信息工程学院

**数据库系统开发与应用课程设计**

题 目： 小区信息管理系统

学生姓名：

学 号：

专业班级： 计算机161

任课教师：

完成时间： 2018年12月10日

摘要

随着我国经济的迅速发展，城镇化的推广普及和各种购房政策及福利制度的实施，住宅式小区已经成为了人们选择住处的首选。与此同时，小区的规模和数量也在不断地扩大和增大，小区住户信息的统计与管理便成了很大的问题。在海量的住户信息面前，利用计算机技术来解决海量数据的相关问题已是大势所趋。在此背景下，经数据调查，需求分析，系统设计和系统实现后，我们采用使用Spring MVC 作为项目的基本框架，整个系统在Guns开源框架上进行业务逻辑的编写。以IntelliJ Idea作为为开发工具设计管理端Web程序，管理端Web程序提供管理员对小区各项资源的管理，运用云服务器上的MySQL作为后台数据库，使用Navicat进行可视化连接，设计并开发了功能完善，操作简便，界面美观，高效实用，可靠安全的小区信息管理系统。本系统主要完成以下工作：

1. 小区管理员可以登记管理小区住户的住房情况，登记小区的各个楼栋，楼栋的各个单元，单元的各个房间的情况。
2. 小区管理员可以向用户发布通知，管理已发布的通知。
3. 小区管理员查看报修或投诉信息，并解决报修或投诉。
4. 小区管理员可以登记车位信息，查看车位租用状况。
5. 小区管理员可以查看关于性别比例，年龄分布，租借情况的统计信息。
6. 系统为手机端提供Api接口。

**关键词：小区信息、管理系统、IntelliJ Idea、MySQL、SpringMVC**

目录

[第一章 需求分析 1](#_Toc533017488)

[1.1需求分析 1](#_Toc533017489)

[1.1.1小区资源的基本信息 1](#_Toc533017490)

[1.1.2小区住户的基本信息 1](#_Toc533017491)

[1.1.3用户对系统的要求 2](#_Toc533017492)

[1.2 系统功能的设计和划分 4](#_Toc533017493)

[1.3数据流图 5](#_Toc533017494)

[1.3.1各子系统的分数据流图 5](#_Toc533017495)

[1.3.2总数据流图 8](#_Toc533017496)

[1.4 数据字典 9](#_Toc533017497)

[1.4.1 数据项 9](#_Toc533017498)

[1.4.2 数据结构 11](#_Toc533017499)

[1.4.3 数据流 12](#_Toc533017500)

[1.4.4 数据存储 13](#_Toc533017501)

[1.4.5 处理过程 13](#_Toc533017502)

[第二章 概念结构设计 16](#_Toc533017503)

[2.1 各实体属性的ER图 16](#_Toc533017504)

[2.2 总ER图 18](#_Toc533017505)

[第三章 逻辑结构设计 19](#_Toc533017506)

[3.1 将E-R图转换为关系模型 19](#_Toc533017507)

[3.1.1原实体对应的关系模型如下： 19](#_Toc533017508)

[3.1.2关系模型规范化 19](#_Toc533017509)

[3.1.3处理后的所有关系模型 19](#_Toc533017510)

[第四章 物理设计及数据库实施 20](#_Toc533017511)

[4.1 基本表设计与创建 20](#_Toc533017512)

[4.2视图 22](#_Toc533017513)

[4.3索引 23](#_Toc533017514)

[4.5 触发器 24](#_Toc533017515)

[4.6 数据库的关系图 25](#_Toc533017516)

[第五章 系统的设计与实现 25](#_Toc533017517)

[5.1 系统框架简介 25](#_Toc533017518)

[5.2 业务的实现 26](#_Toc533017519)

[5.2.1 信息列表的显示 27](#_Toc533017520)

[5.2.2 新增信息 30](#_Toc533017521)

[5.2.3 修改信息 33](#_Toc533017522)

[5.2.4 删除信息 35](#_Toc533017523)

[5.3 数据统计功能的实现 35](#_Toc533017524)

[5.3.1 男女比例与年龄分布 35](#_Toc533017525)

[5.3.2 车位租借次数与租借金额 40](#_Toc533017526)

[5.4 API接口的实现 43](#_Toc533017527)

[5.4.1 接口的安全性 43](#_Toc533017528)

[5.4.2 主要接口的实现 43](#_Toc533017529)

[第六章 结论和展望 46](#_Toc533017530)

[参考文献 47](#_Toc533017531)

# 第一章 需求分析

1.1需求分析

本系统的最终用户为系统管理员和小区住户。该系统的成功开发，将可以巨大便利系统管理员对住户信息的统计与管理。根据查询资料及对相关人员的咨询，我们讨论分析，得出小区管理员和小区住户对本系统的实际要求：

1.1.1小区资源的基本信息

小区管理员应对小区可用资源得信息进行及时地完善和更新，对小区内所有住房的居住情况和车位的租借情况统计整理，并向有需求的住户提供有效信息和便利服务。

1. 住房居住的基本信息：

小区管理员应对小区内所有住房被使用情况进行实时地完善与更新，以方便有需求的住户使用。这些住房信息应该包括住房号码，住房状态（使用状态和空闲状态），住房地址，住房所属区域。

1. 车位租借的基本信息：

小区管理员应对小区内的所有车位租借情况进行实时地完善与更新，以方便有需求的住户租借。车位租借的情况应包括车位号码，车位地址，租借状态（使用状态和空闲状态）

1.1.2小区住户的基本信息

当住户入住该小区后，通过手机端APP，登记住户个人信息，以方便小区管理员对住户进行管理，与住户进行必要的业务交流。住户需要登记以下信息。

1. 住户登录的基本信息：

登录手机APP时，每位住户通过注册自己的手机号码和登录密码来获得小区 服务的权限，而系统会为每位住户指定一个唯一的用户ID，用于后台信息的 统计和管理。

1. 小区住户的基本信息：

每位小区住户都具有必要注册的个人信息，以保证小区内住户个人信息的完 整性，确保小区住户们的安全，其中包括系统为每位住户指定的唯一ID，身 份证号码，姓名，性别，手机号码，年龄，民族，职业，住房号码，登录APP 密码。

1.1.3用户对系统的要求

**一、小区住户：  
 a.信息要求：**

小区住户需要及时得到小区空闲住房的信息，可租借车位的信息，租借车位需要租借时间，租借时长，支付费用等信息，需要得到小区官方的公告通知，需要对自己提交的投诉，报修或建议等事务查询官方回复。

**b.处理要求：**

1. 注册和维护自己的登录信息和个人的基本信息。并要求因注册手机号的不同，系统为其再指定一个用于后台数据库操作的唯一ID。
2. 注册个人信息时，通过获得小区管理员提供的未被注册的空闲住房，选定其中一个注册。小区管理员也可以根据这一操作，既是录入信息并更新住房信息。
3. 住户可以随时更新公告信息，得到小区管理员的通知。
4. 住户可以提出投诉，报修和建议等事务请求。提交事务后，便可以随时查看提交事务的实时信息及官方回复情况。
5. 住户可以根据小区管理员提供的剩余车位信息，并租借一个想要的车位。系统将会提供该车位的信息，住户也可以停止租借该车位，同时支付必要的租用车位的费用。
6. **安全性与完整性要求：**
7. 安全性要求：
8. 系统应设计根据手机端APP提交的手机号和对应的密码鉴别是否是合法的用户，若不是，应拒接登陆请求。在住户注册个人的同时，应提醒注册登录密码信息。
9. 每当手机端APP向服务器API发出数据库操作请求时，都需要提前申请token秘钥，以保证每次操作都是合理的。
10. 完整性要求：
    1. 根据实体完整性的要求，各种主要信息记录要保持完整性，即内容不能为空；
    2. 各种数据间相互的联系要保持正确性；
    3. 相同的数据在不同记录中要保持一致性。

**二、小区管理员：**

**a.信息要求：**

小区管理员应需要录入小区可用资源的所有必要信息，例如空闲住房和空闲车位等，可以查看小区所有住户的个人信息和登录APP信息，还需要根据小区的实际情况，具有可公告的信息，需要可以接收小区住户的投诉，报修建议等事务信息，并具有及时回复的信息。

**b.处理要求：**

1. 创建和维护小区管理员的登录信息。
2. 根据小区住户提交的注册请求，创建和维护住户们的登录信息和个人信息。
3. 发布公告信息，根据小区的实时情况，通知小区住户们必要的生活信息。
4. 提供住房被使用的信息。在小区住户们注册个人信息时，需根据小区管理员提供的住房信息，注册未被正在使用的住房进行注册操作，小区管理员可根据住户操作及时更新小区住房的情况。
5. 提供车位租借情况的信息。小区应向住户们提供车位正被使用的情况信息，以方便住户们租借空闲车位。小区管理员在接收到住户们的租借车位请求时，需要及时更新小区车位信息，以方便他人租借。住户可以随时停止租借车位，同时需要支付租借车位所需要的费用，系统也会提供给住户本次租借车位操作的信息。

**c.安全性要求：**

1. 系统应要求小区管理员登录时，需要验证登录账号和登录密码正确且对应，以确保整个系统的安全性。
2. 住户每次通过手机APP先服务器API发出操作请求时，所需要的token秘钥，需要小区管理员授权与分配，并实时更新授权秘钥。

1.2 系统功能的设计和划分

根据如上得到的用户需求，我们将本系统按照所完成的功能分成以下两部分：

**一、用户部分**

* 1. 注册个人信息
  2. 根据手机号与登录密码登录
  3. 发起投诉或报修
  4. 查看发起的投诉或报修
  5. 查看剩余车位的数量
  6. 租用车位
  7. 结束租用车位
  8. 获得最新的一条公告

**二、管理员部分**

* 1. 注册新的管理员
  2. 根据账号密码登录管理平台  
     登记查看修改删除小区人员信息
  3. 登记查看修改删除小区租房信息
  4. 登记查看修改删除车位信息
  5. 解决报修或投诉
  6. 查看租借信息
  7. 发布查看修改删除通知信息
  8. 查看统计信息

