评阅教师	设计成绩	评阅日期

海南大学计算机与网络空间安全学院

《数据库系统》课程设计报告

班　　级：_____

成　　员：江雪雨

指导老师：_____

完成日期：2020 年 6 月 25 日

目录

学生公寓管理系统.....1
1. 设计任务.....2
1.1 设计目的.....2
1.2 设计内容.....2
2. 需求分析.....3
2.1 需求分析任务.....3
2.2 需求分析过程.....3
3. 概念结构设计.....7
3.1 概念结构设计任务.....7
3.2 概念结构设计方法.....7
4. 逻辑结构设计.....10
4.1 逻辑结构设计任务.....10
4.2 E-R 图向关系模型的转化.....10
5. 物理结构设计.....12
6. 数据库的实施.....14
6.1 系统的设计思想.....14
6.2 主要技术.....14
6.3 管理页面.....14
6.4 系统调试与测试.....21
6.5 各功能代码设计.....23
7. 总结.....35

学生公寓管理系统

摘要： 随着社会的发展以及教育水平的提高，当今在校生数量越来越多。与此同时，目前还有一些学校还在使用手工的方式对学生的住宿信息进行管理，手工记录对于规模小的学校来说还可以勉强接受，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，不但浪费了许多时间，而且效率也低。针对这个情况，设计了一套学生公寓管理系统。学生公寓管理系统采用的是计算机化管理，通过强大的计算机技术给公寓管理人员和学生带来便利，通过网络可以在系统上查询学生公寓状况，同时管理人员还可以对学生的信息进行增删改查，最大程度了解所有学生的住宿信息。

关键词：公寓管理系统，数据库系统，课程设计

1. 设计任务

1.1 设计目的

不但能够实现少量的管理人员完成大量的人员安排公寓管理服务，使公寓管理更加规范化、科学化，人性化。而且同时还要实时动态的掌握每个公寓管理的基本信息，以及统计相关数据更新。

1.2 设计内容

(1)基本要求

学校有若干公寓，每栋七层，每层 16 个房间，每个房间 4 个床位，需要一个公寓管理系统实现管理。

(2) 基本功能

①寝室分配:根据系别、年级、班级分配寝。查询寝室状态和入住信息。

②学生管理:实现入住学生信息的维护和查询功能。

③信息查询:按公寓楼号、学生姓名等查询住宿信息。

④财产管理:实现对公寓财产的管理功能。

⑤出入登记:实现对学生搬出公寓的贷物进行登记和对外来人员进行登记等功能。

⑥系统管理:参数设置(包括公寓楼号、寝室房号、系别、年级、班级的设置)、权限管理和系统维护(数据备份、数据恢复)

2. 需求分析

2.1 需求分析任务

需求分析的任务是通过详细调查现实世界要处理的对象，充分了解原系统的工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。调查的重点是“数据”和“处理”，通过调查，收集与分析，获得用户对数据库的信息要求处理要求，安全性与完整性要求。

为了完成需求分析的任务，首先要调查清楚用户的实际要求，与用户达成共识，然后分析与表达这些需求。

随着信息技术的发达和科技的发展，学生数量逐渐增多，对于管理学生住宿信息的需求也逐渐变大，基于这种需求，学生公寓管理系统不但使学校能享受更好的管理效率，还能满足对学生和宿管的信息管理需求。

2.2 需求分析过程

2.2.1 用户需求分析

随着社会的发展以及教育水平的提高，当今在校生数量越来越多。与此同时，目前还有一些学校还在使用手工的方式对学生的住宿信息进行管理，手工记录对于规模小的学校来说还可以勉强接受，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，不但浪费了许多时间，而且效率也低。针对如此，设计了一套学生宿舍管理系统。

2.2.2 处理对象分析

系统要处理的对象包括学生基本信息、宿管基本信息、楼宇基本信息、宿管基本信息、住宿基本信息、超级管理员基本信息等六个方面，各个对象包括信息如下所示：

(1) 学生基本信息(Student)：包括学生学号、学生姓名、学生性别、学生密码等方面的信息，可以方便学生信息的查询和更新；

(2) 宿管基本信息(Dormitory Manager)：包括宿管编号、宿管姓名、宿管性别、宿管密码等，以方便管理人员对宿舍管理人员的任用、信息查询及更改；

(3) 楼宇基本信息(Building)：包括楼宇名称、所属宿管、所属位置等方面，这样可以方便管

理者对楼宇的管理，提高查询效率；

(4) 宿舍基本信息(Dormitory)：包括宿舍编号、所属楼宇、所属楼层、最多可住人数、已住人数;

(5) 住宿基本信息(Live)：包括学生姓名，宿舍编号，入住日期等，可以方便管理人员对学生进行信息查询及更改；

(6) 系统管理(Admin)：超级管理员编号，超级管理员名字，超级管理员密码等；

2.2.3 功能需求分析

1．登陆功能

已存在数据库中的用户输入自己的信息后可以进行登录，用户可以根据提示输入用 ID、密码，并选择权限，点击登陆即可进入相应的管理界面。

2．学生功能

学生登录系统后权限是最小的，学生只能查看自己的姓名，性别，学号，ID，密码，缴费信息等信息，或者对自己的信息作出修改。

3．宿管功能

同样，也只有存在数据库中的宿管才能登录该系统。宿管登录系统后，权限比学生登录多很多。

（1）学生管理

宿管可以查看其所管理宿舍的学生信息，并可以为学生分配宿舍，甚至可以修改学生的住宿信息；

（2）宿管管理

宿管可登陆系统对自己的姓名，密码，ID 等信息进行增、删、改，查；

（3）楼宇管理

宿管可以登录系统查看其所管理的楼宇信息，比如楼宇的位置，名称，ID 等信息。同时也可以对自己所管辖的楼宇进行检索。此功能是为了方便当一个宿管管辖的楼宇过多时，查看某座楼宇的信息。

（4）宿舍管理

宿管可以登录系统查看其所管理的宿舍信息，比如宿舍的所属楼宇，宿舍号， ID，所属楼层，最大入住人数，已住人数等信息。同时也可以对自己所管辖的宿舍进行添加或者删除。也可以按要求对自己管辖的宿舍进行检索。此功能是为了方便当一个宿管管辖的宿舍过多时，查看某宿舍的信息。

（5）住宿管理

宿管可以登录系统查看其所管理的学生宿舍信息，并能对宿舍信息进行增、删、改、查。比如添加住宿学生，调整住宿学生的宿舍信息，为学生退宿等。同时宿管还可以按照学生姓名查看学生的宿舍信息，或者按照宿舍信息查看宿舍中的学生入住信息。

4．管理员功能

管理员的权限是最大的，管理员登录系统后的功能最为全面，用到了前面所涉及的所有数据库的信息。这里在进行系统设计是将管理员数量预设为一名，预设其姓名为 admin，密码也是 admin。

（1）学生管理

管理员不仅可以查看其所管理宿舍的学生信息，并可以为学生分配宿舍，甚至可以修改学生的住宿信息，他拥有最高的权限，可以向数据库添加一个新的学生信息，或者删除一个学生信息。

（2）宿管管理

同样，管理员不仅可以对一个宿管的姓名，密码，ID 等信息进行增、删、改，查。也可以向数据库中添加一个宿管的信息，或者在数据库中删除一个宿管的信息。

（3）楼宇管理

管理员登录系统查看所有楼宇信息，包括楼宇的位置，名称，ID 等信息。同时也可以按楼宇的 ID 或者位置对楼宇进行检索。同时可以对楼宇信息进行增、删、改、查。

（4）宿舍管理

管理员可以登录系统查看所有楼宇的所有宿舍信息，包括宿舍的所属楼宇，宿舍号， ID，

所属楼层，最大入住人数，已住人数等信息。当然可以所有宿舍进行添加或者删除。也可以按要求对宿舍进行检索。或者从数据库中添加或者删除一个宿舍的信息。

（5）住宿管理

管理员登录系统可以查看所有学生的宿舍信息，并能对宿舍信息进行增、删、改、查。比如添加住宿学生信息，调整住宿学生的宿舍信息，为学生退宿等。同时管理员还可以按照学生姓名查看学生的宿舍信息，或者按照宿舍信息查看宿舍中的学生入住信息。不过管理员主要对住宿信息进行宏观调控，查看等操作。

