河北水利電力学院

本科毕业设计

基于 Spring Boot 的大学生宿舍自选管理系统的设计与实现

Design and Implementation of A Self Selected Management System for College Students' Dormitory Based on Spring Boot

姓　　名　刘仕恒

学　　号　1008519070310

院　　系　计算机系

专　　业　软件工程

指导教师　崔晶

二〇二三 年 六 月 九 日

毕业设计原创性声明

本人所提交的毕业设计《基于 Spring Boot 的大学生宿舍自选管理系统的设计与实现》，是在导师的指导下，独立进行研究工作所取得的原创性成果。除文中已经注明引用的内容外，本毕业设计不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中标明。

本声明的法律后果由本人承担。

毕业设计作者（签名）：　　指导教师确认（签名）：

年　月　日　　年　月　日

毕业设计版权使用授权书

本毕业设计作者完全了解河北水利电力学院有权保留并向国家有关部门或机构送交毕业设计的复印件和磁盘，允许毕业设计被查阅和借阅。本人授权河北水利电力学院可以将毕业设计的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其它复制手段保存、汇编毕业设计。

保密的毕业设计在_____年解密后适用本授权书。

毕业设计作者（签名）：　　指导教师（签名）：

年　月　日　　年　月　日

摘　要

本文主要目的是设计一个大学生宿舍自选管理的系统，重点实现自选功能。随着现代社会的不断发展，大学生的生活和管理需求也愈加多元化和复杂化，传统的管理方式已无法满足大学生的需求，所以加强大学生宿舍自主权，增加大学生对宿舍管理的参与度和高校后勤处管理方便性是非常必要的。本系统有大学生在线自选宿舍、报修申请、管理员管理等功能。其中，自选是本系统的核心特点，该自选功能由超级管理员为预选宿舍的班级分配预选宿舍，并由大学生在本系统中进行宿舍的选择。

大学生宿舍自选管理系统是一款前后端分离的项目，前端运用 LayUI 框架完成页面的布局设计，后端使用 Spring Boot 框架进行开发，数据存储方面采用 MySQL 作为数据支持。使用 Redis 缓存数据。通过需求分析、系统设计、和系统实现等阶段实现大学生宿舍自选管理系统的开发，这对于大学生宿舍管理的改革和提升具有一定的参考价值。

关键词：大学生；宿舍自选；前后端分离

ABSTRACT

The main purpose of this article is to design a self selection management system for college students' dormitories, with a focus on achieving the self selection function. With the continuous development of modern society, the living and management needs of college students have become increasingly diverse and complex. Traditional management methods can no longer meet the needs of college students. Therefore, it is necessary to strengthen the autonomy of college students' dormitories, increase their participation in dormitory management, and facilitate the management of university logistics offices. This system has functions such as online dormitory selection, repair application, and administrator management for college students. Among them, self selection is the core feature of this system. This self selection function is assigned by super administrators to pre selected dormitories for classes, and college students choose dormitories in this system.

The college student dormitory self selection management system is a front-end and back-end separated project. The front-end uses the LayUI framework to complete the layout design of the page, and the back-end uses the Spring Boot framework for development. In terms of data storage, MySQL is used as data support. Use Redis to cache data. The development of a college student dormitory self selection management system through requirements analysis, system design, and system implementation stages has certain reference value for the reform and improvement of college student dormitory management.

Key words: College students; Dormitory self selection; Front and rear end separation

目　　录

第 1 章　绪论

1.1　课题背景

大学生宿舍是学生学习和生活的关键场所之一，在高校管理工作中也极为重要。不过，传统的宿舍管理模式存在不少问题[1]。学生宿舍管理信息难以及时公开、传达、反馈等等。然而在这种方式下，大学生的个性化需求往往得不到充分的考虑，难以满足大学生自行选择宿舍等需求[2]。因此，为了提高宿舍管理的高效化、个性化和自主化程度，本文以大学生宿舍为背景，设计和开发一款对大学生宿舍自行选择和宿舍管理的宿舍自选管理系统。该系统的核心功能是为大学生提供自主、便捷、高效的自选体验[3]。

如此一来，为了提高大学生宿舍的管理效率和生活品质，宿舍管理应该充分了解学生的需求和个性化要求，实现“以人为本”的宿舍管理目标[4]。

1.2　目的和意义

1.2.1　课题目的

大学生宿舍自选管理系统是一款面向大学校园的宿舍管理的系统，主要为学生提供宿舍自主选择、离宿、报修和管理等服务，同时也提供管理员进行宿舍管理的功能。该系统的优点在于，它可以为学生提供自主、便捷，高效的自选体验，同时提高宿舍管理的效率和透明度，让学生和管理员之间的沟通更加便捷。大学宿舍是所有大学生在校时间里都要经历的一大部分，在大学生的日常生活和学习起到了至关重要的作用。因此，大学生宿舍管理系统对于大学校园管理具有一定的推进作用。

1.2.2　课题意义

大学生宿舍自选管理系统的课题意义在于改进大学宿舍管理的方式，提高宿舍管理的效率和透明度，从而为学生提供更好的宿舍管理体验[5]。具体来说，这个系统的意义和价值大概表现在以下几个方面：

（1）提升宿舍管理的效率：传统的宿舍管理方式其实存在各类信息不透明、相关流程繁琐等问题，而大学生宿舍自选管理系统则可以实现宿舍信息、学生信息、宿舍选择、报修申请的快速处理和更加及时的反馈，从而提升宿舍管理的效率[5]。

（2）提升宿舍管理的透明度：学生可以在系统上直观地了解宿舍床位状况、空床位信息对于管理员，通过该系统可以及时了解学生宿舍、报修、离宿记录等方面的信息，以加强管理工作，提高透明度。

（3）促进宿舍生活的改善：将宿舍管理数字化、可视化，可以加强学生对自身宿舍生活的管理和把控。同时，超级管理员和宿管员可以在系统中了解宿舍状况，定位和解决管理问题，因此会更好地保障学生宿舍生活本身的品质的提升和需求[6]。

（4）增加学生参与度和满意度：通过大学生宿舍自选管理系统，学生参与宿舍管理的过程变得更加主动与便捷，他们可以自主选择合适的宿舍、申请宿舍报修等，从而提高他们参与宿舍管理的积极性和满意度[7]。

基于以上意义，大学生宿舍自选管理系统可以更好地推进大学校园管理的数字化与智能化，为未来的校园管理提供创新思路和借鉴。

1.3 国内外现状

目前，大学生宿舍自选管理系统已经成为宿舍管理领域的一个热门课题，国内外已经涌现出不少相关的研究。以下是目前国内外现状的概述：

（1）国内现状：国内的相关研究主要聚焦于大学宿舍管理中应用信息技术和数字化技术的实际应用与推广。这些研究从不同角度和层面对大学生宿舍自选管理系统的开发和实践问题进行了深入探讨，包括系统架构设计、用户需求分析、数据信息和数据处理等方面[8]。

（2）国外现状：国外方面，美国、欧洲等地区的大学已经广泛应用数字化和智能化技术进行宿舍管理，比如在宿舍投资、床位管理、维修管理等方面开发和使用了大量的软件和应用程序。其中一些系统可以为学生提供更个性化的选择和服务，同时也可以帮助管理员更好地加强宿舍监管和管理[9]。

总的来说，国内外相关研究都反映出数字化和智能化技术在大学宿舍管理中的重要性和应用前景。当前，随着科技的不断进步和互联网技术应用的广泛推广，数字化和智能化的宿舍管理势必会成为未来高校宿舍管理的必然趋势[10]。

1.4 本文主要内容

调查大学生对于宿舍管理系统的需求，包括在线选宿舍、报修申请、公告查看等方面的需求，为设计宿舍自选管理系统提供依据。

根据需求分析的结果,设计宿舍自选管理系统的功能模块及选择其相关技术，并进行系统实现。然后对大学生宿舍自选管理系统进行测试，检查其性能、稳定性、安全性、易用性等方面的表现。

1.5 本文结构

结合大学生宿舍自选管理系统的需求分析，进行系统设计和开发，直到最后实现大学

生宿舍自选管理系统的开发。

本文共有 6 章，内容如下：

第 1 章为绪论，阐述了大学生宿舍自选管理系统的项目的开发背景、目的意义以及国内外现状。

第 2 章是系统开发的平台与技术。通过此章确认系统使用的技术。

第 3 章是需求分析，通过此章确认系统的主要开发模块。

第 4 章为系统设计，介绍了系统的整体核心思路以及各模块和数据库的设计。

第 5 章为系统实现，主要模块的实现与核心代码的部分展示与描述。

第 6 章为系统测试，对系统进行全面测试，为系统排除已检测到的问题。

第 2 章　系统开发的平台与技术

2.1　开发和使用框架

在系统实现的过程中，系统的开发环境如表 2.1 所示。

表 2.1　开发环境

环境项	环境参数
操作系统	Windows 11
数据库可视化工具	Navicat
开发工具	IDEA
浏览器	Google Chrome

在系统实现的过程中，系统的开发框架如表 2.2 所示。

表 2.2　开发框架

前端	后端
LayUI、jQuery	Spring Boot

2.2　相关技术介绍

2.2.1　LayUI 简介

LayUI 是一个 UI 框架，具有丰富的组件和精美的界面设计，易于定制和维护。同时 LayUI 还提供了丰富的文档和社区支持，方便开发者快速上手，并且可以根据自己的需求进行修改和扩展

2.2.2　Spring Boot 简介

Spring Boot 是一款基于 Spring 框架的快速开发框架，它能够简化 Spring 应用程序的开发过程并提高开发效率[11]。Spring Boot 采用了约定优于配置的理念，通过自动化配置和快速启动来减少开发人员对 Spring 繁琐配置的依赖。Spring 框架主要分为三个部分：容器、面向切面编程、控制反转。Spring 框架可对 MyBatis 框架和 Spring MVC 框架给予支持。MyBatis 的优势是灵活以及提供接口编程[12]。

2.2.3　MyBatis 简介

MyBatis 简化了 Java 程序开发中访问关系型数据库的流程。它易于使用和配置，不需

要任何 ORM 映射类，允许开发者自定义 SQL 映射语句和参数映射[13]。

2.3 本章小结

本章介绍了大学生宿舍自选管理系统开发过程中所用到的部分相关技术，明确开发本系统采用这些框架的优势。

第 3 章　需求分析

3.1　系统目标及流程分析

3.1.1　系统目标

大学生宿舍自选管理系统是一个方便大学生在线选择宿舍的管理系统，通过该系统能够为学校提供一个便捷的宿舍管理工具，提高学生的自主性和自我管理，与此同时推进大学后勤处的发展。

3.1.2　系统主要流程分析

超级管理员要在后台添加楼房信息（包括楼层数），然后为每栋楼房设置宿舍编号，并对宿舍进行初始化（添加宿舍信息并分配宿舍编号）。之后需要为宿舍手动添加预选信息，包括班级名字、选择开始时间和结束时间。在为宿舍预选班级分配对应的宿舍后，学生就能在指定时间内进行宿舍的选择。若没有为班级设置预选宿舍信息，则不能进行宿舍选择。超级管理员负责所有的楼栋的和学生的信息管理，而宿管员只负责一栋楼的信息管理。

