目录

**[一、 课题背景 2](#_Toc21264)**

**[二、 系统需求分析 5](#_Toc27488)**

[2.1功能需求 5](#_Toc28361)

[2.2功能描述 5](#_Toc11120)

**[三、 系统总体设计 16](#_Toc23815)**

[3.1学生宿舍管理系统 16](#_Toc7436)

[3.2系统结构设计 19](#_Toc31818)

**[四、 系统详细设计 24](#_Toc26522)**

[4.1学生管理模块 24](#_Toc5567)

[4.2报修信息模块 25](#_Toc22913)

[4.3申请报修模块 26](#_Toc2135)

**[五、 系统实现（关键代码和测试用例） 27](#_Toc8050)**

**[六、 实习总结 41](#_Toc31098)**

**[七、 参考文献 43](#_Toc11975)**

# 课题背景

|  |
| --- |
| 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。 2．科学学术意义和应用前景 选择学生宿舍管理系统作为题目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，方便高校对学生宿舍的管理。 |

# 系统需求分析

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2.1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。 2.2功能描述2.2.1面向管理员的功能描述 表2.2.1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2.2.2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2.2.3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2.2.4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2.2.5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2.2.6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2.2.7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2.2.8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2.2.9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2.2.10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2.2.11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2.2.12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2.2.13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2.2.14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2.2.15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2.2.16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2.2.17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2.2.18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2.2.19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2.2.20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2.2.21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2.2.22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2.2.23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2.2.24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2.2.25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2.2.26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2.2.27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2.2.28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2.2.29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2.2.30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2.2.31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2.2.32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2.2.33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2.2.34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2.2.35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2.2.36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2.2.37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2.2.38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 | |

# 系统总体设计

|  |
| --- |
| 3.1学生宿舍管理系统3.1.1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。 3.1.2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。 3.1.3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。 3.1.4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。 3.1.5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。 3.1.6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。 3.1.7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 3.1.8自己的床位高亮显示 突出显示用户的床位。 3.1.9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。 3.1.10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。  **3.1.11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。 3.1.12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。 3.1.13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。 3.1.14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。 3.1.15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。3.1.16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。 3.1.17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。  **3.1.18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。 3.2系统结构设计3.2.1系统后台功能设计 wps  图3.2.1 系统总体设计  wps  图3.2.2 管理员模块详细设计  wps  图3.2.3 学生模块详细设计    图3.2.4我的宿舍详细设计    图3.2.5 楼宇信息模块功能详细设计 3.2.2 任务分配规划 |
|  |

# 系统详细设计

|  |
| --- |
| 4.1学生管理模块4.1.1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。 4.1.2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。 4.1.3性能 灵活性：窗口响应绝大部分的菜单和操作  时间性：想要鼠标单击时间在1-2秒之间 4.1.4输入项 **4.1.4.1 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。  **4.1.4.2 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。  **4.1.4.3 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。  **4.1.4.4 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。 4.1.5输出项 学生信息列表 4.2报修信息模块4.2.1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。 4.2.2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。4.2.3性能 灵活性：窗口响应绝大部分的菜单和操作  时间性：想要鼠标单击时间在1-2秒之间 4.2.4输入项 **4.2.4.1 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。  **4.2.4.2 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。  **4.2.4.3 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。  **4.2.4.4 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。  **4.2.4.5 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。 4.2.5输出项 报修信息列表 4.3申请报修模块4.3.1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。 4.3.2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。4.3.3性能 灵活性：窗口响应绝大部分的菜单和操作  时间性：想要鼠标单击时间在1-2秒之间 4.3.4输入项 **4.2.4.1 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。  **4.2.4.2 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。 4.3.5输出项 申请报修信息列表 |