（6）系统管理

与学生和宿管登录该系统一样，管理员也能对自己的信息进行查看和修改，此外管理员还能查看当前管理员的数量，以及各个管理员的当前状态（是否可用）。

2.2.4 安全性与完整性分析

安全性先通过视图机制，不同的用户只能访问系统授权的视图，这样可提供系统数据一定程度上的安全性，再通过用户授权机制，欲用户登陆来识别用户级别，根据这个级别来分配用户权限，达到数据更高层次的安全保密功能。

完整性要求用于描述学生基本信息、宿管基本信息、楼宇基本信息、宿管基本信息、住宿基本信息、超级管理员基本信息。

3. 概念结构设计

3.1 概念结构设计任务

概念结构设计的任务是将需求分析得到的用户需求抽象为信息结构。

3.2 概念结构设计方法

设计概念结构通常有四类方法：自顶向下，自底向上，逐渐扩张以及混合策略。本次学生公寓管理系统采用的是自底向上的方法，首先定义全局的概念结构的框架，然后逐步细化。根据自顶向上地进行需求分析然后再自底上地进行概念设计。

3.2.1 概念结构设计步骤

概念结构的设计可分为两步：

(1) 抽象数据并设计局部视图。根据需求分析，总结出该宿舍管理系统中的实体包括：学生,宿管，楼宇，宿舍，住宿，管理员等六部分。

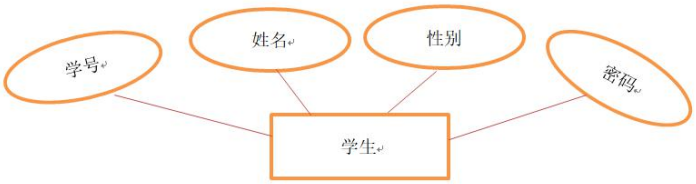


图 1 学生实体局部 E-R 图

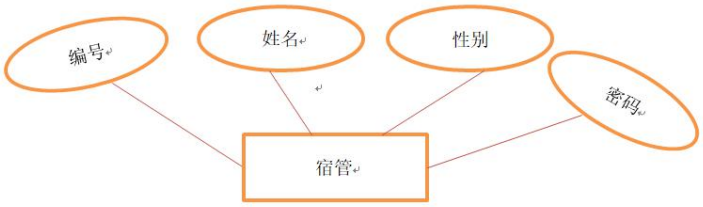


图 2 宿管实体局部 E-R 图

图 3 楼宇实体局部 E-R 图

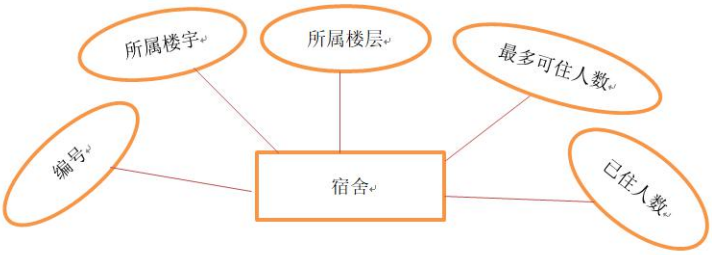


图 4 宿舍实体局部 E-R 图

图 5 住宿实体局部 E-R 图

图 6 超级管理员实体局部 E-R 图

(2) 集成局部视图，得到全局的概念结构。

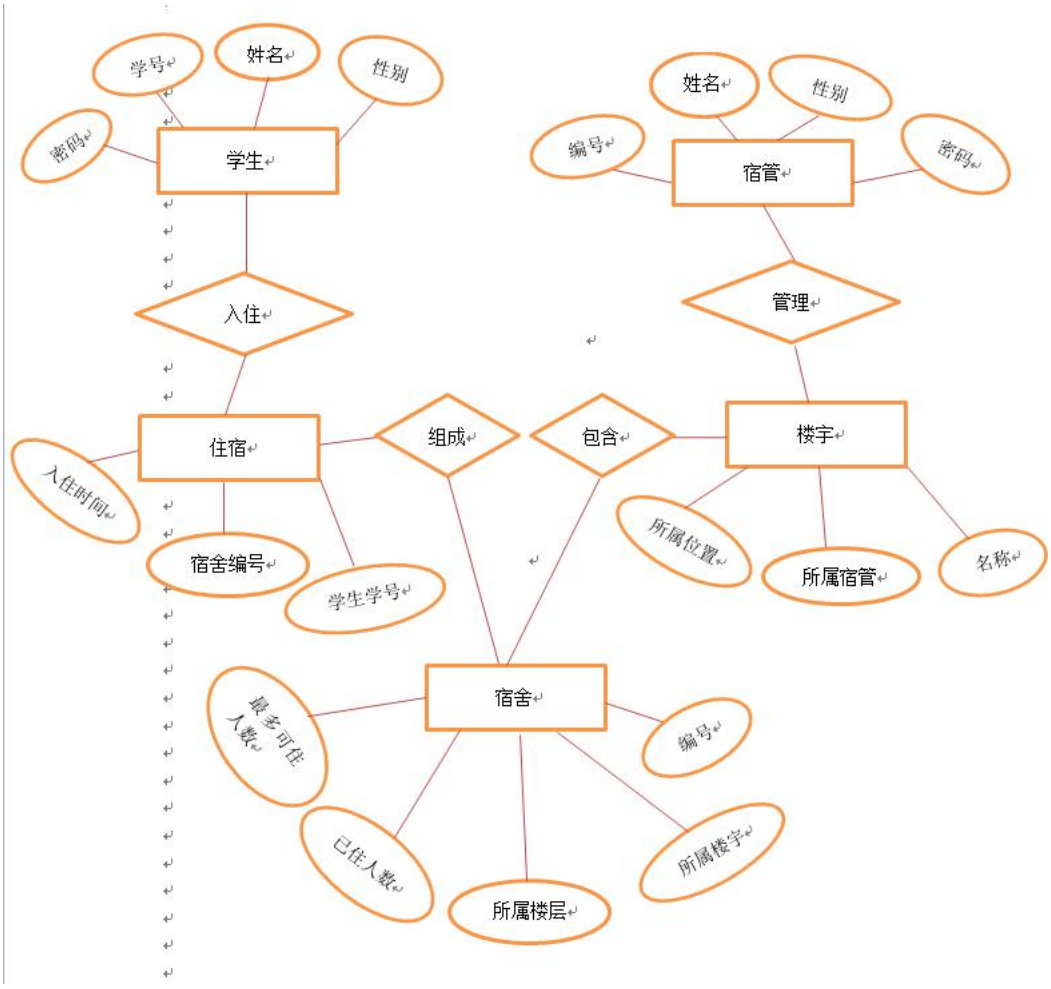


图 7 学生公寓管理 E-R 图

4. 逻辑结构设计

4.1 逻辑结构设计任务

逻辑结构设计的任务是把概念结构设计好的基本 E-R 图转换为与选用数据库管理系统产品所支持的数据模型相符合的逻辑结构。

4.2 E-R 图向关系模型的转化

(1) 学生表(Student)：学生学号 sn、学生姓名 name、学生性别 sex、学生密码 password

(2) 宿管表(Dormitory Manager)：宿管编号 sn、宿管姓名 name、宿管性别 sex、宿管密码 password

(3) 楼宇表(Building)：楼宇名称 name、所属宿管 dormitoryManagerId、所属位置 location

(4) 宿舍表(Dormitory)：宿舍编号 sn、所属楼宇 buildingId、所属楼层 floor、最多可住人数 maxNumber、已住人数 liveNumber;

(5) 住宿表(Live)：学生姓名 name，宿舍编号 dormitoryId，入住日期 liveDate;

(6) 系统表(Admin)：超级管理员编号 id，超级管理员名字 name，超级管理员密码 password;

