浙江大学城市学院
ZHEJIANG UNIVERSITY CITY COLLEGE

毕业设计(论文)

题　　目　基于 AngularJS 的宿舍管理系统设计与开发

姓　　名　周晢昊

学　　号　31501105

专业班级　计算机 1501 班

所在学院　计算学院

指导教师(职称)　罗荣良(副教授)

二〇一九 年 五 月 十八 日

基于 AngularJS 的宿舍管理系统设计与开发

【摘要】 学生寝室生活是大学生活中至关重要的一部分，是否良好的学生生活甚至影响了学生在大学中的幸福感以及学习态度，通过对宿舍管理系统设计的开发，利用互联网的便利服务于学校师生，能使得我们的学习、生活更加有规律，提高学校的管理水平。在本文介绍的宿舍管理系统中，采取了前后端分离的设计方式，采用了基于 AngularJS 技术的前端设计，通过 AngularJS 实现界面事件的双向绑定，后端则采用了 SpringMVC+jpa 来实现业务逻辑，并实现业务解耦。本文中的宿舍管理系统主要解决了在日常事务中管理者一些人工记录人员的不便问题，更加有利于管理人员的管理，例如可以以 excel 形式对学生信息进行批量导入，管理并修改学生信息。学生可以查看管理员在系统中发布的公告，以及向管理员提交事务申请等操作。

【关键词】 宿舍管理，AngularJS，SpringMVC，前后端分离

Design and Development of Dormitory Management System Based on AngularJS

【**Abstract**】 Student dormitory life is a vital part of College life. Whether good student life even affects students'well-being and learning attitude in college. Through the development of dormitory management system, using the convenience of the Internet to serve school teachers and students, we can make our study and life more regular and improve the management level of the school. In the dormitory management system introduced in this paper, the front-end and back-end are separated. The front-end design based on Angular JS technology is adopted. The two-way binding of interface events is realized through Angular JS. The back-end uses Spring MVC+jpa to realize business logic and realize business decoupling. The dormitory management system in this paper mainly solves the inconvenience of some manual record personnel in the daily affairs of managers, and is more conducive to the management of managers. For example, the student information can be imported in batches in the form of excel, and the student information can be managed and modified. Students can view announcements issued by administrators in the system and submit transaction applications to administrators.

【**Key Words**】 Dormitory Management，AngularJS，SpringMVC，separated front-end and back-end

目录

图目录

表目录

第 1 章 绪论

1.1 研究背景和意义

1.1.1 研究背景

在这个信息化逐渐普及的社会之中，随着目前国民生活水平的不断提高，人们的知识水平不断地提高，大学生的数量也是在逐年上升，而宿舍作为学生在学校生活的第二个家，良好的宿舍环境能更好地帮助学生们的日常生活。目前大部分的宿舍仍然是以人工管理的形式进行管理，一些琐碎的事务会有不便，而人们对于生产力的要求变得越来越高，需要更加高效便捷的方式来应对生活中所要面对的方方面面的问题。一个合理的宿舍管理可以有效帮助宿舍管理人员进行对学生的管理，也有助于学生们在宿舍中的事务处理，不会因消息的延迟传达造成各种不便。

1.1.2 研究意义

在以往之中，很多宿舍的后勤部门采用的都还是比较原始的宿舍管理方式，消息具有滞后性，学生在宿舍的信息往往得不到及时有效的更新，而登记学生入住、更换宿舍、离校等等信息也都是以人工记录的方式进行统计，人工统计的方式麻烦费时并且效率低下，不足以满足实际的需求。学生在于宿舍的事务并不仅仅是居住，管理人员除了对人员进行管理之外，还有收取水电费、日常通知、卫生检查等等，学生需要与管理人员进行直接交涉处理任务，并不方便。

1.1.3 国内外研究现状

学生宿舍管理是大学管理之中很重要的一环，在某种程度对于学校的发展起着至关重要的作用。而随着计算机技术和互联网技术在各个领域的日益渗透，许

多国内外高校都提出了实现数字化校园建设的发展规划[1]。校园信息化的技术概念首次是有一位美国教授 Kenneth C.Green 提出的，并且申请了一份高校信息化管理的研究项目。之后的几年项目的实行得到了大多高校的认可，遍及高校众多领域包括科研教学，课题研究以及信息管理等[2]。美国作为一个具有强大经济实力的国家，是世界上最早开展教育信息化，并且就目前来看也是教育信息化发展水平最高的国家，信息技术的发达带动了企业信息化建设的发展，也间接影响了美国高校教育信息化的进步，不仅有力推动了美国国内教育改革，也为其他国家和地区教育信息化的发展提供方向上的指引和经验上的借鉴[3]。

相比较国内，CERNET 项目正式启动之后，不少高校深受影响开始进行数字化发展教育，建立网上信息教学网，部分大型高校逐渐开始构建各领域的信息化系统包括教学教务系统，办公自动化系统以及学生信息登录系统等，信息化建设不断扩大，许多学校在科技研究，教育教学以及人事管理方面都实现了数字信息化，建立了各自的网络应用 app 系统。而且一些高校从国外引进信息技术在校园局域网的基础上开设了属于该校的宿舍管理系统，通过大型的网站平台为进行信息的共享以及更近，人性化的设计提高了学生对其不同标准的接纳程度。但是仍有很大一部分高校的宿舍管理由于资金的缺乏还采用传统的方式进行，特别是在一些水电费支付上仍使用现金支付，排队支付，这样的人工交付的形式在地方受限的情况下是一种人为地资源浪费，排队时间长，发生错误率高。最后一方面，是系统的多样性较难控制[4]。由于学校的校风不同，信息管理制度下达的规定标准不同，因此开设的宿舍管理系统各有千秋，产生了不同版本的单机版系统引用。为了方便学生进出寝室时接受系统的分配安排，寝室资源的合理安排分配，各大高校的宿舍管理部门应理性地，规范地开发各自的高校宿舍管理系统[5]。

举个例子，从宿舍分配这一功能来看，如果以传统方式手动分配宿舍，日常的个位数修改可能问题不大，但是在新生入学这种时段，手动分配的方法就需要花费大量的人力、物力去将学生、寝室信息进行分类处理，效率低、易出错，在事务繁忙的时期带来麻烦，难以满足目前大学宿舍分配和管理的要求。因此在高校校园信息化建设的背景下，在高校新生的入住系统中增加宿舍安排模块，不仅可以有效提高入住的效率与宿舍分配的质量还可以对高校校园信息化建设起到推动作用[6]。

1.2 研究目的和研究内容

1.2.1 研究目的

本系统开发的最终用户为宿舍管理人员与在宿舍的学生，研究的内容如何以网页端的形式来解决管理人员与学生在宿舍的日常事务处理，功能包括宿舍管理、人员管理、卫生管理、出入人员登记、通知等，尽可能的能在通过网页上处理学生在宿舍的所有事务。

1.2.2 研究内容

本系统主要研究如何在 windows 上部署 springboot java 应用，在前端基于 angularJs 框架完成一个能够满足宿舍管理者与学生在寝室中日常事务需求的宿舍管理系统。

1.2.3 拟解决的关键问题

(1)在宿舍生活的学生在处理日常事务的时候需要与管理人员进行直接交涉，往往会因为个人的因素导致两者之间得不到及时的交流，不仅会给学生，也会给管理人员带来不便。学生处理部分事务的手续显得繁琐，比如一系列申请(在外住宿申请等)，往往一次并不能够解决问题，需要和管理人员进行多次交流，而这过程就显得有些冗长。包括一些琐碎事务，如订水、缴纳水电费、申请维修等，如果能通过网上系统便能减少手续。

(2)传统宿舍管理人员的人工管理并不方便，因受时间、地点及人为等因素制约，流程难以高效流转，对一些陈旧数据的统计困难程度会随时间而逐渐上升。并且管理是需要有制度流程的，当管理人员进行更替时需要对制度流程重新学习，并且大部分宿舍的管理流程是通过经验以及口传而熟悉的，管理实际的运作流程并不如预料中流畅。

第 2 章 系统的开发环境与技术介绍

2.1 MVVM 设计模式

好的设计模式可以将前端的功能模块较好的区分开来，降低各模块之间的耦合度，各个部分有其自己的明确功能，可以各自处理自己的任务，这样每个模块之间的划分清晰，功能明确，就易于前端代码的维护，在之后的代码开发中带来很大的便利。前端常用的设计模式就有 MVC(Model-View-Controller)、MVP(Model-View-Presenter)和 MVVM（Model－View－ViewMode）。

因为 AngularJS 框架与 MVVM 设计模式关系更为密切，因此下文也着重介绍 MVVM 设计模式。MVVM（Model－View－ViewModel），即模型－视图－视图模型。MVVM 是微软 WPF(Windows presentation Foundation)和 Silverlight 架构师 John Gossman 于 2005 年提出的，主要是用于分离应用界面层和业务逻辑层[7]。

MVP（Model-View-Presenter）与 MVVM 模式的不同之处在于，MVVM 模式采用了双向绑定（data－binding）机制，当我们对 View 层中的数据进行更新的时候， ViewModel 也会随之进行改变，最后 Model 层中的数据也会发生变化，反之亦然。基于 AngularJS 的前端框架采用的就是 MVVM 模式。图 2.1、2.2、2.3 分别表示了 MVC、MVP、MVVM，方便进行对比。

Model 指数据访问层，和 MVC(Model-View-Controller)和 MVP 模式中的 Model 是一样的存在，来负责业务逻辑和数据封装。View 层指的是视图界面，负责显示出界面。ViewModel 层获取到模型（Model）层的数据，对数据进行封装处理，然后通知 View 层修改界面进行更新。