1.3数据流图

1.3.1各子系统的分数据流图

**1.**房间信息分数据流图如图1-1所示：

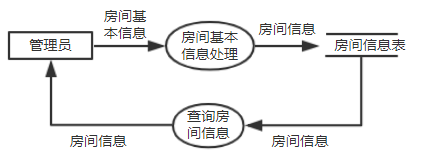


图1-1 房间信息分数据流图

**2.**用户信息分数据流图如图1-2所示：

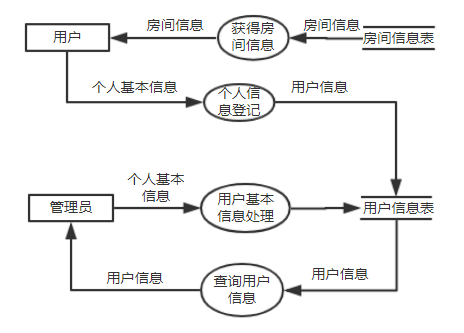


图1-2 用户信息分数据流图

**3.**通知信息分数据流图如图1-2所示：

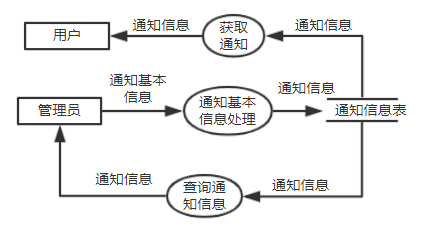


图1-3 通知信息分数据流图

**4．**车位信息分数据流图如图1-4所示：

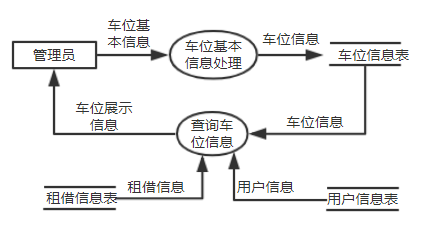


图1-4 车位信息分数据流图

**5．**租借信息分数据流图如图1-5所示：

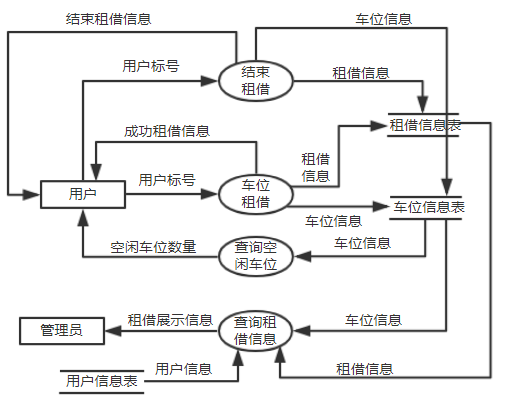


图1-5 租借信息分数据流图

**6．**报修投诉信息分数据流图如图1-6所示：

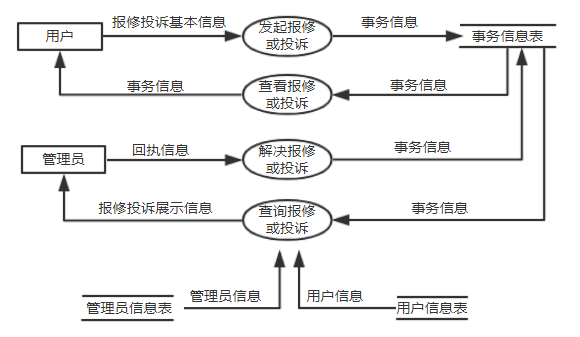


图1-6 报修投诉信息分数据流图

**7．**管理员信息分数据流图如图1-7所示：

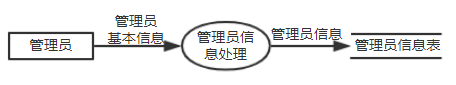


图1-7 管理员信息分数据流图

1.3.2总数据流图

总数据流图如图1-8所示：

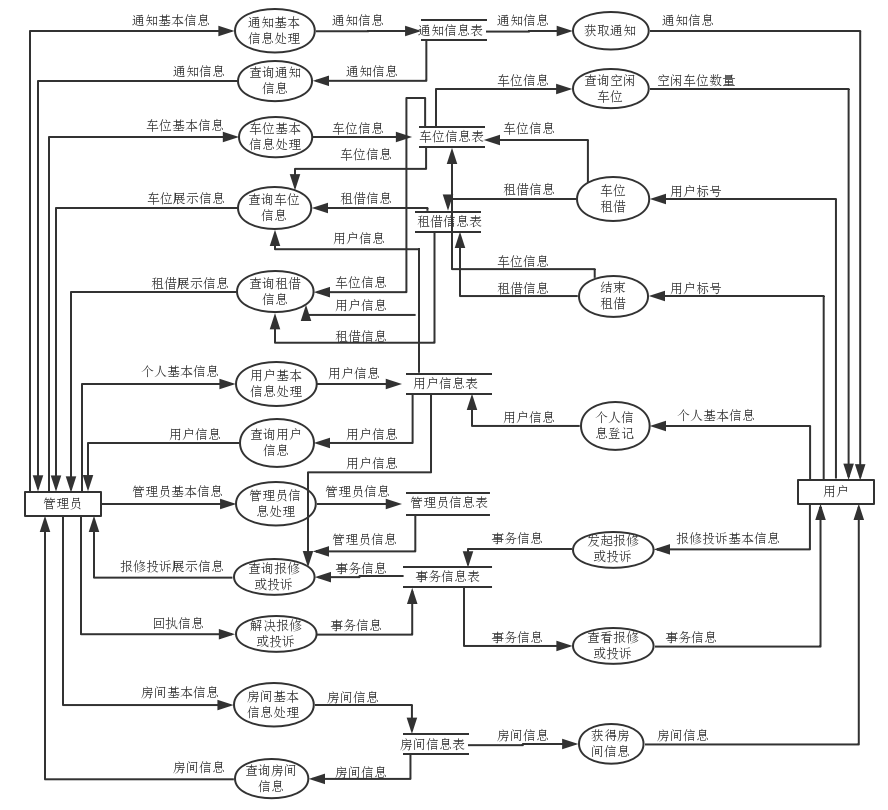


图1-8 总数据流图

1.4 数据字典

### 1.4.1 数据项

1. 用户信息数据项如表1-1所示：

表1-1 用户信息数据项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 别名 | 属性名 | 类型 | 宽度 | 默认值 | 允许非空 | 备注 |
| id | 用户标号 | int | 11 |  | No | 主键（自增） |
| room\_id | 房间标号 | int | 11 |  | No | 外键（room\_info） |
| id\_number | 身份证号码 | varchar | 18 |  | No |  |
| name | 姓名 | varchar | 20 |  | No |  |
| gender | 性别 | tinyint | 1 |  | No | 1代表男，0代码女 |
| phone | 电话 | varchar | 11 |  | No |  |
| age | 年龄 | int | 3 |  | NO |  |
| race | 民族 | varchar | 255 | NULL | YES |  |
| job | 职业 | varchar | 255 | NULL | YES |  |
| passwd | 手机登录密码 | varchar | 25 | null | No |  |

1. 通知信息数据项如表1-2所示：

表1-2 通知信息数据项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 别名 | 属性名 | 类型 | 宽度 | 默认值 | 允许非空 | 备注 |
| id | 通知标号 | int | 11 |  | NO | 主键，自增 |
| creater | 管理员标号 | int | 11 |  | NO |  |
| create\_time | 创建时间 | datetime |  |  | NO |  |
| title | 通知标题 | varchar | 255 |  | NO |  |
| content | 通知内容 | text |  |  | NO |  |

1. 房间信息数据项如表1-3所示：

表1-3 房间信息数据项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 别名 | 属性名 | 类型 | 宽度 | 默认值 | 允许非空 | 备注 |
| room\_id | 房间标号 | int | 11 |  | NO | 主键，自增 |
| type | 房间类型 | tinyint | 1 |  | NO | 1代表具体某房间，0代表某个区域 |
| name | 房间名 | varchar | 255 |  | NO |  |
| address | 房间地址 | varchar | 255 |  | NO | 具体地址 |
| Parent\_id | 房间父节点 | int | 11 |  | NO | 父节点 |

1. 车位信息数据项如表1-4所示：

表1-4 车位信息数据项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 别名 | 属性名 | 类型 | 宽度 | 默认值 | 允许非空 | 备注 |
| carport\_id | 车位标号 | int | 11 |  | NO | 主键，自增 |
| Carport\_loc | 车位地址 | varchar | 255 |  | NO |  |
| is\_used | 车位状态 | tinyint | 1 |  | No | 1为正在使用，0为未使用 |

1. 事务信息数据项如表1-5所示：

表1-5 事务信息数据项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 别名 | 属性名 | 类型 | 宽度 | 默认值 | 允许非空 | 备注 |
| affair\_id | 事务标号 | int | 11 |  | NO | 主键，自增 |
| id\_number | 用户标号 | int | 11 |  | NO | 外键（user\_info） |
| admin\_id | 管理员标号 | Int | 11 |  | Yes | 外键 |
| create\_time | 事务请求时间 | date |  |  | NO |  |
| content | 事务内容 | text |  |  | NO |  |
| title | 事务标题 | varchar | 255 |  | NO |  |
| receipt | 回执内容 | text |  |  | YES |  |
| solved | 是否解决 | tinyint | 1 | 0 | NO | 1为解决，0未解决 |

1. 管理员信息数据项如表1-6所示：

表1-6 管理员信息数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 说明 | 类型 | 宽度 | 默认值 | 允许非空 | 备注 |
| admin\_id | 管理员标号 | int | 11 |  | NO | 主键自增 |
| account | 管理员账号 | Varchar | 45 |  | NO |  |
| passwd | 管理员密码 | Varchar | 45 |  | NO |  |
| name | 管理员姓名 | varchar | 45 |  | NO |  |

1. 租借信息数据项如表1-7所示：

表1-7 租借信息数据项

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 说明 | 类型 | 宽度 | 默认值 | 允许非空 | 备注 |
| carport\_id | 车位标号 | int | 11 |  | NO | 主键，外键（carport\_info） |
| id\_number | 用户标号 | varcahr | 11 |  | NO | 主键，外键（user\_info） |
| start\_time | 开始租借时间 | datetime |  |  | NO | 主键 |
| end\_time | 结束租借时间 | datetime |  | NULL | YES |  |
| payment | 租借金额 | float | 255 | NULL | YES |  |
| state | 租借状态 | tinyint | 1 | 0 | NO | 0为租借未完成，1位租借完成 |

### 1.4.2 数据结构

各个数据结构的组成如表1-8示：

表1-8 各个数据结构的组成

|  |  |
| --- | --- |
| 数据结构名 | 组成 |
| 房间基本信息 | 房间名、房间父节点、房间类型 |
| 房间信息 | 房间标号、房间地址、房间名、房间父节点、房间类型 |
| 个人基本信息 | 身份证号码、房间标号、姓名、性别、电话、年龄、民族、职业、手机登录密码 |
| 用户信息 | 用户标号、身份证号码、房间标号、姓名、性别、电话、年龄、民族、职业、手机登录密码 |
| 通知基本信息 | 通知标题、通知内容 |
| 通知信息 | 通知标号、管理员标号、创建时间、通知标题、通知内容 |
| 车位基本信息 | 车位地址 |
| 车位信息 | 车位标号、车位地址、车位状态 |
| 车位展示信息 | 车位地址、车位状态、车位使用人姓名 |
| 用户标号 | 用户标号 |
| 空闲车位数量 | 空闲车位数量 |
| 租借信息 | 车位标号、用户标号、开始租借时间、结束租借时间、租借金额、租借状态 |
| 成功租借信息 | 车位地址、开始租借时间 |
| 结束租借信息 | 租借时长、租借金额 |
| 租借展示信息 | 车位地址、租借人、开始租借时间、结束租借时间、租金额、租借状态 |
| 管理员基本信息 | 管理员账号、管理员密码、管理员姓名 |
| 管理员信息 | 管理员标号、管理员账号、管理员密码、管理员姓名 |
| 报修投诉基本信息 | 事务标题、事务内容、用户标号 |
| 事务信息 | 事务标号、用户标号、管理员标号，事务请求时间、事务标题、事务内容、回执内容、是否解决 |
| 回执信息 | 回执内容 |
| 报修投诉展示信息 | 发起人姓名、解决人姓名、事务请求时间、事务标题、事务内容、回执内容、是否解决 |