(7) 包含：宿舍编号 sn，楼宇 id

(8) 管理：宿管编号 sn，楼宇 id

(9) 入住：学生学号 sn，住宿 id

(10) 包含：楼宇 id，宿舍编号 sn

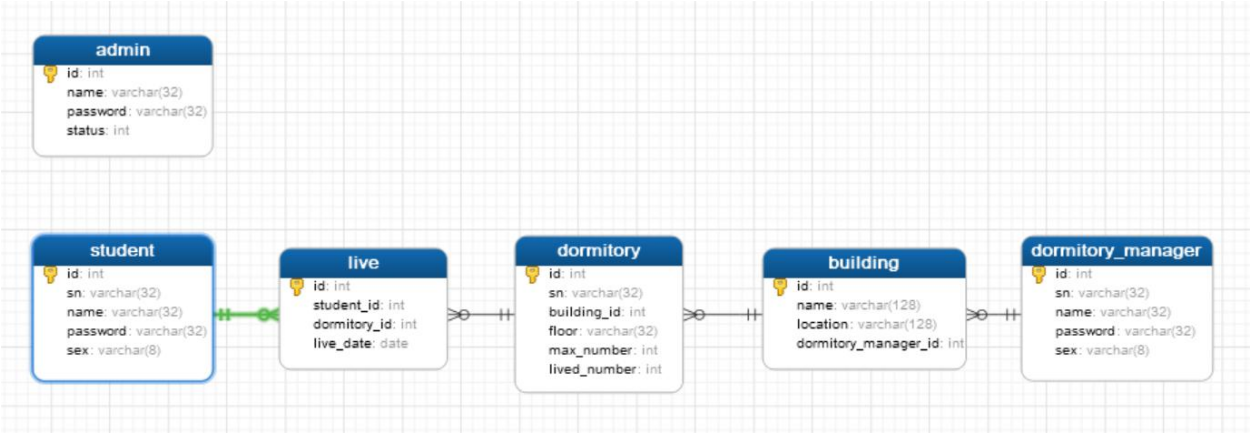

admin
id: int
name: varchar(32)
password: varchar(32)
status: int
student
id: int
sn: varchar(32)
name: varchar(32)
password: varchar(32)
sex: varchar(8)
live
id: int
student_id: int
dormitory_id: int
live_date: date
dormitory
id: int
sn: varchar(32)
building_id: int
floor: varchar(32)
max_number: int
lived_number: int
building
id: int
name: varchar(128)
location: varchar(128)
dormitory_manager_id: int
dormitory_manager
id: int
sn: varchar(32)
name: varchar(32)
password: varchar(32)
sex: varchar(8)

图 8 数据库关系图

5. 物理结构设计

根据逻辑设计出的逻辑模式，DBMS 及计算机系统所提供的手段和施加的限制，设计数据库的内模式，即文件结构，各种路径，控件分配，记录的存取方式等，为逻辑数据结构选取一个最合适的应用环境的物理结构。

名	类型	长度	小数点	允许空值 (	
id	int	11	0	☐	1
sn	varchar	32	0	☐	
name	varchar	32	0	☐	
password	varchar	32	0	☐	
sex	varchar	8	0	☐	

图 9 学生基本信息表

名	类型	长度	小数点	允许空值 (	
id	int	11	0	☐	1
sn	varchar	32	0	☐	
name	varchar	32	0	☐	
password	varchar	32	0	☐	
sex	varchar	8	0	☐	

图 10 宿管基本信息表

名	类型	长度	小数点	允许空值 (	
id	int	11	0	☐	1
name	varchar	128	0	☐	
location	varchar	128	0	☑	
dormitory_manager_id	int	11	0	☐	

图 11 楼宇基本信息表

名	类型	长度	小数点	允许空值 (	
id	int	11	0	☐	1
sn	varchar	32	0	☐	
building_id	int	11	0	☐	
floor	varchar	32	0	☐	
max_number	int	2	0	☐	
lived_number	int	2	0	☐	

图 12 宿舍基本信息表

名	类型	长度	小数点	允许空值 (	
id	int	11	0	☐	1
student_id	int	11	0	☐	
dormitory_id	int	11	0	☐	
live_date	date	0	0	☐	

图 13 住宿基本信息表

名	类型	长度	小数点	允许空值 (	
id	int	11	0	☐	1
name	varchar	32	0	☐	
password	varchar	32	0	☐	
status	int	1	0	☐	

图 14 超级管理员基本信息表

6. 数据库的实施

6.1 系统的设计思想

首先考虑系统安全性，本系统默认管理员已经将学生信息录入数据库中，管理员以及宿管都可以增加或者删除数据库中学生表的学生信息，但是学生不能自行添加注册信息，只能登陆管理自己的信息。通过 B/S 模式实现人员登陆管理，只有存在于数据库学生表中的学生，以及存在的宿管和管理员才可以登录本系统，有效避免了非法人员登陆系统。此外本系统还对用户的权限进行了分级设置，用户在输入自己的姓名，密码以及验证码后，需要选择合适的身份才能登陆系统，用户级别分为学生、宿管和管理员，不同级别的用户有不同的操作权限。用户登录系统后才可以进行相应权限下的增、删、改、查操作。

其次，是对数据库的操作。在用户成功登录系统后所做的每一次的增、删、改、查都动态的对后台数据库中相应表中的信息作出了修改，该部分主要应用 JSP 技术实现对数据库信息的动态操作。数据库采用 MySQL，能够清晰的展现大量的数据，能存放和读取大量的数据。通过 JSP 技术实现对数据库信息的动态管理并应用 navicat 数据库可视化工具随时检验自己在开发过程中的操作是否有误。

6.2 主要技术

6.2.1 前端

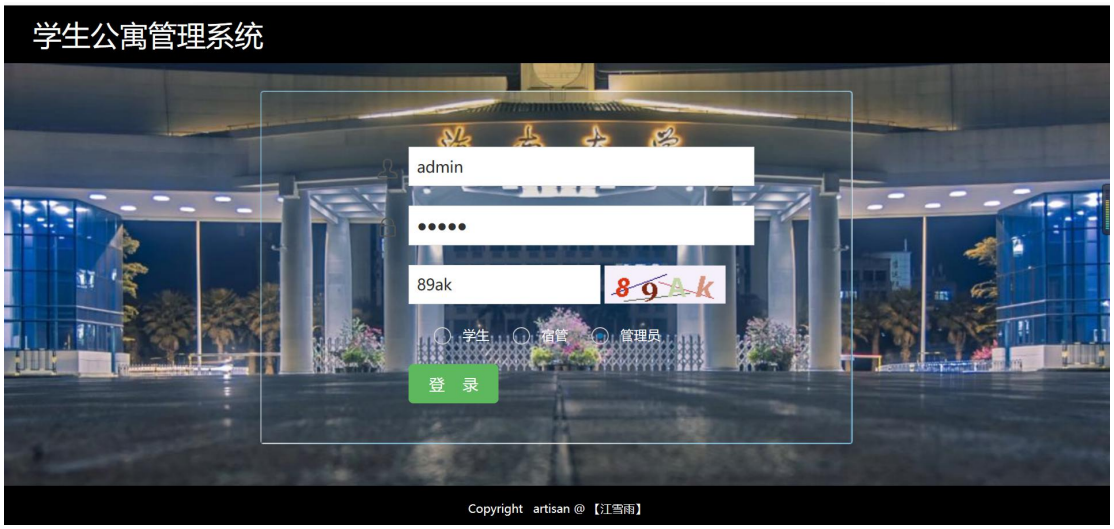
html5 ,css3, jsp,java servlet， EasyUI 框架