3.2　系统需求分析

3.2.1　功能需求分析

（1）登录功能：为了选择宿舍，学生必须先在系统中注册并登录。

（2）选择宿舍功能：学生需要在系统中线上选择可用的空床位。

（3）宿舍信息管理功能：设置管理员权限，以便管理宿舍信息，如宿舍编号、楼号、床位等

（4）学生管理功能：设置管理员权限，以便添加和删除学生信息，并查看和编辑学生的详细信息，例如姓名、手机号、性别、学号等。

（5）床位分配调整功能：超级管理员和宿管员可选择需要调整的床位进行床位的调整。

（6）宿舍报修功能：学生可以在系统中提交报修的具体描述。管理员会收到申请并进行审核。

（7）宿舍评价功能：超级管理员和宿管员可对宿舍进行评价。

（8）数据报表功能：管理端的首页展示各种报表，包括宿舍床位使用率、学生入住情况、宿舍闲置数量等。

本系统主要分为学生模块、超级管理员模块和宿管员模块。

学生模块

学生模块面向大学生群体。主要包括登录注册、首页、在线选宿舍、离宿记录、公告查看、报修申请。学生模块主要提供在线选宿舍的功能，并增加了宿舍管理的功能。

学生用例图如图 3.1 所示：

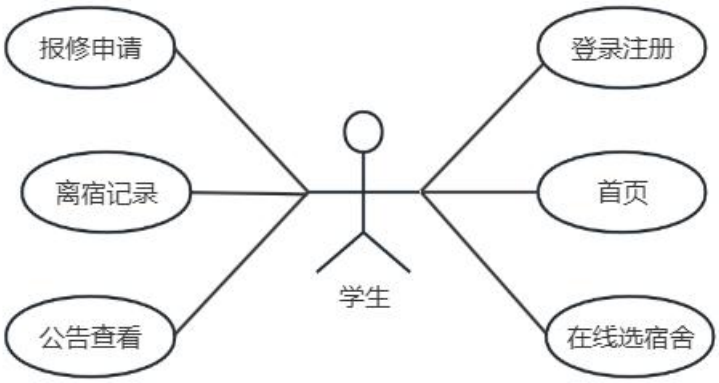

图 3.1　学生用例图

超级管理员模块

超级管理员进行后台管理，主要有登录注册、基础管理、宿舍管理、离宿管理、报修管理、公告管理、系统管理等功能。

超级管理员用例图如图 3.2 所示：

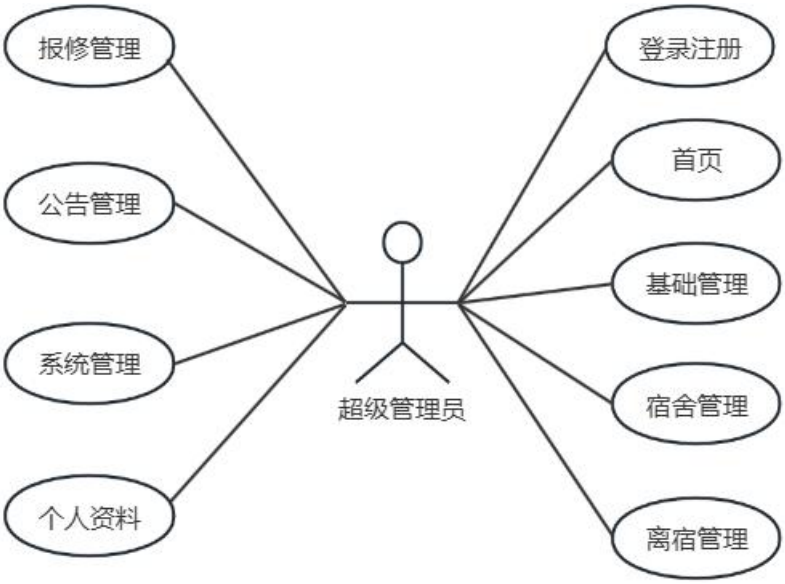

图 3.2　超级管理员用例图

宿管员模块

宿管员进行后台管理，主要功能有登录注册，首页，宿舍管理、报修管理、离宿管理、公告管理。

宿管员用例图如图 3.3 所示：

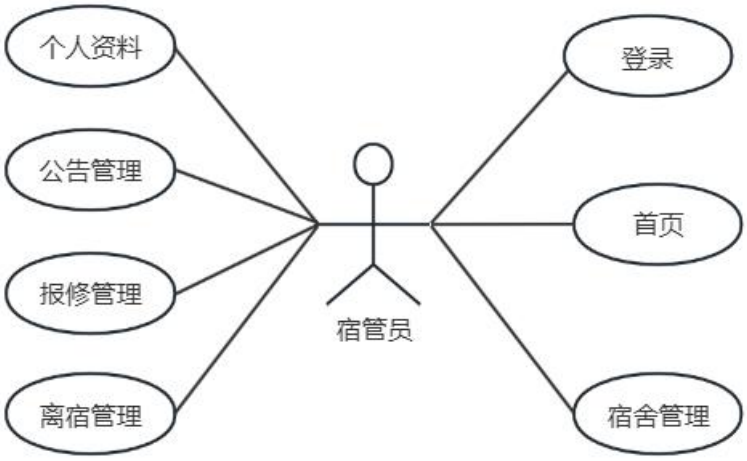

图 3.3　宿管员用例图

3.2.2　非功能性需求分析

（1）易用性

大学生自选管理系统对用户友好，具有良好的 UI 设计，易于学习、实用、安全、可进行个性化定制。

（2）技术可行性

在技术方面，大学生宿舍自选管理系统采用前后端分离的方式实现[14]。后端采用 Spring Boot 和 MyBatis 框架进行实现，数据存储使用 MySQL，数据的缓存使用 Redis，并使用接口进行前后端数据的传输[15]。以上这些技术是开源的并且已经成熟。所以，大学生宿舍自选管理系统在技术方面可行。

（3）经济可行性

大学生自主选择管理系统的运营相当方便，节约了大量人力物力，减少了大量的经济成本。与此同时，该系统的实施增强了学校的宣传力度，提高了高校的影响力，吸引更多学生到该校就读。该系统所使用的技术均来源于网络开源平台，并且开发工具也是免费的，因此并不会增加系统开发的成本。由此看来，大学生宿舍自选管理系统在经济方面可行。

（4）性能

大学生自选管理系统实现了快速响应和高吞吐量，尤其在关键流程和高峰选择时间内，保证系统的稳定性、高效性和准确性。

（5）可伸缩性

大学生自选管理系统能够应对不断增长的用户需求，应该提供有效的扩展性，以支持更多的用户并保持其顺畅运行。

通过以上的非功能性需求分析，可以得出结论：大学生宿舍自选管理系统的设计是可行的。这证明了其值得进一步开发。

3.3 本章小结

本章主要对大学生宿舍自选管理系统进行了客观需求的分析，经过深入分析，对系统的功能需求进行了探究，并且运用用例图描述了每个模块的结构，同时，本章全面分析了系统的非功能性需求，为后续的系统详细设计工作提供了重要的基础和指导。

第 4 章　系统设计

4.1　系统总体设计

该系统在用户角度下进行主体设计，包括超级管理员模块、宿舍管理员模块和学生模块。超级管理员模块用于管理用户信息、学生信息、楼宇信息、宿舍信息、宿舍编号设置、宿舍预选和宿舍分配等。学生模块则用于根据超级管理员所规划的宿舍和床位信息实施宿舍选择。此外，还设计了精美的网页样式布局和数据图表展示，以呈现出良好的视觉效果。系统功能结构图如图 4.1 所示：

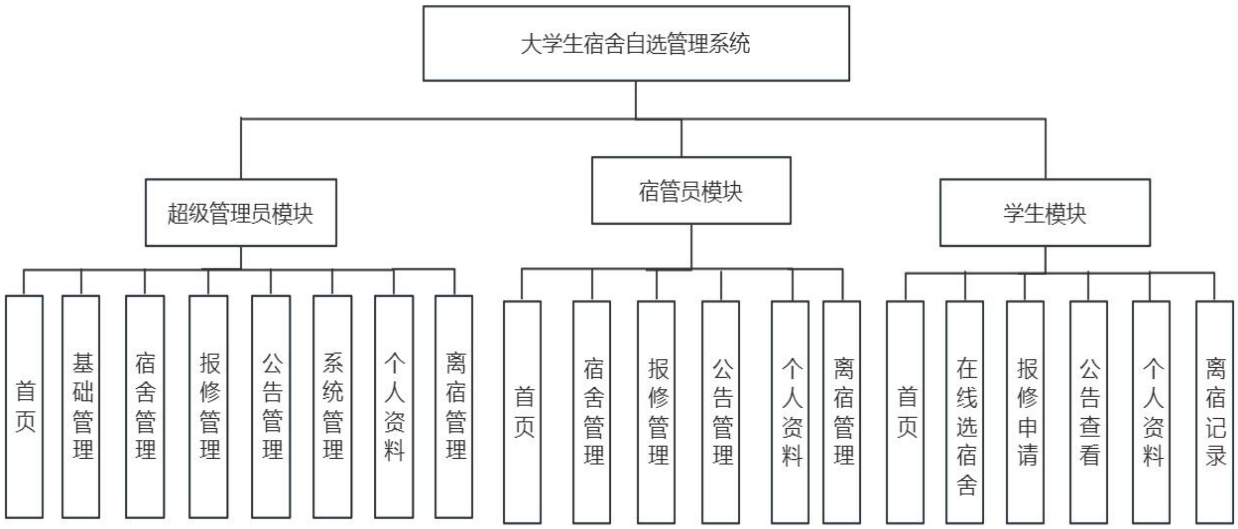

图 4.1　系统功能结构图

4.2　系统功能模块设计

本系统围绕宿舍展开，学生可在线选择宿舍，管理员可管理学生信息、宿舍信息。

4.2.1　用户登录模块设计

在大学生宿舍自选管理系统中，用户分为三类：学生、宿管员和超级管理员。用户需要在登录页面选择自己的身份，并输入正确的账号和密码。若输入的账号密码有误，系统将提示用户重新输入。登录成功后，用户将跳转到系统的首页。

大学生宿舍管理系统的用户登录流程如图 4.2 所示：

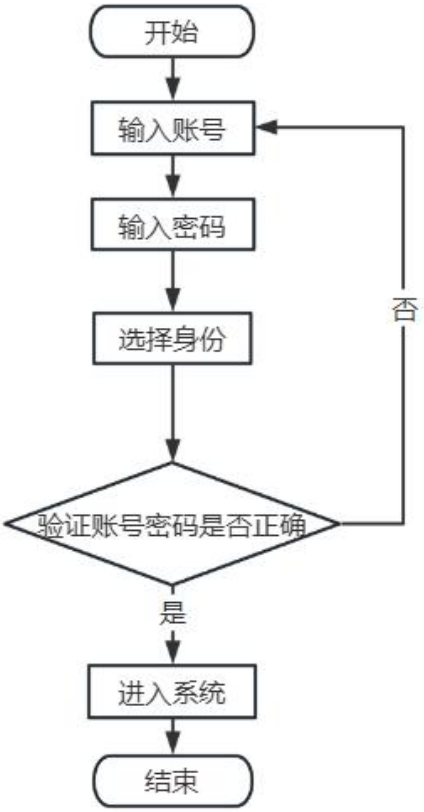

图 4.2　系统登录流程图

4.2.2　学生在线选宿舍模块设计

学生登录成功进入本系统后即可使用本系统，管理员为学生所在的班级分配了宿舍以后，学生在规定时间内登入该系统可进行宿舍的选择，学生可选择空床位，选择成功后，可再次选择，选择宿舍时间结束后，自动保存选择信息。

学生选择宿舍流程图如图 4.3 所示：

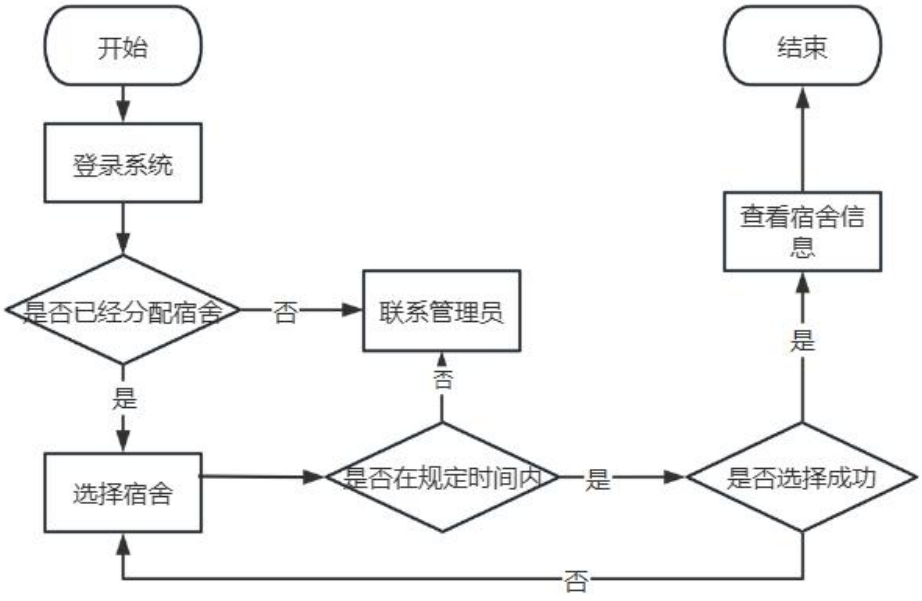

图 4.3　学生选择宿舍流程图

4.2.3 超级管理员分配宿舍模块设计

学生预想选择宿舍，超级管理员必须在后台管理系统为班级添加宿舍预选设置，并为每个班级分配对应的宿舍，其中超级管理员分配宿舍流程图如图 4.4 所示：

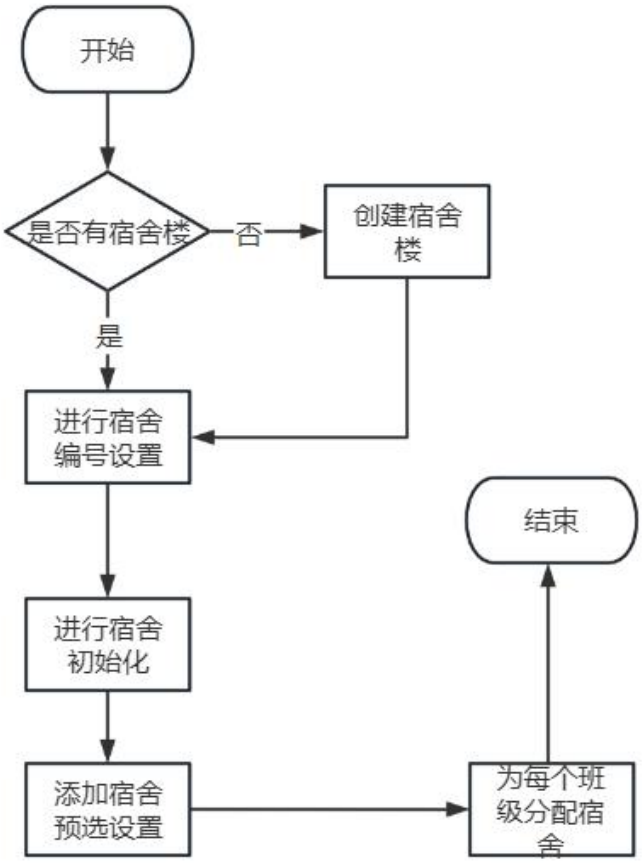

图 4.4 管理员分配宿舍流程图

4.2.4 用户管理模块设计

用户管理主要的功能是超级管理员可以新增超级管理员或者宿管员并为其添加菜单权限；宿管员不能通过注册来使用本系统，只能通过超级管理员来添加宿管员的信息。超级管理员还可以编辑和删除用户信息。用户管理功能模块划分及描述如表 4.1 所示：