### 1.4.3 数据流

关于数据流的分析如下表1-9示：

表1-9 关于数据流的分析

|  |  |  |  |
| --- | --- | --- | --- |
| 数据流名 | 数据流来源 | 数据流去向 | 组成 |
| 通知基本信息处理 | 管理员 | 通知信息表 | 通知信息 |
| 查询通知信息 | 通知信息表 | 管理员 | 通知信息 |
| 车位基本信息处理 | 管理员 | 车位信息表 | 车位信息 |
| 查询车位信息 | 车位信息表、租借信息表、用户信息表 | 管理员 | 车位展示信息 |
| 查询租借信息 | 车位信息表、租借信息表、用户信息表 | 管理员 | 租借展示信息 |
| 用户基本信息处理 | 管理员 | 用户信息表 | 用户信息 |
| 查询用户信息 | 用户信息表 | 管理员 | 用户信息 |
| 管理员信息处理 | 管理员 | 管理员信息表 | 管理员信息 |
| 查询报修或投诉 | 事务信息表、管理员信息表、用户信息表 | 管理员 | 报修投诉展示信息 |
| 解决报修或投诉 | 管理员 | 事务信息表 | 事务信息 |
| 房间基本信息处理 | 管理员 | 房间信息表 | 房间信息 |
| 查询房间信息 | 房间信息表 | 用户 | 房间信息 |
| 获取通知 | 通知信息表 | 用户 | 通知信息 |
| 查询空闲车位 | 车位信息表 | 用户 | 空闲车位数量 |
| 车位租借 | 用户 | 车位信息表、租借信息表 | 车位信息、租借信息 |
| 结束租借 | 用户 | 车位信息表、租借信息表 | 车位信息、租借信息 |
| 个人信息登记 | 用户 | 用户信息表 | 用户信息 |
| 发起报修或投诉 | 用户 | 事务信息表 | 事务信息 |
| 查看报修或投诉 | 事务信息表 | 用户 | 事务信息 |
| 获得房间信息 | 房间信息表 | 用户 | 房间信息 |

### 1.4.4 数据存储

数据存储的分析如下表1-10所示：

表1-10 数据存储的分析

|  |  |  |
| --- | --- | --- |
| 数据存储名 | 输入/出的数据流 | 组成 |
| 通知信息表 | 通知信息 | 通知信息 |
| 车位信息表 | 车位信息 | 车位信息 |
| 租借信息表 | 租借信息 | 租借信息 |
| 用户信息表 | 用户信息 | 用户信息 |
| 管理员信息表 | 管理员信息 | 管理员信息 |
| 事务信息表 | 事务信息 | 事务信息 |
| 房间信息表 | 房间信息 | 房间信息 |

### 1.4.5 处理过程

处理过程如下表1-14所示：

表1-14 处理过程

|  |  |  |  |
| --- | --- | --- | --- |
| 数据流名 | 输入数据流 | 输出数据流 | 处理 |
| 通知基本信息处理 | 通知基本信息 | 通知信息 | 管理员在Web端填入通知标题与内容，系统自动生成唯一的通知标号，管理员标号与创建时间。 |
| 查询通知信息 | 通知信息 | 通知信息 | 管理员可以在Web端可以根据通知标题查询到某一通知。 |
| 车位基本信息处理 | 管理员 | 车位信息 | 管理员在Web端填入车位地址，系统自动生成车位标号与车位状态，车位状态默认为0，标识未被租用。 |
| 查询车位信息 | 车位信息、租借信息、用户信息 | 车位展示信息 | 管理员在Web端可以查看车位展示信息。系统从车位信息、租借信息、用户信息三个表中查出车位地址、车位状态、车位使用人姓名展示给管理员。若此车位未被租用则车位使用人为空。 |
| 查询租借信息 | 车位信息、租借信息、用户信息 | 租借展示信息 | 管理员在Web端可以查看车位展示信息。系统从车位信息、租借信息、用户信息三个表中查出车位地址、租借人、开始租借时间、结束租借时间、租金额、租借状态展示给管理员。 |
| 用户基本信息处理 | 个人基本信息 | 用户信息 | 管理员在Web端可以填入个人基本信息，系统产生唯一的用户标号填入到用户信息表中。 |
| 查询用户信息 | 用户信息 | 用户信息 | 管理员在Web端可以通过姓名与身份证号模糊查询到某用户信息。 |
| 管理员信息处理 | 管理员基本信息 | 管理员信息 | 管理员在Web端可以填入管理员基本信息，系统添加为一的管理员标号。 |
| 查询报修或投诉 | 事务信息、管理员信息、用户信息 | 报修投诉展示信息 | 管理员在Web端可以查询报修或投诉，系统从事务信息表、管理员信息表、用户信息表中获得发起人姓名、解决人姓名、事务请求时间、事务标题、事务内容、回执内容、是否解决显示给管理员。 |
| 解决报修或投诉 | 回执信息 | 事务信息 | 管理员在Web端可以选中一条报修或投诉，点击解决报修或投诉按钮添加回执，系统将管理员的标号与回执内容添加到相应的事务信息中。 |
| 房间基本信息处理 | 房间基本信息 | 房间信息 | 管理员在Web端可以填入房间名称，选择房间父节点，选择房间类型，系统根据房间父节点的地址生成房间的地址、唯一的房间标号填入房间信息表中。 |
| 查询房间信息 | 房间信息 | 房间信息 | 管理员在Web端可以根据房间地址查询到某个房间信息。 |
| 获取通知 | 通知信息 | 通知信息 | 用户在手机端查询通知信息，用户可得到最新的一条通知信息。 |
| 查询空闲车位 | 车位信息 | 空闲车位数量 | 用户在手机端可以查询到系统通过统计车位信息的车位状态为0的车位总数量，计算出的空闲车位的数量。 |
| 车位租借 | 用户标号 | 车位信息、租借信息 | 用户在手机端通过用户标号可以发起车位租借请求。系统接收到请求后自动分配一个空闲的车位给用户，并将此车位状态设为被使用，生成一条租借信息，生成唯一的租借标号，将租借时间、用户标号与租借状态（正在租用）填入租借信息中。租借成功返回车位地址与租借时间给用户。 |
| 结束租借 | 用户标号 | 车位信息、租借信息 | 用户在手机端通过用户标号可以结束租借，若用户未租借车位则无法结束租借，用户租借了车位根据车位的开始租借时间与发起结束租借的时间之间的时长计算租用金额，每半个小时五元钱，之后将租借结束时间与租用金额与租借状态（租用完成）填入相应的租借信息表中。将此次租借对应的车位状态设为未被租用。成功后返回租用总时长与租用金额给用户。 |
| 个人信息登记 | 个人基本信息 | 用户信息 | 用户在手机端填入个人基本信息，经过系统处理生成唯一的用户标号。 |
| 发起报修或投诉 | 报修投诉基本信息、 | 事务信息 | 用户在手机端填入事务标题、事务内容、用户标号，系统通过这些信息生成唯一的事务标号与创建时间填入事务信息表中。 |
| 查看报修或投诉 | 事务信息 | 事务信息 | 用户在手机端可以查看自己发起的所有事务信息。 |
| 获得房间信息 | 房间信息 | 房间信息 | 用户在手机端可以请求房间类型为房间的房间信息，便于用户选择自己居住的房间。 |

