most commonly used framework for Javascript, because Vue has high availability, and a wide range of usage, full interface has a great role; The back-end part uses SpringBoot framework, which is more efficient, safe and reliable, solves the problem of complex and redundant configuration, and also has many non-functional features, which is a technology that must be mastered by computer undergraduates. Backend data is managed using MySQL.

**Keywords:** Vue; SpringBoot framework; MySQL; interaction

---

## 目 录

摘 要 .....	I
ABSTRACT .....	II
第 1 章 绪论 .....	1
1.1 研究目的和意义 .....	1
1.2 研究背景 .....	1
1.2.1 国内发展（应用）现状 .....	1
1.2.2 国外发展（应用）现状 .....	2
1.3 论文结构 .....	2
第 2 章 预备知识及原理说明 .....	3
2.1 MVP 设计模式 .....	3
2.2 开发语言和开发工具 .....	3
2.3 MySQL 数据库 .....	4
2.4 版本控制软件 Git .....	4
2.5 本章小结 .....	4
第 3 章 用户权限设计分析 .....	5
3.1 租客权限设计分析 .....	5
3.2 户主权限分析 .....	5
3.3 管理员权限分析 .....	6
3.4 本章小结 .....	6
第 4 章 系统整体设计 .....	7
4.1 前台设计 .....	7
4.1.1 概述 .....	7
4.1.2 设计举例 .....	7
4.2 后端设计 .....	8
4.3 后端开发可能使用的的关键类 .....	9
4.4 本章小结 .....	9
第 5 章 系统实现 .....	10
5.1 开发环境 .....	10
5.1.1 软件环境 .....	10
5.1.2 硬件环境 .....	11
5.2 前端实现 .....	11
5.2.1 界面设计 .....	11
5.2.2 功能设计 .....	12
5.2.3 动态设计 .....	13
5.3 后端实现 .....	14
5.4 后端开发的关键类 .....	17
5.4.1 数据库设计 .....	17
5.4.2 接口设计 .....	17
5.5 本章小结 .....	17
第 6 章 系统测试与运行 .....	18
6.1 测试 .....	18

6.1.1 资源测试 .....	18
6.1.2 功能测试 .....	19
6.2 运行界面 .....	19
6.3 本章小结 .....	21
第 7 章 总 结 .....	22
致 谢 .....	23
A 附录 .....	24
A.1 论文模板 .....	24
A.2 本文代码 .....	27
参考文献 .....	42

## 图形列表

图 2.1: MVP 架构图 .....	3
图 6.1: 测试流程 .....	18
图 6.2: 登录界面 .....	19
图 6.3: 注册 .....	20
图 6.4: 主页面 .....	21
图 6.5: 管理界面 .....	21

## 表格列表

表 5.1: category 表 ..... 17

## 代码列表

## 公式列表

公式 5.1: 数列求和 ..... 17

# 第 1 章 绪论

## 1.1 研究目的和意义

本课题计划完成一个全面可靠高效能实现信息透明化的房屋租赁管理系统<sup>[4]</sup>。通过完成并完善本选题，总结了本科学期期间的知识点，充分培养了动手能力，学以致用实践代码的编写。并且在解决问题的过程中，对 Java 技术有进一步的认知，提高自己的综合水平，锻炼今后的工作中遇到困难的解决钻研能力。

房屋租赁系统是一个在线平台，用于连接房东和租户，并提供一些自动化功能，使整个租赁过程更加便捷。它的出现可以带来多方面的意义，下面笔者将详细阐述。

首先，房屋租赁系统可以促进市场的透明度和规范性。在传统的租赁市场中，信息不对称、合同不规范等问题比较普遍<sup>[5]</sup>，这给租户和房东都带来了很多麻烦。而房屋租赁系统能够提供租户信用评分、房源信息管理等功能，让市场更加清晰透明。租户可以更加直观地评估房东和房源的可靠程度，从而做出更加明智的决策。房东也可以通过系统进行精准定价、优化房屋配置、提高竞争力，从而获得更好的租客来源和租金收益。

其次，房屋租赁系统可以提高租赁效率和用户体验。通过在线预订、租金支付、合同签署等自动化流程，租户和房东可以省去很多繁琐的手续和沟通过程，节约时间和精力。在租赁过程中，系统还可以提供在线客服、售后服务等功能，让用户获得更好的使用体验和感受。

第三，房屋租赁系统可以促进科技创新和数字化转型。随着信息技术的不断发展，越来越多的企业开始将传统业务与互联网结合，尝试探索新的商业模式。房屋租赁系统作为一种典型的互联网+业态，正是应用了信息技术和数字化手段，使得租赁市场变得更加智能、高效、便捷。同时，系统的开发和运营也需要涉及多种技术和人才，有助于提升整个行业的科技含量和创新能力。

最后，对于平台运营商而言，房屋租赁系统也是创造盈利模式的一个途径。通过收取服务费或广告费等方式获得收入，并通过数据分析和挖掘等手段获取更多商业机会，这些都是房屋租赁系统带来的商业价值。

总之，通过本课题的研究，目的是针对存在的房屋租赁问题进行改善，实现信息的公开透明化，全面收集数据解决房源少的问题。建立平台政策和优化监管功能，提高房屋租赁平台的综合水平，对人们的生活带来便利。

## 1.2 研究背景

### 1.2.1 国内发展（应用）现状

目前国内的城镇化战略使得人口重心移动，流入城市人口过多，城市建设发展加快。由于这些因素房价增长，房租租赁建设紧张，越来越多的房屋租赁软件崛起，但是也存在着巨大缺陷。

首先是链家成立于 2001 年，是中国领先的房地产服务企业。业务覆盖广，房源质量高，服务者素质高。上面的房源基本上是通过中介上传来进行出租，虽然优点颇多，但是需要交过高的中介费，这对刚毕业的大学生和刚在城市工作的人不太友好。

随后 2007 年安居客挤入租房行列，独有的“个人房源”选项虽然采用真实照片，但是少之甚少，大部分还是中介上传并且很多房源还是虚假房源。据调查显示，安居客很多黑中介，会泄露用户个人信息，所以这是严重的监管不到位和信息不透明的现象。

通过对十年来租房平台弊端的总结以及改善，于 2011 年成立自如租房，也是链家产业下的长租公寓品牌。一改往日的中介入驻，使用自己的管家联系户主进行拍照看房，保障房源都是真实可靠的。但盈利模式是赚取差价以及收取服务费，价格往往高于市场价。由此看来，国内的

房屋租赁系统在不断创新、提升用户体验和服务质量方面已经取得了很大的进展。随着市场竞争的加剧和技术的迭代升级，这些平台也将带来更多的变化和发展。

### 1.2.2 国外发展（应用）现状

国外的房屋租赁系统发展状况相对较为成熟已经形成了一些领先的平台和商业模式。下面举几个典型的例子。

Zillow 是一个美国的房地产信息网站，提供买卖房屋、出租房屋、房屋估价和市场分析等服务。它将各种房源信息整合到一起，在线显示给用户，方便他们进行筛选和比较。同时，它还提供比较精准的房屋估价功能，帮助房东和买家得到更好的交易结果。Zillow 目前已经成为美国房地产市场上的重要参与者之一。

Rightmove 创建于 2000 年是英国本地最大的房地产网站。房源多，信息量庞大并且还有很多自定义选项，比如说带不带家具，预算价格和选择区域。缺点是房租坐地起价。价格经常随市场变动，房源多虚拟信息也多不方便管理。

apartment.com 也创建于 2000 年。是美国目前功能完善，房源信息真实，操作方便快捷的房屋租赁网站。优点在于非常定制化满足租客需求，功能全面选择也多，房东友善与租客和睦共处。但需要走很多流程，也需要一些良好信用要求，只适合一些稳定的租客。

总之，国外的房屋租赁系统在创新商业模式、提升用户体验、拓展市场规模等方面都取得了很大的成就。未来随着技术的不断进步和市场的不断竞争，这些平台将继续探索新的发展机遇和商业模式，并为全球的租赁市场带来更多创新和变革。

## 1.3 论文结构

本文的内容主要包括以下八个部分：

第一部分是绪论，分别从国内和国外两个方面对房屋租赁管理系统设计与实现进行了背景介绍，主要说明了这次对房屋租赁管理系统开发的主要原因。

第二部分是对房屋租赁管理系统过程中需要用到的预备知识、概念原理和技术进行解释说明，在正式进行开发前需要对这些相关的理论进行系统地学习，以便理解开发过程。

第三部分是房屋租赁管理系统的权限设计，先对租客权限分析，然后对户主的权限分析以及对管理员权限的分析。

第四部分是对房屋租赁管理系统的整体设计，房屋租赁管理系统分为两个部分，前端设计模式和后端设计模式。

第五部分是系统的功能实现，首先对系统开发环境作了介绍，然后进行核心功能的实现。

第六部分是测试与优化，对游戏的运行效率进行了评估。

第七部分是总结篇，对该游戏进行了综述性的评述，并阐述了当前的不足以及将来可以进一步完善的地方。

最后列出了本文所参考的相关资料。

## 第 2 章 预备知识及原理说明

### 2.1 MVP 设计模式

MVP (Model-View-Presenter)，是一种被广泛用于开发的架构设计模式，是由 MVC 延伸出来的衍生物。作为衍生物，MVP 在整体的设计思想上继承了 MVC，其中 M (Model) 负责数据的处理，V (View) 负责界面数据的可视化和用户交互，而 C&P (Controller&Presenter) 在大体上均负责逻辑的处理。

如图 2.1 中 MVP 架构图所示，在 MVP 设计模式中，Model 的具体任务为从数据库中获取数据，并对数据加以简单处理或传递转存；View 的任务则是将经过处理的数据以合理的形式展现出来，并提供用户可操作的交互点，完成数据的可视化转换；而 Presenter，即是自 MVC 演变的核心，Presenter 主要负责将 Model 所获取传递的数据进行加工转换，随后交付 View 进行展现。在整个 MVP 设计模式中，Presenter 作为 Model 与 View 的传递桥梁，在帮助两者沟通的同时，又隔绝了两者，即 View 无法直接跳过 Presenter 对 Model 进行操作。

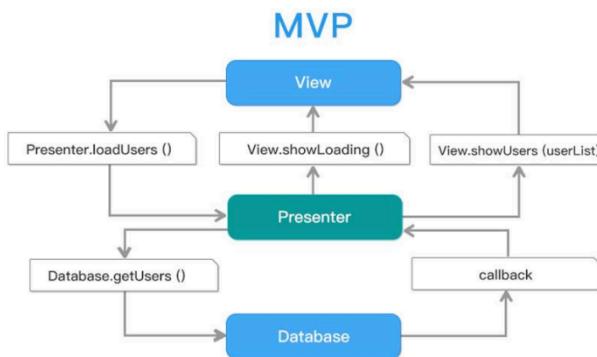


图 2.1 MVP 架构图

这样的分隔式处理，使 Model 和 View 达到完全分离，Model 不受 View 的影响，降低了模块与模块间的耦合；同时这样的设计模式也简化了 Activity 的工作任务，在 Activity 中只需要处理生命周期的任务，使代码在一定程度上更加简洁明了；由于业务逻辑被分配至 Presenter 模块中，使得 Activity 不会因为后台线程的引用而无法回收资源，有效的避免了 Activity 的内存泄漏；模块间的联系紧密程度降低，且各模块有各自明确的分工，这使得代码的层次更加清晰，提高了代码的可读性；而 Presenter 多样的具体实现让单元测试变得更加便捷。

### 2.2 开发语言和开发工具

本次研究的开发语言主要使用 Java。作为 Android 软件目前的主流开发语言，相较于 C++ 更为简单，同时又摘取了许多其它开发语言的优点特性，而面向对象的编程是 Java 语言最主要的特点之一。作为被普遍使用的面向对象的编程语言的代表，Java 不仅拥有所有面向对象编程语言都具备的普遍特性，即封装性、多态性以及继承性，它同时拥有动态联编性，更适应于面向对象的设计方法。除此之外，Java 同时具有简单性、平台独立性、多线程以及安全可靠性的诸多优点。目前市面上的大多数 Android 软件都是运用 Java 语言进行开发，相对的，由于应用了 Java 核心类的知识，使得在安卓开发中 Java 语言占据了强有力的优势。

本次研究的开发工具使用 Android Studio。Android Studio 是由谷歌推出的一款 Android 集成开发工具，它基于 IntelliJ IDEA，和 Eclipse 的安卓开发工具类似，为开发者提供了集成的 Android 开发工具。

## 2.3 MySQL 数据库

MySQL 是一种关系型数据库管理系统(RDBMS)，由瑞典 MySQL AB 公司开发，现在由 Oracle 公司拥有和维护。它是一款高效、可靠、稳定的开源数据库软件，是 LAMP (Linux+Apache+MySQL+PHP) 架构的重要组成部分，具有跨平台性和良好的兼容性。

MySQL 支持多种操作系统 (Linux、Windows、Mac OS 等) 以及多种编程语言 (C、C++、Java、Python 等)，提供了完整的 SQL 标准，包括事务处理、触发器、存储过程、视图等功能。下面是 MySQL 的一些主要特点：

**高性能：** MySQL 采用了多线程、异步 I/O 等技术，可以高效地处理大量数据请求。另外，MySQL 还支持索引、查询优化等功能，可以提高数据的访问速度。

**可靠性：** MySQL 支持数据备份、恢复等功能，可以保证数据的安全性和可靠性。此外，MySQL 还支持主从复制、集群等功能，可以实现高可用性、容错性等特点。

**易用性：** MySQL 的安装和配置比较简单，可以快速上手使用。同时，MySQL 提供了完整的文档和社区支持，用户可以获取到丰富的资源和帮助。

**开放性：** MySQL 是开源软件，可以免费使用和修改。同时，MySQL 还拥有庞大的用户社区和开发者社区，可以实现定制化开发和二次开发等需求。

**可扩展性：** MySQL 提供了完整的插件机制和 API 接口，可以方便地进行扩展和集成。开发者可以基于 MySQL 开发各种类型的应用程序，如电子商务、社交网络、游戏等。

总之，MySQL 是一款成熟、稳定、可靠、高效、易用、开放和可扩展的关系型数据库管理系统。它被广泛应用于互联网领域，支撑着许多大型网站和应用程序的数据存储和处理，如 Facebook、谷歌、雅虎、推特等。