表 4.1 用户管理功能模块表

模块名称	模块功能描述
添加用户	超级管理员添加用户信息
修改用户	超级管理员能修改用户信息
删除用户	超级管理员删除所选用户信息
重置密码	超级管理员可以重置用户密码

4.2.5 学生管理模块设计

学生管理模块的基本管理，主要对学生信息进行管理，其功能模块的具体划分及描述所表 4.2 所示：

表 4.2 学生管理功能模块表

模块名称	模块功能描述
添加学生信息	超级管理员添加学生信息
修改学生信息	超级管理员修改学生的信息
删除学生信息	超级管理员删除学生的信息
导入学生数据	超级管理员导入学生数据
导出学生数据	超级管理员导出学生数据
重置密码	超级管理员重置学生密码

4.2.6 宿舍编号设置模块设计

宿舍编号设置的基本管理，主要提供超级管理员对宿舍编号设置的信息增删改，初始化宿舍编号，其功能模块的具体划分及描述如表 4.3 所示：

表 4.3 楼宇管理功能模块表

模块名称	模块功能描述
添加宿舍编号设置	超级管理员添加宿舍编号设置
修改宿舍编号设置	超级管理员修改宿舍编号设置
删除宿舍编号设置	超级管理员删除宿舍编号设置

4.2.7 宿舍管理模块设计

宿舍的基本管理，本模块是本系统中最主要的模块之一，主要是对宿舍信息的展示、学生的床位选择情况的查看以及对学生床位的分配调整，其功能模块的具体划分及描述如表 4.4 所示：

表 4.4 宿舍管理功能模块表

模块名称	模块功能描述
添加楼层	超级管理员添加楼层
添加宿舍	超级管理员添加宿舍
添加床位	超级管理员添加床位
删除宿舍	超级管理员删除宿舍
删除床位	超级管理员删除床位
分配调整	超级管理员对床位进行分配调整

4.2.8 宿舍预选设置模块设计

宿舍预选设置的基本管理，该模块是为了学生选择宿舍做准备，只有为学生所在的班级进行宿舍预选设置和宿舍分配，学生才可进行宿舍的选择。其功能模块的具体划分及描述如表 4.5 所示：

表 4.5 宿舍预选设置功能模块表

模块名称	模块功能描述
添加宿舍预选设置	超级管理员添加宿舍预选设置
修改宿舍预选设置	超级管理员修改宿舍预选设置
删除宿舍预选设置	超级管理员删除宿舍预选设置
宿舍分配	超级管理员对添加宿舍预选宿舍了的班级进行宿舍分配

4.2.9 公告管理模块设计

公告管理是超级管理员添加公告信息供学生门查看，超级管理员科设置所有楼宇的公告，宿管员只能设置自己负责楼宇的公告,其功能模块划分及描述如表 4.6 所示：

表 4.6 公告功能模块表

模块名称	模块功能描述
添加公告信息	超级管理员添加公告
编辑公告信息	超级管理员编辑公告信息
删除公告信息	超级管理员删除公告信息

4.3 数据库设计

4.3.1 数据库概念结构设计

将大学生宿舍自选管理系统的数据库概念模型分为成各个实体，其中包括：管理员实体、学生实体、楼宇实体、楼层实体、宿舍实体、床位实体等。

管理员实体的属性有用户名、密码、姓名、类型等。管理员实体图如图 4.5 所示：

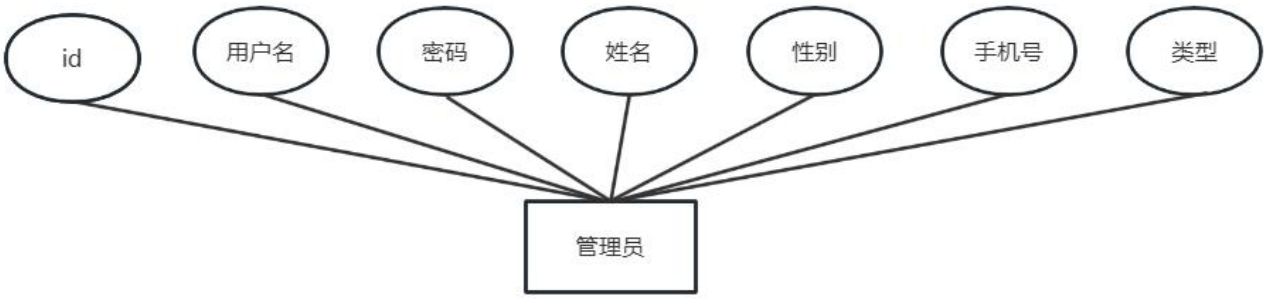

图 4.5 管理员实体图

学生实体的属性有：学号、姓名、性别、专业 id、班级 id 等。学生实体图如图 4.6 所示：

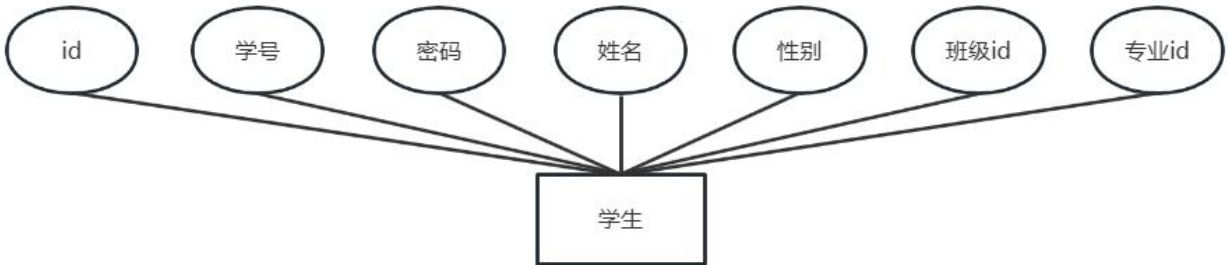

图 4.6　学生实体图

楼宇实体的属性有：名称、层数、宿管员 id。楼宇实体图如图 4.7 所示：

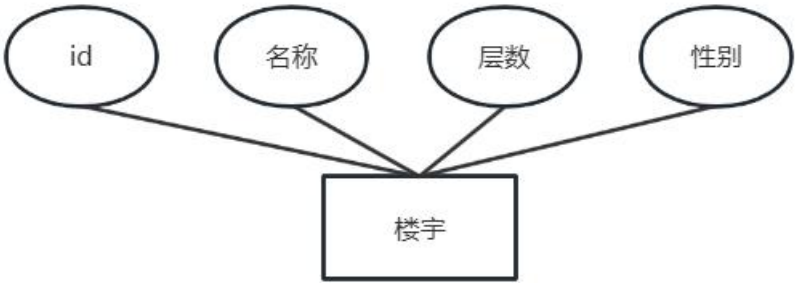

图 4.7　楼宇实体图

楼层实体的属性有：名称、楼宇 id 等。楼层实体图如图 4.8 所示：

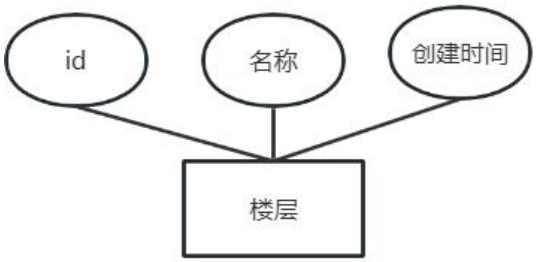

图 4.8　楼层实体图

宿舍实体的属性有：宿舍号、性别、类型、容量等。宿舍实体图如图 4.9 所示：

图 4.9　宿舍实体图

床位实体的属性有：床位号、宿舍 id 等。床位实体图如 4.10 所示：

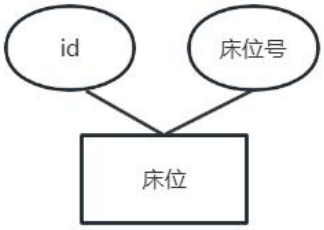

图 4.10　床位实体图

通过以上分析可以得到系统整体 E-R 图，如图 4.11 所示：

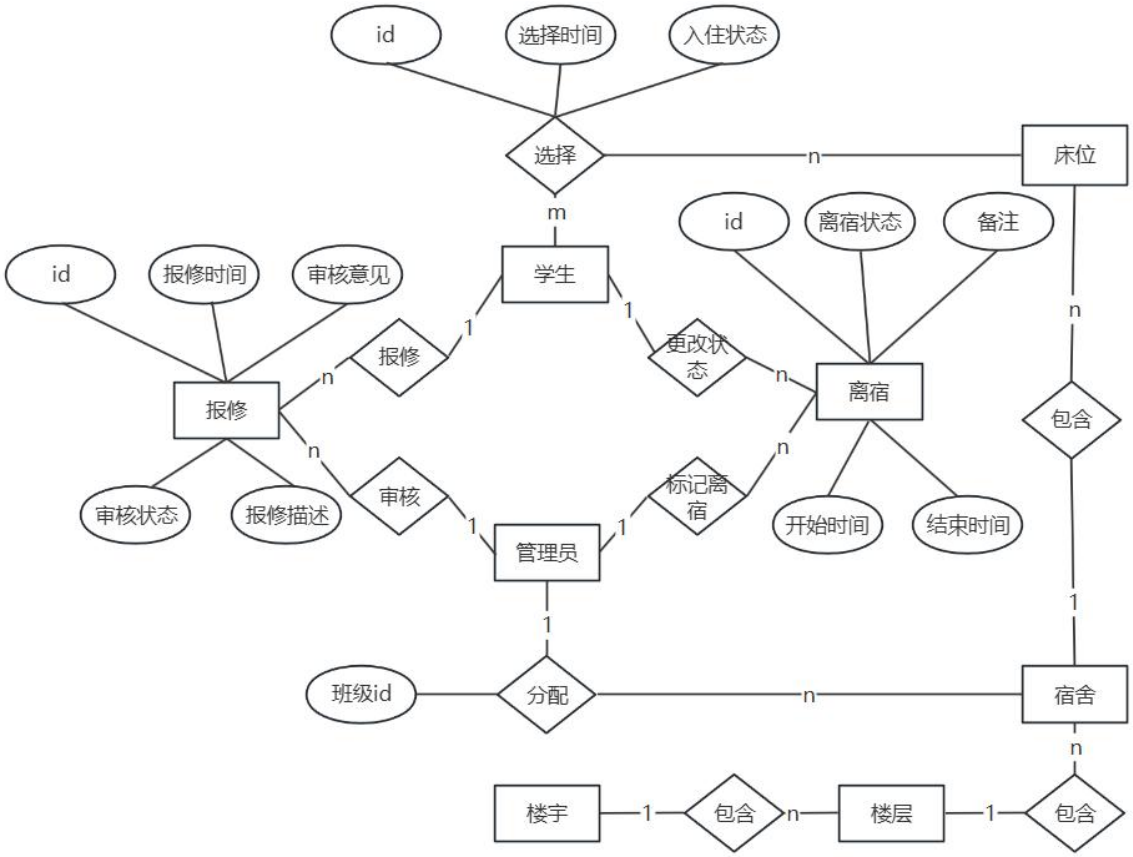

图 4.11　系统 E-R 图

4.3.2　数据库逻辑结构设计

本系统数据库包括：学生信息表（tb_student）、用户信息表（tb_user）、年级表（tb_grade）、组织结构表（tb_org）、楼宇表（tb_building）、楼层表（tb_storey）、宿舍表（tb_dormitory）、床位表（tb_bed）、宿舍编号设置表（tb_dormitory_set）、宿舍学生关联表（tb_dormitory_student）、宿舍预选表（tb_selection）、预选宿舍班级关联表（tb_selection_dormitory）、预选班级关联表（tb_selection_joiner）。

各表单的数据结构如下所示：

（1）学生表：学生表存储学生学号、姓名、身份证号等信息。如表 4.7 所示：

表 4.7　学生信息表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	学生 id
sto_no	varchar	20	否	学号
name	varchar	20	否	姓名
id_card	varchar	50	否	身份证号
sex	int	10	否	性别
phone	varchar	50	否	学生手机号
password	varchar	50	否	密码
clazz_id	Int	10	否	学生所在班级 id
file	varchar	100	否	头像名称

（2）用户信息表：用户信息表存储用户 id、用户名、密码等信息。如表 4.8 所示：

表 4.8　用户信息表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	用户 id
user_name	varchar	50	否	用户名
password	varchar	20	否	密码
phone	varchar	20	否	手机号
name	varchar	20	否	姓名
type	int	10	否	用户类型
status	int	10	否	用户状态
file	varchar	50	否	头像名称

（3）年级表：年级表存储年级 id、年纪名称、类型等信息。如所表 4.9 所示：

表 4.9　年级表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	年级 id
name	varchar	50	否	年级名称