# 系统实现（关键代码和测试用例）

|  |
| --- |
| 5.1 学生页面    修改：    新增：    Controller:  package com.example.springboot.controller;  import cn.hutool.core.map.MapUtil;  import com.example.springboot.util.Result;  import com.example.springboot.entity.Student;  import com.example.springboot.entity.User;  import com.example.springboot.service.StudentService;  import com.github.pagehelper.PageInfo;  import lombok.val;  import org.springframework.web.bind.annotation.\*;  import javax.annotation.Resource;  import javax.servlet.http.HttpSession;  import java.util.HashMap;  @RestController  @RequestMapping("/stu")  public class StudentController {  @Resource  private StudentService studentService;  @PostMapping("/add")  public Result<?> add(@RequestBody Student student) {  int i = studentService.addNewStudent(student);  if (i == 1) {  return Result.success();  } else {  return Result.error("-1", "添加失败");  }  }  @PutMapping("/update")  public Result<?> update(@RequestBody Student student) {  int i = studentService.updateNewStudent(student);  if (i == 1) {  return Result.success();  } else {  return Result.error("-1", "更新失败");  }  }  @DeleteMapping("/delete/{username}")  public Result<?> delete(@PathVariable String username) {  int i = studentService.deleteStudent(username);  if (i == 1) {  return Result.success();  } else {  return Result.error("-1", "删除失败");  }  }  @GetMapping("/find")  public Result<?> findPage(@RequestParam(defaultValue = "1") Integer pageNum,  @RequestParam(defaultValue = "10") Integer pageSize,  @RequestParam(defaultValue = "") String search) {  PageInfo page = studentService.find(pageNum, pageSize, search);  val build = MapUtil.builder(new HashMap<String, Object>())  .put("records", page.getList()) //数据  .put("total", page.getTotal())//总条数  .put("pages", page.getPages())//页数  .put("current", page.getPageNum()).build();//当前页  if (page != null) {  return Result.success(build);  } else {  return Result.error("-1", "查询失败");  }  }  @PostMapping("/login")  public Result<?> login(@RequestBody User user, HttpSession session) {  System.out.println(user.getUsername());  System.out.println(user.getPassword());  Object o = studentService.stuLogin(user.getUsername(), user.getPassword());  if (o != null) {  System.out.println(o);  //存入session  session.setAttribute("Identity", "stu");  session.setAttribute("User", o);  return Result.success(o);  } else {  return Result.error("-1", "用户名或密码错误");  }  }  @GetMapping("/stuNum")  public Result<?> stuNum() {  int num = studentService.stuNum();  if (num > 0) {  return Result.success(num);  } else {  return Result.error("-1", "查询失败");  }  }  @GetMapping("/exist/{value}")  public Result<?> exist(@PathVariable String value) {  Student student = studentService.stuInfo(value);  if (student != null) {  return Result.success(student);  } else {  return Result.error("-1", "不存在该学生");  }  }  }  Service:  package com.example.springboot.service;  import com.example.springboot.entity.Student;  import com.github.pagehelper.PageInfo;  public interface StudentService {  Student stuLogin(String username, String password);  int addNewStudent(Student student);  PageInfo find(Integer pageNum, Integer pageSize, String search);  int updateNewStudent(Student student);  int deleteStudent(String username);  int stuNum();  Student stuInfo(String username);  }  Impl:  package com.example.springboot.service.impl;  import com.example.springboot.entity.Student;  import com.example.springboot.mapper.StudentMapper;  import com.example.springboot.service.StudentService;  import com.github.pagehelper.PageHelper;  import com.github.pagehelper.PageInfo;  import org.springframework.stereotype.Service;  import javax.annotation.Resource;  import java.util.List;  @Service  public class StudentServiceImpl implements StudentService {  @Resource  private StudentMapper studentMapper;  @Override  public Student stuLogin(String username, String password) {  Student student = studentMapper.selectOne(username,password);  if (student != null) {  return student;  } else {  return null;  }  }  @Override  public int addNewStudent(Student student) {  int insert = studentMapper.insert(student);  return insert;  }  @Override  public PageInfo find(Integer pageNum, Integer pageSize, String search) {  PageHelper.startPage(pageNum,pageSize);  List<Student> studentPage = studentMapper.selectPage(search);  PageInfo<Student> pageInfo = new PageInfo<>(studentPage);  return pageInfo;  }  @Override  public int updateNewStudent(Student student) {  int i = studentMapper.updateById(student);  return i;  }  @Override  public int deleteStudent(String username) {  int i = studentMapper.deleteById(username);  return i;  }  @Override  public int stuNum() {  int stuNum = Math.toIntExact(studentMapper.selectCount());  