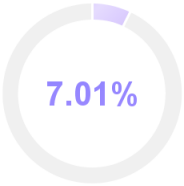
NO. 60a7f568e295603e | 2025-04-22 22:59:51

- 题目： 基于vue校园宿舍管理系统的设计和实现
- 作者： 肖楚凡
- 检测所属单位： -

论文字符数：26320 论文页数：38 表格数量：9 图片数量：24

检测结果



7.01%
全文疑似AIGC生成

92.99%
全文人写概率

结果分布

序号	章节	AI生成文字/章节总字数	AI生成章节占比	人工占比
1	摘要	0/478	0.0%	100.0%
2	第1章 绪论	0/3166	0.0%	100.0%
3	第2章 相关技术和工具介绍	0/2656	0.0%	100.0%
4	第3章 系统分析	165/3020	5.0%	95.0%
5	第4章 系统设计	219/2624	8.0%	92.0%
6	第5章 系统实现	782/5217	14.0%	86.0%
7	第6章 系统测试	147/3787	3.0%	97.0%
8	第7章 总结与展望	0/1275	0.0%	100.0%

*注:格式规范的情况下可准确识别章节，若论文中无章节，可能会识别有误。

片段分布

文字标注

■ 自写片段 ■ 疑似AI生成

基于vue校园宿舍管理系统的设计和实现

吉利学院
GEELY UNIVERSITY OF CHINA

本科毕业论文

题 目： 基于Vue的校园宿舍管理系统设计与实现

学 院： 智能科技学院

年级专业： 2023级计算机科学与技术（专升本）

学生姓名： 肖楚凡

学 号： 231124010338

指导教师：　　周易

二〇二五年五月

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

本人完全了解吉利学院有关保留、使用本科学位论文（设计）的规定，同意吉利学院保留并向国家有关部门或机构送交本毕业论文（设计）的复印件和电子版，允许毕业论文（设计）被查阅和借阅。本人授权，吉利学院可以将本毕业论文（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、缩印或扫描等复制手段保存和汇编本毕业论文（设计）。

作者签名：　　指导教师签名：

摘要

在最近的几年里，高校招生的规模一直在不断地扩大，这导致宿舍管理的复杂程度有了十分突出的增长。从教育部在2022年的统计情况来看，现在全国普通高校里面在校学生的规模已经突破了4000万人，在这么多学校当中，超过70%的学校依旧采用人工登记或者使用Excel表格的方式来管理宿舍信息。传统的这种管理方式存在着三个主要的问题：第一，宿舍分配需要依赖手工来进行操作，这样很容易出现床位分配产生冲突的情况；第二，设备报修的流程需要学生到线下填写表格，而且平均处理事情的周期比较长，一般会达到3-5天；第三，访客登记的信息很难进行追溯，这样存在着一定的安全方面的隐患。

在传统的宿舍管理模式里，宿舍分配主要是依靠人工来进行操作的，管理人员需要手工地去记录学生的相关信息，再依据有限的床位资源来展开分配工作，这样的方式很低效，而且还很容易出现床位分配产生冲突的情况。有时候，可能会出现同一个床位被分配给多个学生这样的错误，或者是因为手工记录不够准确而引起的重复分配问题，这些问题让管理工作的复杂程度提高了，还对学生的住宿体验以及对学校宿舍管理的满意度产生了不良的影响。

关键词：宿舍管理系统；Vue3；宿舍管理优化；权限控制

ABSTRACT

In recent years, the scale of university admissions in China has been steadily expanding, resulting in a significant increase in the complexity of dormitory management. According to

statistics released by the Ministry of Education in 2022, the number of students currently enrolled in regular higher education institutions nationwide has surpassed 40 million. Among these institutions, over 70% still rely on manual registration or Microsoft Excel spreadsheets for managing dormitory information. This traditional mode of management presents three major challenges: First, dormitory allocation depends heavily on manual operations, which often leads to conflicts in bed assignments. Second, the process of equipment repair requires students to fill out forms offline, with an average handling time of 3 to 5 days. Third, visitor registration records are difficult to trace, posing potential security risks.

In the traditional dormitory management model, student accommodation assignments are primarily carried out manually. Administrators must record student information by hand and allocate beds based on limited housing resources. This approach is not only inefficient but also prone to allocation errors. In some cases, a single bed may be assigned to multiple students, or duplicate allocations may occur due to inaccuracies in manual recordkeeping. These issues increase the complexity of administrative tasks and negatively affect both the students' residential experience and their overall satisfaction with the university's dormitory management system.

Keywords: Dormitory management system; Vue3;Repair process optimization; Authority contro

目录

第1章 绪论

1.1 研究背景

在最近的几年里，高校招生的规模一直在不断地扩大，这导致宿舍管理的复杂程度有了十分突出的增长。从教育部在2022年的统计情况来看，现在全国普通高校里面在校学生的规模已经突破了4000万人，在这么多学校当中，超过70%的学校依旧采用人工登记或者使用Excel表格的方式来管理宿舍信息。传统的这种管理方式存在着三个主要的问题：第一，宿舍分配需要依赖手工来进行操作，这样很容易出现床位分配产生冲突的情况；第二，设备报修的流程需要学生到线下填写表格，而且平均处理事情的周期比较长，一般会达到3-5天；第三，访客登记的信息很难进行追溯，这样存在着一定的安全方面的隐患。

在传统的宿舍管理模式里，宿舍分配主要是依靠人工来进行操作的，管理人员需要手工地去记录学生的相关信息，再依据有限的床位资源来展开分配工作，这样的方式很低效，而且还很容易出现床位分配产生冲突的情况。有时候，可能会出现同一个床位被分配给多个学生这样的错误，或者是因为手工记录不够准确而引起的重复分配问题，这些问题让管理工作的复杂程度提高了，还对学生的住宿体验以及对学校宿舍管理的满意度产生了不良的影

响。

在这样的模式之下，设备报修一般需要学生到线下去填写报修单，然后把报修单提交给相关部门等待处理，这种流程特别繁琐，效率也很低，平均的处理周期差不多有3到5天。在这段时间里，学生的正常生活有可能会受到一定的影响，要是出现的是那些会干扰日常生活的设施故障，像水电方面的问题、门锁出现损坏等情况时，长时间地等待以及不确定的结果可能让学生对宿舍管理产生不满的感觉，这还会对学校的整体管理形象造成不好的影响。传统的访客登记方式大多依靠纸质记录来进行，这样的信息管理不够规范，想要查询的时候也会很险阻，当需要去追溯访客的信息的时候，一般很难快速又准确地拿到相关的数据，这种管理方式存在非常严重的安全隐患，没有办法有效地保障学生的住宿安全，在遇到突发事件或者紧急的情况时，因为缺少准确的访客记录，可能会让安全管理变得被动并且出现滞后的问题。

1.2 目的和意义

开发校园宿舍管理系统，能够凭借信息化的手段来实现全流程的数字化管理，这套系统上线之后，主要目标是：是宿舍分配的效率提升60%以上，并且支持自动分配的算法，这样能避免出现人为的失误；

传统的高校宿舍管理方式主要是依靠人工来进行操作的，在这个过程中涉及到很多个环节，像学生入住的时候进行登记、给学生分配宿舍以及学生使用设备出现问题需要报修等，这样的管理方式的工作量特别大，并且很容易出现信息被遗漏或者出现错误的情况，要是引入基于Vue的宿舍管理系统的话，能够实现信息的自动化的处理，可以减少人工的干预，从而提高工作的效率。例如说，系统可以自动去记录并且更新学生的入住信息，把宿舍分配的流程简化，还可以保证数据有较高的准确性以及实时性。

在以往的这种模式当中，学生想要进行报修的话，需要亲自到线下提交报修申请，这样一来，整个流程特别繁琐，而且从申请到问题得到处理所花费的时间也比较长，这对学生的日常生活造成了比较大的影响，后来有了基于Vue的宿舍管理系统之后，学生可以借助这个系统在线提交自己的报修申请，并且还可以够实时地去查看处理的进度，这大大地提升了服务的便捷程度以及透明度。除此之外，这个系统还可以为学生提供一些其他的便利功能，像公告通知功能和失物招领功能等，这也丰富了学生校园生活的体验。

高校信息化代表着现代教育发展的必然趋势，而开发并且应用基于Vue的宿舍管理系统，能够提升宿舍管理的现代化水平，还可以为其他管理领域的信息化提供有效的借鉴和良好示范，进而推动高校整体信息化建设向前发展。

从上面的内容可以得出，依托Vue技术来进行宿舍管理系统的相关研究和开发工作，是有很大帮助的。它能够帮助解决目前高校在宿舍管理方面存在的现实问题，而且对于提高高校管理的信息化程度以及智能化水平也有着非常关键的意义。

1.3 国内外发展情况

1.3.1 国内研究方面

随着智慧校园的不断建设，宿舍管理，作为校园管理里的关键部分，正逐渐向着智能化和信息化的方向进行发展。在国内，许多高校在智慧校园的建设框架之下，积极开展了宿舍管理方面的信息化研究以及开发工作，Vue是一种非常高效的前端开发框架，它慢慢地变成了很多项目当中首选的技术栈。逐渐成为许多项目中的首选技术栈。

沈泽刚的观点是，凭借运用模块化设计的方式，程序的可维护性和可扩展性都得到了十分突出的提升，特别是在大型软件开发项目当中[1]。要是有着良好的模块化架构，能够有效地降低软件开发的复杂程度，拿宿舍管理系统

的设计来说，把沈泽刚的理念融入进去，能够帮我们在系统里开展高效的功能模块划分工作，这样不同的功能板块像宿舍分配、费用管理和报修系统等能够独立进行开发、测试以及维护，进而提高系统的可扩展性和可靠性。我们可以得出这样的结论，采用模块化设计的办法可以有效地提升宿舍管理系统的开发效率，也能让系统的长期维护能力得到加强。

Vue的作者尤雨溪着重指出，Vue在前端开发方面有很显著的优势[2]。这主要体现在它的响应式数据绑定以及组件化开发上，据尤雨溪所说，Vue能够帮开发者达成高效的界面更新和数据管理工作目标，在动态交互式的应用当中，Vue的响应式系统可以使得界面结合宿舍管理系统的开发，尤雨溪的设计理念为我们提供了一种高效、灵活的开发方式，通过Vue的组件化结构，可以将宿舍管理系统中的各个功能模块进行组件化开发，方便后期的扩展和维护。采用模块化设计能够有效地提高宿舍管理系统的开发效率，还可以够增强系统进行长期维护的能力。

1.3.2 国外研究显示

美国麻省理工学院（MIT）在其校园管理系统当中专门开发了一个宿舍管理模块，这个系统有以下这些功能：学生可以借助这个系统在线去申请宿舍、进行住宿方面的安排，还可以上报设施需要维修的情况，该系统的前端采用了Vue框架，并且还结合了响应式设计，这样做能够保证学生们不管使用的是什么设备，都能够有很不错的使用体验。系统凭借和后端数据库实现集成，可以及时对宿舍的入住情况作出更新，这样一来减少了人工方面的干预，让管理的效率得到了提高，该系统前端使用了Vue框架，而且还结合了响应式设计，从而确保学生们在使用不同设备的时候都有良好的使用体验，凭借与后端数据库的集成，系统能够实时对宿舍的入住情况进行更新，减少了人工的干预，提高了管理的效率。