6.2.2 后端

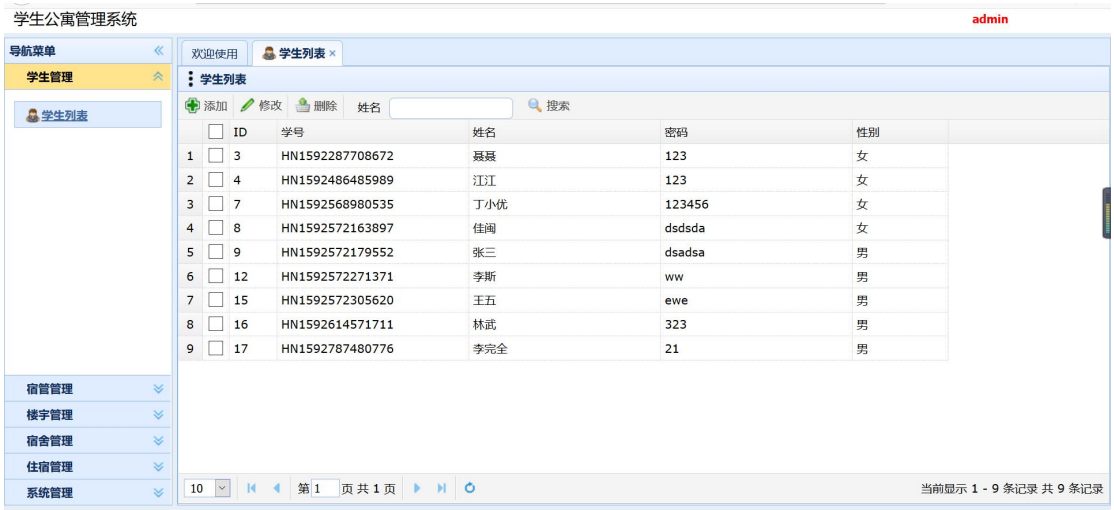
Java 基础，Tomcat 简单使用，MySQL 数据库

6.3 管理页面

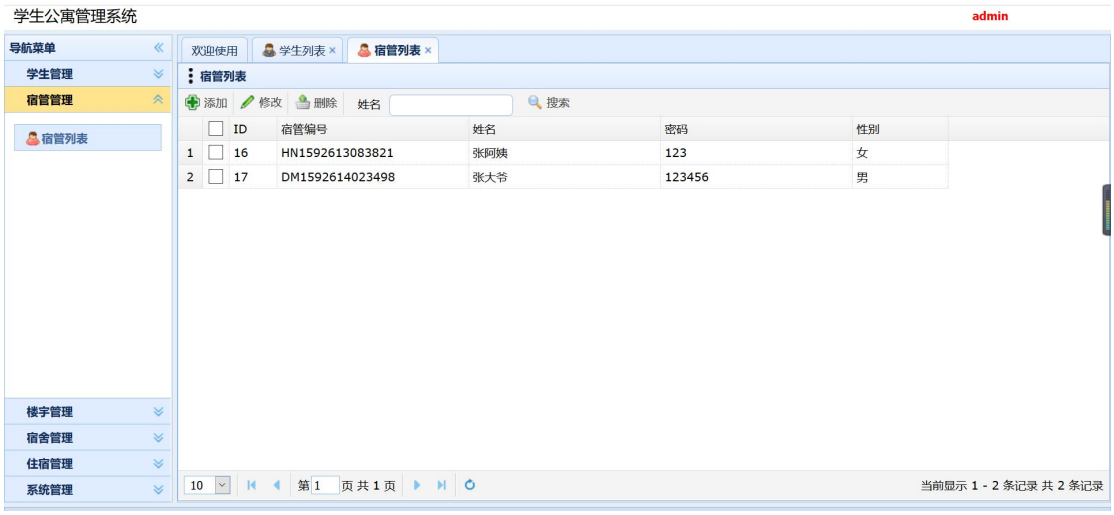
6.3.1 登陆界面--以超级管理员身份登陆



(1) 学生管理--学生列表

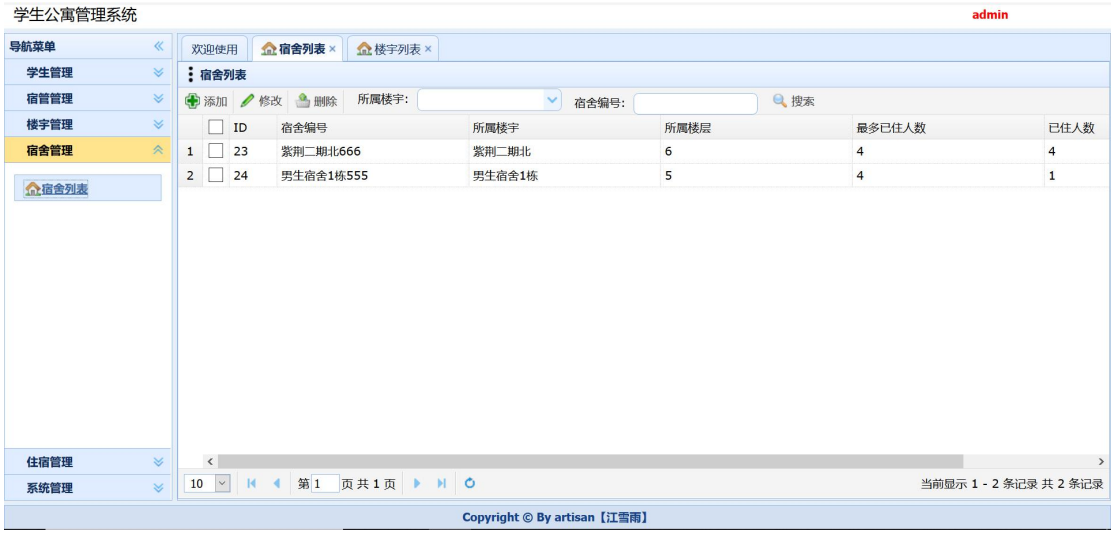


(2) 宿管管理--宿管列表

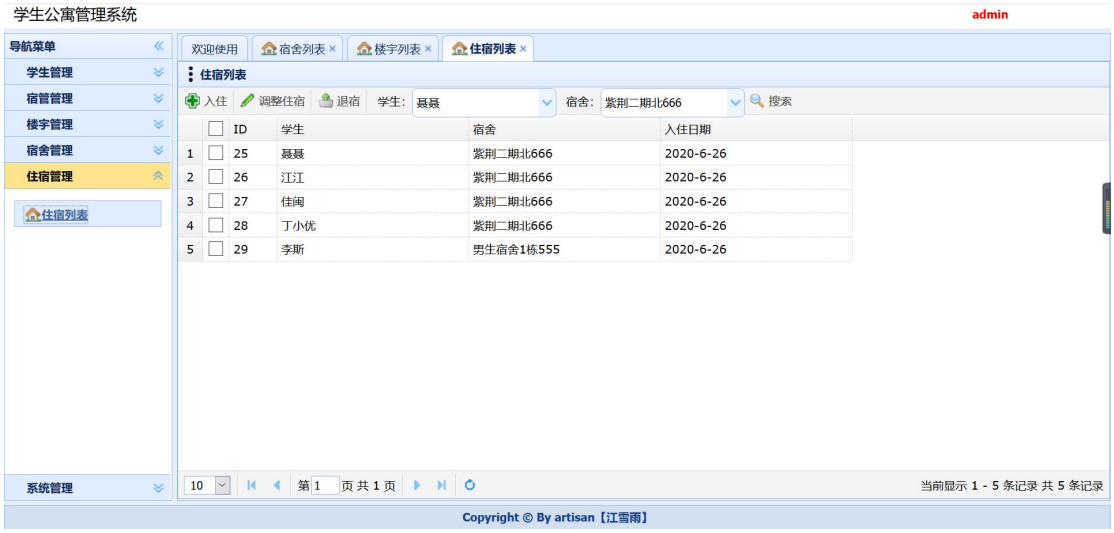


(3) 楼宇管理--楼宇列表

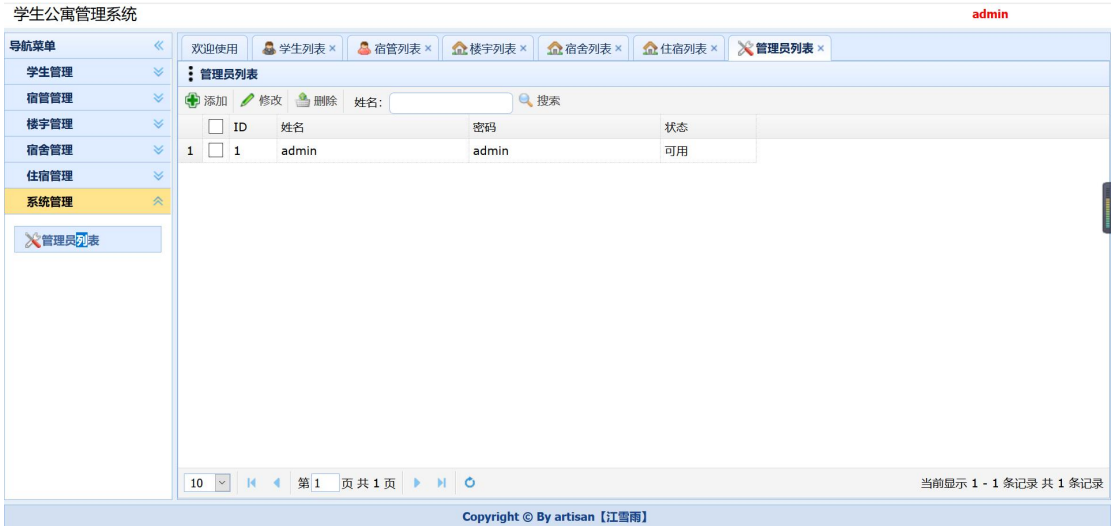
(4) 宿舍管理--宿舍列表



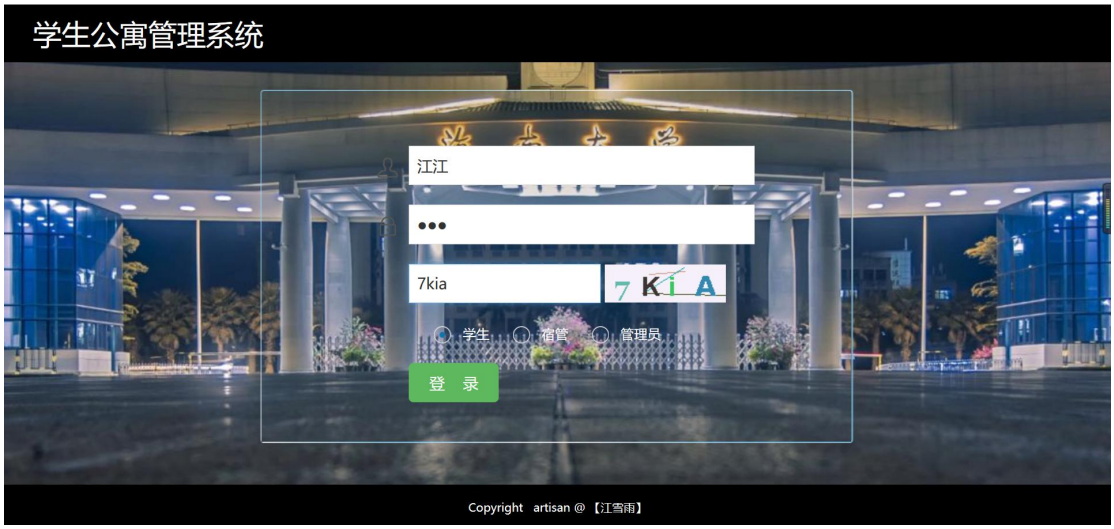
(5) 住宿管理--住宿列表



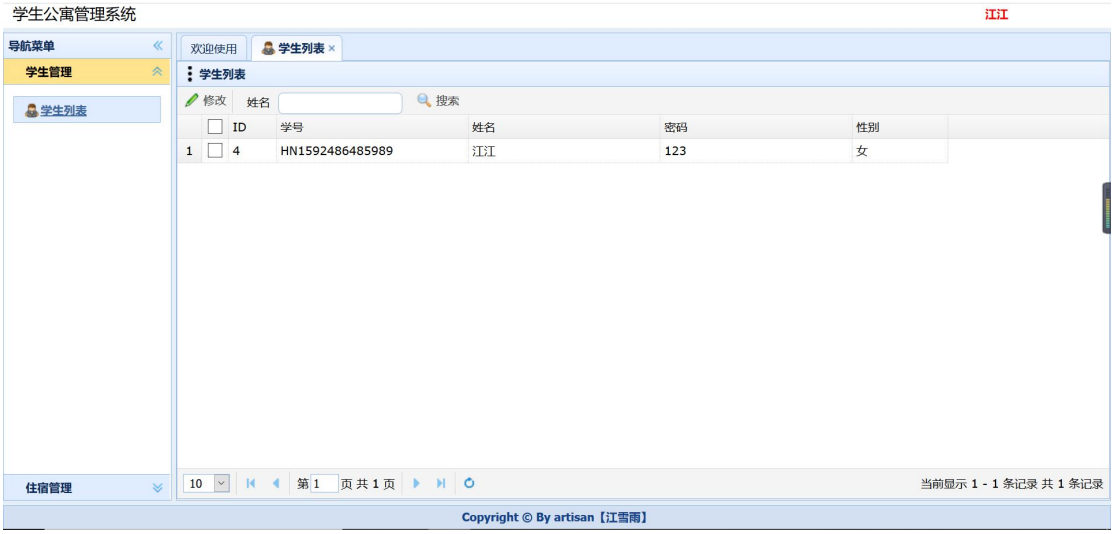
(6) 系统管理--超级管理员列表



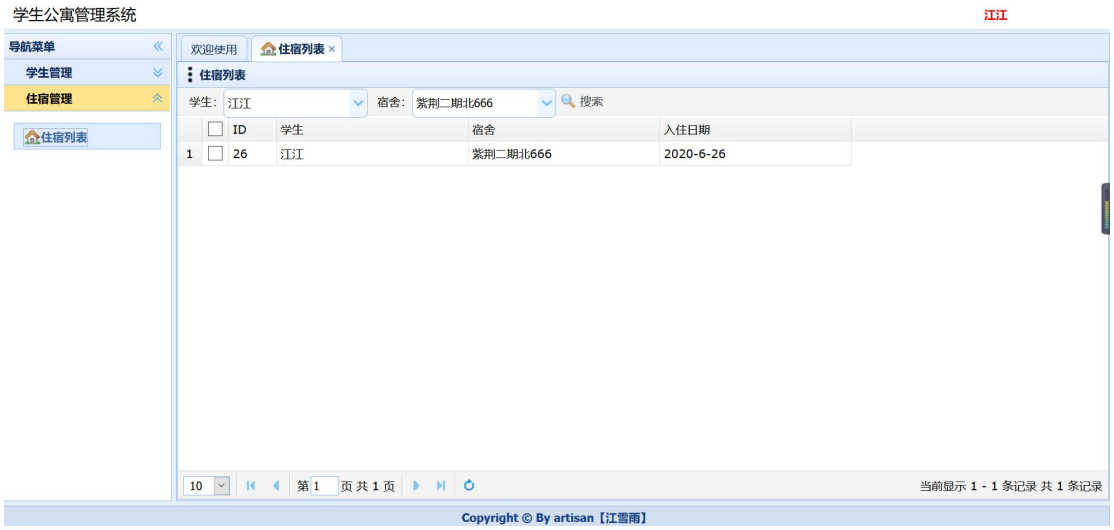
6.3.2 登陆界面--以学生身份登陆



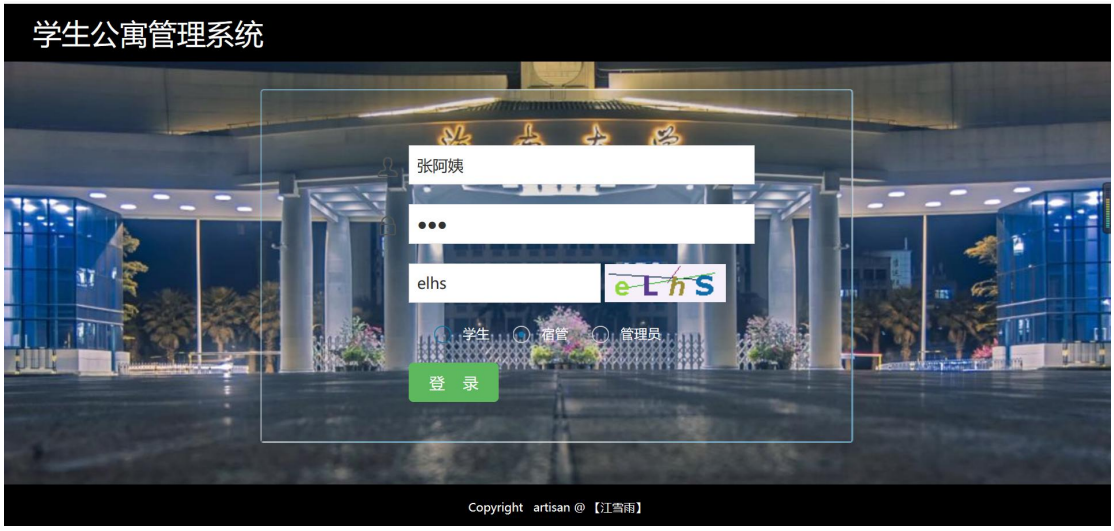
(1) 学生管理--学生列表



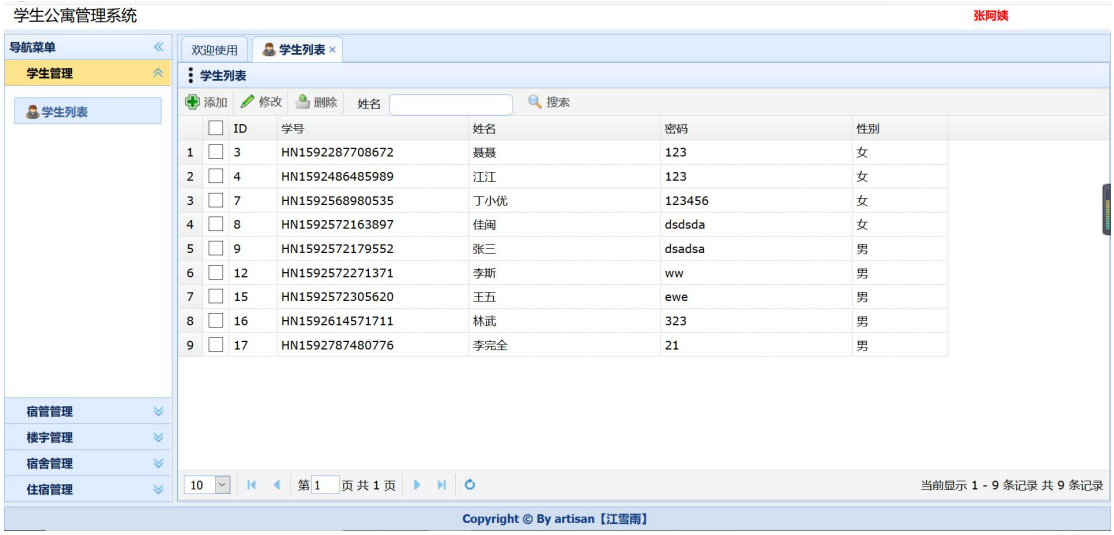
(2) 住宿管理--住宿列表



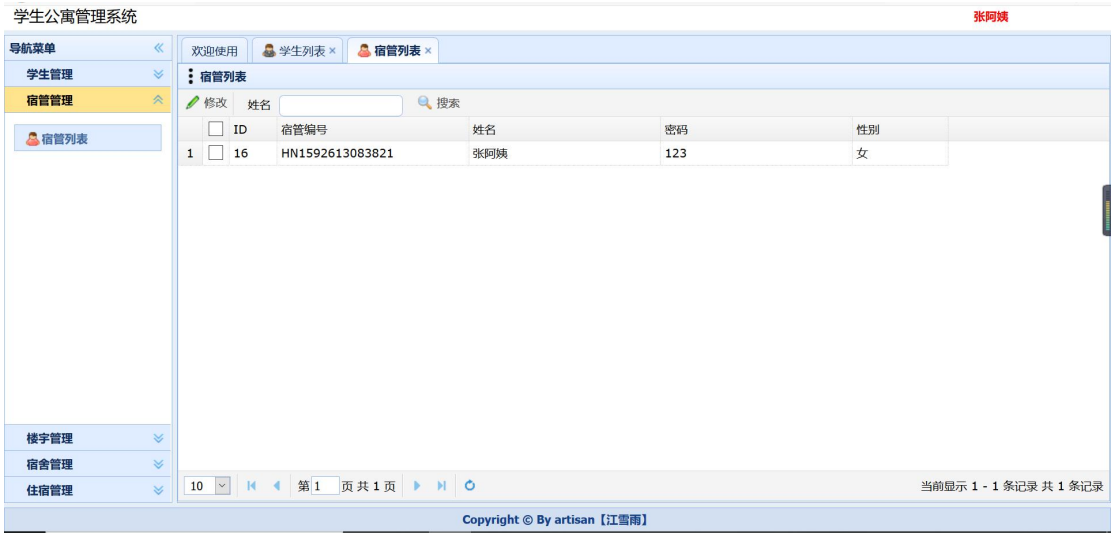
6.3.3 登陆界面--以宿管身份登陆



(1) 学生管理--学生列表

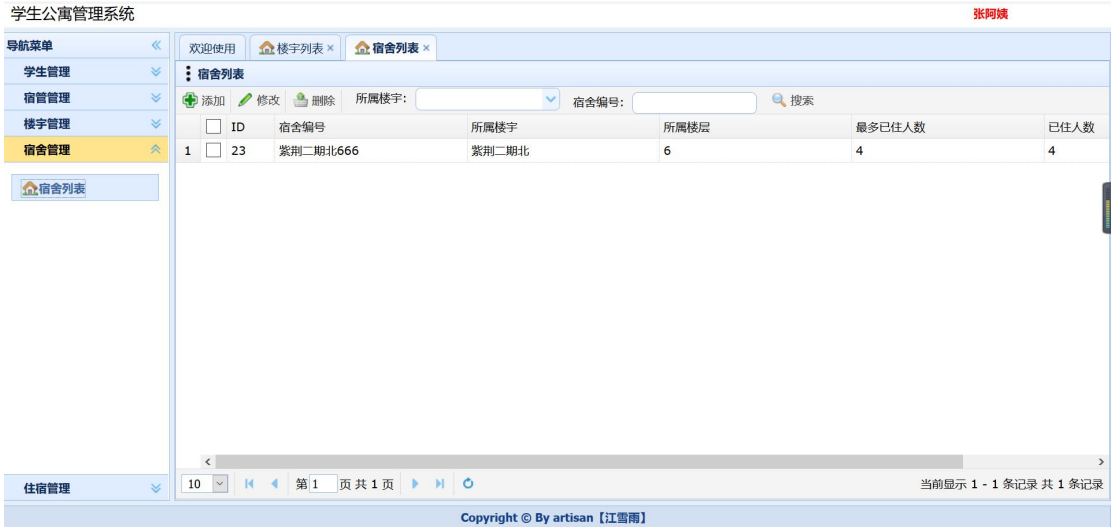


(2) 宿管管理--宿管列表

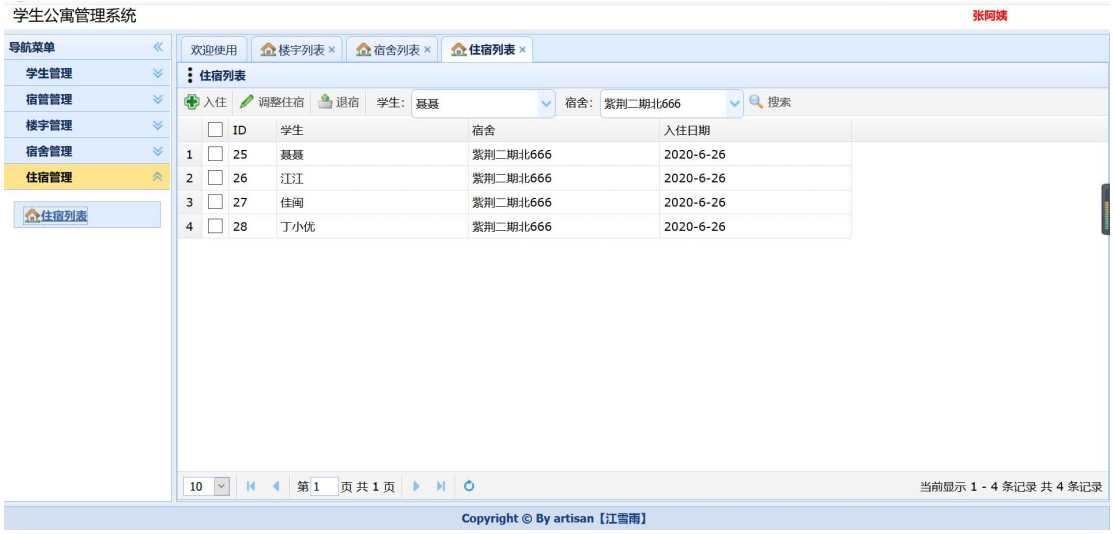


(3) 楼宇管理--楼宇列表

(4) 宿舍管理--宿舍列表



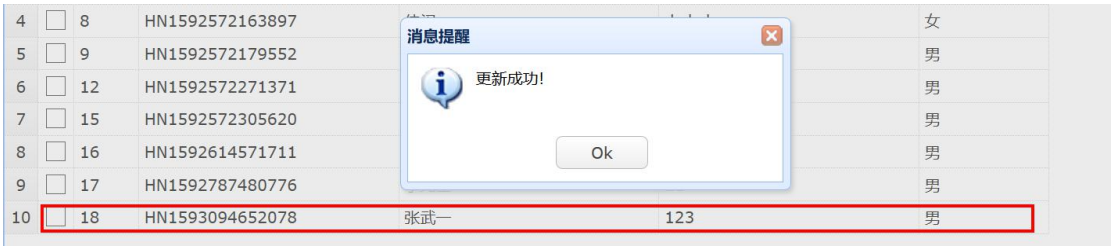
(5) 住宿管理--住宿列表



6.4 系统调试与测试

6.4.1 添加页面

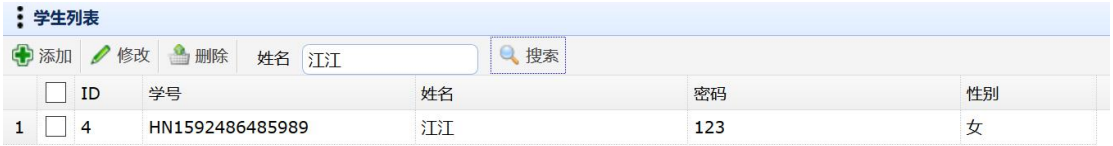
6.4.2 修改页面



6.4.3 删除页面

6.4.3 查询页面

	ID	学号	姓名	密码	性别
1	3	HN1592287708672	聂聂	123	女
2	4	HN1592486485989	江江	123	女
3	7	HN1592568980535	丁小优	123456	女
4	8	HN1592572163897	佳闽	dsdsda	女
5	9	HN1592572179552	张三	dsadsa	男
6	12	HN1592572271371	李斯	ww	男
7	15	HN1592572305620	王五	ewe	男
8	16	HN1592614571711	林武	323	男
9	17	HN1592787480776	李完全	21	男



	ID	学号	姓名	密码	性别
1	4	HN1592486485989	江江	123	女

6.5 各功能代码设计