# 第二章 概念结构设计

根据需求分析说明书设计ER图

2.1 各实体属性的ER图

**1.**小区住户实体的ER图如图2-1所示：

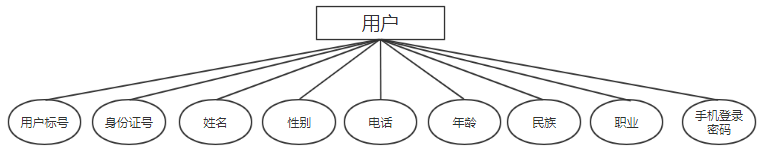


图2-1 小区住户实体的ER图

**2.**通知实体属性的ER图如图2-2所示：

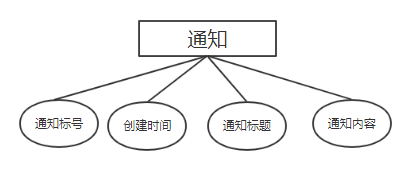


图2-2 通知实体属性的ER图

**3.**房间实体属性的ER图如图2-3所示：

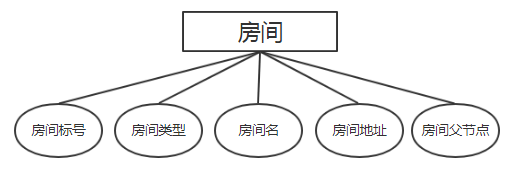


图2-3 **.**房间实体属性的ER图

**4.**车位实体属性的ER图如图2-4所示：

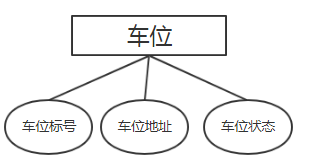


图2-4 车位实体属性的ER图

**5.**事务实体属性的ER图如图2-5所示：

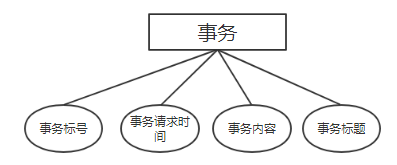


图2-5 事务实体属性的ER图

**6.**管理员实体属性的ER图如图2-7所示：

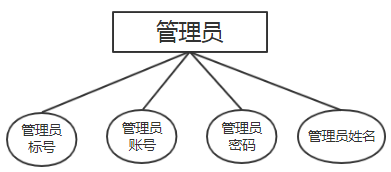


图2-6 管理员实体属性的ER图

2.2 总ER图

设计完成的总ER图如图2-7所示：

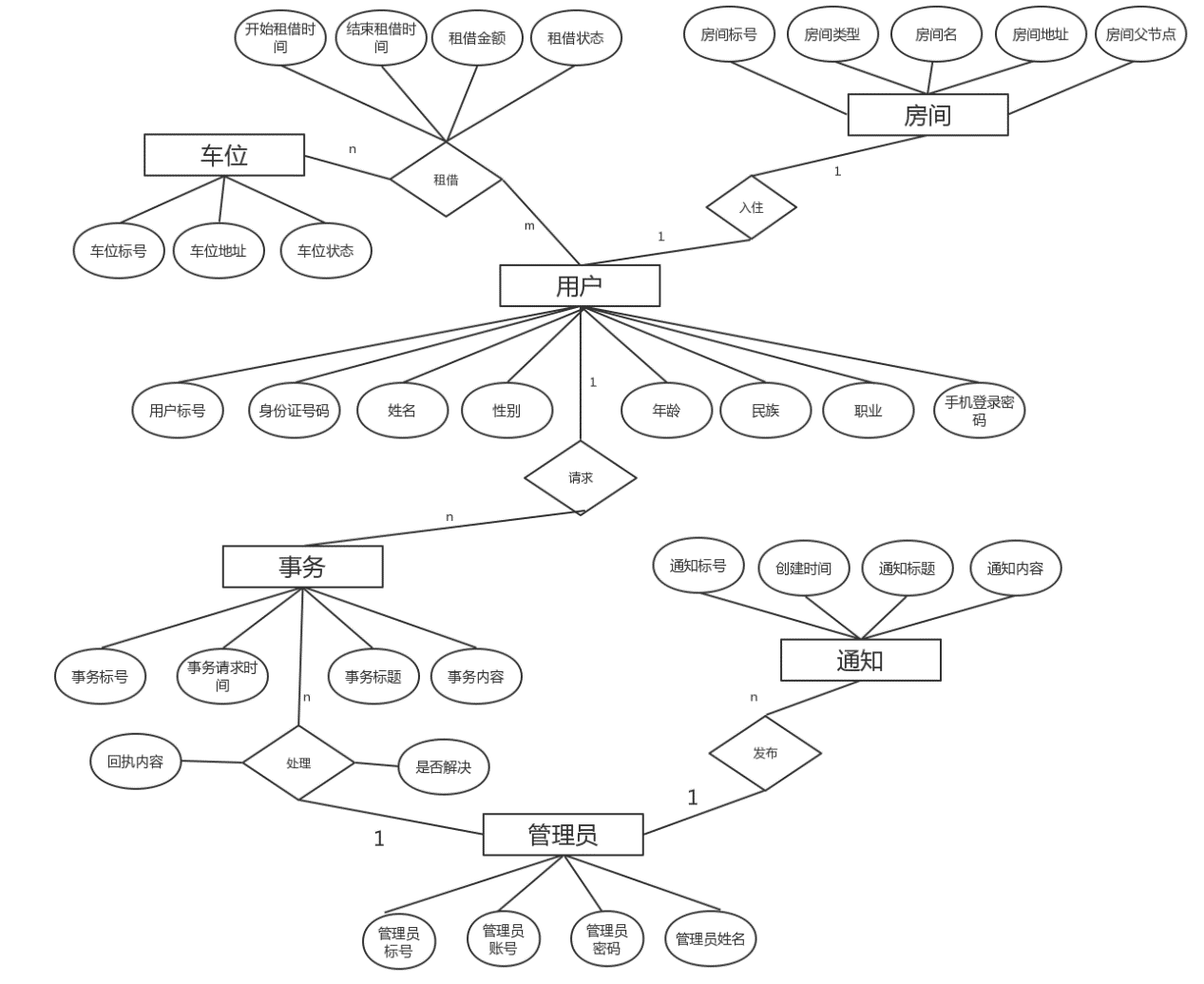


图2-7 总ER图

各实体属性如下：

●用户：{用户标号，身份证号码,姓名,性别,电话,年龄,民族,职业,手机登录密码}

●房间：{房间标号，房间地址，房间名，房间父节点，房间类型}

●通知：{通知标号，创建时间，通知标题，通知内容}

●车位：{车位标号,车位地址,车位状态}

●事务：{ 事务标号、事务请求时间、事务标题、事务内容}

●管理员：{ 管理员标号，管理员账号，管理员密码，管理员姓名 }

# 第三章 逻辑结构设计

3.1 将E-R图转换为关系模型

### 3.1.1原实体对应的关系模型如下：

●用户：{用户标号，身份证号码,姓名,性别,电话,年龄,民族,职业,手机登录密码}

●房间：{房间标号，房间地址，房间名，房间父节点，房间类型}

●通知：{通知标号，创建时间，通知标题，通知内容}

●车位：{车位标号,车位地址,车位状态}

●事务：{ 事务标号、事务请求时间、事务标题、事务内容}

●管理员：{ 管理员标号，管理员账号，管理员密码，管理员姓名 }

### 3.1.2关系模型规范化

1. **将各一对一联系和一对多联系都采用与实体合并的方式转化为关系模式**
2. 用户与入住

·用户：{用户标号，***房间标号***，身份证号码,姓名,性别,电话,年龄,民族,职业,手机登录密码}

③事务与请求与处理

·事务：{事务标号,***用户标号***, ***管理员标号***,事务请求时间,事务标题,事务内容,回执内容,是否解决}

④通知与发布

·通知：{通知标号,***管理员标号***,创建时间,通知标题,通知内容}

1. **将多对多联系转换为一个独立的关系模式**

租借：{***车位标号，用户标号，***开始租借时间，结束租借时间，租借金额，租借状态}

### 3.1.3处理后的所有关系模型

●用户：{用户标号，***房间标号***，身份证号码,姓名,性别,电话,年龄,民族,职业,手机登录密码}

●房间：{房间标号，房间地址，房间名，房间父节点，房间类型}

●通知：{通知标号,***管理员标号***,创建时间,通知标题,通知内容}

●车位：{车位标号,车位地址,车位状态}

●事务：{事务标号,***用户标号***, ***管理员标号***,事务请求时间,事务标题,事务内容,回执内容,是否解决}

●管理员：{ 管理员标号，管理员账号，管理员密码，管理员姓名 }

●租借：{***车位标号，用户标号，***开始租借时间，结束租借时间，租借金额，租借状态}

上述关系模式均规范到3NF

注：下划线为主键，斜体为外键

# 第四章 物理设计及数据库实施

4.1 基本表设计与创建

**1.**用户表的设计如图4-1、4-2所示：

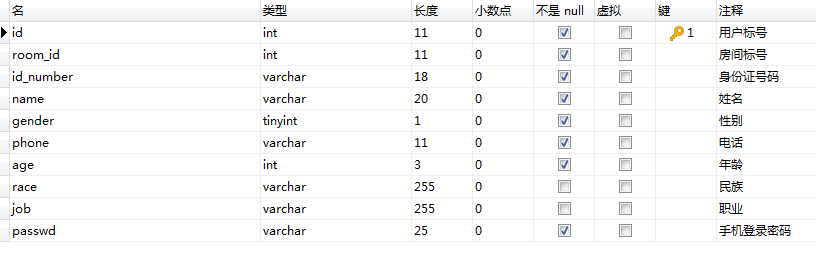


图4-1 用户表的设计

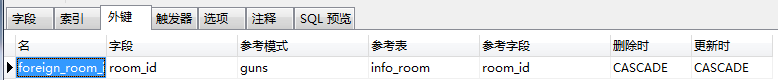


图4-2 用户表的外键设计

**2.**通知表的设计如图4-3、4-4所示：

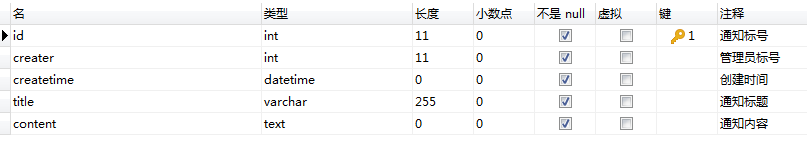


图4-3 通知表的设计

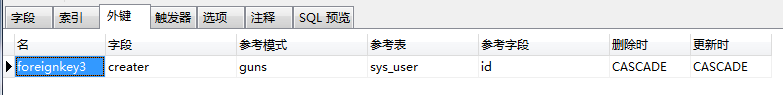


图4-4 通知表的外键设计

**3.**房间表的设计如图4-5所示：

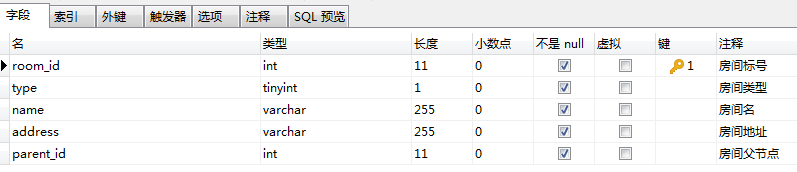


图4-5房间表的设计

**4.**车位表的设计如图4-6所示：



图4-6 车位表的设计

**5.**事务表的设计如图4-7、4-8所示：

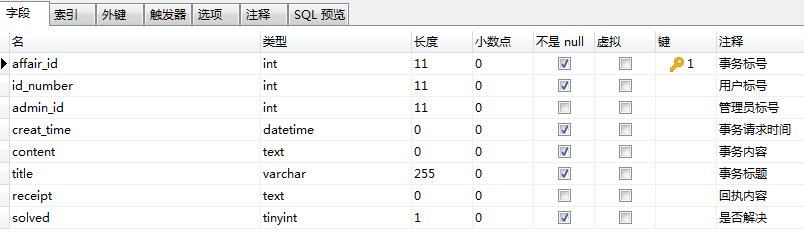


图4-7 事务表的设计

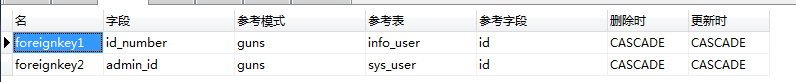


图4-8 事务表的外键设计

**6.**管理员表的设计如图4-9所示：



图4-9 管理员表的设计

**6.**租借表的设计如图4-10、4-11所示：

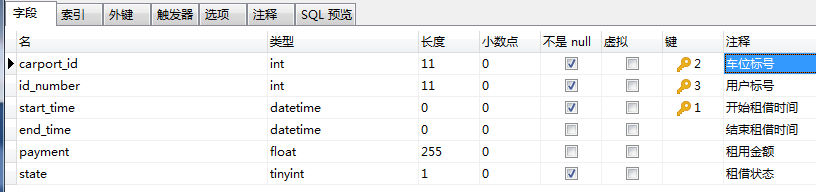


图4-10 租借表的设计

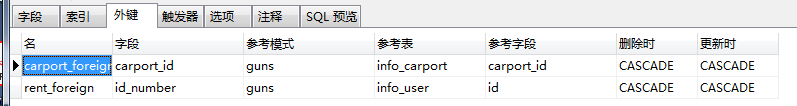


图4-11 租借表外键设计

4.2视图

由于租借车位的信息涉及租借表、车位表、用户表三个表，使用视图将所需要的字段联系到一起，便于查询操作。视图的设计与实现如图4-12、4-13、4-14所示：

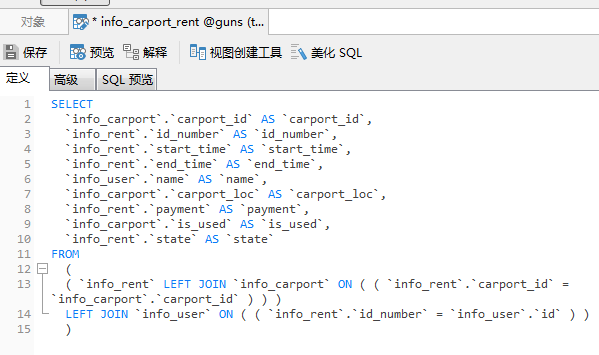


图4-12 视图的sql语言实现

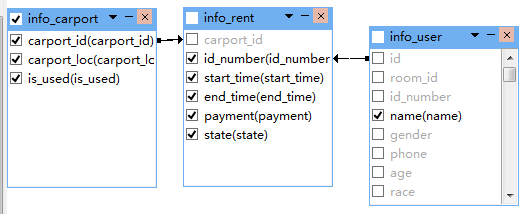


图4-13 视图中表间的关系

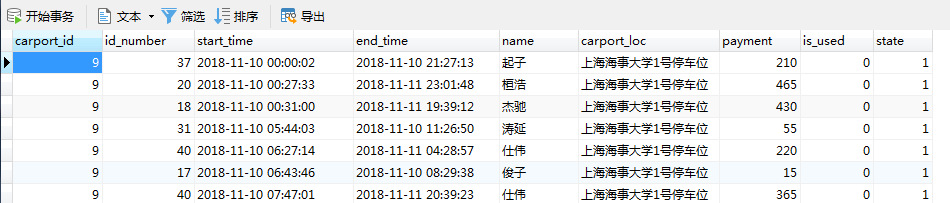


图4-14 视图的示例

4.3索引

为了加快数据库的存取速度，提升数据库的效率，我们在每一个外键上都建立了索引。并在主键上建立了唯一索引。用户表的索引如图4-15所示：

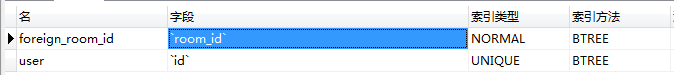
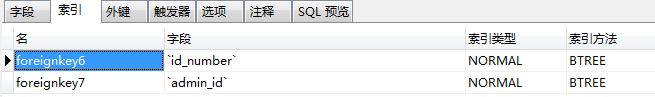


图4-15用户表的索引

事务表的索引如下图4-16所示：

图4-16事务表的索引

通知表的索引如下图4-17所示：

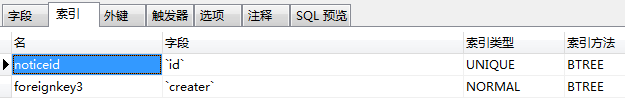


