

**信息科学技术学院**

**《Web程序设计》**

**学生姓名** 魏鹏超

**学生学号** 2208010423

计算机科学与技术 **专业** 224 **班**

2025 **年** 01 **月** 12 **日**

目录

[学生宿舍管理系统 1](#_Toc1544)

[1 绪论 1](#_Toc1358)

[1.1 国内外研究概况以及发展趋势 1](#_Toc12670)

[1.2 科学学术意义和应用前景 2](#_Toc14042)

[1.3 计算机化管理的优势 4](#_Toc12593)

[1.4 参考文献与项目 5](#_Toc27226)

[2关键技术 6](#_Toc12290)

[2.1 后端技术 6](#_Toc3728)

[2.2 前端技术 9](#_Toc7048)

[3数据库设计 15](#_Toc5413)

[3.1 数据库设计概述 15](#_Toc14191)

[3.2 数据库表结构设计 18](#_Toc4377)

[3.3 数据表关系示意图 30](#_Toc14517)

[4系统实现 36](#_Toc392)

[4.1 系统需求分析 36](#_Toc4043)

[4.2 系统总体设计 51](#_Toc29627)

[4.3 系统详细设计 55](#_Toc243)

[4.4 系统实现 64](#_Toc16032)

[5总结 119](#_Toc25677)

# **学生宿舍管理系统**

## 1 绪论

### 1.1 国内外研究概况以及发展趋势

学生宿舍管理系统对于一个学校来说是必不可少的组成部分。目前好多学校还停留在宿舍管理人员手工记录数据的最初阶段，手工记录对于规模小的学校来说还勉强可以接受，但对于学生信息量比较庞大，需要记录存档的数据比较多的高校来说，人工记录是相当麻烦的。而且当查找某条记录时，由于数据量庞大，还只能靠人工去一条条的查找，这样不但麻烦还浪费了许多时间，效率也比较低。当今社会是飞速进步的世界，原始的记录方式已经被社会所淘汰了，计算机化管理正是适应时代的产物。信息世界永远不会是一个平静的世界，当一种技术不能满足需求时，就会有新的技术诞生并取代旧技术。21世纪的今天,信息社会占着主流地位，计算机在各行各业中的运用已经得到普及，自动化、信息化的管理越来越广泛应用于各个领域。我们针对如此，设计了一套学生宿舍管理系统。学生宿舍管理系统采用的是计算机化管理，系统做的尽量人性化，使用者会感到操作非常方便，管理人员需要做的就是将数据输入到系统的数据库中去。由于数据库存储容量相当大，而且比较稳定，适合较长时间的保存，也不容易丢失。这无疑是为信息存储量比较大的学校提供了一个方便、快捷的操作方式。本系统具有运行速度快、安全性高、稳定性好的优点，并且具备完善的报表生成、修改功能，能够快速的查询学校所需的住宿信息。学生宿舍，是大学生在高校校园里一个重要的学习、生活、交往的空间环境，大学生，大约有2/3的时间是在宿舍环境里度过的。随着高校官学规模的扩大，在校学生的学历层次、思想状态、文化水平呈多样化、复杂化趋势，这在很大程度上给学生管理工作带来许多新的问题。随着学生人数的增多，高校学生宿舍管理的信息量日益庞大，而目前我校学生宿舍管理仍采用传统的工作方式——账本化的工作模式，这种模式获取信息慢，查阅困难，准确性差，很难适应高水平宿舍管理的需要，同时带来了大量的人力的浪费，对于宿管的工作带来很多不便。基于此，我们想到需要开发一个“学生宿舍管理系统”来提高宿舍的管理效率，改善管理方式，使其与现代信息化的趋势接轨。由于数据库的存储容量相当大，而且比较稳定，适合较长时间的保存，也不容易丢失。这无疑是为信息存储量比较大的学校提供了一个方便、快捷的操作方式。

### 1.2 科学学术意义和应用前景

科学技术的不断发展为各个领域的管理提供了更加高效、便捷的解决方案。在教育领域，尤其是高校管理中，学生宿舍管理系统的应用日益普及，成为高校信息化建设中不可或缺的一部分。选择学生宿舍管理系统作为课题，能够充分展示信息技术在现代校园管理中的巨大潜力，尤其是在提升管理效率、保障信息安全以及改善用户体验方面。

宿舍管理系统不仅能使使用者（如学生和管理人员）感到操作非常方便，还能为学校提供强大的数据处理能力。通过现代数据库技术，宿舍管理系统能够有效地将学生信息、宿舍安排、维修记录等大量数据存储在数据库中，系统自动化管理减少了人工干预，避免了传统手工记录方式可能出现的误差和漏洞。这对于大规模的高校尤其重要，因为学生人数多，信息复杂，人工管理难度较大。而通过一个统一的、集成的管理系统，可以大大提升信息处理的效率和准确性。

系统中使用的数据库具有较大的存储容量，并且因其稳定性和高效性，非常适合长期存储和管理学生的住宿数据。与传统纸质档案相比，数据库不仅能容纳更为庞大的数据量，还能通过备份等手段有效避免数据丢失的风险。此外，数据库支持高效的数据检索，使得管理人员能够在需要时快速获取所需信息，如学生住宿安排、宿舍状态情况等，极大地节省了时间和人力成本。

从系统的技术层面来看，宿舍管理系统具有诸多优点，首先是运行速度快，能够在短时间内响应用户的查询请求；其次是安全性高，系统采用了权限控制机制，确保只有授权人员才能访问敏感数据，从而保护学生隐私和数据安全；最后是系统的稳定性好，能够保持长期、平稳的运行，确保了校园管理的顺畅进行。

总的来说，学生宿舍管理系统为高校宿舍管理提供了一个高效、智能的解决方案。它的应用不仅大大提高了管理效率，减少了人工操作的错误，降低了管理成本，而且为学生提供了更加舒适、安全的居住环境，进一步促进了学校信息化建设的深入发展。随着技术的不断创新与发展，未来宿舍管理系统的功能将更加完善，操作更加便捷，必将在高校管理中发挥更加重要的作用。

### 1.3 计算机化管理的优势

目前，许多高校仍在使用传统的账本化手工管理方式，这种方式存在获取信息慢、查阅困难、数据准确性差等问题，已经不能满足现代高校宿舍管理的需求。随着学生人数的增加，宿舍管理所涉及的信息量越来越庞大，人工管理不仅效率低下，而且容易出错。

传统的手工记录方式在查找和更新信息时非常繁琐，尤其是当宿舍情况发生变化时，人工更新和同步不同记录的工作量巨大，容易引发错误，导致数据不准确或过时。这种低效的管理模式难以应对当前高校对宿舍管理精确性和及时性的要求。

在信息社会中，计算机化管理逐渐取代了传统的人工记录方式，成为解决宿舍管理问题的有效途径。通过开发一套学生宿舍管理系统，学校能够实现数据的高效存储和管理，提升管理水平，并且能够快速、准确地查询和更新学生宿舍信息。

学生宿舍管理系统采用数据库技术，不仅能够容纳大量的数据，还能保证数据的安全性和稳定性。数据库的使用能够确保数据不会轻易丢失，并通过备份手段保证数据的完整性。系统能够在极短的时间内响应查询请求，快速获取所需信息，避免了传统方式下的低效和重复劳动。

同时，宿舍管理系统的使用能够有效减少人力成本。管理人员只需将学生宿舍的基本信息录入系统，系统会自动进行分类、统计和生成报表，极大地提高了工作效率。通过数据的可视化展示，管理人员可以清晰地了解宿舍资源的使用情况、设备的维修情况等，做到数据驱动决策，提升管理质量。

### 1.4 参考文献与项目

[1] 安慧.基于移动互联网的大学生思想政治教学实效性研究[D].思想政治教育,2018,5:1

[2] 杨勤.基于B/S架构的企业客户信息管理系统设计与实现[D].计算机软件及计算机应用,2017,2:4

[3] 陈广华.基于.NET技术的吉林大学珠海学院团员管理信息系统设计与实现[D].软件工程,2015,8：5

[4] 张南.居民小区户籍管理系统的设计与实现[D].计算机软件及计算机应用,2012,3:15-17

[5] 宋雨田.房地产销售控制管理软件的设计与实现[D].计算机软件及应用,2019,3：34

[6] 张静禹.依托第三方的个人健康管理服务系统研发[D].软件工程,2016,3:19

[7] [Aleksander Bai](https://pubmed.ncbi.nlm.nih.gov/?term=Bai+A&cauthor_id=30371447),[Kristin Fuglerud](https://pubmed.ncbi.nlm.nih.gov/?term=Fuglerud+K&cauthor_id=30371447), [Rannveig A Skjerve](https://pubmed.ncbi.nlm.nih.gov/?term=Skjerve+RA&cauthor_id=30371447), [Till Halbach](https://pubmed.ncbi.nlm.nih.gov/?term=Halbach+T&cauthor_id=30371447).Categorization and Comparison of Accessibility Testing Methods for Software Development[J].Stu Health Technol Infom,2018,256:821-831

[8] [Andreas Bertsatos](https://pubmed.ncbi.nlm.nih.gov/?term=Bertsatos+A&cauthor_id=30797158), [Aikaterini Christaki](https://pubmed.ncbi.nlm.nih.gov/?term=Christaki+A&cauthor_id=30797158), [Maria-Eleni Chovalopoulou](https://pubmed.ncbi.nlm.nih.gov/?term=Chovalopoulou+ME&cauthor_id=30797158).Testing the reliability of 3D-ID software in sex and ancestry estimation with a modern Greek sample[J].Forensic Sci Int,2019,297:132-137

## 2关键技术

### 2.1 后端技术

#### 2.1.1 Spring Boot IMG_256

Spring Boot 是一个基于 Spring 框架的开源框架，它简化了基于 Spring 的应用程序的开发，特别是微服务架构。Spring Boot 的核心特点是零配置和快速开发，它可以通过自动配置、内嵌服务器以及默认配置来简化开发人员的工作。在构建 Web 应用时，Spring Boot 提供了一个内嵌的 Tomcat 服务器，这使得部署变得更加简便。它有助于减少开发过程中的配置时间，并且促进了 Spring 的快速启动。

#### 2.1.2 Mybatis & Mybatis-Plus IMG_256 IMG_256

**Mybatis** 是一个优秀的持久层框架，它使得 Java 应用程序与数据库之间的交互更加高效。与传统的 JDBC 相比，Mybatis 提供了映射 SQL 的功能，可以通过 XML 文件或注解来编写 SQL 查询，简化了数据库操作的复杂性。

**Mybatis-Plus** 是 Mybatis 的增强工具，提供了更加高效、便捷的方式来进行数据库操作。它不仅提供了基础的 CRUD 操作，还增加了分页、自动生成 SQL 等功能，减少了开发者手动编写 SQL 的工作量。Mybatis-Plus 提供的自动注入和动态 SQL 使得开发过程更加灵活和简洁。

#### 2.1.3 MySQL IMG_256

MySQL 是世界上最流行的关系型数据库管理系统之一，广泛应用于 Web 应用程序中。它以稳定性、易用性以及良好的性能为特点。MySQL 作为后台数据库，存储系统中涉及学生宿舍管理的数据，如学生信息、宿舍安排等。它支持高并发操作，并且能够通过索引和优化查询提高数据访问效率。

#### 2.1.4 Lombok IMG_256

Lombok 是一个 Java 库，它通过注解简化了 Java 代码的编写，尤其是在生成 getters、setters、构造函数、toString 等常用方法时。通过使用 Lombok，开发者可以省去大量的样板代码，提高开发效率。它通过注解处理器生成代码，从而减少了手动编写重复代码的工作量。常用的注解有 @Getter、@Setter、@ToString、@EqualsAndHashCode 等。

#### 2.1.5 Commons Libraries IMG_256

Apache Commons 是一组由 Apache 软件基金会开发的开源 Java 库，包含了大量常用的工具类。以下是一些常见的 Commons 库：

* **commons-collections**：为 Java 提供了一些增强的数据结构和算法。
* **commons-lang**：提供了许多常用的字符串、日期等操作的工具类，简化了常见操作。
* **commons-beanutils**：用于操作 Java Bean，提供了属性拷贝、转换等功能。
* **commons-logging**：提供了一层对日志系统的抽象，便于不同日志框架的集成。

#### 2.1.6 JSON相关库 IMG_256

在学生宿舍管理系统中，数据传输通常采用 JSON 格式。为了高效处理 JSON 数据，常用的 JSON 处理库包括：

* **fastjson**：由阿里巴巴开发的高效 JSON 处理库，能够快速地序列化和反序列化 Java 对象。
* **json-lib**：是一个常见的 Java JSON 库，支持将 Java 对象转换为 JSON 格式，反之亦然。
* **commons-collections**：在 JSON 处理过程中，常用来处理集合对象，如 List、Map 等。

#### 2.1.7 Hutool IMG_256

Hutool 是一个开源的 Java 工具库，提供了许多常用的工具方法，涵盖了常见的操作如字符串处理、日期处理、文件操作、网络通信等。Hutool 的特点是轻量级、高效且易用。它封装了大量的常用工具方法，减少了开发过程中重复造轮子的工作。例如，StrUtil 类提供了字符串操作工具，DateUtil 类则提供了日期处理的工具。

#### 2.1.8 Spring Boot DevTools IMG_256

Spring Boot DevTools 是 Spring Boot 提供的一个开发工具包，用于提升开发过程中的效率。它主要提供了自动重启、LiveReload 和调试支持等功能。自动重启使得开发人员在修改代码后，系统能够自动重新加载应用程序，减少了手动重启的时间。LiveReload 功能则可以在前端修改时自动刷新页面，便于前后端协作开发。

#### 2.1.9 Spring Boot Test IMG_256

Spring Boot Test 是用于测试 Spring Boot 应用程序的框架。它提供了简单的测试配置和注解，能够帮助开发者快速编写和运行单元测试、集成测试。通过 @SpringBootTest 注解，可以加载整个 Spring 上下文，并模拟真实环境运行应用程序，使得测试更加接近实际使用情况。Spring Boot Test 支持自动注入依赖、事务回滚等功能，帮助开发者更方便地测试应用程序的各个模块。

### 2.2 前端技术

#### 2.2.1 Vue.js IMG_256

Vue.js 是一个渐进式 JavaScript 框架，旨在构建用户界面。与其他框架相比，Vue 具有非常灵活的结构，可以逐步集成到现有项目中。它的核心库仅关注视图层，而其他功能（如路由、状态管理）则通过插件的形式扩展。Vue 使用声明式渲染和组件化的开发方式，使得开发者可以更清晰地组织代码，提高开发效率。

Vue 的响应式系统是其最大的特点之一，能够自动追踪数据的变化并更新视图。Vue 的模板语法简洁易懂，适合初学者学习。同时，Vue 支持 JSX 和 TypeScript，可以灵活满足不同开发者的需求。由于其易用性、灵活性和高性能，Vue.js 成为现代 Web 开发中非常流行的框架之一。

#### 2.2.2 Vue Router IMG_256

Vue Router 是 Vue.js 的官方路由管理器，专为单页面应用（SPA）设计。它允许开发者在单个页面内通过 URL 控制视图的显示，从而避免了传统多页面应用中的页面刷新，提高了用户体验。Vue Router 可以定义不同的路由规则，将不同的 URL 映射到相应的组件。

Vue Router 支持嵌套路由、动态路由、导航守卫等功能。嵌套路由允许页面结构更加灵活，适应复杂的页面需求；动态路由则能够根据传递的参数来加载不同的组件。导航守卫使得开发者可以在路由切换前、后执行特定的代码，比如验证用户是否登录等操作。通过这些强大的功能，Vue Router 为构建复杂的单页面应用提供了全面的支持。

#### 2.2.3 Vuex IMG_256

Vuex 是 Vue.js 的官方状态管理库，用于管理和共享跨组件的状态。在复杂的应用中，不同的组件可能需要访问共享的状态，使用 Vuex 可以避免组件间的状态传递混乱。Vuex 将状态管理集中到一个存储中，通过四个核心概念——State、Getter、Mutation 和 Action 来处理状态。

在 Vuex 中，State 用于保存数据，Getter 用于从 State 中派生出计算后的状态，Mutation 用于同步修改 State，Action 用于处理异步操作并提交 Mutation。通过这种集中式的状态管理，Vuex 不仅简化了数据的管理，还使得状态的变化变得更加可预测和易于调试。对于大型应用，Vuex 提供了更好的可维护性和可扩展性。

#### 2.2.4 Axios IMG_256

Axios 是一个基于 Promise 的 HTTP 客户端，用于发送 HTTP 请求并接收响应。它支持浏览器和 Node.js 环境，常用于获取远程数据。Axios 提供了简洁的 API，可以轻松发起 GET、POST 等常见的 HTTP 请求，并且能够处理 JSON 数据的解析。Axios 支持请求和响应拦截器，允许在请求发出前或响应返回后执行自定义的操作。

除了基本的请求功能外，Axios 还提供了错误处理机制，可以捕获请求过程中的错误并进行相应的处理。此外，Axios 也支持请求取消、请求超时、并发请求等功能，使得开发者能够更加灵活地控制 HTTP 请求的行为。由于其简单易用和强大的功能，Axios 成为前端开发中常用的 HTTP 请求库之一。

#### 2.2.5 Webpack IMG_256

Webpack 是现代前端开发中不可或缺的工具，它是一个模块打包器。Webpack 的核心功能是将不同类型的资源（如 JavaScript、CSS、图片等）通过 loader 转换为可以在浏览器中运行的文件，然后通过插件对文件进行优化。Webpack 支持模块化开发，可以将代码分割成多个小模块，按需加载，从而提高应用的性能。

除了打包功能，Webpack 还支持热模块替换（HMR），使得开发过程中修改代码后无需刷新浏览器即可立即看到效果。Webpack 的配置灵活，可以根据项目需求进行高度定制。通过结合各种插件，开发者可以对打包过程进行全面的控制，如压缩代码、提取 CSS 文件、代码分割等操作，使得最终的输出文件更加优化和高效。

#### 2.2.6 Babel IMG_256

Babel 是一个 JavaScript 编译器，用于将现代 JavaScript 代码转换为向后兼容的版本，以便在不支持新特性的浏览器上运行。Babel 主要用于将 ES6 及以上版本的代码转译成浏览器可以理解的 ES5 代码。它通过插件的方式支持各种 JavaScript 特性，如箭头函数、类、模板字符串等，使得开发者可以使用最新的语法特性而不必担心浏览器兼容性问题。

Babel 不仅支持语法转换，还能与 Webpack 配合使用，处理 JavaScript 代码的打包和优化。通过配置 Babel，开发者可以选择不同的插件和预设来定制编译过程。Babel 的出现让前端开发者能够更快速地采用新特性，同时保持广泛的兼容性。

#### 2.2.7 Vue CLI IMG_256

Vue CLI 是 Vue.js 的官方脚手架工具，用于快速创建 Vue 项目。通过 Vue CLI，开发者可以轻松搭建项目结构，自动化配置开发环境，避免手动配置繁琐的步骤。Vue CLI 提供了一套标准化的构建配置，支持热重载、代码分割、文件压缩等功能，同时允许开发者根据项目需求定制配置。