return stuNum;  }  @Override  public Student stuInfo(String username) {  Student student = studentMapper.selectOne2(username);  return student;  }  }  StudentMapper:  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="com.example.springboot.mapper.StudentMapper">  <resultMap id="BaseResultMap" type="com.example.springboot.entity.Student">  <id column="username" property="username"/>  <result column="password" property="password"/>  <result column="age" property="age"/>  <result column="name" property="name"/>  <result column="gender" property="gender"/>  <result column="phone\_num" property="phoneNum"/>  <result column="email" property="email"/>  <result column="avatar" property="avatar"/>  </resultMap>  <insert id="insert" parameterType="com.example.springboot.entity.Student">  insert into student(username,  `password`,  age,  `name`,  gender,  `phone\_num`,  email,  avatar  )  values (  #{student.username},  #{student.password},  #{student.age},  #{student.name},  #{student.gender},  #{student.phoneNum},  #{student.email},  #{student.avatar})  </insert>  <update id="updateById" parameterType="com.example.springboot.entity.Student">  update student set password = #{student.password},age = #{student.age},  `name` = #{student.name},gender = #{student.gender},  phone\_num = #{student.phoneNum},`email` = #{student.email},  avatar = #{student.avatar}  where username = #{student.username}  </update>  <delete id="deleteById" parameterType="java.lang.String">  delete from student where `username` = #{username}  </delete>  <select id="selectPage" resultMap="BaseResultMap" parameterType="java.lang.String">  select \* from student where name like concat ('%',#{search},'%')  </select>  <select id="selectCount" resultType="java.lang.Long">  select count(\*) as count from student where username is not null  </select>  <select id="selectOne" resultMap="BaseResultMap">  select \* from student where username = #{username} and password = #{password}  </select>  <select id="selectOne2" resultMap="BaseResultMap"  parameterType="java.lang.String">  select \* from student where `username` = #{username}  </select>  </mapper>  5.2 报修模块页面  管理员端：    修改：    学生端：    新增：    Controller：  **package** com.example.springboot.controller;  **import** cn.hutool.core.map.MapUtil;  **import** com.example.springboot.util.Result;  **import** com.example.springboot.entity.Repair;  **import** com.example.springboot.service.RepairService;  **import** com.github.pagehelper.PageInfo;  **import** lombok.val;  **import** org.springframework.web.bind.annotation.\*;  **import** javax.annotation.Resource;  **import** java.util.HashMap;  @RestController  @RequestMapping(**"/repair"**)  **public class** RepairController {  @Resource  **private** RepairService **repairService**;  @PostMapping(**"/add"**)  **public** Result<?> add(@RequestBody Repair repair) {  **int** i = **repairService**.addNewOrder(repair);  **if** (i == 1) {  **return** Result.*success*();  } **else** {  **return** Result.*error*(**"-1"**, **"添加失败"**);  }  }  @PutMapping(**"/update"**)  **public** Result<?> update(@RequestBody Repair repair) {  **int** i = **repairService**.updateNewOrder(repair);  **if** (i == 1) {  **return** Result.*success*();  } **else** {  **return** Result.*error*(**"-1"**, **"更新失败"**);  }  }  @DeleteMapping(**"/delete/{id}"**)  **public** Result<?> delete(@PathVariable Integer id) {  **int** i = **repairService**.deleteOrder(id);  **if** (i == 1) {  **return** Result.*success*();  } **else** {  **return** Result.*error*(**"-1"**, **"删除失败"**);  }  }  @GetMapping(**"/find"**)  **public** Result<?> findPage(@RequestParam(defaultValue = **"1"**) Integer pageNum,  @RequestParam(defaultValue = **"10"**) Integer pageSize,  @RequestParam(defaultValue = **""**) String search) {  PageInfo page = **repairService**.find(pageNum, pageSize, search);  val build = MapUtil.*builder*(**new** HashMap<String, Object>())  .put(**"records"**, page.getList()) *//数据*  .put(**"total"**, page.getTotal())*//总条数*  .put(**"pages"**, page.getPages())*//页数*  .put(**"current"**, page.getPageNum()).build();*//当前页*  **if** (page != **null**) {  **return** Result.*success*(build);  } **else** {  **return** Result.*error*(**"-1"**, **"查询失败"**);  }  }  @GetMapping(**"/find/{name}"**)  **public** Result<?> individualFind(@RequestParam(defaultValue = **"1"**) Integer pageNum,  @RequestParam(defaultValue = **"10"**) Integer pageSize,  @RequestParam(defaultValue = **""**) String search,  @PathVariable String name) {  System.***out***.println(name);  PageInfo page = **repairService**.individualFind(pageNum, pageSize, search, name);  val build = MapUtil.