图4-17通知表的索引

租借表的索引如下图4-18所示：

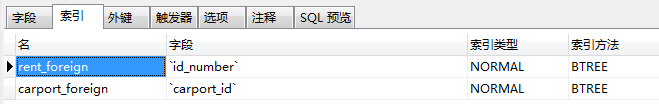


图4-18租借表的索引

4.5 触发器

为解决车位租借与结束车位租借时车位的状态，当租借表中新增租借信息时，将其租借的车位状态改为已被租借，当租借表中更改租借信息时，将其租借车位的状态改为未被租借。为了方便高效地实现此功能，我在租借表中设置了两个触发器。一个触发器在租借表插入数据时修改相应车位表的状态，另一个触发器在租借表修改数据时修改相应车位表的状态。触发器的实现如下图4-19、4-20所示：

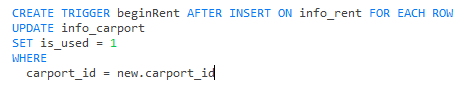


图4-19 租借表插入数据时的触发器

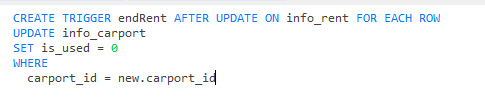


图4-20租借表修改数据时的触发器

4.6 数据库的关系图

各个数据表之间的关系如图4-20所示：

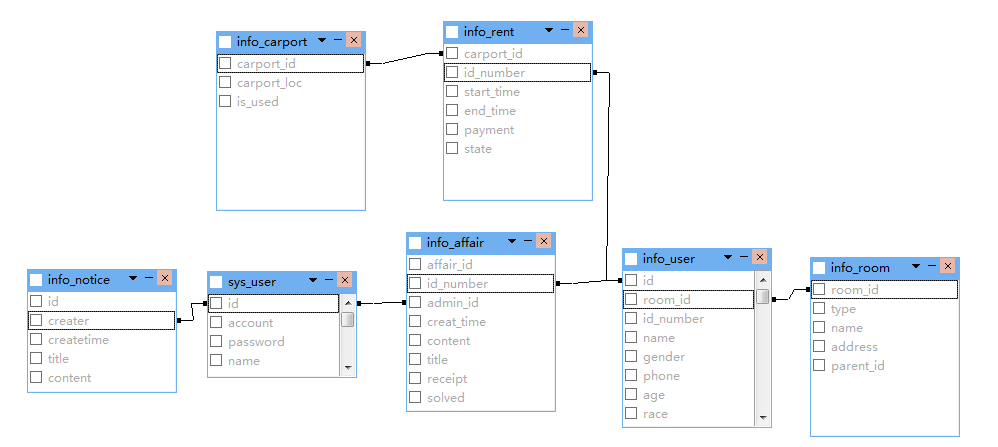


图4-20 各个数据表格之间

# 第五章 系统的设计与实现

## 5.1 系统框架简介

本系统基于guns开源框架。框架基于SpringBoot 2，整合springmvc + mybatis-plus + echarts。Spring Boot是一种JAVA框架，旨在尽可能快地启动和运行，只需最少的Spring前端配置。Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。MyBatis是一个支持普通SQL查询，存储过程和高级映射的优秀持久层框架。MyBatis消除了几乎所有的JDBC代码和参数的手工设置以及对结果集的检索封装。MyBatis可以使用简单的XML或注解用于配置和原始映射，将接口和Java的POJO（Plain Old Java Objects，普通的Java对象）映射成数据库中的记录。Echarts缩写来自Enterprise Charts，商业级数据图表，

它最初是为了满足公司商业体系里各种业务系统的报表需求。

关于系统架构采用的是MVC架构 ，它全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。其逻辑图如下图5-1所示：

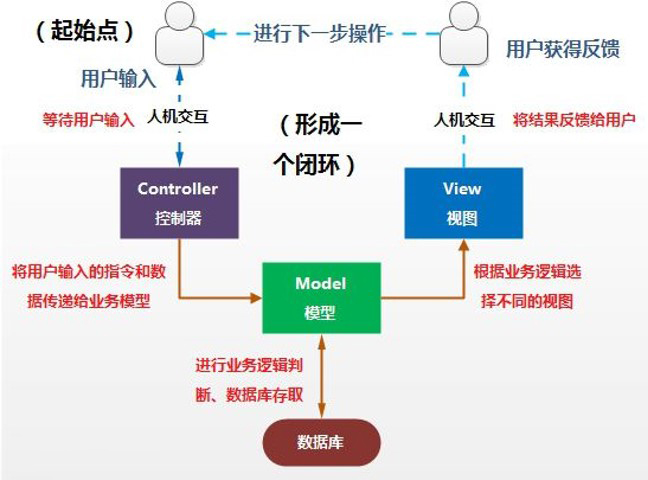


图5-1 MVC逻辑图

视图：将视图层存放在PeopleManage\src\main\webapp中，其子文件夹static\modular存放各个业务功能的JavaScript代码，另一个子文件夹WEB-INF\view 存放各个业务功能的HTML代码。

控制器：将控制器存放在PeopleManage\src\main\java\cn\stylefeng\guns\modular的各个业务功能的controler文件夹中。

模型：将模型存放在PeopleManage\src\main\java\cn\stylefeng\guns\modular\system\model中，各个模型通过其对应的Mapper.xml进行需要的sql查询，通过Mapper接口类来调取Mapper.xml，sql查询的结果。若模型的业务逻辑复杂，将多增加对应的Service类来组合形成更复杂的业务逻辑，使业务逻辑更清晰有条理。

## 5.2 业务的实现

本系统的增删查改模式基本都类似，只是有些具体的业务有不同的需求，系统会有相应的处理，总体框架是没有变的。现在以住房管理为例，详细说明本系统增删查改的逻辑。

### 5.2.1 信息列表的显示

首先用户看到的应该是住房管理的主页面，如图5-2所示：



图5-2住房管理主界面

本界面存储在PeopleManage\src\main\webapp\WEBINF\view\room\_info\info\info.html 其HTML代码如下：

<**div class="ibox float-e-margins"**>  
 <**div class="ibox-title"**>  
 <**h5**>住房管理</**h5**>  
 </**div**>  
 <**div class="ibox-content"**>  
 <**div class="row row-lg"**>  
 <**div class="col-sm-12"**>  
 <**div class="row"**>  
 <**div class="col-sm-3"**>  
 <#NameCon id="condition" name="名称" />  
 </**div**>  
 <**div class="col-sm-3"**>  
 <#button name="搜索" icon="fa-search" clickFun="Info.search()"/>  
 </**div**>  
 </**div**>  
 <**div class="hidden-xs" id="InfoTableToolbar" role="group"**>  
 @if(shiro.hasPermission("/info/add")){  
 <#button name="添加" icon="fa-plus" clickFun="Info.openAddInfo()"/>  
 @}  
 @if(shiro.hasPermission("/info/update")){  
 <#button name="修改" icon="fa-edit" clickFun="Info.openInfoDetail()" space="true"/>  
 @}  
 @if(shiro.hasPermission("/info/delete")){  
 <#button name="删除" icon="fa-remove" clickFun="Info.delete()" space="true"/>  
 @}  
 </**div**>  
 <#table id="InfoTable"/>  
 </**div**>  
 </**div**>  
 </**div**>

表格显示的位置在<#table id="InfoTable"/>，表格由其对应的JavaScript向Controller发出请求获得数据。控制住房管理的JavaScript存放在PeopleManage\src\main\webapp\static\modular\room\_info\info\info.js中。

其代码如下：

*/\*\*  
 \* 住房管理初始化  
 \*/***var *Info*** = {  
 **id**: **"InfoTable"**, *//表格id* **seItem**: **null**, *//选中的条目* **table**: **null**,  
 **layerIndex**: -1  
};  
  