Vue CLI 还提供了插件机制，允许开发者在项目中安装和使用各种功能插件，如 TypeScript 支持、PWA 插件、ESLint 插件等。通过 Vue CLI，开发者可以在短时间内构建出一个具有完整开发环境的 Vue 项目，同时能够灵活地扩展项目功能，提升开发效率。

#### 2.2.8 UI 组件库 IMG_256 IMG_256 IMG_256

UI 组件库是一组预定义的、可复用的用户界面组件，它们封装了常用的 UI 元素，如按钮、输入框、表单、模态框等。UI 组件库的目的是简化开发过程，提高开发效率，并确保界面的一致性。开发者可以通过引入这些组件，快速构建出美观、功能齐全的界面，而无需从头开始编写样式和功能代码。

一些流行的 Vue UI 组件库如 Element UI、Vuetify 和 Ant Design Vue 等，都提供了丰富的组件和良好的文档支持。它们不仅支持响应式设计，适配各种屏幕尺寸，还能根据不同需求进行自定义和扩展。使用 UI 组件库能够大大减少界面开发的工作量，并帮助开发者专注于业务逻辑的实现。

#### 2.2.9 CSS 预处理器 IMG_256

CSS 预处理器（如 Sass、Less）是一种增强 CSS 的工具，能够为 CSS 增加变量、函数、条件语句等编程特性，从而使得 CSS 更加动态和灵活。CSS 预处理器通过扩展 CSS 的语法，提供了一种更简洁、更结构化的方式来编写样式。它能够大大提高代码的复用性和可维护性，尤其是在大型项目中。

使用 CSS 预处理器，开发者可以在样式文件中定义变量、嵌套选择器、函数等，从而使得样式更加模块化，避免重复代码。Sass 和 Less 都有强大的功能，如循环、条件判断和 mixin 等，能够帮助开发者更高效地处理复杂的样式需求。通过编译后生成的标准 CSS 文件，可以兼容所有浏览器，提升开发效率和代码质量。

#### 2.2.10 Vue DevTools IMG_256

Vue DevTools 是一个为 Vue.js 应用开发提供的调试工具，能够帮助开发者查看和调试 Vue 应用的内部状态。它提供了一个可视化界面，显示了当前页面中所有 Vue 组件的状态、属性、事件等信息。开发者可以通过 Vue DevTools 轻松查看组件树、检查 Vuex 状态、监控路由变化等，从而加速开发和调试过程。

Vue DevTools 还支持时间旅行调试功能，允许开发者回溯到应用的某个历史状态，查看状态如何随着时间变化。它还可以帮助开发者优化应用性能，通过查看组件的生命周期、响应式数据的变化等信息，发现并修复性能瓶颈。对于大型应用的调试和优化，Vue DevTools 是一个必不可少的工具。

## 3数据库设计

### 3.1 数据库设计概述

#### 3.1.1 数据库基本信息

数据库名称：dormitorymanage

数据库类型：MySQL 8.0.37

字符集：utf8mb4

排序规则：utf8mb4\_general\_ci

数据库引擎：InnoDB

#### 3.1.2 数据库设计目标

**（1）高效性**：支持快速的查询和数据更新，满足宿舍管理系统的高并发需求。

**（2）可靠性**：通过完整性约束和事务管理确保数据的准确性和一致性。

**（3）可扩展性**：支持未来功能扩展，例如增加统计报表、宿舍评价功能等模块。

**（4）安全性**：通过用户权限控制和数据加密机制，确保数据访问的安全性。

#### 3.1.3 数据库设计原则

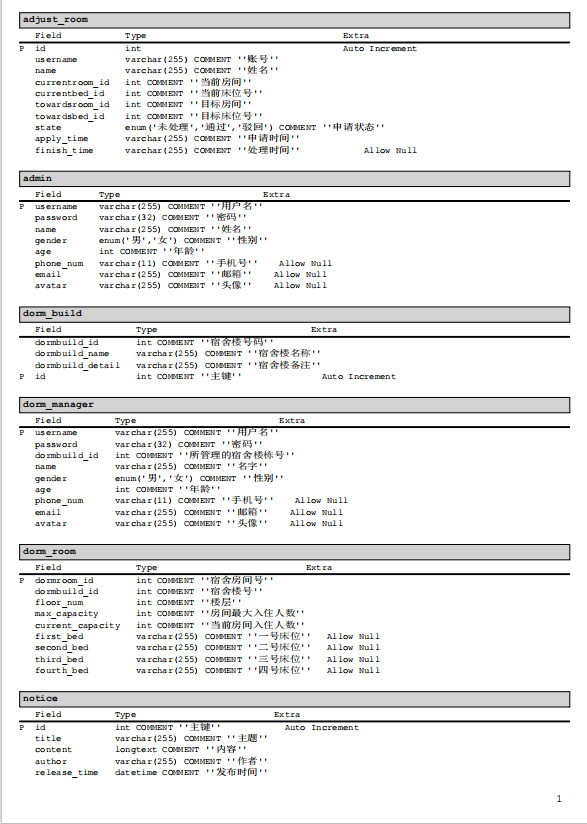
**（1）规范化设计**：严格遵守数据库设计的规范化标准，避免数据冗余，提高存储效率。

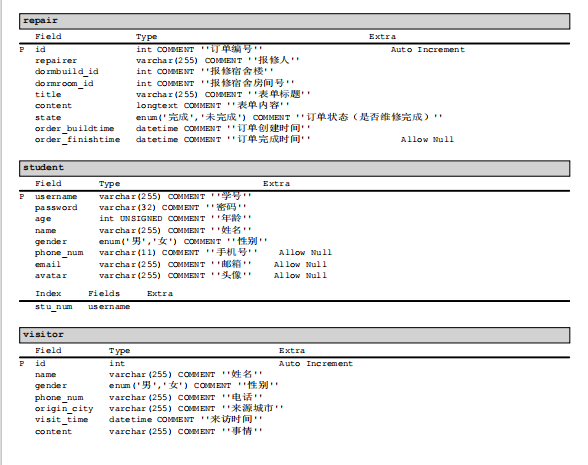
**（2）业务需求导向**：以宿舍管理系统的实际需求为中心，设计合理的表结构和字段关系。

**（3）性能优化**：通过索引、分区等手段提升查询性能，减少复杂查询的开销。

**（4）易维护性**：采用统一的命名规范和清晰的表结构，便于后期维护与开发。

#### 3.1.4 数据库表报告





### 3.2 数据库表结构设计

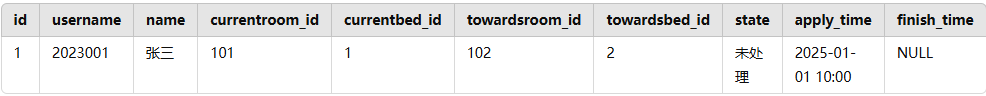
#### 3.2.1 adjust\_room表

##### 表说明

存储学生调宿的申请记录。



##### 数据示例



##### SQL 示例

CREATE TABLE `adjust\_room` (

`id` INT AUTO\_INCREMENT PRIMARY KEY,

`username` VARCHAR(255) NOT NULL,

`name` VARCHAR(255) NOT NULL,

`currentroom\_id` INT NOT NULL,

`currentbed\_id` INT NOT NULL,

`towardsroom\_id` INT NOT NULL,

`towardsbed\_id` INT NOT NULL,

`state` ENUM('未处理', '通过', '驳回') DEFAULT '未处理',

`apply\_time` DATETIME NOT NULL,

`finish\_time` DATETIME,

`reason` TEXT,

`reviewer` VARCHAR(255),

FOREIGN KEY (`currentroom\_id`) REFERENCES `dorm\_room`(`dormroom\_id`),

FOREIGN KEY (`towardsroom\_id`) REFERENCES `dorm\_room`(`dormroom\_id`),

FOREIGN KEY (`reviewer`) REFERENCES `admin`(`username`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

#### 3.2.2 admin表

##### 表说明

存储系统管理员信息，包括宿舍管理员。



##### 数据示例



##### SQL 示例

CREATE TABLE `admin` (

`username` VARCHAR(255) PRIMARY KEY,

`password` VARCHAR(255) NOT NULL,

`name` VARCHAR(255) NOT NULL,

`phone\_num` VARCHAR(20),

`email` VARCHAR(255)

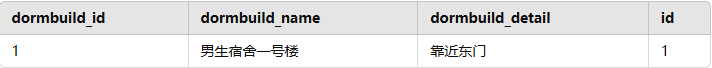
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

#### 3.2.3 dorm\_build 表

##### 表说明



##### 数据示例



##### SQL 示例

CREATE TABLE `dorm\_build` (

`dormbuild\_id` INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(255) NOT NULL,

`address` VARCHAR(255) NOT NULL,

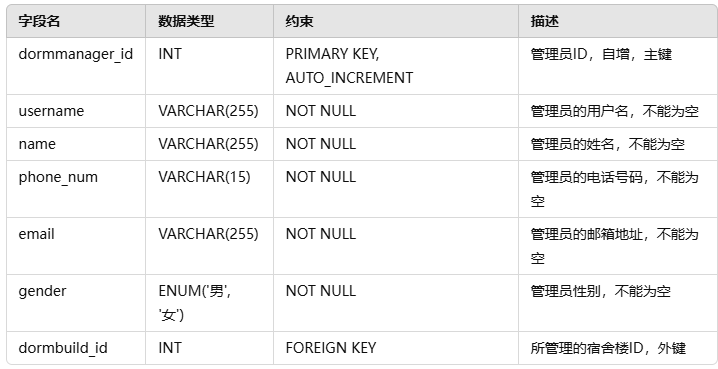
`total\_rooms` INT NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

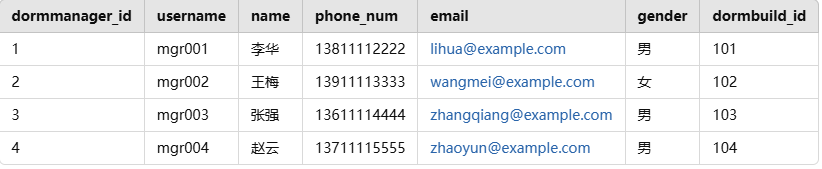
#### 3.2.4 dorm\_manager表

##### 表说明

存储宿舍管理员的基本信息。



##### 数据示例



##### SQL 示例

CREATE TABLE `dorm\_manager` (

`id` INT AUTO\_INCREMENT PRIMARY KEY,

`username` VARCHAR(255) NOT NULL UNIQUE,

`password` VARCHAR(255) NOT NULL,

`name` VARCHAR(255) NOT NULL,

`phone\_num` VARCHAR(20) NOT NULL,

`dormbuild\_id` INT NOT NULL,

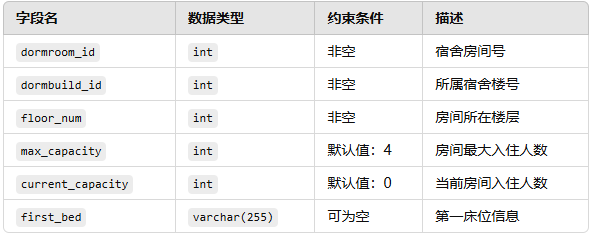
FOREIGN KEY (`dormbuild\_id`) REFERENCES `dorm\_build`(`dormbuild\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

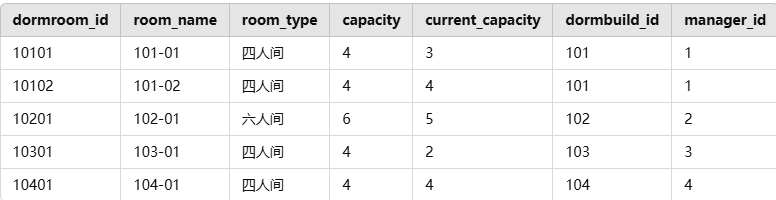
#### 3.2.5 dorm\_room

##### 表说明

存储宿舍房间的基本信息。



##### 数据示例



##### SQL 示例

CREATE TABLE `dorm\_room` (

`dormroom\_id` INT AUTO\_INCREMENT PRIMARY KEY,

`room\_number` VARCHAR(20) NOT NULL UNIQUE,

`capacity` INT NOT NULL,

`current\_occupants` INT DEFAULT 0,

`dormbuild\_id` INT NOT NULL,

FOREIGN KEY (`dormbuild\_id`) REFERENCES `dorm\_build`(`dormbuild\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

#### 3.2.6 notice

##### 表说明

存储通知信息。



##### 数据示例



##### SQL 示例

CREATE TABLE `notice` (

`id` INT AUTO\_INCREMENT PRIMARY KEY,

`title` VARCHAR(255) NOT NULL,

`content` TEXT NOT NULL,

`publisher` VARCHAR(255) NOT NULL,

`publish\_time` DATETIME NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

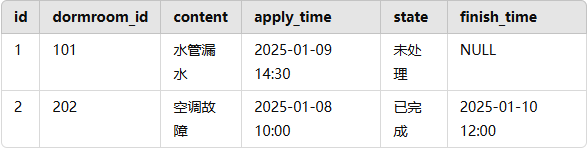
#### 3.2.7 repair

##### 表说明

存储保修信息



##### 数据示例



##### SQL 示例

CREATE TABLE `repair` (

`id` INT AUTO\_INCREMENT PRIMARY KEY,

`dormroom\_id` INT NOT NULL,

`content` TEXT NOT NULL,

`apply\_time` DATETIME NOT NULL,

`state` ENUM('未处理', '处理中', '已完成') DEFAULT '未处理',

`finish\_time` DATETIME,

FOREIGN KEY (`dormroom\_id`) REFERENCES `dorm\_room`(`dormroom\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

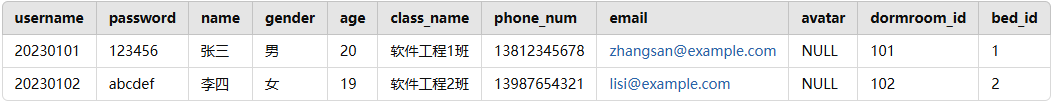
#### 3.2.8 student

##### 表说明

存储学生实体



##### 数据示例



##### SQL 示例

CREATE TABLE `student` (

`username` VARCHAR(255) PRIMARY KEY,

`password` VARCHAR(255) NOT NULL,

`name` VARCHAR(255) NOT NULL,

`gender` ENUM('男', '女') NOT NULL,

`age` INT NOT NULL,

`class\_name` VARCHAR(255) NOT NULL,

`phone\_num` VARCHAR(20),

`email` VARCHAR(255),

`avatar` BLOB,

`dormroom\_id` INT NOT NULL,

`bed\_id` INT NOT NULL,

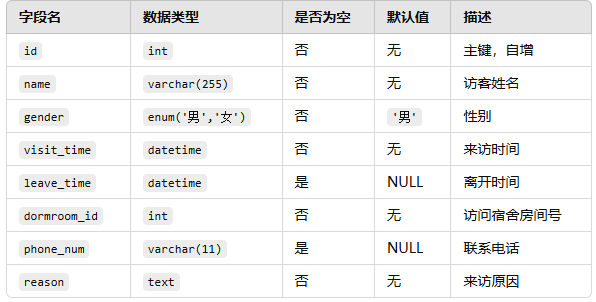
FOREIGN KEY (`dormroom\_id`) REFERENCES `dorm\_room`(`dormroom\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

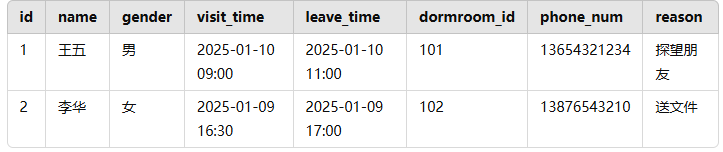
#### 3.2.9 visitor

##### 表说明

存储访客信息



##### 数据示例



##### SQL 示例

CREATE TABLE `visitor` (

`id` INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(255) NOT NULL,

`gender` ENUM('男', '女') NOT NULL,

`visit\_time` DATETIME NOT NULL,

`leave\_time` DATETIME,

`dormroom\_id` INT NOT NULL,

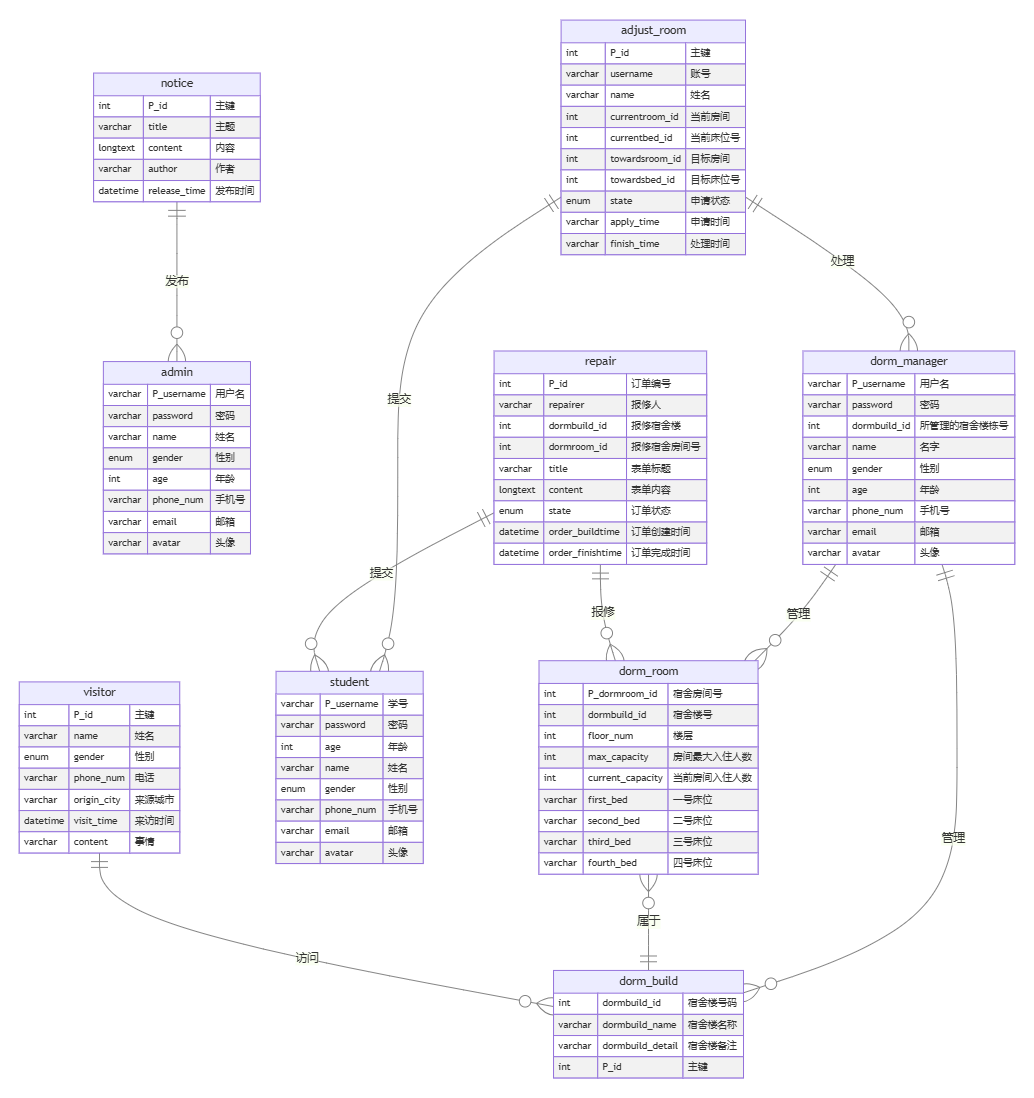
`phone\_num` VARCHAR(20),

`reason` TEXT,

FOREIGN KEY (`dormroom\_id`) REFERENCES `dorm\_room`(`dormroom\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