## 2.4 版本控制软件 Git

Git 是一个开源的分布式版本控制工具，开发者们可使用 GitHub 等平台应用 Git，进行协同开发。Git 由 Linus Torvalds 开发，最初开发是为了满足团队 Linus 内核开发的需求，与常用的集中式版本控制工具不同，Git 最大的特点就是采用了分布式的模式，摒弃了集中式工具中只有中心服务器的模式，每位参与开发的使用者都持有完整版本库，有效避免了使用集中式版本控制系统的协同开发中的不稳定性，不必担心中心服务器数据丢失，同时也加快了使用者们的代码交流。在 Git 的基本工作中主要分为四个版块，图中从左至右分别为：工作区间、暂存区、本地仓库以及远程仓库。其中，工作区间为工程目录文件，主要在工作区间执行 Git 命令；暂存区为暂存上传的代码数据的区间；本地仓库用来存储本地代码，同时是本地代码与远程代码的中转站；远程仓库为中心服务平台，用于远程保存提交的代码。Git 基于这四个模块，构成了自身的基本工作流程：开发者修改工作区间的文件，执行 add 后文件将被添加到暂存区，随后 commit 提交会将文件保存至本地仓库，最终执行 push 推送，文件将会被传输到远程仓库进行保存。

## 2.5 本章小结

本章简要介绍了项目开发所需的相关知识，其中包括 MVP 设计模式的设计原理、所用的主要开发工具以及开发语言、MySQL 数据库以及版本控制器 Git 的简介。以上四个模块的储备知识将为后续项目开发提供可靠的理论支持，加强本次研究的可行性。

## 第3章 用户权限设计分析

### 3.1 租客权限设计分析

房屋租赁系统的租客权限设计是保证系统信息安全、租户合法权益及资金安全的重要环节。在设计时，需要考虑以下几个方面：

1. 注册及登录权限：租客以注册用户身份使用房屋租赁系统，需要提供基本的个人信息并完成注册流程。用户通过登录账号和密码访问系统并进行操作，登录过程需要进行身份验证。
2. 查询与浏览权限：租客可以在系统中浏览已发布的房源信息，并通过搜索等方式查询符合自己需求的房屋信息，但不允许修改任何房源信息。
3. 预订和支付权限：租客可以选择心仪的房源并提交预订请求。在预订期间，租客可以更改预订细节（如租期、价格、入住时间等），但对于一些核心数据，例如房东联系方式，租客无法直接获得。当确认预订后，租客将通过平台支付租金押金等费用。
4. 评价与投诉权限：租客有权在租期结束后评价房东、房源的质量，并且有权在必要的情况下投诉相关问题。评价和投诉信息将成为其他租客或房东参考的重要依据，因此也需要对评价和投诉信息的真实性进行保护。
5. 退款与维权权限：当租客在租期内出现合同纠纷或房源质量问题等情况时，拥有退款和维权的权利。退款和维权流程需要在平台规范的程序下操作，并需要提供相关证据以支持判断。
6. 个人信息保护：租客的个人隐私信息是需要得到网站的严格保护的。房屋租赁系统要求租客提供的信息应该仅限于必要的身份验证、预订等操作所需的必需信息。同时，房屋租赁系统也应该采取各种措施来确保用户信息的保密性。

总之，租客权限设计应该涵盖注册、浏览、预订、支付、评价、投诉、退款、维权和个人信息保护等方面，以保证租户在使用房屋租赁系统时，能够安全、便捷地完成租赁交易，并保护其合法权益。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。

### 3.2 户主权限分析

房屋租赁系统中，户主（即房东）权限设计是保障系统信息安全、保护房东权益的关键环节。在设计时，需要考虑以下方面：

1. 房源发布与编辑权限：作为房东，他们可以在房屋租赁系统上发布自己的房源信息，并对信息进行修改、更新或删除。需要注意的是，为了保证租客的安全和信任，发布的信息需要经过平台的审核才能上线。
2. 订单管理权限：房东可以查看自己的订单列表，包括预订信息、租金支付情况、租客身份信息等，并且可以接受或拒绝预订请求。此外，房东还可以在租期结束后收到租客提供的评价和投诉信息。
3. 报表分析权限：为了了解自己的房产运营状况，房东应该具有数据分析及图表功能，以便更好地了解自己的房产出租状态。通过这个功能，房东可以查看订单、收支情况、退款等相关报表，从而获得及时的反馈和监控。

4. 合同签署权限：当租客确定预订房源之后，系统会生成一份标准的租赁合同，其中包括租期、租金、押金等重要信息。房东需要在签署合同前审查和确认租期等条款，确保自己的权益得到保障。
5. 费用管理权限：房东可以在系统中设定租金、押金以及其他费用，并查看租客的付款情况和账单结算。此外，当存在租客欠费或违约时，房东还可以使用平台提供的退款和维权流程进行相应操作。
6. 房源维护权限：房东需要保持房源的良好状态，包括维修、清洁、安全等方面。在出现问题时，房东可以通过平台提供的房屋维修服务来解决问题，保证租户的居住质量。

总之，房东权限设计需要涵盖房源发布、订单管理、报表分析、合同签署、费用管理和房源维护等方面，以保证房东能够方便、快捷地完成租赁交易，并保障房东的合法权益。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。

### 3.3 管理员权限分析

房屋租赁系统的管理员权限设计是保证系统信息安全、维护平台稳定运营的重要环节。在设计时，需要考虑以下几个方面：

1. 用户管理权限：管理员需要管理所有注册用户的账号和个人信息。包括审查新用户注册申请、处理账号冻结、解锁等操作。
2. 房源管理权限：管理员需要审核并管理所有房东发布的房源信息，包括发布的内容、图片、价格、位置等信息的真实性和合规性。
3. 订单管理权限：管理员有权查看所有订单的状态、流程和相关信息，并对订单进行修改和删除。此外，管理员还能够协调和处理与订单相关的纠纷和退款问题。
4. 资金管理权限：管理员可以管理平台上所有的收支流水，包括租金支付、押金、提现等。应该对平台内的各种资金流通情况进行监控，及时发现和处理异常情况。
5. 系统维护权限：管理员有权查看平台的服务器负载、网站访问量等运营数据，以便根据实际情况优化系统。同时，管理员还需要确保系统的安全性，防范黑客攻击和信息泄露等安全事件。
6. 数据分析权限：管理员可以通过数据分析工具对平台内的各项指标进行分析和预测，以便更好地指导平台的运营策略。通过数据分析，管理员可以了解用户行为、市场趋势等信息，进而制定优化平台的策略。

总之，管理员权限设计需要涵盖用户管理、房源管理、订单管理、资金管理、系统维护和数据分析等多个方面，以保证管理员能够快速、高效地处理各种运营问题，并确保平台的稳定性和安全性。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。

### 3.4 本章小结

本章主要通过绘制了项目的用例图和功能结构图，对用例分析、用例关系进行了较详细介绍。用户权限设计包含了三个部分，并对租客权限、户主权限以及管理员权限都做了并进行简要说明，完成了项目需求分析阶段的整体设计。

## 第4章 系统整体设计

### 4.1 前台设计

#### 4.1.1 概述

房屋租赁系统的管理员权限设计是保证系统信息安全、维护平台稳定运营的重要环节。在设计时，需要考虑以下几个方面：

**用户管理权限：**管理员需要管理所有注册用户的账号和个人信息。包括审查新用户注册申请、处理账号冻结、解锁等操作。

**房源管理权限：**管理员需要审核并管理所有房东发布的房源信息，包括发布的内容、图片、价格、位置等信息的真实性和合规性。

**订单管理权限：**管理员有权查看所有订单的状态、流程和相关信息，并对订单进行修改和删除。此外，管理员还能够协调和处理与订单相关的纠纷和退款问题。

**资金管理权限：**管理员可以管理平台上所有的收支流水，包括租金支付、押金、提现等。应该对平台内的各种资金流通情况进行监控，及时发现和处理异常情况。

**系统维护权限：**管理员有权查看平台的服务器负载、网站访问量等运营数据，以便根据实际情况优化系统。同时，管理员还需要确保系统的安全性，防范黑客攻击和信息泄露等安全事件。

**数据分析权限：**管理员可以通过数据分析工具对平台内的各项指标进行分析和预测，以便更好地指导平台的运营策略。通过数据分析，管理员可以了解用户行为、市场趋势等信息，进而制定优化平台的策略。

总之，管理员权限设计需要涵盖用户管理、房源管理、订单管理、资金管理、系统维护和数据分析等多个方面，以保证管理员能够快速、高效地处理各种运营问题，并确保平台的稳定性和安全性。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。

#### 4.1.2 设计举例

##### 4.1.2.1 登录注册页面

在设计房屋租赁管理系统的登录注册功能时，需要考虑到用户体验、安全性和数据完整性等方面。以下是对登录注册页面的设想：

- 注册功能应该包括基本的个人信息，如用户名、密码、手机号码、邮箱等，并对用户输入信息的格式进行校验，确保其准确性。
- 登录功能可以使用用户名/手机号码/邮箱+密码的方式进行认证，也可以通过第三方登录（企业微信）来实现便捷登录。
- 为了保障用户的账号安全，应该设置密码强度要求，例如要求密码长度在 6-16 个字符之间，包含大小写字母、数字和特殊字符等。
- 对于忘记密码的情况，应该提供重置密码的功能，可以通过手机短信或邮箱验证来进行身份验证。
- 为了防止恶意注册和滥用系统资源，可以设置验证码功能，确保只有真实用户才能注册。
- 在用户注册成功后，可以通过邮件或短信的形式发送激活链接，确保用户账户的有效性。

- 在登录过程中可以使用 JWT Token 等技术来提高系统的安全性，避免明文传输密码等敏感信息。
- 为了方便用户管理自己的个人信息，可以在用户登录后提供个人中心页面，可供用户修改个人资料、查看订单等功能。

综上所述，登录注册是房屋租赁管理系统中的一个基础模块，合理设计并实现这个模块不仅可以提高用户体验，还能保障系统的安全性和数据完整性。

#### 4.1.2.2 增删改查

在设计房屋租赁系统的权限增删改查功能时，需要考虑到用户角色、权限类型、数据保护和操作审计等方面。以下是对增删改查的设想：

- **用户角色：**在系统中需要对不同的用户进行角色分类，例如管理员、普通用户、房东等。每个用户角色都应该有对应的权限列表，限制其对系统中数据的访问和修改。
- **权限类型：**在分配权限时，需要考虑到权限的粒度，例如可以为某个角色分配“查看房源”、“编辑房源”、“删除房源”等具体的权限。
- **数据保护：**在设计权限控制功能时，需要考虑到对数据的访问控制，确保只有拥有合法权限的人员才能访问敏感数据。例如，在实现个人信息修改功能时，可以对用户密码/手机号等敏感信息进行加密处理，并通过访问控制策略管理数据访问权限。
- **操作审计：**在系统中记录所有的权限操作，包括新增、删除和修改等，以便后续跟踪操作历史和查询问题根源。

在实现权限增删改查功能时，笔者们可以采用常用的 RBAC (Role-Based Access Control) 模型，该模型将权限授予给用户角色而非直接授予单个用户，可以有效地保证系统安全性和数据完整性。同时，可以使用框架中提供的权限管理模块来实现基础权限控制功能，例如 Spring Security、Apache Shiro 等。

综上所述，权限增删改查是房屋租赁管理系统中非常重要的一部分，需要合理设计并实现这个功能，以便保障用户数据的安全性和系统的稳定性。

## 4.2 后端设计

后端部分使用了 Java 的 SpringBoot 框架，并且采取的 MySQL 数据库来存储数据。在设计房屋租赁管理系统后端时，需要考虑到架构、技术栈、安全性和扩展性等方面。以下是一些设想部分：

- **架构：**在确定系统架构时，可以采用分层架构或微服务架构，将业务逻辑、数据访问等功能进行分离并单独部署，提高系统的可维护性和扩展性。
- **技术栈：**在选择技术栈时，可以考虑到开发效率、系统稳定性、性能和安全等方面，并选取相应地技术实现。在这次实现中将使用 SpringBoot 框架技术实现系统功能。
- **安全性：**在保障系统安全性方面，可以采用多种措施，例如对用户输入的数据进行严格验证和过滤，使用 SSL 加密协议保证数据传输的安全，使用 JWT Token 等技术防止 CSRF 等攻击。
- **扩展性：**在设计系统时，需要充分考虑到系统的可扩展性，使得系统能够满足未来的需求。例如，可以引入消息中间件，将不同的模块解耦，提高系统的可维护性和扩展性。

另外，在设计数据库时，需要考虑到数据表之间的关系，确保数据的完整性和一致性。例如，在房源表中可以添加房东 ID 字段，与房东信息表进行关联，便于查询、更新等操作。

综上所述，房屋租赁管理系统后端设计是非常重要的一部分，需要充分考虑到架构、技术栈、安全性和扩展性等方面，以实现系统的高可用性、高可扩展性和高性能。

### 4.3 后端开发可能使用的的关键类

在设计和开发房屋租赁管理系统后端时，可能会用到许多关键类，包括控制器类、服务类、DAO 类、实体类等。以下是一些后端开发过程中可能是用到的关键类：

**控制器类：**控制器类主要负责接受用户请求，调用相应的服务类完成业务逻辑的处理，并将处理结果返回给前端页面。例如，可以根据业务需求设计房源控制器类、订单控制器类等。

**服务类：**服务类主要负责业务逻辑的处理，包括对数据的读取、修改、删除等操作。例如，可以设计房源服务类、订单服务类等。

**DAO 类：**DAO (Data Access Object) 类主要负责数据访问操作，与数据库交互并提供基本的 CRUD (Create, Read, Update, Delete) 操作。例如，可以设计房源 DAO 类、订单 DAO 类等。

**实体类：**实体类代表了系统中的各种对象，包括房源、订单、用户等等。这些实体类通常包含实体属性、构造方法以及与数据库中表的映射关系，用于实现对象的持久化存储和操作。

除此之外，还可能会使用到工具类、配置类等，用于实现系统中的各种辅助功能。例如，可以使用 MyBatis 来实现数据库访问，使用 Redis 等缓存技术提高系统性能，使用 Spring Security 等安全框架来实现权限控制。

在设计关键类时，需要充分考虑到系统的业务需求和技术实现，遵循面向对象编程的原则，使得系统的代码结构简洁、可读性好、易于维护。

### 4.4 本章小结

本章通过对房屋租赁系统的分析，结合实际的工作情况，对房屋租赁系统设计的工作方面，包括前端部分和后端部分三个模块进行了介绍。

## 第 5 章 系统实现

### 5.1 开发环境

选择合适的开发工具与语言可以有效的提升开发效率。

#### 5.1.1 软件环境

##### 5.1.1.1 Visual Studio Code

Visual Studio Code，也被简称为 VS Code，是一款免费、开源、轻量级的集成开环境（IDE）。由软于 2015 年发布，现已成为一款颇受开发者喜欢的代码编辑器。Visual Studio Code 有以下的优点：