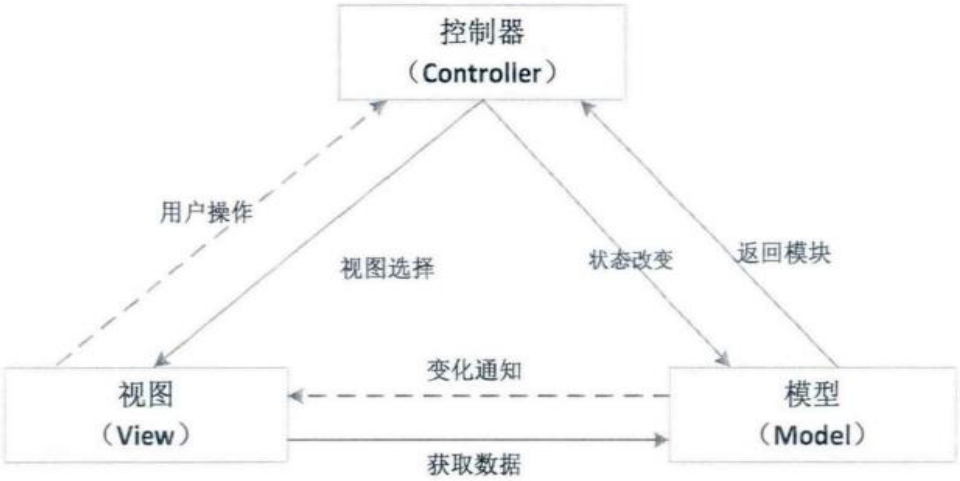

图 2.1 MVC 模式图

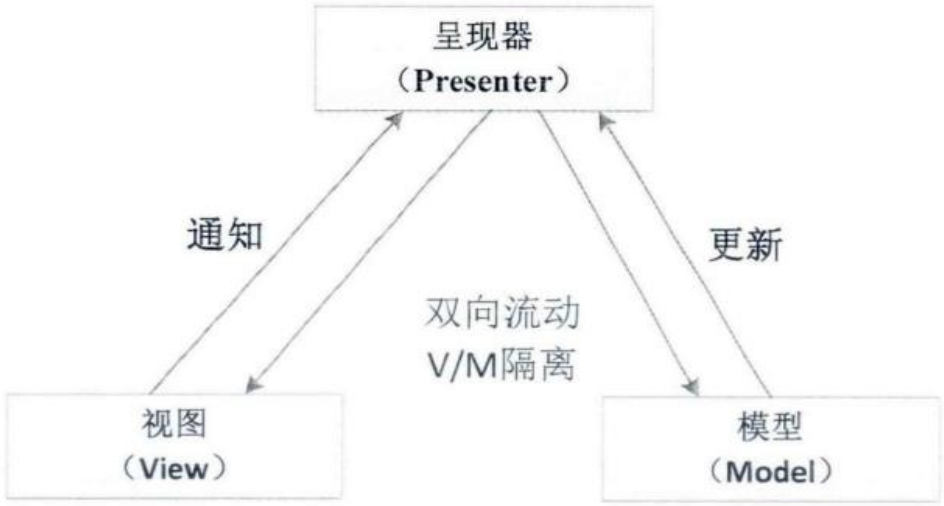

图 2.2 MVP 模式图

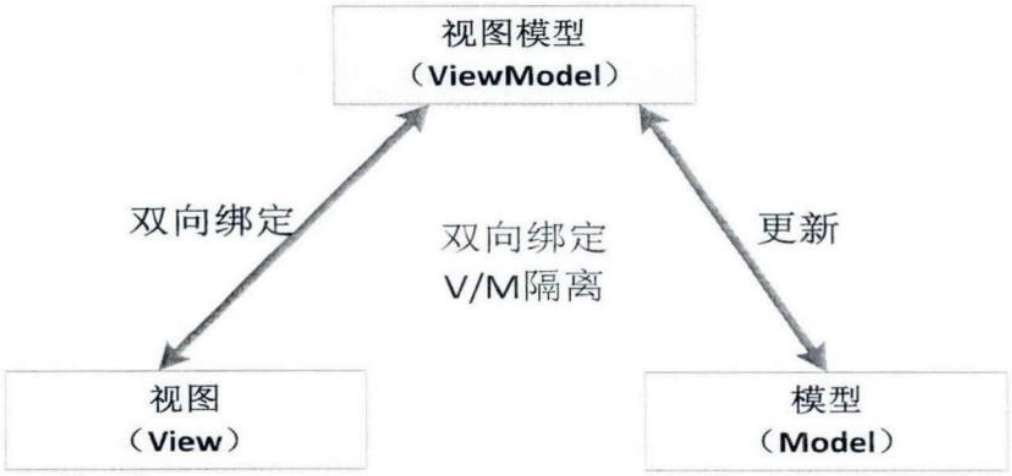

图 2.3 MVVM 模式图

2.2 AngularJS 框架介绍

2.2.1 AngularJS 背景

AngularJS 于 2009 年由 Misko Hevery 等人创建，后为 Google 所收购。AngularJs 是一个 JavaScript 框架，可以扩展应用程序中的 HTML 标签，从而能够在 Web 应用程序中通过 HTML 标签来声明动态内容。Angular 有很多特性，最为核心的是 MVVM 开发模式、模块化，自动化双向数据绑定技术、语义化标签、一览注入、测试驱动开发等等[8]。

2.2.2 选用 AngularJS 理由

在以往的 web 应用开发中，前端页面的呈现都是通过一个个 HTML 界面来实现的，为每一个界面都配置相应的 HTML 界面，但是随着对网页应用的要求越来越高，这样子的方式就显得非常繁琐不便，不能够满足越来越复杂应用的要求，所以现在通常会采用类库和框架来解决这些不足，不再一味构建静态网页，而是根据页面需求搭建对应的动态网页。类库(Class Library)是一个综合性的面向对象的可重用类型集合，这些类型包括：接口、抽象类和具体类。程序开发人员把已

有的开发成果保存在一个类库中，可以被自己的程序或者他人调用，可以避免一些大量重复代码的出现，也可以把类库总结为一个功能模块，用户需要使用这个功能的时候，就可以调用类库，例如 JS 中最常见的 Jquery。框架是一些已经实现了的非常常用的重复性的业务逻辑，用来将程序员从大量重复性的工作中解放出来，如 Angular 等[9]。

2.2.3 AngularJS 特性

因为 Angular 前端开发框架对于数据的处理具有高效性，可以将数据和视图分离开来，提供一个更高层次的抽象来将应用系统的开发进行简化，所以对于 CRUD(增加、查询、更新、删除)应用系统，即管理信息系统的前端页面更加适合，但是对于操作很繁琐，应用复杂的例如游戏或者图像处理类的应用就不合适了[10]。

AngularJS 是一个新出现的强大客户端技术，它利用并且扩展 HTML，CSS 和 JavaScript，弥补了它们的一些非常明显的不足[11]。因此可以使用扩展的 HTML 标签来实现一些动态内容。AngularJS 有五个最重要的功能和特性：数据双向绑定，模板化，MVVM 开发模式，服务和依赖注入机制和指令[12]。

(1)数据双向绑定：单向绑定非常简单，就是把 Model 绑定到 View，当我们用 JavaScript 代码更新 Model 时，View 就会自动更新，传统的网页页面开发过程中，Model 发生改变的时候，程序开发人员需要人工处理 DOM 元素，将改变应用到 View 中，而当 View 发生改变的时候，也需要手工获取变化的数据。而双向绑定和单向绑定就不同，如果 View 在用户操作过程中被更新了，Model 也会随之自动更新，这种情况就是双向绑定，这一大特性就是 AnguarJS 最大的优点，在开发人员编码的过程中，不用再编写大量去同步数据的代码，可以很好地减少在这一方面上的开发时间，从而可以花更多的精力在应用功能的开发上。数据双向绑定机制，同步了 DOM 和 Model，尤其当用户交互比较多，数据操作多的时候，更加反应出其优点[10]。

(2)模板：在 AngularJS 中一个 HTML 文件就是一个模板。HTML 模板将会被浏览器解析到 DOM 中，DOM 会成为 AngularJS 编译器的输入，AngularJS 将会遍历 DOM 模板来生成一些 directive(指令)，所有的指令都负责针对 view 来设置数据绑

定[12]。

(3)MVVM 开发模式：通过上文对 MVVM 模式的分析，MVVM 以 Model 为中心，让 View 层和 Model 层进一步分离解耦，各层次功能明确，各司其职，减少了前端开发工作量，提高了开发效率[10]。

(4)依赖注入机制：依赖注入指的是将服务注入到（映射到）一个对象中，从而成为维护客户端的一部分，AngularJS 中就有许许多多的服务来提供不同的功能，其中包括了 value、factory、provider、constant 等，这些服务可以通过在 controller 的定义时，以参数的形式将其配置进去，Angular 就可以侦测到相应的服务提供给程序。依赖注入机制允许开发人员可以请求需要的依赖，而不是去寻找它们[10]。

(5)指令：在 AngularJS 中可以创建自定义的标签，用来装饰元素或者操作 DOM 属性[12]。在 AngularJS 的指令配置中，可以通过调整 restrict 的值从而在 E（元素 element）A（属性 attribute）C（类名 class）使用指令，在编译 HTML 的在 DOM 元素上附加上我们事前编写好的行为。

2.3 SpringBoot 框架介绍

介绍 spring 框架之前首先要介绍的是 spring 框架，spring 框架是 Rod Johnson 创建的一个为解决企业应用开发复杂性的开源框架，是 java 应用最广的框架之一。Spring MVC 框架的核心是 IOC。IOC（Inversion of control）实现了配置式的对象管理方法，降低了类之间的耦合度，很好的支持了 AOP（Aspect Orient Programming）面向切面编程模式[13]。Dispatch Servlet 负责转发请求给相应的应用程序，处理后返回相应模型和视图.在此模式中，通常采用基于注解的方式，对象和组件间的映射关系都写到配置文件中，这样的方式能对请求和处理进行灵活的匹配[14]。

相对来说，传统的 spring 框架在配置过程中，繁琐的配置文件会给开发者带来不少的麻烦，容易理不清思路，因此，在本系统中使用了 spring boot 框架进行开发。Spring boot 是由 Pivotal 团队提供的全新框架，不同于原本传统方式一个个的配置 XML 文件，它采取了特定的方式来配置，用来简化新 spring 应用的初始搭建以及开发过程，开发人员也不需要获取一个个配置模板文件然后根据自己的项目一步步修改。使用 spring boot 的好处就是简单、快速、方便，以往我们搭建

spring web 项目时，需要配置 web.xml、配置数据库连接，配置 spring 事务、配置加载配置文件的读取，开启注解、配置日志文件等等一系列工作，使用 spring boot 之后只需要简单的几个配置就可以实现搭建。

2.4 Maven 框架介绍

Maven 是一个项目管理工具，它包含了一个项目对象模型(Project Object Model)，一组标准集合，一个项目生命周期(Project Lifecycle)，一个依赖管理系统(Dependency Management System)，和用来运行定义在生命周期阶段(phase)中插件(plugin)目标(goal)的逻辑，使用 Maven 的时候，用一个明确定义的项目对象模型来描述项目，然后 Maven 可以应用横切的逻辑，这些逻辑来自一组共享的（或者自定义的）插件[15]。在多个成员项目开发的过程中，只需要使用相同的 pom.xml 文件，Maven 可以帮助成员们在很短的时间内完成项目环境配置，导入项目所需的各类 jar 包，由于许多项目的设置都比较类似，因此相同的 Maven 配置也可以重复，能够使开发人员配置项目更加地轻松。