学生宿舍管理系统总体上来讲是实现了不同状态（权限）下，对数据库中不同信息的增、删、改、查以及相应的前端页面设计。下面分别展示不同阶段重点部分的代码：

6.5.1 登陆页面

```
<title>登录|学生公寓管理系统</title>
<meta name="keywords" content="学生公寓管理系统">
</head>
<body>
<div class="header" style="padding: 0;">
    <h2 style="color: white; width: 400px; height: 60px; line-height: 60px;
margin: 0 0 0 30px; padding: 0;">学生公寓管理系统</h2>
</div>
<div class="loginWraper">
  <div id="loginform" class="loginBox">
```

```
    <form id="form" class="form form-horizontal" method="post">
      <div class="row cl">
        <label class="form-label col-3"><i
class="Hui-iconfont">&#xe60d;</i></label>
        <div class="formControls col-8">
          <input id="login-name" name="name" type="text" placeholder="账号"
class="input-text size-L">
        </div>
      </div>
      <div class="row cl">
        <label class="form-label col-3"><i
class="Hui-iconfont">&#xe60e;</i></label>
        <div class="formControls col-8">
          <input id="login-password" name="password" type="password"
placeholder="密码" class="input-text size-L">
        </div>
      </div>
      <div class="row cl">
        <div class="formControls col-8 col-offset-3">
          <input id="login-vcode" class="input-text size-L" name="vcode"
type="text" placeholder="请输入验证码" style="width: 200px;">
          <img title="点击图片切换验证码" id="vcodeImg"
src="CpachaServlet?method=loginCpacha&vl=4&fs=28"></div>