Mitrari认为，随着信息技术不断地发展[3]，宿舍管理系统可以借助云计算平台来实现数据存储以及处理的集中化操作，这样做减少了对于本地硬件设施的依赖程度，而移动应用可以让宿舍管理员以及学生在任何时间、任何地点都方便地管理宿舍相关信息，这提升了系统所有的便捷性和实时性特点，他们的设计方案让宿舍管理实现了自动化，这样一来减少了人工干预的情况以及管理方面的成本。得出的结论是，云计算和移动应用进行结合之后，让宿舍管理系统变得更加智能化、更高效，而且能够契合现代校园管理对于灵活性以及可扩展性方面的需求。

Liu.J认为，依托物联网技术构建的宿舍管理系统，有潜力实时检测宿舍里各种设备的状态，例如电器、照明设施以及温度情况等，并且能够凭借传感器所采集到的数据，对宿舍的环境实现智能方面的调控，从他们的研究结果来看，物联网技术可以有效地提升宿舍的智能化程度，它能够给宿舍管理员提供实时更新的数据，还可以让学生拥有更加舒适的居住环境。靠着对智能化设备进行控制，宿舍管理系统除了让管理效率得到提升之外，还可以对能源使用开展优化管理，进而降低整个校园的运营成本。最终得出的结论是，基于物联网技术的宿舍管理系统有着非常巨大的发展潜力，它能为校园管理提供更加精准的数据支持以及决策依据，还可以够有效地提高校园管理工作的效率和质量。

第2章 相关技术和工具介绍

2.1 Vue3框架概述

Vue 3是由尤雨溪主导开发的一款前端JavaScript框架，它是Vue系列目前最新的版本，在性能、功能以及开发体验等各个方面，Vue 3和Vue 2相比有十分突出的改进，从具体情况来看，在响应式数据绑定、组件化开发以及引入组合API等这些方面，Vue 3给开发者提供了更加强大的功能。以宿舍管理系统前端开发为例，使用Vue 3可以为我

们提供一个更加高效、更加灵活，并且更容易进行维护的开发平台。Vue 3的一个核心特点是它有了全新的响应式系统，Vue 3是借助Proxy来实现对数据的响应式管理的，和Vue 2使用的Object.defineProperty这种方式相比，Proxy在性能以及灵活性方面都有着非常显著的提升，Vue 3的响应式系统可以更加高效地去追踪数据的变化，而且支持更复杂的数据结构，这样提高了宿舍管理系统中数据处理的效率。拿例子来说，当用户去修改宿舍信息或者提交报修请求的时候，Vue 3能够实时更新页面的内容，保证用户界面的准确性和及时性。

Vue 3的一个关键特性是引入了组合API这种新的API提供了更加灵活的功能组合方式，让组件的可重用性和可维护性得到了很大程度的提高，在Vue

2中，使用的传统方式是选项式API，这种API依靠定义data、methods、computed等选项来对组件的状态和逻辑进行管理。然而在Vue3里，组合API把组件的逻辑分割成了多个小的并且可以重复使用的函数，它允许开发者依据功能把这些逻辑整理到一起，而不是按照固定不变的选项来进行配置，这对于开发像宿舍管理系统一样功能复杂、需求经常发生变化的应用来说特别关键，它帮我们把不同模块的逻辑区分开来，提高了系统的可扩展性和可维护性。

Vue 3对虚拟DOM的性能进行了改进，并且提高了对TypeScript的支持程度，它凭借对虚拟DOM渲染机制进行优化，能够让Vue 3更加高效地去处理大量DOM节点的更新工作，在宿舍管理系统里，可能会涉及到复杂的数据交互和页面渲染情况，这特别关键。而且，Vue 3提升了对TypeScript的支持，这样让开发者在编写代码的时候，能够得到更佳的类型检查和自动补全体验，减少了开发过程中存在的错误，还提高了代码的可读性和可维护性。

从上面的内容可以了解到，Vue 3依靠它非常先进的响应式系统、组合API功能、性能方面的优化，以及对TypeScript的良好支持，给宿舍管理系统提供了一个十分强大又高效的前端开发框架，这个框架极大地提高了开发工作的效率，也让系统的可维护性得到了提高。

2.2 Pinia 状态管理

Pinia 是由 Vue 3 官方所推出的用于状态管理的库，它是 Vuex 的继承者，在性能、体积以及易用性等方面有着更出色的表现，Pinia 的架构设计紧紧贴合 Vue 3 的响应式系统，能够更加高效地达成组件之间的状态共享与管理工作，在宿舍管理系统实际的开发过程当中，Pinia 提供了有结构清晰、逻辑统一特点的全局状态管理能力，大大提升了各个模块之间数据传递的稳定性以及一致性，为系统实现高效运行给予了有力的支持。和 Vuex 相比，Pinia 在设计方面做了许多改进，第一，Pinia 的 API 更加简洁且好使用，特别是在进行模块化开发以及状态管理时，开发者不需要像使用 Vuex 那样借助 mutation、action 等中间层来对状态进行管理。Pinia 直接凭借"store"对象来对状态进行管理，组件可以直接借助 getter 来获取状态数据，凭借 action 来更新状态，这令得代码更加简洁，更便于被理解。

Pinia有更高的性能表现，它借助使用Vue 3的响应式系统，能够更加高效地去追踪状态的变化情况，并且对性能进行了优化，这种优化在处理大量数据更新和交互的场景时特别明显，对于宿舍管理系统而言，Pinia可以帮我们来管理登录状态、宿舍信息、学生信息等全局状态，还可以够保证数据在不同的页面以及模块之间保持数据的一致性。

Pinia的一个优点是它原本就支持TypeScript，这样一来，开发者能够享受到更佳的类型推导以及类型检查，这利于减少开发过程中出现的错误，还可以提高代码的可维护性，在开发宿舍管理系统的时候，使用Pinia可以帮我们更好地对全局状态进行管理，从而让系统里面的数据共享与同步变得更加稳定且可靠。

借助Pinia，宿舍管理系统可以实现更加高效的状态管理，它凭借简洁的API设计以及出色的性能，提高了开发的效率，也让系统的可靠性得到了提升。

2.3 Element-plus组件库

Element Plus是Element UI针对Vue 3打造的版本，它是一款以Vue3为基础进行开发的UI组件库，这款组件库的目的是给开发者提供一套有高质量的UI组件，从而帮助开发者能够快速地构建出美观并且实用性高的Web应用，Element Plus承了Element UI所有的优点，对其性能以及兼容性进行了优化，并且特别适合用在Vue3的项目当中。在开发宿舍管理系统的时候，ElementPlus为我们提供了数量丰富的UI组件以及灵活的样式自定义功能，这大大提升了系统使用的体验。

Element Plus确为开发者提供了非常多常用的UI组件，像按钮、表单、对话框、表格、分页、日期选择器等等，这些组件几乎是覆盖到了开发Web应用所需要的基础组件了，在宿舍管理系统当中，学生信息管理、宿舍分配以及报修系统等功能，都是借助Element Plus的UI组件来实现的。拿宿舍信息管理页面来说，它运用了Element Plus的表格组件来展示宿舍编号、入住人数以及剩余床位等信息，并且还提供了排序、筛选以及分页等功能，这样大大地提高了用户的交互体验。

Element Plus的设计风格秉持着简洁并且一致的原则，它可以帮助开发者构建出美观并且符合现代设计规范的界面，借助Element Plus所提供的样式定制功能，我们能够依据学校具体的实际需求来对组件的颜色、字体以及布局进行调整，从而让系统界面和学校的整体风格保持统一。除此之外，Element Plus还支持按需加载，仅引入当前所需要的组件，这样能有效地减少项目打包的体积，进而提高系统加载的速度。

Element Plus 为开发者提供了丰富的功能支持，其中包括表单验证、日期选择以及上传文件等常用功能，这对于像宿舍管理系统这类应用来说是非常有帮助的，拿宿舍申请表单来举例，借助 Element Plus 的表单组件，我们能够快速地完成表单校验工作，从而保证用户输入的数据是准确并且没有错误的。

第3章 系统分析

3.1 系统需求分析

为满足宿舍管理业务数字化的目标，系统划分为学生端与管理端两大角色模块。学生端面向普通用户，具备账户创建、宿舍分配、信息填写与修改等功能。管理端服务于管理人员，负责宿舍资源配置与学生数据维护等任务。系统结构图如3-1所示

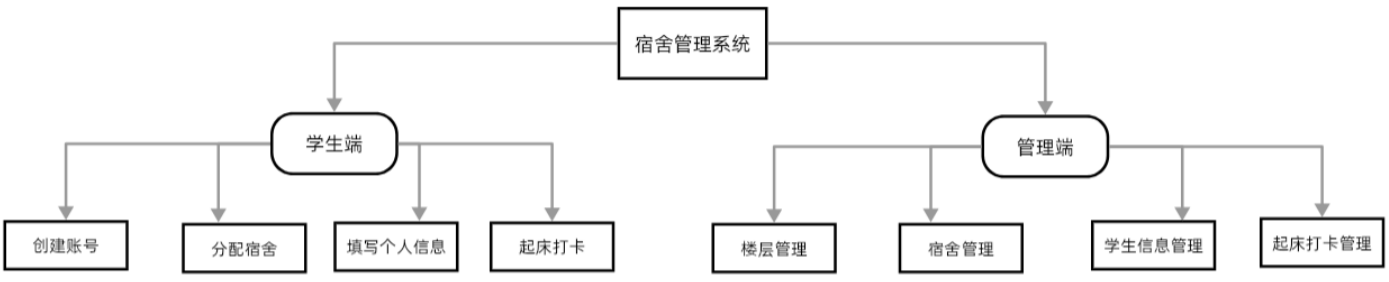


图3-1 宿舍系统结构图

功能需求层面，学生可通过注册功能生成账户并登录，填写与维护个人信息，完成宿舍分配操作。管理端具备楼层与宿舍信息维护、宿舍评价录入、学生信息检索与晚归行为统计等管理功能。

非功能需求方面，系统需保障良好的响应速度与用户交互体验，支持跨平台运行。安全性方面，需防范非法访问与数据泄露，保证系统稳定运行与数据一致性。

3.2 系统角色与权限分析

本系统的角色分为学生用户和管理用户两个类别。其中学生用户是整个系统设计的第一优先级，在功能实现的过程中也始终围绕学生用户的使用场景进行构建，学生用户在初次进入系统时必须先完成注册流程，系统设置了手机号与验证码结合的验证机制 在确保身份的同时也能提升注册的便捷程度，用户输入有效手机号后系统将自动发送六位数字验证码 验证通过后进入下一步流程，完成基础信息的填写即可完成注册，注册完成后学生用户将进入系统的个人中心模块，该模块是所有操作的起点。

学生端的权限主要集中在三个板块，第一个是账户管理功能 这个板块允许用户修改当前密码并发起账号注销请求。修改密码过程中系统会要求用户提供原密码与新密码的组合来防止误操作，注销账号操作会触发二次确认以防用户误触，第二个是信息维护功能，该功能覆盖了用户日常使用中最常需要维护的字段，包括当前联系方式，紧急联系人，学院 所属专业等字段，学生用户可以随时进入模块进行信息的查看与更新，所有字段都有输入格式校验机制确保信息准确性，第三个是宿舍绑定功能 这是该系统在学生端最具交互性的功能，模块界面设置了一个可视化的房间列表，学生可以在页面上看到每一个宿舍房间的编号所在楼层是否已满，还提供了筛选器支持，用户可以按楼层，房间容量，是否靠近电梯等维度来筛选空房间。