### 3.3 数据表关系示意图



**图片使用 mermaid 绘制，代码如下 ：**

erDiagram

"adjust\_room" {

int P\_id "主键"

varchar username "账号"

varchar name "姓名"

int currentroom\_id "当前房间"

int currentbed\_id "当前床位号"

int towardsroom\_id "目标房间"

int towardsbed\_id "目标床位号"

enum state "申请状态"

varchar apply\_time "申请时间"

varchar finish\_time "处理时间"

}

"admin" {

varchar P\_username "用户名"

varchar password "密码"

varchar name "姓名"

enum gender "性别"

int age "年龄"

varchar phone\_num "手机号"

varchar email "邮箱"

varchar avatar "头像"

}

"dorm\_build" {

int dormbuild\_id "宿舍楼号码"

varchar dormbuild\_name "宿舍楼名称"

varchar dormbuild\_detail "宿舍楼备注"

int P\_id "主键"

}

"dorm\_manager" {

varchar P\_username "用户名"

varchar password "密码"

int dormbuild\_id "所管理的宿舍楼栋号"

varchar name "名字"

enum gender "性别"

int age "年龄"

varchar phone\_num "手机号"

varchar email "邮箱"

varchar avatar "头像"

}

"dorm\_room" {

int P\_dormroom\_id "宿舍房间号"

int dormbuild\_id "宿舍楼号"

int floor\_num "楼层"

int max\_capacity "房间最大入住人数"

int current\_capacity "当前房间入住人数"

varchar first\_bed "一号床位"

varchar second\_bed "二号床位"

varchar third\_bed "三号床位"

varchar fourth\_bed "四号床位"

}

"notice" {

int P\_id "主键"

varchar title "主题"

longtext content "内容"

varchar author "作者"

datetime release\_time "发布时间"

}

"repair" {

int P\_id "订单编号"

varchar repairer "报修人"

int dormbuild\_id "报修宿舍楼"

int dormroom\_id "报修宿舍房间号"

varchar title "表单标题"

longtext content "表单内容"

enum state "订单状态"

datetime order\_buildtime "订单创建时间"

datetime order\_finishtime "订单完成时间"

}

"student" {

varchar P\_username "学号"

varchar password "密码"

int age "年龄"

varchar name "姓名"

enum gender "性别"

varchar phone\_num "手机号"

varchar email "邮箱"

varchar avatar "头像"

}

"visitor" {

int P\_id "主键"

varchar name "姓名"

enum gender "性别"

varchar phone\_num "电话"

varchar origin\_city "来源城市"

datetime visit\_time "来访时间"

varchar content "事情"

}

"adjust\_room" ||--o{ "student" : "提交"

"adjust\_room" ||--o{ "dorm\_manager" : "处理"

"dorm\_manager" ||--o{ "dorm\_build" : "管理"

"dorm\_room" }o--|| "dorm\_build" : "属于"

"dorm\_manager" ||--o{ "dorm\_room" : "管理"

"repair" ||--o{ "dorm\_room" : "报修"

"repair" ||--o{ "student" : "提交"

"notice" ||--o{ "admin" : "发布"

"visitor" ||--o{ "dorm\_build" : "访问"

## 4系统实现

### 4.1 系统需求分析

#### 4.1.1 功能需求

本系统开发的主要目的是采用信息化和网络化的方式，用于对宿舍进行管理，方便管理员对宿舍进行管理以及对同学们宿舍生活需求提供相应的帮助和记录，达到及时、有效的掌握学生信息以及处理寝室问题的目的。

#### 4.1.2 功能描述

表4-1查看学生信息

|  |  |
| --- | --- |
| 名称 | 查看学生信息 |
| 操作角色 | 管理员 |
| 功能描述 | 对学生信息进行遍历与展示 |
| 操作 | 1.管理员进入学生信息界面  2.系统展示该系统的全部学生用户信息 |
| 输出 | 学生信息列表 |

表4-2添加学生信息

|  |  |
| --- | --- |
| 名称 | 添加学生信息 |
| 操作角色 | 管理员 |
| 功能描述 | 新增一条学生用户信息 |
| 操作 | 1.点击学生页面的新增按钮  2.在弹出的模态框中输入正确的学生信息  3.点击确定，新增成功 |
| 输出 | 学生信息列表 |

表4-3删除学生信息

|  |  |
| --- | --- |
| 名称 | 删除学生信息 |
| 操作角色 | 管理员 |
| 功能描述 | 删除需要删除的学生信息 |
| 操作 | 1.找到需要删除的学生信息，点击删除按钮  2.跳出提示信息，点击确认，删除成功 |
| 输出 | 学生信息列表 |

表4-4编辑学生信息

|  |  |
| --- | --- |
| 名称 | 编辑学生信息 |
| 操作角色 | 管理员 |
| 功能描述 | 修改需要编辑的学生信息 |
| 操作 | 1.找到需要修改的学生信息，点击编辑按钮  2.跳出学生信息模态框，在模态框中修改需要改的信息  3.点击确认，编辑成功 |
| 输出 | 学生信息列表 |

表4-5查询学生信息

|  |  |
| --- | --- |
| 名称 | 查询学生信息 |
| 操作角色 | 管理员 |
| 功能描述 | 对需要的学生信息进行模糊查询 |
| 操作 | 1.点击列表上方的文本框，输入需要查找的学生的有关信息  2.点击搜索，列表展示查找到的相关的学生信息 |
| 输出 | 查询到的学生信息列表 |

表4-6查看报修信息

|  |  |
| --- | --- |
| 名称 | 查看报修信息 |
| 操作角色 | 管理员 |
| 功能描述 | 对报修信息进行遍历与展示 |
| 操作 | 1.管理员进入报修信息界面  2.系统展示该系统的全部报修信息 |
| 输出 | 报修信息列表 |

表4-7添加报修信息

|  |  |
| --- | --- |
| 名称 | 添加报修信息 |
| 操作角色 | 管理员 |
| 功能描述 | 新增一条报修信息 |
| 操作 | 1.点击报修页面的新增按钮  2.在弹出的模态框中输入正确的报修信息  3.点击确定，新增成功 |
| 输出 | 报修信息列表 |

表4-8删除报修信息

|  |  |
| --- | --- |
| 名称 | 删除报修信息 |
| 操作角色 | 管理员 |
| 功能描述 | 删除需要删除的报修信息 |
| 操作 | 1.找到需要删除的报修信息，点击删除按钮  2.跳出提示信息，点击确认，删除成功 |
| 输出 | 报修信息列表 |

表4-9编辑报修信息

|  |  |
| --- | --- |
| 名称 | 编辑报修信息 |
| 操作角色 | 管理员 |
| 功能描述 | 修改需要编辑的报修信息 |
| 操作 | 1.找到需要修改的报修信息，点击编辑按钮  2.跳出报修信息模态框，在模态框中修改需要改的信息  3.点击确认，编辑成功 |
| 输出 | 报修信息列表 |

表4-10查询报修信息

|  |  |
| --- | --- |
| 名称 | 查询报修信息 |
| 操作角色 | 管理员 |
| 功能描述 | 对需要的报修信息进行模糊查询 |
| 操作 | 1.点击列表上方的文本框，输入需要查找的报修的有关信息  2.点击搜索，列表展示查找到的相关的报修信息 |
| 输出 | 查询到的报修信息列表 |

表4-11审核报修信息

|  |  |
| --- | --- |
| 名称 | 审核报修信息 |
| 操作角色 | 管理员 |
| 功能描述 | 对学生上报的报修信息进行审核 |
| 操作 | 1.找到需要审核的报修信息，点击编辑按钮  2.在弹出的报修信息模态框中选择订单状态，完成或未完成  3.点击确定，修改成功 |
| 输出 | 报修信息列表 |

表4-12申请报修

|  |  |
| --- | --- |
| 名称 | 申请报修 |
| 操作角色 | 学生 |
| 功能描述 | 学生上报需要报修的信息 |
| 操作 | 1.点击报修申请信息表格上方的新增按钮  2.弹出新增模态框，输入需要报修的信息内容  3.点击确认，报修成功 |
| 输出 | 申请报修信息列表 |

表4-13查询报修申请

|  |  |
| --- | --- |
| 名称 | 查看报修申请 |
| 操作角色 | 学生 |
| 功能描述 | 学生查看自己报修过的信息内容 |
| 操作 | 1.学生进入申请报修信息界面  2.系统展示该学生的全部申请报修信息 |
| 输出 | 申请报修信息列表 |

表4-14查询申请报修信息

|  |  |
| --- | --- |
| 名称 | 查询申请报修信息 |
| 操作角色 | 学生 |
| 功能描述 | 对需要的申请报修信息进行模糊查询 |
| 操作 | 1.点击列表上方的文本框，输入需要查找的申请报修的有关信息  2.点击搜索，列表展示查找到的相关的申请报修信息 |
| 输出 | 查询到的申请报修信息列表 |

表4-15查询单条报修信息

|  |  |
| --- | --- |
| 名称 | 查询单条报修信息 |
| 操作角色 | 学生/管理员 |
| 功能描述 | 对需要细看的报修信息进行详细查看 |
| 操作 | 1.点击需要查看的报修信息详情按钮  2.弹出详情模态框，展示详细报修信息 |
| 输出 | 详细报修信息 |

表4-16登录

|  |  |
| --- | --- |
| 名称 | 用户登录网站 |
| 操作角色 | 系统管理员、学生 |
| 功能描述 | 用户可登录自己的宿管系统账号 |
| 操作 | 1. 用户在首页登录框输入正确的账号和密码 |
| 输出 | 登录成功 |

表4-17访客管理

|  |  |
| --- | --- |
| 名称 | 查看访客管理模块 |
| 操作角色 | 系统管理员 |
| 功能描述 | 用户可以登记修改访客的信息 |
| 操作 | 1. 用户进入访客管理模块 2. 对访客信息进行增删改查操作 |
| 输出 | 列表显示访客信息 |

表4-18个人信息管理

|  |  |
| --- | --- |
| 名称 | 查看个人信息模块 |
| 操作角色 | 用户 |
| 功能描述 | 用户可查看修改个人信息 |
| 操作 | 1. 用户进入个人信息模块 2. 用户点击编辑按钮 3. 弹出编辑模态框 4. 按要求输入信息点击确定修改成功 |
| 输出 | 显示个人信息 |

表4-19进入我的宿舍页面

|  |  |
| --- | --- |
| 名称 | 进入我的宿舍页面 |
| 操作角色 | 用户 |
| 功能描述 | 进入我的宿舍 |
| 操作 | 1.用户登陆成功后。  2.点击我的宿舍就能进入此页面。 |
| 输出 | 我的宿舍 |

表4-20我的宿舍展示信息

|  |  |
| --- | --- |
| 名称 | 我的宿舍展示信息 |
| 操作角色 | 用户 |
| 功能描述 | 展示房间信息和床位信息 |
| 操作 | 1. 点击我的宿舍。 |
| 输出 | 我的宿舍信息 |

表4-21进入楼宇信息页面

|  |  |
| --- | --- |
| 名称 | 进入楼宇信息页面 |
| 操作角色 | 管理员 |
| 功能描述 | 进入楼宇信息页面 |
| 操作 | 1.管理员账号登录。  2.在侧边栏点击信息管理  3.点击楼宇信息 |
| 输出 | 楼宇信息页面 |

表4-22添加楼宇

|  |  |
| --- | --- |
| 名称 | 添加楼宇 |
| 操作角色 | 管理员 |
| 功能描述 | 管理员可以添加楼宇数据 |
| 操作 | 1. 点击右上角的加号 2. 输入正确的编号，名称，备注 3. 点击确定 |
| 输出 | 添加后的楼宇列表 |

表4-23删除楼宇数据

|  |  |
| --- | --- |
| 名称 | 删除楼宇数据 |
| 操作角色 | 管理员 |
| 功能描述 | 管理员可以删除不需要的楼宇数据 |
| 操作 | 1. 点击列表中的删除图标。 2. 点击确定删除 3. 删除成功。 |
| 输出 | 删除后的楼宇列表 |

表4-24修改楼宇信息

|  |  |
| --- | --- |
| 名称 | 根据需要修改楼宇信息 |
| 操作角色 | 管理员 |
| 功能描述 | 管理员可以根据需要修改楼宇信息 |
| 操作 | 1. 用户点击操作栏的修改图标。 2. 填写新的名称和备注。 3. 点击确定 4. 修改成功 |
| 输出 | 修改后的楼宇信息列表 |

表4-25根据编号搜索楼宇信息

|  |  |
| --- | --- |
| 名称 | 根据编号搜索楼宇信息 |
| 操作角色 | 管理员 |
| 功能描述 | 根据编号搜索楼宇信息 |
| 操作 | 1. 输入编号。 2. 点击搜索按钮。 3. 显示搜索后的楼宇信息 |
| 输出 | 搜索后的楼宇信息 |

表4-26根据备注筛选楼宇信息

|  |  |
| --- | --- |
| 名称 | 根据备注筛选楼宇信息 |
| 操作角色 | 管理员 |
| 功能描述 | 根据备注筛选楼宇信息 |
| 操作 | 1. 点击备注。 2. 勾选男宿舍或者女宿舍。 |
| 输出 | 显示筛选后的楼宇信息 |

表4-27进入首页

|  |  |
| --- | --- |
| 名称 | 进入首页 |
| 操作角色 | 管理员 |
| 功能描述 | 进入管理员首页 |
| 操作 | 1.管理员登录成功进入首页。 |
| 输出 | 学生统计、住宿统计、报修统计、空宿舍统计情况以及宿舍通告、宿舍学生人数分布图、天气和日历 |

表4-28查看房间信息

|  |  |
| --- | --- |
| 名称 | 查看房间信息 |
| 操作角色 | 管理员 |
| 功能描述 | 管理员进入房间信息页面查看房间信息 |
| 操作 | 1. 管理员点击左侧信息管理下的房间信息 |
| 输出 | 房间信息 |

表4-29添加房间信息

|  |  |
| --- | --- |
| 名称 | 添加房间信息 |
| 操作角色 | 管理员 |
| 功能描述 | 添加一个房间 |
| 操作 | 1. 管理员点击右上角的新增按钮 2. 输入房间的楼栋号、楼层号、房间号、最多可住人数和已住人数 3. 点击确定，成功添加一个房间 |
| 输出 | 新增成功，返回房间信息；新增失败，提示错误 |

表4-30删除房间信息

|  |  |
| --- | --- |
| 名称 | 删除房间信息 |
| 操作角色 | 管理员 |
| 功能描述 | 对一个房间信息进行删除 |
| 操作 | 1. 管理员进入房间信息界面。 2. 点击每行房间信息右侧的删除按钮进行删除 |
| 输出 | 删除成功，返回房间信息；删除失败，提示错误 |

表4-31修改房间信息

|  |  |
| --- | --- |
| 名称 | 修改房间信息 |
| 操作角色 | 管理员 |
| 功能描述 | 管理员将一个房间的信息进行修改 |
| 操作 | 1. 管理员进入房间信息界面。 2. 点击右侧修改按钮。 3. 依次输入需要修改的信息。 4. 点击确定按钮 |
| 输出 | 修改成功，返回房间信息界面；修改失败，提示错误 |

表4-32模糊查询房间信息

|  |  |
| --- | --- |
| 名称 | 模糊查询房间信息 |
| 操作角色 | 管理员 |
| 功能描述 | 管理员查询一个房间的信息 |
| 操作 | 1. 管理员进入房间信息界面。 2. 在左上角搜索框内输入想要查找的任何信息。 3. 点击搜索按钮。 |
| 输出 | 搜索查询成功，返回查找的房间信息界面；查找失败，返回空信息 |

表4-33分配学生宿舍床位

|  |  |
| --- | --- |
| 名称 | 分配学生宿舍床位 |
| 操作角色 | 管理员 |
| 功能描述 | 管理员对床位进行学生分配 |
| 操作 | 1.管理员进入房间信息界面，点击房间的详情按钮。  2.选择一个床位。  3.点击操作，若该床位无人，则填入学生的学号；若有人需要重新分配，则点击删除按钮删除原来的学生再填入学号。  4.点击确定 |
| 输出 | 分配成功，返回房间信息界面；删除失败，提示错误 |

表4-34查看通知信息

|  |  |
| --- | --- |
| 名称 | 查看信息 |
| 操作角色 | 管理员 |
| 功能描述 | 查看全部通知信息 |
| 操作 | 1.管理员进入通知信息界面。  2.查看所有的通知信息。 |
| 输出 | 通知信息列表 |

表4-35新增通知信息

|  |  |
| --- | --- |
| 名称 | 新增通知信息 |
| 操作角色 | 管理员 |
| 功能描述 | 增加一则通知 |
| 操作 | 1.管理员进入通知信息界面。  2.点击右上角新增按钮，输入对应的信息  3.点击确定 |
| 输出 | 新增成功，返回通知信息页面；新增失败，提示错误 |

表4-36删除通知信息

|  |  |
| --- | --- |
| 名称 | 删除通知信息 |
| 操作角色 | 管理员 |
| 功能描述 | 删除一则通知 |
| 操作 | 1.管理员进入通知界面。  2.点击右侧删除按钮进行操作 |
| 输出 | 删除成功，返回通知页面；删除失败，提示错误 |

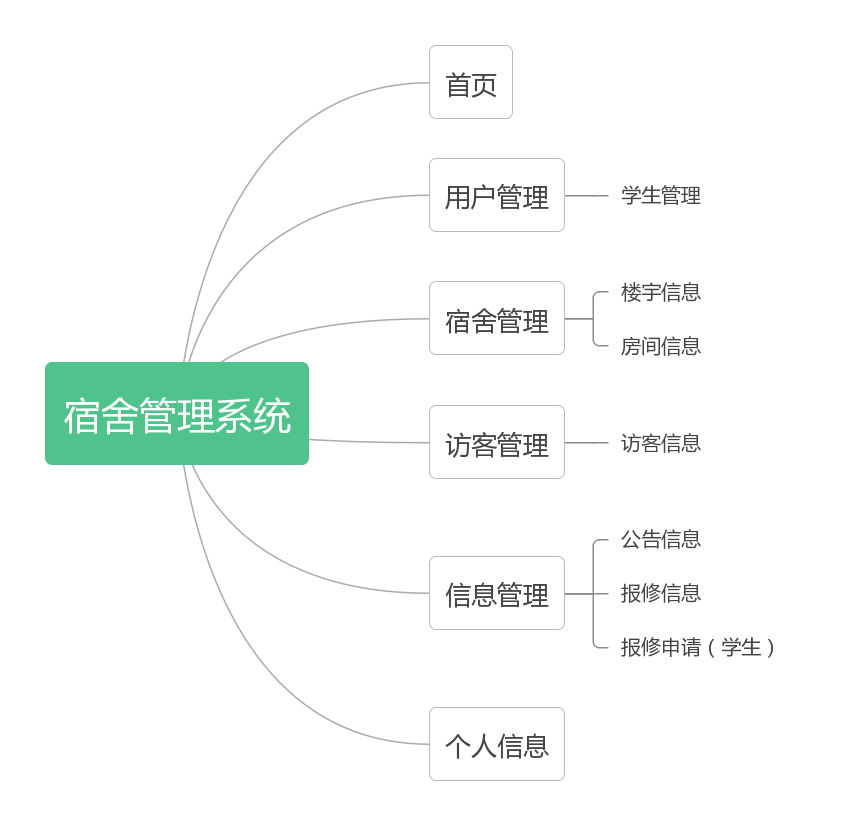
表4-37修改通知信息

|  |  |
| --- | --- |
| 名称 | 修改通知信息 |
| 操作角色 | 管理员 |
| 功能描述 | 修改通知信息 |
| 操作 | 1.点击通知右侧编辑按钮  2.对想要修改的信息进行操作  3.点击确定修改完毕 |
| 输出 | 修改成功，返回当前界面；修改失败，提示错误 |