1. 语言支持：VS Code 支持数十种编程语言，包括 JavaScript、TypeScript、Python、Java 等。它能通过插件市场获取更多语言支持。
2. 调试工具：VS Code 拥有内置的试器，支持 Node、TypeScript 和 JavaScript 的调试通过插件市场，可以获取其他语言的调试器。
3. 智能代码提示：VS 支持自动补全、变量定义跟踪等功能，能够提高开发效率。
4. 版本控制：VS Code 支持 Git 和 SVN 等版本控制系统，可以直接在编辑器中进行版本管理。
5. 扩展性：Code 支持丰富插件生态，用户可以通过插件市场获取各类插件，例如风格检查、项目等。
6. 跨平台：VS Code 支持 Windows、macOS 和 Linux 操作系统。

由此看来，Studio Code 以其简洁易用、功能强大、跨平台等特点受到越来越多开发者的推崇，并成为全球范围内非受欢迎的代码编辑器。

##### 5.1.1.2 IDEA

IDEA 是 JetBrains 公司开发的一款 Java IDE，优秀的代码编辑器和调试工具。它支持多种编程语言，包括 Java、Kotlin、Groovy、Scala 等。IDEA 是基于 IntelliJ IDEA Community Edition 打的，具有轻量且易于学习的特点。

IDEA 提供了智能代码补全、重构、代码分析、版本控制以及其他实用工具，使得开发者可以更加有效地进行开发这款 IDE 还包括量的插件，例如 UI 设计器、数据库管理、Web 开发，这些插件可以帮开发人员进一的提高开发效率。

与其他 IDE 相比，IDEA 在代码提示、Refactor 等方面表现越，并且具有强大的搜索功能和快捷键设置。好的辅助功能使得 IDEA 为 Java 开发者的选 IDE 之一。

总之，作为业界标准之一的 Java 开发工具之一，IDEA 在提高编写效率、缩短开发周期等方面具有重要作用。

##### 5.1.1.3 MySQL 数据库

MySQL 是一个广泛使用的关系型数据库管理系统，开源免费。MySQL 是轻量级且易于安装部署，在开发 Web 应用非常流行。MySQL 持多种操作系统，并可以通过多种编语言进行访问。MySQL 功能丰富，提供了完整的数据管理、查询和处理功能，可以存储大型数据集并支持高并

发访问。MySQL 的安全性也得到了很好的保证，支持基于角的访问控制和、表、列级别的权限管理。

#### 5.1.1.4 Navicat

Navicat 是一款功能强大的数据库管理工具，支持多种数据库管理系统，例如 MySQL、MariaDB、Oracle、PostgreSQL 等，也包括多种操作系统，例如：Windows、macOS、Linux 等。Navicat 是一款商业软件，提供了丰富的和全面的性能优化，旨在帮开发者更加高地管理和维护数据库。

#### 5.1.2 硬件环境

操作系统: Windows 10

处理器: Intel(R) Core(TM) i5-8300H CPU

显卡: NVIDIA GeForce GTX 1060

### 5.2 前端实现

#### 5.2.1 界面设计

1. 登录界面：需要输入用户名，密码以及验证码登录。





2. 用户管理面：管理员可以查看和管理所有用户的信息，包括注册时间、联系方式等。
3. 房源管理界面：管理员可以查看和管理所有房源的，包括图片、描述价格、位置等，并能对新增、更新或删除房源信息。
4. 订单管理界面：管理员可以查看和管理所有订单的信息，包括订单状态、支付情况等，并能对订单进行审核或取消。
5. 系统设置界面：管理员可以针对系统进行相关设置，如邮件通知、短信提醒等。
6. 日志管理界面：管理员可以查看系统运行日志及操作日志。
7. 权限管理界面：管理员可以管理系统用户的权限，分配不同角色和权限。
8. 数据备份和恢复界面：管理员可以进行数据库备份和恢复操作。

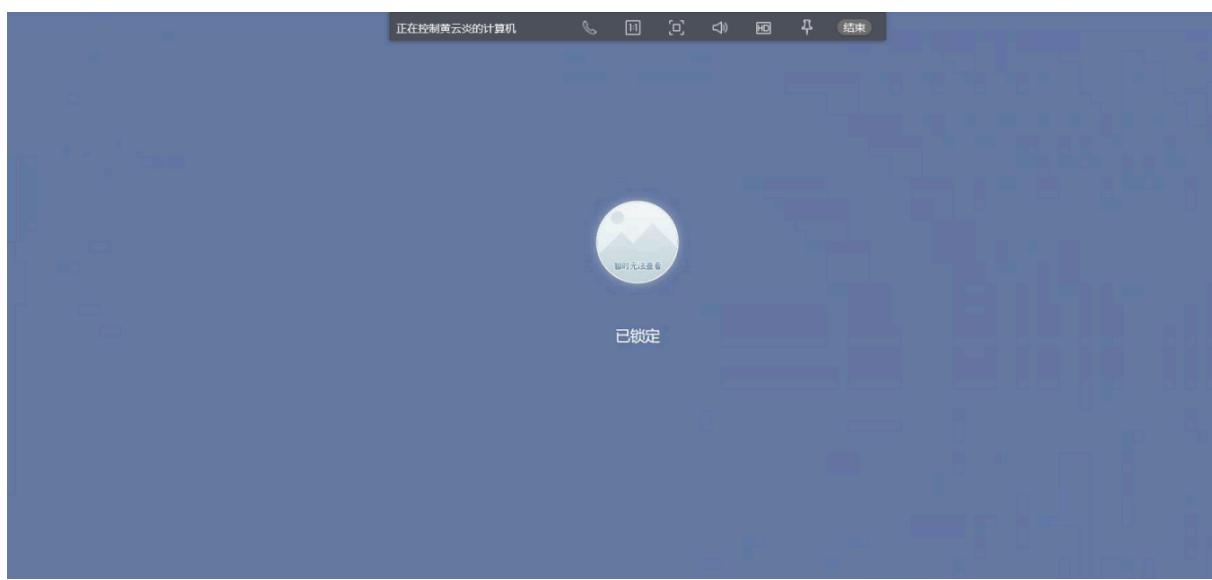
### 5.2.2 功能设计

1. 注册界面：没有登陆过的用户先要注册了之后才能登陆。需要输入手机号、姓名、密码，并通过图片验证码才能进行注册。
2. 用户管理面：管理员可以查看和管理所有用户的信息，包括注册时间、联系方式等。
3. 房源管理界面：管理员可以查看和管理所有房源的，包括图片、描述价格、位置等，并能对新增、更新或删除房源信息。

4. 订单管理界面：管理员可以查看和管理所有订单的信息，包括订单状态、支付情况等，并能对订单进行审核或取消。
5. 系统设置界面：管理员可以针对系统进行相关设置，如邮件通知、短信提醒等。
6. 日志管理界面：管理员可以查看系统运行日志及操作日志。
7. 权限管理界面：管理员可以管理系统用户的权限，分配不同角色和权限。
8. 数据备份和恢复界面：管理员可以进行数据库备份和恢复操作。

### 5.2.3 动态设计

1. 响应式设计：现在越来越多的人使用手机和平板电脑上网，因此，一个好的房屋租赁管理系统必须具备响应式设计，以适应不同设备的屏幕尺寸和分辨率。通过这种方式，用户可以随时随地访问系统而不必担心屏幕尺寸或布局问题。
2. 数据可视化：一个好的房屋租赁管理系统应该具有良好的数据可视化功能，以帮助用户更好地了解租赁业务的状况。例如，可以使用柱状图或折线图等方式展示关于租户数量、租金收入和支出等关键指标的数据。这样，用户可以通过数据可视化快速评估其租赁业务的运营状况。
3. 自动化流程：一个好的房屋租赁管理系统还应该具备自动化的流程，以提高用户的工作效率。例如，当一份合同到期时，系统应该能够自动提醒用户需要更新合同，而不必手动查找过期合同。这种自动化流程可以帮助用户大大减少操作时间和精力。
4. 安全性：一个好的房屋租赁管理系统必须具备安全性。租赁管理系统通常包含很多敏感数据，例如租客的个人信息和账单信息等，因此必须确保这些数据不会被未经授权的第三方访问或泄露。系统应该采用加密方法来保护存储在数据库中的信息，同时还应该设定权限系统，以限制每个用户可以访问的数据量和范围。

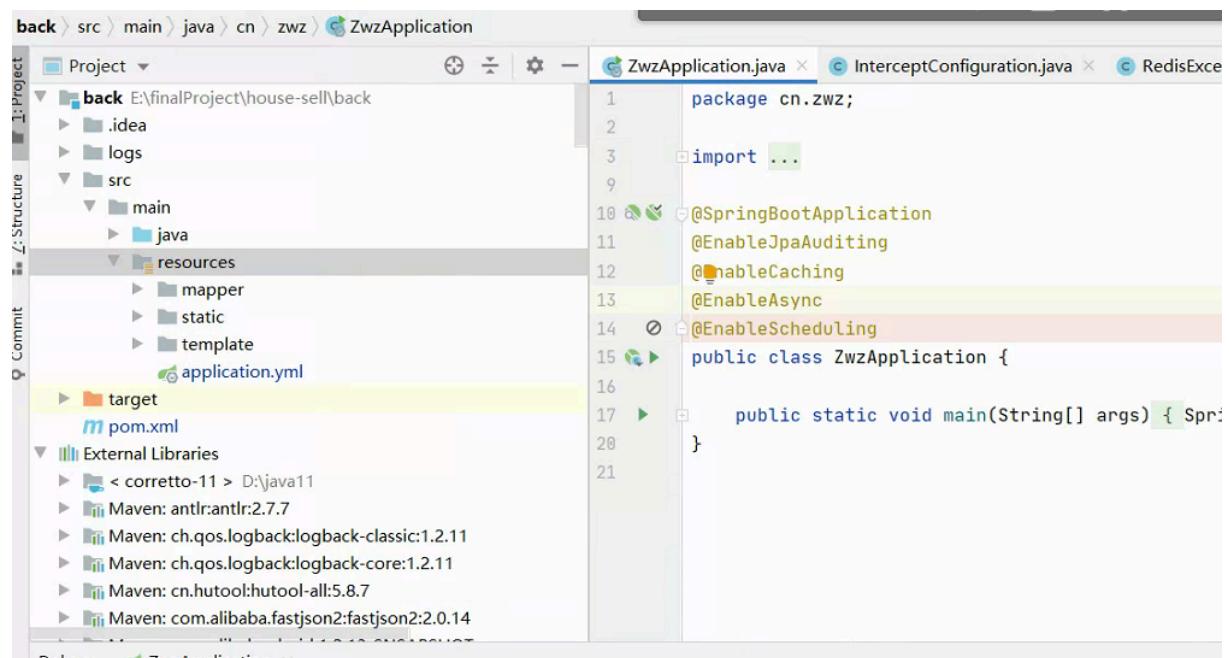


总之，房屋租赁管理系统是一个非常重要的应用程序，可以帮助房地产公司、物业管理公司和个人房东等有效管理其租赁业务。通过考虑响应式设计、多语言支持、数据可视化、自动化流程以及安全性等关键因素，可以确保该系统具有良好的用户体验，并能够在市场上获得成功。

### 5.3 后端实现

后端部分使用了 Java 的 SpringBoot 框架，并且采取的 MySQL 数据库来存储数据。大体步骤如下：

- 先搭建一个 SpringBoot 项目，使用了 IDEA 开发工具来进行后端开发。
- 确定系统架构和数据库结构，设计 ER 图创建相应的表格
- 在 pom.xml 文件中添加必的依赖，包括 Boot 和 MySQL 驱动。
- 编实体类（Entity）和 DAO 层（Data Access Object）对应的 Repository 接口，以及对应的 SQL 语句。
- 创建服务层（Service）并实现事务管理。
- 配置数据库连接池和相关的参数，包括数据库 url、用户名和密码等。
- 编写控制层（Controller）及相应的 API 接，提供相应的增删改查功能。
- 进行单元测试和集成测试，保证功能正常。
- 部署项目到服务器上，让用户能够直接使用并进行后续维护。



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under 'back'. It includes a 'src' folder containing 'main' (with 'java' and 'resources' subfolders), 'target', 'pom.xml', and 'External Libraries'. The 'resources' folder contains 'mapper', 'static', 'template', and 'application.yml'.
- Code Editor:** Displays the file 'ZwzApplication.java' with the following code:

```

1 package cn.zwz;
2
3 import ...
4
5 @SpringBootApplication
6 @EnableJpaAuditing
7 @EnableCaching
8 @EnableAsync
9
10 @EnableScheduling
11
12 public class ZwzApplication {
13
14     public static void main(String[] args) { Spr
15 }
16
17 }
```
- Toolbars and Status Bar:** Standard IntelliJ IDEA toolbars and status bar at the bottom.