2.5 开发工具介绍

2.5.1 Intellij IDEA

Intellij IDEA 是 jetBrains 公司开发的产品，是一款 java 的 IDE (Integrated Development Environment) 。和 Eclipse 一样，IDEA 都是目前 java 开发最常用的工具。IDEA 在业界中受到很多人的喜爱，特别是在智能代码助手、代码自动提示、各类版本控制工具(git、svn 等)、重构、CVS 整合、J2EE 支持、JUnit、代码分析、创新的 GUI 设计等方面的功能都有它独到的优越之处。java 开发者在实际的 java 开发过程中，通过这款 IDE 可以更好地加快编码效率。IDEA 的版本目前有两个，一个是开源免费使用的 Community 版本，本系统使用的就是该版本，该版本已经可以满足大部分 java 开发的需求，而 Ultimate 是免费试用一个月之后进行收费的版本，提供了更多方便的功能，也支持更多格式的文件编辑。

2.5.2 Visual Studio Code

Visual Studio Code 是 Microsoft 发布的跨平台源代码编辑器，主要针对于现代 Web 和云应用的代码编写。该编辑器支持多种格式的文件编辑，包括 java、html、js、json、python 等等，支持语法高亮、智能代码补全、自定义热键、括号匹配、代码片段、代码对比 Diff、GIT 等特性，并针对网页开发和云端应用开发做了优化。在本系统的开发过程中，为了弥补 Intellij IDEA 在对于 web 前端文件的编辑功能的不足，遂采用该编辑器编辑。

第 3 章 系统设计

3.1 需求设计

3.1.1 总体业务需求

本宿舍管理系统的主要功能是给宿舍管理人员有一个好的平台更好的对寝室进行管理，进行对寝室学生的分配调度，并且在全寝室公布通知。

系统的主要用户有两种：系统管理员与普通的学生用户。系统管理员负责对于寝室房间以及寝室学生的管理，可以对学生可修改信息（变更寝室、联系方式等易变信息）进行修改，同时可以发布寝室通知。而普通的学生用户可以对自己的信息进行查看修改，查看寝室通知。

从功能角度划分，本系统的功能包括:

（1）寝室房间管理：学生和管理员用户都可以通过该模块了解到各寝室内的人员情况，学生可以参考其中信息以及根据自身情况进行申请换寝室的操作，管理员也可以了解到各寝室的人员情况，方便管理。管理员用户还可以进行添加和删除寝室房间的操作

（2）学生管理：该模块的功能主要针对于管理员用户设计，给学生用户所提供的接口仅有查看寝室楼内的学生信息，呈现的是学生的不重要信息列表，无其余操作，管理员可以进一步查看学生的详细信息并对其进行修改，同时在新生入住的时候可以进行人员的导入操作，支持单个导入，也支持以.xls 为后缀的 excel 文件的学生列表导入（需要按照规定的格式填写），在批量导入信息的同时系统也会进行寝室的自动分配。

（3）公告管理：管理员可以进行寝室公告的发布与删除，学生可以通过改接口查看到管理员发布的公告。

（4）权限登录管理：鉴于本系统应用于学校的宿舍管理，因此在系统的登录模块中并不提供注册接口，当管理员用户在导入新生数据的情况下，每添加一个学生

就会为该学生自动创建以该学生学号为账号，学号后 6 位为密码的账号。在登录之后，用户也可以自行修改密码。

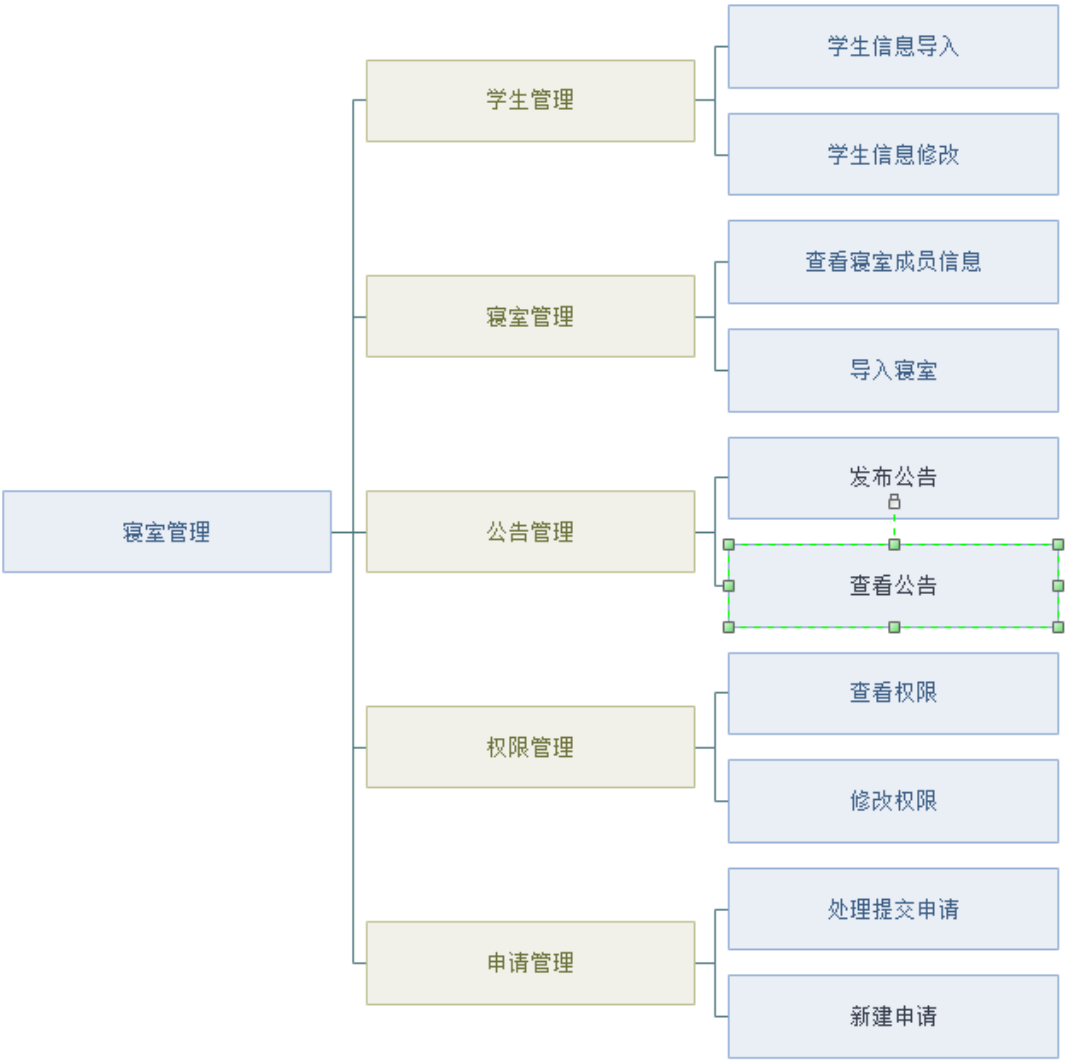

图 3.1 系统框架图

3.1.2 系统非功能性需求

3.1.2.1 系统界面需求

作为一个 web 应用系统，在网页上的界面要求界面颜色风格统一，为了使得界面柔和一点，界面大部分采取了暗色调。因为本系统的功能环绕着管理这一主题功能，所以各项条目信息需要以有条理的形式表现出来，例如表格、图表等形

式。各项操作不应太过繁琐，能够通过简单的按钮达成目的功能。

3.1.2.2 系统性能需求

根据初步的需求分析，宿舍管理系统的性能需求主要有以下几点：

（1）可靠性要求

宿舍管理系统针对管理人员和宿舍学生们设计的系统，要求性能高，实效性强，在可靠性的要求上可能比较高。要求在长时间内所有事务功能够正确地完成，系统也需要24小时长时间工作运行。

（2）安全性要求

宿舍管理系统需要保证用户的信息数据不被窃取，避免遭受非法攻击，泄露个人信息。系统要求有明确的权限设置管理，非admin用户不允许调用响应接口，在前端和后端都需要对其接口有所限制。

（3）可维护性及可适应性

目前来说，所有的网上管理系统在实际运行当中都会出现各种各样的问题，因此无论如何都还需要不断地改进与维护，结构条理不清晰的系统在日后的维护上需要浪费许多的人力物力以及时间，所以系统在设计之初就必须要设计的易于维护和扩展，因此该系统要有很强的可操作性和适应性。

（4）可扩展性

一个系统的可扩展性，是指该系统适应变化的能力。本系统也需要一定的可拓展性。当例如新生入住这种特殊时期，系统的用户量的变化量是非常大的，在短期内的访问量也是很高的，因此，系统必须在这种时期能够适应满足用户们的需求，不能够因为特殊情况而导致系统的正常功能不能正常使用。这一方面的性能需求可以从硬件、软件两方面进行考虑。系统的硬件方面，可以从CPU、内存、网络带宽等角度对系统进行扩展和升级。而在软件的方面，在事务代码的编写上必须做好模块的划分，这样子在系统部分功能需要改变的情况下，不必大量地修改相关代码，只需要小的改动就可以满足需求。

（5）响应速度

系统各个API请求的平均响应时间应低于1s, web首页打开速度在5s以下，web登陆速度在15s以下。

3.2 数据库设计

本系统采用的数据库为 MySQL，MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，是目前来说非常常用的数据库之一。许多全球规模最大，发展最快的组织，包括 Facebook，Google，Adobe，Alcatel Lucent 和 Zappos，都依靠 MySQL 来节省时间和金钱，为其高容量网站，关键业务系统和套装软件提供支持。

在本系统中数据库总共定义了 8 个实体对象：

（1）寝室实体（dorm），用来保存寝室信息。其中包括了寝室号（id），寝室容量（volume）。

（2）通知实体（notice），用来保存通知的所有信息，包括 id，标题（title），发布日期（date），内容（content）。

（3）操作日志实体（operation_log），用来记录用户登录的信息，包括 id,操作者 id(manipulator_id)，操作类型（operation_type），操作详细信息（operation_detail），操作时间(operation_time)。

（4）学生实体(permission)，用来记录所有学生的信息，包括学号(id)，学生姓名（name），寝室号（dorm），分院(branch)，电话(tel)，邮件(email)，专业班级(class)，床位(dorm_pos)

（5）申请实体（applicaton），用来记录所有申请记录的信息，包括申请 id（id），申请日期（date），主题（subject），申请学生学号（student_id），邮箱（email），申请类型（type），申请优先级（priority），申请具体内容（content），申请所处状态（status）。