表4-38模糊查找通知信息

|  |  |
| --- | --- |
| 名称 | 查找通知信息 |
| 操作角色 | 管理员 |
| 功能描述 |  |
| 操作 | 1.点击管理员头像  2.点击修改密码按钮  3.输入旧密码和新密码  4.点击确认按钮 |
| 输出 | 修改成功，返回医生信息界面；修改失败，提示错误 |

### 4.2 系统总体设计



#### 4.2.1学生信息模块

管理员由首页进入学生信息模块，页面展示所有学生信息列表，包括学生姓名，性别，年龄，手机号，邮箱等信息。管理员可以对学生信息列表进行关键字模糊查询操作，也可以对需要的学生信息进行修改和删除操作，也可以添加新的学生信息。当在模态框中添加修改学生信息时，格式或信息错误会提示错误信息，正确则会提示操作成功。

#### 4.2.2报修信息模块

管理员由首页进入报修信息模块，页面展示所有报修信息列表，包括报修标题，宿舍号，房间号，申请人，订单状态，订单创建时间，订单完成时间等信息。管理员可以对报修信息列表进行关键字模糊查询操作，也可以对需要的报修信息进行修改和删除操作，也可以添加新的报修信息。当在模态框中添加修改报修信息时，格式或信息错误会提示错误信息，正确则会提示操作成功。另外，管理员可以审核由学生提交的报修信息，改变其审核状态。管理员还可以单独查看某条报修信息的详情。

#### 4.2.3报修申请模块

学生由首页进入申请报修模块，页面展示该学生所有申请报修信息列表，包括报修标题，宿舍号，房间号，申请人，订单状态，订单创建时间，订单完成时间等信息。学生可以对报修申请信息列表进行关键字模糊查询操作，也可以申请新的报修信息。学生还可以单独查看某条申请报修信息的详情。

#### 4.2.4登录

登录模块主要负责用户登录该宿舍管理系统，输入正确账号和密码即可登录，分系统管理员、学生两种权限，不同的权限进入偶看到的菜单不一样。

#### 4.2.5访客管理模块

用户进入访客管理模块，列表显示访客信息，用户可对访客信息进行增删改查操作。

#### 4.2.6个人信息模块

用户进入个人信息模块后可查看个人信息，点击编辑按钮，在模态输入相关信息后点击确定完成个人信息编辑。

#### 4.2.7进入我的宿舍页面

用户登录后点击我的宿舍进入我的宿舍详情页面，自动展示楼宇信息，床位信息

#### 4.2.8自己的床位高亮显示

突出显示用户的床位。

#### 4.2.9管理员端-楼宇信息

管理员登录后点击信息管理，再点击楼宇信息即可看到所有的楼宇信息。

#### 4.2.10添加楼宇信息

用户点击操作栏的添加按钮图标，输入正确的楼宇信息点击确定，即可看到添加后的楼宇信息。

**4.2.11删除楼宇信息**

用户点击操作栏的删除按钮图标，点击确定，即可看到删除后的楼宇信息。

#### 4.2.12修改楼宇信息

用户点击操作栏的修改按钮图标，输入需要修改的信息点击确定，即可看到修改后的楼宇信息。

#### 4.2.13通过编号搜索楼宇信息

用户在搜索输入框输入需要搜索的编号，点击搜索图标即可显示该编号下的楼宇信息。

#### 4.2.14通过备注筛选楼宇信息

用户在点击备注选择男宿舍或者女宿舍，点击筛选，即可看到筛选的楼宇信息。

#### 4.2.15页数跳转

用户在页数上需要跳转的页码点击回车即可跳转到该页。

#### 4.2.16管理员对房间信息管理模块

点击左侧导航栏，管理员进入房间信息管理页面，页面展示所有房间的信息，包括房间号、楼栋号、楼层、可住人数和已住人数。管理员可以输入关键字对房间的信息进行模糊查询。管理员添加房间，弹出模态框，输入房间信息即可添加。管理员可以对某一房间点击修改按钮，在模态框内修改房间信息。管理员可以点击删除按钮，弹窗确认删除，完成删除房间操作。

#### 4.2.17管理员对公告信息管理模块

点击左侧导航栏，管理员进入公告信息管理页面，页面展示所有公告的信息，包括通知标题、内容、作者和发布时间。管理员可以输入关键字对公告的信息进行模糊查询。管理员添加公告，弹出模态框，输入公告信息即可添加。管理员可以对某一公告点击修改按钮，在模态框内修改公告信息。管理员可以点击删除按钮，弹窗确认删除，完成删除公告操作。

#### 4.2.18管理员首页

管理员登录成功之后进入首页，首页显示学生统计、住宿统计、保修统计和空宿舍统计。页面还展示当前公告信息、宿舍学生人数分布图、天气和日历。

### 4.3 系统详细设计

#### 4.3.1 学生管理模块

##### 模块描述

管理员由首页进入学生信息模块，页面展示所有学生信息列表，包括学生姓名，性别，年龄，手机号，邮箱等信息。

##### 功能

管理员可以对学生信息列表进行关键字模糊查询操作，也可以对需要的学生信息进行修改和删除操作，也可以添加新的学生信息。当在模态框中添加修改学生信息时，格式或信息错误会提示错误信息，正确则会提示操作成功。

##### 输入项

1. **查询学生信息功能**

管理员点击列表上方的文本框，输入需要查找的学生的有关信息，点击搜索，列表展示查找到的相关的学生信息。

1. **添加学生信息功能**

管理员点击学生页面的新增按钮，在弹出的模态框中输入正确的学生信息，

点击确定，新增成功。

1. **删除学生信息功能**

管理员找到需要删除的学生信息，点击删除按钮，跳出提示信息，点击确认，删除成功。

1. **修改学生信息功能**

管理员找到需要修改的学生信息，点击编辑按钮，跳出学生信息模态框，在模态框中修改需要改的信息，点击确认，编辑成功。

##### 输出项

学生信息列表

#### 4.3.2 报修信息模块

##### 模块描述

管理员由首页进入报修信息模块，页面展示所有报修信息列表，包括报修标题，宿舍号，房间号，申请人，订单状态，订单创建时间，订单完成时间等信息。

##### 功能

管理员可以对报修信息列表进行关键字模糊查询操作，也可以对需要的报修信息进行修改和删除操作，也可以添加新的报修信息。当在模态框中添加修改报修信息时，格式或信息错误会提示错误信息，正确则会提示操作成功。另外，管理员可以审核由学生提交的报修信息，改变其审核状态。管理员还可以单独查看某条报修信息的详情。

##### 输入项

**(1) 查询报修信息功能**

管理员点击列表上方的文本框，输入需要查找的报修的有关信息，点击搜索，列表展示查找到的相关的报修信息。

**(2) 添加报修信息功能**

管理员点击报修页面的新增按钮，在弹出的模态框中输入正确的报修信息，

点击确定，新增成功。

**(3) 删除报修信息功能**

管理员找到需要删除的报修信息，点击删除按钮，跳出提示信息，点击确认，删除成功。

**(4) 修改报修信息功能**

管理员找到需要修改的报修信息，点击编辑按钮，跳出报修信息模态框，在模态框中修改需要改的信息，点击确认，编辑成功。

**(5) 审核报修信息功能**

管理员找到需要审核的报修信息，点击编辑按钮，跳出报修信息模态框，在模态框中修改其审核状态，点击确认，审核成功。

##### 输出项

报修信息列表

#### 4.3.3 申请报修模块

##### 模块描述

学生由首页进入申请报修模块，页面展示该学生所有申请报修信息列表，包括报修标题，宿舍号，房间号，申请人，订单状态，订单创建时间，订单完成时间等信息。

##### 功能

学生可以对报修申请信息列表进行关键字模糊查询操作，也可以申请新的报修信息。学生还可以单独查看某条申请报修信息的详情。

##### 输入项

**(1) 查询申请报修信息功能**

学生点击列表上方的文本框，输入需要查找的申请报修的有关信息，点击搜索，列表展示查找到的相关的申请报修信息。

**(2) 添加报修信息功能**

学生点击报修页面的新增按钮，在弹出的模态框中输入正确的申请报修信息，点击确定，新增成功。

##### 输出项

申请报修信息列表

#### 4.3.4 宿舍管理模块

##### 模块描述

管理员可以在宿舍管理模块中查看所有宿舍楼的信息，包括宿舍楼名称、楼层、房间号、宿舍类型、宿舍容量等信息。管理员可以对宿舍信息进行添加、删除、修改等操作。

##### 功能

1. **查询功能**：管理员可以通过输入宿舍楼名称、房间号等信息来查询宿舍信息。
2. **新增宿舍信息**：管理员可以新增宿舍信息。
3. **删除宿舍信息**：管理员可以删除不再使用或已废弃的宿舍信息。
4. **修改宿舍信息**：管理员可以修改已有宿舍楼的信息，例如更改房间号、容量等。

##### 输入项

1. **查询宿舍信息功能**：管理员点击查询框输入信息进行查询。
2. **添加宿舍信息功能**：管理员在模态框中填写新增宿舍信息。
3. **删除宿舍信息功能**：管理员在宿舍列表中点击删除按钮，弹出确认框后删除信息。
4. **修改宿舍信息功能**：管理员在宿舍信息列表中点击编辑按钮，修改相关内容后确认。

##### 输出项

宿舍信息列表

#### 4.3.5 用户管理模块

##### 模块描述

管理员可以管理系统中的所有用户信息，学生可以管理自己的个人信息。管理员可以进行添加、删除、修改用户的操作，而学生可以修改自己的密码、联系方式等基本信息。

##### 功能

1. 管理员功能：管理所有用户（包括学生和管理员）的信息，进行查询、添加、删除、修改。
2. 学生功能：查看并修改自己的个人信息，如姓名、性别、联系方式、密码等。

##### 输入项

（1）管理员查询功能：管理员可以通过用户名、姓名等信息查询所有用户。

（2）学生个人信息修改功能：学生在个人信息页面修改自己的信息。

##### 输出项

（1）用户信息列表（管理员）：展示所有用户的信息。

（2）学生个人信息（学生）：展示当前学生的个人信息。

#### 4.3.6 通知管理模块

##### 模块描述

管理员可以发布通知给所有学生，通知可以包括宿舍管理、维修通知、活动安排等信息。学生可以查看通知的详细内容。

##### 功能

1. 发布通知：管理员可以新增通知，并指定发布时间、通知内容等。
2. 查询通知：管理员和学生都可以查看通知列表。
3. 删除通知：管理员可以删除不再需要的通知。
4. 查看通知详情：管理员和学生可以查看通知的详细内容。

##### 输入项

1. 新增通知功能：管理员在模态框中输入通知标题、内容等信息后发布。
2. 查询通知功能：通过输入关键词，管理员和学生可以查看相关通知。

##### 输出项

通知信息列表

#### 4.3.7 数据统计模块

##### 模块描述

系统可以展示各类数据统计信息，包括学生统计、住宿人数、报修统计和空宿舍统计。

##### 功能

统计各类需要统计的数据，例如学生统计、住宿人数、报修统计和空宿舍统计。

##### 输入项

选择需要统计的条件

##### 输出项

统计结果报表

### 4.4 系统实现

主要展示系统界面实现、关键代码和测试用例。

#### 4.4.0 登录界面

##### 界面展示



##### 代码实现

**Login.Vue：**

<template>  
 <div class="login-container">  
 <div style="margin-right: 150px; min-width: 300px">  
 <img alt="" src="../../public/login\_Team.png" style="width: 600px"/>  
 </div>  
 <div style="width: 400px; margin: 150px auto">  
 <div style="color: black; font-size: 30px; text-align: left; padding: 30px 0">登陆</div>  
 <el-form ref="form" :model="form" :rules="rules" size="large">  
 <el-form-item prop="username">  
 <el-input v-model="form.username" clearable prefix-icon="avatar"></el-input>  
 </el-form-item>  
 <el-form-item prop="password">  
 <el-input v-model="form.password" prefix-icon="lock" show-password></el-input>  
 </el-form-item>  
 <el-form-item :model="form" prop="identity">  
 <el-radio v-model="form.identity" label="stu" style="color: grey">学生</el-radio>  
 <el-radio v-model="form.identity" label="dormManager" style="color: grey">宿舍管理员</el-radio>  
 <el-radio v-model="form.identity" label="admin" style="color: grey">系统管理员</el-radio>  
 </el-form-item>  
 <el-form-item>  
 <el-button :disabled="!disabled" style="width: 20%" type="primary" @click="login">登 录</el-button>  
 </el-form-item>  
 </el-form>  
 </div>  
 </div>  
</template>  
<script src="@/assets/js/Login.js"></script>  
<style scoped>@import '../assets/css/Login.css';</style>

##### 测试用例

输入密码进行登录即可。

#### 4.4.1 首页页面

##### 界面展示



##### 代码实现

**前端代码 ：**

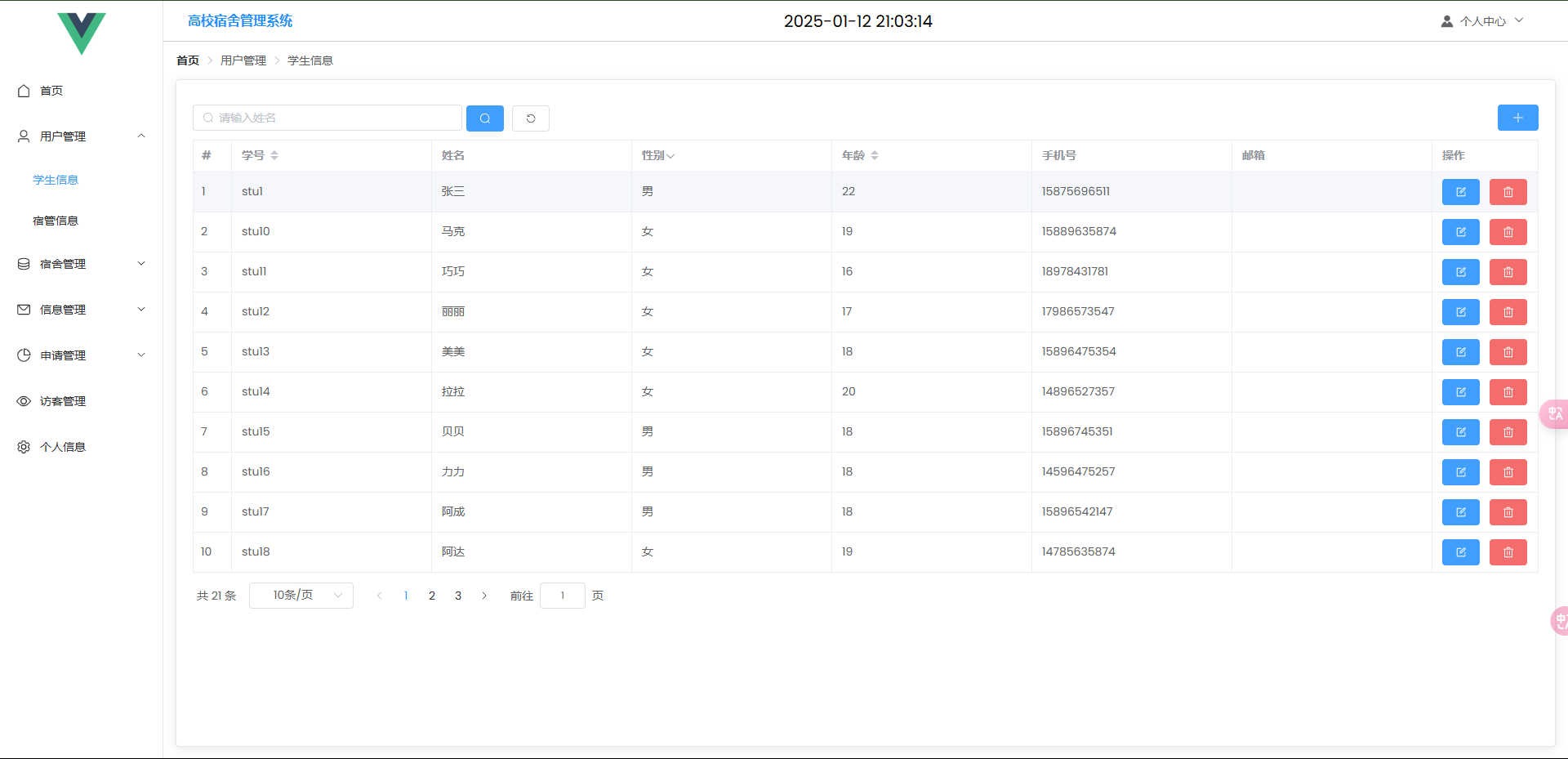
<template>  
 <el-card style="margin: 15px; min-height: calc(100vh - 80px)">  
 <!-- 头部数据-->  
 <div>  
 <el-row :gutter="20" class="topInfo">  
 <el-col :span="6">  
 <div id="stuNumDiv" class="el-colDiv">  
 <div id="ssv1-main-text" class="nowDiv">实时</div>  
 <span class="title">学生统计</span><br/>  
 <span class="digital">{{ this.studentNum }}</span><br/>  
 <span class="last-span">当前分类总记录数</span>  
 </div>  
 </el-col>  
 <el-col :span="6">  
 <div id="haveRoomDiv" class="el-colDiv">  
 <div id="ssv2-main-text" class="nowDiv">实时</div>  
 <span class="title">住宿人数</span><br/>  
 <span class="digital">{{ this.haveRoomStudentNum }}</span><br/>  
 <span class="last-span">当前分类总记录数</span>  
 </div>  
 </el-col>  
 <el-col :span="6">  
 <div id="repairNum" class="el-colDiv">  
 <div id="ssv3-main-text" class="nowDiv">实时</div>  
 <span class="title">报修统计</span><br/>  
 <span class="digital">{{ this.repairOrderNum }}</span><br/>  
 <span class="last-span">当前分类总记录数</span>  
 </div>  
 </el-col>  
 <el-col :span="6">  
 <div id="emptyRoom" class="el-colDiv">  
 <div id="ssv4-main-text" class="nowDiv">实时</div>  
 <span class="title">空宿舍统计</span><br/>  
 <span class="digital">{{ this.noFullRoomNum }}</span><br/>  
 <span class="last-span">当前分类总记录数</span>  
 </div>  
 </el-col>  
 </el-row>  
 </div>  
 <!-- 下部-->  
 <div style="display: flex;width: 100%;margin-top: 40px;align-items: center;justify-content: center;">  
 <!-- 左侧 宿舍通告-->  
 <div style="margin-right: 5%">  
 <span style="font-size: 22px;display: block;margin-bottom: 30px;margin-left: 10px;">宿舍通告</span>  
 <el-timeline>  
 <el-timeline-item v-for="(activity, index) in activities.slice(0, 8)" :key="index"  
 :timestamp="activity.releaseTime">  
 <span style="font-size: 15px">{{ activity.title }}</span>  
 </el-timeline-item>  
 </el-timeline>  
 </div>  
 <!-- 中部-->  
 <div style="height: 588px">  
 <span style="  
 font-size: 22px;  
 display: block;  
 margin-bottom: 30px;  
 margin-left: 10px;  
 ">宿舍学生人数分布图</span>  
 <home\_echarts/>  
 </div>  
 <!-- 右侧-->  
 <div style="margin-left: 5%">  
 <!-- 天气组件-->  
 <weather/>  
 <!-- 日历组件-->  
 <el-card style="width: 380px; max-height: 440px; margin-top: 17px">  
 <Calender/>  
 </el-card>  
 </div>  
 </div>  
 </el-card>  
</template>  
<script src="@/assets/js/Home.js"></script>  
<style scoped>@import '../assets/css/Home.css';</style>