Project tree:

- cn.zwz
  - basics
  - baseClass
  - baseVo
  - code
  - exception
  - lock
  - log
  - mybatisplus
  - parameter
  - redis
  - security
  - utils
  - data
  - house
  - test
- ZwzApplication
- resources
- target
- pom.xml

Code (ZwzApplication.java):

```

1 package cn.zwz;
2
3 import ...
4
5 @SpringBootApplication
6 @EnableJpaAuditing
7 @EnableCaching
8 @EnableAsync
9 @EnableScheduling
10 public class ZwzApplication {
11
12     public static void main(String[] args) { Spring
13 }
14
15 }
```

Project tree:

- baseClass
- baseVo
- code
- exception
- lock
- log
- mybatisplus
- parameter
- redis
- security
- utils
- data
  - controller
  - CaptchaController
  - DepartmentController
  - DictController
  - DictDataController
  - FileController
  - IpInfoController
  - LogController
  - MyDoorController
  - MyUserController
  - PermissionController
  - RedisController
  - RoleController
  - SecurityController
  - SettingController
  - UploadController
  - UserController

Code (CaptchaController.java):

```

1 @Transactional
2 public class CaptchaController {
3
4     @Autowired
5     private StringRedisTemplate redisTemplate;
6
7     @RequestMapping(value = "/draw/{captchaId}", method = RequestMethod.GET)
8     @ApiOperation(value = "根据验证码ID获取图片")
9     public void draw(@PathVariable("captchaId") String captchaId, HttpServletResponse response) throws IOException {
10         String codeStr = redisTemplate.opsForValue().get(captchaId);
11         CreateVerifyCode createVerifyCode = new CreateVerifyCode( imageWidth: 116, imageHeight: 36, codeCount: 4, lineCount: 10, codeSize: 16 );
12         CreateVerifyCode.createVerifyCode( codeStr, response.getOutputStream() );
13         response.setContentType("image/png");
14         createVerifyCode.write(response.getOutputStream());
15     }
16
17     @RequestMapping(value = "/init", method = RequestMethod.GET)
18     @ApiOperation(value = "初始化验证码")
19     public Result<Object> init() {
20         String codeId = UUID.randomUUID().toString().replace( target: "-", replacement: "" );
21         redisTemplate.opsForValue().set(codeId, new CreateVerifyCode().randomStr( size: 4 ), timeout: 2L, TimeUnit.MILLISECONDS );
22         return ResultUtil.data(codeId);
23     }
24 }
```

Project tree:

- data
  - controller
    - CaptchaController
    - DepartmentController
    - DictController
    - DictDataController
    - FileController
    - IpInfoController
    - LogController
    - MyDoorController
    - MyUserController
    - PermissionController
    - RedisController
    - RoleController
    - SecurityController
    - SettingController
    - UploadController
    - UserController
  - dao
  - entity
  - service
  - serviceimpl
  - utils
  - vo
  - house
  - test
  - ZwzApplication- resources
- target

Code (RedisController.java):

```

1 @Controller
2 public class RedisController {
3
4     @Autowired
5     private StringRedisTemplate redisTemplate;
6
7     @RequestMapping(value = "/set", method = RequestMethod.POST)
8     @ApiOperation(value = "设置键值对")
9     public Result<Object> set(@RequestParam("key") String key, @RequestParam("value") String value, @RequestParam("time") Long time, @RequestParam("timeUnit") TimeUnit timeUnit) {
10         redisTemplate.opsForValue().set(key, value, time, timeUnit);
11         return ResultUtil.success();
12     }
13
14     @RequestMapping(value = "/delByKeys", method = RequestMethod.POST)
15     @ApiOperation(value = "删除")
16     public Result<Object> delByKeys(@RequestParam("keys") String[] keys) {
17         for (String redisKey : keys) {
18             redisTemplate.delete(redisKey);
19         }
20         return ResultUtil.success();
21     }
22
23     @RequestMapping(value = "/delAll", method = RequestMethod.POST)
24     @ApiOperation(value = "全部删除")
25     public Result<Object> delAll() {
26         redisTemplate.delete(redisTemplateHelper.keys(STEP_STR_IN_REDIS));
27         return ResultUtil.success();
28     }
29
30     @RequestMapping(value = "/getKeySize", method = RequestMethod.GET)
31     @ApiOperation(value = "获取实时key大小")
32     public Result<Object> getKeySize() {
33         Map<String, Object> map = new HashMap<>(INIT_SIZE_IN_REDIS);
34         map.put("keySize", redisTemplate.getConnectionFactory().getConnection().dbSize());
35         map.put("time", DateUtil.format(new Date(), DATE_FORMAT_IN_REDIS));
36         return ResultUtil.data(map);
37     }
38 }
```

Project View (Top):

```

public class User extends ZzwBaseEntity {
    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "姓名")
    @NotNull(message = "姓名不能为空")
    @Size(max = 28, message = "姓名长度不能超过20")
    private String nickname;

    @ApiModelProperty(value = "账号")
    @Column(unique = true, nullable = false)
    @Pattern(regexp = "^[a-zA-Z0-9_\\u4e00-\\u9fa5]{4,16}$", message = "账号长度不合法")
    private String username;

    @ApiModelProperty(value = "密码")
    @NotNull(message = "密码不能为空")
    private String password;

    @ApiModelProperty(value = "密码强度")
    @Column(length = 2)
    private String passStrength;

    @ApiModelProperty(value = "手机号")
    @Pattern(regexp = "^[1][3,4,5,6,7,8,9][0-9]{9}$", message = "手机号格式错误")
    private String mobile;

    @ApiModelProperty(value = "部门ID")
    private String departmentId;
}

```

Explorer View (Bottom Left):

- FRONT
  - router
  - store
  - modules
    - index.js
  - styles
  - views
    - code
    - demo
    - dict
    - file
    - home
    - house
    - log
    - main-components
    - menu
    - password
    - role
    - roster
    - template
    - login.vue
  - main.less
  - Main.vue
  - register.vue
  - App.vue
- OUTLINE

Terminal View (Bottom Right):

```

node + 20:33:17
2019-06-19 20:33:17 [INFO] Compiled successfully in 16916ms

```

Code Editor View (Bottom Center):

```

<template>
<div class="own-space">
    <Card>
        <Divider dashed>
            个人门户
            <Button type="success" v-show="!editFlag" @click="editFlag = true">开始编辑</Button>
            <Button type="warning" v-show="editFlag" @click="saveEdit">保存提交</Button>
        </Divider>
        <Form ref="userForm" :model="userForm" :label-width="90" label-position="left">
            <Row :gutter="16">
                <Col :span="12">
                    <FormItem label="登录账号" prop="username">
                        <Input v-model="userForm.username" readonly style="width: 100%" />
                    </FormItem>
                </Col>
                <Col :span="12">
                    <FormItem label="姓名" prop="nickname">
                        <Input v-model="userForm.nickname" readonly style="width: 100%" />
                    </FormItem>
                </Col>
            </Row>
            <Row :gutter="16">
                <Col :span="12">
                    <FormItem label="账号类型" prop="typeTxt">

```

## 5.4 后端开发的关键类

在使用 Spring Boot 框架中实现功能有许多关键类，下面将介绍部分关键类。

1. `@RestController`: 注解表示该类是一个 RESTful API 的控制器。
2. `@RequestMapping`: 该注解表示该方法对应的 URL 请求路径。
3. `@RequestBody`: 该注解表示该参数是请求体中的数据。
4. `@Autowired`: 该注解表示该属性需要自动注入某个对象。
5. `JpaRepository`: 该接口提供了很多通用的数据库操作方法
6. `@Entity`: 该注解表示该类应数据库中的一张表。
7. `@Table`: 该注解表示该类对应数据库中的一张表，来指定表的名称和其它属性。
8. `@GeneratedValue`: 该注解表示该属性的值数据库自动生成。
9. `@Id`: 该注解表示该属性是实体类的主键。
10. `@Column`: 该注解表示该属性对应数据库中的一列，用来指定字段的名称和其它属性。
11. `@`: 该注解表示方法执行时需要开启一个事务。
12. `@Modifying`: 该注解表示该方法执行时需要修改数据库中的数据。

### 5.4.1 数据库设计

表 5.1 category 表

字段名称	类型	长度	是否 null	主键	字段说明
cid	int	5	是	是	
cname	varchar	10	否		

### 5.4.2 接口设计

勾股定理可用公式： $a^2 + b^2 = c^2$  表示。

公式 5.1 数列求和

$$\sum_{k=1}^n k = \frac{n(n+1)}{2} \quad (5.1)$$

## 5.5 本章小结

本章结束了房屋租赁管理系统的所需的硬件环境。对前端页面和后端的具体实现进行了介绍。同时，对后端关键类的实现方法进行了阐述。

## 第 6 章 系统测试与运行

### 6.1 测试

软件测试通常是指验证与确认两部分。该系统的主要进行了以下几个方面的测试：资源测试、功能测试、任务测试。测试的流程图 6.1 所示。

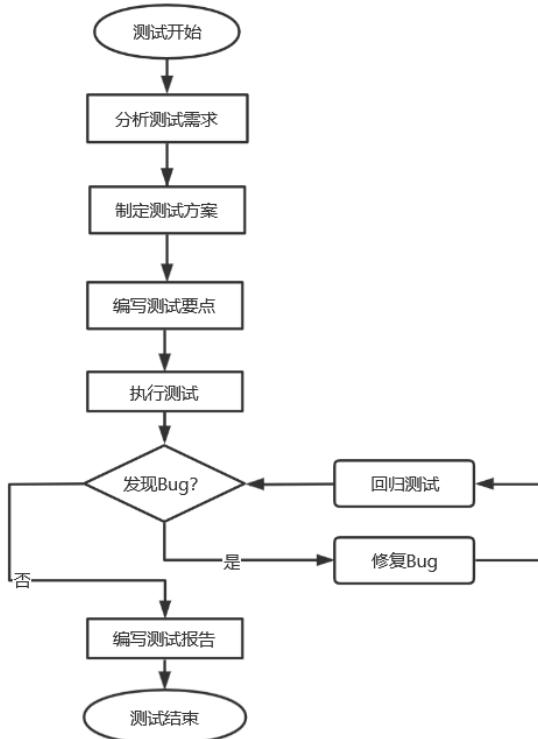


图 6.1 测试流程

#### 6.1.1 资源测试

房屋租赁管理系统的资源测试是对系统进行性能，以确保系统在高负载、大流量情况下仍然能够稳定运行。下面介绍具体的测试内容和流程：

1. 负载测试：这种测试通过模拟用户并发访问系统，同时观察系统响应时间、吞吐量等指标来评估系统在承受并发请求时的能力。测试时需要针对不同的业务场景（例如搜索房源、订单、支付等）进行测试，并分别记录响应时间、请求成功率、错误码等指标。
2. 压力测试：将系统负载逐渐增加，直到达到系统极限，观察系统在极端情况下的表现。测试阶段需要设定并观察系统响应时间、CPU 利用率、内存使用量、带宽利用率等关键指标，以保证系统在硬件资源撑不住的情况下和内部逻辑出现瓶颈时能保持应有的能力和稳定性。
3. 并发测试：模拟多个用户并发访问同一个页面或接口，以检测系统在高并发情况的表现。并发测试数据需要根据业务特点精确制定，去模拟正常和异常下的并发请求比如打开一个页面，提交同一个表单，支付多少订单，同时踢出多少用户等。
4. 数据量测试：将大数据导入系统中进行测试，以检验系统的性能和稳定性。测试过程中需要特别关注系统在数据查询、更新、删除等操作时表现，同时还需要关注内存和 CPU 利用率的变化趋势；
5. 日志跟踪测试：通过记录系统日志并进行分析，可以追踪系统在某个时间段或者某个业务场景下执行的情况。这种测试可以发现系统运行时出现的异常、慢查询和错误。

经过以上测试，可以获取到系统各方面的性能参数与指标从而进行较好的化和调整。

### 6.1.2 功能测试

房屋租赁管理系统的功能测试是对系统进行项功能方面的测试，以保证系统满足需求并能正常运行。下面具体是测试内容的流程：

1. 用户管理功能测试：测试系统能否正确地实现用户账号注册、，个人信息修改和找回等功能，并检查这些功能的安全性和稳定性。
2. 房源管理功能测试：测试系统能否支持房源的添加、修改、查询和删除等操作，并能够准确地显示房源信息，同时测试系统搜索功能是否准确返回相应搜索结果。
3. 租户管理功能测试：测试系统能否支持租户信息注册、修改和删除等操作，并能准确地记录租房合同信息，租金缴纳情况和租房状态信息。
4. 订单管理功能测试：测试系统能否自动生成订单且订单信息的准确性，订单状态处理的准确性。
5. 支付功能测试：测试系统是否支持多种支付渠道，能够安全地处理付款信息，管理退款信息。
6. 系统管理功能测试：测试系统管理功能包括系统日志记录、数据备份和恢复，网站 SEO、营销推广等部分，以及其他额外功能如系统主题切换、语言转等。

经过以上测试笔者们可以确认系统各项功能的完整性和质量，确保系统能够足用户的需求并能够稳定运行。

## 6.2 运行界面

经过测试与修改以后，系统就可以正常运行了。图 6.2 显示用户进入系统的登录界面。



图 6.2 登录界面

图 6.3 表示注册界面



图 6.3 注册

图 6.4 表示用户的主页面



图 6.4 主页面

图 6.5 表示用户管理界面

编号	用户名	登录账号	头像	所属部门	手机	邮箱	操作
1	用户4	用户4		行政综合部	17896525487	user4@qq.com	<a href="#">编辑</a> <a href="#">重置密码</a> <a href="#">禁用</a> <a href="#">删除</a>
2	测试卖家	17859654121		综合发展部	17859654121	17859654121@qq.co m	<a href="#">编辑</a> <a href="#">重置密码</a> <a href="#">禁用</a> <a href="#">删除</a>
3	测试买家	17859654125		行政综合部	17859654125	17859654125@qq.co m	<a href="#">编辑</a> <a href="#">重置密码</a> <a href="#">禁用</a> <a href="#">删除</a>
4	管理员	管理		设计研发部	17857054388	916077357@qq.com	<a href="#">编辑</a> <a href="#">重置密码</a> <a href="#">禁用</a> <a href="#">删除</a>

图 6.5 管理界面

### 6.3 本章小结

本章主要介绍了系统测试以及最终系统运行结果。首先介绍了这个测试流程，包括资源测试和功能测试。通过系统实际运行页面截图，辅以文字描述对系统的最终运行结果进行了效果展示说明。

## 第 7 章 总 结

本篇论文主要介绍了房屋租赁管理的系统与实现，通过对系统的详细描述和体实现，展现了其在房屋租赁管理中的重要作用。

首先介绍了课题的背景和研究意义，指出了房屋租赁管理系统的必要性和重性。接下来，本文分析了当前房屋租赁行业的现状和问题，并提出了本文设计的系统解决问题的方法。房屋租赁管理设计和实现基于 Java，使用 SpringBoot 开发框架进行开发，并采用 MySQL 数据库进行数据管理。通过对用户需求的分析和系统功能的划分，本文实现了系统的基础功能和高级功能，包括房屋信息管理、租客信息管理等核心模块。

在具体实现中，采用了 MVC 模式，将系统的业务逻辑、视图呈现和数据处理相互分离，从而提高了系统的可重用性、可扩展性和可维护性。

最后通过对本文设计和实现的总结，可以得出如下结论：

1. 房屋租赁管理系统的有效设计和实现能够提高房屋租赁行业的管理效率和服务质量，满足用户需求，为用户带来更好的服务体验。
2. 本文所采用的设计方法和技术手段是有效的，被应用于类似的管理系统的开发中，并具一定的普适性。
3. 本文的研究仍然存在一些不足之处，例如对用户需求的分析不够深入，系统安全性的处理还完善等，这些问题需要在今后的研究中得到进一步的完善和改进。

综上所述，本文的研究为房屋租赁管理系统的有效设计和实现提供了一定的参考和借鉴值，也为相关领域的研究者提供了一点微薄之力。

## 致 谢

在笔者的本科生涯即将结束之际，笔者不禁回首往事，深感时光如梭。这些年来有许多人和事让笔者受益匪浅，使得笔者度过了充实而难忘的四年。在这篇毕业论文致谢中，笔者要对他们表达真情实感的感激之意。

首先，笔者要感谢笔者的父母，在笔者茁壮成长的过程给予了笔者无尽的关爱和支持。他们默默无闻地为笔者付出，从小到大笔者一直被他们宠了。他们为了笔者能够顺利完成学业，所有的膳食语出等供应都是最好的，虽然这些都可能已成为家长义务但是，对于笔者感恩永存。