（4）组织结构表：组织结构表存储组织 id、组织名称、类型等信息，如表 4.10 所示：

表 4.10　组织结构表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	组织 id
name	varchar	50	否	组织名称
type	int	10	否	类型（1-学院/2-系/3-专业/4-班级）

续表 4.10　组织结构表

字段名	数据类型	大小	是否主键	描述
grade_id	int	10	否	年级 id
parent_id	int	10	否	父级组织 id
status	int	10	否	状态
deleted	int	10	否	删除标示

（5）楼宇表：楼宇表存储楼宇 id、层数、性别等信息，如表 4.11 所示：

表 4.11　楼宇表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	楼宇 id
storey_num	int	10	否	层数
sex	int	10	否	性别（0-女/1-男）
user_id	int	10	否	宿管员 id
remark	varchar	100	否	备注

（6）楼层表：楼层表存储楼层 id、楼层名称、创建时间等信息，如表 4.12 所示：

表 4.12　楼层表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	楼层 id
name	varchar	50	否	名称
building_id	int	10	否	楼宇 id
create_time	datetime		否	创建时间

（7）宿舍表：宿舍表存储宿舍 id、宿舍号、性别、类型等信息，如表 4.13 所示：

表 4.13　宿舍表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	宿舍 id
no	varchar	50	否	宿舍号
sex	int	10	否	性别（0-女/1-男）
type	int	10	否	类型（4/6/8 人间）
capacity	int	10	否	容量
storey_id	int	10	否	楼层 id
building_id	int	10	否	楼宇 id
version	int	10	否	版本号

（8）床位表：床位表存储床位相关信息，如表 4.14 所示：

表 4.14　床位表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	床位 id
bno	varchar	10	否	床位号
dormitory_id	int	10	否	宿舍 id

（9）宿舍编号设置表：宿舍编号设置表宿舍编号设置的相关信息，如前缀、开始值、结束值等，如表 4.15 所示：

表 4.15　宿舍编号设置表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	宿舍编号设置 id
prefix	varchar	10	否	前缀
start	int	10	否	开始值
end	int	10	否	结束值
building_id	int	10	否	楼宇 id
storey_id	int	10	否	楼层 id
capacity	int	10	否	容量

（10）宿舍学生关联表：宿舍学生关联表存储宿舍和学生的关联的相关信息，如表 4.16 所示：

表 4.16　宿舍学生关联表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	关联 id
bed_id	int	10	否	床位 id
dormitory_id	int		否	宿舍 id
student_id	int	10	否	学生 id
checkin	datetime		否	选择时间
status	int	4	否	状态（0-待入住/1-已入住）

（11）宿舍预选表：宿舍预选表存储宿舍预选的相关信息，如开始时间、结束时间等，如表 4.17 所示：

表 4.17　宿舍预选表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	宿舍预选 id
name	varchar	100	否	名称
start_time	datetime		否	开始时间
end_time	datetime		否	结束时间
remark	varchar	100	否	备注

（12）预选宿舍班级关联表：预选宿舍班级关联表存储预选宿舍班级关联的相关信息，如表 4.18 所示：

表 4.18　预选宿舍班级关联表

字段名	数据类型	大小	是否主键	描述
id	int	4	否	id
dormitory_id	int	10	否	宿舍 id
clazz_id	int	10	否	班级 id

（13）预选班级关联表：预选班级关联表存储预选 id、班级 id 等信息，如表 4.19 所示：

表 4.19　预选班级关联表

字段名	数据类型	大小	是否主键	描述
id	int	4	是	关联 id
selection_id	int	10	否	预选 id
clazz_id	int	10	否	班级 id

4.4　本章小结

本章主要概括了系统的总体设计，分别从数据库概念结构设计、数据库物理结构设计、两个方面对本系统数据库进行详细介绍，设计了系统所需的实体图和 E-R 图，最后对大学生宿舍自选管理系统的数据进行完善。

第 5 章　系统实现

5.1　功能模块设计实现

5.1.1　用户登录模块实现

用户登录的界面如图 5.1 所示：

图 5.1　登录页面展示

用户输入用户名和密码，选择身份后登入。

代码是通过 login 方法实现，后端接收前端传来的 userName、password 和 type 用户；类型，进行判断是否登录成功。代码如图 5.2 所示：

```
Student entity = studentService.login(user.getUserName(), Md5Utils.getMd5(user.getPassword()));
if (entity != null) {
    String token = JWTUtil.signForStudent(entity);
    Map map = new HashMap();
    map.put(JWTUtil.token, token);
    map.put("student", entity);

    redisCache.setCacheObject("student", JSONUtil.toJsonStr(entity));
    return Result.ok( msg: "登陆成功", map);
```

图 5.2　登录后端代码

5.1.2 学生模块首页实现

本系统的学生首页由两个部分组成，包括宿舍概览和当前宿舍选择。该页面在main.html 中实现。宿舍概览使学生在选择完宿舍后清晰准确地查看所选宿舍的相关信息。当前宿舍选择则可让大学生浏览其所选宿舍的成员信息。

该页面展示如下图 5.3 所示：

图 5.3 学生模块首页展示

使用的是后端的 selectRoommate 方法，根据学生的 id 查询出学生所在的宿舍，然后得到宿舍的 id，在根据宿舍 id 查出选择该宿舍的学生，然后返回给前端数据。代码如图 5.4 所示：

```
DormitoryStudent dormitoryStudent =
        dormitoryStudentService.getOne(new LambdaQueryWrapper<DormitoryStudent>()
                .eq(DormitoryStudent::getStudentId, studentId));
if (dormitoryStudent == null) {
    return Result.ok( msg: "暂无"); }
Integer dormitoryId = dormitoryStudent.getDormitoryId();
//根据宿舍id查出该宿舍的所有学生
List<DormitoryStudent> studentList =
        dormitoryStudentService.list(new LambdaQueryWrapper<DormitoryStudent>()
                .eq(DormitoryStudent::getDormitoryId, dormitoryId));
```

图 5.4 学生模块首页后端代码

5.1.2 学生在线选宿舍页面实现

学生在线选宿舍页面分为两部分，即宿舍信息和床位信息。此页面在 stu/select.html 中编写。宿舍信息包括楼栋、楼层和宿舍号码等基本信息，以及可住人数和已入住人数等情况。床位信息包括床号和可供选择的复选框。

学生在线选宿舍页面展示如图 5.5 所示：

图 5.5　在线选宿舍页面展示

虽然本系统中有很多重要的页面，但在线选宿舍的页面是其中之一。该页面的主要目的是让学生能够基于学校分配的宿舍自由选择。页面清晰地展示了宿舍和床位的信息。

选择床位后端代码如图 5.6 所示：

```
@PostMapping("/selectDormitorySubmit")
public Result selectDormitorySubmit(@RequestBody Map<String, String> map) {
    Student param = UserHolder.getStudent();
    Student student = studentService.getStudent(param.getId());
    String bedId = map.get("bedId"); String dormitoryId = map.get("dormitoryId");
    List<Selection> selections = selectionService.selectByClazzId(student.getClazzId());
    if (selections == null || selections.size() == 0) {
        return Result.fail( msg: "操作失败，未设置！请联系管理员");  }
    Selection selection = selections.get(0);
    if (selection.getStartTime().getTime() > System.currentTimeMillis()
            || System.currentTimeMillis() > selection.getEndTime().getTime()) {
        return Result.fail( msg: "操作失败，不在时间段内选择");}
    Integer sex = student.getSex(); //查询登录学生的性别
    Dormitory dormitory = dormitoryService.getById(dormitoryId);
    if (!Objects.equals(sex, dormitory.getSex())) {
        return Result.fail( msg: "请选择正确的宿舍");
    }int row;  try {
        row = dormitoryStudentService.selectDormitorySubmit(student.getId(), Integer.par
            if (row == 0) {   return Result.fail( msg: "选择失败，请稍后重试"); }
    } catch (Exception e) {
        return Result.fail( msg: "版本冲突！,请稍后再试");
    } return Result.ok( msg: "选择成功！");
```

图 5.6　学生选择床位后端代码

超级管理员可以通过添加预选配置来规划哪些班级在何时选择宿舍。为每个预选宿舍的班级分配宿舍，然后该班级的学生即可登录该系统进行宿舍选择。但是学生需要留意选择宿舍的时间，一旦时间过期，就不能再进行选择。

宿舍预选页面展示如图 5.7 所示：

图 5.7　宿舍预选页面展示

分配宿舍页面如图 5.8 所示：

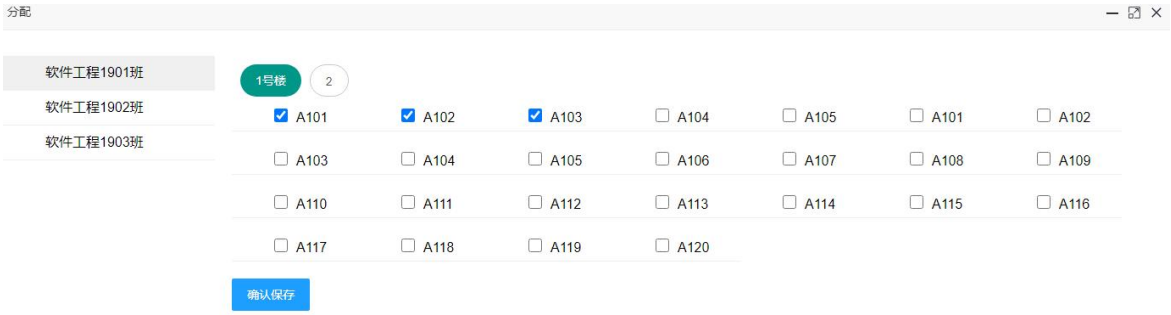

图 5.8　分配宿舍页面展示

后端部分代码是通过前端传来的班级 id 和宿舍 ids 循环来进行分配，如图 5.9 所示：

```
public int saveSelectionDormitory(String clazzId, String dormitoryIds) {
    String[] ids = dormitoryIds.split( regex: ",");
    for (String id : ids) {
        if (!StringUtils.isEmpty(id)) {
            SelectionDormitory selectionDormitory = new SelectionDormitory();
            selectionDormitory.setClazzId(Integer.parseInt(clazzId));
            selectionDormitory.setDormitoryId(Integer.parseInt(id));
            selectionDormitoryMapper.saveSelectionDormitory(selectionDormitory);
        }
    }return 1;
}
```

图 5.9　宿舍分配后端部分代码

5.1.3　学生离宿记录页面实现

该页面由一个表格组成，表格有姓名，宿舍，开始时间，结束时间，状态，备注几个属性。清晰的展示了学生的离宿记录。学生还可更改离宿记录，方便学生在回宿后进行登记。

学生离宿记录页面如图 5.10 所示：

图 5.10　学生离宿页面展示

学生的离宿记录管理作用在于对当天晚上离宿的学生的管理。其由超级管理员或者宿管员添加。也可对学生的离宿记录进行编辑和删除。

学生离宿页面后端代码是通过根据学生的 id 查出该学生的所有离宿舍记录，然后通过循环将学生的宿舍信息赋值到 Absence 中。代码如图 5.11 所示：

```
@PostMapping("/selectAbsence")
public Map<String, Object> selectAbsence(@RequestBody Absence absence, HttpServletRequest request) {
    Student param = (Student) request.getAttribute( s: "student");
    absence.setStudentId(param.getId());
    PageInfo<Absence> pageInfo = absenceService.queryByPage(absence);
    pageInfo.getList().forEach(entity -> {
        Student detail = studentService.getStudent(entity.getStudentId());
        entity.setStudent(detail);
        LambdaQueryWrapper<Dormitory> queryWrapper = new LambdaQueryWrapper<>();
        queryWrapper.eq(Dormitory::getId, entity.getDormitoryId());
        Dormitory dormitory = dormitoryService.getOne(queryWrapper);
        entity.setDormitory(dormitory);
    });
    return Result.ok(pageInfo);
}
```

图 5.11　学生离宿后端部分代码

5.1.4　超级管理员首页页面实现

超级管理员首页界面由 4 个部分组成的，其中包括数据统计、系统公告、报表统计、使用率统计。数据统计模块是一个表格，展示楼宇、宿舍数量、入住人数、闲置数量、使用率的数据。而报表统计是对这几个数据呈折线图展示，更加形象。其中宿舍使用率更是使用了饼图对其展示。用户很清晰的就可以看到几栋楼的使用率的情况。用户体验好。系统公告模块是对发布的所有的公告的一个展示。

超级管理员首页所图 5.12 所示：

图 5.12　超级管理员首页展示

5.1.5　用户管理页面实现

用户管理页包括用户列表页面添加用户页面、和用户编辑页面，分别在 user/list.html，user/add.html，user/edit.html 中编写。

用户管理页如图 5.13 所示：

图 5.13　用户管理页面展示

超级管理员可以在管理后台添加该系统的用户，新增管理后台用户的时候可以给用户选择超级管理员或者宿管员。

添加用户页面如图 5.14 所示：

图 5.14　添加用户页面展示

用户管理页面后端部分代码是通过前端传来的 User 信息，进行插入操作。代码如图 5.15 所示：

```
@PostMapping("/addUser")
public Result addUser(@RequestBody User user) {
    List<Integer> menuIds = user.getMenuIds();
    boolean flag = false;
    flag = userService.addUser(user);
    user = userService.getUser(user);
    for (Integer menuId : menuIds) {
        flag = userMenuService.saveUserMenu(user.getId(), menuId);}
    if (flag) { return Result.ok( msg: "新增用户成功！"); }
    return Result.fail( msg: "新增用户失败！");
}
```