再来看管理端，管理用户的职责是维护学生宿舍数据，整体权限较高。系统默认只有经过管理员账户审核创建的用户才拥有管理权限，管理用户的主要功能包括宿舍信息录入，学生入住审核，楼层管理以及宿舍评价查看等，管理端操作界面与学生端不同，更加偏向于表格列表形式，批量处理功能被大量使用，管理用户在系统中可以对某一栋楼或某一个楼层的宿舍分布进行整体查看并作出调整，例如可以直接清空某个房间的入住信息，也可以添加新的宿舍数据，使整个宿舍资源的配置更灵活更可控。此外管理员还可以查看学生入住后的评分反馈信息，作为宿舍环境改进的重要参考。宿舍管理界面如图3-2所示

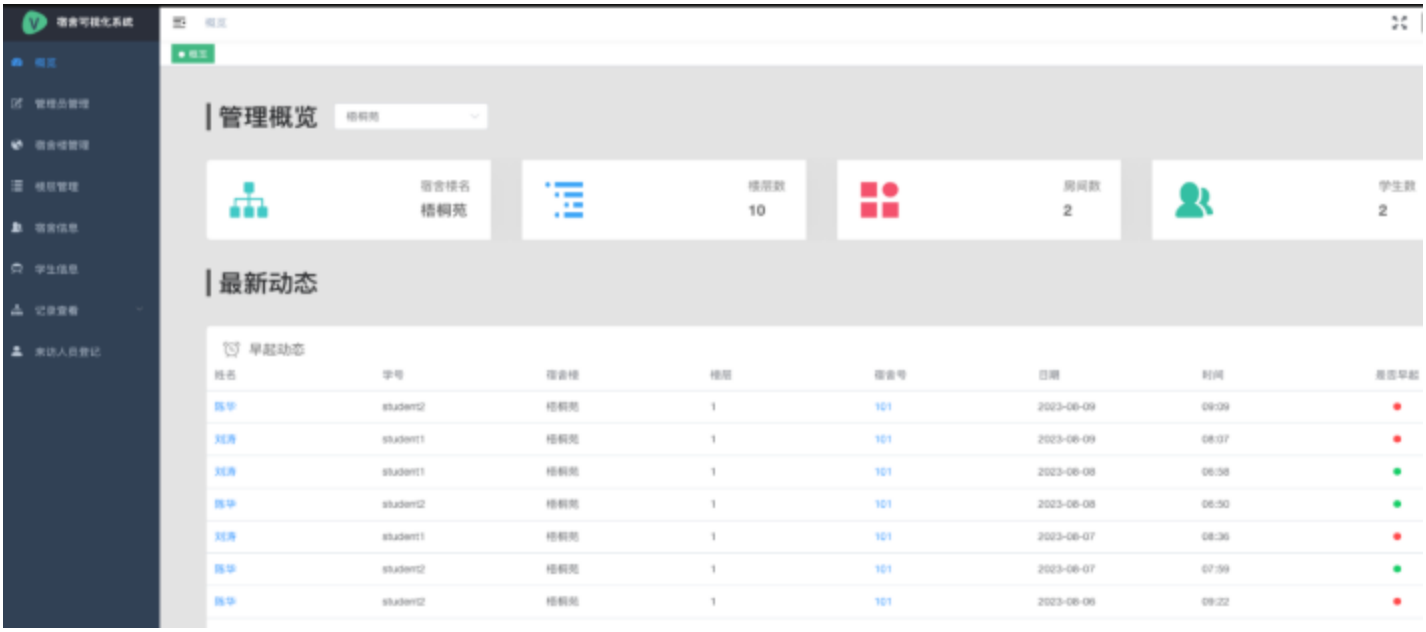


图3-2 宿舍管理界面

3.3 系统功能模块分析

3.3.1学生端功能模块

学生端模块主要是聚焦那些和用户的个体宿舍有关的需求，用户要是想使用这个系统的话，需要凭借注册页面来提交自己的手机号以及短信验证码，然后进行验证，如果验证能够顺利通过，那么用户就可以完成系统的注册了。在用户第一次登录系统的时候，系统会借助一系列的行为来引导用户去填写基础的资料，这些基础资料包含了姓名、学号、联系方式、紧急联系人、学院、专业等多个字段。用户填写好这些信息之后，信息还会经过格式的校验，校验没有问题之后才会提交给服务器进行存储，等到用户把信息填写完成之后，这样才可以进入宿舍分配的流程当中。

宿舍分配模块推出了两种绑定模式，分别是自助选择和系统推荐，在自助选择模式下，会呈现出当前楼栋、楼层以及宿舍空余床位的状态，并且还支持进行多条件筛选以及排序操作，这样一来能够提高用户挑选的效率。而系统推荐模式是根据算法规则去匹配相似的学院、年级或者兴趣标签，然后自动为用户推送最合适的宿舍候选列表，可以减少用户的操作步骤，完成绑定之后还会自动记录下绑定的时间以及宿舍的编号。

信息维护模块支持部分字段的实时编辑，包含联系电话、紧急联系人与个人描述等。为保障数据一致性与系统安全，敏感字段如学号、账号等设置为只读字段，仅管理员可通过后台接口进行修改。每一次信息更新请求均由后端执行字段校验、字段范围限制、历史数据记录存档，并通过 Sequelize 提交至数据库存储层，确保操作可追踪与数据恢复能力。学生端的用例图如图3-3所示

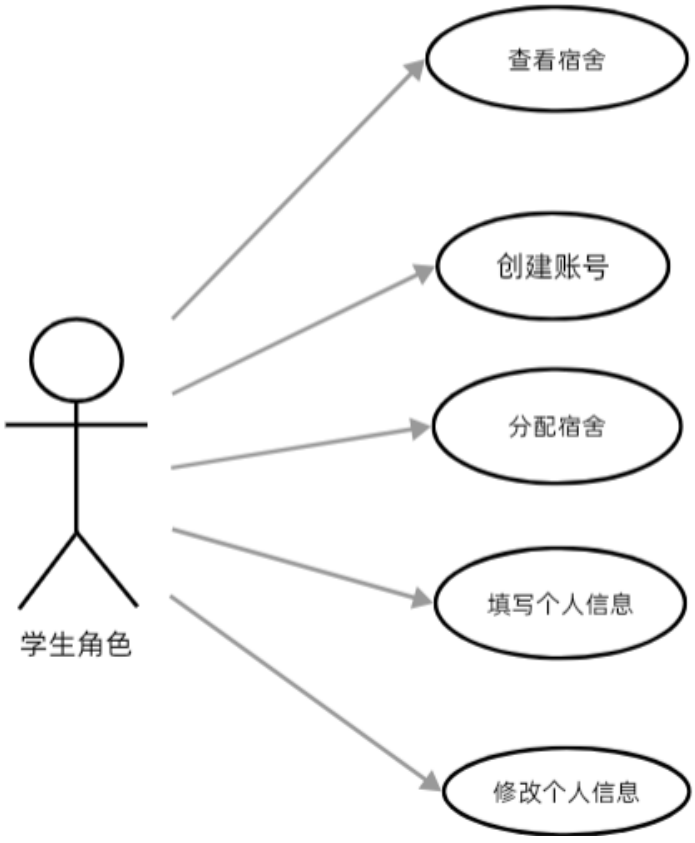


图3-3 学生端用例图

3.3.2 管理端功能模块

管理端功能模块如图3-4 所示：

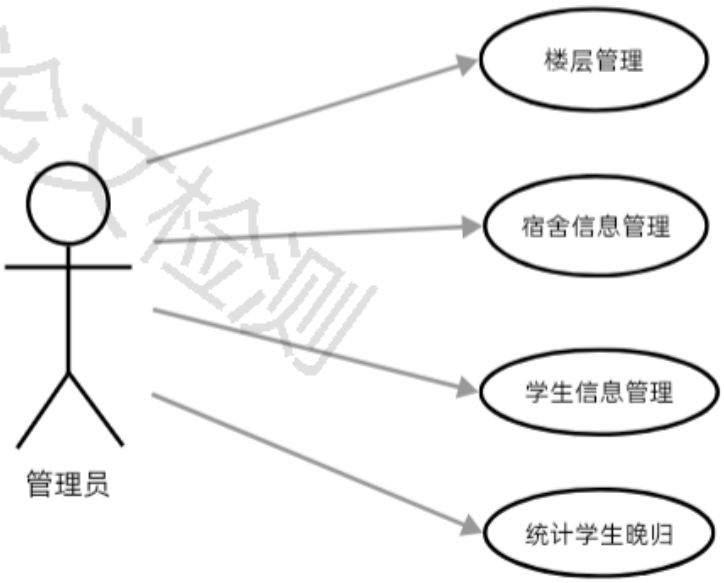


图 3-4 管理员用例图

管理端的设计包含了五个关键的功能模块，分别是楼层管理、宿舍信息管理、宿舍评价、学生信息查看以及晚归统计，这每一个单独的模块在系统当中都有各自对应的 tasks场景。其中，楼层管理这个模块主要涉及到的是系统的基础配置方面的工作；宿舍信息管理这个模块主要是和系统的资源调度相关；宿舍评价这个模块主要是对质量进行评估；学生信息查看这个模块主要是对用户进行管理；而晚归统计这个模块则主要进行行为分析。

楼层管理模块能够支持对宿舍楼栋以及它内部的楼层结构开展可视化的配置工作，管理员可以创建新的楼栋实体，并且定义好这座楼的楼层数量、每单一层的房间数量以及房型的具体结构，系统里面是有内置一些结构预设模板的，这些模板是供快速配置时使用的，要是需要的话，管理员还可以手动去编辑楼层节点，例如说移动、合并或者删除某个楼层节点。在整个配置的过程中，数据模型的结构会实时同步，系统还会对结构的完整性以及房间编号的唯一性进行校验。

宿舍信息管理模块把宿舍当作核心的数据单元来呈现，会显示出宿舍的编号、它所在的楼层、可用的床位数量、现在入住的学生名单以及入住的历史记录，这个模块有批量导入宿舍数据的功能，允许上传Excel格式的文件，系统会对文件进行格式规范性的验证，对重复的数据进行过滤，还会进行自动编号的处理操作，它支持根据楼栋、楼层、宿舍编号等这些条件来筛选宿舍的信息，并且能够导出宿舍入住情况的详细明细。宿舍评价模块是按照宿舍这个单位来录入各项评分结果的，它会从多个不同的维度去进行评分，例如说清洁程度如何、安全方面得分怎么样、以及设施维护的情况等等，这些评分结果的来源主要有两个方面，一方面是学生之间相互评价而得出的成绩，另一方面则是管理员巡查之后得到的结果。接着，系统会对每一项评分都进行加权方面的计算，最后会呈现出每个宿舍综合评分的具体情况以及排名的结果，而且，评价的相关记录可以依据学期或者每个月度来进行归档保存，并且还能够以图表的形式来展示这些信息，还支持开展横向的比较和分析工作。

学生信息查看模块旨在帮助管理员实现对学生的相关信息进行快速的访问以及精准的筛选，这个模块能够支持多种不同条件的组合筛选操作，像姓名、学号、学院、年级、宿舍号以及入住状态等这些字段都是可以作为筛选条件来使用的，而且得到的筛选结果还提供了分页浏览的功能，并且还可以导出相关数据，会生成符合标准格式的Excel 报表，这样做的话方便管理员在线下开展管理工作以及对相关数据进行汇总操作。

晚归统计模块是借助门禁系统所收集到的数据，以及根据系统事先设定好的标准归寝时间，自动地去生成学生的晚归记录，它能够支持按照天、周、月这些不同的维度来统计学生迟归的频次，并且还可以提供迟归学生名单的导出功能以及行为趋势的图表分析功能。在进行统计的时候，它的统计逻辑会考虑到像节假日、学生的请假状态等这些可能会产生干扰的因素，运用标签识别的机制把无效的迟归记录给排除掉，从而提升数据的准确程度。