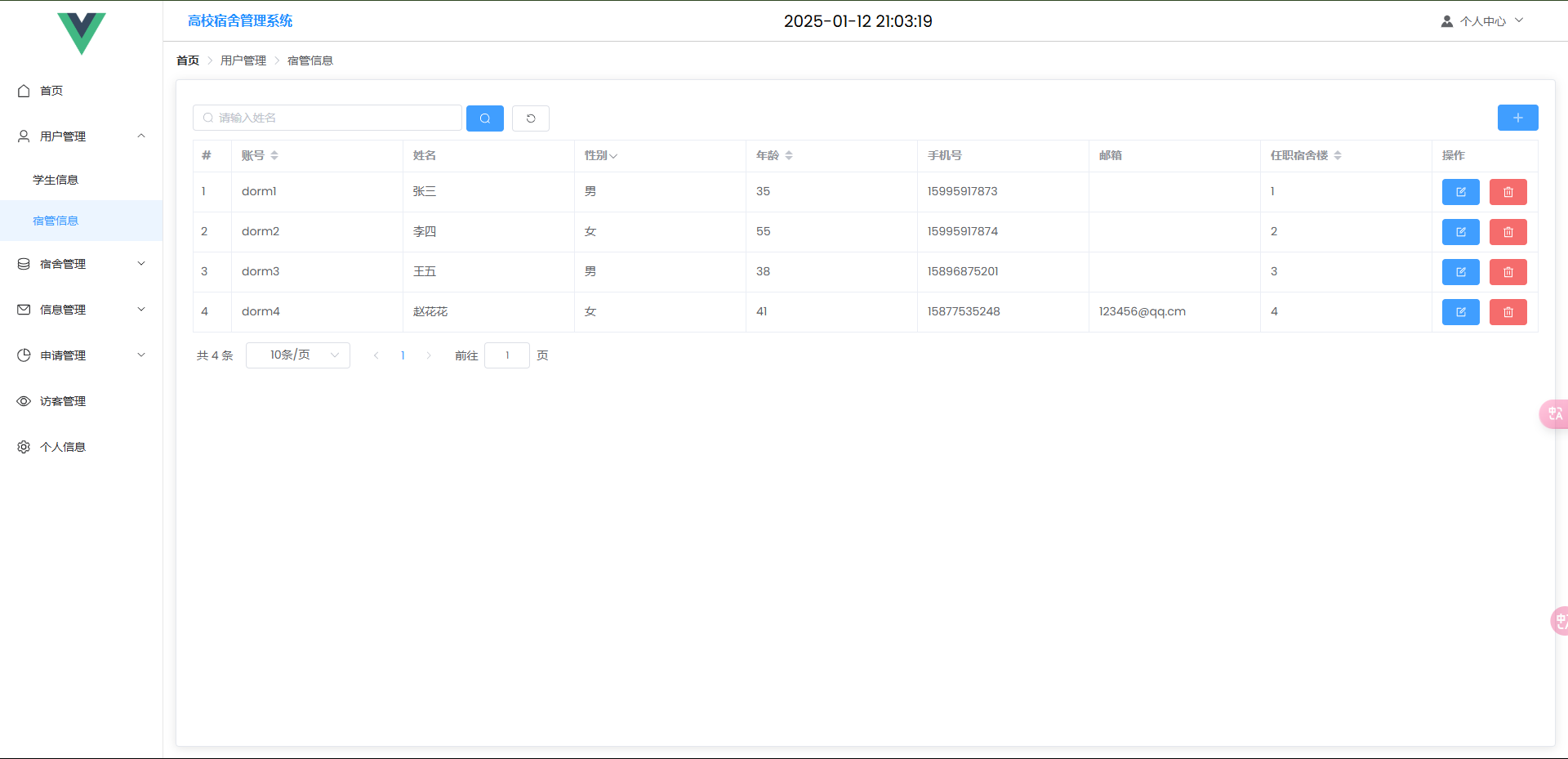
##### 测试用例

无（不涉及功能， 首页仅作界面展示）

#### 4.4.2 用户管理界面

##### 界面展示





##### 代码实现

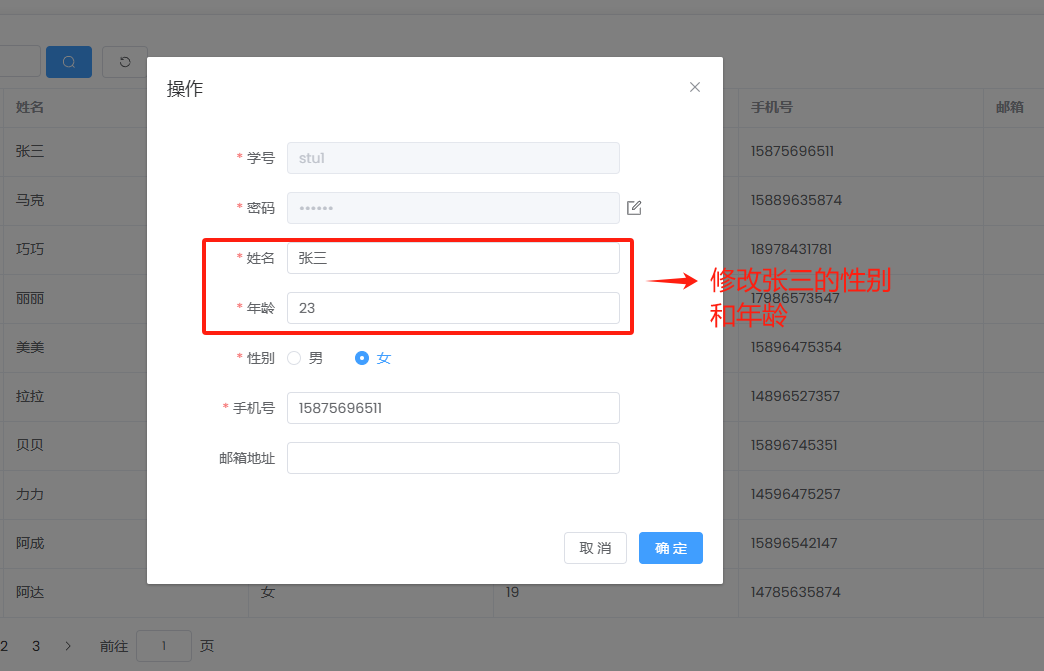
**1、StudentServiceImpl :**

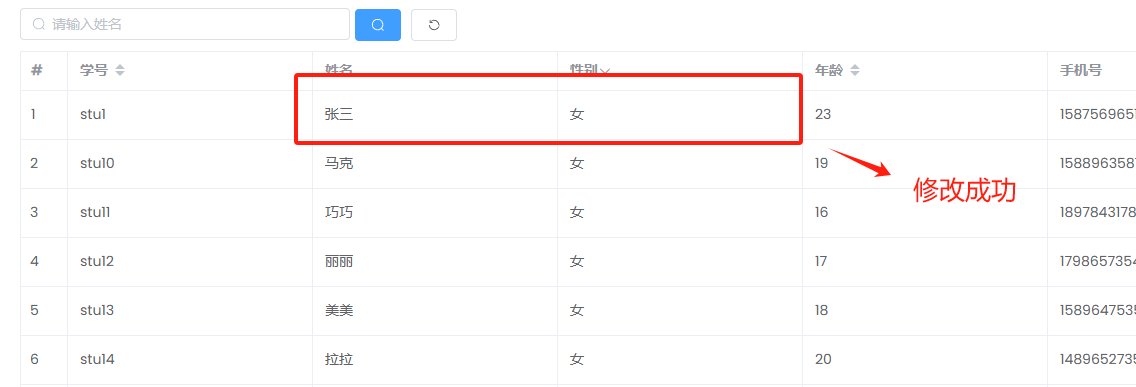
package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.entity.Student;  
import com.example.springboot.mapper.StudentMapper;  
import com.example.springboot.service.StudentService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
  
/\*\*  
 \* @author AthenaKnovesp  
 \*/  
@Service  
public class StudentServiceImpl extends ServiceImpl<StudentMapper, Student> implements StudentService {  
  
 /\*\*  
 \* 注入DAO层对象  
 \*/  
 @Resource  
 private StudentMapper studentMapper;  
  
 /\*\*  
 \* 学生登陆  
 \*/  
 @Override  
 public Student stuLogin(String username, String password) {  
 QueryWrapper<Student> qw = new QueryWrapper<>();  
 qw.eq("username", username);  
 qw.eq("password", password);  
 Student student = studentMapper.selectOne(qw);  
 if (student != null) {  
 return student;  
 } else {  
 return null;  
 }  
 }  
  
 /\*\*  
 \* 学生新增  
 \*/  
 @Override  
 public int addNewStudent(Student student) {  
 int insert = studentMapper.insert(student);  
 return insert;  
 }  
  
 /\*\*  
 \* 分页查询学生  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<Student> qw = new QueryWrapper<>();  
 qw.like("name", search);  
 Page studentPage = studentMapper.selectPage(page, qw);  
 return studentPage;  
 }  
  
 /\*\*  
 \* 更新学生信息  
 \*/  
 @Override  
 public int updateNewStudent(Student student) {  
 int i = studentMapper.updateById(student);  
 return i;  
 }  
  
 /\*\*  
 \* 删除学生信息  
 \*/  
 @Override  
 public int deleteStudent(String username) {  
 int i = studentMapper.deleteById(username);  
 return i;  
 }  
  
  
 /\*\*  
 \* 主页顶部：学生统计  
 \*/  
 @Override  
 public int stuNum() {  
 QueryWrapper<Student> qw = new QueryWrapper<>();  
 qw.isNotNull("username");  
 int stuNum = Math.toIntExact(studentMapper.selectCount(qw));  
 return stuNum;  
 }  
  
 /\*\*  
 \* 床位信息，查询该学生信息  
 \*/  
 @Override  
 public Student stuInfo(String username) {  
 QueryWrapper<Student> qw = new QueryWrapper<>();  
 qw.eq("username", username);  
 Student student = studentMapper.selectOne(qw);  
 return student;  
 }  
}

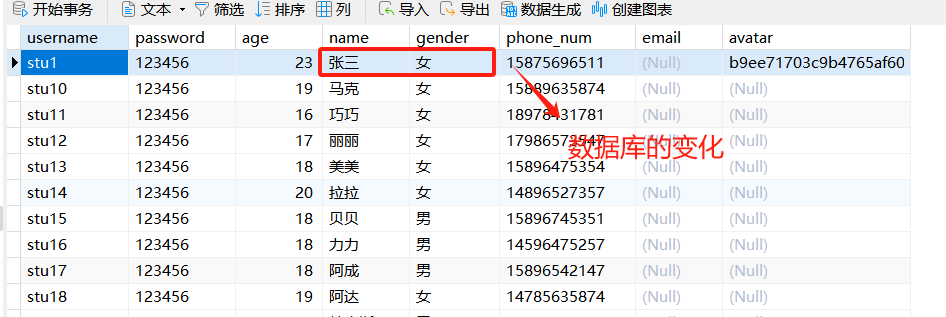
1. **DormManagerServiceImpl :**

package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.entity.DormManager;  
import com.example.springboot.mapper.DormManagerMapper;  
import com.example.springboot.service.DormManagerService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
  
/\*\*  
 \* @author AthenaKnovesp  
 \*/  
  
@Service  
public class DormManagerServiceImpl extends ServiceImpl<DormManagerMapper, DormManager> implements DormManagerService {  
  
 /\*\*  
 \* 注入DAO层对象  
 \*/  
 @Resource  
 private DormManagerMapper dormManagerMapper;  
  
 /\*\*  
 \* 宿管登录  
 \*/  
 @Override  
 public DormManager dormManagerLogin(String username, String password) {  
 QueryWrapper<DormManager> qw = new QueryWrapper<>();  
 qw.eq("username", username);  
 qw.eq("password", password);  
 DormManager dormManager = dormManagerMapper.selectOne(qw);  
 if (dormManager != null) {  
 return dormManager;  
 } else {  
 return null;  
 }  
 }  
  
 /\*\*  
 \* 宿管新增  
 \*/  
 @Override  
 public int addNewDormManager(DormManager dormManager) {  
 int insert = dormManagerMapper.insert(dormManager);  
 return insert;  
 }  
  
 /\*\*  
 \* 宿管查找  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<DormManager> qw = new QueryWrapper<>();  
 qw.like("name", search);  
 Page dormManagerPage = dormManagerMapper.selectPage(page, qw);  
 return dormManagerPage;  
 }  
  
 /\*\*  
 \* 宿管信息更新  
 \*/  
 @Override  
 public int updateNewDormManager(DormManager dormManager) {  
 int i = dormManagerMapper.updateById(dormManager);  
 return i;  
 }  
  
 /\*\*  
 \* 宿管删除  
 \*/  
 @Override  
 public int deleteDormManager(String username) {  
 int i = dormManagerMapper.deleteById(username);  
 return i;  
 }  
}