其余三个实体，用户（user）、角色(role)、权限(permission)记录相互之间的关系，其中包括了各自的 id 以及相互之间的外键联系。

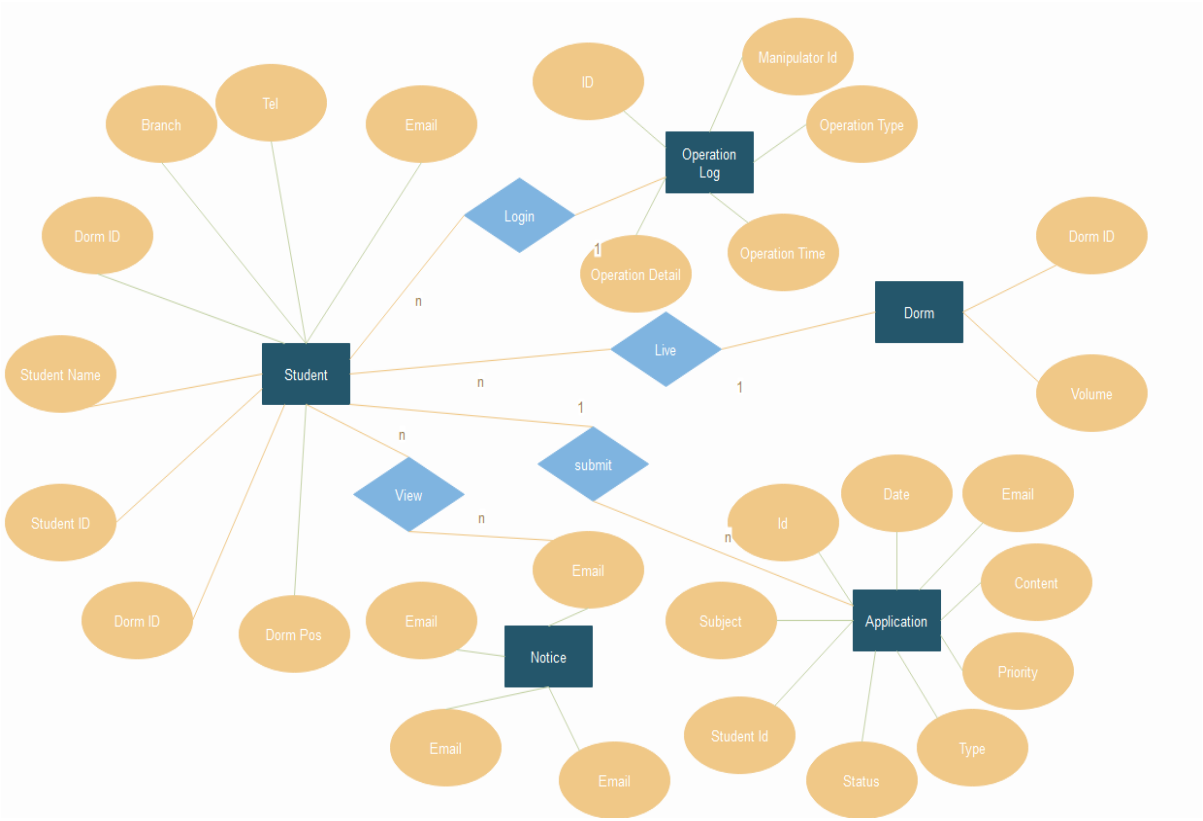
Manipulator Id
ID
Branch
Tel
Email
Operation Type
Operation Log
Login
Dorm ID
Operation Time
Operation Detail
Dorm ID
Dorm
Live
Student Name
Student
Volume
submit
Student ID
View
Date
Email
Id
Email
Content
Dorm ID
Dorm Pos
Email
Subject
Application
Notice
Priority
Email
Student Id
Type
Email
Status
1
n
n
1
n
1
n
n

图 3.2 数据库 ER 图

第 4 章 系统实现

4.1 用户登录模块

用户登录模块的系统流程图如图 3-2 所示：

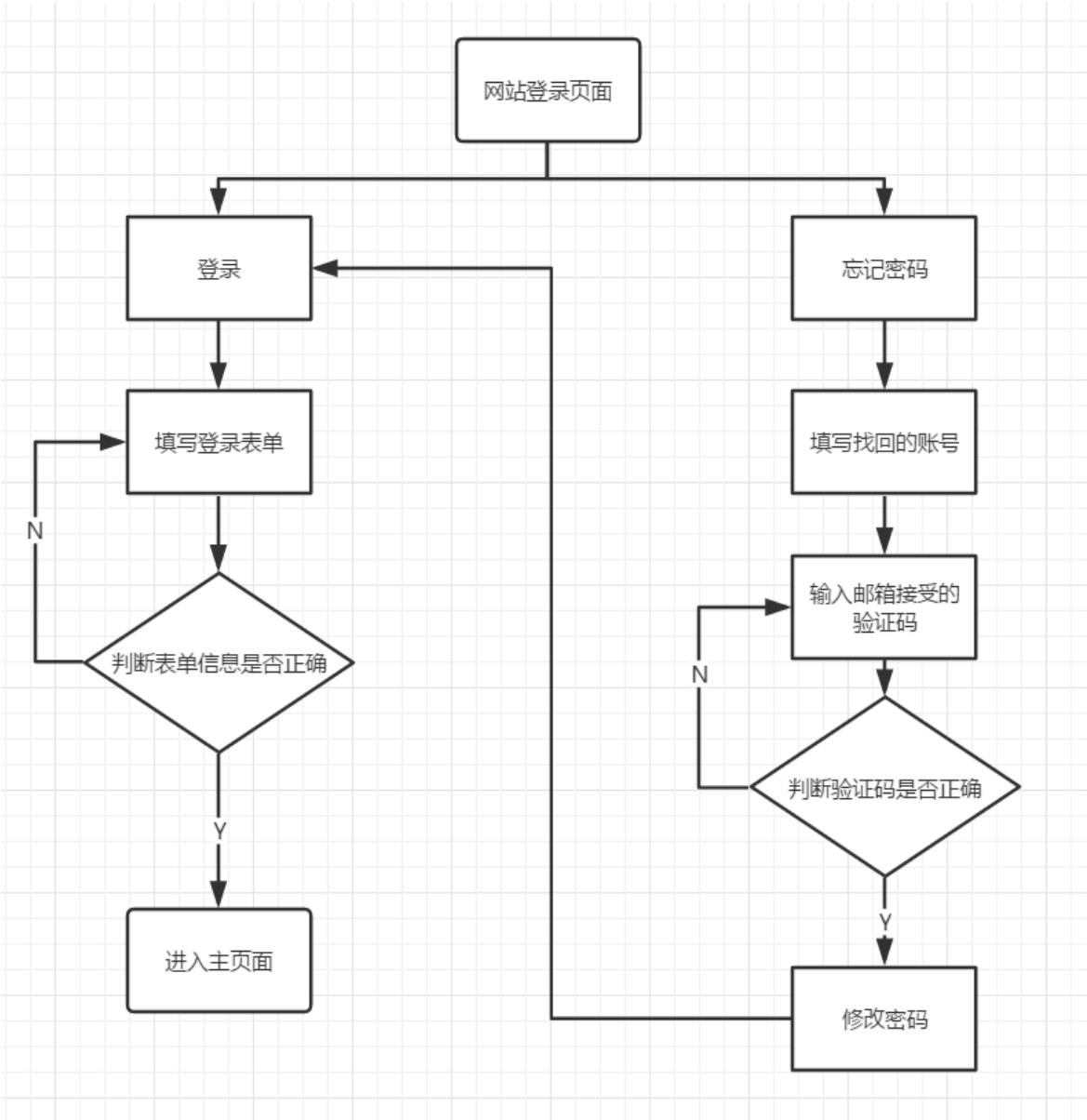

图 4.1 用户登录流程图

4.1.1 用户登陆模块

用户登录模块的核心类为 User 类、Role 类和 Permission 类，其中包含了用户名（字段 username），密码（字段 password），激活码（字段 activation_code）等信息。在 JAVA 的 Spring 框架下，利用了@OneToMany 和@ManyToOne 注解将上述三类对象映射到数据库，使得互相之间建立起外键，将一对多的关系绑定到对象中，省去了一些查询，User 类中自动加载了对应数据库中 role 表中 user_id 属性的数据，Role 和 Permission 也同样。为之后系统中的权限控制提供了便利的接口。

计算机的安全问题现在已经是屡见不鲜。蠕虫、黑客、木马、病毒、拒绝服务攻击等术语在人们面前也好像不再陌生了，每个计算机用户都遇到过这些麻烦的事情，因为加密工作的不到位，受到攻击之后，丢失了重要数据，系统遭到破坏，用户信息被窃取，从而造成了各式各样的损失，因此安全问题也是不得不注意的。在网站登录模块中最重要的安全问题自然就是数据传输中的加密问题了。

在数据前端与后端的传输过程之中，如果使用明文传输数据容易被拦截，从而获得用户信息，因此前端在传输过程中首先对用户信息进行了 RSA 加密。

RSA 加密算法是一种非对称加密算法。在公开密钥加密和电子商业中 RSA 被广泛使用。RSA 的安全性依赖于大数分解，对极大整数做因数分解的难度决定了 RSA 算法的可靠性，到目前为止，世界上还没有任何可靠的攻击 RSA 算法的方式。只要其钥匙的长度足够长，用 RSA 加密的信息实际上是不能被解破的[16]。

表 4.1 RSA 算法简要流程

步骤	说明	描述	备注
1	找出质数	P、Q	-
2	计算公共模数	$N = P * Q$	-
3	欧拉函数	$\varphi(N) = (P-1)*(Q-1)$	-
4	计算公钥 E	$1 < E < \varphi(N)$	E 的取值必须是整数 E 和$\varphi(N)$必须是互质数
5	计算私钥 D	$E * D \,\%\, \varphi(N) = 1$	-
6	加密	$C = M^E \bmod N$	C:密文 M:明文
7	解密	$M = C^D \bmod N$	C:密文 M:明文

RSA 算法需要生成一对 RSA 密钥，其中一个为私钥，由后端保管而不公开，同时由后端生成公钥，在用户进入登录界面时通过 Ajax 操作访问/login/key 传递到

前端，对表单中的数据进行 RSA 加密后再进行传输，这就比较安全的保证了用户信息的安全。

这是一段生成公钥的代码：

```
public static String generateBase64PublicKey() {
    PublicKey publicKey = keyPair.getPublic();
    return new String(Base64.encodeBase64(publicKey.getEncoded()));
}
```