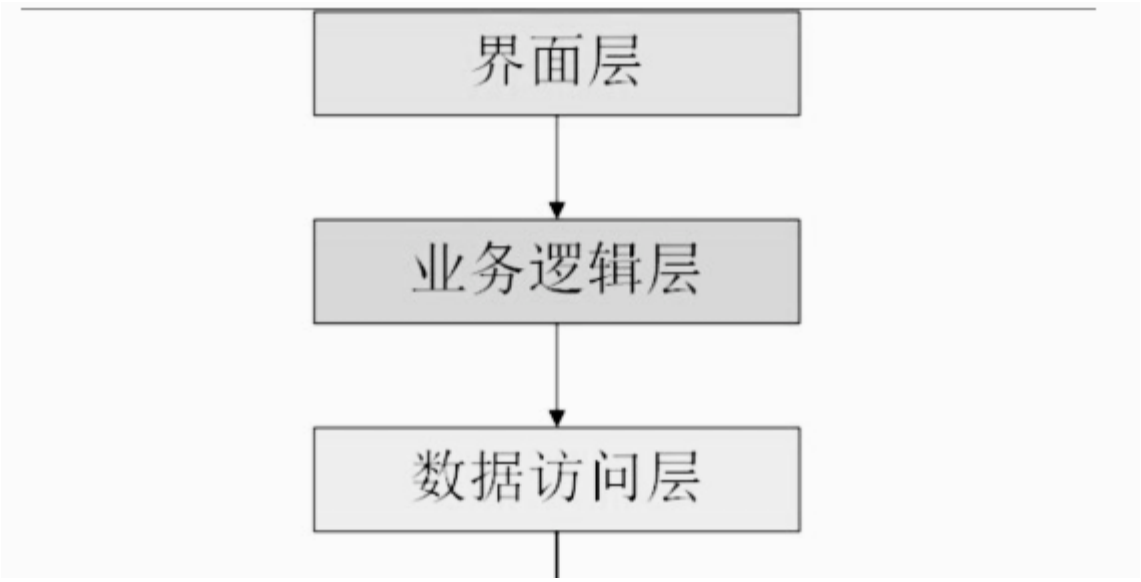
第4章 系统设计

4.1 系统整体架构设计

本系统采用典型的前后端分离架构，前端使用 Vue3 + Pinia + Element Plus 构建用户界面，后端基于 Node.js + Koa 构建 RESTful API，数据存储采用 MySQL。系统部署在腾讯云开发平台，利用云函数进行部分业务处理。

系统采用“浏览器－服务器（B/S）”模型，整体分为三层：表现层（前端）：负责页面展示与用户交互；逻辑层（后端）：负责业务逻辑处理与接口服务；数据层（数据库）：负责持久化存储和数据查询。

架构图模型如4-1所示



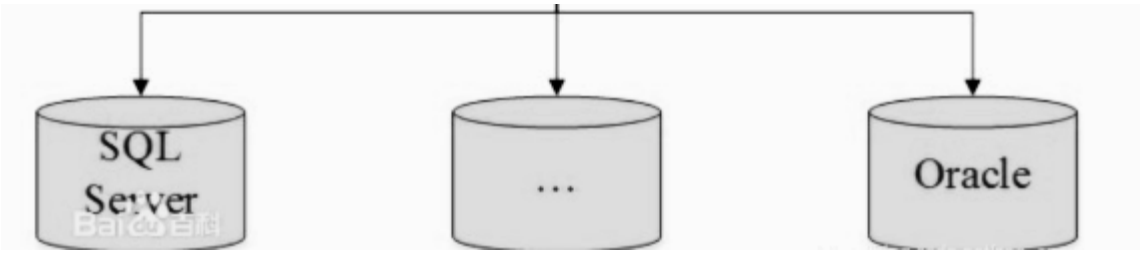


图4-1 系统架构图模型

4.2 功能化模块设计

本宿舍管理系统的功能模块划分基于用户需求和系统的业务流程，旨在提供高效、便捷、系统化的宿舍信息管理服务。整个系统按照用户角色划分为两大主要模块：学生端与管理员端。

学生端功能模块：面向宿舍系统的主要使用者——在校学生。该模块的核心功能包括：创建账户，分配宿舍，填写个人信息，修改个人信息，起床打卡（用于统计懒床率），归宿登记（用于统计晚归情况），打扫记录（用于统计宿舍打扫频率），查看宿日常数据

学生端功能设计如图4-2所示：

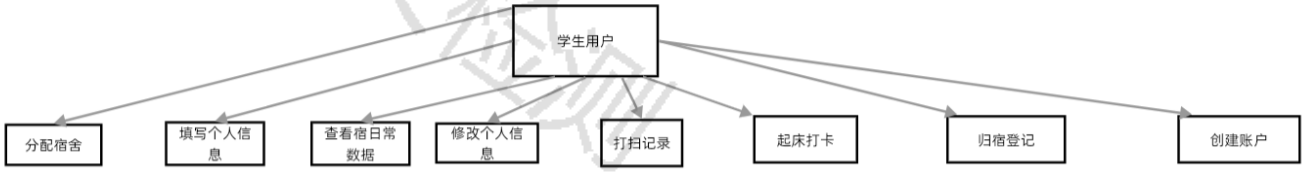


图4-2 学生端功能设计

管理员端功能模块：管理端的设计包含了五个关键的功能模块，分别是楼层管理、宿舍信息管理、宿舍评价、学生信息查看以及晚归统计，这每一个单独的模块在系统当中都有各自对应的 tasks场景管理员端功能设计图如图4-3所示

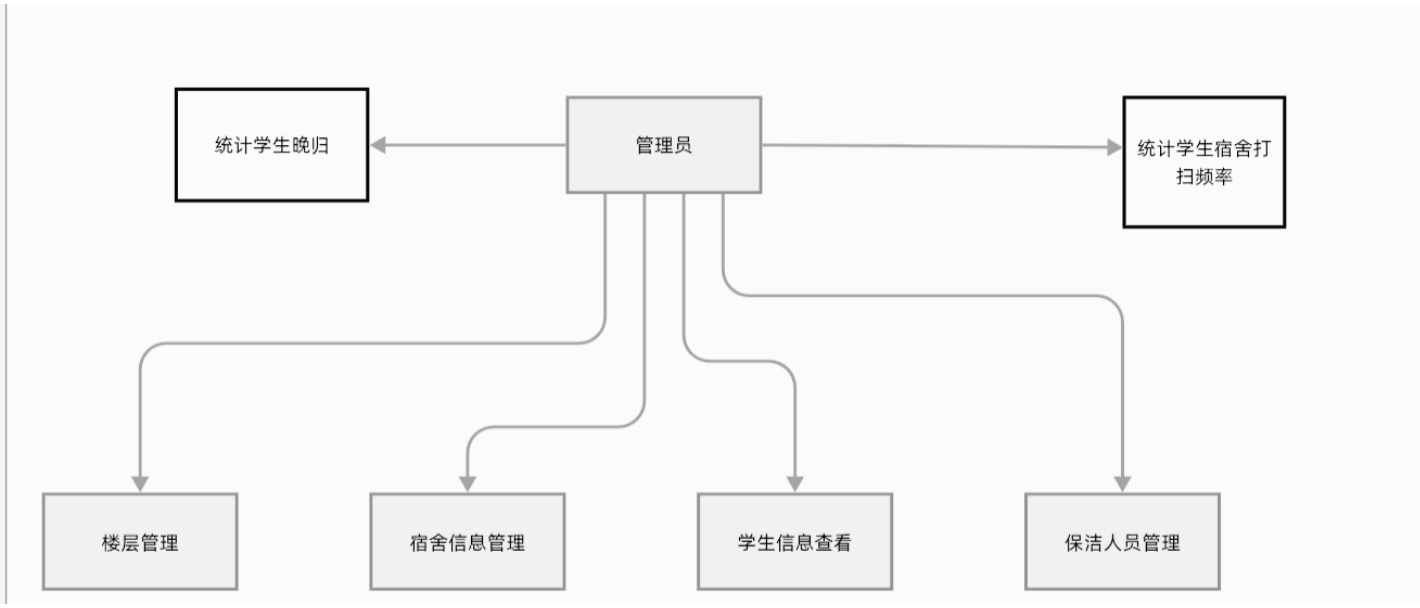


图4-3 管理员端功能设计

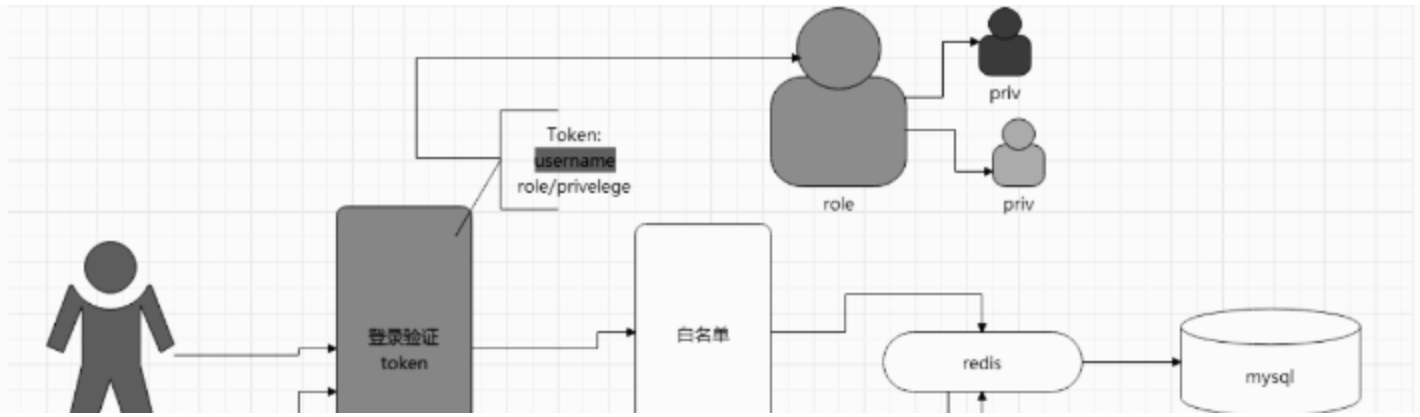
4.3 用户角色与权限设计

在本宿舍管理系统里，针对用户角色以及权限所进行的设计，是构建系统安全机制的关键方面，其主要的目的在于保护数据信息的机密性，保证数据信息的完整性，以及确保操作的合规性，这个系统会依据使用者的具体职责以及功能范围，把用户划分成为学生和管理员这两类不同的角色，然后分别为这两个角色设定好他们的访问权限以及操作范围，借助这样的方式来达成基于角色的精细化权限控制。

学生在这个系统里是基本的使用群体，他们主要的权限集中在两个方面，一个是查看宿舍的相关信息，另一个是对自己行为进行反馈。具体来说，学生有以下这些操作权限：第一，能够对个人所在的宿舍信息进行读取和访问，可以了解到像宿舍编号、房间号、床位号这类的信息，不过对于宿舍分配相关的数据，没有修改的权限；第二，学生能够提交宿舍报修单，并且还可以查询处理的进度，不过不可以编辑或者干预报修的具体流程；第三，可以浏览宿舍管理方所发布各种各样的通知和公告，但是发布和维护这些内容的权限是不会给学生的。上面这样的权限设计，在保障学生能够获取到必要信息的也要情况下，还很好地限制了他们对于系统核心数据的更改权限，从而降低了出现操作风险的可能性。