*/\*\*  
 \* 初始化表格的列  
 \*/****Info***.initColumn = **function** () {  
 **return** [  
 {**field**: **'selectItem'**, **radio**: **true**},  
 {**title**: **'区域标识号'**, **field**: **'room\_id'**, **visible**: **true**, **align**: **'center'**, **valign**: **'middle'**,**width**:**"100px"**},  
 {**title**: **'名称'**, **field**: **'name'**, **visible**: **true**, **align**: **'center'**, **valign**: **'middle'**},  
 {**title**: **'地址'**, **field**: **'address'**, **visible**: **true**, **align**: **'center'**, **valign**: **'middle'**},  
 {**title**: **'类型'**, **field**: **'type'**, **visible**: **true**, **align**: **'center'**, **valign**: **'middle'**,  
 formatter: **function**(value,row,index){  
 **if**(row.**type**==**true**)  
 {  
 value=**'<div style="text-align: center;" ><i class="fa fa-bed"></i>&nbsp;房间</div>'**;  
 }  
 **else** value=**'<div style="text-align: center;" >区域</div>'**;  
 **return** value ;  
 }},  
 {**title**: **'父节点'**, **field**: **'parent\_id'**, **visible**: **false**, **align**: **'center'**, **valign**: **'middle'**}  
 ];  
};  
**$**(**function** () {  
 **var** defaultColunms = ***Info***.initColumn();  
 **var** table = **new BSTreeTable**(***Info***.**id**, **"/info/list"**, defaultColunms);  
 table.setExpandColumn(2);  
 table.setIdField(**"room\_id"**);  
 table.setCodeField(**"room\_id"**);  
 table.setParentCodeField(**"parent\_id"**);  
 table.setExpandAll(**true**);  
 table.init();  
 ***Info***.**table** = table;  
});

JavaScript页面中的表格控制工具BSTreeTable向Controler发出请求，请求地址为info/list,Controler接收到请求后返回所需的信息，BSTreeTable根据后端返回的信息对后端返回的信息进行处理，table.init初始化函数对返回的数据进行处理，formatter函数对房间类型进行处理，当房间类型为1时显示为房间，当房间类型为0时显示为区域。

房间管理的控制器位置在PeopleManage\src\main\java\cn\stylefeng\guns\modular\room\_info\controller\InfoController.java

，控制器收到/info/list请求地址后执行list函数，list函数调用infoService.list方法后返回房间管理的列表，再使用RoomWarpper处理数据，最后返回给BSTreeTable显示到前端页面中。控制器代码如下：

*/\*\*  
 \* 住房管理控制器  
 \** ***@author*** *lichuang  
 \** ***@Date*** *2018-11-10 14:57:27  
 \*/*@Controller  
@RequestMapping(**"/info"**)  
**public class** InfoController **extends** BaseController {  
  
 **private** String **PREFIX** = **"/room\_info/info/"**;  
  
 */\*\*  
 \* 获取住房管理列表  
 \*/* @RequestMapping(value = **"/list"**)  
 @ResponseBody  
 **public** Object list(String condition) {  
  
 List<Map<String, Object>> list = **this**.**infoService**.list(condition);  
 **return super**.warpObject(**new** RoomWarpper(list));  
 }  
}

infoService中的list方法，在InfoMapper.xml中实现，其代码如下所示：

<**select id="list" resultType="map"**>  
 select \* from info\_room  
 <**if test="condition != null and condition != ''"**>  
 where address like CONCAT('%',#{condition},'%')  
 </**if**>  
</**select**>

经过上述各个步骤，数据才能从数据库中取出来，放到前端界面进行展示。

### 5.2.2 新增信息

在info.html页面点击添加按钮，info.javascrpit会向控制器发送一个添加请求，请求地址为/info/info\_add，其代码如下所示：

***Info***.openAddInfo = **function** () {  
 **var** index = **layer**.open({  
 **type**: 2,  
 **title**: **'添加住房'**,  
 **area**: [**'800px'**, **'420px'**], *//宽高* **fix**: **false**, *//不固定* **maxmin**: **true**,  
 **content**: ***Feng***.**ctxPath** + **'/info/info\_add'** });  
 **this**.**layerIndex** = index;  
};

当InfoController接收到/info/info\_add请求后会返会info\_add.html页面，其代码如下所示：

@RequestMapping(**"/info\_add"**)  
**public** String infoAdd() {  
 **return PREFIX** + **"info\_add.html"**;  
}

info\_add.html页面如图5-3所示：

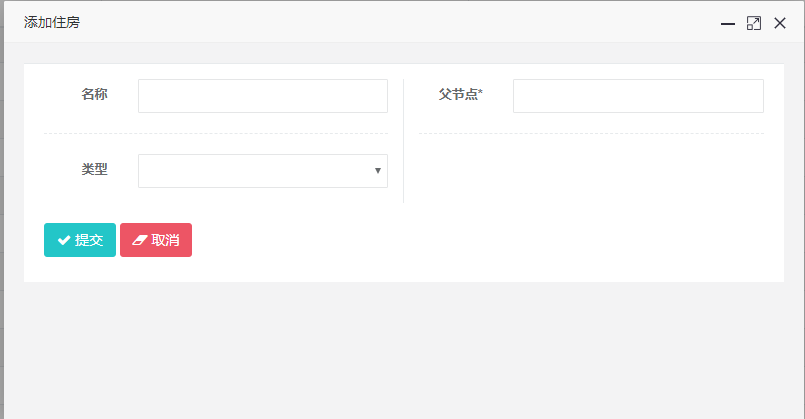


图5-3新增住房界面

控制此页面的JavaScript是info\_info.javascript,其主要代码如下所示：

*/\*\*  
 \* 初始化住房管理详情对话框  
 \*/***var *InfoInfoDlg*** = {  
 **infoInfoData** : {}  
};  
  
*/\*\*  
 \* 收集数据  
 \*/****InfoInfoDlg***.collectData = **function**() {  
 **this** .set(**'roomId'**)  
 .set(**'type'**)  
 .set(**'name'**)  
 .set(**'parentId'**)  
 .set(**'address'**)  
}  
  
*/\*\*  
 \* 提交添加  
 \*/****InfoInfoDlg***.addSubmit = **function**() {  
  
 **this**.clearData();  
 **this**.collectData();  
  
 *//提交信息* **var** ajax = **new $ax**(***Feng***.**ctxPath** + **"/info/add"**, **function**(data){  
 ***Feng***.success(**"添加成功!"**);  
 ***window***.**parent**.***Info***.**table**.refresh();  
 ***InfoInfoDlg***.close();  
 },**function**(data){  
 ***Feng***.error(**"添加失败!"** + data.responseJSON.**message** + **"!"**);  
 });  
 ajax.set(**this**.**infoInfoData**);  
 ajax.start();  
}  
 当用户填完所有信息，点击提交按钮时，会向后端送请求，请求地址为info/add，参数为填写的信息。

当InfoControler接收到此信息时会调用add函数，add函数会判断其是不是顶层区域，如果是顶层区域，此房间的地址就是房间名，如果不是顶层区域，此房间的地址就是上一层房间的地址再加上此房间名。然后使用mybatis简化的插入方法，infoService.insert(info)函数将信息插入到数据库中。其实现代码如下所示：@RequestMapping(value = **"/add"**)  
@ResponseBody  
**public** Object add(Info info) {  
 **if**(info.getParentId()!=0)  
 info.setAddress(**infoService**.selectById(info.getParentId()).getAddress()+info.getName());  
 **else** info.setAddress(info.getName());  
 **infoService**.insert(info);  
 **return** *SUCCESS\_TIP*;  
}

### 5.2.3 修改信息

在info.html选中一条记录，再在页面点击修改按钮，info.javascrpit会找到选择记录的id，然后向Controller发送修改请求，请求地址info/info\_update,当请求完成后打开新的子页面，如图其代码如下所示：***Info***.openInfoDetail = **function** () {  
 **if** (**this**.check()) {  
 **var** index = **layer**.open({  
 **type**: 2,  
 **title**: **'住房详情'**,  
 **area**: [**'800px'**, **'420px'**], *//宽高* **fix**: **false**, *//不固定* **maxmin**: **true**,  
 **content**: ***Feng***.**ctxPath** + **'/info/info\_update/'** + ***Info***.**seItem**.**id** });  
 **this**.**layerIndex** = index;  
 }  
};

当控制器接收到修改请求后，先根据选中条目id获得完整的房间信息，然后将信息注入到info\_edit.html页面中，然后再返回info\_edit.html页面。其代码实现如下所示：

@RequestMapping(**"/info\_update/{infoId}"**)  
**public** String infoUpdate(@PathVariable Integer infoId, Model model) {  
 Info info = **infoService**.selectById(infoId);  
 model.addAttribute(**"item"**,info);  
 **if**(info.getParentId()!=0)  
 model.addAttribute(**"pName"**, **infoService**.selectById(info.getParentId()).getName());  
 **else** model.addAttribute(**"pName"**,**"顶级"**);  
 LogObjectHolder.*me*().set(info);  
 **return PREFIX** + **"info\_edit.html"**;  
}

info\_edit.html界面如下图5-4所示：

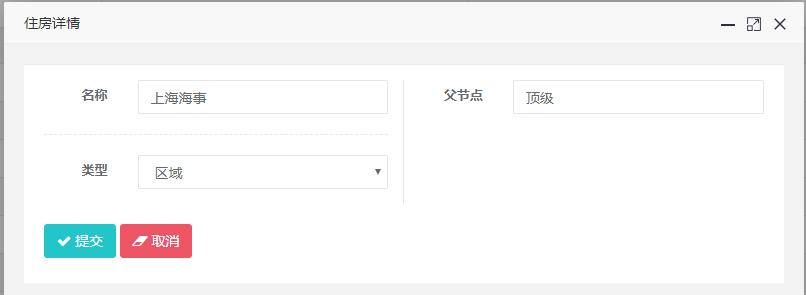


图5-4住房详情界面

当修改成功后，info\_info.html会收集用户更改的信息，然后向控制器发送更改请求，请求地址为/info/update其代码如下：

***InfoInfoDlg***.editSubmit = **function**() {  
  
 **this**.clearData();  
 **this**.collectData();  
  
 *//提交信息* **var** ajax = **new $ax**(***Feng***.**ctxPath** + **"/info/update"**, **function**(data){  
 ***Feng***.success(**"修改成功!"**);  
 ***window***.**parent**.***Info***.**table**.refresh();  
 ***InfoInfoDlg***.close();  
 },**function**(data){  
 ***Feng***.error(**"修改失败!"** + data.responseJSON.**message** + **"!"**);  
 });  
 ajax.set(**this**.**infoInfoData**);  
 ajax.start();  
}

当控制器接收到修改请求后先找出原来的房间信息，然后查看房间名有没有被修改过，如果房间名被修改了，那此房间所有的子节点的地址都要修改，这样才能保持地址的正确性，然后使用mybatis提供的更新方法，将新数据更新到数据库中。

@RequestMapping(value = **"/update"**)  
@ResponseBody  
**public** Object update(Info info) {  
 Info oldinfo=**infoService**.selectById(info.getRoomId());  
 String name=info.getName();  
 String oldaddress=oldinfo.getAddress();  
 String newaddress;  
 **if**(info.getParentId()==0)newaddress=name;  
 **else** newaddress=oldaddress.substring(0,oldaddress.length()-oldinfo.getName().length())+name;  
 info.setAddress(newaddress);  
 **infoService**.updateById(info);  
 *ChangeChildAddress*(newaddress,oldaddress,info.getRoomId());  
 **return** *SUCCESS\_TIP*;  
}