其次笔者要感谢笔者的导师卓能文老师，他是一个温暖、善良且富有智慧的人。他为笔者们的实项目指明了前进方向，耐心细致地给予指导和帮助。在研究过程中遇到困难和障碍时，他会及时指出问题所在并提出建设性意见。虽然笔者只是一名徒弟，但他从始至终给笔者的关怀和帮助让笔者受益匪浅。

再次，笔者要感谢笔者的同学们，你们是笔者路走来的陪伴者。在学习中若遇到困难，你们会及时贡献自己的智慧和时间来帮助笔者解决问题。在生活每次共同的经历也能让笔者感到非常快乐。与你们在一起的时光不仅是笔者难忘的回忆，也是让笔者成长的催化剂。笔者将永远怀念笔者们一起度过的岁月。

此外，笔者还要感谢吉利学院为笔者提供的良好学习和交流环境。无论从课程设置、图书馆资源和学术研究都给了笔者很大的帮助。在这里，笔者收获了知识和思维方式，找到了未职业发展的方向。

最后，笔者还要感谢笔者的朋友们，谢谢你们的建议和支持。你们用你们的话语温暖笔者，在笔者孤独时在笔者身边陪伴，让笔者感到自己是个幸运的人。

总之，笔者的日记本永远记录着与你们共进退的日子，未来笔者们无论在世界的哪个角落，未来笔者们依然并行不悔。上天在笔者人生路上给予笔者许多的善意希望笔者们能够再次相遇在人生旅途的某处。感谢你们，祝笔者们回首枯藤老树时，仍是少年郎！

## A 附录

您可在 <https://github.com/soarowl/geelypaper.git> 检查最新代码，或提 PR。

### A.1 论文模板

```

1  #import "@preview/i-figured:0.2.3"
2  #import "@preview/sourcerer:0.2.1": code
3  #import "@preview/tbl:0.0.4"
4
5  #let paper(
6    title: "",
7    faculty: "",
8    class: "",
9    author: "",
10   studentnumber: "",
11   adviser: "",
12   date: datetime.today().display(),
13   cnabstract: [],
14   cnkeywords: (),
15   enabstract: [],
16   enkeywords: (),
17   body,
18 ) = {
19   set document(title: title, author: author, keywords: cnkeywords.join(", ") +
enkeywords.join("; "))
20   set text(font: ("Times New Roman", "SimSun"), lang: "zh")
21
22 //***** 图形、代码及表格列表设置
23 // this `level: 2` instructs the figure counters to be reset for every
24 // level 2 section, so at every level 1 and level 2 heading.
25 show heading: i-figured.reset-counters.with(level: 1)
26
27 // 设置标题为黑体
28 show heading: it => {
29   text(font: ("Times New Roman", "SimHei"), it)
30 }
31
32 // 每章分页，标题居中
33 show heading.where(level: 1): it => {
34   pagebreak(weak: true)
35   align(center, it)
36 }
37
38 // this `level: 2` instructs the figure numbering to include the first
39 // two levels of the current heading numbering.
40 // how this should behave with zeros can be set using `zero-fill`.
41 // e.g., setting `zero-fill: false` and `leading-zero: false` assures
42 // there is never a `0` in the numbering.
43 show figure: i-figured.show-figure.with(level: 1)
44 // master 版本不能编译
45 show math.equation: i-figured.show-equation
46
47 // set figure(numbering: "1-1") // don't work, maybe a typst bug
48 set figure.caption(position: top, separator: [#h(1em)])
49 show figure.where(kind: image): set figure.caption(position: bottom)
50 //*****
51

```

```

52 //***** 代码框设置
53 show raw.where(block: true): it => {
54   code(it)
55 }
56 //*****
57
58 //***** 表格设置
59 show: tbl.template.with(box: false, breakable: true, tab: " | ")
60 //*****
61
62 //***** 标题页设置
63 v(5fr)
64 align(center, image("logo.png", width: 50%))
65 v(10pt)
66 set align(center)
67 text(3em, "本科毕业设计")
68
69 v(10fr, weak: true)
70 let hline() = [#v(-0.7em)#line(length: 20em)]
71 table(
72   columns: (20%, auto),
73   stroke: none,
74   [题 #h(2em)目: ],
75   [#title#hline()],
76   [学 #h(2em)院: ],
77   [#faculty#hline()],
78   [年级专业: ],
79   [#class#hline()],
80   [学生姓名: ],
81   [#author#hline()],
82   [学 #h(2em)号: ],
83   [#studentnumber#hline()],
84   [指导教师: ],
85   [#adviser#hline()])
86 )
87
88 v(10fr, weak: true)
89 text(1.1em, date)
90
91 set align(left)
92 pagebreak()
93 //*****
94
95 set par(first-line-indent: 2em, justify: true)
96 show par: set block(spacing: 0.65em)
97 // Workaround 3: Automatically add empty paragraph after heading
98 show heading: it => {
99   it
100  par(text(size: 0.35em, h(0.0em)))
101 } // Only works for paragraphs directly after heading
102
103 //***** 版权页
104 align(center, text(1.5em, font: "SimHei", [学术诚信声明]))
105 [本人郑重声明: 所呈交的毕业论文(设计), 是本人在导师的指导下, 独立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, 本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体, 均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。]
106 table(

```

```

107  columns: (1fr, 1fr, 1fr, 1fr),
108  stroke: none,
109  [], [], [#align(right, [作者签名: ])], [#v(1em)#line(length: 100%)]
110 )
111
112 v(2em)
113 align(center, text(1.5em, font: "SimHei", [版权使用授权书]))
114 [本人在导师指导下所完成的毕业设计（论文）及相关的资料（包括图纸、试验记录、原始数据、实物照片、图片、录音带、设计手稿等），知识产权归属吉利学院。本人完全了解吉利学院有关保存、使用毕业设计（论文）的规定。本人授权吉利学院可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业设计（论文）。如果发表相关成果，一定征得指导教师同意，且第一署名单位为吉利学院。本人离校后使用毕业设计（论文）或与该论文直接相关的学术论文或成果时，第一署名单位仍为吉利学院。]
115 table(
116   columns: (1fr, 1fr, 1fr, 1fr),
117   stroke: none,
118   [#align(right, [作者签名: ])], [#v(1em)#line(length: 100%)],
119   [#align(right, [指导教师签名: ])], [#v(1em)#line(length: 100%)]
120 )
121
122 pagebreak()
123 //*****
124
125 //***** 页眉、页脚
126 set page(
127   header: [#h(1fr)吉利学院本科毕业设计 #h(1fr)#line(length: 100%, stroke: 2pt)],
128   number-align: center,
129 )
130 //*****
131
132 set heading(numbering: none)
133 set page(numbering: "I")
134 counter(page).update(1)
135
136 //***** 中文摘要
137 heading([摘 #h(1em)要])
138 cnabstract
139 v(1em)
140 let cn = cnkeywords.join(", ")
141 par(first-line-indent: 0em)[
142   #text(font: "SimHei", [关键词: ])
143   #cn
144 ]
145 pagebreak()
146 //*****
147
148 //***** 英文摘要
149 heading([ABSTRACT])
150 enabstract
151 v(1em)
152 let en = enkeywords.join("; ")
153 par(first-line-indent: 0em)[
154   #text([*Keywords:* ])
155   #en
156 ]
157 pagebreak()
158 //*****
159

```

```

160 //***** 目录
161 set par(first-line-indent: 0em, justify: true)
162 outline(title: [目 #h(2em)录], indent: true, depth: 3)
163 i-figured.outline(title: [图形列表])
164 i-figured.outline(target-kind: table, title: [表格列表])
165 i-figured.outline(target-kind: raw, title: [代码列表])
166 // master 版本不能编译
167 i-figured.outline(target-kind: math.equation, title: [公式列表])
168 pagebreak()
169 //*****
170
171 //***** 正文
172 set heading(
173   numbering: (..nums) => {
174     let vals = nums.pos()
175     if vals.len() == 1 {
176       let value = str(vals.at(0))
177       return "第" + value + "章"
178     }
179     else {
180       return nums.pos().map(str).join(".")
181     }
182   }
183 )
184
185 set page(numbering: "1")
186 set par(first-line-indent: 2em, justify: true)
187 counter(page).update(1)
188
189 body
190 //*****
191 }
192

```

## A.2 本文代码

```

1 #import "paper.typ":* typst
2
3 #let cnabstract = [随着科学技术的迅速发展，导致农村劳动力过剩，大部分人流入城市寻求生计，最终成为城市人口。这就使得城市人口流动增加，房屋租赁也成为人们关心的重中之重。目前已有的房屋租赁方式有中介和小区物业进行代挂，但是这种传统的人为管理的方式存在很多弊端。比如说房源的真假难以分辨，看房过程繁琐，甚至还存在很多中介跑路的情况。当然市面上也有一部分看房软件，但经过调查出现很多监管不到位，房源虚假，中介费高和房源少等问题 @barb。
4
5 所以，笔者做了一款房屋租赁系统来试图解决人们看房的困难。一款房屋租赁系统的存在可以带来很多好处。首先，它可以让房东和租户更容易地连接起来，节省彼此的时间和精力。其次，系统可以提供一些自动化功能，如在线预订、租金支付和合同签署等，使整个租赁过程更加快捷方便。此外，系统还可以提供租户信用评分、房源信息管理等功能，有助于提高租赁市场的透明度和规范性。最后，对于平台运营商而言，这类系统也是创造盈利模式的一个途径，因为他们可以通过收取服务费或广告费等方式获得收入 @camb。
6
7 该系统采用前后端分离的设计理念，前端主要采用 Vue 框架。当前 Vue 是 Javascript 使用最常用的框架，因为 Vue 可用性高，并且用法多、范围广、对界面饱满有很大作用；后端部分使用 SpringBoot 框架，SpringBoot 框架更加高效安全可靠，解决了配置复杂冗余的问题，而且还具有很多非功能特性，是作为计算机本科生必须掌握的技术；后台数据使用 MySQL 进行管理。]
8

```

```

9 #let enabstract = [With the rapid development of science and technology, there
is a surplus of labor force in rural areas. Most of them flow into cities to
seek livelihoods and eventually become urban population. This makes urban
population mobility increase, housing rental has become the top priority of
people's concern. At present, there are existing ways of housing rental agents
and residential properties, but this traditional way of artificial management
has many drawbacks. For example, the real estate is difficult to distinguish
between the real estate and the real estate, and there are even many
intermediaries running away. Of course, there are some house-viewing software
on the market, but after investigation, there are many problems such as
inadequate supervision, false housing, high agency fees and few housing
resources @donne.
10
11 So, I built a rental system to try to solve the problem of people looking at
houses. The existence of a rental system can bring many benefits. First, it
allows landlords and tenants to connect more easily, saving each other time and
effort. Secondly, the system can provide some automatic functions, such as
online booking, rent payment and contract signing, to make the whole leasing
process faster and more convenient. In addition, the system can also provide
tenants with credit scores, housing information management and other functions,
helping to improve the transparency and standardization of the rental market.
Finally, for platform operators, such systems are also a way to create a
revenue model, as they can earn revenue by charging for services or
advertising.
12
13 The system adopts the design concept of separating the front and rear ends, and
the front end mainly uses the Vue framework. At present, Vue is the most
commonly used framework for Javascript, because Vue has high availability, and
a wide range of usage, full interface has a great role; The back-end part uses
SpringBoot framework, which is more efficient, safe and reliable, solves the
problem of complex and redundant configuration, and also has many non-
functional features, which is a technology that must be mastered by computer
undergraduates. Backend data is managed using MySQL.]
14
15 #show: paper.with(
16   title: "房屋租赁管理系统设计与实现",
17   faculty: "智能科技学院",
18   class: "2021 级计算机科学与技术 (专升本) 3 班",
19   author: "袁天罡",
20   studentnumber: "211124010635",
21   adviser: "卓能文",
22   date: "2023 年 5 月",
23   cnabstract: cnabstract,
24   cnkeywords: ("Vue", "SpringBoot 框架", "MySQL", "交互"),
25   enabstract: enabstract,
26   enkeywords: ("Vue", "SpringBoot framework", "MySQL", "interaction"),
27 )
28
29 = 绪论
30
31 == 研究目的和意义
32 本课题计划完成一个全面可靠高效能实现信息透明化的房屋租赁管理系统 @drill。通过完成并完善本选题,
总结了本科学期间的知识点,充分培养了动手能力,学以致用实践代码的编写。并且在解决问题的过程中,
对 Java 技术有进一步的认知,提高自己的综合水平,锻炼今后的工作中遇到困难的解决钻研能力。
33
34 房屋租赁系统是一个在线平台,用于连接房东和租户,并提供一些自动化功能,使整个租赁过程更加便捷。它的
出现可以带来多方面的意义,下面笔者将详细阐述。
35