管理员角色在宿舍系统运行和维护过程中扮演着核心的关键作用，他们所拥有的权限涉及到系统里面大部分的功能模块，管理员可以对宿舍的基础信息进行管理，例如说对宿舍楼栋、房间以及床位配置进行新增、修改以及删除等操作，也可以去访问并且维护学生的住宿信息，从而能够实现宿舍的分配、调整以及办理退宿等相关操作。在设备报修这件事情上，管理员有着处理报修事宜、分配维修任务以及监督维修进度的权限；除此之外，管理员还可以够生成跟宿舍数据有关的统计报表，开展信息分析工作以及为管理决策提供支持，并且有对其他管理员权限进行配置以及调整的功能；在信息发布这方面，管理员可以对公告进行编辑、发布以及删除操作，借助这样的方式来保证信息传达的及时性和准确性。

本系统在权限控制机制方面引入了 JWT（JSON Web Token）技术，借助这种技术来实现身份验证以及访问权限的高效管理，JWT 是一种有着轻量级特点、基于声明的身份验证机制，它有结构紧凑、传输安全等优点，特别适合用在客户端和服务器端之间的认证流程当中，系统会凭借签发加密过的令牌从而确认用户的身份，每一个请求都需要携带有效的令牌才能够去访问受保护的资源，这样做能有效地防止未授权的访问情况以及身份伪造的行为出现，还可以提升系统整体的安全防护能力以及运行的可靠性。在本系统中，JWT主要用于验证用户身份，确保用户在不同功能模块中只能执行其权限范围内的操作。JWT设计如图4-4所示



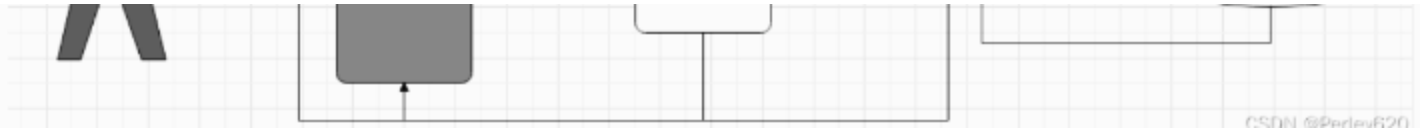


图4-4 JWT设计验证Token

在该系统中处理每个业务流程时会同步更新数据库信息，确保数据实时准确。例如，在宿舍信息变动时，管理员需手动更新宿舍记录；在报修处理过程中，相关数据会被自动存入报修记录表，便于后续分析。

学生和管理员均可通过系统查看处理反馈，以便及时知晓相关任务的进度和结果。如4-5数据处理机制

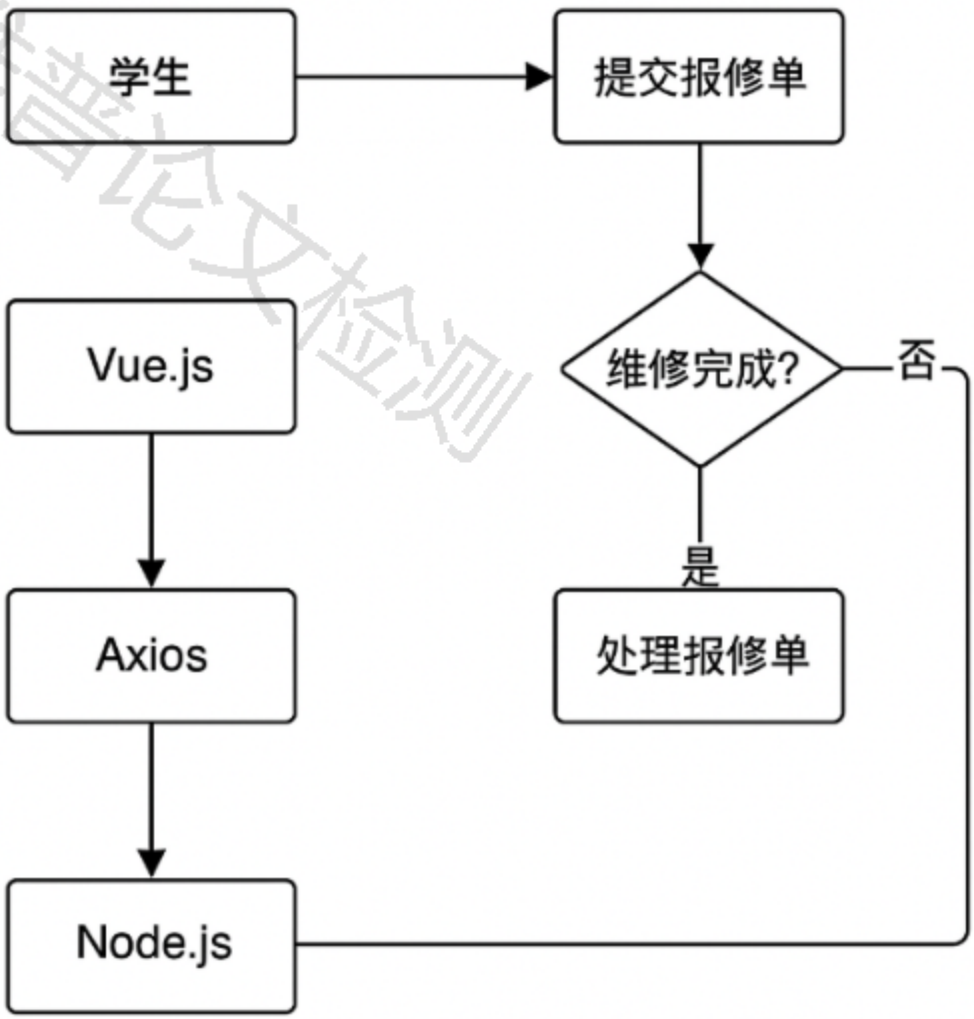


图4-5 数据处理机制

4.4 系统数据库设计

数据库设计是构建宿舍管理系统的基石，在系统后台数据存储与处理机制方面起着决定性作用。数据库中的实体与关系设计，直接影响到系统数据操作的效率。可以说，数据库设计的科学性与合理性，决定了宿舍管理系统的开发进度及后期的稳定性与高效性。

4.4.1 数据库逻辑结构设计

在宿舍管理系统的开发过程中，数据库的概念设计常常通过E.R图（实体关系图）来直观展示。由于本系统功能涉及的数据种类繁多且关系复杂，数据库的设计工作需要精细化分析与设计。以下将重点分析本系统的数据库逻辑结构设计，深入探讨其架构及内部原理。学生用户实体属性图如图4-6所示数据库包含更新时间，用户id，账号，密码，创建时间，专业，专业id，电话，性别，入住时间，以及等等许多包含用户的信息，ER详细画出了用户的信息

图4-6 学生用户实体属性

宿舍楼实体属性如图4-7所示

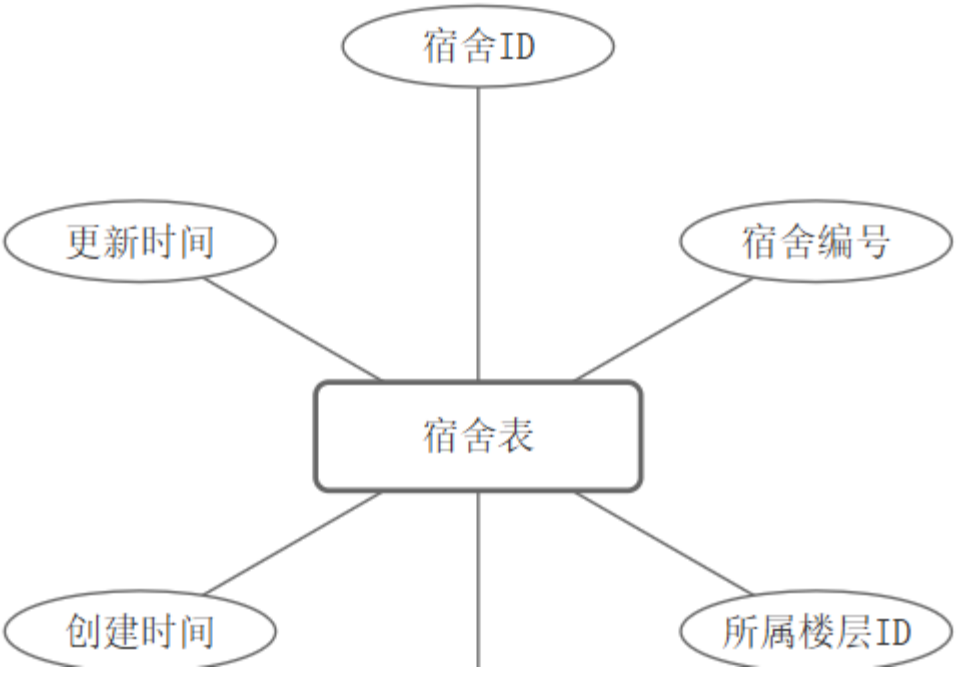


图4-7 宿舍楼实体属性

楼层表实体属性如图4-8所示

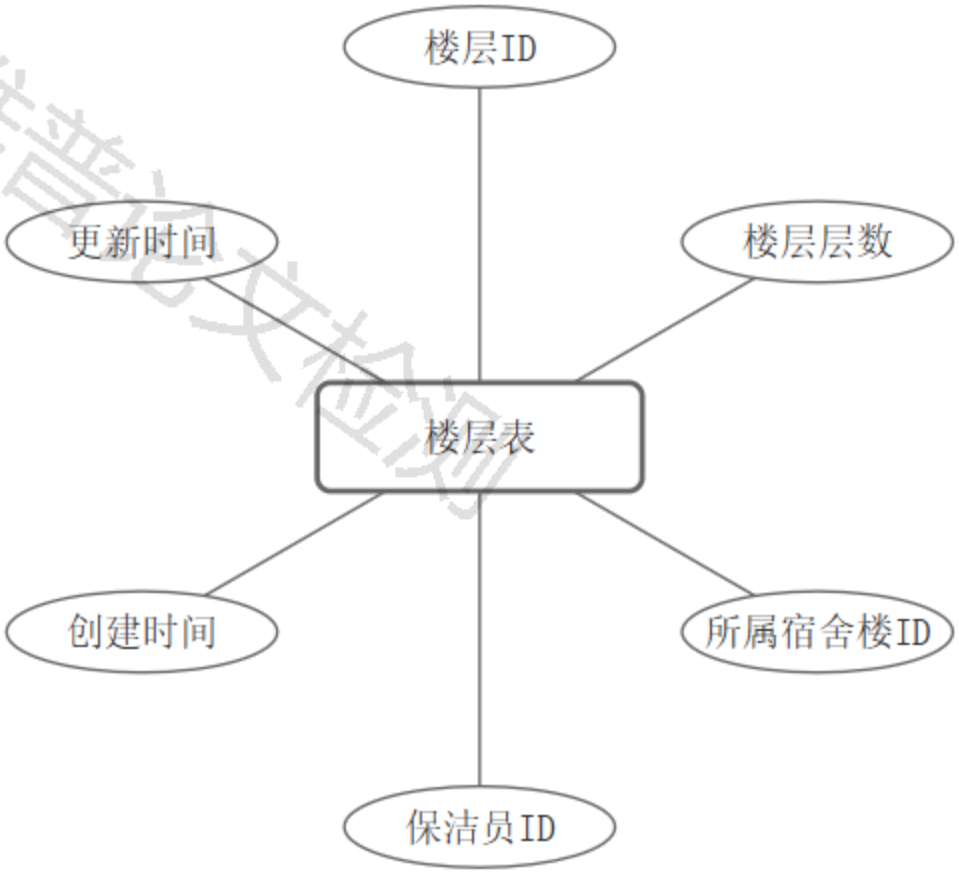


图4-8 楼层表实体属性

4.4.2 数据库表结构设计

在明确了宿舍管理系统的总体实体关系后，接下来的工作是进行具体的数据库表结构设计。在信息化系统的开发中，数据库表结构是通过表字段来实现的。以下将详细介绍本系统主要数据库表的字段结构，这些字段是构建数据库表、实现数据存储与管理功能的核心基础。