图 5.15　用户管理页面后端部分代码

5.1.6 机构管理页面实现

机构管理页面包括组织列表页面、添加组织页面和编辑组织页面。分别在 org/list.html、org/add.html 和 org/edit.html 中编写。

机构管理页面如图 5.16 所示：

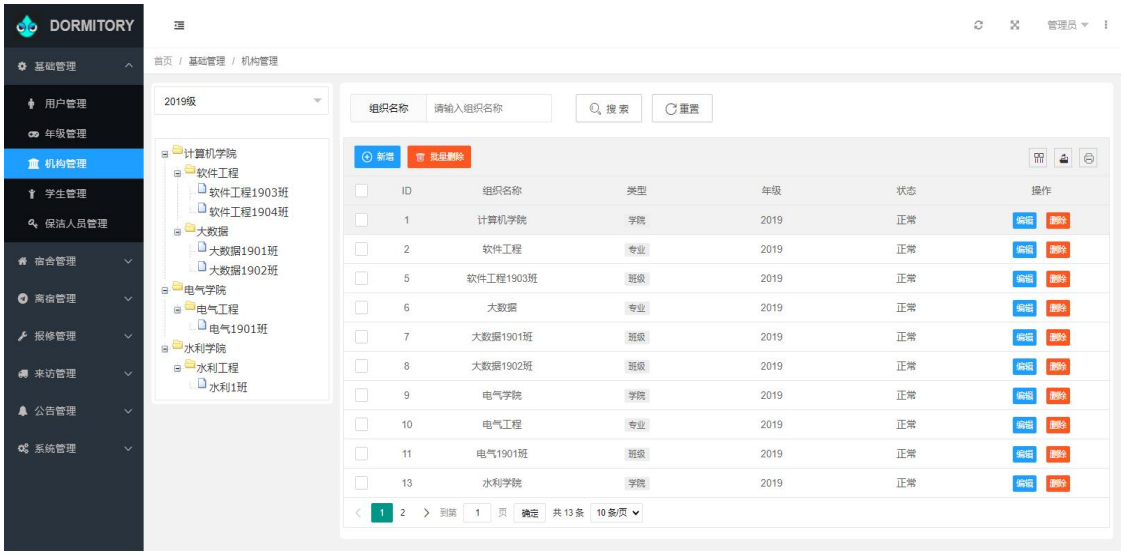

图 5.16 机构管理页面展示

超级管理员可以在机构管理页面查看结构列表，由树形结构，也有表格结构。两种方式呈现，展示更加丰满。可以根据年级选择对应年级的结构列表，可以根据机构名称搜索相应的机构。

添加机构页面如图 5.17 所示：

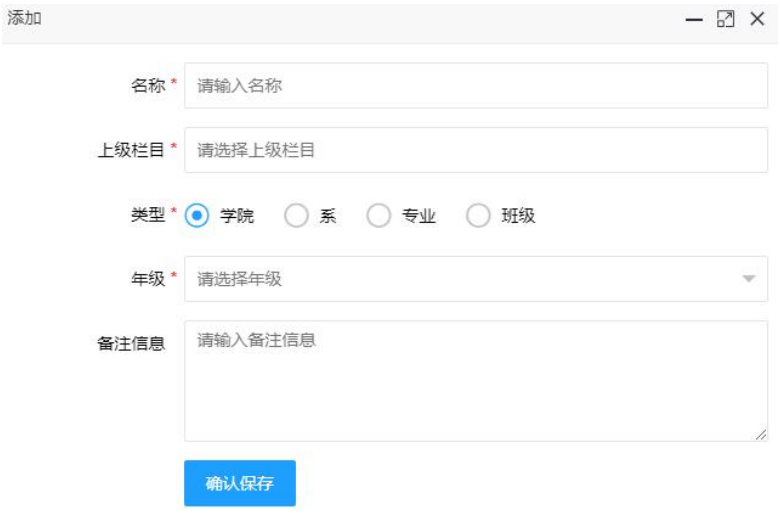

图 5.17 机构管理添加页面展示

机构管理前端部分代码如图 5.18 所示：

```
function loadTree(){
    //默认展示gradeId为1
    axios.post('org/tree',{gradeId:1,limit:10000}).then(function (response) {
        ztree.init($("#ztree"), setting,response.data);
    }).catch(function (error) {
        console.log(error);
    });
}
```

图 5.18　机构管理页面前端部分代码

后端主要代码是通过选择的栏目的 parentId 是否为 0 来判断该节点是否为父节点。代码如图 5.19 所示：

```
for (Org entity : list) {
    if (entity.getParentId() == 0) {
        Map<String, Object> map = new HashMap<>();map.put("id", entity.getId()); map.put("name", entity.getName());
        if (entity.getType() < 4) {
            map.put("isParent", true); map.put("open", true);map.put("children", getChild(entity, list));
        } else {
            map.put("isParent", false);
        } trees.add(map);
```

图 5.19　机构管理页面后端部分代码

5.1.7　学生管理页面实现

学生管理页面由学生列表页面、添加学生页面、编辑学生页面组成。分别在 student/list.html、student/add.html、student/edit.html 页面编写。学生管理页面如图 5.20 所示：

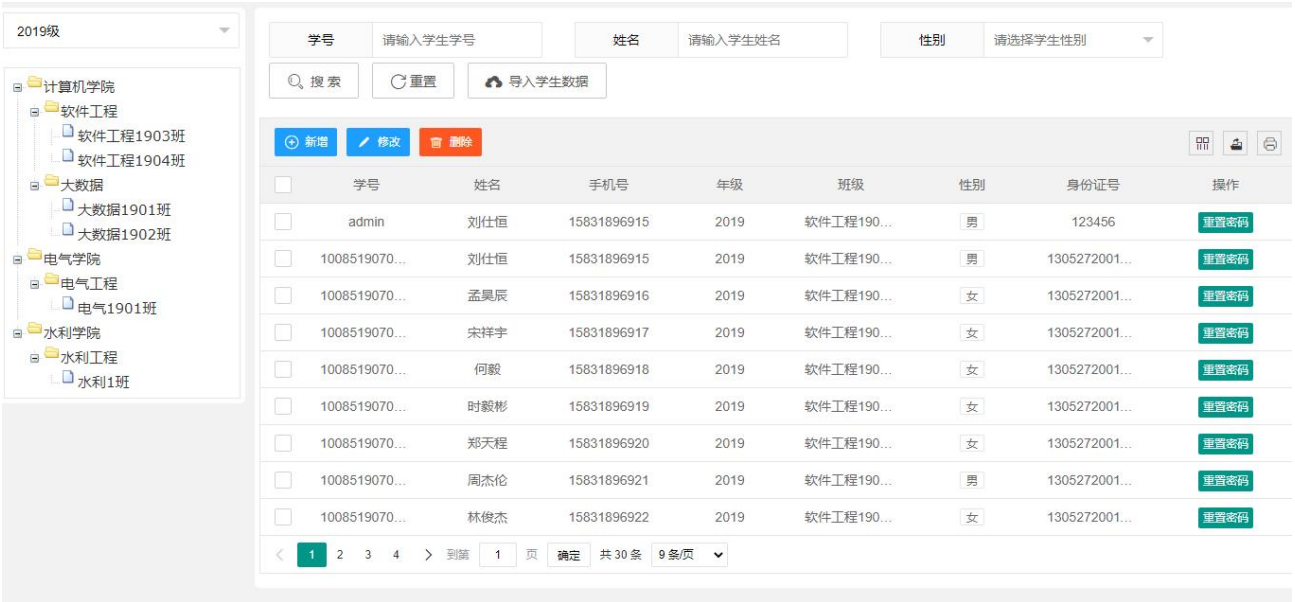

图 5.20　学生管理页面展示

超级管理员在该页面可以查看学生的列表信息，跟机构管理相似，也有表格结构跟树形结构两种。添加学生的页面如图 5.21 所示：

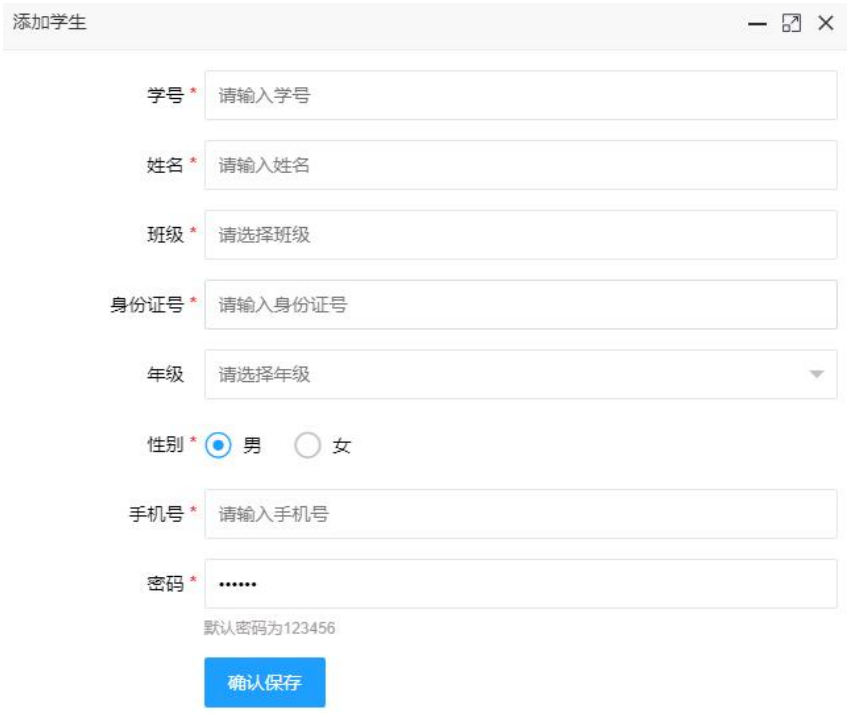

图 5.21　学生管理添加学生页面展示

5.1.8　宿舍编号设置页面的实现

宿舍编号设置页面由宿舍编号列表页面、新增宿舍编号页面和编辑宿舍编号页面组成。分别在 dormitoryset/list.html、dormitoryset/add.html 和 dormitoryset/edit.html 中编写。宿舍编号设置页面如图 5.22 所示：

图 5.22　宿舍编号设置页面展示

超级管理员在该页面可以查看宿舍编号设置的详情。最上方是宿舍管理显示的是当前选

择的楼宇。左侧是该楼宇的哪一层。中间部分是宿舍编号设置的列表，最重要的一个功能是宿舍初始化，在确认宿舍编号设置无误后，可以点击这个按钮实现宿舍编号的初始化。

新增宿舍编号的页面如图 5.23 所示：

图 5.23　宿舍管理新增宿舍页面展示

宿舍初始化的后端部分代码是通过宿舍编号设置的信息进行循环为宿舍的每一层进行编号的插入操作。

代码如图 5.24 所示：

```
dormitorySets.forEach(item -> {
    Dormitory entity = new Dormitory();
    //给宿舍初始化
    for (int i = item.getStart(); i <= item.getEnd(); i++) {
        entity.setNo(item.getPrefix() + i);
        entity.setBuildingId(item.getBuildingId());
        entity.setStoreyId(item.getStoreyId());
        entity.setSex(building.getSex());
        entity.setCapacity(item.getCapacity());
        dormitoryMapper.insert(entity);
        //初始化床位信息
        for (int j = 1; j <= entity.getCapacity(); j++) {
            Bed bed = new Bed();
            bed.setBno(entity.getNo() + "-" + j);
            bed.setDormitoryId(entity.getId());
            bedMapper.insert(bed);
        }
```

图 5.24　宿舍初始化的后端部分代码

5.1.9 宿舍管理页面的实现

宿舍管理页面由宿舍列表页面、添加宿舍页面、添加床位页面和床位页面分配调整组成，分别在 dormitory/list.html、dormitory/addDormitory.html、dormitory/addBed.html 和 dormitory/edit.html 中编写。

最上面显示的是宿舍楼，点击每栋楼的按钮，分别渲染出哪栋楼的宿舍，左侧是该栋楼的每一层，点击对应的层数，渲染出该层的宿舍的信息。右侧为对应宿舍楼对应层的宿舍。宿舍管理页面如图 5.25 所示：