##### 测试用例

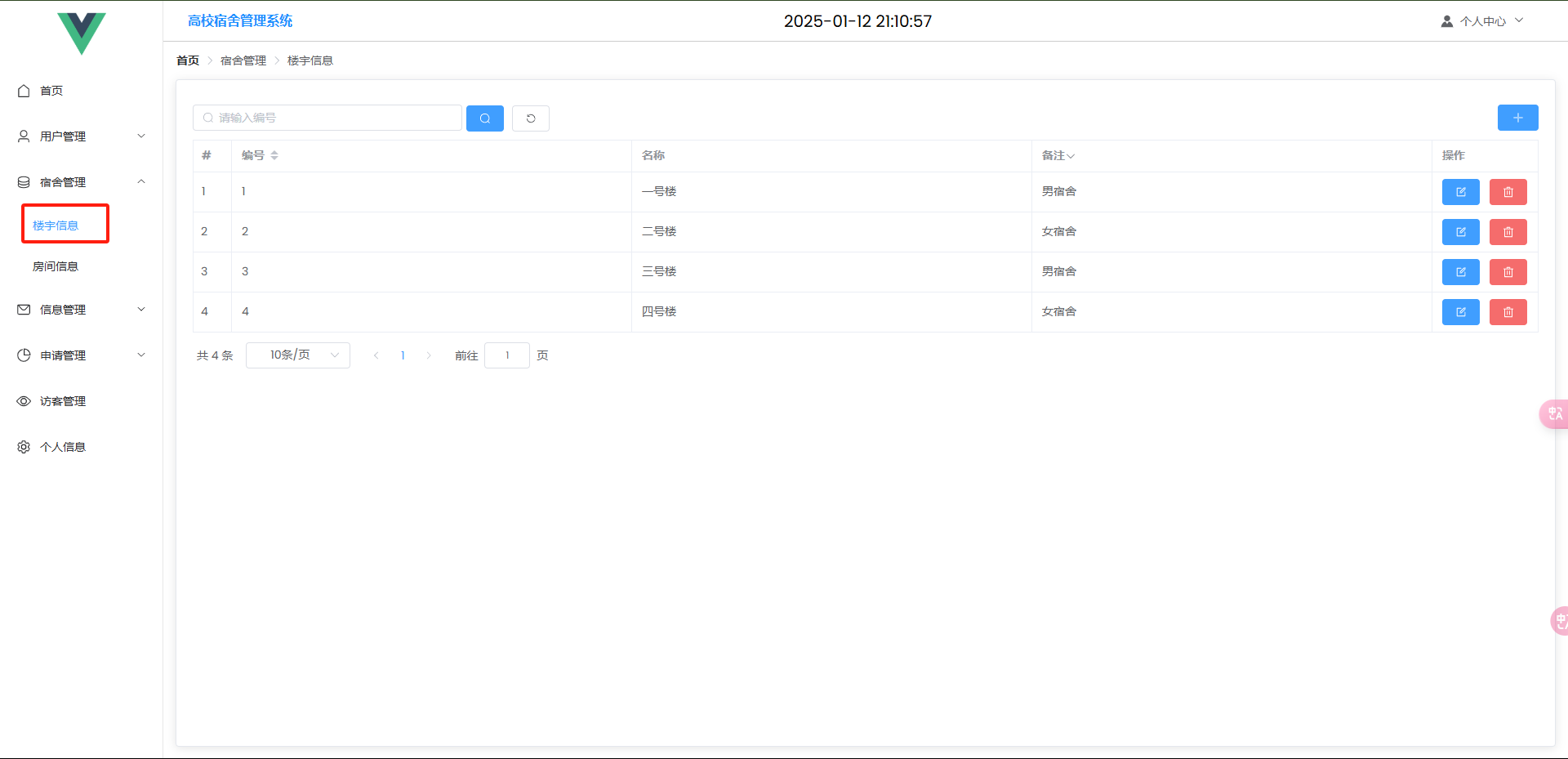


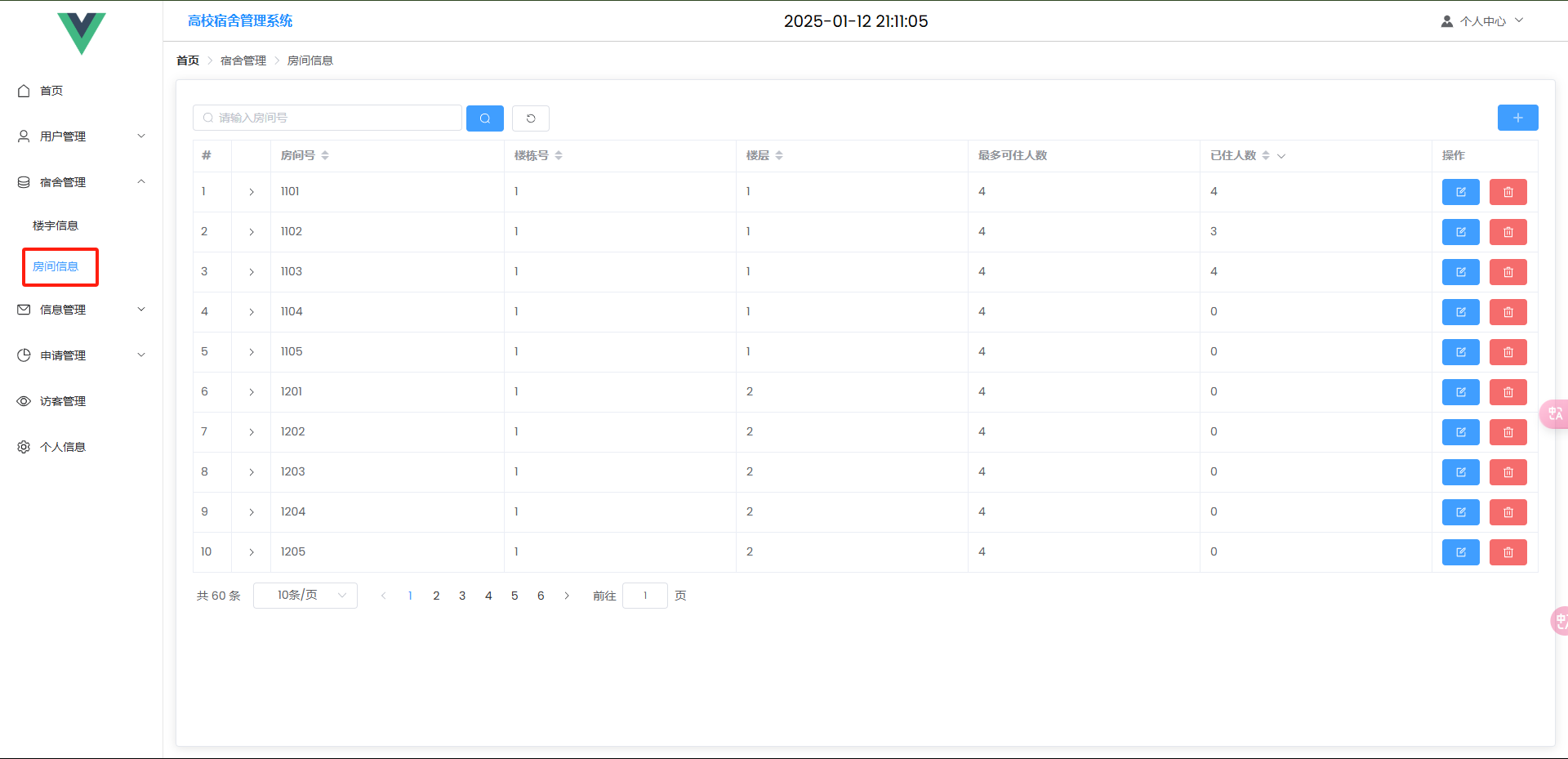




#### 4.4.3 宿舍管理界面

##### 界面展示





##### 代码实现

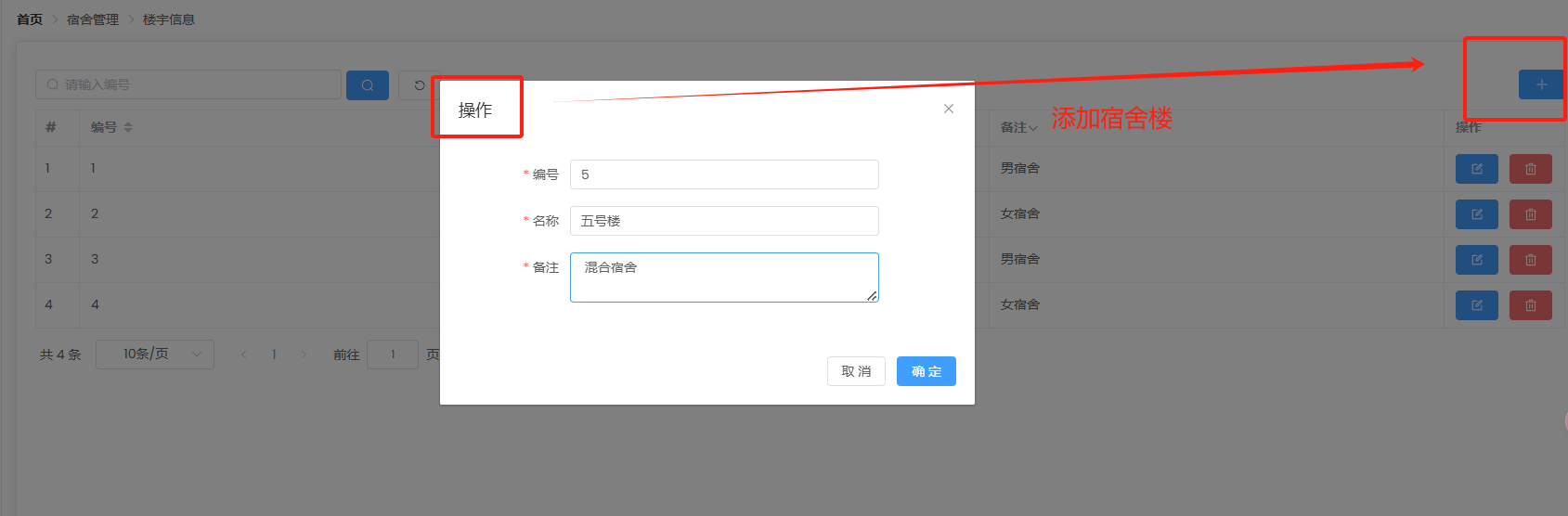
1. **DromBuildImpl ：**

package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.entity.DormBuild;  
import com.example.springboot.mapper.DormBuildMapper;  
import com.example.springboot.service.DormBuildService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
import java.util.List;  
  
/\*\*  
 \* @author AthenaKnovesp  
 \*/  
@Service  
public class DormBuildImpl extends ServiceImpl<DormBuildMapper, DormBuild> implements DormBuildService {  
  
  
 /\*\*  
 \* 注入DAO层对象  
 \*/  
 @Resource  
 private DormBuildMapper dormBuildMapper;  
  
 /\*\*  
 \* 楼宇添加  
 \*/  
 @Override  
 public int addNewBuilding(DormBuild dormBuild) {  
 int insert = dormBuildMapper.insert(dormBuild);  
 return insert;  
 }  
  
 /\*\*  
 \* 楼宇查找  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<DormBuild> qw = new QueryWrapper<>();  
 qw.like("DormBuild\_id", search);  
 Page buildingPage = dormBuildMapper.selectPage(page, qw);  
 return buildingPage;  
 }  
  
 /\*\*  
 \* 楼宇信息更新  
 \*/  
 @Override  
 public int updateNewBuilding(DormBuild dormBuild) {  
 int i = dormBuildMapper.updateById(dormBuild);  
 return i;  
 }  
  
 /\*\*  
 \* 楼宇删除  
 \*/  
 @Override  
 public int deleteBuilding(Integer id) {  
 int i = dormBuildMapper.deleteById(id);  
 return i;  
 }  
  
 /\*\*  
 \* 首页 获取建筑名称  
 \*/  
 @Override  
 public List<DormBuild> getBuildingId() {  
 QueryWrapper<DormBuild> qw = new QueryWrapper<>();  
 qw.select("dormbuild\_id");  
 List<DormBuild> dormBuilds = dormBuildMapper.selectList(qw);  
 return dormBuilds;  
 }  
  
}

1. **DormRoomImpl :**

package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.common.JudgeBedName;  
import com.example.springboot.entity.AdjustRoom;  
import com.example.springboot.entity.DormRoom;  
import com.example.springboot.mapper.DormRoomMapper;  
import com.example.springboot.service.DormRoomService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
  
import static com.example.springboot.common.CalPeopleNum.calNum;  
  
  
/\*\*  
 \* @author AthenaKnovesp  
 \*/  
@Service  
public class DormRoomImpl extends ServiceImpl<DormRoomMapper, DormRoom> implements DormRoomService {  
  
 @Resource  
 private DormRoomMapper dormRoomMapper;  
  
 /\*\*  
 \* 首页顶部：空宿舍统计  
 \*/  
 @Override  
 public int notFullRoom() {  
 QueryWrapper<DormRoom> qw = new QueryWrapper<>();  
 qw.lt("current\_capacity", 4);  
 int notFullRoomNum = Math.toIntExact(dormRoomMapper.selectCount(qw));  
 return notFullRoomNum;  
 }  
  
 /\*\*  
 \* 添加房间  
 \*/  
 @Override  
 public int addNewRoom(DormRoom dormRoom) {  
 int insert = dormRoomMapper.insert(dormRoom);  
 return insert;  
 }  
  
 /\*\*  
 \* 查找房间  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<DormRoom> qw = new QueryWrapper<>();  
 qw.like("Dormroom\_id", search);  
 Page roomPage = dormRoomMapper.selectPage(page, qw);  
 return roomPage;  
 }  
  
 /\*\*  
 \* 更新房间  
 \*/  
 @Override  
 public int updateNewRoom(DormRoom dormRoom) {  
 int i = dormRoomMapper.updateById(dormRoom);  
 return i;  
 }  
  
 /\*\*  
 \* 删除房间  
 \*/  
 @Override  
 public int deleteRoom(Integer dormRoomId) {  
 int i = dormRoomMapper.deleteById(dormRoomId);  
 return i;  
 }  
  
 /\*\*  
 \* 删除床位上的学生信息  
 \*/  
 @Override  
 public int deleteBedInfo(String bedName, Integer dormRoomId, int calCurrentNum) {  
 UpdateWrapper updateWrapper = new UpdateWrapper();  
 updateWrapper.eq("dormroom\_id", dormRoomId);  
 updateWrapper.set(bedName, null);  
 updateWrapper.set("current\_capacity", calCurrentNum - 1);  
 int update = dormRoomMapper.update(null, updateWrapper);  
 return update;  
  
 }  
  
 /\*\*  
 \* 床位信息，查询该学生是否已由床位  
 \*/  
 @Override  
 public DormRoom judgeHadBed(String username) {  
 QueryWrapper qw = new QueryWrapper();  
 qw.eq("first\_bed", username);  
 qw.or();  
 qw.eq("second\_bed", username);  
 qw.or();  
 qw.eq("third\_bed", username);  
 qw.or();  
 qw.eq("fourth\_bed", username);  
 DormRoom dormRoom = dormRoomMapper.selectOne(qw);  
 return dormRoom;  
 }  
  
 /\*\*  
 \* 主页 住宿人数  
 \*/  
 @Override  
 public Long selectHaveRoomStuNum() {  
 QueryWrapper<DormRoom> qw1 = new QueryWrapper<>();  
 qw1.isNotNull("first\_bed");  
 Long first\_bed = dormRoomMapper.selectCount(qw1);  
  
 QueryWrapper<DormRoom> qw2 = new QueryWrapper<>();  
 qw2.isNotNull("second\_bed");  
 Long second\_bed = dormRoomMapper.selectCount(qw2);  
  
 QueryWrapper<DormRoom> qw3 = new QueryWrapper<>();  
 qw3.isNotNull("third\_bed");  
 Long third\_bed = dormRoomMapper.selectCount(qw3);  
  
 QueryWrapper<DormRoom> qw4 = new QueryWrapper<>();  
 qw4.isNotNull("fourth\_bed");  
 Long fourth\_bed = dormRoomMapper.selectCount(qw4);  
  
 Long count = first\_bed + second\_bed + third\_bed + fourth\_bed;  
 return count;  
 }  
  
 /\*\*  
 \* 获取每栋宿舍学生总人数  
 \*/  
 @Override  
 public Long getEachBuildingStuNum(int dormBuildId) {  
  
 QueryWrapper<DormRoom> qw1 = new QueryWrapper<>();  
 qw1.eq("dormbuild\_id", dormBuildId);  
 qw1.isNotNull("first\_bed");  
 Long first\_bed = dormRoomMapper.selectCount(qw1);  
  
 QueryWrapper<DormRoom> qw2 = new QueryWrapper<>();  
 qw2.eq("dormbuild\_id", dormBuildId);  
 qw2.isNotNull("second\_bed");  
 Long second\_bed = dormRoomMapper.selectCount(qw2);  
  
 QueryWrapper<DormRoom> qw3 = new QueryWrapper<>();  
 qw3.eq("dormbuild\_id", dormBuildId);  
 qw3.isNotNull("third\_bed");  
 Long third\_bed = dormRoomMapper.selectCount(qw3);  
  
 QueryWrapper<DormRoom> qw4 = new QueryWrapper<>();  
 qw4.eq("dormbuild\_id", dormBuildId);  
 qw4.isNotNull("fourth\_bed");  
 Long fourth\_bed = dormRoomMapper.selectCount(qw4);  
  
 Long count = first\_bed + second\_bed + third\_bed + fourth\_bed;  
  
 return count;  
 }  
  
 /\*\*  
 \* 根据调宿申请表对房间表内的学生床位进行调整  
 \*/  
 @Override  
 public int adjustRoomUpdate(AdjustRoom adjustRoom) {  
 //调宿人  
 String username = adjustRoom.getUsername();  
 //当前房间号  
 int currentRoomId = adjustRoom.getCurrentRoomId();  
 //当前床位名称  
 String currentBedName = JudgeBedName.getBedName(adjustRoom.getCurrentBedId());  
 //目标房间号  
 int towardsRoomId = adjustRoom.getTowardsRoomId();  
 //目标目标房间号  
 String towardsBedName = JudgeBedName.getBedName(adjustRoom.getTowardsBedId());  
  
 QueryWrapper qw = new QueryWrapper();  
 qw.eq("dormroom\_id", currentRoomId);  
 qw.isNotNull(currentBedName);  
 DormRoom dormRoom1 = dormRoomMapper.selectOne(qw);  
  
 if (dormRoom1 == null) {  
 return -2;  
 }  
 int currentCapacity1 = calNum(dormRoom1);  
  
 UpdateWrapper uw1 = new UpdateWrapper();  
 uw1.eq("dormroom\_id", currentRoomId);  
 uw1.set(currentBedName, null);  
 uw1.set("current\_capacity", currentCapacity1 - 1);  
 int result1 = dormRoomMapper.update(null, uw1);  
  
 DormRoom dormRoom2 = dormRoomMapper.selectById(towardsRoomId);  
 int currentCapacity2 = calNum(dormRoom2);  
  
 if (result1 == 1) {  
 UpdateWrapper uw2 = new UpdateWrapper();  
 uw2.eq("dormroom\_id", towardsRoomId);  
 uw2.set(towardsBedName, username);  
 uw2.set("current\_capacity", currentCapacity2 + 1);  
 int result2 = dormRoomMapper.update(null, uw2);  
 return result2;  
 }  
 return -1;  
 }  
  
  
 /\*\*  
 \* 检查该房间是否满了  
 \*/  
 @Override  
 public DormRoom checkRoomState(Integer dormRoomId) {  
 QueryWrapper qw = new QueryWrapper();  
 qw.eq("dormroom\_id", dormRoomId);  
 qw.lt("current\_capacity", 4);  
 DormRoom dormRoom = dormRoomMapper.selectOne(qw);  
 return dormRoom;  
 }  
  
 /\*\*  
 \* 检查该房间是否存在  
 \*/  
 @Override  
 public DormRoom checkRoomExist(Integer dormRoomId) {  
 DormRoom dormRoom = dormRoomMapper.selectById(dormRoomId);  
 return dormRoom;  
 }  
  
  
 /\*\*  
 \* 检查床位是否有人  
 \*/  
 @Override  
 public DormRoom checkBedState(Integer dormRoomId, int bedNum) {  
 QueryWrapper qw = new QueryWrapper();  
 qw.eq("dormroom\_id", dormRoomId);  
 qw.isNull(JudgeBedName.getBedName(bedNum));  
 DormRoom dormRoom = dormRoomMapper.selectOne(qw);  
 return dormRoom;  
 }  
}