而下面这段代码则在获取到加密后的数据流之后，依据服务端保存的私钥对数据流进行解密，最终获取到了原数据。

```
private static byte[] decrypt(byte[] byteArray) {
    try {
        Provider provider   =new
org.bouncycastle.jce.provider.BouncyCastleProvider();
        Security.addProvider(provider);
        //Cipher: 提供加密和解密功能的实例
        //transformation: "algorithm/mode/padding"
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding",
provider);
        PrivateKey privateKey = keyPair.getPrivate();
        //初始化
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] plainText = cipher.doFinal(byteArray);
        return plainText;
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

4.1.2 忘记密码模块

登录界面上还提供了忘记密码接口。在所有学生新入住的时候，账号密码都初始化为其的学号，学生需要登录进系统手动对密码进行修改。如果在之后忘记

了密码的话，学生就可以从登录界面上的忘记密码接口来找回密码。

用户需要填写需要找回密码的用户名，然后点击获取验证码按钮，为了防止恶意的对找回密码的频繁操作，在这里给按钮设置了 60s 的间隔时间，在一次点击之后需要 60s 才能再次点击，在点击之后，java 后台会以 SMTP（Simple Mail Transfer Protocol）协议的方式给和学生们绑定的学校邮箱发送验证码邮件，在收到邮件之后在忘记密码界面上输入验证码即可重置密码（验证码仅在 30min 以内有效）。下面的代码就主要实现了后端是如何发送给目标邮箱验证码邮件，针对邮件发送过程中返回的各种情况也都进行了处理。

```
private static boolean checkEmail(String email) {
        String host;
        String hostName = email.split("@")[1];
        Record[] result;
        SMTPClient client = new SMTPClient();
        try {
            // 查找 DNS 缓存服务器上为 MX 类型的缓存域名信息
            Lookup lookup = new Lookup(hostName, Type.MX);
            lookup.run();
            if (lookup.getResult() != Lookup.SUCCESSFUL) {// 查找失败
                logger.error("邮箱（" + email + "）校验未通过，未找到对应的 MX 记录!");
                return false;
            } else {// 查找成功
                result = lookup.getAnswers();
            }
            // 尝试和 SMTP 邮箱服务器建立 Socket 连接
            for (Record aResult : result) {
                host = aResult.getAdditionalName().toString();
                logger.info("SMTPClient try connect to host:" + host);

                // 尝试 Socket 连接到 SMTP 服务器
                client.connect(host);
```

```
                    // 查看响应码是否正常
                    // 所有以 2 开头的响应码都是正常的响应
                    if (!SMTPReply.isPositiveCompletion(client.getReplyCode()))
{
                        // 断开 socket 连接
                        client.disconnect();
                    } else {
                        logger.info("找到 MX 记录:" + hostName);
                        logger.info("建立链接成功：" + hostName);
                        break;
                    }
                }
                logger.info("SMTPClient ReplyString:" + client.getReplyString());
                String emailSuffix = "163.com";
                String emailPrefix = "15382327056";
                String fromEmail = emailPrefix + "@" + emailSuffix;
                // 尝试和 SMTP 服务器建立连接,发送一条消息给 SMTP 服务器
                client.login(emailPrefix);
                logger.info("SMTPClient login:" + emailPrefix + "...");
                logger.info("SMTPClient ReplyString:" + client.getReplyString());

                // 设置发送者，在设置接受者之前必须要先设置发送者
                client.setSender(fromEmail);
                logger.info("设置发送者 :" + fromEmail);
                logger.info("SMTPClient ReplyString:" + client.getReplyString());

                // 设置接收者,在设置接受者必须先设置发送者，否则 SMTP 服
务器会拒绝你的命令
                client.addRecipient(email);
                logger.info("设置接收者:" + email);
                logger.info("SMTPClient ReplyString:" + client.getReplyString());
```

```
                logger.info("SMTPClient ReplyCode：" + client.getReplyCode() +
"(250 表示正常)");
                if (250 == client.getReplyCode()) {
                    return true;
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                try {
                    client.disconnect();
                } catch (IOException e) {
                    logger.error(e);
                }
            }
            return false;
        }
```

4.2 宿舍管理模块

宿舍管理也是本系统的主要功能模块之一。其中以管理员端的功能为主，有添加寝室、删除寝室,还有对寝室学生们的信息查看、修改，对学生进行导入等功能。

本模块中的主要重点在于对于寝室人员的导入以及在导入动作之后对各个人员的寝室自动分配算法设计。

Microsoft Excel 是 Microsoft的一款电子表格软件。目前来说电子表格形式大部分都是以 excel 形式存在，因此在导入学生个人信息的时候，我们接受文档的形式也是 excel 的形式。为处理这种文件形式，在 maven 中导入了 org.apache.poi 包。为此编写工具类 DocUtil 来对 excel 进行读写。遗憾的是，目前该 jar 包暂时只支持.xls（Microsoft Excel 97-03 文件）文件流的读取，当面临目前从 office 2007 开始使用的 .xlsx 来说则需要通过转换成 .xls 文件之后才能进行操作，目前来说还需要改进。

4.2.1 寝室房间管理模块

在寝室房间管理模块上可以进行的操作有对寝室的添加、删除、批量导入、查看寝室人员的操作。寝室的导入较为简单，并没有什么约束条件，只需要提供符合标准的 excel 文件进行导入即可。如果需要查看寝室学生，可以点击红色的寝室现有人数，从而进一步查看寝室内的宿舍学生及床位，再进一步点击学生时，可以跳转到学生管理界面查看对应学生信息。下面是寝室管理的增删操作代码：

```
$scope.dormDelete = function (id) {
                $http({
                    method: 'GET',
                    url: '/dorm/dormDelete',
                    params: {
                        'id': id
                    }
                }).success(function (data) {
                    if (data.message === 'S') {
                        initTable.ajax.reload();
                    } else {
                        $scope.onModel.modelShow('error', data.data);
                    }
                }).error(function (data) {
                    $scope.onModel.modelShow('error');
                });
            }
    $scope.addDorm = function () {
                var modal = $modal.open({
                    backdrop: 'static',
                    templateUrl:  'dorm-module/dorm.add.html',  //script 标签中定义的 id
                    controller: 'dormAddCtrl', //modal 对应的 Controller
                    size: 'md'
                });
```

```
                modal.result.then(function (result) {
                    if (result == 'S') {
                        $scope.onModel.modelShow('success', '添加成功');
                        $timeout(function () {
                            $state.reload();
                        }, 1500)
                    } else {
                        $scope.onModel.modelShow('error', result);
                        //                        $timeout(function() {
                        //                            $state.reload();
                        //                        },1500)
                    }
                }, function (reason) {
                    //                $state.reload();
                })
            }
```

点击增加寝室之后，出现的模态框可以填写需要增加寝室的具体信息，在填写完毕之后提交就可以完成添加寝室的工作。

寝室列表的呈现采用了表格形式，整合了 DataTables 插件。在之后的列表呈现也是采取了相同的方式。控制表格的初始化代码如下：

```
var initTable = $("#tableEmailsList").DataTable({
    "dom": '<"listtable"fit>pl',
    "responsive": true,
    "oLanguage": {
        "sEmptyTable": "没有记录",
        "sInfo": "共有 _TOTAL_ 项，正在展示第 _START_ 到 _END_ 项",
        "sInfoEmpty": "无记录",
        "sInfoFiltered": "(从 _MAX_ 条中筛选)",
        "sInfoPostFix": "",
        "sInfoThousands": ",",
        "sLengthMenu": "每页显示 _MENU_ 条",
```

```
        "sLoadingRecords": "加载中...",
        "sProcessing": "处理中...",
        "sSearch": "",
        "sZeroRecords": "没有记录",
        "oPaginate": {
            "sFirst": "最初的",
            "sLast": "最后的",
            "sNext": "下一页",
            "sPrevious": "上一页"
        }
    },
    "pageLength": 10,
    "order": [
        [0, "asc"]
    ],
    "lengthMenu": [
        [10, 25, 50, -1],
        [10, 25, 50, "所有"]
    ],
    "aoColumnDefs": [{
        "bSortable": false,
        "aTargets": [-1]
    }],
    "stateSave": true,
    "ajax": "/dorm/dormList",
    "columns": [{
            "data": "id"
        },
        {
            "data": null,
            "render": function (data, type, row, meta) {
                var html = data['volume'] - data['remain'];
```