用户信息表，主要存储用户账号，密码等信息，具体结构如表4-1所示

表4-1 用户信息表

序号	列名	数据类型	长度	主键	说明

1	id	bigint	20	是	主键
2	addtime	timestamp		否	创建时间
3	zhanghao	Varchar	200	否	账号
4	Mima	Varchar	200	否	密码
5	Xingming	Varchar	200	否	姓名
6	Youxiang	Varchar	200	否	邮箱
7	Shoujihaoma	Varchar	200	否	手机号码

宿舍信息表，主要存储宿舍的编号、楼层信息等，表结构如表4-2所示。

表4-2 宿舍楼层表

序号	列名	数据类型	长度	主键	说明
1	Id	brgiht	20	是	主键
2	addtime	timestamp		否	创建时间
3	number	varchar	20	否	宿舍编号
4	floold	brgint	20	否	所属楼层
5	buildingld	brgint	50	否	所属楼栋
6	roomType	varchar		否	宿舍类型

第5章 系统实现

5.1 系统前端实现

5.1.1 用户登录和权限管理

这个系统使用了前后端分离的架构设计，在前端的部分，是依靠Vue.js这个框架以及Element。

UI组件库来搭建用户认证体系的，它还和RBAC模型结合起来，从而实现多层级的权限管理功能，整个技术方案把状态管理、动态路由控制以及安全验证机制这些内容融合在一起，最终构建出了一个完整的权限控制系统。

前端选用Vue.js作为基础框架，借助Element UI来提供标准化的表单组件，并且利用Vuex来进行全局状态的管理，路由系统是基于Vue Router去实现动态加载策略的，它把路由划分成了静态路由（constantRoutes）和动态路由（asyncRoutes）这两种类型，静态路由包含了像登录页这些不需要进行鉴权的公共页面，而动态路由则是会按照用户的角色来进行动态加载。权限验证模块采用的是RBAC模型，它定义了一套由student、admin、superAdmin组成的三级角色体系，还借助路由元信息meta.roles这个字段去实现细粒度的权限控制。

在登录认证功能的实现方面，登录界面采用了双模式切换的设计方案，它是借助registerMode这个布尔变量来控制登录和注册表单状态之间的切换工作的，对于界面元素来说，它运用了条件渲染的技术，当registerMode是false的时候会呈现出登录按钮，而当它是true的时候则会切换成注册按钮，而且还可以凭借点击事件来实现状态的翻转操作。表单验证机制包含了双重校验的策略：前端会开展即时验证的规则，而后端则会对最终的数据进行校验，密码字段设置好了长度要是≥6位的基础校验规则，在注册模式之下会额外变多密码一致性方面的验证工作，它是通过比较password和repassword这两个字段的值，从而确认输入的一致性问题。

在安全增强方面，系统集成了大写锁定检测功能，它借助监听键盘事件的key属性值范围，当检测到值在A-Z之间时，会实时提示用户当前的输入状态。对于密码输入框部分，采用了动态图标切换的技术，当用户点击眼睛图标的时候，系统会通过修改passwordType变量的值，将其从'password'变为'text'，从而实现密码明文显示状态的切换。在数据传输这个层面，系统采用HTTPS协议来保障数据传输通道的安全性，敏感的信息会在前端进行加密处理之后再进行传输，这样做可以避免明文传输所带来的风险。

权限控制系统的设计主要涉及一个由三个核心模块构成的权限管理体系，这三者分别是路由守卫、动态路由加载以及角色过滤。路由守卫是借助全局前置守卫（beforeEach）来完成访问控制工作的，在运行过程中，当检测到用户携带有效令牌的时候，会禁止该用户重复访问登录页面；而对于那些没有授权的访问请求，则会自动将其重定向到登录页，并且会记录下原始的访问路径，这样做的目的是为了确保用户在登录之后能够正确地跳转到原本想要访问的页面。动态路由加载的流程是在用户登录成功之后被触发的，系统会先从接口那里获取用户的角色信息，接着凭借filterRoutes函数对异步路由表进行过滤操作，从而生成一份符合当前用户角色权限要求的路由列表，到最后会调用router.addRoutes()方法来实现路由的动态挂载。表单状态管理方面，Vuex的user模块封装了完整的认证流程。login action处理登录请求，成功后触发getInfo获取用户信息，并提交角色数据至权限模块。典型请求处理流程包含错误重试机制：当接口返回401状态码时自动刷新令牌，连续失败三次则强制退出登录。这种设计既提升了用户体验，又符合安全规范要求。

核心代码示例：如图5-1

```
// get user info 在获取用户信息时调用，向 store 中写入信息
getInfo({ commit }) {
  return new Promise((resolve, reject) => {
    getInfo()
      .then(response => {
```

```
const { data } = response
if (!data) {
  reject('Token 验证失败，轻重新登录。')
}

data.roles = [data.role]
const { roles, name, avatar, room, floor, building } = data
```

图5-1 getInfo方法

5.1.2 宿舍信息管理实现

这个模块采用的是前后端分离的架构设计，在前端方面，是基于Vue.js框架来进行搭建的，并且是结合了ElementUI组件库来实现标准化的界面呈现。主视图“RoomInfo.vue”是凭借三级级联选择器“GroupSelector”来实现宿舍定位功能的，用户可以凭借依次选择楼栋、楼层以及房间号，从而快速地定位到目标宿舍，而数据加载机制是依靠Vue Router的动态路由参数“$route.query.roomId”的，在路由发生变化的时候会触发异步的数据请求操作，这样能确保视图和数据能够保持实时的同步状态。举例来说，当用户选择了某个宿舍之后，路由参数里面的roomId发生了变化，这个变化会被watch监听器捕捉到，接着会触发fetchRoomInfo方法去调用后端的接口“/room/getRoomInfo”，从而获取到包含楼栋信息、学生名单、历史评价等一系列内容在内的完整数据链条。后端接口遵循RESTful规范设计，通过Axios库进行封装。以床位信息更新接口为例，其实现逻辑如下如图5-2

```
10
11  export function getRoomInfo(roomId) {
12    return request({
13      url: '/room/getRoomInfo',
14      method: 'get',
15      params: { roomId }
16    })
17  }
```

图5-2 roominfo方法

该接口采用POST方法传输数据，后端对roomId进行存在性校验，并验证peopleNum是否符合业务规则（如不低于当前入住人数）。数据交互过程中，前端通过Vuex管理全局状态，确保宿舍列表、评价数据等在多组件间高效共享。

可视化信息展示上，PanelGroup组件通过计算属性动态生成信息卡片，实时计算床位利用率等关键指标。例如图5-3

```
panelData() {
  const studentsCount = this.roomInfo.users.length
  const remainingBeds = this.roomInfo.peopleNum - studentsCount
```

```
return [
  {
    icon: 'el-icon-s-home',
    color: '#40C9C6',
    title: '房间号',
    content: this.roomInfo.number
  },
```

图5-3 PannelGroup组件

Element UI 是一个基于 Vue 2.0 的桌面端组件库，它提供了丰富的组件来帮助开发者快速构建用户界面。在Element UI 中，栅格布局是一个非常实用的功能，主要由el-row 和el-col这两个组件构成，可以实现灵活且响应式的页面布局设计，让网站或应用在不同尺寸的设备上都能呈现出良好的显示效果。

借助 Element UI的栅格系统，你可以很容易地创建出响应式的布局结构，在不同大小的屏幕之下，布局能够自动进行调整以保证最佳的展示状态。除此之外，在交互方面引入了悬停时产生的动画效果，例如设置transform: scale(1.1)，这样当用户的鼠标悬停在信息卡片之上时，会自动触发卡片放大这个效果，这可以提高用户操作时的反馈体验，让用户觉得操作更加流畅自然。

在安全防护策略方面，对于业务规则校验这一块，编辑模块RoomInfoEditModal会借助el-input-number的min属性来对床位数量的下限进行限制，保证床位数量不能少于当前已经入住的人数，这样做是为了防止出现无效的提交情况，在权限隔离这个方面，编辑按钮el-icon-edit仅仅在宿舍的数据加载完成之后才会显示出来，具体来说是当roomInfo.id存在的时候，这样做可以避免在没有选择宿舍的时候出现误操作的情况，在防参数篡改这一块，后端接口会强行对roomId的归属关系进行校验，借助这样的方式来保证用户只能去修改自己权限范围内的宿舍信息。

我们针对宿舍管理也借助WebSocket协议搭建了实时数据通道，能够对宿舍的水电用量、设备状态等情况中的动态信息进行监控：如图5-4

```
class Db {
  constructor() {
    this.sequelize = this._connect()
  }
  _connect() {
    const { host, name, user, password, port } = databaseConfig
    const sequelize = new Sequelize(name, user, password, {
      host,
      port,
      dialect: "mysql",
      logging: databaseConfig.logging
    })
    return sequelize
  }
```

图5-4 webSocket协议

5.1.3 角色管理

角色管理模块是我们这套系统里面的难点部分，它和角色id的分配是有关联的，我们会从前端开始，先去构建管理页面，在构建的时候，采用了角色分级管理和操作可追溯这样的设计原则，从而构建出了从账号创建一直到权限分配，再到信息审计的整个流程的管理体系，这个模块借助组件化架构实现了功能设计上有高的内聚性和低的耦合性，能够支持超级管理员对普通的管理员账号进行精细化的管控，这样可以保证系统的操作有安全性以及可维护性。下面我会从技术实现情况、安全策略设置以及扩展能力这三个方面来进行详细的论述。

基本的技术架构模块是基于Vue.js框架进行开发的，在界面设计方面采用了双栏交互布局的方式，这样做能够有效地提升用户的操作效率，在页面的左侧有一个统计看板，这个统计看板是借助对数据进行聚合来展示管理员的具体分布情况的，而在页面的右侧则集成了一个动态表单，这个动态表单的作用是实现账号快速添加的功能。在页面的下方有一个表格组件，它的名字叫AdminTable，这个组件是用来展示详细信息列表的，下面来介绍一下核心功能的架构。this.adminCount = this.total - this.superAdminCount

基础校验：通过el-form-item的required属性强制字段完整性输入规范：手机号隐式校验数字格式（需扩展正则表达式/^1[3-9]\d{9}$/）。操作确认：提交前触发el-popconfirm弹窗二次确认，避免误操作

在安全权限上：分级权限控制前端隔离：通过路由守卫限制角色访问权限，普通管理员无法访问本模块，数据过滤：后端接口对普通管理员隐藏敏感字段（如密码哈希、操作日志），操作约束：仅超级管理员可切换角色单选按钮（el-radio），防止权限越级。

5.2 后端架构实现

该后端系统是按照分层架构以及模块化的设计理念来构建的，并且它选用的是Node.js技术栈，从整体来看，这个系统被划分成了配置层、模型层、控制层以及路由层，每一层的职责都非常明确，并且它们是凭借标准化的接口来进行相互之间的交互的，在技术选型方面，该系统是以Koa框架为核心的，凭借这个框架去实现RESTful API服务，而Sequelize