      </div>

      <div class="mt-20 skin-minimal" style="text-align: center; color:white">
        <div class="radio-box">
```

```
            <input type="radio" id="radio-2" name="type" checked value="2" />
            <label for="radio-1">学生</label>
        </div>
        <div class="radio-box">
            <input type="radio" id="radio-3" name="type" value="3" />
            <label for="radio-2">宿管</label>
        </div>
        <div class="radio-box">
            <input type="radio" id="radio-1" name="type" value="1" />
            <label for="radio-3">管理员</label>
        </div>
    </div>

      <div class="row">
        <div class="formControls col-8 col-offset-3">
          <input id="submitBtn" type="button" class="btn btn-success radius
size-L" value=" 登    录 ">
        </div>
      </div>
    </form>
  </div>
</div>
<div class="footer">Copyright   artisan @ 【江雪雨】</div>
</body>

</html>
```

6.5.2 数据库操作基础类

数据库操作基础类， 利用泛型和反射机制来抽象数据库基本的增删该查操作。

所有新增插入操作抽象封装代码如下：

```
public boolean add(T t) {
    if(t == null)
        return false;
   String buildSql = buildSql(CURD_ADD);
   System.out.println(buildSql);
   //sql 语句已生成 接下来的 sql 语句
      try {
       PreparedStatement prepareStatement = con.prepareStatement(buildSql);
       Field[] fields = t.getClass().getDeclaredFields();
       for(int i=1;i<fields.length;++i) {