```
                    html = "<a href='#' ng-click='openDorm($event)'>" + html + "
</a>"
                    return html;
                },
                "fnCreatedCell": function (td, cellData, rowData, row, col) {
                    $compile(td)($scope);
                }
            },
            {
                "data": "remain"
            },
            {
                "data": null,
                "render": function (data, type, row, meta) {
                    if ($rootScope.isAdmin) {
                        var html = "<button class='btn btn-success' ng-click =
dormDelete(" + data['id'] + ") ng-show = 'isAdmin'>删除</button>"
                        return html;
                    } else {
                        return "";
                    }
                },
                "fnCreatedCell": function (td, cellData, rowData, row, col) {
                    $compile(td)($scope);
                }
            }
        ],
        "rowCallback": function (row, data) {
        }
    });
```

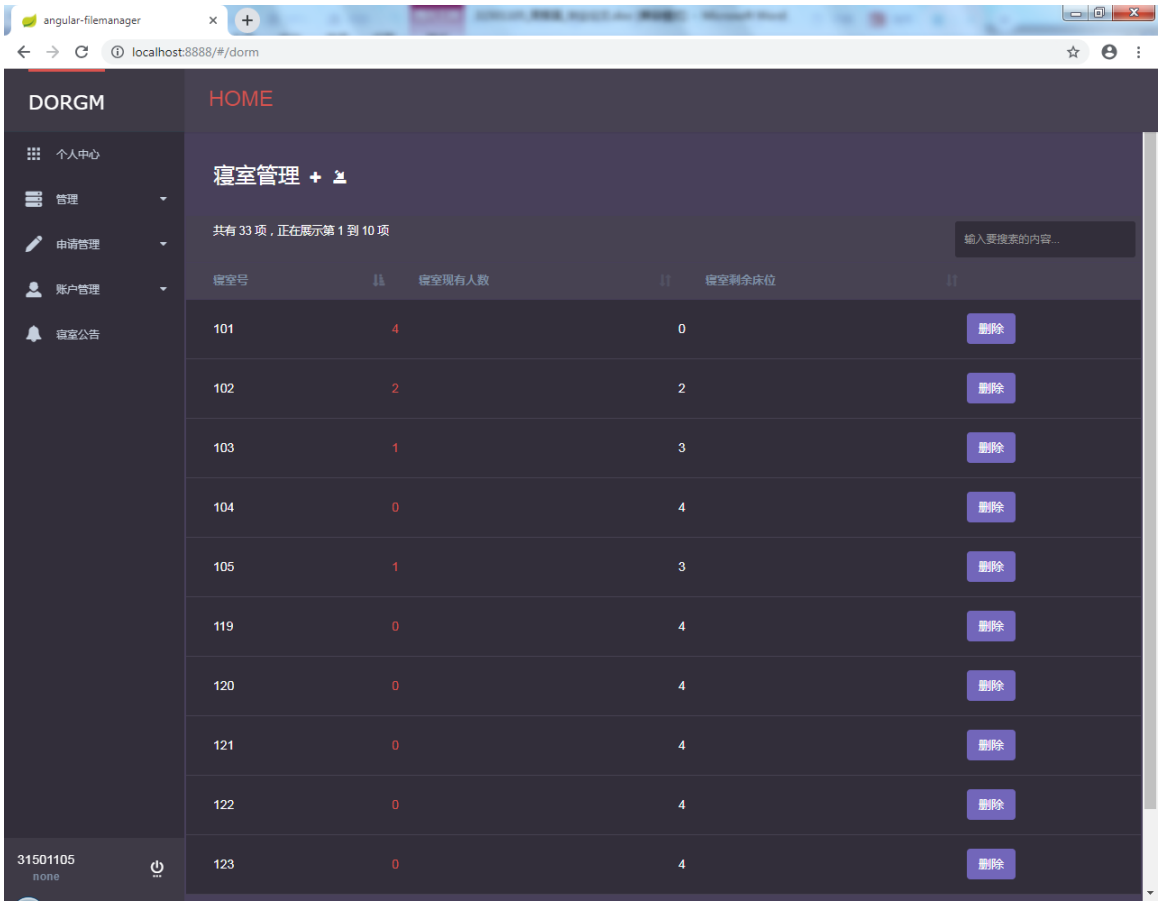

图 4.2 寝室房间管理模块主界面

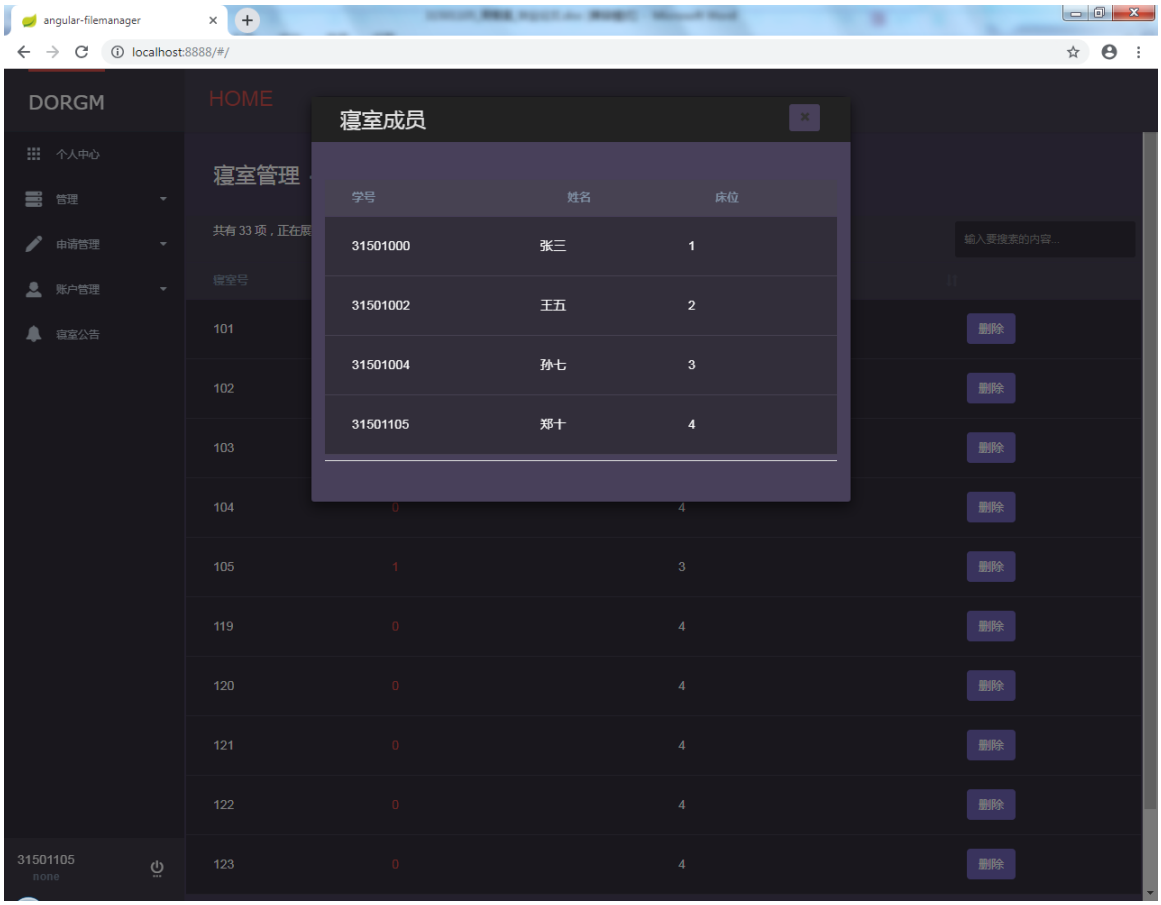

图 4.3 寝室房间管理模块查看寝室人员界面

4.2.2 学生管理模块

具体的宿舍管理的事务处理逻辑代码写在了 DormController 和 DormService 类中，鉴于在实际情况中，录入大量学生的情况下进行人工寝室分配的情况下一般都是按线性分配，这就要求提供有规则（比如连续学号）的学生名单才能够使得最终分配的寝室之中的学生有概率相识。经问卷调查结果得出学生们相比于和不同分院专业的同学做室友更倾向于和学科专业和自己有联系的同学做室友。因此我们在分配寝室的决策上采取了一个较为简单的算法。

在每次添加一名学生进入寝室的时候，会遍历每一个寝室，与其中的已有学生进行比较。在本系统中，选择寝室的优先级为寝室内有同班级同学>寝室内有类似专业班级（如计算机 1501 与计算机 1502）的同学>空寝室>寝室内无类似专业班级的同学。因为问卷内没有更为详细的条目，在这里我们认为寝室内有多名同班

级同学与寝室内只有一名同班级同学并无差别。优先度的计算我们通过计算来衡量。Levenshtein 距离，又称编辑距离，首先由俄国科学家 Levenshtein 提出，所以又叫 Levenshtein Distance，指的是两个字符串之间，由一个转换成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。相应的计算放在了工具类 EditorDistance 中。经过此类比较之后自动将学生一个个安排到寝室当中。导入学生的伪代码如下：

```
public boolean importStudents(File file) {
    //检查 excel 文件是否含有所需各字段
    if(checkExcelPattern(file)){
        //读取学生信息
        List<Student> students = readExcel(file);
        List<Student> failed = new ArrayList<>();
        for(Student student:students){
            //对单个学生进行自动分配
            if(!addStudent(student)){
                failed.add(student);
            }
        }
        //生成导入失败学生名单
        exportFailed(failed);
    }
    else{
        return false;
    }
}
```

具体的界面如图 4.2 所示。

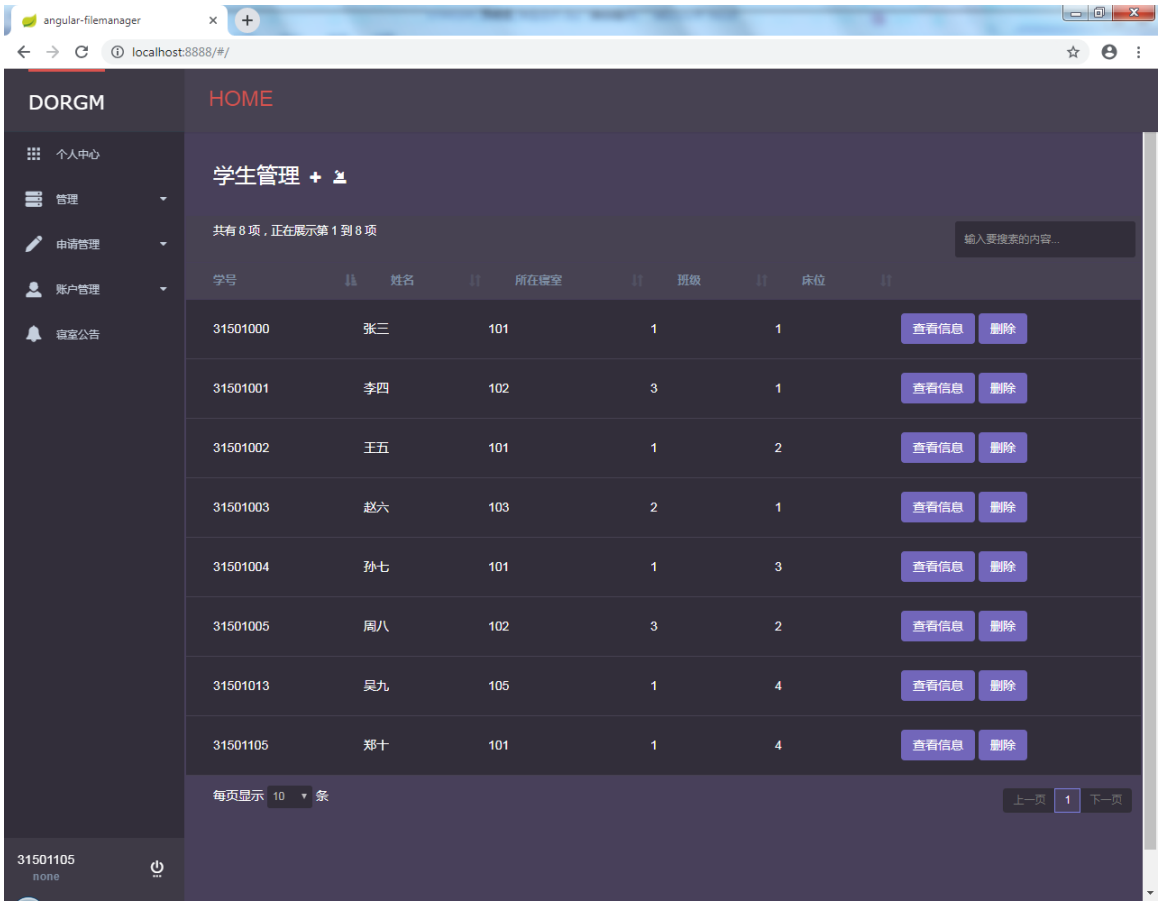

图 4.4 学生管理模块主界面

4.3 学生申请模块

在以往宿舍的日常事务之中，学生有事寻找宿舍管理人员却无果的情况下常常存在，给管理人员与宿舍中的学生们之间建立一个方便的联系方法也是必要的。本系统也实现了这么一个接口。学生可以于本系统中根据自身的情况填写申请(Application 对象)，选择自己提交的申请事务类型之后也可以填写更为详细的信息方便管理人员处理。申请提交到后台之后，管理人员可以通过其相关接口一览所有申请，可以根据其优先度或者类型选择进行处理，提交上来的申请的状态有四种：待处理、处理中、完成、拒绝（拒绝状态的存在主要针对于一些重要的事务，或者是在网上无法简单操作的事务，例如非学期更替期间的换寝、有例如在外住宿的申请，这些类事务往往需要和管理人员进行面对面的交流才能够得到很好地解决）。在申请刚被提交的时候，申请的状态为待处理，管理员可选择的操作有

处理或拒绝，在选择处理操作之后状态变为处理中，管理员可选择的操作有完成或拒绝，在选择完成之后申请的状态变为完成状态。无论在哪个步骤中选择了拒绝后状态都会变为已被拒绝，在学生端同样可以查看到自己所提交过的申请状态。该部分模块的代码主要在 application.controller.js 和 ApplicationController 中定义。提交申请界面如图 4.4 所示。在点击提交之后，表格提交的代码如下：