*builder*(**new** HashMap<String, Object>())  .put(**"records"**, page.getList()) *//数据*  .put(**"total"**, page.getTotal())*//总条数*  .put(**"pages"**, page.getPages())*//页数*  .put(**"current"**, page.getPageNum()).build();*//当前页*  **if** (page != **null**) {  **return** Result.*success*(build);  } **else** {  **return** Result.*error*(**"-1"**, **"查询失败"**);  }  }  @GetMapping(**"/orderNum"**)  **public** Result<?> orderNum() {  **int** num = **repairService**.showOrderNum();  **if** (num >= 0) {  **return** Result.*success*(num);  } **else** {  **return** Result.*error*(**"-1"**, **"报修统计查询失败"**);  }  }  }  service：  **package** com.example.springboot.service;  **import** com.example.springboot.entity.Repair;  **import** com.github.pagehelper.PageInfo;  **public interface** RepairService {  **public int** showOrderNum();  **int** addNewOrder(Repair repair);  PageInfo find(Integer pageNum, Integer pageSize, String search);  PageInfo individualFind(Integer pageNum, Integer pageSize, String search, String name);  **int** updateNewOrder(Repair repair);  **int** deleteOrder(Integer id);  }  Impl：  **package** com.example.springboot.service.impl;  **import** com.example.springboot.entity.Repair;  **import** com.example.springboot.mapper.RepairMapper;  **import** com.example.springboot.service.RepairService;  **import** com.github.pagehelper.PageHelper;  **import** com.github.pagehelper.PageInfo;  **import** org.springframework.stereotype.Service;  **import** javax.annotation.Resource;  **import** java.util.List;  @Service  **public class** RepairServiceImpl **implements** RepairService {  @Resource  **private** RepairMapper **repairMapper**;  @Override  **public int** addNewOrder(Repair repair) {  **int** insert = **repairMapper**.insert(repair);  **return** insert;  }  @Override  **public** PageInfo find(Integer pageNum, Integer pageSize, String search) {  PageHelper.*startPage*(pageNum,pageSize);  List<Repair> orderPage = **repairMapper**.selectPage(search);  PageInfo<Repair> pageInfo = **new** PageInfo<>(orderPage);  **return** pageInfo;  }  @Override  **public** PageInfo individualFind(Integer pageNum, Integer pageSize, String search, String name) {  PageHelper.*startPage*(pageNum,pageSize);  List<Repair> orderPage = **repairMapper**.selectPage1(search,name);  PageInfo<Repair> pageInfo = **new** PageInfo<>(orderPage);  **return** pageInfo;  }  @Override  **public int** updateNewOrder(Repair repair) {  **int** i = **repairMapper**.updateById(repair);  **return** i;  }  @Override  **public int** deleteOrder(Integer id) {  **int** i = **repairMapper**.deleteById(id);  **return** i;  }  @Override  **public int** showOrderNum() {  **int** orderCount = Math.*toIntExact*(**repairMapper**.selectCount());  **return** orderCount;  }  }  RepairMapper：  *<?*xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="com.example.springboot.mapper.RepairMapper">  <resultMap id="BaseResultMap" type="com.example.springboot.entity.Repair">  <id column="id" property="id"/>  <result column="repairer" property="repairer"/>  <result column="dormbuild\_id" property="dormBuildId"/>  <result column="dormroom\_id" property="dormRoomId"/>  <result column="title" property="title"/>  <result column="content" property="content"/>  <result column="state" property="state"/>  <result column="order\_buildtime" property="orderBuildTime"/>  <result column="order\_finishtime" property="orderFinishTime"/>  </resultMap>  <insert id="insert" parameterType="com.example.springboot.entity.Notice">  insert into repair(repairer,  `dormbuild\_id`,  dormroom\_id,  title,  content,  `state`,  order\_buildtime,  order\_finishtime  )  values (  #{repair.repairer},  #{repair.dormBuildId},  #{repair.dormRoomId},  #{repair.title},  #{repair.content},  ifnull(#{repair.state},'未完成'),  #{repair.orderBuildTime},  #{repair.orderFinishTime})  </insert>  <update id="updateById" parameterType="com.example.springboot.entity.Repair">  update repair set repairer = #{repair.repairer},dormbuild\_id = #{repair.dormBuildId},  dormroom\_id = #{repair.dormRoomId},title = #{repair.title},  content = #{repair.content},`state` = #{repair.state},  order\_buildtime = #{repair.orderBuildTime},order\_finishtime = #{repair.orderFinishTime}  where id = #{repair.id}  </update>  <delete id="deleteById" parameterType="java.lang.Integer">  delete from repair where id = #{id}  </delete>  <select id="selectPage" resultMap="BaseResultMap" parameterType="java.lang.String">  select \* from repair where title like concat ('%',#{search},'%')  </select>  <select id="selectPage1" resultMap="BaseResultMap">  select \* from repair where repairer = #{name} and title like concat ('%',#{search},'%')  </select>  <select id="selectCount" resultType="java.lang.Long">  select count(\*) as count from repair  </select>  </mapper> |