           fields[i].setAccessible(true);//调用 setAccessible()方法，我们可以
实现对 student 私有属性字段的操作
           prepareStatement.setObject(i,fields[i].get(t));
       }
       return prepareStatement.executeUpdate()>0;
      }
      catch(Exception e) {
       e.printStackTrace();
      }

    return false;//默认返回 flase
    }
```

所有修改操作抽象封装代码如下：

```
 public boolean update(T t) {
     String sql = buildSql(CURD_UPDATE);
     try {
            PreparedStatement prepareStatement = con.prepareStatement(sql);
            Field[] declaredFields = this.t.getDeclaredFields();
            for( int i=1;i<declaredFields.length;i++) {
            declaredFields[i].setAccessible(true);
            prepareStatement.setObject(i,declaredFields[i].get(t));//获取字
段值
            }
            declaredFields[0].setAccessible(true);//设置可获取权限

prepareStatement.setObject(declaredFields.length,declaredFields[0].get(t));
        return prepareStatement.executeUpdate()>0;
     } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
     return false;
     }
```

所有删除操作抽象封装代码如下：

```
  public boolean delete (String[] ids) {
     String sql = buildSql(CURD_DELETE) + StringUtils.join(ids,",")+")";
         try {
            PreparedStatement prepareStatement = con.prepareStatement(sql);
```

```
            return prepareStatement.executeUpdate()>0;
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    return false;
    }
```