```
$scope.submitForm = function () {
    $scope.onModel.modelLoading('loading', '提交中');
    $http({
        method: 'POST',
        url: '/application/applicationAdd',
        data: $.param($scope.application),
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded'
        }
    }).success(function (data) {
        if (data.message === 'S') {
            $scope.onModel.modelShow('success', '提交成功');
            $timeout(function () {
                $state.go("application/list");
            }, 2000)
        } else {
            $scope.onModel.modelShow('error', '提交失败');
        }
    }).error(function (data) {
        $scope.onModel.modelShow('error', '提交失败');
    });
}
```

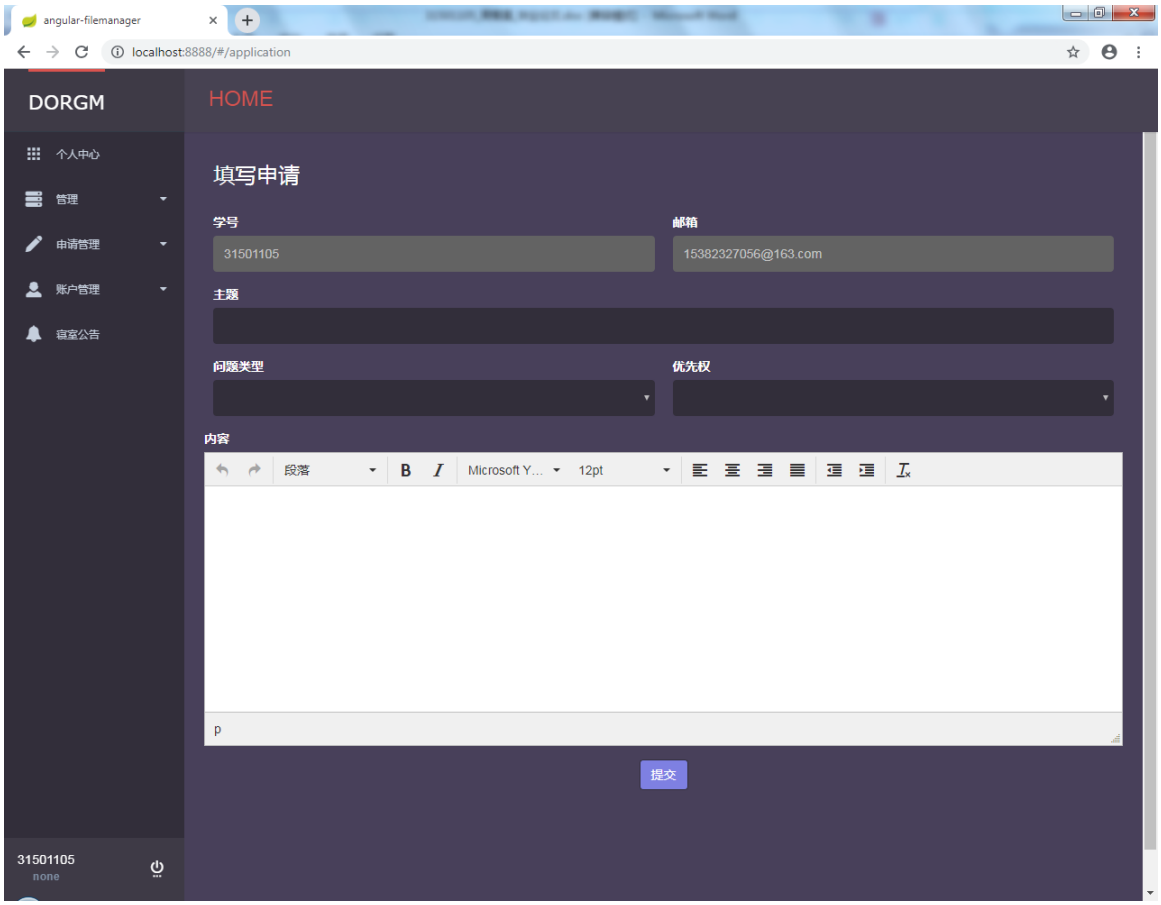

图 4.5 提交申请界面

4.4 公告模块

公告模块有两个界面，一个界面以时间为顺序，用来表示公告一览，显示了所有发布的公告。学生点击公告主题之后，可以查看到公告的具体内容。而针对管理员则额外提供了添加公告和删除公告的接口。添加公告的界面如图 4.5 所示。

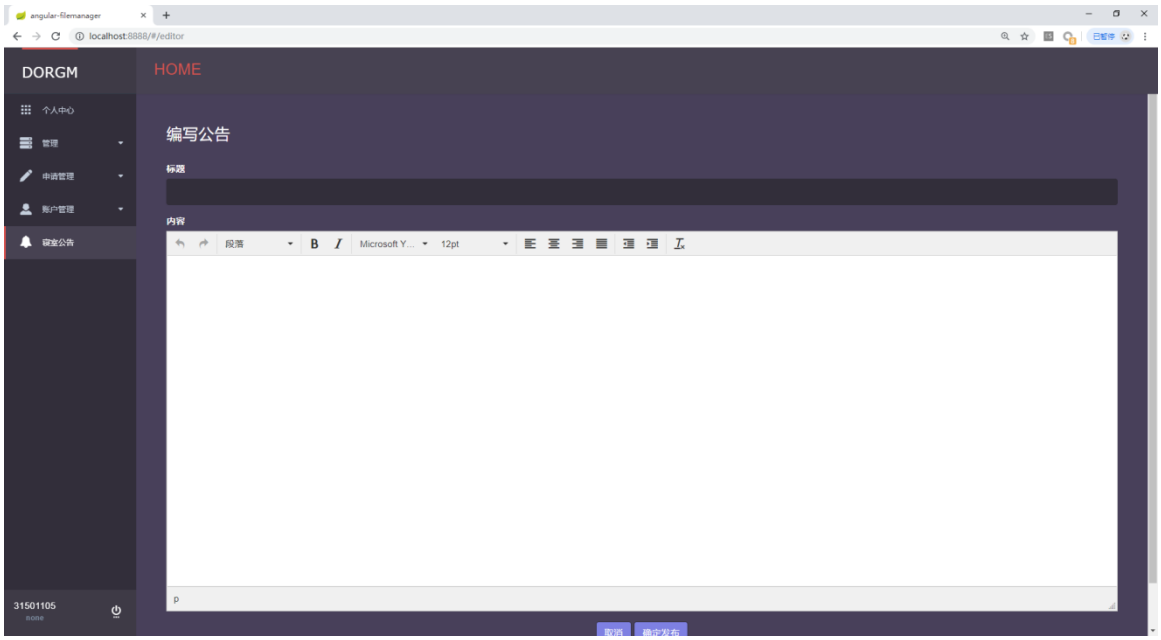

图 4.6 编写公告界面

其中用来编辑公告的文本编辑器整合了 angular 和 tinymce。Tinymce 是一个轻量级的在线文本编辑器，部署起来较为简便，能很好地和 angularJs 进行整合，tinymce 在 JS 文件中进行配置。配置代码具体如下：