### 5.2.4 删除信息

用户选择某一条记录，然后点击删除按钮，前端会发送一个删除请求到控制器，请求地址为/info/delete其实现代码如下所示：

***Info***.delete = **function** () {  
 **if** (**this**.check()) {  
 **var** ajax = **new $ax**(***Feng***.**ctxPath** + **"/info/delete"**, **function** (data) {  
 ***Feng***.success(**"删除成功!"**);  
 ***Info***.**table**.refresh();  
 }, **function** (data) {  
 ***Feng***.error(**"删除失败!"** + data.responseJSON.**message** + **"!"**);  
 });  
 ajax.set(**"infoId"**,**this**.**seItem**.**id**);  
 ajax.start();  
 }  
};

当控制器接收到删除请求时，先通过TreeTool的removeTreeNode方法找出本房间的所有子节点，方法返回所有子节点的id与自己的id，然后通过返回的所有id删除房间信息。

@RequestMapping(value = **"/delete"**)  
@ResponseBody  
**public** Object delete(@RequestParam Integer infoId) {  
 List<Integer> list= TreeTool.*removeTreeNode*(**infoService**.list(**null**),infoId);  
 **for**(Integer i :list)  
 **infoService**.deleteById(i);  
 **return** *SUCCESS\_TIP*;  
}

## 5.3 数据统计功能的实现

### 5.3.1 男女比例与年龄分布

本系统统计了小区人员的性别信息，与年龄信息，如下图5-5所示：

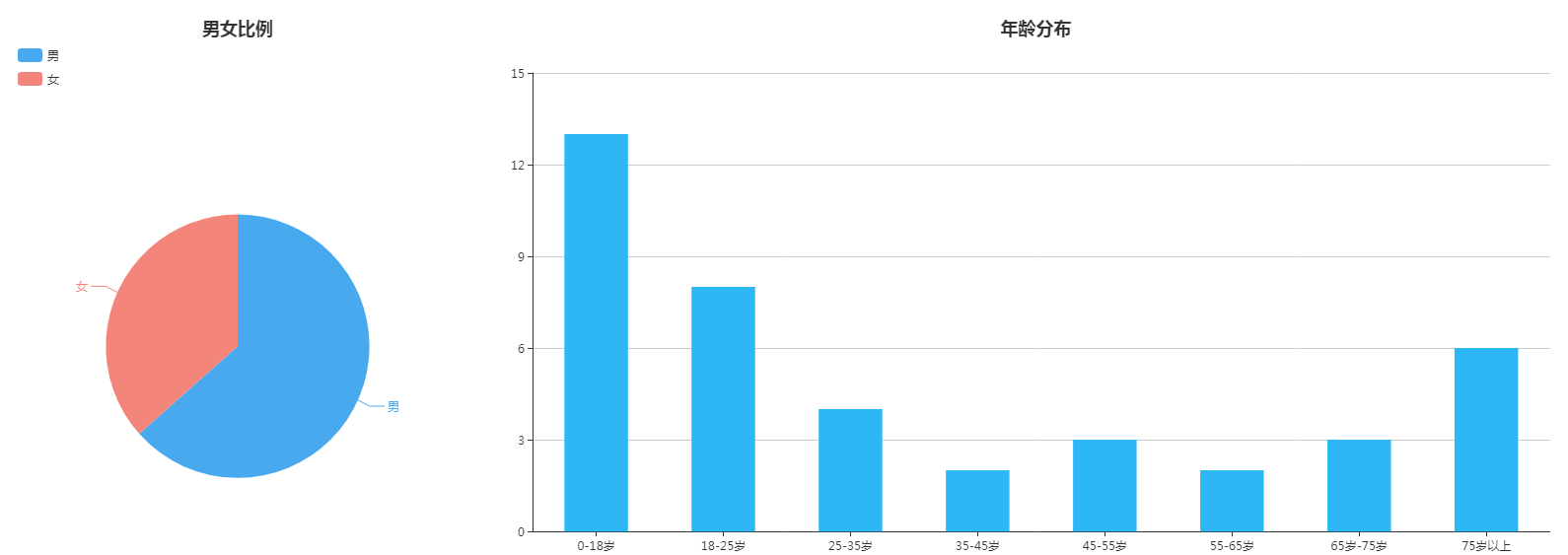


图5-5男女比例与年龄分布的统计

图表使用的是百度Echart插件，前端向控制器发送获取性别图表的请求，控制器操作业务模型查询男女人数的信息，查询完成后再返回给前端的图表，前端将信息插入图表中显示出来，初始化男女比例图表的代码如下所示：

***InitChart***.initSexChartOrigin=**function** () {  
 **option** = {  
 **grid**:{  
 **top**:**"50px"**,  
 **left**:**"50px"**,  
 **right**:**"15px"**,  
 **bottom**:**"50px"** },  
 **color**: [**'#49A9EE'**,**'#F3857B'**],  
 **title** : {  
 **text**: **'男女比例'**,  
 **x**:**'center'** },  
 **tooltip** : {  
 **trigger**: **'item'**,  
 **formatter**: **"{a} <br/>{b} : {c} ({d}%)"** },  
 **legend**: {  
 **orient**: **' horizontal'**,  
 **x**:15,  
 **y**:30,  
 **data**:[**'男'**,**'女'**]  
 },  
 **series**: [  
 {  
 **name**:**'男女比例'**,  
 **type**:**'pie'**,  
 **radius** : **'55%'**,  
 **center**: [**'50%'**, **'60%'**],  
 **data**:[  
  
 ],  
 **itemStyle**: {  
 **emphasis**: {  
 **shadowBlur**: 10,  
 **shadowOffsetX**: 0,  
 **shadowColor**: **'rgba(0, 0, 0, 0.5)'** }  
 }  
 }  
 ]  
 };  
 **return option**;  
}

初始化年龄分布的图表代码如下所示：

***InitChart***.initAgeChartOrigin=**function** () {  
 **option** = {  
 **color**: [**'#2DB7F5'**],  
 **tooltip** : {  
 **trigger**: **'axis'**,  
 **axisPointer** : { *// 坐标轴指示器，坐标轴触发有效* **type** : **'shadow'** *// 默认为直线，可选为：'line' | 'shadow'* }  
 },**title** : {  
 **text**: **'年龄分布'**,  
 **x**:**'center'** },  
 **grid**: {  
 **left**: **'3%'**,  
 **right**: **'4%'**,  
 **bottom**: **'3%'**,  
 **containLabel**: **true** },  
 **xAxis** : [  
 {  
 **type** : **'category'**,  
 **data** : [**'0-18岁'**, **'18-25岁'**, **'25-35岁'**, **'35-45岁'**, **'45-55岁'**,**'55-65岁'**,**'65岁-75岁'**,**'75岁以上'**],  
 **axisTick**: {  
 **alignWithLabel**: **true** }  
 }  
 ],  
 **yAxis** : [  
 {  
 **type** : **'value'** }  
 ],  
 **series** : [  
 {  
 **name**:**'人数'**,  
 **type**:**'bar'**,  
 **barWidth**: **'50%'**,  
 }  
 ]  
 };  
 **return option**;  
}

发送获取图表信息请求，并将请求完成获得到的数据插入到图表中的代码如下所示：

***Data***.getSexData = **function**() {  
 **$**.get(**'/data\_statistics/getAgeAndSEX/?type=sex'**).done(**function** (sexinfo) {  
 *// 填入数据* ***Data***.**sexchart**.hideLoading();  
 ***Data***.**sexchart**.setOption({  
 **series**: [  
 {  
 **data**:[  
 {**value**:sexinfo[**"男"**]==0?**null**:sexinfo[**"男"**], **name**:**'男'**},  
 {**value**:sexinfo[**"女"**]==0?**null**:sexinfo[**"女"**], **name**:**'女'**},  
 ]  
  
 }  
 ]  
 });  
 });  
};

***Data***.getAgeData = **function**() {  
 **$**.get(**'/data\_statistics/getAgeAndSEX/?type=age'**).done(**function** (ageinfo) {  
 *// 填入数据* ***Data***.**agechart**.hideLoading();  
 ***Data***.**agechart**.setOption({  
 **series**: [  
 {  
 **data**:[ageinfo[**"age1"**],  
 ageinfo[**"age2"**],  
 ageinfo[**"age3"**],  
 ageinfo[**"age4"**],  
 ageinfo[**"age5"**],  
 ageinfo[**"age6"**],  
 ageinfo[**"age7"**],  
 ageinfo[**"age8"**],  
 ]  
 }  
 ]  
 });  
 });  
}

其请求的控制器代码如下所示：@RequestMapping(value = **"/getAgeAndSEX"**)  
@ResponseBody  
**public** Object getSexCharts (@RequestParam Map<String, Object> paramMap ) {  
 Map<String, Object> charts = **this**.**infoUserService**.getSexAndAgeCharts(paramMap);  
 **return** charts;  
}

根据传递的参数，选择不同的sql语言进行查询。若参数为age，则分别统计各个年龄的人数；若参数为sex，则分别统计男女的人数。其代码如下所示：

<**select id="getSexAndAgeCharts" resultType="map"**>  
 <**if test="type != null and type != ''and type=='age'"**>  
 select sum(case when age **&lt;**=16 then 1 else 0 end ) age1,  
 sum(case when age>16 and age **&lt;**=25 then 1 else 0 end ) age2,  
 sum(case when age>25 and age **&lt;**=35 then 1 else 0 end ) age3,  
 sum(case when age>35 and age **&lt;**=45 then 1 else 0 end ) age4,  
 sum(case when age>45 and age **&lt;**=55 then 1 else 0 end ) age5,  
 sum(case when age>55 and age **&lt;**=65 then 1 else 0 end ) age6,  
 sum(case when age>65 and age **&lt;**=75 then 1 else 0 end ) age7,  
 sum(case when age>75 then 1 else 0 end ) age8  
 </**if**>  
 <**if test="type != null and type != ''and type=='sex'"**>  
 select sum(case when gender=1 then 1 else 0 end ) 男,  
 sum(case when gender=0 then 1 else 0 end ) 女  
 </**if**>  
 from info\_user  
</**select**>