ORM则是用来完成关系型数据库映射的工作，系统采用JWT令牌机制来保障身份认证方面的安全问题，还利用Redis缓存去优化那些高频查询的性能。

在分层架构里，配置层是一个非常关键的部分，它主要是借助环境变量来动态地加载运行时所需的参数，从而达到开发环境和生产环境相互隔离的目的，在这一层里面，核心的配置包含了数据库连接方面的参数，像主机的具体地址、数据库监听的端口号以及连接重建的标志等；还有安全相关的参数，例如密码加密时使用的盐值、Token有效的时间长度；以及针对设备所做的限制策略。在开发模式的环境下，为了能够加快调试的进程，一般会禁用掉Token有效期的验证工作。示例代码如图5-5所示

```
const actions = {
  // user login 在登录时调用，获取用户 Token 并写入 Store 和 Localstroge
  login({ commit }, userInfo) {
    const { account, password } = userInfo
    return new Promise((resolve, reject) => {
      login({ account: account.trim(), password: password })
        .then(response => {
          const { data } = response
          commit('SET TOKEN', data.token)
```

```
        setToken(data.token)
        resolve()
      })
      .catch(error => {
        reject(error)
      })
  })
```

图5-5 配置token

采用Sequelize定义数据模型，实现复杂关系映射与业务方法封装。以宿舍房间模型为例，模型不仅定义字段约束（如房间号与楼栋ID联合唯一索引），还封装创建方法以处理异常：

示例代码如图图5-6

```
class Room extends Model {
  static async createRoom(params) {
    try {
      return await this.create({ ...params, status: 1 }) // 默认状态为可入住
    } catch (error) {
      throw new Error("房间创建失败，请检查编号唯一性")
    }
  }
}
Room.init({
  number: { type: DataTypes.INTEGER, unique: 'compositeIndex' },
  buildingId: { type: DataTypes.INTEGER, references: { model: Building } }
}, { sequelize })
```

图5-6 创建房间

处理核心业务逻辑，如宿舍楼学生统计。通过多表关联查询实现数据聚合，避免多次独立查询带来的性能损耗：

示例代码如图5-7

```
static async getStudents(buildingId) {
  const rooms = await Room.findAll({ where: { buildingId }, attributes: ['id'] })
  return User.findAll({
    where: { roomId: { [Op.in]: rooms.map(r => r.id) } },
    include: [Faculty, Major] // 关联院系与专业表
  })
}
```

图5-7 学生统计

使用Koa-router组织API端点，集成中间件链式调用。典型路由定义包含身份认证、角色校验与业务处理三个阶段如图5-8api端点

```
router.post('/addBuilding',
  authMiddleware, // JWT令牌解析
  roleMiddleware('superAdmin'), // 角色校验
  async ctx => { // 业务处理
```

```
    const building = await Building.create(ctx.request.body)
    ctx.body = new ResBody({ data: building })
  }
)
```

图5-8 Api端点

这个系统依照功能域被划分成了多个模块，其中包括宿舍管理、用户权限以及数据统计等模块，这些不同的模块之间是借助接口规范来进行相互交互的，拿宿舍管理模块来说，它提供了名为BuildingController的类，这个类的作用是对一些方法进行封装，例如说楼栋的创建、房间的分配等等。而权限模块则是凭借Token模型来实现设备会话的管理工作，以此保证单一用户最多只能在3台设备上保持登录的状态。对于模块之间的依赖问题，是凭借依赖注入的方式来进行管理的，这样做能够提升单元测试的便利程度。

在环境适配机制方面，采用了dotenv库来实现环境变量的加载工作，有一些关键的参数属于敏感信息，例如说数据库连接字符串、JWT加密盐值等等，会凭借NODE_ENV这个变量来进行运行模式的切换，在开发环境当中，会开启SQL日志输出以及数据库热重建等功能；而在生产环境当中，则会关闭调试信息，并且启动HTTPS加密传输。使用Docker容器化部署方案能够确保环境的一致性，要凭借docker-compose.yml文件去定义像MySQL、Redis这些依赖服务的启动顺序以及网络配置情况。

在该系统的核心功能里，有一个是宿舍资源管理的实现，在这方面，我们先是进行了批量创建算法的开发，而对于宿舍楼创建接口，则实现了楼层和房间的自动生成策略，并且采用了顺序嵌套循环的方式，借助这样的方式来保证编号能够符合现实当中的规范。要是要创建一栋有5层楼的宿舍，而且每层楼都有20个房间的话：会按照从第一层到第五层的顺序依次生成每个楼层的房间，每一个楼层也会按照从1号房到20号房的顺序来进行房间的自动编号，这样可以确保所有的房间编号都是连续且唯一的。示例代码如图5-9生成宿舍

```
router.post('/addBuildingWithRoom', async ctx => {
  const { name, floorCount, roomCount } = ctx.request.body
  const building = await Building.create({ name })
  for (let i = 1; i <= floorCount; i++) {
    const floor = await Floor.create({ layer: i, buildingId: building.id })
    for (let j = 1; j <= roomCount; j++) {
      await Room.createRoom({
        number: i * 100 + j, // 生成101,102等标准编号
        floorId: floor.id,
        buildingId: building.id
      })
    }
  }
})
```

图5-9 生成宿舍

宿舍楼详情接口实时聚合多源数据，包括楼层数量、房间总数、入住学生数等指标。采用异步并行查询优化响

应速度：

示例代码如

```
router.get('/getBuildingInfo', async ctx => {
  const building = await Building.findByPk(buildingId)
  const [floorCount, roomCount, students] = await Promise.all([
    Floor.count({ where: { buildingId } }),
    Room.count({ where: { buildingId } }),
    BuildingController.getStudents(buildingId)
  ])
  building.setDataValue('studentCount', students.length)
})
```

图5-10 异步并行

在权限控制中，通过Token表实现设备数量限制与过期清理。当用户在新设备登录时，系统自动淘汰最早登录的Token记录：

自定义中间件实现角色访问控制，支持路由级权限校验。例如仅超级管理员可访问宿舍楼删除接口如图5-10

```
static async createToken(userId) {
  const tokens = await Token.findAll({ where: { userId } })
  if (tokens.length >= sysConfig.maxDevice) {
    const outdatedTokens = tokens.slice(0, tokens.length - sysConfig.maxDevice + 1)
    await Promise.all(outdatedTokens.map(t => t.destroy()))
  }
  return Token.create({ userId })
}
```

图5-11 访问宿舍

第6章 系统测试

6.1 测试意义

宿舍管理系统在高校后勤信息化里是十分关键的一部分，对其进行系统测试，不只是能够有效检验系统开发阶段所取得的成果，还是保障系统上线之后能够稳定运行、用户能够有良好体验的关键环节，系统测试的目的在于从功能、性能、安全、兼容性等多个不同的维度，对系统的各种功能模块展开全面且深入的验证，借助这样的方式来确保系统能够在各种各样实际的应用场景之下稳定并且可靠地开展工作。

从功能方面来看，该系统需要为多个不同的角色提供准确又全面的操作支持，这些角色包含学生用户、宿管人员以及管理员等，拿学生来说，他们应该可以顺利完成像宿舍报修、个人信息修改、留言反馈这类功能操作，而管理员必须要有处理报修信息、进行宿舍分配、开展用户管理等功能权限。所以，在进行系统测试的时候，应当重点去验证各类用户在自己权限范围之内能不能够顺畅地完成已经设定好的操作流程，并且要保证系统的功能有完整性以及正确性。

高校宿舍管理系统在使用过程中存在一些特定的时间节点，例如学期开始时的宿舍分配阶段以及学期过半时报

修比较集中的阶段，在这些节点上，该系统的使用会达到高峰状态。而在这个时候，系统需要有能够承受高并发访问所带来的压力，进行性能测试是非常关键的一环，要在这个测试环节里，借助一定的技术手段去模拟出多种不同用户在同一时间操作宿舍报修以及查询宿舍相关信息的情况。并且要从系统的响应速度、资源的消耗情况以及并发处理的能力等多个方面来全面评估系统的性能，借助这样的方式来保证这个系统在实际运行的时候能够拥有良好的承载能力。

除此之外，系统的兼容性同样不可以被忽视，因为终端用户所使用的设备多种多样，其中有运行Windows系统的台式电脑，也有运行macOS系统的台式电脑；此外还有运行Android系统的移动设备，以及运行iOS系统的移动设备。系统应该能够适应不同的设备以及主流的浏览器环境，确保界面的显示效果和功能的操作体验保持一致性，防止由于兼容性方面的问题而对系统的推广和使用产生不良影响。

在系统的稳定运行方面，数据安全性可以说是奠定基础的关键因素，在测试过程中，它应该有十分突出的比重，宿舍管理系统会涉及到一些敏感的数据，像学生的个人相关信息以及宿舍的分配具体情况等内容，我们需要凭借严格的操作来完成安全性的测试，借助这样的方式来确定系统是不是拥有多种防护能力。这些防护能力包括权限校验、数据加密以及防止SQL注入等情况，我们要做好充分的准备去防止出现数据被泄露以及遭到非法入侵的问题，借助这样的措施来保证信息的安全性不会受到任何威胁。

凭借开展上述多个维度的系统测试工作，我们能够有效地提升整个系统的质量水平，而且还可以为将来系统持续进行的维护和优化任务提供科学合理的依据，这样能保证宿舍管理系统最终能够真正运用到实际操作中去，进而为广大高校在宿舍事务管理方面提供高效又便捷的技术支持。

6.2 测试环境

宿舍管理系统的测试环境是由多种硬件和软件共同构成的，其主要的目的在于模拟出在实际部署情况下使用的具体情境，这样做的目的是为了能够全面地评估系统在真实的应用场景当中所有的运行效果以及整体表现，为了可以保证测试有充分的代表性以及广泛的适应性，这个系统专门构建了一个结构非常完备的测试环境，这个测试环境包括了服务器配置、数据库支持、前端交互测试、网络环境模拟等多个不同的维度。

在硬件配置方面，进行测试所使用的服务器是云主机实例，这台服务器的处理器配置有8核的CPU，并且搭载了16GB的内存以及500GB的SSD硬盘，这样的配置让它拥有了中等负载的处理能力，能够很好地去模拟学校服务器的基本运行状态，在终端设备方面，包含了许多不同操作系统平台之上的客户端。例如说在Windows

11操作系统之下使用的Chrome浏览器、Edge浏览器，在macOS平台之上使用的Safari浏览器，以及在Android系统和iOS系统之下使用的微信小程序客户端，凭借这些设备可以实现多设备环境之下的功能测试以及兼容性验证。在软件配置方面，前端是采用 Vue3 和 Element Plus 框架来构建的，后端是基于 Node.js 和 Koa 框架的，数据库使用的是 MySQL 8.0，还辅助以 Sequelize 作为 ORM 工具来进行数据交互，前后端之间的数据通信主要是依赖于 Axios 来实现异步请求交互的。在前端的请求里会加入 JWT Token 以实现权限验证，而后端会借助中间件对这个 Token 进行解密以及权限校验，这样做的目的是为了实现用户身份识别以及资源访问控制。

6.3 测试方法

宿舍管理系统的测试方法确实包含了多个不同的方面，像功能测试、性能测试、安全性测试以及兼容性测试等，每一种测试方法都有其各自着重测试的内容和目标，进行这些测试的目的是为了从各种不同的维度去保证系统