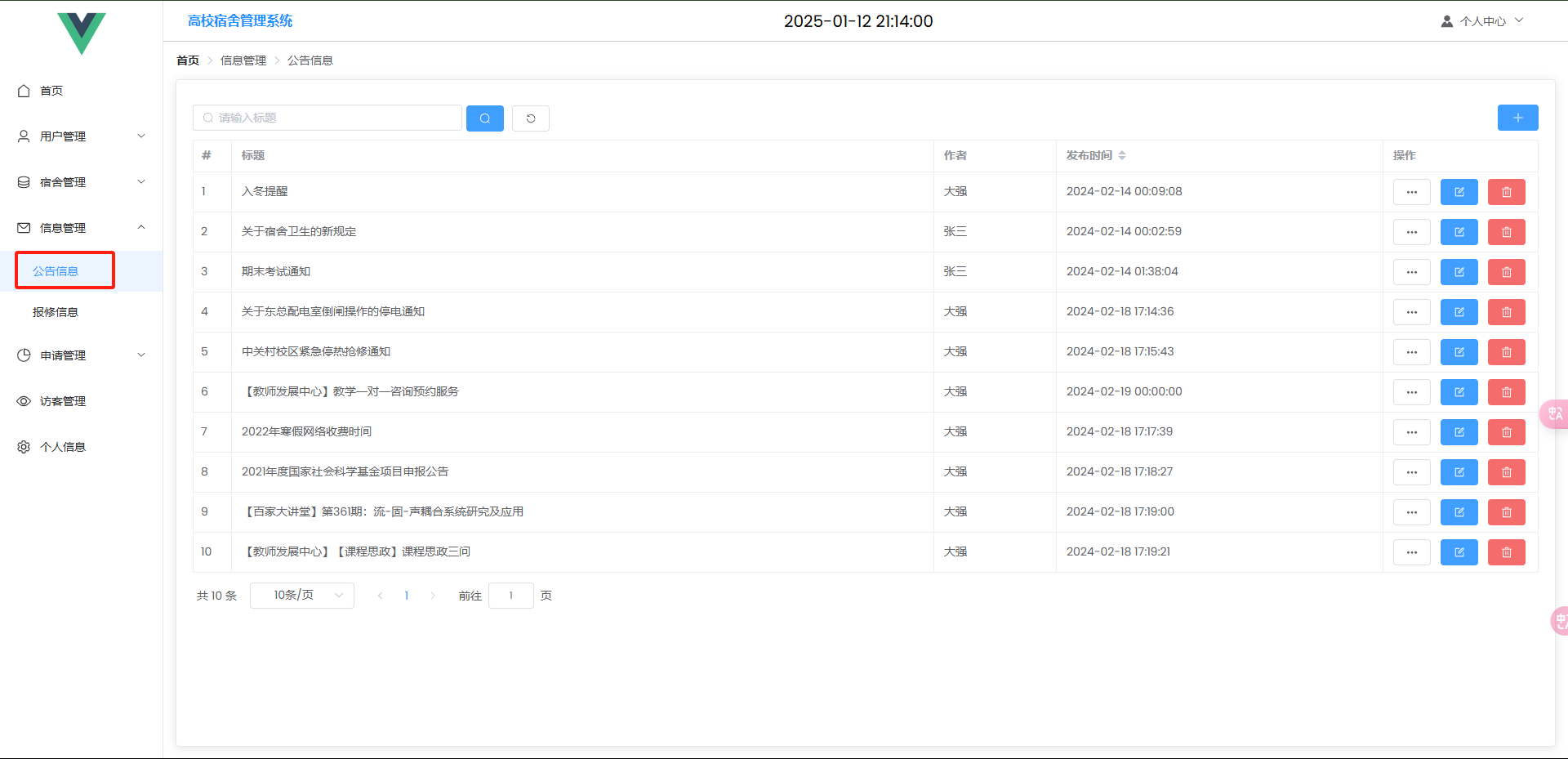
##### 测试用例

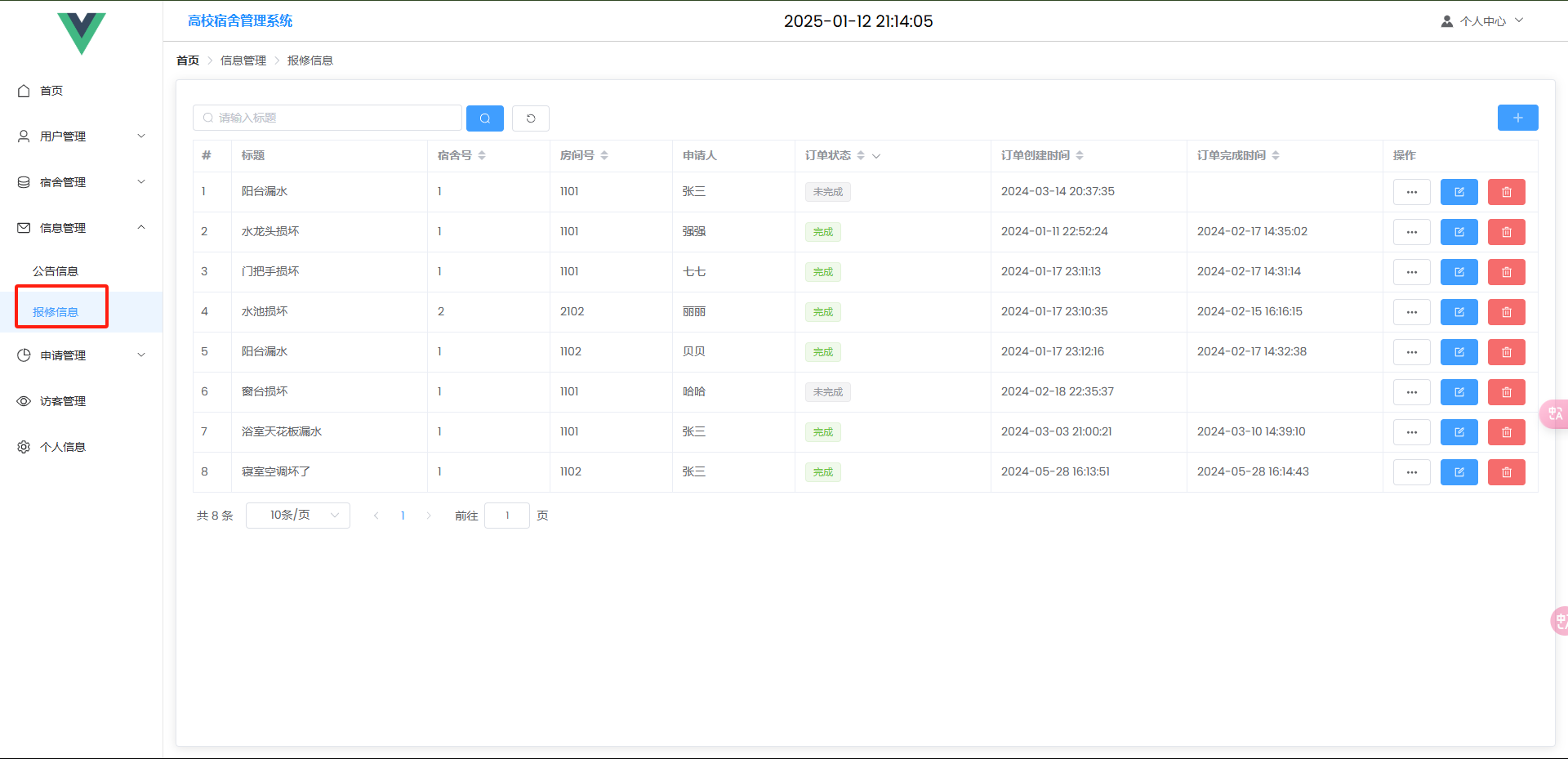




#### 4.4.4 信息管理界面

##### 界面展示





##### 代码实现

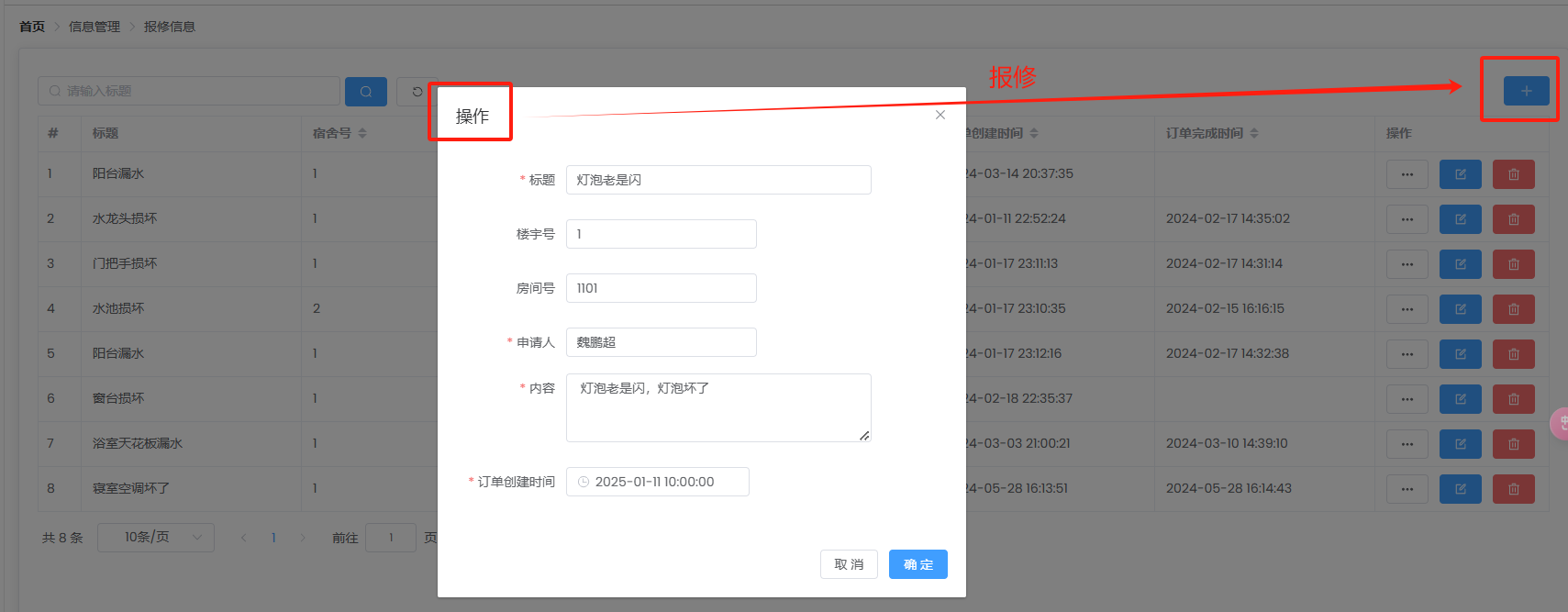
1. **NoticeServiceImpl:**

package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.entity.Notice;  
import com.example.springboot.mapper.NoticeMapper;  
import com.example.springboot.service.NoticeService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
import java.util.List;  
  
/\*\*  
 \* @author AthenaKnovesp  
 \*/  
@Service  
public class NoticeServiceImpl extends ServiceImpl<NoticeMapper, Notice> implements NoticeService {  
  
  
 /\*\*  
 \* 注入DAO层对象  
 \*/  
 @Resource  
 private NoticeMapper noticeMapper;  
  
 /\*\*  
 \* 公告添加  
 \*/  
 @Override  
 public int addNewNotice(Notice notice) {  
 int insert = noticeMapper.insert(notice);  
 return insert;  
 }  
  
 /\*\*  
 \* 公告查找  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<Notice> qw = new QueryWrapper<>();  
 qw.like("title", search);  
 Page noticePage = noticeMapper.selectPage(page, qw);  
 return noticePage;  
 }  
  
 /\*\*  
 \* 公告更新  
 \*/  
 @Override  
 public int updateNewNotice(Notice notice) {  
 int i = noticeMapper.updateById(notice);  
 return i;  
 }  
  
 /\*\*  
 \* 公告删除  
 \*/  
 @Override  
 public int deleteNotice(Integer id) {  
 int i = noticeMapper.deleteById(id);  
 return i;  
 }  
  
 /\*\*  
 \* 首页公告展示  
 \*/  
 @Override  
 public List<?> homePageNotice() {  
 QueryWrapper<Notice> qw = new QueryWrapper<>();  
 qw.orderByDesc("release\_time");  
 List<Notice> noticeList = noticeMapper.selectList(qw);  
 return noticeList;  
 }  
}

1. **RepairServiceImpl ：**

package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.entity.Repair;  
import com.example.springboot.mapper.RepairMapper;  
import com.example.springboot.service.RepairService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
  
/\*\*  
 \* @author AthenaKnovesp  
 \*/  
  
@Service  
public class RepairServiceImpl extends ServiceImpl<RepairMapper, Repair> implements RepairService {  
  
 /\*\*  
 \* 注入DAO层对象  
 \*/  
 @Resource  
 private RepairMapper repairMapper;  
  
 /\*\*  
 \* 添加订单  
 \*/  
 @Override  
 public int addNewOrder(Repair repair) {  
 int insert = repairMapper.insert(repair);  
 return insert;  
 }  
  
 /\*\*  
 \* 查找订单  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<Repair> qw = new QueryWrapper<>();  
 qw.like("title", search);  
 Page orderPage = repairMapper.selectPage(page, qw);  
 return orderPage;  
 }  
  
 @Override  
 public Page individualFind(Integer pageNum, Integer pageSize, String search, String name) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<Repair> qw = new QueryWrapper<>();  
 qw.like("title", search);  
 qw.eq("repairer", name);  
 Page orderPage = repairMapper.selectPage(page, qw);  
 return orderPage;  
 }  
  
 /\*\*  
 \* 更新订单  
 \*/  
 @Override  
 public int updateNewOrder(Repair repair) {  
 int i = repairMapper.updateById(repair);  
 return i;  
 }  
  
 /\*\*  
 \* 删除订单  
 \*/  
 @Override  
 public int deleteOrder(Integer id) {  
 int i = repairMapper.deleteById(id);  
 return i;  
 }  
  
 /\*\*  
 \* 首页顶部：报修统计  
 \*/  
 @Override  
 public int showOrderNum() {  
 QueryWrapper<Repair> qw = new QueryWrapper<>();  
 int orderCount = Math.toIntExact(repairMapper.selectCount(qw));  
 return orderCount;  
 }  
}

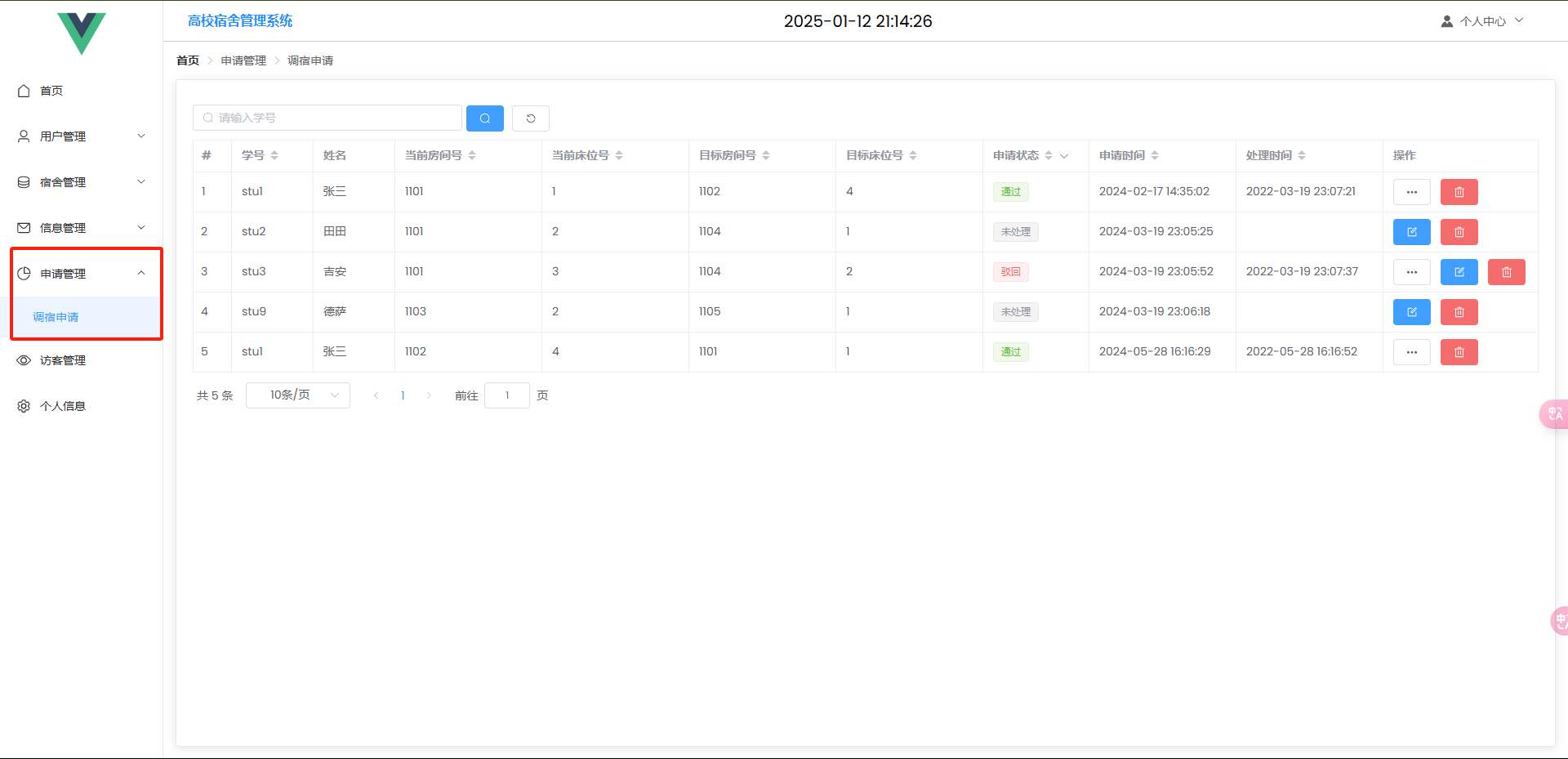
##### 测试用例





#### 4.4.5 申请管理界面

##### 界面展示

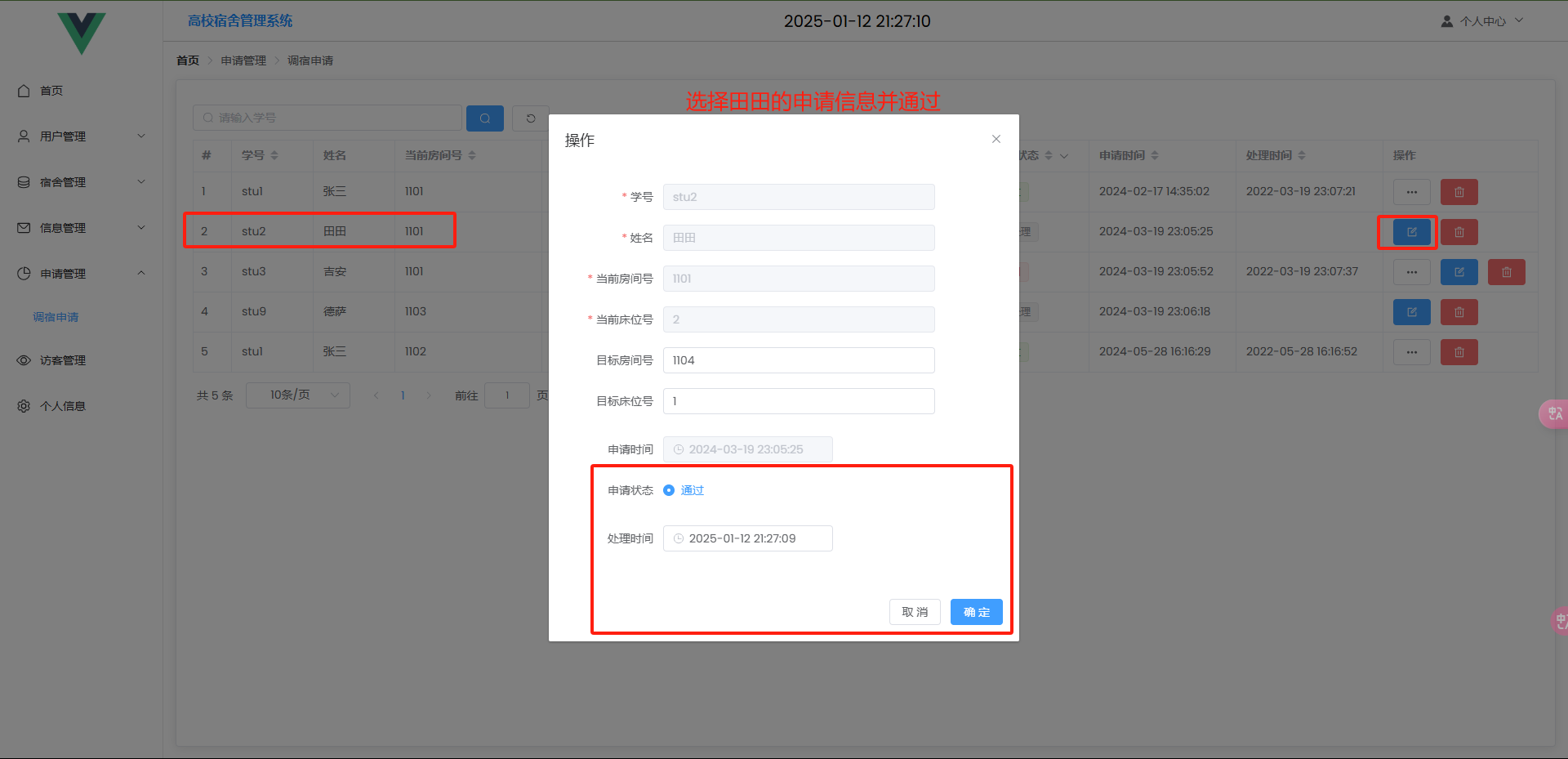


##### 代码实现

**AdjustRoomServiceImpl :**

package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.entity.AdjustRoom;  
import com.example.springboot.mapper.AdjustRoomMapper;  
import com.example.springboot.service.AdjustRoomService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
  
@Service  
public class AdjustRoomServiceImpl extends ServiceImpl<AdjustRoomMapper, AdjustRoom> implements AdjustRoomService {  
  
  
 @Resource  
 private AdjustRoomMapper adjustRoomMapper;  
  