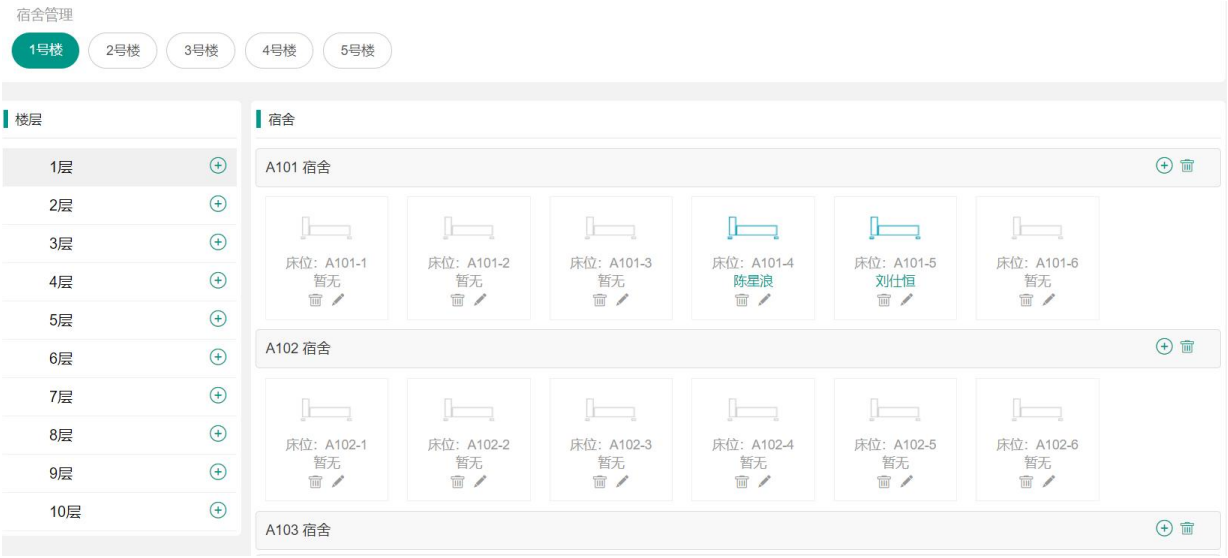

图 5.25 宿舍管理页面展示

宿舍管理页面上侧展示的楼宇的数据，左侧展示的是对应楼宇的每一层，中间部分是对应楼宇对应楼层宿舍的展示。每个宿舍显示的是该宿舍的床位信息。

宿舍管理页面后端部分代码如图 5.26 所示：

```
@PostMapping("/list/{storeyId}")
public Result list(@PathVariable("storeyId") Integer storeyId) {
    LambdaQueryWrapper<Dormitory> queryWrapper = new LambdaQueryWrapper<Dormitory>()
            .eq(Dormitory::getStoreyId, storeyId);
    List<Dormitory> dormitoryList = dormitoryService.list(queryWrapper);
    Storey storey = storeyService.getOne(new LambdaQueryWrapper<Storey>().eq(Storey::getId, storeyId));
    //将楼层对应的宿舍缓存到redis
    String key = STUDENT_DORMITORY + storey.getName();
    redisCache.setCacheObject(key, JSONUtil.toJsonStr(dormitoryList));

    return Result.ok(dormitoryList);
}
```

图 5.26 宿舍管理页面后端部分代码

5.1.10 公告管理页面的实现

公告管理页面由公告列表页面、添加公告页面、编辑公告页面组成。分别在notice/list.html、notice/add.html 和 notice/edit.html 中编写。

公告管理页面如图 5.27 所示：

图 5.27 公告管理页面展示

公告管理页面超级管理员跟宿舍员都可查看、编辑、删除。区别是，超级管理员的权限是所有楼宇，而宿管员的权限是自己负责的楼宇。

添加公告的页面如图 5.28 所示：

图 5.28 公告管理添加公告页面展示

添加公告的后端部分代码如图 5.29 所示：

```
@PostMapping("/saveNotice")
public Result saveNotice(@RequestBody Notice notice) {
    User user = UserHolder.getUser();
    notice.setUserId(user.getId());
    notice.setCreateTime(new Date());
    //将公告插入公告表
    int flag = noticeService.insertNotice(notice);
    //插入公告_接收者关联表
    List<Integer> buildingIds = notice.getBuildingIds();
    for (Integer buildingId : buildingIds) {
        NoticeReceive  noticeReceive = new NoticeReceive();
        noticeReceive.setBuildingId(buildingId);
        noticeReceive.setNoticeId(notice.getId());
        noticeReceiveService.saveNoticeReceive(noticeReceive)
    }
    if (flag > 0) {
        return Result.ok( msg: "新增公告成功！");
    } else {
        return Result.fail( msg: "新增公告失败！");
    }
}
```

图 5.29　添加公告的后端部分代码

5.2　本章小结

本章的主要内容是展示大学生宿舍自选管理系统的功能模块实现情况。呈现了系统主要功能界面的效果图和部分程序代码。

第 6 章　系统测试

6.1　测试环境

大学生宿舍自选管理系统开发设计完成后，需要对其进行系统测试。使用 Windows10 操作系统，2.2.2 RELEASE 版本的 Spring Boot 框架，MySQL.8.0.30 数据库，apache.maven.3.3.9，1.8.0_45 版本的 JDK，IntelliJ IDEA 开发工具，Google Chrome 浏览器进行测试。测试的主要目的就是一定程度上提高程序的质量，让程序达到自己所期望的标准。对本系统的各个模块的每一步操作来测试功能是否达到预期效果，页面与页面跳转是否对应，是否完整且不报错。

6.1.1　超级管理员/宿管员/学生登录功能测试

超级管理员/宿管员/学生登录功能测试主要是对超级管理员/宿管员/学生进行系统登录测试。主要的测试内容为：分别输入超级管理员/宿管员/学生账号和密码，测试该系统是否正常进行。

测试表如表 6.1 所示：

表 6.1　超级管理员/宿管员/学生登录测试表

用例 ID	测试内容	测试步骤	期望结果	结论
1	是否提示账号输入不正确	输入错误的账号	请输入正确的账号和密码	达到预期的效果，测试通过
2	是否提示密码输入不正确	输入错误的密码	请输入正确的账号和密码	达到预期的效果，测试通过
3	是否提示密码输入不正确	账号和密码为空	请输入正确的账号和密码	达到预期的效果，测试通过
4	能否成功登录	填写正确信息，选择身份，点击登录	登录成功	达到预期的效果，测试通过

6.1.2　楼宇管理功能测试

楼宇管理功能测试主要对超级管理员对楼宇信息的添加、修改、删除操作是否成功进行测试。

测试表如表 6.2 所示：

表 6.2　楼宇管理测试表

用例 ID	测试内容	测试步骤	期望结果	结论
1	楼宇列表是否正常显示	查看系统是否显示已有的楼宇信息	正常显示	达到预期的效果，测试通过
2	能否添加楼宇	点击“添加楼宇”，填写正确信息并保存	添加成功	达到预期的效果，测试通过
3	能否修改楼宇信息	填写正确信息，提交修改信息	修改成功	达到预期的效果，测试通过
4	能否删除楼宇	选中相关楼宇，点击删除	删除成功	达到预期的效果，测试通过

6.1.3　宿舍编号设置管理功能测试

宿舍编号设置管理功能测试主要对超级管理员是否能对宿舍编号设置的添加、修改、删除、宿舍初始化操作进行测试。

测试表如表 6.3 所示：

表 6.3　宿舍编号设置管理测试表

用例 ID	测试内容	测试步骤	期望结果	结论
1	宿舍编号设置是否正常显示	查看是否显示已有的宿舍编号设置列表	正常显示	达到预期的效果，测试通过
2	楼栋、楼层能否正常显示	查看已存在的楼宇和对应的楼层是否正常显示	正常显示	达到预期的效果，测试通过
3	能否添加宿舍编号设置	点击“新增设置”，填写并保存	添加成功	达到预期的效果，测试通过
4	能否修改宿舍编号设置	填写正确信息，提交修改信息	修改成功	达到预期的效果，测试通过
5	能否删除宿舍编号设置	选中宿舍编号设置，点击删除	删除成功	达到预期的效果，测试通过
6	能否进行宿舍初始化	选中宿舍编号设置。点击宿舍初始化	初始化成功	达到预期的效果，测试通过

6.1.4　宿舍管理功能测试

宿舍管理功能测试主要对超级管理员/宿管员能否对添加宿舍、删除宿舍、添加床位、删除床位，床位分配调整操作进行测试。

测试表如表 6.4 所示：

表 6.4　宿舍管理测试表

用例 ID	测试内容	测试步骤	期望结果	结论
1	楼宇、楼层、宿舍、床位能否正常显示	查看是否显示已有的信息	正常显示	达到预期的效果，测试通过
2	能否添加宿舍	点击“添加宿舍”，填写并保存	添加成功	达到预期的效果，测试通过
3	能否删除宿舍	选中相关宿舍，点击删除	删除成功	达到预期的效果，测试通过
4	能否进行床位分配调整	选择相关床位，点击“分配调整”，填写并保存	调整成功	达到预期的效果，测试通过

6.1.5　宿舍分配功能测试

宿舍分配功能测试主要对超级管理员是否能对已经设置宿舍预选的班级的分配宿舍的操作进行测试。其功能模块测试如表 6.5 所示：

表 6.5　宿舍分配测试表

用例 ID	测试内容	测试步骤	期望结果	结论
1	分配班级和宿舍能否正常显示	查看是否显示已有的班级和宿舍信息	正常显示	达到预期的效果，测试通过
2	能否分别为每个班级进行宿舍分配	选择相应班级，并选择为该班级分配的宿舍并提交	分配成功	达到预期的效果，测试通过

6.1.6　在线选宿舍功能测试

在线选宿舍功能测试主要对学生是否能对自己所在班级分配的宿舍中进行宿舍的选择的操作进行测试，其功能模块测试，如表 6.6 所示：

表 6.6　在线选宿舍功能测试表

用例 ID	测试内容	测试步骤	期望结果	结论
1	是否显示宿舍	查看是否显示已分配的宿舍信息	正常显示	达到预期的效果，测试通过
2	能否进行宿舍的选择	选中相应的床位进行选择	选择成功	达到预期的效果，测试通过
3	超出选择时间是否能够进行宿舍选择	设置选择时间在当前时间之前	选择失败	达到预期的效果，测试通过

6.2 本章小结

本章首先介绍了大学生宿舍宿舍自选管理系统的测试环境。随后，采用黑盒测试方法对楼宇管理、宿舍编号设置、在线选宿舍、宿舍分配和宿舍管理模块进行了测试，并详细说明了这些模块的测试过程和测试结果。该测试证明了大学生宿舍宿舍自选管理系统的功能是否完善，程序是否健壮，并验证了本次系统开发流程的合理性。通过对系统的测试可以发现潜在的问题并及时解决，保障系统质量的提高。

结论

本文立足与大学生宿舍自选管理系统的开发，最重要的是括简化管理程序、提高管理效率、加强宿舍管理责任心和提高学生参与感等。

通过与传统宿舍管理方式的比较，可以看出自选管理系统的实际效果已经得到了明显的改善。在系统的实际应用中，管理人员能够减少重复的工作流程，在更短的时间内完成更多的工作，提高了宿舍管理的效率。同时，通过大学生的积极参与，宿舍管理责任心得到有效的加强，管理人员更加容易在保证管理质量的同时，获得学生们的理解和支持。

综上所述，大学生宿舍自选管理系统是一种十分有效的宿舍管理方式，可以大大提升宿舍管理的质量和效率。在今后的研究和实践中，应该继续完善和完善这一系统，以更好地推广和应用。

参考文献

[1]唐瑞明,谭倩芳.高校宿舍信息化管理系统设计[J].电子技术与软件工程,2023(04):258-262.

[2]黄强.基于 Web 的高职院校学生宿舍管理系统设计与结构分析[J].无线互联科技,2022,19(20):78-80.

[3]彭灿华.基于云平台的高校毕业设计管理系统设计与实现[J].无线互联科技,2021,18(07):79-81.

[4]Zhao Kai. Design and Analysis of Campus Dormitory Management System Based on Java[J]. The Frontiers of Society, Science and Technology,2020,2.0(17.0).

[5]胡橙凤.基于 B/S 架构高校宿舍管理系统设计与实现[J].电脑知识与技术,2020,16(06):61-62+69.DOI:10.14004/j.cnki.ckt.2020.0640.

[6]Xiaochen Geng,Sha Liu. Application of Modular Interface Design in Student Dormitory Management System*[P]. 4th International Conference on Culture, Education and Economic Development ofModern Society (ICCESE 2020),2020.

[7]唐瑞明,李论,陈珊.高校宿舍管理系统综述[J].电子技术与软件工程,2020(04):64-66.

[8]Yu Yang. Design and Implementation of Student Information Management System Based on Spring boot[J]. Advances in Computer, Signals and Systems,2022,6(6).

[9]任初明,徐延宇,付清香.基于学生视角的大学校园文化认同调查[J].教育理论与实践,2022,42(36):8-12.

[10]吴昌政.基于前后端分离技术的 Web 开发框架设计[D].南京邮电大学,2020.DOI:10.27251/d.cnki.gnjdc.2020.000727.

[11]于昕,廖晨伶,周卫丽.基于 MVC 的软件架构重构与优化研究[J].吉林化工学院学报,2021,38(07):49-52.DOI:10.16039/j.cnki.cn22-1249.2021.07.010.

[12]欧阳宏基,葛萌,程海波.MyBatis 框架在数据持久层中的应用研究[J].微型电脑应用,2023,39(01):73-75.

[13]张旭刚,张昕,高若寒.基于 Spring Boot 与 MyBatis 框架构建动态读写分离模型[J].微型电脑应用,2021,37(02):84-86+98.

[14]何煜琳.基于网站制作的 Web 前端开发优化[J].软件,2021(02):112-114.

[15]王志亮,纪松波.基于 SpringBoot 的 Web 前端与数据库的接口设计[J].工业控制计算机,2023,36(03):51-53.