```

- 36 首先，房屋租赁系统可以促进市场的透明度和规范性。在传统的租赁市场中，信息不对称、合同不规范等问题比较普遍 @foia，这给租户和房东都带来了很多麻烦。而房屋租赁系统能够提供租户信用评分、房源信息管理等功能，让市场更加清晰透明。租户可以更加直观地评估房东和房源的可靠程度，从而做出更加明智的决策。房东也可以通过系统进行精准定价、优化房屋配置、提高竞争力，从而获得更好的租客来源和租金收益。
- 37 其次，房屋租赁系统可以提高租赁效率和用户体验。通过在线预订、租金支付、合同签署等自动化流程，租户和房东可以省去很多繁琐的手续和沟通过程，节约时间和精力。在租赁过程中，系统还可以提供在线客服、售后服务等功能，让用户获得更好的使用体验和感受。
- 38 第三，房屋租赁系统可以促进科技创新和数字化转型。随着信息技术的不断发展，越来越多的企业开始将传统业务与互联网结合，尝试探索新的商业模式。房屋租赁系统作为一种典型的互联网+业态，正是应用了信息技术和数字化手段，使得租赁市场变得更加智能、高效、便捷。同时，系统的开发和运营也需要涉及多种技术和人才，有助于提升整个行业的科技含量和创新能力。
- 39 最后，对于平台运营商而言，房屋租赁系统也是创造盈利模式的一个途径。通过收取服务费或广告费等方式获得收入，并通过数据分析和挖掘等手段获取更多商业机会，这些都是房屋租赁系统带来的商业价值。
- 40 总之，通过本课题的研究，目的是针对存在的房屋租赁问题进行改善，实现信息的公开透明化，全面收集数据解决房源少的问题。建立平台政策和优化监管功能，提高房屋租赁平台的综合水平，对人们的生活带来便利。
- 41 == 研究背景
- 42 === 国内发展（应用）现状
- 43 目前国内的城镇化战略使得人口重心移动，流入城市人口过多，城市建设发展加快。由于这些因素房价增长，房租租赁建设紧张，越来越多的房屋租赁软件崛起，但是也存在着巨大缺陷。
- 44 首先是链家成立于 2001 年，是中国领先的房地产服务企业。业务覆盖广，房源质量高，服务者素质高。上面的房源基本上是通过中介上传来进行出租，虽然优点颇多，但是需要交过高的中介费，这对刚毕业的大学生和刚在城市工作的人不太友好。
- 45 随后 2007 年安居客挤入租房行列，独有的“个人房源”选项虽然采用真实照片，但是少之甚少，大部分还是中介上传并且很多房源还是虚假房源。据调查显示，安居客很多黑中介，会泄露用户个人信息，所以这是严重的监管不到位和信息不透明的现象。
- 46 通过对十年来租房平台弊端的总结以及改善，于 2011 年成立自如租房，也是链家产业下的长租公寓品牌。一改往日的中介入驻，使用自己的管家联系户主进行拍照看房，保障房源都是真实可靠的。但盈利模式是赚取差价以及收取服务费，价格往往高于市场价。
- 47 由此看来，国内的房屋租赁系统在不断创新、提升用户体验和服务质量方面已经取得了很大的进展。随着市场竞争的加剧和技术的迭代升级，这些平台也将带来更多的变化和发展。
- 48 === 国外发展（应用）现状
- 49 国外的房屋租赁系统发展状况相对较为成熟已经形成了一些领先的平台和商业模式。下面举几个典型的例子。
- 50 Zillow 是一个美国的房地产信息网站，提供买卖房屋、出租房屋、房屋估价和市场分析等服务。它将各种房源信息整合到一起，在线显示给用户，方便他们进行筛选和比较。同时，它还提供比较精准的房屋估价功能，帮助房东和买家得到更好的交易结果。Zillow 目前已经成为美国房地产市场上的重要参与者之一。
- 51 Rightmove 创建于 2000 年是英国本地最大的房地产网站。房源多，信息量庞大并且还有很多自定义选项，比如说带不带家具，预算价格和选择区域。缺点是房租坐地起价。价格经常随市场变动，房源多虚拟信息也多不方便管理。
- 52 apartment.com 也创建于 2000 年。是美国目前功能完善，房源信息真实，操作方便快捷的房屋租赁网站。优点在于非常定制化满足租客需求，功能全面选择也多，房东友善与租客和睦共处。但需要走很多流程，也需要一些良好信用要求，只适合一些稳定的租客。
- 53

67 总之，国外的房屋租赁系统在创新商业模式、提升用户体验、拓展市场规模等方面都取得了很大的成就。未来随着技术的不断进步和市场的不断竞争，这些平台将继续探索新的发展机遇和商业模式，并为全球的租赁市场带来更多创新和变革。

68

69 == 论文结构

70 本文的内容主要包括以下八个部分：

71

72 第一部分是绪论，分别从国内和国外两个方面对房屋租赁管理系统设计与实现进行了背景介绍，主要说明了这次对房屋租赁管理系统开发的主要原因。

73

74 第二部分是对房屋租赁管理系统过程中需要用到的预备知识、概念原理和技术进行解释说明，在正式进行开发前需要对这些相关的理论进行系统地学习，以便理解开发过程。

75

76 第三部分是房屋租赁管理系统的用户权限设计，先对租客权限分析，然后对户主的权限分析以及对管理员权限的分析。

77

78 第四部分是对房屋租赁管理系统的整体设计，房屋租赁管理系统分为两个部分，前端设计模式和后端设计模式。

79

80 第五部分是系统的功能实现，首先对系统开发环境作了介绍，然后进行核心功能的实现。

81

82 第六部分是测试与优化，对游戏的运行效率进行了评估。

83

84 第七部分是总结篇，对该游戏进行了综述性的评述，并阐述了当前的不足以及将来可以进一步完善的地方。

85

86 最后列出了本文所参考的相关资料。

87

88 = 预备知识及原理说明

89

90 == MVP 设计模式

91 **MVP (Model-View-Presenter)**，是一种被广泛用于开发的架构设计模式，是由 **MVC** 延伸出来的衍生物。作为衍生物，**MVP** 在整体的设计思想上继承了 **MVC**，其中 **M (Model)** 负责数据的处理，**V (View)** 负责界面数据的可视化和用户交互，而 **C&P (Controller&Presenter)** 在大体上均负责逻辑的处理。

92

93 如 `@fig:mvp` 中 **MVP** 架构图所示，在 **MVP** 设计模式中，**Model** 的具体任务为从数据库中获取数据，并对数据加以简单处理或传递转存；**View** 的任务则是将经过处理的数据以合理的形式展现出来，并提供用户可操作的交互点，完成数据的可视化转换；而 **Presenter**，即是自 **MVC** 演变的核心，**Presenter** 主要负责将 **Model** 所获取传递的数据进行加工转换，随后交付 **View** 进行展现。在整个 **MVP** 设计模式中，**Presenter** 作为 **Model** 与 **View** 的传递桥梁，在帮助两者沟通的同时，又隔绝了两者，即 **View** 无法直接跳过 **Presenter** 对 **Model** 进行操作。

94

```
#figure(
  caption: [MVP 架构图],
  image("img/mvp.png", width: 50%)
)
```

95

96

97

98

99

100 这样的分隔式处理，使 **Model** 和 **View** 达到完全分离，**Model** 不受 **View** 的影响，降低了模块与模块间的耦合；同时这样的设计模式也简化了 **Activity** 的工作任务，在 **Activity** 中只需要处理生命周期的任务，使代码在一定程度上更加简洁明了；由于业务逻辑被分配至 **Presenter** 模块中，使得 **Activity** 不会因为后台线程的引用而无法回收资源，有效的避免了 **Activity** 的内存泄漏；模块间的联系紧密程度降低，且各模块有各自明确的分工，这使得代码的层次更加清晰，提高了代码的可读性；而 **Presenter** 多样的具体实现让单元测试变得更加便捷。

101

102 == 开发语言和开发工具

103 本次研究的开发语言主要使用 **Java**。作为 **Android** 软件目前的主流开发语言，相较于 **C++** 更为简单，同时又摘取了许多其它开发语言的优点特性，而面向对象的编程是 **Java** 语言最主要的特点之一。作为被普遍使用

的面向对象的编程语言的代表，**Java** 不仅拥有所有面向对象编程语言都具备的普遍特性，即封装性、多态性以及继承性，它同时拥有动态联编性，更适应于面向对象的设计方法。除此之外，**Java** 同时具有简单性、平台独立性、多线程以及安全可靠性的诸多优点。目前市面上的大多数 **Android** 软件都是运用 **Java** 语言进行开发，相对的，由于应用了 **Java** 核心类的知识，使得在安卓开发中 **Java** 语言占据了强有力的优势。

- 104  
 105 本次研究的开发工具使用 **Android Studio**。**Android Studio** 是由谷歌推出的一款 **Android** 集成开发工具，它基于 **IntelliJ IDEA**，和 **Eclipse** 的安卓开发工具类似，为开发者提供了集成的 **Android** 开发工具。
- 106  
 107 == MySQL 数据库  
 108 **MySQL** 是一种关系型数据库管理系统(RDBMS)，由瑞典 **MySQL** AB 公司开发，现在由 **Oracle** 公司拥有和维护。它是一款高效、可靠、稳定的开源数据库软件，是 **LAMP** (**Linux+Apache+MySQL+PHP**) 架构的重要组成部分，具有跨平台性和良好的兼容性。
- 109  
 110 **MySQL** 支持多种操作系统(**Linux**、**Windows**、**Mac OS** 等)以及多种编程语言(**C**、**C++**、**Java**、**Python**等)，提供了完整的 **SQL** 标准，包括事务处理、触发器、存储过程、视图等功能。下面是 **MySQL** 的一些主要特点：
- 111  
 112 高性能：**MySQL** 采用了多线程、异步 **I/O** 等技术，可以高效地处理大量数据请求。另外，**MySQL** 还支持索引、查询优化等功能，可以提高数据的访问速度。
- 113  
 114 可靠性：**MySQL** 支持数据备份、恢复等功能，可以保证数据的安全性和可靠性。此外，**MySQL** 还支持主从复制、集群等功能，可以实现高可用性、容错性等特点。
- 115  
 116 易用性：**MySQL** 的安装和配置比较简单，可以快速上手使用。同时，**MySQL** 提供了完整的文档和社区支持，用户可以获取到丰富的资源和帮助。
- 117  
 118 开放性：**MySQL** 是开源软件，可以免费使用和修改。同时，**MySQL** 还拥有庞大的用户社区和开发者社区，可以实现定制化开发和二次开发等需求。
- 119  
 120 可扩展性：**MySQL** 提供了完整的插件机制和 **API** 接口，可以方便地进行扩展和集成。开发者可以基于 **MySQL** 开发各种类型的应用程序，如电子商务、社交网络、游戏等。
- 121  
 122 总之，**MySQL** 是一款成熟、稳定、可靠、高效、易用、开放和可扩展的关系型数据库管理系统。它被广泛应用于互联网领域，支撑着许多大型网站和应用程序的数据存储和处理，如 **Facebook**、谷歌、雅虎、推特等。
- 123  
 124 == 版本控制软件 **Git**  
 125 **Git** 是一个开源的分布式版本控制工具，开发者们可使用 **GitHub** 等平台应用 **Git**，进行协同开发。**Git** 由 **Linus Torvalds** 开发，最初开发是为了满足团队 **Linus** 内核开发的需求，与常用的集中式版本控制工具不同，**Git** 最大的特点就是采用了分布式的模式，摒弃了集中式工具中只有中心服务器的模式，每位参与开发的使用者都持有完整版本库，有效避免了使用集中式版本控制系统的协同开发中的不稳定性，不必担心中心服务器数据丢失，同时也加快了使用者们的代码交流。在 **Git** 的基本工作中主要分为四个版块，图中从左至右分别为：工作区间、暂存区、本地仓库以及远程仓库。其中，工作区间为工程目录文件，主要在工作区间执行 **Git** 命令；暂存区为暂存上传的代码数据的区间；本地仓库用来存储本地代码，同时是本地代码与远程代码的中转站；远程仓库为中心服务平台，用于远程保存提交的代码。**Git** 基于这四个模块，构成了自身的基本工作流程：开发者修改工作区间的文件，执行 **add** 后文件将被添加到暂存区，随后 **commit** 提交会将文件保存至本地仓库，最终执行 **push** 推送，文件将会被传输到远程仓库进行保存。
- 126  
 127 == 本章小结  
 128 本章简要介绍了项目开发所需的相关知识，其中包括 **MVP** 设计模式的设计原理、所用的主要开发工具以及开发语言、**MySQL** 数据库以及版本控制器 **Git** 的简介。以上四个模块的储备知识将为后续项目开发提供可靠的理论支持，加强本次研究的可行性。
- 129  
 130 = 用户权限设计分析  
 131 == 租客权限设计分析

- 132 房屋租赁系统的租客权限设计是保证系统信息安全、租户合法权益及资金安全的重要环节。在设计时，需要考虑以下几个方面：
- 133
- 134 + 注册及登录权限：租客以注册用户身份使用房屋租赁系统，需要提供基本的个人信息并完成注册流程。用户通过登录账号和密码访问系统并进行操作，登录过程需要进行身份验证。
- 135
- 136 + 查询与浏览权限：租客可以在系统中浏览已发布的房源信息，并通过搜索等方式查询符合自己需求的房屋信息，但不允许修改任何房源信息。
- 137
- 138 + 预订和支付权限：租客可以选择心仪的房源并提交预订请求。在预订期间，租客可以更改预订细节（如租期、价格、入住时间等），但对于一些核心数据，例如房东联系方式，租客无法直接获得。当确认预订后，租客将通过平台支付租金押金等费用。
- 139
- 140 + 评价与投诉权限：租客有权在租期结束后评价房东、房源的质量，并且有权在必要的情况下投诉相关问题。评价和投诉信息将成为其他租客或房东参考的重要依据，因此也需要对评价和投诉信息的真实性进行保护。
- 141
- 142 + 退款与维权权限：当租客在租期内出现合同纠纷或房源质量问题等情况时，拥有退款和维权的权利。退款和维权流程需要在平台规范的程序下操作，并需要提供相关证据以支持判断。
- 143
- 144 + 个人信息保护：租客的个人隐私信息是需要得到网站的严格保护的。房屋租赁系统要求租客提供的信息应该仅限于必要的身份验证、预订等操作所需的必需信息。同时，房屋租赁系统也应该采取各种措施来确保用户信息的保密性。
- 145
- 146 总之，租客权限设计应该涵盖注册、浏览、预订、支付、评价、投诉、退款、维权和个人信息保护等方面，以保证租户在使用房屋租赁系统时，能够安全、便捷地完成租赁交易，并保护其合法权益。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。
- 147
- 148 == 户主权限分析
- 149 房屋租赁系统中，户主（即房东）权限设计是保障系统信息安全、保护房东权益的关键环节。在设计时，需要考虑以下方面：
- 150
- 151 + 房源发布与编辑权限：作为房东，他们可以在房屋租赁系统上发布自己的房源信息，并对信息进行修改、更新或删除。需要注意的是，为了保证租客的安全和信任，发布的信息需要经过平台的审核才能上线。
- 152
- 153 + 订单管理权限：房东可以查看自己的订单列表，包括预订信息、租金支付情况、租客身份信息等，并且可以接受或拒绝预订请求。此外，房东还可以在租期结束后收到租客提供的评价和投诉信息。
- 154
- 155 + 报表分析权限：为了了解自己的房产运营状况，房东应该具有数据分析及图表功能，以便更好地了解自己的房产出租状态。通过这个功能，房东可以查看订单、收支情况、退款等相关报表，从而获得及时的反馈和监控。
- 156
- 157 + 合同签署权限：当租客确定预订房源之后，系统会生成一份标准的租赁合同，其中包括租期、租金、押金等重要信息。房东需要在签署合同前审查和确认租期等条款，确保自己的权益得到保障。
- 158
- 159 + 费用管理权限：房东可以在系统中设定租金、押金以及其他费用，并查看租客的付款情况和账单结算。此外，当存在租客欠费或违约时，房东还可以使用平台提供的退款和维权流程进行相应操作。
- 160
- 161 + 房源维护权限：房东需要保持房源的良好状态，包括维修、清洁、安全等方面。在出现问题时，房东可以通过平台提供的房屋维修服务来解决问题，保证租户的居住质量。
- 162
- 163 总之，房东权限设计需要涵盖房源发布、订单管理、报表分析、合同签署、费用管理和房源维护等方面，以保证房东能够方便、快捷地完成租赁交易，并保障房东的合法权益。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。
- 164
- 165 == 管理员权限分析