# 

# 实习总结

|  |
| --- |
| 指缝很宽，时间太瘦，悄悄从指缝间溜走。  令大家都没想到的是，突然起来的通知打乱了所有人的计划，老师的教学计划以及所有组的项目规划都是被极大程度的削减了。或许在不久的将来，这次的项目会成为大家的一个遗憾吧。  最后一次的学习机会，我们学习了有关Spring Boot的相关后端知识，与原本SSM框架的项目相比，Spring Boot给我最深刻的感觉就是方便，简化。我们的最后一次团队项目是由Spring Boot+Vue共同搭建的前后端分离的大项目，是一个能检验我们两个阶段学习成果的大好机会，但由于时间原因，项目只能大打折扣，无法呈现出最完美的一面，由起初的每人四个模块变成后面的每人两个模块，直接是少了一半。但通过这个阶段的学习无疑是给自己的专业能力又提升了一个档次，为接下来的毕业设计以及工作实习都是一个很好的铺垫。每一次的团队项目都是锻炼自己专业能力与团队协作能力的一次好机会，这些能力在进入社会工作上都是必须的，大学期间的每一次课程设计包括实训答辩都是对自己这些能力的检验与提升。  最后，也是感谢学校能给我们这次能够锻炼自己的能力，也感谢老师的默默付出与谆谆教诲。 |

# 

# 参考文献

|  |
| --- |
| [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 |