抽象封装所有的分页查询列表操作代码如下：

```
public Page<T> findList(Page<T> page) {

    //获取查询语句
    String sql = buildSql(CURD_SELECT);
    sql += buildSearchSql(page);
    sql += " limit " + page.getOffset() + "," + page.getPageSize();
    //处理 sql 语句
    try {
        PreparedStatement prepareStatement = con.prepareStatement(sql);
        prepareStatement = setParams(page,prepareStatement);
        ResultSet executeQuery = prepareStatement.executeQuery();
        List<T> conten = page.getConten();
        while(executeQuery.next()) {
            T entity = t.newInstance();//newInstance() 是 java 反射框架中类对象(Class)创建新对象的方法
            Field[] declaredFields = t.getDeclaredFields();
            for(Field field:declaredFields) {
             field.setAccessible(true);
```

```
            field.set(entity,
executeQuery.getObject(StringUtli.convertToUnderLine(field.getName())));//
给实体字段设置值（从数据库出来的）

            }
            conten.add(entity);
        }
        page.setConten(conten);
    }
    catch(Exception e) {
        e.printStackTrace();
    }

    page.setTotal(getTotal(page));
    return page;
    }
```

获取符合条件的所有记录数代码如下：

```
 public int getTotal(Page<T> page)  {
     String sql = buildSql(CURD_COUNT);
     sql += buildSearchSql(page);

     try {
        PreparedStatement prepareStatement = con.prepareStatement(sql);
            prepareStatement = setParams(page,prepareStatement);
            ResultSet executeQuery = prepareStatement.executeQuery();
            if(executeQuery.next()) {
```

```
                return executeQuery.getInt("total");
            }

    } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    return 0;
    }
```

构造查询 sql 语句代码如下：

```
  private String buildSearchSql(Page<T> page) {

    String sql = "";
   //判断有没有条件查询 进行遍历
    List<SearchProperty> searchProperties = page.getSearchProperties();
    for(SearchProperty searchProperty:searchProperties ) {

        switch(searchProperty.getOperator()) {
        case GT:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" > ?";// 记得加上占
位符
            break;
        }
        case GTE:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" >= ?";
```

```
            break;
        }
        case EQ:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" = ?";
            break;
        }
        case LT:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" < ?";
            break;
        }
        case LTE:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" <= ?";
            break;
        }
        case LIKE:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" like ?";
            break;
        }
        case NEQ:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" <> ?";
            break;
        }
```

```
        case in:{
            sql += " and "+
StringUtli.convertToUnderLine(searchProperty.getKey()) +" in
("+searchProperty.getValue()+")";
            break;
        }
    }
    }
     sql = sql.replaceFirst("and", "where");
     System.out.println(sql);
     return sql;
     }
```

构造一般查询语句代码如下：

```
private String buildSql(int type) {
        // TODO Auto-generated method stub
        String sql = "";
        switch(type) {
        case  CURD_ADD:{
        String sql1 = "insert into " +
StringUtli.convertToUnderLine(t.getSimpleName())+"(";//表名：类名
        Field[] declaredFields = t.getDeclaredFields();//getDeclaredFields()
返回 Class 中所有的字段，包括私有字段

        //循环输出[反射：Field 类--获取类中 public 类型的属性]
        for(Field fields:declaredFields) {
            sql1 +=StringUtli.convertToUnderLine(fields.getName())+",";
        }
```

```
        sql1 = sql1.substring(0, sql1.length()-1) + ")";//截掉一个逗号;
        String sql2 = "values (null,";
        String[] params = new String[declaredFields.length-1];
        Arrays.fill(params, "?");
        sql2 += StringUtils.join(params,",")+")";
        sql = sql1 + sql2;

        break;
        }

        case  CURD_SELECT:{
        sql = "select * from " +
StringUtli.convertToUnderLine(t.getSimpleName());//获取数据表的名称
        break;
        }

        case  CURD_COUNT:{
        sql = "select count(*) as total from " +
StringUtli.convertToUnderLine(t.getSimpleName());
        break;
        }

        case  CURD_UPDATE:{
        sql = "update " + StringUtli.convertToUnderLine(t.getSimpleName()) +
" set ";
        Field[] declaredFields = t.getDeclaredFields();
        for (Field field:declaredFields) {
```

```
            if(!"id".equals(field.getName())) {
                sql += StringUtli.convertToUnderLine(field.getName()) + "=?,";
            }
        }
        sql = sql.substring(0,sql.length()-1) + "    where id = ? ";
        break;
        }

        case  CURD_DELETE:{
        sql = "delete from " + StringUtli.convertToUnderLine(t.getSimpleName())
+ " where id in (";
        break;
        }


        default:
        break;
        }
        System.out.println(sql);
        return sql;
          }
```

7. 总结

在课程设计期间，我遇到了许许多多的问题，一一去请教同学，也不断查资料去汲取更多的新知识，如防止 sql 注入，数据库底层封装，数据库的连接失败，session 的使用，登陆页面的重定位等等，发现问题并解决问题的过程，使得我探索并解决问题的能力有了一个提高。这整个课程设计过程中，使我对程序编写的整个过程有了一个统筹全局的思想，因为课设的需求分析、程序编写、程序调试、撰写报告这些过程是环环相扣的，下面是一些部分收获：

1. 底层数据库的操作是放在 BaseDao 中,里面都是对应的增删改查操作，每个实体都有一个对应的 Dao，各实体的 Dao 继承 BaseDao，在 servlet 中负责获取参数，判断数据的合理性和进行跳转，当条件符合的时候则调用对应实体的 Dao 操作进行增删改查，然接着把结果集传回页面，之后便是在页面进行循环把数据展示出来。但是由于模块信息的问题，需要用到另外一张表的信息的时候，再把对应实体的 Dao 进行初始化，再调用它的数据库操作来获取数据。

2. 数据库连接，获得 Connection 是放在构造函数里的，创建一个对象就获取一个 Connection，一开始没留意到这个问题，多运行几次，创建的 Dao 多了,产生很多没有关闭的 Connection，没有注意到关闭数据库。

3. 解决了 java 程序和数据库的中文乱码以及程序中文件读写、网络传输中中文乱码的问题。在基于 java 的编程中，经常会碰到汉字的处里及显示的问题，比如一大堆乱码或问号。这是因为 java 中默认的编码方式是 UNICODE，而我们通常使用的文件和 DB 都是基于 GB2312 或者 BIG5 等编码,所以会出现这样的问题。通过设置 resp.setCharacterEncoDing(“utf-8”)便可以解决解决问题了。

4. 在使用 Dbutils 之前，我们 Dao 层使用的技术是 JDBC，JDBC 的弊端：

（1）数据库链接对象、sql 语句操作对象，封装结果集对象，这三大对象会重复定义

（2）封装数据的代码重复，而且操作复杂，代码量大

（3）释放资源的代码重复

使用使用 Dbutils 之后，封装了 JDBC 的代码，简化开发人员对 dao 层的操作。

框架的作用：能够帮助程序员，提高程序的开发效率。

5. java 程序都是通过 JDBC（Java Data Base Connectivity）来访问数据库的。JDBC 定义了一系列的接口规范，具体的实现是由各数据库厂商去实现，是一种典型的桥接模式。

JDBC 的工作量大：需要先注册驱动和数据库信息、操作 Connection、通过 statement 对象执行 SQL，将结果返回给 resultSet，然后从 resultSet 中读取数据并转换为 pojo 对象，最后需要关闭数据库相关资源。而且还需要自己对 JDBC 过程的异常进行捕捉和处理。

6. 在开始使用 EasyUI 之前，要认清楚一个原则，因为 EasyUI 是封装的 CSS 及 JS 库，所以在修改 EasyUI 组件的样式和功能时，最好是采用 EasyUI 封装后的 CSS 样式和 JS 方法，而不是自己去写 CSS 和 JS，这样不容易引起冲突，样式统一性也得到了保证。