166 房屋租赁系统的管理员权限设计是保证系统信息安全、维护平台稳定运营的重要环节。在设计时，需要考虑以下几个方面：

167

168 + 用户管理权限：管理员需要管理所有注册用户的账号和个人信息。包括审查新用户注册申请、处理账号冻结、解锁等操作。

169

170 + 房源管理权限：管理员需要审核并管理所有房东发布的房源信息，包括发布的内容、图片、价格、位置等信息的真实性和合规性。

171

172 + 订单管理权限：管理员有权查看所有订单的状态、流程和相关信息，并对订单进行修改和删除。此外，管理员还能够协调和处理与订单相关的纠纷和退款问题。

173

174 + 资金管理权限：管理员可以管理平台上所有的收支流水，包括租金支付、押金、提现等。应该对平台内的各种资金流通情况进行监控，及时发现和处理异常情况。

175

176 + 系统维护权限：管理员有权查看平台的服务器负载、网站访问量等运营数据，以便根据实际情况优化系统。同时，管理员还需要确保系统的安全性，防范黑客攻击和信息泄露等安全事件。

177

178 + 数据分析权限：管理员可以通过数据分析工具对平台内的各项指标进行分析和预测，以便更好地指导平台的运营策略。通过数据分析，管理员可以了解用户行为、市场趋势等信息，进而制定优化平台的策略。

179

180 总之，管理员权限设计需要涵盖用户管理、房源管理、订单管理、资金管理、系统维护和数据分析等多个方面，以保证管理员能够快速、高效地处理各种运营问题，并确保平台的稳定性和安全性。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。

181

182 == 本章小结

183 本章主要通过绘制了项目的用例图和功能结构图，对用例分析、用例关系进行了较详细介绍。用户权限设计包含了三个部分，并对租客权限、户主权限以及管理员权限都做了并进行简要说明，完成了项目需求分析阶段的整体设计。

184

185 = 系统整体设计

186 == 前台设计

187 === 概述

188 房屋租赁系统的管理员权限设计是保证系统信息安全、维护平台稳定运营的重要环节。在设计时，需要考虑以下几个方面：

189

190 用户管理权限：管理员需要管理所有注册用户的账号和个人信息。包括审查新用户注册申请、处理账号冻结、解锁等操作。

191

192 房源管理权限：管理员需要审核并管理所有房东发布的房源信息，包括发布的内容、图片、价格、位置等信息的真实性和合规性。

193

194 订单管理权限：管理员有权查看所有订单的状态、流程和相关信息，并对订单进行修改和删除。此外，管理员还能够协调和处理与订单相关的纠纷和退款问题。

195

196 资金管理权限：管理员可以管理平台上所有的收支流水，包括租金支付、押金、提现等。应该对平台内的各种资金流通情况进行监控，及时发现和处理异常情况。

197

198 系统维护权限：管理员有权查看平台的服务器负载、网站访问量等运营数据，以便根据实际情况优化系统。同时，管理员还需要确保系统的安全性，防范黑客攻击和信息泄露等安全事件。

199

200 数据分析权限：管理员可以通过数据分析工具对平台内的各项指标进行分析和预测，以便更好地指导平台的运营策略。通过数据分析，管理员可以了解用户行为、市场趋势等信息，进而制定优化平台的策略。

201

202 总之，管理员权限设计需要涵盖用户管理、房源管理、订单管理、资金管理、系统维护和数据分析等多个方面，以保证管理员能够快速、高效地处理各种运营问题，并确保平台的稳定性和安全性。同时，由于每个房屋租赁系统的需求和特点不同，具体的权限设计还需要根据实际情况进行灵活调整和优化。

203

204 **==== 设计举例**

205 **===== 登录注册页面**

206 在设计房屋租赁管理系统的登录注册功能时，需要考虑到用户体验、安全性和数据完整性等方面。以下是对登录注册页面的设想：

207

208 - 注册功能应该包括基本的个人信息，如用户名、密码、手机号码、邮箱等，并对用户输入信息的格式进行校验，确保其准确性。

209

210 - 登录功能可以使用用户名/手机号码/邮箱+密码的方式进行认证，也可以通过第三方登录（企业微信）来实现便捷登录。

211 - 为了保障用户的账号安全，应该设置密码强度要求，例如要求密码长度在 6-16 个字符之间，包含大小写字母、数字和特殊字符等。

212

213 - 对于忘记密码的情况，应该提供重置密码的功能，可以通过手机短信或邮箱验证来进行身份验证。

214

215 - 为了防止恶意注册和滥用系统资源，可以设置验证码功能，确保只有真实用户才能注册。

216

217 - 在用户注册成功后，可以通过邮件或短信的形式发送激活链接，确保用户账户的有效性。

218

219 - 在登录过程中可以使用 JWT Token 等技术来提高系统的安全性，避免明文传输密码等敏感信息。

220

221 - 为了方便用户管理自己的个人信息，可以在用户登录后提供个人中心页面，可供用户修改个人资料、查看订单等功能。

222

223 综上所述，登录注册是房屋租赁管理系统中的一个基础模块，合理设计并实现这个模块不仅可以提高用户体验，还能保障系统的安全性和数据完整性。

224

225 **===== 增删改查**

226 在设计房屋租赁管理系统的权限增删改查功能时，需要考虑到用户角色、权限类型、数据保护和操作审计等方面。以下是对增删改查的设想：

227

228 - 用户角色：在系统中需要对不同的用户进行角色分类，例如管理员、普通用户、房东等。每个用户角色都应该有对应的权限列表，限制其对系统中数据的访问和修改。

229

230 - 权限类型：在分配权限时，需要考虑到权限的粒度，例如可以为某个角色分配“查看房源”、“编辑房源”、“删除房源”等具体的权限。

231

232 - 数据保护：在设计权限控制功能时，需要考虑到对数据的访问控制，确保只有拥有合法权限的人员才能访问敏感数据。例如，在实现个人信息修改功能时，可以对用户密码/手机号等敏感信息进行加密处理，并通过访问控制策略管理数据访问权限。

233

234 - 操作审计：在系统中记录所有的权限操作，包括新增、删除和修改等，以便后续跟踪操作历史和查询问题根源。

235

236 在实现权限增删改查功能时，笔者们可以采用常用的 RBAC (Role-Based Access Control) 模型，该模型将权限授予给用户角色而非直接授予单个用户，可以有效地保证系统安全性和数据完整性。同时，可以使用框架中提供的权限管理模块来实现基础权限控制功能，例如 Spring Security、Apache Shiro 等。

237

238 综上所述，权限增删改查是房屋租赁管理系统中非常重要的一部分，需要合理设计并实现这个功能，以便保障用户数据的安全性和系统的稳定性。

239

240 **== 后端设计**

241 后端部分使用了 Java 的 **SpringBoot** 框架，并且采取的 **MySQL** 数据库来存储数据。在设计房屋租赁管理系  
统后端时，需要考虑到架构、技术栈、安全性和扩展性等方面。以下是一些设想部分：

242

243 - 架构：在确定系统架构时，可以采用分层架构或微服务架构，将业务逻辑、数据访问等功能进行分离并单  
独部署，提高系统的可维护性和扩展性。

244 - 技术栈：在选择技术栈时，可以考虑到开发效率、系统稳定性、性能和安全等方面，并选取相应技术实  
现。在这次实现中将使用 **SpringBoot** 框架技术实现系统功能。

245

246 - 安全性：在保障系统安全性方面，可以采用多种措施，例如对用户输入的数据进行严格验证和过滤，使用  
**SSL** 加密协议保证数据传输的安全，使用 **JWT Token** 等技术防止 **CSRF** 等攻击。

247

248 - 扩展性：在设计系统时，需要充分考虑到系统的可扩展性，使得系统能够满足未来的需求。例如，可以引  
入消息中间件，将不同的模块解耦，提高系统的可维护性和扩展性。

249

250 另外，在设计数据库时，需要考虑到数据表之间的关系，确保数据的完整性和一致性。例如，在房源表中可以  
添加房东 ID 字段，与房东信息表进行关联，便于查询、更新等操作。

251

252 综上所述，房屋租赁管理系统后端设计是非常重要的一部分，需要充分考虑到架构、技术栈、安全性和扩展性  
等方面，以实现系统的高可用性、高可扩展性和高性能。

253

254 == 后端开发可能使用的的关键类

255 在设计和开发房屋租赁管理系统后端时，可能会用到许多关键类，包括控制器类、服务类、**DAO** 类、实体类等。  
以下是一些后端开发过程中可能是用到的关键类：

256

257 控制器类：控制器类主要负责接受用户请求，调用相应的服务类完成业务逻辑的处理，并将处理结果返回给前  
端页面。例如，可以根据业务需求设计房源控制器类、订单控制器类等。

258

259 服务类：服务类主要负责业务逻辑的处理，包括对数据的读取、修改、删除等操作。例如，可以设计房源服务  
类、订单服务类等。

260

261 **DAO** 类：**DAO** (**Data Access Object**) 类主要负责数据访问操作，与数据库交互并提供基本的 **CRUD**  
(**Create, Read, Update, Delete**) 操作。例如，可以设计房源 **DAO** 类、订单 **DAO** 类等。

262

263 实体类：实体类代表了系统中的各种对象，包括房源、订单、用户等等。这些实体类通常包含实体属性、构造  
方法以及与数据库中表的映射关系，用于实现对象的持久化存储和操作。

264

265 除此之外，还可能会使用到工具类、配置类等，用于实现系统中的各种辅助功能。例如，可以使用 **MyBatis**  
来实现数据库访问，使用 **Redis** 等缓存技术提高系统性能，使用 **Spring Security** 等安全框架来实现权  
限控制。

266

267 在设计关键类时，需要充分考虑到系统的业务需求和技术实现，遵循面向对象编程的原则，使得系统的代码结  
构简洁、可读性好、易于维护。

268

269 == 本章小结

270 本章通过对房屋租赁系统的分析，结合实际的工作情况，对房屋租赁系统设计的工作方面，包括前端部分和后  
端部分三个模块进行了介绍。

271

272 = 系统实现

273 == 开发环境

274 选择合适的开发工具与语言可以有效的提升开发效率。

275

276 === 软件环境

277 ===== **Visual Studio Code**

278 **Visual Studio Code**，也被简称为 **VS Code**，是一款免费、开源、轻量级的集成开环境（**IDE**）。由微软于  
2015 年发布，现已成为一款颇受开发者喜欢的代码编辑器。**Visual Studio Code** 有以下的优点：

279

280 + 语言支持: VS Code 支持数十种编程语言, 包括 JavaScript、TypeScript、Python、Java 等。它能通过插件市场获取更多语言支持。

281

282 + 调试工具: VS Code 拥有内置的调试器, 支持 Node、TypeScript 和 JavaScript 的调试通过插件市场, 可以获取其他语言的调试器。

283

284 + 智能代码提示: VS 支持自动补全、变量定义跟踪等功能, 能够提高开发效率。

285

286 + 版本控制: VS Code 支持 Git 和 SVN 等版本控制系统, 可以直接在编辑器中进行版本管理。

287

288 + 扩展性: Code 支持丰富插件生态, 用户可以通过插件市场获取各类插件, 例如风格检查、项目等。

289

290 + 跨平台: VS Code 支持 Windows、macOS 和 Linux 操作系统。

291

292 由此看来, Studio Code 以其简洁易用、功能强大、跨平台等特点受到越来越多开发者的推崇, 并成为全球范围内非受欢迎的代码编辑器。

293

294 === IDEA

295 IDEA 是 JetBrains 公司开发的一款 Java IDE, 优秀的代码编辑器和调试工具。它支持多种编程语言, 包括 Java、Kotlin、Groovy、Scala 等。IDEA 是基于 IntelliJ IDEA Community Edition 打的, 具有轻量且易于学习的特点。

296

297 IDEA 提供了智能代码补全、重构、代码分析、版本控制以及其他实用工具, 使得开发者可以更加有效地进行开发这款 IDE 还包括大量的插件, 例如 UI 设计器、数据库管理、Web 开发, 这些插件可以帮助开发人员进一步提高开发效率。

298

299 与其他 IDE 相比, IDEA 在代码提示、Refactor 等方面表现卓越, 并且具有强大的搜索功能和快捷键设置。好的辅助功能使得 IDEA 为 Java 开发者的首选 IDE 之一。

300

301 总之, 作为业界标准之一的 Java 开发工具之一, IDEA 在提高编写效率、缩短开发周期等方面具有重要作用。

302

303 === MySQL 数据库

304 MySQL 是一个广泛使用的关系型数据库管理系统, 开源免费。MySQL 是轻量级且易于安装部署, 在开发 Web 应用非常流行。MySQL 支持多种操作系统, 并可以通过多种编程语言进行访问。MySQL 功能丰富, 提供了完整的数据管理、查询和处理功能, 可以存储大型数据集并支持高并发访问。MySQL 的安全性也得到了很好的保证, 支持基于角色的访问控制和、表、列级别的权限管理。

305

306 === Navicat

307 Navicat 是一款功能强大的数据库管理工具, 支持多种数据库管理系统, 例如 MySQL、MariaDB、Oracle、PostgreSQL 等, 也包括多种操作系统, 例如: Windows、macOS、Linux 等。Navicat 是一款商业软件, 提供了丰富的和全面的性能优化, 旨在帮助开发者更加高效地管理和维护数据库。

308

309 === 硬件环境

310 操作系统: Windows 10

311

312 处理器: Intel(R) Core(TM) i5-8300H CPU

313

314 显卡: NVIDIA GeForce GTX 1060

315

316 == 前端实现

317 === 界面设计

318 + 登录界面: 需要输入用户名, 密码以及验证码登录。

319 #image("img/login.png", width: 70%)

320 #image("img/register.png", width: 70%)