 /\*\*  
 \* 添加调宿申请  
 \*/  
 @Override  
 public int addApply(AdjustRoom adjustRoom) {  
 int insert = adjustRoomMapper.insert(adjustRoom);  
 return insert;  
 }  
  
 /\*\*  
 \* 查找调宿申请  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<AdjustRoom> qw = new QueryWrapper<>();  
 qw.like("username", search);  
 Page orderPage = adjustRoomMapper.selectPage(page, qw);  
 return orderPage;  
 }  
  
 /\*\*  
 \* 删除调宿申请  
 \*/  
 @Override  
 public int deleteAdjustment(Integer id) {  
 int i = adjustRoomMapper.deleteById(id);  
 return i;  
 }  
  
  
 /\*\*  
 \* 更新调宿申请  
 \*/  
 @Override  
 public int updateApply(AdjustRoom adjustRoom) {  
 int i = adjustRoomMapper.updateById(adjustRoom);  
 return i;  
 }  
  
  
}

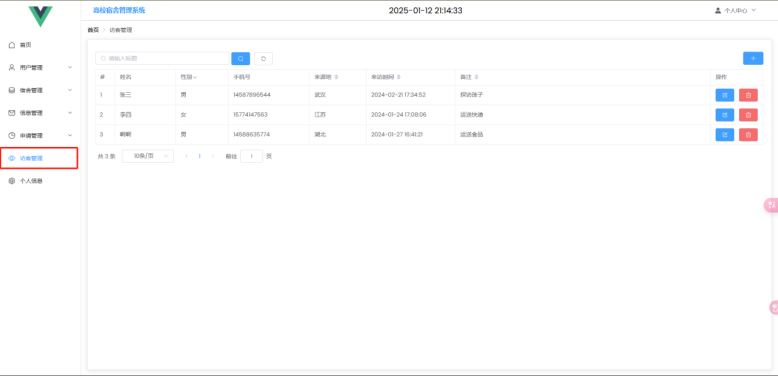
##### 测试用例





#### 4.4.6 访客管理界面

##### 界面展示

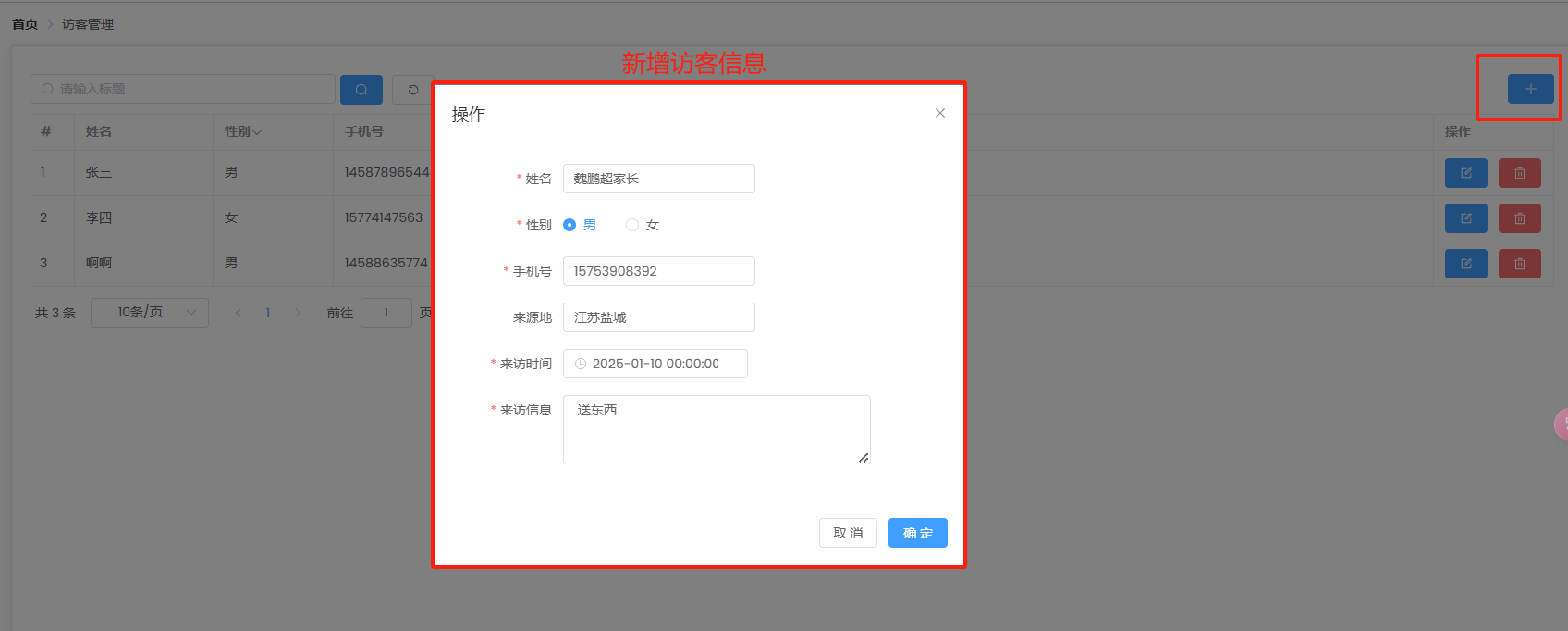


##### 代码实现

**VisitorServiceImpl :**

package com.example.springboot.service.impl;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.example.springboot.entity.Visitor;  
import com.example.springboot.mapper.VisitorMapper;  
import com.example.springboot.service.VisitorService;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
  
/\*\*  
 \* @author AthenaKnovesp  
 \*/  
@Service  
public class VisitorServiceImpl extends ServiceImpl<VisitorMapper, Visitor> implements VisitorService {  
  
 @Resource  
 private VisitorMapper visitorMapper;  
  
 /\*\*  
 \* 访客添加  
 \*/  
 @Override  
 public int addNewVisitor(Visitor visitor) {  
 int insert = visitorMapper.insert(visitor);  
 return insert;  
 }  
  
 /\*\*  
 \* 访客查询  
 \*/  
 @Override  
 public Page find(Integer pageNum, Integer pageSize, String search) {  
 Page page = new Page<>(pageNum, pageSize);  
 QueryWrapper<Visitor> qw = new QueryWrapper<>();  
 qw.like("name", search);  
 Page visitorPage = visitorMapper.selectPage(page, qw);  
 return visitorPage;  
 }  
  
 /\*\*  
 \* 访客信息更新  
 \*/  
 @Override  
 public int updateNewVisitor(Visitor visitor) {  
 int i = visitorMapper.updateById(visitor);  
 return i;  
 }  
  
 /\*\*  
 \* 访客删除  
 \*/  
 @Override  
 public int deleteVisitor(Integer id) {  
 int i = visitorMapper.deleteById(id);  
 return i;  
 }  
}

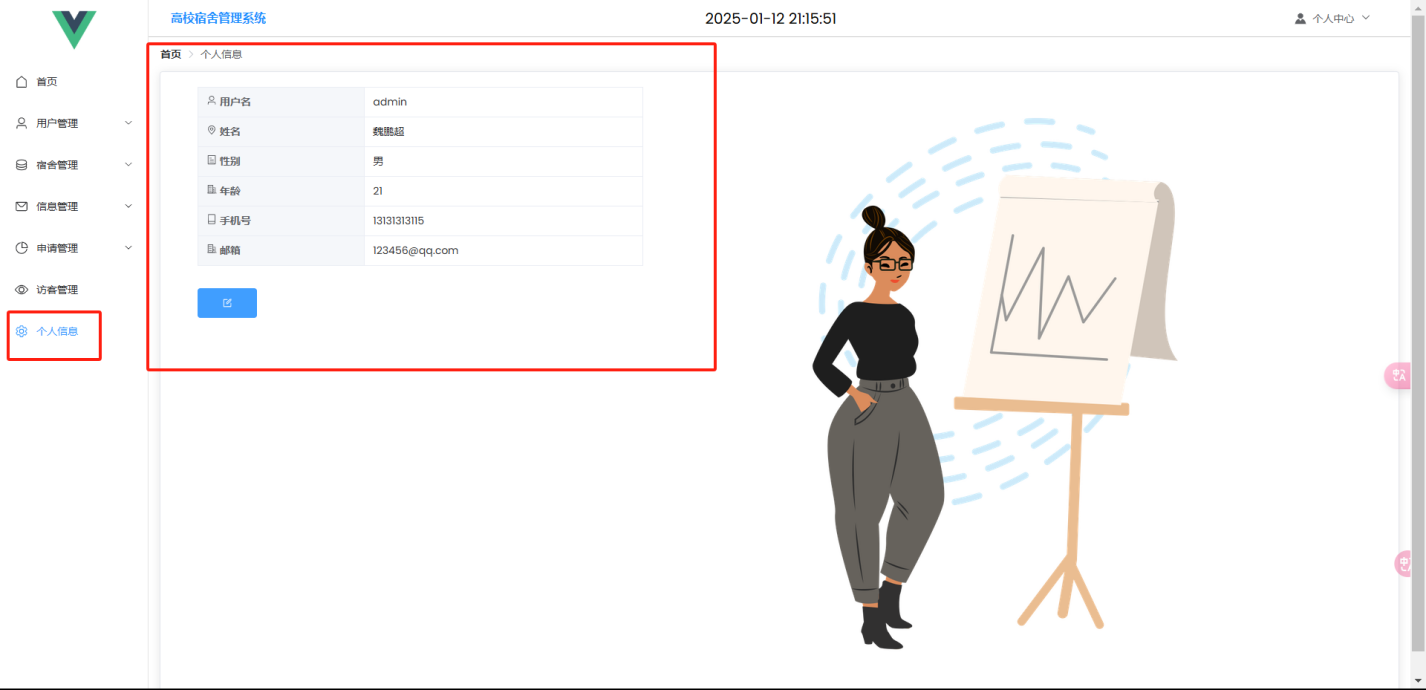
##### 测试用例





#### 4.4.7 个人信息界面

##### 界面展示

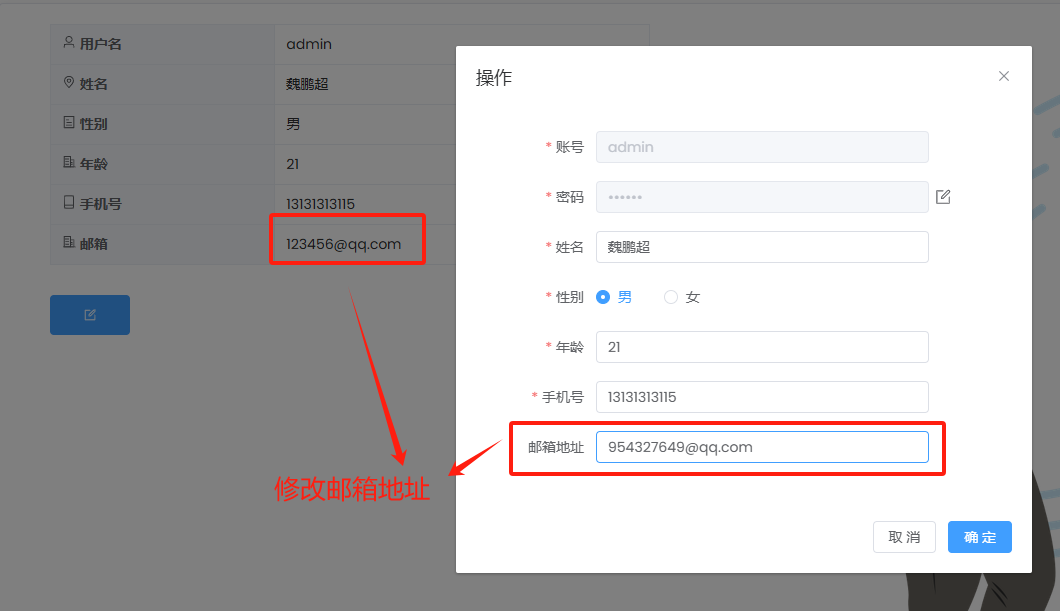


##### 代码实现

**SelfInfo.Vue :**

<template>  
 <div>  
 <el-breadcrumb separator-icon="ArrowRight" style="margin: 16px">  
 <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>  
 <el-breadcrumb-item>个人信息</el-breadcrumb-item>  
 </el-breadcrumb>  
 <el-card style="margin: 15px; min-height: calc(100vh - 111px)">  
 <div style="display: flex">  
 <div style="width: 600px; margin-left: 30px; position: relative">  
<!-- <div>-->  
<!-- <el-upload :on-success="uploadSuccess" :show-file-list="false"-->  
<!-- action="http://localhost:9090/files/upload/"-->  
<!-- class="upload-demo"-->  
<!-- >-->  
<!-- <div class="AvatarDiv">-->  
<!-- <el-avatar icon="UserFilled" style="width: 80px; height: 80px"></el-avatar>-->  
<!-- <img :src="'data:image;base64,' + image" :style="imgDisplay"-->  
<!-- style="width: 80px; height: 80px; border-radius: 40px"/>-->  
<!-- <div class="editImg">-->  
<!-- 更换头像-->  
<!-- <el-icon color="color">-->  
<!-- <edit></edit>-->  
<!-- </el-icon>-->  
<!-- </div>-->  
<!-- </div>-->  
<!-- </el-upload>-->  
<!-- </div>-->  
 <el-descriptions :column="1" :size="large" border style="min-width: 500px" title="">  
 <el-descriptions-item>  
 <template #label>  
 <div>  
 <el-icon>  
 <user/>  
 </el-icon>  
 用户名  
 </div>  
 </template>  
 {{ username }}  
 </el-descriptions-item>  
 <el-descriptions-item>  
 <template #label>  
 <div>  
 <el-icon>  
 <location/>  
 </el-icon>  
 姓名  
 </div>  
 </template>  
 {{ name }}  
 </el-descriptions-item>  
 <el-descriptions-item>  
 <template #label>  
 <div>  
 <el-icon>  
 <tickets/>  
 </el-icon>  
 性别  
 </div>  
 </template>  
 {{ gender }}  
 </el-descriptions-item>  
 <el-descriptions-item>  
 <template #label>  
 <div>  
 <el-icon>  
 <office-building/>  
 </el-icon>  
 年龄  
 </div>  
 </template>  
 {{ age }}  
 </el-descriptions-item>  
 <el-descriptions-item>  
 <template #label>  
 <div>  
 <el-icon>  
 <iphone/>  
 </el-icon>  
 手机号  
 </div>  
 </template>  
 {{ phoneNum }}  
 </el-descriptions-item>  
 <el-descriptions-item>  
 <template #label>  
 <div>  
 <el-icon>  
 <office-building/>  
 </el-icon>  
 邮箱  
 </div>  
 </template>  
 {{ email }}  
 </el-descriptions-item>  
 </el-descriptions>  
 <el-tooltip content="修改信息" placement="bottom">  
 <el-button icon="Edit" size="large" style="margin-top: 30px; width: 80px" type="primary" @click="Edit">  
 </el-button>  
 </el-tooltip>  
 </div>  
 <div style="margin-left: 130px">  
 <img alt="" src="../../public/self\_Space.png"/>  
 </div>  
 </div>  
 <div>  
 <!-- 弹窗-->  
 <el-dialog v-model="dialogVisible" title="操作" width="30%" @close="cancel">  
 <el-form ref="form" :model="form" :rules="rules" label-width="120px">  
 <el-form-item label="账号" prop="username">  
 <el-input v-model="form.username" disabled style="width: 80%"></el-input>  
 </el-form-item>  
 <el-form-item label="密码" prop="password">  
 <el-input v-model="form.password" :disabled="disabled" :show-password="showpassword" show-password  
 style="width: 80%"></el-input>  
 <el-tooltip content="修改密码" placement="right">  
 <el-icon size="large" style="margin-left: 5px; cursor: pointer" @click="EditPass">  
 <edit/>  
 </el-icon>  
 </el-tooltip>  
 </el-form-item>  
 <el-form-item :style="display" label="确认密码" prop="checkPass">  
 <el-input v-model="form.checkPass" show-password style="width: 80%"></el-input>  
 </el-form-item>  
 <el-form-item label="姓名" prop="name">  
 <el-input v-model="form.name" style="width: 80%"></el-input>  
 </el-form-item>  
 <el-form-item label="性别" prop="gender">  
 <el-radio v-model="form.gender" label="男">男</el-radio>  
 <el-radio v-model="form.gender" label="女">女</el-radio>  
 </el-form-item>  
 <el-form-item label="年龄" prop="age">  
 <el-input v-model.number="form.age" style="width: 80%"></el-input>  
 </el-form-item>  
 <el-form-item label="手机号" prop="phoneNum">  
 <el-input v-model.number="form.phoneNum" style="width: 80%"></el-input>  
 </el-form-item>  
 <el-form-item label="邮箱地址" prop="email">  
 <el-input v-model="form.email" style="width: 80%"></el-input>  
 </el-form-item>  
 </el-form>  
 <template #footer>  
 <span class="dialog-footer">  
 <el-button @click="cancel">取 消</el-button>  
 <el-button type="primary" @click="save">确 定</el-button>  
 </span>  
 </template>  
 </el-dialog>  
 </div>  
 </el-card>  
 </div>  
</template>  
<script src="@/assets/js/SelfInfo.js"></script>  
<style scoped>@import '../assets/css/SelfInfo.css';</style>

##### 测试用例





## 5总结

这个项目是从 1.3 考完计算机组成原理开始做的，自己之前做过一个图书管理系统、一个车票管理系统，还有一个智能简历解析的系统。但是前两个项目比较简陋，当时完全是自己设计的，彼时作为一个开发的小白，完全不懂设计模式跟设计理念。这两个项目都是单纯的控制台项目，压根没有前端、后端这一说，数据库也是没有，当时觉得涉及到文件操作都是巨大的进步了。

智能简历解析系统是这学期做的，是我们 C# 的课程设计，这学期学了一门软件工程导论，学了一些开发的理念，加上平时同同学的交流，决定我们小组好好完成这门课设。中间过程是艰辛的，但最后做出来的还算满意，核心的简历解析是调用的API（因为本科生自己做不太可能），界面用 WPF 设计的（为了更美观，我们抛弃了 Winform 组件），还有一些数据库的操作。更重要的是，这个项目用了很多设计理念，也锻炼了自己的沟通能力。

这学期的Web 程序设计课程，课内也做了很多样例，可是 Java 的技术太多太杂，各种名词又不绝于耳，总感觉每次做的样例都是懵懵懂懂，很多东西只是照着报告做一遍，为什么配置 pom.xml，为什么这个代码这么写，这个注解有什么用，都是不太清楚。

所以打算接着这个机会，想要学一下 Java 开发的技术，咨询了一些从事 Java 开发的已经工作的学长和朋友之后，我打算做一个 SpringBoot + Vue 的 JavaWeb 项目，但是迫于时间压力，很多代码和设计的框架参考了很多 Github 上的一些开源项目。做这个项目的话，我选择留在青岛，并没有回家，陆陆续续做到现在算是完成了。

关于自己的一些做项目期间的学习体会和心得，我把这些东西放在了自己的博客 <https://weipengchao.top/> （因为是用 Github 自带的 .github.io搭建的，有时访问需要科学上网）以及Github 的仓库 : <https://github.com/Running-Turtle1/JavaWebLearning>。

最后感谢老师的默默付出与谆谆教诲。