```
$scope.tinymceOptions = {
        menubar: false,
        height: 500,
        plugins: "nonbreaking",
        nonbreaking_force_tab: true,
        toolbar: 'undo redo | formatselect | bold italic backcolor | fontselect fontsizeselect | alignleft aligncenter alignright alignjustify | bullist numlist outdent indent | removeformat',
        language: 'zh_CN',
        content_css: [
            '//fonts.googleapis.com/css?family=Lato:300,300i,400,400i',
            '//www.tiny.cloud/css/codepen.min.css'
        ]
```

其中 menubar 控制是否出现菜单栏，height 控制编辑框的高度，plugins 导入的是 tinymce 的一些选项插件，而上述配置中的 nobreaking 的导入是为了防止在网

页中操作 tab 键使得切换到另一个输入框，以缩进来替代，toolbar 则选择一些编辑时的功能，language 决定了 tinymce 的语言，content_css 则控制一些内在元素的 css。因为在文本编辑中所保存的并不仅仅是公告的文字内容，在其中进行公告的排版之后保存的是页面的 html 内容，因此在完成新建公告之后在公告一览页面中进行对页面的具体内容进行查看时，之前的排版格式能够得到很好的还原，能够较好地满足管理者在编辑公告时的需求。

第 5 章 系统测试

5.1 功能测试

java 后台的测试采取了 swagger 工具，swagger 工具配置在 spring boot 框架中。我们可以使用 swagger 注解在 controller 上标记注解可以用来描述各 api，在启动项目之后访问地址 http://localhost:8888/swagger-ui.html，之后就可以通过网页上的可视化界面对各 api 进行测试。页面如图 5.1 所示。

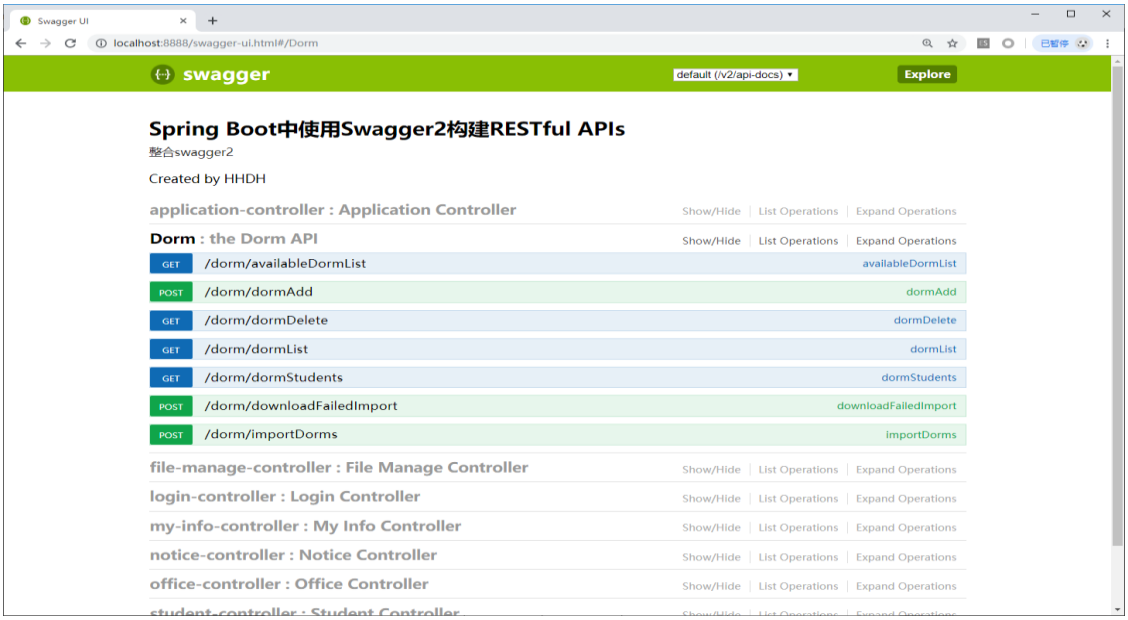

图 5.1 swagger 测试界面

报告见表 5.1：

表 5.1 功能测试报告

模块	描述	预期测试结果	实际测试结果
登录模块	输入正确的账号密码	返回 code “S”，表示登录成功	返回 code “S”，表示登录成功
	输入错误的账号密码	返回 code “F”，data 里包含失败信息	返回 code “F”，data 里包含失败信息
寝室模块	调用 dormList 接口	返回所有寝室列表	返回所有寝室列表
	调用 availableDormList 接口	返回所有有空位的寝室列表	返回所有有空位的寝室列表
	输入寝室 id 调用增删改查接口	返回具体操作结果集	返回具体操作结果集
	输入一个文件调用 importDorms 接口	符合文件要求的软件返回返回 code “S”，表示登录成功，不符合的返回 code“F”,data 里包含失败信息	符合文件要求的软件返回返回 code “S”，表示登录成功，不符合的返回 code“F”,data 里包含失败信息
	调用 studentList 接口	返回所有学生列表	返回所有学生列表
	输入学生 id 调用增删改查接口	返回具体操作结果集	返回具体操作结果集
申请模块	调用 applicationList 接口	返回所有申请列表	返回所有申请列表
	输入申请 id 调用增删改查接口	返回具体操作结果集	返回具体操作结果集
公告模块	调用 noticeList 接口	返回所有公告列表	返回所有公告列表
	输入申请 id 调用增删改查接口	返回具体操作结果集	返回具体操作结果集

5.2 性能测试

系统的性能测试主要对各事务操作的响应时间进行测试，因为用户的操作主要集中在系统的几个模块中，所以下面通过对这几个重要模块进行了测试。通过对测试数据的分析，对系统的性能测试总结如下。

（1）并发用户数 20 个以下时，系统的性能测试结果如图 5.1 所示。

表 5.2 并发用户数<20 的性能测试表格

事务操作	平均响应时间	最大响应时间	事务成功率
用户登录	1.75	2.87	100%
查看寝室	0.78	0.97	100%
查看学生	1.23	1.45	100%
查看公告	0.48	0.64	100%
查看申请	0.57	0.93	100%
新建申请	2.21	3.12	100%

从表中可以看出，如果在系统中并发操作的用户的数量在 20 以下的话，完成每个事务操作的平均响应时间都在 3 秒以下，事务操作也都能够成功，能较好满足用户的需求。

（2）并发用户数为 50 时，系统的性能测试结果如图 5.2 所示。

表 5.3 并发用户数=50 时的性能测试表格

事务操作	平均响应时间	最大响应时间	事务成功率
用户登录	3.10	6.24	100%
查看寝室	1.34	2.11	100%
查看学生	1.46	2.43	100%
查看公告	0.94	1.36	100%
查看申请	1.04	2.23	100%
新建申请	3.52	4.13	100%

从表中可以看到，当并发用户数为 50 时，登录操作和新建申请操作所需要的响应时间有明显的增加，而其他操作所需要的响应时间并没有大幅上升，初步估计是 angularJs 框架不需要进行频繁的获取 html 资源，各页面都以组件的的形式进行加载，一些页面花费的主要时间在于与后台的 ajax 数据获取。

结论

为了更好地满足寝室中管理者和学生的需求，提高学校服务学生的效率，本文对宿舍管理人员以及学生进行了需求分析，基于现有的几项成熟的技术，以 AngularJS 为前端框架，SpringBoot 为后端框架完成了对宿舍管理系统的设计与开发。本文所做的研究工作总结如下：

（1）分析了本文系统的研究背景、研究意义和国内外研究现状。

（2）介绍了在开发本文的宿舍管理系统过程中的开发环境和所使用的技术框架。

（3）对系统进行了需求分析，给出了几个系统功能性需求，主要包括登录、宿舍管理、学生管理、公告发布，并进行了数据库设计。

（4）针对于几个重要的功能模块，详细地介绍了如何实现，并简要介绍了几个关键算法。

（5）基于 AngularJS 技术完成了宿舍管理系统的前端设计，引入一些基于 AngularJS 的插件，利用了其特性，使得与前端的数据交互更为方便直接，更好地实现了与用户之间的交互。

本文设计与开发的宿舍管理系统，仅仅实现了设计中一些日常寝室事务的功能管理，实际上还是有许多的不足之处需要后续过程中的改进和完善：

（1）对于整个软件工程流程中的理论知识和系统建模的知识的掌握还远远不够，上文仅仅给出了系统框架图与部分流程图。在完整中的软件工程过程中，还应包括功能活动图、业务模块图、类图、时序图等，在之后的日子里我也要不断地学习软件工程和建模相关知识，不断提高我系统分析的水平，争取在日后能够针对管理系统做出更好的建模。

（2）软件的功能还不够丰富。目前的系统暂时只考虑了与目前宿舍管理工作最基础的事务处理，而对于一个完善的学生宿舍管理系统来说，本文设计实现的登录、人员、宿舍管理、住宿分配、公告发布的功能模块只满足了部分需求，在实际过程中，管理员与学生的需求肯定不仅仅局限于此，例如还应该有学生宿舍每周卫生的检查成果以及卫生评比、学生宿舍违纪信息管理还有可以提供寝室公告附件下载等等。因此，今后还需要结合学校发展的需要和宿舍管理工作的实际情况，进一步去了解管理者与学生更加具体的需求，从而能够去设计和完善该系统，以实现和提供更加丰富的宿舍

管理功能，以提高学生在宿舍中的生活质量和宿舍管理的效率。

（3）本系统在网页上的用户页面设计的还不够好。无论从美观还是从人机交互的角度上，本系统都还是有许多不足。由于所学知识的局限，没有实现更加美观舒适的用户界面，所以在之后的软件开发过程中，我也需要弥补自己在这一方面的不足，不断精进自己在这一方面的能力，在以后能够设计出更好的用户界面。

参考文献

[1] 孙金娟.我国高校数字校园建设解决方案[D].上海:华东师范大学,2004.

[2] 孟世和.重庆宽仁医院医疗设备管理信息系统的设计与实现[D].北京:电子科技大学,2013.

[3] 李理、李哲.美国高校数字校园的发展及推动因素研究[J].中国教育信息化(高教职教.2009(19):16.

[4] 王明德.中国家电工业现状分析[J].中国经济信息.2001,(7):33-41.

[5] 万德生.高校宿舍管理系统的设计与实现[D].吉林:吉林大学,2016.

[6] AngularJS 中文网[OL].http://baike.baidu.com/item/javascript/321142?fr=aladdin.

[7] 刘立.MVVM 模式分析与应用[J].微型电脑应用,2012,28(12):57-60.

[8] AngularJS 简介 _AngularJs 教程 [OL]. http://runoob.com/AngularJS/AngularJS.intro.html

[9] Adam Freeman.The Anatomy of an AngularJS App[M].Apress:2014-06-15.

[10] 王少丽.基于 AngularJS 的前端开发框架的设计与应用[D].大连海事大学,2018.

[11] 格林，夏德瑞，大漠穷秋.用 AngularJS 开发下一代 Web 应用[J].中国科技信息.2013,(23):90.

[12] 孙连山,李云倩.MVVM 框架在 Web 前端的应用研究[J].电脑知识与技术,2016,12(06):45-46.

[13] Spring 框架 [OL]. https://baike.baidu.com/item/spring%E6%A1%86%E6%9E%B6/2853288?fr=aladdin.

[14] 孙赟.基于 Spring+Extjs 的高校住宿管理系统的设计与实现[J].电子技术与软件工程.2016(24):63-65.

[15] 项目构建工具 Maven 开源社区网[OL].http://www.oschina.net/p/maven.

[16] 罗超,贾克斌,李周贤. 一种基于数字水印和非对称加密技术的 QR 二维码编译系统[A]. 中国高科技产业化研究会智能信息处理产业化分会.第九届全国信号和智能信息处理与应用学术会议专刊[C].中国高科技产业化研究会智能信息处理产业化分会:中国高科技产业化研究会,2015:6.

[17] Yu-Geng Song.Parallel Incremental Frequent Itemset Mining for Large Data[J].Journal of Computer Science ＆ Technology.2017,32(2):1.

致谢

感谢我的导师罗荣良老师，在百忙之中结合了我以往的情况，在我选题的时候给出了宝贵的意见，分析论文中存在的问题，在论文完成之前一步步地指导我。同时，我要感谢我的班主任杨枨老师，杨枨老师在大学四年中给我们班级的教导以及方向的指导都使得我受益良多，他严谨的作风也给我们全班所有同学留下了深刻的印象。在论文完成之际，谨向导师致以最崇高的敬意和由衷的感谢！

我也要衷心感谢浙江大学城市学院计算分院的各位老师们，传授了我丰富的专业知识，同时教育了我在面对问题时所应该采取的态度和解决问题的方法！我还要感谢计算 1501 全班与我一起学习的同学们，在大学四年中我们一起解决了许多问题，有了愉快的学习氛围，让我在大学中能够更好的获得知识。同时我也要感谢学校，是学校给我们提供了这个良好的学习环境，营造出了如此良好的学习氛围。

我也要感谢在实习期间所在公司的领导和同事们，是他们给我提供了一个实践的平台，在真实的项目中体会到了软件的运行流程，同时也在我遇到困难的时候给了我莫大的帮助。

最后，谨向百忙之中抽出时间评审本论文的老师们致以最诚挚的谢意！