321

322 + 用户管理面: 管理员可以查看和管理所有用户的信息, 包括注册时间、联系方式等。

323  
 324 + 房源管理界面：管理员可以查看和管理所有房源的，包括图片、描述价格、位置等，并能对新增、更新或删除房源信息。  
 325  
 326 + 订单管理界面：管理员可以查看和管理所有订单的信息，包括订单状态、支付情况等，并能对订单进行审核或取消。  
 327  
 328 + 系统设置界面：管理员可以针对系统进行相关设置，如邮件通知、短信提醒等。  
 329  
 330 + 日志管理界面：管理员可以查看系统运行日志及操作日志。  
 331  
 332 + 权限管理界面：管理员可以管理系统用户的权限，分配不同角色和权限。  
 333  
 334 + 数据备份和恢复界面：管理员可以进行数据库备份和恢复操作。  
 335  
 336 === 功能设计  
 337 + 注册界面：没有登陆过的用户先要注册了之后才能登陆。需要输入手机号、姓名、密码，并通过图片验证码才能进行注册。  
 338  
 339 + 用户管理面：管理员可以查看和管理所有用户的信息，包括注册时间、联系方式等。  
 340  
 341 + 房源管理界面：管理员可以查看和管理所有房源的，包括图片、描述价格、位置等，并能对新增、更新或删除房源信息。  
 342  
 343 + 订单管理界面：管理员可以查看和管理所有订单的信息，包括订单状态、支付情况等，并能对订单进行审核或取消。  
 344  
 345 + 系统设置界面：管理员可以针对系统进行相关设置，如邮件通知、短信提醒等。  
 346  
 347 + 日志管理界面：管理员可以查看系统运行日志及操作日志。  
 348  
 349 + 权限管理界面：管理员可以管理系统用户的权限，分配不同角色和权限。  
 350  
 351 + 数据备份和恢复界面：管理员可以进行数据库备份和恢复操作。  
 352  
 353 === 动态设计  
 354 + 响应式设计：现在越来越多的人使用手机和平板电脑上网，因此，一个好的房屋租赁管理系统必须具备响应式设计，以适应不同设备的屏幕尺寸和分辨率。通过这种方式，用户可以随时随地访问系统而不必担心屏幕尺寸或布局问题。  
 355 + 数据可视化：一个好的房屋租赁管理系统应该具有良好的数据可视化功能，以帮助用户更好地了解租赁业务的状况。例如，可以使用柱状图或折线图等方式展示关于租户数量、租金收入和支出等关键指标的数据。这样，用户可以通过数据可视化快速评估其租赁业务的运营状况。  
 356 + 自动化流程：一个好的房屋租赁管理系统还应该具备自动化的流程，以提高用户的工作效率。例如，当一份合同到期时，系统应该能够自动提醒用户需要更新合同，而不必手动查找过期合同。这种自动化流程可以帮助用户大大减少操作时间和精力。  
 357 + 安全性：一个好的房屋租赁管理系统必须具备安全性。租赁管理系统通常包含很多敏感数据，例如租客的个人信息和账单信息等，因此必须确保这些数据不会被未经授权的第三方访问或泄露。系统应该采用加密方法来保护存储在数据库中的信息，同时还应该设定权限系统，以限制每个用户可以访问的数据量和范围。  
 358 #image("img/lock.png")  
 359  
 360 总之，房屋租赁管理系统是一个非常重要的应用程序，可以帮助房地产公司、物业管理公司和个人房东等有效管理其租赁业务。通过考虑响应式设计、多语言支持、数据可视化、自动化流程以及安全性等关键因素，可以确保该系统具有良好的用户体验，并能够在市场上获得成功。  
 361  
 362 == 后端实现  
 363 后端部分使用了 Java 的 SpringBoot 框架，并且采取的 MySQL 数据库来存储数据。大体步骤如下：

```

364 - 先搭建一个 SpringBoot 项目，使用了 IDEA 开发工具来进行后端开发。
365 - 确定系统架构和数据库结构，设计 ER 图创建相应的表格
366 - 在 pom.xml 文件中添加必的依赖，包括 Boot 和 MySQL 驱动。
367 - 编实体类 (Entity) 和 DAO 层 (Data Access Object) 对应的 Repository 接口，以及对应的 SQL 语句。
368 - 创建服务层 (Service) 并实现事务管理。
369 - 配置数据库连接池和相关的参数，包括数据库 url、用户名和密码等。
370 - 编写控制层 (Controller) 及相应的 API 接，提供相应的增删改查功能。
371 - 进行单元测试和集成测试，保证功能常。
372 - 部署项目到服务器上，让用户能够直接使用并进行后续维护。
373
374 #image("img/ideal.png")
375 #image("img/idea2.png")
376 #image("img/idea3.png")
377 #image("img/idea4.png")
378 #image("img/idea5.png")
379 #image("img/idea6.png")
380 #image("img/idea7.png")
381
382 == 后端开发的关键类
383 在使用 Spring Boot 框架中实现功能有许多关键类，下面将介绍部分关键类。
384 1. @RestController: 注解表示该类是一个 RESTful API 的控制器。
385 2. @RequestMapping: 该注解表示该方法对应的 URL 请求路径。
386 3. @RequestBody: 该注解表示该参数是请求体中的数据。
387 4. @Autowired: 该注解表示该属性需要自动注入某个对象。
388 5. JpaRepository: 该接口提供了很多通用的数据库操作方法
389 6. @Entity: 该注解表示该类应数据库中的一张表。
390 7. @Table: 该注解表示该类对应数据库中的一张表，来指定表的名称和其它属性。
391 8. @GeneratedValue: 该注解表示该属性的值数据库自动生成。
392 9. @Id: 该注解表示该属性是实体类的主键。
393 10. @Column: 该注解表示该属性对应数据库中的一列，用来指定字段的名称和其它属性。
394 11. @: 该注解表示方法执行时需要开启一个事务。
395 12. @Modifying: 该注解表示该方法执行时需要修改数据库中的数据。
396
397 === 数据库设计
398 #figure(
399   caption: [category 表],
400   kind: table,
401   supplement: "表",
402   ``tbl
403   C | C | C | C | C | Cx
404   L | L | R | C | C | Lx.
405   -
406   字段名称 | 类型 | 长度 | 是否 null | 主键 | 字段说明
407   -
408   cid | int | 5 | 是 | 是
409   -
410   cname | varchar | 10 | 否
411   ``..
412
413 )
414
415 === 接口设计
416 勾股定理可用公式: $a^2 + b^2 = c^2$ 表示。
417
418 #figure(
419   caption: [数列求和],

```

```

420 kind: math.equation,
421 supplement: "公式",
422 $ sum_(k=1)^n k = (n(n+1)) / 2 $
423 )
424
425 == 本章小结
426 本章结束了房屋租赁管理系统的所需的硬件环境。对前端页面和后端的具体实现进行了介绍。同时，对后端关键类的实现方法进行了阐述。
427
428 = 系统测试与运行
429 == 测试
430 软件测试通常是指验证与确认两部分。该系统的主要进行了以下几个方面的测试：资源测试、功能测试、任务测试。测试的流程 @fig:testflow 所示。
431 #figure(
432   caption: [测试流程],
433   image("img/testflow.png", width: 50%)
434 )
435
436 === 资源测试
437 房屋租赁管理系统的资源测试是对系统进行性能，以确保系统在高负载、大流量情况下仍然能够稳定运行。下面介绍具体的测试内容和流程：
438 1. 负载测试：这种测试通过模拟用户并发访问系统，同时观察系统响应时间、吞吐量等指标来评估系统在承受并发请求时的能力。测试时需要针对不同的业务场景（例如搜索房源、订单、支付等）进行测试，并分别记录响应时间请求成功率、错误码等指标。
439 2. 压力测试：将系统负载逐渐增加，直到达到系统极限，观察系统在极端情况下的表现。测试阶段需要设定并观察系统响应时间、CPU 利用率、内存使用量、带宽利用率等关键指标，以保证系统在硬件资源撑不住的情况下和内部逻辑出现瓶颈的能保持应有的能和稳定性。
440 3. 并发测试：模拟多个用户并发访问同一个页面或接口，以检测系统在高并发情况的表现。并发测试数据需要根据业务特点精确定制，去模拟正常和异常下的并发请求比如打开一个页面，提交同一个表单，支付多少订单，同时踢出多少用户等。
441 4. 数据量测试：将大数据导入系统中进行测试，以检验系统的性能和稳定性。测试过程中需要特别关注系统在数据查询、更新、删除等操作时表现，同时还需要注内存和 CPU 利用率的变化趋势；
442 5. 日志跟踪测试：通过记录系统日志并进行分析，可以追踪系统在某个时间段或者某个业务场景下执行的情况。这种测试可以发现系统运行时出现的异常、慢查询和错误。
443 经过以上测试，可以获取到系统各方面的性能参数与指标从而进行较好的化和调整。
444
445 === 功能测试
446 房屋租赁管理系统的功能测试是对系统进行项功能方面的测试，以保证系统满足需求并能正常运行。下面是测试内容的流程：
447 1. 用户管理功能测试：测试系统能否正确地实现用户账号注册、个人信息修改和找回等功能，并检查这些功能的安全性和稳定性。
448 2. 房源管理功能测试：测试系统能否支持房源的添加、修改、查询和删除等操作，并能够准确地显示房源信息，同时测试系统搜索功能是否准确返回相应搜索结果。
449 3. 租户管理功能测试：测试系统能否支持租户信息注册、修改和删除等操作，并能准确地记录租房合同信息，租金缴纳情况和租房状态信息。
450 4. 订单管理功能测试：测试系统能否自动生成订单且订单信息的准确性，订单状态处理的准确性。
451 5. 支付功能测试：测试系统是否支持多种支付渠道，能够安全地处理付款信息，管理退款信息。
452 6. 系统管理功能测试：测试系统管理功能包括系统日志记录、数据备份和恢复，网站 SEO、营销推广等部分，以及其他额外功能如系统主题切换、语言转等。
453
454 经过以上测试笔者们可以确认系统各项功能的完整性和质量，确保系统能够满足用户的需求并能够稳定运行。
455
456 == 运行界面
457 经过测试与修改以后，系统就可以正常运行了。@fig:login 显示用户进入系统的登录界面。
458 #figure(
459   caption: [登录界面],

```

```

460   image("img/login.png")
461 )
462
463 @fig:register 表示注册界面
464 #figure(
465   caption: [注册],
466   image("img/register.png")
467 )
468
469 @fig:home 表示用户的主页面
470 #figure(
471   caption: [主页面],
472   image("img/home.png")
473 )
474
475 @fig:manage 表示用户管理界面
476 #figure(
477   caption: [管理界面],
478   image("img/manage.png")
479 )
480
481 == 本章小结
482 本章主要介绍了系统测试以及最终系统运行结果。首先介绍了这个测试流程，包括资源测试和功能测试。通过
        系统实际运行页面截图，辅以文字描述对系统的最终运行结果进行了效果展示说明。
483
484 = 总 #h(1em)结
485 本篇论文主要介绍了房屋租赁管理的系统与实现，通过对系统的详细描述和体实现，展现了其在房屋租赁管理
        中的重要作用。
486
487 首先介绍了课题的背景和研究意义，指出了房屋租赁管理系统的必要性和重性。接下来，本文分析了当前房屋
        租赁行业的现状和问题，并提出了本文设计的系统解决问题的方法。房屋租赁管理设计和实现基于 Java，使
        用 SpringBoot 开发框架进行开发，并采用 MySQL 数据库进行数据管理。通过对用户需求的分析和系统功
        能的划分，本文实现了系统的基础功能和高级功能，包括房屋信息管理、租客信息管理等核心模块。
488
489 在具体实现中，采用了 MVC 模式，将系统的业务逻辑、视图呈现和数据处理相互分离，从而提高了系统的可
        重用性、可扩展性和可维护性。
490
491 最后通过对本文设计和实现的总结，可以得出如下结论：
492 1. 房屋租赁管理系统的有效设计和实现能够提高房屋租赁行业的管理效率和服务质量，满足用户需求，为用户带来
        更好的服务体验。
493 2. 本文所采用的设计方法和技术手段是有效的，被应用于类似的管理系统的开发中，并具一定的普适性。
494 3. 本文的研究仍然存在一些不足之处，例如对用户需求的分析不够深入，系统安全性的处理还完善等，这些
        问题需要在今后的研究中得到进一步的完善和改进。
495 综上所述，本文的研究为房屋租赁管理系统的有效设计和实现提供了一定的参考和借鉴价值，也为相关领域的研究者
        提供了一点微薄之力。
496
497 #set heading(numbering: none)
498 = 致 #h(1em)谢
499 在笔者的本科生涯即将结束之际，笔者不禁回首往事，深感时光如梭。这些年来有许多人和事让笔者受益匪
        浅，使得笔者度过了充实而难忘的四年。在这篇毕业论文致谢中，笔者要对他们表达真情实感的感激之意。
500
501 首先，笔者要感谢笔者的父母，在笔者茁壮成长的过程给予了笔者无尽的关爱和支持。他们默默无闻地为笔者
        付出，从小到大笔者一直被他们宠了。他们为了笔者能够顺利完成学业，所有的膳食语出等供应都是最好的，
        虽然这些都可能已成为家长义务但是，对于笔者感恩永存。
502

```

503 其次笔者要感谢笔者的导师卓能文老师，他是一个温暖、善良且富有智慧的人。他为笔者们的实项目指明了前进方向，耐心细致地给予指导和帮助。在研究过程中遇到困难和障碍时，他会及时指出问题所在并提出建设性意见。虽然笔者只是一名徒弟，但他从始至终给笔者的关怀和帮助让笔者受益匪浅。

504

505 再次，笔者要感谢笔者的同学们，你们是笔者路走来的陪伴者。在学习中若遇到困难，你们会及时贡献自己的智慧和时间来帮助笔者解决问题。在生活每次共同的经历也能让笔者感到非常快乐。与你们在一起的时光不仅是笔者难忘的回忆，也是让笔者成长的催化剂。笔者将永远怀念笔者们一起度过的岁月。

506

507 此外，笔者还要感谢吉利学院为笔者提供的良好学习和交流环境。无论从课程设置、图书馆资源和学术研究都给了笔者很大的帮助。在这里，笔者收获了知识和思维方式，找到了未来职业发展的方向。

508

509 最后，笔者还要感谢笔者的朋友们，谢谢你们的建议和支持。你们用你们的话语温暖笔者，在笔者孤独时在笔者身边陪伴，让笔者感到自己是个幸运的人。

510

511 总之，笔者的日记本永远记录着与你们共进退的日子，未来笔者们无论在世界的哪个角落，未来笔者们依然并行不悔。上天在笔者人生路上给予笔者许多的善意希望笔者们能够再次相遇在人生旅途的某处。感谢你们，祝愿笔者们回首枯藤老树时，仍是少年郎！

512

```
513 #counter(heading).update(0)
514 #set heading(numbering: "A.1")
515 = 附录
516
517 您可在 #link("https://github.com/soarowl/geelypaper.git") 检查最新代码，或提 PR。
518
519 == 论文模板
520 #code(raw(read("paper.typ"), lang: "typc"), lang: "typst")
521
522 == 本文代码
523 #code(raw(read("paper_demo.typ"), lang: "typc"), lang: "typst")
524
525 #bibliography("basic.yml", style: "gb-7714-2015-numeric")
526
```

## 参考文献

- [1] GÜNTHER-HAUG B. Den Boden unter den Füßen verlieren[M]. München: MVG, 2020.
- [2] authoritative[Z/OL]. [2020-11-29]. <https://dictionary.cambridge.org/dictionary/english/authoritative>.
- [3] DONNE J. The "Anniversaries" and the "Epicedes and Obsequies": 卷 6[M]. Bloomington: Indiana University Press, 1995.
- [4] STEYERL H. Drill[Z]. 2019.
- [5] Freedom of Information Act: Pub. L. No. 107-296, 80 Stat. 250[A]. 1967.