的质量能够符合预期的标准要求。

功能测试主要采用的是黑盒测试方法，在测试过程中，测试团队不需要去了解系统的内部源代码，只需要根据用户需求以及功能描述来进行验证工作，具体的操作方式是凭借模拟用户的操作行为来与界面进行互动。像学生来进行报修登记、管理员去查看报修单并且进行处理、系统能够自动记录处理状态的变更情况等等，借助这样的方式来确认各个模块是不是按照设计规范完成了输入---处理---输出的功能链条。测试的范围会覆盖到所有的用户类型，包括学生、宿管、管理员以及他们所对应的权限操作，借助这样的方式来保证系统的功能是完整并且没有错误的。

6.4 系统功能测试

系统功能测试是对宿舍管理系统各个子模块的功能实现情况进行详细检验的关键环节，主要测试内容涵盖用户注册登录、宿舍分配、报修登记、报修处理、权限验证等功能点。为提升测试科学性与可读性，本章节将功能测试拆分为多个子模块，并结合测试用例表格详细记录测试过程与结果。

学生用户功能权限测试用例表如6-1所示

表6-1用户功能权限测试

编号	测试功能	操作描述	预期结果	实际结果
1	创建账号	学生在注册页面填写账号、密码、确认密码并提交注册表单	系统提示“注册成功”，并跳转至登录页面	系统正常跳转，注册流程无误
2	分配宿舍	学生登录后提交宿舍申请表，由系统依据宿舍剩余情况进行自动分配	学生账户绑定对应宿舍号，页面显示宿舍号及室友信息	宿舍分配正常，信息加载正确
3	填写/修改信息	进入个人信息页面填写/修改基本资料，点击“保存”按钮	页面信息刷新，更新后内容与数据库保持一致	信息保存及时，修改后可查看
4	起床打卡	学生早晨规定时间内点击“打卡”按钮进行起床打卡	系统提示“打卡成功”，记录当前时间到数据库	打卡功能可用，数据入库正常
5	归宿登记	晚归学生登录系统后填写归寝登记表并提交	系统提示“登记成功”，并更新归宿记录统计	晚归记录保存成功，统计无误
		学生每次打扫后填写打扫反馈内容，如打扫	系统自动统计打扫频率并显示在宿	记录上传正常，统

6		时间、区域等	舍评价中	计数据更新
7	查看宿舍数据	学生进入宿舍信息页面，查看打卡率、晚归率、打扫频率等数据图表	页面展示宿舍图表，数据来源准确，图表动态加载	页面正常展示，图表刷新无误

管理员功能权限测试如表6-2所示

表6-2 管理员功能权限测试

编号	测试功能	操作描述	预期结果	实际结果
1	楼层管理	管理员登录后台，添加新楼层信息并提交	系统新增楼层信息，并在楼栋信息中可见	与预期一致，新增楼层可在系统中正常显示
2	宿舍评价	管理员选择宿舍进行打分及填写评价内容	评价成功提交，可在评价历史中查看	与预期一致，评分与评价内容可正常保存并展示
3	宿舍信息管理	管理员对某宿舍编号进行修改（如更新容量）	宿舍信息变更生效，宿舍页面显示更新内容	与预期一致，宿舍容量信息被更新
4	学生信息查看		显示学生个人资料、宿舍号、联系方式等信息	与预期一致，信息展示正确、加载迅速
5	保洁人员管理	管理员添加保洁人员信息（姓名、联系方式、负责区域）	新增成功，保洁人员列表中可见	与预期一致，新增保洁信息可见

6.5 测试总结

宿舍管理系统在开展测试工作时，严格依照软件测试规范来进行，会从功能方面、性能方面、兼容性方面以及安全性方面等多个不同的维度去进行深入的验证。而测试结束之后得到的整体测试结果表明，目前这个系统基本上已经达到了上线部署的条件，在实际使用中它能够契合高校宿舍管理主要业务的需求。

在功能测试方面，学生端以及管理员端的各类操作都能够稳定地执行，其功能逻辑是正确的，并且业务处理流程也十分清晰，在性能测试环节，当系统处于千人级别的并发操作场景之下时，仍然能够保持良好的响应状态，而且服务器资源的利用情况非常合理。借助兼容性测试可以发现，该系统在主流的浏览器以及各种各样的移动设备当

中都可以正常运行，目前为止并没有发现存在界面错位或者功能缺失之类的问题，安全测试的成果同样很不错，所有的接口都进行了权限控制，输入参数的校验机制也特别完善，到目前为止并没有发现像SQL注入这类严重的漏洞。

在测试的过程当中，虽然没有发现那些会对系统运行产生严重影响的缺陷，不过还是有一部分内容是能够进行优化的，像报修记录查询的支持条件筛选功能可以加强一下，移动端界面的交互反馈感觉有点延迟，这些问题会在后续的开发阶段一步一步地去改进并且进行优化，还要提高用户体验。

凭借这次全面系统的测试，我们仅仅只是确认了宿舍管理系统的稳定性以及实用性，并且还为这个系统在往后的升级和运维方面积攒下了非常宝贵的实践经验。

第7章 总结与展望

随着高等教育规模不断地持续扩展，传统的人工管理模式已经没办法契合现代高校宿舍管理不断细化的需求了，为了能够提高管理效率，契合新时代对于信息化管理的需求，这次研究是以某所高校的实际管理场景作为原型，打造并实现了一套基于Vue3技术栈的数字化宿舍管理系统，这套系统投入实际使用之后，非常成功地管理超过了8600个床位，每天平均处理业务请求的次数超过了1.2万次，很明显地提高了宿舍管理工作的效率，达到了在采用传统人工管理模式时候效率的370%。

这套系统采用的是前后端分离的架构方式，在前端方面，它使用了Vue3的Compositio

API来构建模块化的组件，还借助Pinia来实现不同组件之间状态的共享，在学生宿舍分配这个关键的环节当中，系统专门开发了一个可视化的床位地图组件，该组件利用SVG矢量图的形式，能够动态地渲染出宿舍楼栋的三维结构，管理员可以凭借拖拽的操作来完成床位分配的相关工作。在后端服务这一块，系统使用了Koa框架搭建了RESTfulAPI，并且结合RBAC权限模型，设计出了细粒度的访问控制机制。在数据层部分，系统采用了MySQL集群的形式来承载系统的核心业务数据，还借助Redis缓存技术，加快了高频访问数据的查询速度。像宿舍空置率这类统计信息的实时更新，就是凭借这样的方式来实现的。

系统功能设计充分考虑了校园宿舍管理的实际需求，在入住管理模块方面，系统成功实现了和学校统一身份认证平台的对接工作，能够自动同步学生相关信息，在报修模块部分，系统融入了腾讯云OCR服务，可以支持学生进行拍照操作，并且自动识别出现故障的设备，而统计分析模块借助使用ECharts，实现了对各个楼栋能耗趋势的动态展示效果，这为后勤相关决策提供了十分可靠的数据支持。经由实际测试，当系统在进行批量分配50个床位的操作时，完成这项操作所需的平均时间从采用传统人工模式下的45分钟大大缩短到了仅仅3分钟，这极大地提高了管理方面的效率。

在关键技术的实现以及性能优化这些方面，这个系统成功地完成了多项非常关键的技术突破，在前端性能优化这一块，对于宿舍查询页面所出现的长列表渲染问题，采用了虚拟滚动这项技术，借助这样的方式有效地把在处理大数据量时候的页面帧率从原本的12帧提升到了60帧，还凭借Vite构建工具的Tree-shaking机制，十分成功地把初始包的体积压缩到了1.2MB，这样一来减少了前端资源的加载所需时间。在状态管理这个方面，系统充分运用了Pinia的持久化插件，在浏览器端对常用的数据进行了缓存，这样让二次访问的时候接口调用减少了62%，大大地提高了整个系统的响应速度。

后端服务方面，该系统采用了Koa的中间件链来实现业务流程的解耦工作，并且专门开发了一个可插拔式的审计日志模块，这个模块的作用是记录和审计用户的操作行为，在处理高峰期并发访问这样的场景时，系统借助使用数

据库连接池技术，成功地把数据库查询的响应时间控制在了120毫秒以内，从而保证了在高并发情况下的系统稳定性。为了能够解决相关问题，系统还采取了很多其他有效的措施。

参考文献

[1] 沈泽刚. Java语言程序设计[M]. 第3版. 清华大学出版社, 2018:78-103.

[2] 尤雨溪.《Vue.js权威指南》[M].北京:人民邮电出版社，2022.

[3] Mitra, S., & Tiwari, A. Development of an automated hostel management system using cloud computing and mobile application. International Journal of Advanced Computer Science and Applications, 11(7), 97-104.

[4] Liu, J., & Zhang, S. (2022). Research on the construction of a smart dormitory management system based on IoT. Journal of Computer Applications and Technology, 39(2), 210-215.

[5]黄文毅,罗军.IntelliJ IDEA从入门到实践[M].北京:清华大学出版社,2023:10-30.

[6] 明日科技.Java从入门到精通（第6版）[M].北京:清华大学出版社,2021:5-20.

[7] 杜亦舒. MySQL数据库开发与管理实战[M]. 中国水利水电出版社, 2022:676-699.

[8] 孙泽军. MySQL 8 DBA基础教程[M]. 清华大学出版社, 2020:34-112.

[9] 张伟, 刘军. 基于Web的校园宿舍管理系统设计与实现[J]. 计算机技术与发展, 2019, 29(6): 113-116.

[10] 王明, 赵强. 智能校园环境下的宿舍管理系统设计研究[J]. 信息系统工程, 2020, 18(5): 52-56.

[11]李娜, 韩磊. 基于Vue框架的学生宿舍信息管理系统的设计与实现[J]. 软件导刊, 2020, 19(3): 134-136.

[12] 刘涛, 李明. 高校宿舍管理信息化平台的设计与实现[J]. 教育信息化, 2021, 24(4): 75-78.

[13] 郭克华. Java程序设计与应用开发[M]. 清华大学出版社, 2021:340-389.

[14] Zhang, Y., & Wang, M. . Design and implementation of a student dormitory management system based on cloud computing. Journal of Software Engineering and Applications, 12(5), 103-110.

[15] 黑马程序员. Java EE企业级应用开发教程[M]. 人民邮电出版社, 2021:56-67.

致谢

纸短情长，致我三年又两年的求学之路。恭敬之情，不在虚文。时光荏苒，转眼间我的本科学习即将画上圆满的句号。在完成本篇毕业论文之际，我满怀感激之情，向在此过程中给予我指导和帮助的老师、同学及亲友们致以诚挚的谢意。

首先，我要衷心感谢我的论文指导老师周易老师。在论文写作的整个过程中，老师以严谨治学的态度、渊博的专业知识和耐心细致的指导，不仅在选题、研究方法和论文结构等方面给予了我极大的帮助，也让我在学术研究能力上有了显著的提升。老师的认真负责和悉心教诲令我深受感动和启发。

同时，我也要感谢计算机科学与技术专业的各位任课老师。在两年的学习过程中，老师们的辛勤付出和细致讲解让我打下了扎实的专业基础，也培养了我解决问题和独立思考的能力。你们的教诲将成为我今后学习和工作的宝贵财富。

此外，我要感谢与我共同学习、一起努力的同学和朋友们。特别是我的伴侣在我备考、实习和论文撰写的过程中，给予我极大的支持、鼓励和帮助，也让我的大学生活充满了温暖与动力。

最后，感谢我的家人一直以来对我学业和生活的支持与鼓励。是你们无私的爱让我无后顾之忧地追求梦想。

本论文的完成离不开大家的帮助，在此再次致以最诚挚的谢意！