### 5.3.2 车位租借次数与租借金额

本系统可以选择时间来统计车位租借的次数与车位租借的金额，其实现如下图5-6、5-7所示：

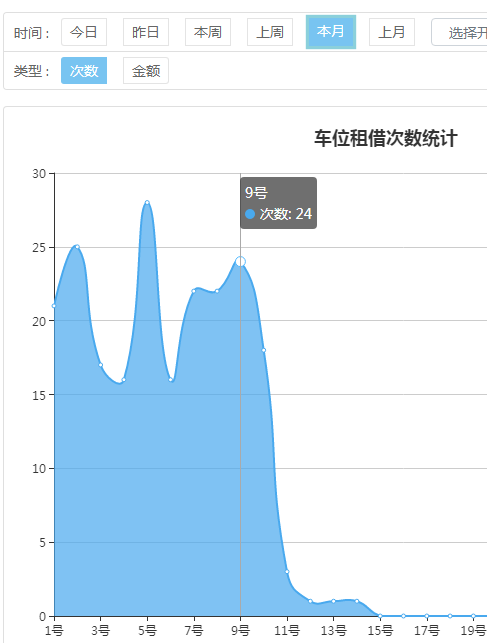


图5-6车位租借次数的统计

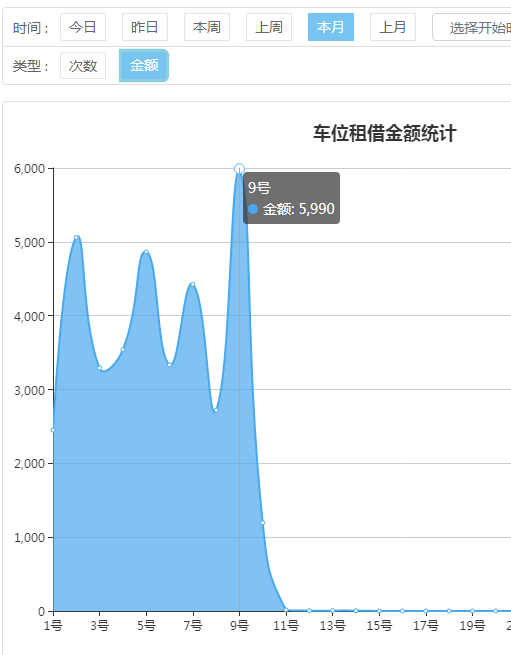


图5-7车位租借金额的统计

获得车位信息时，主要传递两个参数，一个参数是选择的时间request\_val,可选项有今日，本周等；另一个参数是选择的类型selectType，可选项有次数与金额。根据不同的参数显示不同的标题，形成对应的下标。若是本周与上周，下标为周一到周日；若是今日或明日，下标为24个小时；若是本月或上月下标为1号到31号。其按周查询的实现的代码如下所示：

***Data***.getPeopleData = **function**() {  
**$**.get(**'/data\_statistics/people/?condition='**+***Data***.**request\_val**+**'&type='**+***Data***.**selectType**).done(**function** (peopleinfo) {  
 **if**(***Data***.**request\_val**==**"thisweek"**||***Data***.**request\_val**==**"lastweek"**)  
 {  
 **if**(***Data***.**selectType**==**"num"**){title=**"车位租借次数统计"**;name=**"次数"**}  
 **if**(***Data***.**selectType**==**"money"**){title=**"车位租借金额统计"**;name=**"金额"**}  
 data\_arry=***DataFormat***.ParaseWeek(peopleinfo);  
 option={  
 **title** : {  
 **text**: title,  
 **x**:**'center'** },  
 **xAxis**: {  
 **data** : [**'周一'**,**'周二'**,**'周三'**,**'周四'**,**'周五'**,**'周六'**,**'周日'**]  
 },  
 **series**: [  
 {  
 **name**:name,  
 **data**:data\_arry,  
 }]  
 }  
 }  
 ***Data***.**peoplechart**.hideLoading();  
 ***Data***.**peoplechart**.setOption(option);  
 });  
}

其请求的控制器代码如下：

@RequestMapping(value = **"/people"**)  
@ResponseBody  
**public** Object selectRentChart (@RequestParam Map<String, Object> paramMap) {  
 List<Map<String, Object>> charts = **this**.**infoRentService**.selectRentChart(paramMap);  
 **return** charts;  
}

数据库查询的操作以上周为例，将租借视图的租借结束时间按周分类。由于Mysql中每周的第一天为周日，所以要在记录的日期上减去一天，成为我们习惯的周一为每周的第一天，之后格式化成某一周。然后判断它的周数是否等于上周。查询记录的时间为上周，并将时间格式化为星期的格式，其查询结果如下图5-8所示：

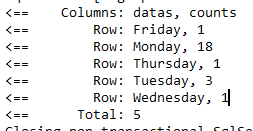


图5-8按上周统计的示例

其代码实现如下所示：

<**select id="selectRentChart" resultType="map"**>  
 <**if test="condition != null and condition != ''and condition=='lastweek'"**>  
 select DATE\_FORMAT(info\_carport\_rent.end\_time,'%W') datas,  
 <**if test="type == 'num' "**>  
 *count*(*\**)  
 </**if**>  
 <**if test="type == 'money' "**>  
 sum(payment)  
 </**if**>as counts  
 FROM info\_carport\_rent  
 WHERE YEARWEEK(date\_format(info\_carport\_rent.end\_time,'%Y-%m-%d')- INTERVAL 1 DAY) = YEARWEEK(now()- INTERVAL 7 DAY)  
 and state=1  
 </**if**>  
 GROUP BY datas  
 ORDER BY datas;  
</**select**>

## 5.4 API接口的实现

### 5.4.1 接口的安全性

由于手机端的需要，系统一共提供了九个业务接口，包括新增人员、查询密码是否正确、查看发起的投诉或报修、查看剩余车位数量、获得房间信息、获得最新一条公告、发起投诉或报修、查看剩余车位数量、租用车位、结束租借。这些接口为手机端进行业务操作提供了便利性。但因为HTTP协议是开放的，可以任人调用。所以，如果接口不希望被随意调用，就需要做访问权限的控制，认证是好的用户，才允许调用API。为了增强安全性，我们使用了Bearer Token对用户身份进行认证。为了验证使用者的身份，需要客户端向服务器端提供一个可靠的验证信息，称为Token，这个token通常由Json数据格式组成，通过hash散列算法生成一个字符串，所以称为Json Web Token。只有用户获得了Token之后，在HTTP请求的头部加入token，才能正常地进行业务操作。

### 5.4.2 主要接口的实现

当用户想要租用车位时，需要将用户id发送给接口，系统查询车位是否还有空闲，若无空闲返回无空闲车位并退出。当还有剩余车位时，系统根据用户id查询用户是否正在租用车位，若正在租用车位，则返回错误信息并退出。若用户未租用车位，则系统随机分配一个车位给用户，并将用户id，车位标号，租用时间存入租用表，当租用表存入数据时，租用表的插入触发器，将对应的车位状态改为已被租用。租用成功后，系统将车位地址与租用时间返回给用户。租用车位的接口代码如下所示：

**public** Object rentCarport(@RequestParam(value=**"userId"**) Integer userId) {  
 **if**(ToolUtil.*isNotEmpty*(userId))  
 {  
 **try** {  
 **if** ((Long) **infoCarportMapper**.getCarportNum().get(**"CarportNum"**)!=0)  
 {  
 **if**(**infoRentMapper**.selectRentByIdNum(userId).size()!=0)  
 **return new** ErrorResponseData(500, **"您已租用了一个车位，不能在租用其他车位"**);  
 InfoCarport infoCarport=**infoCarportMapper**.rentCarport();  
 InfoRent infoRent=**new** InfoRent();  
 **if**(infoCarport!=**null**)  
 {infoRent.setCarportId(infoCarport.getCarportId());  
 infoRent.setIdNumber(userId);  
 infoRent.setStartTime(**new** Date());  
 infoRent.insert();  
 Map<String,Object> map=**new** HashMap<>();  
 map.put(**"carport\_loc"**,infoCarport.getCarportLoc());  
 map.put(**"createtime"**,infoRent.getStartTime());  
 **return** map;  
 }  
 **else return new** ErrorResponseData(500, **"车位已用完"**);  
 }  
 **else return new** ErrorResponseData(500, **"车位已用完"**);  
  
 }  
 **catch**(Exception e)  
 {  
 e.printStackTrace();  
 **return new** ErrorResponseData(500, **"租用车位错误"**);  
 }  
 }  
 **else return new** ErrorResponseData(500, **"请检查参数是否为空"**);  
 }

当用户想要结束租借时，只需将用户id传给结束租借接口，接口通过数据库查询相应的用户id是否有未完成的租借信息，若没有返回错误信息。若存在未完成的租借信息，系统获得当前时间，将当前时间与租借时间相减获得两个时间相差的毫秒数，再将这个毫秒数转化为分钟。然后进行收费，每30分钟5元钱，不到30分钟也按5元钱算。之后将租借结束时间、租借状态、与租借金额更新到租借表。租借表的更新触发器会将相应的车位状态更改为未被租用。完成所有操作后将租借总时长与租借金额返回给用户。其实现代码如下所示：

**public** Object stopRent(@RequestParam(value=**"userId"**) Integer userId) {  
 **if**(ToolUtil.*isNotEmpty*(userId))  
 {  
 **try** {  
 Map<String,Object> map=**new** HashMap<>();  
 map.put(**"id\_number"**,userId);  
 map.put(**"state"**,0);  
 List<InfoRent> infoRents=**infoRentMapper**.selectByMap(map);  
 **if**(infoRents.size()==1)  
 {  
 InfoRent infoRent=infoRents.get(0);Date start=infoRent.getStartTime();  
 Date end=**new** Date();  
 **long** interval=(end.getTime()-start.getTime())/1000/60;  
 **float** payment;  
 **if**((interval/30)>0) payment=(interval/30)\*5;  
 **else** payment=5;  
 **infoRentMapper**.stopRent(payment,start,end,infoRent.getCarportId(),1);  
 Map<String,Object> result=**new** HashMap<>();  
 result.put(**"rentTime"**,interval);  
 result.put(**"payment"**,payment);  
 **return** result;  
 }  
 **else return new** ErrorResponseData(500, **"不存在该用户的租借订单或有多个订单"**);  
 }  
 **catch**(Exception e)  
 {  
 e.printStackTrace();  
 **return new** ErrorResponseData(500, **"结束租借报修错误"**);  
 }  
 }  
 **else return new** ErrorResponseData(500, **"请检查参数是否为空"**);  
 }

# 第六章 结论和展望

在这次的课程设计中，我花费了很多的心思与精力。只要努力就有收获，这个成品我还是比较满意的。在这个过程中，我不但对JAVA Springboot2框架和数据库有了深刻的认识和了解，同时还学会了自学与独立思考的能力。此外，在我所负责的领域中，还有几个地方考虑不足：

1. 报修与投诉管理员回复一次默认已经解决了问题，但是实际情况可能需要管理员多次与住户进行交流才能够完全解决问题。
2. 租借流程中缺少管理员对租借的确认，如果系统分配的车位因为一些不可预知的原因不能租借，管理员不能及时反馈给用户。
3. 通知信息无法插入图片。
4. 插入信息没有控制其格式，可能会导致插入失败。

我会根据上述说的不足加以改进，使系统更人性化。

# 参考文献

[1]王珊,数据库系统概论[M].高等教育出版社, 2014.

[2]埃克尔,JAVA编程思想(第4版)[M].机械工业出版社, 2007.

[3]杨开振,深入浅出Spring Boot 2.x[M].人民邮电出版社, 2019.

[4]朱要光,Spring MVC+MyBatis开发从入门到项目实战[M].电子工业出版社, 2018.

[5]Stylefeng,Guns技术文档.2018-12-05, https://github.com/stylefeng/Guns.