致谢

在本次毕业设计的完成过程中，我得到了很多人的协助和支持，在此我要向他们表达最真挚的谢意。

先要感谢我的指导崔晶老师，他在我有难题的时候给予了耐心的指导。他不仅在论文撰写上提供了许多宝贵的意见和建议，还常常与我探讨专业问题，使我受益匪浅。同时，我也要感谢我的辅导员王建文老师，四年来王老师总是不辞辛苦为我们创造更舒适的学习环境，生活和学习中处处为我们着想，让我们安心学习，您的良苦用心我感激不尽。

感谢我的爸爸妈妈和家人，他们一直以来默默地支持着我，无论我遇到多少困难和挫折，他们都从不放弃地支持和鼓励我，给予我无私的关爱和支持，让我在学习和生活中觉得踏实稳定和安心。

感谢我的同学和朋友，他们陪伴我度过了美好的大学时光，在我的成长过程中给予了我很多的帮助和支持，与他们相处的时光让我感到无比快乐和幸福。

最后，我要感谢所有为本次毕业提供帮助和支持的人士，你们的帮助和支持是我完成毕业设计的重要动力，谢谢你们！

附录 A　系统核心代码

A.1　前端主要代码

（1）楼宇的展示的主要代码如下:

```
function loadBuilding() {
    axios.post("building/BuildingQueryByPage", {}).then(function (res) {
        res.data.forEach(item => {
            let btn = '<button class="layui-btn layui-btn-primary layui-btn-radius " ' +
                'id="building" data-key="' + item.id + '">' + item.name + '</button>'
            let btnObj = $(btn)
            btnObj.click(function () {
                let buildingId = $(this).data('key')
                $(this).removeClass('layui-btn-primary')
                $(this).siblings().addClass('layui-btn-primary')
                loadStory(buildingId)
            })
            $("#building").append(btnObj)
        })
        //触发点击事件(刷新显示第一个)
        $('#building button:first').trigger('click');
    }).catch(function (error) {
        console.log(error)
    })
}
```

（2）宿舍的展示的主要代码如下:

```
function loadDormitory(storeyId) {
    axios.post("dormitory/list/" + storeyId).then(function (res) {
        $(".dormitory").empty()
        res.data.forEach(item => {
            let html = `
                <li>
                    <p style="" class="layui-colla-title" data-key="${item.id}">${item.no} 宿舍
                        <span>
                            <button class="layui-icon layui-icon-add-circle bed-add" title="添加床位"></button>
                            <button class="layui-icon layui-icon-delete dormitory-delete" title="删除宿舍"></button>
                        </span>
                    </p>
                <ul class="bed">
                <li></li><li></li><li></li><li></li>
                </ul>
                </li>
            `
```

（3）床位的展示的主要代码如下:

```
function loadBed(dormitoryId,objbed){
    axios.post('bed/list/'+dormitoryId).then(function (response) {
        objbed.empty();
        if(response.data.length==0){
            objbed.append('暂无床位信息');
        }
        response.data.forEach(item=>{
            let html = `
                <li data-key="${item.id}">
                <img ${item.student?'src="../images/bed2.png"':'src="../images/bed.png"'} >
                <p>床位：${item.bno}</p>
                <p style="${item.student?'color:#00998b':''}">${item.student?item.student.name:'暂无'}</p>
                <p>
                <i class="layui-icon layui-icon-delete bed-delete" title="删除床位"></i>
                <i class="layui-icon layui-icon-edit bed-edit" title="分配调整"></i>
                </p>
                </li>
            `;
```

（4）宿舍展示部分 css 代码如下：

```
.dormitory > li {
    width: 100%;
    clear: both;
    margin-bottom: 5px;
}

.dormitory > li > p {
    height: 40px;
    background-color: #f8f8f8;
    line-height: 40px;
    font-size: 15px;
    cursor: pointer;
    padding-left: 10px;
    border: 1px solid #dddddd;
    border-radius: 4px;
}
```

A.2 后端主要代码

（1）后端查询宿舍列表主要代码如下：

```
@PostMapping("/list/{storeyId}")
public Result list(@PathVariable("storeyId") Integer storeyId) {
    LambdaQueryWrapper<Dormitory> queryWrapper = new LambdaQueryWrapper<Dormitory>()
            .eq(Dormitory::getStoreyId, storeyId);
    List<Dormitory> dormitoryList = dormitoryService.list(queryWrapper);
    Storey storey = storeyService.getOne(new LambdaQueryWrapper<Storey>().eq(Storey::getId
    //将楼层对应的宿舍缓存到redis
    String key = STUDENT_DORMITORY + storey.getName();
    redisCache.setCacheObject(key, JSONUtil.toJsonStr(dormitoryList));

    return Result.ok(dormitoryList);
}
```

（2）后端添加宿舍主要代码截图如下：

```
@PostMapping("save")
public Result save(@RequestBody Dormitory dormitory) {
    //先查询该栋楼现存在的所有宿舍
    List<Dormitory> dormitoryList = dormitoryService.listDormitory(dormitory);
    for (Dormitory item : dormitoryList) {
        if (item.getNo().equals(dormitory.getNo())) {
            return Result.fail( msg: "该栋楼的该宿舍已经存在，请选择其他宿舍号");
        }
    }
    boolean flag = dormitoryService.save(dormitory);
    if (flag) {
        return Result.ok( msg: "新增成功");
    }
    return Result.ok( msg: "新增失败");
}
```

（3）后端删除宿舍主要代码如下：

```
@GetMapping("/delete")
public Result delete(Integer dormitoryId) {
    boolean flag = dormitoryService.removeById(dormitoryId);
        return Result.ok( msg: "删除成功");
}
```

（4）后端宿舍初始化的主要代码如下：

```
@PostMapping("/initDormitory")
public Result initDormitory(@RequestBody Dormitory dormitory) {
    dormitoryService.initDormitory(dormitory);
    return Result.ok( msg: "初始化成功！");
}
```

（5）后端查询宿舍数量的 MyBatis 的主要代码如下：

```
<select id="selectCountDormitory" resultType="java.lang.Integer" parameterType="java.lang.Integer">
    select count(*)
    from tb_dormitory
    where building_id = #{buildingId}
</select>
```

附录 B　软件使用说明书

B.1　项目启动

（1）打开 IDEA 和 WebStorm，分别拉取 Git 上面的代码，如图 B.1、图 B.2 所示：

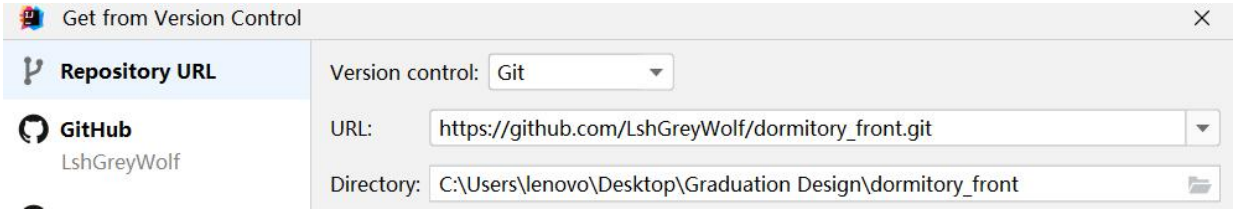

图 B.1　拉取 Git 前端代码

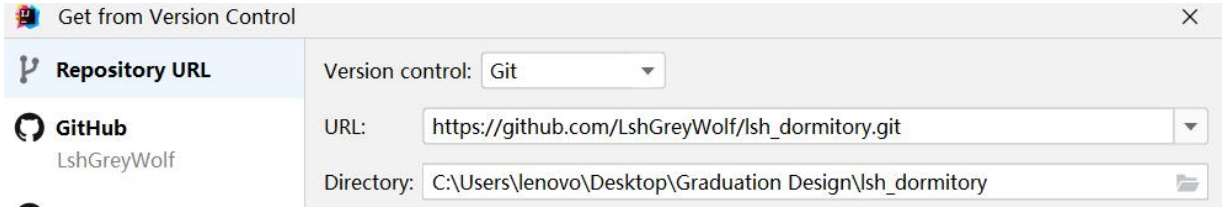

图 B.2　拉取 Git 后端代码

（2）打开 IDEA 项目结构，配置 JDK，使用 JDK1.8，如图 B.3 所示：

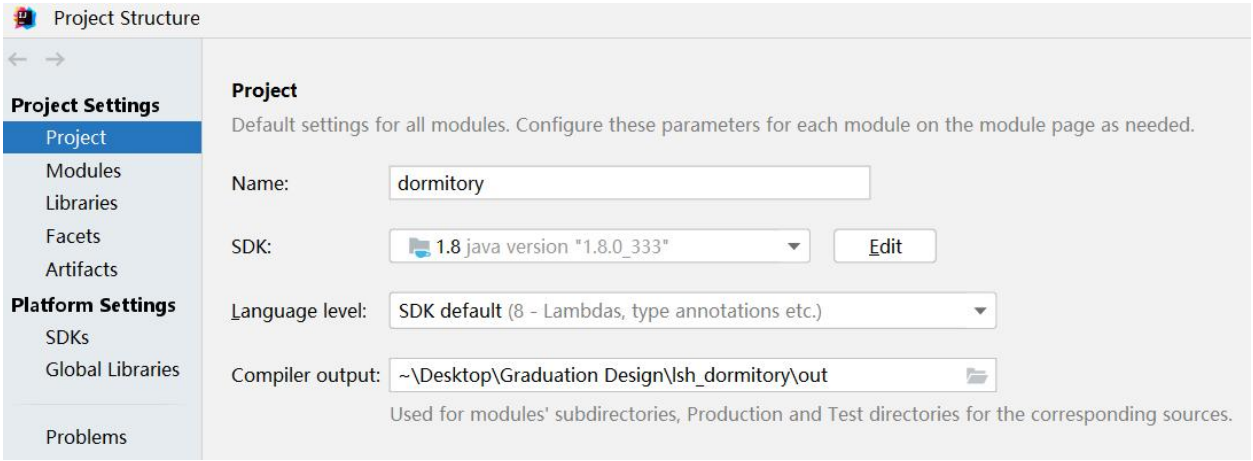

图 B.3　配置 JDK

（3）打开 IDEA 设置，配置本地 Maven 仓库，如图 B.4 所示：

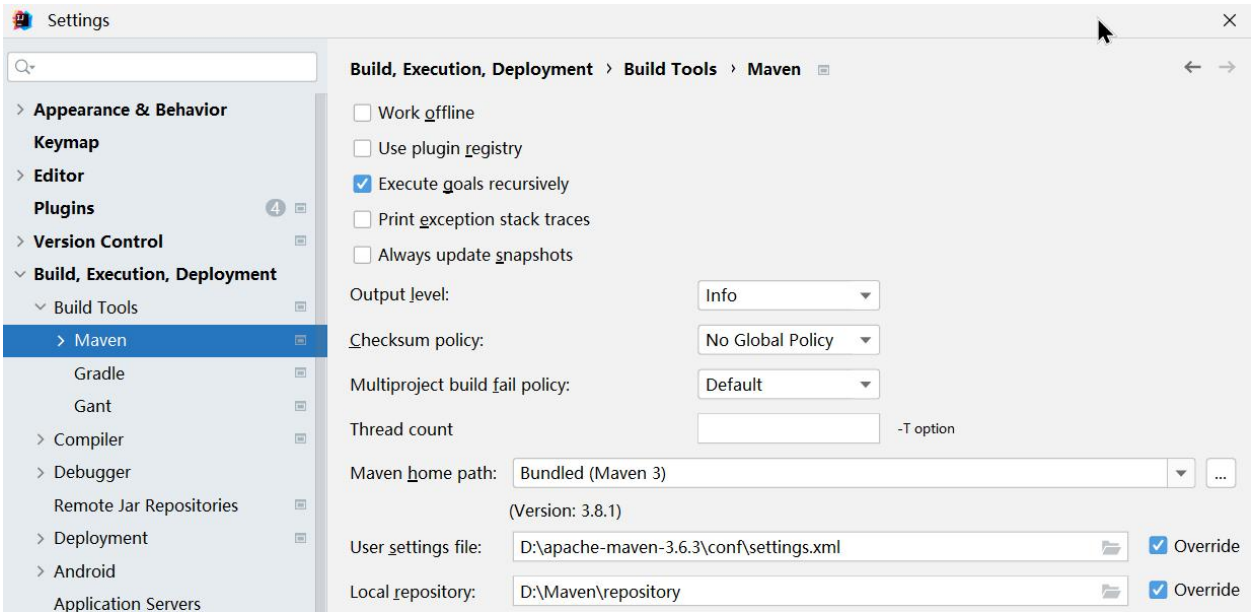

图 B. 4　配置 Maven 仓库

（4）连接 MySQL 数据库，输入账号：root，密码：123。成功登录如图 B.5 所示。

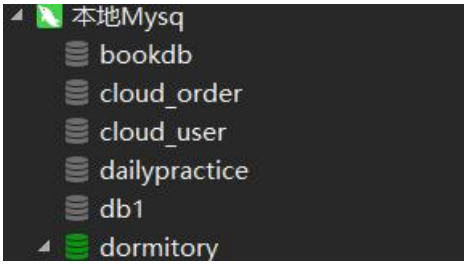

图 B. 5　启动 MySQL

（5）连接 Redis 数据库，打开电脑终端输入 redis-server 命令，成功启动即可。如图 B.6 所示：

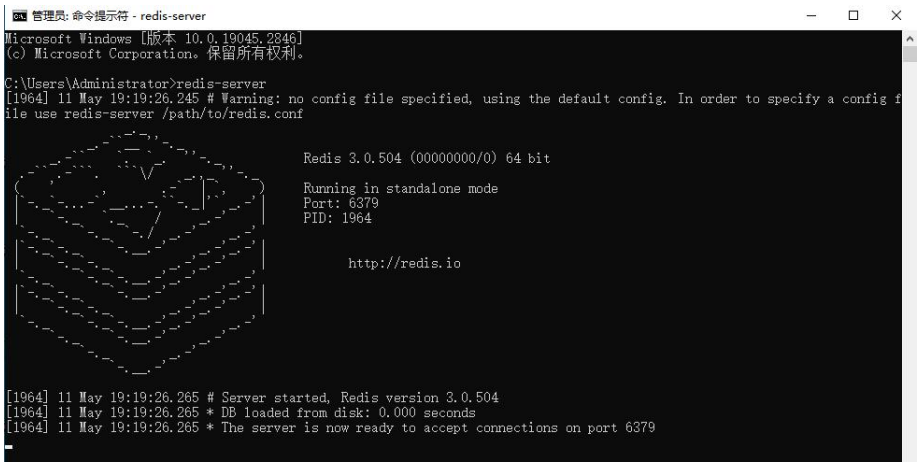

图 B. 6　启动 Redis

（6）启动项目，在 IDEA 中找如图所示按钮，看到控制台“项目启动成功”字样即可。

如图 B.7 所示：

图 B. 7　启动后端代码

（7）然后在浏览器输入 http://localhost:63344/dormitory-front/page/login.html 地址。如图 B.8 所示：

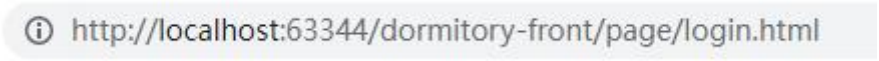

图 B. 8　访问浏览器地址

本系统的用户分为三类，分别为超级管理员、宿管员、学生，对应身份的账号可以使用不同的功能。其测试账号如下：

超级管理员账号：admin，管理员密码：admin

用户账号：zhangsan，用户密码：123456

学生账号：1008519070310，学生密码：123456

B. 2　软件功能介绍

B. 2. 1　学生端介绍

大学生宿舍自选管理系统的学生端主要是为学生提供自主选择宿舍的功能，其余还有报修申请、公告查看、离宿记录等功能供学生使用。学生登录时，需要输入账号、密码。如果账号、密码错误，会提示用户“输入错误，请重新正确的账号、密码”；输入如果账号、密码正确，将跳转到系统的首页。

如图 B.9 所示：

图 B. 9　登录注册

学生成功进入系统后便可进行宿舍的选择。

如图 B.10 所示：

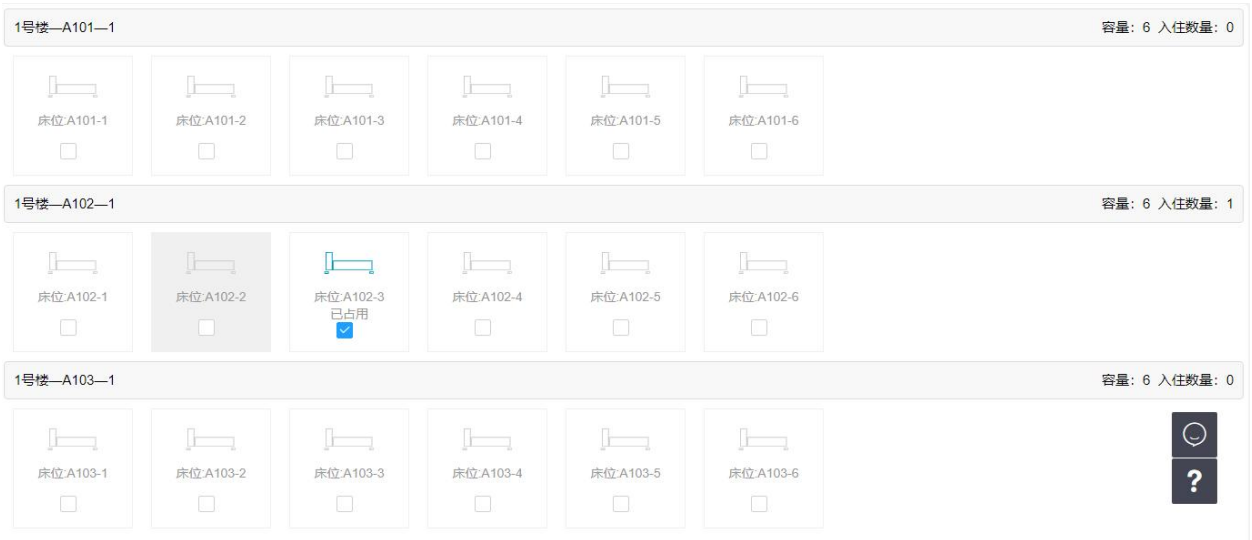

图 B. 10　宿舍选择

学生选择完宿舍后，学生端的首页将展示该宿舍的相关信息。如图 B.11 所示：

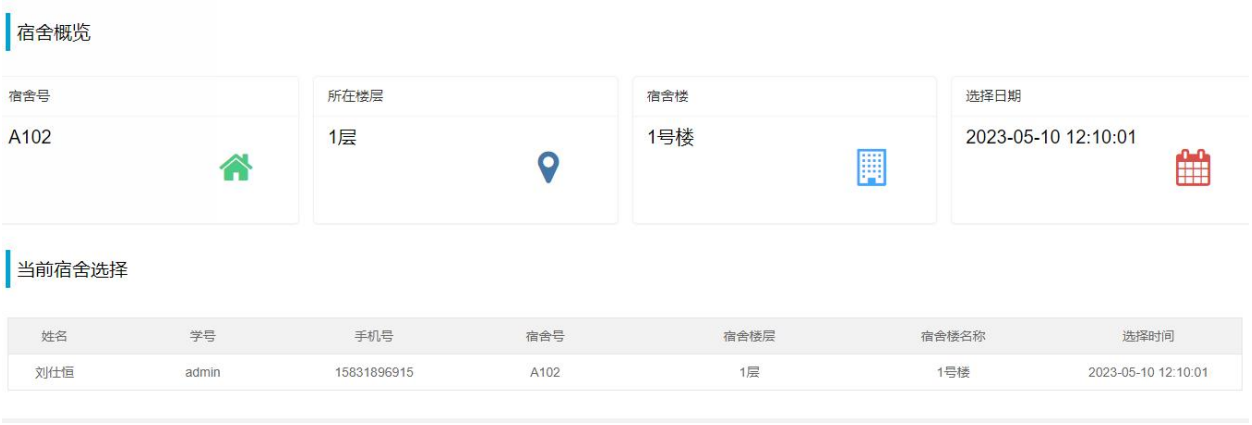

图 B. 11　宿舍信息页

学生可以更改自己的离宿状态。学生在离宿后管理员将状态该学生改为离宿。如果学生在规定时间回来，自己可更改离宿状态为已回，超出时间不可更改。若规定时间未回，系统自动将状态改为缺勤。

如图 B.12 所示：

图 B. 12　离宿记录页

学生可以申请报修。在申请时需要填写报修描述，管理员登录系统看到后会审核。如图 B.13 所示：

图 B. 13　报修申请

学生可以进行公告查看。如图 B.14 所示：

图 B. 14　公告查看

B. 2. 2　管理员端介绍

大学生宿舍自选系统主要对学生、楼宇、宿舍、床位等管理。可以实现对他们的增删改查。同时还有公告管理、报修管理、系统管理等。

后台管理员可以对学生进行管理。可以对学生的信息进行增删改查、重置密码、导入导出学生数据。

界面展示如图 B.15 所示：

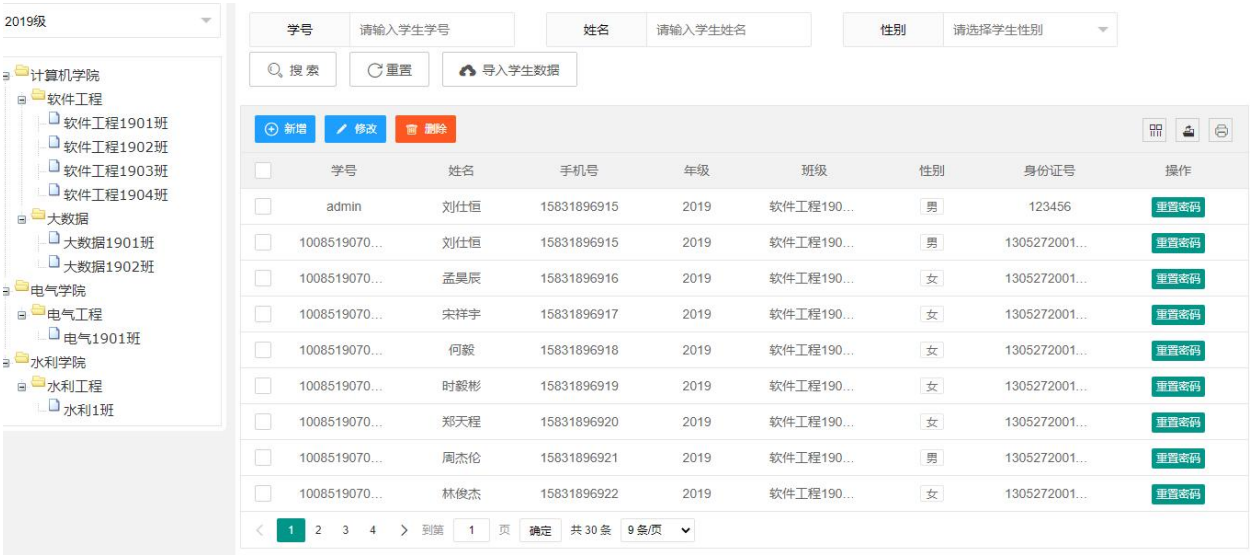

图 B. 15　学生管理展示

后台管理员可以对楼宇进行管理。

界面展示如图 B.16 所示：

图 B. 16　楼宇管理展示

管理员对宿舍管理。

界面如图 B.17 所示：

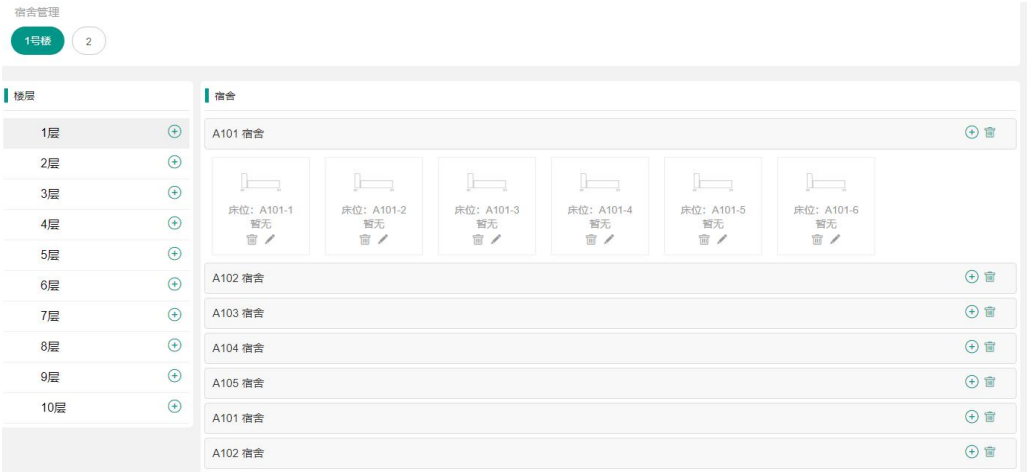

图 B. 17　宿舍管理展示

管理员可对报修审核。

如图 B.18 所示：

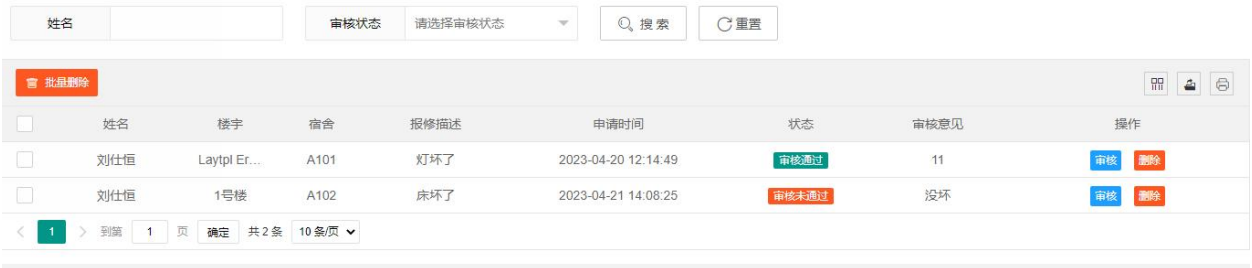

图 B. 18　报修管理展示

审核界面展示如图 B.19 所示：

图 B. 19　审核报修展示